IST-718: Final Project

Human Protein Atlas Single Cell Classifier

Team: SriHari Busam, Sharat Sripada

Code repository: https://github.com/srihari-busam/hpa-deep-learning

Introduction

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*.

Advances in high-resolution microscopy has made it possible to generate images of cells comprising proteins at greater pace and volume than those generated manually. As such, the need is greater than ever to automate biomedical image analysis and classification

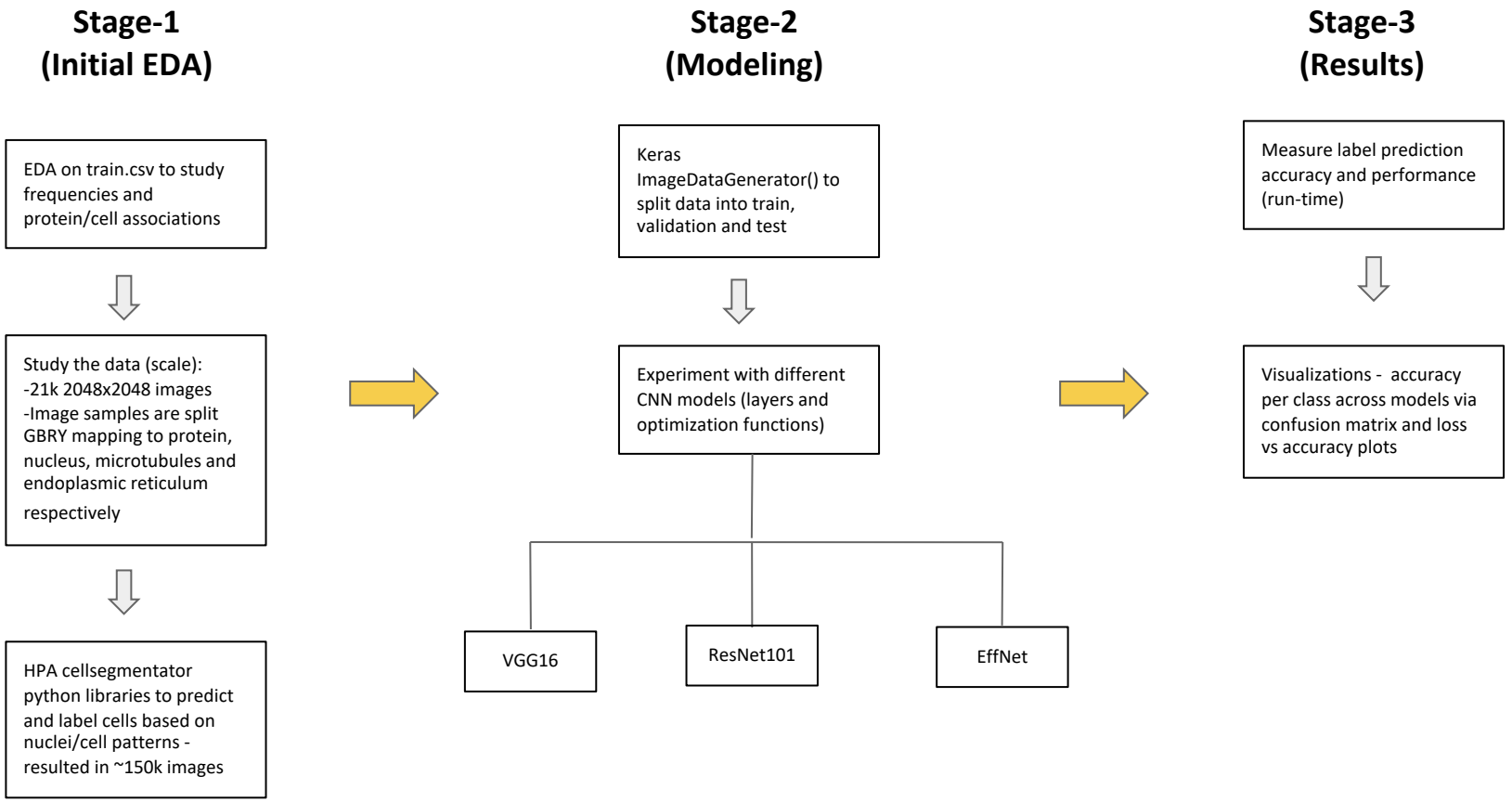
Kaggle competition:
https://www.kaggle.com/c/hpa-single-cell-image-classification/overview

Problem Statement
To develop models capable of classifying mixed patterns of proteins in microscopic images

Goals
- To hit prediction accuracy on the validation set >44% (Kaggle Leaderboard at the time we started work on this project)
- Experiment and build different convnets and weigh them based on Key Performance Indices (KPIs) like *Accuracy* and *Time* to compile/build the model

Classes of proteins
0. Nucleoplasm
1. Nuclear membrane
2. Nucleoli
3. Nucleoli fibrillar center
4. Nuclear speckles
5. Nuclear bodies
6. Endoplasmic reticulum
7. Golgi apparatus
8. Intermediate filaments
9. Actin filaments
10. Microtubules
11. Mitotic spindle
12. Centrosome
13. Plasma membrane
14. Mitochondria
15. Aggresome
16. Cytosol
17. Vesicles and punctate cytosolic patterns
18. Negative

High-level project Pipeline



HPA CELL SEGMENTATION LIBRARY

The *cellsegmentator* library has the following utilities:

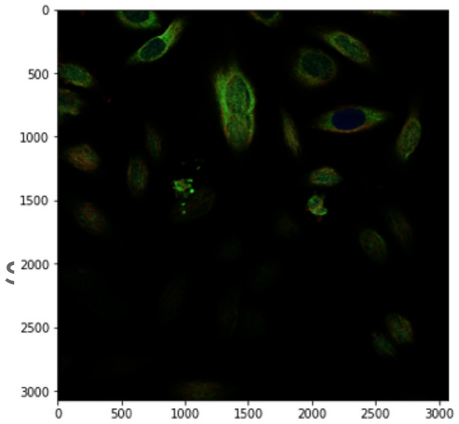
- **pred_nucluei:**

  takes a list of image arrays (nuclei in third channel - blue) and returns list of neural network outputs of nuclei segmentation
- **pred_cells:**

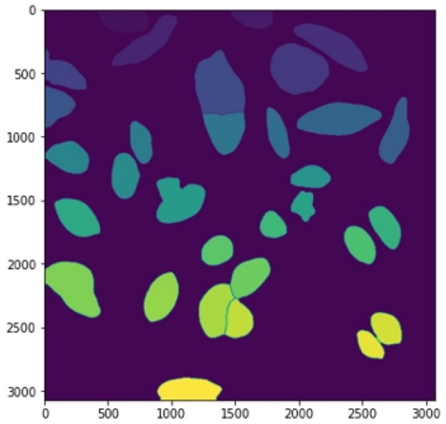
  takes a list of images (microtubules, endoplasmic reticulum, and nuclei in that order) and returns list of outputs of cell segmentation
- **label_nucluei:**

  takes nuclei prediction for an image and returns the labeled nuclei mask array
- **label_cells:**

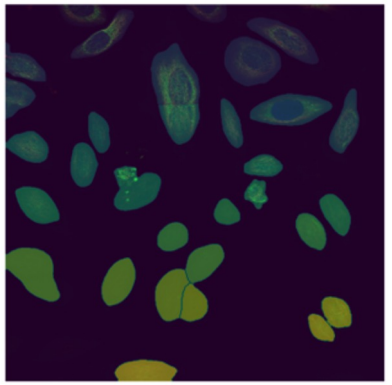
  takes nuclei and cell prediction for an image and returns the labeled nuclei and cell mask arrays as a tuple



Original Image



Mask generated by cell segmentation pred_cells()



Resulting Image - Mask superimposed over Original Image

About the data

- Training dataset - 21,806 samples (x4 - Red, Green, Blue, Yellow)
- Each image is 2048 X 2048 resolution
- Data posed following challenges
  - Weak labels
    - Images has more than one cell and the label only tells what types of proteins/cells are present

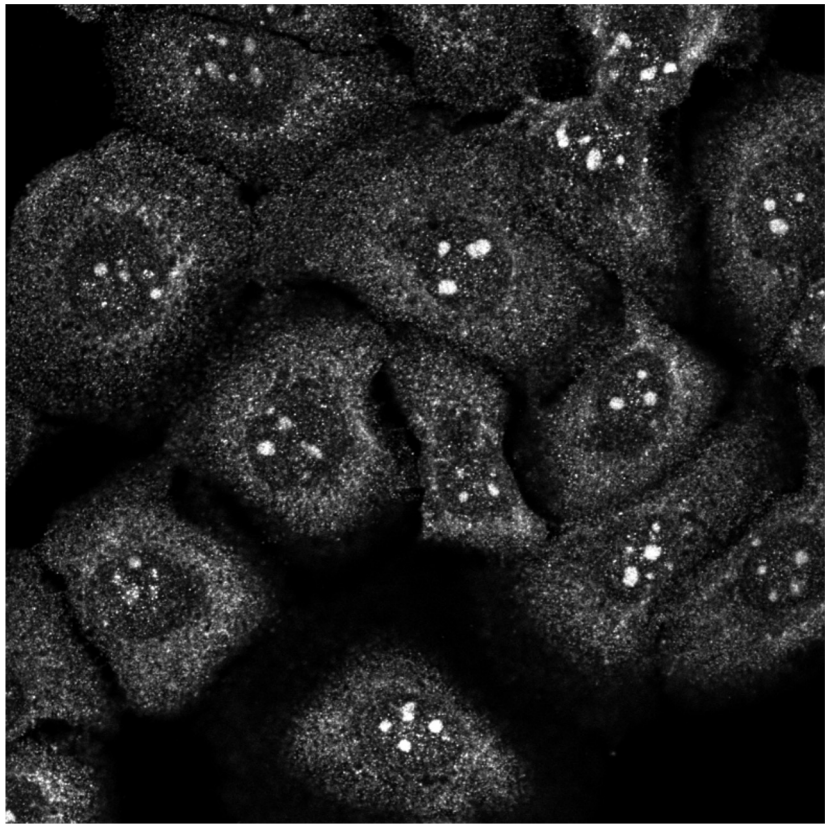


Image has 3 types of cells: Nucleoli, Mitochondria, Cytosol

ImageId: 0fe26432-bb9a-11e8-b2b9-ac1f6b6435d0

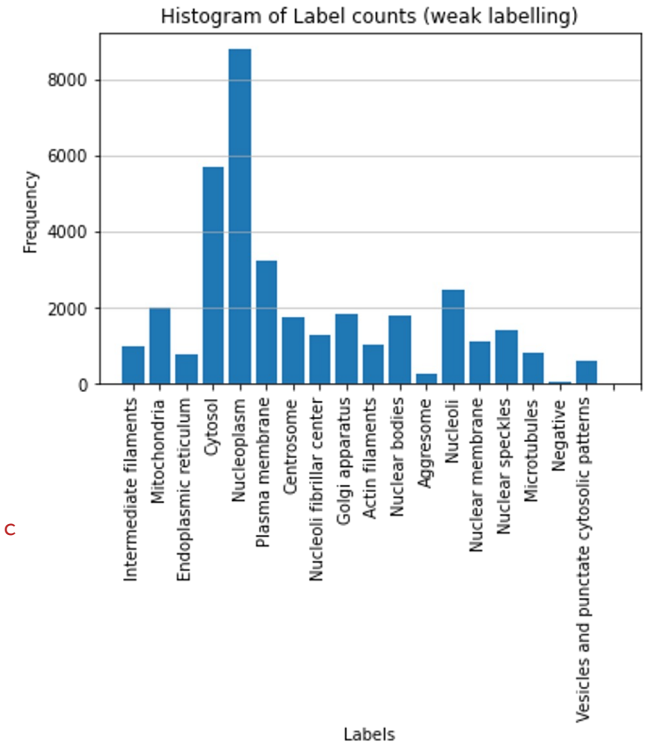
Exploratory Data Analysis (on weak labels/train.csv)

- Initial EDA was on train.csv comprising weak labels linked to each image
- Sample of transformed data shown below where an image may have upto 5 labels

  NOTE: 19 is a custom label showing no data
- Histogram showing the distribution of various proteins/other cell components across 21807 images

	ID	Label-1	Label-2	Label-3	Label-4	Label-5
1	5c27f04c-bb99-11e8-b2b9-ac1f6b6435d0	8	5	0	19	19
2	5fb643ee-bb99-11e8-b2b9-ac1f6b6435d0	14	0	19	19	19
3	60b57878-bb99-11e8-b2b9-ac1f6b6435d0	6	1	19	19	19
4	5c1a898e-bb99-11e8-b2b9-ac1f6b6435d0	16	10	19	19	19
5	5b931256-bb99-11e8-b2b9-ac1f6b6435d0	14	0	19	19	19

```
Classes of proteins
    0: 'Nucleoplasm',
    1: 'Nuclear membrane',
    2: 'Nucleoli',
    3: 'Nucleoli fibrillar center',
    4: 'Nuclear speckles',
    5: 'Nuclear bodies',
    6: 'Endoplasmic reticulum',
    7: 'Golgi apparatus',
    8: 'Intermediate filaments',
    9: 'Actin filaments',
    10: 'Microtubules',
    11: 'Mitotic spindle',
    12: 'Centrosome',
    13: 'Plasma membrane',
    14: 'Mitochondria',
    15: 'Aggresome',
    16: 'Cytosol',
    17: 'Vesicles and punctate cytosolic patterns',
    18: 'Negative'
```



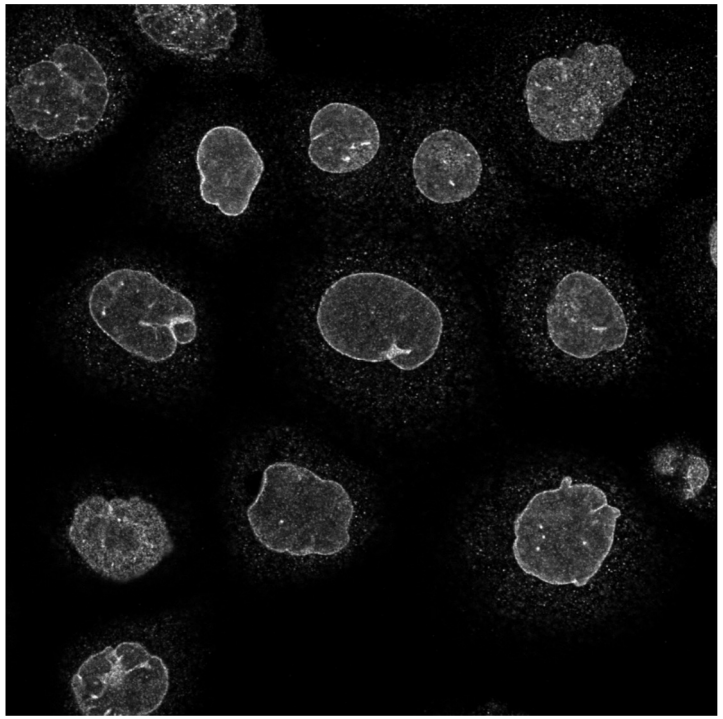
Exploratory Data Analysis (on weak labels/train.csv)

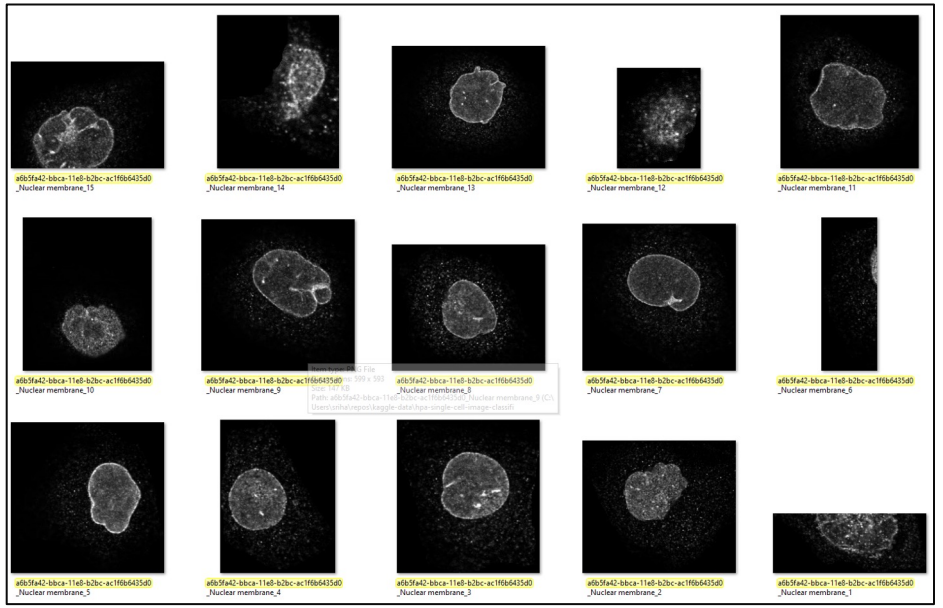
- Finally, we were interested in understanding proteins or other cell components that occurred frequently using the Apriori algorithm/Associated Rule Mining technique
- The table alongside shows relationships - highlighted ones relatively with high confidence while others in 25-30%

	Association	Support	Confidence	Lift
0	Microtubules -> Mitotic spindle	0.001009	0.282051	7.518839
1	Nucleoplasm -> Cytosol -> Mitotic spindle	0.000780	0.566667	2.173568
2	Nucleoplasm -> Mitochondria -> Vesicles and pu...	0.000642	0.875000	2.168950
3	Nuclear bodies -> Nucleoplasm -> Vesicles and ...	0.000459	0.833333	2.065666
4	Microtubules -> Mitotic spindle -> None	0.001009	0.282051	7.518839
5	Nucleoli -> Mitochondria -> Plasma membrane	0.000917	0.259740	2.310851
6	Nucleoli -> Cytosol -> Endoplasmic reticulum	0.001513	0.268293	2.386940
7	Nucleoplasm -> Cytosol -> Mitotic spindle -> None	0.000780	0.566667	2.174715
8	Nucleoplasm -> Mitochondria -> None -> Vesicle...	0.000642	0.875000	2.169443
9	Nucleoli -> Nucleoplasm -> Cytosol -> Intermed...	0.000780	0.265625	2.403410
10	Nuclear bodies -> Nucleoplasm -> Vesicles and ...	0.000459	0.833333	2.066136
11	Nucleoli -> Mitochondria -> Plasma membrane ->...	0.000917	0.259740	2.310851
12	Nucleoli -> Endoplasmic reticulum -> Cytosol -...	0.001513	0.268293	2.386940
13	Intermediate filaments -> Cytosol -> Nucleopla...	0.000780	0.265625	2.405406

Solving the data challenge

- The idea was to convert weak labels to concrete labeled data
- Approach
  - Identified images that only has one class. There are about **10,412** images with single cell type only
  - Used the previous competition cell segmentation model to separate them into individual cell images(python magic used to isolate the images)
  - This approach resulted in **151,707** strong labeled samples
  - It took **~2 days** to convert the data!!





Imageid shown: a6b5fa42-bbca-11e8-b2bc-ac1f6b6435d0_**Nuclear membrane**

Data cleanup & scoping for modeling

- After conversion removed all files <4kb and >1MB
  - Reasoning:
    - Small size - may not have enough features
    - Large size - compression may introduce artifacts
- Scoped to 12 classes only
  - When the experiments started Kaggle has about ~40% as the top. We want to hit atleast 50% accuracy and one way we thought is to have enough samples to train.
  - Chosen classes with more than 1k samples
  - “Negative” class images are removed as these could be any cell image and could be a noise

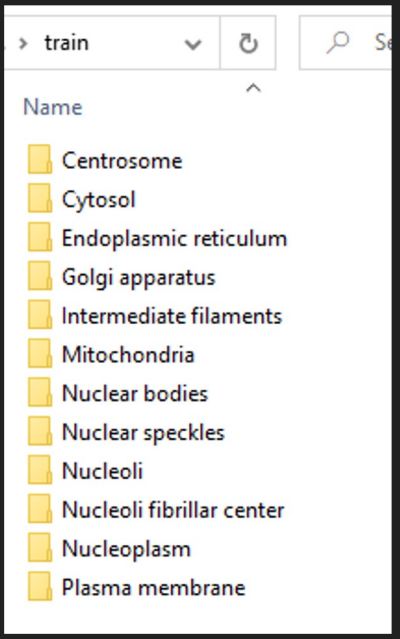
Setting up the data

- Initial experiment was done using manual image manipulation
- Later we leveraged keras image generator to do all the image manipulation heavy lifting
- Leveraged "**flow_from_directory**" to feed training and validation data with image resize to **528 X 528** (Actual EfficientNetB6 is built on the specific resolution)

Image manipulation

```
# image generator for train and validation.
# For trainin and test image manipulation para
gen = ImageDataGenerator(
  rotation_range=40,
  width_shift_range=0.2,
  height_shift_range=0.2,
  shear_range=0.2,
  zoom_range=0.3,
  horizontal_flip=True,
  vertical_flip=True,
  preprocessing_function=preprocess_input
)
```

Setup for "flow_from_directory"

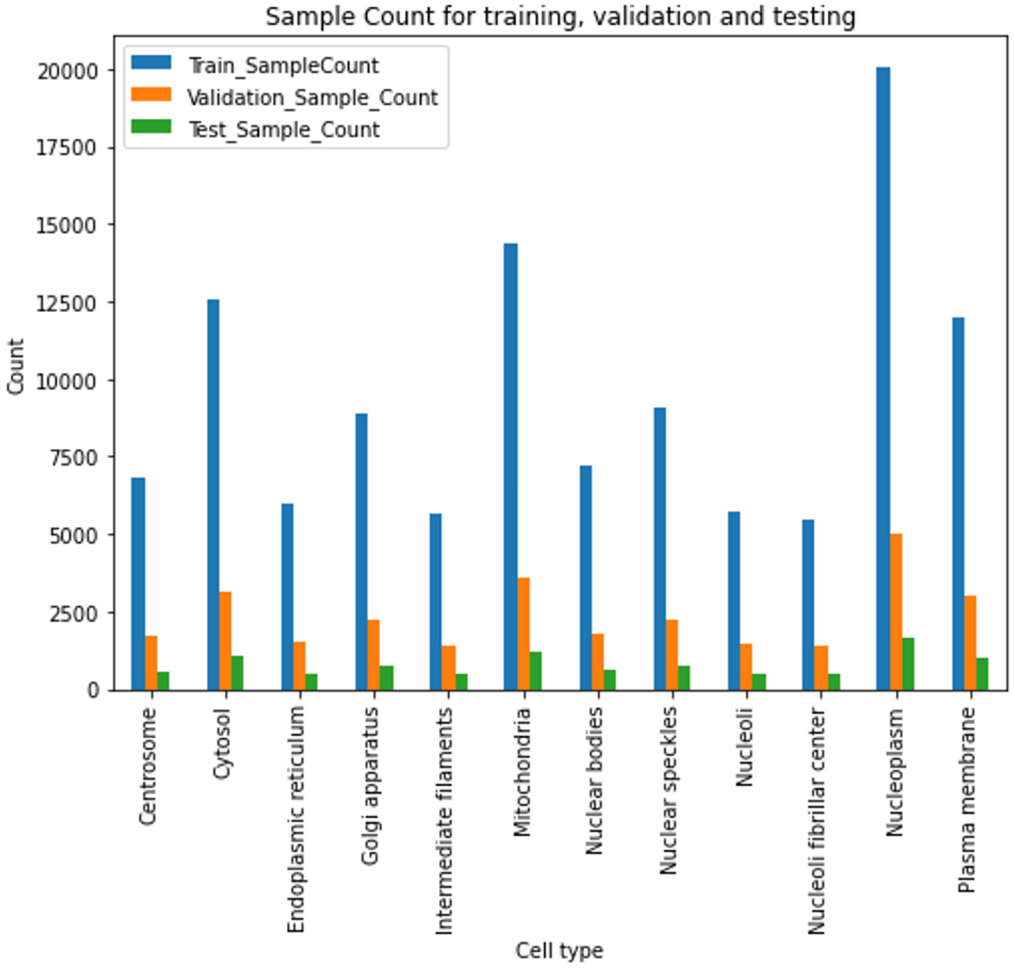


Stratified samples details

Total concrete labeled used for the analysis: 151,707 images

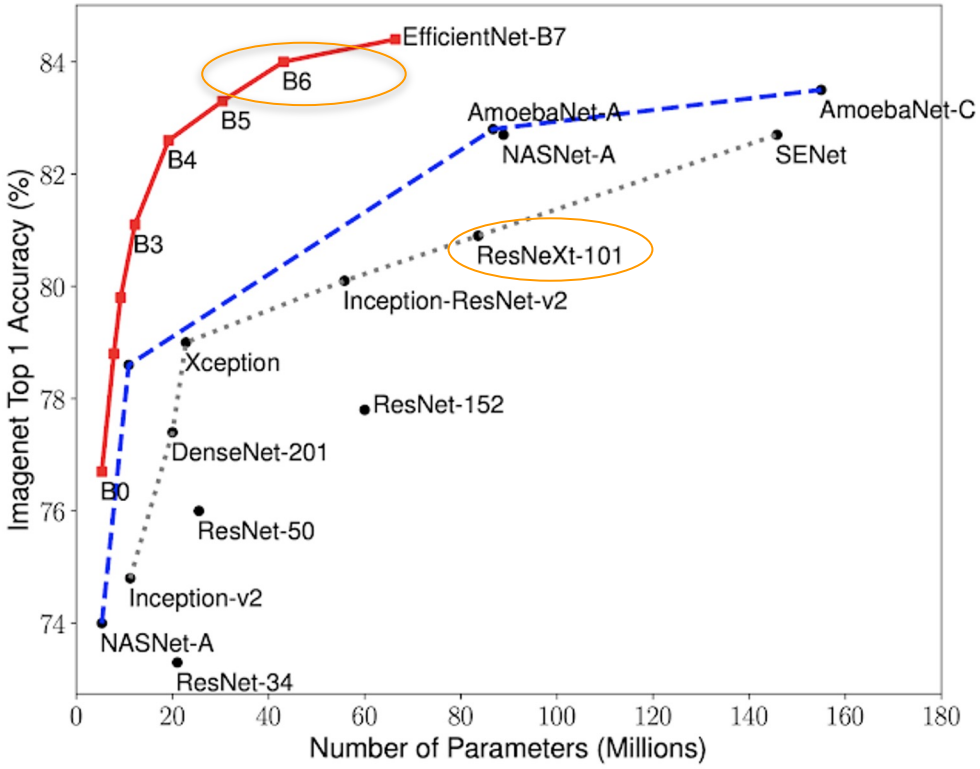
Samples are divided into training, validation and testing as follows

- Total training samples: 113,780
- Total validation samples: 28,445
- Total test samples: 9,482



Introduction to CONVNET

- Considered the following models:
  - VGG16
  - ResNet101
  - EfficientNetB6
- Python libraries
  - *PyTorch* for segmentation of images to individual cells
  - *Keras with Tensorflow 2.4* backend for all the model building
- Hardware/GPU
  - CPU: AMD 5800X 8C 16Threads with 128 GB RAM
  - GPU: NVIDIA RTX 3090 with 24GB RAM

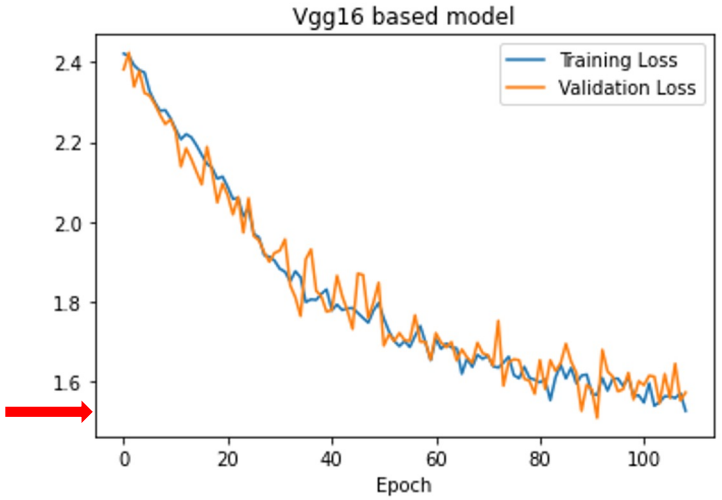


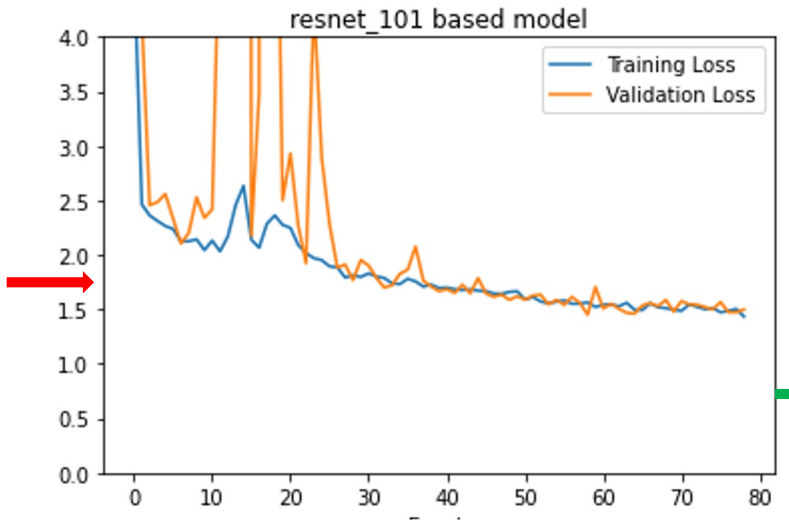
EfficientNet performance compared to other models on Imagenet dataset

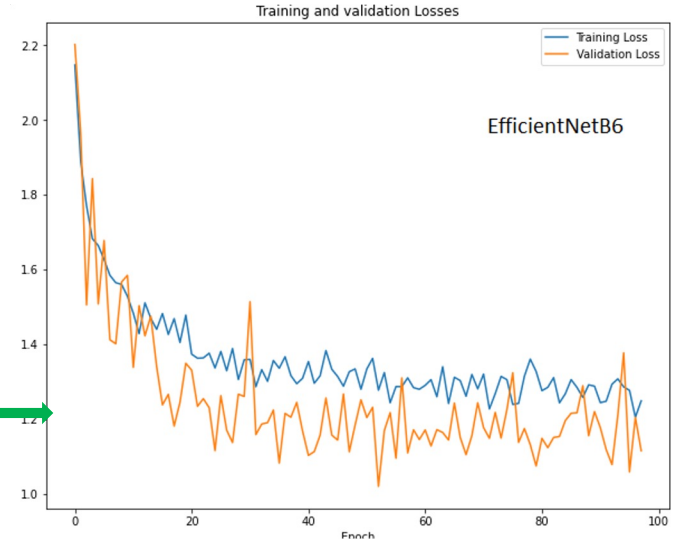
Convnet models - experiments & results

Model based on	Input Details	Trainable params	Optimizer	Training Time	Training Accuracy	Validation Accuracy
VGG16 16 layers Weights: None Weights locked : NO	Res:512 X 512 Batchsize: 32 Steps per epoch:100 Validation steps: 20	14,714,688	Adam (lr=0.0001)	~5hrs	**46.59%**	**48.59%**
Resnet101 101 layers Weights: imagenet Weights locked : NO	Res:512 X 512 Batchsize: 24 Steps per epoch:120 Validation steps: 30	48,844,300	NAdam (lr=0.0001)	~5hrs	**49.78%**	**51.94%**
EfficientNetB6 Weights: imagenet Weights locked : NO	Res:512 X 512 Batchsize: 7 Steps per epoch:150 Validation steps: 10	40,763,364	Nadam (lr=0.0001) (tried Ada, rmsprop,SGD)	~8hrs	**70.08%**	**67.14%**

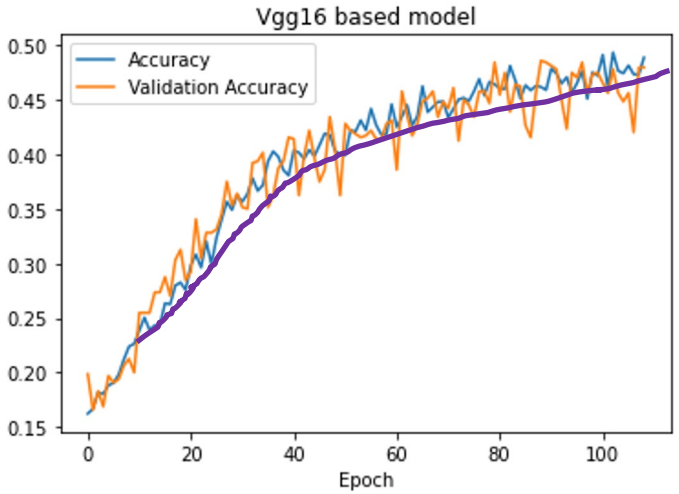
Loss & Accuracy

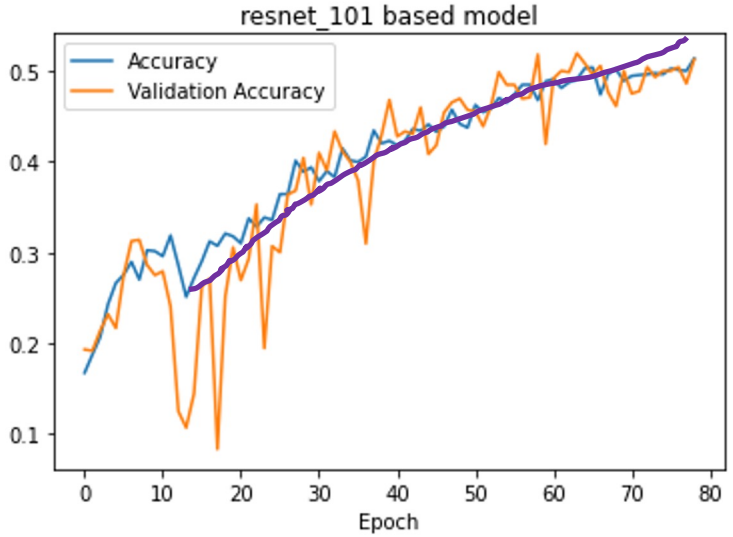


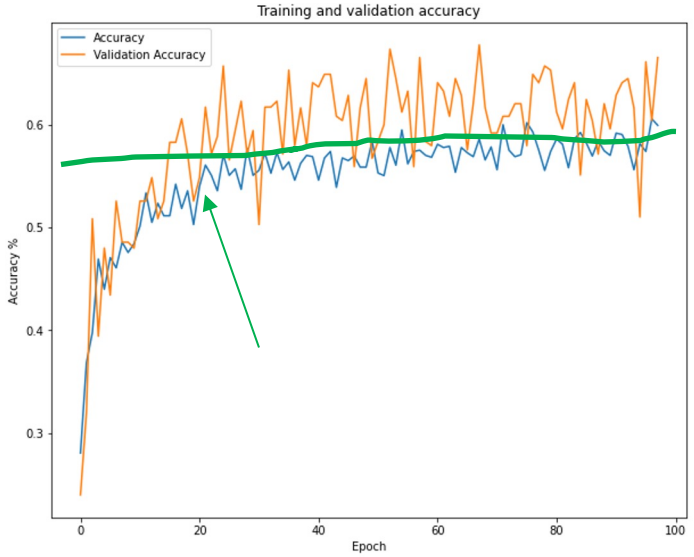




Accuracy







Epoch

Confusion matrix(actual vs predicted %) – VGG

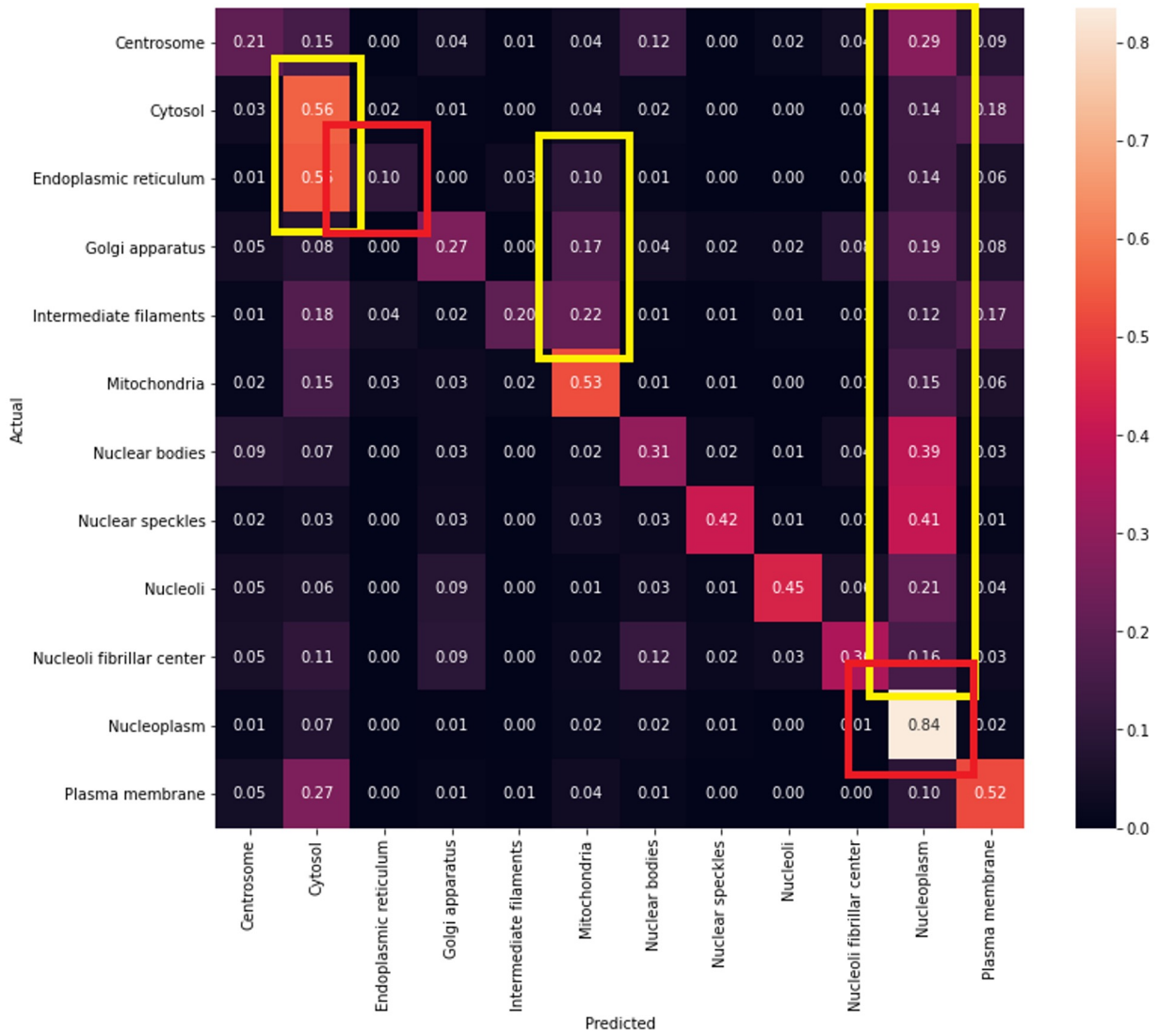
Validation samples: 28,445
Overall Validation Accuracy: 48.59%

Test samples : 9,482

Highest Accuracy:
- Nucleoplasm (84%)

Lowest Accuracy:
- Endoplasmic reticulum – 10%
- Intermediate filaments – 20%

Top mis classification:
- Cytosol
- Mitochondria
- Nucleoplasm



Confusion matrix(actual vs predicted %) – Resnet

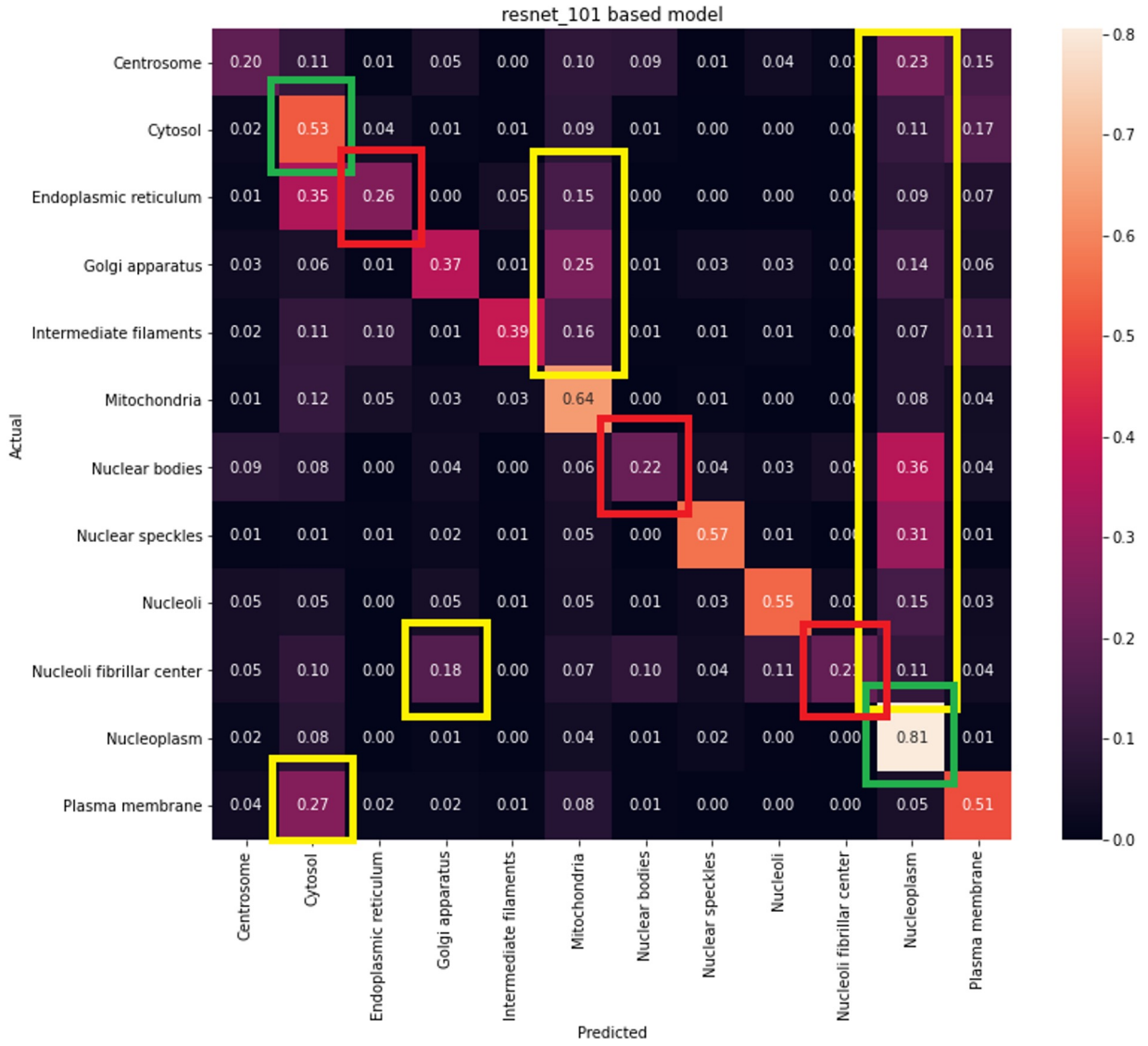
Validation samples: 28,445
Overall Validation Accuracy: 51.94%

Test samples : 9,482

Highest Accuracy:
- Nucleoplasm (81%)
- Cytosol (54%)

Lowest Accuracy:
- Endoplasmic reticulum – 26%
- Nucleoli fibrillar center– 21%
- Nuclear bodies – 22 %

Top mis classification:
- Cytosol
- Mitochondria
- Nucleoplasm
- Golgi apparatus



Confusion matrix(actual vs predicted %) – EfficientNetB6

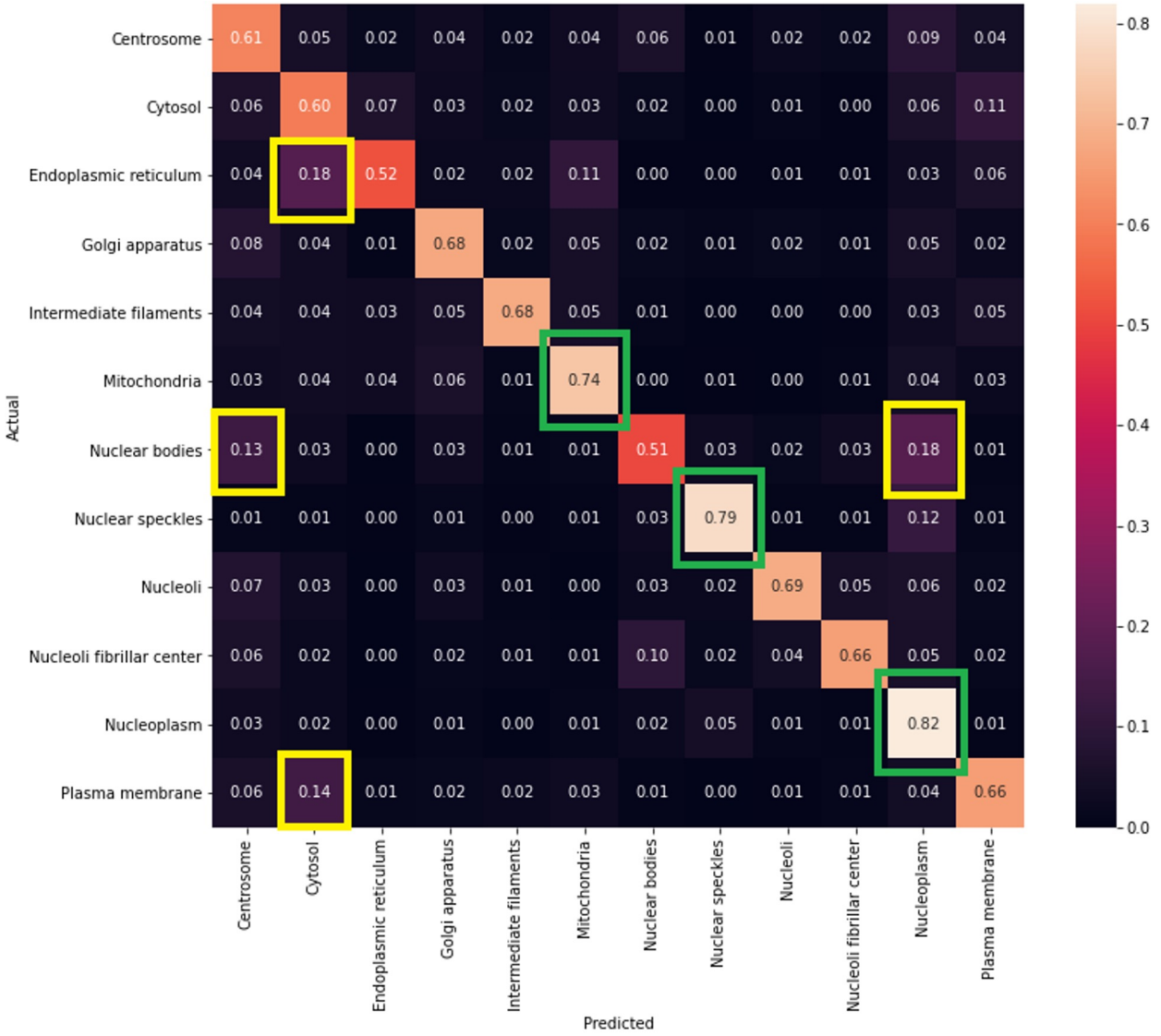
Validation samples: 28,445
Overall Validation Accuracy: 67.14%

Test samples : 9,482

Highest Accuracy:
- Nucleoplasm (82%)
- Nuclear speckles (79%)
- Mitochondria (74%)

Lowest Accuracy:
- Endoplasmic reticulum – 52%

Top mis classification:
- Nuclear bodies
- Nucleoplasm
- Cytosol



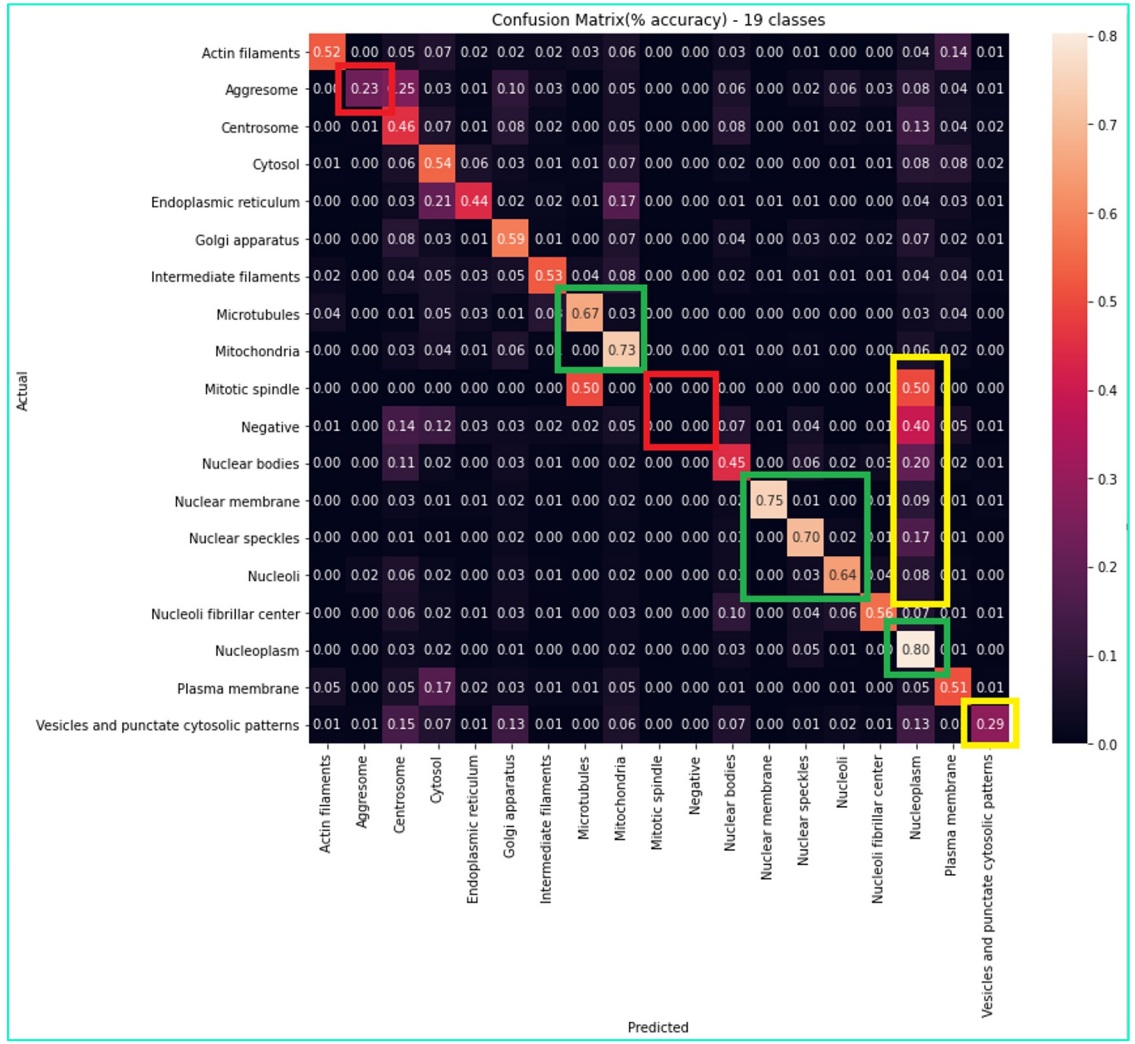
Conclusion

When we started the project, the leaderboard on the Kaggle competition was at **~44%** accuracy and beating this number became our immediate goal. As a team, we scoped the problem from 19 to 12 class detection but was able to achieve ***~67%*** *accuracy for the 12 classes* identified.

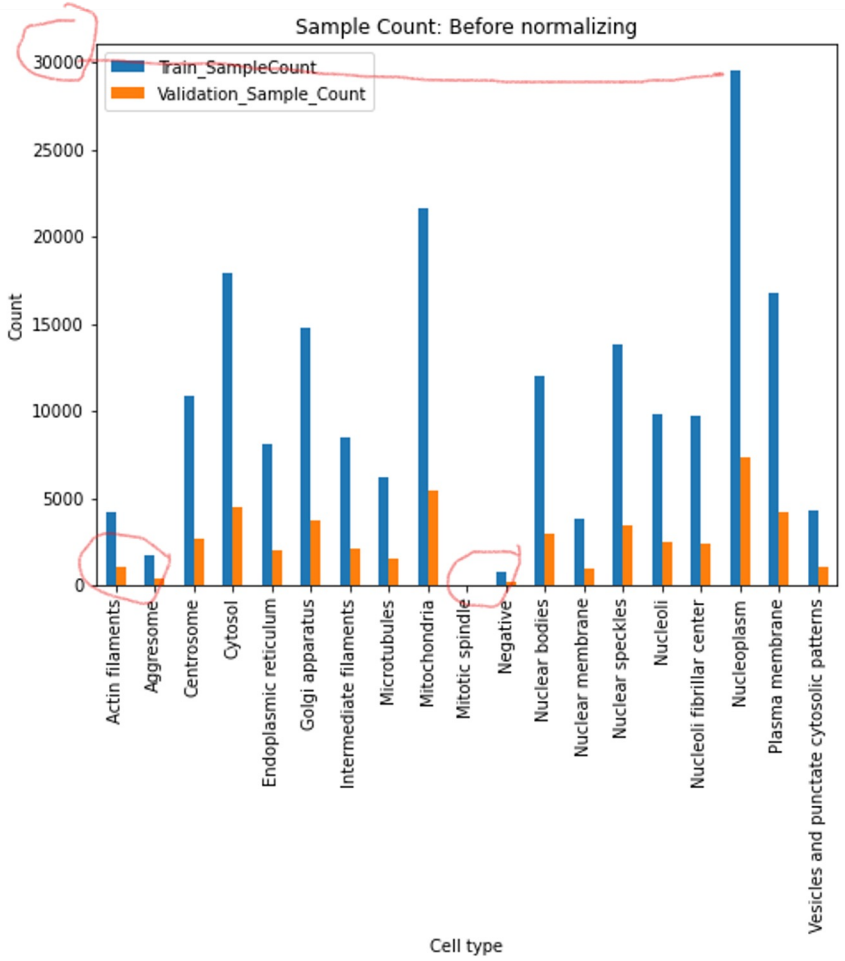
There is still room to improve models and experiment more. However, we conclude at this point that we met our accuracy goal for the project.

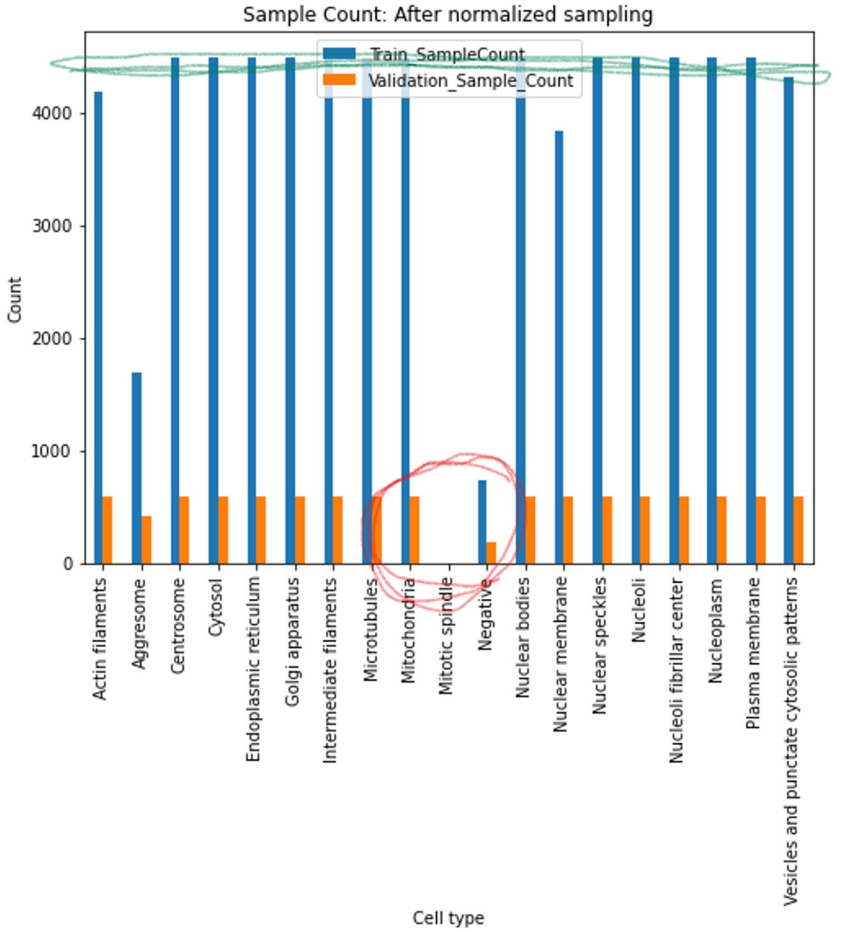
Latest model results classifying all the protein types (With Samples skew)

- Total validation samples - 48,632
- Overall accuracy ~60%
- Poor accuracy classes have low training samples
  - “Mitiotic spindle” has only 6 samples for training
  - “Negative” only has 700 samples
  - “Aggresome” - 1696
- Quite a bit of proteins are classified as “Nucleoplasm”



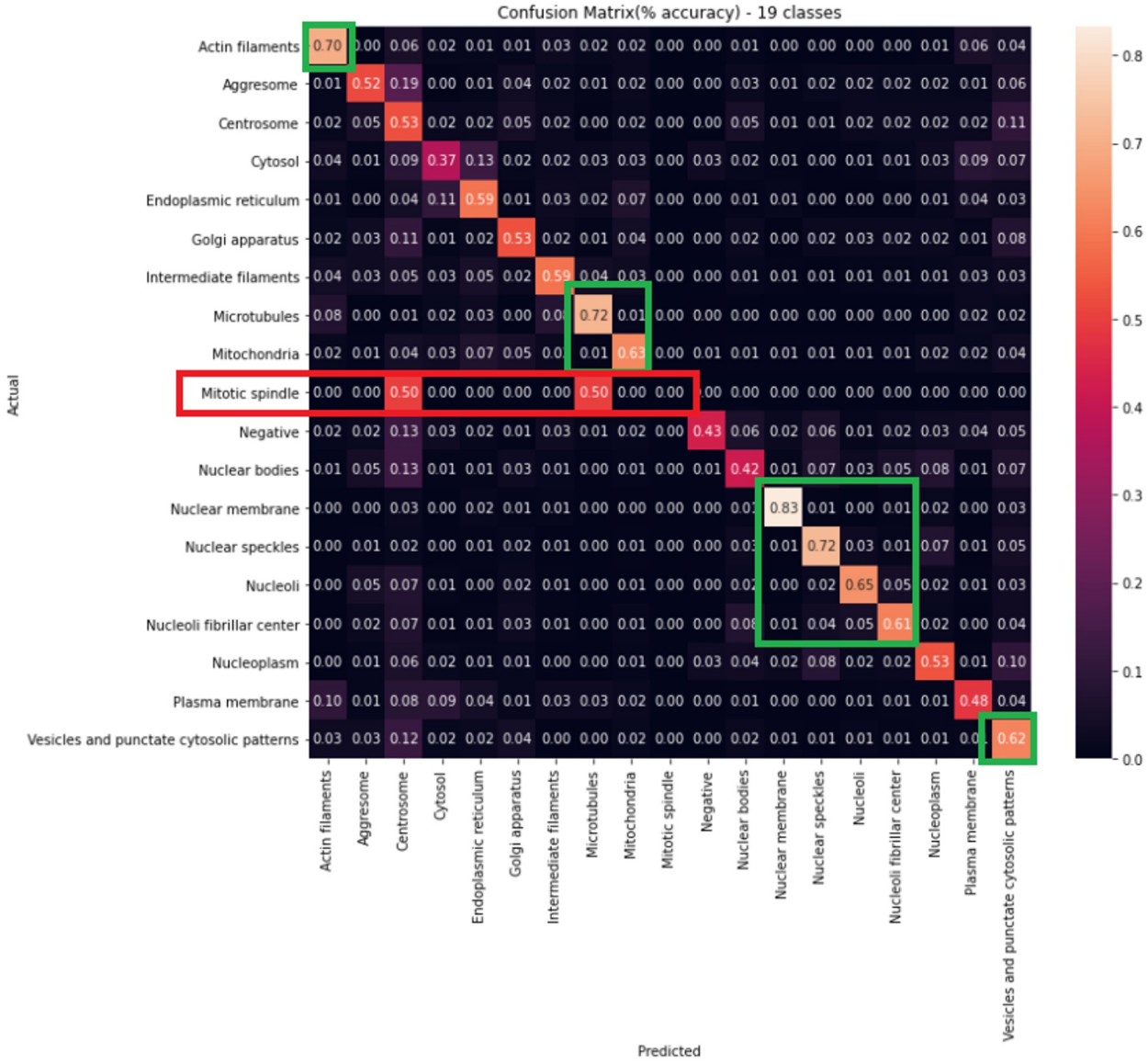
Understanding sample skew





After adjusting sample skew

- **From previous model**
  - Better at “Negative” class detection.
  - Reduced false positives on Nucleoplasm
  - 8 classes have more than 60+ accuracy over(2+ over previous model)
  - Not the best like other model for Nucleoplasm(probably other models can be used to detect that protein)
  - There is more room for improvement



Next steps

- Running the modeling for all the 19 classes and compare the accuracy
- Reduce imbalance in training data to provide samples for all classes equally and experiment
- Run the 19-class model for Kaggle test data and submit results
- Tooling support to help with data cleanup(identify noise images)
- Invest more on the optimizer internals and research on high performing models and tweak them and experiment. Experiment with other models like Mobilenet
- Get help from an SME in the protein cell area to get advice on how to improve the models
- Analyze the models in the context of scenarios like:
  - Whether model may not be good overall but could help finding a specific protein very well
  - Finding which cells have less success and remove them from the list and build a separate model

Q&A

Backup slides

References

- Kaggle : https://www.kaggle.com/c/hpa-single-cell-image-classification/leaderboard