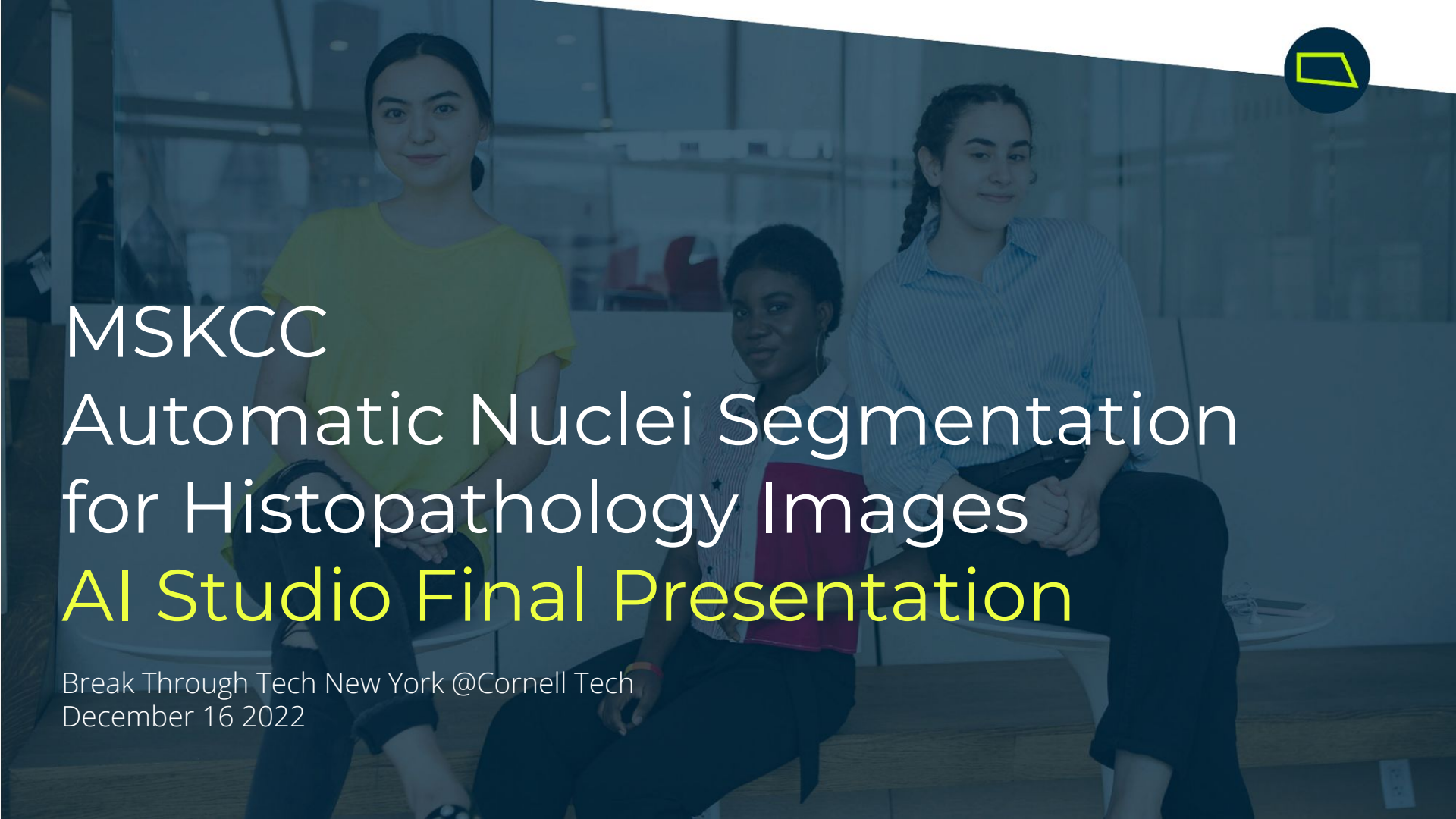


MSKCC
Automatic Nuclei Segmentation for Histopathology Images
AI Studio Final Presentation

Break Through Tech New York @Cornell Tech
December 16 2022

Introductions

Our Team

Chichi Mba
Columbia University

Nicole Joseph
The Cooper Union

Kaylee Fang
Columbia University

Leona Torres
Queens College

Selina Sun
New York University

Our AI Studio TA and Challenge Advisor

Ananya Ganesh
AI Studio TA

Alex Hollingsworth
Challenge Advisor

Presentation Agenda

1. Introductions
2. Project Overview
3. Data Preprocessing
4. Model Selection and Evaluation
5. Final Thoughts

AI Studio Project Overview

Create a deep learning model that can accurately segment nuclei and separate overlapping or touching nuclei in H&E stained tissue sections

Our Goals

- Learn more about deep learning, computer vision, and image processing techniques
- Understand how to work with a small data set
- Differentiate between the various types of image segmentation
- Create a neural network that can accurately segment nuclei and separate overlapping and touching cells in H&E-stained tissue sections using Deep Learning.

Business Impact

- Accurately segmented nuclei provide important insights for pathologists:
  - Physical characteristics of nuclei (size, shape)
  - Spatial distribution
- Reduces need for manual annotations of nuclei
- Manual separation of images is tedious and time consuming
- Automated nuclei segmentation is important to gaining further insight into cell features and functionality

Our Approach

Step 1

Data Understanding and Choosing Augmentation Techniques

Clean + preprocess the images (normalization, tiling), and apply augmentation techniques

Step 2

Semantic Segmentation

Train a U-Net model to semantically segment nuclei and evaluate using Dice Score

Step 3

Instance Segmentation

Train a Mask RCNN model to segment the nuclei cells and identify overlapping cells. Evaluate with Jaccard Index.

Step 4

Improving our Models

Modifying our models to produce better evaluations(training for longer epochs, using more augmentation techniques)

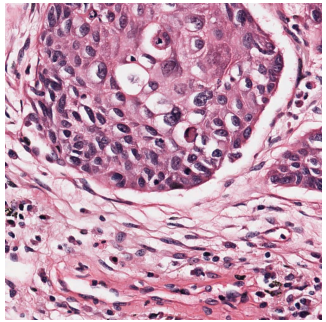
Resources We Leveraged

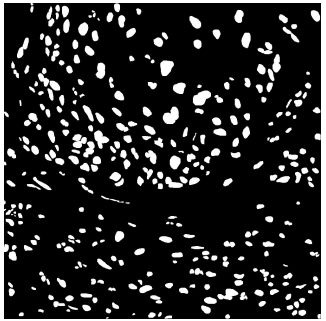
Data Preprocessing

The Dataset

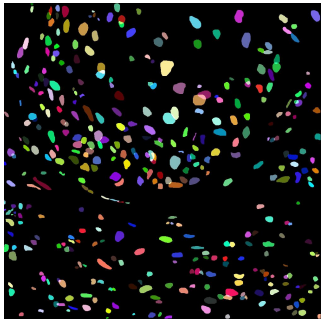
- **MoNuSeg dataset**
  - Public domain data
  - Obtained by annotating tissue images of several patients with tumors of different organs at multiple hospitals
  - Tissue images across 30 patients and 7 different organs
- 44 H&E stained tissue images, each 1000x1000
  - 30 for training, 14 for testing

H&E stained tissue

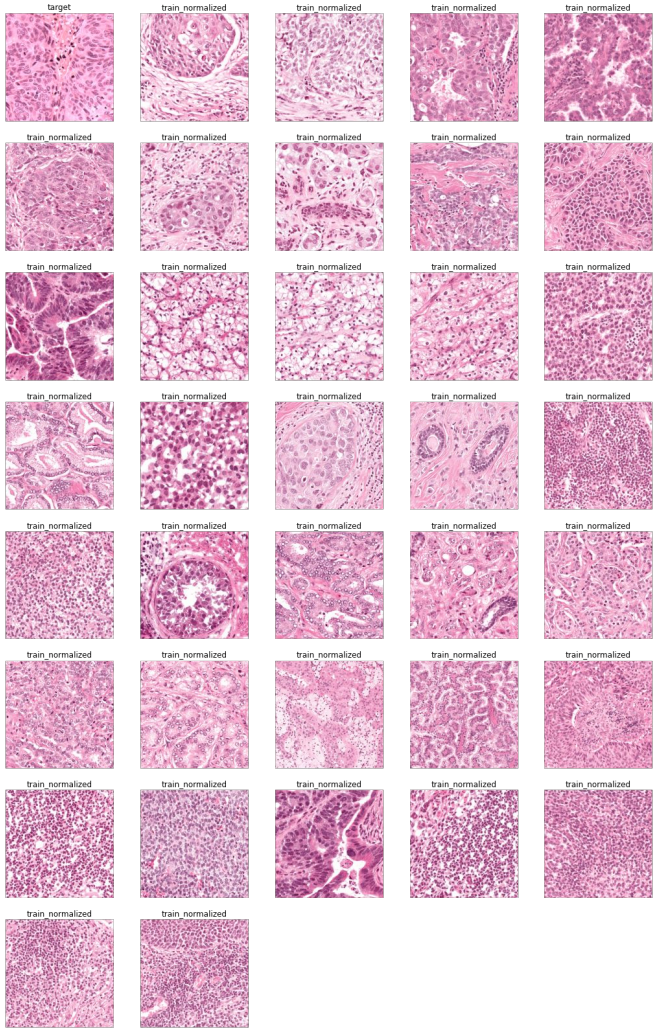




Ground-Truth Binary Mask



Ground-Truth Color Mask



Stain Normalization

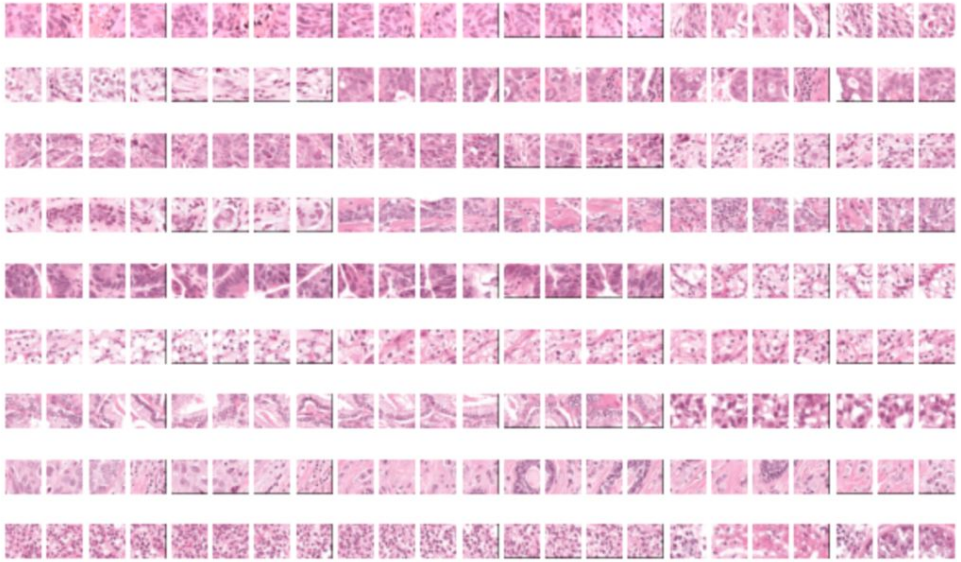
- Every image in the training and test dataset was normalized according to one random image from the dataset
  - Normalizes based on stain color and hue
  - Transforms all features to be on a similar scale
- Improves model training and performance
- Tools used: **StainTools** library (https://staintools.readthedocs.io/en/latest/)

Image Tiling

- Each 1000x1000 image + mask was tiled into 16 256x256 tiles
- Benefits of tiling:
  - Faster training and performance
  - Increases batch diversity
- Tools used: **tiler** library (https://pypi.org/project/tiler/)

▾ Tiling

```
[55] # tiling tissue images
     tile_dataset(data)
```

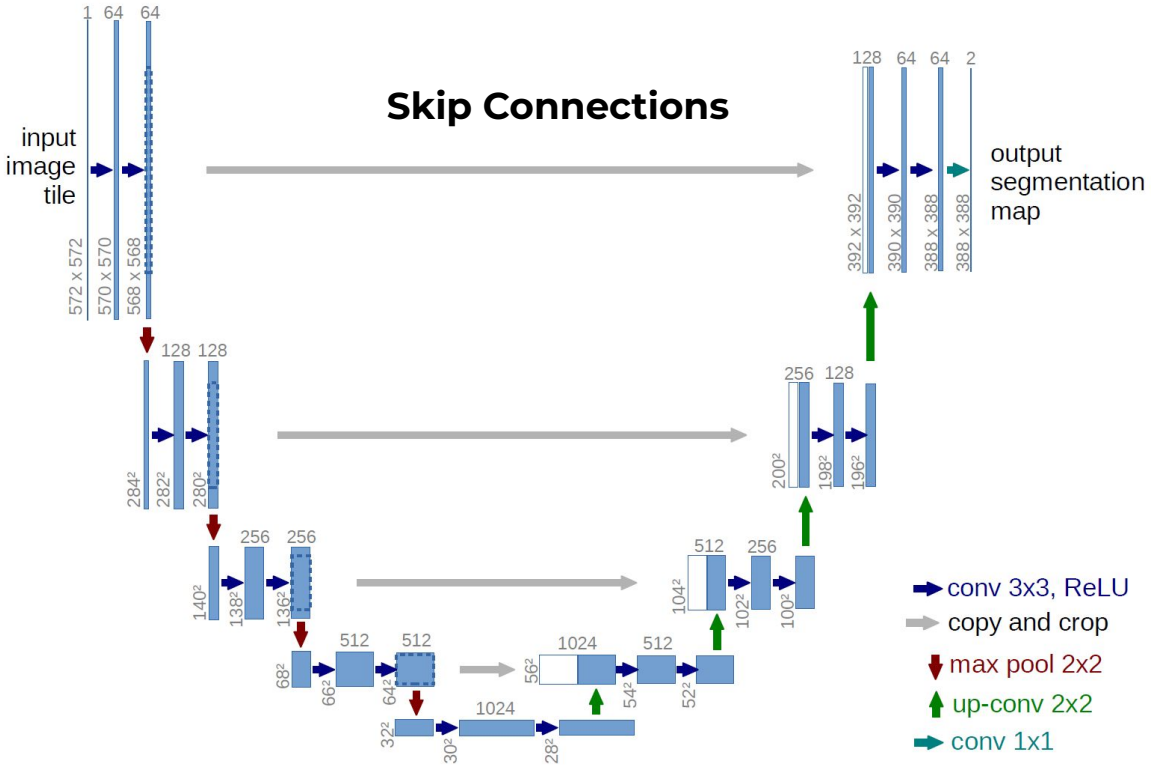


Preprocessing Object

```
[ ]  class PreProcessing:
       def __init__(self, path_to_training, path_to_testing):
         self.training_imgs_path = path_to_training
         self.testing_imgs_path = path_to_testing

         # names of images
         self.train_names = np.sort(np.array([file_name for file_name in listdir(self.training_imgs_path) if isfile(join(self.training_imgs_path, file_name))]))
         self.test_names = np.sort(np.array([file_name for file_name in listdir(self.testing_imgs_path) if isfile(join(self.testing_imgs_path, file_name))]))

         # normalize images
         self.training_imgs = []
         self.testing_imgs = []

         # normalized and tiled images
         self.training_imgs_tiled = []
         self.testing_imgs_tiled = []

         # normalized, tiled and augmented training images
         self.training_imgs_aug = []
```

Model Selection and Evaluation

U-Net Architecture

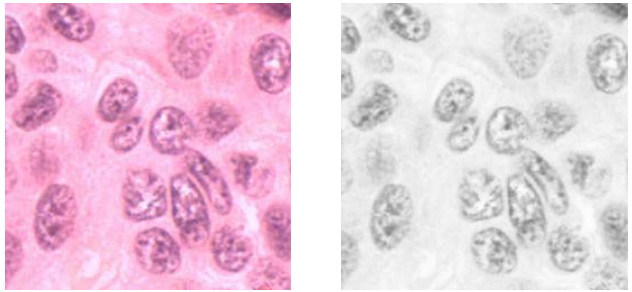


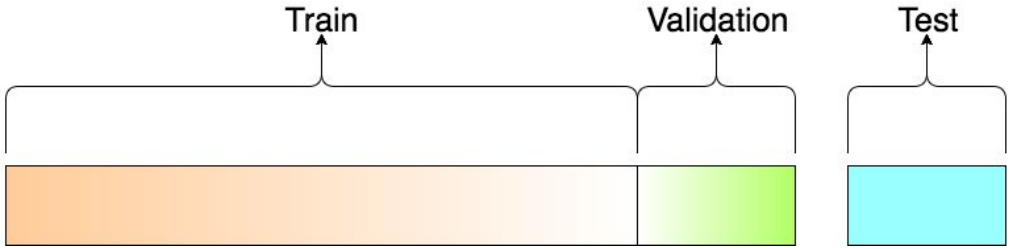
Example Code

```
#Contraction/Encoder path (going down the U) - this is made up of convolutional layers (c1, c2, etc)
# feature space = 16; apply 3x3 convolutions; padding = same to keep output width and height same as input
c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(s)
c1 = Dropout(0.1)(c1)
c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(c1)
p1 = MaxPooling2D((2, 2))(c1)
```

- Scaled down feature space dimensions by a factor of 8
- For the outputs, we used a sigmoid activation function to ensure that the mask pixels are in a [0, 1] range

Training Process





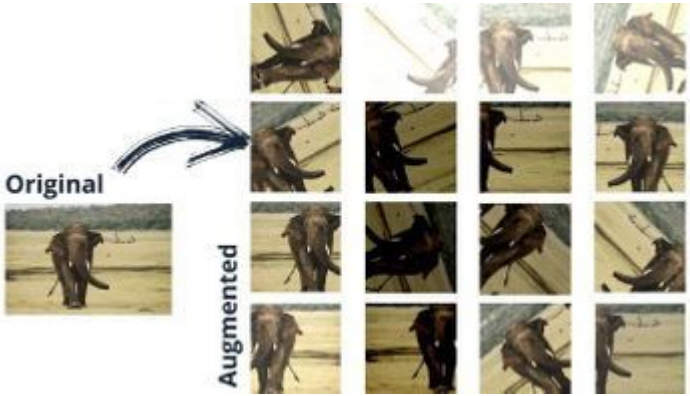
Stage 1: When we first started training the U-Net model, we converted the images to grayscale
Best Accuracy Score: ~20%

Stage 2: Added a validation set that was 10% of the training data

Best Accuracy Score: ~40%

Stage 3: Data Augmentation (on-the-fly)

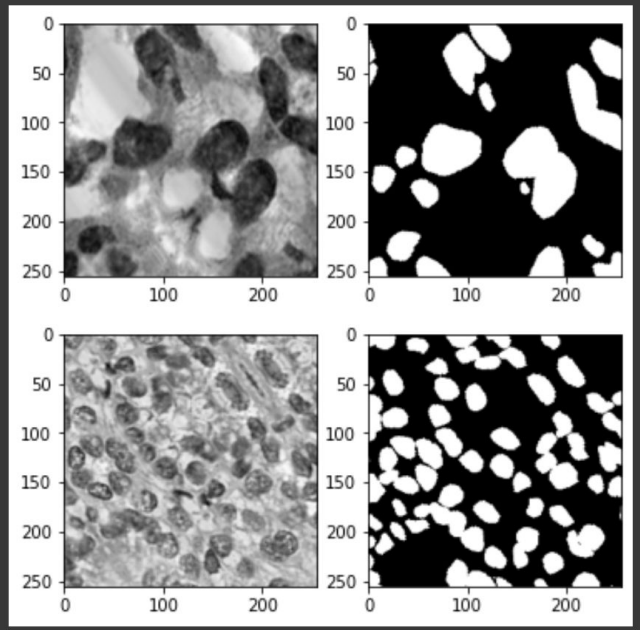
```
img_data_gen_args = dict(rotation_range=90,
                         width_shift_range=0.3,
                         height_shift_range=0.3,
                         zoom_range=0.3,
                         horizontal_flip=True,
                         vertical_flip=True,
                         fill_mode='reflect')
```



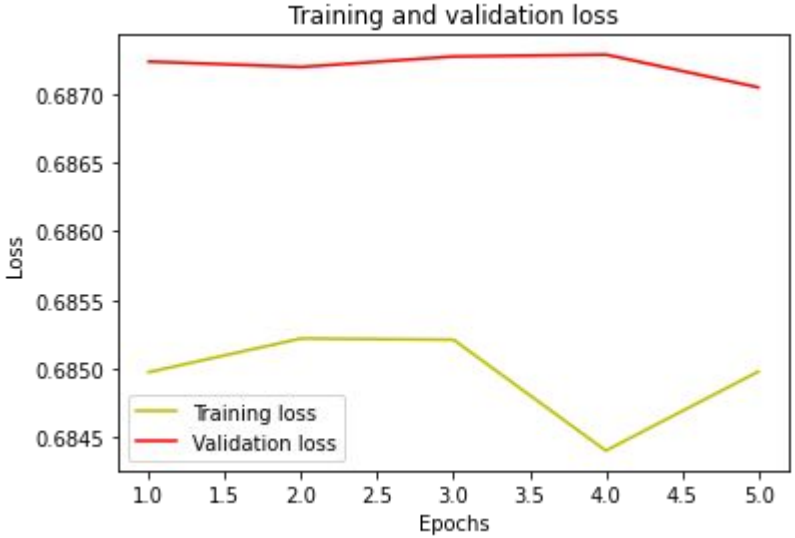
Stage 3: Data Augmentation increases the amount of data needed to train robust AI models. Applied to both images and masks.
Best Accuracy Score: ~60%

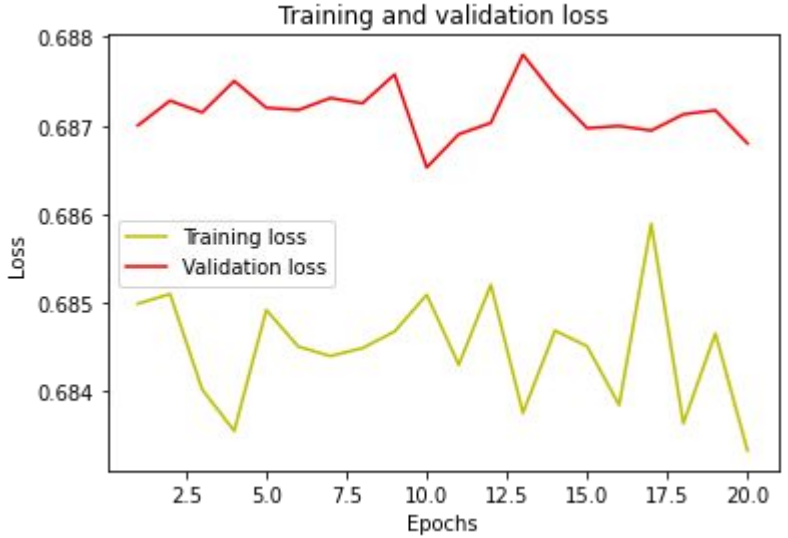
On-the-fly data augmentation via the keras Image Data Generator ensures that the augmented images are matched up with their corresponding masks

```
[57]  x = image_generator.next()
      y = mask_generator.next()
      for i in range(0,2):
          image = x[i]
          mask = y[i]
          plt.subplot(1,2,1)
          plt.imshow(image[:,:,0], cmap = 'gray')
          plt.subplot(1,2,2)
          plt.imshow(mask[:,:,0], cmap = 'gray')
          plt.show()
```



U-Net Results

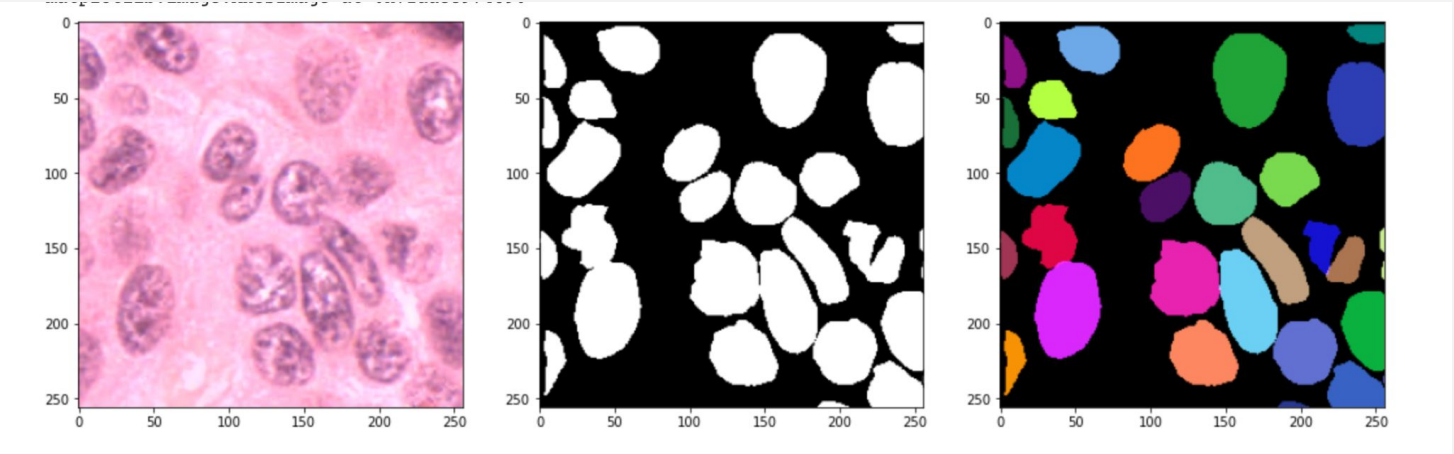




Stage 4: Fix mask and image tile error (some were not matched up)
Best Accuracy Score: ~75%

```
[64] loss = history.history['loss']
     val_loss = history.history['val_loss']
     epochs = range(1, len(loss) + 1)
     plt.plot(epochs, loss, 'y', label='Training loss')
     plt.plot(epochs, val_loss, 'r', label='Validation loss')
     plt.title('Training and validation loss')
     plt.xlabel('Epochs')
     plt.ylabel('Loss')
     plt.legend()
     plt.show()
```

Mask R-CNN for Instance Segmentation



Model Descriptions

Model Name	Description	Results	Why we chose it?
U-Net	**Semantic segmentation** - segments objects in image from background	A binary mask (black and white)	Gold standard for biomedical image segmentation problems
Mask R-CNN	**Instance segmentation** - combines image segmentation with object detection to segment objects from each other	A mask that shows each distinct nuclei cells	Proven to be a well performing model for instance segmentation

Final Thoughts

Obstacles

- We had to ensure that the dimensions of the input images for the U-Net matched up
- The mrcnn package for the Mask RCNN model is only compatible with a certain version of tensorflow which forced us to use virtual machines
- Using virtual machines on Google Cloud allowed us to run our U-Net model for more epochs to get a more accurate model

Insights and Key Findings

- On the fly augmentation removes the problem of having to match up masks and the predicted masks
- Converting the training images to grayscale may simplify computations and reduce time
    - However, training with RGB images provides information in all three channels and improves model performance
- The Dice Score and Jaccard Index are similar metrics for evaluating model performance, but the Dice Score provides more nuance in comparing pixel similarity

Potential Next Steps

- Get more training data
- Experiment with U-Net architecture
- Implement advanced techniques for hyperparameter tuning
    - Learning Rate Scheduler
- Watershed segmentation to improve accuracy


Questions?