Homework 6

All assignments need to be submitted via github classroom:
https://classroom.github.com/g/wEvsucZA
and on gradescope. You can submit in groups of maximum size two.

The homework is due 04/28/18 at 12am (midnight).

All the tasks are to be completed using the keras Sequential interface. Running on your own laptops might take some time, so make sure to start computations well in advance of the deadline. Feel free to use any other compute resources at your disposal.
You should not use k-fold cross-validation for any of these tasks. You can use StratifiedShuffleSplit to create a single train-validation split for use with GridSearchCV.
Use of GridSearchCV might not be the best option for any task but task1, though.

Task 1 [10 Points]

Run a multilayer perceptron (feed forward neural network) with two hidden layers and rectified linear nonlinearities on the iris dataset using the keras Sequential interface. Include code for model selection using GridSearchCV and evaluation on an independent test-set.

Task 2 [30 Points]

Train a multilayer perceptron on the MNIST dataset using the traditional train/test split. Use an separate 10000 samples (from the training set) for model selection and to compute learning curves. Compare a “vanilla” model with a model using drop-out. Visualize learning curves for all models.

Task 3 [30 Points]

Train a convolutional neural network on the SVHN dataset in format 2 (single digit classification). You should achieve at least 85% test-set accuracy with a base model. Also build a model using batch normalization. Your final accuracy will be included in the grading. You can compare against other approaches reported here if you’re curious. You are not required to use the unlabeled data.
Hint: Make sure you are doing the reshape for the training set correctly. A direct reshape might give you garbled images. Display an image after reshaping to make sure they are correct.

Task 4 [30 points]

Load the weights of a pre-trained convolutional neural network included in keras, see https://keras.io/applications/, and use it as feature extraction method to train a linear model or MLP on the pets dataset (http://www.robots.ox.ac.uk/~vgg/data/pets/). You should achieve at least 70% accuracy. It's recommended you store the extracted features on disk so you don't have to recompute them for model selection.
We will be working with the 37 class classification task.
Hint: Make sure that you apply the same preprocessing to the images that was applied for training the model.

Some additional advice to help you along:

- If you have a GPU at your disposal, make sure all your code is running on GPU (it needs to be "fast enough" for this to actually help). Use "`sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))`" to start the tensorflow session to see which device is being used and confirm it is the device you intended.
- Preprocess the images before training a model.
- Test your code on a small part of the data before training the model. You don't want your code to fail on a print statement after waiting for the network to train.
- You should be able to train models for all tasks within less than 30 minutes. If you exceeding that, either you have too big of a model, or there is a mistake in your code.
- Installation following https://www.tensorflow.org/install/ might give better runtime performance than the standard conda distribution which doesn't include many CPU optimization options.
- If you're adventurous, you can create a google cloud account, which comes with $300 in credits that you can use to run a GPU instance. You can find a lot of material on that online (for example here), but we won't provide assistance for that as part of the course.