Evaluating Methods to Prevent Overfitting in Neural Networks Through Complexity Reduction

Chayan Bhattacharjee[1], Pooja Amin[2]
[1,2]*Department of Information Technology, Patkar Varde College*

***Abstract* – Manages deep networks of various parameters during the training and testing phase. As the number of parameters increases, these networks acquire the ability to adapt to different types of data records, contributing to incredible strength. However, this ability can also make neuronal networks sensitive, which are prone to over-adaptation. Over--Several strategies can be used to tackle the problem of adaptation. In this article, we will explore various methods to prevent the model from limiting model complexity, data expansion, weight normalization, occurrence, and early stopping.**

***Keywords*- Overfitting, Neural network generalization, Regularization, Model simplification**

I. INTRODUCTION

A wide range of parameters, from thousands to millions, provides considerable flexibility to networks of neurons, allowing them to absorb a wide range of complex data records. This unique ability has advanced in many areas that have pose challenges in the traditional age of machine learning, such as image detection, object detection, and natural language processing. However, this important advantage can also represent a potential drawback. Inadequate control over the model's learning process can lead to over-regulation. This is a scenario in which neuronal networks are over-tuned to training data, hindering their ability to generalize and create accurate predictions. It is important to understand the underlying causes of this problem and to investigate strategies to ensure the effective design of neuronal networks.

In practice, the detection of over-adjusting in models can be a major challenge. It is not uncommon for a trained model to be used for production before you realize the problem. The true validity of a model can only be assessed when exposed to new data. Therefore, during the training phase it is important to simulate the actual conditions as closely as possible. To achieve this, we recommend splitting the data record across three segments. Training statements, development rates (also known as cross-validation or holdout), and test sets. The model is only learned from the training statement during the holdout set to monitor progress and gain knowledge for model optimization. The test set is reserved for evaluation of the power of the model after completing the training process. Using completely new data allows for a fair assessment of the effectiveness of the algorithm. Only then can we be confident that decisions made during the learning process will lead to more effective solutions. Historically, the data standard employed a division of 60:20:20. In the age of big data, where data records can contain millions of entries, these fixed conditions are no longer suitable. Ultimately, the corresponding split depends on the size of the data record. For example, if millions of entries are available, the 98:1:1 division could have a greater advantage. Our development and test sets must be large enough to ensure high reliability in the performance of the model. Several approaches, such as recording additional data, are widely used and consistently effective. In contrast, methods such as forms of regularization require differentiated understanding and considerable expertise. Excessive and strict limitations in neural networks can interfere with learning functions. Now let's take some of the most frequently used strategies, over adapt, and investigate the reasons for their effectiveness.

II. METHODS

1. Constraining Model Complexity

A model trained with almost unlimited numbers of examples ultimately reaches a point where its learning ability is stable. Its capabilities are sufficient to overcome the training dataset. By reducing the power of the model, the likelihood of over adaptation can be minimized. The complexity of the neural network model that defines its capacity is determined by its

architecture, not only by its weights, but also the number of nodes and layers. As a result, it is possible to reduce the complexity of neuronal networks and reduce over adaptation in two ways.

a. Change the network architecture (weight volume) to adjust the complexity of the network.

b. Changes in network complexity (weight values) due to variations in network parameters.

As a result, the network cannot remember all data points because it has fewer parameters to learn, and instead forces them to generalize. During the implementation of this method, the number of layers to eliminate the appropriate size of the network must be decided.

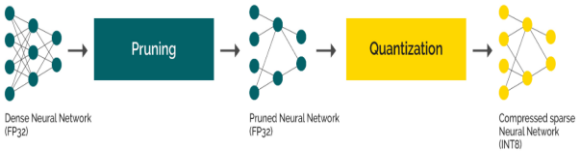

Fig. 1 Neural Networks are compressed through pruning

When employing this technique to address the problem, it is essential to consider the input and output dimensions of the different layers within the neural network. During the implementation of this method, it is necessary to ascertain the following:

- The number of layers to eliminate
- The appropriate size of your network
- The quantity of neurons required in each layer

Although there is no definitive guideline for answering these questions, several well-known strategies exist, which are outlined as Grid Search which uses Grid Search Cross-Validation to find the best number of neurons or layers to take out and trimming which is used to enhance our overfitted model, we can gradually remove nodes or connections until it performs well on new datasets, aligning with the model development process. This method mainly focuses on simplifying the neural network to lower the chances of overfitting.

2. Data Augmentation.

One effective method for mitigating overfitting is to expand the training dataset. As previously mentioned, a smaller training dataset allows the network to exert more influence over the data. To enhance the training dataset's size, specifically by increasing the number of images, data augmentation can be employed. This approach is a straightforward way to diversify the dataset and enlarge the training set. Various prevalent techniques for image augmentation encompass flipping, translation, rotation, scaling, brightness adjustment, and the introduction of noise, among others. In practical situations, acquiring large volumes of data can be labour-intensive and time-consuming, making the collection of new data an impractical solution. This technique is shown in the below diagram.

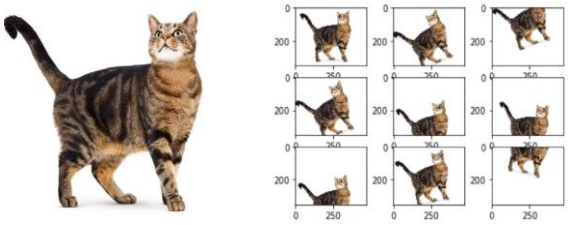

Fig. 2 Data Augmentation Techniques

Data augmentation facilitates the generation of multiple analogous images, which enables the network to learn from diverse representations of the same category of objects observed from various perspectives. This strategy not only enhances the dataset but also reduces the risk of overfitting. By incorporating additional data, the model is less prone to memorizing every sample, thereby encouraging improved generalization. The images obtained through Data Augmentation will feature a lion presented in various ways, including:

- a rotated perspective,
- an upside-down view, or
- a cropped section highlighting the lion's mane.

The application of the latest augmentation technique, known as cut-out, enables the network to associate the defining feature of male lions—their mane—with the corresponding class. Data augmentation primarily seeks to expand the training dataset, thereby reducing the likelihood of the network overfitting to the entire dataset, which consists of both original and augmented images. This process promotes better generalization. However, it may also lead to an increase in overall training loss, as the network may struggle to make accurate predictions for the augmented images. Consequently, the optimizer, the algorithm tasked with optimization, adjusts the network to enhance its ability to recognize generalized patterns within the training dataset. A visual representation of this discussion is provided below:

Before Data Augmentation:

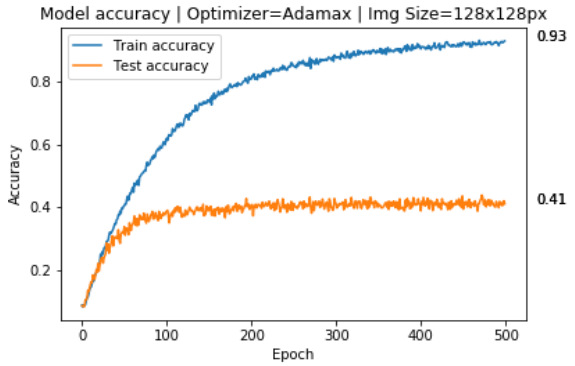

Fig. 3 Overfitting: high train-test gap
Source: Stackexchange.com

After Data Augmentation:

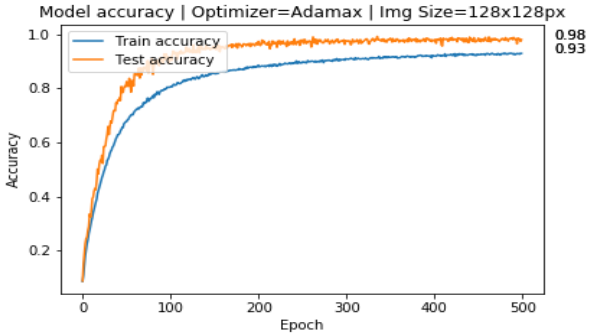

Fig. 4 Optimal generalization: test exceeds training.
Source: Stackexchange.com

3. Weight Regularization.

This approach to minimizing FIT overnight emphasizes the stabilization of congested networks by incorporating a "weight" parameter that regulates excessively high weight values within the network. As a rule, excessive models are problematic, as even a small change at the beginning can lead to significant variations in the output. Weight control deals with the fact that large weights impose punishment. This encourages the optimization algorithm to decrease these values, thereby enhancing network stability and overall performance. This method mustn't alter the network's architecture; only the weight values are modified. To prevent excessive adjustments, penalties or constraints are incorporated into the loss function. The concept of regularization acts as a framework for optimization algorithms, such as probabilistic gradient descent, which not only minimizes the difference between predicted and actual values but also aims to reduce the loss function.

L1 Regularization (Lasso Regression):

The cost function incorporates the total of the absolute values of the weights (Wj). This approach promotes sparsity within the model, leading to the elimination of certain weights and facilitating feature selection.

Formula: $\text{Cost} = \Sigma (yi - \Sigma xijWj)^2 + \lambda \Sigma |Wj|$

L2 Regularization (Ridge Regression):

The cost function includes the sum of the squares of the weights (Wj^2). This shrinks the weights towards zero but doesn't typically eliminate them entirely. It helps to reduce the impact of less important features.

Formula: $\text{Cost} = \Sigma (yi - \Sigma xijWj)^2 + \lambda \Sigma Wj^2$

The above two equations represent L1 & L2 Weight Regularization Techniques. The Regularized Equation consists of two key elements:

(i) The initial element signifies the discrepancy between the actual target and the predicted target, commonly referred to as the loss function.

(ii) The following component represents the weight penalty, commonly referred to as the regularization term.

Types of Regularization:

- L1 regularization: This technique incorporates a penalty on weights, which is determined by the aggregate of the absolute values of the weights present in the network.
- L2 regularization: This method involves introducing a penalty on weights that is derived from the sum of the squares of the weights within the network.

Table 1. Types of Regularization

L1 Regularization	L2 Regularization
The penalty component relies on the absolute magnitudes of the model's parameters	he penalty component relies on the squares of the model's parameters
It results in sparse solutions, where certain parameters are reduced towards zero	This leads to solutions that are not sparse, indicating that the model employs all available parameters.
This method is particularly sensitive to outliers. .	This approach is resilient to outliers
It identifies a selection of the most significant features.	This model incorporates all features into the model.

Optimization is non-convex.	Optimization is convex.
The penalty component exhibits reduced sensitivity to correlated features.	Additionally, the penalty term exhibits heightened sensitivity to correlated features
The presence of numerous correlated features in high-dimensional datasets is beneficial.	It is particularly beneficial for high-dimensional datasets with numerous correlated features
This approach is commonly referred to as Lasso regularization.	This method is commonly referred to as Ridge regularization.

L2 regularization is preferable for complex data, as it effectively captures the underlying patterns, while L1 regularization is suitable for simpler datasets. Consequently, selecting the appropriate regularization method hinges on the specific problem we aim to address.

4. Dropout.

Dropout is a regularization method aimed at reducing overfitting in neural networks. In contrast to conventional techniques like L1 and L2 regularization, which modify the cost function, Dropout changes the structure of the network itself. Throughout the training process, it randomly disables a portion of the neurons, except for those in the output layer, during each iteration. This can also be achieved by assigning a probability known as the Dropout Rate (typically set at 0.5) to each neuron, thereby temporarily removing them from the calculations. As training progresses, neurons with the highest probability of being dropped are removed, effectively creating a smaller network with each epoch. The variability in input values promotes a balanced strategy within the network, ensuring that it does not become excessively dependent on particular features, which in turn minimizes bias and noise.

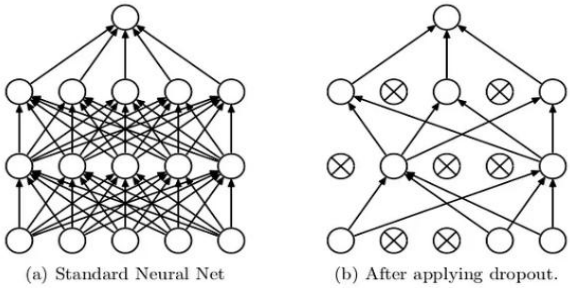

Fig. 5 Dropout: an easy method to stop overfitting in neural networks
Source: towardsdatascience.com

This approach is sometimes known as the Ensemble Technique for Neural Networks. It emulates the training of several neural networks by selectively disabling different clusters of neurons. Consequently, each of these networks may overfit in unique manners, and the overall impact of dropout is to mitigate overfitting. This technique to prevent overfitting has proven to reduce overfitting to a variety of problem statements that include,

- Image classification,
- Image segmentation,
- Word embedding,
- Semantic matching etcetera, etc.

5. Early Stopping.

Early stopping is a regularization approach employed in the training of models that utilize iterative methods such as Gradient Descent. Given that all neural networks rely on optimization algorithms such as gradient descent for learning, early stopping is relevant across various problems. This method aids in reducing overfitting by adjusting the model to more accurately correspond with the training data during each iteration. It is widely recognized that prolonged training can result in overfitting within neural networks. Initially, the model's performance on the test set improves, but beyond a certain threshold, further adjustments to enhance the model's fit to the training data can actually increase generalization error. Early stopping serves as a guideline for determining the optimal number of iterations to execute before the model starts to overfit. This concept is illustrated in the diagram below.

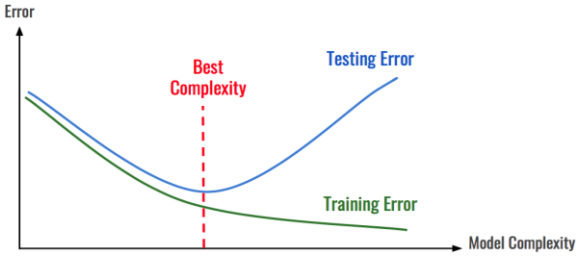

Fig. 6 The model on the left is too simple, It overfits to the right.

It is clear that after multiple iterations, the test error starts to increase while the training error keeps decreasing. This suggests that the model is undergoing overfitting. To mitigate this problem, we stop the training process as soon as overfitting is identified. The

network parameters at this point of early termination are considered the best configuration for the model. To further decrease the test error beyond this early stopping stage, the following strategies may be implemented. Decreasing the learning rate. Use a learning rate scheduler algorithm would be recommended.

• Use a different Optimization Algorithm.

• Use weight regularization techniques like L1 or L2 regularization.

III. CONCLUSION

To summarize, in our detailed discussion, we delved into the critical concept of overfitting and its common occurrence within the realm of neural networks, emphasizing the need for effective strategies to counter this phenomenon during the training process. We discussed five essential strategies that have been shown to be effective in mitigating overfitting in neural networks: reducing model complexity to improve generalization, utilizing early stopping to avoid excessive training, harnessing data augmentation to expand the training dataset, applying regularization techniques to impose limits and prevent overly large parameter values, and strategically using dropouts to bolster the model's resilience. By comprehensively understanding and implementing these methods, practitioners of neural networks can significantly enhance the model's capacity to generalize to new, unseen data, thereby improving overall performance and dependability. Recognizing the importance of these techniques is vital, as they provide the groundwork for developing more robust and precise neural network models, ensuring that the issue of overfitting is effectively managed during the training process. Ultimately, adhering to these recommendations and continuously refining training methodologies will enable practitioners to navigate the intricacies of neural network training with greater assurance and skill, facilitating the advancement of more effective and influential AI applications across various fields.

REFERENCE

[1] Srivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya, & Salakhutdinov Ruslan. (2014). Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929–1958.

[2] Seitz, P., & Schmitt, J. (2023). Alternating Transfer Functions to Prevent Overfitting in Non-Linear Regression with Neural Networks. Journal of Experimental and Theoretical Artificial Intelligence.

[3] Iscen, A., Tolias, G., Avrithis, Y., & Chum, O. (2018). A Simple Way to Prevent Neural Networks from Overfitting. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 15, 7642–7651

[4] Piotrowski, A. P., & Napiorkowski, J. J. (2013). A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling. Journal of Hydrology, 476, 97–111.

[5] Wang, M., He, L., Lin, J., & Wang, Z. (2022). Rethinking Adaptive Computing: Building a Unified Model Complexity-Reduction Framework With Adversarial Robustness. IEEE Transactions on Neural Networks and Learning Systems, 33(4), 1803–1