Human Protein Atlas

Team members

Alexander Umale

Vigneshwaran Vasanthakumar

Introduction

The aim of this project is to develop models capable of classifying mixed patterns of proteins in microscope images. This model will be used to build a tool integrated with a smart-microscopy system to identify a protein's location from a high-throughput image. Historically, classification of proteins has been limited to single patterns in one or a few cell types, but in order to fully understand the complexity of the human cell, models must classify mixed patterns across a range of different human cells.

Dataset

Dataset consists of microscopic images of subcellular protein patterns. The size of the dataset is 17GB and it consists of 512x512 PNG files. There are four filters for each sample.

- The green filter is for the protein of interest.
- The blue filter is for the nucleus.
- The red filter is for the microtubules.
- The yellow filter is for the endoplasmic reticulum.

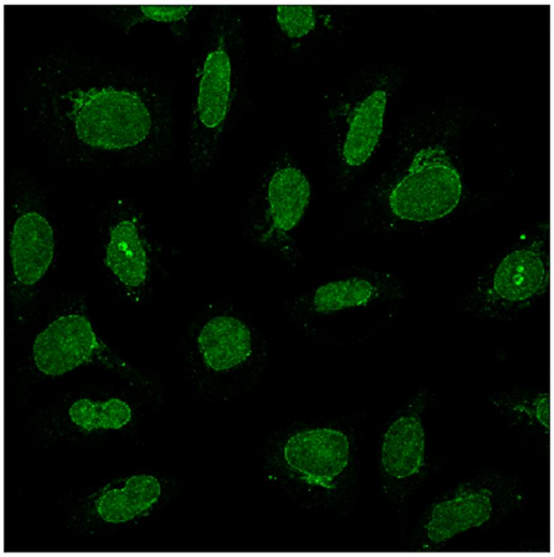
The green filter will be used to predict the labels and the other filters will simply be used as a reference. There are in total 28 different labels present in the dataset.

The labels are represented as integers that map to the following:

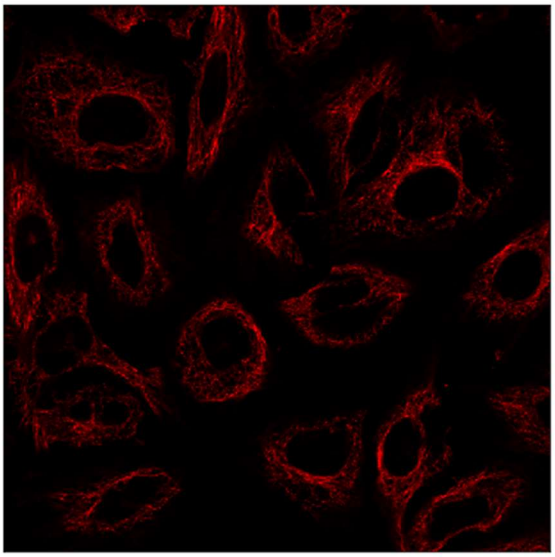
```
0.  Nucleoplasm
1.  Nuclear membrane
2.  Nucleoli
3.  Nucleoli fibrillar center
4.  Nuclear speckles
5.  Nuclear bodies
6.  Endoplasmic reticulum
7.  Golgi apparatus
8.  Peroxisomes
9.  Endosomes
10.  Lysosomes
11.  Intermediate filaments
12.  Actin filaments
13.  Focal adhesion sites
14.  Microtubules
15.  Microtubule ends
16.  Cytokinetic bridge
17.  Mitotic spindle
18.  Microtubule organizing center
19.  Centrosome
20.  Lipid droplets
```

21. Plasma membrane
22. Cell junctions
23. Mitochondria
24. Aggresome
25. Cytosol
26. Cytoplasmic bodies
27. Rods & rings

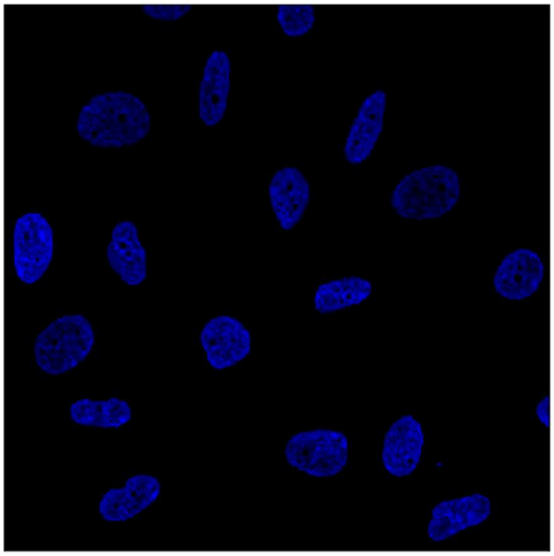
Protein of interest



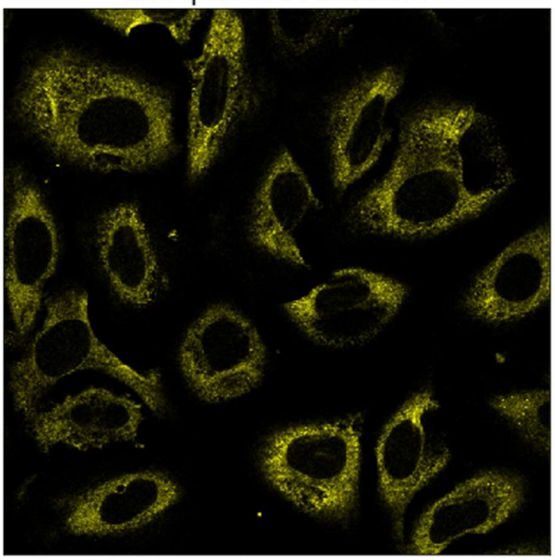
Microtubules



Nucleus

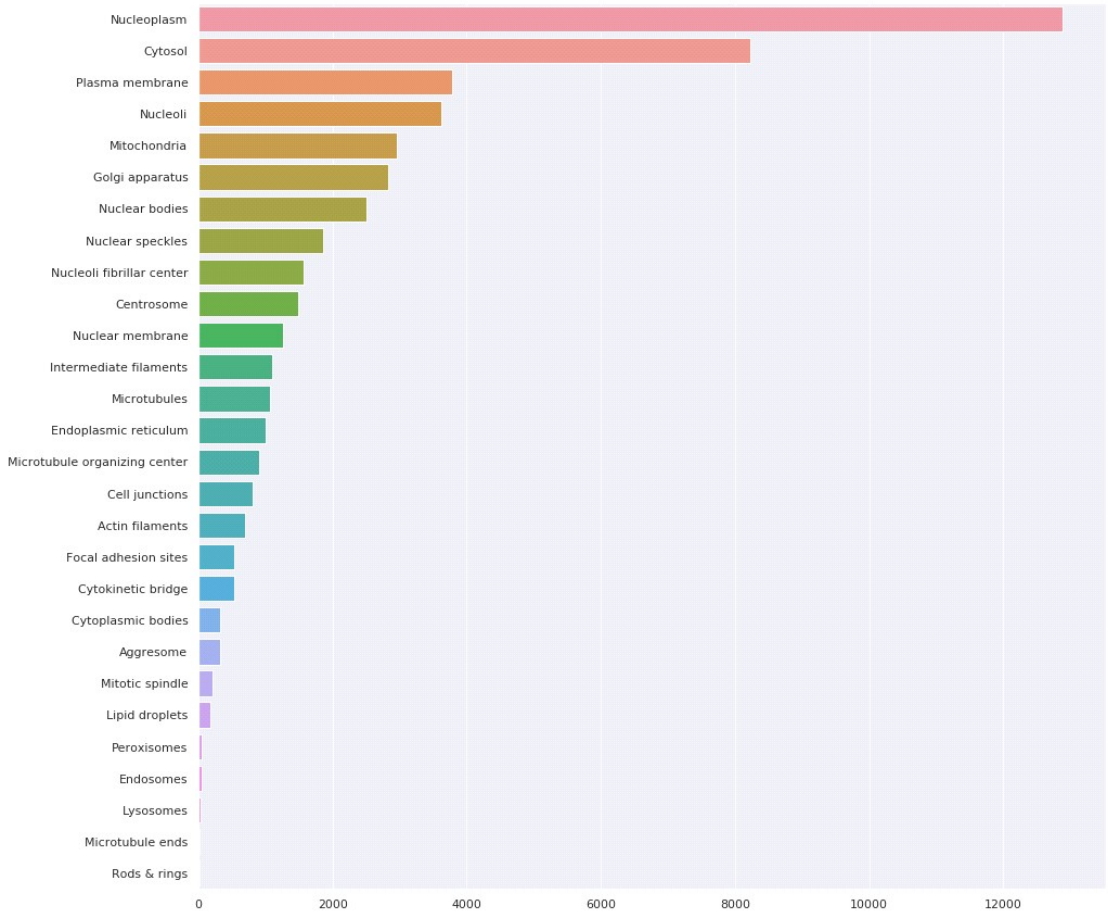


Endoplasmic reticulum

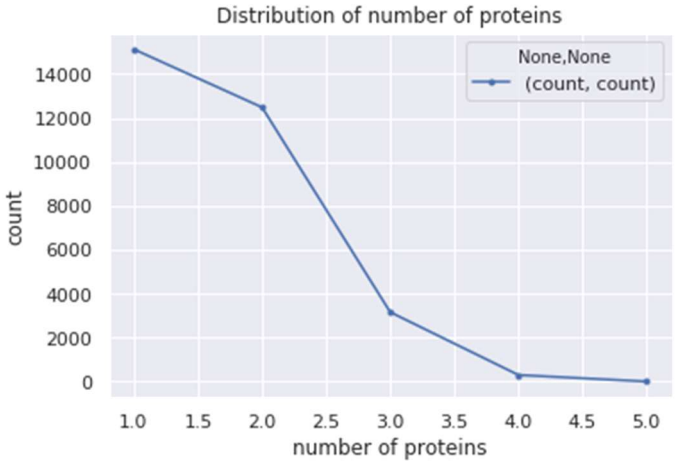


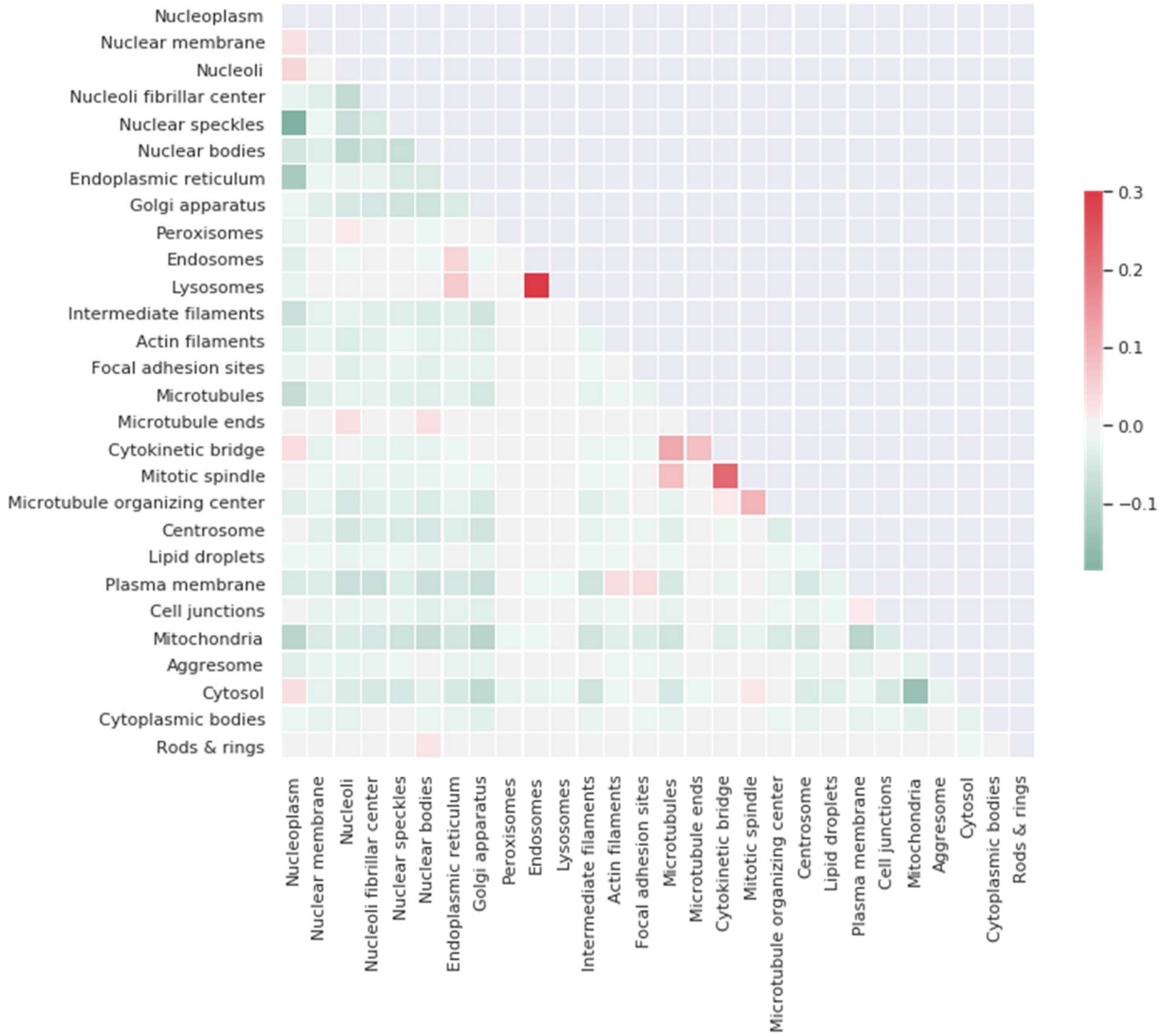
Label occurrence in train data:

The labels are unevenly distributed in the dataset. In the entire dataset Nucleoplasm is in almost all the images, followed by Cytosol. This is one of the factors which could reduce the scoring metric, due to false positives.



Label distribution in training data:





As can be seen in above plot, labels are not only unevenly distributed they are also sparse in. This dataset will work very good in a single label classification problem. However, the task at hand is of multi-label classification and this sparse and uneven distribution will result in high false positive and negatives.

From the correlation plot we can see that even though there's a strong correlation between Lysosomes and Endosomes, Mitotic Spindle and Cytokinetic bridge; rest of the labels are highly un-correlated. So not only is the dataset sparse and uneven it's also highly uncorrelated making the task even more difficult, as the Neural net will have hard time finding pattern in the dataset distribution.

Model

To setup a baseline of how good our model is doing we first used a few older models.

First, we used a simple convolutional neural network with three convolutional layers and one dense layer. Primarily this was used to see if our data pre-processing is properly working with a Neural Network.

Simple CNN Model details:

- INPUT: Input layer of shape [None, 512, 512, 4]
- CONV0: 16 filters, uniformly initiated kernel of size [5,5] with no padding
- CONV1: 32 filters, uniformly initiated kernel of size [3,3] with no padding
- CONV2: 64 filters, uniformly initiated kernel of size [3,3] with no padding
- DENSE0: 28 units, uniformly initiated

The second model we used is **VGG16**, available in Keras library. VGG16 consists of 16 layers. This model is pre-trained on the ImageNet dataset. Planned training using transfer learning is of 50 epochs on entire training dataset.

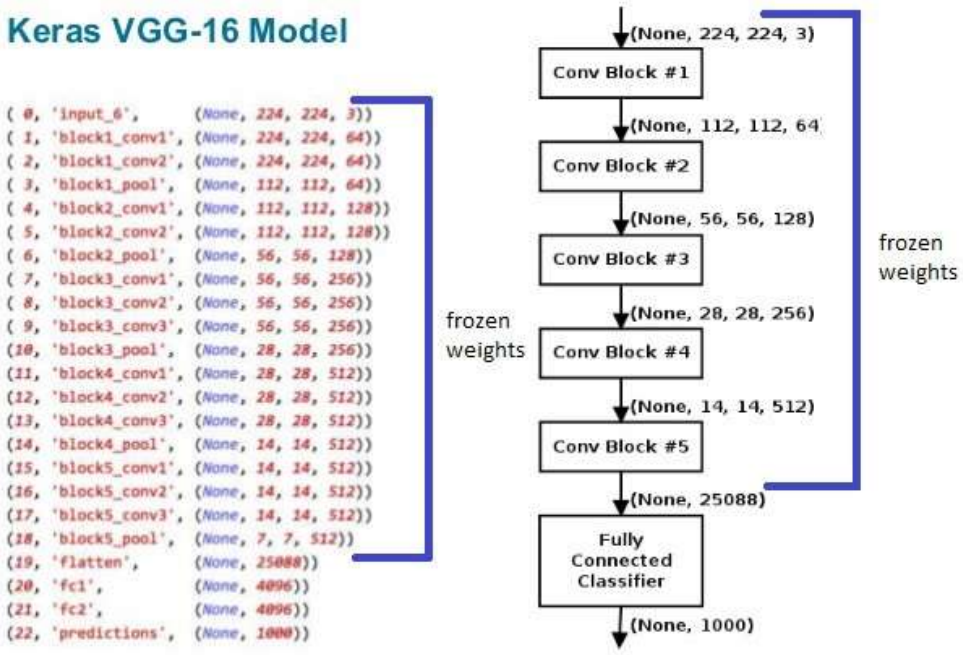


Figure 1 VGG16

Third model we used is the **ResNet-50**. This model was trained from scratch on the entire dataset for planned 100 epochs in Keras using tensorflow Dataset API.

layer name	output size	18-layer	34-layer	50-layer	101-layer
conv1	112×112	7×7, 64, stride 2			
		3×3 max pool, stride 2			
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512] ×3

Figure 2 ResNet-50

The model we actually planned to use for this task is **DenseNet**, specifically its 121-layer model DenseNet121. It consists of a very deep 121 convolution layers with very shallow width.

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56				
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6			
Transition Layer (1)	56 × 56				
	28 × 28				
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12			
Transition Layer (2)	28 × 28				
	14 × 14				
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24			
Transition Layer (3)	14 × 14				
	7 × 7				
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16			
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

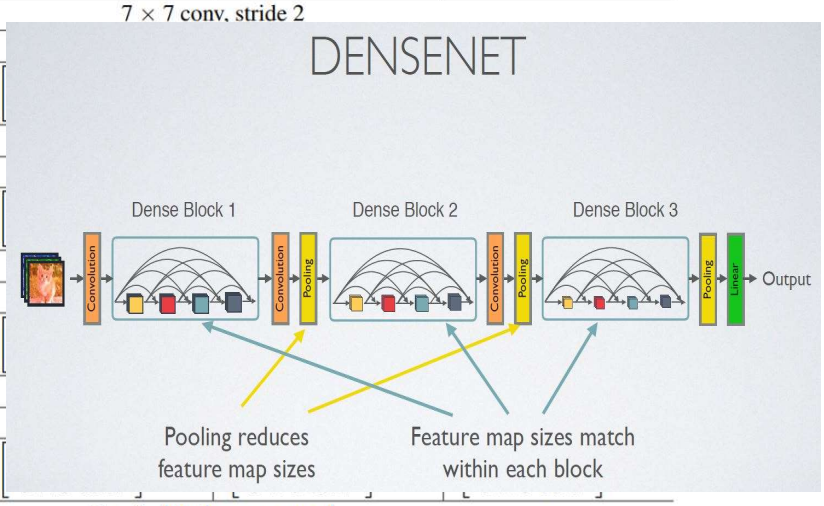


Figure 3 DenseNet-121

Since keras prebuilt VGG16 and ResNet50 will be used we are limited to use 3 images out of 4 as these models pose a limit of channel to be either 1 or 3. We are choosing Green, Red and Yellow images for the task.

Data Processing: Data batch generation

To use the dataset with the models they have to be batched together and normalized. The data consist of 4 images for each data point in the dataset. To use all four images, we decoded each image as a single channel array then concatenate them along the channel axis. This gives us a single tensor of dimension Width*Height*4, images being in order RGBY. The dataset is split into 90:10 ratios of train and holdout validation datasets.

The dataset is loaded using a pre-generated image id list reshuffling it decoding all 4 images for an id. Along with this a label file is also loaded converting the label from a list of numbers to n-hot encoding format. The image tensor, label and image-id are batched together using a batching function and these three values are returned. To improve processing time, we prefetch another batch of images beforehand.

This entire process is made easy by Tensorflow's Dataset API.

Experiment

For all the 4 models we used a fixed learning rate of 0.01. Batch size is 32, 16, 16, 8 for the model Simple CNN, VGG, ResNet, and DenseNet. An early stopping criteria of break patience of 5 is used.

Machine used for the experiment is google cloud with 2 vCPU 16GB RAM, Nvidia Tesla K80 12GB vRAM.

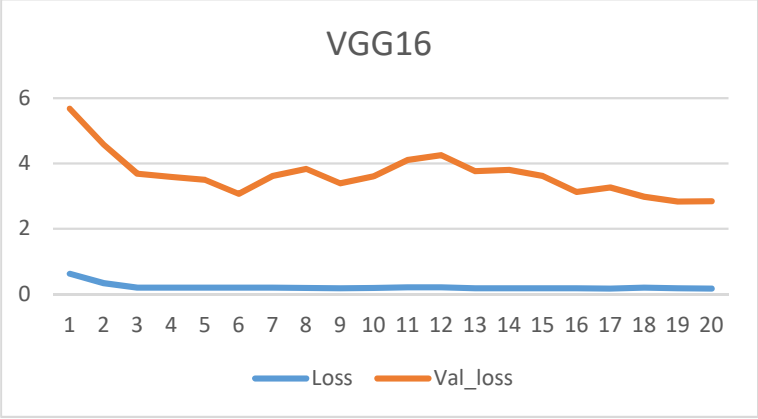
Loss is Sigmoid Cross Entropy and for accuracy matric F1 score is used.

Simple CNN

Simple CNN is trained for 100 epochs but stopped at 24 epochs due to early stopping. Final F1 score obtained is 0.23 on the validation dataset. Training stopped after 4.3 hours.

VGG16

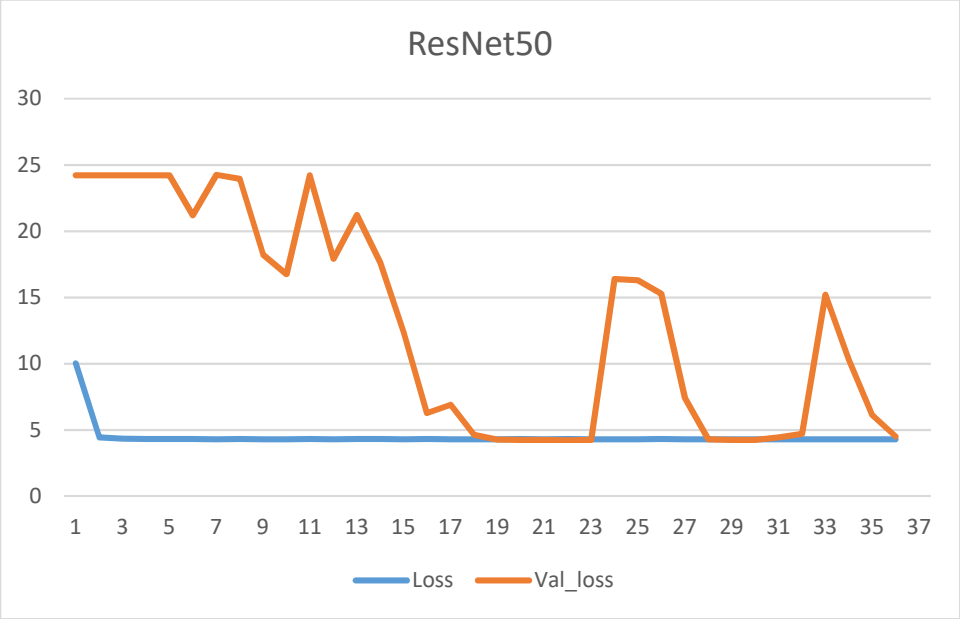
The training time for this model on the Human Protein Atlas dataset was 13.2 hours. There was a significant improvement in the accuracy matric.



With VGG 16 we were able to get best F1 score of 0.458

RESNET50

ResNet was trained for 100 epochs where it stopped at 36 due to early stopping. This took 15.66 hrs and scored 0.492 on F1 metric.



DenseNet

Densenet is one of the newest models right now. So we wanted to see how does it perform compared to others. Our last try with this model which is DenseNet121 took 8.12 hours to complete 16 epochs at which it stopped. The best F1 score obtained here was 0.5327.

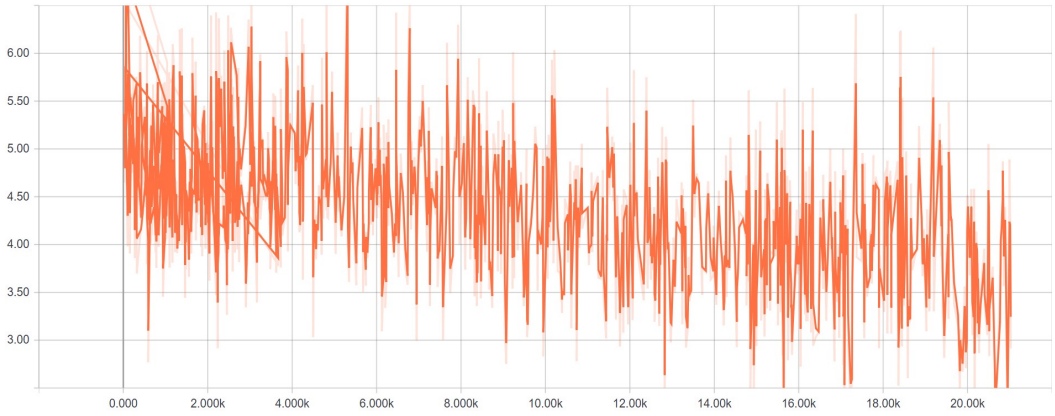


Figure 4 Training Loss

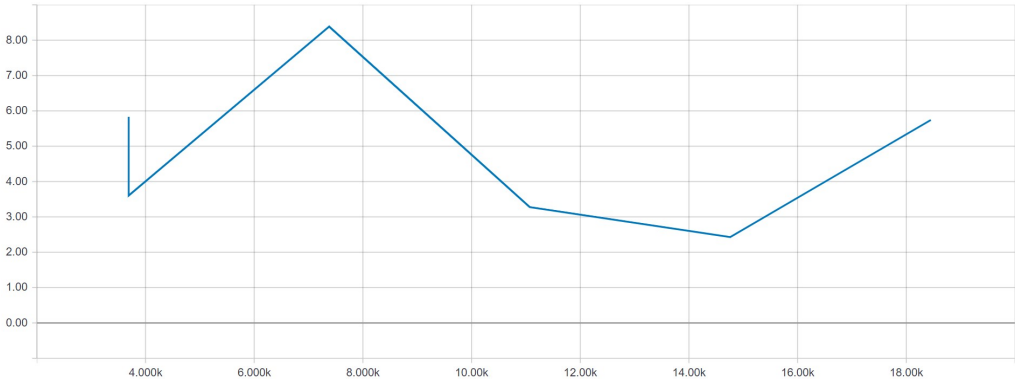


Figure 5 Validation Loss

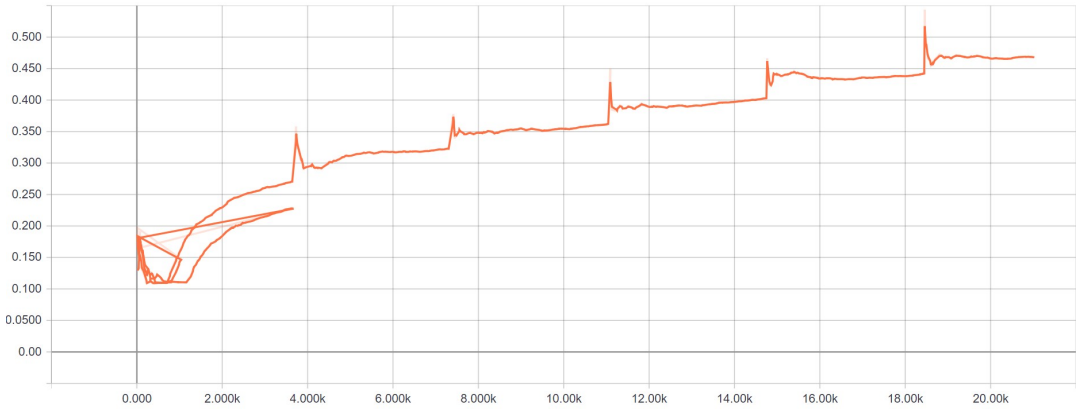


Figure 6 Training F1 Score

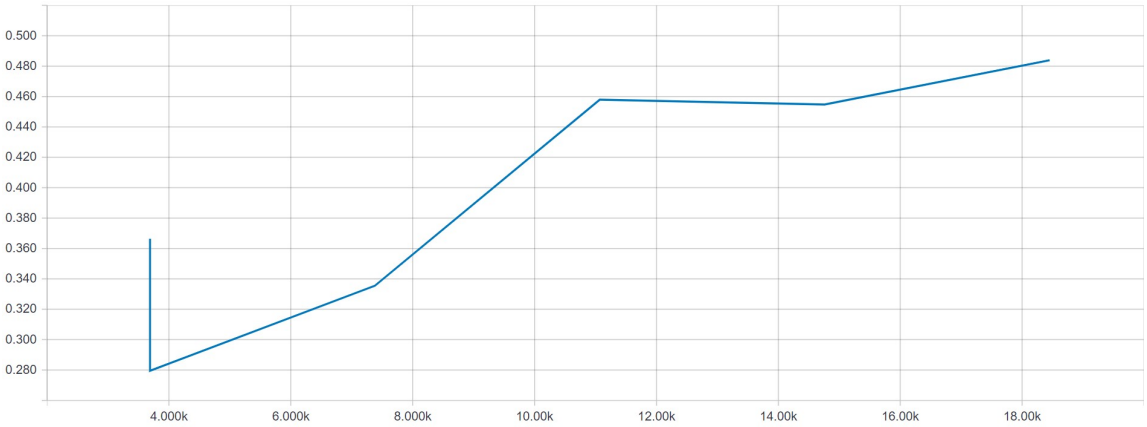
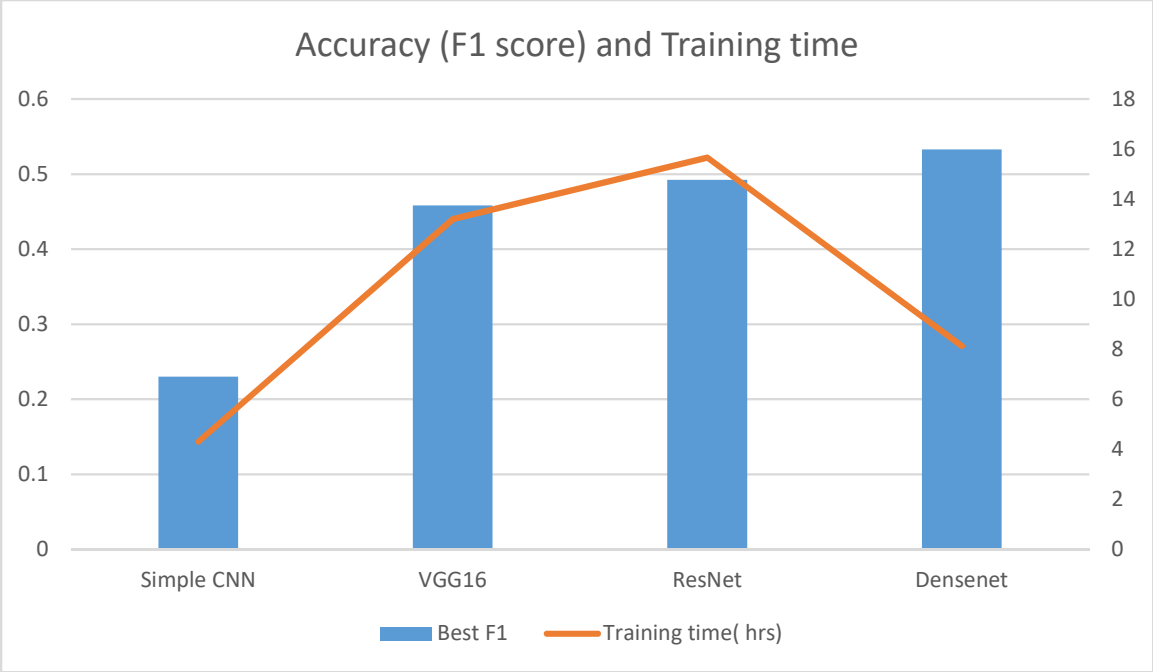


Figure 7 Validation F1 Score



Above is plotted Various Models with their score and Training time

References:

Protein Atlas Dataset: https://www.kaggle.com/c/human-protein-atlas-image-classification/data

DenseNet: https://arxiv.org/pdf/1608.06993.pdf#page=4

F1 Metric in Keras: https://medium.com/@thongonary/how-to-compute-f1-score-for-each-epoch-in-keras-a1acd17715a2

Keras VGG16: https://keras.io/applications/#vgg16

Keras ResNet50: https://keras.io/applications/#resnet50