Transfer Learning

PIRL, POSTECH

Hanul Roh

Transfer Learning

- Exploit the representation power learned from large scale dataset (e.g. ImageNet)

- Reason why…
  - Too few dataset
  - Too much time to train model from scratch
  - Good initialization in a neural network

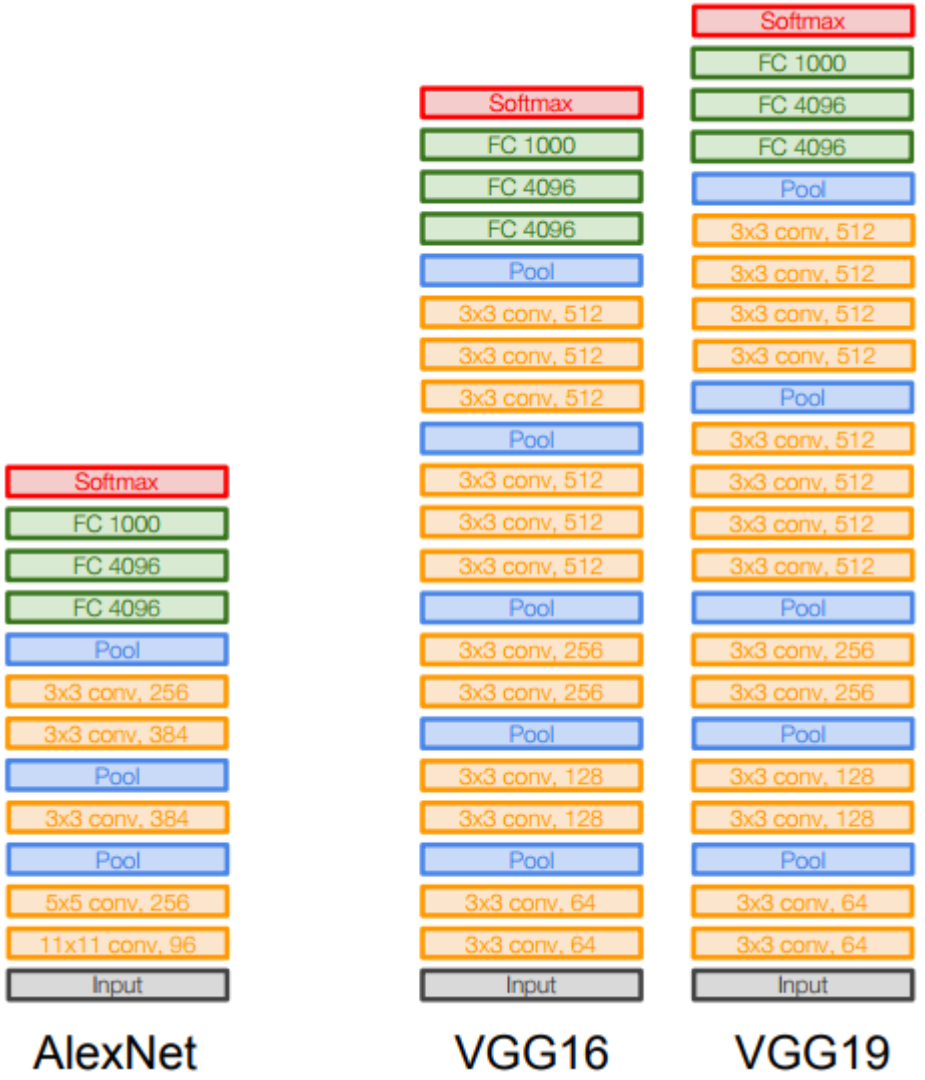
VGGNet

[Simonyan and Zisserman, 2014]

- Small filters, Deeper networks

8 layers (AlexNet, 2012)
-> 16 – 19 layers (VGGNet)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2



VGGNet

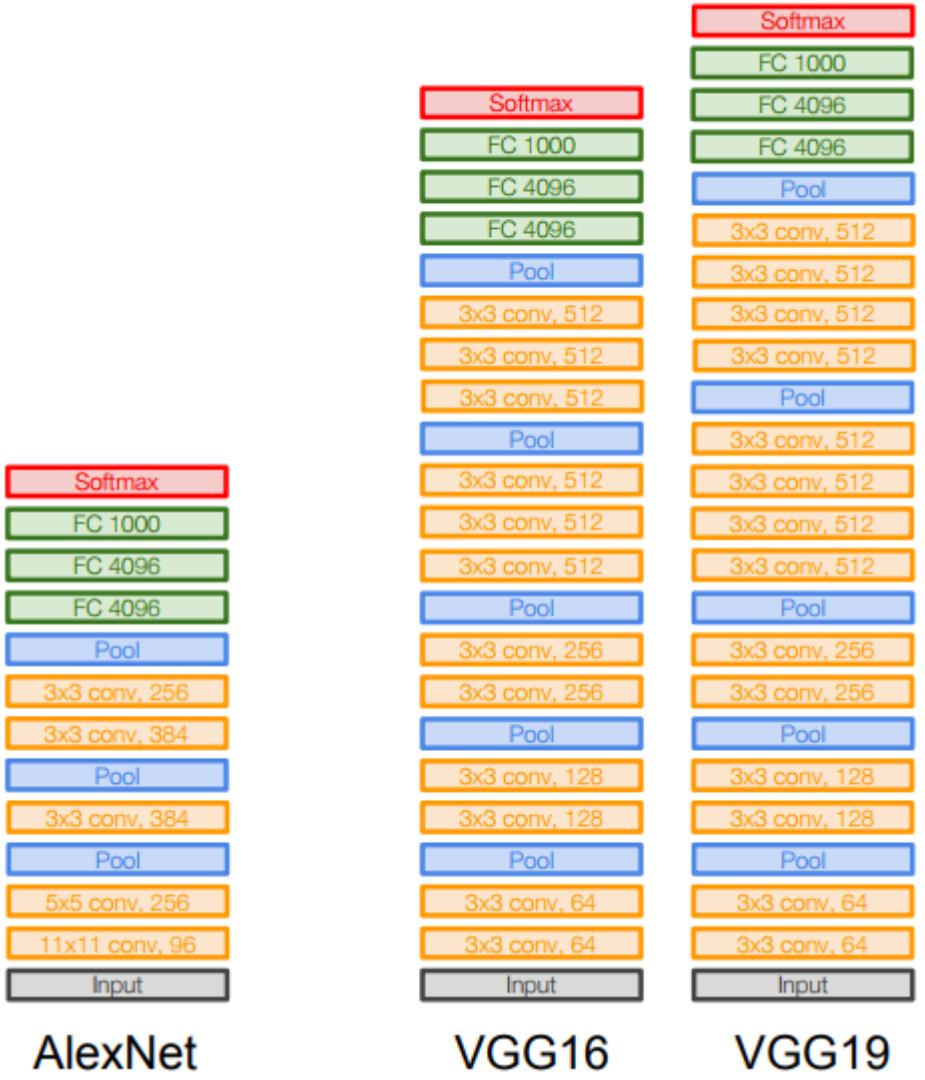
[Simonyan and Zisserman, 2014]

- Why use smaller filters? (3x3 conv)

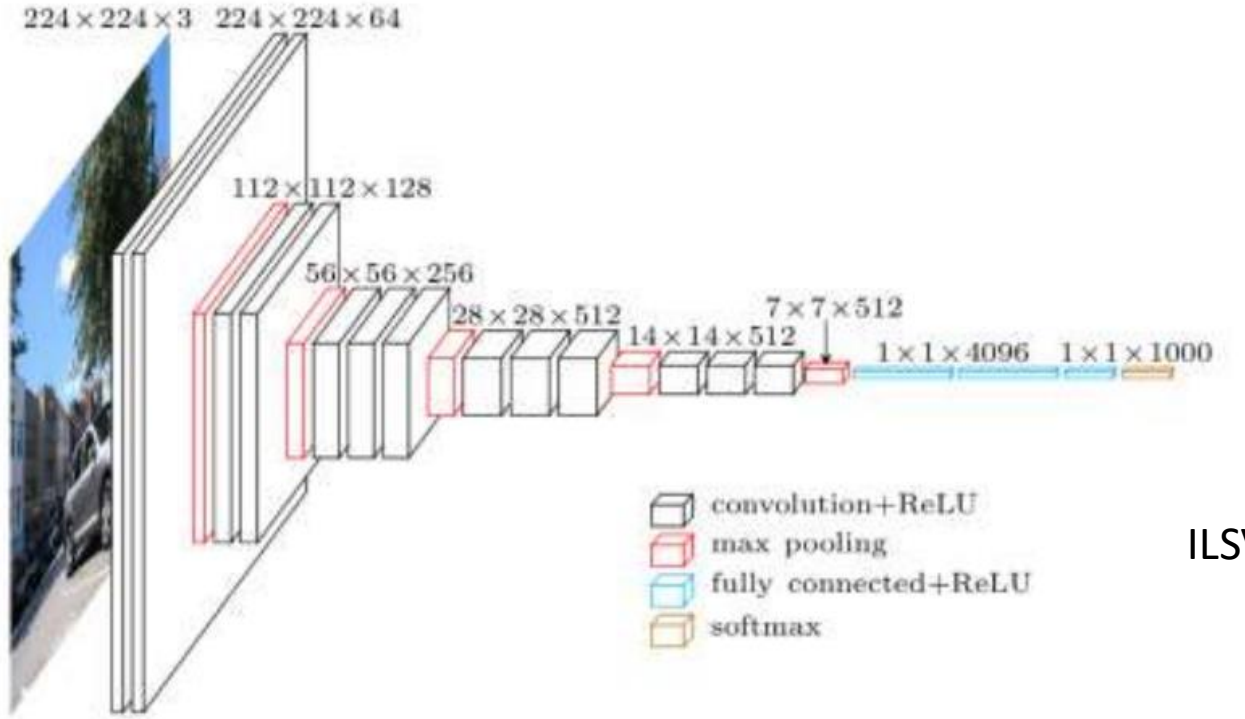
Stack of three 3x3 conv (stride 1) layers has same **effective receptive fields** as one 7x7 conv layer

But deeper, more non-linearities

And fewer parameters: 3*(3*3*C) vs. 7*7*C for C channels per layer



VGG-16 Network



The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

ILSVRC'14 2nd in classification, 1st in localization

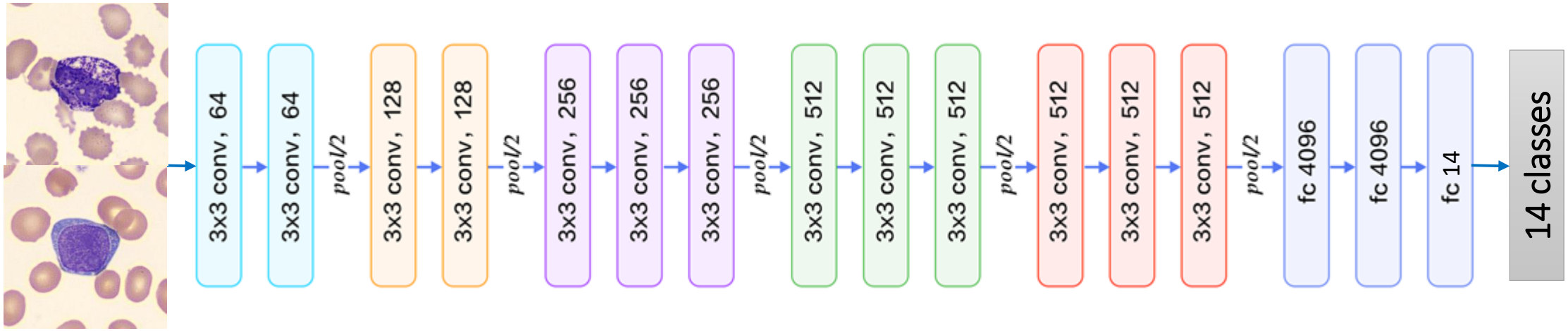
ImageNet Large Scale Visual Recognition Challenge

IMAGENET

- 1000 categories, 1.2M training data, 100K test data (ILSVRC 2014)

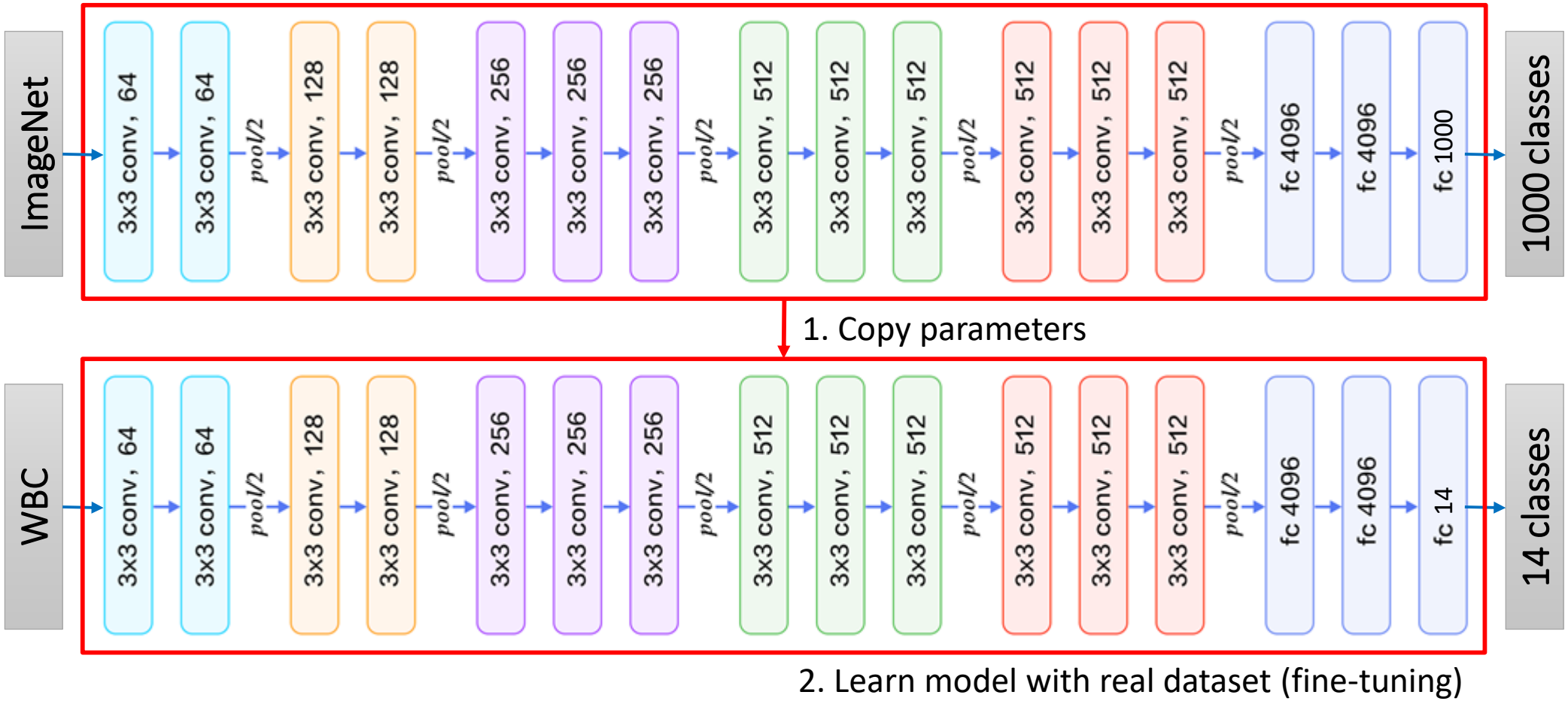
Other datasets summary here: http://deeplearning.net/datasets/ (Music, text, artificial, faces, etcs)

WBC Classification using VGG-16 Network

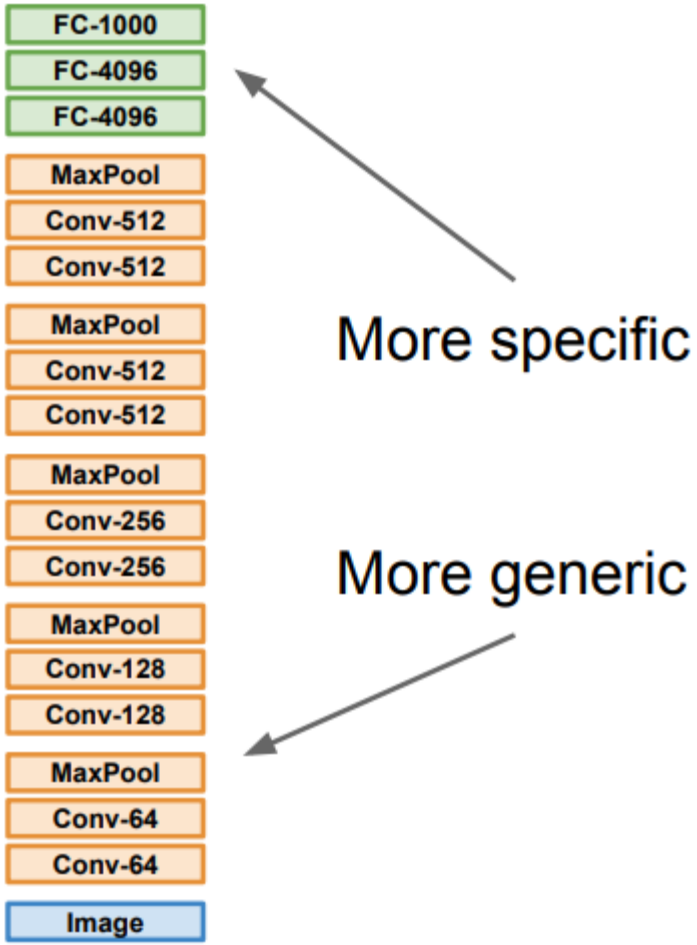


- Imbalanced dataset (total 0.1M images)
  - Some class less than 200 images, the other more than 40000 images

Fine-tuning VGG-16



Fine-tuning Strategy



	Very similar dataset	**Very different dataset**
Very little data	Use Linear Classifier on top layer	Very difficult problems
Quite a lot of data	Fine-tuning a few layers	Fine-tuning more layers

VGG Architecture

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Pre-trained Models

- How to use popular pre-trained models?
- In tensorflow, use TF-Slim library
  - AlexNet, VGGNet, ResNet, Inception, MobileNet
  - https://github.com/tensorflow/models/tree/master/research/slim

Pre-trained Models

- Need two steps:
  - Construct a network equivalent to the pre-trained model with same name space in checkpoints from TF-Slim

Use tf.contrib.slim to construct pre-trained models (here, vgg-16)

```
from nets import vgg
with tf.contrib.slim.arg_scope(vgg.vgg_arg_scope()):
    logits, _ = vgg.vgg_16(images, num_classes=10, is_training=True)
```

* **vgg.py** includes construction functions for vgg models
  - defined in https://github.com/tensorflow/models/tree/master/research/slim/nets

* Or, you can define own function, but should follow the name space in the checkpoint

```
vgg_16/conv1/conv1_1/weights
vgg_16/conv1/conv1_1/biases
vgg_16/conv1/conv1_2/weights
vgg_16/conv1/conv1_2/biases
vgg_16/conv2/conv2_1/weights
vgg_16/conv2/conv2_1/biases
vgg_16/conv2/conv2_2/weights
vgg_16/conv2/conv2_2/biases
vgg_16/conv3/conv3_1/weights
vgg_16/conv3/conv3_1/biases
vgg_16/conv3/conv3_2/weights
vgg_16/conv3/conv3_2/biases
vgg_16/conv3/conv3_3/weights
vgg_16/conv3/conv3_3/biases
vgg_16/conv4/conv4_1/weights
vgg_16/conv4/conv4_1/biases
vgg_16/conv4/conv4_2/weights
vgg_16/conv4/conv4_2/biases
vgg_16/conv4/conv4_3/weights
vgg_16/conv4/conv4_3/biases
vgg_16/conv5/conv5_1/weights
vgg_16/conv5/conv5_1/biases
vgg_16/conv5/conv5_2/weights
vgg_16/conv5/conv5_2/biases
vgg_16/conv5/conv5_3/weights
vgg_16/conv5/conv5_3/biases
```

  - Select parameters to be copied and copy them using tf.train.Saver()

Pre-trained Models

- Need two steps:
  - Construct a network equivalent to the pre-trained model with same name space in checkpoints from TF-Slim
  - Select parameters to be copied and copy them using tf.train.Saver()

Select parameter variables

```
slim = tf.contrib.slim
exclude_layers = ['vgg_16/fc8']
variables_to_restore =
    slim.get_variables_to_restore(exclude=exclude_layers)
```

Restore the parameters

```
restorer = tf.train.Saver(variables_to_restorer)
sess = tf.Session()
restorer.restore(sess, save_path=checkpoint_path)
```

```
variables_to_restore = []
for var in tf.global_variables():
    excluded = False
    for exclusion in exclude_layers:
        if var.op.name.startswith(exclusion):
            excluded = True
            break
    if not excluded: variables_to_restore.append(var)
```

Exercise

- Download git from
  - https://github.com/mixcheck/cnntutorial
- Train vgg-16 using CIFAR-10 from scratch
- Train vgg-16 using CIFAR-10 from pre-trained model
  - Only fc8 layer
  - All layers
- Train ResNet-V1-50 from pre-trained model
  - https://github.com/tensorflow/models/tree/master/slim#Pretrained