Level 4 Project

Week 3/16

Schedule for weeks 2-3 (15-16)

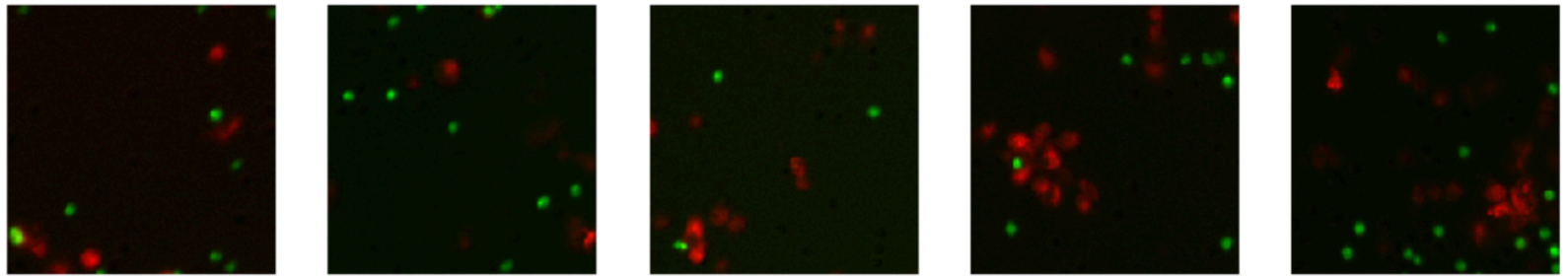
- Complete autoencoder tuning and image pre-processing.
  - Deliverables:
    - Code for an autoencoder model
    - Code for pre-processing
    - Explanation for the choices made
- Collate results obtained by research for evaluation
  - Deliverables:
    - Detailed evaluation plan
    - Report on current research methods for evaluating interactions
    - Excel tables of research metrics that can be methodically parsed
- Finalise image segmentation techniques
  - Deliverables:
    - Code for the image segmentation techniques, along with an explanation of why it was chosen

Autoencoder model

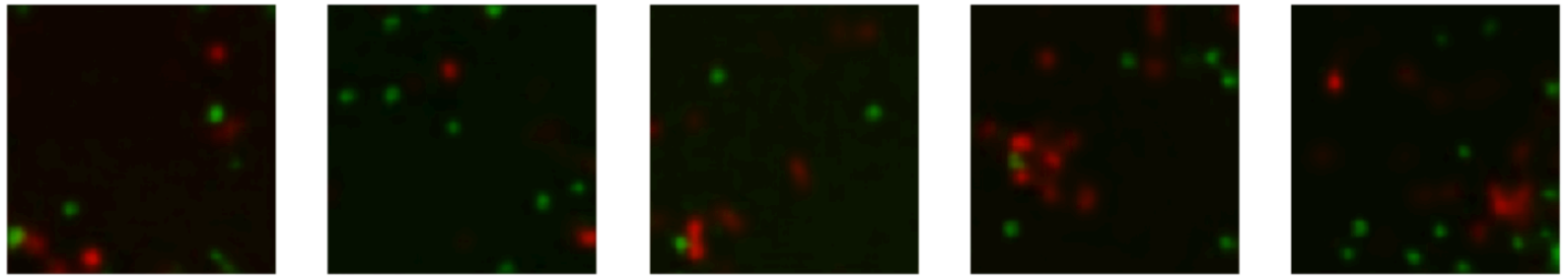
```
c = 3

input_img = Input(shape=(imw, imh, c))

x = Conv2D(64, (3, 3), padding='same')(input_img)
x = PReLU()(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(32, (3, 3), padding='same')(x)
x = PReLU()(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(32, (3, 3), padding='same')(x)
x = PReLU()(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(16, (3, 3), padding='same', strides=2)(x)
x = PReLU()(x)

encoded = Flatten()(x)

x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), padding='same')(x)
x = PReLU()(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), padding='same')(x)
x = PReLU()(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(64, (3, 3), padding="same")(x)
x = PReLU()(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(c, (3, 3), activation='sigmoid', padding='same')(x)

decoder = Model(input_img, decoded)
decoder.compile(optimizer='adam', loss='binary_crossentropy')
```

Striding comparison

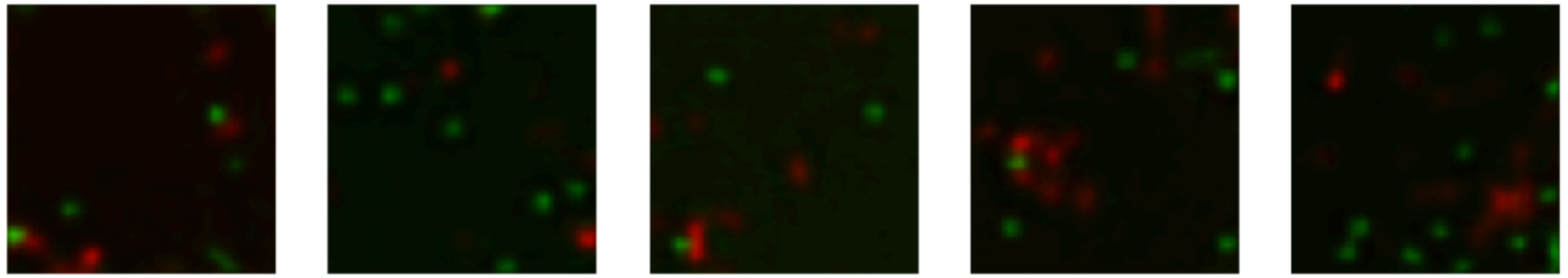
```
[46]  plot_range(x_combined, rn=520)
```



```
[45]  # structure with strides
      plot_range(decoded_imgs, rn=520)
```



```
      # structure without strides
      plot_range(decoded_imgs, rn=520)
```

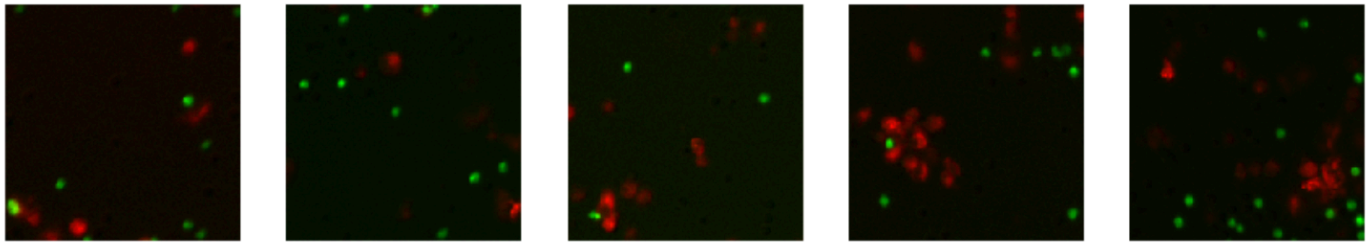


- Difference is minimal but seems slightly brighter
- Captures some overlap better

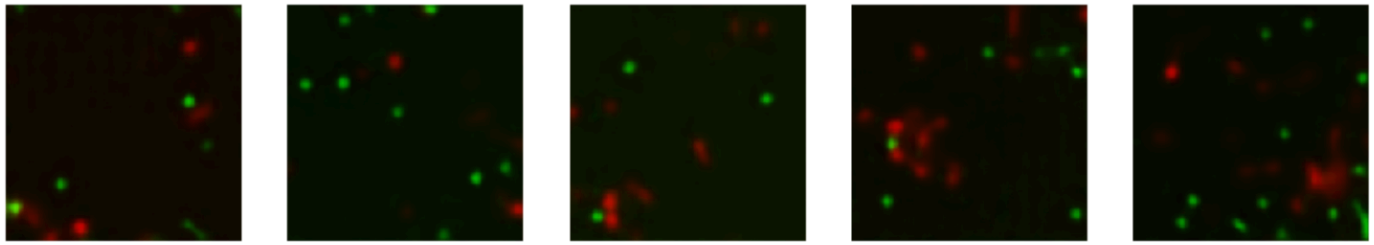
How big should the convolutional layers be?

- Tried both decreasing feature size and increasing feature size
  - Very similar performance but decreasing feature size has smaller dimensions once encoded, hence my choice
- Slightly bigger features has better performance
- Final dimensions are 2,302

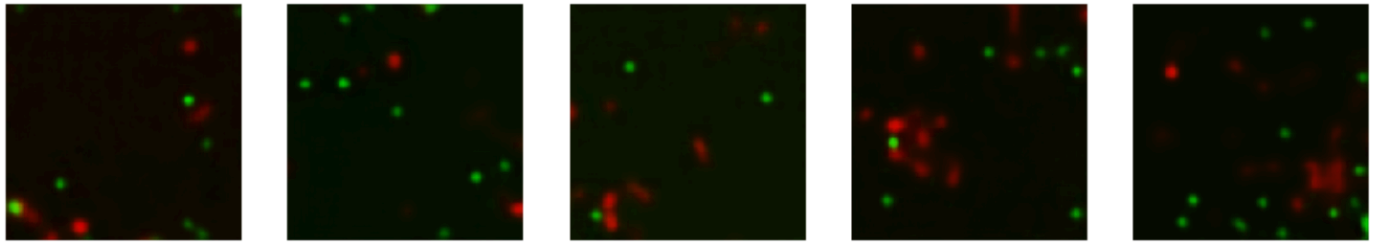
- original



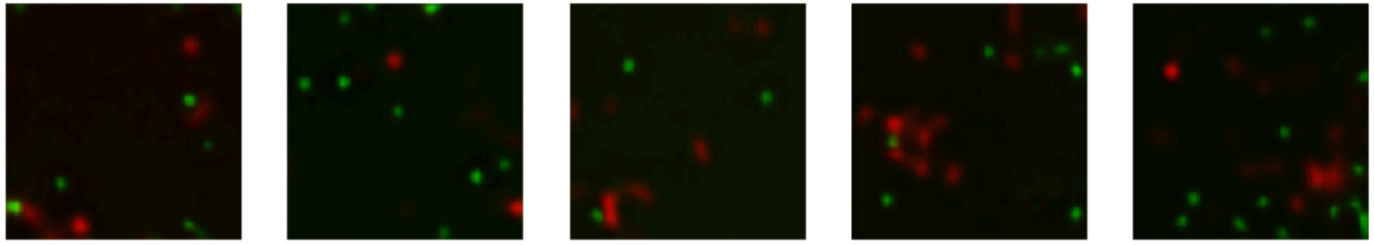
```
[66]  # structure with strides, bigger
      plot_range(decoded_imgs, rn=520)
```



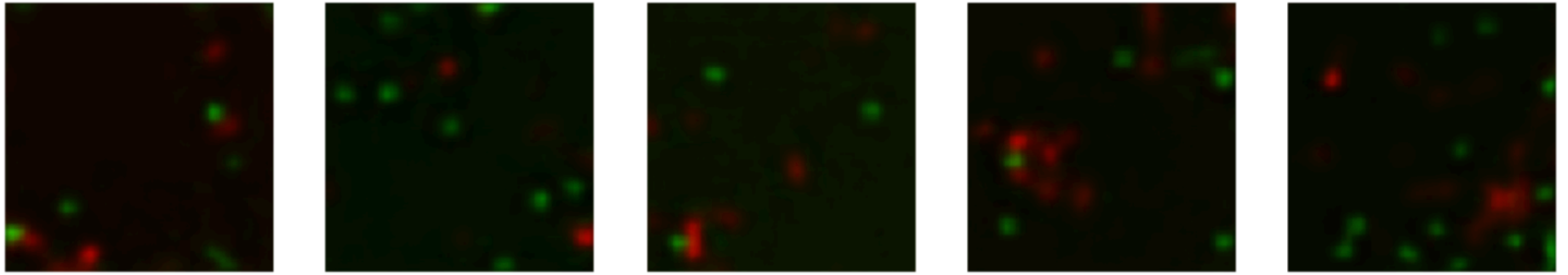
```
[59]  # structure with strides, inverted features, bigger
      plot_range(decoded_imgs, rn=520)
```



```
[55]  # structure with strides, inverted features
      plot_range(decoded_imgs, rn=520)
```



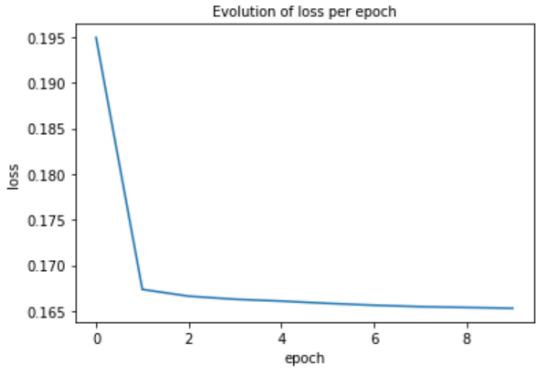
- No strides, smaller size

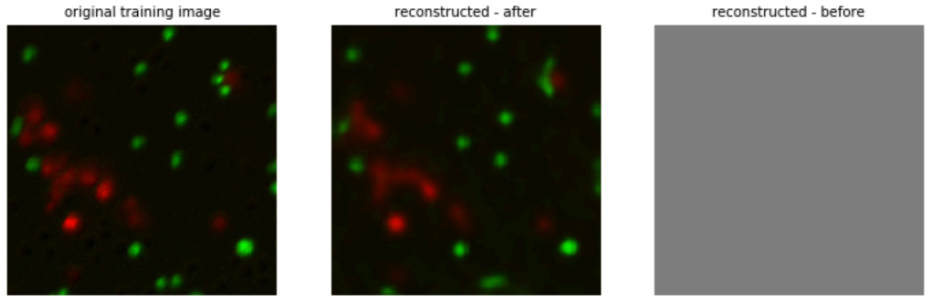


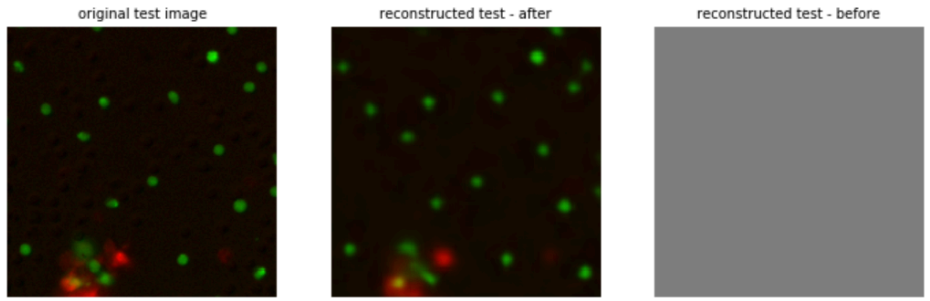
- Highlights improvements achieved

PReLU vs. ReLU

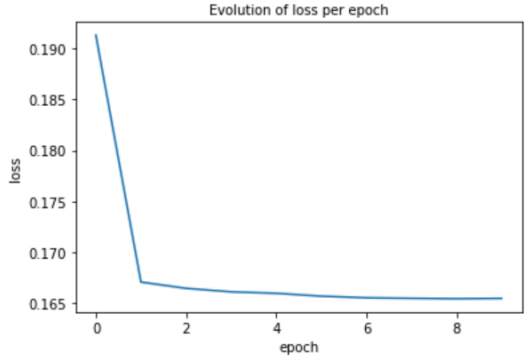
```
9600/9600 [==============================] - 37s 4ms/step - loss: 0.1653
Weight difference: [ 0.25939307 -0.04599345]
```

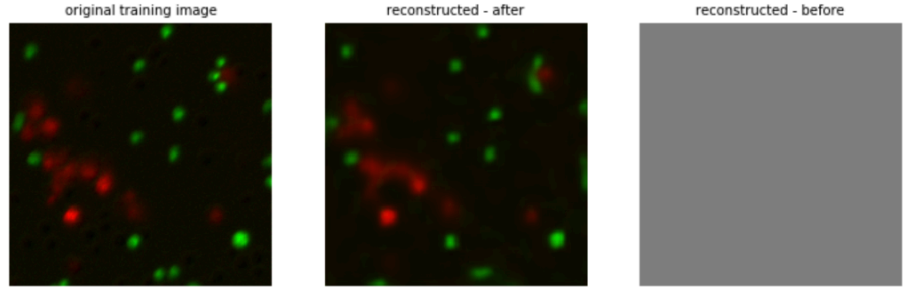


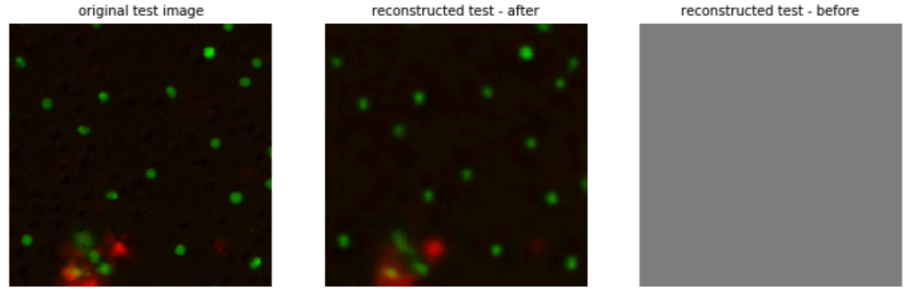





```
Epoch 10/10
9600/9600 [==============================] - 28s 3ms/step - loss: 0.1655
Weight difference: [ 0.19250129 -0.05974737]
```







PReLU vs. ReLU

- Very similar
- PReLU seems slightly better
- PReLU has been shown to outperform ReLU

  ”Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” He et al.

  “Empirical Evaluation of Rectified Activations in Convolutional Network,” Xu et al.)

Why this autoencoder?

- Not many details in the images to captures means we can reduce the size of feature maps
  - Similar research used similarly structured CNNs

  “Robust and accurate quantification of biomarkers of immune cells in lung cancer micro-environment using deep convolutional neural networks”

  “Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis”
  - Simpler has been shown to work well

  “Deep learning nuclei detection: A simple approach can deliver state-of-the-art results“

- Did testing to see what worked well for *this* dataset

Processing code

```
def preprocess(data, labels, mask=False):
  # avoid changing the dataset directly
  data = np.copy(data)

  # initialise arrays for filling in
  x_data = np.ndarray(shape=(len(data)//2, 192, 192, 3))
  y_data = np.ndarray(shape=(len(data)//2))

  # initialise index values
  idx = 0
  i = 0

  # loop through images and process
  while idx < (len(data)):
    # ignore 100, 300, etc. values as they will already have been p
rocessed
    if (idx % 100 == 0) and (idx % 200 != 0):
      idx += 100
    else:
        # if the image is "faulty" we cannot low_clip and apply min
max -> NaN
        if is_faulty(data[idx]) or is_faulty(data[idx+100]):
            tcell = minmax(data[idx])
            dcell = minmax(data[idx+100])
            y_data[i] = 3
        else:
            tcell = minmax(low_clip(data[idx]))
            dcell = minmax(low_clip(data[idx+100]))
            y_data[i] = labels[idx]

        # mask out the background
        if mask:
          x_data[i, ..., 0] = dcell*get_mask(dcell) # red-coloured
          x_data[i, ..., 1] = tcell*get_mask(tcell) # green-
coloured
        else:
          x_data[i, :, :, 0] = dcell
          x_data[i, :, :, 1] = tcell
        idx+=1
        i+=1

        # try and save memory
        tcell = None
        dcell = None

  print('Images preprocessed. Size of dataset: {}'.format(len(x_dat
a)))
  return x_data, y_data
```

Collating results for evaluation

- DMSO_metrics
  - DMSO from CK19, CK21, CK22
  - Question: Can the algorithm differentiate between 3 categories where differences should be most noticeable?
- CK19_metrics
  - 6 sets of each unstimulated, OVA, ConA
  - Question: Can the algorithm differentiate between 3 categories which all have the same number of data points?
- CK22_metrics
  - Mix of unstimulated and OVA
  - Question: Can the algorithm differentiate between two sets, if not three?
- Question
  - Difference between compound concentration and drug?
    - E.g. Compound Conc. uM at 10 vs. ConA (5 ug/ml)

Evaluation plan

- https://github.com/leonore/l4-project/blob/master/data/evaluation_plan.md
- Artifacts to evaluate:
  - Autoencoder
  - Clustering
  - Image segmentation

Sklearn vs OpenCV k-means

- OpenCV is 12x faster
- Same performance

OpenCV k-means vs thresholding

- Thresholding is 28x faster
- But masks obtained are not the same

Next steps for this week

- Work on getting all datasets finalised
  - Try reducing memory footprint by changing the datatype
  - Might have to reduce dataset by half (much bigger than DMSO)
- Immunology side:
  - Organise session to sit at GE software and take notes
  - Read provided articles
- Look at practical uses of UNet

Next steps for the coming week

- Ahead of schedule
- Jupyter notebooks → Python files
- Live visualisation for outlier images on a graph
- Start evaluating, possibly