## Template

|  |  |  |
| --- | --- | --- |
| Property | value | Description |
| Model | MobileNetV2\_tl | Transfer Learning |
| #classes/labels | {0: NO, 1:YES} | Binary classification |
| Total #images | 1736 |  |
| Train set | 1389 |  |
| Val set | 227 |  |
| Test set | 69 |  |
| batch\_size | 32 |  |
| input\_shape | (224, 224) | (64, 64), (150, 150), (224, 224) |
| normalizetion | [-1, 1] | [0, 1] / [-1, 1] |
| pre\_process | tekboart | base\_mode, tekboart |
| Dropout | 0.2 |  |
| L2 Regularization | None |  |
| Data Augmentation | YES (keras) |  |
| Class weight? | Yes |  |
| lr\_rate | 0.001 |  |
| lr\_schedule | decay\_step=100 |  |
| #epochs | 20 |  |
| Metrics | ‘loss’, ‘accuracy’, ‘precision’, ‘recall’, ‘auc’, ‘prc’, ‘tp’, ‘fp’, ‘tn’, ‘fn’ |  |
| callback | EarlyStopping(patience=5, ‘val\_loss’) |  |
| lr\_rate (for fine-tune) | 0.001 / 100 |  |
| lr\_schedule\_fine-tune | No |  |
| # un-freezed layers | +100 (54 out of 154) | Train layers from 100 and up |
| #epochs (for fine-tune) | 10 |  |
| Callback (for fine-tune) | No |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Performance Measure | Train | Val | Test |
| Loss |  |  |  |
| Accuracy |  |  |  |

\* the ResNet50\_v2 worked a lot faster & better than the ResNet50 we built from scratch, which is not a surprise as we had only 1736 images in total, and we know that transfer learning works great with small data.

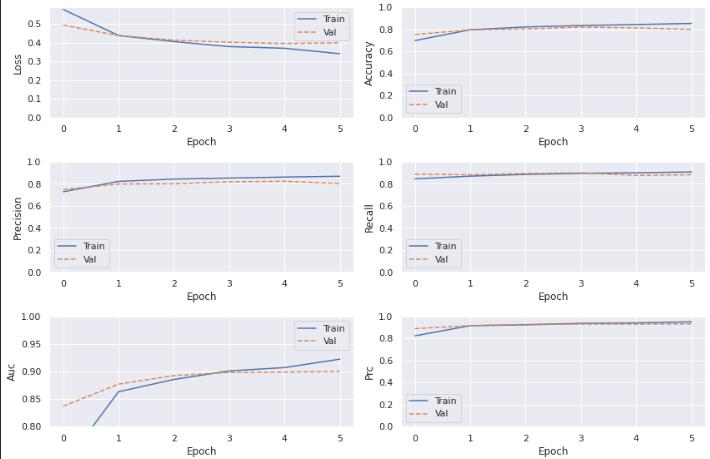
\* using the pretrained ResNet50v2 with [-1, 1] normalized inputs worked better than the [0, 1] values, so it’s safe to assume that ResNet50v2 had been trained with [-1, 1] values, rather than [0, 1].

## ResNet50v2\_tl

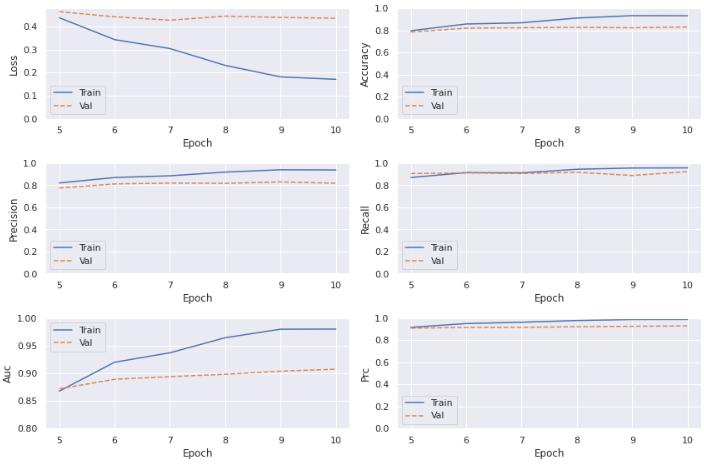
|  |  |  |
| --- | --- | --- |
| Property | value | Description |
| Model | MobileNetV2\_tl | Transfer Learning |
| #classes/labels | {0: NO, 1:YES} | Binary classification |
| Total #images | 1736 |  |
| Train set | 1389 |  |
| Val set | 227 |  |
| Test set | 69 |  |
| batch\_size | 32 |  |
| input\_shape | (224, 224) | (64, 64), (150, 150), (224, 224) |
| normalizetion | [-1, 1] | [0, 1] / [-1, 1] |
| pre\_process | tekboart | base\_mode, tekboart |
| Dropout | NO |  |
| L2 Regularization | None |  |
| Data Augmentation | YES (keras) |  |
| Class weight? |  |  |
| lr\_rate |  |  |
| lr\_schedule |  |  |
| #epochs |  |  |
| Metrics |  |  |
| callback |  |  |
| lr\_rate (for fine-tune) |  |  |
| lr\_schedule\_fine-tune |  |  |
| # un-freezed layers |  | Train layers from 100 and up |
| #epochs (for fine-tune) |  |  |
| Callback (for fine-tune) |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Performance Measure | Train | Val | Test |
| Loss |  |  |  |
| Accuracy |  |  |  |

\* using 224x224 images, in comparison with (150x150) ones, reduced overfitting and caused the model (i.e., ResNetV2\_pretrained) to perform a lot better



\* train only our added binary output (Dense(1))

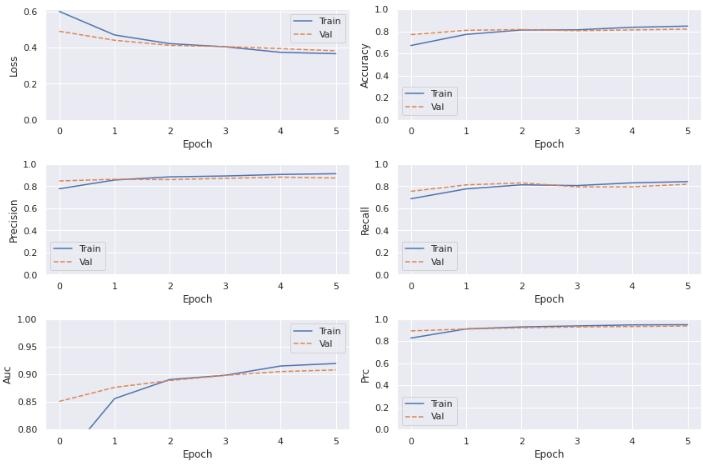


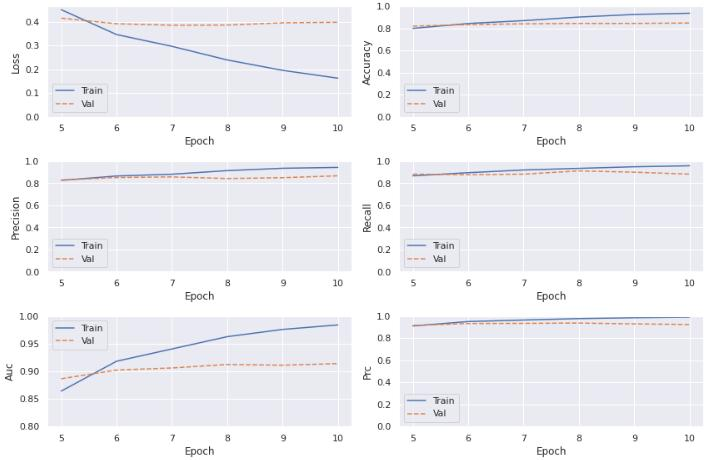


\* after fine-tune (by unlocking 50 of top layers )

**Results when using weight\_class (as our data is imbalanced) (no dropout)**

\* the result are better when applying the weight\_class, but not sure whether it’s the result of randomness in model (e.g., data augmentation,







## MobileNetV2

|  |  |  |
| --- | --- | --- |
| Property | value | Description |
| Model | MobileNetV2\_tl | Transfer Learning |
| #classes/labels | {0: NO, 1:YES} | Binary classification |
| Total #images | 1736 |  |
| Train set | 1389 |  |
| Val set | 227 |  |
| Test set | 69 |  |
| batch\_size | 32 |  |
| input\_shape | (224, 224) | (64, 64), (150, 150), (224, 224) |
| normalizetion | [-1, 1] | [0, 1] / [-1, 1] |
| pre\_process | tekboart | base\_mode, tekboart |
| Dropout | 0.2 |  |
| L2 Regularization | None |  |
| Data Augmentation | YES (keras) |  |
| Class weight? | Yes |  |
| lr\_rate | 0.001 |  |
| lr\_schedule | decay\_step=100 |  |
| #epochs | 20 |  |
| Metrics | ‘loss’, ‘accuracy’, ‘precision’, ‘recall’, ‘auc’, ‘prc’, ‘tp’, ‘fp’, ‘tn’, ‘fn’ |  |
| callback | EarlyStopping(patience=5, ‘val\_loss’) |  |
| lr\_rate (for fine-tune) | 0.001 / 100 |  |
| lr\_schedule\_fine-tune | No |  |
| # un-freezed layers | +100 (54 out of 154) | Train layers from 100 and up |
| #epochs (for fine-tune) | 10 |  |
| Callback (for fine-tune) | No |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Performance Measure | Train | Val | Test |
| Loss | .0914 | .3489 | .1720 |
| Accuracy | .9647 | .8728 | .9375 |

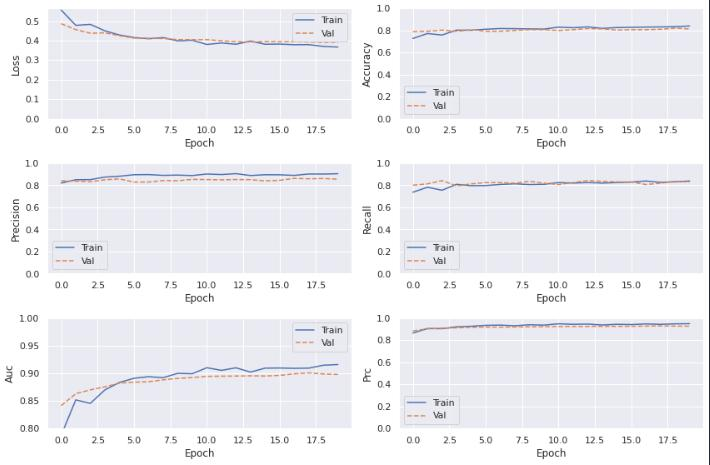


Figure 1: train the self added layers (i.e., Dense(1))

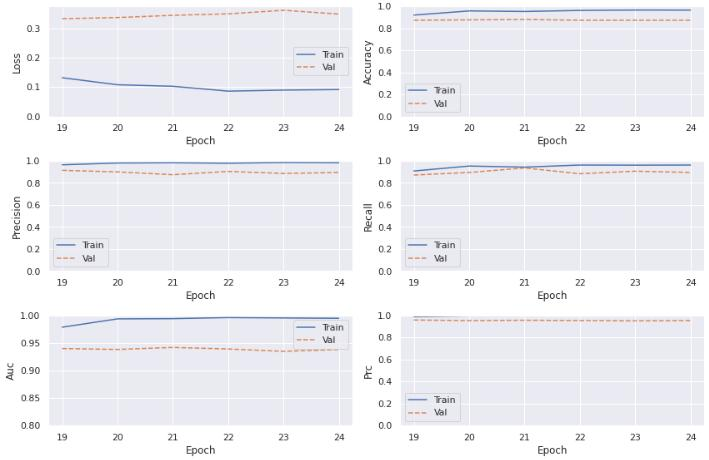
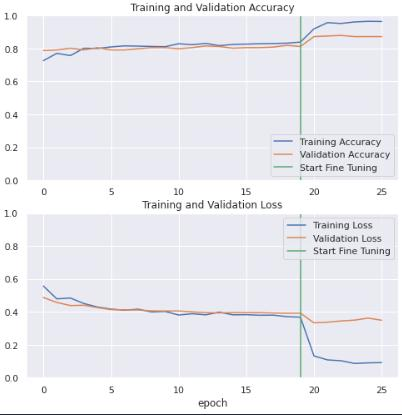
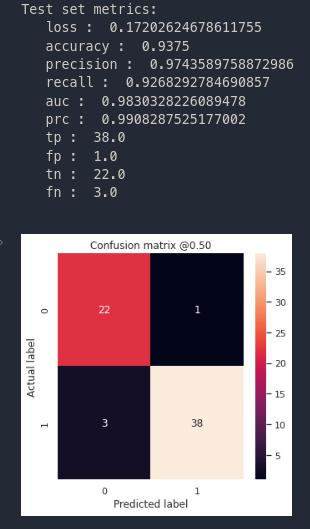


Figure 2: unfreeze some later layers (in base\_model) & re-train them

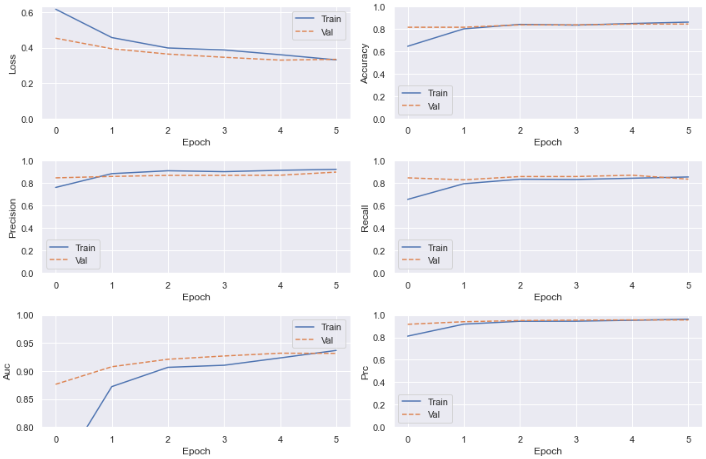




## ResNet50v2\_tl

|  |  |  |
| --- | --- | --- |
| Property | value | Description |
| Model | ResNet50v2\_tl | Transfer Learning |
| #classes/labels | {0: NO, 1:YES} | Binary classification |
| Total #images | 1736 |  |
| Train set | 1389 |  |
| Val set | 227 |  |
| Test set | 69 |  |
| batch\_size | 32 |  |
| input\_shape | (512, 512) | (64, 64), (150, 150), (224, 224) |
| normalizetion | [-1, 1] | [0, 1] / [-1, 1] |
| pre\_process | tekboart | base\_mode, tekboart |
| Dropout | 0.2 |  |
| L2 Regularization | None |  |
| Data Augmentation | YES (keras) |  |
| Class weight? | NO |  |
| lr\_rate | 0.001 |  |
| lr\_schedule | decay\_step=100, decay\_rate=0.6, staircase=True |  |
| #epochs | 6 |  |
| Metrics | ‘loss’, ‘accuracy’, ‘precision’, ‘recall’, ‘auc’, ‘prc’, ‘tp’, ‘fp’, ‘tn’, ‘fn’ |  |
| callback | EarlyStopping(patience=5, ‘val\_prc’) |  |
| lr\_rate (for fine-tune) | 0.001 / 10 |  |
| lr\_schedule\_fine-tune | No |  |
| # un-freezed layers | +140 (50 out of 190) | Train layers from 140 and up |
| #epochs (for fine-tune) | 8 |  |
| Callback (for fine-tune) | No |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Performance Measure | Train | Val | Test |
| Loss | .0914 | .3489 | .1720 |
| Accuracy | .9647 | .8728 | .9375 |



MobileNetV3

MobileNetV3Large