Embedded Systems (EEL3090)

Course Project

Topic: DL Model Optimization for Lightweight Visual Wake Words Task

By:- Aditi Tiwari (B20EE005)
Dhanushree Sisodiya (B20EE015)

Objective:

The objective of this project is to design a deep learning based model that can detect whether a person is present or not from web cameras or CCTV images.

Model Design:

Dataset:

The dataset consists of CCTV footage images (both indoor and outdoor), half of them with humans and half without humans.

Model:

- **We have implemented three types of models for this project**.

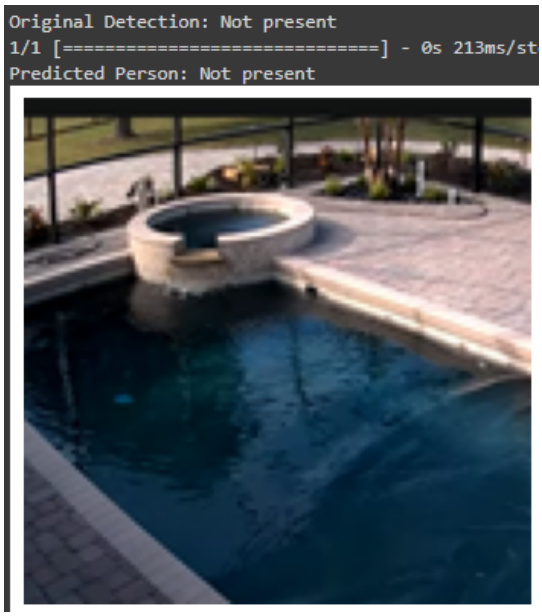
1. Pre-trained neural network model (using the VGG16 model)
2. Existing object detection model using OpenCV (Haar Cascade Detection Model)
3. Deep learning model (**Neural network Architecture**) from scratch.

- Firstly, we installed and imported dependencies such as Keras, Numpy, and Pandas libraries for designing neural architecture models, extracted information from images in terms of an array of pixels.

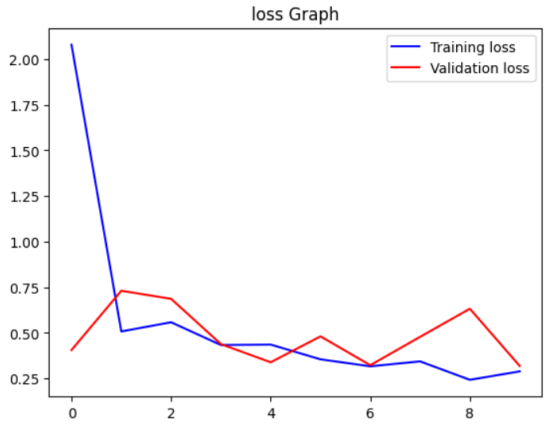
1) **Pre-trained Model**

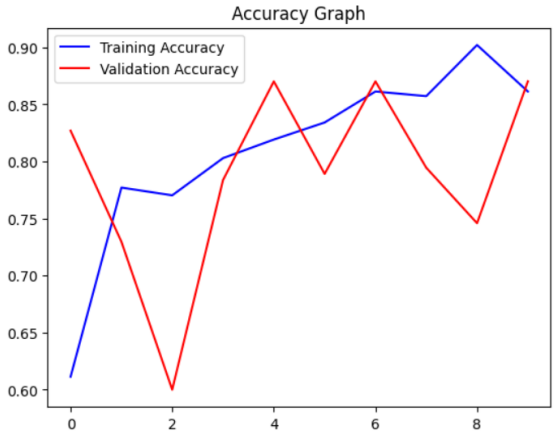
   Using the VGG16 model, we have implemented a pretrained model, which is a pre-built neural network model. We used this model for person detection.

   Predicted Results on Images-



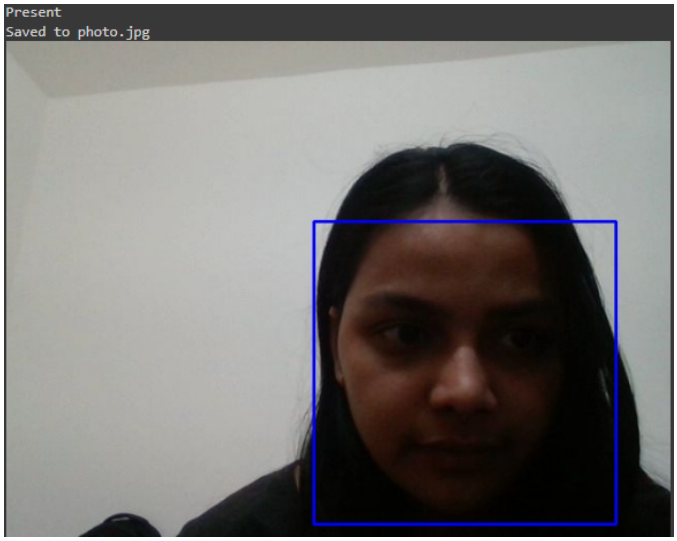
Accuracy and Loss Plot for Training and testing dataset -





Accuracy for VGG16 -
Train accuracy - 88.09%
Test accuracy - 81.62%

2) **Existing person detection model (Haar Cascade model)**
   It is a type of neural network model that identifies objects in images or videos. Here we are getting images using webcam access and detecting the availability of people. It makes a boundary box if it finds any person in the camera.
   Predicted result -



3) **Deep learning model from scratch**

We have implemented a deep learning neural network model from scratch.
Number of epochs (iteration) = 10
We trained our neural network model for 10 iterations over the dataset.

Part - i) Perform model robustness check, i.e., if the weights/layers

Results -

By changing the number of layers in a neural network -

Number of Layers	Train Accuracy	Test Accuracy	Train Loss	Test Loss
7 convolutions + 4 dense layers	86.06	66.49	0.285	1.7956
6 convolutions + 3 dense	81.06	72.97	0.4352	0.6106
4 convolutions + 3 dense layers	77.27	68.65	0.4470	0.8079
2 convolutions + 2 dense layers	97.43	68.11	0.0918	1.1150

Conclusion -

1. When we decrease the number of layers, train accuracy first decreases and then starts increasing (because of overfitting) , while test accuracy first increases (up to an optimum model) and then decreases.
2. Train losses increase as we decrease the number of layers, and test losses first increase then again decrease, while testing losses decrease up to optimum model and then starts increasing again.
3. The optimal model is of 6 convolution layers, and 3 dense layers because both, test accuracy as well as train accuracy are good in this case compared .

4. When our model is too deep or too shallow, our training dataset leads to overfitting, which is why we are getting low training accuracy in that case.
5. Our optimal model is a model that is neither too deep nor too shallow, as shown in the second case.

Part - ii) Reducing the size of the neural network design (in terms of the number of parameters),

Results -

Parameter in Last Conv layer (Out Channel)	Train Accuracy	Test Accuracy	Train loss	Test loss
512	81.73	79.46	0.3967	0.4985
256	82.95	70.27	0.4033	0.6267
128	80.78	69.73	0.4240	0.5873

Conclusion -

1. As we reduce the number of output channels (overall decreasing the size of the neural network in terms of the number of parameters) in the last convolution layer, both training and testing accuracy decrease.
2. As we reduce the number of output channels in the last convolution layer, train, and test, both losses increase, but in very small amounts.
3. Hence, the first case (512 output channels in the last convolution layer) is preferred for good neural network model design.

Part - iii) An explanation of how and why your model works -

In our deep neural network architecture design, we have two blocks. One consists of convolution, and the other contains dense (linear) layers.

How the algorithm is working -

Initially, our image(3 dimensional RGB image) size is (3*128*128). We are giving our image size as input to the neural network. Then we are convoluting our image with kernel size 3 with the Relu activation function (define boundaries) and giving

the convoluted image to the max pool (reducing image by factor 2) size 2. After all the convolutional layers and max pooling, we need to flatten our image in one dimension, and then we are using a dense layer, which takes the output of the last layer as input. After the last dense layer, we are using one more dense layer to classify the output (whether a person is detected or not) for the image.
Then we are compiling the model using the Adam optimizer and loss as binary_crossEntropy (as the output class has only two labels, person present or not present).
Finally, we are training our model in batches of 50 and 10 iterations on the training dataset and evaluating its performance on the testing dataset.

Improved Neural architecture model -

We have improved our model by increasing the number of epochs and decreasing the batch size. It is giving accuracy even higher than the pretrained **VGG16** model.
Best Accuracy at 16th epoch -
Train accuracy - 90.73 %
Test accuracy - 82.80 %

Understanding the model-

From the accuracy table of the model, we can understand that when our neural network model is very deep or very shallow, it leads to overfitting.
And when we increase the number of parameters, generally, accuracy increases because it adjusts the weight of layers based on the results.