# LEARNING RATE INVESTIGATION

IE 8521 COURSE PROJECT REPORT

**Yu Yang**[*]
School of Statistics
University of Minnesota
yang6367@umn.edu

**Liwei Huang**
School of Mathematics
University of Minnesota
hual1842@umn.edu

December 9, 2019

## ABSTRACT

Learning rate is one of the most important hyper-parameters in deep neural network training. An appropriate learning rate scheme can help reduce the training time and boost the model performance. In this report, we propose four decaying learning rate sequences and three updating strategies to investigate the effect of learning rate schemes, in terms of convergence time and model performance. We run experiments on MNIST, and CIFAR-10 datasets, both invovling convolutional neural networks. It is found that for the MNIST dataset, learning rate updating schemes don't introduce much improvement. While for the CIFAR-10 dataset, by applying certain designed learning rate updating schemes, both the convergence time and the model performance get improved.

***K*eywords** Learning rate · Convergence time · Model performance

## 1 Introduction

Learning rate is a very important hyper-parameter in the configuration of a deep neural network. It controls the extent to which we update the parameters at each training step. Choosing an appropriate learning rate can be challenging since a value too small may lead to a long training process while a value too large would give fluctuation in loss and sub-optimal results. Therefore, it is crucial to investigate the effect of learning rate updating schemes on convergence time and final model performance.

In this report, we propose three updating strategies and four learning rate sequences, and use two datasets which are widely used in deep learning as examples to investigate the effect of learning rate updating dynamics. The report goes as follows: Section 2 describes the datasets, models, and loss function that are used in the experiments; Section 3 proposes four decaying sequences and three updating strategies; Section 4 shows the results and analyses of the experiments; Section 5 concludes the report.

## 2 Dataset and Model Description

### 2.1 Datasets and Task

We use two datasets that are commonly used in the basic deep learning models as examples: MNIST dataset[4] and CIFAR-10 dataset[5]. The MNIST dataset is composed of gray-scale hand written 0-9 which are $28 \times 28$ in size, containing 60,000 training and 10,000 testing examples. And the CIFAR-10 dataset consists of 10 classes of $32 \times 32$ RGB images with 50,000 training and 10,000 testing examples in total. Example pictures are shown in Appendix 6.1.

For MNIST dataset, the goal is to correctly classify the digits; and for CIFAR-10 dataset, the goal is to correctly classify the pictures with certain classes. They are both 10-class classification problems.

[*]Check `https://github.com/yuyangstatistics/lr_decaying` for code and more results.

### 2.2 Convolutional Neural Network

CNN based models have been successfully exploited in image classification [1]. There are many variants of CNN that works pretty well, such as VGG16[2], ResNet50[3]. But for simplicity and considering the time limit, we only use two very simple CNN neural network frameworks with three convolution layers inside to be the classification models, one for MNIST, and the other for CIFAR-10. The detail of the models is shown in Appendix 6.2.

### 2.3 Loss Function

Since both MNIST and CIFAR-10 are multi-class classification problems, cross-entropy loss would be a good choice as the loss function. The definition is as follows:

$$l(x, class) = -\log\left(\frac{\exp\left(x[class]\right)}{\sum_j \exp\left(x[j]\right)}\right) = -x[class] + \log\left(\sum_j \exp\left(x[j]\right)\right) \tag{1}$$

## 3 Learning Rate Decaying Schemes

### 3.1 Decaying Sequences

By lecture notes, when the learning rate sequence is divergent, the algorithm converges faster, so we consider three divergent decaying sequences. Also, exponentially decaying sequence is included due to its wide usage in deep learning practice. The detail of the sequences are as follows and the comparison of the four sequences is shown in Figure 1.

Let $r_0$ denote the initial learning rate, and $r_n$ denote the learning rate at step $n$.

1. **seq1**: $r_n = \frac{r_0}{n}$, is a divergent sequence.
2. **seq2**: $r_n = \frac{r_0}{\sqrt{n}}$, is another divergent sequence, but converges to 0 slower than seq1.
3. **seq3**: $r_n = 0.9^n r_0$, is an exponentially decaying sequence which is used quite often in deep learning, but is convergent.
4. **seq4**: $r_n = q_n r_0$, where $q_n = (1 - \frac{((n-1) \mod 10)}{10})\frac{1}{[\frac{n}{10}]+1}$, is a decaying cyclic sequence, which allows the learning rate a boost when the value gets too small.

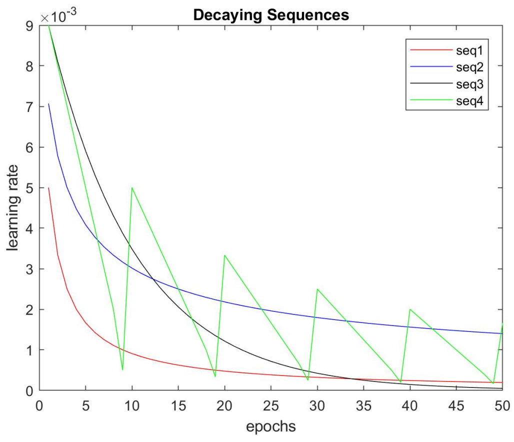


Figure 1: Decaying Sequences

### 3.2 Updating Strategies

We propose three dynamic learning rate updating strategies as below.

1. **By Epoch**: for a given sequence, update learning rate after finishing every epoch. See Algorithm 1 for details.

2. **By Cutoff**: for a given sequence, update learning rate when the change in validation loss is smaller than predefined cutoffs. Cutoff is defined to be $c_0 = loss_0 * 0.01,\ c_{t+1} = c_t * 0.2$. See Algorithm 2 for details.

3. **By Oscillate**: for a given sequence, update learning rate when the validation loss is larger than the previous one. See Algorithm 3 for details.

**Algorithm 1** Update by epoch

```
1: Set initial learning rate: $lr = r_0$
2: Run the model and update parameters with $lr$
3: Obtain a sequence of learning rate $R = \{r_1, r_2, \cdots, r_n\}$
4: for $t \leftarrow 1, 2, \cdots, n$ : do
5:     $lr \leftarrow R.pop(0)$
6:     run the model and update parameters with $lr$
```

**Algorithm 2** Update by cutoff

```
1: Set initial learning rate: $lr = r_0$
2: Run the model and update parameters with $lr$
3: Obtain a sequence of learning rate $R = \{r_1, r_2, \cdots, r_n\}$
4: for $t \leftarrow 1, 2, \cdots, n$ : do
5:     if $|valid\, loss_t - valid\, loss_{t-1}| < cutoff$ then
6:         $lr \leftarrow R.pop(0)$
7:         $cutoff = cutoff \times 0.2$
8:     run the model and update parameters with $lr$
```

**Algorithm 3** Update by oscillate

```
1: Set initial learning rate: $lr = r_0$
2: Run the model and update parameters with $lr$
3: Obtain a sequence of learning rate $R = \{r_1, r_2, \cdots, r_n\}$
4: for $t \leftarrow 1, 2, \cdots, n$ : do
5:     if $valid\, loss_t - valid\, loss_{t-1} > 0$ then
6:         $lr \leftarrow R.pop(0)$
7:     run the model with learning rate $lr$.
```

## 4 Experiments

### 4.1 Experiment Configuration

In total, we consider 14 different settings: twelve given by the combination of the 4 sequences and 3 strategies using SGD optimizer with initial learning rate $r_0$ being 0.01; one given by SGD optimizer with fixed learning rate 0.001; one given by Adam optimizer with fixed learning rate 0.001. For each setting, we run the experiments on both MNIST dataset and CIFAR-10 dataset.

We split both datasets into training set and validation set, and use the convergence time and accuracy on the validation set as the comparison metrics. For simplicity, we don't consider the computational time of specific algorithms and use the number of epochs at convergence as the metric of convergence time. And we use the final accuracy as the metric of model performance.

In order to dig out the effect of learning rate, we do comparison in two stages. Firstly, for each strategy, we compare the results of four sequences and choose the optimal setting in terms of convergence time and final accuracy respectively. Secondly, we compare the optimal settings in each strategy with the benchmarks in terms of convergence time and final accuracy.

### 4.2 Analysis on MNIST

Figure 2, 3, 4 show the results of comparison under the three strategies respectively. Figure 5 shows the optimal settings' comparison result. And Table 1 gives a numerical illustration on the performance of these settings. We summarize our findings and analyses as follows:

(1) Under "By Epoch", seq3 converges the fastest and seq4 achieves the highest final accuracy.

(2) Under "By Cutoff", seq4 converges the fastest and seq4 achieves the highest final accuracy.

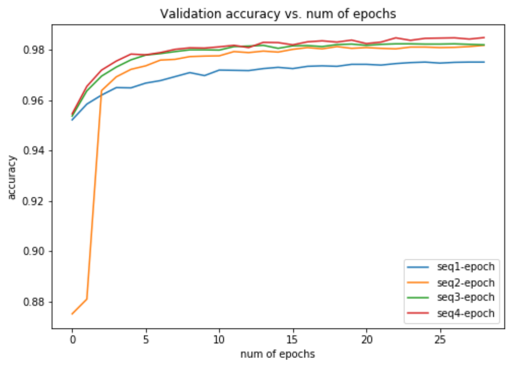
(3) Under "By Oscillate", seq3 converges the fastest and achieves the highest final accuracy.

(4) In terms of convergence time, the setting "seq3 By Epoch" converges after 19 epochs, is the fastest.

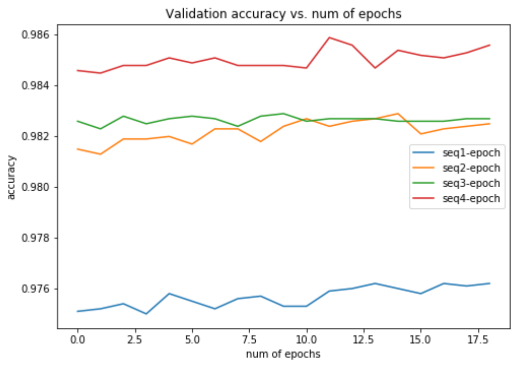
(5) In terms of final accuracy, the setting "seq4 By Cutoff" achieves the highest accuracy, which beats both benchmarks. Also, under all updating strategies, seq4 is always the best while seq1 is always the worst. The reason we suppose is that seq1 has much smaller learning rate and decays very quickly in the first few iterations, which slows down the progress of accuracy.

(6) The difference among these settings are very small, either in convergence time or in final accuracy. The reason is possibly that this classification problem is so easy that even with fixed learning rate, high accuracy can be achieved.

(7) Under "By Cutoff" and "By Oscillate", seq3($0.9^n r_0$) and seq4(cyclic sequence) behave similarly. This phenomenon matches with our intuition, since in these two scenarios, learning rate gets updated for less than 20 times, and according to Figure 1, seq3 and seq4 are close in the first 20 iterations.

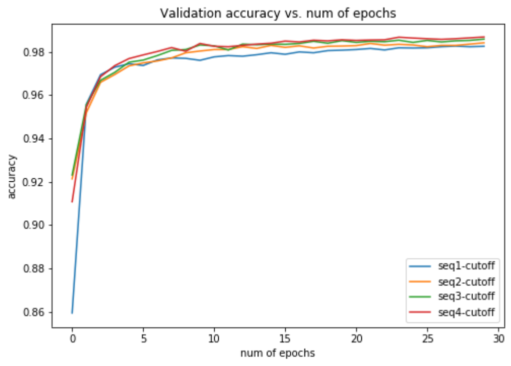


(a) Comparison of sequences under "By Epoch"

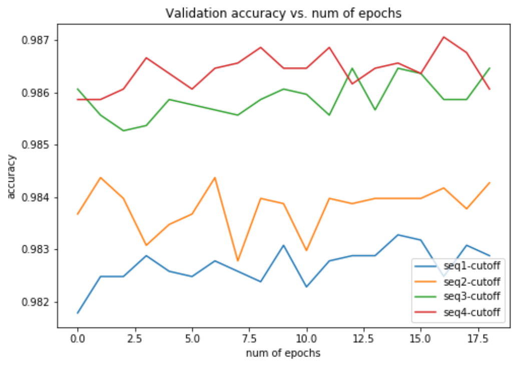


(b) A closer look at final accuracy under "By Epoch"

Figure 2: MNIST Comparison under "By Epoch"

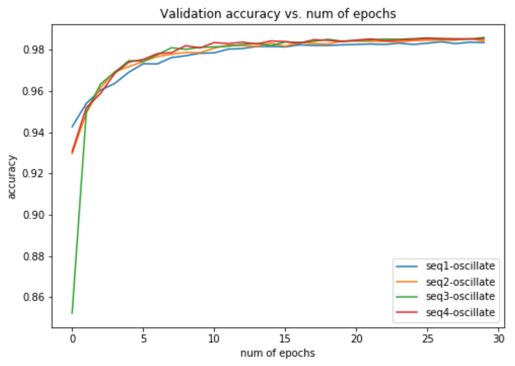


(a) Comparison of sequences under "By Cutoff"

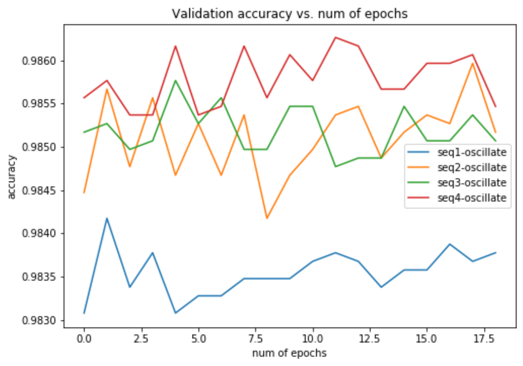


(b) A closer look at final accuracy under "By Cutoff"

Figure 3: MNIST Comparison under "By Cutoff"

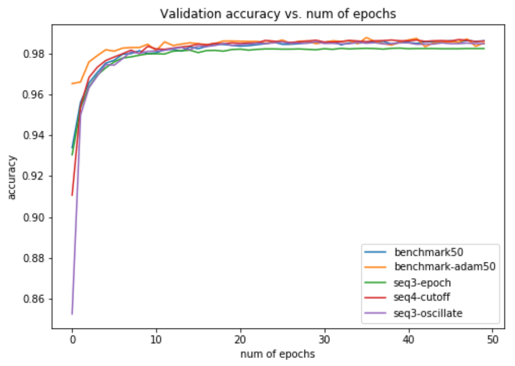


(a) Comparison of sequences under "By Oscillate"

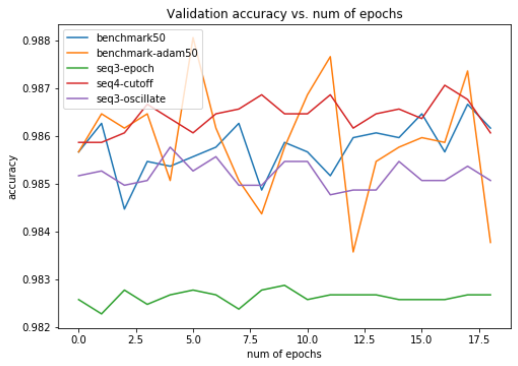


(b) A closer look at final accuracy under "By Oscillate"

Figure 4: MNIST Comparison under "By Oscillate"



(a) Convergence Time Comparison,first 50 epoch



(b) A closer look at final accuracy

Figure 5: MNIST Convergence time comparison with benchmark

## 4.3 Analysis on CIFAR-10

Figure 6a, 6b, and 7a show the results of comparison under three strategies respectively. Figure 7b shows the optimal settings' comparison result. Table 2 gives a numerical illustration about the performance of these settings. We summarize our findings and analyses as follows:

(1) Under "By Epoch", seq1 converges the fastest, and seq3 achieves the highest final accuracy.

(2) Under "By Cutoff", only seq1 converges and has an increasing trend, while the other sequences have relatively large fluctuation and have decreasing trends. Seq1 achieves the highest final accuracy.

(3) Under "By Oscillate", seq1 converges the fastest, and we can clearly see the rank of convergence time: seq1 < seq2 < seq4 < seq3. And the four sequences achieves similar final accuracy results.

(4) In terms of convergence time, the setting "seq1 By Epoch" and "seq1 By Oscillate" converge after 8 epochs, are the fastest. Also, seq1 converges the fastest under all three updating strategies, and note that seq3 and seq4, which are much larger than seq1 in the first few iterations according to Figure 1, undergo acute fluctuation in the first few epochs, which jointly suggests that the initial learning rate 0.01 may be too large and shrinking it may help avoid fluctuation and thus help with convergence.

(5) In terms of accuracy, the setting "seq3 By Epoch" achieves the highest accuracy, outperforms the two benchmarks a lot. Also, all settings under "By Epoch" and "By Oscillate" outperform the benchmarks, which suggests that it is worth the effort to do learning rate decaying. Another thing to note is that the results under "By Cutoff" are less satisfying. The reason is that in this setting, the learning rate gets updated for less than 5

times, and hence the learning rate are kept at a relatively large level, which would then lead the parameters away from the optimal, as illustrated by the downside trend in Figure 6b.

(6) Unlike MNIST, the results given by three updating strategies show different patterns, and the difference between settings are much clearer in the CIFAR experiment.

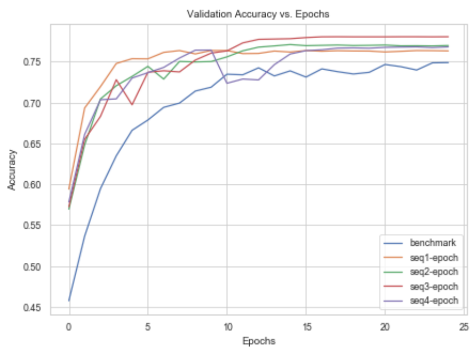
(7) Similar to MNIST result, under "By Cutoff" and "By Oscillate", the behaviors of seq3 and seq4 are similar, which can be interpreted using similar reasoning as in MNIST.

Table 1: MNIST Comparison

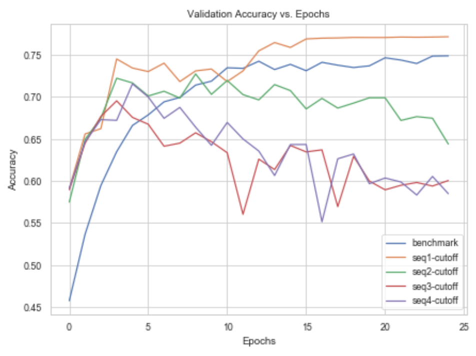
| *Method* | | Accuracy | Convergence |
|---|---|---|---|
| *Benchmark* | *SGD* | 0.98264 | 34 |
| | *Adam* | 0.98574 | 30 |
| *By Epoch* | *seq1* | 0.97608 | 48 |
| | *seq2* | 0.98230 | 38 |
| | *seq3* | 0.98263 | **19** |
| | *seq4* | 0.98527 | 40 |
| *By Cutoff* | *seq1* | 0.98290 | 32 |
| | *seq2* | 0.98405 | 28 |
| | *seq3* | 0.98614 | 29 |
| | *seq4* | **0.98656** | 22 |
| *By Oscillate* | *seq1* | 0.98372 | 25 |
| | *seq2* | 0.98544 | 27 |
| | *seq3* | 0.98514 | 24 |
| | *seq4* | 0.98586 | 30 |

Table 2: CIFAR Comparison

| *Method* | | Accuracy | Convergence |
|---|---|---|---|
| *Benchmark* | *SGD* | 0.7487 | 25 |
| | *Adam* | 0.7309 | 17 |
| *By Epoch* | *seq1* | 0.763 | **8** |
| | *seq2* | 0.7696 | 12 |
| | *seq3* | **0.7803** | 11 |
| | *seq4* | 0.7679 | 14 |
| *By Cutoff* | *seq1* | 0.7714 | 15 |
| | *seq2* | 0.6438 | 25 |
| | *seq3* | 0.6004 | 25 |
| | *seq4* | 0.5848 | 25 |
| *By Oscillate* | *seq1* | 0.7677 | **8** |
| | *seq2* | 0.773 | 11 |
| | *seq3* | 0.7692 | 15 |
| | *seq4* | 0.7673 | 12 |

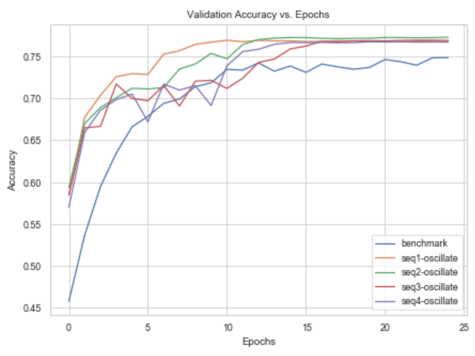


(a) Comparison of sequences under "By Epoch"

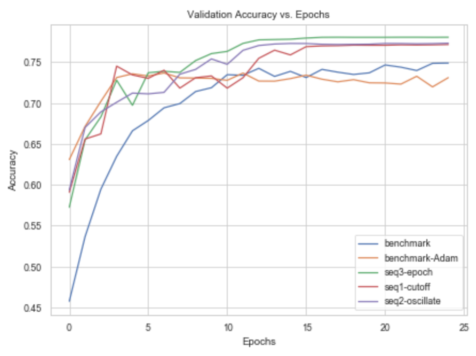


(b) Comparison of sequences under "By Cutoff"

Figure 6: CIFAR Comparison Plots



(a) Comparison of sequences under "By Oscillate"



(b) Comparison of optimal settings

Figure 7: CIFAR Comparison plots

### 4.4 Overall Analyses and Guidelines

In the experiments mentioned above, we noticed that for the two datasets we considered, the behaviour of the proposed settings differs a lot, with regard to both convergence time and final accuracy. Here are some of our analysis on the cause of the difference and some guidelines based on the results in the previous sections.

(1) CIFAR-10 data is much more complicated than the MNIST data. CIFAR-10 data is composed of RGB pixels while MNIST data is composed of sparse gray-scale pixels. In the MNIST case, it is much easier to achieve high accuracy with fixed learning rate, and therefore, the effect of decaying learning rate is less obvious.

(2) MNIST and CIFAR-10 use different neural network models, and thus the dimension of parameters are different, which in turn influence the characterisitcs of the objective functions.

(3) Note that although seq3 is a convergent sequence, its performance is still pretty good, with regard to both convergence time and final accuracy. We suppose the reason might be that in practice, the number of epochs being trained is usually not large, and thus it will not lose much power when compared with divergent sequences.

(4) If we care more about convergence time, then choose the "seq3 By Epoch" updating scheme for MNIST dataset, and choose the "seq1 By Oscillate" updating scheme for CIFAR-10 dataset, .

(5) If we care more about final accuracy, then choose the "seq4 By Cutoff" updating scheme for MNIST dataset, and choose the "seq3 By Epoch" updating scheme for CIFAR-10 dataset.

(6) For practitioners, the complexity of the datasets and models should be taken into consideration when deciding which learning rate updating scheme to use. Based on the comparison between MNIST and CIFAR-10, we may infer that the more complex the datasets, the more boosting in performance with a proper choice of learning rate updating scheme.

## 5 Conclusion

In this report, we performed experiments on MNIST and CIFAR-10 datasets under 14 different settings, using the proposed strategies and decaying sequences. Also, we described our findings and analyses on the experimental results and provided some guidelines for practitioners. Our experiments showed for different datasets, the optimal learning rate updating scheme would be different and that by choosing proper learning rate updating scheme, the performance of the model and the convergence of the training process would be improved. For the future work, we will study the effect of learning rate in a more generic framework and on more datasets.

## References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[3] K He, X Zhang, S Ren, and J Sun. Deep residual learning for image recognition. computer vision and pattern recognition (cvpr). In *2016 IEEE Conference on*, volume 5, page 6, 2015.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

## 6 Appendix

### 6.1 Dataset Example Images

**MNIST**

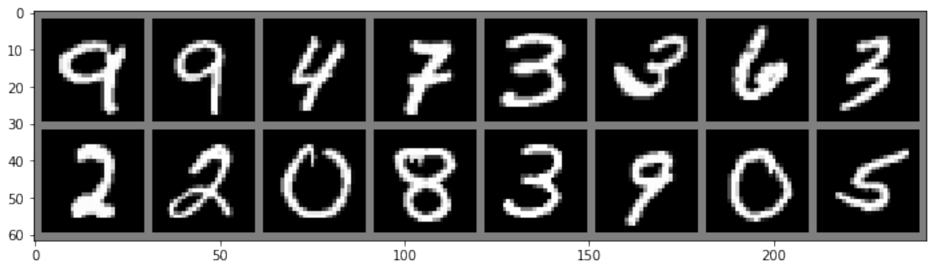


Figure 8: MNIST Example Images

**CIFAR-10**

Figure 9: CIFAR-10 Example Images

### 6.2 Network Architectures

**MNIST model architecture** Let `dk` denotes a $3 \times 3$ Convolution-Maxpooling-ReLU layer with $k$ filters, stride 1, and the kernel size of maxpooling being 2. `fk-l` denotes a $k \times l$ fully connected layer. The network consists of: `d20, f3380-128, ReLU, f128-10, Softmax`.

**CIFAR-10 model architecture** Let `dk` denotes a $3 \times 3$ Convolution-Elu-Maxpooling layer with $k$ filters, stride 1, padding 1, and the kernel size of maxpooling being 2. `fk-l` denotes a $k \times l$ fully connected layer. The network consists of: `d16, d32, d64, Dropout, f1024-500, Elu, Dropout, f500-10`.