**LAB 1**

from keras.models import Sequential

model = Sequential()

#Stacking layers using .add():

from keras.layers import Dense

model.add(Dense(units=64, activation='relu', input\_dim=100))

model.add(Dense(units=1, activation='softmax'))

#Once your model works, configure its learning process with .compile():

model.compile(loss='binary\_crossentropy', optimizer='sgd', metrics=['accuracy'])

# Generate dummy data

import numpy as np

data = np.random.random((1000, 100))

labels = np.random.randint(2, size=(1000, 1))

# Check the shape of inputs

data.shape

# Check the shape of labels

labels.shape

# Train the model, iterating on the data in batches of 32 samples

model.fit(data, labels, epochs=10, batch\_size=32)

from keras import models

from keras.layers import Dense, Dropout

from keras.utils import to\_categorical

from keras.datasets import mnist

from keras.utils.vis\_utils import model\_to\_dot

from IPython.display import SVG

NUM\_ROWS = 28

NUM\_COLS = 28

NUM\_CLASSES = 10

BATCH\_SIZE = 128

EPOCHS = 10

def data\_summary(X\_train, y\_train, X\_test, y\_test):

    """Summarize current state of dataset"""

    print('Train images shape:', X\_train.shape)

    print('Train labels shape:', y\_train.shape)

    print('Test images shape:', X\_test.shape)

    print('Test labels shape:', y\_test.shape)

    print('Train labels:', y\_train)

    print('Test labels:', y\_test)

# Load data

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

# Check state of dataset

data\_summary(X\_train, y\_train, X\_test, y\_test)

# Reshape data

X\_train = X\_train.reshape((X\_train.shape[0], NUM\_ROWS \* NUM\_COLS))

X\_train = X\_train.astype('float32') / 255

X\_test = X\_test.reshape((X\_test.shape[0], NUM\_ROWS \* NUM\_COLS))

X\_test = X\_test.astype('float32') / 255

# Categorically encode labels

y\_train = to\_categorical(y\_train, NUM\_CLASSES)

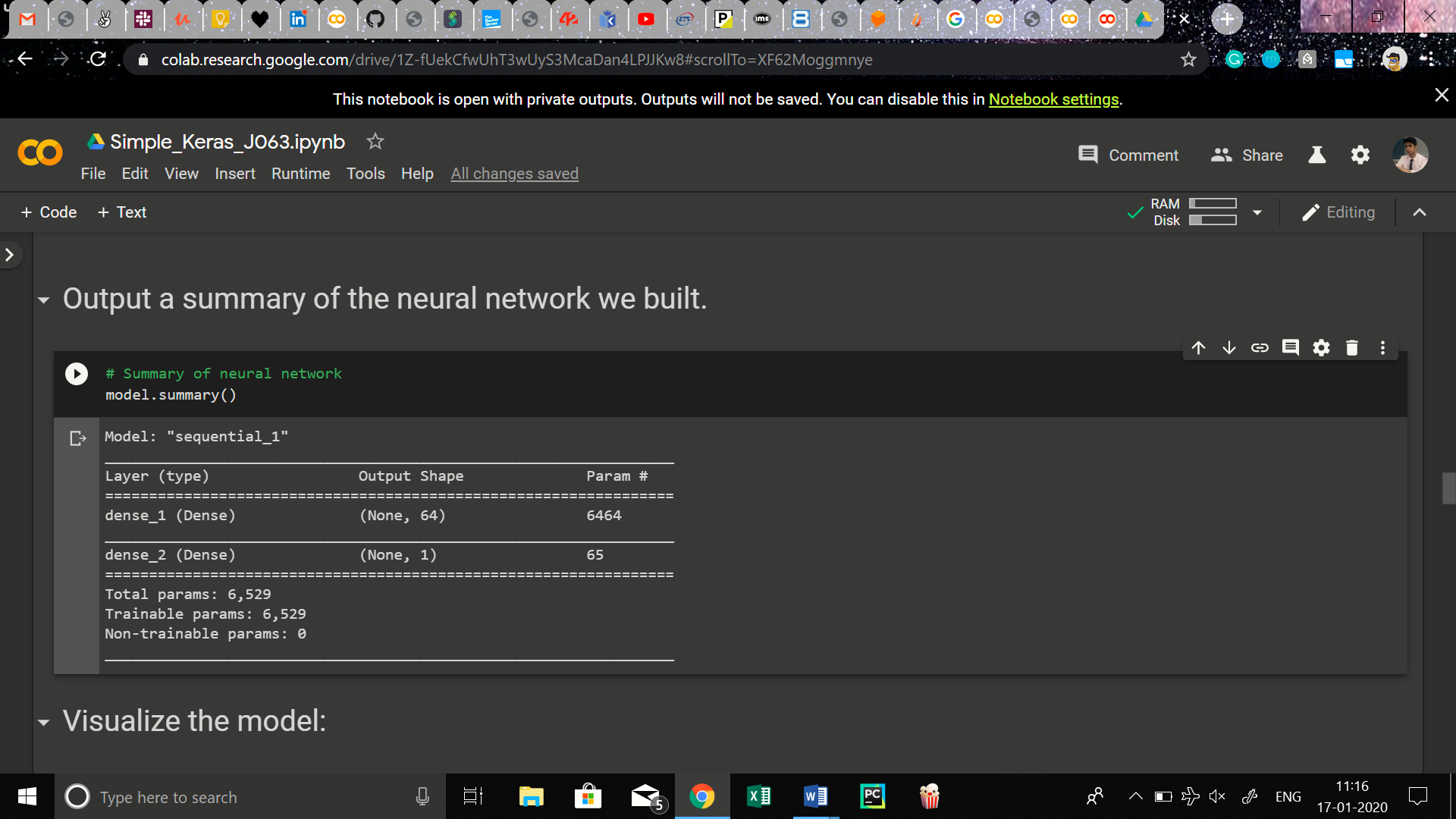
y\_test = to\_categorical(y\_test, NUM\_CLASSES)

# Check state of dataset

data\_summary(X\_train, y\_train, X\_test, y\_test)

# Summary of neural network

model.summary()



# Output network visualization

SVG(model\_to\_dot(model).create(prog='dot', format='svg'))

#import packages

from keras.datasets import cifar10

from keras.models import Sequential

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.utils import to\_categorical

#import dataset

(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()

#change shape from image to vector

X\_train = X\_train.reshape(50000, 32 \* 32 \* 3)

X\_test = X\_test.reshape(10000, 32 \* 32 \* 3)

#preprocess

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

X\_train /= 255.0

X\_test /= 255.0

#change labels from numeric to one hot encoded

Y\_train = to\_categorical(y\_train, 10)

Y\_test =  to\_categorical(y\_test, 10)

model = Sequential()

model.add(Dense(1024, input\_shape=(3072, )))

model.add(Activation('relu'))

model.add(Dense(512))

model.add(Activation('relu'))

model.add(Dense(512))

model.add(Activation('relu'))

model.add(Dense(10))

model.add(Activation('softmax'))

model.compile(loss='categorical\_crossentropy',

                  optimizer='adam',

                  metrics=['accuracy'])

# training

history = model.fit(X\_train, Y\_train,

                        batch\_size=128,

                        nb\_epoch=10,

                        verbose=1,

                        validation\_data=(X\_test, Y\_test))

import matplotlib.pyplot as plt

# Plot training & validation accuracy values

plt.plot(history.history['acc'])

plt.plot(history.history['val\_acc'])

plt.title('Model accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.show()

# Plot training & validation loss values

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Model loss')

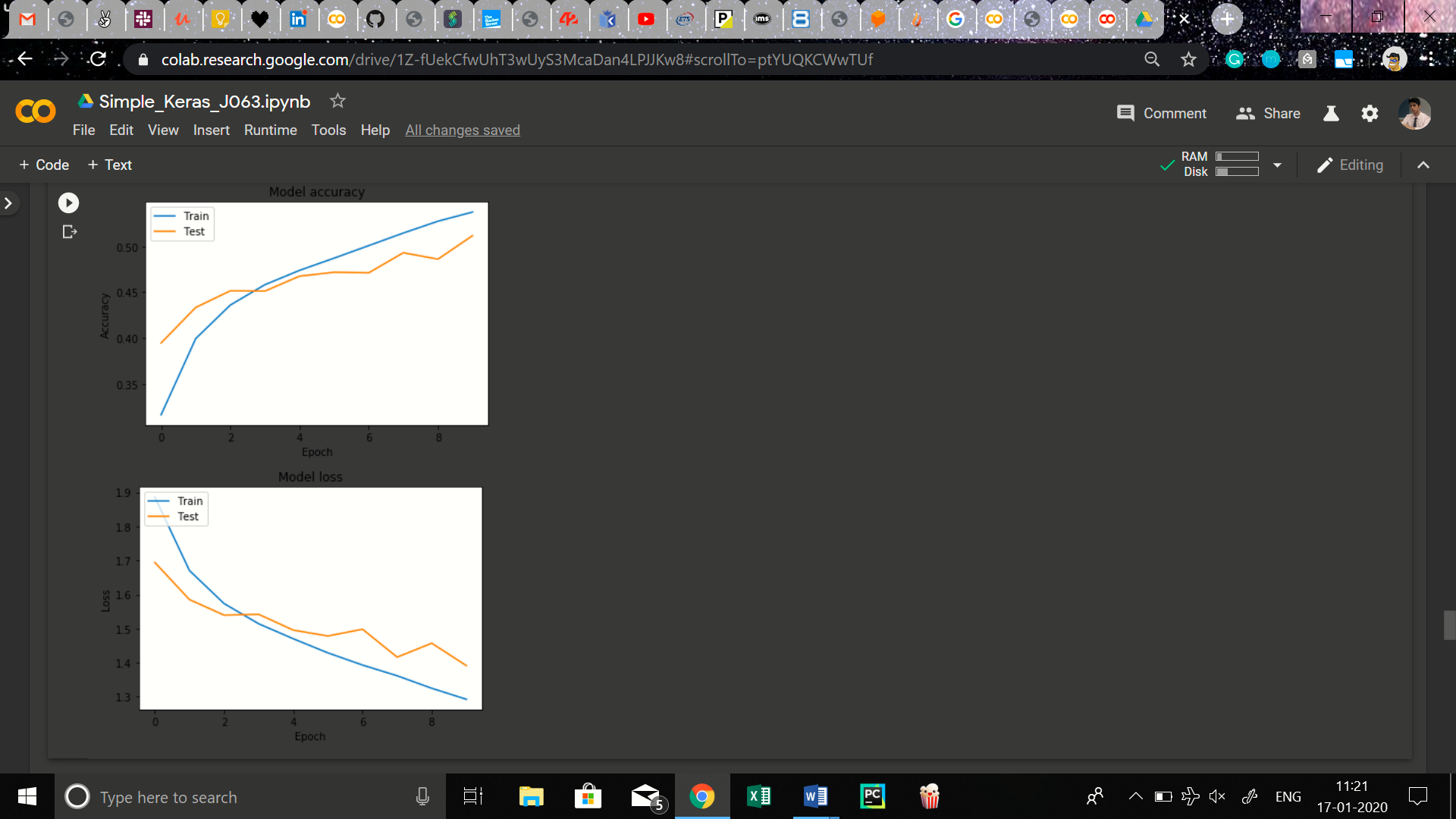
plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Test'], loc='upper left')

plt.show()

->



import pandas

from keras.models import Sequential

from keras.layers import Dense

from keras.wrappers.scikit\_learn import KerasClassifier

from keras.utils import np\_utils

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import KFold

from sklearn.preprocessing import LabelEncoder

from sklearn.pipeline import Pipeline

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import OneHotEncoder

from keras.optimizers import Adam

iris\_data = load\_iris()

x = iris\_data.data

y\_ = iris\_data.target.reshape(-1, 1) # Convert data to a single column

# One Hot encode the class labels

encoder = OneHotEncoder(sparse=False)

y = encoder.fit\_transform(y\_)

# Split the data for training and testing

train\_x, test\_x, train\_y, test\_y = train\_test\_split(x, y, test\_size=0.01)

model = Sequential()

model.add(Dense(10, input\_shape=(4,), activation='relu'))

model.add(Dense(10, activation='relu'))

model.add(Dense(3, activation='softmax'))

# Compile

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Train the model

model.fit(train\_x, train\_y, verbose=2, batch\_size=5, epochs=200)

# Test on unseen data

results = model.evaluate(test\_x, test\_y)

print('Final test set loss: {:4f}'.format(results[0]))

print('Final test set accuracy: {:4f}'.format(results[1]))