Lab: AlexNet

Transfer Learning with TensorFlow

Transfer learning is the practice of starting with a network that has already been trained, and then applying that network to your own problem.

Because neural networks can often take days or even weeks to train, transfer learning (i.e. starting with a network that somebody else has already trained) can greatly shorten training time.

How do we apply transfer learning? Two popular methods are **feature extraction** and **finetuning**.

1. **Feature extraction**. Take a pretrained neural network and replace the final (classification) layer with a new classification layer, or perhaps even a small feedforward network that ends with a new classification layer. During training the weights in all the pre-trained layers are frozen, so only the weights for the new layer(s) are trained. In other words, the gradient doesn't flow backwards past the first new layer.
2. **Finetuning**. This is similar to feature extraction except the pre-trained weights aren't frozen. The network is trained end-to-end.

The labs in this lesson will focus on feature extraction since it's less computationally intensive.

Getting Started

1. Set up your environment with the **Udacity CarND Starter Kit (https://github.com/udacity/CarND-Term1-Starter-Kit)**.
2. Clone the repository containing the code.

```
git clone https://github.com/udacity/CarND-Alexnet-Feature-Extrac
tion
cd CarND-Alexnet-Feature-Extraction
```

3. Download the **training data (https://d17h27t6h515a5.cloudfront.net/topher/2016/October/580a829f_train/train.p)** and **AlexNet weights**

(https://d17h27t6h515a5.cloudfront.net/topher/2016/October/580d880c_bvlc-alexnet/bvlc-alexnet.npy). These links are also available at the bottom of this page under **Supporting Materials**.

4. Make sure the downloaded files are in the code directory as the code.
5. Open the code in your favorite editor.

Feature Extraction via AlexNet

Here, you're going to practice feature extraction with **AlexNet (https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiG34CS7vHPAhVKl1Q imagenet-classification-with-deep-convolutional-neural-networks.pdf&usg=AFQjCNFlGsSmTUkJw0gLJ0Ry4cm961B7WA&bvm=bv.136593572,d.cGw)**.

AlexNet is a popular base network for transfer learning because its structure is relatively straightforward, it's not too big, and it performs well empirically.

There is a TensorFlow implementation of AlexNet (adapted from **Michael Guerhoy and Davi Frossard (http://www.cs.toronto.edu/~guerzhoy/tf_alexnet/)**) in `alexnet.py`. You're not going to edit anything in this file but it's a good idea to skim through it to see how AlexNet is defined in TensorFlow.

The team behind AlexNet. Source: Wired.

Coming up, you'll practice using AlexNet for inference on the image set it was trained on.

After that, you'll extract AlexNet's features and use them to classify images from the **German Traffic Sign Recognition Benchmark dataset (http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset).**

Credits

This lab utilizes:

- An implementation of AlexNet created by **Michael Guerzhoy and Davi Frossard (http://www.cs.toronto.edu/~guerzhoy/tf_alexnet/)**
- AlexNet weights provided by the **Berkeley Vision and Learning Center (http://bvlc.eecs.berkeley.edu/)**
- Training data from the **German Traffic Sign Recognition Benchmark (http://benchmark.ini.rub.de/?section=gtsrb&subsection=news)**

AlexNet was originally trained on the **ImageNet database (http://www.image-net.org/).**

Supporting Materials

Train (https://d17h27t6h515a5.cloudfront.net/topher/2016/October/580a829f_train/train.p)

Bvlc Alexnet Weights (https://d17h27t6h515a5.cloudfront.net/topher/2016/October/580d880c_bvlc-alexnet/bvlc-alexnet.npy)

NEXT