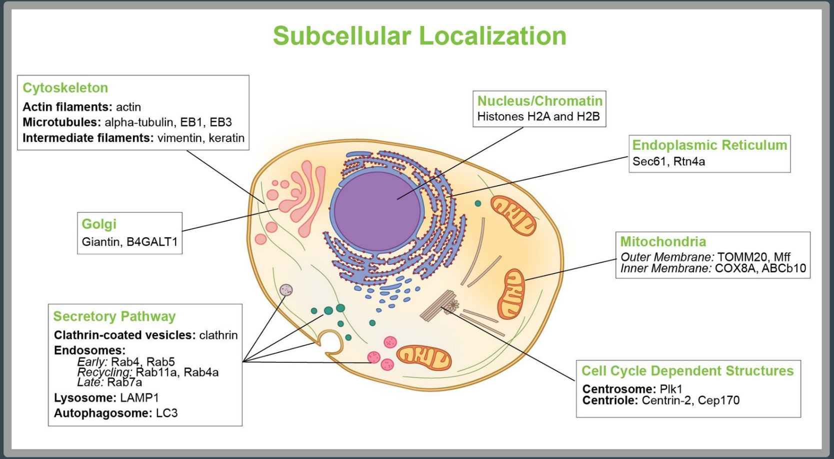
# ECE 228 Project

Mudit Jain, Shruti Joshi
06.03.2021

# Background

- In this project, we aim to classify cell subtypes based on localization of a specific protein within the cell in images obtained from confocal microscopy.
- Proteins localize in different organelles within the cells that enable them to perform different functions.
- This functional heterogeneity of cells can be inferred by looking at the different localizations of proteins within the cells.
- Identifying these cell subtypes can help with understanding cell biology, leading to drug discovery and better disease treatment.

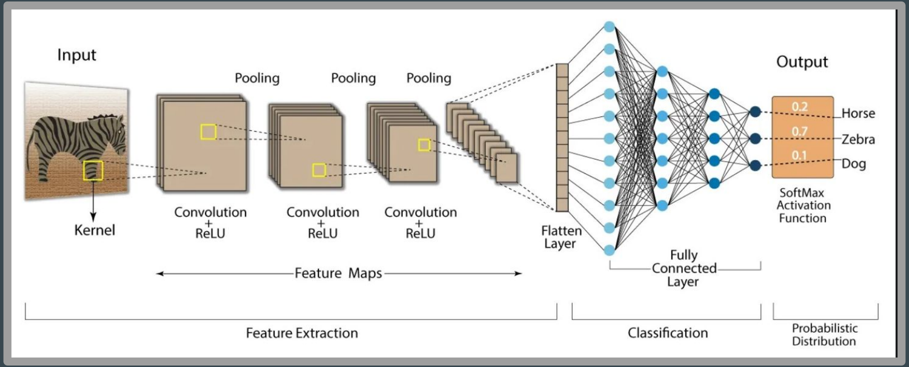


# Literature Survey

- Previous efforts to automate the classification of protein localization from microscopy images included methods such as:
    - k-NN classifiers
    - Support Vector Machines (SVM)
    - Convolutional Neural Networks
    - Decision Trees
- However, these approaches focused on a limited number of single patterns in a single cell type.
- These approaches only consider proteins present in a single location, excluding about ~50% of the proteins that localize to multiple locations.
- This offers an incomplete description of the cell biology, preventing understanding of the more complex subcellular architecture.

# How can Machine Learning help solve this problem?

- Deep learning has shown incredible results with complex, high-dimensional data in the recent years.
- CNNs have been widely applied to Computer vision tasks, such as image recognition and segmentation.
- Since our objective in this project is to classify cell subtypes from microscope images, we believe that CNNs are well suited to tackling the task.



# Dataset

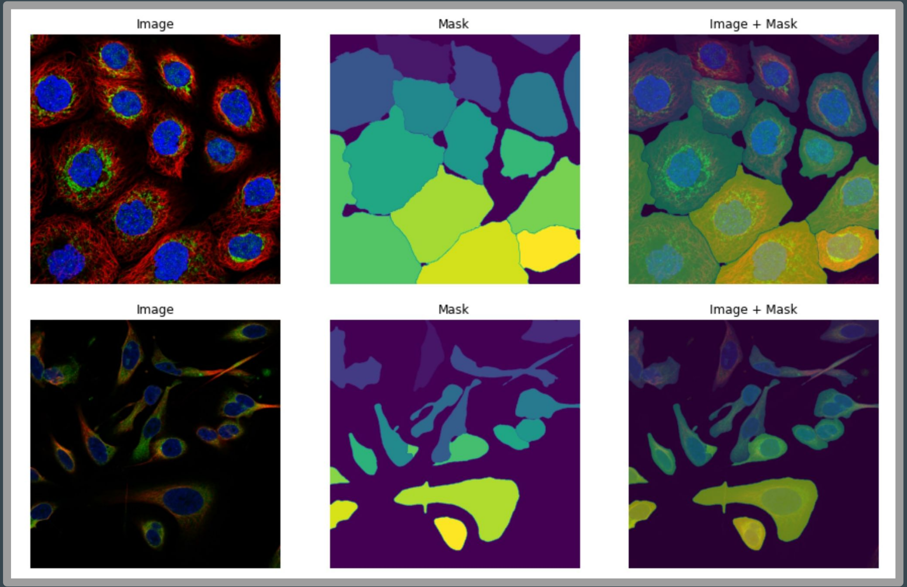
- The dataset is obtained from the Human Protein Atlas (HPA), acquired using confocal microscopy.
- Data is organised into train.zip and test.zip folders, both containing image samples (a mix of 1728x1728, 2048x2048 and 3072x3072 PNG files).
- Each image sample contains 4 files - blue, green, red, yellow - denoting different filters on the protein pattern represented by the sample.
  - Red - Microtubule channels
  - Blue - Nuclei channels
  - Yellow - Endoplasmic Reticulum (ER)
  - Green - Protein of interest
- The labels are present in the train.csv file. The Columns of this file are:
  - IDs - Base filename of the sample. Each sample consists of 4 files for the 4 filters
  - Label - The label assigned to each sample.

# Dataset

- Each image in the dataset consists of multiple cells, which may have all or a subset of the image-level labels.
- There are a total of 19 different labels - 18 for specific locations and 1 for negative result.
- Our task is to predict the label for each cell in the images in the test set, based on the image level labels available in the train set.
- The green (protein) filter is used to predict the label, while the other filters are used as references.
- This dataset comprises of 17 different types of cells, with highly different morphologies.

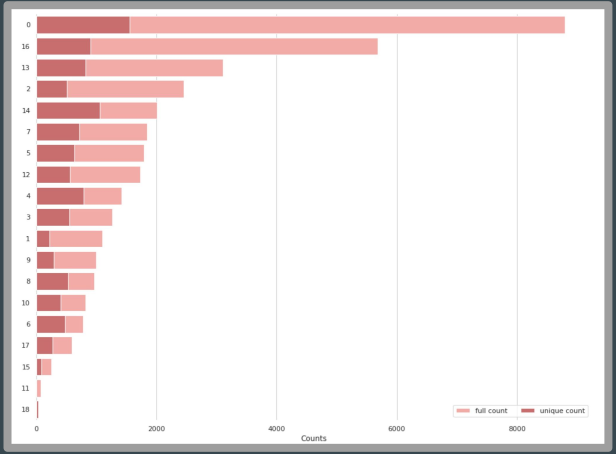
# Feature Extraction

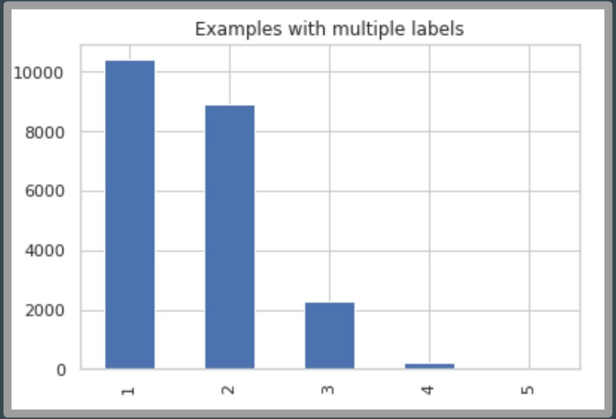
- We used the HPA Cell Segmentation module to create masks for individual cells.
- The cells extracted using these masks form the dataset for our multi-label classifier.
- Since it is a weakly-supervised problem, we only consider the images which have a single label for training our classifiers.



# Data Analysis

- An analysis of label distribution showed that labels 0 and 16 were the most common, while there were barely any images with labels 11 and 18, pointing to class imbalance.
- Each image had multiple labels and the number of unique label combinations in the train set was 432.
- We can also see that most images had 1 or 2 labels, and the maximum number of labels for any image was 4

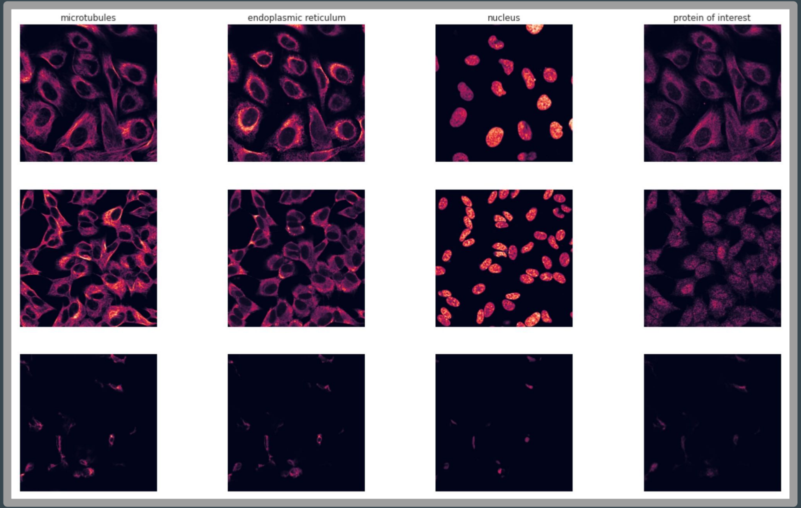




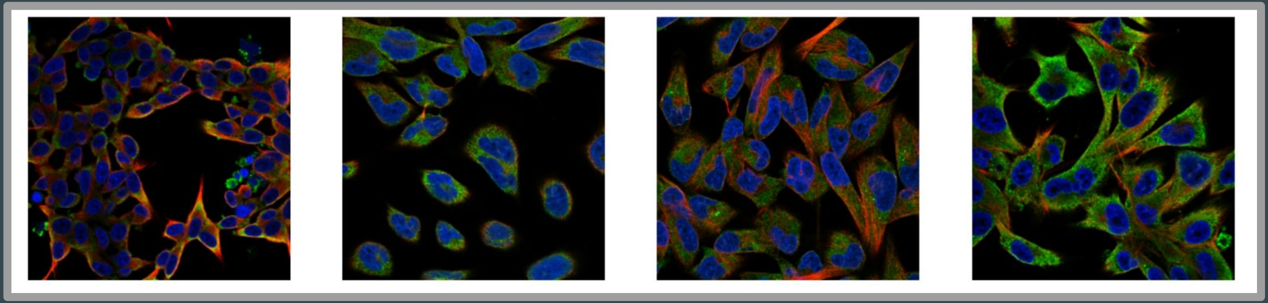
# Data Analysis

- An image seen through different filters highlights different components of the cells.
- Some images show faint staining.
- Figure at the bottom shows the image with all 4 filters combined.

**Different channels for an image**

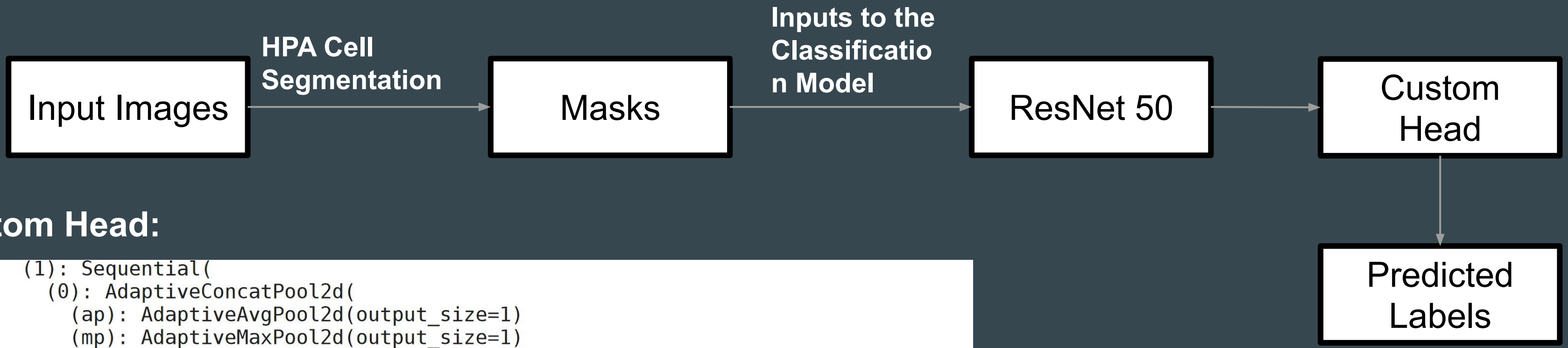


**Images with all 4 channels combined**



# Details on the Model Used

- We used ResNet-50 architecture as the backbone of the model and added a custom head to it.



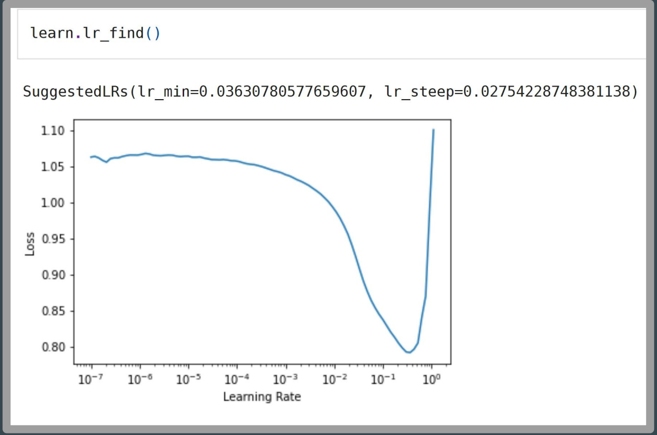
**Custom Head:**

```
  (1): Sequential(
    (0): AdaptiveConcatPool2d(
      (ap): AdaptiveAvgPool2d(output_size=1)
      (mp): AdaptiveMaxPool2d(output_size=1)
    )
    (1): Flatten(full=False)
    (2): BatchNorm1d(4096, eps=1e-05, momentum=0.1, affine=True, track_running_s
tats=True)
    (3): Dropout(p=0.25, inplace=False)
    (4): Linear(in_features=4096, out_features=512, bias=False)
    (5): ReLU(inplace=True)
    (6): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_st
ats=True)
    (7): Dropout(p=0.5, inplace=False)
    (8): Linear(in_features=512, out_features=19, bias=False)
  )
)
```

# Choosing the Optimal Learning Rate

- We used the learning rate finder from the fastai API to settle on an optimal LR.
- To start, we make a first guess on the learning rate based on the loss vs LR graph where the slope is the steepest, which gives us a value of 0.01
- We train the model with this LR and re-run the learning rate finder to fine-tune the LR. Here, we choose 0.01 again so that the loss doesn't increase with further training.

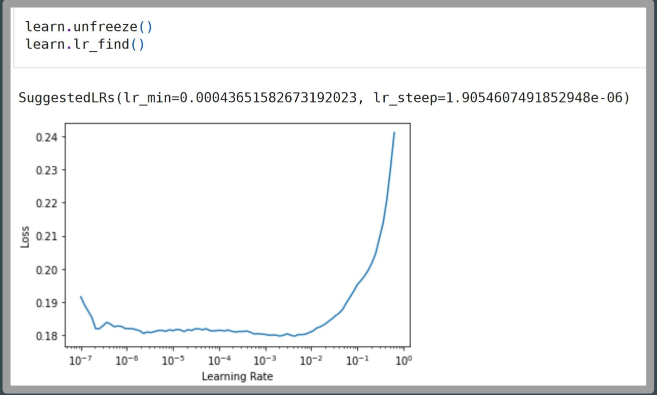
**First Guess for LR = 0.01, where slope is steepest**



```
learn.lr_find()
```

```
SuggestedLRs(lr_min=0.03630780577659607, lr_steep=0.02754228748381138)
```

**Fine tuning after some training**

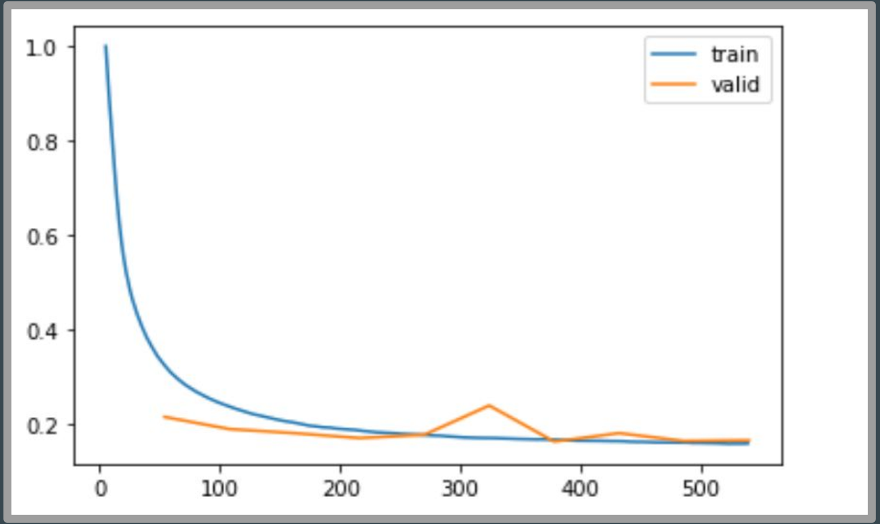


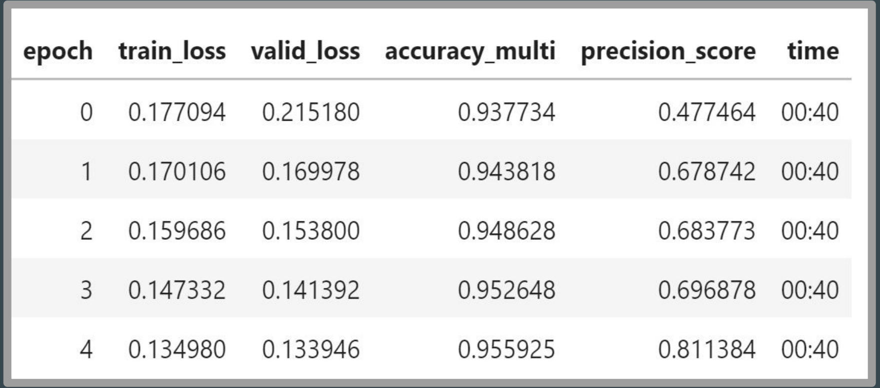
```
learn.unfreeze()
learn.lr_find()
```

```
SuggestedLRs(lr_min=0.00043651582673192023, lr_steep=1.9054607491852948e-06)
```

# Results

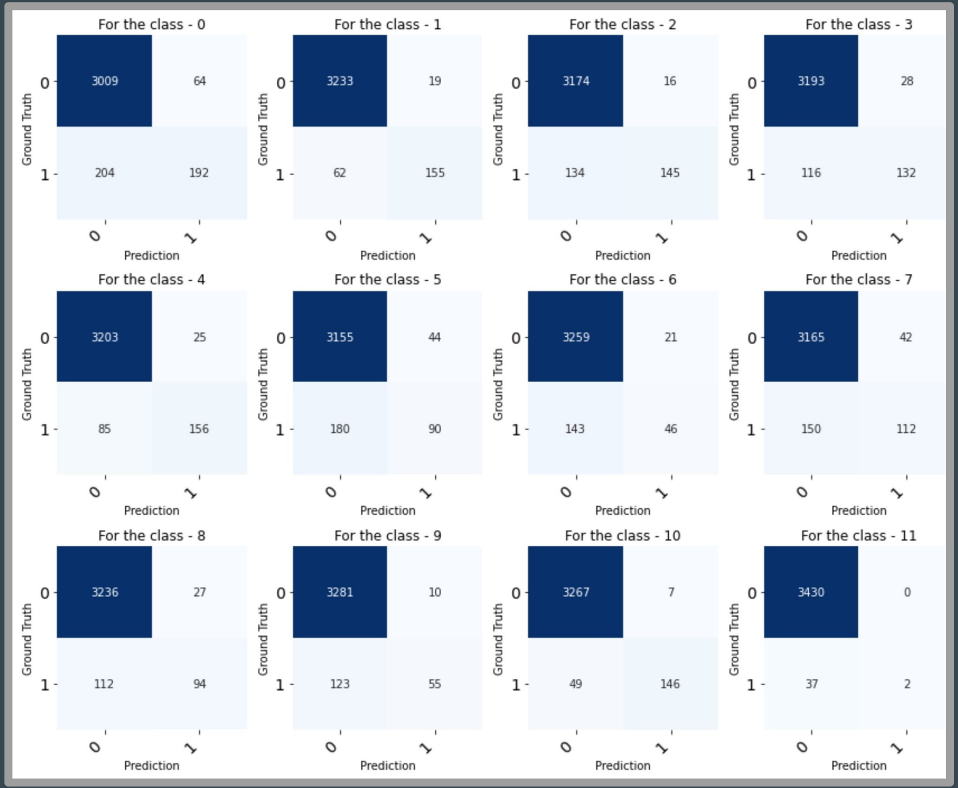
- The training and validation loss plots show that the loss decreased with training.
- Our model got an accuracy of 95.6% on the training set after fine tuning the LR.
- The confusion matrices for each label is shown in the next slide.

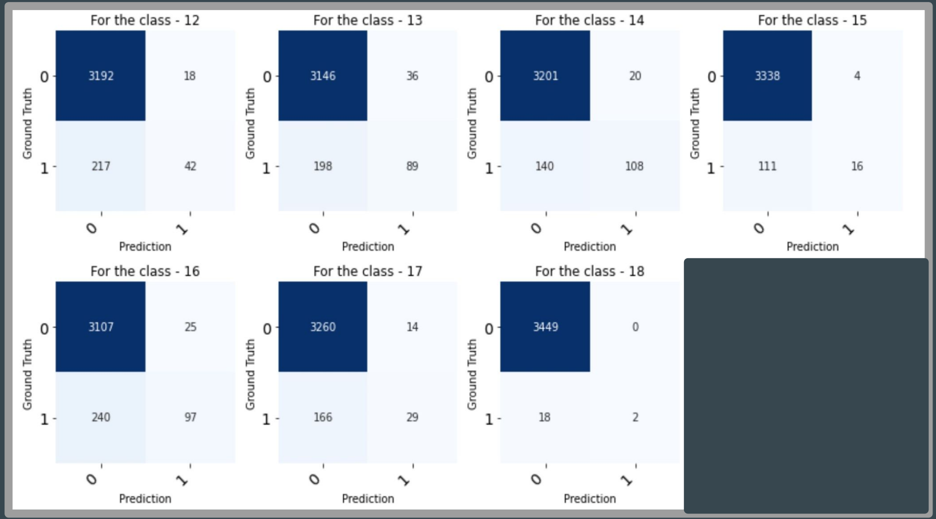




| epoch | train_loss | valid_loss | accuracy_multi | precision_score | time |
|---|---|---|---|---|---|
| 0 | 0.177094 | 0.215180 | 0.937734 | 0.477464 | 00:40 |
| 1 | 0.170106 | 0.169978 | 0.943818 | 0.678742 | 00:40 |
| 2 | 0.159686 | 0.153800 | 0.948628 | 0.683773 | 00:40 |
| 3 | 0.147332 | 0.141392 | 0.952648 | 0.696878 | 00:40 |
| 4 | 0.134980 | 0.133946 | 0.955925 | 0.811384 | 00:40 |

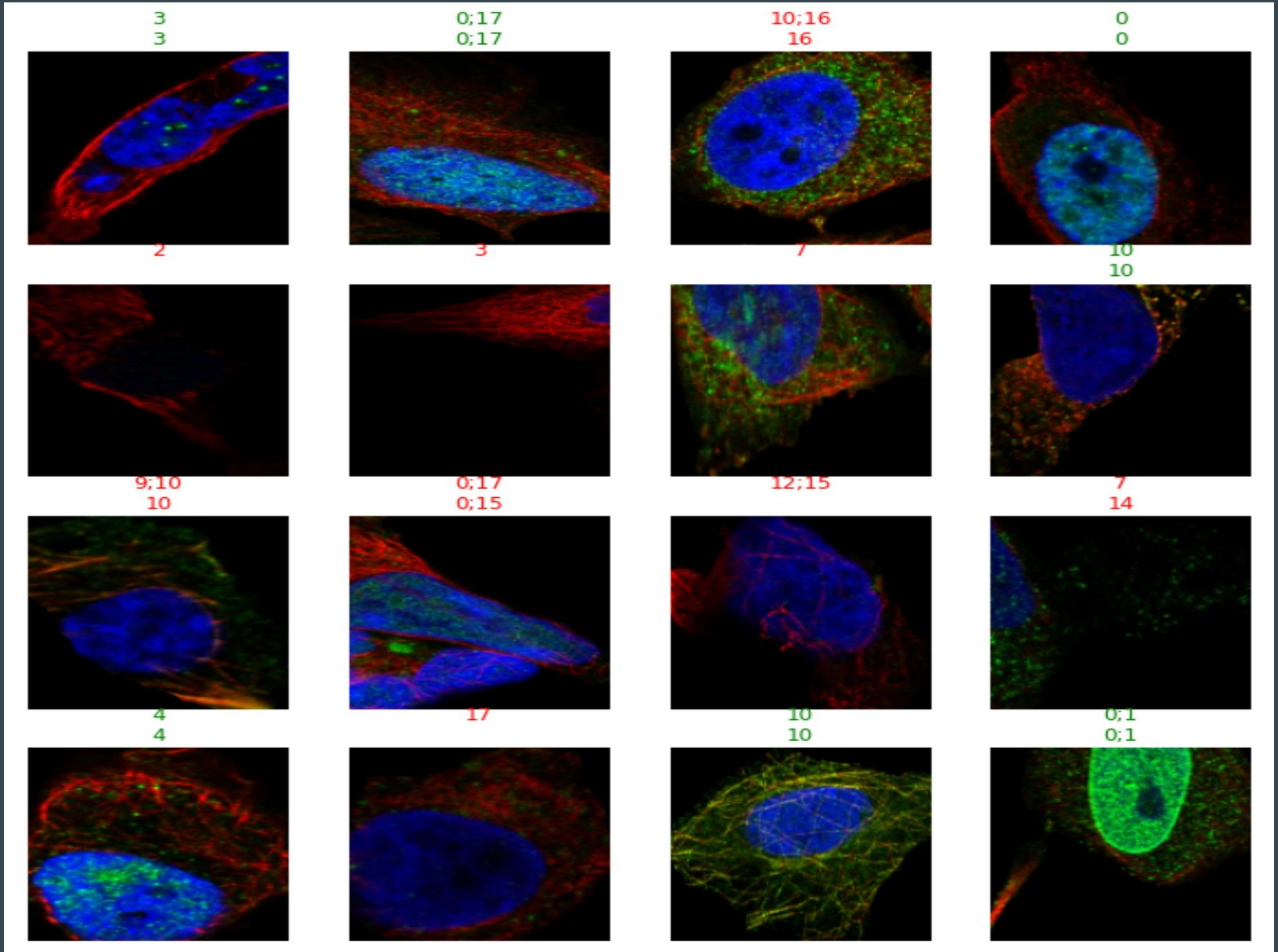
# Confusion Matrices for Each Label





# Results

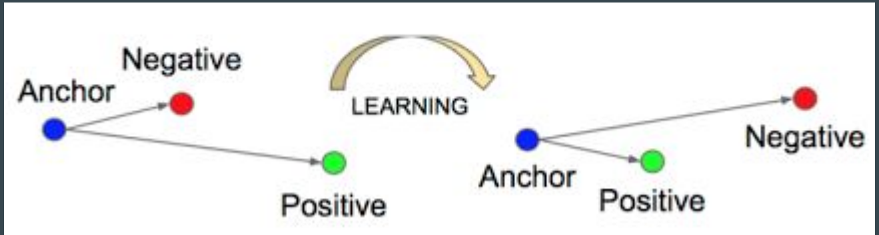
Average precision score, micro-averaged over all classes: 0.35



# Work to be finished before the final report

We have additional worked on a custom classifier which has resnet50 as its backbone and a custom head. However the additional modification is that we have added a custom loss function

The custom loss function is a combination of cross entropy loss and triplet loss. We have extracted the embeddings of the anchor, positive and negative image and generated the triplet loss. This loss helps increase the separability of the classes thus giving us a better classification.



# References

1. Ouyang, Wei, et al. “Analysis of the Human Protein Atlas Image Classification Competition.” Nature Methods, vol. 16, no. 12, 2019, pp. 1254–1261., doi:10.1038/s41592-019-0658-6.
2. Sullivan, Devin P, et al. “Deep Learning Is Combined with Massive-Scale Citizen Science to Improve Large-Scale Image Classification.” Nature Biotechnology, vol. 36, no. 9, 2018, pp. 820–828., doi:10.1038/nbt.4225.
3. Thul, Peter J., et al. “A Subcellular Map of the Human Proteome.” Science, vol. 356, no. 6340, 2017, doi:10.1126/science.aal3321.
4. HPA-Cell-Segmentation, https://github.com/CellProfiling/HPA-Cell-Segmentation