1. (8/15) In this question you will perform a simple multi-task/transfer learning experiment, by appropriately modifying the CIFAR VGG network to simultaneously train on the CIFAR-10 and CIFAR-100 datasets.

What you need to do:

- Download the CIFAR-100 dataset from
  
  http://www.cs.toronto.edu/~kriz/cifar.html
  
  Like for CIFAR-10 before, do not normalize the data in any way, except for using single precision floating point numbers to represent the images.

- Implement the network such that it can be trained simultaneously on the CIFAR-10 and CIFAR-100 datasets, by minimizing the weighted sum

  \[ \alpha \text{CIFAR10loss} + (1 - \alpha) \text{CIFAR100loss} \]

- You may have to experiment to find a value for \( \alpha \) that works OK.

- Use "valid"-mode convolutions in the convolutional layers.

- Use the CIFAR-100 "fine" labels (100 labels), unless your runs take too long, in which case you may use the "coarse" labels. Clearly state which ones you used in your results.

- Unlike Assignment 1, do not use horizontal flipping or any other form of data augmentation.

- As in Assignment 1, you may want to use batch-normalization on the last layer(s).

- Perform multiple training runs, each time training the network on \( n \) randomly chosen examples from the CIFAR-10 training dataset, where, for example, \( n \in \{1000, 5000, 10000\} \). You may try larger values for \( n \) (up to the full CIFAR-10 training set) but will probably see diminishing returns in the effect of the regularizer. In each run, use the 50000 − \( n \) remaining training examples for validation.

  Use the whole CIFAR-100 training set in all cases; the sole purpose of the CIFAR-100 dataset here is to act as a regularizer for the CIFAR-10 task.

  For each \( n \), also train a model without transfer learning for comparison.

What to hand in:
• Some example images from the CIFAR-100 training set.
• For each \( n \), plots of the learning curves showing iterations on the x-axis and negative log-likelihood (for the respective CIFAR-10 subset) on the y-axis. Also write down the performance on the test data (always using the whole CIFAR-10 test set) for the model that performs best on the validation data.
• As a sanity check, add plots of the training and validation losses for CIFAR-100.
• For each \( n \), add the plots and performances (CIFAR-10 only) for the unregularized model.
• The value for \( \alpha \) that you used (values in case you varied \( \alpha \)).

2. (7/15) Get your hands on an Imagenet-pretrained CNN from which you can extract the penultimate layer features. One option is to use one of the feature transform networks from

http://sklearn-theano.github.io/

But you may use any pretrained network of your choice.
Upsample (using a method of your choice) the CIFAR-10 images to the appropriate size so they can be input to your pretrained CNN. Then extract the penultimate layer features for all training and test images. Train a regularized logistic regression (AKA softmax) classifier on the features, where you set the weight decay term using a validation set.

What to hand in:
• The training/test performance of the classifier.
• The corresponding optimal weight decay term.
• State (roughly) what kind of network/architecture you used.
• How did you do the upsampling?