# DL Model Optimization for Lightweight Visual Wake Words Task


- Author: Mruganshi Gohel (B20CS014) , Shubham Ghelani (B20EE019)


## How to run code:

- Download the .ipynb file.
- Login into google colab (or Jupyter Notebook if you have it)

(preferred to run on google colab)

1. **If you are using google colab**

- import .ipynb file into colab (file→open)
- Firstly download the dataset (submitted as a zip on GC)
- import the dataset(zip) on the drive
- import dataset zip file by changing the cell-3 code in the .ipynb file

```
!unzip zip-location-on-drive
```

and run all cells

2. if u**sing a Jupyter Notebook**

- just import the dataset from your PC no need to import it on the drive.
- run all cells

## Required Libraries to Download

1. Numpy

```
pip install numpy
```

2. pandas

```
pip install pandas
```

3. matplotlib (for visualization)

```
pip install matplotlib
```

4. tenserflow

```
pip install tenserflow
```

5. PIL (for image)

```
pip install pillow
```

## Problem Description

The Lightweight Visual Wake Words (LVWW) task involves the development of an artificial intelligence (AI) model that can automatically identify whether a person is present or not in the camera input of mobile phones or microcontrollers. This is a useful

application for a range of use cases such as security systems, automated attendance systems, and monitoring applications.

The LVWW task is particularly challenging as the model needs to be optimized for deployment on resource-constrained devices with limited computing power and memory. Therefore, the model must be lightweight and efficient, while maintaining high accuracy in detecting people.

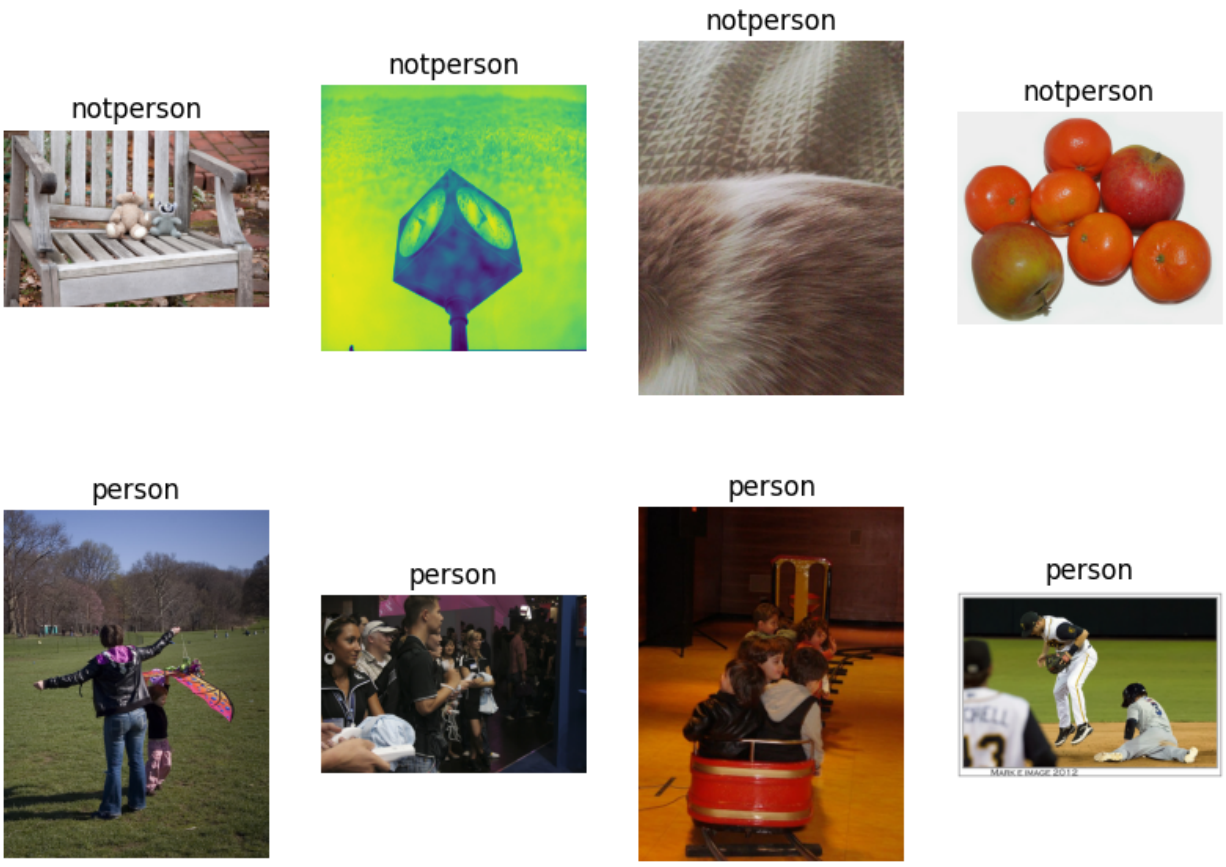
## Approach

In this project, we need to build a model which can automatically detect whether a person is present or not in given image. The input image would be mostly from camera of the phone. Based upon the features of the image and its configuration, we need deep learning models to detect whether the person is present or not in the image. The plan of implementation was as follows:

- Stabilizing and reducing the dataset
- Preprocessing the dataset
- Building the Model, Training and Testing the Model
- Model Robustness Check
- Analyzing the Neural Network Design by changing the size of the neural network
- Observing and concluding the final results

## Dataset Description

The dataset is derived from COCO2017 and reduced to ~150MB using the Python script, written in report and submitted along with codes.

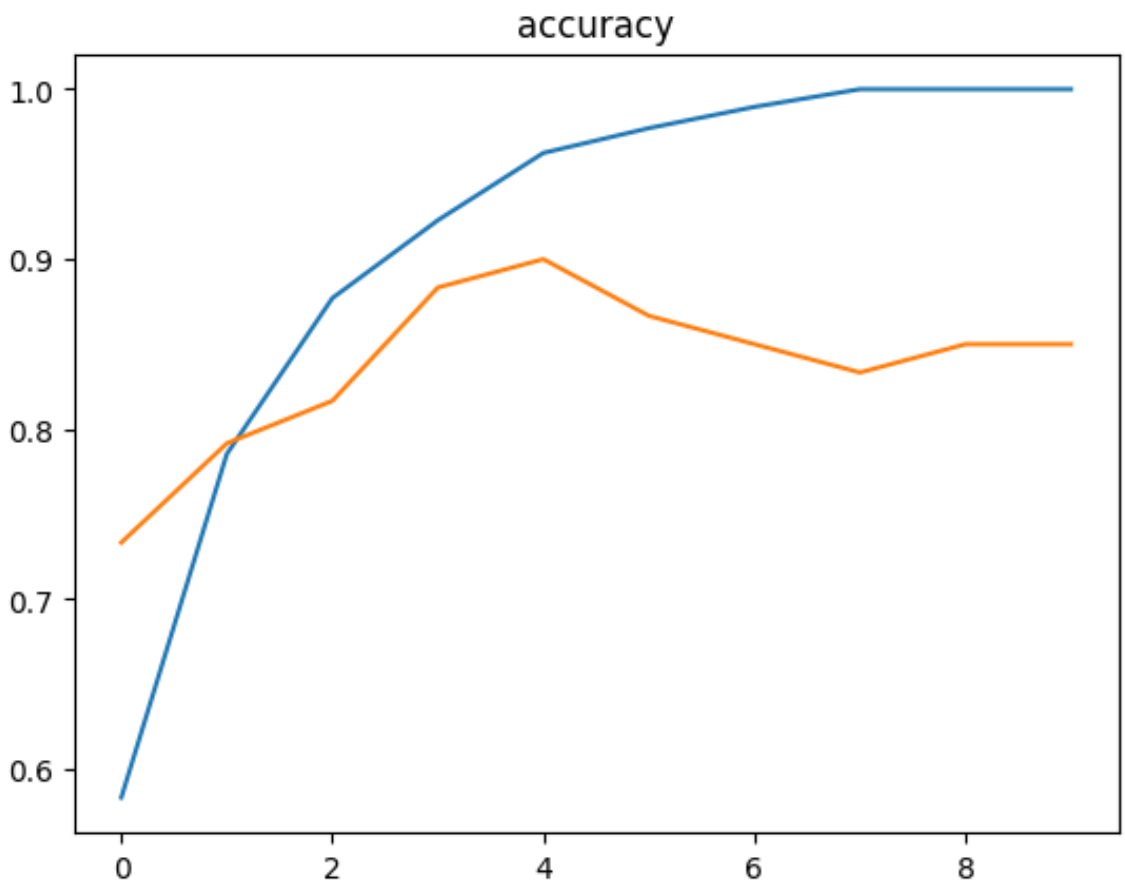


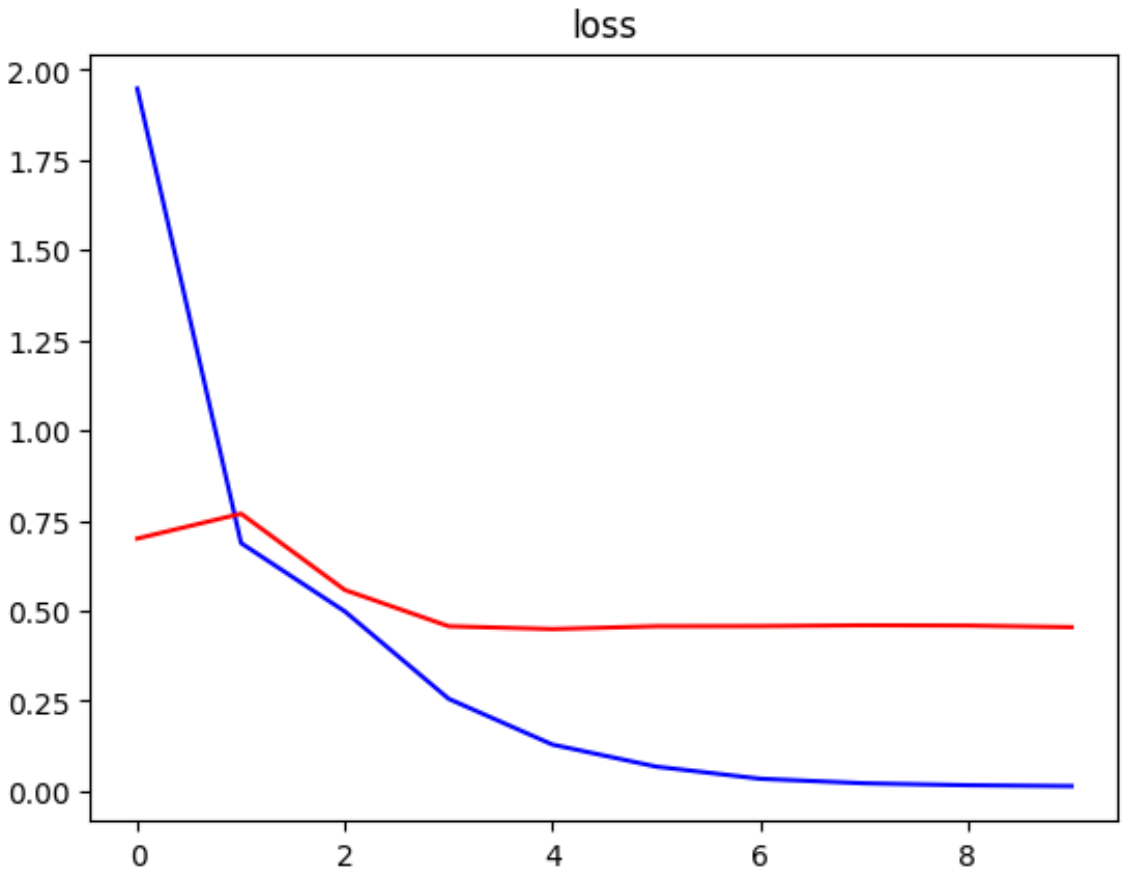

## Experimental results

Results of Model

| | Accuracy |
|---|---|
| Training Data | 97.29% |
| Testing Data | 90% |

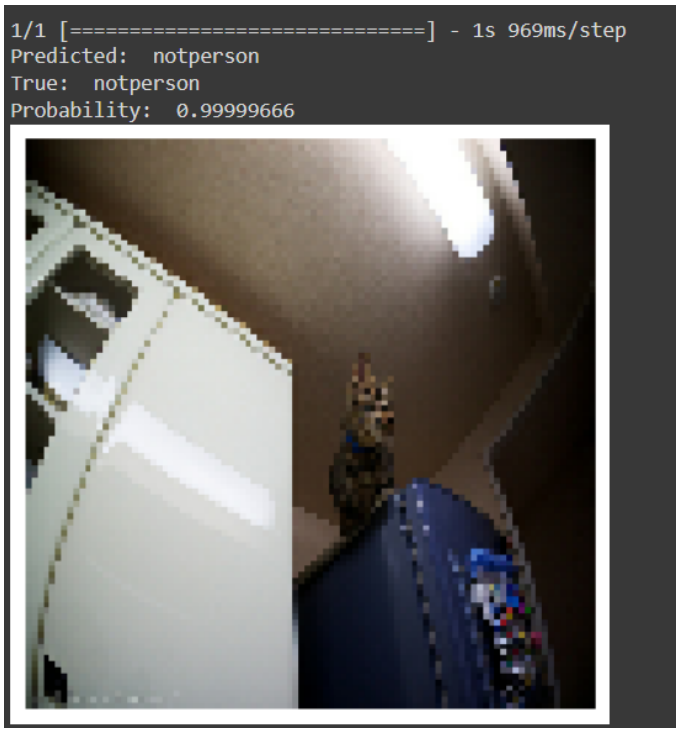
The following images shows the trend of loss function,  Accuracy function


accuracy
1.0
0.9
0.8
0.7
0.6
0
2
4
6
8




## Prediction

a random image from the validation set is taken and checked whether the data is properly and with the correct model trained or not.




## References

https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c

https://towardsdatascience.com/visualizing-and-preprocessing-image-dataset-e3ad574f7be6

https://analyticsindiamag.com/a-deep-dive-into-image-data-preprocessing-by-tensorflow/

https://www.quora.com/How-are-smaller-sized-deep-learning-models-i-e-MobileNetv2-EfficientNet-B1-NasNet-Mobile-advantageous-over-larger-sized-models-i-e-VGG-16

https://github.com/Mxbonn/visualwakewords