Akshat Phumbhra

Deep Learning

12/12/2021

Image Classification

I. Introduction

Image classification is a fascinating problem in deep learning. This ability to classify images adds meaning to the field of computer vision. This paper presents a deep learning solution to an image classification problem. Our problem is to categorize all the pixels of a digital image into one of the 10 defined classes. Such classification has critical use cases in digital image analysis. Here, I present a supervised classification where the model is trained on some target classes and then used to classify unseen images. While there is no limit to what can be classified in a real-world setting, this project is limited to the ten classes of the CIFAR-10 dataset.

II. Previous Solutions

CIFAR-10 is a well-understood dataset and widely used for benchmarking computer vision algorithms in the field of machine learning. The problem is "*solved*." It is relatively straightforward to achieve 80% classification accuracy. Top performance on the problem is achieved by deep learning convolutional neural networks with a classification accuracy above 90% on the test dataset. Given that CIFAR-10 is a solved problem, there are several neural networks out there that can achieve very high accuracy scores on this dataset. The most notable of these are the ResNet and DenseNet networks. Both are able to achieve higher than 95% accuracy on this dataset. This is also why I decided to create my own network from scratch as opposed to using transfer learning. I feel like creating this network gave me an opportunity to learn a lot more about neural networks than I would have by simply implementing transfer learning.

III. Dataset

CIFAR is an acronym that stands for the Canadian Institute For Advances Research and the CIFAR-10 dataset was developed along with the CIFAR-100 dataset by the researchers at the CIFAR institute. The dataset is comprised of 60,000 32x32 pixel color photographs of objects from 10 classes. The class labels and their standard associated integer values are listed as follows: 0: airplane, 1: automobile, 2: bird, 3: cat, 4: deer, 5: dog, 6: frog, 7: horse, 8: ship, 9: truck. These are very small images, much smaller than a typical photograph, and the dataset was intended for computer vision research.

IV. Proposed Method

We first prepare our data for training. Since we have 10 classes and each of these are represented as unique integers, we can use a one hot encoding for the class element of each sample. That is, we transform the integer into a 10-element binary vector with a 1 for the index of the class value. We also know that the pixel values for each image are between 0 and 255. We normalize these pixel values and rescale them to the range [0, 1]. This is done by first converting the data types from integers to floats and then dividing by the maximum value (255). We now define our neural network model. We base our approach on the general architectural principles of the VGG models. This architecture involves stacking convolutional layers with small 3x3 filters followed by a max pooling layer. Together these layers form a block. We repeat these blocks where each the number of filters in each block is increased with the depth of the network. We use the reLU activation function and the He weight initialization as these are generally best practices. Our model uses 3 such VGG blocks. To further improve our model, we use dropout regularization. Dropout is a technique that randomly drops nodes out of the network. This has a regularization effect as the remaining nodes must adapt to pick up the slack of the removed nodes. The dropout rate is set as a hyperparameter. This rate decides the percentage of the nodes that we drop. Our dropout rates are set as 20%, 40% and 50% for each of the VGG blocks. Finally, we add batch normalization in order to stabilize the learning and perhaps accelerate the learning process. Batch normalization is a technique designed to automatically standardize the inputs to a layer. That is, it changes the inputs of the layer to have 0 mean and standard deviation of 1.

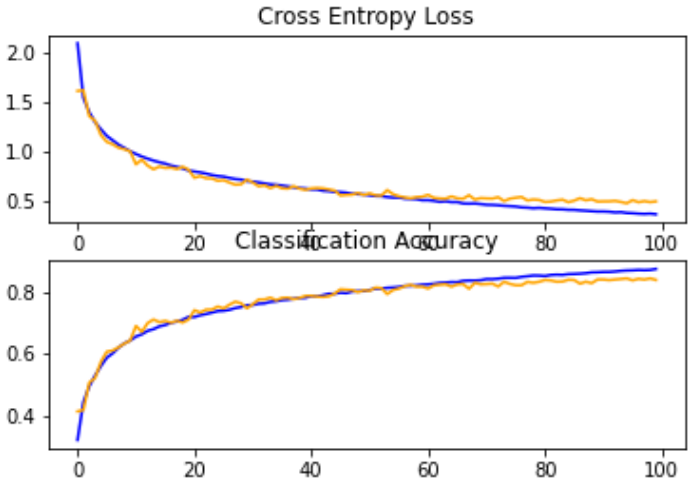
V. Evaluation Method

The method to evaluate this model is relatively simple. Keras provides a quantitative evaluation metric that works perfectly for a project such as this one. This evaluation gives us the classification accuracy for the model. Further, we also create a plot of the cross-entropy loss and classification accuracy during the model fit process. This shows how the model performed on the training set as opposed to the test set. We can tell if our model was overfit or underfit by looking at this plot.

VI. Results

We see that our model achieved an 84.62% classification accuracy on the test set. Considering that we built our model from scratch, this is a promising accuracy score. We were further able to analyze the results of the model by making predictions from using images that were not from the CIFAR-10 dataset. Using random images from Google, we noticed that the model was considerably accurate when classifying non-living objects like airplanes or trucks. However, it often struggled to differentiate between the animals. That is, it was confused between dogs and frogs. We also noticed that sometimes it classified airplanes as birds. This, however, feels like an

acceptable error since given the low resolution of the images in the dataset it is often difficult to tell these apart. Lastly, we can see below a plot of the cross-entropy loss and classification accuracy of the model as it trained.
From the image, we see that the metrics for both training and test sets were about the same. It was only at higher accuracies that we see some signs of overfitting.



VII. Discussion

Looking at the progress made over the course of this semester, I realized that I was able to add more and more components to my model to continuously improve its performance. When I created my first model, it only had a classification accuracy of 64%. By implementing other techniques like dropout regularization and batch normalization, I was able to get my accuracy above 80%. I also learned about different architectures for model building which is something I would not have been able to do if I simply implemented transfer learning. Given more time, I would have liked to make a graphical user interface so that classifying new images would be easy. Overall, I am happy with the progress I made and will continue to work on this model to increase my accuracy above 90%.