Homework 2
NN Basics, CNNs, AEs, and GANs.[1]

GEC-1539: ML&DS - Development of Applications

OUT: Sunday, May 1st, 2022
DUE: Sunday, May 22nd, 2022, 09:00pm, Beijing Time

Lecturer: Mark Vogelsberger
Mentor: Haizhou Shi
TAs: Huazhi Dong, Yu Wang, Beidi Zhao

Reporter: **LI HAODONG**

Instructions

Homework 2 covers the basic knowledge of neural networks, convolutional neural networks, auto-encoders, and generative adversarial networks. There are two main parts in this homework: the first part contains several questions for each topic. To answer them, you will have to go through the lecture notes carefully and grasp a solid understanding of the material. The second part contains code implementation and experiments. Read through the Instructions below before you start working on the homework.

- **Collaboration Policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only.
- **Late Submission Policy:** The submission of an assignment made past the deadline will get 70% of the credits. Any assignment submitted more than 3 days past the deadline will get zero credit.
- **Submission Policy**
  - **Written problems submission:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, please write your solution in the LaTeX files provided in the assignment and submit in a PDF form. Put your answers in the question boxes (between `\begin{soln}` and `\end{soln}`) below each problem. **Handwritten solutions are not accepted and will receive**

[1] Compiled on Wednesday 11th May, 2022 at 02:49

zero credit. Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. Please upload the written problems via the Eds platform.

- **Code submission:** All code must be submitted under the specific instruction given in the problem. We will be using the Edstem platform to auto-grade your code. **If you do not submit your code there, you will not receive any credit for your assignment.** The Edstem platform will auto-detect the possible plagiarism. Please make sure you familiarize yourself with the academic integrity information for this course.

Important Notes on the Programming Problems:

- Do not use any toolboxes except those already imported in the code template.
- Read the doc-strings/comments in the template very carefully before you start.
- Reach out for help on Edstem Platform or during office hours when you struggle.
- Do not change any function signatures because your code will be auto-graded.
- Try to vectorize the computation as much as possible (e.g. compute in the form of matrix multiplication, utilize numpy functions instead of loops, etc.)
- Use Python 3.6 or above, and the latest version of numpy.

1 Written Questions (50 points)

In this section, you will work through a number of problems covering the topics of Neural Networks, Convolutional Neural Networks, Auto-Encoders, and Generative Adversarial Networks. At the time when this homework is released, some of the contents have not been covered in the lecture. You don't need to worry about that and you can come back to finish the homework once we cover them in the lecture. Here is an example of a question. Use the command '**\CorrectChoice**' to select the correct answer.

1. (0 points) Assignment turned in late without prior approval will incur a penalty. How much is the penalty? Up to 3 days: ___ After 3 days: ___.

   Select one:

   ○ 10%, 60%
   ○ 30%, 60%
   [✓ ● 30%, 100%]
   ○ 50%, 100%

1.1 Neural Network Basics (14 points)

1. (2 points) In most of the cases, for regression problem, what is the activation function for the output layer?

   [✓ ● sigmoid function]
   ○ tanh function
   ○ ReLU function
   ○ None of them above

2. (2 points) For the binary classification problem, what's the distribution we assume the output variable has? What's the activation function we use for the output layer?

   ○ Gaussian; sigmoid function
   ○ Gaussian; identity function ($f(x) = x$)
   [✓ ● Bernoulli; sigmoid function]
   ○ Bernoulli; identity function ($f(x) = x$)

3. (2 points) Which of the following statement is true about the gradient descent?

   ○ Gradient descent is the only way we can train a neural network.
   ○ With a learning rate that's small enough, gradient descent will always find the global minimum if infinite steps of updates are allowed.
   [✓ ● Stochastic gradient descent helps the model escape from the local minimum by adding noises to the loss function.]

- ○ To perform gradient descent, we only need to compute the forward pass for a neural network.

4. (2 points) Which of the following statement is true about the weight penalty regularization for neural networks?

   - ○ Applying L1 regularization to a layer of the neural network will likely result in a 0 bias term.
   - ● Regularization will increase the representative power of the neural network, which is helpful to the model that suffers from underfitting.
   - ○ When applying the L2 regularization to a layer of the network with SGD optimizer, it's equivalent to decaying the weights by a certain multiplicative factor at each step.
   - ○ L2 regularization will lead to sparse weights.

5. (2 points) Which of the following statement is true about early stopping for neural networks?

   - ○ We should stop training the model when we observe an increase of the training loss.
   - ○ We should stop training the model when we observe that the training loss keeps going up for certain epochs.
   - ○ We should stop training the model when we observe an increase of the validation loss.
   - ● We should stop training the model when we observe that the validation loss keeps going up for certain epochs.

6. (2 points) Which of the following statement is true about the Dropout layer in neural networks?

   - ● For dropout probability $p = 0.3$, we randomly mask the weights for 30% of the times.
   - ○ For dropout probability $p = 0.3$, we randomly mask the weights for 70% of the times.
   - ○ For dropout probability $p = 0.3$, we randomly mask the activations for 30% of the times.
   - ○ For dropout probability $p = 0.3$, we randomly mask the 30% of the layers during training.

7. (2 points) Suppose a layer generates the output $[0.3, 0, 1, 0.7, 0, 0.2]$ after the ReLU activation. The dropout layer **Dropout(p=0.2)** randomly samples a mask $[1, 1, 0, 1, 1, 0]$. What's the result of the "dropped-out" activation during training and testing (according to the original paper of Dropout[2])?

   - ● train=$[0.3, 0, 0, 0.7, 0, 0]$, test=$[0.24, 0, 0.8, 0.56, 0, 0.16]$
   - ○ train=$[0.3, 0, 0, 0.7, 0, 0]$, test=$[0.3, 0, 1, 0.7, 0, 0.2]$
   - ○ train=$[0.3, 0, 0, 0.7, 0, 0]$, test=$[0.06, 0, 0.2, 0.14, 0, 0.04]$
   - ○ train=$[0.3, 0, 1, 0.7, 0, 0.2]$, test=$[0.3, 0, 0, 0.7, 0, 0]$

1.2 Convolutional Neural Networks (14 points)

1. (2 points) Suppose we have two matrices:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$$
$$\boldsymbol{B} = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$$

   What's the result of convolving these two matrices?

   - ○ $\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$
   - ○ $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 5 & 11 \\ 6 & 23 & 29 \end{bmatrix}$
   - ● $\begin{bmatrix} 19 & 25 \\ 37 & 43 \end{bmatrix}$
   - ○ $\begin{bmatrix} 0 & 3 & 3 \\ 8 & 18 & 14 \\ 8 & 19 & 11 \end{bmatrix}$

[2] https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

2. (3 points) Consider the following statements with respect to the pooling layer:
    1. It reduces computational complexity
    2. It increases image resolution
    3. It does not have trainable parameters
    4. It is always has to be followed by an activation layer

    Which among the following options represents the statement(s) from above that are not true with respect to the pooling layer?

    - ○ Only 2
    - ○ Both 2 and 3
    - [✓ ● 2, 3 and 4]
    - ○ 1,2 and 3

3. (3 points) 2-D Convolutional layers expect inputs of shape (batch_size, H, W, C), where C is the number of channels. Suppose you are passing 5 28x28 full-color images through a Conv2D layer with 14 filters each with 4x4 kernel, a stride of (1,1) with padding='same'. What is the input dimensions and the output dimensions?

    NOTE: The options are of the format: Input Dimensions → Output Dimensions

    - ○ (5,3,3,1) → (5,28,28,7)
    - [✓ ● (5,28,28,3) → (5,28,28,14)]
    - ○ (5,28,28,1) → (5,7,7,14)
    - ○ (5,28,28,3) → (5,7,7,14)

4. (3 points) During backpropagation, the gradient from the next layer is passed back to only that neuron which achieved the max during max-pooling at the forward pass. All other neurons get zero gradient.

    - ○ TRUE
    - [✓ ● FALSE]

5. (3 points) Which of the following statements about CNNs is ***false***?

    - ○ CNNs are generally better than MLPs dealing with image data since MLPs cannot learn pixel dependencies present in the images.
    - [✓ ● In CNNs, the number of parameters to be trained is significantly lower than the MLPs, therefore reducing the chance of overfitting.]
    - ○ Pooling layers are usually applied before the activation function.
    - ○ Convlutional layers with 'padding=same' and 'strides=1' will not change the height and width of the feature map.

1.3 Auto Encoders (10 points)

1. (2 points) Which of the following statements about the internal representations of a neural network is true?

   ○ The representation space near the output layer preserves more information than that near the input layer.

   ● The representation space near the output layer preserves more about the high-level information that's related to the downstream task.

   ○ The amount of information preserved in the network first goes down, and then goes up. The layer in the middle forms the "information bottleneck".

   ○ The representation space yielded near the output layer of a neural network will be more generalizable, i.e., more suitable for other downstream tasks.

2. (3 points) Suppose we have an encoder network $\mathcal{E}$, a decoder network $\mathcal{D}$, and a dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$, where each data point is a d-dimensional vector. Please write down the $l2$-loss function for the Auto Encoder $\mathcal{E}(\mathcal{D}(\cdot))$.

   Solution

$$\begin{aligned}\text{error} &= \|\hat{\boldsymbol{X}} - \boldsymbol{X}\|^2 \\ &= \|\mathcal{D}(\mathcal{E}(\boldsymbol{X})) - \boldsymbol{X}\|^2 \\ \boldsymbol{X} &= \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}\end{aligned}$$

3. (2 points) Which of the following statements about the vanilla autoencoder is true? (Select all that apply)

   ○ In Auto Encoder, the dimension d' of the representation yielded after the encoder can only be smaller than the original data's dimension d.

   ● Generally, we don't have the control over the representation spaces yielded by the encoder, which restricts the use of Auto Encoder as a generative method.

   ● The Auto Encoder has the problem of generalization. It can generate the data that has been seen during training, but cannot generate new data well.

4. (3 points) Which of the follow statements about the autoencoders is ***false***?

   ● The Sparse Auto Encoder (SAE) wants the weights of the encoder and decoder to be sparse.

   ○ The Denoise Auto Encoder (DAE) can have more robust representation generated by the encoder network.

   ○ The Variational Auto Encoder (VAE) has two losses to be optimized at the same time. The first loss is the reconstruction loss, and the second loss makes sure the latent representation is close to the normal distribution.

1.4 Generative Adversarial Networks (12 points)

1. (6 points) Which of the following statements about the GANs is true? Select all that apply.

   - [✓] During training, each time the generator generates a batch of fake data, we use the fake data and the true data to train the discriminator until it converges.
   - ○ In GAN training, we can jointly train the discriminator and the generator with only one forward pass and one backward pass.
   - [✓] Mode collapse of GANs refers to the phenomenon that the generator keeps generating very similar data. Although the generated data is very verisimilitude, it's quite against the original motivation of GANs.
   - ○ In the style transfer application, we should find a paired dataset where each data sample has multiple different styles.
   - [✓] The GANs are implicit generative model, since they don't assign the exact probability to a data sample of being a true data.

2. (6 points) Prove that when the discriminator is optimal, maximizing the loss function $\mathcal{J}(D^\star, G)$ is equivalent to minimizing the JS-Divergence between the true data distribution P_{data} and the generated data distribution P_G.

   NOTE: the loss function of the vanilla GAN is defined and represented as:

$$\begin{aligned}\mathcal{J}(G, D) &= \mathbb{E}_{\boldsymbol{x}\sim P_{\text{data}}}[-\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}}[-\log(1 - D(G(\boldsymbol{z})))] \\ &= -\int \left[P_{\text{data}}(\boldsymbol{x}) \log D(\boldsymbol{x}) + P_G(\boldsymbol{x}) \log(1 - D(\boldsymbol{x} = G(\boldsymbol{z})))\right] d\boldsymbol{x}\end{aligned}$$

   The optimal discriminator function is the function whose value is optimal everywhere:

$$D^\star(x) = [D(x)]^\star$$

   Please see my answer in the next page.

Solution

$$D_q^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

$$\begin{aligned}
\Rightarrow \quad \mathcal{J}(G) &= \max_D \mathcal{J}(G, D) \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[-\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}\left[-\log\left(1 - D(G(\boldsymbol{z}))\right)\right] \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[-\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x}\sim P_G}\left[-\log\left(1 - D(\boldsymbol{x})\right)\right] \\
&= \mathbb{E}_{\boldsymbol{x}\sim P_{\text{data}}}\left[-\log\frac{P_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + P_G(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim P_G}\left[-\log\frac{P_G(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + P_G(\boldsymbol{x})}\right] \\
&= -\int\left[P_{\text{data}}(\boldsymbol{x})\log\frac{P_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + P_G(\boldsymbol{x})} + P_G(\boldsymbol{x})\log(1 - \frac{P_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + P_G(\boldsymbol{x})})\right] d\boldsymbol{x} \\
&= -\int\left[P_{\text{data}}(\boldsymbol{x})\log\frac{P_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + P_G(\boldsymbol{x})} + P_G(\boldsymbol{x})\log\frac{P_{\text{G}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + P_G(\boldsymbol{x})}\right] d\boldsymbol{x}
\end{aligned}$$

$$\begin{aligned}
LSD(P_{\text{data}}||P_G) &= KLD\left(P_{\text{data}}\,\|\frac{P_{\text{data}} + P_G}{2}\right) + KLD\left(P_G\|\frac{P_{\text{data}} + P_G}{2}\right) \\
&= \frac{1}{2}\left[P_{\text{data}}(\boldsymbol{x})\log\frac{2P_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + P_G(\boldsymbol{x})} + P_G(\boldsymbol{x})\log\frac{2P_{\text{G}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + P_G(\boldsymbol{x})}\right]
\end{aligned}$$

$$\begin{aligned}
\Rightarrow \quad \mathcal{J}(G) &= \max_D \mathcal{J}(G, D) \\
&= \log(4) - 2LSD(P_{\text{data}}||P_G)
\end{aligned}$$

Above all, we could conclude that maximizing the loss function $\mathcal{J}(D^\star, G)$ is equivalent to minimizing the JS-Divergence between the true data distribution P_{data} and the generated data distribution P_G.

2 Coding Problems (50 points)

In this section, we will be implementing, training, and evaluating the neural networks with the image data. Different from the last homework, where we require you to implement the code with Numpy only, the coding problems for this homework will be all about using the amazing machine learning and deep learning libraries. There are two coding problems in total. The first one is a 25-point (warm-up) problem, where you are asked to implement an MLP model to deal with the image classification problem with Keras. The second task requires you to implement a hand-crafted CNN model with PyTorch. We recommend you to run the coding problems on the Google Colab[a] platform. ***The links to your colab notebooks should be provided in the following solution box.***

[a] https://colab.research.google.com/notebooks/welcome.ipynb?hl=zh-cn

Solution

- 3 Link to problem 2.1: Link to My 2.1 Code on Google CoLab
- Link to problem 2.2: Link to My 2.2 Code on Google CoLab

If you find trouble opening the links above, here are GitHub links of them:

- Link to problem 2.1: Link to My 2.1 Code on GitHub
- Link to problem 2.2: Link to My 2.2 Code on GitHub

Since the file is large, so probably it is hard to render. I would appreciate it if you could download them to have a look locally. Thanks a lot!

2.1 Image Classification with MLP (25 points)

In this problem, you are asked to implement an MLP to perform an image classification task. The following requirements shall be met to get the full 25 points.

- The dataset should be the full MNIST[3] dataset;
- The data should be flattend and properly normalized before fed into the MLP;
  HINT: *tf.keras.datasets.mnist.load_data()*[4]
- Use the **Sequential** model in Keras;
  HINT: *tf.keras.Sequential()*[5]
- Use the idea of validation to tune your model;
- Try at least two regularization methods we introduced during lecture:
  - L1/L2 regularization: *tf.keras.regularizers*[6]
  - Dropout: *tf.keras.layers.Dropout()*[7]
  - Data augmentation: *tf.keras.preprocessing.ImageDataGenerator()*[8]
  - Early stopping: *tf.keras.callbacks.EarlyStopping()*[9]
- Try at least two optimizers we introduced during lecture: *tf.keras.optimizers* [10]
  - SGD
  - Momentum
  - Adam

[3] http://yann.lecun.com/exdb/mnist/
[4] https://keras.io/api/datasets/mnist/
[5] https://keras.io/api/models/sequential/
[6] https://keras.io/api/layers/regularizers/
[7] https://keras.io/api/layers/regularization_layers/dropout/
[8] https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
[9] https://keras.io/api/callbacks/early_stopping/
[10] https://keras.io/api/optimizers/

1. (2 points) Your code of data pre-processing (including loading the data, train-val split, and data normalization):

Solution

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()

y_train = np_utils.to_categorical(y_train, 10)
y_test = np_utils.to_categorical(y_test, 10)

X_train = X_train.reshape(60000, 28*28)
X_test = X_test.reshape(10000, 28*28)
X_train = X_train.astype(’float32’)
X_test = X_test.astype(’float32’)

X_train /= 255.
X_test /= 255.
```

2. (5 points) Your code of Keras **Sequential** model definition:

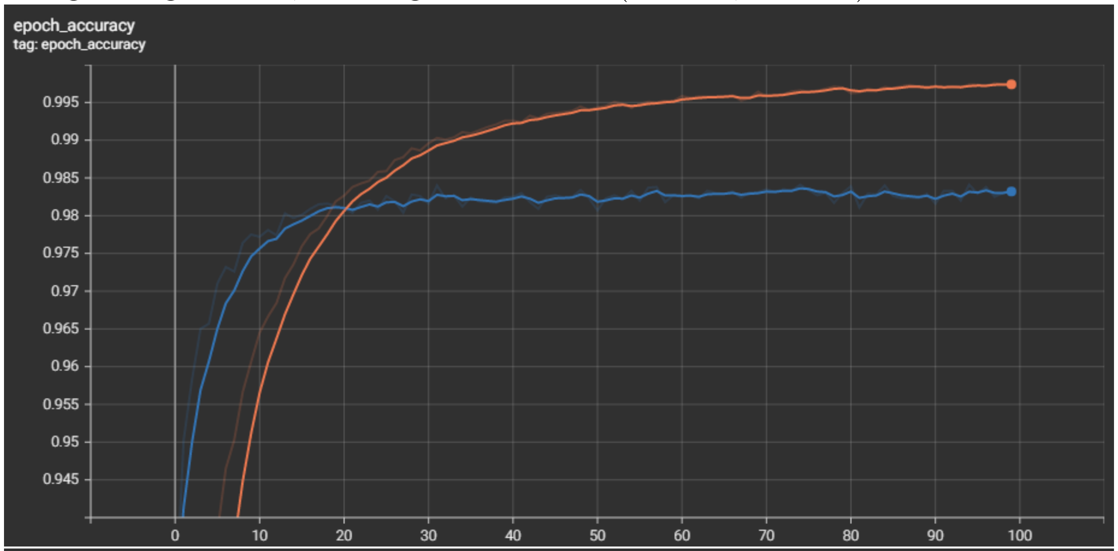
Solution

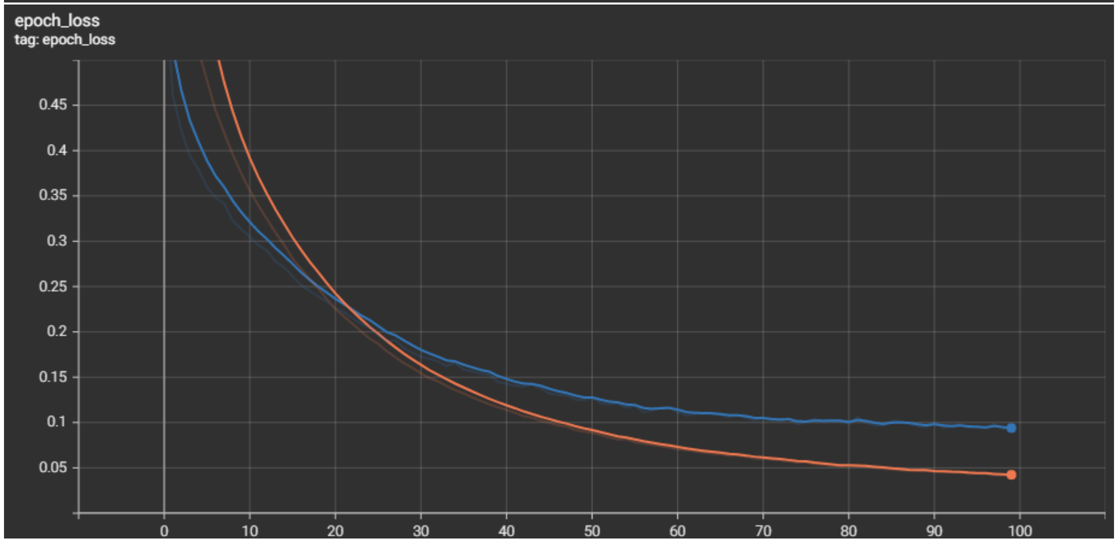
```
model = Sequential()
model.add(Dense(512, input_shape=(28*28, )))
model.add(Activation(’relu’))
model.add(Dropout(0.2))
model.add(Dense(512, kernel_regularizer=regularizers.L1L2(l1=1e-5, l2=1e-4)))
model.add(Activation(’relu’))
model.add(Dropout(0.2))
model.add(Dense(512,  kernel_regularizer=regularizers.L1L2(l1=1e-5,  l2=1e-4)))
model.add(Activation(’relu’))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Activation(’softmax’))
```

3. (9 points) Your results of trying out different regularization techniques (include training and validation loss curves, and your explanation):

Solution

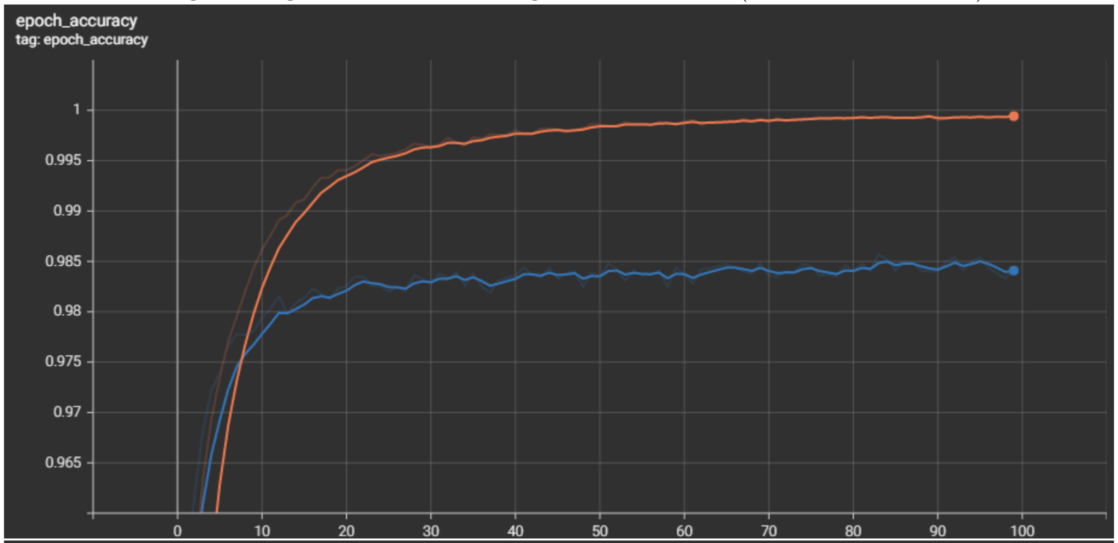
Using data generator, with regularizers.L1L2(l1=1e-5, l2=1e-4)

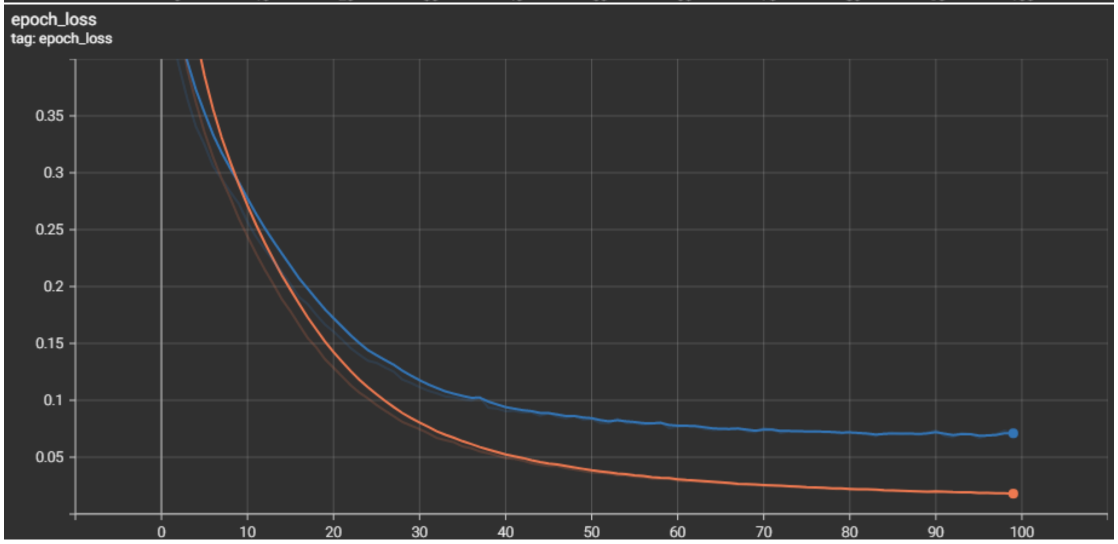




Solution

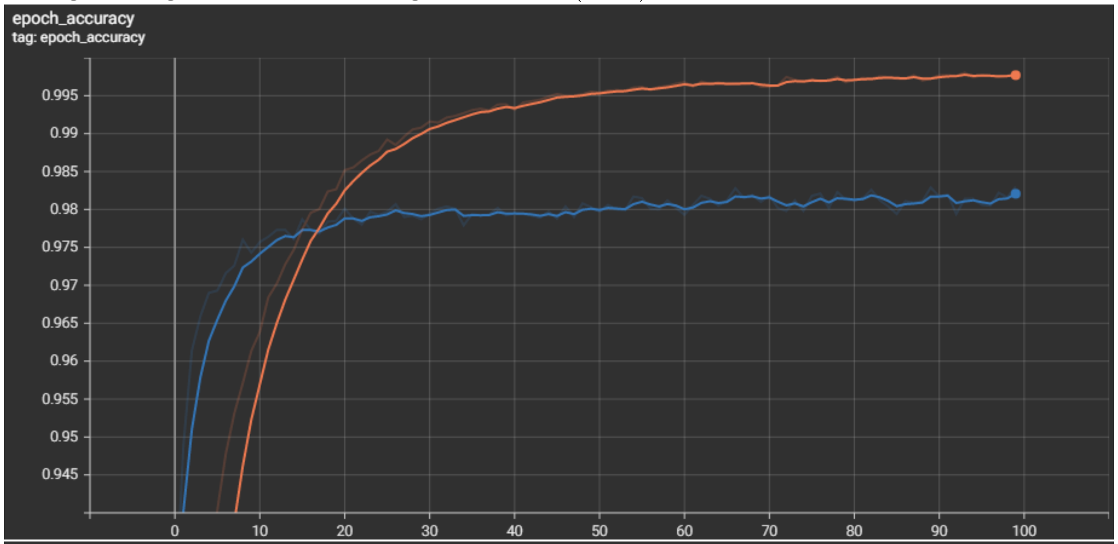
Without using data generator, with regularizers.L1L2(l1=1e-5, l2=1e-4)

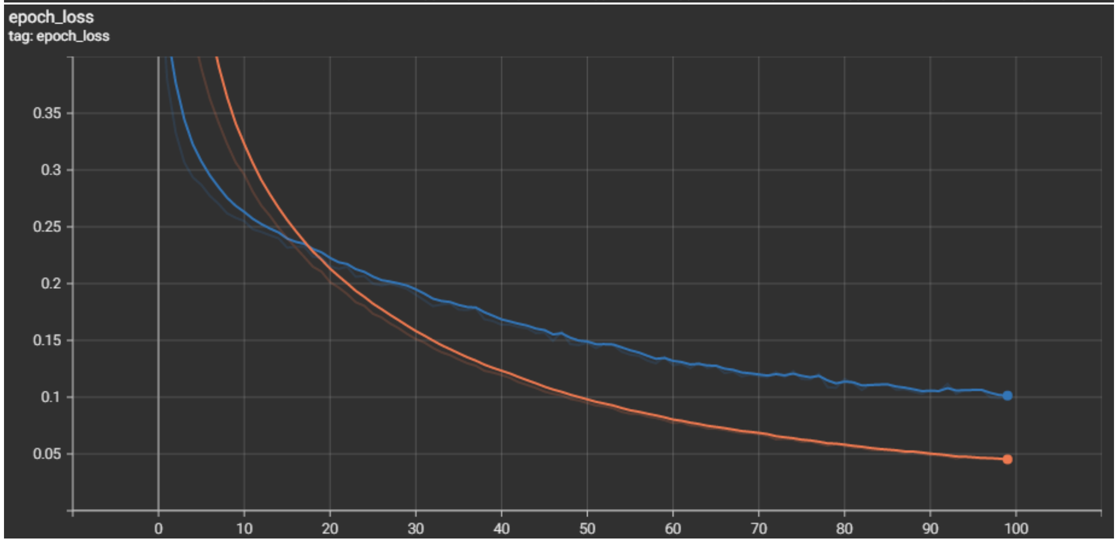




Solution

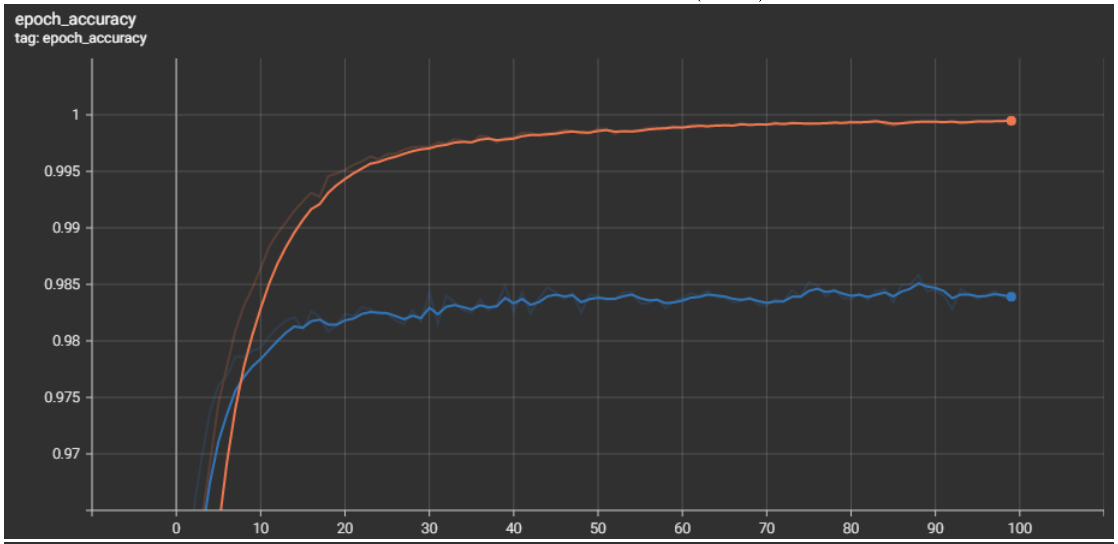
Using data generator, with regularizers.l1(1e-5)

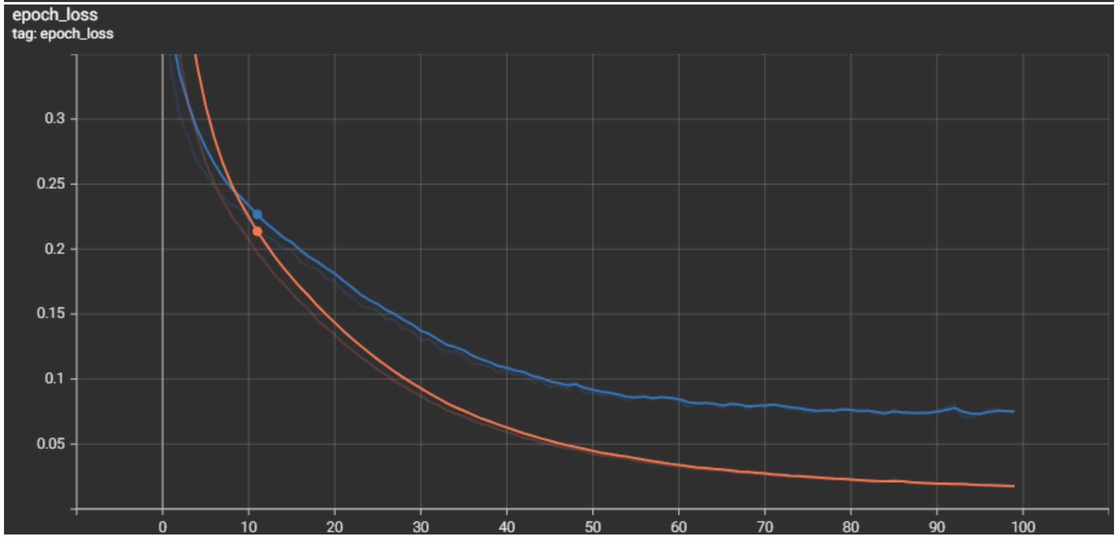




Solution

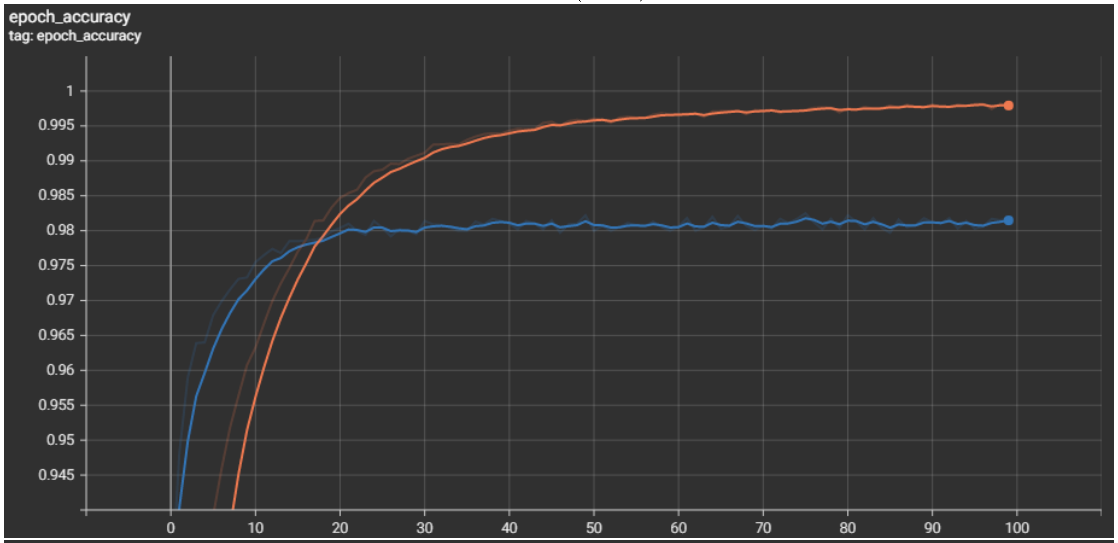
Without using data generator, with regularizers.l1(1e-5)

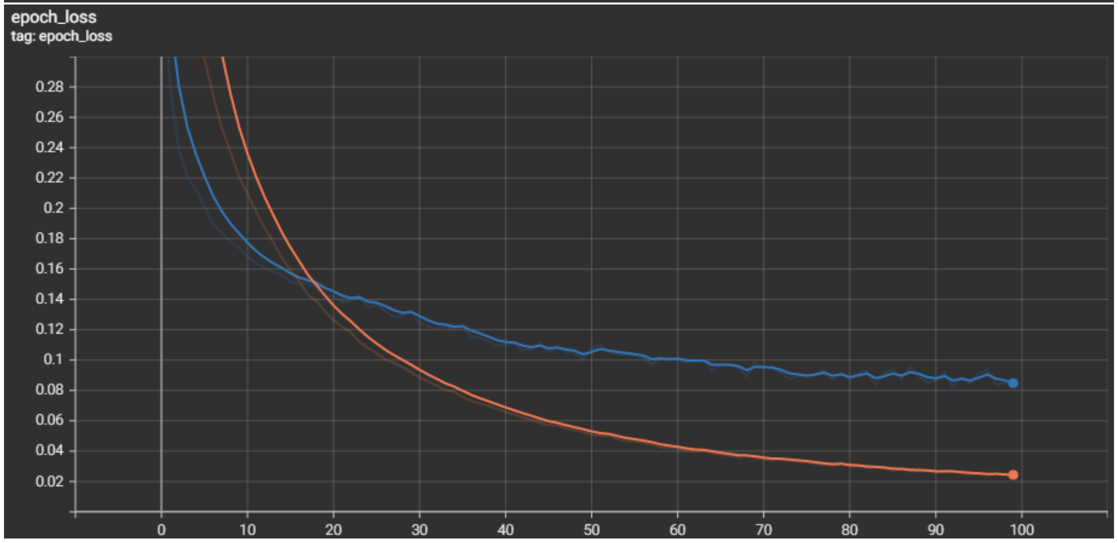




Solution

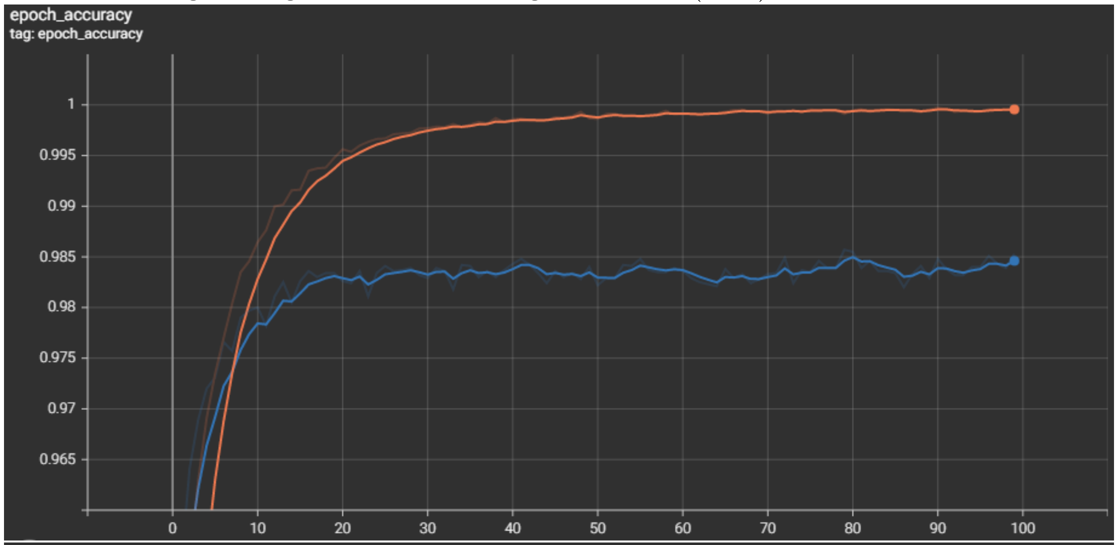
Using data generator, with regularizers.l2(1e-4)

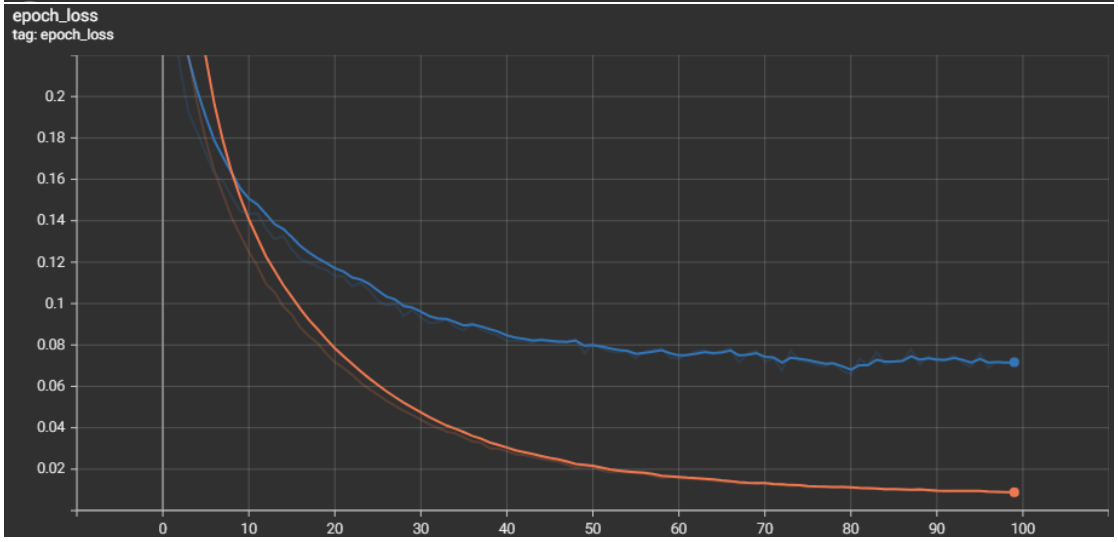




Solution

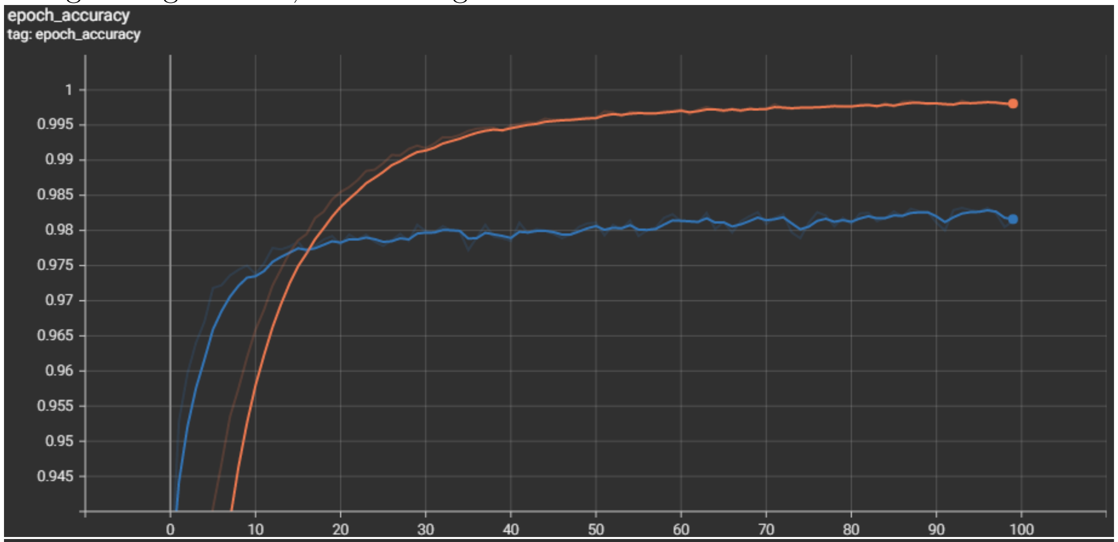
Without using data generator, with regularizers.l2(1e-4)

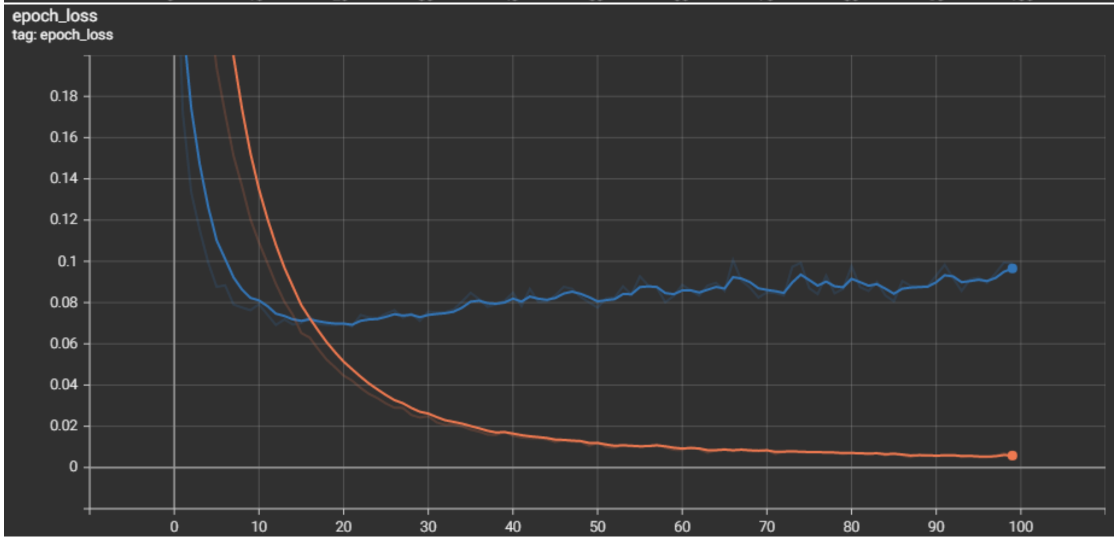




Solution

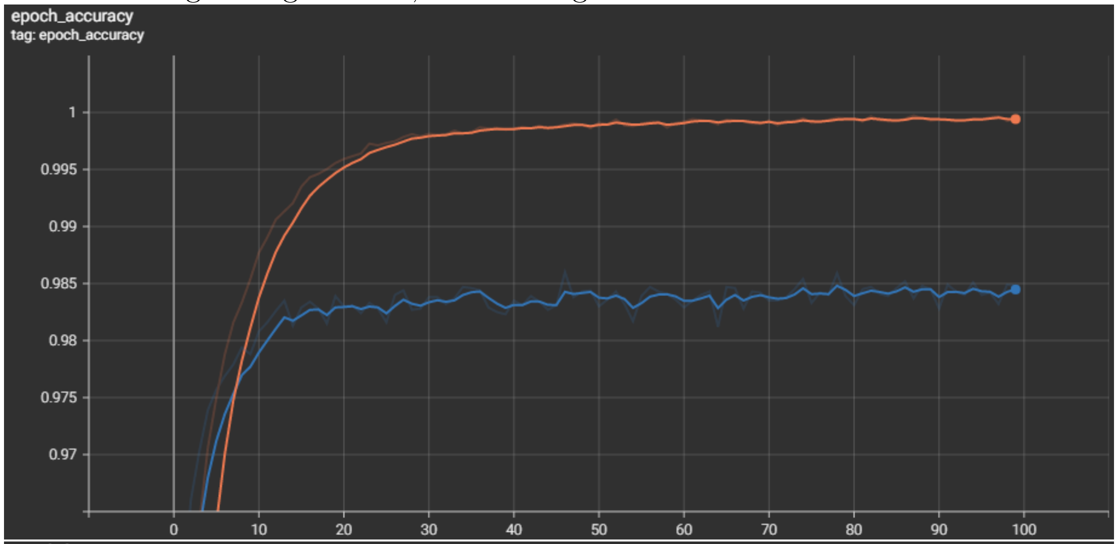
Using data generator, without regularizers

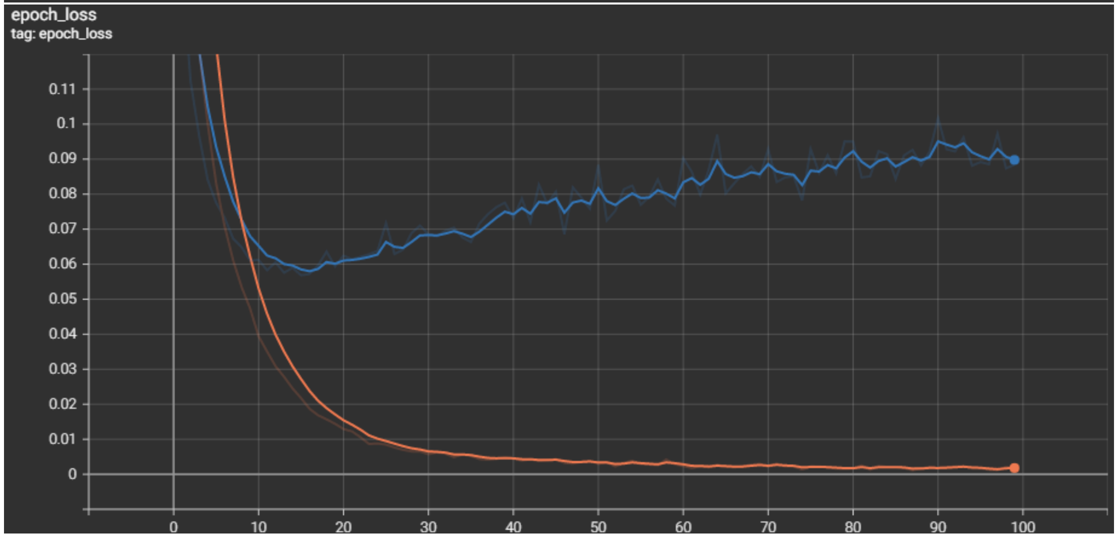




Solution

Without using data generator, without regularizers





Solution

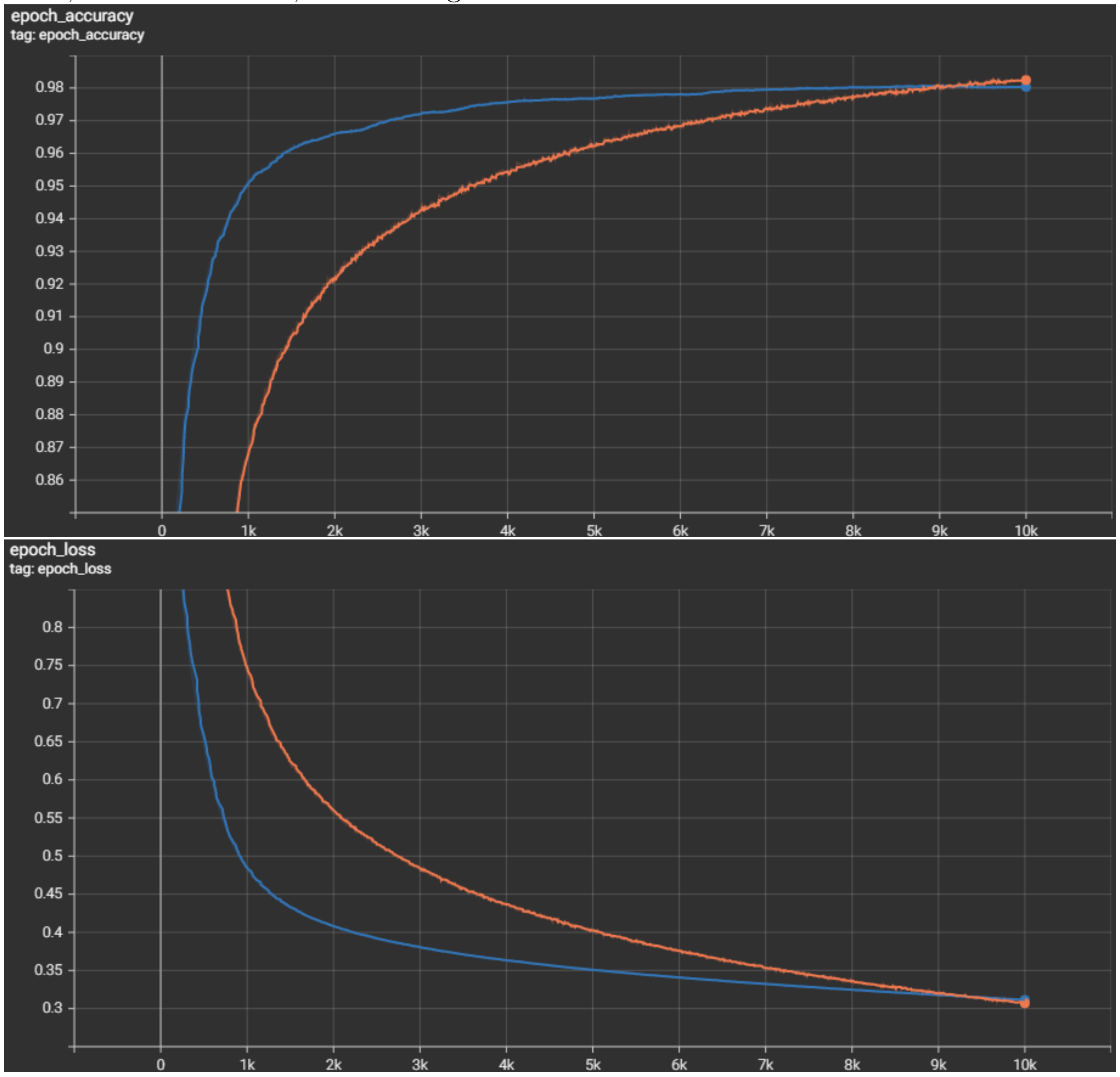
Here are my analysis and explanation:

1. While applying both L1 and L2 regularizer, the validation accuracy is higher than applying any of them.
2. Besides, as for the data generator, we could conclude that data generator is good to avoid overfitting. But since the mnist dataset is a little fixed with position and rotation, there is no obvious difference between using data generator or not.
3. The third point is that although we find hard to tell any difference in the accuracy of applying only L1 and L2 regularizers, we could find that the loss when using L2 is lower, especially with data generator applied. That is probably because the derivative of square operation is more influential than linear operation. And the old weight of model is kind of harmful.
4. The most obvious point I found is the difference between using regularizer or not. From the last four visualization plots, it is easy to say that applying regularizer is important for the convergence of the model, or the loss will increase with rapid velocity. This four plots can also indicates that the old weights of the classification model could probably be harmful.

Solution

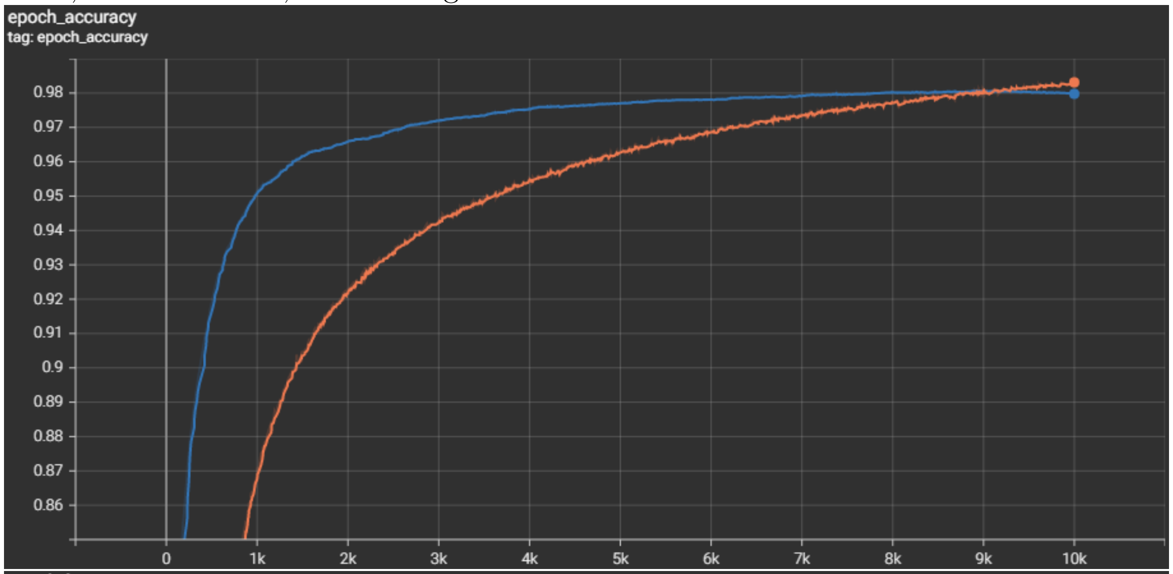
Below are the results when applying different optimizer with different parameters for momentum.

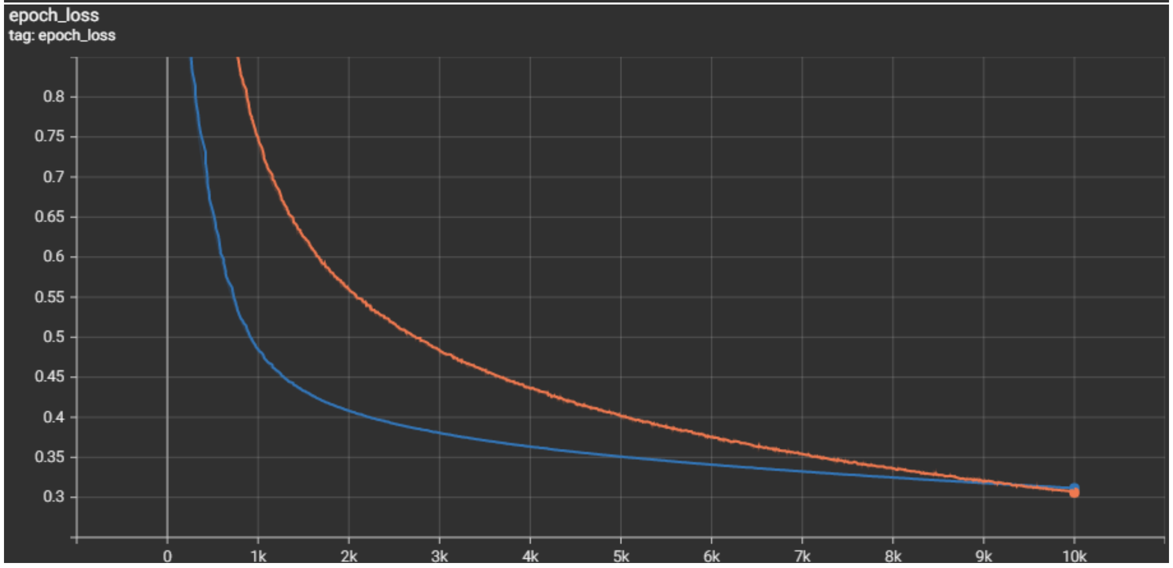
SGD, momentum=0.7, with data generator:



Solution

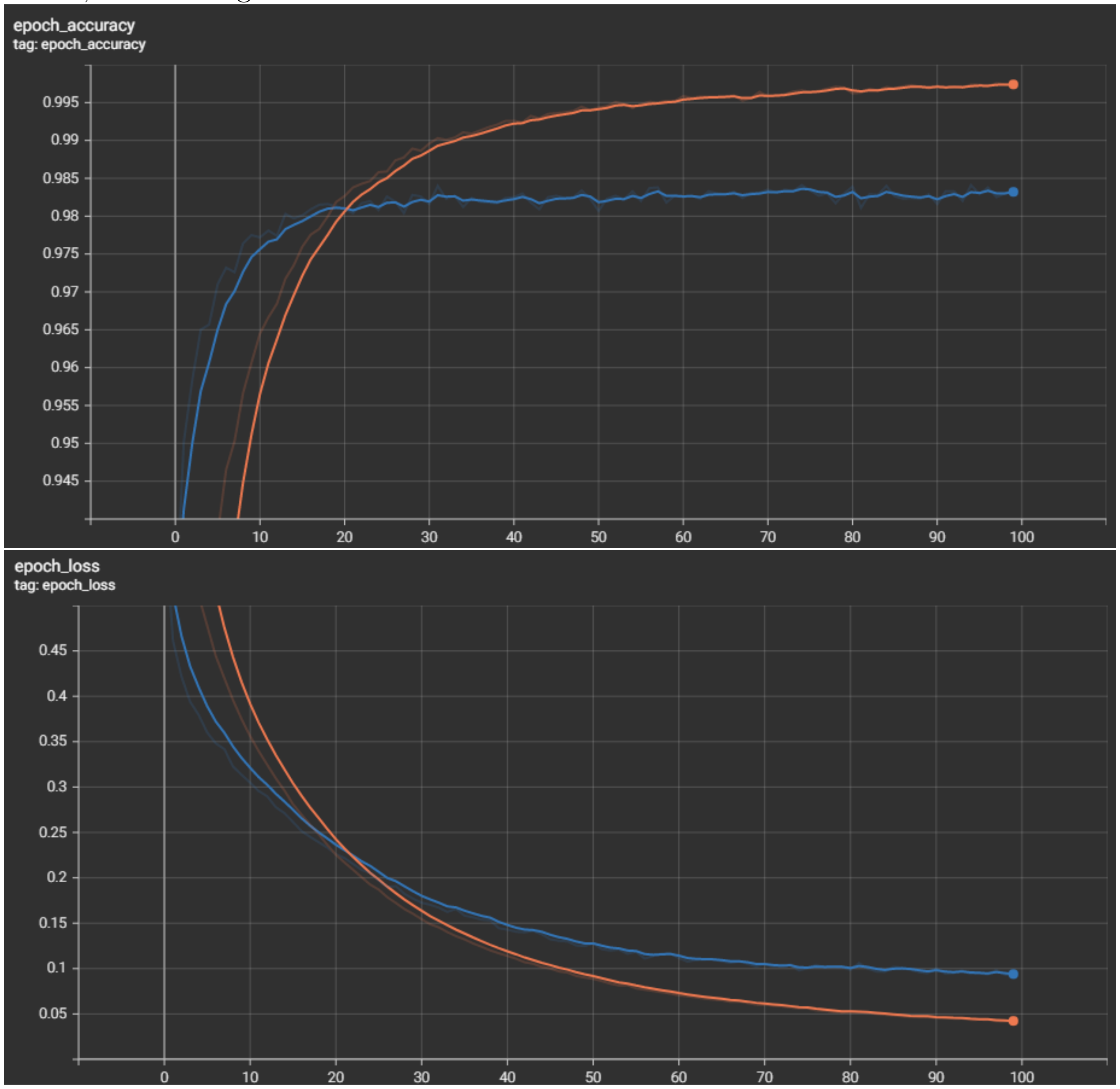
SGD, momentum=0, with data generator:





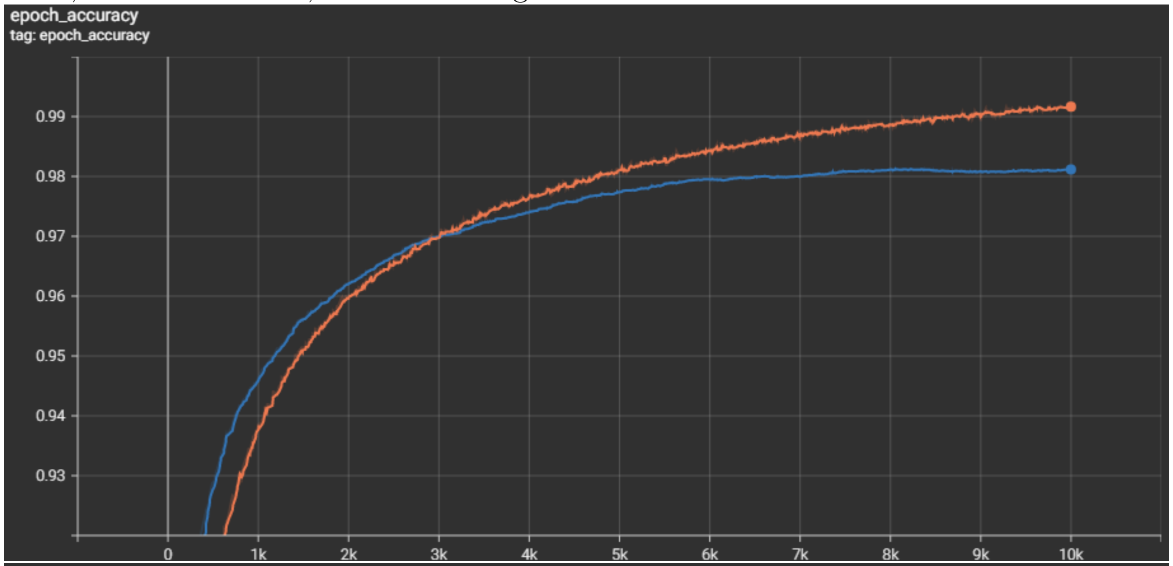
Solution

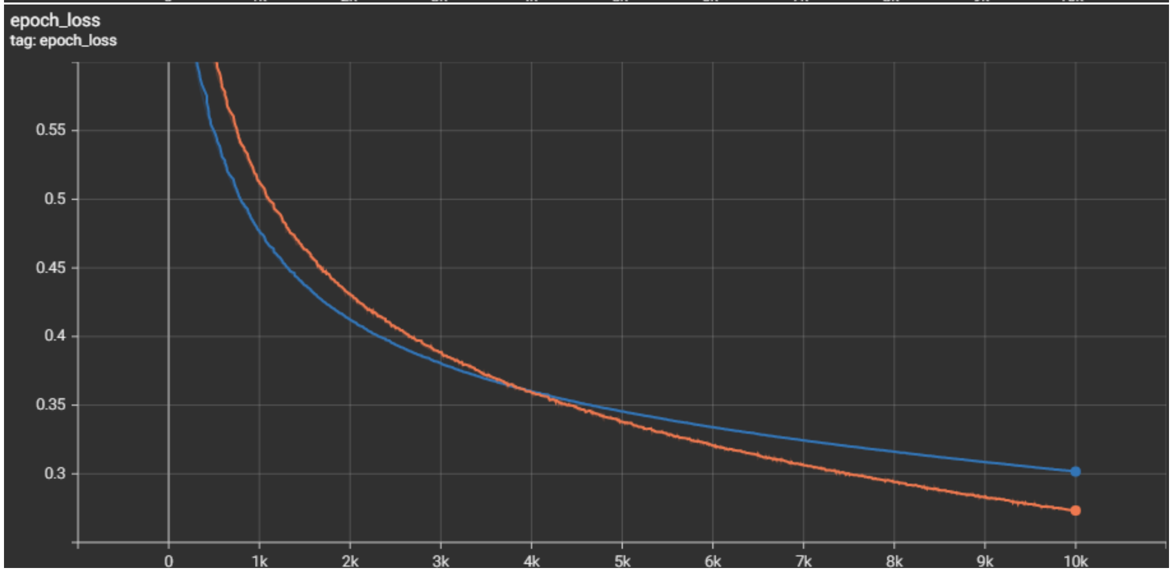
Adam, with data generator:



Solution

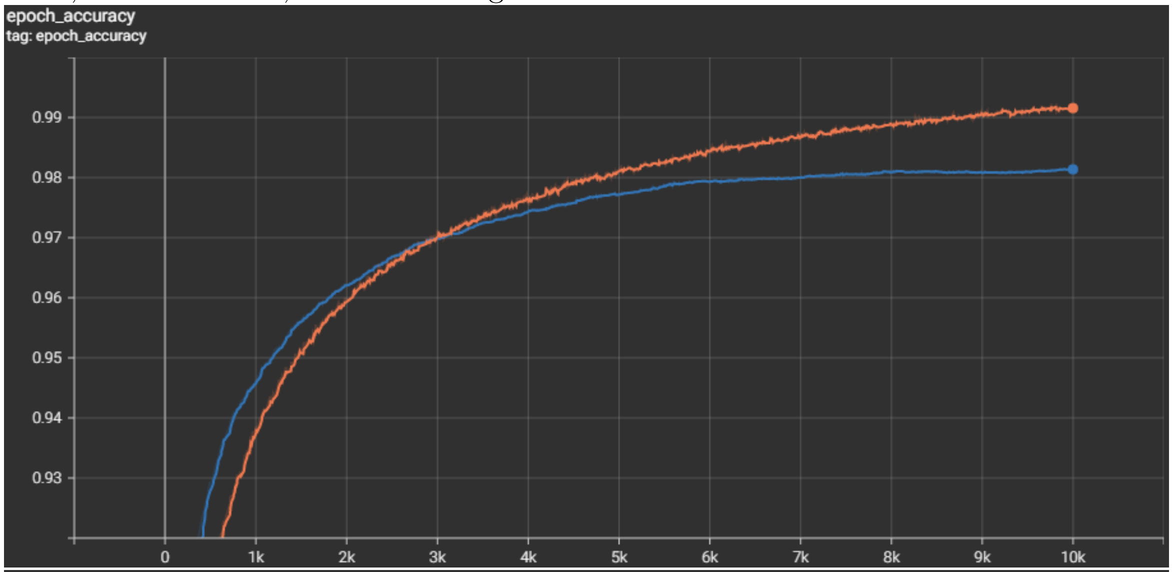
SGD, momentum=0.7, without data generator:

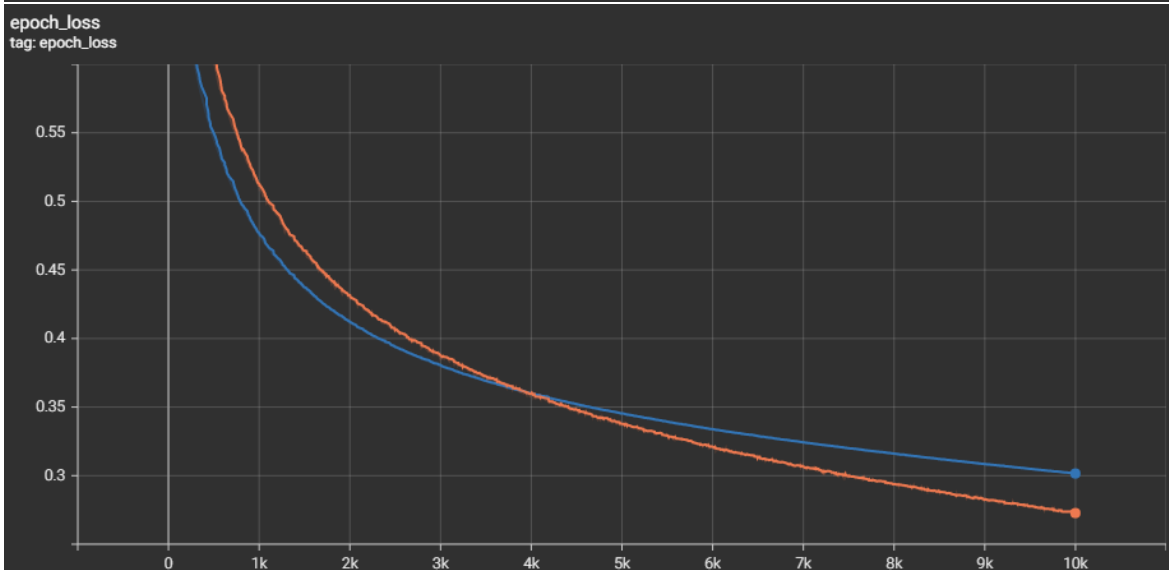




Solution

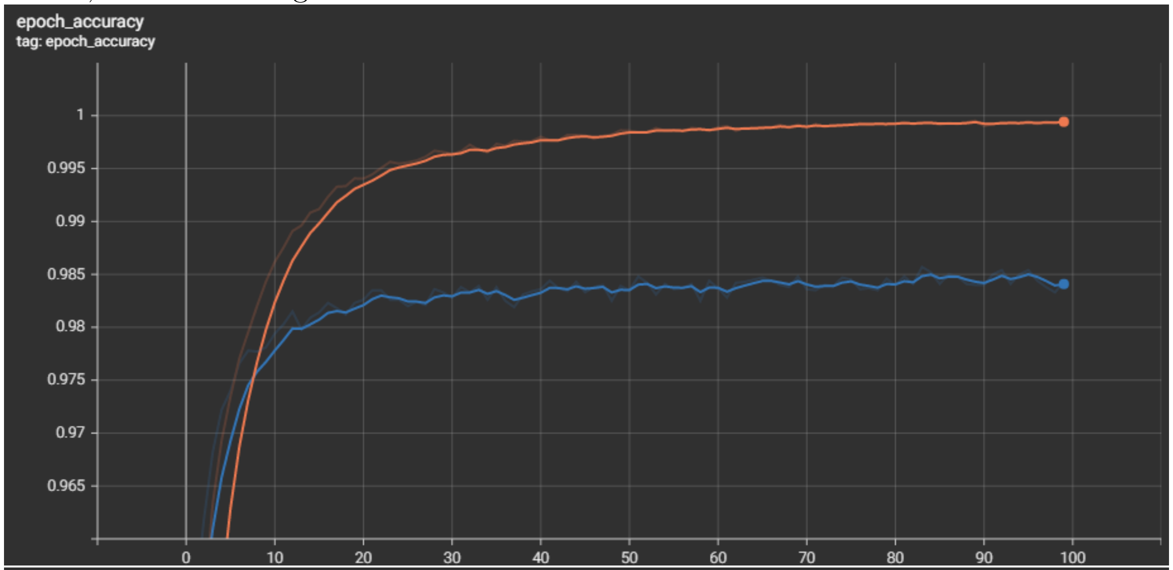
SGD, momentum=0, without data generator:

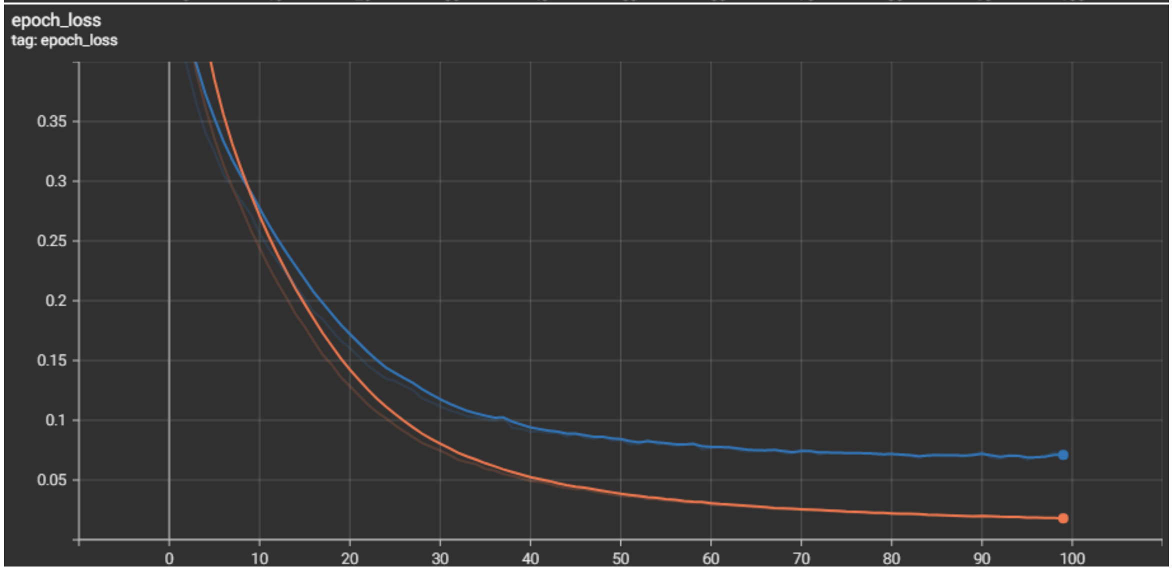




Solution

Adam, without data generator:

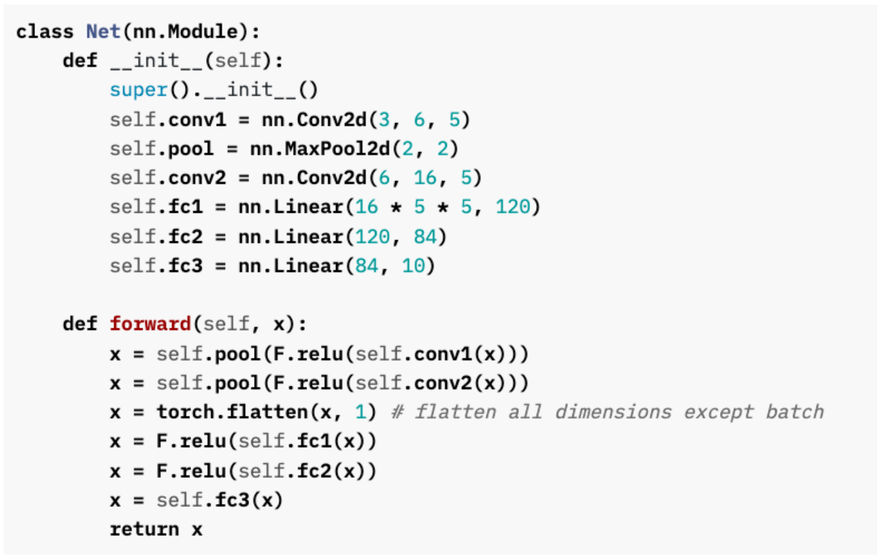




2.2 Image Classification with CNNs (25 points)

In this problem, you are asked to implement a CNN to perform an image classification task. The following requirements shall be met to get the full 25 bonus points.

- The dataset should be the full CIFAR10[11] dataset: *torchvision.datasets.CIFAR10*[12]
- Use the PyTorch framework for training the model.
- Use the **DataLoader** to iterate through your dataset.[13]
- Define the CNN model inheriting the **torch.nn.Module**[14]
- Construct the base convolutional neural network according to the PyTorch tutorial[15]. The structure of the base convolutional neural network is:



```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

- Compare the difference between the base model with and without the ReLU activation.
- Replace the Pooling layer with the convolutional layer that results in the same output feature map size, and compare the performances between them.
- Make the base CNN deeper and compare their performance.

[11] https://www.cs.toronto.edu/~kriz/cifar.html
[12] https://pytorch.org/vision/stable/generated/torchvision.datasets.CIFAR10.html
[13] https://pytorch.org/docs/stable/data.html
[14] https://pytorch.org/docs/stable/generated/torch.nn.Module.html
[15] https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

1. (2 points) Your code of **DataLoader** and iterating the **DataLoader**:

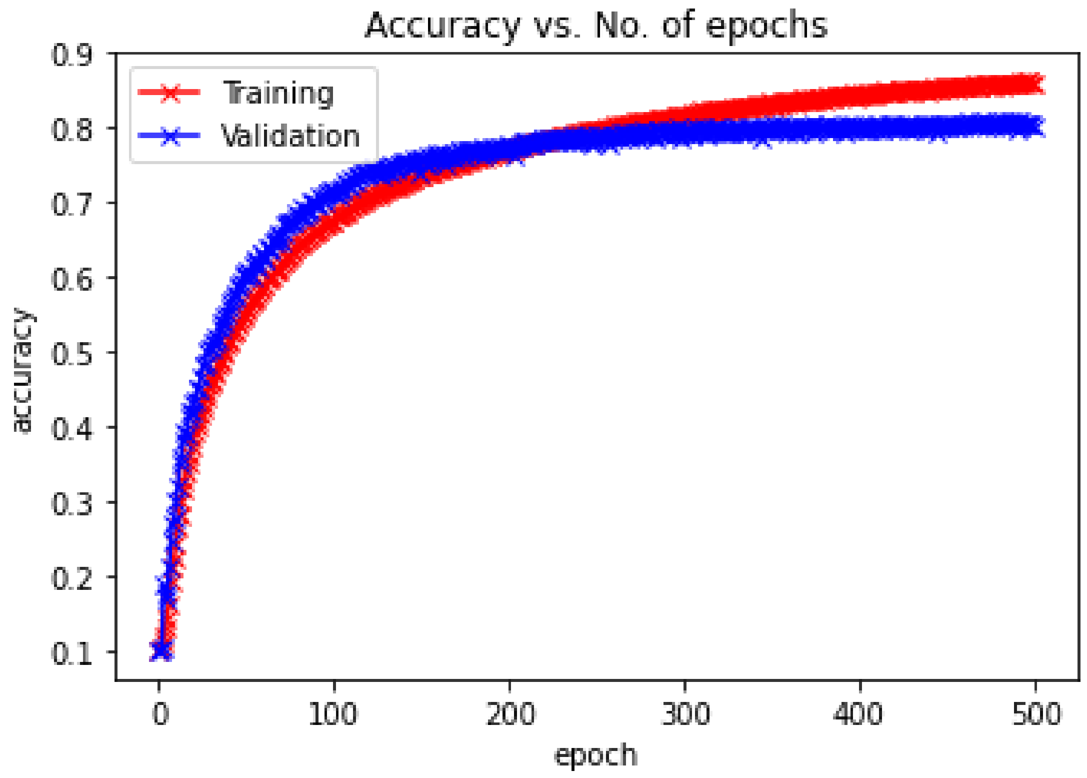
Solution

```
transform = transforms.Compose([transforms.ToTensor(),
transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

train_data = datasets.CIFAR10('data', train=True,
download=True, transform=transform)

test_data = datasets.CIFAR10('data', train=False,
download=True, transform=transform)

num_workers = 0
batch_size = 128

train_loader = DataLoader(dataset=train_data,
batch_size=batch_size, shuffle=True,
num_workers=num_workers)

test_loader = DataLoader(dataset=test_data,
batch_size=batch_size, shuffle=True,
num_workers=num_workers)

train_dataiter = iter(train_loader)
test_dataiter = iter(test_loader)
```

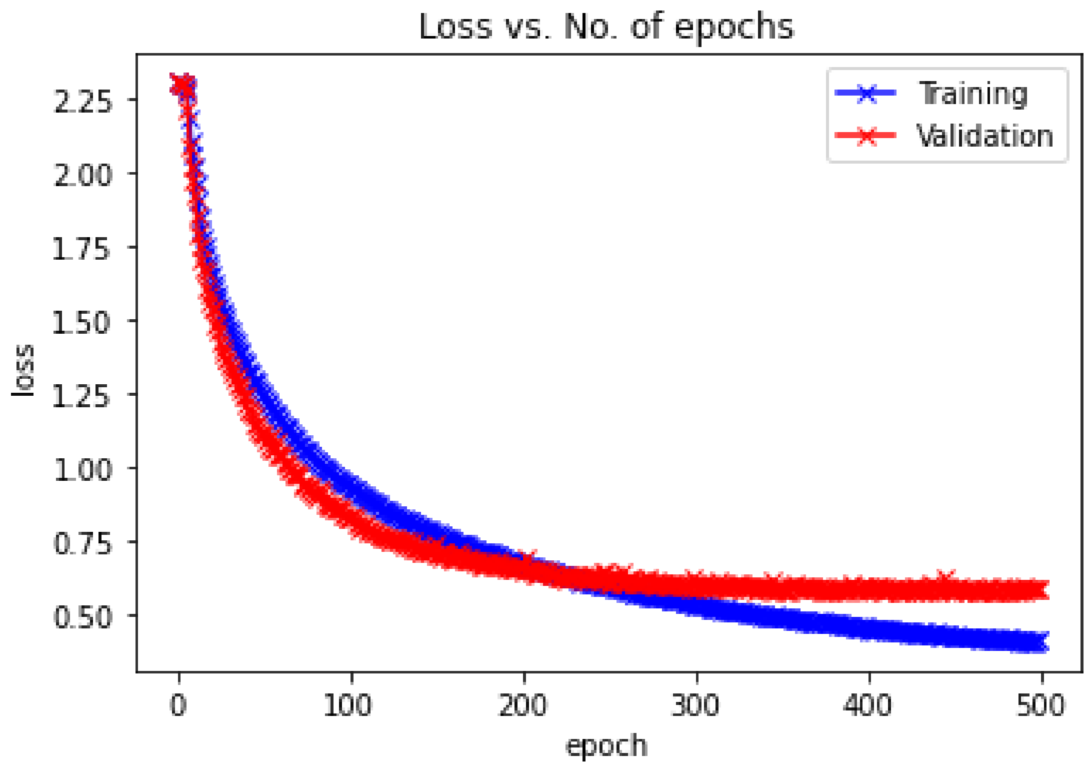
2. (4 points) Report the base CNN's performance (including training and validation losses, training, validation and test accuracy). What's your conclusion about the base CNN model?

Solution

```
Net(
  (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=1024, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=64, bias=True)
  (fc3): Linear(in_features=64, out_features=10, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
)
```



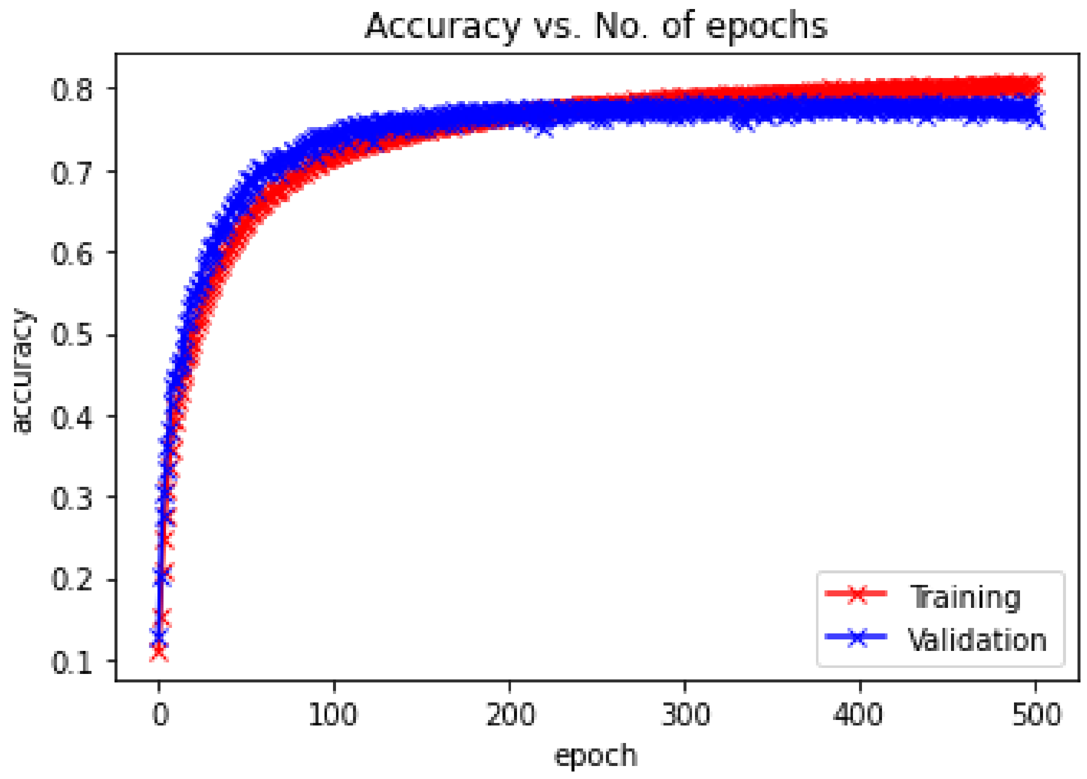
Solution



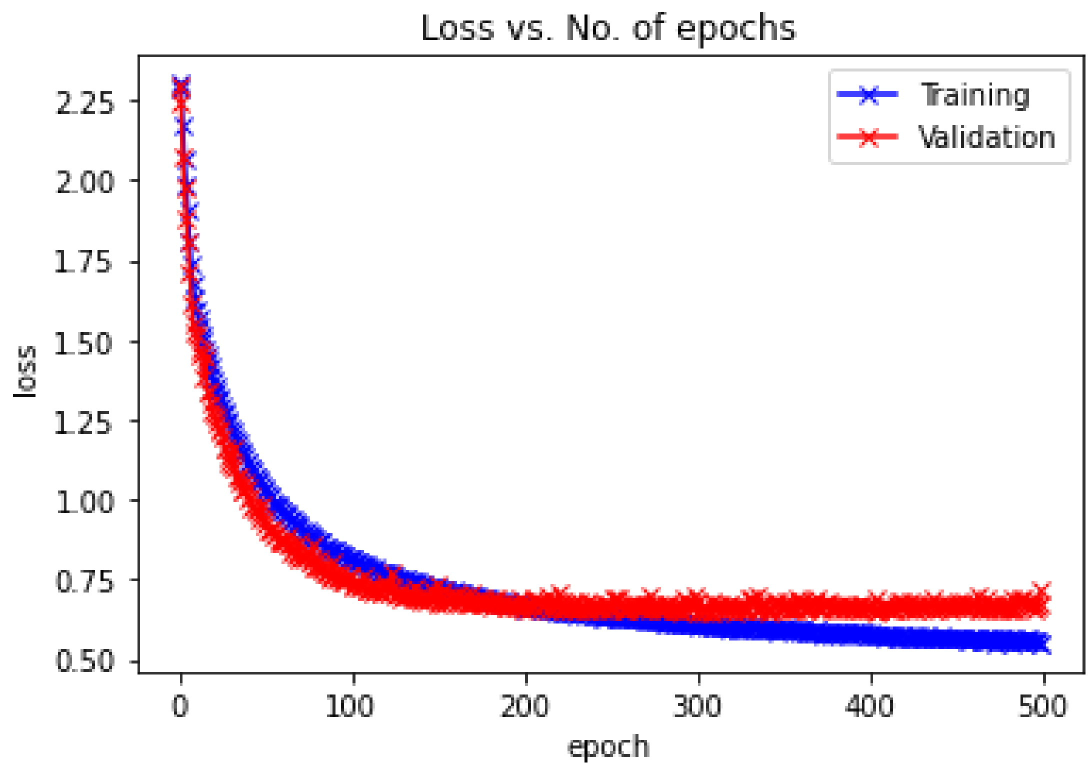
1. Final validation loss is 0.5911966656684875
2. Final validation accuracy is 0.8028
3. Nothing is out of my expection.

3. (5 points) Remove the ReLU activation inside the base CNN model and compare their performances (including training and validation losses, training, validation and test accuracy). What's your conclusion?

Solution



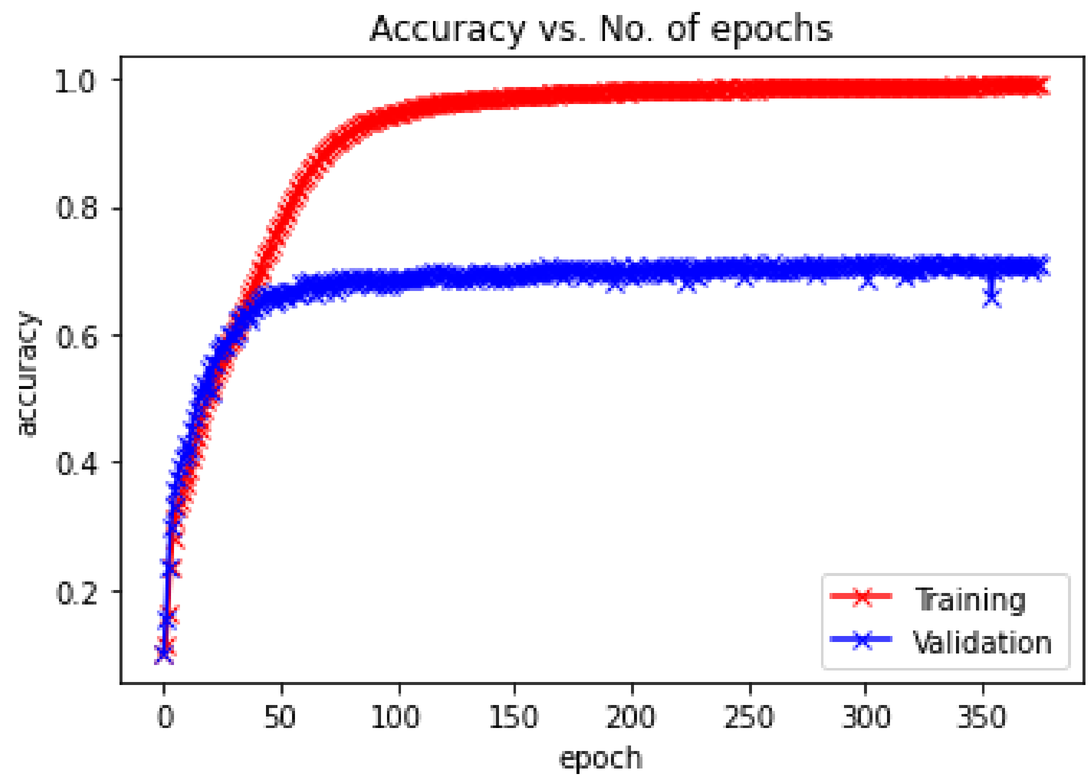
Solution



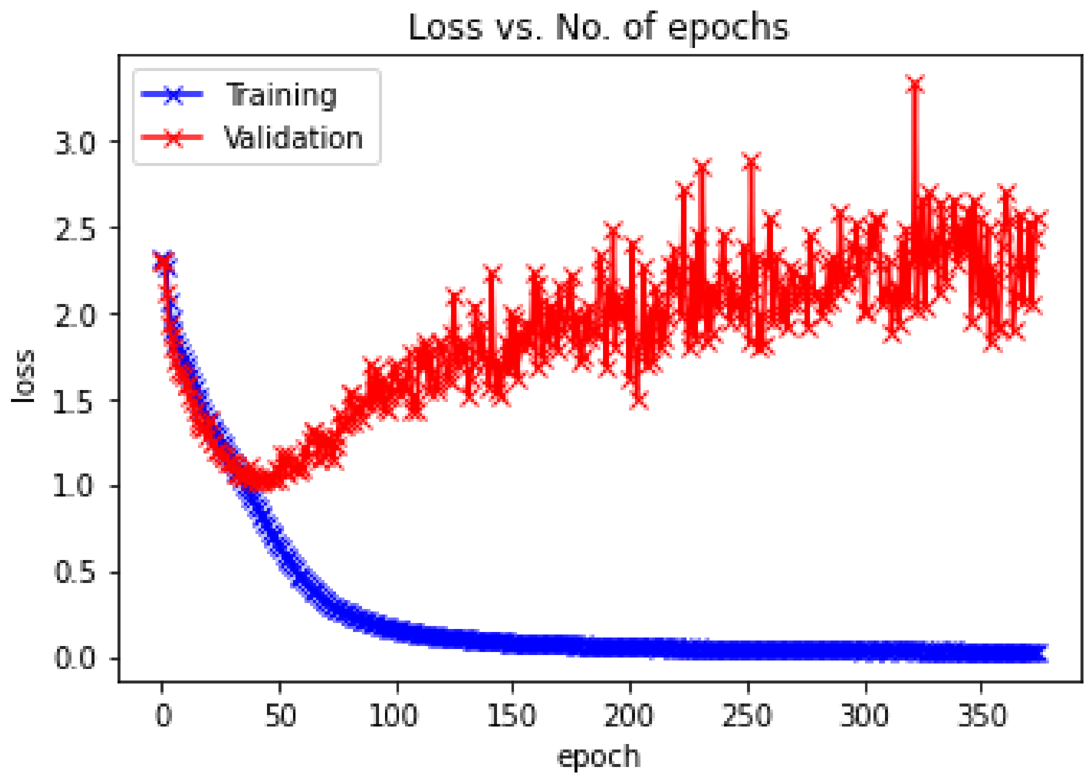
1. Final validation loss is 0.7137349595069885
2. Final validation accuracy is 0.7637
3. The loss is higher and the accuracy is lower. The non-linearity ability is decreased.

4. (7 points) Replace the Pooling layer with the convolutional layer that results in the same output feature map size, and compare the performances between them (including training and validation losses, training, validation and test accuracy). What's your conclusion?

Solution



Solution



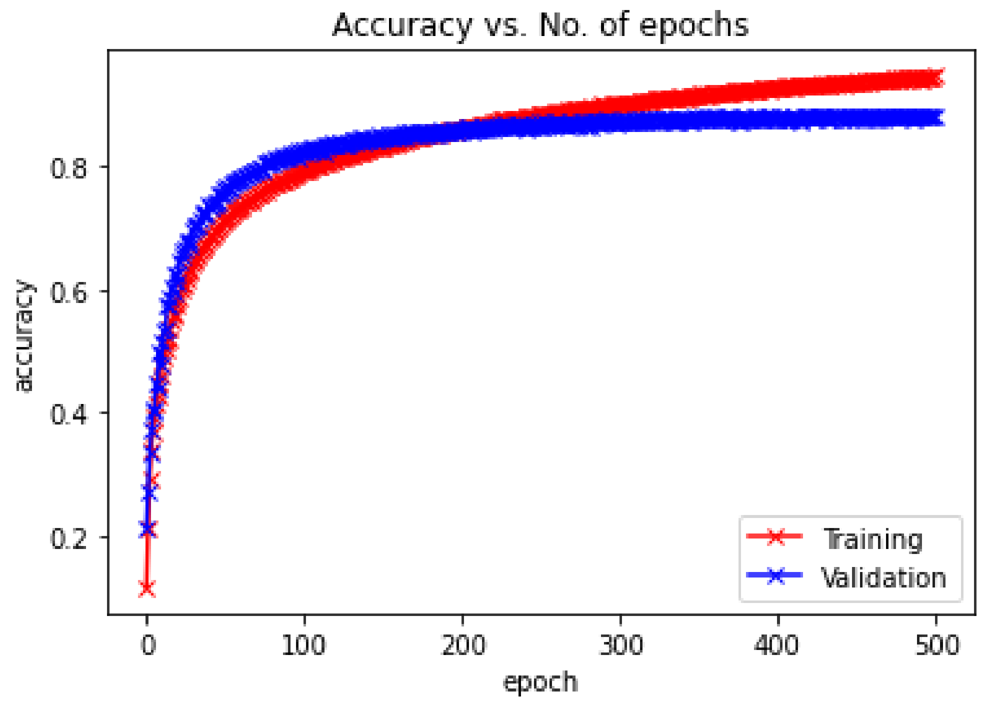
1. Final validation loss is 2.552897264099121
2. Final validation accuracy is 0.7084
3. The model is heavily overfitted. Polling layer will significant reduce the amount of computation as well as enhance the features.

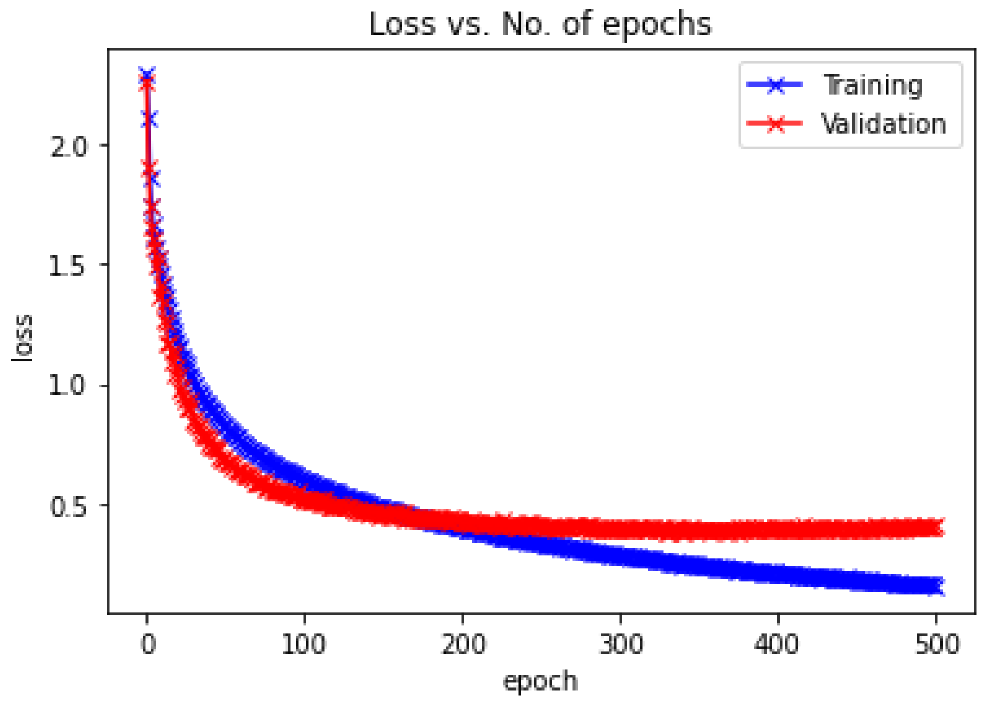
5. (7 points) Make the model deeper and compare the performances between them (including your model structure, training and validation losses, training, validation and test accuracy). What's your conclusion?

Solution

```
CNN(
  (conv_layer): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Dropout2d(p=0.5, inplace=False)
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU(inplace=True)
    (10): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (13): Dropout2d(p=0.5, inplace=False)
    (14): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (16): ReLU(inplace=True)
    (17): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU(inplace=True)
    (19): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (20): Dropout2d(p=0.5, inplace=False)
  )
  (fc_layer): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=4096, out_features=1024, bias=True)
    (2): ReLU(inplace=True)
    (3): Linear(in_features=1024, out_features=512, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

Solution





1. Final validation loss is 0.4078276218414307
2. Final validation accuracy is 0.8783
3. Since the network is deeper, the model's learning ability is stronger. Therefore the loss is obviously lower and the accuracy is higher.

6. (10 points) BONUS: If your currently attained best model is underfitting, try to alter its structure to overfit, otherwise try some regularization techniques. Report the results and what's your conclusion?

Solution

Options to make the model underfitting:

1. Decrease the layers of convolution and full-connection.
2. Increase the number of dropout layers or increase the possibility of dropout layers.
3. Increase the coefficient of regularizers.
4. Use large number of pooling layers.
5. Others.

Options to make the model overfitting:

1. Increase the layers of convolution and full-connection.
2. Decrease the number of dropout layers or decrease the possibility of dropout layers.
3. Decrease the coefficient of regularizers or set regularizer None.
4. Do not use data generator.
5. Do not use pooling layer.
6. Others.

In order to achieve better performance, I also applied vision transformer, convmixer, and other models. Here are the results.

Solution

Collaboration Questions Please answer the following:

After you have completed all other components of this assignment, report your answers to the the following collaboration policy questions.

1. Did you receive any help whatsoever from anyone in solving this assignment? Is so, include full details.
2. Did you give any help whatsoever to anyone in solving this assignment? Is so, include full details.
3. Did you find or come across code that implements any part of this assignment ? If so, include full details even if you have not used that portion of the code.

Solution

1. NO.
2. NO.
3. NO.