Investigating deep learning algorithms in the context of Human Protein Image Classification

Mélanie Bernhardt
ETH Zürich
melanibe@ethz.ch

Mélanie Gaillochet
ETH Zürich
gamelani@ethz.ch

Andreas Georgiou
ETH Zürich
geandrea@ethz.ch

Dimitrios Koutsoukos
ETH Zürich
dkoutsou@ethz.ch

Abstract

This project aims to investigate how deep learning architectures and algorithms affect the accuracy of neural networks in the area of image classification. Based on the Human Protein Image Classification task, our goal was to assign one or more protein labels to 4-channel medical images, given a set of 28 possible labels to choose from. To solve challenges posed by the dataset, we investigated different techniques to overcome class imbalance and tested different thresholds for the final output probabilities. With regards to classification itself, we started by increasing the complexity of simple Convolutional Neural Networks (CNNs) and integrating different regularization methods such as dropout, batch normalization or data augmentation. We also experimented with multiple architectures that are considered state-of-the-art in the current literature. In the end, our investigation led us to propose 3 models: Densely Connected Convolutional Networks, Residual Convolutional Networks as well as a novel combination based on model averaging, where we combined the predictions of the aforementioned two models. Our best model achieved an F1-score of 0.248 on the test set (where the score was provided as part of the Kaggle competition results).

1 Introduction

Starting as a Computer Science specificity, Deep Learning has been progressively used in many other domains. One of the most prominent ones, due to the vast amount of unprocessed data available and also its importance for humans, is biology, where numerous approaches, such as genomics and biological image analysis, have been studied [1][3]. Our Deep Learning project focuses on the latter part and is based on the "Human Protein Atlas Image Classification" Kaggle competition. Based on medical image analysis, the main goal of this competition is to classify mixed patterns of proteins in microscope images using computer vision. In this project, our objective is to assess the impact of different network architectures and regularization techniques on the performance of our neural network classifier for this image classification task. Our focus is thus more oriented towards testing the impact of each of these architectures and techniques than engineering a good final classifier.

Human cells contain proteins that control most cellular processes. The localization of proteins at the subcellular level impacts cellular functions and interactions. Biologists are thus greatly interested in gaining knowledge of the spatial organization of proteins in the cell as it helps to understand its underlying mechanism [2]. The images that we are analyzing are obtained through an immunofluorescence technique. As explained in [2], this approach allows us to visualize the protein of interest in green, while reference markers for microtubules, endoplasmic reticulum and nucleus that outline the cell appear respectively in red, yellow and blue. For this reason, each sample in the dataset is represented by four images (one per filter). From small dot-like nuclear bodies, to larger structures such as the nucleus, the distinct patterns in the images together with the reference markers make it possible to precisely determine the spatial distribution of a protein within the cell. The protein's location can then be assigned to one or several of the 28 structures and substructures currently annotated in the training set[1]. This training set was composed of 31,072 images with corresponding labels, while the test set contained 11,702 images. Each sample was comprised of four 512x512 input channels, one per filter color (red, blue, green, yellow) obtained via the immunofluorescence technique described previously.

The task presents several challenges. The first one involves the nature of the task: multi-label and multi-class classification. Indeed, image classification with many classes (up to 28 in our case) is challenging, particularly when several labels can be

[1] Figure 1 in [2] shows an example of such annotated images.

assigned to a sample. The second challenge revolves around the dataset itself, both because of its composition and distribution. The samples we are handling are 4-channel input images, requiring the modification of the standard implementation of common architectures - usually meant to be used with RBG 3-input channels images. Finally, different methods have to be developed to prevent a highly imbalanced dataset (for further detail look into Section 2) from luring the networks into predicting the most dominant classes in the training dataset.

Our investigations can be summarized as follows. We have first implemented a non-neural approach based on a random forest classifier, where features were manually extracted from the original images. This was used as a baseline to compare later results. We also implemented a simple 4-layer convolutional network with and without regularization techniques (such as dropout and batch normalization). We then increased the complexity of the models tested by adapting both networks presented in related biology work (DeepLoc, DeepYeast) to our task and famous standard architectures (ResNet, DenseNet) used in image classification contexts. We further increased the complexity of our model by using a bagged version of ResNet architecture and by model averaging. Finally for our prediction we experimented with different thresholds on the output probabilities and we also applied data augmentation to compensate for the class imbalance.

2 Models and Methods

2.1 Baseline models

To determine how good our deep neural architectures were, we implemented a baseline based on a random forest classifier. The inputs to the classifier were the outputs of the feature extraction process whereby the 512x512 images were divided into 64 patches of size 64x64. To extract the final features, histograms with 10 bins were computed for each patch. For our classifier, we used 1000 decision trees. The number of decision trees to use, was determined using 3-fold cross validation.

Through this project, we wished to determine and quantify the influence of parameters such as input size, layer depth or simple regularization techniques on the performance of deep neural networks, given our initial task. We thus began by implementing a simple 4-layer convolutional neural network with ReLU activation functions and max pooling. We trained the model in two different versions: one without regularization (refered to as "CP4"), and one with added dropout [11] and batch normalization [5] in each of the 4 convolutional blocks (refered to as "CBDP4"). Results yielded by the baseline models described above are shown in Table 1 of Section 3.

2.2 Models from related work

Just as with many image classification tasks, Convolutional Neural Networks (CNNs) are a prominent method in microscopy images. In [9], where the task was multi-label classification with 12 classes, the authors introduced DeepYeast, an 11-layer CNN. Their network achieved an accuracy of 0.91 on the test set over 12 subcellular localizations. We implemented the 11-layer network with its 8 convolutional layers (of respectively 64 units for the first 2 layers, 128 for the next 2, and 256 for the rest) and 3 fully connected layers (of 256 units each) (see Figure 2 of the Appendix). We must note that while displaying some similarities, the original task itself was slightly less broad than ours (only 1 label required out of a possible 12), and the dataset was accordingly also slightly different, since the original dataset used for DeepYeast contained only sub-images of a single protein. In opposition to that, the data we were working on could contain images of several proteins. We also implemented DeepLoc, a CNN with an architecture very similar to that of DeepYeast developed by [7], which outperformed the previous SVM approaches on protein subcellular localization with an average precision of 0.84 on a 16-label classification task.

Resnet is a famous architecture of the network that was originally trained for the classification task on the ImageNet dataset containing 1.28 million training images assigned to one class out of 1000 possible classes. It was originally introduced to compensate with the vanishing gradient problem that very deep neural networks face. It is based on the implementation of a "plain" network, where shortcut connections are added to skip pairs of 3x3 convolutional layers (see Figure 3 of the Appendix). The plain network itself can be composed of a variable number of convolutional layers. However, for our task, we used the versions with 34 and 101 convolutional layers. Results yielded by our implementation of the aforementioned models are shown in Table 1 of Section 3.

In order to deal with the lack of contribution carried by some layers of the Resnet, [4] developed Densenets, Densely Connected Convolutional Networks, that are very narrow and add a noticeably small set of feature maps. Their main feature is that each layer has direct access to the gradients from the loss function and the original input image. However, instead of having a skip connection like in Resnets, DenseNets concatenate (rather than sum) the output feature maps of the layer with the incoming feature maps. To compensate for the problems caused by the different feature maps size, Densenets follow the same approach as Resnets and are divided into DenseBlocks, where the dimensions of the feature maps remain constant within a block. What changes between each block is the number of filters. Finally, the network is defined by Transition layers between DenseBlocks that are responsible for downsampling, and a growth rate, which controls how much information is added to the network at each

layer. In our case, we experimented with the most widely used Densenet, comprised of 121 layers. A schematic representation of Densenets can been seen in Figure 4 of the Appendix.

2.3 Approaches to class imbalance

In addition to the high number of possible labels we could predict from - each sample could have one or more labels among a choice of 28 (as mentioned earlier), the training dataset was also highly imbalanced, with a ratio of 11:12885 between the least frequent and the most frequent class label. That means that some classes were severely under-represented (see figure 1 of Appendix, which shows the number of times each label was assigned to an image). Thus, to account for this class imbalance and prevent the networks from solely predicting the dominant class (since this would yield a high accuracy), we investigated several methods including using the focal loss function and data augmentation.

As explained in [8], focal loss can be used in strongly imbalanced datasets to prevent the vast number of easy negatives from overwhelming the classifier during training. In other words, focal loss restrains the model from solely predicting the most frequent class by reducing the loss contribution from those dominant examples. It adds a regulating factor $(1-p_t)^\gamma$ with a tunable parameter γ to the cross-entropy loss, such that the focal loss can be defined as $FL(p_t) = -(1-p_t)^\gamma \log(p_t)$. At the data level, we rebalanced the data distribution by using data augmentation on the training dataset (excluding the data used for validation). This technique allowed us to create new samples for the under-represented classes to reduce the imbalance ratio, by rotating and flipping each image k times where

$$k \propto \frac{\#\text{ images with most frequent label}}{\#\text{ images with observed label}}$$

Given that images could have several labels, the minimum number of transformations was always taken. We must note that since the class imbalance was quite important, data augmentation could only slightly improve the class imbalance ratio. Indeed, since we could only generate up to 7 "new" images (3 rotations and 1 flip possible), the least frequent labels still remained the least frequent labels. In addition, because several labels could be assigned to the same sample, the augmentation of an image with a less represented label class could also lead to the increase in frequency of a well-represented label class.

2.4 Ensemble methods

2.4.1 Bagging

In order to prevent our model from overfitting and improve the efficiency of our classifier, we applied bagging (or bootstrap aggregation) to our different models. Indeed, bagging has been known to reduce the variance of the decision rule by aggregating the outputs of a large number of models, where each model was trained on a bootstrap sample of the original training set. Thus, we applied bagging to the Resnet model which was one of the best performing models (considered on its own) on the test set. We trained 10 times this model for 10 epochs on 10 different bootstrap samples.

2.4.2 Model averaging

We also investigated the impact of model averaging as a method to improve the performance of our model. The main idea behind these ensemble methods is to combine different neural networks, in order to obtain a more robust classifier. More precisely, we used our 2 best models to predict probabilities for each class, and then averaged the probabilities over all the models. The final predicted classes were then based on these averaged probabilities. Our final averaging model was composed of the averaged predictions from Resnet-34 and Densenet-121 (individual results of these models can be found in Table 1).

2.5 Training procedure

All our models were trained using the ADAM optimizer [6] with a learning rate of 10^{-4}. We also experimented with different learning rates; however, we did not observe any difference in terms of performance. Training was done by optimizing the cross-entropy loss function. We used 90% of the dataset for training and 10% for validation. The validation set was used to spot overfitting, tune the number of epochs to use and choose the prediction function. We also implemented the weighted cross-entropy (weighting each class contribution to the loss by the inverse frequency of this class) and the focal loss (explained in more detail in Section 2.3) as well as directly optimizing F1-score as a loss function. However, these approaches did not show an overall improvement of the final F1-score, and we chose to report only the results obtained with the standard cross-entropy loss.

2.6 Label Prediction

Besides investigating the models and regularization techniques presented above, we also tried several techniques to predict labels from the output probabilities. First of all, it is important to note that the predicted probabilities are obtained by applying

a sigmoid output activation function as in a binary classification context and not a softmax function as in standard multi-class tasks. Hence, our network predicts the probability of each class of being correct independently of the other class probabilities. This implementation had to be used because in a multi-label classification task, different labels can be predicted at the same time, independently from the other labels. There are mainly 2 ways to predict labels starting from class probabilities:

- Predict as the correct class every class for which the probability is higher than 0.5. Given this setting, however, most of the images had no class predicted at all (i.e. all probabilities were below 0.5), which was coherent with the fact that there were 28 possible classes. For all such images, we simply predict the most probable class as the correct one and set all others to 0. However, this breaks the class independence assumptions.
- Predict using a custom threshold lower than 0.5. This gets around the problem of not predicting any classes, since it increases the sensitivity of our models. For example, every time a predicted probability is above 0.05, the predicted label is added to the list of correct classes.

We aimed to find how to predict the labels in order to maximize the F1-score on the test set. Hence, we used 4 different prediction functions (first method and second method with 3 different thresholds - that is 0.05, 0.1 and 0.20) on the validation set and monitored the validation F1-score for each prediction technique during training with a Tensorboard graph. It appeared that the second method was more appropriate than the first method regardless of the model trained. The best thresholds were 0.1 and 0.05. As these two thresholds appeared equivalent, we chose 0.05 to make sure that at least one class label is always predicted.

3 Results

In the following section we will detail the results in terms of F1-score obtained by our different models and compare them with our baselines. Note that the evaluation metric used in this project (and in the Kaggle competition) is the macro F1-score defined as:

$$\text{F1} = 2 \cdot \frac{P \cdot R}{P + R}, \text{ where } P = \frac{TP}{TP + FP}, \ R = \frac{TP}{TP + FN}$$

and TP : True Positives, FP : False Positives, FN : False Negatives

Model	Batch size	Number of epochs	Kaggle public test set	Kaggle private test set
Baseline Models (No neural network and simple CNN's)				
Random Forest	-	-	0.041	0.037
CP4	128	7	0.132	0.133
CBDP4	64	15	0.104	0.106
Models from the literature				
DeepYeast	32	15	0.150	0.155
DeepLoc	16	15	0.191	0.184
Densenet-121	16	25	0.242	0.248
Resnet-101	16	25	0.219	0.214
Resnet-34	16	25	0.236	0.233
Training with data augmentation				
Resnet-101	16	14	0.226	0.215
Ensemble methods				
Bagging Resnet-101	16	10	0.157	0.160
Model averaging	16	25	0.241	0.243

Table 1: Results given in terms of F1-score, for each model tested. The test set contained 11,702 images of which 29% belonged in the public test set and 71% in the private one. All models (except for the Random forest one) used input images resized to 256x256 pixels. The number of epochs was chosen by visual inspection of the validation loss (to avoid overfitting). The details of each model are presented in Section 2).

Our best result was given by the Densenet-121 model, with a an F1-score of 0.248 (resp. 0.242) on Kaggle's private (resp. public) test set.

4 Discussion

As shown by Table 1, the use of different models and regularization methods had different effects on the F1 score obtained on the (public and private) test sets. The scores we achieved have to be put in perspective with the number of challenges posed by the task (see Section 1). Even though our best score could not exceed the threshold of 0.250, the results that we obtained allow us to make several observations.

First, the fact that our random forest classifier obtained the lowest score on all (public and private) test sets by a significant margin indicates that this task requires the use of neural networks that can extract complex features. More than that, since the two simple 4-layer CNN's ("CP4" and "CBDP4") obtained the lowest score on the test sets, we can infer that complex models (deeper architectures with more than 10 layers) are more suited for this type of image classification task.

As shown by the F1 score of 0.155 obtained on the private test set for our model based on DeepYeast, models that performs well on a similar task do not necessarily perform well when the task and dataset are slighlty modified. The same applies for the DeepLoc architecture, although the results in that case are slightly better. The difference in terms of performance of our model compared to the one reported in [9] could be explained by the fact that DeepYeast was originally tuned for classification where only 1 predicted class was allowed out of the 12 (rather that 28) classes. Another important factor is that the dataset consists of 4-channel images and multiple labels are correct for a single sample.

Then, we can see that bagging did not produce the desired results. An explanation for that could be that the resampling in such an unbalanced dataset gives even more emphasis on the dominant classes. Unfortunately, because training on the augmented dataset was too expensive in terms of computational and time requirements, we could not train the bagging model on the augmented dataset. However, with sufficient computational resources, it would be interesting to investigate the results that one could obtain using the bagged model on the augmented data. In addition to that, our most complex model involved taking the average prediction of the two models that obtained the best scores on the private test set (Resnet-34 and Dense-121). However, the result from this average was not better than the score obtained by the best of these models. It seems that the corresponding models had a high agreement in the classes that they predicted, which could not allow their averaging to surpass the individual scores. Furthermore, training with an augmented dataset did not significantly improve the result of Resnet-101. Finally, given that transfer learning rarely improves the accuracy of trained models but is mainly used for saving computational time in retraining, and given our previous results, we decided not to experiment with it as we realized that our dataset may be too different from the ones used in other protein identification challenges.

Finally, we must note that our experimentation with hyperparameters such as learning rate, optimizer and the loss function type did not yield better scores. With that and all the previously mentioned observations, we can conclude that model architecture is one of the most essential factors in solving tasks like image classification.

5 Summary

In this project we tried to investigate the effect of various deep learning architectures and algorithms in solving the Human Protein Image Classification task. Given the nature and distribution of our dataset, we investigated several techniques to overcome multi-labeling problems and class imbalance. By following a structured approach and starting from very simple models we investigated how depth and regularization affect the performance of neural networks. Our experiments confirmed that by increasing the depth, the model makes better predictions. Interestingly, architectures like Resnets and Densenets, which provide some direct connections between layers, are likewise very helpful in increasing the amount of information the network learns. Finally, we experimented with different thresholds on the output probabilities of the models and added a level of complexity to our models by combining the best models through probability averaging and bagging. Our results[2] show that image classification can be intricately difficult; nevertheless, the fact that challenges like the Human Protein Atlas Classification task engage an increasing number of people shows that deep learning has an extremely high potential in the field.

[2] All our code and our experiments can be found in this github repository: https://github.com/dimkou/DL-AtlasKaggle

References

[1] Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular systems biology*, 12(7):878, 2016.

[2] The Protein Atlas. The organelle proteome. `https://www.proteinatlas.org/humancell/organelle`. Accessed: 2018-11-30.

[3] Juan C Caicedo, Sam Cooper, Florian Heigwer, Scott Warchal, Peng Qiu, Csaba Molnar, Aliaksei S Vasilevich, Joseph D Barry, Harmanjit Singh Bansal, Oren Kraus, et al. Data-analysis strategies for image-based cell profiling. *Nature methods*, 14(9):849, 2017.

[4] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269. IEEE, 2017.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Oren Z Kraus, Ben T Grys, Jimmy Ba, Yolanda Chong, Brendan J Frey, Charles Boone, and Brenda J Andrews. Automated analysis of high-content microscopy data with deep learning. *Molecular systems biology*, 13(4):924, 2017.

[8] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[9] Tanel Pärnamaa and Leopold Parts. Accurate classification of protein subcellular localization from high-throughput microscopy images using deep learning. *G3: Genes, Genomes, Genetics*, 7(5):1385–1392, 2017.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Appendix

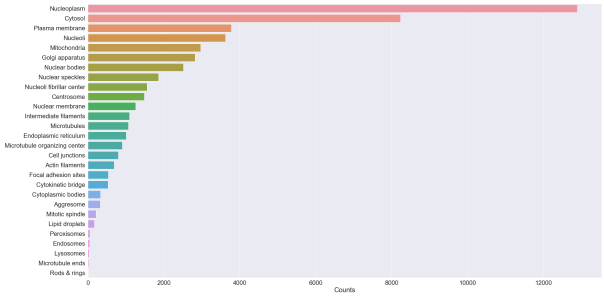

Figure 1: Frequency counts for each of the 28 labels in the dataset consisting of 31072 samples. Note that samples can have several labels.

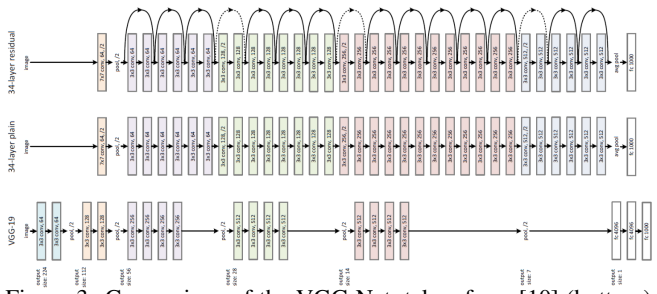

Figure 3: Comparison of the VGG Net, taken from [10] (bottom), with a 34-layer CNN (middle) and a residual 34-layer CNN (top).

Figure 2: Architecture of DeepYeast (image taken from [9])

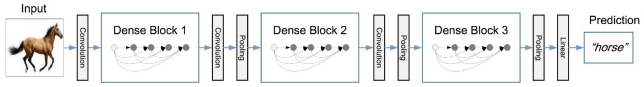

Figure 4: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling. Image taken from [4]