Convolutional Networks II

Outline

- Text Classification with 1D CNNs
- Visualization of CNNs
- Transfer Learning

Text Classification

Sentence Classification

Convolutional Neural Networks for Sentence Classification

Yoon Kim
New York University
yhk255@nyu.edu

http://www.aclweb.org/anthology/D14-1181

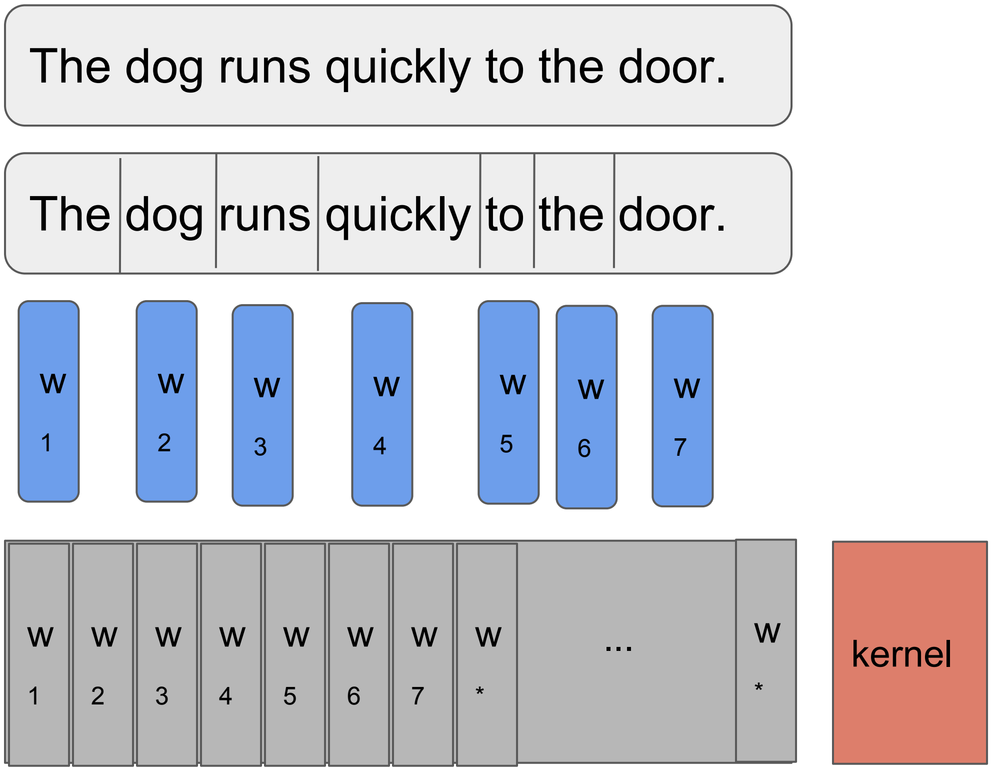
Sentence Classification

- Convolution works well for 2D images, but what about sequences, which are inherently 1 dimensional?
- And if we have a sequence of words, how to get to a numeric representation that would be well-suited for convolution?

Sentence Classification

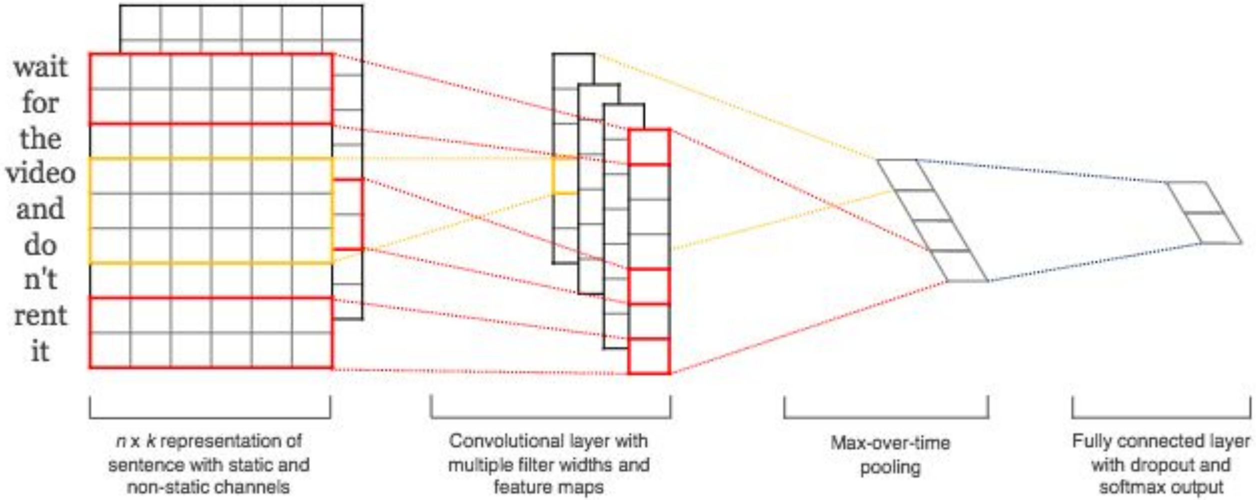
- We'll use word embeddings to generate a fixed size representation for each word
- Then all sentences will be either truncated or padded so that they are the same length
- Now we have a 2D array for each sentence, can use convolution
    - Our kernel sizes will be as "tall" as the embedding dimension, so we only really convolve in one direction (from left to right)
- The convolutional kernels will be learning which ordered sub-sequences of words are highly predictive of our target variable

Procedure



- Raw text sentence
- Break into component words (tokenize)
- Embedding vector for each word (fixed length)
- Stack word embeddings, and pad to standardized sequence length
- Kernels should be the same “height” as embedding dimension, but any width is fine

Sentence Classification



Summary

- As we'll see, the typical choice for sequence-based tasks is Recurrent Neural Networks
- But CNNs can also work very well - the temporal signals are captured by the kernels, instead of the recurrence
- Depending on your sequence, you can embed whatever would be most appropriate
  - character embeddings

Network Visualization and Understanding

What are CNNs learning?

1. Images yielding strong activations
2. Visualizing kernel heat maps
3. Visualization of feature maps
4. Visualizing kernel “preferences”

Further Reading

- Chapter 5 of F. Chollet *Deep Learning with Python*
- *http://cs231n.github.io/understanding-cnn/*

Find Training Images That Maximally Activate

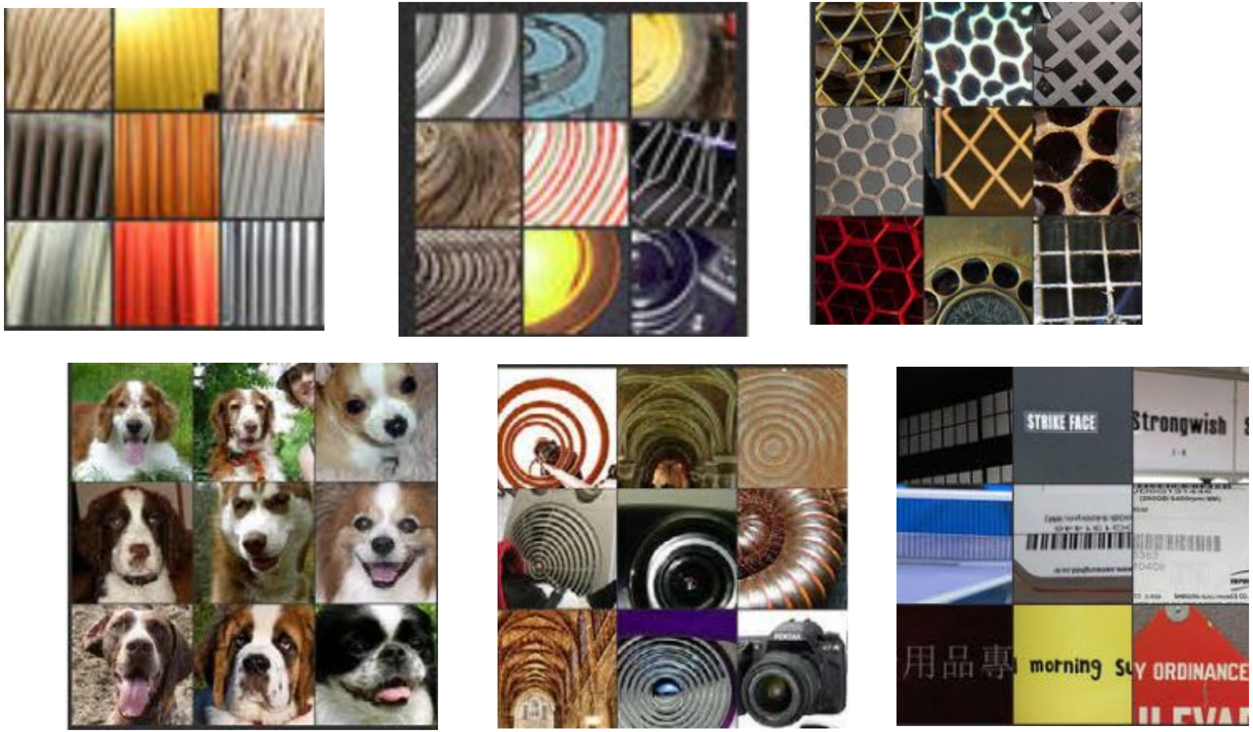
Neuron 1

Neuron 2

Neuron 3

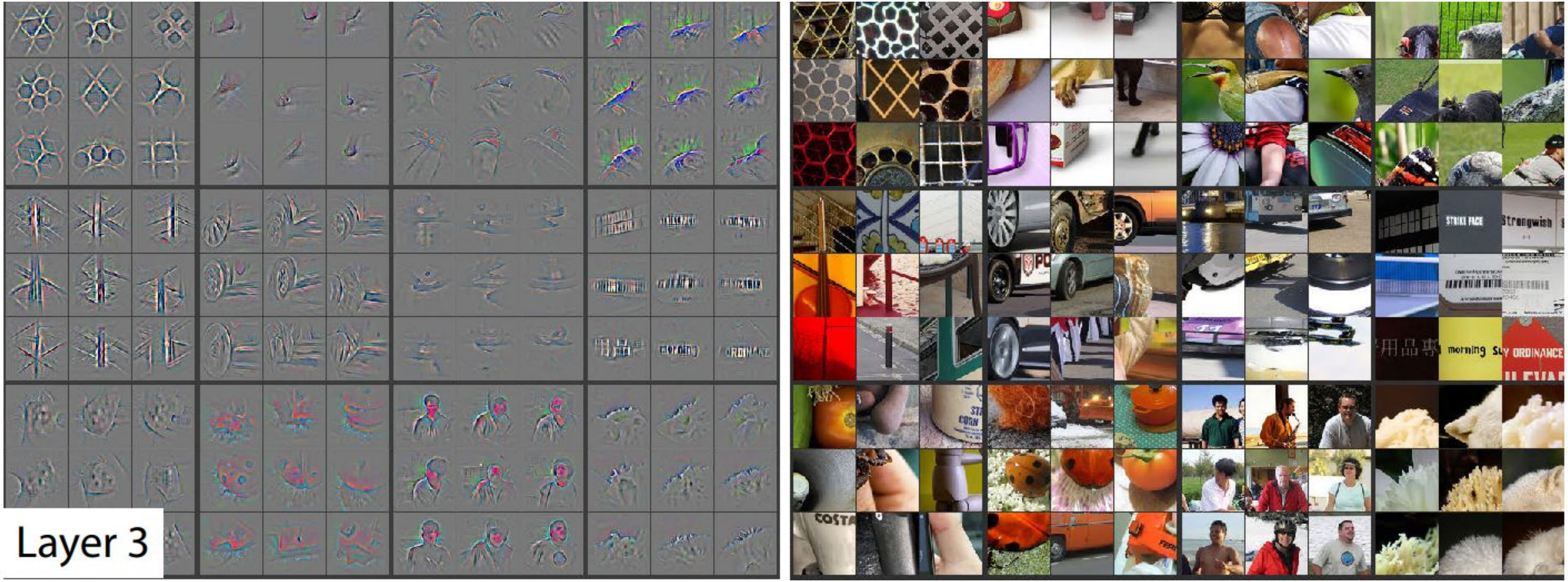
https://arxiv.org/pdf/1311.2524.pdf

Image Patches That Maximally Activate



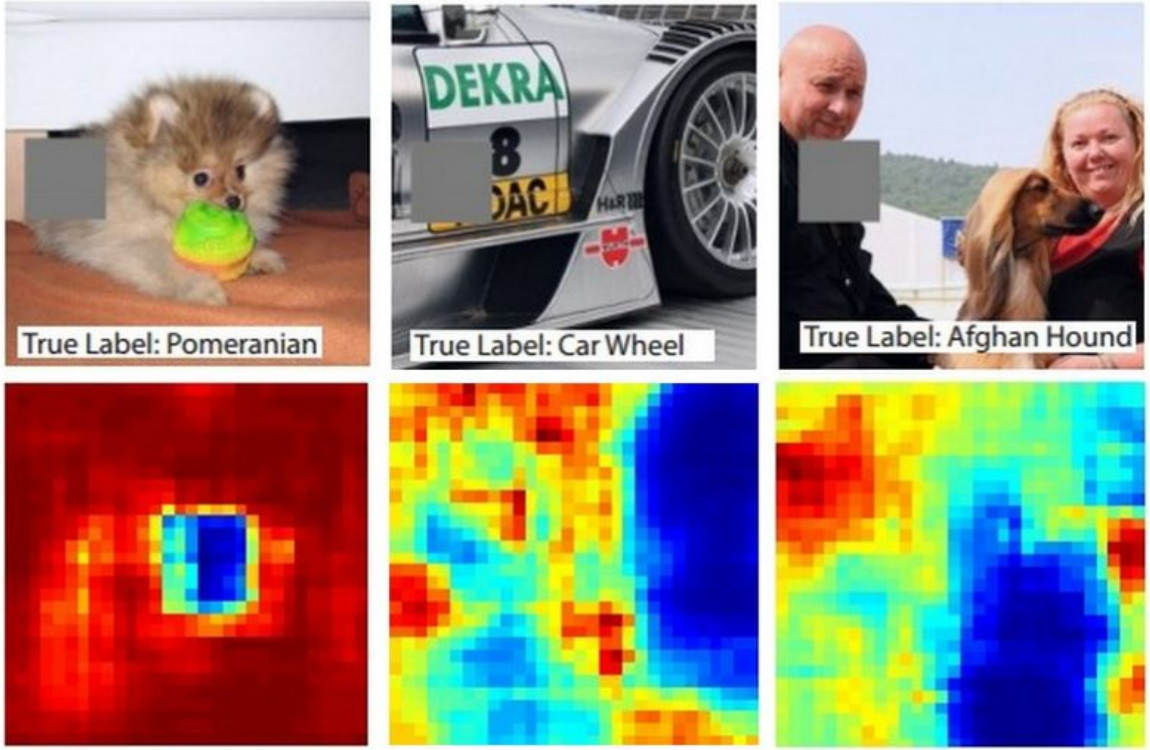
[Visualizing and Understanding Convolutional Networks]

Image Patches That Maximally Activate



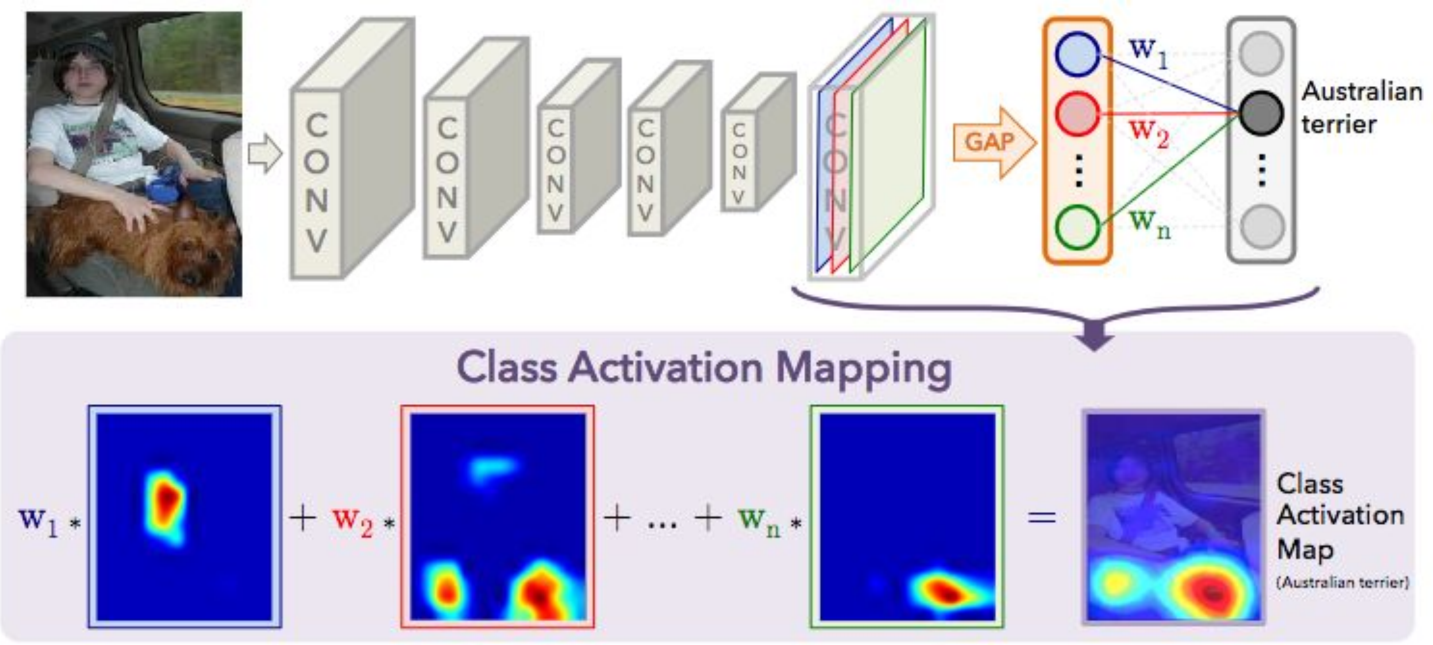
[Visualizing and Understanding Convolutional Networks]

Heat Maps - Occlusions



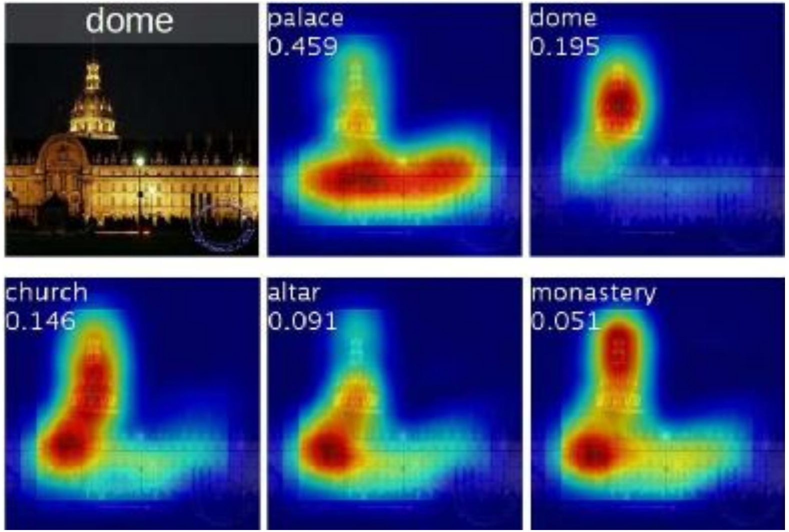
[Visualizing and Understanding Convolutional Networks]

Heat Maps - Class Activation Maps



http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf

Heat Maps - Class Activation Maps

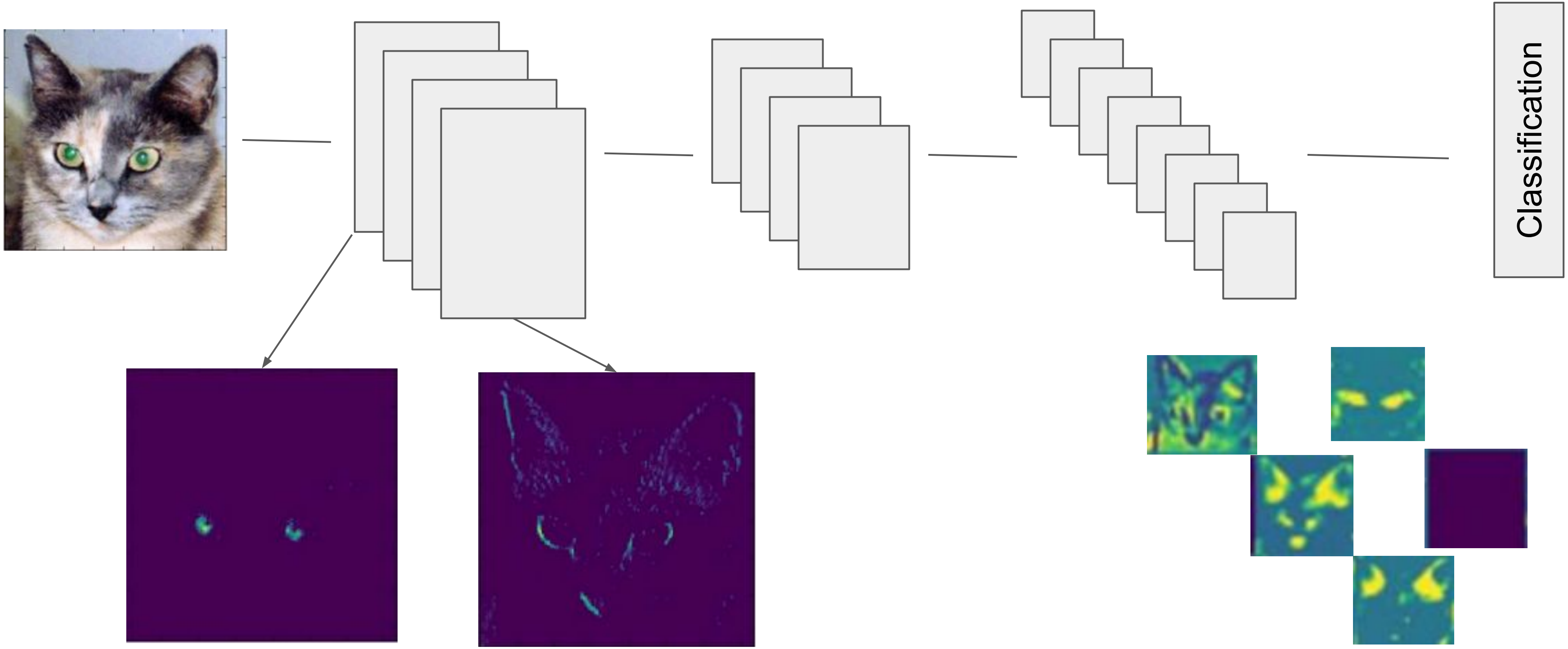


Top 5 classes based on CAMs
http://cnnlocalization.csail.mit.edu/Zhou_Learning_Deep_Features_CVPR_2016_paper.pdf

Visualizing Feature Maps

- Input an image into a previously-trained network
- Visualize a feature map at various locations in the network
- For features deep in the network, it will be challenging to match the feature map activations directly to pixel locations in input images - Deconvolutional Networks

Visualizing Feature Maps

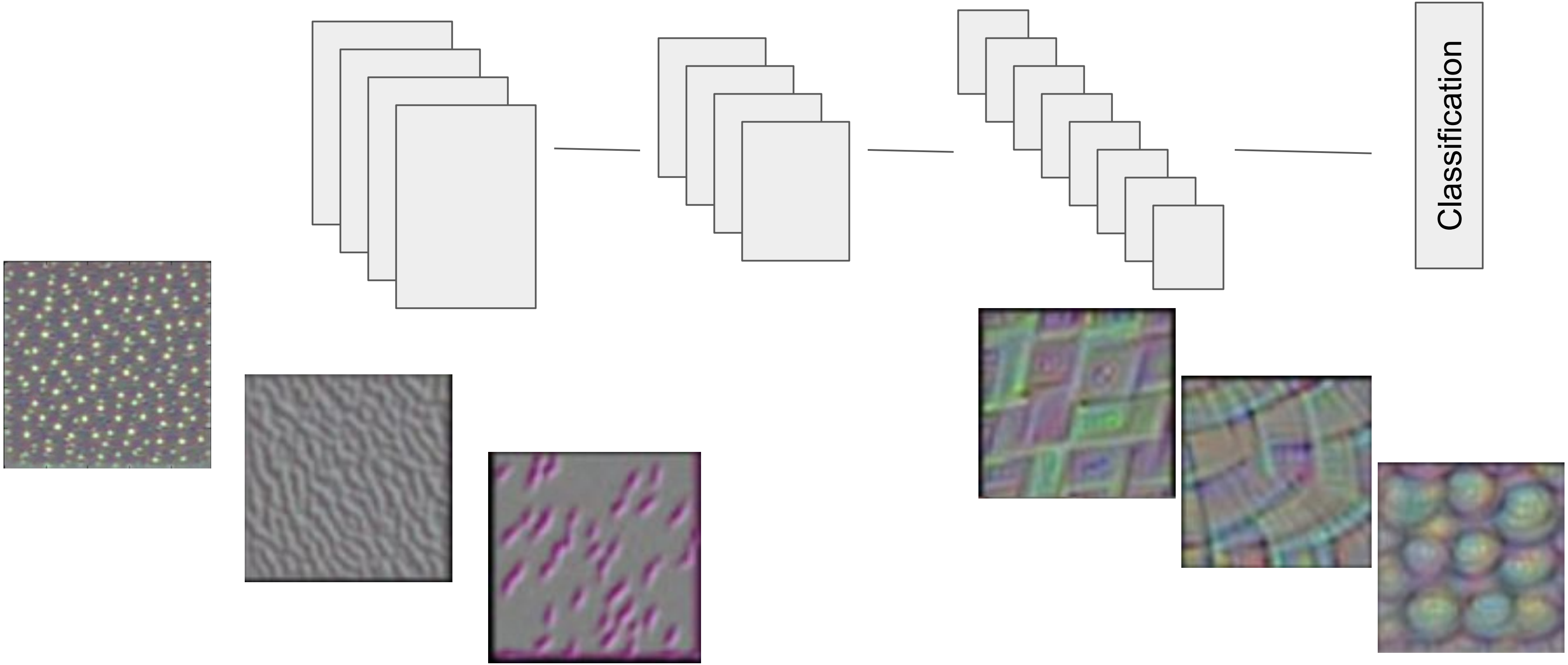


Visualizing Kernel “Preferences”

Gradient Descent over pixel space

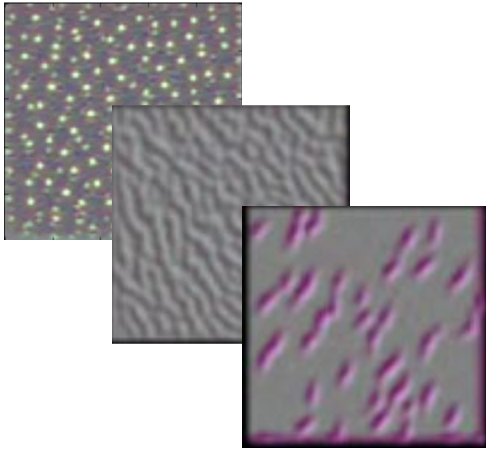
a. For any previously trained model, we can pick a kernel of interest and generate synthetic images that maximally activate that kernel
b. We begin with a completely random image and perform Gradient Descent over pixel space to adjust pixels so that they more maximally activate the kernel
c. Our Cost function is the kernel mean activation, and we optimize that function by adjusting pixels of our randomized starting image

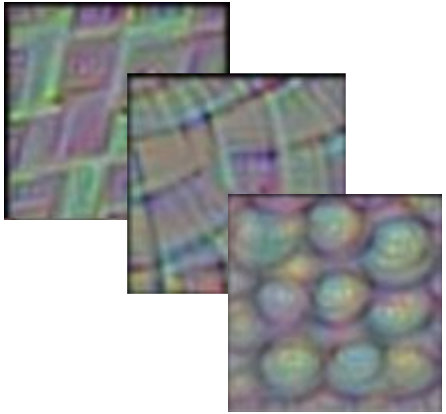
Visualizing Kernel “Preferences”



Visualizing Kernel “Preferences”

- Notice that kernels early in the network are tuned for simple visual features - edges, lines, blobs, color patches
- Kernel deep in the network are able to combine those simple features into complex abstractions and seem to be tuned for complicated objects



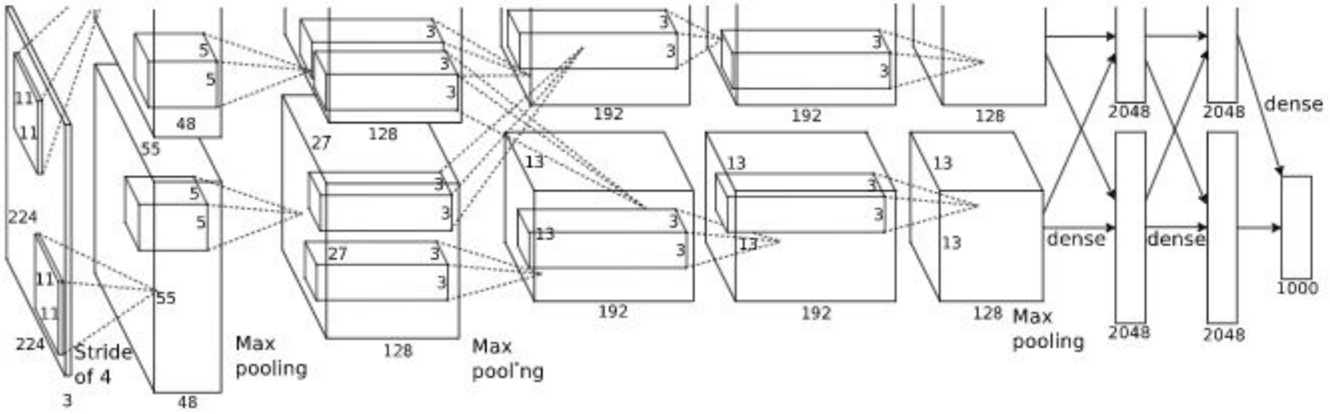


Transfer Learning

A Brief History of Important Deep CNNs

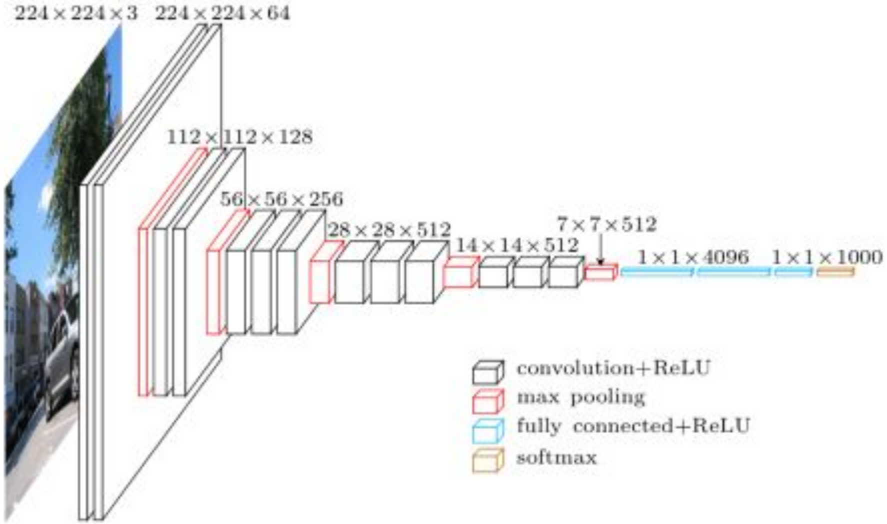
AlexNet

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf



A Brief History of Important Deep CNNs

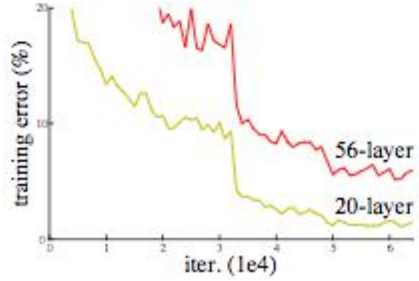
VGG16 https://arxiv.org/pdf/1409.1556.pdf

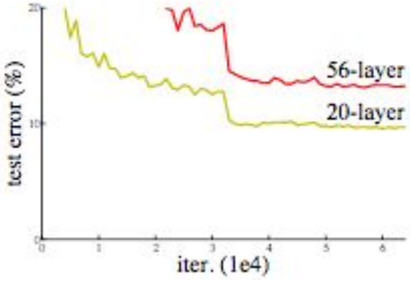


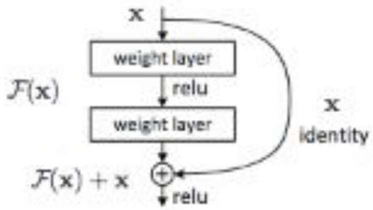
A Brief History of Important Deep CNNs

ResNet

https://arxiv.org/pdf/1512.03385.pdf



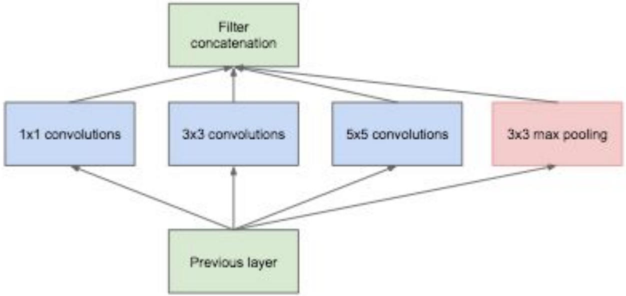




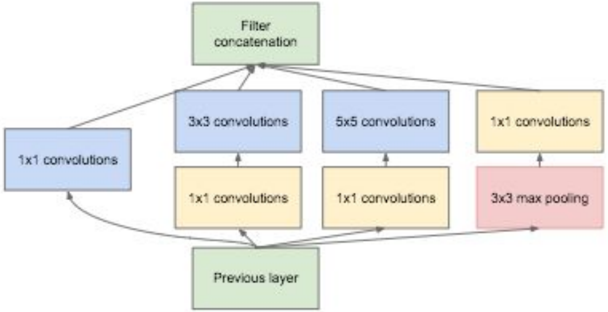
A Brief History of Important Deep CNNs

Inception

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



(a) Inception module, naïve version



Shoulders of giants

All of these networks (and more) are available pre-trained in Keras

- https://keras.io/applications/

Easy to explore these networks and their outputs without having to retrain them

- Don’t re-invent the wheel - each of these networks took days or weeks to train, and consumed a lot more computing resources than we have available to us.
- It is beneficial to be able to take advantage of these.

Transfer Learning

- A deep CNN trained on a large and diverse image dataset (like ImageNet) will have inevitably learned kernels and representations that are very useful for lots of kind of images
- For any given network (like VGG) the last few layers are specific to the task at hand - classification over a particular set up classes
- But the bulk of the convolutional stack is *probably* quite generic in recognizing features that are common to all images - edges, curves, and compositions
- With **transfer learning**, we can use a previously model and *reuse* most of it for a new task
- Very hard to train CNN models with small datasets, but transfer learning opens up lots of exciting possibilities

Transfer Learning

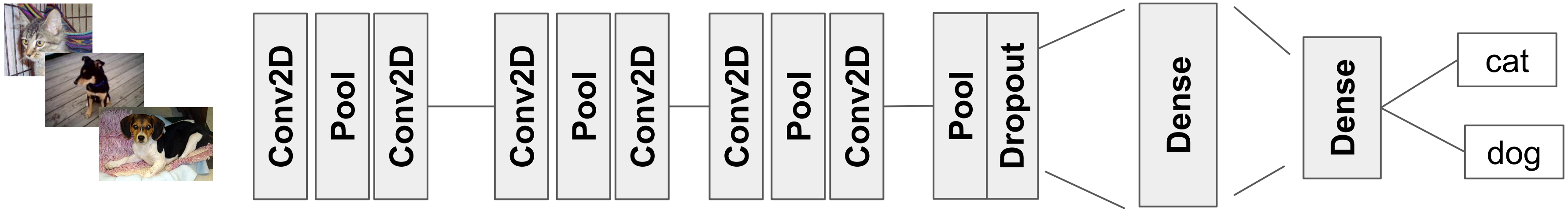
Image Classification - Cats vs Dogs

Option 1
Train a New model from scratch

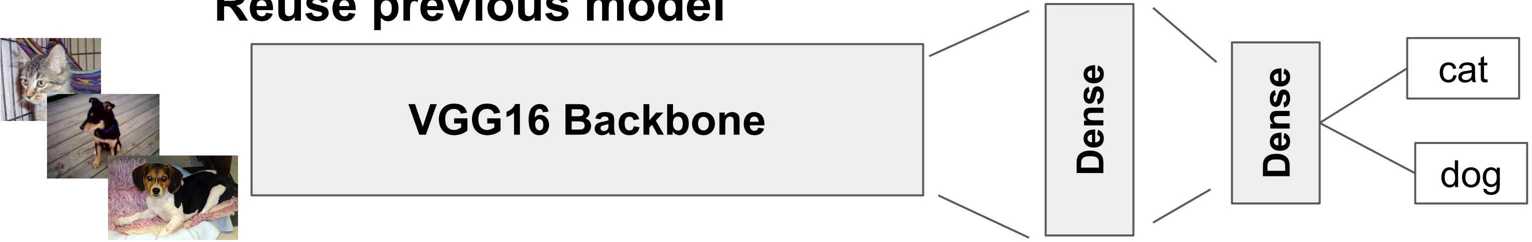
Option 2
Reuse previous model

Transfer Learning

Option 1
Train a New model from scratch

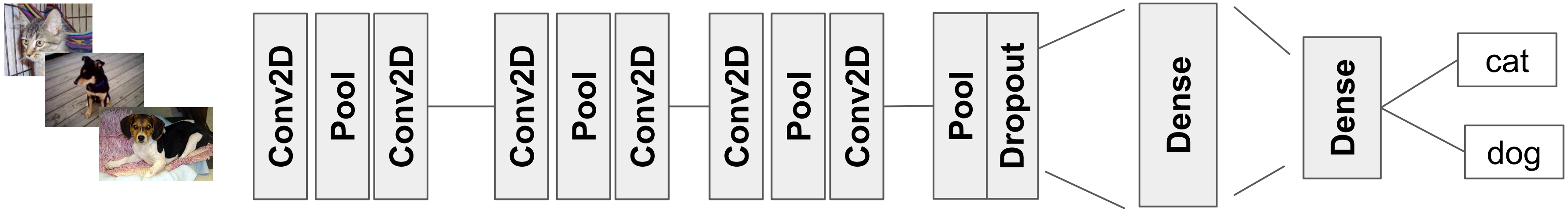


Option 2
Reuse previous model



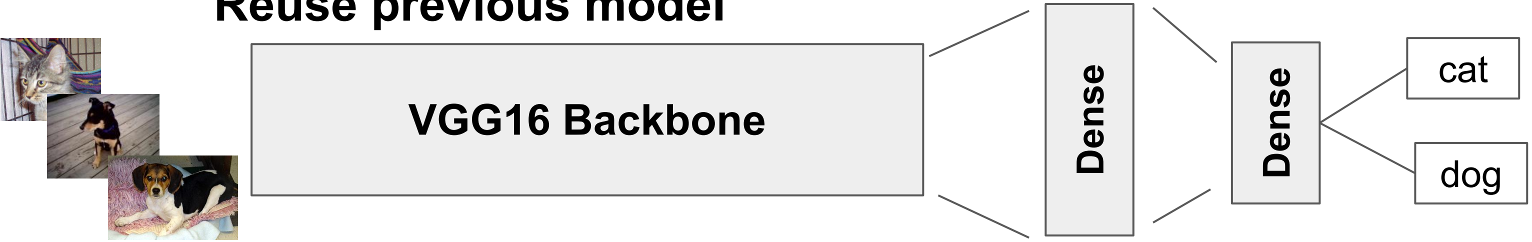
Transfer Learning

Option 1
Train a New model from scratch

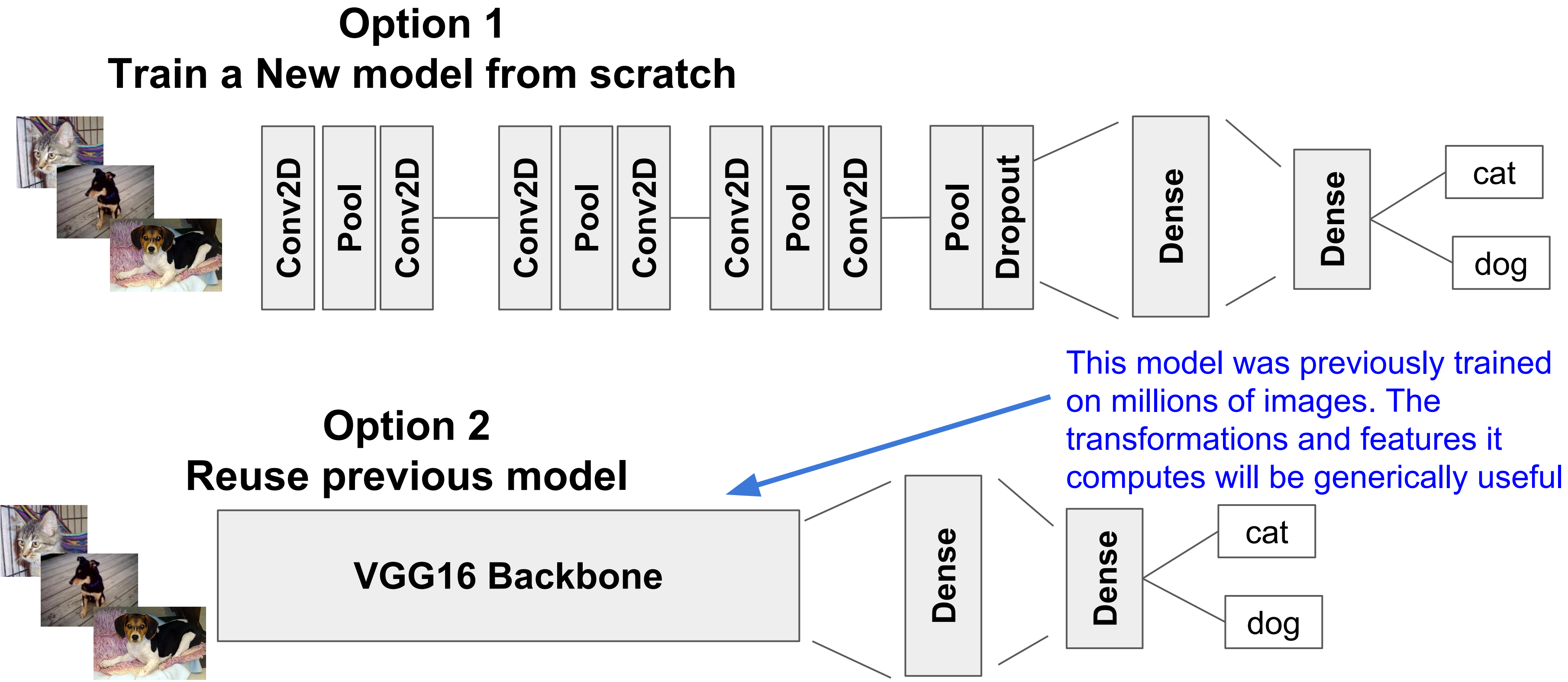


Will be difficult to effectively learn all these weights if we have a tiny amount of training data

Option 2
Reuse previous model



Transfer Learning



Types of Transfer Learning

1. **Feature Extraction**
   a. Use a base convolution stack to extract a feature representation of each image (no free parameters). Then train any classifier of your choice (even liner methods) on those feature vectors.
   b. Most commonly - use an ANN for the classifier (perhaps a multi-layer one). Then train the model with backprop, but only train the Dense layer weights, and make sure that the convolutional weights can’t change (“freeze” these weights).
2. **Fine Tuning**
   a. Let some of the convolutional weights be allowed to adjust during training. The details are up to you to decide how much of the convolutional stack remains frozen.
   b. Remember that the earliest parts of the stack will be learning generic features like edges. Deeper into the stack, the kernels are probably tuned to the original training task. You need to evaluate how similar your images are to the original training task, and how many (or how few) new images you have for your challenge.

Transfer Learning

- With transfer learning it is easy to train powerful classifiers even if you have only a few hundred or thousand images for your problem domain (common in medical use cases).
- Further reading
    - https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
    - http://cs231n.github.io/transfer-learning/