Homework Assignment 3

CSE 253: Neural Networks

Winter 2017

Instructions

Due Sunday, February 19^{th}:

1. Please hand in your assignment via Vocareum. The report should be in NIPS format.
2. You should submit your code on Vocareum along with your report. For your own purposes, keep your code clean with explanatory comments, as it may be reused in the future.
3. You will be using Keras, a Deep Learning library, for the tasks in this assignment.
4. Please work in teams of size 4-5. In *extraordinary circumstances* (e.g., you have a highly contagious disease and are afraid of infecting your teammate), we will allow you to do it on your own. Please discuss your circumstances with your TA, who will then present your case to me.

Transfer Learning

Transfer Learning aims at solving new tasks using knowledge gained from solving related tasks.
In this assignment, you will learn to apply variants of Transfer Learning for the task of Image Classification. Deep architectures of Convolutional Neural Networks, with exploding number of parameters, cannot be trained on datasets of incomparable size. It is common practice to use existing models, trained on very large scale datasets, as feature extractors or initial layers of a ConvNet that can be fine-tuned to learn the underlying model of the dataset of our new task.

Transfer Learning with VGG16 (50 points)

We will classify images in a dataset similar to the ImageNet dataset, using a pre-trained CNN model. We will also explore the performance of Transfer Learning for classifying images in a dataset that are not so similar to ImageNet.

Keras is a Deep Learning library that provides high-level utilities to build deep networks. It also provides pretrained CNN models for image classification - VGG16 ConvNet, VGG19 ConvNet, ResNet, etc. trained on ImageNet, a very large scale visual recognition dataset.

The VGG16 ConvNet model, originally trained to classify images into 1000 classes, will be used to classify images our datasets. VGG16 consists of a set of Convolution Layers (including MaxPooling) followed by fully-connected layers and finally a Softmax Layer which is used to predict 1000 classes of ImageNet.

Here, we are going to try to reconstruct the graph of performance of ConvNet vs. number of training examples used, that we saw in class. To do this, we shall replace the Softmax Layer of the VGG16 pre-trained model by our own Softmax Layer which will predict classes of of our dataset. We will attempt to train only the Softmax Layer of the new model on just a few samples of our dataset and observe if it performs well.

I. Caltech256 Classification (25 points)
Here we will explore the application of Transfer Learning to the task of recognizing a set of object categories from images. In particular, we will be looking at the classification of the Caltech256 dataset.

1. **Read and prepare data**
   (a) Read in the Caltech256 images by walking through the directories of the dataset and converting images to an array representation.
   Tip: you can use either Keras image preprocessing utilities or any other utilities of your choice
   (b) Pre-process the data by subtracting the mean and performing global contrast normalization.
   Tip: the Keras VGG16 module has functions to help with this
   (c) Convert the target labels of the images, i.e., the class numbers to categorical/one-hot vector outputs of dimension 256.
   Tip: you can use Keras np_utils module
   (d) Shuffle the entire data and prepare hold-out set by taking a portion of the training set.

2. **Create Caltech256 ConvNet**
   (a) Create a Keras ConvNet model with weights of the VGG16 model. Use the starter file for this.
   (b) Replace the existing Softmax Layer with a new one whose output dimension is 256 (number of classes in our dataset)
   (c) Ensure only this last Softmax Layer is going to be trained. You can do this by setting *trainable* to *False* for all the layers except the last Softmax Layer.
   (d) Verify that the model has been updated and is appropriate.
   Tip: Use model.summary() for this

3. **Training** *(4 points)*
   Train the ConvNet with a small number of the training images per class. The idea is to replicate the Figure in the last lecture, which is Figure 9 from this paper. Try repeating the same, a few times, with larger subsets. As this will get more expensive computationally, the more examples you use, you don't have to replicate that whole figure. E.g., try 2,4,8, and 16 training examples per category (more if you have time). Report the performance of the model.

4. **Inference**
   (a) Plot train and test loss vs. iterations *(2 points)*
   (b) Plot classification accuracy vs. iterations *(2 points)*
   (c) Plot classification accuracy vs. number of samples used per class *(2 points)*
   (d) Discuss the plots and report your inference from them. *(6 points)*
   (e) Visualize filters from layers first and last Convolution Layers of the trained model. *(4 points)*

5. **Feature Extraction** *(5 points)*
   (a) Experiment using the intermediate Convolutional Layers as input to the Softmax Layer.
   (b) Train the network again on a subset of the data.
   (c) Discuss what you observe and provide empirical results to support it.

II. Urban Tribes Classification (25 points)
Here we will explore the application of another variant of Transfer Learning to the task of recognition of social styles of people. In particular, we will be looking at the classification of the Urban Tribes dataset into groups such as bikers, hiphop, hipster, formal, goth and many others!

Problem

1. **Read in the data**

(a) The dataset consists of images with filenames of the format *xxx_group_pic01234.jpg*. Here, the *xxx* stands for the group name. Read in the images, convert them to an array of pixel values with each row corresponding to one image and vector component corresponding to RGB values.
Tip: you can use Keras utilities to read in images as arrays

(b) Create the target labels of the images as categorical outputs of dimension 11.
Tip: you can use keras np_utils module

(c) Shuffle the entire data and prepare hold-out set by taking a portion of the training set.

2. **Training, Inference and Feature Extraction**
Repeat the exact same procedure as above (for Caltech256) on the Urban Tribes dataset.

III. Bonus Question (5%) : Temperature-based Softmax Regression

For this problem, we will be using the CalTech256 dataset.
Here we are going to explore using the Softmax Layer of VGG16, which is trained on the ImageNet dataset, as an input to our Softmax Layer to predict the classes of Caltech256. The idea here is to predict target labels for CalTech256 test images using their similarity to categories in ImageNet (this is a version of Shimon Edelman's idea in his paper: "Representation is Representation of Similarities").

I.e., instead of removing the VGG16 softmax layer, leave it on, and use it as input to the softmax for CalTech 256. This is not what is typically done, so this is a little bit of a research project. The idea is to use a "temperature parameter" on the softmax to reveal the similarities of the CalTech 256 objects to the 1000 categories in ImageNet. The temperature is inserted into the calculation of the last layer of VGG16 thus:

$$y_i = \frac{exp(a_i/T)}{\sum_j \exp(a_j/T)}, \text{ where T is the temperature parameter} \quad (1)$$

For further reading, refer to Hinton's Dark Knowledge paper.

Problem

1. Read in the Caltech256 images and prepare the dataset of image array and target labels.
2. Load the VGG16 Net using the starter file.
3. Modify the activation function of the VGG output layer with the temperature-based Softmax function.
4. Add a Softmax output layer that takes input from the VGG softmax layer to predict Caltech256 classes.
5. Train only the last Softmax layer on a subset of Caltech256 images, modifying the variable T to find the best T for this purpose. *Hint: since T is a scale parameter, try searching exponentially: Try 1, 2, 4, 8, 16.*
6. Report the best T that you found and discuss your results.