Artificial Neural Networks and Deep Learning

Week 6

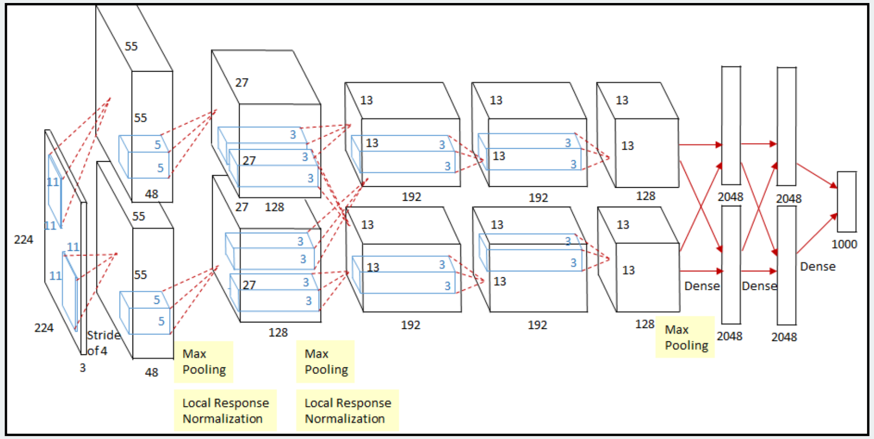
Transfer learning

Transfer learning

Reusing a model trained on one problem, on another problem

The **problem** with training **big** deep learning models

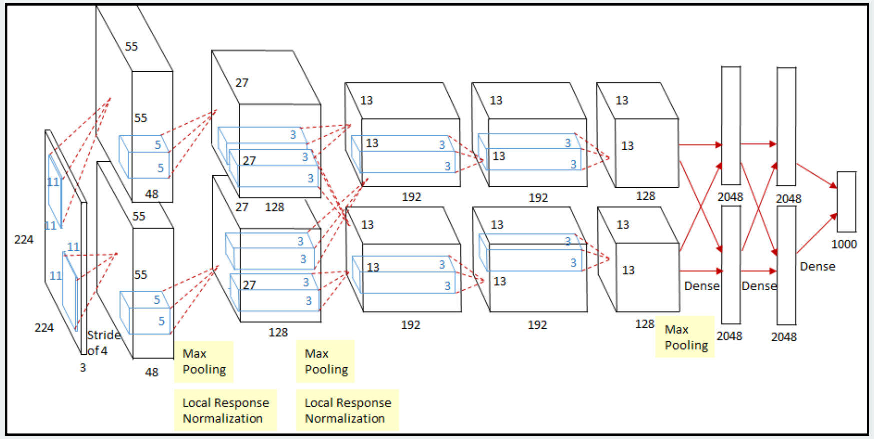
- Extremely long training times (up to weeks)
- Expensive cloud computing fees, or GPU cost and electricity bills
- Huge CO_2 footprint (as much as 5 cars)



The **problem** with training **big** deep learning models

> **Solution:** *Reuse pre-trained models!*

- Extremely long training times (up to weeks)
- Expensive cloud computing fees, or GPU cost and electricity bills
- Huge CO_2 footprint (as much as 5 cars)

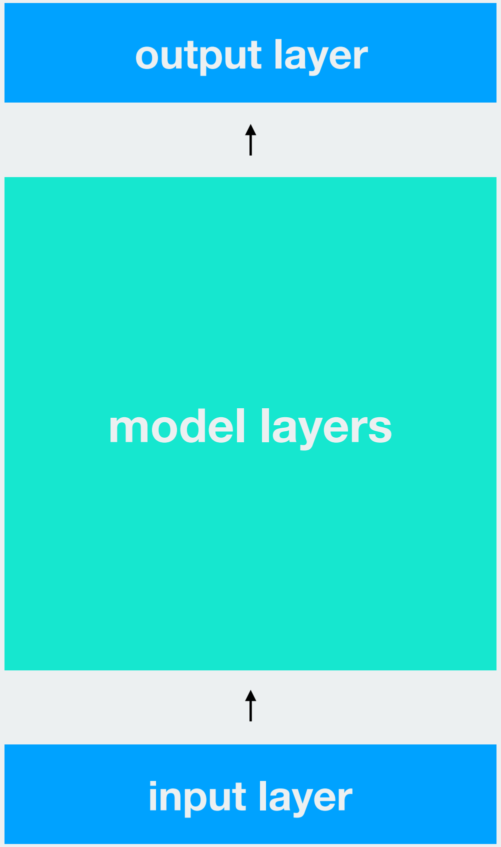


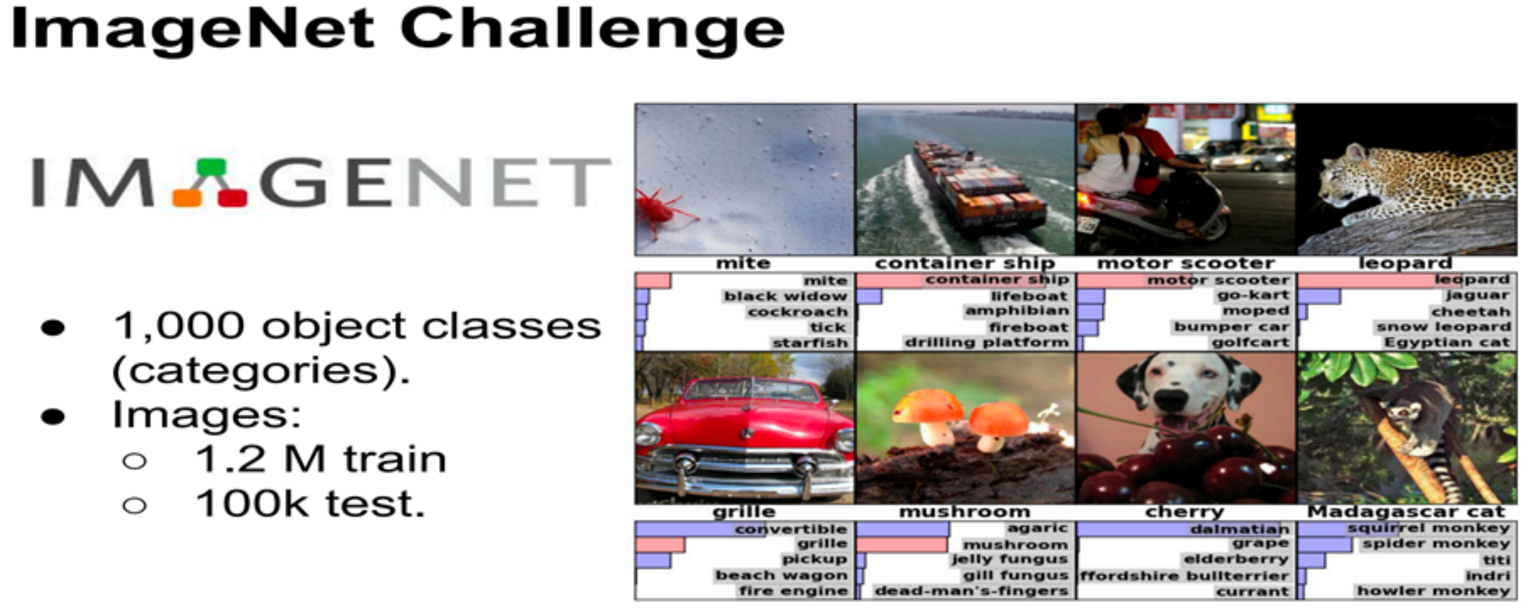
Transfer learning

> Fundamental idea

1. Train on one (huge) dataset

2. Reuse model to improve training on another dataset





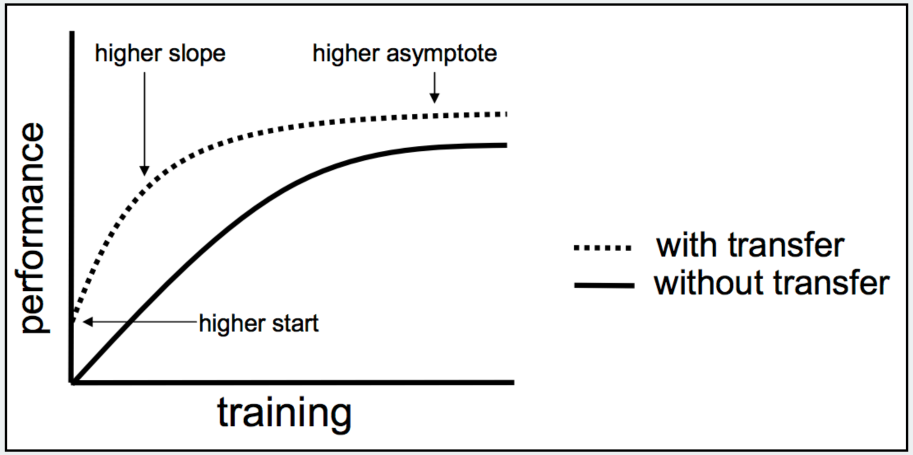
ImageNet Challenge

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

Transfer learning

> Benefits

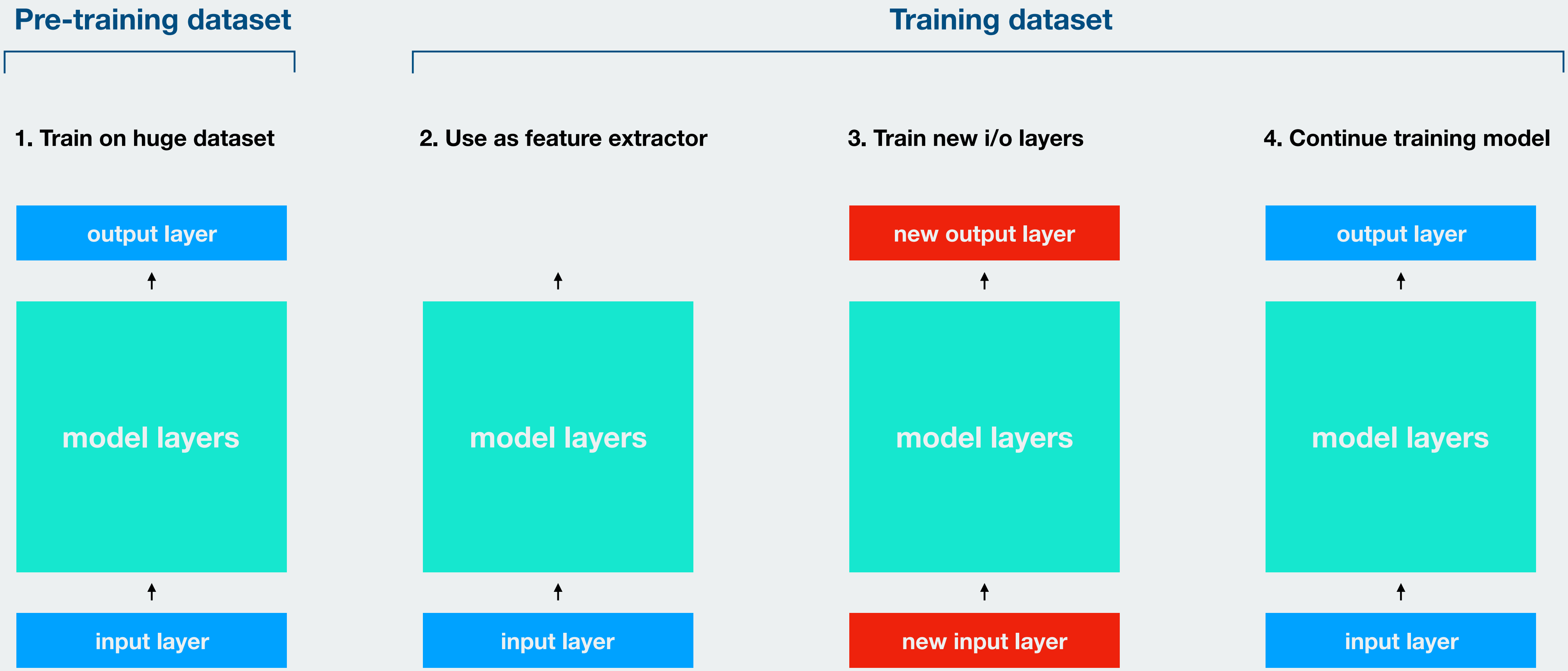
- Makes training on new data much faster
- Enables training on small datasets
- Helps avoid overfitting. Initial weights are usually better than random, helping avoid many local minima.

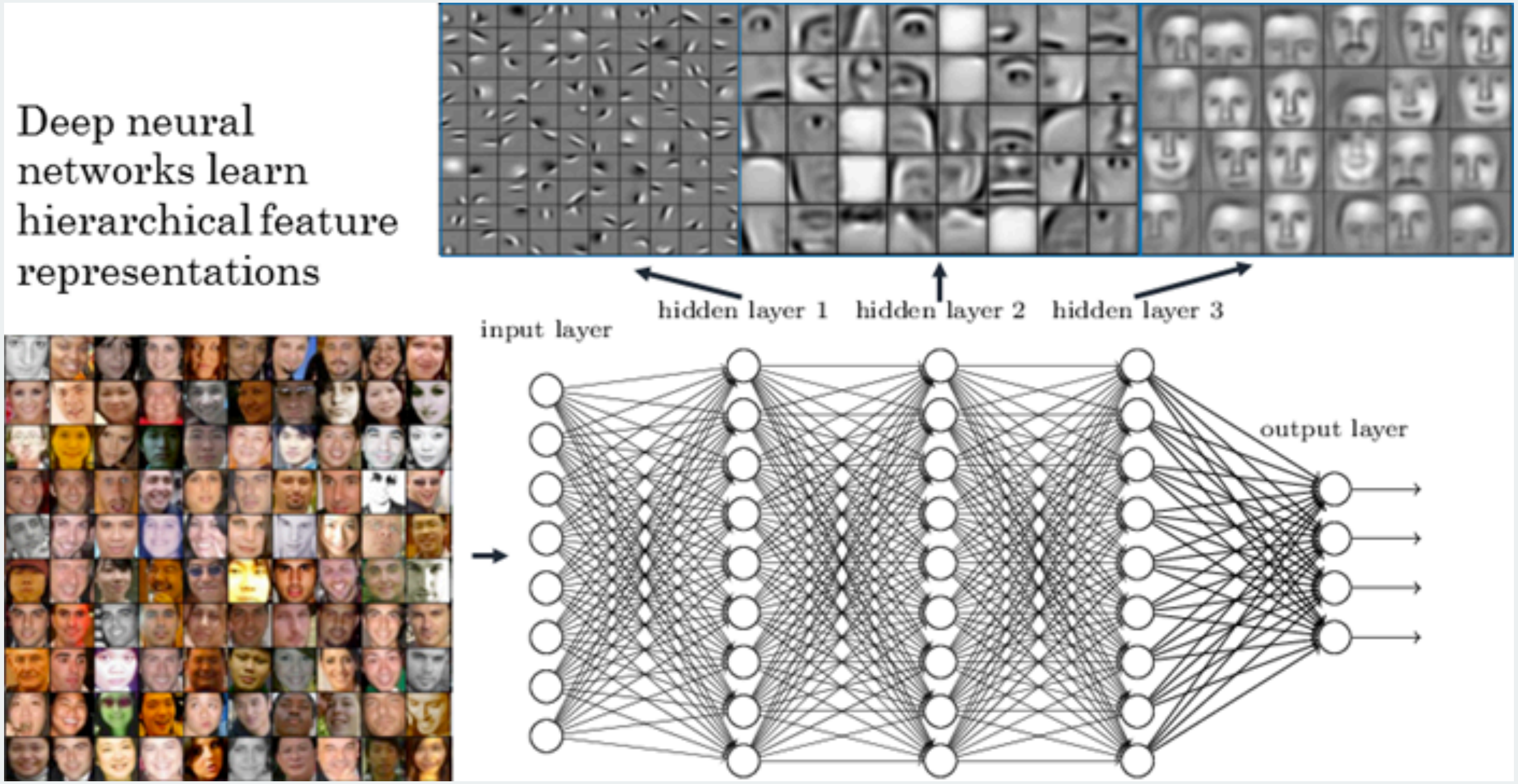


https://machinelearningmastery.com/transfer-learning-for-deep-learning/

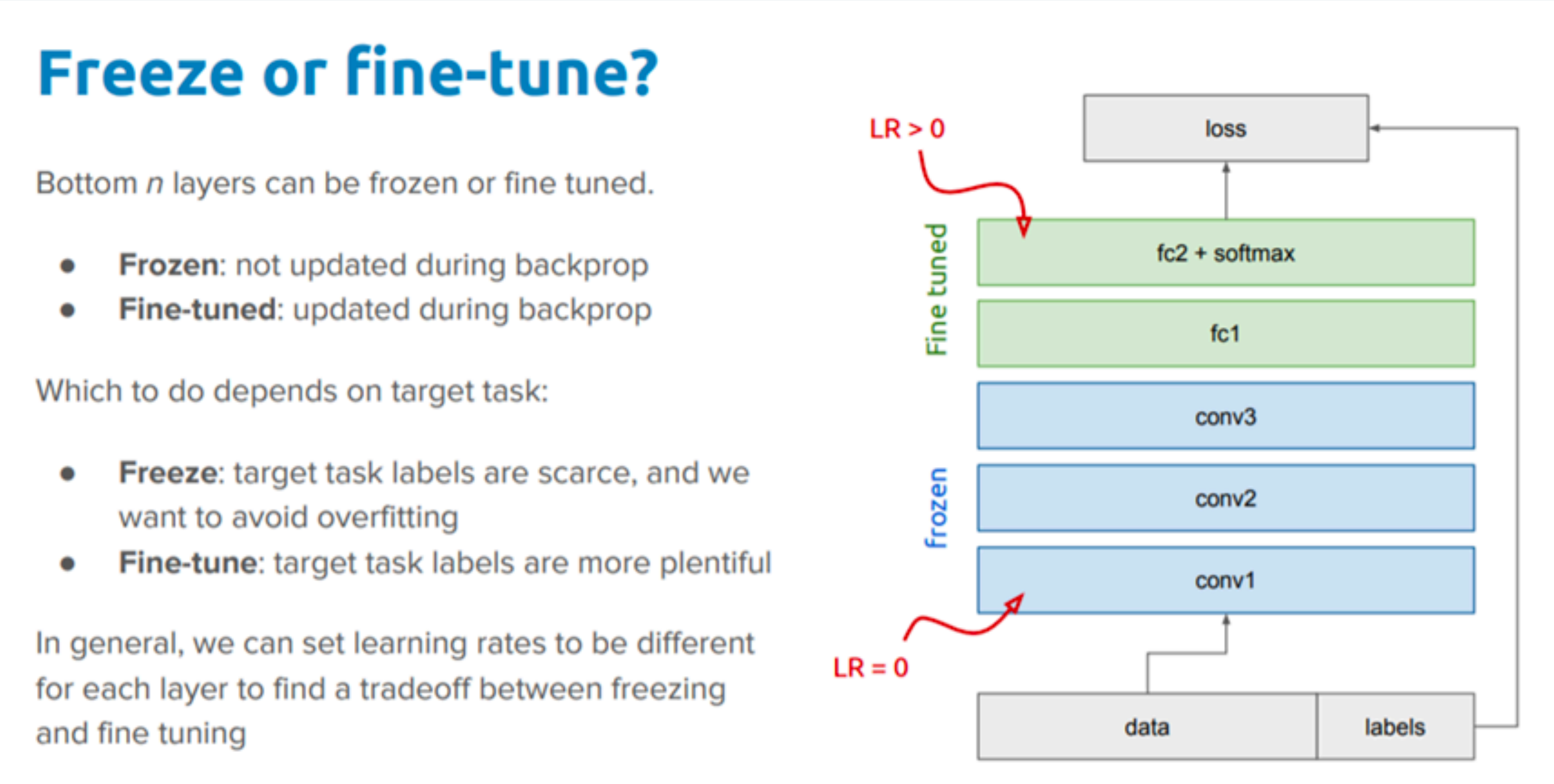
Transfer learning

> Fundamental idea (nuanced)





Deep neural networks learn hierarchical feature representations



Freeze or fine-tune?

Bottom *n* layers can be frozen or fine tuned.

- **Frozen**: not updated during backprop
- **Fine-tuned**: updated during backprop

Which to do depends on target task:

- **Freeze**: target task labels are scarce, and we want to avoid overfitting
- **Fine-tune**: target task labels are more plentiful

In general, we can set learning rates to be different for each layer to find a tradeoff between freezing and fine tuning

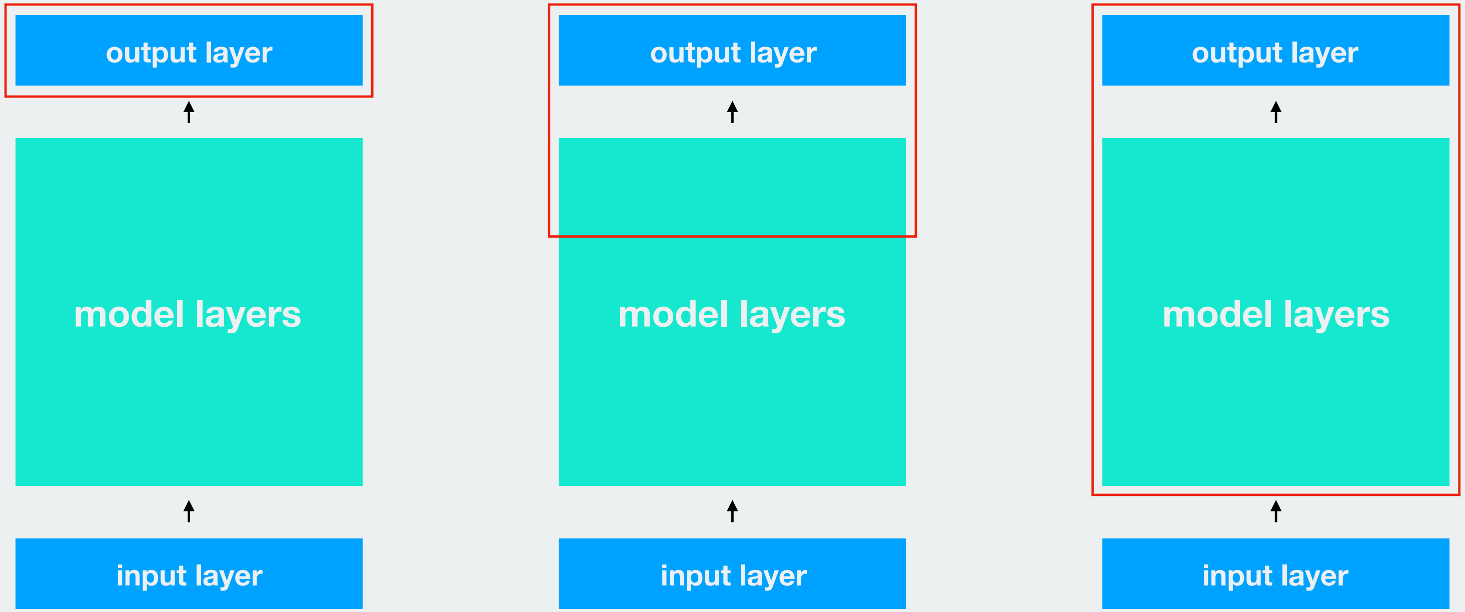
Transfer learning

> Strategies

1. If training dataset is **small** only train **last layers**

2. If training dataset is **big** train more **last layers**

3. If training dataset is **huge** train all **layers** with reduced learning rate. 1/10th of orig. LR is good choice



Transfer learning

> Strategies

	Similar dataset	Different dataset
Little data	Use pre-trained model as feature extractor and do classification with new features and simpler model	Difficult. Maybe consider using a different pre-trained model or use different feature extractors
Much data	Finetune a few layers towards the end of the network, with lowered LR	Finetune a large number of layers, with lowered LR

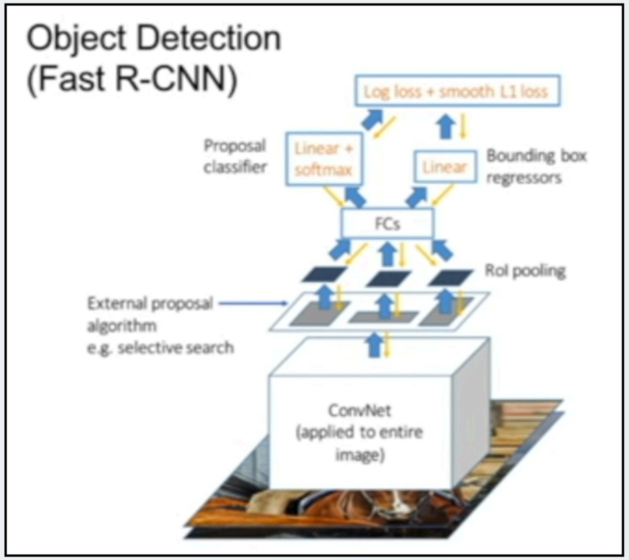
Transfer learning

> Further

- Transfer learning is extremely pervasive, especially for image data

- Also used for language modeling. There exists publicly available *word embeddings* which encode words as vectors in an efficient way (Word2Vec).

- Most research and industrial projects start with some pretrained model and then build something on top of that.



Object Detection (Fast R-CNN)

Log loss + smooth L1 loss
Proposal classifier
Linear + softmax
Linear
Bounding box regressors
FCs
RoI pooling
External proposal algorithm e.g. selective search
ConvNet (applied to entire image)

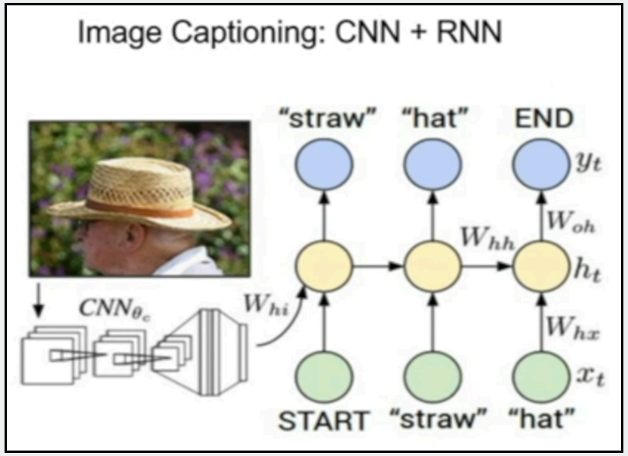
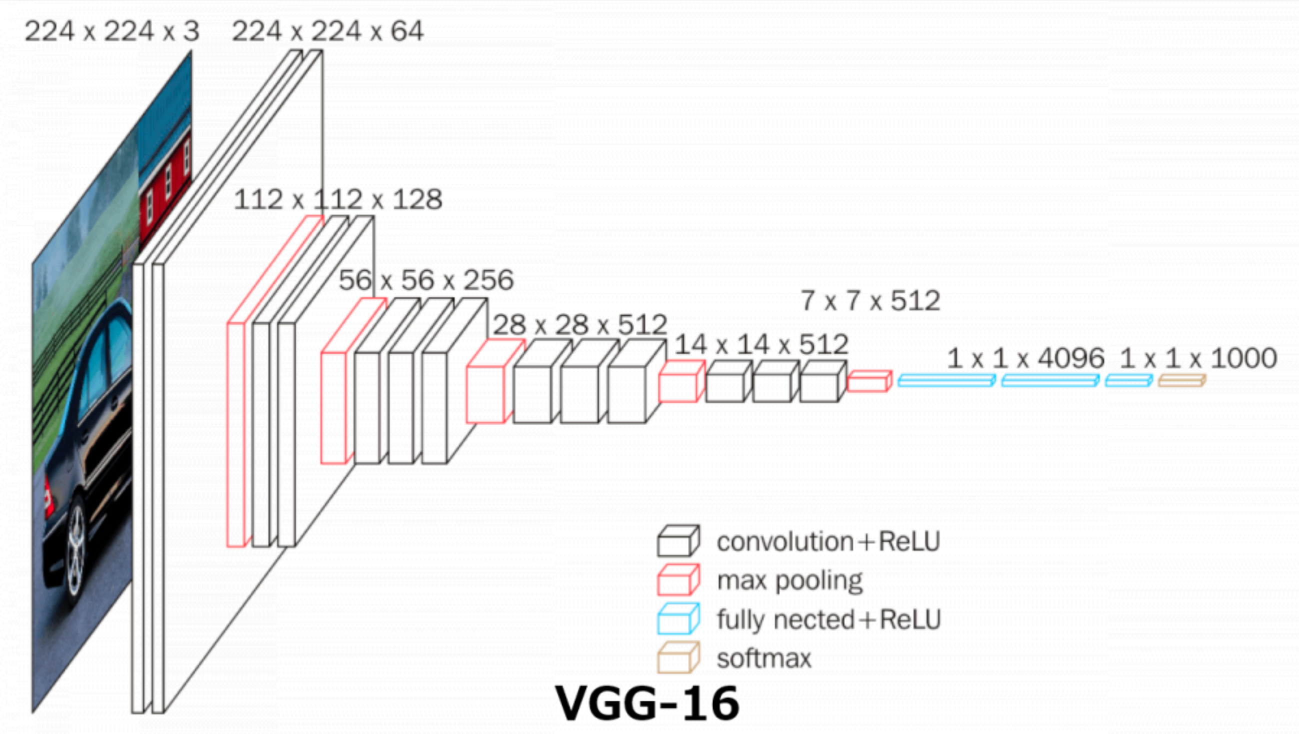
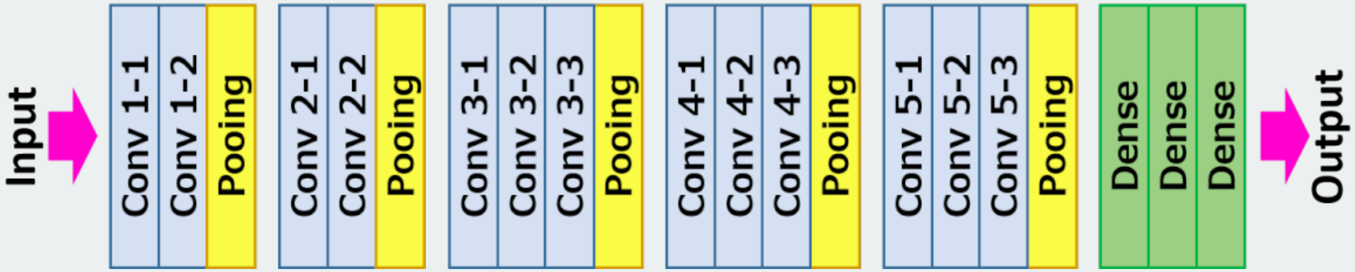


Image Captioning: CNN + RNN

Popular Architectures

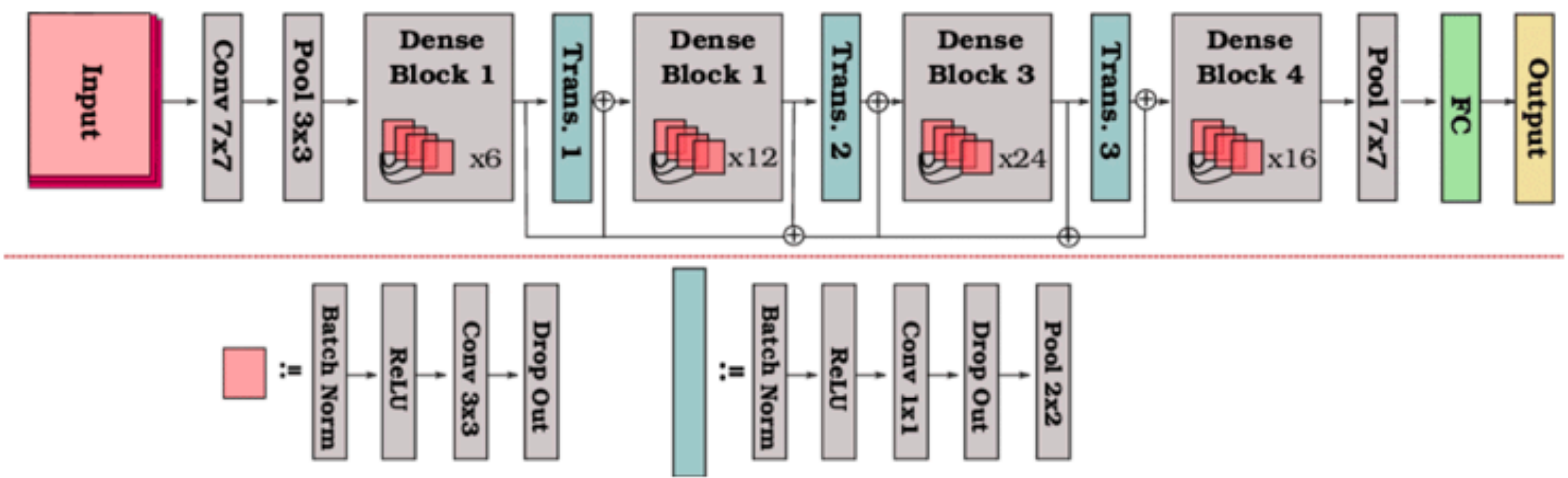
VGG16



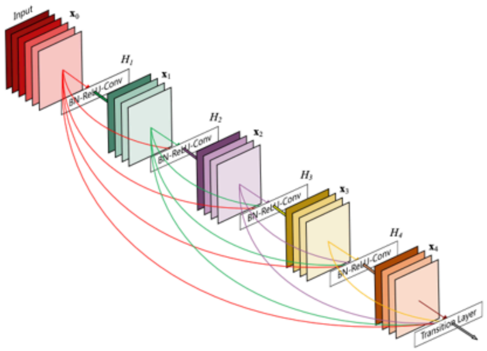


Popular Architectures

DenseNet121



G. Huang



DenseNet Structure

$$a^{[l]} = g\big([a^{[0]}, a^{[1]}, a^{[2]}, \ldots\ldots\ldots\ldots, a^{[l-1]}]\big)$$