Universiteit Leiden
The Netherlands

Bachelor Datascience and Artificial Intelligence

Analyzing DropConnect Retention Rates and the Block Structure of Parameter Correlation Matrices in Feed-Forward Networks

Justin Meurs

First supervisor:
Evert van Nieuwenburg
Second supervisor:
Björn van Zwol

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

22/01/2026

Abstract

DropConnect is a stochastic regularization method that improves generalization by randomly removing weights during training, effectively sampling an ensemble of sparse sub-networks. While its empirical effectiveness is proven, the mechanisms underlying its robustness are not defined. Recent theoretical work by Barboza proposed a statistical-mechanical interpretation of DropConnect, suggesting that robustness to weight removal is linked to parameter correlations and system-size compression, which can manifest as block-diagonal structures in correlation matrices.

In this thesis the relationship between DropConnect retention rates and the correlation structure of network parameters is investigated empirically. Feed-forward networks are trained on the MNIST and CIFAR-10 dataset, while systematically varying network architectures and retention rates. The robustness of DropConnect is analyzed using curves estimating accuracy as a function of retention rate. Ensembles of these architectures are created and analyzed using principal component analyses and correlation matrices to visualize the effective dimensionality and correlation structure.

The results confirm earlier findings for MNIST. Accuracy exhibits a pronounced plateau across a wide range of retention rates, the number of parameters greatly exceeds the effective dimensionality and stable block-diagonal correlation structures are observed. For CIFAR-10 the plateau behaviour is less pronounced and block-structure weakens as the ensemble size grows, indicating a lack of stable system-size compression for the considered architectures. These findings suggest that block-diagonal correlation structures are not required for DropConnect to be effective.

Contents

1 Introduction

Deep neural networks have emerged as powerful function approximators, achieving state-of-the-art performance across a wide range of domains, from computer vision to natural language processing [XM19]. Despite their practical successes, many aspects of their internal dynamics remain poorly understood. Particularly, regularization methods like Dropout and its generalization DropConnect are known to improve generalization performance and prevent overfitting [LW13] [NS14]. However, the mechanisms that make them effective are still the subject of research [Bar25] [GG16] [SWL13] DropConnect works by stochastically removing individual weights during training, effectively sampling from a large ensemble of sparse sub-networks. Recent theoretical research by Barboza [Bar25], suggests that this process can be interpreted through statistical mechanics. Using this method each sampled sub-network represents a configuration of a larger ensemble which can be studied using techniques from statistical mechanics. Within this framework it is suggested that the behaviour of DropConnect does not only depend on the rate at which the parameters are dropped, but also on how strongly the remaining parameters inside this network are correlated.
The key result from this approach is the hypothesis that networks showing a block-diagonal parameter correlation structure may benefit more from DropConnect. The weight matrices corresponding to these networks show a natural partitioning into communities of strongly correlated parameters. Such structure indicates a form of system size compression, where the effective number of independent degrees of freedom is smaller than the raw parameter count. This property is predicted to create robustness to random weight removal and to generate a certain 'plateau' in test accuracy scores across a range of DropConnect retention rates.

Previous empirical research, however, has been very limited. The experiments conducted by Barboza in his master's thesis were performed on the MNIST dataset. This research revealed that the effective number of independent degrees of freedom was capped at approximately 100 regardless of the architecture used. The thesis hypotheses that these results suggest that MNIST is too simple to meaningfully test the hypothesis. To properly test the hypothesis one should turn to more complex datasets and architectures where richer correlation structures can emerge.
This thesis aims to address this gap. First by validating the hypothesis that MNIST is too simple by repeating the PCA analysis with a larger number of networks. Then the research is continued by studying networks trained on more complex datasets such as CIFAR-10 and by systematically varying network architectures while applying DropConnect to analyze the effect of the block structure on the robustness of DropConnect. Doing so this research contributes to a better understanding of the interaction between model architecture, parameter correlation and regularization effectiveness. This gives the research question: What is the relationship between DropConnect efficacy and the block-diagonal structure of the parameter correlation matrices across different neural network architectures and datasets?

2 Definitions

This chapter includes the mathematical and conceptual foundations used in the thesis. The definitions presented here describe the notation, terminology and objects needed to understand the related work, research and results.

2.1 Neural Networks

Neural networks consist of artificial neurons or neurons, that are connected by edges to other neurons. The edges model the synapses in the brain and send and receive signals between neurons if the signal is strong enough. Artificial neural networks very commonly are made up of multiple layers with varying methods of connection. There exists a simple feed-forward neural network(FFN), or perceptron, where data flows from input to output without any cycles. There is a multilayer perceptron(MLP), which the input is passed through at least one hidden layer to the output, using a non-linear activation function like ReLU for complex patterns. A multilayer perceptron is also a feed-forward neural network. There also exist convolutional neural networks and recurrent neural networks, but these are not relevant for this research.
The multilayer perceptron is said to have L hidden layers, L is also said to be the length of the networks. Each layer l has width N_l where $max_{i=1}^{L}(N_i)$ is said to be the width of the network. Given a certain input vector $x \in \mathbb{R}^d$, where d is the input size, the network computes a prediction by applying a sequence of transformations and nonlinear functions described by the following function for the forward pass [IGC16]:

$$h^{(l)} = g^{(l)}(W^{(l)T} l^{(l-1)} + b^{(l)})$$

where:

$h^{(0)}$	equals input vector x
$W^{(l)} \in \mathbb{R}^{N_l \times N_{l-1}}$	is the weight matrix for layer l
$b^{(l)} \in \mathbb{R}^{N_l}$	is the bias vector for layer l
$g^{(l)}$	is a nonlinear activation function, like ReLU or sigmoid

Table 1: Table explaining the parameters used in the formula for the forward pass.

With these parameters, the configuration of a network can be described as $\theta = \{W^{(1)}, b^{(1)}, ..., W^{(L)}, b^{(L)}\}$. Using this set of weights and biases, the current state of a trained feed-forward neural network can be saved and loaded.

2.2 Dropout and DropConnect

This subsection will introduce the regularization methods Dropout and DropConnect. Regularization methods are used to prevent overfitting and improve generalization of neural networks. Both implementations behave similarly and introduce dynamic sparsity within the models.

2.2.1 Dropout

Dropout a method firstly introduced in [NS14]. In this method each neuron in a layer is independently set to zero, with probability $1 - p$. In 1 a schematic example is given for the deactivation of a neuron in a fully connected layer. Here p denotes the retention rate, being the probability that a neuron is kept. Formally, dropout uses a mask vector [NS14]:

$$R^{(l)} \sim Bernoulli(p)^{N_l}$$

This vector is then multiplied as follows:

$$h^{(l)} = R^{(l)} \bigodot g^{(l)}(W^{(l)T} l^{(l-1)} + b^{(l)})$$

Where $\bigodot$ denotes the Hadamard product of the two matrices. This means that for any layer l, $R^{(l)}$ is the vector of independent Bernoulli random variables with each have probability p of being 1 [NS14].

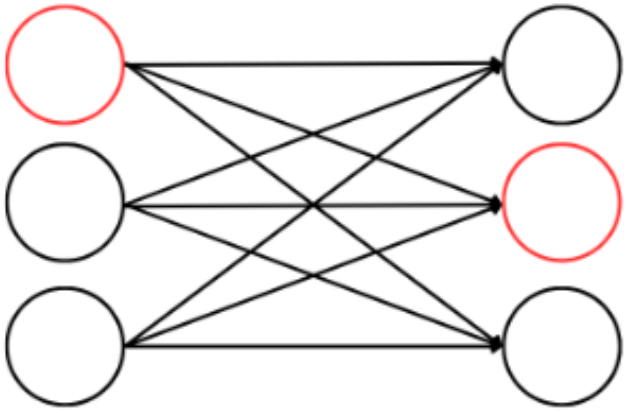


Figure 1: Schematic example for the operation of dropout. The red neurons are deactivated in this training episode.

2.2.2 DropConnect

DropConnect generalizes dropout by dropping individual weights instead of the activations. The dropping rate here is also defined as $1 - p$. A schematic example for the operation of DropConnect can be found in 2. For each layer the mask is calculated as follows [LW13]:

$$M^{(l)} \sim Bernoulli(p)^{N_l - 1 \times N_l}$$

This results in the following formula for the forward pass:

$$h^{(l)} = g^{(l)}((M^{(l)} \bigodot W^{(l)})^T l^{(l-1)} + b^{(l)})$$

Here $\bigodot$ once again means the Hadamard product of the two matrices. This means that for any layer l, $M^{(l)}$ is the vector of independent Bernoulli random variable. Note that each training a different $M^{(l)}$ is generated, meaning the training essentially averages over a number of sub-networks.

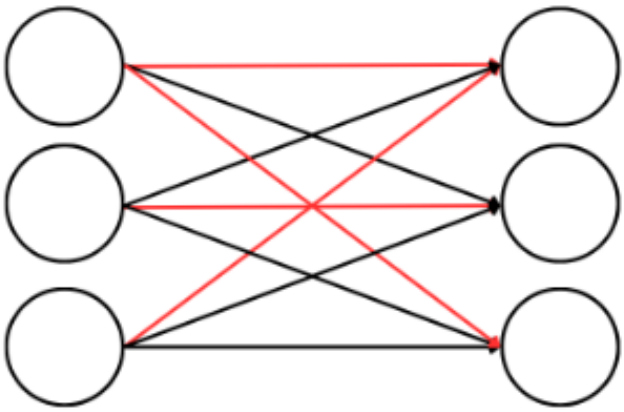


Figure 2: Schematic example for the operation of DropConnect. The red edges are deactivated connections in this training.

2.3 Parameter ensembles

Since the training of DropConnect is stochastic, repeated training of a network with identical architecture, but different seeds produces a family, or ensemble, of models denoted as:

$$X = \begin{bmatrix} | & | & & | \\ \theta^{(1)} & \theta^{(2)} & \dots & \theta^{(M)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{N \times M}$$

Where $\theta^{(m)} \in \mathbb{R}^N$ is the flattened parameter vector of the m-th neural network, M is the number of trained networks and N is the number of parameters in the networks.

2.4 Sample Covariance Matrix

Given matrix X as defined in section 2.3, the sample covariance matrix can be calculated as follows:

$$C = \frac{1}{M} X X^T \in \mathbb{R}^{N \times N}$$

This formula gives us the following properties: C is symmetric and positive semi-definite. From the formula can also be derived that $rank(C)$ is at most $min(N, M)$. That gives us that, for $M < N$, the covariance matrix has at least $N - M$ zero eigenvalues. The covariance reflects co-adaptation, that is parameters that constantly move together across independent trainings.

2.5 Correlation Matrix

The correlation matrix normalizes covariance by the variance of each parameter:

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} C_{jj}}}$$

This correlation matrix reveals the underlying structure of the parameter interactions. Block diagonal patterns correspond to groups of parameters that behave collectively. A block-diagonal structure would look similar to the following matrix:

$$R = \begin{bmatrix} B_1 & 0 & \dots & 0 \\ 0 & B_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B_k \end{bmatrix}$$

Where B_i corresponds to a group of strongly correlated parameters. Block structure implies low dependence between groups of parameters.

2.6 Eigenvalue Spectrum and PCA

The non-zero eigenvalues of C can be computed by computing the singular values of X. The singular value decomposition is defined as:

$$X = U\Sigma V^T$$

Where $U \in \mathbb{R}^{(N\times N)}$ is an orthogonal unitary matrix, $\Sigma \in \mathbb{R}^{(M\times M)}$ is diagonal with singular values σ_i and V^T is an orthogonal matrix. This gives us:

$$C \propto XX^T = (U\Sigma V^T)(U\Sigma^T V^T) = U\Sigma\Sigma^T U^T$$

Meaning the eigenvalues λ_i of C can be calculated using the squares of the singular values of X: $\lambda_i = \sigma_i^2$, for$i = 1, 2, \dots, rank(X)$

3 Related Work

3.1 Regularization in machine learning and deep learning

When training a large model, many factors have to be taken into account to determine the type and parameters of the model. A model can be selected based on the performance on the training data or its generalization capabilities. Another factor to take into account is overfitting and underfitting. Underfitting happens when the model performs poorly on the training data and test data, while overfitting happens when the model performs well on the training data but performs poorly on the test data.

Regularization methods are used and designed to reduce the test error, possibly allowing for higher training error [IGC16]. Classical regularization methods like L1 and L2 penalties work by constraining the magnitude of model parameters, reducing variance. L1, or LASSO, works by adding the absolute value of magnitude of the coefficient to the loss function. This can help shrink some coefficients to zero, which leads to only selecting the important features. L2 regularization, or Ridge regression, works by adding the square magnitude of the coefficient as a penalty term to the loss function.

A comprehensive overview of other regularization methods is given by [TZ22], where the variations of regularization techniques are described from machine learning to deep learning up to open issues

and research trends. This paper provides a thorough description of these techniques, gives their characteristics and discusses how to use certain regularization methods for specific tasks.
The paper starts by introducing norm-based regularization methods for machine learning tasks. These include the previously introduced L1 and L2 regularization and further introduces elastic net. Elastic net can be described as a combination of L1 and L2 where both the absolute norm of the weights and the squared measure of the weights are added with the help of a hyperparameter that controls the ratio between the two. These and other norm-based methods work by directly influencing the structure and variance of the parameters.

For deep-learning tasks several regularization methods are introduced. These can be split in three applications: the noise of the data, the limited size of the data and the complexity of the classifiers. A common practice is to apply data augmentation. The motivation behind creating more data using data augmentation is to remove class imbalance in data, to increase the dataset to a sufficient size and to obtain more information from the new data. Data augmentation works by either works by adjusting parameters like rotation, size, orientation and brightness. Another way is to apply data augmentation techniques to intermediate representations rather than the input data. Two examples of this technique are to stochastically blend two viable tensors in to one new tensor. Another option is to add the difference of two tensors to another tensor, using kNN to form new data points. There exist more complex data augmentation techniques, but those are not very relevant in this context.
Another way to apply regularization in a neural network is by applying normalization. Batch normalization incorporates data standardization into the network by normalizing the activations in batches. Other normalization methods include layer normalization, where inputs are normalized when crossing the features for each individual sample, rather than batches. Instance normalization normalizes each channel in a sample and group normalization normalizes over the group of channels over groups of channels for each sample.
The last group of regularization methods mentioned is the group of dropout and methods using similar techniques. The first two are dropout and dropconnect, introduced in 2.2.1 and 2.2.2 respectively. Other methods introduced are standout, which drops more confident neurons less frequently than less confident neurons. Curriculum dropout, which first roughly sorts the training data from easier to harder and uses a time schedule to increasingly drop more neurons in network. And finally DropMaps where for a training batch each feature is kept with the retention rate p. After training the whole feature map is kept and multiplied by p.

3.2 Stochastic regularization using dropout and DropConnect

3.2.1 Dropout

Dropout is one of the most prominent stochastic regularization methods. By randomly deactivating neurons during training, dropout reduces co-adaptation and prevents memorizing specific pathways, which might lead to overfitting. Its original introduction in 2012 by Hinton et al [NS12] proved the effectiveness of using this technique across multiple datasets. Since this introduction several perspectives have been developed.
The first interpretation is that of Gal et al [GG16]. Bayesian models offer a mathematical framework

to reason about model uncertainty, with a higher computational cost than models not capturing this uncertainty. Other neural networks typically ignore this uncertainty because it is computationally unfeasible to consider all possibilities in big neural networks. Rather than considering no other possibilities, Bayesian neural networks consider a range of possibilities instead of committing to just one set of weights. In [GG16] a framework is presented to cast dropout training in neural networks as approximate Bayesian inference in deep Gaussian processes. This can be done since dropout creates many slightly different models, all with the same underlying parameters, showing similar behaviour to Bayesian networks. This method gives possibilities to study the uncertainty of dropout using Bayesian tools. This is useful because it turns dropout into a probabilistic model with theoretical foundation.
Another perspective is described by Wager et al [SWL13]. This interpretation presents dropout as adaptive regularization rather than simply considering the method as stochastic. From this standpoint, the dropout regularizer is considered to be closely connected to the L2 regularizer. By stochastically removing features the effective penalty the network experiences for each feature during training changes. This means some features get penalized based on how often they are dropped. Building on this theory, a semi-supervised method is proposed that uses unlabeled data to estimate how strong the dropout regularizer should be.

3.2.2 DropConnect

DropConnect generalizes dropout by dropping weights instead of neurons. DropConnect was introduced by Wan et al [LW13] and works as described in 2.2.2. This allows for a larger ensemble of possible models compared to dropout. Dropout samples from $2^{N_{neurons}}$ sub-networks, whereas DropConnect samples from $2^{N_{weights}}$ sub-networks.
As described in 2.2.2 a different mask is applied every forward pass, which corresponds to a different sub-network with a unique connectivity pattern. As with dropout, this stochastically reduces co-adaptation of parameters. Forcing the network to learn representations that are robust and improve generalization.
Wan et al [LW13] demonstrated empirically that DropConnect improves generalization across several benchmark datasets, including MNIST and CIFAR-10. Often outperforming both standard training and benchmarks set by dropout when applied to fully connected layers. They further defined the output of DropConnect as a mixture of models over masks M: $o = \mathbf{E}_M[f(x; \theta, M)] = \sum_M p(M) f(x; \theta, M)$. In this formula $p(M)$ are the weights, θ are the network parameters and M is the DropConnect layer mask. This formula states that the predictions are averaged over all sub-networks generated using training, this theory also mentioned in the dropout paper by Hinton et al [NS12].
Despite this empirical success, the original DropConnect paper did not provide a detailed theoretical explanation for why this ensemble averaging leads to improved generalization. Also a further explanation for choosing the retention value p should be chosen as a function of the network size or dataset complexity.

3.3 Statistical Mechanics of DropConnect

A more principled theoretical framework for understanding DropConnect was recently developed by Barboza [Bar25], who approached the problem using tools from statistical mechanics. In this framework, DropConnect is treated as an exact sum over sub-networks, allowing the construction of a partition function that encodes the statistical weights of all possible connectivity configurations. This partition function is of the form

$$Z = \sum_{\vec{k}} \left[\prod_{l=1}^{L} \binom{N_l}{k_l} p^{N_l - k_l} (1-p)^{k_l} \right] Z_{(N_1 - k_1, \ldots, N_L - k_L)}$$

Where L is the number of layers, N_l is the width of layer l, p is the retention rate and $\vec{k}$ denotes how many weights are dropped in each layer. ($Z_{(N_1-k_1,\ldots,N_L-k_L)}$ can also be written as $Z_{\vec{N}-\vec{k}}$.This formulation of the mixture of models over architectures allows for the tools of statistical mechanics to be applied.

To make analytical progress, the thesis introduces a mean field theory approach to approximate $Z_{\vec{N}-\vec{k}}$. In this approximation, each sub-network is turned into a Gaussian field characterized by an inverse propagator $A_{\vec{k}}$. DropConnect is then interpreted as a mixture of these fields, weighted by the Bernoulli probability of each sub-network configuration. Using this interpretation, expressions for the propagator and its dependence on p can be derived explicitly.

An important theoretical contribution of the thesis is the analysis of how the partition function scales with network size. Through scaling analysis it is found that when a sub-network exhibits subextensive scaling, rather than extensive scaling, the overall dropout partition shows plateau-like behaviour as a function of the dropout rate. When this is the case, the logarithm of the partition function scales slower than linearly, indicating that strong correlations reduce the effective dimensionality of the system.

To validate the framework, the DropConnect partition function formalism is applied to a rare case where an exact partition function is known, deep neural networks with Gaussian priors presented by B. Hanin and A. Zlokapa [HZ23]. By inserting this exact expression into the DropConnect sum, the plateau-like behaviour in the partition function as a function of retention probability is demonstrated as can be seen in 3.

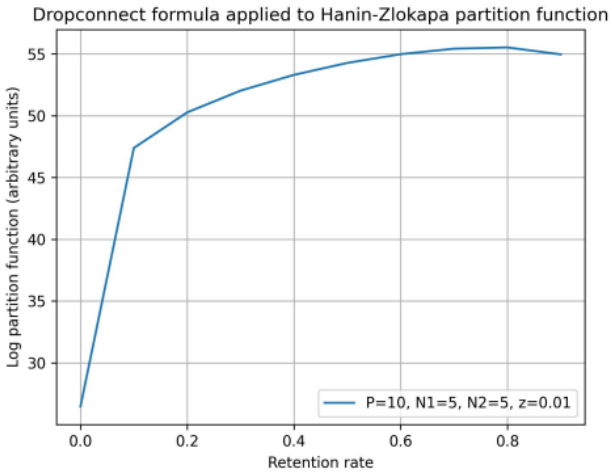
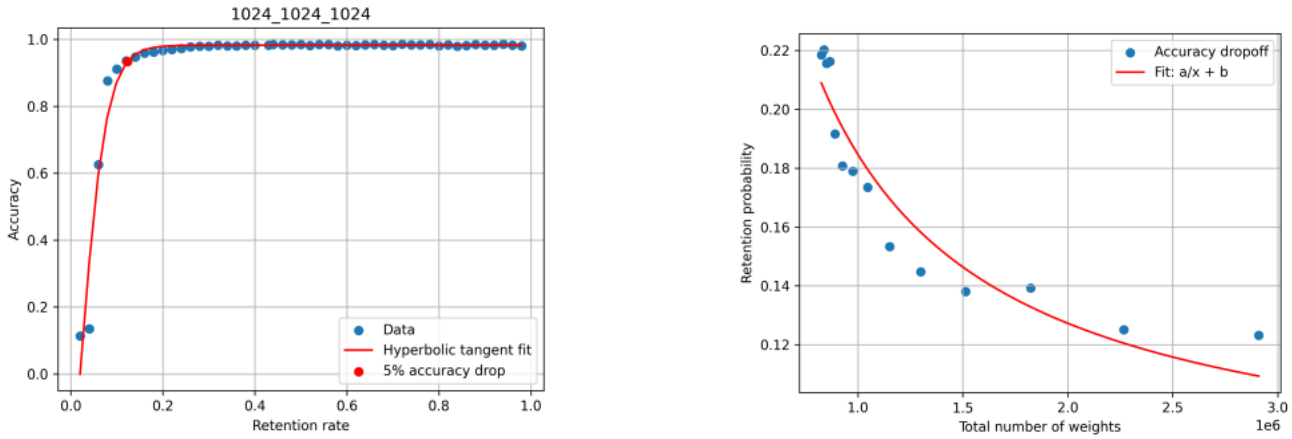


Figure 3: The plateau like structure observed when applying the dropout formula to the Hanin-Zlokapa partition function.

The theoretical predictions are then further tested using numerical experiments on the MNIST dataset. This is done by applying fully connected feed forward networks with ReLU activations using DropConnect while systematically varying the network size and retention rate. From these experiments clear plateau behaviour in test accuracy can be seen as a function of the retention rate, like in 5a. Furthermore a sharp drop in performance can be seen below a critical retention rate and an inverse relation between the critical retention probability and network size can be seen in 12b. These findings are in line with the predicted scaling of the critical retention rate: $p_{crit} \propto \frac{1}{Nc}$.



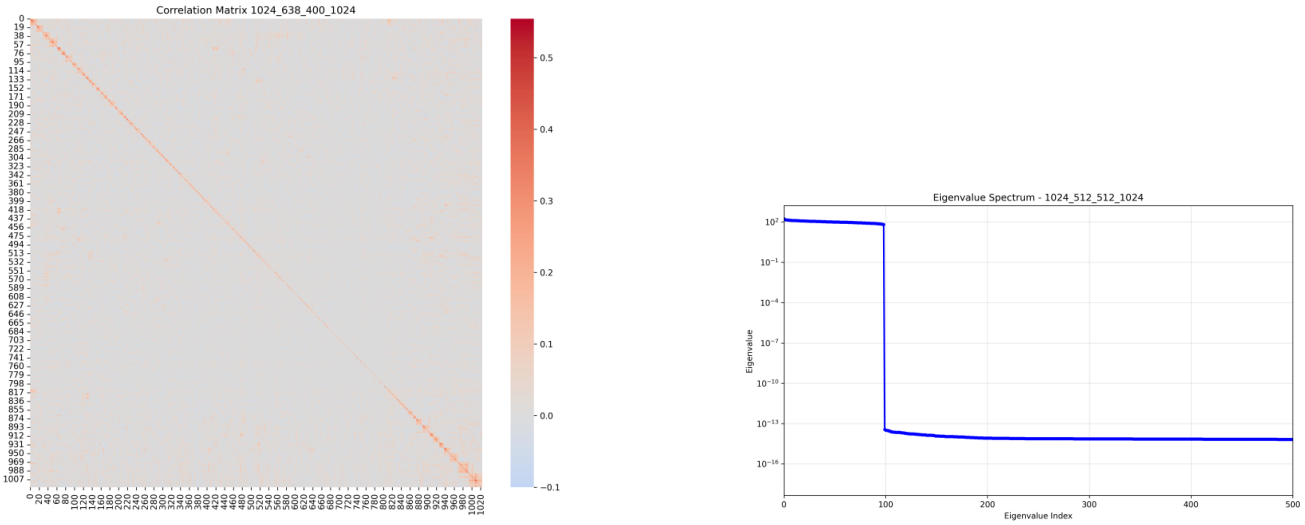
(a) Retention rate vs accuracy

(b) Critical retention rate vs network size

Figure 4: Two plots visualizing the critical retention rate. Plot a visualizes the accuracy as a function of the retention rate. Plot b visualizes the critical retention rate as a function of the network size.

The thesis proceeds by exploring the hypothesis that DropConnect works best when the network exhibits strong internal correlations. Using concepts from statistical mechanics, it is argued that

correlations reduce the number of independent degrees of freedom. This phenomenon is known as system size compression. This idea is illustrated using an analogy with spin systems and block diagonal correlation matrices. If parameters cluster into correlated groups, the effective configuration space shrinks from exponential to power-law size, enabling the subextensive scaling required for the plateau behaviour.
This hypothesis is tested by computing correlation matrices of the trained networks and performing principal component analysis. Surprisingly, the effective dimensionality remained roughly the same across different architectures with the same number of weights.



(a) Correlation matrix example

(b) Eigenvalue PCA analysis

Figure 5: Two examples of the results from the thesis of Barboza [Bar25]

These results suggest that the MNIST dataset is too simple to fully test the correlation structure hypothesis. Here a diagnostic tool is introduced to detect if a network size greatly exceeds the size needed for the dataset, that is if the number of parameters greatly exceeds the effective dimensionality given by PCA analysis.
This is also one of the conclusions drawn in the thesis. The other conclusions are that DropConnect can be interpreted as an exact ensemble over sub-networks, plateau behaviour arises naturally in networks featuring subextensive scaling and that the critical retention probability scales inversely with network size.

4 Methods

4.1 Overview of Experimental Design

The experiments wish to answer the question what the relationship is between the efficacy of DropConnect and the block-diagonal structure of the parameter correlation matrices across different architectures. The PCA analysis done by Barboza [Bar25] will be repeated on an ensemble of 1000 instead of 100 networks to validate the hypothesis that MNIST is too small. This is done because the PCA analysis being capped at 100 features could arise from the fact that 100 networks are

being used. Testing this hypothesis with a larger number of networks would give definitive results on the question if MNIST is too simple. This research will be done in three stages. In the first stage multiple ensembles of identical neural networks will be trained to construct parameter ensembles. In the second stage the covariance and correlation matrices of the trained parameters are computed with the help of PCA and eigenvalue decomposition. Finally these results are analyzed and a conclusion is drawn based on the complexity found with the PCA analysis and the correlation structure found.

4.2 Datasets

4.2.1 MNIST

The MNIST dataset is a benchmark dataset of grayscale handwritten numbers from 0-9, used for testing classification tasks. The MNIST dataset consists of 60000 labeled training samples and 10000 test samples. These samples have a resolution of 28×28 pixels, resulting in 784 features per image. The values of these features describe how dark that pixel is. An example of the MNIST dataset is shown in 6.

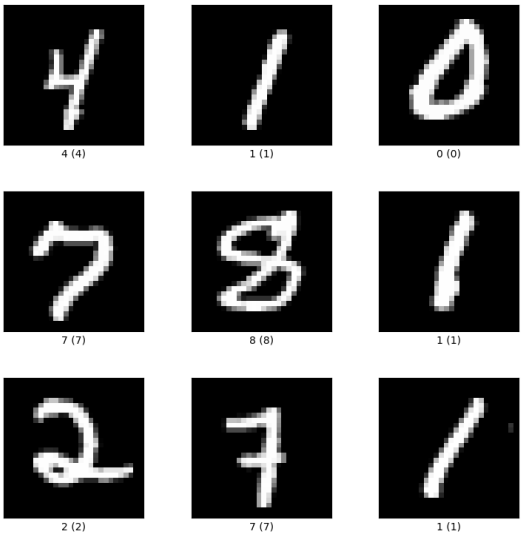


Figure 6: Small sample from the MNIST dataset

4.2.2 CIFAR-10

The CIFAR-10 dataset is also a benchmark dataset for classification tasks, but a bit more complicated than MNIST. The dataset is a subset of the 80 million tiny images dataset, just like CIFAR-100. The dataset consists of 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. These classes are completely mutually exclusive, meaning that there is no overlap between two classes. The images are presented as 32×32 color images divided across 50000 training images and 10000 testing images. The dimensionality of this dataset is significantly higher than the MNIST

dataset, since CIFAR-10 has 3072 features compared to the 768 features from MNIST. A small example of images in CIFAR 10 can be seen in 7.

Figure 7: A small example of labeled images from the CIFAR-10 dataset

4.3 Neural network architecture

All experiments will be done on multilayer perceptrons as described in 2.1. This choice aligns with the theoretical assumptions made by Barboza in [Bar25], since this guarantees compatibility with the statistical-mechanical treatment of DropConnect. This is done by directly accessing the flattened weight vectors and avoids obscure correlation structures.
The network implemented features one input layer of size 28×28 for MNIST and $32 \times 32 \times 3$ for CIFAR-10. There are four hidden layers, the first and fourth layer are of size $L = 1024$ and the second and third layers have a varying size. A visual representation of the network can be seen in 8. The sizes are varied across different model ensembles to study the architectural bottlenecks on correlation structure. The values for N_1 and N_2 and two more values M_1 and M_2 are selected by applying the following formula:

$$LN_1 + N_1N_2 + LN_2 = LM_1 + M_1M_2 + LM_2$$

This formula can be adapted by adding L^2 to both sides:

$$(L + N_1)(L + N_2) = (L + M_1)(L + M_2)$$

Then this can be rewritten as:

$$u_1u_2 = v_1v_2$$

With $u_1 = (L+N_1)$, $u_2 = (L+N_2)$, $v_1 = (L+M_1)$ and $v_2 = (L+M_2)$. From this a common divisor k will be considered such that $v_1 = u_1/k$, then $v_2 = u_2k$ satisfies $u_1u_2 = v_1v_2$. With the requirement that $u_1, u_2, v_1, v_2 > 1024$. So k will be selected that this remains true. The first variables used are the same as in the thesis by Barboza [Bar25]. These values are $N_1 = 638, N_2 = 400$ and $M_1 = 176, M_2 = 1012$. For further research there was another set added. This set had values $N_1 = 316, N_2 = 891$. When applying $k = \frac{4}{5}$ to $u_1 = 1024 + 316 = 1340$ and $u_2 = 1024 + 891 = 1915$. We get $v_1 = \frac{1340}{4/5} = 1340 * \frac{5}{4} = 1675$ and $v_2 = 1915 * \frac{4}{5}$. Both v_1 and v_2 are > 1024, thus satisfy the

constraint. Converting v_1 and v_2 back to M_1 and M_2 we get: $M_1 = v_1 - 1024 = 1675 - 1024 = 651$ and $M_2 = v_2 - 1024 = 1532 - 1024 = 508$.

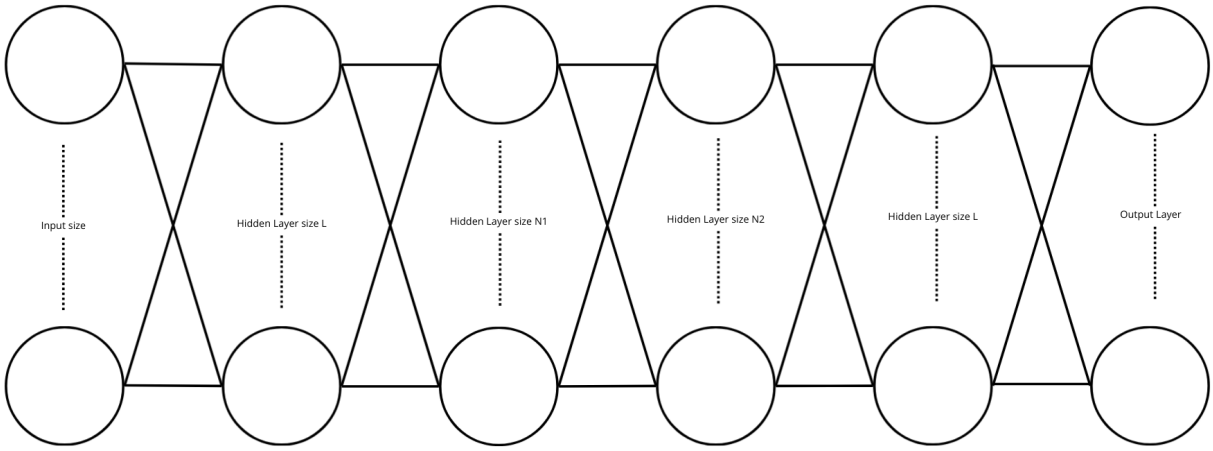


Figure 8: A visual representation of the MLP architecture. The input size is $28 \times 28 = 784$ for MNIST and $32 \times 32 \times 3 = 3072$ for CIFAR-10. The size L is set at 1024 and the sizes N_1 and N_2 are varied. The output size is 10 for both MNIST and CIFAR-10.

DropConnect is applied to all hidden layers, using a custom linear model implementation. In each custom layer, each individual weight is retained with probability p and set to zero at probability $1 - p$. Formally, in each the mask, as introduced in 2.2.2, the weights in a layer are multiplied by a Bernoulli mask. This version reaches the same end results, but not by applying a mask. It is reached by setting the weights individually. Effectively mirroring the interpretation of DropConnect as sub-network sampling.

4.4 Training and model ensemble

All models will be using the Adam optimizer for training with a fixed learning rate of 10^{-3}. The loss function is the standard cross-entropy loss. Models will be trained using mini-batches of size 128 over 100 epochs. No other methods other than DropConnect are used. To limit unnecessary training early stopping is implemented with a patience of 3. The parameters corresponding to the highest accuracy are retained.
With this setup a total of 200 independently trained networks are made and stored in .pth files.

This is done by repeating the full training sequence and storing the models, giving a collection of models that can be treated as ensemble for a specific architecture.
Both MNIST and CIFAR-10 are handled within in this code. The code loops over the four architectures described in 4.3. The models are stored in folders $mnist/\{N_1\}\_\{N_2\}\_\{L\}$ and $cifar\_10/\{N_1\}\_\{N_2\}\_\{L\}$. The models are stored with files formatted as: $\{n_{experiment}\}\_\{dropoutrate\}.pth$. The retention rate used to produce the ensemble is 1.0, giving a dropout rate of 0.0.

4.5 Retention rates

As described in 2.2.2, dropconnect works by applying a mask where the weights are removed at a factor of $1-p$, where p is the retention rate. To explore the relation between the retention and the accuracy, p is spread uniformly over [0.02, 1.00] with 50 intervals. This gives a list of equally spaced p values from 0.02 to 1.00 with 0.02 between each entry.
Using this setup the retention rate is tested systematically across the architectures and datasets. The p value and its corresponding accuracy are saved in a csv file of the form $acc\_vs\_p\_\{N_1\}\_\{N_2\}\_\{L\}\_\{dataset\}.csv$, which can be used to create the plots to analyze the plateau effect and fit a line approximating the point at which the accuracy dropped to 95% of its original value. The models itself are also saved in the same way as described in 4.4. This is done to prevent having to train the model again if saving the csv file fails.
From these data points, a scatter plot is made with a smooth curve fitting to the points. Using this curve, a critical retention rate is defined by the smallest value of p for which the accuracy reaches at least 95% of the accuracy obtained at the highest retention rates. This provides a measure for comparing robustness to weight removal across different architectures.

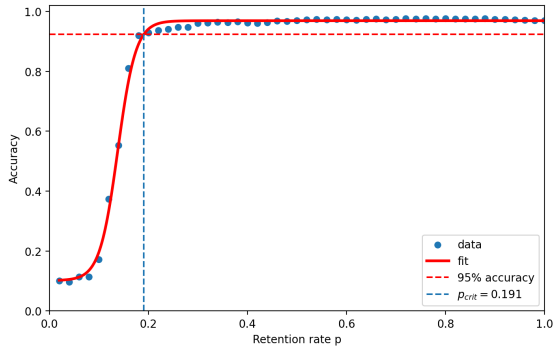
4.6 PCA Analysis of the Covariance matrix and Correlation matrix

The final layer of the models is analyzed using the parameter ensemble method as described in 2.3. Each column of the matrix is a flattened weight vector of the final layer of a trained network. With the knowledge that followed from 2.4 and 2.6 that there are at most $M-1$ nonzero eigenvalues for covariance matrix C, we can explain the sudden drop in eigenvalues observed in the thesis by Barboza [Bar25]. To validate the assumption that MNIST is not complex enough, a new experiment is done using an ensemble of size 1224, which is very close to $1.2 \times N^{\frac{3}{4}}$ which gives 1221 from 10240 weights in the final layer. Due to limited time to conduct this research, it was done only on the architecture of parameters $N_1 = 638, N_2 = 400, L = 1024$. The eigenvalues are computed from the flattened weight vectors using the definition in 2.6. On all other architectures the PCA analysis is also conducted, although it may not allow for a conclusive outcome since M is limited.
The correlation matrix is then constructed using the $corrcoef$ function from numpy. This function normalizes the covariance matrix by the variance of each parameter, giving a measure of linear dependence between pairs of weights. The numpy function follows the definition given in 2.3.
To reveal the potential block-diagonal structure, also mentioned in 2.3, the correlation matrix is reordered using hierarchical clustering based on a distance measure which is defined as $1 - |correlation\_value|$. This reordering groups strongly correlated parameters together, which allows for easier empirical identification of correlated groups. Then the reordered matrix is down-sampled using block averaging to preserve larger scale structure while reducing noise.

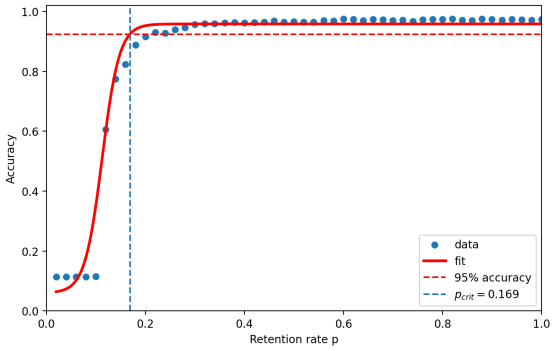
5 Results

5.1 MNIST

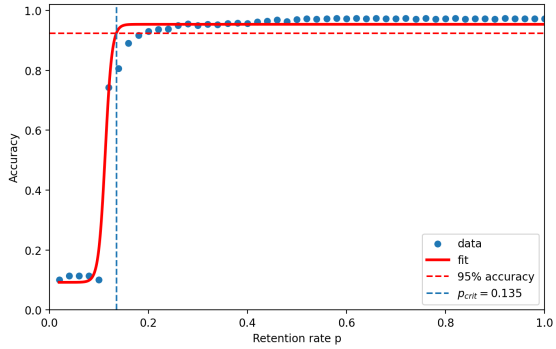
5.1.1 Accuracy and retention rate



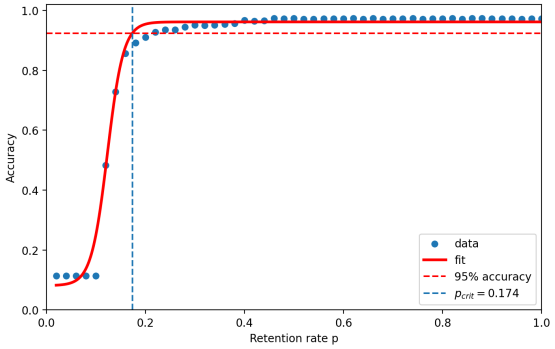
(a) Accuracy as a function of the retention rate for $N_1 = 176, N_2 = 1012$



(b) Accuracy as a function of the retention rate for $N_1 = 638, N_2 = 400$



(c) Accuracy as a function of the retention rate for $N_1 = 316, N_2 = 891$

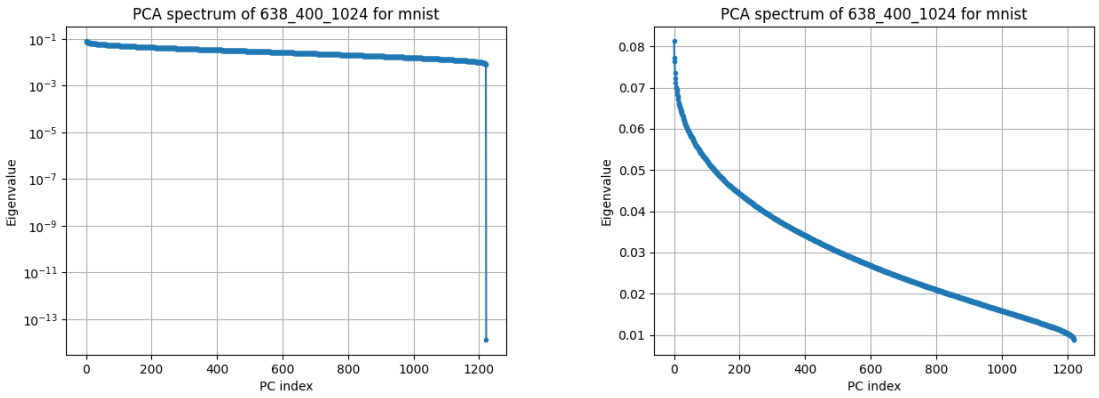


(d) Accuracy as a function of the retention rate for $N_1 = 651, N_2 = 508$

Figure 9: Plots of accuracy as a function of the retention rate for various architectures on the MNIST dataset.

In 9 the accuracy plots can be seen for the different architectures trained on the MNIST dataset. Here it can be seen that the critical p value varies across different architectures. Also there is empirical evidence for the plateau effect across all networks and there is rapid performance degradation from the critical retention rate.

5.1.2 Eigenvalue spectra



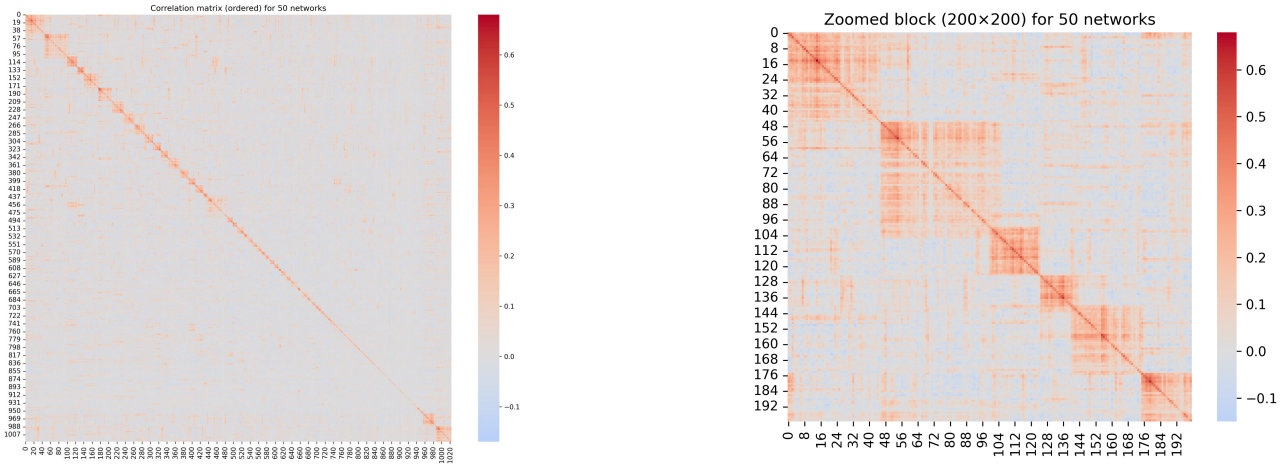
(a) Eigenvalue spectrum of 638_400_1024 for an ensemble of 1224 networks on the MNIST dataset

(b) Eigenvalue spectrum of 638_400_1024 for an ensemble of 1224 networks on the MNIST dataset without the last entry for readability

Figure 10

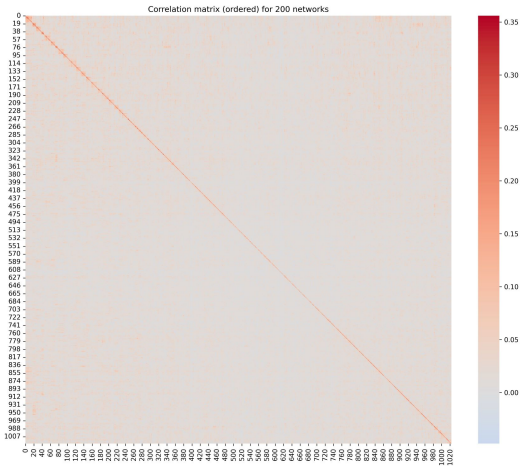
Figure 10 shows the eigenvalue spectrum of the covariance matrix for an ensemble of 1224 independently trained networks using $N_1 = 638, N_2 = 400, L = 1024$ on the MNIST dataset. Other eigenvalue spectra can be seen in 16 for ensembles featuring 200 networks. With the increased ensemble size, a rapid decay in the spectrum can be seen, indicating that the effective dimensionality is in fact too is too low for the number of parameters. This finding is in line with the theory that the network size is too large for the MNIST dataset, as proposed by Barboza [Bar25].

5.1.3 Correlation matrices

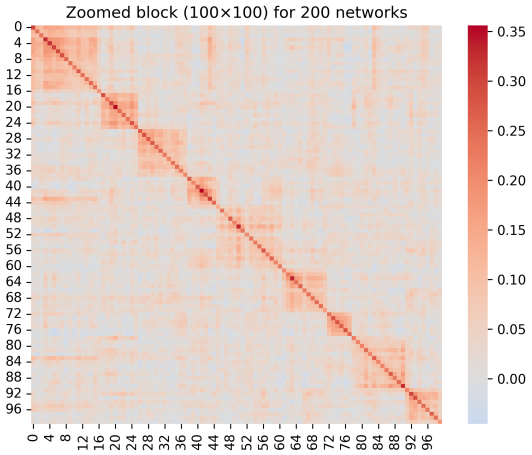


(a) Correlation matrix of architecture 316_891_1024 using 50 networks trained on the MNIST dataset

(b) Zoomed in snapshot of the correlation matrix of architecture 316_891_1024 using 50 networks trained on the MNIST dataset



(c) Correlation matrix of architecture 316_891_1024 using 200 networks trained on the MNIST dataset



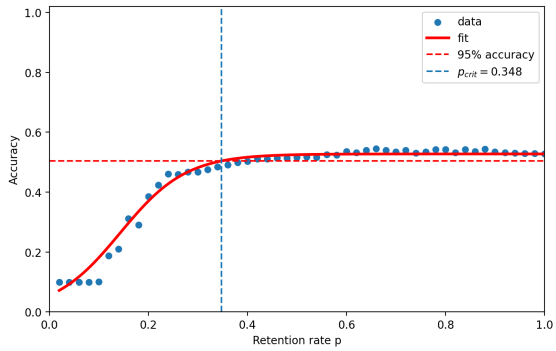
(d) Zoomed in snapshot of the correlation matrix of architecture 316_891_1024 using 200 networks trained on the MNIST dataset

Figure 11

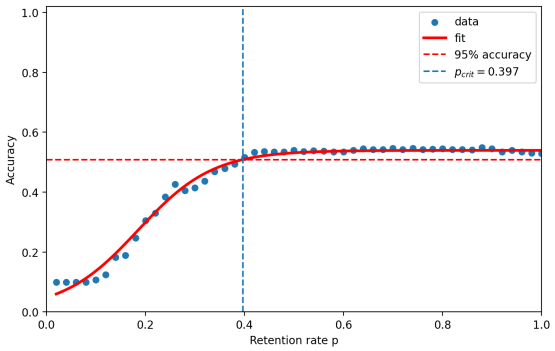
In 11 the correlation matrix can be seen for the architecture with $N_1 = 316, N_2 = 891, L = 1024$, trained on the MNIST dataset. More plots can be seen in 17 and 18. In these plots it can be seen that the block structure persists for both 50 networks and 200 networks. This stability suggests that parameter co-adaptation is already saturated and does not increase with ensemble size.

5.2 CIFAR-10

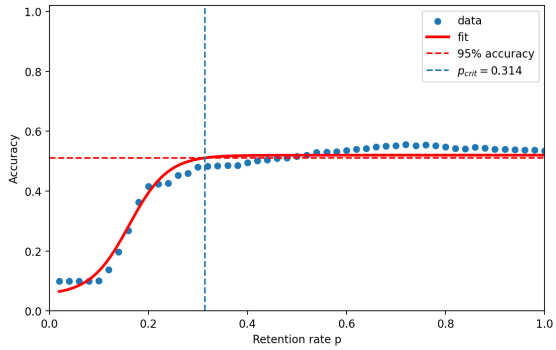
5.2.1 Accuracy and retention rates



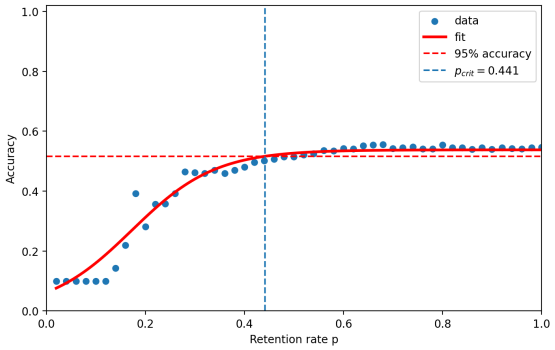
(a) Accuracy as a function of the retention rate for $N_1 = 176, N_2 = 1012$



(b) Accuracy as a function of the retention rate for $N_1 = 638, N_2 = 400$



(c) Accuracy as a function of the retention rate for $N_1 = 316, N_2 = 891$

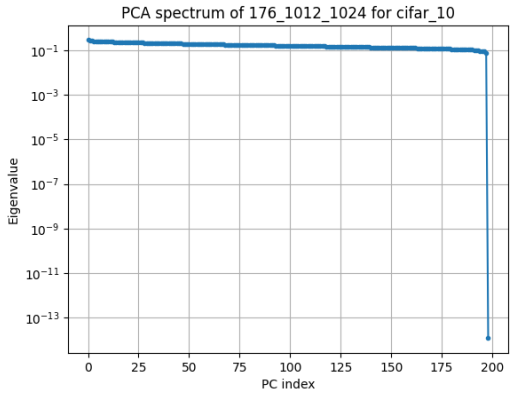


(d) Accuracy as a function of the retention rate for $N_1 = 651, N_2 = 508$

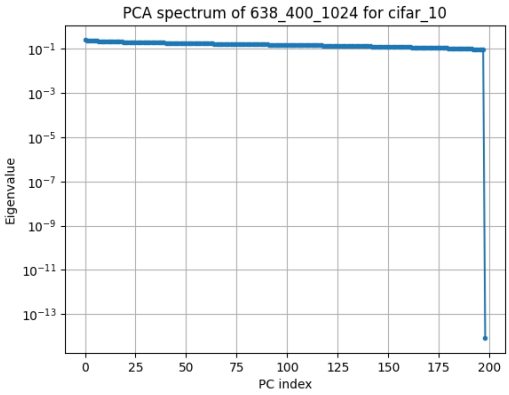
Figure 12: Plots of accuracy as a function of the retention rate for various architectures on the CIFAR-10 dataset.

As can be seen in 12, CIFAR-10 exhibits plateau-like behaviour in test accuracy as a function of the retention rate across all architectures. This effect however, is less pronounced compared to the MNIST dataset and the performance degradation is more gradual. This suggests that, though DropConnect remains effective, the robustness to DropConnect is reduced compared to simpler datasets.

5.2.2 PCA analysis



(a) Eigenvalue spectrum of 176_1012_1024 for an ensemble of 200 networks on the CIFAR-10 dataset

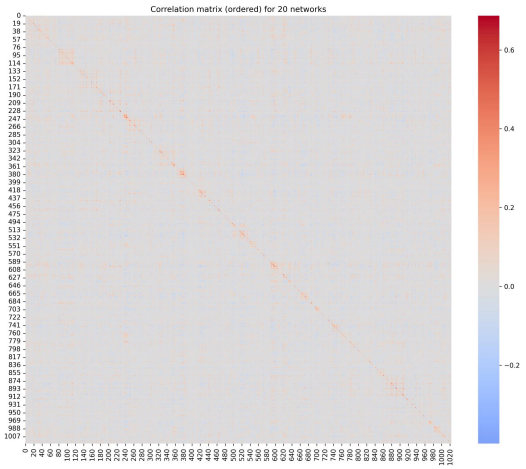


(b) Eigenvalue spectrum of 638_400_1024 for an ensemble of 200 networks on the CIFAR-10 dataset

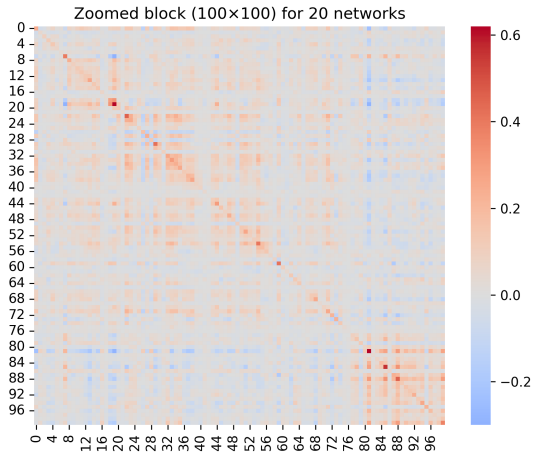
Figure 13

Across all architectures, as depicted in 13 and 15, a consistent trend in decay can be seen for the eigenvalues. When comparing this to the trend found in 16, no concrete differences can be seen for ensembles featuring 200 networks.

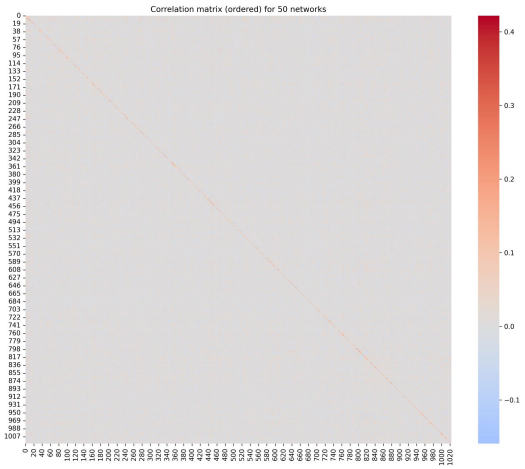
5.2.3 Correlation matrices



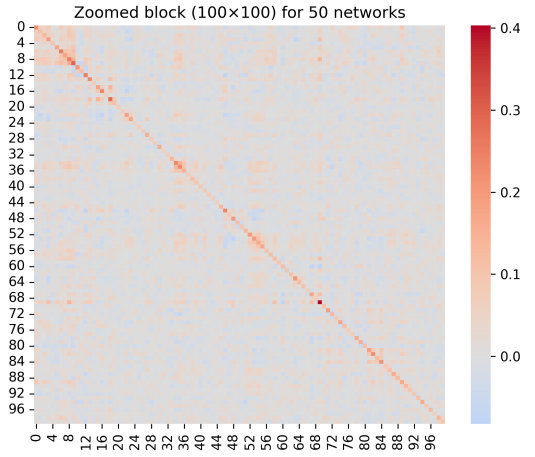
(a) Correlation matrix of architecture 316_891_1024 using 20 networks trained on the CIFAR-10 dataset



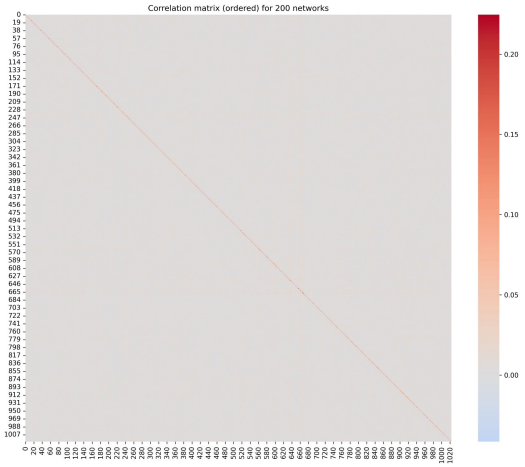
(b) Zoomed in snapshot of the correlation matrix of architecture 316_891_1024 using 20 networks trained on the CIFAR-10 dataset



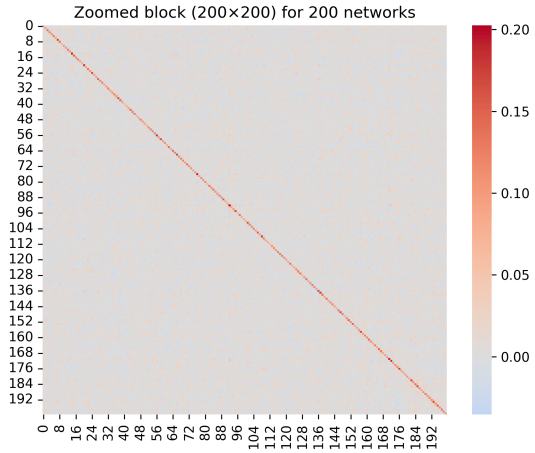
(c) Correlation matrix of architecture 316_891_1024 using 50 networks trained on the CIFAR-10 dataset



(d) Zoomed in snapshot of the correlation matrix of architecture 316_891_1024 using 50 networks trained on the CIFAR-10 dataset



(e) Correlation matrix of architecture 316_891_1024 using 200 networks trained on the CIFAR-10 dataset



(f) Zoomed in snapshot of the correlation matrix of architecture 316_891_1024 using 200 networks trained on the CIFAR-10 dataset

Figure 14

In 14 the correlation matrix can be seen for the architecture with $N_1 = 316, N_2 = 891, L = 1024$ trained on the CIFAR-10 dataset. More plots can be seen in 20 and 21. In these plots a faint block structure can be seen in the correlation matrices using 20 networks, but these weaken when the amount of networks is increased as follows from the correlation matrices computed using 50 and 200 networks. This suggest that the block pattern is not persistent as the pattern is not visible when the ensemble size grows beyond a certain M.

In contrast to MNIST, the lack of block pattern stability observed using CIFAR-10 suggests that when using these architectures the predicted compression regime is not present. This means that using these architectures and this dataset, the condition for robust block-diagonal correlation

patterns, as predicted by Barboza, is not met.

6 Conclusions and Further Research

6.1 Conclusion

In this thesis the relationship between the effectiveness of DropConnect and the internal correlation structure of parameters in a network was investigated, following the research and statistical-mechanical framework introduced by Barboza [Bar25]. Using ensembles of interdependently trained MLPs with varying architectures, the robustness of DropConnect was analyzed together with the eigenvalue spectra and correlation matrices of the parameter ensembles.
For the MNIST dataset, the conclusion drawn by Barboza is confirmed. The accuracy-retention plots exhibit a pronounced plateau for all considered architectures. PCA analysis revealed that the effective dimensionality is saturated at a far lower value than the number of parameters and remains invariant across different architectures. Increasing the ensemble size reveals that the effective dimensionality is intrinsically low, which confirms that the dataset is too simple for the architectures considered. The correlation matrices show stable correlation structure under increasing ensemble size. Together, these findings confirm that MNIST is too simple to serve as a decisive test case for the hypothesis Barboza introduced.
For CIFAR-10 there is empirical evidence for a different correlation structure. While a block-diagonal structure is observed using smaller ensemble sizes, this phenomenon does not remain stable when the ensemble size grows. This indicates that the block structure is not a persistent feature of the learned solutions. Although plateau-like behaviour is produced in the accuracy as a function of the retention rate, this plateau is less pronounced and performance degradation occurs more gradually compared to MNIST.
Taken together, these results show evidence that DropConnect does not require strong block-diagonal correlation structures. However, when such structures do emerge, this coincides with enhanced robustness to weight removal as a more pronounced plateau and accuracy drop below p_{crit} can be seen. For the architectures used, MNIST satisfies this condition whereas CIFAR-10 does not.

6.2 Further Research

There are several directions for future research from this point. Firstly the analysis for CIFAR-10 could be repeated on wider and deeper networks to test whether the block-diagonal correlation structure only arises when model capacity greatly exceeds dataset complexity. Furthermore the research could be repeated on other, intermediate, datasets to further clarify how dataset structure, architecture and correlation structure influence the effectiveness of DropConnect.

References

[Bar25] G. Barboza. Generalized statistical mechanics of dropconnect. Master's thesis, Universiteit van Amsterdam, 2025.

[GG16] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[HZ23] B. Hanin and A. Zlokapa. Bayesian interpolation with deep linear networks. *Proceedings of the National Academy of Sciences*, 120(23):e2301345120, 2023.

[IGC16] Y. Bengio I. Goodfellow and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[LW13] S. Zhang Y. Le Cun R. Fergus L. Wan, M. Zeiler. Regularization of neural networks using dropconnect. *Proceedings of the 30th International Conference on Machine Learning*, 28:1058–1066, 2013.

[NS12] A. Krizhevsky I. Sutskever R. Salakhutdinov N. Srivastava, G. Hinton. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[NS14] A. Krizhevsky I. Sutskever R. Salakhutdinov N. Srivastava, G. Hinton. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[SWL13] S. Wang S. Wager and Percy Liang. Dropout training as adaptive regularization. *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 1:351–329, 2013.

[TZ22] Y. Tian and Y. Zhang. A comprehensive survey on regularization strategies in machine learning. *Information fusion*, 80:146–166, 2022.

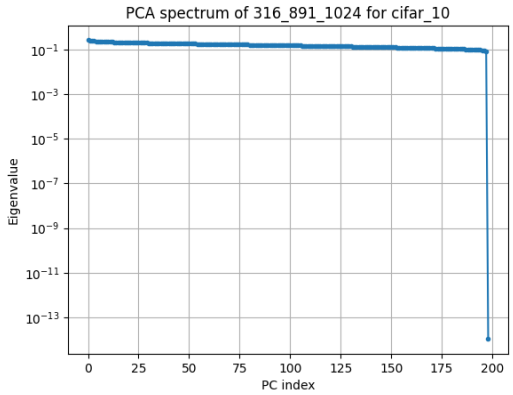
[XM19] C. Zhou M.A. Helvie H. Chan L.M. Hadjiiski Y. Lu X. Ma, J. Wei. Automated pectoral muscle identification on mlo-view mammograms: Comparison of deep neural network to conventional computer vision. *Medical Physics (Lancaster)*, 46(5):2103–2114, 2019.

7 Appendix

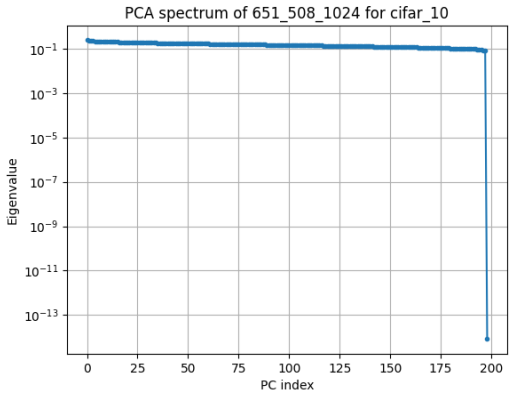
7.1 Code

The code used in this thesis can be reached using this link: https://github.com/JustinMeurs/BachelorThesisProject

7.2 Extra eigenvalue spectra

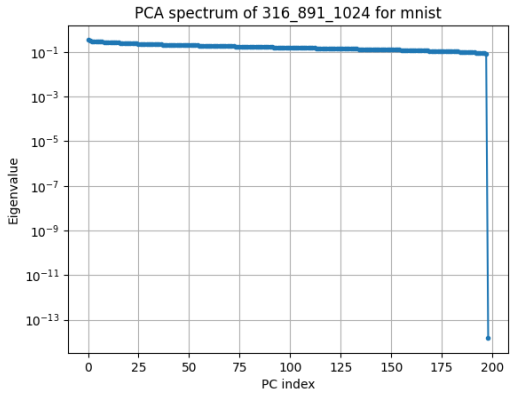


(a) Eigenvalue spectrum of 316_891_1024 for an ensemble of 200 networks on the CIFAR-10 dataset

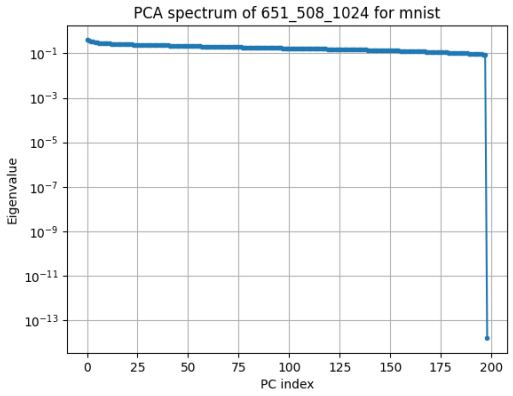


(b) Eigenvalue spectrum of 651_508_1024 for an ensemble of 200 networks on the CIFAR-10 dataset

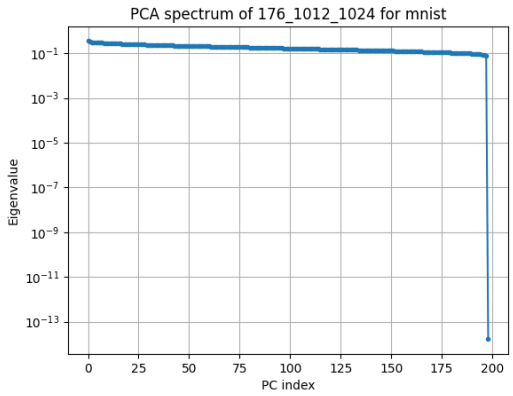
Figure 15



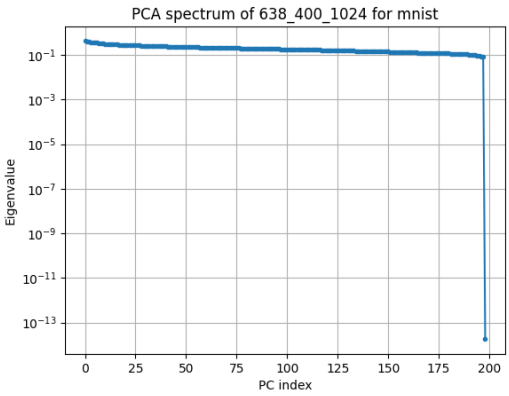
(a) Eigenvalue spectrum of 316_891_1024 for an ensemble of 200 networks on the MNIST dataset



(b) Eigenvalue spectrum of 651_508_1024 for an ensemble of 200 networks on the MNIST dataset



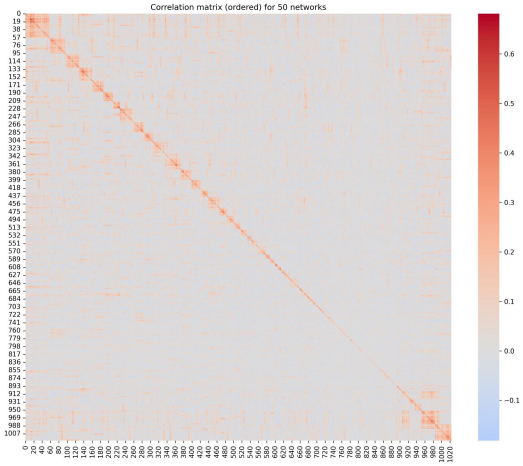
(c) Eigenvalue spectrum of 176_1012_1024 for an ensemble of 200 networks on the MNIST dataset



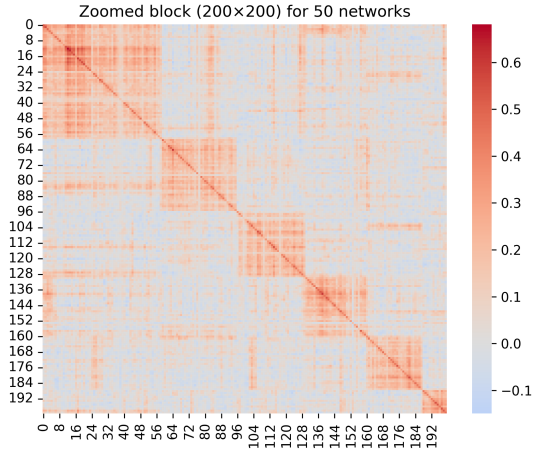
(d) Eigenvalue spectrum of 638_400_1024 for an ensemble of 200 networks on the MNIST dataset

Figure 16

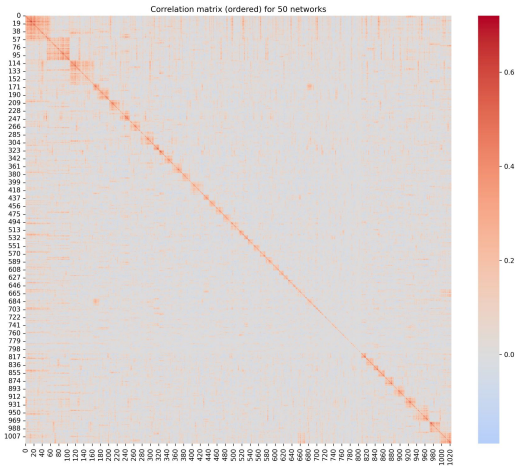
7.3 Extra correlation matrices



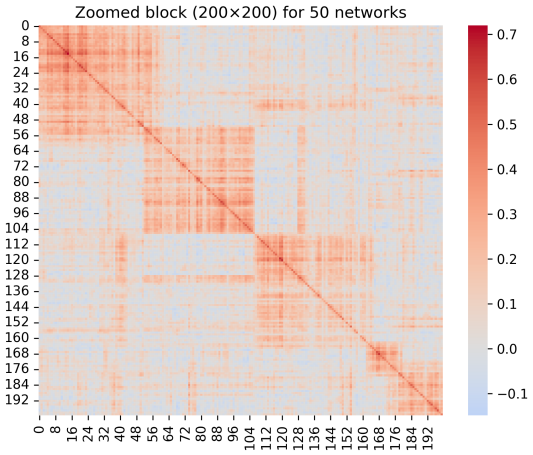
(a) Correlation matrix of architecture 176_1012_1024 using 50 networks trained on the MNIST dataset



(b) Zoomed in snapshot of the correlation matrix of architecture 176_1012_1024 using 50 networks trained on the MNIST dataset

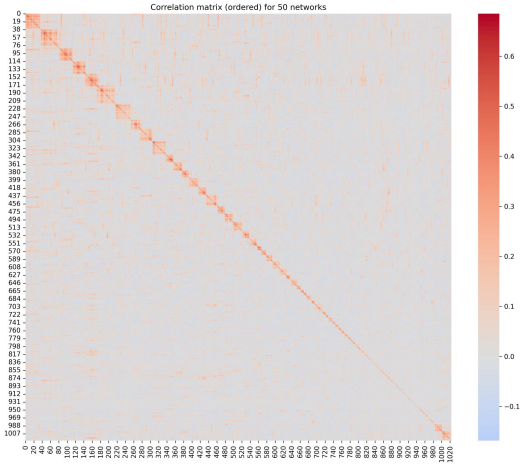


(e) Correlation matrix of architecture 651_508_1024 using 50 networks trained on the MNIST dataset

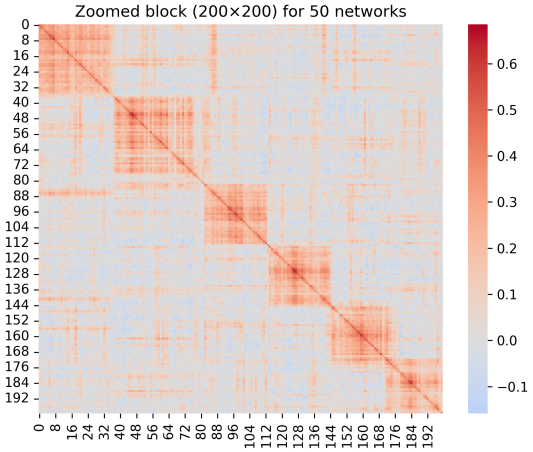


(f) Zoomed in snapshot of the correlation matrix of architecture 651_508_1024 using 50 networks trained on the MNIST dataset

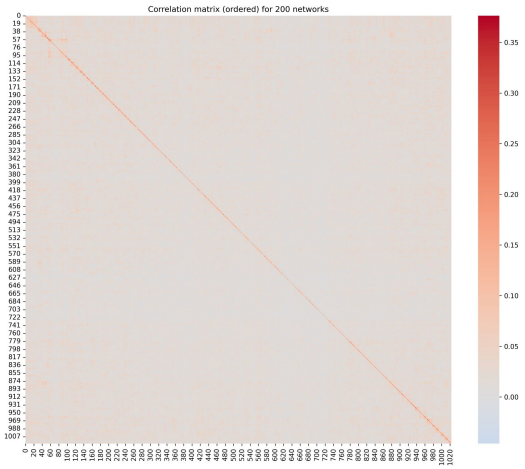
Figure 17



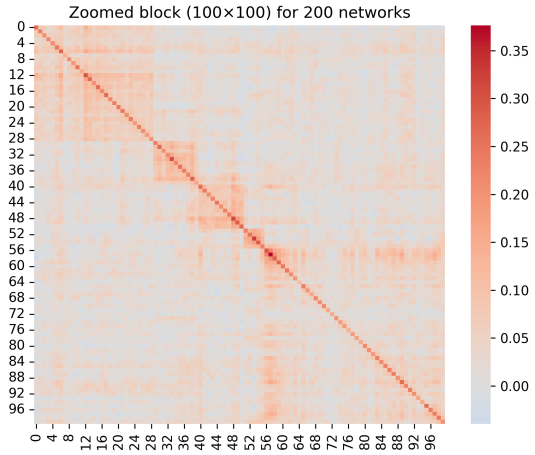
(c) Correlation matrix of architecture 638_400_1024 using 50 networks trained on the MNIST dataset



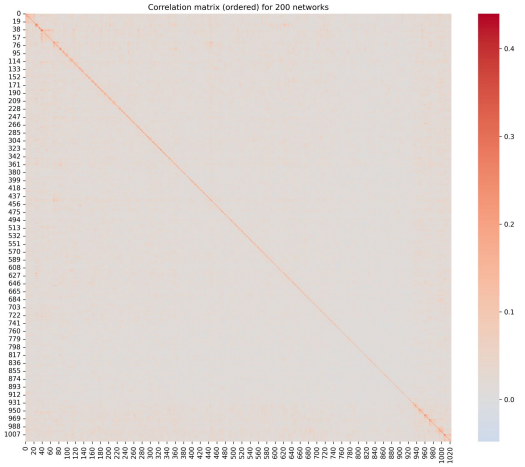
(d) Zoomed in snapshot of the correlation matrix of architecture 638_400_1024 using 50 networks trained on the MNIST dataset



(a) Correlation matrix of architecture 176_1012_1024 using 200 networks trained on the MNIST dataset

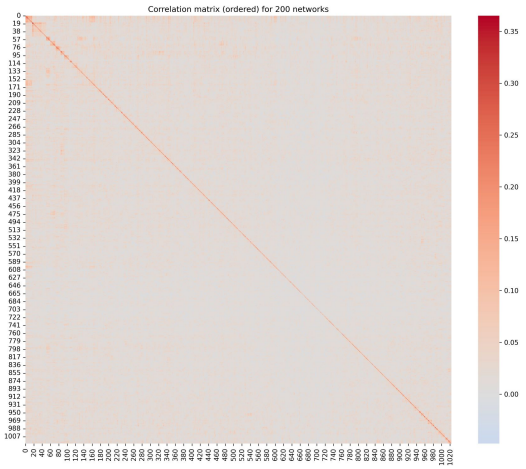


(b) Zoomed in snapshot of the correlation matrix of architecture 176_1012_1024 using 200 networks trained on the MNIST dataset

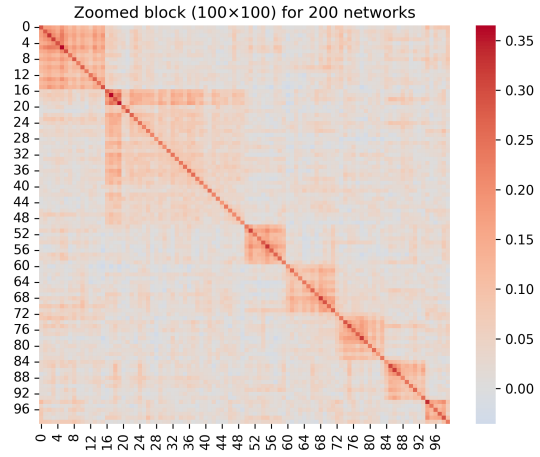


(c) Correlation matrix of architecture 638_400_1024 using 200 networks trained on the MNIST dataset

(d) Zoomed in snapshot of the correlation matrix of architecture 638_400_1024 using 200 networks trained on the CIFAR-10 dataset

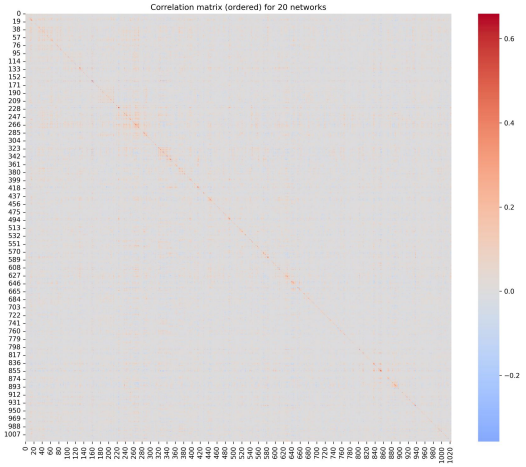


(e) Correlation matrix of architecture 651_508_1024 using 200 networks trained on the MNIST dataset

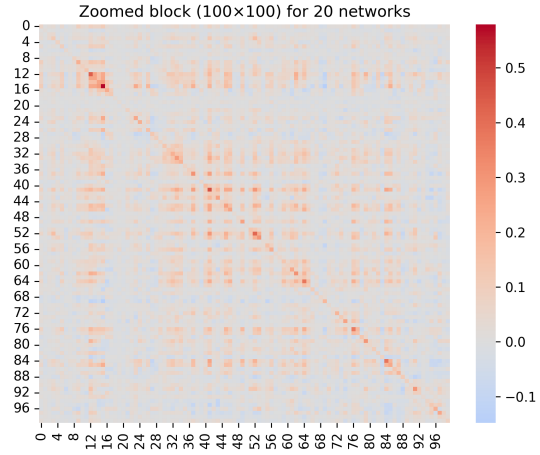


(f) Zoomed in snapshot of the correlation matrix of architecture 651_508_1024 using 200 networks trained on the MNIST dataset

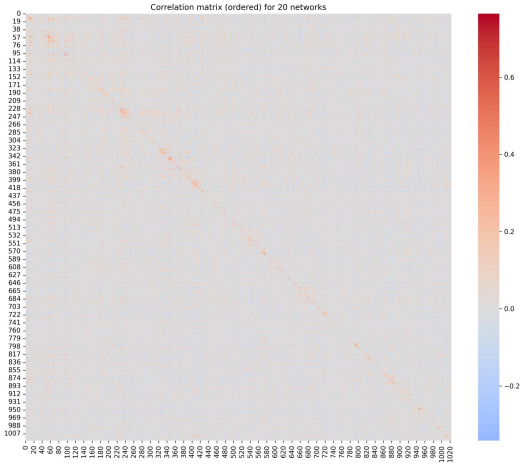
Figure 18



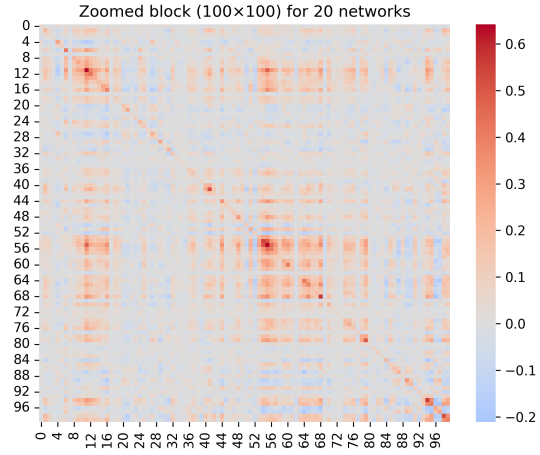
(a) Correlation matrix of architecture 176_1012_1024 using 20 networks trained on the CIFAR-10 dataset



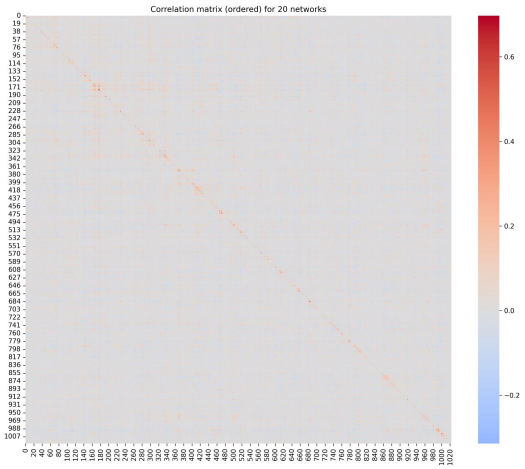
(b) Zoomed in snapshot of the correlation matrix of architecture 176_1012_1024 using 20 networks trained on the CIFAR-10 dataset



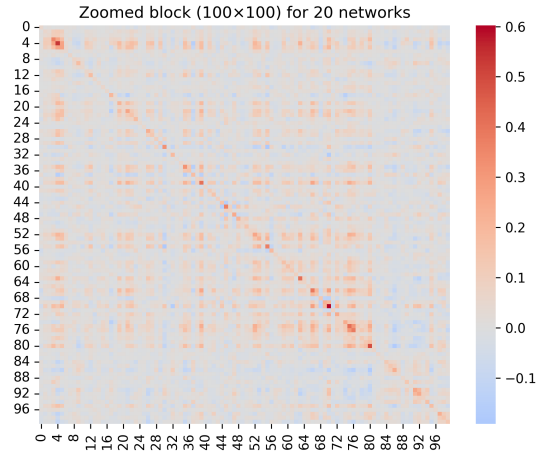
(c) Correlation matrix of architecture 638_400_1024 using 20 networks trained on the CIFAR-10 dataset



(d) Zoomed in snapshot of the correlation matrix of architecture 638_400_1024 using 20 networks trained on the CIFAR-10 dataset

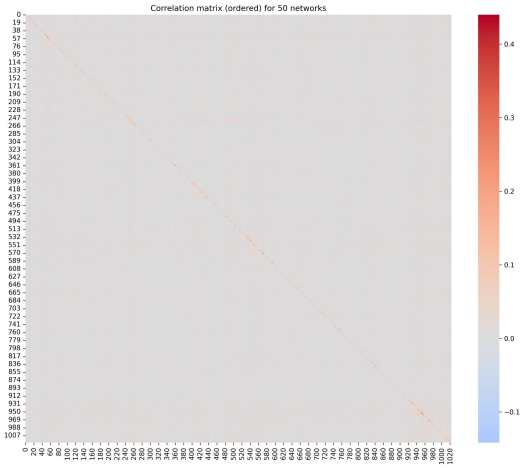


(e) Correlation matrix of architecture 651_508_1024 using 20 networks trained on the CIFAR-10 dataset

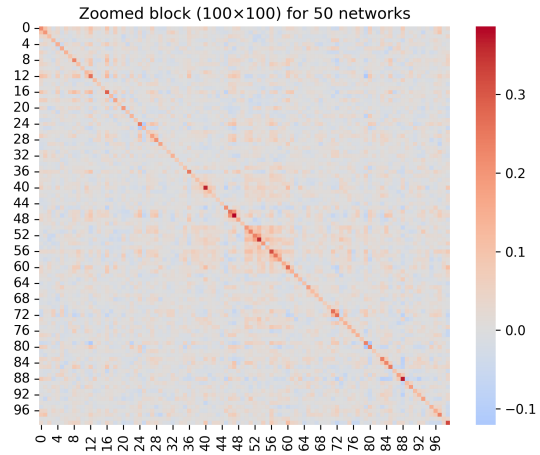


(f) Zoomed in snapshot of the correlation matrix of architecture 651_508_1024 using 20 networks trained on the CIFAR-10 dataset

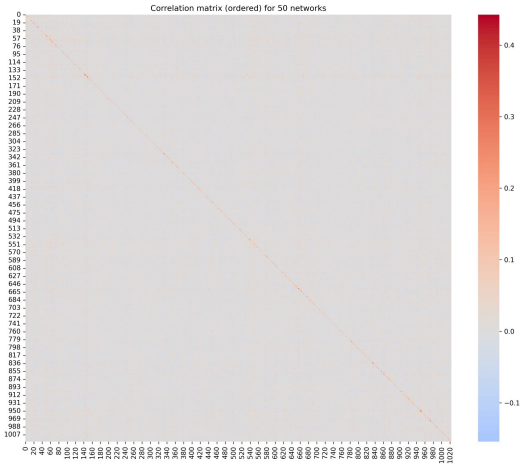
Figure 19



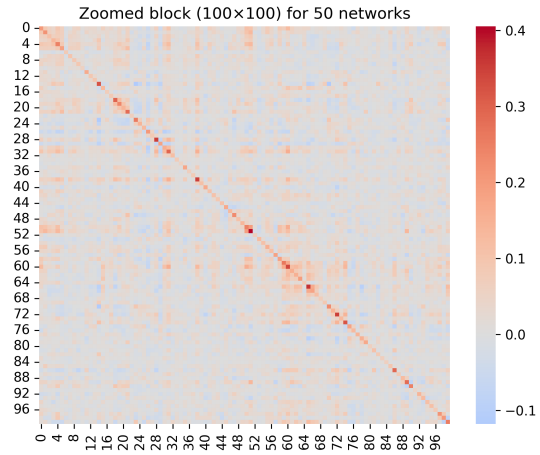
(a) Correlation matrix of architecture 176_1012_1024 using 50 networks trained on the CIFAR-10 dataset



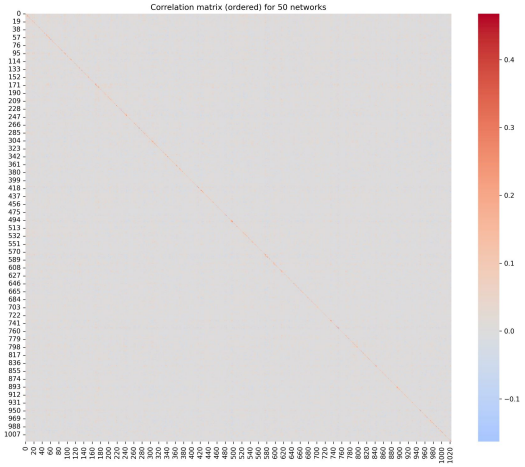
(b) Zoomed in snapshot of the correlation matrix of architecture 176_1012_1024 using 50 networks trained on the CIFAR-10 dataset



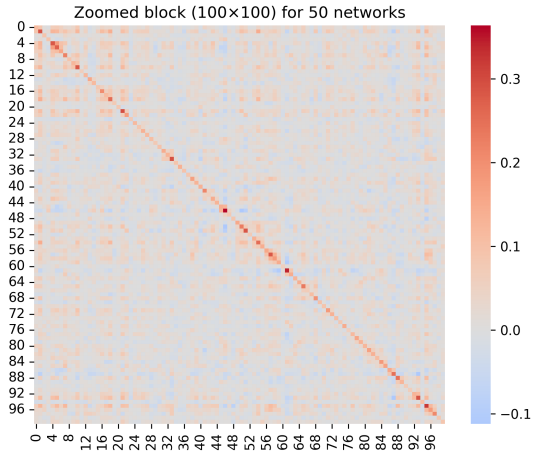
(c) Correlation matrix of architecture 638_400_1024 using 50 networks trained on the CIFAR-10 dataset



(d) Zoomed in snapshot of the correlation matrix of architecture 638_400_1024 using 50 networks trained on the CIFAR-10 dataset

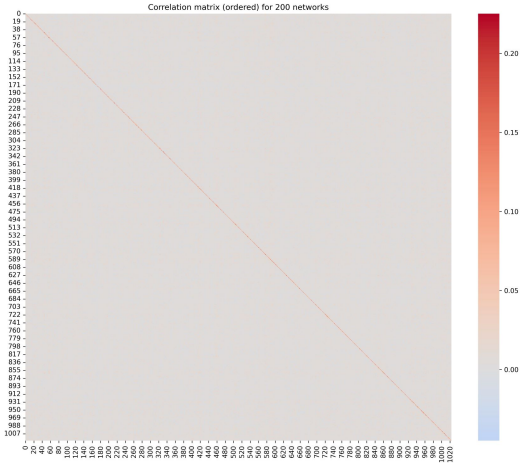


(e) Correlation matrix of architecture 651_508_1024 using 50 networks trained on the CIFAR-10 dataset

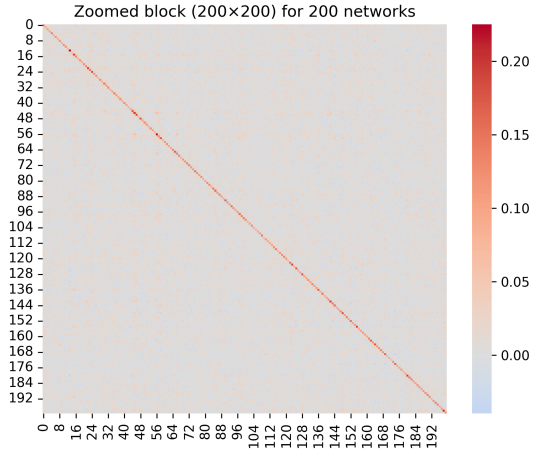


(f) Zoomed in snapshot of the correlation matrix of architecture 651_508_1024 using 50 networks trained on the CIFAR-10 dataset

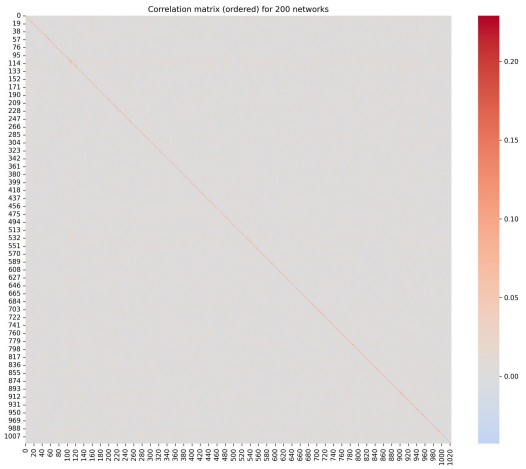
Figure 20



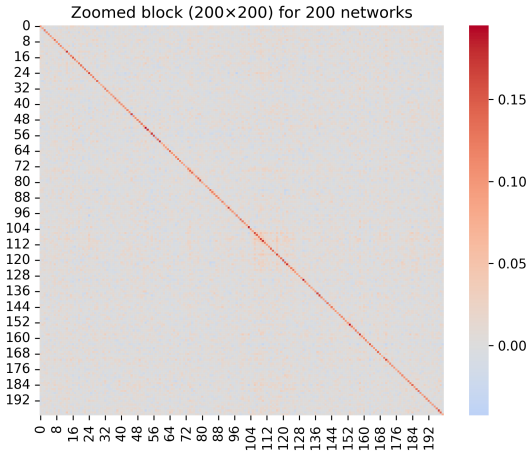
(a) Correlation matrix of architecture 176_1012_1024 using 200 networks trained on the CIFAR-10 dataset



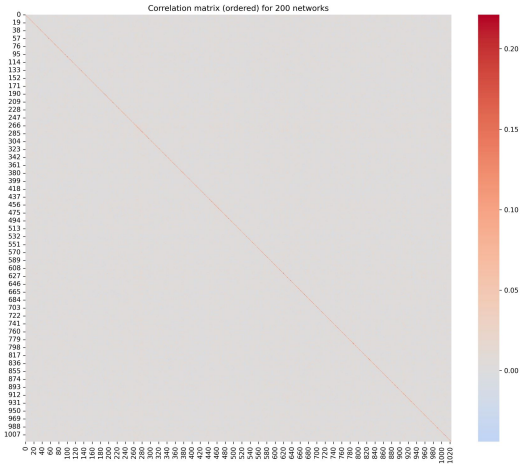
(b) Zoomed in snapshot of the correlation matrix of architecture 176_1012_1024 using 200 networks trained on the CIFAR-10 dataset



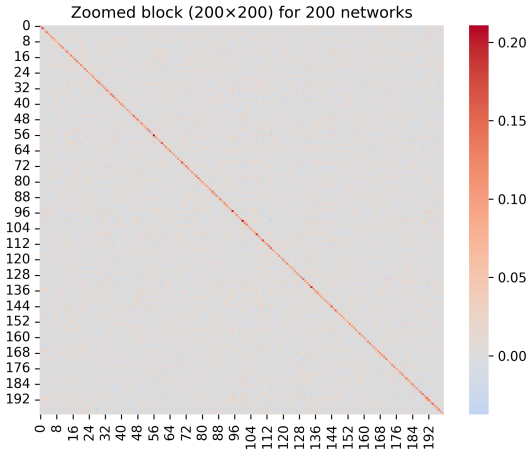
(c) Correlation matrix of architecture 638_400_1024 using 200 networks trained on the CIFAR-10 dataset



(d) Zoomed in snapshot of the correlation matrix of architecture 638_400_1024 using 200 networks trained on the CIFAR-10 dataset



(e) Correlation matrix of architecture 651_508_1024 using 200 networks trained on the CIFAR-10 dataset



(f) Zoomed in snapshot of the correlation matrix of architecture 651_508_1024 using 200 networks trained on the CIFAR-10 dataset

Figure 21