```
# Seyun Kim ECE472 Deep Learning
# Homework 3


import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import io
from sklearn.model_selection import train_test_split
import keras
from keras.models import Model
from keras.layers import *
from keras import optimizers


from google.colab import files
uploaded = files.upload()
```

Choose Files mnist_train.csv

- **mnist_train.csv**(application/vnd.ms-excel) - 109640201 bytes, last modified: 9/21/2020 - 100% done

```
Saving mnist_train.csv to mnist_train (3).csv


test_upload = files.upload()
```

Choose Files mnist_test.csv

- **mnist_test.csv**(application/vnd.ms-excel) - 18303650 bytes, last modified: 9/21/2020 - 100% done

```
Saving mnist_test.csv to mnist_test (1).csv


# Load MNIST train and test data
df_train = pd.read_csv(io.BytesIO(uploaded['mnist_train.csv']))
df_test = pd.read_csv(io.BytesIO(test_upload['mnist_test.csv']))
# Random permutation for MNIST dataset
# df_train = df_train.iloc[np.random.permutation(len(df_train))]
# df_train.head()


df_train.shape
```

```
(60000, 785)


df_test.head()


# Splitting train data to train and validation data
df_label = df_train.iloc[:, 0]
df_pixel = df_train.iloc[:,1:785]

# X: pixels, y: label
X_train, X_val , y_train, y_val = train_test_split(df_pixel, df_label, test_size = 0.2, rando
```

```
X_test = df_train.iloc[:,1:785]
y_test = df_train.iloc[:,0]

# Convert Dataframe to numpy array
X_train = X_train.to_numpy() # (40200, 784)
X_val = X_val.to_numpy() # (19800, 784)
X_test = X_test.to_numpy()


# Pixel normalization
X_train = X_train.astype('float32')
X_val = X_val.astype('float32')
X_test = X_test.astype('float32')

X_train /= 255
X_val /= 255
X_test /= 255


# Make label one hot encoded
digits = 10
y_train = keras.utils.to_categorical(y_train, num_classes = digits)
y_val = keras.utils.to_categorical(y_val, num_classes = digits)
y_test = keras.utils.to_categorical(y_test, num_classes = digits)


# Constructing neural network
layers = [300, 150, 150, 200]
num_layers = len(layers)
dropout_rate = 0.7

input = keras.Input(shape=(784,))
x = Dense(layers[0], activation='relu', kernel_regularizer = 'l2', name = "Hidden_Layer_1")(i
x = Dense(layers[1], activation='relu', kernel_regularizer = 'l2', name = "Hidden_Layer_2")(x
# x = Dropout(dropout_rate)(x)
x = Dense(layers[2], activation='relu', kernel_regularizer = 'l2', name = "Hidden_Layer_3")(x
x = Dropout(dropout_rate)(x)
x = Dense(layers[3], activation='relu', kernel_regularizer = 'l2', name = "Hidden_Layer_4")(x

output = Dense(digits, activation='softmax', name = "Output_Layer")(x)

model = Model(input, output)
model.summary()
```

```
Model: "functional_101"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_59 (InputLayer)        [(None, 784)]             0
_________________________________________________________________
Hidden_Layer_1 (Dense)       (None, 300)               235500
_________________________________________________________________
Hidden_Layer_2 (Dense)       (None, 150)               45150
_________________________________________________________________
Hidden_Layer_3 (Dense)       (None, 150)               22650
_________________________________________________________________
dropout_129 (Dropout)        (None, 150)               0
```

```
# Defining hyperparameters
learning_rate = 0.3
iterations = 30
batch_size = 300
```

```
Trainable params: 335,510
```

```
# Compile model
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])



# Fit model
model.fit(X_train, y_train,
          batch_size=batch_size,
          epochs=iterations,
          verbose=2,
          validation_data= (X_val, y_val)
)


score = model.evaluate(X_test, y_test, verbose=0)
print(f'Test loss: {score[0]} /n Test accuracy: {score[1]}')
```

```
Test loss: 0.39818233251571655 /n Test accuracy: 0.9640499949455261
```

```
prediction = pd.DataFrame(model.predict(X_test, batch_size=200))
prediction = pd.DataFrame(prediction.idxmax(axis = 1))
prediction.index.name = 'ImageId'
prediction = prediction.rename(columns = {0: 'Label'}).reset_index()
prediction['ImageId'] = prediction['ImageId'] + 1

prediction.head()
prediction.head()
```

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