Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e de Tecnologia– CCET
Departamento de Computação– DC
Programa de Pós-Graduação em Ciência da Computação– PPGCC

Claudio Filipi Gonçalves dos Santos

Avoiding Overfitting: new algorithms to improve Generalization in Convolutional Neural Networks

São Carlos
2022

Claudio Filipi Gonçalves dos Santos

Avoiding Overfitting: new algorithms to improve Generalization in Convolutional Neural Networks

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Visão Computacional

Supervisor: João Paulo Papa

São Carlos
2022

This work is dedicated to little Mariana, who unfortunately is not here among us anymore.

Acknowledgements

Words cannot express my gratitude to my mother and my father for supporting me the entire way until here, never questioning the paths I took in my career. I am also thankful to my cousins Carla, Murilo, Rodolfo, Jaqueline, Beatriz, Yasmin, Raul, Ruda, and Laura.

I am grateful to Dr. João Paulo Papa who invited me to perform this research and I am also grateful to Marcos, Thierry, Luis Claudio, Leandro, Clayton, Rafael, Diego, and Daniel for supporting all the publications we developed.

I would like to mention my lifetime friends who sustained me in this challenge: Maria, Caio, Vitor, Danilo, Fabiana, Tulio, Priscila, Tatiane, Lucas, and Marcelo, from SESI, and Carol, Sacha, Rafael, Frederick, Camilo, Phelipe, Rodrigo, Pedro, Filipe, Fernando, Wagner, Diego, Marcio, Marcelo, Eduardo, Tulio, João Carlos, João Felipe, Felipe, Daniel, Joab , Arthur, and Augusto, from 51.

This work has been supported by the following Brazilian research agencies: FAPESP (grants 2013/07375-0, 2014/12236-1, and 2019/07665-4), CNPq (grant 308529/2021-9), and Petrobras (grant 2017/00285-6).

*"There's no such thing as neutral education.
Education either functions as an instrument to
bring about conformity or freedom."
(Paulo Freire)*

Resumo

Aprendizagem Profundo alcançou resultados estado-da-arte em vários domínios, como processamento de imagem, processamento de linguagem natural e processamento de áudio. Para alcançar tais resultados, usa-se redes neurais com várias camadas de processamento juntamente com uma enorme quantidade de informações rotuladas. Uma família particular de Aprendizagem Profundo são as Redes Neurais Convolucionais (do inglês, *Convolutional Neural Networks, CNNs*), que funcionam utilizando camadas convolucionais derivadas da área de processamento digital de sinais, sendo muito úteis para detectar características relevantes em dados não estruturados, como áudio e imagens. Uma forma de melhorar os resultados nas CNNs é o uso de algoritmos de regularização, que visam dificultar o processo de treinamento, mas geram modelos que generalizam melhor para inferência quando usados em aplicações. O presente trabalho contribui na área de métodos de regularização para CNNs, propondo mais métodos para uso em diferentes tarefas de processamento de imagens. Esta tese apresenta uma coletânea de trabalhos desenvolvidos pelo autor durante o período de pesquisa, que foram publicados ou submetidos até a atualidade, apresentando: (i) um levantamento, listando trabalhos recentes de regularização e destacando as soluções e problemas da área; (ii) um método de queda de neurônios para uso nos tensores gerados durante o treinamento das CNNs; (iii) uma variação do método mencionado, alterando as regras de descarte, visando diferentes características do tensor; e (iv) um algoritmo de regularização de rótulos utilizado em diferentes problemas de processamento de imagens.

Palavras-chave: Redes Neurais Convolucionais. Regularização.

Abstract

Deep Learning has achieved state-of-the-art results in several domains, such as image processing, natural language processing, and audio processing. To accomplish such results, it uses neural networks with several processing layers along with a massive amount of labeled information. One particular family of Deep Learning is the Convolutional Neural Networks (CNNs), which works using convolutional layers derived from the digital signal processing area, being very helpfull to detect relevant features in unstructured data, such as audio and pictures. One way to improve results on CNN is to use regularization algorithms, which aim to make the training process harder but generate models that generalize better for inference when use in applications. The present work contributes in the area of regularization methods for CNNs, proposing more methods for using in different image processing tasks. This thesis presents a collection of works developed by the author during the research period, which were published or submited until present time, presenting: (i) a survey, listing recent regularization works and highlighting the solutions and problems of the area; (ii) a neuron droping method to use in the tensors generated during CNNs training; (iii) a variation of the mentioned method, changing the droping rules, targeting different features of the tensor; and (iv) a label regularization algorithm used in different image processing problems.

Keywords: Convolutional Neural Networks. Regularization.

List of Figures

Figure 1 – Percentage of the regularization works surveyed in the manuscript. . 30

Figure 2 – The LeNet-5 structure. Inspired in the picture from (??). 31

Figure 3 – How Cutout works. Extracted from (??) 33

Figure 4 – How RandomErasing works. Extracted from (??). 34

Figure 5 – Two different examples using Mixup. Extracted from (??) 36

Figure 6 – How CutMix works. Extracted from (??). 36

Figure 7 – Simulation using grayscale (a)-(c) and colored images (d)-(f): (a) original grayscale image and its outcomes after (b) Dropout and (c) MaxDropout transformations, respectively, and (d) original colored image and its outcomes after (e) Dropout and (f) MaxDropout transformations, respectively. In all cases, the drop rout ate is 50%. 61

Figure 8 – Random training samples from: (a) CIFAR-10 and (b) CIFAR-100 datasets. 64

Figure 9 – Convergence over CIFAR-10 test set. 65

Figure 10 – Convergence over CIFAR-100 test set. 65

Figure 11 – Simulation using colored images. The original colored image is presented in (a), and its outcomes are bestowed after (b) Dropout, (c) MaxDropout, and (d) MaxDropoutV2 transformations using a dropout rate of 50%. 75

Figure 12 – Convergence analysis regarding the CIFAR-10 and CIFAR-100 datasets. 77

Figure 13 – Simulation of Label Smoothing and Random Label Smoothing over a batch of labels during training for a classification model (active label in bold). Traditionally, the active label is set to“1" while all other classes’ indices are set to ‘0". In Label Smoothing, the active class is set to a constant (and higher) value, while the inactive classes are set to a smaller invariant value. In Random Label Smoothing, the active label receives a random (and higher) value (e.g., greater than 0.5) while the inactive labels receive a random value that, summed with the active label value, reaches 1. 84

List of Tables

Table 1 – Summarization of the approaches considered in the survey. 28

Table 2 – Error (in %) for each classification dataset using different methods and models. The following acronyms were used: MM = ManifoldMixup, PBA = Population Based Augmentation, RE = RandomErasing, BA = BatchAugmentation, AA = AutoAugment, Fast AA = Fast AutoAugment, and RA = RandAugment. 50

Table 3 – Summarization of the approaches considered in the survey and their respective source codes. 55

Table 4 – Results of MaxDropout and other regularizers 66

Table 5 – Results of the MaxDropout combined with Cutout. 66

Table 6 – Results of Dropout and MaxDropout over the WRN. 67

Table 7 – Ablation results concerning MaxDropout over ResNet18. 67

Table 8 – Average classification error rate (%) over CIFAR-10 and CIFAR-100 datasets. Notice ResNet18 results are provided as the baseline. 76

Table 9 – Average classification error rate (%) over CIFAR-10 and CIFAR-100 datasets combining MaxDropout and MaxDropoutV2 with Cutout. . . 77

Table 10 – Time consumed in seconds for training ResNet-18 in CIFAR-10 dataset. 78

Table 11 – Time consumed in seconds for training ResNet-18 in CIFAR-100 dataset. 78

Table 12 – Mean error (%) concerning CIFAR-10 dataset. 78

Table 13 – Mean error (%) concerning CIFAR-100 dataset. 79

Table 14 – Image classification experiment over CIFAR-100 dataset. 87

Table 15 – Results on CIFAR-100 using ResNet-18 with one or more regulization methods. 88

Table 16 – Results on CIFAR-100 using PyramidNet with one or more regulization methods. 88

Table 17 – PSNR results on Div2K and RealSR datasets for EDSR using regulization methods. 89

Table 18 – Results on Zurich RAW to RGB Dataset for PyNet. 89
Table 19 – List of publications developed by the author. 92

Contents

1 INTRODUCTION **19**
1.1 Hypothesis **20**
1.2 Thesis Organization **20**

2 RELATED WORKS **23**
2.1 Abstract **23**
2.2 Introduction **24**
2.2.1 How regularization works 24
2.2.2 Regularization vs. Normalization 26
2.2.3 Scope of this work 26
2.2.4 Comparison with Other Works 29
2.2.5 Where do regularizers work primarily? 29
2.2.6 Lack of Label Regularizers 30
2.3 Convolutional Neural Networks **31**
2.4 Regularization based on data augmentation **33**
2.4.1 Cutout 33
2.4.2 RandomErasing 33
2.4.3 AutoAugment 34
2.4.4 PBA 35
2.4.5 RandAugment 35
2.4.6 Mixup 35
2.4.7 CutMix 36
2.4.8 CutBlur 36
2.4.9 BatchAugment 37
2.4.10 FixRes 37
2.4.11 Bag-of-Tricks 37
2.5 Regularization based on internal structure changes **38**

2.5.1 Dropout and variants 38
2.5.2 MaxDropout 38
2.5.3 DropBlock 38
2.5.4 TargetDrop 39
2.5.5 AutoDrop 39
2.5.6 LocalDrop 39
2.5.7 Other methods 40
2.5.8 Shake-Shake 40
2.5.9 ShakeDrop 40
2.5.10 Manifold Mixup 40
2.6 Label Regularization 41
2.6.1 Label Smoothing 41
2.6.2 TSLA 42
2.6.3 SLS 42
2.6.4 JoCor 42
2.7 Methodology 43
2.7.1 Datasets 43
2.7.2 Architectures 44
2.8 Experimental Results 47
2.8.1 State-of-the-art Regularizer? 51
2.8.2 Defining a basic protocol 51
2.8.3 Use of minor architectures 52
2.8.4 Use of more complex datasets 52
2.8.5 Other problems besides classification 53
2.8.6 Source Code Links 53
2.9 Conclusion 56

3 MAXDROPOUT 57
3.1 Abstract 57
3.2 Introduction 58
3.3 Related Works 59
3.4 Proposed Approach 60
3.5 Experiments 62
3.5.1 Neural Network Structure 63
3.5.2 Training Protocol 63
3.5.3 Datasets 64
3.6 Experimental Results 64
3.6.1 Training Evolution 65
3.6.2 Comparison Against Other Regularizers 65
3.6.3 Working Along with Other Regularizers 66

3.6.4 MaxDropout x Dropout 66
3.6.5 Ablation Study 67
3.7 Discussion 67
3.8 Conclusions and Future Works 68

4 MAXDROPOUTV2 69
4.1 Abstract 69
4.2 Introduction 70
4.3 Related Works 71
4.4 MaxDropoutV2 as an improved version of MaxDropout 72
4.4.1 MaxDropout 73
4.4.2 MaxDropoutV2 73
4.5 Methodology 74
4.5.1 Dataset 74
4.5.2 Experimental Setup 75
4.6 Experimental Results 76
4.6.1 Classification Error 76
4.6.2 Combining Regularization Techiniques 77
4.6.3 Performance Evaluation 77
4.6.4 Evaluating Distinct Drop Rate Scenarios 78
4.6.5 Discussion 79
4.7 Conclusion and Future Works 80

5 RETHINKING REGULARIZATION WITH RANDOM LABEL SMOOTHING 81
5.1 Abstract 81
5.2 Introduction 81
5.3 Related Works 82
5.4 Proposed Approach 83
5.4.1 Image Classification 84
5.4.2 Image Super-Resolution 84
5.5 Methodology 85
5.5.1 Scenarios 85
5.5.2 Neural Network Architecture 86
5.5.3 Training Protocol 86
5.5.4 Datasets 86
5.6 Experimental Results 87
5.6.1 Image Classification 87
5.6.2 Image Super-Resolution 88
5.7 Discussion 89

5.7.1 Lack of Label Regularization Methods 89
5.7.2 Lack of Comparison for General-purpose Applications 90
5.8 Conclusions and Future Works 90

6 CONCLUSION 91
6.1 Publications and Other Works 92
6.2 Future Works 93

Chapter 1

Introduction

Machine Learning models have been used for several years to solve different types of problems, such as classification (**??**), data reconstruction (**??**), and biometry (**??**). To use such techniques, it usually took some steps: first, it needed data preparation, such as normalization; then another process called feature extraction was applied to detect some relevant structures in the data; finally, the machine learning model was trained using the information retrieved from the feature extraction process. This process is still relevant if the information is structured, such as database information or sheets, however, for unstructured data, such as images or sound, it is not the most effective process to achieve a relevant result.

Deep neural networks, also known as Deep Learning, have achieved state-of-the-art results in several tasks related to unstructured data, such as image classification (**??**), natural language processing (**??**), and image reconstruction (**??**). Although these models still use data preparation, the process of feature extraction and the final task, such as classification, is done during training. These neural networks are assembled in several layers, which perform the feature extraction, and then the final layer performs the desired task. The process of generating a model is more simple, however, it comes with more costs for training because it needs more data to generate layers that extract relevant information, and computational costs to process all the data.

The Convolutional Neural Networks (CNNs) are a set of Deep Learning models often used to solve computer vision problems, such as image classification (**??**), object detection (**??**), and image super-resolution (**??**). It was originally created to classify digits from the MNIST data set (**??**), but now it is widely used in several different problems. Its success in computer vision problems is due to the use of a convolutional process among the layers, which can generate and detect relevant features in image-related data after the training process.

One key aspect of the Deep Learning models is that, usually, it needs a lot of labeled data to create reliable models. In some cases, it is really hard to generate data for training for different reasons. One case is to generate medical images to detect some diseases. For instance, a magnetic resonance image has two costs problems: one is to generate the image itself, which values can be a problem; the other point is to ask for the help of a specialist to label the image correctly. These difficulties can be a problem for Deep Learning models because they need a lot of labeled data to achieve the desired results.

The data scarcity can lead to a problem known as overfitting, which means the model can achieve good results on the training set but it does not reflect on the validation data or in real-world applications. One way to solve this problem is to use regularization algorithms to diminish the overfitting problem.

Regularization is defined as a set of techniques that lead to more difficulties in training time but generate more reliable models to use (**??**). One simple way to apply regularization is artificial data augmentation, where one can, for instance, apply some transformation in the input data without corrupting the semantics. For example, for image processing problems, one can move the image in some pixels to generate another picture with the same relevant information. Even this very simple transformation can lead to more accurate models (**??**).

1.1 Hypothesis

The hypothesis and contributions of the present thesis regard answering the following questions: a) could the Dropout logic be changed, i.e., instead of randomly dropping neurons, using another dropping police, improve neural networks results? And b) could a random change in the labels generate neural networks with better results? Three new algorithms are proposed to answer the questions. i) a new logic to drop neurons during training time, based on the maximum output tensor values, ii) an adaption of the previous method that considers the tensor structure for convolutional neural networks, and iii) a label level regularization. The results shown in the following chapters support the hypothesis. This thesis is formed from a collection of works published and submitted by the authors during the research period.

1.2 Thesis Organization

Chapter 2 presents an in-depth analysis of regularization methods developed for and applied for convolutional neural networks, comparing results of recently developed algorithms (no older than 5 years). In this research, thorough verification of the methods shows what are the strongest points of the research in the area and what could be done to improve the research related to regularization.

Chapter 3 presents a new regularization method, called MaxDropout, which regularizes the neural network based on the maximum output value of a given tensor, forcing the Deep Learning model to learn features from minor values.

The paper presented in Chapter 4 is an improvement of the previously mentioned method, targeting specifically the drop for tensors generated by the convolutional process, achieving similar results to the prior method, however, being faster during training.

Regularization on label level is not known as a common approach, however, in Chapter 5 a new algorithm is demonstrated. It works by changing the values on the label during training time, improving results on image classification, image super-resolution, and software ISP via Deep Learning. Finally, Chapter 6 supplies a conclusion, other contributions of this thesis, and future works.

Chapter 2

Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks

This chapter presents the content publish in the journal ACM Computing Surveys (**??**), and it is used as theoretical background of this study, listing recent regularization algorithms, their strenghts and some problems in the area.

2.1 Abstract

Several image processing tasks, such as image classification and object detection, have been significantly improved using Convolutional Neural Networks (CNN). Like ResNet and EfficientNet, many architectures have achieved outstanding results in at least one dataset by the time of their creation. A critical factor in training concerns the network's regularization, which prevents the structure from overfitting. This work analyzes several regularization methods developed in the last few years, showing significant improvements for different CNN models. The works are classified into three main areas: the first one is called "data augmentation", where all the techniques focus on performing changes in the input data. The second, named "internal changes", which aims to describe procedures to modify the feature maps generated by the neural network or the kernels. The last one, called "label", concerns transforming the labels of a given input. This work presents two main differences comparing to other available surveys about regularization: (i) the first concerns the papers gathered in the manuscript, which are not older than five years, and (ii) the second distinction is about

reproducibility, i.e., all works refered here have their code available in public repositories or they have been directly implemented in some framework, such as TensorFlow or Torch.

2.2 Introduction

Convolutional neural networks (CNNs) have achieved relevant results on several computer vision-related tasks, such as image classification and object detection in scenes. Such success can be explained by how the convolutional neuron works: it highlights given features according to the spatial properties of the image. The initial layers highlight less complex features, such as borders; however, more dept layers can detect more complex traits, like entire objects or faces of people. Nowadays, it is hard to find any other computer vision technique applied without any CNNs, from biometrics to disease detection.

One key aspect concerning CNNs is how to stack the convolutional kernels to accomplish the best result on a given task. It is widespread to use the same basic architecture on several different tasks, just changing the output. For instance, the basic block used for EfficientNet (**??**), a neural network used for image classification, is also used on the Efficient-Det (**??**) architecture to tackle the object detection task.

The architecture may be the central part of a computer vision model; however, there are other relevant points before starting the training step. For instance, the optimization technique can influence the final result. Even the kernels' initial random values can influence how well the model will perform in the end. This study focuses on one of these aspects that can influence the final result: the regularization algorithms. Depending on the chosen regularization strategy used, some architectures can achieve a relevant gain on the final results. One important aspect of using a good regularizer is that it does not influence the final model's performance. It means that, independently of using or not one regularizer, the model's computational cost for inference is the same. However, in some cases, it can influence performance during the training phase, using a little computational overhead or pre-train epochs. In any way, the results of the output usually overcompensate this cost.

2.2.1 How regularization works

CNNs are usually used for computer vision tasks, such as image classification and object detection, to create models as powerful as human vision. If the amount of information available is considered, it becomes clear the training task requires more data variability than possible. Considering a healthy human with a regular brain and eyes, we retain new information around 16 hours per day, on average, disregarding the time we sleep. Even considering huge datasets such as ImageNet, the number of images available is minimal compared to the quantity of data a human brain receives through the eyes. This unavailability of new data may lead to a situation known as overfitting, where the model learns how to represent well the training data, but it does not perform well on new information, i.e., the test data. This situation usually happens when the model has been trained exhaustively in the available training information that it cannot generalize well in other new information.

As an artificial neural network, the training step of CNNs can be described as an optimization problem, where the objective is to find out the weight values which, given an input and a loss function, can transform the information in the desired output, such as a label, with the lowest possible error. One way to achieve this goal is to minimize the following function:

$$\min_{U,V} ||X - WY^T||_F^2, \tag{1}$$

where $||.||_F^2$ is the Frobenius norm, $X \in \mathbb{R}^{m \times n}$ defines the input data, and $W \in \mathbb{R}^{m \times d}$ and $Y \in \mathbb{R}^{n \times d}$ denote the weight matrix and the target labels, respectively. According to (**??**), the Frobenius norm imposes the similarity between X and WY^T. This interpretation has one main advantage: this formulation enables the optimization through matrix factorization, producing a structured factorization of X. However, it is only possible to achieve a global minimum if W or Y^T is fixed for optimizing both matrices together converts the original equation into a non-convex formulation. This problem can be solved if the matrix factorization is changed to a matrix approximation as follows:

$$\min_{A} ||X - A||_F^2, \tag{2}$$

where the target is to estimate the matrix A, which ends up in a convex optimization, meaning it has a global minimum that can be found via gradient descent algorithms. When using regularization, this equation becomes:

$$\min_{A} ||X - A||_F^2 + \lambda\Omega(A), \tag{3}$$

where $\Omega(\cdot)$ describes the regularization function based on A, and λ is the scalar factor that sets how much influence the regularization function infers on the objective function.

One key aspect of the regularization methods, independent of the training phase it works, is to prevent the model from overfitting the training data. It operates by increasing the variability of the data on different stages of a CNN. When working with images, the most straightforward method is random image changing, like rotation and flipping. Several deep learning frameworks, such as Keras and TensorFlow, have their implementation available, facilitating this kind of regularization and improving the results. Although this type of regularization works well, some points should be taken into consideration. For example, some transformations may distort the image into another existing class in the classification. The more straightforward example is baseline image classification on the MNIST data set: if the rotation is too several, an input "6" may be transformed into a "9", leading the model to learn wrong information.

2.2.2 Regularization vs. Normalization

A general problem in machine learning is to tune the parameters of a given model to perform well on the training data and eventually new information, i.e., the test set. The collection of algorithms that aims to reduce the error on the data that does not belong to the training set is called regularization techniques.

One main difference between the normalization and regularization techniques is that the second is not performed after the training period, while the first is kept in the model. For example, Cutout (**??**) and MaxDropout (**??**) original codes show they do not execute anything during the inference, but the BatchNormalization (**??**) executes its algorithm in deducing the test set.

2.2.3 Scope of this work

This study focuses on the most recent regularization techniques for CNNs. Other studies (**????**) focus on older and more general regularization methods. Here, we consider three main points:

- ❏ *Recently developed*: besides Dropout (**??**), no other study is older than four years, making this study very much up-to-date;
- ❏ *Code availability*: all related algorithms in this study are available in some way, usually on Github. We considered it an essential point because it avoids studies with possibly inaccurate results and allows reproducibility when necessary;
- ❏ *Results*: all techniques here were able to improve the results of the original models significantly.

In this work, the regularization algorithms are divided into three main categories, each one in a given section: the first one is called “data augmentation", and it describes the techniques that change the input of a given CNN. The second category is called “internal changes", and it describes the set of algorithms that changes values of a neural network internally, such as kernel values or weights. The third category is called “label", in which techniques perform their changes over the desired output. Table 1 gives a list of all methods discussed in this work.

Reference	Short Name	Description	Where
(??)	Bag of Tricks	Combines several regularizers to show how it improves CNN	Input
(??)	Batch Augment	Increases the size of the mini-batch	Input
(??)	FixRes	Performs train and test with different image sizes	Input
(??)	Cutout	Removes part of the image	Input
(??)	CutMix	Replaces part of the image using other parts of other images	Input / Label
(??)	RandomErasing	Replaces part of the image by noise or paint the region	Input
(??)	Mixup	Mixes two images from different classes	Input / Label
(??)	AutoAugment	Learns how to provide better data augmentation based on information from the training data set	Input
(??)	Fast AutoAugment	Reduces the training time of the agent from (??)	Input
(??)	RandAugment	Learns augmentation policies during training	Input
(??)	PBA	Population Based algorithm for data augmentation	Input
(??)	CutBlur	Replaces regions from high-resolution images with low resolution pieces	Input / Label
(??)	Dropout	Drops random neurons	Internal
(??)	MaxDropout	Drops neurons based on their activation	Internal
(??)	GradAug	Trains sub-networks from the original CNN	Internal
(??)	Local Drop	Dropout and DropBlock based on the Radamacher complexity	Internal
(??)	Shake-Shake	Gives different weights to each branch of the residual connection	Internal

(??)	ShakeDrop	Improves Shake-Shake by generalizing to other models	Internal
(??)	Manifold Mixup	Act like Mixup, however, in the middle layers of a CNN	Internal / Label
(??)	DropBlock	Drops entire regions from a tensor	Internal
(??)	AutoDrop	Learns drop pattern	Internal
(??)	Label Smoothing	Replaces one-hot encoding vectors to smoothed labels	Label
(??)	TSLA	Two-stage algorithm for label smoothing	Label
(??)	SLS	Quantifies label smoothing based on feature space	Label
(??)	JoCoR	Co-relates labels for label smoothing	Label

Table 1: Summarization of the approaches considered in the survey.

Although we divided the methods into three different strategies, Table 1 highlights that some algorithms work on two different levels. For instance, CutMix and CutBlur work on both input and label levels. The majority of the methods work on input or internal structures, which shows a lack of research on label regularization methods.

2.2.4 Comparison with Other Works

In a quick search, it is possible to find a diversity of works using Convolutional Neural Networks, such as image classification (????????), object detection (????), and image reconstruction (??????). However, the frequency of works for regularization compared to other problems is very low. As far as we are concerned, we found only two recent surveys about regularization for deep neural networks.

The first one (**??**) is an extensive analysis of regularization methods and their results. Although it is an interesting work, it focuses considerably on older methods, such as adding noise to the input, DropConnect (**??**) and Bagging (**??**). Those methods are still broadly used and have their importance; however, they are not exactly new.

Another relevant work found was a survey focused only on dropout-based approaches (**??**). Dropout (**??**) is undoubtedly an important regularization method for different types of neural networks, and it has influenced several new approaches over the years, besides being used in several different architectures.

In this work, we show very recent developments on strategies for improving the results of Convolutional Neural Networks. As one can observe, it presents works as recent as the one published on 2021 (**????**). The following sub-sections present more insights and statistical information about the works surveyed in the manuscript.

2.2.5 Where do regularizers work primarily?

Even though most of the works are applied to the input, there are many studies dedicated to internal structures and the label layer. Figure 1 depicts the proportion of the scientific works presented in this survey.

Around 44% of the works relies on changes on the input, most known as data augmentation strategies. The easiness of changing parameters and structures in a CNN's input may explain such an amount of works. Image processing- and computer vision-driven applications still play a significant role when dealing with deep learning. The second most common regularization approaches stand for the ones that perform changes in the internal structures. Dropout (**??**) contributed considerably to advance in this research area. Several works (**??????**) are mainly based on Dropout, while some of them (**????**) are new approaches.

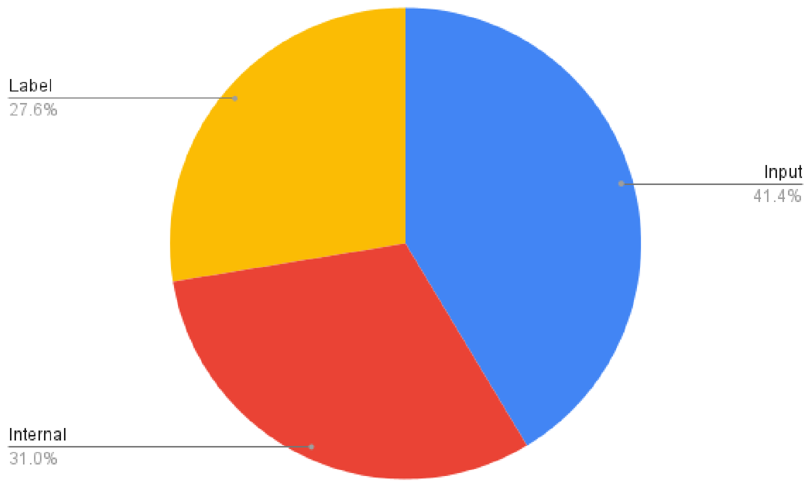


Figure 1: Percentage of the regularization works surveyed in the manuscript.

2.2.6 Lack of Label Regularizers

We want to highlight the importance of more research on regularizers that work on a neural network label level. Although around 22% of the works make changes on the label as a regularization strategy, we found two relevant works on the area only (**??**). Some hypotheses may be raised here.

The first one is that the label level is not intuitively changed as the input or in the middle-level of a neural network. Performing changes in both levels is more natural, for it is visually more obvious to understand what is going on during training and inference. However, it is harder to explain what happens when label changes are performed. Even though the original work (**??**) argues that it prevents the overconfidence problem, it fails to explain why such a situation is avoided.

Another explanation is the lack of mathematical explanation for most approaches. Fortunately, some techniques such as Dropout (**??**) and Mixup (**??**) present interesting insights about their inner mechanism. An algebraic proof that label smoothing works well may be an essential step for the development of new strategies concerning the last level's regularization.

Finally, it is always good to remember that one of the most critical steps for developing a machine learning area is creating reliable-labeled datasets. Although we focused on regularization strategies, it is worth remembering that, eventually, a breakthrough on the way we work with labels may lead to more powerful systems. Therefore, we emphasize that more works related to the label-level regularization are worth researching.

2.3 Convolutional Neural Networks

Neural networks have been used since the 1950s when the first neuron emulation, called Perceptron (??), was developed. However, it can primarily address linearly separable feature spaces. However, in the 1980s, the development of the backpropagation algorithm (??) to set new values in a structure that uses several Perceptrons in more than one layer, called the Multilayer Perceptron (MLP), made it possible to solve non-linear problems as well. Even with these advances, it still lacks some relevant results to solve unstructured data problems, such as images.

In late 1990, a new neuron structure emerged based on the 2D convolution process, the so-called Convolutional Neural Network (??). The 2D convolution process can find different features on an image, depending on the convolutional kernel's size and values. What makes a CNN so valuable for image processing is the possibility of stacking convolutional processes to find out different features whose training can be accomplished using the well-known backpropagation algorithm. Figure 2 illustrates a standard structure of a Convolutional Neural Network.

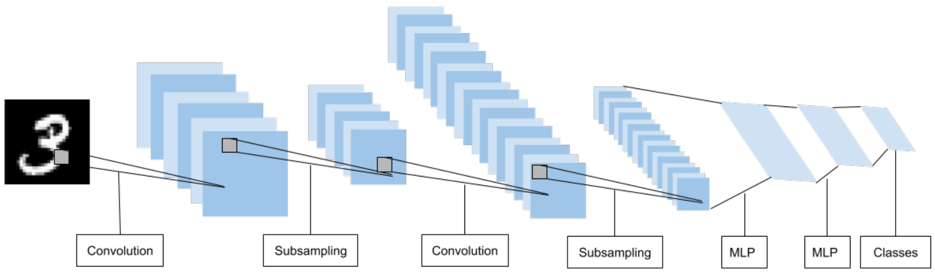


Figure 2: The LeNet-5 structure. Inspired in the picture from (??).

Even being so powerful, it still needs lots of data to achieve relevant results, thus requiring considerable computational power. In the middle of the 2000s, GPUs' use accelerated the training process hugely, becoming possible to solve image processing problems in a feasible time. From 2010, the first relevant result emerged. The AlexNet structure (??) achieved first place in the Image Net classification challenge, overcoming the runner-up result by more than 10%. It is an 8-layer CNN with an MLP on top to perform the classification. Since then, other CNN structures have appeared, each one with new features in their structure.

The Visual Geometry Group developed the VGG architecture (??) which demonstrates, for the first time, that stacking convolution layers with smaller kernels perform better than shallow layers with bigger kernels, even then performing over the same region. This architecture achieved first place in the ImageNet classification challenge in 2012. Another architecture family with relevant results is the Inception (??????), which was developed by Google by parallelizing kernel operations in the same layer and then fusing them before the next layer.

About the same time the first Inception architecture showed up, Microsoft presented the Residual Network, most known as ResNet (**??**). It works by fusing the output of layers with the same dimensions before the pooling operation. It looks like a simple operation at first, but later on, it has been shown that this residual connection helps the backpropagation algorithm to handle better the well-known vanishing/exploding gradient shortcoming (**??**).

Neural Architecture Search, known as NAS (**??**), developed a new way to find better CNN architectures. Using an agent trained by the Q-Learning technique (**??**), it can find out the CNN that can achieve the best result according to some rules. The drawback of this technique is that it takes a considerable amount of time to discover the best neural network architecture. However, recent studies (**??**) showed how to improve the search algorithm, making it faster to discover new architectures.

Later in 2018, Google showed the NAS could be improved when some rules are better designed, such as the computational limit, input size, and other parameters, and incorporate other architectures, such as Squeeze-and-Excitation (**??**), ending up in the EfficientNet family (**??**). The original work showed eight different architectures (called B0-7), which perform using the same quantity of floating points operation (FLOP) as other architectures but achieving better results. In the same study, the EfficientNet architectures delivered state-of-the-art results in five different datasets.

All works discussed until now operate on the image classification problem. However, CNN's can be used in several other tasks. One interesting problem is object detection in natural scenes. The R-CNN (**??**), for instance, works in two stages, being the first to find interest regions on the image, and the final stage classifies each region in the desired objects. The You Only Look Once, known as YOLO (**??**), goes one step further and performs the localization and classification steps in the same stage.

Another task well solved by CNN concerns image reconstruction. In this case, most of them are Fully Convolutional Networks (FCN), which means that every single layer on the neural network is a convolutional layer. One relevant work in this area is the Residual Dense Network, which has a version for super-resolution (**??**) and image denoising (**??**) purposes. Another significant development is the DnCNN (**??**), which not only resolves problems for image denoising, JPEG deblocking, and super-resolution but has a version that can solve the three problems without any information about the input image, performing a blind reconstruction.

The Generative Adversarial Network (GAN) was first developed using MLP (**??**); however, it is used mainly with convolutional layers to solve diverse problems. One problem tackled by GAN is the style transfer, in which the StackGAN (**??**) shows a very nice result, being able to change the style completely without losing relevant information. Another work with good results is the ERSGAN (**??**), which deals with the super-resolution of images. The neural network shows outstanding results by training a Residual-in-Residual Dense Network (RRDN) using the GAN approach.

2.4 Regularization based on data augmentation

When thinking about changes in the training data for CNNs, maybe the most intuitive way is to perform alterations on the input for all changes can be visualized (or at least imagined) before the beginning of the model's training. Since the first CNN model (**??**), basic data augmentation, such as flipping and noise adding, has proven it can help the trained model generalize better.

2.4.1 Cutout

One straightforward but powerful technique to perform data augmentation is the well-known Cutout (**??**). During training, it randomly removes regions of the image before feeding the neural network. In (**??**), the authors exhaustly analyzed what would be the ideal size of the removed region in the CIFAR-10 and CIFAR-100 datasets. The ideal size varies according to the number of instances per class and the number of classes for a given dataset. For example, the best results on the CIFAR-10 dataset were accomplished by removing a patch of size 16×16, while for CIFAR-100 the region size concerning best results was 8×8. For the SVHN dataset, the best crop size was found out by using a grid search, which outputs the 20×20 size as ideal. Regarding the STL-10 dataset (**??**) the cut size for the best result was 32×32. Figure 3 shows how Cutout works.

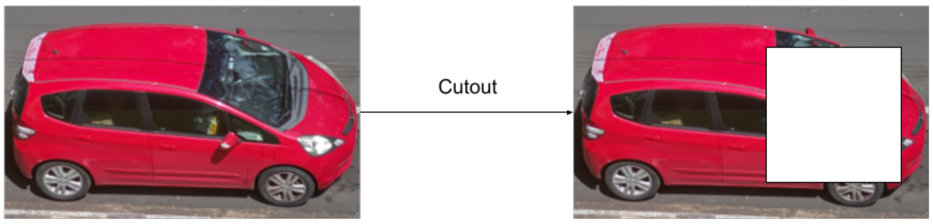


Figure 3: How Cutout works. Extracted from (**??**)

2.4.2 RandomErasing

RandomErasing (**??**) was further developed based on the Cutout technique. While the latter removes random crops of the image, RandomErasing is concerned about removing and randomly adding information on the blank space, such as noise. Different from Cutout, RadomErasing does not remove pieces of the image every time. In this work, the authors evaluated the method on three different classification datasets (CIFAR-10, CIFAR-100, and Fashion-MNIST), the PASCAL VOC 2007 (**??**) dataset for object detection, and three different CNN architectures for person re-identification (IDE (**??**), TriNet (**??**), and SVDNet (**??**)). For the classification task, four different architectures were used for evaluation purposes: ResNet (**??**), ResNet with pre-activation (**??**), Wide Residual Networks (**??**) and ResNeXt (**??**), including four distinct setups for the two first architectures. In all cases, the RandomErasing approach accomplishes a relevant error reduction (at least 0.3%). For the object detection task, the mean average precision (mAP) was increased by 0.5 when the model was trained only with the available data from the dataset and 0.4 gain when the training data was combined with the PASCAL VOC 2012 training dataset (**??**). Figure 4 shows how RandomErasing works.

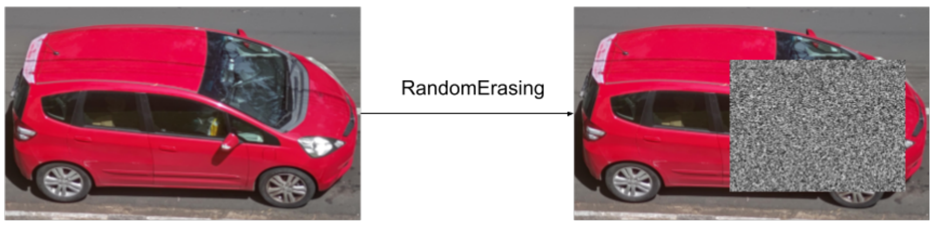


Figure 4: How RandomErasing works. Extracted from (**??**).

2.4.3 AutoAugment

AutoAugment (**??**) tries to find out what transformations over a given data set would increase the accuracy of a model. It creates a search space for a given policy using five different transformations ruled by two additional parameters: the probability of applying a given alteration (which are: Cutout, SamplePairing, Shear X/Y, Translate X/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, and Sharpness) and the magnitude of this change. These policies are then fed into a "child" model, which is a CNN trained with part of the training data set. The accuracy of this CNN is informed to a "controller" model, which is a Recurrent Neural Network (RNN) - more specifically, a Long-Short Term Memory. This RNN outputs the probabilities of a given policy to be used in the future. At the end of the controller training procedure, the five best policies (each one with five sub-policies) are used to train the final model used to evaluate the data set. Using these generated policies and sub-policies, AutoAugment accomplished state-of-the-art results on CIFAR-10, CIFAR-100, SVHN, and ImageNet datasets. One huge advantage of this approach is the transferability of these policies across different datasets: in the original work, the policies found out for ImageNet were used to train five other different datasets, improving the results significantly even when the AutoAugment technique was not trained on them.

One disadvantage of this approach is the time used to train the controller model: for the ImageNet dataset, for instance, it took around 15,000 hours of processing, which may be impracticable in several cases. Fast AutoAugment (**??**) aimed at overcoming such a bottleneck with a new algorithm, reducing the time related to the search procedure significantly, besides producing similar results.

2.4.4 PBA

Population Based Augmentation (PBA) (**??**) not only showed a novel augmentation algorithm but demonstrated schedule policies instead of fixed policies, which improves the results from the previous studies (**????**). At every 3 steps, it changes half the policies, being 1/4 changes in the weights and the other 1/4 a change in the hyperparameter. While AutoAugment implies an overhead of 5,000 hours for training over the CIFAR-10 data set, PBA increases it by only 5 hours.

2.4.5 RandAugment

As mentioned before, a huge bottleneck for the methods which look for finding the best data augmentation involves their computational burden since it may take longer than the own neural network training. Another problem is related to the strategies found during the search, which may end up in a sub-optimal strategy, i.e., it does improve the results locally; however, it does not lead to the best global result for it uses a shallower neural network and assumes that this rule can be applied to any other, and possibly, deeper architecture. RandAugment (**??**) uses the 14 most common policies found on previous works (**??????**) and performs the search of the magnitude of each policy during training, thus removing the need for a preliminary exploration step and tailoring the data amplification to the current training CNN. Results show that the method is not only faster than previous approaches (**??????**) but improves the outcomes significantly.

2.4.6 Mixup

One possibility for training CNN concerns mixing two images from the training data set and forcing the model to determine which class this mixture belongs reliably. However, it is not widespread how to generate the encoding label for such a mixture. Providing this new input/output training pair allows the model to learn more features from corrupted inputs. The original work shows that models using such an approach can improve results not only in the image classification task but in speech recognition, stabilization in generative adversarial networks, tabular datasets, and other problems. Figure 5 demonstrates how mixup works.

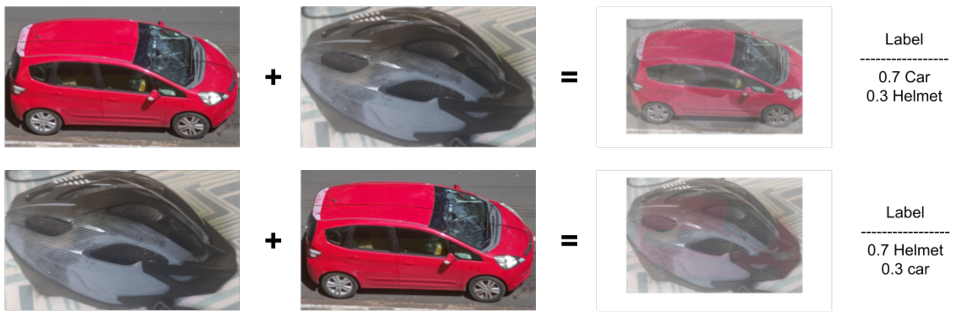


Figure 5: Two different examples using Mixup. Extracted from (**??**)

2.4.7 CutMix

Another strategy to mix inputs and labels to improve the results is the CutMix (**??**). Unlike Mixup, CutMix replaces entire regions from a given input and changes the label by giving the same weights as the area used by each class. For example, if a cat's image is replaced in 30% by an image of an airplane, the label is set to be 70% cat and 30% airplane. This strategy shows a significant improvement in results. By using techniques that map the most activated regions (e.g., grad-CAM (**??**)), one can observe that the generated heat maps highlight better the areas that define the object of interest more accurately. Figure 6 illustrates the technique.

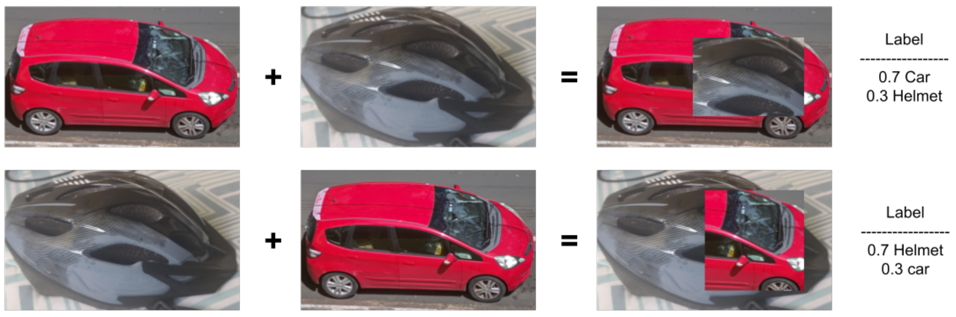


Figure 6: How CutMix works. Extracted from (**??**).

2.4.8 CutBlur

Several Deep Learning tasks targeting image processing, such as image classification or object detection, can improve their models by using data augmentation. Several works, such as AutoAugment (**????**), Cutout (**??**), and RandomErasing (**??**) can improve results significantly by applying some clever but straightforward transformations on the training images. However, for super-resolution (SR) tasks, the literature lacks works that proposed regularization techniques to handle the problem explicitly.

Even though the aforementioned techniques can be used and possibly improve results, they are not natively designed to cope with the SR problem. The only approach found, so far, is the CutBlur (**??**), which works by replacing a given area on the high-resolution image (HR) with a low resolution (LR) version from a similar region. The authors showed that CutBlur helps the model generalize better on the SR problem but that the same technique can be applied to reconstruct images degraded by gaussian noise.

2.4.9 BatchAugment

One important hyperparameter for training CNNs concerns the mini-batch size, which is used to calculate the gradient employed in the backpropagation. Usually, the GPU's upper limit is employed for such a hyperparameter, which is crucial to speed up the convergence during training. The Batch Augmentation work (**??**) cleverly uses this limit. Instead of just fulfilling the entire memory with different instances from the dataset, it considers half of the memory limit using the default set up for data augmentation and then duplicates all instances with different data augmentation possibilities. It sounds like a straightforward technique; however, results demonstrate that neural networks that use such an approach have a significant improvement on the final results. Another point is that, by duplicating the augmented images, the analysis showed that it is necessary fewer epochs for convergence.

2.4.10 FixRes

The image resolution may influence both the training step efficiency and the final classification accuracy. For instance, the research on EfficientNet (**??**) highlights this idea by making the input size one of the parameters that influence the final result. However, if a model is trained, for example, with a resolution of 224×224, the test set's inference uses the exact resolution. The work proposed by (**??**) highlighted that the resolution of the test set should be higher than the resolution used for training. This change not only produces a more reliable neural network but it trains faster than the traditional approach, for it requires less computational effort for training since the images used for such a purpose are smaller than the ones used for inference. The proposed approach shows it can improve the results on other datasets when transfer learning is used.

2.4.11 Bag-of-Tricks

One critical point of the works analyzed here is that they frequently do not combine any other regularizer with their current research. Hence, it is hard to know how two regularizers can influence one another. The Bag of Tricks research (??) performs this investigation by combining several known regularization methods, such as Mixup (??), Label Smoothing (??) and Knowledge Destilation (??). The ablation study shows that if some cleverness is applied, the final result can be significantly improved. For instance, a MobileNet (??) using this combination of methods improved its results by almost 1.5% in the ImageNet dataset, which is a significant gain. However, the research lacks a deeper evaluation of methods for regularization among layers, such as Dropout (??).

2.5 Regularization based on internal structure changes

Regularization methods can work in different ways. In this paper, we define internal regularizers as the ones that change the weights or kernel values during training without any explicit change on the input. This section is divided into two main parts: the first presents a deeper description of how dropout works and some of its variants, such as SpatialDropout and DropBlock. In the second part, we describe other methods that aim to perform other tensors' operations, such as Shake-shake regularization.

2.5.1 Dropout and variants

Dropout (??) was proposed as a simple but powerful regularizer that aims to remove some neurons, thus forcing the entire system to learn more features. The original work shows it can be applied not only on CNNs but in Multilayer Perceptrons (MLPs) and Restricted Boltzman Machines (RBMs). The probability of dropping out each neuron is estimated through Bernoulli's distribution at each step of the training phase, thus adding some randomness in the process. The original work shows dropped neural networks can generalize better than standard ones.

2.5.2 MaxDropout

While Dropout (??) randomly removes the neurons in the training phase, Maxdropout (??) deactivates the neurons based on their activations. It first normalizes the tensor's values and then sets to 0 every single output greater than a given threshold p, so the higher this value, the most likely it to be deactivated. The original work shows it can improve ResNet18 results on CIFAR-10 and CIFAR-100 (??) datasets, and it also outperforms Dropout on the WideResNet-28-10 model (??).

2.5.3 DropBlock

CNN works so well on images and related fields (such as video) that it can generate correlated regions among its neurons. For instance, in the image classification task, heatmaps generated by techniques like grad-CAM (**??**) show that, when correctly estimated, CNN highlights regions of interest around the object that has been classified. DropBlock (**??**) shows that removing entire areas of a given tensor (i.e., feature map) can help the model to generalize better. By using ResNet-50 and AmoebaNet-B models on the image classification task, RetinaNet on object detection, and ResNet-101 for image segmentation, it shows that it can improve results better than Dropout and other internal regularizers (**????**). DropBlock is applied on every feature map of the CNN, starting the training with a small ratio and slowly increasing its value. Its experiments show relevant results on the ImageNet dataset, increasing the baseline accuracy by almost 2% when using ResNet-50, beating other regularizers, such as Cutout and AutoAugment, and around 0.3% when using AmoebaNet-B. In the object detection task, the RetinaNet model is improved by more than 1.5 in the AP metric.

2.5.4 TargetDrop

Attention mechanism can be incorporated into a given regularizer so it can act in the appropriate region. For instance, the TargetDrop (**??**) combines this mechanism with DropBlock. During training, it allows the entire system to remove most discriminative areas on a given channel. Results show this method not only accomplishes better results than DropBlock but, by using grad-CAM (**??**), demonstrates more consistency in the region that determines to which class a given input belongs.

2.5.5 AutoDrop

Although effective, Dropout lacks spatial information for choosing what neuron to drop. DropBlock's strategy is to drop entire random regions on hidden layers instead of singular neurons, thus forcing a CNN to learn better spatial information. However, the drop pattern is manually designed and fixed, which may be improved if these patterns could be learned during training. AutoDrop (**??**) forces the CNN to learn the best drop design according to information from training by using a controller that learns, layer by layer, the best drop pattern. Results in CIFAR-10 and ImageNet show that these patterns improve results and can be transferred in between data sets.

2.5.6 LocalDrop

The Rademacher complexity was used to redefine both Dropout and DropBlock (**??**). After an extensive mathematical analysis of the problem, a new two-stage regularization algorithm was proposed. Although very time-consuming, the proposed method achieves relevant improvement on different CNN architectures targeting image classification.

2.5.7 Other methods

In the last few years, the use of residual connections, first introduced in the well-known neural architecture ResNet (**??**), and their further improvements (**????**) have achieved relevant results on several tasks. Later studies (**??**) have shown that such a success is due to the creation of a structure called "identity mapping", which is the reconstruction of the original input. The residual connection then forces the model to learn how to construct these structures.

2.5.8 Shake-Shake

One way to force regularization on these architectures is to give different weights to each branch of the residual connections during training. The original ResNets works by adding the weights on each branch without any differentiation. During training, Shake-shake (**??**) works on 3-branch ResNets by changing the multiplication factor of each branch on the forward pass and multiplying by a different value on the backward pass, thus changing how each branch affects the final result. For the inference, it multiplies each branch by a factor of 0.5.

Results on CIFAR-10 show that such an approach can improve outcomes by at least 0.15%, achieving almost 0.6% improvement on the best result. Results on the CIFAR-100 were improved by 0.4%; however, in this specific case, the removal of weights' changes on the backward pass ends up in slightly better results, improving by 0.5%. Besides the improvement, this method only works on 3-branches ResNet, making it hard to compare other methods directly.

2.5.9 ShakeDrop

One improvement to tackle the problems of Shake-shake is the ShakeDrop (**??**). It works not only on ResNeXt architecture but on ResNet, Wide ResNet, and PyramidNet too. To accomplish such results, ShakeDrop changes the formulation proposed by Shake-shake. The combination of these perturbations on the branches shows ShakeDrop has more tools not to be trapped on local minima. Results show that it can outperform the original results obtained by each architecture mentioned earlier.

2.5.10 Manifold Mixup

A neural network is usually generalized as a function that, given input data and a set of learnable parameters, outputs the target value accordingly. The Manifold Mixup (**??**) acts like the Mixup (**??**), however, operating in any internal layer of a CNN, and not only in the input layer. A deep neural network can be considered a set of smaller neural networks. Each one outputs some desired features; therefore, if all subnets work well, the final result can be regarded as a good one. Yang et al. (??) propose a new strategy to design the loss function: it first calculates the traditional loss of a mini-batch through the feedforward process. After that, it generates sub-networks from the original one and then computes one loss for each model by supplying the same mini-batch using different image transformations. Finally, the final loss is calculated by adding the traditional loss with the losses from each sub-network. This technique shows a great potential improvement in different datasets and CNN architectures.

2.6 Label Regularization

Revisiting some information on Table 1, other methods use label smoothing as part of their regularization strategy. For instance, Mixup (??) averages the values of the labels depending on the interpolation between two different images. The same rule is applied for the Manifold Mixup technique (??); however, the data interpolation is computed among the layers and the same calculus is used for resetting the label values.

Another regularizer that uses label transformation is Cutblur (??). In this case, the transformation is used so wisely that, during training, the label could be inverted with the input, making the input as the label, and the model would converge as expectedly. The reason for this expected result is due to the cut size of the low-resolution and high-resolution images, which are not defined beforehand. It means that the input can be a low-resolution image with a crop from the high-resolution image, and the label would be the high-resolution image with the crop from its low-resolution counterpart. Therefore, inverting the label and input still makes sense.

Other methods can also have their results improved by using some rationale borrowed from label smoothing. For instance, Cutout (??) removes parts from the input, so it makes sense to "remove" part of the label according to the crop size as well. Pretend the crop size is 25% of the image, so the active class could be dropped from 1 to 0.75. The same strategy can be applied to RandomErasing (??). Methods that drop neurons during training, such as Dropout (??) could, for example, drop the values of the hot label by the same range of the total active neurons deactivated during training.

2.6.1 Label Smoothing

It is widespread in a general classification task to use the one-hot vector to encode the labels. Dating back from 2015 (??), label smoothing proposes a regularization technique in the label encoding process by changing the value on each position of the hone-hot representation.

Label smoothing works by preventing two main problems. First, the well-known overfitting, i.e., the situation where the model learns the information about the training set but cannot generalize the classification in the test set. The second and less obvious is overconfidence. According to the authors (??), by using the smoothing factor over the encoding label, the softmax function applied over the vector produces values closer to the smoothed encoded vector, limiting the value used in the backpropagation algorithm and producing a more realistic value according to the class.

2.6.2 TSLA

One difficulty of using label smoothing is to find out what value of ϵ (i.e., smoothing factor) is the ideal, either for a general or for a specific data set. The original work suggests that $\epsilon = 0.1$ is the excellent condition; however, the Two-Stage Label Smoothing (TSLA) (??) suggests that, in general, the gradient descent combined with the label smoothing technique can only improve the results until a certain point of training, after that it is better to set all values to 0 and 1 for the active class. For instance, when training the ResNet18 in the CIFAR-100 data set for 200 epochs, results suggest the best performance is achieved when label smoothing is used until the epoch 160.

2.6.3 SLS

Usually, it is not straightforward to define appropriate values for the label smoothness factor. Structural Label Smoothing (SLS) (??) proposes to compute such a value by estimating the Bayes Estimation Error, which, according to authors, helps define the label's boundaries for each instance. Several experiments show that this approach can overcome the traditional label smoothing method on different occasions. Although the work is fully evaluated on MobileNet V2 (??), it does not consider other neural network architectures. Even though some popular data sets were used for comparison purposes, e.g., CIFAR and SVHN, the work is limited to MobileNet-V2 only.

2.6.4 JoCor

This work proposes a new approach to avoid the influence of noisy labels on a neural networks. JoCoR (??) trains two similar neural networks on the same data set and tries to correlate two different labels. The method calculates the loss by adding the cross-entropy losses of both networks plus the contrastive loss between them and then uses only the most negligible losses on the batch to update the parameter of the architectures. The authors argue that both networks agree with the predictions by using the smallest values to update parameters, and the labels tend to be less noisy. Although the method was developed for weakly supervised problems, it could easily fit traditional supervised problems, such as data classification, to improve outcomes. The downside of this method is using two neural networks for training, which requires more processing and memory.

2.7 Methodology

We provide a direct comparison among each regularizer described in this work. We divided each table by model for a more transparent comparison and then provided the result for each dataset available on the original work and related works. The results are shown on the classification task, showing how each algorithm performed in the most common datasets.

2.7.1 Datasets

The last important part of training a neural network and defining the baseline is to decide what dataset should be used for evaluation, either for training and validation. For the sake of research in regularization in image processing, two datasets are considered the most frequent, i.e., CIFAR, Imagenet, and SVHN.

2.7.1.1 CIFAR

The original CIFAR (Canadian Institute For Advanced Research) dataset consisted of 80 million images; however, due to some ethical problems, such as offensive and prejudicial images, the authors decided to make it unavailable [1]. Instead, two other subsets have been frequently used in regularization research: (i) CIFAR-10 and (ii) CIFAR-100.

The CIFAR-10 subset is compounded by $60,000$ 32×32 images, divided into $50,000$ figures for training and the remaining $10,000$ for test/validation. It is divided into ten classes between animals (bird, cat, deer, dog, frog, and horse) and objects (airplane, automobile, ship, and truck).

The CIFAR-100 is also built by $60,000$ 32×32 images, divided into $50,000$ figures for training and the remaining $10,000$ for test/validation. However, it is divided into 100 classes, being harder to classify than its counterpart version. Objects and animals also compound the classes. Besides, both versions of the CIFAR dataset use the same images for training and testing.

2.7.1.2 ImageNet

Ordinarily called a "dataset", the ImageNet is a project developed to improve artificial intelligence tasks, such as image classification. ImageNet dataset is usually associated with the "2012 ImageNet Large Scale Visual Recognition Challenge" (ILSVRC), which comprises $1,240,000$ $224x224$ images for training and $50,000$ for validation purposes divided into $1,000$ classes. It has one order of magnitude bigger than the CIFAR subsets.

[1] More information: http://groups.csail.mit.edu/vision/TinyImages/

ImageNet is historically relevant in the deep learning community, mainly for those who work with image classification, for it was in the 2010 ILSRVC that the first practical work using CNN was demonstrated: the AlexNet (**??**), a neural network compounded by convolutional and a multilayer perceptron (MLP) layer, trained end-to-end, achieved first place in the context, outperforming the runner-up by more than 10% in the accuracy metric. Since then, several deep learning architectures have been designed to cope with image classification problems on that dataset (**????????**).

2.7.1.3 SVHN

Used less frequently than the datasets mentioned above, but it is still a good baseline, the Street View House Numbers (SVHN) (**??**) is a set of images that comprises houses' numbers. Some characteristics come from the MNIST data set (**??**), like the 0-9 digits as the label; however, it has another order of magnitude of difficulty and number of instances.

According to the website [2], there are two versions of the dataset. The first one is a collection of images containing two or more digits in the same number, forming more complex instances and not being frequently used. All images are colored and vary in resolution and size.

Regularization works often uses the second version of the dataset. The same images from the first set are once more used; however, they are now segmented by each digit, so every label is among the 0-9 range. These images are scaled in 32×32 size, varying in resolution and color. There are three divisions of the dataset. The first is the original training set, formed by $73,257$ labeled instances. The second division contains $26,032$ images, and it is used for evaluation purposes. The third subset is called "extra" and includes $531,131$ designated figures. Realize that some works use the "extra" and "training" parts for training models, and others use the "training" portion to the size and time taken for training. Results reported in Table 2 are the ones that use both sets in training.

2.7.2 Architectures

For a fair comparison, two regularization methods must use the same architecture. In all works mentioned earlier, at least one of the architectures described in this subsection is used.

2.7.2.1 ResNet

The oldest architecture used in most of the regularization works, the ResNet family (**??**) is still one of the most commonly used CNNs. It stands for the first neural

[2] http://ufldl.stanford.edu/housenumbers

network to use residual connections, which is the concatenation of the output from previous layers with further transformations. The residual connection is powerful, functional, and easy to implement.

Several works use two variants of the ResNet. These variants are different not only because of the depth of the neural network, but the blocks have a different constitution. The ResNet-18, as the name suggests, is built from 18 layers with residual connections between every block, each block having two or three layers of a sequence of convolution and batch normalization (**??**), depending on the position of the block, with the third layer as a pooling layer by changing the stride of the convolution to 2 instead of 1. The other variant is the ResNet-50, which uses 50 layers; however, built-in more complex blocks, the so-called "bottleneck". Every block has three or four layers of convolution and batch normalization, again depending on the block's position. The fourth block works as a pooling layer, with the same rules as previously described.

2.7.2.2 Wide Residual Network - WRN

Another widespread architecture in regularization works is the Wide Residual Network (WRN) (**??**). It uses the same concept of residual connection between layers, but it has some structural differences from ResNet. The first one is the use of the concept of pre-activation (Pre-Act) layers. Both ResNet-18 and ResNet-50 use a sequence of convolution, batch normalization, and ReLU activation in their blocks. The Pre-Act block changes this sequence, i.e., it first employs batch normalization, then ReLu activation, and finally the convolution over the input. As shown in the original work (**??**), this sequence can outperform the traditional chain.

The second difference is the change in the widening and depth of the neural networks. It is widespread to observe the WRN being called "WRN-d-k", with k as the widening factor and d is the depth factor. The depth is the usual concept, i.e., it means the number of convolutional layers; however, the widening changes the structure significantly. When $k = 1$, it has the same structure as the ResNet; however, it means the layer has more convolutional kernels in a given layer when this number increases. This small change can generate a much shallower network (with 16 layers) with similar results as the ResNet-1001, containing $1,001$ layers. In the regularization works, the most common architecture is to employ the WRN-28-10, but it is possible to find some of them using the WRN-16-8 either.

The last distinction is the use of Dropout in the original architecture. The number of convolutional layers increases drastically on each layer, which may lead to overfitting (**??**). Dropout regularization between the convolutional layers after the ReLU activation helps perturb the batch normalization operation, which prevents overfitting.

2.7.2.3 ResNeXT

Intuitively, increasing the number of layers, blocks, or the number of kernels on some layers leads to the idea of better final results. For instance, some studies increase the number of blocks (**??**) or the number of convolution kernels in the layers (**??**) heavily. ResNeXt (**??**) introduces the concept of cardinality to accomplish better results.

Since ResNet (**??**), most of the neural networks are composed of the main branch, i.e., convolutional, activation, and batch normalization operations, followed by residual connections. In the ResNeXT architecture, the main string is divided by its cardinality value: for example, if a ResNet has a branch with 32 convolutions, followed by 64 and then another 32 convolutions, a ResNeXT block with cardinality 32 divides the main branch in 32 streams of 1, 2, and 1 convolution processes, and then concatenate all units before adding the residual value. It looks just a tiny difference in the general design; however, results in CIFAR and ImageNet datasets show that such a remodeling leads to better results. Comparing to previous architechtures (**??????**), it showed better outcomes.

2.7.2.4 PyramidNet

The last most common neural network is the one that achieves the best general results among the four mentioned here. The PyramidNet (**??**) shows some new procedures to improve outcomes from previous convolutional neural networks. The first difference is the size of each residual block. While most neural networks either keep the size of the output or downsample it and increase the feature map in the following layer, the PyradmidNet gradually increases the dimensionality in the subsequent layer. Such a procedure has been shown to improve the results in the classification task.

Such an increase in the feature map's size can also occur inside a residual block. It means that adding outcomes from the residual branch can be a problem concerning the dimensionality of the input tensor. To solve this, the authors proposed a Zero-Padded Shortcut Connection, which adds the values from a previous smaller tensor into a bigger one. According to He et al. (**??**), this operation might influence the gradient value because any change in this branch (even a scalar multiplication or a dropout regularization) might lead to wrong backpropagation values; however, the study shows that a zero-padded shortcut does not influence the values because no other operation is performed in the residual connection.

The last improvement is a new residual building block. This study shows that better results can be accomplished if the building block uses fewer ReLU activations. The first ReLU activation of the block does not influence that much in the nonlinearity of the system so that it can be removed.

2.8 Experimental Results

Convolutional Neural Networks are usually designed to achieve the best possible performance in image processing, depending on the targeting difficulty. Sometimes, the same basic structure can be used in two or more problems, i.e., one needs to change the output layer according to the labels. For instance, the EfficientNet structure (**??**) is re-used in the Efficient-Det work (**??**). Concerning regularization techniques, other components can be tricky to get rid of. Table 2 shows the results of several models on CIFAR-10, CIFAR-100, SVHN, and ImageNet datasets. The next sections overview an in-depth discussion about the experiments considered in this paper.

Method	Classification datasets.			
	CIFAR-10	CIFAR-100	SVHN	ImageNet
ResNet18 (??)	4.72 ± 0.21	22.46 ± 0.31	-	-
+ Cutout (??)	3.99 ± 0.13	21.96 ± 0.24	-	-
+ RandomErasing (??)	4.31 ± 0.07	24.03 ± 0.19	-	-
+ MaxDropout (??)	4.66 ± 0.14	21.93 ± 0.07	-	-
+ MaxDropout + Cutout (??)	3.76 ± 0.08	21.82 ± 0.13	-	-
+ Mixup (??)	4.2	21.1	-	-
+ MM (??)	2.95 ± 0.04	20.34 ± 0.52	-	-
+ TSLA (??)	-	21.45 ± 0.28	-	-
+ TargetDrop (??)	4.41	21.37	-	-
+ TargetDrop + Cutout (??)	3.67	21.25	-	-
+ LocalDrop (??)	4.3	22.2	-	20.2
WRN (??)	4.00	19.25	-	21.9
+ Dropout (??)	3.89	18.85	1.60 ± 0.05	-
+ MaxDropout (??)	3.84	18.81	-	-
+ TargetDrop (??)	3.68	-	-	-
+ GradAug (??)		16.02	-	-
+ Dropout + Cutout (??)	3.08 ± 0.16	18.41 ± 0.27	1.30 ± 0.03	-
+ Dropout + PBA (??)	2.58 ± 0.06	16.73 ± 0.15	1.18 ± 0.02	-
+ Dropout + RE (??)	3.08 ± 0.05	17.73 ± 0.15	-	-
+ Dropout + BA + Cutout (??)	2.85	19.87	-	-
+ ShakeDrop (??)	4.37	19.47	-	-

+ Dropout + RE (??)	3.08 ± 0.05	17.73 ± 0.15	-	-
+ Dropout + Mixup (??)	2.7	17.5	-	-
+ Dropout + MM (??)	2.55 ± 0.02	18.04 ± 0.17	-	-
+ Dropout + Fast AA (??)	2.7	17.3	1.1	-
+ Dropout + RA (??)	2.7	16.7	1.0	-
+ AutoDrop (??)	3.1		-	-
+ AutoDrop + RE (??)	2.1	-	-	-
ResNeXt (??)	3.58	17.31	-	19.1
+ RE (??)	3.24 ± 0.04	18.84 ± 0.18	-	-
+ FixRes (??)	-	-	-	13.6
+ ShakeDrop (??)	3.67	17.80	-	20.34
+ Shake-Shake (??)	3.08 ± 0.05	17.73 ± 0.15	-	-
+ Shake-Shake + Fast AA (??)	2.0	14.9	-	-
+ Shake-Shake + Cutout (??)	2.56 ± 0.07	15.20 ± 0.21	-	-
+ Shake-Shake + PBA (??)	2.03 ± 0.11	15.31 ± 0.28	1.13 ± 0.02	-
+ Shake-Shake + AA (??)	2.0 ± 0.1	14.30 ± 0.2	1.0	-
ResNet-50 (??)	-	-	-	23.49 ± 0.07
+ ShakeDrop (??)	-	25.26	-	-
+ Bag of Tricks (??)	-	-	-	21.67
+ GradAug (??)	-	-	-	20.33
+ LocalDrop (??)	5.3	26.2	-	21.1
+ Dropout (??)	-	-	-	23.20 ± 0.04
+ Cutout (??)	-	-	-	23.48 ± 0.07

+ AA (**??**)	-	-	-	22.4
+ BA (**??**)	-	-	-	23.14
+ Fast AA (**??**)	-	-	-	22.37
+ Mixup (**??**)	-	-	-	22.1
+ DropBlock (**??**)	-	-	-	21.65 ± 0.05
+ FixRes (**??**)	-	-	-	17.5
+ RA (**??**)	-	-	-	22.4
+ AutoDrop (**??**)	-	-	-	21.3
+ AutoDrop + RA (**??**)	-	-	-	19.7
PyramidNet (**??**)	3.48 ± 0.20	17.01 ± 0.39	1.0	- 19.2
+ GradAug (**??**)		13.76	-	20.94
+ ShaekDrop (**??**)	3.08	14.96	-	20.94
+ ShaekDrop + Cutout (**??**)	2.3	12.2	-	-
+ ShaekDrop + AA (**??**)	1.5 ± 0.1	10.7 ± 0.2	-	-
+ ShaekDrop + Fast AA (**??**)	1.8	11.9		20.94
+ ShaekDrop + RA (**??**)	1.5	-	-	15.0
+ ShaekDrop + PBA (**??**)	1.46 ± 0.07	10.94 ± 0.09	-	15.0

Table 2: Error (in %) for each classification dataset using different methods and models. The following acronyms were used: MM = ManifoldMixup, PBA = Population Based Augmentation, RE = RandomErasing, BA = BatchAugmentation, AA = AutoAugment, Fast AA = Fast AutoAugment, and RA = RandAugment.

2.8.1 State-of-the-art Regularizer?

Defining the best regularization technique is not something trivial. For example, the baseline for determining the best image classifier is the one that achieves the best results in the 2012 ILRSVC image classification challenge dataset. During this work, the current research with the best impact on the mentioned dataset is the Meta Pseudo Labels approach (**??**). One may argue that the best result achieved by a regularization technique on a given architecture might be considered the best regularization method.

According to Table 2, which is a compilation of the results in the most common architectures, one can observe that AutoAugment performs better than PBA using ResNeXT architecture plus Shake-shake regularization in the CIFAR-10 dataset. However, when both regularization algorithms are compared using PyramidNet and Shake-Drop regularization, the opposite happens: PBA achieves better results on CIFAR-10 than AutoAugment. Further analysis showed it is possible to observe other variations in the results.

The best possible assumption about state-of-the-art regularization is based on the results and sorting them into work areas. For example, the likely best regularizer concerning the input layer is the RandAugment, for it does not affect the time spent for training and achieves satisfactory results. For internal regularizers, it is even more challenging. Take ShakeDrop as an example. It has not been evaluated within ResNet-18, while MaxDropout was not assessed for the PyramidNet. Based only on a guess, ShakeDrop appears to have the best results in this particular part. Unfortunately, there are only two regularizers that work directly on labels. For this reason, the TSLA might be considered the best one to be used on a label level.

2.8.2 Defining a basic protocol

There are several aspects to be considered for a fair evaluation of a new regularizer. The primary purpose of using regularization is to improve a given baseline architecture by using some operations in the input data, among layers, or in the label. However, a slight difference in the training protocol may infer a better result, not necessarily related to the operations from the regularizer. Another protocol can be removing any other regularization method, even small data augmentation and weight decay. As such, it is possible to verify how a new regularizer can improve a baseline architecture without any other influence.

Some papers (**??????**) train ResNet-18 using the same data transformations, i.e., random flipping, padding pixels, and using the same values for the weight decay. Some works use the same neural network, claim to have some relevant results but do not make the source code available, turning the evaluation process not trustful since there might be other transformations working on training, such as Dropout (**??**). Therefore, the primary condition to be cited in this survey is to have the source code available so that it can be compared to other methods directly.

On the other hand, it might be crucial for different reasons to use more than one regularizer in the same evaluation. For instance, the Wide Residual Network (**??**), a common architecture used for evaluating new regularizers, has in its layers a dropout regularization. Therefore, wherever a new regularization is proposed (in the input, among layers, or in the label), it should be able to work with the dropout regularization. Another point is that some regularizers incorporate other techniques naturally. For instance, the AutoAugment (**??**) and the Fast AutoAugmentat (**??**) incorporate as one of their policies the Cutout (**??**). Therefore, a new regularization technique should be able to work with another regularizer and improve the results when both are used together.

2.8.3 Use of minor architectures

As a general rule, regularization adds little overhead during training time (AutoAugmentation (**??**) is, perhaps, the only one that increases training time to find out the better policies for data augmentation) and no overhead at all at inference. For this reason, the use of regularizers for avoiding early overfitting of a neural network is strongly recommended. Still, it should be encouraged, sometimes, to use more than one at the same time. No matter what the problem is, everyone has the desire to improve results; however, it is particularly necessary for shallower neural networks.

One point missing in all works analyzed in this survey is the lack of proper investigation concerning lightweight CNNs. Architectures like MobileNet-V3 (**??**) should be boosted in regularization works for these smaller designs usually have fewer parameters or make use of less complex calculations. In the same direction, quantization (**??**) should be dissected to know how a given regularization algorithm influences either training a quantized neural network and performing the quantization after training.

EfficientNet (**??**) provides a clever calculation for defining how an efficient CNN architecture should be designed, based on the width, depth, and resolution. However, for faster and less resourceful hardware, this calculation presents better results when neural networks's resolution and depth are designated as more important than width. It is possible to verify that in the TinyNet work (**??**). It might be a good idea to provide comparisons using this minimal and fast neural network architecture to show that new regularizers can improve results for smaller CNNs.

2.8.4 Use of more complex datasets

The most common datasets used in regularization works concern objects and animals, which humans can easily distinguish. Another characteristic of these datasets is that they are perfectly balanced, meaning that every possible class has a similar amount of samples in the training and test validation. Usually, in medical and some real-world problems, such a balancing is hard to obtain.

In health-related problems, any increase in the results can lead to a safer treatment or even avoid misuse of medication and death. For these reasons, some datasets, like the Breast Cancer Histopathological Image Classification (BreakHis) (**??**), might be used to increase the work's relevance. In this specific case, where the results may infer in a life-threatening situation, the idea is to use a deeper CNN, like the Efficient-Net family (**??**) or ResNet (**??**).

2.8.5 Other problems besides classification

In the past, CNNs and other neural networks were mainly used for the image classification task. However, more recently, CNNs were also employed in other tasks, such as object detection and speech recognition. For example, the YOLO architecture (**??**) is a Fully Convolutional Network, which means that every layer performs a 2D convolutional process. In that sense, some changes on the loss calculation allow final layers to find out where objects on a given image are located. Another domain where CNNs have state-of-the-art results is image reconstruction. The Residual Dense Network has outstanding results on image reconstruction from noisy (**??**) and low-resolution images (**??**).

There are two suggestions in this case. The first one is the use of regularization techniques in such different tasks or, at least, a reason for not using them in other domains. The second proposal is the development of new regularization targeting these specific problems. The only work found so far to solve different problems than image classification is the CutBlur (**??**), thus highlighting the lack of works in this direction.

2.8.6 Source Code Links

As mentioned before, we only considered papers with the source code available. Table 3 presents the list of links concerning the source codes for every paper surveyed in this work.

Reference	Short Name	Source Code
(??)	Bag of Tricks	MXNet: https://github.com/dmlc/gluon-cv
(??)	Batch Augment	PyTorch: https://github.com/eladhoffer/convNet.pytorch
(??)	FixRes	PyTorch: https://github.com/facebookresearch/FixRes
(??)	Cutout	Pytorch: https://github.com/uoguelph-mlrg/Cutout
(??)	CutMix	PyTorch: https://github.com/clovaai/CutMix-PyTorch
(??)	RandomErasing	PyTorch: https://github.com/zhunzhong07/Random-Erasing
(??)	Mixup	PyTorch: https://github.com/facebookresearch/mixup-cifar10
(??)	AutoAugment	TensorFlow: https://github.com/tensorflow/tpu/blob/master /models/officialefficientnet/autoaugment.py
(??)	Fast AutoAugment	PyTorch: https://github.com/kakaobrain/fast-autoaugment
(??)	RandAugment	TensorFlow: https://github.com/tensorflow/tpu/tree /master/models/official/efficientnet
(??)	PBA	TensorFlow: https://github.com/arcelien/pba
(??)	CutBlur	PyTorch: https://github.com/clovaai/cutblur
(??)	Dropout	PyTorch: https://pytorch.org/docs/ stable/_modules/torch/nn/modules/dropout.html
(??)	MaxDropout	Pytorch: https://github.com/cfsantos/MaxDropout-torch/
(??)	GradAug	PyTorch: https://github.com/taoyang1122/GradAug
(??)	Local Drop	Available in the paper
(??)	Shake-Shake	Torch: https://github.com/xgastaldi/shake-shake
(??)	ShakeDrop	Torch: https://github.com/imenurok/ShakeDrop
(??)	Manifold Mixup	PyTorch: https://github.com/ vikasverma1077/manifold_mixup

(??)	DropBlock	TensorFlow: https://github.com/tensorflow/tpu/tree/master/models/official/resnet
(??)	AutoDrop	TensorFlow: https://github.com/google-research/google-research/tree/master/auto_dropout
(??)	Label Smoothing	TensorFlow: https://github.com/tensorflow/models
(??)	TSLA	Available in the paper
(??)	SLS	Available in the paper
(??)	JoCoR	https://github.com/hongxin001/JoCoR

Table 3: Summarization of the approaches considered in the survey and their respective source codes.

2.9 Conclusion

Regularization is a vital tool to improve the final CNN results since it helps prevent the model from overfitting on the training data. This work aimed at showing the most recent commitments in the area, targeting to deliver a brief resume on how they work and their main results.

This work introduced a lineup of recent regularizers that can fit in most neural networks for outcome improvement. Although some can drastically increase the training time, such as AutoAugment, most do not require any relevant extra time, and none influences the time taken for inference. Right after the introduction, we provide a brief explanation of how CNN works and a little history of its development, and then we divided all works analyzed in this paper as follows:

- "input regularization", where the models work before the image is fed to the network;
- "internal regularization", when the regularization algorithms work after the image is feedforwarded to the model; and
- "label regularization", when the algorithm performs on the output layer.

Besides, the methodology presents the most popular datasets used to evaluate regularization techniques and the most traditional CNN architectures for such a task. Such information is crucial, for it helps standardize an evaluation protocol from now on.

Along with the reported results for each work, we provided our opinion on setting up a state-of-the-art regularizer, an essential but trustful protocol evaluation for new regularizers, which can help compare the results and provide insights for researchers in this area. The same section highlights some issues we found in most of the works:

- the lack of using simpler architectures, which are the ones that could be more benefited from the use of regularizers; and
- the lack of an evaluation of methods on more complex data, such as unbalanced data sets, to provide richer information for other researchers.

Last but not least, we encourage the development of new regularization techniques on tasks other than image classification, such as object detection and image reconstruction.

Chapter 3

MaxDropout: Deep Neural Network Regularization Based on Maximum Output Values

This chapter presents the content publish in the conference 25th Internation Conference on Pattern Recognition (ICPR 2020) (**??**), and it proposes a new regularization method, changing the neuron randon drop rules of the classical Dropout (**??**) by dropping the most active neurons on a given tensor.

3.1 Abstract

Different techniques have emerged in the deep learning scenario, such as Convolutional Neural Networks, Deep Belief Networks, and Long Short-Term Memory Networks, to cite a few. In lockstep, regularization methods, which aim to prevent overfitting by penalizing the weight connections, or turning off some units, have been widely studied either. In this paper, we present a novel approach called MaxDropout, a regularizer for deep neural network models that works in a supervised fashion by removing (shutting off) the prominent neurons (i.e., most active) in each hidden layer. The model forces fewer activated units to learn more representative information, thus providing sparsity. Regarding the experiments, we show that it is possible to improve existing neural networks and provide better results in neural networks when Dropout is replaced by MaxDropout. The proposed method was evaluated in image classification, achieving comparable results to existing regularizers, such as Cutout and Ran-

domErasing, also improving the accuracy of neural networks that use Dropout by replacing the existing layer by MaxDropout.

3.2 Introduction

Following the advent of deeply connected systems and the new era of information, tons of data are generated every moment by different devices, such as smartphones or notebooks. A significant portion of the data can be collected from images or videos, which are usually encoded in a high-dimensional domain. Deep Learning (DL) techniques have been broadly employed in different knowledge fields, mainly due to their ability to create authentic representations of the real world, even for multimodal information. Recently, DL has emerged as a prominent area in Machine Learning, since its techniques have achieved outstanding results and established several hallmarks in a wide range of applications, such as motion tracking (**??**), action recognition (**??**), and human pose estimation (**????**), to cite a few.

Deep learning architectures such as Convolutional Neural Networks (CNNs), Deep Autoencoders, and Long Short-Term Memory Networks are powerful tools that deal with different image variations such as rotation or noise. However, their performance is highly data-dependent, which can cause some problems during training and further generalization for unseen examples. One common problem is overfitting, where the technique memorizes the data either due to the lack of information or because of too complex neural network architectures.

Such a problem is commonly handled with regularization methods, which represent a wide area of study in the scientific community. The employment of one or more of such techniques provides useful improvements in different applications. Among them, two well-known methods can be referred: (i) so-called "Batch Normalization" and (ii) "Dropout". The former was introduced by Ioffe et al. (**??**) and performs data normalization in the output of each layer. The latter was introduced by Srivastava et al. (**??**), and randomly deactivates some neurons present in each layer, thus forcing the model to be sparse.

However, dropping neurons out at random may slow down convergence during learning. To cope with this issue, we introduced an improved approach for regularizing deeper neural networks, hereinafter called "MaxDropout" [1], which shuts off neurons based on their maximum activation values, i.e., the method drops the most active neurons to encourage the network to learn better and more informative features. Such an approach achieved remarkable results for the image classification task, concerning two important well-established datasets.

[1] https://github.com/cfsantos/MaxDropout-torch

The remainder of this paper is presented as follows: Section 5.3 introduces the correlated works, while Section 5.4 presents the proposed approach. Further, Section 4.6 describes the methodology and datasets employed in this work. Finally, Sections 5.6 and 5.8 provide the experimental results and conclusions, respectively.

3.3 Related Works

Regularization methods are widely used by several deep neural networks (DNNs) and with different architectures. The main idea is to help the system to prevent the overfitting problem, which causes the data memorization instead of generalization, also allowing DNNs to achieve better results. A well-known regularization method is Batch Normalization (BN), which works by normalizing the output of a giving layer at each iteration. The original work (**??**) showed that such a process speeds up convergence for image classification tasks. Since then, several other works (**????**), including the current state-of-the-art on image classification (**??**), also highlighted its importance.

As previously mentioned, Dropout is one of the most employed regularization methods for DNNs. Such an approach was developed between 2012 and 2014 (**??**), showing significant improvements in neural network's performance for various tasks, ranging from image classification, speech recognition, and sentimental analysis. The standard Dropout works by creating, during training time, a mask that direct multiples all values of a given tensor. The values of such a mask follow the Bernoulli distribution, being 0 with a probability p and 1 with a probability $1 - p$ (according to the original work (**??**), the best value for p in hidden layers is 0.5). During training, some values will be kept while others will be changed to 0. Visually, it means that some neurons will be deactivated while others will work normally.

After the initial development of the standard Dropout, Wang and Manning (**??**) explored different strategies for sampling since at each mini-batch a subset of input features is turned off. Such a fact highlights an interesting Dropout feature since it represents an approximation by a Markov chain executed several times during training. Since the Bernoulli distribution tends to a Normal distribution when the dimensional space is high enough, such an approximation allows Dropout to its best without sampling.

In 2015, Kingma et al. (**??**) proposed the Variational Dropout, a generalization of Gaussian Dropout in which the dropout rates are learned instead of randomly attributed. They investigated a local reparameterization approach to reduce the variance of stochastic gradients in variational Bayesian inference of a posterior over the model parameters, thus retaining parallelizability. On the other hand, in 2017, Gal et al. (**??**) proposed a new Dropout variant to reinforcement learning models. Such a method aims to improve the performance and better calibration of uncertainties once it is an intrinsic property of the Dropout. In such a field, the proposed approach allows the agent to adapt its uncertainty dynamically as more data is seen.

Later on, Molchanov et al. (**??**) explored the Variational Dropout proposed by Kingma et al. (**??**). The authors extended it to situations when dropout rates are unbounded, leading to very sparse solutions in fully-connected and convolutional layers. Moreover, they achieved a reduction in the number of parameters up to 280 times on LeNet architectures, and up to 68 times on VGG-like networks with a small decrease in accuracy. Such a fact points out the importance of sparsity for parameter reduction and performance overall improvement.

Paralleling, other regularization methods have been emerged, like the ones that change the input of the neural network. For instance, Cutout (**??**) works by literally cutting off a region of the image (by setting the values of a random region to 0). This simple approach shows relevant results on several datasets. Another similar regularizer is the RandomErasing (**??**), that works in the same manner, but instead of setting the values of the region to 0, it changes these pixels to random values.

By bringing the concepts mentioned above and works close to the proposed approach, one can point out that the MaxDropout is similar to the standard Dropout, however, instead of randomly dropping out neurons, our approach follows a policy for shutting off the most active cells, representing a selection of neurons that may overfit the data, or discourage the fewer actives from extracting useful information.

3.4 Proposed Approach

The proposed approach aims at shutting out the most activated neurons, which is responsible for inducing sparsity in the model, at the step that encourage the hidden neurons to learn more informative features and extract useful information that positively impacts the network's generalization ability.

For the sake of visualization, Figures 11a-c show the differences between the proposed approach and the standard Dropout, in which Figure 11a stands for the original grayscale image and Figures 11b and 11c denote their corresponding outcomes after Dropout and MaxDropout. It is important to highlight that Dropout removes any pixel of the image randomly, while MaxDropout tends to inactivate the lighter pixels.

The rationale behind the proposed approach can be better visualized in a tensor-like data. Considering the colored image showed in Figure 11d, one can observe its outcome after Dropout and MaxDropout transformations in Figures 11e and 11f, respectively. Regarding standard Dropout, the image looks like a colored uniform noise, while MaxDropout could remove entire regions composed of bright pixels (i.e., pixels with high activation values, as expected).

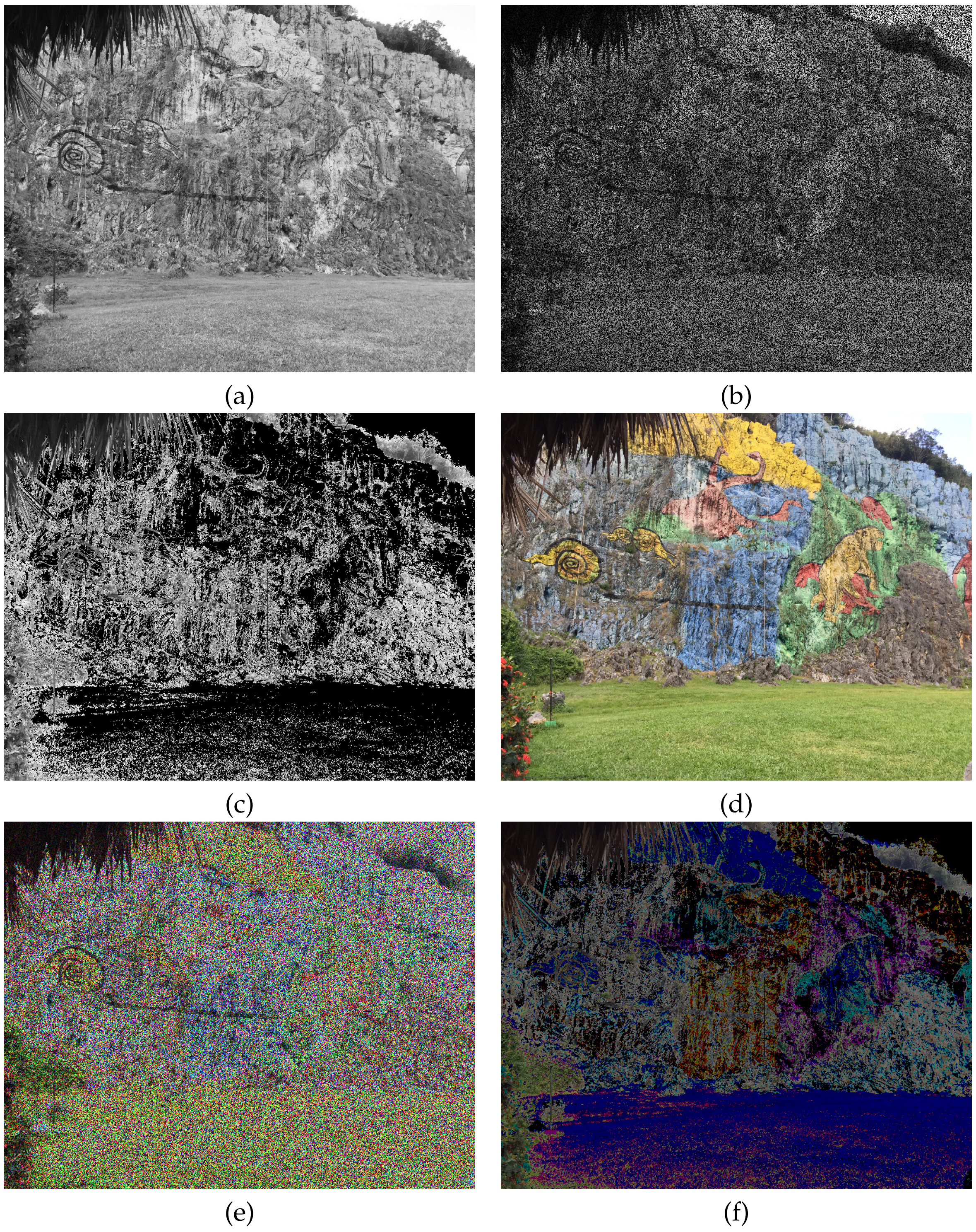


Figure 7: Simulation using grayscale (a)-(c) and colored images (d)-(f): (a) original grayscale image and its outcomes after (b) Dropout and (c) MaxDropout transformations, respectively, and (d) original colored image and its outcomes after (e) Dropout and (f) MaxDropout transformations, respectively. In all cases, the drop rout ate is 50%.

For the sake of clarification purposes, Algorithm 4.1 implements the proposed MaxDropout[2]: the main loop in Lines $1-9$ is in charge of the training procedure, and the

[2] The pseudocode uses Keras syntax.

inner loop in Lines $2-8$ is executed for each hidden layer. Line 3 computes a random value uniformly distributed that is going to work as the dropout rate r. The output of each layer produces an $x \times y \times z$ tensor, where x and y stand for the image's size, and z denotes the number of feature maps produced for each convolutional kernel. Line 4 creates a copy of the original tensor and uses an L_2 normalization to produce an output between 0 and 1.

Listing 3.1: Original MaxDropout code

```
1 while training do
2     for each layer do
3       rate = U(o, r)
4       norm_tensor = L2Normalize(tensor)
5       max = Max(tensor)
6       keptIdx = IdxOf(norm_tensor, (1 − rate) * max)
7       returnTensor = tensor * keptIdx
8     end for
9 end while
```

Later, Line 5 finds the biggest value in the normalized tensor, once it may not be equal to one[3]. Line 6 creates another tensor with the same shape as the input one and assigns 1 where $(1 - rate) \times max$ at a certain tensor position is greater than a given threshold; otherwise it sets such a position to 0. Finally, Line 7 creates the tensor to be used in the training phase, where each position of the original tensor is multiplied by the value in the respective position of the tensor created in the line before. Therefore, such a procedure guarantees that only values smaller than the threshold employed in Line 3 go further on.

3.5 Experiments

In this section, we describe the methodology employed to validate the robustness of the proposed approach. The hardware used for the paper is an Intel Xeon Bronze® 3104 CPU with 6 cores (12 threads), 1.70GHz, 96GB RAM with 2666Mhz, and a GPU Nvidia Tesla P4 with 8GB. Since most of the regularization methods aim to improve image classification tasks, we decided to follow the same protocol and approaches for a fair comparison.

[3] Depending on the floating-point precision, the maximum value can be extremely close but not equal to one.

3.5.1 Neural Network Structure

Regarding the neural network structure, we evaluated the proposed approach in two different practices. For the former experiments, regularization layers were added to a neural network that does not drop any transformation between layers. Concerning the latter experiments, the standard Dropout (**??**) layers were changed by the MaxDropout one to compare results.

For the first experiment, ResNet18 (**??**) was chosen because such an architecture has been used in several works for comparison purposes when coming to new regularizer techniques. ResNet18 is compounded by a sequence of convolutional residual blocks, followed by the well-known BatchNormalization (**??**). As such, a MaxDropout layer was added between these blocks, changing the basic structure during training but keeping it to inference purposes.

In the second experiment, a slightly different approach has been performed. Here, a neural network that already has the Dropout regularization in its composition was considered for direct comparison among methods. The WideResNet (**??**) uses Dropout layers in its blocks with outstanding results on image classification tasks, thus becoming a good choice.

3.5.2 Training Protocol

In this work, we considered a direct comparison with other regularization algorithms. To be consistent with the literature, we provided the error rate instead of the accuracy itself (**??????**). Nonetheless, to ensure that the only difference between the proposed approach and the baselines used for comparison purposes concerns the MaxDropout layer, we strictly followed the protocols according to the original works.

To compare MaxDropout with other regularizers, we followed the protocol proposed by DeVries and Taylor (**??**), in which five runs were repeated, and the mean and the standard deviation are used for comparison purposes. For the experiment, the images from the datasets were normalized per-channel using mean and standard deviation.

During the training procedure, the images were shifted four pixels in every direction and then cropped into 32x32 pixels. Besides, the images were horizontally mirrored with a 50% probability. In such a case, two comparisons were provided. In the first case, besides the data augmentation already described, only the MaxDropout was included in the ResNet18 structure, directly comparing to the other methods. Regarding the second case, the Cutout data augmentation was included, providing a direct comparison of the results, showing that the proposed approach can work nicely.

As previously mentioned, to evaluate the MaxDropout against the standard Dropout, we chose the Wide Residual Network (**??**), and the same training protocol and parameters were employed to make sure the only difference concerns the type of neuron dropping.

3.5.3 Datasets

In this work, two well-established datasets in the literature were employed, i.e., CIFAR-10 (**??**) and its enhanced version CIFAR-100 (**??**). Using such datasets allows us to compare the proposed approach toward important baseline methods, such as the standard Dropout (**??**) and CutOut (**??**). Figure 8 portrays random samples extracted from the datasets mentioned above.

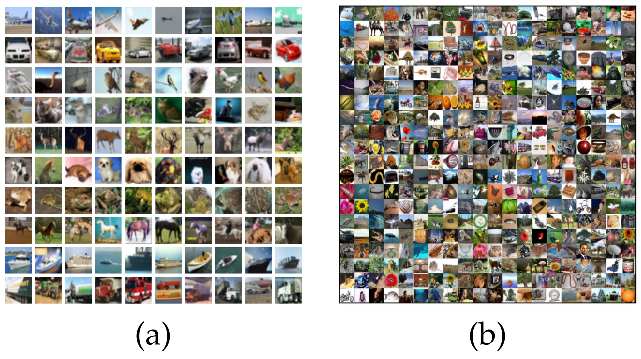


Figure 8: Random training samples from: (a) CIFAR-10 and (b) CIFAR-100 datasets.

CIFAR-10 dataset comprises 10 classes equally distributed in $60,000$ colored image samples, with a dimension of 32x32 pixels. The entire dataset is partitioned into $50,000$ training images and $10,000$ test images. On the other hand, CIFAR-100 dataset holds similar aspects of its smaller version, but now with 100 classes equally distributed in $60,000$ colored image samples, with 600 images samples per class. Nonetheless, the higher number of classes and the low number of samples per class make image classification significantly hard in this case.

3.6 Experimental Results

This section is divided into four main parts. First, we provided a convergence study during training for all experiments. Later, we compared the results of MaxDropout with other methods showing that, when combined with other regularizers, MaxDropout can lead to even better performance than their original versions. Finally, in the last part, we make a direct comparison between the proposed approach and standard Dropout by replacing the equivalent layer with the MaxDropout in the WideResNet.

3.6.1 Training Evolution

Figures 9 and 10 depict the mean accuracies concerning the test set considering the 5 runs during training phase. Since we are dealing with regularizers, it makes sense to analyze their behavior during training and, for each epoch, compute their accuracy over the test set. One can notice that the proposed approach can improve the results even when the model is near to overfit.

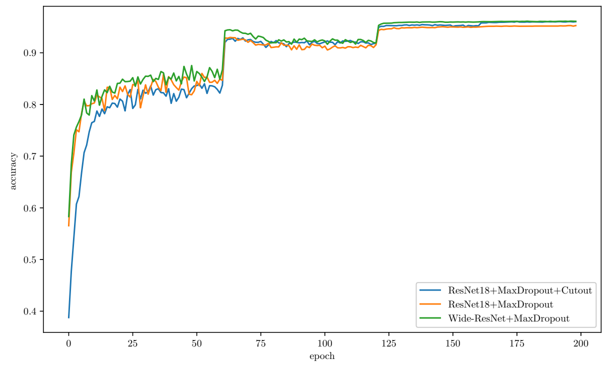


Figure 9: Convergence over CIFAR-10 test set.

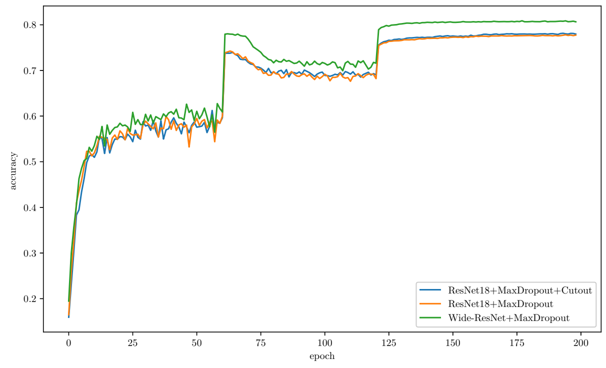


Figure 10: Convergence over CIFAR-100 test set.

3.6.2 Comparison Against Other Regularizers

As aforementioned, we considered a comparison against some baselines over five runs and exposed their mean accuracies and standard deviation in Table 4. Such results evidence the robustness of the proposed approach against two other well-known regularizers, i.e., Cutout, and the RandomErasing.

From Table 4, one can notice that when MaxDropout is incorporated within ResNet18 blocks, it allows the model to accomplish relevant and better results. Regarding the CIFAR-10 dataset, the model that uses MaxDropout achieved an absolute reduction of around 0.54% in the error rate when compared to ResNet18 (and approximately 12% on the relative error). However, concerning the CIFAR-100 dataset, the model achieved

Approach	CIFAR-100	CIFAR-10
ResNet18 (**????**)	24.50 ± 0.19	5.17 ± 0.18
ResNet18+RandomErasing (**??**)	24.03 ± 0.19	4.31 ± 0.07
ResNet18+Cutout (**??**)	21.96 ± 0.24	$\mathbf{3.99 \pm 0.13}$
ResNet18+MaxDropout	$\mathbf{21.94 \pm 0.23}$	4.63 ± 0.11

Table 4: Results of MaxDropout and other regularizers

over 2.5% less error than the same baseline (and approximately 12% on the relative error), besides being statistically similar to Cutout.

3.6.3 Working Along with Other Regularizers

Since MaxDropout works inside the neural network by changing the hidden layers' values, it permits the concomitant functionality with other methods that change information from the input, such as Cutout. Table 5 portrays the results of each stand-alone approach and their combination. From these results, one can notice a slight improvement in performance considering the CIFAR-100 dataset, but it ends up as a relevant gain on CIFAR-10 dataset, reaching the best results so far.

Regularizer	CIFAR-100	CIFAR-10
Cutout (**??**)	21.96 ± 0.24	3.99 ± 0.13
MaxDropout	21.94 ± 0.23	4.63 ± 0.11
MaxDropout + Cutout	$\mathbf{21.82 \pm 0.13}$	$\mathbf{3.76 \pm 0.08}$

Table 5: Results of the MaxDropout combined with Cutout.

3.6.4 MaxDropout x Dropout

One interesting point such a work stands for concerns the following question: Is the MaxDropout comparable to the standard Dropout (**??**)? To answer this question, we compared the proposed approach against standard Dropout by replacing it with MaxDropout on the Wide Residual Network (WRN).

From Table 6, one can observe the model using MaxDropout works slightly better than standard Dropout, leading to dropping in the error rate regarding CIFAR-100 and CIFAR-10 datasets by 0.04 and 0.05%, respectively. Although it may not look an impressive improvement, we showed that the proposed approach has a margin to improve the overall results, mainly when the threshold of the MaDropout is taken into account (i.e., ablation studies)[4].

[4] We did not show the standard deviation since the original study did not present such an information as well.

Model	CIFAR-100	CIFAR-10
WRN (**??**)	19.25	4.00
WRN + Dropout (**??**)	18.85	3.89
WRN + MaxDropout	**18.81**	**3.84**

Table 6: Results of Dropout and MaxDropout over the WRN.

3.6.5 Ablation Study

As aforementioned, we conducted several experiments with ResNet18 to determine the optimal rate (r) employed on previous subsections for the MaxDropout. Such results are presented in Table 7.

MaxDropout Rate (r)	CIFAR-100	CIFAR-10
05	22.05 ± 0.17	4.76 ± 0.09
10	22.06 ± 0.32	4.71 ± 0.09
15	22.16 ± 0.20	$\mathbf{4.63 \pm 0.11}$
20	21.99 ± 0.21	4.70 ± 0.08
25	$\mathbf{21.94 \pm 0.23}$	4.70 ± 0.06
30	22.08 ± 0.24	4.67 ± 0.12
35	22.10 ± 0.29	4.71 ± 0.16
40	22.17 ± 0.34	4.79 ± 0.20
45	22.31 ± 0.29	4.71 ± 0.11
50	22.33 ± 0.23	4.75 ± 0.10

Table 7: Ablation results concerning MaxDropout over ResNet18.

One can notice that r values vary with the dataset employed and the number of classes[5], i.e., for CIFAR-100 the lower error rate was achieved with r = 25%, while for CIFAR-10 the r value was 15%. Such variance might be related to the number of available images for each class. Therefore, the best r values were employed for each dataset on the main experiments previously addressed.

3.7 Discussion

Unfortunately, the approaches employed for comparison purposes did not release their training evolution for a direct comparison in Section 5.5.3. Nevertheless, it is possible to observe that all models performed very well for the image classification task. In Table 4, MaxDropout shows a result as good as Cutout for CIFAR-100 dataset, demonstrating it performs as expected when improving baseline models' results. However, it

[5] CIFAR-10 and CIFAR-100 uses mainly the same images but, with different classes

did not perform as well for CIFAR-10 dataset, but it still improves the baseline model results by almost 0.5%.

Results from Table 5 show that the MaxDropout supports the improvement when another regularizer is used along with. Although Cutout has been used to demonstrate the proposed approach's effectiveness, one can consider other similar regularizers. The most interesting results can be found in Table 6, where MaxDropout is directly compared to the standard Dropout. It shows relevant gains over the baseline model, and it performs a little better than Dropout using the same drop rate, indicating that it may be the case to find out the best drop rates for MaxDropout, which can be data or model-dependent.

3.8 Conclusions and Future Works

In this paper, we introduced MaxDropout, an improved version of the original Dropout method. Experiments show that it can be incorporated into existing models, working along with other regularizers, such as Cutout, and can replace the standard Dropout with some accuracy improvement.

With relevant results, we intend to conduct a more in-depth investigation to figure out the best drop rates depending on the model and the training data. Moreover, the next step is to re-implement MaxDropout and make it available in other frameworks, like TensorFlow and MXNet, and test in other tasks, such as object detection and image segmentation.

Nonetheless, we showed that MaxDropout works very well for image classification tasks. For future works, we intended to perform evaluations in other different tasks such as natural language processing and automatic speech recognition.

Chapter 4

MaxDropoutV2: An Improved Method to Drop out Neurons in Convolutional Neural Networks

This chapter presents the content publish in the conference 10th Iberian Conference on Pattern Recognition and Image Analysis(IbPRIA 2022) (**??**), and it proposes an adaptation of the MaxDropout method to Convolutional Neuron Networks, achieving similar results but being faster for training.

4.1 Abstract

In the last decade, exponential data growth supplied the machine learning-based algorithms' capacity and enabled their usage in daily life activities. Additionally, such an improvement is partially explained due to the advent of deep learning techniques, i.e., stacks of simple architectures that end up in more complex models. Although both factors produce outstanding results, they also pose drawbacks regarding the learning process since training complex models denotes an expensive task and results are prone to overfit the training data. A supervised regularization technique called MaxDropout was recently proposed to tackle the latter, providing several improvements concerning traditional regularization approaches. In this paper, we present its improved version called MaxDropoutV2. Results considering two public datasets show that the model performs faster than the standard version and, in most cases, provides more accurate results.

4.2 Introduction

The last decades witnessed a true revolution in people's daily life habits. Computer-based approaches assume the central role in this process, exerting fundamental influence in basic human tasks, such as communication and interaction, entertainment, working, studying, driving, and so on. Among such approaches, machine learning techniques, especially a subfield usually called deep learning, occupy one of the top positions of importance in this context since they empower computers with the ability to act reasonably in an autonomous fashion.

Deep learning regards a family of machine learning approaches that stacks an assortment of simpler models. The bottommost model's output feeds the next layer, and so on consecutively, with a set of possible intermediate operations among layers. The paradigm experienced exponential growth and magnificent popularity in the last years due to remarkable results in virtually any field of application, ranging from medicine (**??????**) and biology (**??**) to speech recognition (**??**) and computer vision (**??**).

Despite the success mentioned above, deep learning approaches still suffer from a drawback very commonly observed in real-world applications, i.e., the lack of sufficient data for training the model. Such a constraint affects the learning procedure in two main aspects: (i) poor classification rates or (ii) overfitting to training data. The former is usually addressed by changing to a more robust model, which generally leads to the second problem, i.e., overfitting. Regarding the latter, many works tackled the problem using regularization approaches, such as the well-known batch normalization (**??**), which normalizes the data traveling from one layer to the other, and dropout (**??**), which randomly turns-off some neurons and forces the layer to generate sparse outputs.

Even though dropout presents itself as an elegant solution to solve overfitting issues, Santos et al. (**??**) claim that deactivating neurons at random may impact negatively in the learning process, slowing down the convergence. To alleviate this impact, the authors proposed the so-called MaxDropout, an alternative that considers deactivating only the most active neurons, forcing less active neurons to prosecute more intensively in the learning procedure and produce more informative features.

MaxDropout obtained significant results considering image classification's task, however, at the cost of considerable computational cost. This paper addresses such an issue by proposing MaxDropoutV2, an improved and optimized version of MaxDropout capable of obtaining similar results with higher performance and substantial reduction of the computational burden.

Therefore, the main contributions of this work are presented as follows:

❑ to propose a novel regularization approach called MaxDropoutV2, which stands for an improved and optimized version of MaxDropout;

- ❑ to evaluate MaxDropoutV2 overall accuracy and training time performance, comparing with the original MaxDropout and other regularization approaches; and
- ❑ to foster the literature regarding regularization algorithms and deep learning in general.

The remainder of this paper is organized as follows. Section 4.3 introduces the main works regarding Dropout and its variation, while Section 4.4 presents the proposed approach. Further, Sections 4.5 and 4.5.2 describe the methodology adopted in this work and the experimental results, respectively. Finally, Section 4.7 states conclusions and future work.

4.3 Related Works

The employment of regularization methods for training deep neural networks (DNNs) architectures is a well-known practice, and its use is almost always considered by default. The focus of such approaches is helping DNNs to avoid or prevent overfitting problems, which reduce their generalization capability. Besides, regularization methods also allow DNNs to achieve better results considering the testing phase since the model becomes more robust to unseen data.

Batch Normalization (BN) is a well-known regularization method that employs the concept of normalizing the output of a given layer at every iteration in the training process. In its seminal work, Ioffe and Szegedy (**??**) demonstrated that the technique is capable of speeding up the convergence regarding the task of classification. Further, several other works (**????**) highlighted its importance, including the current state-of-the-art on image classification (**??**).

Among the most commonly employed techniques for DNN regularization is the Dropout, which is usually applied to train such networks in most of the frameworks used for the task. Developed by Srivastava et al. (**??**), Dropout shows significant improvements in a wide variety of applications of neural networks, like image classification, speech recognition, and more. The standard approach has a simple and efficient work procedure, in which a mask that directly multiplies the weight connections is created at training time for each batch. Such a mask follows a Bernoulli distribution, i.e., it assign values 0 with a probability p and 1 with a probability $1 - p$. The authors showed that the best value for p in hidden layers is 0.5. During training, the random mask varies, which means that some neurons will be deactivated while others will work normally.

Following the initial development of the Dropout method, Wang and Manning (**??**) focused on exploring different sampling strategies, considering that each batch corresponds to a new subnetwork taken into account since different units are dropped out.

In this manner, the authors highlighted that the Dropout represents an approximation of a Markov chain executed several times during training time. Also, the Bernoulli distribution tends to a Normal distribution in a high dimensional space, such that Dropout performs best without sampling.

Similarly, Kingma et al. (**??**) proposed the Variational Dropout, which is a generalization of the Gaussian Dropout with the particularity of learning the dropout rate instead of randomly select one value. The authors aimed to reduce the variance of the stochastic gradients considering the variational Bayesian inference of a posterior over the model parameters, retaining the parallelization by investigating the reparametrization approach.

Further, Gal et al. (**??**) proposed a new Dropout variant to reinforcement learning models. Such a method aims to improve the performance and calibrate the uncertainties once it is an intrinsic property of the Dropout. The proposed approach allows the agent to adapt its uncertainty dynamically as more data is provided. Molchanov et al. (**??**) explored the Variational Dropout proposed by Kingma et al. (**??**). The authors generalized the method to situations where the dropout rates are unbounded, giving very sparse solutions in fully-connected and convolutional layers. Moreover, they achieved a reduction in the number of parameters up to 280 times on LeNet architectures and up to 68 times on VGG-like networks with a small decrease in accuracy rates. Such a fact highlights the importance of sparsity for robustness and parameter reduction, while the overall performance for "simpler" models can be improved.

Another class of regularization methods emerged in parallel, i.e., techniques that change the neural network's input. Among such methods, one can refer to the Cutout (**??**), which works by cutting off/removing a region of the input image and setting such pixels at zero values. Such a simple approach provided relevant results in several datasets. In a similar fashion emerged the RandomErasing (**??**), which works by changing the pixel values at random for a given region in the input, instead of setting these values for zero.

Roder et al. (**??**) proposed the Energy-based Dropout, a method that makes conscious decisions whether a neuron should be dropped or not based on the energy analysis. The authors designed such a regularization method by correlating neurons and the model's energy as an index of importance level for further applying it to energy-based models, as Restricted Boltzmann Machines.

4.4 MaxDropoutV2 as an improved version of MaxDropout

This section provides an in-depth introduction to MaxDropout-based learning.

4.4.1 MaxDropout

MaxDropout (**??**) is a Dropout-inspired (**??**) regularization task designed to avoid overfitting on deep learning training methods. The main difference between both techniques is that, while Dropout randomly selects a set of neurons to be cut off according to a Bernoulli distribution, MaxDropout establishes a threshold value, in which only neurons whose activation values higher than this threshold are considered in the process. Results provided in (**??**) show that excluding neurons using their values instead of the likelihood from a stochastic distribution while training convolutional neuron networks produces more accurate classification rates.

Algorithm 4.1 implements the MaxDropout approach. Line 2 generates a normalized representation of the input tensor. Line 3 attributes the normalized value to a vector to be returned. Further, Lines 4 and 5 set this value to 0 where the normalized tensor is bigger than the threshold. This process is only performed during training. Concerning the inference, the original values of the tensor are used.

Listing 4.1: Original MaxDropout code

```
1 def MaxDropout(tensor, threshold):
2     norm_tensor = normalize(tensor)
3     return_tensor = norm_tensor
4     if norm_tensor > threshold:
5         return_tensor = 0 where
6     return return_tensor
```

Even though MaxDropout obtained satisfactory results for the task, it was not tailored-designed for Convolutional Neural Networks (CNNs), thus presenting two main drawbacks:

- ❏ it does not consider the feature map spacial distribution produced from a CNN layer output since it relies on individual neurons, independently of their location on a tensor; and
- ❏ it evaluates every single neuron from a tensor, which is computationally expensive.

Such drawbacks motivated the development of an improved version of the model, namely MaxDropoutV2, which addresses the issues mentioned above and provides a faster and more effective approach. The following section describes the technique.

4.4.2 MaxDropoutV2

The main difference between MaxDropout and MaxDropoutV2 is that the latter relies on a more representative feature space. While MaxDropout compares the values from each neuron directly, MaxDropoutV2 sums up these feature maps considering the depth axis, thus providing a bidimensional representation. In a nutshell, consider a CNN layer output tensor with dimensions $32 \times 32 \times 64$. The original MaxDropout performs $32 \times 32 \times 64$, i.e., $65,536$ comparisons. The proposed method sums up the values of the tensor over axis one (which would be the depth of the tensor) for each 32×32 kernel, thus performing only $1,024$ comparisons.

Algorithm 4.2 provides the implementation of the proposed approach. Line 2 performs the sum in the depth axis. Similar to Algorithm 4.1, Line 3 generates a normalized representation of the sum in depth of the input tensor. Line 4 creates the mask that defines what positions of the original tensor should be dropped, i.e., set to 0. Notice that the process is performed faster in MaxDropoutV2 due to the reduced dimensionality of the tensor. Further, in Lines 5 and 6, the tensor is unsqueezed and repeated so the mask can be used along all the tensor dimensions. Finally, the mask is applied to the tensor in Line 8 and returned in Line 9. These operations are only performed during training, similar to the original.

Listing 4.2: MaxDropoutV2 code

```
1 def MaxDropout_V2(tensor, threshold):
2   sum_axis = sum tensor along axis 1
3   sum_axis = normalized(sum_axis)
4   mask = 0 where sum_axis > threshold
5   mask_tensor = tensor.shape[0]
6   repetitions of mask
7
8   return_tensor = tensor * mask_tensor
9   return return_tensor
```

Fig. 11 depicts an example of application, presenting an original image in Fig. 11a and a simulation of output colors considering Dropout, MaxDropout, and MaxDropoutV2, for Figs. 11b, 11c, and 11d, respectively.

4.5 Methodology

This section provides a brief description of the datasets employed in this work, i.e., CIFAR-10 and CIFAR-100, as well all the setup considered during the experiments.

4.5.1 Dataset

In this work, we consider the public datasets CIFAR-10 and CIFAR-100 (**??**) to evaluate the performance of MaxDropoutV2 since both datasets are widely employed in similar regularization contexts (**????????**). Both datasets comprise $60,000$ color images

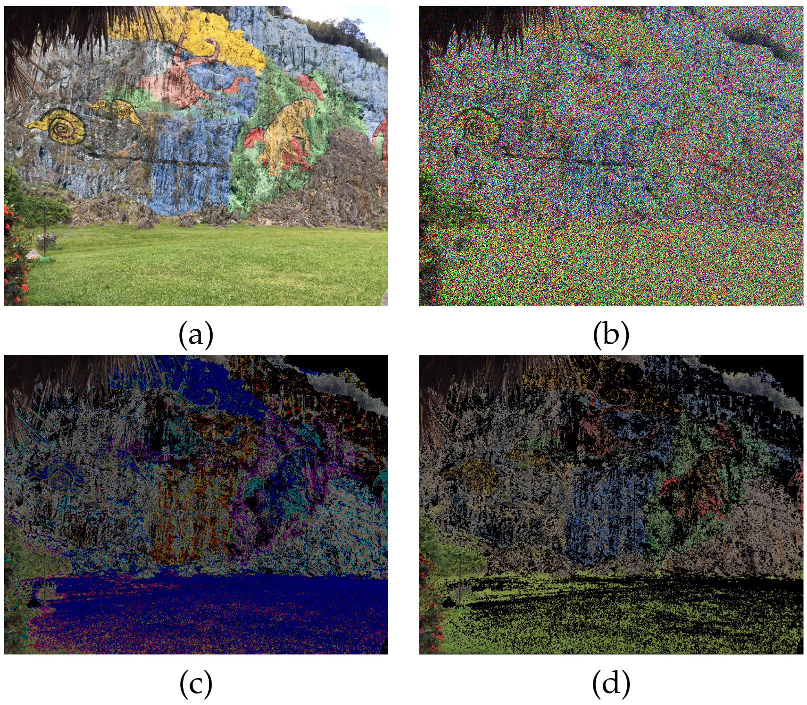


Figure 11: Simulation using colored images. The original colored image is presented in (a), and its outcomes are bestowed after (b) Dropout, (c) MaxDropout, and (d) MaxDropoutV2 transformations using a dropout rate of 50%.

of animals, automobiles, and ships, to cite a few, with a size of 32×32 pixels. Such images are divided such that $50,000$ instances are employed for training, and $10,000$ samples are considered for evaluation purposes. The main difference between CIFAR-10 and CIFAR-100 regards the number of classes, i.e., CIFAR-10 comprises 10 classes while CIFAR-100 is composed of 100 classes.

4.5.2 Experimental Setup

To provide a fair comparison, we adopted the same protocol employed in several works in the literature (**??????**), which evaluate the proposed techniques over the ResNet-18 (**??**) neural network. Regarding the pre-processing steps, each image sample is resized to 32×32 pixels for further extracting random crops of size 28×28 pixels, with the addition of horizontal flip. The network hyperparameter setup employs the Stochastic Gradient Descent (SGD) with Nesterov momentum of 0.9 and a weight decay of 5×10^{-4}. The initial learning rate is initially set to 0.1 and updated on epochs 60, 120, and 160 by multiplying its value by 0.2. Finally, the training is performed during a total of 200 epochs and repeated during five rounds over each dataset to extract statistical measures. It is important to highlight that this protocol is used in several other works related to regularization on Deep Learning models (**??????**). In this work, we compare our proposed method against other regularizers that explicitly target to improve the results of CNNs.

Regarding the hardware setup, experiments were conducted using an Intel 2x Xeon®E5-2620 @ 2.20GHz with 40 cores, a GTX 1080 Ti GPU, and 128 GB of RAM.

4.6 Experimental Results

This section provides an extensive set of experiments where MaxDropoutV2 is compared against several baselines considering both classification error rate and time efficiency. Additionally, it also evaluates combining MaxDropoutV2 with other regularization techniques.

4.6.1 Classification Error

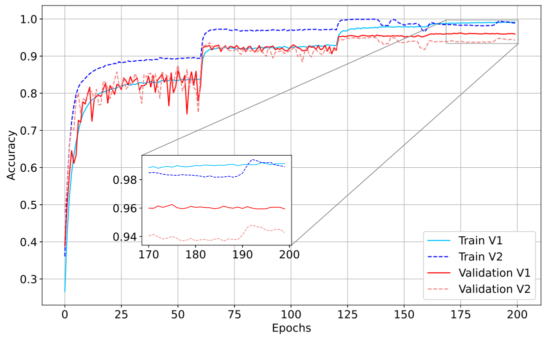
Table 8 shows the average error rate for all models and architectures regarding the task of image classification. Highlighted values denote the best results, which were obtained over five independent repetitions.

Table 8: Average classification error rate (%) over CIFAR-10 and CIFAR-100 datasets. Notice ResNet18 results are provided as the baseline.

	CIFAR-10	**CIFAR-100**
ResNet-18 (**??**)	4.72 ± 0.21	22.46 ± 0.31
Cutout (**??**)	$\mathbf{3.99 \pm 0.13}$	21.96 ± 0.24
RandomErasing (**??**)	4.31 ± 0.07	24.03 ± 0.19
LocalDrop (**??**)	4.3	22.2
MaxDropout (**??**)	4.66 ± 0.13	21.94 ± 0.07
MaxDropoutV2 (ours)	4.63 ± 0.04	$\mathbf{21.92 \pm 0.23}$

From the results presented in Table 8, one can observe that Cutout obtained the lowest error rate over the CIFAR-10 dataset. Meanwhile, MaxDropoutV2 achieved the most accurate results considering the CIFAR-100 dataset, showing itself capable of outperforming its first version, i.e., MaxDropout, over more challenging tasks composed of a higher number of classes.

Additionally, Figure 12 depicts the convergence evolution of MaxDropout and MaxDropoutV2 over the training and validation splits, in which the training partition comprises $50,000$ samples, and the validation contains $10,000$ samples. In Figure 12, V1 stands for the MaxDropout method, and V2 stands for the proposed approach. One can notice that MaxDropoutV2 does not overpass the MaxDropout validation accuracy, mainly on the CIFAR-10 dataset. However, when dealing with more classes and the same number of training samples, both performances was almost the same, indicating the robustness of MaxDropoutV2.



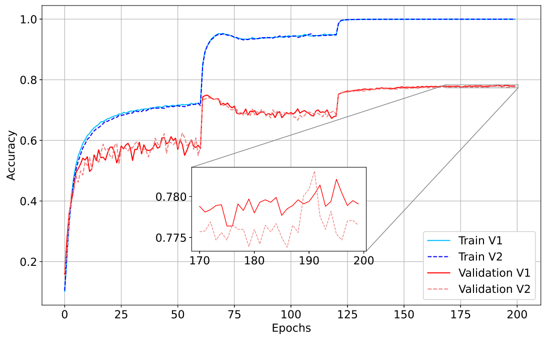


Figure 12: Convergence analysis regarding the CIFAR-10 and CIFAR-100 datasets.

4.6.2 Combining Regularization Techiniques

A critical point about regularization concerns avoiding overfitting and improving the results of a given neural network architecture in any case. For instance, if some regularization approach is already applied, including another regularization should still improve the outcomes. In this context, MaxDropoutV2 performs this task with success, as shown in Table 9.

Table 9: Average classification error rate (%) over CIFAR-10 and CIFAR-100 datasets combining MaxDropout and MaxDropoutV2 with Cutout.

	CIFAR-10	**CIFAR-100**
ResNet-18 (**??**)	4.72 ± 0.21	22.46 ± 0.31
Cutout (**??**)	3.99 ± 0.13	21.96 ± 0.24
MaxDropout (**??**)	4.66 ± 0.13	21.94 ± 0.07
MaxDropoutV2	4.63 ± 0.04	21.92 ± 0.23
MaxDropout + Cutout (**??**)	$\mathbf{3.76 \pm 0.08}$	$\mathbf{21.82 \pm 0.13}$
MaxDropoutV2 + Cutout (**??**)	3.95 ± 0.13	$\mathbf{21.82 \pm 0.12}$

4.6.3 Performance Evaluation

The main advantage of MaxDropoutV2 over MaxDropout regards its computational time. In this context, Tables 10 and 11 provides the average time demanded to train both models considering each epoch and the total consumed time. Such results confirm the hypothesis stated in Section 4.4.2 since MaxDropoutV2 performed around 10% faster than the standard version.

Table 10: Time consumed in seconds for training ResNet-18 in CIFAR-10 dataset.

	Seconds per Epoch	**Total time**
MaxDropout (**??**)	32.8	6,563
MaxDropoutV2 (ours)	**29.8**	**5,960**

Table 11: Time consumed in seconds for training ResNet-18 in CIFAR-100 dataset.

	Seconds per Epoch	**Total time**
MaxDropout (**??**)	33.1	6,621
MaxDropoutV2 (ours)	**30.2**	**6,038**

4.6.4 Evaluating Distinct Drop Rate Scenarios

This section provides an in-depth analysis of MaxDropoutV2 and MaxDropout (**??**) results considering a proper selection of the drop rate parameter. Tables 12 and 13 present the models' results while varying the drop rate from 5% to 50% considering CIFAR-10 and CIFAR-100 datasets, respectively.

Table 12: Mean error (%) concerning CIFAR-10 dataset.

Drop Rate	**MaxDropoutV2**	**MaxDropout**
5	**4.63 ± 0.03**	4.76 ± 0.09
10	4.67 ± 0.13	4.71 ± 0.09
15	4.76 ± 0.12	**4.63 ± 0.11**
20	4.66 ± 0.13	4.70 ± 0.08
25	4.75 ± 0.11	4.70 ± 0.06
30	4.63 ± 0.16	4.67 ± 0.12
35	4.70 ± 0.18	4.71 ± 0.16
40	4.74 ± 0.13	4.79 ± 0.20
45	4.65 ± 0.16	4.71 ± 0.11
50	4.71 ± 0.04	4.75 ± 0.10

Even though MaxDropoutV2 did not achieve the best results in Table 8, the results presented in Table 12 show the technique is capable of yielding satisfactory outcomes considering small drop rate values, i.e., 5%, while the standard model obtained its best results considering a drop rate of 15%. Additionally, one can notice that MaxDropoutV2 outperformed MaxDropout in eight-out-of-ten scenarios, demonstrating the advantage of the model over distinct circumstances.

Table 13: Mean error (%) concerning CIFAR-100 dataset.

Drop Rate	MaxDropoutV2	MaxDropout
5	22.26 ± 0.31	22.05 ± 0.17
10	22.19 ± 0.13	22.06 ± 0.32
15	22.25 ± 0.23	22.16 ± 0.20
20	22.26 ± 0.30	21.98 ± 0.21
25	22.02 ± 0.13	$\mathbf{21.94 \pm 0.23}$
30	$\mathbf{21.92 \pm 0.23}$	22.07 ± 0.24
35	22.00 ± 0.07	22.10 ± 0.29
40	22.09 ± 0.16	22.16 ± 0.34
45	21.95 ± 0.15	22.31 ± 0.29
50	22.13 ± 0.19	22.33 ± 0.23

In a similar fashion, Table 13 provides the mean classification error considering distinct drop rate scenarios over CIFAR-100 dataset. In this context, both techniques required larger drop rates to obtain the best results, i.e., 25 and 30 for MaxDropout and MaxDropoutV2, respectively. Moreover, MaxDropoutV2 outperformed MaxDropout in all cases when the drop rates are greater or equal to 30, showing more complex problems demand higher drop rates.

4.6.5 Discussion

According to the provided results, the proposed method accomplishes at least equivalent outcomes to the original MaxDropout, outperforming it in terms of classification error in most cases. Moreover, MaxDropoutV2 presented itself as a more efficient alternative, performing around 10% faster than the previous version for the task of CNN training.

The main drawback regarding MaxDropoutV2 is that the model is cemented to the network architecture, while MaxDropout applicability is available to any network's architecture. In a nutshell, MaxDropoutV2 relies on a matrix or high dimensional tensors designed to accommodate CNNs' layers outputs, while the standard MaxDropout works well for any neural network structure, such as Multilayer Perceptrons and Transformers, for instance.

4.7 Conclusion and Future Works

This paper presented an improved version of the regularization method MaxDropout, namely MaxDropoutV2, which stands for a tailored made regularization technique for convolutional neural networks. In short, the technique relies on a more representative feature space to accommodate the convolutional layer outputs.

Experimental results showed the method significantly reduced the time demanded to train the network, performing around 10% faster than the standard MaxDropout with similar or more accurate results. Moreover, it demonstrated that MaxDropoutV2 is more robust to the selection of the drop rate parameter. Regarding future work, we will evaluate MaxDropoutV2 in distinct contexts and applications, such as object detection and image denoising.

Chapter 5

Rethinking Regularization with Random Label Smoothing

This chapter presents a paper under revision in the journal Pattern Recognition Letters. This work proposes a new label regularization for three different image processing tasks: image classification, image super-resolution and software ISP via Deep Learning.

5.1 Abstract

Regularization helps to improve machine learning techniques by penalizing the models during training. Such approaches act in either the input, internal, or output layers. Regarding the latter, label smoothing is widely used to introduce noise in the label vector, making learning more challenging. This work proposes a new label regularization method, Random Label Smoothing, that attributes random values to the labels while preserving their semantics during training. The idea is to change the entire label space into fixed arbitrary values. Results show improvements in image classification and super-resolution tasks, outperforming state-of-the-art techniques for such purposes.

5.2 Introduction

Neural networks are acknowledged to be learn-by-example techniques. Assuming the training set is representative, the problem becomes finding proper loss functions to avoid local optima and a good backbone. We know that some Convolutional

Neural Networks (CNNs) are more accurate than others. In image classification tasks, ResNet (**??**) usually performs better than VGG (**??**), for it has a more complex architecture and residual connections that help generalization (**??**).

Changing the training protocol is another approach to increase generalization. Notably, neural networks can obtain better results when more (and proper) training instances are available. One can also train a model in some larger dataset before fine-tuning it to the desired problem, i.e., transfer learning (**??**). Results show it can boost the outcomes significantly in a variety of problems. Google improved results in the ImageNet challenge (**??**) by creating a huge dataset with more than 300 million images to first train a model and further fine-tune it in the ImageNet set (**??**).

However, some scenarios do not allow us to collect additional training data, for the labeling cost is prohibitive. Regularization can come to this aid by either making training harder or the loss function landscape smoother (**??**). We expect to achieve improvements in the model's generalization after the application of such approaches. Classical regularization approaches in deep learning include data augmentation, which shall consider semantics after transformation. A model that trains on the MNIST dataset (**??**), for instance, can rotate images to some extent only. Rotating a "6" in 180 degrees ends up in a "9", generating a wrong instance-label pair.

This work introduces Random Label Smoothing (RLS), a new regularization approach that works on the output layer. RLS operates by randomly changing values in the label vector (ground truth). We demonstrate state-of-the-art results in image classification and super-resolution, evidencing it can be used in a broad range of application domains.

The manuscript is organized as follows. Sections 5.3 and 5.4 present some related works and the proposed approach, respectively. Section 5.5 introduces the methodology, and Section 5.6 demonstrates the robustness of the proposed approach in experiments under different scenarios. Section 5.7 provides a brief discussion about the outcomes, and Section 5.8 states conclusions.

5.3 Related Works

Several regularization techniques are available for helping neural networks to accomplish better results in different domains. Regularization based on data augmentation is classic and with many approaches in the literature. AutoAugment (**??**) performs data augmentation by first learning the best policy for creating synthetic samples. However, it may take too long to determine the best data augmentation strategy for a given data set. Aiming to make this process faster, Fast AutoAugment (**??**) calculates the gradient from just one batch, decreasing the computational effort considerably. Other methods, such as Cutout (**??**) and RandomErasing (**??**), work by removing random areas of the image. The former removes a patch and leaves its content empty, while the other fills it with some random noise.

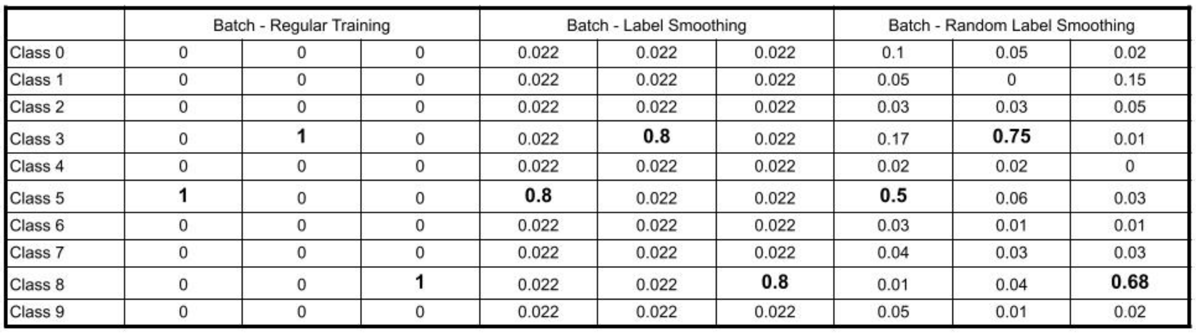
Other methods operate by changing the feature maps generated during training. Dropout (**??**) randomly drops neurons, while MaxDropout (**??**) eliminates the most active ones, i.e., the neurons with the highest activation values. An improved version, called MaxDropoutV2 (**??**), includes a more efficient approach to finding the most active neurons. Instead of directly comparing values on the output feature map from a given layer, it first sums the value of each neuron in the depth axis for further performing the comparison. MaxDropoutV2 carries more semantic information than its original counterpart. Additional methods consider other internal aspects when training CNNs. Shake-Shake (**??**) changes the weights of the inference and the backpropagation values on training time in a multi-branch model, such as ResNeXT (**??**). Results show it can significantly improve the results.

A recent analysis of regularization methods for CNNs (**??**) raised some interesting drawbacks in the area. The first one is the shortage of algorithms that perform regularization on a label level. The other point concerns the application domain, i.e., most regularization techniques designed for deep nets focus on image classification. Label Smoothing (**??**) changes the values of the output layer (label vector), i.e., it decreases the value of the position that represents the true label and increases the values of the inactive labels. The Two-Stage Label Smoothing (TSLA) (**??**) changes the label values to some extent during training. The work shows that stopping label smoothing in a late training stage helps the model to generalize better.

5.4 Proposed Approach

According to (**??**), there are several issues related to some new regularization methods. We address quite a few of them in this work. Regularization approaches are often evaluated in a single context only, primarily on image classification. Here, we also consider image super-resolution. Still, according to (**??**), a good regularization technique should improve results even if the model is already using another regularization technique, which RLS is capable of.

Traditional data augmentation usually changes features in the input data. A simple way to perform data augmentation is to rotate the image to the left or to the right in random degrees. Another way is to crop some areas of the input image, e.g., Cutout (**??**). Following a similar logic, RLS augments the data by performing random but controlled changes in the label of every single instance during training. We explain how to do that for image classification and super-resolution tasks.



	Batch - Regular Training			Batch - Label Smoothing			Batch - Random Label Smoothing		
Class 0	0	0	0	0.022	0.022	0.022	0.1	0.05	0.02
Class 1	0	0	0	0.022	0.022	0.022	0.05	0	0.15
Class 2	0	0	0	0.022	0.022	0.022	0.03	0.03	0.05
Class 3	0	**1**	0	0.022	**0.8**	0.022	0.17	**0.75**	0.01
Class 4	0	0	0	0.022	0.022	0.022	0.02	0.02	0
Class 5	**1**	0	0	**0.8**	0.022	0.022	**0.5**	0.06	0.03
Class 6	0	0	0	0.022	0.022	0.022	0.03	0.01	0.01
Class 7	0	0	0	0.022	0.022	0.022	0.04	0.03	0.03
Class 8	0	0	**1**	0.022	0.022	**0.8**	0.01	0.04	**0.68**
Class 9	0	0	0	0.022	0.022	0.022	0.05	0.01	0.02

Figure 13: Simulation of Label Smoothing and Random Label Smoothing over a batch of labels during training for a classification model (active label in bold). Traditionally, the active label is set to"1" while all other classes' indices are set to '0". In Label Smoothing, the active class is set to a constant (and higher) value, while the inactive classes are set to a smaller invariant value. In Random Label Smoothing, the active label receives a random (and higher) value (e.g., greater than 0.5) while the inactive labels receive a random value that, summed with the active label value, reaches 1.

5.4.1 Image Classification

Concerning image classification, we vary the output values that define the label in a controlled but random range of values. In the "active" position, i.e., the index represented by the value "1" that encodes the label (one-hot representation), we randomly decrease its values between 0.05 and 0.49, guaranteeing that the active label will always have the greatest value. For the inactive positions (defined as "0"), we divided the amount of value used earlier among these positions. For instance, if the problem has 10 classes and the removed value is 0.3, the active position is set to 0.7, and all the other 9 positions receive a portion of the remaining value.

By doing these transformations in the labels during training, we create various acceptable labels for a given instance, working as an augmented label algorithm. Our results show it is functional for image classification, helping the model to generalize better and overcoming other methods, such as TargetDrop (**??**) and MaxDropout (**??**). Figure 13 demonstrates how RLS works for a classification problem in a toy example.

5.4.2 Image Super-Resolution

For image reconstruction, we first tried a similar approach to the classification task by randomly changing the label's values following a Gaussian distribution[1]. However, it did not work as expected. The new reference image (modified ground truth) has Gaussian noise by changing the pixel values using a Gaussian distribution. The entire system then learns to reconstruct images with noise.

[1] Now, the pixel values encode the labels.

Changing the values pixel-by-pixel with different random values did not seem to be a good strategy, for we lose semantic details. We, therefore, decided to perturb all pixels by the same amount, i.e., a random value that can be either added or subtracted by the pixel value in a given training interaction. The problem is now defining what values range leads to the best results.

We achieved promising outcomes by reverse-engineering the results of the neural networks employed in this study. We used the results (i.e., PSNR - peak signal-to-noise ratio) of each architecture to set a range of values with better results. For instance, PyNET (**??**) achieved a 21.19 dB of PSNR; then, converting this value to a gray-scale amount results in about 12 units. Therefore, all pixel values (for all dataset images) were either subtracted or added concerning that amount (in this example). For EDSR (Enhanced Deep Residual Networks for Single Image Super-esolution) (**??**), the results are a PSNR of 29.21 dB for Div2k (**??**) and 28.89 dB for the RealSR dataset (**??**). Therefore, we set 4.5 units for both datasets.

We verified if the above methodology could be further improved in the last experiment. We found out that we could achieve even better results by using half of the range than using the full range. In this case, we used 6 units for the PyNet and 2.25 for EDSR. Even though the image regions may be different, smaller differences in continuous regions can help the entire model understand that smaller errors are more acceptable than the exact value.

5.5 Methodology

This section provides a complete description of the experimentation protocol we used to evaluate RLS. We divided the experiments into three main parts: first, all four architectures we used for evaluating purposes are described. Right after, we presented the training protocol and later a description of the datasets.

5.5.1 Scenarios

We considered three different scenarios to evaluate RLS. They all have, in some ways, differences in the input data or the label level. The first one is standard image classification, and the second task concerns image super-resolution. In this case, the neural network's task is to magnify the image input, creating another but amplified. For example, for a magnification of four times, if the input has the size of 200×200, the model's objective is to create an output of size 800×800.

Last but not least, another challenge is to simulate an image signal processor (ISP), which basically creates an RGB image from a CFA (color filter array) acquired by the camera's sensor. Therefore, given a CFA input, the task is to learn a CNN that can generate its corresponding RBG output.

5.5.2 Neural Network Architecture

As mentioned earlier (**??**), a good regularization technique should improve results in different problems to show it can enhance a given CNN outcome. We tested four neural backbones in different neural architectures to provide a fair evaluation: two for image classification, one for single image super-resolution, and one for software ISP.

The first CNN we use to evaluate RLS is ResNet (**??**), more precisely, ResNet-18. We have chosen this architecture because it is widely used for evaluating regularization techniques, allowing a natural comparison. Such neural backbone comprises a sequence of convolutional and pooling layers, with pooling after a sequence of two or three convolutional layers. The significant innovation in its architecture concerns the residual connections, which may improve effectiveness to a certain extent.

EDSR (**??**) is one of the scarce neural networks used to evaluate regularization methods (**??**), ending up in another natural choice. It stands for a residual convolutional network with a sequence of convolutional-ReLU activation-convolutional operations in its residual blocks and pixel shuffle operations (**??**) to perform image super-resolution in the end. PyNET (**??**), a multi-branch CNN that has several layers in parallel and uses different measures for error calculation, is interesting in evaluating problems related to image and signal processing, specifically image reconstruction.

5.5.3 Training Protocol

For the image classification problem, we considered the protocol suggested by (**??**). The images were redimensioned to 32×32 pixels and then randomly cropped in 28×28 patches. Stochastic gradient descent with Nesterov momentum is used for gradient calculation. The learning rate starts at $10e-2$ and is multiplied by $10e-1$ on epochs 80, 120, and 160.

Concerning image super-resolution, we did not find any defined or suggested protocol. We, therefore, followed the same parameters used in CutBlur (**??**) and PyNET (**??**). We understand a natural comparison to these previous works by observing the same parameters.

In all scenarios, five training runs were performed to avoid comparing results only by chance. Our results report the mean and standard deviation values for each performance measure.

5.5.4 Datasets

We used a different dataset for each task evaluated in this work to allow a fair comparison against other methods. In each case, we selected datasets that, according to our research, are the most used ones on each application domain considered here, i.e., image classification and super-resolution.

For the image classification task, we appointed CIFAR-100 (??), one of the most used datasets to evaluate regularization techniques (??). It comprises 50, 000 images from 100 different classes for training purposes and 10, 000 images as the validation set.

For the single image super-resolution task, we considered two different datasets inspired in (??). The first one stands for the Div2K (??) dataset, with 800 pairs of low and high-resolution RGB images for training and 100 for evaluation. The other dataset is RealSR (??), which comprises 459 pairs of images for training and 100 for model validation. We used a magnification of four times for comparison purposes in both cases.

The last one is the Zurich RAW to RGB Dataset (??), which evaluates techniques for image reconstruction. This dataset is divided into 46, 839 pairs of RGB Bayer filter data/RGB image for training and 1, 204 similar pairs for testing purposes.

5.6 Experimental Results

This section provides outcomes of RLS against some state-of-the-art regularization approaches. RLS is first evaluated over image classification tasks (Section 5.6.1) and later on image super-resolution problems (Section 5.6.2).

5.6.1 Image Classification

Table 14 presents the error rate concerning ResNet-18 in CIFAR-100 dataset. RLS has achieved the best outcome solely, outperforming seven other techniques. The first row in the table is our baseline, i.e., ResNet-18, without any regularization.

Method	Error (%)
ResNet-18 (??)	24.50
Cutout (??)	21.96
RandomErasing (??)	24.03
MaxDropout (??)	21.93
MaxDropoutV2 (??)	21.92
TSLA (??)	21.45
TargetDrop (??)	21.45
RLS	**21.18 ± 0.35**

Table 14: Image classification experiment over CIFAR-100 dataset.

5.6.1.1 Working Along with Other Regularizers

As mentioned by (??), it is vital to check how a particular regularization algorithm works along with other regularization methods. Here, we provide some interesting outcomes. Table 15 presents results of ResNet-18 using Cutout and other methods working together. The proposed approach can outperform MaxDropout and Target-Drop working jointly with Cutout by more than 0.5% on average, which is to be considered a good improvement.

Method	Error (%)
ResNet-18 (**??**)	24.50
Cutout (**??**)	21.96
MaxDropout + Cutout (**??**)	21.82
MaxDropoutV2 + Cutout (**??**)	21.82
TargetDrop + Cutout (**??**)	21.25
RLS + Cutout	**20.6 ± 0.16**

Table 15: Results on CIFAR-100 using ResNet-18 with one or more regulization methods.

Combining PyramidNet with ShakeDrop and RLS results in some improvement too. Table 16 shows the outcomes of PyramidNet without any regularization, using ShakeDrop, and using ShakeDrop+RLS. On average, that combination allowed an improvement of 0.1%.

Method	Error (%)
PyramidNet (**??**)	16.35
ShakeDrop (**??**)	16.22
ShakeDrop + RLS	**16.12 ± 0.14**

Table 16: Results on CIFAR-100 using PyramidNet with one or more regulization methods.

5.6.2 Image Super-Resolution

Table 17 shows the outcomes of EDSR backbone using different regularization techniques from (**??**). Some conclusions can be drawn in this scenario. The first one concerns Div2k dataset, whose results show that RLS using half of the perturbation value (RLS-Half) outperforms all methods, including the situation when all techniques (i.e., EDSR, Cutout, Cutmix, Mixup, RGB permutation, Blend, and Cutblur) are used together (All).

The second analysis concerns the outcomes of the RealSR dataset. Although RLS did not overcome the neural network trained using all methods, still, it has the best individual result.

We considered an additional experiment related to image reconstruction. Table 18 shows the outcomes of PyNET using RLS algorithm considering PSNR and Multiscale

	Div2K	RealSR
EDSR	29.21	28.89
Cutout	29.22 ± 0.01	28.95 ± 0.06
Cutmix	29.22 ± 0.01	28.89 ± 0.00
Mixup	29.26 ± 0.05	28.98 ± 0.09
CutMixup	29.27 ± 0.06	29.03 ± 0.14
RGB Permutation	29.30 ± 0.09	29.02 ± 0.13
Blend	29.23 ± 0.02	29.03 ± 0.14
Cutblur	29.26 ± 0.05	29.12 ± 0.23
All	29.30 ± 0.30	$\mathbf{29.16 \pm 0.27}$
RLS-Gaussian	28.03 ± 0.07	27.97 ± 0.04
RLS-Full	29.31 ± 0.01	29.05 ± 0.04
RLS-Half	$\mathbf{29.32 \pm 0.01}$	29.15 ± 0.03

Table 17: PSNR results on Div2K and RealSR datasets for EDSR using regulization methods.

Structural Similarity Index Measure (MS-SSIM) (**??**) quality measures. An improvement in the original results (i.e., standard PyNet) can be observed when RLS is applied. It is worth mentioning that, even though there are several regularization methods available for CNNs, none of them tackles, or at least is evaluated, in the context of image reconstruction. As far as we are aware (**??**), this is the first regularization approach that improves the results of deep learning models in the aforementioned task.

Method	PSNR	MS-SSIM
PyNet (**??**)	21.19	0.862
Pynet + Gaussian-RLS	20.86	0.850
Pynet + Full-RLS	21.21	0.863
Pynet + Half-RLS	$\mathbf{21.22 \pm 0.01}$	**0.867**

Table 18: Results on Zurich RAW to RGB Dataset for PyNet.

5.7 Discussion

Providing new regularization algorithms is not straightforward for it often needs the knowledge of a specialist in the problem. Deep learning by itself is already an area of research that demands plenty of work when some improvements are required. This section provides some discussion about the outcomes obtained in the previous section.

5.7.1 Lack of Label Regularization Methods

Achieving the best results using a neural network is always desired, and regularization methods should be encouraged in most cases, as far as it does not break the semantics of the dataset. The label can be considered safe to test for multi-class classification, regardless of the application domain. The output is often hot-encoded, so there are few possibilities to harm or lose semantics.

This work is about a regularization method for Convolutional Neural Networks. It is fair and easy to compare with other algorithms because it follows the same evaluation protocol. However, more techniques rather than TSLA are desired to perform a better and direct comparison.

5.7.2 Lack of Comparison for General-purpose Applications

There might be a bias toward creating deep learning regularization algorithms only for the image classification task. We could find several regularization methods (**????????**) for comparison purposes; however, we found only one for directly comparing in the context of image super-resolution (**??**). Besides, as far as we are aware, no other in-depth study compared regularization techniques for image reconstruction.

The scarcity of works that aimed to compare regularization techniques in problems other than image classification is worrying. Indeed, we found some works (**????**) that also complain about this scarcity of research on regularization algorithms in more problems. We encourage the researchers to develop new methods for other image processing problems, for there might be a promising area of research.

5.8 Conclusions and Future Works

We presented the RLS technique for label-level regularization concerning Convolutional Neural networks. Our results demonstrate that it can outperform other techniques when applied to different image processing problems. As such, we tackle not only the enhancement of neural networks but the problem of generalizing regularization algorithms, complained by (**??**). RLS can be combined with other techniques and be used within any backbone.

We intend to apply RLS to other problems in future works, such as natural processing language processing. Another intent is to check if there are random distributions that can improve our current results.

Chapter 6

Conclusion

The present thesis was organized into six chapters, described as follows: an introduction that describes the context, the motivation, and the main contribution to the subject. Chapter 2 presented a background review in the area in a survey format, which was published in the ACM Computing Surveys (**??**), with the title “Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks". In this work, we show the works we analized, as well as point some problems in recent researches targeting new regularization methods. Chapter 3 presented a paper published in 25th International Conference on Patter Recognition (ICPR 2020), named “MaxDropout: Deep Neural Network Regularization Based on Maximum Output Values" and Chapter 4 presented the paper “MaxDropoutV2: An Improved Method to Drop Out Neurons in Convolutional Neural Networks", presented on the 10th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2022). The former presented a new approach for dropping neurons during training while the latter presented an improved technique targeting convolutional neural networks.

The content of the Chapter 5 was submited to the Patten Recognition Letters (PR Letters) with the title “Rethinking Regularization with Random Label Smoothing". In this research, we demonstrated an algorithm that can perform regularization on a label level in different tasks, accomplishing relevant results in the tested problems, suggesting this approach can be used in several different tasks.

The results obtained in this thesis confirms the hypothesis that we can reduce the overfitting of convolutional neural networks by applying new regularization algorithms during training, encouraging the development for new techniques applied to Deep Learning models.

Name	Type	Qualis	Year	Status
Does pooling really matter? an evaluation on gait recognition	Conference	A3	2019	Published
A hybrid approach for breast mass categorization	Conference	A4	2019	Published
BreastNet: breast cancer categorization using convolutional neural networks	Conference	A3	2020	Published
Does Removing Pooling Layers from Convolutional Neural Networks Improve Results?	Journal	A4	2020	Published
Image Denoising using Attention-Residual Convolutional Neural Networks	Conference	A3	2020	Published
Normalizing images is good to improve computer-assisted COVID-19 diagnosis	Book Chapter	—	2021	Published
MaxDropout: Deep Neural Network Regularization Based on Maximum Output Values	Conference	A2	2021	Published
Improving Pre-Trained Weights through Meta-Heuristics Fine-Tuning	Conference	A4	2021	Published
MaxDropoutV2: An Improved Method to Drop Out Neurons in Convolutional Neural Networks	Conference	A4	2022	Published
Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks	Journal	A1	2022	Published
Gait Recognition Based on Deep Learning: A Survey	Journal	A1	2022	Published
ISP meets Deep Learning: A Survey on Deep Learning Methods for Image Signal Processing	Journal	A1	2022	Submitted
Rethinking Regularization with Random Label Smoothing	Journal	A1	2022	Submited

Table 19: List of publications developed by the author.

6.1 Publications and Other Works

Table 19 presents a list of studies developed during the research period.

6.2 Future Works

I have been working as Artificial Inteligence Research Lead and Consultant in Eldorado Research Institute since 2020. As a researcher the work includes but is not limited to image super-resolution, software ISP via Deep Learning, sales forecasting, and XAI, coordinating a team of about 15 researchers. As future work, I intend to continue researches in the Deep Learning area, aiming the mentioned problems and perhaps others that can request my attention.

Bibliography

AGUSTSSON, E.; TIMOFTE, R. Ntire 2017 challenge on single image super-resolution: Dataset and study. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops**. [S.l.: s.n.], 2017.

BREIMAN, L. Bagging predictors. **Machine learning**, Springer, v. 24, n. 2, p. 123–140, 1996.

CAI, J. et al. Ntire 2019 challenge on real image super-resolution: Methods and results. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2019.

CAO, S.; NEVATIA, R. Exploring deep learning based solutions in fine grained activity recognition in the wild. In: IEEE. **2016 23rd International Conference on Pattern Recognition (ICPR)**. [S.l.], 2016. p. 384–389.

CARRATINO, L. et al. **On Mixup Regularization**. 2020.

CAVAZZA, J. et al. Dropout as a low-rank regularizer for matrix factorization. In: PMLR. **International Conference on Artificial Intelligence and Statistics**. [S.l.], 2018. p. 435–444.

CHEN, X.; YUILLE, A. L. Articulated pose estimation by a graphical model with image dependent pairwise relations. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2014. p. 1736–1744.

COATES, A.; NG, A.; LEE, H. An analysis of single-layer networks in unsupervised feature learning. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. **Proceedings of the fourteenth international conference on artificial intelligence and statistics**. [S.l.], 2011. p. 215–223.

CUBUK, E. D. et al. Autoaugment: Learning augmentation policies from data. **arXiv preprint arXiv:1805.09501**, 2018.

____. Randaugment: Practical automated data augmentation with a reduced search space. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2020. p. 702–703.

DENG, J. et al. ImageNet: A Large-Scale Hierarchical Image Database. In: **CVPR09**. [S.l.: s.n.], 2009.

DEVRIES, T.; TAYLOR, G. W. Improved regularization of convolutional neural networks with cutout. **arXiv preprint arXiv:1708.04552**, 2017.

dos Santos, C. F. G. et al. Normalizing images is good to improve computer-assisted covid-19 diagnosis. In: KOSE, U. et al. (Ed.). **Data Science for COVID-19**. Academic Press, 2021. p. 51–62. ISBN 978-0-12-824536-1. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128245361000332>.

DOULAMIS, N. Adaptable deep learning structures for object labeling/tracking under dynamic visual environments. **Multimedia Tools and Applications**, Springer, v. 77, n. 8, p. 9651–9689, 2018.

EVERINGHAM, M. et al. **The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results**. 2012. Http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

GAL, Y.; HRON, J.; KENDALL, A. Concrete dropout. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2017. p. 3581–3590.

GASTALDI, X. Shake-shake regularization. **arXiv preprint arXiv:1705.07485**, 2017.

GHIASI, G.; LIN, T.-Y.; LE, Q. V. Dropblock: A regularization method for convolutional networks. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2018. p. 10727–10737.

GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2014. p. 580–587.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GOODFELLOW, I. J. et al. Generative adversarial networks. **arXiv preprint arXiv:1406.2661**, 2014.

GUO, Y. A survey on methods and theories of quantized neural networks. **arXiv preprint arXiv:1808.04752**, 2018.

HAN, D.; KIM, J.; KIM, J. Deep pyramidal residual networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 5927–5935.

HAN, K. et al. Model rubik's cube: Twisting resolution, depth and width for tinynets. **arXiv preprint arXiv:2010.14819**, 2020.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

____. Identity mappings in deep residual networks. In: SPRINGER. **European conference on computer vision**. [S.l.], 2016. p. 630–645.

____. Identity mappings in deep residual networks. In: SPRINGER. **European conference on computer vision**. [S.l.], 2016. p. 630–645.

HE, T. et al. Bag of tricks for image classification with convolutional neural networks. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2019. p. 558–567.

HERMANS, A.; BEYER, L.; LEIBE, B. In defense of the triplet loss for person re-identification. **arXiv preprint arXiv:1703.07737**, 2017.

HINTON, G.; VINYALS, O.; DEAN, J. Distilling the knowledge in a neural network. **arXiv preprint arXiv:1503.02531**, 2015.

HO, D. et al. Population based augmentation: Efficient learning of augmentation policy schedules. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2019. p. 2731–2741.

HOFFER, E. et al. Augment your batch: better training with larger batches. **arXiv preprint arXiv:1901.09335**, 2019.

HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. **arXiv preprint arXiv:1704.04861**, 2017.

HU, J.; SHEN, L.; SUN, G. Squeeze-and-excitation networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018. p. 7132–7141.

IGNATOV, A.; GOOL, L. V.; TIMOFTE, R. Replacing mobile camera isp with a single deep learning model. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2020. p. 536–537.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **arXiv preprint arXiv:1502.03167**, 2015.

KINGMA, D. P.; SALIMANS, T.; WELLING, M. Variational dropout and the local reparameterization trick. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2015. p. 2575–2583.

KRIZHEVSKY, A. **Learning multiple layers of features from tiny images**. [S.l.], 2009.

KRIZHEVSKY, A.; NAIR, V.; HINTON, G. Cifar-10 and cifar-100 datasets. **URl: https://www. cs. toronto. edu/kriz/cifar. html**, v. 6, p. 1, 2009.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v. 25, p. 1097–1105, 2012.

LABACH, A.; SALEHINEJAD, H.; VALAEE, S. Survey of dropout methods for deep neural networks. **arXiv preprint arXiv:1904.13310**, 2019.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Ieee, v. 86, n. 11, p. 2278–2324, 1998.

LECUN, Y.; CORTES, C. MNIST handwritten digit database. 2010. Disponível em: <http://yann.lecun.com/exdb/mnist/>.

LI, W.; DASARATHY, G.; BERISHA, V. Regularization via structural label smoothing. In: CHIAPPA, S.; CALANDRA, R. (Ed.). **Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics**. PMLR, 2020. (Proceedings of Machine Learning Research, v. 108), p. 1453–1463. Disponível em: <https://proceedings.mlr.press/v108/li20e.html>.

LIM, B. et al. Enhanced deep residual networks for single image super-resolution. In: **Proceedings of the IEEE conference on computer vision and pattern recognition workshops**. [S.l.: s.n.], 2017. p. 136–144.

LIM, S. et al. Fast autoaugment. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2019. p. 6665–6675.

LU, Z. et al. Localdrop: A hybrid regularization for deep neural networks. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, 2021.

MELLOR, J. et al. Neural architecture search without training. **arXiv preprint arXiv:2006.04647**, 2020.

MNIH, V. et al. Human-level control through deep reinforcement learning. **nature**, Nature Publishing Group, v. 518, n. 7540, p. 529–533, 2015.

MOLCHANOV, D.; ASHUKHA, A.; VETROV, D. Variational dropout sparsifies deep neural networks. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2017. p. 2498–2507.

MORADI, R.; BERANGI, R.; MINAEI, B. A survey of regularization strategies for deep models. **Artificial Intelligence Review**, Springer, v. 53, n. 6, p. 3947–3986, 2020.

____. A survey of regularization strategies for deep models. **Artificial Intelligence Review**, Springer, v. 53, n. 6, p. 3947–3986, 2020.

NETZER, Y. et al. Reading digits in natural images with unsupervised feature learning. 2011.

NODA, K. et al. Audio-visual speech recognition using deep learning. **Applied Intelligence**, Springer, v. 42, n. 4, p. 722–737, 2015.

PASSOS, L. A. et al. A hybrid approach for breast mass categorization. In: SPRINGER. **ECCOMAS Thematic Conference on Computational Vision and Medical Image Processing**. [S.l.], 2019. p. 159–168.

PHAM, H.; LE, Q. V. Autodropout: Learning dropout patterns to regularize deep networks. **arXiv preprint arXiv:2101.01761**, 2021.

PHAM, H. et al. Meta pseudo labels. **arXiv preprint arXiv:2003.10580**, 2020.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 779–788.

RIBANI, R.; MARENGONI, M. A survey of transfer learning for convolutional neural networks. In: IEEE. **2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)**. [S.l.], 2019. p. 47–57.

RODER, M. et al. Intestinal parasites classification using deep belief networks. In: IEEE. **The 19th International Conference on Artificial Intelligence and Soft Computing (ICAISC)**. [S.l.], In Press.

____. Energy-based dropout in restricted boltzmann machines: Why not go random. **IEEE Transactions on Emerging Topics in Computational Intelligence**, p. 1–11, 2020.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

SANDLER, M. et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018. p. 4510–4520.

SANTANA, M. C. et al. A novel siamese-based approach for scene change detection with applications to obstructed routes in hazardous environments. **IEEE Intelligent Systems**, IEEE, v. 35, n. 1, p. 44–53, 2019.

SANTOS, C. F. G. d. et al. Maxdropout: Deep neural network regularization based on maximum output values. In: **Proceedings of 25th International Conference on Pattern Recognition, ICPR 2020, Milan, Italy, 10-15 January, 2021**. [S.l.]: IEEE Computer Society, 2020. p. 2671–2676.

SANTOS, C. F. G. d.; PAPA, J. a. P. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, jan 2022. ISSN 0360-0300. Just Accepted. Disponível em: <https://doi.org/10.1145/3510413>.

SANTOS, C. F. G. d. et al. Maxdropoutv2: An improved method to drop out neurons in convolutional neural networks. In: SPRINGER. **Iberian Conference on Pattern Recognition and Image Analysis**. [S.l.], 2022. p. 271–282.

SANTOS, C. F. G. dos et al. Does pooling really matter? an evaluation on gait recognition. In: NYSTRÖM, I.; HEREDIA, Y. H.; NÚÑEZ, V. M. (Ed.). **Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications**. Cham: Springer International Publishing, 2019. p. 751–760. ISBN 978-3-030-33904-3.

SELVARAJU, R. R. et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2017. p. 618–626.

SHI, W. et al. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 1874–1883.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. **Journal of Big Data**, Springer, v. 6, n. 1, p. 60, 2019.

SIMON, M.; RODNER, E.; DENZLER, J. Imagenet pre-trained models with batch normalization. **arXiv preprint arXiv:1612.01452**, 2016.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SPANHOL, F. A. et al. Breast cancer histopathological image classification using convolutional neural networks. In: IEEE. **2016 international joint conference on neural networks (IJCNN)**. [S.l.], 2016. p. 2560–2567.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

SUN, C. et al. Revisiting unreasonable effectiveness of data in deep learning era. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2017. p. 843–852.

SUN, Y. et al. Svdnet for pedestrian retrieval. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2017. p. 3800–3808.

SUN, Z.; HE, S. Idiopathic interstitial pneumonias medical image detection using deep learning techniques: A survey. In: **Proceedings of the 2019 ACM Southeast Conference**. [S.l.: s.n.], 2019. p. 10–15.

SZEGEDY, C. et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2017. v. 31, n. 1.

____. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 1–9.

____. Rethinking the inception architecture for computer vision. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 2818–2826.

TAN, M.; LE, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. **arXiv preprint arXiv:1905.11946**, 2019.

TAN, M.; PANG, R.; LE, Q. V. Efficientdet: Scalable and efficient object detection. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2020. p. 10781–10790.

TOMPSON, J. et al. Efficient object localization using convolutional networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 648–656.

TOSHEV, A.; SZEGEDY, C. Deeppose: Human pose estimation via deep neural networks. **2014 IEEE Conference on Computer Vision and Pattern Recognition**, p. 1653–1660, 2014.

TOUVRON, H. et al. Fixing the train-test resolution discrepancy. **arXiv preprint arXiv:1906.06423**, 2019.

VERMA, V. et al. Manifold mixup: Better representations by interpolating hidden states. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2019. p. 6438–6447.

WAN, L. et al. Regularization of neural networks using dropconnect. In: PMLR. **International conference on machine learning**. [S.l.], 2013. p. 1058–1066.

WANG, S.; MANNING, C. Fast dropout training. In: **international conference on machine learning**. [S.l.: s.n.], 2013. p. 118–126.

WANG, X. et al. Esrgan: Enhanced super-resolution generative adversarial networks. In: **Proceedings of the European Conference on Computer Vision (ECCV) Workshops**. [S.l.: s.n.], 2018. p. 0–0.

WANG, Z.; SIMONCELLI, E. P.; BOVIK, A. C. Multiscale structural similarity for image quality assessment. In: IEEE. **The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003**. [S.l.], 2003. v. 2, p. 1398–1402.

WEI, H. et al. Combating noisy labels by agreement: A joint training method with co-regularization. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020.

WOLF, T. et al. Transformers: State-of-the-art natural language processing. In: **Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations**. [S.l.: s.n.], 2020. p. 38–45.

XIE, S. et al. Aggregated residual transformations for deep neural networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 1492–1500.

XU, Y. et al. Towards understanding label smoothing. **arXiv preprint arXiv:2006.11653**, 2020.

YAMADA, Y. et al. Shakedrop regularization for deep residual learning. **IEEE Access**, IEEE, v. 7, p. 186126–186136, 2019.

YANG, T.; ZHU, S.; CHEN, C. Gradaug: A new regularization method for deep neural networks. **arXiv preprint arXiv:2006.07989**, 2020.

YOO, J.; AHN, N.; SOHN, K.-A. Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2020. p. 8375–8384.

YUN, S. et al. Cutmix: Regularization strategy to train strong classifiers with localizable features. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. [S.l.: s.n.], 2019. p. 6023–6032.

ZAGORUYKO, S.; KOMODAKIS, N. Wide residual networks. In: WILSON, E. R. H. R. C.; SMITH, W. A. P. (Ed.). **Proceedings of the British Machine Vision Conference (BMVC)**. BMVA Press, 2016. p. 87.1–87.12. ISBN 1-901725-59-6. Disponível em: <https://dx.doi.org/10.5244/C.30.87>.

ZHANG, H. et al. mixup: Beyond empirical risk minimization. **arXiv preprint arXiv:1710.09412**, 2017.

____. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2017. p. 5907–5915.

ZHANG, K. et al. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. **IEEE transactions on image processing**, IEEE, v. 26, n. 7, p. 3142–3155, 2017.

ZHANG, Y. et al. Residual dense network for image super-resolution. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018. p. 2472–2481.

____. Residual dense network for image restoration. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, 2020.

ZHENG, L.; YANG, Y.; HAUPTMANN, A. G. Person re-identification: Past, present and future. **arXiv preprint arXiv:1610.02984**, 2016.

ZHONG, Z. et al. Random erasing data augmentation. In: **AAAI**. [S.l.: s.n.], 2020. p. 13001–13008.

ZHU, H.; ZHAO, X. Targetdrop: A targeted regularization method for convolutional neural networks. **arXiv preprint arXiv:2010.10716**, 2020.

ZOPH, B. et al. Learning transferable architectures for scalable image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018. p. 8697–8710.

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Relatório de Defesa de Tese
Candidato: Claudio Filipi Gonçalves dos Santos

Aos 24/06/2022, às 14:00, realizou-se na Universidade Federal de São Carlos, nas formas e termos do Regimento Interno do Programa de Pós-Graduação em Ciência da Computação, a defesa de tese de doutorado sob o título: Avoiding Overfitting: New Algorithms to Improve Generalization in Convolutional Neural Networks, apresentada pelo candidato Claudio Filipi Gonçalves dos Santos. Ao final dos trabalhos, a banca examinadora reuniu-se em sessão reservada para o julgamento, tendo os membros chegado ao seguinte resultado:

Participantes da Banca	**Função**	**Instituição**	**Resultado**	**Resultado Final**
Prof. Dr. João Paulo Papa	Presidente	UFSCar	______	______
Prof. Dr. Ricardo Cerri	Titular	UFSCar	______	
Prof. Dr. Diego Furtado Silva	Titular	UFSCar	______	
Prof. Dr. Joao Roberto Bertini Junior	Titular	UNICAMP	______	
Profa. Dra. Renata de Paris	Titular	IE	______	

Parecer da Comissão Julgadora*:

Encerrada a sessão reservada, o presidente informou ao público presente o resultado. Nada mais havendo a tratar, a sessão foi encerrada e, para constar, eu, Ivan Rogério da Silva, representante do Programa de Pós-Graduação em Ciência da Computação, lavrei o presente relatório, assinado por mim e pelos membros da banca examinadora.

João Paulo Papa	
Prof. Dr. João Paulo Papa	Representante do PPG: Ivan Rogério da Silva
Prof. Dr. Ricardo Cerri	Prof. Dr. Diego Furtado Silva
Prof. Dr. Joao Roberto Bertini Junior	Profa. Dra. Renata de Paris

Certifico que a defesa realizou-se com a participação à distância do(s) membro(s) Ricardo Cerri, Diego Furtado Silva, Joao Roberto Bertini Junior, Renata de Paris e, depois das arguições e deliberações realizadas, o(s) participante(s) à distância está(ao) de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.

João Paulo Papa

Prof. Dr. João Paulo Papa

( ) Não houve alteração no título ( ) Houve alteração no título. O novo título passa a ser:

Observações:
a) Se o candidato for reprovado por algum dos membros, o preenchimento do parecer é obrigatório.
b) Para gozar dos direitos do título de Mestre ou Doutor em Ciência da Computação, o candidato ainda precisa ter sua dissertação ou tese homologada pelo Conselho de Pós-Graduação da UFSCar.

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Parecer indicação Prêmio CAPES de Tese

A banca considera que a tese sob o título: "Avoiding Overfitting: New Algorithms to Improve Generalization in Convolutional Neural Networks", apresentada pelo(a) Candidato(a) Claudio Filipi Gonçalves dos Santos possui grau de excelência para ser indicada ao Prêmio CAPES de Tese?

( ) Sim ( ) Não

Justificativas

Prof(a). Dr(a). João Paulo Papa
(presidente)

Prof(a). Dr(a). Ricardo Cerri

Prof(a). Dr(a). Diego Furtado Silva

Prof(a). Dr(a). João Roberto Bertini Júnior

Prof(a). Dr(a). Renata de Paris

Certifico que a defesa realizou-se com a participação à distância dos membros **Ricardo Cerri, Diego Furtado Silva, João Roberto Bertini Júnior** e **Renata de Paris** e depois das arguições e deliberações realizadas, os participantes à distância estão de acordo com o conteúdo do parecer da banca examinadora redigido neste parecer.

Prof(a). Dr(a). João Paulo Papa
Presidente da Comissão Examinadora
(UFSCar)