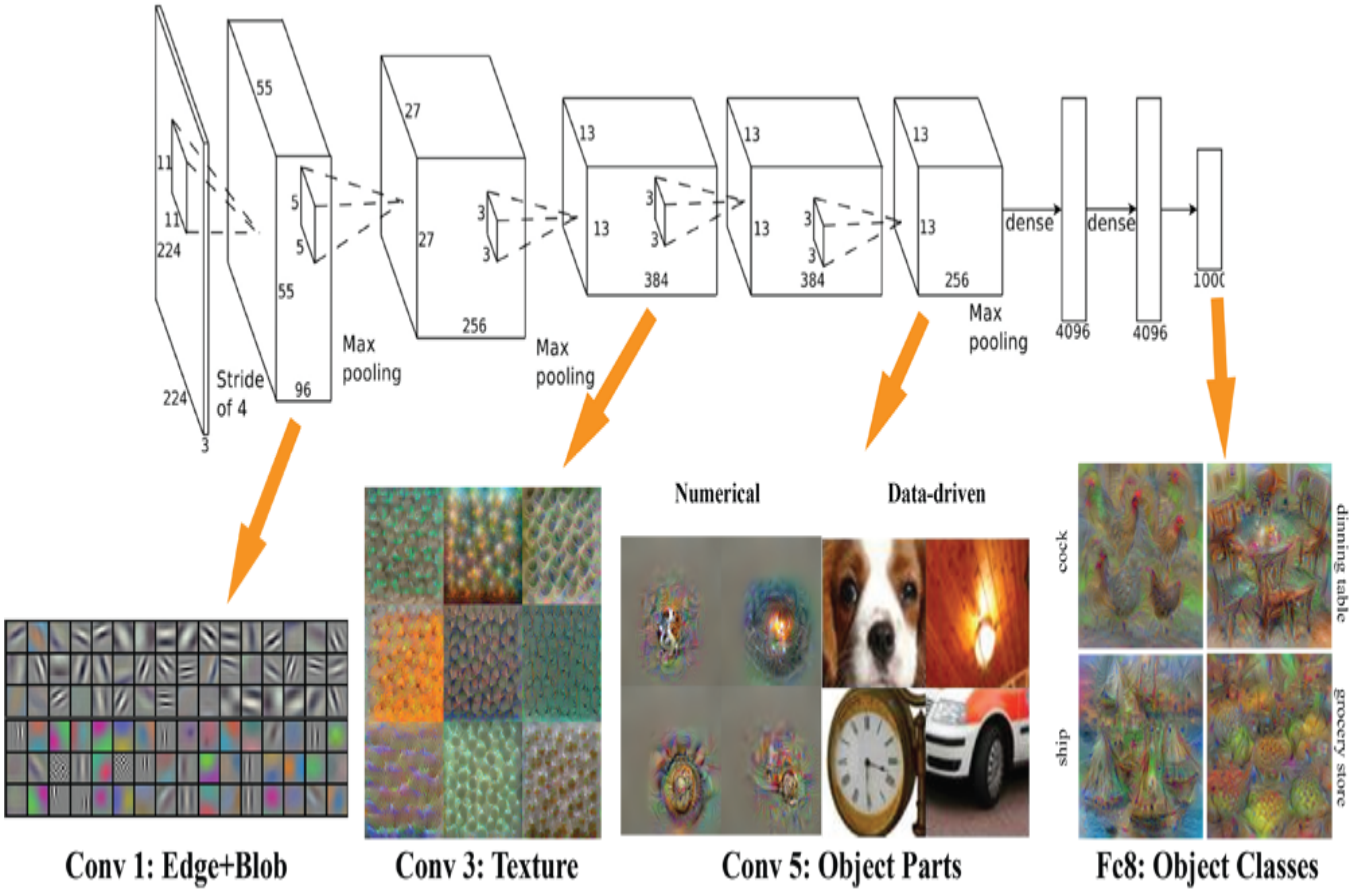

(deepNetVis.png) AlexNet / VGG-F network visualized by mNeuron (http://visiono3.csail.mit.edu/cnn_art/index.html).

Deep Learning with TensorFlow
Introduction to Computer Vision (..)

Please bear with us...

This is a coursework with new use of Google Cloud Platform, so please expect a few bumps in the mechanics. If you get stuck, please post on Piazza or visit TA hours, and we will help you through.

Logistics

- Files: projDeepLearning.zip (95 MB) (projDeepLearning.zip).
- Part 1: Questions
  - Questions + template: In the zip: questions/
  - Hand-in process: Gradescope (http://www.gradescope.com/) as PDF. Submit anonymous materials please!
- Part 2: Code
  - Writeup template: In the zip: writeup/
  - Required files: Use 'createSubmissionZIP.py'
  - Hand-in process: Gradescope (http://www.gradescope.com/) as ZIP file. Submit anonymous materials please!

Overview

We will design and train convolutional neural networks (CNNs) for scene recognition using the TensorFlow system. Remember scene recognition with bag of words (../proj3), which achieved 50 to 70% accuracy on 15-way scene classification? We're going to complete the same task on the 15 scenes database with deep learning and obtain a higher accuracy.

Task 1: Train a CNN to recognize scenes with a provided architecture and a small dataset of 1,500 training examples. This isn't really enough data, so we will try to:

- Add standardization (a kind of normalization).
- Add regularization.
- Add data augmentation. See here (https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced) for a gentle introduction, and here (https://tensorpack.readthedocs.io/modules/dataflow.imgaug.html) for all the supported augmentations in TensorPack.
- Change the architecture.

Forbidden functions: `sklearn.preprocessing`—please implement standardization yourself.

Task 2: Fine tune the VGG16 pre-trained CNN to recognize scenes, where the CNN was pre-trained on ImageNet. Fine tuning is the process of using pre-trained weights and only a few gradient updates to solve your problem, which itself might have a slightly different output. In our case, VGG was trained on ImageNet (1000-way classification) but we wish to use it for 15-way scene recognition. You will have to download the pre-trained weights from `/course/cs143/pretrained_weights/vgg16.npy`, or we have a copy of the model that you can download into Google Cloud Platform by calling `$> wget http://cs.brown.edu/courses/csci1430/proj4/vgg16.npy`.

These are the two most common approach to recognition problems in computer vision today: either train a deep network from scratch—if you have enough data—or fine tune a pre-trained network.

Rubric

- +50 pts: Task 1: Build a convolutional network with jittering, pre-processing of the input images to remove the mean and divide by the standard deviation (standardization), dropout regularization, and an additional convolutional layer which achieves at least 50% validation accuracy (for any single training epoch, not all) on the 15 scene database. *Note:* the stencil code will report *validation* accuracy and not test accuracy; we will only consider validation accuracy here.
- +30 pts: Task 2: Fine-tune VGG-F to achieve at least 85% validation accuracy (for any single training epoch, not all) on the 15 scene database.
- +05 pts: Writeup with design decisions and evaluation.
- +10 pts: Extra credit (up to ten points)
- -5*n pts: Lose 5 points for every time (after the first) you do not follow the instructions for the hand in format

Write up

We provide you with a LaTeX template `writeup/writeup.tex`. Please compile it into a PDF and submit it along with your code. *We conduct anonymous TA grading, so please don't include your name or ID in your writeup or code.*

Task:

- Describe your process and algorithm, show your results, describe any extra credit, and tell us any other information you feel is relevant.
- Report your classification performance for each step in Task 1, plus for Task 2.
- Include graphs of your loss function over time during training.

Extra Credit

Be sure to analyze in your writeup.pdf whether your extra credit has improved classification accuracy. Each item is "up to" some amount of points because trivial implementations may not be worthy of full extra credit. Some ideas:

- up to 10 pts: Gather additional scene training data (e.g., from the SUN database (http://groups.csail.mit.edu/vision/SUN/) or the Places database (http://places.csail.mit.edu/)) and train a network from scratch. Report performance on those datasets and then use the learned networks for the 15 scene database with fine tuning.
- up to 10 pts: Try a completely different recognition task. For example, try to recognize human object sketches (http://cybertron.cg.tu-berlin.de/eitz/projects/classifysketch/) (download the .png files). Or try to predict scene attributes (https://cs.brown.edu/~gen/sunattributes.html). The scene attributes are not one-vs-all (an image simultaneously has many attributes) so you'll need to configure TensorFlow accordingly. There are many other recognition data sets available.
- up to 10 pts: Produce visualizations using your own code or methods such as mNeuron (http://vision03.csail.mit.edu/cnn_art/index.html), Understanding Deep Image Representations by Inverting Them (https://github.com/aravindhm/deep-goggle), DeepVis (http://yosinski.com/deepvis), or DeepDream (https://github.com/google/deepdream).
- up to 10 pts: 1 point for every percent accuracy over 70% when training from scratch on the 15 scene database. The highest accuracy we've achieved is 66%, so we expect this to require many bells and whistles such as extensive jittering of the training data, carefully tuned network structure and per-layer training rates, etc.
- up to 10 pts: 1 point for every percent accuracy over 90% when fine-tuning from VGG-F. You don't get extra credit for switching to another network (like one trained on the Places database). The challenge here is to adapt a very big network to a relatively small training set.

Setting up Project 4 on Google Cloud Platform (recommended)

We will use Google Cloud Platform through GCP Tutorial (https://docs.google.com/document/d/1xPOX4mL4cJoJ5nBhSR88eVr1pvp77qXk3MsAWkvRgxE/edit?usp=sharing). Each account can only get a coupon once. Please make sure you are using your brown.edu account, not your cs.brown.edu or accounts of other domains. You will receive $50 credit to use. To stop consuming your free credit unnecessarily, please shut down the VM instance after using it each time.

Setting up Project 4 in Google Colaboratory

If you have used up all the coupons for GCP, you can also use Google Colab. Google Colab (https://colab.research.google.com/notebooks/welcome.ipynb) provides a platform with free GPUs. We can use Google Cloud Platform through the Colab Tutorial (https://docs.google.com/document/d/1LKSLvSy21PZOIs7vVZt_FpmxOdUQnhhNiIYiE188R-A/edit?usp=sharing).

Setting up Project 4 on a Department Machine (not recommended)

We have two additional Python virtual environments with TensorFlow and TensorPack installed. These can be used like our existing `/course/cs143/cs1430_env` environment, but they reside instead at `/course/cs143/tf_cpu` and `/course/cs143/tf_gpu`. BUT! *There are very few department lab machines with GPUs!* We will not tell you which ones, because this is *not recommended*. You will potentially run into all kinds of trouble, like other people remotely logging into your machine, setting off a GPU job, and then this killing your GPU job because the card ran out of memory.

We will not help you with these issues! Please use Google Cloud Platform!

Setting up Project 4 for TensorFlow on local machine (not recommended)

We will use TensorFlow through Python (https://www.python.org/). This is all set up on the departmental machines. For a personal machine, please visit the TensorFlow Website (https://www.tensorflow.org/install/) for installation instructions. We will also use an additional library called TensorPack (https://github.com/ppwwyyxx/tensorpack), which provides convenience functions.

For example, James' installation process on his laptop on Windows through PowerShell:

1. Install Python 3.6 (https://www.python.org/downloads/release/python-363/)
2. Use Python 3's package manager `pip3` to install TensorFlow: `PS> pip3 install --upgrade tensorflow`
3. Use Python 3's package manager `pip3` to install TensorPack: `PS> pip3 install --upgrade tensorpack`
4. Test the install:
    - `PS> python`
    - `>>> import tensorflow as tf`
    - `>>> hello = tf.constant('Hello, TensorFlow!')`
    - `>>> sess = tf.Session()`
    - `>>> print(sess.run(hello))`

If you have an NVIDIA GPU on a local machine and want to use it, it's a little more complicated to set up; please venture for yourself.

TensorFlow Tutorial

Before we start building our own deep convolutional networks, please look at Getting Started with TensorFlow (https://www.tensorflow.org/get_started/get_started). Please also go through the basic classification tutorial here (https://www.tensorflow.org/tutorials/keras/basic_classification), and the

CNN on MNIST example here (https://www.tensorflow.org/tutorials/estimators/cnn).

Credits

Project description and code by Aaron Gokaslan, James Tompkin, James Hays. Originally solely by James Hays, but translated to TensorFlow from MatConvNet by Aaron. GCP and Colab setting up by Ruizhao Zhu, Zhoutao Lu and Jiawei Zhang. We also referenced materials from Brown's CSCI1470 Deep Learning and welcomed the suggestions of its previous TA Philip Xu.