Bioinformatics, YYYY, 0–0
doi: 10.1093/bioinformatics/xxxxx
Advance Access Publication Date: DD Month YYYY
Manuscript Category

Genomics

Human Protein Atlas Image Classification

Aditya Sinha[1*], Praneet Vibhu Balabolu[2]

[1]Department of Electrical Engineering, Columbia University, [2]Department of Electrical Engineering, Columbia University

Abstract

Motivation: By mapping the presence of proteins in different parts of the cell, we can identify malfunctioning cells, enabling us to diagnose genetic defects as well as possible pathogen invasion. This can also aid in dating the protein molecule to find time of synthesis.

Results: Our simple LeNet based parallel CNN framework shows that this problem can be solved by treating the reference images as separable. We also make an argument for use of batch normalization, data augmentation and weighted loss to improve accuracy for unbalanced classes.

Availability: The code for the model is available at *https://bitbucket.org/praneetvibhu/genomics* . The model predictions have been submitted to Kaggle with the Kaggle Id *adityasinha379*.

Contact: as5624@columbia.edu, pb2735@columbia.edu

Supplementary information: www.kaggle.com/c/human-protein-atlas-image-classification

1 Introduction

In this project, we aim to classify mixed patterns of proteins in different organelles of the cell. This problem is pertinent from a genomic viewpoint because proteins are both ubiquitous and essential to human life, and by mapping the presence of proteins in different parts of the cell, we can identify malfunctioning cells, enabling us to diagnose genetic defects as well as possible pathogen invasion. This can also help us identify, approximately, how long ago the protein blob was synthesized. Newly synthesized proteins are found closer to the ER, since ribosomes are mostly bound to the surface of the rough ER.

To accomplish this task, we are given the scans of the nuclei, microtubules and endoplasmic reticulum (ER). These three together define the entire structure of the cell, as the placement of the remaining organelles can be approximately determined using these as references. Therefore, these along with the actual protein distribution is sufficient for any machine learning algorithm to perform the required classification. In other words, the three reference images provide us with an outlay of the cell, which is sufficient to predict the localization of all the 28 organelles in which the protein could be present. This is analogous to the concept of a basis in mathematics, but not quite, since here, given the reference images, we can't come up with an analytical solution for the organelle localizations in space. This is our main motivation to use a machine learning algorithm based on experience (training data) to predict protein organelle localization labels.

The input data is in the form of sparse images, and for that purpose, we use a Convolutional Neural Network (CNN) based architecture. CNNs are widely used for image classification as they auto-generate the image filters and require minimal preprocessing to extract features. The CNN we used is loosely based on the LeNet architecture [1] proposed by Yann LeCun, used for character recognition and subsequent modified versions for object classification.

The main challenge posed by the problem at hand is due to the wide variety of cell morphologies for which protein data has been given to us.

2 Methods

To predict the protein organization localization labels, we are given, for each sample, 3 reference images and one image with protein locations. Since the network is complex, for simplicity of training, we first mean pool the input images with a 4x4 kernel to get 128x128 images.

Thinking along the idea proposed in the likes of [2], we propose a parallel architecture of 3 CNNs, with each branch getting one reference image and the protein image as input (2 channel input). In each of these networks, we use a convolutional, pooling and a batch normalization layer. This builds the correlation between the reference image and protein image (R+G, B+G, Y+G). Then, we proceed to flatten the outputs of the hidden layers to dense fully connected layers, and this is where we merge the outputs of the 3 CNNs into one fully connected layer, as shown in Fig. 1. We then do some further processing with another dense layer to account for relations between the reference images themselves and then take the outputs from an affine layer.

After feedforward operation of the batch, losses are calculated for the multi-label classification problem, where we use the cross-entropy values at the output to calculate sigmoid cross-entropy loss. For this purpose, we convert the target output for training to a one-hot (or multi-hot, in this case) tensor, ie. a binary array of 1's and 0's. This loss

is then backpropagated from all the output nodes with the standard backpropagation algorithm. One thing worth noting is that upon converting the target output for the batch to a one-hot tensor, we get a sparse binary matrix for training. This poses a problem as it results in the network's outputs to be biased towards 0. To correct for this, we modify the sigmoid loss by using a weighted loss, with a higher weight if the target output is 1, ie. the label is present.

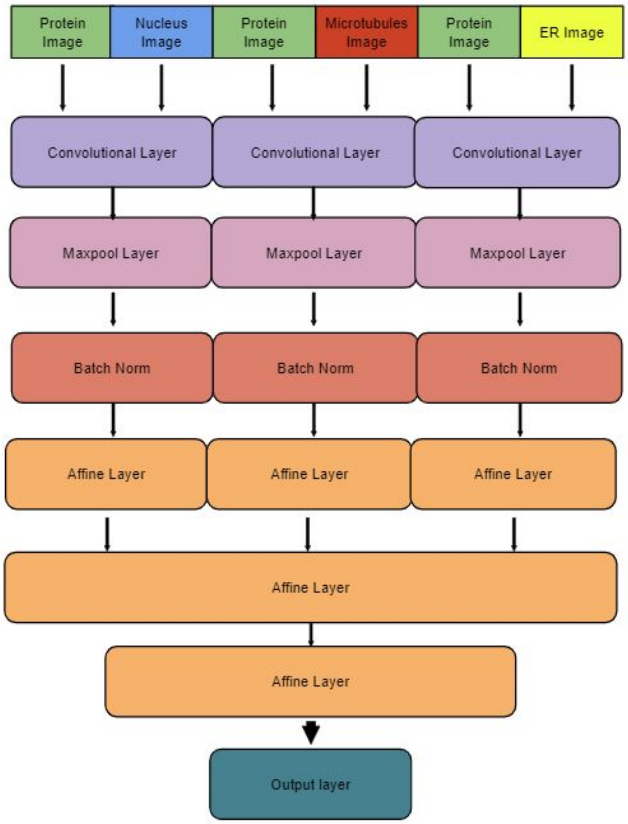

Figure 1: Architecture of the CNN

Another challenge posed by the data is the unbalance of classes, where labels 0 and 25 are much more frequent than the others in training, making the network biased to those labels and leading to overfitting. We have partially addressed this issue by introducing a batch normalization layer, but we posit that using data augmentation and changing the definition of loss to take into account the class unbalance would help improve accuracy and detection of other labels.

Due to RAM limitations, to perform training, the input images have been mean pooled to 128x128, leading to smoothening out of data and a loss in the possible number of features that could be extracted. This is also a big contributor to the low value of accuracy obtained. We suspect that using the 512x512, or better yet, the high-res 2048x2048 images would yield much better results.

3 Results

When we used the sigmoid cross entropy loss, the activations all reduced to zero. This was because each image has on an average only 2 labels, which in one hot encoding would mean 26 zeros and only 2 ones. This caused the model to converge towards all zeros, thereby giving no meaningful results.

Using weighted cross entropy loss, seemed to overcome this problem. This loss function increased the weight of positive interactions to allow the model to converge on them better than rather than dying out. This however, resulted in additional spurious labels being added to each sample. Therefore, with this neural network, we got an accuracy of 3.3% on the kaggle competition.

Using batch normalization after each convolutional layer managed to improve the imbalance in the data by normalizing the results over the batch. This increased the accuracy to 5.6%.

Model	Accuracy
CNN with sigmoid	0%
CNN with weighted loss	3.33%
CNN with batch norm and weighted loss	5.6%

4 Discussion

Our idea in this project was to identify the correlation of the protein image with each of the bases separately and combining these features to get the output. Compared to the basis images, the protein image is sparser. If all the four images would have been given to the neural network with equal importance, it is likely that the three basis images would dominate the result rather than the sparse protein image. Therefore, we ran three parallel CNNs each with one basis and the protein image and then correlating these three in one fully connected layer. We were unable to compare this architecture with existing architectures as we did not compensate data imbalance and also as we reduced the resolution to allow the system to run the neural network. Another major handicap to the system is the lack of data. We could use data augmentation, to increase the number of samples which will also soothe the data imbalance to a certain extent.

We have posited that increasing the model complexity would increase the accuracy. However, deep CNNs often suffer from the problem of vanishing gradients in backpropagation. To solve this problem for multi-label image classification, a possible solution is to use CNN-RNNs, as in [3]. Use of Long Short Term Memory (LSTM) cells and recurrent connections within the network add depth to the network while preserving gradients in backpropagation and preventing the training from getting stuck.

It is also possible to consider an ANN architecture with Directed Acyclic Graphs (DAG) to address the problem of multi-label classification of protein labels, as has been done in [4] for non-sparse data.

Acknowledgements

We would like to thank Prof. Wei-Yi Cheng for teaching us Introduction to Genomics, especially the machine learning parts, which were paramount in addressing this problem. Furthermore, we would like to thank Prof. Zoran Kostic, whose course on Neural Networks and Deep Learning has provided us with the necessary implementation skills and guided us in writing the code for the layer functions used in this project.

References

[1] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998. doi: 10.1109/5.726791

[2] Lidy, T. (2016). Parallel Convolutional Neural Networks for Music Genre and Mood Classification.

[3] Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., & Xu, W. (2016). CNN-RNN: A Unified Framework for Multi-label Image Classification. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2285-2294.

[4] Szalkai, B., & Grolmusz, V. (2018). Near Perfect Protein Multi-Label Classification with Deep Neural Networks. Methods, 132, 50-56 .