Capstone Project

Machine Learning Engineer Nanodegree

Dhirendra Kumar
March 08, 2019
Capstone proposal review link.

Human Protein Atlas Image Classification

Classify subcellular protein patterns in human cells

Definition

Project Overview

The Goal of this project is to develop a model capable of classifying mixed patterns of proteins in microscope images. However, unlike most image labeling tasks, where binary or multiclass labeling is considered, in this competition each image can have multiple labels.

Here all image samples are represented by four filters (stored as individual files), **the protein of interest (green)** plus three cellular landmarks: **nucleus (blue)**, **microtubules (red)**, **endoplasmic reticulum (yellow)**. Therefore, An additional challenge is 4-channel input to the model (RGBY), which is different from ones used in most of pretrained models (RGB input).

Images we will be using for this are generated with the help of microscope and at a far greater pace than what can be manually evaluated. Therefore, the need is greater than ever for automating biomedical image analysis to accelerate the understanding of human cells and disease.

Problem Statement

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. And automating the biomedical image analysis will accelerate the understanding of human cells and disease.

And that can be achieved by using CNN models. However, unlike most image labeling tasks, where binary or multiclass labeling is considered, in this competition each image can have multiple labels. Which can be solved by using a function similar to sigmoid function to get more than one labels for sample images.

I would use a pretrained Resnet architecture it is one of the best architecture used for image classification and I want leverage the benefits of transfer learning because I have seen that with the use of transfer learning the better results can be produced in less time.

Evaluation Metrics

The official evaluation of the developed model is done by kaggle using **F1 score**. **F1 score** of a model is calculated by the the **Harmonic Mean** of **precision** and **recall**. where an F_1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

Where precision is the ratio of the number of true positive and the all positive labels predicted by proposed model, And recall is **is the number of correct positive results divided by the number of all positive results returned by the classifier, and** *r* **is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).**

To understand it better let's say we have total **m+n** labels for some dataset, in which labels for **m** samples are **positive** and for the rest of **n** is **negative**, and when we applied the proposed model to these samples, it predicted **positive** for **P** samples and prediction was correct for **Q (Q<=P)** samples, then precision is **Q/P** and recall is **Q/m.**

And I think the F1 score is a better evaluation metrics than any other evaluation metrics, because the data here is highly imbalanced, to understand this better let's take an example of credit card fraud detection and assume there are data of 100,000 transactions and 1200 of them are fraudulent transactions.

Now, let's assume a model that is not so good and predicts that the all transaction are good, then if we calculate accuracy here that will be more than 98%, and precision will be 100% since we made zero mistakes among the one who predicted positive which is what precision tries to measure. Now for the recall, the recall is how many fraudulent transaction did we catch, and since we catch none of them it will be 0% so the average b/w precision and recall is 50%, which is very good score for such a bad model since it is not able to catch any fraudulent transaction.

Now if we assume a model that says opposite of the previous model and predicts that all the transactions are fraudulent the precision will be 1200/100,000 and recall will be 100% because the model caught all of them, and the average of them gives a high score to such a bad model. So these model does not give the right score to the model if the data is highly imbalanced. That is why we use F1 score because we want to score our model better if both precision and recall are high and F1 score

helps us do that. And it raises a flag if one of them is small. As we can see in our example the F1 score will be zero for the first model and very close to zero for the second model indicating that the these models are not the good model for the problem.

Analysis

Data Exploration and Visualization

The data format is two-fold:

The bulk of the data is in the images (scaled set of 512x512 PNG) - **train.zip** and **test.zip**. Within each of these is a folder containing four files per sample. Each file represents a different filter on the subcellular protein patterns represented by the sample.

The labels for **train.zip** are provided for each sample in **train.csv.** And the labels for **test.zip** are not provided because the prediction of the developed model will be compared with the actual labels after the competition ends. So, here images from the **train.zip** will be used for the development of the model and the images from **test.zip** will be used for evaluation of the model.

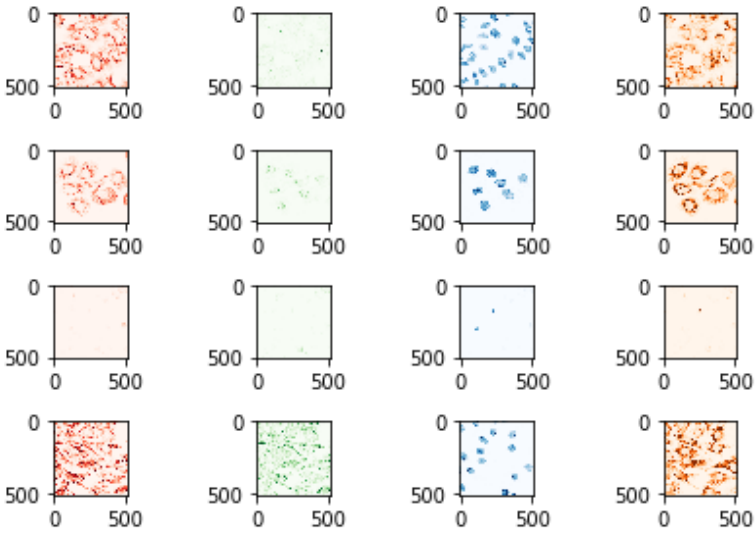
There are in total 28 different **protein organelle localization labels** present in the dataset.

- 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"

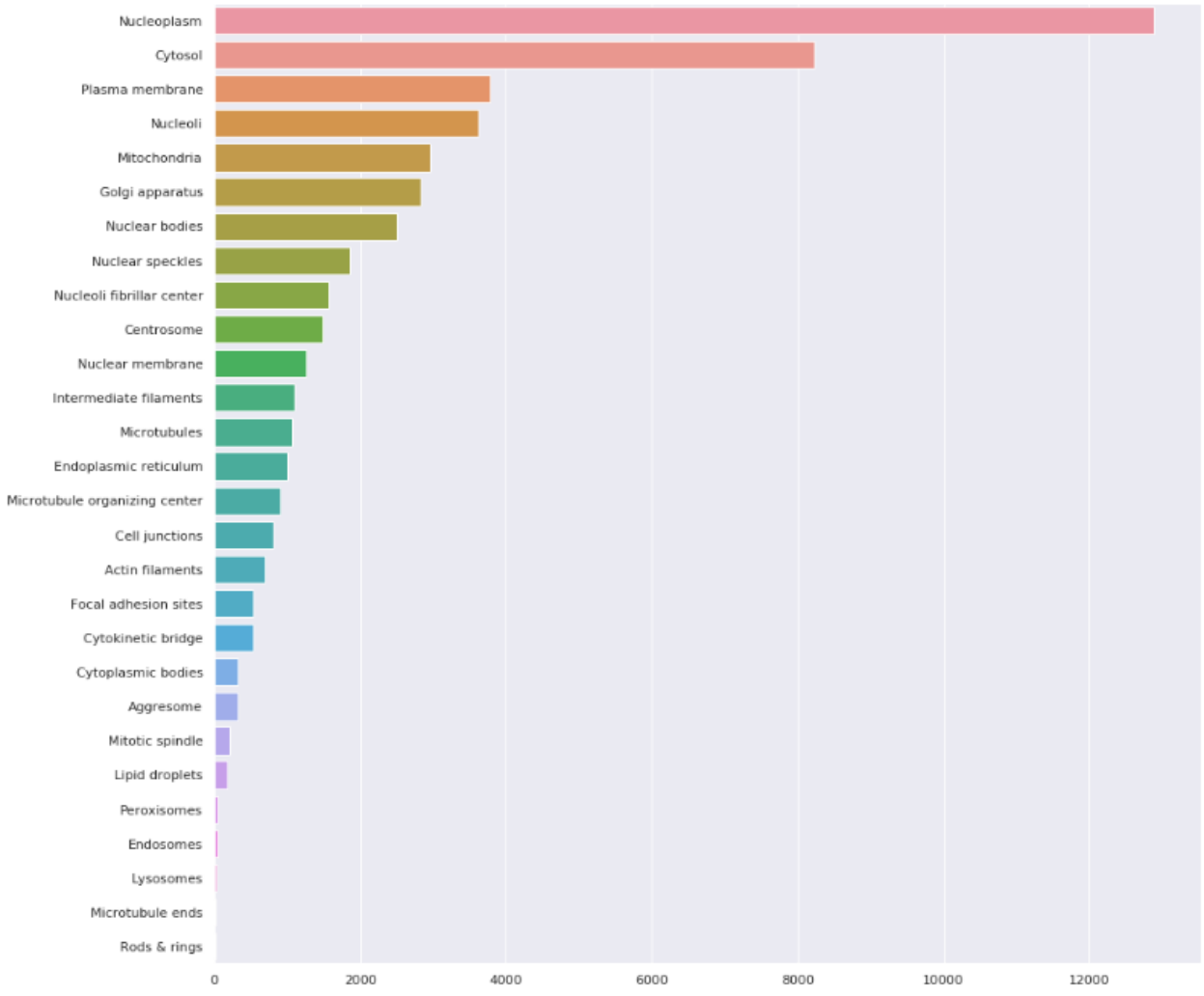
The dataset is acquired in a highly standardized way using one imaging modality (confocal microscopy). However, the dataset comprises 27 different cell types of highly different morphology, which affect the protein patterns of the different organelles. All image samples are represented by four filters (stored as individual files), **the protein of interest (green)** plus three cellular landmarks: **nucleus (blue)**, **microtubules (red)**, **endoplasmic reticulum (yellow)**. The green filter should hence be used to predict the label, and the other filters are used as references.

The data for this project is provided by Kaggle and Human Protein Atlas, should be found here.



Visualization of Red, Green, Blue and yellow (respectively) colour channels for some images is given in this image.

Now, here is a visualization of which proteins occur most often in train images.



Algorithms and Techniques

The goal of this competition is classification of mixed protein patterns. That can be achieved by using CNN models. However, unlike most image labeling tasks, where binary or multiclass labeling is considered, in this competition each image can have multiple labels. Which can be solved by using a function similar to sigmoid function to get more than one labels for sample images.

An additional challenge is 4-channel input to the model (RGBY), which is different from ones used in most of pretrained models (RGB input), And we are allowed to use all of them or only one colour channel of your choice (Green colour channel is recommended by the kaggle and I will use only green channel to produce results.)

I used pretrained **Resnet34** image classification model to build my model and train the model on training data and to predict results for test data.

Resnet is a neural network architecture which uses Convolutional layers, pooling layer and relu activation layers to capture and understand the information contained an image. And uses that information to separate one object in an image to the other. As we know the deeper the neural network the better it will be to capture the information of the data but after a certain point increasing the depth leads to exploding or vanishing gradients problem if weights are not properly initialized. And deeper networks experience degradation in convergence - with accuracy getting saturated and errors remaining higher than the shallower ones.

There are different network like this that are used for image classification but Resnet provide an advantage over them and allows us to build a deeper network without the problem of exploding or vanishing gradients, which perform better.

And the way Resnet does it without losing accuracy or increasing error is by using feedforward neural network with shortcut connections. As the paper says:

```
Shortcut connections are those skipping one or more layers. In our case, the
shortcut connections simply perform identity mapping, and their outputs are added
to the outputs of the stacked layers. Identity shortcut connections add neither
extra parameter nor computational complexity. The entire network can still be
trained end-to-end by SGD with backpropagation, and can be easily implemented using
common libraries without modifying the solvers.
```

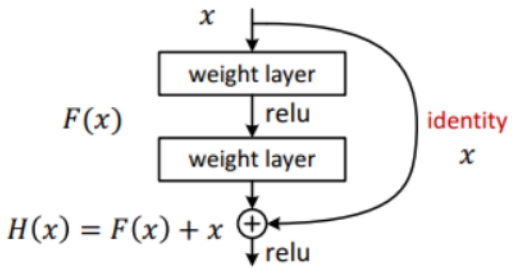


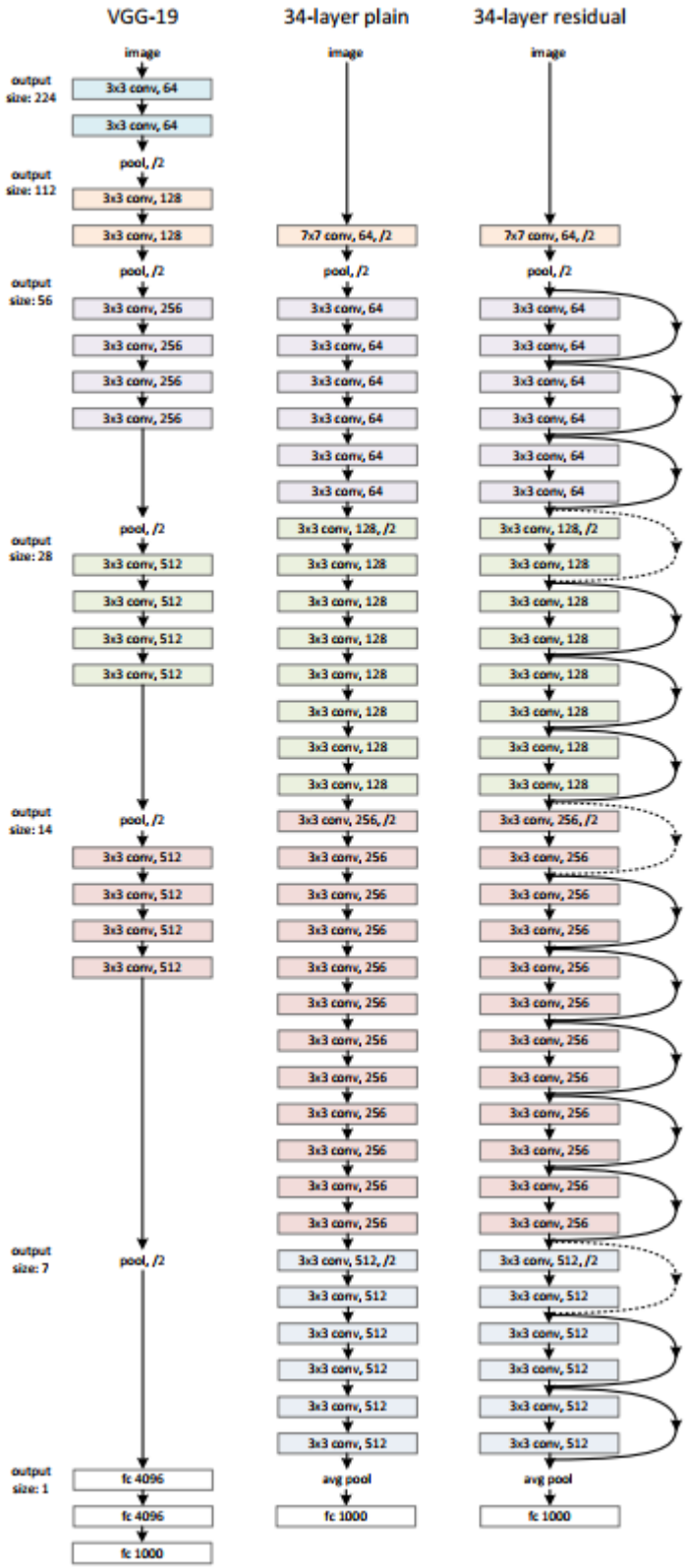
Fig.: The reusable residual network. (Img credit: https://arxiv.org/abs/1512.03385)

The network can be mathematically depicted as:

`H(x) = F(x) + x` , where `F(x) = W2*relu(W1*x+b1)+b2`

> During training period, the residual network learns the weights of its layers such that if the identity mapping were optimal, all the weights get set to 0. In effect F(x) become 0, as in x gets directly mapped to H(x) and no corrections need to be made. Hence these become your identity mappings which help grow the network deep. And if there is a deviation from optimal identity mapping, weights and biases of F(x) are learned to adjust for it. Think of F(x) as learning how to adjust our predictions to match the actuals.

Here is a arch of ResNet 34 .



And here are the steps I followed and Hyperparameters I tuned and can be tuned to get the best results.

- Define the model
- Define the evaluation metrics
- Define Batch size (how many images to look at once during a single training step)
- Define the regularization that you are going to use (weight decay, dropout, L1 or L2 regularization), I used dropout (0,5)
- Find a good learning rate (how fast to learn; this can be dynamic)
- Train your models until it stops improving (Define number of epochs)

During training, both the training and the validation sets are loaded into the RAM. After that, random batches are selected to be loaded into the GPU memory for processing, Although the images are very different then Imagenet Dataset but I used pretrained **Resnet34** just to have a good starting point and to make training faster a used GPU provided by kaggle in Kaggle kernels.

Benchmark Model

The official evaluation of the developed model is done by kaggle using **F1 score**. **F1 score** of a model

Since the proposed project is actually a kaggle competition, the benchmark will be the benchmark F1 score provided by kaggle for this competition for test set on **Leaderboard** which is **0.01955.**

So, I used **Resnet34** and F1 score for evaluation metrics and ended up in top 29% participants with the F1 score of 0.44936 (Kaggle link)

Methodology

Data Pre-processing

The images given here were 4 different images for 4 colour channels per samples, so first I had to figure out which colour channel I was going to use and which colour channels were giving the best results. Some people on kaggle combined all of colour channel and made a single image with 4 colour channels but I only used green colour channel for training and validation.

The labels for each samples were provided in **.csv file.** The given data was very imbalanced, there are some classes with large no. of samples and some classes with very few samples. And to overcome that problem I tried oversampling but that did not work as I was just making of the copies

of rare cases to create more data and splitting the final data to get training and validation data, So I was not sure with my validation score that is why I decided not to use oversampling. Some people used stratified cross validation, but there were some classes with only one samples that is why I thought that the results on validation data can be misleading. So I used only data augmentation for better results.

Implementation

I tried pretrained Resnet 34 model to classifying mixed patterns of proteins in microscope images and used F1 Score 'macro' for evaluation metrics and ended up in top 29% participants in this Kaggle competition.

- I used only green color channel images to train the model and used pretrained weights of the model just to have a good starting point.
- I imported Resnet 34 model from `from fastai.conv_learner` library and F1 Score `from sklearn.metrics` and created a learner object using the pretrained weights of Resnet 34, data and F1 score as Evaluation metrics.
- Used fastai's `.lr_find()` function to find an optimal learning rate for the model then I unfreezed the parameters of the model and used the optimal learning rate to train my model.

Code Snippet:

```
# evaluation metrics: F1 score 'macro'

def f1(preds, targs, start=0.17, end=0.24, step=0.01):
  with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    return max([f1_score(targs, (preds>th), average='macro')
            for th in np.arange(start,end,step)])

metrics=[f1]
learn = ConvLearner.pretrained(f_model, data, ps=0.5, metrics=metrics)
lr = 0.05
learn.unfreeze()
lrs=np.array([lr/10,lr/3,lr])
learn.fit(lrs/4, 5, cycle_len=1, cycle_mult=2)
```

Here we have F1 score of 0.37 on validation data which is 20% of the total training data and randomly chosen for validation and the score will be improved for testing data, because our training data has some rare cases with only one sample, So there are few cases where some validation samples appears in validation set not in the training set.

Here I want point out out that the F1 score of the model on validation data was 0.0989 and it is improved to 0.37 when I trained my model with the optimal parameters.

Refinement

- Used fastai's `.lr_find()` function to find an optimal learning rate.
- Used different learning rate to train different layers of the Neural Network because the initial layers need less training as they are already good at capturing simples features like edges and curved.
- Used learning Stochastic Gradient Descent With Restarts to ensure a better training of the model. And Dropout for regularization to avoid overfitting.
- Used test time Augmentation on test data so that our model can make better prediction by utilizing more data.

Results

Model Evaluation and Validation

I believe that the final model is reasonable and aligning with solution expectations, it gives a good validation score on validation data and the final model genelizes well with the unseen data as it gives a good score on kaggle leaderboard. The difficulty here was to choose a good learning rate and % of dropout, So to choose to use Cyclical Learning Rates method for setting the learning rate as described in the paper [Cyclical Learning Rates for Training Neural Networks](), which practically eliminates the need to experimentally find the best values and schedule for the global learning rates. And I experimented with the % of dropout I would use for the best results and used the best one.

To see if our model generalizes well to the unseen data, I would refer to test score against the both test data as the labels of the test data on kaggle is not provided, the data is truly unseen and it perform good on public and private leaderboard, confirms that the model generalizes well to unseen data.

And to see if our model is sensitive to small changes in our training data I did not use K Fold CV but what I did is I tuned the hyperparameters of the model with training data by separating it into two part training and validation and tested against test data, then I used the whole training data (Training and Validation data) with the exact same hyperparameter and tested against test test data again and As I was expecting it would perform better as this time some extra training data was added for training, it did perform better confirming that our model is robust to small changes in training data.

And to see If we can trust his model I would again refer to test scores of validation data public leaderboard score and private leaderboard score on kaggle and as it perform good in all three cases, but there are models in this competition who perform very good with test data. And As the training data was highly imbalanced, that is why we should fully trust this model.

Here is some predictions for the test images,

And as the labels for test data is not provided in the competition all we can see if the Final F1 score of the model and we can not see any confusion metrics or for which cases our model's performance was good or bad.

	Id	Predicted
0	00008af0-bad0-11e8-b2b8-ac1f6b6435d0_blue...	0 1 11 12 14 18 21 22 23 25
1	00008af0-bad0-11e8-b2b8-ac1f6b6435d0_gree...	0 10 12 14 16 18 21 25
2	00008af0-bad0-11e8-b2b8-ac1f6b6435d0_red.png	0 1 3 6 8 12 16 18 24
3	00008af0-bad0-11e8-b2b8-ac1f6b6435d0_yell...	0 1 3 6 12 14 15 16 18 23 24
4	0000a892-bacf-11e8-b2b8-ac1f6b6435d0_blue...	0 1 3 8 11 12 14 18 21 22 23 25

- The no. in the '**Predicted**' column separated by space refer to different kind of proteins as can be seen in data exploration section. And there are more than one proteins available in every image sample, that is why there are more than on labels for each image.

Justification

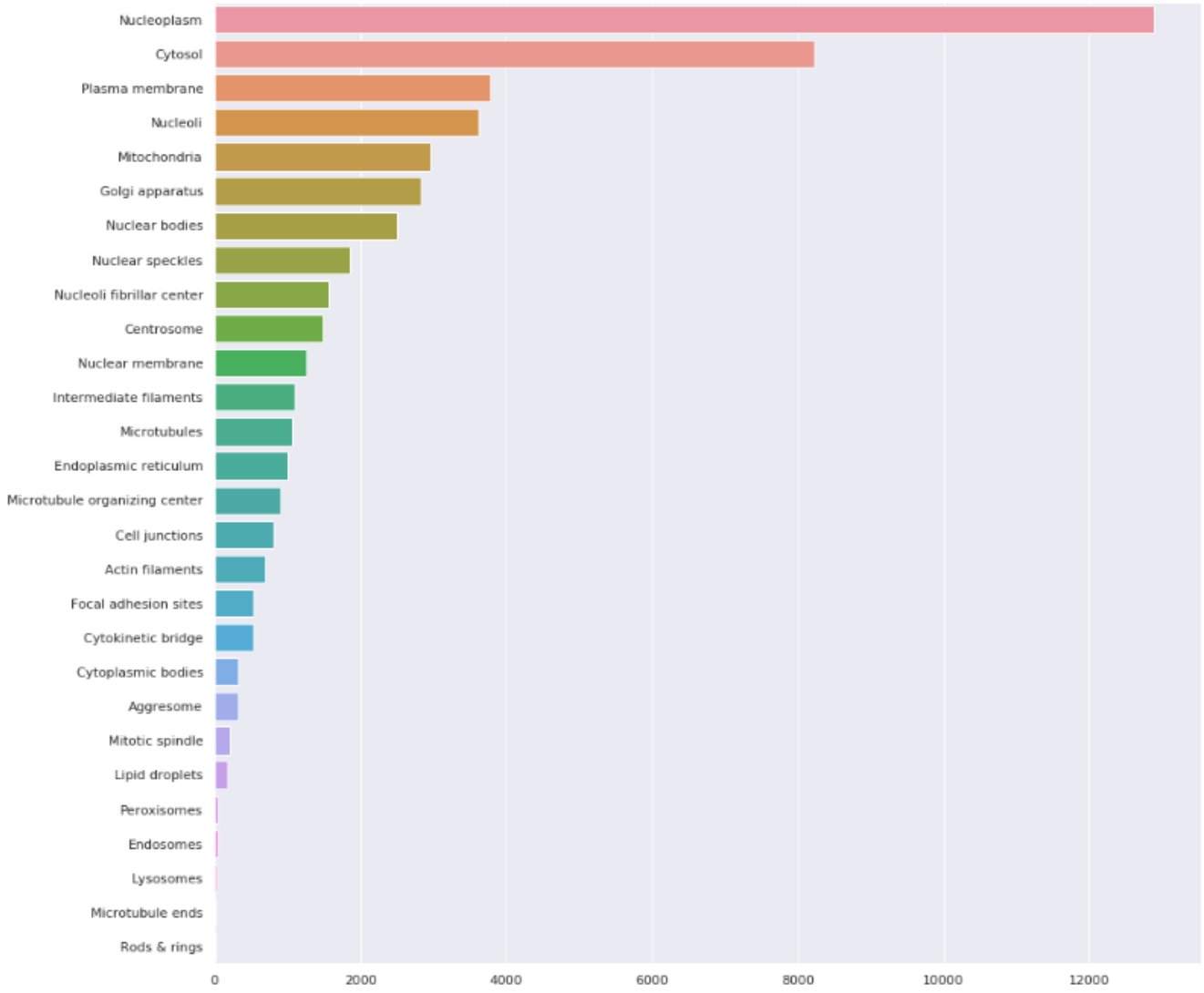
Since the proposed project is actually a kaggle competition, the benchmark will be the benchmark F1 score provided by kaggle for this competition for test set on **Leaderboard** which is **0.01955.**

And our model gives the final F1 score of 0.44936 and found stronger than the benchmark result.

Conclusion

Free-Form Visualization

Now, here is a visualization of which proteins occur most often in train images. This visualization shows that the data is highly imbalanced where some proteins occurring more often than the other ones and there are some classes that occurs as low as just once in the training dataset, that causes the problem of information loss when we try to remove some for validation of our model from training data. So there are few cases where some validation samples appears in validation set not



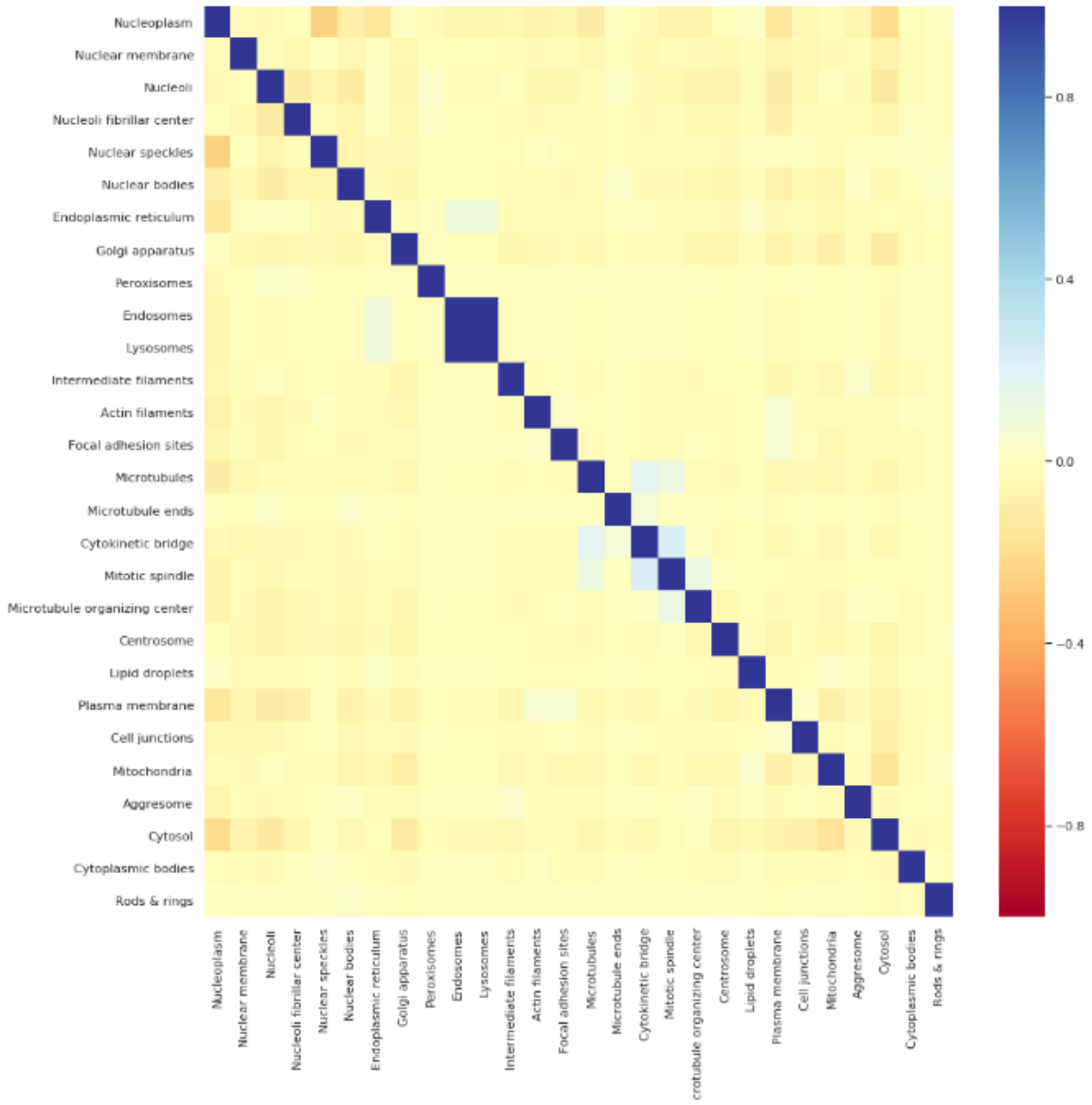
in the training set. We can see that most common protein structures belong to coarse grained cellular components like the plasma membrane, the cytosol and the nucleus. In contrast small components like the lipid droplets, peroxisomes, endosomes, lysosomes, microtubule ends, rods and rings are very seldom in our train data. For these classes the prediction will be very difficult as we have only a few examples that may not cover all variabilities and as our model probably will be

confused during ins learning process by the major classes. Due to this confusion we will make less accurate predictions on the minor classes.

So we can just make multiple copies of rare cases or can use some other oversampling techniques, but that may cause overfitting and the validation score might be misleading to because the exact same image may appear in both training and validation data.

Which targets are correlated?

Let's see if we find some correlations between our targets. This way we may already see that some proteins often come together.



- We can see that many targets only have very slight correlations.
- In contrast, endosomes and lysosomes often occur together and sometimes seem to be located at the endoplasmic reticulum.
- In addition we find that the mitotic spindle often comes together with the cytokinetic bridge. This makes sense as both are participants for cellular division. And in this process microtubules and their ends are active and participate as well. Consequently we find a positive correlation between these targets.

Reflection

The Goal of this project was to develop a model capable of classifying mixed patterns of proteins in microscope images. The labels for each training samples were provided in **.csv file,** which is common in kaggle competitions, there are multiple labels for each samples, because there are more than one proteins available in every images. There are four colour channels (Red, Green, Blue and yellow) for every sample and I preferred to use only green channel, because experimented with all colour channels and got best results with green channel images. In data exploration I found that the data is highly imbalanced because there are some labels that occur more often than others and even there are some cases with only one sample, So I tried over sampling but that did improve the model's performance. So i separated my training data into training and validation data and trained Resnet 34 model with only green colour channel with data augmentation, dropout and Gradient Descent With Restarts. Trained my model for 31 epocs, used F1 Score 'macro' as evaluation metrics and got good results with validation data.

And once I was satisfied with validation score I used whole training data (Training +validation data) with same hyperparameter to train my final model to make prediction for testing data.

Then I used test time Augmentation on test data so that our model can make better prediction by utilizing more data.

Improvement

As I have seen in public kaggle kernels for this projects the better results can be achieved if we use all 4 colour channels and combine them for and use that to train our model. But this creates additional challenges like 4-channel input to the model (RGBY), which is different from ones used in most of pretrained models (RGB input), training images will be large which would take more time and computational resources to train the model.

I would have considered using SMOTE (Synthetic Minority Over-sampling Technique) if you knew how just to see if the problem of imbalanced data can be avoided or can be made less effective.

References

1. Human Protein Atlas Image Classification.
   https://www.kaggle.com/c/human-protein-atlas-image-classification
2. F1 score.
   https://en.wikipedia.org/wiki/F1_score
3. kaggle kernel
   https://www.kaggle.com/iafoss/pretrained-resnet34-with-rgby-0-460-public-lb
4. Protein Atlas - Exploration and Baseline
   https://www.kaggle.com/allunia/protein-atlas-exploration-and-baseline
5. Decoding the ResNet architecture
   http://teleported.in/posts/decoding-resnet-architecture/