FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Data Augmentation for Ordinal Data

Airton Matos Andrade Fernandes Tiago



Mestrado em Visão por Computador (MVCOMP)

Supervisors:
Prof. Ricardo P. M. Cruz (PhD)
Prof. Jaime S. Cardoso (PhD)

February 21, 2024

Data Augmentation for Ordinal Data

Airton Matos Andrade Fernandes Tiago

Mestrado em Visão por Computador (MVCOMP)

Approved in oral examination by the committee:

Chair: Prof. Miguel Velhote
External Examiner: Prof. Pedro Antonio Gutiérrez
Supervisor: Prof. Ricardo Cruz

February 21, 2024

Resumo

A regressão ordinal é usada quando os rótulos de classe têm classificações inerentes. Funções de perda (loss functions) tradicionais como Mean Squared Error (MSE) e Cross-Entropy (CE) muitas vezes se mostram insuficientes, pois falham em refletir essa ordinalidade. Este trabalho introduz uma abordagem inovadora que aplica técnicas de aumento de dados usando distribuição unimodal e Ordinal Encoding loss para aprimorar os métodos tradicionais de Mixup. Mixup é uma técnica que aumenta o conjunto de dados combinando duas imagens de classes diferentes. Nossa proposta combina todas as classes usando pesos ordinais. Para avaliar a eficácia da nossa abordagem, conduzimos uma série abrangente de 64 combinações experimentais de diferentes métodos e funções de loss em dois conjuntos de dados de imagem ordinais distintos. Os experimentos comparam vários métodos incluindo: (i) baseline sem aumento mixup, (ii) o mixup original, (iii) um método de mixup ordinal da literatura que restringe o mixup a classes adjacentes, e (iv) nosso método de mixup que combina imagens de todas as classes usando ponderação exponencial unimodal. Além disso, também exploramos diferentes combinações de τ para controlar a variância do método exponencial. Os resultados indicam um aprimoramento significativo no desempenho do modelo com nossa abordagem, evidenciado por uma redução de 25,92% no Mean Absolute Error (MAE) no conjunto de dados SMEAR2005 em comparação com o Mixup clássico. Em contraste, o conjunto de dados Seeing Through Fog (FogChambers) mostrou resultados ligeiramente melhores do que o Mixup clássico. Esta análise experimental extensiva sublinha o potencial do nosso método e prepara o terreno para futuras pesquisas explorando conjuntos de dados diversos e técnicas alternativas de aumento como Cutmix.

Palavras-chave: ordinal classification, ordinal regression, data augmentation, mixup, unimodal distributions, artificial neural networks, computer vision.

Abstract

Ordinal regression is used when class labels have inherent rankings. Traditional loss functions like Mean Squared Error (MSE) and Cross-Entropy (CE) often prove insufficient as they fail to reflect this ordinality. This work introduces a novel approach that applies data augmentation techniques using unimodal distribution and Ordinal Encoding loss to enhance traditional Mixup methods. Mixup is a technique that augments the dataset by combining two images of different classes. Our proposal combines all the classes using ordinal weights. To assess the effectiveness of our approach, we conducted a comprehensive series of 64 experimental combinations of different methods and loss functions on two distinct ordinal image datasets. The experiments compare various methods including: (i) a baseline with no mixup augmentation, (ii) the original mixup, (iii) an ordinal mixup method from the literature that restricts mixup to adjacent classes, and (iv) our mixup method that combines images of all classes using unimodal exponential weighting. Apart from that we also explored different τ combinations to control the variance of the exponential method. The results indicate a significant enhancement in model performance with our approach, evidenced by a 25.92% reduction in Mean Absolute Error (MAE) on the SMEAR2005 dataset compared to the classic Mixup. In contrast, the Seeing Through Fog (FogChambers) dataset showed slightly better results than the classic Mixup. This extensive experimental analysis underlines the potential of our novel method and sets the stage for future research exploring diverse datasets and alternative augmentation techniques like Cutmix.

Keywords: ordinal classification, ordinal regression, data augmentation, mixup, unimodal distributions, artificial neural networks, computer vision.

Acknowledgments

I would like to express my deepest appreciation to my supervisor, Professor Dr. Ricardo Cruz, and my co-supervisor, Professor Dr. Jaime Cardoso, for their invaluable guidance and support throughout my Master's Thesis. Their expertise, insights, and encouragement were pivotal in shaping both the direction and success of my research. Additionally, I would like to extend my sincere thanks to all the coordinators of the IMCV Master's program, especially Jaime Cardoso, Ana Maria Mendonça, Miguel Velhote, Victor Brea, and Luís Filipe Teixeira. Their efforts in ensuring a smooth and supportive educational journey have been invaluable. Moreover, a special mention goes to my wife, Fabiana Behrendt Pereira Fernandes Tiago, whose unwavering support, understanding, and love have been my cornerstone throughout all the challenges I faced during this journey.

Project ATLAS provided the computing resources necessary for the development of this thesis. This work was supported by Portuguese National Funds, through AICEP, E. P. E., and the Innovation and Digital Transition Program (COMPETE2030). Project ATLAS – Trusted Autonomous Navigation, with Funding Reference 01/RPA/2022-C679908640-00009887.

Airton Matos Andrade Fernandes Tiago

"Whether you think you can, or you think you can't
either way you're right "

Henry Ford

Contents

1 Introduction **1**

2 Related Work **3**
2.1 Ordinal Methods . . . 3
2.2 Mixup . . . 4
2.3 Mixup Variants . . . 6
2.4 Mixup for Ordinal Regression . . . 7

3 Proposal **9**

4 Experimental Details **11**
4.1 Methodology . . . 11
4.1.1 Nothing or No Mixup . . . 11
4.1.2 Classic Mixup . . . 12
4.1.3 Adjacent Mixup . . . 12
4.1.4 Exponential Mixup . . . 12
4.1.5 Others . . . 13
4.2 Mixup Adaptation . . . 13
4.3 Data Preparation Methods . . . 16
4.4 Datasets . . . 16
4.4.1 SMEAR2005 . . . 17
4.4.2 Seeing Through Fog (FogChambers) . . . 18
4.5 Model Architecture and Hardware Enhancement . . . 20
4.6 Losses . . . 20
4.7 Optimization Method Selection . . . 21
4.8 Training methodology . . . 21
4.9 Data augmentation . . . 22
4.10 Data Loaders in Model Training and Testing . . . 22
4.11 Evaluation Metrics . . . 23
4.12 Logs . . . 24

5 Results and Discussion **25**
5.1 Analysis of Experiment Results . . . 26
5.1.1 Analysis of SMEAR2005 . . . 27
5.1.2 Analysis of FogChambers . . . 27
5.2 General Insights . . . 28

6 Conclusion **31**

References **33**

List of Figures

2.1 Example of mixup applied to SMEAR2005 5

4.1 Impact of different tau in exponential function 13
4.2 Example of SMEAR2005 classes 17
4.3 Class distribution for SMEAR2005. 18
4.4 Example of FogChambers classes 19
4.5 Class distribution for FogChambers. 20

5.1 Confusion matrices 29

List of Tables

4.1 Example of Lambda values for different classes and methods based on FogChambers dataset 14
4.2 Datasets used for ordinal experimentation. 16

5.1 Analysis of experiments for different methods 26
5.2 Analysis of different taus for exponential method 26

Chapter 1

Introduction

With more than 384.000 results returned when searching for "data augmentation" in Google Scholar, it is feasible to state that data augmentation has been broadly researched worldwide. To give a perspective, when searching for "computer vision" we get approximately 2.7 million results, therefore "data augmentation" represents around 14% when comparing the total search for "computer vision." It is a fact that data augmentation is a growing field with a representative amount of searches when compared to computer vision. Although we do not need to separate them from each other, it is important to note that data augmentation can be applied to different areas of machine learning. Being this work focused on computer vision, our attention will be primarily on this relation.

Data augmentation techniques significantly influence computer vision, mainly through their contribution to enhancing model generalization. This enhancement is achieved by diversifying data samples, which involves applying different transformations to the original dataset. A common challenge in computer vision projects is the scarcity of data available for training machine learning models. Consequently, it is vital to comprehend how to optimize the use of existing data and, when necessary, generate synthetic data for the development of these models.

One important challenge from the perspective of machine learning in the computer vision field, is the ability to enhance the machine learning model generalization to new, unseen instances, often due to a scarcity of training data. Therefore, when there is a generalization in the model's ability to learn, this is what defines the ability to predict new cases that were not seen during the training. Model generalization, which is the model's ability to perform well on unseen data, is an important aspect of model performance. To achieve this, we need to have careful consideration to prevent overfitting, which occurs when a model learns to replicate the training data too closely, resulting in perfect performance in training but poor and undesired performance on test and validation sets. For this matter, various strategies can be adopted to improve generalization capabilities, and certainly, data augmentation can be considered as a particularly effective approach. Through image transformations, we not only increase the number of training samples but also improve our understanding of the impact of these transformations on model learning.

Ordinal regression, a crucial yet challenging task in machine learning, involves predicting a

discrete variable $Y \in \{C_1, C_2, \ldots, C_k\}$, where a well-defined ranking exists such as in regression, $C_1 \prec C_2 \prec \ldots \prec C_k$. Unlike cardinal variables, ordinal regression does not assume quantitative differences or distances between classes. For instance, consider the grading of diabetic retinopathy, where patients are classified as healthy (0), mild (1), moderate (2), severe (3), or proliferative (4). Another example is the cervical cancer risk classification [11], which is one of the datasets to be used in the experiments, where patients are usually classified into 7 classes, ranging from normal to carcinoma. These problems are commonly explored through binary classification (normal or abnormal), however, the stage of the disease could also be important to make the patient able to start a recovery process before it is too late. Therefore, there is not only the need to predict the disease's severity but also to accurately determine the order of these categories.

This work aims to investigate prominent data augmentation methods, such as Mixup [26] and variants with a key focus on ensuring the preservation of the inherent ordinality of labels during model development by adapting these data augmentation techniques.

In the next chapter, related work is reviewed, delving into the specifics of the data augmentation methods by showing the methodology and their potential adaptations for our research, besides possible variants and ordinal regression. Chapters 3 and 4 present our proposal and the modifications made to the original works that serve as our baseline. Subsequent chapters detail various experiments conducted to evaluate the effectiveness of our approach against established benchmarks. Chapter 5 discusses the results and offers an analysis. The work concludes with Chapter 6, summarizing our findings and implications.

Chapter 2

Related Work

Our primary objective is to investigate the adaptability of data augmentation techniques in the context of modeling ordinal data. To this end, we are carrying out a comprehensive literature review, the details of which will be presented in this chapter. This review will cover both the selection of data augmentation methods that could be compatible with various loss functions and the consideration of the intrinsic ranking present in ordinal regression challenges. We begin with an examination of prevalent loss functions utilized in this domain, followed by a discussion on data augmentation strategies related to Mixup and Cutmix algorithms.

2.1 Ordinal Methods

Pattern recognition tasks often require the classification of examples into categories that follow a natural sequence or hierarchy [18]. For tasks that involve classifying instances into naturally ordered categories, the Mean Squared Error (MSE), represented as $\text{MSE}(y,\hat{y}) = \sum(y-\hat{y})^2$, is useful because it assigns greater penalties for predictions that are further from the actually class, which is often beneficial. However, this benefit comes with the drawback of assuming the distances between classes, for example, it assumes that proliferative (4) is twice as bad as moderate – which is undesirable. Alternatively, Cross-Entropy (CE), represented by $\text{CE}(y,\hat{p}) = -\sum_k \mathbb{1}(y=k)\log(\hat{p}_k)$, where the model outputs $\hat{p}_k = \text{P}(y=k)$ for each class k, performs well in general but does not differentiate between the ordered nature of the classes in an ordinal dataset.

Consequently, to address these shortcomings, certain researchers have introduced ordinal losses that enhance the ranking capability of models. Additionally, different researchers have recommended the use of specialized activation functions that inherently enforce an ordinal ranking on model outputs. This discussion will be further developed as follows where we include specific examples of these ordinal losses. To provide a background that will further support our proposal, we will follow with a brief explanation of ordinal losses that could enhance the ranking capability of models as explored by [8].

Ordinal Encoding [8], is given by $\text{OE}(y,\hat{c}) = -\sum_k \mathbb{1}(y>k)\log(\hat{c}_k) + \mathbb{1}(y\leq k)\log(1-\hat{c}_k)$, where the model outputs the inverse cumulative distribution $\hat{c}_k = \text{P}(y>k)$. Since the model output

is in the cumulative probability space, then the model only outputs $K-1$ classes – for example, if there are 3 classes, then $c_1 = (0,0)$, $c_2 = (1,0)$, and $c_3 = (1,1)$. By promoting a cumulative distribution, the loss is promoting a ranking between the classes. It is worth noting that OE is widely used in ordinal regression problems.

Another loss is CO2 [3], $\text{CO2}(y,\hat{p}) = \text{CE}(y,\hat{p}) + \sum_k \mathbb{1}(k \leq y)\max(0, \delta + \hat{p}_k - \hat{p}_{k+1}) + \mathbb{1}(k \geq y)\max(0, \delta + \hat{p}_{k+1} - \hat{p}_k)$, which operates directly in the probability space $\hat{p}_k = f(\cdot)$ and promotes ranking by penalizing incorrect differences between adjacent probability classes. That is, if the true class is $y = 2$ then probabilities below $y < 2$ should be decreasing and probabilities above $y > 2$ should be increasing. The hyperparameter δ promotes a minimum difference between adjacent classes.

Yet another loss is Unimodal Regularization [16], $\text{URL}(y,\hat{p}) = -\sum_k w_k \log(\hat{p}_k)$, where each output probability is weighted by w_k. This is a form of soft encoding. For example, for $y = 3$, then $w = (0,0,1,0)$ represents cross-entropy. But the authors propose such values as $w = (0.05, 0.2, 0.55, 0.2)$ to promote ordinality. The class y is transformed to produce smooth unimodal distributions. Such transformations can include the Poisson or Binomial distributions, or the authors also propose a transformation such as $\phi(y,k) = \exp(-|y-k|/\tau)$.

As previously mentioned different authors have recommended the use of specialized activation functions that inherently enforce an ordinal ranking on model outputs [8]. Examples of activation functions include Binomial Unimodal and UnimodalNet as follows. Binomial Unimodal [18] where the model predicts a single value: the binomial distribution parameter p, for each observation. From this value, the probability distribution is created using the binomial formula, $P(y) = C(n,k-1)p^{k-1}(1-p)^{n-k-1}$. This ensures that the output is always ordinal. UnimodalNet [7] where the model predicts a vector with length $K, \hat{y}_k = f(\cdot)$, that is then re-arranged to produce an ordinal probability distribution, $\hat{p}_k = \text{softmax}(\min(\hat{y}'_k, \hat{y}''_k))$, where $\hat{y}'_k$ is an increasing line $\hat{y}'_k = \hat{y}_{k+1} + \hat{y}_k$ and $\hat{y}''_k$ is a decreasing line $\hat{y}''_k = \hat{y}_{x-k+1} + \hat{y}_{x-k}$ (with $x = \#$ classes).

Moreover, an approach that was created for ordinal regression but indeed does not apply to Mixup, was the work developed by [16] which described a novel approach for dealing with Deep Neural Network (DNN) based ordinal regression problems. In their approach, there is a reformulation in the neurons in the last layer of the network and the softmax function is modified. This reformulation is specifically designed to address the ordinal nature of the class labels, ensuring that the resulting cumulative probabilities are monotonically decreasing, which is a desirable property in ordinal regression tasks. The Unimodal Label Regularization was also introduced as a solution for the noisy label. This approach was used as an inspiration for our proposed unimodal exponential method.

2.2 Mixup

The Mixup algorithm [26] is a simple data augmentation technique that trains a neural network on convex combinations of pairs of examples and their labels. Therefore, the idea is to regularize the neural network to favor simple linear behavior in between training examples. Given a pair of

examples (x_i, y_i) and (x_j, y_j), where x represents the input data and y the labels, Mixup constructs a new example $(\tilde{x}, \tilde{y})$ as follows:

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda) x_j, \\ \tilde{y} &= \lambda y_i + (1-\lambda) y_j, \end{aligned} \tag{2.1}$$

where $\lambda \in [0,1]$ is a mixing coefficient drawn from a Beta distribution, typically $Beta(\alpha, \alpha)$, for a hyperparameter $\alpha > 0$.



Figure 2.1: Example of mixup applied to SMEAR2005

Figure 2.1 presents an example of applying Mixup to the SMEAR2005 dataset [11], which was also utilized in our experiments. To generate this image, a lambda value of 0.3 was specified, indicating that the composition consists of 30% from class (0) and 70% from class (3). Consequently, this approach appears to be unsuitable for ordinal classes, prompting us to investigate it further.

In their preliminary experiments, they discovered that blending three or more examples with weights sourced from a Dirichlet distribution did not enhance the outcomes but rather led to a higher computational burden. Therefore, a single data loader for a minibatch was used, followed by the application of Mixup after a random shuffle of this batch, a strategy that proved equally effective while lessening input/output demands. Additionally, the strategy of combining inputs solely when they shared identical labels did not yield the performance enhancements typically associated with Mixup.

With all of that, we can understand Mixup as a form of data augmentation that has the goal of encouraging the model to behave linearly in between training examples. This idea is supported by the reduction of oscillations when predicting samples that were not part of the training set, which leads to an increase in model generalization. To evaluate its effectiveness, they trained and evaluated different model architectures on top of the ImageNet 2012, CIFAR10, and CIFAR100 datasets.

It is worth noting that they conducted their experiments using a single Nvidia Tesla P100 GPU and PyTorch [17] as the main framework for 200 epochs on the training set, and using 128 examples per minibatch. The learning rate started at 0.1 and decreased after epochs 100 and 150 by dividing by 10. They also set the weight decay as 10^{-4}.

Apart from image data, they also tested the model in speech and tabular data, which shows that Mixup is indeed data-agnostic. However, since it is not related to what this work is researching we will not explore these experiments.

2.3 Mixup Variants

After the Mixup algorithm [26] creation, several authors proposed Mixup variants such as Manifold Mixup [22], which extends the Mixup concept beyond input space, applying interpolation at the level of hidden representations in a neural network. Therefore, this helps in learning much more robust representations and encourages the model to generalize better to unseen data. Puzzle Mix was created by [14] by changing the original concept of Mixup to focus on optimizing the mixing policy while still respecting the underlying local statistics of the data. Therefore, the authors used saliency information to decide which parts of images to mix, resulting in more effective training and better model performance.

Similarly to Mixup, Cutmix [24] is also a data augmentation strategy that also works with pairs of images and was developed to improve the network generalization by cutting patches and pasting them among training images where the ground truth labels were also mixed proportionally to the area of the patches. This indeed retains the regularization effect of regional dropout, and according to their study, it effectively outperforms other state-of-the-art augmentation models such as Cutout [10] and Mixup[26].

The authors argue that to prevent Deep Convolutional Neural Networks (CNNs) from focusing too much on a small part of an input image or even a small set of intermediate activations, different methods were proposed such as Dropout among others. However, usually, this problem is solved by dropping a small portion of the image, or filling it with random noise, which certainly reduces the amount of informative pixels on training images. This led to the creation of Cutmix, which aims to neither drop part of the input image, nor introduce noise on it, but rather include a patch from another image that could be potentially more useful for the training process due to the valuable information and part of the training data. Therefore, their idea was to utilize the parts that were previously deleted by other techniques, to enhance the model's generalization.

One similarity that is shared between Cutmix and Mixup, is the fact that both approaches combine two samples, where the ground truth label of the new sample is given by the linear interpolation of one-hot labels. However, the authors argue that Mixup could confuse the model because the result after applying Mixup could be ambiguous and unnatural.

The Cutmix algorithm works by shuffling the order of the minibatch input and the target along the first axis of the tensors. Then, the lambda and the cropping region (x_1, x_2, y_1, y_2) are obtained. After that, the mixing process starts, where the inputs are mixed by replacing the cropping region of the input with the region of input 2. The target label is also mixed by interpolating. Therefore, the labels of the original and the patched image are mixed in proportion to the area of the patch and the whole image. Finally, the augmented images are fed into the model, and the loss is computed

using the mixed labels. The model is then updated based on this loss. Therefore, the Cutmix algorithm can be summarized in the following steps.

- **Mini-Batch Processing**: In each training iteration, a batch of images and their labels are selected.
- **Shuffling and Patching**: This batch is shuffled to create a new set of images and labels. A patch from a shuffled image is cut and pasted onto the original image.
- **Randomization**: The size and location of the patch are determined randomly, based on a mixing ratio (lambda) chosen from a uniform distribution.
- **Label Mixing**: The labels of the original and the patched image are mixed in proportion to the area of the patch and the whole image.
- **Model Training**: The augmented images are fed into the model, and the loss is computed using the mixed labels. The model is then updated based on this loss.

After the development of Cutmix, new variants arose such as the Attentive Cutmix [23] which is a variation from the original Cutmix adapted to include attention mechanisms within the Mixup process. Therefore, their method pays attention to selectively mixing features of the input images, aiming to preserve important features while still benefiting from the regularization effect of Mixup.

2.4 Mixup for Ordinal Regression

For ordinal regression applied along with Mixup, there are already some studies published in the last years, here we will provide a short overview of the overall idea behind these studies.

A related work [20] suggested deep learning models for the analysis of the quality of collaboration in elementary and secondary school classrooms. This study addressed two main challenges: the lack of training data and the high skewness in the class labels. To solve these obstacles, the authors of the paper introduced a controlled form of the Mixup method. Considering the classic mixup technique, the random pairing of two samples and their interpolation by a coefficient λ, drawn from the Beta distribution. Their controlled mixup variant was developed to maintain the class distribution imbalance in the synthesized data. To overcome this limitation and produce a predetermined number of samples for a given category, they suggested a two-fold adjustment of the original procedure. Moreover, the domain of λ is restricted within $[\tau, 1]$ where τ is a lower bound threshold. Through this 'tuning' they guarantee some level of originality in the augmented samples. Therefore, to expand the dataset with more instances of a specific class they intentionally couple samples from this "primary class" with samples from its adjacent or neighboring classes. This pairing strategy is represented as $(primary-class, [adjacent-class-1, adjacent-class-2])$, where 'primary-class' signifies the class selected for the augmentation and 'adjacent-class-1' and 'adjacent-class-2' indicates its immediate neighbors. This method aims to preferentially edit the dataset focusing on classes requiring additional representation. With this, they emphasized the

order of label space in terms of an assessment of group coherence. Furthermore, another interesting point elaborated in their work was the application of the cross-entropy loss ordinal function used in conjunction with Mixup.

[21] introduces a method for predicting pre-operative survival in diffuse glioma patients, incorporating a tumor subtyping network into the survival prediction model. This network provides vital tumor-type-related features for accurate prognosis. A key innovation in their approach is the ordinal manifold Mixup technique, which aligns the feature distribution of tumor types according to their risk grades, ensuring consistent and meaningful feature augmentation. This technique significantly enhances the original manifold Mixup by addressing multi-class classification challenges and ensuring label-feature consistency.

Chapter 3

Proposal

From our literature review, we did not find research papers that utilized the same approach as us in terms of using an adapted version of the Mixup data in the same selected datasets. This is an interesting point to be further utilized as an initial benchmark for our proposed approach.

In our research, we focused on modifying the original Mixup algorithm to extend its application to ordinal data, where the ranking of the labels should persist in these data types. The proposed adaptation involves integrating the Ordinal Encoding loss function to maintain the ordinal nature of the data. Furthermore, we also used Cross-entropy (CE) as a benchmark for evaluating the model's performance by comparing it with the original Mixup algorithm.

Our proposed method is inspired by the previously mentioned Unimodal Regularization Loss (URL), except that this loss focuses only on soft-encoding the Y target variable, but we also want to ordinally augment the X input variable. Therefore, we focused on adapting the Mixup code to accept different methods, and in the implementation of the unimodal exponential method. As we saw before, Mixup is a popular data augmentation technique that takes two images X_1 and X_2, with respective labels $Y_1 = (1,0,0)$ and $Y_2 = (0,0,1)$, and produces a new image $X_3 = 0.25X_1 + 0.75X_2$ with the label $Y_3 = (0.25,0,0.75)$. However, this approach in its original form is not suitable for ordinal labels. For that reason, this data augmentation technique could be adapted to be more suited to ordinal methods. For example, by producing $X_4 = 0.125X_1 + 0.75X_2 + 0.125X_3$ with label $Y_4 = (0.125,0.75,0.125)$.

The core of our implementation is a blend of standard and specialized Python libraries, with PyTorch [17] serving as the foundational framework for constructing and training our models. The structure of our code is organized to manage data loading, model construction, and the various phases of training and testing. By utilizing argparse for command-line argument parsing, we enhanced the flexibility of our experimental framework, allowing for a versatile and adaptable approach to experimentation. Indeed, this was also useful to perform multiple experiments in parallel, as you will see more in Chapter 5. The ability to run multiple experiments at the same time is essential when you want to optimize your resources and time available.

It is important to note that the main idea of this research was to adapt the original Mixup code and iterate through different possibilities so then we could see how the usage of different

augmentation methods would impact the performance in different settings and datasets.

The following chapter will go deeper into details to explore all the experimentation details. Each section will cover a different perspective of our implementation.

Chapter 4

Experimental Details

Within this section, you will find all the details of the experimental process that we have taken to develop this work.

4.1 Methodology

A crucial element of our study was the exploration and implementation of Mixup variants. These techniques, which involve creating training images by blending multiple images and their labels, were expanded to include not only standard methods like classic and adjacent class mixing but also more sophisticated approaches that employ unimodal exponential distributions. This exploration allowed us to compare the efficacy of these methods against traditional approaches and a non-Mixup baseline. Therefore, these techniques were carefully implemented and selected to expand a little beyond the traditional Mixup methods. The chosen methods were based on related works and also by imagining what would be interesting to consider when analyzing different scenarios within our experiments. One possibility that we tried and decided to discard was the Poisson unimodal distribution. This was done because we realized that apart from being more complex to tune, it was also not performing as well as we thought it would be, therefore, it might be useful to try it again in future studies.

Presented below is a brief description of each method implemented in our research.

4.1.1 Nothing or No Mixup

As a control, we include experiments without using the Mixup augmentation. This comparison will provide clarity on the actual impact of Mixup techniques on the learning process and final model performance with ordinal data. By adding this method, we can try to identify what is the realistic impact of our experiments when compared to not applying any Mixup strategy, but just training the model as it would be. Here it is valid to remember that even though we did not use Mixup for this method, in the experiments section you will notice that even for this case we also tried to apply other augmentation as will be explained later.

4.1.2 Classic Mixup

The canonical version of the Mixup serves as one of our baselines because it was the basis of our work. As previously explained, it consists of a linear interpolation between pairs of sample images and their corresponding labels. This method was crucial for establishing a benchmark against which the efficacy of other, more complex Mixup strategies, can be measured. For this part, we did not change anything in the code and kept the same parameters utilized by [26].

4.1.3 Adjacent Mixup

Recognizing the unique nature of ordinal data, the Adjacent Mixup method was designed to focus on adjacent classes. By blending data from consecutive ordinal classes, this method aims to improve the model's sensitivity to subtle differences between closely ranked categories, a key aspect when dealing with ordinal data. This was one of the approaches also tested by Mixup authors and indeed can serve as one of the possible methods to test against ordinal data as well.

4.1.4 Exponential Mixup

The Mixup Exponential approach introduces a different combination strategy where we generate an unimodal exponential distribution for imposing unimodality and act as the weights of each sample image before the blending process. This method is expected to reveal complex, non-linear relationships within the ordinal data, potentially uncovering deeper insights into the nature of the rankings and how they might be best represented through the Mixup process. The initial idea of proposing this method along with Mixup was based on the research carried out by [16] where they proposed an unimodal label regularization strategy as mentioned before.

Figure 4.1 shows how the exponential function as proposed in Section 2 (4.1) changes for different values of tau (τ). Where each colored line is related to a different tau, which we defined as being one of the following options [0.15, 0.45, 0.6, 1.0)]. The possible values for tau were defined by following the criteria of having initial values that could range from small to large. This was mainly due to the nature of the exponential curve, where we would like to test different values of τ to get different behaviors. As we can see in Figure 4.1, the tau parameter significantly influences the variance of the output distribution. Therefore, lower tau values produce steeper curves, and higher tau values result in flatter, more spread-out curves. This was interesting for our study because we evaluated how the different values of taus impact the data augmentation performance.

$$\exp(\ell, k) = \text{softmax}\left(-\frac{|\ell - k|}{\tau}\right) \tag{4.1}$$

Equation 2 shows how exponential is calculated, and in our case, ℓ will be the lambdas from Mixup, and k is the class number. For simplicity, we do not use the τ symbol instead of just refer to it as tau. This function is inspired by the encoding transformation function from Unimodal Regularization [16], previously mentioned in Section 2.1.

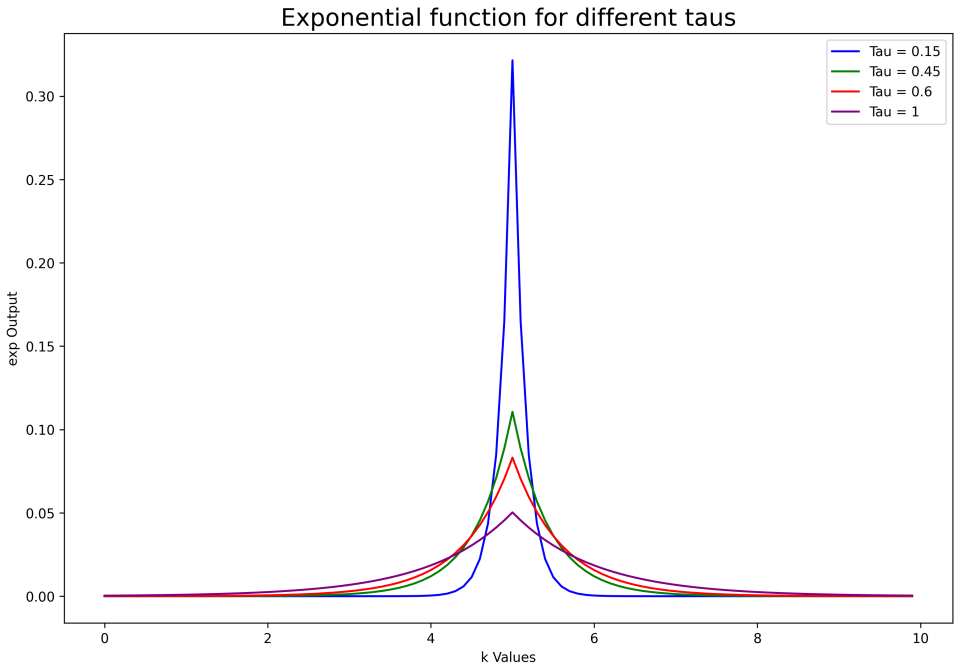


Figure 4.1: Impact of different tau in exponential function

When the tau is smaller, like the 0.15 represented in Figure 4.1, we have a sharper peak, or narrower distribution that is usually centered around the ℓ value. Therefore, in this case, the function becomes more sensitive for differences between k and ℓ, which results in lower variance, and also means that the probability will certainly be more concentrated around ℓ. On the other hand, for larger values of tau which are represented in purple in Figure 4.1, we will have a broader and smoother distribution which translates into a higher variance. Therefore, in this case, it will be less sensitive to the differences between k and ℓ.

4.1.5 Others

As previously mentioned, at the beginning of our research we thought about including other methods as well such as the unimodal poisson distribution and uniform distribution, but then we ended up discarding them in this report, since we already have covered other methods that provided better results and possibilities. The main idea of considering them was to include different statistical distributions to generate more nuanced and data-specific Mixup variants.

4.2 Mixup Adaptation

To make the different experimentations, we adapted the original code of Mixup-cifar10 [1] and refactored their function called Mixup data. This function acts as one of the most important parts of our code and is a slightly different implementation of the Mixup technique, but uses most of the same code. We designed it to work with various methods of blending images and their corresponding labels. The main idea of adapting the original code was to accommodate various

methods of blending images and their corresponding labels inside the same function. Therefore, our solution is primarily designed to initialize key variables, such as the number of classes (K) and the number of images in a batch (N), and to create a lambda matrix with dimensions `(N, K)`. It is worth noting that we implemented a matrix to facilitate the logic, but could also be just a vector. This matrix, initially filled with zeros, is pivotal in determining the lambda values for each image in the batch, which dictate the blending ratios that are used to combine different images from different classes into only one final image. For each method, there is a different if-else cause that will provide the necessary steps to populate the lambda matrix.

Methods such as adjacent, classic, and exponential each have their unique logic for generating the lambda values. The logic for each method was based on either their original paper or an adaption done by us. As we mentioned before, these values are a crucial part because they represent the proportion of each class's contribution in the mixed image. For example, the exponential method employs an exponential distribution influenced by τ, offering a much more complex approach to blending. During this work, we explored different values for τ in the exponential method, as we decided to start with the τ equal to 0.15 as it was also explored by [16].

Once the lambda matrix is established, we proceed to mix the images. This is achieved by multiplying all images of a class by the corresponding lambda value and then summing these results to produce the final mixed image. This process effectively blends images from different classes as dictated by the lambda values, which results in a batch of mixed images. Each mixed image is thus a composite of contributions from multiple classes, as previously determined by the lambda matrix.

We conclude by returning the batch of mixed images alongside the lambda matrix. The mixed images are then utilized for training the model, and the lambda matrix serves to illustrate the degree of contribution from each class in every mixed image. This approach not only enhances the training data with diverse examples but also aligns with the specific requirements of different Mixup strategies, making it highly adaptable for various datasets and scenarios.

Initially, we stored all the lambda values for all the experiments, however, once we tested our solution and made sure that it was working as expected, this step was not necessary anymore. To gain a better visualization of the Mixup data processing function explained above, we have also included the Algorithm 1 as follows.

Table 4.1: Example of Lambda values for different classes and methods based on FogChambers dataset

Class	**Nothing**	**Classic**	**Adjacent**	**Exp**
Class 0	0.000	0.000	0.000	0.000
Class 1	1.000	0.000	0.343	0.000
Class 2	0.000	0.000	0.657	0.001
Class 3	0.000	0.457	0.000	0.997
Class 4	0.000	0.000	0.000	0.001
Class 5	0.000	0.543	0.000	0.000

Algorithm 1 Mixup Data Modified Algorithm

procedure MIXUP_DATA($list_images$, $method$)
  Initialize $lambdas_matrix$ with zeros of size $N \times K$
  $K \leftarrow$ LEN($list_images$)
  $N \leftarrow list_images[0].shape[0]$
  if $method =$ "Nothing" **then**
    for i **in** RANGE(N) **do**
      $l \leftarrow$ RANDOMINT$(0, K)$
      $lambdas_matrix[i, l] \leftarrow 1$
    end for
  else if $method =$ "Classic" **then**
    for i **in** RANGE(N) **do**
      $\alpha \leftarrow 1.0$
      $l1 \leftarrow$ RANDOMINT$(0, K)$
      $l2 \leftarrow$ RANDOMINT$(0, K)$
      $\beta \leftarrow$ BETADISTRIBUTION(α, α)
      $lambdas_matrix[i, l1] \leftarrow \beta$
      $lambdas_matrix[i, l2] \leftarrow 1 - \beta$
    end for
  else if $method =$ "Adjacent" **then**
    for i **in** RANGE(N) **do**
      $class_1 \leftarrow$ RANDOMINT$(0, K-2)$
      $class_2 \leftarrow class_1 + 1$
      $\alpha \leftarrow 1.0$
      $\beta \leftarrow$ BETADISTRIBUTION(α, α)
      $lambdas_matrix[i, class_1] \leftarrow \beta$
      $lambdas_matrix[i, class_2] \leftarrow 1 - \beta$
    end for
  else if $method =$ "Exponential" **then**
    for i **in** RANGE(N) **do**
      $l \leftarrow$ RANDOMINT$(0, K) + 1$
      $k \leftarrow$ ARANGE$(1, K + 1)$
      $lambdas_matrix[i] \leftarrow$ EXP$(k, l, tau = args.tau)$
    end for
  end if
  $mixed_image \leftarrow$ Apply lambdas for each class and sum
  return $mixed_image, lambdas_matrix$
end procedure

Table 4.1 shows an example of how the classes would be mixed in each method by using the lambda matrix. It is possible to see that in the method nothing, or no application of Mixup, just one class is selected, and in the other methods it will change based on their logic. Specifically for the exponential, we can see that usually one class is selected and the contribution for the others is relative to the parameter τ that was previously defined. We will explore each method in more detail in the subsequent sections.

4.3 Data Preparation Methods

By preparing our dataset, we emphasized the need for precision and flexibility, enabling the selection of different loss functions and Mixup methods. To effectively implement the Mixup process during training, we divided the dataset into class-specific loaders. In each training iteration, images from these loaders were combined by the chosen Mixup strategy, resulting in a set of blended images and labels. Subsequently, the network was trained on these composite images, utilizing the specified loss function and optimizers such as Adam, SGD, and Adadelta. In the beginning, we decided to test different optimizers, but we failed to continue doing it because it would also expand enormously our experiment combinations, which could confuse us. Therefore, we decide to focus on Adadelta to not commit to the need to define each hyperparameter such as learning rate and decay.

4.4 Datasets

A compilation of various ordinal datasets can be found in [2], which includes both tabular and image datasets. For this work, our attention will primarily be on image datasets focusing on SMEAR2005 [11] and Seeing Through Fog [6], which here we will call FogChambers. Apart from testing our work in the aforementioned datasets, we also tested the Mixup code in the CIFAR10 dataset as mentioned by their study, and re-tested our solution on the same dataset to validate the usability and correctness of our proposed method. Since our main focus is on ordinal data, we will not explore any results related to non-ordinal datasets in this work.

Therefore, we tried to replicate and leverage some of the datasets previously examined in the literature review.

Table 4.2: Datasets used for ordinal experimentation.

Dataset	**Category**	**N**	**Classes**
SMEAR2005 [11]	Biomedical	917	7
Seeing Through Fog [6]	object detection	1034	6

The datasets mentioned in the Table 4.2 vary widely in terms of their application domains, size, and complexity. The first one, SMEAR2005, is a biomedical dataset that contains a relatively small number of images (917) but has a higher number of classes (7), making it suitable for detailed classification tasks in medical research. In contrast, the FogChambers dataset, used in

object detection and potentially in autonomous driving applications, has a larger number of images (1034) but fewer classes (5), indicating its focus on categorizing visibility conditions critical for vehicular navigation systems. Each dataset offers unique challenges and opportunities for research in ordinal classification. The diversity in the datasets also allows for the testing and validation of models across different domains, enhancing the generalizability and robustness of the proposed methods. To work with the different datasets we took advantage of the PyTorch Dataset class that was adapted for our needs. For both datasets, the image size utilized was defined as 256.

4.4.1 SMEAR2005

The SMEAR2005 dataset is a collection of cytology images with seven different classes representing various cell types and can be used to train machine learning models to predict cervical malignancy. This dataset was explored in different studies such as [5] where they presented a novel model for the supervised classification of ordinal data.

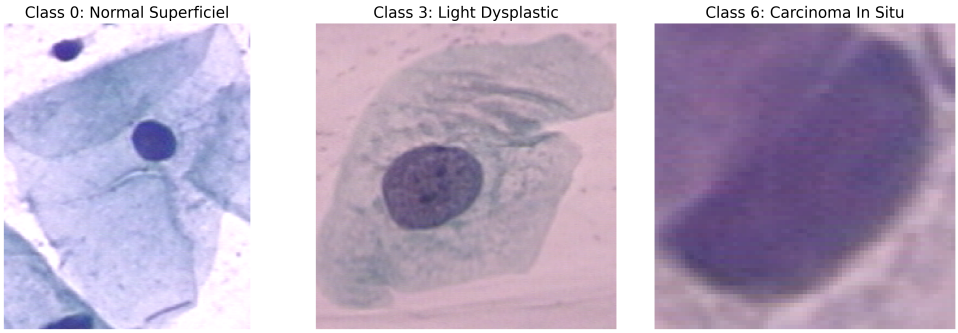


Figure 4.2: Example of SMEAR2005 classes

Figure 4.2 shows an example of 3 different classes, in order, being the first class 0, which is a normal superficial, followed by class 3 which is a light dysplastic, and then class 6 which is a carcinoma in situ. As you may have noticed, the classes are encoded as integers ranging from 0 to 6. Images are loaded from the specified directory, and their labels are assigned based on the subfolder names which correspond to different cell types. Below there is a brief explanation of the possible classes for the dataset.

1. **Normal superficiel**: This category refers to cells that appear normal and are located in the superficial layers of the epithelium.

2. **Normal intermediate**: These are normal cells found in the intermediate layers of the epithelium.

3. **Normal columnar**: This refers to normal columnar cells, typically found in glandular structures or the inner lining of organs.

4. **Light dysplastic**: Cells in this category show mild dysplasia, indicating early changes that could potentially lead to cancer.

5. **Moderate dysplastic**: Here, cells exhibit moderate dysplasia, a more advanced stage of abnormal growth, suggesting a higher risk of developing into cancer.

6. **Severe dysplastic**: This category represents severe dysplasia, where cells are highly abnormal and the risk of cancer is more imminent.

7. **Carcinoma in situ**: Refers to a stage where cancer cells are present but have not yet spread beyond the epithelium. It's a pre-invasive stage of cancer.

As we can see, the order of the classes is ranked by its severity and that is why it is important to use an ordinal data approach to create a machine learning model capable enough to distinguish between the target's order. Therefore, it is important to note that different classifications may also incur in different treatments, and getting the classes wrongly predicted could potentially lead to an enormous impact on a patient's life

The total size of the dataset is 917 images with 7 classes ($K = 7$), and the training set size is 733 images. The class distribution is shown in Figure 4.3. The dataset is split into training and testing sets using a stratified k-fold method to ensure a balanced representation of each class.

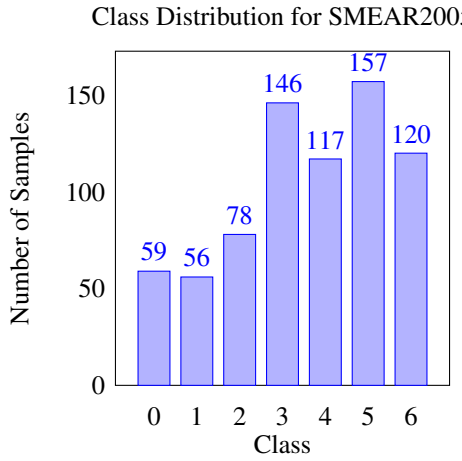


Figure 4.3: Class distribution for SMEAR2005.

4.4.2 Seeing Through Fog (FogChambers)

In this section, we discuss the adaptation of the FogChambers [6] dataset from its original regression problem format to an ordinal classification framework. This transformation is pivotal for applications in meteorological visibility, where categorizing visibility into distinct classes is more beneficial than predicting precise values.

Fig 4.4 shows an example containing 3 samples where the classes 0, 3, and 5 are presented along with their respective bins and meteorological visibility. For this dataset, we focused on the images from cerema-*cam_stereo_left_lut* which is an 8-bit RGB left stereo camera image used for annotating the dataset, with custom tone mapping presented in [6].

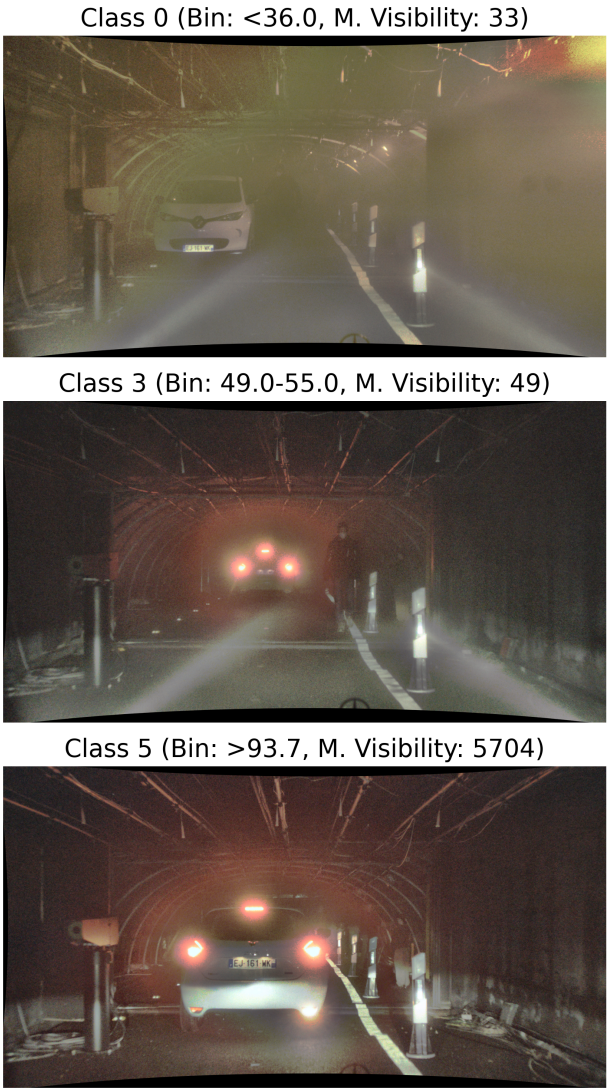


Figure 4.4: Example of FogChambers classes

The dataset consists of images categorized based on meteorological visibility. Each image file is associated with a visibility level, which is then discretized into one of the six bins. The total size of the dataset is 1034 images with 6 classes ($K = 6$), and the training set size is 827 images, which makes the test set 20% of the total images. The class distribution is as shown in Figure 4.5.

Since the FogChambers dataset was not initially designed to be used in ordinal classification, we discretize the continuous visibility measurements into predefined ranges. The bins were defined as [36.0, 43.0, 49.0, 55.0, 93.7] to have an almost balanced label distribution. As we also did for the SMEAR2005 dataset, a random splitting was used to select the test data by using 20% of the available data. The splitting process, based on a specific seed and fold parameter, ensures that the training and testing data vary in different experiment runs, adhering to the standard practices in cross-validation techniques. Each data item in the dataset consists of an image and its corresponding visibility class that were stored in JSON files.

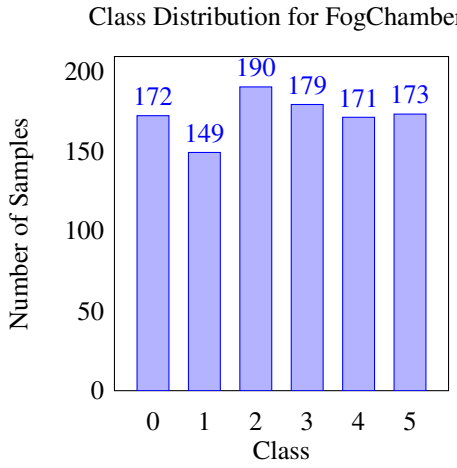


Figure 4.5: Class distribution for FogChambers.

The adaptation of the FogChambers dataset could be easily aligned with practical needs in meteorological visibility prediction in autonomous driving, emphasizing the categorization of visibility into intuitive and actionable classes rather than precise, continuous measures. However, this will depend on how this information could be used. However, it would also be interesting to compare the performance of using it as ordinal versus applying a regression model to predict the value. Transforming the target of this dataset in an ordinal problem also opens up possibilities to do the same for other datasets that could take advantage of having an ordinal data perspective instead of binary or regression commonly used tasks. With that, is important to note that the application that we will be using should be always the guidance for what we will develop as scientists.

4.5 Model Architecture and Hardware Enhancement

Our methodology adapted the famous ResNet18 architecture [13], a choice driven by its proven effectiveness in image classification tasks, and also used in the Mixup paper. The architecture was modified to align with our research objectives, particularly in accommodating the number of classes present in our dataset. Furthermore, all the experiments were run using the powerful NVIDIA A100-SXM4-40GB GPU, which was provided by the University of Porto ATLAS research group. This combination offered a robust and efficient framework for our study. To maximize the capabilities of the GPU, we developed a bash script for parallelizing computations. This script systematically iterated through various combinations of datasets, methods, optimizers, and loss functions, optimizing our research process and enhancing computational efficiency.

4.6 Losses

In our code implementation, we have explored a variety of different loss functions, including traditional and custom-developed ones. Since the Mixup paper explored just the cross-entropy loss for their solution, we decided to also include more options such as Ordinal Encoding loss,

which is more suitable for accurately quantifying the error in predictions considering the ordinal nature of the targets. This approach ensures that the loss calculation aligns with the inherent ranking structure of the data, potentially leading to more effective learning and prediction. In our implementation, we calculate the total loss for predictions using a similar concept as Mixup augmentation, by weighting individual class losses with corresponding lambda matrix coefficients and summing them up to obtain the total loss. The criterion utilized can be either Cross-Entropy (CE) or Ordinal Encoding (OE). Furthermore, it is important to note that for Cross-Entropy, we also use a weight multiplication to be similar to Unimodal Regularization Loss (URL), and take into account the mixing values pre-calculated in the blending process.

4.7 Optimization Method Selection

For the optimization, our primary choice was the Adadelta optimizer [25]. We also explored other options, including Adam [15] and Stochastic Gradient Descent (SGD) [19], to ensure a comprehensive assessment of available methodologies. Our decision to continue with Adadelta was influenced by its user-friendly feature of not requiring predefined initial parameters such as learning rate. This aspect allowed us to concentrate our resources more effectively on the diverse experiments at hand. Notably, in the Mixup paper, SGD was used with a dynamic learning rate and a predetermined weight decay. However, our research focus was directed more toward analyzing losses and methods rather than optimizing model performance. Hence, we reserved the exploration of these optimization nuances for future studies, ensuring a more targeted and efficient approach in our current research.

4.8 Training methodology

Our training methodology was conducted over 200 epochs of training, which was executed 3 times for each experiment. This systematic approach ensured consistent and comparative analyses across the different Mixup methodologies. We evaluated the model's performance meticulously using a range of metrics, including accuracy and mean absolute error (MAE), on both training and testing datasets. Furthermore, we generated and analyzed confusion matrices to obtain deeper insights into the model's class-specific performance.

The batch size, which refers to the number of data samples that are processed together in a single forward and backward pass during training, was defined as 16. This choice was based on the size of our datasets to maintain representability and make the training possible. We also based it on quick tests to see how the results would be affected for different batch sizes. However, since the main purpose of this work was not to define the best batch size for each approach, we decided to stick to a batch size being 16 and focus our efforts on other experiments.

4.9 Data augmentation

Apart from using Mixup-related methods to augment our data, we also utilized a common data augmentation by doing image transformations as explored by [7] and commonly found in computer vision tasks. This strategy was applied to the training dataset and is composed of a series of transformations that have the goal of enhancing the robustness and generalizability of the machine-learning model by introducing variability into the training data. The following list enumerates and explains each transformation step in the train transform sequence:

- **Image Resizing:** The application of resize increases the image dimensions by 5% on each side, with the antialias parameter that is used to enhance image quality after the resize.
- **Random Cropping:** By randomly cropping the image to its initial size, we introduce variability and aid the model in learning to recognize different image regions.
- **Color Adjustment:** By randomly changing image brightness and contrast, we add parameters to control the extent of these adjustments. This transformation is crucial for improving the model's resilience to variations in lighting conditions, which in the case of FogChambers dataset, or even autonomous vehicles tasks is essential because the light condition is usually out of control.
- **Random Horizontal Flipping:** Here we introduced a horizontal flip to the image with a default probability of 50%. This step aims to develop model invariance to object orientation within images.
- **Normalization:** The last step is the normalization which standardizes the image by adjusting each color channel based on the mean and standard deviation. The values utilized were the same as [7].

A flag was used to ensure that the training data skipped or used the augmentation process above. Therefore it will be applying the same transformations as those utilized for the test data which is the conversion to float32, followed by an image resize and normalization.

After applying these transformations, the training and test datasets are instantiated accordingly. Data loaders are then initialized to facilitate the efficient feeding of data into the model during both the training and testing phases. It is noteworthy that these augmentations are not applied to the test dataset to ensure its representation of real-world, unaltered data.

4.10 Data Loaders in Model Training and Testing

Another modification that we did in the original code was to create a function to handle the dataloaders. The objective of doing it was to generate individual data loaders for each class within the dataset, which is an important step for the effective application of our method during model training.

The process was structured to first prepare class indices, which are created as a list of indices for every class in the dataset. This was done by iterating over the dataset and assigning indices to their corresponding classes. With these indices, we then create a DataLoader for each class by forming subsets of the dataset and configuring these DataLoaders with specific batch sizes and shuffling. Parallel processing is also employed to enhance efficiency, and for our case was essential because of the amount of experiment combinations that we utilized.

In the training phase, the DataLoader makes possible selective sampling and mixing of images from different classes, which was developed to suit all the Mixup methods in use, such as adjacent class mixing, exponential, and classic. For testing, the Mixup was not applied, however, the structured arrangement of DataLoaders still contributes to efficient model processing and evaluation.

Overall, making these dataloaders was a key step for splitting the dataset into batches based on classes. This helps not just in using different Mixup ways during training, but also in checking the model well during testing.

4.11 Evaluation Metrics

To ensure the statistical validity and comprehensive understanding of our experiments, we employed a variety of performance metrics. These metrics are particularly chosen to effectively interpret the results from different methods and loss functions.

- **Accuracy (Acc):** Standard accuracy is the proportion of correctly classified samples. It's calculated as:

$$\text{Accuracy}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(\hat{y}_i = y_i) \tag{4.2}$$

  (where $\mathbb{1}(\cdot)$ is the indicator function)

- **Accuracy (1-OFF):** We included the 'Accuracy 1-OFF' metric as used in [4], which measures the proportion of samples accurately classified within the target class or any adjacent class. This metric is particularly useful for ordinal data, as it provides a more nuanced view of predictions that are close to the correct class. The formula is given by the following equation.

$$\text{1-OFF}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(\hat{y}_i \geq y_i - 1 \wedge \hat{y}_i \leq y_i + 1) \tag{4.3}$$

- **Mean Absolute Error (MAE):** MAE is a standard metric for evaluating the accuracy of predictions. It is calculated as the average of the absolute differences between the predicted and actual values. The formula for MAE is:

$$\text{MAE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i| \tag{4.4}$$

  where y_i and $\hat{y}_i$ are the true and predicted values, respectively, and N is the number of observations.

- **Maximum Mean Absolute Error (MMAE):** As proposed and explored in [9, 12], MMAE assesses the average of the largest errors for each sample, focusing on the worst-case scenarios. Therefore, MMAE is the MAE value of the class with a higher distance from the true values to the predicted ones. The formula for MMAE is:

$$\text{MMAE}(\mathbf{y}, \hat{\mathbf{y}}) = \max_{k \in [1,K]} \left(\frac{1}{N_k} \sum_{i:y_i=k} |\hat{y}_i = k| \right), \tag{4.5}$$

  where N_k is the number of observations belonging to class K.

Furthermore, to improve the statistical reliability of our results, each experiment was replicated three times. This methodology allowed us to calculate both the mean and the population standard deviation (std) of the results, thereby providing a more comprehensive and robust statistical analysis. It is worth noting that we decided to use the population standard deviation instead of the sample standard deviation because our population was composed of a small set of running times. Moreover, we conducted all experiments over 200 epochs. This extended duration was crucial to observe the progressive improvements across epochs and to achieve a better outcome. However, it is important to bear in mind that our focus was on testing the different combinations, and not on obtaining the highest possible performance, which would require more extensive fine-tuning and potentially different model architectures.

4.12 Logs

Throughout our research, a rigorous log of results and metadata was maintained, such as model parameters, training duration, and performance metrics (including a confusion matrix for each experiment). This detailed logging was essential in facilitating a thorough analysis and comprehension of our findings. Finally, we preserved the final model, inclusive of its state, optimizer configuration, and aggregated training metrics, for future reference and potential in-depth analysis.

Chapter 5

Results and Discussion

In this section, we present and discuss the results obtained from an extensive series of experiments. These experiments included two datasets, two loss functions, four different methods, and four variations of the tau parameter, and also with augmented and non-augmented data. By augmenting, we refer to the application of the data augmentation explained in sub-section 4.9. The results were grouped and presented in two different tables for clarity. As previously mentioned, our main goal besides developing a novel approach based on the Mixup method, was to focus on different experimentations and adaptations, so then we would be able to get more basis for our findings.

Table 5.1 is focusing in all the methods, with a fixed tau value of 0.15 for the exponential method. On the other hand, Table 5.2 focuses on experiments related to the variance of the tau parameter within the exponential method. Both tables provide comprehensive details on each combination and include both datasets, which facilitated easy comparisons across various combinations. Therefore, the tables can be analyzed by line, by columns, or by a block of experiments as suggested in the table lines. As previously mentioned, each experiment was conducted three times, and the reported values are the averages, with the population standard deviation provided after the $\pm$ signal. Another important point to bear in mind is that we also included a data augmentation transformation applied before the methods, which can also be compared without it. As expected, all the experiments that were run using the extra data augmentation explored in sub-section 4.9, had longer processing times. Since our idea was to focus on the different experiments on test data, we will not show the training-related metrics in this work.

To evaluate the models, we considered various metrics beyond simple accuracy, recognizing that accuracy alone may not be ideal for ordinal datasets. Metrics such as accuracy 1-OFF, Mean Absolute Error (MAE), and Mean Maximum Absolute Error (MMAE) were also employed. The best-performing results are highlighted in bold for each block of experiments. All the results presented in Table 5.2 and Table 5.1 were based on test data that corresponds to 184 examples for SMEAR2005 and 207 examples for FogChambers, which represents 20% of each dataset.

We employed various augmentation methods to explore their impact on model performance with ordinal data. The methods include Classic Mixup, Adjacent Mixup, Mixup Exponential, and a control group with no Mixup (Nothing).

Table 5.1: Analysis of experiments for different methods

		SMEAR2005				FogChambers			
		Without Augmentation		With Augmentation		Without Augmentation		With Augmentation	
	Method	**CE**	**OE**	**CE**	**OE**	**CE**	**OE**	**CE**	**OE**
Acc	Nothing	48.01 ± 8.49	45.29 ± 4.99	44.93 ± 0.51	37.14 ± 4.66	76.84 ± 7.04	77.78 ± 4.46	**68.60** ± 4.44	67.89 ± 1.48
	Classic	41.85 ± 1.78	41.48 ± 2.45	46.20 ± 5.23	34.06 ± 8.44	**80.97** ± 5.29	75.88 ± 6.42	54.72 ± 11.20	58.26 ± 5.68
	Adjacent	41.67 ± 4.66	37.86 ± 1.28	49.46 ± 6.83	33.15 ± 5.38	62.03 ± 12.08	65.80 ± 10.15	59.55 ± 1.46	47.34 ± 7.51
	Exponential*	45.29 ± 3.12	**49.46** ± 1.60	46.56 ± 8.45	**50.54** ± 5.12	78.45 ± 8.68	75.75 ± 5.41	67.92 ± 0.83	59.20 ± 10.30
1-OFF	Nothing	74.64 ± 4.66	81.16 ± 4.99	70.47 ± 4.22	72.83 ± 6.54	96.58 ± 2.28	97.97 ± 1.03	94.75 ± 2.24	**97.00** ± 0.61
	Classic	68.84 ± 4.99	75.00 ± 3.47	71.38 ± 5.86	69.56 ± 8.43	96.97 ± 1.07	**98.87** ± 0.58	84.35 ± 13.63	92.56 ± 3.50
	Adjacent	74.82 ± 1.79	81.70 ± 4.10	75.00 ± 4.63	75.00 ± 7.39	94.01 ± 3.74	98.52 ± 1.13	95.01 ± 1.31	87.63 ± 8.92
	Exponential*	75.73 ± 6.22	**84.42** ± 1.85	70.47 ± 11.58	**81.89** ± 1.56	96.97 ± 2.58	98.13 ± 1.31	95.27 ± 1.46	94.78 ± 1.44
MAE	Nothing	0.97 ± 0.15	0.83 ± 0.13	1.11 ± 0.09	1.03 ± 0.16	0.27 ± 0.09	0.24 ± 0.05	0.37 ± 0.07	**0.35** ± 0.02
	Classic	1.25 ± 0.14	0.94 ± 0.07	1.08 ± 0.13	1.08 ± 0.18	**0.23** ± 0.07	0.25 ± 0.07	0.72 ± 0.39	0.51 ± 0.10
	Adjacent	1.03 ± 0.08	0.90 ± 0.09	0.97 ± 0.17	1.01 ± 0.18	0.45 ± 0.17	0.36 ± 0.11	0.46 ± 0.00	0.66 ± 0.17
	Exponential*	0.99 ± 0.15	**0.76** ± 0.02	1.07 ± 0.29	**0.80** ± 0.06	0.25 ± 0.12	0.26 ± 0.07	0.37 ± 0.02	0.46 ± 0.12
MMAE	Nothing	1.72 ± 0.27	1.82 ± 0.33	1.98 ± 0.02	1.91 ± 0.27	0.62 ± 0.29	**0.46** ± 0.12	0.76 ± 0.17	**0.60** ± 0.10
	Classic	3.11 ± 0.94	1.96 ± 0.14	1.92 ± 0.20	1.89 ± 0.48	0.62 ± 0.21	0.48 ± 0.13	1.30 ± 0.49	0.95 ± 0.32
	Adjacent	1.72 ± 0.43	1.66 ± 0.31	1.94 ± 0.48	1.73 ± 0.14	0.85 ± 0.31	0.66 ± 0.20	1.00 ± 0.27	1.37 ± 0.38
	Exponential*	2.39 ± 0.62	**1.38** ± 0.22	1.84 ± 0.25	**1.46** ± 0.10	0.54 ± 0.37	0.54 ± 0.12	0.68 ± 0.06	0.89 ± 0.08

∗ proposed method.

Table 5.2: Analysis of different taus for exponential method

		SMEAR2005				FogChambers			
		Without Augmentation		With Augmentation		Without Augmentation		With Augmentation	
	Tau	**CE**	**OE**	**CE**	**OE**	**CE**	**OE**	**CE**	**OE**
Acc	0.15	45.29 ± 3.12	49.46 ± 1.60	46.56 ± 8.45	**50.54** ± 5.12	**78.45** ± 8.68	75.75 ± 5.41	**67.92** ± 0.83	59.20 ± 10.30
	0.45	**54.71** ± 2.96	43.48 ± 11.53	47.46 ± 6.90	41.12 ± 7.43	76.87 ± 4.33	67.15 ± 4.65	57.26 ± 5.46	55.30 ± 6.95
	0.60	43.30 ± 2.85	44.02 ± 5.23	44.75 ± 6.82	36.41 ± 10.38	57.94 ± 11.20	51.08 ± 9.39	38.13 ± 2.09	48.34 ± 10.03
	1.00	29.89 ± 1.60	35.51 ± 4.38	30.61 ± 5.37	37.14 ± 2.44	39.13 ± 3.93	51.50 ± 6.23	32.05 ± 1.98	41.93 ± 1.31
1-OFF	0.15	75.73 ± 6.22	**84.42** ± 1.85	70.47 ± 11.58	**81.89** ± 1.56	96.97 ± 2.58	**98.13** ± 1.31	**95.27** ± 1.46	94.78 ± 1.44
	0.45	81.34 ± 3.59	76.45 ± 9.10	72.83 ± 7.10	76.99 ± 7.64	95.52 ± 1.76	95.94 ± 2.45	84.83 ± 6.37	92.69 ± 3.91
	0.60	75.73 ± 3.56	78.80 ± 3.35	76.99 ± 6.16	72.65 ± 9.79	86.86 ± 9.76	87.31 ± 5.46	60.64 ± 4.69	79.39 ± 16.37
	1.00	62.32 ± 2.81	71.92 ± 4.74	63.04 ± 6.29	75.91 ± 2.56	68.95 ± 6.17	88.79 ± 6.31	51.60 ± 3.69	74.07 ± 4.37
MAE	0.15	0.99 ± 0.15	**0.76** ± 0.02	1.07 ± 0.29	**0.80** ± 0.06	**0.25** ± 0.12	0.26 ± 0.07	**0.37** ± 0.02	0.46 ± 0.12
	0.45	0.81 ± 0.12	0.93 ± 0.30	1.00 ± 0.18	0.93 ± 0.20	0.28 ± 0.07	0.37 ± 0.07	0.62 ± 0.15	0.52 ± 0.11
	0.60	1.00 ± 0.09	0.92 ± 0.10	0.98 ± 0.19	1.02 ± 0.26	0.61 ± 0.27	0.63 ± 0.16	1.32 ± 0.13	0.83 ± 0.39
	1.00	1.41 ± 0.10	1.09 ± 0.10	1.41 ± 0.10	1.09 ± 0.10	1.08 ± 0.16	0.61 ± 0.13	1.59 ± 0.15	0.92 ± 0.08
MMAE	0.15	2.39 ± 0.62	**1.38** ± 0.22	1.84 ± 0.25	**1.46** ± 0.10	0.54 ± 0.37	**0.54** ± 0.12	**0.68** ± 0.06	0.89 ± 0.08
	0.45	1.61 ± 0.30	1.81 ± 0.48	1.65 ± 0.24	1.69 ± 0.20	0.68 ± 0.17	0.93 ± 0.32	1.37 ± 0.28	1.05 ± 0.34
	0.60	1.64 ± 0.16	1.87 ± 0.52	2.07 ± 0.76	1.73 ± 0.21	1.27 ± 0.38	1.33 ± 0.41	2.61 ± 0.43	1.61 ± 0.88
	1.00	3.20 ± 0.47	2.71 ± 0.26	3.52 ± 0.28	3.05 ± 0.14	2.21 ± 0.32	1.32 ± 0.16	3.22 ± 0.33	1.74 ± 0.28

5.1 Analysis of Experiment Results

The results are methodically presented in Tables 5.1 and 5.2, where 5.1 focus on the comparative analysis of different methods and Table 5.2 explores the different values of tau parameter to investigate the variance.

Comparing the results from the SMEAR2005 and FogChambers datasets offers insights into the adaptability and effectiveness of various methods and loss functions in different contexts. In total, we carried out 64 experiments to investigate a wide range of methods, loss function, and tau value combinations. However, this total grows considerably when factoring in the multiple runs needed to compute averages and standard deviations. It also does not include all the unsuccessful attempts made in between these runs.

5.1.1 Analysis of SMEAR2005

For the SMEAR2005 dataset, the exponential method proved effective, for both cases with and without augmentation and using ordinal encoding as loss. The consistency in high 1-OFF accuracy across methods is clear, underscoring the models' generalization capability in ordinal classification. The exponential method also showed lower MAE and MMAE, highlighting its performance in predictions. From Table 5.1 it is evident that the exponential method was the best performer in all the experiments. Another point that is worth noting is that by using OE instead of CE we have achieved better performance metrics, with often smaller standard deviations as well.

In the experiments with augmentation, the performance of the exponential method varied much more, which could suggest that its complexity could be playing a factor in it. Despite this, high 1-OFF accuracy was maintained across the different methods, and a slight increase in MAE and MMAE was noted, indicating a trade-off between augmentation and precision.

The SMEAR2005 experiments, as also noted in the FogChambers dataset, revealed a decrease in performance metrics with an increase in Tau value. However, the overall performance metrics for SMEAR2005 were lower than those observed in the FogChambers experiments, regardless of the Tau value. This suggests that the SMEAR2005 dataset might be more challenging or that the methods and parameters are not as well-suited for this dataset.

In the SMEAR2005 experiments, the exponential method outperformed others in terms of accuracy, particularly with OE, and showed the lowest MAE and MMAE. In contrast to the FogChambers results, the nothing method did not perform as well, indicating the possible dataset-specific efficacy of these methods. The adjacent and classic methods continued to show moderate performance, similar to their FogChambers counterparts.

5.1.2 Analysis of FogChambers

In the FogChambers dataset, the tau value of 0.15 in the exponential method consistently showed excellent performance, especially in terms of Acc and MAE, both with and without augmentation. The highest accuracy was observed at classic mixup with 80.97% without augmentation and 68.60% with augmentation for no mixup (nothing). Analyzing the settings without augmentation, the Classic method with CE resulted in the highest accuracy, demonstrating the strength of the Mixup algorithm in linearly interpolating between two classes. Moreover, 1-OFF accuracy was notably high across all methods, with the Classic and Nothing methods excelling, indicating robustness in ordinal classification even when not using any Mixup variant. The MAE and MMAE metrics also confirmed the precision of the Classic and Nothing methods.

On the other hand, with augmentation, the exponential method experienced a decline in accuracy, possibly due to the added complexity of its combination strategy. However, the ordinal classification was evident as 1-OFF accuracy remained high. The augmented settings also resulted in slightly higher MAE and MMAE, indicating an increase in prediction errors.

The settings that used the OE loss function obtained a better performance when applied along with data augmentation than the OE without data augmentation.

Apart from the average performance, it is also worthwhile to examine the standard deviations. Where we can observe that they are often higher for CE compared to OE. This suggests that there is greater variance in the results obtained with CE than with OE.

Overall, the FogChambers dataset had more variation in the results than the SMEAR2005.

5.2 General Insights

Across both datasets, the smaller value for tau often yielded the best results in settings with ordinal encoding loss, highlighting its effectiveness for ordinal data. Our proposed method, exponential, consistently outperforms the simpler methods for the SMEAR2005 dataset, suggesting that its unimodal strategy might be advantageous for this dataset. Augmentation strategies, particularly the adjacent and classic methods, showed varied impacts, indicating a dataset-dependent effectiveness. Interestingly, the control group (nothing) frequently performed on par with the augmented methods, which can suggest the inherent strength of the ResNet18 model even in the absence of augmentation in ordinal data scenarios.

Figure 5.1 displays four examples of the obtained confusion matrices in a sample of our experiments. From this figure, focusing on the colors and not on the small numbers, we observe that all the methods exhibit a strong diagonal color, indicating good overall performance. However, it is also noticeable that the models frequently make errors in predicting adjacent classes. Therefore, when utilizing accuracy with a 1-OFF tolerance, all incorrectly predicted adjacent classes are considered correct. This approach provides a clearer perspective on whether the models tend to make errors in adjacent classes or not.

The usage of confusion matrices proves to be valuable for visually identifying the most significant errors and their proximity. In all the experiments conducted, we saved confusion matrices similar to the ones shown in Figure 5.1. These matrices are instrumental for conducting detailed investigations and gaining insights into the specific areas where each combination tends to produce incorrect predictions. However, please note that Figure 5.1 serves as an illustrative example to underscore the advantages of employing accuracy with a 1-OFF tolerance, especially in the context of ordinal data.

Overall, our analysis indicates that the effectiveness of methods and tau values is specific to each dataset. Lower tau values and the Exponential method generally lead to better accuracy and reduced error rates. This tendency is especially marked when considering metrics such as 1-OFF accuracy and MMAE for ordinal datasets, highlighting the necessity of multiple performance metrics and tailoring methodological approaches to the distinct features of each dataset.

Moreover, our findings emphasize the importance of employing multiple performance metrics in the comparative analysis of different experimental setups. In our experiments, it was evident that the 1-OFF accuracy consistently outperformed standard accuracy. This observation is particularly relevant for our study because, although the datasets have nearly equal class proportions, in an ordinal context, errors in adjacent classes are preferable to those in distant classes. It is also

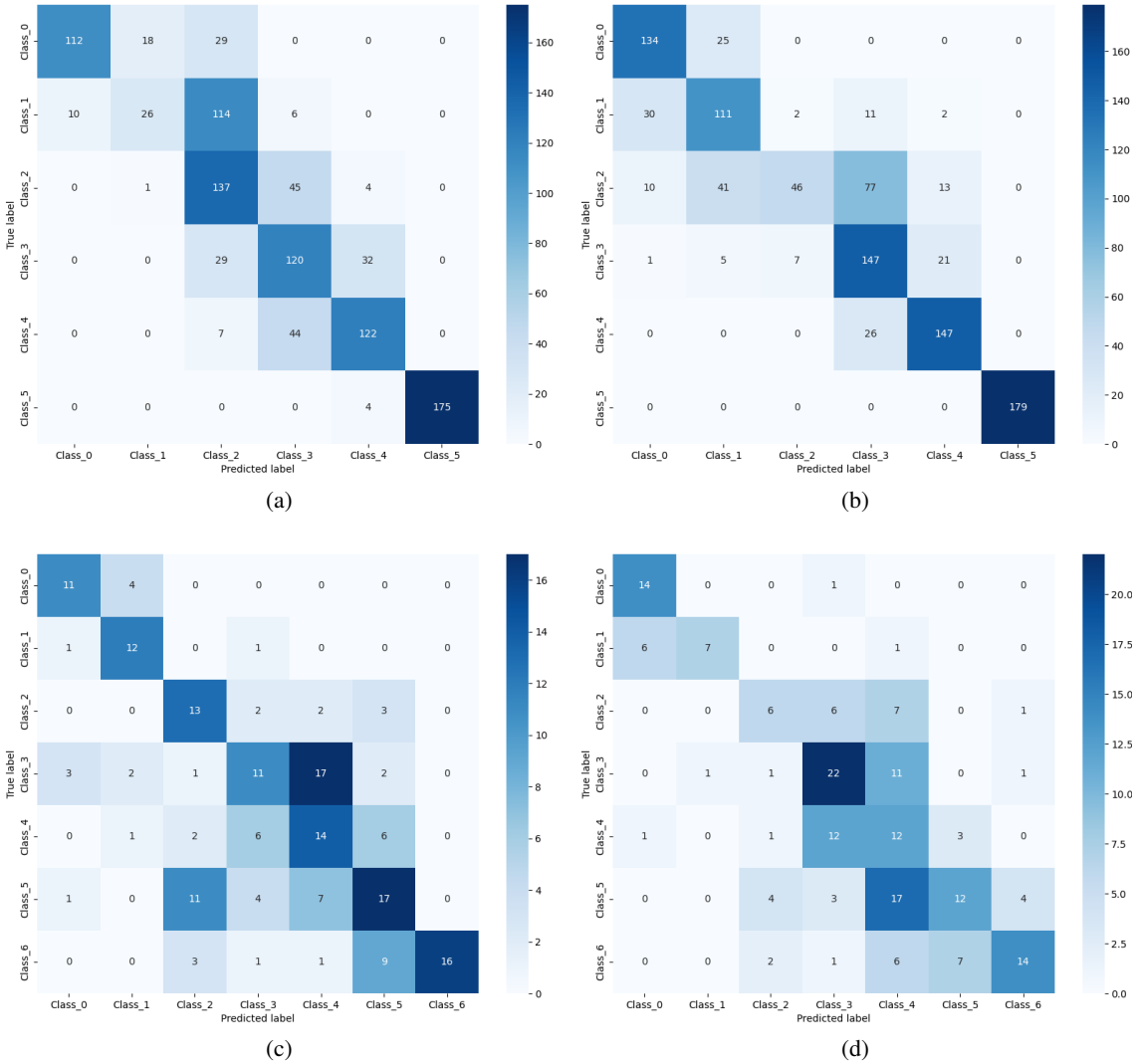


Figure 5.1: (a) FogChambers using CrossEntropy and method nothing with data augmentation (b) FogChambers using CrossEntropy and classic method with no augmentation (c) SMEAR2005 using OrdinalEncoding and method exponential (d) SMEAR2005 using OrdinalEncoding and method exponential with data augmentation

noteworthy that the adjacent method frequently showed favorable results even in comparison with the classic Mixup.

Both datasets exhibited a decline in performance metrics with an increase in Tau value. However, the overall performance on the SMEAR2005 dataset was generally lower across all methods and metrics compared to the FogChambers dataset. This indicates that the SMEAR2005 dataset might present a more complex challenge, requiring further optimization of methods and parameters. Moreover, our method performed well specifically for SMEAR2005, indicating that it might also be interesting to test similar datasets to benchmark the performance. For the FogChambers, our method also performed well, but could not reach the biggest performance, even though it got nearly the same when considering the standard deviation.

Chapter 6

Conclusion

This work explored various possibilities of using Mixup as a data augmentation tool for ordinal data and we also proposed a novel method that we called here exponential. Throughout our research, the employment of diverse methods and losses facilitated diverse comparisons. However, given our decision to focus on only two datasets, a potential area for enhancement in future studies would be to expand our research to include more ordinal datasets from different domains and types, such as tabular data as explored by [26]. This can open the paths to assess model performance across a bigger range of classes and applications in ordinal settings.

Another aspect that could significantly influence future research is the endeavor to enhance the performance achieved in this work by fine-tuning the model's hyperparameters and experimenting with more versatile optimizers such as Adam or SGD. Additionally, adapting Cutmix to a multiclass ordinal problem and comparing its efficacy with our approach represents another promising avenue. Such an exploration would undoubtedly offer valuable insights by contrasting our method with a different data augmentation technique. Furthermore, since it was possible to see that by setting smaller values for τ, it might be worth investigating if even smaller values would lead to a better result, but also having in mind the trade-off of having strongly smaller values for the lambdas that will set the contribution of each class in the training.

Moreover, the comparative analysis reveals that while certain methods and loss functions can be effective across different datasets, their performance can vary significantly based on the dataset's characteristics and challenges. Therefore, a careful selection and optimization of methods and parameters are crucial for different datasets in the field of ordinal data augmentation. However, as we can see in the results, even though our method did not achieve the best overall performance in the FogChambers dataset, indeed it got closer, and almost equal when considering the standard deviation range.

In conclusion, our study provided a novel approach to using unimodal exponential distribution as an alternative to Mixup for ordinal data. The extensive experiments provided valuable insights into the effectiveness of various Mixup-based methods in the field of data augmentation for ordinal classification. Through our methodical and detailed experimental setup, we were able to assess the impact of these data augmentation techniques on model performance, thereby contributing

valuable findings to the field. A key result that can be drawn from this work is that by applying our exponential method along with ordinal encoding for SMEAR2005 with an augmented dataset we could decrease the MAE by 25.92% when compared to the original Mixup (classic) algorithm applied to the same dataset, and without augmentation, we obtained 14.89% of decrease in MAE. For the FogChambers dataset, the results were slightly better for the classic Mixup, which could be re-tested in the future by following some of the approaches mentioned above to enhance the performance.

References

[1] Mixup CIFAR 10, code repository. https://github.com/facebookresearch/Mixup-cifar10.

[2] T. Albuquerque. Ordinal datasets. https://github.com/tomealbuquerque/ordinal-datasets. Accessed: September-2023.

[3] Cardoso JS. Albuquerque T, Cruz R. Ordinal losses for classification of cervical cancer risk. *PeerJ Computer Science*, 7:e457, 2021.

[4] Rafael Ayllón-Gavilán, David Guijo-Rubio, Pedro Antonio Gutiérrez, Anthony Bagnall, and Cesar Martínez. Convolutional and deep learning based techniques for time series ordinal classification. https://arxiv.org/abs/2306.10084, 06 2023.

[5] Dalal Bardou, Kun Zhang, and Sayed Mohammad Ahmad. Classification of breast cancer based on histology images using convolutional neural networks. *IEEE Access*, 6:24680–24693, 2018.

[6] Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer, and Felix Heide. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather. pages 11679–11689, 06 2020.

[7] Jaime S. Cardoso, Ricardo Cruz, and Tomé Albuquerque. Unimodal distributions for ordinal regression. https://arxiv.org/abs/2303.04547, 2023.

[8] Jianlin Cheng, Zheng Wang, and Gianluca Pollastri. A neural network approach to ordinal regression. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1279–1284, 2008.

[9] M. Cruz-Ramírez, Cesar Martínez, Javier Sánchez-Monedero, and Pedro Antonio Gutiérrez. Metrics to guide a multi-objective evolutionary algorithm for ordinal classification. *Neurocomputing*, 135:21–31, 07 2014.

[10] Terrance DeVries and Graham Taylor. Improved regularization of convolutional neural networks with cutout. 08 2017.

[11] Georgios Dounias, B Bjerregaard, Jan Jantzen, Athanasios Tsakonas, Nicholas Ampazis, G Panagi, and Evangelia Panourgias. Automated identification of cancerous smears using various competitive intelligent techniques. *Oncology reports*, 15 Spec no.:1001–6, 02 2006.

[12] Antonio Durán-Rosal, Julio Camacho-Cañamón, Pedro Antonio Gutiérrez, M Victoria Guiote, Ester Rodríguez-Cáceres, Juan Vallejo-Casas, and Cesar Martínez. Ordinal classification of the affectation level of 3D-images in Parkinson diseases. *Scientific Reports*, 11, 03 2021.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[14] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning (ICML)*, 2020.

[15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[16] Xiaofeng Liu et al. Unimodal regularized neuron stick-breaking for ordinal classification. *Neurocomputing*, 388:34–44, 2020.

[17] A. Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, page 8024–8035. Curran Associates, Inc., 2019.

[18] Joaquim F. Pinto da Costa, Hugo Alonso, and Jaime S. Cardoso. The unimodal model for the classification of ordinal data. *Neural Netw.*, 21(1):78–91, jan 2008.

[19] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[20] Anirudh Som, Sujeong Kim, Bladimir Lopez-Prado, Svati Dhamija, Nonye Alozie, and Amir Tamrakar. *A Machine Learning Approach to Assess Student Group Collaboration Using Individual Level Behavioral Cues*, pages 79–94. 01 2020.

[21] Zhenyu Tang, Zhenyu Zhang, Huabing Liu, Dong Nie, and Jing Yan. *Pre-operative Survival Prediction of Diffuse Glioma Patients with Joint Tumor Subtyping*, pages 786–795. 10 2023.

[22] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6438–6447, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[23] Devesh Walawalkar, Zhiqiang Shen, Zechun Liu, and Marios Savvides. Attentive Cutmix: An enhanced data augmentation approach for deep learning based image classification. pages 3642–3646, 05 2020.

[24] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. Cutmix: Regularization strategy to train strong classifiers with localizable features. pages 6022–6031, 10 2019.

[25] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *ArXiv*, abs/1212.5701, 2012.

[26] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.