UNIVERSITY OF SOUTHAMPTON

Analysis and Applications of Deep Cascade Learning

by

Xin Du

A thesis submitted in for the
degree of Doctor of Philosophy

in the
Faculty of Engineering and Physical Science
Department of Electronics and Computer Science

Friday 11th March, 2022

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCE
DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Xin Du

This dissertation is on the analysis and applications of a constructive architecture for training Deep Neural Networks, which are usually trained by End-to-End gradient propagation with fixed depths. End-to-End training of Deep Neural Networks has proven to offer impressive performances in a number of applications such as computer vision, machine translation and in playing complex games such as GO. Cascade Learning, the approach of interest here, trains networks in a layer-wise fashion and has been demonstrated to achieve satisfactory performance in large scale tasks such as the popular ImageNet benchmark dataset, at substantially reduced computing and memory requirements. Here we focus on the nature of features extracted from Cascade Learning. By attempting to explain the process of learning using Tishby et al.s' Information Bottleneck theory, we derive an empirical rule (Information Transition Ratio) to automatically determine a satisfactory depth for Deep Neural Networks. We suggest that Cascade Learning packs information in a hierarchical manner, with coarse features in early layers and more task-specific features in later layers. This is verified by considering Transfer Learning whereby features learned from a data-rich source domain assist in learning a data-sparse target domain. Using a wide range of inference problems in medical imaging, human activity recognition and inference from single cell gene expression between mice and humans, we demonstrate that Transfer Learning from a cascade trained model outperforms results noted by previous authors. An exception to this is the single cell gene expression problem where a single hidden layer network happens to be an adequate solution.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ANN Artificial Neural Network

AI Artificial Intelligence

ADL Activities of Daily Living

AC Auxiliary Classifier

BGD Batch Gradient Descent

BM Bone Marrow

CPU Central Processing Unit

CL Cascade Learning

CNN Convolutional Neural Network

CV Cross Validation

DL Deep Learning

DBN Deep Brief Network

DNA Deoxyribonucleic Acid

DNN Deep Neural Network

E2E End-to-End

EDGE Ensemble Dependency Graph Estimator

GPU Graphics Processing Unit

HAR Human Activity Recognition

IMU Inertial Measurement Unit

KDE Kernel Density Estimation

KL Kullback Leiber

Tanh Hyperbolic Tangent Function

TL Transfer Learning

TCL Transfer Learning from Cascade Learning

TE2E Transfer Learning from End-to-End

TSCL Transfer Learning from Semantic Cascade Learning

TSNE T-distributed Stochastic Neighbor Embedding

IB Information Bottleneck

ITR Information Transition Ratio

LSTM Long Short-Term Memory

MI Mutual Information

MINE Mutual Information Neural Estimation

ML Machine Learning

MLP Multi-layer Perceptron

PCA Principal Component Analysis

PWD Pair Wise Distance

RBM Restricted Boltzmann Machine

RCL Random Cascade Learning

ReLU Rectified Linear Unit

RMSprop Root Mean Square Propagation

RNA Ribonucleic Acid

RNN Recurrent Neural Network

SCL Semantic Cascade Learning

SVM Support Vector Machine

SVCCA Singular Vector Canonical Correlation Analysis

SVD Singular Value Decomposition

SC Single Cell

RNA-Seq RNA-Sequencing

SGD Stochastic Gradient Decent

Declaration of Authorship

I, Xin Du , declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- Either none of this work has been published before submission, or parts of this work have been published as described in Chapter 1.

Signed: XIN DU

_____

Date: Friday 11th March, 2022

_____

Acknowledgements

First and foremost I would like to thank my supervisors Prof. Mahesan Niranjan and Dr. Katayoun Farrahi for encouraging me to take the PhD. Thanks to their patience for helping and teaching me to acquire invaluable knowledge from which I will benefit in a lifetime.

Secondly, I would like to thank my parents for their emotional and financial support during this academic journey.

I extend my gratitude to all people I met these past years. Thanks to their time to discuss relevant ideas regarding my PhD thesis which was priceless.

Finally, I would like to thank the wonderful VLC Research Group for offering a friendly and outstanding work environment. The regular journal club, recommendations and activities let me have an enjoyable and meaningful PhD journey.

Chapter 1

Context and Contributions

In recent years, several challenging problems in Artificial Intelligence (AI) such as computer vision (Russakovsky et al., 2015), speech recognition (Dahl et al., 2013) and playing complex games such as Go (Silver et al., 2016) have been addressed using Deep Neural Networks (DNNs), achieving significant increases in performance (Chung et al., 2014; Zhang et al., 2002). DNNs are neural architectures with large number of parameters organised in layers. They are trained using huge datasets, consuming substantial computing resources. In addition to the computing demand of such architectures, propagation of gradients through deep layers comes with a well-known problem known as vanishing gradients (Hochreiter, 1998): i.e., gradients obtained by multiplying small numbers become so small in magnitudes that weights in early layers of a network do not change significantly. Techniques such as the use of Rectified Linear Unit (ReLU) and residual connections are used to circumvent this issue. Further, reducing the complexity of models is of interest, not only from the point of view of computation, but also as a way of inducing better generalisation (Burnham and Anderson, 2004; Rissanen, 1978). The architecture of a network, too, is usually set in an ad hoc manner and adaptively determining it to suit the complexity in data has also attracted attention in the literature.

Taken together, the above observations have motivated the Cascade Learning (CL) architecture introduced by Marquez et al. (2018), with its inspiration coming from Fahlman and Lebiere's work on Cascade Correlation. This work, Deep CL, forms the context of the research pursued in this dissertation.

We carry out an analysis of learning in the cascade architecture using recently introduced framework of Information Bottleneck (IB) (Shwartz-Ziv and Tishby, 2017; Tishby and Zaslavsky, 2015), which uses the Mutual Information (MI) between the pairs input-representation and representation-target to define an information plane. Learning is analysed by considering the dynamics on this plane. We compare cascade and End-to-End (E2E) learning on this plane using a set of specifically constructed synthetic and several real-word classification problems. The insights drawn from this leads to the

definition of an Information Transition Ratio (ITR) which is shown to be a useful quantity in automatically setting the number of layers required to achieve adequate generalisation.

We consider applications of CL in several Transfer Learning (TL) problems where features learned from a source domain with large amounts of data could be transferred to a related target domain in which the availability of data is low. We suggest that a property of cascade training is to pack coarse or generic features into early layers and problem-specific detailed features into later ones. This means early layers of cascade-trained models provide good source features to transfer to a target problem. We study problems in a number of diverse tasks: medical image classification, Human Activity Recognition (HAR) and gene expression analysis to confirm this claim.

1.1 Contributions

The main contributions of this work are listed as follows.

- **Cascade Learning Dynamics on the Information Plane**
  We compare the learning dynamics of E2E and cascade trained models on the information plane, illustrating on a range of problems that the central claim in Shwartz-Ziv and Tishby's claim associating Deep Learning (DL) performance to information compression does not hold. Cascade trained models that do not show such compression also achieve comparable generalisation. As part of this analysis, we also propose an ad hoc rule that is helpful in setting the depth of a neural network, the ITR. Using a range of synthetic and real world problems, we show this ratio to be a useful heuristic.

- **Knowledge Transfer Across Species**
  We show the use of TL on an interesting biological problem of making inferences from Single Cell (SC) gene expression data across different species: mice and humans. We show that computational models learned on data from mice can be usefully transferred to classifying human tissues with only a small amount of human data. Somewhat disappointingly, this problem, posed in very high dimensions, is solvable with a single hidden layer neural network and was not a good candidate to illustrate methods meant for deep networks (heuristic for network depth selection).

- **Transfer Learning from Cascade Learning**
  We show that TL from cascade trained models performs better than from E2E trained models, either in terms of accuracies or in terms of computational cost. This is demonstrated on several computer vision problems taken from natural and medical images, and on human activity recognition problems. Our central argument is that while E2E training often involves arbitrarily set DNNs, CL only

requires a few layers to be trained. We further suggest that TL performance is due to the way CL packs information into layers, coarse first and finer problem-specific in later layers. In the literature, such an observation about early layers offering better transferable features is made with E2E trained models as well. However, this is only by chance that happens during gradient descent training whereas CL achieves it by design.

1.2 Structure of Thesis

This thesis is structured as follows:

Chapter Two reviews literature relevant to this study. It covers some early attempts at constructive neural network architectures and focuses on Fahlman and Lebiere's Cascade Correlation algorithm which motivated Deep CL of Marquez et al. and followed up here.

Chapter Three presents the comparison between CL and E2E learning from the perspective of IB theory. We explore the differences by validations on synthetic datasets, small benchmark datasets and larger computer vision datasets such as CIFAR 10 and ImageNet. We further propose ITR as a criterion used to determine the structure of cascade networks.

Chapter Four presents work carried out on SC gene expression data for knowledge transfer across species, showing that models trained with data from mice (a model organism on which experiments may be performed) can be transferred to making inferences about human biology (where performing experiments is limited) with only small amounts of human data.

In Chapter Five, we consider TL for HAR problems. The empirical work here offers indirect support for our view that CL packs coarse information in its early layers and finer task-specific information in later ones. The work achieves state-of-the-art performance on benchmark datasets, with considerably lower computational cost at a fraction of free parameters.

Chapter Six presents work on TL across natural and medical image data. Our observations made in Chapter Five using the HAR problem extend to these domains too. On the CheXpert benchmark in particular, we find small cascade trained models outperform TL from far more sophisticated models (ResNet) considered by previous researchers. The same result is demonstrated on a substantive Covid-19 diagnosis problem (BIMCV dataset). We use variants of cascade training formulation to train the source models and three different methods of analysing the way the models considered differ in the way they make classification decisions.

Chapter Seven summarises the findings of questions explored in this thesis. We finally

discuss future work regarding further improvements of Semantic Cascade Learning (SCL) and CL, as well as potential scalability of Transfer Learning from Cascade Learning (TCL) to other applications.

Some derivations related to the IB theory and the Singular Vector Canonical Correlation Analysis (SVCCA) and additional experiments details are included in the appendix.

1.3 Publications

- Journal articles

  a) Published paper at *Nature Communications Biology*.
  Patrick S. Stumpf, Xin Du, Haruka Imanishi, Yuya Kunisaki, Yuichiro Semba, Timothy Noble, Rosanna CG Smith et al. "Transfer learning efficiently maps bone marrow cell types from mouse to human using single-cell RNA sequencing." Communications Biology, vol. 3, no. 1, pp. 1-11, 2020.

  b) Published paper at *Entropy*.
  X. Du, K. Farrahi, and M. Niranjan, "Information bottleneck theory based exploration of cascade learning." Entropy, vol. 23, no. 10, pp. 1-16, 2021.

- Conference articles

  a) Published paper at *UbiComp/ISWC 2019*.
  Xin Du, Katayoun Farrahi, and Mahesan Niranjan. "Transfer learning across human activities using a cascade neural network architecture." In Proceedings of the 23rd International Symposium on Wearable Computers, pp. 35-44. 2019.

- Presentations and abstracts

  a) Accepted poster at *Artificial Intelligence and Augmented Intelligence for Automated Investigations for Scientific Discovery (AI3SD) 2019 and Quantitative Systems Biology Workshop*.
  "Transfer learning across species on single cell datasets using a neural network."

  b) Accepted poster at *Advanced Course on Data Science & Machine Learning (ACDL) 2018*.
  "Understanding deep neural networks learning using information theory."

  c) Accepted presentation at *UK Mobile, Wearable and Ubiquitous Systems Research Symposium (MobiUK) 2019*.
  "Cascade transfer learning on human activity recognition."

d) Invited presentation at *University of Southampton Symposium on Interdisciplinary Machine Learning 2019*.
"Applications based on cascade learning."

- In preparation

a) A manuscript.
Xin Du, Junwen Wang, Katayoun Farrahi, and Mahesan Niranjan. "Deep transfer cascade learning from natural to medical images."

Chapter 2

Constructive Architectures and Transfer Learning

Deep Learning (DL) considers neural network architectures that contain many layers. For decades, research and practice in the use of networks have been to use shallow networks (i.e., one or two hidden layers) following the observation that a single hidden layer network is capable of universal approximation. Recent trends however have been towards deep architectures, consisting of over a hundred layers, which are conveniently trained using automatic differentiation (Griewank, 1989), trained efficiently using Graphics Processing Units (GPUs) on large-scale datasets (e.g., ImageNet (Deng et al., 2009)) achieving impressive performances. However, despite such successes, many open problems remain with respect to architecture selection, problem in which dataset sizes are small and the need for better insights into how networks are able to show good generalisation. In this chapter, starting from some early attempts at adapting neural network architectures and focus on the literature on training neural networks in a layer-wise fashion, the topic of interest in this dissertation.

From the time of early work by McCulloch and Pitts (1943), modelling the biological neuron's firing behaviour by a weighted sum of its synaptic inputs followed by a threshold or saturating nonlinearity has been in use. Connecting several of these would model a neural network. When used for a purpose of pattern recognition, this is an Artificial Neural Networks (ANNs). In pattern recognition, it is known that for Gaussian distributed classes with equal covariance matrices, the Bayes optimal posterior is a logistic: a weighted sum of inputs squashed by a saturating nonlinearity. This architecture can form linear boundaries between classes. A single neuron, as discussed in detail by Minsky and Papert (2017), has precisely the same limitation. A combination of neurons can deal with complex class boundaries, approximating the Bayes optimal classifier when the class distribution are more complex than Gaussian. While the perceptron (or logistic regression) can be easily trained, given data, a multi-layer perceptron was seen as diffi-

cult for several decades. The invention of the back-propagation algorithm (Rumelhart et al., 1986), essentially implementing the chain rule (Swokowski, 1979) in an elegant computational structure, was a breakthrough. The back-propagation algorithm enables the efficient computation of gradients of a loss function with respect to weights, not only in the output layer connecting to the targets, but also internal hidden layers for which no explicit target is available.

The application of the chain rule requires parts of the computed gradients to be propagated backwards. During this, with the use of a saturating nonlinearity in the units, small values get multiplied. Hence, when the architecture is deep, the gradients reaching the early layers tends to be vanishingly small, and then the further training of neural networks may be stopped in such End-to-End (E2E) learning mechanisms.

2.1 Adaptive Architectures

Designers of ANNs to solve any problem, usually take a fixed architecture (e.g., E2E networks), or try several architectures and select the best by Cross Validation (CV). There has been steady interest among researchers in adaptively selecting an architecture to match the complexity of the problem. This is achieved either by dynamically growing an existing model or by shrinking a trained architecture by pruning out neurons from a trained model. An example of an architecturally dynamic network using the method of Radial Basis Functions (RBF) is that of Platt (1991), later extended by Kadirkamanathan and Niranjan (1993) with a function estimation perspective and its probabilistic formulation for novelty detection by Roberts and Tarassenko (1994). The reverse of this, i.e., that of pruning neurons in a multi-layer perceptron is discussed in Optimal Brain Damage (LeCun et al., 1990).

Among this family of approaches, the Cascade Correlation architecture proposed by Fahlman and Lebiere (1990) is a powerful member, which further motivates Marquez et al. (2018) to propose Deep Cascade Learning (CL) applied to Deep Neural Networks (DNNs) in the setting of layer-wise training.

2.1.1 Cascade Correlation

Instead of updating the weights of an entire network with fixed topology like E2E training, Fahlman and Lebiere present an algorithm termed Cascade Correlation, starting with a small network and gradually adding new hidden units one by one to the network. Once a new unit is added, all previous units are frozen and the network up to that point acts as a permanent feature extractor. Cascade Correlation attempts to maximise S, the magnitude of the correlation between the residual error signal (to be eliminated), and the output of a new added unit. As a formulation of S shown in Equation 2.1, E_o is

the residual output error observed at unit o, and V_p is the candidate unit's value on the training pattern p. The $\overline{V}$ and $\overline{E_o}$ are averaged quantities of V and E_o over all patterns.

$$S = \sum_o \left| \sum_p (V_p - \overline{V})(E_o - \overline{E_o}) \right| \tag{2.1}$$

Fahlman and Lebiere (1990) apply Cascade Correlation to a two-spiral benchmark problem consisting of 194 continuous valued inputs with balanced binary outputs. To solve this task, the algorithm was implemented using the following steps:

Step 1: A one-layer network is trained until the error function plateaus.

Step 2: If the performance is satisfied, stop.

Step 3: Otherwise, freeze all the existing network weights.

Step 4: Create a candidate unit and give it inputs and trainable connections from all pre-existing units in the network.

Step 5: Maximise the correlation between created unit outputs and last measured residual error (see Equation 2.1) by training the candidate unit[1] until correlation plateaus.

Step 6: Install the above candidate unit as a new added unit in the active network by freezing all its input weights.

Step 7: Repeat steps 3 to 6 if the stop criteria[2] are not reached, otherwise stop training.

As one of the first layer-wise approaches of training neural networks, this model begins with some inputs and one or more output units decided by tasks, but no hidden units. If the performance of this small network is not satisfactory, new units are added in the following steps, otherwise, stop at the first step. Once new units are added into the network where all units are densely connected, these can be seen as appending residual connections between previously learnt features and a newly added unit as shown in Figure 2.1.

The properties of the Cascade Correlation algorithm can be summarised as follows: (a) The size of a network is not set a priori by an arbitrary choice. Instead, it is automatically determined by the progress during training and a stopping criterion. Hence, it is possible to combine it with CV to adaptively determine a suitable network size by

[1] Instead of a single candidate unit, a pool of candidate units with different activation functions can also be created and the one with best correlation score will be selected. This training process can be performed in parallel.

[2] The stop criteria can be the maximum number of iterations or no improved validation error.

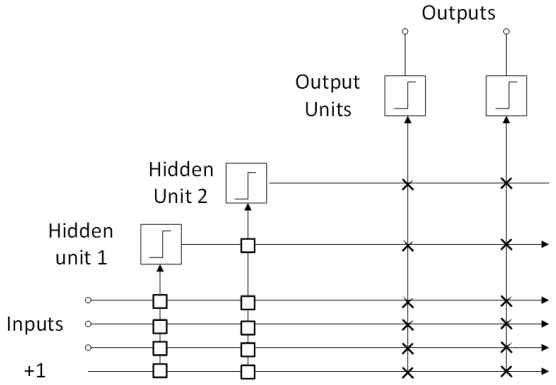


FIGURE 2.1: The architecture of Cascade Correlation. Two hidden units are added to the model where boxed connections are frozen and crossed connections are trainable. Vertical lines sum all incoming activation, and +1 is a bias input. Figure adapted from (Fahlman and Lebiere, 1990).

sequential construction rather than search across a range of architectures. (b) There is a computational advantages in a limited resource setting because small models are explored first and the network grows. Finally, (c) incrementally increasing the size of the network without removing previously learned units is appealing because only new information (not learned previously) enables network growth.

Inspired by this algorithm, Marquez et al. (2018) develop Deep CL objective to extend advantages of Cascade Correlation to modern DL. Building on that work, this dissertation presents an analysis of learning dynamics during CL and considers a range of Transfer Learning (TL) applications of it.

2.1.2 Resource-Allocating Networks

Following the Cascade Correlation algorithm, a couple of adaptive architectures are developed. A Resource-Allocating Network (RAN) is one proposed by Platt (1991) for radial basis function models, which appends a new computational unit to fit an unusual pattern from the data. The RAN starts with no hidden units and grows gradually when a novel observation is presented to the network. The outputs of this network are a linear combination of the hidden units' responses. The network sequentially sees patterns by storing some (forming the centres of the basis functions) and increases this only when patterns seen after are sufficiently novel.

For example, if the model remembers K patterns currently, the novelty of a new pattern $\boldsymbol{u}_{K+1}$ is measured by two conditions depending on two thresholds ϵ (scale of resolution in input space) and e_{min} (achievable minimum error). For each pair of input $\boldsymbol{x}$ and output $\boldsymbol{y}$, the first condition is $\|\boldsymbol{x} - \boldsymbol{u}_k\| > \epsilon_n$ to judge if input $\boldsymbol{x}$ is far away from the existing patterns. The second condition is whether the error of the network output is greater than the error threshold, measured by $e = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\| > e_{min}$. When none of the

conditions is met, the model uses gradient descent to update $\boldsymbol{\alpha}_k$, the weights of layer and the stored patterns $\boldsymbol{u}_k$, without adding a new unit to the model. Otherwise, a new unit is added to the model to correct the error. The responses of units and the output of the model are shown in Equation 2.2 and 2.3 respectively.

$$\Phi_k(\boldsymbol{x}) = \exp\left(-\frac{1}{\sigma_k^2}\left\|\boldsymbol{x} - \boldsymbol{u}_k\right\|^2\right) \tag{2.2}$$

$$f(\boldsymbol{x}) = \boldsymbol{\alpha}_0 + \sum_{k=1}^{K} \boldsymbol{\alpha}_k \Phi_k(\boldsymbol{x}) \tag{2.3}$$

where $\boldsymbol{\alpha}_0$ is the bias.

The author demonstrates that RAN is a network that can find compact representations without laborious computations on a time series prediction task. Kadirkamanathan and Niranjan (1993) further propose a RAN-Extended Kalman Filter (RAN-EKF) by replacing the Least Mean Squares (LMS) filter with EKF for gradient decent. This updating improves the compactness of the network with better performance and less time to converge.

2.2 Layer-wise Training

Extending from constructive architectures that consider unit-wise increase, constructing neural networks layer-wise is also considered in the literature. These include Deep Belief Networks (DBNs), Multi-layer Perceptrons (MLPs) and neural networks with convolutional layers.

2.2.1 Deep Belief Networks

As a generative graphical model, DBNs are a type of unsupervised neural networks trained to reconstruct their inputs. DBNs can be seen as a composition of Restricted Boltzmann Machines (RBMs) which is a generative model learning a probability distribution over a set of inputs. Hinton et al. (2006) demonstrate a method with two stages of training DBNs. In the first stage, the author used RBMs[3] (Fischer and Igel, 2012) (or autoencoders (Bengio et al., 2006)) to perform unsupervised learning for obtaining low-level features without over-fitting. In the second stage, a supervised fine-tuning is applied to the model to co-adapt the features to labels. For deep, densely-connected belief networks, this algorithm trains one layer at a time and requires knowledge of the

[3]In RBMs, the neurons are restricted to be from a bipartite graph where a pair of nodes from two sets of units have a symmetric connection and the nodes from the same set of units have no connection.

structure of network prior to its execution, instead of adaptation. In the unsupervised stage, the wake-sleep algorithm[4] (Hinton et al., 1995) is used to reconstruct the training data by generation. Namely, the objective of this stage is to minimise the differences between training data and generated outputs. RBMs are used to generate a complementary prior to get a posterior, which is a factorial distribution. Thus, the distribution of inputs can be approximated by the distribution of generations by taking derivatives of the log probability of the data. For each layer, the outputs of the previous layer (being an approximation of the inputs) will be the inputs of the next layer, with the weights of the previous layer being frozen. This process is iterative until the desired number of layers are reached. In the supervised stage, all pre-trained parameters are trainable to minimise the error function of predicting targets from input data using gradient decent.

This algorithm accomplished the training of networks, without massively over-fitting, and improved the performance of neural networks to be comparable with state-of-the-art approaches, in the layer-wise fashion. Thereafter, Bengio et al. (2006) further propose greedy layer-wise learning based on the work of DBNs. They demonstrated that greedy layer-wise unsupervised learning contributed to the improvement of generalisation by initialising weights in a region near good local minima. The authors also mentioned greedy layer-wise supervised learning, which trains each new hidden layer in a supervised network consisting of one hidden layer, and use trained weights of this new hidden layer as the initialisation of this new hidden layer in the entire network. After pre-training of all hidden layers, a fine-tuning stage will be applied to all layers for fitting targets.

However, it was illustrated that the greedy layer-wise supervised learning performed worse than the greedy layer-wise unsupervised learning. A possible explanation is that this procedure is too greedy by discarding some of information about targets. Differing from their works, the CL used in our work is also a supervised layer-wise training, but each layer is randomly initialised without the fine-tuning stage (see Section 2.3 for details). Building on RBMs and DBNs, Lee et al. (2009) propose convolutional DBNs for visual recognition tasks (e.g., MNIST and Caltech-101) and showed competitive performance in comparison to state-of-the-art results of the period.

2.2.2 Training MLPs Layer by Layer with Internal Representations

Apart from DBNs, layer-wise training is also appropriate for other DNNs (including MLPs and Convolutional Neural Networks (CNNs)), keeping advantages such as sav-

[4]The wake-sleep algorithm is an unsupervised algorithm consisting of two training phases: wake and sleep phases, to produce a density estimator. For a stack of layers, each pair of layers has a recognition weight and a generative weight in this algorithm. In the wake phase, neurons are fired by recognition connections (from inputs to outputs) while the generative connections (from outputs to inputs) are modified to increase the probability of correct reconstruction of inputs in the layer close to inputs. In the sleep phase, the direction is reversed so that neurons are fired by the generative connections and the recognition connections are updated for increasing the probability of correct reconstruction in the layer further to the inputs.

ing computational resources, automatically adapting topology of networks and better generalisation in some cases. Lengellé and Denoeux (1996) present that by gradually adding neurons into the sequential layer and layer-wise training the network, which can show improvements in terms of performance and robustness on simple tasks compared with E2E learning. Instead of using targets, this algorithm maximises the separability of representations in a hidden layer so that these representations can render discrimination easier in the sequential layer.

A network starts from an input layer and a hidden layer with several units, where the weights between the input and the hidden layer are initialised randomly (or taken as the optimal transformation matrix from a discrimination analysis). A new randomly initialised unit is then added to the hidden layer, of which the weights are updated iteratively so as to increase the separability of representations. Depending on the type of structures, this unit can only connect to the units in the previous layer or also connect to the units in the same layer simultaneously. This hidden layer is extended by adding units until no further improvement of separability can be gained, and then the process can be repeated with adding a new layer and updating the weights of this new layer. The authors suggest that for tasks requiring a large network, adding layers rather than the units in a layer may be preferable to reducing computational complexity.

2.2.3 Adaptive Structural Learning

Cortes et al. (2017) extend adaptive learning to append sub-networks (not only units) by proposing Adaptive Structural Learning (AdaNet). In this work, the network starts from several candidate sub-networks (can be multi-layer) trained by Stochastic Gradient Decent (SGD), and then selects the candidate structure with the smallest prediction errors to be appended into the network, discarding others. This process continues in iterations until no candidate structure contributes to decreasing the error. In each sub-network, the units across layers (not only within the same layer) can be connected. By training the model on a 2-class version of CIFAR 10 (Krizhevsky et al., 2010), a computer vision dataset, the authors present a better performance and less computational demands than methods using grid search, without needing to find a suitable topology of networks.

2.2.4 Layer-wise Training Using Kernel Similarity

As is widely admitted in the community, DNNs can construct compact and precise latent representations of data with improved performance. Kulkarni and Karande (2017) propose using layer-wise training of DNNs to build up better representations in a hierarchical way for classification tasks. The algorithm maximises the distance between features of different classes and minimises the feature distance within the same class via a Gaussian kernel. For each layer, the input of the current layer are the learned features

of the previous layer. The features are normalised to have a zero mean and a $L2$ regularisation term is added during optimisation (see Algorithm 1 for details). On both MNIST and CIFAR 10 tasks, this algorithm showed comparable performance to traditional E2E training of DNNs. Moreover, on fewer data, this layer-wise learning shows significant performance improvements compared with traditional E2E training. The authors also demonstrate that early layers from this algorithm provided richer features than the layer from a same position of E2E networks by visualising filters from each layer.

Algorithm 1 Training procedure for the k_{th} layer of a MLP given previously trained (k-1) layers (adapted from (Kulkarni and Karande, 2017))

Variables:
W_k: Weights of the k_{th} layer
D_{k-1}: Inputs of the k_{th} layer
X_k: Outputs of the k_{th} layer (feature representations of data, $\boldsymbol{x} \in X$)
$l_i \& l_j$: The labels of the i_{th} and j_{th} training points from n points
$T(i,j) = \begin{cases} 1 & \text{if } l_i = l_j \\ 0 & \text{otherwise} \end{cases}$: A target kernel function
μ: Learning rate, λ: A parameter controls the importance of the regularisation
Procedure Training the k_{th} layer
$W_k \leftarrow$ Random initialisation
Until convergence:
X_k=Tanh$(D_{k-1}W_k)$
$\boldsymbol{x} = \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2}$, for each $\boldsymbol{x} \in X$
$K = \exp \frac{-(-1-X_k^T X_k)}{\sigma^2}$
$\text{cost}_{min} = \frac{1}{n^2} \|K - T\|_F^2 + \lambda \|W_k\|_2^2$
$gW = \frac{d \text{ cost}}{dW}$
$W_k = W_k - \mu g W$

2.2.5 Progressive Growing of Generative Adversarial Networks

Layer-wise learning has also been applied to generative models. Karras et al. (2018) propose Progressive Growing of Generative Adversarial Networks (PGGANs) for generating super-resolution images. Figure 2.2 shows a training schematic of PGGANs that includes two parts: (a) generators used to generate image, which ideally should be indistinguishable from training data and (b) a discriminator trained to assess whether generated images are discriminated against real images. The algorithm starts with generating low resolution images and progressively increases the resolution by adding more layers to the network. The authors believed this training method can force the model to learn large-scale structure firstly, and then discover finer scale details instead of learning all scales simultaneously. Progressing training on CeLebA (Liu et al., 2015), which is a dataset consisting of $10M$ real images, provides the following advantages: (a) the generation of smaller images given by early layers is substantially more stable as less class information and fewer modes are seen and (b) the training results in a learning process that is two to six times faster and with comparable quality of generated images. Compared with CL which freezes trained layers, all existing layers in this model are trainable.

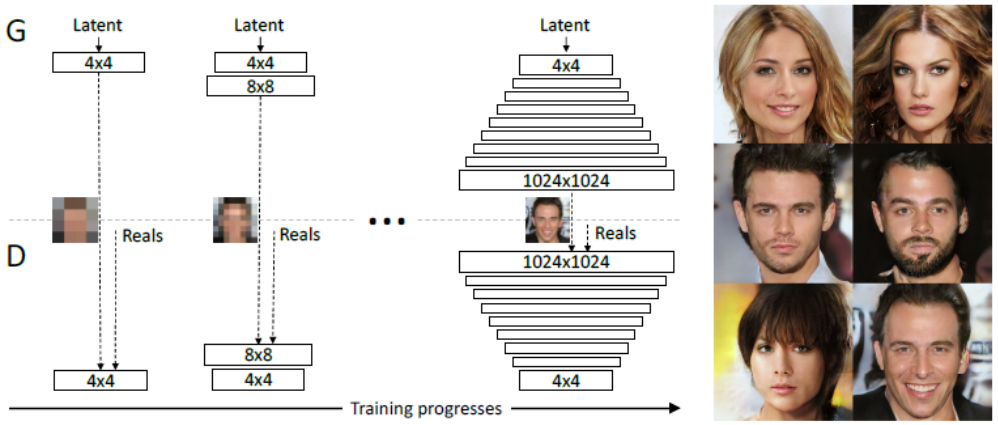


FIGURE 2.2: The diagram of PGGANs. G is the generator and D is the discriminator. Starting from a low spatial resolution of 4×4 pixels for both the generator and discriminator, the algorithm advances by gradually adding layers to both parts so that the generated images will have increasing spatial resolution. Throughout the training process, all existing layers are trainable. Six generated images are shown as examples on the right. This figure is adapted from the original work (Karras et al., 2018).

2.2.6 Locally Supervised Learning

In some of the literature, layer-wise learning is also connected to local learning. For example, taking local objective functions into account to train neural networks in a similar way to layer-wise training, Xu and Principe (1999) and T Nguyen and Choi (2019) use information potential and Information Bottleneck (IB) principle (Tishby et al., 1999) respectively, to constrain the representation of each layer in the E2E multi-hidden layers network to preserve as much information of the input as possible. Then, they add up the mutual information, between representations and the input, from all the layers together as the objective function to optimise. In that situation, both demonstrate that the layer-wise constraints showed better generalisation performance. Nøkland and Eidnes (2019) further propose a similarity loss combined with prediction loss for improving the generalisation of layer-wise training. This similarity loss encourages network to produce distinct representations for different classes.

Wang et al. (2021) focus on improving generalisation of layer-wise learning as well, based on IB theory (see Chapter 3 for details). The authors believe the simple classification loss (e.g., cross entropy) used to train each layer locally collapses task-relative information at early layers. To eliminate the diminishing of task-relative information, they propose an information propagation (InfoPro) loss to encourage local layers to preserve more task-relevant information and gradually discard task-irrelevant information. On five computer vision benchmark datasets, the proposed algorithm gains improvements in both performance and saving up to 40% of computational resources of ResNet structures. In the experiments, the task-relative information is estimated by using local test performance to approximate mutual information, which may be affected by the selection

of local prediction parts.

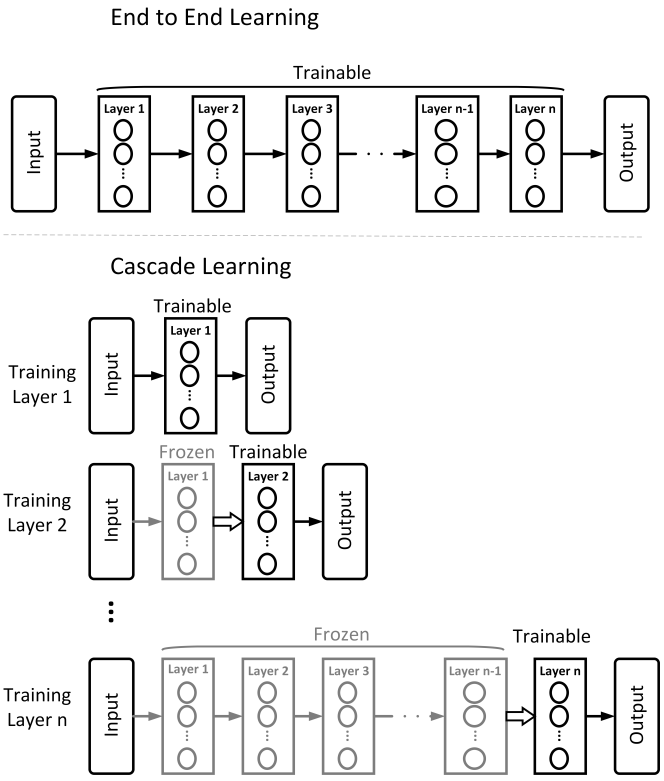


FIGURE 2.3: The schematic of Cascade and E2E training. CL starts from a single hidden layer network, including an output block. This output block is used to generate predictions of targets and consists of several dense layers depending on the requirements of the tasks. After the training of one layer, the output of this layer is the input of the next layer by freezing the trained weights of the current layer. For every layer, a randomly initialised output blocks will be connected for prediction. In contrast to CL, E2E training updates the weights of all necessary layers together in each iteration.

2.3 Deep Cascade Learning

The particular mechanism that interests us is a recently introduced architecture known as Deep CL proposed by Marquez et al. (2018), in which a network is trained layer by layer to gain significant savings in computation and memory at the expense of some performance accuracy on vision benchmark problems (e.g. MNIST, CIFAR 10 and CIFAR 100). The authors mainly introduce Deep CL in the setting of convolutional layers, but this learning rationale can also be extended to other deep feed-forward networks by circumventing the well-known vanishing gradients problem. Deep CL trains DNNs by gradually adding layers to the networks as opposed to the Cascade Correlation algorithm, which is adding units to networks. As shown in Figure 2.3, the training process in CL is progressively adding the layer to the existing network and only training the weights of the currently added hidden layer and the corresponding output block, whereas all the previous layers

are frozen. For each layer, an activation function, and sometimes also an averaged pooling layer, will be applied to the output of the current layer as suggested by the literature.

With this learning strategy, Marquez et al. present that the confusion matrix of outputs is shown to improve in a layer-wise fashion on several multi-class problems, while in E2E learning no such systematic distinguishable features are observed. At the final stage of training, both CL and E2E learning show satisfactory performance, which is also comparable between them. However, CL has significant merits in saving memory demands. For E2E learning, intermediary results, including the number of parameters of the entire network and the amount of data of all layers, need to be stored in the memory for an iteration of propagation. In CL, all previous trained layers are frozen as a permanent feature extractor, thus only the parameters of one layer (the current layer) need to be stored in RAM (GPUs or Central Processing Units (CPUs)) and all others can be cached. Hence, the memory can potentially be used more effectively, and it is possible to be integrated into smaller devices. The training iteration can be stopped by reaching a pre-defined depth or by monitoring the changes in errors on a validation set as new layers are added. We will return to this point in the next chapter, where, by computing an information ratio, we will propose a useful heuristic for model size (depth) selection.

Marquez et al. also show several ways of improving the performance of CL, such as progressively adding training epochs over layers while the first layer only has a small amount of training epochs for avoiding over-fitting. Trinh (2019) also mentions the similar way of slowing down the training of early layers for further improvements of generalisation, but the effects are limited. The author also explores the influence of adding $L2$ regularisation with disappointing results, showing that this operation is not helpful for improving the learning ability of CL. A potential reason is that the Frobenius norm applied to the weight matrices may be not the right metric for capacity control.

Beyond the advantages of gaining memory saving and better intermediate representations, Belilovsky et al. (2019) present that this layer-wise training can also scale to the much more challenging problem of ImageNet, and shows comparable performance with popular architectures (e.g. AlexNet (Krizhevsky et al., 2017) and Visual Geometry Group networks (VGG) network (Simonyan and Zisserman, 2014)) by introducing an auxiliary classifier and an invertible down-sampling operator (Dinh et al., 2017). The down-sampling operator is selected to reduce inherent losses caused by architectural elements such as average pooling. Afterwards, Belilovsky et al. (2020) propose Decoupled Greedy Layer-wise (DGL) learning for training networks with parallelization in layers.

There are two types of DGL learning: Sync-DGL and asynchronous DGL. In Sync-DGL, each set of parameters are updated in parallel across all layers, each of which processes a sample or batch of data and passes it to the next layer immediately without waiting for other samples to be processed, while the updating of the current layer happens simulta-

neously. Hence, the input of each subproblem solver changes over time. In asynchronous DGL, a replay buffer is shared between adjacent layers to enable them to reuse the sample. The delay of each layer is decided by a distribution, according to which the selected layer will process the outputs from last layer and update the weights of the selected layer, and then store corresponding outputs to the buffer.

The authors demonstrate that DGL can render efficient training of CNNs without losing performance. With the same objective to accelerate training, Brock et al. (2017) propose a variant of exact layer-wise learning to speed up E2E learning. The algorithm progressively freezes layers by annealing the learning rates of each layer to zero. According to cosine annealing, once the learning rate of a layer reduces to zero, this layer will be at an inference mode by saving computational resources as the back passes of the layer will be excluded. Ro and Choi (2021) address a similar problem by proposing auto-tuning of the learning rate in the setting of fine tuning of an E2E network. The authors connect learning rates to the weight variation during fine tuning (in TL) to make early layers have smaller weight variation for keeping low level features while later layers have larger learning rates to fit target tasks.

Ma et al. (2020) further scale layer-wise training to the ResNet (He et al., 2016), which is a deeper structure. They propose the latter layers of networks should have a more intensive distribution of convolutional layers between two pooling layers to extract more abstract features and improve the performance of later layers, which is termed down-sampling acceleration in the time domain. The work validates this idea on the CIFAR 10 and CIFAR 100 datasets by training ResNet in a layer-wise fashion and improving performance by around four percent.

2.4 Transfer Learning

The success of DL on several problems is due to the availability of large-scale datasets in computer vision (e.g., $1M$ images in ImageNet) and natural language processing problems. This, however, is not always the case. For example, medical problems often are posed on small datasets of approximately $100 - 1000$ images (Lundervold and Lundervold, 2019). Human Activity Recognition (HAR) problems (Harel and Mannor, 2011) also has the issue of data scarcity due to large consumption of time expenses and human resources in labour (Pan and Yang, 2009). The way to deal with low data settings while taking advantages of the representations learned by deep architectures is TL.

In TL, there are two domains: source and target domains, related to each other. Ideally, we hope the model learned on the source domain (D_S) can also generalise well on the target domain (D_T). However, in reality, performance degradation is caused by a variety of reasons roughly summarised as the bias between domains. TL aims to reduce the performance degradation and enable us to utilise learned knowledge from a source domain

and apply them to a related target domain where not enough data can be collected for training a model starting from scratch (Pan and Yang, 2009).

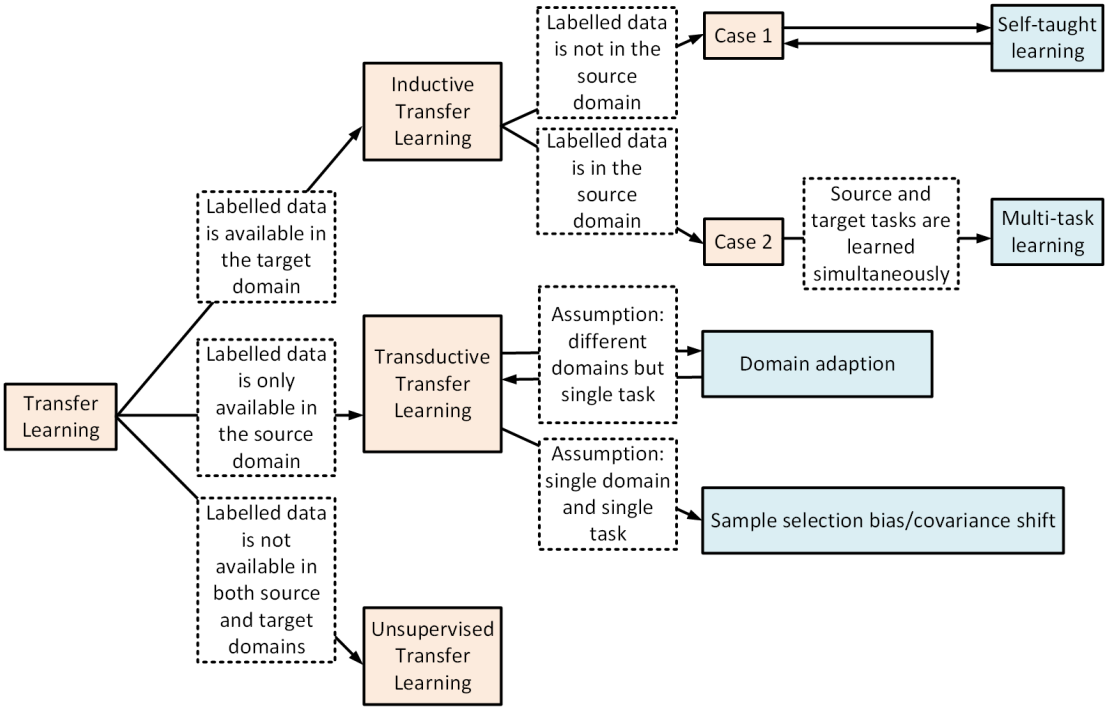


FIGURE 2.4: The strategies of TL (adapted from (Pan and Yang, 2009)).

2.4.1 Transfer Learning in Traditional Machine Learning

TL is not only specific to DL but also to traditional Machine Learning (ML). Pan and Yang summarised TL in traditional ML algorithms as three parts: inductive TL, transductive TL and unsupervised TL according to domains, tasks at hand and the availability of data (Pan and Yang, 2009). Figure 2.4 shows a categorisation of traditional TL. In inductive TL, the source and target domains have different tasks and it does not matter if both domains are same or not. When labelled data is not available in the source domain, inductive TL is similar to self-taught learning (Raina et al., 2007) where the information of labels from both domains may be different and then cannot be used directly across domains. When labelled data is available in the source domain, inductive TL resembles multi-task learning but only aims to achieve high performance in the target domain by leveraging knowledge from the source domain. In transductive TL, the source and target domains are different (either feature space or marginal distribution of data) while the targets of them are same (e.g., the source domain has handcraft images of digits while the target domain has real images of the same digits). Moreover, only the source domain has labelled data while the target domain has unlabelled data. When the marginal distributions of data from two domains are different, domain adaption (Daume III and Marcu,

2006) is related to transductive TL. In unsupervised TL, the algorithm aims at solving unsupervised tasks in the target domain (e.g., clustering, dimensionality reduction and density estimation) while labelled data is not available in the source domain, too.

2.4.2 Transfer Learning in Deep Learning

In DL, TL is applied by two most popular strategies: pre-trained models as feature extractors and fine tuning on pre-trained models. As a feature extractor, the model with layered architecture is firstly trained on the source domain to learn representations of input data and a predication layer is usually included in the model to fit the source domain task. Hence, we can utilise this pre-trained model without the prediction layer as a feature extractor and transfer knowledge learned from the source domain to the target domain. For fitting target domain tasks, a new prediction layer is usually added after the extractor. During knowledge transfer, the weights of pre-trained models (extractors) are not updated on the target domain data. In fine tuning setting, not only the new added prediction layer are trained on the target domain, parts of weights of the pre-trained model also need to be retrained for fitting target domain tasks. The applications of TL in DL achieves success in many areas, such as knowledge transfer from ImageNet to medical images (Kim et al., 2017) and knowledge transfer across different languages (Johnson et al., 2017). Belilovsky et al. (2019) show that layer-wise training can further contribute to performance as a pre-training method, which is a variant of TL. More applications of TL in DL are introduced in Chapter 4, 5 and 6.

2.5 Summary

Throughout this chapter, we introduce literature of layer-wise learning, in contrast to E2E learning, from simple perceptrons to DNNs, and TL strategies in both traditional ML and DL. Overall, layer-wise learning shows advantages in saving computational resources (including complexity or memory) while outperformed or comparable performance to E2E learning are offered. Moreover, the automatic adaption of network structures reduces the cost of searching for appropriate architectures which is consuming in E2E learning. CL and its extension greedy layer-wise learning can further scale to large ImageNet datasets with the above advantages. However, the majority of comparisons between CL and E2E learning are on the surface level of comparing performance, to the best of our knowledge. To further explore layer-wise training and comparing it with E2E learning, more analytical exploration is necessary.

Above all, the previous authors' achievements strengthen our particular interest in the information plane of CL being based on IB theory, as this plane can precisely reflect the training dynamics from the view of information transformation (shown in Chapter

3). Moreover, based on the demonstration that layer-wise learning can extract different features from E2E learning given by papers (Kulkarni and Karande, 2017; Marquez et al., 2018), further meaningful exploration would be whether TL can benefit from this layer-wise scheme because of better intermediate representations, as shown in Chapter 5 and 6.

Chapter 3

Interpretation of Cascade Learning based on Information Bottleneck Theory

In this chapter, we propose using an elegant Information Bottleneck (IB) theory to analyse learning dynamics of Cascade Learning (CL) which has shown promising results in saving computation and storage costs. As a novel learning mechanism of Deep Neural Networks (DNNs), CL also addresses the problem of vanishing gradients which causes inhibition of significant alteration of weights in End-to-End (E2E) networks' early layers. The details of CL were introduced in Chapter 2, and here, we focus on using IB theory to investigate differences between two learning mechanisms, CL and E2E. Learning dynamics is analysed by considering two mutual information terms, plotted on an information plane. The first of these is interpreted as information compression, $I(X;T)$, the mutual information between inputs X and learned representations T. The second term, $I(T;Y)$, the mutual information between representations and targets Y relates to how well the model makes predictions. Through observing trajectories of each layer in a cascade network, we observe that performance is not closely linked to information compression, as high performance is not consistently accompanied by information compression. This observation differs from conclusions about E2E learning given by Shwartz-Ziv and Tishby (2017). Additionally, we find that the later layers of a cascade network can inherit information about targets from previous layers. Gradually, later layers extract more target specific features than earlier layers. Following this line of thought, we propose an Information Transition Ratio (ITR), $I(T;Y)/I(X;T)$ towards evaluating the specificity of extracted features. We investigate the feasibility of using this ratio to determine a stopping criterion of training with comparison to Singular Vector Canonical Correlation Analysis (SVCCA). Part of work reported in this chapter has been published in the journal Entropy, entitled "Information Bottleneck Theory Based Exploration of Cascade Learning" (Du et al., 2021).

3.1 Notations

We define capitals as random variables. For example, X and Y are two random variables which are distributed according to fixed probabilities $p(X)$ and $p(Y)$ respectively, where $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ are vectors of specific values representing the realisation of these variables. T is a representation of X, and $T \in \mathcal{T}$. The notation $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{T}$ are spaces to which random variables belong. $p(\cdot)$ is the probability mass function (p.m.f.) of discrete random variables (for the sake of illustration, we only consider the distribution of discrete variables in this section). As an example, the p.m.f of T is shown in Equation 3.1, where $p(T|X)$ is a conditional probability of T given X. Based on the probability of random variables, entropy can represent the inherent uncertainty or information contained by random variables. $H(\cdot)$ is entropy and $I(\cdot)$ means Mutual Information (MI), where Figure 3.1 shows relations between them through two variables. $H(X|T)$ means the conditional entropy of X given T. The formulated relations between them are shown by Equations A.2 and A.3 in Appendix A.

$$p(T) = \sum_{X} p(X)p(T|X) \tag{3.1}$$

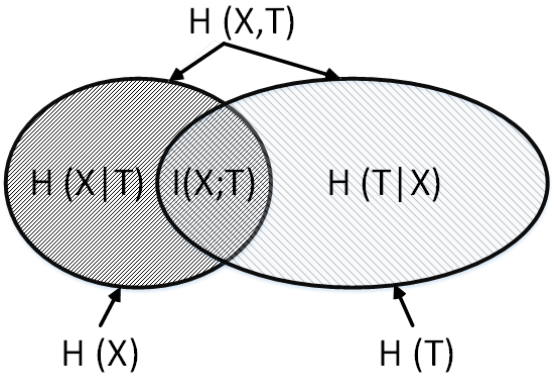


FIGURE 3.1: An illustration of relation among entropy, joint entropy, conditional entropy and MI between two random variables.

3.2 Related Work

Tishby et al. (1999)'s IB theory optimises the trad-off between economical representations of a data distribution and making information predictions from these representations. Representing data at lower data rates arises in the subject or rate distortion theory (Berger, 2003), which relates loss (or distortion) incurred by low bit rate representation of data, and has been widely applied in data compression systems such as low bit rate speech and image coding (Ortega and Ramchandran, 1998). While a pre-defined bit

rate is the driver of compression in rate distortion theory, it is analogous to what is seen in information bottleneck where a compressed representation suitable for prediction is attained. Namely, subject to relative information in an input variable X about an output variable Y, a "bottleneck" is formed by a limited code-words of input X. On the basis of this framework, Tishby and Zaslavsky (2015) further suggest using IB as a foundation of Deep Learning (DL), where DNNs are quantified by MI between layers' representations T and inputs X and outputs Y. Afterwards, Shwartz-Ziv and Tishby (2017) propose using above quantification to analyse DL dynamics. In both works, the authors use a notion of an information plane consisting of $I(X;T)$ and $I(T;Y)$ axes to observe the learning dynamics of DNNs. From empirical observations, they believe there are two stages, fitting/learning and compressing, in training DNNs. In the fitting stage, both $I(T;Y)$ between representations and targets and $I(X;T)$ between inputs and representations increase. Subsequently in the compressing stage, $I(T;Y)$ further increases before saturating, while $I(X;T)$ decreases (reflecting information compression), thereby inducing generalisation. The above observations have been illustrated using a toy binary classification example with a 12 dimensional input.

Although this framework provides an explanation of learning dynamics by analysing quantified interaction between information compression and predictive ability, Amjad and Geiger (2018) and Saxe et al. (2018) propose critical appraisals of the above information theory based analyses and conclusions given by Shwartz-Ziv and Tishby (2017). These authors attribute the observations made about the dynamics on information planes to the properties of the examples chosen rather than being general attributes of complex learning. Geiger (2021) gives a further summary of these appraisals by empathising the influences of estimators of MI. Loosely speaking, the appraisals can be summarised by three aspects: a) if the information compression is a inherent attribution of training neural networks; b) whether the information compression is closely linked to generalisation; and c) whether the effects of estimators in information compression are significant.

Regarding to the first aspect, Saxe et al. point out that the trajectory of information compression on the information plane is more of a consequence of the saturating non-linear activation function (Hyperbolic Tangent Function (Tanh)) and is not observed when the non-linearity is the Rectified Linear Unit (ReLU). Additionally, they sometimes observe information compression at the early phase of training (the learning stages). The authors of (Chelombiev et al., 2018; Saxe et al., 2018) challenge Shwartz-Ziv and Tishby (2017)'s conclusions and propose the Stochastic Gradient Decent (SGD) is not the (only) reason behind information compression, as the compression also takes place when networks are trained with Batch Gradient Descent (BGD) and Adam. Goldfeld et al. (2019a) and Schiemer and Ye (2019) believe that information compression is caused by densely clustered representations in latent spaces, which is in line with Geiger's claims about geometric clustering (Capoyleas et al., 1991). Moreover, Cheng et al. (2019) illustrate that the compression depended on datasets as well, where complex data is

hard to compress by a shallow network. In relation to the second viewpoint, Chelombiev et al. (2018) illustrate that the compression attributed to generalisation does not always take place through plotting the compression score against accuracy, where the higher rates of compression does not show significant correlation with generalisation.

Referring to the last point relating to estimators, the dominant role of the discussion regards four classes of estimators: binning estimators (Shwartz-Ziv and Tishby, 2017), Kernel Density Estimation (KDE) (Kolchinsky and Tracey, 2017), variational and neural network based estimators (Belghazi et al., 2018), kernel-based estimators (Wickstrøm et al., 2020; Yu et al., 2020). Some of other estimators (Balda et al., 2018; Gabrié et al., 2019; Goldfeld et al., 2019b; Noshad et al., 2019; Shwartz-Ziv and Alemi, 2020) are also mentioned. Strictly speaking, every estimator has its limitation. For example, binning estimators are naturally limited by the selection of bin size. The interplay between bin size and a neural networks' architecture has a great effect on the dynamics on a information plane. For small bin sizes, every sample falls into a different bin and as a result $\hat{I}(X;T)$ converges to $\log|\mathcal{D}|$, where $|\mathcal{D}|$ is the cardinality of dataset $\mathcal{D}$. A big bin size leads to more samples falling into the same bins, especially for narrow layers, which results in more information compression. According to Geiger's description, KDE is strongly affected by the value of the variance δ^2. Parameter selection differs according to the datasets and architecture sizes. Variational and neural network based estimators are not stable over several runs as discussed by Geiger. Kernel-based estimators are hard to use as the Hadamard products (Horn, 1990) will be numerically problematic for layers with many filters. Gabrié et al. (2019) propose a replica method from statistical physics to estimate the differential entropy (approximated MI). To make this estimate, a network with wide layers is trained on a synthetic dataset to satisfy the orthogonal invariance of the weight matrices. The trajectory of learning dynamics depends on the choice of the estimator, hence the information plane obtained by different estimators are not directly comparable (Geiger, 2021). Therefore, the effective conflicting claims can only be obtained when the same estimators are used in a particular situation. More importantly, Geiger believes the information plane based analysis can shed significant light on the training process of a neural network as long as the proper estimation of MI is considered. Overall, this approach can be advisable in comparing similar architectures where learning mechanism differs.

On the basis of above foundation, we propose using the framework of information bottleneck theory to analyse two learning mechanisms, CL and E2E, of neural networks with unified architectures, estimators and configurations of training. Specifically, we are concerned about what the difference is between two learning mechanisms and explore this by validating benchmark performance on various datasets from different domains. As mentioned in Chapter 2, CL trains a neural network layer by layer to gain significant reduction in computation and memory at the expense of some performance accuracy on easy problems (e.g., MNIST and CIFAR 10) (Marquez et al., 2018). Belilovsky et al.

(2019) further illustrate that this layer-wise training can also scale to ImageNet which is a much more challenging problem, and shows comparable performance with popular architectures (e.g., AlexNet and VGG). Furthermore, Du et al. (2019) utilise Transfer Learning (TL) to propose that CL packages features in a different way to E2E learning (more details are given in Chapter 5). However, all these works just present the attributes of cascade networks from the shallow performance view without further analysis. For deeply investigating the characteristics of this novel learning mechanism, we relate features extracted from each of trained layers from a cascade network to inputs and targets through MI on the information plane and compare observed dynamics to an E2E network with same structures. In the following section, we will give a brief introduction to IB theory for a deeper understanding of the aforementioned information planes.

3.3 The Information Bottleneck Theory

Evolved from rate-distortion theory, the IB is an information theoretic principle utilised to extract related information contained in a random variable $X \in \mathcal{X}$ (an input), about another random variable $Y \in \mathcal{Y}$ (an output). Namely, the MI $I(X;Y)$ is a quantification of correlations between inputs and outputs. For the sake of readability, we only introduce IB theory in this section, more details of the rate distortion theory and their intrinsic connections are shown in Section A.6. As shown in Equation 3.2, MI is defined based on the statistical dependence between two variables (X and Y). $D_{KL}[p(X,Y)||p(X)p(Y)]$ is the Kullback Leiber (KL) (Kullback, 1987) divergence between the two distributions p and q, $p(X,Y)$ is a joint distribution, $H(X)$ is the entropy of X, and $H(X|Y)$ is the conditional entropy of X and Y. Namely, relevant and irrelevant features in X are essentially determined by Y. Capturing the relevant information is the process of establishing constructive representations of X, and removing the irrelevant information is the compression of X.

$$\begin{aligned} I(X;Y) = D_{KL}[p(X,Y)||p(X)p(Y)] &= \sum_{X\in\mathcal{X},Y\in\mathcal{Y}} p(X,Y)\log\left(\frac{p(X,Y)}{p(X)p(Y)}\right) \\ &= \sum_{X\in\mathcal{X},Y\in\mathcal{Y}} p(X,Y)\log\left(\frac{p(X|Y)}{p(X)}\right) \\ &= H(X) - H(X|Y) \end{aligned} \tag{3.2}$$

In regards to compression, minimal sufficient statistics[1] (Cover and Thomas, 1999) is an important concept. By defining $\widehat{X}$ as the minimal sufficient statistics of X i.e., the

[1]In statistics, a statistic is sufficient with respect to a statistical model and its associated unknown parameters, if "no other statistic that can be calculated from the same sample provides any additional information as to the value of the parameter" (Fisher, 1922). A sufficient statistic is considered minimal sufficient if it can be represented as a function of any other sufficient statistic. In other words, $S(X)$ is minimal sufficient if and only if: $S(X)$ is sufficient, and if $T(X)$ is sufficient, then there exists a function f such that $S(X) = f(T(X))$ (Fisher, 1922).

simplest mapping of X for extracting the MI $I(X;Y)$, $\widehat{X}$ is the representation of X with respect to Y. Under an assumption of a Markov chain[2] $Y \leftrightarrow X \leftrightarrow \widehat{X}$, an optimal representation $\widehat{X} \in \widehat{\mathcal{X}}$ is obtained by minimising the MI $I(X;\widehat{X})$ under the constraint of $I(Y;\widehat{X})$. This optimisation can be formulated as the minimisation of an Lagrangian function as shown in Equation 3.3 subject to the Markov chain constraint, $I(X;\widehat{X}) \geqslant I(Y;\widehat{X})$, due to data processing inequality (Beaudry and Renner, 2012; Kinney and Atwal, 2014).

$$\mathcal{L}[p(\widehat{X}|X)] = I(X;\widehat{X}) - \beta I(Y;\widehat{X}) \tag{3.3}$$

In terms of DNNs, the output of each layer, T, can be seen as the minimal sufficient statistics or a representation of inputs, X, in conjunction with the Markov chain $Y \leftrightarrow X \leftrightarrow T$ (Shwartz-Ziv and Tishby, 2017). In that case, we consider a model where the category Y (the true label) is chosen firstly and then an image X (the input) is selected from that category for obtaining a representation T of the input. Therefore, the theoretical IB formulation of DNNs is shown as Equation 3.4, where β is the positive Lagrange multiplier operated as a trade-off argument between the level of preserving information about targets and the complexity/compression of the representation. To obtain the solution of this Lagrange function, an iterative algorithm based on the Blahut-Arimoto algorithm (Dupuis et al., 2004) is needed, which is introduced in Section A.6.1 and A.6.2 in Appendix A. Instead of obtaining solutions, we utilise this framework through tracking trajectories of $I(X;T)$ and $I(T;Y)$ to observe the transformation of information caused by training.

$$\mathcal{L}[p(T|X)] = I(X;T) - \beta I(T;Y) \tag{3.4}$$

3.4 Experiments

Aiming at tracking and observing the difference of learning dynamics between CL and E2E learning, we set up experiments over several datasets in various domains to validate our hypotheses, and concurrently utilise the two kinds of estimators of MI. In the implementation, we match the parameter settings of CL with E2E on the same data for fair comparison.

3.4.1 Datasets

We construct four synthetic datasets, use five benchmark datasets taken from the UCI repository (Dheeru and Karra Taniskidou, 2017) and utilise three benchmark datasets

[2] A Markov chain is "a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event" (Serfozo, 2009). In that case, $\widehat{X}$ is a compressed representation of X, hence $\widehat{X}$ should be completely defined given X. Namely, $p(\widehat{X}|X,Y) = p(\widehat{X}|X)$, which further implies $p(X,Y,\widehat{X}) = p(X,Y)p(\widehat{X}|X)$. An equivalent formulation is to require the Markovian independence relation $Y \leftrightarrow X \leftrightarrow \widehat{X}$.

in the computer vision field. The another synthetic dataset used by Shwartz-Ziv and Tishby for classification can be easily learnable by a network with a single hidden layer, which is inappropriate to our cascade architecture (as layer-wise training needs at least two hidden layers). Therefore, to create the synthetic datasets, we construct a randomly initialised 4 hidden layer network with a structure of 10-7-5-3 where each number refers to the width of a layer. The network has 12 dimensional binary inputs and binary outputs, and is used to generate binary classification datasets with 4096 inputs. From 50 generated datasets, we select four datasets for which a single hidden layer network yields a non-optimal performance (12-10-1 network gives $91\% \pm 1\%$ with $L2$ (0.01) regularisation). We also select other high dimensional datasets from the UCI repository where a single hidden layer neural network may not produce optimal performance. A summary of used data is shown in Table 3.1 and more details can be found in Section A.1.1.

TABLE 3.1: Datasets used and their summary statistics

Dataset	(N,P)[1]	Domain	Input	Pos (%)	Neg (%)
Synthetic 17	(4096, 12)	Artificial	Binary	63	37
Synthetic 28	(4096, 12)	Artificial	Binary	62.8	37.2
Synthetic 44	(4096, 12)	Artificial	Binary	37	63
Synthetic 48	(4096, 12)	Artificial	Binary	35	65
Dexter[2]	(600, 20000)	Text classification	Continuous sparse	50	50
Dorothea[2]	(1150, 100000)	Drug discovery	Binary sparse	50	50
Epileptic[3]	(11500, 178)	Epileptic seizure detection	Continuous dense	80	20
Gisette[2]	(6000, 5000)	Digit recognition	⋮	30	70
Human activity recoginition (HAR)[4] (6-class)	(16043, 561)	Sensor record	⋮	-	-
MNIST	(70000, 784)	Image	⋮	-	-
CIFAR-10[5]	(60000, 1024)	Image	⋮	-	-

1 N: The number of data, P: The dimension of data. "Pos and Neg" stand for the percentage of positive and negative samples in binary classification tasks.
2 From the work (Guyon et al., 2004).
3 From the work (Andrzejak et al., 2001).
4 Proportions of data (Anguita et al., 2013) in six classes are in sequence 17.2%, 14.7%, 14.5%, 18.3%, 18.4%, and 16.9%.
5 From the work (Krizhevsky, 2009).

3.4.2 Methodologies

Throughout, we use the Adam optimiser with an initial learning rate decided by tasks (details are shown in Table A.1 in Section A.1.2). For synthetic datasets, we use the network with 4 hidden layers (10-7-5-3, which is the same as the network used to construct

datasets), to carry out both CL and E2E learning. Each layer uses a Tanh activation function and the network is trained using a cross entropy loss (Murphy, 2012). Remaining details of architectures used on different datasets are given in Table A.1 in Appendix A. Based on the purpose of alleviating the natural defects of different estimators, we take into account three different estimators of MI: (a) a discrete binning or histogram approach used by Shwartz-Ziv and Tishby, (b) a Pair Wise Distance (PWD) based approach proposed by Kolchinsky and Tracey (2017), and (c) an Ensemble Dependency Graph Estimator (EDGE) proposed by Noshad et al. (2019). The pseudocode for these MI estimation methods are given in Appendix A.5, and a comparison between binning and PWD estimation is shown in Appendix A.5.1. The binning approach constructs histograms of the two distributions across which MI is to be measured. There is a compromise between resolution and data sparsity as the choice of bins/intervals size is difficult. The PWD based method utilises a KL divergence based upper bound and a Bhattacharyya distance (Kailath, 1967) based lower bound to estimate MI. Both bounds are defined based on element-wise distance. EDGE is a non-parametric estimator with linear time complexity. We use T_i to represent the output of i_{th} layer. To facilitate the narrative, we use T to substitute T_i for simplification. Briefly, we train two networks, CL and E2E, on the same data with the same structure, and estimate $I(X;T)$ and $I(T;Y)$ for each epoch of each layer from each network using the same estimator. For each epoch, we use a point on the information plane to reflect the learning status, and then get the learning dynamics of each network up to the end of training for comparison. All trajectories shown in the information plane are averaged over multiple runs (10 runs for Synthetic 17, 5 runs for HAR and MNIST datasets). The binning and PWD estimation are used for models on synthetic datasets and realistic datasets from UCI. The EDGE is used for models trained on MNIST and CIFAR 10. The calculation of SVCCA similarity is averaged over 5 runs (details of SVCCA similarity see Appendix A.7). The following section shows our comparison and discoveries from empirical experiments.

3.5 Results and Analyses

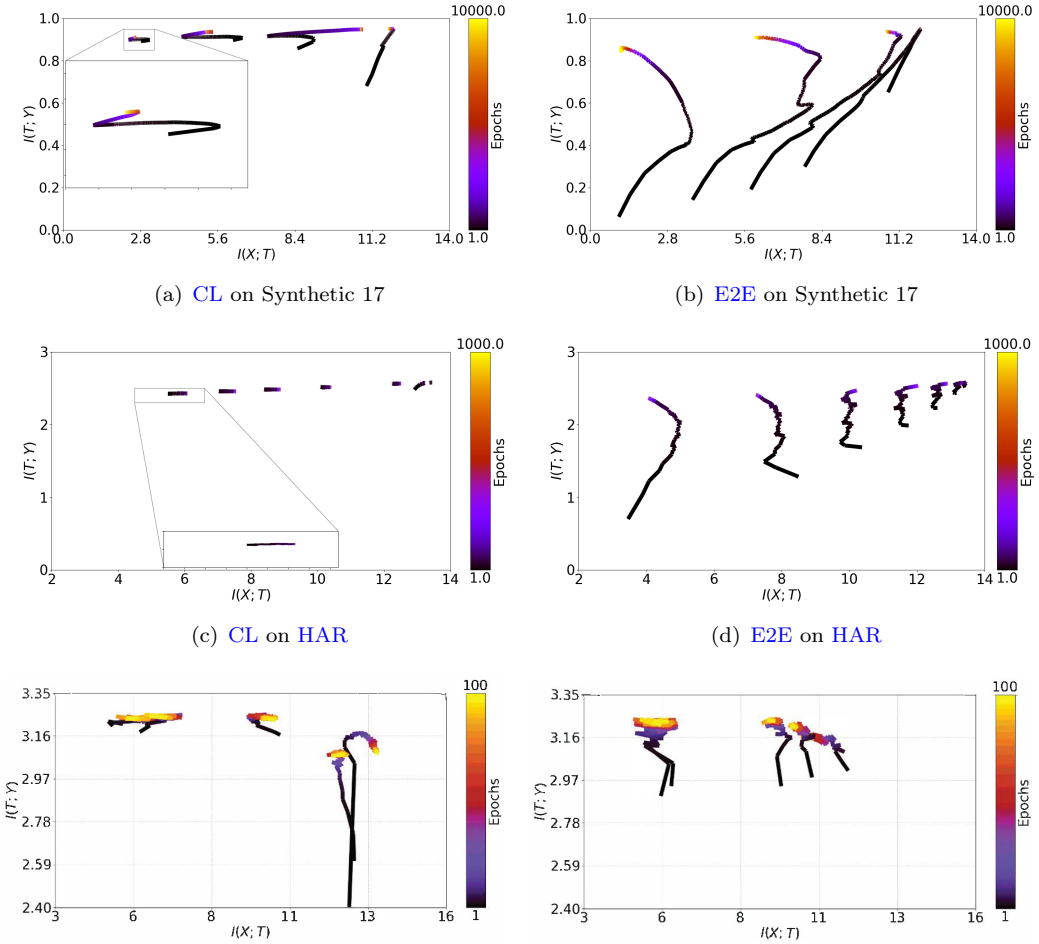
Results are illustrated through three parts: (a) the comparison of two learning mechanisms' information planes; (b) the connection between MI and performance; and (c) the relation between networks' depth and ITR. In summary, even if CL has rarely visible information compression the classification accuracy of it is comparable to E2E, which may suggest generalisation has no intimate link to compression. Additionally, ITR potentially shows feasibility of giving a stopping criterion when training cascade networks, which is different from the ratio $I(T;Y)/I(X;Y)$ given by Shamir et al. (2010), but has a similar principle (i.e., measuring the fraction of relevant information that T captures).

3.5.1 Information Planes of Cascade and End-to-End Learning

For comparing the learning dynamics of CL and E2E learning, we use information planes given by Shwartz-Ziv and Tishby's work to track both learning mechanisms' trajectories during training as shown in Figures 3.2, and give a repetition of Shwartz-Ziv and Tishby's work in Section A.2 for a sanity test. As shown in Figure 3.2, the information plane of CL (Figures 3.2(a), 3.2(c) and 3.2(e)) and E2E (Figures 3.2(b), 3.2(d) and 3.2(f)) has significant differences while the performance is comparable when the dataset and training configurations are the same. To be more specific, two different observations are as follows: (a) there is no visible information compression from each layer of CL at the final stage of training while E2E has obvious trajectories of information compression; (b) the $I(T;Y)$ starting point of each CL layer are subsequent to the previous layer's value in contrast to E2E of which starting points are decided by randomly initialisation. Section A.4 in Appendix A shows the comparison of information planes on other datasets (e.g., Figures A.7 and A.8) whose results have the consistent attributes. We also give a visualisation of the latent representations from both E2E and CL based on T-distributed Stochastic Neighbor Embedding (TSNE) projection, which is shown in Figure A.14 in Appendix A.4. Representations of intermediate layers of cascade networks show the better separability in comparison to layers of E2E networks.

3.5.1.1 Information Compression and Generalisation

As the information plane of CL shows no discernable information compression while E2E learning information planes have the visible information compression at the final stage of training, we compare learning curves and learning dynamics only on information planes of E2E for finding the connection between the information compression and the learning ability. As shown in Figure 3.3, we observe that information compression happens before and after the saturation of accuracy on test datasets. Before the saturation of testing performance, the information compression (decrements of $I(X;T)$) is accompanied with the increasing $I(T;Y)$. Therefore, the causality between information compression and generalisation given by Shwartz-Ziv and Tishby (2017) is not convincing. More results on other datasets are shown in Figure A.9. For CL, Figure A.10 shows the similar observation that the increasing accuracy is not always accompanied with the information compression. Interestingly, later layers will start compressing information earlier than former layers. At the same epoch, later layers will have relative greater gradients compared to early layers. Hence, later layers will try to give representations closer to targets, much more quickly than early layers by reducing target-irrelevant information. Overall, there is reason to believe that the information compression manner of an E2E network may come from more flexible communication between layers of a large network compared to a single hidden layer network (CL). Contrary to E2E, each hidden layer of CL does not have real-time communication with other hidden layers. Therefore, we



(a) CL on Synthetic 17

(b) E2E on Synthetic 17

(c) CL on HAR

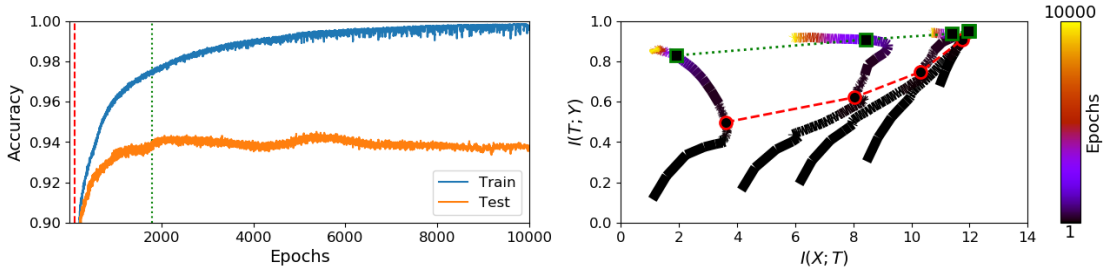
(d) E2E on HAR

(e) CL on MNIST

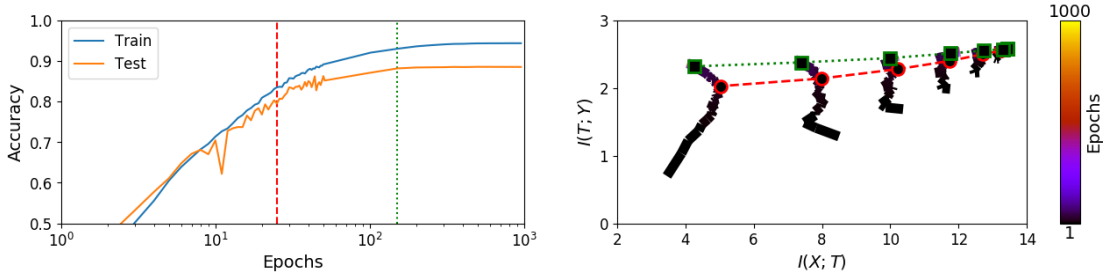
(f) E2E on MNIST

FIGURE 3.2: The comparison of trajectories on information planes between both CL and E2E learning on three different datasets. The first column shows information planes of CL and the second column gives information planes of E2E learning. From top to bottom, there are three different datasets which are Synthetic 17, HAR and MNIST in that order. From comparison, trajectories of E2E show visible behaviours of compressing information (reduction of $I(X;T)$) while CL's trajectories barely include this behaviour. Moreover, trajectories of E2E learning on simple synthetic datasets cannot show consistent dynamics with learning on harder realistic tasks (as shown in Figures A.7 and A.8). Overall, only the data process inequality chains are persistent across layers (i.e., $I(X;T_1) \geqslant I(X;T_2) \cdots \geqslant I(X;T_n)$).

believe the generalisation cannot only benefit from the information compression on the basis of above observations. For fair comparison, the training of above E2E networks contains no regularisation, but an exploration of adding regularisation and layers in E2E can be found in Section A.3. In the following section, we further explore the connection between MI and performance by placing emphasis on CL.



(a) Synthetic 17

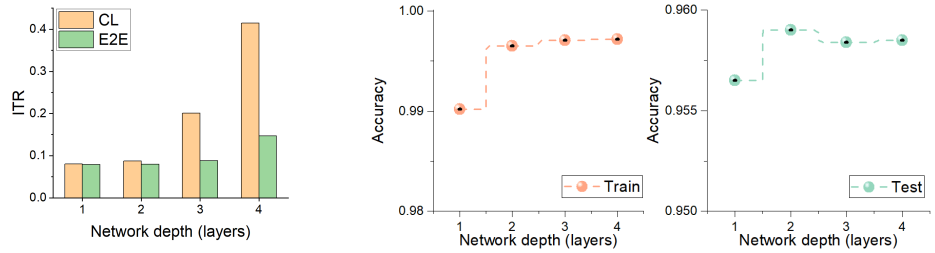


(b) HAR

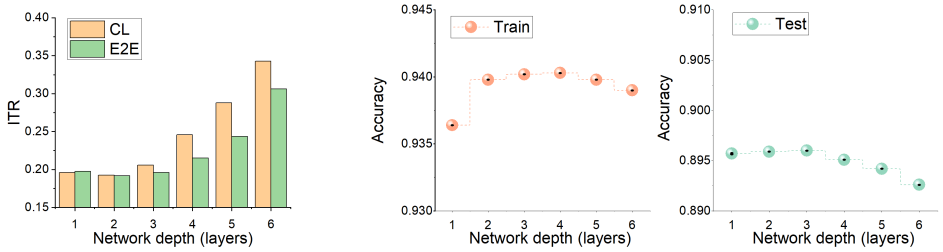
FIGURE 3.3: The connection between information compression and generalisation on artificial (Synthetic 17) and realistic datasets (HAR). Left panels show learning curves of E2E networks on both training and testing sets from Synthetic 17 and HAR datasets, respectively. Right panels show corresponding information planes for tracking learning dynamics. The red dashed line in the left panel indicates the same training epochs to red circles accompanying with dashed line in the right panel, when inflection points happens along the $I(X;T)$ axis. The green dotted line and green squares, shown in left and right panel respectively, highlight the epoch when testing performance starts saturation. From both cases, the behaviours of information compression happens between red circles and green squares when the learning settles to generalise, however it also happens after saturation of accuracy on test data (e.g., Synthetic 17). Comparisons on other datasets are shown in Figure A.9.

3.5.2 The Connection between the Information Transition Ratio and Performance

Regarding the depth of networks, we plot out how ITR, $I(T;Y)/I(X;T)$, (of the training end) change with respect to layers as shown in Figure 3.4. By focusing on the ITR of CL in Figure 3.4(a) for the synthetic dataset, we can observe the first two layers have the similar value of ITRs. Reflected by the performance shown in Figure 3.4(b), both training and testing performance of CL are increasing in this period. At the third layer, the ratio sharply increases while the testing performance slightly decreases. Simultaneously, the training performance keeps increasing. For E2E learning, the variation of ratios is relatively smoother compared to CL, which is in line with our inference in Section 3.5.1 that meaningful features are randomly distributed across layers because of flexible communications (potentially, this ratio is not suitable to decide the depth of E2E networks). The similar conjunction can also be observed in Figures 3.4(c) and 3.4(d) on the Human Activity Recognition (HAR) dataset.



(a) ITR over layers (Synthetic 17). (b) The performance of CL on Synthetic 17 data.



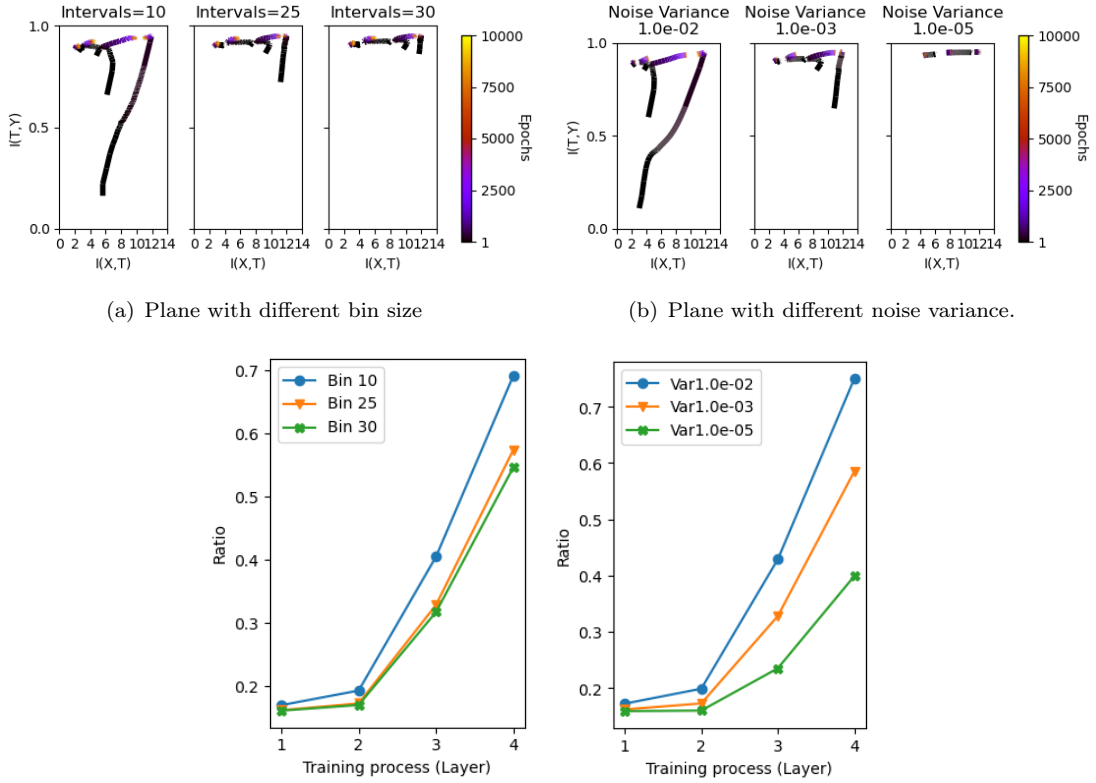
(c) ITR over layers (HAR). (d) The performance of CL on HAR data.

FIGURE 3.4: The conjunction of sharp increases of ITR ($I(T;Y)/I(X;T)$) and over-fitting of CL on Synthetic 17 and HAR datasets. ITR is computed at different layers for both CL and E2E networks. The sharp increases of ITR over layers and over-fitting always synchronise.

We also explore how performance changes with respect to the depth (i.e., number of hidden layers) of E2E networks on three different datasets (see Figure A.6 in Appendix). We can see the network with two hidden layers shows the best generalisation on the test set of the synthetic dataset. The trend of performance with respect to the depth of E2E networks is similar to the tendency of CL. As mentioned in section 3.2, the trajectories on the information planes are heavily affected by estimators even for the same estimator, different configurations may result in diverse trajectories. However, the trend of ITR over layers is relatively more stable compared to trajectories on the information plane. The comparison of them on two datasets can be found in Figures 3.5 and 3.6. Based on this comparison, in the following section we explore the feasibility of utilising ITR to help deciding the size of cascade networks.

3.5.3 When to Stop Training with Cascade Learning Using the Information Transition Ratio

From Figure 3.4(a), we observe that the best generalisation performance on test data comes from the layer or the one layer before where a rapid increase in the ITR is produced. This may mean the CL architecture on the performance plateau or on the early boundary of the performance plateau has the optimal generalisation. We further test this hypothesis on other data and summarise results in Table 3.2. These results confirm that on all tested datasets, the model depth with the best test performance occurs at the



(a) Plane with different bin size.

(b) Plane with different noise variance.

(c) Ratio with different bin size.

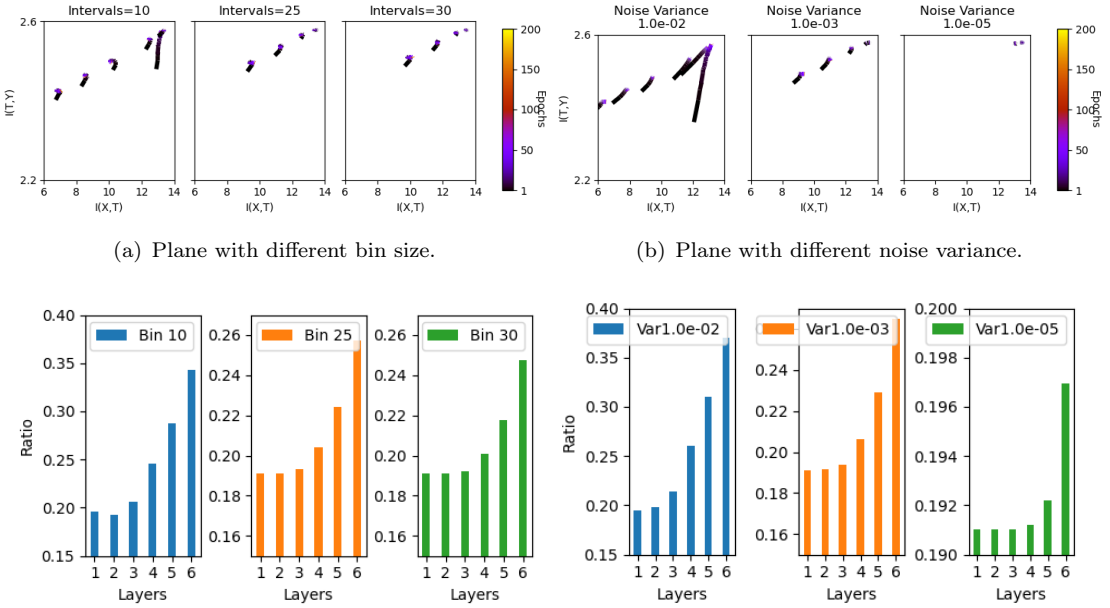
(d) Ratio with different noise variance.

FIGURE 3.5: A stability comparison between information plane and ITR of CL on synthetic 17 data with different estimators, binning and PWD. (a) and (c) show the information plane and corresponding ratio of each layer at the end of training which is estimated by binning methods with different bin sizes. (b) and (d) show the similar context while information is estimated by PWD estimators with different noise variances. There are three different bin sizes, 10, 25, and 30. The selection of noise variance has three options, $1e^{-2}$, $1e^{-3}$ and $1e^{-5}$. While the trajectories varies significantly with respect to parameter selection, the tendency of ITR is much more stable. Figure 3.6 shows same comparison on HAR data.

depth with a rapid incline in the ITR (or one layer before). Figure A.12 shows consistent tendency on a larger 23-class ImageNet task. We further extend these observations to a higher resolution which shows the alteration of ITR with respect to epochs in Figure A.13 in Appendix A. Apart from the effects of initialisation on first few epochs of each layer, the ITR keeps in a stable value on each layer. These results are significant in addressing the open question of when to stop training a cascade architecture, or how many layers to add while training to obtain optimal/significant performance.

3.5.4 Information Transition Ratio and Singular Vector Canonical Correlation Analysis

For visualising the learning dynamics, Raghu et al. (2017) propose using SVCCA to measure the correlation between representations during and after training to explain how



(a) Plane with different bin size.

(b) Plane with different noise variance.

(c) Ratio with different bin size.

(d) Ratio with different noise variance.

FIGURE 3.6: The information plane of CL on the HAR data with different estimators. (a) and (c) are based on binning estimation, and (b) and (d) are at the basis of PWD estimation. There are three different bin sizes, 10, 25, and 30. The selection of noise variance has three options, $1e^{-2}$, $1e^{-3}$ and $1e^{-5}$.

TABLE 3.2: ITR and necessary depth of models.

Dataset	CL		Dataset	CL	
	Ratio[1]	Test[2]		Ratio[1]	Test[2]
Synthetic 17	Layer 3	Layer 2	Synthetic 28	Layer 3	Layer 2
Synthetic 44	Layer 3	Layer 2	Synthetic 48	Layer 3	Layer 2
HAR	Layer 4	Layer 3	Gisette	Layer 3	Layer 2
Dorothea	Layer 2	Layer 1	Epileptic	Layer 3	Layer 2
CIFAR 10	Layer 4	Layer 3/4	MNIST	Layer 3	Layer 2
ImageNet23	Layer 4	Layer 4	Dexter	Layer 2	Layer 1/2

1 Where the rapid increment of ratio happens.
2 Where the optimal/significant performance occurs on test data.

extracted features evolves during training. Moreover, both IB theory and SVCCA can be solved based on finding eigenvalues of a matrix especially the variables are Gaussian variables. For example, Chechik et al. (2005) mention the differences and similarities between Gaussian IB and canonical correlation analysis (see Section A.7.3 in Appendix A for details). Hence, we give a comparison between our proposed ITR based on IB theory and the SVCCA, as a stopping criterion of training.

We address to the multi classification tasks, MNIST and CIFAR 10, under CL scheme in this section. We obtain the SVCCA similarity of the representations between each intermediate training epoch and final epoch of the same layer to track transformation of

representations during training. We assume that for the CL scheme, once the SVCCA similarity has converged, any remaining training process is redundant, inspired by Raghu et al. (2017).

As shown in Figure 3.7, the ITR shows a sharp increase at the forth layer while the alteration of the SVCCA similarity of this layer is negligible on the MNIST dataset. For each layer, the SVCCA similarity starts from a small value and increases to 1 after a few training epochs. For CIFAR 10 learnt by networks with convolutional layers, the ITR sharply increases at the forth layer in Figure 3.8 while the performance saturates at the same layer. There is no clear regularity of SVCCA similarity tendency over layers, but the SVCCA similarity of each layer will be converged after approximate the 10_{th} epoch. Reflected on the Figure 3.8(b), the majority of training epochs do not significantly contribute to improving the performance of current layer.

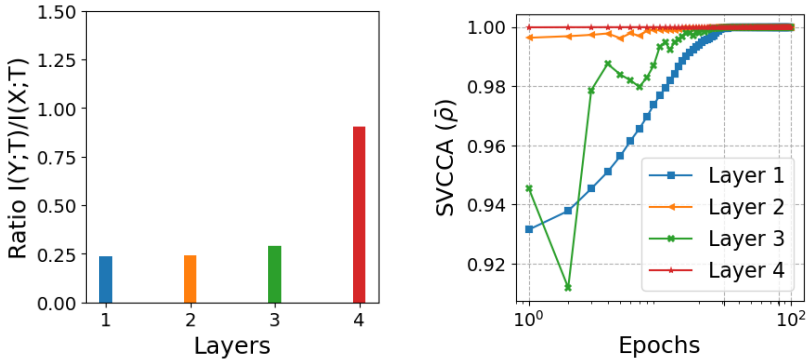
Hence we assume we can stop the training of a layer once the SVCCA of the layer is converged. For validating it, we train the same network with the above stopping paradigm which is only training 10 epochs of each layer on CIFAR 10. As a result, the performance negligibly drops from 85.11% (150 epochs) to 84.9% (10 epochs). Therefore we can potentially use SVCCA as an indicator of cutting off point during training a layer for saving training computational resources (both time and memory) almost without performance sacrifice.

In both cases, ITR increases rapidly after a certain layer, which is noticeable. After the certain layer, the training of further layers is dispensable as they barely contribute to performance. Hence, ITR shows the better ability of deciding the depth of networks in comparison to SVCCA similarities which do not show clear regularity over layers. SVCCA may be used to observe when the training of each layer of a CL model starts to become unnecessary as the further training after some epochs only shows negligible pruning of the latent representations in training. Moreover, ITR is much more suitable to be a stopping criterion than SVCCA similarities as it can be estimated in real-time when the representations from the end of training are dispensable.

Considering the connection between Gaussian IB and SVCCA, we also investigate how the ratio will change based on a hypothesis that all the variables are Gaussian as shown in Figures A.17. Since the variables, X, Y *and* T, are not exact Gaussian variables, the estimated MI do not match with previous non-parametric estimations perfectly (see Appendix A.5.2 for more detailed discussion).

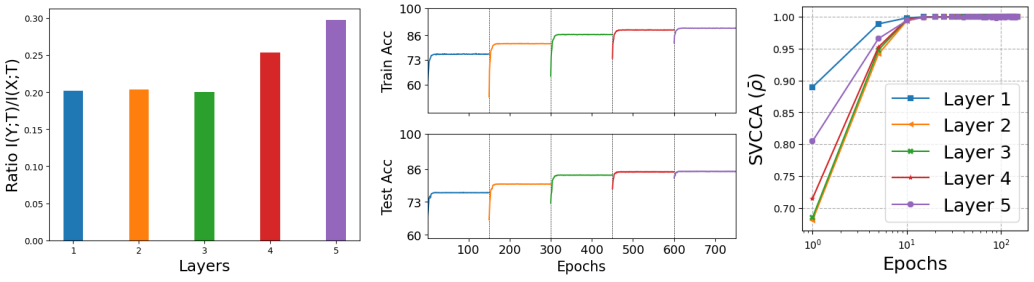
3.6 Discussion

Up to now, we have illustrated the difference between CL and E2E through information planes on both artificial and practical datasets, and explored the connection between the



(a) Ratio against the depth of the model (b) SVCCA between middle epochs and last epoch of a same layer.

FIGURE 3.7: The comparison of learning dynamic analyses between information ratio and SVCCA on MNIST using CL. (a) ITR rapidly increases from the forth layer; and (b) the SVCCA similarity starts to converge after the 30_{th} epoch of the first layer approximately, and layers after the third layer did not show obvious change of SVCCA during training. Meanwhile, the optimal classification performance is $98.47\% \pm 0.05$ at the third layer. All the results are the averaged values over 5 runs.



(a) ITR (b) Performance (c) SVCCA

FIGURE 3.8: The learning dynamic analysis of CL on CIFAR 10. (a) ITR starts growing rapidly at the forth layer; (b) both training and test performance is gradually increasing layer by layer and testing accuracy shows saturation at the forth layer while training accuracy keeps growing; and (c) the SVCCA similarity converges on the few early epochs after which the majority of training epochs is dispensable.

ITR and generalisation for finding a stopping criterion of adding layers to a CL network. During the above exploration, we notice there are many other authors present related work. Therefore, we give a comparison to previous works by discussing the following three aspects: (a) whether information compression is inherent; (b) if ratio can help decide the depth of a CL network and (c) why the IB theory based objective function can improve performance of CL.

3.6.1 Information Compression Reflected on End-to-End & Cascade Learning

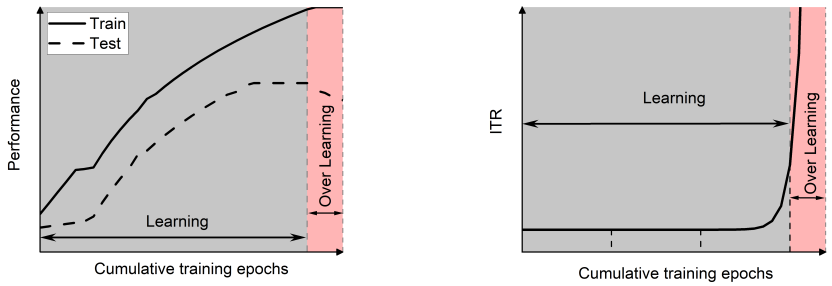
Since the publication of Shwartz-Ziv and Tishby, there has been some in-depth discussions on how realistic information compression is as a factor explaining learning in DNNs.

For example, attributing the causes of the observed compression phase to stochasticity in training (Balda et al., 2018; Shwartz-Ziv and Tishby, 2017), the specific activation function used (Saxe et al., 2018), initialisation of models (Chelombiev et al., 2018) and the method used for estimating MI (Schiemer and Ye, 2019). There are also authors who believe information compression is not inevitable such as (Fang et al., 2018). However, a majority of disagreements are based on multiple factors causing uncertainties such as datasets, model architectures, estimators and optimiser. Geiger points out the conflict or agreement based on multivariate comparison is superficial(Geiger, 2021). For example, Jónsson et al. (2020) observe information compression based on Mutual Information Neural Estimation (MINE) (Belghazi et al., 2018), while Li and Liu (2021) believe there is no information compression in Convolutional Neural Networks (CNNs). In order to eliminate such uncertainties, we unify configurations of training on each dataset for comparing the information plane of two learning mechanisms, CL and E2E. When we observe the dynamics on the information plane using CL over a range of problems, we find that the compression phase is not always observed compared to E2E, and when it is, it is mainly restricted to the early layers of learning (as long as over-learning is not happening). This would suggest that when reasonable features have been learnt in early layers, and their weights frozen in training subsequent layers, there is no flexibility within the network to move in a direction that is sometimes seen as a compression phase. Meanwhile, as the communication between layers from E2E network, the extracted useful features may be distributed over different layers, which may result in the obvious information compression. Additionally, this flexibility of communication between layers may also cause the information compression through over-fitting under a significant number of training epochs as discussed in Section 3.5.1. In extreme cases, the infinite training of E2E may force all layers representations to be similar to targets by discarding task-irrelevant information, while CL has limitation of a single hidden layer's power and has no communication between layers and then cannot over-discarding those task-irrelevant information. In addition, Wickstrøm et al. (2020) also shows that the information compression seems more prominent or happens after over-fitting through a tensor-based estimator (Yu et al., 2020).

3.6.2 Qualitative Learning processes Related to Information Transition Ratio

According to the observation of $I(X;T)$ and $I(T;Y)$ in our experiments, we notice the ratio $I(T;Y)/I(X;T)$, which we term as ITR, can be used as a guideline in determining when to stop the training of CL. Our analysis suggests that the training can be divided into two stages, as shown in Figure 3.9, through comparing performance and the ITR.

Inspired by the ratio $I(T;Y)/I(X;Y)$ given by Shamir et al. (2010), we explore the connection between ITR, $I(T;Y)/I(X;T)$, and performance. Shamir et al. demonstrated



(a) Performance with training process.

(b) Ratio with training process.

FIGURE 3.9: The learning process of neural networks. (a) shows the evolution of performance on both train and test datasets during training the model. (b) shows the ITR ($I(T;Y)/I(X;T)$) with respect to the cumulative training epochs. The training can be divided into two stages: a) learning the features and fitting to a reasonable performance represented by the grey area; b) over learning the features as the pink area.

that the size of test datasets can only increase up to a factor $I(X;Y)/I(T;Y)$ of the training sets for obtaining the similar performance on both train and test sets. The underlying reason is the bottleneck variable, T, only captures a portion, $I(T;Y)/I(X;Y)$, of relevant information. Under the similar reason, we assume $I(T;Y)$ controls the specificity of extracted features to training samples (a risk term), and $I(X;T)$ is in charge of the complexity of features (a regularisation term). Therefore, the trade-off between these two terms can be measured by their relative speed of changing (the ITR). We assume that the larger values of the ITR (when $I(X;T) \neq 0$) represents more specific extracted features to the targets, since $I(T;Y)$ is larger while $I(X;T)$ is relative small. Namely, the features/representations (T) extracted by the model gradually lose parts of information for better fitting to the training targets (Y). As this line of thought, the first stage correlates to the learning stage where the model trains to extract general useful features and loses some redundant features, reflected in the increasing ITR. In this stage, as the general feature is dominant, $I(X;T) \gg I(Y;T)$, the value of the ITR is relatively small but slowly increases with time. The second stage of training can be viewed as over-training, where the model keeps extracting features more specific to the target than the first stage. This results in an increased or saturated prediction accuracy on the train data and sharper increases of ITR in comparison to the first stage. The prediction accuracy on test data decreases in this stage, implying over-fitting on specific features to the train data. We focus on finding the boundary between first and second stages to balance test performance and specific information from training data. In the next section, we further discuss the relation between ITR and the depth of networks.

3.6.3 Information Transition Ratio & Network Depth

Shwartz-Ziv and Tishby state generalisation benefits from information compression.

However, our results show that the information compression ($I(X;T)$ reduction) not only happens before over-fitting but also after it. Therefore we, like many other authors (Darlow and Storkey, 2019; Gabrié et al., 2019; Goldfeld et al., 2019a), believe that information compression does not induces improvements of generalisation. Much research also focuses on $I(Y;T)$. For example, Perin et al. (2020) illustrated using $I(T;Y)$ as a metric to decide which epoch to stop training to avoid over-fitting. However, Amjad and Geiger (2020) and Cheng et al. (2019) illustrated larger $I(T;Y)$ cannot guarantee better generalisation. For further exploring the connection between generalisation and information dynamics, we investigate the connection between accuracy and ITR in this work. As the demonstration given by Cheng et al. (2018) and Cheng et al. (2019) who believe both $I(X;T)$ and $I(T;Y)$ contribute to final performance, we further believe there is an optimal balance range between $I(X;T)$ and $I(T;Y)$ for significant performance. Although the relationship between accuracy and $I(T;Y)$ is not a simple linear positive correlation, we find the specialisation of features from the network may be quantified by the transition ratio, $I(X;T)/I(T;Y)$. The sharp growth of ratio will result in too specific representations and under-performance on the test set based on the experiments for cascade training. The relative speed of evolution between compression and prediction shows a more reasonable relation to the performance on test data. The ITR is a useful way to represent this relative speed and decide the depth of networks while limited data is available and can only be used for training (as the ratio is estimated on the training set). Furthermore, as this ratio relates to the specialisation of features, it potentially gives the ability of guiding where the starting point of TL should be, which can be a meaningful direction for investigating in the future. Moreover, ITR is more stable than trajectories on the information plane when the configurations of estimators changes (Schiemer and Ye, 2019) as shown in Figures 3.5 and 3.6. We also provide a way to find the rapid increase using the convolution of the ITR with a step function. The peak of this convolution output can pinpoint a layer at which the ITR increases sharply (see Figure 3.10). As shown in the Figure 3.10, the step convolution is the output of convolution of ITR with a step function. We use the peak of this output over layers to select the reasonable depth of a network. In this process, ITR of each dataset is normalised to the range [0,1] and subtract the mean of ITR on each dataset separately. The result of this step convolution shows a maximum at the point of interest, successfully picking them up in 10 of the 12 datasets we have worked with. On the two where this fails, Dexter is a very easy dataset that is separable by a single hidden layer network; and in the ImageNet problem, the difference between the layer we identified by inspection and the peak of step convolution differ by just one layer.

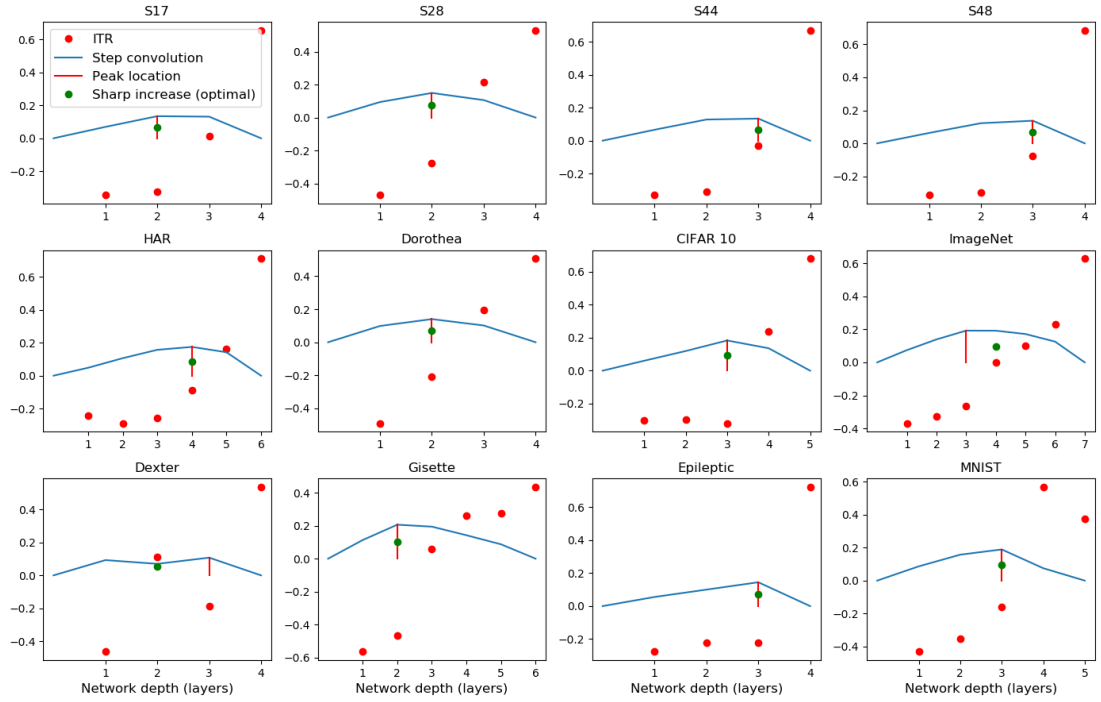


FIGURE 3.10: A depth indicator of networks given by the convolution of ITR and a step function on different datasets. Red dots are the values of ITR over layers. Blue solid lines are outputs of convolution of ITR and a step function. Red vertical lines give the location of the peak value of convolution outputs, and the green dots locate the depth of network showing optimal performance or sharp increase of the ITR.

3.6.4 Objective Functions Related to Information Bottleneck and Cascade Learning

Under the popularity of IB theory and CL, many authors also raise interest in exploring layer-wise learning by using the IB analysis. Instead of comparing the information during learning between E2E and layer-wise training like we do, those authors focus on objective functions for improving the performance of layer-wise learning. Elad et al. (2019) explore the effects of layer-wise training based on this information framework, and a noise-regularised version of the information bottleneck function is used to replace the cross-entropy loss. However, in some cases the estimated information over layers is against information inequality process (later layers show more information than early layers). Thereafter, Wang et al. propose the inferior performance of layer-wise learning is caused by gradually decreasing $I(T;Y)$ over layers through comparing $I(T;Y)$ of greedy layer-wise learning and E2E learning (Wang et al., 2021). They further suggest this inferior performance can be improved by replacing cross-entropy loss with an information propagation (InfoPro) loss, which can avoid a collapse of task-relevant information ($I(Y;T)$) at each layer during training. However, Amjad and Geiger (2020) and Cheng et al. (2019) argue that a high $I(T;Y)$ may be not the main reason for successful performance. Therefore, we believe that Wang et al. (2021) should have focused on

both terms of the proposed InfoPro loss, where the first term tries to maximally retain information about the input while the second term attempts to maximise task-relevant information. We believe the synergistic effect of two terms is a much more reasonable explanation of improved performance given by (Wang et al., 2021) instead of only considering $I(T;Y)$, which has underling connections to our proposed ITR. In addition, Fang et al. (2018) also illustrated in Figure 4 that the E2E network which keeps a high value of both $I(X;T)$ and $I(Y;T)$ terms will give better generalisation by comparing Multi-layer Perceptron (MLP), LeNet (LeCun et al., 1989) and DenseNet (Huang et al., 2017). In aggregate, we believe both $I(X;T)$ and $I(Y;T)$ have contributions to generalisation, hence, paying attention on the relative speed of transformation between them may be helpful in generalisation.

3.7 Summary

Information theory is a useful tool to understand the learning process in multi-layer networks. When trained to learn a mapping between input-output data, networks develop complex representations from which predicting the target is easier than from the raw inputs. Shwartz-Ziv and Tishby's work introduces the idea that representation learning is achieved by information compression during later stages of learning. In this chapter, we have used this framework to help with understanding the information dynamics of a constructive approach by building multi-layer networks using the layer-wise trained cascade architecture. The dynamics of learning observed on the cascade architecture is different from that of an E2E trained model. Any compression phase we observe is mainly restricted to the early layers, and presumably once it has extracted useful features, later layers show no such obvious compression for removing the redundancy at later stages. Additionally, we notice the specificity of features from different layers can be related to a transition ratio comparing the two mechanisms based on information learning dynamics. This ratio reflects the relative relationship between the information from labels and original inputs, and shows the ability of finding suitable (or optimal) depth of a network on the tasks. We already validate this phenomenon over datasets from various domains and networks. We also compare this IB based ratio to SVCCA, where an underlying connection exists, in terms of stopping training. We believe the ITR is better as a criterion of stopping adding layers to CL. Inspired by Fang et al. (2018), a potential point in the future works is extending the analysis of deciding the number of neurons of a fixed layer in CL based on the proposed ITR.

Chapter 4

Neural Network based Transfer Learning for Single Cell Classification

In this chapter, we explore Single Cell (SC) classification tasks based on Transfer Learning (TL) between two species: mice and humans. As an ideal model organism, mouse models are widely used in researching human diseases such as diabetes, cancer, epilepsy, and obesity which may be caused by gene mutations (Bedell et al., 1997). Furthermore, biomedical research on mice is cheap and needs less time for each life cycle (e.g., one mouse year equals about 30 human years) (Rosenthal and Brown, 2007). Hence, a greater number of experiments can be implemented on mice than on humans. Moreover, gathering large amounts of data from humans is difficult due to the limitations from ethic, privacy, rare diseases, cell culture cycles, environments, etc. (Guasch and Fuchs, 2005).

There have been a number of studies compare the gene expression profiles at the tissue level across species (Merkin et al., 2012). For example, La Manno et al. (2016) report a majority of cell types and gene expressions are conserved across species (e.g., mice and humans). However, this conservation may only exists in parts of tissues. To further understand the feasibility of mapping information across species, we explore the applicability of classifiers trained on SCRNA-Sequencing (RNA-Seq) transcriptome data taken from mouse cells and evaluate their performance on the equivalent measurements in human cells. We show that classification performance is retained in some cell types but not on others. We then apply TL to retrain classifiers pre-trained on the mouse model data to the human system, showing significant improvements in overall performance while limited human data is available. The work is demonstrated on three different tissues (datasets): a local Bone Marrow (BM) dataset leading to (Stumpf et al., 2020) and two publicly archived datasets (GSE84133 and GSE76381). The work on the BM dataset

presented in this chapter was published in the journal Nature Communications Biology, entitled "Transfer learning efficiently maps bone marrow cell types from mouse to human using single-cell RNA sequencing" (Stumpf et al., 2020).

4.1 Background and Related Work

Transcriptome analysis is an important way used to map genotypes to phenotypes in biology and medicine. One cell usually only reflects the activity of part of genes while the information of identical genotypes is shared over all cells. Hence, the conventional assumption that cells from a given tissue are homogeneous likely misses important cell-to-cell variability. Nowadays, SCRNA-Seq is widely used for analysing transcriptome in biology to provide more precises understanding of the transcriptome in individual cells (Hwang et al., 2018).

In contrast to conventional bulk RNA-Seq which measures the average expression level for every gene across a large population of cells, the SCRNA-Seq provides quantitative measurements of every gene expression in a single cell so that it enhances the feasibility of studying heterogeneous systems for delineating cell lineage relationship (Petropoulos et al., 2016)), and characterising outlier cells for discovering cancers (Shaffer et al., 2017; Shetta and Niranjan, 2020). SCRNA-Seq also provides the possibility of investigating new biological hypotheses where cell-specific transformations in the transcriptome are essential such as identification of cell types, inference of gene regulatory networks and stochasticity of gene expression across cells. A brief description of obtaining SCRNA-Seq data with high dimensions is shown in Appendix B.1.

The combinatorial effects of diverse biological processes determine the identity of an individual cell, including cell's response to local environment signal (e.g., the binding of a signalling molecule to a receptor), physical environments (e.g., oxygen availability) and nutrient availability (Wagner et al., 2016). However, both high dimensional SC data and biological processes are difficult to analyse. For example, original high dimensional SC data is often noisy, and determining which biological signals should be considered need to be customisable for different situations (Sun et al., 2019). Therefore, dimensionality reduction is a popular technique used to reduce the original data into a lower dimensional latent space based on the hypothesis that the above transformation may correspond to biological influences on the transcriptome (Cleary et al., 2017; Stein-O'Brien et al., 2018; Zhu et al., 2017).

Nonetheless, evaluating the accuracy of mapping and interpreting the low dimensional representations of the biological processes are a challenge which requires a gold standard provided by biological validation (Stein-O'Brien et al., 2019). A majority of applications do not have such a gold standard (e.g., fixed criteria or rules to define a specific cell type). Instead of directly analysing these challenging datasets, an alternate method is utilising

knowledge learned from different but related data sources. Intuitively, a similar set of biological process should be reflected across multiple datasets and measurement assays of the same biological system, in addition, subsets of cellular features may be preserved across related biological contexts (Stein-O'Brien et al., 2019).

The limited number of high quality labelled data is another challenge in identifying individual cells. Although, advanced protocols result in an increasing number of SC data, so far, the majority of SC data are labelled manually by characterising clusters of cells or utilising fluorescence-activated cell sorting (FACS). Both of these methods have inherent inadequacies. Manually clustering is affected by the choice of clustering algorithms and the limitation of knowledge from annotators. For example, owing to the high dimensionality, distances between cells (data points) become similar, hence distance based clustering methods tends to be unreliable (Kiselev et al., 2019). FACS requires extra experiments in addition to the sequencing which cannot be incorporated into the high throughput methods. As a consequence of these limitations, obtaining large datasets of labelled SCRNA-Seq is scarce. To address the above problems we propose to use TL, a method which leverages learned knowledge from the related source to improve the performance of the current targets even if there is only limited data in the target domain.

As introduced in Section 2.4, TL is the branch of machine learning that exploits the fact that if two datasets share common latent spaces, a feature mapping between the two can identify and characterise relationships between the data defined by the individual latent spaces (Caruana, 1997; Pan et al., 2008). Hence TL is a technique to assess the transferability of features across multiple datasets. TL aims to mimic the human ability to learn new concepts from limited examples by associating new information with prior understanding (Pan and Yang, 2009). In the TL process, information gained from solving a problem in a source domain is passed to another related problem in a target domain thereby improving target domain performance. The gain from such knowledge-transfer is particularly apparent whenever data is abundant in the source domain but scarce in the target domain. In this case, new concepts can be efficiently learned in the target domain from limited training samples via leveraging of prior knowledge from the source domain.

TL on SC data has raised several authors' interests. Stein-O'Brien et al. (2019) propose SC Coordinated Gene Activity in Pattern Sets (scCoGAPS) from which a latent space from the source SC data is defined. The evaluation of the latent space on the target data is implemented by TL via a tool named projectR. scCoGAPS is a non-negative matrix factorisation (NMF) algorithm used to define latent spaces from the source data which is modified for accomplishing a well-defined latent space consisting of shared biology across independent and biologically related datasets. In scCoGAPS, SCRNA-Seq data is a gene expression matrix ($X^{N\times p}$) of which each column represents a cell and each row represents an observed gene expression value. scCoGAPS decomposes this data

matrix into two related matrices, the amplitude matrix ($A^{N\times r}$) and the pattern matrix ($P^{r\times p}$), where r is the rank. The row of A quantifies the variation among genes, and the column of P quantifies the variation among cells. The number of columns of A is the same as the number of rows of P, and represents the number of dimensions in the low-dimensional representation of the data. The projectR is achieved by a generalised least squares fit to the target data via estimating the patterns associated with the amplitude matrix from NMF. Mathematically, a least squares problem $\|Y - AQ\|^2$ is solved by estimating the matrix $Q^{r\times q}$, where $Y^{N\times q}$ is a gene expression matrix on the target domain, and A is from the source domain. The number of target domain cells q is usually smaller than p (the number of source domain cells). Using scCoGAPS and projectR, Stein-O'Brien et al. explore the low dimensional latent space on the mouse retina data, and demonstrate their approach can annotate latent spaces and transfer annotations for classifying cells. However, their approach did not consider technical differences between source and target domains (e.g., from bulk RNA-Seq to SCRNA-Seq), which may cause spurious predictions on the target domain (Lotfollahi et al., 2020).

Peng et al. (2021) apply a similar methodology named Common Factor Integration & Transfer Learning (cFIT) on fetal brain SCRNA-Seq data. cFIT is implemented by minimising the objective function on the source domain, as shown in Equation 4.1, to capture various batch effects. $X_j^{n_j\times p}$ is a gene expression data from the j_{th} batch at a single lab using a single technology, n_j denotes the number of cells from this batch, and p is the number of genes. By this matrix factorisation, $H_j^{n_j\times r}$ captures the biological heterogeneity from cell types, and $W^{p\times r}$ constructs a common factor space where r determines the dimension of a latent space. $\mathbf{b}_j^{p\times 1}$ is a vector used to capture batch-associated shift, and $\Lambda_j^{p\times p}$ is a diagonal matrix used to control the discrepancy of gene expression caused by batch-specific technical effects. $N = \sum_{j=1}^{M} n_j$ is the total number of cells from the source domain and $W, H_j, \Lambda_j, \mathbf{b}_j \geq 0$ (each element of the matrix or vector is non-negative). γ is a positive parameters to decide the penalisation for ensuring the identifiability of the model across batches. In TL, an estimated matrix W from the source domain are used as a reference matrix in the target domain. To minimise an objective function $G(H_{target}, \Lambda_{target}, \mathbf{b}_{target}; W_{source})$ (shown in Equation 4.2), $(H_{target}, \Lambda_{target}, \mathbf{b}_{target})$ can be recovered on the target domain and subject to the non-negative constraints. Compare to previous methods, Peng et al. believe their method impose weaker assumption on the composition of cells from different sources and helped establish a comprehensive landscape of brain cell type diversity. However, this work only explores the possibility of transferring knowledge on one tissue (brain), and the selection of parameters r shows significant influences in the performance of cell classification.

$$J(W, \{H_j, \Lambda_j, \mathbf{b}_j\}_{j=1}^{M}) \coloneqq \sum_{j=1}^{M} \|X_j - (H_j W^\top \Lambda_j + \mathbf{1}_{n_j} \mathbf{b}_j^\top)\|_F^2 + \gamma N \sum_{l=1}^{p} \left(\sum_{j=1}^{M} \frac{n_j}{N} \Lambda_j(l,l) - 1 \right)^2 \tag{4.1}$$

$$G(H_{target}, \Lambda_{target}, \mathbf{b}_{target}; W_{source}) \coloneqq \|X - (HW^\top \Lambda + \mathbf{1}\mathbf{b}^\top)\|_F^2 \tag{4.2}$$

Wang et al. (2019a) propose SC analysis via expression recovery harnessing external

SCRNA-Seq data (SAVER-X) to attain data denoising via TL by coupling a deep autoencoder with a Bayesian model, and find SAVER-X markedly improves the accuracy of cell-type identification. The external data is borrowed from public domain data (e.g., $500K$ human immunocytes and $200K$ peripheral blood mononuclear cells) on which a deep autoencoder is trained, and then the weights of the model is retrained on the target domain data for knowledge transfer. The adaptive retention of transferable features across datasets is achievable by this two-step method. The denoised data is obtained by replacing the prediction of poorly fitted genes with the mean of their target data.

Lieberman et al. (2018) propose classification of SCs by TL (CaSTLe) based on a series of univariate feature selection methods and extreme gradient boosting (XGBoost). However, affected by the number of selected features (genes), the performance is not robust to noise as demonstrated by Lieberman et al.. Therefore, to achieve optimal performance, this approach requires rigorous fine tuning to select the feature count. Wang et al. (2019b) propose batch effect removal using deep autoencoders (BERMUDA) which use TL to do batch correction (i.e., the procedure of removing variability, caused by technical differences, from the data). The primary idea requires utilising the similarities between the cell clusters to align cell populations from batches (e.g., the type of sequencing machine). Mieth et al. (2019) propose using prior knowledge from TL can improve the unsupervised cluster of SCRNA-Seq data through autoencoders. The knowledge is transferred from a well annotated dataset to an unlabelled data. However both of them do not investigate the feasibility of knowledge transference across species.

Our work addresses above practical situations which are not considered by previous authors. For example, we explore the situation where only a small amount of data (e.g., hundreds of samples instead of hundreds of thousands of samples) on the target domain is accessible, in which one of essential values of TL lies. Our work precisely explores small data regimes in the target domain in the setting of TL across species. Instead of using traditional matrix factorisation methods, We use Artificial Neural Networks (ANNs) to explore the possibility of knowledge transfer on three different tissues. The details of exploration are shown in the following sections.

4.2 Datasets

In order to validate the feasibility of knowledge transfer across species, we use data from three different tissues, BM, pancreas and mid-brain for both the human and mouse species, of which details are shown in Table 4.1. For reducing external effects, we select common classes from both species to implement TL. The following subsections show descriptions of each tissue.

TABLE 4.1: Datasets.

Tissues	Species	Data size (cells, genes)	Categories
BM	Mouse	(5503,4372)	14
	Human	(9393,4372)	11
Pancreas(Baron et al., 2016)	Mouse	(1416,10315)	10
	Human	(7568,10315)	10
Mid-Brain(La Manno et al., 2016)	Mouse	(1269,8579)	16
	Human	(1231,8579)	16

4.2.1 Bone Marrow

BM tissue, a spongy substances, is chosen as it manufactures stem cells which have the potential to differentiate into different blood cells. Each type of blood cells play an important role, such as red blood cells carry oxygen to tissues in the body, white blood cells fight infections and platelets stop bleeding by making blood clotted. As a tissue with a well-established physiology in mice that is broadly conserved, and yet only partially understood in humans, it raises our interest. As described by Stumpf et al. (2020), gene expression signatures of mice are collected from three different mice by using droplet-based SCRNA-Seq (Drop-Seq (Macosko et al., 2015)). Overall, $6,800$ SC transcriptomes are sequenced, yielding greater than 9×10^4 reads[1] per cell on average. Following pre-processing and filtering, a total of $5,503$ cells are retained, expressing on average $2,684$ transcripts per cell.

BM human samples are sequenced from three patients undergoing routine hip replacement surgery at Southampton General Hospital. In total, $25,000$ SC transcriptomes from the three patients are sequenced yielding on average 5×10^4 reads per cell. The data for $9,393$ cells expressing on average $3,070$ transcripts per cell is obtained. The sparsity of gene expression of both species is shown in Appendix B Figures B.3 and B.4.

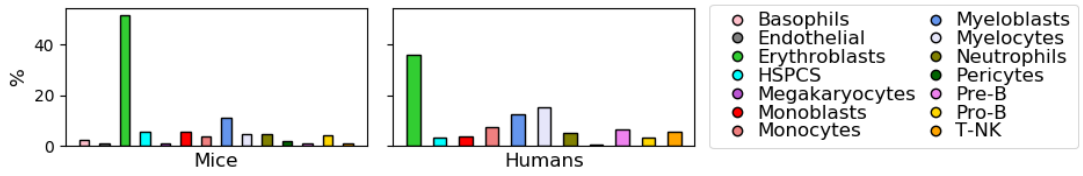


FIGURE 4.1: The distribution of cell types in mice and humans of the BM dataset. There are 14 cell types for mouse (left) and 11 types of human cells, shown using the same colours. TL work uses these 11 common cell types.

[1] A read is an inferred sequence of a fundamental unit of double-stranded nucleic acids consisting of two nucleobases bound to each other. Each unit is called a base.

The label of cell categories is obtained by unsupervised clustering using the Louvain method (Waltman and Van Eck, 2013) performed in the Seurat package to identify the various hematopoietic and niche-cell types. Then, we use the likelihood-ratio test (McDavid et al., 2013) to identify the differentially expressed genes to label the clusters given by the Louvain algorithm. In the likelihood-ratio test, we set prevalence $> 25\%$; fold-change > 2; and p value < 0.001, and screen the obtained cluster markers as biomarkers for given bone marrow cell populations. Figure 4.1 shows proportion of different cell types for both species.

4.2.2 Pancreas

For exploring the feasibility of knowledge transfer across species on other tissues, we select the pancreas tissue of which the dysfunction is clinically important in type 1 (T1D) and type 2 diabetes mellitus (T2D), pancreatitis, and cancer. As described by Baron et al. (2016), inDrop, a droplet-based SCRNA-Seq method is invoked to determine the transcriptomes of individual pancreatic cells. In total, four human donors and two strains of mice are included in the experiments and over 1.2×10^4 cells forming 15 clusters are obtained. The details of the above clusters can be found in (Baron et al., 2016) or the National Center for Biotechnology Information (NCBI) gene expression omnibus website with reference genomes (GSE84133). As shown in Figure 4.2, we select the first 10 common classes ordered by the amount of cells to obtain enough data within each class for the classification tasks.

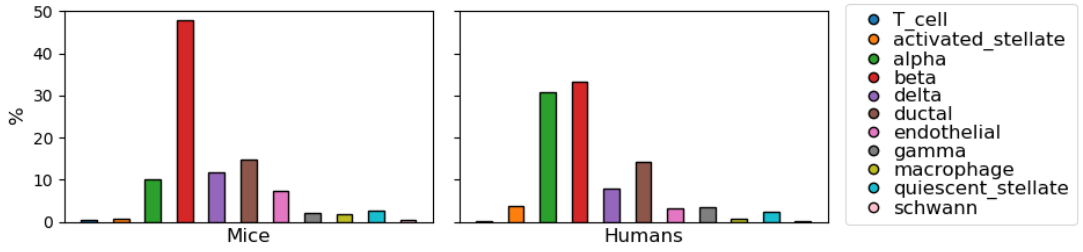


FIGURE 4.2: The proportion of cell types found in the pancreas of each species.

4.2.3 Mid-Brain

Mid-brain is another tissue which is associated with movement in auditory and visual processing. La Manno et al. (2016) define cell types of the ventral mid-brain in both humans and mice for a better molecular understanding of human mid-brain development. Coded by GSE76381 on NCBI, this SC data contains human embryo ventral mid-brain

cells between 6 and 11 weeks of gestation and mouse ventral mid-brain cells at six developmental stages between E11.5 to E18.5. We establish a 16-class cell recognition task and the distribution of classes is shown in Figure 4.3.

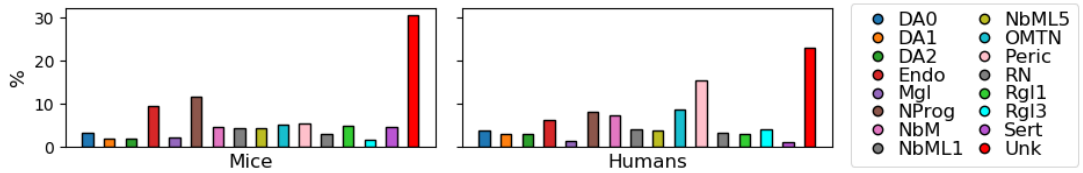


FIGURE 4.3: The proportion of cell types from the mid-brain in mice and humans.

4.3 Methodologies

We use two simple classifiers, the Support Vector Machine (SVM) and decision tree to recognise different cell types of BM SCRNA-Seq data from both mice and humans. We compare their performance to a simple ANN which will be used to complete TL in the later steps. After which, we directly test the mouse model on the corresponding human tissue datasets to investigate the transferability between the two species without retraining. Here, we refer to mice as the source domain and humans as the target domain. We use a small amount of data from the target domain to retrain a model trained on the source domain. Finally, we investigate the effects of progressively adding the data used in target domain during knowledge transfer. The details of experiments are as follows.

4.3.1 Feature Selection and Traditional Classifiers

In this part, we explore the learning ability of traditional classifiers on recognising cell types, including SVMs and decision trees. For each of them, we investigate the effects of reducing dimensionality of inputs compared to using unprocessed high-dimensional data.

SVMs classify samples by finding a hyperplane which can maximise the margin between classes. For this classifier, we utilise two ways of reducing the dimensionality of inputs: Principal Component Analysis (PCA) and gene selection. At the first way, we use PCA to map the high dimensional data to lower dimensional space, and then send the projected data to SVMs. Various number of PCA components are selected to observe the influence on performance. PCA is implemented using the *sklearn.decomposition* library with the

default setting[2] except "whiten=True"[3]. SVM is performed by the library *sklearn.svm*, with the default kernel (radial basis function). The second method of reducing dimensionality of inputs is selecting different number of genes from fewer to more, where genes are sorted by the variance of gene expression across cells.

Unlike SVMs, decision trees are based on a multistage/hierarchical decision scheme to perform classification. The tree is composed of a root node, a set of internal nodes and a set of terminal nodes (leaves). Each node of a tree makes a binary decision to separate classes. In the experiments, we apply dimension reduction of input data by PCA, compared to data with full dimensions. To perform decision trees, we use library *sklearn.tree.DecisionTreeClassifier* with the default criterion (gini) to measure the quality of each spilt. The depth of the decision tree is 10, which is decided based on all leaves being pure.

4.3.2 Artificial Neural Networks

Although traditional classifiers can achieve competitive performance, it is not easy to transfer knowledge based on them. Hence we also explore the performance of ANNs on classifying cell types as the foundation of TL, and also compare it to traditional classifiers.

We use single hidden layer neural networks with various number of neurons to recognise the cell types. We investigate the uncertainty of performance using the stratified k-fold method to spilt the training and test datasets so that both have the same distribution. The single layer neural networks are trained for 30 epochs and optimised by Root Mean Square Propagation (RMSprop)(Tieleman and Hinton, 2012) with a learning rate 0.001, performed by the *keras* library. For investigating the effects of regularisation, we also compare the performance of model using $L2$ regularisation with factor 0.001 and without regularisation. Unless otherwise specified, the setting of networks is the same.

Up to now, all the experiments have been implemented based on the data with all gene expressions. However, a great number of genes are rarely expressed in the majority of the cells as shown in Figure B.2. Therefore, we further investigate the effects of reducing utilised genes on prediction performance. We sort the genes by the variance of expressions over cells from big to small, and then progressively reduce genes, eliminating

[2]There are two essential parameters, gamma and C. Gamma is a positive parameter to define how much influence a single input has. A larger gamma results in effects on closer other inputs. The value of gamma is set as "scaled", and the value equals to $1/(the\_number\_of\_features * the\_variance\_of\_inputs)$. C is used to trade-off the balance between the misclassification of data and the simplicity of the decision surface. Lower C results in smoother surface.

[3]We set it as *True* for improving the predictive accuracy of the downstream estimators as whitening can remove part of information from the transformed signal, namely reducing the relative variance scales of the components. The components vectors are multiplied by the square root of n samples and then divided by the singular values to ensure uncorrelated outputs with unit component-wise variances.

by the smallest variance. For comparison, we also randomly reduce the same proportion of genes and then classify the cells using the remaining gene expression data.

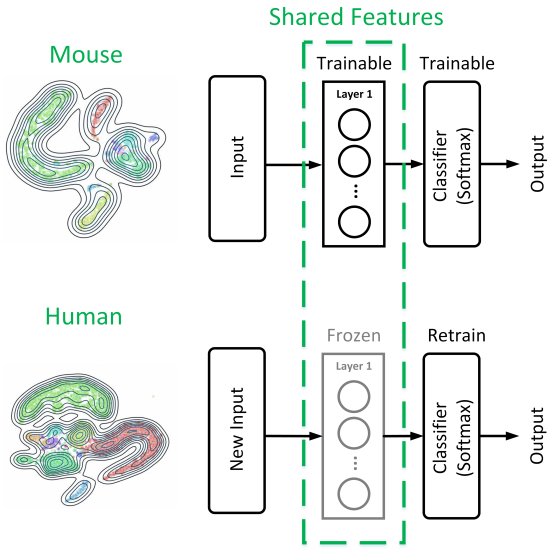


FIGURE 4.4: Schematic diagram of TL from mouse to human. Weights the network trained on mouse data, apart from those of the final layer, are frozen and the final layer retrained using human data. Thus, the input to hidden layer transformation (or mapping) is being transferred from mice to humans. Input distributions are shown using Gaussian kernel densities applied to TSNE projections.

4.3.3 Transfer Learning

After validating the feasibility of using neural networks to recognise cell types, we further investigate the possibility of transferring knowledge between species (from mice to humans). Figure 4.4 shows the training scheme of TL. Firstly, a single hidden layer neural network is trained on the (mouse) source domain data. Then, the weights of its hidden units are frozen while the output layer, with a softmax activation function, is retrained on the (human) target domain data. As described in the Section 2.4.2, we use a trained model as a feature extractor to validate the feasibility of knowledge transfer across species, without fine tuning of weights of hidden layers. In the Chapters 5 and 6, we further explore the case of fine tuning weights of hidden layers after observing the transferability of extracted features. From experiments on the source domain, we take the network with 18 neurons to transfer knowledge, as this model has sufficient learning ability to classify the cells on the source domain. We also explore the effects of reducing the amount of used genes on TL and the transferability affected by the amount of data used on the target domain to retrain the model.

Table 4.2 summarises all the scenarios explored in this chapter. In scenarios 1, 2 *and* 3, we randomly apply an 80 : 20 spilt on the data to create the training set and testing datasets. In scenario 4, the human data is split into two equal parts when producing

the train and test datasets independent and identically distributed (i.i.d). 50% of the human train data is spilt by a 20-fold stratified Cross Validation (CV). We then randomly select increasing number of data (more folds are contained) as the set used to retrain the model. The remaining 50% of the human test data is used as a fixed set to evaluate models in scenarios 4, 5, *and* 6. All experiments are evaluated over multiple runs to explore the uncertainty. Considering the imbalance in distribution of categories, we use both accuracy and weighted F_1 score (which considers both precision and sensitivity)[4] to evaluate the performance.

TABLE 4.2: The scenarios applied into exploring the transferability between mice and humans. In scenarios 5 *and* 6, the mixed data consists of both mouse and human data. The details of the mixed data is described in Figure B.5.

Scenarios	*1*	*2*	*3*	*4*	*5*	*6*
Train	Mouse	Human	Mouse	Mouse	Mixed	Mixed
Retrain	–	–	–	Human	–	Human
Test	Mouse	Human	Human	Human	Human	Human

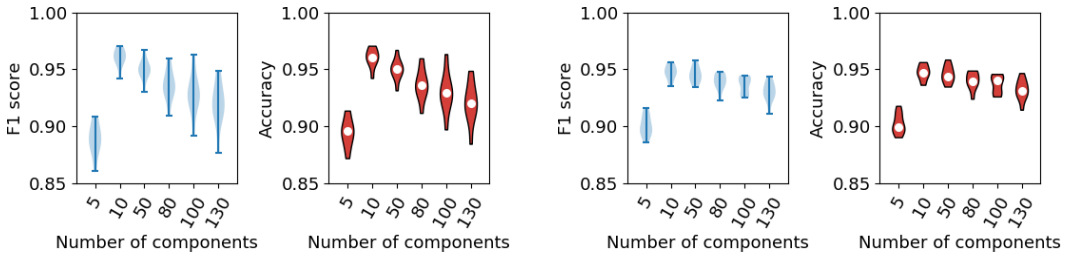
4.4 Results

In this section, we show and analyse results on three tissues, BM, pancreas and mid-brain. After comparing the learning ability of neural networks to traditional Machine Learning (ML) methods, we notice, both of them can provide satisfactory performance on the same dataset. To investigate the feasibility of knowledge transfer, we further implement TL based on neural networks with respect to growing amounts of data on the target domain.

4.4.1 Support Vector Machines

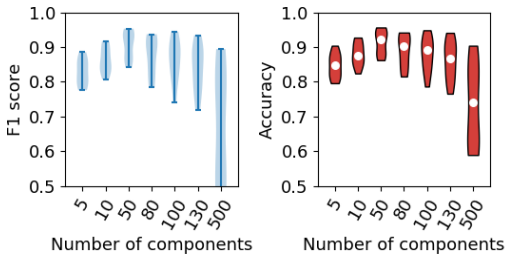
As shown in Figure 4.5, SVMs combined with PCA, show compelling learning ability on both mouse (scenario 1) and human (scenario 2) data of BM and pancreas tissues. The results of the developing mid-brain tissue are in the range 0.4 to 0.6, which may indicate the developing tissues are hard to recognise due to the variable absence of gene expressions in different developing stages. Moreover, the performance is affected by the number of PCA components for all three tissues. After increasing the number of PCA

[4] F_1 score is a metric to evaluate the performance as equation: $F_1 = 2 \times \frac{precision \times recall}{precision + recall}$, where $recall = \frac{True\ Positive\ (TP)}{TP + False\ Negative\ (FN)}$ and $precision = \frac{TP}{TP + False\ Positive\ (FP)}$.
Weighted F_1 scores calculates the F_1 scores for each class independently and then adds them together using a weight depending on the number of true labels of each class.

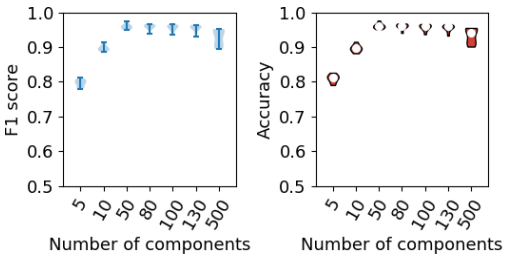


(a) BM of mice

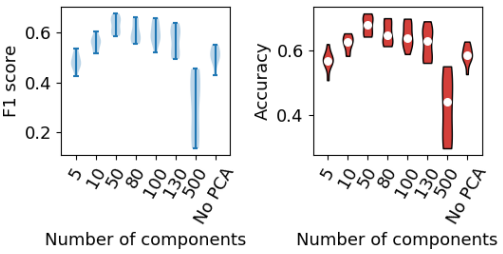
(b) BM of humans



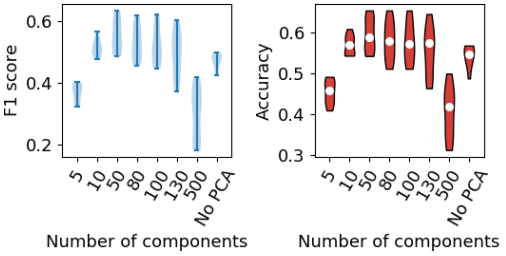
(c) Pancreas of mice



(d) Pancreas of humans



(e) Mid-brain of mice



(f) Mid-brain of humans

FIGURE 4.5: The performance of PCA plus SVM on three tissues (BM, pancreas and mod-brain), evaluated by weighted F1 score and accuracy. The first and second columns provide results on mice and human, respectively. The performance is investigated with respect to the growing number of components given by PCA. "No PCA" means data without PCA is used as inputs of classifiers. Uncertainties in results are evaluated by ten-fold CV.

components, we notice the relationship between performance and the number of components is nonlinear. For example, a SVM implemented on 10 components shows optimal performance for the BM tissue. For pancreas and mid-brain tissues, 50 components are optimal. We also find a consistent trend on both evaluations, F_1 score and accuracy. We also notice using features from all genes shows better performance than using 500 principal components on the mid-brain datasets, which is unexpected. After carefully checking the results, this may result from the developing of cells where different genes will be expressed in the different developing stages.

Figure 4.6 shows performance of SVMs using different number of genes on BM, pancreas and mid-brain tissues. We observe that the selection of different number of genes marginally affects the ability of recognising cell types, and there is no clear tendency

to indicate whether more genes contribute more to the performance. Compared to the results given by ANNs (same to results in Figure 4.9), SVMs show slightly inferiority for BM and pancreas tissues. For mid-brain, the differences between SVMs and ANNs reach 20%. In comparison to PCA plus SVMs (shown in Figure 4.5), training SVMs on selected genes may cause decrements of performance in recognising cells. As the selection of parameters (gamma and C) of SVMs may cause significant differences in performance, we also train several SVMs by searching gamma from a list, $[0.05, 0.1, 0.5, 1, 10, 100, 1000]$, and C from the list, $[0.01, 0.05, 0.1, 0.15, 0.5, 1, 10, 100]$. As a result, SVMs with default setting described in Section 4.3 provide significant performance. Although in some cases, the performance of SVMs can be further improved with carefully fine tuning of parameters, this is not the point we want to chase in this work.

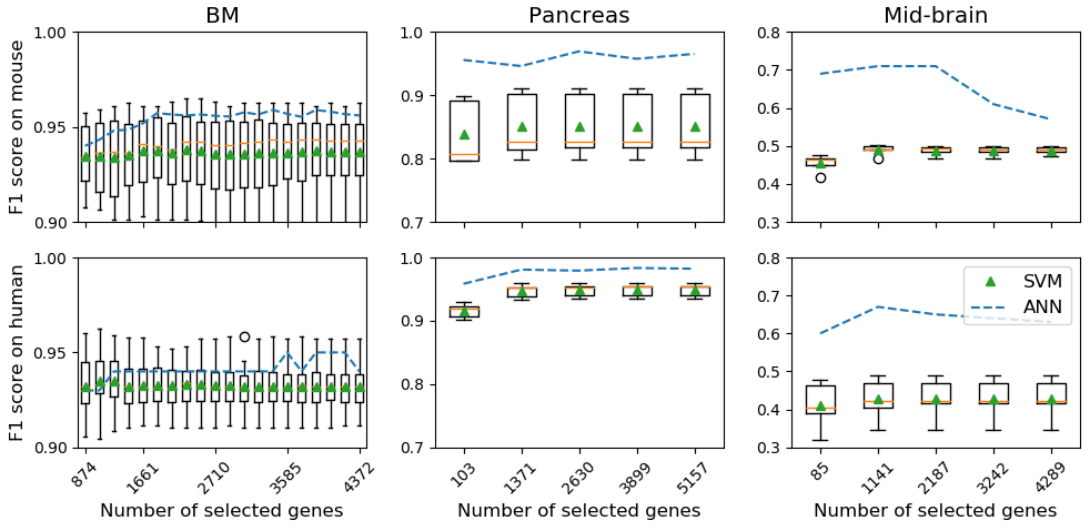
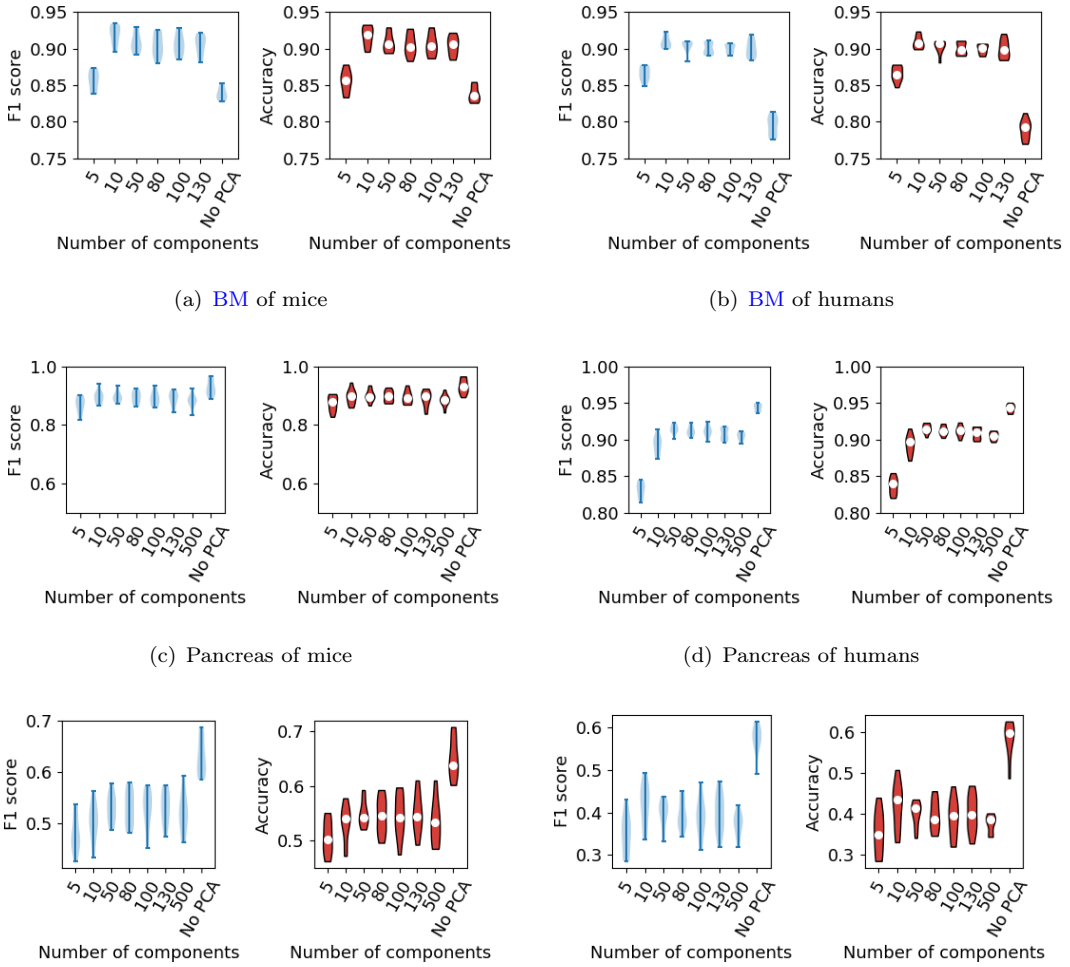


FIGURE 4.6: The comparison between SVMs and ANNs over an increasing number of selected genes. Each column shows performance of classification measured by F_1 score on one dataset. The top line shows results on mice, and the bottom line is for humans. The green triangle is the mean over ten-fold CV using SVMs and the blue dashed line represents the mean results given by ANNs.

4.4.2 Principal Component Analysis and Decision Trees

Figure 4.7 shows performance of a decision tree associated with PCA based on the same scenarios of Figure 4.5. All performance are measured by both F_1 score and accuracy. In this case, the best performing decision tree is inferior to the best performing SVM (see Figure 4.5). However, compared to the SVM, the decision tree shows relatively more stable learning performance with respect to the number of PCA components. We also directly apply a decision tree to the data using all genes (i.e., no PCA), however find unsatisfactory results on the BM tissue which may be caused by the over-learning of noise. For the pancreas and mid-brain tissues, using all genes improves the performance

of decision trees.



(a) BM of mice

(b) BM of humans

(c) Pancreas of mice

(d) Pancreas of humans

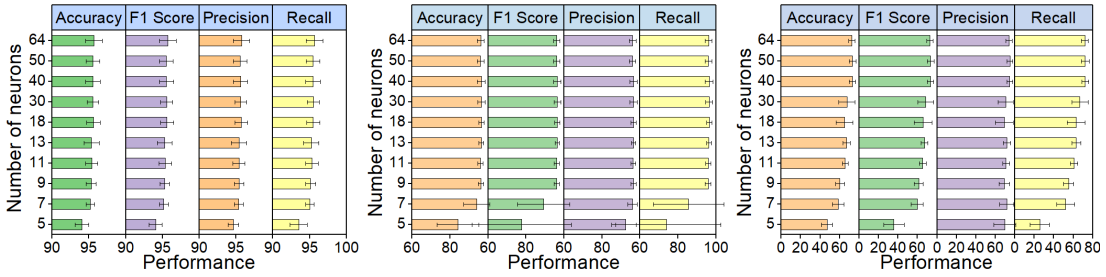
(e) Mid-brain of mice

(f) Mid-brain of humans

FIGURE 4.7: The performance of decision tree with PCA transformation of three tissues from mice (the first column) and humans (the second column). The ability of recognising cell types is evaluated by both F1 score and accuracy due to the imbalance data distribution. "No PCA" means training decision tree using all genes.

4.4.3 Artificial Neural Networks

As ANNs have repeatedly shown to be powerful learning architectures, we consider varying sized neural networks on the data using all the genes. As shown in Figure 4.8(a), the performance of single layer neural networks increases with respect to the number of neurons of the hidden layer and saturates at 18 neurons on BM and pancreas data (40 neurons on mid-brain data). We also try the case with adding $L2$ regularisation with a factor of 0.001 in training models. For the BM tissue, adding regularisation drops performance from around 95% to 90% measured by accuracy and F_1 score where recall and precision are around 85% and 98%, respectively. For the pancreas tissue, the



(a) BM (b) Pancreas (c) Mid-brain

FIGURE 4.8: The performance of single hidden layer neural networks on mice for BM, pancreas and mid-brain tissues. All models are trained without adding regularisation, and the size of each model is gradually increased with adding more neurons to the single hidden layer.

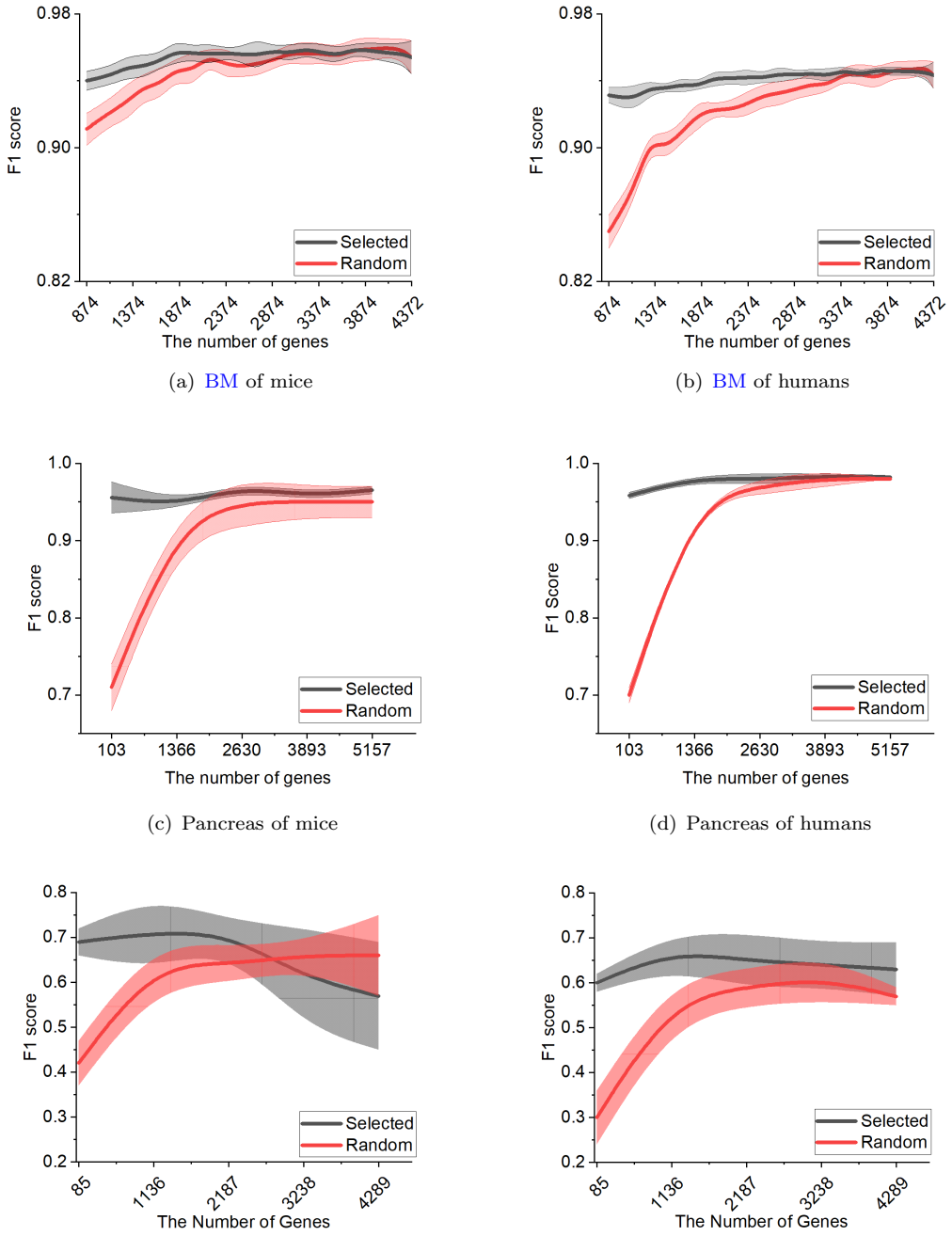
performance of models trained by adding the regularisation is in the range from 80% to 90% measured by accuracy (from 70% to 80% measured by F_1 score, recall is less than 70%, and precision is round 98%). The decrements is about 10% in comparison to no regularisation in training. Similarly, the accuracy of classification drops about 20% (accuracy is around 40%), when the regularisation is added, on the mid-brain tissue.

4.4.3.1 The Effects of Gene Selection

As the SC data is sparse, we are also interested in how many genes are essential for recognising cell types. Figure 4.9 shows the performance of a single hidden layer neural network (with 18 neurons for BM and pancreas datasets, with 40 neurons for mid-brain datasets) over the progressively increasing number of genes. Limited genes, selected by sorted variance, contain better knowledge in recognising BM and pancreas cells, in comparison to randomly selected. This dominance gradually decreases with an increasing number of used genes. There are around 20% of genes that show significant importance in recognising cell types, including gene '*ENSMUSG00000039959*' as an example of the BM tissue. However, for the developing mid-brain data, there is no monotonic trend. Furthermore, the optimal number of used genes is divergent between species. Therefore, sorting genes simply by the variance of expression over cells is a reasonable but not optimal way of selecting the relative essential genes for all tissues. A suitable schema used to select the genes still needs further investigation, however is not our current objective. For reducing the influence of uncertain factors, we utilise all available common genes between species to implement the TL in the following session.

4.4.4 Transfer Learning

After validating the feasibility of using neural networks to recognise cell types on SC data, we further show the transferability of features extracted by neural networks across



(a) BM of mice

(b) BM of humans

(c) Pancreas of mice

(d) Pancreas of humans

(e) Mid-brain of mice

(f) Mid-brain of humans

FIGURE 4.9: The influences of gene selection on recognising cell types of BM, pancreas and mid-brain tissues for both mice (the first column) and humans (the second column). The genes are selected in two modes, selected by the variance of gene expression (red line) and randomly selected (grey line). The shaded areas show the uncertain range of 10-fold CV.

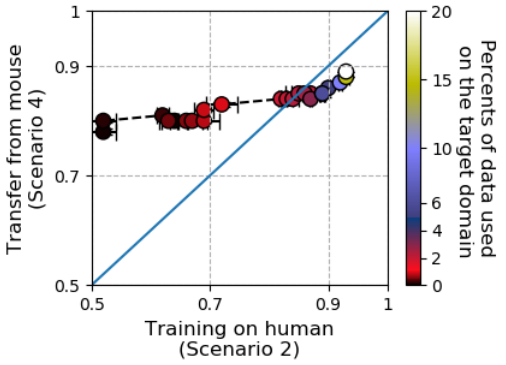
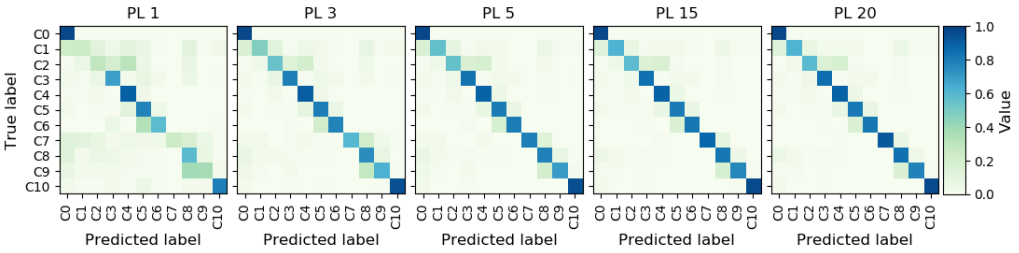


FIGURE 4.10: The performance of TL from mouse to human on BM dataset. 0 percent stands for the direct test on the data in the target domain following scenario 3 shown in Table 4.2. It shows comparison between direct learning (scenario 2) and TL (scenario 4). TL shows significant superiority of performance when there is limited available data on the target domain. Figure 4.13(a) shows the influence of initialisation on the classifier.

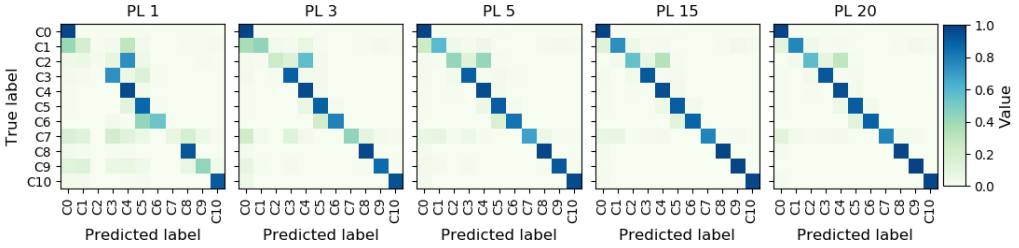
species in this section. Unless otherwise specified, we utilise all available common genes across the two species in the following experiments.

Figure 4.10 presents the comparison of performance ($F1$ score) between direct training on human data (scenario 2) and transfer from mouse data (scenario 4 from Table 4.2). An increasing amount of data is used to retrain the model on the target domain. The result suggests using a small amount of data to retrain the model is helpful to improve the performance on the target domain compared to directly testing (scenario 3). The performance of TL increases monotonically over the growing proportions of data used to retrain the model, saturating at around 20% of the data. The performance of TL can reach around 90% which is significant in supporting that knowledge of SC data being transferable across species. Details of this performance measured by other metrics are shown in Figure B.6.

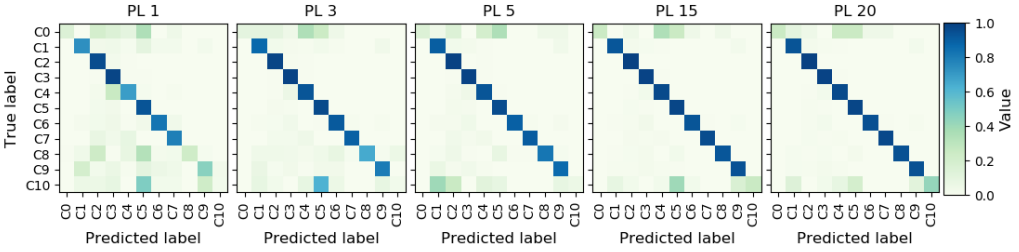
Furthermore, TL significantly outperforms direct learning on the target domain (scenario 2) as shown in Figure 4.10. When a small amount of data (less than approximately 2.5%) is available, TL shows more than 20% improvements in recognising cells for the BM tissue. Similar superiority in classifying cells can also be found in pancreas and mid-brain tissues, when the target data is limited. Figures 4.11 and 4.12 show the breakdown details of each class's transferability based on confusion matrices. For the BM tissue, Figure 4.11(a) shows confusion matrices of TL, and Figure 4.11(b) shows confusion matrices on direct recognising BM human cells, with models trained on a growing amount of data. The comparison between them further reveals that directly learning has poorer learning ability on limited data than TL. For example, knowledge of some cell types, such as Erythroblasts, Monocytes, Myeloblasts, Myelocytes, Neutrophils and T-NK, have high transferability even with limited target domain data (such as PL1 and PL2). Considering the imbalance of cell type distributions as shown in Figure 4.1, we analyse whether the



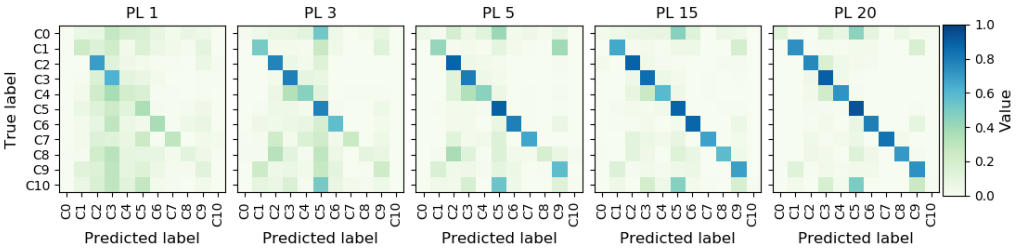
(a) TL from mouse BM datasets (Scenario 4)



(b) Training on human BM datasets (Scenario 2)



(c) TL from mouse (Scenario 4)

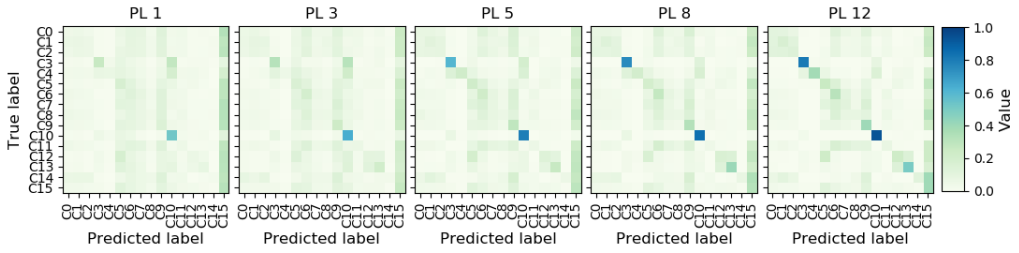


(d) Training on human pancreas datasets (Scenario 2)

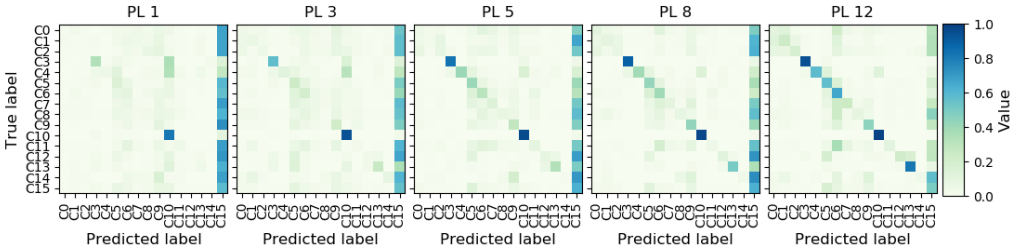
FIGURE 4.11: The normalised confusion matrices on the growing number of human data for both TL ((a) and (c)) and direct learning ((b) and (d)). For both BM and pancreas tissues, a growing percentage of human data is selected to train the model, where PLx means the increasing levels of percentage from 1 to 20. The model is tested on fixed percent of unseen human data. At (a) and (b), "C0-C10" corresponds to cell types of the BM tissues: Erythroblasts, HSPCs, Monoblasts, Monocytes, Myeloblasts, Myelocytes, Neutrophils, Pericytes, Pre-B, Pro-B, and T-NK. At (c) and (d), the axis values "C0-C10" correspond to the names of the classes from the pancreas tissue: T-cell, Activated Stellate, Alpha, Beta, Delta, Ductal, Endothelial, Gamma, Macrophage, Quiescent Stellate, Schwann.

classes with higher weight will be recognised easier. In short, we eliminate this potential hazard as features from Monocytes, Myeloblasts, Neutrophils and T-NK cells have high transferability even though they have a relatively small percentage of data. Compared to direct learning (scenario 2) without knowledge transfer (see Figure 4.11(b)), Figure 4.11(a) shows features transferred from mouse are obviously helpful towards reducing the confusion between Monoblasts and Myeloblasts cells.

In line with results on BM, the superiority of learning ability of TL on small amount of pancreas data can be observed by comparing Figure 4.11(c) with Figure 4.11(d). Figures 4.12(a) and 4.12(b) show the consequence of recognising mid-brain cells by TL and direct learning, respectively. As recognising developing cells (mid-brain tissues) is much more difficult than recognising mature cells (BM and pancreas tissues), both of the learning methods do not provide satisfactory performance. However, TL still reduces confusion of classification on limited target data, especially for Unk cells.



(a) TL from mid-brain of mouse (Scenario 4)



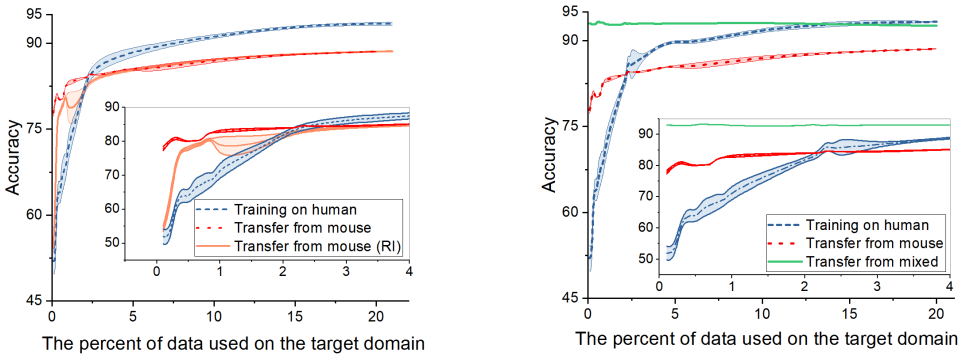
(b) Training on mid-brain of human (Scenario 2)

FIGURE 4.12: The comparison of confusion matrices of TL and direct learning on the growing percentage of human mid-brain data. (a) and (b) are performance of TL from mouse data and directly training on human data, respectively. The axis values "C0-C15" correspond to classes: DA0, DA1, DA2, Endo, Mgl, NProg, NbM, NbML1, NbML5, OMTN, Peric, RN, Rgl1, Rgl3, Sert, Unk.

4.5 Discussion

We have validated the transferability between species on the three tissues, we also have interest in what kinds of effects may cause transferability drop, e.g., initialisation of classifiers, the distributions of samples, the size of the model and regularisation. We take the BM tissue as an example to do further exploration regards to above elements.

Considering the initialisation of classifiers, we compare the effects of randomly initialising classifiers and using pre-trained weights as the initial values. As shown in Figure 4.13, the red dashed line in Figure 4.13(a) shows the results of TL based on random initialisation (shown in Figure 4.4), and the original solid line in Figure 4.13(b) shows the performance of the classifier using pre-trained weights from the source domain. Comparing each illustrates that the pre-trained classifier show better performance than the random initialised classifier when target domain data is limited (less than around 2%). Both TL cases outperform direct training (the blue dashed line) on the target domain when limited data is available.



(a) (b)

FIGURE 4.13: The performance comparison between TL and direct learning on the progressively increasing BM data on the target domain. In (a), the classifiers are initialised by two ways, random (the blue dashed line and orange solid line) and pre-trained (red dashed line). The pre-trained classifier improves the performance of TL when there rare data is available on the target domain, and outperforms direct learning. (b) shows transferability of features from the model being trained on the mixed data. Compared to being directly trained on the target domain, TL from mixed data shows significant improvements.

We further explore if transferability is improved when the model has seen parts of mixed data (scenarios 5 *and* 6) during training on the source domain. The exploration of transferring knowledge from mixed data is inspired by a realistic situation in the natural language processing area[5]. In this use case, the company needed a model to recognise the English based on the voice of the speaker, however, there are various types of English accents and dialects (Winata et al., 2020). Training a model from scratch is too expensive, but also, English with different accents and dialects is always lacking data to train the big model. Hence, the model is trained on the standard English to reduce training costs. Similarly, impacted by the operations of experiments, patients and ethnic variations, the prediction of cell types is in a similar situation. Therefore, we give preliminary results on exploring the transferability of features extracted from mixed data. The corresponding results are shown in Figure 4.13(b) as the yellow dashed line and the green dashed

[5]As described in the website https://emerj.com/partner-content/crowdsourced-natural-language-speech-training-use-cases-explanation/

line. We find significant improvements brought by mixed data on the source domain when limited data is available on the target domain in comparison to other scenarios. However, when there is an abundance of target domain data, the improvements are negligible compared to direct learning on the target domain (scenario 2).

Considering the imbalance in the distribution of classes, we investigate the situation that all classes are equally sampled from the train data on the source domain, and others kept the same. Consequently, we observe no significant change in the performance of TL. Observing the affect of model size, we find that networks with more neurons do not show better performance and in some cases show slightly worse transferability. Taking regularisation into account, we also notice regularisation may be harmful to transferability. To validate this, we consider two types of regularisation: adding a dropout layer to the model and direct $L2$ regularisation. Both ways show considerable reduction of TL performance (from around 90% dropping to 60%). Based on this observation, we assume models working well in the source domain may extract too specific features to source domain data, so that the features will have poor transferability. To further investigate this point, we extend our work to the next chapter 5 where TL based on Cascade Learning (CL) is introduced. As an adaptive learning algorithm, CL shows the potential to progressively extract features from coarse to fine, layer by layer, where coarse features may have better transferability.

Moreover, we notice that the performance on the mid-brain data is not ideal (around 70%). We assume the developing cells (mid-brain) have more complex gene expression features, especially the same type of cells in the different developmental stages which may have completely different information for the gene expression. Kharchenko et al. (2014) and Sarkar and Stephens (2021) also refer to another situation called gene 'dropouts' in which a gene is observed at a moderate expression level in one cell but is not detected in another cell. Taking these potential risks into account, the relative poor results of classification on the mid-brain data are reasonable.

Although the performance of recognising developing cells need further improvements, TL can enhance the capability of learning when target domain data is limited. The consistent advantages are shown on three types of tissues (see Figures 4.11 and 4.12). Building on these promising results on the small regime data, we further explore TL in another two applications, Human Activity Recognition (HAR) and medical image classification, where data scarcity is a widely concerned issue. Similar observations are reported by us (see Chapters 5 and 6).

4.6 Summary

In this chapter, we have investigated the capacity of traditional methods (the SVM and decision tree) and neural networks on single cell classification tasks, where we notice

the comparable performance from all of them (neural networks sometimes show better results). We further validate the feasibility of knowledge transfer across species based on features from neural networks. From exploring three tissues, two mature tissues and one developing tissue, we demonstrate that knowledge can be transferred across species (from mouse to human). To be more precise, TL shows significant improvements when a limited number of training target data samples are accessible compared to directly learning without transferring knowledge. This discovery has important practical implications for exploring the characteristics of single cell human data for the reason that training a network from scratch would need amounts of data which is hard to collect.

Chapter 5

Transfer Learning from Cascade Learning for Human Activity Recognition

In previous chapters, we have introduced Cascade Learning (CL) as a computationally efficient method of training Deep Neural Networks (DNNs) and analysed its behaviour in comparison to End-to-End (E2E) training on the information plane. We also considered Transfer Learning (TL) applied to genomics problem in which it turned out a simple single hidden layer neural network gave sufficient performance. In this chapter, we continue to discuss TL on harder problems that require multiple layers of hidden units. In comparing cascade and E2E trained networks as source models, we seek to understand how features from different layers are useful in achieving significant performance in a target problem. Our observation is that a nature of CL is to extract coarse representations about a problem in early layers of a network and fine details in later layers. As such TL from early layers of a cascade trained model often performs better than transfer from later layers or from corresponding E2E trained models. In this chapter, we demonstrate this observation on Human Activity Recognition (HAR) problems and the next chapter present work on natural and medical images. The work reported in this chapter is published as a peer reviewed conference paper (Du et al., 2019).

5.1 Related Work

Recognising human activities from sensor-based measurements is a challenging and useful problem in Machine Learning (ML) with a wide range of potential applications, particularly related to personalised healthcare. Interest in this topic has grown significantly in recent years with increasing availability of cheap wearable sensors integrated into everyday devices such as smart phones. Remote monitoring of the elderly in homes (Kuo

et al., 2004) and early diagnosis of complex diseases (Milne et al., 2017) are examples of activity recognition applications.

In early work, the common way of solving HAR tasks including three stages: extracting features (e.g., Fourier or statistical features); prepossessing the extracted features; and using a classifier (e.g., Support Vector Machines (SVMs), Random Forest, Gaussian Mixture Models, k-Nearest neighbour and Hidden Markov Models) to recognise activities based on those features (Bulling et al., 2014; Calatroni et al., 2011; Lara and Labrador, 2013; Ordóñez et al., 2014; Roggen et al., 2015). However, this traditional pipeline usually encounters two issues: the extraction of hand-crafted features is limited by human domain knowledge (Wang et al., 2019c); and the performance of classification based on hand-crafted features can be inferior as various activities may need different features (Huynh and Schiele, 2005).

Advances in algorithm, much of the impressive development has been seen in recent years using neural networks and increasing the depth of networks is thought to enable the extraction of features that helps in creating accurate inferences. DNNs have also been applied to HAR by several authors (Hammerla et al., 2016; Ronao and Cho, 2016) taking advantage of their ability at carrying out feature extraction and classification simultaneously. As with a number of other problems, extracting relevant features automatically by training is seen as an advantage over the use of hand-crafted features as used in works (Bulling et al., 2014; Lara and Labrador, 2013; Ordóñez et al., 2014). The most popular Deep Learning (DL) approaches applied in HAR include Multi-layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs) (Ordóñez and Roggen, 2016), Recurrent Neural Networks (Edel and Köppe, 2016), and the Long Short-Term Memory (LSTM) (Guan and Plötz, 2017).

Although the above approaches show state-of-the-art generalisation performance, the computational complexity and memory requirements of DNNs is noted to be generally high (Chellapilla et al., 2006; He and Sun, 2015). These requirements limit the further application in HAR where one of the goals includes the necessity to integrate these networks on wearable/edge devices (e.g., smart phones and smart watches) for real-time detection (Nguyen et al., 2021), such as anticipating the users of certain incidents or monitoring diseases (Agarwal and Alam, 2020). For different users, the model may need to be trained on the device to fit the specificity of personal data. Hence, HAR requires compact models with lower computational cost solutions where CL provides natural advantages as aforementioned in Chapter 2. This is one motivation for the pursuit of the CL architecture in this work.

In addition to the need of compact models, data scarcity is another issue in HAR (Al Machot et al., 2020; Baxter et al., 2015). Collecting and annotating sensory activity data are expensive and time consuming, which make annotation scarcity be a remarkable challenge for sensor-based activity recognition. Data for some emergent or unexpected

activities is especially hard to obtain as training data, which is another factor of data scarcity. Furthermore, activity patterns are person-dependent, namely different users may have diverse activity styles which further exacerbates the annotated data shortage problem. As mentioned in Chapter 4, TL is one of the main ways to alleviate the data shortage problem. Knowledge transferring across different users is also a potential way in solving problems caused by person-dependent activity patterns in practical applications.

There have been several authors (Calatroni et al., 2011; Kurz et al., 2011; Morales and Roggen, 2016) who have considered TL on HAR problems, and Cook et al. (2013) provide a review of their work. TL combined with DNNs on HAR proposed by Morales and Roggen (2016) is the closest to our work. They use an eight-layer convolutional neural network consisting of 64 kernels (5×5) in each layer and a final LSTM layer of 128 cells. They copy and freeze the first few layers of source model and only train the later layers with random initialisation for implementing TL. This is a large network with a total of $986,257$ parameters and Graphics Processing Unit (GPU) support is necessary. In addition to requiring heavy computation, the results reported for multi-class TL classification tasks are in the region of 0.50 (a low F_1 score). However, the feed-forward cascade architecture we report in this work requires far fewer parameters ($49,224$) and could be run with Central Processing Unit (CPU) computing alone, achieving significantly higher transferability (see Section 5.4).

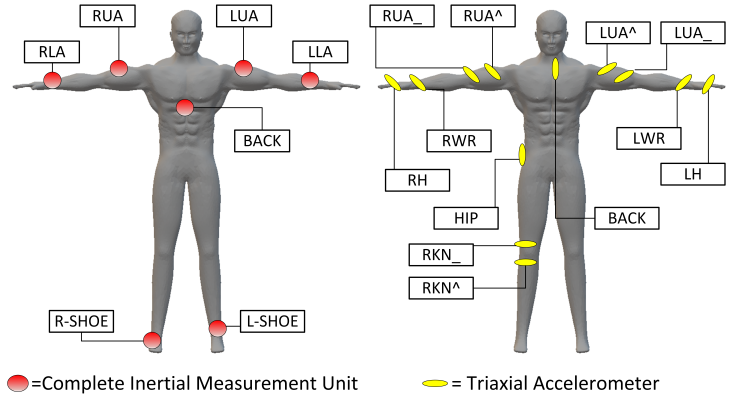


FIGURE 5.1: The on-body placement of sensors of the `Opportunity` dataset (Roggen et al., 2010).

5.2 Datasets

We use two benchmark datasets (`Opportunity` (Dua and Graff, 2017) and `Skoda` Mini checkpoint (`Skoda`) (Roggen and Zappi, 2015)) with different data acquisition protocols and the numbers of activity classes. `Opportunity` contains 18 activities performed by four subjects measured with wearable sensors, 3D accelerometers and Inertial Measurement Units (IMUs). Figure 5.1 shows locations of sensor placement on the body. The 18

activities in this dataset relate to behaviours in the kitchen such as opening and closing doors, and motions made during cleaning. `Skoda` is a dataset relating to quality control activities in a car production setting where 11 activities are contained such as closing of the vehicle's trunk and opening/closing of an engine hood. Sensors of `Skoda` consist of 20 3D accelerometers placed on both arms whose locations are shown in Figure 5.2. This dataset was sampled with frequency $98Hz$, which we down-sampled to $30Hz$ for consistency with `Opportunity`. In both cases, the three axes of each accelerometer are treated as separate channels.

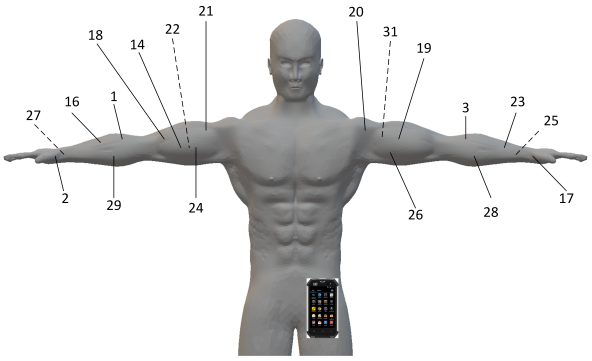


FIGURE 5.2: The on-body placement of sensors of the `Skoda` dataset (Stiefmeier et al., 2007).

5.3 Methodologies

We investigate TL from CL on HAR tasks by constructing three tasks:

- (Task 1) the performance comparison between a cascade and an E2E architecture on the same multi-class activity recognition tasks to establish that CL can achieve comparable performance to E2E training;
- (Task 2) a comparison of features' transferability across tasks within a given dataset where features are obtained from E2E or cascade trained models;
- (Task 3) transfer of learned features across datasets, again comparing features learned by the two different learning mechanisms. We repeat all experiments with five runs to assess the uncertainty with 400 training epochs in each run.

Performance is measured by micro F_1 scores as well as weighted F_1 scores[1] to account for imbalance across classes. The Adam optimiser is used in all experiments with a heuristically determined initial learning rate of 0.0001.

[1] Weighted F_1 scores calculates the F_1 scores for each class independently and then adds them together using a weight depending on the number of true labels of each class.
Micro F_1 scores uses the global TP, FN, and FP and calculates the F_1 score directly.

5.3.1 Task 1: Activity Classification with Cascade Learning

Empirically, we use MLP networks in this experiment. For the `Opportunity` dataset, networks have 5 layers with 25 units for each layer. For the `Skoda` dataset, MLPs have 128 input units and three hidden layers of 64, 64 and 32 units. The number of output units is decided by the number of classes. We use Hyperbolic Tangent Function (Tanh) activation function for hidden layer units and softmax at outputs. For the `Opportunity` dataset, we use all the IMU sensor measurements and for `Skoda` we use the sensors placed on the right arm (determined by activities). For `Opportunity`, we train the model by using the data from Activities of Daily Living (ADL)1, ADL2, ADL3 and drill session, and test the model using data from ADL4 and ADL5 (the same protocols as (Morales and Roggen, 2016)). For `Skoda`, we randomly take 20% of the data as testing sets and the remainder as training sets. Both training and testing data are normalised to $[0, 1]$ as in (Ordóñez and Roggen, 2016).

5.3.2 Task 2: Transfer Learning within an Activity Dataset

In this task, we establish three sub-tasks in the setting of TL. Firstly, we demonstrate the possibility of knowledge transfer across users based on CL. Then, we compare the transferability of features acting on different objects through binary and multi-class classification tasks respectively.

5.3.2.1 Multi-class Classification across Users

For comparison with Morales and Roggen (2016), we use the same evaluation method to explore the performance. The source data consists of data from subjects 1, 2 and 3 where the training data consists of all data from subject 1, and data from subjects 2 and 3 in ADL1, ADL2, ADL3 and drill session. The test data includes all the data from subjects 2 and 3 in ADL4 and ADL5. The target data is all the data from subject 4 where the train and test spilt are divided in the same way as the source domain (i.e. ADL4 and ADL5 are the test data as in (Morales and Roggen, 2016)).

5.3.2.2 Multiple Binary Classification Tasks

To compare the transferability of features from cascade *vs*. E2E trained models within a domain of different tasks, we use the `Opportunity` dataset which has open and close activities on different objects (e.g., fridges, doors, drawers and dishwashers). The underlying tasks have similarities in the required movement, but may have differences in required force and posture. There is a set of TL experiments in which the source domain model is trained for recognising differences between open and close on the same objects

(e.g., Door 1) and all open/close pairs on other objects are target domains to transfer. The transfer is implemented by taking features from each hidden layer of source models and training a classifier (layer) to fit the subsequent target domains. The details of these sub-tasks are summarised in Table 5.1. The MLP networks used for these sub-tasks consists of six layers with 25 hidden units in each. The method of dividing the train and test data remains the same as previous work (Morales and Roggen, 2016).

TABLE 5.1: Task 2: Summary of source and target domains used for experiments reported on task 2.

Subtasks	Source domain	Target domains
Multi-class across users	Subjects 1,2 and 3	Subject 4
Binary (open *vs.* close)	`Door1`	Door2
		Fridge
		Dishwasher
		Drawer1
		Drawer2
		Drawer3
Multi-class	14 classes (Open/close 7 objectives)[1]	4 classes (Open/close Type one Type Two)[2]

1: 7 objectives include: Door1, Door2, Fridge, Dishwasher, Drawer1, Drawer2 and Drawer3.

2: Type one and Type two are separated by the difference of hand movements. Type one: Doors and Fridge. Type Two: Dishwasher and Drawers.

5.3.2.3 Multi-class Classification Tasks

We further test transferability of features extracted from cascade and E2E trained networks in a more challenging multi-class setting. We build a 14−class problem with open and close on seven different objects (fridge, door 1, dishwasher etc.,) as the source problem and a four-class problem as the target problem as summarised in Table 5.1. On the target domain we identify the opening and closing of the Type one objects (doors and fridge) and the opening and closing of Type two objects (drawers and dishwasher) where Type one and Type two are grouped according to the similarity of hand movements in activities. This grouping is done to be consistent with the confusion matrices resulting in Task 1. Hence, the four target classes in Task 2 are: `Open Type One`, `Close Type One`, `Open Type Two` and `Close Type Two`. To clarify the source and target domains for all experiments done for Task 2, Table 5.1 shows the summary of all the source and target domains.

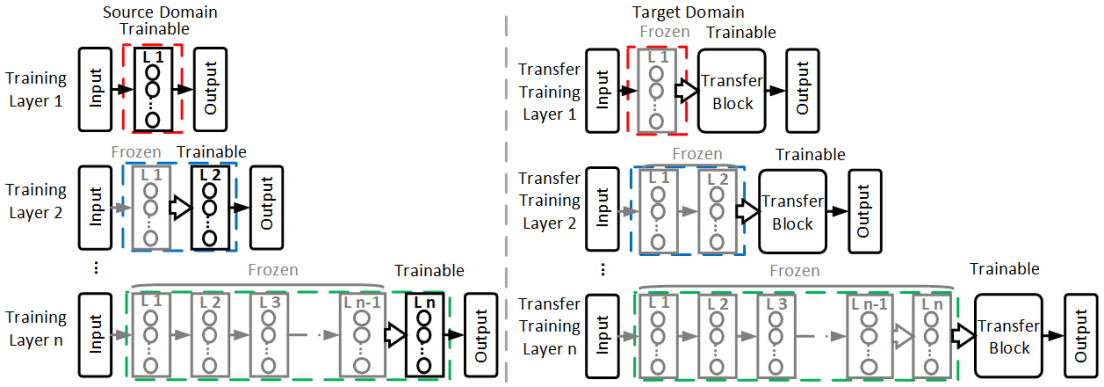


FIGURE 5.3: The scheme of TL in a CL setting. The layers in dashed boxes with the same colour shared by the source and target domain for transferring knowledge. All layers in grey are frozen. A transfer block is added to target domain models after the shared layers for fitting task domain tasks. The structures of transfer blocks are listed in Section 5.3.3. The classifier layer of each layer on the source domain is included in the output layers.

5.3.3 Task 3: Transfer Learning across Datasets

To investigate TL across two different domains of HAR, we use both the `Skoda` and `Opportunity` datasets to set up this task. Sensors from similar positions on the left arm of `Skoda` and the right arm of `Opportunity` are chosen as 18 dimensional inputs with open and close actions. The six selected accelerometers are located at positions $1, 2, 16, 21, 27, 29$ in Figure 5.2 for the `Skoda` data and two IMUs, RLA and RUA shown in Figure 5.1 for the `Opportunity`. The setting is transferring features from the `Skoda` to the `Opportunity` dataset in order to solve a binary classification problem.

We consider three different ways of changing the transfer block (Figure 5.3): (i) training a randomly initialised classifier; (ii) training a new hidden layer and a classifier layer with features transferred from the source problem; and (iii) using the trained weights of the source model as initial conditions and training the entire network on the target problem. Note, of these (i) and (iii) could be seen as differing only in the initial conditions of gradient descent training. Hence, we show all the results from the method (ii) for TL.

5.4 Results

5.4.1 Task 1: Activity Classification with Cascade Learning

Tables 5.2 and 5.3 show the performance comparison between E2E and CL on the `Opportunity` and `Skoda` multi-class classification problems respectively. On both tasks our E2E and CL models achieve comparable performance to results reported by Yang

et al. (2015) and Zeng et al. (2014). Although our simple neural network architecture shows lower performance than the huge DeepConvLSTM model (Ordóñez and Roggen, 2016) for HAR on `Opportunity` for classification, without considering TL, performance is not the focus of this part *per se*. Instead, we are interested in cascade trained networks for TL and the performance on a target problem.

TABLE 5.2: Task 1: Classification performance of Cascade and E2E learning on the 18-class `Opportunity` Dataset. LX means the X_{th} layer from the network. The bold text show the best performing case. For the remaining tables, the same colour coding and layer notation is used. Due to the imbalance among classes (including the null class with the majority), the micro F_1 score shows higher performance.

Model	Micro F_1 score (%)	Weighted F_1 score (%)
CL L1	86.52±0.31	84.16±0.36
CL L2	**86.88±0.30**	85.14±0.39
CL L3	86.78±0.23	85.36±0.40
CL L4	86.76±0.38	85.46±0.48
CL L5	86.72±0.31	**85.50±0.47**
E2E	86.32±0.67	85.08±0.55

TABLE 5.3: Task 1: Classification Performance of CL and E2E Learning on the test `Skoda` Dataset. In this table, we include two situations, including null class (11-class) and no null class (10-class). As the null class results in an imbalanced distribution, we compare the weighted and micro F_1 scores on 11-class task where weighted F_1 shows slightly worse performance. Weighted F_1 score counters the imbalance issues and gives more reasonable overall results.

Evaluation	CL L1	CL L2	CL L3	CL L4	E2E
Weighted F_1 score (%) (No Null Class)	80.20±1.3	85.06±1.4	85.82±1.8	**86.40±1.3**	85.66±1.3
Micro F_1 score (%) (No Null Class)	81.06±1.2	85.24±1.3	86.18±1.3	**86.44±1.3**	85.96±1.2
Weighted F_1 score (%)	71.34±2.1	77.70±1.6	79.08±1.5	**79.54±1.5**	78.36±1.0
Micro F_1 score (%)	72.70±1.8	78.40±1.5	79.66±1.4	**79.98±1.3**	79.14±1.0

In consideration of classifying human activities, our architecture achieves $86.88 \pm 0.30\%$ in comparison to DeepConvLSTM which achieves a 91.5% micro F_1 score. However, considering the number of parameters, our simple architecture only requires 49224, whereas DeepConvLSTM requires 999122[2]. We also have significant savings in the field of training time, requiring 1 to 2 seconds per epoch with a CPU in contrast to DeepConvLSTM which requires approximate 3 seconds per epoch with GPU. According to results shown in Tables 5.2 and 5.3, we further note that CL shows a progressive increase in perfor-

[2] $996800 + (128 * 18) + 18$

mance as additional layers are included, and the corresponding performance outperforms E2E learning on both datasets under the same configuration of training models.

TABLE 5.4: Task 2: TL performance across users of CL and E2E learning on the 18-class `Opportunity` Dataset.

Model	Micro F_1 score (%)	Weighted F_1 score (%)
CL L1	**84.18±0.33**	**80.70±0.37**
CL L2	83.56±0.67	80.16±0.54
CL L3	82.52±0.52	79.06±0.53
CL L4	82.14±0.63	78.42±0.63
CL L5	81.76±0.62	77.70±0.41
CL L6	81.34±0.70	77.10±0.58
E2E L1	80.90±0.46	76.24±0.44
E2E L2	81.44±0.64	77.12±0.48
E2E L3	82.80±0.43	78.82±0.33
E2E L4	82.92±0.52	79.26±0.46
E2E L5	82.64±0.75	79.08±0.92
E2E L6	81.68±0.52	77.58±0.51

5.4.2 Task 2: Transfer Learning Within a Dataset

For TL in this and following subsections, a single hidden layer and a classification layer are added to the extracted features and trained using target domain data. The results are reported in three parts.

5.4.2.1 Multi-class Classification across Users

In comparison to TL across users within `Opportunity` shown in (Morales and Roggen, 2016), Table 5.4 shows corresponding performance of TL based on CL. For consistency in results presented in our work, Table 5.4 displays results obtained by using data from the IMU sensors. Additional experiments are built by using 15 channels from accelerometers being the same sensor set-up as (Morales and Roggen, 2016). Using the exact same set-up as (Morales and Roggen, 2016), our results achieve a performance of 75 - 78% measured by micro F_1 score, which significantly outperforms the best results (around 60% micro F_1 score) presented by Morales and Roggen (2016), (the third column in Figure 3 in Morales and Roggen (2016)). As shown in Table 5.4, we notice the best performance of TL for the cascade architecture is approximately 4% higher than TL for the E2E architecture. We also learn that the transferability decreases with respect to the depth of CL. Worse performance is given when we transfer later layers of the CL network in contrast to the E2E network of which the performance first increases with subsequent layers, but then decreases if we add transfer too many layers.

TABLE 5.5: Task 2: TL performance of binary classification within the `Opportunity` dataset using CL and E2E learning. The performance is evaluated using a weighted F_1 score (%).

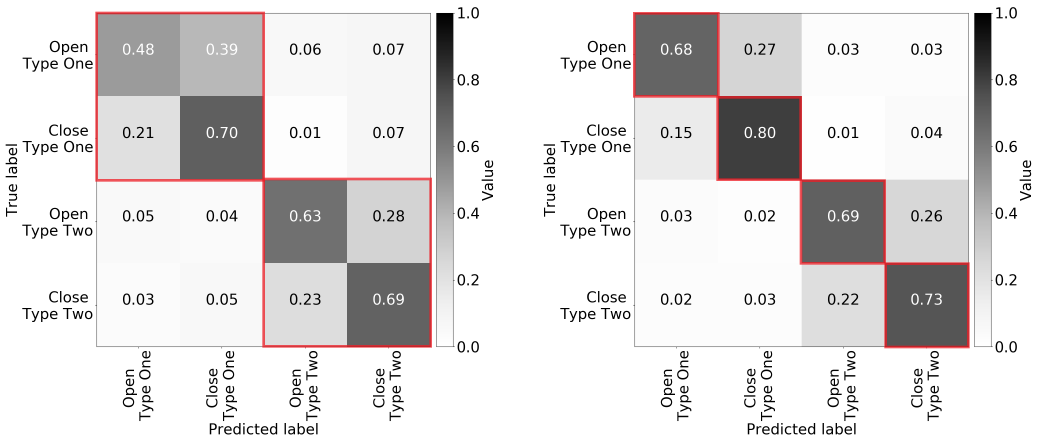
Model	Source: D1	Target: D2	Fridge	DW	Dr1	Dr2	Dr3
CL L1	73.04±1.4	**71.20±1.7**	71.84±1.0	**63.18±3.0**	67.78±4.1	**59.14±1.6**	**65.62±2.9**
CL L2	73.44±1.5	54.60±3.7	63.56±2.5	56.22±1.5	60.40±1.4	55.38±1.0	49.38±2.3
CL L3	73.42±1.6	49.56±6.3	61.26±0.8	58.28±1.8	56.26±2.6	56.40±2.5	51.40±1.4
CL L4	73.44±1.5	42.70±6.9	49.44±4.3	50.26±8.1	34.58±4.5	47.46±4.0	50.16±1.6
CL L5	73.42±1.5	41.32±6.8	45.34±4.5	45.46±7.5	32.38±3.5	46.76±3.5	48.34±2.8
CL L6	73.42±1.5	39.56±4.7	43.98±1.3	43.14±5.0	31.62±1.7	45.52±0.7	46.34±5.1
E2E L1	–	64.64±1.3	**76.50±1.4**	60.30±6.4	**69.34±2.7**	54.16±2.3	56.48±1.4
E2E L2	–	65.92±1.9	72.68±4.8	55.20±8.1	66.36±4.0	55.12±4.0	55.64±2.4
E2E L3	–	64.64±3.7	71.44±4.9	55.16±4.9	63.48±6.2	54.32±3.8	51.82±3.1
E2E L4	–	63.36±4.1	68.44±5.5	54.38±4.8	62.08±3.5	51.78±4.2	52.16±4.7
E2E L5	–	59.78±3.8	62.78±3.6	52.26±4.8	59.52±4.8	52.12±1.7	49.58±3.3
E2E L6	76.82±1.5	56.32±5.1	59.34±3.2	54.22±4.6	53.94±8.4	51.84±4.2	50.62±3.5

5.4.2.2 Multiple Binary Classification Tasks

In order to validate the difficulty of transfer tasks, we initially consider two different cases of the task to explore: (a) training on Open Door 1 *vs.* Open Door 2 as a source problem and testing on Close Door 1 *vs.* Close Door 2 as target; and (b) training Open *vs.* Close on Door 1 and testing the classifier on Open *vs.* Close on Door 2. Under these circumstances, the test accuracy (with balance between two classes around 48% *vs.* 52%) of case (a) is 89 ± 0.25%, which suggest that recognising the doors is an easy problem. On the contrary, the micro F_1 score performance of case (b) is 48.25 ± 3.2%, which is a more suitable problem requiring TL. Further problems we selected were based on similar preliminary experiments to this.

Considering TL from the source domain (being opening and closing of door 1) to target domains (being opening and closing of the other six objects), Table 5.5 shows a result comparison of the binary classification tasks described above with training the source domain model using CL and E2E learning respectively. For simplification, D1, D2, DW, Dr1, Dr2, Dr3 are the abbreviations of targets: Door 1, Door 2, Dish Washer, Drawer 1, Drawer 2 and Drawer 3 respectively, which is followed in this work unless otherwise specified. The transferring features are taken from different layers of the source domain networks. From results (being evaluated by weighted F_1 score) shown in Table 5.5, we note the transferability of features extracted by cascade networks shows a consistent monotonic decline with network depth where features are taken from. However, for E2E trained networks, this monotonic decline is partly true. In this setting of TL, features from the first layer of cascade networks show optimal transferability from which our motivating idea (a progressive specialisation included by layer-wise training) is confirmed. Our intuition on this observation is that coarse features are learnt by early layers while

finer features specific to the source task are picked up by later layers. We further notice the features from CL show competitive transferability with features from E2E learning. Although the deeper E2E model shows better performance (around 3%) than all cascade models on the source domain, the performance on the target domain is not significantly affected by the learning ability of model on the source domain. The uncertainty of knowledge transfer is high on these binary classification tasks, but the difference of performance between the first layer and the final layer is higher. Overall, the knowledge transfer to similar objects (from door 1 to door 2 and fridge) shows much higher transferability than to dissimilar objects (dish washer and drawers). Besides, the performance on dissimilar objectives (e.g., dishwasher) has high variance (around 8%), which may affect the observations. For example, knowledge transfer from the first layer of cascade models shows a statistically significant merit than from E2E models, with p-value 0.001 using T-test, when the object is dish washers. We also provide a baseline that training on the source problem and testing on the target problem without any further training, where test performance is only around 48% for all of objectives considered. This further justifies training in target domains as undertaken in this study.



(a) First layer (layer 1)

(b) Last layer (layer 6)

FIGURE 5.4: Confusion matrices of TL in the `Opportunity` dataset, learning from a 14−class problem down to a four-class problem with (a) transfer from layer zero and (b) transfer from layer five.

5.4.2.3 Transfer Learning from 14 Classes to 4 Classes

We show the results of TL from CL on this multi-class task in Figure 5.4 being confusion matrices from the first and the last layer. It is worth noting that we have introduced a hierarchy into the classifier outputs in this task which start from a group of coarse tasks (`Type One` *vs.* `Type Two`) and then spilt into finer tasks within the groups (Open *vs.* Close). Therefore, the later layer shows better performance compared to the early

layer, being different pattern from observation in Table 5.5. A fine-grained classification problem causes this result (that later layers shows better performance on the finer task) which confirms the coarse-to-fine nature of features learned by CL in a layer-wise fashion. Figure C.1 in Appendix C further supports this nature by transferring knowledge to a coarser task.

5.4.3 Task 3: Transfer Learning across Datasets

Figure 5.5 shows the experimental results for the setting of transferring learned knowledge from a task in the `Skoda` data to a task in the `Opportunity` where the computationally simpler cascade trained model shows competitive performance compared to the E2E networks. Deeper layers shows worse transferability (see Figure 5.6). Here again we observe the same patterns of performance noted with the results of Table 5.5.

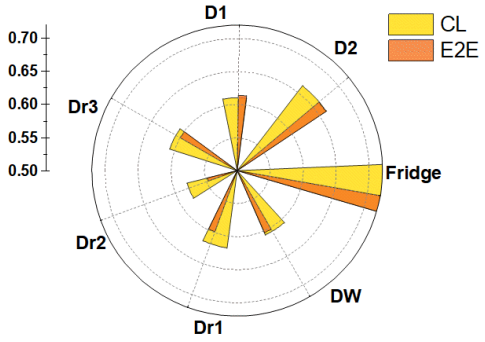
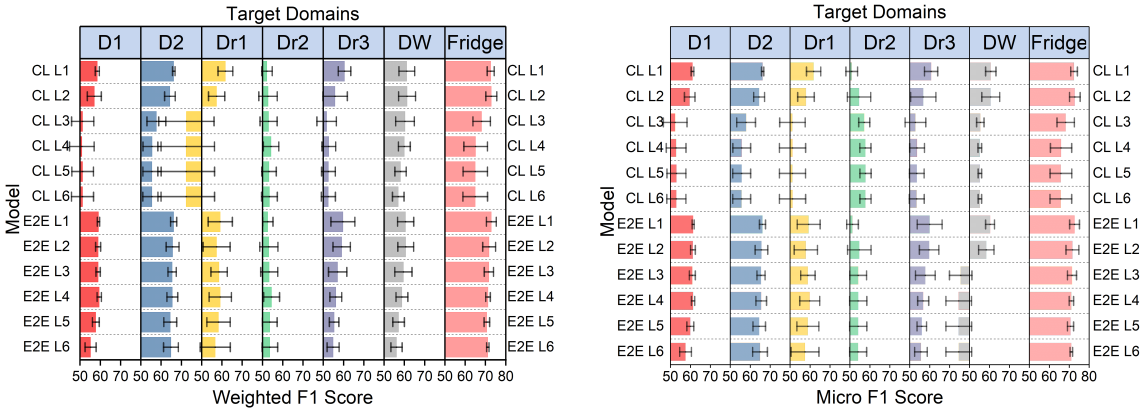


FIGURE 5.5: TL performance from `Skoda` to `Opportunity` based on CL and E2E learning (the optimal performance from layers). Corresponding to task 3, the TL is evaluated by micro F_1 score.

5.5 Summary

In this chapter, we explored TL from cascade learned networks on HAR, the second of the applied problems we study in this dissertation. Unlike the single cell data we dealt with in the previous chapter, multiple hidden layers help and we can observe different behaviours of cascade and E2E trained models. In addition to being computationally cheaper, CL offers a specific advantages for TL arising from the way information is packed in the layers, as distinct from E2E trained models. Such differences were noted in the study of information plane trajectories in Chapter 3 as well. Our intuition that CL provides coarse features in early layers and finer features in later layers is validated by transferring knowledge to target domains layer by layer. In the next chapter, we will explore issues considerably larger than HAR taken from computer vision, medical imaging in particular.



(a) Weighted F_1 score

(b) Micro F_1 score

FIGURE 5.6: TL performance from `Skoda` to `Opportunity` based on CL and E2E learning. Corresponding to task 3, the TL is evaluated by (a) weighted F_1 score and (b) micro F_1 score.

As an approach to learning in layered networks, layer-wise training from CL restricts how information relating to the target is extracted, in contract to the inherent flexibility between layers enjoyed by E2E training under the same parameter configurations. Despite this difference, our results show that CL achieves competitive performance on two HAR classification problems compared to E2E training, and with significantly fewer parameters and training time than work based on DNNs reported by previous authors. We also find a hierarchical manner of relevant feature extraction across layers in CL, which is demonstrated by the result that features taken from different layers show monotonically decreasing transferability from the first to the final layer across tasks. Coarse features transferred from the first hidden layer of cascade networks trained on source domains provide optimal performance. Most importantly, these performance are satisfactory, and most of time (5/7) better than the performance of transferring knowledge from any layer of E2E trained networks. When transferring finer features, the features with better transferability can be obtained from the final layer, which further reinforces this point.

Chapter 6

Knowledge Transfer from Natural to Medical Images based on Cascade Networks

Building on the success of Transfer Learning from Cascade Learning (TCL) seen in the previous chapter, we report on it applications to natural and medical image processing problems. The medical domain, in particular, is of interest because many problems in this area are posed in the low data (Lundervold and Lundervold, 2019), low computational resources (Willemink et al., 2020) regimes. The work reported in this chapter consist of the following:

(a) We empirically demonstrate a case for Transfer Learning (TL) from cascade trained networks using several natural image and two medical image classification problems.

(b) To explore alternate ways of setting up Cascade Learning (CL) models, we introduce Semantic Cascade Learning (SCL), where we set up hierarchical learning problems, exploiting semantic information in class labels, extracted through *Word2Vec* model (Mikolov et al., 2013).

(c) Working towards interpretability, we use subspaces analysis (Singular Vector Canonical Correlation Analysis (SVCCA) (Raghu et al., 2017)) and an image morphology-based measure to characterise distributions in saliency maps (Moreno et al., 2019) of gradient distributions.

6.1 Motivations

With the advances seen in Deep Learning (DL), End-to-End (E2E) learning is the most widely used approach in the DL literature undergoing a rapid development into diversified aspects. However, the training of E2E networks usually has two essential requirements, large amount of annotated data and significant computational resources (Erhan et al., 2009), of which both are difficult to obtain in the medical domain (Castro et al., 2020; Claudio Quiros et al., 2021).

The scarcity of annotated medical data is partly caused due to the diseased population data being small, or annotations having expensive costs. The reverse also tends to be true as there is no reason to let healthy people subject themselves to an intrusive experiment such as taking a biopsy from an organ. Key ethical and legal questions further aggravate the lack of data (Esteva et al., 2021) so that both industry and academia are relying on small public data. Taking the recent example of diagnosis from chest X-rays of Covid-19 patients, Ai et al. (2020) and Fang et al. (2020) have attempted to diagnose Covid-19 from only 51 patients' chest computerised tomography (CT) scans. A striking claim in (Fang et al., 2020) is that CT scans may be more sensitive than real-time polymerise chain reaction (RT-PCR) results. While the results reported are impressive (sensitivity 98%, in comparison to PCR 71% ($p < 0.001$)), the data is not acquired via a controlled experiment and suffers strong bias with respect to the PCR testing (Warnecke et al., 1997). Furthermore, Soares et al. have recently launched a Kaggle competition[1] based on the dataset used in (Soares et al., 2020). While this dataset includes 2482 images in total (1252 positive cases and 1230 negative cases), we note that the number of unique patients is a mere 120. The Covid-CT dataset in (He et al., 2020; Zhao et al., 2020) consists only of 349 CT scans containing clinical findings from 216 patients. All the above reinforce the need to work with small amounts of data in a target medical domain.

To date, the practical setting and the shifts towards personalised medicine using ubiquitous devices (e.g., medical Artificial Intelligence (AI) devices) for monitoring (Park et al., 2020), are built on resource-limited platforms (e.g., portable and battery-operated devices) requiring small models. Hence, a challenging task is bridging the gap between training Deep Neural Networks (DNNs) (for different users) and resource-limited platforms with respect to high demands on processing ability, memory capacity and energy efficiency. The above reasons necessitate studies that are not particularly focused on achieving high performance alone as their only goal, but seek to explore regimes of low computational cost and data scarcity.

While most real-world applications are built on E2E neural networks start with an arbitrarily sized architecture or search for an optimal architecture with repeated attempts, works related to adapting network architectures to the complexity of a problem have

[1] https://www.kaggle.com/plameneduardo/sarscov2-ctscan-dataset

been explored by researchers. Adapting training shows superiority of training DNNs with limited computational resources since learned features can be cached at any point in time such as with Deep CL proposed by Marquez et al. (2018) (see Chapter 2 for details). A layer-wise approach has the additional advantage of allowing for the optimisation of more interpretable objectives (Belilovsky et al., 2019) which is also important in medical applications. As mentioned before the domain of medical inference requires not only a modestly complicated model but also mitigating the issue of data scarcity of which TL has been shown as a promising solution. That is, a TL model trained in a statistically similar problem domain with large amounts of data can provide feature representations that transfer very well to a target domain with data scarcity (Pan and Yang, 2009; Pratt, 1992).

Several researchers have applied TL to medical imaging in recent years (Ravishankar et al., 2016; Shin et al., 2016; Talo et al., 2019; Xie et al., 2019), mostly reporting positive results transferring based on E2E training. However, the existing popular massive E2E networks are usually data hungry, and are likely to overfit when trained on small datasets. The CL we pursue in this work, applied to TL (termed as TCL) for inference from medical images is motivated by the above considerations and has not previously been investigated. The nature of the way that CL works further motivate our investigation. For example, in CL there is a progression in complexity as more and more layers are added resulting in coarse-to-fine feature extraction as previously shown in Chapter 5. Thus, one could expect that features extracted from earlier layers can be effective when transferred to a related, yet different, target problem.

As an aside, progressively complex hierarchical processing is also of interest in sensory processing in biology. Serre (2014), for example reports hierarchical visual sensory processing in the neocortex of the mammalian brain. Moreover, Lee et al. (1998); Serre et al. (2005) develop algorithms for hierarchically extracting information from data in the field of AI. A new paradigm emerges in AI paying attention on finding information representations that exhibit characteristics similar to those of the neocortex. They attempt to imitate a primate visual system in DL with a sequence of processing stages: detection of edges, primitive shapes, and moving up gradually to more complex visual shapes (Bengio, 2009; Lee and Mumford, 2003). Additionally, Yosinski et al. (2014) propose that the recognisable low level image features are extracted by the early layers and more abstract representations are extracted by later layers. Similar to a multi-class classification setting, different pairs of classes also vary in the difficulty with which they can be classified. For example, it is harder to let a classifier distinguish different types of dogs or different types of cars than it is to tell cars from dogs. Information pertaining to this is likely to be found in the class labels. We seek to exploit this observation in a framework we refer to as SCL and its influences in the setting of TL.

6.2 Related Work

There is a recent surge of research in applying TL to medical imaging, including works from Ravishankar et al. (2016); Shin et al. (2016); Talo et al. (2019) and Xie et al. (2019), where popular directions include using a pre-trained model as a feature extractor and fine tuning. The first set of works (Arevalo et al., 2015; Bar et al., 2015; Van Ginneken et al., 2015) transfers knowledge using pre-trained networks as a feature extractor. More specifically, medical images are fed into the trained model, and then the output (features) of a certain layer from this trained model is used to train a new pattern classifier on the target tasks. Recent evidence suggests that features extracted from raw data directly applied to neural networks, as opposed to features that use some prior knowledge of images (e.g., spectra) show superior performance. The second set of works (Carneiro et al., 2015; Chen et al., 2015; Margeta et al., 2017; Schlegl et al., 2014; Shin et al., 2015) replaces the prediction layer of a pre-trained network with a new logistic layer to fit target domain tasks. The target domain data is further used to train the new final layer and keep all other layers the same. This latter approach consistently shows promising results compared to training a network from scratch.

While much of the literature applying TL reports positive results in terms of improved performance, Raghu et al. (2019), making a critical appraisal, suggest otherwise. The authors show that transferability might be interpreted in terms of changing high-level features from later layers for fitting target domain tasks, via observing the SVCCA similarity of a pre-trained model's outputs before and after fine tuning. However, Ke et al. (2021) show an opposite conclusion that the family of architectures instead of the model size is responsible for determining the performance of TL with performance improvements based on models pre-trained on ImageNet. They further believe that pre-training with smaller networks can give better supports of TL than bigger networks in the same family of architectures. Generally, the success of knowledge transfer depends on the transferability of low level image features between the source and target domains, as the higher level features from later layers are potentially too specific resulting in less transferability. Therefore, we suggest a smaller network has a limited capacity of learning, being similar to early layers of cascade networks, extracting overly specific features which leads to better transferability.

Considering TL in a layer-wise manner on medical images, Tajbakhsh et al. (2016) apply fine tuning layer by layer on the target domain, which is different from our work. They keep the entire pre-trained E2E model frozen, and replace the old fully connected layers at the end of models with new fully connected layers to fit the target domain tasks. They make the new added layer and the last convolutional layer to be trainable on target domain data and progressively make more convolutional layers, from bottom to top, to be trainable in search for the optimal performance. The authors define "shallow tuning" as tuning the last few convolutional layers and "deep tuning" as tuning all the convolutional

layers. They suggest this is a more efficient way of TL compared to other literature and suggest this layer-wise manner leads to incremental performance improvements.

In comparison to their layer-wise TL, our TCL further propels the gain of efficiency in the setting of TL. Firstly, unlike with E2E trained models, TCL does not need the entire model to be trained on the source domain. We only need to train the model up to the layer from which we choose to extract features for knowledge transfer. While this is not an issue in applications such as computer vision and natural language tasks where a wide range of pre-trained models are available for download and use, new applications (e.g., biology) will require source models to be trained, which means that TCL can save vast amounts of computational resources for the entire process. Secondly, the bottom-to-up (last to first layer) selection of trainable hidden layers shows potential wasting of computational resources in both memory and training time. High level features from the later layers of a pre-trained model may not contribute to target domain tasks or can even impede finding a solution especially when the dissimilarity between the source domain (natural images) and the target domain (medical images) is high (Azizpour et al., 2015). Contrary to their order of selection, our TCL is an efficient way of transferring knowledge from early layers including low level feature representations and then incrementally include later layers which helps avoid confusion given by later layers. Potentially, it may be able to fit different requirements of feature transferability depending on the progressive increasing similarity of the source and the target domains. In the following section, we show more details of how to implement TCL in this application.

6.3 Datasets and Methodologies

In our experiments, we utilise four widely used datasets (CIFAR 10 (Krizhevsky et al., 2010), CIFAR 100 (Krizhevsky, 2009) ImageNet (Deng et al., 2009)) and DTD (Cimpoi et al., 2014) in the area of computer vision, a chest X-Ray based medical diagnostics dataset (CheXpert (Irvin et al., 2019)) and also a Covid-19 dataset (de la Iglesia Vayá et al., 2021) of chest X-Rays and CT from the Valencian Region Medical Image Bank (BIMCV). There are three parts of experiments based on these datasets: (a) transferring knowledge from models trained in the setting of CL; (b) SCL (a variant of CL) and Transfer Learning from Semantic Cascade Learning (TSCL) (corresponding to TL); and (c) saliency map based visualisation of representations of hidden layers, and a granulometry score used to measure the concentration ratio of activated areas. More details are shown in the following subsections.

6.3.1 Datasets

CIFAR 10, CIFAR 100, ImageNet and DTD are benchmark datasets with equal distribution of classes. Since our goal is comparing the transferability of features from two

different learning mechanisms instead of the pursuit of high performance on benchmark datasets, in part of the experiments we use the reduced version of the datasets for experiments. In particular, in the setting of transferring knowledge from ImageNet to CIFAR 100 and medical datasets, we use a 23-class problem, where the classes are overlapped between ImageNet and CIFAR 100.

The CheXpert dataset is an open source medical image classification problem[2] with $223,000$ chest X-Rays from over $65,000$ patients. In our experiments, we take the problem of five classes of pathologies following the setting given by Raghu et al. (2019): Cardiomegaly, Edema, Pleural Effusion, Atelectasis and Consolidation. The labels of classes are automatically obtained by analysing radiography reports associated with the images. In total, a multi-task binary classification problem is applied to this dataset, where the two classes are disease and non-disease.

For BIMCV, there is a binary task of recognising Covid-19 cases and disease-free cases. We manually scan through them and filter out images with multiple labels and retain 4940 images as a dataset to be used. There are equal number of images for both disease-free (labelled as 'normal') and Covid-19 scans. We randomly take 75% of data as the training samples.

6.3.2 Cascade Learning and Transfer Learning from Cascade Learning

As introduced in Chapter 2, TCL is defined as transferring knowledge learnt by cascade networks on the source tasks to the target tasks in a layer-wise fashion. A schematic of both the CL and TCL approach is shown in Figure 6.1. On the source domain, a $N-$layer cascade architecture is trained in a one-by-one layer-wise fashion keeping previously trained layers being frozen. In contrast, E2E learning trains all layers of a model simultaneously with the same architecture as shown in the lower left corner of Figure 6.1.

For each layer of a cascade network, we add an Auxiliary Classifier (AC) (inspired by (Belilovsky et al., 2019)) consisting of k convolutional layers and f fully connected layers to fit labels and then improve performance in training the n_{th} layer. In all of experiments in this chapter, we set $k = f = 2$ (for keeping the auxiliary networks shallow with respect to the network depth as suggested by (Belilovsky et al., 2019)) and keep the preceding $n-1$ layers frozen, giving a fixed transformation which can be seen as a feature extractor acting on the input data. In a similar manner, we freeze the first n layers and add a randomly initialised AC in training the $(n+1)_{\text{th}}$ layer where the objective is to minimise the cross-entropy objective by Stochastic Gradient Decent (SGD) or Adam (for the DTD dataset).

[2]The data can be obtainable at https://stanfordmlgroup.github.io/competitions/chexpert/.

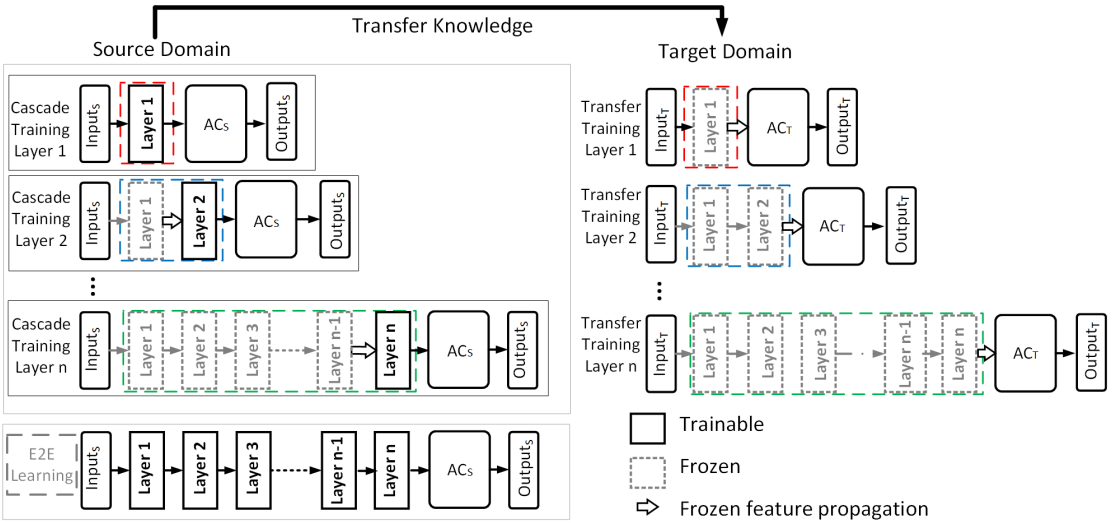


FIGURE 6.1: The schematic diagram of CL, E2E and TCL. From left to right, TCL is implemented by progressively freezing source domain trained layers and re-training randomly initialised AC on the target domain. The subscripts $_S$ and $_T$ stand for source and target domains, respectively. The dashed boxes with the same colour illustrate that the features from these layers are shared across source and target domains in transferring knowledge. TE2E is implemented by the same layer-wise manner. Finally, on the target domain TCL and TE2E have the same size architectures for comparison.

For CIFAR 10 and CIFAR 100 datasets, in the first four layers of a cascade network, we use convolutional layers consisting of 256 filters with kernel size 3×3. For layers beyond the fourth layer, we reduce the number of filters to 128. Subsequent fully connected layers consist of 64 and 32 units leading to the output classification units are attached. Batch normalisation, max pooling and Rectified Linear Unit (ReLU) activation functions are used throughout the experimental work with standard data augmentation (e.g., horizontal flipping, normalisation, and central cropping). For Describable Textures Dataset (DTD) dataset, all six convolutional layers include 256 filters and are connected to two fully connected layers consisting of 128 units. For ImageNet datasets, the first eight layers have 256 filters and the last three layers have 128 filters when subsequent two fully connected layers have 128 units and other settings are consistent. The same architecture are also applied to CheXpert and BIMCV. In comparison to cascade networks, the E2E networks have the same setting and structure on each dataset. To set network hyper-parameters (e.g., number of filters), we explored several architectures heuristically and evaluated their performances on a validation set and chose the best.

We tune the learning rates to have an initial value of 0.1 on CIFAR 100 (0.01 on CIFAR 10, 0.01 on ImageNet, 0.001 on DTD), reducing by a factor 0.8 every 60 epochs (15 epochs on CIFAR 10, 15 epochs on ImageNet). Training took a total of 200 epochs on CIFAR 100 and DTD (60 epochs on CIFAR 10 and 45 epochs on ImageNet, generally following the setup in (Belilovsky et al., 2019)).

For CheXpert, the initial learning rate is 0.01 with 20% of reduction every 5 epochs and in total we use 20 epochs to train the model. Slightly different from other normal classification tasks, the objective function of this multi-task binary classification task is binary cross entropy loss using sigmoid as the activation function of output layers, following the setup in (Huang et al., 2013; Raghu et al., 2019). The evaluation of this dataset is area under receiver operating characteristic curves (AUC) score following the setting in (Raghu et al., 2019). For BIMCV, we use 45 training epochs and initial learning rate 0.01 with 90% of reduction every 20 epochs. All the networks are trained based on standard data augmentation (including horizontal flipping, normalisation and central cropping) with at least 3 runs for validating uncertainty.

In the setting of TCL from the n_{th} layer, a replica of the network trained on the source domain, up to layer n, is set up with adding a randomly initialised classifier, AC_T, as shown in bounding boxes in Figure 6.1 in the target domain. There are two cases of knowledge transferring: with fine tuning and without fine tuning. We term the transferring process as TCL when only the AC_T layers are trainable on the target domain data. Making parameters of all transferred layers to be trainable is additionally termed as the transfer cascade learning with fine tuning approach (TCLFT). We explore the transferability of features from CL layer by layer, and compare them to features from same locations of E2E networks. TL based on E2E networks is referred to as Transfer Learning from End-to-End (TE2E). Considering the coarse-to-fine nature of CL (namely, the early layers make a "coarse" separation on the dividing plane between classes and the later layers refine the separation), we also develop SCL, an extension to the basic CL approach, described below.

6.3.3 Semantic Cascade Learning

We consider an alternate way of setting up a CL problem that takes advantage of hierarchical semantic relationships between classes contained in class labels. This is motivated by the fact that early layers in CL are small networks not capable of forming complex class boundaries. Within a multi-class classification problem, we would usually expect a hierarchical relationship in which higher-level classes are easily separable from lower-level ones. For example, it is easy to classify several types of dogs taken as a group from different types of vehicles taken as a group than it is to classify individual dog species or vehicle types. This points to a formulation in which early layers can be set to classify all dogs from all vehicles as a two-class problem and later layers to be set up to classify the different dog types and vehicle types. To acquire the necessary information to construct this semantic hierarchy, we take the learned representation of class labels from `Word2Vec` embedding[3] and apply hierarchical clustering[4]. An example of this process is illustrated

[3] Can be obtained from website (`https://github.com/mmihaltz/word2vec-GoogleNews-vectors`).

[4] Using the nearest-neighbour chain algorithm (Lu and Tan, 2003) with the Ward's method (Ward Jr, 1963).

in Figure 6.2. By slicing the dendrogram at various levels, we can construct a hierarchy of progressively harder problems to train each successive layer of a CL architecture. In Figure 6.2, the first layer is trained on a two-class problem {Car,Truck, Plane, Ship} *vs*. {Horse, Cat, Dog, Deer, Bird, Frog}. When the second layer is trained, we solve a three-class problem {Car, Truck, Plane and Ship}, {Horse, Cat, Dog}, {Deer, Bird, Frog}. Our expectation is that such a hierarchical formulation is likely to induce the learning of coarse features in early layers and more fine features deeper into the network. For reference, we also experimented with grouping these classes at random (which we refer to as Random CL).

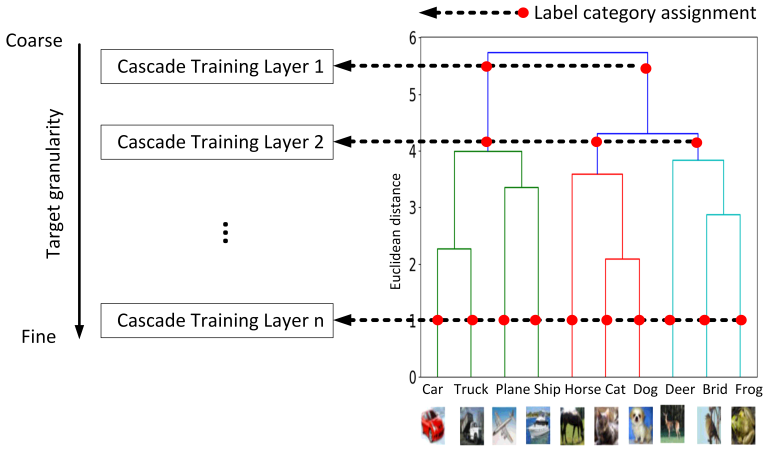


FIGURE 6.2: Overview of SCL, an extension of CL. SCL is based on a hierarchy of progressively harder problems which is introduced by semantically clustering the class labels based on their `Word2Vec` distributed embedding. The number of classes in each layer is same as the number of red dots on the dendrogram. The blocks containing "Cascade Training Layer n" are the CL layers shown in Figure 6.1. This approach is used to encourage the CL architecture to extract coarse features at the early layers and finer ones in the later layers.

6.3.4 Visualisations

In addition to performance comparison, we also provide three different visualisation methods: (a) saliency maps; (b) granulometry scores; and (c) SVCCA similarities, to validate the coarse-to-fine extraction process over layers of CL. Saliency maps highlight regions that have big impact on recognising the class of the image. The granulometry score is a morphological measure of granularity in the image based on corresponding saliency maps to quantify how active contiguous regions are distributed. SVCCA reflects the similarity between representations.

6.3.4.1 Saliency Maps and Granulometry Scores

In our experiments, saliency maps are obtained according to the work of Simonyan et al. (2014) who find activated pixels of images based the gradient of images. Namely, the

pixels with a large gradient (either positive or negative) only need small modification to affect the classification results, hence such pixels correspond to the location of interested regions. By taking the gradient of the class score with respect to the input image, saliency map can be obtained. As colour images usually have three channels (R,G,B) and we take the maximum magnitude across all channels.

From the viewpoint of mathematical morphology, granulometry is an approach to calculate a distribution of grain size in binary images based on a series of morphological opening operations (Dougherty et al., 1989). On the top of that, we propose granulometry score based on the saliency maps of input images to measure the distribution of activated contiguous regions as shown in Algorithm 2. We use binarised saliency maps as the input of granulometry to obtain the distribution of different sized highlighted connected regions. In this process, we take activated regions with the top 25% brightness to do binarisation for filtering out regions with negligible brightness. From the distribution of different sized regions, we take the number of median regions as the output of granulometry algorithm for avoiding the case in which all active regions has low brightness. From each layer we get a granulometry score and then normalised by the maximum.

Algorithm 2 Pseudocode of the Granulometry

Arguments:
    A: A saliency map of an image with size $m \times n$.
Outputs:
    GS: The granulometry score.
Initialisation:
    $B(u)$: A structuring element with size u, e.g., a square or an ellipse.
    AS: The list of grain (connected areas) size.
    M: Binary mask of A. The element of mask is "True" or "False".
    C: A list of grain counts.
    $M \leftarrow$ A$<$ 75% quantile of A. {This step filters out regions having low brightness in the saliency map.}
    AS=[1, int$(0.25 \times max(m,n)/20)$, int$(0.75 \times max(m,n)/20)$, int$(max(m,n)/20)$]
for s in AS **do**
    $newA \leftarrow$ opening$(M, B(s))$.[5]
    $C \leftarrow$ append the counts of connected areas[6] in $newA$.
end for
GS $\leftarrow$ the median of C.

6.3.4.2 Singular Vector Canonical Correlation Analysis

For comparing the representation from E2E learning and CL at the same layer of the same sized networks, we utilise the SVCCA similarity as a measurement of representation differences. We take representations from networks trained by both learning mechanisms

[5]The opening is a mathematical morphology operation including the succession of an erosion and a dilation of an image A with the same structuring element, implemented using the library *scipy.ndimage.binary_opening*. Using a structuring element B (e.g., a $k \times k$ square) to move on the image A pixel-by-pixel, the erosion $A \ominus B$ operation let all pixels in the superimposed area to be zero if one of values of A is zero in this overlapped region, otherwise, they are set to 1. The dilation operation $A \oplus B$ does the opposite way.

[6]During detecting the connected area for each new pixel being 1, if no pixel on its left or right has value 1, we assign a new tag to it. If there are pixels on its left or up are 1, it uses the assigned tag. If pixels on its both left and up are 1, we let them have the same tag as they are connected.

in a layer-wise fashion, and compare their similarity in each aligned direction and average values over all directions. The details of the SVCCA algorithm are shown in Appendix A.7.3.

6.4 Results

We illustrate our results as three parts: (a) performance of CL on the source domain; (b) comparison of feature transferability of E2E and CL between different natural images; and (c) comparison of feature transferability of E2E and CL from natural images to medical images. In the last two parts, we specifically investigate the situation that the target domain has limited data, which we control from less to more, in order to show the advantages of TL for data scarcity. Additionally, by using SCL and TSCL as variants of CL we further validate that features from cascaded networks prominently have better or comparable transferability than from E2E networks in our scenarios using images.

6.4.1 Performance Comparison on Source Domains

Table 6.1 compares performance of various classifiers (including CL and E2E) using the CIFAR 10 and CIFAR 100 problems to set up a baseline. We note that our heuristic implementation of CL produces comparable performance against all models except the heavily optimised VGG architecture on the CIFAR 10 problem. In particular, our results are also comparable to other two layer-wise learning on the same data (i.e., work in (Belilovsky et al., 2019) and (Marquez et al., 2018)). Although, it is possible to achieve better performance on these datasets by extensive tuning, the results presented satisfy our purpose of training a decent enough model in the source domain with the objective of transferring learned features to a target domain. Indeed, both work by Ke et al. (2021) and our later results confirm that over-tuning in the pursuit of high performance in the source domain problem will not contribute to or diminish performance in the target domain.

For SCL and Random CL, Figure 6.3 shows corresponding accuracy comparison between them. From left to right, the number of classes in the tasks increases in sequence: $2, 4, 8, 30, 54, 82, 97,$ and 100. In this setting, we solve progressively harder classification problems layer by layer, hence the performance monotonically decreases along layers as expected for both SCL and Random CL. We also note that problems, defined by semantic clustering of class labels, show systematically higher performance than the cases in which the labels are randomly grouped. After the seventh layer (which includes the 7_{th} layer), both SCL and random CL are solving 100-class problems, hence there is no difference between them, which is rather disappointingly the SCL does not lead to significantly better performance than the random CL on the harder tasks.

TABLE 6.1: Baseline performance comparison: Our implementation of CL on CIFAR 10 and CIFAR 100 tasks compared with accuracies reported in the literature.

Models	CIFAR 100 (%)	Model size	CIFAR 10 (%)	Model size
CL (ours)	65.70 ± 0.04	2 layers[1]	86.70 ± 0.10	6 layers
CL[2]	59	6 layers	84	6 layers
Greedy Layer-Wise[3]	−−	−−	88.3	5 layers
E2E	65.07 ± 0.86	2 layers	87[2](88.4[3])	6(5)layers
AlexNet (E2E)	54.04[4]	3 layers	89[5]	4 layers
VGG (E2E)	68.48[4]	5 layers	92.45[6]	13 layers

1: Best performance taken from the second layer; further layers results in over-training, degrading to $59.01 \pm 0.01\%$ at 12 layers.
2: Token from the work given by Marquez et al. (2018).
3: Token from the work given by Belilovsky et al. (2019).
4: Token from the work given by Ahmed et al. (2016).
5: Token from the work given by Krizhevsky et al. (2017).
6: From `http://torch.ch/blog/2015/07/30/cifar.html`

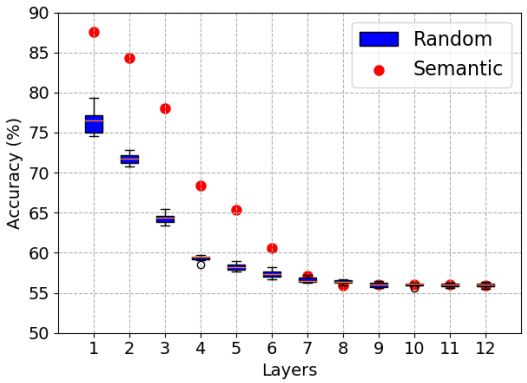


FIGURE 6.3: Solving progressively harder problems by semantic clustering in CL. On the CIFAR 100 problem, Word2Vec representations of class labels are hierarchically clustered and successive layers of CL are trained on finer and finer problems with number of classes increasing in the sequence: 2, 4, 8, 30, 54, 82, 97, and 100. This is benchmarked against random partitioning of the same number of classes in each layer (boxplots of performances from five runs), showing that semantically grouping classes forces the solving of easier problems in early layers. Beyond the seventh layer, both settings solve 100-class problems, hence show indistinguishable performances.

6.4.2 Transfer Learning on Natural Images

Using the baseline models as source models, we implement TL in which features learned from the source models are used to enhance learning in a related target domain. Firstly, we transfer knowledge between multi-class problems where both the source and target problems are natural image classification tasks. There are two cases for this part: from CIFAR 100 to CIFAR 10 and from ImageNet to CIFAR 100 in which a 23-class overlapped subset is chosen for classification.

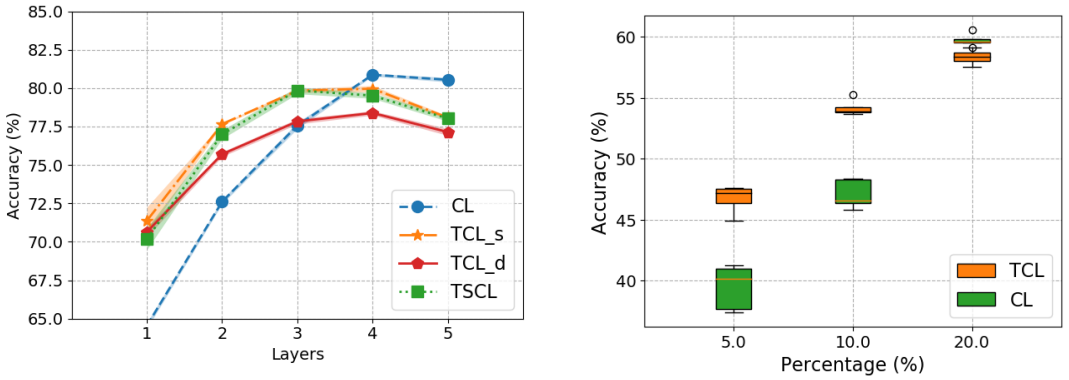
As shown in Table 6.2, we compare the cases in which the source model is trained by different learning mechanisms (i.e., CL, SCL, random CL and E2E) with CIFAR 10 as the target domain. All layers shown in the table are trained for a 10-class task on the

TABLE 6.2: Results of TL in a CL setting with CIFAR 100 as source learning problem and CIFAR 10 as target problem. Three different CL source models are considered (SCL: progressively harder multi-class problems at each layer using semantic clustering of class labels; Random CL: same numbers of classes at each layer as in SCL, but the classes grouped at random; and CL: Each layer solving a 100-class problem) and compared against E2E training of the source model. Taking features for transfer is better towards the early part of the network in all models. Early layers of cascade trained models give significantly better transferable features, though clustering the classes based on the semantics of their labels offers no detectable advantage.

Layers	*TSCL*	*TRandom CL*	*TCL*	*TE2E*
1	79.74±0.03	80.31±0.28	80.18±0.02	74.22±2.17
2	79.81±0.04	80.08±0.25	80.44±0.04	74.88±0.20
3	79.99±0.06	80.06±0.15	81.59±0.03	67.24±0.18
4	80.03±0.04	79.52±0.31	81.61±0.03	60.78±0.24
5	79.19±0.04	77.67±0.14	80.75±0.03	55.17±2.15
6	77.17±0.04	74.81±0.43	77.76±0.05	50.76±0.43
7	73.96±0.04	71.81±0.30	75.53±0.04	40.43±1.67
8	70.51±0.06	68.60±0.39	71.55±0.03	39.44±1.07
9	64.25±0.04	63.35±0.05	64.54±0.06	34.84±0.73
10	58.36±0.05	58.04±0.34	59.44±0.03	29.55±1.18
11	54.53±0.10	53.23±0.67	54.50±0.08	23.67±1.00

target domain. We also explore transferability of features extracted from various layers of a 11-layer neural network trained on CIFAR 100 in this table. We note from Table 6.2 that taking features from early layers of a network is better for knowledge transfer to a new domain than extracting features from later layers. The early layers outperform the later layers by around 20% in this setting (e.g., the difference between the first and the last layer). This is consistent for CL (including its variants), E2E learning, and works shown by Yosinski et al. (2014). It is also seen that all three variants of CL adopted are uniformly better at providing effective features for knowledge transferring irrespective of how they were trained (i.e., a plain CL with all the classes shown at every layer and progressively harder classification tasks chosen at random and by semantic clustering of the labels). Overall, we can say the features from the early layers are more suitable for knowledge transfer than the later layers.

As features from early layers show better transferability, the effect of TL to a target domain with knowledge from shallow layers is considered with results shown in Figure 6.4(a). Here, the source domain is the subset of ImageNet with 23 classes (we term it as ImageNet23) and the target domain is CIFAR 100. In the target domain, when the model is restricted in capacity from one to three layers, TL shows a distinct advantage in performance compared to the model only trained on the target domain. The differences of accuracies decrease with increased capacity allowed in the models. Additionally, even when the source and target domain have the different classes, TL still shows advantages than learning on the target domain without TL when the model has less than two layers. This agrees with work from Neyshabur et al. (2020) saying that lower layers are in charge of more general features.



(a) (b)

FIGURE 6.4: TL with limited capacity and data in the target domain. Both problems use ImageNet23 (the source domain) to train networks, and features are transferred to the CIFAR 100 task. In (a) when a cascade learner is limited in capacity to one to three layers, transfer of features is helpful. TCL_s means the source and target domain have the overlapped classes, and TCL_d means the source and target domain have different classes. In (b), when the available data in the target domain is very small (i.e. 5% and 10% of the dataset), TL offers a distinct advantage.

We also illustrate the effect of TL when the target domain has limited availability of data. As shown in Figure 6.4(b), the performance of both CL and TCL in the target domain increases with the growing amount of available data used to train the model. The improvements of TCL compared to CL also decrease with respect to the amount of available data (from 5% (575 images) to 20% images).

TABLE 6.3: Performance of TL from cascade and E2E trained source models to a medical image classification target (CheXpert data). The source models are trained on ImageNet23 with networks (TCL: Cascade trained network, TE2E: E2E trained network, TSCL: Cascade trained network with semantic clustered classes, TCLFT: Cascade trained source model with fine tuning, TResNet: Transfer from an E2E trained ResNet and ResNet: published results given by Raghu et al. (2019)).

Disease	*TE2E*	*TCL*	*TSCL*	*TCLFT*	*TRestNet*	*ResNet*
Atelectasis	77.57±1.12	79.80±0.93	79.73±0.09	**80.10±0.82**	79.76±0.47	79.52±0.31
Cardiomegaly	75.47±1.26	78.68±0.93	**79.04±0.12**	78.92±1.27	74.93±1.41	75.23±0.35
Consolidation	88.25±0.04	90.16±0.68	89.84±0.61	**90.28±1.01**	84.42±0.65	85.49±1.32
Edema	80.85±1.79	89.87±0.11	90.06±0.34	**90.64±0.21**	88.89±1.66	88.34±1.17
Pleural Effusion	83.61±0.47	91.33±0.13	90.96±0.22	**91.53±0.16**	88.07±1.23	88.70±0.13

6.4.3 Transfer Learning on Chest X-Ray Images

Based on the above basic establishments of TL in the setting of CL, we further explore the effects of TL applying to medical imaging study. Figure 6.5 and Table 6.3 show the corresponding results. From the table we note that each of the three variants of CL significantly outperforms state-of-the-art results published in (Raghu et al., 2019)

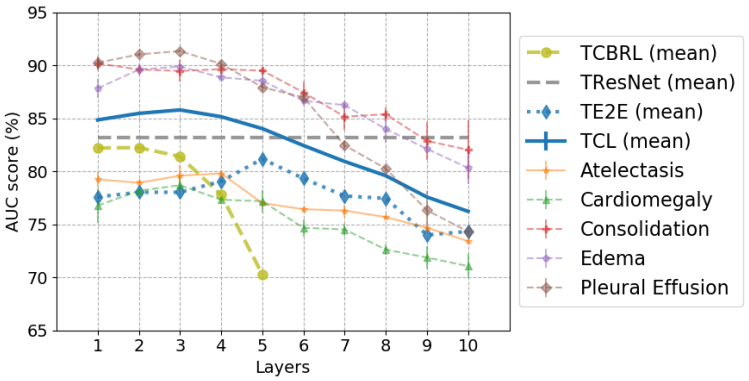


FIGURE 6.5: The performance comparison of TL from various architectures on CheXpert dataset. The continuous blue line is the average AUC over five tasks when transferred from a cascade trained source model. The dotted blue line denotes the average AUC performance of TL from an E2E trained source model. The dashed grey line represents averaged AUC of TResNet, and the lime dashed line gives results of knowledge transfer from an E2E model CBRL (both results taken from Raghu et al. (2019)). All other lines show detailed results of TCL for different diseases.

on this dataset where the model is a ResNet with vast amounts of parameters, and also outperforms the TE2E and the TL from ResNet architecture reported by them. For example, TCLFT shows significantly statistical advantages with p-value 0.0002 than TRestNet using T-test for the Atelectasis disease.

Figure 6.5 shows results of the same comparison, with more details about transferability of features from different layers including the work given by Raghu et al. (2019) with an E2E model. Consistent with previous results, features from later layers show reduction of transferability. Similar to results shown in Table 6.2, the performance decrease is dramatic for an E2E network and more gradual for CL. Overall, TCL significantly outperforms TE2E (from Raghu et al. (2019)). Figure 6.6 further shows the gain of TL on the limited CheXpert dataset, namely only small fractions of the X-Ray images are available for training in the target domain. Up to about 30% of the available training data is used, and the transfer of features from an ImageNet23 task helps to achieve a performance increase (AUC) of over 2.5%.

6.4.4 Transfer Learning on BIMCV

As a final illustration of TL on medical images, we addressed a problem of classifying chest X-Ray scan images from Covid-19 patients using a dataset described previously in Section 6.3. As shown in Figure 6.7, results of various models on the Covid-19 Chest X-Rays are compared. Taking the optimal performance of each model for the comparison, whereas a network (the first three columns of each sub-figure) is hard to learn this task directly without knowledge transfer especially on a small data regime, TCL is able to offer better performance and outperforms a large-scale ResNet network (see the third column).

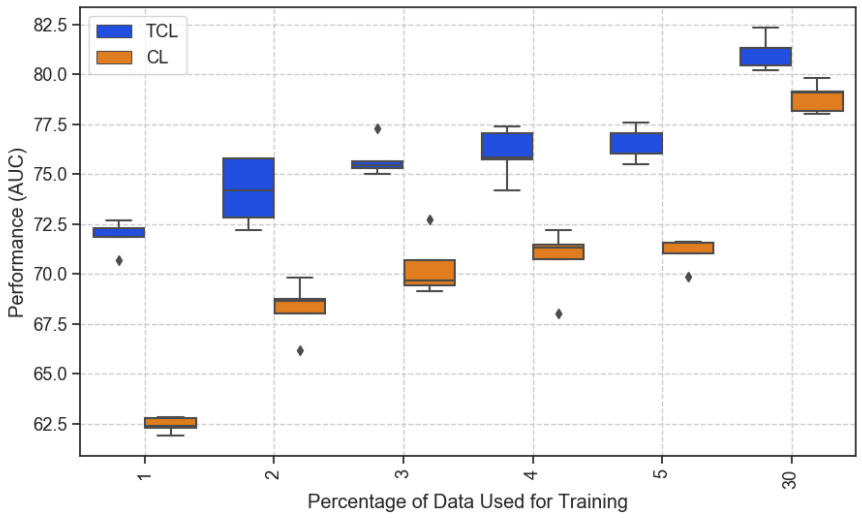


FIGURE 6.6: Comparing the role of TL with limited data in the target domain with natural images as a source domain (ImageNet23) to classify chest X-Ray images.

The improvement is around 5%. TResNet shows inferior performance in comparison to TCL, which is consistent with results on CheXpert (see Table 6.3). When only 5% data is available, TCL (from ImageNet23) shows the optimal performance in comparison to all of other models, while only 5 layers are needed. TL also outperforms direct learning on the small data regime. As a cooperation work with my colleague Junwen Wang, we also apply the similar TL processing to more medical classification tasks of which the results show the similar tendency.

TABLE 6.4: The comparison of necessary parameters from various source models. For CL, the number of parameters is summed up to the n_{th} layer from which the optimal TL performance is obtained on BIMCV data.

Learning mechanisms	On ImageNet	
	Total[1]	Trainable[2]
ResNet50[3]	around 23,521,000	around 23,521,000
CRBL[3]	around 7,840,333	around 7,840,333
CL	2,370,048	59,052
E2E	5,028,480	5,028,480

1 The total number of parameters of the model (excluding classifiers).
2 The number of trainable parameters of a model.
3 The model comes from the work given by Raghu et al. (2019).

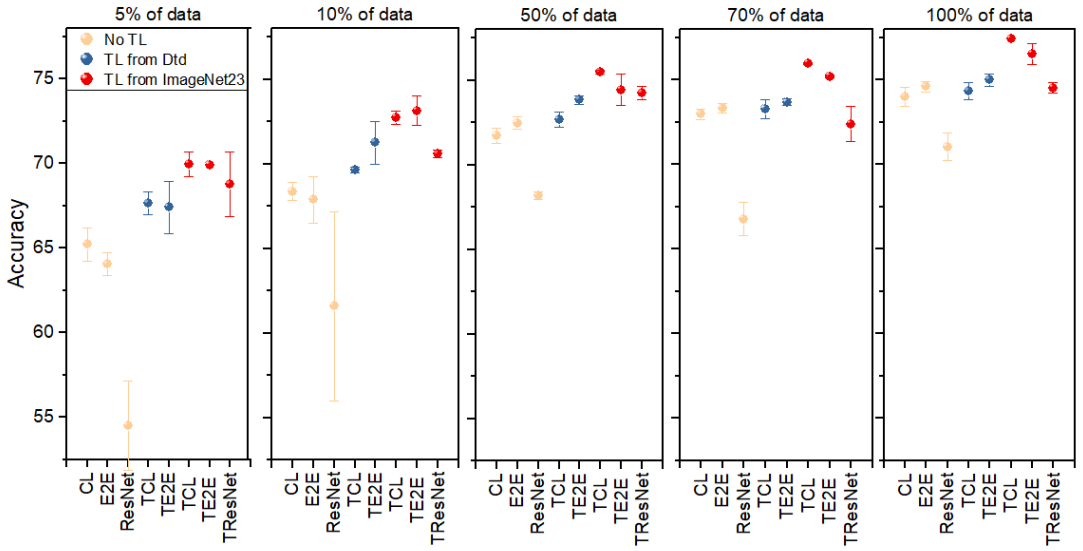


FIGURE 6.7: Comparison between TL and direct learning for the classification of a binary Covid-19 chest X-ray dataset (BIMCV). In comparison to TL, the first three columns in each sub-figure correspond to the performance of learning without knowledge transfer. The last five columns are for TL from two different source domains, DTD and ImageNet. Each sub-figure includes results of models trained on $X\%$ of data on the target domain.

6.5 Discussion

In the medical domain, the addressing of two issues are essential for Computer Vision problems, which are data scarcity and the need for simpler deployable models. TL to extract features from a related domain with similar statistical relationships is an attractive idea to address the issue of data scarcity. With the rapidly growing research on this topic, how effective TL is continues to remain an open question. The controversy between Raghu et al. (2019) and Ke et al. (2021) is a good exposition of this.

Instead of TL building on an E2E network as popular in literature, we propose TCL by transferring knowledge from multi-layer neural networks trained in a layer-wise fashion. We postulate that the progressive growth of network's capacity from layer-wise training encourages coarse features to be learned in early layers, and finer features (specific to targets) to be learned in deeper layers. Hence, more transferable features to a new yet statistically related problem, are enabled to be extracted from early layers of a network.

This could also happen in an E2E trained network as other authors, including Raghu et al. (2019) note, but cascade training helps to enforce this better and not leave it to chance as would happen in E2E training. Our empirical results on natural images and in TL applied to two medical domain problems confirm the better feature transferability from the early layers. Further support for such structured training is hinted at in (Raghu

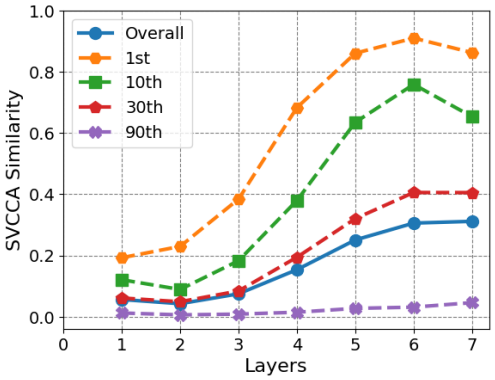
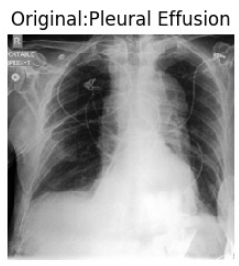


FIGURE 6.8: The SVCCA similarity between CL and E2E on the CIFAR 10 dataset. Dashed lines are the SVCCA similarity on the n_{th} aligned directions and the solid line is the averaged result on all aligned directions. There are small similarities on the early layer but large similarity has been found on later layers, indicating that the learning schemes between cascade and E2E learning have large differences in early layers. As the layer goes deeper there is relative increase SVCCA similarity since both have same learning targets.

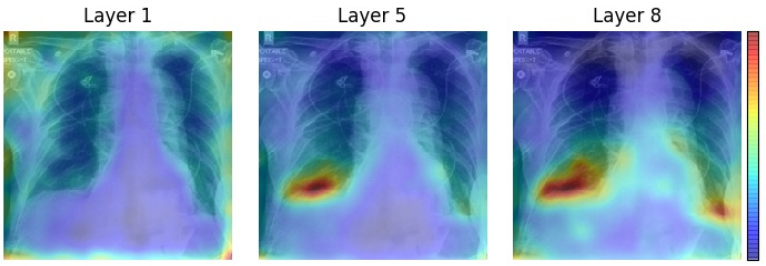
et al., 2019), when the authors suggest *freeze-training*, i.e., in training an E2E network where weights of early layers are stopped from being updated after a few epochs whereas later layers continue to learn. This again could be seen as a strategy similar in spirit to CL. Moreover, for the purpose of TL, TCL only need to train a few layers on the source domain instead of the entire network (as required by an E2E network), which further saves on computational resources. Table 6.4 shows the comparison of the amounts of parameters needed in the source domain across several learning mechanisms. When the optimal performance is obtained, CL needs the least parameters.

To compare two learning approaches, we compute the SVCCA similarity (Raghu et al., 2017) between models trained by both methods layer by layer on the CIFAR 10 dataset. As shown in Figure 6.8, we note that the early layers from the two learning mechanisms have very low similarity, as the cascade learner is training a model of low capacity whereas the E2E training can distribute information more freely across all the layers. When going deeper into the layers, both methods of training capture similar information along some directions as they are in pursuit of the same targets. In particular, the directions with high singular values (i.e., the first direction) show higher similarity than directions with smaller singular values. This may be caused by the learning of noise which can be an open question for future exploration.

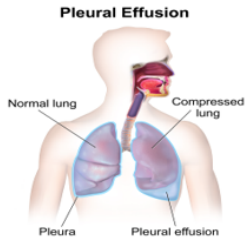
To illustrate the differences between CL and TCL, we show saliency maps (Simonyan et al., 2014) derived at different layers in Figure 6.9 on a CheXpert example of Pleural Effusion characterised by very localised fluid buildup at the lower part of the chest. From Figure 6.9(b), we note that the first layer of CL represents only coarse features of the image, and finer features of the illness show up in later layers. However, layer one of TCL is able to pick up the discriminant features even though it is limited in capacity, as shown in Figure 6.9(d). For TE2E, the distribution of activated areas is more scattered



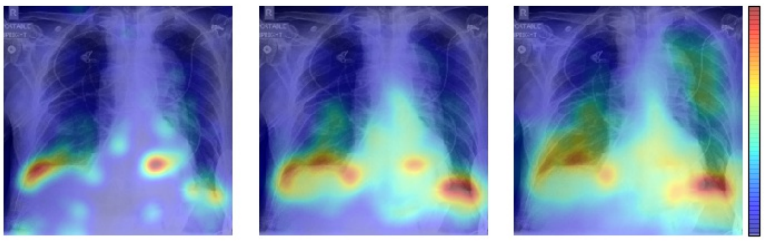
(a)



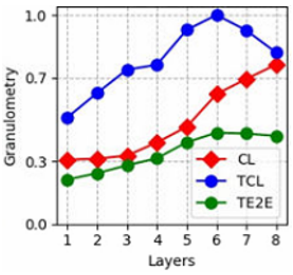
(b) CL



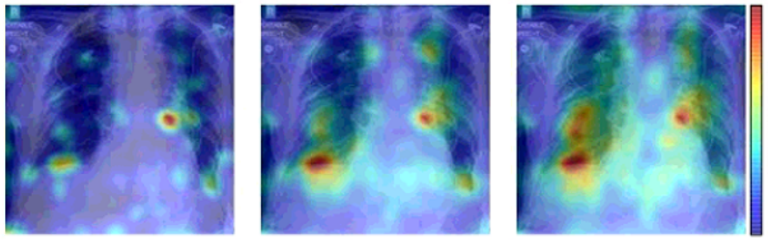
(c) From Commons (2021)



(d) TCL



(e)



(f) TE2E

FIGURE 6.9: Illustration of CL, TCL and TE2E using saliency maps and granulometry scores. (a) an example image from the CheXpert dataset; (b) saliency maps taken at three different layers in CL, showing a gradual increase in spatially specific features being extracted; (c) an illustration of the disease, pleural effusion, relating to (a); (d) saliency maps taken at different layers with TCL, showing that discriminant features are extracted at earlier layers; (e) granulometry scores, an averaged measure on 50 random test images, of spatial morphology of the saliency maps computed at different layers grow monotonically with depth of a cascade network; and (f) saliency maps from various layers with TE2E, showing the activated areas are relative scattered in comparison to TCL (see (d)).

than CL and TCL. Apart from visual difference of saliency maps, in Figure 6.9(e) we also show a quantification of this using a measure called the granulometry score (of which details are found in Section 6.3) (Dougherty et al., 1989). A larger value of this score means the activated contiguous regions (bright areas) of a image are more. This score (averaged on 50 randomly selected images) is computed over the saliency maps derived at various layers, and shows monotonic increase. When TL is applied, the fineness in the features is picked up earlier in the layers and continues to increase. But as we can see in later layers, this leads to the networks specialising to the source problem in which

they are trained, hence the features they capture are of little help in the target domain (as parts of mainly activated areas are not located at the lesion). For TE2E, this score is lower than the score of CL and TCL, as the scattered distribution.

6.6 Summary

In summary, while E2E training of DNNs currently is the default choice of practitioners, sequential, constructive approaches have merits in the important problem of TL where the focus of attention is improving performance on a target domain using limited computational resources. In this chapter, we show empirical evidence for TCL using both natural and medical images and explore the cases in which only limited availability of data exists. We also draw on insights using saliency maps and subspace correlations to compare the differences between different learning mechanisms. Our work suggests that the performance improvements of TL can be caused by CL and its variants, by packing coarse information about the source problem into early layers resulting in better transferability. Therefore, in the source domain we only need to train early layers saving on computational resources while also gaining improved performance.

Chapter 7

Conclusions and Future Work

The work reported in this dissertation is about an analysis and applications of the technique of Deep Cascade Learning (CL), a particular technique for progressive training of neural networks one layer at a time. We purse an information theory-based analysis (Information Bottleneck (IB)) to explore if trajectories of learning for layer-wise training show the same dynamics claimed to be an explanation of how neural networks generalise, put forward by Shwartz-Ziv and Tishby (2017). Using a range of problems, this work has shown that performance comparable to the more traditional End-to-End (E2E) training can be obtained without showing similar dynamics, particularly information compression, a strong claim advanced by Shwartz-Ziv and Tishby (2017). By doing this, we also offer a heuristic by which an optimal depth of a deep network can be automatically determined. We refer to this as the Information Transition Ratio (ITR).

We argue that the nature of layer-wise training forces early layers to learn coarse features of a problem and later layers learning specific ones. This view is indirectly confirmed by comparing performance of Deep CL with E2E learning on Transfer Learning (TL) problems. Learning fine features specific to a source problem may not be advantageous when transferring to a statistically related but different target domain. Such a view is confirmed using a wide range of problems in Human Activity Recognition (HAR), computer vision with natural scenes and two substantive medical imaging problems. Our results on the last two problems using small convolutional networks outperform results published by other authors using a large ResNet architecture. Next, we provide short summaries of the contents of each chapter.

7.1 Conclusions

In Chapter Two, we review learning in Deep Neural Networks (DNNs) with a view of motivating the analysis and applications of layer-wise (cascade) training.

Chapter Three is an analysis of learning in the information plane, an idea introduced and popularised by Shwartz-Ziv and Tishby (2017), building on earlier work on the IB principle. This appealing idea is proposed as a way of explaining how deep networks show high generalisation performance. Our work on studying CL in the information plane, leads to two conclusions. We contradict Shwartz-Ziv and Tishby's results that information compression of features learned in layers is not an explanation of generalisation. This follows from the fact that cascade trained models achieving the same performance in a range of problems do not show similar information compression. We further observe that the speed of change of the two information terms across layers happens in a way that their ratio could be used as a useful heuristic in determining an optimal depth of a network. To perform this study, we specifically construct several synthetic problems, which, unlike the example used in Shwartz-Ziv and Tishby's work could not be solved by a single hidden layer network. We also use several benchmark datasets in support of our empirical work.

Chapter Four addresses a TL problem of making inferences across biological species using Single Cell (SC) gene expression measurements. While this is a challenging and useful problem, and our results demonstrate the effectiveness of TL, the problem turned out to be solvable by a single layer network. Hence, disappointingly, the use of CL and the remarks we make about it do not apply to this problem.

In Chapter Five, we consider a HAR problem, setting up several TL tasks solved by E2E and cascade trained networks. The results obtained show that early layers of CL offer better representations to transfer knowledge to a target task. While this has also been observed with E2E trained networks by previous authors, E2E achieves this as a side effect of Stochastic Gradient Decent (SGD) training, whereas in CL, this is achieved by design. This is because early layers are simple architectures and when forced to learn the target are only likely to extract coarse features, with later layers specialising to details of the source domain. The empirical results on the HAR problem (and on the computer vision tasks, see Chapter Six) offer indirect support to this coarse-to-fine learning view.

In Chapter Six, we further apply Transfer Learning from Cascade Learning (TCL) to medical image classification tasks where data scarcity always exists. In comparison to Transfer Learning from End-to-End (TE2E) and learning from scratch, TL shows significant improvements when limited data is available. In particular, only the few early layers are able to provide the better transferability than a large-scale network ResNet (Raghu et al., 2019) on the CheXpert dataset and a similar conclusion is reached on the BIMCV task. Moreover, early layers has lower Singular Vector Canonical Correlation Analysis (SVCCA) similarities than later layers from cascade and E2E networks, which further validates the coarse-to-fine feature extraction in growing depth, as later layers get more specific features to targets in both learning mechanisms. Since features from early layers of cascade networks have significant transferability, there is no need to train the entire model on the source domain for implementing TCL, which further saves training

time and computational resources, compared to TE2E.

In summary, this work explores the difference between CL and E2E learning. In addition to the benefits of layer by layer feature extraction and different information planes of CL and E2E learning, we develop TCL and compare it to TE2E on a set of tasks, finding that TCL shows significant advantages in performance and in saving computation resources. We conclude that TCL is an alternative to conventional TE2E with adequate performance on limited data.

7.2 Future Work

In this work, we have analysed a particular way of training neural networks using the IB principle advanced by (Shwartz-Ziv and Tishby, 2017), thereby contradicting their view that dynamics on the information plane, especially information compression, are good explanations of generalisation in deep networks. While much of Shwartz-Ziv and Tishby's explanation is illustrated using a simple example, for which a multi-layer network is not necessary, we have constructed several synthetic examples and covered a range of real-world datasets to make our claims. Still, our experimental work is small in comparison to the large problems that are being solved in the current Deep Learning (DL) literature. As such, scaling up our work to tasks comparable to the ImageNet problem would be an immediate next step from this work. ImageNet itself is widely used (or over used) to carry out empirical work, hence other large scale problems in voice recognition or chemical property prediction could also be considered. Is the heuristic we recommend, based on ITR, at such large scales would be an important study to undertake.

Currently, we only propose ITR for deciding the depth of a model, whilst the width of each layer is selected heuristically. Inspired by works of Fahlman and Lebiere (1990) and Platt (1991), it is possible to construct a layer by adding units one by one. We believe the difference of ITR before and after adding a unit can be used to judge the necessity of this unit in constructing a layer. That can be a potential way to realise the automatic adaption of a network structure instead of a tedious search from a list of structures. Moreover, features from such cascade networks may be able to further enhance transferability.

Alternate forms of CL have been considered in the literature recently. Of particular note is the specification of local error functions Wang et al. (2021) and layer-wise decoupled training based on InfoMax principle (Löwe et al., 2019). Of these, the InfoMax principle is closely related to the work we have carried out. Exploring the links between the two would be a useful avenue to purse.

In addition, a proper learning rate scheme may be another way to improve the learning ability of CL. As Ro and Choi (2021) suggested, the fixed learning rate scheme over

layers may hinder the notion that early layers extract general features and later layers extract specific features in the setting of E2E learning. We believe it is worth developing a layer-wise learning rate schedule for achieving a state-of-the-art performance of CL.

CL is a good framework to explore hierarchy in class labels, when available. Several of the image classification tasks are hierarchical (families of dogs and vehicles, for example). While we have touched on this problem in this work and shown that early layers learn much better when the problem posed is at the root of the hierarchy, we have not explored this in any detail. The ImageNet problem and the medical problem of CheXpert have natural hierarchical structures in them. We also used clustering of semantic vector representations to establish a hierarchy. Exploring this in detail will be another line of work well justified by the findings in this dissertation.

Finally, broader applications of TCL such as natural language processing, should be extended, to investigate the adaption of CL in recurrent structures. Doing so, will help scale up repositories of CL and TCL.

Appendix A

This appendix is a supplementary of Chapter 3. Here we show details of data, configurations of models, extended results and derivations related to Information Bottleneck (IB) theory and Singular Vector Canonical Correlation Analysis (SVCCA).

A.1 Datasets and Configurations

Datasets used can be divided into three parts, synthetic data, realistic data from UCI repository and benchmark datasets in the area of computer vision (as an aforementioned summary in Section 3.4.1). In this section, we provide unmentioned details of realistic datasets used in Chapter 3 and the corresponding configurations of models on all used datasets.

A.1.1 Details of Realistic Datasets

Dexter (Guyon et al., 2004) is a dataset to filter texts about "corporate acquisitions" by solving a two-class text classification problem. The inputs are sparse continuous representation given by a bag-of-word embedding, and the output are balanced binary labels. There are 2562 features out of total 9947 original features are always zeros where the features represents the frequencies of occurrence of the word stems. Thereafter, the data is reconstructed by adding 10053 simulated features according to Zipflaw of which the details can be found in (Guyon et al., 2004). Finally, the fraction of non-zero values in the data is around 0.5% with 20000 features. Originally, there was three parts: training data, validating data and testing data, however the testing data are still withheld. Hence, in our experiments we use first two parts of data and make validation data as an unseen set to test models.

Dorothea (Guyon et al., 2004) is a drug discovery dataset for recognising which compounds bind to Thrombin. The binary sparse inputs are descriptions of chemical compounds given by the three-dimensional structural properties of molecules, and targets are predictions of active or inactive ability to bind to Thrombin. The data has $139,351$

original features and only $100K$ top ranking features are kept based on a described criterion in (Guyon et al., 2004). Finally, the non-zero values of data are less than 1%. There are 800 compounds in training data and 350 compounds in testing data.

Gisette (Guyon et al., 2004) is a handwritten digit recognition problem being similar to MNIST. The problem is to discriminate the highly confusable digits "4" and "9" where digits have been size-normalised and centred in a fixed-size image with dimension 28×28. Similar to the above two datasets, the data is reconstructed for the purpose of the feature selection challenge. Particularly, pixels in the middle top part of the features are randomly sampled for obtaining information necessary to disambiguate 4 from 9. Then, the creation of higher order features is implemented by products of sampled pixels to solve the problem in a higher dimensional feature space. In addition, a number of distractor features are also added with random orders. In total, 5000 features are included in this dataset with around 13% non-zero values and a binary classification task is constructed.

Epileptic (Andrzejak et al., 2001) is a commonly used dataset for detecting epileptic seizure. It primitively contains 5 classes and we reconstruct it to a binary classification task for recognising if there has or has not been an epileptic seizure. Inputs consists of data points with the value of the electroencephalo-graph recording at different points in time. From 500 individuals, 4097 data points (each one for 23.5 seconds) can be divided into 23 chunks of which each has 178 data points for a second. Therefore, a dataset with $23 \times 500 = 11500$ rows and 178 dimensions is obtained. More details can be found in the UCI repository.

Human Activity Recognition (HAR) (Anguita et al., 2013) is a dataset obtained from 30 volunteers who carried a waist-mounted smartphone with embedded inertial sensors. In total, there are six activities (walking, walking upstairs, walking downstairs, sitting, standing and laying) based on data captured by 3-axial linear acceleration and 3-axial angular velocity at a constant rate of $50Hz$. Data is pre-processed through noise filters and sampled using fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). More details of data can be found in the UCI website. According to the constructed training and test data given by the website, we evaluated a 6-class task. We also gave a feature selection by ranking the correlation between each feature and targets in a declining order, and the top 40% of features (around 225 features) are used in our experiments.

A.1.2 Training Parameters of All Datasets

Obtaining optimal performance on each of the datasets by extensive pruning of structures is not the aim of this section, therefore we heuristically select architectures for obtaining reasonable performance of both Cascade Learning (CL) and End-to-End (E2E) learning on all datasets in order to provide a fair comparison of them.

TABLE A.1: Configurations of networks for all datasets.

Dataset	Architecture[1]	LR[2]	Dataset	Architecture	LR
Synthetic	[10,7,5,3]	0.001	Epileptic	[32,16,14,12,8,4]	0.01
Dexter	[6,5,4,3]	0.01	Gisette	[32,16,14,10,8,6]	0.001
Dorothea	[20,10,7,3]	0.001	MNIST	model 1:[20,20,20,20,20]	0.01
				model 2: [1024,20,20,20]	0.03
HAR	[10,32,16,10,8,6,4]	0.1	CIFAR 10	layer 1-8:256 layer 9-12:128 (CNN)	0.001
ImageNet23	[256,256,256,256,256,256,256]	0.01			

1 The selection of structures is heuristic based on dozens of simulations or recommendations from publications.
2 LR is the abbreviation of learning rate.

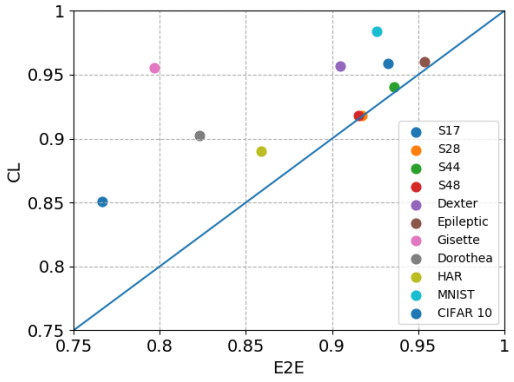


FIGURE A.1: The comparison of performance between CL and E2E on all datasets with a heuristic selection of structures. On the same dataset, the configurations of training and network architectures of both CL and E2E are same. Based on heuristic selection of model size without repeat tweak, CL shows significant advantages compared to E2E learning when over-fitting happens.

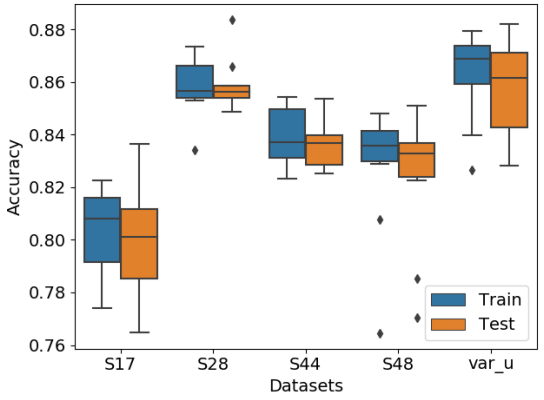


FIGURE A.2: Comparison of linear classifiers' performance over synthetic datasets. *var_u* is the data used by Shwartz-Ziv and Tishby (2017).

Table A.1 shows structures of models on each dataset. An Adam optimiser is used on majority of datasets except MNIST and CIFAR 10 which uses a Stochastic Gradient Decent (SGD) optimiser, accompanied by a learning scheduler with a 0.9 decay on each epoch. Softmax is the activation function of classifiers/last layer while the activation function of the middle hidden layers is Hyperbolic Tangent Function (Tanh) except in the convolutional network on CIFAR 10 and ImageNet23 whose activation function is Rectified Linear Unit (ReLU). The kernel size of convolutional layers is 3×3. The structure and configurations of the model used to train MNIST are same as described in (Noshad et al., 2019). For ImageNet23, we selected 23 classes of images, which overlapping with CIFAR 100, from 1000 classes. The size of each hidden layer used to train ImageNet23 is shown in Table A.1. For CIFAR 10 and ImageNet, an auxiliary classifier is used after the convolutional layers. The auxiliary classifier for CIFAR 10 includes two fully connected layers consisting of 64 and 32 units leading to the output layer. For ImageNet23, we use one convolutional layer with 128 units and two fully connected layers with 256 units for each are included. For all of other datasets, the output layer is the classifier and its size is decided by the number of classes of tasks. Particularly, as the dimension of the **Dexter** and **Dorothea** datasets are much higher than the number of data in them, we add $L2$ $(1 \times e^{-5})$ regularisation to mitigate over-fitting.

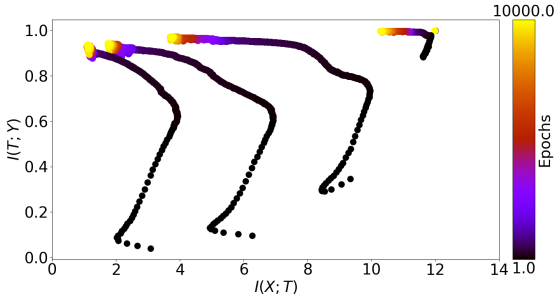


FIGURE A.3: An independent implementation of Shwartz-Ziv and Tishby's work showing dynamics on the information plane of an E2E network, with a Tanh activation function and SGD optimiser. The trajectories are averaged over 10 runs using random initialisation.

As mentioned in Chapter 2, CL provides the flexibility of structure adaptation, while E2E does not have. Using a heuristically selected structure of networks trained by E2E and CL respectively, Figure A.1 shows corresponding performance on each of dataset used in this section. On majority of tasks, CL shows better performance with the heuristic structure without needs of searching the suitable topology of networks.

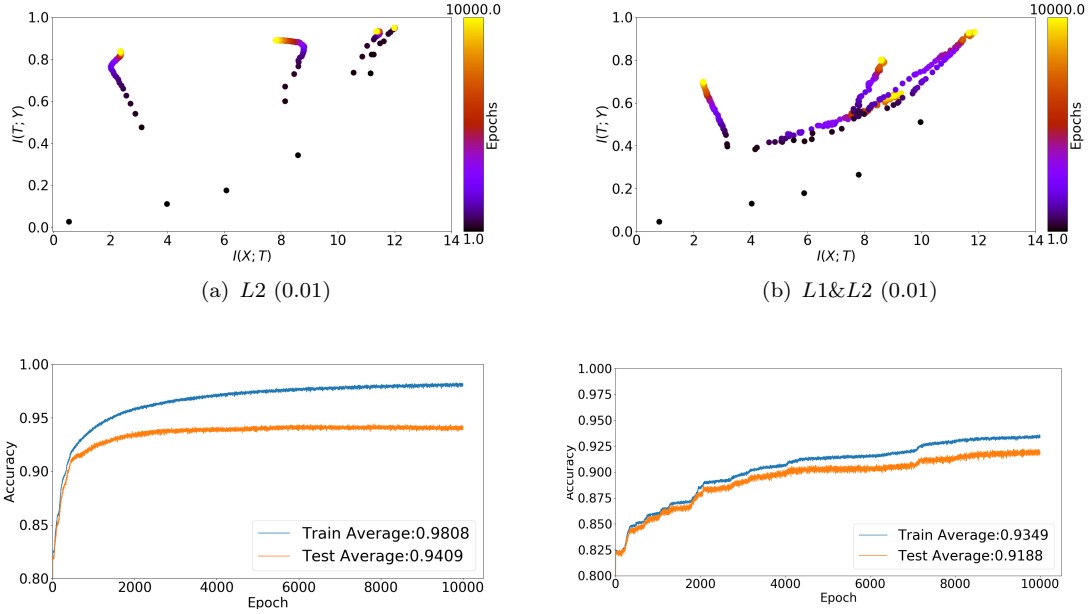
To validate the complexity of different synthetic datasets, we use a Support Vector Machine (SVM) with the linear kernel to produce each synthetic dataset. As shown in Figure A.2, our generated synthetic datasets are relative harder than the task used by Shwartz-Ziv and Tishby (2017).

A.2 Repetition on Shwartz-Ziv and Tishby's Data for End-to-End Training

For validating the implementation of estimators, we firstly give an recreation of partial works in (Shwartz-Ziv and Tishby, 2017). Figure A.3 shows the reproduced information plane by us based on binning estimation on Shwartz-Ziv and Tishby's data. The structure is same as settings in (Shwartz-Ziv and Tishby, 2017), and obtained planes are analogous. This is a solid foundation of following works shown in Section 3.4.

A.3 Extended Analyses of Information Bottleneck Planes for End-to-End Learning

Compared to the information planes of E2E learning shown in Section 3.5.1, we give an further exploration of E2E learning by adding regularisation and changing the depth of the networks. In this section, the results of all experiments are averaged over multiple runs and the information plane of E2E learning includes the trajectory of the prediction layer which is the closest curve to the y-axis.



(a) $L2$ (0.01)

(b) $L1\&L2$ (0.01)

(c) Performance with $L2$ (0.01)

(d) Performance with $L1\&L2$ (0.01)

FIGURE A.4: Information planes of E2E learning with regularisation on the synthetic 17 dataset. (a) and (c) respectively show the information plane and performance of an E2E network with ridge regularisation. (b) and (d) show same content while regularisation contains both lasso and ridge. Compared to information planes shown in Figure 3.3(a), (a) shows less behaviours of information compression over layers and better generalisation. (b) shows invisible information compression from layers (except the last layer).

A.3.1 Effects of Regularisation in End-to-End Learning

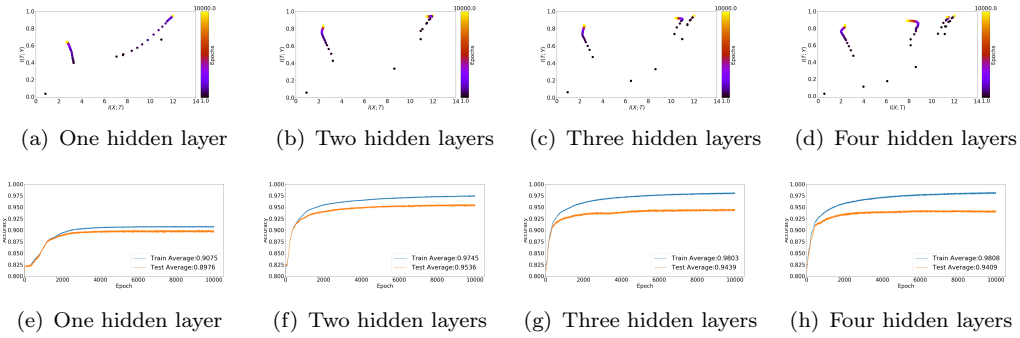
We use two types of regularisation, $L2$ (ridge) and $L1\_L2$ (lasso and ridge), to apply a penalty on the layer's kernel, with all configurations using a factor of 0.01. Compared to the results shown on Figure 3.2(b), Figure A.4(a) shows less information comparison with smaller $I(Y;T)$ values. Comparing Figure 3.3(a) to Figure A.4(c), generalisation is improved by adding ridge regularisation. This observation further supports our conclusion that generalisation cannot directly benefit from information compression or solely high values of $I(Y;T)$. When both lasso and ridge regularisation are applied to networks, the model is under-fitted as shown in Figure A.4(d) in comparison to the model only has lasso regularisation. The corresponding information plane also shows deformation as shown on Figures A.4(b), where information compression is negligible and the trajectories between some layers cross. Here, the crossed trajectories may be a results from the flexible communication between layers, and can be an interesting direction to explore, although it is out of our scope. Chelombiev et al. (2018) also explore the effects of $L2$ regularisation on trajectories on information planes, but they only pay attention on $I(X;T)$ and ignore the change of $I(Y;T)$.

A.3.2 Effects of Adding Layers in End-to-End Learning

Instead of using a fixed structure to train an E2E network, we explore the effects of gradually enlarging the E2E networks' size on the information plane. From Figure A.5, we observe that the model containing two hidden layers shows best generalisation where information compression is imperceptible. Afterwards, increased depth of models does not contribute to generalisation but shows slight improvement of training accuracy and $I(T;Y)$. Some of the middle layers will show information compression. This also indicates information compression cannot be the main/only reason of good generalisation, and the sole consideration of $I(X;T)$ or $I(T;Y)$ may not be the proper way to connect IB theory and generalisation. Figure A.6 shows effects of adding layers to E2E networks on three datasets, which indicates the difficult of searching an optimal structure of E2E networks.

A.4 Results on Other Datasets

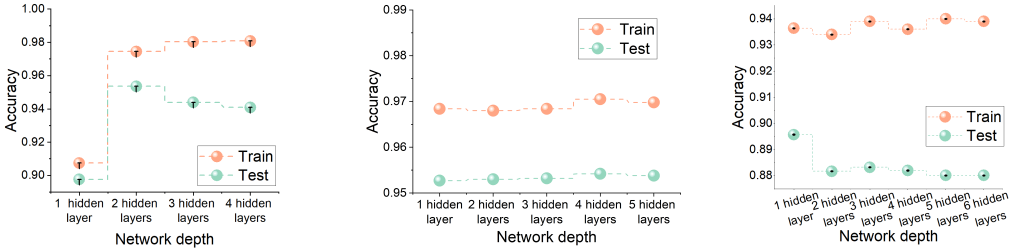
In the main body, we illustrate differences of information planes between CL and E2E training on one of synthetic datasets and the HAR data in Section 3.5.1. In this section, we further provide the related results on other datasets.



(a) One hidden layer (b) Two hidden layers (c) Three hidden layers (d) Four hidden layers

(e) One hidden layer (f) Two hidden layers (g) Three hidden layers (h) Four hidden layers

FIGURE A.5: Information planes and performance of E2E learning with different structures on synthetic 17. Using a L2 regularisation with a factor 0.01, the deepest network has four layers with the number of units in sequence 10,7,5,3.



(a) Synthetic 17 (b) Epileptic (c) HAR.

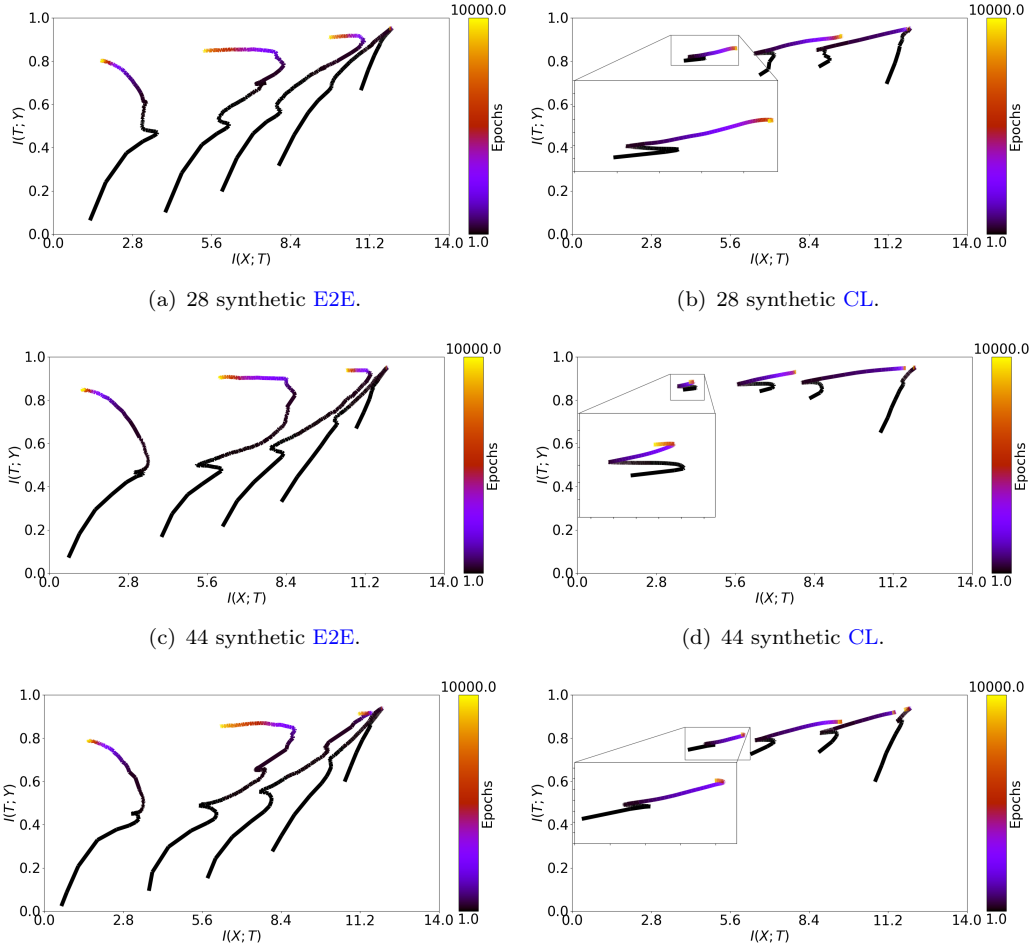
FIGURE A.6: Performance of E2E learning with different structures (gradually increasing) on different datasets. The structure is changed by gradually increasing the number of hidden layers. The accuracy is obtained by training models adding a L2 kernel regularisation with a constant 0.01, and datasets are synthetic 17, Epileptic and HAR in sequence.

A.4.1 Information Planes

Figure A.7 shows information planes of CL and E2E on remaining three synthetic datasets, where the characteristics of both learning mechanisms are consistent with Figure 3.2. In that case, Mutual Information (MI) is estimated every epoch within the first 200 epochs, afterwards it is estimated every 100 epochs, using the binning estimation. The trajectories of the final prediction layers are shown as the most left curve on the information plane of E2E learning.

In Figure A.7, CL shows invisible information compression along the $I(X;T)$ axis once the early layers have been trained, while E2E has visible information compression on the same axis. The starting points, regarding $I(T;Y)$, of both learning mechanisms is different where E2E starts from initialisation but CL starts on the foundation of previous learned layers. Both learning mechanisms have a slight loss of $I(Y;T)$ from early to later layers.

Figure A.8 presents the same information planes on CIFAR 10 where the model contains convolutional layers associated with ReLU activation functions. The information is es-

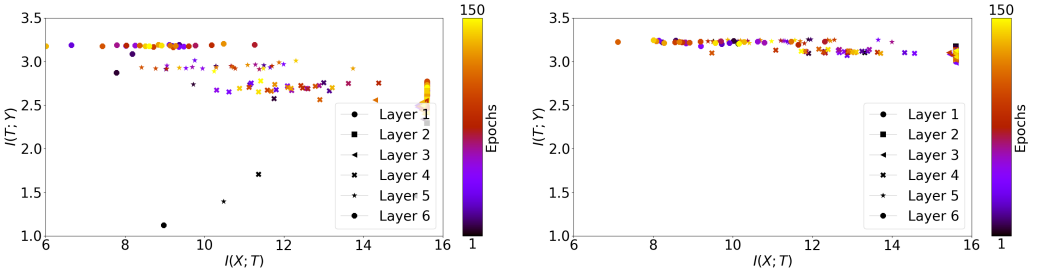


(a) 28 synthetic E2E. (b) 28 synthetic CL.

(c) 44 synthetic E2E. (d) 44 synthetic CL.

(e) 48 synthetic E2E. (f) 48 synthetic CL.

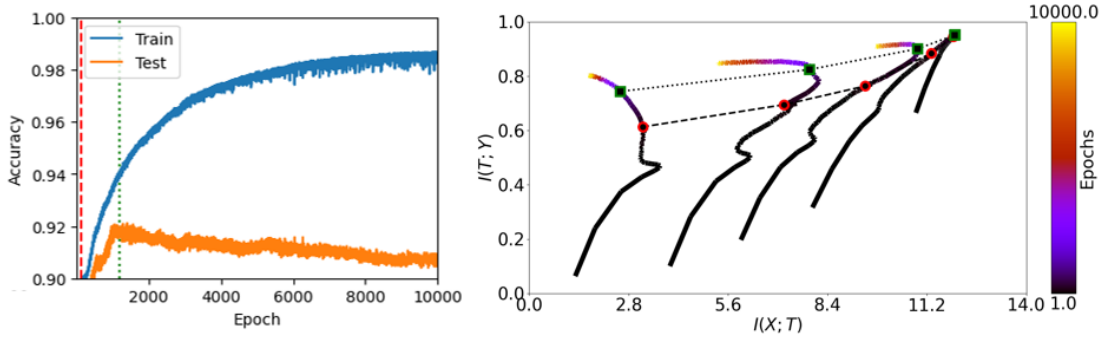
FIGURE A.7: Comparison of information planes of CL and E2E on synthetic datasets. In comparison to E2E learning, there is limited visible information compression dynamics at the end of training in CL's information planes while their performance is similar.



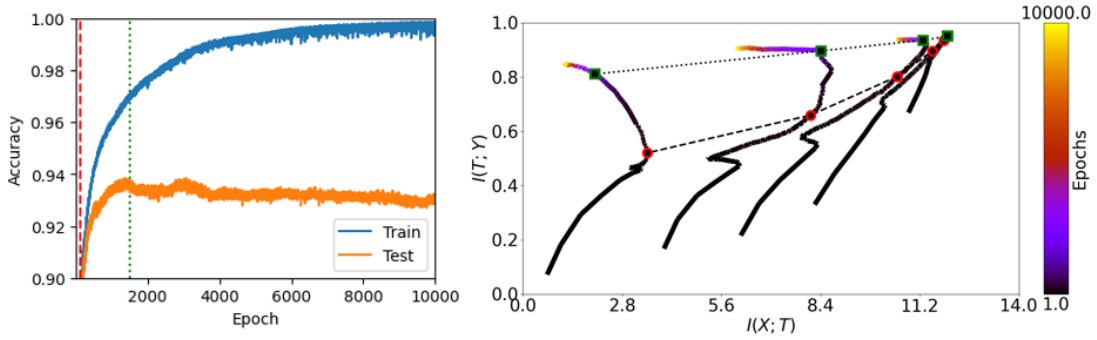
(a) E2E (b) CL

FIGURE A.8: Information planes on CIFAR 10. (a) and (b) show trajectories of E2E learning and CL respectively. For E2E, we got similar information planes as shown in (Jónsson et al., 2020), which do not show the same smooth behaviour as simpler synthetic datasets. Information compression happens both before and after over-fitting.

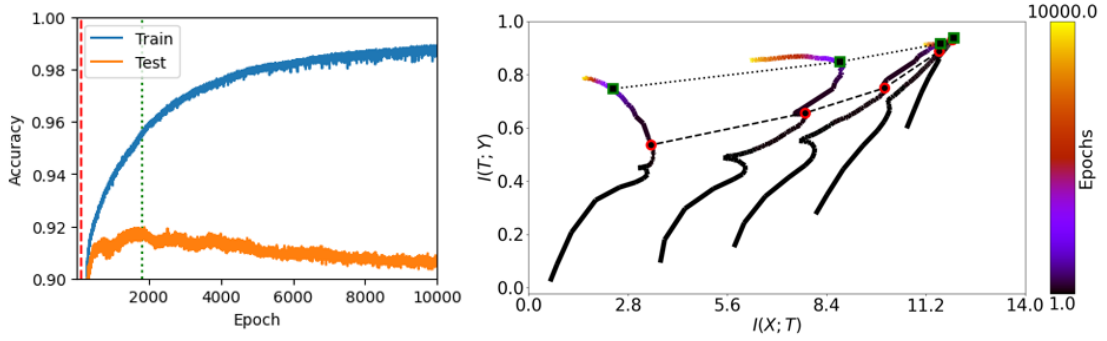
timated for every 5 epochs using Ensemble Dependency Graph Estimator (EDGE) and averaged over at least 2 runs (see Section A.1.2 for details of network configurations). We observe although trajectories of this case are not as smooth as the others, the characteristics are consistent such as the starting points on the information plane of both learning mechanisms. CL do not contain the trend of information compression while E2E does but only slightly.



(a) Synthetic 28



(b) Synthetic 44



(c) Synthetic 48

FIGURE A.9: A comparison between information compression and generalisation on synthetic datasets. Corresponding to the comparison shown in Figure 3.3, this figure shows further comparison between the performance of E2E learning on test datasets (from synthetic 28, 44 and 48) and trajectories on information planes. Similar to Figure 3.3, the dashed line and the dotted line highlight the epoch starting inflection and starting over-fitting respectively.

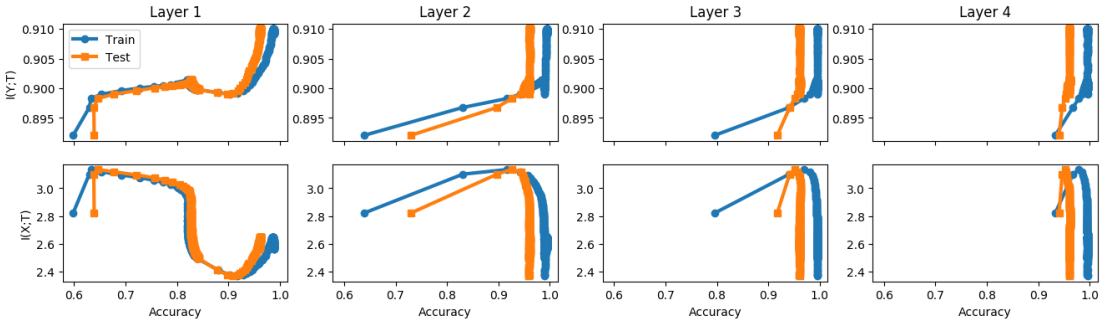
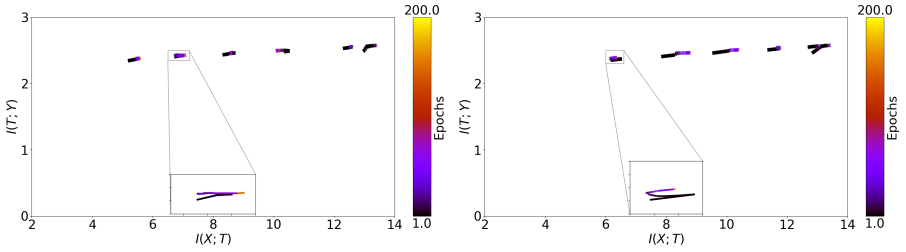


FIGURE A.10: An exploration of connections between information compression and generalisation of CL on synthetic 17 data. Briefly, there is also not clear connection between the information compression and the performance of CL.



(a) No regularisation (b) regularisation($1e^{-5}$).

FIGURE A.11: A comparison of CL information planes between with and without regularisation on HAR data. The regularisation is a penalty on each layer's output with a factor ($1e^{-5}$). By comparing the two sub-figures, adding regularisation increases the value of $I(X;T)$ for all layers except first layer while the trend of absent information compression is consistent. This increment does not further benefit to the accuracy. All results are averaged over 5 runs.

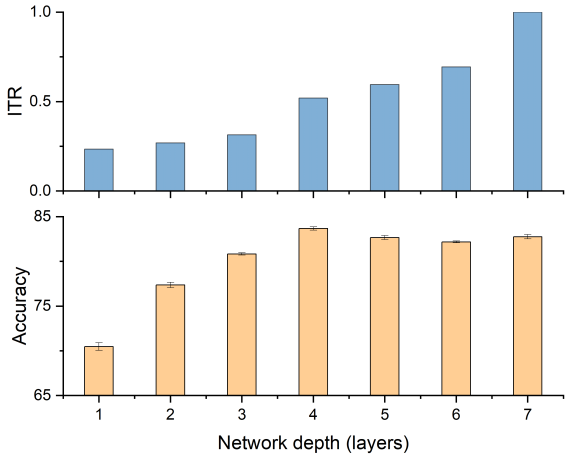


FIGURE A.12: The ITR of CL on the ImageNet dataset. 23 classes are selected in this classification task for saving training time on this large task. The top figure shows a tendency of ITR, and the bottom figure shows performance on test sets, throughout layers of the network. The accuracy achieves an optimal value at the forth layer where ITR shows a sharp increase. Afterwards the forth layer, the accuracy starts to decrease. The conjunction of testing accuracy decreasing and ITR sharp increasing is consistent with Figure 3.4.

A.4.2 Information Compression and Generalisation

We extend our investigations to find when information compression happens in E2E. We observe visible information compression happens after the saturation of test performance as shown in Figures 3.3(a) and 3.3(b). Hence, there may be a suggestion that the information compression happens after over-fitting as mentioned by Wickstrøm et al. and in Section 3.6. Figure A.10 shows a similar comparison between compression and generalisation of CL, from which there is no clear tendency.

Taking the HAR task as an example, we further track how trajectories change with and without regularisation in CL. As shown in Figure A.11, we notice adding proper regularisation may be able to increase values of $I(X;T)$ in CL, which is same as mentioned in Section A.3. This observation further supports our conclusion that information compression may be not the main/only reason of better generalisation.

A.4.3 Information Transition Ratio on ImageNet

Furthermore, Figure A.12 shows Information Transition Ratio (ITR) of CL on a 23-class ImageNet task. Similar to results on other datasets shown in Section 3.5.2, the layer with a sharp increase of ITR can suggest the appropriate depth of cascade networks, from which a satisfactory performance can be obtained.

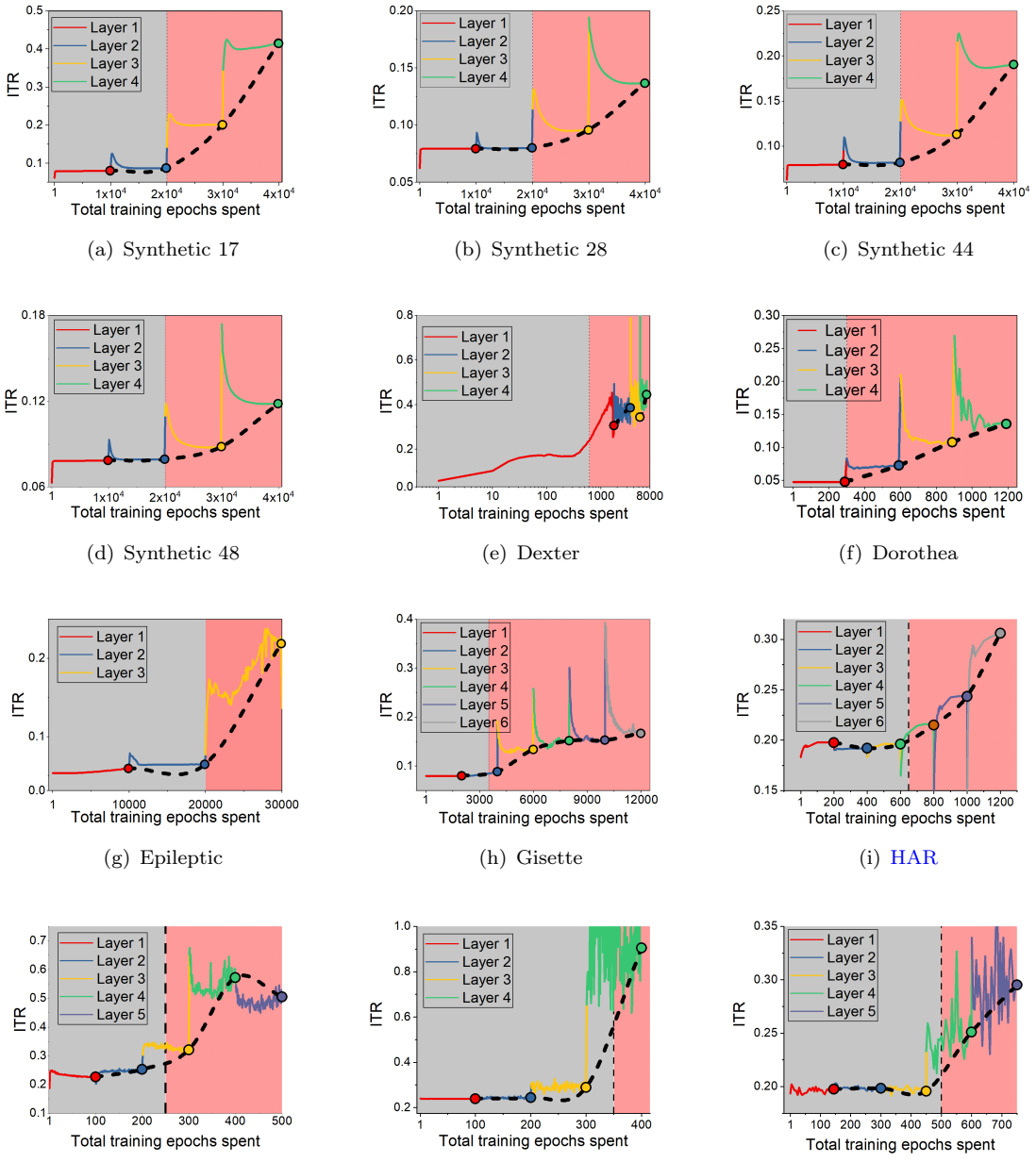
A.4.4 Information Transition Ratio Dynamics of Cascade Learning

As the extension of Table 3.2, Figure A.13 shows the dynamics of ITR on both synthetic and realistic datasets. The layers, having sharp increases of ITR, of networks can produce the satisfactory performance, and the subsequent layers will not significantly contribute to the improvement of performance.

A.4.5 Comparison of Subspaces between End-to-End and Cascade Learning.

On the one hand, we use information planes to track learning dynamics of both learning mechanisms as described in Section 3.5.1. On the other hand, we visualise the latent representations of both CL and E2E through projecting representations onto two dimensional planes, using T-distributed Stochastic Neighbor Embedding (TSNE) (of which the perplexity is 30) from a library, *sklearn*, projections are shown in Figure A.14. For the first layer, features, from both CL and E2E, have low separability at the beginning of training and then gradually increase with epochs. For subsequent layers, CL shows relatively higher separability of features from the beginning of training as features from later

layers are on the top of previous trained features. However, for E2E learning, the features have low separability at the beginning of all layers' training. For middle layers, CL has better separability of features, compared to E2E. Geiger (2021) believed the information compression of E2E is caused by geometric clustering. However, in our observations, CL shows more visible separability in clustering, where information compression is invisible as shown in Figure 3.2. Therefore, a further discussion in that direction may be an open question.



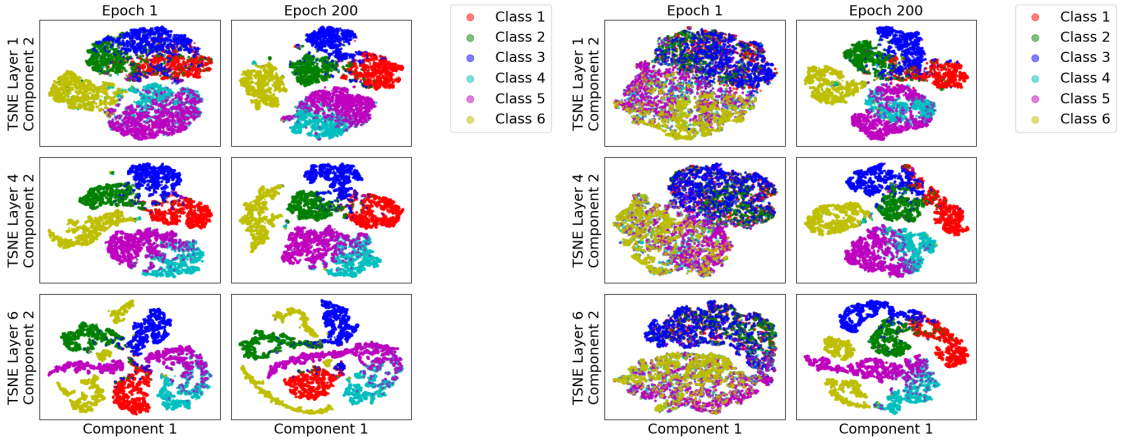
(a) Synthetic 17 (b) Synthetic 28 (c) Synthetic 44

(d) Synthetic 48 (e) Dexter (f) Dorothea

(g) Epileptic (h) Gisette (i) HAR

(j) MNIST (model 1) (k) MNIST (model 2) (l) CIFAR 10

FIGURE A.13: ITR dynamics with respect to total training epochs on both synthetic and realistic datasets. The circles with black margin marks the last training epoch of each layer. The shading areas show learning (grey) and over-fitting (pink) areas respectively.



(a) CL (b) E2E

FIGURE A.14: Visualisations of representations from intermediate layers based on TSNE. The models are trained on HAR problems. Each row of a sub-figure stands for a layer, and each column stands for an epoch. From left to right, we present the 1_{st}, 50_{th} and 200_{th} epoch in the sequence. From top to bottom row, there are the 1_{st}, 4_{th} and 6_{th} layer in the sequence. (a) shows representations from CL while (b) is from E2E learning. By comparing the first column of both sub-figures after the first layer, we notice CL provides more distinguishable features at the beginning of training each layer. Reflected on the information plane, the starting point of $I(Y;T)$ is relatively high. For early middle layers, the features extracted by CL have higher separability.

Algorithm 3 Pseudocode of the Binning Estimation

Arguments:
  X: Inputs, where $\boldsymbol{x} \in X$.
  T_i: The output/representations from i_{th} layer, where vector $\boldsymbol{t}_i \in T_i$.
  Y: Labels, where $y_c \in Y$, $c \in N_2$ is the index of the classes.
Outputs:
  $I(X;T_i)$: MI between X and T_i.
  $I(T_i;Y)$: MI between Y and T_i.
Initialisation:
  N_1: The number of clusters in discrete T_i.
  $P(*)$: The distribution of $*$.
  $p(*)$: The probability of $*$.
  $H(*)$: The entropy of $*$.
  $I(*)$: The MI of $*$.
At each epoch of the i_{th} layer:
  Discretise T_i by binning each dimension of T_i into intervals between $[-1, 1]$.
    {We use 25 intervals according to minimal sufficient statistics (Fisher, 1922).}
  $P(T_i) \leftarrow$ *Calculate the distribution of the discretised* T_i.
  $H(T_i) \leftarrow \sum_{j=1}^{N_1} p(\boldsymbol{t}_{ij}) I(\boldsymbol{t}_{ij}) = -\sum_{j=1}^{N_1} p(\boldsymbol{t}_{ij}) \log_2 p(\boldsymbol{t}_{ij})$
  $H(T_i|X) \leftarrow 0$ {As T_i is the deterministic function of X (Amjad and Geiger, 2018).}
  $H(T_i|Y) \leftarrow -\sum_{c=1}^{N_2} p(y_c) \left[\sum_{j=1}^{N_1} p(\boldsymbol{t}_{ij}|y_c) \log_2 p(\boldsymbol{t}_{ij}|y_c) \right]$
  $I(X;T_i) \leftarrow H(T_i) - H(T_i|X)$
  $I(T_i;Y) \leftarrow H(T_i) - H(T_i|Y)$

[3]Kullback-Leibler (KL) divergence (Cover and Thomas, 1999) based upper bound (Kolchinsky and Tracey, 2017) on entropy of mixture of Gaussians with covariance matrix $\delta^2 * I$. The Bhattacharyya

Algorithm 4 Pseudocode of the PWD Estimation (Kolchinsky and Tracey, 2017; Kolchinsky et al., 2019)

Arguments:
X: Inputs, where $\boldsymbol{x} \in X$, N_s is the number of samples in X.
Y: Labels, where $y_c \in Y$, $c \in N_c$ is the index of classes.
T_i: The representations from i_{th} layer, where vector $\boldsymbol{t}_i \in T_i^{N_s \times d_{T_i}}$, and d_{T_i} is the dimension of T_i.
δ^2: Noise variance.
Outputs:
$I(X;T_i)$: MI between X and T_i.
$I(T_i;Y)$: MI between Y and T_i.
Initialisation:
$H(*)$: The entropy of $*$.
$I(*)$: The MI of $*$.
$p(y_c) \leftarrow$ *The probability of each class.*
At each epoch of the i_{th} layer:

$$H_{KL}(T_i) \leftarrow -\frac{1}{N_s}\left[\sum_{r=1}^{N_s} log\left(\frac{1}{N_s}\sum_{c=1}^{N_s} exp\left(-\frac{\| \boldsymbol{t}_{ir} - \boldsymbol{t}_{ic} \|_2^2}{2 * \delta^2}\right) - \frac{d_{T_i}}{2} log(2\pi\delta^2)\right)\right] + \frac{d_{T_i}}{2}$$

{From the Kullback-Leibler (KL) divergence[3].}

$$H_{KL}(T_i|X) \leftarrow \frac{d_{T_i}}{2} * log(2\pi\delta^2 + 1)$$

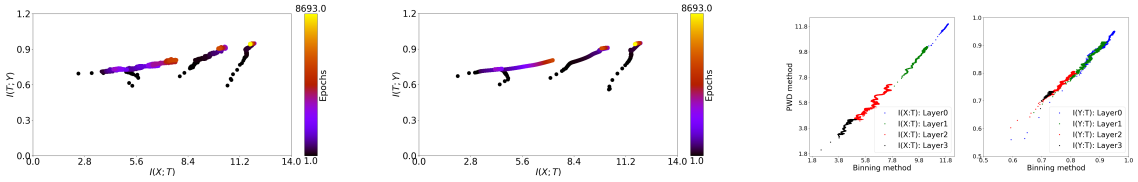
{The entropy of a multivariate Gaussian distribution (Ahmed and Gokhale, 1989; Huber et al., 2008)}

$$H_{KL}(T_i|Y) \leftarrow \sum_{c=1}^{N_c} p(y_c) H_{KL}(T_i|y_c)$$

{Similar to $H_{KL}(T_i)$, $H_{KL}(T_i|y_c)$ is obtained by using the conditional $(T_i|y_c)$ substitute T_i.}

$$I(X;T_i) \leftarrow H_{KL}(T_i) - H_{KL}(T_i|X)$$

$$I(T_i;Y) \leftarrow H_{KL}(T_i) - H_{KL}(T_i|Y)$$



(a) Binning. (b) PWD. (c) Two estimators' comparison.

FIGURE A.15: The comparison of the two estimators of MI. The data is synthetic 17. The number of epochs is decided by early stopping.

A.5 Estimation of Mutual Information

In this section, we give details of three estimations used in Section 3.4.2, binning, Pair Wise Distance (PWD) and EDGE. Algorithm 3 shows binning estimation. Algorithm 4 and 5 show PWD and EDGE based estimation of MI respectively. In the implementation of binning estimation, the number of intervals is 25 for most of the realistic data except synthetic and HAR data of which a comparison among three different intervals (10, 25, 30) is provided. Similarly, we carry out PWD estimations on most of the realistic data by setting the noise variance as $1e^{-3}$ while comparing difference over $1e^{-2}$, $1e^{-3}$ and $1e^{-5}$ on the synthetic and HAR data.

[3] distance based lower bound is $H_{KL}(*) + log(\frac{1}{4}) * \frac{d}{2}$.

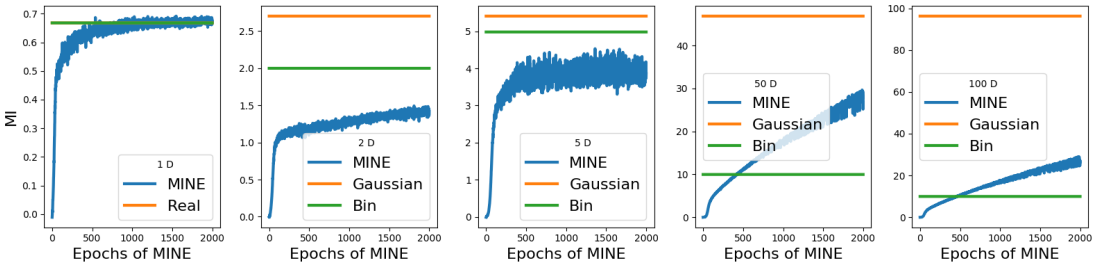


FIGURE A.16: The estimation of MI over several high dimensional Gaussian artificial data based on MINE estimators (Belghazi et al., 2018). From right to left, the dimension of data is in sequence 1,2,5,50,100. The one dimensional data is randomly generated according to a Gaussian distribution where the real estimation means MI between Gaussian variables given by the mean and variance. Binning estimation is based on 30 intervals. Other high dimensional data is also constructed by random generation following Gaussian distributions.

Algorithm 5 Pseudocode of the EDGE (Noshad et al., 2019)

Arguments:

X: Inputs, where $\boldsymbol{x} \in X^{N_s \times d}$, N_s is the number of samples in X.

Y: Labels, where $y_c \in Y^{N_s \times N_c}$, $c \in N_c$ is the index of classes.

T_i: The representations from i_{th} layer, where vector $\boldsymbol{t}_i \in T_i^{N_s \times d_{T_i}}$, and d_{T_i} is the dimension of T_i.

Outputs:

$I(X;T_n)$: MI between X and T_n.

$I(T_n;Y)$: MI between Y and T_n.

Initialisation:

H: Hash map of X and Y, and $H : \mathbb{R}^d \rightarrow \{1, \ldots, F\}$.
Where F is the number of buckets and is a linear function of N_s.
$H(*)$ specifies a vertex index of a dependence graph.

$H(u) = [h(u_1), h(u_2), \ldots, h(u_d))]$. Where u is the each component of vector (e.g., $\boldsymbol{x}$).
Where $h(u) = [\frac{u+b}{\epsilon}]$, $b \in [0, \epsilon]$ is a random number and ϵ is a bandwidth parameter.
{Using Locality-Sensitive Hashing algorithm (Indyk et al., 1997), H maps neighbouring points to a common value.}

$I(*)$: The MI of $*$.

At each epoch of the n_{th} layer:

Create a bipartite graph with two sets of nodes U and V.

Map points in X and T_n to nodes in U and V using H.

Assign the weights ω_i and $\omega_j^{'}$ respectively to nodes v_i and u_j, , where $v_i \in V$ and $u_j \in U$.

$\omega_{ij} \leftarrow$ The weights of the edge (v_i, u_j).

$\omega_i = \frac{N_i}{N}$, $\omega_j^{'} = \frac{M_j}{N}$ and $\omega_{ij} = \frac{N_{ij}/N}{(N_i/N)(M_j/N)}$.
Where N_i and M_j are the number of hashing collisions (instances) at v_i and u_j.
N_{ij} is the number of pairs $(\boldsymbol{x}, \boldsymbol{y})$ mapped to (v_i, u_j), and $0 \leq N_{ij} \leq N_i, N_j$.

The estimates of f_i, f_j and $f_{ij}/f_i f_j$[4] are ω_i, ω_j, and ω_{ij}.

$\hat{I}(X;T_n) \leftarrow \sum_{e_{ij} \in E_G} \omega_i \omega_j^{'} g(\omega_{ij})$. Where e_{ij} is $v_i \rightarrow u_j$ edge in the set E_G.

$\boldsymbol{I}(X;T_n) \leftarrow \sum_{t \in \tau} w(t) \hat{I}(X;T_n)$.

Where w is optimised by minimising $\|w(t)\|_2$ being subject to $\sum_{t \in \tau} w(t) = 1$
and $\sum_{t \in \tau} w(t) t^i = 0$, $i \in \mathbb{N}$, and $i \leq d$. τ is a set of index values Moon et al. (2016).

$I(T_n;Y) \leftarrow$ Repeat steps 13 to 20 with variables T_n and Y.

[4]For two variables X and Y, the general MI between them is $I(X;Y) \equiv \int g(\frac{f_1(x) f_2(y)}{f_{12}(x,y)}) f_{12}(x,y) dx dy$, where g is a smooth convex function with $g(1) = 0$. For Shannon MI, $g(x) = x \log x$.

A.5.1 Comparison between Estimators

For validating the effects of different estimators, we use both binning estimation and PWD estimation to draw the information plane of a CL network on each realistic dataset and synthetic dataset. Limited by the space, we only show a comparison on the synthetic 17 dataset as an example on Figure A.15. We notice that the two estimators show similar trajectories in the information plane on a majority of the data, and the value of estimated MI lays on the diagonal line of Figure A.15(c). However, in some datasets, the robustness is not so ideal and effected by the selection of parameters of each estimators the estimation may vary. Therefore, we utilise the same estimator on all datasets to estimate the MI of both CL and E2E for fair comparison and reduction of uncertainties (as mentioned in Section 3.4.2).

We also try the estimator Mutual Information Neural Estimation (MINE) proposed by (Belghazi et al., 2018), however this estimator shows an unstable estimation with an trend towards infinite as shown in Figure A.16 which is similar as the description in (Elad et al., 2019). Except one dimensional data, the estimated value of MI keeps increasing with training estimators. Namely, MINE cannot provide a stable estimation in these cases. Furthermore, for each estimation, an extra network would need to be trained for at least 2000 epochs, which would be quite time consuming. Therefore, we did not use this estimator in our experiments.

A.5.2 Mutual Information of Gaussian Variables

As mentioned in Section 3.5.2, we also estimate MI by assuming all variables are from Gaussian distribution. Equation A.1 shows the way of estimating MI, given by Huber et al. (2008), where n is dimensions of $*$, and δ is a covariance matrix for variables. Figure A.17 shows the visualisation of the Gaussian covariance matrices, the ITR estimated under Gaussian assumptions, and the intensity of outputs from each layer of cascade networks on both synthetic and realistic data.

$$\begin{aligned} I(X;T) &= H(X) + H(T) - H(X,T) \\ H(*) &= \frac{1}{2}\log\left[(2\pi e)^n |\delta|\right] \end{aligned} \tag{A.1}$$

From both Figure A.17(c) and A.17(d), we notice that outputs from early layers have higher variance (see the diagonal lines) while later layers have higher covariance (i.e., outputs of some neurons in a layer are more similar). Reflected by the intensity plot of the outputs (Figures A.17(e) and A.17(f)), the classes can be easily distinguished from the later layers' outputs. The ratio estimated based on a Gaussian assumption shows an trend with gradually increased values. However, as variables are not exact Gaussian variables, the estimation has disparity with other estimations as shown in Figures A.18

and A.19. The ITR given by Gaussian estimation shows sharp increases at the layer earlier than the optimal layer which is suggested by an appropriate estimator and has the satisfactory performance.

A.6 Rate Distortion Theory and Information Bottleneck Theory

In this section (related to Section 3.3), we introduce more details of connections between IB and rate distortion theory. Equations A.2 and A.3 show the formulated relation between entropy and MI, corresponding to Figure 3.1. Loosely speaking, IB theory replaces external constraints in rate distortion theory with a precise goal related to the targets.

$$H(X|T) = -\sum_{\boldsymbol{x}\in X,\boldsymbol{t}\in T} p(\boldsymbol{x},\boldsymbol{t}) \log \frac{p(\boldsymbol{x},\boldsymbol{t})}{p(\boldsymbol{t})} = -\sum_{\boldsymbol{x}\in X} p(\boldsymbol{x}) \sum_{\boldsymbol{t}\in T} p(\boldsymbol{t}|\boldsymbol{x}) \log p(\boldsymbol{x}|\boldsymbol{t}) \tag{A.2}$$

$$\begin{aligned} I(X;T) &= H(X) - H(X|T) = H(T) - H(T|X) \\ &= H(X) + H(T) - H(X,T) = H(X,T) - H(X|T) - H(T|X) \\ &= \sum_{\boldsymbol{x}\in X} \sum_{\boldsymbol{t}\in T} p(\boldsymbol{x},\boldsymbol{t}) \log \frac{p(\boldsymbol{x},\boldsymbol{t})}{p(\boldsymbol{x})p(\boldsymbol{t})} \\ &= \sum_{\boldsymbol{x}\in X} \sum_{\boldsymbol{t}\in T} p(\boldsymbol{x},\boldsymbol{t}) \log \frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} \end{aligned} \tag{A.3}$$

A.6.1 Rate Distortion Theory

According to the notations in Section 3.1, $I(X;T)$ is used to quantify the compactness of a new representation of an input, where lower values mean more compact transformation. There are two extreme cases, a) $I(X;T) = 0$ when T has a just single value; and b) $I(X;T) = H(X)$ when T is a simply duplication of X. However, compression is not enough for evaluating the quality of representations. Continually reducing details in X can be a way of compressing information getting a compressed representations which will be useless representations in realistic applications. Therefore, additional constraints are necessary.

In rate distortion theory, a distortion measure can be an accomplished way of providing constraints. The distortion quantifies the "distance" between the random variable X and its representation T. Specifically, $d : \mathcal{X} \times \mathcal{T} \to \mathcal{R}^+$ is a defined function used to complete the above quantification with a hypothesis that a smaller distortion implies a better representation. Given such a function shown in Equation A.4, the representation of X induced by mapping $p(\boldsymbol{t}|\boldsymbol{x})$ has an expected distortion. The trade-off between

representations' compactness and a corresponding expected distortion is the elemental adjustment in rate distortion theory (Slonim, 2002). Calligraphic notations denote spaces to which the value of variables belong.

$$\langle d(\boldsymbol{x},\boldsymbol{t})\rangle_{p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x})} = \sum_{\boldsymbol{x}\in X}\sum_{\boldsymbol{t}\in T} p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x})d(\boldsymbol{x},\boldsymbol{t}) \tag{A.4}$$

$$R(D) \equiv \min_{\{p(\boldsymbol{t}|\boldsymbol{x}):\langle d(\boldsymbol{x},\boldsymbol{t})\rangle\leq D\}} I(X;T) \tag{A.5}$$

The rate distortion function, denoted by $R(D)$ (shown in Equation A.5), consists of a given $p(\boldsymbol{x})$ and a distortion measurement $d(\boldsymbol{x},\boldsymbol{t})$. The denotation $R(D)$ is designated as the infimum of all rates, R, under a given constraint on an averaged distortion D. Namely, $R(D)$ is the minimal achievable compression-information over all normalised conditional distributions, $p(\boldsymbol{t}|\boldsymbol{x})$, meeting the distortion constraints.

On a rate distortion plane shown in Figure A.20, $R(D)$ characterises a monotonic trade-off that a higher D value (x axis of the plane) implies a smaller $I(X;T)$ (y axis of the plane). In other words, more relaxed distortion constraints signify stronger information compression. $\{D, I\}$ represents a distortion-compression pair. In the rate distortion plane, if this pair locates above the curve shown in Figure A.20, then there is a compressed representation T with the level of compression quantified by $I(X;T) = I$, accompanying with an expected distortion being upper bounded by D.

$$\mathcal{F}(p(\boldsymbol{t}|\boldsymbol{x})) = I(X;T) + \beta\langle d(\boldsymbol{x},\boldsymbol{t})\rangle_{p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x})} \tag{A.6}$$

A minimisation problem of a convex function needs to be solved for finding the rate distortion function. As shown in Equation A.6, the convex functional $\mathcal{F}(p(\boldsymbol{t}|\boldsymbol{x})$ is minimised over the convex set of all the normalised conditional distributions $p(\boldsymbol{t}|\boldsymbol{x})$. This problem can be solved by introducing a Lagrange multiplier, β, and then minimising the functional $\mathcal{F}(p(\boldsymbol{t}|\boldsymbol{x})$ with the constraint $\sum_{\boldsymbol{t}\in T} p(\boldsymbol{t}|\boldsymbol{x}) = 1$. This variational problem can be solved by making the derivative of $\tilde{\mathcal{F}}(p(\boldsymbol{t}|\boldsymbol{x})$ with respect to $p(\boldsymbol{t}|\boldsymbol{x})$ equal to zero, as shown in Equation A.7. Where the last term is the normalisation constraints and $\lambda(\boldsymbol{x})$ are the normalisation Lagrange multipliers for each $\boldsymbol{x}$. Moreover, the Lagrange multiplier β, bounded by the value of the expected distortion, D, is positive and satisfies Equation A.8.

For finding a rate distortion function $R(D)$ which can minimise the above functional formulation, the Blahut-Arimoto algorithm (Arimoto, 1972; Blahut, 1972) is applied. This algorithm is used to find the minimum distance between two convex sets, as shown in Algorithm 6. Csiszár (1984) has shown that the minimum problem will converge when

the two sets are convex.

$$
\begin{aligned}
\tilde{\mathcal{F}}(p(\boldsymbol{t}|\boldsymbol{x})) &= I(X;T) + \beta\langle d(\boldsymbol{x},\boldsymbol{t})\rangle_{p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x})} + \sum_{\boldsymbol{x}\in X}\lambda(\boldsymbol{x})\sum_{\boldsymbol{t}\in T}p(\boldsymbol{t}|\boldsymbol{x}) \\
\frac{\partial\tilde{\mathcal{F}}}{\partial p(\boldsymbol{t}|\boldsymbol{x})} &= \frac{\partial I(X;T)}{\partial p(\boldsymbol{t}|\boldsymbol{x})} + \beta\frac{\partial\left(\sum_{\boldsymbol{x}\in X}\sum_{\boldsymbol{t}\in T}p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x})d(\boldsymbol{x},\boldsymbol{t})\right)}{\partial p(\boldsymbol{t}|\boldsymbol{x})} + \frac{\partial\left(\sum_{\boldsymbol{x}\in X}\lambda(\boldsymbol{x})\sum_{\boldsymbol{t}\in T}p(\boldsymbol{t}|\boldsymbol{x})\right)}{\partial p(\boldsymbol{t}|\boldsymbol{x})} \\
&= \frac{\partial\left(\sum_{\boldsymbol{x}\in X}\sum_{\boldsymbol{t}\in T}p(\boldsymbol{x},\boldsymbol{t})\log\frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})}\right)}{\partial p(\boldsymbol{t}|\boldsymbol{x})} + \beta\frac{\partial\left(\sum_{\boldsymbol{x}\in X}\sum_{\boldsymbol{t}\in T}p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x})d(\boldsymbol{x},\boldsymbol{t})\right)}{\partial p(\boldsymbol{t}|\boldsymbol{x})} + \lambda(\boldsymbol{x}) \\
&= p(\boldsymbol{x})\log\frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} + \frac{\partial\left(\sum_{\boldsymbol{x}\in X}\sum_{\boldsymbol{t}\in T}\log p(\boldsymbol{t}|\boldsymbol{x}) - \sum_{\boldsymbol{x}\in X}\sum_{\boldsymbol{t}\in T}\log p(\boldsymbol{t})\right)}{\partial p(\boldsymbol{t}|\boldsymbol{x})}p(\boldsymbol{t}|\boldsymbol{x})p(\boldsymbol{x}) + \beta[\dots] + \lambda(\boldsymbol{x}) \\
&= p(\boldsymbol{x})\log\frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} + \left[\frac{1}{p(\boldsymbol{t}|\boldsymbol{x})} - \frac{\partial\left(\sum_{\boldsymbol{x}\in X}\sum_{\boldsymbol{t}\in T}\log p(\boldsymbol{t}|\boldsymbol{x})p(\boldsymbol{x})\right)}{\partial p(\boldsymbol{t}|\boldsymbol{x})}\right]p(\boldsymbol{t}|\boldsymbol{x})p(\boldsymbol{x}) + \beta[\dots] + \lambda(\boldsymbol{x}) \\
&= p(\boldsymbol{x})\log\frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} + \left[\frac{1}{p(\boldsymbol{t}|\boldsymbol{x})} - \frac{1}{p(\boldsymbol{t})}p(\boldsymbol{x})\right]p(\boldsymbol{t}|\boldsymbol{x})p(\boldsymbol{x}) + \beta[\dots] + \lambda(\boldsymbol{x}) \\
&= p(\boldsymbol{x})\left[\log\frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} + 1 - \frac{\sum_{\boldsymbol{x}'}p(\boldsymbol{t}|\boldsymbol{x}')p(\boldsymbol{x}')}{p(\boldsymbol{t})}\right] + \beta\left[p(\boldsymbol{x})d(\boldsymbol{x},\boldsymbol{t})\right] + \lambda(\boldsymbol{x}) \\
&= p(\boldsymbol{x})\left[\log\frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} + 1 - \frac{\sum_{\boldsymbol{x}'}p(\boldsymbol{t}|\boldsymbol{x}')p(\boldsymbol{x}')}{p(\boldsymbol{t})} + \beta d(\boldsymbol{x},\boldsymbol{t}) + \frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})}\right] = 0 \\
\Rightarrow \log\frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} &= -1 + \frac{p(\boldsymbol{t})}{p(\boldsymbol{t})} - \beta d(\boldsymbol{x},\boldsymbol{t}) - \frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})} \\
\Rightarrow p(\boldsymbol{t}|\boldsymbol{x}) &= p(\boldsymbol{t})\exp\left[-\beta d(\boldsymbol{x},\boldsymbol{t}) - \frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})}\right] \\
&= \frac{p(\boldsymbol{t})\exp\left[-\beta d(\boldsymbol{x},\boldsymbol{t})\right]}{\exp\frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})}} \\
&= \frac{p(\boldsymbol{t})}{\mathcal{Z}(\boldsymbol{x},\beta)}\exp\left[-\beta d(\boldsymbol{x},\boldsymbol{t})\right]
\end{aligned}
\tag{A.7}
$$

Where $\log\mathcal{Z}(\boldsymbol{x},\beta) = \frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})}$.

$$\frac{\partial R}{\partial D} = -\beta \tag{A.8}$$

Analogous to two sets of distributions, the Kullback Leiber (KL) divergence is the measurement of how much two distributions differ from each other. Rewriting MI $I(X;T)$ as $D_{KL}[p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x})\|p(\boldsymbol{x})p(\boldsymbol{t})]$, this KL divergence will be an upper bound of the compression information term and equality holds if and only if the marginal distribution $p(\boldsymbol{t}) = \sum_{\boldsymbol{x}} p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x})$. And then the rate distortion function can be rewritten as a double minimisation as shown in Equation A.9. Henceforth, we can use alternating iterations of the marginal distribution and Equation A.7 to obtain the optimal rate distortion function as shown in Equation A.10. Firstly, we randomly initialise $p(\boldsymbol{t})$, and then use Equation A.7 to obtain $p(\boldsymbol{t}|\boldsymbol{x})$ which is for minimising the information term subject to a fixed distortion constraint. The new $p(\boldsymbol{t})$ can be further obtained at the basis of the marginal distribution. This alternative iteration can be defined into mathematical formulation as shown in Equation A.10. Algorithm 6 shows details of this iteration procedure.

Algorithm 6 Pseudocode of Blahut-Arimoto Algorithm (Slonim, 2002)

Inputs:
    Distribution $p(\boldsymbol{x})$.
    Lagrange multiplier β for trade-off, and convergence parameter ϵ.
    A set of representations given by T.
    Distortion measurement $d : \mathcal{X} \times \mathcal{T} \to \mathcal{R}^+$, $\forall \boldsymbol{x} \in X$, $\forall \boldsymbol{t} \in T$.
Outputs: Value of $R(D)$ where its slope equals $-\beta$.
Initialisation: Initialise $R^{(0)} \leftarrow \infty$ and randomly initialise $p(\boldsymbol{t})$.
while True **do**
    $p^{(i+1)}(\boldsymbol{t}|\boldsymbol{x}) \leftarrow \frac{p^{(i)}(\boldsymbol{t})}{\mathcal{Z}^{(i+1)}(\boldsymbol{x},\beta)} \exp\left(-\beta d(\boldsymbol{x},\boldsymbol{t})\right)$, $\forall \boldsymbol{t} \in T$, $\forall \boldsymbol{x} \in X$.
    $p^{(i+1)}(\boldsymbol{t}) \leftarrow \sum_{\boldsymbol{x}} p(\boldsymbol{x}) p^{(i+1)}(\boldsymbol{t}|\boldsymbol{x})$, $\forall \boldsymbol{t} \in T$.
    if $(R^{(i)}(D) - R^{(i+1)}(D)) \leq \epsilon$ **then**
        Break
    end if
end while

$$R(D) = \min_{\{p(\boldsymbol{t})\}} \min_{\{p(\boldsymbol{t}|\boldsymbol{x}):\langle d(\boldsymbol{x},\boldsymbol{t})\rangle \leq D\}} D_{KL}[p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x})\|p(\boldsymbol{x})p(\boldsymbol{t})] \tag{A.9}$$

$$\begin{cases} p_{i+1}(\boldsymbol{t}) = \sum_{\boldsymbol{t}} p(\boldsymbol{x}) p_i(\boldsymbol{t}|\boldsymbol{x}) \\ p_i(\boldsymbol{t}|\boldsymbol{x}) = \frac{p_i(\boldsymbol{t})}{\mathcal{Z}_i(\boldsymbol{x},\beta)} \exp\left[-\beta d(\boldsymbol{x},\boldsymbol{t})\right] \end{cases} \tag{A.10}$$

Where i is the iteration step.

The choice of a distortion measure in rate distortion theory will significantly affect the final results of rate distortion functions. In other words, $R(D)$ relies on an external defined measurement which usually is not directly connected to source properties. Motivated by this drawback, IB theory is proposed as an alternative approach which replaces the distortion measurement by a target variable Y with respect to the source variable X. The details of this theory are introduced in the next section.

A.6.2 Information Bottleneck Theory

Analogues to rate distortion theory as mentioned in section 3.3, the IB theory forces the model to define a precise goal while compressing the source X instead of giving constraints from external definition. Namely, the theory is looking for a compressed representation T which maintains MI about a relative variable Y as high as possible. Therefore, a lower bound constraint at the basis of the relevant information, given by $I(T;Y)$, replaces the upper bound constraint in the rate distortion theory. As shown in Equation A.11, we want to minimise $I(X;T)$ while preserving $I(T;Y)$ either as much as possible or above some minimal level.

$$\mathcal{L} = I(X;T) - \beta I(T;Y) \tag{A.11}$$

In this sense, T is a compressed representation of X, hence T only depends on X and cannot create any new information about Y. Namely, T can only provide the information already given by X. An equivalent formulation is shown as Equation A.12. The corresponding Markovian independence relation is shown in Equation A.13, which also implies Equation A.14 and A.15.

$$\begin{aligned} p(\boldsymbol{t}|\boldsymbol{x},\boldsymbol{y}) &= p(\boldsymbol{t}|\boldsymbol{x}) \\ \therefore p(\boldsymbol{x},\boldsymbol{y},\boldsymbol{t}) &= p(\boldsymbol{x},\boldsymbol{y})p(\boldsymbol{t}|\boldsymbol{x}) \end{aligned} \tag{A.12}$$

$$T \Leftrightarrow X \Leftrightarrow Y \tag{A.13}$$

$$\begin{aligned} p(\boldsymbol{y}) &= \sum_{\boldsymbol{t}} p(\boldsymbol{y}|\boldsymbol{t})p(\boldsymbol{t}) = \sum_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) = \sum_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x}) \sum_{\boldsymbol{t}} p(\boldsymbol{x}|\boldsymbol{t})p(\boldsymbol{t}) \\ \Rightarrow p(\boldsymbol{y}|\boldsymbol{t}) &= \sum_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{t}) \\ &= \sum_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x}) \frac{p(\boldsymbol{t}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{t})} \\ \Rightarrow p(\boldsymbol{y}|\boldsymbol{t}) &= \frac{1}{p(\boldsymbol{t})} \sum_{\boldsymbol{x}} p(\boldsymbol{t}|\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) \end{aligned} \tag{A.14}$$

$$\begin{aligned} p(\boldsymbol{t}) = \sum_{\boldsymbol{x}} p(\boldsymbol{t}|\boldsymbol{x})p(\boldsymbol{x}) &\Rightarrow \frac{\partial p(\boldsymbol{t})}{\partial p(\boldsymbol{t}|\boldsymbol{x})} = p(\boldsymbol{x}) \\ \Rightarrow \sum_{\boldsymbol{y}} p(\boldsymbol{t}|\boldsymbol{y})p(\boldsymbol{y}) = \sum_{\boldsymbol{x}} p(\boldsymbol{t}|\boldsymbol{x}) \sum_{\boldsymbol{y}} p(\boldsymbol{x}|\boldsymbol{y})p(\boldsymbol{y}) & \\ \Rightarrow p(\boldsymbol{t}|\boldsymbol{y}) = \sum_{\boldsymbol{x}} p(\boldsymbol{t}|\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{y}) & \\ \frac{\partial p(\boldsymbol{t}|\boldsymbol{y})}{\partial p(\boldsymbol{t}|\boldsymbol{x})} = p(\boldsymbol{x}|\boldsymbol{y}) & \end{aligned} \tag{A.15}$$

Given a joint distribution $p(\boldsymbol{x},\boldsymbol{y})$, the problem of optimising the trade-off between compression and relevance can be defined as a function shown in Equation A.16, where

the minimisation is over the normalised conditional distributions, $p(\boldsymbol{t}|\boldsymbol{x})$, for which the constraint is satisfied. Namely, $\hat{R}(\hat{D})$ is the minimal achievable compression information while the relevant information is above $\hat{D}$. Equation A.16 is equivalent to the rate distortion function in the rate distortion theory.

$$\hat{R}(\hat{D}) = \min_{\{p(\boldsymbol{t}|\boldsymbol{x}):I(T;Y)\geq\hat{D}\}} I(X;T) \tag{A.16}$$

Reflected on a relevance-compression plane where $I(X;T)$ corresponds to the horizontal axis and $I(T;Y)$ is the vertical axis as shown in Figure A.21, $\hat{R}(\hat{D})$ corresponds to a non-decreasing concave function of $\hat{D}$. Furthermore, the slope of this curve is determined by $\frac{\partial\hat{R}}{\partial\hat{D}} = \beta$ which gradually decreases with the preference shift from compression to preservation of relevant information. In other words, the increase of $I(X;T)$ can be analogous to the growing cardinality of the compressed representation $|\mathcal{T}|$. Alternatively, at the beginning of the curve (i.e minimal $I(X;T)$), we look forward to the most compact representation with $|\mathcal{T}| = 1$. The constraint over $I(T;Y)$ becomes more demanding with the gradually increasing β. After some critical value of β, the single value of $\mathcal{T}$ will be bifurcated for fulfilling the relevant information constraint, which can be seen as a phase transition of the system. The successive increment of β will result in another bifurcation of $\mathcal{T}$. Hence, if the cardinality of the representation T is fixed (e.g. $|\mathcal{T}| = 2$ as shown in Figure A.21), a family of the sub-optimal curves, falling in the relevance compression region, can be obtained.

A.6.3 The Solution of the Information Bottleneck Principle

For finding the optimal solution of Equation A.11, we need functional $\tilde{\mathcal{L}}$ as shown in Equation A.17, where the last term is the normalisation constraints and $\lambda(\boldsymbol{x})$ are the normalisation Lagrange multipliers for each $\boldsymbol{x}$. Taking the derivative of the functional, we obtain the expression of $p(\boldsymbol{t}|\boldsymbol{x})$ at the basis of the divergence $D_{KL}[p(\boldsymbol{y}|\boldsymbol{x})\|p(\boldsymbol{y}|\boldsymbol{t})]$. From the expression we can see the performance of T, as a representation of X, is improved when a distribution $p(\boldsymbol{y}|\boldsymbol{x})$ gets closer to a distribution $p(\boldsymbol{y}|\boldsymbol{t})$. Consequently, the KL divergence decreases so that $p(\boldsymbol{t}|\boldsymbol{x})$ increases. On the contrary, if T is not a good representation of X, the KL divergence is large and $p(\boldsymbol{t}|\boldsymbol{x})$ decreases.

Additionally, the expression implies similar characteristics to the relevance-compression plane shown in Figure A.21 regarding the value of β. A small value of β indicates high diffusion as it has an inhibiting effect on the differences between the KL divergence and different value of T. While the limit $\beta \to 0$, the maximal diffusion implies $p(\boldsymbol{t}|\boldsymbol{x})$ is not depending on X, which corresponds to maximal information compression. While the limit $\beta \to \infty$, the situation is the opposite, where all the emphases are on the preservation of the relevant information.

$$
\begin{aligned}
\tilde{\mathcal{L}} &= I(X;T) - \beta I(T;Y) - \sum_{\boldsymbol{x}} \lambda(\boldsymbol{x}) p(\boldsymbol{t}|\boldsymbol{x}) \\
&= \sum_{\boldsymbol{x}\in X}\sum_{\boldsymbol{t}\in T} p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x}) \log \frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} - \beta \sum_{\boldsymbol{t},\boldsymbol{y}} p(\boldsymbol{t},\boldsymbol{y}) \log \frac{p(\boldsymbol{t}|\boldsymbol{y})}{p(\boldsymbol{t})} - \sum_{\boldsymbol{x},\boldsymbol{t}} \lambda(\boldsymbol{x}) p(\boldsymbol{t}|\boldsymbol{x}) \\
\frac{\partial \tilde{\mathcal{L}}}{\partial p(\boldsymbol{t}|\boldsymbol{x})} &= p(\boldsymbol{x}) \log \frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} + p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x}) \frac{1}{p(\boldsymbol{t}|\boldsymbol{x})} - p(\boldsymbol{x})p(\boldsymbol{t}|\boldsymbol{x}) \frac{1}{p(\boldsymbol{t})} p(\boldsymbol{x}) - \beta[\dots] - \lambda(\boldsymbol{x}) \\
&= p(\boldsymbol{x})[\log p(\boldsymbol{t}|\boldsymbol{x}) - \log p(\boldsymbol{t})] - \beta \frac{\partial \left[\sum_{\boldsymbol{t},\boldsymbol{y}} p(\boldsymbol{t}|\boldsymbol{y}) p(\boldsymbol{y}) \log \frac{p(\boldsymbol{t}|\boldsymbol{y})}{p(\boldsymbol{t})} \right]}{\partial p(\boldsymbol{t}|\boldsymbol{x})} - \lambda(\boldsymbol{x}) \\
&= p(\boldsymbol{x})[\dots] - \beta \left\{ \sum_{\boldsymbol{y}} p(\boldsymbol{y}) \log \frac{p(\boldsymbol{t}|\boldsymbol{y})}{p(\boldsymbol{t})} p(\boldsymbol{x}|\boldsymbol{y}) + \sum_{\boldsymbol{y}} p(\boldsymbol{t}|\boldsymbol{y}) p(\boldsymbol{y}) \left[\frac{p(\boldsymbol{x}|\boldsymbol{y})}{p(\boldsymbol{t}|\boldsymbol{y})} - \frac{p(\boldsymbol{x})}{p(\boldsymbol{t})} \right] \right\} - \lambda(\boldsymbol{x}) \\
&= p(\boldsymbol{x})[\dots] - \beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}) p(\boldsymbol{x}|\boldsymbol{y}) [\log p(\boldsymbol{t}|\boldsymbol{y}) - \log p(\boldsymbol{t}) + 1] + \beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}) p(\boldsymbol{t}|\boldsymbol{y}) \frac{p(\boldsymbol{x})}{p(\boldsymbol{t})} - \lambda(\boldsymbol{x}) \\
&= p(\boldsymbol{x}) \left[\log \frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} \right] - \beta \sum_{\boldsymbol{y}} p(\boldsymbol{x}) p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{t}|\boldsymbol{y})}{p(\boldsymbol{t})} - \lambda(\boldsymbol{x}) \\
&= p(\boldsymbol{x}) \left[\log \frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} - \beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{y}|\boldsymbol{t}) p(\boldsymbol{t})}{p(\boldsymbol{t}) p(\boldsymbol{y})} - \frac{-\lambda(\boldsymbol{x})}{p(\boldsymbol{x})} \right] \\
&= p(\boldsymbol{x}) \left[\log \frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} - \beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{y}|\boldsymbol{t})}{p(\boldsymbol{y})} - \frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})} \right] \\
&= p(\boldsymbol{x}) \left\{ \log \frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} - \beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \left[\log \frac{p(\boldsymbol{y}|\boldsymbol{t})}{p(\boldsymbol{y})} + \log p(\boldsymbol{y}|\boldsymbol{x}) - \log p(\boldsymbol{y}|\boldsymbol{x}) \right] - \frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})} \right\} \\
&= p(\boldsymbol{x}) \left\{ \log \frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} - \beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \left[\log \frac{p(\boldsymbol{y}|\boldsymbol{t})}{p(\boldsymbol{y}|\boldsymbol{x})} + \log \frac{p(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y})} \right] - \frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})} \right\} \\
&= p(\boldsymbol{x}) \left\{ \log \frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} + \beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y}|\boldsymbol{t})} - \beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y})} - \frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})} \right\} = 0 \\
\Rightarrow \log \frac{p(\boldsymbol{t}|\boldsymbol{x})}{p(\boldsymbol{t})} &= -\beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y}|\boldsymbol{t})} + \beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y})} + \frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})} \\
p(\boldsymbol{t}|\boldsymbol{x}) &= \exp \left[-\beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y}|\boldsymbol{t})} + \beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y})} + \frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})} \right] p(\boldsymbol{t}) \\
&= \frac{p(\boldsymbol{t})}{\exp \left[-\beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y})} - \frac{\lambda(\boldsymbol{x})}{p(\boldsymbol{x})} \right]} \exp \left[-\beta \sum_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x}) \log \frac{p(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y}|\boldsymbol{t})} \right] \\
&= \frac{p(\boldsymbol{t})}{\mathcal{Z}(\boldsymbol{x};\beta)} \exp \left[-\beta D_{KL}[p(\boldsymbol{y}|\boldsymbol{x}) \| p(\boldsymbol{y}|\boldsymbol{t})] \right]
\end{aligned}
\tag{A.17}
$$

Where $p(\boldsymbol{y}|\boldsymbol{t}) = \frac{1}{p(\boldsymbol{t})} \sum_{\boldsymbol{x}} p(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{y}|\boldsymbol{t})$.

For getting the exact or approximated solution of this variational principle, an iterative optimisation algorithm as shown in Equation A.18 is needed. As an extension of Blahut-Arimoto algorithm, Equation A.19 (Shamir et al., 2010; Tishby and Zaslavsky, 2015; Tishby et al., 1999) shows updating the expression of the IB iterative algorithm at the i_{th} iteration. However, there are distinct differences between those two algorithms. First, instead of $-\beta$, this IB algorithm is going to converge to a point of which the slope is β^{-1}; Secondly, the IB curve is not defined with respect to a predefined distortion measure so that the minimisation is over three distributions where $p(\boldsymbol{y}|\boldsymbol{t})$ is an additional one, while the Blahut-Arimoto algorithm is based on a fixed distortion hence the minimisation is only over $p(\boldsymbol{t}|\boldsymbol{x})$ and $p(\boldsymbol{t})$. Because of the non-fixed representation, the solution of this algorithm is not guaranteed as an unique one, therefore, we generally can only except it to converge to a locally optimal solution. The details of this IB iterative algorithm are shown in Algorithm 7, where $\mathcal{JS}$ denotes the Jensen-Shannon divergence (El-Yaniv et al., 1997).

$$\begin{aligned} \min_{p(\boldsymbol{y}|\boldsymbol{t})} \min_{p(\boldsymbol{t})} \min_{p(\boldsymbol{t}|\boldsymbol{x})} \tilde{\mathcal{L}}[p(\boldsymbol{t}|\boldsymbol{x}); p(\boldsymbol{t}); p(\boldsymbol{y}|\boldsymbol{t})] &= -\langle \log \mathcal{Z}(\boldsymbol{x}, \beta) \rangle_{p(\boldsymbol{x})} \\ &= I(\boldsymbol{x}; \boldsymbol{t}) + \beta \langle D_{KL}[p(\boldsymbol{y}|\boldsymbol{x}) \| p(\boldsymbol{y}|\boldsymbol{t})] \rangle_{p(\boldsymbol{x},\boldsymbol{t})} \end{aligned} \tag{A.18}$$

$$\begin{cases} p_i(\boldsymbol{t}|\boldsymbol{x}) = \frac{p_i(\boldsymbol{t})}{\mathcal{Z}_i(\boldsymbol{x},\beta)} \exp\left[-\beta d(\boldsymbol{x}, \boldsymbol{t})\right] \\ p_{i+1}(\boldsymbol{t}) = \sum_{\boldsymbol{t}} p(\boldsymbol{x}) p_i(\boldsymbol{t}|\boldsymbol{x}) \\ p_{i+1}(\boldsymbol{y}|\boldsymbol{t}) = \sum_{\boldsymbol{y}} p_i(\boldsymbol{y}|\boldsymbol{x}) p_i(\boldsymbol{x}|\boldsymbol{t}) \end{cases} \tag{A.19}$$

Where $d(\boldsymbol{x}, \boldsymbol{t}) = D_{KL}[p(\boldsymbol{y}|\boldsymbol{x}) \| p(\boldsymbol{y}|\boldsymbol{t})]$, and i is the iteration step.

Algorithm 7 Pseudocode of IB Iterative Algorithm (adopted from (Slonim, 2002))

Inputs:
    Distribution $p(\boldsymbol{x}, \boldsymbol{y})$.
    Lagrange multiplier β for trade-off, and convergence parameter ϵ.
    Carnality parameter M
Outputs: A partition T of X into M clusters.
Initialisation: Randomly initialise $p(\boldsymbol{t}|\boldsymbol{x})$ and find $p(\boldsymbol{t})$ and $p(\boldsymbol{y}|\boldsymbol{t})$ through Equation A.19.
while True **do**
    $p^{(i+1)}(\boldsymbol{t}|\boldsymbol{x}) \leftarrow \frac{p^{(i)}(\boldsymbol{t})}{\mathcal{Z}^{(i+1)}(\boldsymbol{x},\beta)} \exp\left(-\beta d(\boldsymbol{x}, \boldsymbol{t})\right), \forall \boldsymbol{t} \in T, \ \forall \boldsymbol{x} \in X.$
    $p^{(i+1)}(\boldsymbol{t}) \leftarrow \sum_{\boldsymbol{x}} p(\boldsymbol{x}) p^{(i+1)}(\boldsymbol{t}|\boldsymbol{x}), \forall \boldsymbol{t} \in T.$
    $p^{(i+1)}(\boldsymbol{y}|\boldsymbol{t}) \leftarrow \frac{1}{p^{(i+1)}(\boldsymbol{t})} \sum_{\boldsymbol{x}} p^{(i+1)}(\boldsymbol{t}|\boldsymbol{x}) p(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}), \forall \boldsymbol{t} \in T, \ \forall \boldsymbol{y} \in Y.$
    if $\forall \boldsymbol{x} \in X, \mathcal{JS}_{\frac{1}{2},\frac{1}{2}}[p^{(i+1)}(\boldsymbol{t}|\boldsymbol{x}), p^{(i)}(\boldsymbol{t}|\boldsymbol{x})] \leq \epsilon$ **then**
        Break
    end if
end while

A.7 Gaussian Information Bottleneck and Canonical Correction Analysis

As mentioned in Section 3.5.4, Chechik et al. (2005) derive a relationship between IB theory and canonical correction analysis when the probability densities are Gaussian. Empirical results under this assumption are shown in Section 3.5.4. In this section, we explain the details of this connection.

A.7.1 Canonical Correction Analysis

In canonical correlation analysis, we want to find a set of parameters $\boldsymbol{a}$ and $\boldsymbol{b}$ that maximally correlate X and Y in the subspace. We define vectors $\boldsymbol{a} \in \mathbb{R}^n$ and $\boldsymbol{b} \in \mathbb{R}^n$ such that $\tilde{X} = \boldsymbol{a}^\top X$ and $\tilde{Y} = \boldsymbol{b}^\top Y$ have maximal correlation $\rho = \text{corr}\left(\tilde{X}, \tilde{Y}\right)$ in new subspaces. More precisely, Equation A.20 gives a formulated definition about cross-covariance and correlation, and Equation A.21 shows the correlation ρ we want to maximise. By substituting in Equation A.22, we can obtain Equation A.23.

$$\begin{aligned} \Sigma_{\boldsymbol{x}} &= \text{Cov}(X, X) \\ \Sigma_{\boldsymbol{y}} &= \text{Cov}(Y, Y) \\ \text{Cov}[X, Y] &= E[XY] - E[X]E[Y] = \Sigma_{\boldsymbol{xy}} \\ \text{Corr}[X, Y] &= \frac{\Sigma_{\boldsymbol{xy}}}{\Sigma_{\boldsymbol{x}}\Sigma_{\boldsymbol{y}}} \end{aligned} \tag{A.20}$$

$$\rho = \frac{\boldsymbol{a}^\top \Sigma_{\boldsymbol{xy}} \boldsymbol{b}}{\sqrt{\boldsymbol{a}^\top \Sigma_{\boldsymbol{x}} \boldsymbol{a}} \sqrt{\boldsymbol{b}^\top \Sigma_{\boldsymbol{y}} \boldsymbol{b}}} \tag{A.21}$$

$$\begin{aligned} \boldsymbol{c} &= \Sigma_{\boldsymbol{x}}^{\frac{1}{2}} \boldsymbol{a} \\ \boldsymbol{d} &= \Sigma_{\boldsymbol{y}}^{\frac{1}{2}} \boldsymbol{b} \end{aligned} \tag{A.22}$$

$$\rho = \frac{\boldsymbol{c}^\top \Sigma_{\boldsymbol{x}}^{-\frac{1}{2}} \Sigma_{\boldsymbol{xy}} \Sigma_{\boldsymbol{y}}^{-\frac{1}{2}} \boldsymbol{d}}{\sqrt{\boldsymbol{c}^\top \boldsymbol{c}} \sqrt{\boldsymbol{d}^\top \boldsymbol{d}}} \tag{A.23}$$

According to Cauchy-Schwarz inequality (Steele, 2004) shown in Equation A.24, we can achieve Equation A.25 by letting $\boldsymbol{u} = \boldsymbol{c}^\top \Sigma_{\boldsymbol{x}}^{-\frac{1}{2}} \Sigma_{\boldsymbol{xy}} \Sigma_{\boldsymbol{y}}^{-\frac{1}{2}}$ and $\boldsymbol{v} = \boldsymbol{d}$. After reformulating, we obtain the upper bound of the correlation as shown in Equation A.26.

$$|\langle \boldsymbol{u}, \boldsymbol{v} \rangle| \leq \|\boldsymbol{u}\| \|\boldsymbol{v}\| \tag{A.24}$$

$$\boldsymbol{c}^\top \Sigma_{\boldsymbol{x}}^{-\frac{1}{2}} \Sigma_{\boldsymbol{xy}} \Sigma_{\boldsymbol{y}}^{-\frac{1}{2}} \boldsymbol{d} \leq \left(\boldsymbol{c}^\top \Sigma_{\boldsymbol{x}}^{-\frac{1}{2}} \Sigma_{\boldsymbol{xy}} \Sigma_{\boldsymbol{y}}^{-1} \Sigma_{yx} \Sigma_{\boldsymbol{x}}^{-\frac{1}{2}} \boldsymbol{c} \right)^{\frac{1}{2}} (\boldsymbol{d}^\top \boldsymbol{d})^{\frac{1}{2}} \tag{A.25}$$

$$\rho \leq \frac{\left(\boldsymbol{c}^\top \Sigma_{\boldsymbol{x}}^{-\frac{1}{2}} \Sigma_{\boldsymbol{xy}} \Sigma_{\boldsymbol{y}}^{-1} \Sigma_{\boldsymbol{yx}} \Sigma_{\boldsymbol{x}}^{-\frac{1}{2}} \boldsymbol{c}\right)^{\frac{1}{2}}}{\left(\boldsymbol{c}^\top \boldsymbol{c}\right)^{\frac{1}{2}}} \tag{A.26}$$

Connected to Rayleigh quotient (Wu, 2005) shown in Equation A.27, we let $M = \Sigma_{\boldsymbol{x}}^{-\frac{1}{2}} \Sigma_{\boldsymbol{xy}} \Sigma_{\boldsymbol{y}}^{-1} \Sigma_{\boldsymbol{yx}} \Sigma_{\boldsymbol{x}}^{-\frac{1}{2}}$ and $\boldsymbol{r} = \boldsymbol{c}$. Then, to find $\rho_{max} = R(M, \boldsymbol{r})_{max} = \lambda_{max}$. To solve this problem, we can perform eigen-decomposition for M, and obtain the corresponding eigenvalues ρ in a descending order. In that way, we transform the canonical correlation analysis to an eigen-decomposition problem.

$$R(M, \boldsymbol{r}) = \frac{\boldsymbol{r}^\top M \boldsymbol{r}}{\boldsymbol{r}^\top \boldsymbol{r}} \tag{A.27}$$

Where M is a symmetry matrix, and $R(M, \boldsymbol{r})$ reaches maximum when $\boldsymbol{r} = v_{max}$ which is the eigenvector corresponding to maximum eigenvalue of M. In this case, we have $R(M, v_{max}) = \lambda_{max}$.

A.7.2 Gaussian Information Bottleneck

Analogous to canonical correction analysis, the Gaussian IB theory can also be connected to the eigen-decomposition problem. In this section, we introduce the Gaussian IB theory and then show the relation to eigen-decomposition.

As shown in Equation A.28, the entropy of a Gaussian variable can be expressed based on the covariance of the variable, where X is a d dimensional Gaussian variable, and $|\Sigma_{\boldsymbol{x}}|$ is the determinant of $\Sigma_{\boldsymbol{x}}$. Under the assumption that X and Y are joint multivariate Gaussian variables (so that the IB is analytically tractable) with dimensions $d_{\boldsymbol{x}}$ and $d_{\boldsymbol{y}}$. $\Sigma_{\boldsymbol{x}}$ and $\Sigma_{\boldsymbol{y}}$ are the covariance matrices of X and Y, respectively, and $\Sigma_{\boldsymbol{xy}}$ is their cross-covariance matrix. T is a compressed representation of X obtained via a linear transformation, where the dimension depends on β. Hence, we have $T = AX + \xi$ (another Gaussian), where $\xi \sim N(\mathbf{0}, \Sigma_\xi)$, and $T \sim N(\mathbf{0}, \Sigma_{\boldsymbol{t}})$. Thereafter, we can obtain $\Sigma_{\boldsymbol{t}} = A\Sigma_{\boldsymbol{t}} A^\top + \Sigma_\xi$. By the definition of T, we have $\Sigma_{\boldsymbol{tx}} = A\Sigma_{\boldsymbol{x}}$ [3], and $\Sigma_{\boldsymbol{ty}} = A\Sigma_{\boldsymbol{xy}}$ [4]. According to Schur complement formula (Zhang, 2006), we know $\Sigma_{\boldsymbol{t|y}} = \Sigma_{\boldsymbol{t}} - \Sigma_{\boldsymbol{ty}} \Sigma_{\boldsymbol{y}}^{-1} \Sigma_{\boldsymbol{yt}} = A\Sigma_{\boldsymbol{x|y}} A^\top + \Sigma_{\boldsymbol{\xi}}$ [5], and $\Sigma_{\boldsymbol{t|x}} = \Sigma_{\boldsymbol{t}} - \Sigma_{\boldsymbol{tx}} \Sigma_{\boldsymbol{x}}^{-1} \Sigma_{\boldsymbol{xt}} = A\Sigma_{\boldsymbol{x}} A^\top + \Sigma_{\boldsymbol{\xi}} - A\Sigma_{\boldsymbol{x}} \Sigma_{\boldsymbol{x}}^{-1} \Sigma_{\boldsymbol{x}} A^\top = \Sigma_{\boldsymbol{\xi}}$. Using this, we can

[3] $\mathrm{Cov}(T, X) = E(TX) - E(T)E(X) = E[(AX + \xi)X] - 0 = E[AX^2 + \xi X] = AE(X^2) + E(\xi X) = AE(X^2) = A\Sigma_{\boldsymbol{x}}$.

[4] $\mathrm{Cov}(T, Y) = E(TY) - E(T)E(Y) = E[(AX + \xi)Y] - 0 = E[AXY + \xi X] = AE(XY) + E(\xi Y) = AE(XY) = A\Sigma_{\boldsymbol{xy}}$.

[5] $\Sigma_{\boldsymbol{t|y}} = \Sigma_{\boldsymbol{t}} - \Sigma_{\boldsymbol{ty}} \Sigma_{\boldsymbol{y}}^{-1} \Sigma_{\boldsymbol{yt}} = A\Sigma_{\boldsymbol{x}} A^\top + \Sigma_{\boldsymbol{\xi}} - A\Sigma_{\boldsymbol{xy}} \Sigma_{\boldsymbol{y}}^{-1} \Sigma_{\boldsymbol{yx}} A^\top = A[\Sigma_{\boldsymbol{x}} - \Sigma_{\boldsymbol{xy}} \Sigma_{\boldsymbol{y}}^{-1} \Sigma_{\boldsymbol{yx}}] A^\top + \Sigma_{\boldsymbol{\xi}} = A\Sigma_{\boldsymbol{x|y}} A^\top + \Sigma_{\boldsymbol{\xi}}$.

obtain Equation A.29.

$$\begin{aligned} H(X) &\equiv -\int_{\boldsymbol{x}\in X} f(\boldsymbol{x})\log f(\boldsymbol{x})d\boldsymbol{x} \\ H(X) &= \frac{1}{2}\log\left((2\pi\mathrm{e})^d|\Sigma_{\boldsymbol{x}}|\right) \end{aligned} \tag{A.28}$$

$$\begin{aligned} \min_{A,\Sigma_\xi}\mathcal{L} &= I(X;T) - \beta I(T;Y) \\ &= H(T) - H(T|X) - \beta H(T) + \beta H(T|Y) \\ &= \frac{1}{2}\left[\log\left((2\pi\mathrm{e})^{d_{\boldsymbol{t}}}|\Sigma_{\boldsymbol{t}}|\right) - \log\left((2\pi\mathrm{e})^{d_{\boldsymbol{t}|\boldsymbol{x}}}|\Sigma_{\boldsymbol{t}|\boldsymbol{x}}|\right)\right] \\ &\qquad - \frac{1}{2}\beta\left[\log\left((2\pi\mathrm{e})^{d_{\boldsymbol{t}}}|\Sigma_{\boldsymbol{t}}|\right) - \log\left((2\pi\mathrm{e})^{d_{\boldsymbol{t}|\boldsymbol{y}}}|\Sigma_{\boldsymbol{t}|\boldsymbol{y}}|\right)\right] \\ &= \log\left(|\Sigma_{\boldsymbol{t}}|\right) - \log\left(|\Sigma_{\boldsymbol{t}|\boldsymbol{x}}|\right) - \beta\log\left(|\Sigma_{\boldsymbol{t}}|\right) + \beta\log\left(|\Sigma_{\boldsymbol{t}|\boldsymbol{y}}|\right) \\ &= (1-\beta)\log\left(|A\Sigma_{\boldsymbol{x}}A^\top + \Sigma_\xi|\right) - \log\left(|\Sigma_\xi|\right) + \beta\log\left(|A\Sigma_{\boldsymbol{x}|\boldsymbol{y}}A^\top + \Sigma_\xi|\right) \end{aligned} \tag{A.29}$$

By setting $\tilde{A} = \sqrt{D^{-1}}VA$, where[6] $\Sigma_\xi = VDV^\top = VDV^{-1}$, we can rewrite Equation A.29 to Equation A.30 for simplification $\mathcal{L}(A,\Sigma_\xi) = \mathcal{L}(\tilde{A},I)$. Here, $C = |\sqrt{D^{-1}}V| = |\sqrt{D^{-1}}||V^\top|$.

$$\begin{aligned} \min_{\tilde{A},I}\mathcal{L} &= (1-\beta)\log\left(|\tilde{A}\Sigma_{\boldsymbol{x}}\tilde{A}^\top + I|\right) - \log\left(|I|\right) + \beta\log\left(|\tilde{A}\Sigma_{\boldsymbol{x}|\boldsymbol{y}}\tilde{A}^\top + I|\right) \\ &= (1-\beta)\log\left(|\sqrt{D^{-1}}VA\Sigma_{\boldsymbol{x}}A^\top V^\top\sqrt{D^{-1}} + I|\right) - \log\left(|I|\right) \\ &\qquad + \beta\log\left(|\sqrt{D^{-1}}VA\Sigma_{\boldsymbol{x}|\boldsymbol{y}}A^\top V^\top\sqrt{D^{-1}} + I|\right) \\ &= (1-\beta)\log\left(|\sqrt{D^{-1}}VA\Sigma_{\boldsymbol{x}}A^\top V^\top\sqrt{D^{-1}} + \Sigma_\xi V\sqrt{D^{-1}}\sqrt{D^{-1}}V^\top|\right) \\ &\quad - \log\left(|V\sqrt{D^{-1}}\Sigma_\xi\sqrt{D^{-1}}V^\top|\right) \\ &\quad + \beta\log\left(|\sqrt{D^{-1}}VA\Sigma_{\boldsymbol{x}|\boldsymbol{y}}A^\top V^\top\sqrt{D^{-1}} + \Sigma_\xi V\sqrt{D^{-1}}\sqrt{D^{-1}}V^\top|\right) \\ &= (1-\beta)\log\left(C|A\Sigma_{\boldsymbol{x}}A^\top + \Sigma_\xi|C\right) - \log\left(C|\Sigma_\xi|C\right) + \beta\log\left(C|A\Sigma_{\boldsymbol{x}|\boldsymbol{y}}A^\top + \Sigma_\xi|C\right) \\ &= 2(1-\beta)\log(C) + (1-\beta)\log\left(|A\Sigma_{\boldsymbol{x}}A^\top + \Sigma_\xi\right) - 2\log(C) + \beta\log\left(|\Sigma_\xi|\right) + 2\beta\log(C) \\ &\quad + \beta\log\left(|A\Sigma_{\boldsymbol{x}|\boldsymbol{y}}A^\top + \Sigma_\xi|\right) \\ &= (1-\beta)\log\left(|A\Sigma_{\boldsymbol{x}}A^\top + \Sigma_\xi|\right) - \log\left(|\Sigma_\xi|\right) + \beta\log\left(|A\Sigma_{\boldsymbol{x}|\boldsymbol{y}}A^\top + \Sigma_\xi|\right) \\ &= \min_{A,\Sigma_\xi}\mathcal{L} \end{aligned} \tag{A.30}$$

[6] V is an orthogonal square matrix (positive semi-definite matrix), hence we have $I = \Sigma_\xi\Sigma_\xi^{-1} = \Sigma_\xi(VDV^\top)^{-1} = \Sigma_\xi(V^\top)^{-1}D^{-1}V^{-1} = \Sigma_\xi(V^{-1})^{-1}D^{-1}V^\top = \Sigma_\xi(V^{-1})^{-1}\sqrt{D^{-1}}\sqrt{D^{-1}}V^\top = \Sigma_\xi V\sqrt{D^{-1}}\sqrt{D^{-1}}V^\top$

Taking derivative of $\mathcal{L}$ with respect to A, we have Equation A.31.

$$\frac{\partial \mathcal{L}}{\partial A} = (1-\beta)\left(A\Sigma_{\boldsymbol{x}}A^\top + I\right)^{-1} \times 2A\Sigma_{\boldsymbol{x}} + \beta\left(A\Sigma_{\boldsymbol{x}|\boldsymbol{y}}A^\top + I\right)^{-1} \times 2A\Sigma_{\boldsymbol{x}|\boldsymbol{y}} = 0 \quad \text{(A.31)}$$

$$\begin{aligned} \frac{(1-\beta)}{\beta}\left(A\Sigma_{\boldsymbol{x}}A^\top + I\right)^{-1} \times 2A\Sigma_{\boldsymbol{x}} &= -\left(A\Sigma_{\boldsymbol{x}|\boldsymbol{y}}A^\top + I\right)^{-1} \times 2A\Sigma_{\boldsymbol{x}|\boldsymbol{y}} \\ \frac{(\beta-1)}{\beta}\frac{\left(A\Sigma_{\boldsymbol{x}|\boldsymbol{y}}A^\top + I\right)}{\left(A\Sigma_{\boldsymbol{x}}A^\top + I\right)}A &= A\left[\Sigma_{\boldsymbol{x}|\boldsymbol{y}}\Sigma_{\boldsymbol{x}}^{-1}\right] \end{aligned} \quad \text{(A.32)}$$

Rewriting Equation A.31, we will have Equation A.32. Notice the similar form to $\lambda V = XV$. If all the variables are scalar, the optimal projection matrix A will be a row vector. And then, there are two solutions of Equation A.32: (a) A must identically be zero, and (b) A is the eigenvector of $\left[\Sigma_{\boldsymbol{x}|\boldsymbol{y}}\Sigma_{\boldsymbol{x}}^{-1}\right]$ with an eigenvalue $\frac{(\beta-1)}{\beta}\frac{\left(A\Sigma_{\boldsymbol{x}|\boldsymbol{y}}A^\top+1\right)}{\left(A\Sigma_{\boldsymbol{x}}A^\top+1\right)}$. For obtaining a non-degenerate solution of this optimisation task, there is a constraint, $\lambda \leq \frac{\beta-1}{\beta}$, on β and λ[7].

While all variables are high dimensional, the multiplication of $\Sigma_{\boldsymbol{x}|\boldsymbol{y}}\Sigma_{\boldsymbol{x}}^{-1}$ by A must reside in the span of the rows of A, so that A is the eigenvector of $\Sigma_{\boldsymbol{x}|\boldsymbol{y}}\Sigma_{\boldsymbol{x}}^{-1}$. This means that A should be spanned up to n_{th} eigenvectors of $\Sigma_{\boldsymbol{x}|\boldsymbol{y}}\Sigma_{\boldsymbol{x}}^{-1}$. Therefore, by representing $A = WV$ where the rows of V are left normalised eigenvectors of $\Sigma_{\boldsymbol{x}|\boldsymbol{y}}\Sigma_{\boldsymbol{x}}^{-1}$ and W is a mixing matrix that weights these eigenvectors, we have the optimal solution as shown in Equation A.33 (more details of proof can be found in (Chechik et al., 2005)), Where $\{\boldsymbol{V}_1^\top, \boldsymbol{V}_2^\top, \ldots, \boldsymbol{V}_{n_{\boldsymbol{x}}}^\top\}$ are the left eigenvectors of $\Sigma_{\boldsymbol{x}|\boldsymbol{y}}\Sigma_{\boldsymbol{x}}^{-1}$ sorted by their corresponding ascending eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{n_{\boldsymbol{x}}}$. β_i^c are critical β values, $\alpha_i \equiv \sqrt{\frac{\beta(1-\lambda_i)-1}{\lambda_i\gamma_i}}$ are coefficients, $\gamma_i \equiv \boldsymbol{v}_i^\top\Sigma_{\boldsymbol{x}}\boldsymbol{v}_i$, $\mathbf{0}^\top$ is an $n_{\boldsymbol{x}}$ dimensional row vector of zeros, and semicolons separate rows in the matrix A.

Taken together, for a given value of β, the optimal projection is obtained by taking all the eigenvectors whose eigenvalues λ_i satisfy $\beta \geq \frac{1}{1-\lambda_i}$, and setting their norm according to $A = WV$ with W determined by λ, γ and β. Specifically, with the increment of β, the projection variable T will expend the dimensions. Continually, the relative magnitude of each basis vector is re-scaled through a series of critical points (structural phase transitions) until all the relevant information about Y is captured by T, which provides a continuous measure of model complexity.

$$A = \begin{Bmatrix} [\mathbf{0}^\top; \ldots; \mathbf{0}^\top] & 0 \leq \beta \leq \beta_1^c \\ \left[\alpha_1\boldsymbol{V}_1^\top, \mathbf{0}^\top; \ldots; \mathbf{0}^\top\right] & \beta_1^c \leq \beta \leq \beta_2^c \\ \left[\alpha_1\boldsymbol{V}_1^\top, \alpha_2\boldsymbol{V}_2^\top, \mathbf{0}^\top; \ldots; \mathbf{0}^\top\right] & \beta_2^c \leq \beta \leq \beta_3^c \\ \vdots & \end{Bmatrix} \quad \text{(A.33)}$$

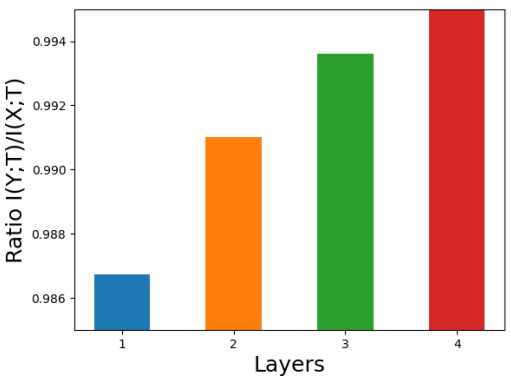
[7]By denoting $\gamma = \frac{A\Sigma_{\boldsymbol{x}}A^\top}{\|A\|^2}$ as the norm of $\Sigma_{\boldsymbol{X}}$ with respect to A, γ will be positive. As $A\Sigma_{\boldsymbol{xy}}\Sigma_{\boldsymbol{x}}^{-1}\Sigma_{\boldsymbol{x}}A^\top = \lambda A\Sigma_{\boldsymbol{x}}A^\top$, we have $A\Sigma_{\boldsymbol{xy}}\Sigma_{\boldsymbol{x}}^{-1}\Sigma_{\boldsymbol{x}}A^\top = \lambda\gamma\|A\|^2$. Rewriting the eigenvalues, we get $\lambda = \frac{\beta-1}{\beta}\frac{\lambda\gamma\|A\|^2+1}{\gamma\|A\|^2+1}$, and then $0 < \|A\|^2 = \frac{\beta(1-\lambda)-1}{\gamma\lambda}$.

A.7.3 Connection between the Gaussian Information Bottleneck and Canonical Correlation Analysis

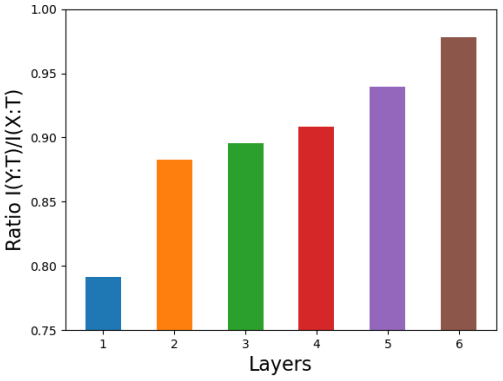
As described in subsections A.7.1 and A.7.2, an obtained optimal representation in Gaussian IB theory is a noisy linear projection to eigenvectors of the normalised regression matrix $\Sigma_{\boldsymbol{x}|\boldsymbol{y}}\Sigma_{\boldsymbol{x}}^{-1}$[8], which is also the basis obtained in canonical correlation analysis (shown in Equations A.26 and A.32). However, the nature of the projection is determined by the parameter β in Gaussian IB theory. In addition to this connection, there are two differences between them: (a) Gaussian IB characterises not only the eigenvectors but also their norm. In contrary, canonical correlation analysis is invariant to a re-scaling of the projection vectors as the correlation coefficient between the projected versions of X and Y is a normalised measure of correlation; (b) canonical correlation analysis can be seen as symmetric as both X and Y are projected to the subspace, while Gaussian IB is non-symmetric, where only the input variables are compressed.

On the top of canonical correction analysis, Raghu et al. (2017) first applied the empirical work of SVCCA into analysing representations between the hidden layer of two networks. They show that the early layers of an E2E network converge faster than later layers in training. We further apply this tool to investigate the learning nature of CL. Before applying canonical correction analysis, the SVCCA uses Singular Value Decomposition (SVD) to select the top n singular values and filters out the lowest 1% in order to remove some directions or neurons that are constantly zero. That is we use SVD to filter 99% of variance-preserving principal components, and then we apply canonical correlation analysis to calculate the canonical similarity in the sub-spaces X' and Y'. We utilise SVCCA, IB and Gaussian IB track the learning dynamics in this work for investigating their connections (see Section 3.5.4 and A.5.2).

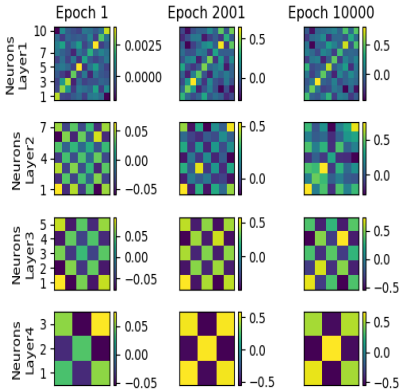
[8] $\Sigma_{\boldsymbol{x}|\boldsymbol{y}} = \Sigma_{\boldsymbol{x}} - \Sigma_{\boldsymbol{x}\boldsymbol{y}}\Sigma_{\boldsymbol{y}}^{-1}\Sigma_{\boldsymbol{y}\boldsymbol{x}}$, so that $\Sigma_{\boldsymbol{x}|\boldsymbol{y}}\Sigma_{\boldsymbol{x}^{-1}} = I - \Sigma_{\boldsymbol{x}} - \Sigma_{\boldsymbol{x}\boldsymbol{y}}\Sigma_{\boldsymbol{y}}^{-1}\Sigma_{\boldsymbol{y}\boldsymbol{x}}\Sigma_{\boldsymbol{x}^{-1}}$.



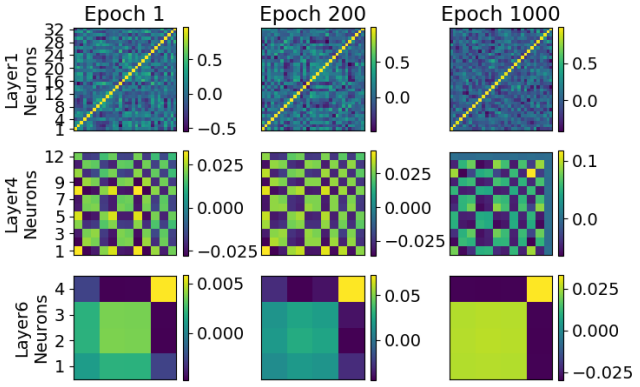
(a) Synthetic 17



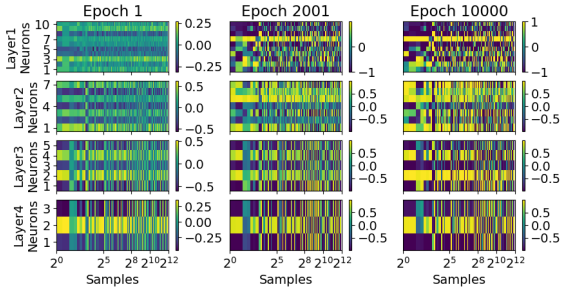
(b) Epileptic



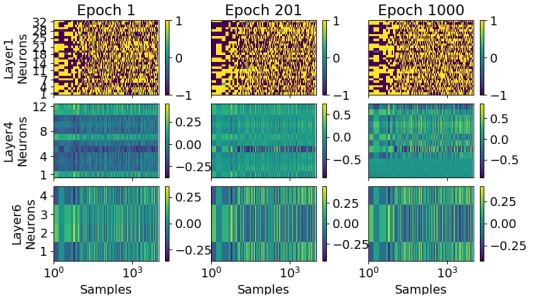
(c) Covariance matrices (Synthetic 17)



(d) Covariance matrices (Epileptic)

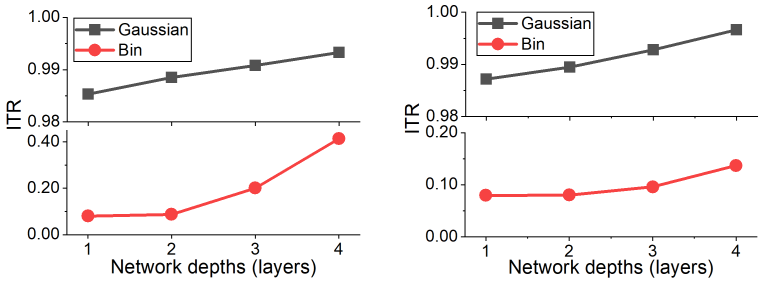


(e) Intensity of outputs (Synthetic 17)

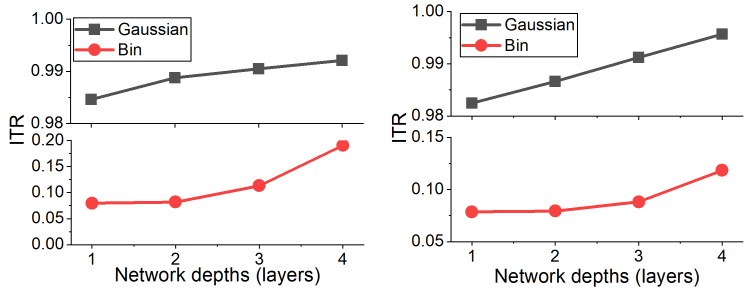


(f) Intensity of outputs (Epileptic)

FIGURE A.17: Gaussianisation results of CL on the synthetic 17 (the first column) and Epileptic datasets (the second column). By assuming all variables are Gaussian, the first row ((a) and (b)) provides corresponding information ratios; the second row ((c) and (d)) shows covariance matrices of each hidden layer's outputs with training epochs; and the third row ((e) and (f)) provides the intensity of every layer's outputs over training stages.



(a) Synthetic 17 (b) Synthetic 28



(c) Synthetic 44 (d) Synthetic 48

FIGURE A.18: Comparisons of ITR estimated by binning and Gaussian estimators on synthetic datasets. Each of sub-figures shows results on one synthetic dataset. Gaussian estimators are given by assuming all variables to be Gaussian variables. Clearly, the sharp increase of ITR, estimated by Gaussian estimators, happens at a layer differing from the layer where the ITR, estimated by binning method, sharply increases.

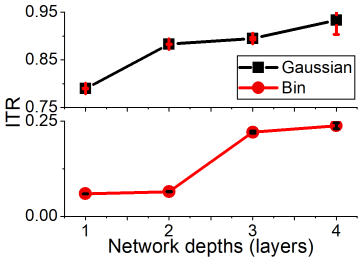


FIGURE A.19: A comparison of ITR between estimations from binning and Gaussian estimators on the Epileptic dataset. In total, there are two runs. In consistent with Figure A.18, the ITR given by two estimators shows different tendencies.

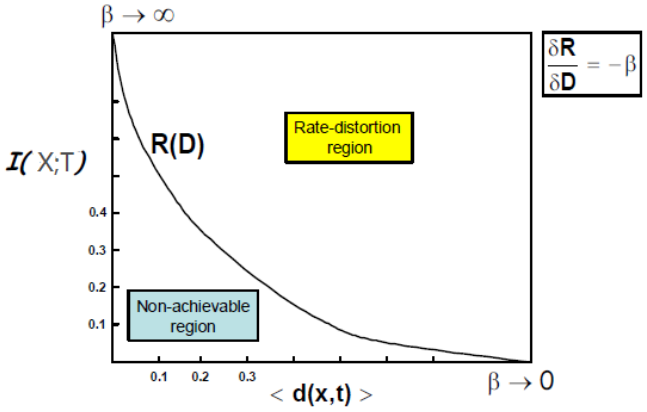


FIGURE A.20: An illustration of a rate distortion plane (Slonim, 2002). The function $R(D)$ separates two regions, the rate distortion region and the non achievable region.

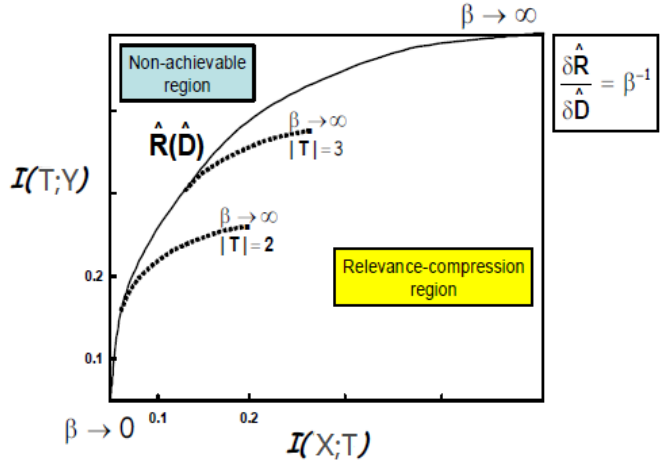


FIGURE A.21: An illustration of a relevance-compression function $\hat{R}(\hat{D})$ (Slonim, 2002). The function $\hat{R}(\hat{D})$ (the solid line) defines a monotonic concave curve in the relevance-compression plane. The region above the curve is a non-achievable region, and the region below the curve is a achievable relevance compression region. The bifurcations in the relevance compression region are induced by the specific cardinality of T where the solid curve corresponds to $|\mathcal{T}| = |\mathcal{X}|$.

Appendix B

B.1 Collection of Single Cell Data

To provide better understanding of Single Cell (SC) data containing high-resolution characterisation of cells this section will give a brief summary of SCRNA-Sequencing (RNA-Seq) processing steps as shown in Figure B.1. Compared to the major breakthrough of bulk RNA-Seq in the late 2000's, SCRNA-Seq is an emerging technology first proposed by Tang et al. (2009). Limited by the tremendous cost of sequencing and protocols, SCRNA-Seq did not gain widespread popularity until 2014 when several protocols (used in the amplification step in Figure B.1) were proposed (e.g., STRT-seq by Islam et al. (2014), CEL-seq by Hashimshony et al. (2012), Smart-seq by Picelli et al. (2014), SMART-seq2 by Picelli et al. (2013), MARS-seq by Jaitin et al. (2014), SCRB-seq by Soumillon et al. (2014), Drop-seq by Macosko et al. (2015), InDrop-seq by Klein et al. (2015), CEL-seq2 by Hashimshony et al. (2016), and Seq-well by Gierahn et al. (2017)) as shown in Figure B.2.

Currently, SCRNA-Seq consists of isolating the SC and the corresponding Ribonucleic Acid (RNA) as shown in Figure B.1. The following steps, being the same as bulk RNA-Seq method, are: Reverse Transcription (RT), amplification, library generation and sequencing. RT converts RNA to complementary DNA (cDNA) labelled by an unique barcode. To do so, individual cells are either divided into separated wells (like in early methods) or encapsulated in droplets into a microfluidic device (like in current methods). Then, the cDNA will be mixed for sequencing and the barcode is used to identify the transcripts from a particular cell. Before sequencing, the amplification step has two options, polymerase chain reaction (PCR) or in vitro transcription (IVT), used to amplify RNA. After sequencing, the SC gene expression data, ranging from 10^2 to 10^6 cells, can be analysed by machine learning algorithms.

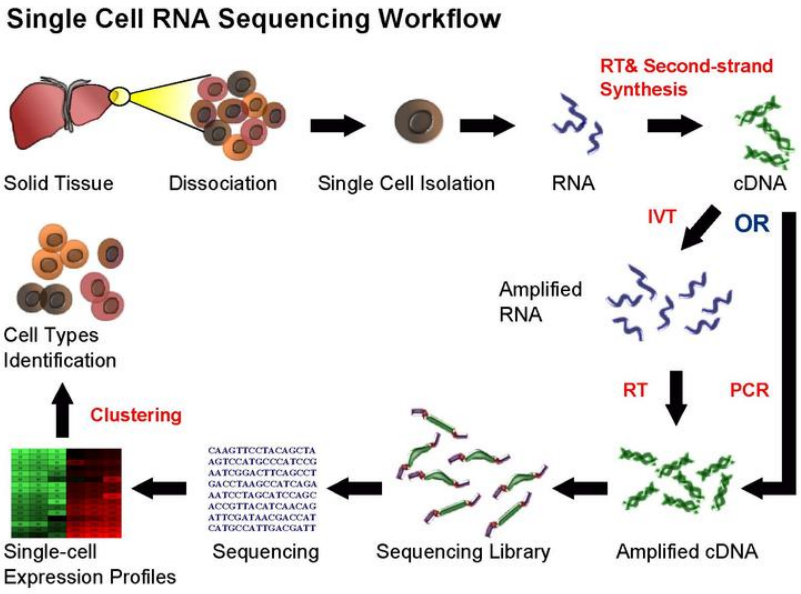


FIGURE B.1: SCRNA-Seq workflow (taken from Wikipedia (contributors, 2020)).

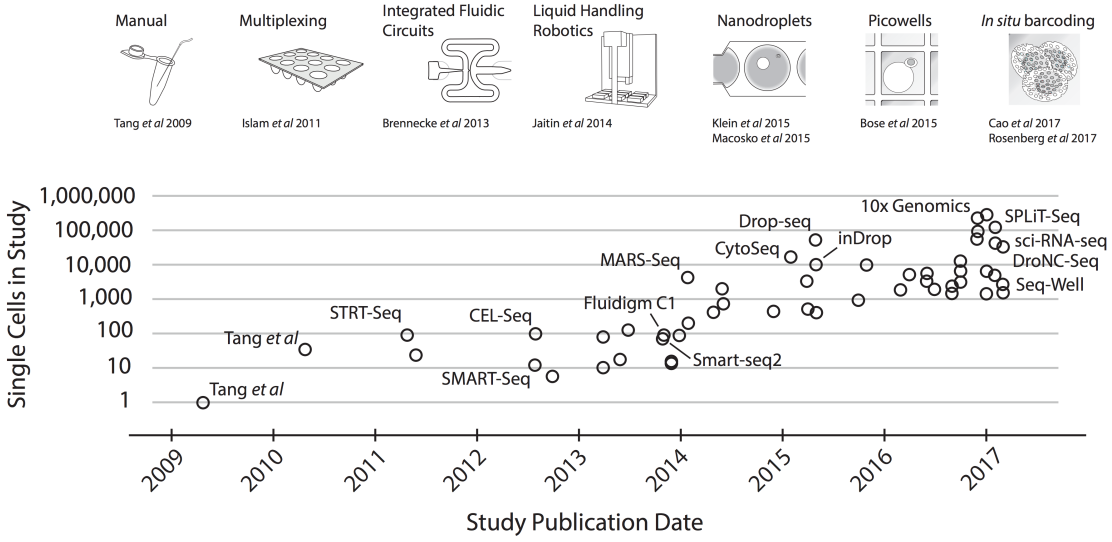
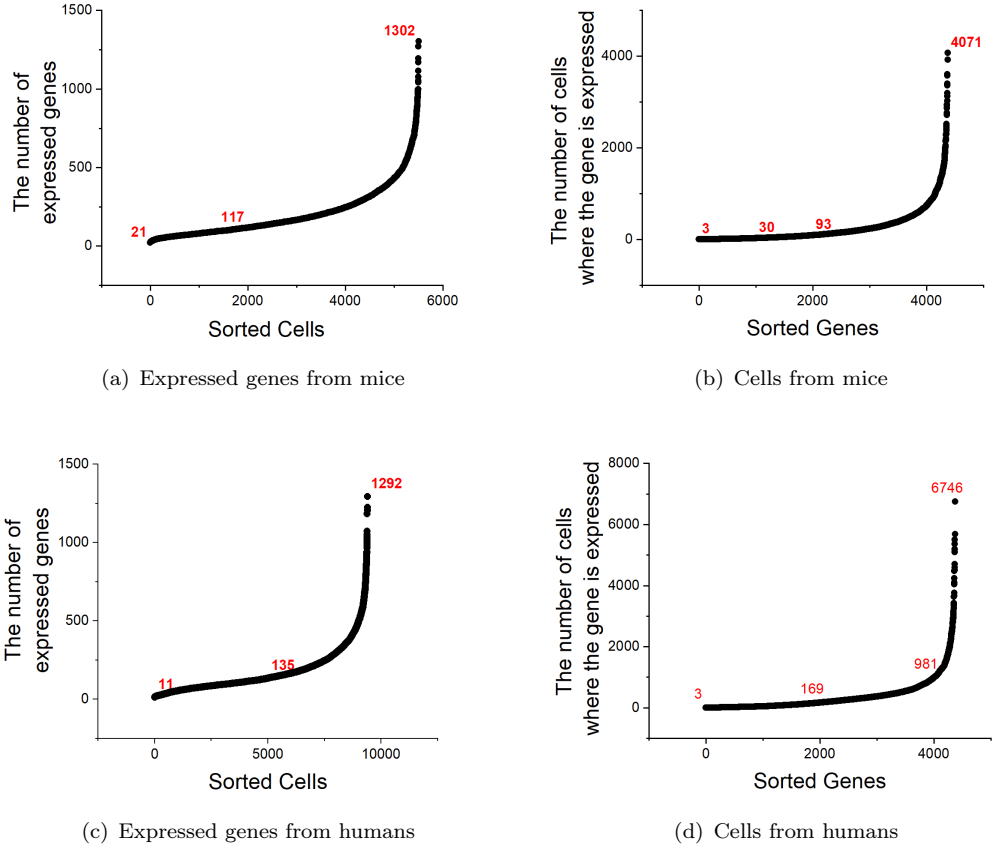


FIGURE B.2: The development of SC sequencing protocols (image taken from Macosko et al. (2015)).

B.2 Bone Marrow Datasets

As mentioned in Section 4.2.1, SC data is high dimensional and sparse. Taking the Bone Marrow (BM) dataset as an example, we analyse the data through sorting cells according to the number of expressed genes in the cells, and sorting genes in terms of the number of cells where the genes are expressed. As shown in Figures B.3(a) and B.3(c), a great number of cells contain information from only a small part of expressed genes. Figure B.3(b) and B.3(d) show that only a few genes are expressed in many cells. Majority of

genes are not expressed in part of cells, which supports results that a small number of genes significantly contribute to recognise cell types (see Figure 4.9).



(a) Expressed genes from mice

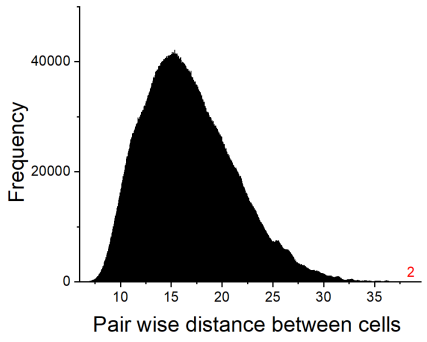
(b) Cells from mice

(c) Expressed genes from humans

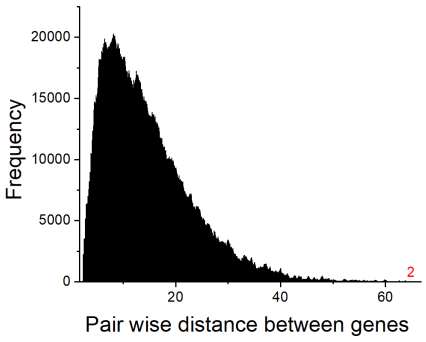
(d) Cells from humans

FIGURE B.3: Variations in number of expressed genes per cell and number of cells in which a gene is expressed for mouse and human SCRNA-Seq data. (a) and (b) are from a mouse dataset. Where (a) presents the distribution of expressed genes over sorted cells, and (b) shows the distribution of cells where the sorted gene expressed. (c) and (d) illustrate a same substance but on a human dataset. Numbers marked in red corresponds to values on the y axis.

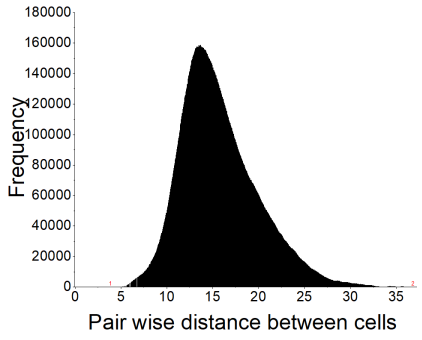
We further calculate the pair wise distance between cells and pair wise distance between genes to investigate the difficulty of distinguishing the cell types, showing results in Figure B.4. As Figures B.4(a) and B.4(c) do not show the number of peaks related to the cell classes, we can infer this using the distance to classify the cells may be not feasible.



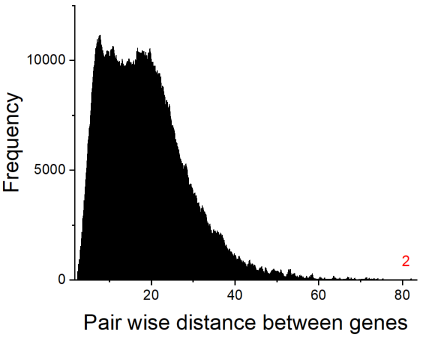
(a) Pair wise distance between cells from mice



(b) Pair wise distance between genes of mice



(c) Pair wise distance between cells from humans



(d) Pair wise distance between genes of humans

FIGURE B.4: Distributions of pair wise distances across genes and cells for the BM dataset from both mice (see (a) and (b)) and humans (see (c) and (d)). The red numbers are the values corresponding to y axis.

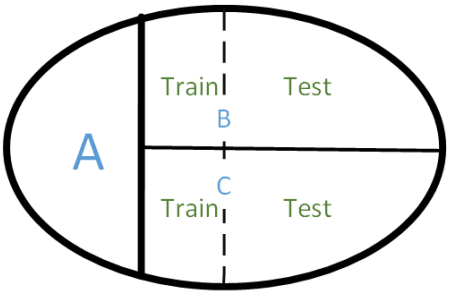


FIGURE B.5: The diagram of data separation in scenarios 5 *and* 6 (in Table 4.2). A is randomly selected and used as the source domain data to train the model. Parts B and C are used as two individual domains. For each domain, we divide the data into train and test datasets. The corresponding percent of training data is scenario-specific and the percent of the test data is fixed as 50%.

B.2.1 Mouse and Human Bone Marrow Tissue

As described by Stumpf et al. (2020), the 8-week old C57BL/6 female mice[1] were used to collect the mouse BM tissues. The human BM tissues are collected from patients undergoing routine hip replacement surgery, with informed consent, under the regional ethics committee' approval (reference 18/NW/0231). More details can be found in the work given by Stumpf et al.. Figure B.5 shows how data is constructed in scenarios 5 and 6 as described in Section 4.3.3. For comparing with scenario 4, the test data is fixed as 50%. The train sets in B and C are selected with a growing percentage, and corresponding results are shown in Figure 4.13(b).

B.2.2 Transfer Learning on Bone Marrow Data

As mentioned in Section 4.4.4, Figure B.6 shows a comparison of different metrics used to measure Transfer Learning (TL) performance. In that case, the target domain classifier is pre-trained on the source domain and retrained on the growing number of data points from the target domain. Using human data to test a pre-trained model without retaining, we obtain high precision (majority of cells classified as a type are truly from this type) and low recall (a small number of cells from a type are correctly recognised). With the increasing of cells used to retrain models, recall is further improved.

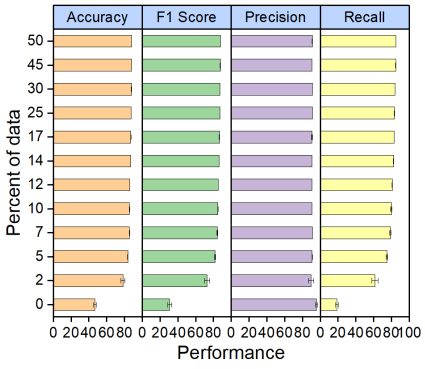


FIGURE B.6: The comparison of performance metrics on BM dataset. With respect to the percent of data used to retrain the classifier, the performance of TL on the target domain is measured by accuracy, F_1 score, precision and recall. 0 percent stands for the direct test on the data in the target domain without retraining following scenario 3 shown in Table 4.2.

[1]C57BL/6, usually referred to as "C57 black 6", "C57" or "black 6", is a king of common laboratory inbred strain mouse. It can be seen as the "genetic background" of genetically modified mice which will be used as models of human disease. As best-selling mouse strain, they are widely used due to the availability of congenic strains, easy breeding, and robustness (Köntgen et al., 1993).

Appendix C

C.1 Transfer Learning from Finer Classes to Coarser Classes within Opportunity

The results shown in the Section 5.4 suggest a monotonic decline in transferability from features taken from early layers to later ones from cascade networks. However, the later layers are necessary for increased recognition accuracy of the source domain classification. This observation confirms that cascade training packs feature information in a specific way, with coarse information in early layers and finer details related to the source task in later layers.

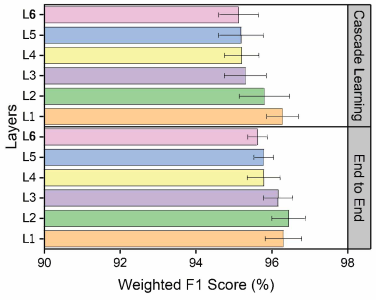


FIGURE C.1: TL from 14-class to 2-class within Opportunity. Evaluated by weighted F_1 score (%), features from early layers of a cascade network show better transferability to this coarser target task.

To explore the coarseness of features from variant layers of a network, we also check the performance of Transfer Learning (TL) from 14-class to binary (Type one & Type two) based on both cascade networks and End-to-End (E2E) networks. In this setting, the task in the target domain is the coarser classification problems than the task in the source domain, hence coarse features learned by early layers can provide benefits to this coarser task. As shown in Figure C.1, we notice the performance decreases monotonically layer by layer with the increasing depth of cascade networks, while the performance of Transfer Learning from End-to-End (TE2E) has non-monotonic trend across layers.

This observation further reinforces our assumption that Cascade Learning (CL) extracts features from coarse to fine layer by layer.

Bibliography

P. Agarwal and M. Alam, "A lightweight deep learning model for human activity recognition on edge devices," *Procedia Computer Science*, vol. 167, pp. 2364–2373, 2020.

K. Ahmed, M. H. Baig, and L. Torresani, "Network of experts for large-scale image categorization," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 516–532.

N. A. Ahmed and D. Gokhale, "Entropy expressions and their estimators for multivariate distributions," *IEEE Transactions on Information Theory*, vol. 35, no. 3, pp. 688–692, 1989.

T. Ai, Z. Yang, H. Hou, C. Zhan, C. Chen, W. Lv, Q. Tao, Z. Sun, and L. Xia, "Correlation of chest CT and RT-PCR testing in coronavirus disease 2019 (COVID-19) in China: a report of 1014 cases," *Radiology*, vol. 296, no. 2, pp. 642–665, 2020.

F. Al Machot, M. R Elkobaisi, and K. Kyamakya, "Zero-shot human activity recognition using non-visual sensors," *Sensors*, vol. 20, no. 3, pp. 1–17, 2020.

R. A. Amjad and B. C. Geiger, "Learning representations for neural network-based classification using the information bottleneck principle," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 9, pp. 2225–2239, 2020.

R. A. Amjad and B. C. Geiger, "How (not) to train your neural network using the information bottleneck principle," *CoRR, abs/1802.09766*, 2018.

R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger, "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state," *Physical Review E*, vol. 64, no. 6, pp. 061 907–061 915, 2001.

D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones." in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, vol. 3, 2013, pp. 437–442.

J. Arevalo, F. A. González, R. Ramos-Pollán, J. L. Oliveira, and M. A. G. Lopez, "Convolutional neural networks for mammography mass lesion classification," in *the 37th*

Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, 2015, pp. 797–800.

S. Arimoto, "An algorithm for computing the capacity of arbitrary discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 14–20, 1972.

H. Azizpour, A. Sharif Razavian, J. Sullivan, A. Maki, and S. Carlsson, "From generic to specific deep representations for visual recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) workshops*, 2015, pp. 36–45.

E. R. Balda, A. Behboodi, and R. Mathar, "An information theoretic view on learning of artificial neural networks," in *12th International Conference on Signal Processing and Communication Systems (ICSPCS)*. IEEE, 2018, pp. 1–8.

Y. Bar, I. Diamant, L. Wolf, and H. Greenspan, "Deep learning with non-medical training used for chest pathology identification," in *Medical Imaging: Computer-Aided Diagnosis*, vol. 9414, 2015, pp. 30–37.

M. Baron, A. Veres, S. L. Wolock, A. L. Faust, R. Gaujoux, A. Vetere, J. H. Ryu, B. K. Wagner, S. S. Shen-Orr, A. M. Klein *et al.*, "A single-cell transcriptomic map of the human and mouse pancreas reveals inter-and intra-cell population structure," *Cell Systems*, vol. 3, no. 4, pp. 346–360, 2016.

R. H. Baxter, N. M. Robertson, and D. M. Lane, "Human behaviour recognition in data-scarce domains," *Pattern Recognition*, vol. 48, no. 8, pp. 2377–2393, 2015.

N. J. Beaudry and R. Renner, "An intuitive proof of the data processing inequality," *Quantum Information & Computation*, vol. 12, no. 5-6, pp. 432–441, 2012.

M. A. Bedell, N. A. Jenkins, and N. G. Copeland, "Mouse models of human disease. part i: techniques and resources for genetic analysis in mice." *Genes & Development*, vol. 11, no. 1, pp. 1–10, 1997.

M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual information neural estimation," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018, pp. 531–540.

E. Belilovsky, M. Eickenberg, and E. Oyallon, "Greedy layerwise learning can scale to ImageNet," in *Proceedings of the International Conference on Machine Learning (ICML)*, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 583–593.

E. Belilovsky, M. Eickenberg, and E. Oyallon, "Decoupled greedy learning of CNNs," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 736–745.

Y. Bengio, *Learning deep architectures for AI.* Found. Trends Mach. Learn, 2009, vol. 2.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS).* MIT Press, 2006, pp. 153–160.

T. Berger, "Rate-distortion theory," *Wiley Encyclopedia of Telecommunications*, 2003.

R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, 1972.

A. Brock, T. Lim, J. M. Ritchie, and N. J. Weston, "Freezeout: Accelerate training by progressively freezing layers," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS) Workshop on Optimization for Machine Learning*, 2017.

A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys (CSUR)*, vol. 46, no. 3, pp. 1–33, 2014.

K. P. Burnham and D. R. Anderson, "Multimodel inference: understanding AIC and BIC in model selection," *Sociological Methods & Research*, vol. 33, no. 2, pp. 261–304, 2004.

A. Calatroni, D. Roggen, and G. Tröster, "Automatic transfer of activity recognition capabilities between body-worn motion sensors: Training newcomers to recognize locomotion," in *8th International Conference on Networked Sensing Systems (INSS)*, 2011.

V. Capoyleas, G. Rote, and G. Woeginger, "Geometric clusterings," *Journal of Algorithms*, vol. 12, no. 2, pp. 341–356, 1991.

G. Carneiro, J. Nascimento, and A. P. Bradley, "Unregistered multiview mammogram analysis with pre-trained deep learning models," in *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer, 2015, pp. 652–660.

R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.

D. C. Castro, I. Walker, and B. Glocker, "Causality matters in medical imaging," *Nature Communications*, vol. 11, no. 1, pp. 1–10, 2020.

G. Chechik, A. Globerson, N. Tishby, and Y. Weiss, "Information bottleneck for Gaussian variables," *Journal of Machine Learning Research*, vol. 6, no. Jan, pp. 165–188, 2005.

K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *10th International Workshop on Frontiers in Handwriting Recognition.* Suvisoft, 2006.

I. Chelombiev, C. Houghton, and C. O'Donnell, "Adaptive estimators show information compression in deep neural networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

H. Chen, D. Ni, J. Qin, S. Li, X. Yang, T. Wang, and P. A. Heng, "Standard plane localization in fetal ultrasound via domain transferred deep neural networks," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 5, pp. 1627–1636, 2015.

H. Cheng, D. Lian, S. Gao, and Y. Geng, "Evaluating capability of deep neural networks for image classification via information plane," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 168–182.

H. Cheng, D. Lian, S. Gao, and Y. Geng, "Utilizing information bottleneck to evaluate the capability of deep neural networks for image classification," *Entropy*, vol. 21, no. 5, pp. 456–479, 2019.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS) Workshop on Deep Learning*, 2014.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 3606–3613.

A. Claudio Quiros, R. Murray-Smith, and K. Yuan, "PathologyGAN: Learning deep representations of cancer tissue," *Journal of Machine Learning for Biomedical Imaging*, vol. 2021, no. 4, pp. 1–48, 2021.

B. Cleary, L. Cong, A. Cheung, E. S. Lander, and A. Regev, "Efficient generation of transcriptomic profiles by random composite measurements," *Cell*, vol. 171, no. 6, pp. 1424–1436, 2017.

W. Commons, "File:blausen 0993 pleural effusion.png — wikimedia commons, the free media repository," 2021, [Online; accessed 22-September-2021]. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:Blausen_0993_PleuralEffusion.png&oldid=565093767

W. contributors, "Single cell sequencing — Wikipedia, the free encyclopedia," 2020, [Online; accessed 28-October-2020]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Single_cell_sequencing&oldid=984417631

D. Cook, K. D. Feuz, and N. C. Krishnan, "Transfer learning for activity recognition: A survey," *Knowledge and Information Systems*, vol. 36, no. 3, pp. 537–556, 2013.

C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, and S. Yang, "Adanet: Adaptive structural learning of artificial neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 874–883.

T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 1999.

I. Csiszár, "Information geonetry and alternating minimization procedures," *Statistics and Decisions*, vol. 1, pp. 205–237, 1984.

G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 8609–8613.

L. N. Darlow and A. Storkey, "What information does a ResNet compress?" *arXiv preprint arXiv:2003.06254*, 2019. [Online]. Available: https://openreview.net/forum?id=HklbTjRcKX

H. Daume III and D. Marcu, "Domain adaptation for statistical classifiers," *Journal of Artificial Intelligence Research*, vol. 26, pp. 101–126, 2006.

M. de la Iglesia Vayá, J. M. Saborit-Torres, J. A. Montell Serrano, E. Oliver-Garcia, A. Pertusa, A. Bustos, M. Cazorla, J. Galant, X. Barber, D. Orozco-Beltrán, F. García-García, M. Caparrós, G. González, and J. M. Salinas, "BIMCV COVID-19-: a large annotated dataset of RX and CT images from COVID-19 patients," 2021. [Online]. Available: https://dx.doi.org/10.21227/m4j2-ap59

J. Deng, D. Wei, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 248–255.

D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017, pp. 1–32. [Online]. Available: https://arxiv.org/abs/1605.08803

E. R. Dougherty, E. J. Kraus, and J. B. Pelz, "Image segmentation by local morphological granulometries," in *Proceedings of the 12th Canadian Symposium on Remote Sensing Geoscience and Remote Sensing Symposium*, vol. 3. IEEE, 1989, pp. 1220–1223.

X. Du, K. Farrahi, and M. Niranjan, "Transfer learning across human activities using a cascade neural network architecture," in *Proceedings of the 23rd International Symposium on Wearable Computers (ISWC)*. ACM, 2019, pp. 35–44.

X. Du, K. Farrahi, and M. Niranjan, "Information bottleneck theory based exploration of cascade learning," *Entropy*, vol. 23, no. 10, pp. 1–16, 2021. [Online]. Available: https://www.mdpi.com/1099-4300/23/10/1360

D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

F. Dupuis, W. Yu, and F. M. Willems, "Blahut-Arimoto algorithms for computing channel capacity and rate-distortion with side information," in *International Symposium on Information Theory*. IEEE, 2004, p. 179.

M. Edel and E. Köppe, "Binarized-BLSTM-RNN based human activity recognition," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2016, pp. 1–7.

R. El-Yaniv, S. Fine, and N. Tishby, "Agnostic classification of Markovian sequences," *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, vol. 10, pp. 465–471, 1997.

A. Elad, D. Haviv, Y. Blau, and T. Michaeli, "The effectiveness of layer-by-layer training using the information bottleneck principle," 2019. [Online]. Available: https://openreview.net/forum?id=r1Nb5i05tX

D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, vol. 5. PMLR, 2009, pp. 153–160.

A. Esteva, K. Chou, S. Yeung, N. Naik, A. Madani, A. Mottaghi, Y. Liu, E. Topol, J. Dean, and R. Socher, "Deep learning-enabled medical computer vision," *NPJ Digital Medicine*, vol. 4, no. 1, pp. 1–9, 2021.

S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. Morgan Kaufmann Publishers Inc., 1990, pp. 524–532.

H. Fang, V. Wang, and M. Yamaguchi, "Dissecting deep learning networks visualizing mutual information," *Entropy*, vol. 20, no. 11, pp. 823–844, 2018. [Online]. Available: https://www.mdpi.com/1099-4300/20/11/823

Y. Fang, H. Zhang, J. Xie, M. Lin, L. Ying, P. Pang, and W. Ji, "Sensitivity of chest CT for COVID-19: comparison to RT-PCR," *Radiology*, vol. 296, no. 2, pp. 432–440, 2020.

A. Fischer and C. Igel, "An introduction to restricted Boltzmann machines," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: Iberoamerican Congress*, vol. 7441. Springer, 2012, pp. 14–36.

R. A. Fisher, "On the mathematical foundations of theoretical statistics," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 222, no. 594-604, pp. 309–368, 1922.

M. Gabrié, A. Manoel, C. Luneau, J. Barbier, N. Macris, F. Krzakala, and L. Zdeborová, "Entropy and mutual information in models of deep neural networks," *Journal of*

Statistical Mechanics: Theory and Experiment, vol. 2019, no. 12, pp. 1–16, dec 2019. [Online]. Available: https://doi.org/10.1088/1742-5468/ab3430

B. C. Geiger, "On information plane analyses of neural network classifiers–a review," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2021.

T. M. Gierahn, M. H. Wadsworth II, T. K. Hughes, B. D. Bryson, A. Butler, R. Satija, S. Fortune, J. C. Love, and A. K. Shalek, "Seq-well: portable, low-cost RNA sequencing of single cells at high throughput," *Nature Methods*, vol. 14, no. 4, pp. 395–398, 2017.

Z. Goldfeld, K. Greenewald, J. Weed, and Y. Polyanskiy, "Optimality of the plug-in estimator for differential entropy estimation under Gaussian convolutions," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2019, pp. 892–896.

Z. Goldfeld, E. Van Den Berg, K. Greenewald, I. Melnyk, N. Nguyen, B. Kingsbury, and Y. Polyanskiy, "Estimating information flow in deep neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 97. PMLR, 09–15 Jun 2019, pp. 2299–2308.

A. Griewank, "On automatic differentiation," *Mathematical Programming: Recent Developments and Applications*, vol. 6, no. 6, pp. 83–107, 1989.

Y. Guan and T. Plötz, "Ensembles of deep LSTM learners for activity recognition using wearables," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 2, pp. 1–28, 2017.

G. Guasch and E. Fuchs, "Mice in the world of stem cell biology," *Nature Genetics*, vol. 37, no. 11, pp. 1201–1206, 2005.

I. Guyon, S. R. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the NIPS 2003 feature selection challenge." in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, vol. 4, 2004, pp. 545–552.

N. Y. Hammerla, S. Halloran, and T. Plötz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," in *25th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2016, pp. 1533–1540.

M. Harel and S. Mannor, "Learning from multiple outlooks," in *Proceedings of the International Conference on Machine Learning (ICML)*. Madison, WI, USA: Omnipress, 2011, pp. 401—-408.

T. Hashimshony, F. Wagner, N. Sher, and I. Yanai, "CEL-Seq: single-cell RNA-Seq by multiplexed linear amplification," *Cell Reports*, vol. 2, no. 3, pp. 666–673, 2012.

T. Hashimshony, N. Senderovich, G. Avital, A. Klochendler, Y. de Leeuw, L. Anavy, D. Gennert, S. Li, K. J. Livak, O. Rozenblatt-Rosen *et al.*, "CEL-Seq2: sensitive highly-multiplexed single-cell RNA-Seq," *Genome Biology*, vol. 17, no. 1, pp. 77–84, 2016.

K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5353–5360.

K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

X. He, X. Yang, S. Zhang, J. Zhao, Y. Zhang, E. Xing, and P. Xie, "Sample-efficient deep learning for COVID-19 diagnosis based on CT scans," *medrxiv*, pp. 1–10, 2020.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The "wake-sleep" algorithm for unsupervised neural networks," *Science*, vol. 268, no. 5214, pp. 1158–1161, 1995.

G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

S. Hochreiter, "Recurrent neural net learning and vanishing gradient," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, pp. 107–116, 1998.

R. A. Horn, "The Hadamard product," in *Proc. Symp. Appl. Math*, vol. 40, 1990, pp. 87–169.

G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 4700–4708.

Y. Huang, W. Wang, L. Wang, and T. Tan, "Multi-task deep neural network for multi-label learning," in *IEEE International Conference on Image Processing*. IEEE, 2013, pp. 2897–2900.

M. F. Huber, T. Bailey, H. Durrant-Whyte, and U. D. Hanebeck, "On entropy approximation for Gaussian mixture random vectors," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, 2008, pp. 181–188.

T. Huynh and B. Schiele, "Analyzing features for activity recognition," in *Proceedings of the Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies*, 2005, pp. 159–163.

B. Hwang, J. H. Lee, and D. Bang, "Single-cell RNA sequencing technologies and bioinformatics pipelines," *Experimental & Molecular Medicine*, vol. 50, no. 8, pp. 1–14, 2018.

P. Indyk, R. Motwani, P. Raghavan, and S. Vempala, "Locality-preserving hashing in multidimensional spaces," in *Proceedings of the ACM Symposium on Theory of Computing*, 1997, pp. 618–625.

J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya *et al.*, "CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparison," in *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference*, vol. 33, 2019, pp. 590–597.

S. Islam, A. Zeisel, S. Joost, G. La Manno, P. Zajac, M. Kasper, P. Lönnerberg, and S. Linnarsson, "Quantitative single-cell RNA-Seq with unique molecular identifiers," *Nature Methods*, vol. 11, no. 2, pp. 163–166, 2014.

D. A. Jaitin, E. Kenigsberg, H. Keren-Shaul, N. Elefant, F. Paul, I. Zaretsky, A. Mildner, N. Cohen, S. Jung, A. Tanay *et al.*, "Massively parallel single-cell RNA-Seq for marker-free decomposition of tissues into cell types," *Science*, vol. 343, no. 6172, pp. 776–779, 2014.

M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado *et al.*, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 339–351, 2017.

H. Jónsson, G. Cherubini, and E. Eleftheriou, "Convergence behavior of DNNs with mutual-information-based regularization," *Entropy*, vol. 22, no. 7, pp. 727–741, 2020. [Online]. Available: https://www.mdpi.com/1099-4300/22/7/727

V. Kadirkamanathan and M. Niranjan, "A function estimation approach to sequential learning with neural networks," *Neural Computation*, vol. 5, no. 6, pp. 954–975, 1993.

T. Kailath, "The divergence and Bhattacharyya distance measures in signal selection," *IEEE Transactions on Communication Technology*, vol. 15, no. 1, pp. 52–60, 1967.

T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018, pp. 1–26.

A. Ke, W. Ellsworth, O. Banerjee, T. T Nguyen, and P. Rajpurkar, "Chextransfer: Performance and parameter efficiency of ImageNet models for chest X-Ray interpretation," in *Proceedings of the Conference on Health, Inference, and Learning (CHIL)*, 2021, pp. 116—-124.

P. V. Kharchenko, L. Silberstein, and D. T. Scadden, "Bayesian approach to single-cell differential expression analysis," *Nature Methods*, vol. 11, no. 7, pp. 740–742, 2014.

H. G. Kim, Y. Choi, and Y. M. Ro, "Modality-bridge transfer learning for medical image classification," in *International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE, 2017, pp. 1–5.

J. B. Kinney and G. S. Atwal, "Equitability, mutual information, and the maximal information coefficient," *Proceedings of the National Academy of Sciences*, vol. 111, no. 9, pp. 3354–3359, 2014.

V. Y. Kiselev, T. S. Andrews, and M. Hemberg, "Challenges in unsupervised clustering of single-cell RNA-Seq data," *Nature Reviews Genetics*, vol. 20, no. 5, pp. 273–282, 2019.

A. M. Klein, L. Mazutis, I. Akartuna, N. Tallapragada, A. Veres, V. Li, L. Peshkin, D. A. Weitz, and M. W. Kirschner, "Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells," *Cell*, vol. 161, no. 5, pp. 1187–1201, 2015.

A. Kolchinsky and B. D. Tracey, "Estimating mixture entropy with pairwise distances," *Entropy*, vol. 19, no. 7, pp. 361–378, 2017.

A. Kolchinsky, B. D. Tracey, and D. H. Wolpert, "Nonlinear information bottleneck," *Entropy*, vol. 21, no. 12, pp. 1181–1196, 2019.

F. Köntgen, G. Süss, C. Stewart, M. Steinmetz, and H. Bluethmann, "Targeted disruption of the MHC class II Aa gene in C57BL/6 mice," *International Immunology*, vol. 5, no. 8, pp. 957–964, 1993.

A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Tront, 2009.

A. Krizhevsky, V. Nair, and G. E. Hinton, "CIFAR-10 (Canadian institute for advanced research)," 2010. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84—-90, 2017.

M. Kulkarni and S. Karande, "Layer-wise training of deep networks using kernel similarity," in *Deep Learning for Pattern Recognition (DLPR) Workshop at International Conference on Pattern Recognition (ICPR)*, 2017.

S. Kullback, "Letters to the editor: The Kullback–Leibler distance," *The American Statistician*, vol. 41, no. 4, pp. 338–341, 1987.

C.-H. Kuo, F.-G. Huang, K.-L. Wang, M.-Y. Lee, and H.-W. Chen, "Development of internet based remote health and activity monitoring systems for the elders," *Journal of Medical and Biological Engineering*, vol. 24, no. 1, pp. 57–66, 2004.

M. Kurz, G. Hölzl, A. Ferscha, A. Calatroni, D. Roggen, and G. Tröster, "Real-time transfer and evaluation of activity recognition capabilities in an opportunistic system," *Machine Learning*, vol. 1, no. 7, pp. 8–14, 2011.

G. La Manno, D. Gyllborg, S. Codeluppi, K. Nishimura, C. Salto, A. Zeisel, L. E. Borm, S. R. Stott, E. M. Toledo, J. C. Villaescusa *et al.*, "Molecular diversity of midbrain development in mouse, human, and stem cells," *Cell*, vol. 167, no. 2, pp. 566–580, 2016.

O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 1990, pp. 598–605.

H. Lee, R. Grosse, R. Ranganath, and T. T Nguyen, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2009, pp. 609–616.

T. S. Lee and D. Mumford, "Hierarchical Bayesian inference in the visual cortex," *Journal of the Optical Socity of America A*, vol. 20, no. 7, pp. 1434–1448, 2003.

T. S. Lee, D. Mumford, R. Romero, and V. A. Lamme, "The role of the primary visual cortex in higher level vision," *Vision Research*, vol. 38, no. 15-16, pp. 2429–2454, 1998.

R. Lengellé and T. Denoeux, "Training MLPs layer-by-layer using an objective function for internal representations," *Neural Networks*, vol. 9, no. 1, pp. 83–97, 1996.

J. Li and D. Liu, "Information bottleneck methods on convolutional neural networks," *Neural Processing Letters*, vol. 53, pp. 1385–1400, 2021.

Y. Lieberman, L. Rokach, and T. Shay, "CaSTLe–classification of single cells by transfer learning: harnessing the power of publicly available single cell RNA sequencing experiments to annotate new experiments," *PloS One*, vol. 13, no. 10, pp. 1–16, 2018.

Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015, pp. 3730–3738.

M. Lotfollahi, M. Naghipourfar, M. D. Luecken, M. Khajavi, M. Buttner, Z. Avsec, A. V. Misharin, and F. J. Theis, "Query to reference single-cell integration with transfer learning," *BioRxiv*, 2020.

S. Löwe, P. O'Connor, and B. S. Veeling, "Putting an end to end-to-end: gradient-isolated learning of representations," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 3039–3051.

Y. Lu and C. L. Tan, "A nearest-neighbor chain based approach to skew estimation in document images," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2315–2323, 2003.

A. S. Lundervold and A. Lundervold, "An overview of deep learning in medical imaging focusing on MRI," *Zeitschrift für Medizinische Physik*, vol. 29, no. 2, pp. 102–127, 2019, special Issue: Deep Learning in Medical Physics.

W. Ma, M. Yu, K. Li, and G. Wang, "Why layer-wise learning is hard to scale-up and a possible solution via accelerated downsampling," in *IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2020, pp. 238–243.

E. Z. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. R. Bialas, N. Kamitaki, E. M. Martersteck *et al.*, "Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets," *Cell*, vol. 161, no. 5, pp. 1202–1214, 2015.

J. Margeta, A. Criminisi, R. Cabrera Lozoya, D. C. Lee, and N. Ayache, "Fine-tuned convolutional neural nets for cardiac MRI acquisition plane recognition," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 5, no. 5, pp. 339–349, 2017.

E. S. Marquez, J. S. Hare, and M. Niranjan, "Deep cascade learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 1–11, 2018.

W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

A. McDavid, G. Finak, P. K. Chattopadyay, M. Dominguez, L. Lamoreaux, S. S. Ma, M. Roederer, and R. Gottardo, "Data exploration, quality control and testing in single-cell qPCR-based gene expression experiments," *Bioinformatics*, vol. 29, no. 4, pp. 461–467, 2013.

J. Merkin, C. Russell, P. Chen, and C. B. Burge, "Evolutionary dynamics of gene and isoform regulation in mammalian tissues," *Science*, vol. 338, no. 6114, pp. 1593–1599, 2012.

B. Mieth, J. R. Hockley, N. Görnitz, M. M.-C. Vidovic, K.-R. Müller, A. Gutteridge, and D. Ziemek, "Using transfer learning from prior reference knowledge to improve the clustering of single-cell RNA-Seq data," *Scientific Reports*, vol. 9, no. 1, pp. 1–14, 2019.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, vol. 2. Curran Associates Inc., 2013, pp. 3111—-3119.

A. Milne, M. Nicolaou, and K. Farrahi, "Discovering the typing behaviour of parkinson's patients using topic models," in *International Conference on Social Informatics*. Springer, 2017, pp. 89–97.

M. Minsky and S. A. Papert, *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

K. R. Moon, K. Sricharan, K. Greenewald, and A. O. Hero, "Improving convergence of divergence functional ensemble estimators," in *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 1133–1137.

F. J. O. Morales and D. Roggen, "Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations," in *Proceedings of the International Symposium on Wearable Computers (ISWC)*. ACM, 2016, pp. 92–99.

G. R. Moreno, M. Niranjan, and A. Prugel-Bennett, "Saliency map on CNNS for protein secondary structure prediction," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 1249–1253.

K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

B. Neyshabur, H. Sedghi, and C. Zhang, "What is being transferred in transfer learning?" in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33. Curran Associates, Inc., 2020, pp. 512–523.

B. Nguyen, Y. Coelho, T. Bastos, and S. Krishnan, "Trends in human activity recognition with focus on machine learning and power requirements," *Machine Learning with Applications*, pp. 100 072–100 090, 2021.

A. Nøkland and L. H. Eidnes, "Training neural networks with local error signals," in *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 4839–4850.

M. Noshad, Y. Zeng, and A. O. Hero, "Scalable mutual information estimation using dependence graphs," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 2962–2966.

F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, pp. 1–25, 2016.

F. J. Ordóñez, G. Englebienne, P. De Toledo, T. Van Kasteren, A. Sanchis, and B. Kröse, "In-home activity recognition: Bayesian inference for hidden Markov models," *IEEE Pervasive Computing*, vol. 13, no. 3, pp. 67–75, 2014.

A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 23–50, 1998.

S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

S. J. Pan, J. T. Kwok, Q. Yang *et al.*, "Transfer learning via dimensionality reduction." in *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference*, vol. 8, 2008, pp. 677–682.

S.-m. Park, D. D. Won, B. J. Lee, D. Escobedo, A. Esteva, A. Aalipour, T. J. Ge, J. H. Kim, S. Suh, E. H. Choi *et al.*, "A mountable toilet system for personalized health monitoring via the analysis of excreta," *Nature Biomedical Engineering*, vol. 4, no. 6, pp. 624–635, 2020.

M. Peng, Y. Li, B. Wamsley, Y. Wei, and K. Roeder, "Integration and transfer learning of single-cell transcriptomes via cFIT," *Proceedings of the National Academy of Sciences*, vol. 118, no. 10, pp. 1–8, 2021.

G. Perin, I. Buhan, and S. Picek, "Learning when to stop: a mutual information approach to fight overfitting in profiled side-channel analysis," *IACR Cryptol. ePrint Arch.*, vol. 2020, pp. 58–80, 2020.

S. Petropoulos, D. Edsgärd, B. Reinius, Q. Deng, S. P. Panula, S. Codeluppi, A. P. Reyes, S. Linnarsson, R. Sandberg, and F. Lanner, "Single-cell RNA-seq reveals lineage and X chromosome dynamics in human preimplantation embryos," *Cell*, vol. 165, no. 4, pp. 1012–1026, 2016.

S. Picelli, Å. K. Björklund, O. R. Faridani, S. Sagasser, G. Winberg, and R. Sandberg, "Smart-seq2 for sensitive full-length transcriptome profiling in single cells," *Nature Methods*, vol. 10, no. 11, pp. 1096–1098, 2013.

S. Picelli, O. R. Faridani, Å. K. Björklund, G. Winberg, S. Sagasser, and R. Sandberg, "Full-length RNA-Seq from single cells using Smart-seq2," *Nature Protocols*, vol. 9, no. 1, pp. 171–181, 2014.

J. Platt, "A resource-allocating network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, 1991.

L. Y. Pratt, "Discriminability-based transfer between neural networks," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. Morgan Kaufmann Publishers Inc., 1992, pp. 204–211.

M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein, "SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2017, pp. 6076–6085.

M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, "Transfusion: Understanding transfer learning for medical imaging," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2019, pp. 3347–3357.

R. Raina, A. Battle, H. Lee, B. Packer, and T. T Nguyen, "Self-taught learning: transfer learning from unlabeled data," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2007, pp. 759–766.

H. Ravishankar, P. Sudhakar, R. Venkataramani, S. Thiruvenkadam, P. Annangi, N. Babu, and V. Vaidya, "Understanding the mechanisms of deep transfer learning for medical images," in *Deep Learning and Data Labeling for Medical Applications*. Springer, 2016, pp. 188–196.

J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

Y. Ro and J. Y. Choi, "Layer-wise pruning and auto-tuning of layer-wise learning rates in fine-tuning of deep networks," in *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Conference*, 2021, pp. 1–10.

S. Roberts and L. Tarassenko, "A probabilistic resource allocating network for novelty detection," *Neural Computation*, vol. 6, no. 2, pp. 270–284, 1994.

D. Roggen and P. Zappi, "Human activity/context recognition datasets," 2015. [Online]. Available: http://har-dataset.org/doku.php?id=wiki:dataset

D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha *et al.*, "Collecting complex activity datasets in highly rich networked sensor environments," in *Seventh International Conference on Networked Sensing Systems (INSS)*. IEEE, 2010, pp. 233–240.

D. Roggen, L. P. Cuspinera, G. Pombo, F. Ali, and L.-V. Nguyen-Dinh, "Limited-memory warping lcss for real-time low-power pattern recognition in wireless nodes," in *European Conference on Wireless Sensor Networks*. Springer, 2015, pp. 151–167.

C. A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert Systems with Applications*, vol. 59, pp. 235–244, 2016.

N. Rosenthal and S. Brown, "The mouse ascending: perspectives for human-disease models," *Nature Cell Biology*, vol. 9, no. 9, pp. 993–999, 2007.

D. E. Rumelhart, G. E. Hinton, J. L. McClelland *et al.*, "A general framework for parallel distributed processing," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, no. 26, pp. 45–76, 1986.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

A. Sarkar and M. Stephens, "Separating measurement and expression models clarifies confusion in single-cell rna sequencing analysis," *Nature Genetics*, vol. 53, no. 6, pp. 770–777, 2021.

A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://openreview.net/forum?id=ry_WPG-A-

M. Schiemer and J. Ye, "Revisiting the information plane," 2019. [Online]. Available: https://openreview.net/forum?id=Hyljn1SFwr

T. Schlegl, J. Ofner, and G. Langs, "Unsupervised pre-training across image domains improves lung tissue classification," in *International Conference on Medical Image Computing & Computer Assisted Intervention (MICCAI) Workshop on Medical Computer Vision*. Springer, 2014, pp. 82–93.

R. Serfozo, *Basics of applied stochastic processes*. Springer Science & Business Media, 2009.

T. Serre, "Hierarchical models of the visual system." *Encyclopedia of Computational Neuroscience*, vol. 6, pp. 1–12, 2014.

T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2. IEEE, 2005, pp. 994–1000.

S. M. Shaffer, M. C. Dunagin, S. R. Torborg, E. A. Torre, B. Emert, C. Krepler, M. Beqiri, K. Sproesser, P. A. Brafford, M. Xiao *et al.*, "Rare cell variability and drug-induced reprogramming as a mode of cancer drug resistance," *Nature*, vol. 546, no. 7658, pp. 431–435, 2017.

O. Shamir, S. Sabato, and N. Tishby, "Learning and generalization with the information bottleneck," *Theoretical Computer Science*, vol. 411, no. 29-30, pp. 2696–2711, 2010.

O. Shetta and M. Niranjan, "Robust subspace methods for outlier detection in genomic data circumvents the curse of dimensionality," *Royal Society Open Science*, vol. 7, no. 2, pp. 1–14, 2020.

H.-C. Shin, L. Lu, L. Kim, A. Seff, J. Yao, and R. M. Summers, "Interleaved text/image deep mining on a very large-scale radiology database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1090–1099.

H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

R. Shwartz-Ziv and A. A. Alemi, "Information in infinite ensembles of infinitely-wide neural networks," in *Symposium on Advances in Approximate Bayesian Inference*. PMLR, 2020, pp. 1–17.

R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *arXiv preprint arXiv:1703.00810*, 2017.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *Proceedings of the Workshop at International Conference on Learning Representations (ICLR)*, 2014.

N. Slonim, "The information bottleneck: Theory and applications," Ph.D. dissertation, The Hebrew University, 2002.

E. Soares, P. Angelov, S. Biaso, M. H. Froes, and D. K. Abe, "SARS-CoV-2 CT-scan dataset: A large dataset of real patients CT scans for SARS-CoV-2 identification," *medRxiv*, 2020.

M. Soumillon, D. Cacchiarelli, S. Semrau, A. van Oudenaarden, and T. S. Mikkelsen, "Characterization of directed differentiation by high-throughput single-cell RNA-Seq," *BioRxiv*, pp. 1–13, 2014.

J. M. Steele, *The Cauchy-Schwarz master class: an introduction to the art of mathematical inequalities*. Cambridge University Press, 2004.

G. L. Stein-O'Brien, R. Arora, A. C. Culhane, A. V. Favorov, L. X. Garmire, C. S. Greene, L. A. Goff, Y. Li, A. Ngom, M. F. Ochs *et al.*, "Enter the matrix: factorization uncovers knowledge from omics," *Trends in Genetics*, vol. 34, no. 10, pp. 790–805, 2018.

G. L. Stein-O'Brien, B. S. Clark, T. Sherman, C. Zibetti, Q. Hu, R. Sealfon, S. Liu, J. Qian, C. Colantuoni, S. Blackshaw *et al.*, "Decomposing cell identity for transfer learning across cellular measurements, platforms, tissues, and species," *Cell Systems*, vol. 8, no. 5, pp. 395–411, 2019.

T. Stiefmeier, D. Roggen, and G. Troster, "Fusion of string-matched templates for continuous activity recognition," in *11th IEEE International Symposium on Wearable Computers (ISWC)*. IEEE, 2007, pp. 41–44.

P. S. Stumpf, X. Du, H. Imanishi, Y. Kunisaki, Y. Semba, T. Noble, R. C. Smith, M. Rose-Zerili, J. J. West, R. O. Oreffo *et al.*, "Transfer learning efficiently maps bone marrow cell types from mouse to human using single-cell RNA sequencing," *Communications biology*, vol. 3, no. 1, pp. 1–11, 2020.

S. Sun, J. Zhu, Y. Ma, and X. Zhou, "Accuracy, robustness and scalability of dimensionality reduction methods for single-cell RNA-seq analysis," *Genome Biology*, vol. 20, no. 1, pp. 1–21, 2019.

E. W. Swokowski, *Calculus with analytic geometry.* Taylor & Francis, 1979.

T. T Nguyen and J. Choi, "Markov information bottleneck to improve information flow in stochastic neural networks," *Entropy*, vol. 21, no. 10, pp. 976–988, 2019.

N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

M. Talo, U. B. Baloglu, Ö. Yıldırım, and U. R. Acharya, "Application of deep transfer learning for automated brain abnormality classification using mr images," *Cognitive Systems Research*, vol. 54, pp. 176–188, 2019.

F. Tang, C. Barbacioru, Y. Wang, E. Nordman, C. Lee, N. Xu, X. Wang, J. Bodeau, B. B. Tuch, A. Siddiqui *et al.*, "mrna-seq whole-transcriptome analysis of a single cell," *Nature Methods*, vol. 6, no. 5, pp. 377–382, 2009.

T. Tieleman and G. Hinton, "Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, pp. 26–31, 2012.

N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Information Theory Workshop (ITW).* IEEE, 2015, pp. 1–5.

N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," in *Annual Allerton Conference on Communications, Control and Computing*, 1999, pp. 368–377.

L. Q. Trinh, "Greedy layerwise training of convolutional neural networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2019.

B. Van Ginneken, A. A. Setio, C. Jacobs, and F. Ciompi, "Off-the-shelf convolutional neural network features for pulmonary nodule detection in computed tomography scans," in *IEEE 12th International Symposium on Biomedical Imaging (ISBI).* IEEE, 2015, pp. 286–289.

A. Wagner, A. Regev, and N. Yosef, "Revealing the vectors of cellular identity with single-cell genomics," *Nature Biotechnology*, vol. 34, no. 11, pp. 1145–1160, 2016.

L. Waltman and N. J. Van Eck, "A smart local moving algorithm for large-scale modularity-based community detection," *The European Physical Journal B*, vol. 86, no. 11, pp. 471–485, 2013.

J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.

J. Wang, D. Agarwal, M. Huang, G. Hu, Z. Zhou, C. Ye, and N. R. Zhang, "Data denoising with transfer learning in single-cell transcriptomics," *Nature Methods*, vol. 16, no. 9, pp. 875–878, 2019.

T. Wang, T. S. Johnson, W. Shao, Z. Lu, B. R. Helm, J. Zhang, and K. Huang, "Bermuda: a novel deep transfer learning method for single-cell RNA sequencing batch correction reveals hidden high-resolution cellular subtypes," *Genome Biology*, vol. 20, no. 1, pp. 1–15, 2019.

Y. Wang, Z. Ni, S. Song, L. Yang, and G. Huang, "Revisiting locally supervised learning: an alternative to end-to-end training," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://openreview.net/forum?id=fAbkE6ant2

J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236–244, 1963.

P. M. Warnecke, C. Stirzaker, J. R. Melki, D. S. Millar, C. L. Paul, and S. J. Clark, "Detection and measurement of PCR bias in quantitative methylation analysis of bisulphite-treated DNA," *Nucleic Acids Research*, vol. 25, no. 21, pp. 4422–4426, 1997.

K. Wickstrøm, S. Løkse, M. Kampffmeyer, S. Yu, J. Principe, and R. Jenssen, "Information plane analysis of deep neural networks via matrix–based Renyi's entropy and tensor kernels," 2020. [Online]. Available: https://openreview.net/forum?id=B1l0wp4tvr

M. J. Willemink, W. A. Koszek, C. Hardell, J. Wu, D. Fleischmann, H. Harvey, L. R. Folio, R. M. Summers, D. L. Rubin, and M. P. Lungren, "Preparing medical imaging data for machine learning," *Radiology*, vol. 295, no. 1, pp. 4–15, 2020.

G. I. Winata, S. Cahyawijaya, Z. Liu, Z. Lin, A. Madotto, P. Xu, and P. Fung, "Learning fast adaptation on cross-accented speech recognition," *arXiv preprint arXiv:2003.01901*, 2020.

C. W. Wu, "On Rayleigh-Ritz ratios of a generalized Laplacian matrix of directed graphs," *Linear Algebra and Its Applications*, vol. 402, pp. 207–227, 2005.

R. Xie, L. Liu, J. Liu, and C. S. Qiu, "Pathological myopic image analysis with transfer learning," in *International Conference on Medical Imaging with Deep Learning–Extended Abstract Track*, 2019.

D. Xu and J. C. Principe, "Training MLPs layer-by-layer with the information potential," in *International Joint Conference on Neural Networks (IJCNN).*, vol. 3. IEEE, 1999, pp. 1716–1720.

J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015, pp. 3995–4001.

J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2014, pp. 3320–3328.

S. Yu, K. Wickstrøm, R. Jenssen, and J. C. Principe, "Understanding convolutional neural networks with information theory: An initial exploration," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 435–442, 2020.

M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional neural networks for human activity recognition using mobile sensors," in *6th International Conference on Mobile Computing, Applications and Services*. IEEE, 2014, pp. 197–205.

F. Zhang, *The Schur complement and its applications*. Springer Science & Business Media, 2006, vol. 4.

Y. Zhang, J. Pulliainen, S. Koponen, and M. Hallikainen, "Application of an empirical neural network to surface water quality estimation in the gulf of finland using combined optical data and microwave data," *Remote Sensing of Environment*, vol. 81, no. 2-3, pp. 327–336, 2002.

J. Zhao, Y. Zhang, X. He, and P. Xie, "COVID-CT-Dataset: a CT scan dataset about COVID-19," *arXiv preprint arXiv:2003.13865*, 2020.

X. Zhu, T. Ching, X. Pan, S. M. Weissman, and L. Garmire, "Detecting heterogeneity in single-cell RNA-Seq data by non-negative matrix factorization," *PeerJ*, vol. 5, pp. 1–20, 2017.