Explorations of Regularizations on the Various Stages of Convolutional Neural Networks (CNNs)

Joonyoung Bae [1] **Muhammad Adil Fayyaz** [1] **Matthew Salaway** [1] **Jacob Waite** [1]

Abstract

Convolutional Neural Networks (CNNs) have demonstrated remarkable performance in image classification tasks, yet they are prone to overfitting, hindering their generalization to unseen data. This paper presents a survey and experimental investigation of regularization methods for CNNs, categorizing them into input, internal, and label techniques. Our empirical results have shown that the models with regularizations generally outperforms the ones without, and having combinations of regularization methods throughout the various stages of training stages achieve optimal result. Our simple baseline model with around 450k parameters could get similar accuracy as AlexNet which has 15 times more parameters with the right combinations of regularization methods, and it showed that the model performance can improve without increase in the complexity of the model.

1. Introduction

Neural networks, one of the most powerful tools in machine learning, excel in various tasks but are prone to overfitting, compromising their generalization ability by performing well on training data but poorly on unseen data.

Regularization techniques, essential for preventing overfitting and enhancing generalization, are integral to neural network training. In image recognition, a prevalent application of Neural Networks, Convolutional Neural Networks (CNN) are one of the most popular. These networks, particularly adept at processing image data with fewer parameters, benefit from a plethora of regulatization techniques, including those specifically designed for them.

We embarked on an exploration of how different regularization methods, categorized based on their application points within the network, influence CNN performance in image recognition. Our investigation, guided by a survey paper on regularization methods for CNNs, titled *Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks* (Santos & Papa, 2022), encompassed techniques targeting input data, internal network operations and label manipulation. Experimentation involved both individual techniques and their combinations to gauge their impact on CNN performance.

2. Methods

To develop a minimal baseline CNN, we opted for a custom architecture designed from the ground up only with Convolutional and MaxPool layers. By crafting our own architecture, we ensure to have a clear understanding of the model's behavior and providing flexibility for experimenting with different settings. The architecture layout is shown in A. Appendix Figure 10. Below we list the methods that we employed for regularization.

2.1. Input Methods

2.1.1. Cutout

The Cutout method (DeVries & Taylor, 2017) is a potent data augmentation technique, widely used in image classification tasks. It randomly selects square regions within images, masking them out to simulate occlusions or missing portions. This approach injects variability and noise into training data, prompting the model to focus on relevant features and develop more robust representations. By specifying parameters like block size and the number of cutouts per image, Cutout effectively regularizes training, counteracting overfitting and enhancing generalization. Moreover, it boosts model resilience by exposing it to partially occluded images during training, leading to improved performance on test data with similar variations.

2.1.2. Random Erasing

Random Erasing (Zhong et al., 2020) is another valuable data augmentation technique used in computer vision tasks, especially in image classification and object detection. Proposed as an enhancement to the Cutout method, Random Erasing operates by randomly selecting rectangular blocks within an image and replacing them with random pixel values or the mean pixel value of the dataset. This process effectively mimics occlusions or missing parts in the image, compelling the model to learn more discriminative features and improving its robustness to variations in the input data.

2.1.3. MIXUP

The Mixup data augmentation method (Zhang et al., 2018) revolutionizes traditional data augmentation in supervised learning, such as image classification. It blends pairs of training examples and labels to create new synthetic examples. Two randomly selected samples are combined by linearly interpolating their feature vectors and labels, resulting in a new augmented example with a mixed feature vector and label. Mathematically, if (x_i, y_i) and (x_j, y_j) are two training examples with labels, the mixed example $(\tilde{x}, \tilde{y})$ is generated as follows: $\tilde{x} = \lambda x_i + (1 - \lambda)x_j$ and $\tilde{y} = \lambda y_i + (1 - \lambda)y_j$, where λ is a random scalar value sampled from a Beta distribution with parameter α.

2.2. Internal Methods

2.2.1. DROPOUT

Dropout is an extremely common regularization method for neural networks. It consists of placing a "dropout layer" after one or more other non-dropout layers of the network (note that this means two or more dropout layers generally do not appear in succession). Each activation value of the previous layer that passes through a dropout layer is set to 0 with a probability of r. When this happens, the activation value is thought of as being "dropped" from the network, hence the name "dropout". r is essentially the "dropout rate" of that dropout layer, since on average a proportion r of the activation values of the previous layer will not make it to the next layer. It is important to note that the dropout rate r is a tuneable hyperparameter.

2.2.2. MAXDROP

The MaxDrop method enhances regularization in deep neural networks by selectively dropping out neurons with the highest activations across the channels or across the feature map, as introduced in (Park & Kwak, 2017). In our implementation, we adopt the process outlined in (do Santos et al., 2021), where a dropout rate, r, is uniformly selected between 0 and a predefined maximum rate. Then, the tensor of the layer is normalized using L2 normalization, and the maximum activation value, max, in the normalized tensor is identified. Any activation in the original tensor that exceeds $(1 - r) \times max$ in the normalized tensor is dropped. This approach encourages the network to rely less on dominant neurons, thus fostering a more robust feature learning.

2.2.3. DROPBLOCK

DropBlock (Ghiasi et al., 2018) is a variant of dropout that is tailored to work more effectively in CNNs. Similarly to Dropout, DropBlock consists of layers that we will call "DropBlock layers". There are two tuneable hyperparemeters in DropBlock; a dropout rate γ, and a "block_size".The idea behind DropBlock is as follows. Convolution layer outputs can be thought of as $w \times h$ "rectangles" of values (activation maps) stacked on top of each other. There are f such slices stacked on top of one another, where f is the number of filters in the convolution layer. A DropBlock layer randomly selects points in each slice at a rate of γ, and then considers all of the values around those selected points within a box of size "block_size". Note that only values that can have boxes of values around them of size "block_size" without going out of bounds of the slice are considered for being selected as center points of the boxes. Afterwards, points within all of the boxes centered on the selected values are set to 0 (including the center points themselves). This process repeats for each of the f value slices passed into the DropBlock layer before passing to the next layer.

DropBlock aims to enhance information suppression compared to regular dropout, especially in CNNs, where spatial correlation is common and thus information can make it through values in each slice surrounding dropped values. We investigated whether this specialized regularization method outperforms standard dropout in CNNs.

2.3. Label Methods

2.3.1. LABEL SMOOTHING (LS)

In a general image multiclass classification model using CNN, the last layer of the neural network is usually the softmax function, $p(k|x) = \frac{exp(z_k)}{\sum_{i=1}^{K} exp(z_i)}$, that converts the logits into probabilities. Then we train the model using SGD and backpropagation with cross entropy loss, $\ell = -\sum_{k=1}^{K} log(p(k|x))q(k|x)$, where $q(k|x)$ is the ground truth label of the datapoint x, that sums up to 1 and can be fractions.

In 2016, (Szegedy et al., 2016) proposed that training the model with hard labels, where $q(y|x) = 1$ and $q(k|x) = 0$ for all $k \neq y$, can cause overfitting. If the target value for the softmax function is 1, the logit corresponding to the true label will become large for the softmax value to converge to 1. Besides handling with adequate learning rate and other hyperparameters, the paper suggested that using a soft label $q'(k|x) = (1 - \alpha)\delta_{k,y} + \alpha/K$, where $\delta_{k,y}$ is the hard label or the Dirac delta function and K is the number of classes. So, now the label has smoothed out with a portion of the labels contributed to uniform distribution.

The gradient of the cross entropy is $\frac{\partial \ell}{\partial z_k} = p(k) - q(k)$ and with the new smoothed label $q'(k)$, model is now learning slower just like the regularization terms added in regressions.

In 2019, (Müller et al., 2019) empirically showed that it improves generalization and acts as an implicit model calibration, a scalar multiplication to the logits before the softmax

function (temperature scaling).

2.3.2. Two-Staged LAbel smoothing (TSLA)

In 2020, (Xu et al., 2020) analyzed the convergence behavior of the stochastic gradient descent with Label Smoothing and proposed Two-Stage LAbel smoothing (TSLA) that improved the performance even further.

The implementation of TSLA is simple, the model is trained with Label Smoothing for the early epochs and without it after. The hyperparameter, β, is the number of the epochs with LS. This simple mixture technique outperformed both models with and without LS in the above paper on complex architectures.

3. Results

Below we show the results we got on different regularization techniques.

3.1. Cutout

The Cutout method was tested on three different settings of length. The length being the side of the square block to remove from the image. Table 1 presents the results.

Length(l)	10	12	14
Max Accuracy	73.70	74.42	**74.65**

Table 1: Maximum Validation Accuracy

A variation of the Cutout method was also tested, in which the Cutout method was altered to remove regions from the image that were in the background instead of the foreground. The intuition behind this change was that the method should improve the regularization strength by removing areas in the background. This adjustment was made under the assumption that the primary features contributing to correct model classification reside in the foreground. The results are presented in Table 2:

Background Cutout	Yes	No
Max Accuracy	**75.94**	73.70

Table 2: Maximum Validation Accuracy with Length = 10

The observed results confirm our hypothesis that the background only Cutout would improve standard Cutout, having increased the validation accuracy scores by roughly 2%.

3.2. Random Erasing

The Random Erasing method was tested on three different settings of area ratio. The area ratio is the maximum area of the erased region relative to the total area of the image. We get the best results with $sh = 0.2$. Table 3 presents the results.

Aspect Ratio(sh)	0.2	0.3	0.4
Max Accuracy	**74.96**	74.80	74.65

Table 3: Maximum Validation Accuracy

3.3. Mixup

The Mixup method was tested on three different settings of lambda λ. λ is the control variable of the Beta distribution used to define the concentration of images in the Mixup method. Table 4 presents the results.

Lambda(λ)	0.2	0.4	0.6
Max Accuracy	77.85	**78.20**	77.99

Table 4: Maximum Validation Accuracy

Below we have a confusion matrix we got from the predictions of the Mixup method. The confusion between class-3 "birds" and class-5 "deers" labels may arise due to their shared characteristic of being "sharp-edged". Birds typically possess sharp-edged features like beaks, while deers have sharp-edged characteristics such as antlers. This similarity in physical attributes might lead to misclassifications in the predictions. We get similar confusion matrices for all of the methods.

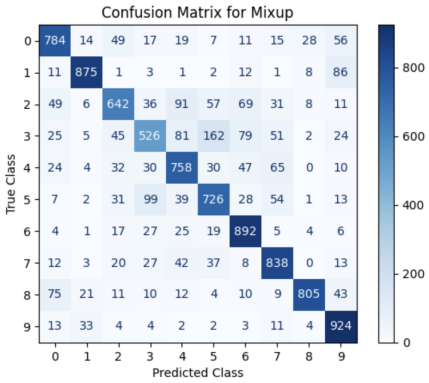

Figure 1: Confusion Matrix of the Mixup Method Predictions

3.4. Dropout

To test dropout regularization, 20 different variants of the base model were created. Each variant had a single dropout layer inserted at one of the five max pooling layers in the network. For each of those five locations, there were four networks with a dropout layer there. Each of the four networks at each location had a different dropout rate. The four dropout rates tested at each location were 0.01, 0.1, 0.2, and 0.5. Those four dropout rates were chosen since they give a rather good range of dropout rate values spanning from

very small at 0.01 to relatively large at 0.5. Each of the 20 networks was trained for 40 epochs using the CIFAR-10 dataset, however only the first 20 epochs were considered for this analysis to compare with the base model.

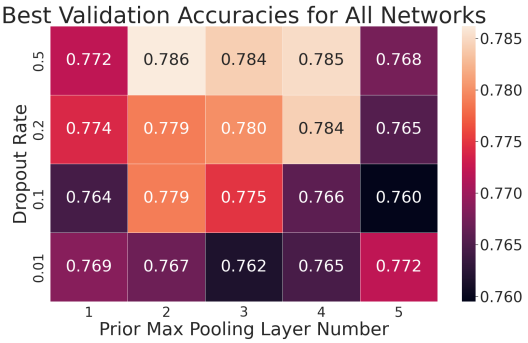

Figure 2: A heatmap showing the best validation accuracy achieved during training for all dropout models.

The best validation accuracy achieved was with the network that had a dropout layer after the second MaxPool layer and with a dropout rate of 0.5, having a validation accuracy of 78.62%. The best validation accuracies for the other models with a dropout rate of 0.5 with dropout layers after the third and fourth MaxPool layers had validation accuracies within roughly 0.2% and 0.1% respectively of the best validation accuracy, which is extremely close. We see generally in the heatmap that, with only one dropout layer in the network, it was better to place the layer not too close to the start or end of the network, but more towards the middle, and with a larger dropout rate close to 0.5.

3.5. MaxDrop

To analyze the MaxDrop regularization technique, we trained 38 different models with varying max dropout rates, number of MaxDrop layers, and placement of MaxDrop Layers. All models had 5 different locations where the MaxDrop layer could be placed, and each of these layers was placed after a Max Pool layer. 30 of the models were created to analyze the effects of layer placement and max dropout rate, keeping the number of MaxDrop layers constant at 2 per model.

Layer Placement	1 & 2	2 & 4	3 & 4	4 & 5
MaxDropout Rate	0.4	0.25	0.1	0.4
Validation Accuracy	0.7677	0.7671	0.7699	0.7681

Figure 3: Fine-tuned MaxDrop rate optimized for validation accuracy at different layer placements.

Thus, of the 30 models, the one with max validation accuracy had a max dropout rate of 0.1 at layers 3 and 4, which had a validation accuracy of 76.81%. Further, it appears the MaxDrop layers at the beginning and end of the CNN perform best with higher MaxDrop rates while those in the middle of the CNN performed best with relatively lower MaxDrop rates. The other 8 of the models were trained with all 5 MaxDrop layers, to test the performance of different max dropout rates. Our hypothesis was that the 5 MaxDrop layer model with the highest validation accuracy would have a similar pattern to the highest accuracy rates in the 2 Maxdrop layer models. Our hypothesis was correct as

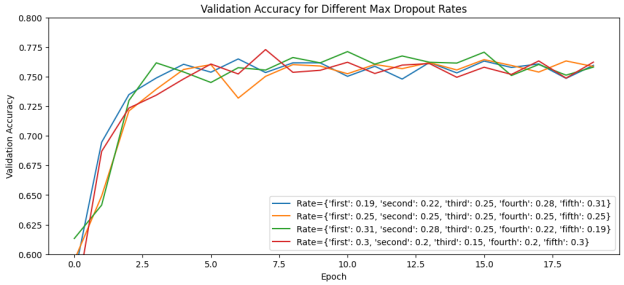

Figure 4: Validation Accuracy with 5 MaxDrop layers, each with a different set of rates

the highest performing model followed the trend found in the 2 MaxDrop layer experiment, where the beginning and ending layers have a higher MaxDrop rate than the middle layers. The top-performing model's validation accuracy was 77.29%, 3.5% higher compared to our baseline model.

3.6. DropBlock (DB)

We created 72 variants of the base model, each with a DB layer at one of the first five convolution layers. Block sizes of 3, 5, and 7 were tested after each convolution layer, except layers 4 and 5, which omitted a block size of 9 due to its larger size than the output slices. Each variant was tested with four different γ values (0.01, 0.1, 0.2, 0.5). Training was conducted for 40 epochs using CIFAR-10, but only the first 20 epochs were analyzed. The best performing model, achieving 78.49% validation accuracy, had a DB layer after the fourth convolution layer, with γ = 0.2, block size = 7. Figures 5, 6, 7 illustrate the highest validation accuracies among models varying in specific parameters.

Figure 5 takes averages over models with DB layers in the same location of the network. We see that generally the validation error increases slightly as the DB is added later in the network. However, we see a massive drop in the validation accuracy for the latest DB layer position. Perhaps this is because at that point the network has made the dimensions of the "slices" output by the convolution layer so small that each value in the slice contains a huge amount of information, and so dropping large blocks of that information loses information to the point of seriously harming the network's ability to learn.

Figure 6 takes averages over models with DropBlock lay-

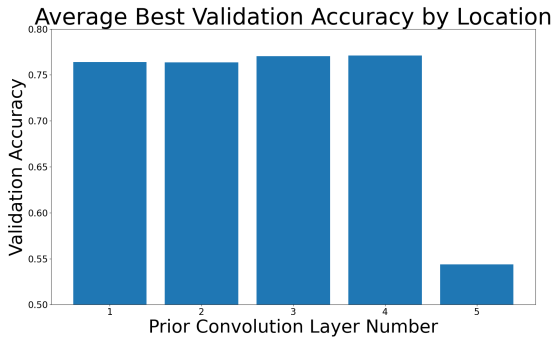

Figure 5: Average Best Val. Accuracy of DB at Different Layers

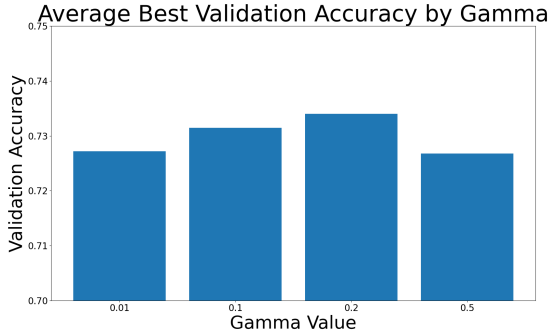

Figure 6: Average Best Val. Accuracy with DB of different γ

ers having the same γ. Again we see a generally positive correlation between γ and average best validation accuracy, but with a rather large drop at the end for $\gamma = 0.5$. This could again be a case of simply too much information being dropped from the network. Interestingly, while γ seems to be closely linked to the dropout rate of regular dropout, we saw earlier that the best dropout rate was 0.5. However, because γ essentially is a block dropout rate versus an individual value dropout rate, many more values are likely to be dropped when $\gamma = 0.5$ as opposed to when the dropout rate in dropout is 0.5. As such, it is likely that small changes γ have a greater effect then changes of the same amount in the dropout rate for dropout, and so great care should be taken to not set γ to be too large.

The final Figure 7 averages models with DropBlock layers of the same block size. The drop at block size 5 may be from dropping most of the information at 8x8 layer. Block size 7 would not fit at 8x8 layer most of the time unless initialized near center. The exact reason is unclear, but generally, smaller block sizes may preserve validation accuracy better, possibly because larger blocks risk blocking too much information, particularly in later layers where slices are smaller. The model with a DropBlock layer in the fourth convolution layer had the best validation accuracy, suggesting that placing a DropBlock layer later in the network is not necessarily detrimental, provided it's not too large.

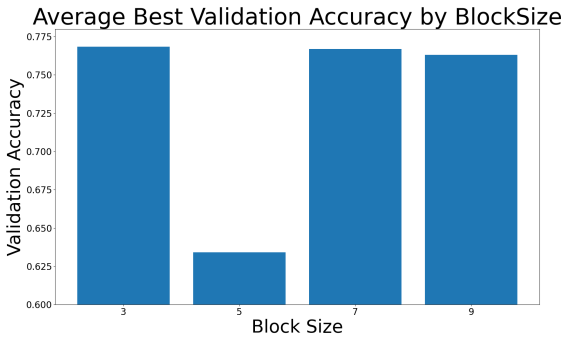

Figure 7: Best validation accuracies averaged over models with DropBlock layers with different block sizes.

3.7. Label Smoothing

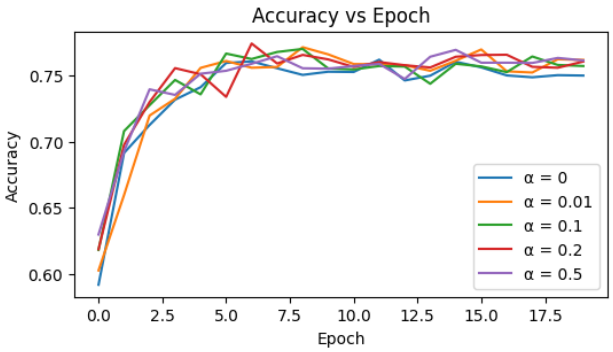

Figure 8: Validation Accuracies of LS with Various α Values

α	0	0.01	0.1	0.2	0.5
Max Accuracy	77.52	76.41	76.49	**78.05**	76.8

Table 5: Maximum Validation Accuracy

Figure 8 and Table 5 show that the best validation accuracy is achieved at $\alpha = 0.2$, showing that Label Smoothing helps the generalization. In the paper (Szegedy et al., 2016), Label Smoothing improved performances of multiple existing large-scale architectures as well.

3.8. TSLA

β	0	1	2	3	5
Max Accuracy	76.61	**76.92**	76.2	76.04	76.43

Table 6: Maximum Validation Accuracy

As shown in Figure 9 and Table 6, the best model was the one trained with 1 epoch of LS and the rest without LS, with α = 0.2 throughout. Even though TSLA underperformed with our baseline model, it was proven to be better at more complicated architectures as shown in (Xu et al., 2020) with additional advantage such as faster convergence than LS.

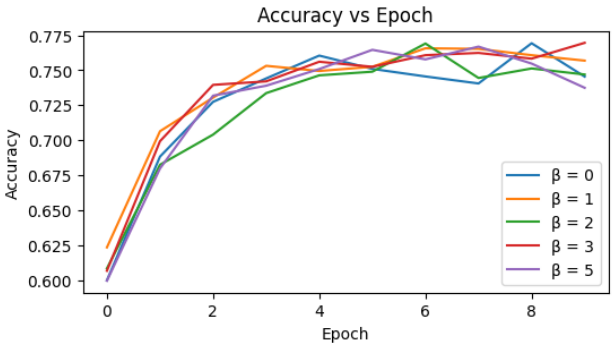

Figure 9: Validation Accuracy with Various β Values

3.9. Combinations and Comparisons

We also have explored if any combination of aforementioned regularization methods can outperform individual methods. We first picked the best regularization methods from each training stages, (Mixup, Dropout, and LS), and empirically tested every combinations of them. We further tested the combinations of more complicated and cutting-edge regularization methods: Mixup, MaxDrop, and TLSA.

	Accuracy	**Top 3 Accuracy**
No Regularization	75.44	93.93
Cutout (l = 10)	73.70	93.08
Random Erasing (sh = 0.2)	74.96	93.77
Mixup (λ = 0.2) **(1)**	77.85	94.16
Dropout**(2.1)**	78.62	85.06
MaxDrop **(2.2)**	77.29	94.69
DropBlock	78.49	83.81
LS (α = 0.2) **(3.1)**	77.41	94.21
TLSA(α=0.2, β=2) **(3.2)**	76.91	94.21
(1)+(2.1)	80.49	95.47
(2.1)+(3.1)	80.08	94.73
(1)+(3.1)	77.98	94.07
(1)+(2.1)+(3.1)	**80.96**	**95.12**
(1)+(2.2)	78.03	94.55
(2.2)+(3.2)	76.59	93.88
(1)+(3.2)	77.93	94.38
(1)+(2.2)+(3.2)	77.80	94.11

Table 7: Performance after Training Baseline Model on CIFAR-10

Shown in Table 7, the combination of Mixup, Dropout, and LS showed the best result in both the validation accuracy and top 3 accuracy. This empirical result manifested spreading the regularization throughout the training process can yield the best result, while it took longer to converge. Fine-tuning of the hyperparameters of each method may further improve the performance. Overall trend showed combining the regularization methods outperforms each of the combined methods individually. For comparison, Alex-net with 15 times more parameters than our baseline model has around 70% accuracy without any regularizations (Zhang, 2021). Even with Dropout added to AlexNet, the accuracy was around 83.47% (Kumar, 2020), which is not far from our result. This proved that having the combinations of regularizations throughout the model training can improve the performance without increasing the complexity of the architecture.

4. Process

CIFAR-100 dataset was used first, but was later changed to CIFAR-10 dataset due to simplicity of our baseline model and large number of classes for visualization and analysis.

4.1. Input/Data Augmentation Methods

We optimized data augmentation methods for the baseline model. With Random Erasing, we added a color spectrum to remove patches instead of averaging color pixels since it improved the model. Mixup required one-hot encoding labels for proper classification; otherwise, it would have become a regression task with incomparable performance.

4.2. MaxDrop

For MaxDrop, we analyzed models with varying MaxDrop Layers and MaxDropout rates. Future work could build off our findings by dynamically changing the MaxDropout rate during training while keeping the optimal trend of rates between layers.

4.3. Label Smoothing

We have tried to test if the reason behind using uniform distribution as the prior distribution is either because we assume the train dataset is of uniform distribution or it is just a way to smooth the learning process. Tested with a train dataset with skewed distribution using LS with Uniform Distribution, the model performed well, proving that the Uniform Distribution is to smooth the learning.

5. Contributions

CIFAR-100 (cif, b) and CIFAR-10 (cif, a) publicly sourced datasets were used (Pre-processed).

We implemented 8 different regularization methods grouped by Input, Internal and Output methods on a custom baseline CNN architecture. Tensorflow and its libraries were used to build and train our baseline model and regularization methods such as LS. We have programmed our own codes for some methods: Cutout, Random Erasing, Mixup, TSLA, and Combinations of Regularization of Methods. Our work can be found at: Github.

References

Cifar-10 (canadian institute for advanced research). cs.toronto.edu/~kriz/cifar.html, a.

Cifar-100 (canadian institute for advanced research). cs.toronto.edu/~kriz/cifar.html, b.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

do Santos, C. F. G., Colombo, D., Roder, M., and Papa, J. P. Maxdropout: deep neural network regularization based on maximum output values. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2671–2676. IEEE, 2021.

Ghiasi, G., Lin, T.-Y., and Le, Q. V. Dropblock: A regularization method for convolutional networks, 2018.

Kumar, V. Alexnet in pytorch cifar10 class. kaggle.com/code/drvaibhavkumar/alexnet-in-pytorch-cifar10-clas-83-test-accuracy, 2020.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

Park, S. and Kwak, N. Analysis on the dropout effect in convolutional neural networks. In *Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part II 13*, pp. 189–204. Springer, 2017.

Santos, C. F. G. D. and Papa, J. P. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. *ACM Computing Surveys*, 54(10s):1–25, January 2022. ISSN 1557-7341. doi: 10.1145/3510413. URL http://dx.doi.org/10.1145/3510413.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Xu, Y., Xu, Y., Qian, Q., Li, H., and Jin, R. Towards understanding label smoothing. *arXiv preprint arXiv:2006.11653*, 2020.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2018.

Zhang, X. The alexnet, lenet-5 and vgg net applied to cifar-10. In *2021 2nd International Conference on Big Data Artificial Intelligence Software Engineering (ICBASE)*, pp. 414–419, 2021. doi: 10.1109/ICBASE53849.2021.00083.

Zhong, Z., Zheng, L., Kang, G., and Li, S. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 13001–13008, 2020.

A. Appendix

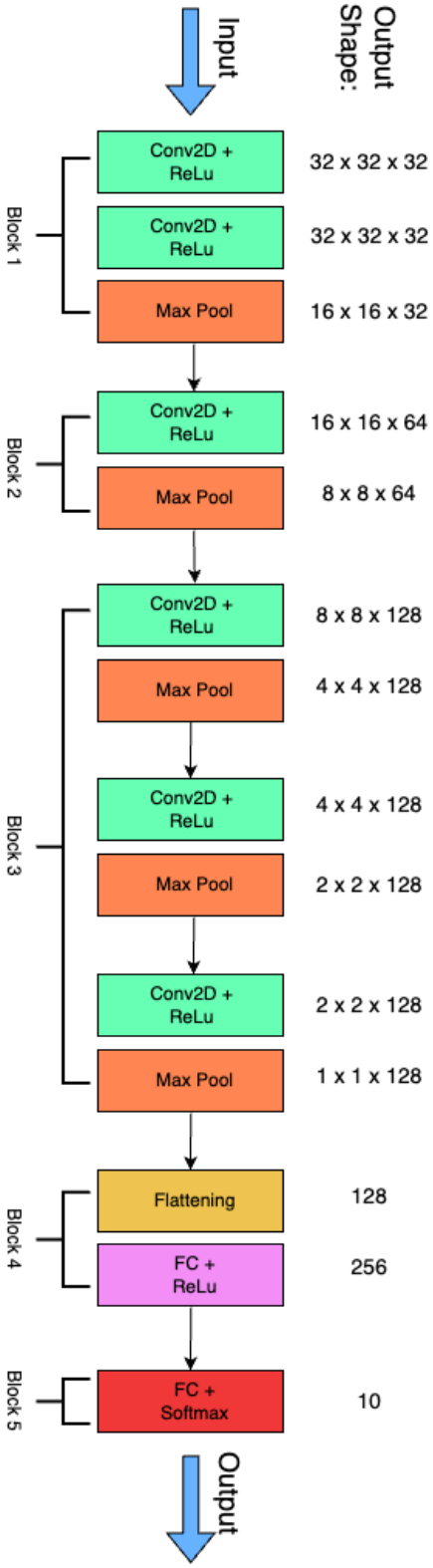

Figure 10: Baseline CNN Architecture