UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Physics and Technology

Leveraging Network Science for the Exploration of Deep Learning Latent Representations

Iver Schei Nørve

FYS-3941 Master's thesis in applied physics and mathematics 30 SP

December 2023

"*In any field find the strangest thing and then explore it*"

-John Archibald Wheeler

“Simplicity is prerequisite for reliability.”
–Edsger Dijkstra

“A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.”
–Alan Turing

Abstract

Deep Neural networks has pushed the boundary for what is achievable in the field of machine learning. At its core the Neural network maps data from an input space, to a highly abstract latent space. The representations of these latent mappings are critical for the neural networks ability to perform the task it is given. However critical, our knowledge on these abstract representations in the latent space is highly restricted. Thus finding new approaches and methods to explore the latent space is needed.

As the function constricting the latent representation is unknown, we propose some new network-science based tools for exploring the latent space. By exploring the nearest neighbour graph of the samples in the latent representations, we observe clusters consisting of samples from the same class, regions with samples from a mix of classes, as well as hub regions. Through exploring the latent representations robustness to perturbation as well as the similarity between different latent representations, we find that it is crucial to consider a suitable number of neighbours for constructing the graph structure, as well as choosing an appropriate similarity measure depending on the scope of the desired property to observe. Further, this nearest neighbour representation of the latent space can be aggregated to construct the class graph, a tool for observing high level relational information on the classes embedding in the latent space.

In addition, this thesis explores the iterative pruning method known as the lottery ticket hypothesis [Frankle and Carbin, 2019]. Our exploration considers the evolution of the latent space over the pruning iterations. Through this exploration we discover that the latent representation of the sparse sub network found by the 'winning ticket' initialisation converges towards a distinct representation in the latent space that is different from the unpruned model it originated from. If so, this finding could indicate that the hypothesis of a 'winning ticket' is inaccurate.

Acknowledgements

First and foremost I would like to extend my deepest gratitude towards my supervisor for this thesis, Benjamin Ricaud. Thank you for your excellent guidance, informative discussions, your patience and confidence in this project. But most important of all, thank you for keeping me motivated through your own curiosity and passion for the project.

In addition to this I would like to thank my entire family for supporting me, not only during this stressful year, but throughout my entire degree, especially to my father Ove for proof reading my thesis. A special acknowledgement goes out to my mother Ildrid, and my brother Arthur. Ildrid, your selflessness and care for others stands as an inspiration for me in my everyday life. Arthur, you have been great source of inspiration throughout my academic pursuits, know that you have inspired me greatly in the way you achieve your goals.

Lastly I would like to thank all my friends from my old program Lektor 8-13 for helping me keep (some of) my sanity during these five and a half years. Similarly, I would like to thank my current classmates Tobias and Christian for the fun lighthearted banter, and for including me in your group when I switched programs. Lastly, A special thanks goes out to my best friend Sander, for supporting me no matter the challenge, know that your friendship is invaluable to me.

Contents

List of Figures

List of Tables

1

Introduction

In the field of artificial intelligence and machine learning, the introduction of deep neural networks has increased the models capabilities to solve a wide variety of complex problems. Ever since the introduction of the LeNet [Lecun et al., 1998] convolutional neural network architecture, convolutional architectures has increased in size, depth and complexity. In the midst of these deep architectures we find what is known as the latent space. This is where the model has encoded and extracted information of the input data, and mapped it to an abstract representation. Ideally these abstract representations capture meaningful representation of the data, however, it is not possible to interpret these encoded representations in a direct manner. As these encodings are results of the internal mapping of the network architecture, their dimensionality depends on the structure of the network and the task it is solving. However, for classification tasks it is common that the latent space where the encodings reside lies in a higher dimensional space.

These abstract feature vectors residing in the latent space forms the core of these deep neural networks, as these representations are critical for the models ability to perform its task. However, these abstract latent space representations are largely unknown. New approaches and tools are needed to understand the properties of the latent space. In terms of classification tasks, a good encoding in the latent space is crucial for both the accuracy, as well as the reliability of a particular model.

Many deep learning models made for classification tasks such as the Resnet architecture [He et al., 2016], maps the input data to a high dimensional latent space. When working in such high dimensional spaces, a set of considerations must be taken [Aggarwal et al., 2001] such as the selection of an appropriate distance metric. The graph of nearest neighbours constructed by these high dimensional embeddings could be a useful tool for exploring the latent space as the graph approximates the topology of the space.

In modern deep learning many common practices such as network pruning are performed without observing how this affects the latent representations of the network. The method known as the lottery ticket hypothesis [Frankle and Carbin, 2019] shows that it is possible to find sparse sub networks within a neural network, capable of retaining the performance of the network. These sparse sub networks are found under the assumption of the existence of a 'winning ticket' initialisation. However, it has not been investigated how such methods impact the encoded representations in the latent space.

In this thesis, the latent space of deep convolutional networks trained for image classification tasks is explored through the use of network science methods. Through exploring the nearest neighbour graph of the encoded representations in the latent space, important geometrical aspects such as clusters and regions of low and large concentration may be uncovered. Similarly exploring these regions with respect to the samples class labels, may uncover the presence of regions of uniform and mixed class labels.

Other aspects of the latent space such as the transferability of the geometrical properties in the final stage of training, as well as between two independently initialised models are investigated. In addition the encoded latent representations susceptibility to change through both perturbation as well as pruning is investigated. The susceptibility is related to the robustness of the geometrical aspects of the latent space as well as the models ability to retent its performance.

Lastly, this thesis introduces a higher level exploration tool of the latent space known as the class graph, constructed by considering the distances on the nearest neighbours graph. This tool aims to capture relations between the class mappings, to indicate which classes are present in similar regions of the latent space.

Thesis outline

The contents of this thesis is structured into six chapters. Firstly the necessary theory needed for the methods used in this thesis such as graph theory, the challenges of higher dimensional spaces, and the basic of Neural networks is covered in the theory chapter (chapter 2). Following the theory, is the method chapter (Chapter 3) outlining our motivation for considering the graph approach, as well as stating the methods used for training and the datasets considered. Chapter 4, namely the experiments chapter covers the experiments run during the investigation of the latent space. Each section of the chapter covers a single experiment. The results are presented in chapter 5. This chapter is sectioned identically to the experiment chapter, such that each section states the results for the given experiment in addition to a brief analysis of the implications. Succeeding the results, is a discussion (chapter 6) where all individual results are considered together to analyse the implications in full. The thesis ends with chapter 7 where our work is concluded, and potential directions for future work in the field is proposed.

2 Theory

This chapter presents important theory and concepts fundamental to the work presented in the later chapters. Important topics such as the basics of set theory, fundamental graph theory, and the potential challenges when working in a higher dimensional space is described. Lastly the chapter presents the basic concepts of the Neural network and describes the architecture used in the work.

Some sections of the following theory part is greatly inspired by, or exactly copied from the theory part in [Nørve, 2023], which is a project paper, written prior to the master thesis.

2.1 Set theory

As much of the succeeding theory and concepts strongly builds on set theory, such as graphs, and related network-science, a brief introduction to the set theory is given, by stating the definition of a set, as well of the less strict definition that constricts the multiset.

Set theory is a fundamental field in mathematics, revolving around collections of grouped objects refereed to as a set. Each object in the group is often refereed to as an element in the set. A set is a well known structure much used in fields such as physics, and computer science. In addition there exists less

strict definitions within set theory such as the definition for multisets. For the thesis both definitions are needed and thus presented below:

2.1.1 Set

A set can be understood as a unordered collection of distinct objects grouped into a whole [Fraenkel et al., 1973]. A set is denoted as $A = \{1, 2, 3\}$ where A is the name of the set, and $1, 2, 3$ are its elements. Sets consists of distinct objects such that the multiplicity of the elements are ignored, thus the following sets are equal; $\{1, 2, 3, 1\}, \{1, 2, 3\}$. In addition a set is said to be an unordered collection, such that the sets; $\{1, 2, 3\} = \{1, 3, 2\}$ are equal.

2.1.2 Multiset

A multiset is defined as a collection of unordered objects grouped into a whole where the objects are allowed to repeat [Blizard et al., 1989]. Thus a multiset may contain several indistinguishable copies of a single element. The occurrences of each element in the multiset is given by the elements multiplicity. By this definition the following multisets; $\{1, 2, 3, 1\}, \{1, 2, 3\}$ are not equal

2.2 Graph theory

2.2.1 defintion

A graph is a mathematical structure consisting of a non empty set of vertices and a set of edges. Each edge connects two vertices together which indicates that there is relational information between the two connected vertices. Formally a graph, denoted G is an object defined by a set of vertices V, and a set of edges E; $G = (V, E)$ [Zhang and Chartrand, 2006]. The vertice set V is finite, and must be non empty however, the edge set E may be empty. Each vertice $v \in V$ can contain information given by a feature vector related to v.

Edges

The edges of a graph can be represented in an undirected or directed manner. Undirected edges allows for traversal both ways between two connected vertices, whereas directed edges restricts traversal both ways. More formally an undirected edge can be represented as an unordered set such that; $\{A, B\} =$

$\{B, A\}, A, B \in V$. A directional edge can be represented by an ordered set such that; $\{A, B\} \neq \{B, A\}, A, B \in V$. For a directional graph the notation is slightly changed to separate the incoming and outgoing edges; E_{in} denotes the set containing the incoming edges to a vertice A while, E_{out} denotes the set of vertices outgoing from a vertice A.

Degree

The degree d of a vertice $v \in V$, is defined as the number of edges $e \in E$ incident on v [Zhang and Chartrand, 2006]. When working with a directed graph the degree is defined for both incoming and outgoing edges. The degree of a vertice is denoted as $d(v)$ for an undirected graph. The degree (connectivity) of a vertice is strongly related to the density of the graph. Therefore the average degree of the vertices in a graph is a common measure to understand the density of the graph.

2.2.2 Temporal graph

The definition of a temporal graph can vary depending on the application. The chosen definition is restricted as it does not account for deletion or insertion of vertices. This is purposely done as the following definition only contains what is needed for the experiments performed.

A graph G is said to be temporal if it's properties changes over a time-parameter t. The changes can occur in the topology of the graph, i.e changes in the set of edges E, or by changes in the feature vectors of the vertices V. A temporal graph is denoted as G_t where the subscript t denotes the discrete observation time of which the graph is observed. As The definition only allows for change in the edge set G_t can be written as $G_t = (V, E_t)$.

2.2.3 k nearest neighbour graph

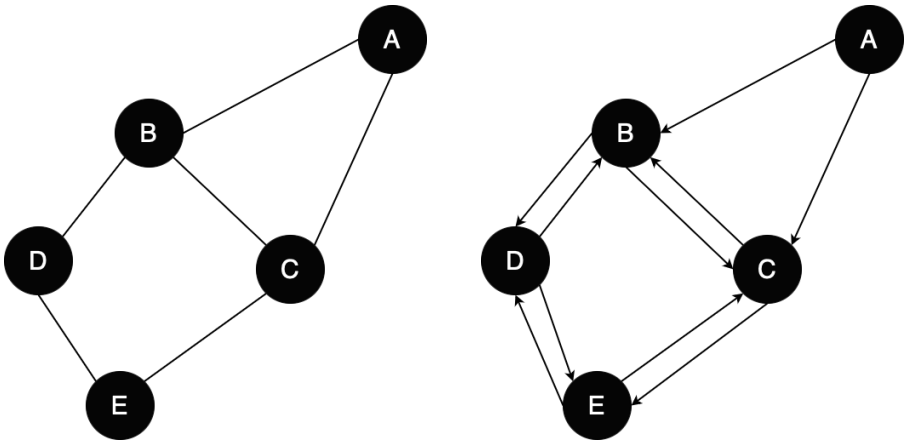
The k nearest neighbour graph is the resultant graph structure constructed by the k nearest neighbours method where each vertice has a minimum degree of k. Consider a set of n samples with a dimensionality of d: $D = \{\boldsymbol{x}_n\}_{n=1}^{N}$, $\boldsymbol{x} \in \mathbb{R}^d$. The k nearest neighbours method revolves around constructing edges between the k closest neighbours to each samples $x \in \mathbb{R}^d$. This is repeated for all samples $\boldsymbol{x} \in D$.

More formally [Eppstein et al., 1992] gives the following definition: Given the

set of samples $D = \{\boldsymbol{x}_n\}_{n=1}^{N}$ present in a $\mathbb{R}^d$ dimensional space, the nearest neighbour to a sample $\boldsymbol{x}_i$ is $\boldsymbol{x}_j, i \neq j$, with a minimum distance to $\boldsymbol{x}_i$ measured by some distance metric. An edge $e(v) = (\boldsymbol{x}_i, \boldsymbol{x}_j)$ is then constructed between the samples. This is performed for each sample $\boldsymbol{x}_i$ such that each sample has an edge to it's k closest neighbouring samples.

As stated, all vertices in the resulting graph structure will have a minimum degree of k. However, vertices can have a higher degree than k if the topology of the data in the $\mathbb{R}^d$ dimensional space allows for it. Consider two samples $a, b \in D$: In the case where a is present in the set of k vertices closest to b, without b being present in the set of the k closest vertices to a, then a, b will have a higher degree than k given an undirected construction of G.

This concept is illustrated in figure 2.1, where two k nearest neighbour graphs are constructed from the set of samples $X = \{A, B, C, D, E\}$, with a $k = 2$. Note that for the undirected graph (2.1a), it can be observed that the vertices B, C has a higher degree than $k = 2$ for the reason described above, since vertice A is closest to B, C, however A is not amongst the 2 closest to B or C.



(a) Undirected k nearest neighbours graph the set of samples.

(b) Directed k nearest neighbours graph for the set of samples.

Figure 2.1: Figures demonstrating an undirected and directed k nearest neighbours graph from the same sample set $X = \{A, B, C, D, E\}$, with $k = 2$.

2.2.4 Shortest path

A fundamental problem in graph theory is finding the shortest path between two vertices v_i, v_j. The shortest path is commonly defined by a set A of vertices which minimises the total edge distance d between v_i, v_j.

Dijkstra's Shortest Path Algorithm [DIJKSTRA, 1959], is a widely used method to find the shortest path between two vertices v_i, v_j. Given a weighted graph $G(V, E)$ where each edge $e \in E$ is associated to a non negative distance $d(e)$, representing the distance between the vertices connected by e. For an unweighted graph the distance $d(e) = 1 \quad \forall\, e \in E$. Then the shortest path can be found with the following approach according to [DIJKSTRA, 1959]:

First three sets; A, B, C are defined to divide the vertices in during the algorithm:

A Contains the vertices with the current minimum distance to v_i. The vertices are added to A in the order of increasing minimum length to v_i.

B Contains the vertices which are to be explored next, i.e the vertices that are connected to at least one vertice in A but not yet present in A.

C Contains the remaining vertices in G.

Likewise three sets; I, II, III are defined to divide the edges during the algorithm:

I Contains the edges present in the shortest paths between the vertices in A and v_i.

II Contains the branches that are next to be a part of I}. There is only one edge in II, which leads to each vertice in B.

III Consists of the remaining edges in G not yet considered, or rejected by the solution.

The algorithm is initialised with all vertices present in set C, and all edges present in set III. Vertice v_i is transferred to set A. Consider the following two step approach:

Step 1: Evaluate the edges e_r connecting vertices v_r in sets B or C to the last vertice transferred to A. If v_r is present in B: Consider if traversing along e_r results in a shorter path from v_i to v_r than the path using the corresponding branch in II: e_r is rejected if the path is longer than the established path in II. However if the path using e_r leads to a shorter path between v_i, v_r then reject the established path, and replace it with e_r in set II. In the case that v_r is present in C, then v_r is transferred to B and the edge e_r is transferred to II.

step 2: Under the restriction that only one edge in II is considered with the edges in I. Then all vertices in B has a distance to v_i. The vertice $v_r \in B$ that has the shortest distance to v_i is transferred to from B to A, likewise the corresponding edge $e_r \in$ II is transferred from II to I.

The above two step algorithm is repeated until the vertice v_j is transferred to A. A then contains the vertices present on the shortest path from v_i to v_k, similarly I contains the edges defining the shortest path. When the set A is found, then the shortest path distance can be computed as:

$$\min d(v_i, v_j) = \sum_{n=1}^{|A|-1} d(A_n, A_{n+1}) \tag{2.1}$$

Diameter The diameter of a graph G is defined as the number of vertices in the longest shortest path present in G.

2.2.5 Clustering in graphs

Clusters is a partition of a set where there is a high similarity between the items in the partition [Meidiana et al., 2019]. Finding the vertices that forms a cluster within a graph is often desirable as it can give valuable information about the graph topology. There are several methods for measuring, and detecting clusters in a graph, such as the clustering coefficient [Latapy, 2008] and the Louvain comunity detection method [Blondel et al., 2008].

Clustering coefficient

For a graph $G = \{V, E\}$, the clustering coefficient for a vertice v with a minimum degree of 2 is defined as the probability that any two random neighbours of v is connected [Latapy, 2008]. If two neighbours of v, denoted b, c is connected, then $\{a, b, c\}$ forms a triangle, the clustering coefficient is found as the number of triangles that includes v divided by the degree of v; $d(v)$.

The clustering coefficient is often used for describing an entire graph, and is then found as the average clustering coefficient for all vertices $v \in V$.

Louvain community detection

When analysing graphs, the identification of isolated groups of vertices is often of interest. These isolated vertices forms a cluster, or community that is weakly

connected to other clusters. The Louvain clustering method [Blondel et al., 2008] finds the communities by partitioning the cluster such that a modularity function Q is maximised. Q can be defined given a Graph $G = (V, E)$. Let $m = |V|$ be the number of vertices in G, for vertices $i, j \in V$ let k_i, k_j denote the degree of i, j and A_{ij} denote the connectivity between i, j then:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j), \qquad \delta(c_i, c_j) = \begin{cases} 1, & c_i = c_j \\ 0, & otherwise \end{cases} \tag{2.2}$$

In equation 2.2 c_i, c_j denotes the community labels assigned to vertices i, j. The community labels c are found by iteratively performing a two phase process:

1. Modularity optimisation phase

- Assign a unique community label c_i to each vertice i.
- Iteratively assign vertices to the neighbouring community that maximises the increase in Q.
- Run until max Q is reached.

2. Community aggregation phase
Construct a higher level network by aggregating the vertices in each community to a single vertice in the new network. The edges in the new network represents the total weights between the communities found in the modularity optimisation phase.

The two phases are run iteratively until no further improvement in Q is found. As a result of this iterative method the Louvain method dynamically assigns the number of clusters in the network such that Q is maximised. One disadvantage of the Louvain method is the greedy process performed in step 1. Thus it is not guaranteed to find the global optimum.

2.2.6 Graph comparison methods

Graphs can be compared on several scales, global metrics can be computed, such as the number of equal edges. Graphs can also be compared at a vertice, or edge level giving a measure of local similarities.

Jaccard similarity

The Jaccard similarity, or Jaccard index is a method that defines a similarity score J between two sets A and B. The measure J is computed as the ratio between the intersection and union of the two sets [TT, 1958]:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.3}$$

J then becomes a value in the interval $[0, 1]$ where a higher value corresponds to a higher similarity between sets A, B. By the definition this measure is symmetric such that $J(A, B) = J(B, A)$.

The Jaccard similarity is not a graph specific measure, however can be used on graphs by comparing the set of neighbouring edges for two vertices v_1, v_2. Observe vertice B, C in figure 2.1a. B's connected to the vertices $S_B = \{A, C, D\}$, and C's connected to vertices $\{A, B, E\}$, The Jaccard similarity between A, C can then be found as:

$$J(B, C) = \frac{2}{6} = \frac{1}{3}$$

Graph average Jaccard similarity

To construct a measure for global similarity between two graphs consisting of the same set of vertices, the average Jaccard similarity can be used. Let $E(v)$ denote the set of edges connected to vertice v, then the graph average Jaccard similarity between two graphs $G_A = (V_A, E_A), G_B = (V_B, E_B)$ can be defined as:

$$J_{avg}(G_A, G_B) = \frac{\sum_{n \in V_A} J(E_A(n), E_B(n))}{|V_A|}, \quad V_A = V_B \tag{2.4}$$

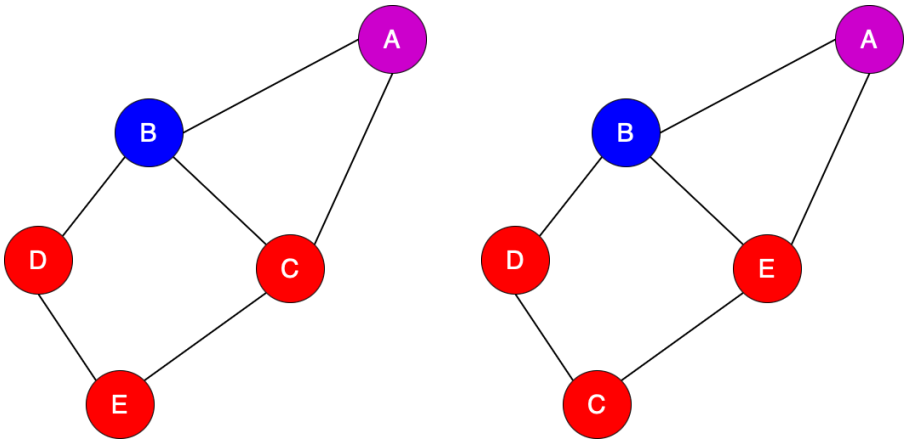
Class neighbourhood similarity

The class neighbourhood similarity measure is proposed as a less strict measure of similarity between pairwise equal vertices in a temporal graph. Consider two graphs with the same set of vertices V; $G_A = (V, E_A)$, $G_B = (V, E_B)$. Each vertice $v_i \in V$ has a corresponding discrete label $y_i \in \mathbb{N}$.

Consider a vertice $v_i \in V$, and let S_A be the set of connected vertices to v_i in G_A. Similarly let S_B be the neighbours of v_i in G_B. Then The following multisets, Y_A, Y_B can be constructed with the class property for each vertice in S_A, S_B: $Y_A = \{y_i \mid v_i \in S_A\}$, $Y_B = \{y_i \mid v_i \in S_B\}$. The class neighbourhood similarity (CNS) between two graphs can be defined:

$$CNS(G_A, G_B) = \sum_{v \in V} \frac{|Y_A \cap Y_B|}{|Y_A|} \tag{2.5}$$

The class neighbourhood similarity evaluates the to a value between $[0, 1]$, describing the similarity in the neighbourhoods with respect to the class property of the neighbouring vertices. It should be noted that the metric is non symmetric as it is divided by the length of set Y_A and does not account for the change in set length between Y_A, Y_B. Thus $CNS(G_A, G_B) \neq CNS(G_B, G_A)$.



(a) G_A **(b)** G_B

Figure 2.2: Two undirected graphs G_A, G_B. The colour of each vertice corresponds to the class property y_i of the vertice.

Consider the undirected graphs G_A, G_B in figure 2.2, the class neighbourhood similarity is specified to be calculated between two graphs, however the example will show the similarity for only one vertice; B. In graph G_A, vertice B has the following multiset $Y_A = \{red, red, purple\}$, likewise for G_B, vertice B has the multiset $Y_B = \{red, red, purple\}$. the class neighbourhood similarity is then found to be:

$$\frac{|\{red, red, purple\} \cap \{red, red, purple\}|}{|Y_A|} = \frac{3}{3} = 1 \tag{2.6}$$

In terms of class the neighbourhood properties, the neighbourhood of vertice B is the same. However if the same comparison was made with the Jaccard similarity; then vertice B would get a Similarity of $J = 0.5$, as it measures the similarities with respect to exact neighbours and not the neighbour property.

2.3 The curse of high dimensionality

The curse of high dimensionality is a problem that arises when working with high dimensional data. For each additional feature there is an exponential increase in the unit volume of that space. Let 5 samples be evenly spaced on a unit line $\mathbb{R}^1$, observe that for a unit area in $\mathbb{R}^2$ 25 is required to span the area with equal density to the $\mathbb{R}^1$ space. This exponential increase in the unit volume continues as the space $\mathbb{R}^d$ increases. This characteristic makes higher dimensional spaces harder to work with. As a consequence, some requirements not present in the lower dimensional spaces needs to be met.

As discussed in [Aggarwal et al., 2001], it is important to choose an appropriate distance measure when working in higher dimensional spaces. They show this by observing the distance between the maximum and minimum distance from the origo for arbitrary data. Given some data with $N \in \mathbb{R}^d$ samples and an L_k norm, let $Dmax_d^k$ be the furthest distance from the origo for the N points. Similarly let $Dmin_d^k$ be the closest distance to the origo for all N points it is shown that:

$$C_k \leq \lim_{d \to \infty} E\left[\frac{Dmax_d^k - Dmin_d^k}{d^{1/k-1/2}}\right] \leq (n-1) \cdot C_k \tag{2.7}$$

In equation 2.7 n denotes N samples drawn from an arbitrary distribution, and C_k is a constant dependent on k present such that the data can be viewed as drawn from an arbitrary distribution. The equation shows that the relative distance between the maximum and minimum distance $Dmax_d^k - Dmin_d^k$ depends on the denominator $d^{1/k-1/2}$. As the number of dimensions d approaches infinity, there is three outcomes depending on the chosen norm k. Observe that when $k > 3$ the denominator converges to zero. For $k = 2$ the expression approaches a nonzero boundary value, however for the manhattan norm, $k = 1$ the expression diverges to infinity.

From the arguments above it is clear that the L_k norms where $k > 3$ performs

poorly in higher dimensional spaces. For the two remaining norms, L_1 and L_2 it can be observed that the $L1$ manhattan norm $||\boldsymbol{x}|| = \sum_{i=1}^{n} |x_i|$ should be favoured when working in higher dimensional spaces when considering that the relative $max - min$ distance remains unbounded.

2.4 Neural networks

Neural networks has become a fundamental part of modern deep learning [Alpaydin, 2020]. They consist of an input layer, intermediate hidden layers, and an output layer. Each layer is commonly followed by a non linear function, such that the network can learn non linear patterns in the data. Each layer in the network has weights associated to it that are applied on the data. The weights are often called the parameters of the models. These parameters are the parts of the network that is optimised during the training of the model. The structure, of the parameters, and how they are applied on the input data depends on the layer archetype.

The neural networks capability to discover complex patterns in data, and leverage that to create complex decision boundaries is unparallel to traditional machine learning algorithms. At its core the neural network relies on a hidden latent representation to uncover the complex patterns used for decision making. Depending on the task the dimensionality of the latent space can be higher or lower than the dimensionality of the input data. Typically the latent space dimensionality is lower in tasks that aim to compress information into a sparser representation. For classification tasks it is common that the networks latent space has a high dimensionality.

Neural networks can be constructed by a mixture of different layers depending on the task it is designed to solve. When working with images a common architecture consists a set of convolutional layers encoding the data, feeding a latent representation to a set of fully connected layers that performs the classification. This architecture known as Convolutional neural networks or CNN for short was introduced as early as 1998 [Lecun et al., 1998] with the introduction of LeNet5.

2.4.1 Fully connected layers

The fully connected layer in a neural network performs a linear mapping of some n dimensional input data $\boldsymbol{x} \in \mathbb{R}^n$, to a m dimensional output vector $\boldsymbol{y} \in \mathbb{R}^m$ through a $n \times m$ weight matrix $\boldsymbol{W}$ [Koutroumbas and Theodoridis, 2008]. To allow for learning non linear patterns, an activation function f is commonly

applied to the output vector $\boldsymbol{y}$, creating the vector $\boldsymbol{z} \in \mathbb{R}^m$. The inclusion of a bias term w_0 is commonly included to allow for learning a shift to capture offsets in the patterns. The mapping is defined by the following equation:

$$\boldsymbol{z} = f(\boldsymbol{W}\boldsymbol{x} + w_0) \tag{2.8}$$

2.4.2 Convolutional layers

The convolutional layer was a proposed layer architecture to improve pattern detection, and performance in grid structured data such as images [Goodfellow et al., 2016]. By applying filters through the convolutional operation, it is possible to detect patterns while reducing the number of learnable parameters $\boldsymbol{W}$. Each convolution layer typically consists of several filters, such that the layer can detect multiple different patterns in the input signal. The introduction of the convolutional layer allows for several advantages when working on grid structured data, such as parameter sharing, and translation equivariance:

For a function to be equivalent the following relation must be satisfied: For a function $f(x)$ to be equivalent to a function g then:

$$f(g(x)) = g(f(x)) \tag{2.9}$$

This relation shows that the order of which functions are applied, does not change the resultant output. This is relevant for images in the context of shifts, because of the equivalent nature of the convolution operation, the same object translated in the input will give the same output regardless of the spatial position of the object in the image.

2.5 Resnet

Resnet is a series of convolutional neural network architectures, introduced by [He et al., 2016]. The Resnet architecture introduced building blocks capable of residual learning via skip connections and the identity mapping. This allows for constructing deeper neural network stacks while retaining the spatial resolution in the feature maps. Each block is defined by the function ([He et al., 2016] eq 1):

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \{W_i\}) + \boldsymbol{x} \tag{2.10}$$

In equation 2.10 $\boldsymbol{y}$ denotes the output mapping, $\boldsymbol{x}$ denotes the input mapping, and $\mathcal{F}(\boldsymbol{x}, \{W_i\})$ represents the function we estimate by stacked convolutional layers. For this equation to hold true, the dimensionality of $\boldsymbol{x}$ and $\mathcal{F}(\boldsymbol{x}, \{W_i\})$

needs to be the same. This is the purpose of the identity mapping which rescales $\mathcal{F}$ to the dimensions of $\boldsymbol{x}$, thus allowing for an element-wise addition between $\mathcal{F}$ and $\boldsymbol{x}$.

Several of these operations can be stacked to create one bigger layer able to learn representations without reducing spatial resolution.

Resnet-18

The Resnet-18 is the most compact Resnet model introduced in [He et al., 2016]. The layer structure of the basic Resnet-18 is shown in table 2.1. The experiments were performed with the models available from the Pytorch [Paszke et al., 2019] library. This implementation of the Resnet-18 includes batch normalisation, and dropout regularization after each convolutional layer. The core architecture remains equal to the one described in table 2.1.

Layer name	Output size	architecture
conv1	112×112	$7 \times 7, 64$, stride 2
conv2_x	56×56	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
		downsample stride 2
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
		downsample stride 2
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
		downsample stride 2
	1×1	average pool
fc, softmax	$1 \times n_{classes}$	linear $512 \times n_{classes}$

Table 2.1: Resnet 18 architecture

2.6 The lottery ticket hypothesis

The lottery ticket hypothesis [Frankle and Carbin, 2019], is a hypothesis proposing that there exists a sparse sub-network within the full network that is capable of giving similar performance to the full network. Thus a method was proposed to find these sub-networks structures through an iterative pruning process.

By assuming the initialisation of the network parameters denoted θ_0 is the 'lottery ticket' initialisation. Then the network is trained, after which a small percentage p of weights are creating a pruning mask m_0. The remaining weights not deactivated by m_0, are initialised to the values originally initialised to θ_0. This process is repeated for several iterations to detect the sparse sub-networks capable with similar performance to the original.

More formally, given a neural network $f(x; \theta 0)$ where $\theta_0 \sim \mathcal{D}_\theta$ denotes the randomly initialised parameters. The paper proposed the following training scheme to find the sub-networks:

1. Randomly initialise a neural network $f(x; \theta_0)$ (where $\theta_0 \sim \mathcal{D}_\theta$)
2. Train the network for j iterations resulting in the trained weights θ_j
3. Prune a small percentage p% of the unpruned parameters creating a mask m_0.
4. Reset the network parameters to the original initial parameters $\theta_0 \odot m$
5. Repeat steps 2-4 n times each iteration increasing the mask $m_n = m_n + m_{n-1}$ with the additional pruned parameters.

By following the described method, the similarly performing sub-networks can be discovered. It was discovered that with this method it is possible to reduce the number of parameters in the original network by $80 - 90\%$. As opposed to a one-shot pruning method where a larger percentage of weights are pruned in one step, the lottery ticket hypothesis method retains accuracy while reducing the networks more than what is possible with a one-shot pruning method.

3 Method

This chapter gives a description of the steps taken in order to explore the latent space of deep Neural Networks. In detail it will elaborate on which architecture is explored, as well as which intermediate layer used for extracting the latent space representation. Furthermore the chapter presents our motivation for the graph approach, as well as explain the considerations taken for constructing the graphs used for the analysis. Lastly the chapter will give a brief description of the datasets and training schemes used.

3.1 Latent data extraction

The latent representations of the input data is dependent on the model architecture as well as the layer depth it is extracted from, both in regards to the values, and the dimensionality of the representations. For all experiments surrounding the latent representations we have been working with the Resnet-18 architecture, with the latent representation extracted between the average pooling and the linear classifier (table 2.1). This latent representation is after all convolutional layers has operated on the input data, such that the data presumably is at it's richest point in terms extracted information. The dimensionality of the latent representations at this layer is; $\mathbb{R}^{512}$.

3.2 Motivating the graph approach

3.2.1 Capturing relations

When training a model, the samples position in the latent space changes. The final latent distribution of the data is dependent on the initial parameters θ_0, the training scheme, as well as the hyper-parameters used. Thus comparing absolute positions in the latent space would be dependent on the convergence of the model. This can pose a challenge when comparing the two trainings of the same model architecture with different initialisations, observering the changes with respect to pruning, or to observe the evolution of the latent space during training.

As opposed to using the absolute positions, a graph approach could favourably be used as it is the relations between the data that is important. By constructing a graph from the data representation in the latent space such as with a k nearest neighbours graph it could be possible to capture the relative relations between samples in the latent distribution. With the graph approach an absolute mapping is omitted while the relative information in the latent representations is retained. These relative graph representations could give a good foundation for observing relations, properties, and changes in the latent space.

3.2.2 Approximating the latent manifold

As the function constricting the mapping to the latent space is unknown, it limits the possible exploration methods. Thus to explore the latent representations, an approximation method is necessary to explore the the topology of the space. The closest estimation of the latent space available to us, is the latent representations of the input data. For this reason, the construction of a k nearest neighbours graph G using these latent representations is proposed. The graph G, then functions an approximation of the latent space topology. depending on the dimensionality of the latent representation, a suitable distance metric should be considered. In addition it should be noted that the choice of k could impact the properties of the representation G gives.

3.3 Constructing the graph of the latent space

Firstly for the construction of the k nearest neighbours graph of the latent space it is important to consider that the latent space of the Resnet-18 is very high dimensional; $\mathbb{R}^{512}$. The high dimensionality constricts the number of distance

metrics capable of capturing the high dimensional distances in a good way. For our work, we construct the graph G with the $L1$, manhattan norm distance. The $L1$ is considered for it's property of retaining an unbounded relative min - max distance in higher dimensional spaces. Secondly the selection of k for constructing the graph must be considered. We explore the properties in the latent space for several values of $k \in \{4, 8, 16\}$ for constructing G. Thus the evolution of the properties with respect to k can be observed

Lastly, for the construction of the graphs, the test or validation data is considered in favor of the training data. By constructing the k nearest neighbour graph on unseen data, it is safer to assume that the latent representations of the data is generalised and not an artefact of overfitted training data. Computationally our work with graphs such as such as construction, exploration, and computation has been performed using the networkX package [Hagberg et al., 2008], a well known, high performance network-science library for Python.

3.4 Graph visualisations

In the result section, we present visualised representations of some graph structures are presented. These visualisations are generated using the graph analysis tool Gephi [Bastian et al., 2009]. The graph topology shown in the figures, is generated using the ForceAtlas2 layout algorithm [Jacomy et al., 2014]. This layout algorithm creates the topology by treating the graph as a physical system, where the vertices has repulsive forces between them scaled by the degree. Whereas the edges applies a linear force pulling connected vertices together relative to the strength of the weight between them. By letting the algorithm run, it will find an equilibrium state that the system converges to.

As an example of a visualisation created by the ForceAtlas2 layout, consider the graph visualisation presented in figure 3.1. The graph presented was constructed with the latent representation of the Cifar100 test dataset with the Resnet-18 model. A $k = 4$ was used for the construction, considering the $L1$ norm between each latent sample.

3.5 Datasets

For the experiments conducted, two datasets has been considered - namely Cifar100, and Imagenet1k. The two datasets are different in computational complexity, and is selected based on the given experiments computational demand.

3.5.1 Cifar100

Cifar100 [Krizhevsky et al.,] is a 100 class balanced dataset consisting of 32×32 pixel images. See table 3.1 for the number of samples in the training and test set. To be used with the original Resnet-18 architecture, the images are rescaled to a size of 224×224, before given as an input to the model.

Split	n samples
Training	50000
Test	10000

Table 3.1: Distribution of samples in the training and test set of Cifar100.

3.5.2 Imagenet1k

Imagenet1k [Russakovsky et al., 2015] is a widely used dataset for benchmarking model performances. It consists of 1000 classes distributed equally across all classes. The images varies in size, however the common process is to rescale them to a 256×256 image, followed by cropping the center of the image such that the final size of the image is 224×224. See table 3.2 for the number of samples in the training and validation set.

Split	n samples
Training	1.281.167
Validation	50.000

Table 3.2: Distribution of samples in the training and test set of Imagenet1k.

3.6 Training schemes

For the experiments there has been a mix of two supervised training methods: A traditional training scheme running for a fixed number of epochs, and a scheme utilising an earlystopping criterion to dynamically stop the training when a certain criterion has been met. For each experiment in chapter 4 there will be a small section remarking which training method has been applied as well as defining the criterion, loss, and scheduler function as well as the hyperparameters used for the given experiment.

3.6.1 Fixed training scheme

An experiment utilising the fixed training scheme will run for a fixed number of epochs independent of training progress and state. It is applied to experiments where it is of interest to make absolute comparisons between different training runs.

3.6.2 Earlystopping training scheme

An experiment utilising the earlystopping training scheme will run until a predetermined earlystopping criterion is reached or at most for a fixed number of epochs. The earlystopping mechanism can reduce training time as well as reduce the risk of overfitting the model depending on the criterion chosen. Our earlystopping criterion is implemented based on the observed validation loss, and will cancel training when the validation loss is observed to not decrease between sequential epochs. To allow for some flexibility the earlystopper saves the models parameters at the lowest loss observed, and is allowed to run for a few epochs longer to observe if the criterion will decrease further. The number of epochs it is allowed to search for a new minima is denoted as the *patience* argument of the earlystopper.

Described below is the process performed by the earlystopper after each epoch of training:

1. Calculate validation loss for the epoch L_{val}.
2. If the the validation loss is the lowest observed during training then :
    - Save model parameters.
    - Save the lowest observed loss L_{val}.
    - Reset patience count to 0.
3. Else continue training but increment a patience count by one.
4. If the patience count reaches the predetermined patience of the earlystopper
    - Break training and return the saved model parameters from the best state.

Figure 3.1: Undirected $k = 4$ nearest neighbour graph representation. Each vertice represents the latent space representation for a Cifar100 sample on the Resnet-18 model. The vertices is colored by the Ground truth of the sample, and scaled by the degree.

4

Experiments

This chapter describes each experiment performed for our exploration of the latent space. Each section corresponds to an individual experiment, where factors such as the steps taken to investigate, the methods used, as well as describing the training scheme used to produce the results stated in chapter 5.

4.1 Observing the stability in the k nearest neighbours graph

Exploring the stability of the latent representations is important to indicate the rigidity of the network-science method. As the experiment explores the stability in terms of susceptibility to noise, similarity during training, as well as between models, the results aids establishing baseline properties for comparing further exploration against.

4.1.1 Training scheme for the experiment

For this experiment two Resnet-18 models A, B is trained, each model has their initial parameters independently initialised to each other. Both models has been trained with a fixed training scheme, on the same hyperparameters.

Table 4.1 shows the hyperparameters, optimiser, and scheduler used during training.

Parameter	Value
Dataset	Cifar100
Number of epochs	25
Optimiser	SGD($lr = 0.1$, $momentum = 0.9$)
Scheduler	Step at epochs 10, 20, 30, $factor = 0.1$

Table 4.1: Parameters used in training scheme for observing the stability in the k nearest neighbour graph.

For each epoch during training the model parameters are saved. When training is complete We select 6 sequential epochs from where the test accuracy is observed to stabilise for both models. The latent representations of the test data is then extracted for both models. Further the two latent representation are used to construct the temporal graphs G_{A_t}, G_{B_t} according to the k nearest neighbour method. Each timestep t corresponds to the epoch the representation is extracted from.

4.1.2 Experiment variations

Stability with respect to noise.

For this experiment only one latent representation at a single timestep is needed. This experiment is introduced to observe the the internal robustness of the latent representation. By gradually perturbing the latent representation with a zero mean Gaussian noise with increasing variance, we are interested in observing how the latent representations changes with respect to the amount of variance in the noise added to the system. For this experiment we have a $n \times d$ matrix $\boldsymbol{X}$ representing the n data points in the $\mathbb{R}^d$ latent space, as well as the graph $G_{original}$ constructed by using the latent representation $\boldsymbol{X}$. A perturbed latent representation Y is constructed in the following way:

$$Y = \boldsymbol{X} + \mathcal{N}(0, p \cdot \boldsymbol{\sigma}^2) \tag{4.1}$$

Here in equation 4.1, $\mathcal{N}(0, p \cdot \boldsymbol{\sigma}^2)$ denotes a $n \times d$ matrix with zero mean Gaussian noise. Note that $\boldsymbol{\sigma}^2$ is a $\mathbb{R}^d$ vector where each element corresponds to the variance in the given dimension of $\boldsymbol{X}$, such that each column of $\mathcal{N}(0, p \cdot \boldsymbol{\sigma}^2)$ has a variance equal to the corresponding element in $\boldsymbol{\sigma}^2$. The scalar p, refereed to as the *percentwise relative noise variance* is used to scale the variance of the

noise used for perturbing X.

For each representation Y a graph G_{noised} is constructed, and compared to the unperturbed latent representation represented by $G_{original}$. The similarity between the two graphs are calculated between each perturbation.

Furthermore, to observe how the perturbation of the latent space affects the models ability to perform classifications, we feed the perturbed latent space Y into the fully connected layer of the classifier. Thus the accuracy of the model can be computed as a function of the noise added.

Change in internal stability during training

Exploring how rigorous the latent space representation is with respect to itself during training, can give an indication as to how stable the latent representation is. This internal stability is found by comparing the graphs of the latent representations between epochs $t, t+1$: For each graph G_{A_t}, G_{B_t} is compared to itself in the next timestep $G_{A_{t+1}}$, $G_{B_{t+1}}$.

Comparing the similarity between two training runs

Given different initialisations, two models are not guaranteed to converge to the same minima in the loss surface, even if trained on the exact same scheme and equal hyperparameters. Thus it is of interest to observe to which extent the latent space representations for the different convergences can be compared. With this experiment we aim to investigate the similarities of the convergences through comparing the respective graph representations G_{A_t} and G_{B_t}. The graphs comparisons are performed between G_{A_t}, G_{B_t} at corresponding epochs t.

4.2 Class graph representation

Machine learning algorithms such as the deep neural networks are well known to be black-box methods. Understanding how the network maps relationships between the classes can aid in giving a high level understanding of the models latent space. This is where we propose to introduce the class graph as a higher level observation tool to indicate relations between classes based on the latent representation of the data.

Consider a dataset $D = \{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$ where $x_n \in \mathbb{R}^d$, the dataset is divided

into $C = \{c_1, c_2, \dots c_m\}$ classes. A graph G is constructed with the k nearest neighbours method using the the latent representation of the samples in D using the Manhattan distance metric. Each vertice v_i in G corresponds to the sample $\boldsymbol{x}_i$ with label y_i. Further the shortest path distances $d(v_i, v_j)$ between all pairs of vertices v_i, v_j in G is calculated resulting in a $N \times N$ symmetric adjacency matrix ADJ with shortest path distances between all vertices.

From the adjacency matrix ADJ an average shortest path distance between all samples in in each class c_i, c_j can be computed. Let $S_i = \{n | y_n = c_i\}_{n=1}^{N}$ be the set of indices where $y_n = c_i$, similarly let $S_j = \{m | y_m = c_j\}_{m=1}^{M}$ where $y_m = c_j$ then the average shortest path distance between classes c_i, c_j can be found as:

$$\overline{d}(c_i, c_j) = \frac{1}{|S_i| \cdot |S_j|} \sum_{k \in S_i} \sum_{l \in S_j} ADJ_{kl} \tag{4.2}$$

Performing this for all i, j we get an $|C| \times |C|$ adjacency matrix, where each index i, j gives the average shortest path distance between the classes $i, j \in C$. The average shortest path in this adjacency matrix is then considered when constructing the k nearest neighbours class graph representation; G_{class}.

4.3 Exploring the changes in latent representation across an iterative pruning process

4.3.1 Training scheme for the experiment

For this experiment we train a single Resnet-18 model in accordance to the method proposed lottery ticket hypothesis paper (described in 2.6). The pruning is performed only in the convolutional layers of the network, with the fully connected layer left unpruned during the entire experiment. An earlystopping training scheme is used for each run. Table 4.2 shows the functions and hyperparameters used for the training. We perform the experiment for two pruning ratios; $p = 0.20$ and $p = 0.01$.

Parameter	Value
Number of pruning iterations	15
Number of epochs	100
Optimiser	SGD($lr = 0.1$, $momentum = 0.9$)
Scheduler	Step at epochs 10, 20, 30, $factor = 0.1$
Earlystopper	Max validation accuracy, $patience = 3$

Table 4.2: Parameters used in training scheme for the iterative pruning

4.3.2 Experiment

With this experiment we explore the latent space evolution during the iterative pruning process. For each iteration the model parameters θ_j and pruning mask m is stored to be used for the exploration. As there is no investigation into the change of the latent representations, we will be observing the similarity of the latent representations of each pruned model to the latent representation of the original unpruned model. This could give an indication as to how much information from the entire network that is retained in the sparser sub network discovered. As the model is reinitialised to the same initial parameters at each pruning iteration, the experiment can also give a new perspective on the 'winning ticket' hypothesis with regards to initialisation.

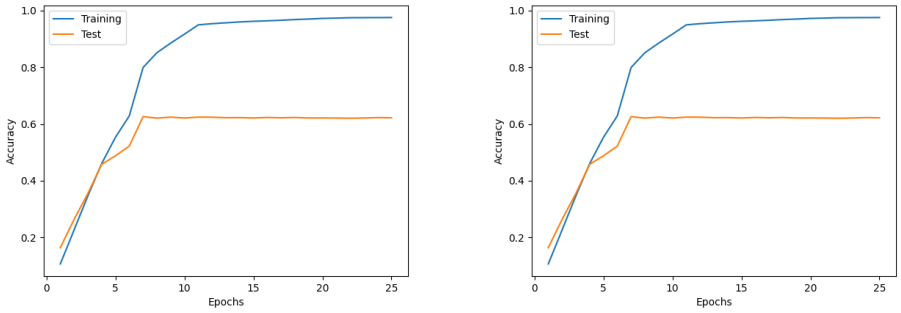
5

Results

This chapter contains the results produced by the experiments outlined in chapter 5. Each section has a corresponding section in chapter 5 explaining the experiment. The results chapter is split into three sections; An investigation of the stability of the k nearest neighbours method, the results of the class graph aggregation, as well as the findings on the exploration of iterative pruning method.

5.1 Observing the stability in the k nearest neighbours graph

The results shown for this experiment is calculated with latent data extracted from two identically trained models, with different initialisations. The figures presented in 5.1 shows the training and test accuracy for the two models during training. Both runs converge in a similar manner to a test accuracy around 60%. The latent representations has been extracted for epochs 10-15, since the test accuracy has converged at this point.



(a) Accuracy for training and test set during run 1.

(b) Accuracy for training and test set during run 2.

Figure 5.1: Training and test accuracy as for Cifar100 on Resnet-18 as function of epochs during model training.

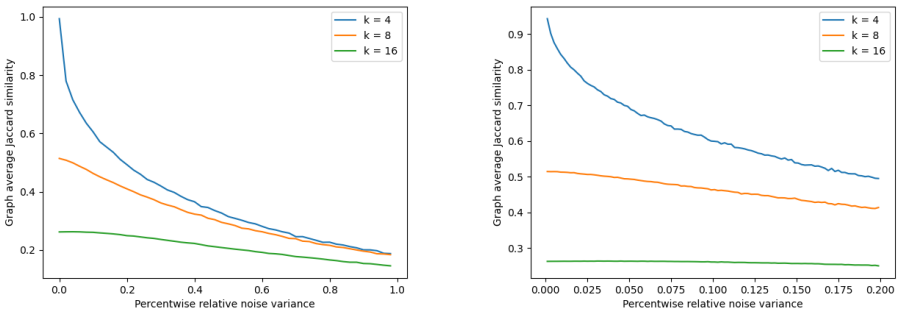
5.1.1 Change in internal stability with respect to noise

When calculating the variance in the dimensions of the latent data $\boldsymbol{X}$, we find that the elements in $\boldsymbol{\sigma}^2$ are quite low, indicating a dense representation. Table 5.1 shows the maximum, minimum, and mean value for the variance vector $\boldsymbol{\sigma}^2$. As each dimension in the latent data $\boldsymbol{X}$ has its own variance in $\boldsymbol{\sigma}^2$ we scale the added noise relative to the values in the variance vector $\boldsymbol{\sigma}^2$ with the factor p in the range $[0, 1]$. Figure 5.2 shows the graph average Jaccard similarity with respect to the percent-wise noise added to the latent representation. Each plot corresponds to the results for a given k used for the graph construction.

Metric | Value	
Max	0.34
Min	0.038
Mean	0.13

Table 5.1: Metrics related to the variance vector $\boldsymbol{\sigma}^2$ for each dimension in the latent data $\boldsymbol{X}$

From figure 5.2 it can be observed that the the Graph average Jaccard similarity decreases rapidly with the increased noise added to the latent representation. This indicates that the datas representation in the latent latent space is very susceptible to small perturbations. Furthermore, when observing the results with respect to k, it can be observed that a lower k achieves higher similarities, however has a steeper decrease with increasing noise. The higher values of k is observed to be more stable with regards to similarity decrease when the noise increases, however is more susceptible to small noise. To observe the graph structures from a different perspective, the class neighbourhood similarity in figure 5.3 is investigated:



(a) Full figure showing Jaccard similarity for all noise variance.

(b) Figure zoomed in on the smaller noise variance.

Figure 5.2: Full and zoomed plots presenting the graph average Jaccard similarity between $G_{original}$ and G_{noised} as a function of the noise added relative to the systems global variance. Each line corresponds to the similarity achieved for a given k.

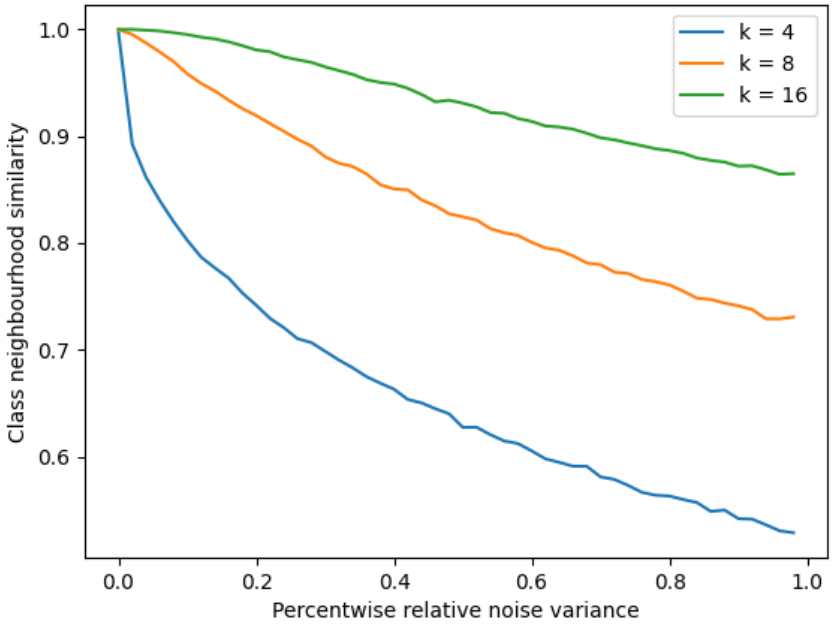


Figure 5.3: Plot showing the Class neighbourhood similarity between $G_{original}$ and G_{noised} as a function of the noise added relative to the systems global variance.

From the class similarity results in figure 5.3, it can be noted that it shows higher similarities with respect to the noise added than the graph average Jaccard similarity. This could indicate that there is well defined clusters for the classes in the latent representation, such that the orientation changes within the clusters, but not with respect to the classes. With respect to k the same trend can be observed regarding the decrease with respect to noise. However, it is observed that all values of k achieves higher similarities with little noise added.

to get an indication on how the perturbation affects the model, the accuracy is found for each perturbation of the latent representation. these results are presented in figure 5.4. The classification is performed for the same perturbations used for comparing the similarity.

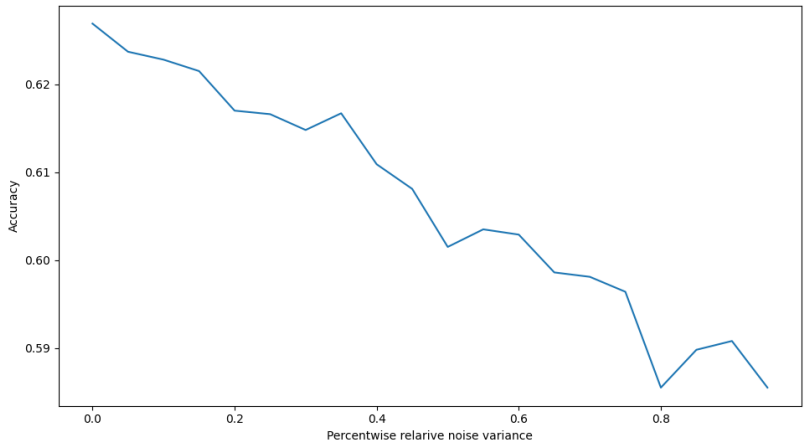


Figure 5.4: The achieved accuracy when sending the noised latent space through the fully connected layer of the trained Resnet-18.

By studying the classification accuracies in figure 5.4, it can be observed that there is a decrease in accuracy when introducing noise to the latent space. However it should be noted that the decrease in accuracy is quite low; at the point of introducing as much variance as is present in the system i.e a percentwise relative noise variance of 1, an accuracy decrease of only around 4% is observed. This is a very minimal change when compared to the significant decrease with respect to the graph average Jaccard similarity when injected with a percentwise relative noise variance of $p = 1$, especially for the lower values of k.

To get an understanding of why such a small decrease in accuracy is observed relative to the larger decreases observed in the similarity measures (figures 5.2, 5.3); some properties of $G_{original}$ constructed with $k = 4$ is investigated. The longest path, or diameter of $G_{original}$ is found to be 13, in addition the average shortest path length is found to be: 6.4. Finally the distribution of shortest path lengths between all pairs of vertices v_i, v_j in $G_{original}$ is calculated. The distribution is shown in figure 5.5. This distribution gives an approximation to the lengths between the samples i, j on the latent space manifold, when connections are kept sparse such that only the closest neighbours are traversable.

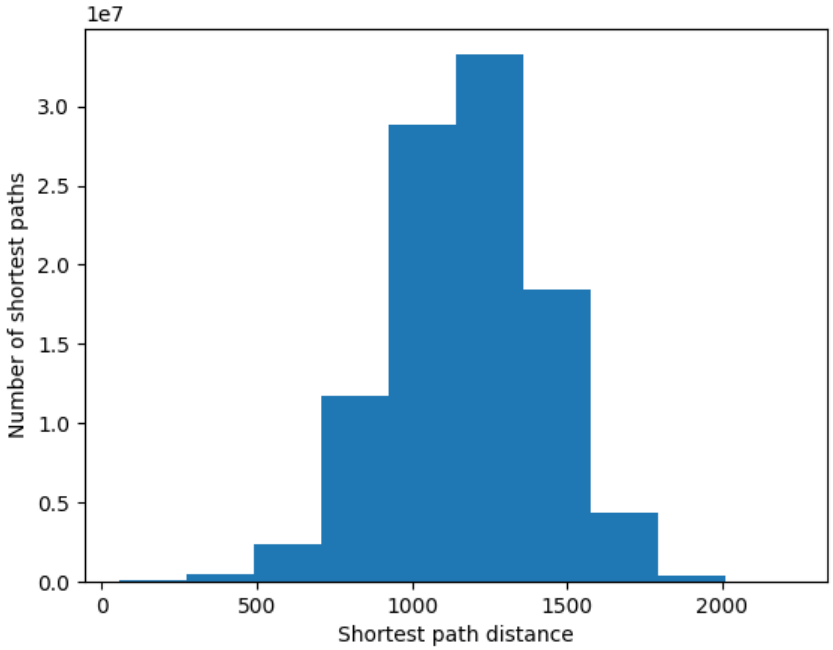


Figure 5.5: The distribution of shortest path length between every pair of vertices v_i, v_j in G.

From the shortest path length distribution in figure 5.5, it can be observed that the majority of shortest path lengths is between $[800, 1500]$.

We estimate the average Manhattan norm for a $\mathbb{R}^d, d = 512$, vector consisting of Gaussian sampled noise: $\boldsymbol{N} = [\mathcal{N}(0, \sigma_n^2]_n^d,\ \sigma_n^2 \in \boldsymbol{\sigma}^2$. By performing a Monte-Carlo approximation we find that the average Manhattan norm of the noise vector $\boldsymbol{N}$ to be: $|\boldsymbol{N}| \approx 150$.

Comparing the norm of $\boldsymbol{N}$ to the observed shortest path length distribution in 5.5, it can be noted that the norm of the noise vector is significantly lower

than most path lengths. This could explain why such a small accuracy decrease is observed, as the noise potentially only shifts the vertices position in the local regions they are present in. Such that the observed decrease in accuracy is caused by the samples present between two regions of well defined class clusters. This observation further supports what was initially discussed with respect to the arguments made for the class neighbourhood similarity, in which there exists well defined clusters for the distinct classes.

5.1.2 Change in internal stability during training

To observe the internal stability of the latent space at the end of training the temporal graph G_t is constructed from the latent representation for the epochs $t \in [10, 15]$. Each state t is compared to the next state t+1. First the comparison with respect to the Graph average Jaccard similarity is observed:

Compared epoch $t, t+1$	$k = 4$	$k = 8$	$k = 16$
10, 11	0.91	0.92	0.93
11, 12	0.91	0.91	0.91
12, 13	0.90	0.91	0.92
13, 14	0.89	0.90	0.90
14, 15	0.91	0.91	0.92

Table 5.2: Graph average Jaccard similarity Between G_{A_t}, and $G_A A_{t+1}$. The results are shown for a selection of k used during the construction of the k nearest neighbour graph structure. The leftmost column shows the epochs $t, t+1$ where the latent space used for constructing the graphs are retrieved.

From table 5.2 it can be observed that the graph average Jaccard similarity reported is stable around 91-93%. This high similarity indicates that the latent representations are rigid with respect to it's final state when the validation accuracy has plateaued. The high similarity is observed across all values of k, with a very small increase in similarity per doubling of k. The same investigation is also performed with respect for the class neighbourhood similarity measure, and presented in table 5.3:

Investigating the results presented in table 5.3, it can be observed that similarity with respect to the class neighbourhood is found to be high with similarities around 95% between the steps $t, t+1$. Considering the high values found for the graph average Jaccard similarity (table 5.2), this is of no suprise as the neighbourhood similarity is a less strict measure than the graph average Jaccard similarity. Hence, if the set of connected vertices are very similar across t for a vertice v, then it follows that the multiset of the class neighbourhood of v will naturally also be almost equal, as the class property of each vertice

Compared epoch $t, t+1$	$k = 4$	$k = 8$	$k = 16$
10, 11	0.96	0.96	0.97
11, 12	0.95	0.95	0.96
12, 13	0.95	0.96	0.96
13, 14	0.94	0.95	0.96
14, 15	0.95	0.95	0.96

Table 5.3: Class neighbourhood similarity Between G_{A_t}, and $G_A A_{t+1}$. The results are shown for a selection of k used during the construction of the k nearest neighbour graph structure. The leftmost column shows the epochs $t, t+1$ where the latent space used for constructing the graphs are retrieved.

$v \in V$ do not change with respect to t.

5.1.3 Comparing the similarity between two training runs

Comparing the similarity between two training runs 1, 2 can give an indication of the presence of structures in the latent space representation that is critical for the learning process. The two temporal graphs G_{A_t}, G_{B_t} are constructed, and their similarity is compared at each step t. The resultant graph average Jaccardsimilarity is reported in table 5.4:

Compared epoch t	$k = 4$	$k = 8$	$k = 16$
10	0.15	0.18	0.21
11	0.16	0.18	0.21
12	0.16	0.18	0.21
13	0.15	0.18	0.21
14	0.16	0.18	0.21
15	0.15	0.18	0.21

Table 5.4: Graph average Jaccard similarity Between G_{A_t}, and G_{B_t}. The results are shown for a selection of k used during the construction of the k nearest neighbour graph structure. The leftmost column shows the epoch t corresponding to the latent space used for constructing the graphs.

From table 5.4 it can be noted that the reported similarities are systematically low across all steps t, with a similarity around $15 - 20\%$ depending on k. For each doubling of k there is observed a small increase in the reported similarity. When investigating the convergence of the two runs in figure 5.1, it can be observed that the two runs converge very similarly, however according to the graph average Jaccard similarity they appear almost entirely different. To inves-

tigate the similarities between the two runs further, the class neighbourhood similarity is measured, and reported in table 5.5:

Compared epoch t	$k = 4$	$k = 8$	$k = 16$
10	0.53	0.57	0.64
11	0.52	0.57	0.64
12	0.53	0.57	0.63
13	0.53	0.56	0.64
14	0.53	0.57	0.63
15	0.52	0.56	0.64

Table 5.5: Class neighbourhood similarity Between G_{A_t}, and G_{B_t}. The results are shown for a selection of k used during the construction of the k nearest neighbour graph structure. The leftmost column shows the epoch t corresponding to the latent space used for constructing the graphs.

With respect to the class neighbourhood similarity reported in table 5.5, it can be noted that the two training runs are found to be more similar. The similarity with respect to class neighbourhood is found to stable across all steps t with a similarity between $52 - 64\%$, with the similarity increasing with the doubling of k. The higher similarity with regards to the class neighbourhood, indicates that the two different models has learned some similar patterns in the latent representation with regards to the orientation of the classes.

5.2 Class graph representation

In this section the class graph representation for both the Imagenet1k, and Cifar100 datasets is presented. Although the class graph for both datasets are presented, the main focus will be on the class graph constructed for the Imagenet1k dataset. Both class graphs are created by aggregating the shortest path distances on the respective $k = 4$ nearest neighbour sample graphs of the Resnet-18 latent space.

5.2.1 Imagenet1k

A visualisation of the class graph G_{class} for the Imagenet1k validation set is shown in figure 5.6

From the class graph layout (figure 5.6) it can be observed that the network topology appears to be composed of local clusters, and denser regions centered

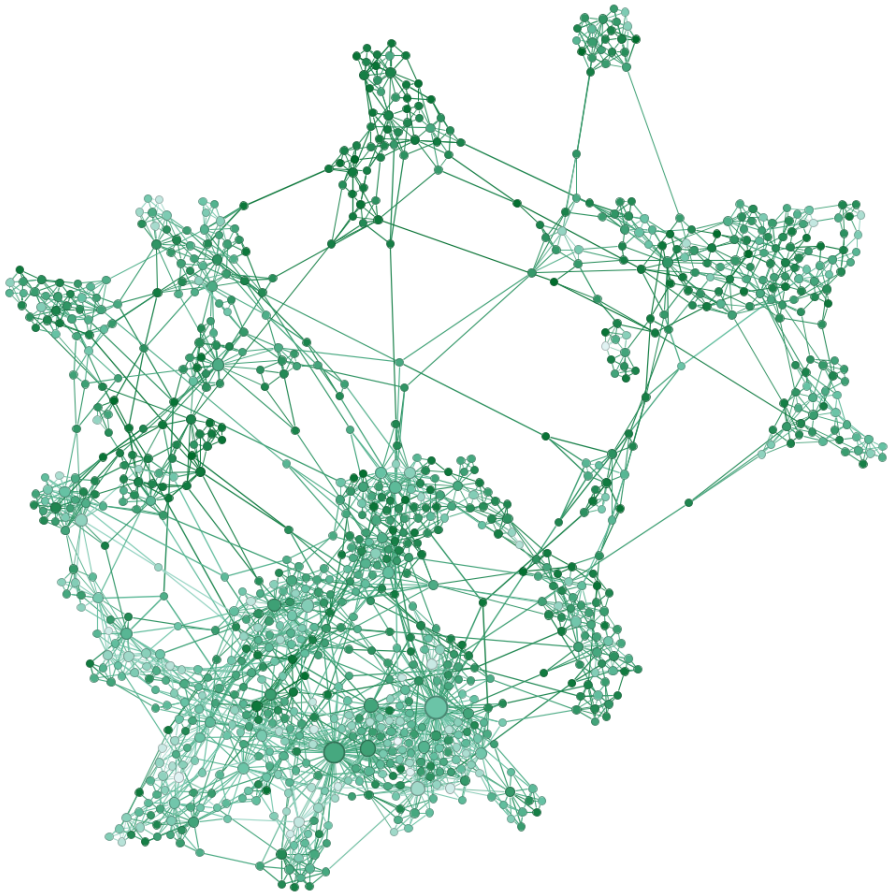


Figure 5.6: The class graph of Imagenet1k showing the entire structure of the graph. the size of the vertices are scaled by their degree, and the hue corresponds to the class average accuracy.

around a few hub vertices. Some common graph measures, as well as the degree distribution of G_{class} are calculated and presented. The measures are presented in table 5.6, and the degree distribution is shown in figure 5.7.

Measure	Value
Diameter	14
Average path length	6.4
Average Degree	6.3
Clustering coefficient	0.46

Table 5.6: Statistics for the classgraph G_{class} of Imagenet1k (figure 5.6).

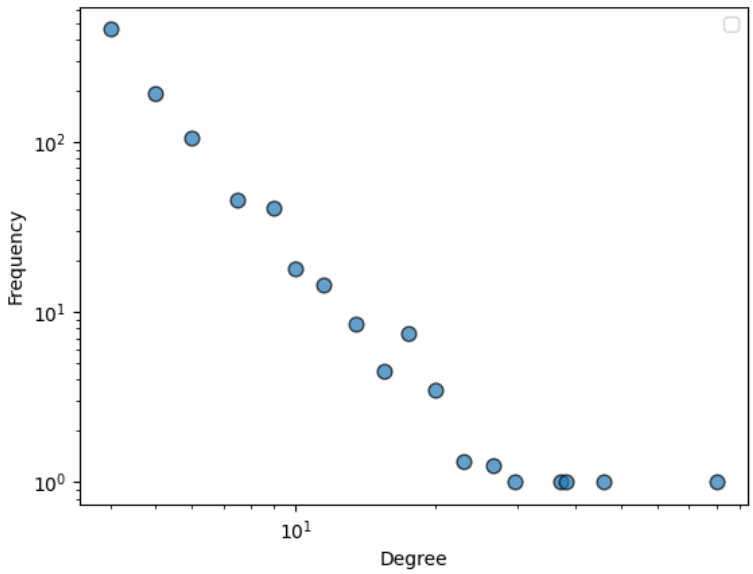


Figure 5.7: Degree distribution of the imagenet class graph G_{class} presented on a logarithmic scale

Clustering

The degree distribution 5.7 is observed to follow a very linear trend, with a tail containing vertices with a significantly higher degree than the average degree. Indeed when inspecting the computed measures in table 5.6 it can be observed that the average degree of 6.3 (table 5.6) is significantly lower than the highest degree vertices present in the network. The linear trend gives a good indication that the degree distribution of the class graph follows the power law distribution.

Further exploration of the class graph includes further inspection into the clustering properties of G_{class}. The average degree reported in table 5.6 was found to be 0.46, indicating the presence of some clustering properties through the number of triangles found. However, to assert that the inspected clustering (figure 5.6) is not just an artefact of the Force-Atlas algorithm, the Louvain community detection algorithm is computed for G_{class}. The Louvain algorithm detected 17 distinct clusters within G_{class} that contains between [1.1%, 12.6%] of the vertices depending on the cluster. The detected clusters in the Imagenet1k clssgraph are presented in figure 5.8

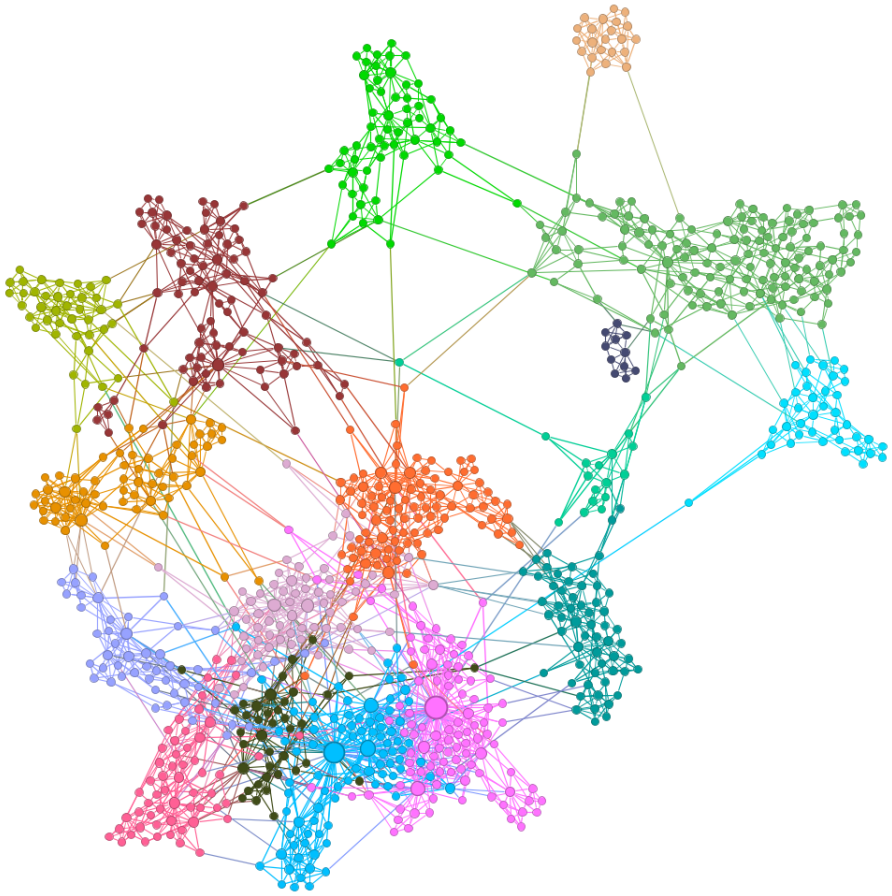


Figure 5.8: The class graph of Imagenet1k showing the entire structure of the graph. the size of the vertices are scaled by their degree, the color corresponds to the cluster found by the Louvain method. Thus there are 17 distinct colors in the figure.

A small selection of the clusters detected in figure 5.8 are inspected in a qualitative manner by looking at the label of each class contained in the clusters.

By observing the labels corresponding to the vertices in figure 5.9, it can be observed that the classes this detected cluster represented are all related to feline species. Considering that the classes are all feline species, it should be of no suprise that they are clustered in the latent representation, considering that they have several shared features across the classes. However, it is still of interest to be able to observe this clustering tendency, as it can give indications on how the model relates the various classes.

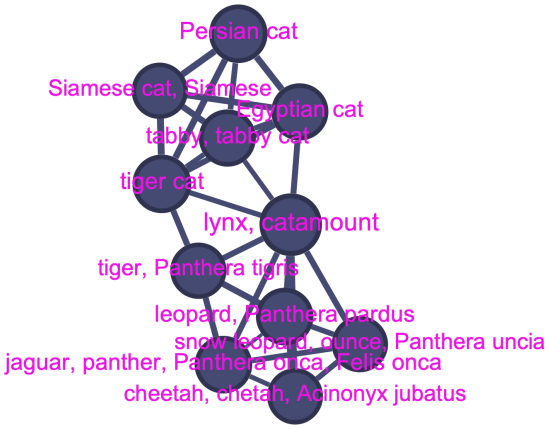
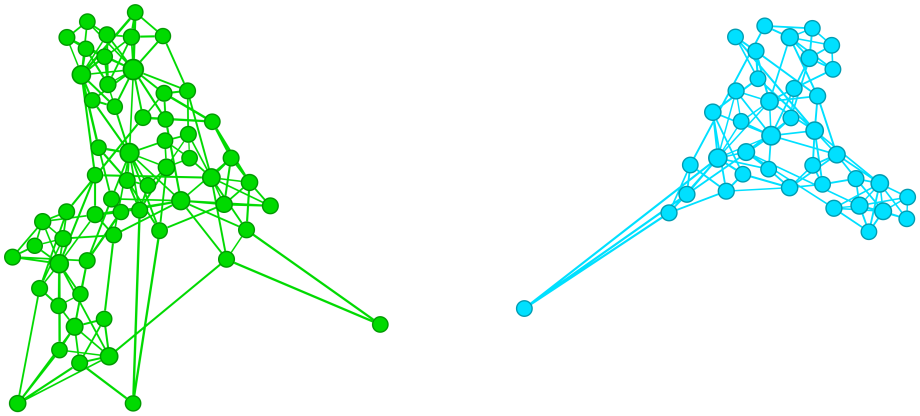


Figure 5.9: A single community from figure 5.8. The labels on each vertice shows the label for the class it represents.



(a) A community with classes consisting mostly of different types of birds.

(b) A community with classes consisting mostly of different dog breeds.

Figure 5.10: Two other community examples that corresponds to mostly types of birds or dogs. The communities are to big to show each label in a presentable manner.

The two other examples of clusters shown in figure 5.10, contains mostly classes related to bird types and dog types respectively. The labels are not included in these clusters as there are to many classes causing hard to read figures. In these communities it can also be noted that the vertices that are pulled far apart from the dense area of the hub appears to less connected to the main type of object that the classes represents. To allow for exploration of the communities in the class graph of Imagenet1K, it has been made publicly

available through a external repository[1]. Through this it is possible to explore all communities within the graph, and observe the classes the communities are comprised of.

Hubs

In the degree distribution shown in figure 5.7, there are observed to be two vertices with significantly higher degree than other classes at degrees of 85 and 76 respectively. These classes appears as the big hubs in the lower part of figure 5.6, and corresponds to the *mini-skirt*, and *band aid* class respectively.

The appearance of these hubs is very interesting as it indicates that these classes exists in a central part of the latent representation, where there are several other class representations surrounding them given the high degree. Why exactly these classes are so well connected is hard to state exactly. However, when exploring the images of these classes it can be observed that many of the images has a human in the image in addition to the target class object. The presence of humans as parts of the images can also be observed for many of the classes connected to these hubs in the class graph. A hypothesis is that the human in the images acts as a shared feature between all the classes such that they converge towards a similar, yet distinct area in the latent representation.

5.2.2 Cifar100

The class graph of Cifar100 is is presented in figure 5.11.

The class graph representation for Cifar100 is constructed with the same method as for the Imagenet1k class graph. However, now it is constructed by the latent space for the cifar100 test set on Resnet-18.

From the class graph layout of the Cifar100 class graph (figure 5.11 it can be observed that the network appears to be tied around a few central high degree hub vertices. When observing the degree distribution in figure 5.12, this observation is confirmed. In addition by observing the metrics in table 5.7, it can be observed that the clusering coefficient is found to be 0.43. However, it should be noted that most triangles in the Cifar100 class graph passes through some high degree hub vertices.

1. Imagenet1k class graph (https://github.com/Ivernoerve/imagenet_class_graph)

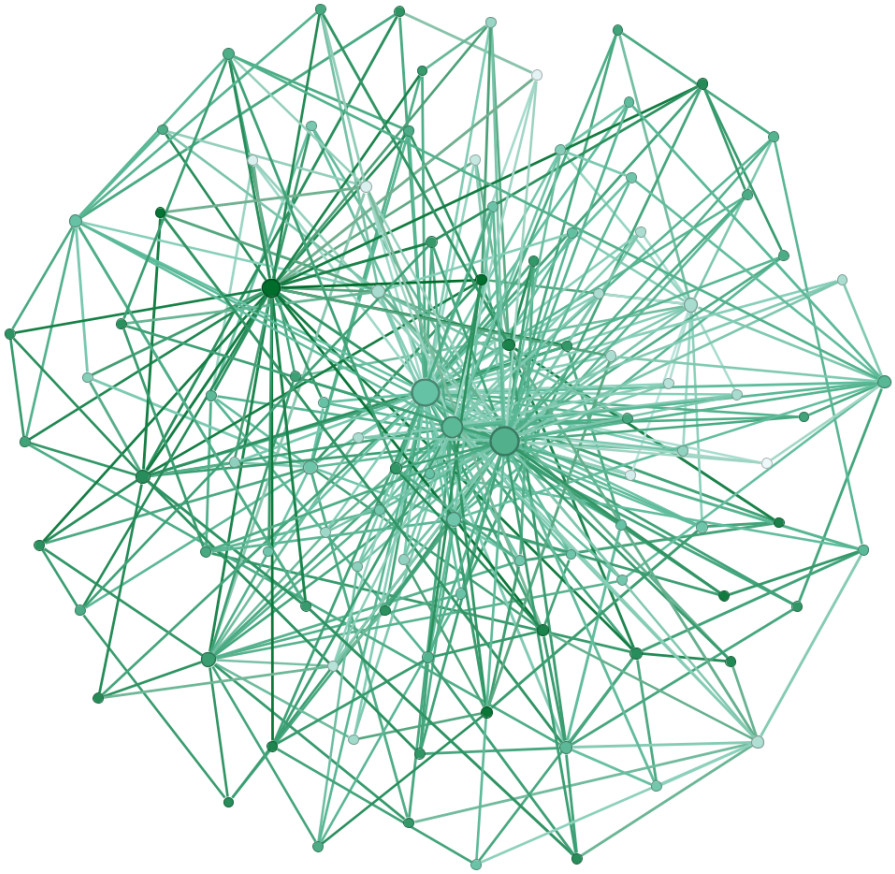


Figure 5.11: The class graph of Cifar100 showing the entire structure of the graph. the size of the vertices are scaled by their degree, and the hue corresponds to the classwise accuracy.

Measure	Value
Diameter	3
Average path length	2
Average Degree	7.7
Clustering coefficient	0.43

Table 5.7: Statistics for the classgraph of Cifar100 (figure 5.11).

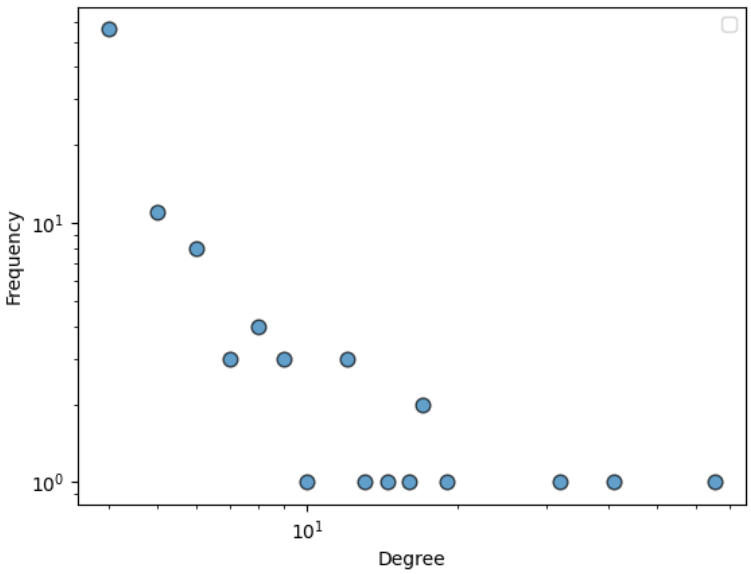


Figure 5.12: Degree distribution of the Cifar100 class graph G_{class} presented on a logarithmic scale

5.3 Exploring the changes in latent representation across an iterative pruning process

In this section we present the findings from analysing the evolution of a Resnet-18 latent space across an iterative pruning, also known as the lottery ticket hypothesis. The results are divided into two subsections, each shows the results for performing the pruning with a high, and a low pruning ratio.

5.3.1 High pruning ratio

The results produced for the high pruning ratio, pruned $p = 20\%$ of the weights for each pruning iteration. We start by presenting the accuracy achieved by the network as well as the ratio of pruned weights in the convolutional layers per iteration of the iterative pruning. Note that iteration 0 corresponds to the unpruned Resnet-18, this will be refereed to as the unpruned state of the model. The accuracies and ratio of pruned parameters are shown in table 5.8.

Iteration	Ratio of pruned weights	Achieved accuracy
0	0	61.8%
1	20%	62.0%
2	36%	61.5%
3	49%	61.5%
4	59%	61.4%
5	67%	61.5%
6	73%	61.3%
7	79%	61.8%
8	83%	61.5%
9	86%	61.4%
10	89%	61.5%
11	91%	61.6%
12	93%	61.5%
13	94%	61.2%
14	95%	61.2%
15	96%	61.3%

Table 5.8: Table showing the ratio of pruned weights and the achieved accuracy on Cifar100 testdata for the model at each iteration of the experiment.

From table 5.8 it can be observed that the model accuracy remains stable across the pruning process when compared to the accuracy of the unpruned state. We find that after 15 iterations of pruning the number of parameters in the convolutional layers gets reduced by 96%. This confirms what is shown prior by [Frankle and Carbin, 2019] that the subnetwork with parameters $\theta_j \odot m$ manages to retain the accuracy of the unpruned state.

For each pruned state we construct the temporal graph G_t for the pruning process with the k nearest neighbours method for $k = 4, 8, 16$. We compare each pruned latent representation ($t > 1$), to the original unpruned latent representation ($t = 0$). Each comparison is performed with the graph average Jaccard similarity and presented in table 5.9. The leftmost column shows the iterations compared.

When observing the results in table 5.9 it can be observed that the graph average Jaccard similarity is low when compared to the unpruned state of the model. Further it can be observed that the similarity decreases when comparisons are made to later iterations, indicating that the loss of active parameters plays a significant role in the convergence in the latent space.

iteration	$k = 4$	$k = 8$	$k = 16$
0, 1	0.2	0.22	0.26
0, 2	0.16	0.18	0.21
0, 3	0.14	0.17	0.2
0, 4	0.13	0.16	0.19
0, 5	0.13	0.15	0.18
0, 6	0.13	0.15	0.18
0, 7	0.12	0.15	0.18
0, 8	0.12	0.15	0.18
0, 9	0.12	0.14	0.17
0, 10	0.12	0.14	0.17
0, 11	0.12	0.14	0.17
0, 12	0.12	0.14	0.17
0, 13	0.12	0.14	0.17
0, 14	0.12	0.14	0.17
0, 15	0.12	0.14	0.17

Table 5.9: Table showing the graph average Jaccard similarity for all pruning iterations with a pruning ratio of $p = 20\%$. All states are compared to the unpruned state.

With respect to k it can be observed that the similarity increases slightly for each doubling of k. This increase in similarity with respect to k, could indicate that only the local neighbourhoods are significantly changed, whereas the global structure of the graph in terms of clustering properties is retained. To explore this further we observe the evolution in the class neighbourhood similarity shown in figure 5.13.

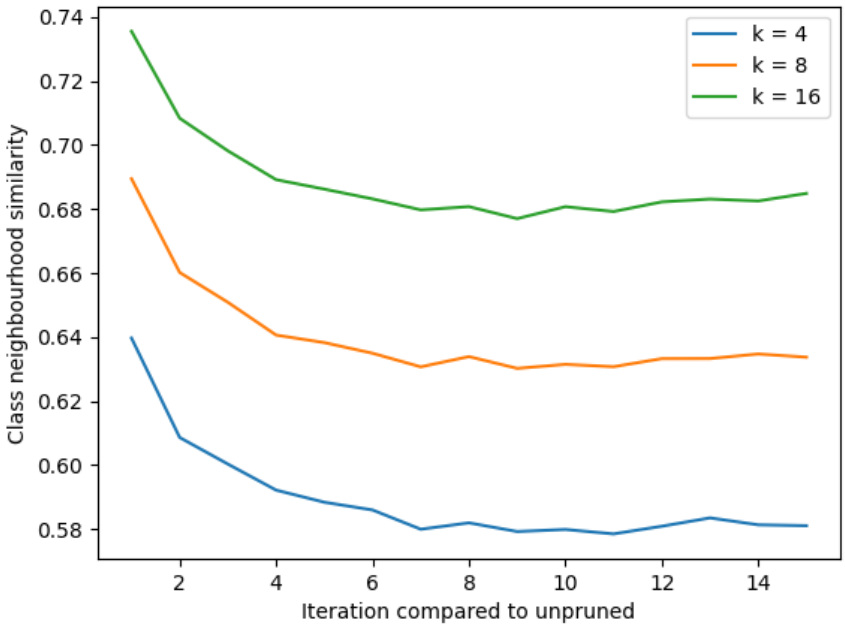


Figure 5.13: plots showing the Class neighbourhood similarity between the unpruned state G_0, and each pruned iteration G_t. The x axis shows the iteration t used in the comparison.

Initially when inspecting the class neighbourhood similarity in figure 5.13, the same small increase in similarity with respect to k can be observed. Further the figure shows a similar trend as the graph average Jaccard similarity in table 5.9 with a small decrease that stabilises after some iterations of pruning. Lastly the class neighbourhood similarity is observed to be significantly higher than the graph average Jaccard similarity, giving stong indications that the clustering properties in the latent representation is retained in the iterations t.

5.3.2 Low pruning ratio

When attempting to explain the large dissimilarity when comparing the latent spaces, we hypothesised that the high pruning rate of $p = 20\%$ could drastically change the convergence in the latent space. To test this hypothesis, the experiment is performed again fr a significantly lower pruning ratio $p = 1\%$.

Table 5.10 shows the ratio of pruned convolutional weights and the accuracy for the lower pruning rate. The accuracy remains stable, however, note that the parameters of the convolutional layers in the final state is only reduced with 14%.

The similarity is presented in the same manner as for the high pruning ratio, each iteration is compared to the unpruned state, and performed for a range of k for the graph construction. The results are presented in table 5.10.

Iteration	Ratio of pruned weights	Achieved accuracy
0	0%	62.7%
1	1%	62.4%
2	2%	61.8%
3	3%	61.9%
4	4%	61.5%
5	5%	62.2%
6	6%	61.6%
7	7%	61.8%
8	8%	62.3%
9	9%	62.8%
10	10%	62.2%
11	10%	60.5%
12	11%	61.8%
13	12%	62.6%
14	13%	61.6%
15	14%	62.2%

Table 5.10: Table showing the ratio of pruned weights and the achieved accuracy on Cifar100 testdata for the model at each iteration of the experiment.

iteration	$k = 4$	$k = 8$	$k = 16$
0, 1	0.19	0.21	0.25
0, 2	0.19	0.21	0.25
0, 3	0.19	0.21	0.25
0, 4	0.19	0.21	0.25
0, 5	0.19	0.21	0.25
0, 6	0.19	0.21	0.25
0, 7	0.19	0.21	0.25
0, 8	0.19	0.21	0.25
0, 9	0.19	0.21	0.25
0, 10	0.19	0.21	0.25
0, 11	0.19	0.21	0.25
0, 12	0.19	0.21	0.25
0, 13	0.19	0.21	0.25
0, 14	0.19	0.21	0.25
0, 15	0.19	0.21	0.25

Table 5.11: Table showing the graph average Jaccard similarity for all pruning iterations with a lower pruning ratio of $p = 1\%$. All states are compared to the unpruned state.

Observing the results for the lower pruning rate seen in table 5.11, a similar result as for the high pruning rate can be observed. The graph average Jaccard similarity is also here found to be low when compared to the initial unpruned state. However a small increase in similarity when comparing to the high pruning rate is observed, this holds true for all values of k.

When comparing this to the results found for a high pruning rate there are some noteworthy observations to be made. Firstly it can be observed that the graph average Jaccard similarity is constant across all comparisons for the low pruning rate, as opposed to the decrease observed in the high pruning rate. Further the stable similarity in the low pruning rate is very similar to the initial similarity observed for the high pruning rate, indicating that the smaller decreases in active parameters is better at retaining representational knowledge of the latent space. To further assess the similarities for the lower pruning ratio, the class neighbourhood similarity presented in figure 5.14 is inspected.

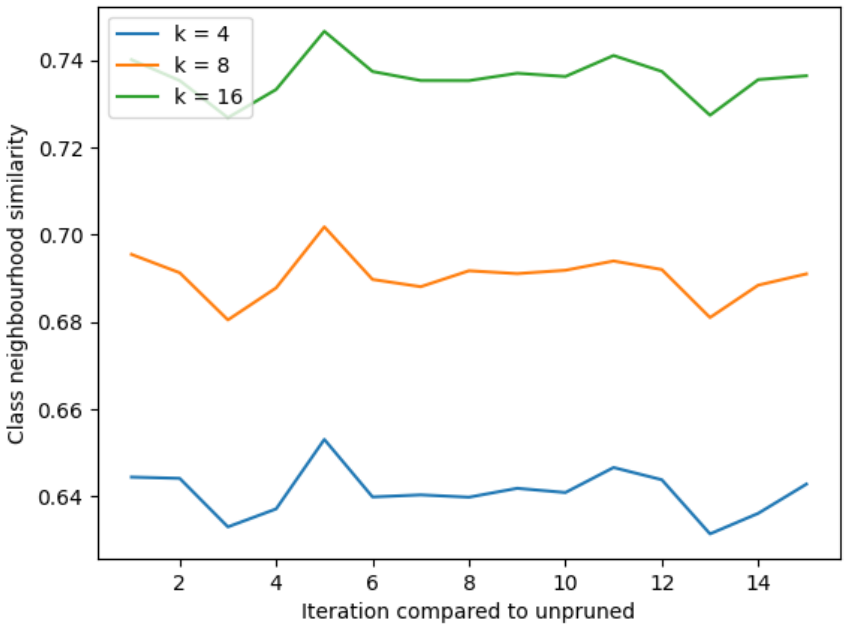


Figure 5.14: plots showing the Class neighbourhood similarity between the unpruned state G_0, and each pruned iteration G_t. The x axis shows the iteration t used in the comparison.

When inspecting the class neighbourhood similarity, it can be observed a similar trend of stability as for the graph average Jaccard similarity. The small increases in class neighborhood with respect to the doubling of k is also observed in this figure. The similarities are observed to be stable around the highest similarity observed for the high pruning ratio 5.14, further indicating that the choice of pruning rate affects the models ability to preserve information of the latent representation of the unpruned model.

6 Discussion

In this chapter we will discuss the results presented in chapter 5 relating the separate experiments together, while attempting to highlight the connections between them and the insight they give us about the latent space. Further this section will discuss challenges, and address the limitations of the work presented.

6.1 Latent space regions and their complexity

When exploring the latent space of deep neural networks, it is important to consider the various regions the representations are defined in. Investigating the properties of these regions can bring insight on the models ability to represent the patterns found within the data. For a model trained for a classification task we find properties (section 5.1) indicating that the latent space can be roughly divided into two region types; namely well defined, and mixed regions. Both regions are of interest to detect, as both can contribute towards understanding the models ability to handle the complexity posed by the dataset in terms of finding patterns withing the distributions.

well defined region: A well defined region is comprised of representations from samples that stems from mostly the same class present in a distinct area of the latent space isolated from the other classes. Such regions can indicate

that the model has learned good patterns for segregating the class. These are the desired regions to have the latent space be comprised of, as they indicate that the model is capable of finding distinct patterns to separate classes into distinct regions creating clear separations.

Mixed regions: A mixed region is comprised of representations of samples that stems from a range of different classes. Such regions can indicate challenges posed by the distributions in the dataset, such as an ambiguous pattern that is shared between several classes. These can be formed for several reasons, such as the lack of patterns capable of separating the data. Similarly mixed regions may form if the model is not strong enough to detect deeper complicated patterns in the data. We contemplate a third factor that can form mixed regions in classifiers tasked to separate many classes, based on the meaningfulness of the patterns in terms of the scope of separability. Some ambiguous pattern present within several classes may aid in separating these classes from the other classes, however within the classes sharing the ambiguous pattern, it should be useless. An observation connected to this third reason is elaborated on in the last paragraph of the next section.

6.2 The intuition of the class graph

The Imagenet1k class graph and the following exploration of it presented in section 5.2, gives a strong indication that the network-science driven method is capable of capturing meaningful, and intuitive insight in the latent representations. Even though the latent representations are very high dimensional at $\mathbb{R}^{512}$, the class graph shows tendencies of well defined and denser clusters that contains many similar classes. Considering the known problems when working with high dimensional spaces, the class graph shows that the proposed network-science driven method has a promising potential of capturing relational mappings in higher dimensional spaces.

The clusters found through the Louvain community detection further supports the observed clustering properties in the space, where the clusters found are dense within with fewer outgoing edges to other clusters (figure 5.8). The clusters observed through the Louvain method is not necessarily a concrete evidence for the existence of clusters due to the algorithm. As the algorithm aggregates in a greedy manner, the initialisation is important for the convergence. Further the choice of resolution in the algorithm can affect the number of clusters. Here it was run with a standard value of 1 such that neither bigger or smaller clusters were favoured. However, as reported, many of the dynamically detected clusters found by Louvain does appear in agreement with intuition

when inspected the class labels forming the clusters.

From the class graph (figure 5.6) the presence of hub classes could be observed. The neighbourhoods of the two hub classes, namely *miniskirt*, and *band-aid* are found to share a common feature in many of the images for the classes. This feature, namely a human being, or parts of a human could be the reason for this observed hubbness phenomenon. During the learning process, the model should learn to discard the human feature as many classes share this feature. However, the classes are observed to be present in a similar region of the latent space, leading to the hypothesis that the human feature can be a factor for the observed hubness-phenomenon. As Imagenet1k contains 1000 classes, it might be that the human feature aids in the separation of classes often related to humans against other classes such as the dogs and birds. Thus on a global scope between separating all classes, the human feature can be important. However, within the local hub region it should potentially not be considered, as all classes in the regions shares the feature.

As we are observing the latent representation between the convolutional encoder and the fully connected layer of the Resnet-18, the fully connected layer could potentially have a problem in ignoring the activations caused by the shared feature found by the convolutional encoder of the network. As each element in the latent representation is computed as the average pool of each convolutional filter, a shared activation within a filter can have a significant impact on the value given by the pooling operation. This potential problem could be one of the reasons we observe the hubs in the class graph around the shared human feature.

6.3 Observation scope, and the selection of k

From the results shown in this thesis, some considerations regarding the selection of k should be considered. The results presented has shown some clear trends with regards to similarity with respect to k. If the goal is to observe the evolution of the latent representations on a local scale, then it can be argued that constructing G with a smaller k is favourable. It was observed that the graphs created with $k = 4$ has a high potential for capturing similarities with regards to the Jaccard similarity (figure 5.2). However, it is very susceptible to noise, such that small changes can be have great impact on the small neighbourhood and thus easily detect local structural changes.

However, if the exploration revolves around observing larger less strict structures, i.e not comparing direct neighbours but comparing the properties of neighbouring vertices, the selection of a larger k for constructing G seems

favourable. This can be observed in the results showing the class neighbourhood similarity, both with respect to the injection of noise (figure 5.3), and with respect to the iterative pruning performed in the lottery ticket hypothesis (figures 5.13, 5.14).

From the discussion above there is no clear indication of the best value of k. Rather the value of k should be carefully chosen based on the property to be observed. A small k is good for strict comparisons and for observing changes in the smaller local scopes, however, fails to explain bigger structures in the graph. A larger value of k is observed to perform better for investigating larger topological structures such as clusters and regions. The graph constructions made with a larger k is however found to struggle with capturing the smaller local structures in the graph.

6.4 Similarity measures, and the transferability of latent representations

For the analysis done in this thesis, the similarity between two latent representations has been measured with regards to two graph similarity measures; namely the graph average Jaccard Similarity (table 5.4), and the class neighbourhood similarity (table 5.5). The Graph average Jaccard similarity is a strict measure of equality, as it requires the neighbourhood of a vertice to be comprised of the same vertices to get a high similarity. With regards to observing clustering properties in the latent space, this measure can be too restrictive and thus fail to find transferable representations between two latent representations. This argument is made very clear across several experiments, and is observed with regards to perturbation (section 5.1), training comparisons, as well as pruning both small and large percentages of weights (section 5.3). The graph average Jaccard similarity is observed to decrease rapidly, or systematically result in a low value, across several values of k. All these experiments points to the fact that small changes, such as perturbation, or pruning, can cause significant changes in the local neighbourhoods in the models latent representation.

For this reason less strict measures of similarity can be advantageous, as they are observed to capture more transferable patterns in the latent representations. When observing properties and topology of the latent representations, less strict similarity measures has better potential for observing broader patterns such as class clustering and class bordering. If within a well defined class cluster, small changes can rearrange neighbours entirely. However, the bigger cluster structure could be observed to be retained through only comparing the label

property of the vertices, rather than the direct neighbours. Likewise, this is of interest when observing and compering mixed regions present between well defined clusters two clusters. These mixed regions will then have a class neighbourhood consisting of several classes, thus a small rearrangement could impact the class neighbourhood. For this reason a measure such as the class neighbourhood can convey information on a broader, larger pattern that is more transferable between latent representations.

6.5 A note on the lottery ticket hypothesis

With the information given by comparing the stability of the latent representation during training (table 5.2), where a high similarity between the sequential latent spaces at epochs $t, t+1$, was observed, it was hypothesised that there would be similar results when iteratively pruning the weights. However, as our results shows with regards to the evolution of the latent representations for a high pruning rate: Both the graph average Jaccard similarity (tables 5.9, 5.11) and class neighbourhood similarity (figures 5.13, 5.14) between the unpruned and pruned states, shows a similarity trend that converges towards the similarity found between two independently initialised models (tables 5.4, 5.5). With this observation in mind it could be hypothesised that the latent representation of the sub-network retains core properties such as the clustering properties of the original state, however has properties unique to itself.

From the results found with the lower pruning rate, there is no clear decreasing trend in the similarities with respect to the pruning iterations. Rather it stays at a constant value similar to the similarity found for the comparisons with the first pruning done for a high pruning ratio. Even a small change in the available parameters of the model, is observed to seemingly change representations in a significant way. This could support the indication that the model will diverge from the original representation as more parameters are pruned.

From the arguments made above about the latent representation convergence towards it's own distinct representation as the ratio of pruned weights increase, there is an interesting point to be made about the iterative pruning method that was proposed. Given a more complex problem, i.e a dataset that has deeper patterns, it is important to have a model capable of capturing them, such that the latent space can be arranged into well defined regions. For such a complex dataset, a weaker model could struggle to find the underlying patterns, such that several mixed regions would be present in the latent space. How does such an increase in data complexity change the pruning methods ability to reduce the networks parameters? Given that the sparse sub-network found through the pruning appears to converge towards it's own distinct latent representation,

will the lottery ticket hypothesis method be able to yield equally impressive reductions when finding this sub-network when posed with the harder, mixed representations of data? The class graph representation of the Cifar100 dataset (figure 5.11) indicates that the classes appear to exist in distinct regions of the latent space, yet with a high connectivity towards a few classes in the core of the latent space connecting them. This is pointing to the fact that the Cifar100 is not a very complex dataset in terms of defining clusters for the classes.

If it is the case that a more complex dataset reduces the lottery ticket hypothesis ability to reduce network parameters with an equally impressive amount, then the whole idea of the 'winning ticket' initialisation could be questioned. A 'winning ticket' would in such a case be restricted by the data complexity, such that easy to find patterns must be present in the data if the sparse sub network is to retain the performance of the original unpruned network.

6.6 Challenges and limitations

In this section we will discuss challenges and limitations with the work shown in this thesis. The limitations presented here could potentially impact the robustness of the discoveries found in the latent representations.

6.6.1 Challenges

At the start of this thesis we worked on the Imagenet1k dataset. This is well known to be a computationally demanding dataset with it's 1.28 million 224×224 pixel images. This in addition to working with the iterative pruning scheme proposed by [Frankle and Carbin, 2019] which requires many sequential trainings of the network, made the initial experiments time consuming to run as a result of the combined computational complexity. Unfortunately the training on Imagenet collapsed, rendering the results useless in terms of analysing the latent space.

Another challenge posed during the work is the computational complexity of finding the shortest paths: Dijkkstra's shortest path algorithm has a time complexity of $O(|V|^2)$, where V is the set of edges in a graph G. Performing this for each pair of vertices in G will scale very badly with the number of vertices in G because of the quadratic time complexity scaling. Thus, results such as the class graph G_{class} which relies on having all pair shortest paths in $G = (V, E)$, quickly becomes computationally expensive as $|V|$ increases.

6.6.2 Limitations

Model selection: The results shown from exploring the latent space is restricted to one model architecture, namely the Resnet-18. For this reason the results shown can contain artefacts only present in the Resnet-18's latent space, or at worst, the properties uncovered can be totally unrepresentative for other latent spaces.

Datasets: Similarly only two datasets were used for the experiments. The limited number of datasets could potentially also limit the robustness of the properties discovered, as the observed properties could be artefacts produced by the datasets. Although we explore the latent space of Imagenet1k on the pretrained Resnet-18, the experiments requiring training a model is only performed with Cifar100. Cifar100 is a well known reference dataset, however when compared to Imagenet1k it is a far less complex dataset.

7

Conclusion

In summary, this thesis reviews the findings from exploring the latent space deep neural networks with a network-science approach. The latent space exploration includes the investigation of properties such as class relational representations, regions with well defined clusters, and observing to which extent these properties are transferable between the latent representations of two distinctly trained models. Further the exploration uncovers the stability, and susceptibility of the latent space with regards to perturbation through the introduction of noise, as well as through iteratively retraining and pruning the weights of the model to find sparser sub-networks. As the function constricting the latent space of neural networks is unknown, we approximated the topology considering the k nearest neighbours graph constructed by the latent representations of images given to the neural network. This graph created the foundation for which the exploration was performed.

The class graph created from the shortest paths on the constructed nearest neighbour graph (section 5.2) shows promising capabilities in mapping relational information between the classes embedding in the latent space. The information given by the class graph can help in understanding the embeddings of the latent space and detect bigger regions containing similar classes, as well as denser hub regions.

With regards to stability, we observed the similarity in the latent representations between several epochs when the model has converged in terms of validation accuracy. With respect to both similarity measures, namely the graph average

Jaccard similarity (table 5.2) and class neighbourhood similarity (table 5.3) we discovered high similarities in the range 89 – 93%, 94 – 97% respectively for all values of k used during construction showing that the latent representations converge to a point where only minor rearrangements occurs between epochs. With the same similarity measures, we observed the transferability between the latent representations between two independently initialised models trained with the same method. Here it was observed that the Jaccard similarity gave a low similarity in the range 15 – 21% (table 5.4) increasing with a larger value of k. The class neighbourhood similarity gave a higher similarity in the range 53 – 64% (table 5.5) also increasing with k. These results stresses the importance of selecting an appropriate similarity measure, as well as a suitable k for constructing the graph, depending on the properties of the latent space to be observed. For detecting similar structures i.e well defined regions between two distinct latent representations, the combination of graph constructions with a larger k, and a similarity measure such as the class neighbourhood proves better suited for the task through our results.

Further we observed the susceptibility of the latent representation with regard to perturbation. When introducing Gaussian noise to the latent representation, we discovered that the graph average Jaccard similarity between the original, and perturbed latent representation decreased rapidly (figure 5.2). This finding indicates that the direct neighbouring structure of the latent representations is very sensitive to perturbation. However, from the results from observing the class neighbourhood similarity with respect to the same noise (figure 5.3), it is clear that the neighbourhoods with respect to the class label are less susceptible to the introduced noise. For $k = 16$ we observe a decrease of 12% in neighbourhood similarity when introducing noise with as much variance as is present in the original latent space. This observation indicates a strong presence of separated well defined regions comprised of representations with the same class, such that when noise is introduced, only the structure of direct neighbours are affected. However, the distinct cluster retains it's structure. This observation was further supported when observing the models performance on the noised latent representation (figure 5.4). Only a 4% decrease in accuracy was found when introducing noise with as much variance as is in the system, emphasising the importance of these well defined clusters in the latent representation.

Through exploring the evolution of the latent representation for a network pruned iteratively by the lottery ticket method, we discover some properties that potentially could point to the fact that there is no winning ticket. By observing the graph average Jaccard similarity when pruning with a high pruning rate, we observed that the similarity converged to the range $0.12 - 0.17\%$ depending on k (table 5.9). The similarities found here are lower than the similarity when comparing the latent representations of two unique models for all k (figure 5.5), showing that the pruned model creates it's own distinct latent representation

through the pruning. The Class neighbourhood through the pruning had a similar trend, where it was observed to decrease during the iterative process (figure 5.13). By using a lower pruning rate between each iteration of the lottery ticket, we observe that both the graph average Jaccard similarity (table 5.11) and the class neighbourhood similarity (figure 5.14) remain stable across all iterations. Given that the sparse sub network appears to converge to its own latent representations, the concept of a 'winning ticket' could potentially be questioned, as the sub networks ability to retain the performance of the original network could rely on the complexity of the patterns in the data.

7.1 Further work

With the limitations of this work in mind discussed in section 6.6, there are several interesting aspects of the latent representations to uncover. Firstly, it is crucial to perform the experiments on a wider combination of network architectures and datasets. This is necessary to establish whether the properties observed in the latent space through our work is universally applicable. Before such a wider exploration is performed there is not enough evidence to propose any hard conclusions related to the latent representations of deep learning models.

Many of the performed experiments shows that the local neighbourhood structures in the latent representation are very susceptible to change when perturbed. The perturbation can occur both through noise as well as through an iterative pruning on the network. However, it was observed that the network is able to retain most, if not all of it's accuracy after such a perturbation even when the latent representation appears severely different with respect to the original unperturbed network. Through the experiments performed in this thesis the similarity has been observed, with respect to the strict neighbourhood of vertices by the Jaccard similarity, and the less strict class neighbourhood similarity. Both of these measures are measures similarities on a local scope and aggregates an average for the entire network. It is yet to compare the models with a global measure, such as by observing the similarity of the close to zero Laplacian eigenvalues of the graphs. By observing the spectral similarity, similar properties could be compared on a global scope between two latent representations.

From what has been observed when comparing the results from the iterative pruning, it could be of interest to test the lottery ticket hypothesis on a more complex dataset such as Imagenet1k. Is it still possible to prune 90% of the convolutional weights when attempting to separate a more complex dataset, or will it be observed to collapse for a lower ratio of pruned weights? It could

be that the hypothesis only works on datasets that are easy to embed in the latent space as defined clusters for each class, and thus fails to achieve the same results for more complex and messy data.

While discussing the potential use cases of the class graph tool, we theorised on using the regional information given by the graph for constructing region specific classifiers for difficult regions similar to the Mixture of Experts [Jacobs et al., 1991] idea. By constructing an architecture consisting of a router model together with several smaller models that are trained on local regions, we could potentially leverage the patterns within the on a global and a local scale. The router model would consider the input with respect to all classes, select the region it belongs to, and send it to the 'expert' model. The 'expert' networks would be trained on a subset of classes from the respective region on the class graph it is responsible for. Given the discussion on hubs in the class graph (observed in figure 5.8) such a mixture of experts architecture could potentially be better at exploiting the patterns relative to the scope. The routing model could learn from the patterns shared between classes to perform a global scope assertion on the respective region, while the expert models could learn patterns to distinguish the classes within the region. Such a mixture of experts architecture would be more computationally expensive than a single model that considers all classes. However, the smaller expert models could potentially learn better representations for separating the data within the region that the model considers as it would not be influenced by the patterns important on a global scale between all classes.

Bibliography

[Aggarwal et al., 2001] Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *Database Theory—ICDT 2001: 8th International Conference London, UK, January 4–6, 2001 Proceedings 8*, pages 420–434. Springer.

[Alpaydin, 2020] Alpaydin, E. (2020). *Introduction to machine learning*.

[Bastian et al., 2009] Bastian, M., Heymann, S., and Jacomy, M. (2009). Gephi: an open source software for exploring and manipulating networks. In *Third international AAAI conference on weblogs and social media*.

[Blizard et al., 1989] Blizard, W. D. et al. (1989). Multiset theory. *Notre Dame Journal of formal logic*, 30(1):36–66.

[Blondel et al., 2008] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

[DIJKSTRA, 1959] DIJKSTRA, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

[Eppstein et al., 1992] Eppstein, D., Paterson, M., and Yao, F. (1992). On nearest-neighbor graphs. *Lect. Notes Comput. Sci.*, 17:263–282.

[Fraenkel et al., 1973] Fraenkel, A. A., Bar-Hillel, Y., and Levy, A. (1973). *Foundations of set theory*. Elsevier.

[Frankle and Carbin, 2019] Frankle, J. and Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Hagberg et al., 2008] Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring

network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Jacobs et al., 1991] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

[Jacomy et al., 2014] Jacomy, M., Venturini, T., Heymann, S., and Bastian, M. (2014). Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PloS one*, 9(6):e98679.

[Koutroumbas and Theodoridis, 2008] Koutroumbas, K. and Theodoridis, S. (2008). *Pattern recognition*. Academic Press.

[Krizhevsky et al.,] Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research).

[Latapy, 2008] Latapy, M. (2008). Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theoretical Computer Science*, 407(1):458–473.

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

[Meidiana et al., 2019] Meidiana, A., Hong, S., Eades, P., and Keim, D. A. (2019). A quality metric for visualization of clusters in graphs. *CoRR*, abs/1908.07792.

[Nørve, 2023] Nørve, I. S. (2023). Exploration of the latent representations in deep learning models. 1.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge.

International Journal of Computer Vision (IJCV), 115(3):211–252.

[TT, 1958] TT, T. (1958). An elementary mathematical theory of classification and prediction. Technical report, Internal IBM Technical Report.

[Zhang and Chartrand, 2006] Zhang, P. and Chartrand, G. (2006). *Introduction to graph theory*. Tata McGraw-Hill.