Advanced Machine Learning
Convolutional Networks

Theodoros Sofianos (1867968)   Katsiaryna Zavadskaya (1847985)

April 2020

1 Implement a Convolutional Neural Network

1.1 Network architecture

We are starting from the implementation of the five-layered convolutional neural network. The network consists of the five blocks, where each of the blocks is composed by convolution, maxpooling and ReLU operations. After training our network for 20 epochs we got 78.4 % validation accuracy. The loss and accuracy curves can be seen in the Figure (1).

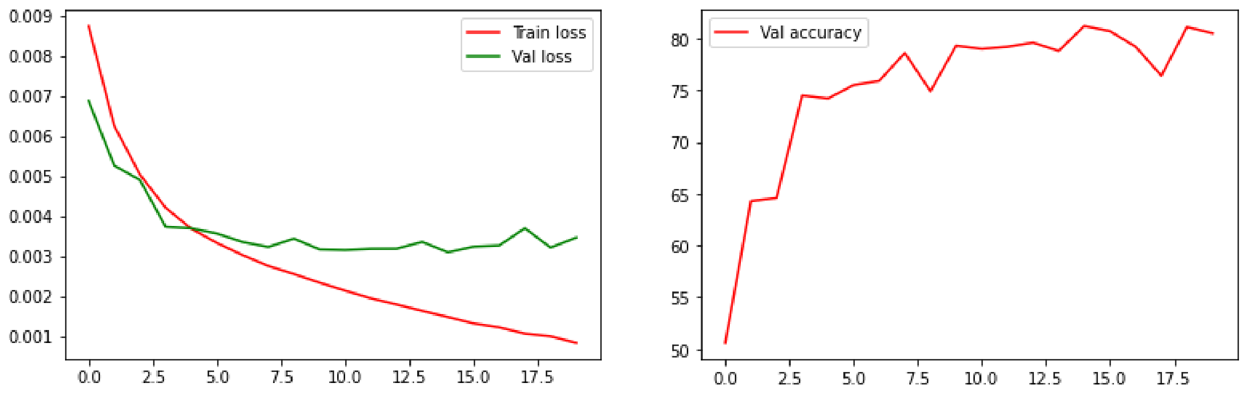

Figure 1: Simple five-layered model

1.2 Model Size

The total number of parameters of the network is: 7.678.474.

1.3 Visualization of the Filters

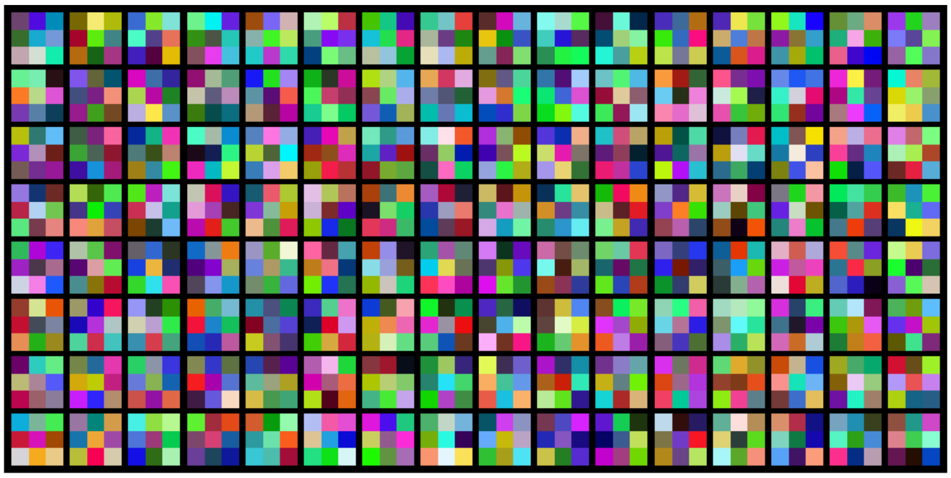

Figure 2: Weights of 1st conv layer before training

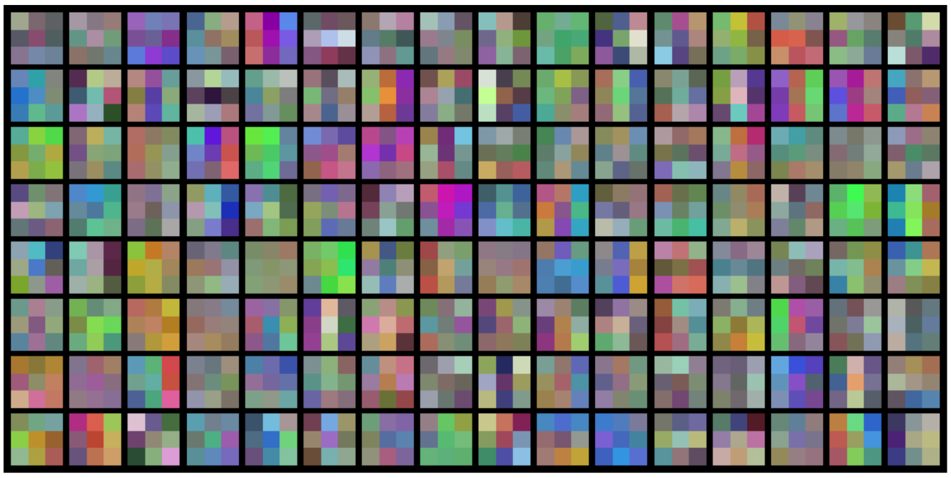

Figure 3: Weights of 1st conv layer after training

We can see that in the Figure (2), the filters are randomly initialized, so they are just 3x3 kernels of random colors and they all look similar to each other. Also, they do not seem to be able to capture the edges of an image. On the other hand, the filters after being trained (Figure 3) are all different with each other. These filters seem capable of understanding the colors of an image, since they do not have just random colors. They are mostly green and blue, something that perhaps could be explained by the fact most of the CIFAR-10 objects have background either trees-grass (the animals) or sea-sky. They are also more blurry and they try to capture the edges and some basic shapes,such as horizontal and vertical lines. As we know, the convolution filters in the first layers tend to capture the low-level features of an image, such as color and horizontal/vertical edges, while the high-level features are captured by the last convolution layers.

2 Improve training of Convolutional Networks

2.1 Batch normalization

In order to improve performance of our network, inside each of the blocks we added batch normalization right after convolution operation. The accuracy we got is 81.2 % comparing to 78.4 % of the simple model. By checking accuracy figures of the simple (Figure 1) and current (Figure 4) models, we see that model with batch normalization is having better results taking same time. While simple model is achieving it's top accuracy around 14th epoch; our current model needs 16 epochs to show higher performance.

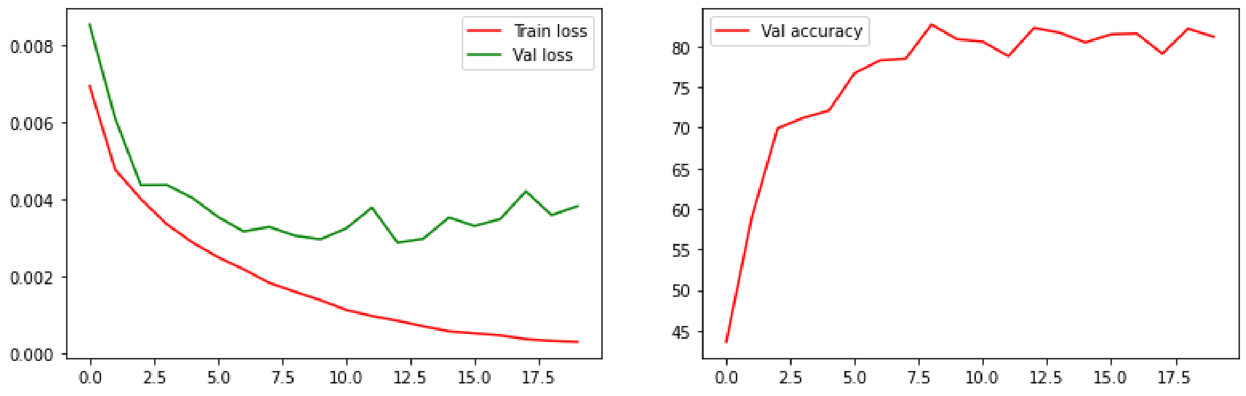

Figure 4: Simple five-layered model with batch normalization

2.2 Early stopping

Early stopping was used in order to prevent overfitting of our model. We compare the difference in train and validation losses of Simple model and Simple model with batch normalization trained for 50 epochs with losses of these two models with applying early stopping technique.

As we can see from the Figures (5) and (6), when we train Simple model for 50 or 13 epochs, we receive very similar validation accuracy. However in case of 50 epochs, the overfitting of the model is clearly seen, since training loss becomes 0, while validation loss grows. By using early stopping we can find a point where our model starts to overfit. Thus, putting Figures (5) and (6) next to each other, we see that, in fact, for the Simple model with it's current capacity, the number of epochs to find a good fit is equal to 13.

The same trend is visible on the Figures (7) and (8). We again see overfitting of the model while training for 50 epochs and Figure (8) shows breaking point of the learning process – 16th epoch.

Note that the graphs are showing 4 more epochs, since our early stopping was implemented in a way that model trains a bit more to confirm that 4 consecutive epochs are not improving validation accuracy, before its training stops. At the same time, only the model until the epoch with the highest validation accuracy is saved.

In summary, early stopping is a powerful regularization technique, which allows us to alleviate overfitting by reducing training epochs, but maintaining good validation accuracy.

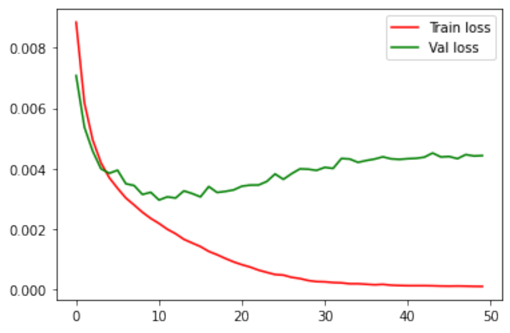

Figure 5: Simple model no early stopping.
Validation accuracy 79.9 % after 50 epochs

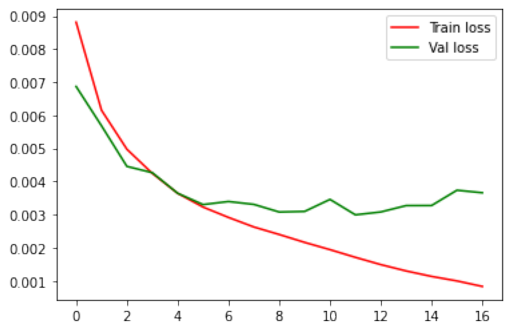

Figure 6: Simple model with early stopping at epoch 13.
Validation accuracy 80.4 % after 13 epochs

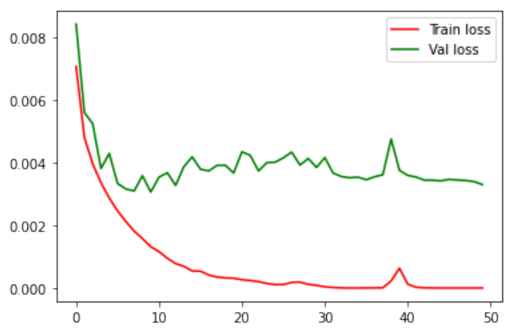

Figure 7: Simple model with BN no early stopping
Validation accuracy 83.0 % after 50 epochs

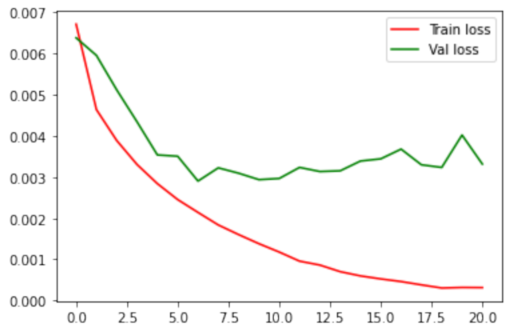

Figure 8: Simple model with BN and early stopping
Validation accuracy 82.6 % after 16 epochs

3 Implementing the feedforward model

3.1 Data augmentation

Data augmentation is an important technique that helps us increase the training size but also avoid overfitting, because by rotating, flipping or cropping an image, the network learns to not just focus on specific pixel locations of the original input image, but rather the image as a whole, since the network will use the same filters for the original and the transformed images. Also, with color-jittering and grayscale transformations, the network avoids to focus on colors in order to classify objects, but instead will try to capture the edges and shapes. However, CIFAR-10 dataset has some homogeneous characteristics, such as that the images are all taken in daylight and the object is always centered. This is why we tried to perform some 'mild' data augmentation techniques, focusing mainly on the color transformation of the original training images. A synoptic table of all our experiments is presented below. Note that the network we implemented works for 32x32 images, so we will maintain these dimensions and we could only change the scale of the original image or add padding. All models were trained at most for 30 epochs, with early stopping activated. We experimented with single but also composed transformations. Judging by the results, the best data augmentation technique for our model is the experiment number 8, which performs a rotation of at most 15 degrees on both axis, a translation of the original image pixels by a factor of 0.1 for both height and width, plus color jittering, with brightness and contrast ranging by a small factor(0.2 or 0.3), plus or minus, from the ones of the original image. This model actually improves the model implemented in Question 2 by almost 3%. The rest of the data augmentation techniques do not help the model improve significantly, actually with some of them the validation accuracy decreases.

Experiment	**Method(s)**	**Parameters**	**Validation Accuracy**
1	RanomResizedCrop	size=[32,32],scale=(0.5,1)	81%
2	RanomResizedCrop	size=[32,32],scale=(0.8,2.5)	81.8%
3	RanomResizedCrop	size=[32,32],scale=(1.5,3)	79.3%
4	ColorJitter	brightness=(0.7,2),contrast=(0.8,1.6)	80.7%
5	Grayscale	number of output chanels=3	77%
6	RandomAffine	degrees=(10,40),translation=(0.1,0.1),scale=(0.8,1.2)	81.4%
7	RandomAffine + ColorJitter	degrees=(10,10),translation=(0.1,0.1),scale=(0.9,1.2) brightness=(0.7,1.5),contrast=(0.8,1.4)	78.9%
8	RandomAffine + ColorJitter	degrees=(15,15),translation=(0.1,0.1) brightness=(0.7,1.3),contrast=(0.8,1.3)	84.1%
9	ColorJitter + RandomAffine	brightness=(0.7,1.3),contrast=(0.8,1.3) degrees=(15,15),translation=(0.1,0.1)	80%
10	RandomAffine	degrees=15,translate=(0.1,0.1)	82.6%
11	RandomGrayscale	p=0.1	81.1%
12	RandomRotation	degrees=(-60,60)	80.0%

Table 1: Data Augmentation techniques and results

3.2 Dropout

In order to improve generalization of the model we use dropout technique. By varying a probability parameter of dropping the input activations — p in the range $[0.1, 0.9]$, we train our model, using the same p for all the 5 layers. The data augmentation from the previous subsection was disabled in order to compare with baseline model (p=0).

As we can see from the accuracy curves (Figure 9), all the high dropout values $p \in [0.6, 0.9]$ decrease model performance comparing with baseline model. The three values of p, which are beneficial for our network are $\{0.1, 0.2, 0.3\}$. All of them end up giving the same accuracy around 84.5 %. We expect that with p being a small number, the model might overfit. However, none of the configurations are overfitting, which is visible from the train (Figure 13) and validation (Figure 14) losses. Since three values $\{0.1, 0.2, 0.3\}$ give us similar accuracy, we picked middle variant and established dropout rate of our model to be $p = 0.2$.

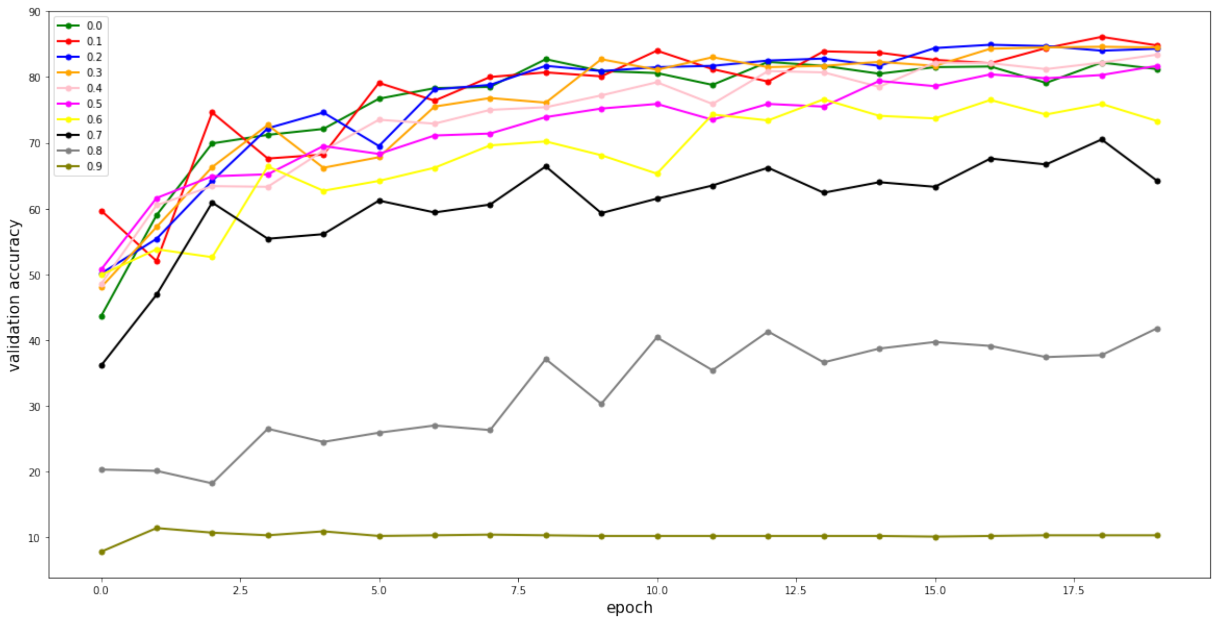

Figure 9: Validation accuracies for different values of dropout $p \in [0.0, 0.9]$

4 Implementing the feedforward model

4.1 Exploitation of VGG_11_bn model

Accuracy of the network is 61 %. The loss and accuracy curves can be seen in the Figure (10).

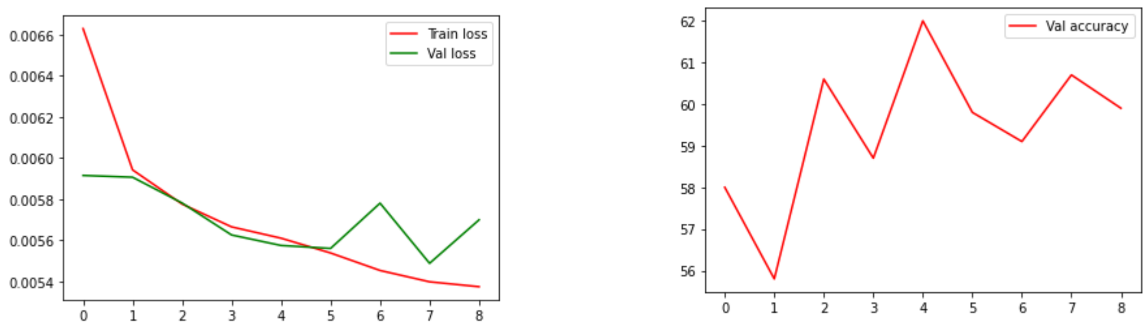

Figure 10: Results of Question 4.a. Best validation accuracy was 62%, before early stop activation

4.2 Fine-tuning the network

Below are the results of the 2 fine tuned models, with and without weights initialization from ImageNet respectively. It is not surprising that the model that initializes its weights with ImageNet performs much better over 30 epochs, since VGG 11 was trained on around 14 million images from the ImageNet, so even if there is a domain-shift and we need to train more than just a couple of dense layers, the weights of the conv layers of Imagenet is a much better starting point rather than a random initialization. It is almost impossible to replicate the results of VGG 11 or any other model trained on ImageNet by copying the same architecture and initializing the weights from scratch, since these models were run on high-end,distributed computing hardware. That is the whole point of using pretrained models, since not everyone has the resources to train such deep networks over huge datasets.

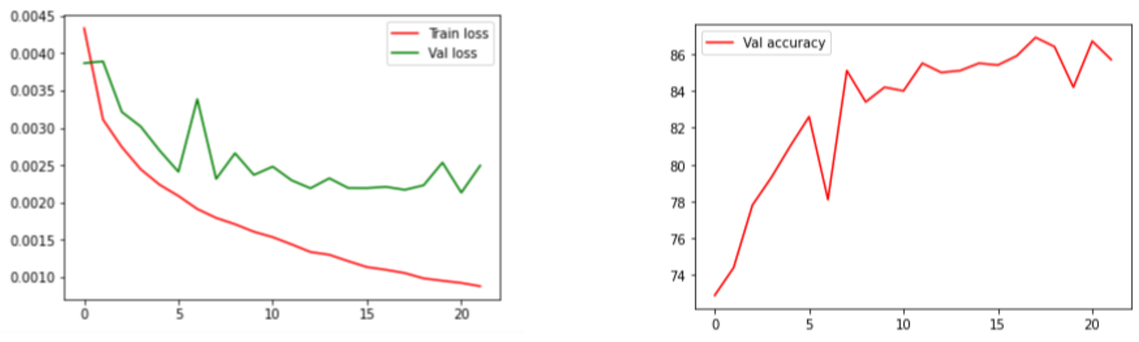

Figure 11: Results of the model when fine tuned and initialized with ImageNet pretrained weights. Accuracy of the network is 86.0 %. Note that the validation accuracy is 86.9%, before early stopping was activated

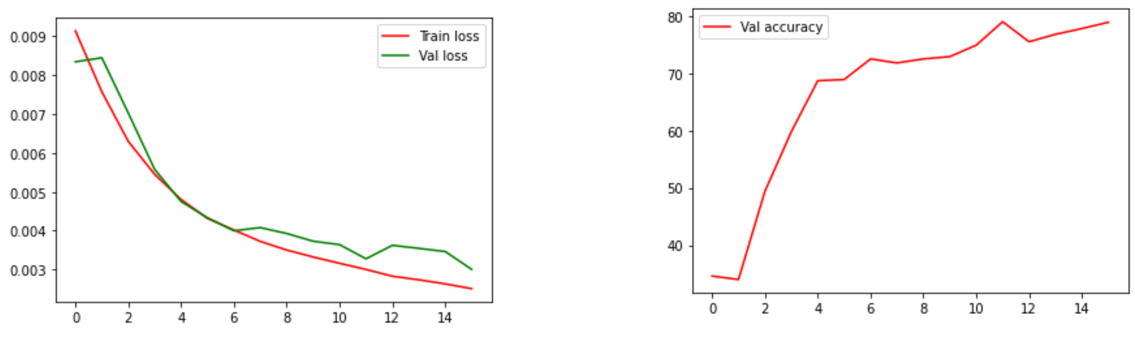

Figure 12: Results of the model when fine tuned, without being initialized with ImageNet pretrained weights. Accuracy of the network is 79.6%. Note that the validation accuracy is 79%, before early stopping was activated

5 Appendix

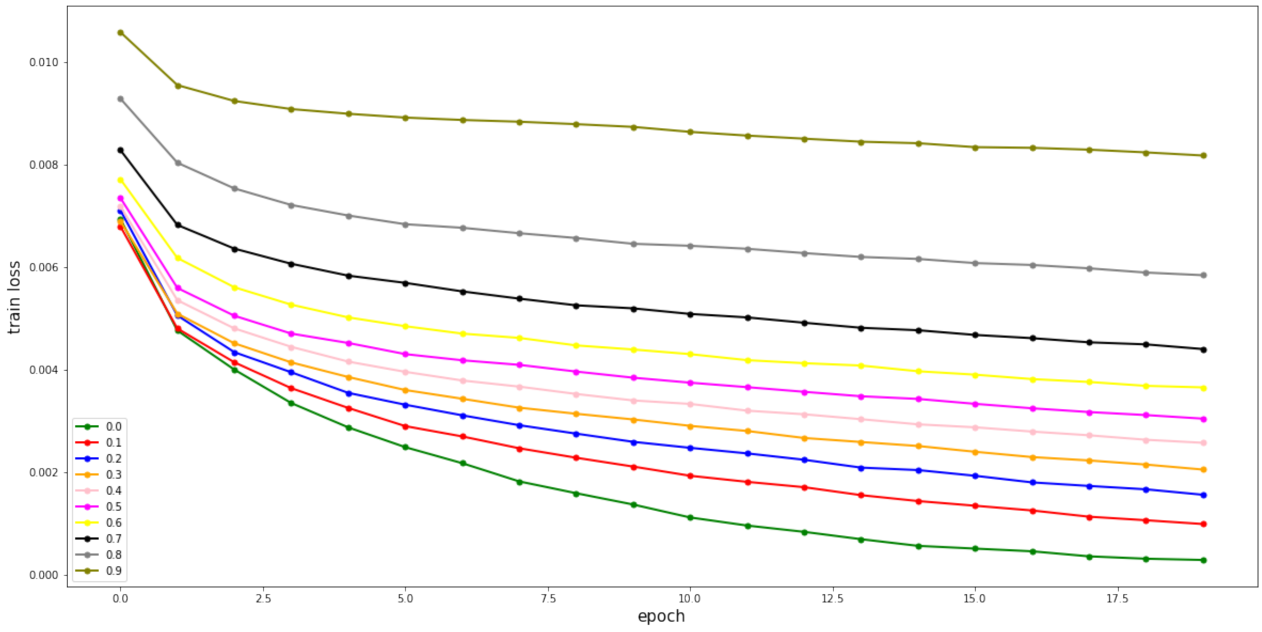

Figure 13: Train losses for different values of dropout $p \in [0.0, 0.9]$

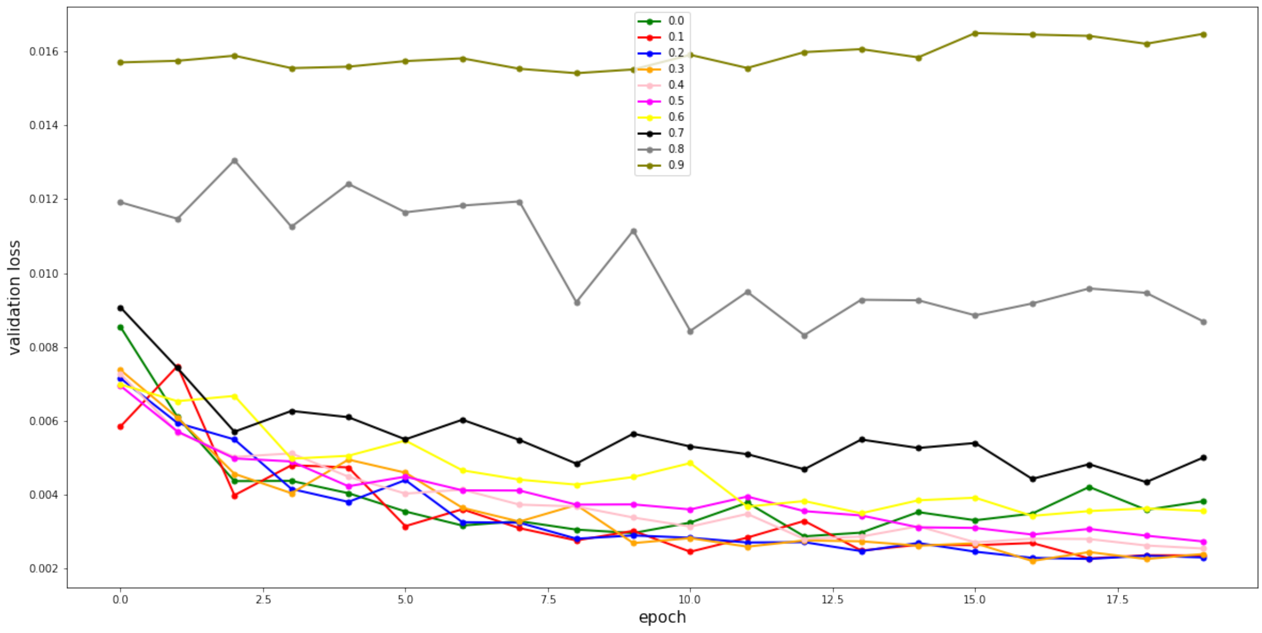

Figure 14: Validation losses for different values of dropout $p \in [0.0, 0.9]$