Investigating Overfitting in Convolutional Neural Networks

1. Introduction

Neural networks are a form of machine learning models, roughly inspired by how the human brain processes information. They have become increasingly prevalent in our developing world, providing powerful and unique solutions to a wide variety of problems. However, with power also comes risk, as training a neural network too much on a small dataset can cause overfitting. Overfitting occurs when a model learns the training data too well by essentially memorizing it, causing it to perform well on data in its training set but poorly on unseen data. While it might seem that simply increasing a model's size would always boost accuracy by allowing it to capture more patterns, extra parameters often give the network enough capacity to store the training set outright, allowing it to overfit. Smaller networks, by contrast, are forced to find the underlying patterns in the data because of their limited size. In this study, we investigate whether increasing the number of parameters in a convolutional neural network leads to greater overfitting when trained on a small dataset, as measured by the difference between training and test accuracy.

2. Statistical Question

Does increasing the number of parameters in a fixed CNN architecture cause a greater difference between training and testing accuracy?

Hypotheses:

$$H_0 : \beta = 0$$

$$H_a : \beta > 0$$

Where β is the true slope of the population least-squares regression line that relates number of parameters of the model to the difference in accuracy of the model on the train dataset and the test dataset (train - test).

3. Data Collection

We trained 15 sets containing 3 convolutional models each on a randomly selected small subset of size 100 images from the CIFAR-10 dataset, each model having the same architecture (*see figure 1 below*). Models within a set had the same number of parameters, and separate sets had different numbers of parameters, uniformly ranging from approximately 1 million to 50 million. On the initialization of each model in each set, the values of the parameters are randomly set, so each model can be seen as randomly selected from all possible models of their respective size and architecture before training. We controlled the parameter count by multiplying the amount of filters for the Conv2D layers and the amount of neurons for the hidden Dense layers by a scalar factor, n.

Model Architecture

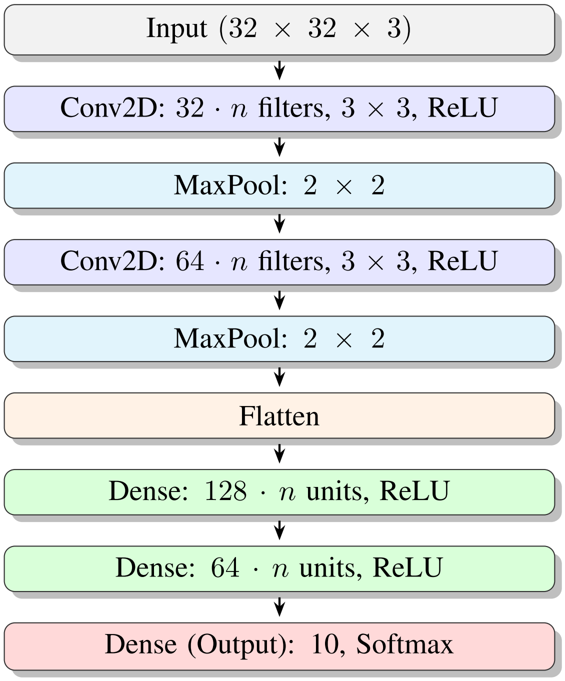
Input $(32 \times 32 \times 3)$
Conv2D: $32 \cdot n$ filters, 3×3, ReLU
MaxPool: 2×2
Conv2D: $64 \cdot n$ filters, 3×3, ReLU
MaxPool: 2×2
Flatten
Dense: $128 \cdot n$ units, ReLU
Dense: $64 \cdot n$ units, ReLU
Dense (Output): 10, Softmax

Figure 1: General Architecture of Our Models

For each set, each model was trained for 60 epochs on the randomly selected subset of 100 images and then tested on the full test set of CIFAR-10. The difference between the train and test accuracy (train - test) was then computed for each model, and the average of that value was computed for each set of models by adding up all of the differences and then dividing by the size of the set. Then, the data were graphed on a scatter plot with the x-axis as parameter number and the y-axis as the calculated average difference between train and test accuracy for each set of models.

Data Display

Data Analysis

Conclusion

Reflection