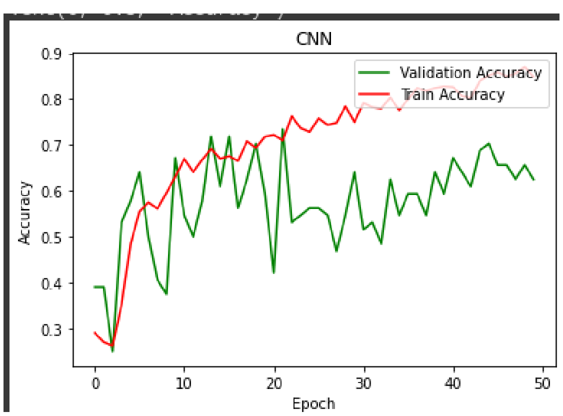
Bioimage Informatics_Hw3

1. The training and validation accuracy increased significantly after each epoch. The architecture of the network included. To gain a higher accuracy, just by simply increasing the number of epochs (to 50) boosted the training values. Also, implementing three linear layers helped increase the accuracy. It is mostly because the depth of a neural network helped the network to learn more, understanding the weights and biases that helps the nn to predict more accurately. Having the epoch number allowed the neural network to learn more by backpropagation and learning the weights that work. Further increasing the weight would increase the accuracy more. The default learning rate of 0.001 for Adam optimizer was used, and that boosted the accuracy significantly for the model. Max pooling was applied, as it selected the brighter pixels from the image. The training, validation, and test accuracy was 84.88%, 62.50% and 77.41% (which is projected to increase with epoch).

```
|Epoch: 35 | Train loss: 0.436 | Train Acc: 77.58% | Val Loss : 0.956 | Val Acc: 54.69% | Time used: 12s |
|Epoch: 36 | Train loss: 0.441 | Train Acc: 79.98% | Val Loss : 0.662 | Val Acc: 59.38% | Time used: 11s |
|Epoch: 37 | Train loss: 0.408 | Train Acc: 82.37% | Val Loss : 0.717 | Val Acc: 59.38% | Time used: 11s |
|Epoch: 38 | Train loss: 0.415 | Train Acc: 81.75% | Val Loss : 0.772 | Val Acc: 54.69% | Time used: 11s |
|Epoch: 39 | Train loss: 0.407 | Train Acc: 82.37% | Val Loss : 0.815 | Val Acc: 64.06% | Time used: 11s |
|Epoch: 40 | Train loss: 0.397 | Train Acc: 82.75% | Val Loss : 0.752 | Val Acc: 59.38% | Time used: 13s |
|Epoch: 41 | Train loss: 0.383 | Train Acc: 82.60% | Val Loss : 0.724 | Val Acc: 67.19% | Time used: 11s |
|Epoch: 42 | Train loss: 0.399 | Train Acc: 80.67% | Val Loss : 0.772 | Val Acc: 64.06% | Time used: 11s |
|Epoch: 43 | Train loss: 0.398 | Train Acc: 80.29% | Val Loss : 0.884 | Val Acc: 60.94% | Time used: 11s |
|Epoch: 44 | Train loss: 0.373 | Train Acc: 83.91% | Val Loss : 0.639 | Val Acc: 68.75% | Time used: 12s |
|Epoch: 45 | Train loss: 0.349 | Train Acc: 85.15% | Val Loss : 0.843 | Val Acc: 70.31% | Time used: 11s |
|Epoch: 46 | Train loss: 0.349 | Train Acc: 85.80% | Val Loss : 0.700 | Val Acc: 65.62% | Time used: 11s |
|Epoch: 47 | Train loss: 0.345 | Train Acc: 84.95% | Val Loss : 0.658 | Val Acc: 65.62% | Time used: 11s |
|Epoch: 48 | Train loss: 0.331 | Train Acc: 85.57% | Val Loss : 0.727 | Val Acc: 62.50% | Time used: 11s |
|Epoch: 49 | Train loss: 0.324 | Train Acc: 86.96% | Val Loss : 0.736 | Val Acc: 65.62% | Time used: 12s |
|Epoch: 50 | Train loss: 0.308 | Train Acc: 84.88% | Val Loss : 0.651 | Val Acc: 62.50% | Time used: 11s |
| Test Loss: 0.493 | Test Acc: 77.41% |
```

The validation loss helps guide to select a trained model before over fitting happens. The training accuracy gradually increases over each epoch and the rate of increase of validation accuracy also slowly decreases and the best accuracy value point is chosen, to select the model, on which the test data is applied. And this helps prevent overfitting of the data. Around 20 epochs, the validation accuracy seems good. The training accuracy for the model is above 70% as well.



2.
Autoencoder: The loss values are observed to decrease.

The architecture initially consisted of two convolutional layers (referenced from the slides), which resulted in the following loss below. Using a loss criterion MSE (mean squared error), facilitates lower loss compared to the loss criterion of BCE. Secondly, using a tanh() function also helped to reduce the loss.

```
cpu
| Epoch: 1 | Train Loss: 0.0047 | Valid Loss : 0.0035 | Time: 47
| Epoch: 2 | Train Loss: 0.0035 | Valid Loss : 0.0020 | Time: 47
| Epoch: 3 | Train Loss: 0.0031 | Valid Loss : 0.0030 | Time: 45
| Epoch: 4 | Train Loss: 0.0030 | Valid Loss : 0.0023 | Time: 45
| Epoch: 5 | Train Loss: 0.0029 | Valid Loss : 0.0021 | Time: 47

| Test Loss: 0.002 |
```

The autoencoder architecture:

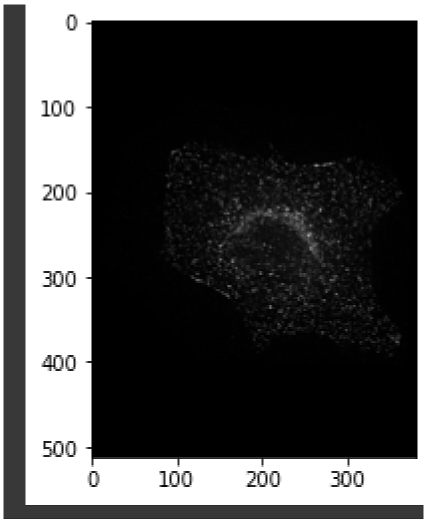
```
class Autoencoder(nn.Module):
  def __init__(self):
    super(Autoencoder, self).__init__()
    self.encoder = nn.Sequential(nn.Conv2d(3, 6, kernel_size = 2,stride = 1),
    nn.ReLU(True),nn.Conv2d(6,12, kernel_size =2, stride =1), nn.ReLU(True),
    nn.Conv2d(12, 24, kernel_size = 2, stride =1), nn.ReLU(True)
    )

    self.decoder = nn.Sequential(nn.ConvTranspose2d(24, 12, kernel_size = 2, stride =1),
    nn.ReLU(True),
    nn.ConvTranspose2d(12, 6, kernel_size = 2, stride =1), nn.ReLU(True),
    nn.ConvTranspose2d(6,3, kernel_size =2, stride = 1), nn.ReLU(True), nn.Tanh()
    )
```

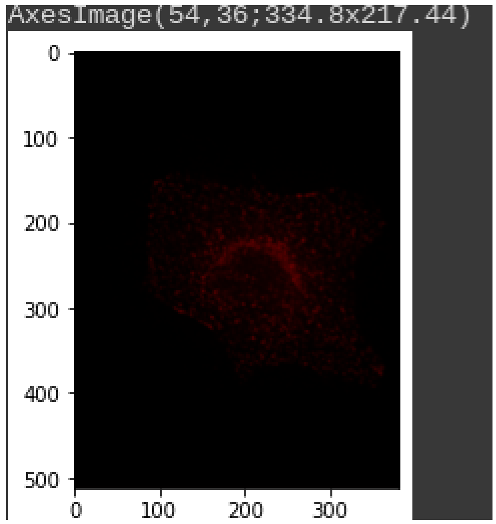
When one extra convolutional layer was added (nn.Conv2d(12, 24)), the loss was observed to decrease further, and hence the architecture chosen consisted of three conv2D layers, accompanied with ReLU activation, as it only considers distinct features of the image (choosing max between 0,1).
The autoencoder was trained using Adam optimizer, with the default learning parameter of 0.0001. Although, having a low learning rate, can take computationally longer time but helps to find and improve the model performance and lower the loss. The number of epochs was kept at 5, as it was tested from a cpu device.

```
cpu
| Epoch: 1 | Train Loss: 0.0031 | Valid Loss : 0.0021 | Time: 327
| Epoch: 2 | Train Loss: 0.0031 | Valid Loss : 0.0020 | Time: 96
| Epoch: 3 | Train Loss: 0.0029 | Valid Loss : 0.0022 | Time: 90
| Epoch: 4 | Train Loss: 0.0027 | Valid Loss : 0.0018 | Time: 92
| Epoch: 5 | Train Loss: 0.0024 | Valid Loss : 0.0017 | Time: 92
```

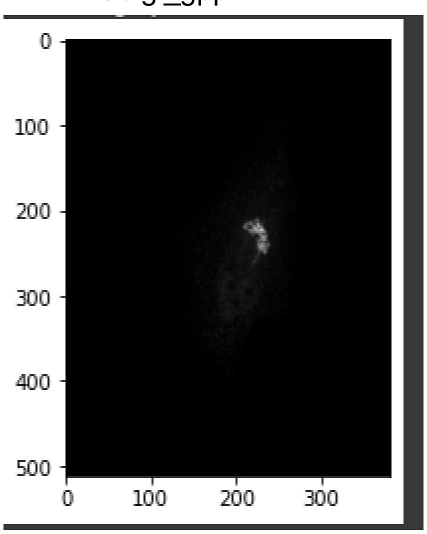
The image reconstruction was satisfactory, however, the colab image plotting system applied color on the reconstructed image (given that it was a 3-channel image).

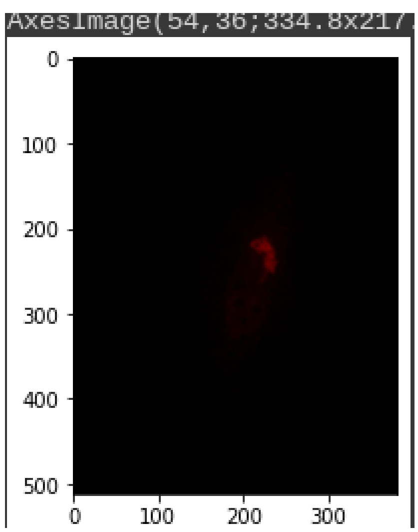
1. Mitochondria:



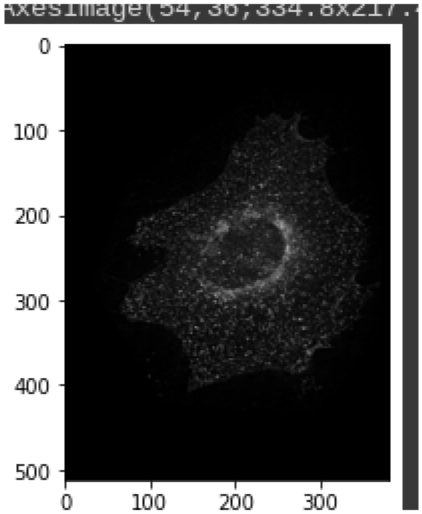


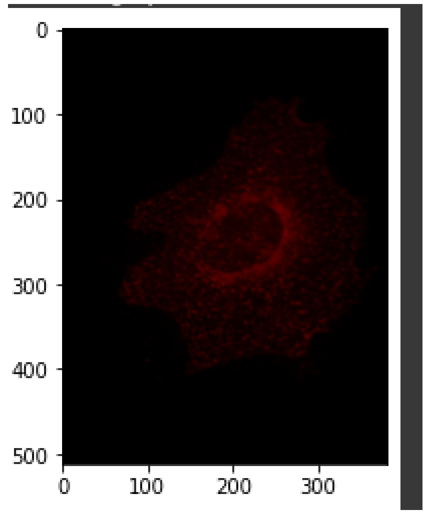
2. Golgi_gpp



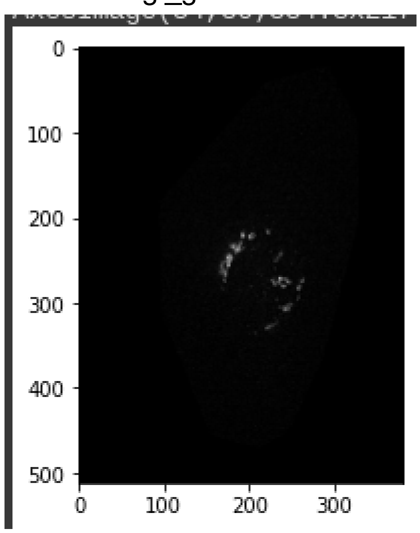


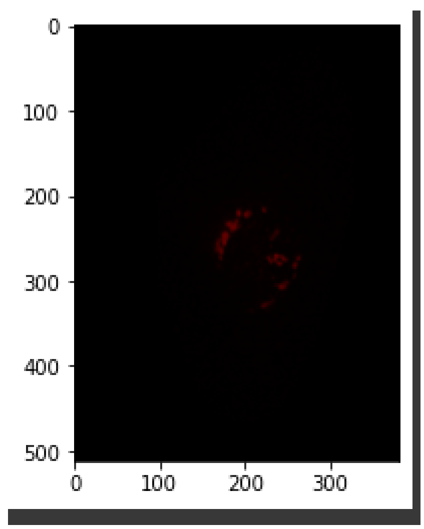
3. Endoplasmic Reticulum:



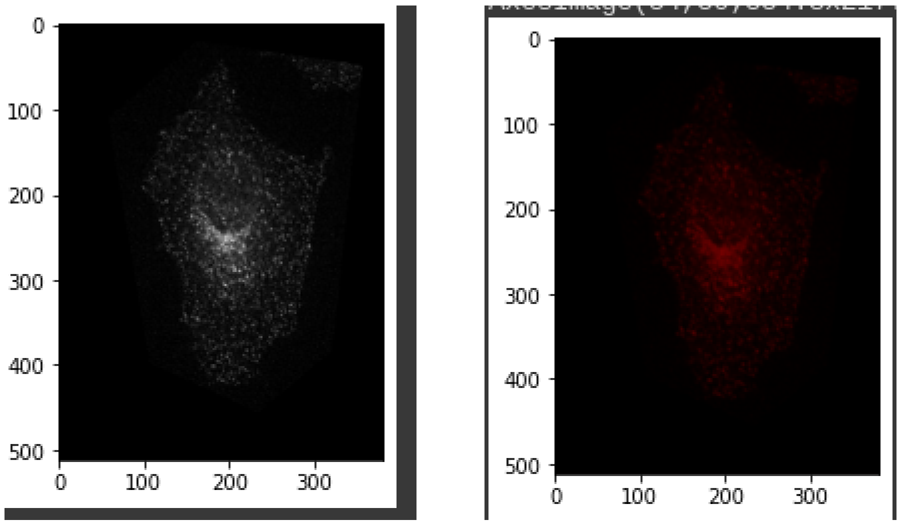


4. Golgi_gia:

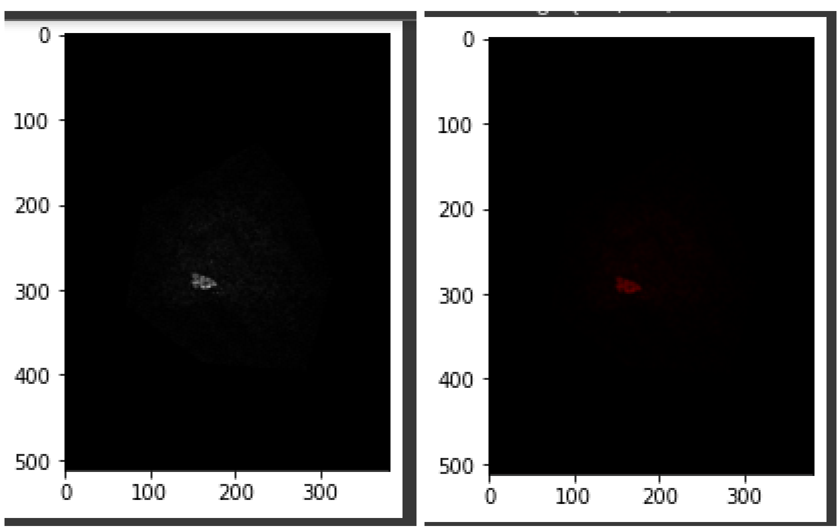




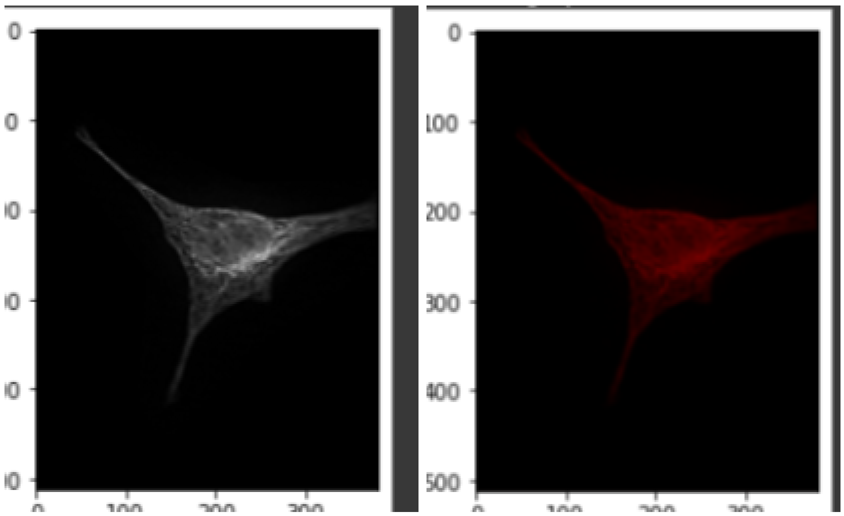
5. Endosome:



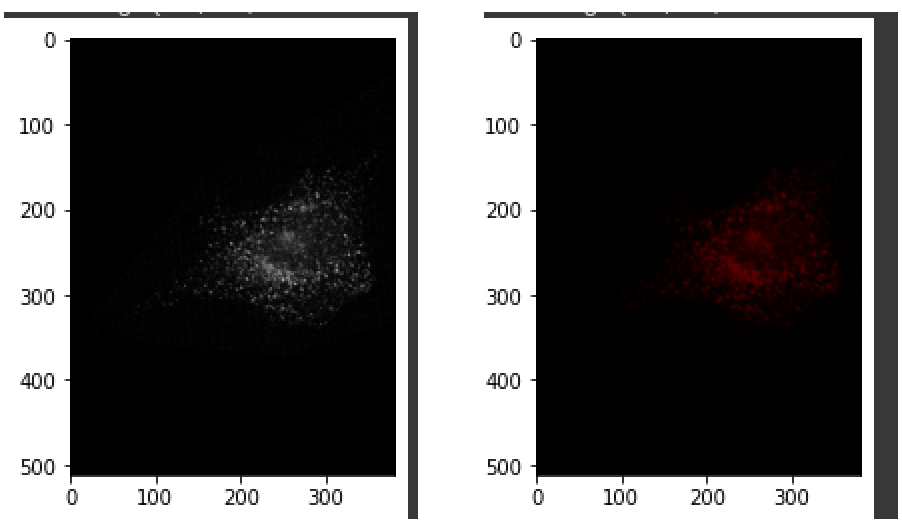
6. Nucleolus



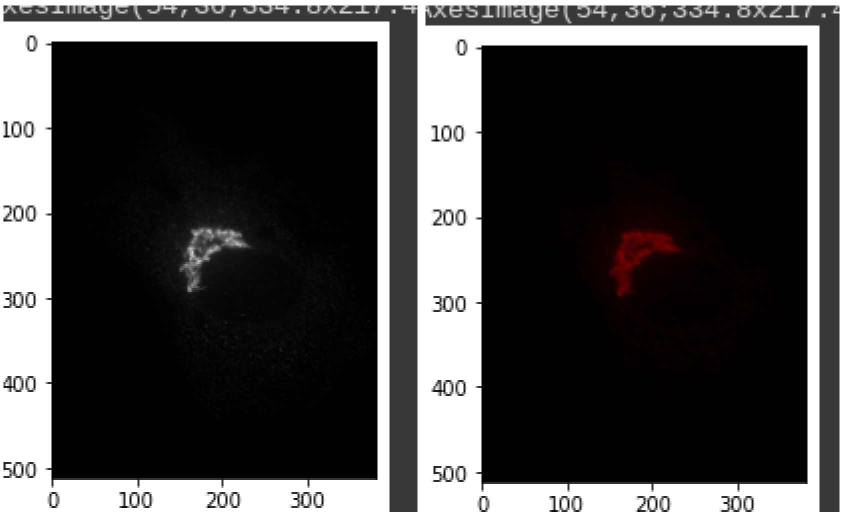
7. ActinFilaments



8. Lysosome



9. Nucleus:



10. Mircotubules