Assignment 4: Tensorflow and Data Privacy

By

Rachael Joan Dias

0651897

Addressed to,

Professor Brian Srivastava

Trent University

AMOD-5410H-A-2019GW-PTBO Big Data

1. Machine Learning Problems

The Keras library contains several pretrained models, each model is broken into 2 parts, model architecture and pretrained weights, since the weights are very large the are not bundled with Keras. I used the VGG16 model and loaded ImageNet weights. ImageNet is a large database that contains 14 million images that belong to 20,000 classes [1].

We start by importing all the necessary libraries, from Keras we use the VGG model which we initialize by specifying imagenet weights.

```
import keras
import numpy as np
from keras.applications import vgg16, inception_v3, resnet50, mobilenet

#Load the VGG model
vgg_model = vgg16.VGG16(weights='imagenet')
```

Next, we must load images and preprocess them so that we can apply them to the VGG model. We import all the preprocessing libraries from Keras, and other libraries such as numpy, os and matplotlib that are needed for converting the data to the right format and visualization

The os library helps us to read all the images that are saved in a folder. “BASE_DIR” is the current working directory and within that directory we specify the directory “images” which contains all the images which we will use for our prediction. Using a for loop we iterate through all the .jpg, .png and .jpeg files that are present.

```
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.applications.imagenet_utils import decode_predictions
from keras.applications import vgg16, inception_v3, resnet50, mobilenet
import numpy as np
import os
import matplotlib.pyplot as plt
%matplotlib inline

BASE_DIR = os.path.dirname(os.path.abspath("__file__"))
image_dir = os.path.join(BASE_DIR,"images")

for root, dirs, files in os.walk(image_dir):
    for file in files:
        if file.endswith("jpg") or file.endswith("png") or file.endswith("jpeg"):

            path=os.path.join(root, file)
```

We use load_img() to load the image, then we convert it into a size of 224x2244 and save it as the original image. In the next step we convert the image into a numpy array using the img_to_array() function. Since the network requries a 4-dimensional Tensor as input we convert the numpy array to batch format by adding an extra dimension to the image.

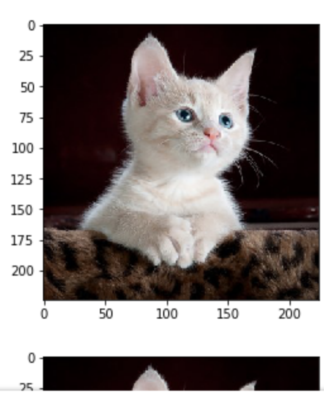
```python
# load an image in PIL format
original = load_img(path, target_size=(224, 224))
print('PIL image size',original.size)
plt.imshow(original)
plt.show()

# convert the PIL image to a numpy array
# IN PIL - image is in (width, height, channel)
# In Numpy - image is in (height, width, channel)
numpy_image = img_to_array(original)
plt.imshow(np.uint8(numpy_image))
plt.show()
print('numpy array size',numpy_image.shape)

# Convert the image / images into batch format
# expand_dims will add an extra dimension to the data at a particular axis
# We want the input matrix to the network to be of the form (batchsize, height, width, channels)
# Thus we add the extra dimension to the axis 0.
image_batch = np.expand_dims(numpy_image, axis=0)
print('image batch size', image_batch.shape)
plt.imshow(np.uint8(image_batch[0]))
```

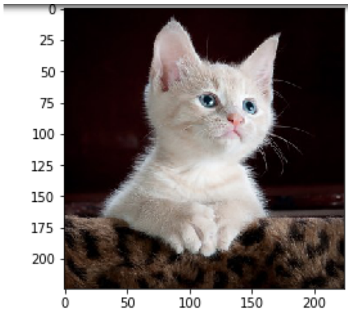
Once this is complete we pass the image to the vgg16 model by passing it to the preprocess_input() function. Predictions can be made on the image, as the predictions returned are an array the decode_predictions() function will give us the labels for each of the predictions

```python
# prepare the image for the VGG model
processed_image = vgg16.preprocess_input(image_batch.copy())

# get the predicted probabilities for each class
predictions = vgg_model.predict(processed_image)
# print predictions

# convert the probabilities to class labels
# We will get top 5 predictions which is the default
label = decode_predictions(predictions)
print (label)
```

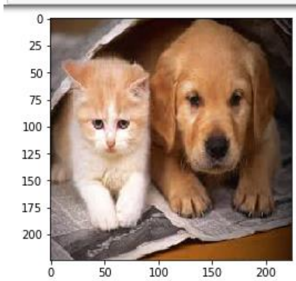
Below we can see the output image. The first image is in PIL format, the second image is the numpy array and finally we have the predictions for the image. The VGG16 model predicts that the cat looks like an Egyptian cat, tabby, lynx, tiger cart or Persian cat. Similarly, for all the other images the model makes a couple of predictions.

```
PIL image size (224, 224)
```

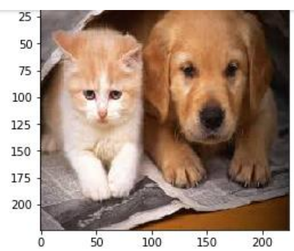




```
numpy array size (224, 224, 3)
image batch size (1, 224, 224, 3)
[[('n02124075', 'Egyptian_cat', 0.4049535), ('n02123045', 'tabby', 0.3942576), ('n02127052', 'lynx', 0.10080725), ('n0212315
9', 'tiger_cat', 0.05541087), ('n02123394', 'Persian_cat', 0.012944666)]]
PIL image size (224, 224)
```

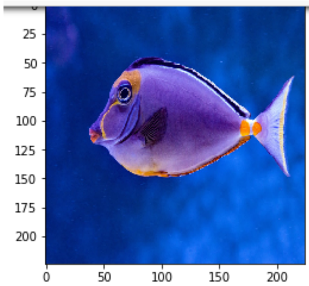


The model can detect the golden retriever from the image



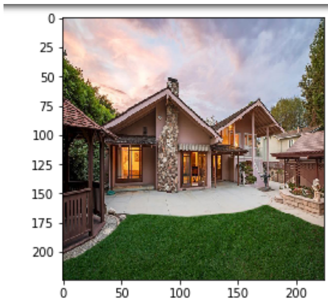
```
numpy array size (224, 224, 3)
image batch size (1, 224, 224, 3)
[[('n02099601', 'golden_retriever', 0.62682486), ('n02099712', 'Labrador_retriever', 0.26682746), ('n02808304', 'bath_towe
l', 0.011000982), ('n02088466', 'bloodhound', 0.006998717), ('n02090379', 'redbone', 0.0055709733)]]
PIL image size (224, 224)
```

The model predicts that the fish looks like a puffer fish, anemone fish, stingray, electric ray and rock beauty fish

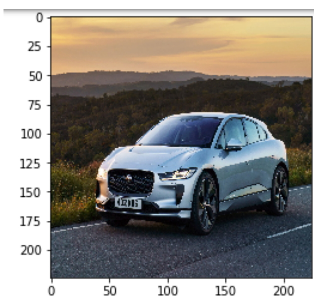



```
numpy array size (224, 224, 3)
image batch size (1, 224, 224, 3)
[[('n02606052', 'rock_beauty', 0.81457144), ('n01496331', 'electric_ray', 0.09466354), ('n01498041', 'stingray', 0.03860913
2), ('n02655020', 'puffer', 0.011262267), ('n02607072', 'anemone_fish', 0.009718927)]]
PIL image size (224, 224)
```

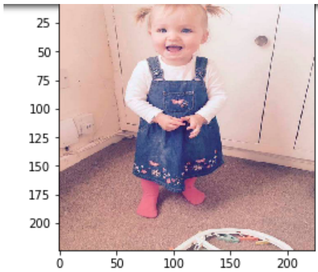
The model predicts that the image below looks like a boathouse




```
numpy array size (224, 224, 3)
image batch size (1, 224, 224, 3)
[[('n02859443', 'boathouse', 0.42237583), ('n03899768', 'patio', 0.42175072), ('n03776460', 'mobile_home', 0.028971152), ('n0
3028079', 'church', 0.022965368), ('n09332890', 'lakeside', 0.020443752)]]
```

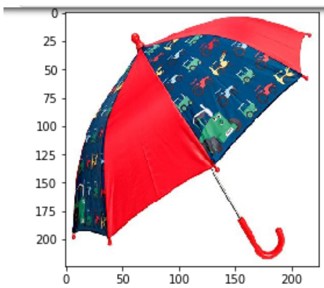



```
numpy array size (224, 224, 3)
image batch size (1, 224, 224, 3)
[[('n03930630', 'pickup', 0.25085935), ('n04037443', 'racer', 0.16108784), ('n04285008', 'sports_car', 0.1478414), ('n0310024
0', 'convertible', 0.12275199), ('n02974003', 'car_wheel', 0.10857254)]]
```




```
numpy array size (224, 224, 3)
image batch size (1, 224, 224, 3)
[[('n03877472', 'pajama', 0.061883893), ('n15075141', 'toilet_tissue', 0.053983476), ('n03935335', 'piggy_bank', 0.04350390
3), ('n04447861', 'toilet_seat', 0.04228444), ('n02971356', 'carton', 0.030240832)]]
PIL image size (224, 224)
```

The first prediction for the image below is umbrella




```
numpy array size (224, 224, 3)
image batch size (1, 224, 224, 3)
[[('n04507155', 'umbrella', 0.99815375), ('n03888257', 'parachute', 0.0009230035), ('n03792972', 'mountain_tent', 0.000412372
84), ('n03355925', 'flagpole', 0.00026127658), ('n04371430', 'swimming_trunks', 6.1023005e-05)]]
```

Script Name: assgn#4_1.ipynb

2. **TensorFlow Sample Problem**

a. Applying the Tensorflow model that was trained on the IMDB dataset I was able to classify positive and negative reviews of my dataset. I used the airline tweets dataset which consists of reviews of American airlines. The predictions gave values between 0 and 1, 0 represents negative sentiments and 1 indicates positive sentiments

The screenshots below show how the model was trained on the IMDB dataset

```
from __future__ import absolute_import, division, print_function

import tensorflow as tf
from tensorflow import keras

import numpy as np

print(tf.__version__)
```

```
1.13.1
```

```
imdb = keras.datasets.imdb

(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=10000)
```

```
print("Training entries: {}, labels: {}".format(len(train_data), len(train_labels)))
```

```
Training entries: 25000, labels: 25000
```

```
print(train_data[0])
```

```
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 3
5, 480, 284, 5, 150, 4, 172, 112, 167, 2, 336, 385, 39, 4, 172, 4536, 1111, 17, 546, 38, 13, 447, 4, 192, 50, 16, 6, 147, 2025,
19, 14, 22, 4, 1920, 4613, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15, 13, 1247, 4, 22, 17, 515, 17, 12, 16, 626, 18, 2,
5, 62, 386, 12, 8, 316, 8, 106, 5, 4, 2223, 5244, 16, 480, 66, 3785, 33, 4, 130, 12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 48,
25, 1415, 33, 6, 22, 12, 215, 28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 117, 5952, 15, 256, 4, 2, 7, 3766, 5, 723, 36, 71,
43, 530, 476, 26, 400, 317, 46, 7, 4, 2, 1029, 13, 104, 88, 4, 381, 15, 297, 98, 32, 2071, 56, 26, 141, 6, 194, 7486, 18, 4, 22
6, 22, 21, 134, 476, 26, 480, 5, 144, 30, 5535, 18, 51, 36, 28, 224, 92, 25, 104, 4, 226, 65, 16, 38, 1334, 88, 12, 16, 283, 5,
16, 4472, 113, 103, 32, 15, 16, 5345, 19, 178, 32]
```

```
len(train_data[0]), len(train_data[1])
```

```
(218, 189)
```

```
# A dictionary mapping words to an integer index
word_index = imdb.get_word_index()

# The first indices are reserved
word_index = {k:(v+3) for k,v in word_index.items()}
word_index["<PAD>"] = 0
word_index["<START>"] = 1
word_index["<UNK>"] = 2  # unknown
word_index["<UNUSED>"] = 3

reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])

def decode_review(text):
    return ' '.join([reverse_word_index.get(i, '?') for i in text])
```

```
decode_review(train_data[0])
```

```
"<START> this film was just brilliant casting location scenery story direction everyone's really suited the part they played an
d you could just imagine being there robert <UNK> is an amazing actor and now the same being director <UNK> father came from th
e same scottish island as myself so i loved the fact there was a real connection with this film the witty remarks throughout th
e film were great it was just brilliant so much that i bought the film as soon as it was released for <UNK> and would recommend
it to everyone to watch and the fly fishing was amazing really cried at the end it was so sad and you know what they say if you
cry at a film it must have been good and this definitely was also <UNK> to the two little boy's that played the <UNK> of norman
and paul they were just brilliant children are often left out of the <UNK> list i think because the stars that play them all gr
own up are such a big profile for the whole film but these children are amazing and should be praised for what they have done d
on't you think the whole story was so lovely because it was true and was someone's life after all that was shared with us all"
```

```python
train_data = keras.preprocessing.sequence.pad_sequences(train_data,
                                                        value=word_index["<PAD>"],
                                                        padding='post',
                                                        maxlen=256)

test_data = keras.preprocessing.sequence.pad_sequences(test_data,
                                                       value=word_index["<PAD>"],
                                                       padding='post',
                                                       maxlen=256)
```

```python
len(train_data[0]), len(train_data[1])
```

```
(256, 256)
```

```python
print(train_data[0])
```

```
[   1   14   22   16   43  530  973 1622 1385   65  458 4468   66 3941
    4  173   36  256    5   25  100   43  838  112   50  670    2    9
   35  480  284    5  150    4  172  112  167    2  336  385   39    4
  172 4536 1111   17  546   38   13  447    4  192   50   16    6  147
 2025   19   14   22    4 1920 4613  469    4   22   71   87   12   16
   43  530   38   76   15   13 1247    4   22   17  515   17   12   16
  626   18    2    5   62  386   12    8  316    8  106    5    4 2223
 5244   16  480   66 3785   33    4  130   12   16   38  619    5   25
  124   51   36  135   48   25 1415   33    6   22   12  215   28   77
   52    5   14  407   16   82    2    8    4  107  117 5952   15  256
    4    2    7 3766    5  723   36   71   43  530  476   26  400  317
   46    7    4    2 1029   13  104   88    4  381   15  297   98   32
 2071   56   26  141    6  194 7486   18    4  226   22   21  134  476
   26  480    5  144   30 5535   18   51   36   28  224   92   25  104
    4  226   65   16   38 1334   88   12   16  283    5   16 4472  113
  103   32   15   16 5345   19  178   32    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0]
```

```python
# input shape is the vocabulary count used for the movie reviews (10,000 words)
vocab_size = 10000

model = keras.Sequential()
model.add(keras.layers.Embedding(vocab_size, 16))
model.add(keras.layers.GlobalAveragePooling1D())
model.add(keras.layers.Dense(16, activation=tf.nn.relu))
model.add(keras.layers.Dense(1, activation=tf.nn.sigmoid))

model.summary()
```

```
WARNING:tensorflow:From C:\Users\17059\Anaconda3\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py:435: colocate
_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding (Embedding)        (None, None, 16)          160000    
_________________________________________________________________
global_average_pooling1d (Gl (None, 16)                0         
_________________________________________________________________
dense (Dense)                (None, 16)                272       
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 17        
=================================================================
Total params: 160,289
Trainable params: 160,289
Non-trainable params: 0
_________________________________________________________________
```

```python
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['acc'])
```

```python
x_val = train_data[:10000]
partial_x_train = train_data[10000:]

y_val = train_labels[:10000]
partial_y_train = train_labels[10000:]
```

```python
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=40,
                    batch_size=512,
                    validation_data=(x_val, y_val),
                    verbose=1)
```

```
Epoch 36/40
15000/15000 [==============================] - 1s 88us/sample - loss: 0.1074 - acc: 0.9707 - val_loss: 0.2998 - val_acc: 0.8849
Epoch 37/40
15000/15000 [==============================] - 1s 93us/sample - loss: 0.1031 - acc: 0.9719 - val_loss: 0.3023 - val_acc: 0.8833
Epoch 38/40
15000/15000 [==============================] - 1s 96us/sample - loss: 0.0992 - acc: 0.9735 - val_loss: 0.3060 - val_acc: 0.8826
Epoch 39/40
15000/15000 [==============================] - 1s 86us/sample - loss: 0.0960 - acc: 0.9741 - val_loss: 0.3096 - val_acc: 0.8821
Epoch 40/40
15000/15000 [==============================] - 2s 122us/sample - loss: 0.0920 - acc: 0.9763 - val_loss: 0.3119 - val_acc: 0.882
4
```

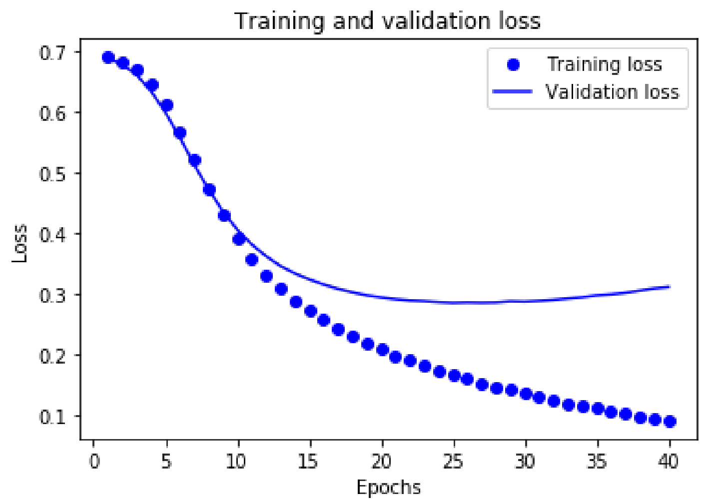
```python
results = model.evaluate(test_data, test_labels)

print(results)
```

```
25000/25000 [==============================] - 1s 42us/sample - loss: 0.3329 - acc: 0.8716
[0.33292325684547425, 0.8716]
```

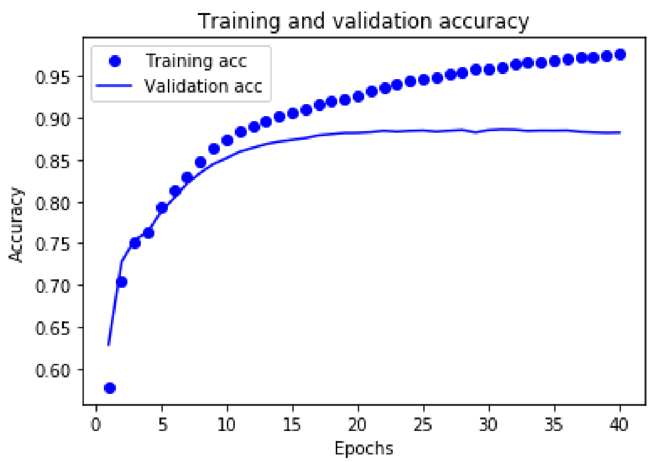
```python
history_dict = history.history
history_dict.keys()
```

```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

```python
import matplotlib.pyplot as plt

acc = history_dict['acc']
val_acc = history_dict['val_acc']
loss = history_dict['loss']
val_loss = history_dict['val_loss']

epochs = range(1, len(acc) + 1)

# "bo" is for "blue dot"
plt.plot(epochs, loss, 'bo', label='Training loss')
# b is for "solid blue line"
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```




```
plt.clf()   # clear figure

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



Below is a screenshot of the data the first column is airline sentiment which indicates if the reviews are positive or negative the text column contains passenger reviews.

airline_se	airline	name	text
1	Virgin Am	jnardino	@VirginAmerica plus you've added commercials to the experience... tacky.
0	Virgin Am	jnardino	@VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces & they have little recourse
0	Virgin Am	jnardino	@VirginAmerica and it's a really big bad thing about it
0	Virgin Am	jnardino	@VirginA
1	Virgin Am	cjmcginni:	@VirginAmerica yes, nearly every time I fly VX this â€œear wormâ€ wonâ€™t go away :)
1	Virgin Am	dhepburn	@virginamerica Well, I didn'tâ€¦but NOW I DO! :-D
1	Virgin Am	YupitsTate	@VirginAmerica it was amazing, and arrived an hour early. You're too good to me.
1	Virgin Am	HyperCam	@VirginAmerica I <3 pretty graphics. so much better than minimal iconography. :D
1	Virgin Am	HyperCam	@VirginAmerica This is such a great deal! Already thinking about my 2nd trip to @Australia & I haven't even gone on my
1	Virgin Am	mollander	@VirginAmerica @virginmedia I'm flying your #fabulous #Seductive skies again! U take all the #stress away from travel http:/
1	Virgin Am	sjespers	@VirginAmerica Thanks!
0	Virgin Am	smartwate	@VirginAmerica SFO-PDX schedule is still MIA.

First, we import the pandas module to read the .csv file and specify the encoding as utf-8 the file is saved as a data frame "df" from which we select only the text column and save as another dataframe test_samples.

From Tensorflow we import the preprocessing modules required Tokenizer and pad_sequences. The text column data has to be converted into tokens using the tokenzier, and then into a series of sequences using the pad_sequence function. This can be fed to the sentiment analysis model to make predictions. As, you can see we get an array of values between 0 and 1, those which are closer to 0 are classified as negative sentiments while those which are closer to 1 are positive sentiments [2].

```
import pandas as pd

df=pd.read_csv('Tweets.csv',encoding='utf-8')
test_samples=df['text']
```

```
from tensorflow.python.keras.preprocessing.text import Tokenizer
from tensorflow.python.keras.preprocessing.sequence import pad_sequences

vocabulary_size = 10000
tokenizer = Tokenizer(num_words= vocabulary_size, filters='')
tokenizer.fit_on_texts(test_samples)
sequences = tokenizer.texts_to_sequences(test_samples)
data = pad_sequences(sequences, maxlen= 256)
pred=model.predict(x=data)
print(pred)
```

```
[[0.52036375]
 [0.43659645]
 [0.38662496]
 ...
 [0.6484683 ]
 [0.49651265]
 [0.5504872 ]]
```

The first sentinment is positive while the second and third sentiments are negative. The model predicts .52, .436 and .386 for first, second and third respectively so we can assume that the first review is a positive review and the second and third reviews are negative.

```
print(df)
```

```
   airline_sentiment         airline         name  \
0                  1  Virgin America     jnardino
1                  0  Virgin America     jnardino
2                  0  Virgin America     jnardino
3                  0  Virgin America     jnardino
4                  1  Virgin America   cjmcginnis
5                  1  Virgin America     dhepburn
```

Script Name: Assgn#4_2a

b. MNSIT model

```
from __future__ import print_function

import math
import os
import glob
from IPython import display
from matplotlib import cm
from matplotlib import gridspec
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import metrics
import tensorflow as tf
from tensorflow.python.data import Dataset

tf.logging.set_verbosity(tf.logging.ERROR)
pd.options.display.max_rows = 10
pd.options.display.float_format = '{:.1f}'.format

mnist_dataframe = pd.read_csv(
  "https://download.mlcc.google.com/mledu-datasets/mnist_train_small.csv",
  sep=",",
  header=None)

# Use just the first 10,000 records for training/validation.
mnist_dataframe = mnist_dataframe.head(10000)

mnist_dataframe = mnist_dataframe.reindex(np.random.permutation(mnist_dataframe.index))
```

```
mnist_dataframe.head()
```

	0	1	2	3	4	5	6	7	8	9	...	775	776	777	778	779	780	781	782	783	784
5701	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7271	8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
584	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8400	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5174	6	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

```
mnist_dataframe.loc[:, 72:72]
```

	72
5701	0
7271	0
584	0
8400	0
5174	0
...	...
9949	0
6507	0
375	0

```
def parse_labels_and_features(dataset):
  """Extracts labels and features.

  This is a good place to scale or transform the features if needed.

  Args:
    dataset: A Pandas `Dataframe`, containing the label on the first column and
      monochrome pixel values on the remaining columns, in row major order.
  Returns:
    A `tuple` `(labels, features)`:
      labels: A Pandas `Series`.
      features: A Pandas `DataFrame`.
  """
  labels = dataset[0]

  # DataFrame.loc index ranges are inclusive at both ends.
  features = dataset.loc[:,1:784]
  # Scale the data to [0, 1] by dividing out the max value, 255.
  features = features / 255

  return labels, features
```

```
training_targets, training_examples = parse_labels_and_features(mnist_dataframe[:7500])
training_examples.describe()
```

	1	2	3	4	5	6	7	8	9	10	...	775	776	777	778	779	780	781	782	783
count	7500.0	7500.0	7500.0	7500.0	7500.0	7500.0	7500.0	7500.0	7500.0	7500.0	...	7500.0	7500.0	7500.0	7500.0	7500.0	7500.0	7500.0	7500.0	7500.0
mean	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
std	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	1.0	1.0	0.8	0.0	0.0	0.0	0.0	0.0	0.0

8 rows × 784 columns

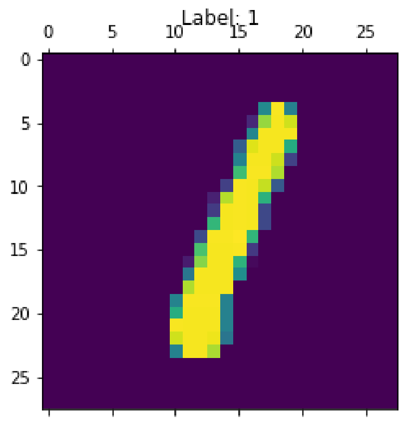
```
validation_targets, validation_examples = parse_labels_and_features(mnist_dataframe[7500:10000])
validation_examples.describe()
```

	1	2	3	4	5	6	7	8	9	10	...	775	776	777	778	779	780	781	782	783
count	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0	...	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0	2500.0
mean	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
std	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75%	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.5	0.1	0.0	0.2	1.0	0.2	0.0	0.0	0.0

8 rows × 784 columns

:

```
rand_example = np.random.choice(training_examples.index)
_, ax = plt.subplots()
ax.matshow(training_examples.loc[rand_example].values.reshape(28, 28))
ax.set_title("Label: %i" % training_targets.loc[rand_example])
ax.grid(False)
```

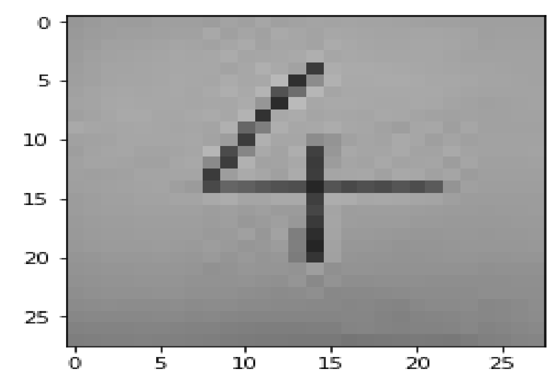



```
def construct_feature_columns():
  """Construct the TensorFlow Feature Columns.

  Returns:
    A set of feature columns
  """

  # There are 784 pixels in each image.
  return set([tf.feature_column.numeric_column('pixels', shape=784)])
```

```
def train_nn_classification_model(
    learning_rate,
    steps,
    batch_size,
    hidden_units,
    training_examples,
    training_targets,
    validation_examples,
    validation_targets):
  """Trains a neural network classification model for the MNIST digits dataset.

  In addition to training, this function also prints training progress information,
  a plot of the training and validation loss over time, as well as a confusion
  matrix.

  Args:
    learning_rate: A `float`, the learning rate to use.
    steps: A non-zero `int`, the total number of training steps. A training step
      consists of a forward and backward pass using a single batch.
    batch_size: A non-zero `int`, the batch size.
    hidden_units: A `list` of int values, specifying the number of neurons in each layer.
    training_examples: A `DataFrame` containing the training features.
    training_targets: A `DataFrame` containing the training labels.
    validation_examples: A `DataFrame` containing the validation features.
    validation_targets: A `DataFrame` containing the validation labels.

  Returns:
    The trained `DNNClassifier` object.
  """

  periods = 10
```

In order to apply the classifier to my handwritten image we need to convert the image to grayscale and resize it to 28x28. The code below converts the image to grayscale.

```
import numpy as np
import keras
from matplotlib import pyplot as plt
from PIL import Image
from keras.preprocessing.image import img_to_array
%matplotlib inline
import pandas as pd

#converting image to Gryscale
filename="myimage.jpg"
x=Image.open(filename)
x=x.convert("L")
plt.imshow(x)
x.save('x.jpg')
```



Next, the image has to be in the correct size and format after converting it to an array of 28x28 we then convert it to a dataframe and pass the image to the parse_labels_and_features() function. Now, we can use the classifier to make predictions on the image, the classifier does not give very good results on the image

```
#converting image to 28X28
x = np.resize(x, (28,28,1))
im2arr = np.array(x)
im2arr = im2arr.reshape(1,28,28,1)
mnsit = pd.DataFrame(im2arr.reshape(len(im2arr),-1))
df1 = pd.DataFrame(np.array([3]))
mytest = pd.concat([df1, mnsit], ignore_index=True,axis=1)

#parsing labels and features
test_targets, test_examples = parse_labels_and_features(mytest)
test_examples.describe()
predict_test_input_fn = create_predict_input_fn(
    test_examples, test_targets, batch_size=10)

#making predictions
test_predictions = classifier.predict(input_fn=predict_test_input_fn)
test_predictions = np.array([item['class_ids'][0] for item in test_predictions])
print(test_predictions)
accuracy = metrics.accuracy_score(test_targets, test_predictions)
print("Accuracy on test data: %0.2f" % accuracy)
```

```
[0]
Accuracy on test data: 0.00
```

Script Name: assgn#4_2b.ipynb

3. Data Privacy

In November of 2018, Marriott International hotels reported that cyber thieves had stolen data from 500 million customers. The breach began on systems supporting Starwood hotel brands in the year 2014. The cyber thieves continued to remain in the system after the Marriott group acquired Starwood in 2016 and remained undiscovered until September 2018 [3].

It is believed that for some of the victims only their name and contact information was compromised. However, the attackers also extracted passport information, Starwood Preferred Guest numbers, travel information and other personal information. There are claims that credit card information and expiration dates of as many as 100 million customers were stolen, although Marriott is not certain if the hackers were able to decrypt the credit card numbers [3].

The breach was attributed to a Chinese intelligence group which was trying to gather information on US citizens. A few individuals who were brief after an investigation also reported that the intelligence group hacked health insurers and security clearance files of millions of more Americans [3].

It is suspected that the hackers were working on behalf of the Ministry of State Security, the country's Communist-controlled civilian spy agency. This discovery was made while the Trump was planning actions to target China's trade, cyber and economics policies. The actions included criminal accusations against Chinese hackers working for intelligence services and the military, the Trump administration also plans on revealing intelligence reports that reveal Chinese efforts to set up a database containing the names of executives and American governments officials with security clearance [3].

Some of the other options included making it harder for Chinese companies to obtain critical components from telecommunications equipment, as stated by a senior American official [3].

Despite the 90-day truce negotiated by president Trump and President Xi Jinping in Buenos Aires, the administration believes this might do very little to change China's behavior. Since China has coerced American companies to hand over valuable technology if they wish to enter the Chinese market, this also includes theft of industrial secrets on behalf of state-owned companies [3].

The Yahoo data breach was one of the largest data breaches in history, it impacted over 3 billion user accounts. While Yahoo was making negotiations to sell itself to Verizon it revealed that it had been a victim of the biggest data breach, that had been likely committed by a state-sponsored hacker in 2014, the announcement was made in September 2016, 2 years after the breach happened. The data breach compromised the real names, email addresses, date of birth and phone number of 500 million users. Yahoo reported that most of the passwords involved had been hashed by the bcrypt algorithm [3].

A few months later in the month of December 2018 it covered up the earlier record with the disclosure that a breach in 2013, by another group of cyber thieves had compromised 1 billion accounts. Aside from the names, dates of birth, email addresses and passwords that were protected not as well as the ones involved in the 2014 breach, it is estimated the even security questions and answers were compromised. By October of 2017, Yahoo reviewed that estimate, stating that all 3 billion user accounts had been compromised [3].

The data breaches severely affected Yahoo's sales prices knocking of an estimated $350 million of the company's sales price. Yahoo, founded in 1994, had been valued at $100 billion, ultimately Verizon paid $4.48 billion for Yahoo's core internet business. The agreement meant that the two companies had to share regulatory and legal liabilities from the breaches. Yahoo failed to thoroughly investigate the data breaches and carried on carelessly which ultimately led to the downfall of the company [3].

In late 2016 personal information of 57 million Uber accounts and 600,00 drivers was exposed, Uber did not handle the breach well. By late 2016 the company learned that 2 hackers had extracted names, email addresses, and mobile numbers of 57 million users on the Uber app. They also hacked the drivers license numbers of 600,00 Uber drivers. Apart from this not other information such as credit card and Social Security numbers were stolen. The hackers accessed the information through Uber's GitHub account, where thy were able to find username and password credentials to Uber's AWS account. The username and password credentials should have never been on the GitHub account in the first place [3].

It wasn't until about a year after the incident happened that Uber made the breach public. The worst part was that they paid the hackers $100,000 to destroy the data with no way of verifying that it was destroyed. Eventually Uber fired it's CSO because of the breach, putting all the blame on him [3].

The breach cost Uber a lot of money and ruined it reputation. At the time when the breach was announced the company was negotiating to sell a stake of the company to Softbank. Uber's was initially valued at $68 billion by the time the deal was sealed in December; its value dropped to $48 billion [3].

Data is growing at an exponential rate; it is estimated that nearly 1.7 megabytes of data are created every second. This makes it extremely hard for organizations to keep up and protect their customer's personal information. Poor security practices such as the Uber data breach put organization at risk of a data breach [4].

A data breach can cost an organization millions of dollars and ruin its reputation. In the year 2017, the average cost of a breach was $3.62 million. If a data breach occurs the organization faces intense regulatory penalties from an array of institutions. Companies dealing with customer information in the European Union that face a huge breach because of lack of security control can face a penalty of up to 4% Adjusted Gross Revenue or 20 million euros. In order to avoid this companies must make investments in key security technologies such as data archiving, backup and redundant infrastructure to protect their data [4].

Human error is the biggest challenge that is faced in data privacy and security. Employees that are unaware may use weak passwords, mistakenly delete data, browse websites that are not acceptable. Data loss prevention tools can help prevent leaking sensitive data [4].

References

[1] https://www.learnopencv.com/keras-tutorial-using-pre-trained-imagenet-models/

[2] https://stackoverflow.com/questions/51699001/tokenizer-texts-to-sequences-keras-tokenizer-gives-almost-all-zeros

[3] https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html

[4] http://blog.cipher.com/the-5-biggest-challenges-in-global-data-privacy-and-data-protection