DATS6203 – Machine Learning II, Fall 2020
Final Project Report
Group 2: Tran Hieu Le, Voratham Tiabrat, Aashish Nair

Human Protein Atlas Image Classification

Abstract

Identifying the pattern and organelle localization of proteins would provide more insights about human living cells and accelerate the diagnostic of diseases. This project presents the utilization of Convolutional Neural Network (CNN) to classify mixed patterns of proteins in order to provide full understanding of the complexity of human living cells.

Introduction

Proteins are complex molecules made of thousands of amino acids, being responsible for many important functions in human body such as execution and regulation of issues and organs. Understanding the complexity of cell structure plays a key role in investigating new medicine and improving medical treatment. Therefore, the classification of proteins has become a field of interest for many scientists and biomedical researchers.

In the past, there were many studies on the classification of proteins, but they are restricted to single patterns in a few types of cells. With the development of technology nowadays, the mixed patterns of proteins can be observed from microscope images, which provides the opportunity to progress the research of human proteins. However, the complexity and highly variety of morphology in human living cells make it difficult to identify the structure and the number of protein patterns in organelles. Moreover, the rare appearances of small protein structures such as endosomes, lysosomes, microtubule ends, rods and rings limit the classification capability of these classes.

This project makes use CNN architectures and transfer learning to classify the mixed patterns of proteins. The imbalance of data is handled by different techniques such as resampling, data augmentation and focal loss.

Dataset

The dataset for the project comes from the Kaggle competition which is originally provided by the Human Protein Atlas – a Sweden-based program researching proteins in cells, tissues and organs:

https://www.kaggle.com/c/human-protein-atlas-image-classification/data

The "train.csv" data contains 31072 samples of 27 different living cells. Each image is represented by four channels: the protein of interest which is the main filter, and three cellular landmarks as references: nucleus, microtubules and endoplasmic reticulum. The target is the protein pattern including 28 categories labeled from 0 to 27. Each image may have one or more labels.

Platform

Due to the complexity of the dataset and the requirement of image augmentation, the project needs to be executed in a virtual machine with powerful GPU. The main frameworks are Keras, Tensorflow and Pytorch which are popular in deep learning. The primary programming language is Python.

Methodology

The original dataset is split into training, validation and testing sets. Multi-label stratification is employed to generate training, validation and testing sets. The training set is used to train the model, whereas the validation set is used for fine-tuning hyperparameters. The testing data is used for final evaluation.

Since the training data contains 20,000 – 100,000 images after oversampling, minibatches training is required. The optimal batch size is 256 or 512 depending on the training data size. The optimal optimizer is Adam. After tracking the convergence of loss function with different learning rates, a learning rate in the range of 5e-3 to 3e-4 is recommended. More experiments show that learning rate 1e-3 is the most stable and optimal for all models in the project. Therefore, 1e-3 is selected as the base learning rate.

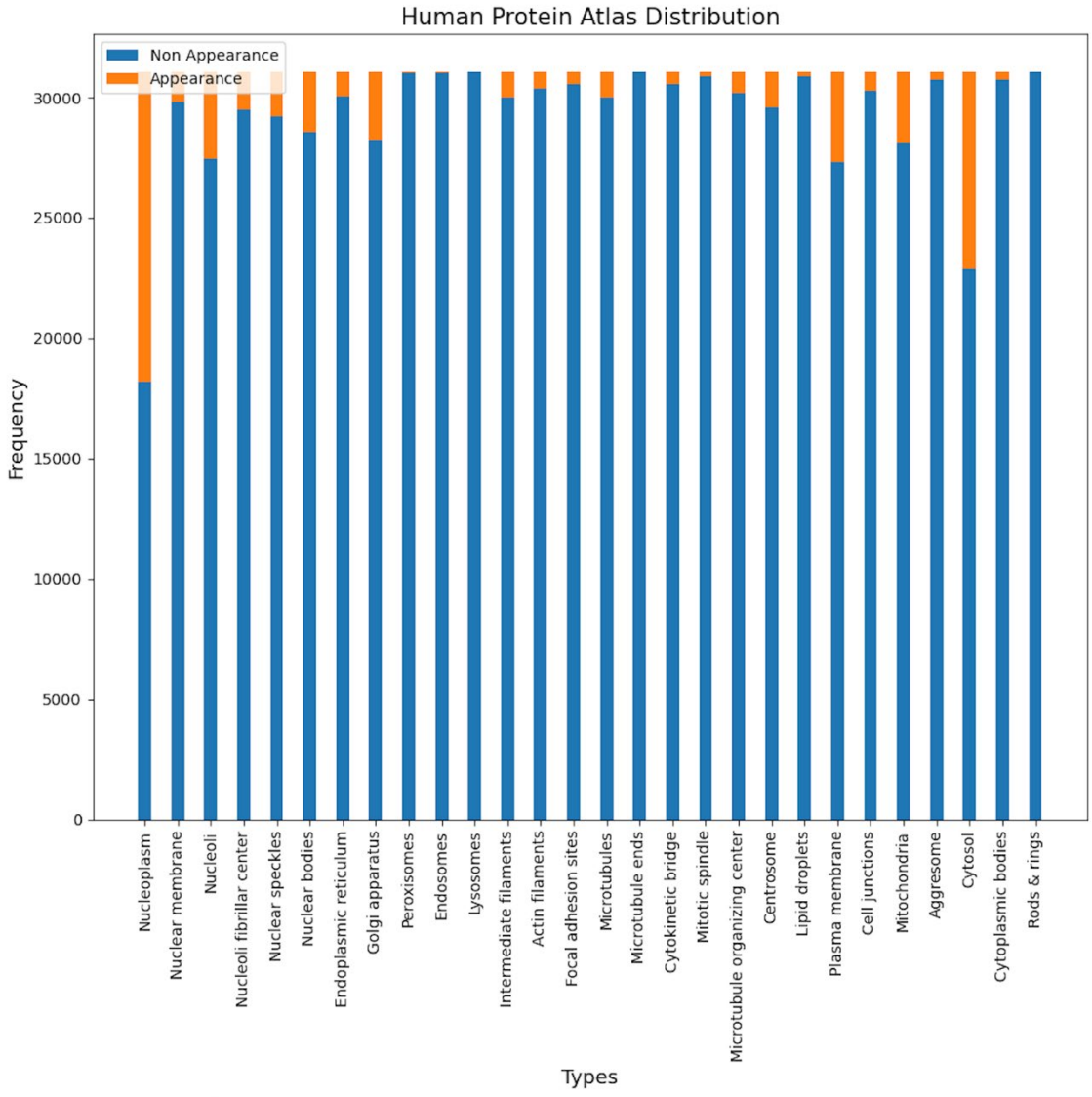
Pretrained models such as VGG, ResNet and DenseNet are used as backbone architecture to implement transfer learning as they are effective in image recognition problems. Since the dataset is quite different from the ImageNet, the pretrained models require a substantial entire retraining.

To handle the imbalance of dataset, popular methods such as resampling, data augmentation, cost sensitive and focal loss are experimented. Another issue in the dataset is the 4 channels for each image. To address this, the input channel of the first convolution layer of the pretrained model is modified to 4. Another technique to manage model complexity called 1x1 convolution is also applied.

Due to the imbalance of the target variable, accuracy is not a good measurement for model performance. The macro F1 score will be the primary score since the competition also uses this metric for evaluation. Sample F1 score is used as reference for the prediction of multi labels in a particular image.

Results

The following figure is the visualization of the class distribution in the original data. It is easy to notice the high imbalance level of the labels. Rare classes such as microtube ends, rods & rings appears only 10-15 times in a total of 31,072 cells.



We will train baseline models from scratch and apply different ideas, techniques and tricks to address the imbalance problem.

Keras framework:

Aashish Nair's part:

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. 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 image classification problem. The model was trained several times with experimental adjustments made between each. The following table displays the performance metrics of the best VGG16 model.

Model	**Validation**		**Testing**
	BCE Loss	Macro F1 score	Macro F1 score
VGG16	0.1724	0.1421	0.1103

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. The majority 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 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 of the other models, so the idea was rejected.

Voratham Tiabrat's part – Keras and Tensorflow framework:

Model description	Validation set		Testing set	
	Macro F1	Sample F1	Macro F1	Sample F1
CNN: Baseline CNN (4 Conv2d layers, Linear layer)	0.1979	0.4040	0.1582	0.3122
Resnet50: Baseline model	0.2866	0.4971	0.2783	0.4924
Resnet50_2: Oversampling Augmentation	0.5305	0.5795	0.4894	0.5774
Resnet152: Oversampling Augmentation	0.5700	0.6020	0.5133	0.5982

According to the table, the baseline model are basic CNN and Resnet50 model have very poor F1-score. The first CNN model is a non-pretrained model which the performance is obviously low, so we try to implement the pre-trained model instead. Even Resnet50 performance is better than the basic CNN but the F1-score still consider quite low. The reason is that there are some classes that appear only around 10 times in dataset (minority class) and the model could not be able to predict it. There are 7 classes considered the minority class which consist of Peroxisomes, Endosomes, Lysosomes, Microtubule ends, Mitotic spindle, Lipid droplets and Rods & rings. Therefore, oversampling by data augmentation of the minority classes is applied to improve the model. After using the generated data and tuning the hyperparameters, the macro F1-score of Resnet50 model is significantly improve to 0.5 for validation set, but still less than 0.5 for testing set.

Then we move on to another pre-trained, Resnet152, to see whether it would perform better performance than the Resnet50 or not. As a result, the Resnet152 performance is slightly better than the Resnet50 and also F1-score for both validation and testing set is above 0.5. Even data augment technique would be help to improve the model, but it is important to note that the dataset is multi-labels so when we generate more data for minority class, it is likely that we also increase the number of majority class too.

Tran Hieu Le's part – Pytorch framework:

The baseline model is ResNet34 trained from scratch. After that, we implement different techniques and ideas to improve the classification. The following table shows the models with description and their performances in the 1st stage of the project.

Model description	Validation set			Testing set	
	BCE Loss	Macro F1	Sample F1	Macro F1	Sample F1
Resnet34_baseline: Baseline model Resnet34(4)	0.135016	0.166794	0.422629	0.153524	0.410948
Resnet34_1: Resnet34(4) Focal loss	0.172575	0.263734	0.470276	0.257581	0.465790
Resnet34_2: Resnet34(4) Focal loss Oversampling Augmentation Differential LR	0.114301	0.612564	0.637246	0.601216	0.631511
Resnet34_3: 1x1 Convolution Resnet34(3) Focal loss Oversampling Differential LR Augmentation	0.118927	0.615857	0.637563	0.601246	0.631303
Densenet121: 1x1 Conv layer DenseNet121(3) Focal loss Oversampling Augmentation Differential LR	0.110288	0.628404	0.672417	0.609551	0.659747

According to the table, the baseline model has very poor performance. The model is not able to predict the minor classes, leading to low macro F1 score. Applying focal loss to the training greatly improved the score from 0.15 to 0.26. This indicates that focal loss is a better selection for loss function in the training process.

However, it is worthwhile to note that the focal loss in validation process shows strange patterns which would be the cause of the model complexity and the rarity of minor classes in the validation set (the rarest classes such as microtube ends, rods and rings have only 1 or 2 samples in the validation data). Moreover, due to the implement of oversampling on minorities, the distribution of classes in the training set is different from the distribution in the validation set. As a result, the focal loss functions for training and validation sets should be different in terms of the parameters alpha and gamma (in this project, alpha = 0.25 and gamma = 2 for both datasets). In other words, alpha and gamma are hyperparameters to be fine-tuned based on the dataset. Therefore, BCE loss is selected as the validation loss to ensure the stability of the validation process.

Since focal loss shows great improvement in the classification, it is continued to be used with the combination of image augmentation and oversampling in the training process. The macro F1 score further improves from 0.26 to 0.6. The application of 1x1 convolution technique to reduce the depth of input data from 4 channels to 3 channels show a little increase in the score. The deployment of Densenet121 model as the backbone architecture results in a better score than Resnet34.

In the 1st stage, all the models are trained entirely from the beginning, which may lead to the corruption of the pretrained weights due to the randomness of initial weights of the top layers ("fc" layer in Resnet and "classifier" layer in Densenet) and 1x1 convolution layer. Therefore, the weights of top layers require a pretraining before unfreezing the entire pretrained model to preserve the power of transfer learning.

In the 2nd stage, the top layer is pretrained using an augmented data while the rest of the model are frozen. After that, the model is trained entirely using the same learning rate to ensure all the layers can learn generic information. In both processes, focal loss is used to fine-tune the weights since the purpose is to let the model generalize features and characteristics of images rather than classify unseen data. Finally, differential learning rate is applied to train the entire model again so that the model can learn specific and complex features which are important to the classification of unseen data. In this final training, the validation loss function is BCE.

Model description	Validation			Testing	
	Loss	Macro F1	Sample F1	Macro F1	Sample F1
Resnet34_2, Unfreeze top layers LR = 1e-3	Focal loss: 0.021771	0.093188	0.345450	0.076483	0.343027
Resnet34_2, Unfreeze all layers Lr = 1e-3	Focal loss: 0.015171	0.416472	0.605719	0.404191	0.600911
Resnet34_2, Unfreeze all layers Differential LR	BCE loss: 0.110295	0.635510	0.660989	0.639354	0.658912
Resnet34_2's architecture and description are taken from the 1st stage.					

According to the results, the model Resnet34 has a huge increase in the macro F1 score from 0.6 to 0.64 when the weights of the top layer are pretrained before unfreezing all layers. This trick would be applied to Densenet121 as well. However, I haven't finished this for Densenet121 since it requires a lot of time to train the model and find optimal differential learning rates.

Future Improvements

There are many ideas and methods to extend the accuracy of classification and macro F1 score:

- The most effective way is to use external dataset to gather more samples containing minor classes. This provides more information for the training data and increases the appearances of rare labels in validation data.
- Fine-tuning threshold for macro F1 score: In this project, the default threshold for macro F1 score is 0.5 which is not an optimal selection for strong imbalanced data. A fine-tuning process for thresholds of 28 classes would help to improve the classification.
- Modification of focal loss function: Due to the strong imbalance of labels in the dataset, the focal loss function should have a specific weight for each class. Furthermore, alpha and gamma should be fine-tuned to obtain the optimal values.
- Bigger resolution of images: Images with larger resolution would provide more detailed information which is useful for the model to learn and make classification.

- K-fold stratification: There is no guarantee that the validation data is representative due to the random splitting of original data. K-fold stratification is a good solution to this issue. However, this method requires external data to make sure that the number of samples in rare classes are sufficient to be divided into multiple folds while retaining the distribution of the complete training data.
- Experimenting different CNN architectures.
- Ensemble models.

Conclusion

This project features the classification of mixed patterns of proteins in human living cells using CNN architectures. The primary challenge is the strong class imbalance which is a critical issue in classification. To address this difficulty, oversampling, data augmentation and focal loss (a method that demonstrates amazing results with highly imbalanced datasets) are employed during the training, which results into a much better improvement comparing to the baseline model. To improve the model performance, differential learning rate is applied: small learning rate for the bottom layers to let the model learn generic features while not disturbing the original pretrained parameters, slightly higher learning rate for middle layers to fine-tune the weights to capture complex and specific information, and the optimal learning rate for top layers to determine the prediction.

References

https://arxiv.org/pdf/1708.02002.pdf

https://towardsdatascience.com/transfer-learning-using-differential-learning-rates-638455797f00

https://www.nature.com/articles/s41592-019-0658-6

https://leimao.github.io/blog/Focal-Loss-Explained/

https://www.kaggle.com/allunia/protein-atlas-exploration-and-baseline