

▼ Bhagyashree Manoj Lambat

Week3&4: task 2 create a digit-recognizer

As a second task assigned by SYNC INTERN'S SYNC INTERN'S at  
Artificial Intelligence intern program

```
import pandas as pd
import numpy as np
import seaborn as sns
import random
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import tensorflow as tf
import tensorflow.keras
from tensorflow import keras
from sklearn.model_selection import train_test_split

from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, BatchNormalization, MaxPool2D, Flatten, Dense, Dropout
import warnings

sns.set()
%matplotlib inline
warnings.filterwarnings("ignore")

train="/kaggle/input/digit-recognizer/train.csv"
test="/kaggle/input/digit-recognizer/test.csv"

df_train = pd.read_csv(train)
df_train.head()
```

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel1774	pixel1775	pixel1776	pixel1777	pixel17
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	

5 rows × 785 columns

```
df_test = pd.read_csv(test)
df_test.head()
```

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel1774	pixel1775	pixel1776	pixel1777	pixel
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	

5 rows × 784 columns

```
df_train.shape
```

(42000, 785)

```
df_test.shape
```

```
(28000, 784)
```

```
df_train['label'].unique()
```

```
array([1, 0, 4, 7, 3, 5, 8, 9, 2, 6])
```

```
df_train['label'].nunique()
```

```
10
```

```
y_train = df_train['label']
```

```
df_train.drop(['label'], axis=1, inplace=True)
```

```
df_train.head()
```

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777	pixel
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	

```
5 rows × 784 columns
```

```
df_train=df_train/255.0
```

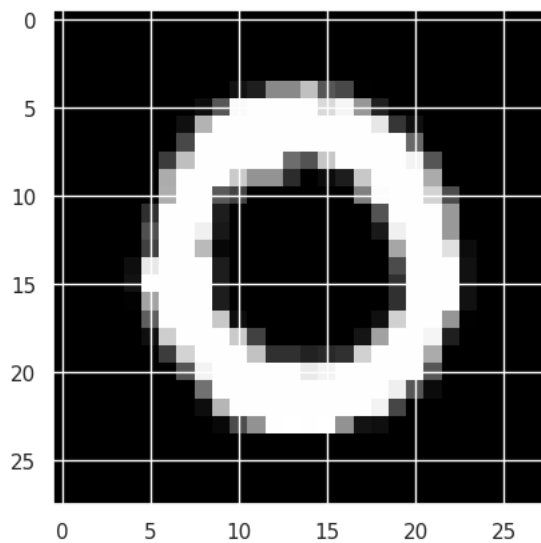
```
df_train=df_train.values.reshape(-1,28,28,1)
```

```
X_train, X_val, y_train, y_val = train_test_split(df_train,y_train, test_size=0.2, random_state=42,shuffle=True)
```

```
img = np.reshape(df_train[1], (28, 28))
```

```
plt.imshow(img, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x70d1882507c0>
```



```
size = 28
channels = 1
batch = 128
epochs = 100
```

```
alphabet="0123456789"
```

```
plt.figure(figsize=(15, 15))
```

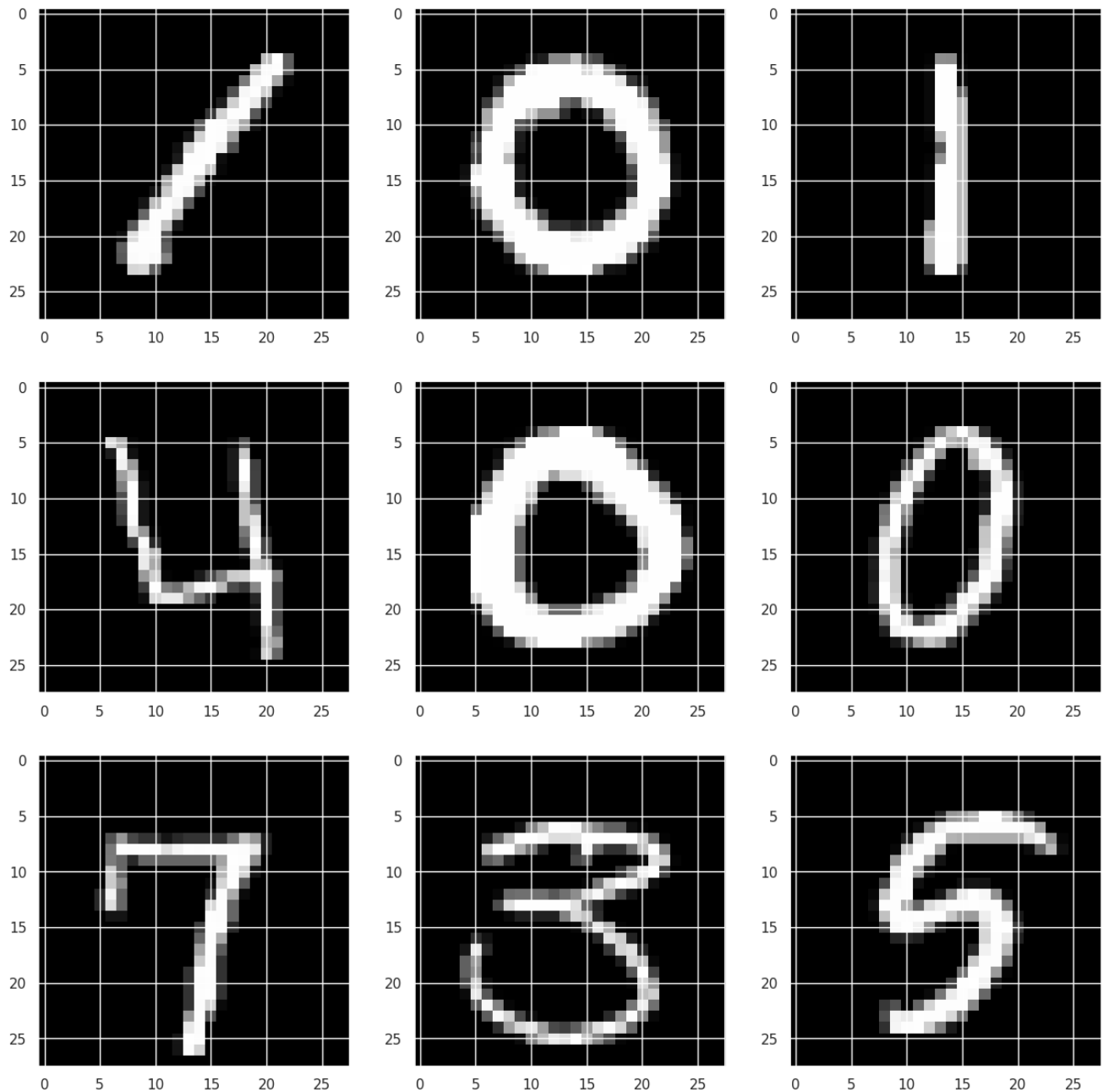
```
for i in range(9):
```

```
    plt.subplot(3, 3, i+1)
```

```
    img = np.reshape(df_train[i], (28, 28))
```

```
    plt.imshow(img, cmap='gray')
```

```
    #plt.xlabel([y_train[i]])
```



```

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=(28, 28, 1)))
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu'))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, kernel_size=(3, 3),activation='relu',input_shape=(28, 28, 1)))
model.add(Conv2D(64, kernel_size=(3, 3),activation='relu'))
model.add(MaxPool2D((2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.25))

model.add(Dense(128, activation = "relu"))
model.add(Dropout(0.5))

model.add(Dense(10, activation = "softmax"))

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

```

Model.summary()

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
dropout (Dropout)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 128)	36992
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 128)	0
dropout_1 (Dropout)	(None, 6, 6, 128)	0
conv2d_2 (Conv2D)	(None, 4, 4, 512)	590336
max_pooling2d_2 (MaxPooling2D)	(None, 2, 2, 512)	0
dropout_2 (Dropout)	(None, 2, 2, 512)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 4096)	8392704
dropout_3 (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 1024)	4195328
dropout_4 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 256)	262400
dropout_5 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 10)	2570
=====		
Total params: 13,480,650		
Trainable params: 13,480,650		
Non-trainable params: 0		

```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint, CSVLogger, LearningRateScheduler, TensorBoard
earlystop=EarlyStopping(patience=10)
filepath = "number.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1,save_best_only=True, mode='max')
log_fname = 'number.csv'
csv_logger = CSVLogger(filename=log_fname,separator=',',append=False)
callback_learningrate = ReduceLROnPlateau(monitor='loss', mode='min', min_delta=0.01, patience=3, factor=.75, min_lr=0.00001, verbose=1)
callbacks_list = [checkpoint, csv_logger,earlystop,callback_learningrate]

history = model.fit(X_train, y_train, epochs=15,batch_size=128, validation_data=(X_val, y_val), callbacks=callbacks_list)
203/203 [=====] - 2s /ms/step - loss: 0.1134 - accuracy: 0.9603 - val_loss: 0.0644 - val_accuracy: 0.9776
Epoch 3/15
261/263 [=====>.] - ETA: 0s - loss: 0.0780 - accuracy: 0.9776
Epoch 3: val_accuracy improved from 0.97976 to 0.98357, saving model to number.h5
263/263 [=====] - 2s 7ms/step - loss: 0.0787 - accuracy: 0.9774 - val_loss: 0.0543 - val_accuracy: 0.98357
Epoch 4/15
260/263 [=====>.] - ETA: 0s - loss: 0.0637 - accuracy: 0.9826
Epoch 4: val_accuracy improved from 0.98357 to 0.98833, saving model to number.h5
263/263 [=====] - 2s 7ms/step - loss: 0.0634 - accuracy: 0.9826 - val_loss: 0.0394 - val_accuracy: 0.98833
Epoch 5/15
262/263 [=====>.] - ETA: 0s - loss: 0.0545 - accuracy: 0.9848
Epoch 5: val_accuracy improved from 0.98833 to 0.98857, saving model to number.h5
263/263 [=====] - 2s 7ms/step - loss: 0.0544 - accuracy: 0.9848 - val_loss: 0.0354 - val_accuracy: 0.98857
Epoch 6/15
261/263 [=====>.] - ETA: 0s - loss: 0.0473 - accuracy: 0.9867
Epoch 6: val_accuracy improved from 0.98857 to 0.98893, saving model to number.h5
263/263 [=====] - 2s 7ms/step - loss: 0.0476 - accuracy: 0.9867 - val_loss: 0.0337 - val_accuracy: 0.98893
Epoch 7/15
260/263 [=====>.] - ETA: 0s - loss: 0.0425 - accuracy: 0.9879
```

```

263/263 [=====] - 2s 8ms/step - loss: 0.0350 - accuracy: 0.9905 - val_loss: 0.0244 - val_accuracy: 0.99
Epoch 10/15
259/263 [=====.] - ETA: 0s - loss: 0.0307 - accuracy: 0.9910
Epoch 10: val_accuracy improved from 0.99214 to 0.99274, saving model to number.h5
263/263 [=====] - 2s 7ms/step - loss: 0.0309 - accuracy: 0.9910 - val_loss: 0.0263 - val_accuracy: 0.99
Epoch 11/15
261/263 [=====.] - ETA: 0s - loss: 0.0308 - accuracy: 0.9915
Epoch 11: val_accuracy did not improve from 0.99274

Epoch 11: ReduceLROnPlateau reducing learning rate to 0.0007500000356230885.
263/263 [=====] - 2s 7ms/step - loss: 0.0307 - accuracy: 0.9915 - val_loss: 0.0336 - val_accuracy: 0.99
Epoch 12/15
262/263 [=====.] - ETA: 0s - loss: 0.0247 - accuracy: 0.9931
Epoch 12: val_accuracy did not improve from 0.99274
263/263 [=====] - 2s 8ms/step - loss: 0.0248 - accuracy: 0.9931 - val_loss: 0.0329 - val_accuracy: 0.99
Epoch 13/15
261/263 [=====.] - ETA: 0s - loss: 0.0218 - accuracy: 0.9939
Epoch 13: val_accuracy improved from 0.99274 to 0.99357, saving model to number.h5
263/263 [=====] - 2s 8ms/step - loss: 0.0217 - accuracy: 0.9939 - val_loss: 0.0234 - val_accuracy: 0.99
Epoch 14/15
258/263 [=====.] - ETA: 0s - loss: 0.0201 - accuracy: 0.9945
Epoch 14: val_accuracy did not improve from 0.99357
263/263 [=====] - 2s 8ms/step - loss: 0.0204 - accuracy: 0.9943 - val_loss: 0.0279 - val_accuracy: 0.99
Epoch 15/15
258/263 [=====.] - ETA: 0s - loss: 0.0195 - accuracy: 0.9944
Epoch 15: val_accuracy improved from 0.99357 to 0.99405, saving model to number.h5

Epoch 15: ReduceLROnPlateau reducing learning rate to 0.0005625000048894435.
263/263 [=====] - 2s 7ms/step - loss: 0.0193 - accuracy: 0.9945 - val_loss: 0.0224 - val_accuracy: 0.99

# Split the data into features and target
X_test = df_test
X_test=X_test/255.0
X_test=X_test.values.reshape(-1,28,28,1)

pred = model.predict(X_test)

875/875 [=====] - 2s 2ms/step

testImgPred = [np.argmax(pred[x]) for x in range(50)]

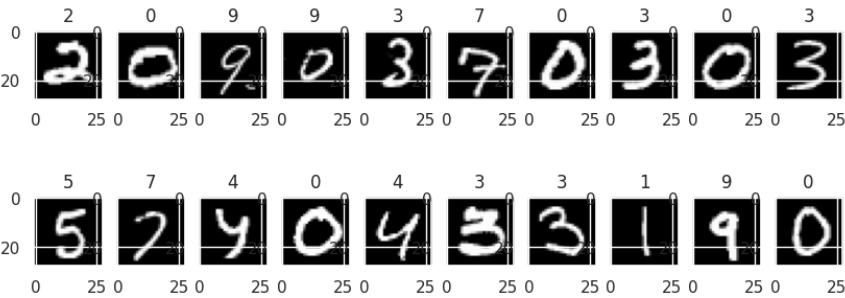
print("Shape of prediction:", pred.shape)

Shape of prediction: (28000, 10)

plt.figure(figsize=(10,10))

for i in range (50):
    plt.subplot(5, 10, i+1)
    img = np.reshape(X_test[i], (28, 28))
    plt.imshow(img, cmap='gray')
    plt.title(testImgPred[i])
plt.show()


```




9 1 1 5 7 4 2 7 4 7

```
#Going with first model
#Creating the CSV to Submit
y_pred = [np.argmax(x) for x in pred]

submissionDict = {'ImageId' : df_test.index + 1}
submission = pd.DataFrame(submissionDict)
submission['Label'] = y_pred

20 
submission.head()

submission.to_csv('digit_submission.csv', index=False)
0 
submission.head()
```

	ImageId	Label
0	1	2
1	2	0
2	3	9
3	4	9
4	5	3