5/20/2021

MET CS 767 Assignment 3T: CNN’s

*Aidan Duffy*

Copy the implementation [here](https://colab.research.google.com/drive/1Yg-NXKlYzfvv9jI2MCxWZErovpR57_YC?usp=sharing) to your Google drive. Systematically modify this code in four ways, attempting to improve the output, and report the results, using this Word file as a template. Since the accuracy of the given implementation is already high, consider reducing the size of the CIFAR training set—or substituting it so that the baseline implementation leaves more percentage room for improvement. You are free to combine changes but each of the sections below should contain at least one new change. If necessary, show changes that make the result worse, with your explanation.

Please leave the gray text and the headings unchanged.

# The first way I modified the code to attempt improvement

## 1.1 Description of what I did and reason this could be an improvement (one paragraph)

The first alteration in an attempt to improve the model was I doubled the number of epochs from ten to twenty. When epochs were set to 10, the accuracy had just surpassed 70%. While an increase in the number of epochs can lead to overfitting, the model was continuing to improve when it was at ten. Therefore, I wanted to ensure that additional epochs would not lead to a more accurate model because it would have strong overfitting. I noticed that the value of the loss of the model continued to increase steadily with additional epochs, even with a fairly flat accuracy level. Thus, I set it to 20 epochs.

## 1.2 Comparison of the result with the original output, with explanation

The initial accuracy rate with an epoch set to 10 was just over 70%. Once I set the number of epochs to 20, it ended up being 68.42%. This means that the model began to overfit as reflected by the decrease in the accuracy values.

## 1.3 URL of the Colab code

The link will be the same for all of them, and it is [here](https://colab.research.google.com/drive/1qpsGPkGxv2nsfVAEJyu6JLLKt55i-lWQ?usp=sharing). Here is a screenshot of the change:

Text, application

Description automatically generated

# The second way I modified the code to attempt improvement

## 2.1 Description of what I did and reason this could be an improvement (one paragraph)

For the second improvement attempt, I reverted back to epochs, then for the setup of the model, I add an additional parameter for each of the convolution layers: the *kernel\_initializer.* I used the following value for this parameter: “regularizers.l2(1e-4)”. After researching, I realized that adding the kernel initializer parameter would assist in the for preventing overfitting. This, in turn, ought to help increase the overall accuracy of the model.

## 2.2 Comparison of the result with the original output, with explanation

The initial accuracy rate with an epoch set to 10 was just over 70%. Once I added the kernel\_initializer parameter with the aforementioned value, the accuracy ended up at 69%, remaining fairly consistent. I plan to keep this change for future alterations as my research indicated this should be helping.

## 2.3 URL of the Colab code

The link will be the same for all of them, and it is [here](https://colab.research.google.com/drive/1qpsGPkGxv2nsfVAEJyu6JLLKt55i-lWQ?usp=sharing). Here is a screenshot of the additions in the code:

Text, letter

Description automatically generated

# The third way I modified the code to attempt improvement

## 3.1 Description of what I did and reason this could be an improvement (one paragraph)

The next improvement I attempted was I added additional convolution layers and increased some of their number of kernels. I added an extra layer of 32 kernels, then added two 128 layers. I also completed batch normalization and added dropout layers. In tangent with the kernel regularizer, the dropout layers should also aid in the prevention of overfitting. The less overfitting should also help increase the overall accuracy. Also, upon researching, I found that the BatchNormalization function should transform the layers so the mean activation is close to 0 and the stddev is close to 1. Finally, the additional convolution layers should help increase the accuracy of the system.

## 3.2 Comparison of the result with the original output, with explanation

The accuracy from the second alteration started at 69%. After these changes, an accuracy of 70% was already achieved after the fourth epoch. After ten total epochs finished, the accuracy was 75.7%. The highest accuracy was 75.9% after the eighth epoch, indicating some overfitting was possible. As I mentioned, the dropout layers helped with regularization, performance, lowering overfitting. The BatchNormalization() function helped with efficiency since it helps gain similar performance levels with in less time. In my research, it also helps with what is known as “internal covariate shift,” or the change in the network activations distribution that result from the change in network parameters throughout the training process, by stabilizing the learning process and reducing the necessary amount of epochs as mentioned earlier.

## 3.3 URL of the Colab code

The link will be the same for all of them, and it is [here](https://colab.research.google.com/drive/1qpsGPkGxv2nsfVAEJyu6JLLKt55i-lWQ?usp=sharing). A screenshot of the changes are below:

Text

Description automatically generated

# The fourth way I modified the code to attempt improvement

## 4.1 Description of what I did and reason this could be an improvement (one paragraph)

My final improvements were as follows: I added padding = same to each of the convolution layers, I increased the filters on the dense layers on top after everything was flattened from 64 to 128, and I specified directly the parameters for the Adam optimizer (The learning rate was set to 0.0003 and decay to 1e-6). The dense layers was increased to reflect the largest convolution layer though I maintained the dense layer that had the same number of filters as the number of classes overall. I added the padding so that there is enough padding such that that the output and input have the same dimensions. Finally, I wanted more direct input over the Adam optimizer’s parameter for the sake of experimentation to get the optimal accuracy values. Finally, I decreased the number epochs to 3 in order to address some of the overfitting concerns.

## 4.2 Comparison of the result with the original output, with explanation

As noted above, the initial result before this step was 75.7%. After the attempted improvement, the accuracy was 81.5%. This was steady throughout most of the epochs. It seems all of the additions were able to prevent overfitting, achieve superior performance, although they did increase runtime. Th learning rate was low so the whole model was able to slowly converge on the correct answer, the normalizations helped, the dense layers worked with the other convolutions layers well, and the padding helped increase the accuracy of the model. The decrease in the number of epochs certainly helped prevent overfitting as well. When I attempted it with more, the accuracy began to decrease after a few epochs.

## 4.3 URL of the Colab code

The link will be the same for all of them, and it is [here](https://colab.research.google.com/drive/1qpsGPkGxv2nsfVAEJyu6JLLKt55i-lWQ?usp=sharing). Screenshots of all changes are below:

# Text Description automatically generatedText, letter Description automatically generatedText Description automatically generated

# References

Show that you used a wide variety of resources by listing them below and clearly indicating in the body above where you used. Make sure to use proper referencing in your paper. We suggest using APA format, but other formats are fine as long as they clearly distinguish your work from work of others in your response. In general, observe the stated plagiarism rules.

[1] <https://medium.com/@icecreamlabs/3x3-convolution-filters-a-popular-choice-75ab1c8b4da8>

[2] <https://www.kaggle.com/sid2412/cifar10-cnn-model-85-97-accuracy>

[3] <https://appliedmachinelearning.blog/2018/03/24/achieving-90-accuracy-in-object-recognition-task-on-cifar-10-dataset-with-keras-convolutional-neural-networks/>

[4] <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/>

[5] <https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/>

[6] <https://developers.google.com/machine-learning/glossary/#convolutional_neural_network>

# Evaluation

