Basic Image Classifier using CNN

- **Prithvi Raj .M(106121097)**

Problem Statement:

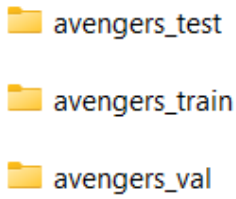
- Create a labeled dataset of Avengers images- Captain America, Iron Man, Black Widow, Hulk, Thor. (Try *scraping* images from the internet instead of manually creating the dataset)
- Train a CNN that is able to classify an unseen image with reasonable accuracy.

My Approach:

- I used a library called “Bing Image Downloader” library, which lets us download the desired images from Bing image search.

```
downloader.download('iron man', limit=40,  output_dir='avengers_val', timeout=120, verbose=True)
```

- It eased the task of downloading a ton of images from the internet. The downloaded images are stored in a new folder named after the output_dir parameter.
- I stored the images in 3 folders, namely-



- I split a total of 800 images into 100,40 and 20 pictures for training, validation, and testing for each of the five characters.
- As I worked on the task using Google Colab, I had difficulty uploading the dataset as a whole. So, I used a zip file and Zipfile library in python to access the dataset from Colab.

```
from zipfile import ZipFile
filename= "/content/drive/MyDrive/Databyte task 1.zip"
with ZipFile(filename,'r')as zip:
  zip.extractall()
```

Core Idea:

- Tensorflow was the backbone of the whole model.
- I used Keras to randomize and resize the input dataset.

```
image_train = keras.preprocessing.image.ImageDataGenerator(
    rescale = 1./255,
    width_shift_range = 0.12,
    height_shift_range = 0.12,
    horizontal_flip=True
)
```

- The model consisted of 3 consecutive pairs of 2D Convolutional – Max Pooling layers, with padding used in many layers.
- I used ReLU to activate nodes in each layer, except the final layer, where the softmax function was recommended.
- After going through many blogs that said Adam optimizer was the best-suited one, I decided to use it.
- I divided the dataset into batches of size 50 and fed them to the model.
- Using matplotlib, I was also able to plot the model accuracy and loss graph in training and the testing dataset.

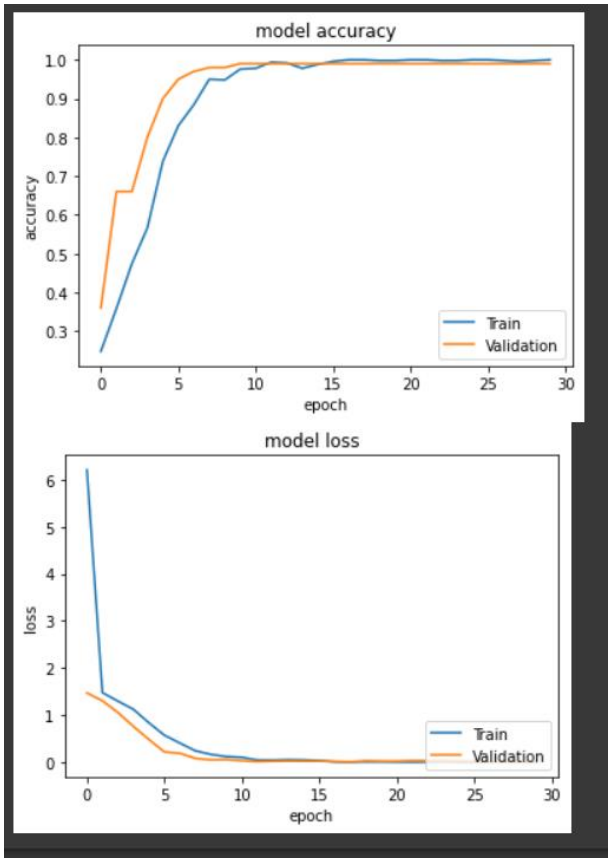
```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='lower right')
plt.show()
```

My Observations:

- I observed three accuracy trends when changing the dataset or the model parameters.
- At first, I fed the image dataset into the model with no height, width, or zoom variations.
- The model quickly predicted the classes and learned at a high rate.

```
  # Remove the CWD from sys.path while we load stuff.
Epoch 1/30
10/10 [==============================] - 37s 3s/step - loss: 6.2070 - accuracy: 0.2480 - val_loss: 1.4706 - val_accuracy: 0.3600
Epoch 2/30
10/10 [==============================] - 28s 3s/step - loss: 1.4763 - accuracy: 0.3580 - val_loss: 1.3044 - val_accuracy: 0.6600
Epoch 3/30
10/10 [==============================] - 27s 3s/step - loss: 1.2964 - accuracy: 0.4740 - val_loss: 1.0539 - val_accuracy: 0.6600
Epoch 4/30
10/10 [==============================] - 27s 3s/step - loss: 1.1222 - accuracy: 0.5660 - val_loss: 0.7605 - val_accuracy: 0.8000
Epoch 5/30
10/10 [==============================] - 26s 3s/step - loss: 0.8366 - accuracy: 0.7380 - val_loss: 0.4833 - val_accuracy: 0.9000
Epoch 6/30
10/10 [==============================] - 28s 3s/step - loss: 0.5728 - accuracy: 0.8300 - val_loss: 0.2176 - val_accuracy: 0.9500
Epoch 7/30
10/10 [==============================] - 27s 3s/step - loss: 0.4055 - accuracy: 0.8840 - val_loss: 0.1902 - val_accuracy: 0.9700
Epoch 8/30
10/10 [==============================] - 27s 3s/step - loss: 0.2419 - accuracy: 0.9500 - val_loss: 0.0808 - val_accuracy: 0.9800
Epoch 9/30
10/10 [==============================] - 27s 3s/step - loss: 0.1660 - accuracy: 0.9480 - val_loss: 0.0502 - val_accuracy: 0.9800
Epoch 10/30
10/10 [==============================] - 27s 3s/step - loss: 0.1191 - accuracy: 0.9760 - val_loss: 0.0548 - val_accuracy: 0.9900
Epoch 11/30
10/10 [==============================] - 26s 3s/step - loss: 0.1032 - accuracy: 0.9780 - val_loss: 0.0307 - val_accuracy: 0.9900
Epoch 12/30
10/10 [==============================] - 28s 3s/step - loss: 0.0462 - accuracy: 0.9940 - val_loss: 0.0130 - val_accuracy: 0.9900
Epoch 13/30
10/10 [==============================] - 27s 3s/step - loss: 0.0446 - accuracy: 0.9920 - val_loss: 0.0214 - val_accuracy: 0.9900
Epoch 14/30
10/10 [==============================] - 28s 3s/step - loss: 0.0535 - accuracy: 0.9780 - val_loss: 0.0257 - val_accuracy: 0.9900
Epoch 15/30
10/10 [==============================] - 28s 3s/step - loss: 0.0479 - accuracy: 0.9880 - val_loss: 0.0224 - val_accuracy: 0.9900
```



- The accuracy levels began to drop when I randomized the dataset. Also, the dropout layer parameter greatly affected the predicting ability of the model. The second trial:

```
Epoch 1/15
10/10 [==============================] - 51s 5s/step - loss: 5.0616 - accuracy: 0.2160 - val_loss: 1.5843 - val_accuracy: 0.2800
Epoch 2/15
10/10 [==============================] - 48s 5s/step - loss: 1.5926 - accuracy: 0.2720 - val_loss: 1.5557 - val_accuracy: 0.3300
Epoch 3/15
10/10 [==============================] - 47s 5s/step - loss: 1.5564 - accuracy: 0.3100 - val_loss: 1.3972 - val_accuracy: 0.4700
Epoch 4/15
10/10 [==============================] - 49s 5s/step - loss: 1.4522 - accuracy: 0.4140 - val_loss: 1.1819 - val_accuracy: 0.5700
Epoch 5/15
10/10 [==============================] - 48s 5s/step - loss: 1.3421 - accuracy: 0.4400 - val_loss: 1.0514 - val_accuracy: 0.6800
Epoch 6/15
10/10 [==============================] - 48s 5s/step - loss: 1.2473 - accuracy: 0.4500 - val_loss: 1.0156 - val_accuracy: 0.5400
Epoch 7/15
10/10 [==============================] - 48s 5s/step - loss: 1.2720 - accuracy: 0.4740 - val_loss: 1.1090 - val_accuracy: 0.5600
Epoch 8/15
10/10 [==============================] - 46s 5s/step - loss: 1.2110 - accuracy: 0.5020 - val_loss: 0.8637 - val_accuracy: 0.6000
Epoch 9/15
10/10 [==============================] - 49s 5s/step - loss: 1.1353 - accuracy: 0.5400 - val_loss: 0.8056 - val_accuracy: 0.7000
Epoch 10/15
10/10 [==============================] - 48s 5s/step - loss: 1.1089 - accuracy: 0.5480 - val_loss: 0.7942 - val_accuracy: 0.7600
Epoch 11/15
10/10 [==============================] - 46s 5s/step - loss: 1.1018 - accuracy: 0.5480 - val_loss: 0.7499 - val_accuracy: 0.7300
Epoch 12/15
10/10 [==============================] - 47s 5s/step - loss: 1.1542 - accuracy: 0.5260 - val_loss: 0.8727 - val_accuracy: 0.7100
Epoch 13/15
10/10 [==============================] - 48s 5s/step - loss: 1.1061 - accuracy: 0.5860 - val_loss: 0.6864 - val_accuracy: 0.7400
Epoch 14/15
10/10 [==============================] - 46s 5s/step - loss: 1.0116 - accuracy: 0.5840 - val_loss: 0.6486 - val_accuracy: 0.7600
Epoch 15/15
10/10 [==============================] - 46s 5s/step - loss: 1.0097 - accuracy: 0.6260 - val_loss: 0.7407 - val_accuracy: 0.7100
```

- And in the final trial, I fixed the dropout parameter to 0.4 and reduced the number of epochs from 30 to 15.

```
Epoch 1/15
10/10 [==============================] - 50s 5s/step - loss: 9.5532 - accuracy: 0.2120 - val_loss: 1.8005 - val_accuracy: 0.2600
Epoch 2/15
10/10 [==============================] - 47s 5s/step - loss: 1.6363 - accuracy: 0.2380 - val_loss: 1.5267 - val_accuracy: 0.2800
Epoch 3/15
10/10 [==============================] - 47s 5s/step - loss: 1.5713 - accuracy: 0.2640 - val_loss: 1.3956 - val_accuracy: 0.3900
Epoch 4/15
10/10 [==============================] - 46s 5s/step - loss: 1.4901 - accuracy: 0.3600 - val_loss: 1.2834 - val_accuracy: 0.5300
Epoch 5/15
10/10 [==============================] - 46s 5s/step - loss: 1.4790 - accuracy: 0.4240 - val_loss: 1.2212 - val_accuracy: 0.5500
Epoch 6/15
10/10 [==============================] - 46s 5s/step - loss: 1.4274 - accuracy: 0.4300 - val_loss: 1.0893 - val_accuracy: 0.6500
Epoch 7/15
10/10 [==============================] - 46s 5s/step - loss: 1.3361 - accuracy: 0.4720 - val_loss: 1.0585 - val_accuracy: 0.6400
Epoch 8/15
10/10 [==============================] - 48s 5s/step - loss: 1.2895 - accuracy: 0.5080 - val_loss: 0.9442 - val_accuracy: 0.7200
Epoch 9/15
10/10 [==============================] - 46s 5s/step - loss: 1.2732 - accuracy: 0.5060 - val_loss: 0.8854 - val_accuracy: 0.6800
Epoch 10/15
10/10 [==============================] - 46s 5s/step - loss: 1.2137 - accuracy: 0.5380 - val_loss: 0.9000 - val_accuracy: 0.7300
Epoch 11/15
10/10 [==============================] - 46s 5s/step - loss: 1.1583 - accuracy: 0.5440 - val_loss: 0.7957 - val_accuracy: 0.7500
Epoch 12/15
10/10 [==============================] - 46s 5s/step - loss: 1.1737 - accuracy: 0.5580 - val_loss: 0.7573 - val_accuracy: 0.7400
Epoch 13/15
10/10 [==============================] - 47s 5s/step - loss: 1.1339 - accuracy: 0.5980 - val_loss: 0.8004 - val_accuracy: 0.7000
Epoch 14/15
10/10 [==============================] - 46s 5s/step - loss: 1.1326 - accuracy: 0.5800 - val_loss: 0.7578 - val_accuracy: 0.8000
Epoch 15/15
10/10 [==============================] - 46s 5s/step - loss: 1.0740 - accuracy: 0.5940 - val_loss: 0.7707 - val_accuracy: 0.7500
```