Classification of Subcellular Protein Localizations
Category: Computer Vision

Pradeep C Mylavarapu
pcm001@stanford.edu

03/15/2020

1 Introduction

Understanding how proteins are used in different types of cells in the human body, will help biomedical researchers better understand human cells & diseases. One of the challenges faced by biomedical researchers today is automating the classification of protein localization in various cells. Researchers typically identify protein localizations in cells through manual observation of microscope images. This is a challenging & time consuming task for researchers. Due to recent advancements in high throughput microscopy, the rate of generation of image data is much higher than the rate at which they can be manually classified. This gave rise to a need to automate the classification process. In this project I developed a deeplearning based classifier to classify protein localizations in the microscope images of different cells in the human body.

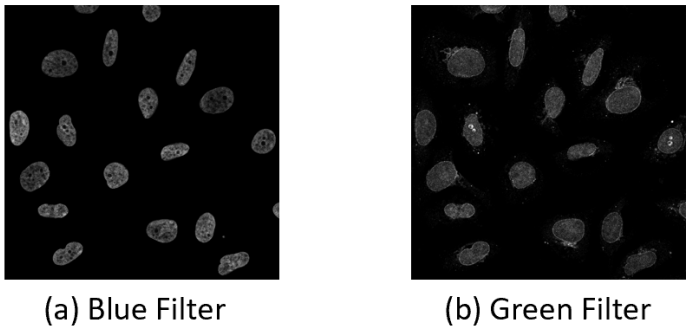
2 Related Work

For this project, the main point of reference is the research paper from Nature Methods "Analysis of the Human Protein Atlas Image Classification Competition". The paper was published in collaboration with the winners of the Kaggle contest. Most of the top performing teams used a variant of Resnet or Inception architecture for this classification task. One of the challenging aspects of this problem is that each image can correspond to multiple labels since proteins are simultaneously used in different parts of the cell. In addition the data is also highly imbalanced with some classes having very few training examples. This makes the task of identifying the protein localizations difficult even for human experts. Due to this reason, the metric that was used to determine the best performing model was macro F1 score. The macro F1 score was estimated to be close to 0.71 for human experts & some of the top performing models achieved a score in the range of 0.5 to 0.55.

3 Dataset

For this project, I used the dataset from the Kaggle Competition "Human Protein Atlas Image Classification". The dataset consists of 31,072 training examples. For each training example in the dataset, the input is set of 4 greyscale images of size 512x512 pixels. These are generated from post-processing microscope images using different filters. They are tagged as red/blue/green/yellow to indicate the filter that was used. The images generated using the green filter highlight the protein localizations while the images from other filters highlight different cell features. Hence as a starting point, the green-filter images were used as the input to the classification model.

The output is a vector of size 28 that corresponds to 28 unique classes. These classes indicate which part of the cell the proteins are localized in. Since proteins are simultaneously used in multiple areas within the cell, each training example can correspond to more than one class.



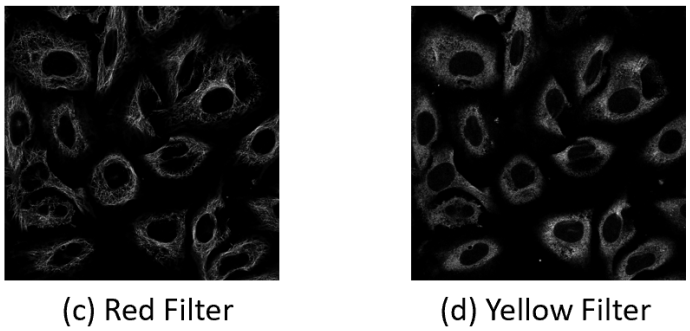


Figure 1: Greyscale images for a single training example. Generated using Blue/Green/Red/Yellow filters. The image from the Green filter (b) highlights the protein localizations

4 Architecture

4.1 Input Selection

Since this is an image classification problem, I decided on using a variant of Resnet architecture due to their superior performance on image classification tasks. I used a Resnet-50 implementation as a starting point. I used the green filter images from the dataset as inputs to the model since these images highlight the protein localizations. The deeplearning framework used for this implementation is Keras.

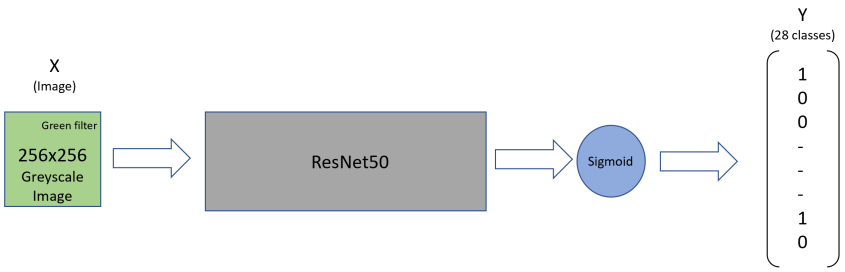


Figure 2: Initial Classifier implementation using only the green filter images as inputs. The input is a 256x256 pixel greyscale image and the output is a vector of size 28. Each input can correspond to more than 1 output class

4.2 Loss Function

Since this is a multi-label classification problem, I used a sigmoid activation for the output layer with binary-crossentropy loss function.

4.3 Evaluation Metrics

Out of the 31,072 examples, I used 24,000 images for the training set, 3,000 images for the cross-validation set & the remaining 4,072 images for the test set. To evaluate the performance of the model, I computed the precision/recall & Macro F1 score across training/dev/test sets for each of the models that were trained.

5 Evaluation/Results

5.1 Implementation Challenges

One of the initial challenges that I faced was handling the huge amount of data. Since it was not possible to convert the entire dataset into vectors and store them, I came up with an approach to convert batches of images into vectors & use them directly at runtime to train/evaluate the model. Despite this, I faced challenges in training the model because of the image size. Since 512x512 pixel images were too big, so I had to scale the images down to a smaller size.

For the initial implementation, I used Resnet50 implementation & used the green-filter images as inputs. For this model the input images were scaled down to 128x128 pixels. The performance on this model was very poor, with the macro F1 score staying close to 0.1.

Due to the poor performance, I changed the Resnet50 implementation by adding an additional Conv layer to the Identity block. I also changed the input layer to use 256x256 images and reduced the minibatch size to 16 to avoid running out of memory. This implementation improved the performance compared to the first model. But the observed macro F1 score was only around 0.16. To improve performance on rare classes, I used data augmentation techniques like random flipping, rotations, mirroring on the rare class images and added them to the training set. The performance only improved marginally.

5.2 Error Analysis

After performing error analysis on some of the examples that were being predicted incorrectly, I came to the conclusion that using just the green-filter images was not sufficient to make accurate predictions. The reason is for some training examples, there are very few details in the image making it difficult even for a human expert to make accurate predictions.

(a) Training Example 1 (b) Training Example 2

Figure 3: Green-filter images for 2 different training examples. These images hardly provide any useful information, making it difficult even for a human expert to predict

So I modified the architecture of the model to take a set of stacked images as input, instead of just 1 single image. For every training example, I stacked all 4 (green/blue/red/yellow filter) images and used them as input to the model. For this to work all 4 images need to be superimposed along the same axis. The reason for this approach is that even though the images from blue/red/yellow filters do not show the

protein localizations, they highlight various features of the cell. As an experiment, I first tried trained a new model with 3 images stacked together. Encouraged by the results I changed the model to use all 4 images for every training example.

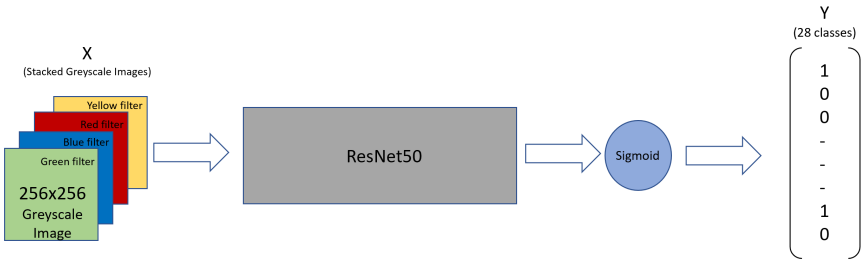


Figure 4: New Model implementation using all 4 (green/blue/red/yellow) images stacked together

Using stacked images significantly improved performance of the classifier. With data augmentation techniques such as random flipping, mirroring & rotations, the overall number of good predictions on several rare classes improved as well.

5.3 Results

The overall Macro F1 score steadily improved across the different models, with the stacked-image model achieving the best performance. The F1 score on the final implementation is around 0.33.

Out of the 28 classes the models performed very well on the classes Nucleoplasm(Class-0) & Cytosol(Class-25) for which there was a large amount of data available. The performance on these classes steadily improved from the single image model to the stacked-image model.

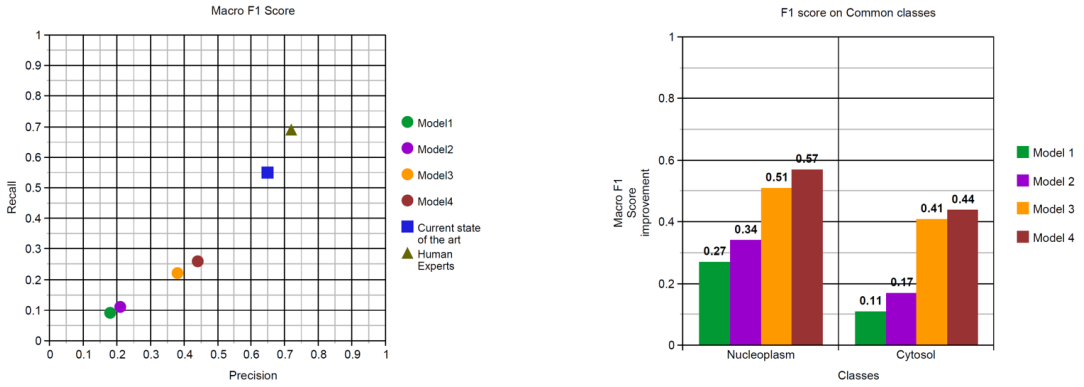


Figure 5: Macro F1 score Improvements across the 4 models that I tried. Model 1 used 128x128 Green-filter Images. Model 2 is the enhanced Resnet50 with 256x256 Green-filter Images as inputs. Model 3 used 3 Images (Green/Blue/Red Filters) stacked together. Model 4 used all 4 images (Green/Blue/Red/Yellow Filters) stacked together

In addition when I moved to the stacked-image model, the number of accurate predictions went up significantly on classes like Nucleoli (Class-2), Golgi Complex(Class-7) and Intermediate Filaments (Class-11). There were marginal improvements seen in classes like Nucleoli fibrillar center (Class-3), Nuclear speckles (Class-4), Microtubule organizing center (Class-18), Centrosome (Class-19).

Data augmentation led to significant improvements for the classes Microtubule organizing center (Class-18), Mitochondria (Class-23). Prior to training with augmented data, the number of accurate predictions on these classes was close to 0 on the test set.

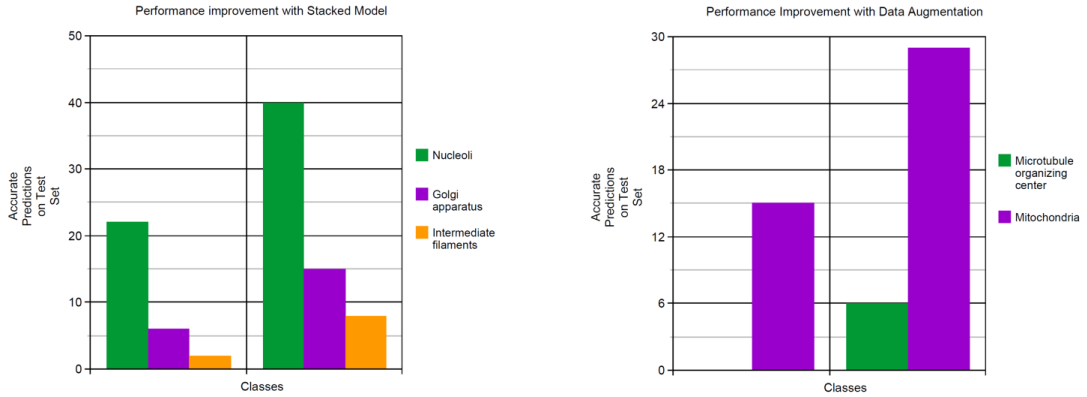


Figure 6: Improvements in number of accurate predictions with Stacked-model & Data augmentation

For some of the rare classes such as Endosomes (Class-9), Lysosomes (Class-10), Rods & rings (Class-27), none of the models were able to make any accurate predictions on the test set. This was mainly due to lack of availability of data. The training set contained less than 30 images for these classes. Gathering more data is likely to help with performance improvement for these 3 classes.

6 Conclusions & Future Work

Moving from a single-image model to a stacked-image model resulted in a significant increase in terms of performance. Overall the models performed very well on the common classes & struggled on rare classes that had very less data. One approach to improving the performance of the models would be to collect more data on the rare classes. Another possible experiment would be to try to use transfer learning by using a pretrained model as a starting point.

References

[1] Wei Ouyang et al. Analysis of the human protein atlas image classification competition. *Nature Methods*, 16(10):1254–1261, 2019.

[2] Kaiming He et al. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[3] BAOQI LI et al. An improved resnet based on the adjustable shortcut connections. *2018 IEEE Access*, pages 18967–18974, 2018.