50.021 -AI

Alex

Week 05: Fine Tuning aka Retraining

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

Vision - state of the art

- Batch normalization – Sergey Ioffe et al. **This is not a vision specific idea!**
- Resnets - residual short cuts – Kaiming He et al.. **This is not a vision specific idea!**
- Ensembles: dont train one network, train multiple of them. Why? Neural network training lets you find one of the many local optima. Everytime you train a network, and every epoch (when you traverse the set of all samples) you see the images in a random order. So you will get everytime a slightly different trained neural network - different by the learned parameters. It helps to average over multiple neural networks. **This is not a vision specific idea!**
- custom designs: small convolution kernels but stacking them deeply – no one would use a 15×15 kernels – way too many parameters!!
- googlenet/inception: use modules with effectively differing kernel sizes: 1×1, 3×3, 5×5 by stacking of 3×3 and fuse their output ... looking at multiple scales in parallel

  other vision topics:

- some papers for detection: Faster RCNN – precise, YOLO – near realtime (J. Redmon)
- learning to generate chairs: papers by Thomas Brox (last author)
- generative adversarial networks

Finetuning

The goal of this lesson is to understand about finetuning when training a neural net.

One does not train deep neural from scratch. Always reuse weights from similar tasks as initialization, except your data is in the order of hundred thousands and more.

In the practice session you will be asked to take an alexnet (bcs it trains fast), initialize it with weights from a 1000 class imagenet task, and then retrain it for 102 flowers classes. Is it surprising that one can re-use weights from 1000 object classes that are mostly things and animals for flowers? The low level filters likely will be very similar.

What needs to be changed? Well the last layer for sure: one has now 102 output classes instead of 1000.

Mnist and Cifar-10 work well without finetuning, however note the simplicity of the tasks: images with 28×28, or 32×32 have limited variability compared to 300×500 images! Mnist and Cifar-10 are very useful for testing small ideas, but they are not necessarily typical for deep learning tasks.

Finetuning is a general principle. Reuse weights from lowest layers obtained from a similar task. Use them until the first later when you change something. You want to use a complicated neural net that does something totally else than classification or detection? Finetuning is still applicable.

Example: suppose you want to tell stories from video frames. so you want to process a sequence of images, and then output a sequence of text.

You will consider to do the textual outputs by a recurrent neural net, but you will need to input feature vectors computed from every frame of the video. If you dont want to use 3D-convolutions, then you can process every image frame by a neural net trained for classification, and input as features the activations of neurons from some higher layer of this neural net!

Finetuning has some special cases: when the number of training data is very small, then one may want to retrain only the last 1-2 layers. This can be achieved by forcibly set $dE/dz = 0$ for activations below the layers that one wants to retrain. The tensorflow tutorial on image retraining does exactly that. This has the advantage, that one can precompute the features for each image for the last layer for which one does not change weights. However this is only possible, if one does not use random image augmentations. So here one has a time vs efficiency tradeoff, I tried to let you see the more general case

Task

- check the AMI that I shared with your amazon account. You do not see any, hmm I might forgot to add permissions to you (guaranteed nothing personal), approach me.
- consider `finetune_pil.py` or `finetune.py`. The first uses PIL, the latter python3-opencv to do the image loading and transformation tasks.
- change `train_file, val_file, replacementpath` to the right path to trainfile.txt,valtestfile.txt, and the path where the jpg-images of oxford flowers are in.
- train the problem with 30 epochs **without** loading weights. to do this: outcomment the line `model.load_initial_weights(sess)`
- train the problem with 30 epochs **with** loading weights. Compare the validation data performance.
- for a problem with 102 classes, and equal numbers of images in each class at validation time, what is the accuracy if one would guess a label?