Utrecht University

Department of Information and Computing Sciences

Computing Science Master Thesis

Morphological Cell Classification under Weak Supervision: A Learning from Label Proportions Approach

---

Author:
Tingyang Jiao

Supervisors:
Prof. Wilson dos Santos Silva
Prof. Ad Feelders

November 2024

Abstract

Classification of cells is an important field for biology and pathology research, and there have been many effective models integrating it with machine learning techniques. In the supervised learning scenario, annotations on cells are crucial for cell classification, and they can be obtained by cell image analysis and biological staining. However, due to restriction on cost, time efficiency and limitations of certain microscopes, obtaining annotations of all individual cells for the training of fully supervised machine learning models is always more challenging. For this reason, it's highly necessary to investigate the feasibility of applying machine learning model on classification of cells with their morphological features, but only proportion labels for groups of cells are known. The models developed and tested in this thesis are from Learning from Label Proportions (LLP), a sub-domain of weakly supervised learning. LLP models approach data in bags instead of instances, aiming at prediction on instance level labels but assuming only proportions of each class in each bag are known. We developed several LLP based machine learning models specifically for tabular data containing morphological features of cells. We tested and compared these models, examining technique details and bringing future research directions. Our findings indicate that LLP models can achieve competitive performance on morphological cell classification with only proportion labels known, bringing values on further research in biology by reducing the need for comprehensive cell labeling.

Keywords: Machine Learning, Weakly Supervised Learning, Learning from Label Proportions, Neural Networks, Morphological Cell Classification

Contents

Chapter 1

Introduction and Research Question

The intersection of machine learning and biology has emerged as a popular interdisciplinary field, with many successful applications and encouraging advancements. One particular domain is in the research of immunotherapy. With the combination of EX Vivo 3D Tumor Testing[1] and High-Content Screening (HCS)[2] tools, researchers are able to capture images of tumor cells and extract their corresponding morphological features, such as size, shape and Zernike Vectors. A diverse range of machine learning techniques, from traditional models such as Support Vector Machines to Deep Neural Networks, have been applied on the analysis of these morphological features and further classification on cell types and status, benefiting biological researchers on further understanding of tumor cell behaviours and the drug development.

Despite the availability of sufficient morphological features of cells, it's always more challenging to have detailed annotations of each cell in the sample. Sufficient annotations on samples are crucial for supervised learning models in classification. However, in many cases, it is more practical to annotate clusters of cells instead of individual cells, due to the restriction of microscopy or to minimize cost. Even though, precise classification on single cells are still required to further understanding the characteristics of samples. To address this issue, we define main research questions of this project as following:

Research Questions

Is it possible to predict labels of individual cells from their morphological features, using only proportions labels of groups of cells during training of a machine learning model? How do such models compare with fully supervised learning models with instance level cell labels known during training?

The motivation of this project was originally from a morphological cells dataset from a company with only cluster level labels known but requirements on instance level classifi-

cation. However, because there is no available labels of individual cells as the test set for classification, another public dataset is necessary for the evaluation of models in this thesis. Detailed introduction of dataset will be included in chapter 4. Moreover, for models meeting requirements above, we also investigate the performance of them in various conditions, such as different hyperparameter settings, distribution of proportion labels, and potential performance improvement when combining some of these models together. Detailed experiment result will be discussed in chapter 5.

In this thesis, chapter 2 provides background information about how morphological features and annotations of cells are generated, which is crucial for further understanding the dataset in this project and the research purpose. In chapter 2 we also review several competitive machine learning techniques applied in morphological cellular analysis, and more importantly the sub-domain of machine learning specifically related with our research question: Learning From Label Proportions (LLP). Chapter 3 is about technique details of several machine learning models directly utilized in our project. In chapter 4 we introduce the dataset and how experiments are designed to evaluate the models. In chapter 5 we discuss the overall performance of our models and how their performance varies under different conditions, and possibilities to further improve them. To conclude, we will discuss how the research result address our research questions and potential future works.

Chapter 2

Background and Related Work

Considering this is an interdisciplinary research, contents discussed in this chapter will include the biological background to better explain the source of research question, as well as related machine learning techniques. We start from biological background about how cellular objects are extracted and analyzed, as well as how researchers make annotations on them by staining. We then conclude the application of machine learning techniques on morphological cellular analysis and classification. We then introduced some known methods in LLP (learning from label proportions) which related to our research questions that only proportion labels are known in classification.

2.1 Biological Background - Ex Vivo 3D Micro Tumor Testing and Image based Analysis

Immunotherapy has been recently proved as a powerful technique with modality for the therapy of cancer. The progress on immunotherapy research does not only help patients in clinic, but also benefit the development of new anti-cancer drugs[3]. In the research of immunotherapy, study on the tumor microenvironment (TME) is the key part to understand important functions behind tumor and test if patient can respond to a specific treatment. In this case, we need a model which can simulate the TME, and the 3D cell culture models are proved to be more representative than 2D models[4]. One category of models is called "ex vivo", which means tissues are directly extracted from a living organism and tested outside of human body, with advantage in rebuilding the level of cellular complexity[1].

For the analysis on these 3D cellular objects, researchers use High-Content Screening (HCS) as a tool to collect massive valuable data from cells. High-Content Screening, involving both high-content imaging (HCI) and high-content analytical (HCA) tools, is able to perform imaging, segmentation and visualization of cellular objects and evaluate their interaction

with treatments, therefore bringing benifits on further research [2]. One straightforward example of techniques discussed in this section is illustrated as a workflow in figure 2.1, where the tissue processing part is corresponding to ex-vivo tumor testing discussed above, and the 3D imaging and analysis are corresponding to High-Content Screening. The pipeline discussed in this section is considered as a typical example of how morphological features in cellular dataset are generated.

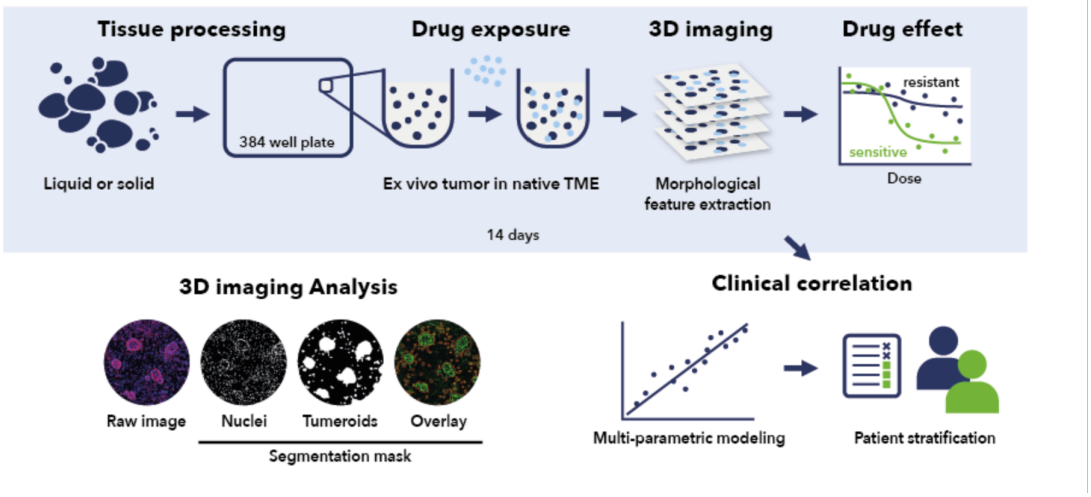


Figure 2.1: Overview of 3D Micro Tumor Testing and Image Analysis, Feature Extraction

2.2 Biological Background - Biological Staining

Staining is a widely applied technique in biology which is designed to get the amount, component as well as other information about the target tissue [5]. Some related researches also proved that study on computational staining of pathology images can benefit the analysis on tumor microenvironment (TME), by serving as labels for researchers to discover the complex mechanisms within tumor microenvironment [6]. One example about the application of pathology staining is EpCAM (epithelial cell adhesion molecule; CD326): in previous research, some specific types of cancer, such as gastrointestinal carcinoids, had strong EpCAM expression represented as high staining intensity [7]. Such a correlation makes EpCAM pathology staining results potential as proportion labels for the research on cell samples by identifying the group of cells positive to bio-markers. The staining, together with the cellular image analysis discussed in the previous section, describes how researchers can extract morphological features from cells and make proportion annotations.

2.3 Morphological Analysis and Imaging Cell Classification

The types and status of cells can be characterized by their morphological features. In biological research, morphological features can range from basic size and shape measurements such as area and diameter, to more complex properties like the Zernike vector. Several studies have shown the sufficiency and effectiveness of using morphological features in cell classification. In this section, we introduce several applications of machine learning algorithms in morphological cell analysis and classification, from both supervised and unsupervised learning perspective.

2.3.1 Supervised Cell Classification with Morphological Features

With annotations of individual cells known, the morphological classification can become a relatively straightforward process. Alizadeh et al [8] tested Linear Support Vector Machine (SVM) and a simple Artificial Neural Networks (ANN) with only one hidden layer for the classification among 15 different cell lines (population of cells cloned from same ancestors[9]) with their morphological features. The result demonstrates that morphological features are sufficient to classify cell types with some basic machine learning techniques. The successful classification can further benefit on the research of biological transformation, and the prediction of cells behavior. Similarly, Mousavikhamene et al [10] tested SVM with different kernels on the classification of tumor and non-tumor cells by their cytoskeletal structures and cell morphology, further validating the role of machine learning in leveraging morphological features for cell classification. Li et al [11] further tested several machine learning models for classification between healthy and apoptotic cells by morphological features. The machine learning models applied in their work include Logistic Regression, Random Forest, k-NN, Multilayer Perceptron, and Support Vector Machine. From their experiment result, the Multilayer Perceptron (MLP) outperforms other models. Similar work can also be found in [12] in cancer cells detection which also include Naive Bayes Classifier on morphological classification, and [13] on the oral squamous cell carcinoma. All previous works follow the imaging - segmentation - feature extraction - machine learning based classification pipeline. Additionally, with the advancements of Convolutional Neural Network (CNNs), researchers also applied CNN directly on images for automatic feature detection and classification [14].

2.3.2 Unsupervised Analysis on Morphological Features

Although some unsupervised learning techniques have been incorporated as part of the supervised pipeline for visualization and dimension reduction, such as Principle Component

Analysis (PCA), it's still more challenging to study the cell status by morphological features in the absence of cell annotations. A typical unsupervised pipeline from Bhaskar et al[15] for morphological cell analysis is shown in figure 2.2. Besides the preprocessing and feature extraction steps as discussed before, they applied dimension reduction (PCA, t-SNE[16]) and clustering (K-means, HDBSCAN[17]) on morphological features to split cells into groups. But the resulting groups only illustrate morphological similarity, the further inference on cell types or characteristics still requires further works of domain experts. Also, Jude M. Phillip et al[18] introduced an unsupervised learning pipeline combining PCA and k-means clustering, resulting that the percentage distribution of clusters differs between different types of cells, which can further benefit pathology research. Notably, their features are 50 points along the contours of each normalized shape, instead of regular morphological features like area or diameter. Another work worth noting is from DeepCell[19], researchers combined unsupervised image clustering with neural network classification to achieve high accuracy individual cell analysis. However, further correction from domain expert is still required after the clustering, instead of a totally end-to-end auto-classification network.

2.4 Learning from Label Proportions

In several machine learning tasks, instead of fully annotated instances, we may only have imprecise labels or labels from a different layer. More specifically, in classification tasks, sometimes we only have the percentage distribution of classes in data, but the target is to make instance level prediction. This case belongs to a specific domain of semi-supervised learning called learning from label proportions (LLP).

In the framework of LLP, data are grouped into bags, but only the proportion of each class within each bag is known, instead of labels of individual instances. For example, suppose in binary classification tasks we have a dataset of T non-overlapping bags with label Y_1, Y_2, ..., Y_T representing the proportion of positive instances in each bag, where Y_i of bag T_i with size N_i is calculated by

$$Y_i = \frac{1}{N_i} \sum_{j=1}^{N_i} y_{i,j} \tag{2.1}$$

where $y_{i,j}$ is the binary label of the j-th instance in bag i.

The purpose is to predict instance level label y_1, y_2, ..., y_n directly from their corresponding features x_1, x_2, ..., x_n, with only the proportion labels known. There have been several efficient methods proposed to address the LLP problem in recent years, and widely succeed in many domains, from bankruptcy prediction to image classification. However, there are still limited discussions about the LLP models in morphological feature analysis of cells.

One prominent machine learning method in LLP is the Proportion Support Vector Ma-

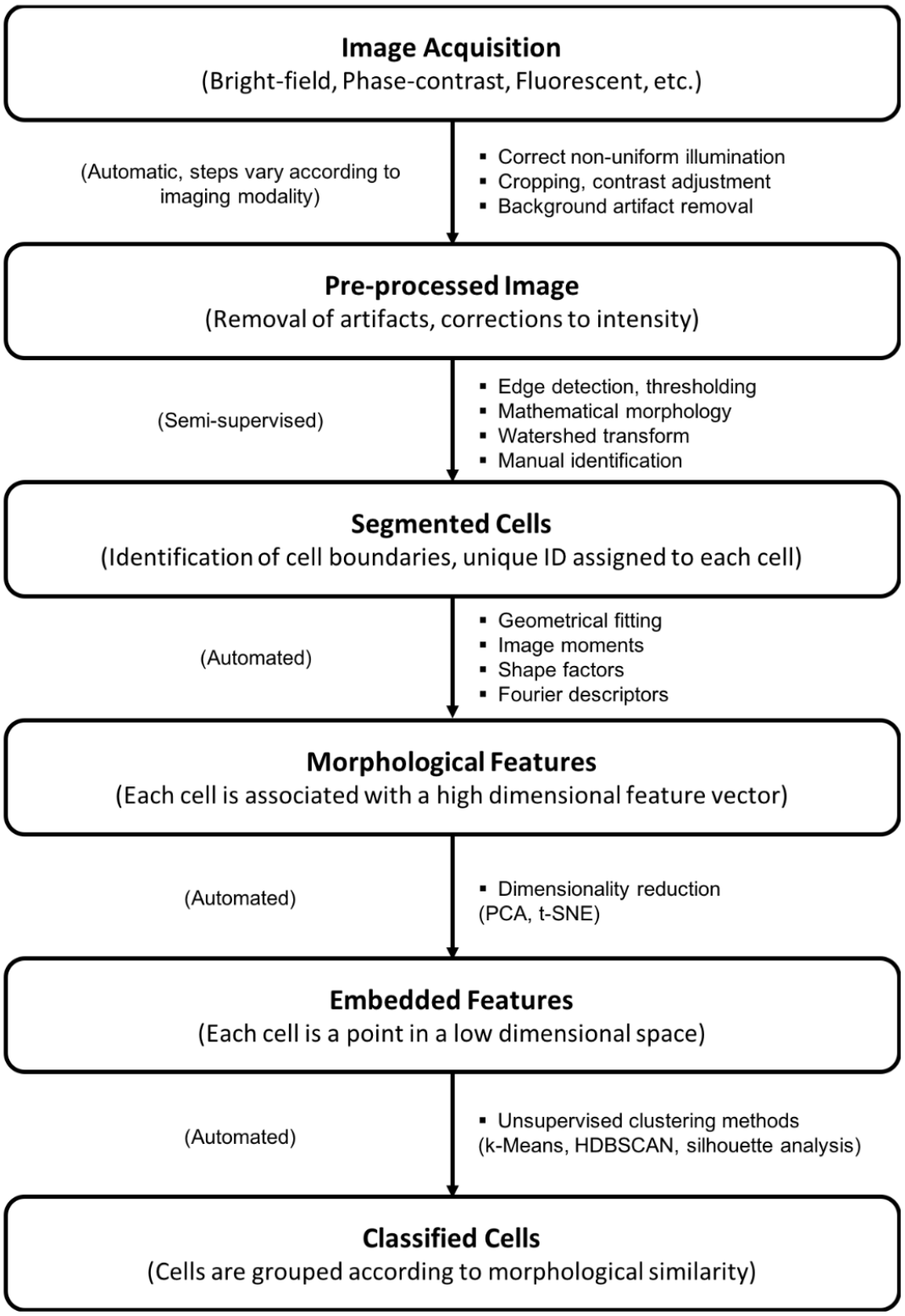


Figure 2.2: An example of unsupervised pipeline on morphological cell features, adapted from Figure 1 in [15]

chines (pSVM)[20]. SVM is a widely applied machine learning model aiming at finding the optimal hyperplane to separate two classes of data points. The pSVM extends it to both optimizing the loss in instance level and ensuring the resulted proportion fit the given proportion label. The author also proposed two methods to solve this optimization problem: an iterative updating algorithm called alter-pSVM and another convex relaxation algorithm called conv-pSVM. Details of the model and optimization process will be further discussed in the methodology chapter. Their model illustrated competitive result in many datasets. In addition to SVM, similar works can also be found in [21] with the Twin Support Vector Machine, and in [22][23] with Non-parallel Support Vector Machine. Moreover, SVM in this type of pipeline can also be replaced by Random Forests to fit the data with higher dimension[24]. Furthermore, Chen et al [25] applied ensemble learning on pSVM, which includes bagging (bootstrap aggregating) and boosting, to further increase the performance by introducing super instance of bags. Similar work can also be found in [26] with Adaboost.

With the recent advance of neural networks, there have been several high-performing deep learning models applied for label proportions. Ehsan et al[27] proposed a method to apply one more layer, called Batch Averager, after the end of neural network to compute the average proportion distribution across all instances in a batch. The core idea behind is to ensure the posterior probability distribution of instances match the bag proportion label. The method illustrated strong performance in co-training tasks of images. Yong Shi et al[28] expanded the work by allowing a small group of instances labeled explicitly in each bag together with the bag proportion to further increase the performance, indicating the potential application of combining LLP with semi-supervised learning scenario. Additionally, other neural network architectures can also be adapted to fit the structure of LLP, such as Generative Adversarial Networks(GAN)[29], Adversarial Autoencoder[30] or Self-ensemble Learning[31]. However, almost all deep learning models on LLP are designed for image or text tasks, with lack of further discussions on tabular data.

Chapter 3

Methodology

In this chapter, we provide technique details of LLP models employed in our experiment. We start from Proportion Support Vector Machine (pSVM)[20], a method derived from traditional SVM but modified to fit the LLP scenario. Besides the pSVM model itself, we will also discuss how to alternatively optimize the model in details. Next, in order to apply deep learning models for our task, we also introduce the batch averager[27]. The batch averager was originally designed for image classification with convolutional neural netowork. We integrate it with a multilayer perceptron (MLP) for tabular data. In addition, inspired by Chen et al in [25], we also proposed one ensemble method to further improve the performance of LLP models. Besides these three methods illustrated above, we also did experiments on combined pipeline integrating some of these methods. Their structures and benefits will be further discussed in the chapter of experiment and result discussion.

3.1 Proportion Support Vector Machine

The effectiveness of Support Vector Machine (SVM) have been proved in many classification tasks, including cell classification based on their morphological features[8][10]. The optimization objective formula for traditional SVM is written as:

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, N, \\ & \xi_i \geq 0, \quad i = 1, \ldots, N. \end{aligned} \tag{3.1}$$

The objective is to find an optimal hyperplane to split between different groups of data points while maximizing the margin between the hyperplane and support vectors. The hyperparameter C indicating the weight on penalty of data points on the wrong side of

margin while maximizing the margin to control the strength of the regularization. Yu et al [20] in their work expanded the equation to fit the scenarios where only proportion labels are known during training:

$$\min_{\mathbf{y},\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N} L(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i) + b) + C_p \sum_{k=1}^{K} L_p(\tilde{p}_k(\mathbf{y}), p_k) \tag{3.2}$$

$$\text{s.t.} \quad \forall_{i=1}^{N},\ y_i \in \{-1, 1\}$$

$$L(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i) - b) = \max(0, 1 - y_i \cdot (\mathbf{w}^T\varphi(\mathbf{x}_i) + b)) \tag{3.3}$$

The part $C\sum_{i=1}^{N} L(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i)+b)$ represents the loss function for instance level classification. We normally use hinge loss (equation 3.3) as L for the case of large margin framework such as SVM. And the part $C_p\sum_{k=1}^{K} L_p(\tilde{p}_k(\mathbf{y}), p_k)$ is the loss function between predicted proportion $\tilde{p}_k(\mathbf{y})$ and ground truth proportion label p_k. Absolute loss is employed for L_p, and C_p represents the weight on proportion loss. The entire updated objective function can be interpreted as finding a combination of data points within each bag to be labeled as 1 to achieve the optimized value on equation (3.2). Such an integer programming problem is NP-Hard. To address this complexity issue, Felix Yu et al[20] also proposed an alternative learning algorithm as following:

- When y is fixed, the instance level labels are decided, the entire optimization process is identical to a standard Support Vector Machine
- When there is a hyperplane decided ($\mathbf{w}$ and b fixed), the objective function becomes:

$$\min_{\mathbf{y}} \sum_{i=1}^{N} L\left(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i) + b\right) + \frac{C_p}{C}\sum_{k=1}^{K} L_p\left(\tilde{p}_k(\mathbf{y}), p_k\right) \tag{3.4}$$

$$\text{s.t.} \quad \forall_{i=1}^{N},\ y_i \in \{1, -1\}.$$

Since all bags are independent during optimization, the function above can also be written as:

$$\min_{\{y_i|i\in\mathcal{B}_k\}} \sum_{i\in\mathcal{B}_k} L\left(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i) + b\right) + \frac{C_p}{C} L_p\left(\tilde{p}_k(\mathbf{y}), p_k\right) \tag{3.5}$$

$$\text{s.t.} \quad \forall_{i\in\mathcal{B}_k},\ y_i \in \{1, -1\}, \quad \text{for } k = 1, 2, \ldots, K.$$

The second step of optimization can be further solved by the following step:

1. Initialize $y_i = -1$ for all instances in all bags

2. For each instance, calculate its contribution to the change in overall hinge loss if flipping it from -1 to 1

3. In each bag, sort the contribution to the change in hinge loss from high to low and calculate the cumulative sum. The i-th value in the cumulative sum is the change in hinge loss if we flip the first i instances from -1 to 1. The difference between it and the hinge loss where all $y_i = -1$ is the first part of objective.

4. Calculate the proportion loss if flipping the first i instances from -1 to 1 as the second part of objective

5. Find the index where weighted sum of first and second objectives is the smallest, flip all instance above this optimal index from -1 to 1

Besides the two steps optimization process above, in [20], an annealing loop is also introduced in the optimization pipeline for avoiding local minima. As illustrated in Algorithm 1, starting from a very small value, hyperparameter C is increased by a small amount in each iteration of optimization until convergence. In the context of SVM, gradually increasing C can be interpreted as adding more penalty on misclassified data points, iteratively resulting a more strict model. During the loop, the input y_i in the current iteration inherit the flipped labels from the previous iteration. Also, several different initializations were used during training to reduce the effects of randomness. The model with the lowest final objective across all random initializations is considered as the optimal solution. We also provide one abstraction of the entire process in figure 3.1, which clearly illustrates the entire process of proportion SVM, from random initialization of instance labels to picking the final optimal model.

Algorithm 1 alter-αSVM

Input: Δ and C
Randomly initialize $y_i \in \{-1, 1\}, \forall_{i=1}^{N}$
$C^* = 10^{-5}C$
while $C^* < C$ **do**
    $C^* = \min\{(1 + \Delta)C^*, C\}$
    repeat
        Fix $\mathbf{y}$ to solve $\mathbf{w}$ and b
        Fix $\mathbf{w}$ and b to solve $\mathbf{y}$ (Eq. (7) with $C \leftarrow C^*$)
    until The decrease of the objective is smaller than a threshold (10^{-4})
end while

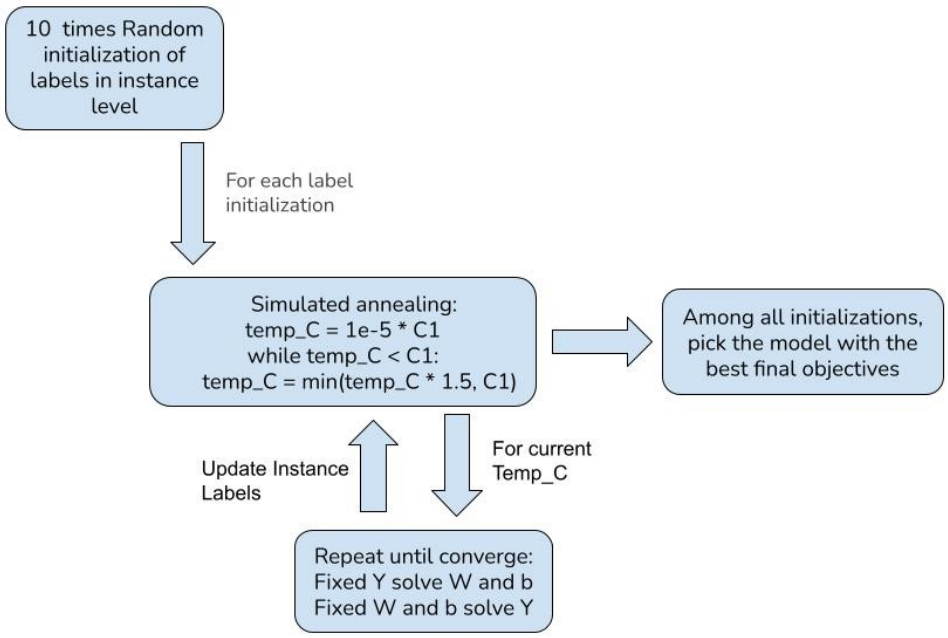


Figure 3.1: pSVM pipeline with annealing loop

3.2 Neural Network and Batch Averager

Based on the previous discussion of related works, deep learning models have been widely employed within LLP pipelines. Since our work primarily focuses on tabular data, we start from the basic structure of Artificial Neural Networks, the Multilayer Perceptrons (MLP). Typically, a MLP consists of the input and output layer with one or more fully connected layers between them. The "fully connected" means each neuron in the current layer is connected with all neurons in the next layer. One example of MLP is shown in figure 3.2. Various different types of nonlinear activation function (ReLU, Sigmoid etc) can be added after each layer for improving model's ability to capture non-linear patterns within data. Backpropagation is the core algorithm during the training of such a neural network. As illustrated in the related work part, when instance labels of cells are known, the MLP shows competitive performance in morphological classification[8][11].

When only proportion labels are known instead of instance labels, normal MLP cannot be applyed directly for solving such a problem. The Batch Averager[27] is an additional layer on neural network specifically designed for learning from proportion labels. A "batch" is a subset of the input data that is fed into the network at one time during training. Suppose for an instance $X_{i,j}$ within bag i of size T_i, in the neural network for supervised classification with k different outputs, the probabilistic output after the last activation layer is:

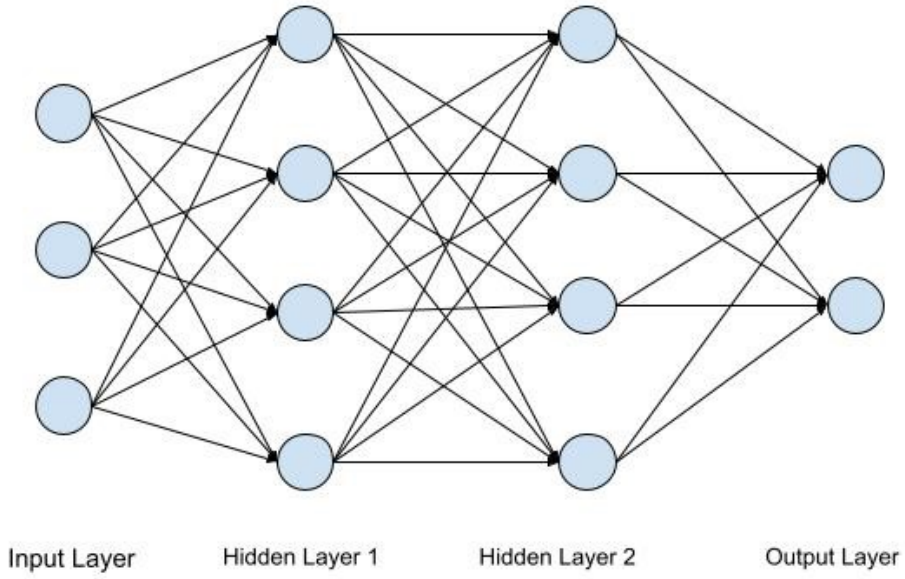


Figure 3.2: An example of MLP with 2 hidden layers

$$P(y = q \mid X_{i,j}) = y_{i,j}^{(q)} \tag{3.6}$$

The batch averager layer is designed to calculate the average probabilistic output of all instances in the entire batch, and treat it as the posterior probability of the bag:

$$\bar{y}_i^{(q)} = \frac{1}{T_i} \sum_{j \in B_i} P(y = q \mid X_{i,j}) = \frac{1}{T_i} \sum_{j \in B_i} y_{i,j}^{(q)} \tag{3.7}$$

The $\bar{y}_i$ is a probability distribution of different categories (two categories for binary classification in our case). Then, the optimization process becomes matching the posterior probability of entire bag i (output of Batch Averager, $\bar{y}_i$) with prior probability (the known proportion label of bag i, $\tilde{y}_i$). The KL-divergence is applied as the loss function for two probability distributions:

$$\Delta(\tilde{y}_i, \bar{y}_i) = \sum_q \tilde{y}_i{}^{(q)} \log \frac{\tilde{y}_i{}^{(q)}}{\bar{y}_i{}^{(q)}} \tag{3.8}$$

From the method discussed above, it is clearly observed that the batch size is required to be the same as the bag size during the training of neural network. If bag size varies in dataset, we need dynamic batch size for training. Also, due to the limited GPU memory in some cases, we need to break large bags into several subgroups as different batches to feed

the network. The requirement can be guaranteed by a specific design of batch generator in many deep learning packages. We illustrate an example of batch generator design in Keras in the algorithm 2. In the algorithm, the function LEN returns the number of total batches required for the training of neural network. The second function GETITEM is designed to generate data of each batch. In these two functions, one bag is directly considered as one batch during the training of neural network if its size is smaller than the maximum allowed batch size. On the contrary, the bag is split into different batches if its size is larger than the maximum allowed batch size. The neural network can integrate this data generator directly to extract a dynamic batch of data for the training.

Algorithm 2 Batch Data Generator

Batch_size ▷ Maximum allowed batch size
Grouped_bags ▷ All bags
Bag_ids ▷ ID of each bag
function LEN ▷ Tell the total number of batches to the network during training
    batches_amount = 0
    for each **bag_id** in **Bag_ids do**
        bag_size = **length of Grouped_bags[bag_id]**
        batches_amount += $\lfloor \frac{\textbf{bag\_size}}{\textbf{Batch\_size}} \rfloor$
        batches_amount += (1 **if bag_size**%**Batch_size** > 0 **else** 0)
    end for
    return batches_amount
end function
function GETITEM(idx) ▷ Generate the data for a specific batch
    current_idx = 0
    for each **bag_id** in **Bag_ids do** ▷ Iterate through bags
        bag = **Grouped_bags[bag_id]**
        bag_size = **length of bag**
        Batches_in_current_bag = $\lfloor \frac{\textbf{bag\_size}}{\textbf{Batch\_size}} \rfloor$
        Batches_in_current_bag += (1 **if bag_size**%**Batch_size** > 0 **else** 0)
        if current_idx + **Batches_in_current_bag** > **idx then** ▷ Reached the bag containing the target batch
            batch_start = (**idx** - **current_idx**) * **Batch_size**
            batch_end = min(**batch_start** + **Batch_size**, **bag_size**)
            batch_data = **bag[batch_start:batch_end]**
            X = **features of batch_data**
            y = **bag-level label corresponding to the current batch_data**
            return X, y
        end if
        current_idx += **Batches_in_current_bag**
    end for
end function

In addition, we can use dropout[32] to avoid overfitting during the training of neural network. The concept of dropout in neural network is straightforward: each neuron in a hidden layer has a probability p to be turned off during training. The network can be prevented from learning over-complex relationships among data to avoid overfitting. One example of MLP after applying dropout of $p = 0.5$ is shown in figure 3.3.

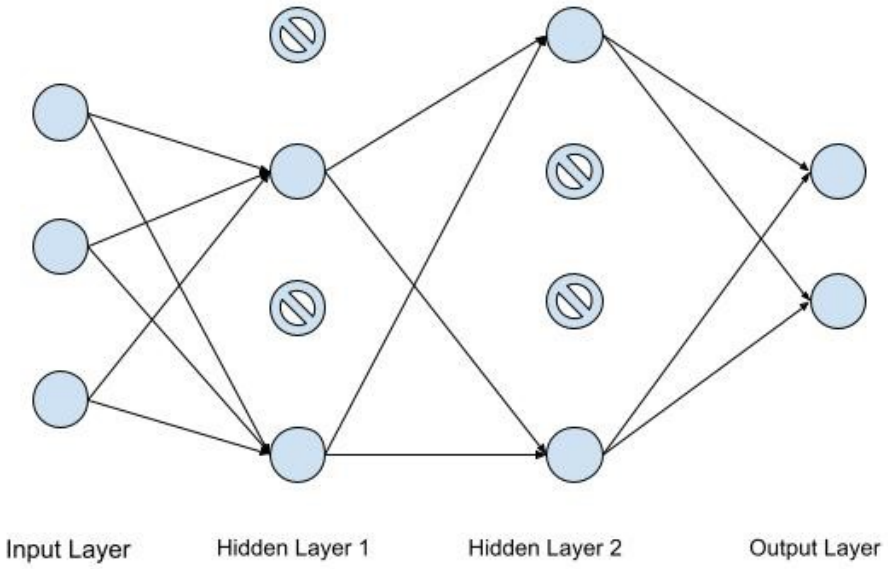


Figure 3.3: An example of MLP with 2 hidden layers and dropout p = 0.5

Combining the MLP model with batch averager, dynamic batch and dropout techniques as we discussed above for learning from label proportion in tabular dataset, one possible pipeline is shown in figure 3.4. The input is the dynamic batch generated from algorithm 2. There are four fully connected layers with more complex structure than figure 3.2 and a specific dropout rate. Besides content discussed before, there are still some important factors in this pipeline. First, the batch averager layer and KL divergence are only applied during training. When instance level prediction is required in testing, this layer should be excluded. Second, despite the sigmoid activation function 3.9 is the most common for binary classification in neural network, since it's required an proportion distribution for KL divergence, we use softmax activation function 3.10 in this pipeline.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.9}$$

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{q} e^{z_j}} \quad \text{for } i = 1, 2, \ldots, q \tag{3.10}$$

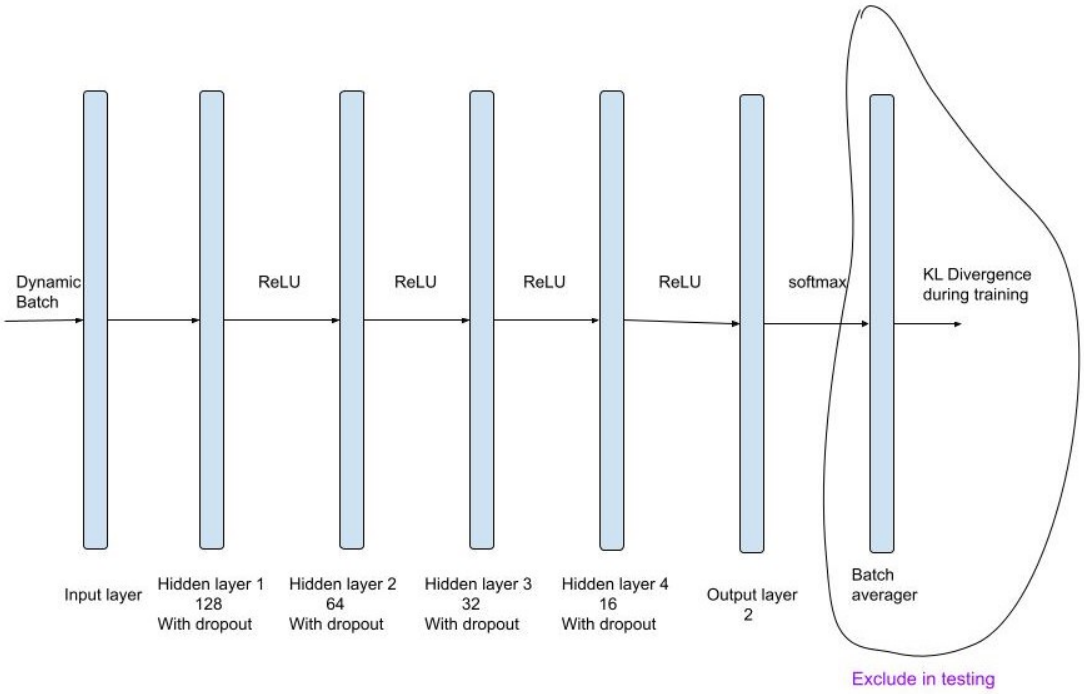


Figure 3.4: An MLP pipeline for Learning from Label Proportions

3.3 Ensemble Methods for LLP

Ensemble methods have been widely used in machine learning tasks to further improve the overall performance and stability of models. One of the most commonly used ensemble methods is bagging[33]. Bagging (bootstrap aggregating) employs M different weak classifiers where each classifier is only trained in a random sub-sample of training set with replacement. The M weak classifiers are eventually aggregated together for final classification. The bagging and other ensemble methods are most often applied with decision tree models, but application on SVM, as well as pSVM method as discussed previously, is also possible.

Inspired by the works by Chen et al in [25], we propose one bagging method for LLP scenario, as illustrated in algorithm 3. We use random sampling with replacement in bag level instead of instance level in normal supervised learning method. Also, we believe prior weight on bags during sampling is necessary. In our experiment specifically, we define the prior weight as $|Y_i - 0.5| + \epsilon$ where Y_i is the proportion label of the bag i. The reason behind it will be further discussed in following chapters.

Algorithm 3 Bagging in LLP

M ▷ Number of weak classifiers
N ▷ Number of unique bags in the training set of LLP
Weights ▷ Prior knowledge about the importance of each bag
Prob $= \frac{\text{Weights}}{\sum \text{Weights}}$ ▷ Normalized weights (prior probabilities)
Classifier set $= []$ ▷ Empty set to hold weak classifiers
for $i = 1$ to M **do**
    Randomly sample a bag from the training set N times with replacement using the prior probabilities
    Train a weak classifier on the Bootstrap sample
    Classifier set.append(weak classifier) ▷ Store the trained classifier
end for
The classifier set is then used for prediction with majority vote

Chapter 4

Dataset and Experiment Settings

In this chapter, we first introduce the alternative dataset used for testing the performance of our models due to the lack of instance level cell labels for testing purposes in the original dataset. We then discuss selected models, hyperparameters as well as the experiment design for model evaluation. We end this part with programming tools used during experiments of this project.

4.1 Dataset Description

The dataset we selected is from The "Cell Image Library" (CIL)[34]. The CIL is a comprehensive large scale cell dataset containing more than 900,000 five-channel fields of view for individual cells, captured by advanced microscopy, and more than 30k different tested compounds. All cells within this dataset are from the single cell line: the U2OS (osteosarcoma cells), which means all cells in our experiment belong to the same type. The cell line in biological research generally refers to a specific population of cells cloned from same ancestors[9], making them ideal target for testing the influences of medicine or other chemical compounds. The "five-channel fields" means five fluorescent channels to paint different cellular components. In addition to images, the dataset also includes morphological features generated from five-channel fields, providing sufficient amount of information on studying cellular states. An example of five-channel cell analysis in this dataset is shown in figure 4.1, where five fluorescent channels are illustrated separately. The pipeline of generating this dataset is considered as a typical example of cellular imaging analysis and very similar to our original dataset, which makes it an ideal dataset for testing our LLP pipelines.

There are more than 500 features for each cell in the entire dataset. Because we need to make the usage of this dataset as close to our original dataset as possible, among all five fluorescent channels, we only use morphological features related with channel "Nucleus" and "Actin, Golgi, plasma membrane", resulting in around 218 features totally. Each feature

can be considered as belonging to one of these categories:

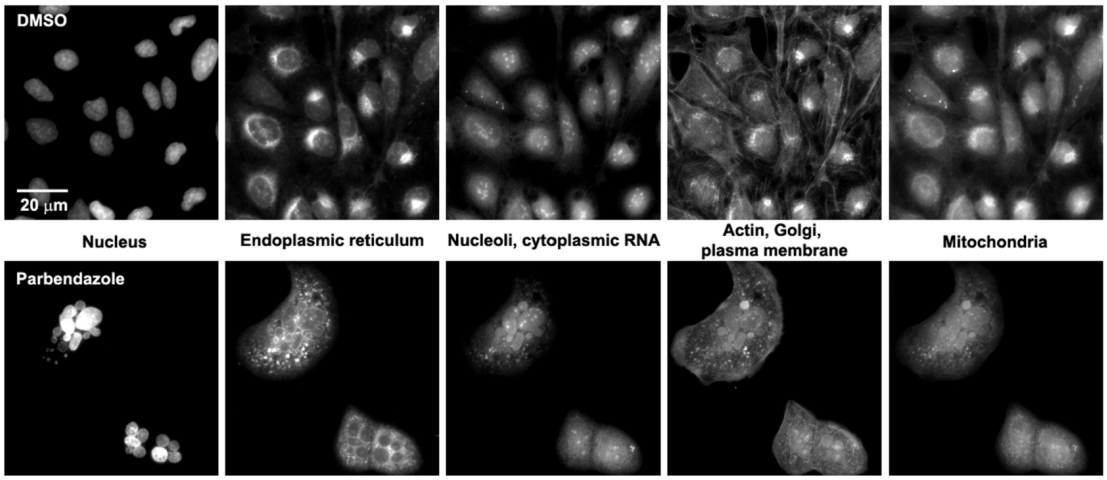
- AreaShape: Area, Eccentricity, Perimeter, Zernike Vector etc.
- Correlation
- Granularity
- Intensity
- Location
- Radial Distribution
- Texture

Within the CIL dataset, each well in a batch contains a group of individual cell objects, with around 500 - 600 cells in most of the wells. A well is either treated with a specific chemical compound or with no treatment as the negative control group. From biological perspective, for cells from the same cell line such as the U2OS in our dataset, study on how a group of cells differ morphologically from the negative control group after being treated with a specific tested compound can help the study on mechanism of this compound, further benefiting the drug development[35][34]. In our study, we select two wells from this dataset for binary classification: one well without any compound treatment and the other treated well which is significantly different from the first one in morphological features. Their difference will be proved by experiments in the upcoming sections. In our case, we consider the classification between negative control wells and wells significantly vary in morphological features after being treated with tested compound as an approximation of classification between cells with different characteristics.

4.2 Experiment Setup

4.2.1 Model Selection for LLP in Morphological Cell Classification

There are three main models tested in our experiment for performance evaluation on morphological cell classification with only proportion labels. The pSVM have been widely employed to solve many LLP problems or as part of other LLP pipelines, so we select it as the core model in our task. Considering the relatively simple nature of our data, all of the support vector machines in our experiment are with linear kernel only instead of more complex kernels such as RBF. We set $C = C_p = 1$ as the baseline to test the performance, other possible values will be further discussed in the next chapter. For the LLP model with neural network,



(a) five-channel analysis on cell images

Dye	Organelle or cellular component	CellProfiler
Hoechst 33 342	Nucleus	DNA
Concanavalin A/Alexa Fluor 488 conjugate	Endoplasmic reticulum	ER
SYTO 14 green fluorescent nucleic acid stain	Nucleoli, cytoplasmic RNA	RNA
Phalloidin/Alexa Fluor 594 conjugate, wheat germ agglutinin (WGA)/Alexa Fluor 594 conjugate	F-actin cytoskeleton, Golgi, plasma membrane	AGP
MitoTracker Deep Red	Mitochondria	Mito

(b) Fluorescent dyes and their corresponding cell components

Figure 4.1: An example of five-channel microscopy analysis of cellular image used in dataset

due to the tabular dataset we have and the relatively straightforward structure of data, we apply batch averager and dynamic batch technique with a simple multilayer perceptron as illustrated in figure 3.4 as our deep learning model. After several experiments, we choose a dropout rate of 0.2 added in each hidden layer to achieve a balance between preventing overfitting and keeping a competitive accuracy. The neural network is optimized by Adam[36], a stochastic gradient descent method, and the learning rate we choose is 0.001. We also add a new pipeline which uses the output of MLP with batch averager as the intermediate result, then uses this result as the input of pSVM for final classification. More specifically, we first have the training set and their classified labels after the training of the neural network, then these intermediate labels are further employed as the input for the training of pSVM, instead of random initializations as discussed in algorithm 1. In this case, the pSVM is only executed once without several times of random initializations on instance labels. Its advantages will be discussed in the next chapter. Furthermore, besides these three main models, as discussed in the methodology section, it's also possible to introduced ensemble methods on pSVM. We will also test it's performance in our experiments. We set the amount of weak classifier M = 125 in our experiments.

Besides all LLP models discussed above, it is also necessary to evaluate the upper bound of classification in our dataset. In our experiment, the upper bound is defined as the scenario where all instance level labels are known, making it a fully supervised learning problem. We use a Support Vector Machine with linear kernel in supervised learning scenario to assess the upper bound performance. Since LLP models can only access proportion labels during training, it is expected that the performance of LLP models is below the upper bound, but how far it is from the upper bound serves as an important metric to test its performance.

4.2.2 Cross Validation and Bag Split

For most LLP experiments in previous studies, there were no pre-defined bags in dataset, requiring researchers to manually create bags with different sizes. It is also the approach in our experiment. Most previous works tested models on bags with various sizes start from 2 until at most 64 or 128. However, considering there are typically larger number of cells in morphological classification tasks, the bag size in our experiment starts from around 150 originally, then the bag size is further increased as data augmentation is introduced. The technique of data augmentation will be discussed in the next section.

Also, to better assess the model's robustness and generalization capabilities across different splits of the data, we use 5-fold performance testing in our experiment, which is very close to the 5-fold cross validation but slightly modified for the LLP scenario. The data is first divided into 5 folds, with 4 of them are treated as training set and the remaining group as testing data in each iteration. Then only the training set is further divided into bags with

different pre-defined proportion labels since no bag level information should be provided in the testing set. The illustration of how to test the model performance is shown in figure 4.2. In addition, we also specify different distributions of proportion labels, such as [0.1, 0.3, 0.4, 0.6, 0.8] or [0.1, 0.2, 0.6, 0.7, 0.9], where each number is the proportion of positive instances in each bag. The experiment settings above can guarantee that one model's competitive performance is not restricted to only one specific train/test split or one specific split of bags, but with robustness in many different cases.

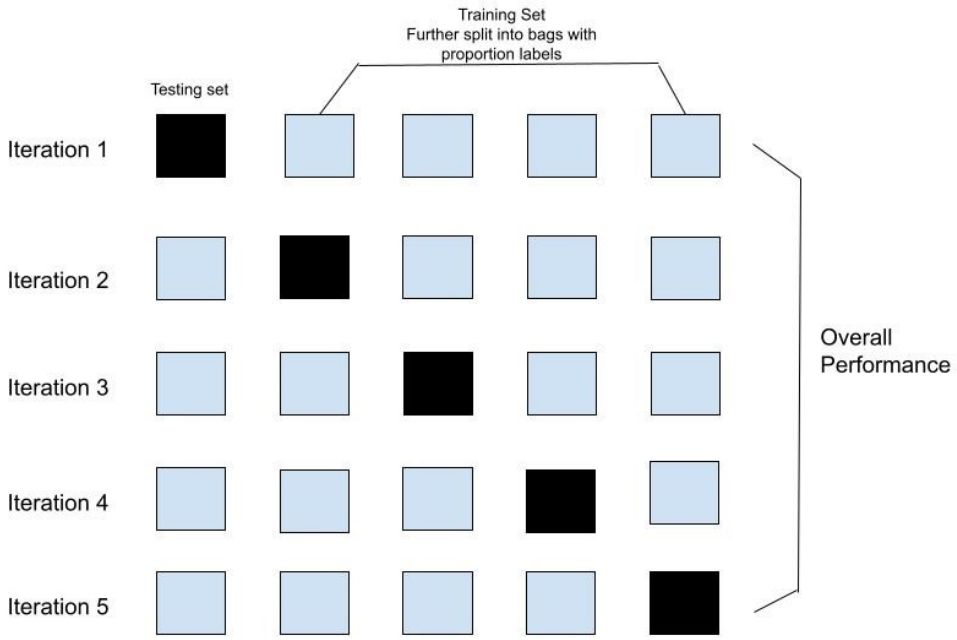


Figure 4.2: 5-fold cross testing of LLP model

4.2.3 Well Selection and Data Augmentation

As discussed in the introduction of dataset, the classification task in our experiment involves distinguishing between a well treated with a specific chemical compound and another well without treatment as negative control group, and cells in these two wells should be significantly different in morphological features. We selected wells with id "A02" and "A13" from plate 24279 in CIL dataset as our classification targets. One method to prove that the cells in two selected wells are morphologically different is by training a supervised learning model on their morphological features. For this task, a Support Vector Machine with linear kernel

results in around 94.3% accuracy in classifying cells in these two wells, proving that two groups of cells are linearly separable by morphology. As discussed before, this is also considered as the upper bound of classification in LLP scenario. In addition, we also applied one statistical test, the MANOVA (multivariate analysis of variance)[37], to check the difference between cells in these two wells. The resulting p-value is less than 0.001, which indicates that there is a significant morphological difference between cells in "A02" and "A13", and they are feasible selections for testing our LLP models.

For the data amount, there are 569 individual cells in well 'A13' and 422 cells in 'A02'. As a result, in the 5-fold testing introduced before, around 800 data points will be included in the training set in each iteration. However, it is necessary to further evaluate the model's performance in different data sizes. In this case, a data augmentation method is required. The data augmentation on tabular data can be challenging, and we choose SMOTE (synthetic minority over-sampling technique) [38], one of the most influential over-sampling methods in machine learning, as our augmentation algorithm. The basic idea behind SMOTE is straightforward: suppose within one class of data there is one data point X with nearest neighbours $X1$, $X2$, $X3$, random interpolation between $X-X1$, $X-X2$, $X-X3$ respectively are considered as 3 new sampled points $N1$, $N2$, $N3$ of this class, as illustrated in figure 4.3. The advantage of SMOTE is that it can generate new data points instead of simply over-sampling the original data, while still preserving the characteristics of each original class. When evaluating models with data augmentation, the SMOTE algorithm is only applied in the training set in each iteration of 5-fold evaluation, with the test set remaining unchanged.

4.3 Programming Tools and Libraries

For further re-implementation and evaluation on our work, we list the tools and libraries used in our project in this section. All experiments are executed in a laptop with Intel i7 11800H CPU, 32GB RAM and RTX 3060 Laptop GPU. The programming tool is python (3.11). Core libraries used during our experiment include NumPy 1.24.3, Pandas 2.2.1 for data processing, scikit-learn 1.4.1 for machine learning models and tensorflow 2.13.1 for building neural network.

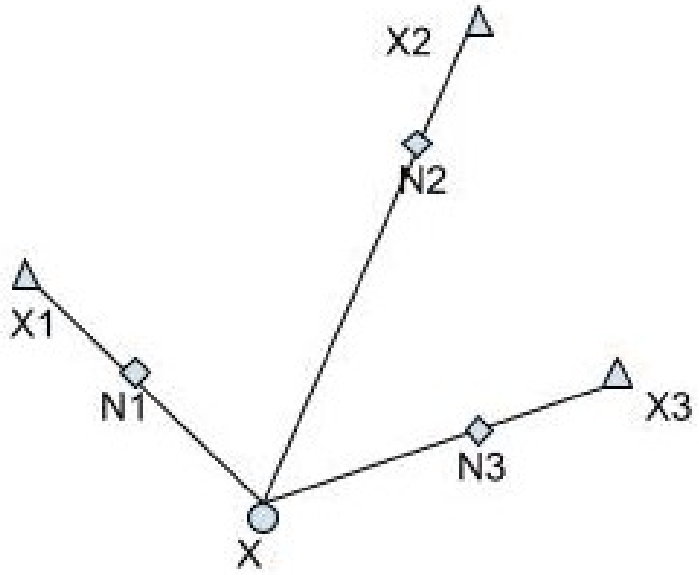


Figure 4.3: An example of SMOTE sampling in 2 dimension data points

Chapter 5

Result and Discussion

5.1 Overall Performance

In this section, we evaluate the performance of our models in various conditions, including different bag sizes, bag proportion labels, and other model configurations. As previously discussed in the experiment settings, the training set of each fold is partitioned randomly into bags with different proportion labels. We firstly test the performance of models on 5 bags with proportion label distribution [0.1, 0.3, 0.4, 0.6, 0.8], [0.1, 0.2, 0.6, 0.7, 0.9] and [0.1, 0.25, 0.4, 0.75, 0.9]. All other experiment settings, including the upper bound, are the same as those discussed in the previous chapter. The only setting not discussed before is the maximum allowed batch size during the training of neural network. We set this batch size to the amount larger than the bag size in each experiment, to guarantee that no bags should be further split into subgroups during training. The detailed experiment of how different batch sizes effecting the performance will be further discussed in the future sections. The overall performance of each model, including corresponding average and standard deviation of accuracy among all five iterations of 5-fold performance testing is illustrated in table 5.1. The first part of result presents classification performance between original instances in well A02 and A13, and the second part shows the evaluation of performance with 500 more synthetic instances generated from SMOTE in the training set of each iteration,to test if all models can still perform well in various data sizes. The best performance within each proportion labels and bag sizes is highlighted in bold.

Based on the result shown, it indicates that when there is a relatively diverse distribution of bag proportion labels such as [0.1, 0.3, 0.4, 0.6, 0.8], all models perform well with most experiments having accuracies between 80% and 90% and very small standard deviation, with bag sizes varies from around 150 to more than 300. Considering that the upper bound accuracy is around 94% both with and without data augmentation, there is an approximate 5% - 10% reduction in accuracy from standard supervised learning to only proportion labels

known. Among all four methods tested, MLP with batch averager demonstrates the best performance in almost all experiment settings, with the only exception is that it's performance is a little bit worse than pSVM with proportion labels [0.1, 0.2, 0.6, 0.7, 0.9] and bag size 320. Besides the accuracy, we also use one additional metric, the AUC-ROC ("Area Under the Curve" of the "Receiver Operating Characteristic" curve) as shown in table 5.2, which also indicating similar results. Generally, the overall performance of our models is competitive in the domain of LLP, demonstrating the validity and robustness of LLP methods on morphological cell classification. However, there are still additional factors affecting the performance which will be discussed in the subsequent sections.

Table 5.1: Mean and standard deviation of accuracy for classification between A02: label 1, A13: label 0 with and without data augmentation

(a) Original data without data augmentation (Upper bound: 0.943 ± 0.013)

	pSVM $C = C_p = 1$	MLP Batch Averager	MLP Batch Averager + pSVM $C = C_p = 1$	MLP Batch Averager + Bagging pSVM $C = C_p = 1$, M=125
[0.1, 0.3, 0.4, 0.6, 0.8] Bag size 150	0.808 ± 0.032	**Batch size 256: 0.867 ± 0.042**	Batch size 256: 0.803 ± 0.029	Batch size 256: 0.846 ± 0.023
[0.1, 0.2, 0.6, 0.7, 0.9] Bag size 130	0.848 ± 0.026	**Batch size 256: 0.883 ± 0.033**	Batch size 256: 0.855 ± 0.029	Batch size 256: 0.859 ± 0.036
[0.1, 0.25, 0.4, 0.75, 0.9] Bag size 130	0.840 ± 0.030	**Batch size 256: 0.873 ± 0.020**	Batch size 256: 0.848 ± 0.044	Batch size 256: 0.868 ± 0.018

(b) After data augmentation (Upper bound: 0.942 ± 0.015)

	pSVM $C = C_p = 1$	MLP Batch Averager	MLP Batch Averager + pSVM $C = C_p = 1$	MLP Batch Averager + Bagging pSVM $C = C_p = 1$, M=125
[0.1, 0.3, 0.4, 0.6, 0.8] Bag size 340	0.844 ± 0.041	**Batch size 512: 0.879 ± 0.029**	Batch size 512: 0.841 ± 0.035	Batch size 512: 0.865 ± 0.020
[0.1, 0.2, 0.6, 0.7, 0.9] Bag size 320	**0.912 ± 0.027**	Batch size 512: 0.907 ± 0.032	Batch size 512: 0.890 ± 0.017	Batch size 512: 0.899 ± 0.021
[0.1, 0.25, 0.4, 0.75, 0.9] Bag size 320	0.893 ± 0.019	**Batch size 512: 0.901 ± 0.026**	Batch size 512: 0.887 ± 0.018	Batch size 512: 0.895 ± 0.025

5.2 Ablation Study: Different Combinations of Hyperparameters in pSVM

The value of C and C_p are critical to the performance of the proportion SVM pipeline. In most of our experiments, we use $C = C_p = 1$ as the baseline to evaluate the performance of pSVM and compare it with other methods. However, it's still highly essential to discuss how different values of C and C_p influence the result. We employ the basic comparison between

Table 5.2: Mean and standard deviation of AUC-ROC for classification between A02: label 1, A13: label 0 with and without data augmentation

(a) Original data without data augmentation (Upper bound: 0.988 ± 0.006)

	pSVM $C = C_p = 1$	MLP Batch Averager	MLP Batch Averager + pSVM $C = C_p = 1$	MLP Batch Averager + Bagging pSVM $C = C_p = 1$, M=125
[0.1, 0.3, 0.4, 0.6, 0.8] Bag size 150	0.875 ± 0.047	**Batch size 256: 0.942 ± 0.024**	Batch size 256: 0.890 ± 0.035	Batch size 256: 0.905 ± 0.022
[0.1, 0.2, 0.6, 0.7, 0.9] Bag size 130	0.927 ± 0.016	**Batch size 256: 0.956 ± 0.010**	Batch size 256: 0.925 ± 0.024	Batch size 256: 0.922 ± 0.027
[0.1, 0.25, 0.4, 0.75, 0.9] Bag size 130	0.933 ± 0.019	**Batch size 256: 0.953 ± 0.014**	Batch size 256: 0.920 ± 0.026	Batch size 256: 0.929 ± 0.024

(b) After data augmentation (Upper bound: 0.986 ± 0.010)

	pSVM $C = C_p = 1$	MLP Batch Averager	MLP Batch Averager + pSVM $C = C_p = 1$	MLP Batch Averager + Bagging pSVM $C = C_p = 1$, M=125
[0.1, 0.3, 0.4, 0.6, 0.8] Bag size 340	0.917 ± 0.026	**Batch size 512: 0.948 ± 0.023**	Batch size 512: 0.919 ± 0.024	Batch size 512: 0.924 ± 0.022
[0.1, 0.2, 0.6, 0.7, 0.9] Bag size 320	**0.968 ± 0.013**	Batch size 512: 0.966 ± 0.016	Batch size 512: 0.955 ± 0.012	Batch size 512: 0.959 ± 0.018
[0.1, 0.25, 0.4, 0.75, 0.9] Bag size 320	0.958 ± 0.016	**Batch size 512: 0.963 ± 0.017**	Batch size 512: 0.952 ± 0.011	Batch size 512: 0.959 ± 0.013

the original data within wells A02 and A13 with proportion labels [0.1, 0.3, 0.4, 0.6, 0.8] for the analysis in this part. The result of five fold cross validation of different C and C_p values are shown in table 5.3.

Table 5.3: Comparison of pSVM performance in accuracy across different values of C and C_p

(a) C = 1 with different C_p values

C_p = **0.0001**	C_p = **0.001**	C_p = **0.01**	C_p = **0.1**	C_p = **1**	C_p = **10**	C_p = **100**	C_p = **1000**	C_p = **100000**
0.459 ± 0.084	0.727 ± 0.095	0.803 ± 0.393	0.808 ± 0.045	0.81 ± 0.03	0.808 ± 0.032	0.808 ± 0.032	0.808 ± 0.032	0.808 ± 0.032

(b) $C_p = 1$ with different C values

C = 0.0001	**C = 0.001**	**C = 0.01**	**C = 0.1**	**C = 1**	**C = 10**	**C = 100**	**C = 1000**	**C = 100000**
No solution	0.745 ± 0.070	0.797 ± 0.040	0.81 ± 0.027	0.81 ± 0.03	0.797 ± 0.031	0.782 ± 0.033	0.701 ± 0.037	0.558 ± 0.027

The first part of table illustrates how performance changes while C_p varies with constant C. The increasing of C_p doesn't affect the result, even at extremely high values like 10000. However, the performance drops when C_p decreases, and the accuracy is below 50% with $C_p = 0.0001$. The hyperparameter C_p, as discussed in the methodology section, represents

the weight of proportion loss in the equation:

$$\min_{\mathbf{y},\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N} L(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i) + b) + C_p \sum_{k=1}^{K} L_p(\tilde{p}_k(\mathbf{y}), p_k) \tag{5.1}$$

$$\text{s.t.} \quad \forall_{i=1}^{N},\, y_i \in \{-1, 1\}$$

Also, as discussed before, when $\mathbf{w}$ and b fixed, the objective function becomes:

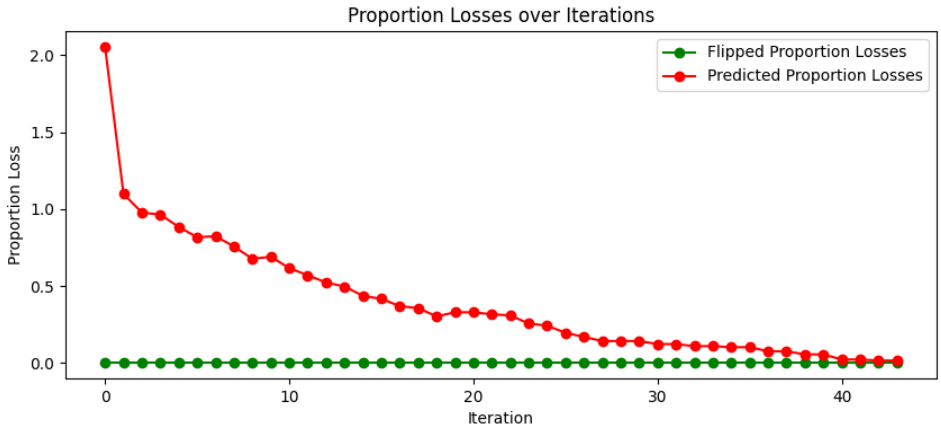
$$\min_{\{y_i | i \in \mathcal{B}_k\}} \sum_{i \in \mathcal{B}_k} L\left(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i) + b\right) + \frac{C_p}{C} L_p\left(\tilde{p}_k(\mathbf{y}), p_k\right) \tag{5.2}$$

$$\text{s.t.} \quad \forall_{i \in \mathcal{B}_k},\, y_i \in \{1, -1\}.$$

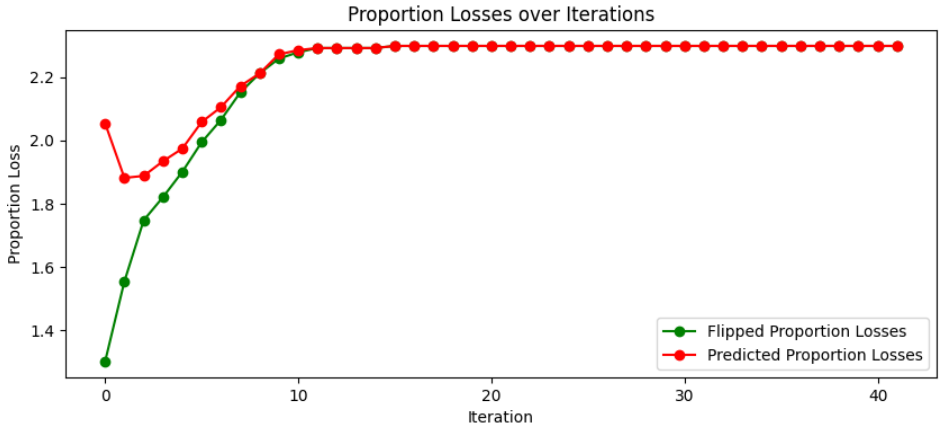
where $k = 1, 2, \ldots, K$ and K is the number of bags.

In algorithm 1, with the annealing loop, the hyperparameter C starts from a very small value (10^{-5} in our example) and gradually increases during the training process. It ensures the weight of proportion loss ($\frac{C_p}{C}$) during the optimization of Y is very high initially. We interpret this setting as achieving the best proportion loss by flipping in the beginning, and then gradually optimizes towards this solution during the entire training process. An example of this process is shown in figure 5.1a, where the green line representing the proportion loss after the flipping and the red line representing the proportion loss of the support vector classifier after each iteration of pSVM. It illustrates the entire process that the flipped proportion loss is minimized into the lowest possible value in the beginning due to the high $\frac{C_p}{C}$, with the support vector classifier gradually learning towards this optimal target. When the C_p is set to an extremely large value, it doesn't effect the training process a lot since the weight $\frac{C_p}{C}$ has been large enough to guarantee the green line is bounded in the bottom. Conversely, when C_p is set to a very small value like 0.0001, the proportion loss after flipping fails to remain in the optimal value, as illustrated in figure 5.1b, leading a significantly worse performance.

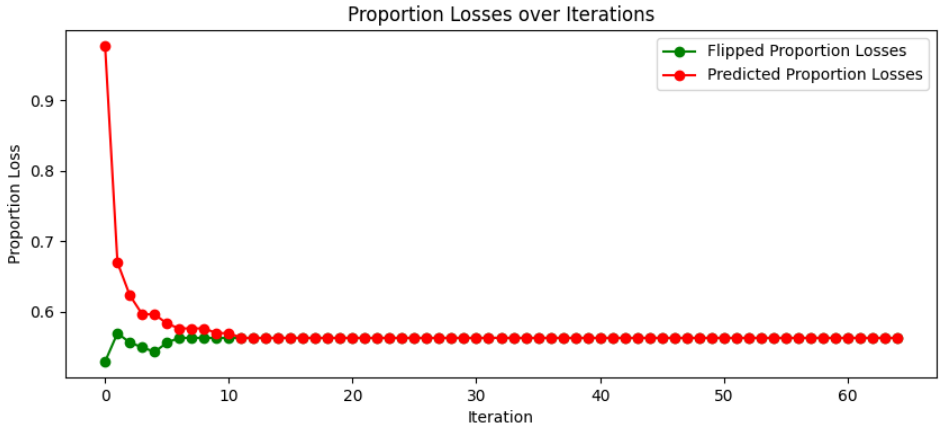
On the other hand, the hyperparameter C in pSVM directly corresponds to the regularization parameter during the training of a traditional SVM for classification, which controls the penalty on instances on the wrong side of the margin. An excessively large C may lead to two potential issues: overfitting, and not enough weight on proportion loss as discussed above. In the second part of table 5.3, it can be observed that the performance starts to drop quickly from $C = 10000$. In our experiment specifically, this drop is not caused by overfitting, as illustrated in figure 5.2 where both training and validation accuracies are low when C is set to 100000. In fact, as shown in figure 5.1c, the flipped proportion loss is much higher than it is in figure 5.1a, which is cause by $\frac{C_p}{C}$ is not sufficiently large when $C = 100000$. Additionally, when C is too small, the performance drops due to the underfitting of SVM



(a) Training process of pSVM with $C = C_p = 1$



(b) Training process of pSVM with $C = 1$, $C_p = 0.0001$



(c) Training process of pSVM with $C = 100000$, $C_p = 1$

Figure 5.1: Comparison of pSVM training process among different C and C_p values

since there is no enough penalty. One special case in the result is when the model fails to converge ("no solution") when C is too small such as 0.0001, because the SVM solver creates a hyperplane such that all data points fall in one side of it. In this case, the result cannot be used for training the SVM in the next iteration of the annealing loop since data points from both classes are needed for training.

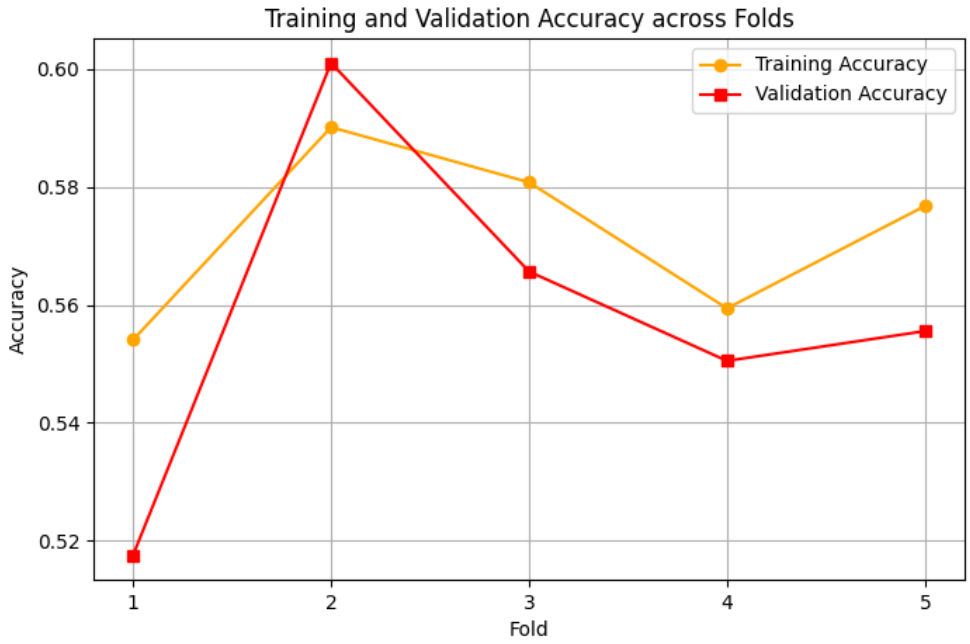


Figure 5.2: An example of training and validation accuracy with $C_p = 1$ and $C = 100000$

5.3 The Effect of Proportion Label Distribution on Model's Performance

In binary classification, one special challenge of LLP arises when the proportion label of a bag is close to 0.5. In an extreme case where a bag containing exact 50% positive instances and 50% negative instances, a model may misclassify all positive instances as negative and vice versa but still have zero loss on bag proportion, but zero accuracy in instance level prediction. Since the only annotation in LLP during training is the bag proportion, a proportion label of 0.5 can be considered as providing almost no information for instance level classification. On the contrary, a bag proportion label close to 1 or 0 can be considered as most informative, since they are much similar to fully supervised learning scenarios. In our experiment, the result of this comparison is shown in table 5.4, where all models performs poorly in our experiment when trained with bag labels [0.5, 0.5, 0.5, 0.5, 0.5]. Their

performance drops from more than 80% accuracy to around 50%, making them unsuitable for practical application. The label close to 0.5 can be considered as a systematic challenge where all LLP frameworks struggle to solve. Similar conclusion was also discussed in [25].

Table 5.4: Performance comparison of models on diverse distribution of proportion labels and all 0.5 proportion labels, in accuracy

	pSVM $C = C_p = 1$	**MLP Batch Averager**	**MLP Batch Averager + pSVM** $C = C_p = 1$	**MLP Batch Averager + bagging pSVM** $C = C_p = 1$, $M = 125$
[0.1, 0.3, 0.4, 0.6, 0.8] Bag size 150	0.808 ± 0.032	**Batch size 256:** 0.867 ± 0.042	**Batch size 256:** 0.803 ± 0.029	**Batch size 256:** 0.846 ± 0.023
[0.5, 0.5, 0.5, 0.5, 0.5] Bag size 130	0.602 ± 0.116	**Batch size 256:** 0.462 ± 0.067	**Batch size 256:** 0.563 ± 0.134	**Batch size 256:** 0.513 ± 0.179

5.4 Batch averager performance with varying batch sizes

The batch size in deep learning refers to the amount of data processed by the neural network per time. In many cases, there are very few influences of different batch sizes on the final accuracy of a model. However, it is not the case with batch averager. As discussed in the methodology section, the batch averager aims at matching the posterior probability of entire bag with prior proportion labels, requiring that all instances of one batch during the training of neural network should be from the same bag. When the bag size is smaller than batch size, the batch size can be dynamically modified to fit the bag size, as shown in algorithm 2. However, when the batch size is not enough to contain one entire bag due to the limited GPU memory, the bag need to be further split into several subgroups to be fed to the neural network. During this process, it's hard to guarantee that each subgroup of bag still keep the original proportion distribution, lead to suboptimal results. Due to the relatively simple model structure and tabular data format in our experiment, the maximum batch size can always be larger than the bag size, as illustrated in table 5.1. However, sufficient batch size is hard to be guaranteed in some other experiments with extremely large bag size, complex data structure, or more complex neural networks. In this case, it's highly necessary to discuss how performance of our models varies with insufficient batch size.

Our experiment results clearly indicate this issue. In table 5.5 with proportion label [0.1, 0.3, 0.4, 0.6, 0.8], the neural network with batch averager can achieve 86.7% accuracy when the batch size is 256 which is much larger than the bag size. However, the performance drops significantly when the batch size decreases below bag size, resulting only 67.1% with batch size 64. Similar result can also be observed with proportion label distribution [0.1, 0.2, 0.6, 0.7, 0.9], confirming that the batch averager layer struggles to learn the structure of entire

bag from subgroups only. The figures 5.3, 5.4, 5.5 provide further insight into the training process of the network with varying batch sizes. These figures illustrate the training and validation accuracy over the entire training process on data with batch sizes 64, 128 and 256 and proportion label [0.1, 0.3, 0.4, 0.6, 0.8], bag size 150 . The result demonstrates that the training process is more stable and can achieve an overall higher performance when the batch size exceeds the bag size. Due to the characteristics of morphological cell classification which often contains large data size, and limitations of GPU memory, the issue where batch size is not sufficient for entire bag is particularly relevant.

Table 5.5: Performance comparison of MLP batch averager on different batch sizes, in accuracy

	pSVM $C = C_p = 1$	**MLP Batch Averager**
[0.1, 0.3, 0.4, 0.6, 0.8] Bag size 150	0.808 ± 0.032	**Batch size 256:** 0.867 ± 0.042 Batch size 128: 0.780 ± 0.035 Batch size 64: 0.671 ± 0.045
[0.1, 0.2, 0.6, 0.7, 0.9] Bag size 130	0.848 ± 0.026	**Batch size 256:** 0.883 ± 0.033 Batch size 128: 0.866 ± 0.019 Batch size 64: 0.778 ± 0.029

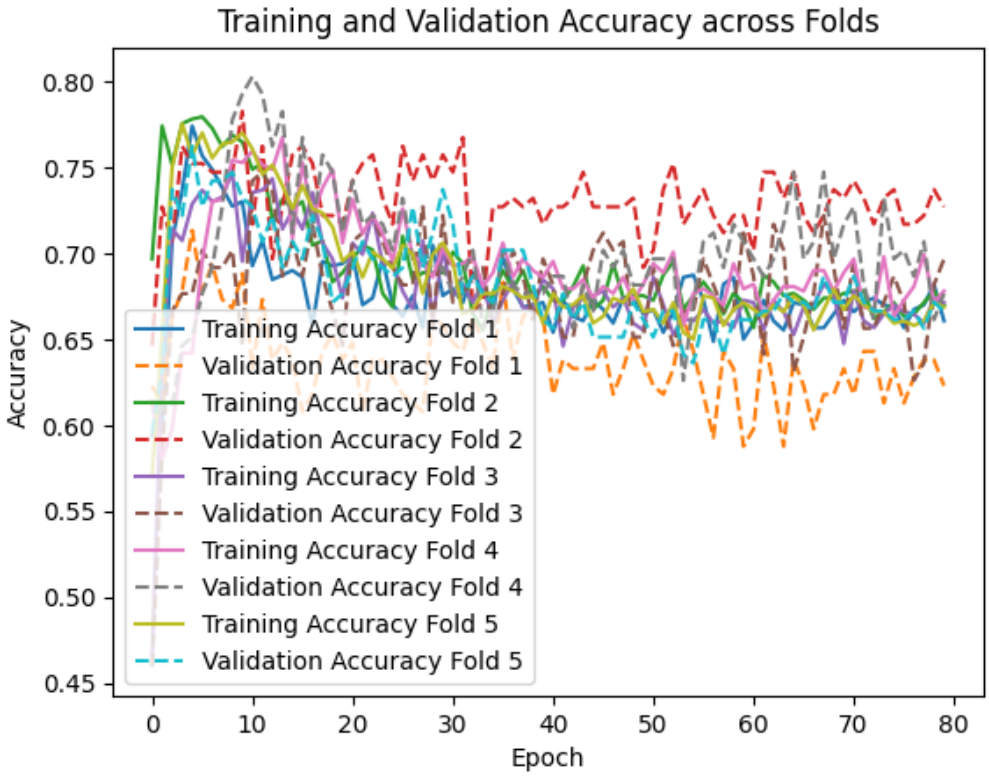


Figure 5.3: Training process with batch size 64 and bag size 150 on proportion labels [0.1, 0.3, 0.4, 0.6, 0.8]

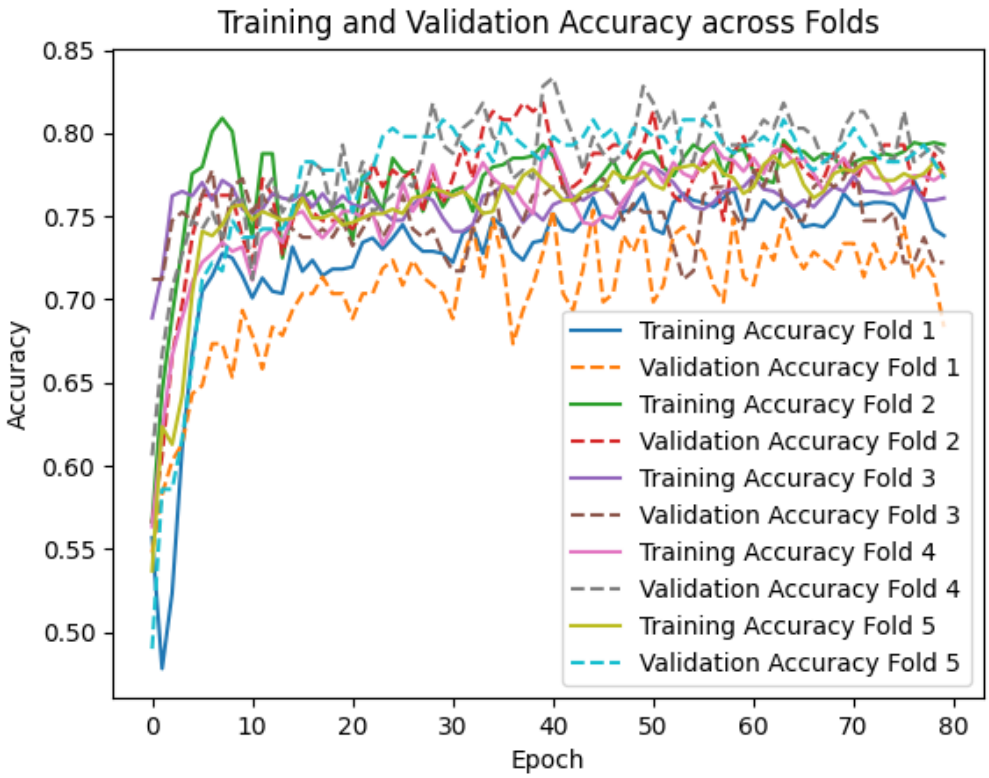


Figure 5.4: Training process with batch size 128 and bag size 150 on proportion labels [0.1, 0.3, 0.4, 0.6, 0.8]

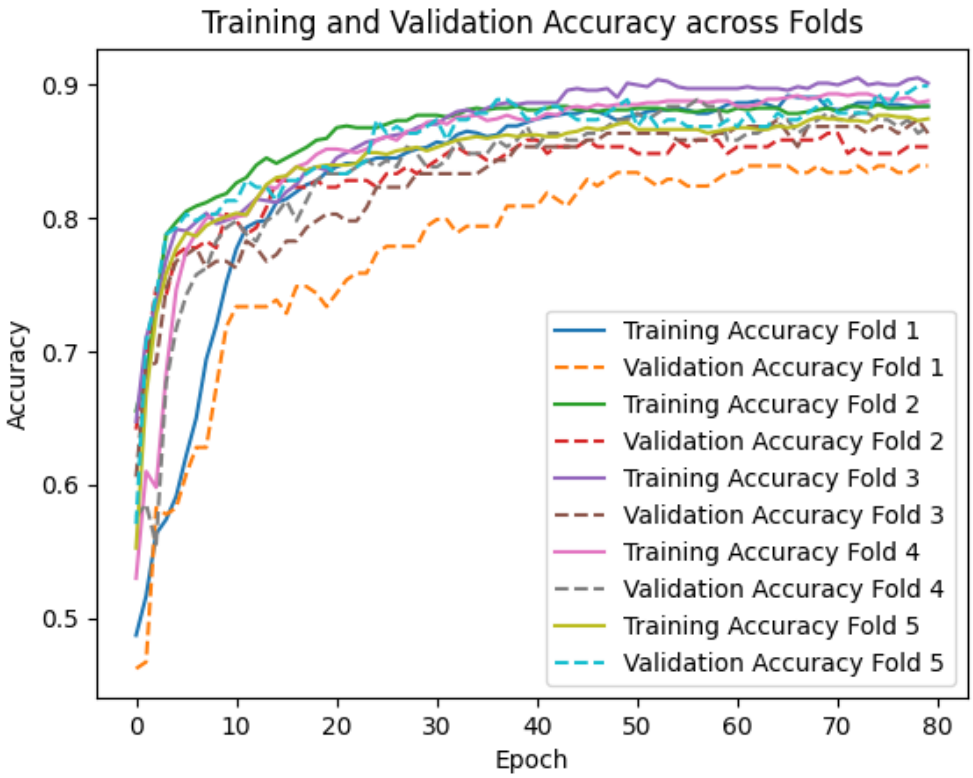


Figure 5.5: Training process with batch size 256 and bag size 150 on proportion labels [0.1, 0.3, 0.4, 0.6, 0.8]

5.5 The Combined Pipeline of MLP Batch Averager and Proportion SVM

As discussed before, the performance of neural network with batch averager is worse, when the batch size during training is less than the bag size. Fortunately, the performance of neural network with batch size smaller than bag size can be further improved by combining it with pSVM, as introduced in the model selection section before. As shown in table 5.6 with proportion distribution [0.1, 0.3, 0.4, 0.6, 0.8] and [0.1, 0.2, 0.6, 0.7, 0.9], even though the accuracy drops to around 60% - 70% with limited batch size, applying pSVM after the neural network can boost accuracy back to more than 80%. Notably, in many cases of our experiments, the accuracy of MLP Batch Averager + pSVM is even higher than using pSVM alone. This can be interpreted as, the result of MLP Batch Averager, might be suboptimal in some cases, but still provided helpful direction of optimization and meaningful intermediate result for pSVM to further improve it as the final stage to boost the performance.

Table 5.6: Illustration on the improvement of accuracy after combining MLP batch averager with pSVM

	pSVM $C = C_p = 1$	**MLP Batch Averager**	**MLP Batch Averager + pSVM** $C = C_p = 1$
[0.1, 0.3, 0.4, 0.6, 0.8] Bag size 150	0.808 ± 0.032	Batch size 256: 0.867 ± 0.042 Batch size 128: 0.780 ± 0.035 Batch size 64: 0.671 ± 0.045	Batch size 256: 0.803 ± 0.029 Batch size 128: 0.815 ± 0.024 Batch size 64: 0.815 ± 0.022
[0.1, 0.2, 0.6, 0.7, 0.9] Bag size 130	0.848 ± 0.026	Batch size 256: 0.883 ± 0.033 Batch size 128: 0.866 ± 0.019 Batch size 64: 0.778 ± 0.029	Batch size 256: 0.855 ± 0.029 Batch size 128: 0.850 ± 0.024 Batch size 64: 0.845 ± 0.024

Another potential benefit of combining MLP batch averager and pSVM is the reduction on training time. The Support Vector Classifier (SVC) solver in many popular machine learning libraries, such as scikit-learn, is based on libsvm[39] quadratic programming problem (QP) solver, which has the time complexity between O(N^2) and O(N^3), where N is the number of data points. Therefore, one of the main challenges of pSVM is that the time used for solving a SVM increases significantly when data size grows. Additionally, as illustrated in the methodology part, the proportion SVM pipeline involves iteratively solving the Support Vector Classifier with a large amount of iterations, which further increase the total computation time.

As discussed in [20], the author advise to run pSVM with 10 times random initializations to minimize the effect of initialization on the final result. For the combined method as discussed before in section 4.2.1, the input of pSVM is the output of neural network as the intermediate result, which makes the pSVM pipeline doesn't need to be repeated for 10 or even more times. The comparison between execution time of different models is shown in figure 5.7. Initially, the linear kernel is used in the support vector classifier part of pSVM

pipeline. It indicates that when the total data size is approximately 1000 with a bag size of 150, the combined pipeline shows little advantages in training time compared with using pSVM only. However, when the data size doubled to around 2000, the combined method shows obvious reduction on training time. Although the linear kernel is primarily used in our experiment, we also test the training time for RBF kernel since it's often necessary for more complex tasks. The gap in training time between pSVM and the combined pipeline is much larger with RBF kernel, where the pSVM is over 4 times slower than the combined method when data size is around 2000. In conclusion, the combined method can both address the challenge of neural network in LLP with limited batch size, and the issue of pSVM with long training time.

Table 5.7: Comparison of training time for different models in different data size, in seconds

(a) Linear Kernel

	pSVM Linear Kernel	**MLP Batch Averager**	**MLP Batch Averager + pSVM Linear Kernel**
Bag size 150 Data size ∼1000	145.69	66.68	151.34
Bag size 300 Data size ∼2000	491	82.81	311.91

(b) RBF Kernel

	pSVM RBF Kernel	**MLP Batch Averager**	**MLP Batch Averager + pSVM RBF Kernel**
Bag size 150 Data size ∼1000	330.63	66.68	173.42
Bag size 300 Data size ∼2000	1930	82.81	423.91

5.6 Performance of the Ensemble Method

As discussed in the methodology section, we also implemented an ensemble method based on bagging in LLP scenario. In the algorithm, each weak classifier is trained on a random subsample of bags with replacement, and the prior weight of sampling on each bag is $|Y_i - 0.5| + \epsilon$ where Y_i is the proportion label of the bag i. The motivation of this prior weight is from our previous experiments which indicate that the bag with proportion label closer to 0.5 is consider as less informative. As a result, the closer the proportion label of a bag is to 0 or 1, the higher probability it has to be selected in a weak classifier. For classification on new instances, the final predicted label of one instance is the majority vote of all M weak classifiers. The impact after incorporating ensemble method with the MLP batch averager + pSVM pipeline is listed in table 5.8. We use 125 weak classifiers (M = 125) as the baseline in our experiment. To also illustrate that the weight prior is improving the performance, we

also illustrate the result of bagging without prior, in which each bag has equal probability to be selected in each weak classifier, for comparison.

Table 5.8: Performance comparison after adding ensemble method, with and without weight prior

	MLP Batch Averager + pSVM $C = C_p = 1$	**MLP Batch Averager + bagging pSVM** $C = C_p = 1$, **M = 125 with prior**	**MLP Batch Averager + bagging pSVM** $C = C_p = 1$, **M = 125 without prior**
[0.1, 0.3, 0.4, 0.6, 0.8] Bag size 150	0.803 ± 0.029	**0.846 ± 0.023**	0.815 ± 0.013
[0.1, 0.2, 0.6, 0.7, 0.9] Bag size 130	0.855 ± 0.029	**0.859 ± 0.036**	0.832 ± 0.035

As illustrated in the result, for bag proportion distributions such as [0.1, 0.3, 0.4, 0.6, 0.8], bagging further improves the performance by around 4 percentage points. However, for bag proportions [0.1, 0.2, 0.6, 0.7, 0.9], the bagging does not significantly improve the performance. We interpret the result as because some bag proportions in the second case (0.2, 0.7, 0.9) have been closer enough to 0 and 1 compared with proportions in the first case (0.3, 0.6, 0.8), making the enhancement of bagging not obvious in this case. The result shows that the bagging with specific prior weight on bags is potential on further improvement of prediction accuracy, but the result also depends on the overall proportion label distribution we have in experiment, making the impact of bagging in our experiment highly context-dependent. However, it does indicate that adding weight prior to each bag in ensemble method is benefiting the model's performance, because there is almost no improvement on bagging result without weight prior.

Chapter 6

Conclusion and Future Work

The purpose of this project is to investigate the feasibility of making classification on cells with only proportion labels on groups of cells known, and analyze the performance of such classification models in different circumstances. After extensive review on background and related works, we decided to combine machine learning models with LLP (Learning from Label Proportions) pipelines as the central approach of investigation on the research question. In our experiment, we apply proportion SVM, Multilayer Perceptron with batch averager , the combined method of these two and also the ensemble method on the "Cell Image Library" dataset, with different data sizes, bag sizes, proportion label distributions and model configurations to thoroughly assess the performance.

To answer the original research question, our findings can confirm that LLP models are practical to make classification where only proportion labels of cells are known with their morphological features, with relatively competitive performance. Generally, compared with the fully supervised learning scenario where annotation on all cells are known, there is an approximately 5 to 10 percentage points reduction on accuracy if we only have proportion labels. Based on our observation on experiment results, proportion SVM can be considered as a general model, recommended as the first candidate for this type of tasks. However, when the data size is large, the time required for training pSVM becomes too long due to the characteristics of SVC solver. Neural network with batch averager has less time complexity with better accuracy in its best case, but the performance drops quickly when the bag size is too large, making the best application of it is when there is a large dataset overall but relatively small bags. We also found that combining Proportion SVM and Neural network with batch averager can solve the issue of neural network with limited batch size, and still keep relatively shorter training time than using proportion SVM only. In addition, the ensemble method has potential to further improve the performance but it dependends on the proportion distributions of bags.

Besides the findings above, we also did some supplementary experiments discussing the

influences of hyperparameters in different models. Trying various candidates of hyperparameters on pSVM is highly recommended in future tasks, but extreme values should always be avoided. For neural network with batch averager, it is always recommended to use as large a batch size as possible within GPU constraints to avoid performance reduction. We also found that the bags with proportion labels close to 0.5 are considered as containing very few information on instance level classification, which means that in further applications of LLP it's highly recommended to avoid too many bags with this type of proportion labels in dataset.

Besides valuable insights discussed above, there are also some limitations on our work worth further exploration. First, due to the limitations on the original dataset we use the CIL dataset as a substitute, which brings more assumptions. We consider the classification between a well treated with chemical compound and another negative control well from the same cell line as an approximation of classification between different types of cells, since they both have morphological differences. But further tests on a dataset with distinct cell lines are still necessary. Moreover, all our models are tested on tabular data with morphological features already available. With the advancement of more complex deep learning models such as Convolutional Neural Networks (CNNs), further works on applying CNN directly with LLP pipelines may further increase the performance.

Bibliography

[1] Biobide. *Ex Vivo vs In Vitro: Understand the Difference*. URL: https://blog.biobide.com/ex-vivo-vs-in-vitro-understand-the-difference.

[2] Marianne Baillargeon. *Next-Generation Lead Compound Identification and Optimization: Using High-Content Screening with 3D Organoids*. Feb. 2022.

[3] John M. Kirkwood et al. "Immunotherapy of cancer in 2012". In: *CA: a cancer journal for clinicians* 62.5 (2012), pp. 309–335.

[4] Nicolas Boucherit, Laurent Gorvel, and Daniel Olive. "3D tumor models and their use for the testing of immunotherapies". In: *Frontiers in Immunology* 11 (2020), p. 603640.

[5] R. W. Horobin. "Biological staining: mechanisms and theory". In: *Biotechnic & Histochemistry* 77.1 (2002), pp. 3–13.

[6] Shidan Wang et al. "Computational staining of pathology images to study the tumor microenvironment in lung cancer". In: *Cancer Research* 80.10 (2020), pp. 2056–2066.

[7] Philip Went et al. "Expression and prognostic significance of EpCAM". In: *J Cancer Mol* 3.6 (2008), pp. 169–174.

[8] Elaheh Alizadeh et al. "Cellular morphological features are predictive markers of cancer cell state". In: *Computers in Biology and Medicine* 126 (2020), p. 104044.

[9] Z. Li. "In Vitro Micro-Tissue and -Organ Models for Toxicity Testing". In: *Comprehensive Biotechnology*. 2nd. Vol. 5. Elsevier, 2011, pp. 551–563. DOI: https://doi.org/10.1016/B978-0-08-088504-9.00503-1.

[10] Zeynab Mousavikhamene et al. "Morphological features of single cells enable accurate automated classification of cancer from non-cancer cell lines". In: *Scientific Reports* 11.1 (2021), p. 24375.

[11] Yi Li et al. "Cell morphology-based machine learning models for human cell state classification". In: *NPJ Systems Biology and Applications* 7.1 (2021), p. 23.

[12] Mohammad R. Hasan et al. “Classification of cancer cells using computational analysis of dynamic morphology”. In: *Computer Methods and Programs in Biomedicine* 156 (2018), pp. 105–112.

[13] Tabassum Yesmin Rahman et al. “Study of morphological and textural features for classification of oral squamous cell carcinoma by traditional machine learning techniques”. In: *Cancer Reports* 3.6 (2020), e1293.

[14] Yukiko Nagao et al. “Robust classification of cell cycle phase and biological feature extraction by image-based deep learning”. In: *Molecular Biology of the Cell* 31.13 (2020), pp. 1346–1354.

[15] Dhananjay Bhaskar et al. “A methodology for morphological feature extraction and unsupervised cell classification”. In: *BioRxiv* (2019), p. 623793.

[16] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.11 (2008), pp. 2579–2605.

[17] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*. Vol. 96. 34. 1996, pp. 226–231.

[18] Jude M. Phillip et al. “A robust unsupervised machine-learning method to quantify the morphological heterogeneity of cells and nuclei”. In: *Nature Protocols* 16.2 (2021), pp. 754–774.

[19] Mahyar Salek et al. “Realtime morphological characterization and sorting of unlabeled viable cells using deep learning”. In: *bioRxiv* (2022), pp. 2022–02.

[20] Felix Yu et al. “Proportion-SVM for Learning with Label Proportions”. In: *International Conference on Machine Learning*. PMLR. 2013, pp. 504–512.

[21] Bo Wang, Zhensong Chen, and Zhiquan Qi. “Linear twin SVM for learning from label proportions”. In: *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. Vol. 3. IEEE. 2015, pp. 88–91.

[22] Zhensong Chen et al. “Learning with label proportions based on nonparallel support vector machines”. In: *Knowledge-Based Systems* 119 (2017), pp. 126–141.

[23] Zhiquan Qi et al. “Learning with label proportions via NPSVM”. In: *IEEE Transactions on Cybernetics* 47.10 (2016), pp. 3293–3305.

[24] Yong Shi et al. “Learning from label proportions on high-dimensional data”. In: *Neural Networks* 103 (2018), pp. 9–18.

[25] Zhensong Chen, Wei Chen, and Yong Shi. “Ensemble learning with label proportions for bankruptcy prediction”. In: *Expert Systems with Applications* 146 (2020), p. 113155.

[26] Zhiquan Qi et al. “Adaboost-LLP: A boosting method for learning with label proportions”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.8 (2017), pp. 3548–3559.

[27] Ehsan Mohammady Ardehaly and Aron Culotta. “Co-training for demographic classification using deep learning from label proportions”. In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2017, pp. 1017–1024.

[28] Yong Shi et al. “Deep learning from label proportions with labeled samples”. In: *Neural Networks* 128 (2020), pp. 73–81.

[29] Jiabin Liu et al. “Learning from label proportions with generative adversarial networks”. In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[30] Bo Wang, Yingte Sun, and Qiang Tong. “LLP-AAE: Learning from label proportions with adversarial autoencoder”. In: *Neurocomputing* 537 (2023), pp. 282–295.

[31] Jiabin Liu et al. “SELF-LLP: Self-supervised learning from label proportions with self-ensemble”. In: *Pattern Recognition* 129 (2022), p. 108767.

[32] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[33] Leo Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.

[34] Mark-Anthony Bray et al. “A dataset of images and morphological profiles of 30 000 small-molecule treatments using the Cell Painting assay”. In: *Gigascience* 6.12 (2017).

[35] Mark-Anthony Bray et al. “Cell Painting, a high-content image-based assay for morphological profiling using multiplexed fluorescent dyes”. In: *Nature protocols* 11.9 (2016), pp. 1757–1774.

[36] Diederik P. Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[37] Russell Warne. “A primer on multivariate analysis of variance (MANOVA) for behavioral scientists”. In: *Practical Assessment, Research, and Evaluation* 19.1 (2014).

[38] Nitesh V. Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357.

[39] Chih-Chung Chang and Chih-Jen Lin. “LIBSVM: a library for support vector machines”. In: *ACM transactions on intelligent systems and technology (TIST)* 2.3 (2011), pp. 1–27.