```python
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py


from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 9s 155us/step - loss: 0.2730 - acc:
Epoch 2/12
60000/60000 [==============================] - 9s 145us/step - loss: 0.0909 - acc:
Epoch 3/12
60000/60000 [==============================] - 9s 145us/step - loss: 0.0674 - acc:
Epoch 4/12
60000/60000 [==============================] - 9s 145us/step - loss: 0.0556 - acc:
Epoch 5/12
60000/60000 [==============================] - 9s 145us/step - loss: 0.0471 - acc:
Epoch 6/12
60000/60000 [==============================] - 9s 145us/step - loss: 0.0433 - acc:
Epoch 7/12
60000/60000 [==============================] - 9s 145us/step - loss: 0.0390 - acc:
Epoch 8/12
60000/60000 [==============================] - 9s 145us/step - loss: 0.0347 - acc:
Epoch 9/12
60000/60000 [==============================] - 9s 145us/step - loss: 0.0320 - acc:
Epoch 10/12
60000/60000 [==============================] - 9s 145us/step - loss: 0.0284 - acc:
Epoch 11/12
60000/60000 [==============================] - 9s 145us/step - loss: 0.0273 - acc:
```

```python
import matplotlib.pyplot as plt
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```

```
Test score: 0.026977979276103632
Test accuracy: 0.9917
```

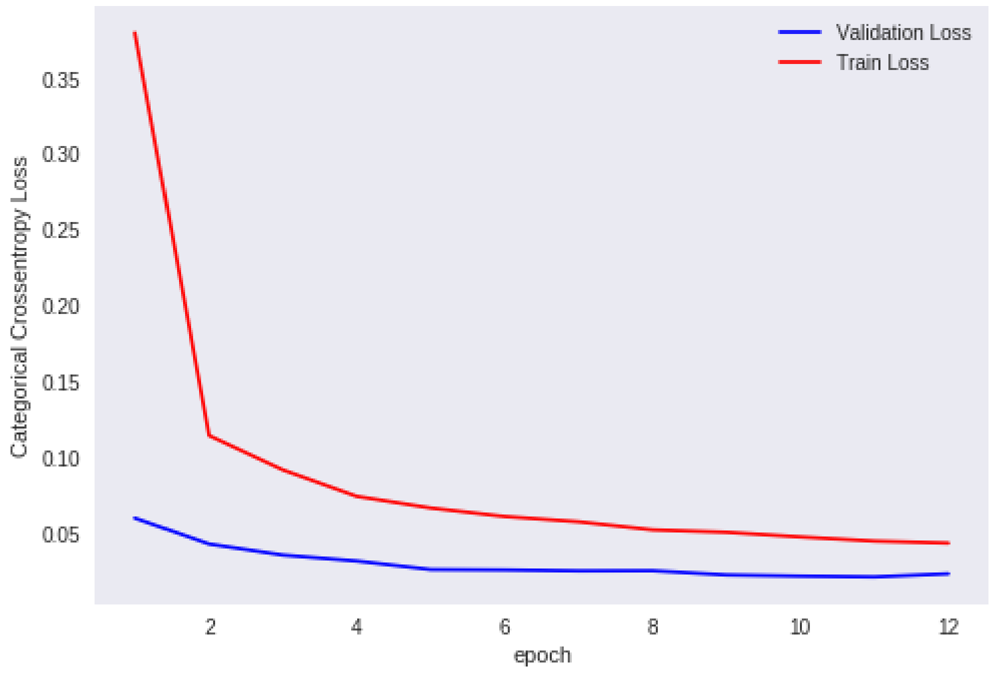
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape, strides=(1, 1), padding = 'same'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 9s 143us/step - loss: 0.3799 - acc:
Epoch 2/12
60000/60000 [==============================] - 8s 132us/step - loss: 0.1151 - acc:
Epoch 3/12
60000/60000 [==============================] - 8s 132us/step - loss: 0.0924 - acc:
Epoch 4/12
60000/60000 [==============================] - 8s 132us/step - loss: 0.0750 - acc:
Epoch 5/12
60000/60000 [==============================] - 8s 132us/step - loss: 0.0674 - acc:
Epoch 6/12
60000/60000 [==============================] - 8s 132us/step - loss: 0.0618 - acc:
Epoch 7/12
60000/60000 [==============================] - 8s 132us/step - loss: 0.0583 - acc:
Epoch 8/12
60000/60000 [==============================] - 8s 132us/step - loss: 0.0530 - acc:
Epoch 9/12
60000/60000 [==============================] - 8s 132us/step - loss: 0.0514 - acc:
Epoch 10/12
60000/60000 [==============================] - 8s 132us/step - loss: 0.0483 - acc:
Epoch 11/12
60000/60000 [==============================] - 8s 133us/step - loss: 0.0456 - acc:
Epoch 12/12
60000/60000 [==============================] - 8s 132us/step - loss: 0.0443 - acc:
Test loss: 0.024088586140600093
Test accuracy: 0.9926
```

```
import matplotlib.pyplot as plt
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = history.history['val_loss']
```

```
ty = history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```

```
Test score: 0.024088586140600093
Test accuracy: 0.9926
```

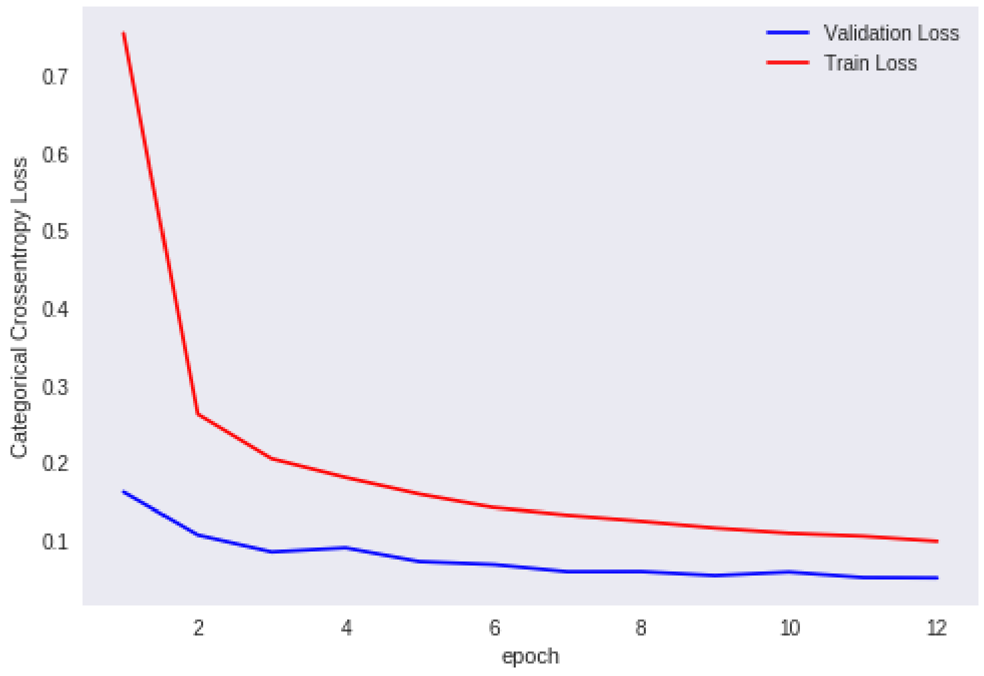


```
model = Sequential()
model.add(Conv2D(32, kernel_size=(7, 7),
                 activation='relu',
                 input_shape=input_shape, strides=(2, 2), padding = 'same'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 6s 101us/step - loss: 0.7557 - acc:
Epoch 2/12
60000/60000 [==============================] - 5s 90us/step - loss: 0.2626 - acc: 0
Epoch 3/12
60000/60000 [==============================] - 5s 90us/step - loss: 0.2049 - acc: 0
Epoch 4/12
60000/60000 [==============================] - 5s 90us/step - loss: 0.1807 - acc: 0
Epoch 5/12
60000/60000 [==============================] - 5s 89us/step - loss: 0.1592 - acc: 0
Epoch 6/12
60000/60000 [==============================] - 5s 90us/step - loss: 0.1419 - acc: 0
Epoch 7/12
60000/60000 [==============================] - 5s 90us/step - loss: 0.1321 - acc: 0
Epoch 8/12
60000/60000 [==============================] - 5s 90us/step - loss: 0.1243 - acc: 0
Epoch 9/12
60000/60000 [==============================] - 5s 90us/step - loss: 0.1156 - acc: 0
Epoch 10/12
```

```
import matplotlib.pyplot as plt
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```

```
Test score: 0.05100429020554293
Test accuracy: 0.9837
```

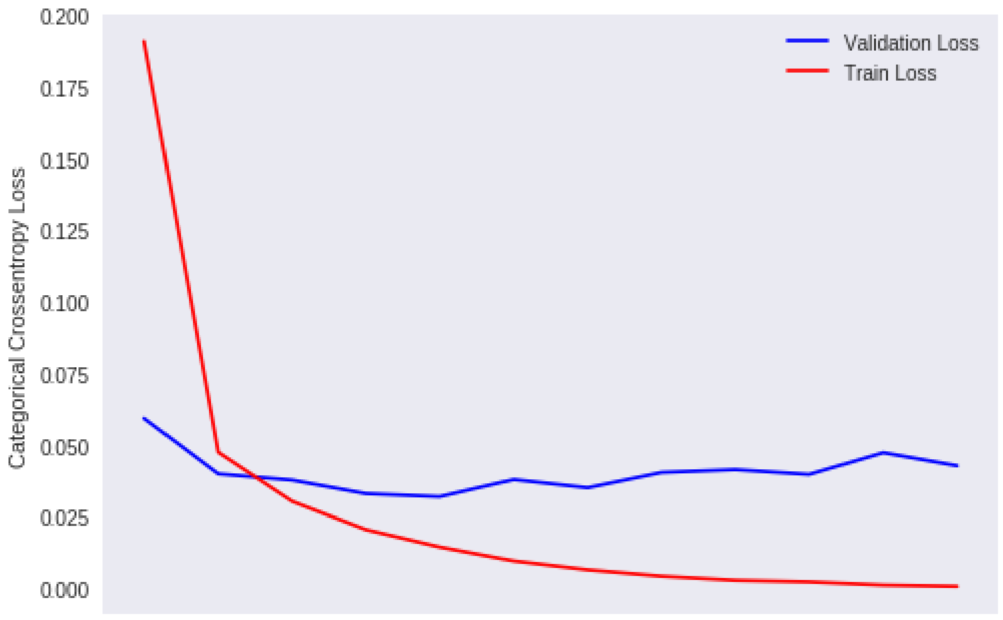


```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 9s 147us/step - loss: 0.1909 - acc:
Epoch 2/12
60000/60000 [==============================] - 8s 134us/step - loss: 0.0474 - acc:
Epoch 3/12
60000/60000 [==============================] - 8s 133us/step - loss: 0.0303 - acc:
Epoch 4/12
60000/60000 [==============================] - 8s 134us/step - loss: 0.0204 - acc:
Epoch 5/12
60000/60000 [==============================] - 8s 135us/step - loss: 0.0143 - acc:
Epoch 6/12
60000/60000 [==============================] - 8s 134us/step - loss: 0.0095 - acc:
Epoch 7/12
60000/60000 [==============================] - 8s 134us/step - loss: 0.0064 - acc:
Epoch 8/12
60000/60000 [==============================] - 8s 134us/step - loss: 0.0042 - acc:
Epoch 9/12
60000/60000 [==============================] - 8s 134us/step - loss: 0.0027 - acc:
Epoch 10/12
60000/60000 [==============================] - 8s 134us/step - loss: 0.0022 - acc:
Epoch 11/12
60000/60000 [==============================] - 8s 134us/step - loss: 0.0011 - acc:
Epoch 12/12
60000/60000 [==============================] - 8s 134us/step - loss: 6.3185e-04 - a
Test loss: 0.04266088300381594
Test accuracy: 0.9902
```

```
import matplotlib.pyplot as plt
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```

```
Test score: 0.04266088300381594
Test accuracy: 0.9902
```

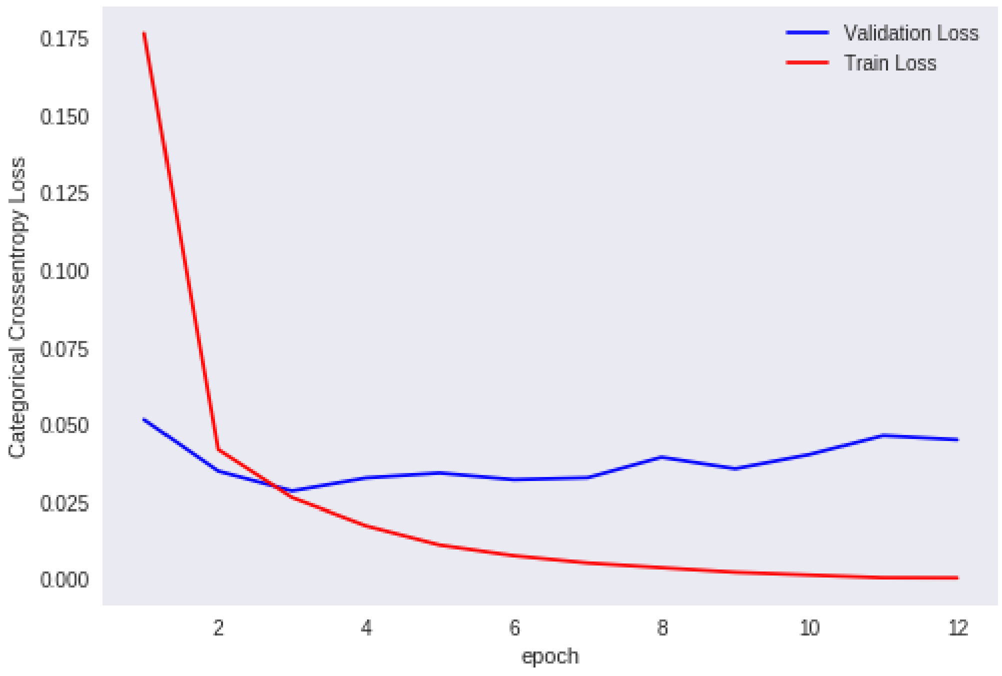


```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 13s 220us/step - loss: 0.1764 - acc:
Epoch 2/12
60000/60000 [==============================] - 12s 205us/step - loss: 0.0419 - acc:
Epoch 3/12
60000/60000 [==============================] - 12s 205us/step - loss: 0.0264 - acc:
Epoch 4/12
```

```
import matplotlib.pyplot as plt
score = model.evaluate(x_test, y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = history.history['val_loss']
ty = history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```

```
Test score: 0.04509090457339796
Test accuracy: 0.9914
```



```
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Architecture", "Accuracy"]

x.add_row(["2 conv layers, kernel size = 3*3, with dropout", 99.17])
x.add_row(["2 conv layers, kernel size = 5*5, 3 max-pooling layers, strides = 1*1, padding
x.add_row(["2 conv layers, kernel size = 7*7, 3 max-pooling layers, strides = 2*2, padding
x.add_row(["2 conv layers, kernel size = 3*3, without droputs", 99.02])
x.add_row(["3 conv layers, without dropout", 99.14])

print(x)
```

```
+--------------------------------------------------------------------------------
|                                                          Architecture
+--------------------------------------------------------------------------------
|                                         2 conv layers, kernel size = 3*3, with dr
| 2 conv layers, kernel size = 5*5, 3 max-pooling layers, strides = 1*1, padding = 
| 2 conv layers, kernel size = 7*7, 3 max-pooling layers, strides = 2*2, padding = 
```

Procedure followed

1. Splitted the MNIST dataset into train and test 2 .Converts a class vector (integers) to binary class matrix 3.Tried different architectures of CNN with dataset like with/without dropout, diffent kernel size, different convolution layers
2. Plotted the epoch vs Train/Test loss of each model