Dicle University
Journal of Engineering
https://dergipark.org.tr/tr/pub/dumf
duje.dicle.edu.tr

Research Article

Effect on model performance of regularization methods

Cafer Budak[1*], Vasfiye Mençik[2], Mehmet Emin ASKER[3]

[1] Dicle University ,Department of Electric-Electronic Engineering, cafer.budak@dicle.edu.tr, Orcid No: 0000-0002-8470-4579
[2] Dicle University, Department of Electric-Electronic Engineering, vasfiyemencik@gmail.com, Orcid No: 0000-0002-3769-0071
[3] Dicle University ,Department of electricity and energy, measker@dicle.edu.tr , Orcid No: 0000-0003-4585-4168

ARTICLE INFO

Article history:

Received 16 November 2021
Received in revised form 23 November 2021
Accepted 1 December 2021
Available online 31 December 2021

Keywords:

Overfitting, Machine learning, Regularization

ABSTRACT

Artificial Neural Networks with numerous parameters are tremendously powerful machine learning systems. Nonetheless, overfitting is a crucial problem in such networks. Maximizing the model accuracy and minimizing the amount of loss is significant in reducing in-class differences and maintaining sensitivity to these differences. In this study, the effects of overfitting for different model architectures with the Wine dataset were investigated by Dropout, AlfaDropout, GausianDropout, Batch normalization, Layer normalization, Activity normalization, L1 and L2 regularization methods and the change in loss function the combination with these methods. Combinations that performed well were examined on different datasets using the same model. The binary cross-entropy loss function was used as a performance measurement metric. According to the results, the Layer and Activity regularization combination showed better training and testing performance compared to other combinations.

Doi: 10.24012/dumf.1051352

* Corresponding author

Introduction

In order to make predictions about test data in machine learning applications, a model is created according to the patterns obtained from the training data. Two things can be mentioned as a result of this process: underfitting and overfitting. As a result of the poor design of the model and the optimization process, the problem is called underfitting cannot model the training data and cannot generalize for new data that the model has not seen before. The fact that the model is not strong enough regularized more than necessary and not trained for enough time causes the network to not learn the relevant inferences from the training data as much as necessary. This situation can cause high errors in both training and test datasets. If the generated model is trained for too long, a higher error rate will likely occur in the test dataset, even though the training dataset has a low error rate[1]. This raises the possibility of overfitting in the model. One of the ways to avoid the overfitting problem is to use more training data. However, this is not possible in some cases, so using regularization techniques is a good solution. Regularization prevents the model from learning a more complex or flexible model by adjusting the coefficient estimates towards zero for the case of overfitting and underfitting the data. In this study, the issue of overfitting is discussed. Regularization layers such as data augmentation[2], Dropout[3] and batch normalization[4], group normalization[5], Layer normalization[6], Instance Normalization[7] methods are used to prevent the overfitting problem. In addition, the methods such as drop-connect[8], maxout[9], Lasso [10], and weight normalization[11] are also used to prevent overfitting problems. However, these methods, which have high network resilience and can fail in low data applications, cannot take benefit of input invariants. Data augmentation, an effective technique to increase the amount and variety of data, is used to create more data from existing data by

applying various transformations (rotations and adding Gaussian noise) to the original data set and teaching a model about invariance in the data field. However, a successful augmentation process for one data set may not be as successful for a different data set. At the same time, data augmentation, which is widely used in practice, increases the computational cost and may not be completely effective in some medical applications such as tumour identification [12]. Despite being trained on large datasets such as ImageNet [13] or datasets containing millions of labelled images, deep network models are still susceptible to overfitting, even with little success in transfer learning[14]. Since deep neural networks do not tend to generalize with a few examples, the problem of overfitting may become inevitable for tasks in new fields.

In this study, the regularization methods used to prevent overfitting and their combinations were compared. The effects on the model performance of binary combinations of these methods were investigated. Wine dataset[15], The MNIST database of handwritten digits[16], the Fashion-MNIST datasets[17] and Cifar10[18] were used in the studies.

The main contribution of this study is as follows:

- To compare the effect of each regularization method on the model in terms of the loss function.
- To contribute to the literature on the relative importance of different regularization methods and their impact on the model performances.
- To show the effect of combinations of different regularization methods on the model performance.

The remainder of this article is organized as follows: Part two is the proposed method, Part three is the evaluation and discussion of the experimental studies and the results obtained. Chapter four contains conclusions.

Related works

In deep neural networks [19], which are arduous to train and used in many computer vision tasks, appropriate model initiation strategies and regularization are used for rapid training [20]. Batch normalization is one of the normalization methods that improve deep learning performance. This method, which accelerates the training time, ensures that the normalization effects are not lost during training and that the models approach the minimum loss point, perform well in large batch sizes. Unlike batch normalization, Layer normalization, which normalizes each feature to zero mean and unit variance, uses the same training and test times calculation. Unlike activation-based normalization methods, weight-based normalization is commonly used to avoid overfitting issues. This method, which adds an independent parameter to the cost function in which the model performance is measured, in other words, imposes constraints on the weights, forces the weights to take small values and tries to avoid the problem of overfitting. Some researchers have increased the number of samples in their datasets to avoid the overfitting problem. [21]. Early stopping [22] for kernel boosting algorithms, early stopping for least squares regression [23], and strong convex problems [24] are parts of optimization and normalization that have been studied to reduce the overfitting problem. There are also studies examining the gap in generalization ability in complex networks [25]. Dropout, one of the methods that reduce the overfitting problem and increase the performance of deep networks, is a widely used stochastic normalization technique. This method randomly removes units from the network to avoid memorization in training data. Cutout[26] is other methods used to prevent overfitting problem.

Material and methods

Convolutional Neural Network

Convolutional Neural Networks (CNN) [27], which is used in many computer vision applications, is one of the deep learning approaches in which multiple layers are trained Automatic diagnosis of cardiovascular disorders[28] and Detection of unregistered electric distribution transformers in agricultural fields [29] are Works using CNN. CNN, which effectively reduces the number of Artificial Neural Network (ANN) parameters, has revolutionized many areas from image processing to voice recognition. In CNN, a part of deep learning, data features become more discreet when the input data spread to more advanced layers. In image classification, the CNN can detect edges in the first layer, simpler shapes in the second layer, and higher-level features in the next layers. Different layers have different tasks in CNN, which consists of convolutional layers, pooling layers, and fully connected layers. Figure 1 shows a general CNN architecture for image classification.

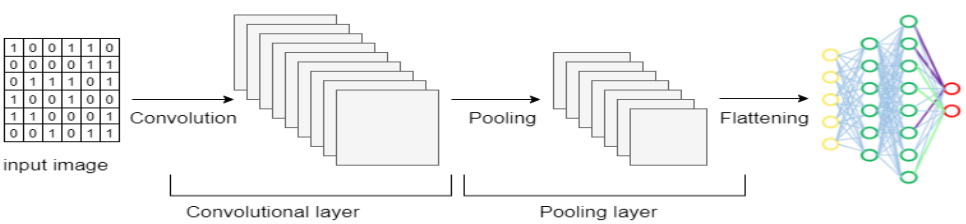


Figure 1.General CNN architecture

Here, convolutional layers and pooling layers extract deep features from the input data and transfer them to fully connected layers to prevent the image from going to artificial neural networks and ensure that the system gives accurate and fast results. Fully connected layers transmit these properties to the output layer. Network training in CNN consists of two stages, a forward stage and a backward stage. In the first stage, the input image is represented by the weights and biases in each layer and the forecast output is used to calculate the loss cost value. In the second step, the value of the gradients of each parameter is calculated backwards with the chain rule, and the value of all parameters is updated used for the next step. This process ends after both steps have been repeated enough and the network training is completed.

Convolutional layers: This layer performs feature extraction using various kernels. There are two basic operations to perform the convolution step. The first of these steps, the linear convolution process, uses kernels to extract features. This convolution operation, whose fundamental purpose is to reduce the size of the input image, is represented as stated in Equation (1). Let f(t) and g(t) be two functions of t. The convolution of f(t) and g(t) is also a function of t, denoted by $(f * g)(t)$, and is defined by the relation

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(T)g(t-T)dT \quad (1)$$

In the second step, the Rectified Linear Unit (RELU) improves the nonlinearity in the network. The output of this process is shown in Equation (2).

$$f(x) = maks(0, x) \quad (2)$$

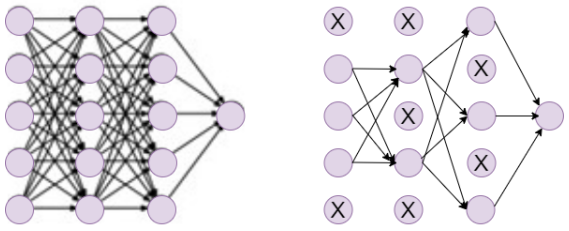
Pooling layers: This layer, which is used to reduce the size of feature maps and network parameters, generally uses different pooling strategies such as average pooling and max pooling. The feature map is made into a one-dimensional column in these layers and transmitted to the neural network.

Fully-connected layers: This layer, which works like a traditional neural network and contains about 90% of the parameters in CNN, is also known as a particular hidden layer. This layer allows the neural network to be transmitted to a vector.

Overfitting problem and regularization techniques

Introducing many parameters in deep learning, which can create deeper architectures to learn more discrete information, can cause overfitting problems. In this problem, the models contain too many terms or use too complex approaches. It would be more practical to distinguish between problems overfitting caused by using a more flexible model than it should be and overfitting caused by using models with irrelevant components. For example, using a more flexible model will add complexity with lower performance than the simple model when used on a dataset that fits the linear model. Using estimators without helpful functionality wastes resources and increases the likelihood of creating undetected estimation errors in the database. Because when using regression at different times to make predictions, it is necessary to measure and record these estimators so that the values in the model can be changed. This can lead to the loss of valuable properties. The ability of one user's results to be copied by another user is effective in portable models. For example, the one-predictive linear regression model that establishes a relationship with the model is portable. Because anyone can apply this model to their data, however, some non-portable models can only be produced by reusing the software data of the user modeling it [30]. For all these reasons, overfitting is undesirable. Recently, many regularization techniques have emerged to prevent overfitting problems. These:

Dropout and DropConnect: During each training, it resets the output of each neuron in the selected layer with a certain probability to avoid complexity on the data and contribute to the improvement of generalization ability. This technique prevents the co-adaptation of feature sensors in the network. Fully-connected layers are effective in editing, but this effect is reduced in convolutional layers. Convolutional layers have fewer parameters and require less regularization than fully connected layers. DropConnect is a well-known technique that randomly drops weights.



(a) (b)

Figure 2. Applying dropout to the neural network (a)Standard Neural Network (b)After applying dropout

Data augmentation: It is the acquisition of new data by applying various transformations of existing data such as horizontal and vertical translation, scaling, squeezing, and horizontal shearing without creating additional costs.

Cutout: It is a simple regularization technique for CNNs. It augments the dataset with partially closed versions of existing samples. This technique forces models to take more consideration of the exact image context. The main difference between the cutout and other dropout

techniques is that neurons are dropped at the input stage instead of layers.

Batch normalization: It changes the input distribution of the hidden layer during training and transforms the input distribution into a standard distribution with a mean of 0 and a variance of 1. Batch normalization enlarges the gradient and helps to eliminate gradient problems. Thus, the convergence of the neural network is faster, and the training takes place in a shorter time. Let P_k be input and Y_k output in a mini-batch, k ∈ [1, 2, ..., K]. Here a mini-batch average μ_B is calculated as:

$$\mu_B = \frac{1}{K}\sum_{k=1}^{K} P_k \quad (3)$$

Variance σ^2_B is calculated as stated in Equation (4).

$$\sigma_B^2 = \frac{1}{K}\sum_{m=1}^{K}(P_k - \mu_B)^2 \quad (4)$$

The normalization value of the input ($P_k^!$) is:

$$P_k' = \frac{P_k - \mu_B}{\sqrt{\sigma_B^2 + \theta}} \quad (5)$$

Here θ is a small positive number. The mini-batch output value is Yk:

$$Y_k = cP_k^! + \alpha \quad (6)$$

It is calculated as Here, c and α parameters are the parameters that can be learned by backpropagation. Layer normalization: It was designed to overcome the disadvantages of batch normalization. With RELU, whose outputs can change a lot, changes in one layer output can cause changes in the next layer inputs. This "covariate shift" problem can be reduced by fixing the mean and the variance of the total inputs in each layer. A layer normalization is calculated on all hidden neurons in the same layer (See Equation (7)).

$$\mu^! = \frac{1}{T}\sum_{n=1}^{T}\beta_n' \quad (7)$$

Here β_n' denotes the collected inputs, and T denotes the number of hidden neurons in a layer.

$$\sigma' = \sqrt{\frac{1}{T}\sum_{n=1}^{T}(\beta_n' - \mu')^2} \quad (8)$$

The main difference between Equation 7 and 8 is that all hidden neurons in a layer share the same terms (μ and σ) in this normalization technique. Unlike batch normalization, it does not impose any restrictions on the size of a mini-batch.

Weight normalization: In this normalization inspired by batch normalization, the weights are re-parameterized, thus improving the optimization problem. Here, the computation of each neuron is treated as a weighted sum of its input features.

$$y = \vartheta(w.x + b) \quad (9)$$

Here, x is an n-dimensional vector of input features, y is neuron output, ϑ(.) is nonlinearity, w is weight, and b is biased. The neural network is trained on each neuron's w, b parameters, and the weight vector is re-parameterized to speed up the optimization convergence.

$$w = \frac{d}{\|f\|} f \quad (10)$$

Here f represents the parameter vector, and d is the scalar parameter.

L2 regularization: The main purpose of this technique is to combine the term regularization with an irregular target.

$$L_\tau(w) = L_{(W)} + \tau\|W\|_2^2 \quad (11)$$

$$L_{(W)} = \sum_{n=1}^{N} L_n\,(y(\,X_n, w, \gamma, \beta)) \quad (12)$$

Here L_n denotes the loss value. In this technique, the weights are forced to reduce.

L1 regularization: This technique, which aims to prevent the overfitting problem by converging the parameters towards 0, destroys the importance of some features.

Dataset

In this study, L2 regularization [31], L1 regularization [32], Dropout, GausianDropout, AlphaDropout, Batch normalization, and Layer normalization regularization methods and their combinations have been used. These methods are frequently used to prevent the overfitting problem. These methods were added to the created base model separately. Their effects on the data set were examined, and the pairwise combinations were compared. The effects of these methods were investigated with the binary cross-entropy loss function. The data set was used as 75 % training set and 15% validation set. In other words, the training set contains 5522 data, and the validation set contains 975 data. The data set consists of

two classes. L1 normalization was added to each layer of the created base model, and training was carried out. At the end of the training, the effect of the base model and the model with normalization added was examined. This situation has been meticulously examined separately for other normalization methods. The best performing binary combination of normalization techniques on The MNIST database of handwritten digits, the Fashion-MNIST datasets, and Cifar10 datasets was investigated.

Experimental results and discussion

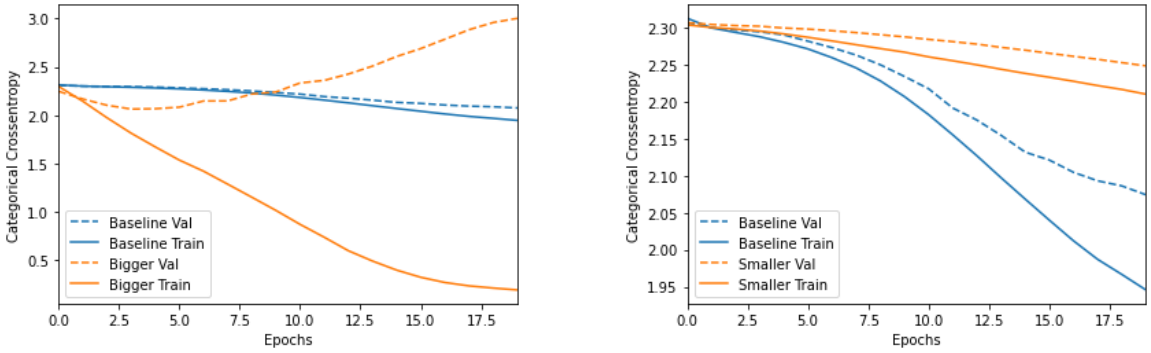
The simplest way to avoid the overfitting problem is to reduce the size of the created model. The size of the model is determined by the number of parameters that can learn (depending on the number of layers of the model and the number of neurons in the layers). In deep learning applications, since the number of learnable parameters determines the capacity of the model, the model with more parameters means that it has more memory capacity. This situation causes the created model to adapt to the training data and not make correct predictions on the test data. On the contrary, if the memory capacity of the created model is limited, the learning process will be troublesome, and the model will have difficulty adapting to the training set.

For this reason, it is necessary to provide a balance (base model) between too much capacity (large model) and less capacity (small model). In experimental studies, three different models were created to understand how to train the model with the appropriate number of epochs, ensure the balance to avoid overfitting in the training set, and provide a better model performance. The created models are named as the large, base, and small models. Information about these models is given in Table 1.

Table1. Details of the models created

Model	number of layers	Optimizer	Loss	Activation	Epoch	Batch size
Small	4	Adam	Binary crosentropy	Relu	20	512
Base	16	Adam	Binary crosentropy	Relu	20	512
Big	512	Adam	Binary crosentropy	Relu	20	512

These models were trained on the training dataset, and the amount of loss in this process was examined. Figure 3 shows the amount of loss during training for the three models.

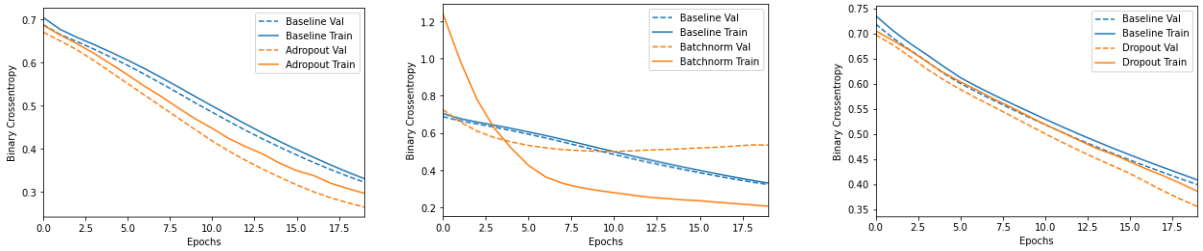


(a) base-big (b) base-small

Figure 3. The amount of loss in the training data set of the models created

Overfitting occurs later than the base model created in a low-capacity network (small model). After the overfit situation occurs, the model performance decreases more slowly than the base model (see Figure 3(b)). On the other hand, overfitting in the large-capacity network has occurred almost from the beginning of the training. In the large-capacity network, the amount of loss approaches zero very quickly. In a large-capacity network, the training data can be modeled more quickly, but this may increase the possibility of overfitting the model. In other words, while a low loss occurs in the training data set, this loss rate may increase in the validation set (see Figure 3(a)). Various normalization techniques were applied to the base model created to prevent the overfitting problem, and the training was carried out. The graph of losses during training for each technique is shown in Figure 4

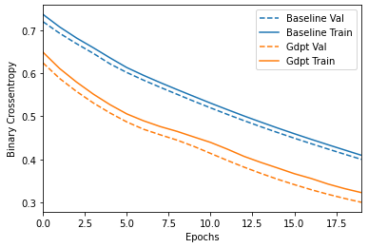
.



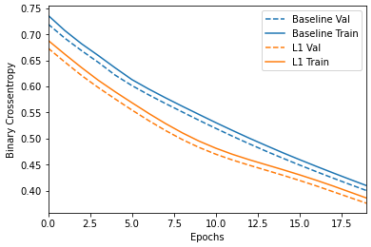
(a) AlphaDropout

(b) Batchnormalization

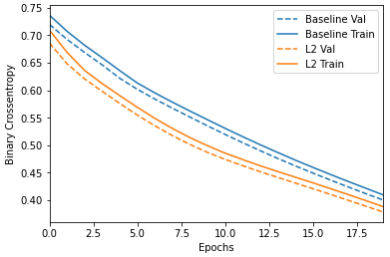
(c) Dropout



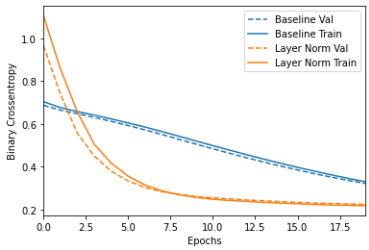
(d) GausianDropout



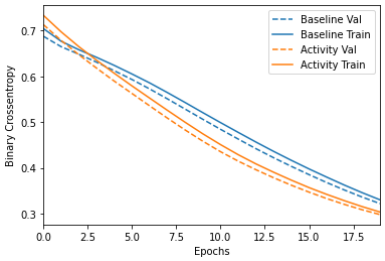
(e) L1 regularızatıon



(f) L2 regularizatıon



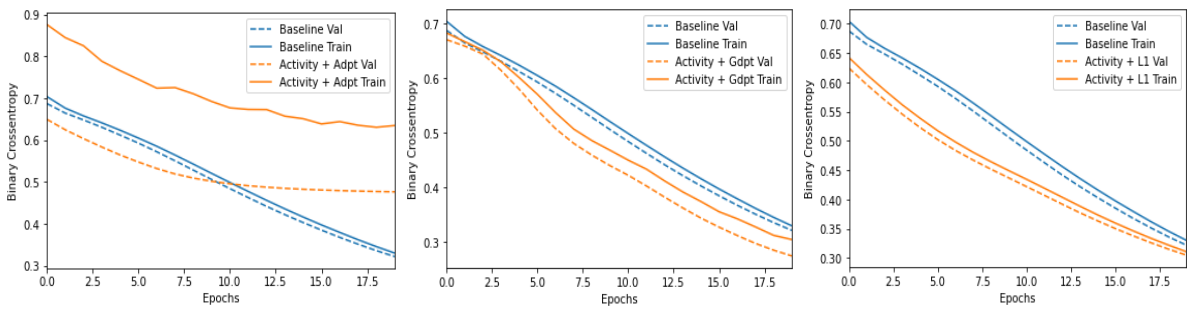
(g) Layer



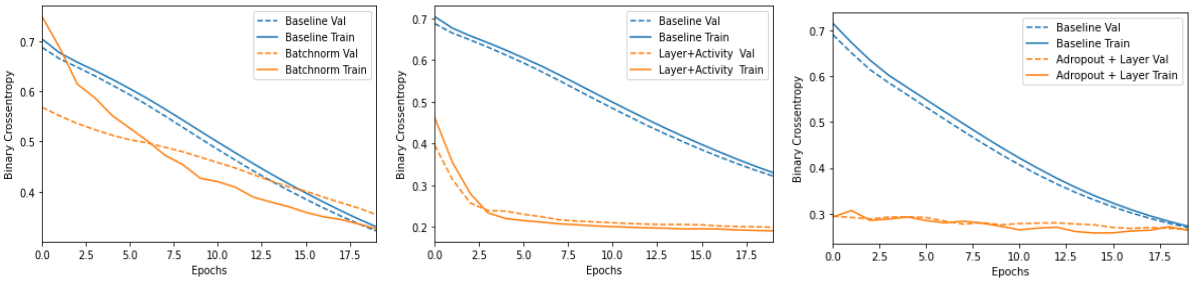
(h) activity

Figure 4. Comparison of normalization methods with the base model

The most common way to prevent overfitting is to add a normalization method called weight normalization to the base model, which allows the weights to take lower values by placing constraints on the network. As a result, the amount of loss in the training and validation data set is seen in Figure 4 (e) (f). Although both base and normalized models had the same number of parameters, both normalization techniques were more resistant to overfitting than the base model. The same is true for other normalization techniques. When normalization techniques are applied, the losses between the training and validation set are reduced, thus increasing the model's accuracy. When these techniques are applied individually to the base model, they significantly reduce overfitting. In the case of using binary combinations of these techniques in this study, the loss during training is shown in Figure 5.



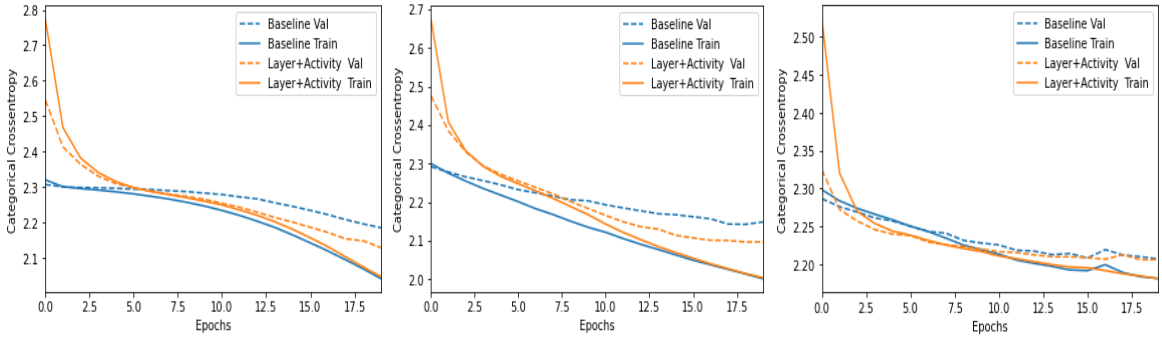
(a) Activitty+AlphaDropout (b) Activitty+GausianDropout (c) Activitty+L1



(d) Batchnorm+Alphadropout (e) Layer +Activity (f) Layer+AlphaDropout

Figure 5. Loss amount of binary combinations of normalization methods

As can be seen in Figure 5, although the Binary Normalization methods used to prevent the problem of overfitting in the model, the Activity (L1-L2) and AlphaDropout pair showed good performance when used alone but did not show the same performance when used together (see Figure 5 (a)). However, when looking at Figure 5(c), the opposite situation is seen. If the binary combinations of normalization methods are applied to the model, Layer+Activity performs better than other combinations. The amount of loss during training in this pair is much lower in the case of dual-use, considering the individual use cases. The effect of the Layer+Activity pair, which performed well in the wine dataset, on different datasets is shown in Figure 6.



(a) Mnist (b) fashion Mnist (c) Cifar10

Figure 6. The effect of Layer-Activity combination on different dataset

As shown in Figure 6, the methods used to prevent overfitting can show different effects on different datasets, even though the same model is used. When both Layer and Activity normalization techniques are used alone in the wine dataset, the amount of loss during training decreases (see Figure (g), (h)). Likewise, binary reduces the amount of loss in the same dataset (see Figure 5(e)). However, it is seen that the amount of loss in the training set does not decrease in the case of using different data sets of the created model (see Figure 6). The methods used to prevent overfitting are mentioned in section 2.2, and in this study, the effect of combinations of these methods on the loss function in the model is explained. In addition, the effect of the normalization combination, which performs better than the other combinations, on different data sets is explained. Normalization combinations made on different datasets may not achieve the same in every dataset.

Conclusion

In this study, different normalization methods and the effect of combinations of these methods on model performance were investigated to prevent overfitting. At the same time, the effect of the better-determined combination on different data sets was also examined. In this context, the use of normalization methods separately or in combination to prevent overfitting can reduce the loss during training. In the study, the Layer-Activity normalization method significantly increases the model performance when compared to other combinations. However, the methods used to prevent overfitting may have different effects in different data sets.

Ethics committee approval andconflict of interest statement

There is no need to obtain permission from the ethics committee for the article prepared.

There is no conflict of interest with any person/institution in the article prepared."

References

[1] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in Selected Papers of Hirotugu Akaike. Berlin, Germany: Springer, 1998, pp. 199–213.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In Advances in neural information processing systems, pp. 1097–1105, 2012

[3] N. Srivastava, G. Hinton, A. Krizhevsky, L. Sutskever and R. Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15.1 2014: pp1929-1958.

[4] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In Proceedings of the 32nd International Conference on Machine Learning (ICML), 2015. Pp 448-456

[5] Y. Wu and K. He, "Group normalization," in European Conference on Computer Vision (ECCV), 2018, pp. 3–19.

[6] J. L. Ba, J. R. Kiros, and G. E Hinton. "Layer normalization". arXiv preprint 2016, arXiv:1607.06450,

[7] D.Ulyanov, V. Andrea, and L. Victor . "Instance normalization: The missing ingredient for fast stylization." arXiv preprint, *2016 arXiv:1607.08022* .

[8] L. Wan, M. Zeiler,, S. Zhang, Y.L. Cun, and R. Fergus,. "Regularization of neural networks using dropconnect". In Proceedings of the 30th International Conference on Machine Learning (ICML-13),2013., pp. 1058–1066

[9] L. Goodfellow, F. Warde, M. David, C. Mehdi, Aaron, and Y. Bengio,. "Maxout networks". Proceedings of the International Conference on Learning Representations (ICLR), 2013, pp 1319- 1327.

[10] R. Tibshirani. "Regression shrinkage and selection via the lasso". Journal of the Royal Statistical Society, Series B,1996 58:267 – 288,

[11] normalization: A simple reparameterization to accelerate training of deep neural networks," in Advances in Neural Information Processing Systems (NeurIPS), 2016, pp. 901–909

[12] S. Akbar , M. Peikari , S. Salama , S.Nofech-Mozes , A. Martel . ""The transition module: a method for preventing overfitting in convolutional neural networks. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, *7*(3), 260-265.. 2019; 7 (3): 260-265.

[13] J. Deng., W. Dong, SocherR. , L.-J.Li, K. Li., and L. Fei-Fei,. "ImageNet: A Large-Scale Hierarchical Image Database". *IEEE conference on computer vision and pattern recognition*. Ieee, 2009. p. 248-255.

[14] R. Girshick, J. Donahue, T. Darrell,, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation". In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pp. 580–587.

[15] P. Cortez., A. Cerdeira., F. Almeida., T. Matos.and J.Reis, "Modeling wine preferences by data mining from physicochemical properties". *Decision support systems*, 2009. *47*(4), 547-553.

[16] Y. LeCun, . "The MNIST database of handwritten digits." *http://yann. lecun. com/exdb/mnist/* 1998

[17] **H.** Xiao K. Rasul and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms". arXiv , 2017,arXiv:1708.07747

[18] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images", Computer Science Department University of Toronto Tech. Rep., 2009,vol. 1, no. 4, pp. 7,.

[19] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in International Conference on Learning Representations (ICLR), 2015

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034

[21] T. Araújo , G. Aresta, E. Castro, J. Rouco, P. Aguiar , C. Eloy, A. Polónia, A. Campilho.."Classification of breast cancer histology images using convolutional neural networks". *PloS One* 12, 2017.

[22] Y.Wei., F. Yang,., and , M. J. Wainwright "Early stopping for kernel boosting algorithms: A general analysis with localized complexities". In Advances in Neural Information Processing Systems, 2017 pp. 6065–6075..

[23] A. Ali,.,J.Z. Kolter,., and R.J. Tibshirani,." A continuoustime view of early stopping for least squares regression". The 22nd International Conference on Artificial Intelligence and Statistics. PMLR, 2019. p. 1370-1378

[24] A.Suggala., A. Prasad., and P.K. Ravikumar,. "Connecting optimization and regularization paths". In Advances in Neural Information Processing Systems,2018 pp. 10608– 10619.

[25] L. Schmidt., S. Santurkar., D. Tsipras., K. Talwar., and A. Madry. "Adversarially robust generalization requires",2018 *arXiv preprint arXiv:1804.11285*.

[26] T. DeVries. and G. W.Taylor, “Improved regularization of convolutional neural networks with cutout”.2017, arXiv preprint arXiv:1708.04552, .

[27] S. Albawi , T.A Mohammed.and S. Al-Zawi, "Understanding of a convolutional neural network," 2017,International Conference on Engineering and Technology (ICET), pp. 1-6,

[28] Ö.F Ertuğrul, E. Acar, E. Aldemir,A. Öztekin A, Automatic diagnosis of cardiovascular disorders by sub images of the ECG signal using multi-feature extraction methods and randomized neural network, 2021, Biomedical Signal Processing and Control, Volume 64, 102260.

[29] E. Acar, Detection of unregistered electric distribution transformers in agricultural fields with the aid of Sentinel-1 SAR images by machine learning approaches, Computers and Electronics in Agriculture, 2020,Volume 175, 105559,

[30] D. M. Hawkins”The Problem of Overfitting”, J. Chem. Inf. Comput. Sci, , 2004, pp 44, 1-12

[31] T. Van Laarhoven,. “L2 regularization versus batch and weight normalization”. arXiv preprint 2017, arXiv:1706.05350..

[32] M.Y.Park., & T. Hastie, L1-regularization path algorithm for generalized linear models. Journal of the Royal Statistical Society: Series B (Statistical Methodology),2007 69(4), 659-677.