[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.]

Due Tuesday 6pm

Coding 1: train a deep neural net with transfer learning

- check the AMI that I shared with your amazon account: it is oregon zone, search for owner: 277133844599. You should see an AMI with ubuntu16_..... (or use your own, of course)
- take either the 102 class flowers dataset or the two-class hymenoptera dataset
- take any deep network you like (a resnet 18 or a mobilenet train fast)
- train a deep neural network: once with loading model weights before training, once without, once with loading model weights before training and training only the last fullyconnected layer.
- Homework: if not done yet, split the dataset in train/val/test. Run training for different values of the learning rate (e.g. modded by factor of 10). Report results with and without finetuning of all layers, and with finetuning but training only the highest layer. For Hymenoptera: also report how you created your used split (sample size). Optionally also do it for different batchsizes (e.g. modified by a factor of 4 or 8)

  what do you need to do for steps when you start with a code like the MNIST training code?

  - write a new dataloader for your training dataset
  - adjust paths for data (and if necessary for label paths/files or files determining splits into train/val/test)
  - decide on some at least basic data augmentation (how to load the images into a fixed size: resizing+center cropping, do more at training time? do what at test time ?)
  - replace the model for mnist by a model more suitable, load its weights before training
  - think of what results to report for homework submission. a naked code will not do it!
- A note: Calling a model constructor with `pretrained=True` does not tell you what really goes on when one. Check `https://`

`github.com/pytorch/vision/blob/master/torchvision/models/resnet.py` or `https://github.com/apache/incubator-mxnet/blob/master/python/mxnet/gluon/model_zoo/vision/resnet.py` – what routine is used to load a model?

Coding 2: train with a dataloaded which upsamples a rare class

- do this as homeworkW:
- consider `sampletr.txt` and `sampleste.txt` (v6 folder) for train and test data. every line is 2-dimensional input data with 1 label.
- plot the training data
- implement a neural net having only one fully connected layer with 2 inputs and one output. you can use `nn.BCEWithLogitsLoss` to train over outputs of that linear mapping as loss.
- train always with 100 iterations, learning rate 0.1, batchsize 128, no weight decay, SGD
- now train with a standard dataloader
- now write a dataloader which ensures that every minibatch consists of 50% of the smaller class and 50% of the larger class, and which gives everytime a random order of samples (note: what must length of the dataset be now? !! how to get such a sampling done?).
- compare for both choices: accuracy, balanced accuracy ($\alpha = 0.5$) in the weighted mean, and true positive rate
- if you are curious/not required: observe that you can find a good solution also with the standard loss when you reduce the batch size.
- bonus: 10% of this homework marks: implement a weighted loss and a generator for an artificial 2-class data set in 2 dimensions different from the current dataset, where the weighted loss works better than standard loss for the same training settings - you will need to show a dataset and repeat the experiment 10 to 20 times to show that the weighted loss finds the better solution (higher balanced accuracy) more often