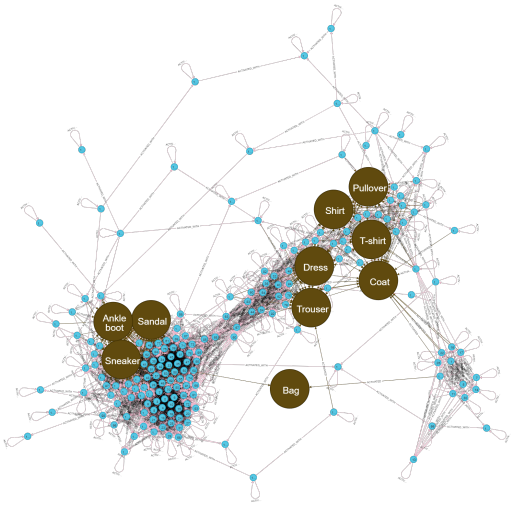
Extracting Semantic Hierarchies from Trained Neural Networks: A Compression Perspective

submitted in partial fulfillment for the degree of master of science

Hristo Petkov
13369776

master information studies
data science
faculty of science
university of amsterdam, braincreators

2021-06-28



	Internal Supervisor	Examiner
Title, Name	Erman Acar	Dr. Maarten Marx
Affiliation	VU	UvA, FNWI, IvI
Email	erman.acar@vu.nl	m.j.marx@uva.nl

	External Supervisor	Third Supervisor	Fourth Supervisor
Title, Name	Maarten Stol	Stijn Verdenius	Vitor Horta
Affiliation	BrainCreators	BrainCreators	Insight: SFI Research Centre
Email	maarten.stol@braincreators.com	stijn.verdenius@braincreators.com	vitor.horta@insight-centre.org

Universiteit van Amsterdam

Extracting Semantic Hierarchies from Trained Neural Networks: A Compression Perspective

Author: Hristo Petkov
University of Amsterdam
Amsterdam, the Netherlands
petkov.1231@gmail.com

ABSTRACT

In this thesis, we explore class hierarchies extracted from trained neural networks. They hold information on how a ConvNet groups output classes in terms of semantic similarity. We compare the class hierarchies obtained from different models on the same dataset. Evaluation of hierarchies is based on a dissimilarity metric from phylogenetics. We explore the impact of neural network compression ratios on the quality of extracted class hierarchies. Our results show that the drastic changes in a model's class groupings are usually in the beginning stages of training. Additionally, we explore different factors that affect the quality of the extracted hierarchies. We indicate that the initial model size has a more prominent influence than the model performance. Finally, we compare our pruned models to their unpruned variants. We demonstrate that some sparsity ratios produce a decreased class hierarchy comparison metric in favor of the compressed ConvNets. From this, we conclude that neural network compression can improve the quality of the extracted class hierarchies.

KEYWORDS

convolutional neural networks, neural network graph extraction, hierarchical community detection, phylogenetic graph comparison, neural network compression, structured magnitude pruning

1 INTRODUCTION

When the goal is to produce as many correct predictions during inference as possible, deep neural networks (DNN) are often the preferred option. They confidently outperform the other machine learning models in terms of accuracy in various domains, such as computing, science, engineering, medicine, agriculture, technology, business, arts [1]. Consequently, state-of-the-art (SOTA) machine learning models in subfields of artificial intelligence (e.g. computer vision or natural language processing) are DNN-based [2]. However, from an application point of view, models performing comparably to SOTA require substantial additional resources, such as time, energy, hardware, and memory, which forces many real-world projects to opt for simpler algorithms even if that means a reduced inference accuracy.

Another concern regarding DNNs is the lack of explainability when it comes to how a particular prediction is derived from the input data. Several sectors, such as banking or healthcare, require clarity and transparency for each decision that affects people or businesses. For instance, if physicians have the decision lineage data, they can understand how the results were reached and determine whether human interference is necessary. Additionally, when underwriting a loan, it should be possible to inform the requester of the reasons for the final decision[1].

In this thesis, we extract class hierarchy trees of a dataset as defined by the trained DNNs. This ontology shows how the corresponding DNNs connect the output classes. It also illustrates which classes are more similar to one another according to the model. We call these ontologies the observed hierarchy graphs. As different models produce modified class hierarchies, we are interested in comparing them. Therefore, we define another class hierarchy, which serves as our ground truth. Then, we can compare several observed hierarchy graphs with the true hierarchy graph and determine which DNN produces the closest ontology to the ground truth.

Before continuing further, we need an algorithm for creating class hierarchies from trained DNNs. A suitable option is co-activation graphs [3], which addresses the issue of explainability. The authors of [3] propose an algorithm for extracting graphs from any DNN, where the nodes correspond to the model's neurons. Furthermore, they explain that the output nodes [2] within a co-activation graph are organized in communities. These communities represent how the underlying DNN would group the classes. Then, the authors of [3] argue that the output nodes also tend to show hierarchical structure. As a result, we can extract a smaller graph from a co-activation graph that encodes the relationships between the classes, which serves as the observed hierarchy graph. Then, we are ready to compare class ontologies as long as we choose a dataset for which we know the true hierarchy graph in advance.

Creating co-activation graphs suffers from scalability issues. Furthermore, our results indicate that smaller co-activation graphs tend to produce purer class hierarchies. Plus, we hinted that a one-to-one mapping exists between neurons of a DNN and nodes of the co-activation graph created by the same DNN. Therefore, reducing the size of a DNN can considerably influence both the observed hierarchy graph and the practical feasibility of the whole system. We opt for neural network compression when exploring how sparsity in DNNs affects graph extraction.

Naturally, there is a trade-off between accuracy and sparsity. Because DNNs are highly overparametized [4] over 90% of the weights can be removed with almost no degradation in performance [5]. In some cases, pruning even over 99.6 % of a network is feasible [6]. There is usually an optimal sparsity threshold that depends on the dataset, the DNN structure, and the pruning algorithm. Compression less than that would either result in slight or no performance

[1] https://enterprisersproject.com/article/2019/5/explainable-ai-4-critical-industries
[2] An output node is a node that corresponds to an output neuron from the DNN used for creating the co-activation graph

decay. However, pruning over that sparsity level generally means substantial performance degradation.

Therefore, our research questions will focus on:

How do the neural network compression ratio and its trade-off with model performance affect class hierarchy extraction through neural co-activation graphs?

- *How can we compare the true class hierarchy to the class hierarchy extracted from a trained DNN by co-activation analysis?*
- *How does the performance of DNNs influence the quality of the extracted class hierarchy obtained from the hierarchical community detection on co-activation graphs?*
- *How does neural network compression affect the quality of the extracted class hierarchy when comparing it to the true class hierarchy? Which factors (model performance, percentage of network sparsity, initial model size) are most influential?*
- *Which patterns can we observe from the interaction between accuracy, compression ratio, and hierarchical graph quality?*

The rest of the thesis is structured as follows. We provide additional information about our separate components in the second section. The third section summarises the related work. In the fourth section, we explain our experimental setup. In the fifth section, we present our results. Then, we discuss our findings in terms of the research question and subquestions. Finally, in the seventh section, we present our conclusion and discuss future work.

2 BACKGROUND

Neural Network Compression. Compressing a DNN refers to reducing the size of an existing network either before, during, or after training. Additional fine-tuning might be necessary to obtain a DNN with converged and restored accuracy. The pruning can either be applied once (one-shot) or repeatedly at different points of training. In the latter case, a small percentage of the DNN is removed several times until the desired sparsity level is reached. Furthermore, compression can occur at connection level (unstructured) or neural level (structured). For example, in structured pruning, a whole neuron is removed by deleting all its incoming and outgoing connections. Different metrics (saliency criteria) exist for evaluating which weight or neuron should be removed.

Graph Extraction. In this thesis, we refer to graph extraction as the possible ways of creating graphs from the topology of a trained DNN [3]. Generally, the nodes of these graphs are the neurons of a model. The difference comes from the methods for creating the edges. In co-activation graphs, the vertices are connected by the statistical correlation between their corresponding neurons. Hence, a DNN is trained to completion, a forward pass is carried out with the whole dataset, and for each neuron n different activation values are obtained, where n is the size of the dataset. Then, the edge weight between two nodes is the Spearman's correlation between the activation values of the corresponding two neurons. The result is a complete graph with edge weights in the range of -1 and 1. Additionally, the authors [3] propose removing connections with values under a certain threshold.

Community Detection. We now give more information on the community detection methods, which are essential for grouping co-activation nodes and creating the hierarchy graphs. The algorithm we chose revolves around the stochastic block model (SBM) [7], which clusters vertices in groups with arbitrary probabilities of connections between one another. Hence, it is crucial to choose the appropriate model for the corresponding network as they produce different blocks (clusters). Similar to machine learning, the terms overfitting and underfitting represent that a more complex SBM can capture more advanced patterns in graphs but can also overfit the data. Conversely, a simpler model is less likely to cause such problems, but it might miss some advanced graph patterns.

Graph Comparison. Lastly, we provide some information about hierarchy graphs and phylogenetic trees [8]. Our hierarchy graphs are similar to hierarchical organizational charts, where the root includes all the output classes present in the dataset. Consequently, the nodes in the first level are the ones directly connected to the root. They represent non-overlapping subsets of the output nodes, and merging the vertices in the first level is equivalent to the root. These rules are preserved on every level of the tree. Therefore, the leaf nodes and the edges define the hierarchy graph because the attribute of each non-leaf node can be derived from the union of the output classes present in its child vertices.

The current definition of hierarchy graphs resembles phylogenetic trees. The latter is used in biology to model evolutionary relationships between species. Because of their extensive application, a considerable amount of research [9] has been focused on phylogenetic trees and how to compare them. Several libraries [3] are available to feasibly compute a dissimilarity measurement between two phylogenetic trees, which allows us to compare hierarchy graphs.

3 RELATED WORK

Combining neural compression and graph extraction is a novel approach to the best of our knowledge in exploring class hierarchical structure. Hence, no previous research is available on this particular topic. Nevertheless, extensive related work is present for the separate components of this thesis - graph extraction, neural network compression, community detection, and graph comparison.

3.1 Graph Extraction

The research in the field of graph extraction from a DNN is rather scarce. Apart from the work on co-activation graphs, we were able to find only one more paper that creates graphs from DNNs. In the original work on building graph representations from embeddings [10], the authors construct a graph with positive and negative nodes per neuron, which includes relevancy by the presence and absence [10]. Additionally, they add vertices per input data entry, which allows the linking between neurons and the output classes. Then, they define positive and negative connections where an edge exists if the spatial average pooling produces a value over a certain threshold or under another one for positive and negative links, respectively.

Another approach is to create co-activation graphs [3], which are topologically different from the previous algorithm as a connection exists between any two nodes independently of their position in the DNN's layers. The authors show that these graphs can encode

[3] https://dendropy.org/

similar knowledge as DNNs. They also use community analysis to demonstrate that the output nodes are structured hierarchically. Then, a centrality analysis illustrates that a considerable amount of nodes in the co-activation graphs are redundant as they hardly contribute to the formation of communities. Furthermore, in their previous work [3], they show that activating only a few concrete neurons can result in a DNN returning a particular class [11]. In [3] it is also argued that creating a pruning algorithm from the centrality measure might be beneficial. Additionally, the authors claim that most of the mistakes in DNNs are caused by classes with many shared neurons (overlapping nodes in the co-activation graph).

3.2 Neural Network Compression

Early attempts for compression of a DNN include the work from 1989 [12], where the second derivative of all parameters is used to compute saliency scores. Additionally, the paper from 2005 [13] resolves to estimating the relevancy of each neuron by multiplying their sensitivity value by the sum of the absolute values of their outgoing weights. Although this hints towards structured compression, the exact terminology is later defined in [14]. The latter work suggests that pruning can be applied to CNN filters by measuring their importance based on incoming weights. Such algorithms are known as structured magnitude pruning and have shown to produce strong results [15]. They compute the saliency scores by aggregating incoming and outgoing weight values of a neuron. Because in structured pruning some of the removed connections of the compressed neurons might be useful, group sparsity inducing regularization is often included [16, 17].

The alternative is to prune the weights of a DNN. For example, SNIP [18] is an approach where the saliency criterion is based on connection sensitivity, which allows finding appropriate weights for the values of the loss function. Remarkably, the proposed compression algorithm is the first to operate before training. Hence, it can reduce the training time substantially. Another approach addresses the issues of signal propagation of SNIP's sensitivity criterion [19]. The authors show that iteratively applying the sensitivity criterion in smaller steps yields significant performance improvement. Furthermore, they propose a second algorithm based on SNIP that prunes structurally. Again, both operate before training.

Apart from choosing the appropriate saliency criterion, many sparsity-inducing regularizers exist for structured pruning. For instance, ℓ_1 regularization synergizes well with unstructured compression because it forces the weights towards zero. Plus, it avoids overfitting. However, for structured pruning, ℓ_1 does not address that removing a neuron, deletes several connections. A solution can be the group Lasso regularization [16]. However, the authors of [20] argue that ℓ_1 might sacrifice the flexibility of the pruned model. They propose a Hoyer-based regularization [17] that applies to structured compression, too. At its core, it offers a more scalable alternative to the otherwise desired bayesian L0 regularization [21].

3.3 Community Detection

When considering the SBM for community detection, the first distinction between the different algorithms is whether they can infer the optimal number of communities. One variant that requires previous information is presented in [22]. It creates clusters with nodes which neighbors show similar patterns to their corresponding neighbors.

In most cases, the number of communities is not known beforehand. Model selection refers to the algorithms used to determine the appropriate number of communities. Some directions include the Akaike information criterion [23] which is prone to overfitting [24], the Bayesian-based selections [25, 26] and the minimum description length (MDL) [27].

A limitation to the MDL model selection is that it produces up to $\sqrt{N}$ clusters with N being the number of nodes in the graph. Hence, smaller communities are likely to be merged into a sequent block [4]. An improvement is the nested variant of the SBM [27], which can increase the maximum number of detectable clusters to $\log N$. Furthermore, this algorithm allows the partitioning of communities into a well-defined hierarchy.

The necessary prerequisite for the successful execution of the nested-SBM is that the edge counts of a graph can form a multigraph [27]. In that case, the nodes are the blocks, and the connections are the edge counts. In other words, the communities are extracted from the SBM. Then, the link between cluster A and B is created by using the edge counts of the vertices present in A and B. The result is a smaller graph on which the SBM is again recursively applied until there is only one block (the root).

3.4 Graph Comparison

Determining the difference between two graphs is equivalent to measuring how far they are from being isomorphic [28]. Additionally, isomorphism has been a problem for decades, which was recently proven to be solvable in quasi-polynomial time [29]. Yet, including contextual information can drastically decrease the time complexity [30]. In the case of phylogenetic trees, one of the most popular metrics is the Robinson-Foulds (RF) [31]. It produces a discrete number which indicates the minimum number of changes of a graph before it becomes isomorphic to another one. From another point of view, this is similar to edit distance [32]. The RF algorithm decreases the previously mentioned quasi-polynomial complexity as its approximations can be solved in sublinear execution time [33].

An improvement in terms of performance is the generalized RF metric [34], which uses global patterns between trajectories. However, this increases its complexity to an NP-hard problem. Several other options are also present. For example, in [35] a graph is decomposed into subsets of pruned trees. Then, a pairing procedure is followed by the actual distance estimation. Alternatively, a more recent study relies on the minimum-weight perfect matching in a complete bipartite graph [36]. This bipartite graph is built from the partitions of leaf pairs from the phylogenetic trees to be compared. The advantage of the proposed method is that minor changes in the target trees lead to small fluctuations in the metric. In our case, we address the problem of evaluating extracted graphs (first research subquestion) by choosing an appropriate distance metric on tree graphs.

[4] Block, community, and cluster refer to the same notion

4 METHODOLOGY

4.1 Datasets

We use two well-known datasets in our experiments - Fashion MNIST [37] and CIFAR100 [38]. Fashion MNIST consists of 70000 grayscale images of clothes, which are grouped into ten classes: t-shirts/tops, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. Each data entry has 28x28 pixel values between 0 and 255. The dataset does not require preprocessing or cleansing. Before training our ConvNets, we divide Fashion MNIST into train, validation, and test splits with 50000, 10000, and 10000 samples, respectively. We refrain from using the test set until we have chosen our hyperparameter settings.

CIFAR100 consists of 60000 32x32 color images, which are split equally into 100 classes. Additionally, the dataset's classes are further grouped into 20 superclasses. In our experiments, we use only 4 of these superclasses (medium-sized mammals, large carnivores, vehicles1, vehicles2) [5], which amounts to 20 classes and 12.000 images. The reasons for this choice will become apparent in section 4.3. For simplicity, we call this subset CIFAR20. Again, the data set does not require preprocessing or cleansing apart from taking the subset for our needs. We again split the data into train, validation, and test sets with 8000, 2000, and 2000 samples, respectively, and we preserve the latter for evaluation.

4.2 Architectures

In this thesis, we introduce two ConvNets, and we select an additional one from existing literature. The main distinction between our three models is the number of neurons present. They directly influence the structure of their corresponding co-activation graph. Our first ConvNet is composed of two convolutional layers with 32 and 64 channels, respectively. We use filters of size 3x3, padding of 1, batch normalization, and max-pooling with a kernel of 2. The dense part consists of two fully connected layers with 64 and 32 neurons and an output layer. Throughout the thesis, we call this model SmallCNN.

Our second model has four convolutional layers with batch normalization and 64, 64, 128, and 128 channels. We again use a filter of 3x3 with a padding of 1. However, the max-pooling is only applied after the second and fourth convolution, which prevents extensive reduction of the latent features. This model also has two fully connected layers with 128 and 64 neurons and an output layer. We call our second model MediumCNN because of its size. Both SmallCNN and Medium CNN use ReLU as their activation function.

The co-activation graph algorithm operates with $O(n^2)$ time complexity, and the nested-SBM has a $O(n \ln^2 n)$ time complexity both depending on the number of neurons. Hence, using SOTA ConvNets is not possible for this thesis, as these methods do not scale to SOTA model sizes. That is why we choose Conv6 [5] as our largest model, which is a reduced version of VGG16 [39]. Altogether, the count of the hidden neurons is 192 for SmallCNN, 576 for MediumCNN, and 1408 for Conv6.

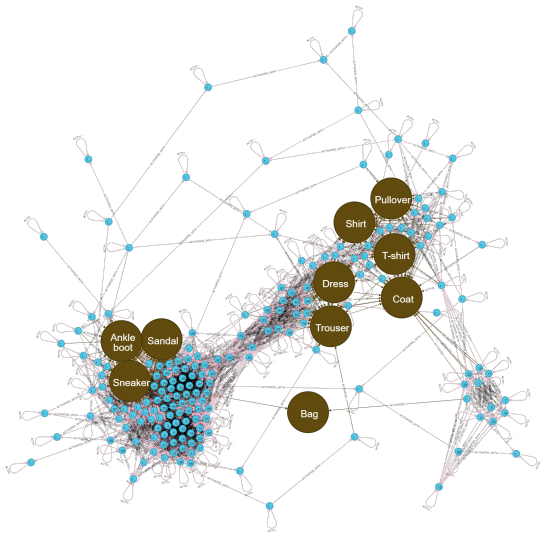


Figure 1: *Co-activation Graph from the Converged SmallCNN on Fashion MNIST. Enlarged nodes correspond to the output neurons, and text depicts the exact class that a vertex represents*

4.3 Experimental Pipeline

4.3.1 *Graph Extraction.*

From all the separate components, graph extraction is the most demanding in terms of time. Because it operates per neuron pair, larger models tend to cause scalability issues. However, at its core, the co-activation graph synthesis computes Spearman correlations between activation values. Therefore, its execution time grows linearly in terms of the dataset size. In our case, the synthesis takes about 8.5 hours for Conv6 on Fashion MNIST and a few minutes for SmallCNN on CIFAR20. Creating co-activation graphs can be expedited by reducing the number of activation values used for calculating the correlations. Nevertheless, to reduce our hyperparameters, we keep every data point when extracting graphs.

The only prerequisite for creating a co-activation graph is the weights of the target ConvNet. Hence, we can save them at different stages of training of our models and extract graphs. This proves to be crucial for answering our second research subquestion, which aims at exploring how accuracy affects class hierarchies. Further discussions about our results are moved to the next section.

The resulting graph from SmallCNN on Fashion MNIST after the last epoch is shown in Figure 1 [6]. The brown vertices are the output nodes, and the text inside them indicates the actual class. Structurally, strongly connected nodes with larger edge weights are grouped close to each other. Even before we apply any community detection on Figure 1, we observe that there is a separation between the classes that represent shoes and clothes. Additionally, the output node bag seems to be separated from the rest.

[5] https://www.cs.toronto.edu/~kriz/cifar.html

[6] Neo4j Visualization - https://neo4j.com/

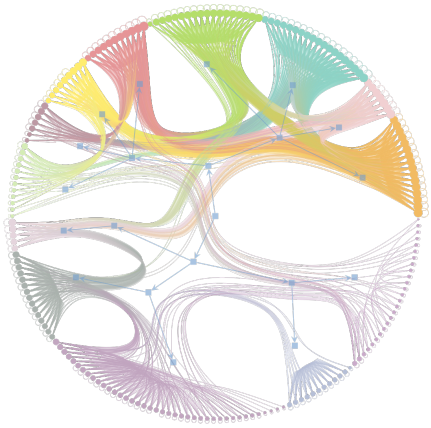


Figure 2: *Hierarchical Community Visualization of the Nested-SBM when Applied on the Graph from Figure 1*

4.3.2 Hierarchical Community Detection.

In this work, we rely on the degree-corrected nested-SBM [27] as our hierarchical community detection algorithm. We also input the edge weights from the co-activation graph in the algorithm.

The nested-SBM produces our extracted hierarchy graphs. For instance, the result of the algorithm on the graph from Figure 1 is visualized in Figure 2. We also provide an enlarged version in Appendix D. The periphery of the figure comprises all the nodes from Figure 1. They are distributed over 13 different colors, where each color represents a separate community. Inside the visualization from Figure 2, we see lines and vertices (squares) that illustrate the hierarchical groupings. For example, the squares with no outgoing connections refer to the lowest hierarchy. Similar to the number of colors, they are again 13. Then, the vertices of the lowest hierarchy are merged into higher-level hierarchies. For instance, the top right four squares are connected into an upper-level hierarchy. Then, it is connected to the upper left four vertices. This process is repeated until all squares are merged into the root vertex, which is located in the middle of Figure 2.

Figure 3 provides a visual representation of the extracted hierarchy obtained from Figure 2. Here, the leaf vertices are composed of only single classes, which might not always be the case. However, for simplicity when visualizing and comparing to the true hierarchy, we extend leaf nodes with more than one class to single class leaf nodes. In the subsequent paragraph, we explain how we produced Figure 3.

The lowest level hierarchy from the nested-SBM consists of 13 communities, whereas the classes in Fashion MNIST are 10. Therefore, there exist clusters with no output nodes in them. These results follow the finding from the original work on co-activation graphs [3, 11]. To obtain our extracted hierarchy graph, we focus only on the communities that contain the class node. The leaf vertices of our ontology are derived from the distribution of output nodes in the lowest hierarchy level of Figure 2. In other words, when only one output node is present in a community from the lowest hierarchy of the nested-SBM, we create a leaf vertex in the extracted hierarchy graph with an attribute the name of the corresponding class (Figure 3). Conversely, if more than one output node is in the bottom-level community, we add a leaf node to Figure 3 with an attribute consisting of all separate classes. After creating the lowest level of the extracted hierarchy graph, we follow the connections from Figure 2 to obtain the rest of the graph up until the root.

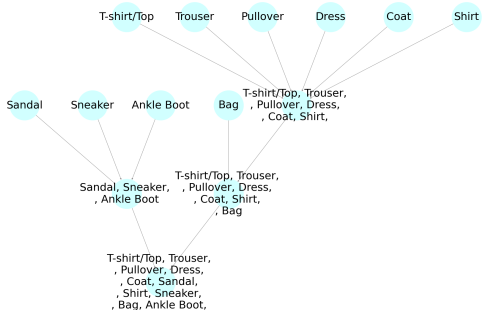


Figure 3: *Observed Hierarchy Graph Derived from Figure 2*

4.3.3 True Hierarchy Graphs.

An essential part of this thesis is comparing hierarchy graphs. This gives us a way to evaluate the class ontologies extracted from our ConvNets. Here, we define our ground truth groupings of the datasets' classes. Figure 4 shows the annotated ground truth hierarchy graph for Fashion MNIST, which we created manually.

The exact formulation of the true hierarchy graph from Figure 4 is up for discussion. We understand this introduces a bias and might not be the optimal structure for the hierarchy levels. However, without the assumption that Figure 4 captures accurately enough the relationships between the classes of Fashion MNIST, we do not have a ground truth with which to compare Figure 3. An alternative is a manual categorical evaluation of observed hierarchy graphs. Even so, we would again need to distinguish between what is right and wrong in terms of class ontology. Hence, it introduces more uncertainty and, due to time constraints, falls outside the scope of this thesis.

We also define an annotated ground truth class hierarchy for Cifar20. When increasing the number of classes, the resulting true hierarchy graph becomes more and more complicated to visualize. That is why we include it in Appendix C, where we create two figures which illustrate its left and right sides when split by the root.

It becomes increasingly complicated to inspect hierarchies visually with over 20 classes. Furthermore, a larger true hierarchy graph encapsulates considerably more annotated bias because of more interconnections and relationships between the classes. While we solve the first issue by automating the graph comparison, it is not trivial to remove the additional bias and potential human mistakes when creating enormous ground truth hierarchies. As a result, we choose a dataset with slightly more classes than Fashion MNIST (20).

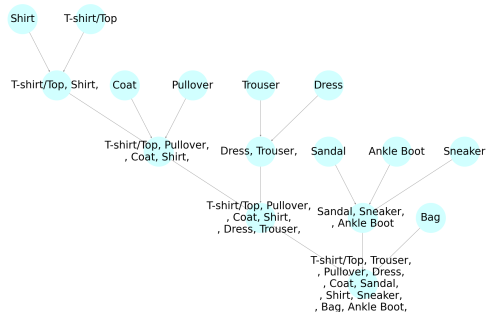


Figure 4: *True Hierarchy Graph*

4.3.4 *Phylogenetic Tree Graph Comparison.*

After creating the two ground truth hierarchy graphs, we are ready to compare them to the observed hierarchy graphs. In the background section, we mentioned that phylogenetic trees are similar to our hierarchy graphs. For both, it holds that a higher-level hierarchy is defined by the lower-level node attributes and node connections. For instance, once we create a vertex for upper body clothes in Figure 4, all higher levels with that node as a child should include the classes Shirt, T-shirt/Top, Coat, and Pullover. This follows the principles of phylogenetic trees.

Since phylogenetic and our hierarchy graphs share the same characteristics, we can choose from the comparison metrics discussed in the related work section. In this thesis, we opted for the most well-known and used one - the original RF metric [31]. Although RF has its downsides, we are confident that it can capture the dissimilarities between our true and observed hierarchies because of its extensive use in previous literature.

In practise, when we apply the RF on Figures 3 and 4, we get a result of 3. The differences come from the observed hierarchy graph, where the class "bag" should be in the higher hierarchy level, and "dress" and "trouser" should be moved one level lower. The measurement is discrete due to binary edge weights - either exist or not. Although this means the RF might be insensitive to small changes in the observed hierarchy graphs, in the results section, we show that the nested-SBM exhibits enough variance to compensate.

4.3.5 *Neural Network Compression.*

A central focus of our work revolves around neural network compression. We want to remove insignificant and shared nodes in our co-activation graphs [3, 11] and we expect to observe changes in the observed hierarchy graph, which we capture with the RF metric. In the subsequent paragraphs, we explain why we decided to go with structured magnitude pruning. We also introduce another compression method that works directly on the topology of co-activation graphs.

Structured Magnitude Pruning. Choosing unstructured pruning would influence the activation values, which would change the edge weights in our co-activation graphs. However, with this approach, we hardly address the problems of insignificant and shared nodes. Conversely, if we opt for structured compression, we can directly change the topology of our graphs. Here, we assume that choosing structured magnitude pruning addresses the stated issues. Because it has proven to accomplish strong results in the trade-off between sparsity and accuracy [15], we believe it should, in theory, delete insignificant and shared co-activation nodes.

Our setup in terms of ConvNet compression is as follows: the saliency criterion is magnitudes; we do a one-shot pruning at 70% convergence and fine-tune until maximum possible accuracy. We also boost our algorithm by applying the group-HS, which is an effective sparsity inducing regularization [20]. It forces redundant nodes towards zero since it is a computationally cheaper variant of ℓ_0 regularization.

After compressing our ConvNets on Fashion MNIST and CIFAR20, the subsequent steps are the same as we previously showed. We save our models' weight matrices, we create the co-activation graphs and the hierarchical communities, and we compare the latter to the true hierarchy graph. At that point, we are ready to conclude how structured magnitude pruning affects our ConvNets grouping of output classes.

4.4 Implementation

Our code is available here[7]. We use PyTorch[8] to define and train our ConvNets. We also received the original code implementation of co-activation graphs from the authors of [3]. The nested-SBM is implemented in the Python library graph-tool [40], and we use the Python library DendroPy [41] for the RF comparison metric. Finally, the implementation of the compression algorithm was facilitated by the authors of [19]. They provided us with a skeleton that can perform structured magnitude pruning on fully connected DNNs, which we extended for ConvNets.

4.4.1 *Hyperparameters.*

We perform extensive hyperparameter tuning for all three models on both datasets by optimizing the learning rate, the weight decay, the number of epochs, and the batch size. Because the number of hidden neurons in our ConvNets considerably influences the topology of the resulting co-activation graphs, we keep them constant. We also only use cross-entropy loss. To find the best hyperparameters, we apply grid search. The possible variations are as follows $batch - size = [200, 500, 700]$, weight decay from 1e−4 to 1e−7 and a learning rate between 1e−4 to 0.01. The values of the latter two decrease by a factor of 10. The exact numbers are shown in Appendix A. We also use data augmentation strategies, such as random horizontal flips, random rotations, random affine, color jitter, after which we normalize our images. This helps with overfitting, although it increases the convergence time on MNIST from a few epochs (about 10) to over 100.

Additionally, we apply cosine annealing [42] on Conv6 to boost the performance of our best model. This method requires an exact number of epochs before training, which means we first apply grid search as we discussed, and then we find the best number of epochs when including cosine annealing for Conv6. Regarding our other two models, we train them until convergence and report the number of epochs.

[7] https://github.com/hpv200/Internship-Master-s-Thesis
[8] https://pytorch.org/

When compressing our ConvNets, we keep the same values for the weight decay, learning rate, and batch size. However, we additionally fine-tune to restore as much accuracy as possible. Apart from choosing the appropriate number of epochs, we also do a grid search on the weight of the group-HS regularization on our loss function. The exact parameters are again shown in Appendix A.

Regarding co-activation graphs, we have only one hyperparameter, which determines the threshold of allowed edge weights. Our early empirical tests show that removing edges with a value below 0.5 produces meaningful graphs. As a result, we have edge weights between 0.5 and 1, which influences the nested-SBM. We address this issue by linearly mapping the range between 0.5 to 1 to a range between 0 and 1.

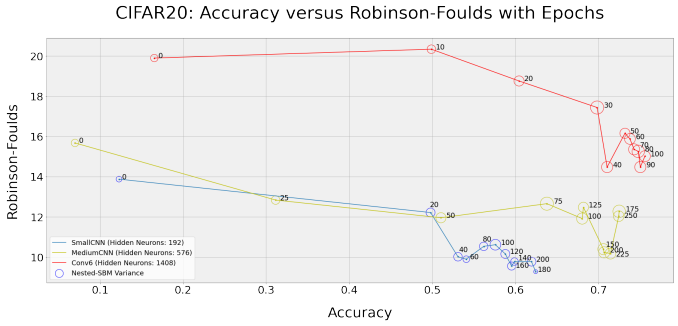


Figure 5: *CIFAR20: Accuracy versus Robinson-Foulds with Epochs*

5 RESULTS

We begin with our findings on CIFAR20, followed by Fashion MNIST. The results on both datasets share the same structure, which is composed of two types of experiments.

(1) We extract co-activation graphs and observed hierarchy graphs at different stages of training of our models. The exact epochs are uniformly distributed in 11 training checkpoints, where we begin after the stochastic gradient descent has updated the weights once, and we end with a converged ConvNet. This formulation allows us to explore how the class hierarchy changes during training.

(2) The second experiment includes DNN pruning. We extract observed hierarchy graphs in the same way as previously described for five sparsity percentages of our models (0.5, 0.675, 0.75, 0.825, 0.9). Then, we explore how the converged uncompressed ConvNets compare to their compressed variants.

5.1 CIFAR20

5.1.1 *First Experiment: Influence of Performance and Initial Model Size on RF.*

Figure 5 shows the results from the first experiment on CIFAR20. Extracting multiple observed hierarchy graphs for the same co-activation graph can lead to different results because of the nested-SBM. We address these fluctuations by measuring the variance of 50 RF values, and we represent it categorically by the circles inside Figure 5. A larger circle indicates a higher variance. Additionally, we keep the same scale throughout all of our graphs. Each point in Figure 5 represents the accuracy on the test set versus the average of 50 RF values at a particular stage of training. We also include all three models separated by the color of the line. Lastly, we add a number associated with every point, which shows the epoch at which we extracted the co-activation graphs and computed the averaged RF.

Our first observation from the first experiment refers to the initial model size. We believe it considerably influences the resulting RF because we notice that the RF values differ at the first epoch (Figure 5). This trend is preserved throughout training as the smaller-sized models seem to produce better RF values. For instance, the averaged RFs for the converged models are again separated by model size - 15.02, 12.04, 9.78 for Conv6, MediumCNN, and SmallCNN, respectively.

Our second observation is focused on model performance. Our ConvNets tend to produce better observed hierarchy graphs when their accuracy increases (Figure 5. For example, Conv6 starts with 19.9 averaged RF at epoch zero and reaches 15.02. This behavior also holds for the other two models. Hence, the RF seems to improve with ConvNet training. However, we fail to see a fixed relation between our two metrics because the models are different and because of statistical variance. For instance, between 61% to 64% test accuracy we obtain RFs of 18.76, 12.66, 9.78 for Conv6, MediumCNN, and SmallCNN, respectively. These numbers suggest that a certain accuracy does not necessarily map to a certain RF measurement.

Finally, our third observation concerns the discrepancies in Figure 5. Despite the trends between our two metrics, some of the points show abnormally decreased RF. For example, at epoch 40 of Conv6, we observe a comparable RF to the converged model. Additionally, epochs 150, 200, and 225 of MediumCNN show better RF than the last epoch, and epoch 180 of SmallCNN has slightly better test accuracy and RF than the converged model. Lastly, we do not see any patterns associated with the variance induced by the nested-SBM.

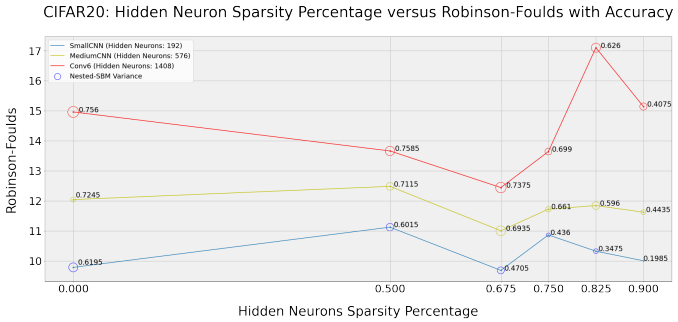


Figure 6: *CIFAR20: Hidden Neuron Sparsity Percentage versus Robinson-Foulds with Accuracy*

5.1.2 *Second Experiment: Influence of Pruning Ratio, Initial Model Size and Performance on RF.*

Figures 6 and 7 depict the results from the second experiment on CIFAR20. The numbers are also summarised in Table 1. Again the

circles categorically represent the variance induced by the nested-SBM. In Figure 6 the x-axis shows the sparsity percentage, where 0.75 means that 75% of the neurons of a model have been pruned. Alternatively, a 0% sparsity refers to our converged unpruned models or the points with the largest epoch numbers from Figure 5. The y-axis is again the RF of the average of 50 comparisons between true and observed hierarchy graphs. Finally, the numbers inside Figure 6 correspond to the test accuracy of our ConvNets. Conversely, in Figure 7 we switch accuracy with sparsity, which gives a different perspective of Figure 6. All points are derived from their corresponding converged models.

	Test Accuracy in %		
Sparsity in %	SmallCNN	MediumCNN	Conv6
0	61.95	72.45	75.6
0.5	60.15	71.15	75.85
0.675	47.05	69.35	73.75
0.75	43.6	61.1	69.9
0.825	34.75	59.6	62.6
0.9	19.85	44.35	40.75

	Averaged Robinson-Foulds		
Sparsity in %	SmallCNN	MediumCNN	Conv6
0	9.78 ± 1.37	12.04 ± 0.77	14.96 ± 1.68
0.5	11.12 ± 1.17	12.48 ± 1.18	13.66 ± 1.44
0.675	9.68 ± 1.08	11 ± 1.5	12.44 ± 1.61
0.75	10.86 ± 0.61	11.72 ± 0.83	13.64 ± 1.05
0.825	10.32 ± 0.74	11.84 ± 1.84	17.1 ± 1.4
0.9	10 ± 0	11.62 ± 0.75	15.14 ± 1.11

Table 1: Results on CIFAR20

Our first observation from the second experiment on CIFAR20 is that there exists a pruning ratio for which the RF metric is lower or comparable to the unpruned converged ConvNet (Figure 6). Although our three models are considerably different in size, removing around 0.675% of the neurons shows an improvement of the averaged RF, except for SmallCNN, which produces relatively similar RF results - from 9.78 ± 1.37 on the unpruned ConvNet to 9.68 ± 1.08 on the pruned one (Table 1). However, because we obtain our finding from only one random weight initialization of our ConvNets, we are reluctant to state that for CIFAR20, the optimal pruning ratio, in terms of RF, is 0.675.

The second observation follows from both Figure 6 and 7. We identify that the initial number of neurons has a considerable effect on the resulting RF scores when pruning our ConvNets. For example, our models do not overlap in terms of y-axis values (Figures 6 and 7).

Our third observation regards the trade-off between model performance and the RF measurement. From Figure 7, we can deduce that the 90% pruned SmallCNN with an accuracy of 19.85% demonstrates a better RF than the unpruned MediumCNN and Conv6 with accuracies of 72,45% and 75.6%, respectively. Moreover, this holds for several other comparisons of this sort.

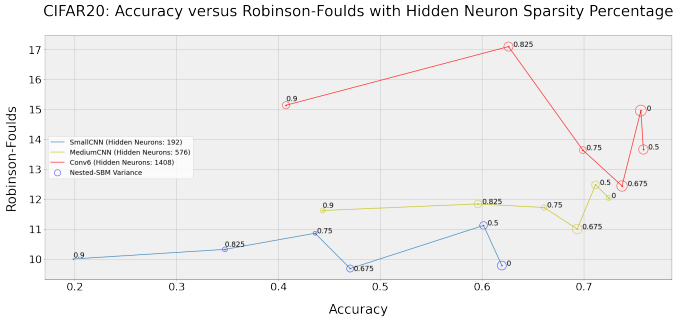


Figure 7: *CIFAR20: Accuracy versus Robinson-Foulds with Hidden Neuron Sparsity Percentage*

5.2 Fashion MNIST

5.2.1 *First Experiment: Influence of Performance and Initial Model Size on RF.*

The first experiment on Fashion MNIST resembles the first one on CIFAR20. The results are summarised in Figure 8, which has the same characteristics as Figure 5.

We again notice that the initial model size influences the quality of the extracted class hierarchies through the RF metric. Generally, our ConvNets with fewer neurons tend to produce better RF scores during training and at convergence. Although our findings on Fashion MNIST are not as consistent as on CIFAR20, MediumCNN yields better RF scores than Conv6. Additionally, SmallCNN again outperforms the other two ConvNets in terms of RF.

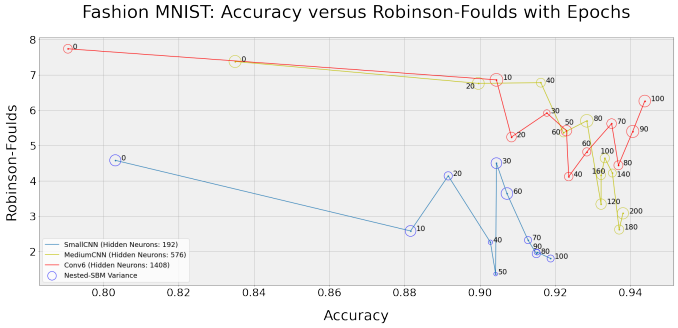


Figure 8: *Fashion MNIST: Accuracy versus Robinson-Foulds with Epochs*

In Figure 8, we see several discrepancies between model epochs and RF scores. They suggest that the relation between ConvNets accuracy and RF is not as prevalent as the number of neurons and RF. Indeed, we notice a decrease in the RF metric between the first and last epoch. However, we still cannot argue about a clear positive correlation between the two. Furthermore, although the final epoch produces the best ConvNet accuracies, neither one of the converged models has the lowest RF score (Figure 8).

5.2.2 *Second Experiment: Influence of Pruning Ratio, Initial Model Size and Performance on RF.*

For our second experiment on Fashion MNIST, we illustrate our

results in Table 2 and in the Figures 9 and 10, which are defined similarly as Table 1 and Figures 6 and 7.

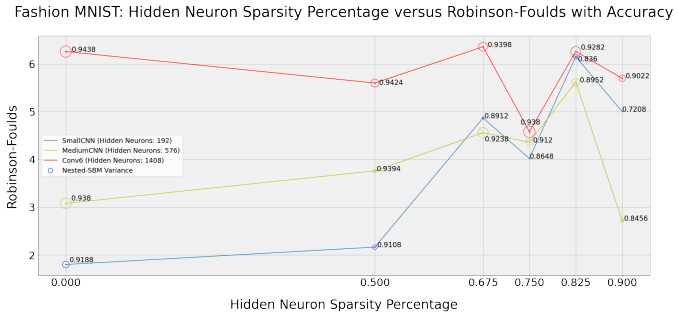


Figure 9: *Fashion MNIST: Hidden Neuron Sparsity Percentage versus Robinson-Foulds with Accuracy*

Earlier, we derived that pruning leads to a lower RF score on CIFAR20. However, we notice that in Figure 9 the lowest RF score of SmallCNN is a result of the unpruned ConvNet. Even though this does not follow the results from the experiments on CIFAR20, we are still inclined to believe that compression can improve the quality of the observed hierarchy graphs. Firstly, the other two ConvNets seem to produce lower RF scores for some sparsity percentages. Secondly, the RF metric of SmallCNN with a pruning ratio of 0.5 is relatively close to its uncompressed variant (Table 2). Furthermore, the unpruned SmallCNN indicates an observed hierarchy graph almost equivalent to the ground truth (1.8±1.07). As a result, it is even harder to improve the RF score.

We now address the initial model size. We notice that for a few of the pruning ratios of SmallCNN, it performs worse than MediumCNN. However, except for these three inconsistencies (Figure 6), the results follow the ones from the experiments on CIFAR20.

	Test Accuracy in %		
Sparsity in %	SmallCNN	MediumCNN	Conv6
0	91.88	93.8	94.38
0.5	91.08	93.94	94.24
0.675	89.12	92.38	93.98
0.75	86.48	91.2	93.8
0.825	83.6	89.52	92.82
0.9	72.08	84.56	90.22
	Averaged Robinson-Foulds		
Sparsity in %	SmallCNN	MediumCNN	Conv6
0	1.8 ± 1.07	3.08 ± 1.72	6.26 ± 1.78
0.5	2.16 ± 0.72	3.76 ± 0.56	5.6 ± 1.26
0.675	4.87 ± 0.36	4.56 ± 1.62	6.36 ± 1.4
0.75	4.02 ± 0.14	4.36 ± 1.06	4.58 ± 1.8
0.825	6.15 ± 0.69	5.62 ± 1.19	6.26 ± 1.56
0.9	5 ± 0.06	2.72 ± 0.54	5.7 ± 1.02

Table 2: Results on Fashion MNIST

Without disregarding the discrepancies caused by SmallCNN, we are inclined to believe our findings from the experiments on CIFAR20 still hold. Moreover, we saw that on CIFAR20, SmallCNN produces similar RF scores when unpruned and pruned with 67.5% (Figure 6). Our explanation for this behavior is based on the substantially low number of initial neurons. In real-world scenarios, compressing models with less than 200 neurons is seldom necessary. However, because we saw the considerable impact of the number of neurons on the class hierarchies, we decided to test extreme cases, such as SmallCNN.

Lastly, we address the trade-off between model accuracy and RF. Figure 10 and Table 2 indicate that for every model, there exist at least two points for which both the accuracy and RF score increases. For instance, although the difference between pruning 90% and 82.5% leads to the largest accuracy decrease, we observe better RF measurements with 90% sparsity. Hence, we conclude that the third observation holds for Fashion MNIST, as well.

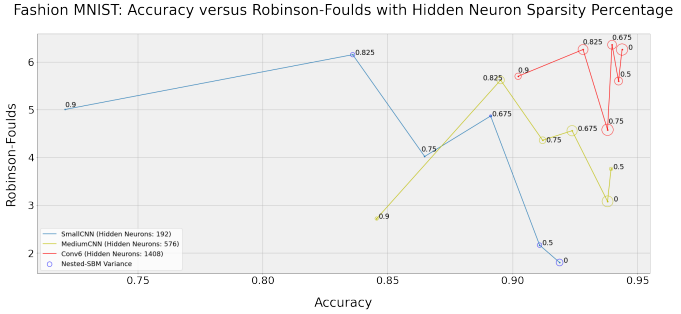


Figure 10: *Fashion MNIST: Accuracy versus Robinson-Foulds with Hidden Neuron Sparsity Percentage*

6 DISCUSSION

6.0.1 How can we compare the true class hierarchy to the class hierarchy extracted from a trained DNN by co-activation analysis? Firstly, we need to derive the structure and most essential characteristics of our hierarchy graphs. They are non-binary rooted trees, where each node has a set of classes assigned to it. In that case, we can focus on specific algorithms with more favorable time complexity than for general graphs.

Subsequently, we identify that the vertex attributes of our graphs exhibit a hierarchical structure. Finally, we notice that at every tree level, the node attributes are strictly defined by the properties of their child vertices. The last characteristic indicates that our trees have previously been researched as phylogenetic trees [8], and several comparison metrics exist. Due to time constraints, we choose the Robinson-Foulds(RF) metric [31] because it has a convenient implementation wrapped around Python libraries. Although previous research has identified some drawbacks from the RF metric, we assume that the introduced bias is negligible as our trees are much smaller than real-world networks.

6.0.2 How does the performance of DNNs influence the quality of the extracted class hierarchy obtained from the hierarchical community detection on co-activation graphs? We show that a model improves upon the RF metric when comparing the first and last

epochs. Additionally, with a few exceptions, the findings on CIFAR20 suggest that the improvement of accuracy during training of our ConvNets and their corresponding RF values are positively correlated. However, the few discrepancies on CIFAR20 indicate that a model might not have an optimal performance but show similar or even improved RF values compared to the converged ConvNet.

The findings are more conclusive on Fashion MNIST. We demonstrate that neither one of the converged models produces the best RF scores. As a result, we believe that model performance does not always influence the observed hierarchy graph quality. These results may also be subject to the bias introduced by the true hierarchy graphs.

6.0.3 How does neural network compression affect the quality of the extracted class hierarchy when comparing it to the true class hierarchy? Which factors (model performance, percentage of network sparsity, initial model size) are most influential? We showed that MediumCNN and Conv6 produce considerably better RF values on specific pruning ratios. However, SmallCNN exhibits comparable and better results when unpruned on CIFAR20 and Fashion MNIST, respectively. Because the latter model has only 192 neurons, further reducing its size is impractical and might be the reason for the caused discrepancies. Therefore, we argue that neural network compression shows a strong influence on the resulting extracted hierarchy graphs. Furthermore, because larger models are overparameterized, pruning a percentage of them seems to produce better RF scores. However, as we decrease the initial number of neurons, we believe that reducing the size of a ConvNet is less beneficial to the quality of its class hierarchy structure.

Our results show that the initial model size affects our ConvNets the most both when we do not apply pruning and when we compress our models. For instance, in the first experiment, we argue that decreasing the number of neurons enhances the RF values. Even on Fashion MNIST, which shows less consistent results than CIFAR20, we observe that usually smaller ConvNets produce better RF scores. Apart from the initial model size, we also demonstrate that the sparsity percentage considerably influences the quality of the class hierarchies. However, we fail to distinguish any relation between model performance and the RF metric when applying compression.

6.0.4 Which patterns can we observe from the interaction between accuracy, compression ratio, and hierarchical graph quality? Tracking the behavior of three-dimensional data can be a complicated task. However, if we consider only sparsity and accuracy, we observe that the model performance decreases with a few exceptions, where pruning 50% yield slightly better accuracy. Then, if we isolate the compression ratio and the RF, we observe that on CIFAR20 with the current set of experiments, pruning about 67.5% produces the best RF scores. However, we cannot deduce an optimal sparsity percentage with regards to the RF for Fashion MNIST. Even though larger models might form an optimal pruning ratio of around 75%, we cannot test this hypothesis due to the time constraints of this thesis. Lastly, if we consider only accuracy and RF, we fail to distinguish any particular patterns because compressed ConvNet performance does not affect the quality of co-activation graphs.

7 CONCLUSIONS

In this thesis, we have extracted class hierarchies from convolutional neural networks. We have also shown how to compare such structures. Finally, we have explored the relationship between deep neural compression and ontology extraction.

By combining several concepts from previous literature, we have demonstrated how to extract class ontology from a trained DNN. Then, again supported by previous work, we have illustrated how to measure the difference between our extracted and true hierarchies. Then, on our unpruned ConvNets, our results indicate that the initial model size is more prevalent than the accuracy for the extraction quality of the class hierarchies. Additionally, we show that training a ConvNet until convergence does not necessarily entail a better class hierarchical structure. When applying compression, we observe no correlation between model performance and the quality of the class ontology. However, we demonstrate that the initial model size again substantially affects the class hierarchy. Lastly, we indicate that except for considerably small models, compressing ConvNets leads to better class hierarchies.

The limitations of this work revolve mainly around the ground truth class hierarchy. For most datasets, there is no previous ontology, which would force future research into a manual annotation. The consequences include the already discussed bias. However, from a different perspective, we believe a solution might be to input the RF score back into the DNN, forcing the model to learn the desired class hierarchy.

Several other directions for future work emerge. Co-activation graphs with Spearman correlation is only one way of graph extraction. A possible direction would explore how changing the graph extraction algorithm influences the class hierarchies through our methods. For instance, the authors of co-activation graphs [11] propose the use of Pearson correlation coefficient [43]. Another option is to change the graph comparison metric. We believe it is feasible to even choose a more complicated metric with increased time complexity, as our hierarchy graphs are relatively small.

Lastly, we discuss changing the compression algorithm. Magnitude is one of the many possible saliency criteria. Here, we propose a way to connect homogeneously graph extraction and pruning.

The authors of co-activation graphs [3] suggest that low centrality nodes (insignificant) play a minor role in the graphs and in the inference results of its corresponding DNN. Moreover, they also reason that vertices with strong connections [9] to multiple output classes (shared) seem to produce most of the incorrect predictions of DNNs. Because of the one-to-one mapping between co-activation nodes and DNN neurons, it is possible to devise a saliency criterion on vertex insignificance and sharing and remove neurons based on a certain threshold. We expect such a compression algorithm to produce better class hierarchies in terms of RF score as it is defined by the characteristics of the graph extraction method.

ACKNOWLEDGEMENTS

I sincerely thank Maarten Stol and Stijn Verdenius from BrainCreators for their continuous support and commitment at every stage of this thesis. I also thank Erman Acar from the VU and Vitor Horta

[9] strong connections = large edge weights

from the Insight Center for their supervision and contributions to this deliverable.

REFERENCES

[1] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.

[2] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[3] Vitor AC Horta, Ilaria Tiddi, Suzanne Little, and Alessandra Mileo. Extracting knowledge from deep neural networks through graph analysis. *Future Generation Computer Systems*, 120:109–118, 2021.

[4] Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? *arXiv preprint arXiv:1312.6184*, 2013.

[5] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[6] Andrei Apostol, Maarten Stol, and Patrick Forré. Flipout: Uncovering redundant weights via sign flipping. *arXiv preprint arXiv:2009.02594*, 2020.

[7] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[8] Katherine St. John. The shape of phylogenetic treespace. *Systematic biology*, 66(1):e83–e94, 2017.

[9] Feng Shi, Qilong Feng, Jianer Chen, Lusheng Wang, and Jianxin Wang. Distances between phylogenetic trees: A survey. *Tsinghua Science and Technology*, 18(5):490–499, 2013.

[10] Dario Garcia-Gasulla, Armand Vilalta, Ferran Parés, Jonatan Moreno, Eduard Ayguadé, Jesús Labarta, Ulises Cortés, and Toyotaro Suzumura. Building graph representations of deep vector embeddings. *arXiv preprint arXiv:1707.07465*, 2017.

[11] Vitor AC Horta and Alessandra Mileo. Towards explaining deep neural networks through graph analysis. In *International Conference on Database and Expert Systems Applications*, pages 155–165. Springer, 2019.

[12] Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. Optimal brain damage. In *NIPs*, volume 2, pages 598–605. Citeseer, 1989.

[13] Xiaoqin Zeng and Daniel S Yeung. Hidden neuron pruning of multilayer perceptrons using a quantified sensitivity measure. *Neurocomputing*, 69(7-9):825–837, 2006.

[14] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[15] Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*, 2019.

[16] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

[17] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(9), 2004.

[18] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[19] Stijn Verdenius, Maarten Stol, and Patrick Forré. Pruning via iterative ranking of sensitivity statistics. *arXiv preprint arXiv:2006.00896*, 2020.

[20] Huanrui Yang, Wei Wen, and Hai Li. Deephoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures. *arXiv preprint arXiv:1908.09979*, 2019.

[21] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

[22] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011.

[23] Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

[24] Thorben Funke and Till Becker. Stochastic block models: A comparison of variants and inference methods. *PloS one*, 14(4):e0215296, 2019.

[25] Xiaoran Yan. Bayesian model selection of stochastic block models. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 323–328. IEEE, 2016.

[26] Tiago P Peixoto. Bayesian stochastic blockmodeling. *Advances in network clustering and blockmodeling*, pages 289–332, 2019.

[27] Tiago P Peixoto. Hierarchical block structures and high-resolution model selection in large networks. *Physical Review X*, 4(1):011047, 2014.

[28] Gary L Miller. Graph isomorphism, general remarks. *Journal of Computer and System Sciences*, 18(2):128–142, 1979.

[29] László Babai. Graph isomorphism in quasipolynomial time, 2016.

[30] Samuel R Buss. Alogtime algorithms for tree isomorphism, comparison, and canonization. In *Kurt Gödel Colloquium on Computational Logic and Proof Theory*, pages 18–33. Springer, 1997.

[31] David F Robinson and Leslie R Foulds. Comparison of phylogenetic trees. *Mathematical biosciences*, 53(1-2):131–147, 1981.

[32] Richard C Wilson and Edwin R Hancock. Levenshtein distance for graph spectral features. In *ICPR (2)*, pages 489–492, 2004.

[33] Nicholas D Pattengale, Eric J Gottlieb, and Bernard ME Moret. Efficiently computing the robinson-foulds metric. *Journal of Computational Biology*, 14(6):724–735, 2007.

[34] Sebastian Böcker, Stefan Canzar, and Gunnar W Klau. The generalized robinson-foulds metric. In *International Workshop on Algorithms in Bioinformatics*, pages 156–169. Springer, 2013.

[35] Marina Marcet-Houben and Toni Gabaldón. Treeko: a duplication-aware algorithm for the comparison of phylogenetic trees. *Nucleic acids research*, 39(10):e66–e66, 2011.

[36] Damian Bogdanowicz and Krzysztof Giaro. Comparing phylogenetic trees by matching nodes using the transfer distance between partitions. *Journal of Computational Biology*, 24(5):422–435, 2017.

[37] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[38] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014.

[41] Jeet Sukumaran and Mark T Holder. Dendropy: a python library for phylogenetic computing. *Bioinformatics*, 26(12):1569–1571, 2010.

[42] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[43] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

A HYPERPARAMETER SETTINGS

	Fashion MNIST			
Sparsity in %	Epoch	Decay	LR Rate	Batch
SmallCNN	100	1e−3	1e−5	500
MediumCNN	100	1e−4	1e−7	200
Conv6	120	0.05	1e−6	500

Table 3: *Hyperparameters on Fashion MNIST*

	CIFAR20			
Sparsity in %	Epoch	Decay	LR Rate	Batch
SmallCNN	250	1e-3	1e−4	300
MediumCNN	250	1e-3	1e-4	300
Conv6	100	0.01	1e-4	500

Table 4: *Hyperparameters on CIFAR20*

	Fashion MNIST		CIFAR20	
ConvNet	Epoch	Group-HS	Epoch	Group-HS
0.5	250	1e-5	300	1e-4
0.675	250	1e-5	300	0.01
0.75	250	1e-5	300	1e-4
0.825	250	1e-4	300	1e-4
0.9	250	1e-3	300	1e-4

Table 5: *Compression Hyperparameters for SmallCNN*

	Fashion MNIST		CIFAR20	
ConvNet	Epoch	Group-HS	Epoch	Group-HS
0.5	200	1e-4	300	1e-4
0.675	200	1e-3	300	1e-4
0.75	200	1e-3	300	1e-3
0.825	200	1e-3	300	1e-3
0.9	200	1e-4	300	1e-3

Table 6: *Compression Hyperparameters for MediumCNN*

	Fashion MNIST		CIFAR20	
ConvNet	Epoch	Group-HS	Epoch	Group-HS
0.5	150	1e-4	150	1e-3
0.675	150	1e-4	150	1e-4
0.75	150	1e-4	150	1e-4
0.825	150	1e-4	150	1e-5
0.9	150	1e-3	150	1e-5

Table 7: *Compression Hyperparameters for Conv6*

B COMPUTATIONAL RESOURCES

To carry out our experiments, we use the educational cluster computer Lisa[10] and google colab[11]. We train and prune our ConvNets, and we create the co-activation graphs on Lisa. Conversely, we run the nested-SBM, and we extract the RF measurements on google colab.

[10]https://userinfo.surfsara.nl/systems/lisa
[11]https://colab.research.google.com/notebooks/intro.ipynb

C CIFAR100 SUBSET TRUE HIERARCHY GRAPH

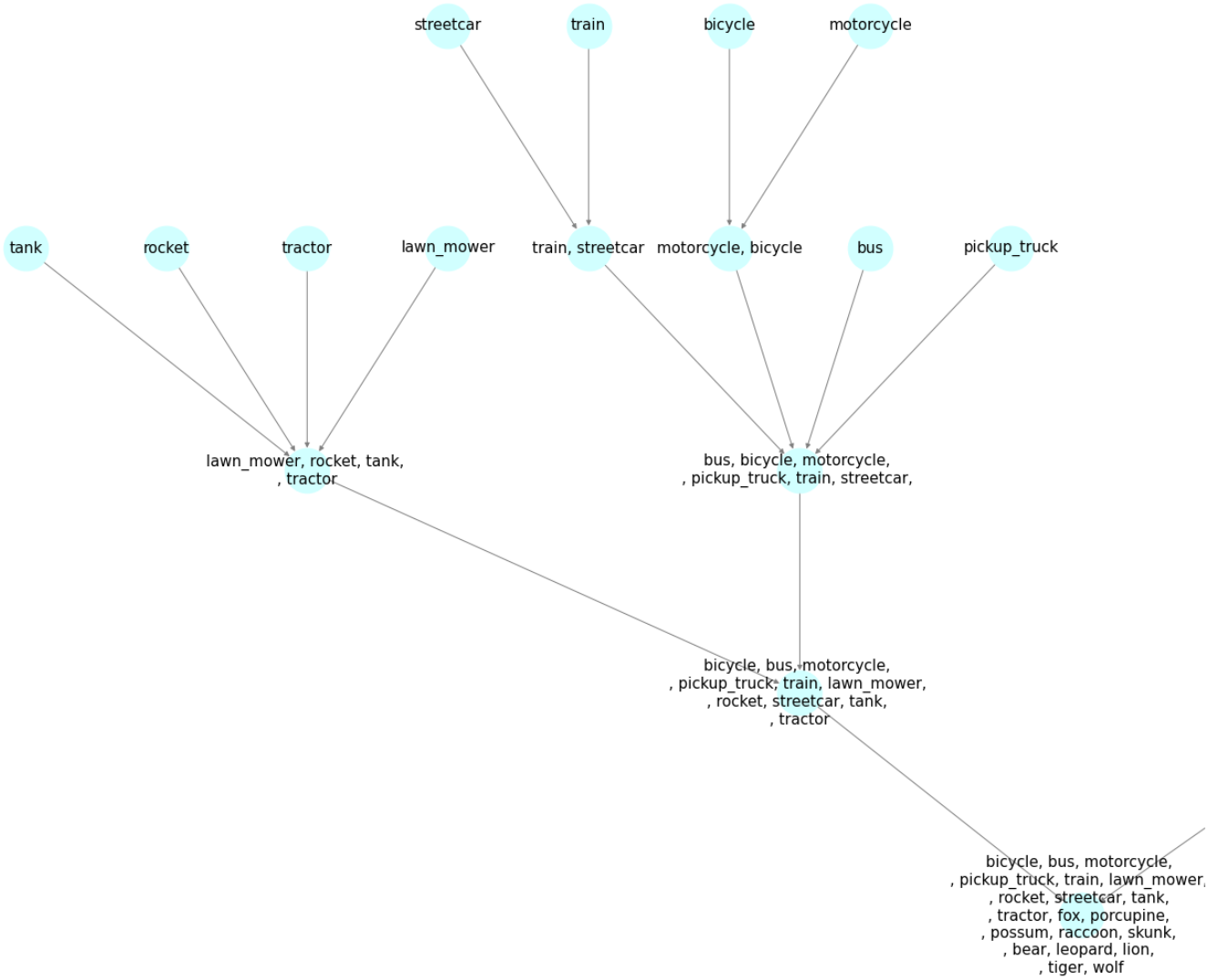


Figure 11: *Left Part of the Subset of CIFAR100 True Hierarchy Graph (20 classes)*

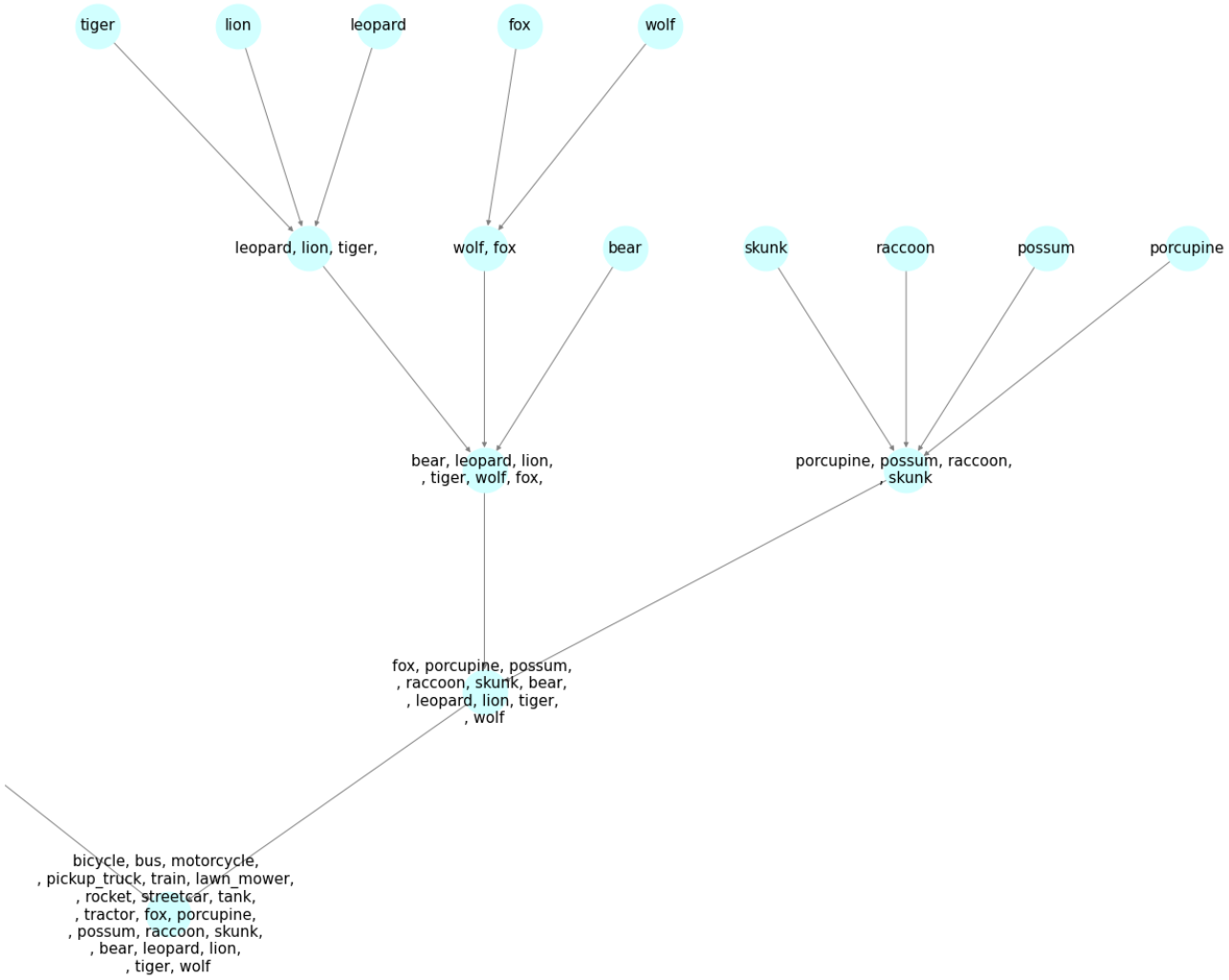


Figure 12: *Right Part of the Subset of CIFAR100 True Hierarchy Graph (20 classes)*

D CIRCLE VISUALIZATION OF THE OUTPUT OF THE NESTED-SBM

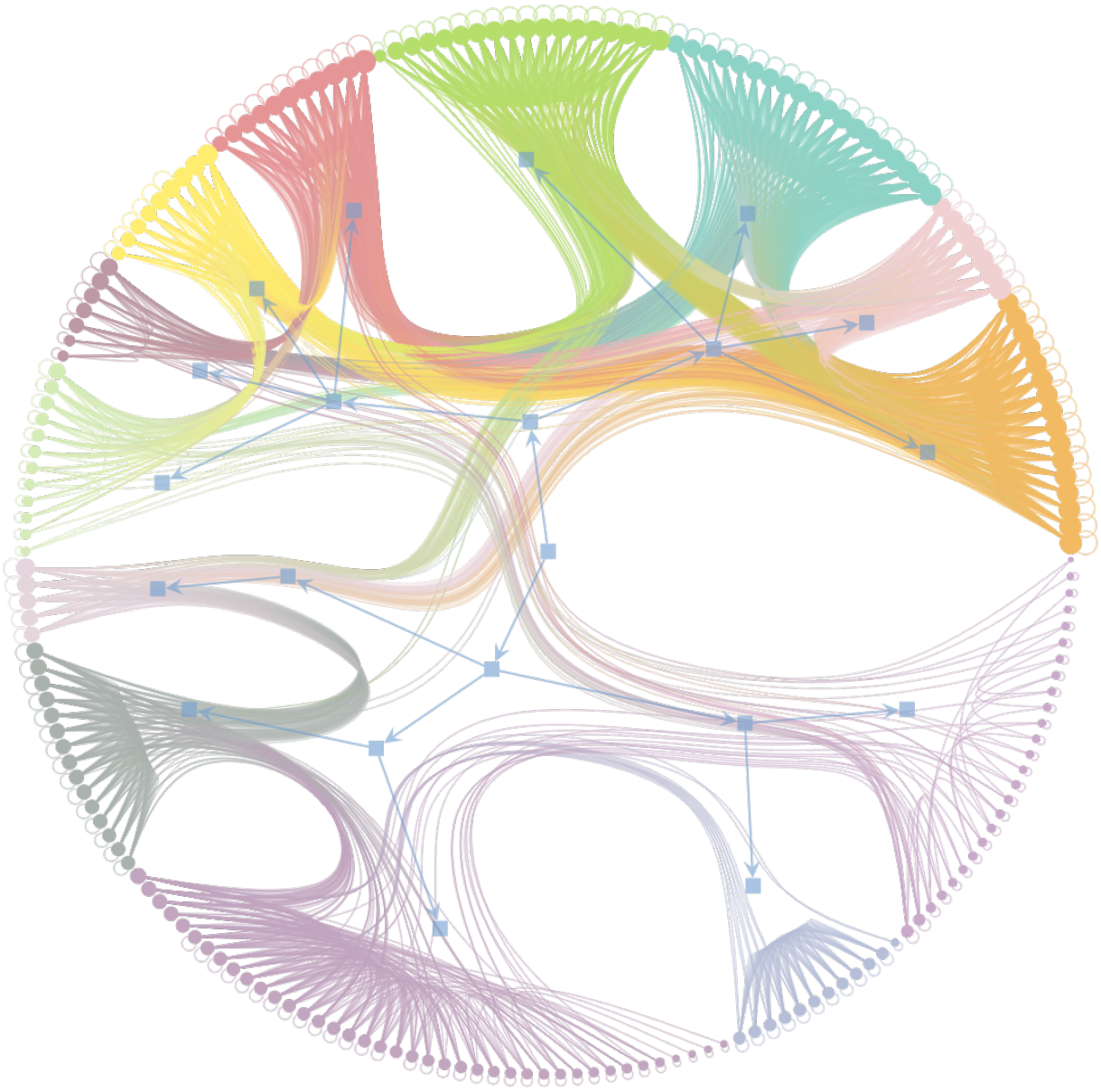


Figure 13: *Enlarged Version of Figure 2*