ASSIGNMENT 2

N Srivijay Ram Prathap 2019H1030559G

Execution Environment : GOOGLE COLABS
#Dataset is accesed directy from the kaggle.

Problem Statement:

Writing a CNN model to classify the image contains a dog or a cat.
The available dataset consists of 25000 image of cats and dogs. I divide it into train and test of sizes 20000 and 5000 respectively.

Implementation:

Libraries Used:

- Tensorflow Keras models :- To implement the model.
- Tensorflow Keras layers :- To add layers to the model.

Layers Needed for CNN:

- Conv2D: This is a basic convolution layer in which the we are using 64 neurons to train the model.
- Dense : this layer is used to predict the final output of the CNN.
- Maxpooling : this layer is required to get the max value from each kernel output.
- Flatten : It involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing.

Data Processing:

- Test set is seperated from the trainset. 2500 dog images and 2500 cat images are moved from the train folder to test folder before fiting the model to train dataset.
- As the dataset contins only image with filename as cat and dog, depending upon the name made a lists for each image and its classification as X and Y, where X is the input for the model and Y shoud be the output for the classification.
- As each image is in different shape, used CV2 lib to reshape all the image to 80*80 dimension.Al the images are converted to greyscale. As most images are only dogs and cats and they have different colors in different breeds it would be better to stick with the patterns of their body. Color might not be huge deciding factor for the classification.
- As each Pixel density varies from 0 to 255 it is complicated to trian a model with these wide range of values. So normalize the pixel value. Ie divide each pixel value by 255 so the new range of values is 0-1.

Training:

- Added a con2d layer with 64 neurons and a filters of dimension 3,3. The activation function used for this layer is rectified linear unit which can be explained as max(0,x) function. This activation is common and basic activation function in CNN. Later add a maxpooling layer to this layer.
- Added the same conv2d layer with same specification to help model to be more flexible and powerful these layers can be added more also but 2 convd layers gave a good training accuracy. If we add one more layer it might be overfitting and it will take more time to train the model.
- As satisfied with two layers now we move to dense but to go for dense layer we need to flatten the data. So a flatten layers is added and then a dense layer of 64 neurons and one more dense layer of one neuron to final ouput value.
- The first dense layer has an activation function of Relu and for the second dense layer using a sigmoid function to get the probability output.
- For optimization using “adam” optimizer which is an adaptive learning rate optimization algorithm. Also tried sophisticated gradient decent, rmsprops but found better training accuracy with adam.
- For loss function using “binary_crossentropy” which is a cross entropy loss function. It is intended to use when the target values are in the set of {0,1}. And the metric for measure the training is accuracy.
- Fit the model to the training dataset with a validation split of 20% , batch size of 32 and epochs of 10.

Results:

Training and validaion accuracy and loss for each epoch :

Epoch 1/10
Training loss : 0.6655
Training accuracy: 0.5894
Validation loss: 0.6082
Validation accuracy: 0.6783

Epoch 2/10
Training loss: 0.5562
Training accuracy: 0.7141
Validation loss: 0.5543
Validation accuracy: 0.7155

Epoch 3/10
Training loss: 0.5067
Training accuracy: 0.7527
Validation loss: 0.5216

```
Validation accuracy: 0.7418


Epoch 4/10
Training loss: 0.4783
Training accuracy: 0.7690
Validation loss: 0.5212
Validation accuracy: 0.7485

Epoch 5/10
Training loss: 0.4503
Training accuracy: 0.7882
Validation loss: 0.4961
Validation accuracy: 0.7697

Epoch 6/10
Training loss: 0.4274
Training accuracy: 0.7990
Validation loss: 0.4978
Validation accuracy: 0.7660

Epoch 7/10
Training loss: 0.4005
Training accuracy: 0.8214
Validation loss: 0.4967
Validation accuracy: 0.7635

Epoch 8/10
Training loss: 0.3703
Training accuracy: 0.8339
Validation loss: 0.4938
Validation accuracy: 0.7810

Epoch 9/10
Training loss: 0.3422
Training accuracy: 0.8479
Validation loss: 0.4816
Validation accuracy: 0.7840

Epoch 10/10
Training loss: 0.3092

Training accuracy: 0.8659

Validation loss: 0.4960

Validation accuracy: 0.7768
```

ACCURACY ON TEST DATASET: 77.92 %

Conclusion:

The CNN models are so powerfull fit there datasets. Trying epochs of 20 made a training accuracy of which might be overfitting of the model. Using a 2 layer network with basic optimization techniques we are able to reach a accuracy of 77.92% on a 5000 size test dataset.