HAPPY MONK

ASSIGNMENT-1

TASK-1

Consider a large dataset (say, a time series) A. Also, consider a smaller dataset B. How do you ensure whether sets A and B identify the same variable? Illustrate it with a Python script.

Description:

Considered two dataset A which is a time series dataset containing the closing price of APPLE and B which is a textual data that contains the news headlines of APPLE.
The objective is to recommend the right decision about the investment can be done in particular stock or not.
Data A

date	
2017-12-26 00:00:00+00:00	170.57
2017-12-27 00:00:00+00:00	170.60
2017-12-28 00:00:00+00:00	171.08
2017-12-29 00:00:00+00:00	169.23
2018-01-02 00:00:00+00:00	172.26
...	...
2022-12-16 00:00:00+00:00	134.51
2022-12-19 00:00:00+00:00	132.37
2022-12-20 00:00:00+00:00	132.30
2022-12-21 00:00:00+00:00	135.45
2022-12-22 00:00:00+00:00	132.23

Data B

	ticker	date	time	title
0	AAPL	Dec-23-22	02:15AM	Most Tech Stocks Had a Miserable Year. Not IBM.
1	AAPL	Dec-23-22	12:06AM	Apple's Australian workers go on Christmas str...
2	AAPL	Dec-23-22	12:00AM	Meta and Alphabet lose dominance over US digit...
3	AAPL	Dec-22-22	07:46PM	Peak TV is coming to an end as streaming servi...
4	AAPL	Dec-22-22	06:00PM	UPDATE 2-Apple Watches violate AliveCor patent...
...	...	...	...	...
95	AAPL	Dec-17-22	05:00AM	Big Tech is Designing Chips Too, Should Semico...
96	AAPL	Dec-16-22	06:13PM	Weekly Round Up
97	AAPL	Dec-16-22	04:54PM	How Apples privacy changes may impact Meta
98	AAPL	Dec-16-22	04:38PM	The No. 1 set and forget stock in tech hardwar...
99	AAPL	Dec-16-22	02:08PM	Apple Stock Breaks Below Key Support.

The stock price prediction for next 7 days.

```
array([[131.75533958],
       [132.0178921 ],
       [132.27101648],
       [132.51873593],
       [132.76466168],
       [133.01172968],
       [133.26222151]])
```

The changes is compared with the current closing price with the change in price in next 7 days. If it is positive, it recommends that it is safe to invest or else there may be risk involved.

ticker	AAPL	sentiment
date		
2022-12-16	-0.018700	-1
2022-12-17	-0.124417	-1
2022-12-18	-0.061650	-1
2022-12-19	0.084360	0
2022-12-20	0.062486	0
2022-12-21	0.107638	1
2022-12-22	-0.014286	-1
2022-12-23	-0.197400	-1

The sentiment of the past 7 days is calculated. If the number of positive sentiments is greater than the number of negative sentiments then the model suggests investing in the stock or else there may be a risk involved in investing.

```
    if increase>decrease:
      price_suggestion='invest'
    else:
      price_suggestion='may be risk'
  if results.sentiment.value_counts()[1]>results.sentiment.value_counts()[-1]:
    news_suggestion='invest'
  else:
    news_suggestion='may be risk'
  if price_suggestion==news_suggestion:
    suggestion=news_suggestion
  else:
    suggestion='may be risk'
  print('Stock investion suggestion for APPLE is',suggestion)
inference(present_value,forecast,results)
```

```
Stock investion suggestion for APPLE is may be risk
```

This shows there may be risks involved in investing.

By taking different dataset to identify same target, it is possible to incur that it suggests investing in stock ,if both the models results positive or else it suggests risk involved when both the models gives opposite results.

TASK-2

Collect data (images) and annotate them for two classes: Person and vehicle. You may use platforms such as LabelImg for annotations. You may limit to 800 images for the dataset. Perform object detection on your collected dataset and find the mean distance between the two classes in each image. You may use YOLOv5 for detection.

Description:

Importing the necessary libraries and modelling yolov5s and the dataset used is the PASCAL Visual Object Classes (VOC) dataset, containing 20 object categories including vehicles, household, animals, and other: aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, TV/monitor, bird, cat, cow, dog, horse, sheep, and person. Using the yolov5s version, As we want only to detect persons and vehicles: Limiting the classes to 0 (person), 1(bicycle), 2(car), 3(motorcycles), 5(bus), 7(truck) based on the VOC dataset.

```
!pip install torch==1.8.1+cu111 torchvision==0.9.1+cu111 torchaudio===0.8.1 -f https://download.pytorch.org/whl/lts/1.8/torch_lts.html

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Looking in links: https://download.pytorch.org/whl/lts/1.8/torch_lts.html
Collecting torch==1.8.1+cu111
  Downloading https://download.pytorch.org/whl/lts/1.8/cu111/torch-1.8.1%2Bcu111-cp38-cp38-linux_x86_64.whl (1982.2 MB)
     |▌                               | 834.1 MB 1.3 MB/s eta 0:14:37tcmalloc: large alloc 1147494400 bytes == 0x655dc000 @  0x7ff2f593
     |▊                               | 1055.7 MB 1.2 MB/s eta 0:12:23tcmalloc: large alloc 1434370048 bytes == 0x2e04000 @  0x7ff2f593
     |▉                               | 1336.2 MB 1.2 MB/s eta 0:08:46tcmalloc: large alloc 1792966656 bytes == 0x585f0000 @  0x7ff2f59
     |▉                               | 1691.1 MB 1.2 MB/s eta 0:04:09tcmalloc: large alloc 2241208320 bytes == 0xc33d8000 @  0x7ff2f59
     |▉                               | 1982.2 MB 1.2 MB/s eta 0:00:01tcmalloc: large alloc 1982177280 bytes == 0x2e04000 @  0x7ff2f593
tcmalloc: large alloc 2477727744 bytes == 0x148d3a000 @  0x7ff2f5931615 0x5d6f4c 0x51edd1 0x51ef5b 0x4f750a 0x4997a2 0x55cd91 0x5d8941
     |▉                               | 1982.2 MB 6.4 kB/s
Collecting torchvision==0.9.1+cu111
  Downloading https://download.pytorch.org/whl/lts/1.8/cu111/torchvision-0.9.1%2Bcu111-cp38-cp38-linux_x86_64.whl (17.6 MB)
     |▉                               | 17.6 MB 701 kB/s
Collecting torchaudio===0.8.1
  Downloading torchaudio-0.8.1-cp38-cp38-manylinux1_x86_64.whl (1.9 MB)
     |▉                               | 1.9 MB 1.4 MB/s
```

Testing the model on an image

```
img='https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcT_Xdwg90bW1jK4WrvmeEraLwpiqNk1a-CGgg&usqp=CAU'
```

```
results = model(img)
results.print()
```

```
image 1/1: 168x300 1 person, 11 cars
Speed: 23.5ms pre-process, 12.8ms inference, 1.7ms NMS per image at shape (1, 3, 384, 640)
```

```
%matplotlib inline
plt.imshow(np.squeeze(results.render()))
plt.show()
```


```
img='https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRhXEbZxhbaVku9wHaV50UGd9iFdWd9LJCKvw&usqp=CAU'
```

```
results = model(img)
results.print()
```

```
image 1/1: 183x275 1 truck
Speed: 30.8ms pre-process, 15.1ms inference, 1.7ms NMS per image at shape (1, 3, 448, 640)
```

```
%matplotlib inline
plt.imshow(np.squeeze(results.render()))
plt.show()
```

The model is able to localize vehicles and people perfectly calculating mean distances between the classes. The midpoint of the objects is calculated and the mean distances of these two points are calculated.

The distance between the two objects is plotted on the image.

TASK-3

Download an image dataset of your choice for binary class classification. Perform the data augmentation techniques like flipping, rotation and transformation. Apply at least two object classification techniques both on the augmented as well as on the original dataset. Display the performance of the Algorithms. Prepare a comparison chart.

Description:

The dataset chosen is Cat and Dog

```
_,ax = plt.subplots(3,3, figsize=(12,12))
for i in range(3):
    for j in range(3):
      ax[i,j].imshow(cv2.cvtColor(images[(i*200)+j], cv2.COLOR_BGR2RGB))
      ax[i,j].axis('off')
      ax[i,j].set_title(le.inverse_transform(ct_types_encoded[(i*200)+j]), size = 20)
```

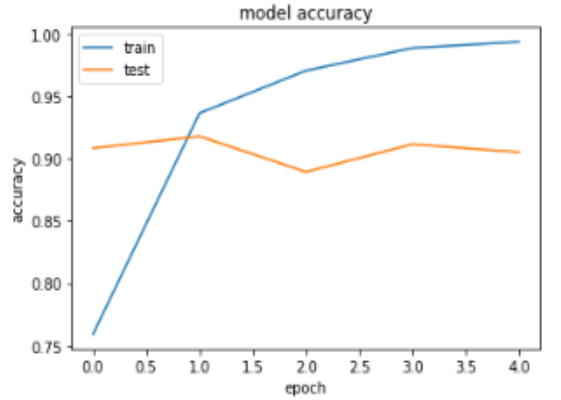
For Data Augmentation and For Image Classification, The following two models were considered

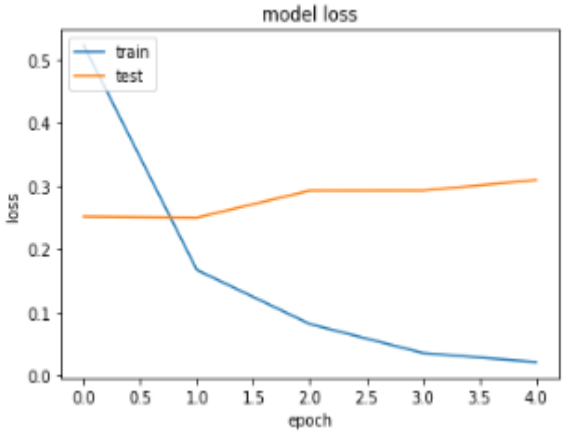
- Transfer learning - ResNet50
- Baseline convolutional neural network

Transfer learning:

ResNet50 allows the model to skip one or more layers. This approach makes it possible to train the network on many layers without affecting performance.

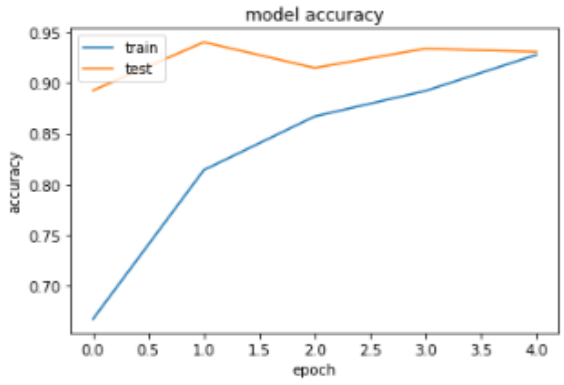
Without Augmentation

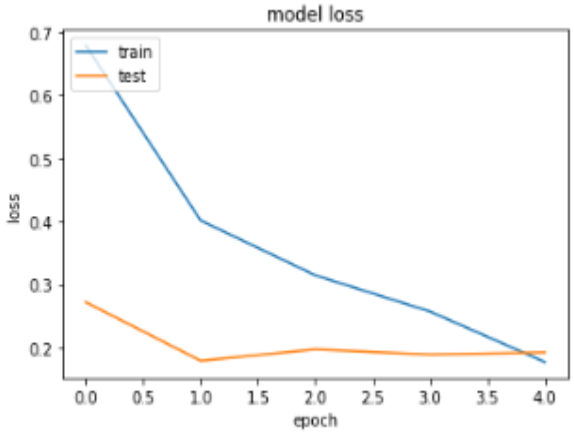




The model overfits the data in very few epochs.

With Augmentation

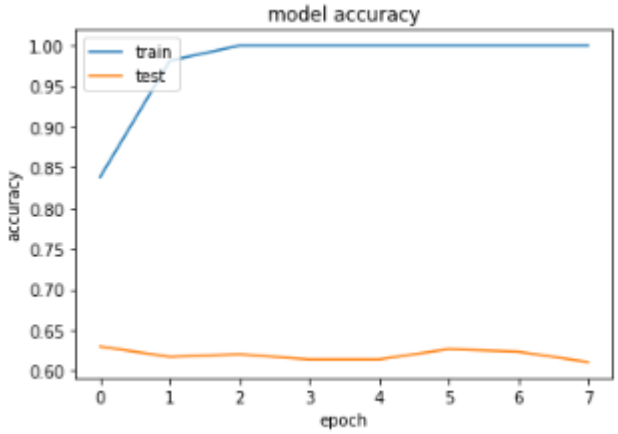


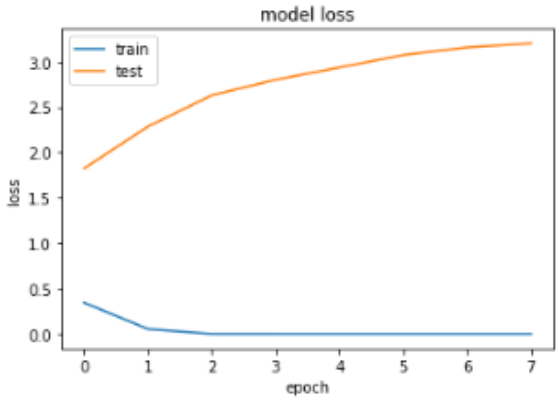


The model is able to fit the data very well.

Baseline convolutional neural network:

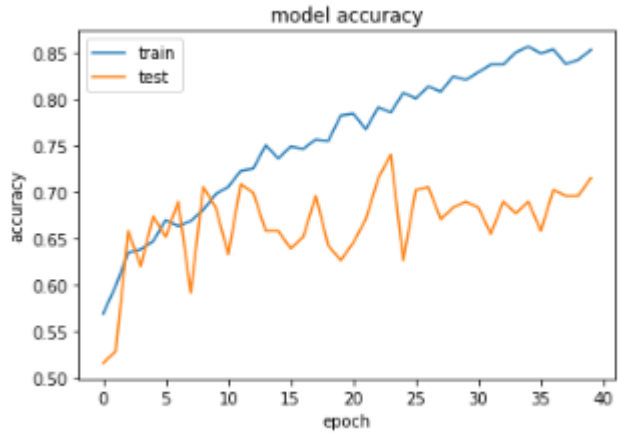
Without Augmentation

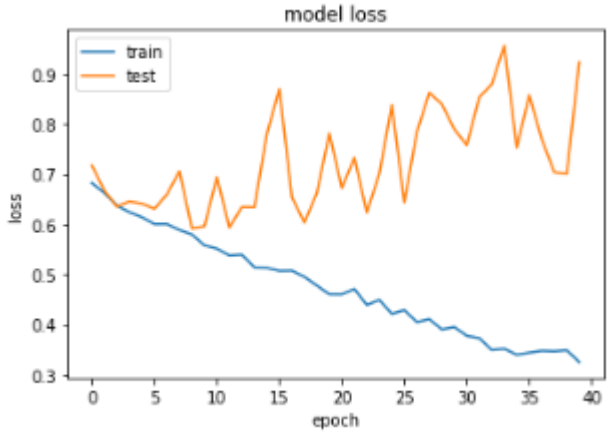




From the graph it is very clear that the model fits well for train data and not very well for test data, this is a case of overfitting.

With Augmentation.





This model can now be able to learn and classify without overfitting.

Comparatively transfer learning with augmentation gives better results than baseline models.

TASK-4

Collect images of vehicles with license plates written in Indian regional languages (eg. Hindi, Kannada, Tamil, Telugu, Bengali, etc.). Apply Image augmentation techniques on the collected images. Maintain separate folders for different language license plates. You may limit to 800 images in the dataset including the augmented images.

Description:

The images for the dataset were randomly collected from different sites for the religional languages like Hindi, Kannada, Tamil, Bengali.

The link for the dataset,

Augmenting a total of 20 images on every image available in each class Loading the dataset and importing the necessary libraries.

```
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
```

```
Mounted at /content/drive
```

```
!pip install unrar
!unrar x /content/drive/MyDrive/tsk4/task4/data.rar/
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: unrar in /usr/local/lib/python3.8/dist-packages (0.4)

UNRAR 5.50 freeware      Copyright (c) 1993-2017 Alexander Roshal


Extracting from /content/drive/MyDrive/tsk4/task4/data.rar
```

These images were augmented based on

```
datagen = ImageDataGenerator(
          rotation_range=40,
          width_shift_range=0.2,
          height_shift_range=0.2,
          shear_range=0.25,
          brightness_range = [0.2,1.8],
          zoom_range=0.2,
          horizontal_flip=True,
          fill_mode='constant')
```

```
import os.path
from os import path

if path.exists('/content/drive/MyDrive/tsk4/task4/hindi_aug') == False:
  os.mkdir('/content/drive/MyDrive/tsk4/task4/Augmented Images/Hindi_augg')
if path.exists('/content/drive/MyDrive/tsk4/task4/kannada_aug') == False:
  os.mkdir('/content/drive/MyDrive/tsk4/task4/Augmented Images/Kannada_augg')
if path.exists('/content/drive/MyDrive/tsk4/task4/tamil_aug') == False:
  os.mkdir('/content/drive/MyDrive/tsk4/task4/Augmented Images/Tamil_augg')
if path.exists('/content/drive/MyDrive/tsk4/task4/bengali_aug') == False:
  os.mkdir('/content/drive/MyDrive/tsk4/task4/Augmented Images/Bengali_augg')
```

```
dct = {}

train_dir ='/content/drive/MyDrive/tsk4/task4/data/'

list_subfolders_with_paths = [f.path for f in os.scandir(train_dir) if f.is_dir()]

for i in list_subfolders_with_paths:
    if i.split('/')[3] in ['hindi','kannada','tamil', 'bengali']:
        dct[i.split('/')[3]] = [f.path for f in os.scandir(i) if f.path.endswith('.jpg')]

dataset = pd.DataFrame.from_dict(dct, orient='index').transpose()
```

Image Augmentation

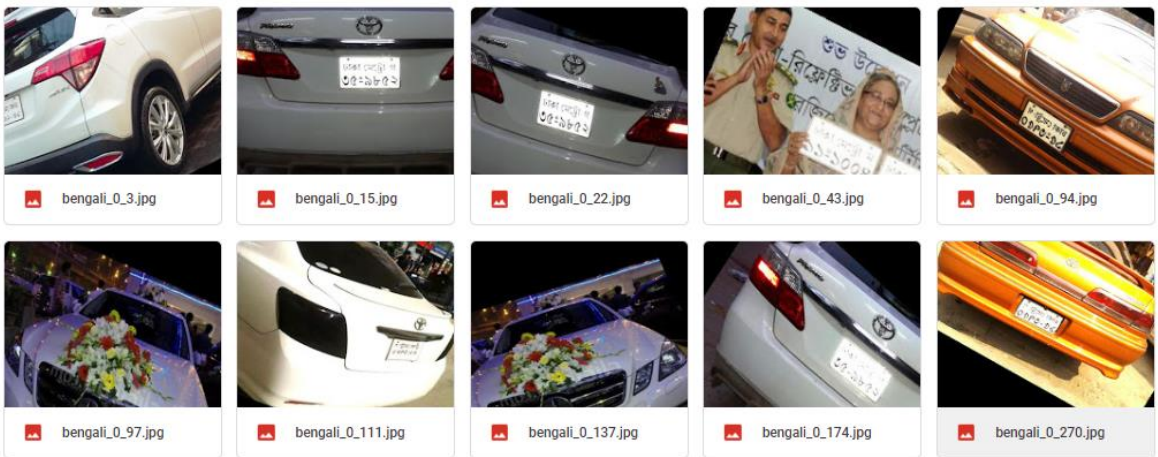
```
for idx, files in file_list.items():
    for file in files :
      # print(file)
      if file != None:
        img = load_img(file)
        x = img_to_array(img)
        x = x.reshape((1,) + x.shape)

        if idx=='hindi':
          i = 0
          for batch in datagen.flow(x, batch_size=1,
                                    save_to_dir='/content/drive/MyDrive/tsk4/task4/Augmented_Images/Hindi_augg', save_prefix='hindi', save_format='jpg'):
              i += 1
              if i > 20:
                  break
        elif idx=='kannada':
          i = 0
          for batch in datagen.flow(x, batch_size=1,
                                    save_to_dir='/content/drive/MyDrive/tsk4/task4/Augmented_Images/kannada_augg', save_prefix='hindi', save_format='jpg'):
              i += 1
              if i > 20:
                  break
        elif idx == 'tamil':
            i = 0
            for batch in datagen.flow(x, batch_size=1,
                                      save_to_dir='/content/drive/MyDrive/tsk4/task4/Augmented_Images/Tamil_augg', save_prefix='tamil', save_format='jpg'):
                i += 1
                if i > 20:
                    break
        else:
            i = 0
            for batch in datagen.flow(x, batch_size=1,
                                      save_to_dir='/content/drive/MyDrive/tsk4/task4/Augmented_Images/Bengali_augg', save_prefix='bengali', save_format='jpg'):
                i += 1
                if i > 20:
                    break
```

The augmented images are saved in the folder below, https://drive.google.com/drive/folders/1Sf8Qj8pEu_wpdw8PYElsBJYJq8x-HQkI

Few Sample output:

Bengali



Tamil

TASK-5

Conversion of PyTorch checkpoint file to .ONNX file.

Create a simple CNN module and train it on the MNIST dataset For simplicity 2 epochs are run Save the model as .pth file

```
torch.save(model.state_dict(), 'task5.pth')
```

To export a model torch.onnx.export function is used - This will execute the model, recording a trace of what operators are used to compute the outputs.

```
trained_model = Net()
trained_model.load_state_dict(torch.load('task5.pth'))

dummy_input = Variable(torch.randn(1, 1, 28, 28))
torch_out = trained_model(dummy_input)
torch.onnx.export(trained_model, dummy_input, "task5.onnx")
```

Finally, the model is exported as .ONNX file

```
!pip install onnxruntime
```

```
import onnxruntime
import numpy as np

ort_session = onnxruntime.InferenceSession("task5.onnx")

def to_numpy(tensor):
    return tensor.detach().cpu().numpy() if tensor.requires_grad else tensor.cpu().numpy()

# compute ONNX Runtime output prediction
ort_inputs = {ort_session.get_inputs()[0].name: to_numpy(dummy_input)}
ort_outs = ort_session.run(None, ort_inputs)

# compare ONNX Runtime and PyTorch results
np.testing.assert_allclose(to_numpy(torch_out), ort_outs[0], rtol=1e-03, atol=1e-05)

print("Exported model has been tested with ONNXRuntime, and the result looks good!")
```