CS230

Final report: Kaggle Human Protein Atlas

Pascal Pompey, Iason Solomos
Department of Computer Science
Stanford University
papompey@stanford.edu, isolomos@stanford.edu

Abstract

This project will focus on applying deep-learning to understand the basic building block of the human body: proteins. Recent advances in medical imagery made it possible to gather sufficiently large data-sets to enable the application of deep-learning for protein recognition. This work will aim towards pushing the state of the art for protein identification.

1 Introduction and related work

Proteins are the lego blocks based on which the human body is built. Understanding their distribution and role is therefore critical to apprehending how our body functions. Recent advances in medical imagery make it possible to gain further insights by collecting large amount of cell data annotated with their proteins content in an attempt to use machine-learning to automate the annotation process.

A recent work applying deep-learning on medical images, Chexnet [8] has shown that it is possible to reach excellent results on medical images by using transfer-learning from models trained on real world images. Classification Models on real world images have progressed a lot recently and contain a wealth of literature; this section will only focus on notable breakthroughs. AlexNet [5] popularized Convolutional Neural Networks (CNNs). VggNets [9] introduced an appealing architectural pattern for selecting the number of filters and their sizes. ResNets [2] enabled training of very deep networks without suffering from exploding or vanishing gradients. InceptionNets [12] combined many operations in order to find the optimal architecture. Finally DenseNets [3] are the current state of the art for image classification on ImageNet [1].

2 Data-set and Problem Formulation

The data-set was provided by Kaggle [10] and comprises of $32k$ images with 512×512 resolution and 4 channels (RGBY). Each image is annotated with a list of labels indicating which proteins (amongst 28 protein types) are present in the image. It is therefore a classical multi-label classification problem. The metric of the competition is the mean of the f_1 score over all proteins.

The image distribution is known to come from 27 different cell types of very different morphology. It is also known that the proteins of interest are visible in the green channel while the other channels expose different types of molecular structures in the cell.

Analysis of the labels: there are 28 possible protein types, however, as indicated in Fig[1], the frequency of each image type varies a lot with some types being ubiquitous and others very rare.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

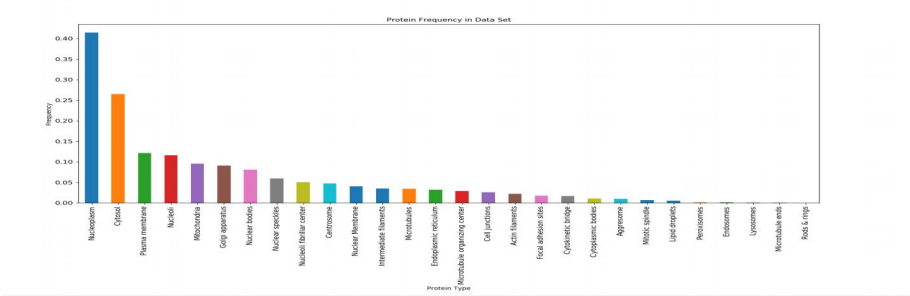


Figure 1: Representation of different protein classes in the data-set. Some proteins are ubiquitous while others are very rare. We observe an exponential decay in the frequency of proteins types.

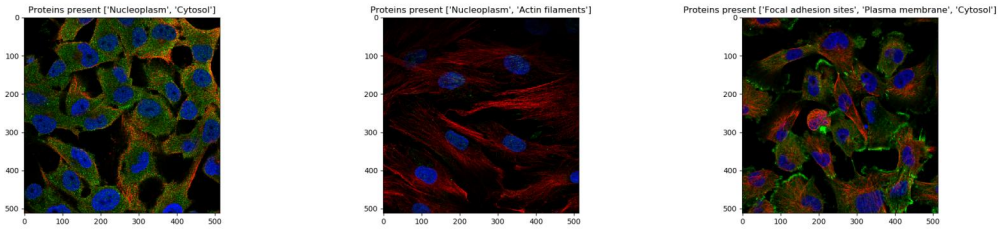


Figure 2: Samples from the data: one immediate observation is that these images are significantly different from that of ImageNet

Experimental setup The original labelled data-set from Kaggle was split into train, development and test data-sets using a $90/5/5$ split. On top of the labelled data, there also was a submission data-set which consists only of images (no labels) and on which our models' output was to be computed in order to submit our prediction to Kaggle for grading. Prior to being used in our models, all images were normalized. Two data augmentation techniques were applied to the images: (1) horizontal and (2) vertical flip.

Hyper-parameters setup The Adam optimizer [4] was used to train the model. We found that the standard parameters used for training resnets performed best: $lr = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. When possible, parameters were fully transferred from the resnet18 model from torch-vision [7], which was pre-trained on the ImageNet data-set [1]. A batch size of 8 was found to maximize GPU efficiency without slowing learning convergence.

3 Transfer-Learning: Adapting Networks from ImageNet

From multi-class to multi-label classification: Some modifications were required to adapt the original ImageNet architecture (which is a 1000 classes multi-class classification problem) to the 28 labels multi-label classification problem of this project. (1) The last fully connected layer was replaced to predict 28 labels and not 1000 classes. (2) The 28 outputs of the last connected layer were fed into a sigmoid representing the probability of the given protein to be in the image. (3) The loss optimized was changed from the Cross-Entropy loss to the Binary Cross Entropy loss (BCE) between the sigmoid's output and a multi-hot encoding of the true labels. Binary cross entropy has the advantage of handling each labels' prediction independently.

Freeze or retrain: Freezing the model except the last layer yielded f_1 scores of 0.2 (vs. 0.8 for the retrained version) on the training set and 0.05 (vs. 0.3) on the test set. This indicates the frozen version was unable to over-fit the training set and therefore suffered from a high bias problem. This can be explained by the fact that protein images form a very different manifold of images than that covered by real-world images; as is abundantly apparent when looking at pictures of Fig[1]. Therefore retraining deep-features in the deeper layers of the network is necessary.

Adapting the image size: The model also needed to be adapted to the 512×512 resolution of images in this project (as opposed to the 256×256 resolution of the ImageNet data-set). Four

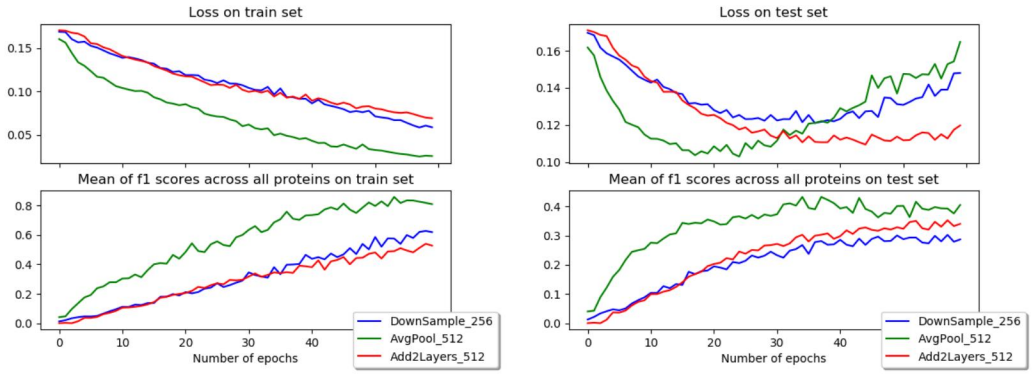


Figure 3: Performance of different techniques to adapt a Resnet18 to a 512×512 image resolution using an RGB model with BCE loss. Method (3), which does an average pooling before the fully connected layer outperforms the two other methods. Adding 2 more layers (Method(4)) seems to lower overfitting. The AvgPool method presents strong overfitting.

methods were experimented with. (1) Change the size of the last fully connected layer to take the new size coming out of the convolutional layers when applied to the new image size. (2) Downsize the original (512×512) image to images of size 256×256 before passing it through the network. (3) Add an adaptive pooling layer right after the last convolutional layer to average its results and downsize them to the usual input size for the fully connected layer. (4) Append additional residual layers which will reduce the image size and increase the number of filters Method (1) did not converge. Results for methods (2), (3), and (4) are presented in Fig.[3].

Adding the Yellow channel: To add the yellow channel, we simply modified the first convolution layer of the resnet model to take a fourth channel. The yellow channel was Xavier [13] initialized while the RGB channels were transferred learned from the ImageNet [1] Resnet model. In our experiments adding the yellow channel to the network did not yield much improvements.

4 Handling class imbalance

The key challenge of this competition is class imbalance. Fig[1] shows the strong exponential decay in labels frequencies: most of the proteins types are very rare and present an imbalance of more than 1 in 10. The usual method for handling class imbalance is to weight the loss more for samples with high information content w.r.t rare labels. The weights, however, must be carefully chosen. Indeed, multiplying the loss by a factor α is the exact mathematical equivalent of modifying the step-size in the SGD optimization step by α. The step size is known to be a crucial hyper-parameter and modifying it carelessly can easily lead to divergence or slow convergence. As slow convergence is preferable to divergence, we required our weights to be strictly in the $[0, 1]$ interval.

Weighted BCE: The first weighting scheme to be applied was to weight each label by its rarity. If a label l had frequency f_l, the BCE loss coming from the neuron responsible for that label was weighted by $1 - f_l$, the probability of the label not to be present.

Focal Loss: One elegant method for choosing the weight for handling class imbalance is the Focal Loss [6]. We define focal loss as:

$$Focal\_Loss(y^*, \hat{y}) = [y^*.\hat{y} + (1 - y^*).(1 - \hat{y})]^\gamma \; BCE(y^*, \hat{y})$$

where y^* is the one hot encoded true label, $\hat{y}$ is the logit predicted by the network, γ is a positive integer and $bce(.)$ is the binary cross entropy loss (following in [6], we chose $\gamma = 2$). This definition is mathematically equivalent to that of the original paper [6] but explicitly exposes the focal-loss as being in essence the same formula as the bce, just without the logarithms.

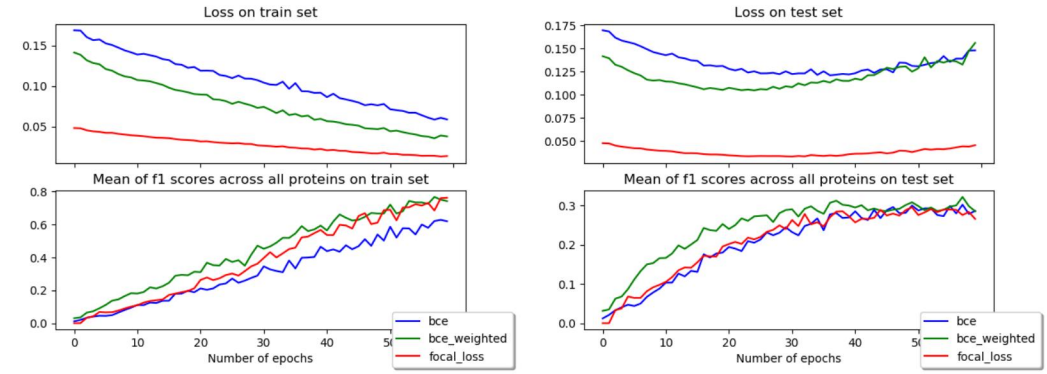


Figure 4: Performance of a resnet18 depending on the weighting scheme applied to the loss. The weighted BCE method seems to perform best and speeds ups learning without being detrimental to its quality. The BCE vanilla and the focal loss eventually reach a similar performance.

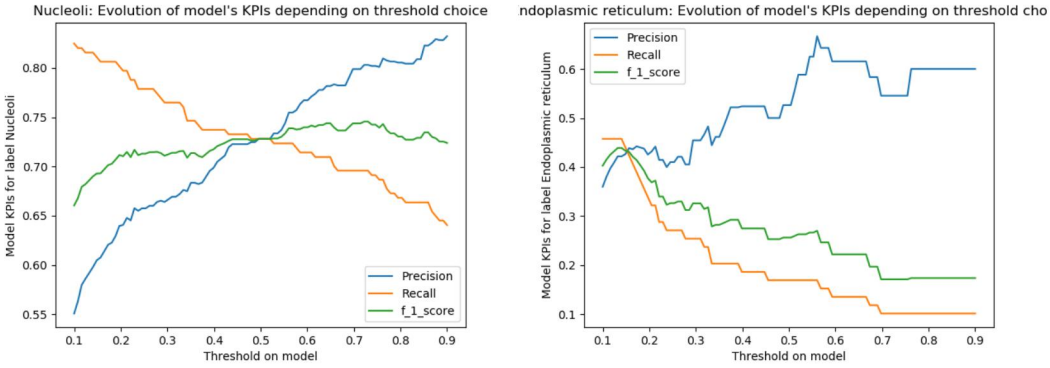


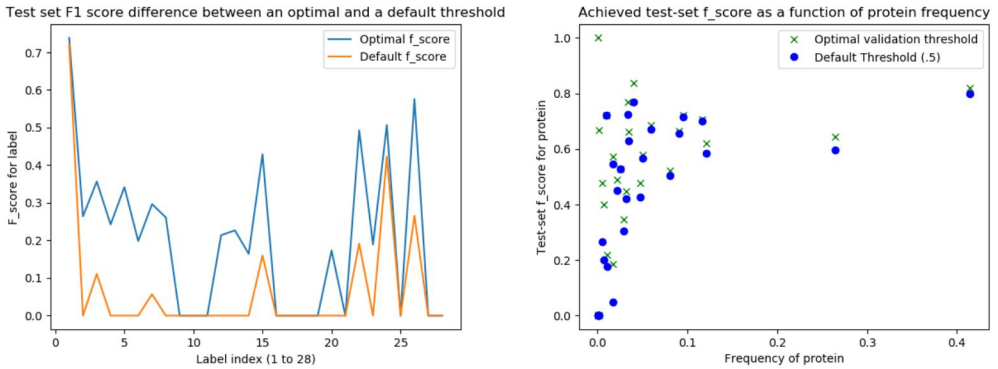
Figure 5: Metrics graphs on the validation set: precision improves with a higher threshold while the recall diminishes. Choosing a threshold different from 0.5 is beneficial for rare proteins.

The better loss: Fig[4] shows the comparison of the different weighting schemes explored in this work. While the weighted BCE speeds up learning, none of the weighting schemes improves the model's performance at convergence. The team suspects that this is due to the BCE loss already having some non linear weighting built in. The $\log$ terms in the BCE loss already enforce that large mistakes on the model's part will be more strongly penalized. That built in penalization is exponential in the model's own error. The team suspects that the effect of multiplying the loss linearly is dwarfed by the BCE's built in log term, and therefore, of limited use.

5 Hyper-parameter tuning and error analysis

Classification Threshold: Using sigmoid with the binary-cross-entropy loss, a question still remains open after training a model: which threshold value shall be chosen to separate the positives from the negatives examples. Fig.[5] shows that choosing a threshold different from 0.5 can be beneficial, especially for rare proteins, where choosing a lower threshold enables to give more emphasis to recall as opposed to precision. As the threshold value is an hyper-parameter, all the metrics were calculated applying the model to the validation set. This simple trick enabled the team to bump its test set f_1 score from 0.43 to 0.53; yielding a significant 0.1 improvement. Fig[6a] shows the gains achieved per label by tuning the threshlod hyper-parameter.

Error analysis: Despite it being very difficult to make anything of protein pictures with an untrained eye, one pattern soon became clearly visible: the lowest f1-scores were reached on rare protein types.



(a) Test set variation of the F1 Score depending on the threshold selected on the validation set.

(b) In general the higher the frequency, the higher the f_1 score.

Figure 6

Fig.[6b] illustrates this clearly. There is a marked inflection point once a protein type is present in less than 5% of the images from which the model performance can degrade spectacularly.

This is a remarkable difference between multi-label and multi-class classification problems: in multi-class problems, every image in the set provides information about one class and the cross-entropy loss ensures that knowledge is spread out to all the other classes. Multi-label classification on the other hand, is more of a multi-task learning problem where all labels are co-trained but each is still handled completely independently from a loss perspective. Therefore, labels with very rare positive cases can be subject to learning starvation if their learning task doesn't correlate with that of other more frequent labels. Fig[6b] indicates that some of the very rare labels do benefit from co-training with more frequent labels and reach f_1 scores that are equivalent or higher than the frequent labels despite very strong data-imbalance. Using multi-task learning makes sense for these cases.

But for the other rare labels, using the whole data-set in a multi-task learning setting is actually detrimental: these labels do not benefit enough from co-training with other labels to compensate for their structural class imbalance. For these labels it would probably be beneficial to define a dedicated learning problem and train a dedicated model on a sub-sample of the original data-set constructed in a way that corrects the data imbalance.

6 Conclusion and Future work

This work focused on identifying the most promising architecture for solving the Kaggle Human Protein atlas multi-label classification challenge [10]. Multiple architectures were trained using ResNet18 [2] as base model and it was found that using Average Pooling with a weighted BCE loss on 512 resolution RGB images was the best setup.

Subsequent optimization of the threshold used to separate positive from negative cases enabled to further increase our f_1 score from $.43$ to $.53$ on the test set, which is competitive with current Kaggle submissions despite it still being based on a simple Resnet18 model.

Finally a closer analysis of the link between class frequencies and the model's f_1 metrics sparked a discussion whether multi-task learning is detrimental for some rare protein labels.

Future work will focus on rare labels that still perform poorly. The team would let go with multitask-learning and each label will have its own dedicated binary classification problem and model. In this setup class imbalance can be handled by sub-sampling the training data in a way that enforce at least 20% of the samples contain the target label. The positive data-points could also be subjected to more aggressive data-augmentation in an effort to generate less imbalanced training sets.

Now that the core architectural questions have been answered, the team is also keen on experimenting with more advanced models like ResNet31 [2], DenseNets [3], or Inception ResNets [11].

7 Contributions

Pascal contributed the AWS setup for training models, the implementation in Pytorch [7], data pre-processing, the various resNet18 architectures, the training and evaluation routines, along with the utilities for computing multi-label metrics, storing model training histories, and generate relevant train/validation/tests graphs. Pascal also conducted the experimental methodology and result analysis. Pascal also wrote the core of the milestone report, final report and poster.

Iason contributed the Microsoft Azure setup for training models as well as the DenseNet implementation.

Our code is on `https://github.com/PascalPompey/cs230_human_protein_atlas`, this repository will be made public once the Kaggle competitions finishes on the 10^{th} of January 2019.

Acknowledgments

We would like to thank Ahmadreza Momeni for his constant guidance and useful feedback throughout the project. We would also like to thank AWS Educate for the AWS credits without which carrying the amount of experiments needed for this work would have not been financially viable.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[3] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[6] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[7] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[8] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *CoRR*, abs/1711.05225, 2017.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[10] Devin P. Sullivan, Casper F. Winsnes, Lovisa Åkesson, Martin Hjelmare, Mikaela Wiking, Rutger Schutten, Linzi Campbell, Hjalti Leifsson, Scott Rhodes, Andie Nordgren, Kevin Smith, Bernard Revaz, Bergur Finnbogason, Attila Szantner, and Emma Lundberg. Deep learning is combined with massive-scale citizen science to improve large-scale image classification. *Nature Biotechnology*, 36:820 EP –, Aug 2018.

[11] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[13] Yee Whye Teh and D. Mike Titterington, editors. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*. JMLR.org, 2010.