NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE

Developing Analytic Tools for Golgi Imaging

Student: Huang Jiadeng

Supervisor: Lu Lei

SCHOOL OF BIOLOGICAL SCIENCES

2021

CONTENT

Huang Jiadeng

Lu Lei

Assoc. Prof. Lu Lei

1. Background

1.1 GLIM

The mammalian Golgi complex is an essential organelle in posttranslational modifications and sorting of secretory/endocytic proteins and lipids. The basic structural unit of the Golgi is the Golgi stack, which comprises 4-11 tightly adjacent and flattened membrane sacs called cisternae and the Golgi stack is further divided into 3 regions, including the cis-, medial-, and trans-Golgi.

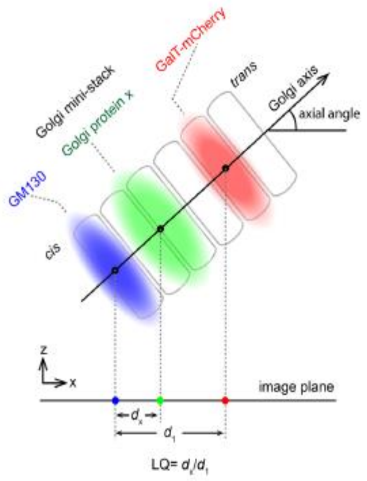


Figure 1 - Golgi Mini Stack

A novel imaging method for quantitative Golgi localization named Golgi protein localization by imaging centres of mass (GLIM) to study the localization of Golgi proteins rapidly, quantitatively, and systematically by conventional light microscopy had been developed by *Tie et al*, *2016*. In this method, cells are triply labelled by either endogenous or ectopically expressed GM130, the test protein, and β1, 4-galactosyltransferase-mCherry (GalT-mCherry) followed by Nocodazole treatment and imaged by a spinning-disk confocal or wide-field microscope as three-dimensional (3D) image stacks, GM130, GalT-mCherry serve as references to mark the cis- and trans-Golgi respectively. GM130, GalT-mCherry, and the test protein normally appear as tight adjacent single fuzzy spots in a ministack, which cannot be well spatially resolved under a conventional light microscope. In GLIM, Golgi ministacks are visually inspected, and those that display only one object in each channel are masked by regions of interest (ROIs), which will be selected for further analysis.

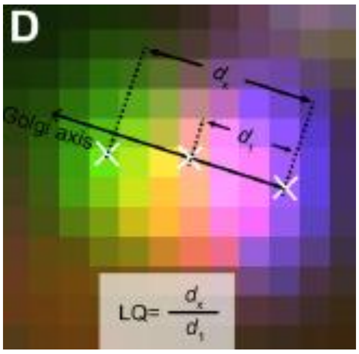


Figure 2 – Single Golgi

To localize the test protein at subpixel accuracy, the centre of fluorescence mass centre of the Golgi ministack within each ROI is calculated. The Golgi ministack axis as a vector from the centre of GM130 to that of GalT-mCherry in the image plane or 3D space (Figure 1). The Figure 2 shows how it looks like in the real micro image and Figure 3 shows an example of our final selection on the micro images.

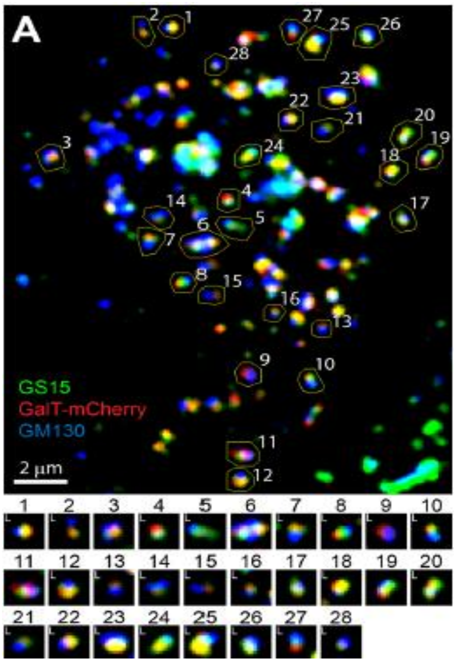


Figure 3 - Example of Final Selection

Normally, inside single cells, the Golgi may have various orientation, which makes them look differently under the microscope, while it may show the side view, oblique view, as well as the en face view (Figure 4). The orientation of a mini stack can be identified by Golgi markers, such as Giantin, Golgin84and GPP130. By assessing the staining patterns from the marked Golgi, the en-face and side view-oriented Golgi mini-stacks should appear as a ring and double-punctum (Figure 5) can be conveniently identified under a super-resolution microscope, such as Airyscan(Tie et al. 2018).

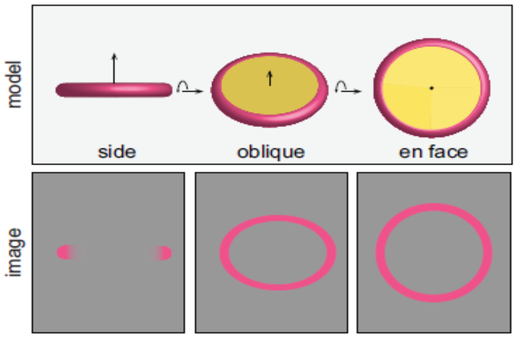


Figure 4

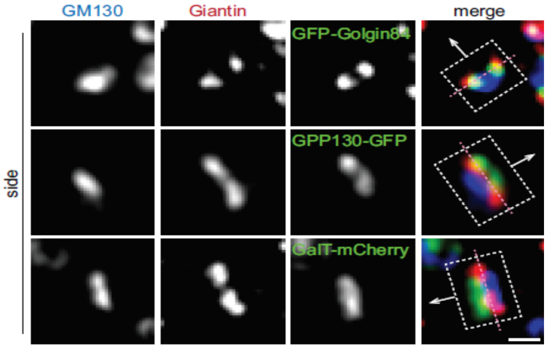


Figure 5

1.2 Object Detection Technique and CNN

The current procedures of GLIM are dependent on manual manipulation, hence it is time consuming, labour intensive and lack of efficiency. To develop the analytic tools for Golgi imaging, we planned to make this Golgi selection step as an automatic process, which is similar to most of the image classification and object localization problems.

The convolutional neural networks (CNN) is a Deep Learning algorithm which can take in an input image, assign importance to various aspects or objects by using learning weights and biases, and be able to differentiate one from the other. The classical composition of image classification neural networks is a group of CNN layers, which will handle the input image and extract features from pixels, and pass through a fully connected dense layers (DNN) for classification process (Figure 4). The proper loss function should be chosen during the training procedure to find the best parameters for the classification problems.

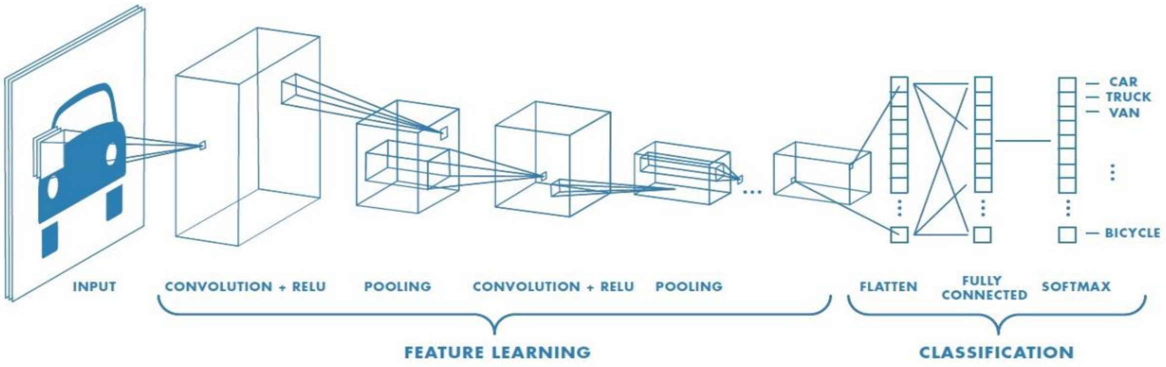


Figure 6 - CNN+DNN Architecture

In our solution, we will extract all the single objects from 40 micro images and build a CNN for further classification.

2. Image Generation & Data Pre-processing

2.1 Image Generation

Our source images include 3-channel images having the label as Cy5, GFP and Cy3, which represent the different Golgi markers that correspond to GM130, the test protein and GalT, inside the Golgi ministack. In those raw images, there is no intuitive pattern to distinguish the Golgi other objects as it is too abstract. The Figure 7 illustrates a sample source image at Cy3 channel.

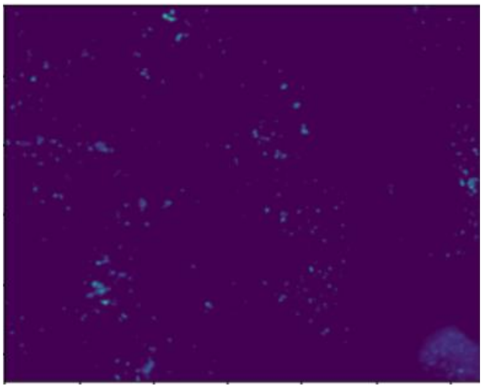


Figure 7 – Example of Cy3 Image

To fulfil our goal, there are a few image pre-processing steps needs to be done prior to further analysis: 1) The source images will be background subtracted. 2) The pixel value will be rescaled to appear as a binary image. 3) The 3 source images will be merged to generate a composite image. After that, the output image will be look like as Figure 8.

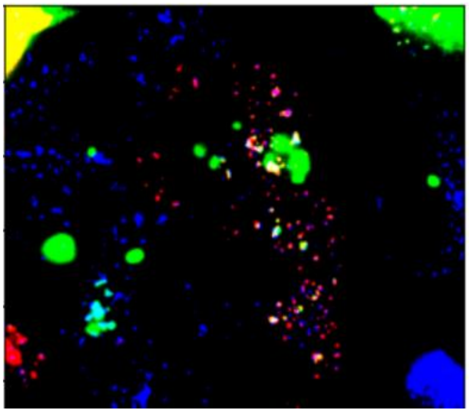


Figure 8 – Example of Merged Image

2.2 Object ROI Coordinates Extraction

While the image classification convolutional neural network can only allow one object at a time to pass through the network, it is necessary to extract all the objects inside the micro images into single image containing only single object and use it for further training and testing purposes. To fulfil this goal, we implemented the coordinates extraction as below per step:

1) Label all the area of objects in image:
   While the pre-processed images are all background subtracted, the background pixel all have the value as 0. In this step, all the pixels having non-zero value are considered as object pixel. A new binary image is generated accordingly, while this image only has pixel value as 0 or 1, representing the background pixel and object pixel respectively (Figure 9).

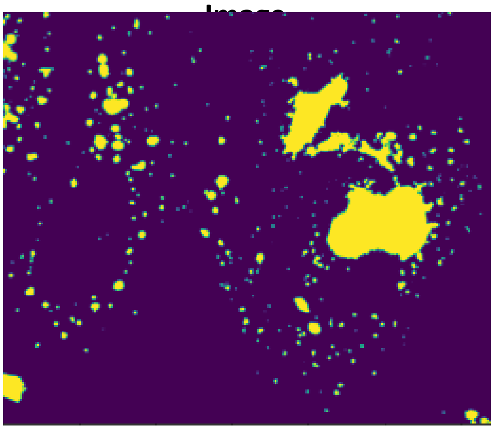


Figure 9 – Example of Binary Image

2) Filter the objects:
   While a Golgi ministack comprises 3 channels, an object which appears blank in any channel is filtered out.
3) Object boundary coordinates extraction:
   In this step, we have applied the Depth First Search (DFS) algorithm to update/extract the coordinates for all the Golgi objects and non-Golgi objects. The final result will be represented as 4 coordinate-related statistics: 1) Top: the topmost point of the pixel within an object; 2) Bottom: the bottommost

point of the pixel within an object; 3) Left: the leftmost point of the pixel within an object; 4) Right: the rightmost point of the pixel within an object. Furthermore, for Golgi object and non-Golgi object, we have different coordinates extraction strategy.

I. For Golgi Object, while those objects are all pre-selected using the GLIM methodology. The regions of interest (ROI) selected are all polygon-like shape, composed by a group of coordinates. Most of the ROI shape of Golgi objects are 16*16. The shape statistics of the pre-selected Golgi are as shown in the below table.

	Max	Min	Median	Mode
Height	42	16	16	16
Width	45	16	16	16

However, this was still a manual process, which makes the boundary formed by those coordinates are not perfectly matching the real boundary of the object. The DFS algorithm is applied here for update purpose only.

II. For non-Golgi Object: The objects we are going to use here is the pre-labelled and filtered object in the previous steps, which is not pre-selected by any prior knowledge, meaning the initial coordinates are unknown. As concluded from the shape statistics from Golgi objects, the largest Golgi has shape as 42*45. So, the DFS algorithm is applied here for the extraction within a 50*50 sliding window taking the stride as 2.

3. Model Selection

3.1 Dataset Preparation

Before the implementation of the training procedure, there is one thing to note that among the images, there were 4 images called as ‘beads’ image, which are some self-made objects to simulate how the Golgi object will be look like in GLIM methodology. However, during the training procedure, we have found that the objects of those image may cause some issue to our model training, while it caused overfitting problem if it is included in training set and caused overestimation of the result if it is included in training set. So, only 36 images were used finally.

The dataset has been further divided into test set and training set. The test set has totally 6 images, while the training set has 32 images. The training set is further divided into the validation set and training set, having the number of images as 26 and 6 respectively, while the validation set is more for us to monitor the training procedure and training set is the only set contributing to updating the model parameters. All the objects will be extracted by the coordinates identified and generate a new image with single object to be reshaped to 17*17 to pass into the neural network.

In the divided training set, the number of Golgi and non-Golgi objects are 966 and 1166, which is approximately balanced. Therefore, oversampling technique was not applied here. In the validation and testing dataset, the positive to negative ratio of the objects are both approximately 1:2, which is more like the real-world distribution of 2 kinds of different objects.

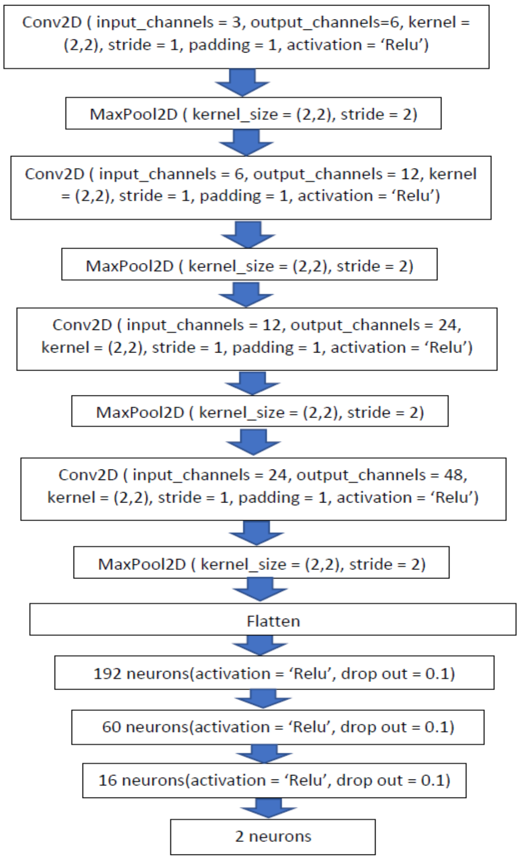

Conv2D (input_channels = 3, output_channels=6, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Conv2D (input_channels = 6, output_channels = 12, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Conv2D (input_channels = 12, output_channels = 24, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Conv2D (input_channels = 24, output_channels = 48, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Flatten
192 neurons(activation = 'Relu', drop out = 0.1)
60 neurons(activation = 'Relu', drop out = 0.1)
16 neurons(activation = 'Relu', drop out = 0.1)
2 neurons

Figure 10 – Model 1 Architecture

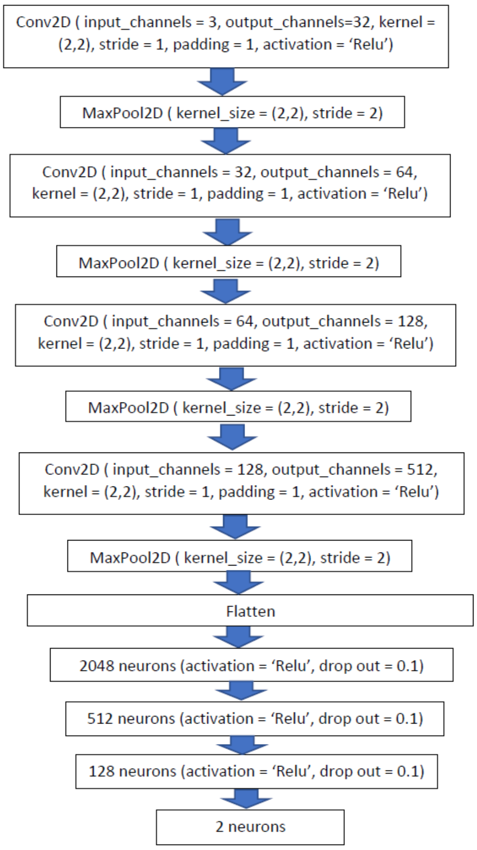

Conv2D (input_channels = 3, output_channels=32, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Conv2D (input_channels = 32, output_channels = 64, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Conv2D (input_channels = 64, output_channels = 128, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Conv2D (input_channels = 128, output_channels = 512, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Flatten
2048 neurons (activation = 'Relu', drop out = 0.1)
512 neurons (activation = 'Relu', drop out = 0.1)
128 neurons (activation = 'Relu', drop out = 0.1)
2 neurons

Figure 11 – Model 2 Architecture

3.2 Model Structure

3.2.1 Model 1

The 1st model trained is considered as a base model, while it has too few features generated in the CNN part comparing with most of the widely used neural networks. The architecture is shown in Figure 10. The input image will pass through 4 convolutional neural network layers, using the kernel size as 2, stride as 1, padding as 1 and activation function as Relu. Each CNN layer is also followed by a Max Pool 2D layer using the kernel size as 1 and stride as 2. After that, the original 3 RGB channels will be transformed to 48 channels by those CNN layers, and will pass through a flatten layer for the Dense layers to execute the classification task. There are totally 3 dense layers with dropout rate as 0.1, having the neurons count as 192, 60 and 16, connecting with the flatten layer, as well as the output layer, which has 2 neurons corresponding to the probability of negative and positive.

However, during the training procedure, the model 1 had an overall poor performance. Therefore, further improvement on the model structure has to be carried out further.

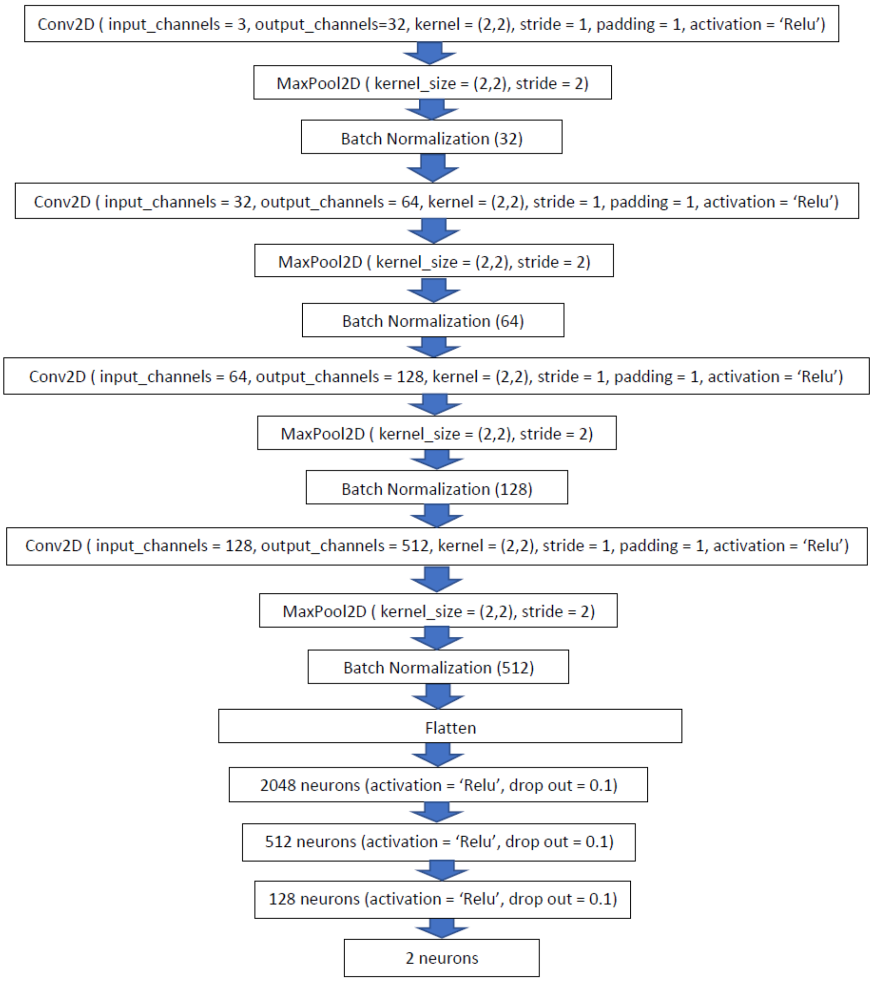

Conv2D (input_channels = 3, output_channels=32, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Batch Normalization (32)
Conv2D (input_channels = 32, output_channels = 64, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Batch Normalization (64)
Conv2D (input_channels = 64, output_channels = 128, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Batch Normalization (128)
Conv2D (input_channels = 128, output_channels = 512, kernel = (2,2), stride = 1, padding = 1, activation = 'Relu')
MaxPool2D (kernel_size = (2,2), stride = 2)
Batch Normalization (512)
Flatten
2048 neurons (activation = 'Relu', drop out = 0.1)
512 neurons (activation = 'Relu', drop out = 0.1)
128 neurons (activation = 'Relu', drop out = 0.1)
2 neurons

Figure 12 – Model 3 Architecture

3.2.2 Model 2

The model 2 has significantly increased number of features to be extracted in the CNN part. The overall architecture of Model 2 is as shown in Figure 11. The number of an composition layers and the parameters including kernel size, stride, padding, etc, remain unchanged. The only difference comparing to model 1 is the number of channels (features) generated by the CNN and the number of neurons in DNN part.

3.2.3 Model 3

The model 3 is an adapted version of model 2, while it only added a batch normalization layer in the CNN part between each CNN layer and Max Pooling 2D layer. For the rest of the architecture and corresponding parameters are staying unchanged (Figure 12).

3.3 Training Procedure & Model Performance Evaluation

3.3.1 Training – Model 1

In model 1, 3 training procedures with different learning rate as 0.1, 0.01 and 0.001, and epoch as 20, 100 and 500 have been executed using the batch size as 20. The loss function applied here is Cross Entropy Loss. However, due to the features extracted in this model is not enough, the training with learning rate as 0.1 failed, while the output model has bad performance was not able to distinguish the positive and negative objects. The 2nd and 3rd version trained also only have the non-ideal performance, having the precision and recall around 60% and 40% respectively.

3.3.2 Training – Model 2

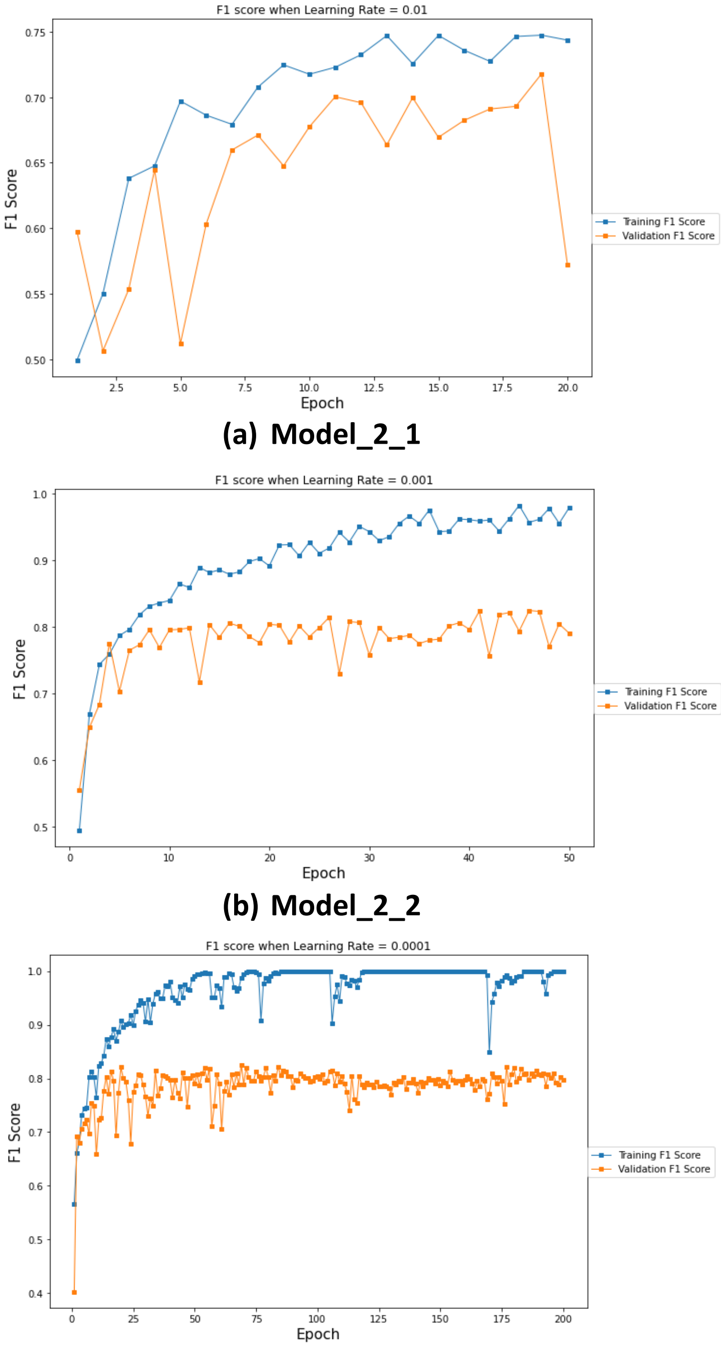
In the training procedure of model 2which uses the Cross Entropy loss function, a few more measurements were carried out to find the best parameters of the model, including fine tuning the leraning rate (0.1, 0.01 and 0.001) and epoch (20, 50 and 200), applying the early stopping, and add the l1 penalty in loss function. However, only the different learning rate and epoch illustrated outstanding differentiatable performance improvement other than the rest of the measurements. Therefore, we will only consider versions of models' performance trained by different learning rate and epoch and these models will be named as model_2_1, model_2_2 and model_2_3. In our case, we emphasize on the performance of classfying Golgi object, so we will only concern about the Precision, Recall, and F1 score (formula as Figure 13) for Golgi object recognition.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Figure 13 – Formula of Precision, Recall & F1 Score



(a) Model_2_1

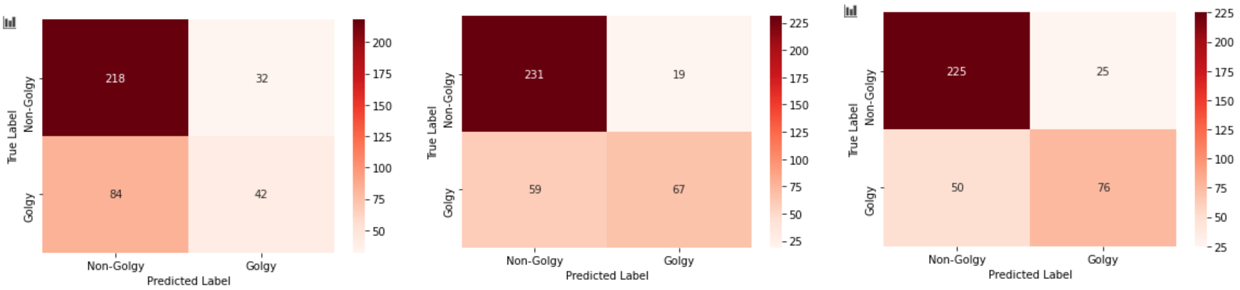
(b) Model_2_2

(c) Model_2_3

Figure 14 – Curve of F1 Score

In the experiment of GLIM, the False Positive samples could severely skew the localization of a Golgi protein and hence, they could be the major concerns. On the other hand, the False Negative samples are having weaker impact, because it can be solved by further labour analysis, even it may still decrease the efficiency of the work. Considering the impact from both measurements, we used the F1 score during the whole training procedure to monitor the training status. From Figure 14, we found that the model_2_2 and model_2_3 have a better convergence on F1 score on both training set and validation set, while it also reaches a higher value to 80% for validation set comparing with model_2_1.

The 3 models had been further applied to the testing dataset precisions 56.7%, 78.9% and 75.2%, while the recall of those are 33.3%, 67% and 60% (Figure 15).



(a) Model_2_1

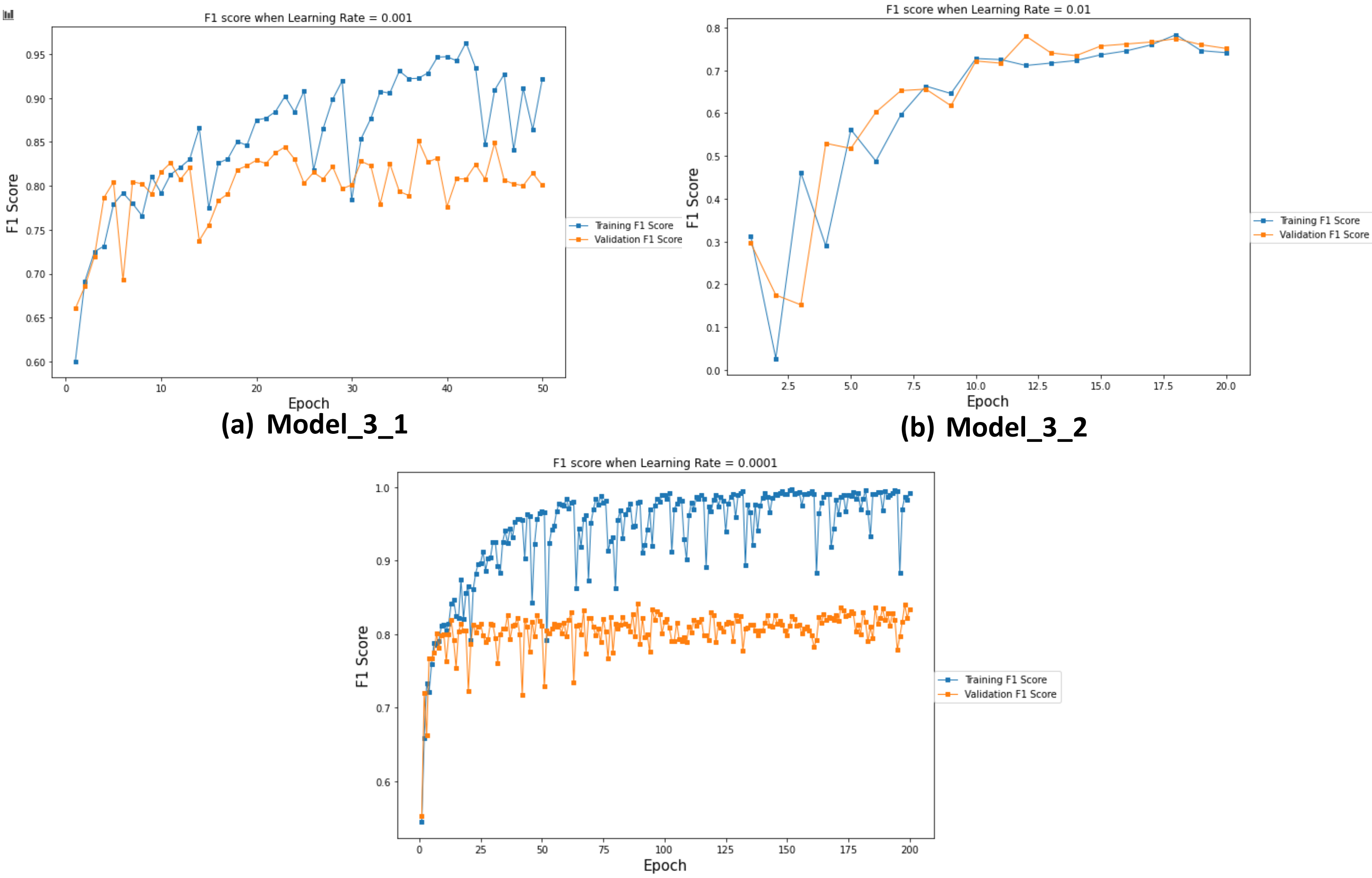
(b) Model_2_2

(c) Model_2_3

Figure 15 – Confusion Matrix of Prediction Result

Considering both the precision and recall, we can conclude that the model_2_2 and model_2_3 have better precision. However, there is an outstanding issue on the performance of 3 versions of model 2, as all of them have a much lower recall comparing with their precision.

3.3.3 Training – Model 3



(a) Model_3_1

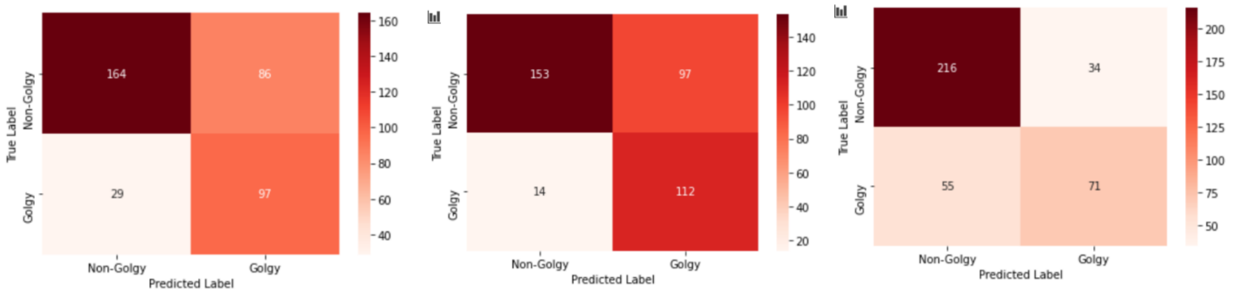
(b) Model_3_2

(c) Model_3_3

Figure 16 – Curve of F1 Score

We are still using the same loss function during the training procedure of model 3. Similar to model 2, the learning rate as 0.01, 0.001 and 0.0001 have been applied with different epoch as 20, 50 and 200. The generated versions of model 3 are labelled as model_3_1, model_3_2, and model_3_3. The figure 16 illustrates how the F1 score chanages in training set and validation set during each training procedure. And we can find the curve are better converged than the previous one, while the 3 curves are all converging to approximately 80% on the validation set.

The trained models were further applied to the prediction on the testing set. The result is as shown in Figure 17 in confusion matrix. We observed that the 3rd version of model 3 has an intuitively better performance with least false positive. Although the recall for the 1st and 2nd version is higher, their precision is still unacceptable.

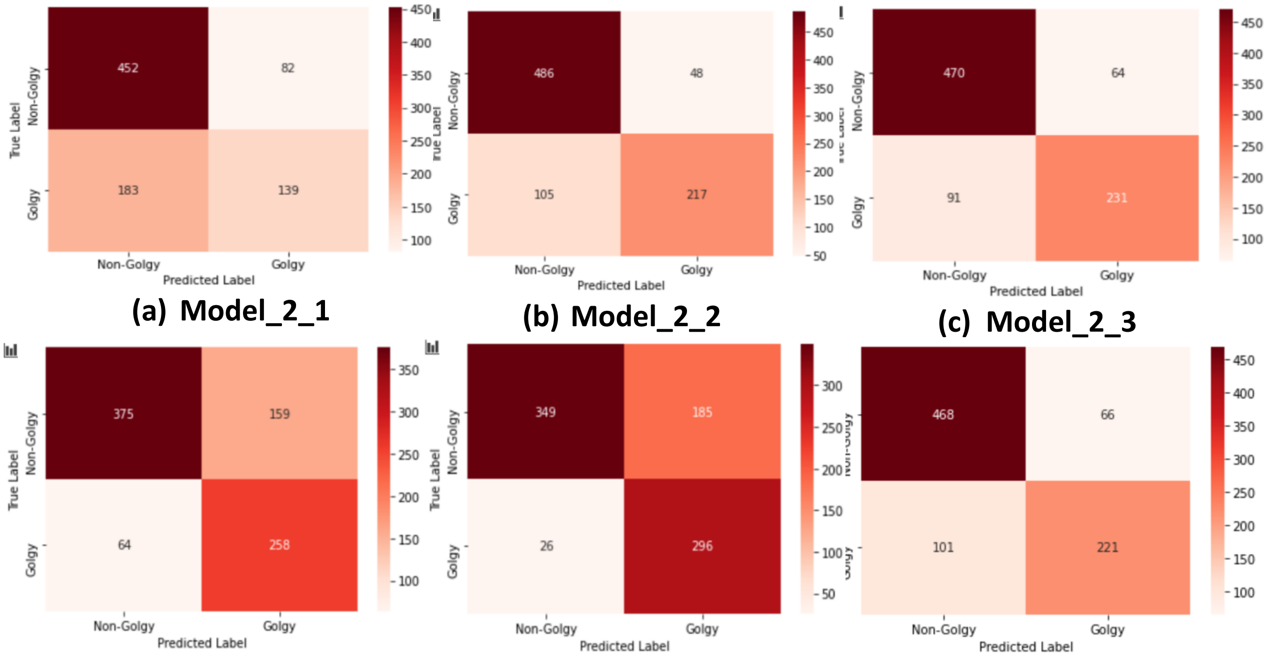


(a) Model_3_1 (b) Model_3_2 (c) Model_3_3

Figure 17 – Prediction Result of Model 3

3.4 Model Selection

From previous performance of the trained models, we noticed that some of them have good performance in both validation and training set, but lower performance in test set, which could be due to the sampling bias. Therefore, to better compare the performance of different models, we have combined the test set and validation set together as our final test set to evaluate the model performance. In the final test set, there are totally 534 non-Golgi objects and 322 Golgi objects. Only the generated



(a) Model_2_1 (b) Model_2_2 (c) Model_2_3

(d) Model_3_1 (e) Model_3_2 (f) Model_3_3

Figure 18 – Prediction Result on Final Test Set

model of model 2 and model 3 will be considered as candidate models, while the performance of model 1 was too poor to be accepted.

The result is shown in Figure 18. We can conclude that overall, the 6 candidate models, the model_2_2, model_2_3 and model_3_3 can be considered having better performance overall, while the number of predicted false positive samples are at the similar level which is lower than 70. Meanwhile, these 3 models also have acceptable predicted true positive samples comparing with the others.

We have further applied an ensemble method for prediction, which is to ensemble the 3 best models together in the prediction task by hard voting (majority voting). The members of the ensemble method include the model_2_2, the model_2_3 and the model 3_3. After applying to the final test set, the result of the ensemble method is as Figure 19 shows.

To compare the performance of all the candidate models, we have calculated the precision, recall and F1 score for positive object classification of every candidate model. The result is show in the table below.

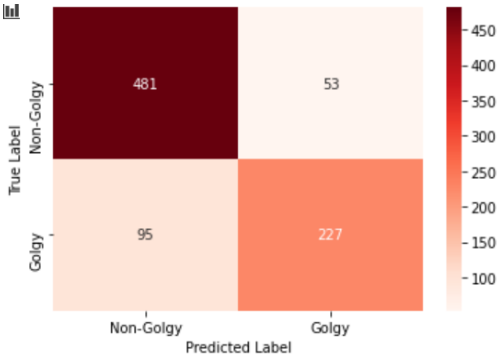


Figure 19 – Prediction Result of Ensemble Method

Model	True Positive	False Positive	False Negative	Precision	Recall	F1 Score
model_2_1	139	82	183	0.628959276	0.431677019	0.511970534
model_2_2	217	48	105	0.818867925	0.673913043	0.739352641
model_2_3	231	64	91	0.783050847	0.717391304	0.748784441
model_3_1	258	159	64	0.618705036	0.801242236	0.698240866
model_3_2	296	185	26	0.615384615	0.919254658	0.737235367
model_3_3	221	66	101	0.770034843	0.686335404	0.725779967
Ensembled Method	227	53	95	0.810714286	0.704968944	0.754152824

As highlighted in the table, the model_2_2, model_2_3, model_3_3 and ensembled method have the overall higher precision and F1 score. And the ensembled method has the highest F1 score and the 2nd highest precision, and its recall is higher than model_2_2 which has the highest precision. So, considering the precision and recall together, the ensembled method will be regarded as our final selection of the classification model.

4. Final Implementation

After the generation of the classification model, there is still a gab to the real implementation in the application in the lab. So, an implementation scheme with the helper program has been developed.

The whole implementation can be divided into 3 steps: 1) Object extraction: the program will scan the image and extract all the potential objects in the input image 2) Classification: classify all the objects to keep all the predicted positive objects as Golgi object, and generate the coordinates to a CSV file. 3) ROI generation: the CSV file will compose a rectangle to select the ROI of positive object in ImageJ tool. The steps 1 and 2 are implemented in python, and the step 3 is implemented in Macro and embedded in ImageJ.

Since the model is not able to ensure the 100% correctness on prediction result, there are still 2 steps of manual process need to be performed: 1) Among the predicted ROI, there may be some false positive samples, which need to be manually excluded. 2) Among the rest of the unselected objects in the image, there may also be some false negative samples, which need to be manually analysed and selected further. The below diagram illustrates the process as stated.

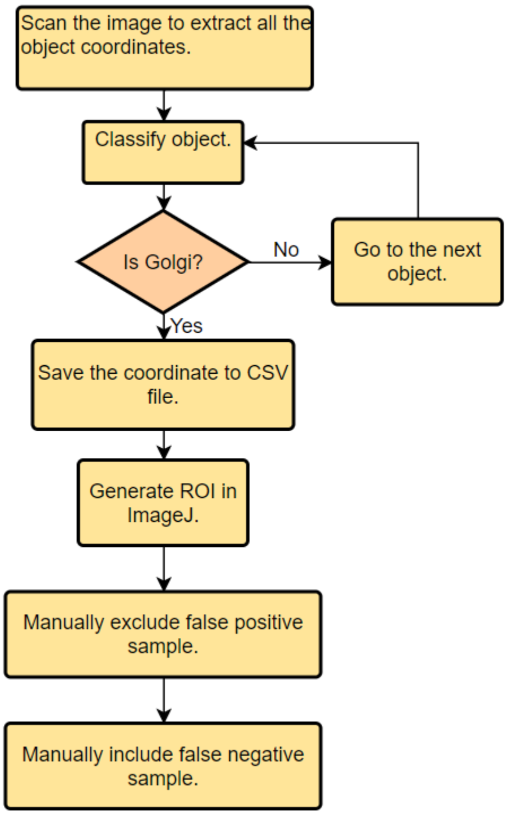

Scan the image to extract all the object coordinates.
Classify object.
Is Golgi?
No
Go to the next object.
Yes
Save the coordinate to CSV file.
Generate ROI in ImageJ.
Manually exclude false positive sample.
Manually include false negative sample.

Figure 20 – Implementation Workflow

The Figure 20 illustrates the diagram of the whole process.

We further applied the auto-selection program to our testing images. The result shows that the performance of the model is useable but sometimes unstable, while on some images it can reach the precision as 88% and recall as 70%, it only has 60% and 50% in precision and recall on some other images.

5. Conclusion

In summary, a few conclusions were made. Firstly, the learning rate 0.01 is always considered as a too high rate to make the model better converged in our case. Secondly, the ensembled methods works better than the rest of the other models. Thirdly, as the result shows, our solution can assists biologist in shortening the time on the Golgi selection work in GLIM method by auto-proposing a group of ROIs to increase the efficiency, even it is still not working perfectly on classifying the Golgi object.

6. Limitation & Future Work

There are also a few limitations on the current solution. And based on that, some possible improvement can be made further accordingly.

Firstly, from the precision and the recall of the final selected method, this solution still has risk in the real application, because it will still mix some false positive sample in the final selection, as well as miss some true Golgi objects. This can be possibly further improved by the following measurement:

1) Improving the model architecture may help to improve the prediction accuracy. In our solution, we have only applied a traditional and classical deep learning model, which is the composition of CNN part and DNN part. However, this concept and architecture is out-dated, considering the advent of new techniques, such as network like VGG, Res Net, R-CNN, YOLO, and GAN, etc. Those different model architecture may have different performance on this problem.
2) Improving the parameter tuning and training procedure may also help to improve the prediction accuracy. Other than the parameters tuning work that has been done in our process, adjusting kernel size, stride, applying weight decay, etc, can also be further implemented to find the better solution of the model.
3) We may also use the machine learning method such as SVM, Random Forest, XGBoost, etc, in the region proposal workflow, while deep learning is less interpretable and more unstable than most of the machine learning method even it is widely used in the object recognition, which may lead to that the training and modelling harder to be traced and adjusted according to the real situation.
4) The most traditional programming may also be applied to increase the prediction accuracy. By using the prior knowledge on how the Golgi mini stack looks like, the programming can be implemented by scratching the logic into a program to identify the Golgi object.

Secondly, on our final implementation plan, there are still a few aspects can be improved:

1) Firstly, the object recognition module is still only a standalone program needed to be executed by entering the command in the commend line window. This can be further improved by adding an user interface, or directly translated the program to be embedded in ImageJ tool.
2) Secondly, the objects are generated by a group of coordinates file in CSV format. This step can also be embeded into a universal program.

Reference:

[1] Hieng Chiong Tie, Divyanshu Mahajan, Bing Chen, Li Cheng, Antonius M. J. VanDongen and Lei Lu. 2016. A novel imaging method for quantitative Golgi localization reveals differential intra-Golgi trafficking of secretory cargos. Molecular Biology of the Cell, Vol. 27, No. 5

[2] Hieng Chiong Tie, Alexander Ludwig, Sara Sandin, Lei Lu. 2018. The spatial separation of processing and transport functions to the interior and periphery of the Golgi stack. eLife 2018;7:e41301