MASK DETECTION

FINAL PROJECT

GROUP - 7

Ankush Gupta
Ronica Gupta
Tanjin Sharma
Abdus Khan

Data and Analytics Goals

Data

- We collected our dataset from Google Image search.
- Our dataset includes 2 types of images, namely images of people with a mask and images of people without a mask.
- The images with mask are labeled as 1 and the ones without are labeled as 0.
- Training records : 2930
- Validation records: 734

Goals

- The objective of the project is to identify if a person is wearing a mask or not. Training a machine learning model for the same includes the following steps:
    - Data Collection and Labelling
    - Data Cleaning and Preprocessing
    - Model Training and Testing

Data Collection

Data Collection

- We scraped data using google Image search.
- Additionally, we also used open source data for mask and unmasked images.

Data Preprocessing

Data Preprocessing

Our data preprocessing pipeline included the following steps:

- Reading Images as binaries from AWS S3
- Adding a label column, 1 for images with mask and 0 for images without mask
  - *Time to read masked images and add label: 8 minutes*
  - *Time to read non-masked images and add label: 1 minute 58.32 seconds*
- Filtering out invalid images based on length of the image arrays in numpy
  - *Time to filter out valid images: 2 minutes 61 seconds*
- Storing a data frame containing a binary array and label corresponding to each image in MongoDB Atlas

MongoDB and Databricks Cluster settings

Cluster Settings- Data Preprocessing

MongoDB Cluster Settings for Data Preprocessing :

- Cluster Tier: M-30
- Type: Sharded Cluster, 2 Shards
- Version: 4.4.12
- Region: AWS/Oregon (us-west-2)

DataBricks Cluster Settings for Data Preprocessing :

- Cluster Mode: Standard
- Databricks Runtime Version: 10.3
- Worker Type: i3.xlarge, 30.5Gb Memory, 4 cores
- Min Workers:2
- Max Workers: 8

Cluster Settings- Deep Learning

MongoDB Cluster Settings for Machine Learning Methods :

- Cluster Tier: M-30
- Sharding: 3 nodes
- Type: Replica Set - 3 nodes
- Version: 4.4.13
- Region: AWS/Oregon (us-west-2)

DataBricks Cluster Settings for Machine Learning Methods :

- Cluster Mode: Standard
- Databricks Runtime Version: 10.3
- Worker Type: g4dn.xlarge , 16GB Memory, 1 GPU
- Min Workers: 1
- Max Workers: 3

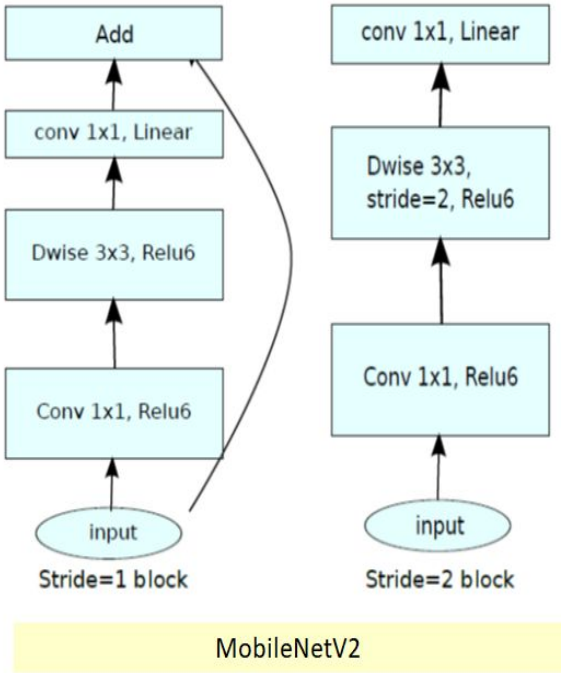
Machine Learning Method and Outcome

Deep Learning Models Trained

- MobileNetV2
- MobileNetV3(small)
- mnasNet 1_0
- efficientnet_b0

MobileNetV2

- Number of layers : 53
- Number of parameters : 3.4 M
- Best Validation Loss : 0.6905
- Best Validation Accuracy : 0.56
- Time : 2300 seconds



MobileNetV3(small)

- Number of layers: 16
- Number of parameters: 5.4 M
- Best Validation Loss : 0.7037
- Best Validation Accuracy : 0.53
- Time : 320 seconds

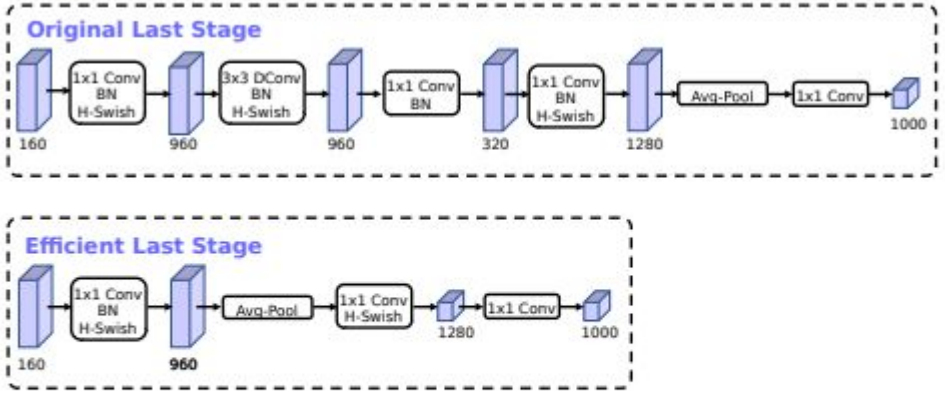
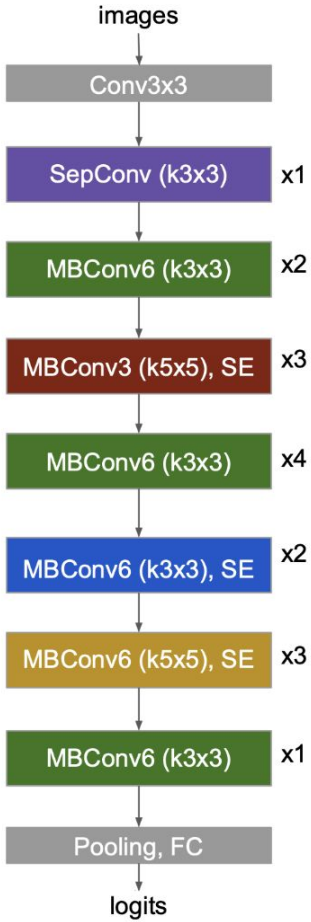


Figure 5. Comparison of original last stage and efficient last stage. This more efficient last stage is able to drop three expensive layers at the end of the network at no loss of accuracy.

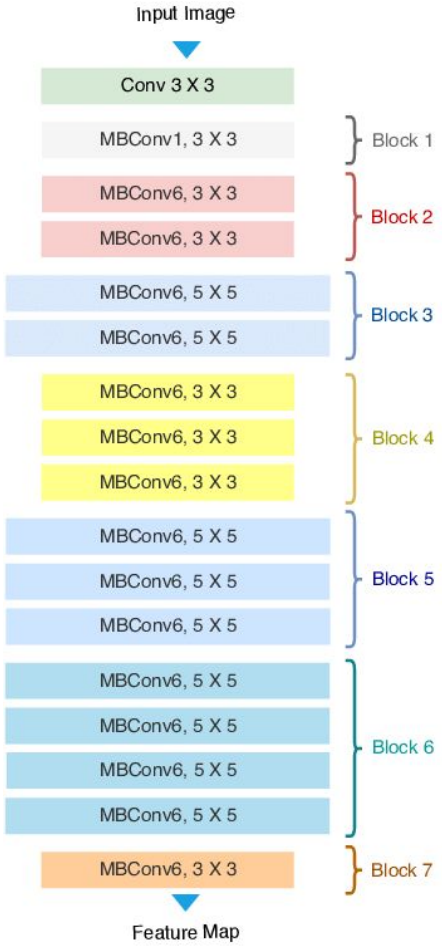
mnasNet 1_0

- Number of layers: 17
- Number of parameters: 4.3 M
- Best Validation Loss : 0.6816
- Best Validation Accuracy : 0.58
- Time : 1900 seconds



efficientnet_b0

- Number of layers: 237
- Number of parameters: 5.2 M
- Best Validation Loss : 0.6941
- Best Validation Accuracy : 0.54
- Time : 1210 seconds



Lessons Learnt

Lessons Learnt

- Reading data directly from S3 is quite slow, whereas storing and reading pre-processed data from MongoDB is much faster.
- Benefits of caching and parquet files for efficiency.
- Image pre-processing using PySpark.
- Different deep learning architectures for Image Classification.
- Importance of clean data for training models.
- Dealing with Image byte arrays.
- Identifying dirty data (images) using different techniques.

Conclusion

Conclusion and Improvements

- Model mnasNet 1_0 gave us the best validation loss of 0.6816.
- Distributed Spark PreProcessing helped to convert binaries to arrays and vice-versa much faster.
- The scraped data from web was dirty and therefore, our models were not very accurate.
- Also, some of the images had a lot of background information, which affected the performance of our model.

Improvements:

- Using face-centered images with minimal background can boost model performance.
- Hyper-parameter fine tuning can also help in improving performance.
- Clusters with more memory can help us train larger models with more layers and parameters.

Thank You Diane!