Microsoft

Transfer Learning

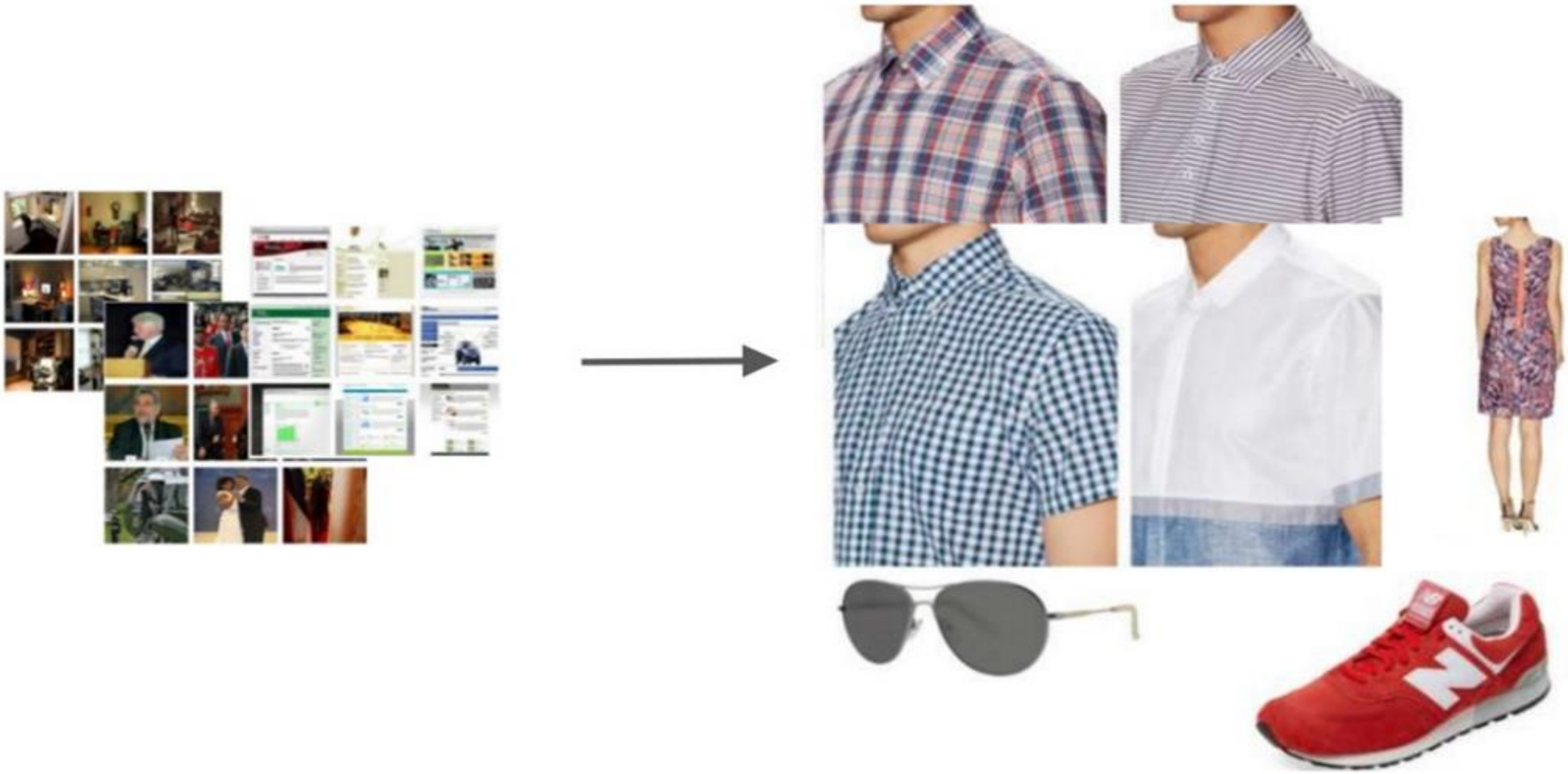
Mithun Prasad, PhD
miprasad@Microsoft.com

Transfer Learning

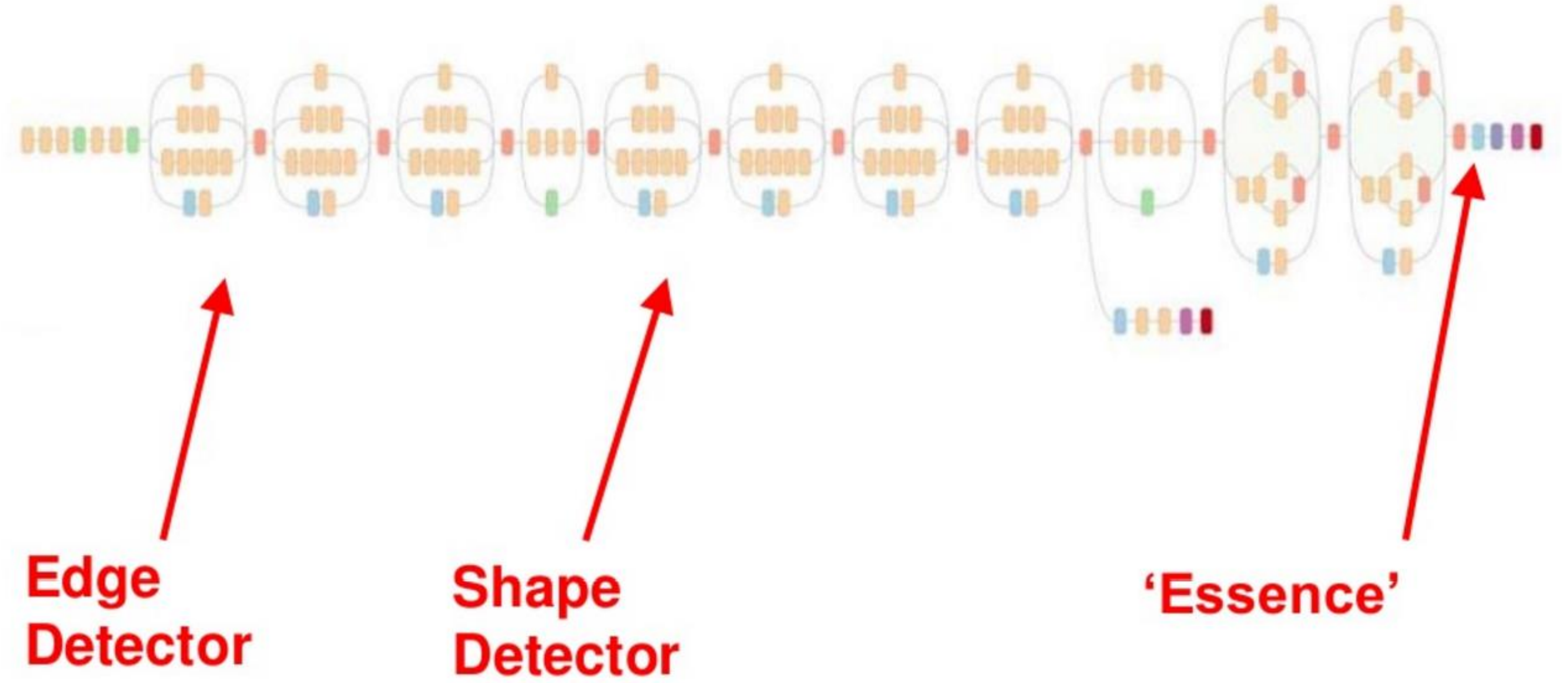
Why use a pre-trained model?

- It’s faster (it’s pre-trained)
- It’s cheaper (no need for GPU farm)
- It generalizes (avoid overfitting)

Transfer Learning



Structure of the Network



Edge Detector

Shape Detector

‘Essence’

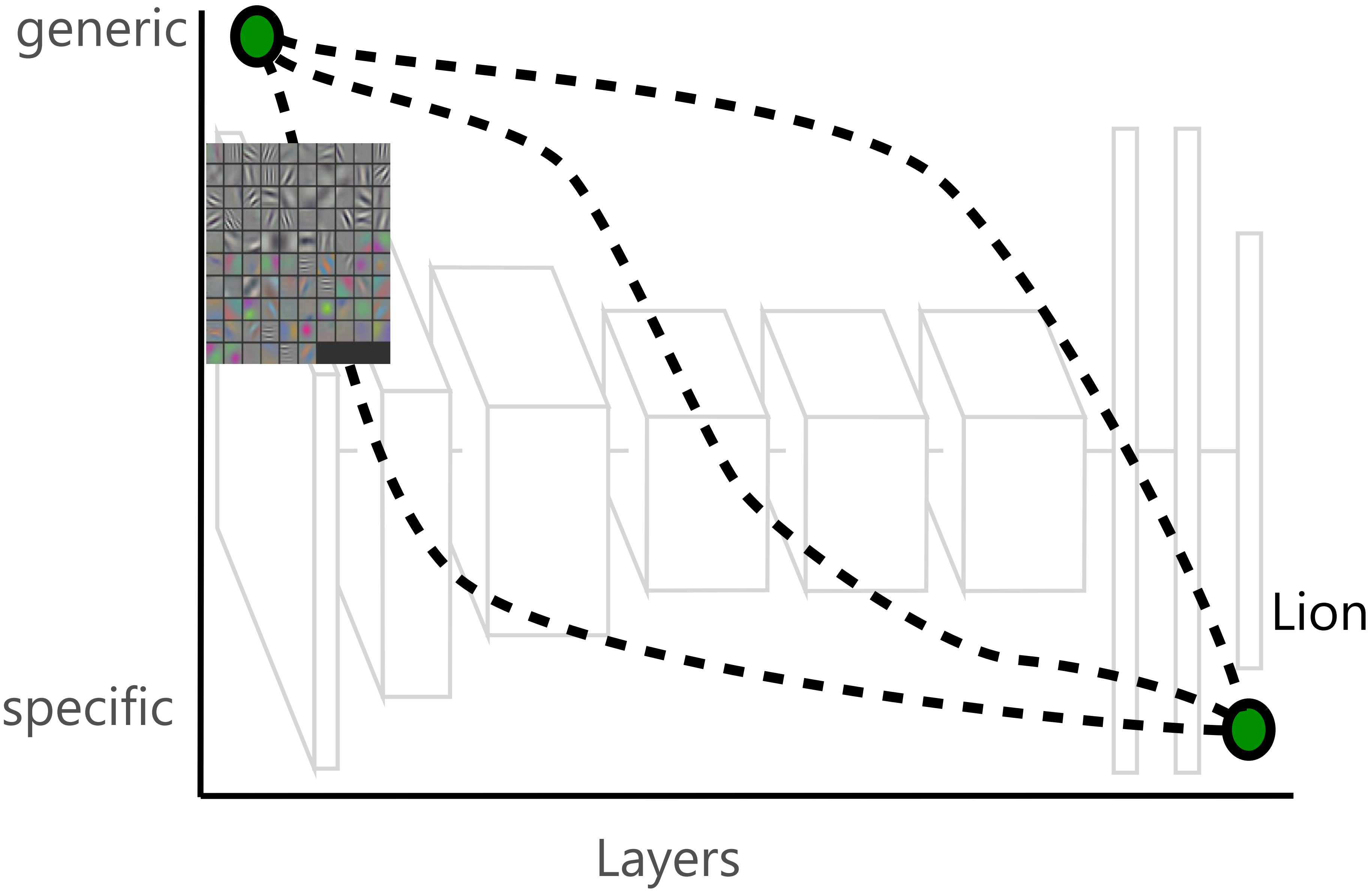
Microsoft

Transfer Learning Strategies

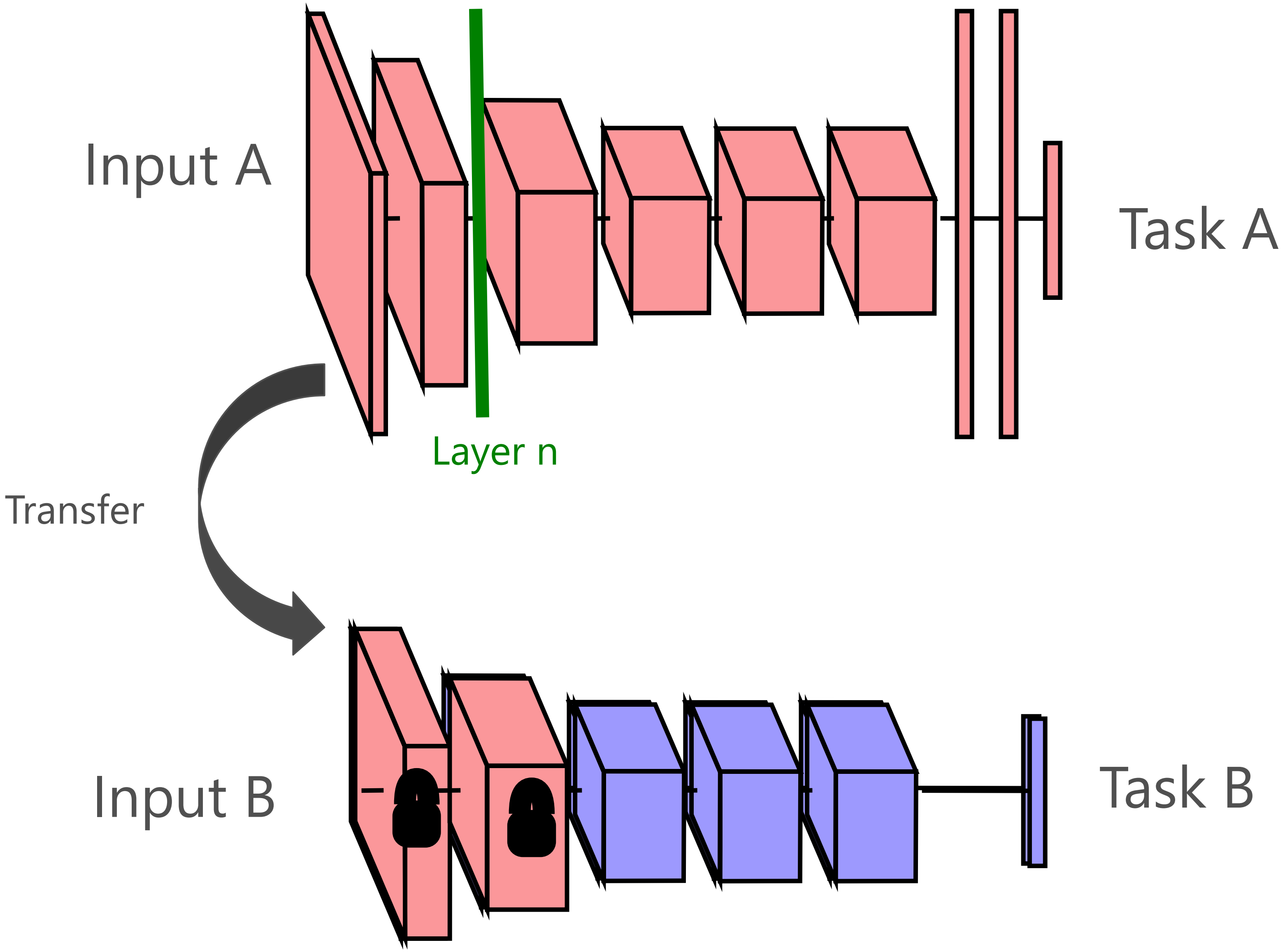
Transfer Learning Strategies

- ConvNet as fixed feature extractor
  - Example: Use a car classifier to further classify the car brand
  - Very fast
- Fine-tuning the ConvNet
  - Fine-tune weights weights of pre-trained CNN
  - Fine-tune all layers / Keep early layers fixed and fine-tune remaining layers
  - Slow

Generic to Specific



Fine-tuning



When To Use Transfer Learning?

- Dependence on size of the new dataset and similarity to the original dataset
- Features are more generic in early layers and more dataset-specific in later layers
- 4 Scenarios

	Relatively Smaller	**Relatively Large**
Similar	- Similar higher-level features - Train classifier on CNN features	- Fine-tune
Different	- Train classifier from activations somewhere early in the network	- Train from scratch

Constraints and Learning Rates

- Assume that the weights from pre-trained CNN are robust
- To avoid distortion too quickly or too much, keep the learning rate and learning rate decay small