Object Recognition and Image Classification: A Report on the Comparison of Deep Learning Models

Training a Feed Forward Artificial Neural Network and a Convolutional Neural Network for Image Classification & Training a Residual Neural Network for Object Detection and Classification

Ereš Ana Marija (114240329)
email: ana.marija.eres@gmail.com

Ilic Ema (200837)
email: ema.ilic9@gmail.com

Kos Dominik (200842)
email: dominik.kos@fer.hr

Martin Srsen (200841)
email: martin.srsen@fer.hr

Deep Learning (103000856), Master's Degree EIT Digital in Data Science

Academic Year 2020/2021

Contents

1 Context

The assignment was divided into two parts and entails two datasets. The first part of the assignment mandates to construct and train a Feed Forward Artificial Neural Network and a Convolutional Neural Network and train it on the famous open-source CIFAR100 Dataset, a labeled dataset of 80 milion images, with 500 training and 100 testing images for each of the 100 classes.

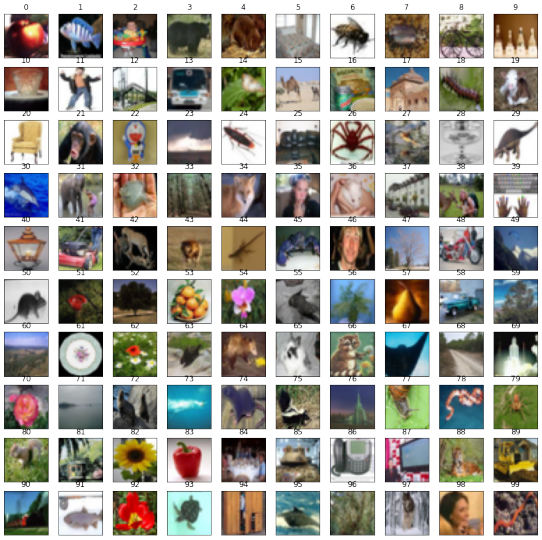

Figure 1: An examaple from each of the 100 classes of objects in the CIFAR 100 dataset

On the other hand, the second part of the assignment advises to train a residual convolutional neural network ResNet50, with 48 Convolutional layers along with one maximum pooling and one average pooling layer, with the scope of image classification. Another network is advised to be trained with the scope of object recognition. The dataset provided is the famous Traffic Sign dataset.

2 Software and Tools Provided

The team was instructed to use a Google Cloud Console and was provided a Student Education Grant coupon of value 50€ per each member in order to successfully install and run a Virtual Machine Instance in the Google Cloud Console's Compute Engine. Thus, the team was able to leverage on four NVidia Tesla K80 GPs. Thus, the team leveraged on TensorFlow2.4 and Keras.CUDA11.0.GPU in order to solve the task, using Python 3.8 in Jupyter Environment in Google Cloud.

3 Assignment 1: Training a Feed Forward Artificial Neural Network and a Convolutional Neural Network for Image Classification on CIFAR100 Image Dataset

Two approaches were used in order to train the networks, both of them described below. Due to a lack of computational resources and the lengthy process of the cross validation algorithm, it was decided to train the CNN using the careful study of the theory and the "Trial and Error" method, while the FFNN was trained both using the Cross Validation as well as the "Trial and Error" method.

3.1 Hyperparameter Tuning: Cross Validation

The initial idea was to use ScikitLearn's Grid Search algorithm for optimizing hyperparameters for both networks. However, it was soon discovered the process would be too lengthy, and due to a processing unit as well as the time constraint, the decision was taken to leverage on the Randomized Search algorithm instead. This hyperparameter-searching algorithm is different insofar as it only takes n random combinations of hyperparameters (instead of all possible combinations), trains the model using them, and returns the one with the best results (the task mandated for using accuracy as the overarching measure of success of a model). Several attempts were made for the Feed Forward Neural Network, for the sake of brevity only 3 will be described here:

- For a randomized search considering only the optimizer, accuracy of 0.151075 was obtained with Adam
- For a randomized search considering only the activation function, accuracy of 0.147500 was achieved with ReLu activation function
- For a randomized search considering activation function, dropout rate, number of layers, number of neurons per layer and learning rate, the accuracy of 0.1154 was obtained.

3.2 Trial and Error Method

This method was used both with the Feed Forward Neural Network as well as the Convolutional Neural Network. Namely, with the careful study of theory behind these models, an optimal model was found with respect to maximizing the desired measure (accuracy).

3.2.1 Feed Forward Neural Network

The Feed Forward Neural Network was constructed in the way described below. The sequential model by tensorflow was initiated and flattened. A densely connected neural network layer was added with 1024 nodes, l1 and l2 kernel regularizer set to 0.00002 and 0.0004, respectively, while the bias regularizer was chosen as Ridge Regularizer (l2) and set to 0.0002. Ridge regularizer was chosen as an activity reglarizer (applied to the output layer), too, and was set to value 0.00002. The activation function was set to Rectified Linear Unit (ReLU), and the dropout regularization was set to 0.25 for this layer. The same identical layer was repeated three more times, with the following modification applied to the final layer: This layer has as many nodes as the number of output classes in the dataset (100), and the activation function chosen was softmax, in order to make sure that all the logits (final activations interpretable as probabilities) sum up to one.

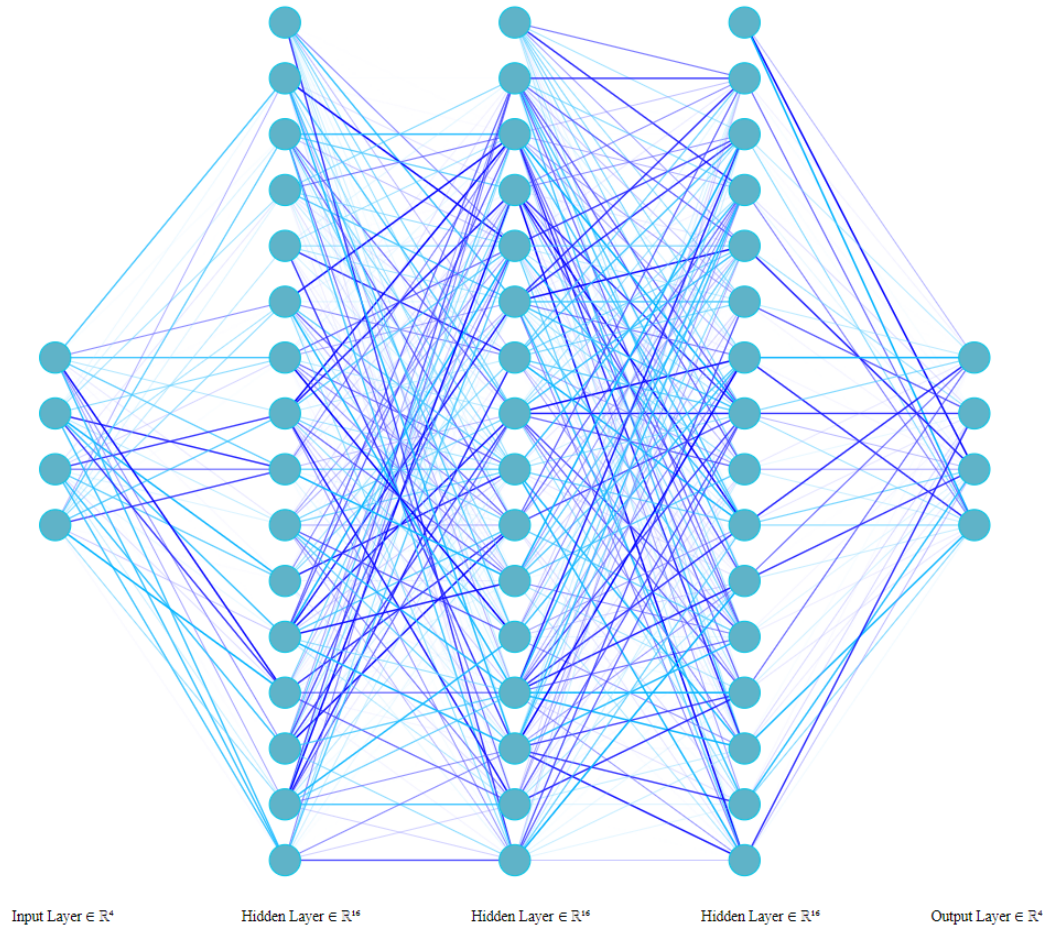

Figure 2: A simplified representation of the trained FFNN using Alex Lenail Software tool. While it was hard to visualize a FFNN with layer sizes of 100, 1024, 1024, 1024 and 100, this simplified version takes on values 4,16,16,16,4. The Edge color represents positive (blue) versus negative (turquoise) weights, while the edge opacity represents the weights.

Stochastic Gradient Descent was chosen as an optimizer with the learning rate set to 0.0012, a decay of 0.000003, and Nesterov momentum set to 0.9.

3.2.2 Convolutional Neural Network

Tensorflow's Sequential model was initiated. Then four Convolutional layers were added using a 'for' loop, with the number of output filters taking on increasing values, and the kernel of height and width three for the convolutional window, a default stride (1,1), and a ridge regression penalization. Along with each of the layers, a batch normalization step was added which further increased the accuracy. This step applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

ReLu activation function was chosen, as was the average pooling layer with a pool size of (2,2), halving the input in both spatial dimensions. This is where the loop ends, and the model procedes

with a Dropout of 0.2 which drops 20% of the input units at each update, which helps prevent overfitting. After that, the input was flattened, Activation function ReLu was chosen for the next layer and the dropout was again set to 20%. Another Dense Layer was added with as many nodes as there are classes (100) and a ridge regression bias regularizer equal to 0.01. Finally, the softmax activation function was chosen and as an Optimizer Adam was selected.

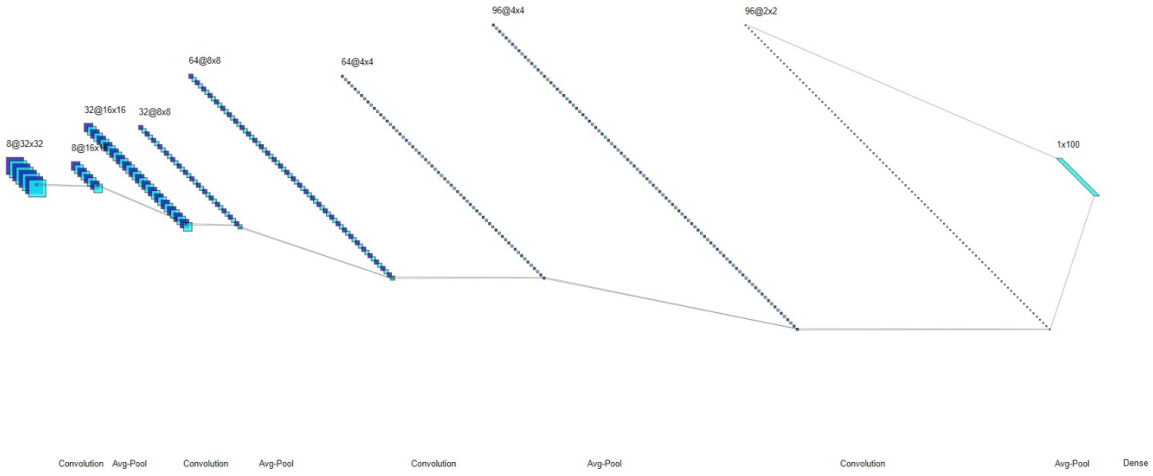

Figure 3: A simplified representation of the trained CNN using Alex Lenail Software tool. While it was hard to visualize a CNN withrealistic depths and dimensions, this simplified version with smaller

4 Assignment 2: Training a Residual Convolutional Neural Network for Object Detection and Classification

The template for ResNet50 was provided along with the aforementioned Traffic Sign dataset.

Figure 4: An examaple from each of the 43 classes of objects in the dataset

4.1 Object Localization

Certain modifications were introduced in order to train the neural network to satisfying results, and the 'Trial and Error' approach was used. As the construction of a residual neural network is rather complex, only the changes which are relevant to the performance achieved will be mentioned here. The first decision taken was to set the 'minimum_size' parameter to 15, which was later used in filtering strategy. More precisely, in Tensorflow's 'reduce_all' function, the 'minumum_size' parameter was compared to an output variable of Tensorflow's 'squeeze' function. If the output was larger than the 'minumum_size' parameter, the boolean input sensor to reduce would be True, otherwise it would be False.

Next, certain parameters were altered to modify the number of proposal areas in each image. The maximum number of positive samples taken during proposal generation, pre and post Non-maximum Suppression (NMS) algorithm, was set to 2000 and 500, respectively, while the NMS threshold was set to 0.6.

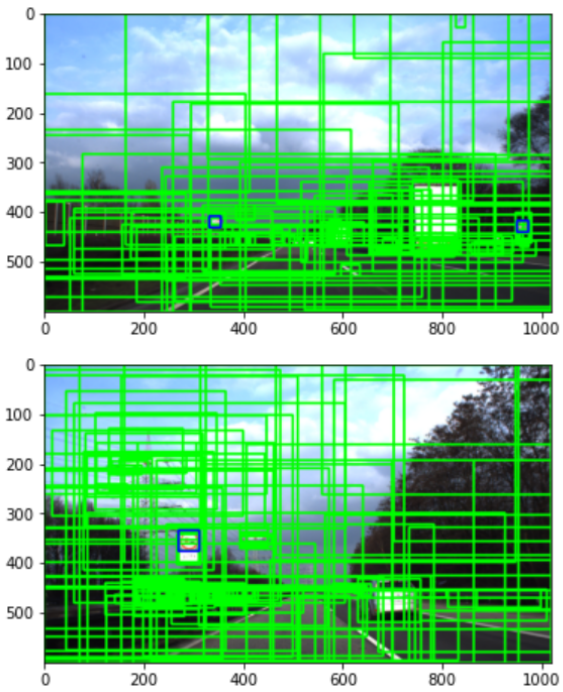

Figure 5: Interesting examples of proposal areas on two images.

4.2 Object Identification

Following that, a 'detector' convolutional neural network was trained on the dataset. Tensorflow's Sequential model was initiated. Then five Convolutional layers were added, with the number of output filters taking on increasing values of two to the power of n, with n taking on values from three to seven, included. The

kernel selected has the height and width of three for the convolutional window.

ReLu activation function was chosen for the first three convolutional layers, and the hyperbolic tangens was chosen for the last two. The dropout rate was set to 0.5 for all of the convolutional layers, except for the first one, where the dropout function was not set. The model procedes with a flattening layer followed by a Dropout of 0.2 which drops 20% of the input units at each update, which helps prevent overfitting. Hyperbolic tangens activation function was chosen for the that layer. Another Dense Layer was added with as many nodes as there are classes (43) and a ridge regression bias regularizer equal to 0.01. Finally, the softmax activation function was chosen with Adam Optimizer.

The model was fit with a batch size of 70 and 40 epochs.

5 Results

The neural networks constructed in the first assignment yielded the following results.

Model	Accuracy	Loss	Time
FFNN Rand.CV	0.151	0.654	0.698
FFNN T&E	0.318	4.760	1.697
CNN T&E	0.453	2.227	1.578

As can be observed, the best performing model was the Convolutional Neural Network, whose construction was achieved using the trial and error method. It reached the accuracy of 45.3% with loss being 2.227. On the two plots below, the accuracy and loss of this model can be compared between the training and the test set.

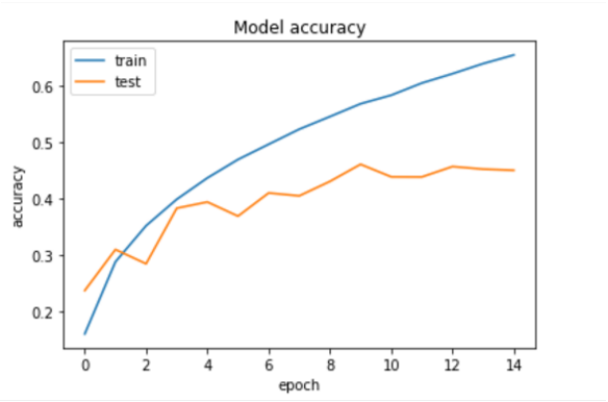

Figure 6: Accuracy of a Convolutional Neural Network

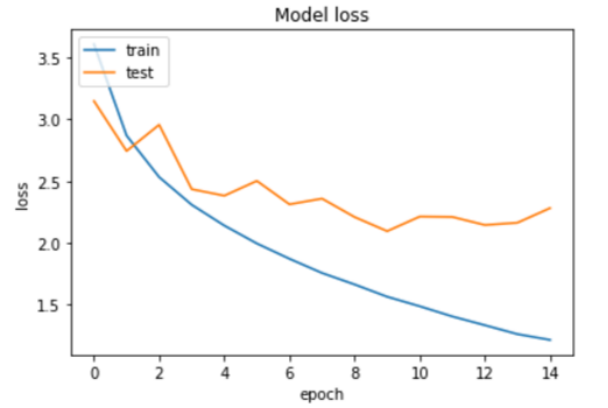

Figure 7: Loss of a Convolutional Neural Network

The second best performing model was the Feed Forward Neural Network achieved with TE method. Namely, the accuracy observed was 31.8%, with the loss of 0.0476% a trained in 1.697 seconds. Finally, the worst performing model was achieved with randomized search on a feed forward neural network, with accuracy of 0.151, loss of 0.654 in 0.698 seconds.

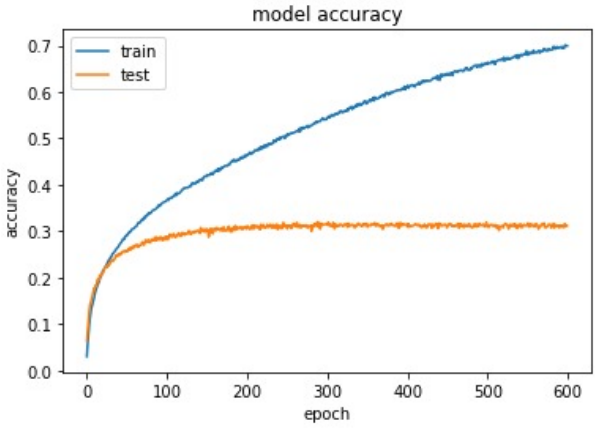

Figure 8: Accuracy of a Feed Forward Neural Network

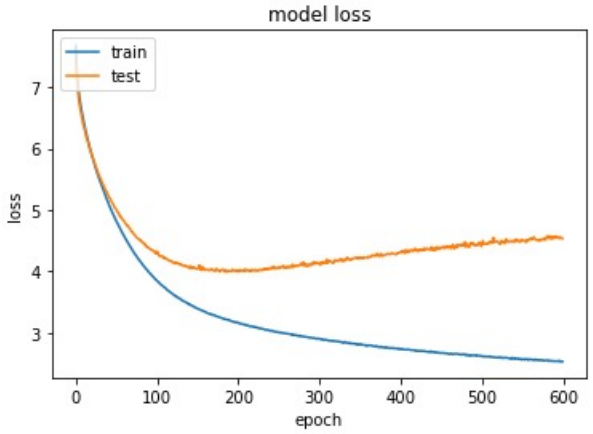

Figure 9: Loss of a Feed Forward Neural Network

On the other hand, the residual neural network trained for the second assignment ultimately yielded the accuracy of 10.85% on the traffic sign dataset, which is a 337.565% improvement with respect to the original template net, which yielded the classification accuracy of only 3.22%.

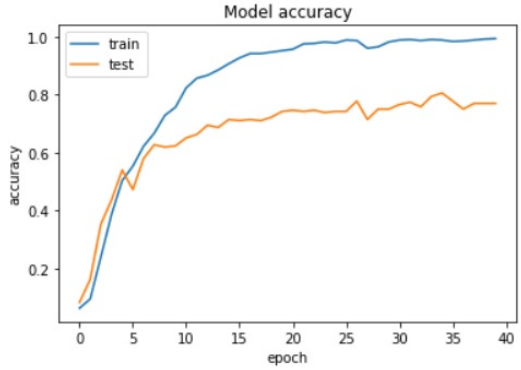

Figure 10: Accuracy of a Convolutional Neural Network

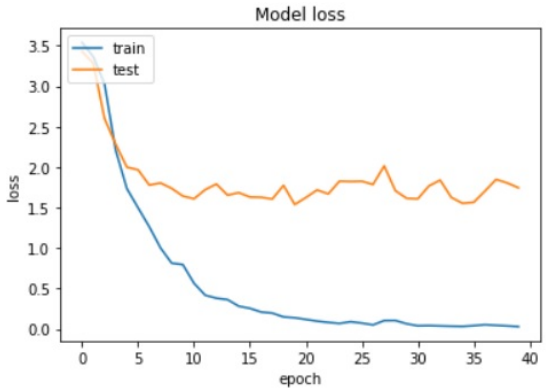

Figure 11: Loss of a Convolutional Neural Network

6 Discussion and Problems

Regarding the first assignment, a possible reason to explain the low accuracy for the the Randomized Search algorithm than with the "Trial and Error" method, might be because the Randomized Search was set to only 10 random combinations of parameters, on which the models were trained. The assumption is that if the K-Fold Grid Search Cross Validation had been used instead, a model would be obtained with the highest accuracy such a model could possibly have.

Moreover, both the FFNN and CNN plots show a 20-40% discrepancy between the train and test set accuracy. This is an example of strong overfitting. In the future work, this problem could be mended by data augmentation methods. Layer Limitation in building a CNN before the 'ResourceExhaustedError' is only 8. The solution that was implemented was building a model with less than 8 layers and leveraging on the other possible properties of the CNN (dropout, pooling layers, etc).

With respect to the second assignment, it was left unclear why the introduction of batch normalization decreased the results in terms of accuracy. The first assignment benefited heavily from the batch normalization, while the second one had the deterioration of the results. A possible explanation for that might be that not all the dataset and neural engine models are mutually compatible, which explains why we always have to train new models and network constructions for different tasks.

In the same assignment, the hyperbolic tangent function was found to yield more successful results than the relu activation function, which was not the case in the first assignment. Relu is said to work better on the hidden layers than the hyperbolic tangent (TanH) because it is a computationally less expensive and faster algorithm to load. Namely, as the size of tensors (convolutional layers) was increasing, it was observed that the TanH function works better. However it is unclear as to why is it so.

Some common problems encountered during the work on both the assignments include Python's Kernel crashing at random time intervals. This impeded the construction of the optimal model as training a neural network machine learning model is in its essence a lenghty and a computationally expensive process. The solution that was implemented was doing Grid Search with only one parameter at a time. Alternatively, when there was a need to run cross validation algorithm taking into account more parameters, Randomized Search was used instead.

A further issue encountered was the slowness of computation due to a lack of parallelization: n_jobs parameter had to be set at 1 in order to successfully run the code which significantly slowed down the process as it couldn't be parallelized.

Finally, accuracy and loss are not the only measures of success of a neural network model. Further on, this same analysis could be deepened by considering F1 score, Precision and Recall as well in determining the optimal model for the task.

7 Conclusion

Regarding the first assignment, while both the Feed Forward Neural Network and Convolutional Neural Network are high quality classification algorithms, CNN brought more successful results. This was expected as it is known that the CNN are one of the most successful machine learning algorithms for image classification. The team believes that with more resources (most importantly time and GPUs), a CNN model could be trained that significantly surpasses the accuracy achieved so far. Data Augmentation would be a beneficial method for diminishing overfitting which was experienced while training the model.

With respect to the second assignment, the residual neural network provided interesting results. As with the CNN, it is assumed that with more resources such as time and processor power, the ResNet could further be optimized by using grid search cross validation with the goal of discovering the optimal parameters to increase accuracy.

References

[1] Qian, Ning. *On the momentum term in gradient descent learning algorithms*. Neural networks, 12(1):145–151, 1999.

[2] Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. *Learning representations by back-propagating errors*. nature, 323(6088):533, 1986.

[3] Simonyan, Karen and Zisserman, Andrew. *Very deep convolutional networks for large-scale image recognition*. In ICLR, 2015.

[4] Sutskever, Ilya, Martens, James, Dahl, George E, and Hinton, Geoffrey E. *On the importance of initialization and momentum in deep learning*. on Machine Learning (ICML), 28:1139–1147, 2013.

[5] Y. You, I. Gitman, and B. Ginsburg. *Large batch training of convolutional networks*. arXiv preprint arXiv:1708.03888v3, 2017.

[6] M. Aziz, S. Yahya, H. A. F. Almurib and C. L. Seow. *Improved Architecture of the Feedforward Neural Network for Image Recognition*. IEACon 2016 - 2016 IEEE Industrial Electronics and Applications Conference.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. arXiv:1512.03385, 2015.

[9] A. Krizhevsk, I. Sulskever, and G. E. Hinton. *Imagenet classification with deep convolutional neural networks*. Neural Information Processing Systems, 25:1097–1105, 2012.