Deep Learning

Project 1

Image classification with Convolutional Neural Networks

Authors: Hubert Ruczyński (BSc), Patryk Tomaszewski (BSc)

28th March 2023

Faculty of Mathematics and Information Science

WARSAW UNIVERSITY OF TECHNOLOGY

Contents

1 Abstract

Convolutional Neural Networks (CNNs) are a type of artificial neural network that has gained popularity in the field of deep learning for their exceptional performance in image and speech recognition tasks. CNNs are characterised by their ability to learn spatial hierarchies of features from input data, which makes them particularly effective for tasks that involve identifying patterns in images or sounds. There is plenty of publicly available neural network (NN) architectures, as well as pretrained weights for them, which can help with the training process. Such solutions however still have to be tuned to achieve reasonable results, and in this paper, we conduct small-scale research on training parameters, augmentation methods, and ensemble voting strategies that enhance the quality of the CNNs predictions.

Keywords: machine learning, convolutional neural networks, CIFAR-10, data augmentation,

2 Study description

The main goal of this study is to examine the behaviour of different Convolutional Neural Network architectures depending on the selected hyper-parameters, augmentation methods, as well as the impact of the ensemble voting on our neural classifiers. To evaluate the CNNs performance we will use a CIFAR-10 dataset ([HP19]), which is a well-known benchmark for multi-label classification.

2.1 CIFAR-10

CIFAR-10 ([HP19]) is a widely used benchmark dataset that consists of 60,000 images that are divided into 50,000 training images and 10,000 test images. The images are all 32x32 in size and are in colour, with 3 channels. The dataset is labelled over 10 categories, which are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

The dataset was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton in 2009, and has been used extensively in machine learning and computer vision research. The dataset is often used as a standard benchmark for image classification tasks, as it is relatively small and easy to work with, while still being representative of real-world images.

CIFAR-10 is particularly useful for evaluating the performance of convolutional neural networks (CNNs), which are a type of artificial neural network that is particularly effective for tasks

that involve identifying patterns in images or sounds. The relatively small size of the CIFAR-10 dataset makes it possible to train CNNs on the entire dataset using modest computing resources, which is not always possible with larger datasets.

2.2 Experiments

Firstly, we've decided to test four CNNs architectures: self-written custom CNN, pretrained Efficient Net (described in [TL20]), pretrained Mobile Net V3 ([HSC+19]), and randomly initialised Mobile Net V3. We were curious how the custom architecture will behave compared to the more sophisticated sibling. Moreover, we wanted to compare the Efficient Net and the Mobile Net, as the second one was created for mobile devices, like smartphones, which results in the smaller architecture of the network. Lastly, the pretrained and random initialisation of the Mobile Net will answer the question if we should always choose the pretrained NNs.

During the first experiment we chose 4 universal hyper-parameters:

- learning_rate,
- batch_size,
- dropout_rate,
- l2 regularisation.

The first two of them were related to the training process, and the other two to the regularisation. Initially, we planned to provide 5 values of every parameter, however, due to the computational power limitations, we've provided only three parameters for the batch_size, as our PC couldn't hold more than 128 element batches. During the training process, we were tuning only one parameter at once in the order given in the list above, but after each hyper-parameter tuning we chose the value that provided the best results for training another one.

For the second experiment we've chosen to analyse the influence of data augmentation, in hopes of improving the accuracy of the model ([PW17]). As the base for this experiment, we selected the best-performing architecture and hyper-parameters from the previous runs. We tested 4 different image augmentation methods:

- shifting,
- cropping,
- increasing contrast,
- cutmix ([YHO+19])

as well as a fifth method, which was the combination of all of the previous ones. It is worth noting that shifting, cropping and increasing contrast are augmentations based on transforming a single image at once, while cutmix is a more sophisticated method that merges information from multiple instances to form a new one. The augmentations were applied by randomly selecting 50% of images, transforming them, and then appending the resulting instances to the original dataset.

With the increased researcher's interest in Automated machine learning (AutoML) we've noticed an increased interest in models ensembles, even in the field of deep learning ([BGNK18]). During the last experiment concerning the ensemble methods, we decided to test two simple, and computationally cheap strategies which are hard and soft voting. The main goal of this task was to compare both methods with each other in order to choose which option is the best one. Additionally, we've also made ensembles of the best k% of the models to see if, with the limitation to the best NNs, the ensemble will achieve better results.

3 Theory

In this section we will provide all the basic and necessary knowledge needed for a full understanding of our study.

3.1 Terminology

Def. 1 Machine learning (ML) - Machine learning is a field of artificial intelligence (AI) that involves the use of algorithms and statistical models to enable computer systems to learn from data without being explicitly programmed. The goal of machine learning is to develop systems that can automatically improve their performance on a specific task over time with more data.

Def. 2 Image classification - Image classification is a task in computer vision that involves assigning a label or category to an image based on its content. It is a common application of machine learning and deep learning and is used in a variety of fields such as medical imaging, security, and autonomous vehicles.

Def. 3 Neural networks (NN) - Neural networks are a type of artificial intelligence that is inspired by the structure and function of the human brain. They consist of layers of interconnected nodes that process and transmit information and are capable of learning and making predictions based on patterns in data.

Def. 4 Convolutional neural networks (CNN) - Convolutional Neural Networks (CNNs, introduced in [ON15]) are a type of artificial neural network that is particularly effective for tasks that involve identifying patterns in images or sound. They use convolutional layers and pooling layers to extract and down-sample features, followed by fully connected layers that perform classification or regression tasks.

Def. 5 Pretrained network - A pretrained neural network is a neural network that has been trained on a large dataset, typically on a specific task, and then fine-tuned on a smaller dataset to perform a related task. The pretraining of the neural network allows it to learn features that can be used for other tasks, which can save time and computational resources compared to training a new neural network from scratch.

Def. 6 Data augmentation - Data augmentation is a technique in machine learning that involves generating new training data by applying a variety of transformations to existing data. This can help to increase the size of the training dataset and improve the performance of machine learning models by reducing overfitting and increasing generalisation. Some common data augmentation techniques include flipping, rotating, and scaling images, as well as adding noise or distortions to the data.

Def. 7 Ensemble voting - Ensemble voting is a technique in machine learning where multiple models are trained and their predictions are combined to produce a final prediction. This can result in improved accuracy and robustness compared to using a single model. There are different types of ensemble voting, including majority voting and weighted voting.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) learn spatial hierarchies of features from input data. This means that they are able to identify patterns in images or sounds by processing the data in a way that mimics the way the human brain processes visual and auditory information.

CNNs are composed of several layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers perform convolutions on the input data, which involves sliding a filter over the input and calculating dot products at each position. This process extracts features from the input data by identifying patterns in the local neighbourhoods of the input.

Pooling layers down-sample the output of convolutional layers by reducing the resolution of the

feature maps. This is done to reduce the computational complexity of the network and to make it more robust to variations in the input.

The output of the pooling layers is then passed to fully connected layers, which perform classification or regression tasks. These layers are similar to the layers in a traditional artificial neural network, and they use the features extracted by the convolutional and pooling layers to make predictions about the input data.

During the training process, the weights of the filters in the convolutional layers and the weights of the connections in the fully connected layers are adjusted using back-propagation. This involves calculating the gradient of the loss function with respect to the weights and updating the weights in the direction that minimises the loss.

4 Proposed solutions and outcomes

4.1 Experiment 1

In the first experiment, which was focused on determining the best hyper-parameters for the classification task on the CIFAR-10 dataset, we used four different architectures for the feature extraction part of the model. The classification part of the model remained the same between extractors, consisting of three densely connected layers, the first two using the relu activation function and having respectively 64 and 32 nodes, and the third with the softmax activation function and 10 nodes, which corresponds to the number of classes in the dataset.

For the first feature extractor, we used a simple convolutional network consisting of 3 VGG blocks. For the second, we used Efficient Net in the B0 variant, with initial resizing of the input images to 64x64 for improved performance. We loaded the weights trained on the Image Net dataset ([DDS+09]), as they are provided in the Keras library, and froze them. That means that during the model training, only the classification part of the model was fine-tuned. For the third and fourth architecture, we used Mobile Net in the V3 small variant, with prior image resizing to 128x128. For one of those architectures we loaded the weights trained on Image Net and froze them, the other one was randomly initialised and left trainable.

For the training process itself, we used categorical cross-entropy as the loss function, and an early stopping mechanism, with a patience of 3, a minimum delta of 0.005, and a maximal 100

epochs. We also extracted a validation set consisting of 20% of instances from the training dataset, which remained the same between all runs. Each execution was repeated 5 times to provide more statistically significant results.

We analysed the influence of 4 different hyper-parameters, by training and evaluating each of the aforementioned models on different values of said hyper-parameters, while all the others remained constant. The hyper-parameters, and the values for which we tested, are provided below:

- learning rate - 0.01, 0.05, 0.1, 0.2, 0.5,
- batch size - 64, 96, 128,
- dropout rate - 0, 0.2, 0.4, 0.6, 0.8,
- L2 regularisation - 0, 0.2, 0.5, 0.7, 1.

Note that the dropout rate and regularisation were only applied to the classification part of the model. The experiment was run in the order presented above, with the first hyper-parameter, learning rate, using default values (batch size of 64, dropout rate of 0.5, and no L2 regularisation), and each consequent execution using values from the best-performing run before.
The results for each parameter have been presented in Figures 1 to 4.

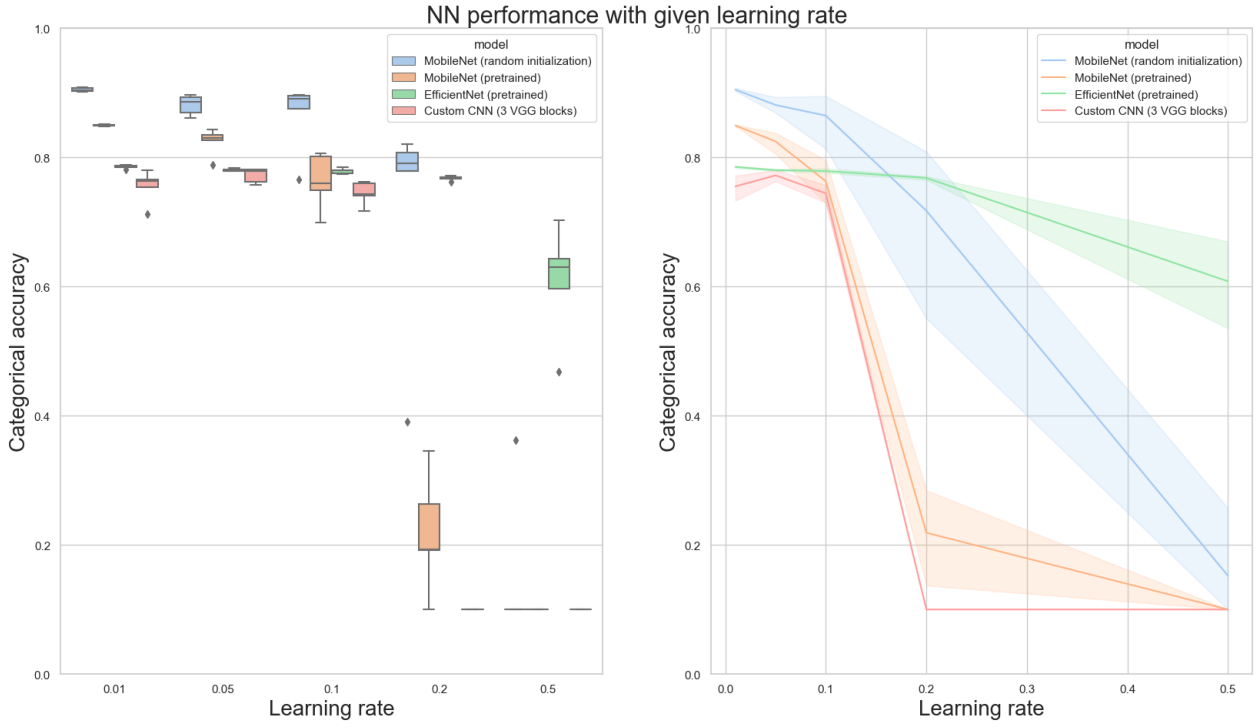

Figure 1: Neural Networks' performance with given learning rate.

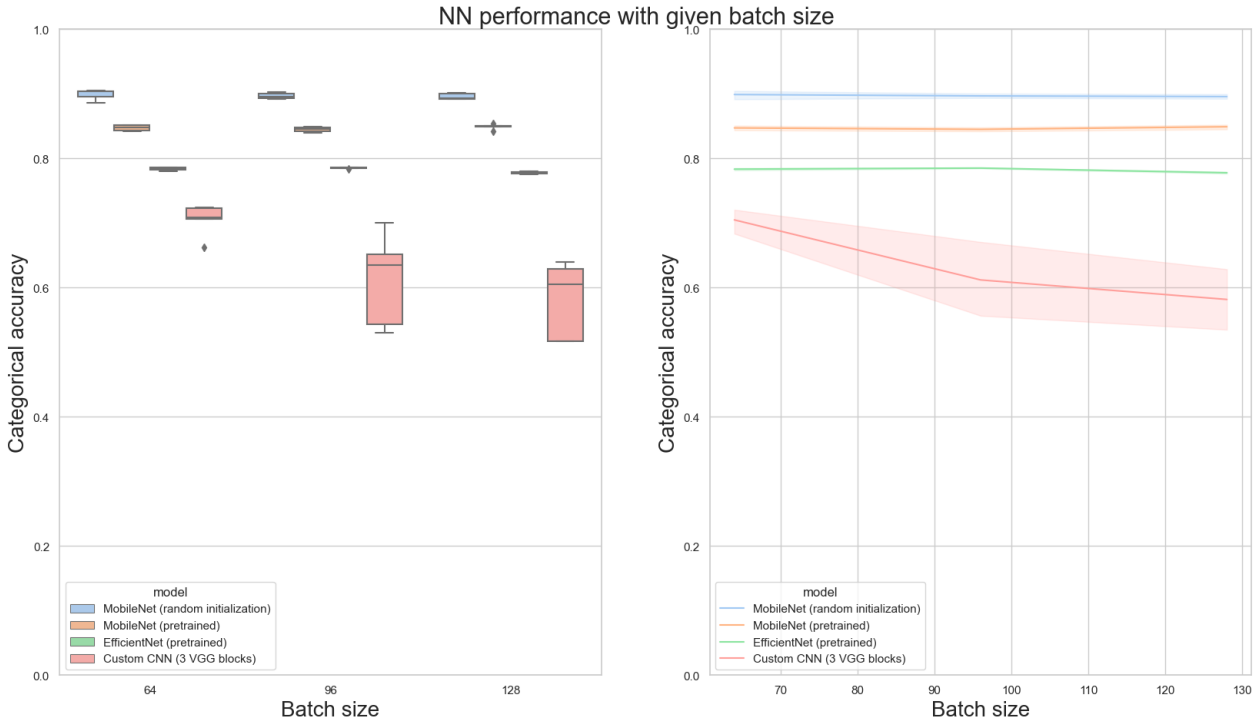

Figure 2: Neural Networks' performance with given batch size.

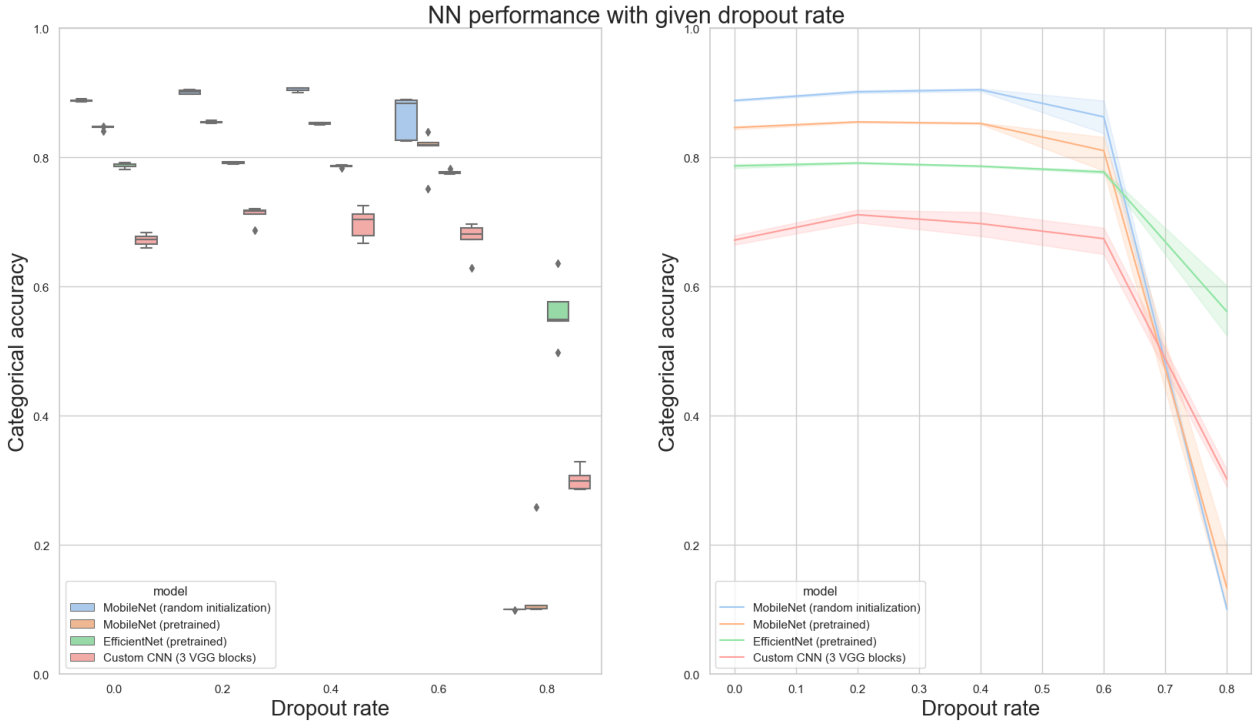

Figure 3: Neural Networks' performance with given dropout rate.

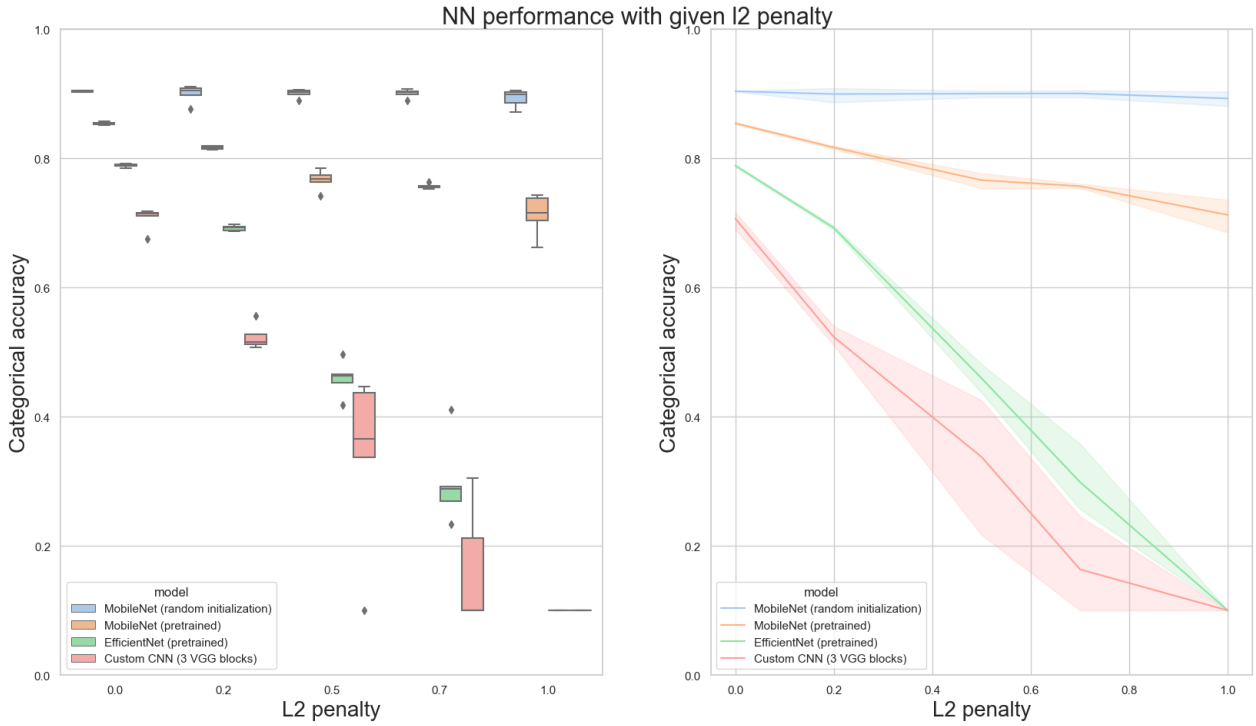

Figure 4: Neural Networks' performance with given l2 penalty.

In Figure 1 we can observe that the lower the learning rate, the better the model performed. The only exception to that would be the increase of accuracy on the custom CNN between 0.01 and 0.05 learning rate. It's also clear to see that a very high learning rate has a severe negative impact on the performance of the model, with three of the architectures dropping to 10% accuracy (which corresponds to the efficiency of random guessing), and Efficient Net reaching around 63% accuracy. This could have been caused by weights explosion, as a higher learning rate increases the risk of overshooting the optimal solution.

Figure 2 suggests that the impact of the batch size is marginal for the model, except for the custom CNN. In this architecture, we can observe a drop in accuracy with the increased batch size. We can also notice that the variance of accuracy is the biggest for this architecture, and it increases even more with the bigger batch size. It is worth noting that while the accuracy of the model remained unchanged, the batch size affected different properties of it, with a high batch size requiring higher memory in order to operate the model, and a lower batch size increasing the training time.

In Figure 3 we can see that a low dropout rate may increase the model performance, while a high one, like 60% or 80%, can reduce it. This is respectable, as the main goal of the dropout

is to increase the generalisation, while a too high of a dropout rate can lead to too few nodes being active in the classification layer for it to perform at all.

Figure 4 shows very interesting results, which suggest that in this task L2 regularisation will only reduce the accuracy. This effect is significant for Efficient Net and Custom CNN while being less severe for pretrained Mobile Net, and practically absent for the randomly initialised Mobile Net. This is surprising, as usually L2 regularisation tends to improve the performance of the model, however, it could be argued that this is caused by the interaction of both regularisation in form of the L2 penalty as well as a dropout.

In all of the above figures, a clear hierarchy between the models can be established, with the randomly initialised Mobile Net being the best-performing architecture, followed by pretrained Mobile Net, pretrained Efficient Net and the custom CNN. While we would expect a pretrained model to perform better than a randomly initialised one, this behaviour could be explained by the nature of the Mobile Net architecture. As it was designed to be lightweight and quick learning, the benefits of prolonged training on the Image Net dataset may not be as significant in comparison to directly finetuning to the desired dataset. This could have also been influenced by the fact that, in the case of the pretrained architecture, we have frozen the feature extraction part of the model.

4.2 Experiment 2

For the second experiment, we focused on the influence of performing data augmentation on the training dataset. We used the best-performing configuration from the previous experiment, which was the randomly initialised Mobile Net with a learning rate of 0.01, batch size of 64, a dropout rate of 0.2, and no L2 regularisation. We repeated the training and evaluation 5 times for each of the following methods:

- shift - randomly shifting the images by up to 20% in each direction and filling in the empty pixels with a wrapped version of the image,
- crop - randomly replacing the border pixels with black, up to 20% of image width, with each axis being treated separately,
- contrast - randomly increasing the image contrast by up to 200%,
- cutmix - applying cutmix,
- all - applying all of the aforementioned sequentially, by firstly shifting, then increasing contrasts, afterwards applying cutmix, and finally replacing the border pixels with black.

The augmented dataset was created by first selecting 50% of the images, then applying one of the methods and combining the result with the original dataset. Each augmentation was executed independently for each of the runs. The results have been presented in Figure 5.

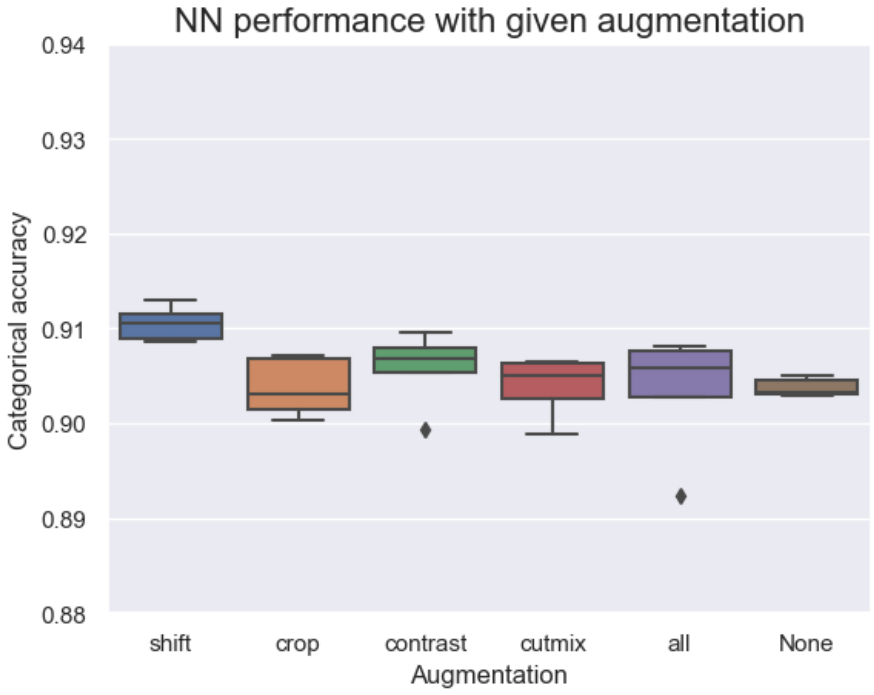

Figure 5: Neural Networks' performance with given augmentation method.

The results suggest that, for CIFAR-10, the best augmentation to use is randomly shifting, improving the final accuracy by over half a percent. We can also observe that for contrast, which was the second best performing, and shift, the spread of the accuracy between executions was low. While for the cropping and cutmix, the random state had a higher influence, which consequently transfers into running all of the methods sequentially. We can also see that all of the methods, except for the cropping, improved the models. This could be explained either by the introduction of fully black pixels or by the incorrectly set maximal value for border width.

4.3 Experiment 3

In the third experiment, we analysed whether the results can be improved by utilising ensembling with either soft voting or hard voting. Soft voting is based on combining the raw predictions from each of the models and then selecting the class which has the highest probability, while hard voting is based on counting how many times each class was selected by a model and picking the most chosen one. To reduce the required computational resources, we used predictions produced by the models from the first and second experiments, in which we accumulated a total of 380.

We ordered them by the accuracy scores they received and then combined 10%, 20%, 30%, 40%, 50%, and 100% of the best-performing ones. The results have been presented in Table 1.

Used models	**Soft voting accuracy**	**Hard voting accuracy**
10%	92.99%	93.07%
20%	92.86%	92.84%
30%	93.06%	93.04%
40%	92.71%	92.49%
50%	92.82%	92.52%
100%	92.17%	91.21%

Table 1: Comparison of ensemble strategies.

The results mentioned in Table 1 show an increase in performance in comparison to the best model from the first and second experiments, which achieved an accuracy of 91.29%. We can see that the difference between soft and hard voting is marginal, and neither can be pointed out as superior in this task. We can also notice that optimal ensembling requires a balance between the number of models and their individual performance.

4.4 Kaggle competition

In addition to the performed experiments, part of our research was finding the best architecture to use for a Kaggle contest based on the CIFAR-10 architecture. While the best-performing method was the use of an ensemble, we were unable to use those models due to the shortcut we took to save computational resources. In the end, we decided to use a randomly initialised Mobile Net, with a learning rate of 0.01, batch size of 64, dropout of 0.2 and no L2 regularisation, trained on the dataset augmented with image shifting. In the end, we managed to achieve the accuracy of (TODO: WSTAWIĆ ACCURACY) on the test set.

5 Evaluation and reproduction notes

Presented results are also available in our GitHub repository (TODO: LINK DO REPO) in the form of a calculated Jupyter Notebook file. The results are closely reproducible, as we provide the original execution code and all the necessary parameters that we set for the neural architectures. The notebook package requirements are provided in Table 2.

The calculations were conducted on the GPU on the PC with Windows 11, CUDA 12.1.0, and GPU Ge Force RTX 3070 Ti (24 GB of memory: 8GB of dedicated memory, and 16GB shared

memory). We highly recommend running the computations on more powerful devices as the experiments lasted almost 16 hours, with time for the particular experiments: ex.1: 850 minutes - 14 hours, ex.2: 94 minutes - 1.5 hours, ex.3: 5 minutes.

Package	**Version**
keras_cv	0.3.5
tensorflow_datasets	4.8.3
pycocotools	2.0.6
keras	2.10.0
pandas	1.5.3
numpy	1.24.2
seaborn	0.12.2
matplotlib	3.7.1
typing-extensions	4.5.0
Keras-Preprocessing	1.1.2
pyarrow	11.0.0
scikit-learn	1.2.2
tensorflow	2.10.1
tqdm	4.61.2

Table 2: Package requirements.

6 Further research

One of the experiments we were unable to perform due to technical limitations, which results could prove to be useful, would be an expanded search on the batch size hyper-parameter. Additionally, the unexpected relationship between the randomly initialised Mobile Net and a pretrained Mobile Net could be further explored by including a model based on the Mobile Net architecture initialised on the pretrained weights, but which has not been frozen. The analysis of the influence of data augmentation could also be expanded by running a grid search on the parameters of the augmentation, such as maximal shift distance, as well as the inclusion of more advanced data augmentation methods ([HB20]).

7 Conclusions

In this paper we've analysed an impact of different hyper-parameters and methods connected to image classification with convolutional neural networks. Firstly, we've shown a huge impact on hyper-parameter values, as all of them greatly change the models' performance. Moreover our case proved that pretrained architectures are not always better option than the randomly initialised ones. Additionally we've also show how different augmentation methods improve the quality of the CNN predictions. Finally we've also proved that even simple ensembling methods like hard and soft voting can greatly improve the outcomes of our model portfolio if we create an ensemble of best performing specimens.

References

[BGNK18] William H. Beluch, Tim Genewein, Andreas Nurnberger, and Jan M. Kohler. The power of ensembles for active learning in image classification. pages 9368–9377, 2018.

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

[HB20] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. Deep network ensemble learning applied to image classification using cnn trees. 2020.

[HP19] Tien Ho-Phuoc. Cifar10 to compare visual recognition performance between deep neural networks and humans. 2019.

[HSC+19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. 2019.

[ON15] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. 2015.

[PW17] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. 2017.

[TL20] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2020.

[YHO+19] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. 2019.

List of Figures

List of Tables