Deep Learning Methods
Project 2

Jakub Kała & Krzysztof Spaliński

April 2020

Abstract. The scope of the second project for Deep Learning Methods course are convolutional neural networks used in image recognition tasks. The goal is to implement and explore several network architectures and data augmentation methods. Model assessment will be carried out based on CIFAR-10 dataset.

1 Documentation

All scripts used in this project have been presented in github repository deep-learning-methods-project-2.

2 CIFAR-10 Dataset

CIFAR-10 [KNH] is a computer-vision dataset used for object recognition. In our case, it is a subset of the 80 million tiny images dataset and consists of 60,000 32x32 color images containing one of 10 object classes, with 6000 images per class. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

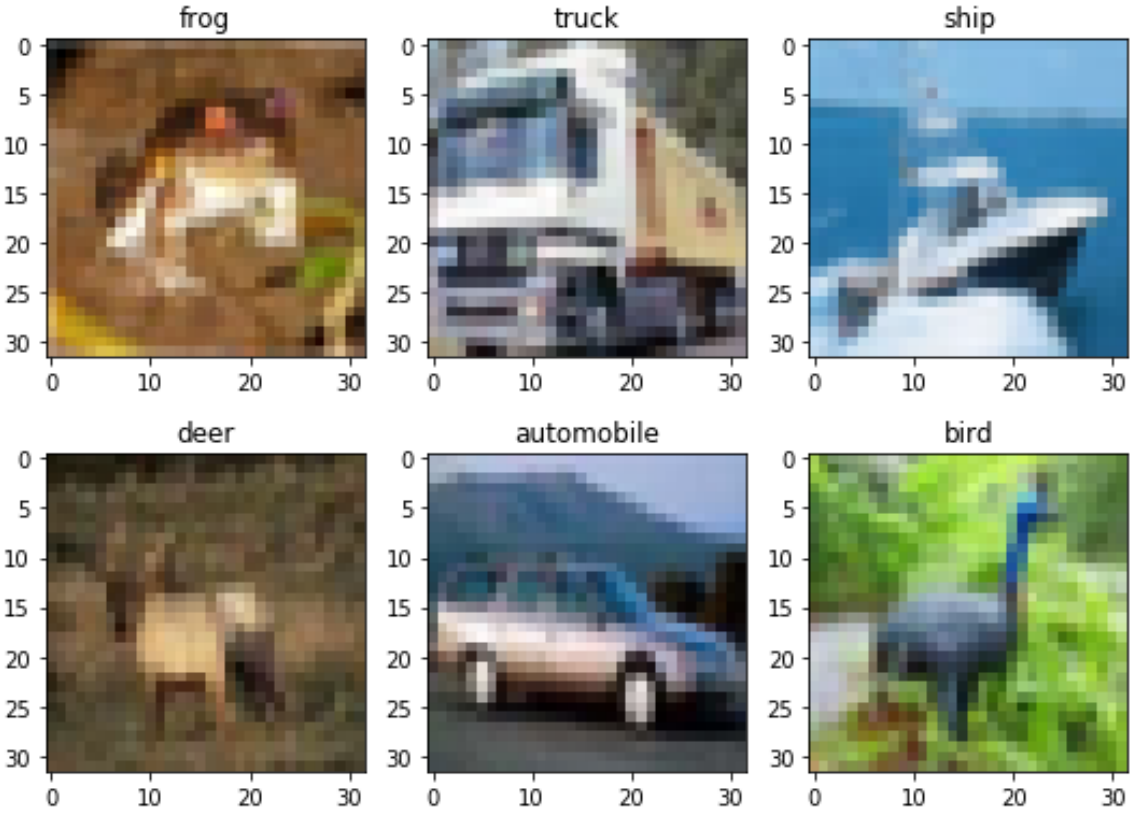


Figure 1: Example images from CIFAR-10 dataset.

Dataset has been split to training and test set (50k and 10k images respectively). In order to discourage certain forms of cheating (such as hand labeling) 290,000 junk images have been added to the test set. These images are ignored in the scoring. The classes are completely mutually exclusive.

The label classes in the dataset are:

- airplane
- automobile
- bird
- cat
- deer
- dog
- frog
- horse
- ship
- truck

3 Convolutional Neural Networks

Covolutional Neural Networks, CNNs, are neural networks employing convolution operation. They are really specialized for data with grid-like topology [GBC].

Convolution is terms of neural network is operation on two tensors: input and kernel (sometimes called a filter). Researchers developed many kernels that can be used for linear image processing, for example for blurring, sharpening, detecting edges etc. In deep learning, elements of filters are trainable parameters.

3.1 Plain CNN architecures

In this experiment we created three different CNN architectures, all based on VGG networks [SZ14] – each network is build from blocks like this: Conv2D, BarchNorm, Conv 2D, BatchNorm, MaxPooling. Size of filters can vary, but in all architectures the number of filters increases in later layers. Last 2 layers of all networks are fully connected layers with ReLU activation function. Architectures are shown in the table blow.

	Plain CNN 1	Plain CNN 2	Plain CNN 3
Network Architecture	Conv2D 16 3x3 filters BatchNorm Conv2D 16 3x3 filters BatchNorm MaxPooling Conv2D 32 3x3 filters BatchNorm Conv2D 32 3x3 filters BatchNorm MaxPooling Conv2D 64 3x3 filters BatchNorm Conv2D 64 3x3 filters BatchNorm MaxPooling Conv2D 128 3x3 filters BatchNorm Conv2D 128 3x3 filters BatchNorm MaxPooling Dense 512 Dense 512 Dense 10 (softmax)	Conv2D 16 3x3 filters BatchNorm Conv2D 16 3x3 filters BatchNorm MaxPooling Conv2D 32 3x3 filters BatchNorm Conv2D 32 3x3 filters BatchNorm MaxPooling Conv2D 64 3x3 filters BatchNorm Conv2D 64 3x3 filters BatchNorm MaxPooling Conv2D 128 3x3 filters BatchNorm Conv2D 128 3x3 filters BatchNorm MaxPooling Conv2D 256 2x2 filters BatchNorm Conv2D 512 2x2 filters BatchNorm MaxPooling Dense 1024 Dense 1024 Dense 128 Dense 10 (softmax)	Conv2D 16 3x3 filters BatchNorm Conv2D 16 3x3 filters BatchNorm MaxPooling Conv2D 32 3x3 filters BatchNorm Conv2D 32 3x3 filters BatchNorm MaxPooling Conv2D 64 3x3 filters BatchNorm Conv2D 64 3x3 filters BatchNorm MaxPooling Conv2D 128 3x3 filters BatchNorm Conv2D 128 3x3 filters BatchNorm MaxPooling Conv2D 256 2x2 filters BatchNorm Conv2D 256 2x2 filters BatchNorm MaxPooling Conv2D 512 2x2 filters BatchNorm Conv2D 512 2x2 filters BatchNorm MaxPooling Dense 1024 Dense 1024 Dense 10 (softmax)
# trainable parameters	826k	1.659kk	3.843kk

Table 1: Detailed information about plain CNN architectures.

Each network was trained until there was no progress on validation set for 20 epochs. Validation set for each process of training was chosen randomly. Each architecture was trained with and without data augmentation. Training process was repeated 10 times, so there was 60 networks trained in total (3 architecturs $\times$ 2 augmentation or no augmentation $\times$ 10). Results of experiments are presented in the chart and table blow.

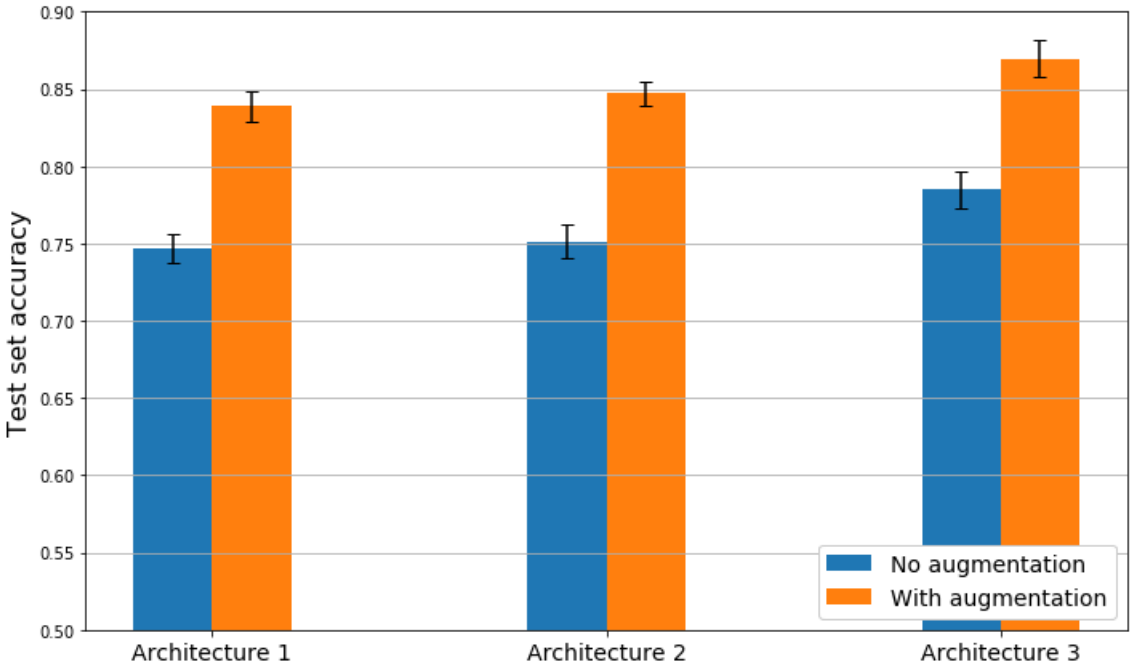


Figure 2: Mean accuracy and its standard deviation on the test set.

	No augmentation		**With augmentation**	
	mean	std	mean	std
Architecture 1	0.747	0.009	0.838	0.010
Architecture 2	0.751	0.011	0.847	0.008
Architecture 3	0.784	0.012	0.869	0.012

Table 2: Detailed information about plain CNN architectures.

As shown above, augmentation of data that CNN is trained on results in much better results on test set. We can also observe, that it does not have a visible impact on standard deviation of the results.

The biggest network, Architecture 3, achieved the best results of accuracy on the test set. Results of Architecture 2 were slightly better than results of Architecture 1 – Architecture 2 also had lower standard deviation (after training on augmented dataset).

4 Neural network with residual blocks

4.1 Residual block overview

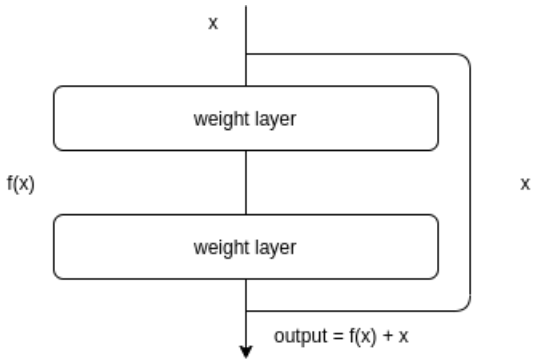


Figure 3: Residual block example with 2 skipped layers (source: self-made).

As the neural networks start converging, a degradation problem occurs. For plain architectures, accuracy gets saturated and then degrades rapidly. This is not due to model overfitting and implementing more complex architectures does not solve the problem. To address this issue, residual networks have been introduced in order to solve the degradation problem. Instead of fitting stacked layers, we let these layers fit a residual mapping. Formulation of this idea might be realized via implementing "shortcut connection" in a neural network. Shortcut connections are created via an identity mapping between one or more skipped layers. Details and some mathematical background can be found in [He+15].

4.2 Experimental comparison of ResNet architectures

In this experiment, we test 4 different architectures based on residual identity blocks. In these networks each residual block has 2 Conv2D layers and a skipping connection between them, but CNNs vary in their complexity. In [3] one can find detailed information about networks: their architecture, number of trainable parameters and average time of epoch evaluation (experiments have been carried out on Tesla T4 GPU on Kaggle platform). Detailed procedure has been presented in github repository.

Experiment pipeline consists of 5 repetitions of model training, 50 epochs each. The results are further averaged and represented in [4].

	Architecture 1	Architecture 2	Architecture 3	Architecture 4
Network Architecture	Conv2D 64 filters 3x ResNet block (64 filters) BatchNorm & AvgPooling Dense (softmax)	Conv2D 128 filters 3x ResNet block (128 filters) Conv2d 64 filters 3x ResNet block (64 filters) BatchNorm & AvgPooling Dense (softmax)	Conv2D 256 filters 3x ResNet block (256 filters) Conv2D 128 filters 3x ResNet block (128 filters) Conv2d 64 filters 3x ResNet block (64 filters) BatchNorm & AvgPooling Dense (softmax)	Conv2d 512 filters 1x ResNet block (512 filters) Conv2D 256 filters 3x ResNet block (256 filters) Conv2D 128 filters 3x ResNet block (128 filters) Conv2d 64 filters 3x ResNet block (64 filters) BatchNorm & AvgPooling Dense (softmax)
# trainable parameters	107k	1.07kk	4.92kk	10.9kk
Average epoch runtime	30s	44s	113s	196s

Table 3: Detailed information about network architectures used in residual CNN comparison.

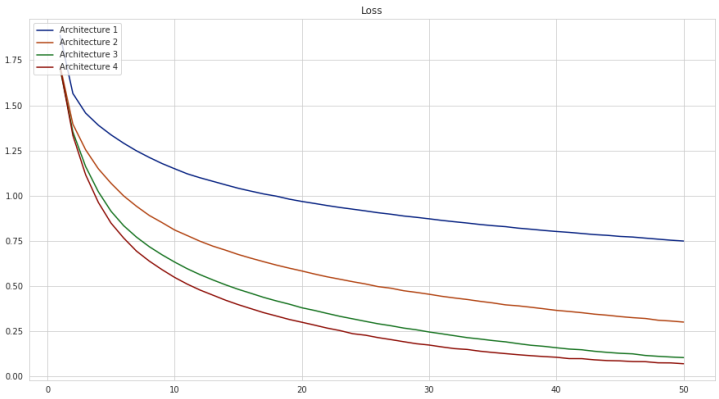
In [4] we can see that networks 1 and 2 struggle with underfitting. There is substantial difference between them and the last two architectures. Although computation complexicity for architecture 4 is much higher than for network 3, statistically, networks 3 and 4 give the same results.

There is also a visualisation of learning process presented in [4] and [5] (plots with errorbars in appendix). We can see that architecture 1 is simply underfitting. Networks 3 and 4 behave in very simmilar manner.

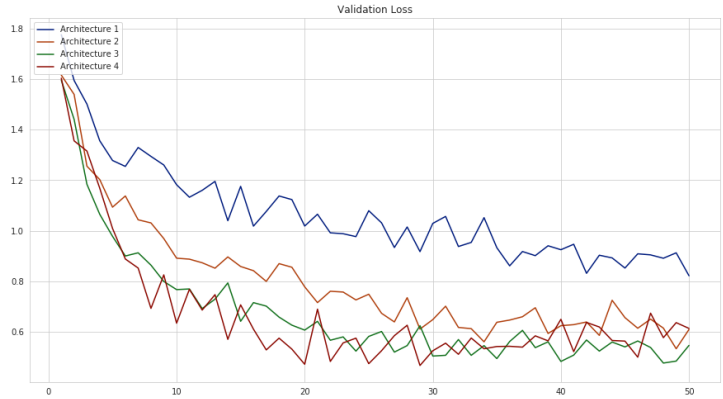
It is also worth noticing that we did not manage to receive significantly better results than plain CNNs.

	Architecture 1	Architecture 2	Architecture 3	Architecture 4
Training accuracy	0.741 ± 0.003	0.896 ± 0.002	0.965 ± 0.01	0.976 ± 0.002
Test accuracy	0.719 ± 0.01	0.829 ± 0.017	0.869 ± 0.008	0.873 ± 0.015
Training loss	0.75 ± 0.007	0.301 ± 0.005	0.105 ± 0.004	0.071 ± 0.005
Test loss	0.821 ± 0.042	0.532 ± 0.065	0.475 ± 0.049	0.466 ± 0.017

Table 4: Top results on CIFAR-10 dataset for each validated architecture

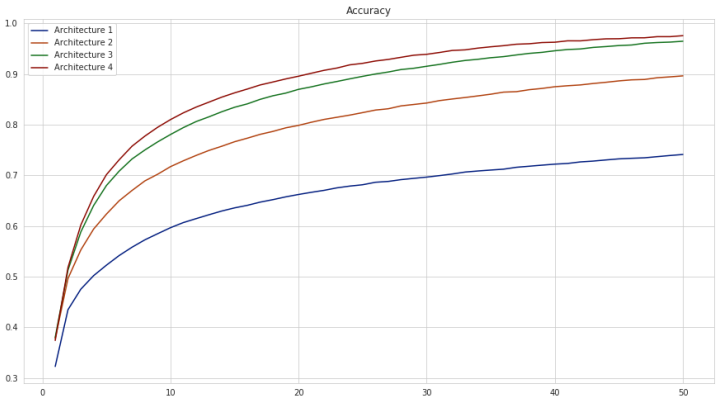


(a) Accuracy on CIFAR-10 training set for Residual CNN architectures

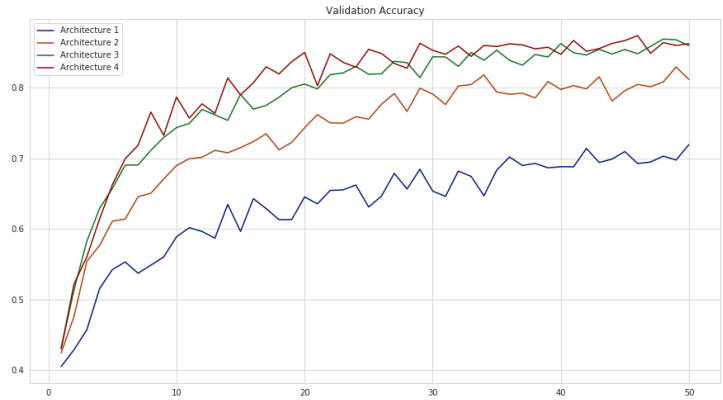


(b) Accuracy on CIFAR-10 test set for Residual CNN architectures

Figure 4: Accuracy comparison for validated CNN architectures



(a) Accuracy on CIFAR-10 training set for Residual CNN architectures



(b) Accuracy on CIFAR-10 test set for Residual CNN architectures

Figure 5: Accuracy comparison for validated CNN architectures

5 Ensemble learning for Kaggle competition

In order to submit predictions for CIFAR-10 competition on Kaggle, we have used all plain CNN networks we trained (Chapter 3) to create a max voting ensemble classifier. All those networks have been trained on train test only, with randomly picked 5% of train test as validation set and 95% used as training data. Results are presented in the table below:

	Only augmented networks used (30 voters total)	Only non-augmented networks used (30 voters total)	Both types of networks used (60 voters total)
Accuracy on test set	0.9107	0.8467	0.8952

We want to point out, that this experiment was not repeated because of its computational complexity (training of those networks on Nvidia GTX 970 took about 12 hours, making predictions took another hour). Nevertheless, results should be stable, because of the number of classifiers that took part in the voting. Screenshot from Kaggle is presented below.

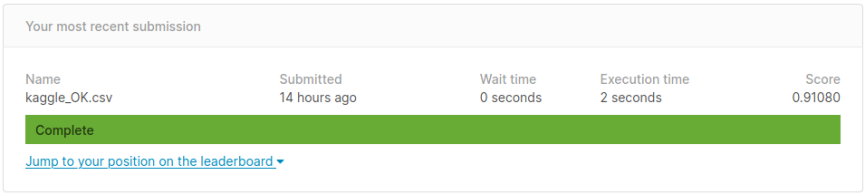


Figure 6: Screenshot with our accuracy score from Kaggle.

The result can be reproduced using this notebook to train networks. Trained models can be found here. Notebook that makes prediction on Kaggle test set can be found here. The final solution CSV file is here.

References

[SZ14] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: 1409.1556 [cs.CV].

[He+15] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: 7 (Dec. 2015).

[GBC] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*.

[KNH] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)". In: (). URL: http://www.cs.toronto.edu/~kriz/cifar.html.

6 Appendix

6.1 Experiment results with error bars

Learning histories for each architecture from ResNet experiment with error bars attached can be found in [7, 8, 9, 10]

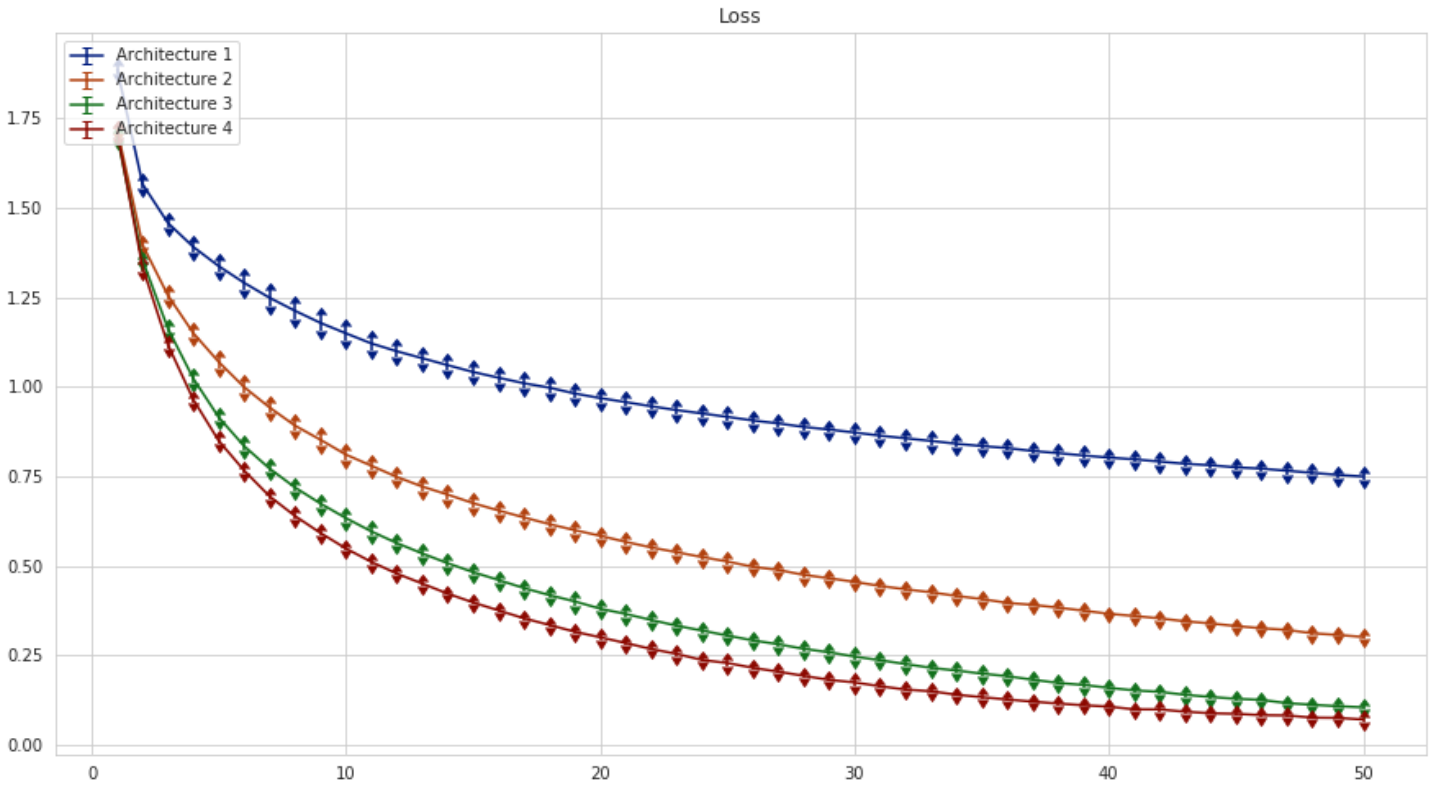


Figure 7: Residual networks experiment - Loss with errors

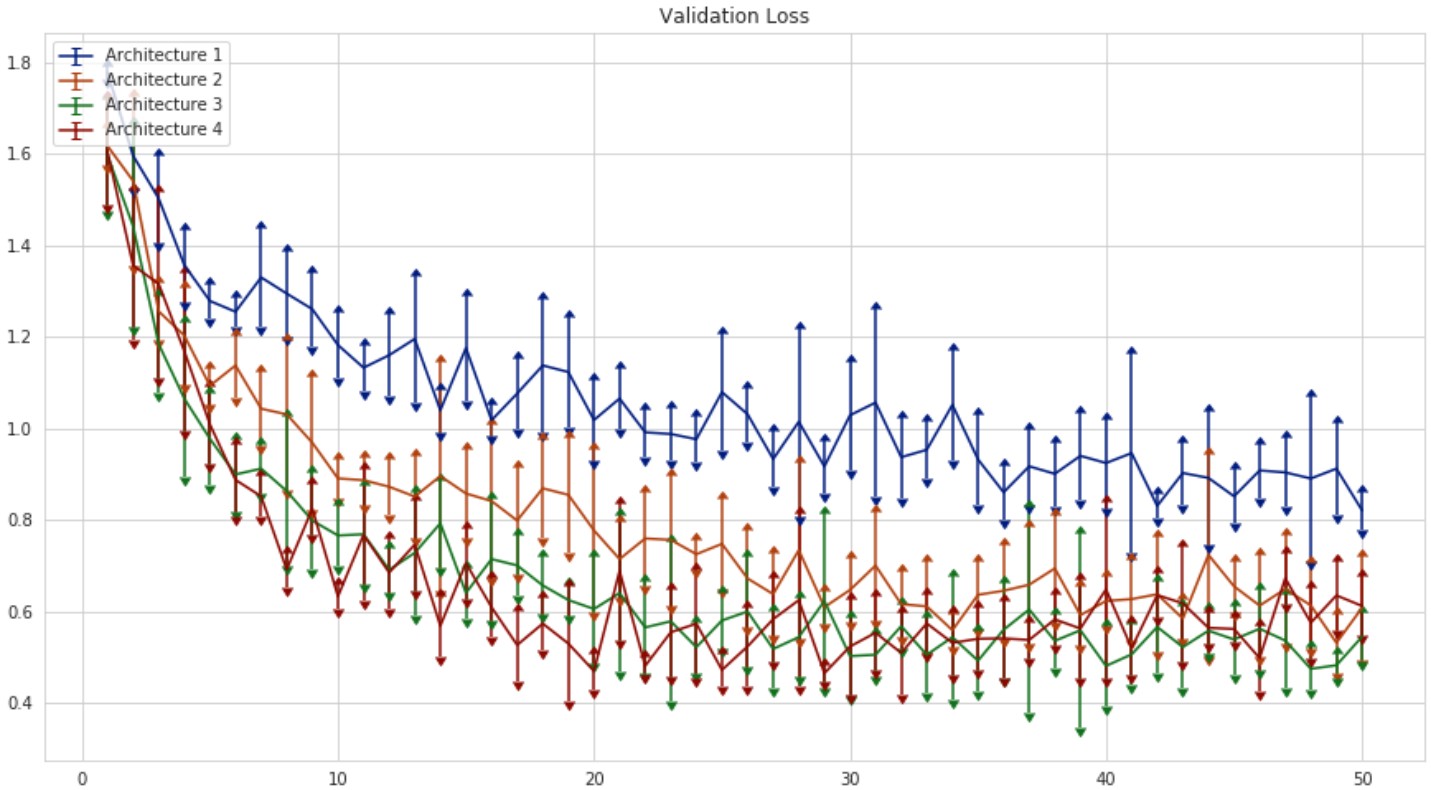


Figure 8: Residual networks experiment - Validation loss with errors

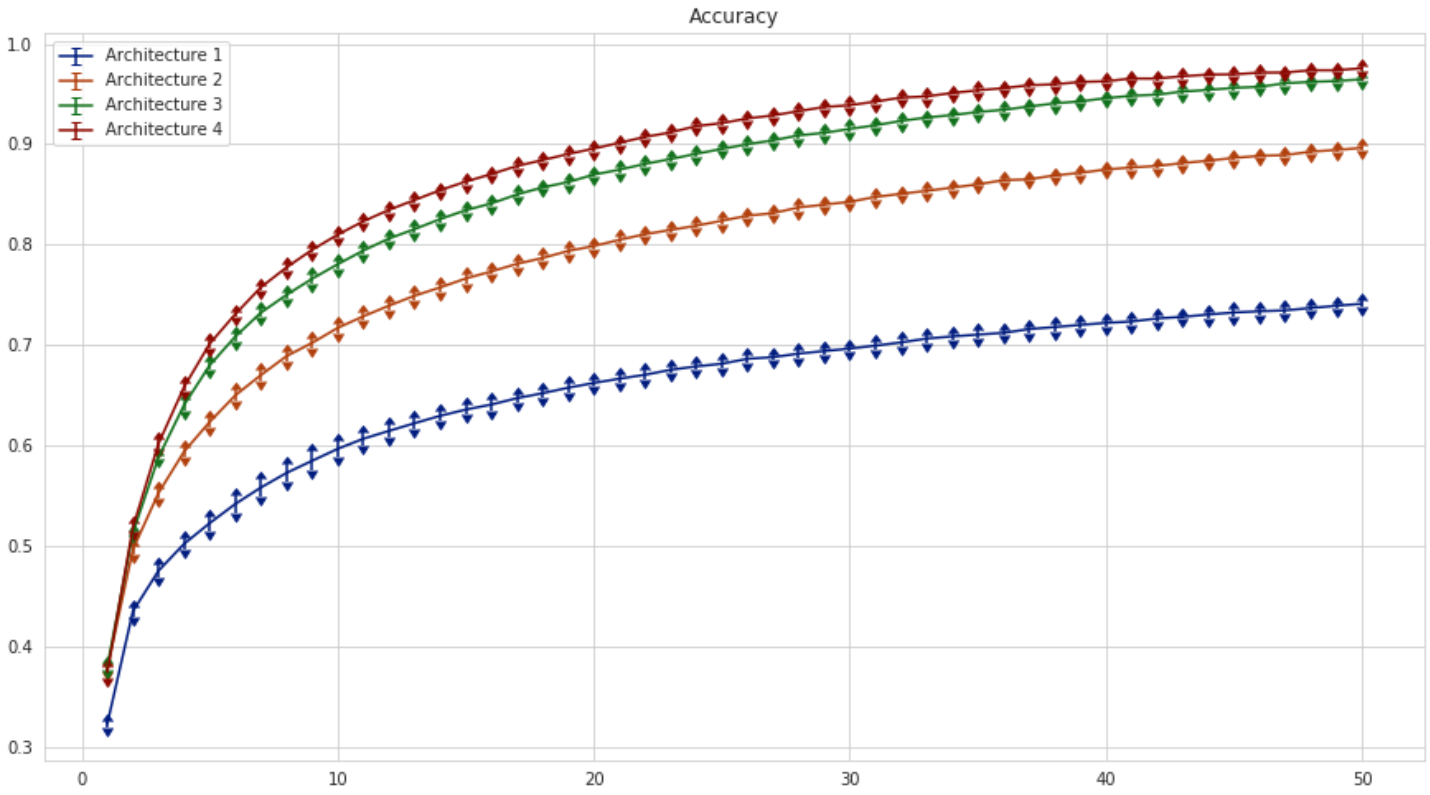


Figure 9: Residual networks experiment - Accuracy with errors

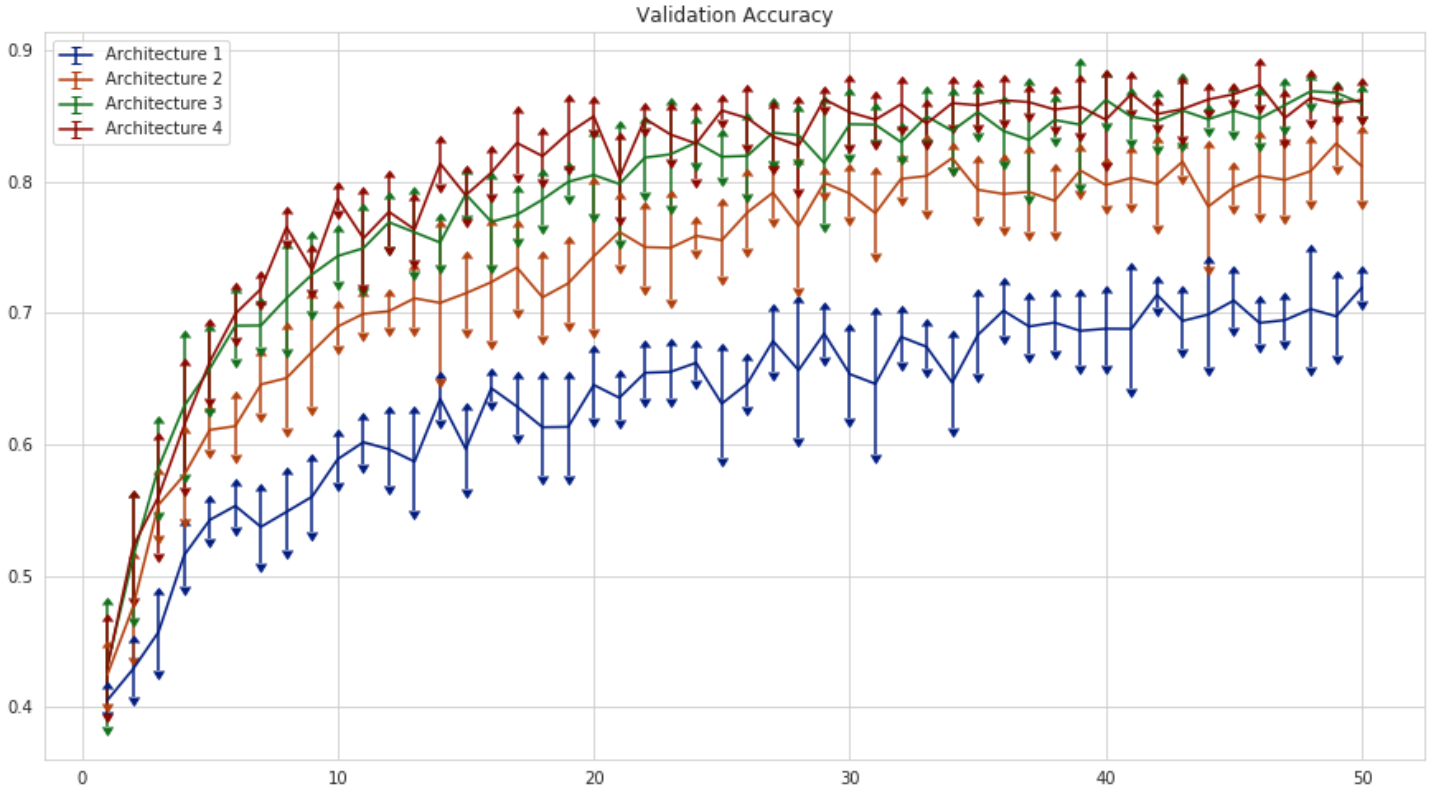


Figure 10: Residual networks experiment - Validation accuracy with errors