Mini-Project 3: Computer Vision using GPU and Transfer Learning

Ronngguang Ou (219817313)
John Leonardo (216649967)
Manuel Herrera (219721880)

CSC 180 Intelligent Systems
Mini-Project 3

Friday, October 25, 2019

Problem Statement

Using the CIFAR-10 dataset consisting of 60,000 color images in 10 classes, can we use these images to train and create a working model to correctly identify and label images in these same classes? The goal is to make a classifier model that can distinguish between ten different classes of images, and output a probability that a test image is any of the given classes. Also, the classifier model should be done using both with and without transfer learning.

Methodology

First, we used Google's research notebook environment and enabled a GPU backend for our notebook. Then, we imported and downloaded the CIFAR-10 dataset so we could use it to train our model, and instantly split it into training and testing sets.

We started with the model that didn't use Transfer Learning. We didn't do any magic with the data, and simply flattened the images and normalized the pixel data. Then, we initialized a model with a Convolutional 2D, Activation, Max Pooling, Flatten, Dropout, and Dense layer. For each activation in the hidden layers we used Elu, and for the final activation layer we used Softmax.

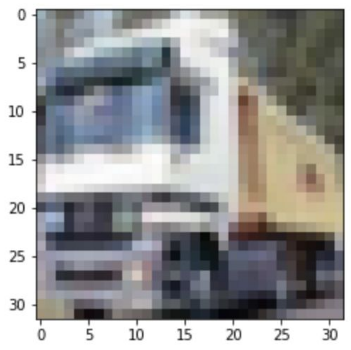
Next, we moved onto the model that uses Transfer Learning. The general idea of Transfer Learning is to reuse an existing model as a starting point for a new model. Sometimes, there is a lot of time and computing power required to get a good model, so there is no point to "reinvent the wheel". To start this off, we used a pre-trained model called VGG16. VGG16 supports images down to 48x48, and since our input data is images of size 32x32, we used a method called "upsampling" to fit these images to this model.

Lastly, we added two more dense layers, including one as our final output layer with 10 neurons. Before doing so, we were sure to freeze the weights in the model, so that the pre-trained model in the transfer process wouldn't be tampered with.

Experimental Results and Analysis

For our experiment, the goal was to create a model that could accurately identify pictures between 10 classes. Here is an example of the image data that we are working with:

```
<matplotlib.image.AxesImage at 0x7fa8c2fca588>
```



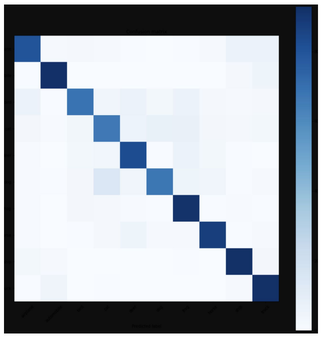
With the model that didn't use Transfer Learning, the final parameter count came out to be 309,290, with 308,394 of those being trainable - making this a pretty lean model in terms of Computer Vision. We used this final collection of parameters/layers to run our model using relu as our activation parameter when adding layers and used categorical cross-entropy for this multiclass classification when compiling our model with the RMSProp optimizer. After just 5 epochs, our neural network hit early stopping and came back with the following values for precision, recall, and f1-score:

```
Accuracy: 0.8223
Averaged F1: 0.819659442013431
              precision    recall  f1-score   support

           0       0.84      0.80      0.82      1000
           1       0.90      0.92      0.91      1000
           2       0.84      0.69      0.75      1000
           3       0.72      0.67      0.69      1000
           4       0.79      0.83      0.81      1000
           5       0.84      0.67      0.75      1000
           6       0.77      0.92      0.83      1000
           7       0.87      0.88      0.87      1000
           8       0.86      0.93      0.89      1000
           9       0.81      0.93      0.87      1000

    accuracy                           0.82     10000
   macro avg       0.82      0.82      0.82     10000
weighted avg       0.82      0.82      0.82     10000
```

Also, here was the confusion matrix for that model:



With the model that used Transfer Learning, using a pre-trained model called VGG16 the final parameter count came out to be 14,714,688, with all of those being trainable - making this a more dense model than the previous. We used this final collection of parameters/layers to run our model using relu as our activation parameter when adding layers and used categorical cross-entropy for this multiclass classification when compiling our model with the Adam optimizer. After just 2 iterations, with 10 total epochs, our neural network hit early stopping and came back with the following values for precision, recall, and f1-score:

```
Accuracy: 0.7007
Averaged F1: 0.697365570010261
              precision    recall  f1-score   support

           0       0.78      0.74      0.76      1000
           1       0.70      0.88      0.78      1000
           2       0.64      0.64      0.64      1000
           3       0.59      0.42      0.49      1000
           4       0.65      0.68      0.66      1000
           5       0.61      0.63      0.62      1000
           6       0.67      0.77      0.71      1000
           7       0.77      0.73      0.75      1000
           8       0.79      0.82      0.80      1000
           9       0.83      0.70      0.76      1000

    accuracy                           0.70     10000
   macro avg       0.70      0.70      0.70     10000
weighted avg       0.70      0.70      0.70     10000
```

Task Division and Project Reflection

Task division for this project were assigned as follows, Manuel handled preprocessing and formatting the data for the model. Jason was in charge of convolutional neural networks (CNN) and configuring the research environment. John handled the transfer-learning CNN model.

Challenges encountered throughout the project included trying to fit the images to a pre-existing model by upsampling, and trying to figure out how to make the best use of the transfer-learning system. Also, another challenge trying not to overcomplicate the data. In terms of getting a more accurate model, it was also extremely challenging to figure out how to parameter tune the models for the better. As models get more complex, it becomes increasingly difficult to make good decisions for the model in terms of parameter tuning.

Overall, we learned while the best approach for AI could and probably is Transfer Learning, it’s definitely not an approach for someone who doesn’t have a lot of time. By nature, you are inheriting someone else’s model, and it requires care, data preprocessing and postprocessing in a way that bests maximizes that pre-trained model. An excellent model is useless if the pre and postprocessing is not correctly executed.