Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/37/20230511

Overfitting of CNN model in cifar-10: Problem and solutions

Zhangjie Xia

New York University Shanghai, Shanghai, 200438, China

zx1357@nyu.edu

Abstract. CNN, proposed by Yann LeCun in the 1980s, has gained high attention and extensive research from both the academia and the industry. It has proved to be useful in a wide variety of fields, including image recognition, which aims to enable the computer to identify different objects in digital images. Despite its usefulness, problems like overfitting occur during the training process of a CNN model, which seriously harm the effectiveness of the model. Firstly, an initial CNN model is built to accomplish image recognition based on data from cifar-10. Secondly, after the presence of overfitting during the training and validation of the initial model, 4 methods, including shallower model, L2 regularization, dropout, data augmentation, are proposed to see how they handle overfitting respectively, and comparisons are made between different methods. Thirdly, the last three methods are combined to see how they handle overfitting together. Finally, conclusion and possible future work are presented. As it turns out, L2 regularization, dropout and data augmentation all reduce overfitting with some slight differences, but shallower model and the combined method cause underfitting rather than overfitting.

Keywords: Overfitting, Cifar-10, Image Recognition, Shallower Model, L2 Regularization, Dropout, Data Augmentation.

1. Introduction

Machine learning describes the process during which machines “learn” to solve problems on their own without being explicitly told what to do by human-developed algorithms. It has been widely applied to every aspect of human’s life, like image recognition, speech recognition, large language models, email filtering, computer vision, etc. Over decades, new subsets of machine learning have been developed to handle different types of problems efficiently, like the convolutional neural network (CNN), which is a regularized type of feed-forward neural network which learns feature by itself via filters optimization. Even though CNN is especially helpful in tasks like image recognition, problems like overfitting can dramatically degrade the performance of the model. Overfitting occurs when a model can’t generalize well from old data to new data, therefore causing the model to perform well on the training data but acts poorly on the testing data [1]. In general, it can happen based on three reasons: noise learning from training data, hypothesis complexity and multiple comparison steps which appear in induction algorithms [1]. Even though overfitting can not be completely prevented, researchers are still working hard to minimize the effect it can have on the model performance.

DOI: 10.54254/2755-2721/37/20230511

2. Related work

Various strategies are developed to handle overfitting nowadays, like "early-stopping" strategy, "network-reduction" strategy, "data-expansion" strategy, "regularization" strategy, etc [1]. Each of them works based on different principles, but they all aim to minimize the difference of accuracy and loss occurred after training and validation.

2.1. Shallower model

Overfitting happens when the model is so powerful that it learns features during training process using limited training data and can't generalize to data that it hasn't seen before. Deep neural networks are more powerful than shallow ones because they contain more hyper-parameters and therefore increase the chance to overfit [2]. In this sense, one intuitive approach to avoid overfitting is to build a shallower model, or in other words, a model that has fewer layers and smaller layer size such that it is not capable of extracting features that aren't general.

2.2. Regularization

There are many regularization techniques, like weight decay, dropout, early stopping, adversarial training, etc., and most of them were designed to do general image classification tasks [3]. They can be further split into three groups: "data augmentation" which changes input data, "internal changes" which modifies feature maps of a neural network and "label" that transform labels of input data [4]. But in general, regularization methods all work by creating variable data in different stages of building a CNN [4]. The ones used here are L2 regularization, dropout and data augmentation.

2.2.1. L2 regularization

To reduce overfitting, fewer complex models are designed by making its weights to be small, thus making the distribution of its weights more "regular", which is therefore called "weight regularization". It includes L1 and L2 regularization. In terms of the more common L2 regularization, the model's weights get regularized by adding a penalty term in the form of squared weights of the entire model to the loss function [5]. Applying L2 regularization eventually means every weight in the model will finally decay towards zero linearly and irrelevant components of weight vectors are suppressed, thus not making them sparse [6].

2.2.2. Dropout

Dropout is an effective regularization method for neural networks, which is widely used nowadays. Hinton and his students at the University of Toronto came up with this idea in 2012. Randomly "dropping" units along with their connections from the neural network layers during training is the core of this technique [7]. By doing this, the network develops better generalization ability through randomly protecting some neurons and making the output uncertain about which features it is combining, thus feature detectors are stopped randomly during the training process and no units are codependent with one another in the end [8].

2.2.3. Data augmentation

Data augmentation was introduced to modify and enlarge the existing training dataset, which may be too small on which the model may overfit, by generating artificial images from existing images [9]. It includes geometric transformation, color space transformation, mixing images, random erasing, etc. Besides just simply creating more training data, data augmentation has also been found useful in generalizing from computer models to real-world tasks [10]. Also, since only slight changes are made to the original data, the labels of the newly-created data won't change, which reduces the cost to collect and label data as well.

DOI: 10.54254/2755-2721/37/20230511

3. Method

The images used here are from cifar-10, which contains 60,000 32x32 color images. It is equally divided into 10 different classes: airplanes, automobiles, birds, dogs, frogs, cats, deer, horses, ships, and trucks, as shown in figure 1.

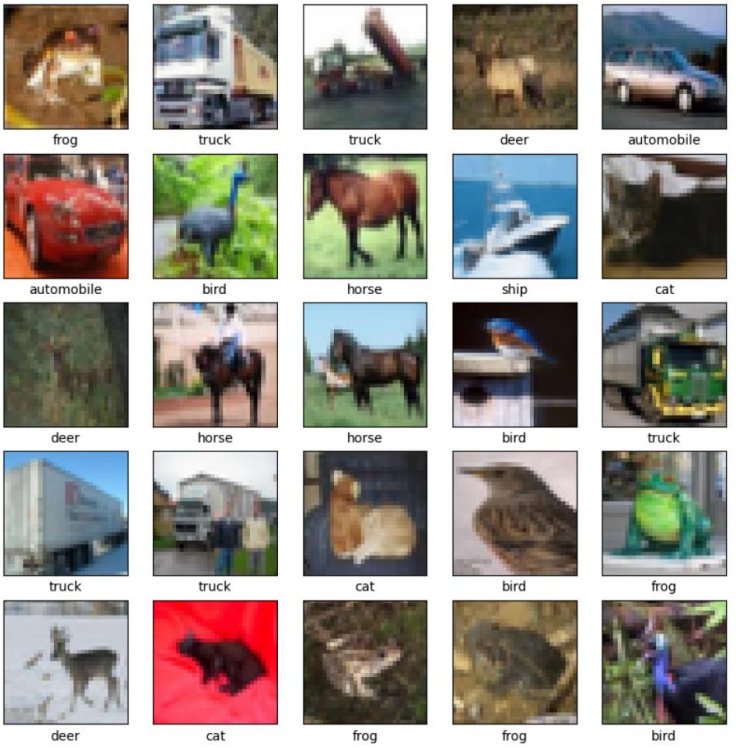

Figure 1. The first 25 images in cifar-10.

There are five training sets and one testing set in cifar-10, and each of these set has 10,000 images that are randomly selected. The testing set has 1,000 images from each class, and the training sets have the rest of the images. Specifically, the testing set is used during validation here. Neural networks tend to yield better results when the inputs are normalized. It can help training since the different features are on a similar scale, which helps to make the gradient descent step stable, allowing models to converge for a given learning rate more quickly. Since the pixel values of images in cifar-10 initially fall in range between 0 and 255, they can be normalized through dividing the pixel values by 255. Figure 2 and figure 3 show the same image from cifar-10 before and after rescaling. The content of this image doesn't change at all, but the pixel values of it can be better fed into the neural network after rescaling.

DOI: 10.54254/2755-2721/37/20230511

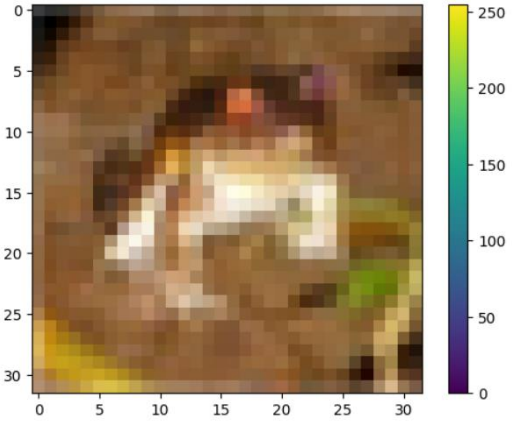

Figure 2. The first image in cifar-10 before rescaling.

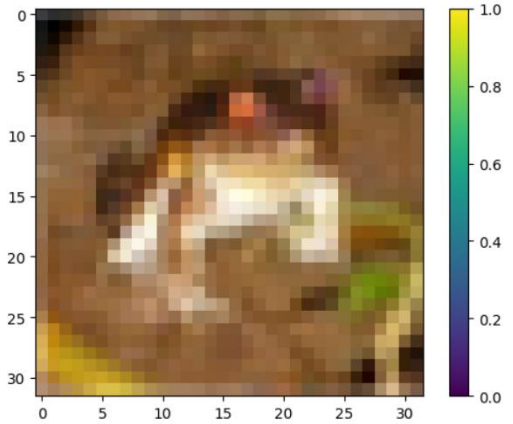

Figure 3. The first image in cifar-10 after rescaling.

Labels of both the training data and testing data are also converted from a class vector to a binary class matrix for later use by calling the to_categorical method. After preprocessing data from cifar-10 and training a CNN model on them, both the accuracy and loss of the training and validation datasets are compared to judge if the model overfits on the training data. Specifically, 2-line graphs are plotted based on the final result: one shows the training and validation loss, and the other shows the training and validation accuracy. The model overfits on the training data if there is a large gap between the training and validation line in both graphs, suggesting that the model has done well on the training data but performs poorly on new data it has never seen. Possible reasons and methods are proposed to analyze and mitigate the effect overfitting has on the model performance. 4 methods are applied at first to address this problem: 1) building a shallower model; 2) using L2 regularization; 3) adding dropout layers; 4) applying data augmentation, including rotation, width shift, height shift and horizontal flip, on the training data. The last three methods are all classified as types of regularization, so finally these three methods are also combined to see how they tackle overfitting together.

4. Implementation and results

Different methods are applied to handle overfitting when it occurs after building an initial CNN model.

4.1. Building initial CNN

Table 1 shows the architecture of the initial CNN built for classifying images in cifar-10.

Table 1. Model architecture of initial model.

Layer	Output Shape	Param #
conv2d	(None, 30, 30, 32)	896
max_pooling2d	(None, 15, 15, 32)	0
conv2d_1	(None, 13,13, 64)	18496
max_pooling2d_1	(None, 6, 6, 64)	0
conv2d_2	(None, 4, 4, 64)	36928
max_pooling2d_2	(None, 2, 2, 64)	0
flatten	(None, 256)	0
dense	(None, 64)	16448
Dense_1	(None, 10)	650
Total params: 73,418		
Trainable params: 73,418		
Non-trainable params: 0		

DOI: 10.54254/2755-2721/37/20230511

After building, adam optimizer is used to compile the model, and categorical_crossentropy is used as metrics to calculate loss and accuracy in order to evaluate model performance. Then the model is trained on the training data and validated on the testing data for 25 epochs. Final results are plotted in two-line graphs, figure 4 and figure 5, to give a more intuitive understanding of the results, showing loss and accuracy of training and validation process. In each graph, a large gap can be found at the end of the two lines. This indicates that the differences of both loss and accuracy become larger as training and validation processes go on, therefore resulting in overfitting for this initial model. A possible reason is that this model is too complicated and powerful so that it learns features from the training data that are not applicable to the testing data. So different methods are used to address this problem.

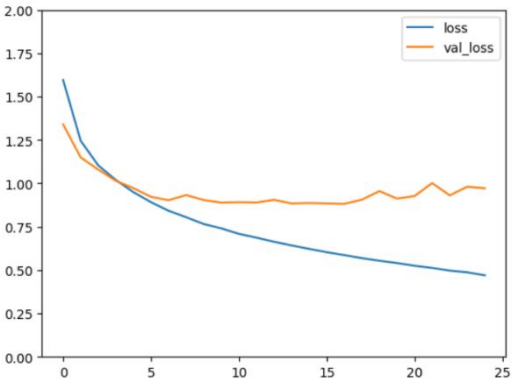

Figure 4. Loss during training and validation of initial model.

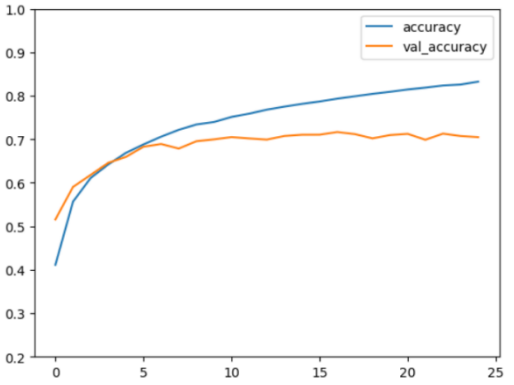

Figure 5. Accuracy during training and validation of initial model.

4.2. Shallower model

The first approach is to build a shallower model which limits the model's ability to generate features from training data that may not be general. Table 2 shows the architecture of this shallower model. The first 6 layers of the initial model are removed, and the model has less parameters to train, meaning it's less complex. This model is compiled, trained and validated the same way as the initial model.

Table 2. Architecture of shallower model.

Layer	Output Shape	Param #
flatten	(None, 3072)	0
dense	(None, 16)	49618
dense_1	(None, 10)	170
Total params: 49,338		
Trainable params: 49,338		
Non-trainable params: 0		

Similarly, figure 6 and figure 7 are plotted to give a more direct understanding to the results. However, in these two graphs, the training line and validation line almost coincide with each other, suggesting the loss and accuracy of both training and validation are quite similar throughout the whole process, which means there's no overfitting.

DOI: 10.54254/2755-2721/37/20230511

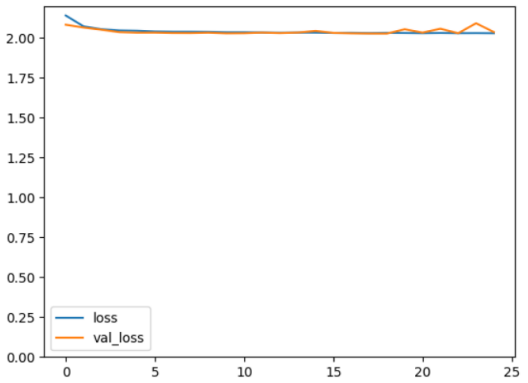

Figure 6. Loss during training and validation of shallower model.

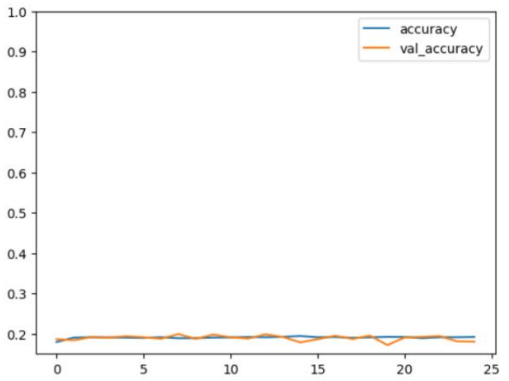

Figure 7. Accuracy during training and validation of shallower model.

However, high loss and low accuracy means the model performs poorly in both training and validation, which leads to another problem: underfitting. It is the opposite of overfitting, and it occurs when the neural network is not trained enough, and the network isn't capable of getting a low error on the training data [11]. In this case, the shallower model is too shallow that it can't extract important characteristics from the training data, thus performing badly after training; and the limited features it extracted also can't be applied to the validation data thoroughly, therefore resulting in bad validation results. In a nutshell, a more complex model needs to be built to handle underfitting.

4.3. L2 regularization

The second method is to use L2 regularization to apply a penalty on layers. kernel_regularizer is used on each Conv2D layer of the initial model where L2 regularization factor is set to 0.001. Table 1 shows the architecture of this regularization model. Then this model is compiled, trained and validated the same way as the initial model. figure 8 and figure 9 are plotted to illustrate loss and accuracy of training and validation. Compared with figure 4 and figure 5, even though there's still a gap at the end of training and validation in each graph, undoubtedly the one for regularization model is smaller than that for initial model, meaning that the overfitting problem is mitigated effectively by using L2 regularization. And the underfitting problem doesn't exist since we get relatively high accuracy and low loss in the end.

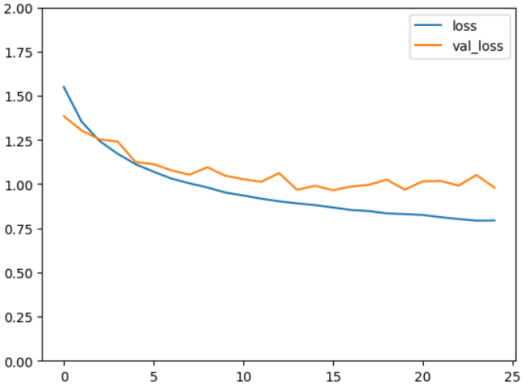

Figure 8. Loss during training and validation of regularization model.

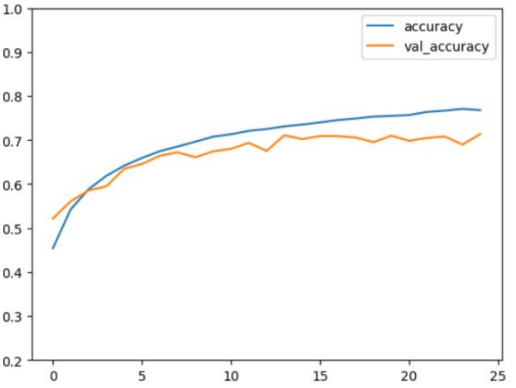

Figure 9. Accuracy during training and validation of regularization model.

DOI: 10.54254/2755-2721/37/20230511

4.4. Dropout

The third method applied is to use dropout. One dropout layer is added after each MaxPool2D layer with dropout rate set to 0.5, which means half of the inputs are dropped out randomly. Table 3 shows the architecture of this dropout model. Then this model is compiled, trained and validated the same way as the initial model.

Table 3. Architecture of dropout model.

Layer	Output Shape	Param #
conv2d	(None, 30, 30, 32)	896
max_pooling2d	(None, 15, 15, 32)	0
dropout	(None, 15, 15, 32)	0
conv2d_1	(None, 13,13, 64)	18496
max_pooling2d_1	(None, 6, 6, 64)	0
dropout_1	(None, 6, 6, 64)	0
conv2d_2	(None, 4, 4, 64)	36928
max_pooling2d_2	(None, 2, 2, 64)	0
dropout_2	(None, 2, 2, 64)	0
flatten	(None, 256)	0
dense	(None, 64)	16448
Dense_1	(None, 10)	650
Total params: 73,418		
Trainable params: 73,418		
Non-trainable params: 0		

Loss and accuracy during training and validation are shown in figure 10 and 11. The gap between training and validation is still relatively small compared with that of the initial model, meaning the overfitting is handled well by dropout. And underfitting is still not a problem because of relatively high accuracy and low loss.

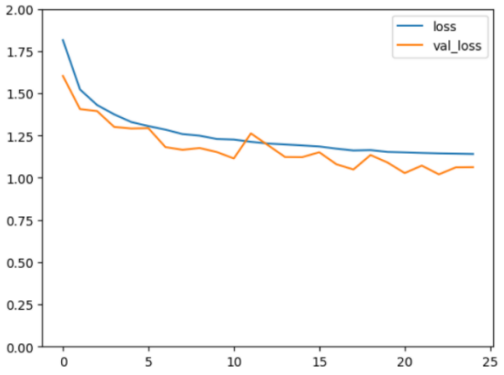

Figure 10. Loss during training and validation of dropout model.

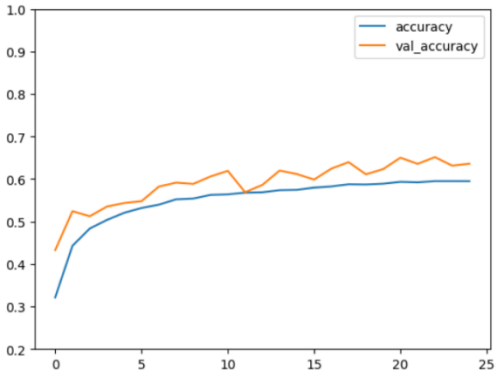

Figure 11. Accuracy during training and validation of dropout model.

However, two things worth noticing here is that (1) in general, the model is less effective than L2 regularization model since loss of this model is higher than L2 regularization model and accuracy of this model is lower than L2 regularization model for both training and validation, (2) the model performs better in validation than in training because of higher accuracy and lower loss during validation than during training. One possible reason for (1) is that the model is relatively small,

DOI: 10.54254/2755-2721/37/20230511

therefore the overall performance will be improved by averaging learning mode when there's a large number of sub-model and each of them must be different from each other, leading to better predictive accuracy of L2 regularization than dropout [12]. One possible explanation for (2) may be the application of regularization terms only when training the model, therefore increasing the training loss. But during validation process, the loss function only contains prediction error, which, in general, causes a lower loss than the training set.

4.5. Data augmentation

The fourth method is to apply data augmentation, specifically, rotation, width shift, height shift and horizontal flip, to the training data. To be specific, rotation rotates the images a certain degrees, width shift shifts the image to the left/right, height shift shifts the image up/down and horizontal flip flips the image upside down. Figure 12 shows the first 10 images before and after applying several data augmentation techniques randomly.

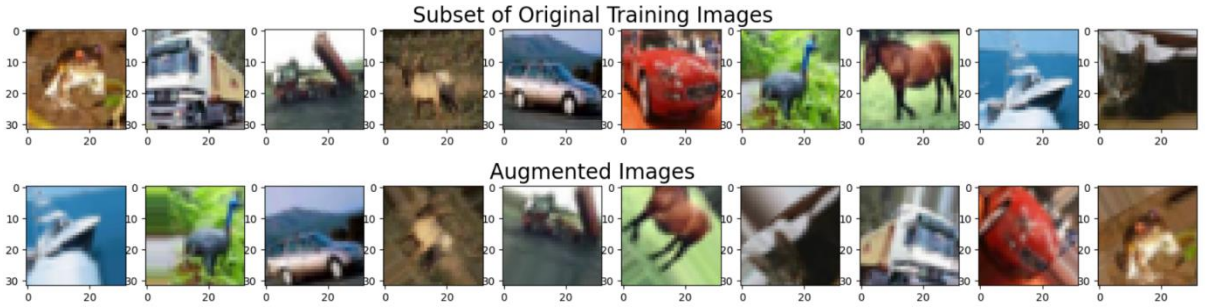

Figure 12. Images before and after data augmentation.

Table 1 shows the architecture of this model, and this model is compiled, trained and validated the same way as the initial model. Figure 13 and 14 demonstrate the loss and accuracy during training and validation. Loss and accuracy don't vary much during training and validation, meaning the overfitting is handled well by data augmentation. The line graphs are similar to the ones for using the dropout method, and relatively high accuracy and low loss suggests that underfitting doesn't occur.

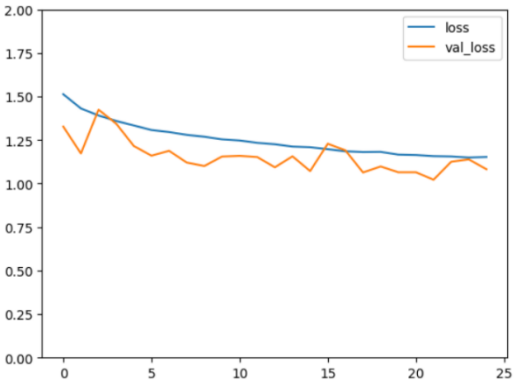

Figure 13. Loss during training and validation of data augmentation model.

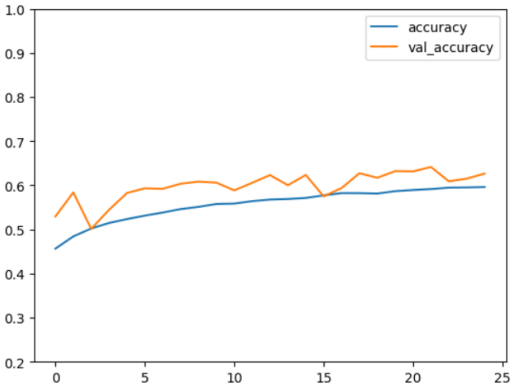

Figure 14. Accuracy during training and validation of data augmentation model.

4.6. Combining method

Method 2 and 3 and 4 are combined to see how they address overfitting together since all of them can be classified as regularization techniques. Table 3 shows the architecture of this combined model, and this model is compiled, trained and validated the same way as the initial model. Figure 15 and 16 demonstrate loss and accuracy during training and validation, and underfitting occurs here because of

DOI: 10.54254/2755-2721/37/20230511

the low accuracy and high loss in both training and validation. Instead of the model being too simple as described before, a possible explanation is that over-regularization causes underfitting because the model is too constrained by regularization that prevents it from capturing underlying patterns from data [13]. Therefore, regularization parameters need to be reduced to deal with underfitting.

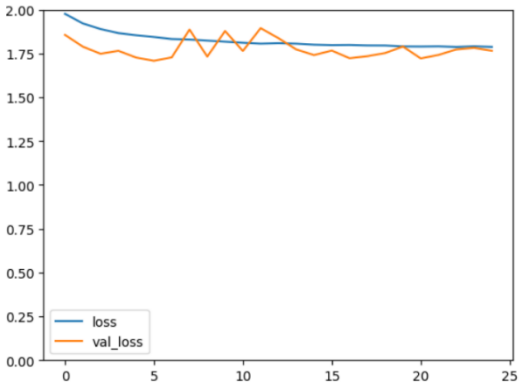

Figure 15. Loss during training and validation of combined model.

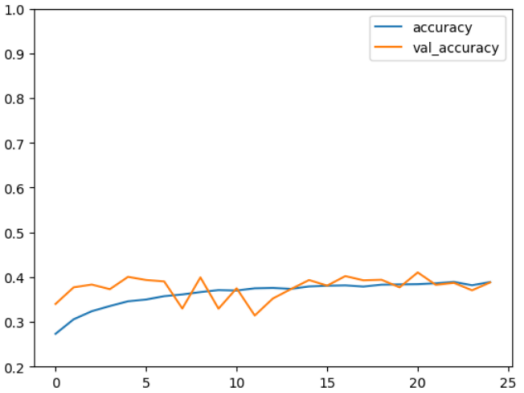

Figure 16. Accuracy during training and validation of combined model.

5. Conclusion

Based on the experiment, after overfitting occurs when building the initial CNN model to classify images in cifar-10, 4 methods are proposed at first to reduce its effect on the model performance: using a shallower model, L2 regularization, dropout and data augmentation. While building a shallower model did prevent overfitting, it causes another problem: underfitting, which means it performs poorly on both training and validation data, therefore suggesting this model is too simple to extract underlying features of the training data, therefore should not be used. The last three methods all work well in reducing the effect of overfitting, and none of them causes underfitting as well. Based on the final result after training and validation, L2 regularization should be considered as the most useful way to eliminate overfitting with a higher accuracy and lower loss in the end. Finally, the last three regularization methods are combined to see if they can get the best result. However, underfitting occurs again in this scenario, and one possible reason is over-regularization. In the future, more methods, like early stopping and batch normalization, can be studied, and overfitting can be studied when using other machine learning methods, like GNN, RNN, etc., therefore giving readers a broader picture of why overfitting occurs and how it can be handled efficiently in general.

References

[1] Ying X 2019 Journal of physics: Conference series vol 1168 p 1
[2] Bejani M M and Mehdi G 2021 A systematic review on overfitting control in shallow and deep neural networks. Artificial Intelligence Review. 54(2021) 1-48
[3] Li Z, Konstantinos K and Ben G 2020 IEEE transactions on medical imaging vol 40 p 1065-1077
[4] Santos C F G D and João P P 2022 Avoiding overfitting: A survey on regularization methods for convolutional neural networks. ACM Computing Surveys (CSUR). 54.10s (2022) 1-25
[5] Rezaeezade A and Lejla B 2022 Regularizers to the Rescue: Fighting Overfitting in DeepLearning-based Side-channel Analysis
[6] Gupta S et al. 2017 Data Science and Analytics: 4th International Conference on Recent Developments in Science, Engineering and Technology vol 799 p 363-371
[7] Srivastava N et al. 2014 Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research. 15.1 (2014)1929-1958

DOI: 10.54254/2755-2721/37/20230511

[8] Qian L et al. 2020 IEEE Access vol 8 p 62830-62840
[9] Shorten C and Taghi M K 2019 A survey on image data augmentation for deep learning. Journal of big data 6.1 (2019)1-48
[10] Perez L and Jason W 2017 The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621 (2017)
[11] Zhang H, Lin Z, and Yuan J 2019 11th international conference on wireless communications and signal processing (WCSP) p 1-6
[12] Phaisangittisagul E 2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS) p 174-179
[13] Quinto B 2020 Introduction to Machine Learning. Next-Generation Machine Learning with Spark: Covers XGBoost, LightGBM, Spark NLP, Distributed Deep Learning with Keras, and More. 1-27