# Coding assignment 1. Based on “Deep Residual Learning for Image Recognition” by K. He, X. Zhang, S. Ren, J. Sun.

**Report by Golobokov Mikhail**

Based on the paper:

K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR),2016

Required Task:

- The basic training and testing pipeline
  - Run the network with 20, 56 and 110 layers on CIFAR10 and CIFAR100
  - Pay attention to the hyper-parameters (learning rate, epochs, etc.)
- Questions that should be answered in the report
  - Paste complete training and testing curves and the final accuracy
  - How are your results compared to the paper? Why better or worse?
  - How is performance changing with the number of network layers? Why?
  - Any significant features that can be recognized in the curves?
  - What is the major difference between CIFAR10 and CIFAR100 results?

Optional Task 1:

- Changing hyper-parameters
  - Based on the results of the basic (required) experiments
  - How does the change of hyper-parameters impact final performance?
- Questions to be discussed in the report
  - What if we multiply the base learning rate by 10, 5, 2, or by 1/10, 1/5, 1/2?
  - What if we double the number of training epochs? What if we half it? Note that the learning rate policy should be adjusted accordingly (please specify details)
  - What if we only use 1/2 or 1/5 of training data? What if we double or half the size of mini-batch? Note that for fair comparison, you need to keep the number of training samples (iterations x batchsize) unchanged

Hardware:

- RAM: 16 Gb;
- GPU: NVidia GTX 1080Ti;

- CPU: AMD Ryzen 2700x.

## ABSTRACT

The article on "Deep Residual Learning for Image Recognition" presents a new approach to neural network learning. This approach solves the problem of degradation of deep networks in learning. In this homework assignment, I will explore the effect of basic neural network settings and hyperparameters on the performance, quality, and speed of the deep residual neural network.

## 1. RESNET – WHAT IS IT?

ResNet — this is an abbreviated name for Residential Network, but what is residual learning?

Deep convolutional neural networks surpassed the human level of image classification in 2015. Deep networks extract low -, medium -, and high-level features in a cross-layered manner, and increasing the number of stacked layers can enrich feature "levels".

When a deeper network begins to collapse, there is a problem: as the depth of the network increases, the accuracy first increases and then rapidly deteriorates. The decrease in training accuracy shows that not all networks are easy to optimize.

Let's formulate the problem so that the deeper levels predict the difference between what the previous lines give out and the target, that is, they can always take the weights to 0 and just skip the signal.

Hence the name-Deep Residual Learning, that is, we learn to predict deviations from past lines.

More specifically, it looks like this. The main building block of the network is this design:

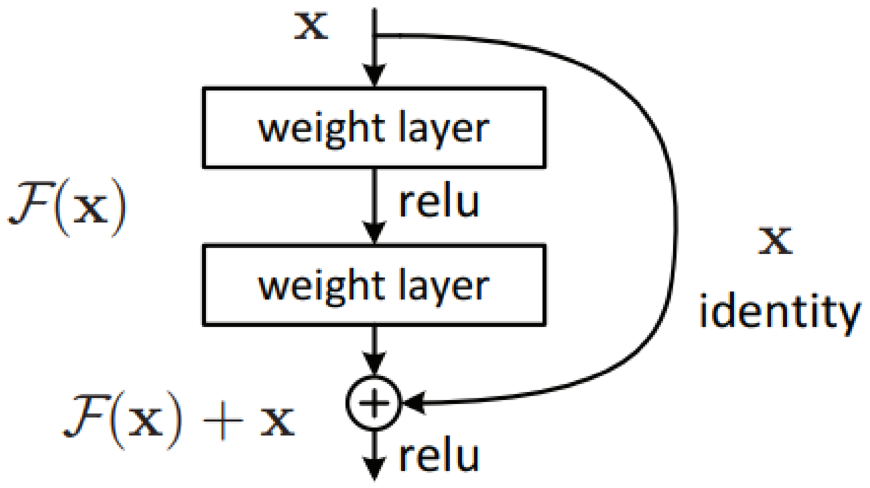

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

*Figure 1 - Residual learning: a building block*

Shortcut connections skip one or more layers and perform identity mapping. Their outputs are added to outputs of the stacked layers. Using ResNet, you can solve many problems, such as:

- ResNet is relatively easy to optimize: "simple" networks (which simply stack layers) show a large learning error when depth increases.
- ResNet makes it relatively easy to increase accuracy by increasing depth, which is more difficult to achieve with other networks.

## 2. RESNET ARCHITECTURE

Classic 34-layer network (Fig. 2, center):
Simple baselines are mostly inspired by the philosophy of VGG networks Convolutional layers basically have 3×3 filters and follow two simple rules:

1. For the same output feature map, layers have the same number of filters;
2. If the size of the feature map is halved, the number of filters is doubled to preserve the temporal complexity of each layer.

It is worth noting that the ResNet model has fewer filters and less complexity than VGG networks.

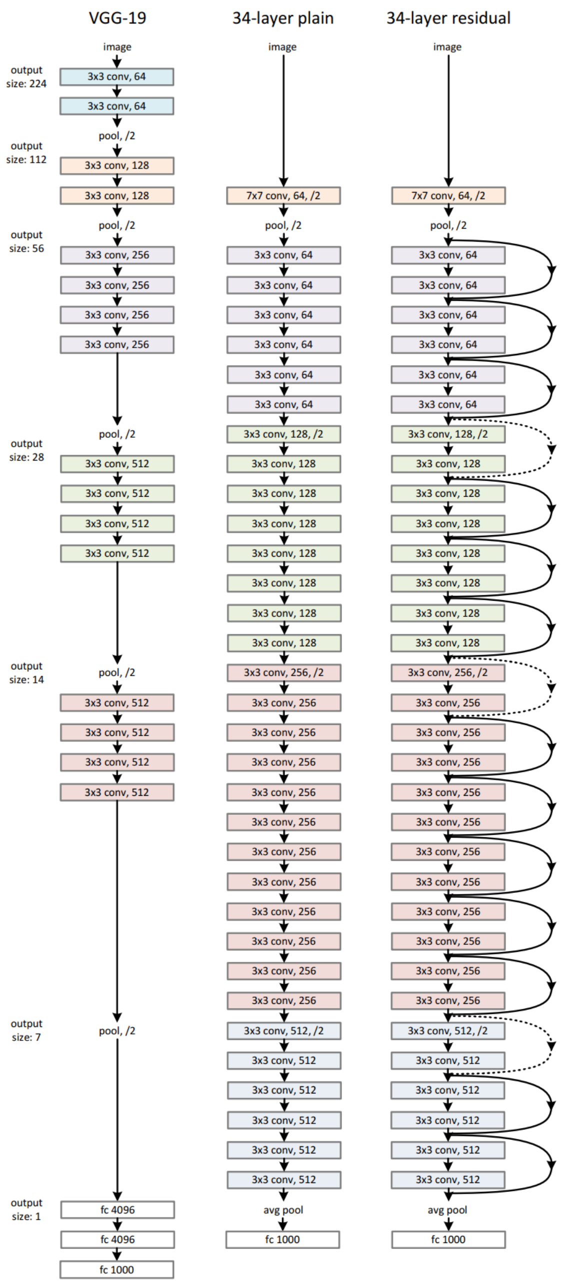


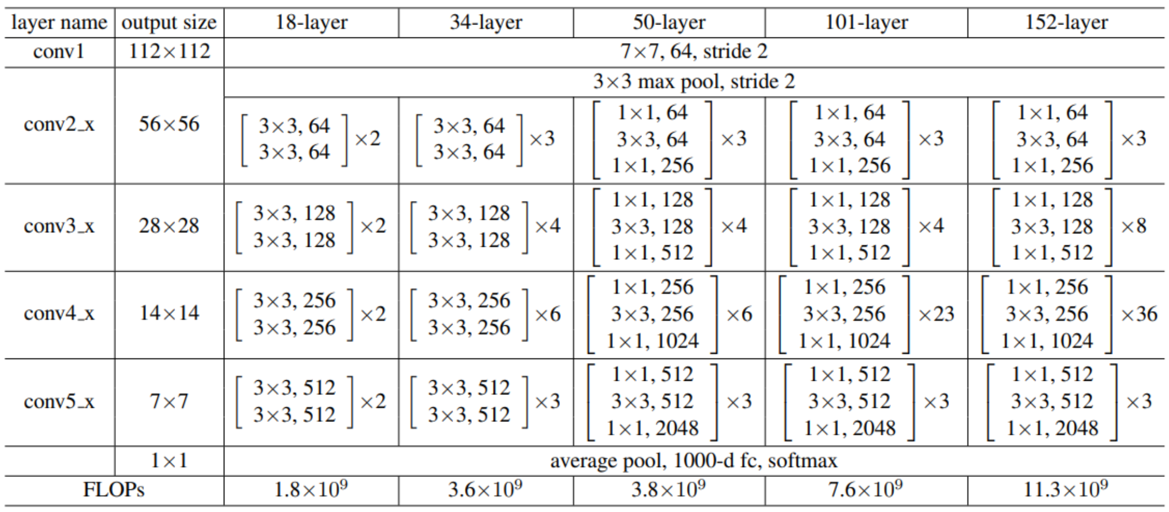
*Figure 2 - Example network architectures for ImageNet*

ResNet: based on the simple network described above, a fast connection has been added (Fig. 2, right), which turns the network into a residual version of it. Identification fast connections F (x {W} + x) can be used directly when the input and output have the same dimensions (solid line fast connections in Fig. 2). When dimensions increase (dotted lines in Fig. 2), it considers two options:

Quick connect performs identity mapping with additional zeros added to increase dimension. This option does not enter any additional parameters.

The fast join projection in F (x {W} + x) is used to map dimensions (done with 1×1 convolution).

For any of the options, if quick connections are made on two-dimensional feature maps, they are performed in step 2.



| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | | |
| conv2_x | 56×56 | 3×3 max pool, stride 2 | | | | |
| | | $\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$ |
| conv3_x | 28×28 | $\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$ | $\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$ |
| conv4_x | 14×14 | $\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$ | $\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$ |
| conv5_x | 7×7 | $\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$ | $\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ | $\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

*Figure 3 - Table 1. Architectures for ImageNet*

Each ResNet block has two levels of depth (used in smaller networks such as ResNet 18, 34) or 3 levels (ResNet 50, 101, 152).

50-layer ResNet: each 3-layer block is replaced in a 34-layer network by this 3-layer bottleneck, resulting in a 50-layer ResNet (see Table above). They use option 2 to increase the dimensions. This model has 3.8 billion FLOPs.

ResNet with 101 and 152 layers: they create ResNet with 101 and 152 layers using more 3-layer blocks (see Table above). Even after increasing depth, 152-layer ResNet (11.3 billion FLOPS) has less complexity than VG-16/19 networks (15.3 / 19.6 billion FLOPs).

## 3. RESEARCH THE EFFECT OF THE NUMBER OF LAYERS (NETWORK DEPTH) ON THE RESULTS OF THE NETWORK (ACCURACY)

Required Task:

- The basic training and testing pipeline
  - Run the network with 20, 56 and 110 layers on CIFAR10 and CIFAR100
  - Pay attention to the hyper-parameters (learning rate, epochs, etc.)
- Questions that should be answered in the report
  - Paste complete training and testing curves and the final accuracy
  - How are your results compared to the paper? Why better or worse?

- How is performance changing with the number of network layers? Why?
- Any significant features that can be recognized in the curves?
- What is the major difference between CIFAR10 and CIFAR100 results?

The basic task required to run the network with 20, 56 and 110 layers on CIFAR10 and CIFAR100 3-times average. The main idea of this task is to explore the influence of network depth on its final accuracy.

Parameters of neural network was as follows:

- Architecture: ResNet;
- Layers: 20, 56, 110;
- Epochs: 350;
- Schedule: 81, 122;
- Gamma: 0.1;
- Weight decay: 1e-4;
- Learning rate: 0.1;
- Batch size: 128;
- Momentum: 0.9.

The learning rates for schedule were the same for the whole research the effect of the number of layers (network depth) (Fig.3):

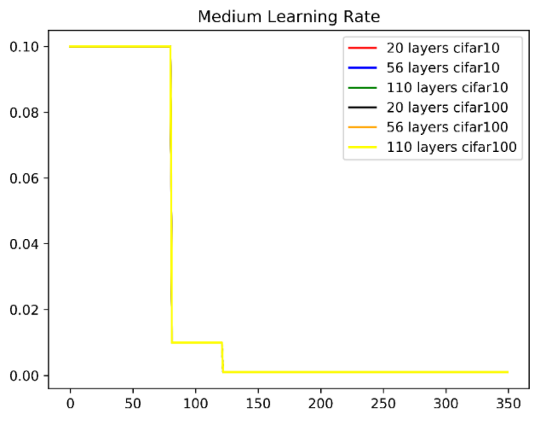


*Figure 4 - Medium learning rate for 20, 56, 110 layers NN using CIFAR 10, 100*

*Horizontally: training epochs; Vertically: learning rate*

Len’s speak about accuracy. The results of my experiments are shown in Fig.5 and Fig.6 accuracy on the training set and on the test set respectively.

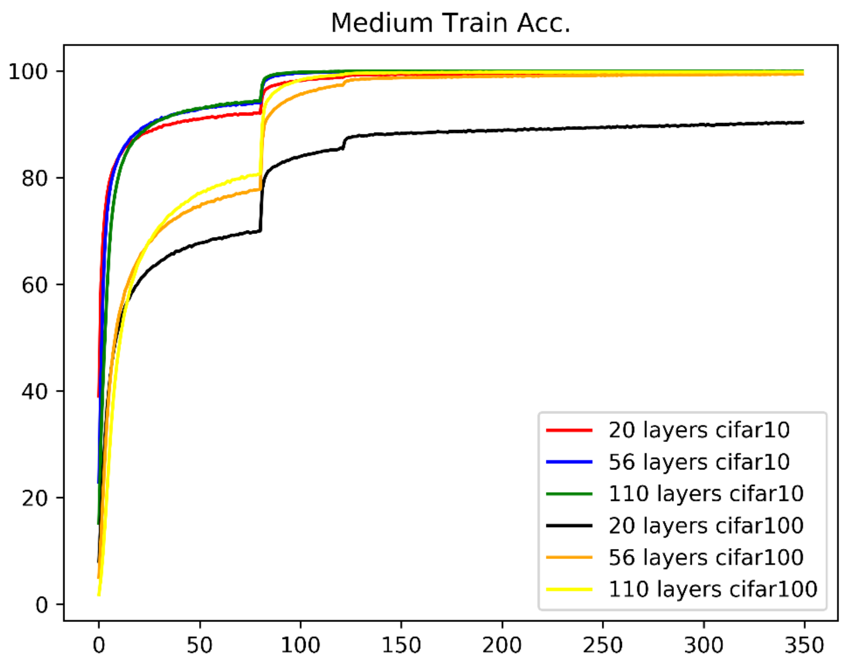


*Figure 5- Medium training accuracy for 20, 56, 110 layers NN using CIFAR 10, 100*

*Horizontally: training epochs; Vertically: accuracy*

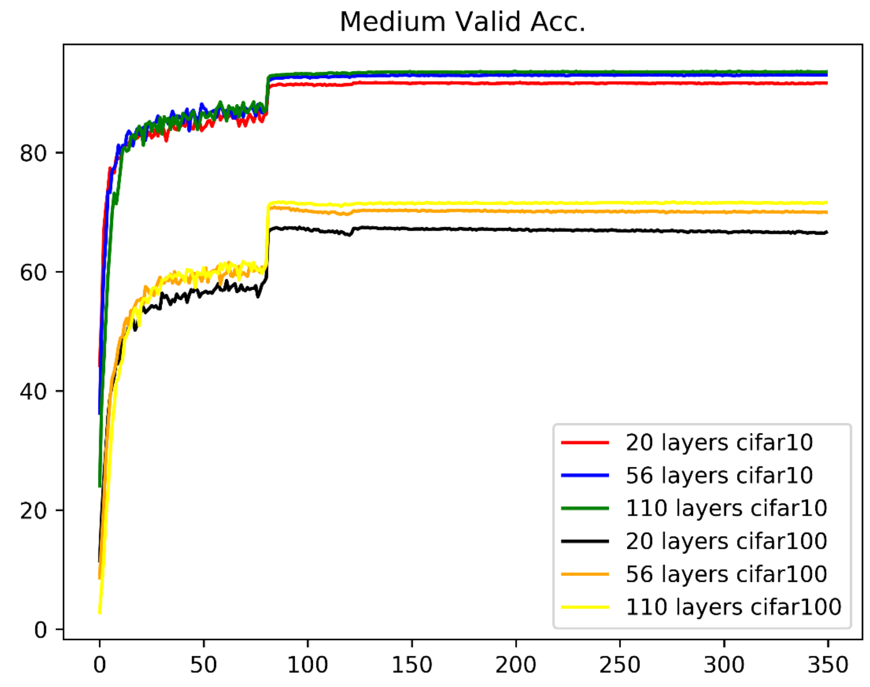


*Figure 6 - Medium valid accuracy for 20, 56, 110 layers NN using CIFAR 10, 100*

*Horizontally: training epochs; Vertically: accuracy*

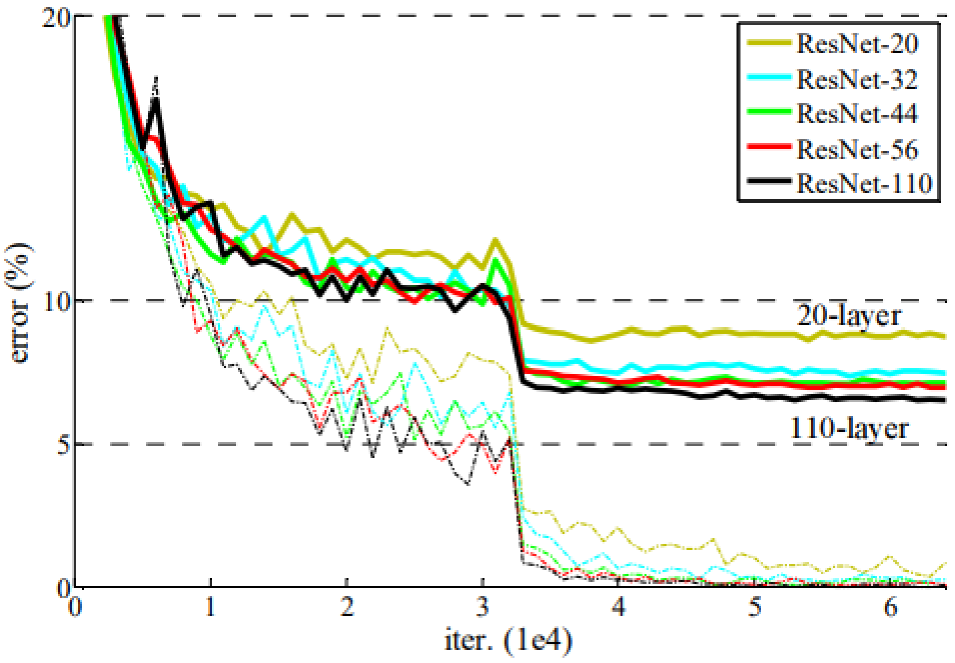


*Figure 7 - Training on CIFAR-10 (from paper)*

*Dashed lines denote training error, and bold lines denote testing error*

As we can see, the achieved results are nearly the same as the author's in the given paper. As you can see in the graphs, after the number of epochs exceeded 100 pieces-we no longer observed rapid growth or strong changes in the accuracy of the network. Over the next 250 epochs the result varied within no more than 1% accuracy. We can see it in table 1.

*Table 1 – Error rate on the test sample on CIFAR-10 (from 90 to 350 epochs)*

| Layers | Min error, % | Max error, % | Avg. error, % | Paper error, % |
|---|---|---|---|---|
| 20 | 8.22 | 8.77 | 8.39 | 8.75 |
| 56 | 6.9 | 7.47 | 7.05 | 6.97 |
| 110 | 6.36 | 6.91 | 6.5 | 6.43 |

The same story is with the results on CIFAR-100. The error rate is no more than 1,5% accuracy.

*Table 2 – Error rate on the test sample on CIFAR-100 (from 90 to 350 epochs)*

| Layers | Min error, % | Max error, % | Avg. error, % | Paper error, % |
|---|---|---|---|---|
| 20 | 32.53 | 33.88 | 33.06 | - |
| 56 | 29.28 | 30.41 | 29.88 | - |
| 110 | 28.31 | 29.04 | 28.49 | - |

Let's speak about the results:

The worst results between other number of layers is 20 layers. The depth of such network is really low. For the example we can see some problems with training 20-layers network on CIFAR-10 dataset in Fig.5: 20 layers is not enough, the number of features which network might to learn is really high and that is the reason to increase the number of layers to 56 or 110 layers. The differences between 56 and 110 layers are:

- The 110 layers network is 2 time bigger than 52 layers. So, the needed compute power is higher 2 times;
- 110 layers networks train faster and will get best result earlier (if we speak about the number of epochs);
- The difference between average 56-layers and 110-layers error rates for CIFAR-10 is 1.5%, for CIFAR-100 is 1.39%.

So, if we want to train CIFAR-10 dataset – it will be enough 20-layers network in case of less compute power, less params, fast learning (best choice for embedded devices).

If we needed to train CIFAR-100 it is better to use 56-layers network in case of it's nearly the same results while taking less computational power to train.

But if we are free of any restrictions – choose 110-layers network as the most accurate among these 3 options. If we'll use 1202-layers network, the results may be worse: the testing error even higher (7.93%) than that of the ResNet-110, which can be due to overfitting.

To sum up everything about results: achieved results are nearly the same as in the paper. It's harder to train CIFAR-100 in case of a smaller number of pictures per class than in CIFAR-10 (6000 pic per class in CIFAR-10 and 600 pics per class in CIFAR-100). The same problem is with CIFAR-100 test dataset. If we have less pictures per class – the generalization ability is getting lower, train harder (less accuracy).

## 4. THE INFLUENCE OF LEARNING RATE ON THE ACCURACY

Probably one of the most obvious hyperparameters that have a strong influence on the learning process and the final result is learning rate.

If the learning rate is too low, then even after training the neural network for a long time, it will be far from optimal results. On the other hand, if the learning rate is too high, the network will very quickly give answers (not the fact that it will be accurate answers).
The optimization problem of the model is dealt with by the SGD (Stochastic Gradient Descent) algorithm. The learning rate tells us how to change the weight of the network, so that it adapts to new conditions. When the network is just beginning to learn – you can use a sufficiently high learning rate to quickly converge to the

global minimum function. The need to use a large value of learning rate is caused by the fact that we need to "jump" local minimum and "fall" in the global. But to accurately localize the global minimum – we need to lower the rate of learning. Thus, our "jumps" become smaller and we can fall "lower" on the error curve. This is how this optimization model works.

So, let's speak about the results of the experiment with learning rate:

We train the ResNet with 20 layers on CIFAR-10 and CIFAR-100. All the hyper-parameters besides the learning rate was set as in the Required Part of this Coding Assignment. We have used six base learning rates for training: 0.01, 0.02, 0.05, 0.1, 0.2, 0.5 and 1, where 0.1 was the baseline.

We change the learning rate 2 times: on 82-th and 122-th epoch, accordingly to the preselected schedule. Each change was a multiplication of the current learning rate by a preselected hyper-parameter Gamma = 0.1. The results are represented in Fig.8 and Fig.9 and in table 3 and table 4.

The learning rates 0.05, 0.1, 0.2 and 0.5 are having nearly the same error rate. The learning rate value = 0.01 shows 2% more error accuracy. This one can say us that we needed in more iterations to find optimal weights or just change our learning rate: too slow, too little steps. The learning rate = 1 is absolutely awful: is 9,44% more error rate than the best accuracy one. That means that SGD can't find the optimal weights. The same thing is with CIFAR-100 results.

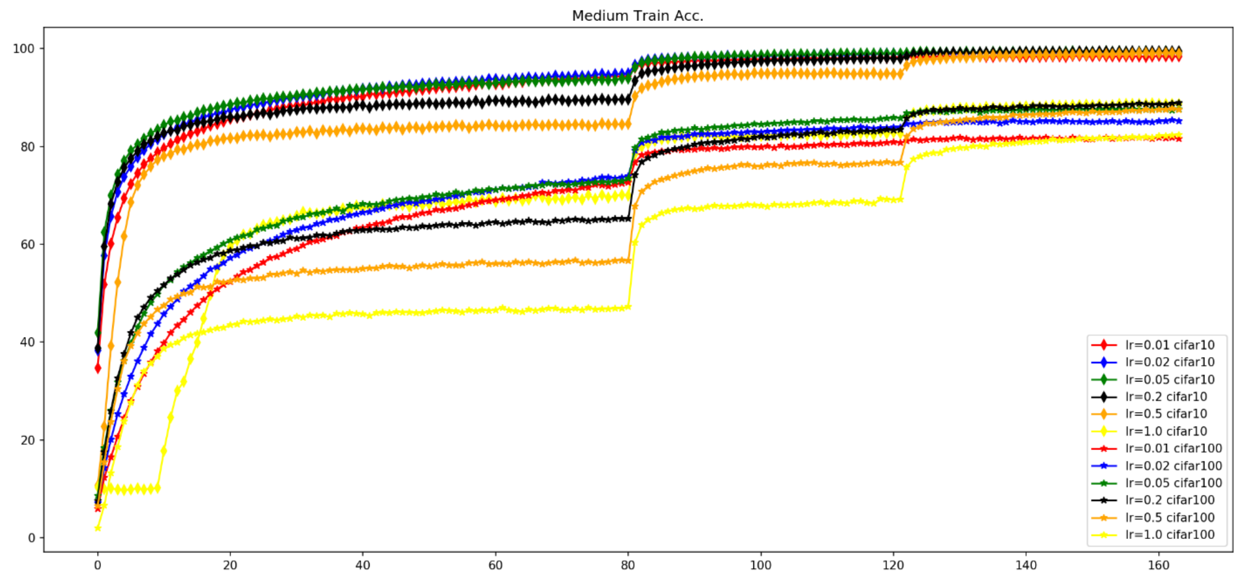


*Figure 8 – 20-layers ResNet different learning rate on train set*

*Horizontally: training epochs; Vertically: accuracy*

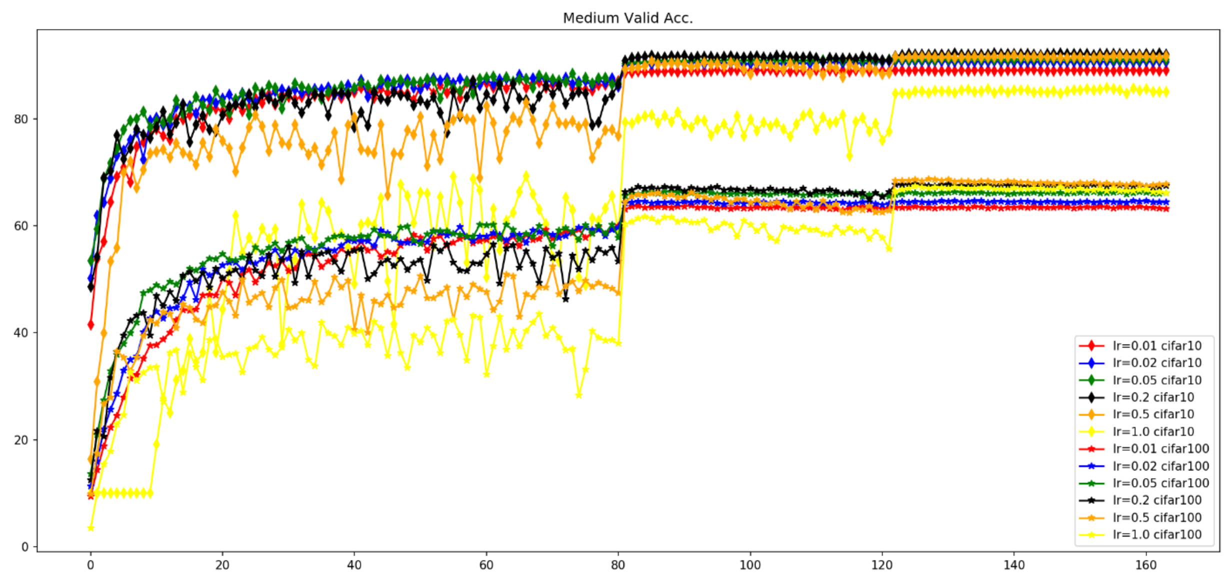


*Figure 9 – 20-layers ResNet different learning rate on test set*

*Horizontally: training epochs; Vertically: accuracy*

*Table3 – Error rate on the test sample on CIFAR-10 (from 90 to 160 epochs)*

| Learning rate | Min error, % | Max error, % | Avg. error, % | Paper error, % |
|---|---|---|---|---|
| 0.01 | 10.79 | 11.17 | 10.98 | - |
| 0.02 | 9.19 | 9.74 | 9.4 | - |
| 0.05 | 8.72 | 9.26 | 8.96 | - |
| 0.2 | 7.76 | 9.26 | 8.2 | - |
| 0.5 | 8.19 | 12.23 | 9.31 | - |
| 1 | 14.21 | 26.93 | 17.64 | - |

*Table 4 – Error rate on the test sample on CIFAR-10 (from 90 to 160 epochs)*

| Learning rate | Min error, % | Max error, % | Avg. error, % | Paper error, % |
|---|---|---|---|---|
| 0.01 | 36.16 | 36.99 | 36.61 | - |
| 0.02 | 35.34 | 36.06 | 35.59 | - |
| 0.05 | 33.59 | 34.54 | 33.94 | - |
| 0.2 | 32.18 | 35.04 | 32.91 | - |
| 0.5 | 31.27 | 37.58 | 33.62 | - |
| 1 | 32.6 | 44.32 | 36.44 | - |

## 5. THE INFLUENCE OF THE NUMBER OF TRAINING EPOCHS ON THE ACCURACY

The number of training epochs is a value, which means how many times the network will have full cycle of forward and backward propagations. In this experiment we want to know how the number of epochs and also schedule values influence on the accuracy of the network. Such parameters as the number of training epochs and schedule values are represented in Table 5.

*Table 5 – The number of training epochs and schedule values.*

| Number of epochs | Schedule values |
|---|---|
| 164 | 81, 22 |
| 82 | 41, 61 |
| 382 | 162, 244 |

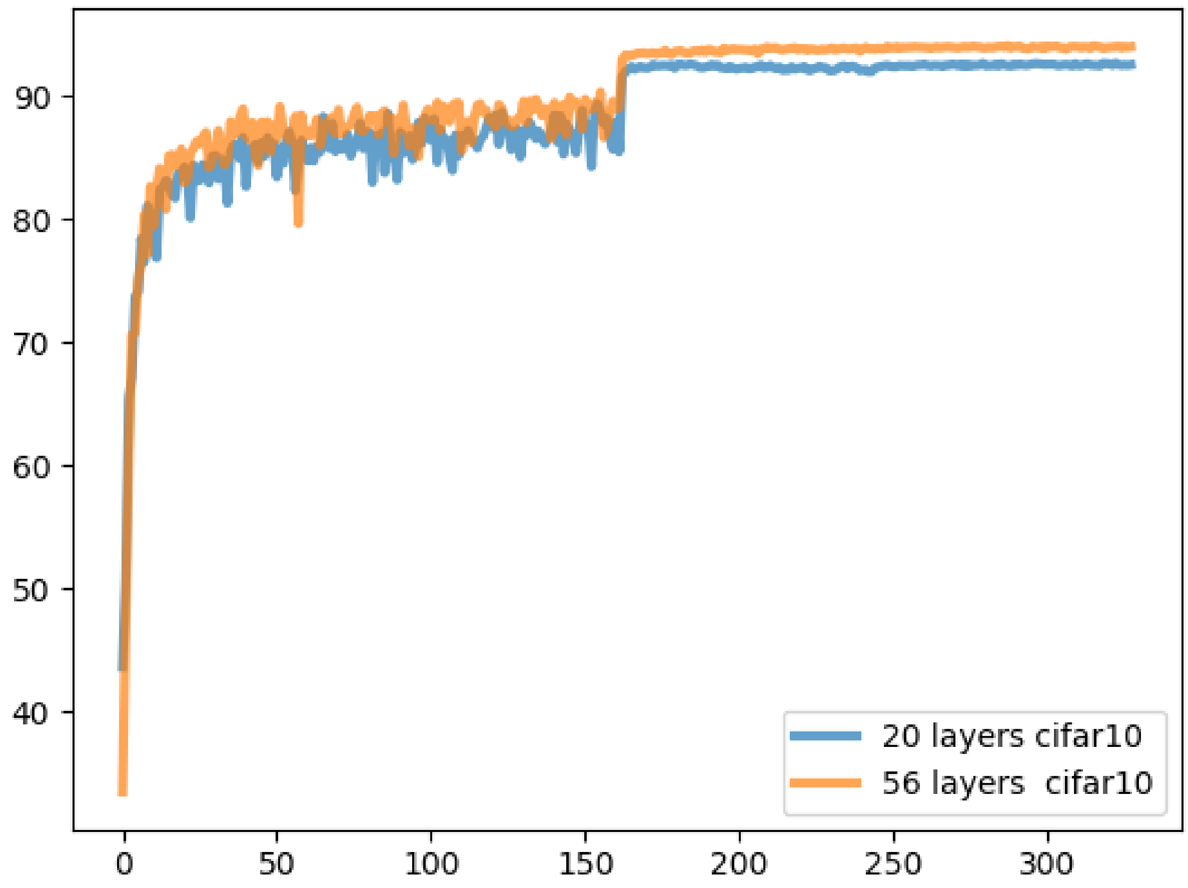


*Figure 10 - ResNet, 20 and 56 layers, CIFAR-10, 328 epochs train set*

*Horizontally: training epochs; Vertically: accuracy*

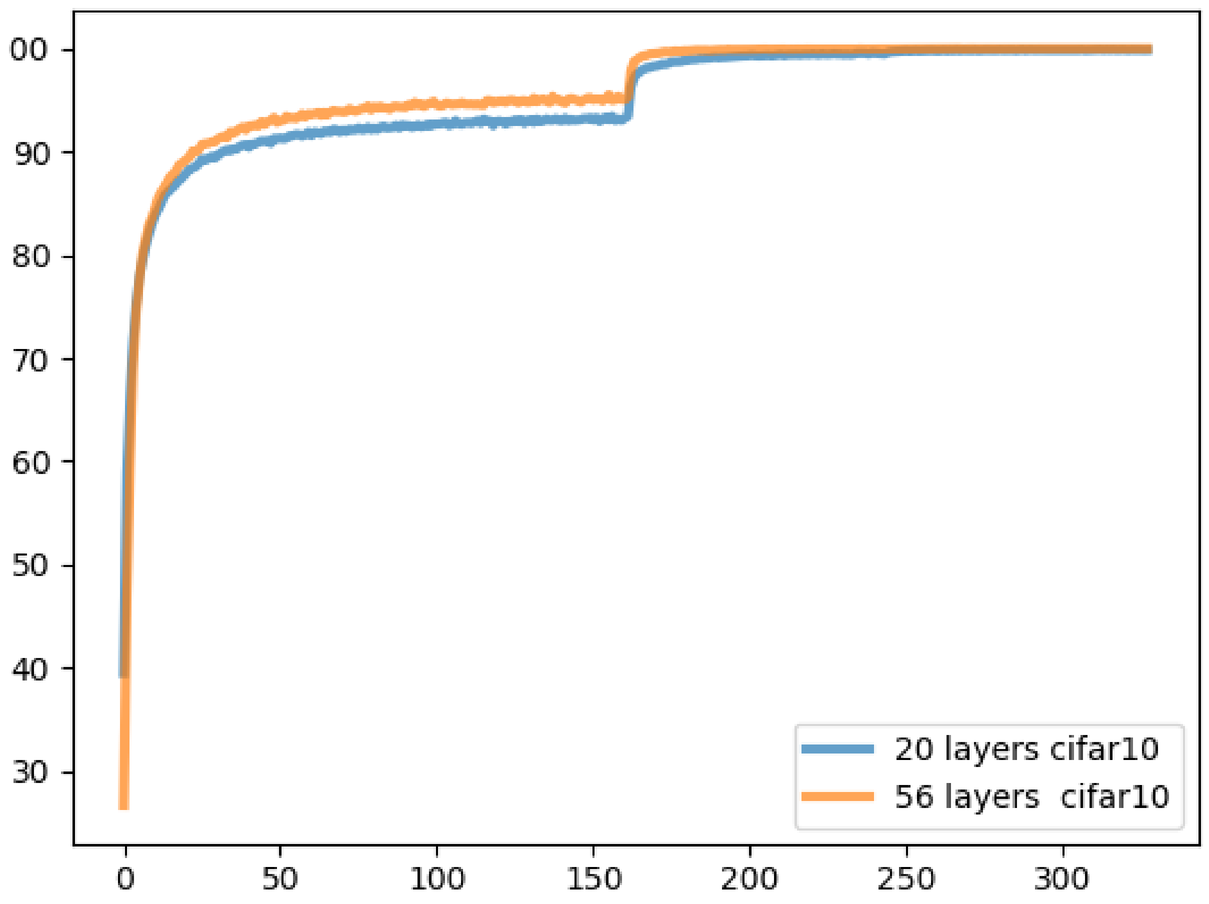


*Figure 11- ResNet, 20 and 56 layers, CIFAR-10, 328 epochs test set*

*Horizontally: training epochs; Vertically: accuracy*

Table 6 – Error rate on CIFAR-10 (164 and 328 epochs)

| Dataset | Number of layers | Number of epochs | Error training accuracy (%) | Error testing accuracy (%) |
|---|---|---|---|---|
| CIFAR-10 | 20 | 164 | 0.64 | 8.45 |
| | | 328 | 0.25 | 7.8 |
| | 56 | 164 | 0.09 | 7.3 |
| | | 328 | 0.01 | 6.1 |

If we double the baseline number of epochs – we will get the value 328. The error rate with this number of training epochs shows that the accuracy become better: 0.65% better accuracy on 20 layers network with 164 epochs and 1.2% better accuracy than 56 layers with 328 epochs network.

The main idea of changing number of epochs – get better generalization ability of the network. On the other side we will have a problem with a big number of epochs - we can retrain the model and the accuracy will begin to fall.

## 6. THE INFLUENCE OF THE DATASET'S SIZE ON THE ACCURACY

As you know, the number of pictures in datasets has a great influence on final results: more generalization ability if we have enough pictures per class, more accuracy, bigger resistance to noise. We choose full size and halfed-sized datasets of cifar-10 in this experiment. Also we choose 1/5 of CIFAR-100 dataset and full CIFAR-100 dataset. The results are in Fig.12 and Fig.13 and Table 7.

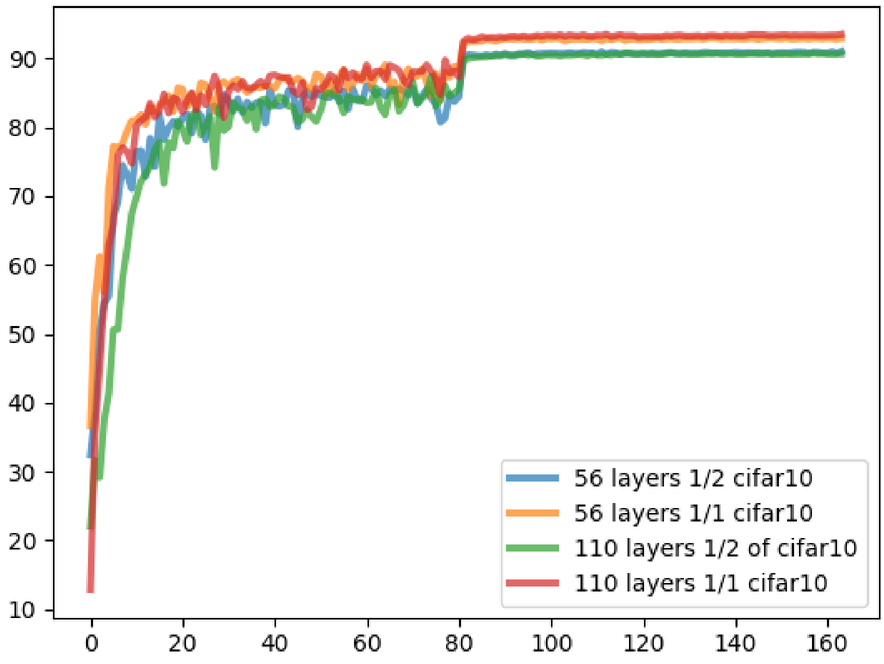


*Figure 12 – training set on different data*

*Horizontally: training epochs; Vertically: accuracy*

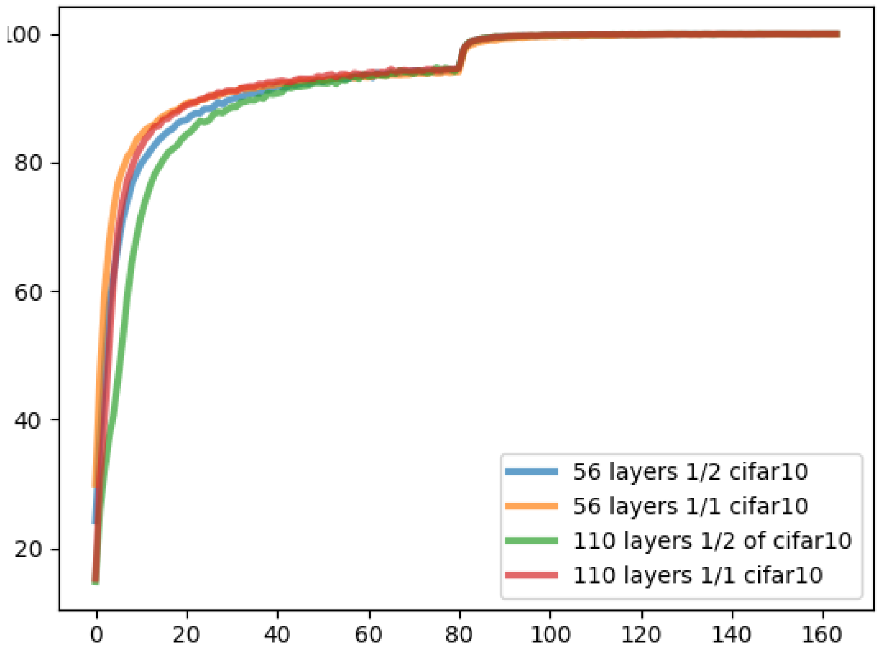


*Figure 13– test set on different data*

*Horizontally: training epochs; Vertically: accuracy*

Table 7 – Error rate on CIFAR-10 and CIFAR-100 (different data)

| Dataset | Number of data | Error testing accuracy (%) |
|---|---|---|
| CIFAR-10 | 1/2 | 9 |
| | 1 | 7.1 |
| CIFAR-100 | 1/5 | 51 |
| | 1 | 30 |

CIFAR-100 dataset is not so representable due to high computational power and less number of pics per class. Table 7 shows us that the difference between ½ of dataset and full datasets is about 1.9%. The size of dataset is a thing that could be a bottleneck for our network. More pics per class – more accuracy.

## 7. THE INFLUENCE OF THE BATCH SIZE ON THE ACCURACY

The batch size shows us how many pics would be used on each iteration (before we update weights). We were training the network using the default batch number – 128, also we take a half and double size. The results of the experiments are in Fig.14 and Fig.15.

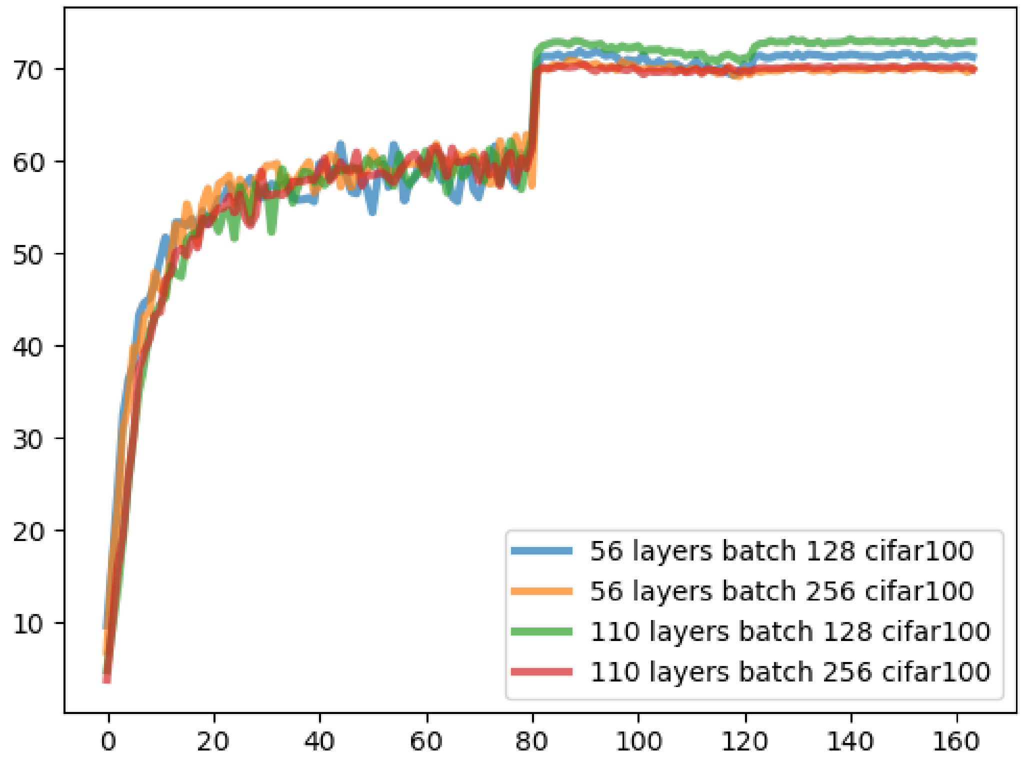


*Figure 14 - different batch size train set CIFAF-100*

*Horizontally: training epochs; Vertically: accuracy*

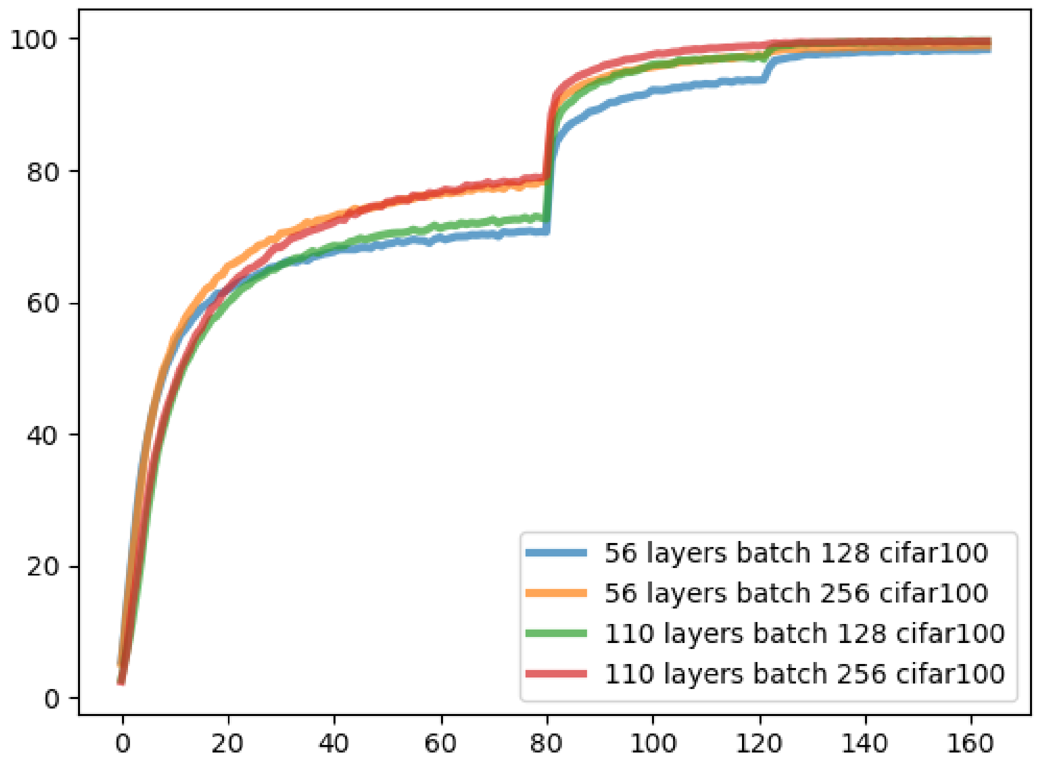


*Figure 15 - different batch size test set CIFAR-100*

*Horizontally: training epochs; Vertically: accuracy*

If we double the batch size – there is nothing change, only more computational power and VRAM.  The accuracy grown faster, but the results are nearly the same. From my point of view, big batch size is excellent to use when we needed to get the results in a short time. So, the main reason is that we are in a lack of time. In other situations I’ll prefer to choose something nearly the default batch size.