What Is TL?

- Introduction & Motivation
- Adapting Neural Networks
- Process

Transfer Learning

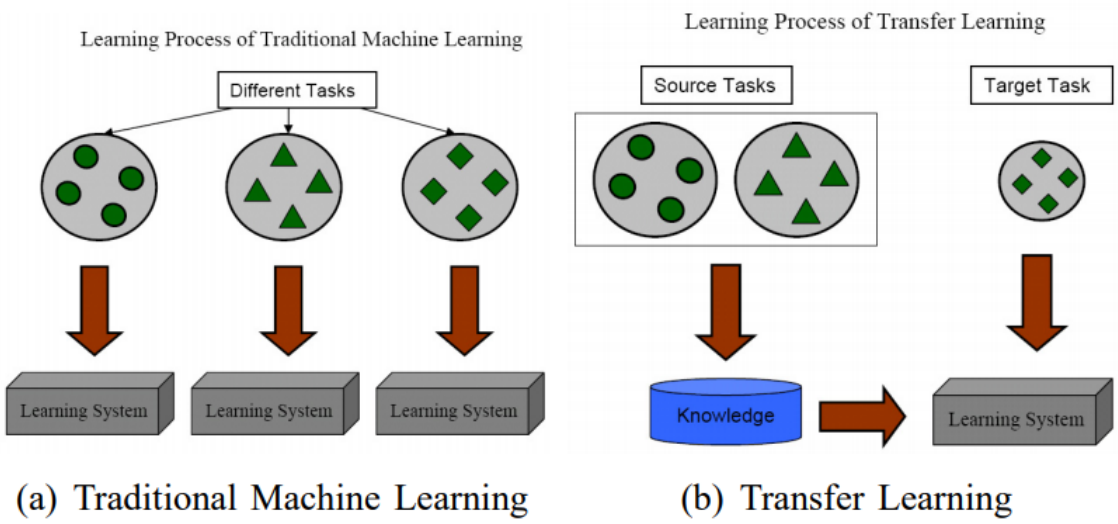
Transfering the knowledge of one model to perform a new task.

"Domain Adaptation"

Motivation

- Lots of data, time, resources needed to train and tune a neural network from scratch
  - An ImageNet deep neural net can take weeks to train and fine-tune from scratch.
  - Unless you have 256 GPUs, possible to achieve in 1 hour (https://research.fb.com/publications/accurate-large-minibatch-sgd-training-imagenet-in-1-hour/)
- Cheaper, faster way of adapting a neural network by exploiting their generalization properties

Traditional vs. Transfer Learning



(image: Survey on Transfer Learning (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.147.9185&rep=rep1&type=pdf))

Transfer Learning Types

Type	Description	Examples
Inductive	Adapt existing **supervised** training model on new **labeled** dataset	Classification, Regression
Transductive	Adapt existing **supervised** training model on new **unlabeled** dataset	Classification, Regression
Unsupervised	Adapt existing **unsupervised** training model on new **unlabeled** dataset	Clustering, Dimensionality Reduction

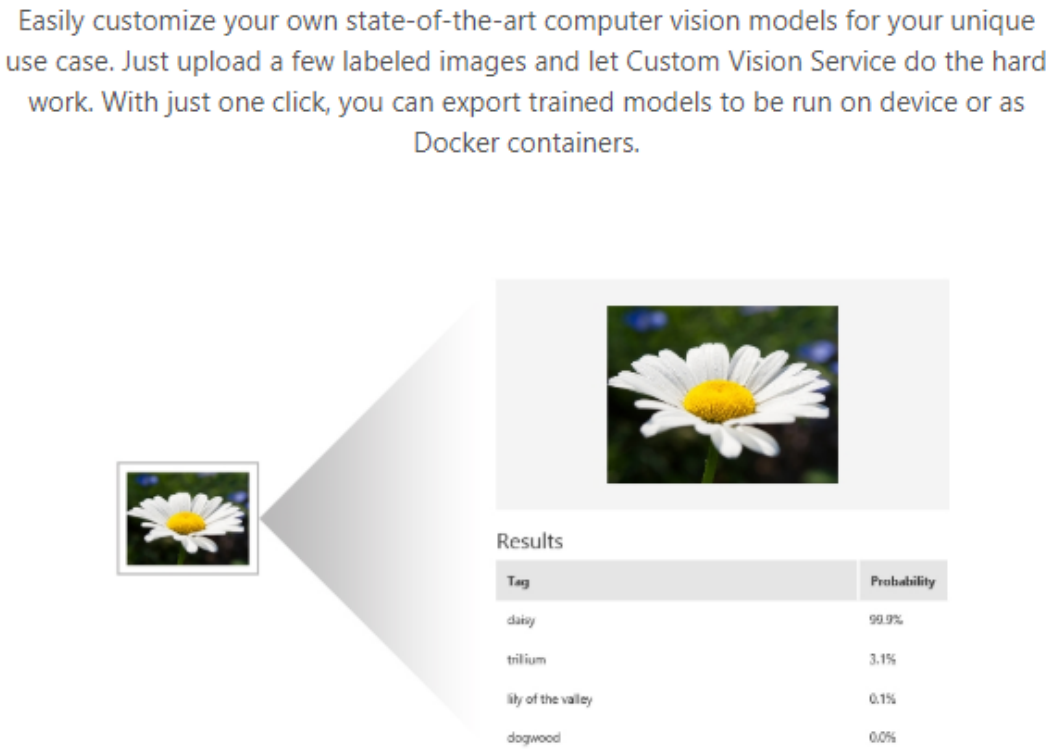
Transfer Learning Applications

- Image classification (most common): learn new image classes
- Text sentiment classification
- Text translation to new languages
- Speaker adaptation in speech recognition
- Question answering

Transfer Learning Services

Transfer learning is used in many "train your own AI model" services:

- just upload 5-10 images to train a new model! in minutes!



(image: https://azure.microsoft.com/en-us/services/cognitive-services/custom-vision-service/ (https://azure.microsoft.com/en-us/services/cognitive-services/custom-vision-service/))

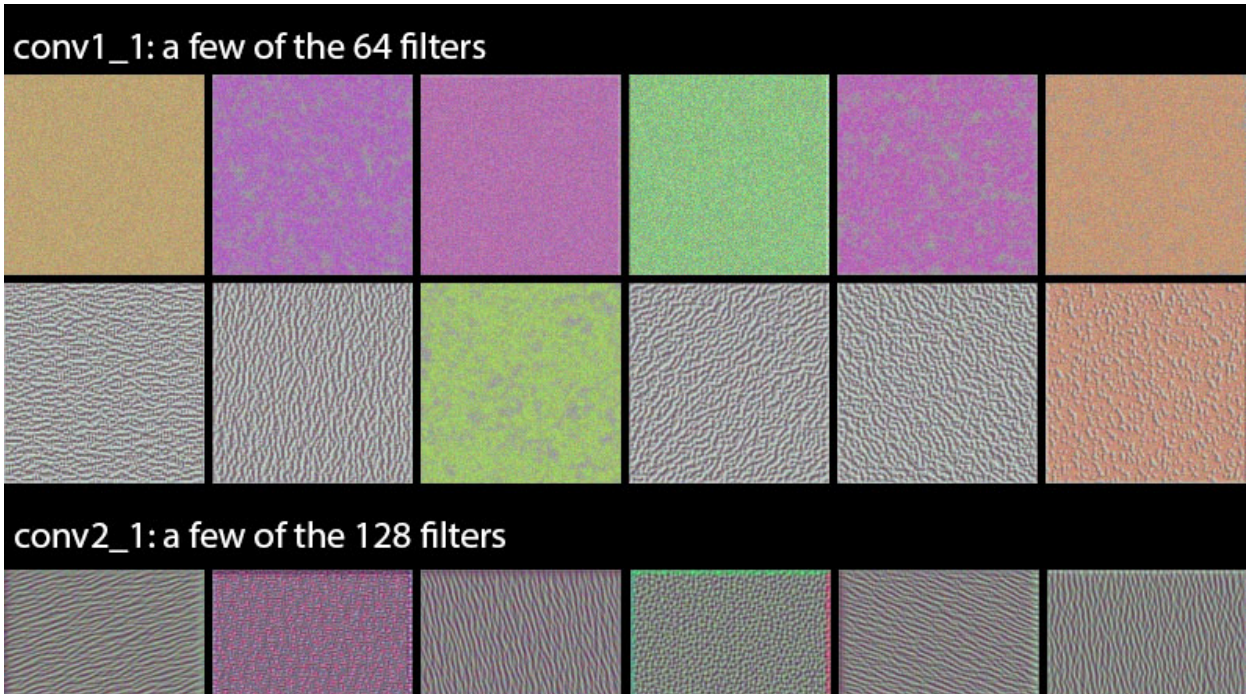
Transfer Learning in Neural Networks

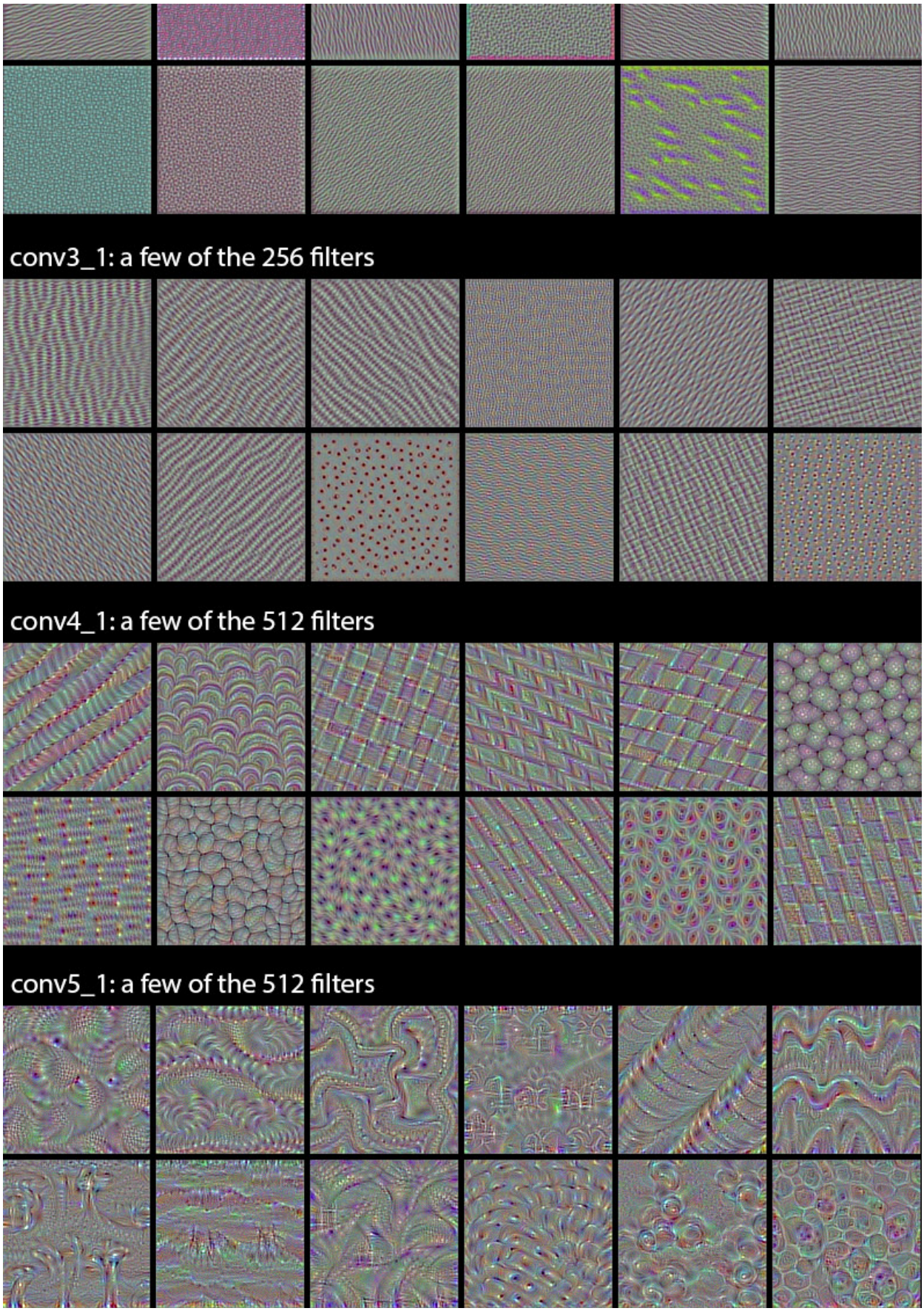
Neural Network Layers: General to Specific

- Bottom/first/earlier layers: general learners
  - Low-level notions of edges, visual shapes
- Top/last/later layers: specific learners
  - High-level features such as eyes, feathers

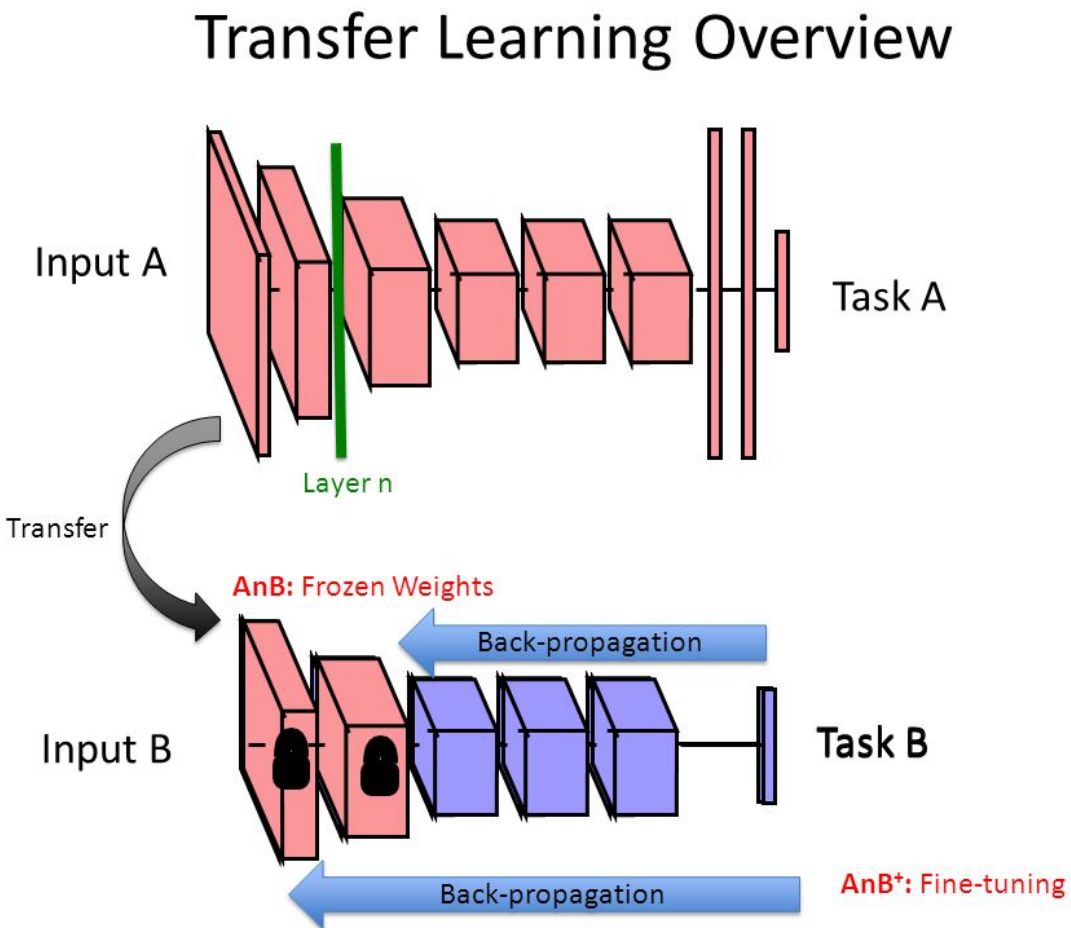
Note: the top/bottom notation is confusing, I'd avoid it

Example: VGG 16 Filters




conv3_1: a few of the 256 filters
conv4_1: a few of the 512 filters
conv5_1: a few of the 512 filters

https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html (https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html)

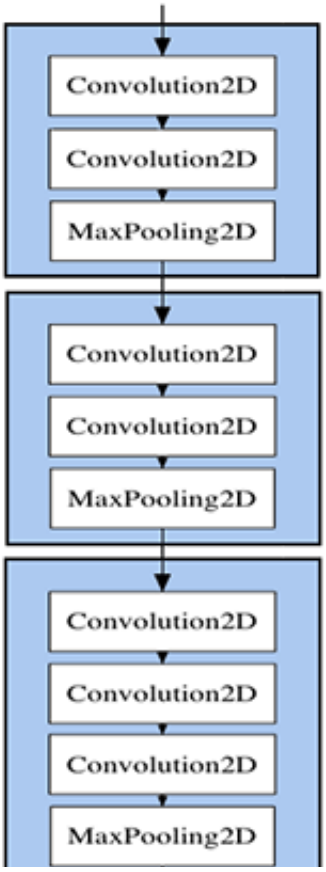


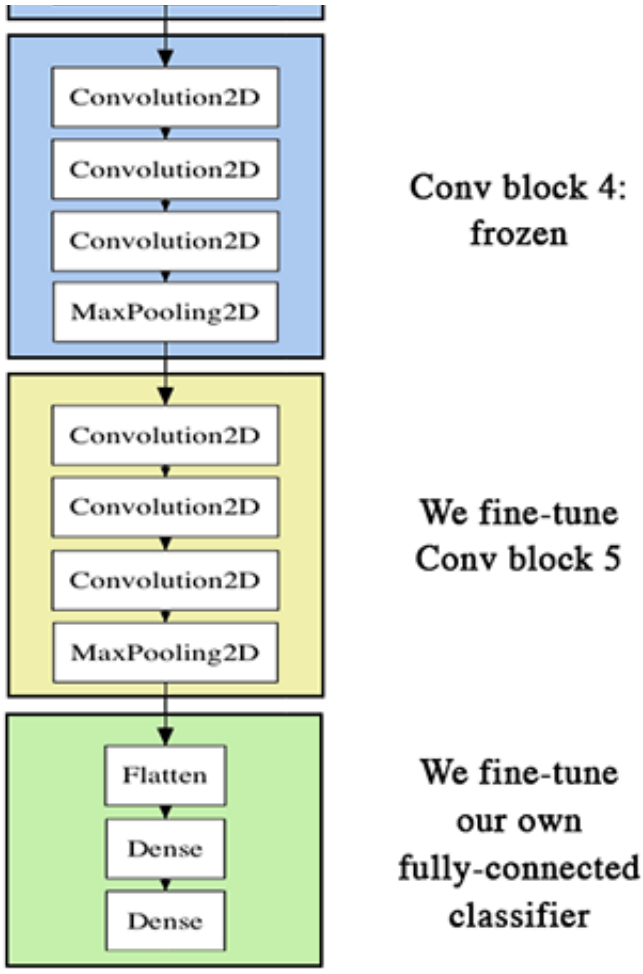
(image: Aghamirzaie & Salomon (http://slideplayer.com/slide/8370683/))

Process

1. Start with pre-trained network
2. Partition network into:
    - Featurizers: identify which layers to keep
    - Classifiers: identify which layers to replace
3. Re-train classifier layers with new data
4. Unfreeze weights and fine-tune whole network with smaller learning rate

Freezing and Fine-tuning



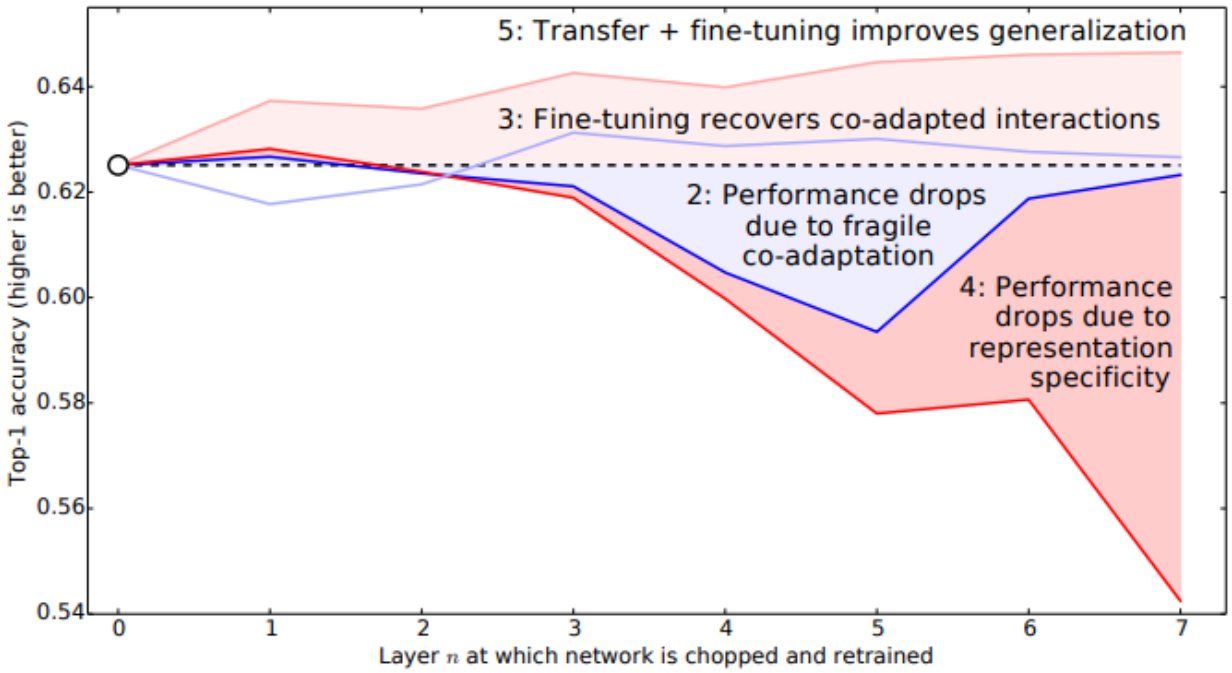


(image: http://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html (http://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html))

Which layers to re-train?

- Depends on the domain
- Start by re-training the last layers (last full-connected and last convolutional)
  - work backwards if performance is not satisfactory

Example



(image: http://arxiv.org/abs/1411.1792 (http://arxiv.org/abs/1411.1792))

When and how to fine-tune?

Suppose we have model A, trained on dataset A Q: How do we apply transfer learning to dataset B to create model B?

Dataset size	Dataset similarity	Recommendation
Large	Very different	Train model B from scratch, initialize weights from model A
Large	Similar	OK to fine-tune (less likely to overfit)
Small	Very different	Train classifier using the earlier layers (later layers won't help much)
Small	Similar	Don't fine-tune (overfitting). Train a linear classifier

https://cs231n.github.io/transfer-learning/ (https://cs231n.github.io/transfer-learning/)

Learning Rates

- Training linear classifier: typical learning rate
- Fine-tuning: use smaller learning rate to avoid distorting the existing weights
    - Assumes weights are close to "good"