Otto-von-Guericke-University Magdeburg

Faculty of Computer Science

Department of Databases and Software Engineering

Aethra: Towards whitebox understanding of deep reinforcement learning for computer systems applications

Master Thesis

Author:

Anh Trang Le

Examiner and Supervisor:

Prof. Dr. rer. nat. habil. Gunter Saake

2nd Examiner:

Prof. Dr. Thomas Leich

Supervisor:

MSc. Gabriel Campero Durand

Magdeburg, 11.09.2020

Anh Trang Le
Aethra: Towards whitebox understanding of deep reinforcement learning for computer systems applications
Master Thesis, Otto-von-Guericke-University Magdeburg, 2020.

Abstract

With the emerging trend of employing machine learning techniques in handling various complex issues in practice, deep reinforcement learning has gained a specific success in solving multiple tasks through their ability to learn from experience, especially in the field of computer systems.

Along with this scenario, the need to understand as well as be able to explain the predictions or decisions made by these machine learning models has increased considerably since the last decades. There is an abundance of research in the past and at the present time that point out the significant role of interpretability in machine learning and also introduce numerous methods being used in obtaining transparency in the inner working of models. However, while the majority of researchers put more focus on revealing the hidden sides in classification and clustering learning models, there is almost no such progress for deep reinforcement learning. Most of the latest publications emphasize only the overall performance and averaged score of the agent at specific tasks, without providing insights on the internal representations of the models and how they impact the performance.

In our thesis, we make a first move towards white-box understanding for deep reinforcement learning models by zooming-in the data structures being learned in the last layers of the neural network architectures, using both clustering and visualization. We then evaluate the correlation between the clusters formed in these layers and the performance of the model on test scenarios regarding both supervised learning and reinforcement learning tasks, considering different correlation metrics as well as statistical confidence tests.

Overall we find that the aspects of supervised vs. reinforcement learning, learning success, saturation on the learned accuracy on the training data, differences between test and training data and the use of regularization all play roles in the structures learned at the different layers, and the correlation of these structures to the test accuracy. We identify cases where the clustering goodness of the structures learned on the last layer correlate better than the train accuracy with the test accuracy; we also detect cases where such structures do not correlate strongly. We find that regularization and reinforcement learning with no distributional shift behave similarly, with high correlation between the outermost layers' structures and the test accuracy. We also point out promising research directions to extend our work, especially, the use of algorithms to automatically detect prototypes and criticisms in the structures learned.

Acknowledgements

First of all, I would like to express my great and sincere thanks to Gabriel Campero Durand for his great supervision and all the tremendous supports regarding both academic and mental supports that he has given me throughout this work.

Furthermore, I would like to thank Prof. Dr. rer. nat. habil. Gunter Saake for not only granting me the opportunity to complete my master thesis in DBSE group but also his support for me on the early stage of my academic career. And many thanks to both Prof. Dr. rer. nat. habil. Gunter Saake and Prof. Dr. Thomas Leich for their agreements on being examiners for my thesis as well as providing their valuable feedback for my work.

Finally, I thank Christoph Molnar for his great open source book, which inspired me a lot during my whole time working on my thesis.

Statement of Authorship

Contents

List of Figures

1 Introduction

Our thesis starts with Section 1.1 that presents an overview of the significant role of deep reinforcement learning in handling real world applications and the urge for gaining transparency in these models, from which we draw our inspiration. In Section 1.2, we highlight the major contributions of our thesis, and close this chapter with the overall thesis structure Section 1.3.

1.1 Motivation

Reinforcement learning (RL) is a well-known type of Machine Learning (ML) technique that is capable of efficiently handling various sequential decision tasks, in different domains including: healthcare [KCB+18], robotics [RGHL09], recommendation systems [IJW+19] and many more. In general, it concerns how an artificial agent learns from experiences gathered while interacting with its environment, and tries to reach the goal of maximizing cumulative rewards.

In recent years, the combination of RL and deep learning (DL) has been proved to be successfully in tackling a variety of complex real-world problems, in which high-dimensional state space used to be a limitation of traditional RL. This new combination has originated the field of deep reinforcement learning (DRL), where RL agents with the help of a neural network can overcome the "curse of dimensionality", showing the ability to convert huge, continuous and complicated raw input data into meaningful representations of the environment, which are then utilized by the agent, to accomplish its objectives.

A usually highlighted achievement in applying successfully deep RL is teaching computers to play classic Atari 2600 games, as done by to Mnih et al. [MKS+15a]. The authors proposed a deep Q-network agent, which can play at a similar level of a professional human player after testing through 49 different games. Or in 2019, the Dota 2 world champion has been beaten by an agent trained by the OpenAI team using a large-scale DRL system named OpenAI Five [OBB+19].

In the domain of computer systems, the applicability of DRL is highly promising for several reasons. Firstly, computer systems are required to continuously offer better performance, so the approaches help enhancing computer system capabilities can be competitively advantageous. Additionally, computer systems often have to solve sequential decision making tasks, based on data or context, which is usually done heuristically. However, heuristics can neither guarantee the optima nor offer good generalization. They can also be hard to evolve. DRL, in contrast, seeks to generalize, evolve and is greatly competitive so that the optima is usually found. Furthermore, DRL makes sense since a large amount of experiences can be quickly gathered in computer systems in order to form "training data". Those are the main reasons why DRL has been gaining its popularity speedily over the last few years. Specifically, in the area of database management system, various optimization tasks have been well handled such as: configuration tuning (Zhang et al., 2019 [ZLR+19], Li et al., 2019 [LZLG19], query optimization (Krishnan et al., 2018 [KYG+18], Marcus et al., 2018 [MP18], Tzoumas et al. [TSJ08]), and others.

Along with the wider use of ML in multiple real-world scenarios, the necessity of having interpretability on how solutions or decisions are reached using learning models is also becoming increasingly demanding. The inadequacy of transparency, in general, makes a model less reliable than existing solutions, as well as it decreases the trust of users when the motives for the predictions cannot be explained. Especially in the domains of healthcare system, autonomous driving (e.g., self-driving car), auto-trading, where wrong decisions may cause huge financial losses or dangers to human life, a model that cannot be reasonably explained is usually hardly acceptable. Furthermore, interpretability can be a crucial aspect to help model building, the idea being that if you understand the reasons why a model does not perform well on some data instances, you can include improvements to overcome this.

There has been some progress on explainable ML since the past several years. Besides using interpretable models such as linear regression or decision trees, a variety of model-agnostic methods are proposed to offer users more flexibility in explaining also non-interpretable models: Partial Dependence Plot (PDP) [Fri00], Global Surrogate [Mol19a], Local Surrogate (LIME) [RSG16c], etc. The general idea behind these methods is to understand predictions provided by a model while seeing this model as a black-box [RSG16a]. For neural networks, feature visualization through activation maximization is considered to be a specific technique developed for describing layers. This is applied by Olah et at. [OMS17a] in interpreting Convolutional Neural Network (CNN) trained on an image dataset. At the same time, Bau Zhou et al. also proposed the network dissection algorithm which uncover the concepts hidden among CNN layers by looking at units inside the network[BZK+17b].

In spite of these progresses towards interpretability in ML, there is almost no such progress for DRL. Almost all state-of-the-art publications puts emphasis on the overall performance at a task, but not the cases where it might fail nor the impact of observation features on the policy of agents. In the research of many authors [MKS+13], [HS15], [WSH+15], [SQAS15], [HMvH+17],[vHGS15], the averaged scores of agents achieved from running experiments on numerous Atari games are reported, in order to evaluate the experience-gathering behavior of their deep Q-learning agents compared to others. Together with overall score, the learning curve showing relation between training time-steps and target return values is also usually used to report and analyze the performance of policy-based agents [MBM+16], [SLM+15], [SWD+17], [HTS+17]. These practices, naturally miss to provide insights on the internal representations of the models. The paper that originally proposed DQN[MKS+15b] in fact uses t-SNE visualizations to illustrate the process, but unfortunately this trend has not proceeded nor has the approach of studying the structure being learned be approached systematically.

Therefore, a research on understanding the behaviors of DRL agents does matter, especially when the model is more widely adopted in real-world systems and requires transparency for a better maintenance strategy. The next chapters include the scope of our thesis as well as the major contributions that we make throughout this work.

1.2 Main Contributions

The main contributions of our work are as follows:

1. We review the current scenarios of interpretable ML as well as multiple methods that are used to obtain transparency in different types of ML models. We consider model-specific approaches in interpreting neural networks, and prototypes and criticisms analysis - a model-agnostic method, that can be used for any ML model. We also point out some research gaps and further applications of prototypes and criticisms that are worth to study in the future, in the context of reinforcement learning.
2. We present our design for the prototypical implementation to evaluate to which extent can the structure learned by a deep learning model in its last layers (regarding the training data), serve as indicators of the potential accuracy and generalization powers of the model. And also, the exploration of this structure when integrating both deep learning and reinforcement learning.
3. We carry out a collection of experiments that cover the visual analysis of the structure being learned in the last layers of supervised learning task, as well as clustering performance, followed by an evaluation of the correlation between clustering goodness measures and model testing accuracy. We also consider the influence of regularization on the structures and correlations being observed. Finally, we include a study related to reinforcement learning.
4. Based on these experiments results we point out the most important observations regarding: the data structures learnt in the last layers, relations between cluster goodness and model testing accuracy of deep learning tasks, with or without regularization and later in deep reinforcement learning.
5. We establish some areas for future research that are promising to provide a more transparent understanding for deep reinforcement learning applications.

1.3 Thesis Structure

The forthcoming parts of the thesis are structured as follows:

- In Chapter 2, we provide our background, in which we review a collection of research work relevant to our thesis topic. We additionally discuss the approaches included in those papers and explain the motivation for our approach by identifying research gaps that we plan to fill in the future.
- Chapter 3 formulates the objectives of our work into research questions. Besides, we also depict the main procedure and components that we base our work on to solve our research questions.

- In Chapter 4, we share the information about all materials, software, libraries and tools required to carry out our experiments.
- Chapter 5 is devoted to discuss the results obtained from our experiments.
- Finally, Chapter 6 wraps up our thesis with a conclusion and our plans for our continuing research work regarding interpretable deep reinforcement learning for computer system applications.

2 Background

In this chapter, we introduce the principal knowledge required to understand our work, and we highlight approaches from the literature that are relevant to our methodology, as well as experimental setup and results. The chapter is organized as follows:

- In Section 2.1, we briefly discuss the role and concepts of interpretability in the context of machine learning. In addition, we introduce an overview of historical and state-of-the-art techniques for interpreting machine learning models.
- Section 2.2 is dedicated to discussing relevant methods that are tailor-made to acquire the transparency in neural networks.
- In Section 2.3, we provide details about model-agnostic methods that can be used to interpret any machine learning models.
- Section 2.4 outlines some research gaps identified in current work, which we consider to suggest an approach worth studying for interpretable deep reinforcement learning.
- Finally, Section 2.5 is the summary of the main content represented in the whole chapter.

2.1 Introduction to Interpretability in Machine Learning

Machine Learning (ML) refers to a set of algorithms or techniques that a computer uses to learn specific tasks and improve its performance through experiences without human instructions. An informal definition considers ML as the task of solving a problem by gathering a dataset and building a statistical model over the dataset[Bur19]. Along with the increase of employing ML models in tackling various real-world problems as well as a visible future of automated ML, there is a growing demand in understanding and interpreting results that are automatically provided by ML models. In [DVK17], the authors define the meaning of interpretability in an ML context as the capability of representing the knowledge (and potential behavior) of the model to a human in a comprehensible manner. The term interpretability is equal to explainability under the viewpoint of Christoph Molnar in [Mol19b], or comprehensibility according to Michael Gleicher in [Gle16]. Transparency is another related concept that is frequently used in this topic, indicating the characteristic of the model being intelligible to humans at all steps of its functioning.

Except for some scenarios where interpretability is unnecessary such as: incorrect results/predictions provided by models cause no harm, or the model is validated to be well-behaved over time such that in practice we are not doubtful about its outcomes at all, there are various reasons that make explainability very important on most of the cases. First, it matters to satisfy human natural inquisitiveness and our intrinsic desire for knowledge [Loe94][Pos91]. When trying to unlock learning, our instinct is to seek answers for the typical questions of what, when, how, why, depending on the level of transparency we want to gain on the concerning problems. The more we ask, the clearer our knowledge about things. And a clearer understanding on the inner working of a model enables us to better detect model misbehaviors that result in unanticipated outcomes. For example, some ML systems can encounter an unfair bias that relates to racism or human discrimination [Bro18] [HBC16] [BG18], such that they tend to gradually develop a prejudice or make wrong assumptions about a group of people regarding their genders, cultures, etc. , by not considering enough information in the training data. In these circumstances, being able to explain how and why model makes its predictions/decisions is essential for human in investigating as well as avoiding model bias [GF17]. Furthermore, interpretability is also indispensable if a model needs to gain an absolute trust from relevant stakeholders, where any inaccuracy in its predictions and recommendations may cause a huge loss in financial aspect, or severe impacts on human health status or even human lives such as in autonomous transportation or healthcare system [Vel19][SM19][Mol19a]. In the case of computer systems, it is of uttermost importance to develop model interpretability, to maintain the competitive high quality of computer sytems when they are extended with ML.

The topic of interpreting a computer system was earlier introduced in rule-based expert systems. In 1984, William van Melle and his team proposed an approach, such that the knowledge collected from specialists in different domains is encoded in a ruleset and stored in the system, which are then retrieved and interpreted to non-expert customers, so that they are provided relevant information in an understandable manner [vMSB]. In 1988, Moore and Swartout published a survey discussing two other rule-based approaches towards explainability in knowledge base systems, and simultaneously, came up with their own solution to improve comprehensibility in generating explanations to clients, considering system-user dialogical interaction [MS88]. Since the last decade, corresponding to the booming trend in utilizing ML for solving complex real-world tasks, the number of publications regarding interpretability techniques for ML tools increases considerably. Numerous scientists have presented their surveys [GBY+18][BC17][CPC19], articles [MSK+19] [DVK17], or books [Mol19b], sharing us overviews, justifications and applications of multiple techniques that support humans in comprehending results generated by ML models. Among them, we are most interested in the book written by Christoph Molnar [Mol19b]. We identify this book to be the current the de-facto textbook on the domain. It covers the major concepts and describes in detail various state-of-the-art approaches in interpretable ML. It presents the work in a well-structured, friendly style with many interesting examples that make contents easier for readers to understand.

In Molnar's book, the highlighted parts we deemed relevant for our review include chapter 2, 4, 5, 6 and 7. Chapter 2, in general, introduces the definition, significance, scope of interpretability and different criteria to categorize current interpreting methodologies. The criteria can be associated with method characteristics, whether they are model-specific or model-agnostic. While model-specific is restricted to a particular model class (e.g. neural networks), model-agnostic can be applied to achieve comprehensibility for any ML models. This criteria can also be the level of transparency provided by methods, if they are global that explains the overall model or local which interprets details of each individual prediction.

Besides Molnar's book, some recent researches also discuss similar terms and techniques in interpretable ML, but additionally extend the research direction of explanability towards a bigger target of the whole field of artificial intelligence (AI), so-called explainable AI or XAI. Some good sources for references are the book of Samek et al. [SMV+19], or the survey article of Adadi and Berrada [AB18].

2.1.1 An Overview of White-Box Models

Chapter 4 describes the models that are naturally explainable in a modular level, such as: linear regression, logistic regression, decision tree, decision rules, etc. They are so-called white-box models with intrinsically interpretable components, for example, weights in linear models or splits in decision trees. To be more detailed, in a *linear regression model*, where the output is measured by summing up the weighted input variables [Rou13], explainability can be attained by applying certain guidelines corresponding to the varied types of feature input. For instance, the interpreting template for numerical feature in a linear equation $y = a_0x_0 + a_1x_1 + ... + b$ is: adding/reducing one unit of x_0 increases/decreases the predicted variable y by a_0, given constant values among remaining features. Or another interpretation can be made related to feature importance, such that the feature with higher weight is more important than the ones with lower weights. Linear models are simple and can help to clarify how predictions are provided based on weighted sum of component features, however, their effectiveness is usually not guaranteed in solving complex tasks where non-linear relationship exists between in- and output variables. *Decision trees* (DT) are another white-box model that can overcome this issue and additionally manage to learn the interaction among features themselves. DT are a supervised ML technique that splits the feed-in data into smaller sets until reaching the leaf node and each data point is assigned to a certain subset. Target variables of a leaf node are estimated by considering all values that are included in this node. A tree can be used to solve a classification task in case of learning discrete values or a regression task if the output is a real number. A DT can be explained following an if-then rule: If feature x meets the cutoff condition c_1 (and...), then the output of the leaf node l is computed by averaging y values of all data points belonging to l. The importance of a feature can be decided by information gain [Qui86], Gini impurity or variance reduction [BFOS17]. The best feature is the one that creates sub-nodes with highest homogeneity or lowest entropy.

2.1.2 An Overview of Model-Agnostic Approaches

Chapter 5 takes a deep dive into model-agnostic approaches, which widen the application range compared to model-specific, such that they can be employed to interpret any ML models. According to the authors in [RSG16b], the criteria expected from a model-agnostic system is to offer flexibility in three aspects: model, explanation and representation. In particular, they are not limited to a certain model type, a specific form of explanation, and allow using various representations of features. There are 10 different methods that are discussed by Molnar in this chapter: Local interpretable model-agnostic explanations (LIME) [RSG16c], Partial Dependence Plot (PDP) [Fri00], Permutation Feature Importance[FRD18], etc. We will go through two examples of them, one uses graphs with feature importance named

Individual Conditional Expectation, the other calculates *Shapley values* to quantify the contribution of the features to model decision. *Individual Conditional Expectation (ICE)* offers an insight about how changes in the features influence the model prediction through a line graph [GKBP13], where each of every instance is plotted as a line. Considering the same example given in the book with cervical cancer dataset [FCF17]. It is depicted by ICE in figure 2.2 and clearly showed that there is a noticeably higher possibility that women can have cervical cancer when they are around 50 years old. Abnormally, there exists some patients who have higher risk of suffering from this disease even when they are still young (20-30 years old). For them, the estimated cancer probability is not much influenced by the age. ICE is simple to understand, but is limited to one feature to be visualized only. Representing more than two features in the same graph requires multiple layers, which causes confusion in comprehending the plot.

Shapley values were originally proposed by Shapley Lloyd in 1953. They evaluates how significantly each feature value of a data instance contributes towards a model forecast compared to the average forecast. Specifically, for a data item i, a Shapley value of a feature value x of i is computed as the mean contribution of x in different combinations with the other features of i [Sha53]. The solid theory behind this method helps fostering human trust. However, its computational cost is a huge drawback where combinations of features add up the calculating time exponentially corresponding to the features number. One extension, called SHAPS, help with the computation cost by enabling some approximations.

Moving forward to chapter 6, Molnar discusses other techniques, which retrieve a certain sample of instances from the input to interpret distribution among the entire data or the whole model behavior, so-called *Example-based explanation (EBE)*. Almost all methods belonging to this group are model-agnostic since they can be employed to explain any ML model, except for the dissimilarity that EBE considers only a subset of data, instead of generating features summaries like some model-agnostic approaches [FRD18][Fri00]. EBE is more appropriate with images or texts rather than tabular data. Because the feature values of each instance in image/textual data have more context and can be humanly viewable/readable, while those in table data could have a less easy to understand structure. There are 4 techniques that are introduced in this chapter: counterfactual explanations, adversarial examples, prototypes and criticisms and influential instances.

Counterfactual explanations explain the causal association between in- and output value of the model: if there was a change in the feature value of an instance x, the model prediction y would have changed accordingly, such that the value of y would be more or less similar to the desired outcome. Counterfactual explanations are transparent to humans, without any assumptions. Besides, they works without opening the box of internal data and model structure in case data privacy is required. However, in complicated scenarios, where a single instance may have a variety of counterfactual explanations, the interpretation will be less comprehensible. *Adversarial examples* refer to the data instances that are intentionally created to trick the model. They are opposite to counterfactual examples in the fact that their aim is to fool a model instead of interpreting it. This method is

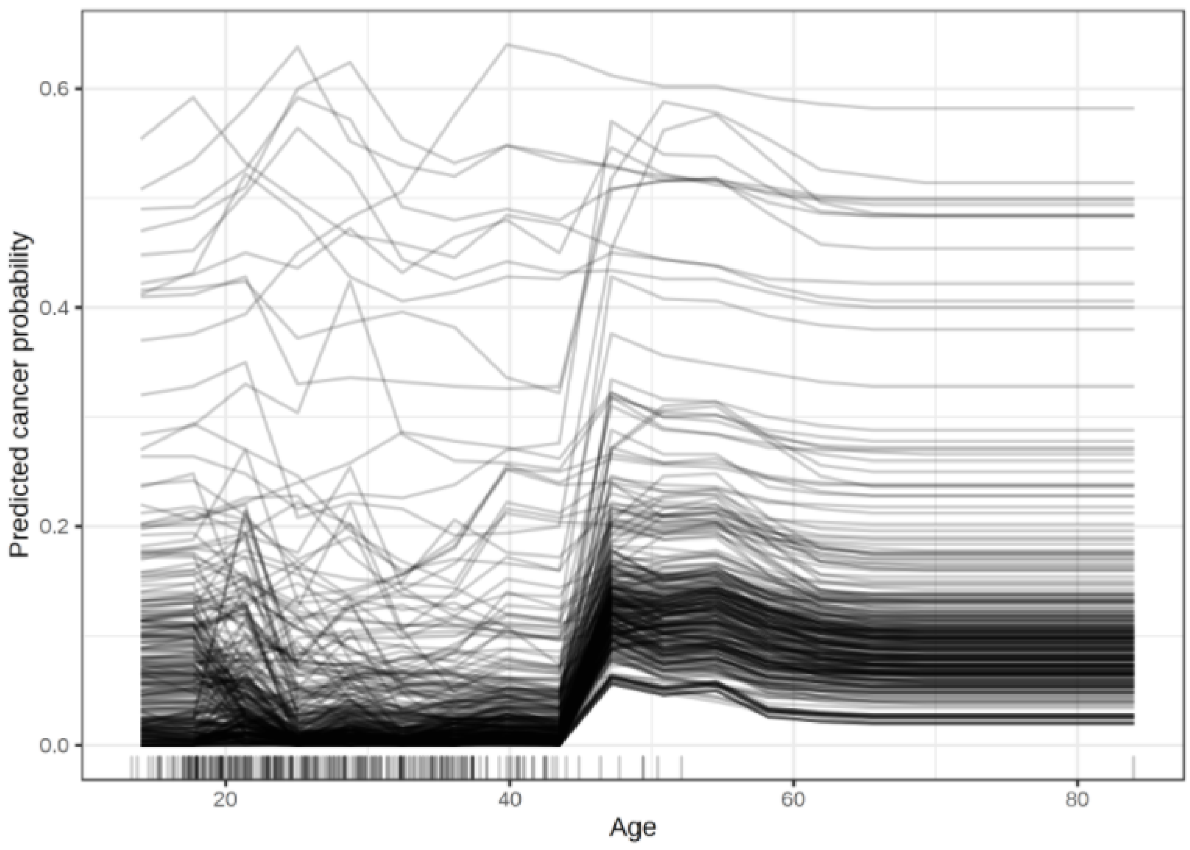


Figure 2.1: ICE illustrating the likelihood that women have cervical cancer according to their age. Source: [Mol19a] https://christophm.github.io/interpretable-ml-book/ice.html

important because of three important principles in preventing external attacks to our system, coined by Biggio and his team in [BR18]: (1) being aware of possible adversaries of the model, (2) always taking the initiative in investigating model vulnerabilities and (3) figuring out solutions to tackle these problems.

In the next method, *prototypes and criticisms* are analyzed to obtain the transparency in a black-box model. In general, prototypes are the instances modelling the entire data, and criticisms are 'black sheep' in the data that cannot be reasonably described by prototypes. More information of this method will be provided in the section 2.3 of our thesis.

Discovering *influential instances* is another example-based method that is useful for debugging and explaining model behaviors. An instance is considered to be influential if the removal of it from training data makes a huge impact to model output. Theoretically, one can be identified by repeatedly training the model after taking out the instance from training set, then evaluating the change in model outcome. However, its computation cost when handling big training data is impractical. Designing an influence function is a better option, recommended by Koh. et al. [KL17], in which, the deletion of an instance is replaced by adding weight to this instance in the loss function. Nevertheless, this function cannot guarantee an absolute accuracy, according to Molnar, due to the quadratic approximation created around model parameters when demonstrating how the actual loss would be.

2.1.3 Model-Specific Approaches for Neural Networks

Continuing with the overview provided by Molnar, in Chapter 7 the author describes the interpretable methods that are specifically used to shed a light on the secret-box of neural networks. They are *feature visualization* and *network dissection*, which we dedicate the section 2.2 in our thesis to present more details about them.

In the next sections we discuss in detail one model-specific approach and one model-agnostic approach, since they serve to provide context for our work.

2.2 A Model-Specific Approach: Learned Features for Convolutional Neural Network

2.2.1 Convolutional Neural Network

A convolutional neural network (CNN) [LeC89] is a class of neural networks that is widely used to learn data with a structured *'grid-like topology'* (Goodfellow et al., 2016) [GBC16]. An image, which is composed by pixels arranged in 2-dimensional grids, is such data type.

CNN, similar to other neural networks, has one input, one output and one or more hidden layers. The input layer initializes data to the network and output layer brings out the results estimated by the model. Hidden layers in the most basic CNN typically include: convolutional and fully connected layer.

Convolutional layer employs a filter to traverse across the input and extract the most significant features out of it by performing the convolution operation on the receptive fields of input layer. The width and height of the filter are human-decided, but its depth must be equal to the number of input channels, for example, filter depth of 1 for gray-scale images and that of 3 for RGB images. A convolutional layer is usually implemented with an activation function that rectifies the summed activation of an input node to a certain activation value of its output node. The rectified linear activation function (ReLU) is a common activation function, which simply converts the negative input activation to the value of 0 in the output activation.

Fully connected layers act differently from convolutional ones, such that they take into consideration all connections of every neuron in the preceding layer.

Another type of hidden layer that is optional but also regularly used in CNN after convolutional layer is max pooling, which efficiently reduces the representation features of the input. We discuss more details about max pooling later in 4.2.2.

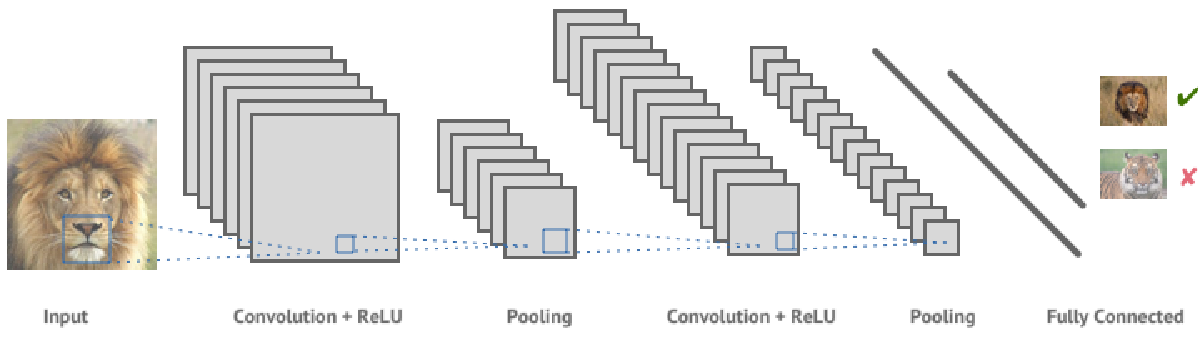


Figure 2.2: An example of a CNN architecture. Source: https://shafeentejani.github.io/2016-12-20/convolutional-neural-nets/

2.2.2 Feature Visualization

The method of feature visualization discussed in Molnar's book is originally introduced by Olah et al. in Distill[OMS17b]. It considers each unit of the neural network, which can be a single neuron, an image channel or the whole network layer. The main idea is to move the unit activation value between two extremes to visually highlight what kind of feature of the input that the model is looking at.

Despite the fact that choosing each neuron as unit costs a lot of time due to the huge amount of those included in a single network, generating feature visualization for individual neuron enables us to gain the most knowledge. For instance, Figure 2.3 shows completely different features the model GoogLeNet [SWY+15], which is trained on ImageNet [DDS+09], when optimizing a certain neuron activation in opposite directions. In particular, the left image that is generated with minimum unit activation displays some circular shapes, while that on the right side with maximum activation brings out shapes that look like Gothic buildings. In other words, this unit takes buildings as the goal, and circles as the things that are not buildings.

This approach is useful for interpreting the model at the local level of individual predictions. Using visualization to depict the features that are learned during the training process is helpful if the nontechnical explanation is needed. However, sometimes it is challenging to clarify some ambiguous visualization such as the left image in Figure 2.3.

2.2.3 Network Dissection

Bau et al. [BZK+17a] proposed their framework, named network dissection, to assess how well an unit (image channel) in CNN investigates real world concepts which are varied in abstraction degree such as: object, color, texture, etc. These are concepts that cannot be measured by feature visualization. They make an assumption that each unit in the neural network identifies concepts in training images in a disentangled fashion, or in other words, different channels learn different concepts. They prove their assumption by following three main steps:

- Step 1: Collect images with multiple concepts and label the data

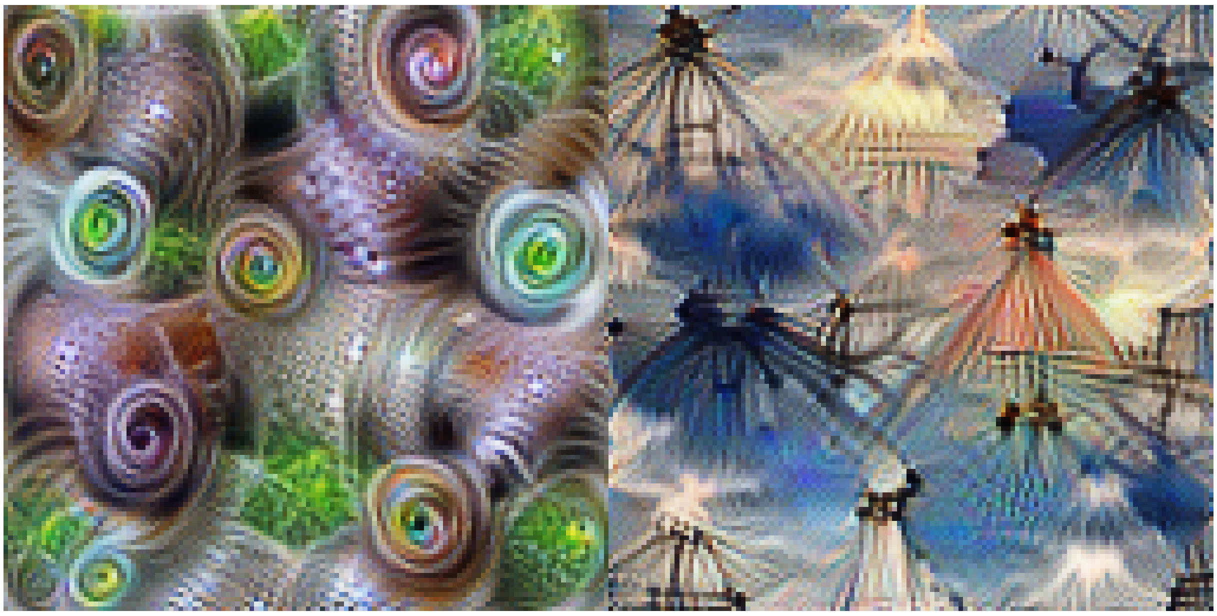


Figure 2.3: Visualizations of neuron 492 from layer mixed4a in GoogLeNet[SWY+15] model when its activation is moved towards negative (left) and positive (right) extremes. Source: https://colab.research.google.com/github/tensorflow/lucid/blob/master/notebooks/feature-visualization/negative_neurons.ipynb

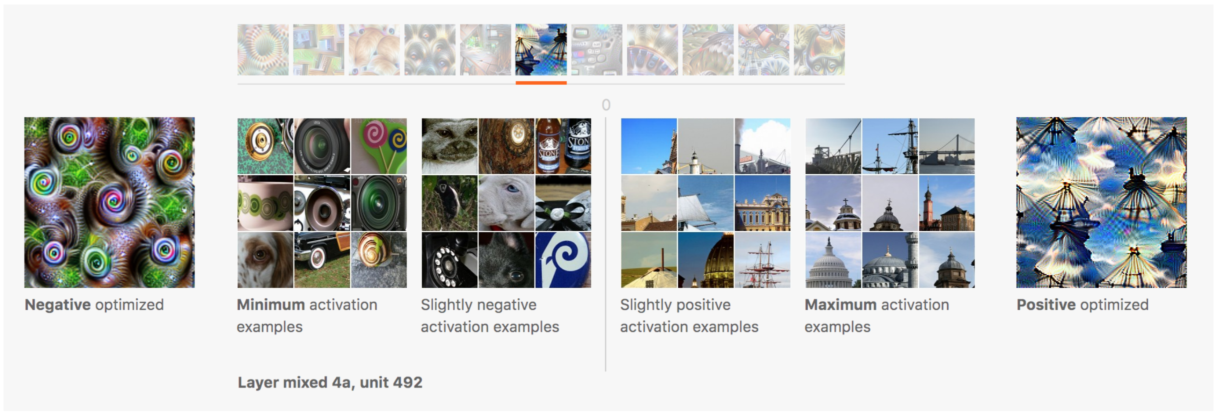

0
Negative optimized
Minimum activation examples
Slightly negative activation examples
Slightly positive activation examples
Maximum activation examples
Positive optimized
Layer mixed 4a, unit 492

Figure 2.4: Newly created images by optimizing activation and training images that optimize the activation. Source: https://colab.research.google.com/github/tensorflow/lucid/blob/master/notebooks/feature-visualization/negative_neurons.ipynb

- Step 2: Per each channel and image, create one mask that captures the highly activated fields based on a predefined threshold T of activation value.
- Step 3: Calculate the Intersection over Union (IoU) score between activation mask obtained from step 2 and labeled concepts from step 1. The score reflects how precise the concept is learnt by the unit.

The benefit of network dissection over feature visualization is providing a metric that can measure how well a network unit learns a particular input concept, but its disadvantage is bypassing the negative activations and only considering the positive ones. Furthermore,

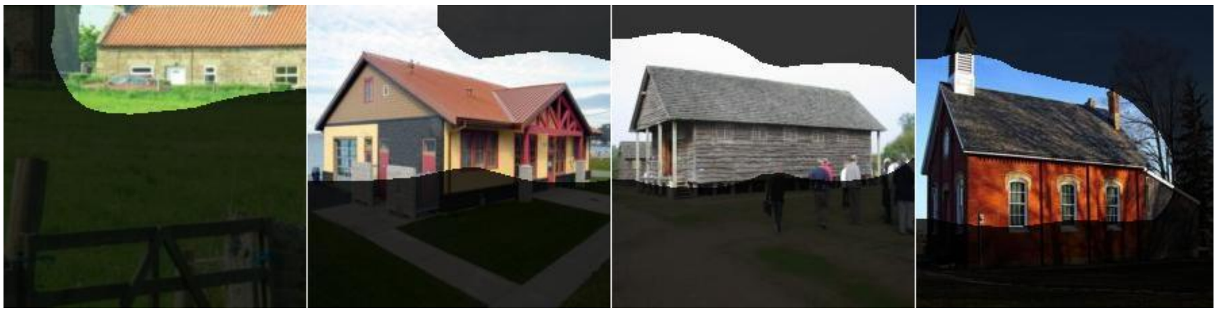


Figure 2.5: Unit 789 from layer inception_4e of GoogLeNet [SWY+15] trained on Broden dataset [BZK+17a] that identifies house, IoU score = 0.137. Source: http://netdissect.csail.mit.edu/

preparing the data and labeling every pixel of all images would be required, and this is extremely time-consuming.

2.3 A Model-Agnostic Approach: Prototypes and Criticisms Analysis

In this section, we discuss in depth the method of using prototypes and criticisms in interpreting a ML model, which belongs to the family of example-based explanations.

First of all, let us understand what are prototypes and criticisms. Prototypes refer to the exemplar instances in the input data, such that they behave similarly to the majority instances in the dataset. Criticisms, on the other hand, cannot be appropriately modeled by prototypes [Mol19a]. According to Been Kim et al. [KKK16], prototypes are not sufficient enough to represent the entire data unless the data distribution is flawless. However, it is almost impossible in practice to have a set of exemplars to ideally model the whole dataset. Furthermore, by frequently using regularization in ML models to prevent overfitting, we may encounter the issue of over-generalization. For this case criticisms would be of interest. Therefore, criticisms are not only significant in providing a deeper comprehension about the complex input space that cannot be well explained by prototypes solely, but their combination helps mitigating over-generalization. Kim et al. has shared their idea of using maximum mean discrepancy (MMD) to detect prototypes and criticisms, named *MMD-critic* [KKK16].

Overall the MMD-critic starts with defining the number of prototypes and criticisms, following by searching both types greedily, such that prototypes are located in high-density areas of data and the opposite for criticisms.

Diving deeper into the searching process, we will briefly explain some metrics and functions that are needed, including: kernel function, witness function, MMD, and MMD^2.

The kernel function [GBR+08] evaluates data densities by measuring similarities among data points. Base on the resulted data densities: MMD approximates discrepancy between two distributions, which is later used to determine if selected prototypes are distributed near data distribution; witness function reflects the discrepancy of two distributions at a particular data instance, which is needed to identify criticisms. These three above are ingredients to create MMD^2. A list of prototypes is collected by greedily adding data instances to the list, and choosing the instances that reduce MMD^2 [KKK16]. MMD^2 has the lowest value of 0 and the smaller values indicate closer distances between prototypes and data distribution. Figure 2.6 shows different prototypes distributions and their MMD^2 values. The set of prototypes in the bottom left image that best presents the given data gains lowest MMD^2.

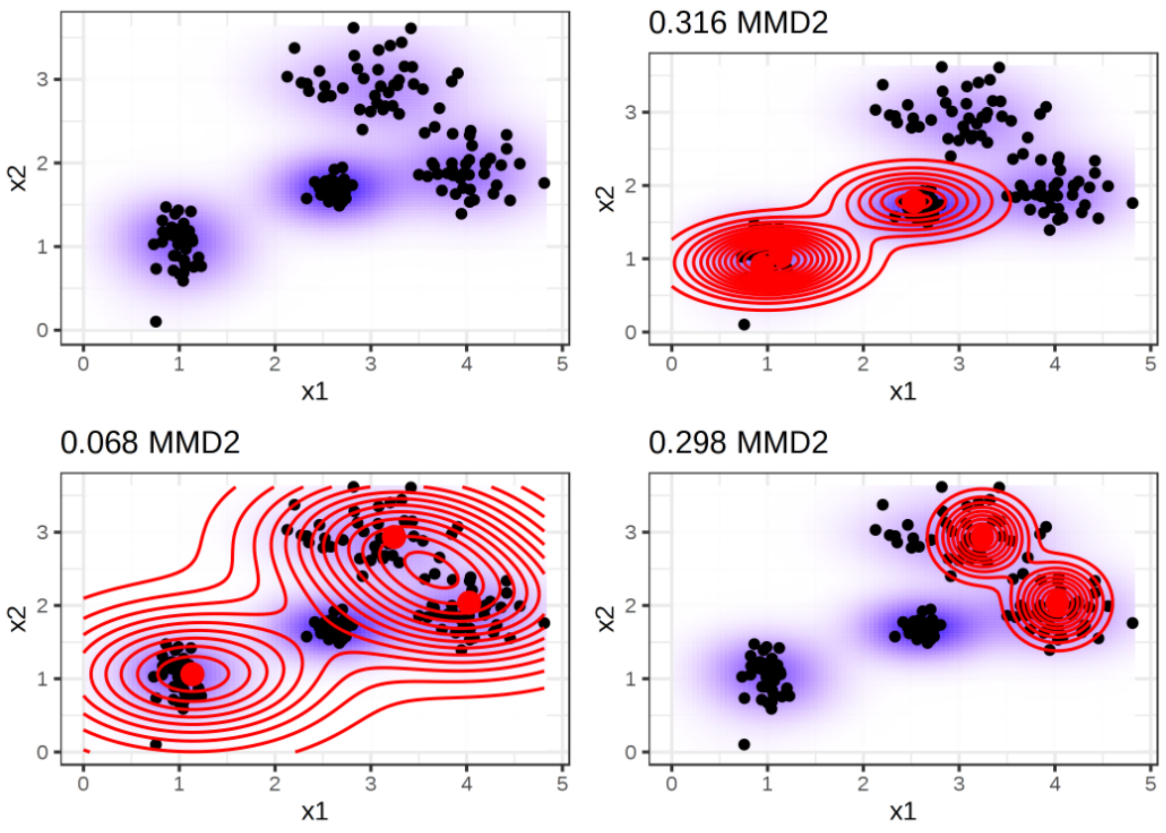


Figure 2.6: Different prototypes distributions in 2-dimensional data and corresponding MMD^2. Source: [Mol19a] https://christophm.github.io/interpretable-ml-book/proto.html

The witness function takes the main role in detecting criticisms by comparing prototypes and data distribution at a specific data instance. The instances with extreme results returned from witness function, or in other words, high absolute witness values, are considered as criticisms. A low negative value indicates that prototypes are located in low-density areas of data, while a high positive value shows the existence of various high-density data regions without prototypes being selected. In Figure 2.7, the instance in the middle with high absolute value can be a good selection for a criticism.

Besides MMD-critic, according to Molnar [Mol19a], clustering algorithms are good alternatives to effectively select prototypes and in our work, we choose K-means as a starting point to describe the space where criticisms and prototypes can be discerned[Llo82]. The algorithm starts with choosing randomly k numbers of instances in the dataset and assumes that they are the initial centroids, then iteratively proceeds in the two following steps:

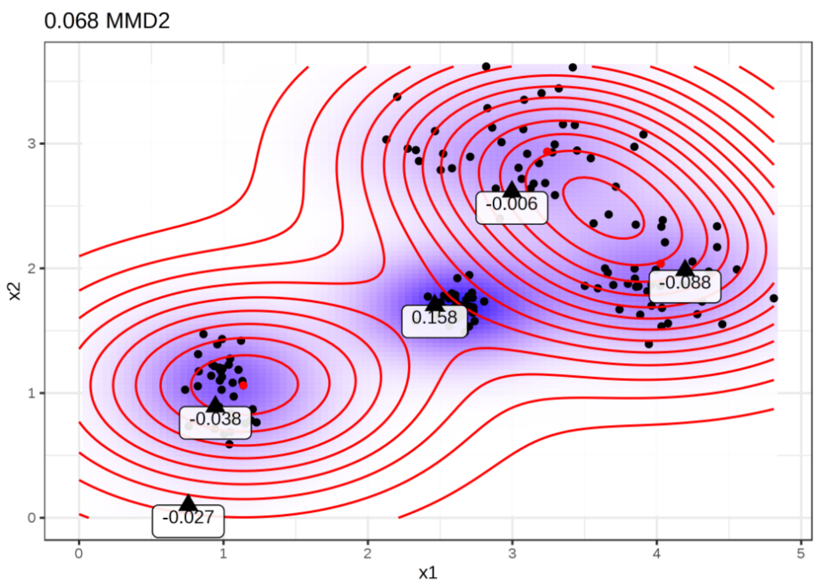


Figure 2.7: Witness function of different data instances in 2-dimensional data. Source: [Mol19a] https://christophm.github.io/interpretable-ml-book/proto.html

- Step 1: Estimate proximity between all instances in the dataset and selected k centroids using Euclidean distance [DD13]. Assign each instance to the cluster with the nearest centroid.
- Step 2: Update clusters centroids by re-computing the mean of clusters.

We repeat the procedure until convergence is achieved (i.e., centroids no longer change or a timeout is reached) and each centroid is the representative for the cluster it belongs to. Figure 2.8 illustrates an example of the K-means algorithm, its returned clusters and their centroids. However, while K-means and other clustering algorithms show their efficiency in detecting prototypes of data, Molnar points out their shortcoming in identifying criticisms.

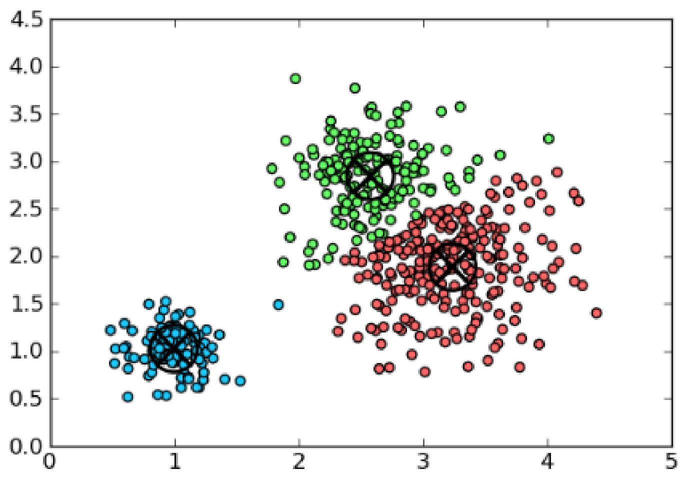


Figure 2.8: Examples of clusters returned after applying K-means and their centroids. Source: https://codefellows.github.io/sea-python-401d7/lectures/k_means.html

2.4 Research Gaps Identified

As we have mentioned previously, there is limited work about model internals and interpretability in DRL. To overcome this we seek a method that could provide such kind of insights, in parallel to the rewards on training cases which are usually reported. In specific we would like a method that is able to provide some understanding on the possible behavior that can be expected over unseen testing cases.

In our work, we choose prototypes and criticisms as a promising general approach for interpreting ML models, and specifically for DRL models, for two main reasons. First, model-specific techniques that are tailor-made for clarifying neural networks, such as: feature visualization and network dissection, aim to provide human explanations at local level of single prediction, which are not our current goal of gaining an overview on the underlying distribution of data. Second, theses techniques can be applied only for specific machine learning model and also, they require a human in the loop. Often these methods require human involvement either for the understanding of the instance-level insights, or for labeling pixels (in the case of network distillation). This is not the best option when we want to move forward to the scenario of automated machine learning.

Hence, the methodology that we select should not be limited to certain types of models, as well as it needs to minimize human interference. Prototypes and criticisms explanation, basically, satisfies our expectation such that a method like MMD-critic is shown to be able to work without human involved, and it can quite useful to explain the underlying distribution of data. Unfortunately, there is no research about how prototypes and criticisms identified by MMD-critic relate to the model performance. Therefore, in future work, we want to further dig deeper in this direction.

In the current scope, in absence of a starting point framework to automate MMD-crtic for ML and DRL, we base our work on the idea of using prototypes and criticisms to comprehend model behaviors. We take as starting point a method with clustering of the learned internal representations of the training data, attempting to understand how they correlate with the testing performance. We propose to study such method in both supervised and reinforcement learning scenarios.

2.5 Summary

Overall, in this chapter, we discuss the principle background in terms of concepts, past and current approaches used to obtain transparency in ML models, which are relevant to this research work, from section 2.1 to 2.3. Besides, some researched gaps existing in our thesis are also pointed out in 2.4. In the next chapter 3, we describe in details our design of the prototypical development for evaluation.

3 Design of the Prototypical Implementation for Evaluation

In this chapter we define our research aims, formulating our precise research questions (Section 3.1). Next, we present the design for a prototypical implementation of machine learning solutions and the interpretability tools, which we will use for our evaluation, addressing the proposed research questions (Section 3.2).

3.1 Research Questions

The overall concern that guides our work is to determine to which extent can the structure learned by a deep learning model in its last layers (regarding the training data), serve as indicators of the potential accuracy and generalization powers of the model. Specifically, we would like to understand the insights that this can produce in different learning setups, when compared to just the training error: Can the trends in structure learning be used as a predictor for model performance on unseen data? If so, what precise metric could be suggested to help practitioners have an idea about their model's performance in the future.

To address these questions, we organize our study into two main sub-problems:

1. For a CNN in charge of an image classification task (e.g., MNIST, CIFAR10), how does the trend in the structure learned by the last layers (for the train data) correlate to the model accuracy (on the test data)? To this end, which metric and clustering approach serve as better indicators for the structure, and why?

2. Considering the same setup as RQ1, what is the influence of regularization methods on the predictive power of the metric identified as correlating the structure to the test data accuracy? Is there a change in the observed correlation, due to the use of regularization?

3. Moving on to reinforcement learning, how does the structure learned by the last layers of a Deep Reinforcement Learning agent correlate to the rewards of the agent, and is the observed behavior (and choice of models), consistent with what we observed for supervised learning?

3.2 Prototypical Implementation for Evaluation

For our evaluation we design a workflow that includes selected machine learning tasks (supervised and reinforcement learning), considering datasets of different difficulties, model logging, check-pointing and core components for model performance analysis.

This workflow and the main components designed for this thesis and our experiments are presented in Figure 3.1.

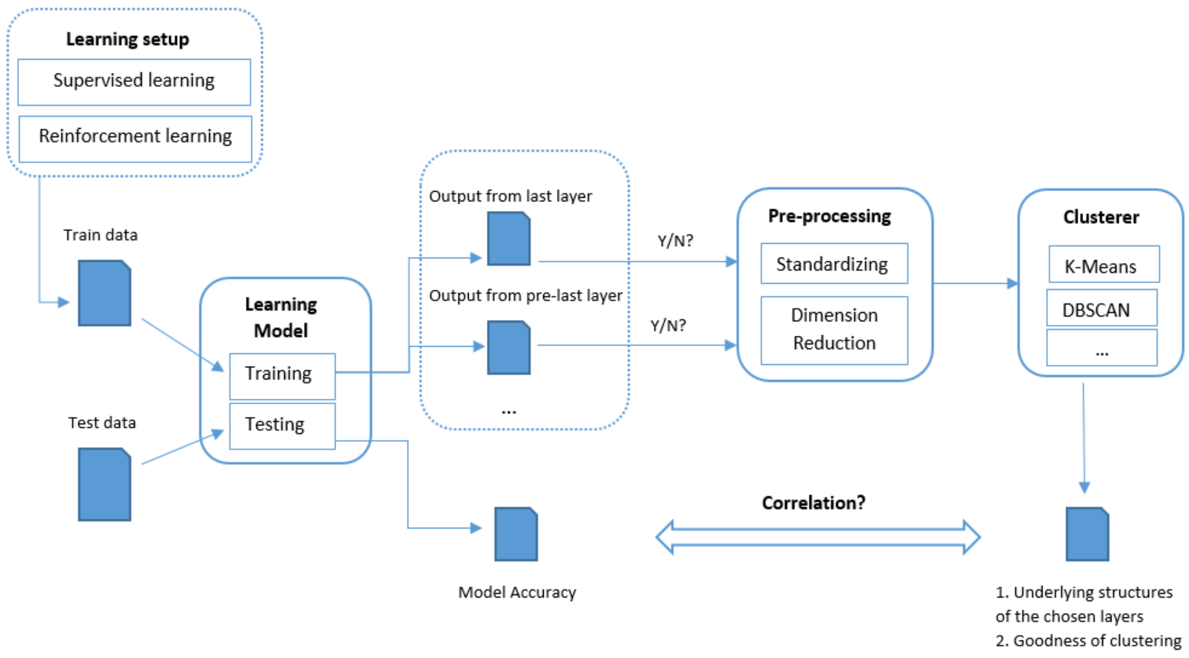


Figure 3.1: Main workflow of experiments

For our work we require three main components: a deep learning model that can work for supervised and reinforcement learning (*Learning model*), and two components that help the analysis of model performance (*Pre-processing* and *Clusterer*).

The flow starts with feeding the selected data into the first component for training and testing learning models. This could include both supervised or reinforcement learning. The traditional outcomes of the training or testing processes would be models training and testing accuracy. We extend these traditional metrics by also including information regarding the structure being learned by the different layers of the networks during the training process.

We forward this information on the structure being learned, next to the traditional metrics, to the next component (*Pre-processing*), if necessary. This step is optional, depending on whether the data in the layers is normalized or very high-dimensional. For the case of a layer with a very large number of neurons, a general clustering algorithm might not work well to give us insights on the underlying structure. In this case a method for dimensionality reduction (e.g. PCA), that precedes clustering could be employed.

The last component to support the analysis of the learned structure is the *Clusterer*. This component responsible for clustering data that comes either directly from the *Learning* *model* layers or the *Pre-processing*, to help in the analysis of the underlying structure of the data. This component also helps us to measure the quality of the obtained clusters. The final target of that our workflow supports is the analysis of the models behavior by enabling the study of the correlation between clustering goodness and model accuracy.

Some core goals that we aim to fulfill with this design are:

- Support of supervised and reinforcement learning.
- A design that is agnostic to the architecture of the neural network model, with the only expectation that the last layers are dense, and produce a result similar to classification. Our design and proposed analysis approach are not immediately suited for a single regression task.
- The incorporation of a pre-processing component, to be able to work with all kinds of high-dimensional data as represented in the last layers.
- The ability of adding, in a plug-and-play manner, clustering algorithms, and clustering goodness measures, to better suit diverse kinds of applications.

3.3 Summary

In this short chapter we introduced the three research questions that guide our work. We then described the core design of our prototype to study the relationship between the structure being learned and resulting model accuracy on unseen data. In the next chapter we describe our concrete choices for implementing the proposed design, setting-up the experimental framework for our thesis.

4 Experimental Setup

The content in this chapter includes all the information required to replicate the experiments described in this work:

- Section 4.1 describes the image classification data sets used for our study to understand the relation between the structure being learned in the layers of the neural network and the final accuracy at the prediction task.
- Section 4.2 introduces details on the deep learning architectures used for our evaluation. In specific we define relevant hyper-parameters chosen for our study.
- Section 4.3 discusses the methods used for pre-processing the data.
- Section 4.5 is a brief list of the software libraries used in our work.
- Section 4.4 shares the main points of how t-SNE visualizes high-dimensional data.
- Section 4.7 defines the evaluation metrics for our current study, including correlation metrics and clustering goodness measurements.
- Section 2.5 encapsulates the chapter with the major points included.

4.1 Data sets

For our experiments we selected two data sets for a supervised learning scenario. These are data sets commonly employed for the study of image classification machine learning models: MNIST ([LBBH98]) and CIFAR10 ([Kri12]). We select these datasets for three reasons: they are established datasets in the field, they are relatively small facilitating repeated training for the purpose of guaranteeing reproducible results for our experiments, and finally they represent two levels of complexity for the learning process (a simple and a slightly more complex case).

4.1.1 MNIST

MNIST, which stands for *Modified National Institute of Standards and Technology database* is an image classification dataset. It was produced by combining and modifying samples from data sets of the original American Institute of Standards and Technology data sets (NIST), which in turn contained hand-written digits from the American Census Bureau and American High School students. Currently the MNIST standard consists of a of total 70,000 black and white handwritten digits in 10 classes (from 0 to 9). In MNIST, each digit has been size-normalized and centered to form a 28x28-pixel-image. For our experiments we selected 60,000 samples for training purposes, while the remaining 10,000 are used for testing to estimate model accuracy. We do not do a random split, but use the pre-defined split as used in scikit-learn.

MNIST is a relatively simple dataset for the task, with state-of-the-art models reporting percentage errors as small as 0.23 since some years ago [CMS12], with the current best solution achieving 0.16 [BKD20].

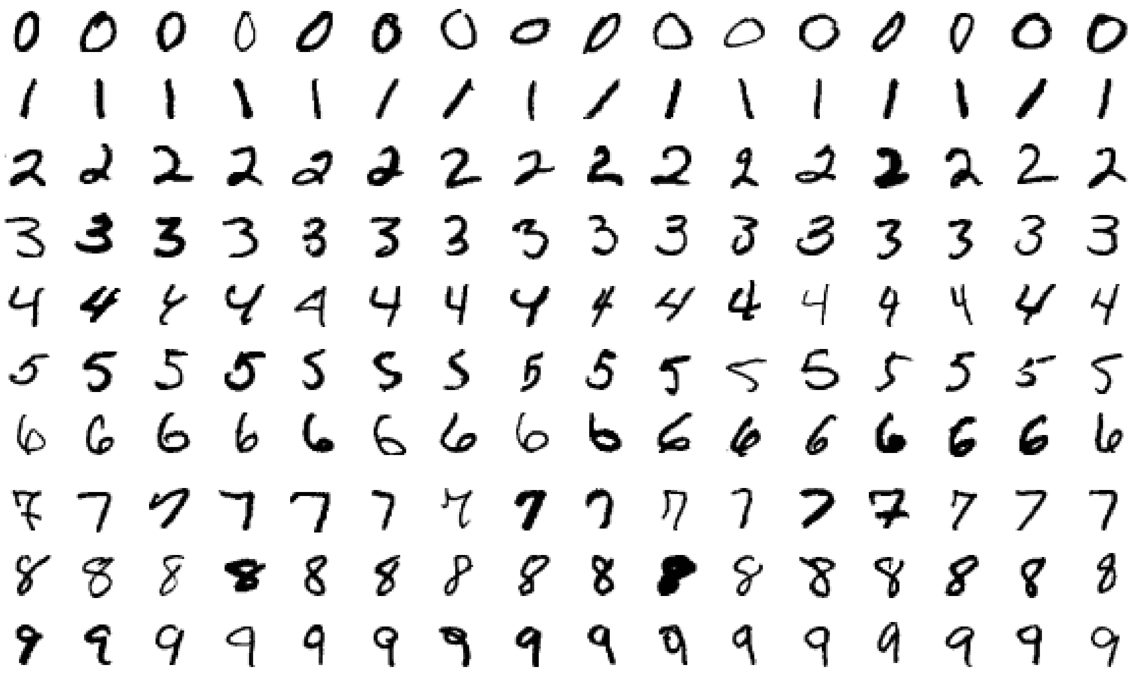


Figure 4.1: Samples from MNIST dataset. Source: https://commons.wikimedia.org/wiki/File:MnistExamples.png

4.1.2 CIFAR10

CIFAR10, a data set from the *Canadian Institute for Advanced Research*, is a more complex data set when compared to MNIST. It comprises 60,000 color images, formatted with a fixed-size of 32x32 pixels, and categorized into 10 classes. These classes represent animals (cats, dogs, birds, deer, frogs and horses) and means of transport (airplanes, cars, ships and trucks).

For training our model, 50,000 samples are picked and the other 10,000 are left for testing. For our study, we do not do a random split, but use the pre-defined split as used in scikit-learn.

A small set of CIFAR10 samples is shown in Figure 4.2.

Apart from the different classes, the image size and the use of color, CIFAR10 also differs from MNIST regarding its complexity. Since some years research efforts has helped practitioners reduce the percentage error from 9.380[GWFM$^+$13] to 0.63[KBZ$^+$19].

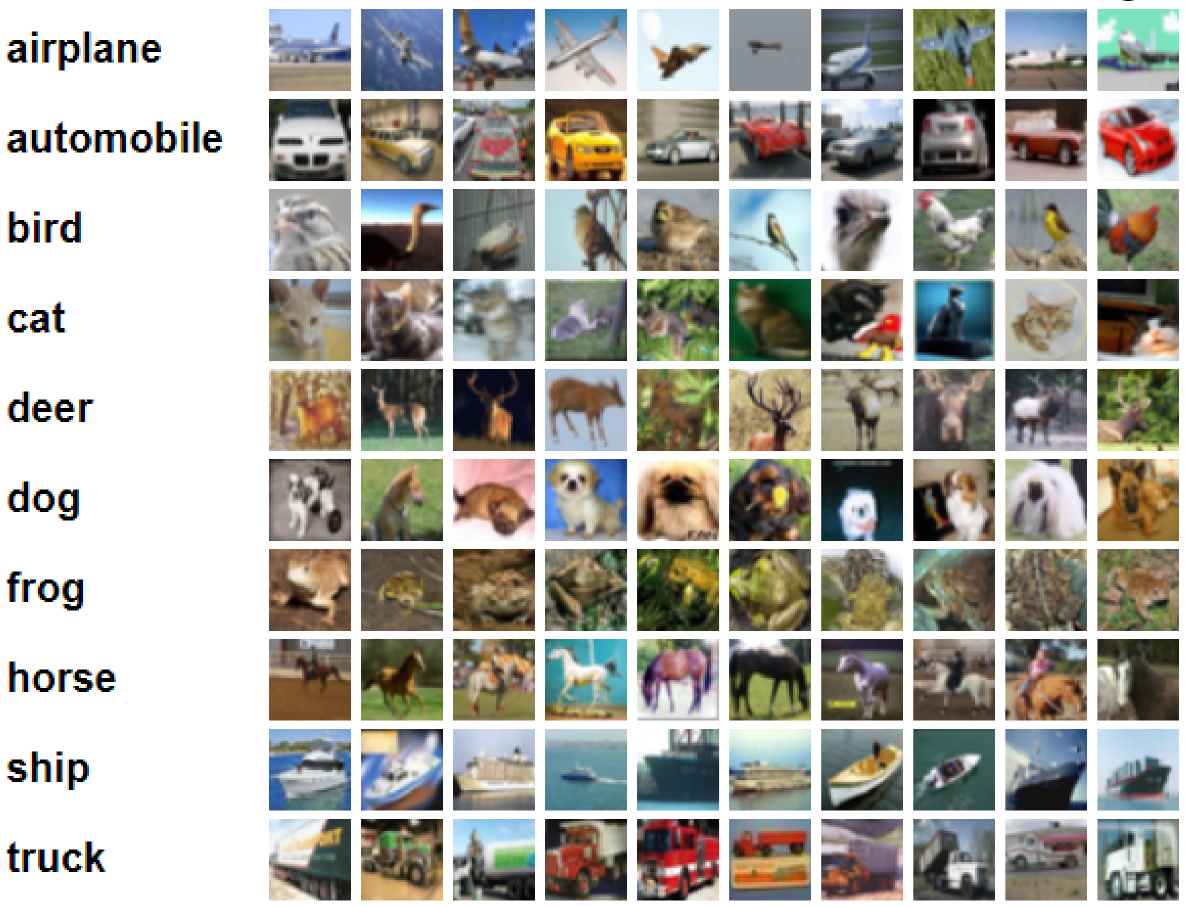


Figure 4.2: Samples from CIFAR10 dataset. Source: https://www.cs.toronto.edu/~kriz/cifar.html

4.2 Models architecture and relevant hyper-parameters

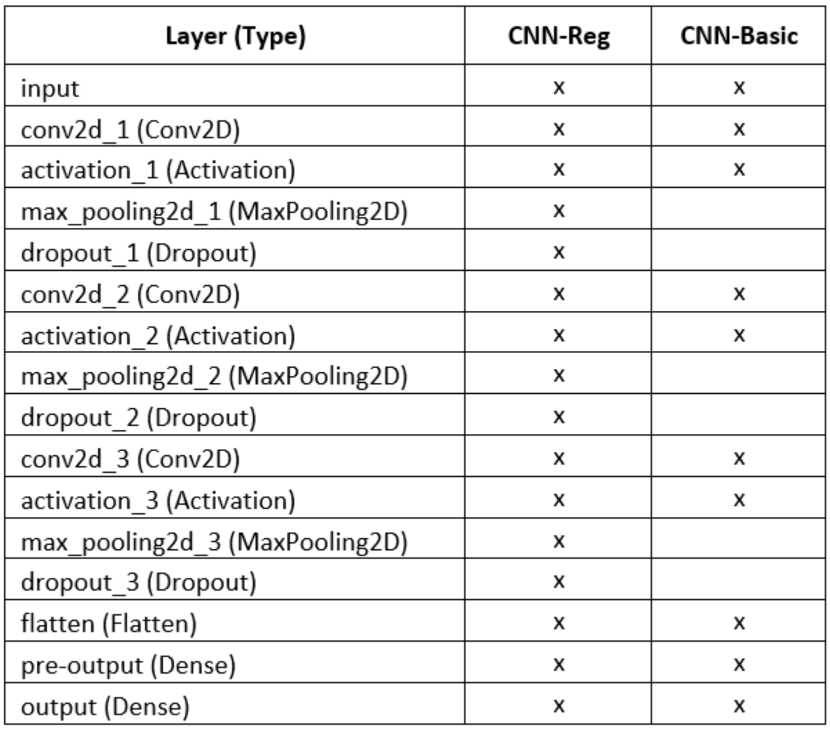
4.2.1 CNN architecture

In our experiments, we use convolutional neural networks, which are today's standard for image classification. We select to use a relatively simple architecture, which helps us to concentrate on the general interpretability problem that we study instead of being scoped to a very specialized architecture. So we build our own networks with a minimization in their complexity and target at gaining an understanding on models' behaviors in structuring data.

In our study we specifically select one basic network without any regularization method and another one with dropout and max pooling. We name these models respectively *CNN-Basic* and *CNN-Reg*. To be more detailed, besides input and output layers, CNN-basic is composed of three convolutional layers as hidden layers and each of them respectively contains 32, 64, 128 kernels of size 3x3, following by one flatten layer and one dense layer (fully connected layer) of 512 neurons. CNN-Reg is similarly constructed, but additionally incorporates in and between its convolutional layers: dropout function and max pooling 2D, whose relevant information and hyper-parameters will be discussed in the next section. Regarding activation function, rectified linear units (ReLU) is used in both models.

We are interested in studying the two last dense layers of the networks with respectively 512 and 10 neurons, and we name them *pre-output* and *output* for ease of reference in later analysis. We focus on these layers because they are simpler to study than the convolutional layers, We also select them because they are also featured in learning setups beyond supervised learning, for example, in reinforcement learning models, which include some convolutional layers (or even graph or tree-structured convolutions), which are then followed by dense layers [MKS+15b].

Regarding the use of regularization, we aim to study this to better understand the role of these improvements. By doing so we seek to understand to what extent the structure being learned can be an indicator of network performance.



Layer (Type)	CNN-Reg	CNN-Basic
input	x	x
conv2d_1 (Conv2D)	x	x
activation_1 (Activation)	x	x
max_pooling2d_1 (MaxPooling2D)	x	
dropout_1 (Dropout)	x	
conv2d_2 (Conv2D)	x	x
activation_2 (Activation)	x	x
max_pooling2d_2 (MaxPooling2D)	x	
dropout_2 (Dropout)	x	
conv2d_3 (Conv2D)	x	x
activation_3 (Activation)	x	x
max_pooling2d_3 (MaxPooling2D)	x	
dropout_3 (Dropout)	x	
flatten (Flatten)	x	x
pre-output (Dense)	x	x
output (Dense)	x	x

Figure 4.3: CNN-Basic and CNN-Reg Architecture

Figure 4.3 shows which components are included into each of the networks used for our study.

4.2.2 Relevant hyper-parameters

In the previous sections, we discussed *dropout* and *max pooling* as two regularization methods in networks. Regularization refers to any approach that aims at preventing overfitting to the training data during learning. Overfitting happens when a model is trained too well that it even learns also the noises or outliers in one data set and reaches high accuracy/low loss during training, but its performance degrades on unseen data or scenarios and cannot generalize well. Commonly regularization refers to adding explicit regularization terms to loss functions[Zha10], but it can also encompass techniques like early stopping or also dropout.

Dropout, during training, ignores a specific number of neurons in the input by re-setting their activation values to zero in a random manner, which means, both these neurons and their in- and out-coming connections in the network are temporarily drawn out [SHK+14]. The higher the dropout rate, the more units are removed. A valid rate in our experiments is in the range of [0,1), dropout = 1 that sets all activation values to zero is not intended and will be counted as a validity error. Some authors report [SHK+14] via experimental observations that a typical rate between 0.5 and 0.8 is usually effectively applied to a hidden layer, depending on the quantity of that layer's neurons. In our experiment, we choose the value of 0.7.

Max pooling is other regularization method that targets at down-sizing input dimensions and features, which are obtained from the layer before the max pooling layer (a convolutional layer in our models). Max pooling uses a n x n filter which is similar to an n x n matrix, and slides it across the entire input field for each of every input channel and returns the highest value of the areas that it slides through. The idea beyond that is to take out the most meaningful information from the input representation with a reducing computational cost, as well as to keep the network learning in a less detailed or more abstract manner for a better generalization performance on future unseen data. The filter size is optional depending on how much we want our model to downsample features. In our case, we choose max pooling 2D that uses a filter of 2 x 2 size.

4.3 Data pre-processing

Clustering in the high-dimensional environment is challenging since each data object can have up to hundreds of features or attributes. Many of them are noisy and irrelevant, which tremendously complicates the process of searching relevance among data objects to form clusters. Under the "curse of dimensionality" (Bellman, 1957) [Bel03], neither the clustering quality is guaranteed nor computational time is tractable. In these situation, dimension reduction techniques are used to transform high-dimensional to low-dimensional data space without loosing the important data properties [SVM14].

Among various dimension reduction techniques, principle component analysis (PCA) [MPH09] is the most widely used linear method that embeds data from high- to a new lower-dimension environment, keeping data variance at maximum level. The main ingredients of PCA are covariance matrix that contains covariance values between each pair of original features and eigenvectors (principles components) that are achieved by performing eigendecomposition on the covariance matrix. The first principle component is the eigenvector with highest eigenvalue that best maximizes the variation of the data, following by second, third, etc. components. In our experiments, we apply PCA with the fixed data variance of 95% on the outputs of the pre-last layer after standardizing the data to zero mean.

4.4 High-dimensional data visualization: t-SNE

t-SNE is abbreviated for t-distributed stochastic neighbor embedding that are proposed by Laurens van der Maaten et al. in 2008 [MH08a]. This method bases on stochastic neighbor embedding (SNE) approach introduced in 2003 by Hinton and Roweis [HR03], but is simpler in optimization and provides much better visualization of data. Laurens van der Maaten and his team has proved an outperformed visualization of t-SNE via their experiments comparing with other techniques such as Isomap [TSL00], Sammon Mapping [Sam69] and Locally Linear Embedding [RS01] on different dataset of MNIST [LBBH98] and COIL-20 [NNM96].

t-SNE starts with calculating probability distribution among pairs of data objects in original high-dimension environment. The larger the probability, the more similar the objects. Its next step is to compute another probability distribution among the same data but embedded in lower-dimensional space and aims at minimizing the Kullback-Leibler divergence, which reflects the proximity of two probability distributions in this embedding space. In our experiment, we follow the suggestion of Laurens van der Maaten and apply t-SNE after performing PCA (variance of 95%) on the original data to remove noisy data as well as reduce calculating time.

4.5 Software/Libraries in use

Some major software and libraries that we use to carry out our experiments include python 3.6, tensorflow 2.3.0, keras 2.4.3 and sklearn 0.22.2.post1.

4.6 Chosen environment for DRL

Our experiments of DRL are generally focused on a Deep-Q-Learning (DQN) agent [MKS+15a] as well as a chosen environment from bsuite. Bsuite is an open source DRL profiling framework from DeepMind. It consists of numerous environments and training scenarios designed to analyse and evaluate the capability of a reinforcement learning agent regarding generalization, exploration, scaling, noise in the environments, credit assignment, memory, etc [ODH+20]. With the framework the authors provide have multiple baseline agents that are assigned to tackle different challenges. We focus on their DQN implementation, which has an architecture similar to the one used in our supervised learning experiments. We also focuson testing this agent on an environment called MNIST, or alternatively a Bandit environment. The agent is put in this environment, where the input data of MNIST is splitted into two subsets for training and testing purposes. On one episode the agent receives one MNIST image, and its task is to classify it. Therefore each episode consists of a single action. The DQN agent, then, learns the data and gets either a reward of +1 for each accurate prediction or a reward of -1 for inaccurate ones. The total training and testing rewards are aggregated from analysing. Our experiments apply the same procedure and environment, only the DQN agent has some changes in its neural network architecture, such that its CNN construction replicates the CNN-Reg described in 4.2. We train for 10000 episodes, which conform one training "epoch" and we test for 1000 episodes, which conform one testing "epoch".

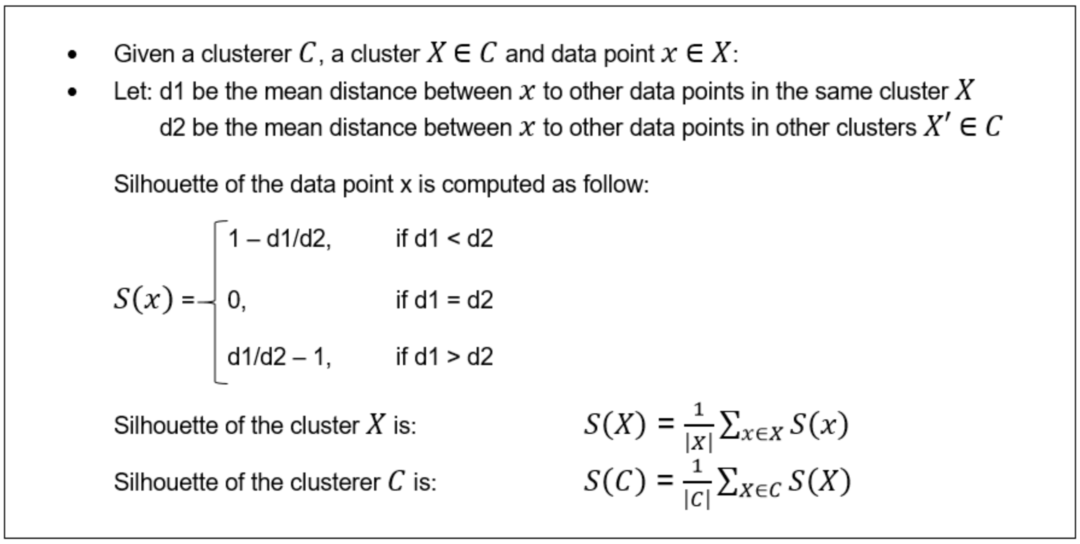
4.7 Evaluation metrics

Moving forward with the components of our workflow, as discussed in the previous chapter (see Figure 3.1)

4.7.1 Clustering goodness measurements

In unsupervised learning, various metrics can be used to evaluate clustering quality. They can be either external measures that require class labels or internal measures that no extra external information needed [GCM08][LLX+13]. In our study, we apply clustering technique on the results obtained from two last layers of our CNN models, where no class labels are available for pre-output layer, therefore, we choose internal validation methods that are Silhouette and Davies Bouldin indexes.

Silhouette coefficient [Rou87] allows users to know about how close or separate the samples within a cluster and also among clusters together, without any presence of data labels. It is computed based on pairwise inter- and intra-cluster distances. Values are in a range of [-1, 1]. In specific, a value close to +1 indicates that the sample is far away from neighboring clusters, or in other words, the sample is probably assigned to the correct cluster. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters, whereas a negative value near -1 indicates that clustering results might be poor.



- Given a clusterer C, a cluster $X \in C$ and data point $x \in X$:
- Let: d1 be the mean distance between x to other data points in the same cluster X
  d2 be the mean distance between x to other data points in other clusters $X' \in C$

Silhouette of the data point x is computed as follow:

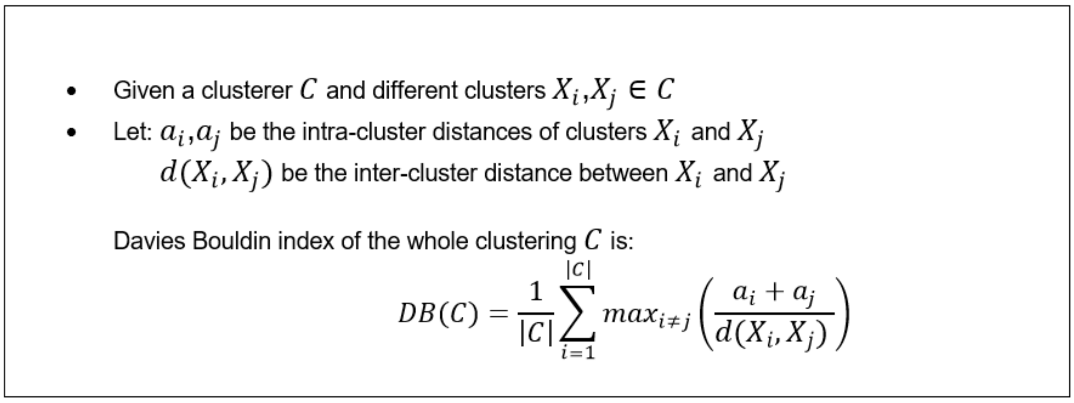
$$S(x) = \begin{cases} 1 - d1/d2, & \text{if } d1 < d2 \\ 0, & \text{if } d1 = d2 \\ d1/d2 - 1, & \text{if } d1 > d2 \end{cases}$$

Silhouette of the cluster X is: $S(X) = \frac{1}{|X|} \sum_{x \in X} S(x)$

Silhouette of the clusterer C is: $S(C) = \frac{1}{|C|} \sum_{X \in C} S(X)$

Figure 4.4: Silhouette Coefficient

Davies Bouldin [DB79][HBV01][Pet06] or in short, DB index is named after the authors' last names David L. Davies and Donald W. Bouldin. It considers the mean of all clusters similarities. A cluster similarity is defined by taking the maximum in the list of values resulted from calculating similarities between a cluster and the other ones. A smaller index indicates greater separation among clusters as well as better clustering quality, and 0 is the smallest value can be obtained. Computing Davies Bouldin index usually takes less time compared to Silhouette coefficient.



- Given a clusterer C and different clusters X_i,X_j $\in C$
- Let: a_i,a_j be the intra-cluster distances of clusters X_i and X_j
  $d(X_i, X_j)$ be the inter-cluster distance between X_i and X_j

  Davies Bouldin index of the whole clustering C is:

$$DB(C) = \frac{1}{|C|}\sum_{i=1}^{|C|} max_{i \neq j}\left(\frac{a_i + a_j}{d(X_i, X_j)}\right)$$

Figure 4.5: Davies Bouldin Index

4.7.2 Correlation measurements

Correlation measures, in general, help explore the associations between variables and figure out the scale of dependencies among them. In our experiments, they are used to statistically capture the relations between models testing accuracy and data structures learnt in two last layers of the models.

Pearson coefficient [Pea05][Wri21][Kir08] quantitatively discovers whether two variables linearly correlated to each other with a scale from -1 to +1. A value close to +1 or -1 represents a strong linear dependency, and 0 means no linear association between variables. Given X and Y, the case that both X and Y increases or decreases simultaneously results in a positive value, whereas X increases as Y decreases returns a negative one.

Spearman coefficient [Spe04][SD78] is also a correlation measure with a similar scale compared to Pearson. However, they have some differences. While Pearson assumes a normal data distribution and only returns accurate results if there exists linear relationships among variables, Spearman is, on the other hand, a non-parametric measure since it has no assumption about data distribution and considers the associations that related by a monotonic function, which includes also non-linear.

4.8 Summary

Generally, in this chapter, we provide all materials that are needed to reproduce the experiments, ranging from datasets, models architecture, agent environment, pre-processing methods, visualization technique until the validating metrics for clustering algorithms. The experiments results are reported in our next chapter 5.

5 Evaluation and Results

In the following chapter we disclose the final details regarding the experimental setup for our work. We also present the results of our empirical evaluation, discussing the most valuable observations. We structure the chapter into specific experiments according to our research questions, for each of them we discuss the setup, our hypotheses, observations and discussion. In detail, this structure is as follows:

- In Section 5.1 we start the chapter with the experiments pertaining the structure being learned in default supervised learning. We cover the visual analysis of the structure being learned, clustering and evaluation measures, finally we study the correlation between clustering goodness measures and testing accuracy.
- In Section 5.2 we consider the impact of regularization on the structures being learned, and on the correlations observed.
- In Section 5.3 we include our studies related to reinforcement learning.
- We close this chapter in Section 5.4, discussing the key takeaways from our experiments, and highlighting open questions.

5.1 Learned Structures in Supervised Learning

In this section we address the first of our research questions:

1. For a CNN in charge of an image classification task (e.g., MNIST, CIFAR10), how does the trend in the structure learned by the last layers (for the train data) correlate to the model accuracy (on the test data)? To this end, which metric and clustering approach serve as better indicators for the structure, and why?

This question focuses on clarifying the correlation between data structures learnt inside two last layers when training CNN-Basic and the final model testing accuracy, as well as discussing which clustering approach and goodness metrics serve as better indicators for the structure.

In order to address this question, we carry out three different experiments, as listed below:

1. Experiment 1.1: Visual Analysis of Learned Structures
2. Experiment 1.2: Clustering and Goodness Metrics
3. Experiment 1.3: Evaluation of Correlations Between Learned Structures and Test Accuracy

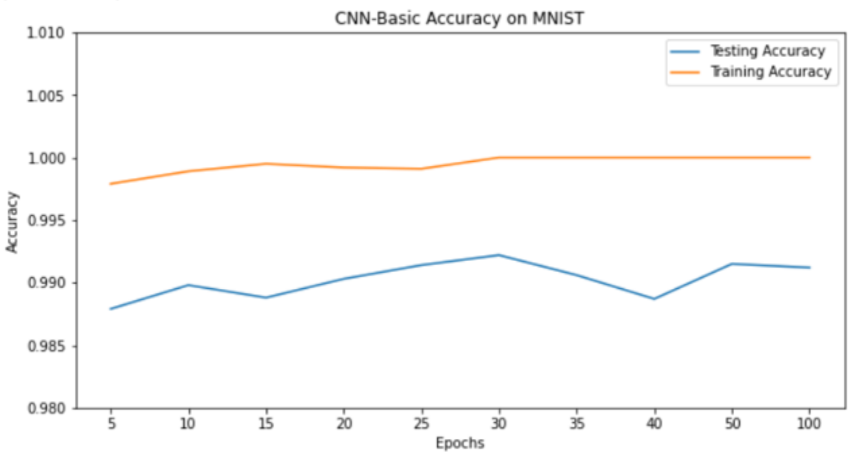
5.1.1 General Experiment Details

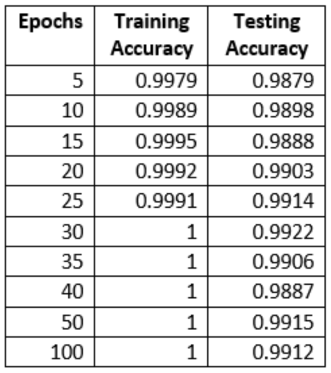
For these experiments we select the MNIST and CIFAR10 datasets. For our experiments we selected 60,000 and 50,000 samples for training purposes (respectively), while using the remaining 10,000 for testing. We rely on the pre-defined split provided by scikit-learn. We use the neural network architecture and training hyper-parameters, as described in Section 4.2, with no regularization. We call the model employed CNN-Basic.

We start with training and testing our models CNN-Basic on train and test samples of MNIST and CIFAR10, to get the necessary outputs for further analysis, including:

- Model training and testing accuracy
- Outcomes obtained from two last layers after training model.

Due to the different complexity levels of the two datasets, we decide to spend less time for training MNIST (up to 100 epochs) and more for CIFAR10 (up to 300 epochs). The number of epochs refers to the times that the model traverses across the whole training dataset for learning. For example, 5 epochs mean the model works through the training data 5 times. Figure 5.1 and Figure 5.2 shows the model performance on both sets.





Epochs	Training Accuracy	Testing Accuracy
5	0.9979	0.9879
10	0.9989	0.9898
15	0.9995	0.9888
20	0.9992	0.9903
25	0.9991	0.9914
30	1	0.9922
35	1	0.9906
40	1	0.9887
50	1	0.9915
100	1	0.9912

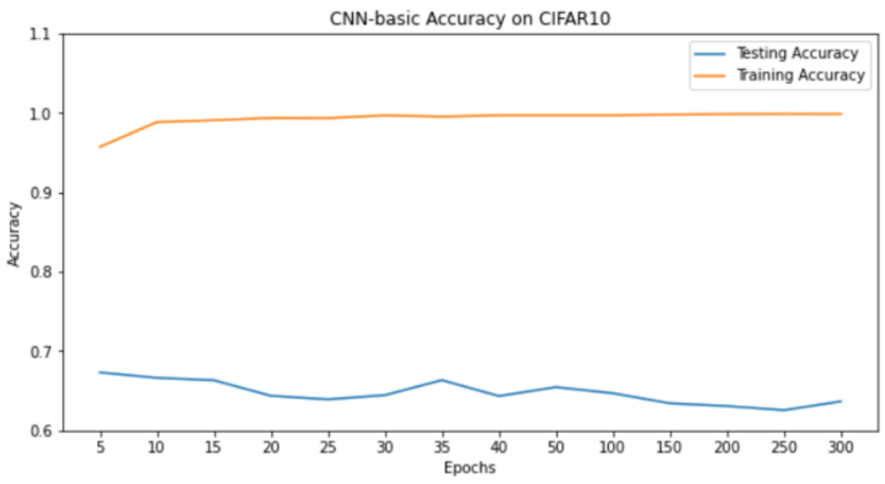
Figure 5.1: CNN-Basic Accuracy on MNIST

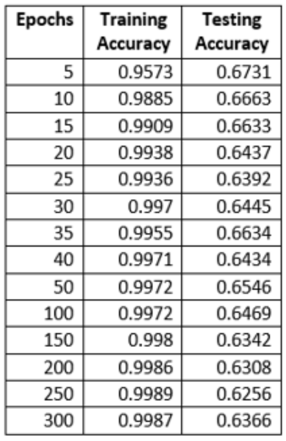
As we can see from the graphs, CNN-Basic's testing accuracy on MNIST is, overall, higher than that on CIFAR10, no matter how much more time we spend on learning for CIFAR10. Besides, the network reaches to the time-point of making no training errors in classifying MNIST much faster than the remaining dataset.

5.1.2 Visual Analysis of Learned Structures

Experiment Details

In this experiment we are concerned with identifying visually the structure being learned for the training data on the last layers. To this end we train the neural network for a





Epochs	Training Accuracy	Testing Accuracy
5	0.9573	0.6731
10	0.9885	0.6663
15	0.9909	0.6633
20	0.9938	0.6437
25	0.9936	0.6392
30	0.997	0.6445
35	0.9955	0.6634
40	0.9971	0.6434
50	0.9972	0.6546
100	0.9972	0.6469
150	0.998	0.6342
200	0.9986	0.6308
250	0.9989	0.6256
300	0.9987	0.6366

Figure 5.2: CNN-Basic Accuracy on CIFAR10

series of epochs, and take snapshots of how the network is behaving over the training data. In these snapshots we record all the output values for the whole training dataset given by the last and the second to last layers (dense layers) of the architecture. We then aim to study the structure for these snapshots, i.e., how the different input data items are mapped across the high dimensional embedded space. To assist us in this study we employ T-SNE [MH08b], an algorithm that maps high dimensional data to a 2D space, while preserving the density of items. For this algorithm we used a perplexity of 30, and a number of steps of 5000.

Hypotheses

For this experiment we have the following hypotheses:

1. Our first hypothesis is that the overall structure being learned for MNIST will visually present more clearly defined clusters than the structure for CIFAR10. We conclude this based on the complexity of the dataset, and how distributed across the image are the features which need to be learned to distinguish between the classes.

2. Our second hypothesis concerns the number of clusters visually present. We expect this number to be close to the target number of clusters.

3. Our final hypothesis has to do with the layers, we expect that the overall structure in the last layer (the one in charge of the classification itself) will be more defined into clusters than the previous layer.

Results

To clarify whether our hypotheses are confirmed, our next step is using T-SNE to visualize the training data items, as extracted from their learned representations int he output and pre-output layer.

Results are plotted in Figure 5.3 and Figure 5.4. It is quite clear from the images that data structures emerge gradually corresponding to the number of epochs: the more we train, the clearer structures we have.

Comparing the two layers, data is visibly better structured in the output layer, especially in CIFAR10. After 300 epochs, although data from the pre-output layer starts to form groups in a more apparent manner than that after only 5 epochs and dense areas are also able to see, the separation between them are still very uncertain. In contrast, the visualization from output layer shows us an obvious formation of ten clusters in data as well as far distances among them. The observation in MNIST is similar, where we can find more complex structures being captured in the plots on the left rather than the right side of pre-output layer.

Our results also show simpler structures in MNIST than in CIFAR10. After 100 epochs data in MNIST can be very well learnt regarding the observable number of clusters (10) as well as their clear separation in both layers, which is not the case in pre-output layer when training the model on CIFAR10, where the clusters are not easy to discern.

Finally, we observe generally as training progresses a cluster structure that could be captured by eliptical or circular shapes, however in the case of MNIST we see shapes that are more density-related and look like squids.

Discussion

Regarding our hypotheses, our results validate some of them. First, results confirm the simpler structure being learned for MNIST. This both related to the higher accuracy reached on the testing data for this dataset, and also to its simpler characteristics.

We find however that the shape of clusters in the output layer for MNIST is unexpected regarding our first hypothesis, because they do not follow a shape easy to distinguish. We consider that this might be due to the case of hand-written digits like 3 and 8, which might have many cases close to each other. We consider that this might be the cause for the squid-like shapes: we have many data points that are decisively a 3, but many 3s could also gravitate, in different ways, towards a representation similar to that of the number 8. Further studies should be done to verify this.

Our second hypothesis was clearly confirmed on all cases, as the number of clusters visually present has 10 for most cases when the training epochs reach their maximum. For the case or CIFAR10, and the pre-output layer, we see 10 clusters emerging but it is not too clear.

Regarding the overall structure in the layers, we find that the structure of the last layer is informative. In the case of MNIST the last layer preserves the same number of target clusters, but the clusters do not follow such an easy to discern shape as those present in the pre-output layer. For the case of CIFAR10 the observation is almost the opposite: the clusters in the output layer are clearly distinguishable and match the target number, but the clusters in the previous layers fail to do so. In the absence of more evaluations, we consider our test to suggest that the output layer might be generally as informative or

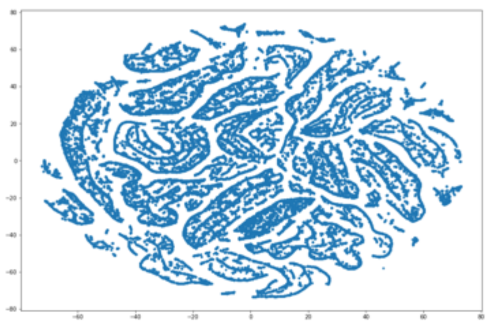
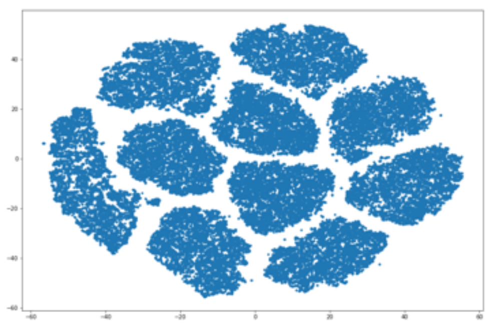
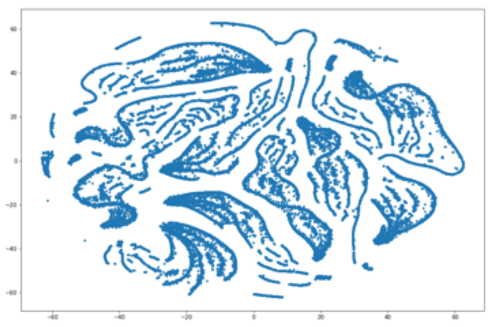
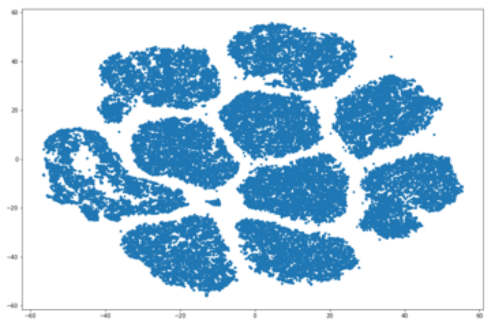
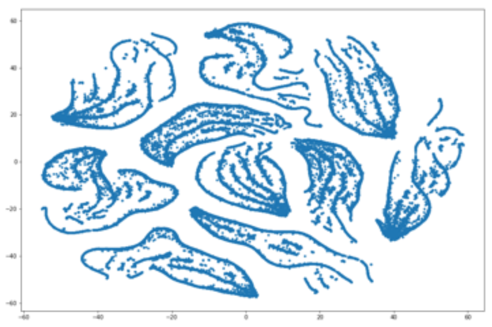
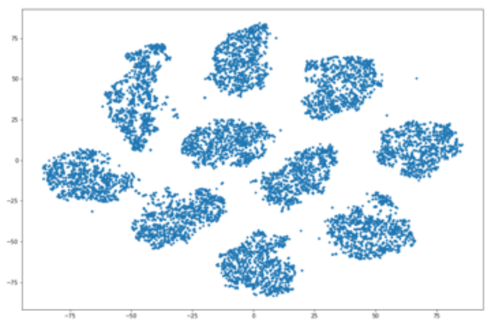
	T-SNE visualization of MNIST trained by CNN-Basic	
	Output layer	**Pre-output layer**
CNN-Basic 5epochs		
CNN-Basic 40 epochs		
CNN-Basic 100 epochs		

Figure 5.3: T-SNE visualization of MNIST trained by CNN-Basic

more than the pre-output layer, however we cannot rule out that on other scenarios this might not be the case.

5.1.3 Clustering and Goodness Metrics

Based on the previous experiments, we are aware that clusters can emerge when training is successful. In this experiment we aim to properly study these clusters, and their quality, by employing an unsupervised clustering method (K-means) and evaluating clustering goodness measures such as the Silhouette coefficient and Davies Bouldin metric.

Based on t-SNE plotting results, the clusters do not seem to be complicated density-based cases, so we choose K-means as the clustering algorithm because of its generality and as well, its working mechanism that intrinsically identifies clusters prototypes.

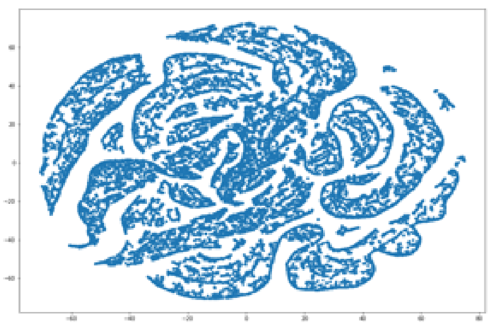
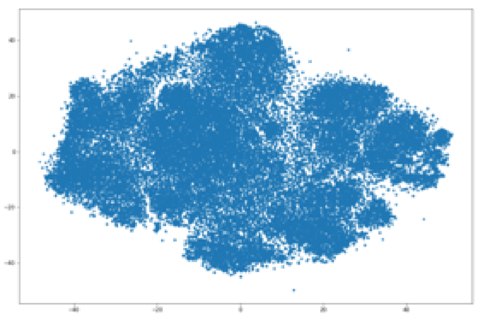
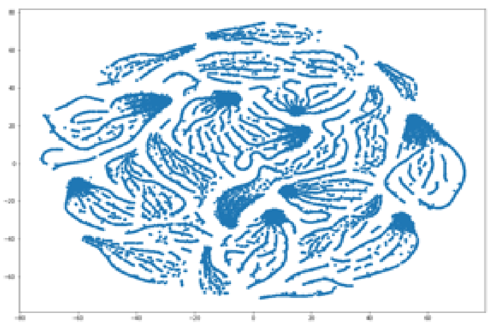
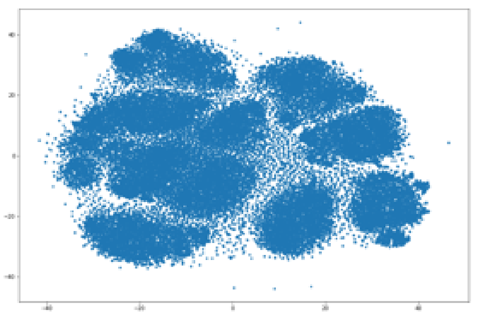
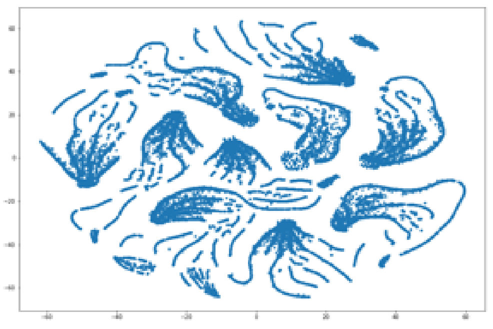
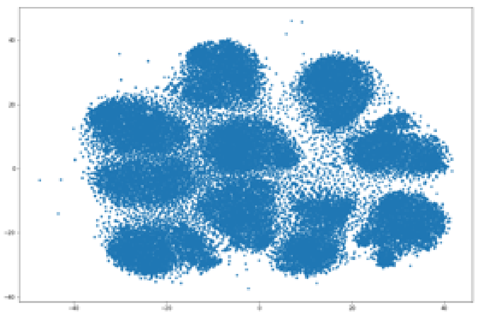
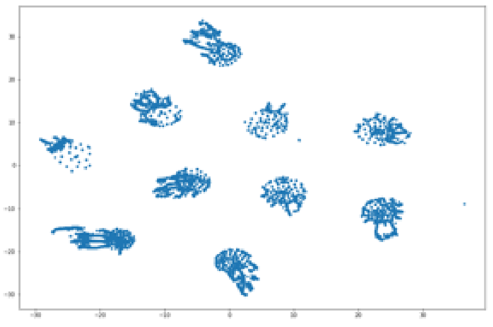
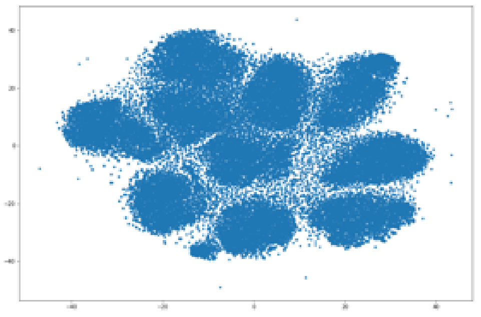
	T-SNE visualization of CIFAR10 trained by CNN-Basic	
	Output layer	Pre-output layer
CNN-Basic 5epochs		
CNN-Basic 40 epochs		
CNN-Basic 100 epochs		
CNN-Basic 300 epochs		

Figure 5.4: T-SNE visualization of CIFAR10 trained by CNN-Basic

Experiment Details

Based on the previous experiments, we are aware that clusters can emerge when training. In this experiment we aim to properly study with more precision these clusters, and their quality, by employing an unsupervised clustering method (K-means) and evaluating clustering goodness measures such as the Silhouette coefficient and Davies Bouldin metric.

For this experiment we used the same train/test splits, training configuration and neural architecture as described previously for MNIST and CIFAR10.

Because of differences in dimensions of output (10 features) and pre-output (512 features), we decide to apply dimensionality reduction to the pre-output layer, so the results for k-means for the two dimension spaces are comparable.

Hypotheses

For this experiment we have the following hypotheses:

1. Towards the end of training, the number of clusters observed as the best configuration should be the same as the output classes of MNIST and CIFAR10 (10).
2. The clustering goodness measures will continue increasing throughout the training, and they will also exhibit saturation (i.e, reach a state where there is not improvement in the measure), which is parallel to the training accuracy saturation.
3. The goodness measures reported for the layers will be consistent with the observations on the T-SNE plots, which prefer the output layer for CIFAR10 and the pre-output layer for MNIST.
4. The overall clustering metrics reportd for MNIST will be better than those of CIFAR10 due to complexity of the datasets.

Results

The results for MNIST are presented in the figures that follow: Figure 5.5, Figure 5.6, Figure 5.7, Figure 5.8.

Altogether these results show a consistent pattern of improvement in the output layer, with a saturation to the optimal value (1 for the silhouette coefficient, and 0 for the Davies Bouldin metric), which is furthermore reached at the precise number of clusters expected. We also observe the knee pattern expected for the Silhouette coefficient, and a similar v-shape pattern with a minima in 10.

The results for the pre-output layer do not show such a successful clustering, with the best results reaching values that are still far from the optima, adding up to only a quarter of it (0.25) or similarly poor values (1.417). In spite of this unsuccesful values, the best point identified is consistently at 10 clusters, with few variations in the case of Davies Bouldin, which suggest this metric to be slightly less robust than the alternative.

The results for CIFAR10 are presented in the figures that follow: Figure 5.9, Figure 5.10, Figure 5.11, Figure 5.12.

The results for CIFAR10 are consistent with those of MNIST with regards to the last layer. The optima value is found consistently at 10 clusters, with a saturation to the value of 1.0 for Silhouette coeffcient and the Davies Bouldin metric of 0.001. The Davies

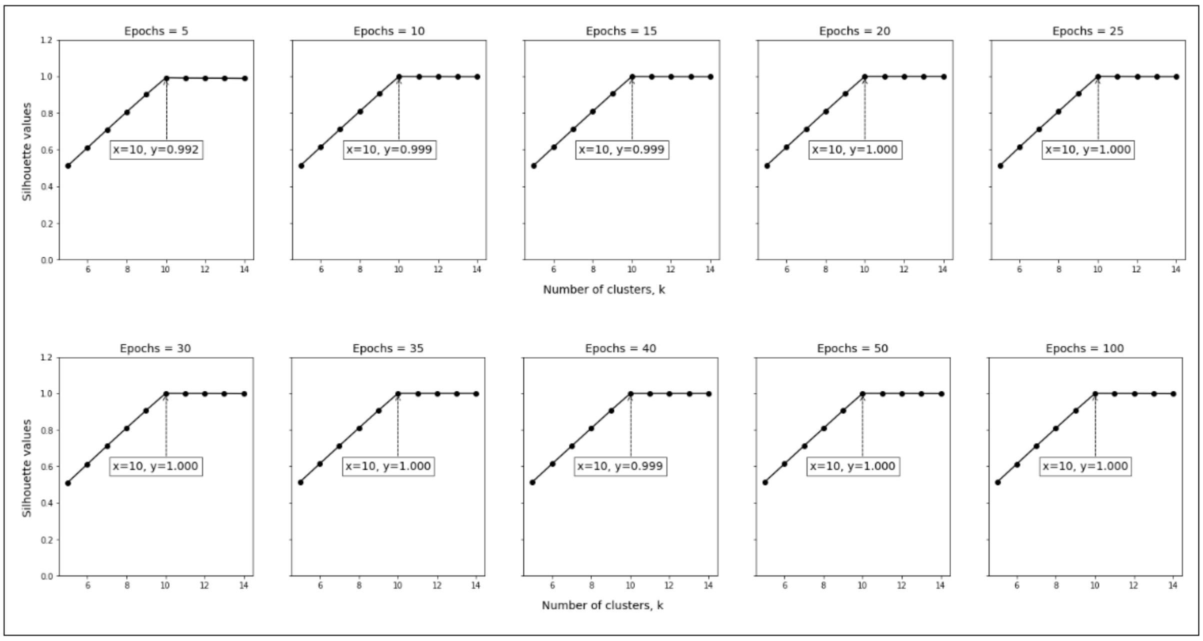


Figure 5.5: [MNIST] CNN-Basic/Output Layer/Clustering validated using Silhouette Coefficient

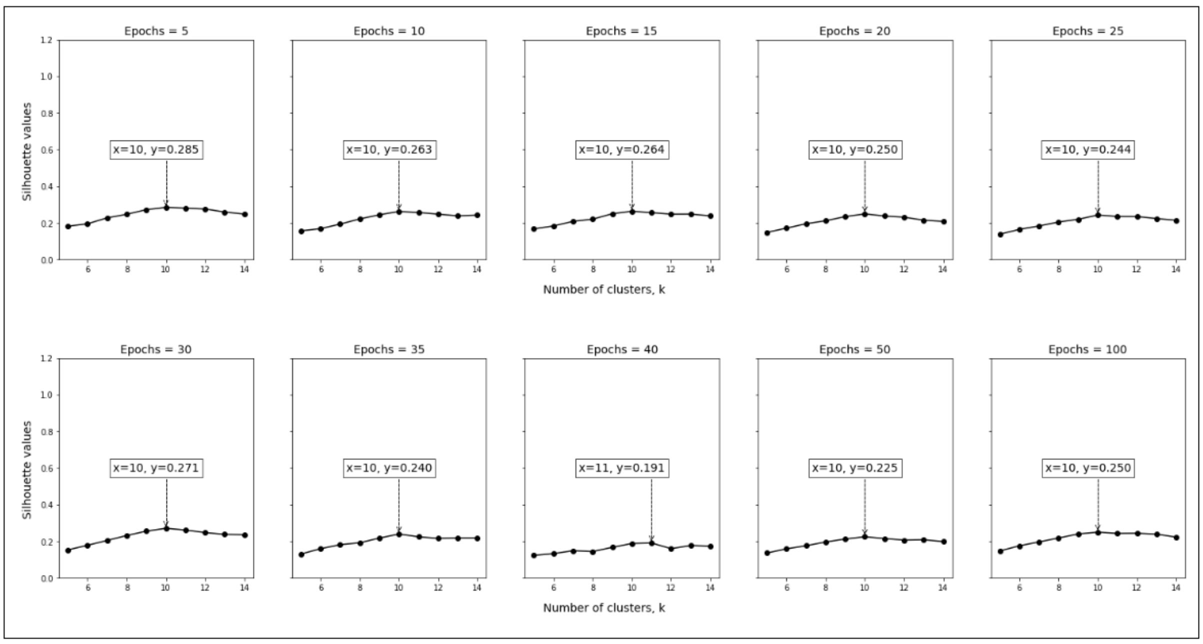


Figure 5.6: [MNIST] CNN-Basic/Pre-output/Clustering validated using Silhouette Coefficient

Bouldin metric also showed less robustness than the alternative metric, with the optimal value showing some variations (11, instead of 10).

The results of the pre-output layer are quite poor, showing that there is a limited amount of structure being learned. Both metrics show a great amount of oscillation in the optima being found, without ever being close to the 10 clusters that could be expected.

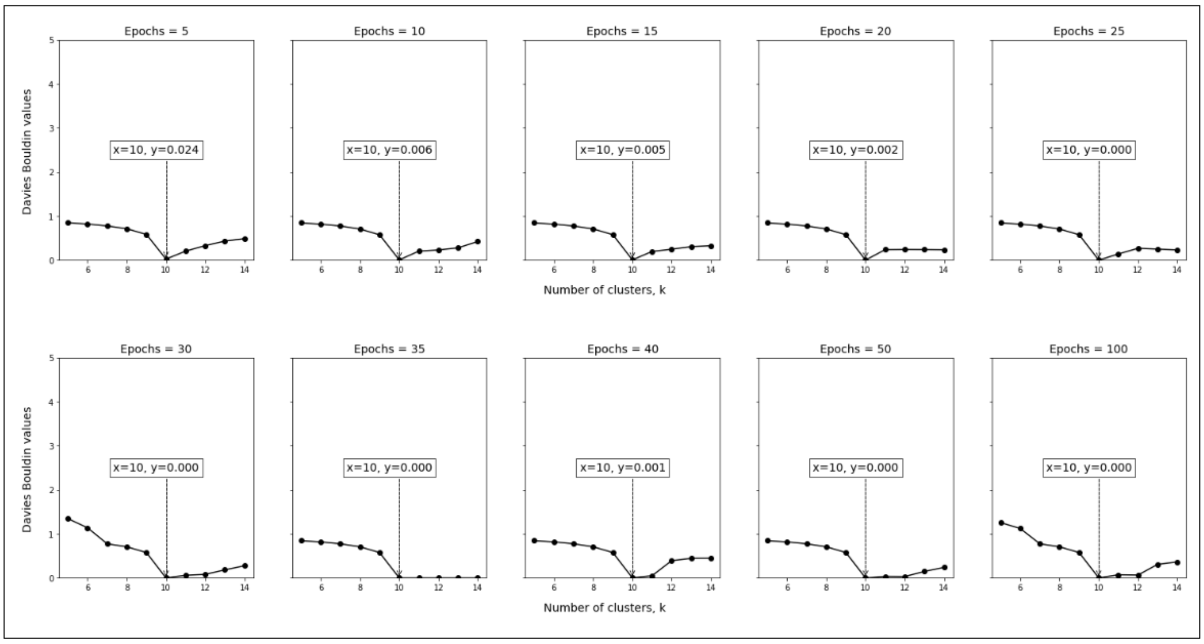


Figure 5.7: [MNIST] CNN-Basic/Output Layer/Clustering validated using Davies Bouldin

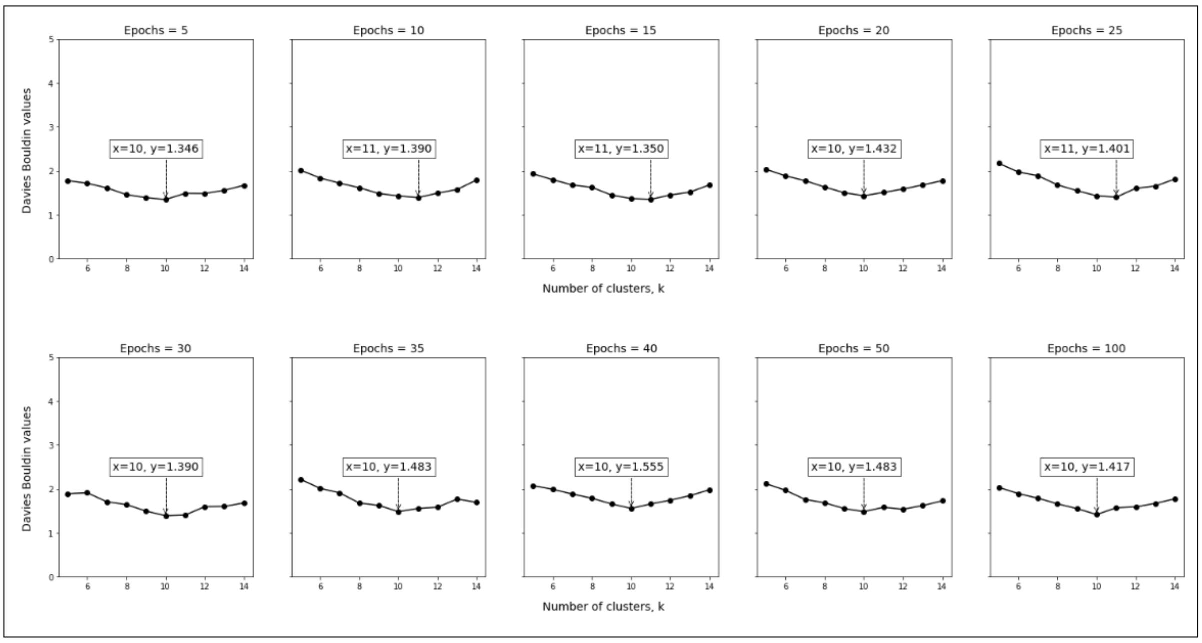


Figure 5.8: [MNIST] CNN-Basic/Pre-output/Clustering validated using Davies Bouldin

Discussion

Our observations from this experiment confirm many of our hypotheses, specially in the output layer. For this layer we see that the number of clusters identified at the end of training is 10, for both datasets. Our observations also show saturation that matches the same iterations when it occurs during training. For the case of MNIST this occurs at epoch 20 in the clustering (when the training accuracy is already 0.9992), and at epoch 30 in the actual training accuracy (when it reaches 1.0). For the case of CIFAR10 saturation occurs

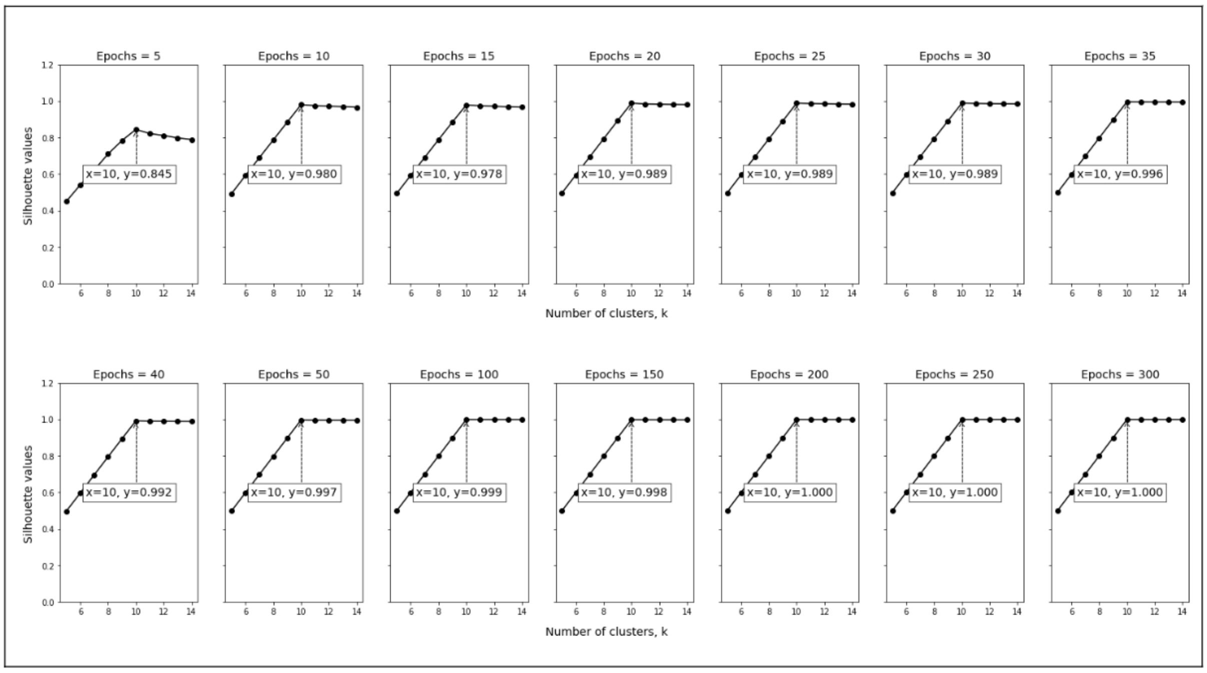


Figure 5.9: [CIFAR10] CNN-Basic/Output Layer/Clustering validated using Silhouette Coefficient

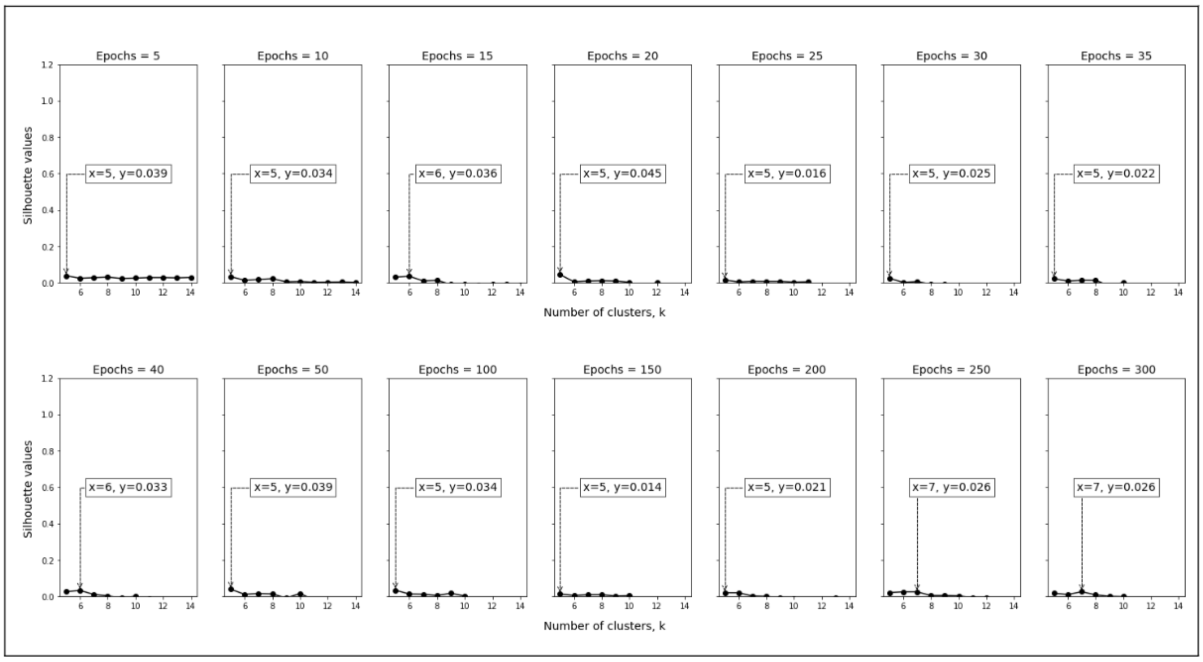


Figure 5.10: [CIFAR10] CNN-Basic/Pre-output/Clustering validated using Silhouette Coefficient

at the epoch 200 in the clustering, while it never converges exactly to a training accuracy of 1.0, but it reaches a value of 0.9986 at the epoch 200.

Regarding the pre-output layer, for MNIST the results achieve a poorer Silhouette coefficient that does not reach the optima of 1.0, but at least consistently reports 10 clusters as the best configuration. Similar results occur for the Davies Boulding met-

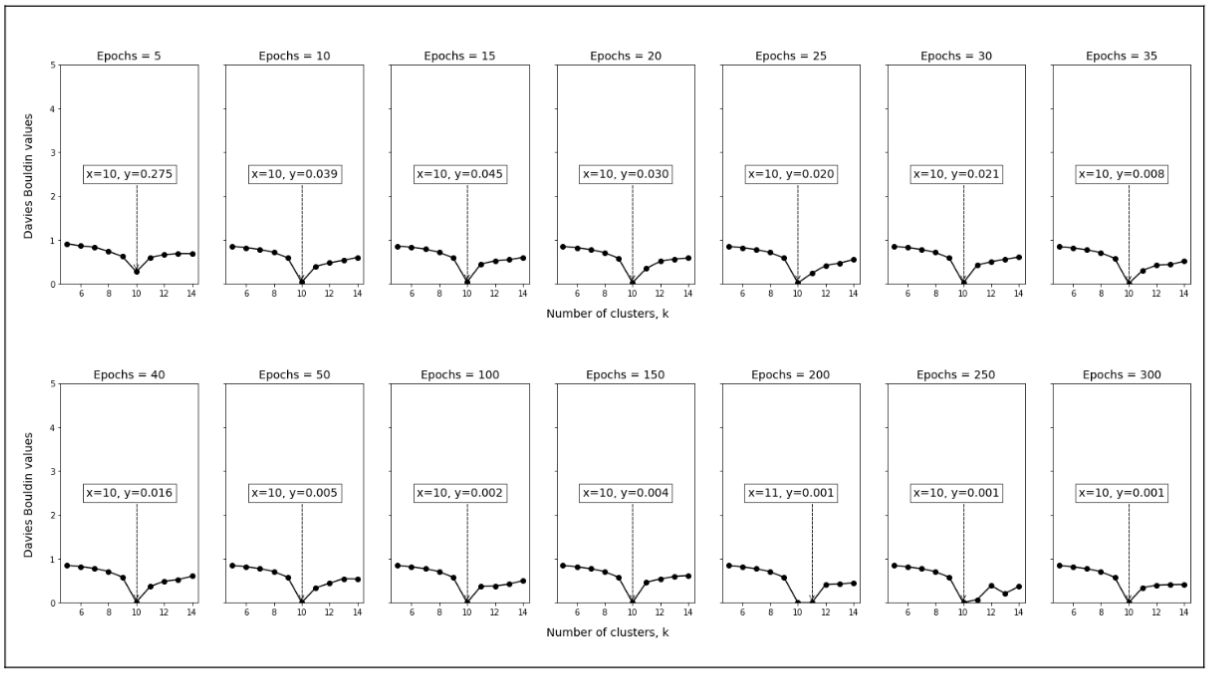


Figure 5.11: [CIFAR10] CNN-Basic/Output Layer/Clustering validated using Davies Bouldin

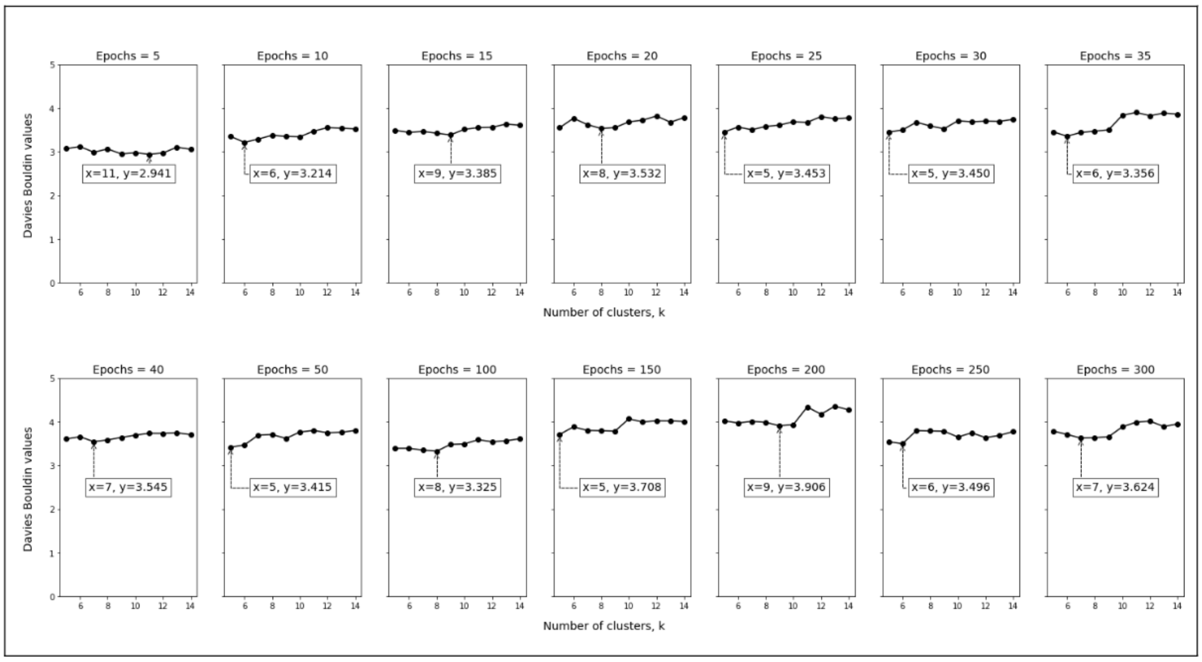


Figure 5.12: [CIFAR10] CNN-Basic/Pre-output/Clustering validated using Davies Bouldin

ric. For CIFAR10, on the other hand the results are markedly poor, failing to reach the optima and not even reporting good results at the ideal 10 clusters for both metrics.

The observations we found for the output layers are consistent with our TSNE results, showing a good clustering measure. We also observe no difference in the goodness of clustering for MNIST and CIFAR10 at this layer. This last result is a bit at odds with the TSNE visualizations, where we observed the clusters for MNIST being a bit less easy to distinguish than those of CIFAR10 for the layer.

Regarding the pre-output layers our results are not consistent with the TSNE observations. There we saw some structure emerging and being easy to distinguish at this layer for MNIST. This was not the case for the clustering goodness metrics.

These last observations are in contrast to our hypothesis regarding them. We consider this to be a consequence of TSNE creating more visually consistent clusters than those actually present in the data at 10 dimensions.

Considering now the last of our hypotheses for this experiment, we observed that the clustering goodness measures do not differ notably in the last layer across the datasets, but we do find that the pre-output layer for MNIST better results than those of CIFAR10. Unfortunately the results for both layers is rather poor.

Concerning the robustness of metrics, our experiments show that the Davies Bouldin metric can be empirically a less robust indicator of the clustering goodness when compared to the Silhouette coefficient. We base this conclusion on the variability observed for the Davies Bouldin coefficient on layers when the optima was being eventually found. On these cases we found the optima to be alternating between 10 and 11, which is not the expected result.

5.1.4 Evaluation of Correlations Between Learned Structures and Test Accuracy

Experiment Details

In this experiment we repeat the training and test configurations as in the other experiments in this section. However, we study in specific the results from the clustering metrics from the layers, as we seek to understand their correlation with the testing accuracy.

Hypotheses

Based on our previous experiments, we have the following hypotheses:

1. We expect that overall the correlation observed between test accuracy and clustering goodness will be better observed for the Silhouette coefficient than for the Davies Bouldin metric.
2. We expect that the output layer will display the strongest correlations.
3. We expect that from the correlations observed to the test accuracy we will find cases on the output layer comparable or better (not severely worst) than that of the training accuracy.
4. We expect to observe a better report on correlation if the reports consider only data before the saturation (on cases where there is saturation), rather than from after it.

Epochs	Training Accuracy	Testing Accuracy	O-Sil	O-Db	PO-Sil	PO-Db
5	0.998	0.988	0.992	0.024	0.285	1.346
10	0.999	0.990	0.999	0.006	0.263	1.428
15	1.000	0.989	0.999	0.005	0.264	1.369
20	0.999	0.990	1.000	0.002	0.250	1.432
25	0.999	0.991	1.000	0.000	0.244	1.430
30	1.000	0.992	1.000	0.000	0.271	1.390
35	1.000	0.991	1.000	0.000	0.240	1.483
40	1.000	0.989	0.999	0.001	0.189	1.555
50	1.000	0.992	1.000	0.000	0.225	1.483
100	1.000	0.991	1.000	0.000	0.250	1.417

Table 5.1: CNN-Basic accuracy on MNIST

	Train Accuracy	P-Val	O-Sil	P-Val	O-Db	P-Val	PO-Sil	P-Val	PO-Db	P-Val
PS-All	0.565	0.055	0.711	0.010	−0.719	0.008	−0.041	0.899	0.112	0.729
PS-Pre	0.542	0.345	0.829	0.083	−0.851	0.067	−0.952	0.012	0.905	0.035
PS-Post	NA	NA	0.901	0.037	−0.902	0.036	0.893	0.041	−0.886	0.046
SP-All	0.473	0.120	0.879	0.000	−0.830	0.001	−0.164	0.611	0.176	0.585
SM-Pre	0.473	0.421	0.879	0.050	−0.830	0.082	−0.164	0.793	0.176	0.777
SM-Post	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Table 5.2: Correlations with Testing Accuracy- CNN-Basic- MNIST

Results

Table 5.1 and Table 5.2 report the training and testing accuracy, with the correlations identified between different measures (in the columns) with the test accuracy, for CNN-Basic and MNIST. Table 5.3 and Table 5.4 display the same information for this architecture and the CIFAR10 dataset.

Discussion

Before beginning our discussion, we should note that statistically significant correlations should correspond to p-values lower than 0.005. Looking at Table 5.2, we find that the measure correlation between test and train accuracy is thus not statistically reliable for our measurements. For most cases, however, the correlations found (independent of the value) are statistically reliable.

We can observe that we find strong correlations when looking at the output layer, with a slightly stronger correlation observed for the Silhouette coefficient. This is consistent with our expectations for both the role of the output layer, and the relative contribution of the clustering goodness measure. We also observed, for the cases where the p-values were reliable enough, that the correlation observed improved when discounting the impact of saturation. The correlation was higher when looking at the area post saturation only, this could be due to the variability during training measurements which is less when saturation is achieved. We used 25 as the cutoff between saturation and non saturation.

Epochs	Training Accuracy	Testing Accuracy	O-Sil	O-Db	PO-Sil	PO-Db
5	0.957	0.673	0.845	0.275	0.026	2.977
10	0.989	0.666	0.980	0.039	0.007	3.343
15	0.991	0.663	0.978	0.045	−0.009	3.513
20	0.994	0.644	0.989	0.030	0.003	3.684
25	0.994	0.639	0.989	0.020	0.004	3.685
30	0.997	0.645	0.989	0.021	−0.021	3.708
35	0.996	0.663	0.996	0.008	0.003	3.836
40	0.997	0.643	0.992	0.016	0.002	3.693
50	0.997	0.655	0.997	0.005	0.016	3.763
100	0.997	0.647	0.999	0.002	0.003	3.490
150	0.998	0.634	0.998	0.004	0.004	4.065
200	0.999	0.631	1.000	0.001	−0.010	3.933
250	0.999	0.626	1.000	0.001	0.004	3.650
300	0.999	0.637	1.000	0.001	−0.000	3.883

Table 5.3: CNN-Basic accuracy on CIFAR10

	Train Accuracy	P-Val	O-Sil	P-Val	O-Db	P-Val	PO-Sil	P-Val	PO-Db	P-Val
PS-All	−0.670	0.009	−0.596	0.025	0.608	0.021	0.413	0.142	−0.684	0.007
PS-Pre	−0.664	0.026	−0.602	0.050	0.609	0.047	0.395	0.230	−0.715	0.013
PS-Post	−0.631	0.566	−0.959	0.183	0.370	0.758	−0.266	0.829	0.749	0.461
SM-All	−0.807	0.000	−0.727	0.003	0.736	0.003	0.327	0.253	−0.556	0.039
SM-Pre	−0.637	0.035	−0.468	0.146	0.503	0.115	0.269	0.424	−0.609	0.047
SM-Post	−0.327	0.788	−0.982	0.121	0.327	0.788	−0.577	0.609	0.676	0.528

Table 5.4: Correlations with Testing Accuracy- CNN-Basic- CIFAR 10

These results suggest that the output layer structure can be a good indicator of the accuracy on held-out test data.

For the case of CIFAR10, we observe too that most cases are able to report many correlation observations that can be deemed to be statistically reliable. We observe first that there is a general negative correlation between test and train data, which suggests over-fitting. This is consistent with the relative low test accuracy observed throughout the training. Contrary to the case for MNIST, we observe comparable results to those of the correlation between test and train accuracy (not such a notable improvement from using the clustering goodness). We also observed the Davies Bouldin index and the Spearman coefficient to produce competitive results on some configurations. Finally we observe some stronger correlations for the pre-output layer than with the output layer, which we consider is inconsistent with our expectations.

We used 150 as the cutoff between saturation and non saturation. Our results for MNIST suggest that clustering goodness measures on train data can provide some amount of insights into the testing accuracy, but that still on a case with a negative correlation between test and train, this contribution is moderate.

Epochs	Training Accuracy	Testing Accuracy	O-Sil	O-Db	PO-Sil	PO-Db
5	0.947	0.985	0.937	0.110	0.342	1.027
10	0.960	0.988	0.964	0.065	0.326	1.093
15	0.964	0.990	0.968	0.056	0.298	1.136
20	0.967	0.990	0.967	0.062	0.269	1.145
25	0.968	0.992	0.971	0.051	0.214	1.221
30	0.969	0.991	0.973	0.049	0.222	1.261
35	0.972	0.991	0.978	0.040	0.218	1.217
40	0.972	0.991	0.976	0.042	0.211	1.224
50	0.974	0.993	0.978	0.041	0.194	1.248
100	0.977	0.992	0.982	0.033	0.148	1.330

Table 5.5: CNN-Reg accuracy on MNIST

5.2 Impact of Regularization on Learned Structures in Supervised Learning

5.2.1 Evaluation of Correlations Between Learned Structures and Test Accuracy

Experiment Details

In this experiment we recreate the scenario of our previous experiment, but we focus on the case of regularization.

Hypotheses

We can list the following hypotheses:

1. We estimate that the correlations will be stronger than those observed so far, in spite of the lowered training and testing results. This is a result of the regularization methods: an improvement for one dataset should contribute for the held-out data.
2. We reason that the output layer will present good correlations overall, with Silhouette coefficient providing a stronger signal. It is uncertain whether the correlation will be an improvement over that between test and train data.
3. We argue that the use of regularization will make the behavior of the correlation metrics on both datasets more similar.

Results

Table 5.6 and Table 5.8 show the correlations observed for the MNIST and CIFAR10 datasets, when using CNN-Reg.

	Train Accuracy	P-Val	O-Sil	P-Val	O-Db	P-Val	PO-Sil	P-Val	PO-Db	P-Val
PS-All	0.961	0.000	0.958	0.000	−0.959	0.000	−0.887	0.000	0.907	0.000
SM-All	0.903	0.000	0.915	0.000	−0.879	0.000	−0.939	0.000	0.806	0.002

Table 5.6: Correlations with Testing Accuracy- CNN-Reg- MNIST

Epochs	Training Accuracy	Testing Accuracy	O-Sil	O-Db	PO-Sil	PO-Db
5	0.489	0.428	0.296	1.098	0.094	1.838
10	0.547	0.411	0.384	0.989	0.080	2.010
15	0.575	0.521	0.398	0.988	0.046	2.041
20	0.596	0.416	0.420	0.872	0.034	2.265
25	0.613	0.462	0.436	0.833	0.018	2.439
30	0.631	0.605	0.457	0.860	0.026	2.420
35	0.637	0.580	0.455	0.854	0.003	2.483
40	0.639	0.627	0.480	0.812	0.014	2.576
50	0.653	0.597	0.505	0.821	−0.001	2.617
100	0.688	0.615	0.549	0.765	−0.028	2.733
150	0.701	0.673	0.558	0.739	−0.043	2.874
200	0.709	0.670	0.567	0.725	−0.041	2.996
250	0.720	0.659	0.593	0.712	−0.070	3.070
300	0.725	0.649	0.603	0.674	−0.071	3.028

Table 5.7: CNN-Reg Accuracy on CIFAR10

	Train Accuracy	P-Val	O-Sil	P-Val	O-Db	P-Val	PO-Sil	P-Val	PO-Db	P-Val
PS-All	0.881	0.000	0.867	0.000	−0.805	0.001	−0.849	0.000	0.868	0.000
SM-All	0.895	0.000	0.903	0.000	−0.881	0.000	−0.864	0.000	0.881	0.000

Table 5.8: Correlations with Testing Accuracy- CNN-Reg- CIFAR 10

Discussion

For both datasets, regularization contributes to more reliable p-values, giving us more certainty on the observed correlations.

The first observation we have is that the results indeed show stronger correlations. This is made more evident in the CIFAR10 dataset.

For the case of MNIST we find that the correlation between test and train accuracy is the highest, followed by that between the test and the output layer with the Silhouette coefficient. The clustering goodness measures on the pre-output layer also reports strong correlations.

For the case of CIFAR10 we find comparable results to those of the MNIST dataset, though with slightly lower (but still high) correlation values.

Altogether these results confirm our expectations with regards to this experiment.

5.3 Learned Structures in Reinforcement Learning

5.3.1 Bandit Environment

Moving on to reinforcement learning, in this section we seek to understand how our observations thus far, regarding clustering goodness and their power to serve as estimators of the test data behavior, are mapped to this learning setup.

To this end we use the so-called bandit environment from B-suite. This is an environment that presents to a reinforcement learning agent a problem that can be solved in a single action. The problem is for image classification. The task of the agent is to classify an image within 10 classes. The agent receives a reward of -1 if the classification is incorrect and 1 otherwise. The agent runs for 300 iterations/epochs. Each iteration consists of 1000 episodes for training and 100 for testing. At the end of an iteration the agent reports its average reward for both testing and training episodes. In order to be consistent with benchmarks like MNIST, the b-suite implementation uses the same data split as we did in previous experiments, to distinguish test from train data.

Experiment Details

For our experiments we use a DQN agent comparable to the basic CNN architecture used thus far. The hyper-parameters of this agent is reported in the experimental setup chapter. We employ both datasets MNIST and CIFAR10 for training the agent.

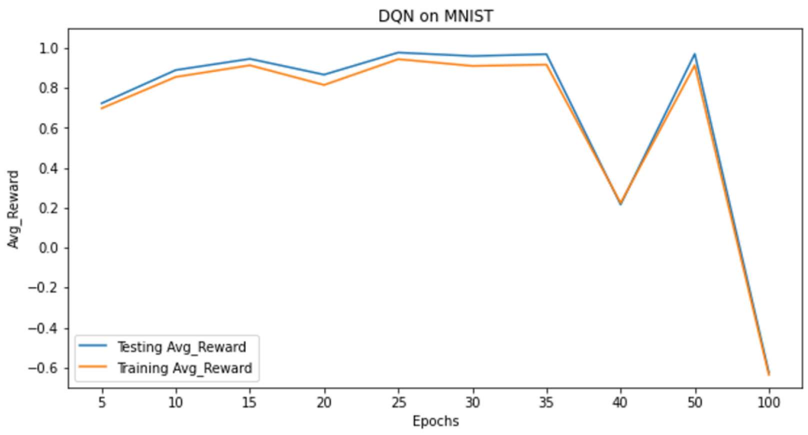
Hypotheses

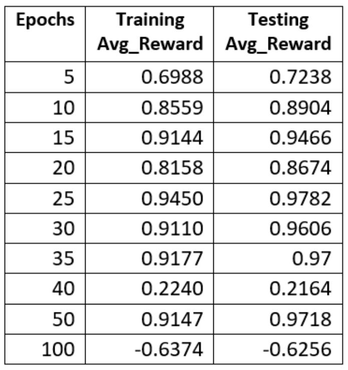
For our evaluation of DQN on both datasets we expect the following:

1. We expect there to be differences with respect to supervised learning. The accuracy learned will be less, as a consequence of the sample inefficiency of the RL method employed.
2. We expect the correlation between the output layer clustering goodness and that of the testing reward to be better than the correlation of the testing and training rewards. We reason that this is due to the variability in the testing and training examples chosen.
3. We expect that results, other than those covering the correlation of test and train accuracy, will be similar to those of CNN-Basic MNIST or CIFAR10.
4. We expect the output layer to provide the best correlations with the test data.

Results

Figure 5.13 and Figure 5.14 show the test and train accuracies (or average reward), for MNIST and CIFAR. On both cases we see that at the selected number of epochs the intended average reward (1.0) was not reached. This is due to the limited number of epochs chosen, and the fact that reinforcement learning algorithms like DQN are sample inefficient. We also see some variation, when compared to supervised learning, with MNIST for example having a great reduction of the reward towards the end of the training.

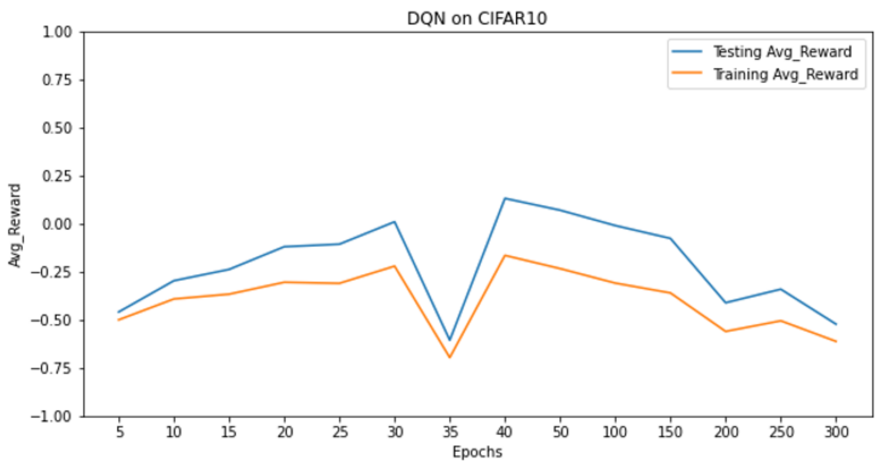


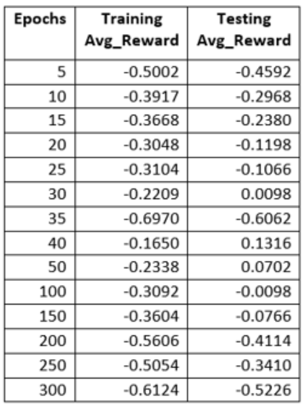


Epochs	Training Avg_Reward	Testing Avg_Reward
5	0.6988	0.7238
10	0.8559	0.8904
15	0.9144	0.9466
20	0.8158	0.8674
25	0.9450	0.9782
30	0.9110	0.9606
35	0.9177	0.97
40	0.2240	0.2164
50	0.9147	0.9718
100	-0.6374	-0.6256

Figure 5.13: DQN Accuracy on MNIST

Figure 5.15 and Figure 5.16 show how the internal representations for MNIST and CIFAR10 are mapped to a latent space. These figures show that compared to supervised learning, the learning process has not been entirely successful. We also see that for MNIST we achieve a comparatively better learning than for CIFAR10, because structures are slightly better there. Result, for the case of MNIST also let us see the oscillation during learning, with groups visually forming and dissolving.





Epochs	Training Avg_Reward	Testing Avg_Reward
5	-0.5002	-0.4592
10	-0.3917	-0.2968
15	-0.3668	-0.2380
20	-0.3048	-0.1198
25	-0.3104	-0.1066
30	-0.2209	0.0098
35	-0.6970	-0.6062
40	-0.1650	0.1316
50	-0.2338	0.0702
100	-0.3092	-0.0098
150	-0.3604	-0.0766
200	-0.5606	-0.4114
250	-0.5054	-0.3410
300	-0.6124	-0.5226

Figure 5.14: DQN Accuracy on CIFAR10

Figure 5.18, Figure 5.17, Figure 5.19, Figure 5.20, Figure 5.21, Figure 5.22, Figure 5.23 and Figure 5.24 collect the results for the clustering goodness measures on the 2 outermost layers of the DQN networks, for both datasets. For MNIST we see relatively better measurements than for CIFAR10. We see that for many epochs MNIST finds its optima around 10 clusters for both layers, with some minor oscillations in the pre-output layers. We also see that the overall goodness value is not very high, as compared to the case of supervised learning for this dataset. For CIFAR10 we see an overall failure at identifying the optimal number of clusters, and also very low goodness measures. Altogether these results suggest that for both datasets the learned structure is currently not very reliable. This is consistent with our knowledge of the limited sample efficiency of reinforcement learning.

Figure 5.25 and Figure 5.26 show the results for the correlations identified between the different clustering measures and the testing accuracy.

Discussion

Considering Figure 5.25 and the correlations between the different clustering goodness measures and the test accuracy, we find overall a very strong correlation between test and train accuracy, which produces strong correlations for both clustering goodness metrics at the different layers. This is specially interesting for our work since we find the structure of pre-output layer to be much more correlated with the test accuracy as in the case of supervised learning. Furthermore the correlation has a strong positive sign for the Silhouette coefficient, indicating that for this setup and stage of learning, the better the structure in the representation being learned, the better the expected reward.

Moving on to Figure 5.26, our results show a strong correlation between test and train accuracy. This is to be expected, since visually these metrics seem related. Furthermore, the correlation is much stronger than in the case of supervised learning. For the other correlation metrics, unfortunately our results do not achieve p-values that would give statistical reliability to the observations. As a consequence, we refrain from making conclusions on the observed correlations. Still at this stage of the training we see that the

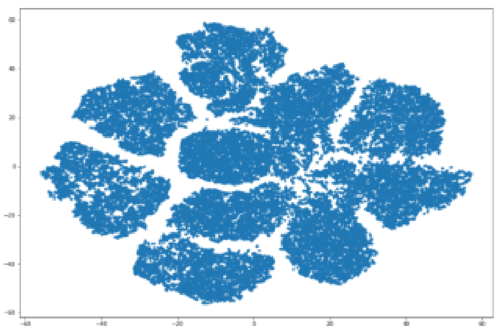
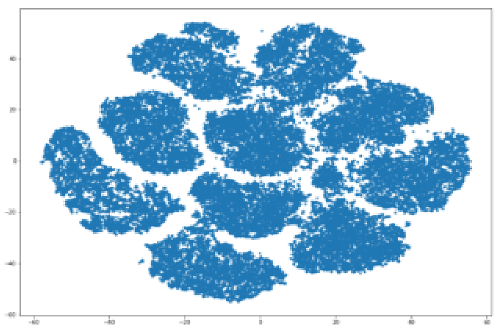
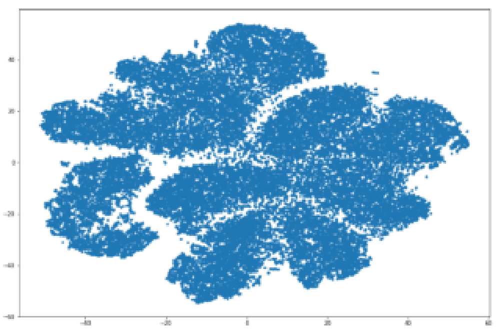
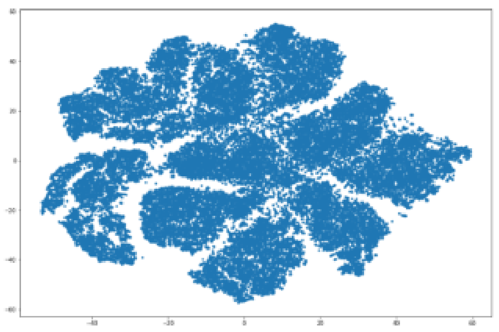
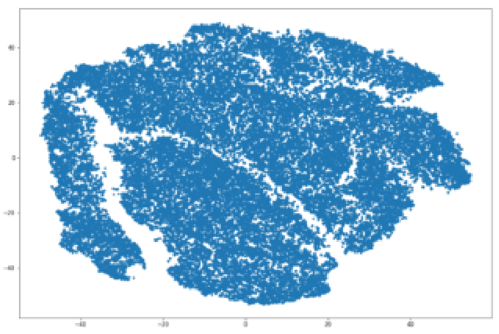
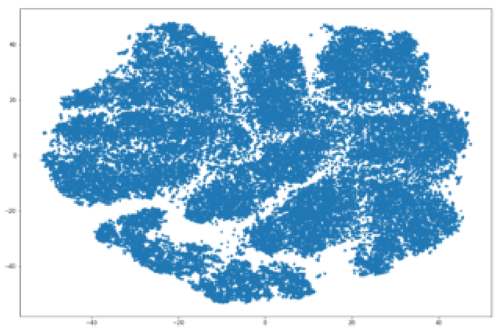
	T-SNE visualization of MNIST trained by DQN	
	Output layer	Pre-output layer
DQN 5epochs		
DQN 40 epochs		
DQN 100 epochs		

Figure 5.15: t-SNE visualization of MNIST trained by DQN

network struggles to represent the groups of different input cases on a high dimensional space, and thus no correlations seem to be merging.

Finally, considering our hypotheses, we see that they are not always verified, leading us to a correction of our expectations. Overall we observe, as expected, a lower accuracy, in contrast to supervised learning. We also see that the correlation between test and train accuracy is surprisingly quite high, making the goodness measures comparable, in terms of their correlation with test accuracy, to the train accuracy. We also find that the structure learned over the train data on all outermost layers has a correlation to the test accuracy. Finally, when comparing the output and preceeding layer, we find the output layer to have a higher degree of correlation.

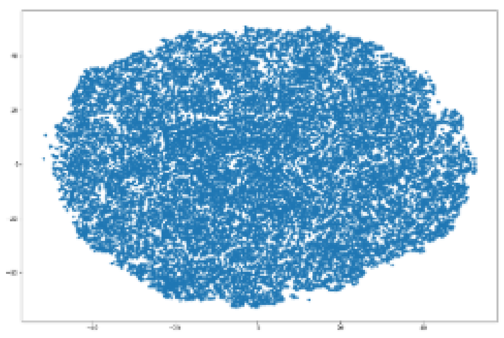
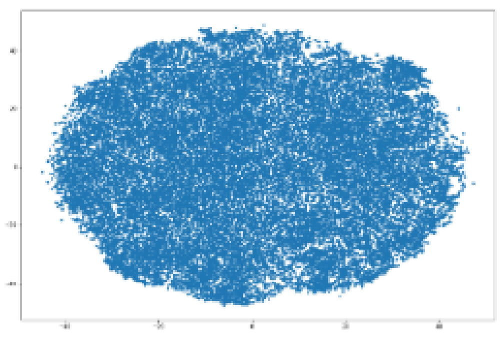
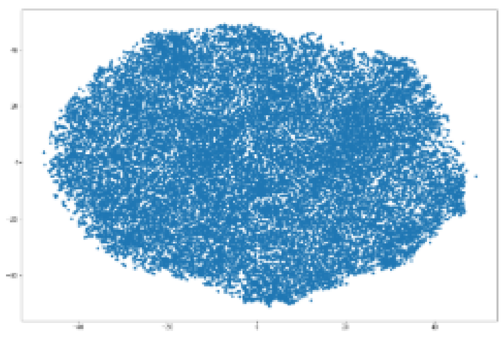
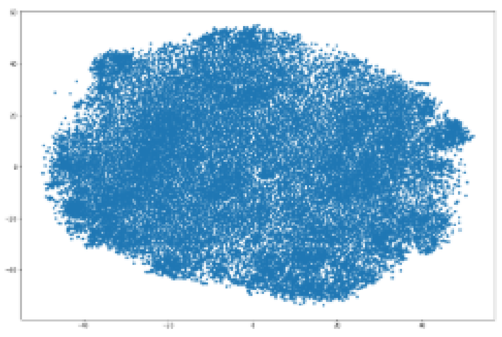
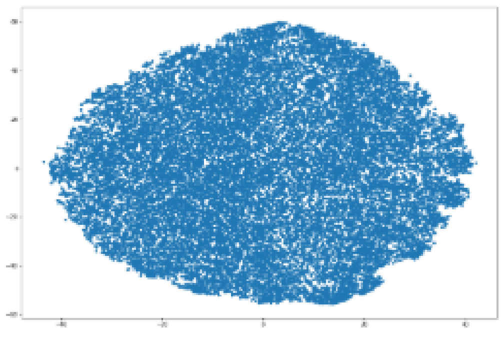
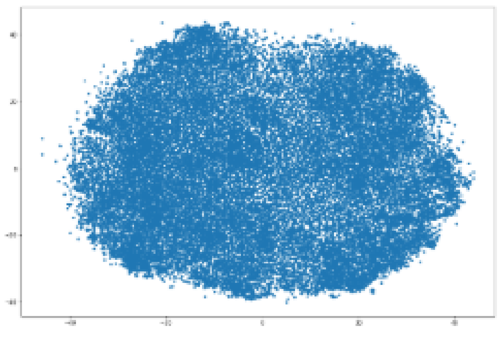
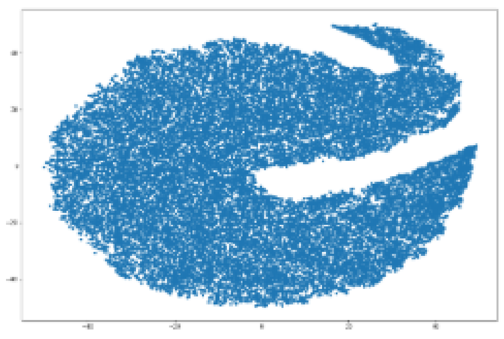
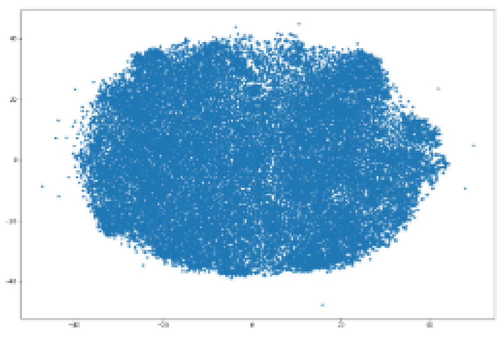
	T-SNE visualization of CIFAR10 trained by DQN	
	Output layer	Pre-output layer
DQN 5epochs		
DQN 40 epochs		
DQN 100 epochs		
DQN 300 epochs		

Figure 5.16: t-SNE visualization of CIFAR10 trained by DQN

5.4 Takeaways

In the following we summarize our main takeaways:

1. In supervised learning we find that the output layer tends to find a structure for the

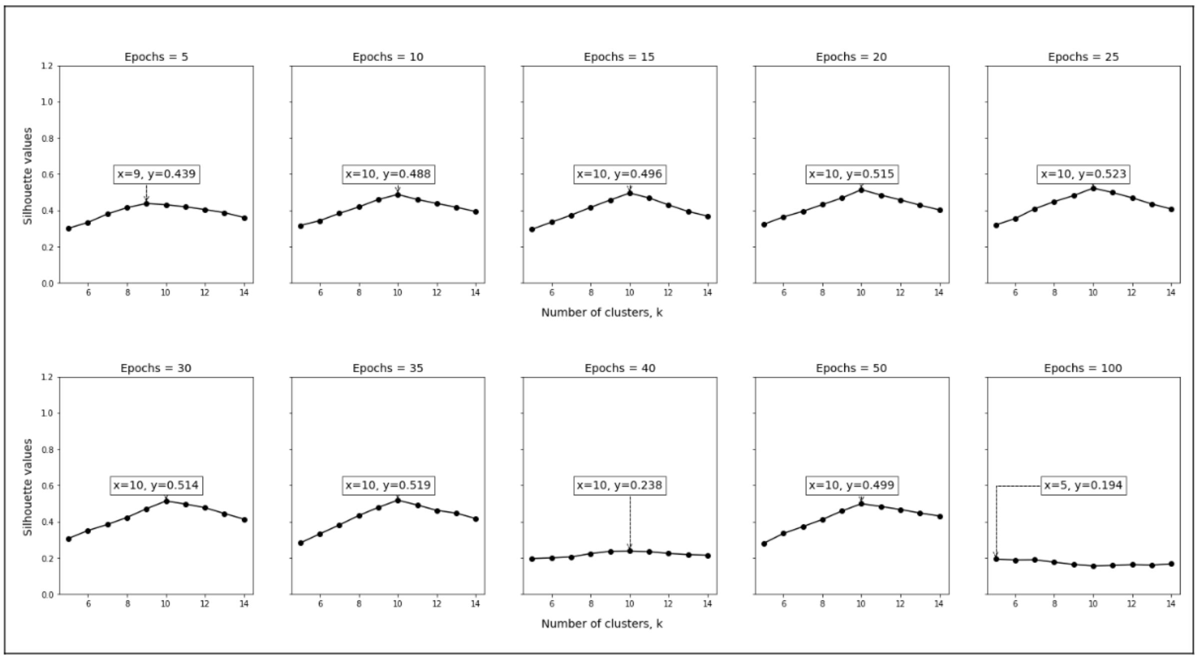


Figure 5.17: [MNIST] DQN/Output Layer/Clustering validated using Silhouette Coefficient

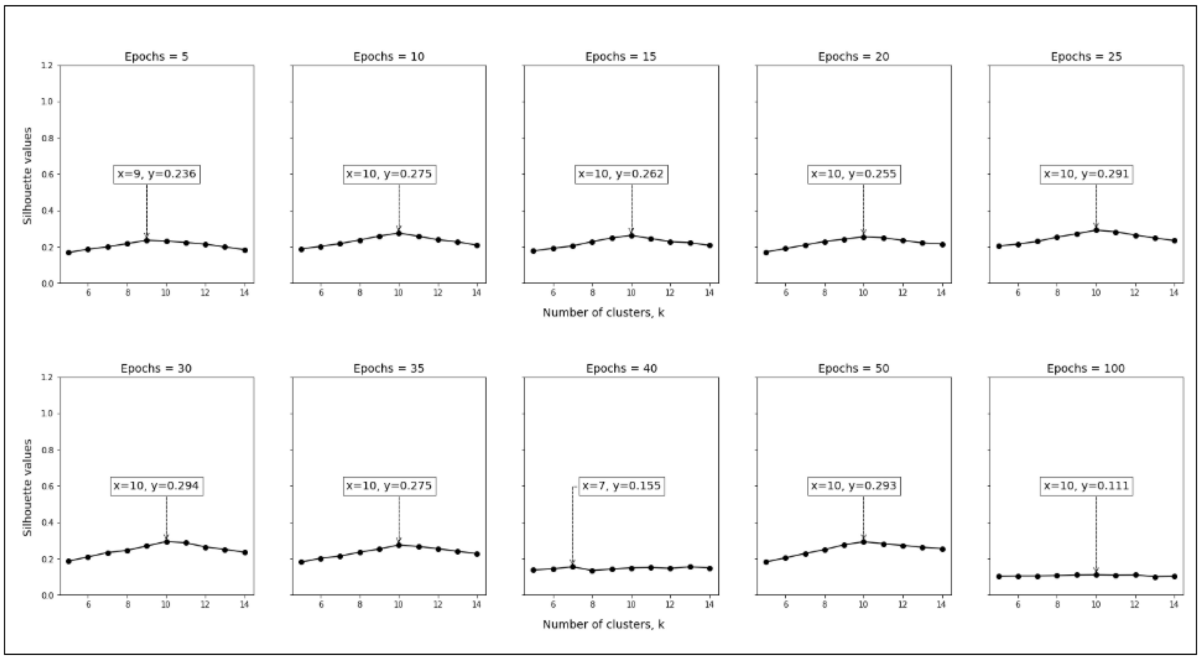


Figure 5.18: [MNIST] DQN/Pre-output/Clustering validated using Silhouette Coefficient

training data that matches the expected number of clusters/classes. For a preceding layer the number of clusters is similar but the signal might not be as clear as for the last layer.

2. The aspects of reinforcement vs. supervised learning, the success of the training process, the relation between the test and train data (e.g. presence of distribution

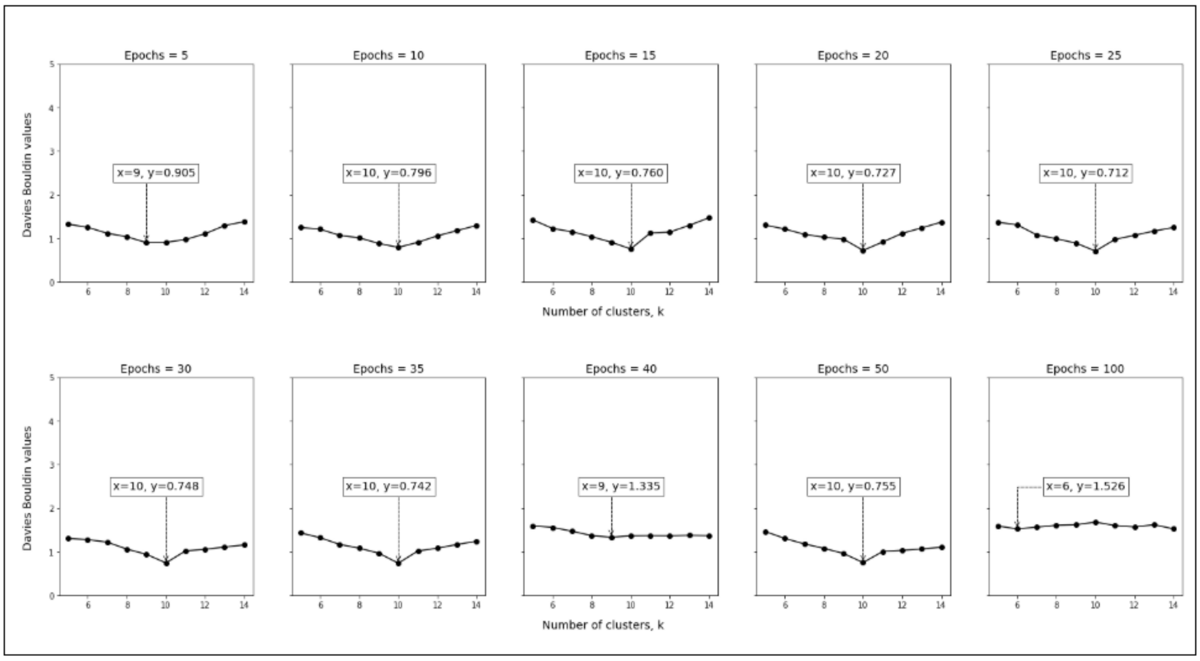


Figure 5.19: [MNIST] DQN/Output Layer/Clustering validated using Davies Bouldin

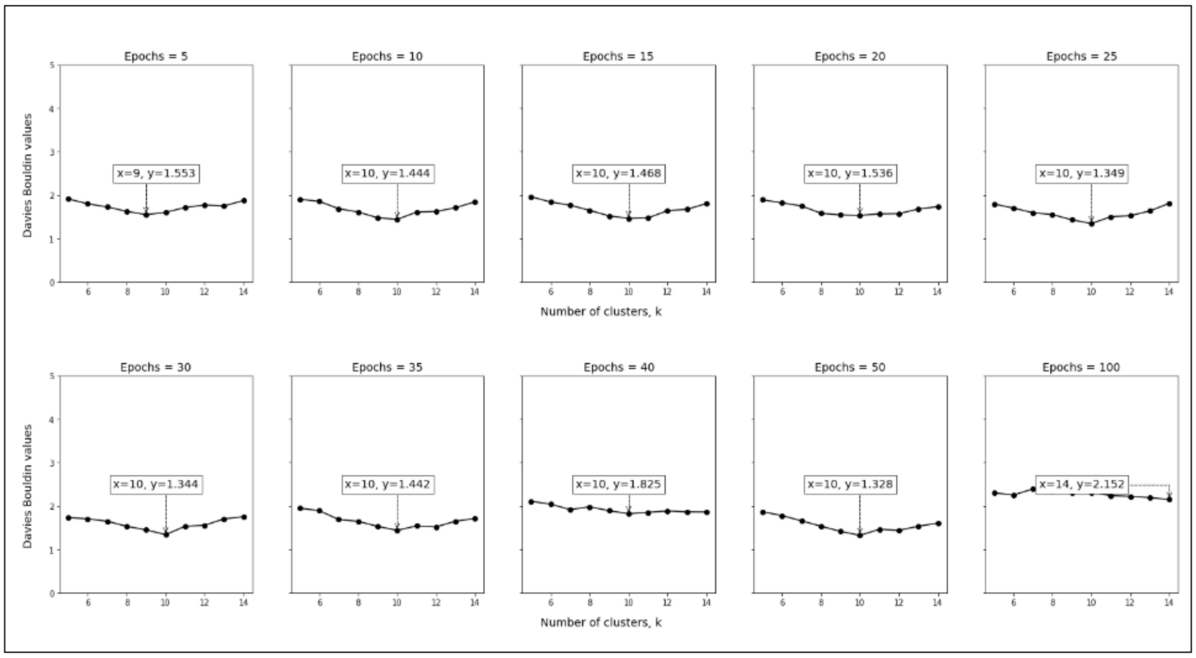


Figure 5.20: [MNIST] DQN/Pre-output/Clustering validated using Davies Bouldin

shift), and the presence or not of saturation on the training accuracy, are all aspects that have an incidence in the correlations and statistical soundness of the correlations observed between clustering goodness metrics for the last layers, and the testing accuracy. Hence these aspects need consideration, and should be treated as separate scenarios of a learning process.

3. For the case of a successful supervised learning, with saturation, we observe a statistically sound stronger positive correlation (compared to that between train and test accuracy) between the output layer clustering goodness and the test accuracy.

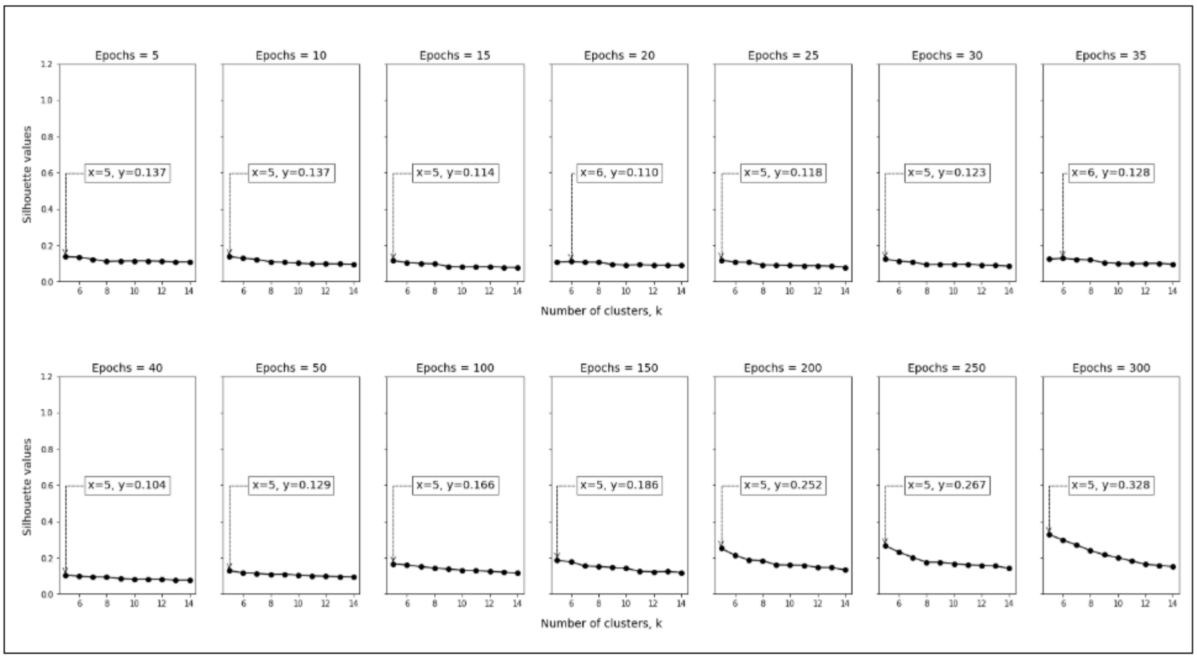


Figure 5.21: [CIFAR10] DQN/Output Layer/Clustering validated using Silhouette Coefficient

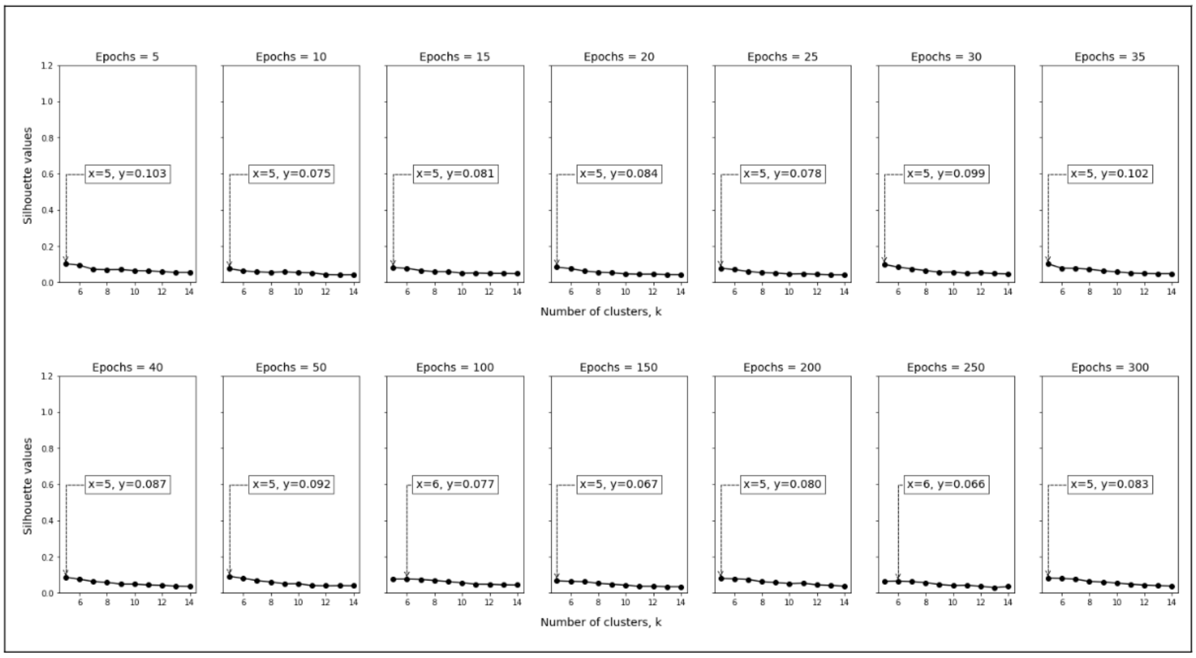


Figure 5.22: [CIFAR10] DQN/Pre-output/Clustering validated using Silhouette Coefficient

This correlation improved when considering only the case post-saturation. These results suggest that although saturation occurred, there was not a big difference between test and train data, hence no distribution shift. For this scenario we only observed satistically valid correlations when distinguishing the stages of pre- and post-saturation, for these cases we found an alternating strong negative (more structure means less testing accuracy) and strong positive correlations for the different

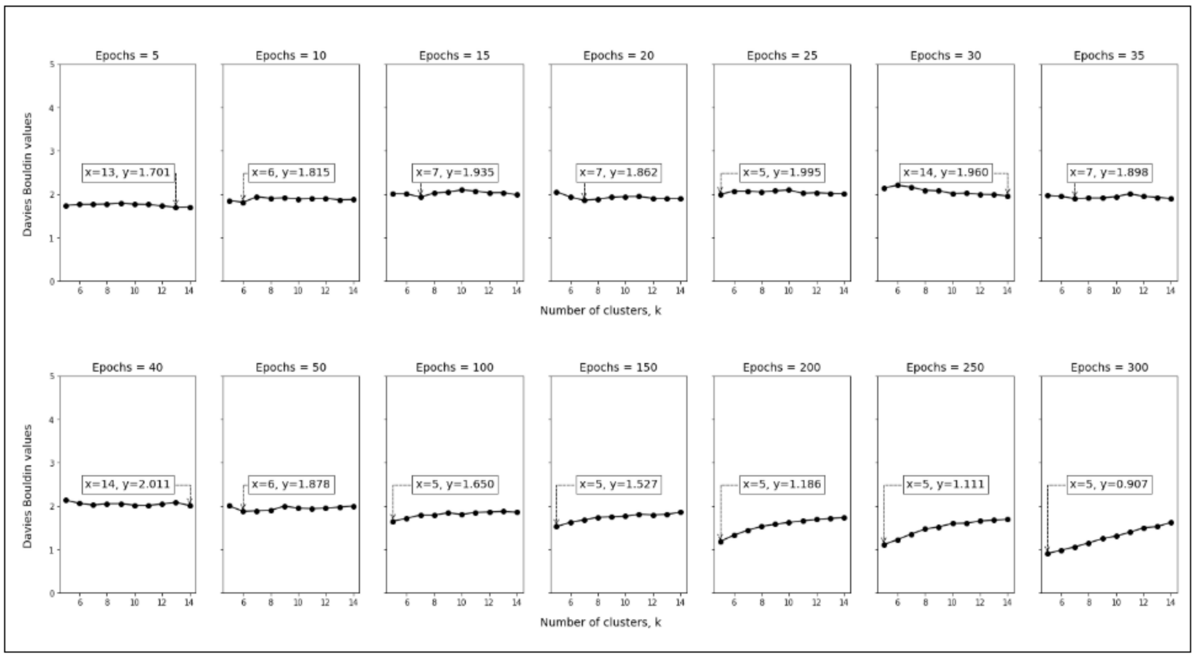


Figure 5.23: [CIFAR10] DQN/Output Layer/Clustering validated using Davies Bouldin

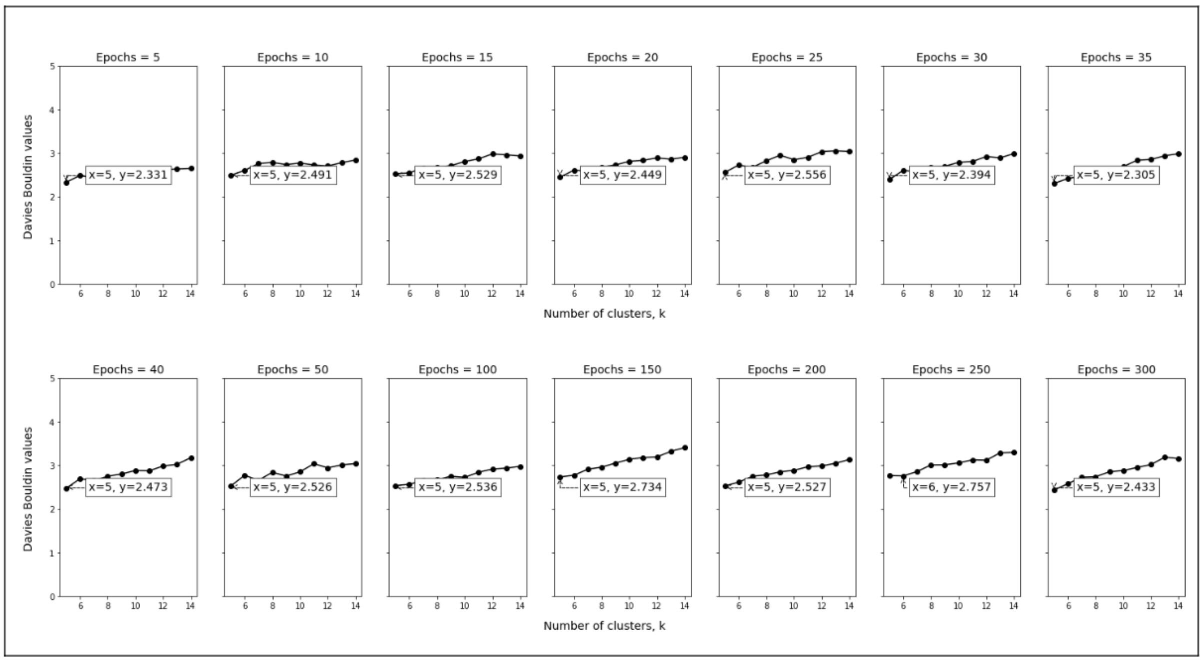
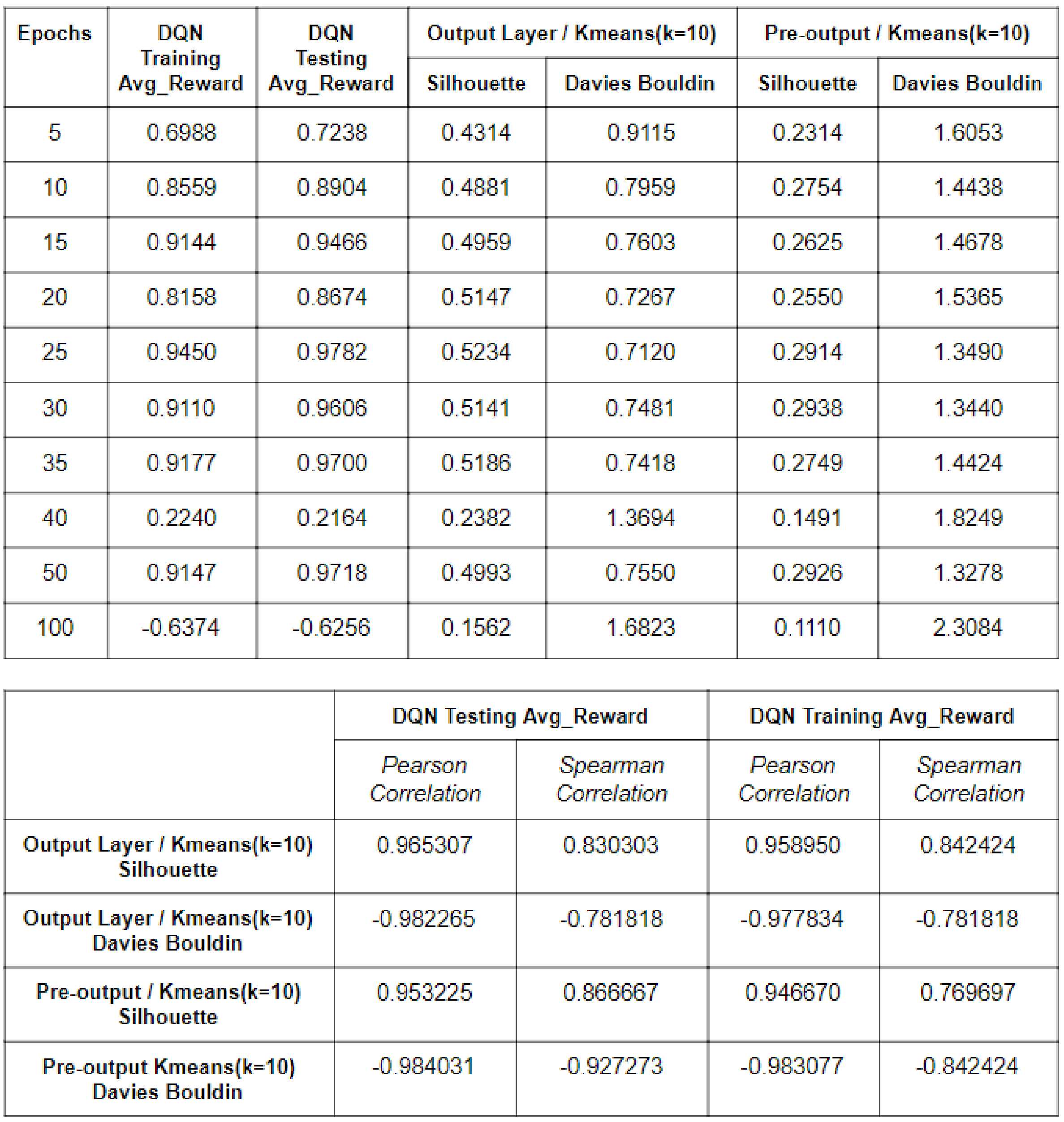


Figure 5.24: [CIFAR10] DQN/Pre-output/Clustering validated using Davies Bouldin

stages. This suggests that after saturation the changes in structure of the pre-output layer might be relevant for the testing accuracy.

4. For the case of a successful supervised learning, with saturation but the possible presence of strong differences between test and train data, we find limited negative correlations between test and train accuracy (i.e., improving on one data split damages the performance on the other). These findings in turn lead to more moderate correlations between the clustering goodness and the test accuracy. This learning



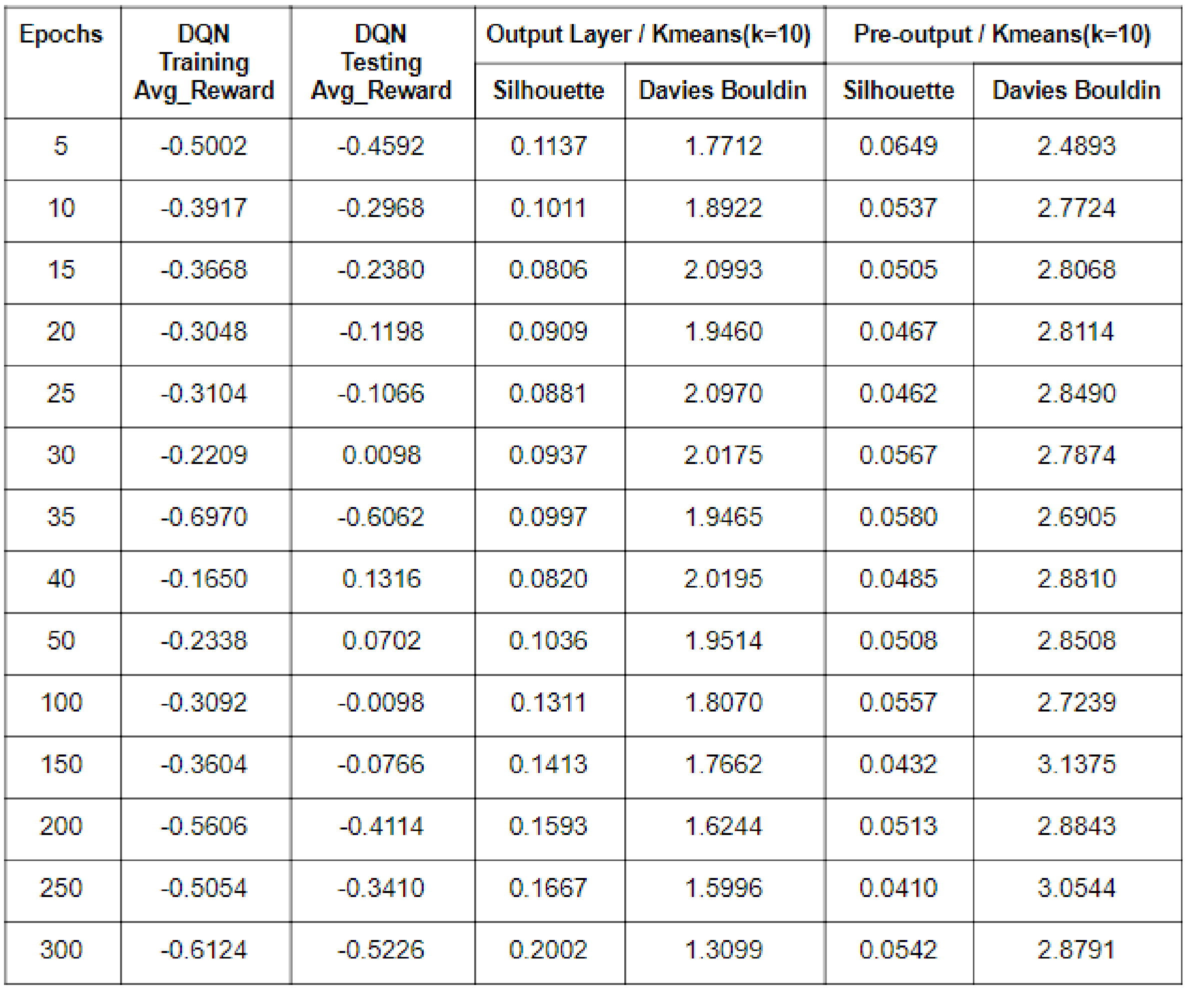
Epochs	DQN Training Avg_Reward	DQN Testing Avg_Reward	Output Layer / Kmeans(k=10)		Pre-output / Kmeans(k=10)	
			Silhouette	Davies Bouldin	Silhouette	Davies Bouldin
5	0.6988	0.7238	0.4314	0.9115	0.2314	1.6053
10	0.8559	0.8904	0.4881	0.7959	0.2754	1.4438
15	0.9144	0.9466	0.4959	0.7603	0.2625	1.4678
20	0.8158	0.8674	0.5147	0.7267	0.2550	1.5365
25	0.9450	0.9782	0.5234	0.7120	0.2914	1.3490
30	0.9110	0.9606	0.5141	0.7481	0.2938	1.3440
35	0.9177	0.9700	0.5186	0.7418	0.2749	1.4424
40	0.2240	0.2164	0.2382	1.3694	0.1491	1.8249
50	0.9147	0.9718	0.4993	0.7550	0.2926	1.3278
100	-0.6374	-0.6256	0.1562	1.6823	0.1110	2.3084

	DQN Testing Avg_Reward		DQN Training Avg_Reward	
	Pearson Correlation	*Spearman Correlation*	*Pearson Correlation*	*Spearman Correlation*
Output Layer / Kmeans(k=10) Silhouette	0.965307	0.830303	0.958950	0.842424
Output Layer / Kmeans(k=10) Davies Bouldin	-0.982265	-0.781818	-0.977834	-0.781818
Pre-output / Kmeans(k=10) Silhouette	0.953225	0.866667	0.946670	0.769697
Pre-output Kmeans(k=10) Davies Bouldin	-0.984031	-0.927273	-0.983077	-0.842424

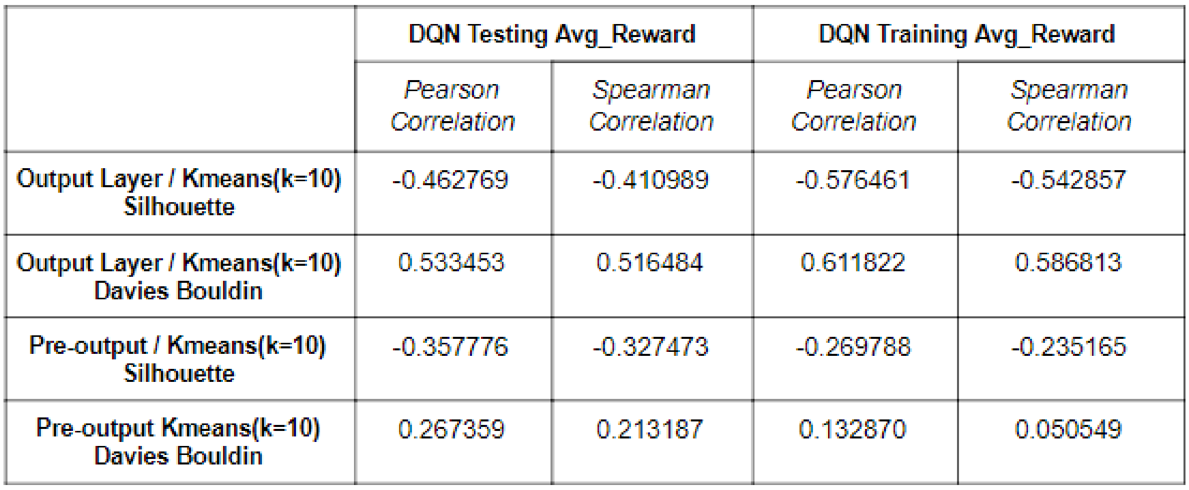
Figure 5.25: [MNIST] Correlation between DQN average reward and clustering goodness of two last layers outputs

scenario also produces a need for more data gathering, as p-values would might tend to be reduced for calculating correlations.

5. We experimented with the use of regularization techniques in supervised learning, finding that these techniques made all scenarios behave similarly to the case of a successful supervised learning, without saturation but with a good relation between test and train data (i.e. no distributional shift). These techniques also contribute to the statistical soundness of the evaluation. Results show then a strong correlation between test and train accuracy, with also strong correlations between the clustering goodness metrics of the output layer (more structure is better). There is also strong but negative (more structure is not better) between the clustering goodness metrics



Epochs	DQN Training Avg_Reward	DQN Testing Avg_Reward	Output Layer / Kmeans(k=10)		Pre-output / Kmeans(k=10)	
			Silhouette	Davies Bouldin	Silhouette	Davies Bouldin
5	-0.5002	-0.4592	0.1137	1.7712	0.0649	2.4893
10	-0.3917	-0.2968	0.1011	1.8922	0.0537	2.7724
15	-0.3668	-0.2380	0.0806	2.0993	0.0505	2.8068
20	-0.3048	-0.1198	0.0909	1.9460	0.0467	2.8114
25	-0.3104	-0.1066	0.0881	2.0970	0.0462	2.8490
30	-0.2209	0.0098	0.0937	2.0175	0.0567	2.7874
35	-0.6970	-0.6062	0.0997	1.9465	0.0580	2.6905
40	-0.1650	0.1316	0.0820	2.0195	0.0485	2.8810
50	-0.2338	0.0702	0.1036	1.9514	0.0508	2.8508
100	-0.3092	-0.0098	0.1311	1.8070	0.0557	2.7239
150	-0.3604	-0.0766	0.1413	1.7662	0.0432	3.1375
200	-0.5606	-0.4114	0.1593	1.6244	0.0513	2.8843
250	-0.5054	-0.3410	0.1667	1.5996	0.0410	3.0544
300	-0.6124	-0.5226	0.2002	1.3099	0.0542	2.8791



	DQN Testing Avg_Reward		DQN Training Avg_Reward	
	Pearson Correlation	*Spearman Correlation*	*Pearson Correlation*	*Spearman Correlation*
Output Layer / Kmeans(k=10) Silhouette	-0.462769	-0.410989	-0.576461	-0.542857
Output Layer / Kmeans(k=10) Davies Bouldin	0.533453	0.516484	0.611822	0.586813
Pre-output / Kmeans(k=10) Silhouette	-0.357776	-0.327473	-0.269788	-0.235165
Pre-output Kmeans(k=10) Davies Bouldin	0.267359	0.213187	0.132870	0.050549

Figure 5.26: [CIFAR10] Correlation between DQN average reward and clustering goodness of two last layers outputs

of the pre-output layer and the test accuracy. These observations are consistent with our expectation of how regularization (e.g. droupout) contributes to the training process: by making more paths in the pre-output layers.

6. Reinforcement learning shows a higher correlation between test and train accuracy,

at least in the scenarios that we studied, which involve no saturation, little differences between test and train (from the batch formation during sampling) and an unsuccessful learning situation. Our early results seem to also indicate that the structure learned by all the outermost layers is highly correlated with the test accuracy. More studies are needed in this area.

Other than the core takeaways that stem from our research questions, we also have some methodological takeaways that we have learned along the way:

1. From our experiments we find cases (e.g. CNN-Basic CIFAR10 pre-output layer) where the t-SNE visualizations do not map immediately to clustering goodness metrics. An easy to distinguish visual grouping in 2 dimensions might internally still produce poor clustering metrics. So even though t-SNE gives insights that are easy to grasp, it does not necessarily portray the goodness of clustering in high dimensions. This observation indicates that both the goodness measures and the t-SNE should be used for better studying the network.
2. In terms of the clustering goodness metric used, we find that the Davies Bouldin metric can be empirically a less robust indicator of the clustering goodness, as compared to the Silhouette coefficient. This observation is based on experiments where we saw more variability in results with this metric.
3. For evaluating the correlations between different clustering metrics, we find that statistical confidence tests are very necessary.

5.5 Summary

In this chapter, we presented the evaluation for our work covering supervised learning and reinforcement learning scenarios. We reviewed the correlation between structures learned at different layers and test accuracy, following by the zoom-in into the role of the clustering quality metric and correlation metrics. Finally, we identified cases where the correlation was working, hence aiding in model understanding, and cases where it was not. In the next chapter, we summarize our work and we show areas for further research.

6 Conclusion and Future Work

6.1 Conclusion

Deep reinforcement learning is promising method to build everyday solutions in computer systems that are capable to learn from experiences. As per our observation, most deep reinforcement learning research focuses on reporting the rewards obtained for a model on different tasks and often with different learning configurations, but does not report well internal aspects of the models during learning, such as considering how the input space is mapped to internal representations. Thus, in our work, we took as motivation that for progressing the field of automated machine learning, it becomes ever more important to enable a professional work on the machine learning models in use, based on a careful understanding on the internal dynamics of the model being learned. To assist in this, we start our work with a short review on model-specific and model-agnostic methods to understand machine learning models. And we take a deeper look on the method that requires human involvement like prototypes and criticisms explanation.

We devise a series of experiments to analyze the learning of supervised and reinforcement learning scenarios, for image classification on MNIST and CIFAR10 data. We design a setup where for each training process we take snapshots of the internal representations of the training and test data learned thus far on the output and pre-output layer. Following this, we study the representations and we seek to understand how the structure learned correlates to the training accuracy, over the different scenarios. Below listed our main takeaways from the experiments corresponding to our research questions in 3:

1. In CNN, the structure of training data found by the output layer matches the expected number of clusters/classes. In the preceding layer, the number of clusters is similar but the signal might not be as clear as for the last layer.

2. The aspects of reinforcement vs. supervised learning, the success of the training process, the relation between the test and train data (e.g. presence of distribution shift), and the presence or not of saturation on the training accuracy, are all aspects that have an incidence in the correlations and statistical soundness of the correlations observed between clustering goodness metrics for the last layers and the testing accuracy. They should be treated as separate scenarios of a learning process.

3. For the case of a successful supervised learning, with saturation, we observe a statistically stronger positive correlation between the clustering quality of output layer and the test accuracy, compared to that between train and test accuracy. This correlation improved when considering only the case post-saturation. These results suggest that although saturation occurred, there was not a big difference between test

and train data, hence no distribution shift. For this scenario we only observed statistically valid correlations when distinguishing the stages of pre- and post-saturation, for these cases we found an alternating strong negative (more structure means less testing accuracy) and strong positive correlations for the different stages. This suggests that after saturation the changes in structure of the pre-output layer might be relevant for the testing accuracy.

4. For the case of a successful supervised learning, with saturation but the possible presence of strong differences between test and train data, we find limited negative correlations between test and train accuracy (i.e., improving on one data split damages the performance on the other). These findings in turn lead to more moderate correlations between the clustering goodness and the test accuracy. This learning scenario also produces a need for more data gathering, as p-values would might tend to be reduced for calculating correlations.

5. We experimented with the use of regularization techniques in supervised learning, finding that these techniques made all scenarios behave similarly to the case of a successful supervised learning, without saturation but with a good relation between test and train data (i.e. no distributional shift). These techniques also contribute to the statistical soundness of the evaluation. Results show then a strong correlation between test and train accuracy, with also strong correlations between the clustering goodness metrics of the output layer (more structure is better). There is also strong but negative (more structure is not better) between the clustering goodness metrics of the pre-output layer and the test accuracy. These observations are consistent with our expectation of how regularization (e.g. droupout) contributes to the training process: by making more paths in the pre-output layers.

6. Reinforcement learning shows a higher correlation between test and train accuracy, at least in the scenarios that we studied, which involve no saturation, little differences between test and train (from the batch formation during sampling) and an unsuccessful learning situation. Our early results seem to also indicate that the structure learned by all the outermost layers is highly correlated with the test accuracy. More studies are needed in this area.

Other than the core takeaways that stem from our research questions, we also have some methodological takeaways that we have learned along the way:

1. From our experiments we find cases (e.g. CNN-Basic CIFAR10 pre-output layer) where the t-SNE visualizations do not map immediately to clustering goodness metrics. An easy to distinguish visual grouping in 2 dimensions might internally still produce poor clustering metrics. So even though t-SNE gives insights that are easy to grasp, it does not necessarily portray the goodness of clustering in high dimensions. This observation indicates that both the goodness measures and the t-SNE should be used for better studying the network.

2. In terms of the clustering goodness metric used, we find that the Davies Bouldin metric can be empirically a less robust indicator of the clustering goodness, as compared to the Silhouette coefficient. This observation is based on experiments where we saw more variability in results with this metric.

3. For evaluating the correlations between different clustering metrics, we find that statistical confidence tests are very necessary.

6.2 Future work

In the future, there are multiple promising research directions regarding interpretable deep reinforcement learning that we desire and recommend to other enthusiasts to conduct further researches:

- We want to consider the structure being learned by distinguishing prototypes and criticisms in the clusters learned in an automatic manner by adopting the MMD-critic algorithm. We also would need to consider an approach to correlated this knowledge to potential test accuracy/rewards.
- For gaining more insights from the t-SNE study, it would be important to also include some source images as part of the visualization, and to consider rewards or accuracy information while mapping the data points.
- Concerning supervised learning, we would like to evaluate more competitive neural network models, like EfficientNet, and use more challenging datasets like subsets of ImageNet or Fashion MNIST.
- For reinforcement learning we consider it important to also pursue this line of research on more complex environments involving at least state changes, to understand better how the neural network structures can help model understanding. Additional models to DQN would be of great interest.

Bibliography

[AB18] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.

[BC17] Or Biran and Courtenay V. Cotton. Explanation and justification in machine learning : A survey or. 2017.

[Bel03] Richard Ernest Bellman. *Dynamic Programming*. Dover Publications, Inc., USA, 2003.

[BFOS17] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and regression trees*. 01 2017. doi:10.1201/9781315139470.

[BG18] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *FAT*, 2018.

[BKD20] Adam Byerly, Tatiana Kalganova, and Ian Dear. A branching and merging convolutional network with homogeneous filter capsules. *arXiv preprint arXiv:2001.09136*, 2020.

[BR18] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, Dec 2018. URL: http://dx.doi.org/10.1016/j.patcog.2018.07.023, doi:10.1016/j.patcog.2018.07.023.

[Bro18] Meredith Broussard. *Artificial Unintelligence: How Computers Misunderstand the World*. The MIT Press, 04 2018. doi:10.7551/mitpress/11022.001.0001.

[Bur19] Andriy Burkov. *The hundred-page machine learning book*, volume 1. Andriy Burkov Quebec City, Can., 2019.

[BZK$^+$17a] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3319–3327, 2017.

[BZK$^+$17b] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations, 2017. arXiv:1704.05796.

[CMS12] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3642–3649. IEEE, 2012.

[CPC19] Diogo Carvalho, Eduardo Pereira, and Jaime Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8:832, 07 2019. doi:10.3390/electronics8080832.

[DB79] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[DD13] Michel Marie Deza and Elena Deza. *Metrics on Normed Structures*, pages 89–99. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-30958-8_5.

[DDS+09] J. Deng, Wei Dong, R. Socher, L. Li, K. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[DVK17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017. arXiv:1702.08608.

[FCF17] Kelwin Fernandes, Jaime Cardoso, and Jessica Fernández. Transfer learning with partial observability applied to cervical cancer screening. pages 243–250, 05 2017. doi:10.1007/978-3-319-58838-4_27.

[FRD18] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously, 2018. arXiv:1801.01489.

[Fri00] Jerome Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29, 2000. doi:10.1214/aos/1013203451.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[GBR+08] Arthur Gretton, Karsten Borgwardt, Malte J. Rasch, Bernhard Scholkopf, and Alexander J. Smola. A kernel method for the two-sample problem, 2008. arXiv:0805.2368.

[GBY+18] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89, 2018.

[GCM08] Derek Greene, Pádraig Cunningham, and Rudolf Mayer. *Unsupervised Learning and Clustering*, pages 51–90. Springer Berlin Heidelberg, 2008.

[GF17] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, Oct 2017. URL: http://dx.doi.org/10.1609/aimag.v38i3.2741, doi:10.1609/aimag.v38i3.2741.

[GKBP13] Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation, 2013. arXiv:1309.6392.

[Gle16] M. Gleicher. A framework for considering comprehensibility in modeling. *Big data*, 4 2:75–88, 2016.

[GWFM$^+$13] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International conference on machine learning*, pages 1319–1327, 2013.

[HBC16] Sara Hajian, Francesco Bonchi, and Carlos Castillo. Algorithmic bias: From discrimination discovery to fairness-aware data mining. pages 2125–2126, 08 2016. doi:10.1145/2939672.2945386.

[HBV01] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17, 10 2001. doi:10.1023/A:1012801612483.

[HMvH$^+$17] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning, 2017. arXiv:1710.02298.

[HR03] Geoffrey Hinton and Sam Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15*, pages 833–840. MIT Press, 2003.

[HS15] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps, 2015. arXiv:1507.06527.

[HTS$^+$17] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments, 2017. arXiv:1707.02286.

[IJW$^+$19] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Morgane Lustman, Vince Gatto, Paul Covington, Jim McFadden, Tushar Chandra, and Craig Boutilier. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology, 2019. arXiv:1905.12767.

[KBZ$^+$19] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 2019.

[KCB$^+$18] Matthieu Komorowski, Leo Anthony Celi, Omar Badawi, Anthony C. Gordon, and Aldo A. Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine*, 24:1716–1720, 2018.

[Kir08] Wilhelm Kirch, editor. *Pearson's Correlation Coefficient*, pages 1090–1091. 2008.

[KKK16] Been Kim, Rajiv Khanna, and Oluwasanmi Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2288–2296, Red Hook, NY, USA, 2016. Curran Associates Inc.

[KL17] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions, 2017. arXiv:1703.04730.

[Kri12] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[KYG+18] Sanjay Krishnan, Zongheng Yang, Ken Goldberg, Joseph Hellerstein, and Ion Stoica. Learning to optimize join queries with deep reinforcement learning, 2018. arXiv:1808.03196.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LeC89] Y. LeCun. Generalization and network design strategies. 1989.

[Llo82] S. P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136, 1982.

[LLX+13] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu, and S. Wu. Understanding and enhancement of internal clustering validation measures. *IEEE Transactions on Cybernetics*, 43(3):982–994, 2013.

[Loe94] George Loewenstein. The psychology of curiosity: A review and reinterpretation. *Psychological Bulletin*, 116:75–98, 07 1994. doi:10.1037/0033-2909.116.1.75.

[LZLG19] Guoliang Li, Xuanhe Zhou, Shifu Li, and Bo Gao. Qtune: A query-aware database tuning system with deep reinforcement learning. *PVLDB*, 12:2118–2130, 2019.

[MBM+16] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016. arXiv:1602.01783.

[MH08a] L. V. D. Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[MH08b] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. arXiv:1312.5602.

[MKS+15a] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–33, 2015. doi:10.1038/nature14236.

[MKS+15b] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[Mol19a] Christoph Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.

[Mol19b] Christoph Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.

[MP18] Ryan Marcus and Olga Papaemmanouil. Deep reinforcement learning for join order enumeration. *Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management - aiDM'18*, 2018. `doi:10.1145/3211954.3211957`.

[MPH09] L. V. D. Maaten, E. Postma, and J. V. D. Herik. Dimensionality reduction: A comparative review. 2009.

[MS88] Johanna D Moore and William R Swartout. Explanation in expert systemss: A survey. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST, 1988.

[MSK+19] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, Oct 2019. URL: http://dx.doi.org/10.1073/pnas.1900654116, `doi:10.1073/pnas.1900654116`.

[NNM96] Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia object image library (coil-20. Technical report, 1996.

[OBB+19] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 2019. `arXiv:1912.06680`.

[ODH+20] Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado Van Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020. URL: https://openreview.net/forum?id=rygf-kSYwH.

[OMS17a] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization. `doi:10.23915/distill.00007`.

[OMS17b] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization. `doi:10.23915/distill.00007`.

[Pea05] K. Pearson. I. mathematical contributions to the theory of evolution. —xiv. on the general theory of skew correlation and non-linear regression. *Drapers' Company Research Memoirs, Biometric Series II*, 1905.

[Pet06] Slobodan Petrovic. A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters. 2006.

[Pos91] Ross Posnock. The trial of curiosity: Henry james, william james, and the challenge of modernity. 1991.

[Qui86] J. R. Quinlan. Induction of decision trees. *MACH. LEARN*, 1:81–106, 1986.

[RGHL09] Martin Riedmiller, Thomas Gabel, Roland Hafner, and Sascha Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27, 2009.

[Rou87] Peter Rousseeuw. Rousseeuw, p.j.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. comput. appl. math. 20, 53-65. *Journal of Computational and Applied Mathematics*, 20:53–65, 11 1987. doi:10.1016/0377-0427(87)90125-7.

[Rou13] Mathieu Rouaud. *Probability, Statistics and Estimation.* 2013.

[RS01] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science (New York, N.Y.)*, 290:2323–6, 01 2001. doi:10.1126/science.290.5500.2323.

[RSG16a] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning, 2016. arXiv:1606.05386.

[RSG16b] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *ArXiv*, abs/1606.05386, 2016.

[RSG16c] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016. arXiv:1602.04938.

[Sam69] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.

[SD78] B. Streitberg and W. Daniel. Applied nonparametric statistics. 1978.

[Sha53] Lloyd Shapley. S. 1953."a value for n-person games.". *Contributions to the Theory of Games*, pages 31–40, 1953.

[SHK$^+$14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[SLM$^+$15] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2015. arXiv:1502.05477.

[SM19] Wojciech Samek and Klaus-Robert Müller. *Towards Explainable Artificial Intelligence*, pages 5–22. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-030-28954-6_1.

[SMV+19] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Hansen, and Klaus-Robert Müller. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. 01 2019. doi:10.1007/978-3-030-28954-6.

[Spe04] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904.

[SQAS15] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2015. arXiv:1511.05952.

[SVM14] C. O. S. Sorzano, J. Vargas, and A. Pascual Montano. A survey of dimensionality reduction techniques, 2014. arXiv:1403.2877.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. arXiv:1707.06347.

[SWY+15] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[TSJ08] Kostas Tzoumas, Timos Sellis, and Christian Jensen. A reinforcement learning approach for adaptive query processing. 2008.

[TSL00] Joshua Tenenbaum, Vin Silva, and John Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 01 2000.

[Vel19] Alfredo Vellido. The importance of interpretability and visualization in machine learning for applications in medicine and health care. *Neural Computing and Applications*, pages 1–15, 02 2019. doi:10.1007/s00521-019-04051-w.

[vHGS15] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015. arXiv:1509.06461.

[vMSB] William van Melle, Edward H Shortliffe, and Bruce G Buchanan. Emycin: A knowledge engineer's tool for constructing rule-based expert systems.

[Wri21] S. Wright. Correlation and causation. *Journal of Agricultural Research*, 20(7):557–585, 1921.

[WSH+15] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning, 2015. arXiv:1511.06581.

[Zha10] Xinhua Zhang. *Regularization*, pages 845–849. Springer US, Boston, MA, 2010. doi:10.1007/978-0-387-30164-8_712.

[ZLR+19] Ji Zhang, Li Liu, Minwei Ran, Zekang Li, Yu Liu, Ke Zhou, Guoliang Li, Zhili Xiao, Bin Cheng, Jiashu Xing, Yangtao Wang, and Tianheng Cheng. An end-to-end automatic cloud database tuning system using deep reinforcement learning. pages 415–432, 2019. doi:10.1145/3299869.3300085.