11-695: Competitive Engineerng
Assignment 1: Image Classification

Spring 2018

Abstract

In this assignment, you will implement an image classification system in TensorFlow [Abadi et al., 2015]. The starter code is provided for you, where a simple softmax classifier has been implemented. Your job is to understand the starter code and implement two models of yours: a feed-forward neural network, and a convolutional neural network. For each network, you have the freedom to explore different architectures, to implement, to train, and to test your model. You will be working with the CIFAR-10 dataset [Krizhevsky, 2009], where we expect your best method to reach 65% accuracy.

1 Code and Dataset

Install TensorFlow. If you have not already, please navigate to TensorFlow's site and follow their instructions to install the framework. The instructions are at

`https://www.tensorflow.org/install/`

Their instructions should be sufficient to install TensorFlow and all of its dependencies on your system. It is possible to complete this assignment without using any GPU, so you do not need to install CUDA or anything related to GPU programming. However, if you wish to, you are more than encouraged to install TensorFlow GPU, which will make your implementation much faster.

Download the CIFAR-10 Dataset. You will be working with the CIFAR-10 dataset, which is available at

`https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz`

This dataset consist of $50,000$ training images and $10,000$ testing images. All images are RGB images with the size of 32×32. Each image is assigned one of 10 pre-defined labels. You can find out more about the CIFAR-10 dataset in the paper by Krizhevsky [2009].

The state-of-the-art performance on the CIFAR-10 dataset is 97.7% accuracy, which is achieved by Neural Architecture Search [Zoph et al., 2017]. In this assignment, your best model is required to reach 65% accuracy. For each 1% beyond that, *rounding up,* you will get +1 point for extra credit. For example, 68.1% accuracy will give you +4 points for extra credit.

The starte code implements some data preprocessing procedures for you. In particular, we reserve $5,000$ training images for validation. Thus, there are $45,000$ images for training. We have also subtracted the channel mean and divided the channel standard deviation from all images. These are the standard data preprocessing steps for the CIFAR-10 dataset.

Starter code. The starter code for your project is available at

http://www.cs.cmu.edu/afs/cs/user/hieup/www/11695/starter_code.zip

After downloading and unzipping the code and the CIFAR-10 data, you can navigate to your code directories, where you will see the folders `src` and `scripts`. You will be writing code in the `src` and running your work using the shell-script files in `scripts`. A simple file, `scripts/cifar10.sh` has been written for you. If you open the file and change the `data_path` flag into the *absolute path* to the directory where you de-compressed CIFAR-10 data, you will be able to run the starter code.

The starter code implements an extremely simple baseline algorithm: a softmax classifier. This algorithm performs the following steps:

- Flattens each CIFAR-10 images into a vector of 3072 numbers,
- Applies a matrix multiplication to turn the image into a logit vector of 10 numbers,
- Applies the softmax functoin on the logits to obtain the class distributions.

This softmax classification is trained using stochastic gradient descent (SGD), with a fixed learning rate, which is not a very good setting for SGD. If you run the starter code, which trains this model for only 1000 steps, you will get around 25% accuracy on the test data.

2 Your Work

The entry point of the program is in the file `src/main.py`. From this file, relevant modules from `src/models.py`. It is your responsibility to read the code and figure out all the relevant points.

You are required to implement the functions `conv_net` and `feed_forward_net` and modify the script `scripts/cifar10.sh` to execute the corresponding models. You can use any techniques that you desire, including but not limited to the following suggestions:

- ℓ_1 regularization, ℓ_2-regularization, etc.
- Activation functions: `tf.sigmoid`, `tf.tanh`, `tf.nn.relu`, etc.
- Momentum training [Nesterov, 1983]. The `TF` code for this is: `tf.train.MomemtumOptimizer`,
- DropOut [Srivastava et al., 2014],
- Batch Normalization [Ioffe and Szegedy, 2015]. For `TF` code, you can use `tf.nn.fused_batch_norm`,
- Residual connections [He et al., 2016], Highway connection [Greff et al., 2017],

You can also use any number of layers and architectures. Remember that networks with more layers are harder to train, but if you can train them appropriately, you can *usually* get a better performance.

Gradings. The grading breakdowns are as follows:

- Feed-forward network: **20 points.**
- Convolutional network: **30 points.**
- Performance: **15 points.**
- Extra credit: up to **35 points.**

When you submit your assignment, please also submit the running scripts, named `scripts/feed_forward.sh` and `scripts/conv_net.sh`. These scripts should look like the provided script `scripts/cifar10.sh` that we provide you, with `--model_name=‘‘feed_forward’’` and `--model_name=‘‘conv’’`, respectively. You can also set other hyper-parameters of your models by adding flags to these scripts and use the appropriately. You will get the full credit for your feed-forward and convolutional network if they run and perform better than our softmax classifier.

For the 15 performance points, your better model (which is very likely the convolutional network) should reach 65% accuracy. For each percent below 65%, you lose 1 point, and for each percent above that, you have 1 point of extra credit. All accuracy percents will be rounded up.

Academic Integrity. As normal, you are encouraged to discuss with your friends and the instructors. Anything they tell you, you can use. However, looking at other's code should not happen at all cost.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Klaus Greff, Rupesh K. Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. In *ICLR*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CPVR*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Yurii E. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Soviet Mathematics Doklady*, 1983.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *JMLR*, 2014.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *Arxiv, 1707.07012*, 2017.