Project I - convolutional neural networks

Topic: Image classification with convolutional neural networks

Mikołaj Malec and Marceli Korbin

1. Table of contents

2. Introduction

The goal of our project was to:

1. compare different deep neural network architectures
2. investigate the influence of the hyper-parameter change

3. investigate the influence of the data augmentation techniques

Deep learning is a subset of machine learning that's based on artificial neural networks. The learning process is deep because the structure of artificial neural networks consists of multiple input, output, and hidden layers. Each layer contains units that transform the input data into information that the next layer can use for a certain predictive task. Thanks to this structure, a machine can learn through its data processing. [1]

Convolutional neural networks (CNN) are distinguished from deep neural networks by using the convolutional layer and pooling layer. The **convolutional layer** computes the output of neurons that are connected to local regions in the input. Regions are in most cases 3x3 matrix (3x3 pixels in case of image). The **pooling layer** computes some pooling functions (for example, maximum) over a specified region, with **stride** expanding layer size by that amount for better performance of the pooling layer. [2]

ResNet is CNN with the concept of adding an intermediate input to the output of a series of convolution blocks. It means saving computed values over specified layers and using them as additional input after some number of layers. ResNet provides an innovative solution to the vanishing gradient problem. [3]

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. In this project, VGG19 pre-trained network from the Keras library was used as a base of the neural network. [4]

Hyperparameters of neural network training, which were tested in this project:

- **Learning rate** - hyper-parameter used to govern the pace at which an algorithm updates or learns the values of a parameter estimate [5]
- **Momentum** - coefficient that is applied to an extra term in the weights up, during training the update direction tends to resist change when momentum is added to the update scheme [6]
- **Dropout rate** - some number of layer outputs are randomly ignored or “dropped out”, has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsible for the inputs [7]
- **Regularization -** regularization updates the general cost function by adding another term known as the regularization term, due to the addition of this regularization term, the values of weight matrices decrease, which prevents overfitting [8]

Data augmentation is the technique of increasing the size of data used for training a model by changing the data used in training by using augmentation techniques. Therefore, the existing data is augmented to make a better-generalized model.
Augmentation techniques used in this project: [9][10]

- **Rotation** - an image is rotated randomly
- **Cropping** - a random portion of the image is selected
- **Translation** - and the image is moved along the x-axis and y-axis

- **Cutout** - randomly masks out square (random size in this project case) region of input during training

3. Instructions for the application

The code, which was run in order to achieve the results, can be found on Kaggle notebooks under the links: [11] (grid search tests), [12] (augmentation tests), [13] (visualization). For the first two notebooks, we recommend using the GPU P100 accelerator, with which the code in each link can be run in 7.5 hours.

4. Network architecture and implementation:

We implemented testing in Python using the **Keras and TensorFlow** library. The goal was to test different approaches to the same problem of classification. Networks architectures were looked up on the internet and chosen and modified in a small way based on several parameters to train.

If not stated otherwise, the activation function is **ReLu**. If the dropout is bigger than 0, it is activated on every convolutional and dense layer.

Network 1 (ResNet 8, 8 layers) [14]

The network was chosen as a representation of Resent family.

Input: 3x32x32

1. 3x3, Conv, 32
2. save I
3. 3x3, Conv, 64
4. 3x3, Conv, 64 + I
5. save I
6. 3x3, Conv, 128
7. 3x3, Conv, 128 + I
8. save I
9. 3x3, Conv, 256
10. 3x3, Conv, 256 + I
11. Dense, 10 (Softmax)

Trainable params: 1,596,538

Network 2 (CNN, 8 layers) [15]

The network was chosen from a paper, which used the method proposed in the paper to find the best architecture. The smaller proposed model was used.

Input: 3x32x32

1. 5x5, Conv, 96
2. 2x2, max pool, 96, /2 stride
3. 5x5, Conv, 80
4. 2x2, max pool, 80, /2 stride
5. 5x5, Conv, 96
6. 5x5, Conv, 64
7. Dense, 256
8. Dense, 10

Trainable params: 1,561,674

Network 3 (VGG19) [16]

It is a pre-trained Keras model - VGG19 on ILSVRC-2012 dataset with added dense layers.

Input: 3x32x32

1. pre-trained Keras model - VGG19 (weights frozen)
2. Flatten (512)
3. Dense, 512
4. Dense, 256
5. Dense, 128
6. Dense, 10

Total params: 20,452,554
Trainable params: 428,170
Non-trainable params: 20,024,384

5. Data

The CIFAR-10 image dataset was used (https://www.kaggle.com/c/cifar-10). CIFAR-10 is an established computer-vision dataset used for object recognition. It consists of 60,000 32x32 color images containing one of 10 object classes, with 6000 images per class. There is 50,000 image in train data and 10,000 in test data.

Data preprocessing

Normalization of images into range 0–1 (from 0–255).

6. Testing methodology:

All testing was done on CIFAR-10 data.

Testing was pleated in two parts:

In the first part: the 3 mentioned network architectures were compared. Different hyperparameters were tested by search on the grid (144 combinations in total):

- Learning rate: 0.1, 0.01, 0.001, 0.0001
- Momentum: 0.5, 0.75, 0.99
- Dropout rate: 0, 0.5
- Regularization: L2, none

In the second part: for the best hyperparameters on the best-performing model, tests were performed using the following augmentation techniques:

1. Rotation
2. Cropping
3. Translation
4. Cutout
5. All of the above at once

The testing metric was accuracy since classes in the dataset are equally balanced. The accuracy change is shown according to the number of epochs. To reduce the number of experiments, but to show expected performance under other hyperparameters, which don't have to be optimized, and by taking the naive assumption, that hyperparameters don't affect each other performance "gain", each hyperparameter accuracy was averaged from all experiments with the same hyperparameter. For example, accuracy over 0.1 learning rate is a mean of experiments with 0.1 learning rate, different momentum, different dropout rate, and different regularization.

To reduce computation experiments were carried out only to the 20th epoch.

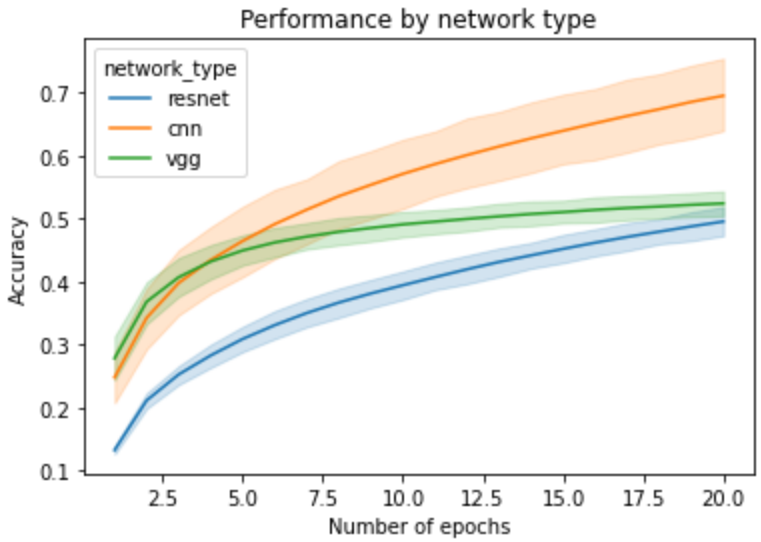
By using data augmentation the data set size wasn't increased.

7. Results

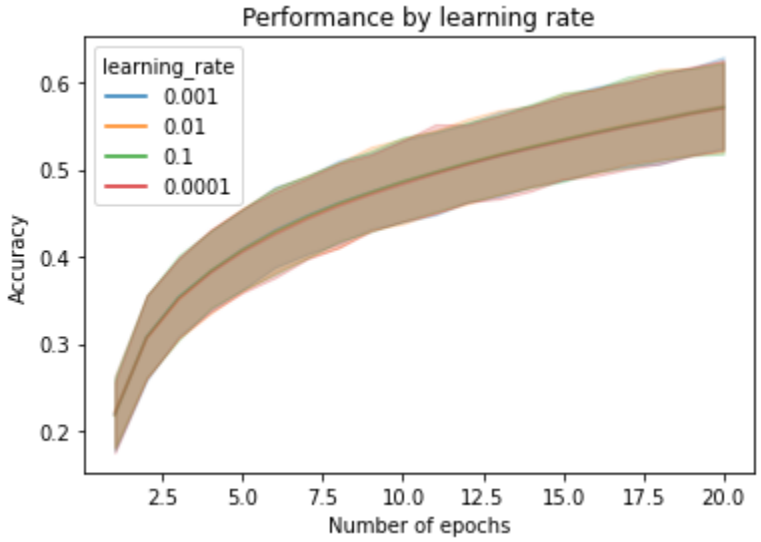
Grid search

Mean of accuracy on testing set with 95% confidence interval:

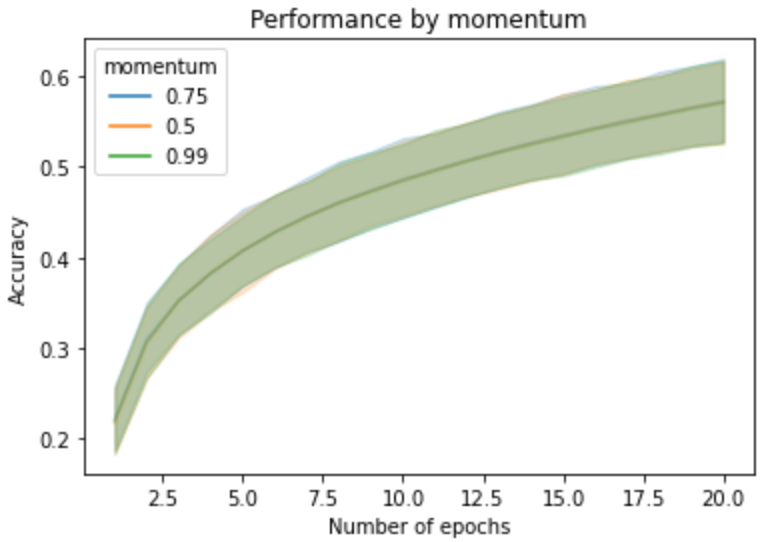
By network type:



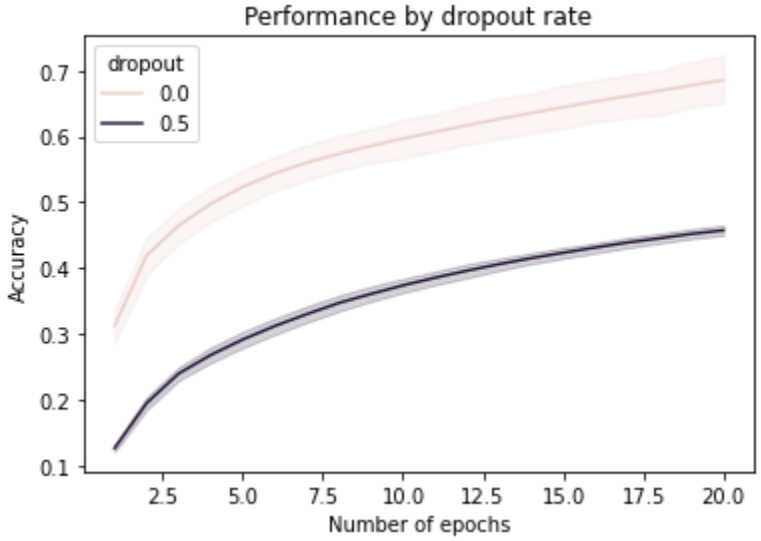
By learning rate:



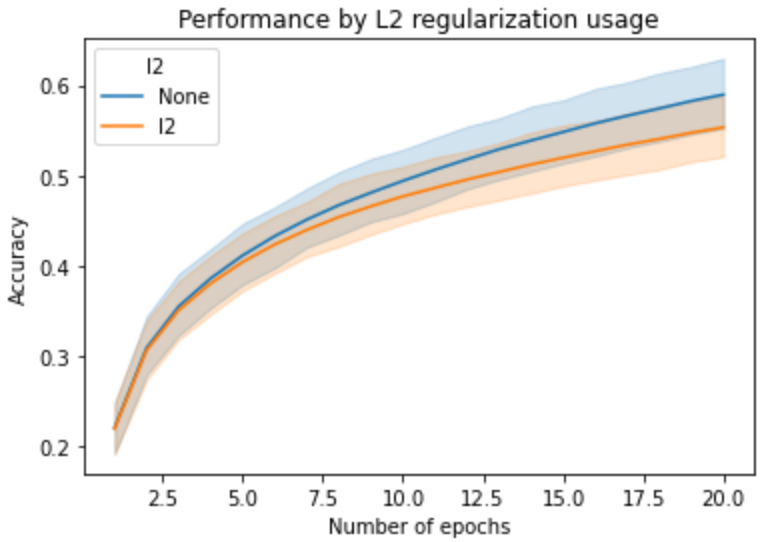
By momentum:



By dropout rate:



By usage of L2 regularization:

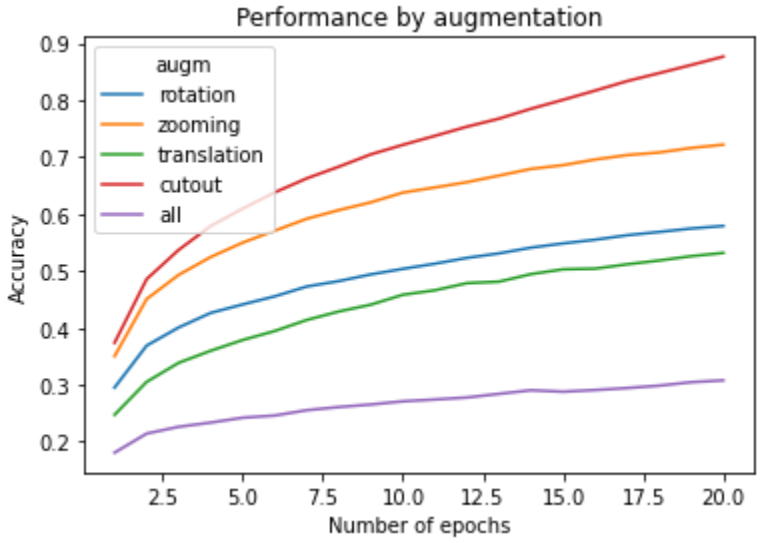


The models are on average more accurate if they use no dropout or regularization, while momentum and learning rate do not influence the accuracy; the best-performing model type is CNN.

Among all the models separately, the highest accuracy after 20 epochs – 93.44% – was achieved by a CNN model with a learning rate of 0.01, a momentum of 0.99, no dropout and no regularization.

Augmentation methods

The best-performing model was later trained on data modified by various augmentation methods.



The highest accuracy after 20 epochs – 87.67% – was achieved while training on data augmented by the cutout.

8. Conclusions

CNN was the best-performing model. That's because we took its architecture knowing it was optimized for this dataset. ResNet was a general-use approach and VGG19 was trained on different datasets. The performance of the ResNet could be improved by choosing a different (bigger architecture, as the computational requirements, should be smaller). The performance of VGG19 could be improved by unfreezing the weights of the imported model and adding one more dense layer.

The behavior of accuracy of training of VGG19 is interesting, as the increase in accuracy dramatically drops after 5 epochs. We guess it's because the model has saturated after this time.

Surprisingly, we didn't find a large impact of learning rate and momentum hyperparameters on the performance of the models.

The use of 0.5 dropout significantly worsens a model. It may be, because it's too much for a model to 'handle', and resulting generalization significantly impacts the model. Using smaller dropout could improve the accuracy.

Usage of L2 regularization worsens a model a little bit.

In general, because dropout and regularization showed worsened results, it may mean that the model is under-fit (except maybe the VGG19 model). Increasing the number of training epochs would be a solution.

Every augmentation method worsened the final model, with using all augmentation methods significantly dropping the accuracy of the model. It's because in our training we didn't increase the dataset size by doing augmentation, which is the main reason, why augmentation increases the performance of the model. In a way, we have shown information loss of each augmentation.

- Cutout has the best performance, as it leaves most of the image intact. In our case, we randomly selected the masked region size, so sometimes the masked region was very small. Also, masking could mask unimportant regions of the image, preserving the important structures for a model to recognize.
- Zoom is second, as it also leaves the image somewhere intact, but it deletes the information. Because zoom is at random places and sizes, deleted information could be none (very small zoom), small (zoom at important structure), or very big (zoom at a not important part of the image).
- Rotation and translation have a bigger impact on the model because they distort the structures in the image. They may have a bigger impact on an over-trained model, but in our case models were still under-fit.
- If all methods mentioned above are implemented, the information loss is significant.

9. Bibliography

[1] Deep learning vs. machine learning in Azure Machine Learning
https://learn.microsoft.com/en-us/azure/machine-learning/concept-deep-learning-vs-machine-learning

[2] CS231n Convolutional Neural Networks for Visual Recognition
https://cs231n.github.io/convolutional-networks/

[3] ResNet: The Basics and 3 ResNet Extensions
https://datagen.tech/guides/computer-vision/resnet/

[4] A Gentle Introduction to Transfer Learning for Deep Learning
https://machinelearningmastery.com/transfer-learning-for-deep-learning/

[5] What is the learning rate in Machine Learning?
https://deepchecks.com/glossary/learning-rate-in-machine-learning/

[6] What is momentum in a neural network?
https://datascience.stackexchange.com/questions/84167/what-is-momentum-in-neural-network

[7] A Gentle Introduction to Dropout for Regularizing Deep Neural Networks
https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

[8] An Overview of Regularization Techniques in Deep Learning (with Python code)
https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

[9] Data augmentation Techniques
https://iq.opengenus.org/data-augmentation/

[10] Cutout
https://paperswithcode.com/method/cutout

[11] DNN project (M. Malec, M. Korbin): grid search
https://www.kaggle.com/code/marcelikorbin/dnn-project-m-malec-m-korbin-grid-search

[12] DNN project (M. Malec, M. Korbin): augmentation
https://www.kaggle.com/code/marcelikorbin/dnn-project-m-malec-m-korbin-augmentation

[13] DNN project (M. Malec, M. Korbin): visualization
https://www.kaggle.com/marcelikorbin/dnn-project-m-malec-m-korbin-visualization

[14] CIFAR-10 RESNET-8
https://www.kaggle.com/code/filippokevin/cifar-10-resnet-8

[15] FAWCA: A Flexible-greedy Approach to find Well-tuned CNN Architecture for Image Recognition Problem
https://www.researchgate.net/publication/326816043_FAWCA_A_Flexible-greedy_Approach_to_find_Well-tuned_CNN_Architecture_for_Image_Recognition_Problem

[16] CIFAR-10 Keras Transfer Learning
https://www.kaggle.com/code/adi160/cifar-10-keras-transfer-learning