Natural Computing Project Report

Péter Horváth, Sjoerd Crooijmans, Nolan Cardozo

1 INTRODUCTION

Ensemble learning is a process in which multiple weak learners are combined strategically to boost the performance on a machine learning task. These methods have been deployed in various domains and are widely seen to be the top performing models in machine learning challenges, for instance on Kaggle. The primary categories of ensemble methods include bagging, boosting and stacking. These methods aim to find a balance between the bias and variance trade off that is encountered in various machine learning tasks. There has been a lot of research and ensemble techniques developed that work well with structured data and with traditional machine learning algorithms. However, although the recent trend shows that neural networks produce state of the art results on several machine learning tasks, there is very limited research conducted on building ensemble methods with them.

Additionally, most of the research done in this field describes the process of using different network architectures to build an ensemble. For instance, building an ensemble by combining a VGG network and a Resnet network. However, due to the stochasticity in the process of training a nueral network, would it be possible to also train the same network multiple times and combine them to get a strong learner? In this project, we aim to study the effect of using several techniques to build an effective ensemble using a single neural network architecture. For the scope of this study, we decided to use three primary techniques: Bagging, weight initialization and data augmentation. We limit our study to image classification tasks in order to make a more robust comparison. We employ both hard voting and soft voting to combine the base learners.

This narrows our study down to the following research question: "Comparing techniques to create different models from a single neural network architecture to create an effective ensemble." This can further broken down in three sub questions:

(1) Can the ensembles of the methods described in our study boost the performance as compared to a single model?
(2) Is hard voting a better choice than soft voting for combining models?
(3) Is the performance of the ensembles created using the techniques specified significantly better than ensembling the same network without the techniques applied?

2 RELATED WORK

2.1 Bagging

Bagging was originally proposed in [3] by Breiman. Bagging works by retraining the classifier on replicate datasets. These replicate datasets are created by drawing randomly ,but with replacement, from the original dataset. The size of the drawn dataset is equal to the size of the original dataset. Breiman concluded using different experiments that bagging could significantly increase the accuracy of a predictor. Nevertheless these tests were performed using other predictors that neural networks. However, Breiman mentions that bagging will work well for all unstable classifiers. An unstable classifier is a classifier of which the predictor can change largely with a small difference in the training set. Breiman mentions in [4] that neural networks are unstable. The original paper of Breiman has another interesting section which is useful for our research. Namely, the section that discusses how many bootstrapping replicates are needed. Nevertheless, he does not give a concrete answer to this question.

2.2 Weight initialization

A neural network can be often seen as a function that contains learnable parameters that help map the input non linearly to the output. These parameters are also known as weights. Over the years, several methods have been developed on how to choose the initial values of the weights for the network to work optimally as discussed in [16] [7] [11]. Methods proposed varied from assigning constant weights such as zeros or ones to more complex methods that sample random weights from complex probability distributions. In 2010, Glorot et. al. [8] proposed the weight initialisation method which is widely popular as the Xavier initialisation method and it is currently being used as a default weight initialisation method in most deep learning frameworks.

Additionally, Chen et. al. [9] in their paper discusses the the possibility of an ensemble of the same network trained multiple times. They also state that due to non-convexity, networks with same structure but different initialization and training vary a lot. However, they do not explore the possibility of building ensembles using different weight initialisation methods.

2.3 Data augmentation

Data augmentation is a popular and effective method to improve (not only) image classification accuracy. It allows us to generate more data in various ways which help the neural network to achieve better generalization and reduce the threat of overfitting. There are many ways to apply data augmentation for image classification tasks, such as affine transformations, generative adversarial networks, etc. [13] [14] [15]. However, choosing the right type of augmentation might not be an easy task because it is not immediately obvious if a transformation is a label preserving transformation in every case. In [17] they apply shifting on the MNIST dataset which gives better results than the baseline. Nevertheless, as they state in the paper, the magnitude of shifting has to be chosen carefully otherwise some digits might become unrecognizable. In general, choosing an appropriate transformation and determining its magnitude are hard tasks. To that end, AutoAgment was proposed in [5] to find the optimal transformation and its optimal magnitude. This algorithm was tested on the CIFAR-10 dataset yielding state-of-the-art results. Recently, a new data augmentation technique, called Random Erasing, was proposed in [19] which was tested on several benchmark datasets including the Fashion-MNIST and the CIFAR-10 datasets. The technique erases a random rectangle portion of the input by replacing the pixel values in this region with random

values [19]. In conclusion, random erasing might be beneficial to reduce overfitting and improve classification performance.

3 TESTING PROCEDURE

In this section, we discuss the design choices for our experimentation.

3.1 Datasets

As stated before, in this project we limit our study to classification tasks. We decided to use three largely popular image classification datasets for our experiments namely: MNIST [6] , MNIST-fashion [18] and CIFAR-10 [10].

The MNIST dataset contains 70000 grayscale images of handwritten digits between 0 to 9. The dataset is pre split into a training set of 50000 images and a test set of 10000 images. The MNIST-fashion dataset, similarly to the MNIST dataset, consists of the same number of training and test samples. However, in this case the contents are grayscale images of 10 articles of clothing like t-shirts, trousers , bags etc. . Lastly, we also use the CIFAR-10 dataset which consists of 60000 colored images that contain 10 classes which can be narrowed down into 2 categories namely: animals(birds,cats, dogs etc.) and modes of transport(trucks,ships,airoplane etc.). We used a total of 50000 images for training and 10000 as the test set.

3.2 Architectures

As mentioned before, the goal of our study was to build effective ensembles using a single neural network. However, we figured out that it is difficult to build a single network structure for all three tasks. For instance, the CIFAR- 10 dataset contains colored images and is generally known to be a tougher problem than the MNIST and MNIST-Fashion datasets. Hence, we decided to use 2 networks, one for the MNIST and MNIST-fashion datasets and one for the CIFAR-10 dataset. It is important to note here that these design choices do not add bias to the experiment since our goal is compare techniques within datasets and not across datasets. The architectures of these two networks can be found in Appendix A B.

Additionally, its important to note that the experiments are not intended to achieve state of the art results but to study the performance boost that the ensembles using the specified techniques give over a single model. Hence, we do not use sophisticated deep networks that yield very high accuracy, but shallower networks that have a decent performance and thus the performance improvement can be studied in a better and well constrained way. For example for the MNIST dataset, it is possible to create a network that achieves an accuracy of over 99% [12], but it is very difficult to improve on this score using ensembling.

3.3 Evaluation metrics

3.3.1 ***Accuracy:*** Since the task is to perform classification we use the accuracy metric. The networks were combined using both hard and soft voting and the resulting classification accuracy was computed. In the hard voting method, we combine the models by using majority voting based on the hard labels outputted by the models. Soft voting on the other hand has a slightly different approach. In soft voting, we do not use the hard labels that the models output but the average of the classification probability or confidence of an image belonging to a particular class.

3.3.2 ***Correlation:*** To measure the correlation between the individual models, the Pearson correlation between the predictions of the networks was used. The predicted class labels were binarized according to whether the given prediction was correct or incorrect. Therefore, the correlation was measured on the correctness of the predictions ignoring the differences between wrong predictions. The correlation of an ensemble is defined as the average correlation between all the pairs of individual models in the ensemble.

3.4 Test for statistical significance

We used the paired t-test to determine if the ensembles created by our techniques had a statistically significant improvement in performance over the base models or in comparison with other techniques.

3.5 Experiment

As stated in section 3.1, the datasets were pre split into training and test sets. We further break down the training set into a training and validation set with a 80%-20% ratio. The training set was used to train the network and the validation set was used to evaluate the performance of the training at each epoch. Finally, the models were tested on the pre defined held out test set. We used the same hyperparameters for all the techniques for a fair and reasonable comparison. We employed the Adam optimizer as our optimization algorithm with a learning rate of 0.0001. Since all the tasks are multi label classification tasks, we used the categorical crossentropy as our loss function. We used a batch size of 128 samples and initiated early stopping based on the validation loss with patience of 3 to avoid overfitting.

For each technique, we trained the networks and combined the outputs in an ensemble as specified in section 4 for a total of 10 times. This is done to make sure that the performance is reproducible, representative and is not a one off random value. Along with the accuracy of the weak learners and the final ensemble, we also derive a correlation matrix using each of the weak learners to understand the amount of variance they have to offer so as to be combined in an effective ensemble. An example of the correlation matrix can be seen in figure 3.5. Finally, we perform a paired t-test using the data for all the 10 runs to compare and understand if the difference between the ensemble created by our techniques are significantly different compared to the base model or other techniques.

Table 1: Example correlation matrix

	Zero	Ones	Random Normal	Random Uniform	Identity	Othogonal	Glorot Normal	Glorot Uniform
Zero	-	0.62	0.65	0.61	0.64	0.63	0.64	0.68
Ones	-	-	0.62	0.61	0.63	0.58	0.61	0.63
Random Normal	-	-	-	0.61	0.62	0.61	0.60	0.61
Random Uniform	-	-	-	-	0.66	0.60	0.63	0.61
Identity	-	-	-	-	-	0.61	0.68	0.66
Othogonal	-	-	-	-	-	-	0.63	0.62
Glorot Normal	-	-	-	-	-	-	-	0.63

4 METHODS

4.1 Bagging

Bagging is implemented as described in the original paper of Breiman (1996). The most important takeaway is that the original dataset is sampled using bootstrap sampling and that the replicate set has the same size as the original dataset. However, there is one deviation from the implementation in the paper. Breiman could not specify the number of models needed in an ensemble. Therefore we keep adding models to the ensemble, till the increase in accuracy is less than 0.01. The accuracy is determined by combining the predictions of the models using voting and checking it against the real labels. Using an increased number of models till the difference is small enough seemed more reliable then just picking a number of models. This is based on table 10 of the paper, which clearly shows a point of diminishing returns when the number of models keeps increasing. Therefore, we think that this is a reliable way to determine the number of models.

4.2 Weight initialization

In total, we explored the possibility of using 8 different weight initialization methods which are available in the keras API. We chose a mix of methods that can be largely put in 3 categories, namely: constant (zeros and ones), matrices based (identity and orthogonal), and distribution based (normal random, random uniform , glorot normal and glorot uniform). We train the network each time using one of the weight initialization methods and combine all the methods to form an ensemble.

4.3 Data augmentation

We utilised three affine transformations, namely rotation, horizontal shifting and vertical shifting. We tried to use transformations which make sense for every dataset, thus flipping was excluded from the transformations because flipping for the MNIST dataset would not produce label-preserving transformations (e.g. digits 6 and 9). Each model in the ensemble uses one of the transformations which means that there are three models in the ensemble. While training a neural network, every image has a 20% chance to be transformed every time the image is fed to the network. The rotation range in degrees is the [-10,10] interval so the model which uses rotation as data augmentation will apply a random rotation in this range with 0.2 probability to an image . The magnitude of horizontal shifting is either -3 or 3 pixels while it is either -2 or 2 for the vertical shifting. Similarly to rotation, shiftings are applied with 0.2 probability to an image. The probability (0.2) of applying the transformations was intentionally set low, we didn't want to apply a stronger augmentation process.

4.4 Baseline ensemble

Multiple networks trained without any of the techniques applied and combined in an ensemble. The number of models is determined in the same way as in the bagging technique.

4.5 Baseline single model

Network trained without any of the techniques and not combined in an ensemble.

5 RESULTS

Figure 5 (in the Appendix) shows the results of the experiments on the MNIST dataset. As can be seen, all the methods performed better than the single model in both hard and soft voting. The performance between the ensembles is the same except for data augmentation which performs slightly worse than the baseline ensemble, but this is only significant in the case of hard voting. In the case of MNIST Fashion the single model performed again less good than the ensembles as can be seen in Figure 6 (in the Appendix). However, in this case the weight initialization technique out performed all the other techniques and the baseline ensemble in the case of hard voting. Nevertheless, this performance benefit mostly disappeared when soft voting was used. All the other methods performed the same. Lastly, the CIFAR-10 dataset, in the Appendix in Figure 7, whereby both data augmentation and bagging work significantly worse then the baseline ensemble. On the other hand weight initialisation performs the same as the baseline ensemble. Similar to the other datasets, the single model performs again less well then the ensembles. Additionally, in all experiments soft voting works better than hard voting as it is shown in Figure 1 and in Table 2.

Table 2: Experiment results

Model	Dataset	Accuracy-soft	Accuracy-hard
baseline model	CIFAR-10	-	65,3%
plain ensemble	CIFAR-10	72,4%	71,8
bagging	CIFAR-10	72,4%	68,9%
weight-init	CIFAR-10	72,5%	71,5%
data augmentation	CIFAR-10	71,3%	69,7%
baseline model	MNIST	-	76,2%
plain ensemble	MNIST	78,8%	78,4%
bagging	MNIST	79,0%	78,5%
weight-init	MNIST	79,0%	78,4%
data augmentation	MNIST	78,5%	77,8%
baseline model	Fashion-MNIST	-	76,4%
plain ensemble	Fashion-MNIST	77,7%	77,3%
bagging	Fashion-MNIST	77,5%	77,2%
weight-init	Fashion-MNIST	77,8%	77,6%
data augmentation	Fashion-MNIST	77,5%	77,1%

6 DISCUSSION

6.1 Baseline ensemble

As discussed earlier, the baseline ensemble performed better than the single model. The difference in resulting networks is already enough to create an effective ensemble as shown by Chen et. al [9]. As can be seen in Figure 3 a portion of this variance is caused by the initialization of the weights, because the correlation between models increases when all the initial weights are the same. However, the resulting models are still different enough to create an effective ensemble, except for the MNIST Fashion dataset. In the other two cases the ensemble with fixed weights performed in between the single model and the baseline ensemble as can be seen in Figure 2. In the end, we can conclude that both the randomization caused by training and weight initialization can help to create an effective ensemble.

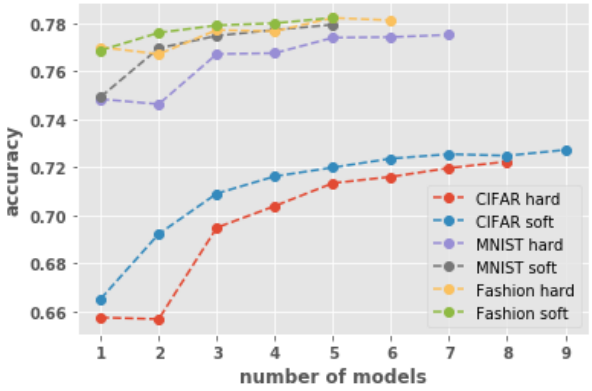

Figure 1: Number of models versus soft and average validation accuracies of the plain ensembles

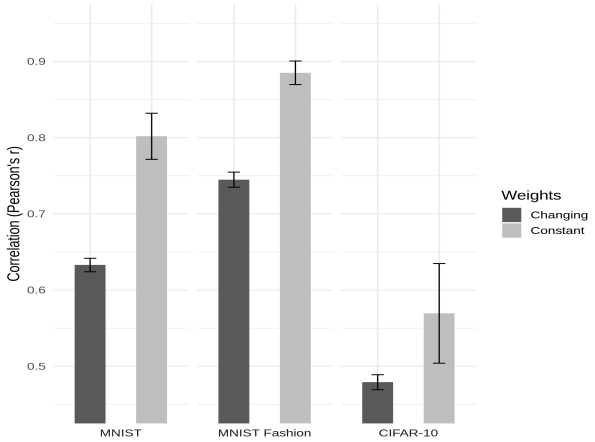

Figure 3: Comparison between the correlation of models in the baseline ensemble trained with constant initial weights and random initial weights. The results for the constant weight experiment were gathered by rerunning the baseline ensemble (soft voting only) and setting the initial weights before training.

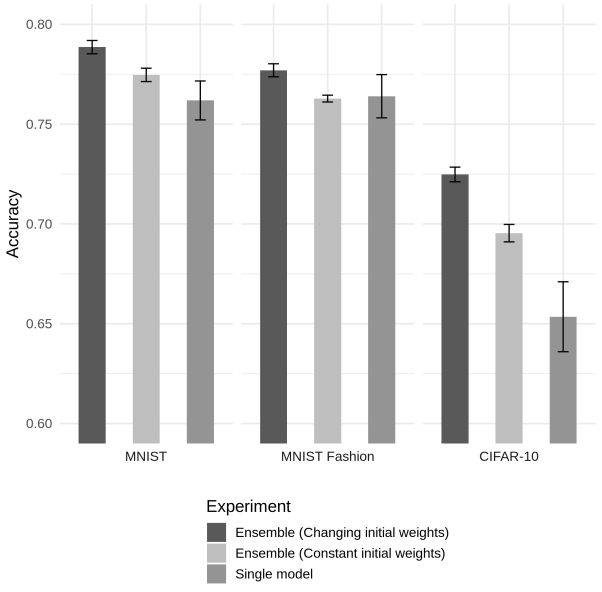

Figure 2: Comparison between the accuracies of the baseline ensemble whereby all the models were initialized with the same constant weights and models whereby the initial weights were randomized. The results for the constant weight experiment were gathered by rerunning the baseline ensemble (soft voting only) and setting the initial weights before training.

6.2 Bagging

As described earlier in the report, bagging performs equally to the baseline and most of the other methods in the case of MNIST and MNIST Fashion. This is caused by the fact that the correlation between the models created using the baseline and bagging are not significantly different. Supplementary, the performance of the individual models was identical. Therefore, we can conclude that the bagging, in its current form, does not create enough difference in the training set to train the model in another way. On the other hand bagging applied to CIFAR-10 performed worse than the baseline ensemble. This was primarily caused by a decrease of 3.7% in the performance of individual models. Nevertheless, bagging causes a 3,7% decrease in correlation in the case of soft voting and a 2,9% decrease in the case of hard voting. This could be caused by the increase in wrong predictions. A new experiment was conducted to check if the decrease in performance was caused by a too complex network for the new less diverse training set. The simpler networks consisted of one dense layer, which worked reasonably well, and both the baseline and bagging experiment were redone. In this case the experiment was only done with soft voting. The performance of the individual models with bagging enabled were still worse than the baseline models. However, the decrease was in this case only 1% instead of the 3.7% seen earlier. At the same time, the correlation between the individual models decreased similarly as before with 3.7%. Anyhow, bagging with the simple model still performed significantly (0.6%) worse than the baseline ensemble. Consequently, we can conclude that bagging does not work well in the case of CIFAR-10 due to the decrease in performance caused by the less diverse training set.

6.3 Weight initialization

As stated earlier in section 2.2, due to non-convexity, networks with same structure but different initialization and training vary a lot. This made us initially believe that creating an ensemble of with different weight initialization methods would yield a better ensemble. After running the experiment, we indeed saw that the ensemble of different weight initialization methods performed significantly better than the single model.

Although, when we tried to inspect where the variance and performance boost comes from, we saw that the weight initialization ensemble performed equally as good as ensembling the same network multiple times with the same initialization. This proves that the major variance or performance boost in the weight initialization ensemble comes from the stochasticity within the network structure itself while training the network with optimizers such as stochastic gradient descent or Adam rather than the methods in which the weights are assigned to start with.

6.4 Data augmentation

Data augmentation did not live up to the expectations. There could be many reasons to this. Firstly, vertical and horizontal shifting performed worse than rotation in most runs. If we look at the datasets, shifting might make some images ambiguous. For example, in the Fashion-MNIST dataset if some images containing a T-shirt, a shirt or a sweatshirt were shifted upwards, then the collar of the clothing would be cropped. This actually might make it harder for the network to learn. Secondly, the images in Fashion-MNIST might not benefit that much from rotation either because every object in every image is centered including the images in the test set. In the case of MNIST, rotation could be beneficial but this depends on the direction of the rotation which is random in our experiment. More importantly, the data augmentation ensemble uses only 3 models in each ensemble while the number of models in each plain ensemble is always higher than 3. This might be the major reason for the plain ensemble achieving higher accuracy, because on an individual level the models which use augmentation perform better than the baseline.

Table 3: Experiment results with rotation ensemble

Model	Average test set accuracy
CIFAR-10 soft	72,4%
CIFAR-10 hard	71,8%
CIFAR-10 rotation soft	**73**, 4%
CIFAR-10 rotation hard	**72**, 4%
MNIST soft	78,8%
MNIST hard	78,4%
MNIST rotation soft	78,4%
MNIST rotation hard	77,8%
Fashion-MNIST soft	77,6%
Fashion-MNIST hard	77,3%
Fashion-MNIST rotation soft	77,3%
Fashion-MNIST rotation hard	76,9%

To support these claims, in a new experiment an ensemble was created for every dataset in which every model uses the rotation augmentation. The number of models in the ensemble was determined by the method implemented in the plain ensemble. The results are depicted in Table 3.

On the Fashion-MNIST dataset, the ensemble of rotation models performs better than the ensemble of the three augmentations, but still falls short to the plain ensemble by 0.4% (hard voting) and by 0.3% (soft voting).

On the MNIST dataset, the rotation ensemble falls short to the plain ensemble by 0.6% (hard voting) and by 0.4% (soft voting). Lastly, rotation is beneficial for the CIFAR-10 dataset. The rotation ensemble is superior to the baseline ensemble by 1% (soft voting) and by 0.6% (hard voting).

In all cases, the differences were statistically significant. These results might not be surprising because rotation should help with CIFAR-10 dataset. In contrast to the MNIST and Fashion-MNIST datasets, the CIFAR-10 dataset contains many images in which the objects are shown in many different angles thus rotation might be beneficial for generalization.

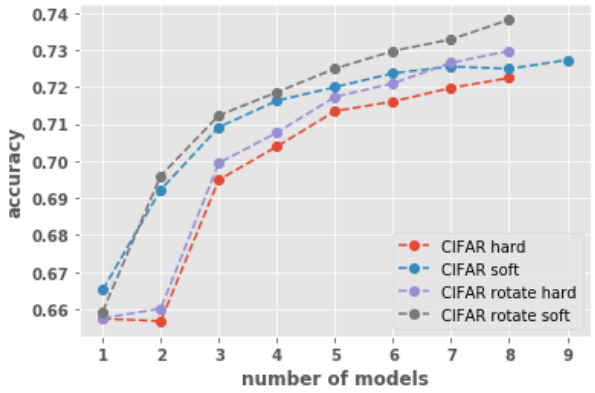

Figure 4: Number of models versus soft and hard average validation accuracies of the plain CIFAR ensembles and rotation ensembles

Figure 4 shows that the number of models in the ensembles has a large impact on the accuracy on the CIFAR dataset. The corresponding figures for the MNIST (Figure 8) and Fashion-MNIST (Figure 9) datasets can be found in the Appendix. In conclusion, the different augmentations offer little to no benefit for increasing the accuracy in contrast to number of models in the ensemble. However, augmentation has the potential to increase accuracy if the augmentation used is carefully chosen.

6.5 General

Weights were reused between runs with hard and soft to save computation time. However, there was a bug in the code and as a result the validation and training set between the experiment with hard and soft voting differentiated. Therefore, the number of models used in bagging and the baseline ensemble is (partly) calculated on the training set and if an extra model was required this was trained on another training set then the other models. This did only effect experiments with hard voting in combination with bagging or the baseline ensemble. We do not believe that this did

influence our results and our conclusion. First of all, the evaluation set, where the results were calculated on, is still separate from the training and validation set. Secondly, we redid the experiment for CIFAR-10 and we saw only a very small and negligible difference between the results. Unfortunately, there was not enough time to rerun the other experiments. On top of that the experiments with soft voting resulted in the most interesting data, and those were not affected by this bug.

7 CONCLUSION

In general, the techniques do not boost the accuracy. Ensembles with more models, however, are effective. Some studies have shown that the optimal number of models in an ensemble could be around the number of class labels [1][2]. However, these studies come to this conclusion using independent classifiers. Still, this might indicate that ensembles with higher number of models (closer to the number of class labels, 10 in every case) are superior to those with less models even if the models are correlated. Secondly, the applied transformations for the data augmentation should be chosen carefully or should not be used at all. Simpler problems do not benefit much from data augmentation even if the technique used makes sense for the problem. More challenging problems can benefit greatly from augmentation but the rule of careful choosing of techniques applies here as well.

REFERENCES

[1] Hamed Bonab and Fazli Can. 2017. Less is more: a comprehensive framework for the number of components of ensemble classifiers. *arXiv preprint arXiv:1709.02925* (2017).

[2] Hamed R Bonab and Fazli Can. 2016. A theoretical framework on the ideal number of classifiers for online ensembles in data streams. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2053–2056.

[3] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[4] Leo Breiman et al. 1996. Heuristics of instability and stabilization in model selection. *The annals of statistics* 24, 6 (1996), 2350–2383.

[5] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. 2018. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501* (2018).

[6] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[7] Mercedes Fernández-Redondo and Carlos Hernández-Espinosa. 2001. Weight initialization methods for multilayer feedforward.. In *ESANN*. 119–124.

[8] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[9] Cheng Ju, Aurélien Bibaut, and Mark van der Laan. 2018. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *Journal of Applied Statistics* 45, 15 (2018), 2800–2818.

[10] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[11] Siddharth Krishna Kumar. 2017. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863* (2017).

[12] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. [n. d.]. The mnist database. http://yann.lecun.com/exdb/mnist/

[13] Luis Perez and Jason Wang. 2017. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621* (2017).

[14] Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data* 6, 1 (2019), 60.

[15] Luke Taylor and Geoff Nitschke. 2017. Improving deep learning using generic data augmentation. *arXiv preprint arXiv:1708.06020* (2017).

[16] Georg Thimm and Emile Fiesler. 1995. Neural network initialization. In *International Workshop on Artificial Neural Networks*. Springer, 535–542.

[17] Sebastien C Wong, Adam Gatt, Victor Stamatescu, and Mark D McDonnell. 2016. Understanding data augmentation for classification: when to warp?. In *2016 international conference on digital image computing: techniques and applications (DICTA)*. IEEE, 1–6.

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

[19] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2020. Random Erasing Data Augmentation.. In *AAAI*. 13001–13008.

8 CONTRIBUTIONS

8.1 Nolan:

- Implemented and experimented with the the weight initialization techniques
- Baseline implementation of MNIST-fashion dataset
- Actively took part in team meetings and discussions
- Wrote a part of the report

8.2 Peter:

- Created the network architectures
- Implementation and experimentation with data augmentation techniques (rotation, horizontal and vertical shifting)
- Implementation of baseline model for the CIFAR-10 dataset
- Implementation and testing of rotation ensemble
- Actively took part in team meetings and discussions
- Wrote a part of the report

8.3 Sjoerd:

- Implemented and experimented with the the bagging technique
- Implemented and experimented with the baseline ensemble
- Baseline implementation of MNIST dataset
- Actively took part in team meetings and discussions
- Wrote a part of the report

9 APPENDIX

A CODE

The code can be found on: https://github.com/SjoerdC/NaturalComputing/tree/mas

B FIGURES

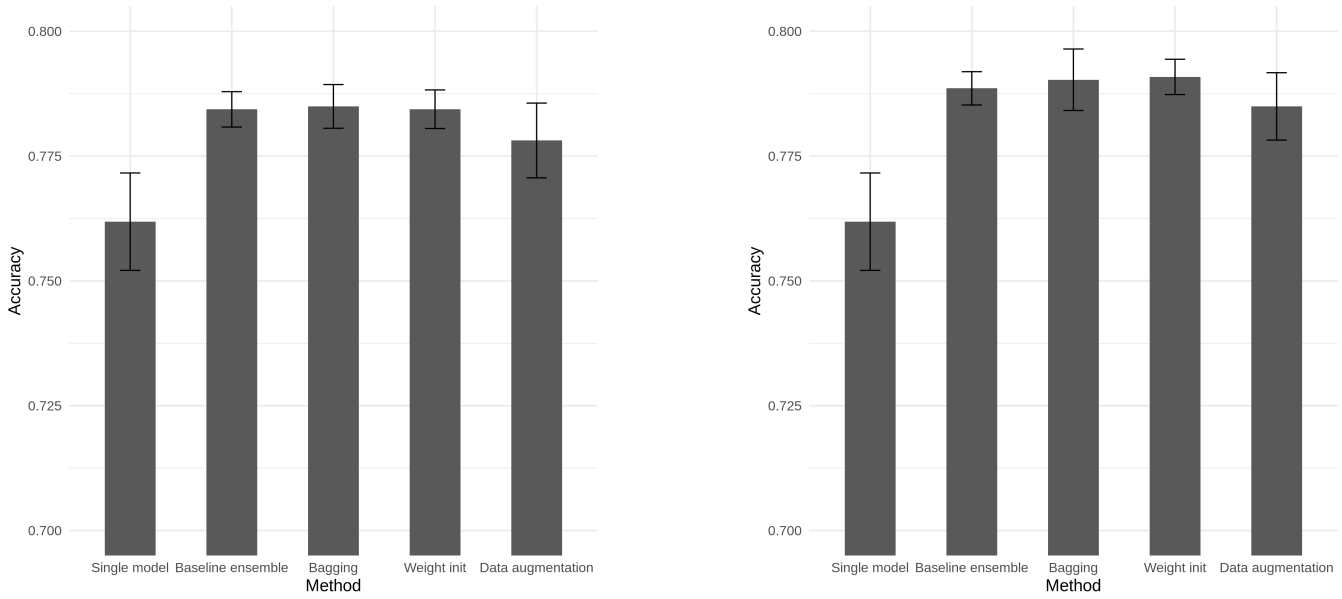

Figure 5: Accuracy of the methods applied to the MNIST dataset using hard (left) and soft (right) voting, the barplots represent the standard deviation.

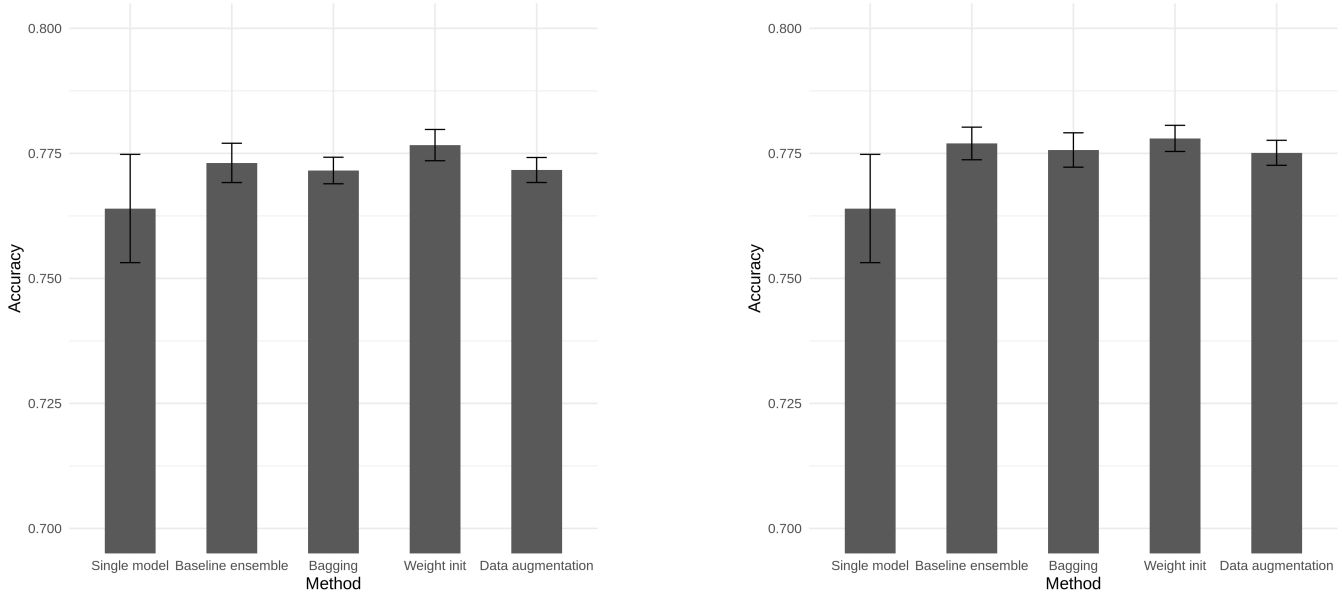

Figure 6: Accuracy of the methods applied to the Fashion-MNIST dataset using hard (left) and soft (right) voting, the barplots represent the standard deviation.

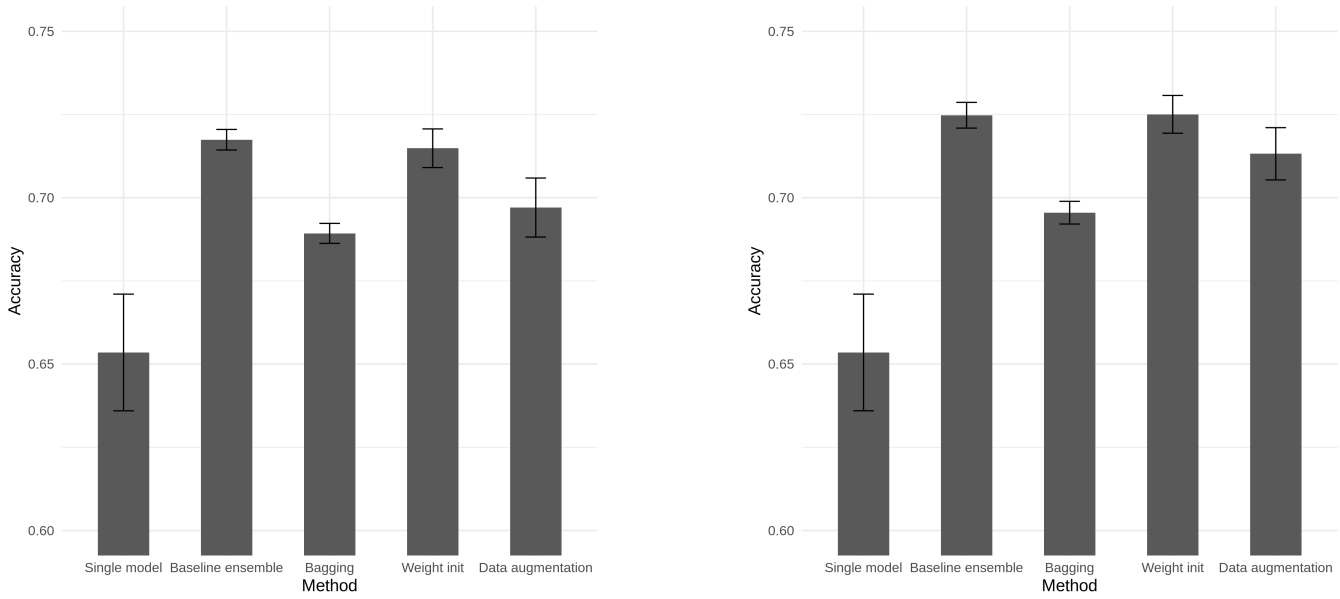

Figure 7: Accuracy of the methods applied to the CIFAR-10 dataset using hard (left) and soft (right) voting, the barplots represent the standard deviation.

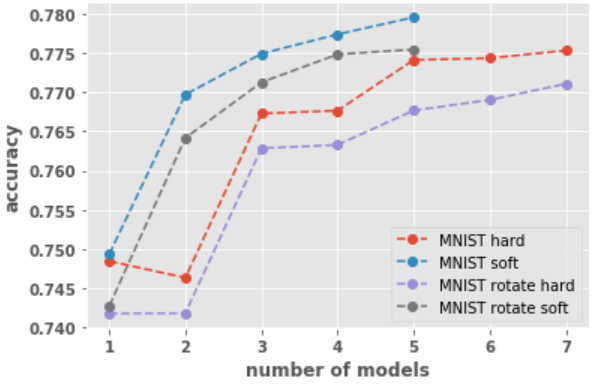

Figure 8: Number of models versus soft and hard average validation accuracies of the plain MNIST ensembles and rotation ensembles

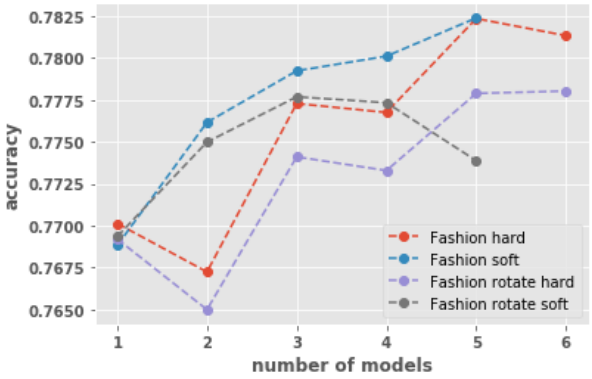

Figure 9: Number of models versus soft and hard average validation accuracies of the plain Fashion-MNIST ensembles and rotation ensembles