Investigating Overfitting in Convolutional Neural Networks

1 Introduction

Neural networks are a form of machine learning models, 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 150 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 each labeled with one of ten classes (e.g., plane, boat, etc). The training dataset was constructed through stratified random sampling by randomly selecting 10 of each class of image, ensuring that it is representative of the entire dataset. The number of filters and neurons were varied between models in a way such that the sizes of the models (in parameter count) were roughly uniformly distributed, ranging from approximately 1m to 25m parameters. The architecture and the ratio of layer sizes were kept consistent between each model, ensuring that the only difference between them was the number of parameters (*see Figure 1 below*). On the initialization of each model, the values of the parameters for that model were 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. All of the above precautions aimed to eliminate confounding with other variables, ensuring that any change in the accuracy of the models was actually caused by the change in parameter count. 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.

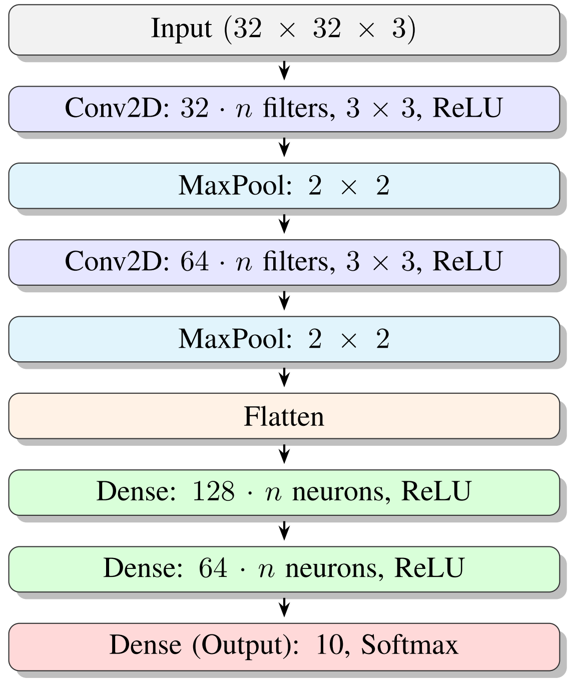

Figure 1: General Architecture of Our Models

Each model was trained for 100 epochs on the randomly selected subset of 100 images, then evaluated 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.

4 Data Display

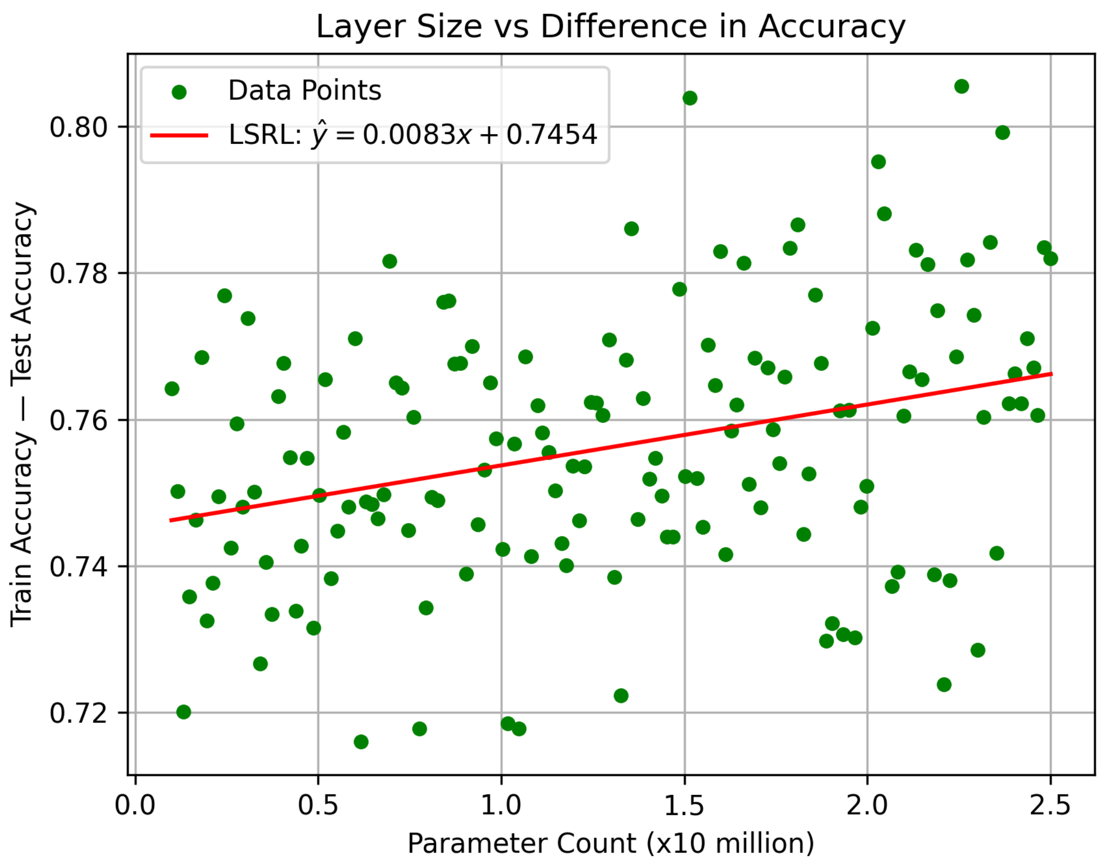

Figure 2: Scatterplot of Parameter Count vs. Difference in Train and Test Accuracy (Train - Test)

Predictor	Coef	SE Coef
Constant	0.7454	
Param Count (x10 mil)	0.0083	0.0020
$S = 0.01703$		R-Sq = 10.37%

Table 1: Regression Summary

Based on the scatter plot (*Figure 2*) and the correlation coefficient of $r = \sqrt{0.1037} = 0.3220$ (*from Table 1*), there appears to be a weak, positive, linear relationship between parameter count and difference between train and test accuracy (train - test). There appear to be a few possible high outliers above $x = 1.5$ million parameters and a few possible low outliers above $x = 0.5$ million parameters.

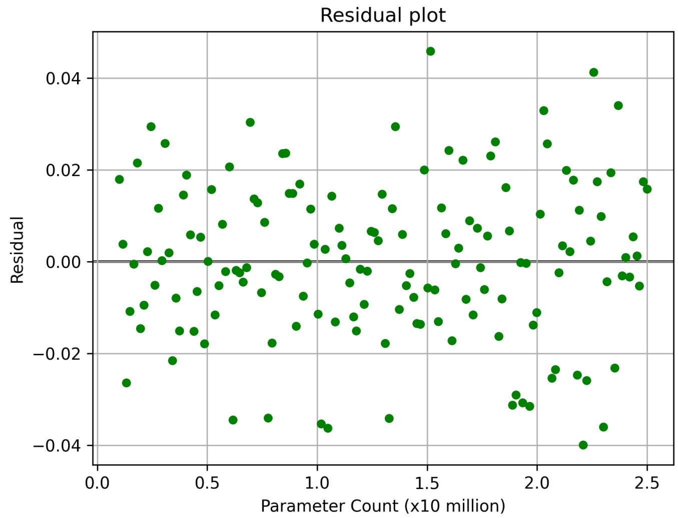

Figure 3: Residual Plot (Parameter Count vs. Residuals)

Because the residual plot (*Figure 3*) appears to have no form and because the standard deviation of the residuals S (*from Table 1*) is small (0.01703), the graphical display seems to indicate that the null hypothesis will be rejected.

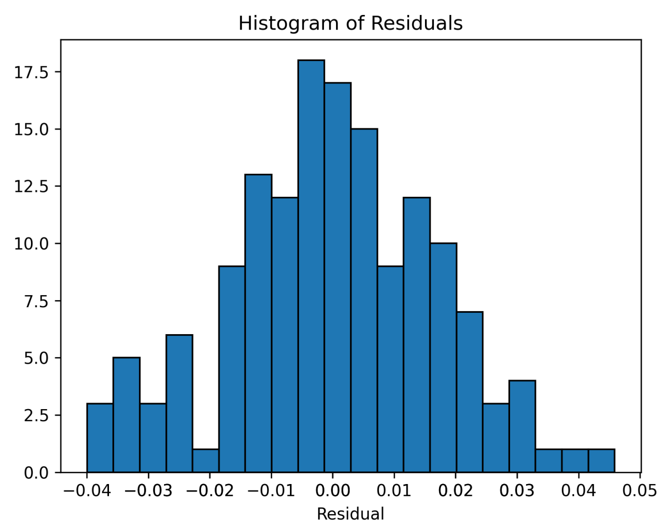

Figure 4: Histogram of Residuals

The histogram of the residuals appears to be approximately symmetric, with a median around 0 and a range of $0.05 - (-0.04) = 0.09$. There does not appear to be any strong skew or outliers.

5 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 randomly selected 100-image subset of the CIFAR-10 dataset to the difference between its accuracy on training and test data (Train Accuracy $-$ Test Accuracy).

5.1 Condition Check

5.1.1 Linear

The scatterplot of the data (*Figure 2*) shows a clear linear trend, and the residual plot (*Figure 3*) shows no remaining curved pattern. Therefore, the linearity condition is satisfied.

5.1.2 Independent

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

5.1.3 Normal

The histogram of the residuals (*Figure 4*) shows no strong skew and no obvious outliers, so the normality condition is met.

5.1.4 Equal Variance

The residual plot (*Figure 3*) shows shows no noticeable $<$ or $>$ shape, so the equal variance condition is satisfied.

5.1.5 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.

5.2 Calculations

5.2.1 Standard Error

We used the following equation to calculate the Standard Error SE_β of the slope β:

$$\text{SE}_\beta = \frac{S}{S_x\sqrt{n-1}}$$

We need the standard deviation of the residuals and of the parameter counts to find SE_β, which we found like this:

$$\text{S} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-2}}$$

$$\text{S}_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

After plugging in all of the data from the 150 models, we got $S = 0.01703$ and $S_x = 0.6972$. Now we can use those to find SE_β:

$$\text{SE}_\beta = \frac{0.01703}{0.6972\sqrt{150-1}} = 0.002008$$

5.2.2 t-statistic

And now we can find the t-statistic, which represents how many standard deviations away our sample slope b is from the population slope β on the sampling distribution, assuming that $\beta = 0$:

$$\text{t} = \frac{b}{SE_\beta}$$

$$\text{t} = \frac{0.008312}{0.002008} = 4.139$$

5.2.3 p-value

Now that we have the t-statistic, we can use it to find the p-value, which represents the probability of getting a test statistic of 4.139 or higher, assuming that the population slope β is 0. Because we have 150 data points, we have $150 - 2 = 148$ degrees of freedom. Thus, we calculate the p-value by finding the amount of area under a t-distribution with 148 degrees of freedom, above $t = 4.139$. The graph below demonstrates this:

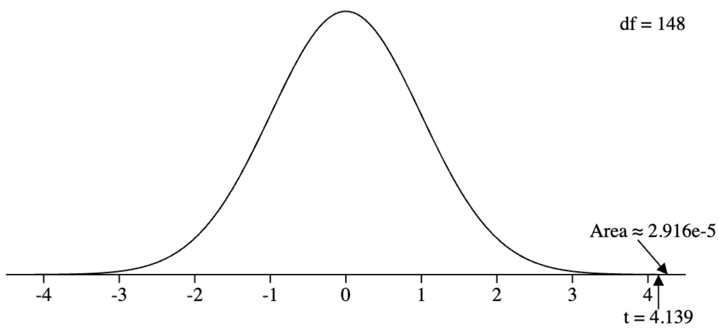

Figure 5: p-value calculation on a t-distribution with df = 148

Based on the area under the curve in *Figure 5* above $t = 4.139$, our p-value is $2.916 \cdot 10^{-5}$.

6 Conclusion

The p-value of $2.916 \cdot 10^{-5}$ indicates that, assuming that the population slope $\beta = 0$, there is around a 0.002916% probability of getting a sample slope as great or greater than our sample slope of 0.008312, in a sample of 150 randomly selected models from the population of convolutional models of our structure and of size 1 million parameters to 25 million parameters, trained on a randomly selected subset of 100 images from the CIFAR-10 train dataset and tested on the full test dataset of CIFAR-10. Because the p-value of $2.916 \cdot 10^{-5}$ is less than $\alpha = 0.05$, we reject the null hypothesis, which states that the population slope β of the least-squares regression line that relates parameter count to difference in train and test accuracy (train - test) is 0, in the population defined above. The data do provide convincing evidence that the population slope β is greater than 0, and that there is a positive correlation between parameter count and difference between train and test accuracy (train - test) in the population defined above. Thus, for that population, we can conclude that increasing the number of parameters in the model does indeed cause overfitting.

7 Reflection

In our project, we aimed to determine whether increasing the parameter count of convolutional models of a specific structure caused the model to overfit, or essentially memorize the train dataset and perform well on images from that set but poorly on newly introduced images. We ensured equal representation of images of different classifications by randomly sampling by strata, with each strata being a group of images with the same classification. By keeping the structure of the models and the dataset on which they were trained on constant, as well as randomizing the parameters of the models on initialization, we prevented confounding with other variables, allowing us to establish a cause-and-effect relationship between parameter count and difference between train and test accuracy (train - test). Through a t-test for the population slope, we found that there is a positive relationship between parameter count and difference between train and test accuracy (train - test), and thus that increasing parameter count increases the difference and causes overfitting.

We could have potentially made a Type I error, because we rejected the null hypothesis. If we had made a Type I error, that would mean that we found convincing evidence that increasing the parameter count of the model between 1m and 25m parameters caused overfitting, when in reality it didn't. A potential consequence of this would be that we would only use models with less than 1 million parameters, believing that models larger than 1 million parameters would perform worse on the test dataset, losing the potential of getting better accuracy by using larger models.

However, our design included many limitations that could be addressed in future experiments. We only trained models with 1m to 25m parameters, which means we can only generalize to models of these sizes. In the future, we could conduct tests with a wider range of parameter sizes, which would allow us to generalize our results to a wider range of model sizes or potentially reveal a new trend with larger parameter sizes.

Another major limitation was the compute time required. Despite training relatively simple models on a small dataset, it took over two hours to train all 150 models. If we wanted to scale the experiment up with more complex models or larger datasets, the compute time required could make the experiment impractical. In the future, we could optimize the experiment by training the models for less epochs or using less models, although this would decrease the power of the test.

Finally, our results are only applicable to models of the specific architecture we used on a 100 image subset of the CIFAR-10 dataset, which is not useful to many people training neural networks. If we wanted the test to be more useful in helping people decide the size of their models, we could perform an experiment on different model architectures and datasets, allowing a broader generalization.

Appendix: Raw Data

Table 2: Raw Data: Parameter Count vs. Train–Test Accuracy Difference

Param.	Acc.	Param.	Acc.	Param.	Acc.	Param.	Acc.	Param.	Acc.	Param.	Acc.
991 995	0.7642	1 156 636	0.7502	1 314 120	0.7201	1 481 652	0.7358	1 653 866	0.7463	1 817 930	0.7685
1 964 108	0.7325	2 122 007	0.7377	2 279 711	0.7495	2 443 067	0.7769	2 612 075	0.7425	2 779 778	0.7594
2 928 578	0.7481	3 083 068	0.7738	3 247 185	0.7501	3 409 755	0.7267	3 574 350	0.7405	3 736 748	0.7334
3 908 963	0.7632	4 058 757	0.7677	4 231 690	0.7548	4 387 487	0.7339	4 537 211	0.7427	4 689 447	0.7547
4 872 926	0.7316	5 030 648	0.7497	5 190 882	0.7655	5 353 628	0.7383	5 518 886	0.7448	5 686 656	0.7583
5 825 435	0.7481	5 997 767	0.7711	6 162 250	0.7160	6 304 105	0.7488	6 472 707	0.7484	6 620 711	0.7465
6 793 468	0.7498	6 942 372	0.7816	7 119 248	0.7650	7 274 427	0.7643	7 455 458	0.7449	7 611 411	0.7603
7 760 207	0.7178	7 947 148	0.7343	8 108 136	0.7494	8 261 687	0.7490	8 413 707	0.7760	8 570 111	0.7762
8 724 930	0.7676	8 884 187	0.7677	9 041 805	0.7389	9 203 915	0.7700	9 364 332	0.7457	9 529 295	0.7531
9 692 511	0.7650	9 860 327	0.7574	10 026 342	0.7423	10 183 690	0.7185	10 352 393	0.7567	10 480 242	0.7178
10 654 715	0.7686	10 813 533	0.7413	10 990 748	0.7619	11 122 467	0.7582	11 298 745	0.7555	11 465 740	0.7503
11 644 706	0.7431	11 780 277	0.7401	11 950 780	0.7537	12 133 478	0.7462	12 271 857	0.7536	12 445 868	0.7624
12 586 011	0.7623	12 773 485	0.7606	12 951 004	0.7709	13 093 955	0.7385	13 269 953	0.7223	13 399 368	0.7681
13 544 767	0.7861	13 727 548	0.7464	13 874 711	0.7629	14 055 863	0.7519	14 204 772	0.7547	14 391 938	0.7496
14 526 700	0.7440	14 678 075	0.7440	14 864 381	0.7778	15 017 502	0.7522	15 155 155	0.8039	15 348 460	0.7520
15 504 047	0.7453	15 643 907	0.7702	15 836 226	0.7647	15 977 573	0.7830	16 119 548	0.7416	16 278 987	0.7585
16 422 292	0.7620	16 623 490	0.7814	16 768 300	0.7512	16 930 907	0.7684	17 077 047	0.7480	17 277 956	0.7671
17 425 583	0.7586	17 591 340	0.7540	17 740 297	0.7658	17 889 882	0.7834	18 099 850	0.7866	18 250 940	0.7443
18 402 658	0.7526	18 572 987	0.7770	18 726 035	0.7677	18 879 711	0.7298	19 034 015	0.7322	19 246 091	0.7612
19 344 507	0.7307	19 500 695	0.7613	19 657 511	0.7302	19 814 955	0.7481	19 973 027	0.7509	20 131 727	0.7725
20 291 055	0.7952	20 451 011	0.7881	20 670 818	0.7372	20 832 261	0.7392	20 994 332	0.7605	21 157 031	0.7665
21 320 358	0.7831	21 484 313	0.7655	21 648 896	0.7812	21 814 107	0.7388	21 918 875	0.7749	22 085 111	0.7238
22 251 975	0.7380	22 419 467	0.7686	22 567 754	0.8055	22 736 428	0.7818	22 905 730	0.7743	23 008 186	0.7285
23 178 495	0.7603	23 349 432	0.7842	23 520 997	0.7418	23 693 190	0.7992	23 866 011	0.7622	24 018 998	0.7663
24 193 001	0.7622	24 367 632	0.7711	24 542 891	0.7671	24 654 011	0.7606	24 830 295	0.7835	25 007 207	0.7820