Data Analysis and Visualization Lab HW2 – Face Mask Recognition

Alexander Gofman 312884323 Tal Kaspani 204528673

1. Exploratory Data Analysis
   a. The Dataset consists of roughly 18,000 train images and 6,000 test images of people faces, some of which are wearing different forms of face masks and covers.
   All images containing a person with a covered face are labeled "1" meaning the person in the image is wearing a mask, it's important to note that mask is not the only form of "face cover" which is recognized as a positively labeled image.
   Examples of training data images (original size):



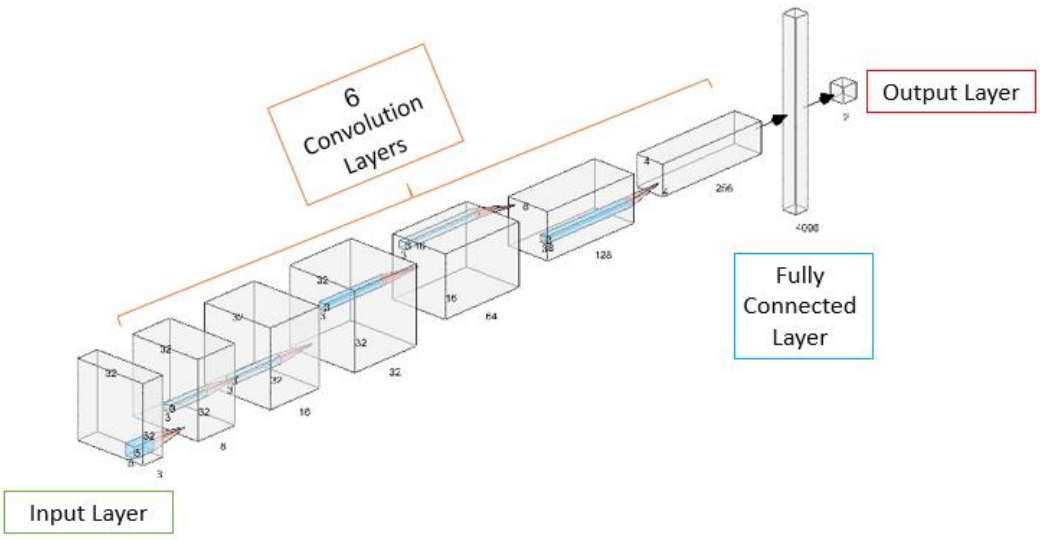

   Examples of test data images (original size):


   b. Key insights from the data:
      - The images are of different size, meaning we will have to resize them before handing them over to our network.
      - Unlike basic open source data sets for deep learning such as Fashion-Mnist, our images are taken in different orientations, angles and directions meaning we don't have to add any form of random rotation and crop for them to improve our network.

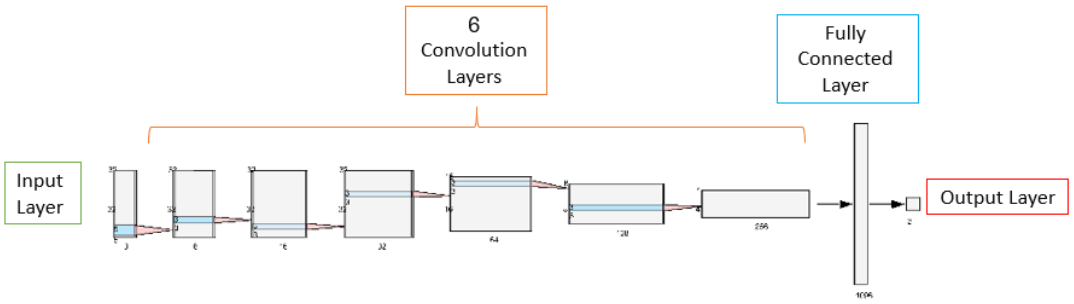
2. Network Experiments – Net #1
    a. Data Loading and Preprocess:
    To use the Pytorch package to build our network we had to build a custom data loader for our dataset, the resulting loader output is 32X32 size images.
    As mentioned above, we chose not to use any form of data augmentation because of the nature of the images in the dataset.
    b. Architecture:
    Our network consists of **6** convolution layers, and **1** fully connected layer:

Parameters	Conv #1	Conv #2	Conv #3	Conv #4	Conv #5	Conv #6	Fully connected
Input	3	8	16	32	64	128	4 * 4 * 256
Output	8	16	32	64	128	256	2
Activation Function	Relu	Relu	Relu	Relu	Relu	Relu	-
Pooling	-	-	-	Max	Max	Max	-
Normalization	-	-	-	Batch	-	Batch	-
Convolution Filter	5	3	3	3	3	3	-
Padding	2	1	1	1	1	1	-

Overview:



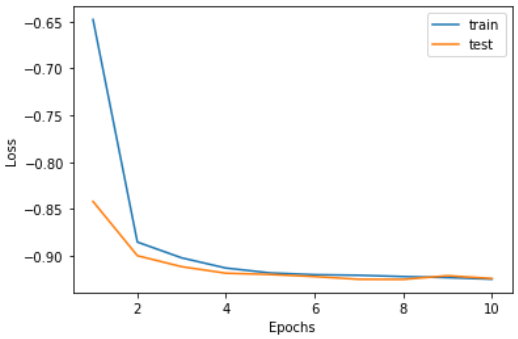
Sideview:

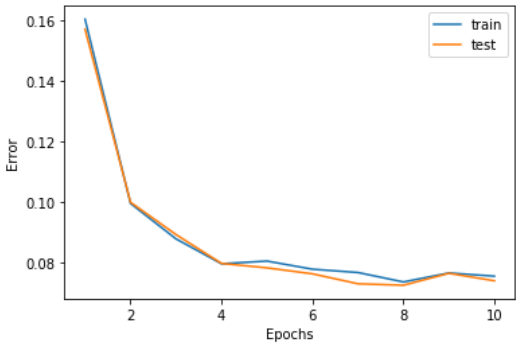


c. Loss Function – We choose the Negative Log Likelihood Loss as we are dealing with a classification task.
d. Optimizer – We choose the Adam optimizer which is an extension of the classical SGD algorithm as it introduces individual learning rate for each parameter and converges faster.
e. Regularization – We choose to use weight decay as our regularization technique instead of the classical dropout.
   Other Parameters:

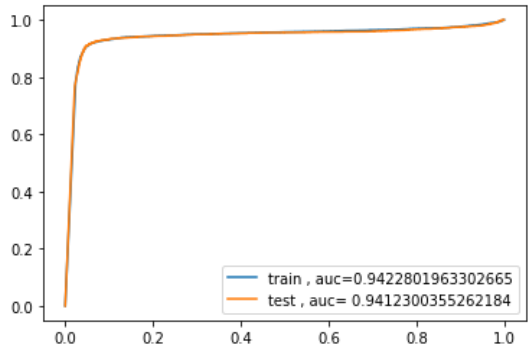
Hyper Parameters	Value
Epoch Number	10
Batch Size	512
Learning Rate	0.001
Weight decay	0.001
Total number of weights	402770

f. Loss and Error graphs:

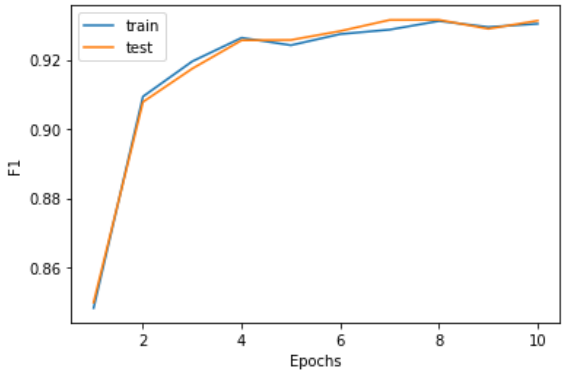




g. ROC AUC graphs:



h. F1 graph (Best score is 0.928):

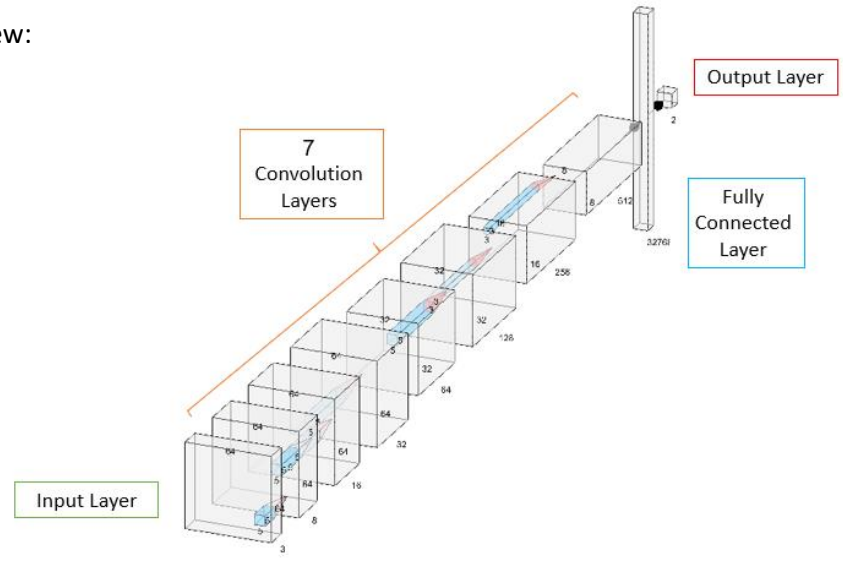


i. Key Insights:
   - Increasing the number of training epochs is necessary to achieve high performance.
   - Increasing the size of the first convolution filter to 5X5 proved very useful to our model.
   - Our model barley used a tenth of the total number of available parameters (400K from 5 Mil) and might be too simple.
   - We might want to use dropout in our next attempt as another form of regularization while we are building more complex networks.

3. Network Experiments – Net #2
   a. Data Loading and preprocess:
      After examining the data closely, we noticed that most images are bigger than 32X32 pixels and compressing them to that size might harm our model performance, hence, we decided that our second networks should use 64X64 Images (we did not choose 128X128 because half of the images on the training set are smaller than that size and extensive stretching might distort the images).
      As mentioned above, in this attempt we also chose not to use any form of data augmentation.
   b. Architecture:
      Apart from adding another convolution layer and increasing the size of filters from 3 to 5, we will use similar architecture as our first attempt:

Parameters	Conv #1	Conv #2	Conv #3		Conv #4	Conv #5	Conv #6	Conv #7		Fully connected
Input	3	8	16		32	64	128	256		8 * 8 * 512
Output	8	16	32		64	128	256	512		2
Activation Function	Relu	Relu	Relu		Relu	Relu	Relu			-
Pooling	-	-	-	Drop out	Max	-	Max	Max	Drop out	-
Normalization	-	-	-		Batch	-	-	Batch		-
Convolution Filter	5	5	5		5	5	3	3		-
Padding	2	2	2		2	2	1	1		-

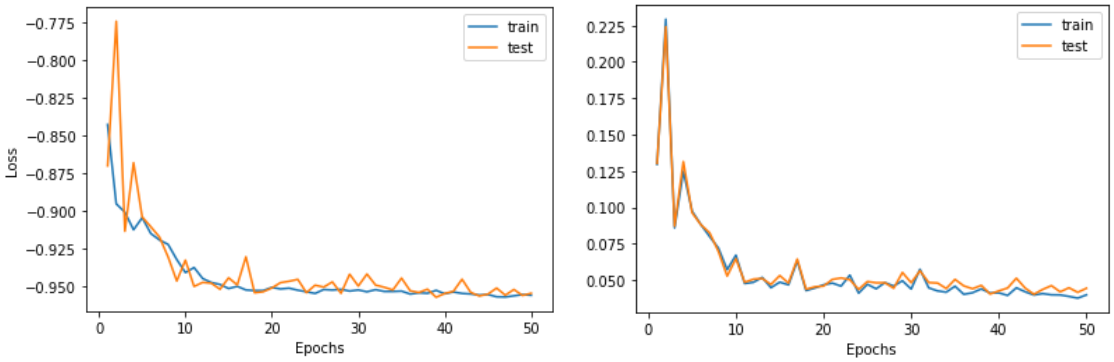
Overview:



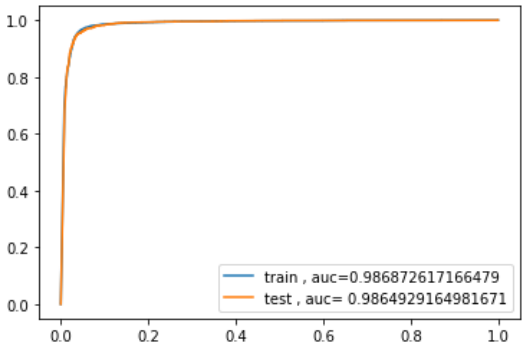
c. Loss Function – We choose the NLLLoss.
d. Optimizer – We choose the Adam.
e. Regularization – In addition to weight decay we inserted two additional dropout layers, one before the pooling stages(after conv 3) and another one before the last fully connected layer(conv 7) .
Other Parameters:

Hyper Parameters	Value
Epoch Number	50
Batch Size	128
Learning Rate	0.001
Weight decay	0.001
Dropout	0.2
Total number of weights	1814866

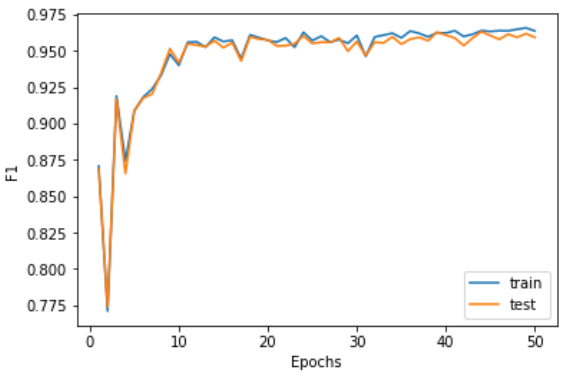
f. Loss and Error graphs:



g. ROC AUC graph:



h. F1 graph (Best score is 0.963):



i. Key Insights:
   - Increasing the number of training epochs proved vital to achieve high performance.
   - Complex models can outperform simple ones.