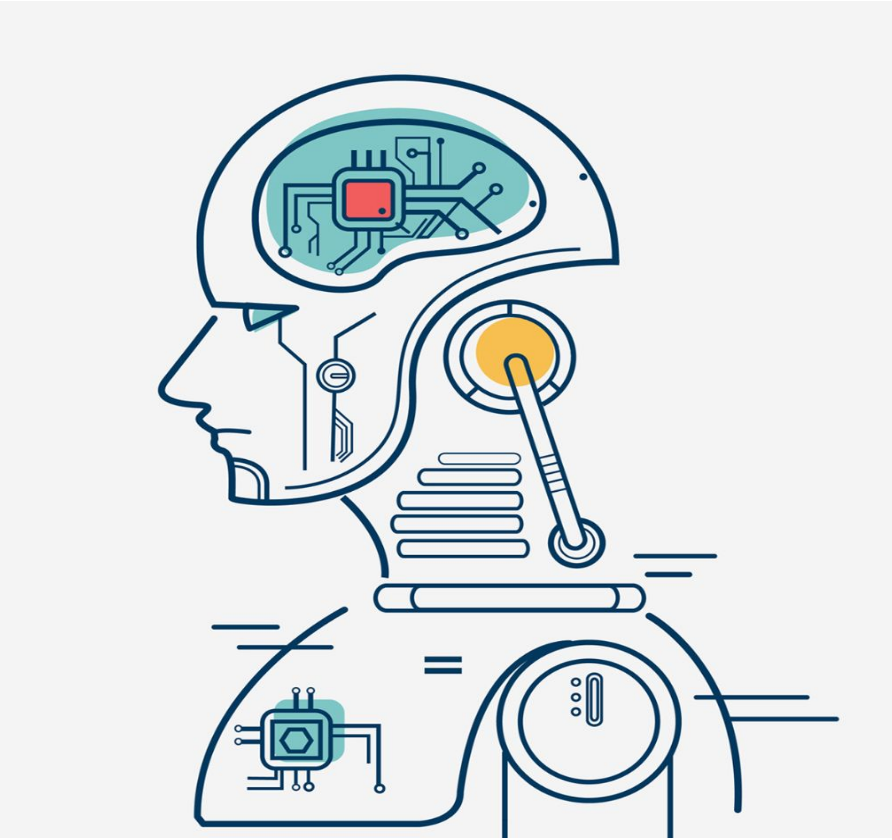
Face Mask Detection Using Image Classification

Group Members: Benjamin Alpert, Parisa Suchdev, Shaurya Swami, Zunyi Liao

Summary

- Introduction
- Data
- Approaches
- Results



Introduction

Problem

- Coronavirus has become a worldwide health pandemic, causing 4.55 million deaths worldwide
- Due to the virus constantly mutating and forming new variants, vaccines that were effective on old variants are not necessarily well-suited to combat the ever-changing new ones.
- One of the best preventative methods is to wear filtered masks that fully cover the mouth and nose:
    - Prevents the virus from entering our bodies (spread through breathing, sneezing, coughing, etc.)
    - Aids in stopping onward transmission of the virus since it is contagious

Applications

- Entry Systems / Door Unlocking
  - Universities
  - Hospitals
  - Airports
- Mask Enforcement
  - Planes
  - Busses
  - Trains
- With refinement, could implement in security cameras

Classification

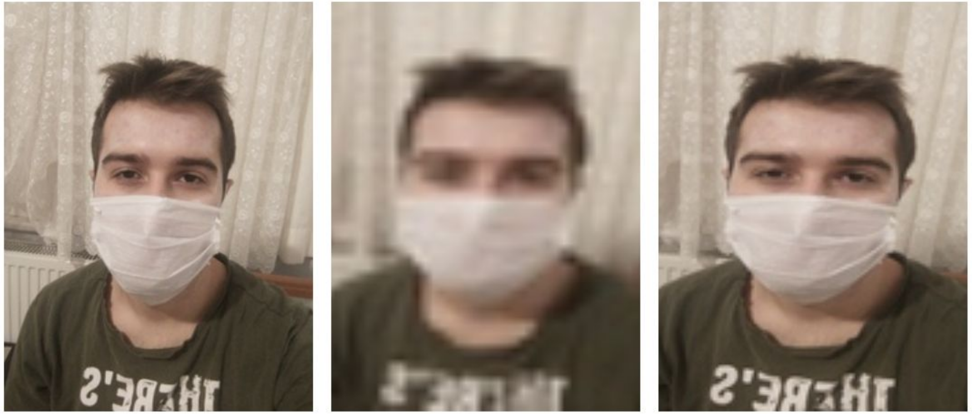
Class 1 Wearing mask correctly
Class 2 Does not cover nose
Class 3 Does not cover nose and mouth
Class 4 There is no mask on the face

One Hot Encoding
High accuracy on class 1
Lower accuracies on class 2, 3, and 4

Data

Data Preprocessing

- Used 170GB of full 500GB
- Size in GB:
  - 170 to 2.6GB
  - 170 to 5.7GB
- Size in Pixels:
  - 1944 x 2592 to 50 x 50
  - 1944 x 2592 to 150 x 150
- Use of GPU and RAM
- ~80,000 images



Data Augmentation

To enhance our model, we randomly flipped the images, cropped the images, rotated the images, zoomed the images, and contrasted the images during the training.

- Random Contrast
- Random Flip
- Random Zoom
- Random Rotation
- Random Cropping

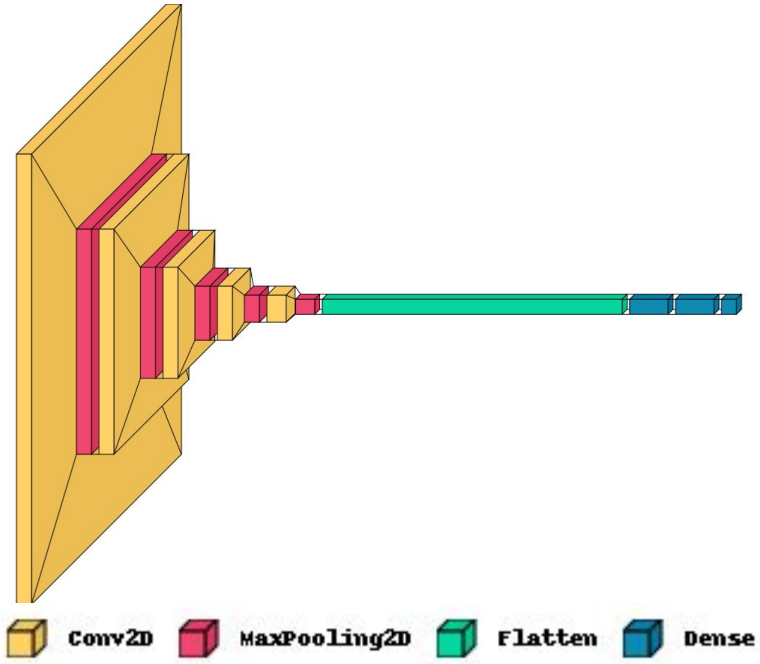
Data Regularization

We rescaled all the feature values from 0-255 to 0-1.

Approaches

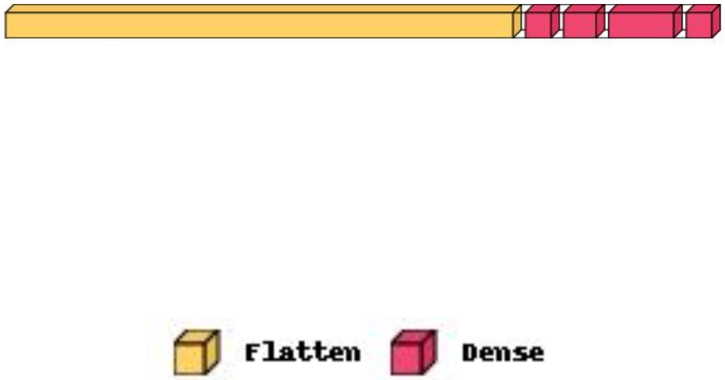
CNN Model

1. Data Augmentation (discussed earlier)
2. Rescale/Regularize from 0-255 to 0-1
3. 5 Conv2D layers with 16, 32, 64, 128, and 256 filters all using the 'relu' as the activation and MaxPooling2D layers in after each Conv2D layer
4. Flattening layer
5. 2 fully connected layers both using 512 filters with 'relu' as the activation function
6. 1 final fully connected layer using 4 filters; 1 for each class



MLP Model

1. Data Augmentation (discussed earlier)
2. Rescale/Regularize from 0-255 to 0-1
3. 3 fully connected layers with a size of 128, 256, and 512 all using the ‘relu’ as the activation
4. 1 final fully connected layer with a size of 4; 1 for each class



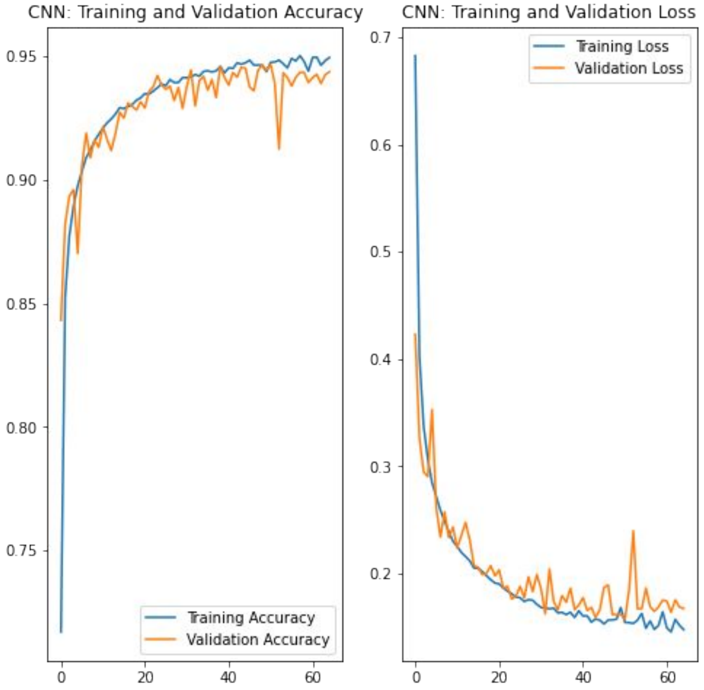
Pseudocode

1. Import tensorflow
2. Bring our training data into memory and split into validation and training subsets using 80% for training and 20% for validation
3. Shuffle the dataset
4. Data Augmentation & Regularization
5. Create the CNN and MLP models
6. Compile the models
7. Train the models
8. Plot training and validation epochs versus accuracy
9. Plot training and validation epochs versus loss

Results

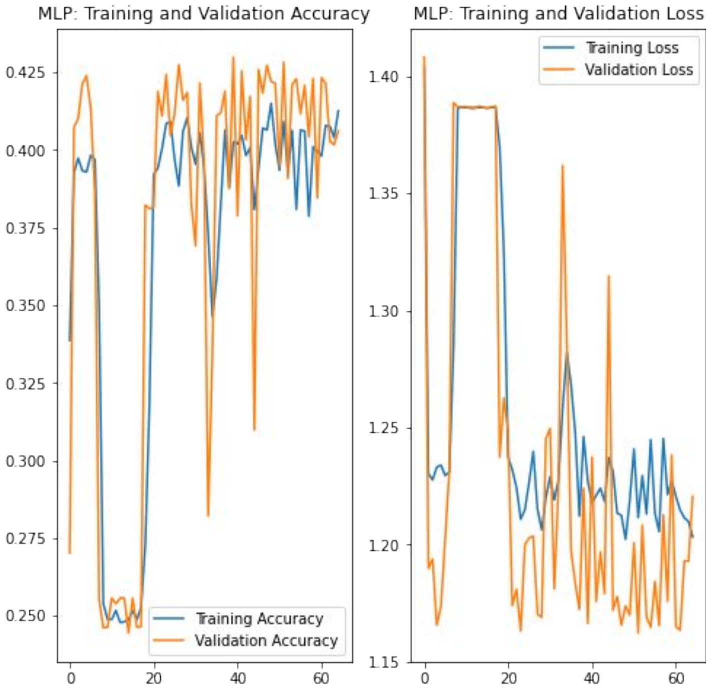
CNN Results

# Epochs	65
Accuracy	0.9495
Loss	0.1475
Validation Accuracy	0.9437
Validation Loss	0.1671



MLP Results

# Epochs	65
Accuracy	0.4126
Loss	1.2033
Validation Accuracy	0.4062
Validation Loss	1.2206



Live Demonstration

Resources and Links

- The datasets for this project can be found at
    - https://www.kaggle.com/tapakah68/medical-masks-part1
    - https://www.kaggle.com/tapakah68/medical-masks-part2
- Our code for this project can be found at
    - https://github.com/BenjaminAlpert/cs254a-final-project
- Webapp can be found at:
    - https://fs.a0-0.com/cs254a-final-project/demo/ (currently does not work on iOS or Safari. works in Chrome)
- Tensorflow, Libraries and Tools
    - https://www.tensorflow.org/tutorials/images/classification
    - https://github.com/tensorflow/tfjs
    - https://github.com/paulgavrikov/visualkeras
    - https://www.tensorflow.org/api_docs/python
    - https://imagemagick.org/

Thank You!