Ben Prescott
MSDS 458 2021SU
7/24/2021

A.2 Second Research/Programming Assignment

Abstract

This research involves the development of multiple neural network architectures, as well as convolutional neural networks, to understand the difference in feature extraction and performance between them. A total of ten experiments were conducted, consisting of various neural network architectures that were trained using the standard CIFAR-10 dataset. Minor preprocessing was required and completed prior to training the networks, and the results of each network's performance visualized and reviewed in a table for easy comparison. The training times of each network architecture were measured, to aid in understanding the tradeoff between resource/time requirements and model performance.

This research also included visualizing a convolutional model's outputs (model #3) to visualize what features of an image the model's layers were identifying. The goal of this experiment was to determine if the model was picking up relevant features when compared with human-based insight of the original image.

Introduction

The purpose of the ten different experiments was to understand the shift from single hidden layer models to more complex multi-layer models, or deep neural networks, to understand how different architectures and layer types impact what the model is identifying and its overall performance. This approach is application to computer vision/image recognition applications, focusing on a deeper understanding of how and what the networks are using to determine their output. Standard training/tuning of the networks may be sufficient in achieving good model performance but understanding how or what the model is using in the training process is important to transparency and understandability.

The first two experiments leveraged standard deep neural networks, consisting of two and three fully connected hidden layers, respectively. Experiments three and four shifted from fully connected architectures to convolutional networks, each consisting of two and three convolutional and max pooling layers, respectively. Experiment five contained multiple sub-experiments, with the first four revisiting the original four experiments but including regularization techniques. The last two sub-experiments used a modified convolutional network architecture, followed by a dramatic increase in convolutional filters to determine if the performance gain was worth the noticeable increase in training time.

An assumption was made prior to starting this research, in that the convolutional networks would outperform the fully connected networks. This assumption was made due to both previous experience with using convolutional networks, as well as the method of feature extraction that convolutional models provide over fully connected models.

Literature Review

There are many articles and papers published with more in-depth understanding of convolutional networks and comparisons against the more 'traditional' fully connected networks. There is a post on Jason Brownlee's website machinelearningmastery.com titled *How to Visualize Filters and Feature Maps in Convolutional Neural Networks* that reviews the process for visualizing feature maps, like what was done in this research.

More in-depth and more current research also exists focusing on the role of individual units within both fully connected and convolutional networks. A paper titled *Understanding the Role of Individual Units in a Deep Neural Network* by Bau et al, focuses on CNN and GAN networks for scene classification and scene generation, specifically. In this paper research is done for "network dissection", reviewing what each layer in a CNN is "looking at" or understanding, such as color, materials, objects, etc. This provides more clarity into what part of a picture each layer is focusing more on, such as a city scene and the first convolutional layer's units picking up more of a car than buildings or pedestrians.

Methods

This research started by loading the CIFAR-10 dataset using the TensorFlow Datasets library. The CIFAR-10 dataset consists of 60,000 images, each 32 pixels wide by 32 pixels tall across three color arrays (Red, Green, Blue). The dataset was split into train and test sets, consisting of 50,000 images and 10,000 images, respectively. The train data was further split to provide 10,000 images for the validation set that would be used to measure performance during model training, resulting in a training set of 40,000 images, a test set of 10,000 images, and a validation set of 10,000 images.

With the CIFAR-10 dataset being well known, we understand that there are ten classes of images – airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. A dictionary was created using these labels and temporarily converting the numeric labels to strings to help visualize a subset of data for review. Rescaling of the x values of train/test/validation was also done, to help with reducing variance and values to between 0 and 1. No processing or transformation of the y data (classification labels) was done, requiring the use of sparse categorical crossentropy rather than using labels that were one-hot encoded.

Experiment #1 and #2 consisted of fully connected networks, with #1 having two hidden layers and #2 having three hidden layers. Each hidden layer leverage 16 nodes to keep the node counts consistent. Both networks were then trained using the normalized x values and standard y values, with the first experiment result in very poor performance (~10% accuracy) and the second experiment showing an increase in performance to ~30% accuracy, while still very poor overall.

Experiment #3 and #4 shifted to convolutional network architectures, with #3 consisting of a CNN with 2 convolutional and 2 max pooling layers, and #4 with 3 convolutional and 3 max pooling layers. Both CNN architectures were followed by a layer to flatten the data prior to sending it through a fully connected network. The first CNN model showed a greater increase in accuracy, achieving ~68% accuracy. The second CNN model, which added an additional convolutional/max pooling layer, further increased the accuracy to ~71%.

Experiment #5 had many sub-experiments: 5A, 5B, 5C, 5D, 5E, and 5F. The first four sub-experiments (A-D) involved revisiting experiments 1-4, adding regularization methods to the network architectures. For 5A and 5B, L2 kernel regularization (ridge regression) was added to the hidden layers, helping in reducing the risk of overfitting by penalizing large weight values. 5C and 5D revisited the CNN models, adding in dropout layers as a form of regularization. Each set of convolutional and max pooling layers was followed by an added dropout of 50%, setting half of the nodes values to zero to help reduce overfitting. 5E and 5F resembled the architecture of 5D, adjusted to include another fully connected layer and a modified filter count for the convolutional layers, with 5F showing a significant increase in filter and node count. The fully connected layers also had L2 regularization applied in both 5E and 5F.

Each network's training time (in seconds) was recorded to compare the resource usage and timing against the model's performance. The intent was to determine if a noticeable change to an architecture, that drastically increased training time, was justified by the increase in accuracy. These training times, along with each model's maximum training and validation accuracy, were added to a table for review.

To wrap up the research, the third experiment's model (CNN with 2 convolutional and 2 max pooling layers) was used to predict the category of an image of a cat, and the model's activation values retrieved to review visually to identify what features of the image each convolutional and max pooling layer identified.

Results

The results of the different experiments helped showcase the benefit that a convolutional neural network can bring over the 'standard' fully-connect network. Shifting from a deep network with a few fully connected hidden layers to a convolutional network with the same number of convolution/max pooling layers, the results determined that model's accuracy would more than double. There are several factors that may contribute to this, but even more simplistic CNNs are able to show a dramatic performance increase for this specific task when compared to a dense network.

The results also indicated that a more complex CNN model, involving either more layers or more filters, does increase performance up to a point. Measuring the model training times is also important when reviewing the performance gain (or loss) of complex models. For example, experiment 5F kept the same architecture as 5E, only modifying the number of filters in each convolutional layer and the number of nodes for the fully connected layers, increasing from 64 filters to 256 and 64 nodes to 128 respectively. However, the training time of the larger network (5F) was nearly double that of 5E, increasing accuracy by about 5%. Depending on the use case, this increase in performance may be negligible, or still deemed unacceptable, only increasing the resource load and training times.

Reviewing experiment #3's convolutional and max pooling values also provided some interesting insight into what features of an image the model was "lighting up". The image that was chosen was that of an orange and white cat, with each convolutional filter picking out different features or highlights in the image. Some would focus more on the white or orange areas of the cat's fur, while others identified ridges or lines present in the image. Each max pooling layer demonstrated the reduction in image size/pixel count, by leveraging the same filter's identified features at a reduced scale. Even though the first pooling layer reduced the image size by 50%, most of the filters in the following convolutional layer were able to identify highlighted areas of the image, even when increasing the filter count of the layer.

Conclusions

This research was conducted to explore more complex network architectures in comparison to single hidden layer fully connected architectures, with an emphasis on image classification. The majority of hyperparameters were kept consistent with other models of the same type of architecture (i.e., fully connected or CNN), to help in showcasing a change in model performance with a small addition/change in a single hyperparameter.

While keeping the labels in a sparse categorical state, the fully connected (dense) networks provided to be insufficient for the task, specifically with two or three hidden layers. Considering that performance of these models may be improved with more hidden layers, more nodes, additional types of regularization, or converting the sparse categorical values into one-hot encoded representations, it is shown that convolutional networks perform much better without significant tweaking to the architecture. The worst performance of a CNN model was about 63% accuracy, which resulted after adding 50% dropout to the three-layer CNN architecture. This may be due to the stochastic nature of the network, or due to too high of a dropout value for the defined architecture.

This research also provided more insight into a convolutional model with pooling layers representing features within an image. This provides us with the ability to review what the network ‘sees’ and be able to adjust the architecture in the event we are removing too many features from the images. In this specific example with CIFAR-10 data, the images are already pixelated, demonstrating how too many max pooling layers (or too much of a reduction) may negatively impact the model’s performance, speaking more towards overly complex models not always achieving better performance.

References

Bau, D., Zhu, J., Strobelt, H., Lapedriza, A., Zhou, B., & Torralba, A. (2020). Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences, 117*(48), 30071-30078. doi:10.1073/pnas.1907375117

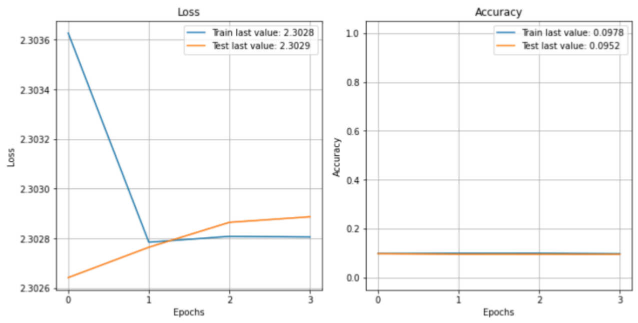
Brownlee, J. (2019, July 05). How to Visualize Filters and Feature Maps in Convolutional Neural Networks. Retrieved July 23, 2021, from https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/

Chollet, F. (2021). *Deep Learning With Python*. S.l.: O'Reilly Media.

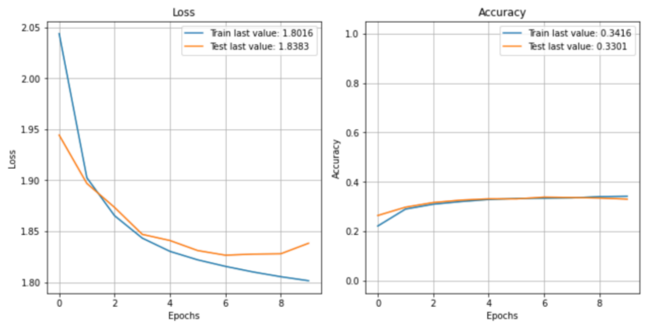
Appendix

Model Performance Plots:

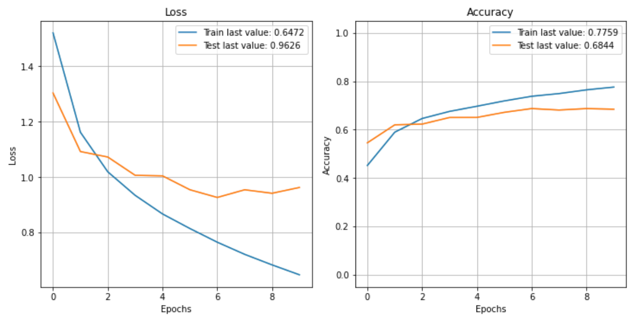
Model 1:



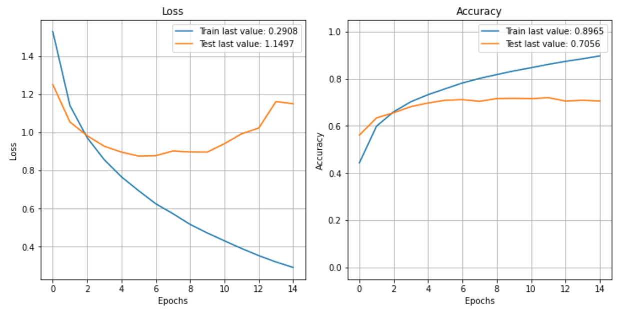
Model 2:



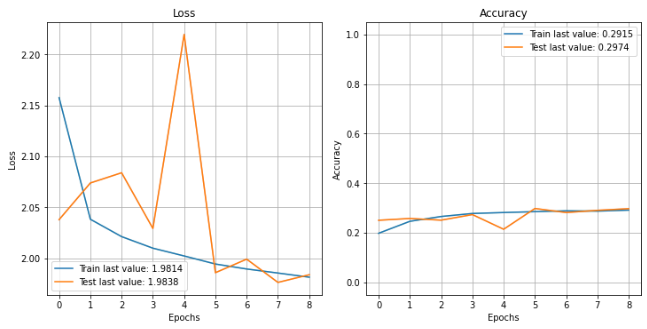
Model 3:



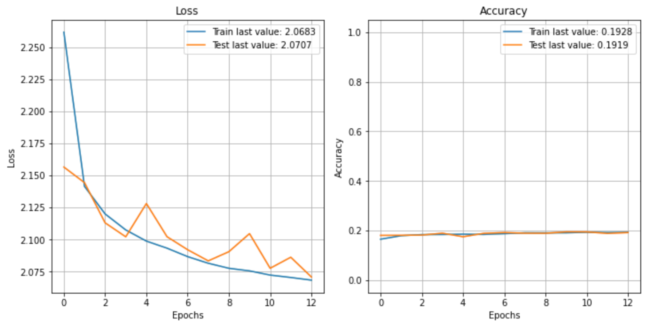
Model 4:



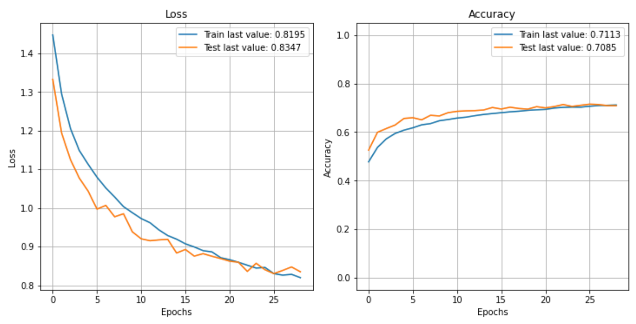
Model 5A:



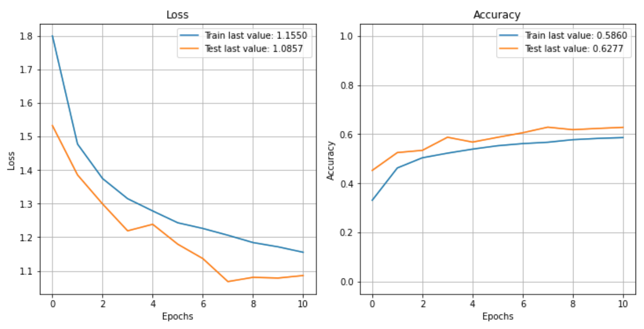
Model 5B:



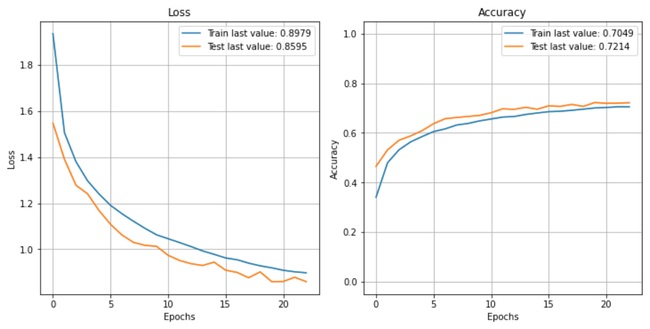
Model 5C:



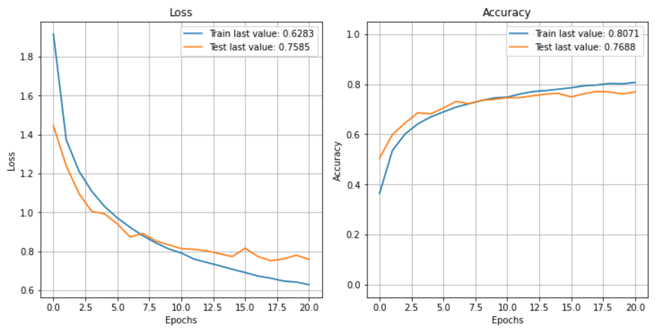
Model 5D:



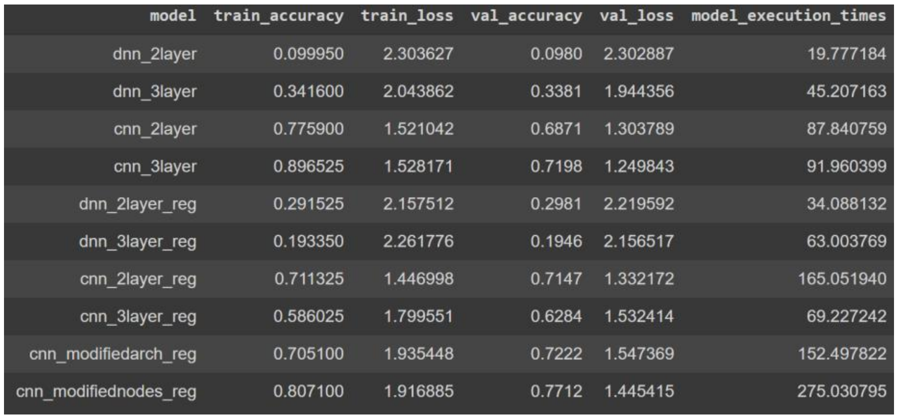
Model 5E:



Model 5F:



Results Table:

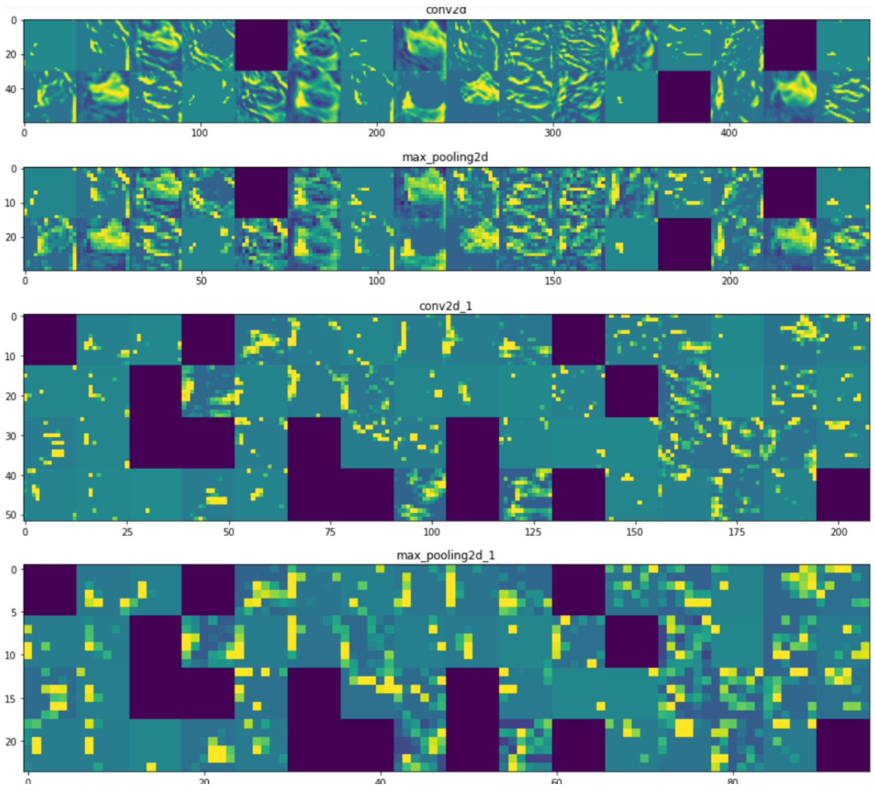


model	train_accuracy	train_loss	val_accuracy	val_loss	model_execution_times
dnn_2layer	0.099950	2.303627	0.0980	2.302887	19.777184
dnn_3layer	0.341600	2.043862	0.3381	1.944356	45.207163
cnn_2layer	0.775900	1.521042	0.6871	1.303789	87.840759
cnn_3layer	0.896525	1.528171	0.7198	1.249843	91.960399
dnn_2layer_reg	0.291525	2.157512	0.2981	2.219592	34.088132
dnn_3layer_reg	0.193350	2.261776	0.1946	2.156517	63.003769
cnn_2layer_reg	0.711325	1.446998	0.7147	1.332172	165.051940
cnn_3layer_reg	0.586025	1.799551	0.6284	1.532414	69.227242
cnn_modifiedarch_reg	0.705100	1.935448	0.7222	1.547369	152.497822
cnn_modifiednodes_reg	0.807100	1.916885	0.7712	1.445415	275.030795

Feature Map Visualization:

Original Image:

Feature Map Visualization:


conv2d
max_pooling2d
conv2d_1
max_pooling2d_1