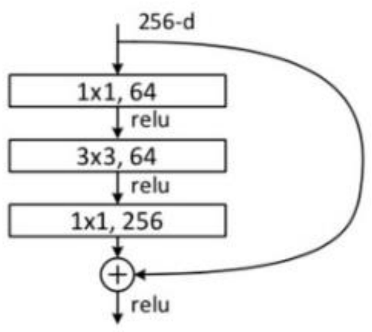
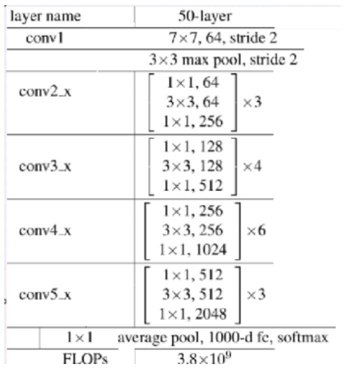
Mask detection report

2019023654 전상우
2019076508 서호협

1. Resnet-50 architecture

ResNet-50 addresses the vanishing gradient problem especially when encountered training very deep networks. The key idea in ResNet-50 is the introduction of residual blocks which allow the network to learn residual functions with reference to the layer inputs.





layer name	50-layer
conv1	7×7, 64, stride 2
	3×3 max pool, stride 2
conv2_x	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5_x	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
1×1	average pool, 1000-d fc, softmax
FLOPs	3.8×10^9

- **Structure of ResNet-50**

-Input Layer: Input image size is 224×224×3

- **Initial Convolution and Max-Pooling:**

-Conv1: 7×7 convolution, 64 filters, stride 2, followed by batch normalization and ReLU activation.
-Max-Pool: 3×3 max pooling, stride 2.

- **Residual Blocks:**

-ResNet-50 contains a series of residual blocks. Each consists of three convolutional layers:
1. 1×1 convolution (bottleneck layer) reducing the number of filters.
(It was the reason we could see some mentions about bottleneck!)
2. 3×3 convolution.
3. 1×1 convolution increasing the number of filters back to the original number.

The residual blocks are organized into 4 stages:
Stage 2: 1 residual block, 3 layers each with 64 filters.
Stage 3: 4 residual blocks, each with 128 filters.
Stage 4: 6 residual blocks, each with 256 filters.
Stage 5: 3 residual blocks, each with 512 filters.
-Each stage has a convolutional layer at the beginning that down samples the spatial dimensions and doubles the number of filters from the previous stage (except for Stage 2).

- **Average Pooling and Fully Connected Layer:**

-Average-Pool: Global average pooling over the 7×7 feature map.
-FC Layer: Fully connected layer with 1000 neurons (for 1000 classes in ImageNet).
-Softmax Layer: Produces the final class probabilities.

- **Residual Block Structure**

Each residual block has a skip connection that bypasses the non-linear transformations with an identity mapping. This identity mapping is either a direct shortcut or a projection shortcut to match dimensions when the input and output dimensions differ.
-Bottleneck Residual Block (used in ResNet-50 and deeper):

-1×1 convolution, reducing the dimension.

-Batch normalization.

-ReLU activation.

-3×3 convolution.

-Batch normalization.

-ReLU activation.

-1×1 convolution, restoring the dimension.

-Batch normalization.

-Addition with the input (skip connection).

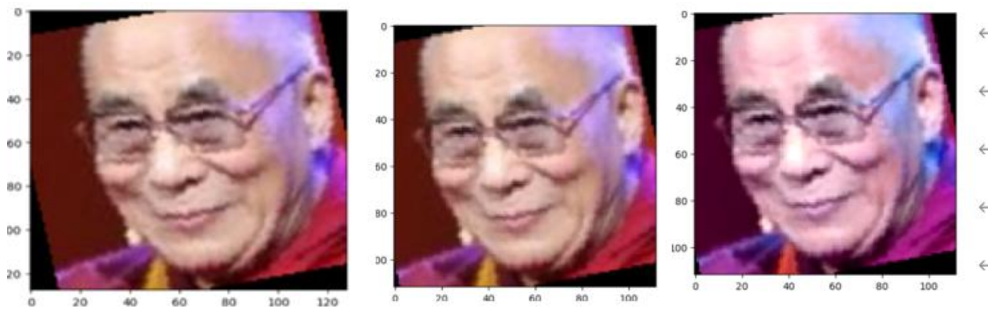
-ReLU activation.

2. MaskTheFace

MaskTheFace library makes samples of augmented images making mask to normal face images. Since there are various types of masks such as KF94 or medical or sports masks, various types of colors and shape were randomly applied when doing data augmentation.

Using above, we could add more argumented data. (rotation, flip, crop, color, brightness…)

Random rotation (15) **Random resized crop (x0.8~x1.2)** **Transform color**

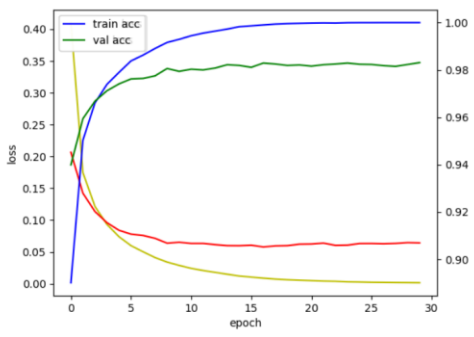


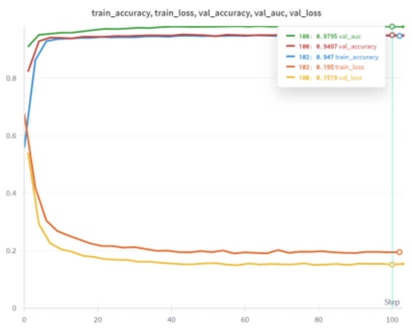
Therfore, we could gather enough masked data set from non-masked data set.

3. Plots

- ROC, accuracy, loss

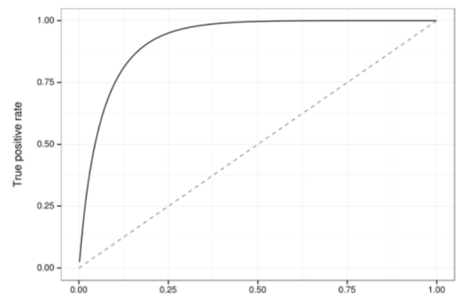
Normal graph **Our graph**

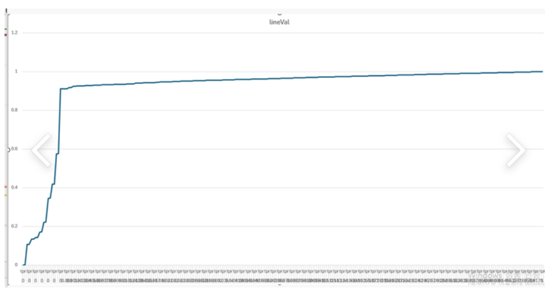




-val_roc curve (table >> curve in Excel)

ideal AUC **our AUC**



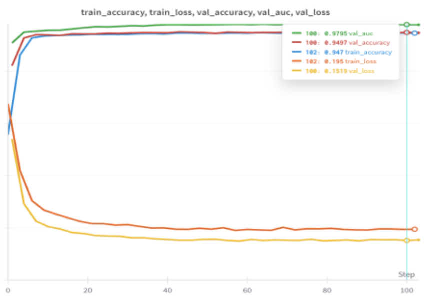


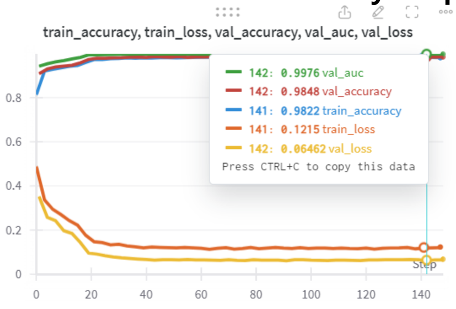
-Both val, train accuracy are saturated about 94~95%, similar to Ideal.
-Each val, train loss saturated about 0.15 / 0.19, showing enough decrease.
-val_auc saturated about 0.98, very close to 1.
-val_roc saturated to 1 meaning very few of false response. (for data set)

4. Discussion

-Not using early stopping

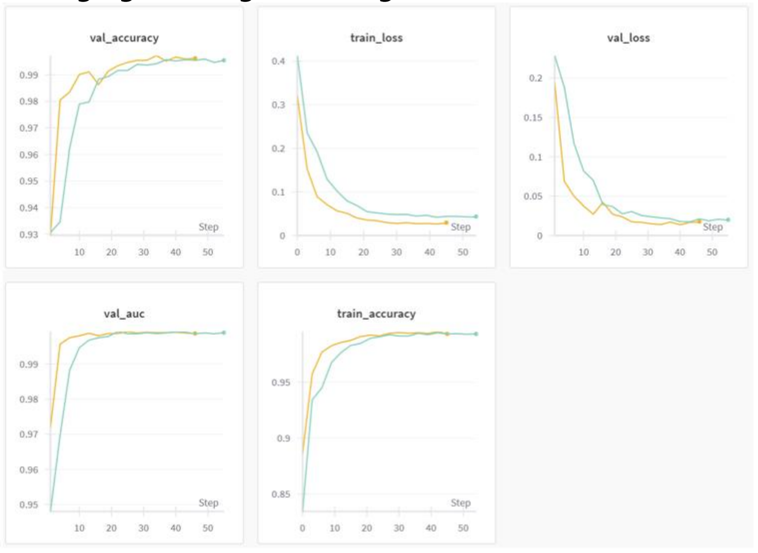
Original training **Without Early stopping**





At original training, at epoch 34, train finished, and its accuracy was about 95%. It trained for 1h 25m.
In contrast, if we didn't apply early stopping, for epoch 19 to 50, training accuracy saturated about 98% and it trained for 2h 09m, costed time more than 30%.
Although early stopping did not figure out the overfitting problem, we were able to find another advantage of early stopping in that it can reduce the expected result to nearly half the time and save computing resources.

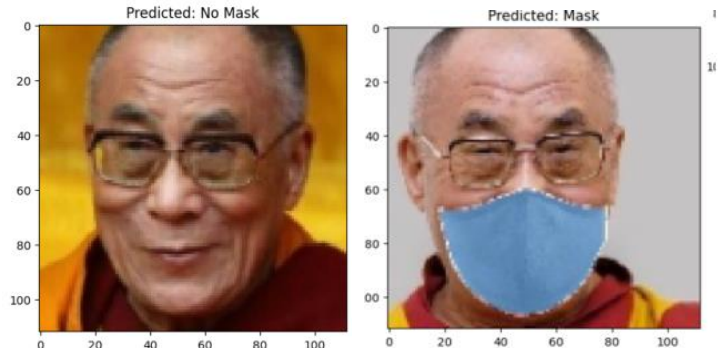
-Changing learning rate (Original (0.001) ▶ lr = 0.05)



As a result of raising the learning rate, the accuracy converged faster, and loss was little upper. And when the learning rate was high, learning ended quickly due to early stopping.

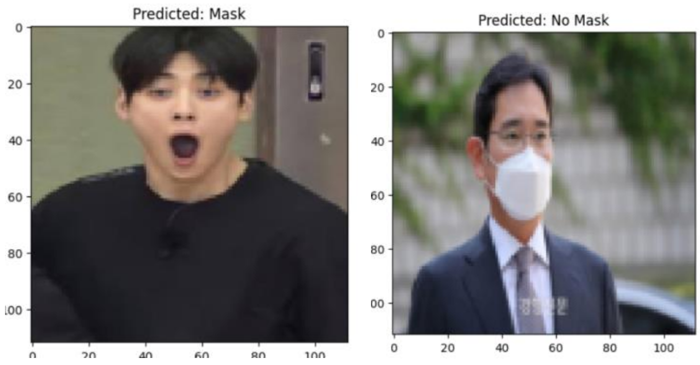
-limitation of current model

I think the learning went well because it showed good shapes and high accuracy during training. Compared to the high accuracy, the percentage of correct answers in the test set was just 55%, so when I checked with the pictures in the training set, I found that they all got it right. For this reason, our model determined that overfitting occurred.



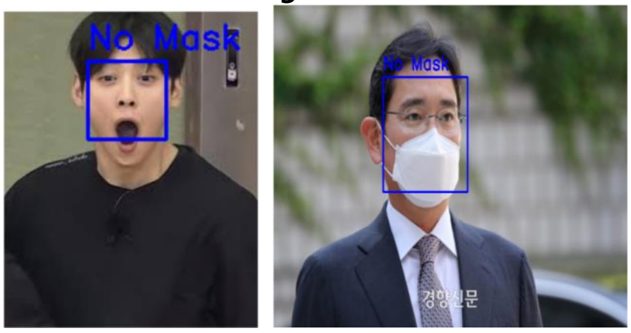
To figure out this problem, we tried various attempts, such as increasing the dropout ratio and lowering the learning rate... etc, but not all of the test images were correct.

Result of our model



As a next best option, we compared the results with mask recognition in the face recognition library, which resulted a similar with ours. It was also wrong for some pictures.

Result of face recognition

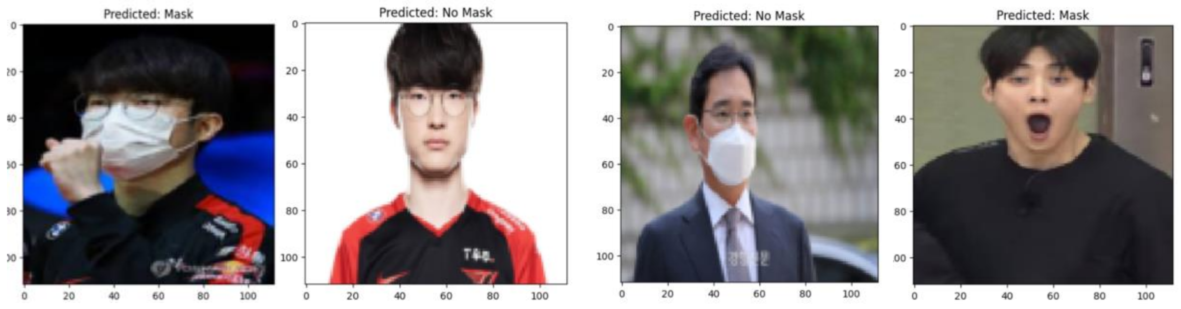


Face_recognition library was also not perfect to some test images.

As a result of observing the learning data, the faces were clearly in the middle of the image, but the photos we tested seemed to have reduced accuracy in face detection because there were various objects as well as human faces.

5. Qualitative evaluation

Mask O >> O **Mask X >> X** **Mask O >> X** **Mask X >> O**

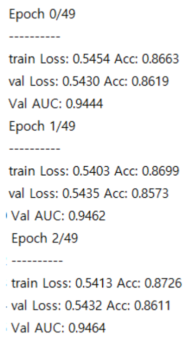


Total accuracy of test set was 55%. (I think black hole considered as mask…)

6. Consideration

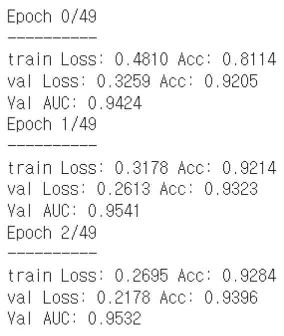
Backpropagation issue

-initial model



```
Epoch 0/49
----------
train Loss: 0.5454 Acc: 0.8663
val Loss: 0.5430 Acc: 0.8619
Val AUC: 0.9444
Epoch 1/49
----------
train Loss: 0.5403 Acc: 0.8699
val Loss: 0.5435 Acc: 0.8573
Val AUC: 0.9462
Epoch 2/49
----------
train Loss: 0.5413 Acc: 0.8726
val Loss: 0.5432 Acc: 0.8611
Val AUC: 0.9464
```

-final model



```
Epoch 0/49
----------
train Loss: 0.4810 Acc: 0.8114
val Loss: 0.3259 Acc: 0.9205
Val AUC: 0.9424
Epoch 1/49
----------
train Loss: 0.3178 Acc: 0.9214
val Loss: 0.2613 Acc: 0.9323
Val AUC: 0.9541
Epoch 2/49
----------
train Loss: 0.2695 Acc: 0.9284
val Loss: 0.2178 Acc: 0.9396
Val AUC: 0.9532
```

The initially generated model showed consistent loss and accuracy and was not trained.

We focused on the consistency of the graph, concluded backpropagation did not occur, which found that gradient vanishing was a problem. As we learned in the lecture, ReLU was used as an active function and batch normalization was applied to solve this problem. Residual connection was passed on because it was applied to Resnet-50 as a default.

As a result, a good type of learning graph came out as it is now, and the accuracy and loss were newly updated for each epoch.

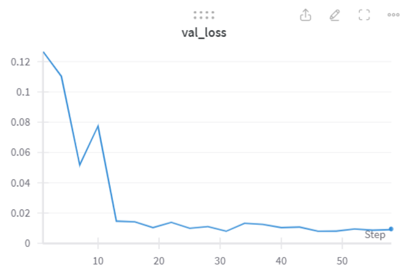
Overfitting isuue

While looking for various data to solve overfitting, I applied many techniques such as dropout and changing learning rate to the current model. Although it did not solve the overfitting problem, I was able to experience other advantages of these techniques.

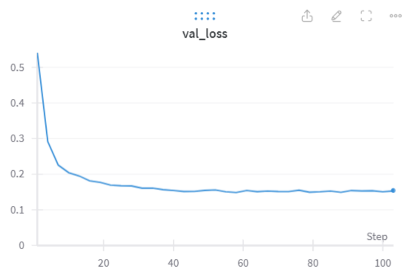
Dropout was able to reduce the learning time as well as overfitting and lowering the learning layer. The initial model took about 3-4 hours for learning, but only about 2 hours by applying dropout.

Changing learning rate helped to alleviate the sound that occurs in the valve loss. Of course, it still appears to move in zigzags, but as you can see in the picture, it solved the phenomenon of splashing in a specific section very well.

Past learning rate



Present learning rate



I think the most necessary competency in creating artificial intelligence is to understand the model training process and find hyperparameters suitable for that model. In addition, I thought that increasing accuracy in the learning process was the top priority, but I could personally feel how fatal overfitting was to the outcome.