I applied transfer learning to my capstone project. After I tried a few models, such as VGG16, Xception, and ResNet50, my mentor recommended I should use VGG16.

I used a pretrained VGG16 model and replaced the top layer with my own. I re-trained the model three times (10, 20, and 50 epochs) and checked the accuracy.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test | | Validation | |
| Recall | Precision | Recall | Precision |
| Epoch 10 | 1 | 1 | 0.765 | 0.591 |
| Epoch 20 | 1 | 1 | 0.738 | 0.617 |
| Epoch 50 | 0.695 | 0.658 | 0.777 | 0.594 |

To improve the accuracy of the validation dataset, I explored the following fine-tuning options with 50 epochs:

1. Use 'relu' for the first dense layer

2. Use a dropout 0.2 in the dense layer

2-1. Use a dropout 0.5 in the dense layer

3. Use 2 dense layers

4. Combine 1&2

5. Combine 2&3

6. Combine 1&3

7. Combine 1,2,&3

8. Change learning rate to 0.001

9. Use ‘relu’ for two dense layers and ‘softmax’ for the last dense

10. Train the last VGG16 layer

11. Train the last two VGG16 layers

12. #11 & dropout 0.1

13. Change learning rate to 0.01

14\_1. Train the last two vgg16 layers and change input size to 112

14\_2. Change input size to 112

14\_3. Change input size to 56

14\_4. Train the last two vgg16 layers and change input size to 56

15. Change the fully connected layer to Last layer / 2.25

16. Train the last two vgg layers and add two dense layers with the input size of 224 – Unable to train due to a lack of resource

17. Train the last vgg layer and add two dense layers with the input size of 224

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test | | Validation | |
| Recall | Precision | Recall | Precision |
| 1. 1st dense layer - relu | 1 | 1 | 0.770 | 0.510 |
| 2. Dropout 0.2 after dense layer | 1 | 1 | 0.218 | **0.673** |
| 2-1 Dropout 0.5 after dense layer | 1 | 1 | 0.747 | **0.596** |
| 3. 2 dense layers | 1 | 1 | 0.746 | **0.596** |
| 4. 1st dense – relu; Dropout 0.2 | 0.923 | 1 | 0.385 | **0.627** |
| 5. 2 dense layers; dropout 0.2 after dense | 0.472 | 0.873 | too low | |
| 6. 1st dense layer – relu; 2 dense layers | 1 | 1 | 0.725 | 0.554 |
| 7. 1st dense layer – relu; 2 dense layers; dropout 0.2 after dense | 0.665 | 1 | too low | |
| 8. Learning rate 0.001 | 1 | 1 | 0.716 | 0.504 |
| 9. ‘relu’ for 2 dense layers and ‘softmax’ for the last dense | predicted all samples positive – stopped teesting | | | |
| 10. Re-train the last VGG16 layer | 1 | 1 | **0.834** | 0.510 |
| 11. Re-train the last 2 VGG 16 layers | 1 | 1 | **0.836** | 0.557 |
| 12. Re-train the last 2 VGG 16 layers; dropout 0.1 | 1 | 1 | 0.741 | 0.546 |
| 13. Learning rate 0.01 | 0.879 | 0.799 | 0.756 | 0.484 |
| 14-1. Train the last 2 VGG16 layers; input size 112 | 1 | 1 | 0.676 | 0.368 |
| 14-2. Input size 112 | 1 | 1 | **0.812** | 0.513 |
| 14-3. Input size 56 | 0.939 | 0.979 | 0.756 | 0.579 |
| 14-4. Train the last 2 VGG16 layers; input size 56 | 1 | 1 | 0.737 | 0.550 |
| 15. Fully connected layer = Last layer / 2.25 | 1 | 1 | 0.735 | 0.510 |
| 16. Train the last 2 VGG16 layers; 2 dense layers; input size 224 | Unable to train due to a lack of resource | | | |
| 17. Train the last VGG16 layer; 2 dense layers; input size 224 | 1 | 1 | **0.782** | 0.506 |

In some of the cases above, only either recall or precision increased. Precision was always lower than recall.

Although the precision was only 0.557 and lower than the original model, #11 showed the highest recall rate.