Kaggle competition: Classify subcellular protein patterns in human cells

https://github.com/chisomama/Kaggle-Protein-Competition

Mohamed El Hhibouri
UNI: me2641
m.elhibouri@columbia.edu

Chisom Amalunweze
UNI: jca2158
Chisom.amalunweze@columbia.edu

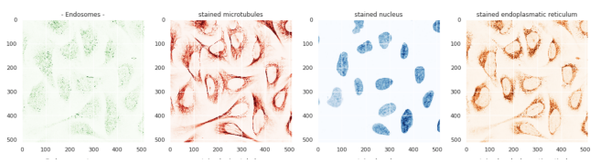
Ethan Furstoss
UNI: ef2603
ef2603@columbia.edu

1. Introduction

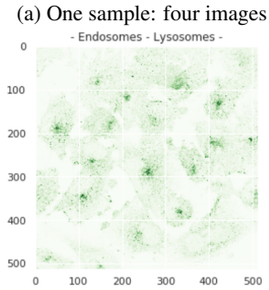
The problem at hand is the classification of proteins into their organelle localization labels. We will be differentiating between 28 different cell types and the input data will be microscopic images of the proteins of interest. The overall purpose of the project is for the submission into the Human Protein Atlas Image Classification Kaggle. Models submitted will contribute to The Human Protein Atlas; a smart microscopy system of identifying a proteins location from a high-throughput image. The goal of the project is to improve upon the baseline all classes implementation for the current protein classification given on the Kaggle competition website. [1]

2. Dataset and Problem Formulation

There are 31,072 unique entries in the data set. The entries are labeled with ids that represent their targeted labels for the given image. There are 27 label with varying frequencies; with the most frequent being Nucleoplasm, being present in 12885 of the entries. 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). There are huge discrepancies between the most frequent label occurrences and the least frequent label occurrences. With the top three highest frequency label occurring as many times as the rest of the data combined. For this reason, we intend to use statistical sampling methods to carry out research on 10% of the dataset at a time. Co labeling occurs, but roughly half of the data consists of a single label for the image, and the large majority of the rest just have two labels for a given image. There does seem to be a slight correlation amongst some sub-cellular components, such as the endosomes and lysosomes; and this doesn't come as much surprise as the two look very similar.



(a) One sample: four images



(b) Used to predict the label

Due to the high resolution of the images, we must compress it to be able to feed it into our networks and run at a reasonable time. We also intend to sample it in order to train our hyperparameters sufficiently. Our innovation, in dealing with compressed sampled versions of our data was to use data augmentation methods to enhance the usability of the data that we had. Upon developing a satisfactory model, we hope to be able to transfer the same weights and parameters to the full dataset and achieve the same results.

2.1. Baseline model and milestone

Our chosen baseline model [3] achieves an F1 score of 0.448 and is currently ranked 52nd in the leaderboard. It implements a light Resnet on low resolution versions of the dataset images. An F1 score of 0.448 is achieved upon running 100 epochs of training on the model. We intend to beat the baseline and achieve a beginning score of 0.5 by running a combination of ResNets in series and in parallel on high resolution images. Additionally, with advanced data augmentation, we believe that we can train for more epochs on more data, thus improving our scores with more complex, but faster models. After achieving this first milestone, we will be looking at other deep learning classification algorithms that we can add to the existing ResNets to improve our score.

2.2. Evaluation metrics

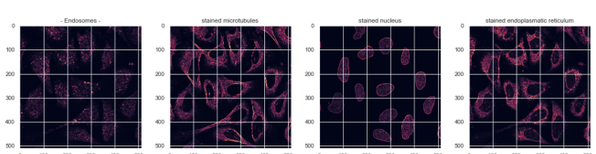
The primary evaluation metric of the Kaggle competition is the macro F1 score. As stated earlier, our first milestone is achieving a score of at least 0.5 on our model. The best loss function would be, of course the metric itself. However the macro is non-differentiable. Thus instead of using binary prediction, we will use probabilities in order to have a continuous loss function. Then if we have a probability p concerning 1 label, we"ll get p true positive and 1-p false negative. If we have a probability p concerning 0 label, we"ll get p true negative and 1-p false positive. Also, a good trick could be to modify this loss function by rounding the value of probability and using it as an activation function for the last layer as proposed in comment in [4].

3. Features augmentation

To reduce computation time, we carried out the below processes using multiple workers running in parallel.

3.1. Normed gradient using Sobel kernel

By convolving with a Sobel kernel our images, we were able to detect edges on the 4 different type of images as the following pictures. Furthermore, by comparing different types of proteins present in images, we noticed that some particular shapes are clearly visible and allow to differentiate labels by eyes.



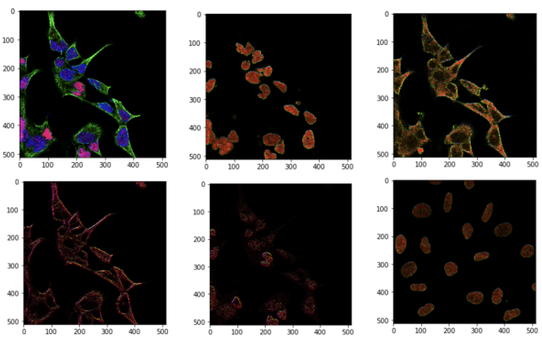
Edges detection using Sobel kernels

3.2. Directional gradient

As each types of cells have different shape, we can extract a great deal of information by using different gradient orientation. Indeed, this gives information about edges in multiple

direction and then differentiated cells. Also, keeping pixels values only above 0 cleans the pictures and keeps only the edges of cells;

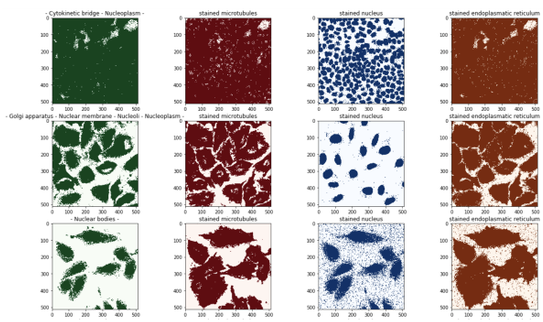
In order to keep a repeatable pipeline, we concatenate those features augmentation as new channels: from (512,512, 4) to (512,512, 92). By plotting them in RBG, visually it is easy to see different combinations of channels and how different they are:



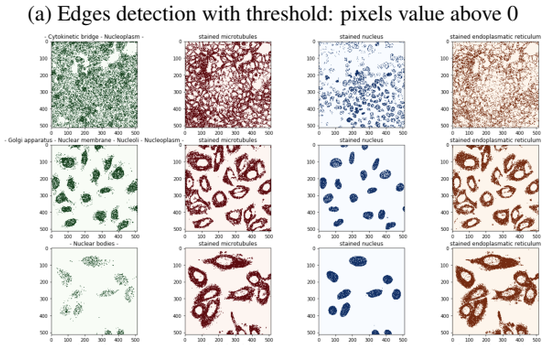
Multiple RBG channels combinaison

3.3. Normed gradient using Sobel kernel and threshold

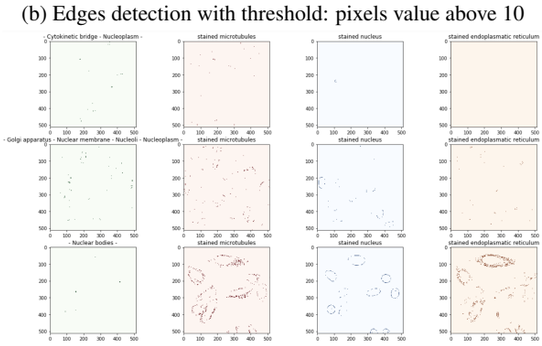
In order to detect those different shapes, we then add a lower bound threshold to detect if some objects could have outstanding pixels values. Indeed, for some particular labels like 'Endoplasmic reticulum', 'Nuclear bodies', and 'Nucleoli', we can observe on nucleus images that there are some particular colored pixels with different density repartition that are not visible for other labels. This is illustrated on the following pictures:



(a) Edges detection with threshold: pixels value above 0



(b) Edges detection with threshold: pixels value above 10

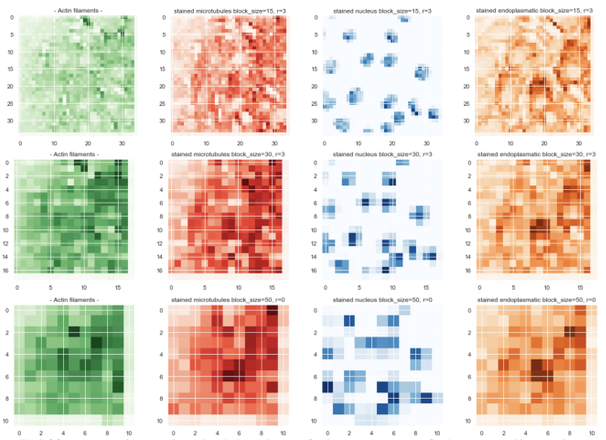


(c) Edges detection with threshold: pixels value above 90

4. Future features engineering

4.1. Blocks

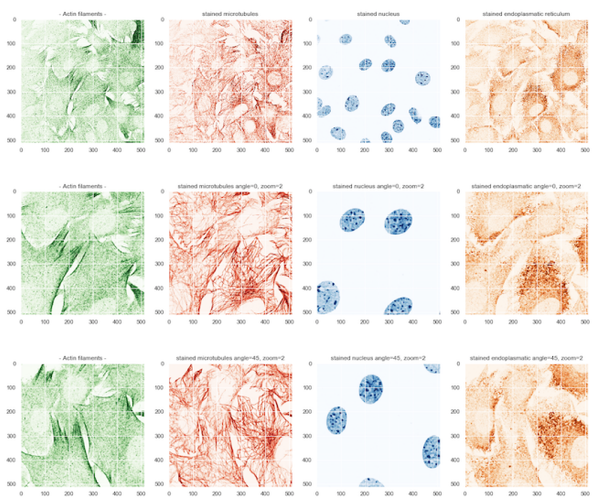
Using blocs with different size and orientation of gradient, we are able to reduce the size of images but also keeping information about images. We want to explore this type of preprocessing and find out if this type of data can keep enough information to have similar macro F1 score. The most interesting would be to improve the speed of training as well as prediction (as asking in the Kaggle competition) by keeping similar accuracy. By using a cross validation on a smaller data set, we will obtain the impact of the size block as well as the number of gradient orientations.



Different size pixels blocks of the mean of the gradient in multiple direction

5. Zoom to target particular cells

Something that could be interesting is to see where the information is gathering. By zooming on one cell and then reduce the image size, we might be able to detect proteins without losing information and thus improve the process speed.



Zoom x3 and rotation 45deg and 90deg

We want to select the best features preprocessing (channels) that maximise the accuracy on each labels. One possibility will be to group labels according to their ?geographical? position (every labels that concerns the nucleus will be predicted with the third channels in blue page 1 for example) and merge the results of each model to one vector. Also, this process can be repeated with different architectures.

6. Baseline model improvement

We took a three pronged approach to solving this problem, opting to build in parallel and then combine our results in an argmax model to achieve a high score in the kaggle competition. Our first task was focused on optimizing our baseline model. The baseline architecture is a ResNet-34

being implemented with fastai. The specific loss function utilized with this model is the focal loss. The focal loss function is designed to be a proper loss function for problems that involves large data imbalances. With 28 label categories and a single label, Nucleoplasm, appearing in over $\frac{1}{3}$ of the data entries, it's helps greatly to have a loss function that is geared towards these kinds of data imbalances. The focal loss essentially does this by focusing on the sparse set of less frequently occurring data and doesn't allow the well-classified examples to overweight the loss assigned to them[2].
Since a 4-channel input of RGBY is being utilized for a pre-trained model accustomed to the normal 3-channel RGB input, the first convolutional layer had to be dropped and replaced with a (7,7,4) to (7,7,64) rather than the previous (7,7,3) to (7,7,64). The optimizer for the model used is the Adam Optimizer. The learning rate utilized was 0.02. This was found by running training with different learning rates, recording the losses, and finding the learning rate that appears to bring the losses to a minimum. The unique thresholds are utilized for each individual class in order to maximize the accuracy score.

7. U-net

We implemented a U-net architecture on a sample set of the data. This was based on a recommendation from our project proposal and is derived from the paper by Ronneberger et al [4]. The advantage of this model is that it is able to identify localized labels within segments of a single image and work with smaller overall datasets. One difference in our implementation is that while the paper relies heavily on data augmentation to supplement its sparse amount of data, we are able to carry out a robust network with a sampling of our training dataset. The next thing we focused on was data and features augmentation. To reduce computation time, we carried out the below processes using multiple workers running in parallel.

8. Linking Augmentation to Model Improvement

Once we will find the best hyperparameters of our features' augmentation, we will augment the size of our data set. As cells images orientation does not bring any information, we can use multiple symmetry and rotations to obtain more data to train. We also will be optimizing hyperparameters in our Resnet baseline as well as our Unet implementation. It is our hope that all three will be greater than the sum of their parts.

Once we will find the best hyper-parameters of our features' augmentation, we will augment the size of our data set. As cells images orientation does not bring any information, we can use multiple symmetry and rotations to obtain more data to train.

9. Potential Issues identification

As the frequency of some labels like Nucleoplasm and Cytosol are high compared to others, this will lead to reduce our F1 score by having a model that predict too often the same labels. We have exploration left to do:
- measure the impact of using a Gaussian noise on images to reduce the impact of the high frequency labels apparitions
- find the best probability threshold for each label
- analyze the recall and precision score separately for each label according to the type of features used
- use different models with different inputs might lead to better results for different labels
- use multiple models with different architecture, analyze their recall and precision, and merge results that lead to the best F1-score

10. Previous work and references

[1] *NikitPatell* "Best Tutorial for Beginner"
https://www.kaggle.com/nikitpatel/best-tutorial-for-beginner
Accessed Oct 25 2018
[2] *lafoss* "Pretrained ResNet with RGBY"
https://www.kaggle.com/iafoss/pretrained-resnet34-withrgby-0-448-public-lb
Accessed Oct 25 2018
[3] *Tsung-Yi Lin et Al* "Focal Loss for Dense Object Detection"
https://arxiv.org/pdf/1708.02002.pdf
Accessed Nov 1 2018
[4] *O. Ronneberger and P.Fischer and T. Brox* "U-Net: Convolutional Networks for Biomedical Image Segmentation"
https://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a/
Accessed Nov 1 2018
[5] *NikitPatell* IEtol "Protein where art thou?"
https://www.kaggle.com/weegee/protein-where-art-thou
Accessed Oct 25 2018
[6] *Michal HaltufBest* "Loss function for F1-score metric"
https://www.kaggle.com/rejpalcz/best-loss-function-forf1-score-metric
Accessed Oct 19 2018
[7] *NikitPatell* "Four Img Combine in single img VGGModel"
https://www.kaggle.com/nikitpatel/four-img-combine-insingle-img-vgg-model
Accessed Oct 11 2018
[8] *Kevin Mader* "Transfer Learning for Human Protein Submission"
https://www.kaggle.com/kmader/transfer-learning-forhuman-protein-submission
Accessed Oct 4 2018