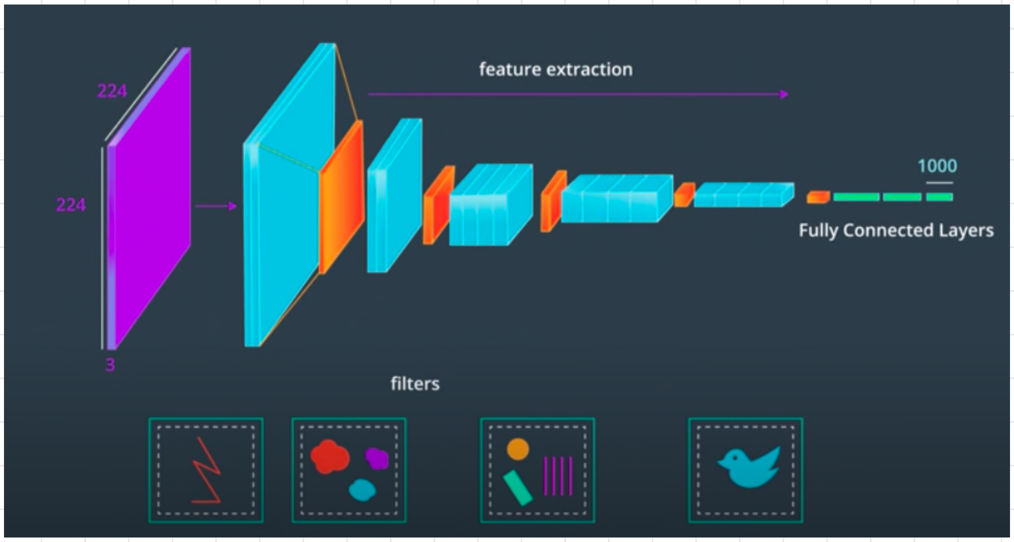
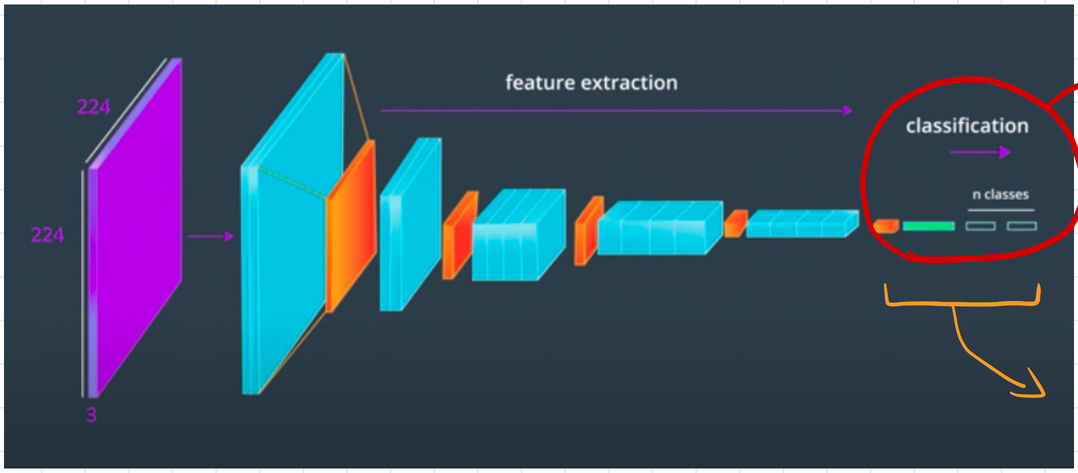
Transfer Learning

Transfer Learning Approach #1 - Useful layers



When applying this network we disgard the final layers which are specific to the original data (of the developers)



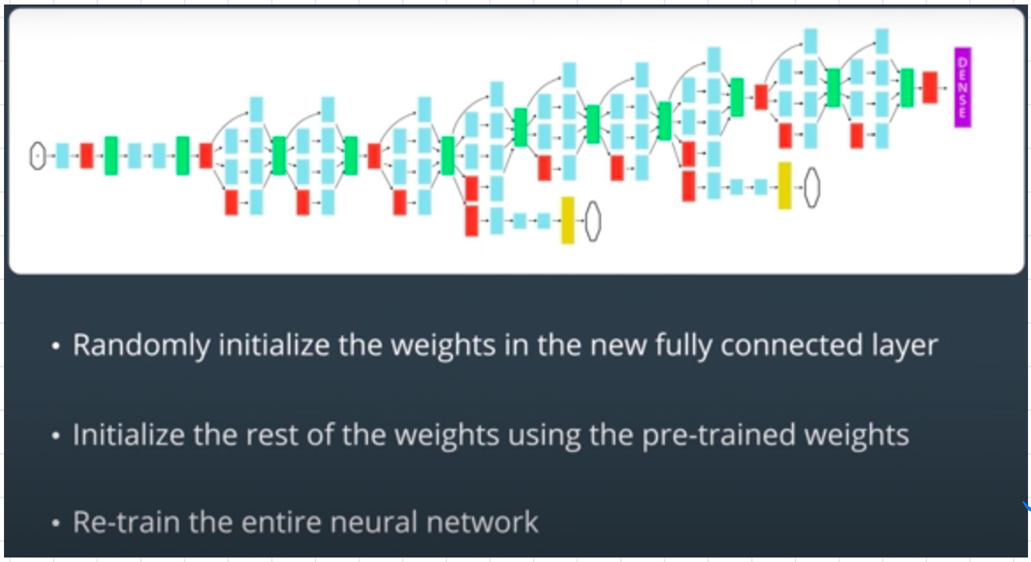
Adding our own linear fc layers as a final classifier.

Train only these final layers for a new task

Keeping the previous layers (trained feature extractors)

However,
method depends on size of dataset & level of similarity w/ the ImageNet dataset.

Approach #2 (4 cases)



- Randomly initialize the weights in the new fully connected layer
- Initialize the rest of the weights using the pre-trained weights
- Re-train the entire neural network

Fine Tuning Example

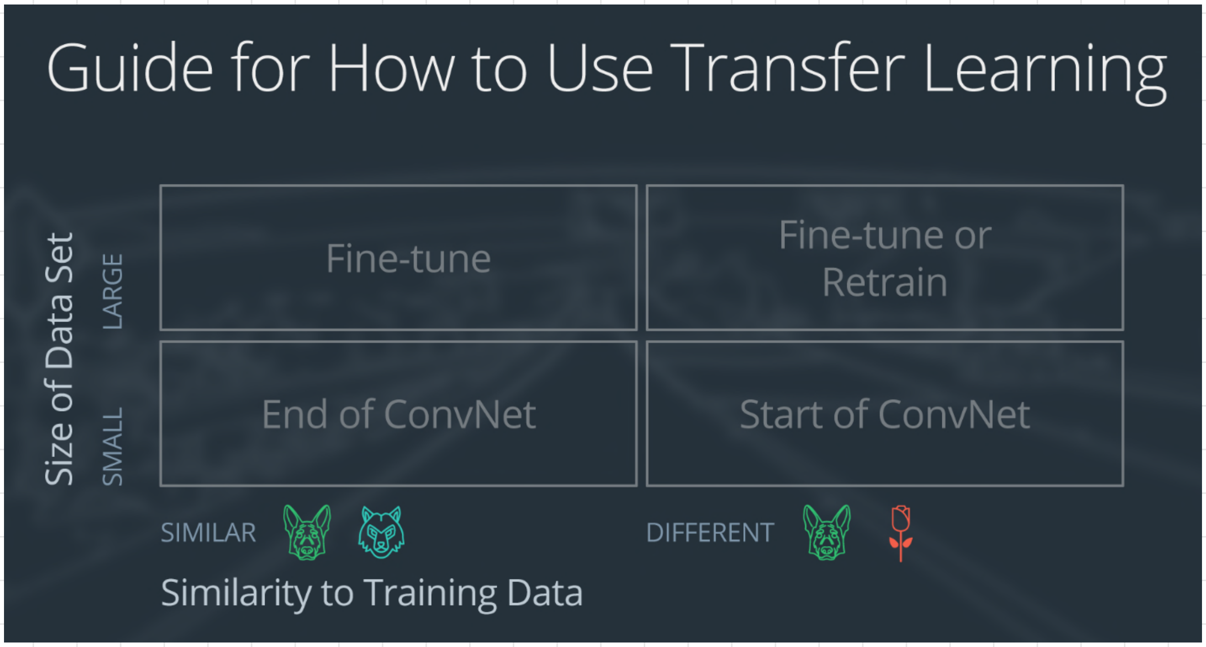
Depending on both:

- The size of the new data set, and
- The similarity of the new data set to the original data set

The approach for using transfer learning will be different. There are four main cases:

1. New data set is small, new data is similar to original training data.
2. New data set is small, new data is different from original training data.
3. New data set is large, new data is similar to original training data.
4. New data set is large, new data is different from original training data.

Prone to overfitting w/ transfer learning



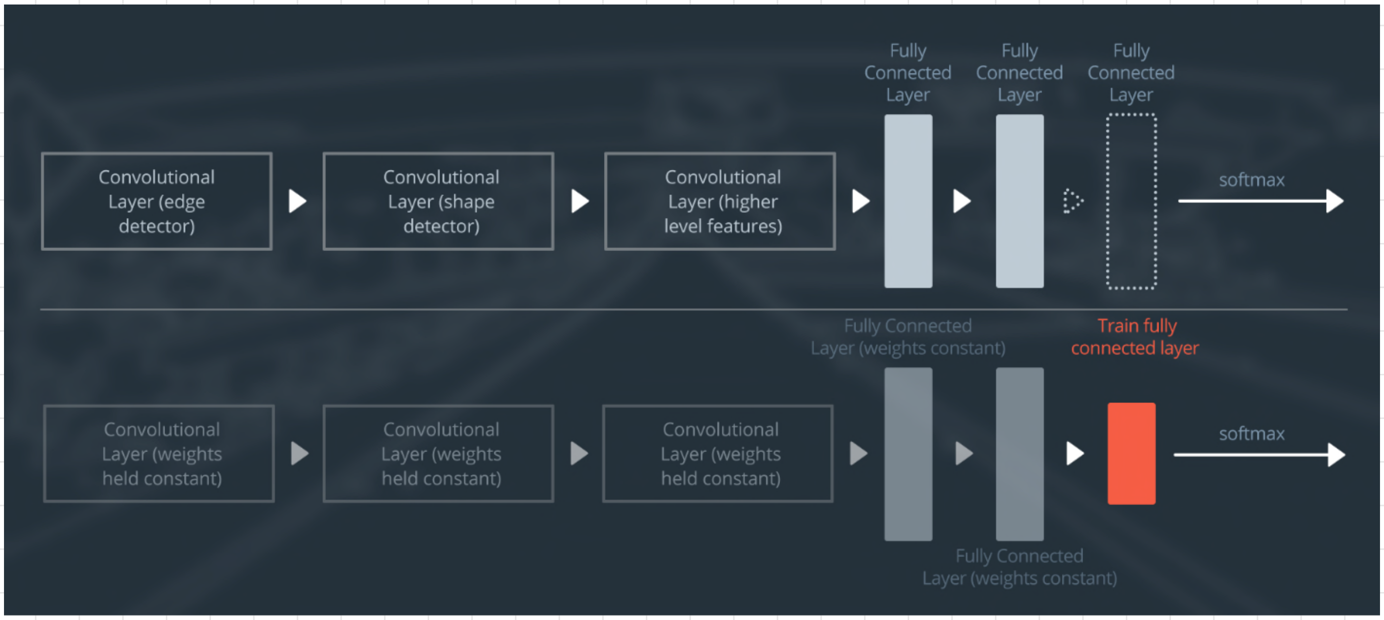
Guide for How to Use Transfer Learning

Size of Data Set	SIMILAR	DIFFERENT
LARGE	Fine-tune	Fine-tune or Retrain
SMALL	End of ConvNet	Start of ConvNet

Similarity to Training Data

Case 1: Small dataset, similar data.

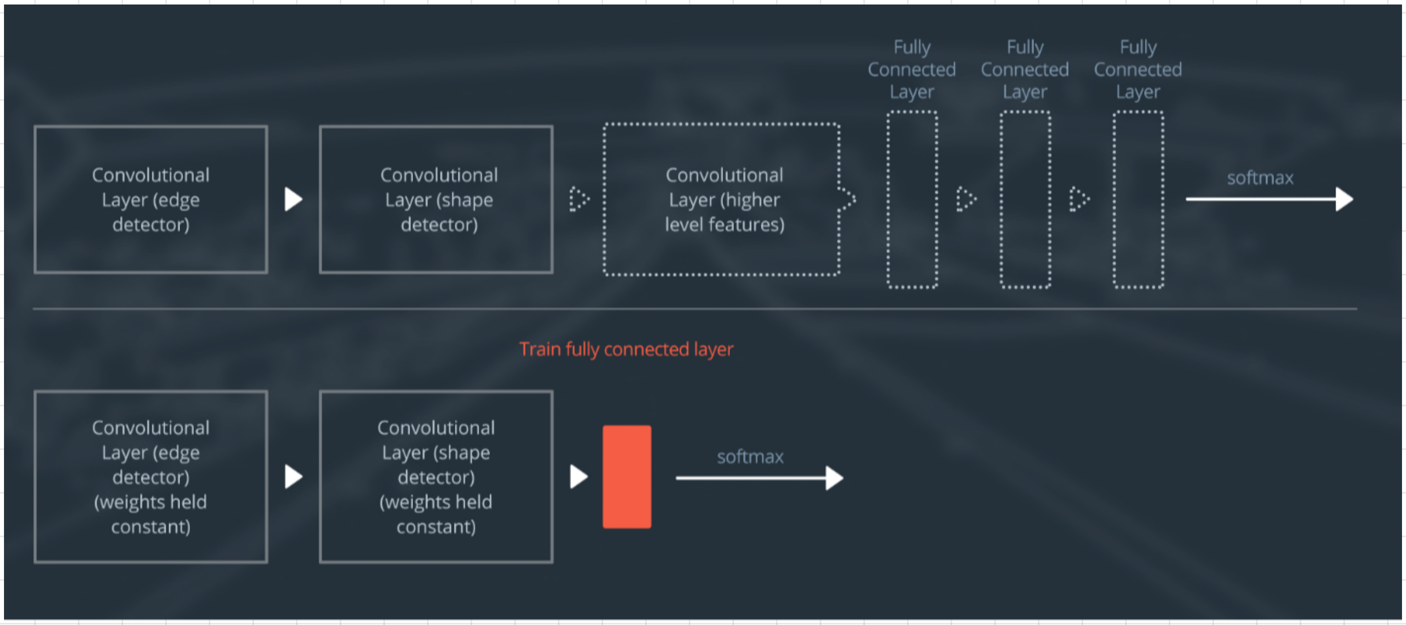
- slice off the end of the neural network
- add a new fully connected layer that matches the number of classes in the new data set
- randomize the weights of the new fully connected layer; freeze all the weights from the pre-trained network → Avoid overfitting
- train the network to update the weights of the new fully connected layer


Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
softmax
Fully Connected Layer (weights constant)
Train fully connected layer
Convolutional Layer (weights held constant)
Convolutional Layer (weights held constant)
Convolutional Layer (weights held constant)
softmax
Fully Connected Layer (weights constant)

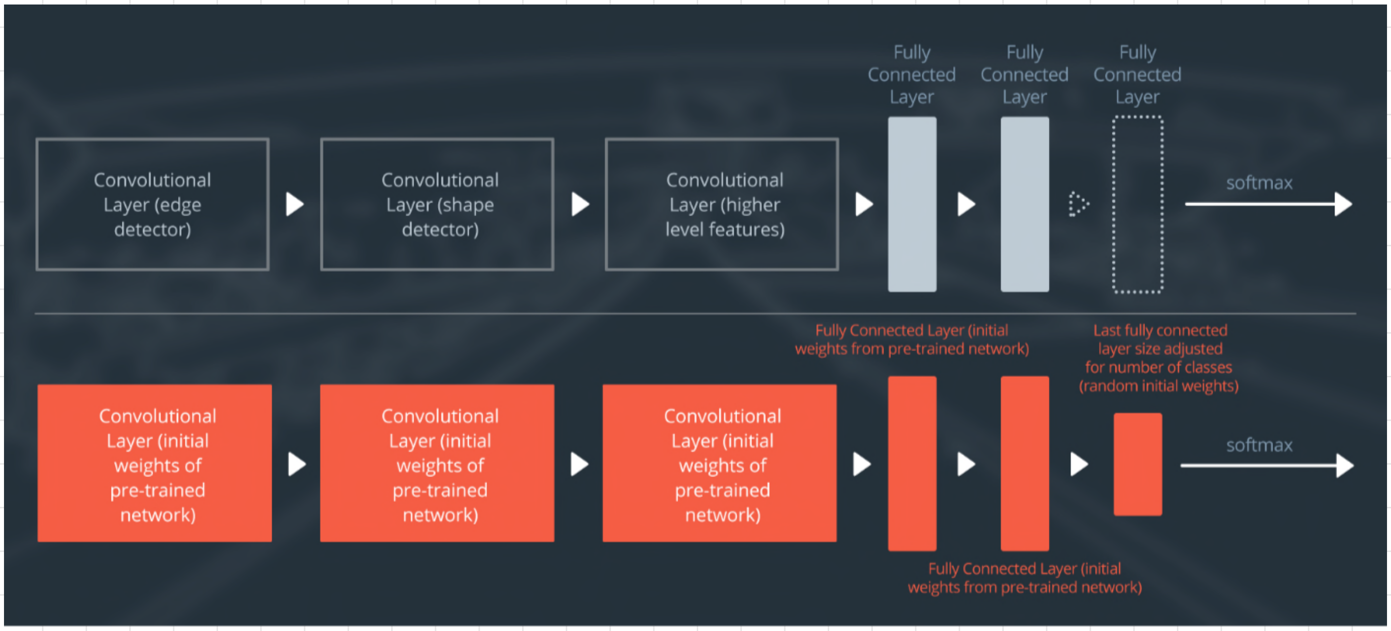
Case 2: Small data, different data

- slice off all but some of the pre-trained layers near the beginning of the network
- add to the remaining pre-trained layers a new fully connected layer that matches the number of classes in the new data set
- randomize the weights of the new fully connected layer; freeze all the weights from the pre-trained network
- train the network to update the weights of the new fully connected layer

constant weights
×
overfitting
small data


Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
softmax
Train fully connected layer
Convolutional Layer (edge detector) (weights held constant)
Convolutional Layer (shape detector) (weights held constant)
softmax

Case 3 : Large data , similar data



Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
softmax
Fully Connected Layer (initial weights from pre-trained network)
Last fully connected layer size adjusted for number of classes (random initial weights)
Convolutional Layer (initial weights of pre-trained network)
Convolutional Layer (initial weights of pre-trained network)
Convolutional Layer (initial weights of pre-trained network)
softmax
Fully Connected Layer (initial weights from pre-trained network)

- Retrain the entire network.
- Retrain all weights

$\underbrace{\hspace{10em}}$ This is called Fine Tuning

Case 4 : Large data, different data

- Fine tune (initialize weights from pre-trained network $\longrightarrow$ makes training faster)
- Or randomly initialize all weights, train from scratch

Fine tune OR Retrain