Tumor Cellularity Assessment

Rene Bidart

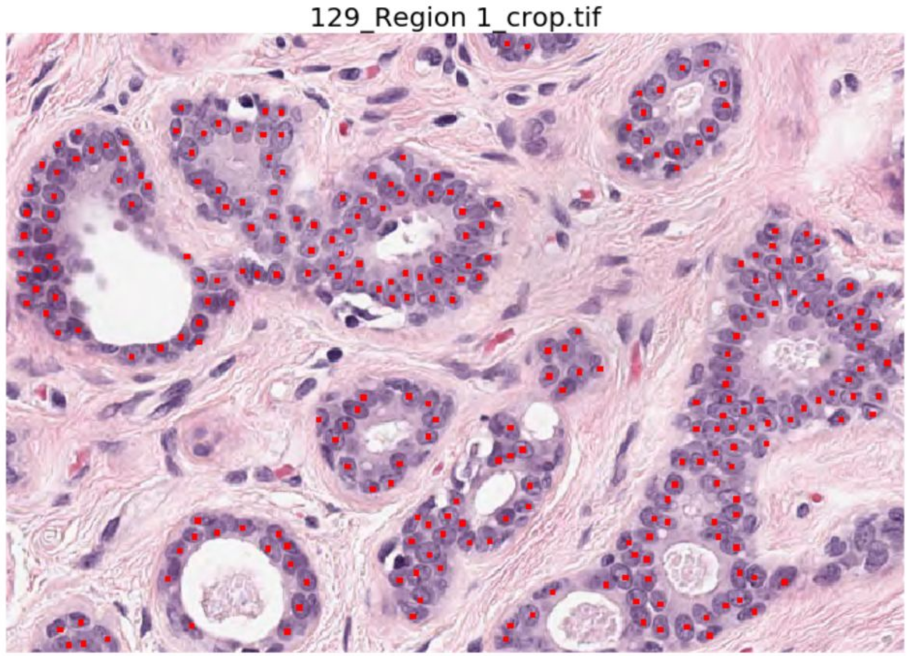
Goals

Find location and class of all cell nuclei:

- Lymphocyte Cells
- Normal Epithelial Cells
- Malignant Epithelial Cells

Why:

- Test effectiveness of pre-surgery treatment
- Doing this manually is time consuming and costly

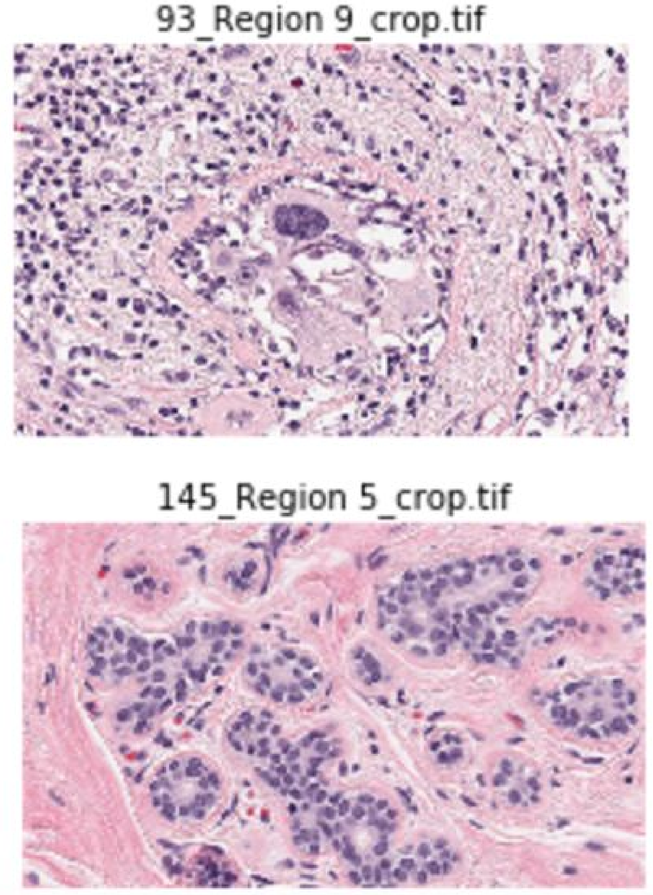


Sample slide with malignant cells labelled in red

Data

154 images of post-op breast cancer tissue samples:

- Subsections of whole slide images of about 512x512 pixels
- Selected to have a mix of Lymphocyte, normal epithelial, and malignant epithelial cells (not random)
- Only the center of the cells are labelled, no segmentation.
- Some cells may be missing labels



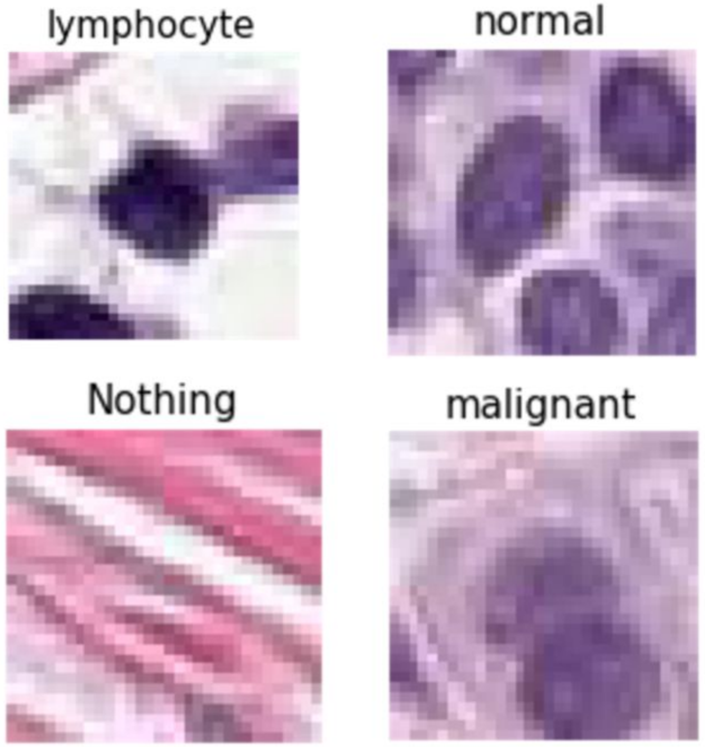
Human Classification

- **Lymphocytes** are small, dark round nucleus

- **Normal epithelial** cells are lighter and slightly bigger than lymphocytes

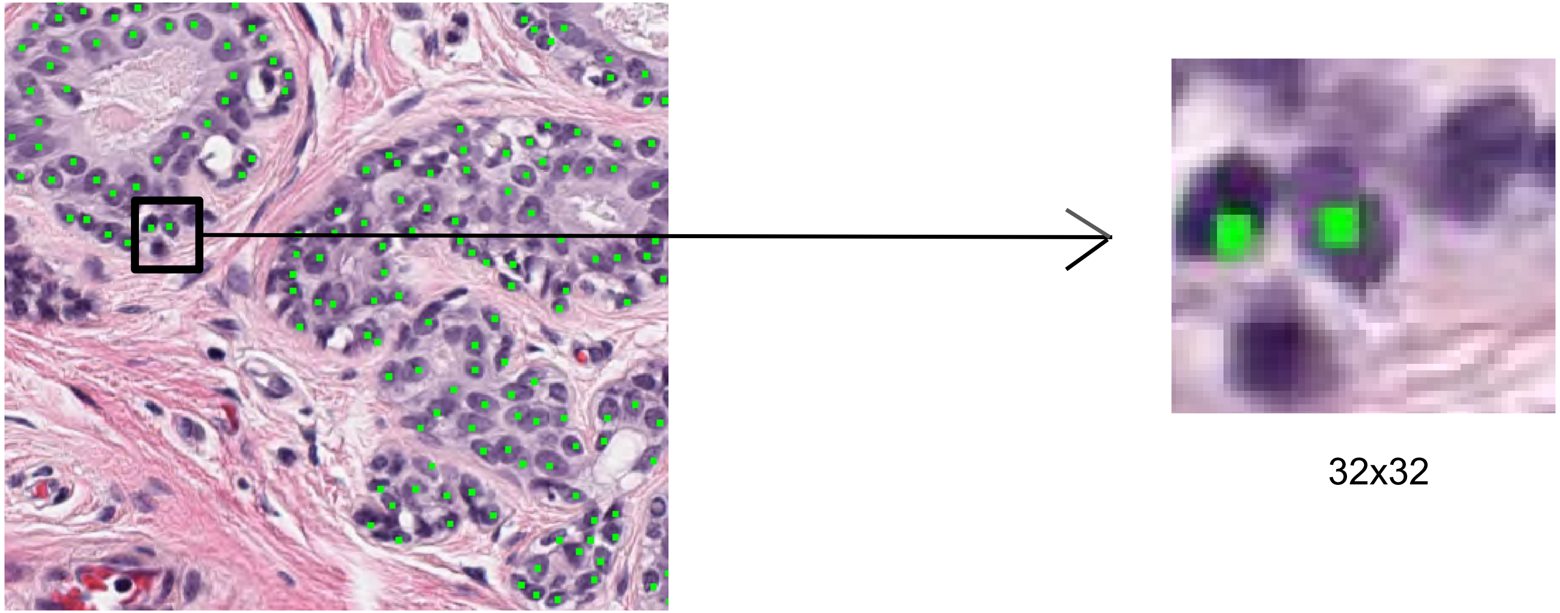
- **Malignant epithelial** are 2-3 times bigger than normal, and have irregular boundaries

- Structural information is also important



Classification

- Classification is easier than localization
- Crop a 32x32 box around each nucleus to create a classification training set
- Add some random non - nucleus samples to create a fourth class



32x32

Model

Nothing complex:

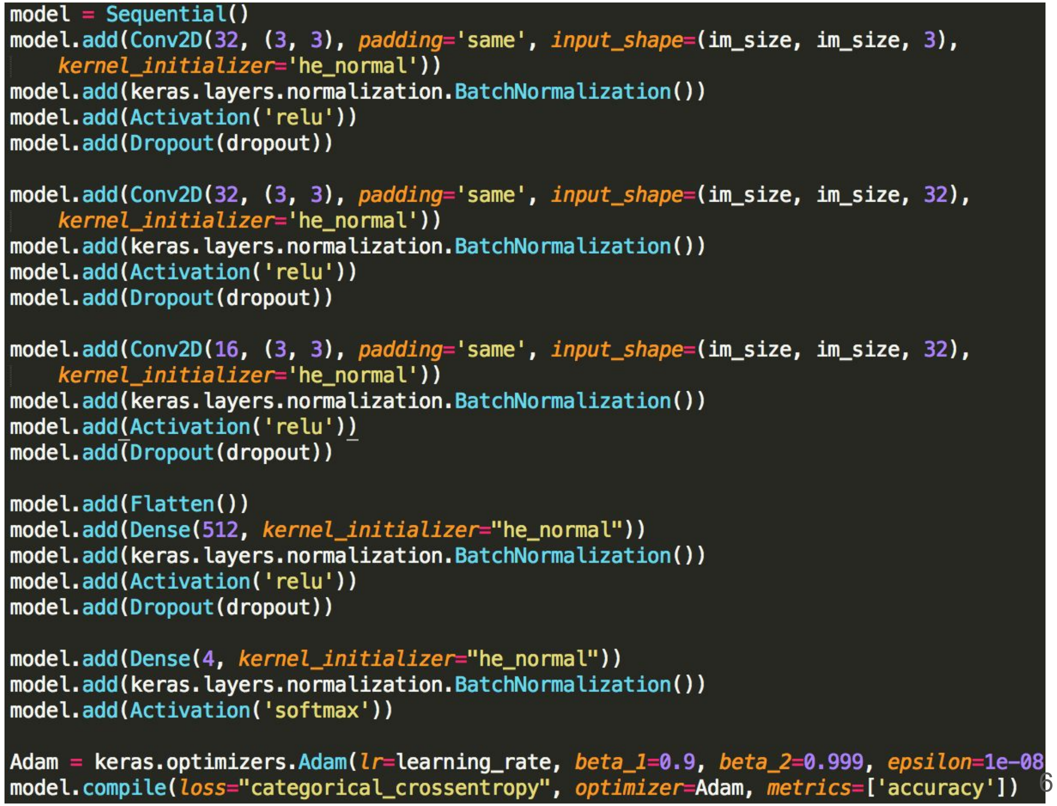
Usual Architecture:

- 3 Convolutional layers
- 2 FC layers
- Batchnorm
- Relu

Hyperparameter optimization the easy way:

- for model in model_list:

- for parameter in paramater_list:



```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same', input_shape=(im_size, im_size, 3),
    kernel_initializer='he_normal'))
model.add(keras.layers.normalization.BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(dropout))

model.add(Conv2D(32, (3, 3), padding='same', input_shape=(im_size, im_size, 32),
    kernel_initializer='he_normal'))
model.add(keras.layers.normalization.BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(dropout))

model.add(Conv2D(16, (3, 3), padding='same', input_shape=(im_size, im_size, 32),
    kernel_initializer='he_normal'))
model.add(keras.layers.normalization.BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(dropout))

model.add(Flatten())
model.add(Dense(512, kernel_initializer="he_normal"))
model.add(keras.layers.normalization.BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(dropout))

model.add(Dense(4, kernel_initializer="he_normal"))
model.add(keras.layers.normalization.BatchNormalization())
model.add(Activation('softmax'))

Adam = keras.optimizers.Adam(lr=learning_rate, beta_1=0.9, beta_2=0.999, epsilon=1e-08
model.compile(loss="categorical_crossentropy", optimizer=Adam, metrics=['accuracy'])
```

Augmentation

Rotations and flips

- Standard augmentation, because we want consistent classification independent of the orientation

Small crops

- The data centers aren't labelled perfectly, so we want consistent classification after small changes

No max-pooling

- Little benefit to reducing image size but there is a significant cost. Empirically this was true.

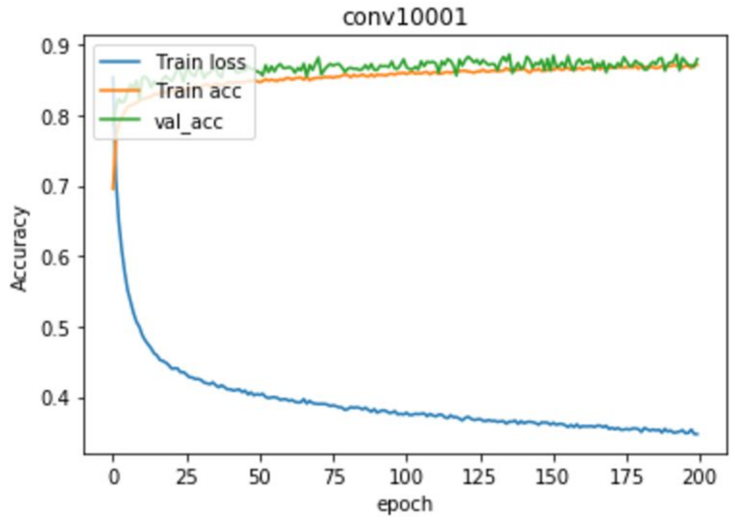
Results

Using 32x32 images:

- 88% classification accuracy

Using 64x64 images:

- 90% classification accuracy

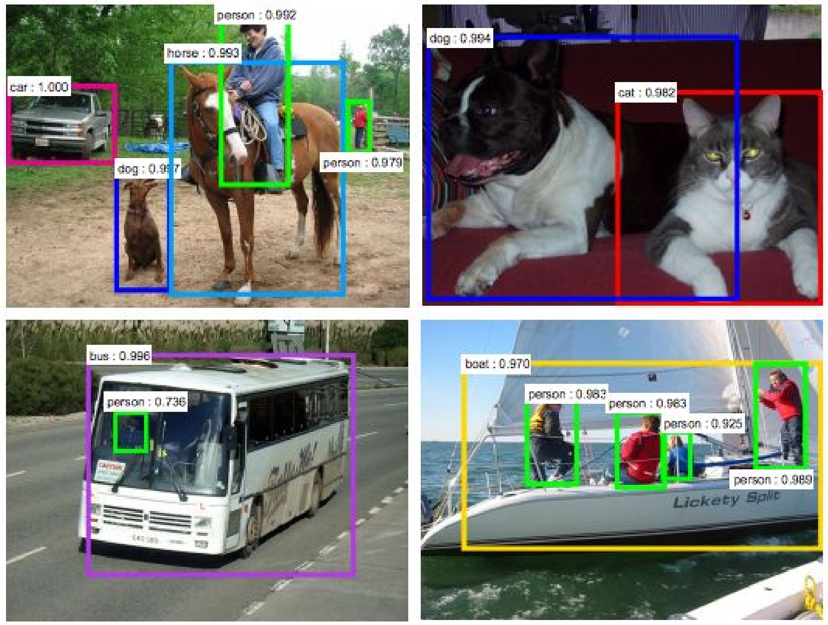


What about localization?

Localization

Popular methods use bounding boxes, like faster RCNN or YOLO

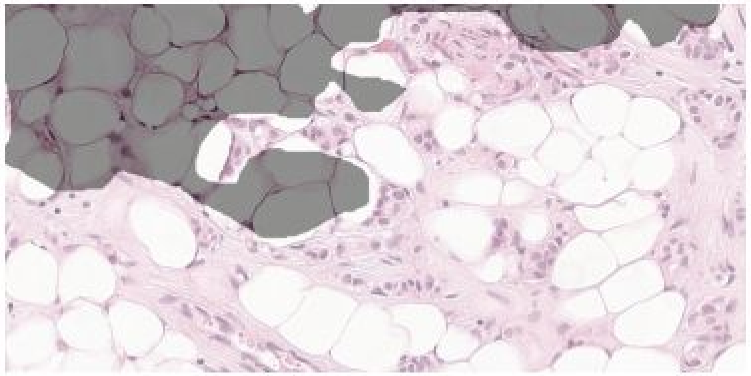
- Need segmented training data
- Designed to perform best on larger objects, but cell nuclei are very small
- These methods will require major changes to be useful in this problem



Localization

Detecting Cancer Metastases on Gigapixel Pathology Images

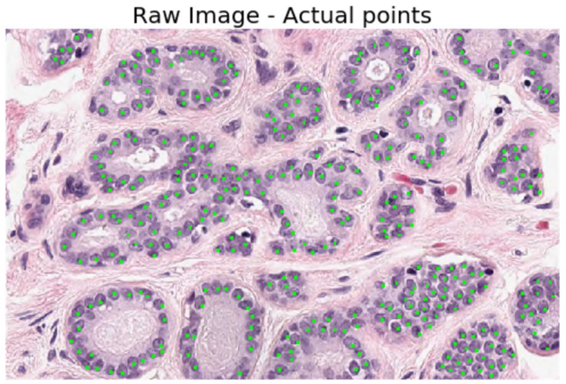
- They use a sliding classifier to create a heatmap of cancer probability
- Both about cancer detection, but they detect tumors, not individual cells
- They had a much larger training set
- Also designed to work on segmented data, but can be more easily fit to this problem

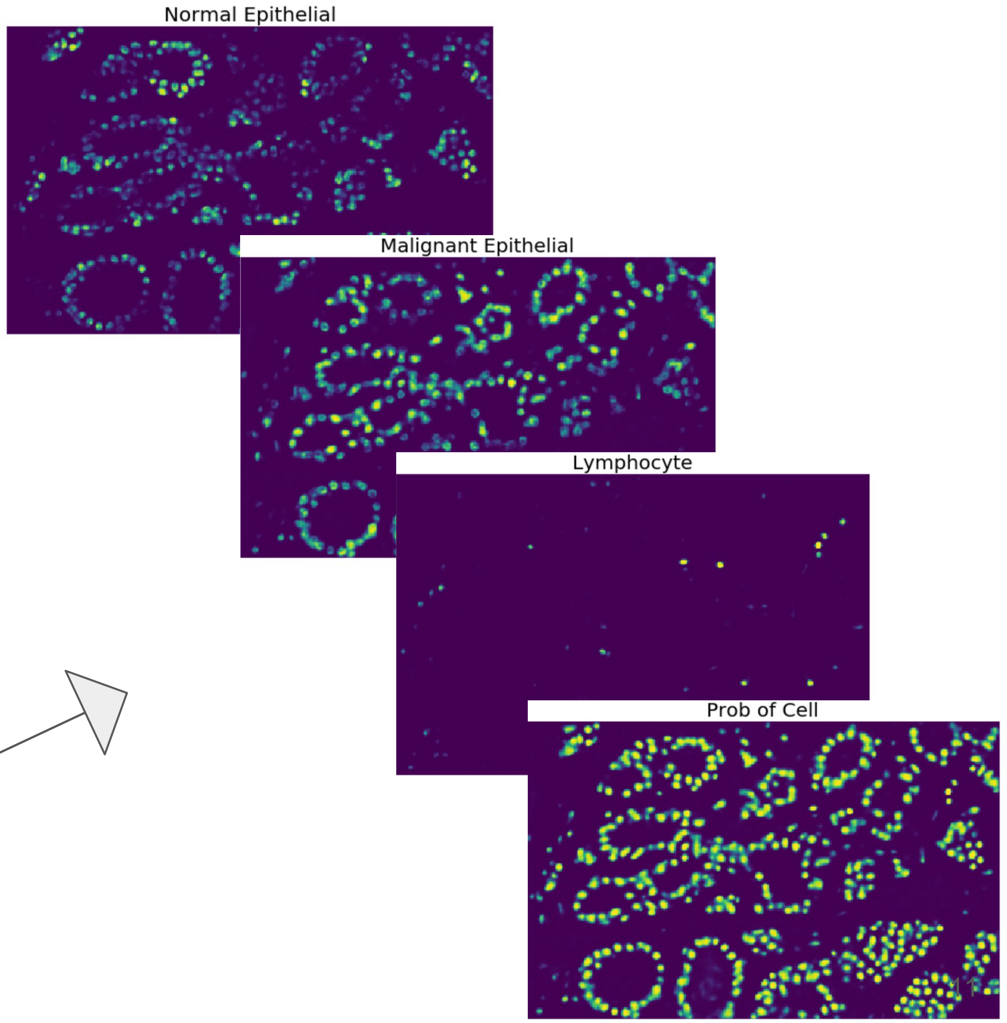


Example segmentation of cancer vs. non-cancer

Heatmaps

- Apply the trained classifier to the entire image, with a stride of 2.
- For each location the classifier is applied, we get 4 class probabilities





Non-Maximum Suppression

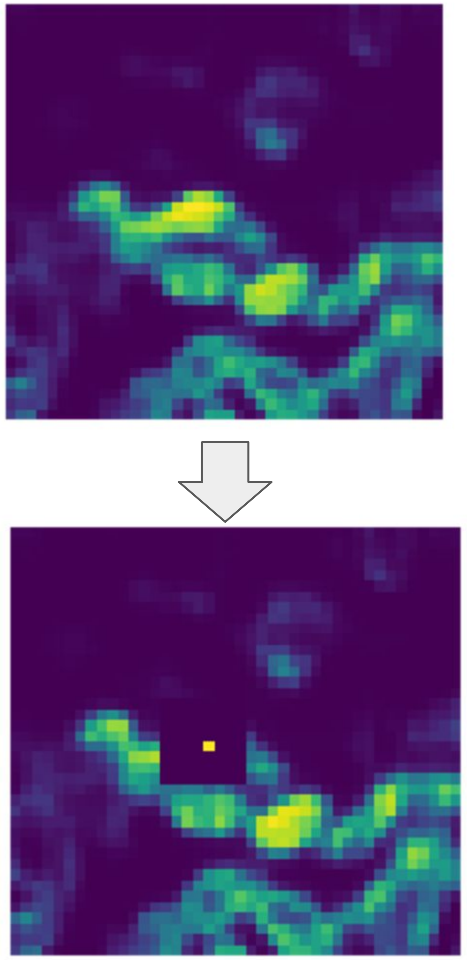
- How to turn this heatmap into point predictions for cell locations?

- Use the cell probability heatmap, and find the most likely cell locations

Procedure:

While max_cell_probability > cutoff:

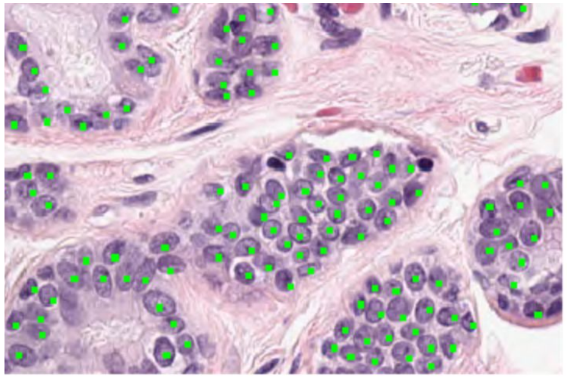
1. Find maximum pixel value (probability of nucleus)
2. Set all other pixel values within radius r to 0



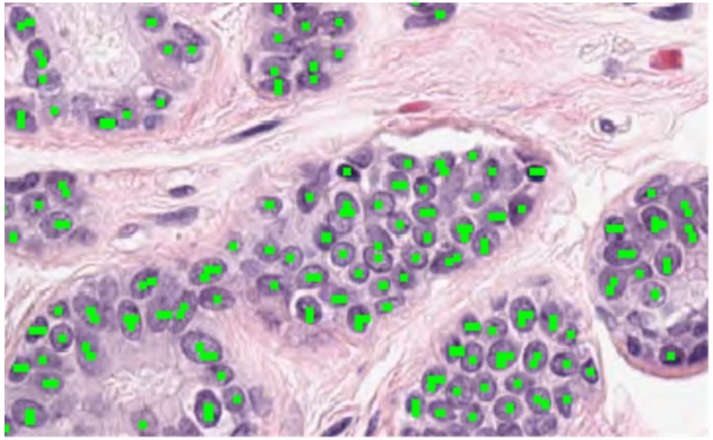
Choosing the radius

- Ideally all cells would be the same shape and size, r would be obvious

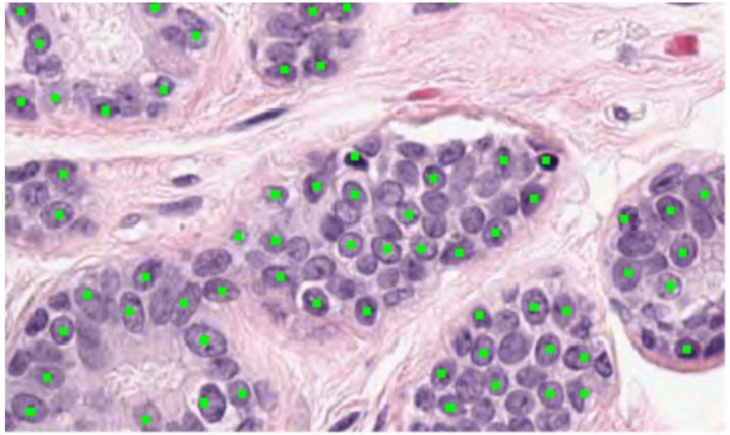
True labels



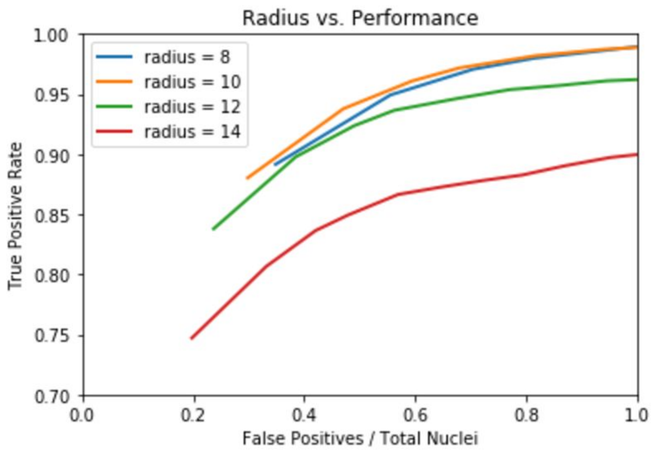
r = 4

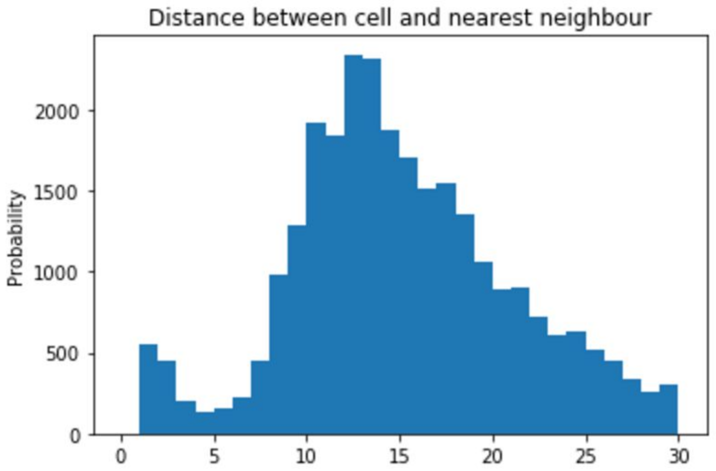


r = 16



Choosing the radius

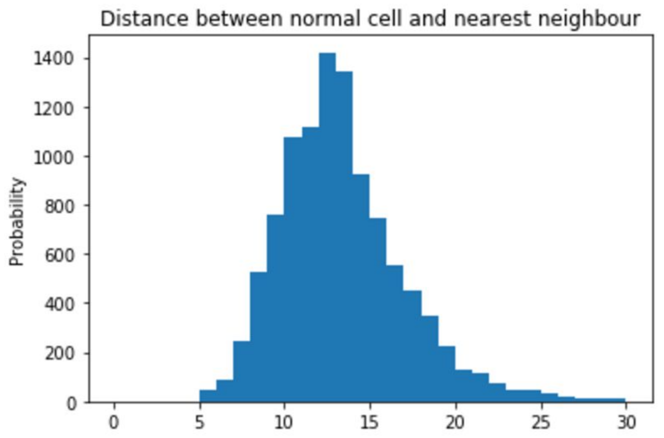


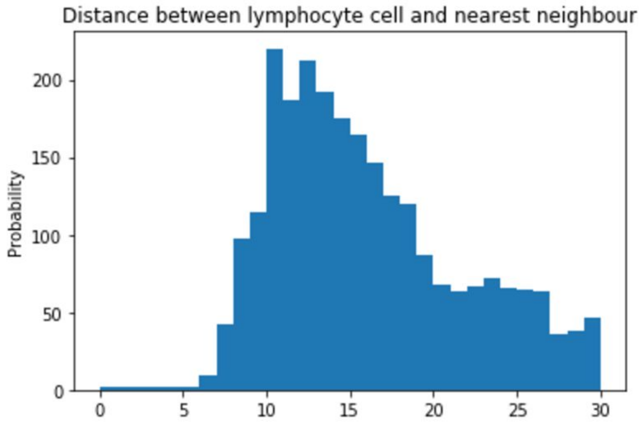


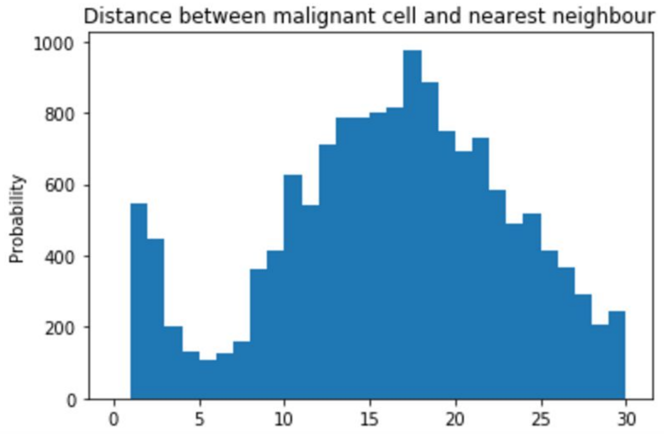
Radius of 10 is best

Radius conditional on class

The three cell types are different sizes, so we can choose the radius conditional on the class

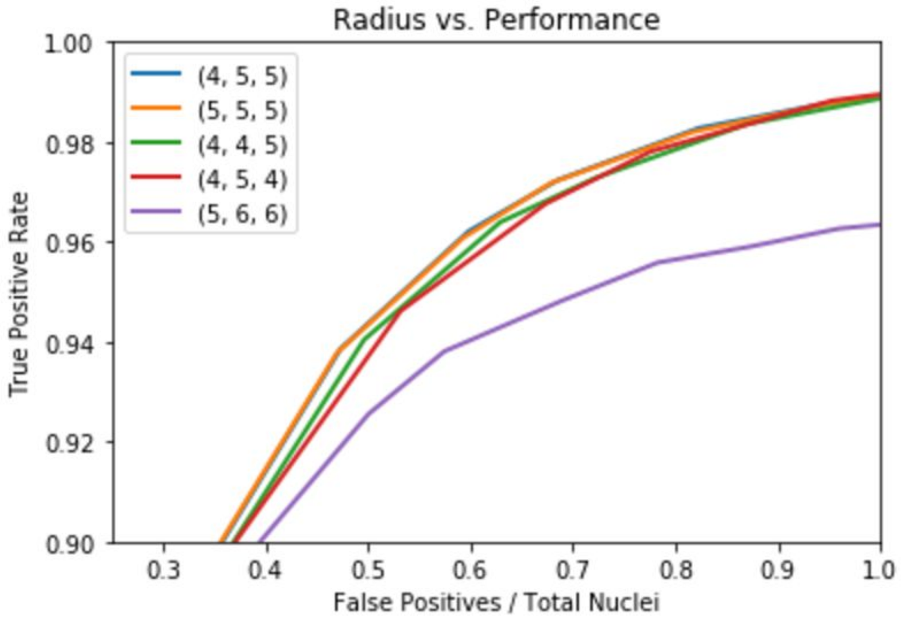






Radius conditional on class

- Histograms didn't show any clear results
- Testing out a few combinations also shows it is best to keep the radius as the same for all classes

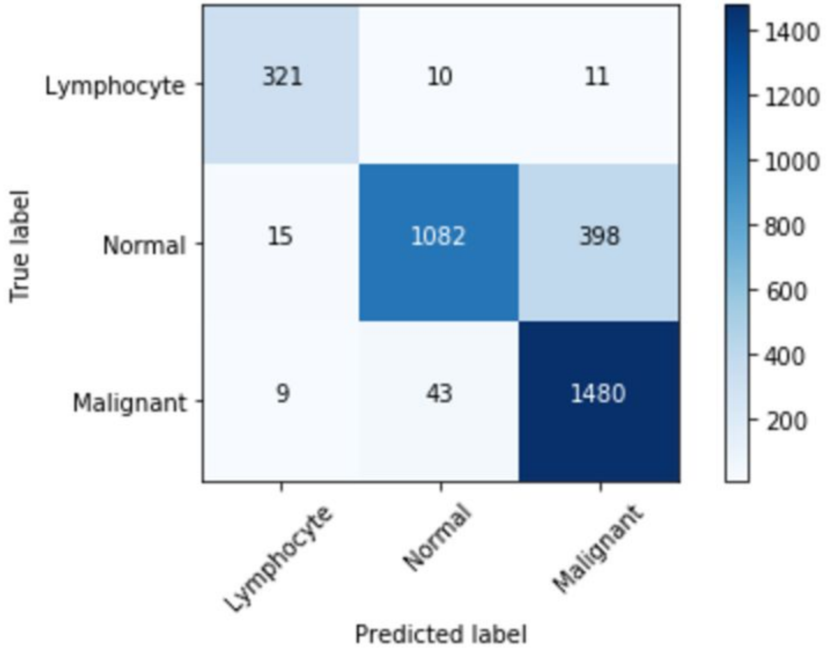


Basic Localization and Classification Results

Localization accuracy is difficult to assess:

- Unlabelled cells cause high false positive rate
- 88% of true cells detected
- 30% of those detected were false positives

Classification accuracy on segmented images: **85.6%**



Improving Classification

- People use more information than the 32x32 window surrounding the cell for classification.
- For example, it is unlikely to have a cluster of all normal cells with one malignant

CNN

- Try training the CNN on a larger image size
- Make a CNN to update the classifications after the heatmap is made

Smoothing

- Useful if we think distance is the most important factor
- KNN
- Laplacian Smoothing

Results

Bigger CNN

- 64 x 64
- Better classification accuracy: 90%
- Worse localization accuracy
  - 88% of true cells detected
  - 34% of those detected were false positives

KNN

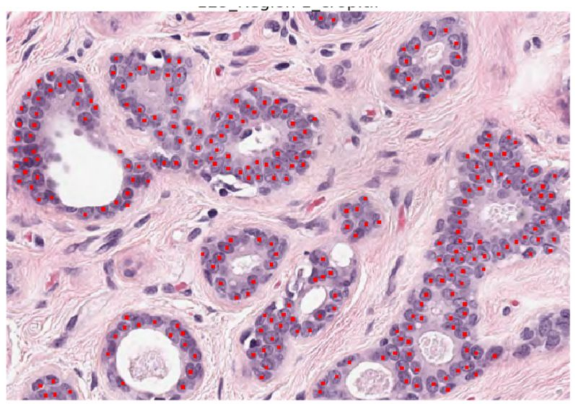
- Terrible

Neighbours	Accuracy
1	85%
2	67%
3	59%
4	59%
5	56%

CNN on Heatmap

If other cell's class and location are the most relevant information for classification, we can:

- Downsample the cell probability heatmap
- Training set will be cropped cell probability maps, centered at the cell we want to update
- Non-maximum suppression is still used to generate locations
- Output classification is classifier output at cell



CNN on heatmap

- Not effective

- Similar accuracy to taking max heatmap value

Other uses:

- Possibly useful for other problems where the sparsity of information may be even greater

- Very high resolution images that can't be fully fed into CNN

Conclusion and future work

- Deep learning is at least as good as traditional ML methods for this problem
- Information within and between cells is relevant for classification
- Issue is in taking into account global information, without drowning out the important local information of the small cells
- For sparser higher resolution images, it may be useful to look at this as an attention problem. Focus on a few areas of importance and combining it together without applying the CNN model to every pixel.