Introduction to TensorFlow

Pierce Spitler
@pspitler3

Background

- Previously Director of Data Science at eyeQ, Inc.
    - Created algorithms to recognize and understand faces
- Georgia Tech alum
- Currently Looking for New Opportunities

Intro after the Intro

- This talk/code assumes knowledge to the point of the tutorial on TensorFlow’s website: https://www.tensorflow.org/versions/r0.10/tutorials/index.html

- Code written in the base TF language, some new technology makes these steps easier (less code)

- Jupyter notebook, slides, and code will be made available

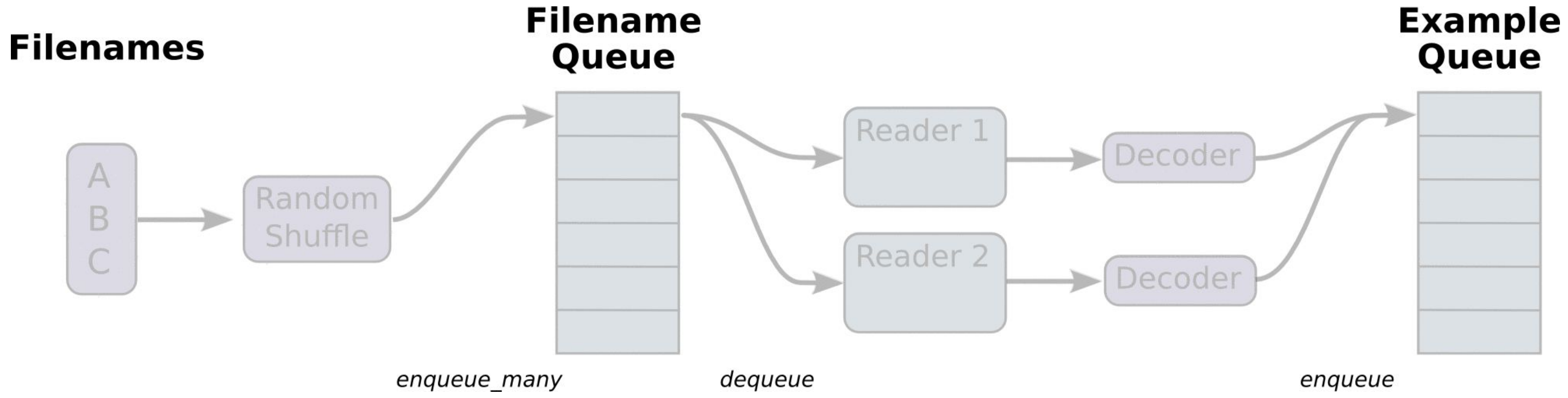
Topics Covered

- Data Readers
- Saving and restoring models
- Visualizing activations
- Transfer Learning
- Visualizing learned features through naive semantic segmentation
- Inception module

Data Readers

- Basic tutorials feed data into TensorFlow models using dictionaries
- There is a better way!
- Creating data queues that sit on top of some data set
- Enables a vehicle for random image transformations which artificially gives the model more data!
  - Random crops
  - Flip image left/right up/down
  - Randomly scale brightness/contrast
  - Perform image whitening
- Link Model to image directory instead of reading in the images first then feeding the model

Data Readers Continued

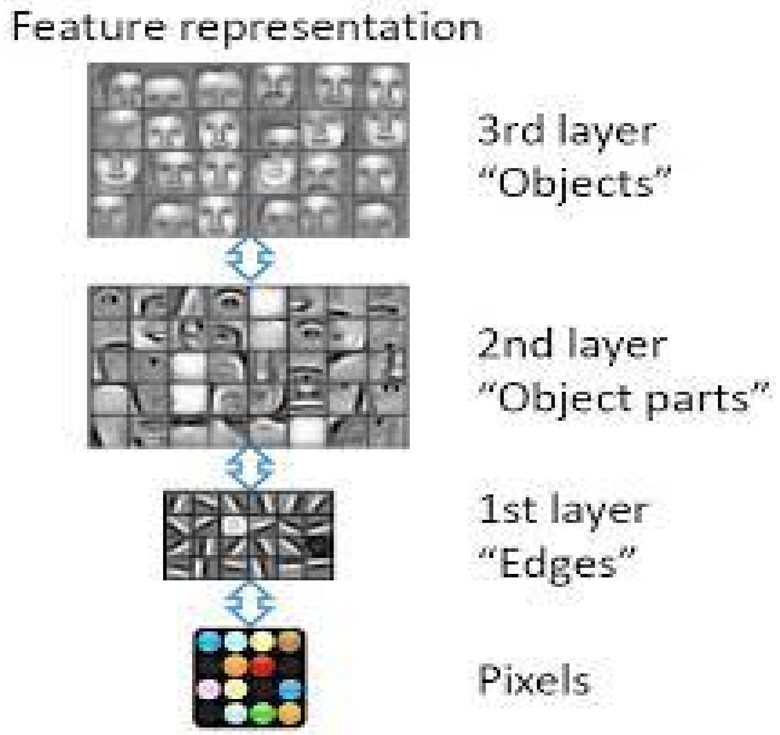


- This code will show readers that point at an image directory and also the creation/usage of a tfrecords binary file
- For more information: https://www.tensorflow.org/versions/r0.10/how_tos/reading_data/index.html

Saving and Restoring Models

- Saving and restoring models is covered in this code base
- A description of how TensorFlow handles this at a high level can be found here: https://www.tensorflow.org/versions/r0.10/how_tos/variables/index.html
- What it enables:
  - Cancelling/resuming the training of models
  - Adjusting the learning rate on the fly
  - Inference on data after the model is trained in a new environment
  - Restoring certain variables for transfer learning
- For a more robust way to save/restore models look into TensorFlow Serving (outside the scope of this talk): https://www.tensorflow.org/versions/r0.10/tutorials/tfserve/index.html

Visualizing Activations



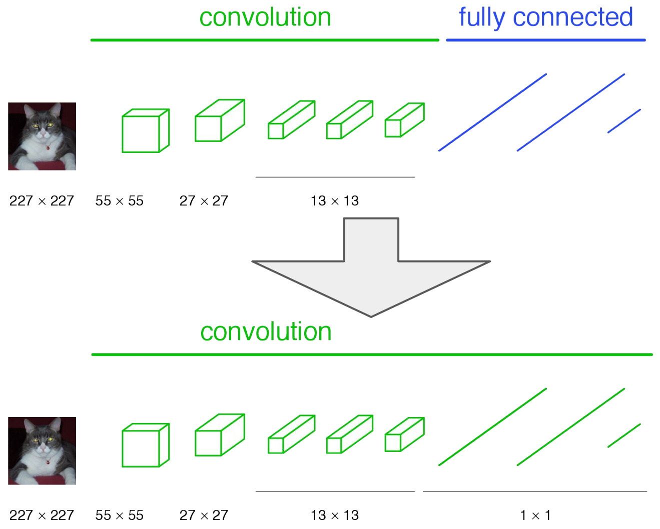
Source: Lee et al. (2009), ICML

Transfer Learning

- Transfer learning is taking weights trained by an existing model and “transferring” the knowledge gained to your use-case
- Since building-block features are learned through the layers of convolutional neural networks, they can be extended to new classes
- Some examples:
  - Using a pretrained CIFAR-10 model, adding a class, and retraining the final few layers to understand that new class
  - Using an existing architecture, but making the final layers fully convolutional to understand image segments (what I do in this code base)

Transfer Learning Continued

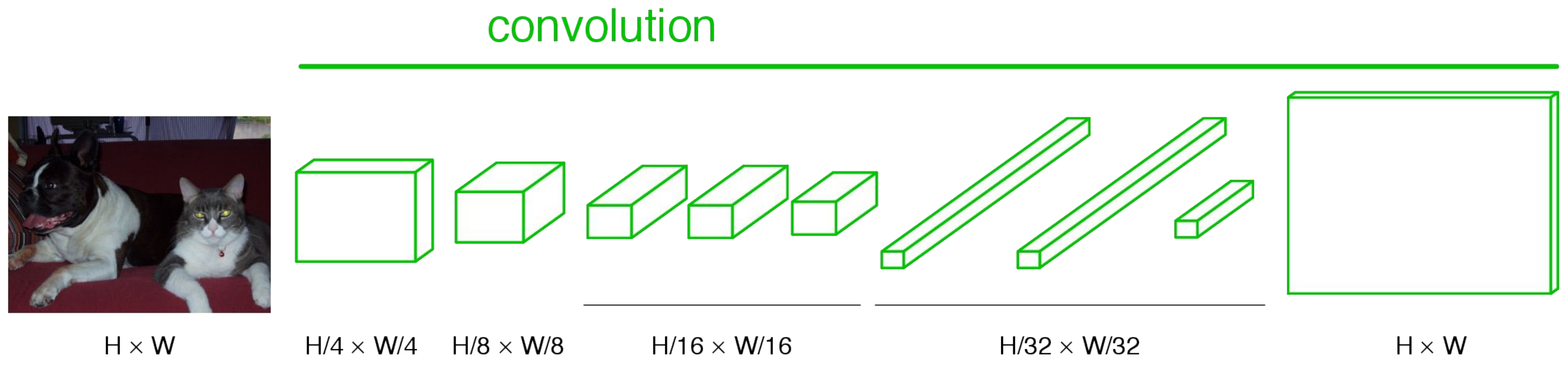
In this code, I take a trained classification algorithm, then retrain the final layers to be fully convolutional.



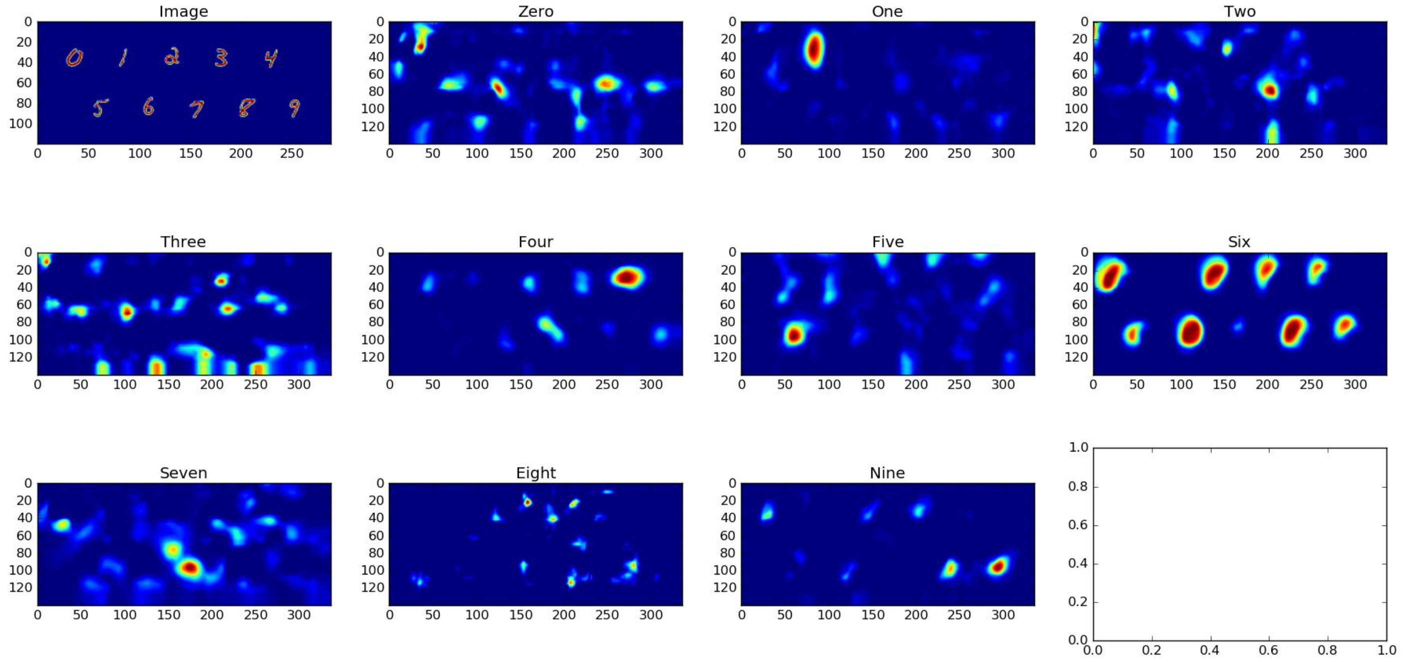
Source: https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf

Naive Semantic Segmentation

- By making the final layers fully convolutional, we can feed an image of any size through our network and upsample the output
- Since we don't have per-pixel labels we can vary the input size of the image to create a cascading pyramid effect



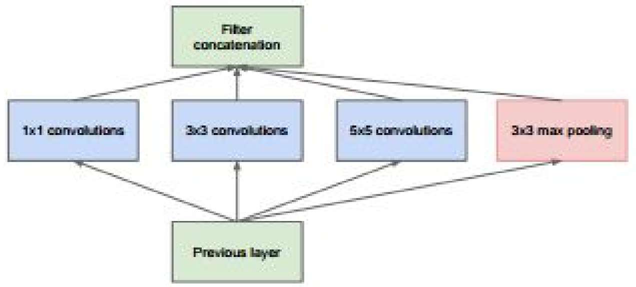
MNIST Output



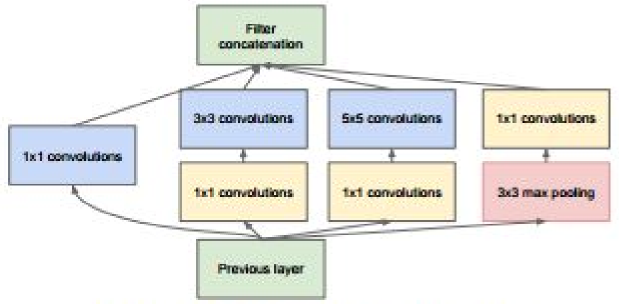
Inception Module

- The idea behind the inception architecture is to not choose what type of convolution to use, but instead use them all
- The dimensionality reduction is necessary to decrease the depth of the previous layer before running the convolutions to speed up computation
- Code attached to this presentation gives implementations of both modules

Source:
http://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

Figure 2: Inception module