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

CLAUDIO FILIPI GONÇALVES DOS SANTOS, Federal Institute of São Carlos - UFSCar, Brazil and Eldorado's Institute of Technology, Brazil
JOÃO PAULO PAPA, São Paulo State University - UNESP, Brazil

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 past 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," 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 referred here have their code available in public repositories or they have been directly implemented in some framework, such as TensorFlow or Torch.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

Additional Key Words and Phrases: Regularization, convolutional neural networks

ACM Reference format:
Claudio Filipi Gonçalves dos Santos and João Paulo Papa. 2022. Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks. *ACM Comput. Surv.* 54, 10s, Article 213 (September 2022), 25 pages.
https://doi.org/10.1145/3510413

1 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

The authors are grateful to the São Paulo Research Foundation through grants 2014/12236-1, 2017/25908-6, and 2019/07665-4, as well as the Brazilian National Council for Scientific, Technological Development grants 307066/2017-7 and 427968/2018-6 and Eldorado Research Institute.
Authors' addresses: C. F. G. dos Santos, Federal Institute of São Carlos - UFSCar, Rod. Washington Luiz, 235, São Carlos, São Paulo, Brazil, Eldorado's Institute of Technology, Av. Alan Turing, 275, Campinas, São Paulo, Brazil; email: cfsantos@ufscar.br; J. P. Papa, São Paulo State University - UNESP, Av. Eng. Luís Edmundo Carrijo Coube, 14-01, Bauru, São Paulo, Brazil; email: joao.papa@unesp.br.

0360-0300/2022/09-ART213
https://doi.org/10.1145/3510413

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 depth 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 [54], a neural network used for image classification, is also used on the EfficientDet [55] 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.

1.1 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, then 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 that, 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 Reference [3], 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, such as 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 dataset: If the rotation is too several, then an input "6" may be transformed into a "9," leading the model to learn wrong information.

1.2 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 [7] and MaxDropout [44] original codes show they do not execute anything during the inference, but the BatchNormalization [25] executes its algorithm in deducing the test set.

1.3 Scope of This Work

This study focuses on the most recent regularization techniques for CNNs. Other studies [36, 46] focus on older and more general regularization methods. Here, we consider three main points:

- *Recently developed*: Besides Dropout [49], 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.

Table 1. Summarization of the Approaches Considered in the Survey

Reference	Short Name	Description	Where
[18]	Bag of Tricks	Combines several regularizers to show how it improves CNN	Input
[22]	Batch Augment	Increases the size of the mini-batch	Input
[57]	FixRes	Performs train and test with different image sizes	Input
[7]	Cutout	Removes part of the image	Input
[67]	CutMix	Replaces part of the image using other parts of other images	Input / Label
[75]	RandomErasing	Replaces part of the image by noise or paint the region	Input
[69]	Mixup	Mixes two images from different classes	Input/Label
[5]	AutoAugment	Learns how to provide better data augmentation based on information from the training dataset	Input
[32]	Fast AutoAugment	Reduces the training time of the agent from [5]	Input
[6]	RandAugment	Learns augmentation policies during training	Input
[21]	PBA	Population-based algorithm for data augmentation	Input
[66]	CutBlur	Replaces regions from high-resolution images with low resolution pieces	Input / Label
[49]	Dropout	Drops random neurons	Internal
[44]	MaxDropout	Drops neurons based on their activation	Internal
[65]	GradAug	Trains sub-networks from the original CNN	Internal
[33]	Local Drop	Dropout and DropBlock based on the Radamacher complexity	Internal
[9]	Shake-Shake	Gives different weights to each branch of the residual connection	Internal
[64]	ShakeDrop	Improves Shake-Shake by generalizing to other models	Internal
[58]	Manifold Mixup	Act like Mixup, however, in the middle layers of a CNN	Internal / Label
[10]	DropBlock	Drops entire regions from a tensor	Internal
[38]	AutoDrop	Learns drop pattern	Internal
[53]	Label Smoothing	Replaces one-hot encoding vectors to smoothed labels	Label
[63]	TSLA	Two-stage algorithm for label smoothing	Label
[31]	SLS	Quantifies label smoothing based on feature space	Label
[61]	JoCoR	Co-relates labels for label smoothing	Label

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.

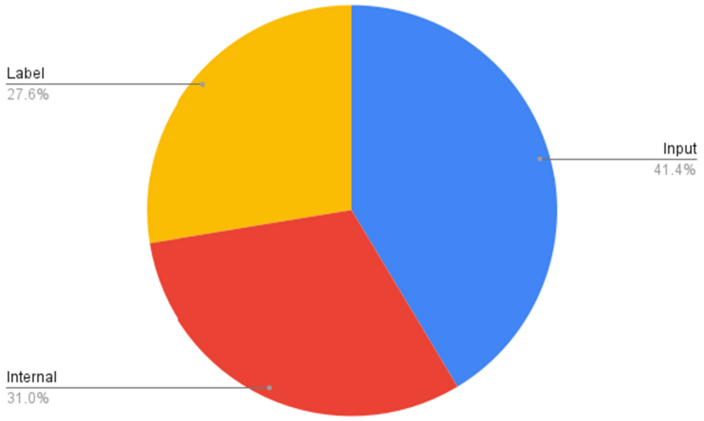

Fig. 1. Percentage of the regularization works surveyed in the manuscript.

1.4 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 [15, 16, 24, 54], object detection [11, 40], and image reconstruction [71–73]. 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 [36] 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 [59], and Bagging [1]. 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 [28]. Dropout [49] 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 published as recently as 2021 [33, 38]. The following subsections present more insights and statistical information about the works surveyed in the manuscript.

1.5 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 [49] contributed considerably to advances in this research area. Several works [33, 38, 44] are mainly based on Dropout, while some of them [9, 64] are new approaches.

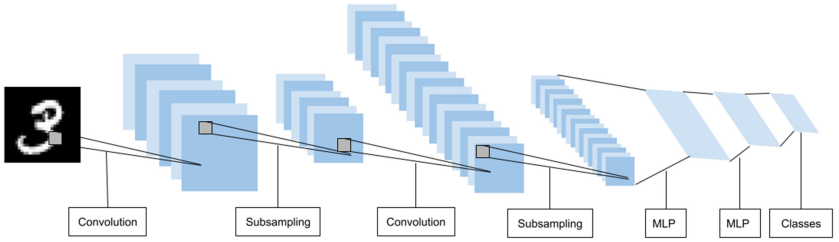

Fig. 2. The LeNet-5 structure. Inspired in the picture from Reference [29].

1.6 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 [53, 63]. 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 [53] 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 [3] and Mixup [2] 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 CONVOLUTIONAL NEURAL NETWORKS

Neural networks have been used since the 1950s when the first neuron emulation, called Perceptron [41], was developed. However, it can primarily address linearly separable feature spaces. However, in the 1980s, the development of the backpropagation algorithm [42] 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 nonlinear 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 [29]. 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.

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 [27] achieved first place in the ImageNet

classification challenge, overcoming the runner-up result by more than 10%. It is an eight-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 [47], 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 [51–53], 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 [16]. 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 (NAS) [77] developed a new way to find better CNN architectures. Using an agent trained by the Q-Learning technique [35], 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 [34] 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 [24], ending up in the EfficientNet family [54]. 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, CNNs can be used in several other tasks. One interesting problem is object detection in natural scenes. The R-CNN [11], 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 (YOLO)** [40] 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 [72] and image denoising [73] purposes. Another significant development is the DnCNN [71], 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 [12]; 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 [70] shows a very nice result, being able to change the style completely without losing relevant information. Another work with good results is the ERSGAN [60], 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.

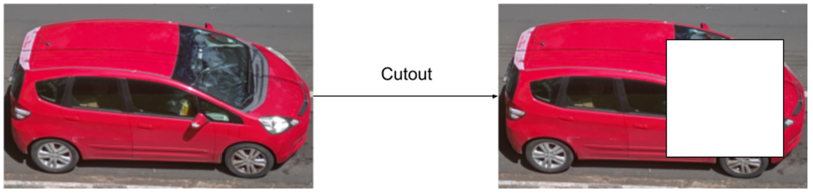

Fig. 3. How Cutout works. Extracted from Reference [7].

3 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 [29], basic data augmentation, such as flipping and noise adding, has proven it can help the trained model generalize better.

3.1 Cutout

One straightforward but powerful technique to perform data augmentation is the well-known Cutout [7]. During training, it randomly removes regions of the image before feeding the neural network. In Reference [7], 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 [4] the cut size for the best result was 32×32. Figure 3 shows how Cutout works.

3.2 RandomErasing

RandomErasing [75] 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 [8] dataset for object detection, and three different CNN architectures for person re-identification (IDE [74], TriNet [19], and SVDNet [50]). For the classification task, four different architectures were used for evaluation purposes: ResNet [16], ResNet with pre-activation [17], Wide Residual Networks [68], and ResNeXt [62], 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 [8]. Figure 4 shows how RandomErasing works.

3.3 AutoAugment

AutoAugment [5] tries to find out what transformations over a given dataset 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

Fig. 4. How RandomErasing works. Extracted from Reference [75].

policies are then fed into a "child" model, which is a CNN trained with part of the training dataset. 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 dataset. 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 [32] aimed at overcoming such a bottleneck with a new algorithm, reducing the time related to the search procedure significantly, besides producing similar results.

3.4 PBA

Population-Based Augmentation (PBA) [21] not only showed a novel augmentation algorithm but demonstrated schedule policies instead of fixed policies, which improves the results from the previous studies [5, 32]. At every three 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 dataset, PBA increases it by only five hours.

3.5 RandAugment

As mentioned before, a huge bottleneck for the methods that 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 [6] uses the 14 most common policies found on previous works [5, 21, 32] 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 [5, 21, 32] but improves the outcomes significantly.

3.6 Mixup

One possibility for training CNN concerns mixing two images from the training dataset and forcing the model to determine which class this mixture belongs reliably. However, it is not widespread

Fig. 5. Two different examples using Mixup. Extracted from Reference [69].

Fig. 6. How CutMix works. Extracted from Reference [67].

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.

3.7 CutMix

Another strategy to mix inputs and labels to improve the results is the CutMix [67]. 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, then 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 [45]), 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.

3.8 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 Auto Augment [5, 32], Cutout [7], and RandomErasing [75], 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 [66], 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.

3.9 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 [22] 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 setup 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.

3.10 FixRes

The image resolution may influence both the training step efficiency and the final classification accuracy. For instance, the research on EfficientNet [54] 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, then the test set's inference uses the exact resolution. The work proposed by Reference [57] 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.

3.11 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 [18] performs this investigation by combining several known regularization methods, such as Mixup [69], Label Smoothing [53], and Knowledge Destilation [20]. The ablation study shows that if some cleverness is applied, then the final result can be significantly improved. For instance, a MobileNet [23] 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 [49].

4 REGULARIZATION BASED ON INTERNAL STRUCTURE CHANGES

Regularization methods can work in different ways. In this article, 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 part 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.

4.1 Dropout and Variants

Dropout [49] 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.

4.2 MaxDropout

While Dropout [49] randomly removes the neurons in the training phase, Maxdropout [44] 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 [26] datasets, and it also outperforms Dropout on the WideResNet-28-10 model [68].

4.3 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 [45] show that, when correctly estimated, CNN highlights regions of interest around the object that has been classified. DropBlock [10] 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 [5, 56]. 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.

4.4 TargetDrop

Attention mechanism can be incorporated into a given regularizer so it can act in the appropriate region. For instance, the TargetDrop [76] 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 [45], demonstrates more consistency in the region that determines to which class a given input belongs.

4.5 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 [38] 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 datasets.

4.6 LocalDrop

The Rademacher complexity was used to redefine both Dropout and DropBlock [33]. 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.

4.7 Other Methods

In the past few years, the use of residual connections, first introduced in the well-known neural architecture ResNet [16], and their further improvements [62, 68] have achieved relevant results on several tasks. Later studies [17] 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.

4.8 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 [9] works on three-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 three-branches ResNet, making it hard to compare other methods directly.

4.9 ShakeDrop

One improvement to tackle the problems of Shake-shake is the ShakeDrop [64]. 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.

4.10 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 [58] acts like the Mixup [69], 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 sub-nets work well, then the final result can be regarded as a good one. Yang et al. [65] 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.

5 LABEL REGULARIZATION

Revisiting some information on Table 1, other methods use label smoothing as part of their regularization strategy. For instance, Mixup [69] averages the values of the labels depending on the interpolation between two different images. The same rule is applied for the Manifold Mixup

technique [58]; 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 [66]. 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 [7] 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 [75]. Methods that drop neurons during training, such as Dropout [49], could, for example, drop the values of the hot label by the same range of the total active neurons deactivated during training.

5.1 Label Smoothing

It is widespread in a general classification task to use the one-hot vector to encode the labels. Dating back from 2015 [52], label smoothing proposes a regularization technique in the label encoding process by changing the value on each position of the one-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 [52], 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.

5.2 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 dataset. The original work suggests that $\epsilon = 0.1$ is the excellent condition; however, the **Two-Stage Label Smoothing (TSLA)** [63] 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 dataset for 200 epochs, results suggest the best performance is achieved when label smoothing is used until the epoch 160.

5.3 SLS

Usually, it is not straightforward to define appropriate values for the label smoothness factor. **Structural Label Smoothing (SLS)** [31] 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 [43], it does not consider other neural network architectures. Even though some popular datasets were used for comparison purposes, e.g., CIFAR and SVHN, the work is limited to MobileNet-V2 only.

5.4 JoCor

This work proposes a new approach to avoid the influence of noisy labels on neural networks. JoCoR [61] trains two similar neural networks on the same dataset 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.

6 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.

6.1 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.

6.1.1 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 10 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.

6.1.2 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 1,240,000 224×224 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.

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 [27], 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 [16, 47, 52, 54].

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

6.1.3 SVHN. Used less frequently than the datasets mentioned above, but it is still a good baseline, the **Street View House Numbers (SVHN)** [37] is a set of images that comprises houses' numbers. Some characteristics come from the MNIST dataset [30], 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 use 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.

6.2 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.

6.2.1 ResNet. The oldest architecture used in most of the regularization works, the ResNet family [16] is still one of the most commonly used CNNs. It stands for the first neural 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 2 or 3 layers of a sequence of convolution and batch normalization [25], 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.

6.2.2 Wide Residual Network - WRN. Another widespread architecture in regularization works is the **Wide Residual Network (WRN)** [68]. 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

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

Table 2. Error (in %) for Each Classification Dataset Using Different Methods and Models

Method	Classification datasets. CIFAR-10	CIFAR-100	SVHN	ImageNet
ResNet18 [7]	4.72 ± 0.21	22.46 ± 0.31	-	-
+ Cutout [7]	3.99 ± 0.13	21.96 ± 0.24	-	-
+ RandomErasing [75]	4.31 ± 0.07	24.03 ± 0.19	-	-
+ MaxDropout [44]	4.66 ± 0.14	21.93 ± 0.07	-	-
+ MaxDropout + Cutout [44]	3.76 ± 0.08	21.82 ± 0.13	-	-
+ Mixup [69]	4.2	21.1	-	-
+ MM [58]	2.95 ± 0.04	20.34 ± 0.52	-	-
+ TSLA [63]	-	21.45 ± 0.28	-	-
+ TargetDrop [76]	4.41	21.37	-	-
+ TargetDrop + Cutout [76]	3.67	21.25	-	-
+ LocalDrop [33]	4.3	22.2	-	20.2
WRN [68]	4.00	19.25	-	21.9
+ Dropout [68]	3.89	18.85	1.60 ± 0.05	-
+ MaxDropout [44]	3.84	18.81	-	-
+ TargetDrop [76]	3.68	-	-	-
+ GradAug [65]		16.02	-	-
+ Dropout + Cutout [7]	3.08 ± 0.16	18.41 ± 0.27	1.30 ± 0.03	-
+ Dropout + PBA [21]	2.58 ± 0.06	16.73 ± 0.15	1.18 ± 0.02	-
+ Dropout + RE [75]	3.08 ± 0.05	17.73 ± 0.15	-	-
+ Dropout + BA + Cutout [22]	2.85	19.87	-	-
+ ShakeDrop [64]	4.37	19.47	-	-
+ Dropout + RE [75]	3.08 ± 0.05	17.73 ± 0.15	-	-
+ Dropout + Mixup [69]	2.7	17.5	-	-
+ Dropout + MM [58]	2.55 ± 0.02	18.04 ± 0.17	-	-
+ Dropout + Fast AA [32]	2.7	17.3	1.1	-
+ Dropout + RA [6]	2.7	16.7	1.0	-
+ AutoDrop [38]	3.1		-	-
+ AutoDrop + RE [38]	2.1	-	-	-
ResNeXt [62]	3.58	17.31	-	19.1
+ RE [75]	3.24 ± 0.04	18.84 ± 0.18	-	-
+ FixRes [57]	-	-	-	13.6
+ ShakeDrop [64]	3.67	17.80	-	20.34
+ Shake-Shake [9]	3.08 ± 0.05	17.73 ± 0.15	-	-
+ Shake-Shake + Fast AA [32]	2.0	14.9	-	-
+ Shake-Shake + Cutout [7]	2.56 ± 0.07	15.20 ± 0.21	-	-
+ Shake-Shake + PBA [21]	2.03 ± 0.11	15.31 ± 0.28	1.13 ± 0.02	-
+ Shake-Shake + AA [5]	2.0 ± 0.1	14.30 ± 0.2	1.0	-
ResNet-50 [10]	-	-	-	23.49 ± 0.07
+ ShakeDrop [64]	-	25.26	-	-
+ Bag of Tricks [18]	-	-	-	21.67
+ GradAug [65]	-	-	-	20.33
+ LocalDrop [33]	5.3	26.2	-	21.1
+ Dropout [10]	-	-	-	23.20 ± 0.04
+ Cutout [10]	-	-	-	23.48 ± 0.07
+ AA [10]	-	-	-	22.4
+ BA [22]	-	-	-	23.14
+ Fast AA [32]	-	-	-	22.37
+ Mixup [69]	-	-	-	22.1
+ DropBlock [10]	-	-	-	21.65 ± 0.05
+ FixRes [57]	-	-	-	17.5
+ RA [6]	-	-	-	22.4
+ AutoDrop [38]	-	-	-	21.3
+ AutoDrop + RA [38]	-	-	-	19.7
PyramidNet [14]	3.48 ± 0.20	17.01 ± 0.39	1.0	- 19.2
+ GradAug [65]		13.76	-	20.94
+ ShakeDrop [64]	3.08	14.96	-	20.94
+ ShakeDrop + Cutout [5]	2.3	12.2	-	-
+ ShakeDrop + AA [5]	1.5 ± 0.1	10.7 ± 0.2	-	-
+ ShakeDrop + Fast AA [32]	1.8	11.9		20.94
+ ShakeDrop + RA [6]	1.5	-	-	15.0
+ ShakeDrop + PBA [21]	1.46 ± 0.07	10.94 ± 0.09	-	15.0

The following acronyms were used: MM = ManifoldMixup, PBA = Population Based Augmentation, RE = RandomErasing, BA = BatchAugmentation, AA = AutoAugment, Fast AA = Fast AutoAugment, and RA = RandAugment.

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.

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 [68]. Dropout regularization between the convolutional layers after the ReLU activation helps perturb the batch normalization operation, which prevents overfitting.

6.2.3 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 [68] heavily. ResNeXt [62] introduces the concept of cardinality to accomplish better results.

Since ResNet [16], 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, then a ResNeXT block with cardinality 32 divides the main branch in 32 streams of 1, 2, and 1 convolution processes, and then concatenates all units before adding the residual value. It looks like 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 [16, 68**?**], it showed better outcomes.

6.2.4 PyramidNet. The last most common neural network is the one that achieves the best general results among the four mentioned here. The PyramidNet [6] 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 it can be removed.

7 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 [54] is re-used in the Efficient-Det work [55]. 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 article.

7.1 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 [39]. 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 ShakeDrop 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.

7.2 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 [7, 44, 75] 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 [49]. Therefore, the primary condition to be cited in this survey is to have the source code available so it can be compared to other methods directly.

However, it might be crucial for different reasons to use more than one regularizer in the same evaluation. For instance, the Wide Residual Network [68], 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 [5] and the Fast AutoAugmentat [32] incorporate as one of their policies the Cutout [7]. Therefore, a new regularization technique should be able to work with another regularizer and improve the results when both are used together.

7.3 Use of Minor Architectures

As a general rule, regularization adds little overhead during training time (AutoAugmentation [5] 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 [23] 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 [13] should be dissected to know how a given regularization algorithm influences either training a quantized neural network and performing the quantization after training.

EfficientNet [54] 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 networkss' resolution and depth are designated as more important than width. It is possible to verify that in the TinyNet work [15]. 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.

7.4 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)** [48], 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 [54] or ResNet [16].

7.5 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 [40] 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 [73] and low-resolution images [72].

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 [66], thus highlighting the lack of works in this direction.

Table 3. Summarization of the Approaches Considered in the Survey and Their Respective Source Codes

Reference	Short Name	Source Code
[18]	Bag of Tricks	MXNet: https://github.com/dmlc/gluon-cv
[22]	Batch Augment	PyTorch: https://github.com/eladhoffer/convNet.pytorch
[57]	FixRes	PyTorch: https://github.com/facebookresearch/FixRes
[7]	Cutout	Pytorch: https://github.com/uoguelph-mlrg/Cutout
[67]	CutMix	PyTorch: https://github.com/clovaai/CutMix-PyTorch
[75]	RandomErasing	PyTorch: https://github.com/zhunzhong07/Random-Erasing
[69]	Mixup	PyTorch: https://github.com/facebookresearch/mixup-cifar10
[5]	AutoAugment	TensorFlow: https://github.com/tensorflow/tpu/blob/master/models/official/efficientnet/autoaugment.py
[32]	Fast AutoAugment	PyTorch: https://github.com/kakaobrain/fast-autoaugment
[6]	RandAugment	TensorFlow: https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet
[21]	PBA	TensorFlow: https://github.com/arcelien/pba
[66]	CutBlur	PyTorch: https://github.com/clovaai/cutblur
[49]	Dropout	PyTorch: https://pytorch.org/docs/stable/_modules/torch/nn/modules/dropout.html
[44]	MaxDropout	Pytorch: https://github.com/cfsantos/MaxDropout-torch/
[65]	GradAug	PyTorch: https://github.com/taoyang1122/GradAug
[33]	Local Drop	Available in the paper
[9]	Shake-Shake	Torch: https://github.com/xgastaldi/shake-shake
[64]	ShakeDrop	Torch: https://github.com/imenurok/ShakeDrop
[58]	Manifold Mixup	PyTorch: https://github.com/vikasverma1077/manifold_mixup
[10]	DropBlock	TensorFlow: https://github.com/tensorflow/tpu/tree/master/models/official/resnet
[38]	AutoDrop	TensorFlow: https://github.com/google-research/google-research/tree/master/auto_dropout
[53]	Label Smoothing	TensorFlow: https://github.com/tensorflow/models
[63]	TSLA	Available in the paper
[31]	SLS	Available in the paper
[61]	JoCoR	https://github.com/hongxin001/JoCoR

7.6 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.

8 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 résumé 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 article 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 feed-forwarded 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 datasets, 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.

REFERENCES

[1] Leo Breiman. 1996. Bagging predictors. *Mach. Learn.* 24, 2 (1996), 123–140.

[2] Luigi Carratino, Moustapha Cissé, Rodolphe Jenatton, and Jean-Philippe Vert. 2020. On Mixup Regularization. arXiv:cs.LG/2006.06049.

[3] Jacopo Cavazza, Pietro Morerio, Benjamin Haeffele, Connor Lane, Vittorio Murino, and Rene Vidal. 2018. Dropout as a low-rank regularizer for matrix factorization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. PMLR, 435–444.

[4] Adam Coates, Andrew Ng, and Honglak Lee. 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 215–223.

[5] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2018. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501* (2018).

[6] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 702–703.

[7] Terrance DeVries and Graham W. Taylor. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552* (2017).

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. 2012. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. Retrieved from http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[9] Xavier Gastaldi. 2017. Shake-shake regularization. *arXiv preprint arXiv:1705.07485* (2017).

[10] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. 2018. DropBlock: A regularization method for convolutional networks. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 10727–10737.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 580–587.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661* (2014).

[13] Yunhui Guo. 2018. A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752* (2018).
[14] Dongyoon Han, Jiwhan Kim, and Junmo Kim. 2017. Deep pyramidal residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5927–5935.
[15] Kai Han, Yunhe Wang, Qiulin Zhang, Wei Zhang, Chunjing Xu, and Tong Zhang. 2020. Model Rubik's cube: Twisting resolution, depth and width for TinyNets. *arXiv preprint arXiv:2010.14819* (2020).
[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.
[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European Conference on Computer Vision*. Springer, 630–645.
[18] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. 2019. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 558–567.
[19] Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737* (2017).
[20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
[21] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. 2019. Population based augmentation: Efficient learning of augmentation policy schedules. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2731–2741.
[22] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. 2019. Augment your batch: Better training with larger batches. *arXiv preprint arXiv:1901.09335* (2019).
[23] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).
[24] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7132–7141.
[25] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).
[26] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2009. CIFAR-10 and CIFAR-100 datasets. Retrieved from https://www.cs.toronto.edu/kriz/cifar.html.
[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* 25 (2012), 1097–1105.
[28] Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. 2019. Survey of dropout methods for deep neural networks. *arXiv preprint arXiv:1904.13310* (2019).
[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[30] Yann LeCun and Corinna Cortes. 2010. MNIST handwritten digit database. Retrieved from http://yann.lecun.com/exdb/mnist/.
[31] Weizhi Li, Gautam Dasarathy, and Visar Berisha. 2020. Regularization via structural label smoothing. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Silvia Chiappa and Roberto Calandra (Eds.), Vol. 108. PMLR, 1453–1463. Retrieved from https://proceedings.mlr.press/v108/li20e.html.
[32] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. 2019. Fast autoaugment. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 6665–6675.
[33] Ziqing Lu, Chang Xu, Bo Du, Takashi Ishida, Lefei Zhang, and Masashi Sugiyama. 2021. LocalDrop: A hybrid regularization for deep neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* (2021). DOI: 10.1109/TPAMI.2021.3061463
[34] Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley. 2020. Neural architecture search without training. *arXiv preprint arXiv:2006.04647* (2020).
[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.
[36] Reza Moradi, Reza Berangi, and Behrouz Minaei. 2020. A survey of regularization strategies for deep models. *Arti. Intell. Rev.* 53, 6 (2020), 3947–3986.
[37] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop*.

[38] Hieu Pham and Quoc V. Le. 2021. AutoDropout: Learning dropout patterns to regularize deep networks. *arXiv preprint arXiv:2101.01761* (2021).
[39] Hieu Pham, Qizhe Xie, Zihang Dai, and Quoc V. Le. 2020. Meta pseudo labels. *arXiv preprint arXiv:2003.10580* (2020).
[40] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 779–788.
[41] Frank Rosenblatt. 1958. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* 65, 6 (1958), 386.
[42] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533–536.
[43] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4510–4520.
[44] Claudio Filipi Goncalves do Santos, Danilo Colombo, Mateus Roder, and João Paulo Papa. 2020. MaxDropout: Deep neural network regularization based on maximum output values. In *Proceedings of 25th International Conference on Pattern Recognition*. IEEE Computer Society, 2671–2676.
[45] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*. 618–626.
[46] Connor Shorten and Taghi M. Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *J. Big Data* 6, 1 (2019), 60.
[47] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[48] Fabio Alexandre Spanhol, Luiz S. Oliveira, Caroline Petitjean, and Laurent Heutte. 2016. Breast cancer histopathological image classification using convolutional neural networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'16)*. IEEE, 2560–2567.
[49] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (2014), 1929–1958.
[50] Yifan Sun, Liang Zheng, Weijian Deng, and Shengjin Wang. 2017. SVDNet for pedestrian retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*. 3800–3808.
[51] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. 2017. Inception-v4, inception-ResNet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
[52] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1–9.
[53] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2818–2826.
[54] Mingxing Tan and Quoc V. Le. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946* (2019).
[55] Mingxing Tan, Ruoming Pang, and Quoc V. Le. 2020. EfficientDet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10781–10790.
[56] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. 2015. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 648–656.
[57] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. 2019. Fixing the train-test resolution discrepancy. *arXiv preprint arXiv:1906.06423* (2019).
[58] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *Proceedings of the International Conference on Machine Learning*. PMLR, 6438–6447.
[59] Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. 2013. Regularization of neural networks using DropConnect. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1058–1066.
[60] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. 2018. ESRGAN: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision Workshops(ECCV'18)*.
[61] Hongxin Wei, Lei Feng, Xiangyu Chen, and Bo An. 2020. 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'20)*.

[62] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1492–1500.
[63] Yi Xu, Yuanhong Xu, Qi Qian, Hao Li, and Rong Jin. 2020. Towards understanding label smoothing. *arXiv preprint arXiv:2006.11653* (2020).
[64] Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. 2019. Shakedrop regularization for deep residual learning. *IEEE Access* 7 (2019), 186126–186136.
[65] Taojiannan Yang, Sijie Zhu, and Chen Chen. 2020. GradAug: A new regularization method for deep neural networks. *arXiv preprint arXiv:2006.07989* (2020).
[66] Jaejun Yoo, Namhyuk Ahn, and Kyung-Ah Sohn. 2020. 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*. 8375–8384.
[67] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6023–6032.
[68] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, Edwin R. Hancock Richard C. Wilson, and William A. P. Smith (Eds.). BMVA Press. DOI: https://doi.org/10.5244/C.30.87
[69] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2017. Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).
[70] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. 2017. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 5907–5915.
[71] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. 2017. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans. Image Process.* 26, 7 (2017), 3142–3155.
[72] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. 2018. Residual dense network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2472–2481.
[73] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. 2020. Residual dense network for image restoration. *IEEE Trans. Pattern Anal. Mach. Intell.* 43, 7 (2020), 2480–2495.
[74] Liang Zheng, Yi Yang, and Alexander G. Hauptmann. 2016. Person re-identification: Past, present and future. *arXiv preprint arXiv:1610.02984* (2016).
[75] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2020. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 13001–13008.
[76] Hui Zhu and Xiaofang Zhao. 2020. TargetDrop: A targeted regularization method for convolutional neural networks. *arXiv preprint arXiv:2010.10716* (2020).
[77] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8697–8710.

Received May 2021; revised October 2021; accepted January 2022