## Transfer Learning

### DenseNet121

The DenseNet 121 model was pretrained on ImageNet data. The transfer approach we use here was to enable training from layer ‘conv4\_block1\_preact\_bn’ onwards and has head layers:

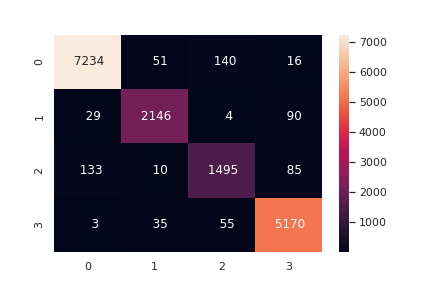
* GlobalAveragePooling2D
* Dense(512, ‘leaky\_relu’)
* Dense(128, ‘leaky\_relu’)
* Dense(4, softmax)

The model was compiled with lr=0.00001 and loss function sparse\_categorial\_crossentropy, and 40 epochs with a batch size of 4.

Training time 20min per epoch on RTX 3060 with 12GiB

The train/validation split was 66788/16696

The results on the validation set are summarized below.



CNV - 0

DME - 1

Drusen - 2

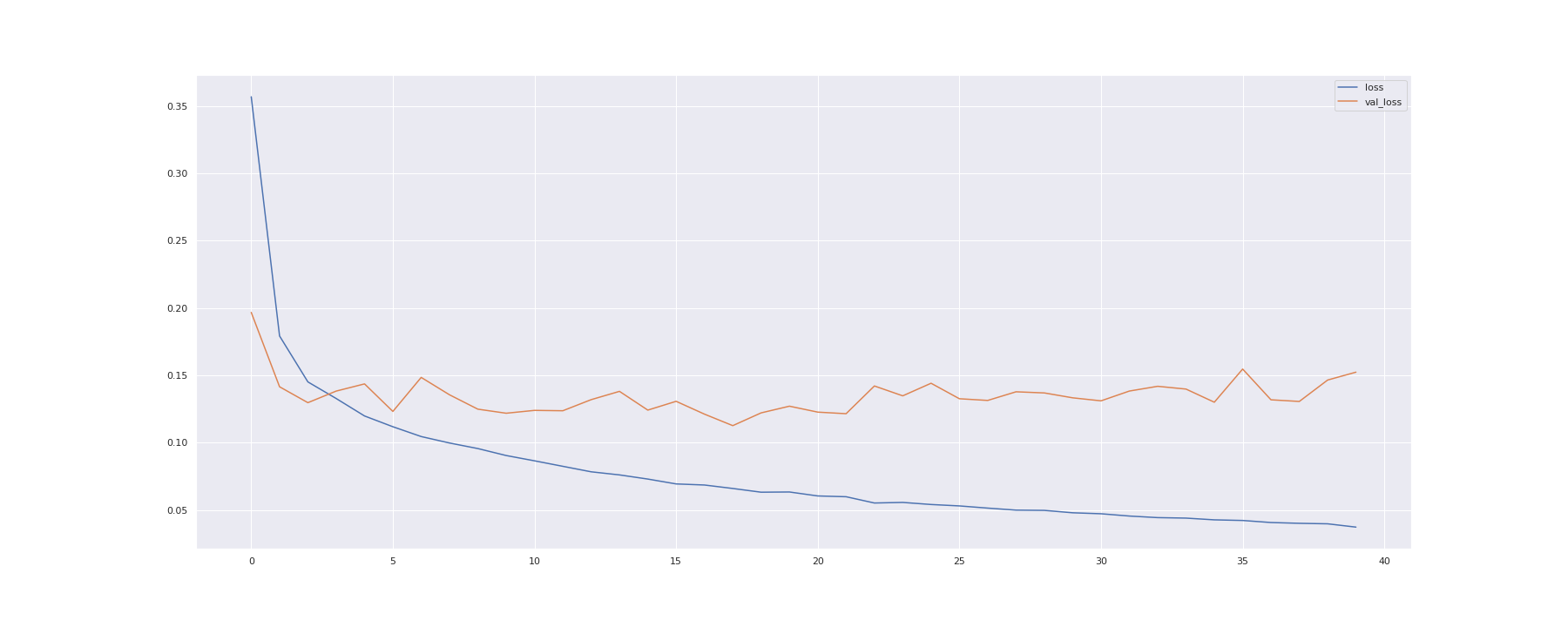
Normal - 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
|  |  |  |  |  |
| CNV | 0.98 | 0.97 | 0.97 | 7441 |
| DME | 0.96 | 0.95 | 0.95 | 2269 |
| DRUSEN | 0.88 | 0.87 | 0.88 | 1723 |
| NORMAL | 0.96 | 0.98 | 0.97 | 5263 |
|  |  |  |  |  |
| accuracy |  |  | 0.96 | 16696 |
| macro avg | 0.95 | 0.94 | 0.94 | 16696 |
| weighted avg | 0.96 | 0.96 | 0.96 | 16696 |

Accuracy



Loss

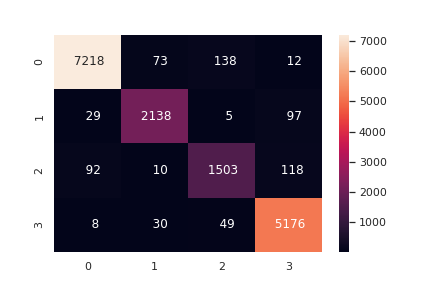


The notebook and the above images and table data are available in our repo on github

### 

### ResNet101

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
|  |  |  |  |  |
| CNV | 0.98 | 0.97 | 0.98 | 7441 |
| DME | 0.95 | 0.94 | 0.95 | 2269 |
| DRUSEN | 0.89 | 0.87 | 0.88 | 1723 |
| NORMAL | 0.96 | 0.98 | 0.97 | 5263 |
|  |  |  |  |  |
| accuracy |  |  | 0.96 | 16696 |
| Macro avg | 0.94 | 0.94 | 0.94 | 16696 |
| Weighted avg | 0.96 | 0.96 | 0.96 | 16696 |



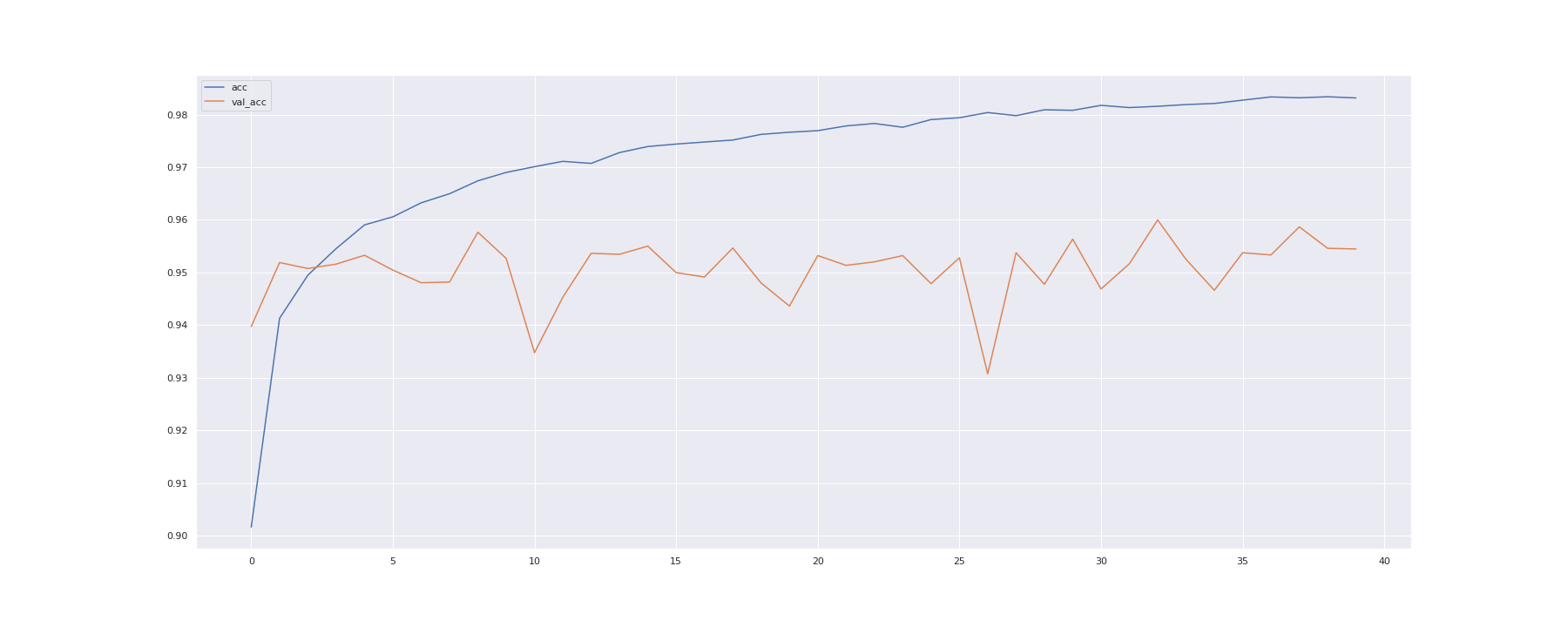
CNV - 0

DME - 1

Drusen - 2

Normal - 3

Accuracy



### 

### Custom Model

Conv2D/Pooling:

Five blocks of the form:

tf.keras.layers.Conv2D(X, (3,3), strides=1, padding='same', activation='leaky\_relu', kernel\_regularizer=regu),

tf.keras.layers.Conv2D(X, (3,3), strides=1, padding='same', activation='leaky\_relu', kernel\_regularizer=regu),

tf.keras.layers.MaxPooling2D(2,2, padding='same'),

training head:

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(256, activation='relu', kernel\_regularizer=regu), tf.keras.layers.Dense(4, activation='softmax')])

Model trained with:

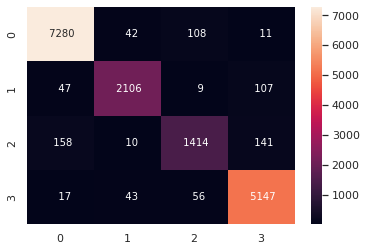
- adam optimizer

keras.optimizers.adam\_v2.Adam(learning\_rate=0.00001

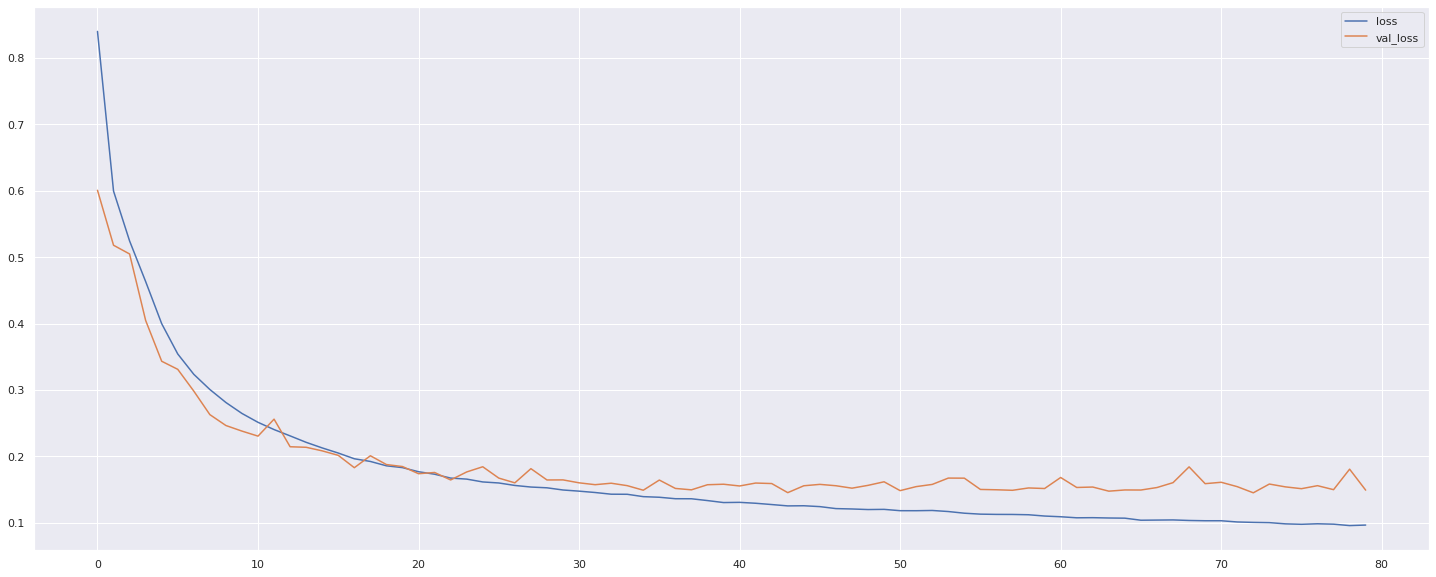
- 80 epochs

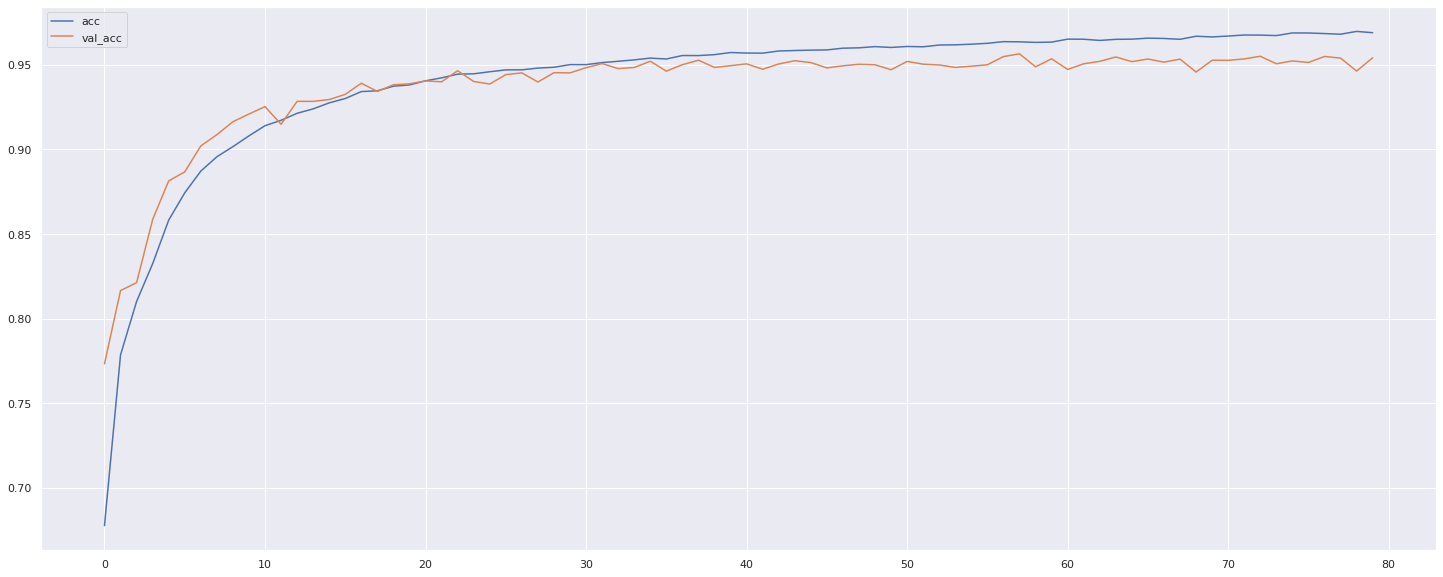
- batch size 4

- time per epoch 20 mins on GTX 1660



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
|  |  |  |  |  |
| CNV | 0.97 | 0.98 | 0.97 | 7441 |
| DME | 0.96 | 0.93 | 0.94 | 2269 |
| DRUSEN | 0.89 | 0.82 | 0.85 | 1723 |
| NORMAL | 0.95 | 0.98 | 0.96 | 5263 |
|  |  |  |  |  |
| accuracy |  |  | 0.96 | 16696 |
| Macro avg | 0.94 | 0.93 | 0.93 | 16696 |
| Weighted avg | 0.95 | 0.96 | 0.95 | 16696 |

Accuracy 

Loss 

Kermany et al.2018

Goal: Clinal Support system

- Old way

- hand crafted feature

- specialized segmentation classifiers

- domain expert involvement

- multiple image classifiers

- (a lot more work) but human and computation

- New way

- off the shelf technologies: Tensorflow, Pretrained deep CNN (InceptionV3)

Gtx 1060 GPU

- relative small data set hundreds of thousands not millions

- very good results 96% + accuracy, sensitivity and specificity

- transferability: X-rays for viral, bacterial pneumonia.

### Summary of SimCLR:

- paper 1

- unsupervised pretraining, followed by fine-tuning on 1% of data (ImageNet 2012)

- simpler than prior contrastive loss system: no specialized architecture (ResNet)

- Use of large batch sizes

- composition of data augmentation

- Use of LARS optimizer (You et al 2017)

- deep and wide network ( e.g ResNet151 2x, 3x)

- paper 2

- ability to transfer “knowledge” to smaller student network

- experience

- use 4 pre-trained SimCLR models ResNet50 1%, ResNet151 1%, ResNet50 10%, ResNet151 3x 10%

- ResNet151 3x 10%: training head only for 6 epochs achieved 91% accuracy 60+ minutes per epoch

- ResNet151 1%: training head only for 6 epochs achieved 89%+ accuracy 40+ mins per epoch

- ResNet50 10%: training head only for 6 epochs achieved 90% accuracy 20+ mins per epoch

- ResNet50 1%: training head only for 6 epochs achieved 89%+ accuracy 20+ mins

Additional paper highlights a potential feature suppression in multi-object images