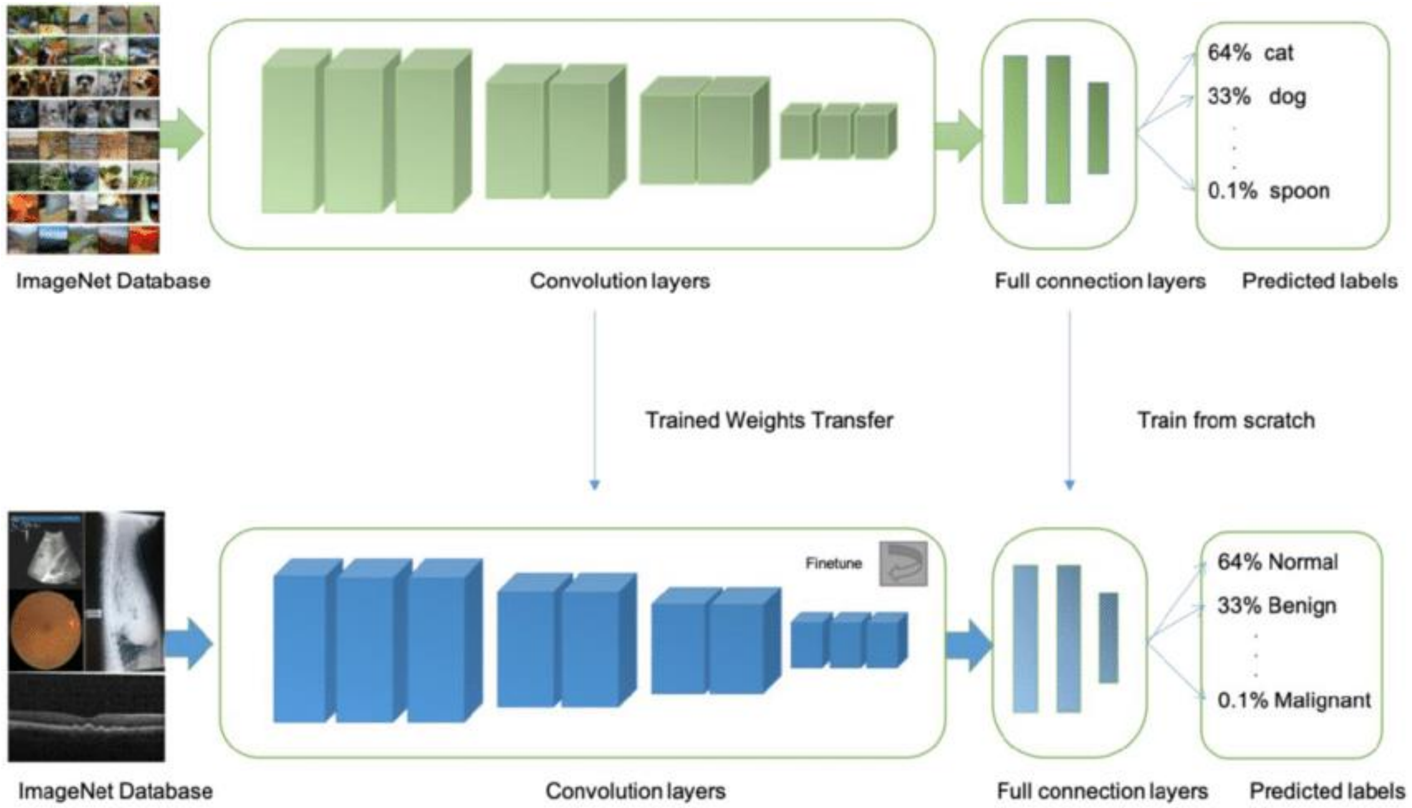
Transfer learning



Transfer learning

Imagenet: 1million

- Assume two datasets, T and S
- Dataset S is
  - fully annotated, plenty of images
  - We can build a model h_S
- Dataset T is
  - Not as much annotated, or much fewer images
  - The annotations of T do not need to overlap with S
- We can use the model h_S to learn a better h_T
- This is called transfer learning

h_S

"My dataset": 1,000

h_T

Why use Transfer Learning?

- A CNN can have millions of parameters
- But our datasets are not always as large
- Could we still train a CNN without overfitting problems?

Convnets are good in transfer learning

- Even if our dataset T is not large, we can train a CNN for it
- Pre-train a network on the dataset S
- Then, there are two solutions
  - Fine-tuning
  - CNN as feature extractor

Solution I: Fine-tune h_T using h_S as initialization

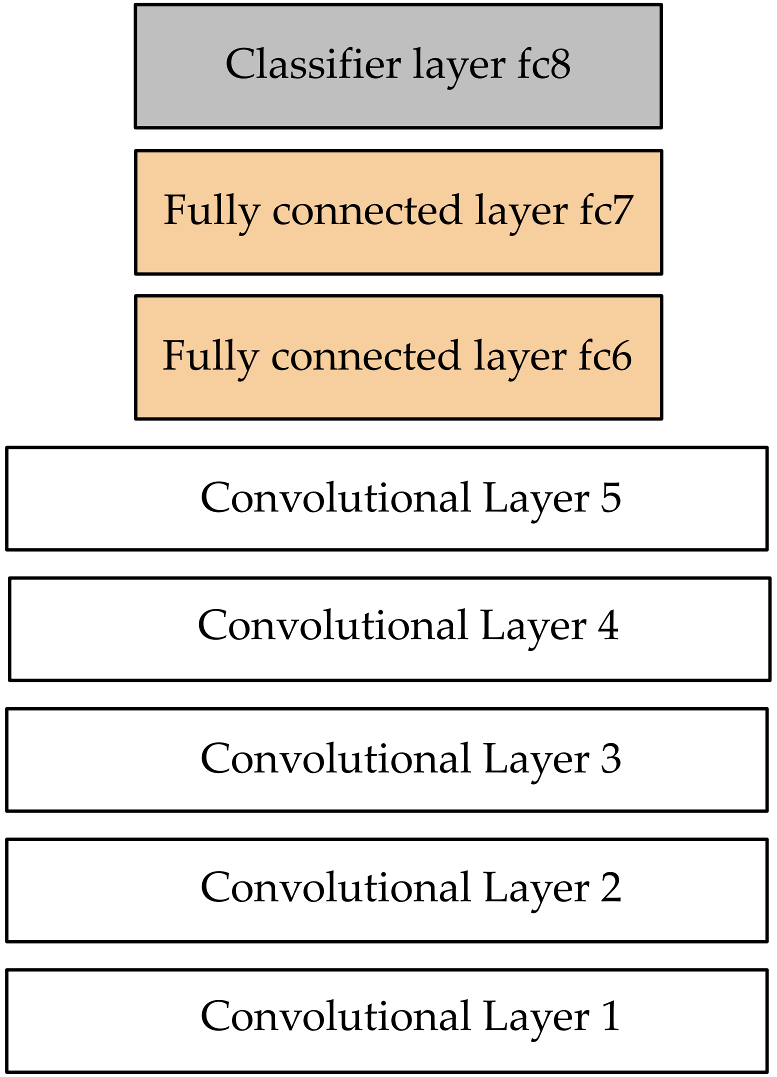
- Assume parameters trained on S are already a good initial solution
- Use them as the initial parameters for our new CNN for the target dataset

$$w_l^S = w_{l,init}^T \text{ for layers } l = 1, 2, \ldots$$

- Better use when your source S is large and target T is small (relatively)
  - E.g. reuse parameters from Imagenet models for smaller datasets
- What layers to initialize and how?

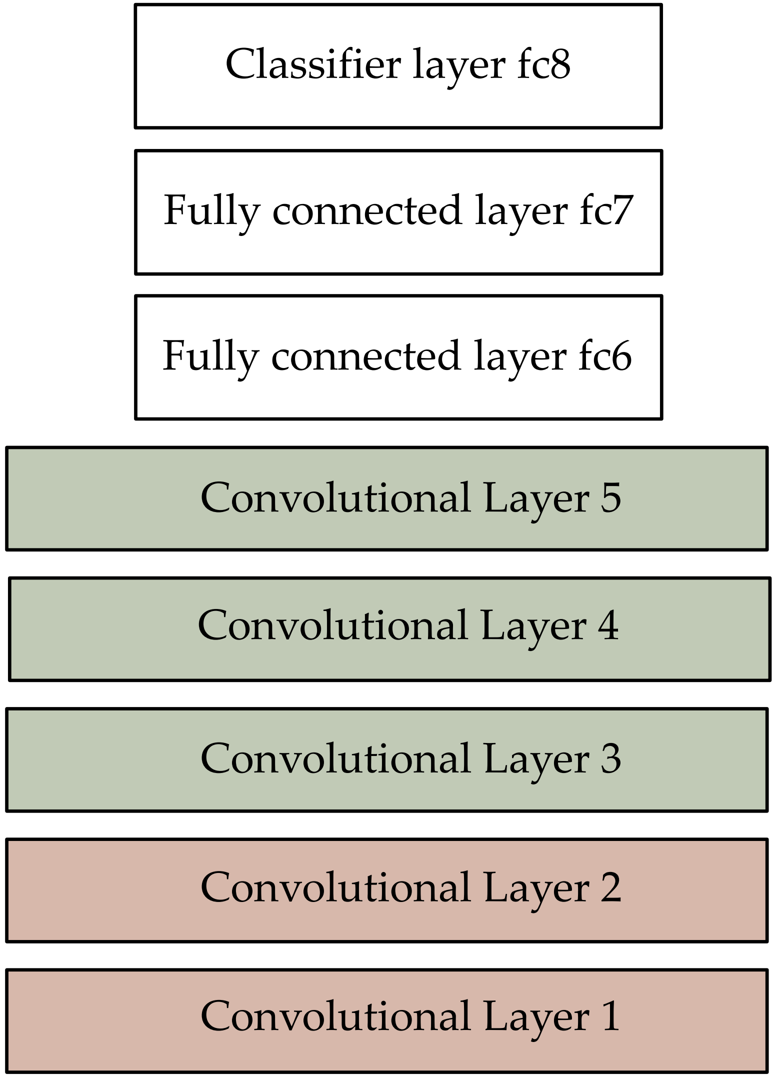
Initializing h_T with h_S

- Classifier layer to loss
  - The loss layer essentially is the “classifier”
  - Same labels → keep the weights from h_S
  - Different labels → delete the layer and start over
  - When too few data, fine-tune only this layer
- Fully connected layers
  - Very important for fine-tuning
  - Maybe delete the last layer before the classification layer if datasets are very different
  - Combine spatial features, more semantics
  - If you have more data, fine-tune these layers first



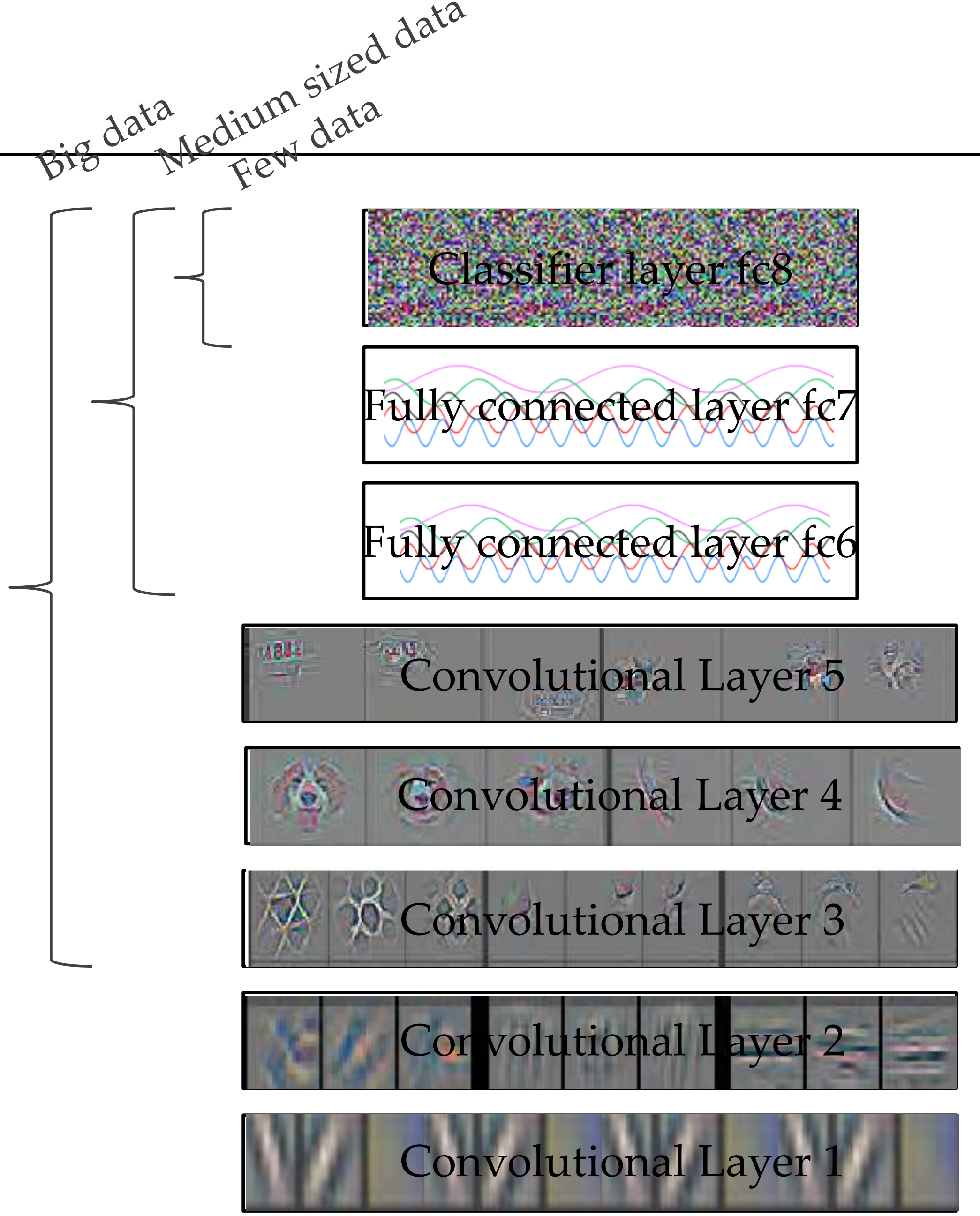
Initializing h_T with h_S

- Upper convolutional layers (conv4, conv5)
  - Mid-level spatial features (face, wheel detectors …)
  - Can be different from dataset to dataset
  - Capture more generic information
  - Fine-tuning pays off
  - Fine-tune if dataset is big enough
- Lower convolutional layers (conv1, conv2)
  - They capture low level information
  - This information does not change usually
  - Probably, no need to fine-tune but no harm trying
  - At this level, maybe no fine-tuning needed



How to fine-tune?

- For layers initialized from h_S use a mild learning rate
  - Your network is already close to a near optimum
  - If too aggressive, learning might diverge
  - A learning rate of 0.001 is a good starting choice (assuming 0.01 was the original learning rate)
- For completely new layers (e.g. loss) use aggressive learning rate
  - If too small, the training will converge very slowly
  - The rest of the network is near a solution, this layer is very far from one
  - A learning rate of 0.01 is a good starting choice
- If datasets are very similar, fine-tune only fully connected layers
- If datasets are different and you have enough data, fine-tune all layers



Solution II: Use h_s as a feature extractor for h_T

- Similar to a case of solution I where you train only the loss layer
  - But can be used with 'external classifiers'
  - Essentially use the network as a pretrained feature extractor
- Use when the target dataset T is very small
  - Any fine-tuning of layer might cause overfitting
  - Or when we don't have the resources to train a deep net
  - Or when we don't care for the best possible accuracy

Which layer?

Table 6. Analysis of the discriminative information contained in each layer of feature maps within our ImageNet-pretrained convnet. We train either a linear SVM or softmax on features from different layers (as indicated in brackets) from the convnet. Higher layers generally produce more discriminative features.

	Cal-101 (30/class)	Cal-256 (60/class)
SVM (1)	44.8 ± 0.7	24.6 ± 0.4
SVM (2)	66.2 ± 0.5	39.6 ± 0.3
SVM (3)	72.3 ± 0.4	46.0 ± 0.3
SVM (4)	76.6 ± 0.4	51.3 ± 0.1
SVM (5)	$\mathbf{86.2 \pm 0.8}$	65.6 ± 0.3
SVM (7)	$\mathbf{85.5 \pm 0.4}$	$\mathbf{71.7 \pm 0.2}$
Softmax (5)	82.9 ± 0.4	65.7 ± 0.5
Softmax (7)	$\mathbf{85.4 \pm 0.4}$	$\mathbf{72.6 \pm 0.1}$

Lower layer features capture more basic information (texture, etc). Good for image-to-image comparisons, image retrieval ← (SVM (2))

Higher layer features capture more semantic information. Good for higher level classification → (SVM (7))

Visualizing and Understanding Convolutional Networks, Zeiler and Fergus, ECCV 2014

Summary

- Shared filters through local connectivity
- Convolutions
- Convolutional Neural Networks
- Alexnet case study
- Visualizing ConvNets
- Transfer learning

Reading material

- Chapter 9