DATS 6203 Final Report

Aashish Nair

Human Protein Atlas Image Classification

Abstract

Models that can classify microscopic images with high accuracy have the potential to revolutionize the practice of medicine. The aim of this project is to take images with 28 classes and multiple labels and build a model that can identify these images with a high f1-score. Ultimately, after utilizing transfer learning, image preprocessing, and hyperparameter tuning, the final model created is a VGG16 model with a validation loss of 0.1724, an f1-score of 0.1421 on the validation set, and an f1-score of 0.1103 on the testing set.

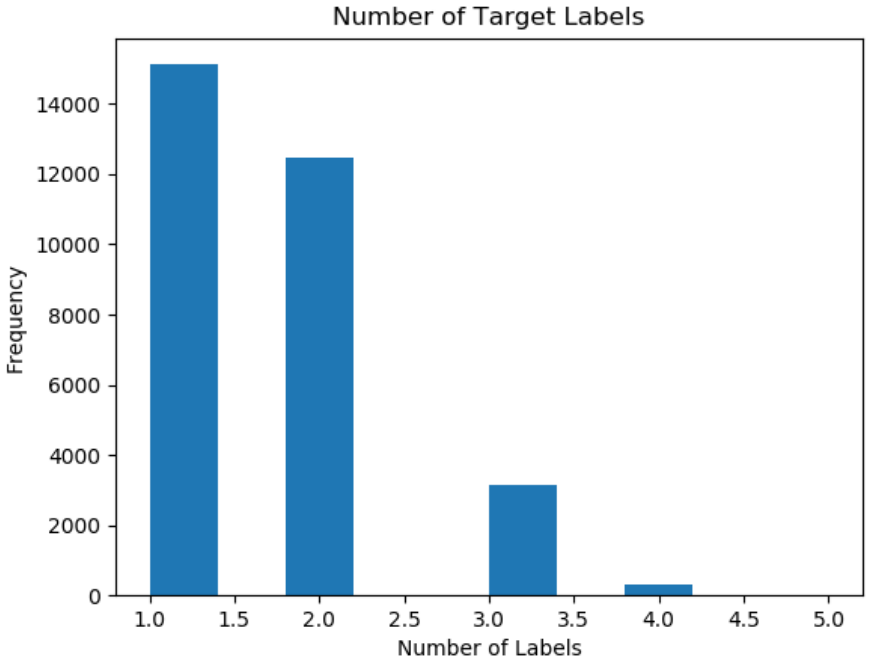
Introduction

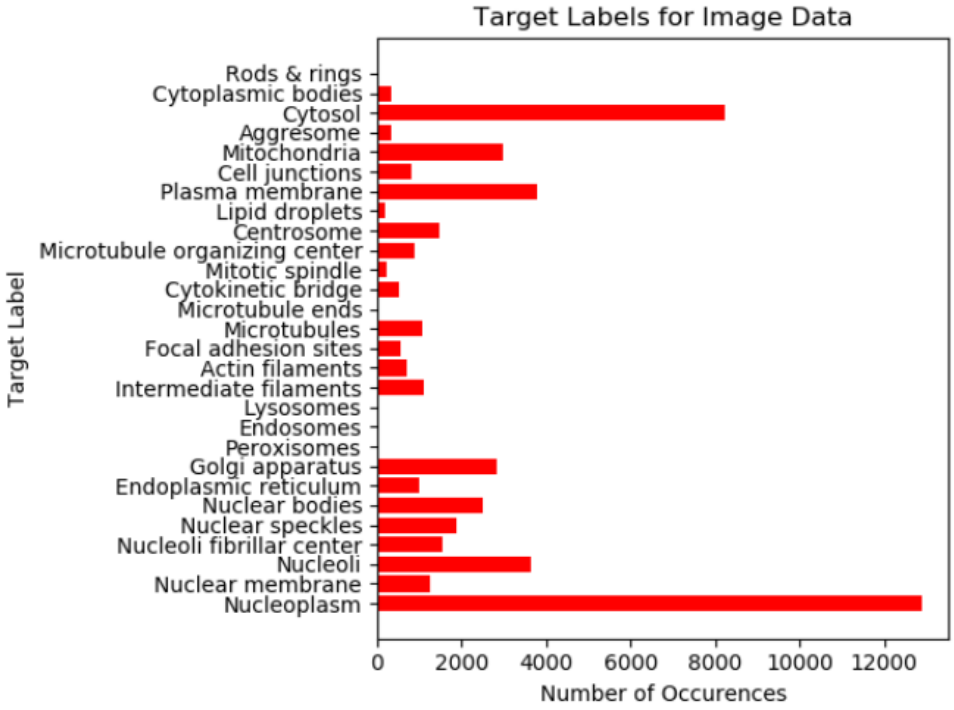
Effectively classifying protein patterns in human cells has a plethora of applications in the medicine industry. Creating a reliable model that can correctly identify these images can enable medical professional to carry out faster and more accurate diagnostics and develop new medicine and medical treatments. In this project, convolutional neural networks will be trained and evaluated with the goal of creating a model that can detect proteins in images with a high accuracy.

Description of the data set

The dataset consists of over 30,000 samples of protein patterns. Each sample in this dataset is represented by 4 images, with each image having the colors of blue, green, red, and yellow. The dataset contains a total of 28 different labels. It should be noted that these images are multilabel, meaning that each image may contain more than 1 label. While most images have only 1 label, there are a considerable number of images that have 2 and 3 labels. There are even images, albeit a few, that have 4 labels.

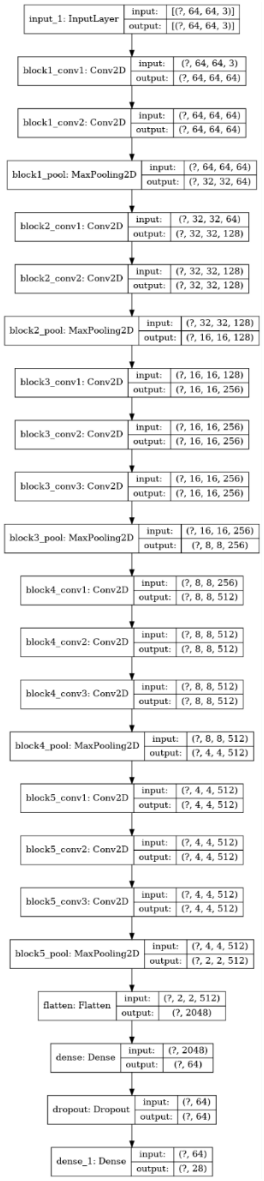
Furthermore, the distribution of the labels in this dataset are not evenly balanced. In fact, images with the labels “Nucleoplasm” and “Cytosol” comprise of over 80% of the images provided. This massive imbalance of data must be accounted for when developing the deep neural network. In terms of dimensions, the images are 512 x 512, though they will be resized prior to training the model. Since any size equal to or above 128 x 128 pixels registers an error from memory shortage, the images will be resized to 64 x 64 pixels.


Number of Target Labels
Frequency
Number of Labels
0
2000
4000
6000
8000
10000
12000
14000
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Target Labels for Image Data
Target Label
Number of Occurences
Rods & rings
Cytoplasmic bodies
Cytosol
Aggresome
Mitochondria
Cell junctions
Plasma membrane
Lipid droplets
Centrosome
Microtubule organizing center
Mitotic spindle
Cytokinetic bridge
Microtubule ends
Microtubules
Focal adhesion sites
Actin filaments
Intermediate filaments
Lysosomes
Endosomes
Peroxisomes
Golgi apparatus
Endoplasmic reticulum
Nuclear bodies
Nuclear speckles
Nucleoli fibrillar center
Nucleoli
Nuclear membrane
Nucleoplasm
0
2000
4000
6000
8000
10000
12000

Description of the deep learning network and training algorithm:

The neural network used in this project will a convolutional neural network (CNN) since they excel in applications like image classification. Instead of building a CNN from scratch, it would be beneficial to implement transfer learning and take advantage of a tried and trusted pre-trained model. The chosen network for this project is the VGG16 model, which contains numerous convolutional layers and pooling layers designed to tackle problems just like this one. Once the vgg16 model is modified with the Keras framework to suit the input from the protein images, it will be trained to distinguish between protein patterns effectively and accurately. The layout of the altered vgg16 model can be seen below:



Experimental setup:

Firstly, the data will be split into training and test sets, with the test set being used to evaluate the trained network. The training set will once again be split into a training set and validation set with the "ImageDataGenerator" function. The training set, and validation set, and test set make up 64%, 16%, and 20% of the available data, respectively. Originally, it was intended to for K cross validation (or stratified K cross validation) to be utilized to split the training data into smaller subsets to ensure better results. However, this method proved to be severely time consuming and provided minimal marginal benefits. Therefore, this approach was soon abandoned.

The performance of the model will be judged based on the loss function and the f1-score. The loss function and f1-score metric chosen are the binary cross-entropy and the macro f1-score as they are most suited for a multilabel classification problem such as this one.

The key training parameters under scrutiny during the development of the neural network are the learning rate, batch size, optimizer, dropout value, and activation function. The ideal value for these parameters were determined by experimenting with different combinations of values and seeing which one yielded the best results. The table below shows the arguments for each of these parameters that were tested.

Parameter	**Set of Values**
Batch Size	16, 32, 64, 128
Learning rate	0.01, 0.005, 0.001, 0.0005, 0.0001
Optimizer	Adam, Adagrad, Adamax
Activation function (hidden layers)	relu, sigmoid, tanh
Dropout	0.1, 0.2, 0.25, 0.3

As previously stated, the given data is incredibly imbalanced, which leads to the model being prone to overfitting. For that reason, the performance of the model can not be evaluated by its accuracy as such a metric would be deceptive and could lead to false conclusions. Thus, the macro f1-score will be utilized as the key metric for gaging how successful the model is.

Moreover, to address the imbalanced dataset and further enhance the training of the model, image augmentation will be implemented into the training, validation, and test sets. With the "ImageDataGenerator" function, the training, validation, and test sets are normalized to fall within the range of 0 and 1 to support the models performance. Furthermore, the images in the training set are increased by altering the existing training set images through rotation, zooming, flipping, and horizontal and vertical shifting.

After testing the model with various parameters, the chosen parameter values are shown in the table below:

VGG16 (Keras framework)	
Parameter	**Assigned Values**
Batch Size	64
Learning Rate	1e-4 with 5e-5 decay
Activation function (hidden layers)	relu
Dropout	0.2
Optimizer	Adam

Ultimately, tuning most of the parameters only yielded marginal benefits. Originally, the validation loss of the model exceeded 0.4. Changing the learning rate to 1e-4 made the most significant improvement, with the validation loss reducing to 0.19. After tweaking the values of other parameters helped reduce this validation loss to 0.17. It should be noted that these parameters are not the optimal values for this model. Given the time constraints, performing a thorough hyperparameter tuning and training the model with every possible combination is simply infeasible.

The design of the VGG16 model was carried out with the Keras framework. The "ImageDataGenerator" function was utilized to normalize the images, split the training set into training and validation sets (80-20 ratio), and increase the amount of data by modifying the original images. Such changes included tweaks with rotation, translation, flipping, brightness, and zoom.

The original VGG16 model was modified by altering the input and output layers as well as adding a few more. As VGG16 only accepts inputs with 3 channels, the inputted data had to be altered to reflect that. The chosen approach for doing so was to input only a portion of the given images. The images now had a shape of (64, 64, 3). In addition, the final layer was altered to ensure that the output data of the function would have 28 values between 0 and 1, as it suits the nature of the multilabel image classification problem. The model was trained several times with experimental adjustments made between each attempt.

Results:

Model Description	Validation Set		Testing Set
	BCE Loss	Macro F1-Score	Macro F1-Score
VGG16 model	0.1724	0.1421	0.1103

The results of the transfer learning, image augmentation, and hyperparameter tuning is displayed in the table above. As can be seen from the macro f1-score of the model, the efforts made to address the data imbalance as well as the prone to overfitting were insufficient. Most of the encoded predictions of the model registered a "1" for the first target label, which is the most popular one. Furthermore, when analyzing the actual 28 values outputted by the model for each prediction, it was noted that many of the values for each category were below 0.5. This means that even though the predicted value of a certain target label was higher than all the others, it would be rounded down to zero, leading to an inaccurate prediction. When encoding the predictions, the idea of changing the threshold for distinguishing between a "0" and "1" was considered. However, this deviated from the methodologies used when developing models, so the idea was rejected.

Conclusion

In summation, the challenge of classifying mixed patterns of proteins in human living cells proved to be a considerable challenge. The sheer number of classes to distinguish from and the number of labels in each image meant that the model needed a significant amount of complexity to yield a high macro f1-score.

For future reference, it is worth noting that there are other potential methods that can be applied to improve the performance of the model to an even greater degree. Firstly, if possible, the images could be inputted into the model at a higher resolution. The Kaggle competition provided these sample images at size 512 x 512. However, given the limited memory capacity, the images had to be resized to 64 x 64. In addition, stratified K-cross validation could also be implemented instead of the typical train/test split. Cross-validation was originally considered during this experiment, but it was decided that training a model this way would require too much time and computing power with little marginal benefits. In addition, the thresholds for F1 score could be tweaked for certain classes. This could yield better results compared to using the default threshold. Finally, it would be worthwhile to explore more complex model architectures in search of models that can register even higher f1-scores.

References

https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

https://machinelearningmastery.com/multi-label-classification-with-deep-learning/

https://www.kaggle.com/guglielmocamporese/macro-f1-score-keras