Predicting Protein Organelle Localization with Transfer Learning

Ruge Zhao (rugezhao), Siyi Tang (tsy935)

I. Introduction

A. Motivation

Recent advances in high-throughput microscopy has provided the opportunities to better understand complex protein patterns in human cells [1]. However, such microscope images are generated at faster pace than how human experts could manually evaluate them. Machine learning techniques have enabled automatic classification of protein subcellular localization in large-scale imaging data [2] [3]. The Human Protein Atlas (HPA) has released a Kaggle image classification challenge with large-scale microscope images of human cells [1].

B. Related work

Previous studies on predicting protein subcellular localization from images used various machine learning techniques. The pioneering work [4] applied an artificial neural network to hand-crafted features extracted from fluorescence microscopy images to classify 10 different subcellular patterns, and the classifier was able to achieve 83% test accuracy. [5] compared the classification performance among eight models, including neural networks, support vector machines (SVMs) and ensemble methods such as AdaBoost and Bagging. More recently, [2] combined a deep convolutional neural network (CNN) with multiple instance learning to achieve classification and segmentation of microscopy images simultaneously with whole image level annotations. [3] trained a 11-layer CNN on yeast cell images for protein localization prediction, and assessed the generality of the learned features in the network layers. These recent works have demonstrated the feasibility of using deep learning approaches in classifying protein subcellular localization from cell images. However, most prior studies focused on single protein pattern in a single cell type, ignoring the fact that proteins exhibit a mixture of patterns across human cells. Therefore, models capable of classifying mixed patterns of proteins in more than one cell type are needed.

In the field of machine learning, transfer learning refers to training a base network on a base dataset and task, and then transfer the learned features to train a target network on target dataset and task [6]. In recent years, transfer learning of deep neural networks has been widely applied to image classification tasks [7] [8] [9], and has achieved state-of-the-art performance.

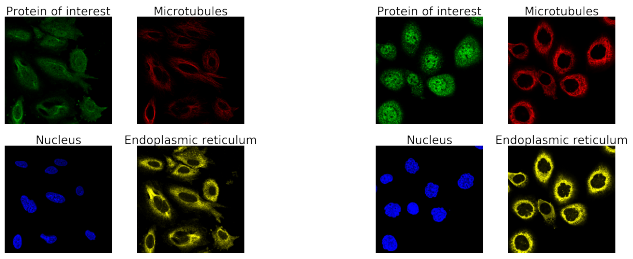
C. Problem statement

In this paper, we aim to develop algorithms to predict multiple protein organelle localization labels in human cell samples. More specifically, we framed the problem as a multi-class multi-label image classification task. The inputs to our algorithms are microscopy images of human cells. The outputs are predicted protein organelle localization labels for each sample. We applied logistic regression, random forest and transfer learning of deep neural network to the imaging data. Furthermore, we explored two loss functions as well as various training techniques.

II. Data

A. Data visualization

We used only the labeled data (31072 samples) provided by HPA in this paper. The dataset consists of microscopy images of human cell samples. Each sample is represented by four channels. The protein-of-interest is represented by the green channel, whereas the blue, red and yellow channels represent the subcellular landmarks: nucleus, microtubules and endoplasmic reticulum respectively. Each sample is labeled with one or more protein organelle locations. Since different cell types have highly different morphology, the protein patterns across samples are very heterogeneous. Fig. 1 shows two cell samples. Although both are labeled with "Nucleoplasm", the protein patterns in the green channel are visually different.



(a) Four channels of first cell sample

(b) Four channels of second cell sample

Fig. 1: Four channels of two cell samples

B. Imbalanced classes

Classes are highly imbalanced in the dataset. For example, the most common class "Nucleoplasmn" occurs 12885 times, whereas the least common class "Rods & ring" only occurs 11 times in the labeled data.

C. Training, validation and test sets

We randomly split the labeled data (31072 samples) by 80%/10%/10% into training (24858 samples), validation (3107 samples) and test sets (3107 samples). The ratio of the classes are similar in all three sets.

III. METHOD

The task was setup as a multi-class multi-label image classification problem. There are 28 classes in total, and each sample is associated with one or more labels. Fig. 2 shows an overview of preprocessing and the transfer learning model. The details of each step as well as the baseline models are described in the subsections below.

A. Preprocessing

We combined the four filters (512×512 resolution) into an array of size $(512, 512, 4)$ for each sample. Next, each of the $(512, 512, 4)$ array was standardized with respect to the training set. Due to computation and resource constraints, we further resized the data to $(64, 64, 4)$ for our baseline models, and to $(256, 256, 4)$ for our transfer learning model. Moreover, we performed data augmentation on the training set such as random rotation, lighting and flipping to reduce overfitting.

B. Baseline models

The multi-class multi-label classification problem was decomposed into binary classification problems. For each of the baseline model, we trained 28 one-vs-all binary classifiers. The baseline models were implemented using scikit-learn packages [10].

1) Logistic regression with L2 regularization: We fit the logistic regression (LR) with L2 regularization (1) model to each class as the first baseline.

$$\underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{m} log(p(y^{(i)}|x^{(i)}, \theta)) - \lambda\theta^T\theta \tag{1}$$

2) Random forest: Our second baseline model is random forest with Gini loss. We performed grid search over the following hyperparameters: the number of estimators (i.e. trees) and the minimum leaf size.

C. Transfer learning of ResNet50 with pre-trained weights

Next, we explored the effectiveness of transfer learning using the deep neural network ResNet50 [11] pre-trained on ImageNet dataset. The transfer learning model was implemented using fastai library [12] [13] and PyTorch [14].

1) ResNet50: ResNet is a deep neural network architecture which uses residual learning to address the degradation problem faced by deep neural networks. ResNet uses shortcut connections between input of one layer and a later layer through identity mapping. ResNet50 is one of the several variants of ResNet with 50 layers.

2) Loss functions: We explored two different loss functions. The first one is the binary cross-entropy (BCE) loss, a standard loss function widely used for multi-label classification problems. Similar to the baseline models, the problem was decomposed into training 28 binary classifiers for each class. The BCE loss is:

$$L_{BCE}(p, y) = -(y \log(p) + (1 - y) \log(1 - p)) \tag{2}$$

where $y \in \{0, 1\}$ is the true label, and $p \in [0, 1]$ is the predicted probability. Equivalently, it can also be written as:

$$L_{BCE}(p_t) = -\log(p_t) \tag{3}$$

where $p_t = p$ if $y = 1$, and $p_t = (1 - p)$ if $y = 0$.

In addition, we explored the focal loss function with an attempt to overcome the class-imbalance issue in our dataset. The focal loss [15] was originally proposed for dense object detection with extreme imbalance between the foreground and background classes (e.g. 1:1000 ratio). It extends BCE loss by including an extra modulating factor $(1 - p_t)^\gamma$:

$$L_{FL}(p_t) = -(1 - p_t)^\gamma \log(p_t) \tag{4}$$

where $\gamma \geq 0$ is the focusing parameter. Informally, if an example is "easy", it would have a large p_t and a small modulating factor $(1 - p_t)^\gamma$. In contrast, if an example is "hard", it would have a small p_t and a larger $(1 - p_t)^\gamma$ compared to the "easy" examples. Hence, in the scenario when the positive examples are much fewer than the negative examples, the modulating factor reduces the loss contribution of misclassified negative examples, and thus encourages the model to focus more on the misclassified positive examples. As shown in equation (4), focal loss is equivalent to BCE loss when $\gamma = 0$.

3) Initializing network weights: The first layer of ResNet50 is a 2D convolutional layer with 3 input channels. However, since our input data consists of 4 color channels, we modified the first layer into 4 input channels. To fully utilize the effectiveness of transfer learning, we initialized the weights of RGB channels with the pre-trained weights, and assigned the pre-trained weights of R channel to the Y channel.

4) Output activation function: We used sigmoid as the activation function at the output of the network. The choice of sigmoid over softmax in the multi-class multi-label problem is because sigmoid assumes independence between classes, while softmax tends to predict only one label.

5) Training of the network: In a convolutional neural network trained for image classification, early layers usually capture more generic features, whereas later layers learn dataset-specific features [6]. Since ResNet50 has been trained on ImageNet, the pre-trained weights in the early layers likely contain generic information that could be useful for our data. In contrast, the last layer needs to be trained more extensively to learn features specific to our dataset. Therefore, we first freezed all the layers except for last one, and trained the network for one epoch. We then unfreezed all the layers and continued training until overfitting.

Furthermore, because the later layers of the network need more training than the early layers, we used differential learning rates during training. For example, if the learning rate for the last fully connected layer was 0.01, we used 0.01/3 for the middle layers, and 0.01/10 for the early layers. The exact assignment of learning rates to the layers was automatically handled by the fastai library.

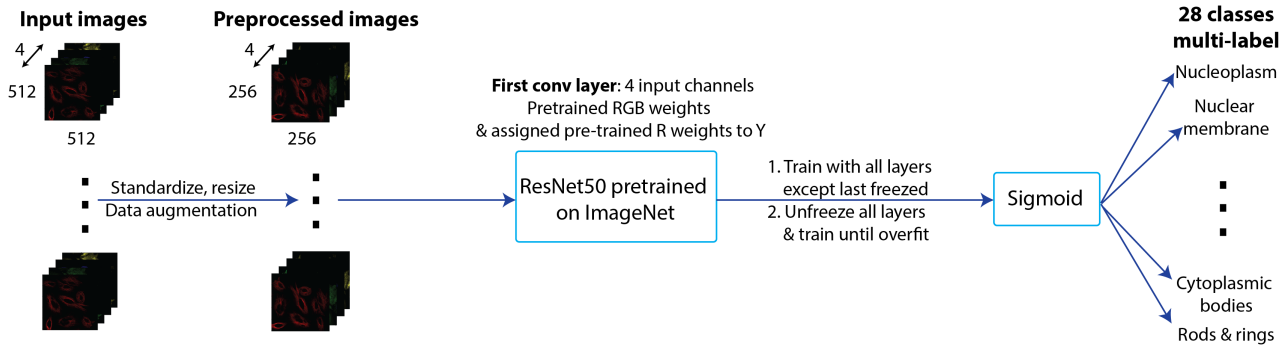


Fig. 2: Overview of preprocessing and transfer learning model

To find the optimal learning rate, we used the lr_find method provided in the fastai library. It starts the learning rate from a small value and increases gradually until the loss no longer decreases. Fig. 3 shows a plot of loss against the learning rate. We picked 0.02 as the optimal learning rate in this case because the loss around 0.02 is relatively low but is still clearly decreasing.

Lastly, we used 50% dropout rate as regularization method to reduce overfitting, and the Adam optimizer to train the network. We set the mini batch size to be 64 to balance between efficiency and quality of training.

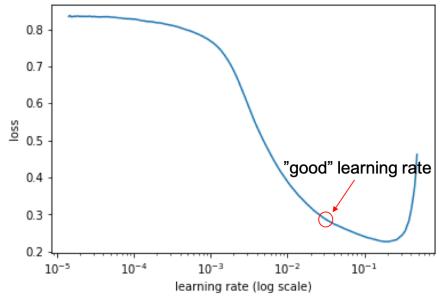


Fig. 3: Training loss vs learning rates during optimal learning rate search.

6) Stochastic gradient descent with restarts: To improve the performance of the neural network, we further explored a technique called stochastic gradient descent with restarts (SGDR) [16] [17]. SGDR gradually decreases the learning rate during one training cycle (e.g. one epoch), and then restarts the learning rate at the beginning of the next training cycle. Intuitively, SGDR encourages the model to converge to a "smooth" local minimum rather than a "spiky" one.

SGDR impelementation is included in the fastai library. Fig. 4 shows a plot of learning rate versus iterations during training using focal loss ($\gamma = 0.5$). In this example, the number of restarts was set as 3. The learning rate followed cosine annealing, and the number of iterations was multiplied by 2 during each restart cycle.

7) Data augmentation at inference: To obtain predictions on validation and test sets, we applied test time augmentation, another useful technique provided by the fastai library.

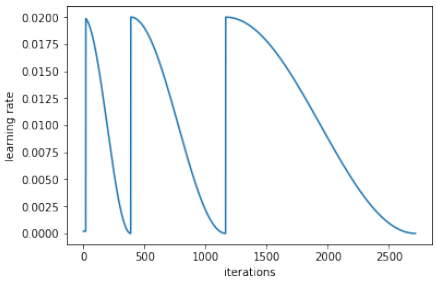


Fig. 4: Learning rate vs number of iterations during training with SGDR.

For each validation/test image, the model makes predictions on the original image as well as several randomly augmented versions of it. The final prediction score of this image is then the average prediction from all the versions.

IV. Results and Discussion

In this section, we report the results of the models using the following metrics: macro and micro averages of precision, recall, F1-score, ROC-AUC, PR-AUC as well as Hamming loss. The main difference between the macro and micro average of an evaluation metric is as follows: macro average first calculates the corresponding metric for each of the one-vs-all classifier, and then takes an arithmetic average over the 28 classes; whereas micro average flattens the prediction results into one dimension and then computes the metrics. Hence, macro averaged metrics treat all classes equally. We focused on macro instead of micro averaged metrics to take into account of the imbalanced classes. We did not use accuracy as a metric because it could be misleading in the case of extreme class imbalance. According to the Kaggle challenge evaluation rule, we tuned the hyperparameters of our baseline models mainly based on the macro F1-scores on the validation set.

A. Baseline

1) Logistic regression with L2 regularization: The first baseline model is logistic regression with L2 regularization. Table I shows the evaluation metrics on validation set with

different regularization strengths (C is the inverse of the regularization strength λ).

The slightly better model found through grid search is the one with strongest regularization ($C = 0.1$), which achieved macro F1-score of 0.11 on validation set. Table V (column "LR") shows the evaluation metrics on test set specific to the model with $C = 0.1$. As shown in the table, logistic regression did not handle the rare occurring classes, especially classes 8, 9, 10, 15, 20, 26, and 27.

2) Random forest: Table II shows the evaluation metrics of RF models with different hyperparameters. We chose the final baseline model with the highest macro F1-score (i.e. 10 trees, 1 minimum leaf size). The evaluation metrics of each individual class on test set are shown in Table V (column "RF").

The poor performance of the RF model is likely because RF is prone to the curse of learning from imbalanced data [18]. For example, similar to logistic regression, the RF baseline model completely failed to classify rare occurring classes 8, 9, 10, 15, 20, 26, and 27. Compared to logistic regression, RF baseline model also failed to predict classes 1, 6, 16, 17, 22, 24.

B. ResNet50 with pre-trained weights

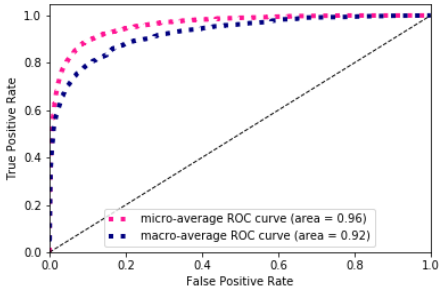
The number of epochs to train each model was determined based on the training and validation losses. We stopped training when the loss was not decreasing anymore. On average, we trained 8 epochs for each model. For example, the training and validation loss of the model using BCE loss and SGDR was 0.100 and 0.092 respectively, which shows that the model had relatively low bias and variance.

1) Focal loss vs binary cross-entropy loss: We experimented with BCE loss and focal loss with different values of the focusing parameter γ. We chose the optimal γ for further analysis based on results on validation set. Table III shows the evaluation metrics of ResNet50 trained with BCE loss and focal loss without SGDR.

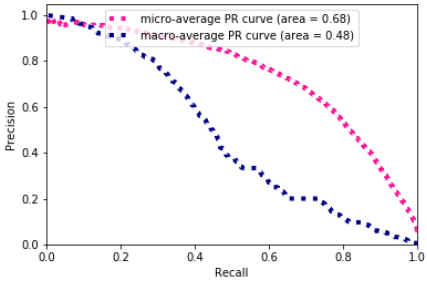
The value of γ achieving the best performance is 0.2. The effect of a larger γ on model performance is not monotonic. Instead we observed two phases: in the first phase as we included a positive γ into the loss function, all metrics have improved compared to BCE loss. This is because the modulating factor in focal loss (equation 4) scales down the loss contribution of the "easier" negative examples. As γ increased further, the performance on the majority classes was compromised more than the gain in correctly classifying minority classes, and thus the overall macro averaged metrics became worse.

2) Training with SGDR: Table IV shows significant improvement in performance for all models using the same loss functions in Table III when trained with SGDR. The model achieving the highest macro F1-score after including SGDR is the one using BCE loss.

3) Prediction on test set: Based on the performance on validation set, the best ResNet50 model is the one trained with BCE loss and SGDR. Therefore, we used this model for prediction on test set. The ROC and PR curves are shown in Fig. 5. Furthermore, a comparison between the best baseline models and best ResNet50 model is shown in Table V. Here, we break down the results into individual classes to better understand how the models perform on predicting each class. All three models failed to predict classes 8, 9, 10, 15, 20, 26 and 27. Random forest had the worst performance. ResNet50 with BCE loss and SGDR achieved significantly better performance on all classes, except for class 16 and 17 as compared to logistic regression.



(a) Macro & micro averaged ROC curves and ROC-AUC values.



(b) Macro & micro average PR curves and PR-AUC values

Fig. 5: Performance of ResNet50 with BCE loss and SGDR on test set.

V. Conclusions and Future Work

In this paper, we explored the effectiveness of transfer learning of deep neural network. We showed that the ResNet pre-trained on ImageNet data has learned transferable features which are applicable to the HPA dataset. Our transfer learning model outperformed traditional machine learning models such as logistic regression and random forest. Using focal loss without SGDR achieved better macro averaged performance than BCE loss without SGDR. When trained with SGDR, focal loss and BCE loss had comparable performance.

There are several limitations in our current work. Because of limited computation power, the resolutions used in the baseline models and ResNet50 were different. Also, we did not experiment with the highest resolution (512×512) possible. Due to the lack of domain knowledge, we treated the task as pure image classification problem neglecting the biological properties of the images and labels.

Going forward, we could train all the models on the full resolution images. Furthermore, we could try techniques such as assigning different weights to classes in the loss function, and oversampling the minority class or undersampling the majority class.

$C = \frac{1}{\lambda}$	Precision(macro)	Recall(macro)	F1(macro)	ROC AUC(macro)	Precision(micro)	Recall(micro)	F1(micro)	ROC AUC(micro)	Hamming Loss
0.1	0.196783	0.100380	**0.115414**	0.606289	0.290527	0.253902	0.270983	0.714256	0.079475
0.5	0.138915	0.099498	0.107770	0.600095	0.275891	0.259965	0.267691	0.672880	0.083268
1.0	0.137816	0.101417	0.108642	0.599700	0.274349	0.262714	0.268405	0.660952	0.083843
10.0	0.133889	0.105683	0.110265	0.598476	0.267938	0.267622	0.267780	0.634119	0.085682

TABLE I: Results of logistic regression with different regularization strengths on validation set

	Precision(macro)	Recall(macro)	F1(macro)	ROC AUC(macro)	Precision(micro)	Recall(micro)	F1(micro)	ROC AUC(micro)	Hamming Loss
10 trees 1 min leaf size	0.133480	0.047638	**0.059083**	0.547293	0.407675	0.212013	0.278955	0.769904	0.063762
10 trees 10 min leaf size	0.073805	0.021800	0.028044	0.589133	0.522871	0.131002	0.209512	0.822719	0.057508
50 trees 1 min leaf size	0.205334	0.030576	0.042665	0.602276	0.552498	0.150761	0.236883	0.823628	0.056508
50 trees 10 min leaf size	0.072214	0.017379	0.024656	0.644569	0.594895	0.105908	0.179805	0.846277	0.056209
100 trees 1 min leaf size	0.234987	0.026651	0.037683	0.609325	0.580298	0.138510	0.223640	0.830479	0.055945
100 trees 10 min leaf size	0.103465	0.016111	0.023502	0.660502	0.608108	0.097807	0.168511	0.849607	0.056152

TABLE II: Results of RF models with different hyperparameters on validation set

	Precision(macro)	Recall(macro)	F1(macro)	ROC AUC(macro)	Precision(micro)	Recall(micro)	F1(micro)	ROC AUC(micro)	Hamming Loss
BCE	0.301531	0.211996	0.218953	0.868241	0.617945	0.553843	0.584141	0.931038	0.045876
FL $\gamma = 0.1$	0.334669	0.258064	0.269699	0.867841	0.604661	0.574195	0.589034	0.933482	0.046611
FL $\gamma = 0.2$	0.326235	0.299990	**0.303533**	0.885463	0.617635	0.600672	0.609035	0.940447	0.044864
FL $\gamma = 0.3$	0.399486	0.262284	0.268867	0.875191	0.608928	0.609168	0.609048	0.932925	0.045496
FL $\gamma = 0.4$	0.379317	0.291222	0.276937	0.874438	0.590072	0.601265	0.595616	0.933884	0.047496
FL $\gamma = 0.5$	0.371751	0.281414	0.282601	0.860760	0.566208	0.618455	0.591180	0.933288	0.049761
FL $\gamma = 1$	0.304092	0.272949	0.274623	0.858550	0.531943	0.564315	0.547651	0.926373	0.054232
FL $\gamma = 1.5$	0.293522	0.263012	0.267929	0.862434	0.560885	0.566094	0.563477	0.926980	0.051025
FL $\gamma = 2$	0.265351	0.195093	0.191348	0.857147	0.568858	0.449713	0.502317	0.915321	0.051841

TABLE III: Results of BCE loss and focal loss (FL) with different γ on validation set (without SGDR)

	Precision(macro)	Recall(macro)	F1(macro)	ROC AUC(macro)	Precision(micro)	Recall(micro)	F1(micro)	ROC AUC(micro)	Hamming Loss
BCE	0.459872	0.337152	**0.361307**	0.916026	0.700396	0.663308	0.681348	0.960546	0.036094
FL $\gamma = 0.1$	0.382480	0.333503	0.347777	0.895704	0.661797	0.662320	0.662058	0.953527	0.039335
FL $\gamma = 0.2$	0.466107	0.314088	0.340966	0.900081	0.704639	0.645327	0.673680	0.956530	0.036369
FL $\gamma = 0.3$	0.432044	0.324924	0.349694	0.900778	0.700595	0.651452	0.675131	0.956821	0.036473
FL $\gamma = 0.4$	0.416219	0.334645	0.356531	0.914285	0.694940	0.662122	0.678134	0.957906	0.036565
FL $\gamma = 0.5$	0.420466	0.316969	0.345315	0.913604	0.709000	0.647500	0.676856	0.957491	0.035967
FL $\gamma = 1$	0.406662	0.312248	0.334786	0.904909	0.677300	0.650267	0.663508	0.953320	0.038370
FL $\gamma = 1.5$	0.401884	0.285879	0.315396	0.904638	0.683564	0.615491	0.647744	0.950590	0.038944
FL $\gamma = 2$	0.411138	0.301696	0.329729	0.904020	0.681799	0.629125	0.654403	0.954059	0.038657

TABLE IV: Results of BCE loss and focal loss (FL) with different γ on validation set (with SGDR)

Class	Precision			Recall			F1-score			Number of Examples		
	LR	RF	BCE with SGDR	LR	RF	BCE with SGDR	LR	RF	BCE with SGDR	Test	Val	Train
0	0.44	0.48	0.77	0.46	0.53	0.92	0.45	0.5	**0.84**	1267	1265	10353
1	0.18	0	0.81	0.07	0	0.74	0.1	0	**0.78**	128	145	981
2	0.13	0.16	0.66	0.16	0.05	0.76	0.14	0.08	**0.71**	329	324	2968
3	0.12	0.15	0.71	0.1	0.02	0.4	0.11	0.03	**0.51**	161	148	1252
4	0.2	0.09	0.72	0.11	0.02	0.7	0.14	0.03	**0.71**	186	203	1469
5	0.15	0.02	0.67	0.15	0	0.53	0.15	0.01	**0.59**	249	251	2013
6	0.07	0	0.72	0.03	0	0.33	0.04	0	**0.45**	101	103	804
7	0.17	0.09	0.74	0.12	0.02	0.74	0.14	0.03	**0.74**	310	284	2228
8	0	0	0	0	0	0	0	0	0	3	4	46
9	0	0	0	0	0	0	0	0	0	4	3	38
10	0	0	0	0	0	0	0	0	0	2	3	23
11	0.11	0.17	0.77	0.06	0.02	0.54	0.07	0.04	**0.64**	124	101	868
12	0.05	0	0.81	0.01	0	0.19	0.02	0	**0.31**	69	70	549
13	0.29	0.33	0.48	0.15	0.02	0.29	0.19	0.04	**0.36**	48	50	439
14	0.09	0.04	0.93	0.04	0.01	0.77	0.06	0.02	**0.84**	96	102	868
15	0	0	0	0	0	0	0	0	0	3	1	17
16	0.25	0	0	0.03	0	0	**0.06**	0	0	61	52	417
17	0.4	0	0	0.07	0	0	**0.12**	0	0	27	20	163
18	0.28	0.14	0.22	0.14	0.01	0.02	**0.18**	0.02	0.04	96	85	721
19	0.17	0.19	0.68	0.1	0.03	0.33	**0.13**	0.06	0.44	153	137	1192
20	0	0	0	0	0	0	0	0	0	20	22	130
21	0.29	0.33	0.67	0.31	0.12	0.59	0.3	0.17	**0.62**	370	394	3013
22	0.06	0	0.55	0.02	0	0.07	0.04	0	**0.13**	81	80	641
23	0.11	0.13	0.69	0.11	0.03	0.76	0.11	0.05	**0.73**	295	301	2369
24	0.25	0	1	0.03	0	0.03	0.05	0	0.05	39	33	250
25	0.37	0.51	0.65	0.38	0.38	0.71	0.38	0.44	**0.68**	841	856	6531
26	0	0	0	0	0	0	0	0	0	29	23	276
27	0	0	0	0	0	0	0	0	0	1	1	9

TABLE V: Prediction results on individual classes on test set. Models shown here are: best logistic regression (LR) model, best random forest (RF) model, and ResNet50 with BCE loss and SGDR.

CONTRIBUTIONS BY TEAM MEMBERS

Logistics: Ruge set up Google cloud instance and cloud disk storage, downloaded dataset, and installed Jupyter-notebook etc.

Preprocessing: Siyi preprocessed the raw data, and saved the large size data file as hdf5 files to fit into memory and storage constraints.

Baseline models: Ruge implemented the logistic regression baseline models. Siyi implemented the random forest baseline models.

Model design and experiments: Siyi implemented the transfer learning model. Ruge and Siyi set up and executed the experiments. Ruge consolidated the results.

Poster and final report: Ruge and Siyi wrote the report, designed the poster and presented together.

REFERENCES

[1] D. P. Sullivan, C. F. Winsnes, L. Åkesson, M. Hjelmare, M. Wiking, R. Schutten, L. Campbell, H. Leifsson, S. Rhodes, A. Nordgren, K. Smith, B. Revaz, B. Finnbogason, A. Szantner, and E. Lundberg, "Deep learning is combined with massive-scale citizen science to improve large-scale image classification," *Nature Biotechnology*, vol. 36, p. 820, Aug 2018.

[2] O. Z. Kraus, J. L. Ba, and B. J. Frey, "Classifying and segmenting microscopy images with deep multiple instance learning," *Bioinformatics*, vol. 32, no. 12, pp. i52–i59, 2016.

[3] T. Pärnamaa and L. Parts, "Accurate classification of protein subcellular localization from high-throughput microscopy images using deep learning," *G3: Genes, Genomes, Genetics*, vol. 7, no. 5, pp. 1385–1392, 2017.

[4] M. Boland and R. Murphy, "A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of hela cells," *Bioinformatics (Oxford, England)*, vol. 17, pp. 1213–23, 01 2002.

[5] K. Huang and R. F. Murphy, "Boosting accuracy of automated classification of fluorescence microscope images for location proteomics," *Bmc Bioinformatics*, vol. 5, no. 1, p. 78, 2004.

[6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[7] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.

[8] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[9] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *ICANN*, 2018.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] J. Howard *et al.*, "fastai," https://github.com/fastai/fastai, 2018.

[13] lafoss, "Kaggle kernel," https://www.kaggle.com/iafoss/pretrained-resnet34-with-rgby-0-460-public-lb, 2018.

[14] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[15] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2017, pp. 2999–3007.

[16] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983v5*, 2016.

[17] L. N. Smith, "Cyclical learning rates for training neural networks," *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472, 2017.

[18] C. Chen and L. Breiman, "Using random forest to learn imbalanced data," *University of California, Berkeley*, Jan 2004.