Assignment 3 Report

CPSC 599, Deep Learning For Vision

Deliar Mohammadi, 30072994
John Zheng,30125258
Xinzhou Li, 30066080

Overview of Problem:
We had about 4000 images (of cars) which needed to be classified into 100 different classes. Overfitting is a problem with only ~40 samples per class in our training data. We approached the problem with many different models and applied several techniques to improve accuracy, they are all discussed below.

Approach 1:
Using transfer learning, we could use a pre-trained model to identify features and then train only the classifier. The pre-trained model would be effective in avoiding overfitting because it was trained on a much larger dataset with much more general feature extractions.
For our first attempt, we used Pytorch's resnet50 with a 5-layer deep fully connected classifier. The classifier had dropout layers to further reduce overfitting.
We froze the feature extraction layers on the pre-trained model and only trained the classifier. Using this approach, we managed to get ~35% on validation data (which is much higher than the expected 1% from randomness).

Image Preprocessing:
Image preprocessing was required for using the pre-trained model, and we copied the pre-processing that the model used (which was a resize to 256, center crop to 224, and a normalize with mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]. Those were the mean and standard deviation of the one million images in ImageNet for the three channels.).
We augmented our images to increase input variance. We applied random transforms (horizontal flips, small colour adjustments etc.) to do so. These augmentations should have also reduced overfitting. We applied these augmentations before the preprocessing step.

Approach 2:
Still using transfer learning, we upgraded our pre-trained network from resnet50 to resnet152. This deeper network has more effective feature extraction.
We also used tanh connections on our fully connected layer instead of ReLU. This is because the ResNet architecture specifically avoids the vanishing gradient problem (by allowing the "residual" part to skip over layers and directly impact deeper layers), so the gradient is still good at the final convolution layer. The fully connected final layers are only 4 layers deep, so the four layers of tanh activations do not cause a problem.
Instead of freezing the feature extraction for the entire training phase, we only froze it for the first ten epochs. After that, we allowed the entire model to be trainable so that the feature extractor could also become more specific towards our dataset.
Training the entire network runs a higher risk of overfitting, so we tested the results at various points so that the model would not be overtrained.

Approach 3:
For our final approach we took the same model from approach 2 and made the following changes:

1. Reduced the layer count in the classifier by 1
2. Eliminated the only dropout layer that we had

3. Added a scheduler to reduce our learning rate after each epoch, this was to prevent overfitting
4. Made minor modification to the transformations for the training dataset to encompass a larger variety of data

To train our final approach, we froze all the Conv layers in our network and only modified the fully connected layer for around 40 epochs. After this we unfroze the conv layers and began training the entire model for about 80 more epochs. Throughout our training we stopped at intervals to record answers.

With these changes we began running the model on our hardware and were unable to finish the epochs in time to see the results, however we will upload the results of this new model to kaggle after the training is done since the competition will remain open for longer than the D2L dropbox.

Overfitting:

Our model could generally achieve ~99% accuracy on the training data, but only ~80% percent on the testing data. This means that our model is still overfitting and that it could be optimized further. With about 40 images per class of car in the training data, it is completely feasible for a human to identify images with >99% accuracy. Here are a list of a few things we did to avoid overfitting our model to the training data.

1. Learning rate scheduler
    a. This will reduce the changes of the network weights at the very last epochs, reducing overfitting
2. Data augmentation
3. Data regularization
4. Testing various fully connected components in the network
5. Outputting answers at every 8 or so epochs to make sure we get answers from the best model

With the above techniques we were able to increase our model accuracy drastically.