**Pezhman Lamei**

**Assignment 2**

**1| Summary report**

**Training from Scratch**

The problem with our data set is that it contains little data. So, we tackle this problem by training a new model from scratch. We start by naively training a small convent on the 1000 training samples, without any regularization, to set a baseline for what can be achieved. This gave us a classification accuracy of 70%. At that point, the main issue was overfitting. Then we applied *data augmentation*, a powerful technique for mitigating overfitting in computer vision. Using data augmentation, we optimized the network to reach an accuracy of 84.5%. To handle the problem of having little data, we increased the size of the training sample to 1500 and then 2500. This showed that the accuracy improved from 70% to 77% and 85.7%. Again, by using the data augmentation technique, we reached a better performance in bigger training samples, 87.8 %, and 90%, respectively.

**Using pretrained network**

Further, we employed a more useful technique for applying deep learning to small datasets: feature extraction with a pretrained network. This technique gave us way better accuracy, around 96.7%, with the smallest training sample (1000). Expectedly, fine-tuning a pretrained network provided us with more accuracy, even better than pretrained network, with 97.6 %. Adding more training samples (from 1000 to 2500) improved our accuracy in two models as same as training from scratch. However, the importance of training sample size is more for training from the scratch method because the efficient pretrained network has basically 97% accuracy and is less dependent on the size of the training sample*.*

|  |  |  |  |
| --- | --- | --- | --- |
| Technique | Training sample size | Accuracy without optimization | Accuracy with optimization |
| Training from Scratch | 1000 | 69.9 % | 84.5% |
| Training from Scratch | 1500 | 77% | 87.8% |
| Training from Scratch | 2500 | 85.7% | 89.9% |
| Pretrained network | 1000 | 96.7% | 97.6% |
| Pretrained network | 1500 | 97.8% | 97.5% |
| Pretrained network | 2500 | 99.2 % |  |

**2| The relationship between training sample size and choice of network**

Deep learning is often described as a method that makes extensive use of data, however this isn't really the case. In fact, it is a subset of deep learning that searches training samples for interesting features using a large amount of data without manually building features. In contrast, if the job is straightforward and the model is modest and well-regularized, it may be easy to train a computer to handle a basic issue with only a few hundred samples. Convnets are particularly data-efficient on perceptual tasks because they learn local, translation-invariant features. Without the requirement for specific feature engineering, training a convnet from start on a very tiny picture dataset will nonetheless produce acceptable results despite the limited amount of data. When we train a model from scratch, we build a new model and train it on a dataset. As a result, employing the right network and optimization approaches, we may apply the deep learning approach to tiny data sets.

Moreover, in this exercise, we used an optimization method, “data augmentation,” to handle the little training sample. Data augmentation takes the approach of generating more training data from existing training samples by *augmenting* the samples via several random transformations that yield believable-looking images. The goal is that at training time, your model will never see the same picture twice. This helps expose the model to more aspects of the data and generalize better.

As our second technique to handle little training samples, we used pretrained network, with is a useful and highly effective approach. A *pretrained network* is a saved network that was previously trained on a large dataset, typically on a large-scale image classification task. If the original dataset is large enough and general enough, then the spatial hierarchy of features learned by the pretrained network can effectively act as a generic model of the visual world, and hence its features can prove useful for many different computer vision problems. This approach used feature extraction, which consists of using the representations learned by a previous network to extract interesting features from new samples. These features are then run through a new classifier, which is trained from scratch.

In sum, we can use deep learning for small data samples if we choose an appropriate approach fitting with the data sample size and complexity of the model.

**3| Results**

*With a training sample of 1000, a validation sample of 500, and a test sample of 500*

Training from Scratch

Chart

Description automatically generated with medium confidence Chart, line chart

Description automatically generated

Test accuracy: 0.699

After data augmentation

Test accuracy: 0.845

Using pretrained network

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

Test accuracy: 0.967

Using fie-tuning

Test accuracy: 0.976

*with a training sample of 1500, a validation sample of 500, and a test sample of 500*

Training from Scratch

Chart

Description automatically generated with low confidence Chart, line chart

Description automatically generated

Test accuracy: 0.770

After data augmentation

Test accuracy: 0.878

Using pretrained network

Chart

Description automatically generated with medium confidence Chart, line chart

Description automatically generated

Test accuracy: 0.978

After fine-tuning

Test accuracy: 0.975

*With a training sample of 2500, a validation sample of 500, and a test sample of 500*

Training from Scratch

Shape

Description automatically generated with medium confidence A picture containing graphical user interface

Description automatically generated

Test accuracy: 0.857

After data augmentation

Test accuracy: 0.899

Using pretrained network

A picture containing shape

Description automatically generatedChart, line chart

Description automatically generated

Test accuracy: 0.992

After fine-tuning