Exploring Image Classification on Multi-class Imbalanced Datasets

Tianyu Wang

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh
2020

Abstract

Imbalanced data set is one of the factor that make classification job difficult, and real world data are often imbalanced, for example, the fraud email detection job for binary classification and species classification for multi class classification, the ratio between majority class and minority class are high in those jobs, which make classification job difficult. In past years, there are many research focusing on the classification of imbalanced data set. In our work, several commonly used methods will be investigated and compared on manually created skewed distributions of several data sets, I will also introduce a Hierarchical method on multi-class imbalanced data sets, which include creating class taxonomy on the original classes, the effeteness of these methods will be study.

Acknowledgement

This project is supervised under Dr. Hakan Bilen. Thanks to his helps and supports.

Table of Contents

Chapter 1

Introduction

Over the recent year, the classification of imbalanced data set has become an important research topic in the area of machine learning as many real-world problem have imbalanced data set. In an imbalanced data set, one or a few classes (majority classes) will have much more number of samples than other classes (minority class), the ratio between classes in some real-world problem can easily exceed 1000[12]. The study of imbalance data set is mainly focused on the binary classification problem, while multi-class imbalance data set haven't been studied such intensely[17]. Our work will be focusing on the multi-class imbalance data set.

The problem caused by unbalanced data set mainly comes from two part [34]. The first part is "global imbalance ratio", the skewed distribution of the data set, the classifier will biased to the majority class as classifying samples to majority class will easily reach a very high overall performance. The second part is "data difficulty factors" which is, the minority classes are naturally not well represented, meaning, it is not possible for the classifier to learn the structure of the minority classed as there are too few samples of them. The second case doesn't have any good solution other than finding more training samples of minority class. Many other study [26] [35][19] have been done to distinguish these two problem by giving minority classes four types, namely "safe", "borderline", "rare" and "outliers". In our work, to minimize the second source of problem, I kept the minority class in the "safe" or "borderline" category, by using data set that have rich number of samples in every class, and create imbalanced data set using subset from these data set.

Several methods have been developed through recent years of study on imbalanced data set problem, conventionally researchers classify these methods as three categories [10]:

- Data-level-method: Mainly use re-sampling technique on the imbalanced data set to re-balanced the ratio of classes, the state-of-art method is called SMOTE, which over-sampling of minority class by creating synthetic minority class samples.
- Algorithm-level-method: Instead of modify the imbalanced data set, this groups of methods directly change the classifier and try to reduce their bias towards

majority class, one of popular approach is cost-sensitive approaches, which will be study in our work.

- Hybrid method: As the data-level-method happens at the prepossessing stage, so it's not conflict with the algorithm-level-method. Hybrid method combine the two methods together.

In our work, I will investigate two commonly used methods: random over sampling and cost-sensitive approach, and introduce a new method using hierarchical classification [32].

1.1 Structure of the report

- Chapter 2 will describe the background for better understanding of our work. I will explain the basic knowledge of neural network and convolutional neural network, describe the data set and framework I are using and describe state-of-art method investigated in our work.
- Chapter 3 explain the reason behind the choice of methods ans metrics.
- Chapter 4 explain the creation of imbalanced data set, the way I simulate skewed distribution on the balanced data set I used, the actual distribution of each imbalanced data set and the performance of flat classification on those data sets.
- Chapter 5 will go through the implementation of three methods used in our work, namely Random Over Sampling, cost-sensitive learning and hierarchical classification. It will explain the detailed algorithms and their implementation on given framework. This chapter will also include a bit about the implementation of simple models I used.
- Chapter 6 will go through all the experiments I done comparing three methods, for each experiment I will given experiment settings, result of experiment and discussion on possible indication of result. The statistic of the result will include accuracy for overall performance, sensitivity and specificity for class-wise performance.
- Chapter 7 will give a summary to our work, provide a conclusion and discuss further work that can improve on our work.

Chapter 2

Background

2.1 Classification

"Classification" or "Pattern Classification" [6], is a problem that have studied intensively in the areas of machine learning, statistics and many other areas, a classification task involve building a classifier to identifying the category/class that new observations belong to based on old observations and their categories/classed. This task have many real life applications, such as tumor classification [27], Handwriting recognition [30], Document classification [2], etc.

Our paper is mainly focus on image classification on imbalanced data set, where sizes of classed in our observed data set is highly imbalanced. Many data from real life are imbalanced, for example, in tumor classification the number benign tumor is much larger than the malignant tumor [24]. The imbalanced data set problem pose a challenge, as canonical machine learning algorithms assume number of object in each class is roughly the same [18], and these algorithm will bias towards the majority class, giving poor performance on minority class.

2.2 Artificial neural network

Artificial Neural Network [11] is a system have been widely applied to solve problem such as classification, for image classification, Convolutional Neural Network [21][23], a branch of Artificial Neural Network, is widely applied and producing state-of-art results, I will briefly describe basics of Artificial Neural Network and Convolutional Neural Network in this section.

2.2.1 Basic components of artificial neural network

The basic components of Artificial neural network are Neurons, each Neuron takes inputs and do a weighted summation of the inputs, and pass the results to a activation function, usually non linear, and get the output. I can describe behaviour of a neuron

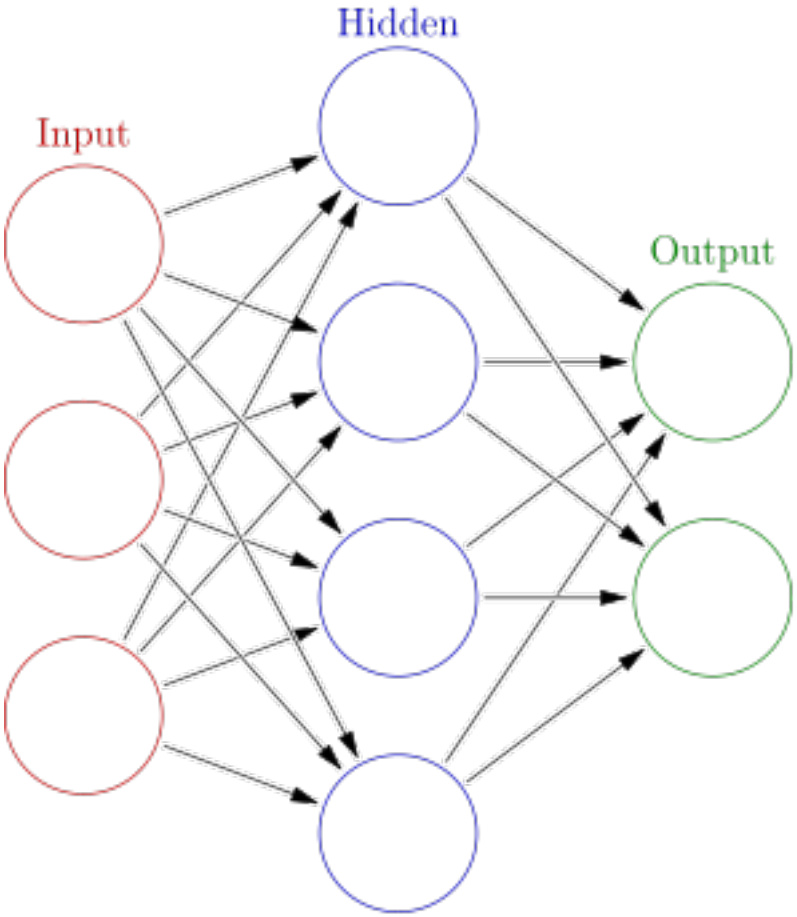

Figure 2.1: An example of a two layers Artificial Neural Network, author by Glosser.ca, can be access on wikipedia

using the following formula:

$$f(x) = g(\sum_{i=0}^{m} w_i x_i) \tag{2.1}$$

m: number of the inputs
x: $x_0 = 1$, x_i to x_m are inputs
w: weights store in the neuron
g(**x**): activation function

Neurons in a Artificial Neural Network are arranged in layers, each layer contains multiple neurons taking inputs from previous layer and outputs will treated as input to next layer, figure 2.1 shows an example of such structure, in this figure, the blue circles and the green circles are neurons, and the red circles are inputs to this neural network, and this neural network producing two outputs. Then a loss function will compute cost based on the outputs and the ground true target values, in classification, the values I want the network to produce would be probability of each class and the loss function

is usually cross-entropy loss [14], and I want to neural network to be able to predict the ground true target with suitable weights in each neurons. Finally, Artificial Neural Network can updates its weight and minimizing the total cost compute from the loss function given the training set with samples and ground true labels by using gradient descent [31] and back propagation [13].

2.2.2 Convolutional neural network

The example Neural Network in 2.1 is fully-connected, where every neurons is connected to all the neurons/inputs in previous layer, this is a commonly used structure [28], but when I have image as inputs, this structure becomes unsutibale, as it doesn't explore spatial structure in the images, pixels close to each other are treat same as pixels far from each other. This structure also have too much parameters when apply to image data, as image data have very high dimension, making the network hard to train. In 1998, Yann LeCun proposed LeNet [23] for classification of hand-written numbers, this approach later become the foundation of Convolutional Nerual Network, which is well-adapted to image data [28]. Instead of fully-connected neurons between layers, Convolutional Nerual Network taking use of kernels to do convolution operation on input image or feature maps from previous layer about producing feature map for next layer [25]. kernels are able to detect local features which explore spatial structure in the images, and they have limit size which is usually 5x5 or 3x3, thus the number weights for each layer would be: **number of kernels** $*$ **kernel height** $*$ **kernel width**, which is much smaller compare to fully connected layer.

2.3 Hierarchical method

hierarchical classification have been applied in many other domains, most of time the classes comes with a nature taxonomy [33], Susan, Dumais, et [7] apply a two-level hierarchical structure for for classifying web content with a given taxonomy , Vailaya. Aditya, et al.[36] apply hierarchical classification of vacation images which is manually grouped. Fan, Jianping, et al.[9] apply hierarchical classification for image classification with hierarchically organized image concept which are manually created. But the effect of hierarchical classification on imbalanced data set haven't been study intensively. Beyan [1] has applied hierarchical decomposition on class imbalance problem and reach a state-of-art result, but that work was mainly focusing on the binary classification instead of multi-class classification. In that work, the author propose a hierarchical method on binary imbalanced data set by apply out-liar detection at each level of hierarchy to separate minority class from the majority class. This is totally different from our approach. Our method is only suitable for multi-class data set, it will create a taxonomy of the classes and try to balance the number of each class at every level of hierarchy, and turning the unbalanced multi-class data set into several balanced binary data set.

2.4 Cost-sensitive learning

Cost-sensitive learning is a commonly used algorithm-level approach for solving imbalanced issue [15], instead of penalize miss-classification equally, Cost-Sensitive learning penalize miss-classification using the cost defines in a cost matrix, a cost matrix in binary class dataset will have following form: [8]:

	actual negative	actual positive
predict negative	$C(0,0) = c_{0,0}$	$C(0,1) = c_{0,1}$
predict positive	$C(1,0) = c_{1,0}$	$C(1,1) = c_{1,1}$

By giving the miss-classification of minority classes higher costs, the classifier, minimizing the total cost of classification, will have a higher sensitivity on minority classes. One of the most important and hard problem in cost-sensitive learning is to define a suitable cost-sensitive matrix. Khan, Salman H et. al(2017) [16] introduced a cost-sensitive deep neural network which are able to optimize the cost matrix during learning time, their approach have been test on several vision datasets and largely outperformance popular data sampling techniques. Details of this approach will be describe in section 5.3.

2.5 Pytorch library and MLP framework

Pytorch[29] library and Pytorch MLP framework are used in this work, in order to have efficient implementation and testing of different methods. Pytorch is an open source machine learning library that enable user to efficiently design and experiment different neural network structure, complex calculation of differentiation and back propagation are all handled by the framework automatically. Pytorch MLP framework is designed on top of Pytorch to have quick experiment setup and running, it's designed for teaching use in course Machine Learning Practical at University of Edinburgh, code for this framework is licensed under Modified BSD License.

Chapter 3

Plan and decision

In this work, I want to investigate different methods on solving the negative effect of class imbalance on muti-class image classification, and try to develop new methods.

3.1 Methods to investigate

As mentioned in the background, there are mainly three types of the method on solving this problem, data-level-method, algorithm-level-method and hybrid method which is a combination of former two kinds of methods. For data-level-method I planned to investigate Random Over Sampling (ROS), as it's showed in [3], ROS doesn't cause over-fitting when applying to vision dataset, it has the best performance for most of the time, and it is easy to implement, I didn't use the popular oversampling technique SMOTE as SMOTE is only applicable to the features of the samples, but for image dataset what I have are pixels, not features, and averaging pixels between two images wouldn't result in a sensible image most of the time. For algorithm-level-method, I planned to implement the cost sensitive loss function in [16], as it is the most promising cost sensitive method that applies to a multi-class image dataset.

3.2 Hierarchical classifier

The hierarchical method is usually used when the class has nature taxonomy as describe in 2.3, but none have applied it specifically on solving imbalance issues on a multi-class dataset. By clustering minority classes together we can bring an imbalanced multi-class dataset to a binary balanced binary class dataset as shown in figure 3.1, by applying this clustering recursively method we can build a class taxonomy for every multi-class dataset. Then, applying a hierarchical classifier using this class taxonomy, every single classifier in the hierarchy will have a balanced binary dataset to train and classify, which rules out the class imbalance problem. But this hierarchical method will also have disadvantages, for example, as there are multiple classifiers from the root to leaf of this hierarchy, the error of each classifier will aggregate, and may result in a high error rate as a whole.

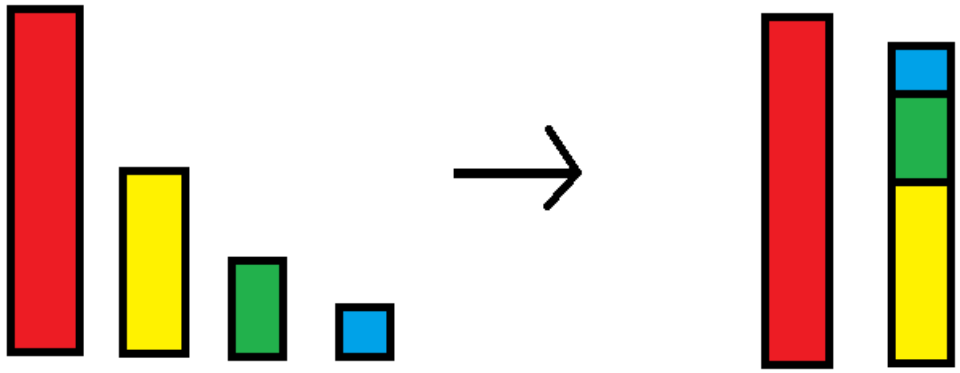

Figure 3.1: An example of clustering of classes, in this figure each bar represent the size of each class, by grouping yellow, green and blue class together we get a more balanced binary class dataset.

3.3 Choice of metric

Accuracy is the most commonly used metric when measuring the performance for classification model, but it's not applicable when dealing with imbalance data-set, for example imagine an imbalance dataset where 90% of samples are all in one majority class, a naive model classify every samples to that majority class will easily achieve 90% accuracy, but clearly, using this model would give no merit. The metric I chose to use is average sensitivity between classes. The formula of sensitivity can be describe as following formula:

$$\text{Sensitivity} = \frac{\#\text{ True Positive}}{\#\text{ True Positive} + \#\text{ False Negative}} \tag{3.1}$$

By using average sensitivity as metric, we weight each class equally, in the former example if we classify every samples to majority class, the model will have 100% sensitivity on the majority class but 0% sensitivity on every other classes, and give a low average sensitivity. Additionally, when the dataset is perfectly balanced, where number of samples in each class is same, average sensitivity would give exactly same as accuracy, as:

$$\begin{aligned} \text{Accuracy} &= \frac{\sum_{i=1}^{m} \#\text{ True Positive in } C_i}{\text{Total number of samples}} \\ &= \frac{1}{m} * \sum_{i=1}^{m} \frac{\#\text{ True Positive in } C_i}{\text{Number of samples in each class}} \\ &= \frac{1}{m} * \sum_{i=1}^{m} \text{Sensitivity of } C_i \end{aligned} \tag{3.2}$$

Due to above reason, average sensitivity should be a good metric for measuring performance of classification model on imbalanced dataset.

Chapter 4

Prepare imbalanced data set

4.1 Original dataset

The Data Set I used are MNIST handwritten digit database [22] and CIFAR10 dataset [20]. MNIST contains 28x28 grayscale images of handwritten digits in 10 classes, it has a training set of 60000 examples, a test set of 10000 examples, the classes in the training set are roughly balanced but have different sizes. I select 40000 examples from the training set as our MNIST base training set, and 10000 examples as the MNIST base validation set, where every class have the same size. I further pad images in training, validation and testing set into shape of 32x32 to suit our design of the classification model. The CIFAR-10 dataset contains 32x32 images of mutually exclusive 10 classes, namely airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck, it has a training set of 50000 examples and a testing set of 10000. The classes in training are perfectly balanced, which enable us to select from the training set, 40000 examples as our CIFAR10 base training set, and 10000 examples as CIFAR10 base validation set, where every class has the exact same size. The evaluation of our result on both datasets is on the testing set.

4.2 Forms of imbalance

Multi-class data imbalance, in reality, has different forms, different numbers of classes can be under-represented or over-represented, I want to simulate as much kind of usual imbalance as possible. The forms of imbalanced I used follows from [3], in [3], the author defines two types of imbalance, *step imbalance* and *linear imbalance*, I additional defines a new types of imbalance, *exponential imbalance*.

In *step imbalance*, a class can only be a majority class or a minority class, while all majority classes have the same size and all minority classes have the same size. A *step imbalance* can be defined by two parameters, μ and ρ, where μ is the fraction of majority class, in the range of [0, 1], can be defined as the following formula:

$$\mu = \frac{|\{i \in \{1,...,N\}\} : C_i \text{ is minority}|}{N} \tag{4.1}$$

where C_i is a set of examples in class i and N is the total number of classes. ρ defines ration of imbalance, is the maximum size of class over the minimum size of the class, can be defined as the following formula:

$$\rho = \frac{\max_i \{|C_i|\}}{\min_i \{|C_i|\}} \tag{4.2}$$

In *linear imbalance*, the size of class decrease linearly from the maximum size to the minimum size. A *linear imbalance* can be defined by one parameter ρ, which is the ratio between maximum size and minimum size using the same definition in *step imbalance*.

Exponential imbalance is similar to *linear imbalance*, but instead, the size of class decrease exponentially. Same as *linear imbalance*, a *Exponential imbalance* can be defined by one parameter, ρ.

For MNIST, I created imbalanced datasets with three forms of imbalance, where $\rho \in 10, 25, 50, 100, 250, 500, 1000, 2000, 4000$ and $\mu \in 0.2, 0.5, 0.8$, so the minimum possible size of a class in an imbalanced MNIST dataset would be 1. For CIFAR10, I created imbalanced datasets with three forms of imbalance, where $\rho \in 2, 10, 20, 40$ and $\mu \in 0.2, 0.5, 0.8$, the maximum ratio of imbalance in CIFAR10 is smaller than in MNIST, as the performance of CIFAR10 will drop much larger with a decrease in the size of the dataset, when testing in the balanced dataset, so there is no need to test larger ratio of imbalance as classifier will be unable to learn anything from minority class with larger imbalance ratio. In all of these forms of imbalance, the maximum size is always the same, which is the size of a class in the original balance dataset.

4.3 Implementation of imbalance dataset

For creating an imbalance dataset, custom datasets that are subsets of the original dataset need to be created. To do so I randomly select samples from the original dataset based on the distribution I need to create and save the indexes of the samples I created as ".npy" in "Indexs", these files can be load by "np.load()" as NumPy array. The code used for the selection of samples is in "Index generator.ipynb". Then I implemented subclasses for dataset.MINST and dataset.CIFAR10 as the custom dataset, the constructor of the custom dataset would take the path to the index file as an additional parameter, it uses this index to select a subset from the original samples it loads.

Chapter 5

Implementation of methods

5.1 Hierarchical method

5.1.1 Class taxonomy

To perform a hierarchical classification, I first need to merge classes in the dataset. My approach is to merge classes based on the size of each class as mentioned in section 3.2. The class taxonomy algorithm will cluster the classes in two parts at each stage, the size of every class in cluster 0 will be bigger than the size of every class in cluster 1, and two clusters will have sizes that are as similar as possible, each cluster will be further split until every single class has been separated. For each clustering, I create a list of indicators, to indicate classes result in cluster 0, classes result in cluster 1, and classes not involved. The lists of indicators which represent the class taxonomy are saved as ".txt" file in "Indexs folder, they can be load by "pickle.load()", using pickle package.

5.1.2 Custom dataset

For each clustering in the hierarchy, the classes in cluster 0, classes in cluster 1, and classes not involved are different, so each classifier in the hierarchy need to be trained on a different dataset. Similar to the implementation of the imbalance dataset I implemented subclasses "CustomMNIST partition" and "CustomCIFAR10 partition" for dataset.MINST and dataset.CIFAR10. The constructor of the custom dataset would take the path to the index file for the imbalanced dataset and a list of indicators as additional parameters. It will first use the index file to load the original imbalance dataset and then use the list of indicators to filter out uninvolved classes and change the labels of the involved classes to either 0 or 1.

5.1.3 Hierarchical classifier

Each classifier in a hierarchy is trained separately, but these binary classifiers need to combine into one hierarchical classifier in testing time. I implemented "hierarchical

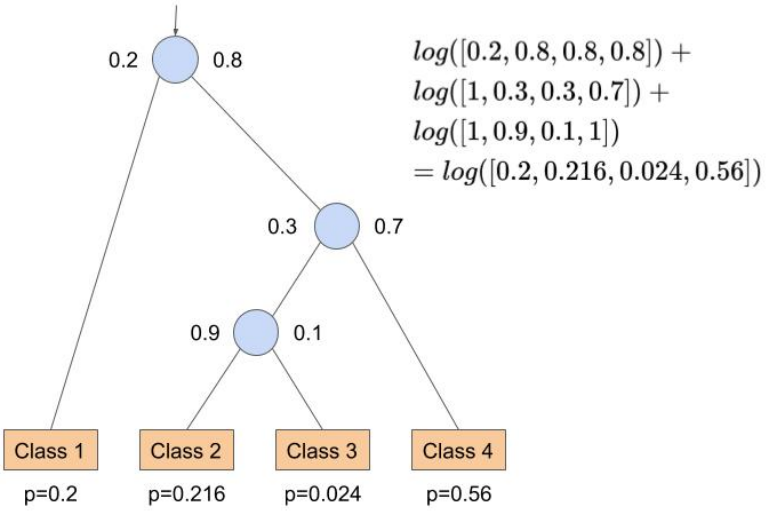

Figure 5.1: An example for hierarchical classification and the calculation of log probability.

model" as the model for the hierarchical classifier in "CustomeModule.py", its constructor takes the classifiers in the hierarchy and corresponding lists of class indicators as input. In the forward pass, each sample will go through all the classifiers exactly once, then it calculates the log probability for each class based on the classifiers' results and the lists of class indicators. Figure 5.1 shows an example of the calculation.

5.2 Random over sampling

The implementation of Random Over Sampling is simple, I created a new index file by duplicating indexes from the index file of the imbalance dataset I described in section 4.3, and the custom dataset can use this new index file to load the over-sampled dataset. The code for creating an index file for the over-sampled dataset is in "Index generator.ipynb".

5.3 Cost sensitive method

As mentioned in section 2.4, I'm using the cost-sensitive approach proposed in Salman H et. al(2017) [16]. The implementation of Cost-Sensitive method can be separated in two parts, the first one part is the implementation of the actual Cost-Sensitive loss function based on the cost matrix, the second part is the implementation of the algorithm that calculates the gradient of the cost matrix and update the cost matrix at every evaluation iteration.

5.3.1 Cost-sensitive CE loss function

The loss function is called "cost-sensitive CE loss function" in the original paper [16], it is a cost-sensitive version of the cross-entropy loss function, and suitable to use as loss function in classification problem. The cost-sensitive CE loss function can be described as:

$$L(\xi, d, o) = -\sum_n d_n \log\left(\frac{\xi_{p,n}\exp(o_n)}{\sum_k \xi_{p,k}\exp(o_k)}\right) \tag{5.1}$$

where ξ is the cost matrix, d is the target label in one-hot vector form, o is the output of the final layer of the neural network, p is the target label. In this equation we can see that the cost matrix doesn't behave like a cost for miss-classification, it behaves like weights, in fact, Salman H et. al(2017) [16] proposed this score-level costs matrix to modify the final layer output and encourage correct classification on minority class. In the cost-sensitive CE loss function, the costs will change the final probability distribution. This cost-sensitive CE loss function is only used during training time, loss function change back to normal cross-entropy loss function during evaluation and testing time, and instead of giving higher probability on minority class, the cost-sensitive CE loss function would give a higher probability to the majority class when classifying an actual minority class, the intuition behind is, by making the classification of minority class harder during training time the classifier will have a better performance when classifying minority class during evaluation and testing time. In my implementation I simply do:

$$o'_i := \log(\xi_{p,n}) + o_i \tag{5.2}$$

then when passing o' to normal cross entropy function:

$$\begin{aligned} L'(d, o') &= -\sum_n d_n \log\left(\frac{\exp(o'_n)}{\sum_k \exp(o'_k)}\right) \\ &= -\sum_n d_n \log\left(\frac{\xi_{p,n}\exp(o_n)}{\sum_k \xi_{p,k}\exp(o_k)}\right) \\ &= L(\xi, d, o) \end{aligned} \tag{5.3}$$

The code version of this formula is the "CoSenLogSoftmaxLoss" function in the "util.py" file.

5.3.2 Optimization of cost matrix

Another important aspect is the optimization of the cost matrix ξ, the author propose an algorithm that can learn the cost matrix adeptly based on class representation (H), data separability (S) and classification errors (R), H is a matrix defined by the histogram

vector h which represent the class distribution, where h_q is the portion of samples in the dataset belongs to class q, H can be described as:

$$H(p,q) = \begin{cases} max(h_p, h_q) & : p \neq q \\ h_q & : p = q \end{cases} \tag{5.4}$$

The equation above is implemented as function "matrix_H()" in the "util.py" file.

S is class-to-class separability, can be described as:

$$S(p,q) = \frac{1}{N'} \sum_i N' \frac{dist_{intraNN}(f_i)}{dist_{interNN(f_i)}} \tag{5.5}$$

In the equation above, N' is the number of samples in class p, $dist_{intraNN}(f_i)$ measure the distance of samples f_i to its nearest neighbor in class p, $dist_{interNN}(f_i)$ measure the distance of samples f_i to its nearest neighbor in class q, this equation is implemented as function "class_separability_matrix()" in the "util.py" file.

R is the normalized confusion matrix of the neural network represents the current classification error, a confusion matrix can be calculated by using "confusion_matrix" in "sklearn.metric" library.

by using matrix H, S, and R, the author defines a target matrix T which can be described as:

$$T = H \circ \exp(-\frac{(S-\mu_1)^2}{2\sigma_1^2}) \circ \exp(-\frac{(R-\mu_2)^2}{2\sigma_2^2}) \tag{5.6}$$

The equation above is implemented as function "matrix_T()" in the "util.py" file.

Using this target matrix T, the cost matrix is updated using gradient descent at every evaluation iteration based on the loss function:

$$F(\xi) = ||T - \xi||_2^2 \tag{5.7}$$

However, the author hasn't explained in detail the reason for not using the target matrix T as the cost matrix directly and hasn't explained the reason behind the design of this target matrix T.

5.4 Custom model

Finally, I implemented a structure of the neural network to be trained on the imbalanced dataset, I designed a simple Convolutional Neural Network, the architecture is showed in Table 5.1, in this table only layers that would change the size of the feature map are shown, class "Custom_05()" in "CustomModel" is the actual implementation of this architecture, it is designed to be as lite as possible while still have a good performance on the original dataset which will be shown in 6.2 Layer Width Height Depth Kernel size Stride

Layer	Width	Height	Depth	Kernel size	Stride
Input	32	32	1	-	-
Convolution	28	28	32	5	1
Max Pooling	14	14	32	2	2
Convolution	12	12	64	3	1
Max Pooling	6	6	64	2	2
Convolution	4	4	128	3	1
Max Pooling	2	2	128	2	2
Fully Connected	1	1	128	-	-
Fully Connected	1	1	10	-	-

Table 5.1: Architecture of the custom neural network

Chapter 6

Experiments

6.1 Experiments setting

As mentioned in section 4.2, there are many different imbalanced distributions created by the different combination of parameters, 45 different combinations for MNIST, 12 different combinations for CIFAR10, for each combination I permuted the class to create 5 different imbalanced distributions, and for each imbalanced distribution, I ran one experiment on each method, namely ROS, Cost-sensitive method, Hierarchical method, and baseline flat. In each experiment, the model is trained using Adam optimizer with the weight decay coefficient equals to 0, the model will be trained for 2000 iterations if the dataset is MNIST, or 20000 iterations if the dataset is CIFAR10, with batch size equals to 100. For every 100 iterations, a model is evaluated on the evaluation set which has the same type of imbalance as the training set, finally, the model with the best evaluation performance will be select as the final model for that experiment and test on a balance testing set for the final performance. For the Hierarchical method, the experiment takes 9 times longer than other experiments as it needs to train 9 classifiers to form a hierarchy and produce the final model. Additionally, for cost-sensitive learning, as it needs to learn the cost matrix automatically, A learning rate needs to be selected for cost matrix optimization, for each experiment using the cost-sensitive method, I trained 5 models under cost matrix learning rate 0.02, 0.05, 0.1, 0.2, 0.5, and select the model with the best evaluation performance as the final model. In total, $45*5*4$ experiments runs on MNIST, $12*5*4$ experiments runs on CIFAR10, and totally $45*5*16$ models being trained on MNIST, $12*5*16$ models being trained on CIFAR10.

6.2 Result

The results on the MNIST dataset are shown in figure 6.1 and the results on the CIFAR10 dataset, are shown on 6.2, in each subfigure of a different kind of distribution, we can see the comparison of different methods with change on the imbalance ratio. First, by observing the performance of flat classification on MNIST and CIFAR10, we can see that the increase of class imbalance ratio would result in a decrease in per-

formance of classification, and in some cases even worse than the performance of the under-sample dataset, which shows the class imbalance do cause harm to the classification. Second, we observe the same effect as Buda, Mateusz, et al. where "the effect of imbalance is significantly stronger for the task with higher complexity" [3].

Comparing the performance of different methods In the results, ROS method almost always had better performance than the baseline method and hierarchical method, it also has the top performance in linear unbalance and step unbalanced with μ equals to 0.2 on MNIST and in all types of distribution on CIFAR10, this showed that the ROS doesn't hurt the performance of classifier when classifying image data, the effect of over-fitting caused by ROS is not trivial, even when nearly 80% of the training samples are duplicate samples (step unbalance with $\mu = 0.8$ and $p = 4000$).

The cost-sensitive method always had better or similar performance than the baseline method, it also has the top performance in exponential distribution on MNIST and linear distribution and exponential distribution on CIFAR10, which show the effeteness of this method. But on contract to the original study [16] which proposed this method, cost-sensitive learning doesn't outperform oversampling, this is due to different ways of oversampling, in the original paper the author uses SMOTE [4] as the oversampling technique, which involves averaging existing samples to produce new samples, this is a commonly used oversampling technique but not suitable for image dataset, as averaging pixels between two images usually won't produce a sensible new image for most of the time, which causes the newly created samples to become noise in the dataset.

The hierarchical method only have better performance than the baseline flat method and the cost-sensitive method when the imbalanced ratio is high in certain distribution, this is easy to understand, as described in section 3.2, the advantage of Hierarchical method is it can rule out imbalance ratio, but it also has the disadvantage of aggregating errors, so when the imbalance ratio is low, the impact of aggregating errors is larger, which cause it to have worse performance, also the clusters in each level of the hierarchy might not perfectly balanced, although the class imbalance is reduced it still exists.

At last, the under-sampling method doesn't seem to be useless, it has the best performance when the imbalanced ratio become high in step unbalance where most of the class are minority class, I only ran the under-sampling method once for each imbalance ratio, not as a subset for each imbalance distribution (we can see the line for under-sampling in each sub-figure are the same line), this may due to randomness issue.

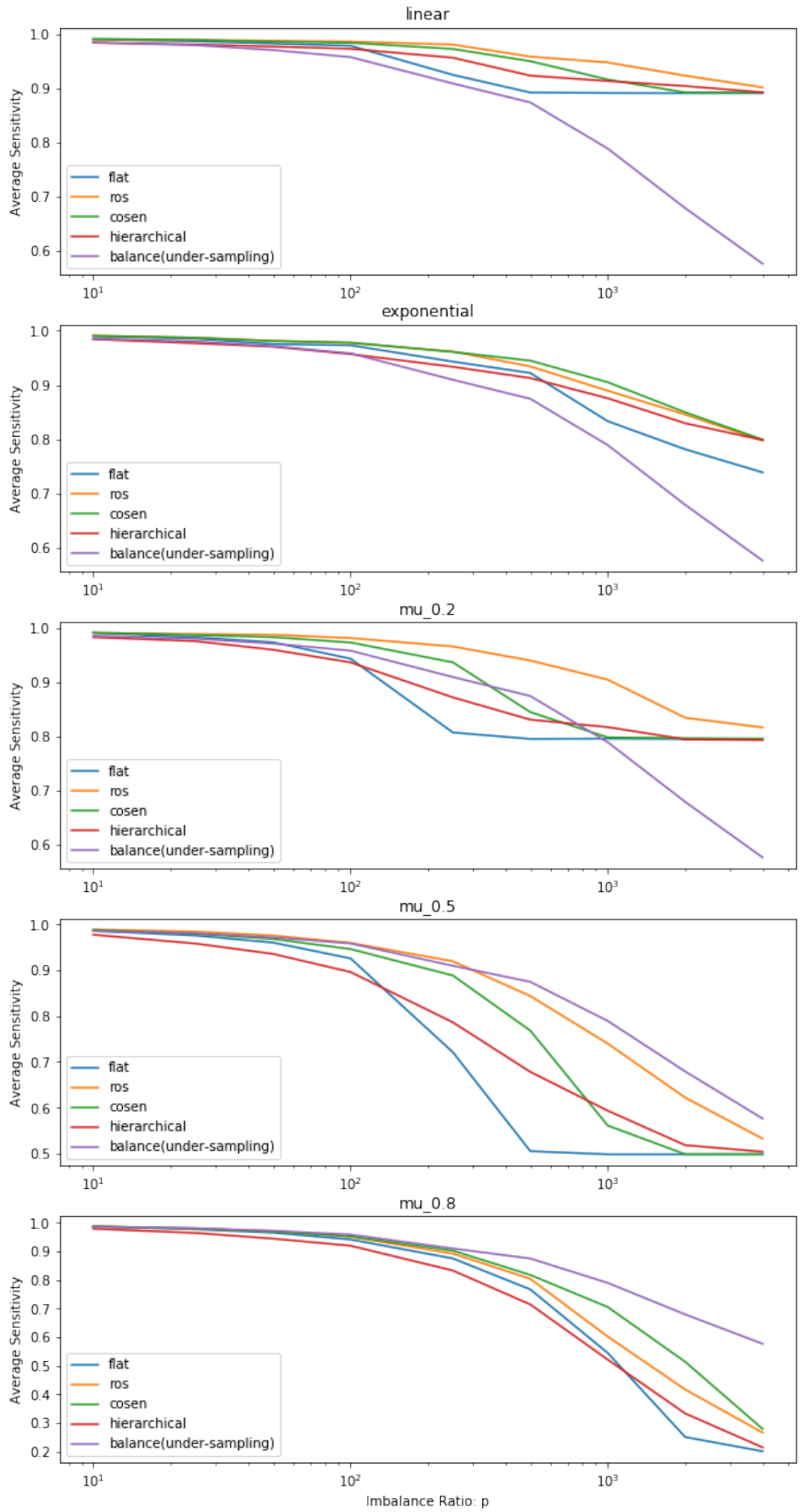

Figure 6.1: Result of each method applying to 3 different types of imbalance distribution on MNIST (mu_0.2, mu_0.5, mu_0.8 are step unbalance under different μ)

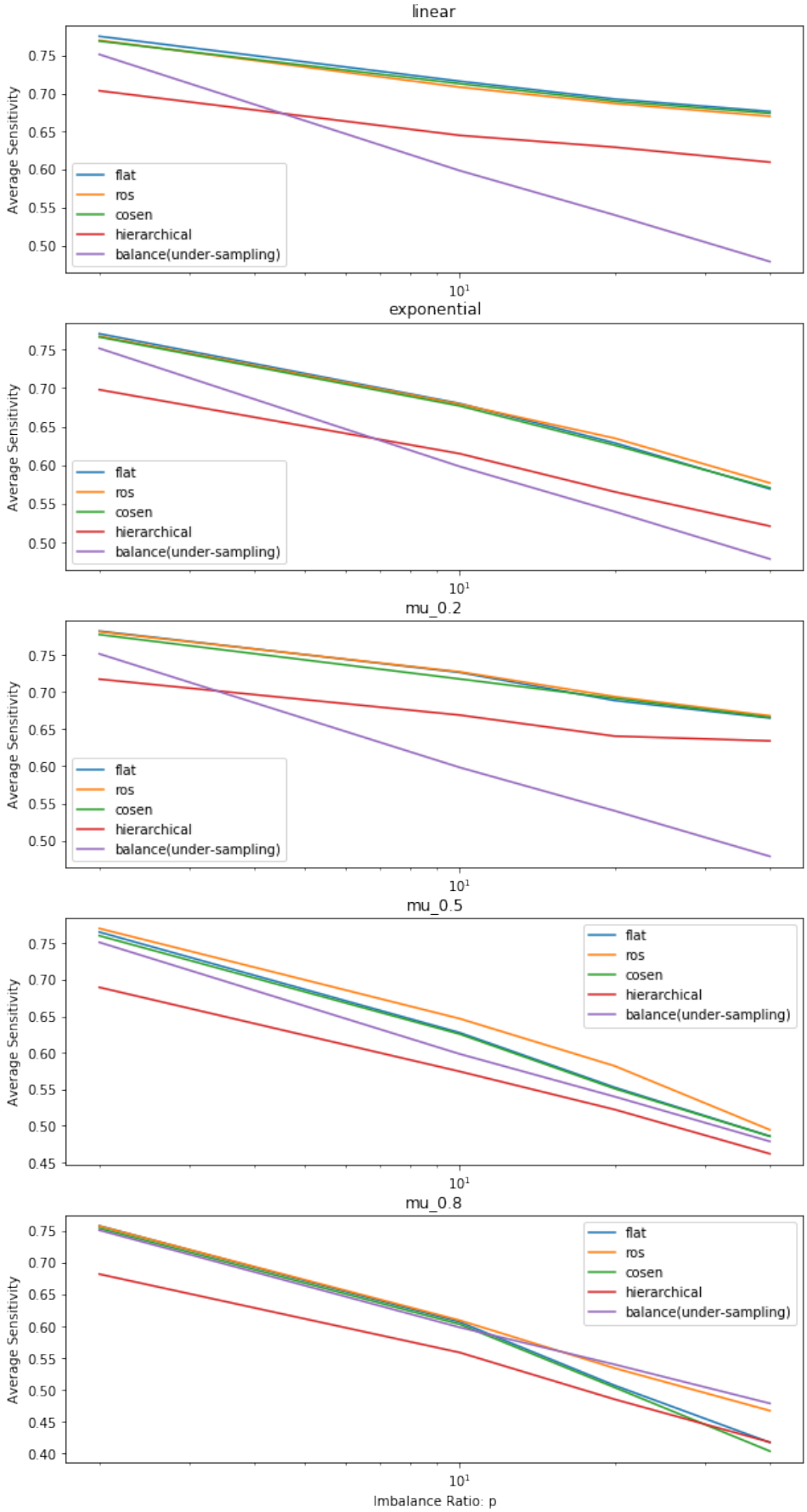

Figure 6.2: Result of each method applying to 3 different types of imbalance distribution on CIFAR10 (mu_0.2, mu_0.5, mu_0.8 are step unbalance under different μ)

Chapter 7

Conclusions and Further work

In this study, I have examined the performance of different methods, Random oversampling, Cost-sensitive learning, and Hierarchical method, on MNIST and CIFAR10 dataset with different imbalance distributions. For the effect of class imbalance, I reach the same conclusions as Buda, Mateusz, et al [3] did, where the class imbalance does cause harm to the performance of a classifier, and the effect of class imbalance is larger in the dataset with higher complexity. For the performance of different methods, I conclude:

1. The Hierarchical method I designed was outperformed by other methods, even worse than the baseline flat classification when the imbalance ratio is low, and should not be considered as a method to solve class imbalance problem.

2. Random oversampling usually have the top performance and haven't observer the effect of overfitting while using Random Over Sampling, as it is easy to implement, it should be considered as a baseline for most of image classification task that has imbalance issue.

3. Under-sampling can outperform Random oversampling when the dataset is in extreme unbalance and most of the class are minority class.

4. Cost-sensitive learning [16] doesn't outperform Random oversampling, but still should be considered as a candidate method on solving class imbalance issue, as it has the best performance on some distribution.

Class imbalance problem is really a hard problem to study, especially when more than two classes are involved. In reality, different dataset will have different complexity and different distribution, it will be impossible to find a single best method for all of them, and they're also lack of a systematic way to find the best method, many studies related to class imbalance problem involve specifically design methods for a single target dataset [37] [5], which shows the difficulty to tackle this problem in a general level. In further work, I would like to design variations of Hierarchical method, trying to reduce the effect of aggregating errors, for example, instead of binary split at each level, a ternary split will reduce the height of a tree which may result in better performance, also random oversampling and under-sampling could be combined with

Hierarchical method to obtain an optimal split of the dataset, where samples in each cluster are perfectly balanced, I believe Hierarchical method still has potential to get a better result. I would also investigate more existing methods and applies them to more complex image datasets. Lastly, the size of the majority class should also be considered as a parameter when creating imbalance distribution, “the effect of imbalance ratio is lager on datasets that are more complex” might not be a valid conclusion when considering the variation of the maximum size of the majority class.

Bibliography

[1] Cigdem Beyan and Robert Fisher. Classifying imbalanced data sets using similarity based hierarchical decomposition. *Pattern Recognition*, 48(5):1653 – 1672, 2015.

[2] Harold Borko and Myrna Bernick. Automatic document classification. *Journal of the ACM (JACM)*, 10(2):151–162, 1963.

[3] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[4] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[5] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4109–4118, 2018.

[6] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[7] Susan Dumais and Hao Chen. Hierarchical classification of web content. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 256–263, 2000.

[8] Charles Elkan. The foundations of cost-sensitive learning. 17(1):973–978, 2001.

[9] Jianping Fan, Yuli Gao, and Hangzai Luo. Hierarchical classification for automatic image annotation. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–118, 2007.

[10] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220 – 239, 2017.

[11] Mohamad H Hassoun et al. *Fundamentals of artificial neural networks*. MIT press, 1995.

[12] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, Sep. 2009.

[13] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[14] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.

[15] Justin M Johnson and Taghi M Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27, 2019.

[16] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8):3573–3587, 2017.

[17] Bartosz Krawczyk. Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence*, 5, 04 2016.

[18] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[19] Bartosz Krawczyk, Michał Woźniak, and Francisco Herrera. Weighted one-class classification for different types of minority class examples in imbalanced data. In *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 337–344. IEEE, 2014.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20:5, 2015.

[24] Lance A Liotta and William G Stetler-Stevenson. Tumor invasion and metastasis: an imbalance of positive and negative regulation. *Cancer research*, 51(18 Supplement):5054s–5059s, 1991.

[25] Jamie Ludwig. Image convolution. *Portland State University*, 2013.

[26] Krystyna Napierala and Jerzy Stefanowski. Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, 46(3):563–597, 2016.

[27] Danh V Nguyen and David M Rocke. Tumor classification by partial least squares using microarray gene expression data. *Bioinformatics*, 18(1):39–50, 2002.

[28] Michael A Nielsen. *Neural networks and deep learning*, volume 2018. Determination press San Francisco, CA, USA:, 2015.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[30] Réjean Plamondon and Sargur N Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1):63–84, 2000.

[31] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[32] Carlos N Silla and Alex A Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.

[33] Carlos N Silla and Alex A Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.

[34] Jerzy Stefanowski. Dealing with data difficulty factors while learning from imbalanced data. pages 333–363, 2016.

[35] Nenad Tomašev and Dunja Mladenić. Class imbalance and the curse of minority hubs. *Knowledge-Based Systems*, 53:157–172, 2013.

[36] Aditya Vailaya, Mário Figueiredo, Anil Jain, and Hong Jiang Zhang. Content-based hierarchical classification of vacation images. In *Proceedings IEEE International Conference on Multimedia Computing and Systems*, volume 1, pages 518–523. IEEE, 1999.

[37] Niloufar Zarinabad, Martin Wilson, Simrandip K Gill, Karen A Manias, Nigel P Davies, and Andrew C Peet. Multiclass imbalance learning: Improving classification of pediatric brain tumors from magnetic resonance spectroscopy. *Magnetic resonance in medicine*, 77(6):2114–2124, 2017.