Teammates:

Sukumar Bodapati – sb5zh@umsystem.edu - 16326105

GitHub link: https://github.com/UMKC-APL-BigDataAnalytics/icp-4-sukumarbodapati

Sri Nikhitha Boddapati – sb4dz@umsystem.edu - 16322565

GitHub link: https://github.com/UMKC-APL-BigDataAnalytics/icp-4-Srinikhitha98

Video link: https://youtu.be/XjlotcEOf2Q

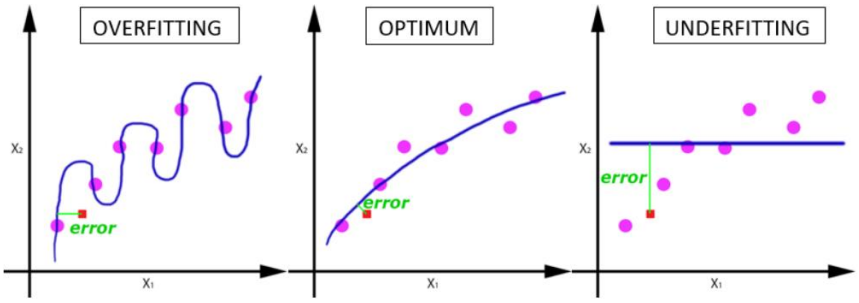
ICP - Big Data App & Analytics – Deep Learning on Image data

Objective:

Change the hyperparameters of the model that we used as part of the ICP4 and validate 5 different images using the model.

Parameters:

- **Epoch:** It is one of the hyperparameter. It is the number of times the algorithm is applied on the dataset. We have increased the epoch size to 25 which has increase weights of the neural networks.
  Epoch size: 10: Images are predicted with less accuracy and loss is medium.
  Epoch size: 25: Images are predicted with high accuracy and loss is less.



When we increase epoch size if the graph is in underfitting, it goes to optimum and if we increase beyond the dataset goes to overfitting. As per this dataset, epoch=25 gives the optimum output.

- **Batch size:** If batch size is too small, then we are observing the noise in the output. So have increase the Batch size from 32 to 64. If the batch size is too high for the dataset it lead to the overfitting. So tried the batch size with 32,64,128.
  For this model and data set , the optimum batch size is 32
- **Optimizer:** They are algorithm's that are used to change the weight and learning rate to reduce the loss of the data while processing through the model.

SGD: Used this optimizer initially on this dataset. Since this is huge data set and SGD performs frequency update with high variance, the accuracy of the model is not good.
Rmsprop: Used this optimizer as well but Adam is a combination of RMSprop and momentum.

For this set and model, **Adam** is the best optimizer.

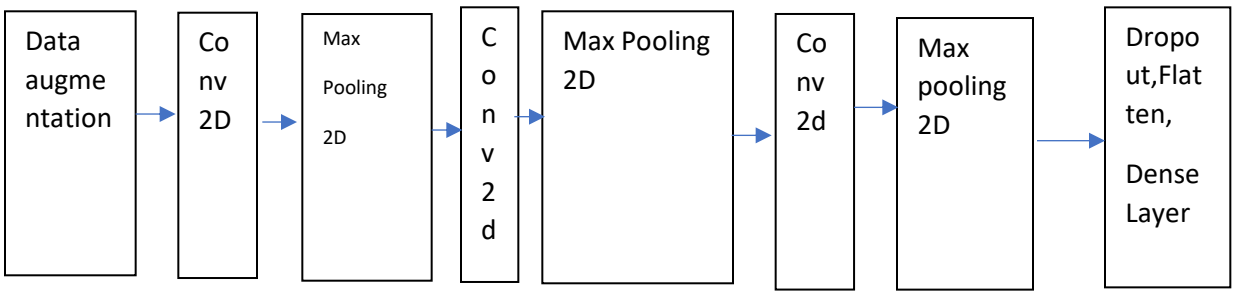
- **Data Augmentation:** To predict the new data with more accuracy, need to duplicate the same image using rotation. Initially developed a model without Data Augmentation the newly added images have been predicted with less accuracy. Then we have implemented the horizontal random flip with rotation of 0.05. Now we can predict images with more accuracy.
- **Loss Function:**
    a. categorical_crossentropy : It is the loss function which uses one-hot encoded vectors.
    b. sparse_categorial_crossentropy :It is a loss function which is similar to categorical_crossentropy but it uses integers.

  In order to save time and computation since sparse loss function uses a single number, we have used sparse categorial crossentropy.

- **Dropout:**
  Dropout is a technique to reduce overfitting of the dataset. Intially we have used dropout of 0.2 but it is leading to overfitting so have used the dropout of 0.1.

Model Building:

In this ICP, we have rebuilt the model again changing hyperparameter's and also the layer to increase the accuracy & decrease the loss of the model.



In this model, first we processed the data set through data augmentation, we have 3 convolution layer with same padding along with max pooling layer .

Convolution layer: Convolutional layers are good for feature extraction from images as they deal with the spatial redundancy by weight sharing. So we have used 3 convolution layers.

Max Pooling: It downscales the image if the image is not used and replaces it with convolution.

Difference between the last model and this ICP model: The difference of the predicted values between training and validation accuracy is very less. Using this model we have predicted the images with more than 95 per cent confidence level.

Code Tasks:

1.Import all the libraries

```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tsf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.datasets import cifar10
from keras.layers import Dense,Dropout,Flatten,BatchNormalization,Activation
from keras.layers.convolutional import Conv2D,MaxPooling2D
```

2.Load the data:

```
[4]  #Loading in the data

     (x_tr, y_tr), (x_te, y_te) = cifar10.load_data()
```

3.If we print training and testing values there are around 50000 training sets and 10000 testing sets in cifar data

4.There are around 10 unique classes in cifar data.

They are: ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'].These are the labels of the data present in first 5 samples.

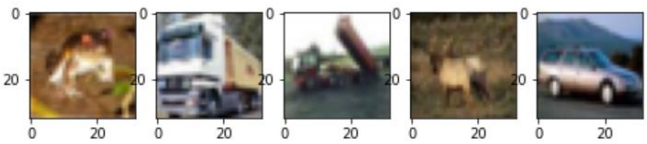
```
]  print('Trained images and labels' + str([x[0] for x in y_tr[0:5]]))
   print('Corresponding classes for the labels: ' + str([cif_class[x[0]] for x in y_tr[0:5]]))

   Trained images and labels[6, 9, 9, 4, 1]
   Corresponding classes for the labels: ['frog', 'truck', 'truck', 'deer', 'automobile']
```

5.Plot the images on the graph:

```
f, axarr = plt.subplots(1, 5)
f.set_size_inches(8, 8)

for i in range(5):
    img = x_tr[i]
    axarr[i].imshow(img)
plt.show()
```



6.Create the duplicates using the data augmentation to improve accuracy of the image prediction.

```
[10] data_augmentation = keras.Sequential(
        [
         layers.experimental.preprocessing.RandomFlip("horizontal", input_shape=(32,32,3)),
         layers.experimental.preprocessing.RandomRotation(0.05),
         layers.experimental.preprocessing.RandomZoom(0.05),
         layers.experimental.preprocessing.Rescaling(1./255),
        ]
    )
```

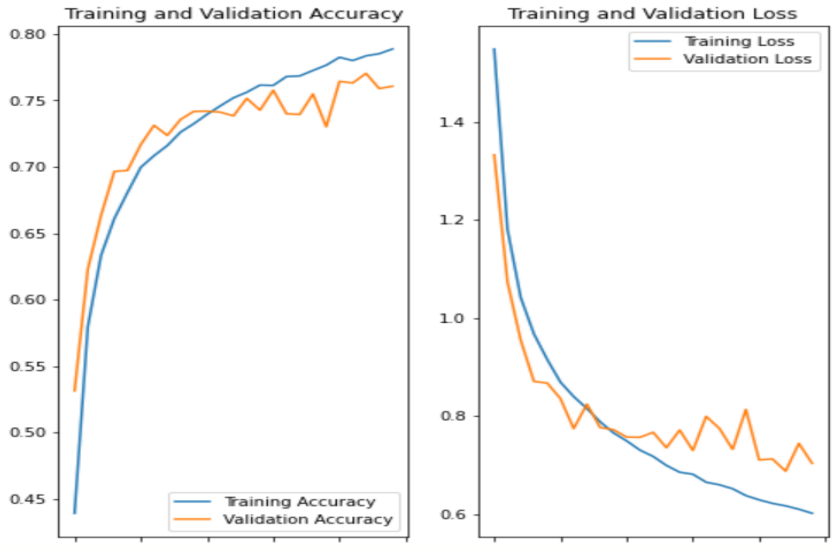
7.Model Building:

```
class_num = 10
#Build the model
mod_builder = Sequential()
#Pass through data augmentation
mod_builder.add(data_augmentation)
mod_builder.add(Conv2D(32, 3, input_shape=x_tr.shape[1:], padding='same',activation='relu'))
mod_builder.add(MaxPooling2D())
mod_builder.add(Conv2D(64, 3, input_shape=x_tr.shape[1:], padding='same',activation='relu'))
mod_builder.add(MaxPooling2D())
mod_builder.add(Conv2D(128,3, input_shape=x_tr.shape[1:], padding='same',activation='relu'))
mod_builder.add(MaxPooling2D())
mod_builder.add(Dropout(0.1))
mod_builder.add(Flatten())
mod_builder.add(Dense(64,activation='relu'))
mod_builder.add(Dense(class_num))
```

8.Model complied:

```
] mod_builder.compile(optimizer='adam',
                      loss=tsf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                      metrics=['accuracy'])
```

9.Training and accuracy of this model:



Predict the New data set:

1.Plane image:

Used the plane image :

Code for prediction

```
plane_url = "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTuy1RHZVySNssgJ7i_tA96iaDlpW8IVPAU3g&usqp=CAU"
plane_path = tf.keras.utils.get_file('airplane', origin=plane_url)
img = keras.preprocessing.image.load_img(
    plane_path, target_size=(32, 32)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = mod_builder.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(cif_class[np.argmax(score)], 100 * np.max(score))
)
```

```
Downloading data from https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTuy1RHZVySNssgJ7i_tA96iaDlpW8IVPAU3g&usqp=CAU
16384/2992 [==========================================================================================================================]
This image most likely belongs to airplane with a 100.00 percent confidence.
```

2.CAR

Image 2:Car

```
car_url = "https://th.bing.com/th/id/OIP.9bThB-0dbI38LLIcQCL7hQHaEK?w=279&h=180&c=7&r=0&o=5&dpr=1.5&pid=1.7"
car_path = tf.keras.utils.get_file('Car', origin=car_url)
img = keras.preprocessing.image.load_img(
    car_path, target_size=(32, 32)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = mod_builder.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(cif_class[np.argmax(score)], 100 * np.max(score))
)
```

```
Downloading data from https://th.bing.com/th/id/OIP.9bThB-0dbI38LLIcQCL7hQHaEK?w=279&h=180&c=7&r=0&o=5&dpr=1.5&pid=1.7
24576/18941 [==================================] - 0s 0us/step
32768/18941 [==========================================] - 0s 0us/step
This image most likely belongs to automobile with a 96.14 percent confidence.
```

3.Horse:

```
[23] horse_url = "https://media.istockphoto.com/photos/horse-picture-id490961528?k=6&m=490961528&s=612x612&w=0&h=K9xn4cac5VXju2
horse_path = tf.keras.utils.get_file('Horse', origin=horse_url)
img = keras.preprocessing.image.load_img(
    horse_path, target_size=(32, 32)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = mod_builder.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(cif_class[np.argmax(score)], 100 * np.max(score))
)
```

```
Downloading data from https://media.istockphoto.com/photos/horse-picture-id490961528?k=6&m=490961528&s=612x612&w=0&h=K9xn4
49152/42208 [==================================] - 0s 0us/step
57344/42208 [==========================================] - 0s 0us/step
This image most likely belongs to horse with a 88.05 percent confidence.
```

4.Ship

```
[24] Ship_url = "https://th.bing.com/th/id/OIP.WU7vqOhXv_z68XmhKnXdnQHaE8?w=230&h=180&c=7&r=0&o=5&dpr=1.5&pid=1.7"
Ship_path = tf.keras.utils.get_file('Ship', origin=Ship_url)
img = keras.preprocessing.image.load_img(
    Ship_path, target_size=(32, 32)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = mod_builder.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(cif_class[np.argmax(score)], 100 * np.max(score))
)
```

```
Downloading data from https://th.bing.com/th/id/OIP.WU7vqOhXv_z68XmhKnXdnQHaE8?w=230&h=180&c=7&r=0&o=5&dpr=1.5&pid=1.7
24576/20856 [==================================] - 0s 0us/step
32768/20856 [==========================================] - 0s 0us/step
This image most likely belongs to ship with a 99.98 percent confidence.
```

5.Bird

```
[25] Bird_url = "
     egF69X1vpOpup61sb7iiTtGCvkV5MowYhgQRgg9jSJ4HjdSPQ6XMs0E/c7M12KgFWpVfBeS4KxUAFdPxXBTVD3BZYRBrtckCqs9dtvgWzrUMmpuk1wCaJYmuyVE6D8/tUqwT6fepTNgW
     01L13VG36igbrdlZuBiMZGKwtOdT1t21Gj0GzTLCFLrIjuHPLRDbfGaV6wdM1qx/DAXGOoD6kbWYPaQg8ATFa/Ter6bqLa/ZcXT6bS20wAEfUXGBwZziKTKm+QtvFiF/8AC1jUXRZtOj
     Bird_path = tf.keras.utils.get_file('Bird', origin=Bird_url)
     img = keras.preprocessing.image.load_img(
         Bird_path, target_size=(32, 32)
     )
     img_array = keras.preprocessing.image.img_to_array(img)
     img_array = tf.expand_dims(img_array, 0) # Create a batch

     predictions = mod_builder.predict(img_array)
     score = tf.nn.softmax(predictions[0])

     print(
         "This image most likely belongs to {} with a {:.2f} percent confidence."
         .format(cif_class[np.argmax(score)], 100 * np.max(score))
     )
```

```
Downloading data from 
24576/16567 [==============================================] - 0s 0us/step
32768/16567 [==============================================================] - 0s 0us/step
This image most likely belongs to bird with a 80.47 percent confidence.
```

Challenges:

1.It was very difficult how model changes by changing hyperparameters.

2.Adding different layers in the model is the big challenge after changing hyperparameters is difficult.