Published by
NIGERIAN SOCIETY OF PHYSICAL SCIENCES
Available online @ https://journal.nsps.org.ng/index.php/jnsps

J. Nig. Soc. Phys. Sci. 6 (2024) 1911

Journal of the Nigerian Society of Physical Sciences

Regularization effects in deep learning architecture

Muhammad Dahiru Liman[a,*], Salamatu Osanga Ibrahim[a], Esther Samuel Alu[b], Sa'adu Zakariya[a]

[a]*Department of Computer Science, Federal University of Lafia, Nasarawa, Nigeria*
[b]*Department of Computer Science, Nasarawa State University Keffi, Nasarawa, Nigeria*

Abstract

This research examines the impact of three widely utilized regularization approaches – data augmentation, weight decay, and dropout –on mitigating overfitting, as well as various amalgamations of these methods. Employing a Convolutional Neural Network (CNN), the study assesses the performance of these strategies using two distinct datasets: a flower dataset and the CIFAR-10 dataset. The findings reveal that dropout outperforms weight decay and augmentation on both datasets. Additionally, a hybrid of dropout and augmentation surpasses other method combinations in effectiveness. Significantly, integrating weight decay with dropout and augmentation yields the best performance among all tested method blends. Analyses were conducted in relation to dataset size and convergence time (measured in epochs). Dropout consistently showed superior performance across all dataset sizes, while the combination of dropout and augmentation was the most effective across all sizes, and the triad of weight decay, dropout, and augmentation excelled over other combinations. The epoch-based analysis indicated that the effectiveness of certain techniques scaled with dataset size, with varying results.

DOI:10.46481/jnsps.2024.1911

Keywords: Deep Learning, Regularization, Overfitting, Weight Decay, Augmentation

Article History :
Received: 19 November 2023
Received in revised form: 25 January 2024
Accepted for publication: 20 February 2024
Published: 11 March 2024

Communicated by: O. Akande

1. Introduction

Various deep-learning architectures have been utilized to build models for different problem domains. These architectures encompass several techniques. Deep belief networks [1], CNNs [2], and Recurrent Neural Networks (RNNs) [3, 4]. According to Moradi *et al.* [5], these models can learn complex representations for a wide range of tasks.

In the present study, we focused on the use of CNN architecture. The CNN architecture consists of a convolutional layer for feature extraction and learning filters, pooling layers for downsampling, and a fully connected layer for classification. Nielsen [6] states that the key principles underlying the design of CNNs are local receptive field, weight sharing, and pooling. Model learning requires sufficient data for the model to learn. The extent to which the model learns effectively is heavily influenced by the volume of datasets used [7]. Insufficient data can lead to overfitting.

In this study, we employed two datasets (CIFAR10 and Flower) of varying sizes to investigate the performance of these techniques. By utilizing these two datasets, we aim to assess the transferability and generalization capabilities of the techniques across datasets of different sizes. An ideal model should be able to generalize well to previously unseen data. Nielsen [6] explains that overfitting commonly occurs when the training data is insufficient. Even with a substantial amount of training data,

*Corresponding Author Tel.: +234-803-899-2197;
Email address: mlimand76@gmail.com (Muhammad Dahiru Liman)

deep models are still prone to overfitting, as noted in [8].

Overfitting poses a significant challenge for model developers as they strive to create models that can be generalized. One approach to preventing overfitting is regularization, as discussed in Moradi *et al.* [5] and Nielsen [6]. Various regularization techniques have been developed over time, including Dropout [9], adversarial training [10], early stopping [11, 12], ensemble methods [13], noise injection [4], batch normalization [14], data augmentation (transformed data) [15], L1 [16], and Weight Decay [17], among others.

Weight decay (L2) is a technique that encourages a network to favor small weights, and it only allows larger weights if they significantly improve the initial loss function [6]. The core idea behind dropout is to introduce independence among the neurons or nodes within a neural network. During training, each node has a probability, denoted as p, of being deactivated (turned off), whereas there is a complementary probability of 1-p that remains active. This selective deactivation helps prevent co-adaptation between neurons [9]. The ability of a model to generalize effectively is typically associated with an ample amount of training data.

Nevertheless, collecting a sufficient quantity of data often proves to be a challenging, time-consuming, and costly endeavor, as mentioned in Nielsen [6]. One straightforward approach to address the issue of limited data is to incorporate transformed data into the existing dataset. Transformed data can be generated by applying various transformations such as affinity, rotation, translation, and cropping to the available data.

The availability of these regularization methods is advantageous for programmers. However, the abundance of options can make it challenging for developers to determine which technique to employ, as pointed out in Moradi *et al.* [5]. Hence, this study aims to investigate the effects of data augmentation, weight decay, and dropout of three popular regularization strategies [6]. We also explore different combinations of these three methods. By examining and comparing the impacts of these strategies, developers can make more informed decisions regarding their utilization, based on the model's architecture and dataset size.

2. Related Work

2.1. Analysis of related work

Overfitting has emerged as a significant concern, and numerous methods have been developed to address this problem. However, developers seeking to address overfitting face the challenge of determining the most effective technique or a combination of techniques for their specific dataset size. With the increasing number of available techniques, it has become crucial for developers to compare and evaluate different options. The following are several comparisons conducted.

In their analysis using the MNIST dataset, Srivastava *et al.* [18] introduced a new technique, called dropout. This technique was compared with several traditional regularization strategies including KL-sparsity, max-norm, L1, and L2. Their comparison involved evaluating the performance of the techniques mentioned and other pairings, as presented in Table 1. They concluded that max-norm combined with dropout yielded the best results with the lowest classification error. However, it would have been beneficial if they had also included the results of dropout alone in their comparison, similar to how they included Max-norm and L2. This provides insights into whether dropout outperforms other techniques. Unfortunately, the table presenting their comparison does not display the results of dropout alone, even though it was intended to be compared with other methods.

Peng *et al.* [19] conducted a comparison of four regularization methods based on embeddings using neural networks. These methods encompassed dropout, re-embedding, penalizing the l2- norm of embeddings, and penalizing the l2- norm of the weights. Their findings revealed that, except for the re-embedding of words, all regularization strategies employed helped mitigate overfitting. The effectiveness of these regularization techniques was found to be influenced by the dataset size. Dropout performed well, l2-norm improved the training performance (better training accuracy), and re- embedding added nothing to the performance. The combination of the l2-norm with dropout produced the best result. This paper presents the importance of the size of the dataset, as the performance of regularization models depends on the size of the dataset. Two datasets were used; experiments I and II used 7000, and 150000 data respectively.

Moradi *et al.* [5] compared the performance of the following regularization techniques Data Augmentation, Adversarial training, Label smoothing, Weight Decay, Batch Normalization, Dropout, Fractional pooling, noisy inputs, noisy weights, Ensemble, and dropoutconnect using CIFAR-10 dataset. The results show that adding noise to the input and weight did not improve the performance, whereas the rest of the techniques improved the performance. The results showed that weight decay and augmentation were acceptable for use. However, for dropout, it is advised to ensure sufficient computational resources. In such cases, either batch normalization or an ensemble can be utilized. In their experiment, batch normalization exhibited a slower convergence rate, whereas the ensemble technique achieved the highest accuracy. The problem here is the use of a single dataset to determine the effectiveness of the techniques.

Raouhi *et al.* [20] conducted a comparison of several regularization techniques, namely Lasso, ridge, and ElasticNet, using weather and climate data. The evaluation of these techniques was based on metrics such as R2 and Mean Squared Error. The authors of this study have made several observations and conclusions. They found that ridge regularization strikes a balance between handling collinearity among independent variables while maintaining model simplicity. On the other hand, they noted that lasso regularization encounters difficulties when dealing with correlated variables. In such cases, the lasso tends to set some variables to zero while retaining only one variable, which leads to a loss of information and can negatively impact the accuracy of the model.

Swastika *et al.* [21] applied some regularization techniques to some deep-learning models for the detection of Malaria. The dataset used was a malaria dataset with 27588 images.

Table 1. Comparison of different regularization methods on MNIST [18]

Method	Text Classification
L2	1.62
L2 + L1 applied towards the end of training	1.60
L2 + KL sparsity	1.55
Max-norm	1.35
Dropout + L2	1.25
Dropout + Max-norm	1.05

The regularization techniques used were L2 regularization, dropout, and data augmentation. These techniques were applied to the following Convolutional Neural Network Architectures BaselineNet-1, MicroVGGNet, ResNet-50, and Rajaraman. The results showed that regularizations increased the accuracy, specificity, and sensitivity of the models.

Kamalov and Leung [22] investigated the impacts of regularization on the performance of the neural network using an imbalanced dataset from Reuters. Three regularization techniques are used: Dropout, L1, and L2. Numerical experiments showed that L1 regularization can tackle overfitting in Neural Networks for imbalanced datasets.

Marin *et al.* [23] conducted an empirical study to evaluate the impact of optimizations and various regularization techniques using three convolutional neural network architectures. The CIFAR-10 and Fashion-MNIST datasets were used for the evaluation. Researchers have focused on two commonly considered metrics to assess the efficiency of the algorithms: generalization, which measures the model's performance on unseen data, and speed of convergence, which evaluates the time taken for the algorithm to converge or reach the optimal value. Based on their evaluation, Marin *et al.* [23] concluded that classical Nesterov, adaptive, Adam, and AdaMax optimizers yielded the highest accuracy. Most of the optimizers achieved zero loss and 100% accuracy within approximately 350 epochs. However, the researchers did not provide a table comparing the performances of the different regularization techniques.

We found one limitation in Moradi *et al.* [5], Srivastava *et al.* [18], Kamalov and Leung [20], Marin *et al.* [21], Zeiler and Fergus [22] and Bishop [23] based on the analyses mentioned above. The use of a single dataset and two datasets of equal sizes to compare and draw conclusions regarding the efficacy of regularization algorithms has been noted as a drawback.

3. Materials and method

3.1. Materials

To address the limitation, we have chosen two datasets to work with. The Flower dataset and the CIFAR10 dataset. The reason for this is to know whether the results can be generalized. There is a need to know if there is consistency in the performance of the models on both datasets.

The CIFAR-10 dataset has a total of 60000 images of 32x32x3 dimensions. It has 10 classes, each consisting of 6000 images. The dataset is divided into 50000, and 10000 for training and testing respectively. 20% of the training set is used as a validation set. Airplanes, Automobiles, trucks, ships, birds, cats, Deer, dogs, Horses, and frogs are made up of the CIFAR-10 classes.

The flower dataset has a total of 3640 images with 5 classes and the same dimension as CIFAR-10. The dataset was divided in the same ratio as CIFAR-10, with 3100 for training and 540 for testing. The validation set is derived from the training set, using 20% of the training data. The flower dataset includes classes such as Tulips, Sunflowers, Roses, Dandelion, and Daisy, with each class having 793, 693, 635, 892, and 627 photos, respectively, according to the provided statistics.

Although the original Kaggle flower data consists of 3,670 images, we used 3,640 images to accommodate a batch size of 20. The Summary of the two datasets is shown in Table 2.

Other Materials include PC, and Google Colab for experiments.

3.2. Method

3.2.1. Design of model architecture

The CNN model consists of three convolutional, and fully connected layers each with a dropout. Each convolutional layer is followed by an activation function called ReLU and a max-pooling layer for downsampling. Similarly, ReLU follows each linear layer with a dropout. The final linear layer uses the SoftMax function for classification.

The purpose of the three convolutional layers is to extract features. The first layer is expected to detect simple features like lines and blobs. The intermediate layers in the second layer are anticipated to recognize more complex patterns such as Stripes, circles, honeycombs, and faces. According to Zeiler and Fergus [24], the third and final layer is responsible for identifying critical features for classification, including animal faces like Trucks, cats, birds, airplanes, ships, deer, dogs, frogs, and horses, as well as flowers like Roses, Sunflowers, Tulips, Dandelion, and Daises.

The ReLU activation function is used to introduce non-linearities in the model. It has a lower bound of zero and no upper bound. As stated by Goodfellow *et al.* [25], ReLU does not saturate.

The architectural structure of the CNN model is presented in Figure 1.

3.3. Performance Evaluation Metrics

In this research, the metric used to evaluate the model's performance is accuracy. Accuracy represents the proportion

Table 2. Summary of the two datasets.

Datase t	Number of classes	Training size	Validation size	Test size	Total size
CIFA R10	10	40000	10000	10000	60000
Flower	5	2480	620	540	364 0

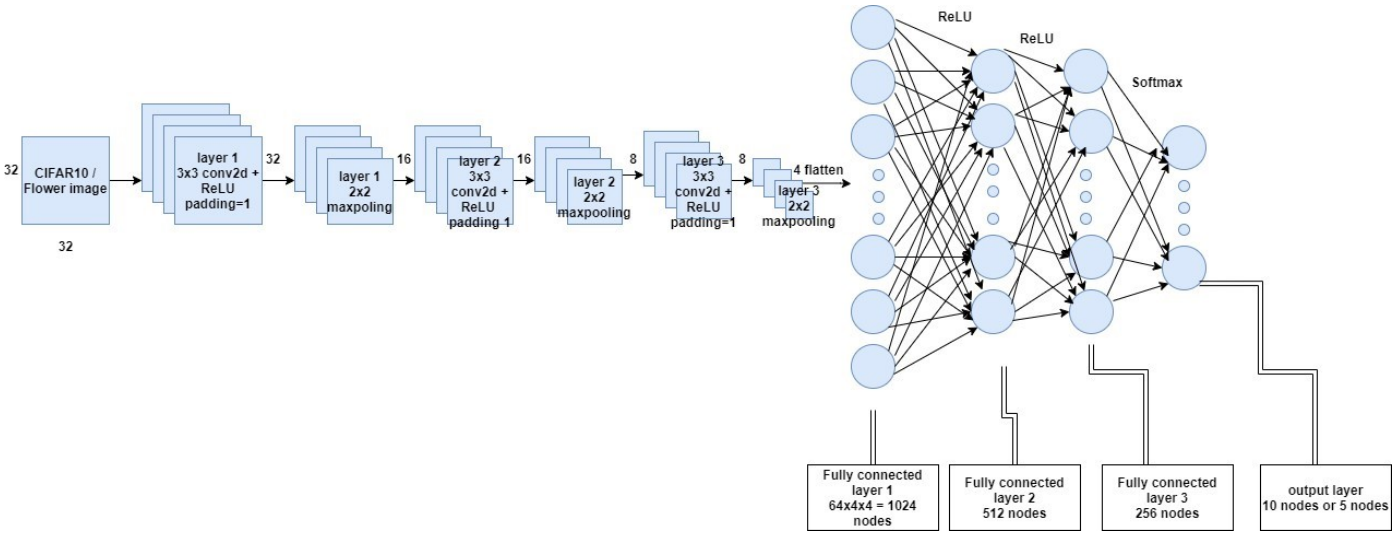

Figure 1. CNN Model Architecture.

of correct predictions made by the model compared to the total number of predictions. It quantifies the number of accurate classifications.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{All observations}} \quad (1)$$

3.4. Research questions

To address the identified limitation, three research questions were proposed.

1. RQ0: How does Augmentation, Weight decay, and Dropout perform on Flower and CIFAR10 datasets?
2. RQ1: How does the pairing of the techniques (Augmentation + Weight Decay, Augmentation + Dropout, and Weight Decay + Dropout) perform on Flower and CIFAR10 datasets?
3. RQ2: How does the combinations of Weight Decay, Augmentation, and Dropout perform on Flower and CIFAR10 datasets?

To investigate the potential benefits or drawbacks, we conducted experiments by combining two and three strategies, inspired by the studies conducted by Srivastava *et al.* [19] and Peng *et al.* [20]. The aim was to determine if these combinations would yield greater benefits compared to using a single strategy or if they would introduce potential issues, such as underfitting.

3.5. Implementation

To practically implement the CNN model, we utilized the PyTorch framework.

1. In this work, we addressed the constraint by loading our data into the dataset using two datasets. The CIFAR10 dataset and the flower dataset were utilized. We obtained the CIFAR10 dataset from Torchvision, and to import the data, we applied PyTorch's transform function for data normalization and conversion to PyTorch tensors. This normalization process, using a mean and standard deviation of 0.5 for each of the three RGB channels, helps the model train more efficiently. We created a trainset and a test set from the CIFAR10 dataset and then generated three loaders for training, validation, and test sets using these sets. Similarly, for the flower dataset obtained from Kaggle, we followed the same steps as CIFAR10 to create a train loader, validation loader, and test loader.
2. To ensure the accuracy of the data, we performed data visualization. You can find visual representations of a training batch from each dataset in Figure 2 and 3.
3. The PyTorch framework is utilized to construct the model using the nn.Module class. The CNN model consists of three convolutional layers and three linear layers. The input image has dimensions of 32x32x3. The image tensor for the first convolutional layer is 32x32x3. The second convolutional layer has an image tensor of 16x16x16, while the third convolutional layer has an image tensor of 8x8x32. After the convolutional layers, the image is flattened and passed to the first fully connected layer, which contains 64x4x4=1024 nodes. The second hidden layer, known as layer 2, is fully connected and has 512 nodes. The final fully connected layer, layer 3, has 256 nodes and utilizes the Softmax algorithm for classifying either 10 or 5 classes. The output of the pooling or convolu-

Figure 2. Batch of training data from CIFAR10.

Figure 3. A batch of training data from flower.

Figure 4. The test result of our Baseline Model.

Figure 5. Result of Weight Decay + Dropout + Augmentation.

tional filter operation is determined using the following formula:

$$W_{out} = \frac{(W_{in} - F + 2P)}{S} + 1 \quad (2)$$

Where F is for the Kernal size of the filter, P is for padding, S is for stride, Win for input width, and Woutfor output size.

The model performance is evaluated using a loss function and an optimization process. The aim is to find the best bias and weight that will minimize the loss/cost function [6]. The learning algorithm used is the Stochastic Gradient Descent while the loss function used is Negative Log Likelihood (NLLL). This loss function was considered because it is used to solve classification problems [26]. Here is the formula for the softmax activation function:

$$pk = \frac{e^{fk}}{\sum_j e^{fj}} \quad (3)$$

Where f stands for class. Equation (4) is the NLLL:

$$L_i = -\log(py_i) \quad (4)$$

Where pyi is the output of softmax, in this case p^k. Li is the loss function. The output of the model (p^k) and the loss (Li) are used during backpropagation.

4. SGD is used in this work. For every iteration a random sample of mini-batch is chosen and the average is taken to

get the cost function [6]. Gradient descent is not efficient and is slower compared to SGD [27]. The road to minima is noisier for SGD compared to gradient descent.

5. The model is trained using training data and evaluated using validation data. The training is done for 100 epochs.
6. To improve the performance of the model, Bayesian optimization (BO) was used. BO was used to search for the best hyperparameters. The Adaptive Experimentation (AX) platform was used because it does hyperparameter search using BO for CNN [28], and also automates the process of choosing hyperparameters in a large space. BO is an approach that uses Bayes Theorem in search for the minimum/maximum of an objective function. It utilizes an acquisition function and a surrogate model to guide the search process [29]. The acquisition function focuses the search on areas where there is a potential for better performance than the current best, while the surrogate model predicts the objective function. Bayesian optimization is chosen in this work because the AX platform used provides it. BO accelerates the search process and enables reaching the global minimum with fewer iterations compared to other methods. In the context of deep learning architecture, where parameter search is costly and noisy, random search or grid search methods are inadequate. In the AX platform, Bayesian optimization is initially performed using the Sobol method for the first five trials and then switches to Gaussian Process-based Bayesian Optimization (GPEI) for the remaining trials. The learning rate, weight decay, and momentum are fine tune using the AX platform. The learning rate and weight decay are varied within the range of 1e-6 - 0.4. The momentum is explored within the range of 0.0 - 1.0. Dropout values range from 0.1 - 1.0. For augmentation, random rotation, and random horizontal flip techniques are employed. The images are randomly rotated horizontally with a default probability of 0.5. Additionally, specific numbers like 5, 10, 15, 20, 25, 30, 35, and 45 are sought during the process.
7. In this stage, we employ a test loop to assess the performance of our model using the test data. Before executing the test, we switch the model to the evaluate mode. Then, we proceed to evaluate the accuracy of our model.
8. We visualize the predictions made by the model for better understanding. Figure 4 and 5 presents the model's predictions for a test sample, comparing the baseline model with the model incorporating augmentation, weight decay, and dropout. In Figure 4, any errors made by the Baseline Model are highlighted in red.

4. Results and discussion

Our experimental findings are shown in Table 3 and 4. To know the best techniques on a given data we need to compare their performances. To get the best performance in a model, we fine-tune the model using the AX platform which results in different hyper parameters. The three research questions will be addressed using the tables.

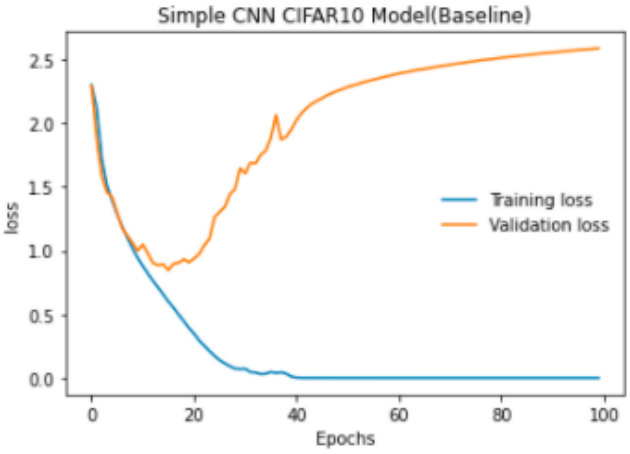

Figure 6. CIFAR10 Baseline model.

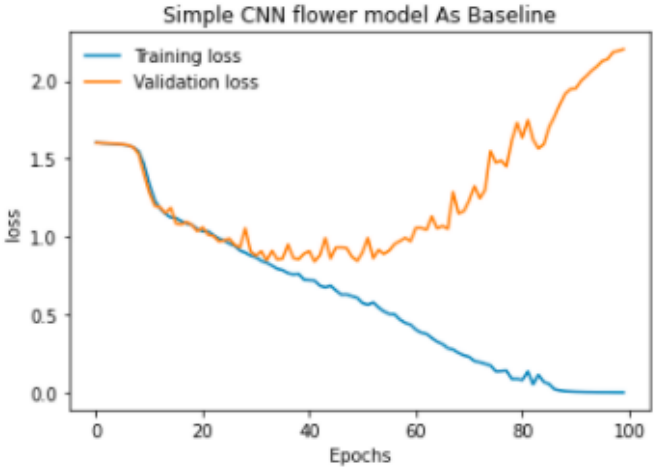

Figure 7. Flower Baseline model.

Table 3 presents the results of the Baseline model, which does not incorporate any regularization techniques. The test accuracy for this model is 73%, while the training accuracy is 100%. These values indicate that the model is overfitting to the training data. Figure 6 and 7 illustrates the training and validation loss of the baseline model for both datasets.

The discrepancy between the training loss and the validation loss depicted in Figure 6 and 7 clearly indicates the presence of overfitting in the model. Our goal is to achieve a scenario where the validation loss decreases in line with the training loss, indicating a better generalization of the model to unseen data.

RQ0: Augmentation vs Weight Decay vs Dropout

Table 3 illustrates that Dropout achieves superior performance on the CIFAR10 dataset compared to Weight Decay and Augmentation. It achieves a test accuracy of 78%, while Augmentation and Weight Decay attain test accuracies of 75% each. Similarly, Table 4 demonstrates that Dropout outperforms the other two methods, achieving a test accuracy of 70%, while

Table 3. CIFAR10 Result.

Hyperparameters							Accuracy (%)	
Techniques	Learning rate	Momentum	Epoch	P	Augmentation	L2	Train	Test
Baseline	0.0059	0.1925	100			0.0000	100	73
Dropout	0.0039	0.2862	100	0.5		0.0000	93	78
Weight Decay	0.0020	0.9000	100			0.0001	100	75
Augmentation	0.0083	0.0000	100		5	0.0000	99	75
Dropout + Weight Decay	0.0055	0.3342	100	0.3		0.0047	89	77
Dropout + Augmentation	0.0039	0.3659	100	0.4	5	0.0000	88	80
Weight Decay + Augmentation	0.0044	0.1719	100		10	0.0002	96	75
Dropout + Weight Decay + Augmentation	0.0027	0.5731	100	0.3	5	6.68e- 06	93	80

Table 4. Results of flower.

Hyperparameters							Accuracy (%)	
Techniques	Learning rate	Momentum	Epoch	p	Augmentation	L2	Train	Test
Baseline	0.0024	0.7911	100			0	100	63
Dropout	0.0086	0.3503	100	0.6		0	82	70
Weight Decay	0.0060	0.9000	100			0.0001	100	66
Augmentation	0.0017	0.8470	100		30	0	77	69
Dropout + Weight Decay	0.0023	0.8177	100	0.2		3.1e- 05	97	66
Dropout + Augmentation	0.0067	0.8252	100	0.4	5	0	97	72
Weight Decay + Augmentation	0.0052	0.7104	100		25	2.3e- 05	87	68
Dropout + Weight Decay + Augmentation	0.0159	0.4353	100	0.4	10	1.5e- 06	90	73

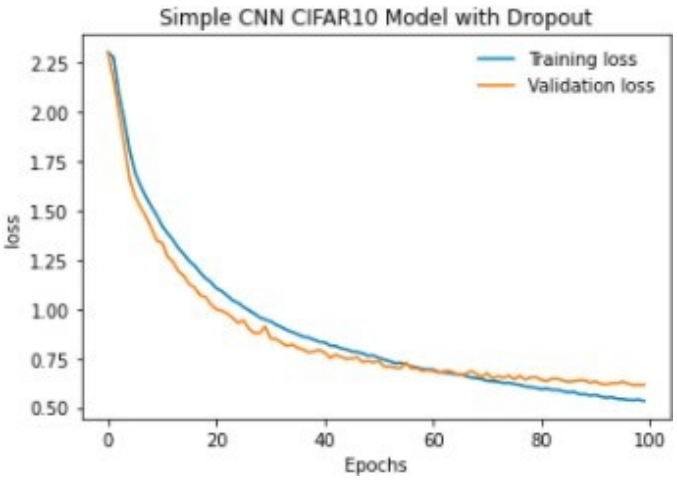

Figure 8. CIFAR10 with Dropout.

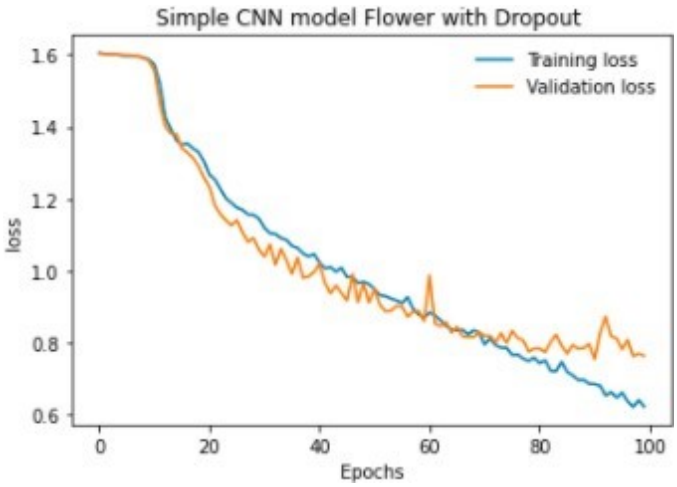

Figure 9. Flower with Dropout.

Augmentation and Weight Decay achieve test accuracy rates of 69% and 66% respectively. These findings highlight the superiority of Dropout over Augmentation and Weight Decay in both small and large datasets. Furthermore, in the Flower dataset, augmentation proves to be more accurate than weight decay, while both methods exhibit equal accuracy in the CIFAR10 dataset. The performance of our model with Dropout in both dataset is shown in figure 8 and 9.

RQ1: Augmentation + Weight Decay vs Augmentation + Dropout vs Weight Decay + Dropout

In this analysis, we explore the impact of the mentioned combinations on the two datasets. We found that Augmentation + Dropout performs well on both datasets. On the CIFAR10 dataset, it achieves a test accuracy of 80%, surpassing the test accuracies of Weight Decay + Dropout (77%) and Augmentation + Weight Decay (75%). Similarly, on the flower dataset,

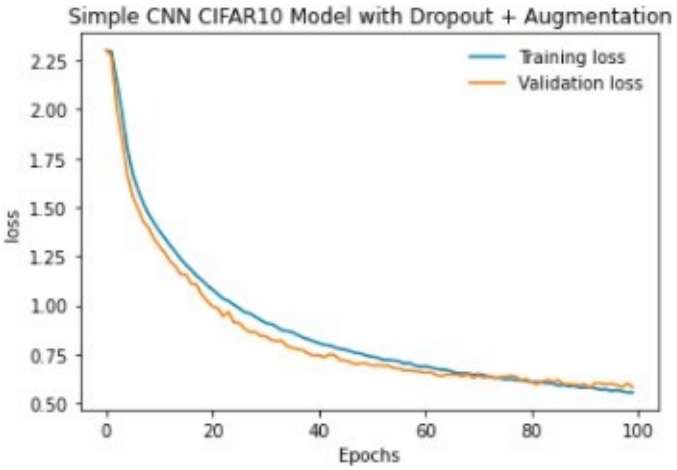

Figure 10. Dropout + Augmentation with CIFAR10.

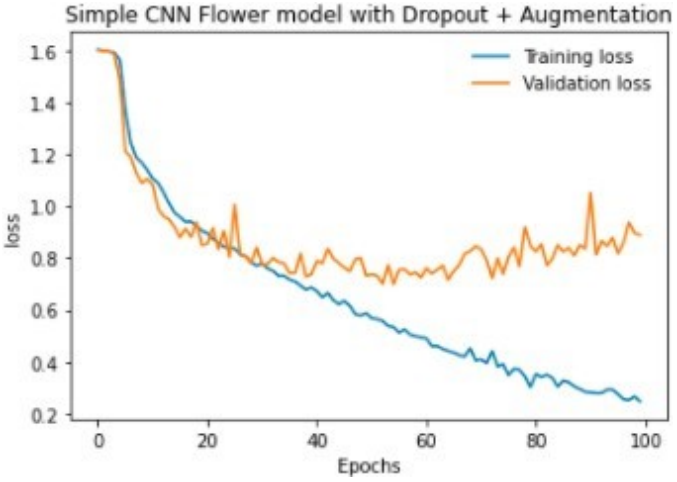

Figure 11. Dropout + Augmentation with Flower.

Augmentation + Dropout achieves a test accuracy of 72%, outperforming the test accuracies of Weight Decay + Dropout (66%) and Augmentation + Weight Decay (68%). The training and validation loss of Augmentation + Dropout on both datasets is depicted in Figure 10 and 11.

RQ2: Augmentation + Weight Decay + Dropout

Lastly, we analyze the combined effect of the three strategies on both datasets and compare their effectiveness to other techniques. The three combinations we tested on the two datasets yielded the best results, although in some cases they performed similarly to Augmentation + Dropout. This indicates that incorporating Weight Decay did not improve the outcome as expected. Its results from Table 3 were identical to Augmentation + Dropout, while Table 4 showed slightly better results, with only a 1% improvement over Augmentation + Dropout. Figure 12 and 13 presents the performance of Weight Decay, Dropout and Augmentation in both dataset.

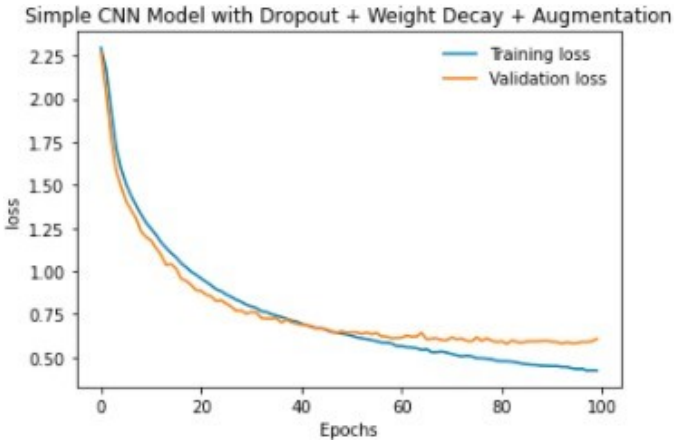

Figure 12. Dropout + Weight Decay + Augmentation with CIFAR10.

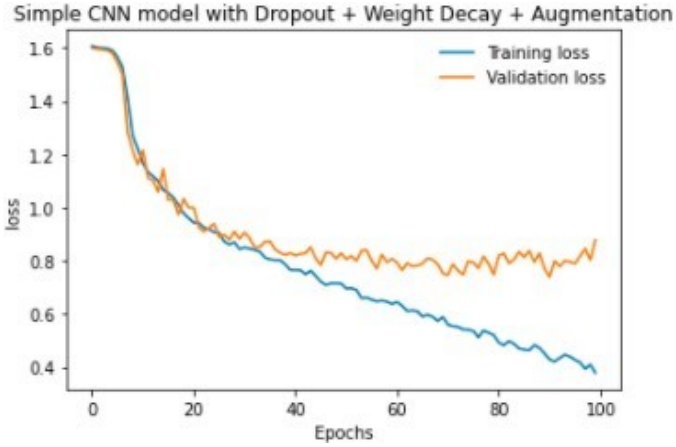

Figure 13. Dropout + Weight Decay + Augmentation with Flower.

4.1. Analysis

To gain a deeper understanding of the behavior of each strategy, we performed two types of analysis on all the strategies using the CIFAR10 dataset. We evaluated each method based on the number of epochs, following the approach used in a study by Moradi *et al.* [5]. Additionally, similar to the method employed by peng *et al.* [20], we evaluated all the strategies considering the size of the datasets.

4.1.1. Analysis based on number of epochs

Table 5 and Figure 14 displays the obtained results. In this analysis, we trained the model on the entire CIFAR10 training dataset, which consists of 50,000 samples, using the hyperparameters specified in Table 4. In contrast, the results reported in Table 3 were based on a setup with 10,000 test samples, 10,000 validation samples, and 40,000 training samples.

4.2. Analysis based on the size of the dataset

We investigate the relationship between dataset size and various regularization methods. We specifically focused on the CIFAR10 dataset due to its size. For each dataset size, we utilized different hyperparameters, except for the largest size of

Table 5. Results of analysis based on number of epochs.

Test Accuracies at Different Epochs (%)										
Techniques	10	20	30	40	50	60	70	80	90	100
Baseline	67	68	69	72	73	73	74	74	73	74
Dropout	54	67	72	74	76	77	78	78	79	79
Weight Decay	69	72	71	72	74	75	75	76	74	75
Augmentation	68	73	74	75	74	75	76	75	75	76
Weight Decay + Dropout	60	70	74	76	77	77	77	77	78	78
Augmentation + Dropout	58	70	74	76	78	79	79	79	80	80
Augmentation + Weight Decay	60	71	74	75	74	74	75	76	75	75
Augmentation + Weight Decay + Dropout	62	71	76	77	79	79	80	80	80	81

Table 6. Results of flower.

Technique	Epoch	10000 Accuracy (%)	20000 Accuracy (%)	30000 Accuracy (%)	40000 Accuracy (%)	50000 Accuracy (%)
Baseline	100	61	67	72	73	74
Dropout	100	69	74	76	78	79
Weight Decay	100	66	72	73	75	75
Augmentation	100	64	69	73	75	76
weight Decay + Dropout	100	63	73	75	77	78
Augmentation + Dropout	100	70	75	77	80	80
Augmentation + Weight Decay	100	65	70	73	75	75
Augmentation + Weight Decay + Dropout	100	70	75	78	80	81

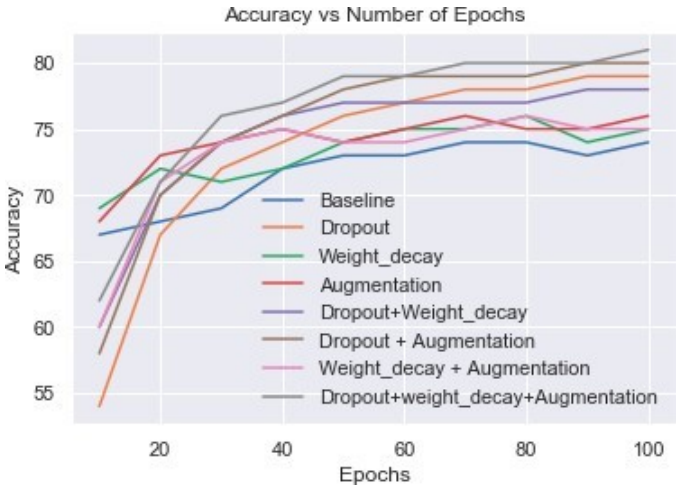

Figure 14. Accuracy vs Number of Epochs on CIFAR10.

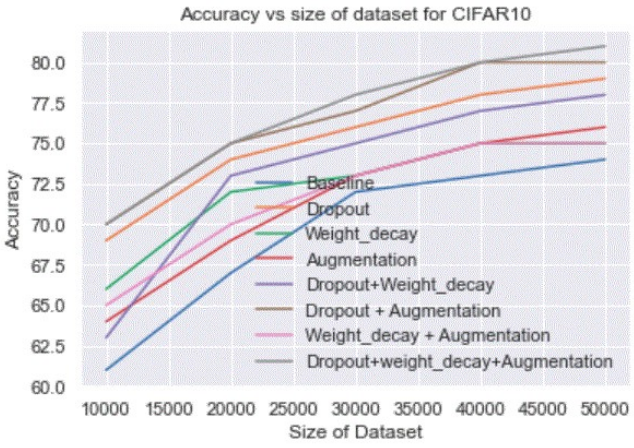

Figure 15. Accuracy vs size of the dataset on CIFAR10.

50,000, where we used 40,000 samples as there was no additional validation data available. We used the appropriate hyperparameters for each dataset size during training. Our findings indicate that while the performance of certain approaches improved as the dataset size increased, others reached a plateau and did not show further improvement as shown in Table 6 and Figure 15.

Summary of findings from analysis

Based on the information presented in Table 5, Weight Decay achieves quicker convergence compared to Augmentation and Dropout for RQ0. On the other hand, Dropout takes longer to converge because of the independence among neurons. For RQ1, Augmentation + Dropout and Weight Decay + Dropout converge at a slower rate compared to Augmentation + Weight Decay. Among all the combinations, Augmentation + Weight Decay + Dropout shows the slowest convergence rate for RQ2, requiring 100 epochs to reach its highest accuracy. Generally, Weight Decay demonstrates faster convergence than other methods, while Augmentation + Weight Decay + Dropout converges more slowly.

As shown in Table 6, Dropout outperforms Weight Decay and Augmentation in all sizes of the CIFAR10 dataset for RQ0, as well as in both CIFAR10 and Flower datasets. In terms of RQ1, Augmentation + Dropout outperforms Augmentation +

Weight Decay and Weight Decay + Dropout in all sizes of the CIFAR10 dataset studied, and it also performs better in the Flower datasets. For RQ2, Augmentation + Weight Decay + Dropout outperforms all other strategies across all dataset sizes, including the CIFAR10 and Flower datasets. The combination of Weight Decay + Augmentation + Dropout is generally the most effective strategy, as indicated in Table 6. Additionally, it was observed that techniques such as Weight Decay, Augmentation + Dropout, and Augmentation + Weight Decay reach a saturation point at dataset sizes of 40,000, while techniques like Dropout, Augmentation, Weight Decay + Dropout, and Augmentation + Weight Decay + Dropout continue to improve steadily with more data.

5. Discussion

Comparing Dropout with Weight Decay and Augmentation, Dropout proves to be the most effective method, although it takes longer to converge. For the CIFAR10 dataset, the most favorable values of p for Dropout were found to be 0.4 and 0.5, while for the flower dataset, the value of p was determined to be 0.6.

Weight Decay performs less effectively than Dropout and Augmentation. However, it converges faster compared to the other methods. At epoch 10, Weight Decay achieves the highest accuracy of 69% on the CIFAR10 dataset and remains superior to Augmentation when the dataset size is smaller, such as 10,000 or 20,000. The suitable values for weight decay were found to be 0.0001, 0.0002, and 0.0003 in most of our studies.

Although Augmentation is less efficient than Dropout, it converges more rapidly. From epoch 10 to 40, Augmentation outperforms Dropout based on our analysis. Random rotation of images by 5 and 10 degrees yields better results on CIFAR10, while values of 25 and 30 produce positive results on the flower dataset. It should be noted that the flower dataset, unlike the CIFAR10 dataset, consists of images of varying sizes that were resized to 32.

Dropout consistently proves to be superior to the combination of Weight Decay in all conducted trials, even though Weight Decay converges faster than Dropout.

Among the combinations, Augmentation + Dropout has demonstrated the highest effectiveness compared to Weight Decay + Dropout and Augmentation + Weight Decay. It ranks second among the seven approaches, with Augmentation + Weight Decay + Dropout being the most effective on both datasets. The optimal Dropout values were found to be 0.3 and 0.4, while random rotation values of 5, 10, and 15 produced positive results. However, Weight Decay initially outperforms this combination due to its early convergence, as Augmentation + Weight Decay + Dropout requires more time to converge.

Augmentation + Weight Decay is less effective than Augmentation + Dropout since it combines weight decay with augmentation. Applying Dropout alone is always preferable to using Augmentation + Weight Decay.

The most effective method across both datasets is Augmentation + Weight Decay + Dropout. The results collected demonstrate its superiority over the other six strategies. However, it should be noted that it performs best given sufficient time. The optimal values for p in this combination were 0.3 and 0.4, and image rotation by 5 or 10 degrees proved beneficial. Moreover, weight decay values of 0.0002, 0.000001, and 0.000006 were identified as efficient.

Overall, our results show that dropout outperforms both augmentation and weight decay similar to the results presented by Moradi *et al.* [5] and Tian and Zhang [30]. Also, in terms of convergence weight decay outperforms dropout and augmentation similar to the results obtained by Moradi *et al.* [5] and Tian and Zhang [30].

6. Conclusion

In this study, we assessed the performance of three regularization strategies – Dropout, Weight Decay, and Augmentation – alongside their various combinations, utilizing a straightforward CNN model across two distinct datasets. The effectiveness of these methods was evaluated concerning dataset size and the total number of training epochs. This evaluation provided insights to address three specific research questions, RQ0, RQ1, and RQ2. Findings indicate that Dropout outperforms Weight Decay and Augmentation (RQ0), and among the various strategy combinations, Dropout coupled with Augmentation leads to superior results (RQ1). When all three techniques are combined, they surpass the individual methods in effectiveness (RQ2). Additionally, it was noted that Weight Decay achieved quicker convergence than the other methods, while the tripartite combination demonstrated a slower rate of convergence.

Future research should investigate the balance between accuracy and convergence speed to better understand the implications of these techniques in model training.

References

[1] G. E. Hinton, S. Osindero & Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006. https://europepmc.org/article/med/16764513.

[2] Y. Lecun, L. Bottou, Y. Bengio & P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE **86** (1998) 2278. https://ieeexplore.ieee.org/document/726791.

[3] A. Graves, Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850. (2013). https://arxiv.org/abs/1308.0850.

[4] Y. Bengio, P. Simard & P. Frasconi, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks **5** (1994) 157. https://ieeexplore.ieee.org/document/279181.

[5] R. Moradi, R. Berangi & B. Minaei, "A survey of regularization strategies for Deep Models", Artificial Intelligence Review **53** (2019) 394. https://link.springer.com/article/10.1007/s10462-019-09784-7.

[6] M. Nielsen, *Neural Networks And Deep Learning*, Determination Press, San Francisco, CA, USA, 2015, pp. 15 – 24. http://neuralnetworksanddeeplearning.com/

[7] P. Y. Simard, D. Steinkraus & J. C. Platt, *Best practices for convolutional neural networks applied to visual document analysis*, Seventh International Conference on Document Analysis and Recognition, Edinburg, United Kingdom, 2003. https://www.researchgate.net/publication/220860992_Best_Practices_for_Convolutional_Neural_Networks_Applied_to_Visual_Document_Analysis.

[8] A. Krizhevsky, I. Sutskever & G. E. Hinton, "ImageNet classification with deep convolutional Neural Networks", Communications of the ACM **60** (2017) 84. https://dl.acm.org/doi/10.1145/3065386.

[9] G. Hinton, S. Nitish, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", arXiv preprint, 2012. https://www.researchgate.net/publication/228102719_Improving_neural_networks_by_preventing_co-adaptation_of_feature_detectors.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, C. Aaron & Y. Bengio, "Generative adversarial nets", in *Advances in Neural Information Processing Systems*, 2014, pp. 2672-2680. https://arxiv.org/abs/1406.2661.

[11] L. Prechelt, "Early stopping — but when?", in *Lecture Notes in Computer Science*, Germany, pp. 53–67, 2012. https://link.springer.com/chapter/10.1007/978-3-642-35289-8_5.

[12] Y. A. LeCun, L. Bottou, G. B. Orr & K. R. Müller, "Efficient backprop," *Lecture Notes in Computer Science*, Springer Verlag, 2012, pp. 9–48. https://link.springer.com/chapter/10.1007/978-3-642-35289-8_3.

[13] L. Breiman, "Bagging predictors", Machine Learning **24** (1996) 123. https://link.springer.com/article/10.1007/BF00058655.

[14] S. Ioffe & C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, Proceedings of the 32nd International Conference on Machine Learning (ICML), 2015, pp. 448–456. https://arxiv.org/abs/1502.03167.

[15] A. Krizhevsky, I. Sutskever & G.E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", Advances in Neural Information Processing Systems (NIPS) **25** (2012) 1097. https://dl.acm.org/doi/abs/10.1145/3065386.

[16] R. Tibshirani, "Regression shrinkage and selection via the lasso", Journal of the Royal Statistical Society: Series B (Methodological) **58** (1996) 267. https://academic.oup.com/jrsssb/article/58/1/267/7027929.

[17] T. Hastie, J. Friedman & R. Tisbshirani, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, New York, 2017, pp. 1–758. https://link.springer.com/book/10.1007/978-0-387-84858-7.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever & R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", The Journal Of Machine Learning Research **15** (2014) 1929. https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf?utm_content=buffer79b43&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer,.

[19] H. Peng, L. Mou, G. Li, Y. Chen, Y. Lu & Z. Jin, *A comparative study on regularization strategies for embedding- based Neural Networks*, Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, (2015). https://aclanthology.org/D15-1252.pdf.

[20] F. Kamalov & H. H. Leung, *Deep learning regularization in imbalanced data*, 2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI), United Arab Emirates, 2020. https://ieeexplore.ieee.org/abstract/document/9256674.

[21] I. Marin, A. Kuzmanic Skelin, & T. Grujic, "Empirical evaluation of the effect of optimization and regularization techniques on the generalization performance of deep convolutional Neural Network", Applied Sciences **10** (2020) 7817. https://www.mdpi.com/2076-3417/10/21/7817.

[22] M. D. Zeiler & R. Fergus, *Visualizing and understanding Convolutional Networks*, Computer Vision - ECCV 2014: 13th European Conference and proceedings, Zurich, Switzerland, 2014, pp. 818–833. https://link.springer.com/chapter/10.1007/978-3-319-10590-1_53.

[23] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2016. https://link.springer.com/in/book/9780387310732.

[24] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, Y. Bengio, *Identifying and attacking the saddle point problem in high- dimensional non-convex optimization*, Proceedings of the 27th International Conference on Neural Information Processing Systems, United States, 2014, pp. 2933–2941. https://arxiv.org/abs/1406.2572.

[25] I. Goodfellow, Y. Bengio & A. Courville, *Deep Learning*, MA: MIT Press Ltd, Cambridge, 2017. https://mitpress.mit.edu/9780262035613/deep-learning/.

[26] E. M. Raouhi, M. Lachgar &A. Kartit, *Comparative study of regression and regularization methods: Application to weather and Climate Data*, Proceedings of the 6th International Conference on Wireless Technologies, Springer Singapore, 2021, pp. 233–240. https://link.springer.com/chapter/10.1007/978-981-33-6893-4_22.

[27] W. Swastika, R. B. Widodo, G. A. Balqis, & R. Sitepu, *The effect of regularization on deep learning methods for detection of malaria infection*, 2021 International Conference on Converging Technology in Electrical and Information Engineering (ICCTEIE),Bandar Lampung, Indonesia, 2021. https://ieeexplore.ieee.org/abstract/document/9650646.

[28] E. Bakshy, L. Dworkin, B. Karrer, K. Kashin, B. Letham, A. Murthy, S. Singh, *AE: A domain-agnostic platform for adaptive experimentation*, In 32nd Conference on Neural Information Processing Systems, Montreal, pp. 1-8, 2018. https://eytan.github.io/papers/ae_workshop.pdf.

[29] J. Mockus, V. Tiesis, A. Zilinskas, "The application of Bayesian methods for seeking the extremum", Towards Global Optimisation 2 (1978) 117. https://www.researchgate.net/publication/248818761_The_application_of_Bayesian_methods_for_seeking_the_extremum.

[30] Y. Tian & Y. Zhang, "A comprehensive survey on regularization strategies in Machine Learning", Information Fusion **80** (2022) 146. https://www.sciencedirect.com/science/article/abs/pii/S156625352100230X.