ECBME 4060 Intro-Genomic Info Sci & Tech

Human Protein Atlas Image Classification

Name: Shilong Yin[1], Diwei Xiong[2]

[1]Department of Biomedical Engineering, UNI: sy2792.

[2]Department of Biomedical Engineering, UNI: dx2183

*To whom correspondence should be addressed.

Mentor: Prof. Wei-Yi Cheng

Code uploaded on https://github.com/Shilong1210/Human-Protein-Atlas-Image-Classification.

Abstract

Motivation: Providing the confocal microscopy image, we are able to see deep inside human cells and their organelles. In order to better understand the function of cell, the best way is to determine the distribution pattern of protein. Therefore, it is a hot topic in the research field to develop models that are able to classify the mixed patterns of proteins in cells, which is the main task of our project.
Results: Based on kernels from Kaggle, we are able to construct a protein atlas image classifier with data generator, training and validation splitter, neural network and threshold generator. Our model gives a prediction with the best score of 0.422.
Availability: CNN, Tensor-flow, Keras and other relevant packages
Contact: dx2183@columbia.edu, sy2792@columbia.edu
Supplementary information: Training and test data are from Kaggle competition: *Human Protein Atlas Image Classification*.

1 Introduction

Protein, which serves as a service provider for the cell to exert specific function, is ubiquitously distributed in a biological system. After the transcription of genomic information, protein starts to form as the product of RNA translation. As the executor of genomic information, protein enables cells to transmit and receive information from outside and behave accordingly; then, it is the coordination of cells that maintains the behavior and operation of the biological system. In order to better understand how a cell work as well as the function related to its organelles, it is significant to classify the protein structure of a cell's organelles and find the occurrence of different organelles among cells. In the past, the detection and classification of protein is far from ideal in that people can only classify single pattern in one or few cells. To fully understand the complexity of cell structure and function, a way to categorize mixed patterns for different range of human cells is in need. Currently, due to the advance development of biological imaging, we are able to produce images across cells with different channels of features using high through-put confocal microscopy. Therefore, in order to facilitate medical research in molecular biology, we need to develop complete and accurate models to identify and classify organelles in cells from a large number of images produced by microscopy. For our part, we decided to use convolution neural network as the core to build our model.

2 Methods

In order to get a relatively accurate classification, we applied CNN (convolution neural network) to get better prediction.
Generally, CNN contains input layer, hidden layer in middle and output layer, just like the Fig 2.1 shown below.

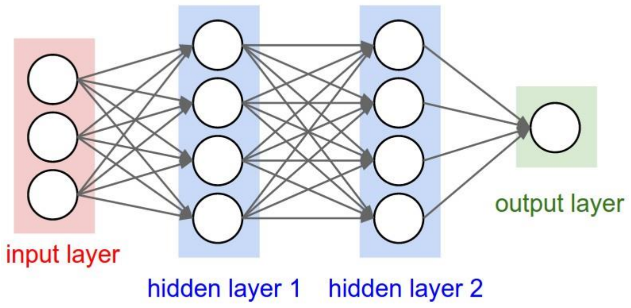

Fig 2.1 Basic structure of CNN

We have training data, the confocal microscopy coming from 4 channels (R: microtubule, G: protein, B: nucleus, Y: endoplastic reticulum) as our input; then, the hidden

layer is the neural network which we will discuss afterwards. The output should be a 28-element vector, with each component reflecting the probability of existence of a certain protein class.
Our input are images, from which we cannot easily extract features by common operations and update weights based on f1 score. Fortunately, there are deep learning libraries at hand, like Tensor-flow and Keras, which help us save the troubles. And I will list several methods we tried during our experiment.

2.1 Pretrained model

Pretrained model is a model that has been trained on some kind of data. Although it may not perform well on a specific task, it can be a good corner stone for a model that is aimed for a similar task. In our case, we want to classify the pattern of protein, which is a task that can be categorized into general image classification. Therefore, using a model pre-trained on millions of images as a base model will be a better choice rather than training a new model from scratches.[1]
However, in our case, there is one drawback for using pretrained model: The input shape is fixed on 3 channels (RGB). But our data includes four channels. Thus, it is a challenge for us to find a solution to merge information from 4 channels into 3.

2.2 Convolution network

Convolution network is used to extract features from 2-D images. The basic principle behind it is to apply filters on the matrix to achieve a certain image processing like blur or edge detection. This process is called hierarchical feature learning, which is similar to how brain identifies the objects. Thanks to the existing libraries, we are not required to produce filters by ourselves but only to decide the number and size of filters. There is a figure below displaying the function of convolution layer. The extracted features will later be used to calculate the weights for each neuron.[2]

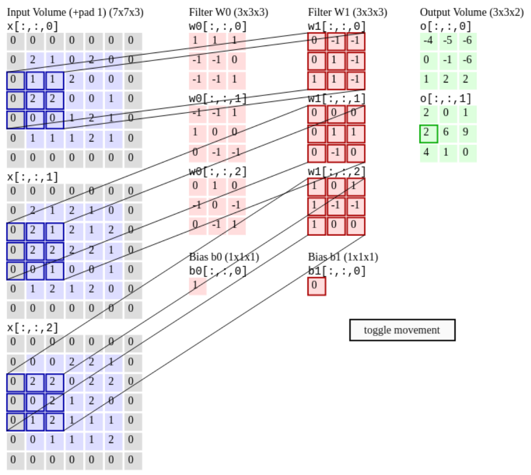

Fig 2.2 Function of convolution network

2.3 Batch normalization

Generally, what normalization do is to adjust and scale the input from a much larger range to [0,1]. The reason why we need this normalization layer is that by normalizing the inputs, the amount by what the hidden unit values shift around will reduce, which means less time for the model to reach the optimal parameter.
Batch normalization can also help the model to increase resistance for overfitting because it has a slight regularization effect.[3]

2.4 Dropout and dense

Dropout and dense layers play a similar role in the neural network, which is to decrease the size of parameters. But the principles behind are quite different.
As for dropout layer, it randomly set a fraction of input units to 0 in order to prevent overfitting. The reason why this works is that it helps the model to reduce the dependencies on a certain unit.[4]
Compared to dropout layer, what dense layer do is much simpler. It just connects several neurons on one layer to one neuron on the next layer to reduce dimensions. One thing is noticeable in this case is that the final layer before the output should includes an activation function called *sigmoid*, which is shown below. It is especially useful for models to predict the probability since its output range is [0,1].[5]

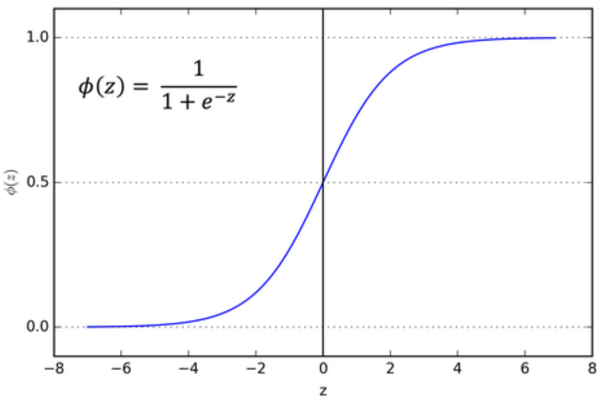

Fig 2.3 Sigmoid function

2.5 Training and validation

Generally, when we train our model with data, we will not use the whole dataset, because it may expose our model to the risk of overfitting. A wiser way would be splitting a fraction of data from original dataset, which is called validation data, and use the rest to train the model. Then we can use the validation dataset to evaluate the model skill. Since the model is independent of validation data, it can serve as an unbiased evaluation criterion for the model. This is typically called a train-test split approach to algorithm evaluation.[6]

Now, we have all the methods needed to build a model. The next step will be to train the model with input data sets and produce predictions, which will be shown in the next section.

3 Results

We have built two models based on kernels in our experiments, which are different in their structures. Now, let's analyze these two models and their results.

3.1 CNN with pretrained model (Best score: 0.345)

With the help of pretrained model of InceptionResNetV2, we no longer need to construct very complex network structures to extract features.[7] However, it is mentioned before that one limitation of using pretrained model is that it requires inputs only from 3 channels. To settle this conflict, the solution original kernel gives is to compress 4 channels into 3 by:

$$\text{Channel 1} = R/2 + Y/2$$
$$\text{Channel 2} = G/2 + Y/2$$
$$\text{Channel 3} = B$$

RGBY means data from 4 channels respectively. However, since green channel contains most information about the protein location, we decide to adjust the weights of each channel to:

$$\text{Channel 1} = R/2 + B/2$$
$$\text{Channel 2} = Y/2 + B/2$$
$$\text{Channel 3} = G$$

in the hope that the model can better predict the pattern of protein in the context of other organelles.

We also improve the learning efficiency of the model by adding a callback, a function that is able to adjust the learning rate of model. In the gradient descent, learning rate defines the step size of learning. It is a good way to adjust learning rate in this process to prevent the model from getting stuck at a local minimum.

What's more, in the original kernel, the learning curve is not very smooth and carries lots of small fluctuations, which is showed below:

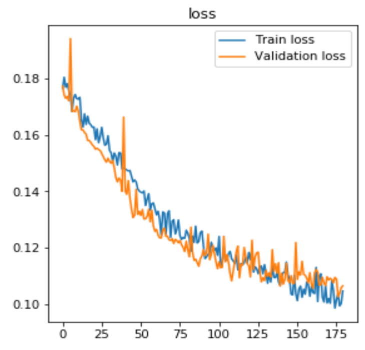

Fig 3.1 Learning curve of original kernel

It is shown that the parameters have experienced many updates, but the quality of gradient descent is not so well. Therefore, we decide to increase the batch size in order to improve the efficiency of learning. With the assistance of adaptive learning rate, the model can reach the optimum value very quickly. See the figure below.

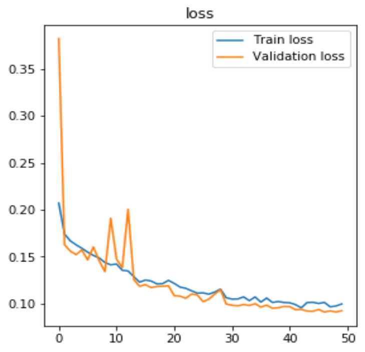

Fig 3.2 Learning curve of our model

As for prediction, we need a probability threshold to determine whether a certain protein class exists. In original kernel, the threshold is fixed to be 0.2 for all classes. However, the probability mass function for each class in training dataset is not equal, which may lead to the case that the minor class will never appear in our predictions. To fix this, we decide to calculate thresholds for each class using validation data with f1 score as evaluation criterion.

By doing these things, our model gets 0.345 on the leader board as the best score, which is an improvement about 28% compared to 0.269 from the original kernel.

3.2 CNN with GAP layer (Best score: 0.422)

GAP is shortened from global average pooling, which is used to reduce spatial dimensions of a three-dimensional tensor:

$$h \times w \times d \to 1 \times 1 \times d$$

The benefits of using global average pooling is that it is able to achieve object localization as well as classification, meaning that the neural network can learn different distribution patterns from training data. This is exactly what this task requires us to do.[8]

The original kernel based on this GAP layer as well as other components forms a much more complex network than that of the other kernel. We did not change structure of the kernel a lot, but we found that the final result is slightly overfitted. See Figure below.

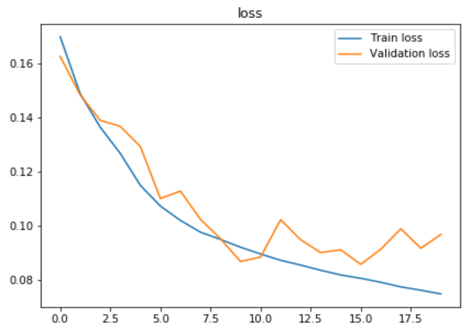

Fig 3.3 Learning curve of original kernel

So, we decide to increase the validation ratio from 0.1 to 0.2, which means that there are more validation date available to determine the real performance of our model. See the figure below.

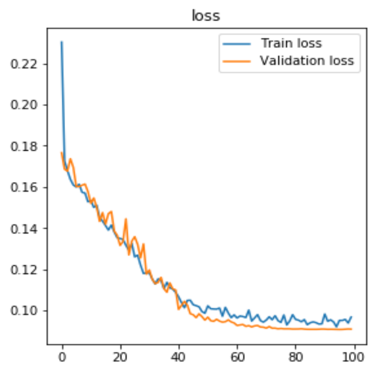

Fig 3.4 Learning curve of our model

What's more, like what we did on the other kernel, we add a callback to adjust the learning rate and change the number of epochs from 20 to 100; Hopefully we can obtain more information about the learning process and also prevent the model from getting stuck into a local minimum.
As for prediction, we found that the original calculator for the threshold of each class may be in troubles when none of the validation data contains a certain protein class. In this case, the threshold for this protein class will be 0, which is extremely terrible, because this means that every sample in our prediction will also contain this protein class. Therefore, we add a small intercept: 1e-4 to all the thresholds in order to fix this small bug.
By doing these things above, our model gets 0.422 on the leader board as the best score, which is an improvement about 10% compared to 0.385 from the original kernel.

Acknowledgements

Allunia. Protein Atlas-Exploration and Baseline. [Online] Available: https://www.kaggle.com/allunia/protein-atlas-exploration-and-baseline.
Vitaly Byrachonok. Pretrained InceptionResNetV2 base classifier. [Online] Available: https://www.kaggle.com/byrachonok/pretrained-inceptionresnetv2-base-classifier.
Michal Haltuf. GapNet-PL [LB 0.385]. [Online] Available: https://www.kaggle.com/rejpalcz/gapnet-pl-lb-0-385.

References

[1] Quora Q&A. What is meant be pre-trained model in CNN? Are they already trained on that particular classes? [Online] Available: https://www.quora.com/What-is-meant-be-pre-trained-model-in-CNN-Are-they-already-trained-on-that-particular-classes.
[2] cs231n. Convolutional Neural Networks (CNNs/ConvNets). [Online] Available: http://cs231n.github.io/convolutional-networks/#pool.
[3] Firdaouss Doukkali. Batch Normalization in Neural Network. [Online] Available: https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c.
[4] Quora Q&A. In Keras, what is a "dense" and a "dropout" layer. [Online] Available: https://www.quora.com/In-Keras-what-is-a-dense-and-a-dropout-layer.
[5] SAGAR SHARMA. Activation Functions: Neural Networks. [Online] Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.
[6] Jason Brownlee. What is the Difference Between Test and Validation Datasets? [Online] Available: https://machinelearningmastery.com/difference-test-validation-datasets/.
[7] Google AI Blog. Improving Inception and Image Classification in TensorFlow. [Online] Available: https://ai.googleblog.com/2016/08/improving-inception-and-image.html.
[8] Alexis Cook. Global Averaging Pooling Layers for Object Localization. [Online] Available: https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/.