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 test data (data that were not in the training set). 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 when the training set is small, 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 its accuracy on the training dataset and the test dataset.

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 100 convolutional models each on the same randomly selected small subset of 100 images from the Canadian Institute For Advanced Research - 10 dataset (CIFAR-10), which consists of 32x32 color images labeled with one one of ten classes (e.g., plane, boat, etc). Each model had the same architecture, the only difference between them being the number of parameters (*see figure 1 below*). The training dataset was constructed by randomly selecting 10 of each class of image to ensure that it is representative of the entire dataset. The number of filters and neurons in each model were varied so that the model sizes are roughly uniform, ranging from approximately 1m - 50m parameters. The architecture and the ratio of layer sizes were also kept the same for each model, making sure the only difference between them is the number of parameters. 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, ensuring the experiment is statistically valid. 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

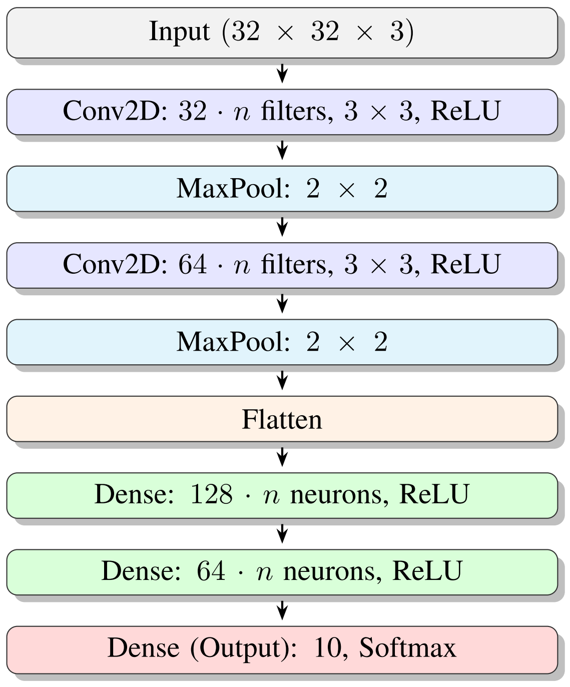
Input $(32 \times 32 \times 3)$
Conv2D: $32 \cdot n$ filters, 3×3, ReLU
MaxPool: 2×2
Conv2D: $64 \cdot n$ filters, 3×3, ReLU
MaxPool: 2×2
Flatten
Dense: $128 \cdot n$ neurons, ReLU
Dense: $64 \cdot n$ neurons, ReLU
Dense (Output): 10, Softmax

Figure 1: General Architecture of Our Models

Each model was trained for 100 epochs on the randomly selected subset of 100 images and then tested on the full test set of the CIFAR-10 dataset, made up of $10,000$ images. The difference between the train and test accuracy (train - test) was then computed for each model and graphed on a scatter plot with the x-axis as parameter count and the y-axis as the difference between train and test accuracy for each model.

Data Display

Data Analysis

We conducted a t-test for the population slope β of the least-squares regression line that relates the number of parameters in a model trained on a 100-image subset of the CIFAR-10 dataset to the difference in accuracy between training and test data (Train Accuracy $-$ Test Accuracy).

We first checked the conditions required to perform a t-test:

Linear: The scatterplot of the data shows a clear linear trend, and the residual plot shows no remaining curved pattern. Therefore, the linearity condition is satisfied.

Independent: Each model was trained independently with no interaction between runs. Thus, the independence condition is met by the design of the experiment.

Normal: The dotplot of the residuals shows no strong skew and no outliers, suggesting the residuals are approximately normally distributed.

Equal Variance: The residual plot shows shows no noticeable $<$ or $>$ shape, so the equal variance condition is satisfied.

Random: The parameters of the model were randomized on initialization, so each model can be seen as randomly selected from the population of models of the same size and structure.

Calculations

Conclusion

Reflection