TF2-05

Neural Networks

Dong Kook Kim

Training epoch/batch

In the neural network terminology:

one **epoch** = one forward pass and one backward pass of *all* the training examples

batch size = the number of training examples in one forward/backward pass. The higher the batch size, the more memory space you'll need.

number of **iterations** = number of passes, each pass using [batch size] number of examples. To be clear, one pass = one forward pass + one backward pass (we do not count the forward pass and backward pass as two different passes).

Example: if you have *1000 training examples*, and your *batch size is 500*, then it will take 2 iterations to complete 1 epoch.

http://stackoverflow.com/questions/4752626/epoch-vs-iteration-when-training-neural-networks

Neural Network Training : GD

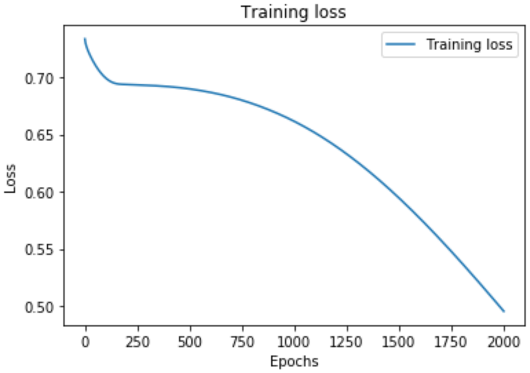
- Using BackPropagation

Exercise 05-5.

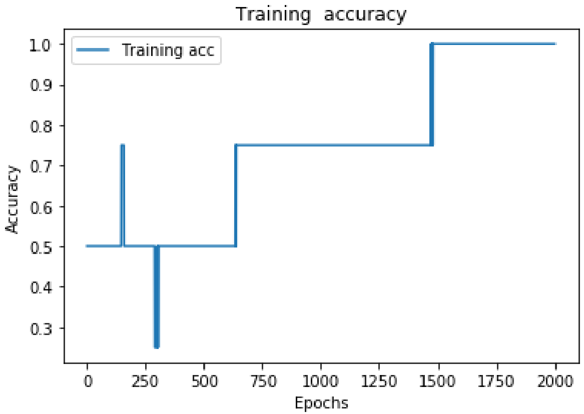
tf2-05-5-xor_nn_2.py

- XOR Problem

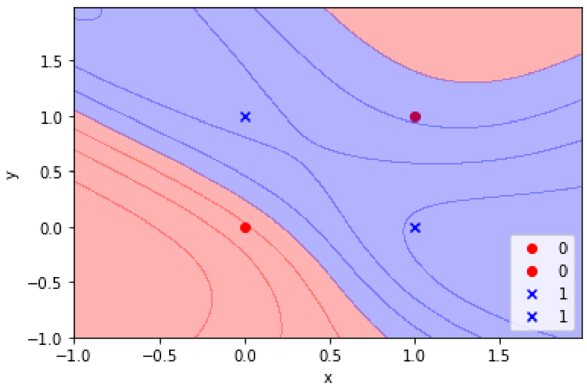
Training Loss



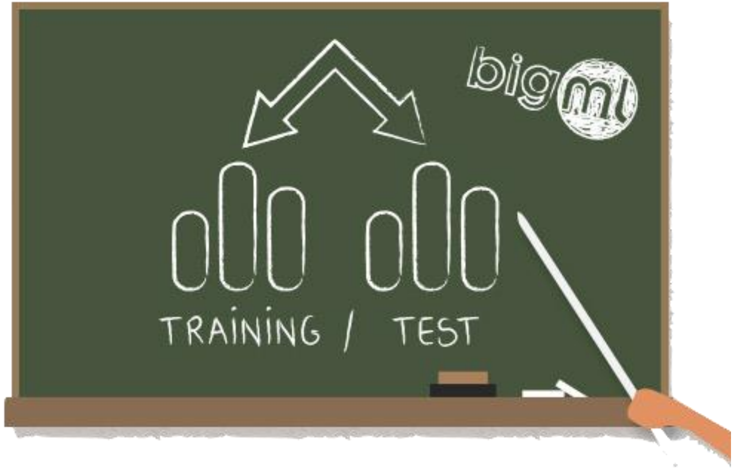
Training Accuracy



Decision Boundary



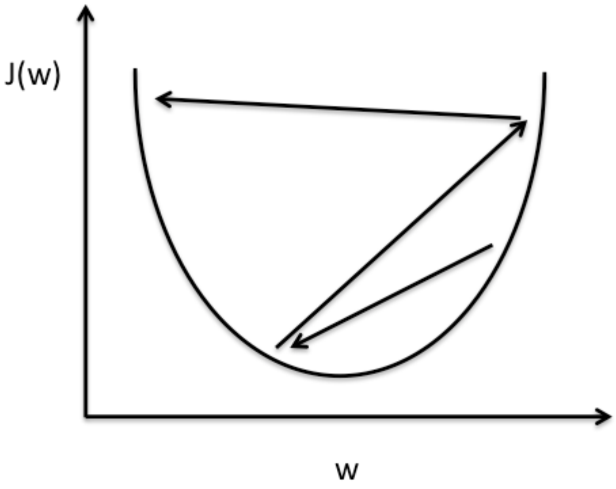
Training and Test datasets



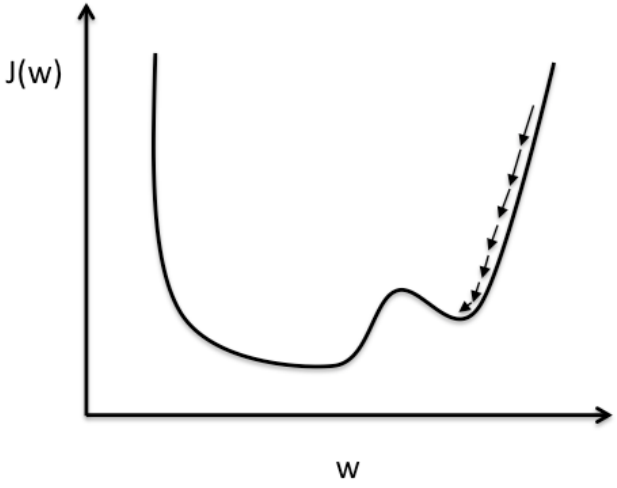
```
x_data = [[1, 2, 1], [1, 3, 2], [1, 3, 4], [1, 5, 5], [1, 7, 5], [1, 2, 5], [1, 6, 6], [1, 7, 7]]
y_data = [[0, 0, 1], [0, 0, 1], [0, 0, 1], [0, 1, 0], [0, 1, 0], [0, 1, 0], [1, 0, 0], [1, 0, 0]]

# Evaluation our model using this test dataset
x_test = [[2, 1, 1], [3, 1, 2], [3, 3, 4]]
y_test = [[0, 0, 1], [0, 0, 1], [0, 0, 1]]
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-1-learning_rate_and_evaluation.py

Learning Rate is Important



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

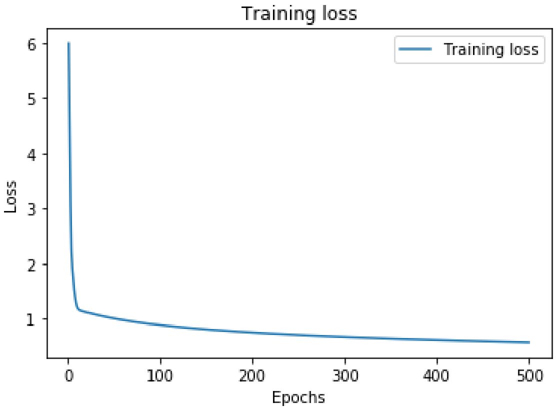
http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

Exercise 05-4.

tf2-05-4-learning_rate.py

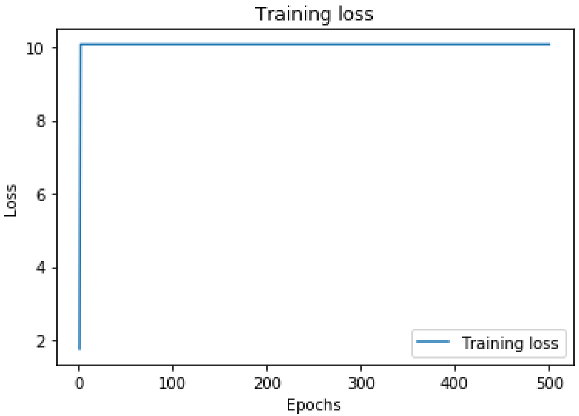
good learning rate

```
sgd = SGD(lr=0.01)
```



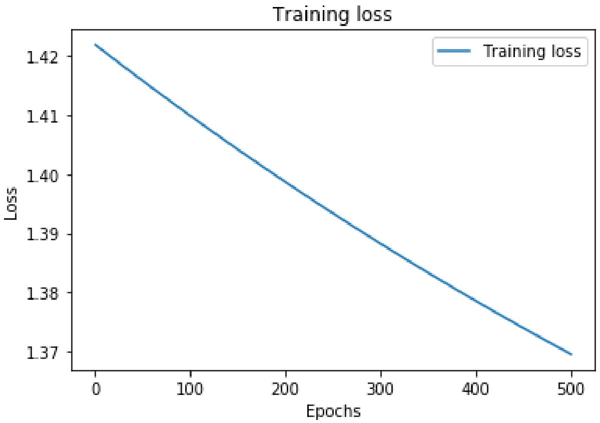
big learning rate

```
sgd = SGD(lr=1.5)
```



small learning rate

```
sgd = SGD(lr=e-5)
```

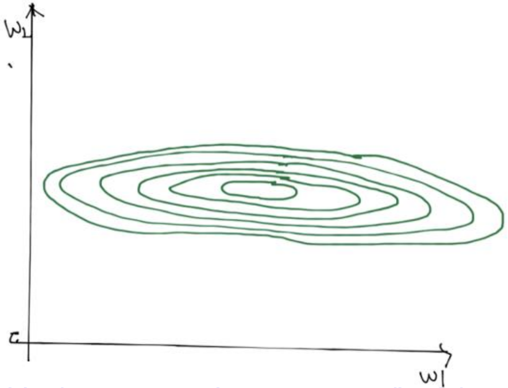


Exercise 05-6.

lab-05-6-linear_regression_without_min_max.py

Non-normalized inputs

```
xy = np.array([[828.659973, 833.450012, 908100, 828.349976, 831.659973],
                [823.02002, 828.070007, 1828100, 821.655029, 828.070007],
                [819.929993, 824.400024, 1438100, 818.97998, 824.159973],
                [816, 820.958984, 1008100, 815.48999, 819.23999],
                [819.359985, 823, 1188100, 818.469971, 818.97998],
                [819, 823, 1198100, 816, 820.450012],
                [811.700012, 815.25, 1098100, 809.780029, 813.669983],
                [809.51001, 816.659973, 1398100, 804.539978, 809.559998]])
```



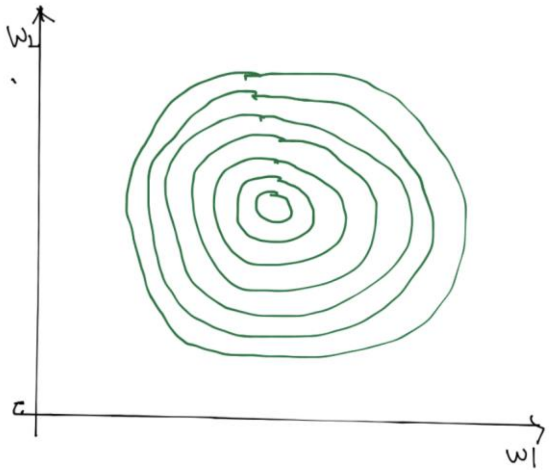
https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-2-linear_regression_without_min_max.py

Normalized inputs (min-max scale)

```
xy = np.array([[828.659973, 833.450012, 908100, 828.349976, 831.659973],
               [823.02002, 828.070007, 1828100, 821.655029, 828.070007],
               [819.929993, 824.400024, 1438100, 818.97998, 824.159973],
               [816, 820.958984, 1008100, 815.48999, 819.23999],
               [819.359985, 823, 1188100, 818.469971, 818.97998],
               [819, 823, 1198100, 816, 820.450012],
               [811.700012, 815.25, 1098100, 809.780029, 813.669983],
               [809.51001, 816.659973, 1398100, 804.539978, 809.559998]])
```

```
xy = MinMaxScaler(xy)
print(xy)
```

```
[[ 0.99999999  0.99999999  0.          1.          1.        ]
 [ 0.70548491  0.70439552  1.          0.71881782  0.83755791]
 [ 0.54412549  0.50274824  0.57608696  0.606468    0.6606331 ]
 [ 0.33890353  0.31368023  0.10869565  0.45989134  0.43800918]
 [ 0.51436     0.42582389  0.30434783  0.58504805  0.42624401]
 [ 0.49556179  0.42582389  0.31521739  0.48131134  0.49276137]
 [ 0.11436064  0.          0.20652174  0.22007776  0.18597238]
 [ 0.          0.07747099  0.5326087   0.          0.        ]]
```

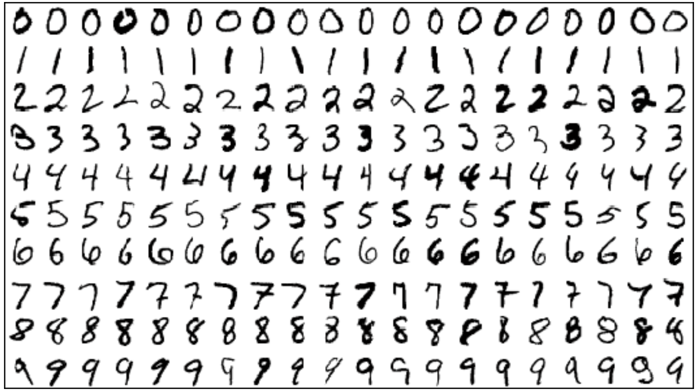


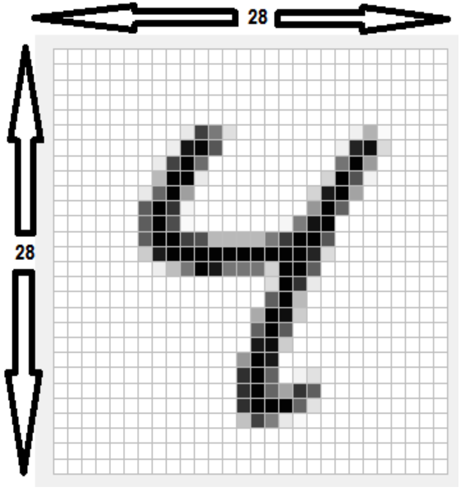
https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-3-linear_regression_min_max.py

Exercise 05-6.

lab-05-6-linear_regression_min_max.py

MNIST Dataset



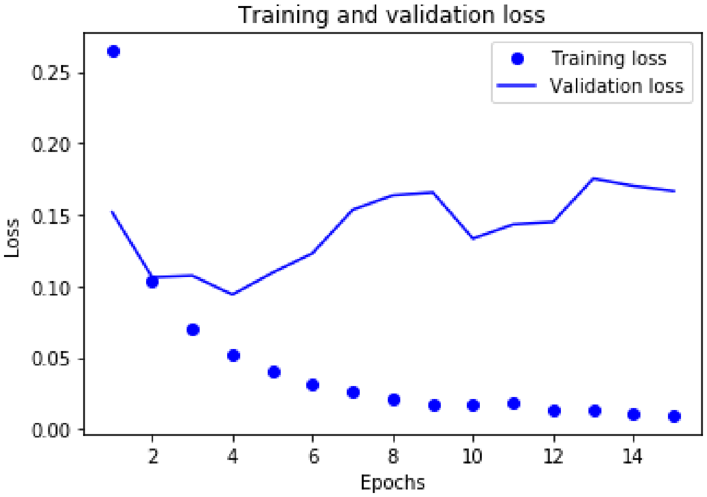


28x28x1 image

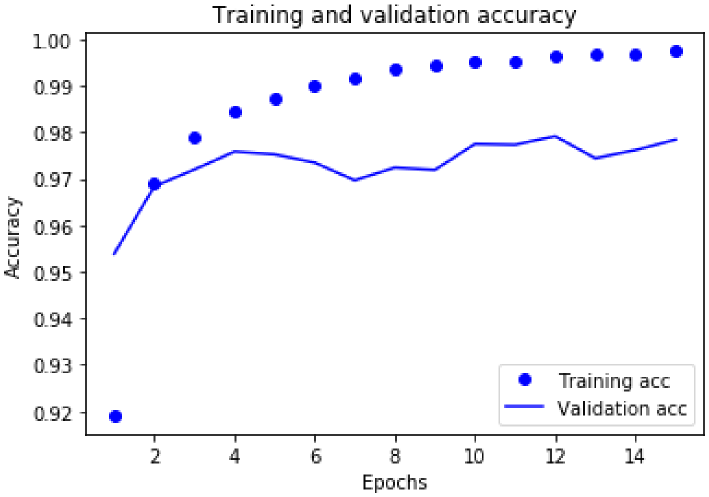
Exercise 05-7.

tf2-05-7-mnist_nn.py

Training Loss



Training Accuracy



Test Accuracy

0.9788

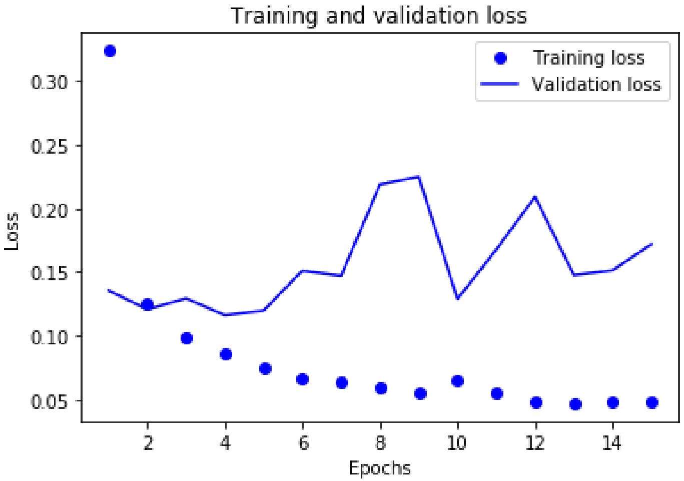
Exercise 05-8.

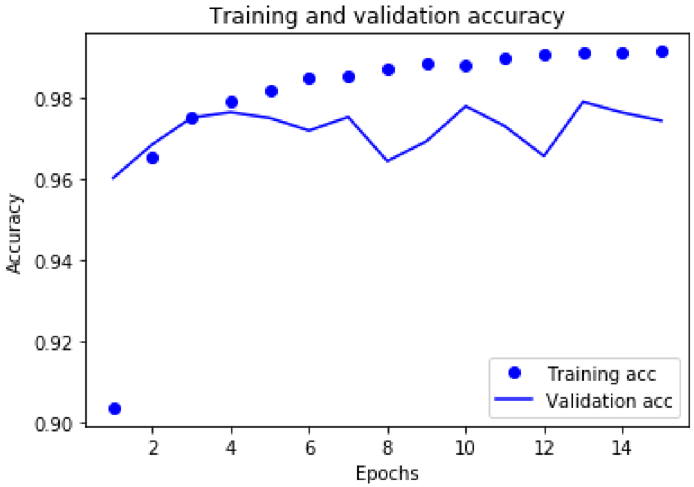
tf2-05-8-mnist_nn_deep.py

Training Loss

Training Accuracy

Test Accuracy

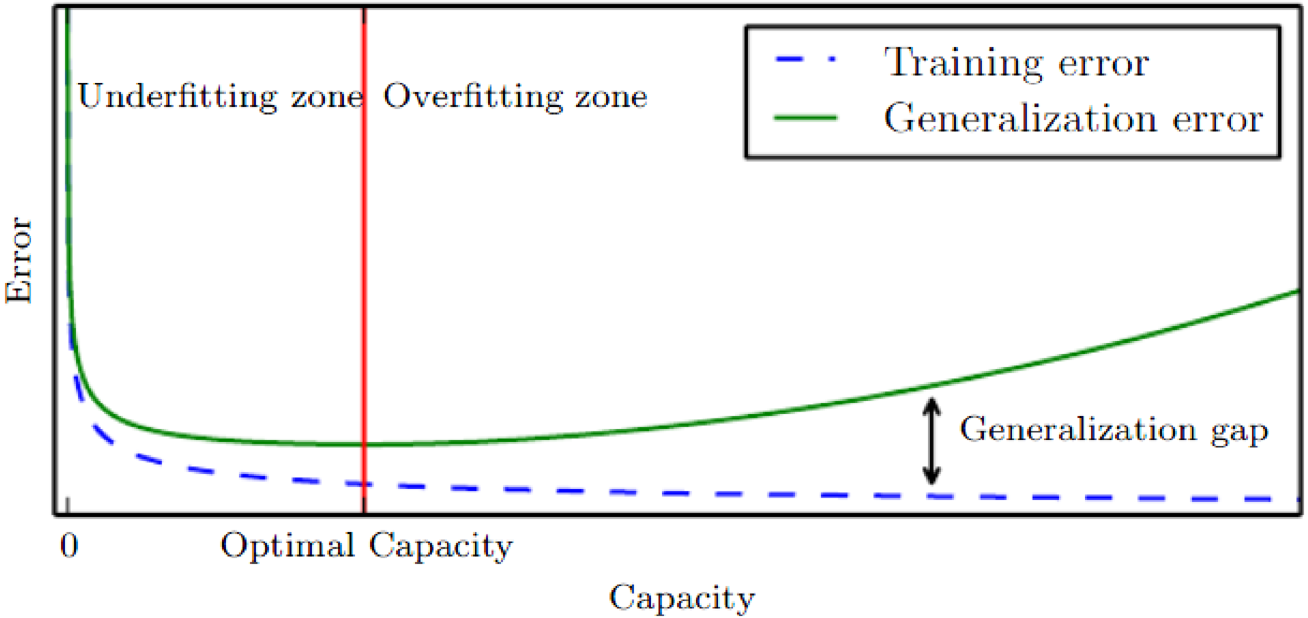




0.9762

Overfitting and Underfitting

- Underfitting : model is not able to obtain a sufficient low error on the training set
- Overfitting: when the gap between the training error and test error is too big



Solutions for Overfitting

- More training data
- Reduce the number of features
- Regularization
  - L2, L1 regularization
  - dropout

Exercise 05-10.

tf2-05-10-imdb_overfitting.py

Baseline Models

```
keras.layers.Dense(16, activation='relu', input_shape=(10000,)),
keras.layers.Dense(16, activation='relu'),
keras.layers.Dense(1, activation='sigmoid')
```

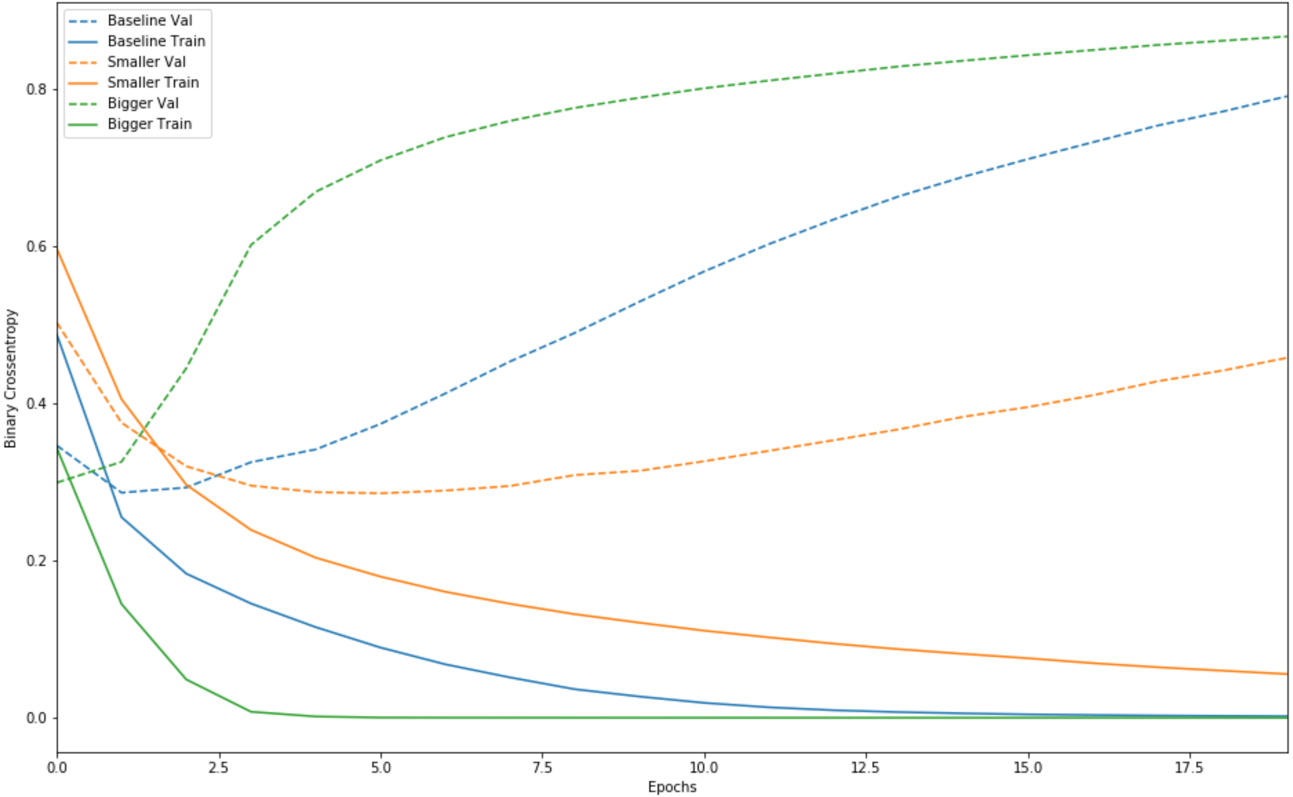
Smaller Models

```
keras.layers.Dense(4, activation='relu', input_shape=(10000,)),
keras.layers.Dense(4, activation='relu'),
keras.layers.Dense(1, activation='sigmoid')
```

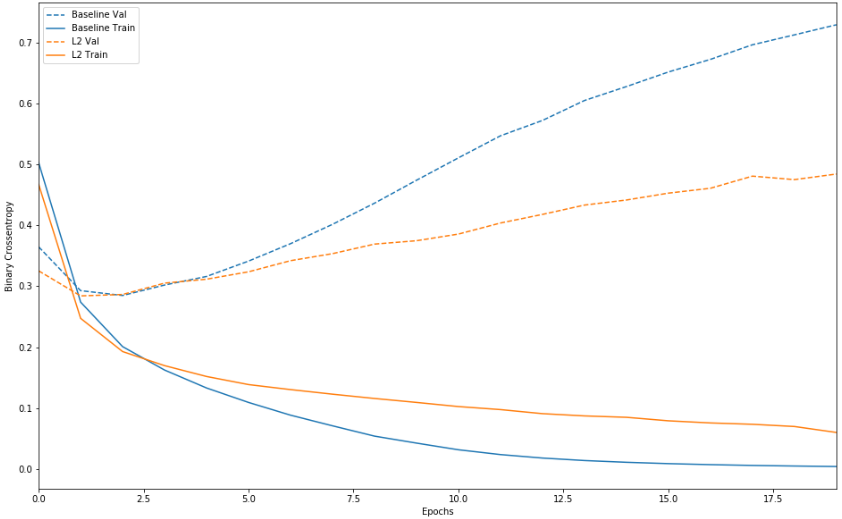
Bigger Models

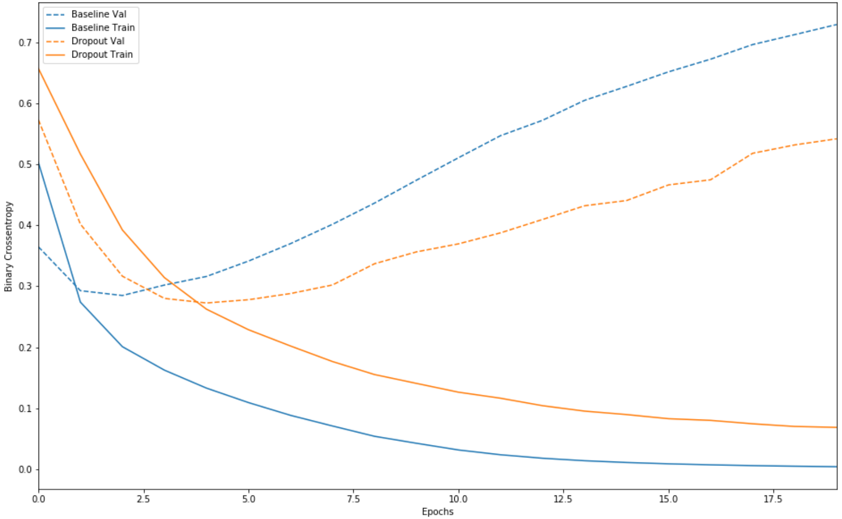
```
keras.layers.Dense(512, activation='relu', input_shape=(10000,)),
keras.layers.Dense(512, activation='relu'),
keras.layers.Dense(1, activation='sigmoid')
```

Training and Validation Losses



L2 Regularization and Dropout





Exercise 05-9.

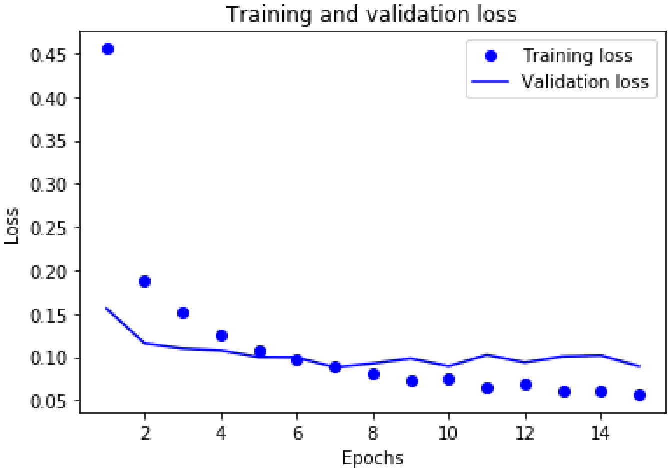
tf2-05-9-mnist_nn_dropout.py

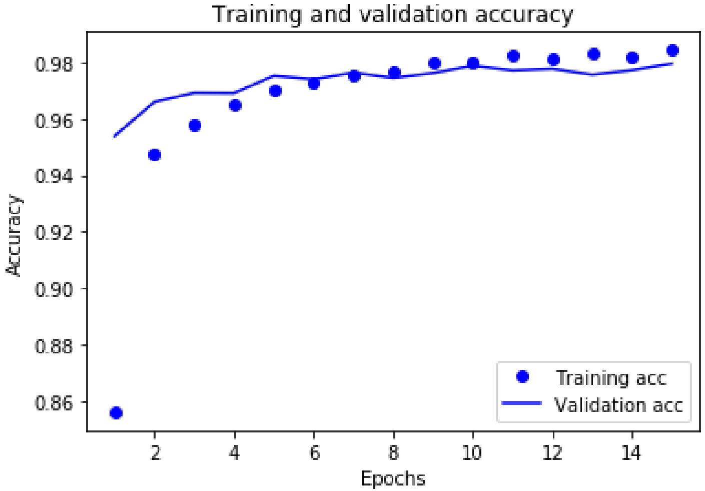
Training Loss

Training Accuracy

Test Accuracy

0.9822


Training and validation loss
Training loss
Validation loss
Loss
Epochs


Training and validation accuracy
Training acc
Validation acc
Accuracy
Epochs