Transfer Learning

CS109B Data Science 2

Pavlos Protopapas, Mark Glickman, and Chris Tanner


TRANSFER LEARNING
WHAT SOCIETY THINKS I DO
WHAT MY FRIENDS THINK I DO
WHAT INVESTORS THINK I DO
WHAT MY MOM THINKS I DO
I'm going to learn jujitsu?
WHAT I THOUGHT I'LL DO
Transfer Learning using
Mobilenet and Keras
WHAT I ACTUALLY DO
imgflip.com

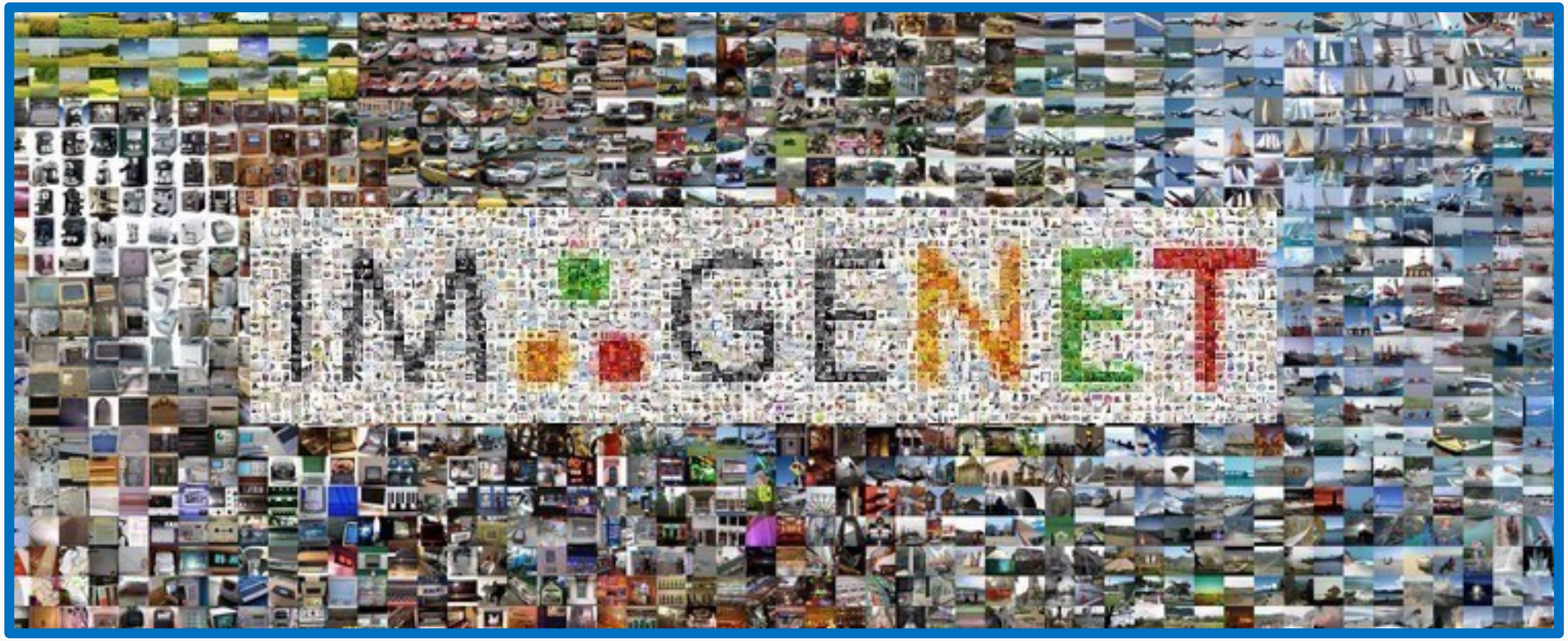
Outline

1. Review/Questions
2. Motivation
3. The Basics idea for Transfer Learning
4. Representation Learning
5. Transfer Learning Strategies
6. Transfer Learning for Deep Learning

CNNs: Story so far

IMAGENET challenge:

- A large visual database designed for use in visual object recognition software research
- More than 14 million images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided

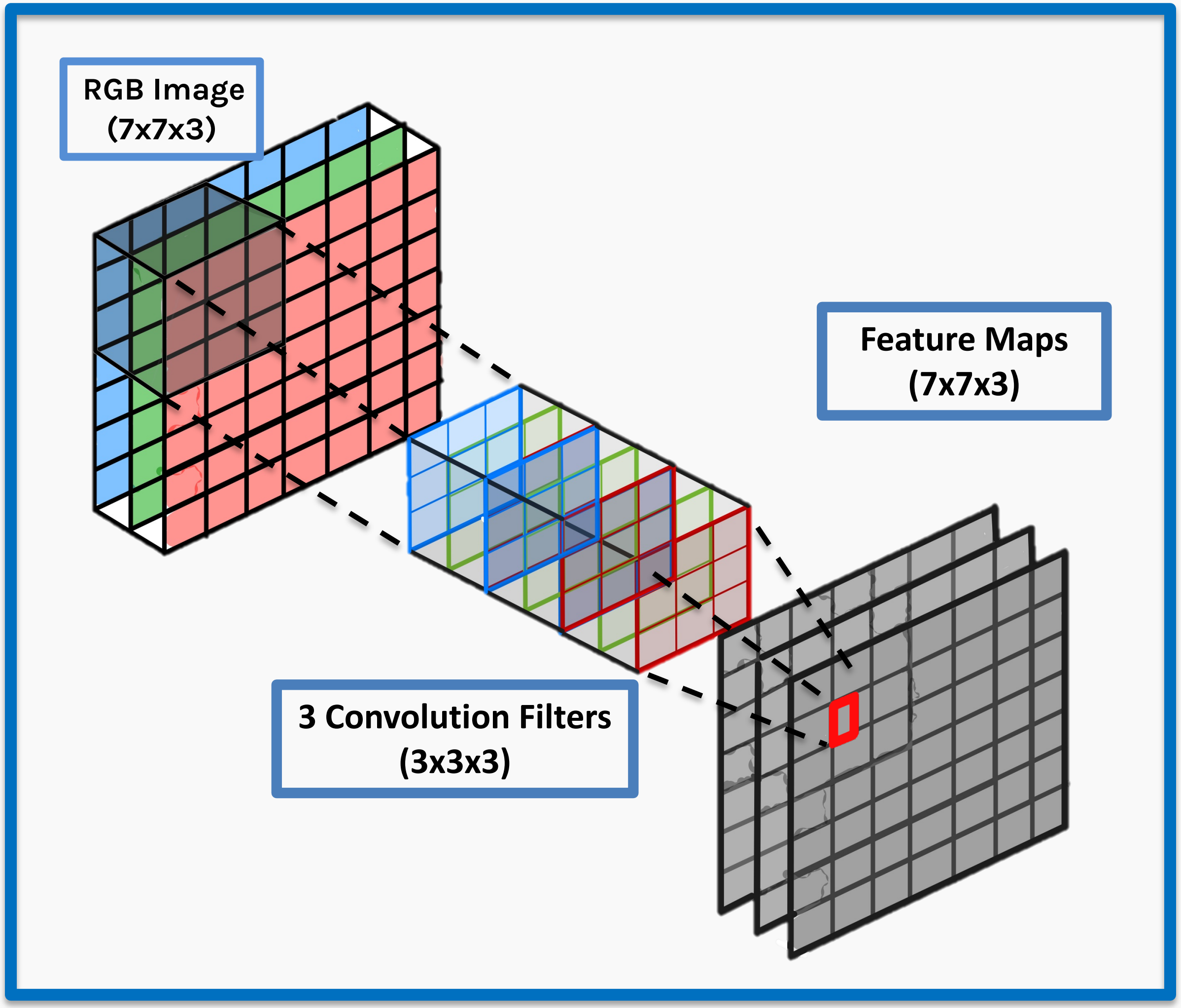


CNNs: "Convolution" Operation

A convolutional neural network typically consists of feature extracting layers and condensing layers.

The feature extracting layers are called convolutional layers & each node in these layers uses a small fixed set of weights to transform the image in the way below.

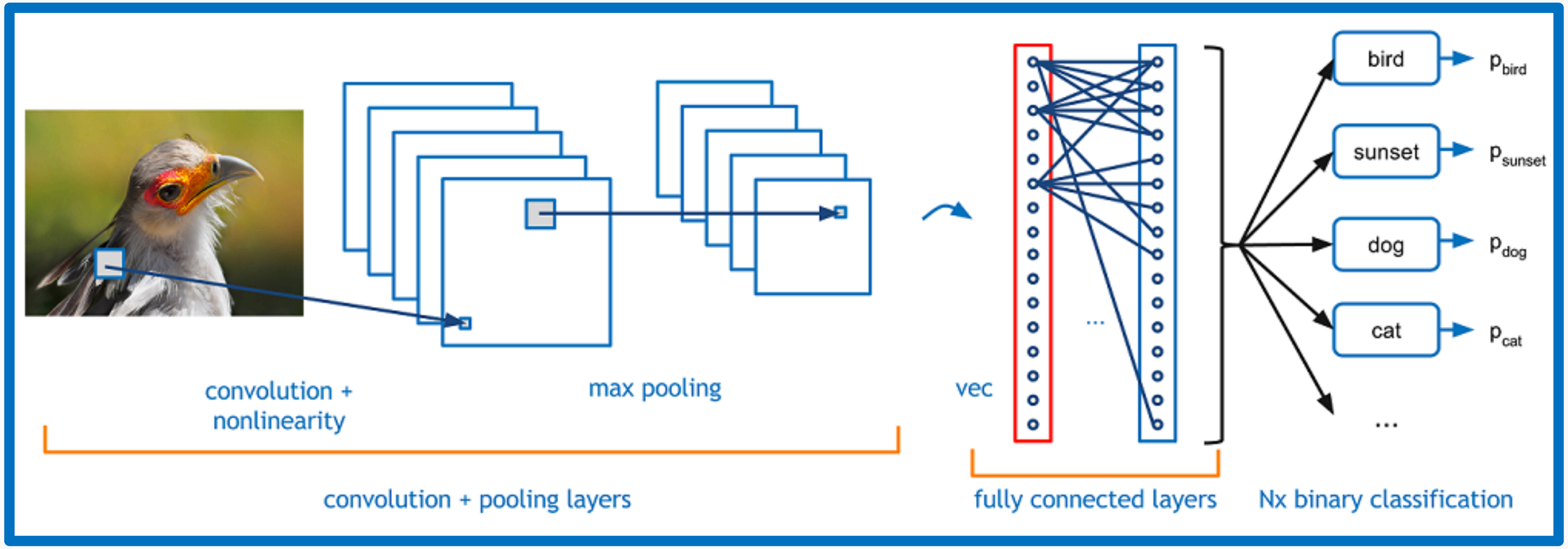
This set of fixed weights for each node in the convolutional layer is often called a kernel.



CNNs: Feature Extraction

Rather than processing image data with a pre-determined set of filters, we want to learn the filters of a CNN for feature extraction. Our goal is to extract features that best helps us to perform our downstream task (e.g. classification).

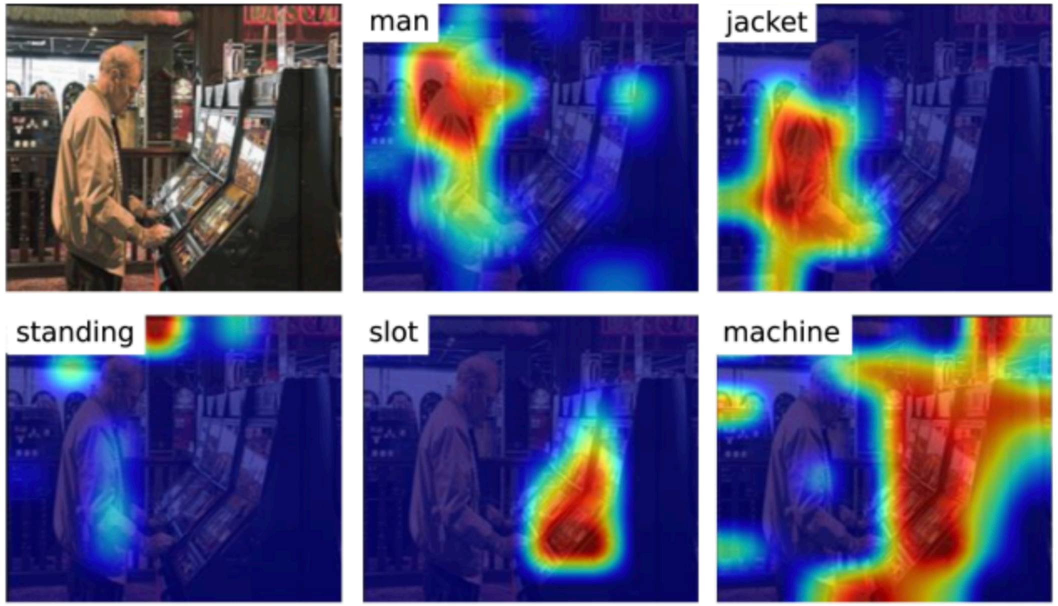
Idea:
We train a CNN for feature extraction and a model (e.g. MLP, decision tree, logistic regression) for classification, simultaneously and end-to-end.

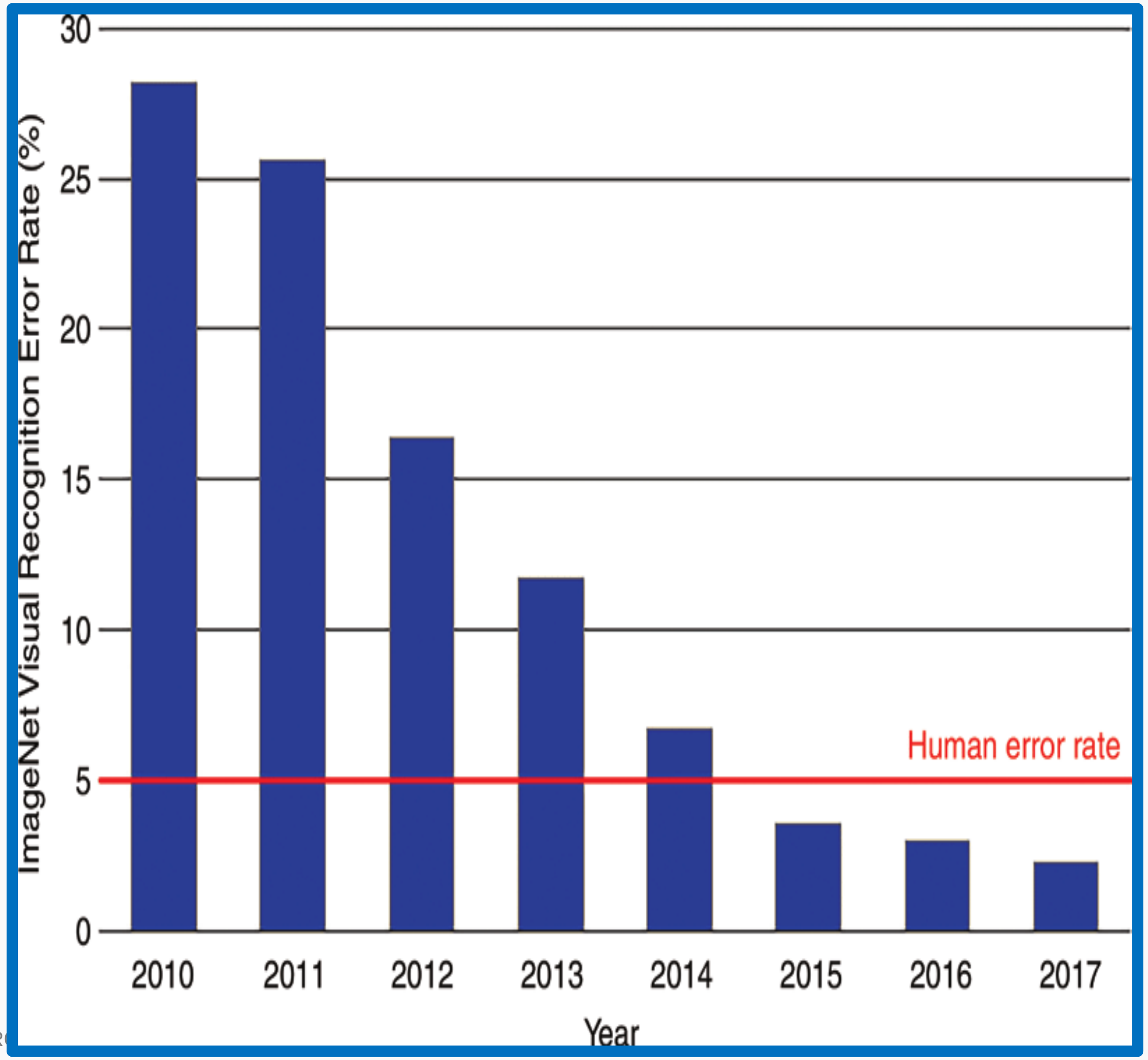


CNNs: Successful object detection

IMAGENET challenge:

- New model architectures consistently outperform even human error rate
- Ablation studies and saliency maps and confirm models are not overfitting





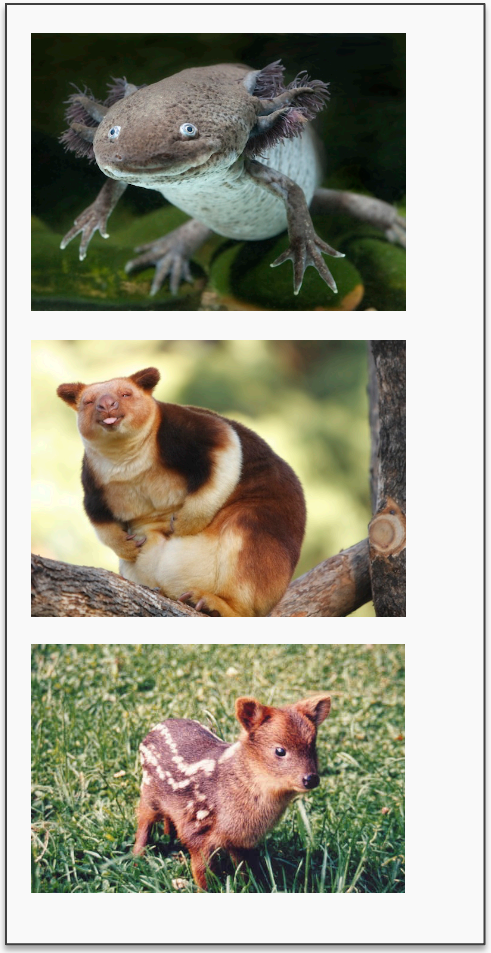
CNNs: So what is the problem?

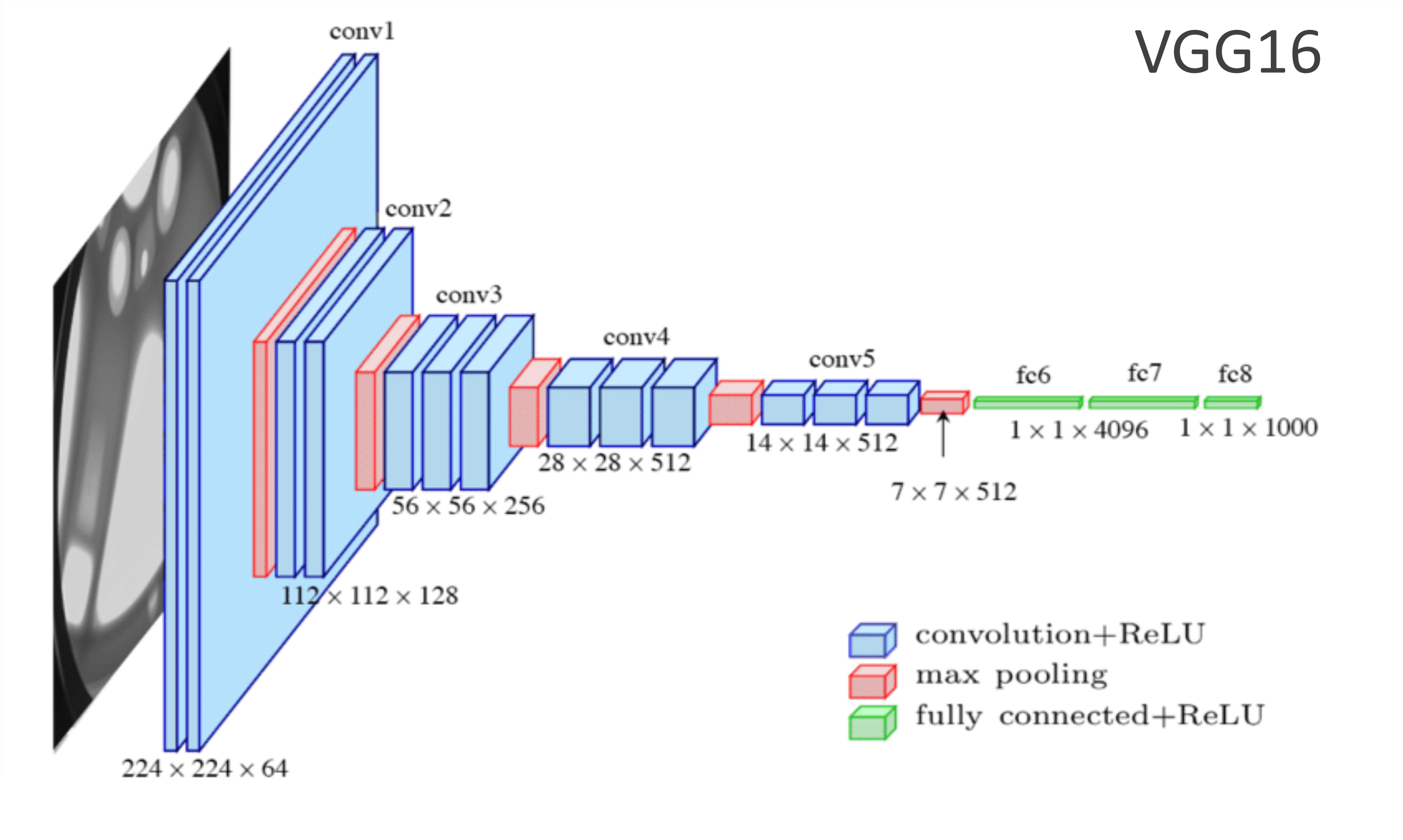
- IMAGENET has more than a 14 million label images and more than 1000 categories.
- However, the imagenet challenge is only a very tiny subset of all possible categories for which we may not have a lot of training data.
- Eg. Can you guess the animals in the images below?



Classify Rarest Animals

Classify Rarest Animals

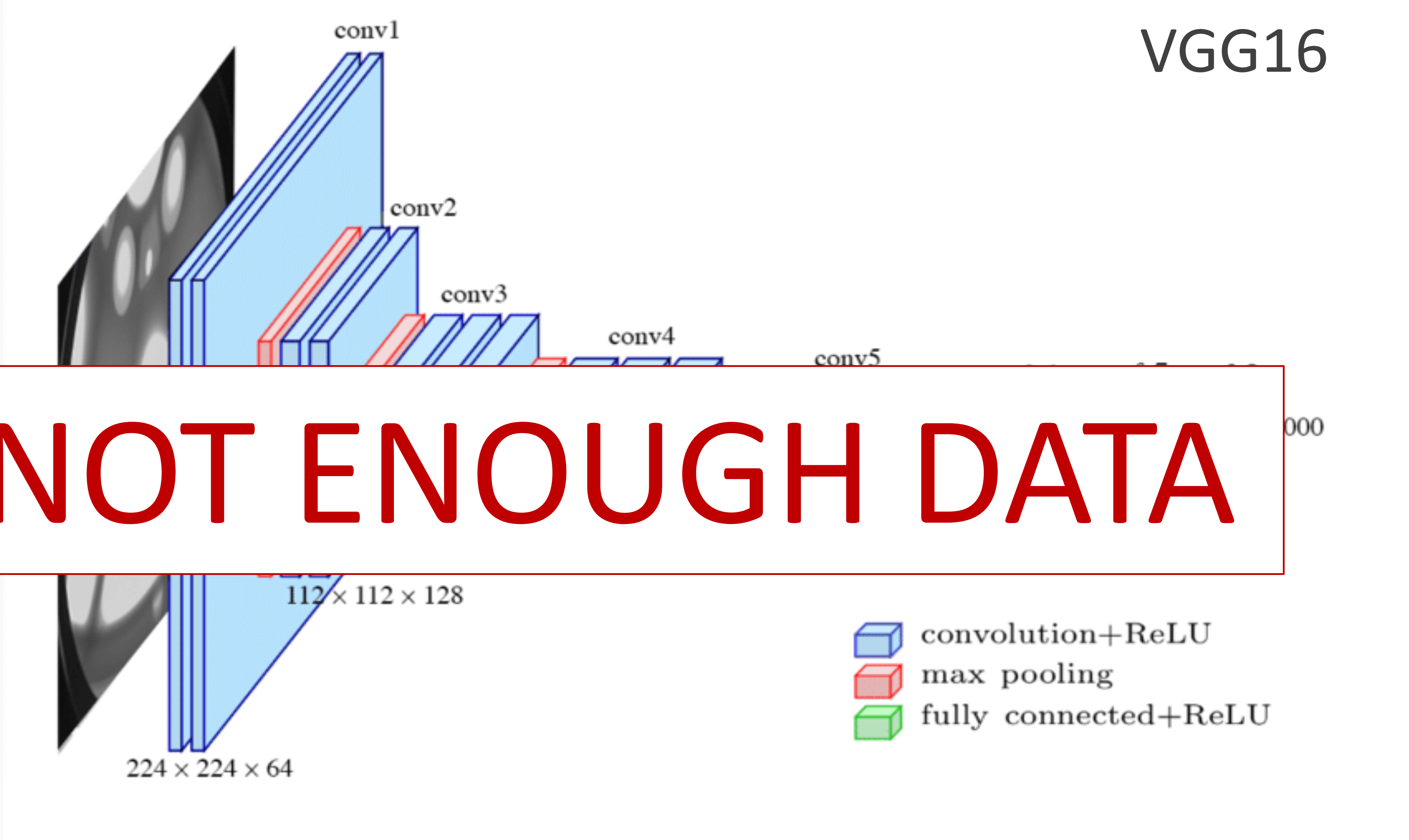




Number of parameters: 134,268,737

Data Set: Few hundred images

Classify Rarest Animals

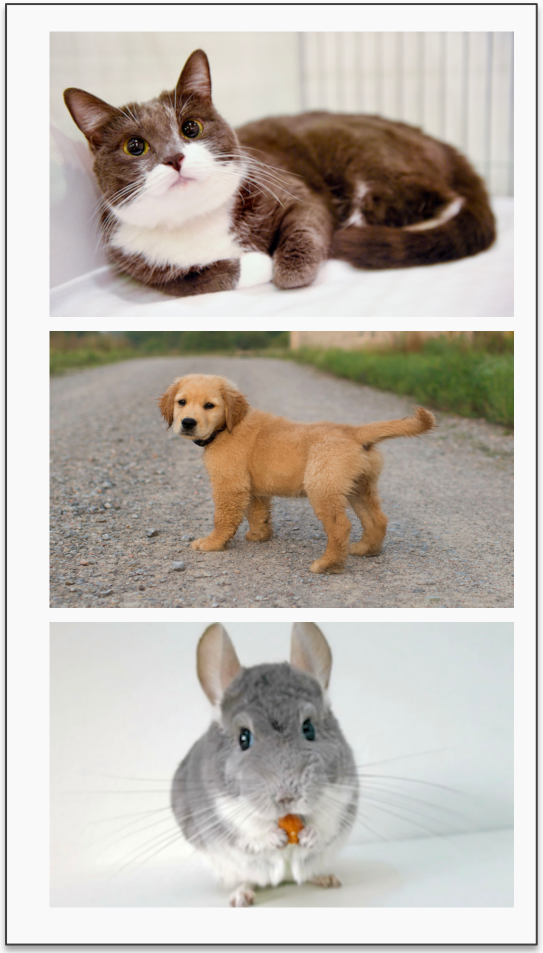


NOT ENOUGH DATA

Number of parameters: 134,268,737

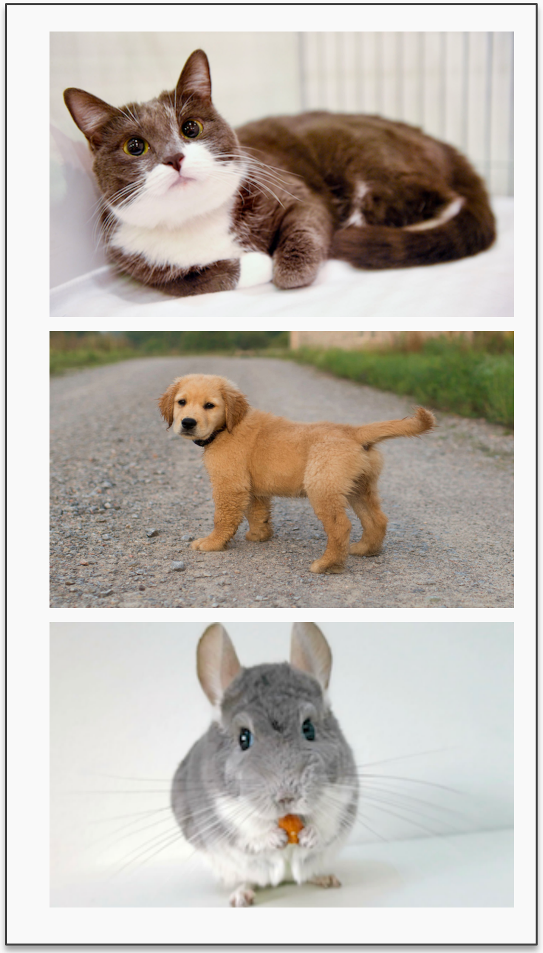
Data Set: Few hundred images

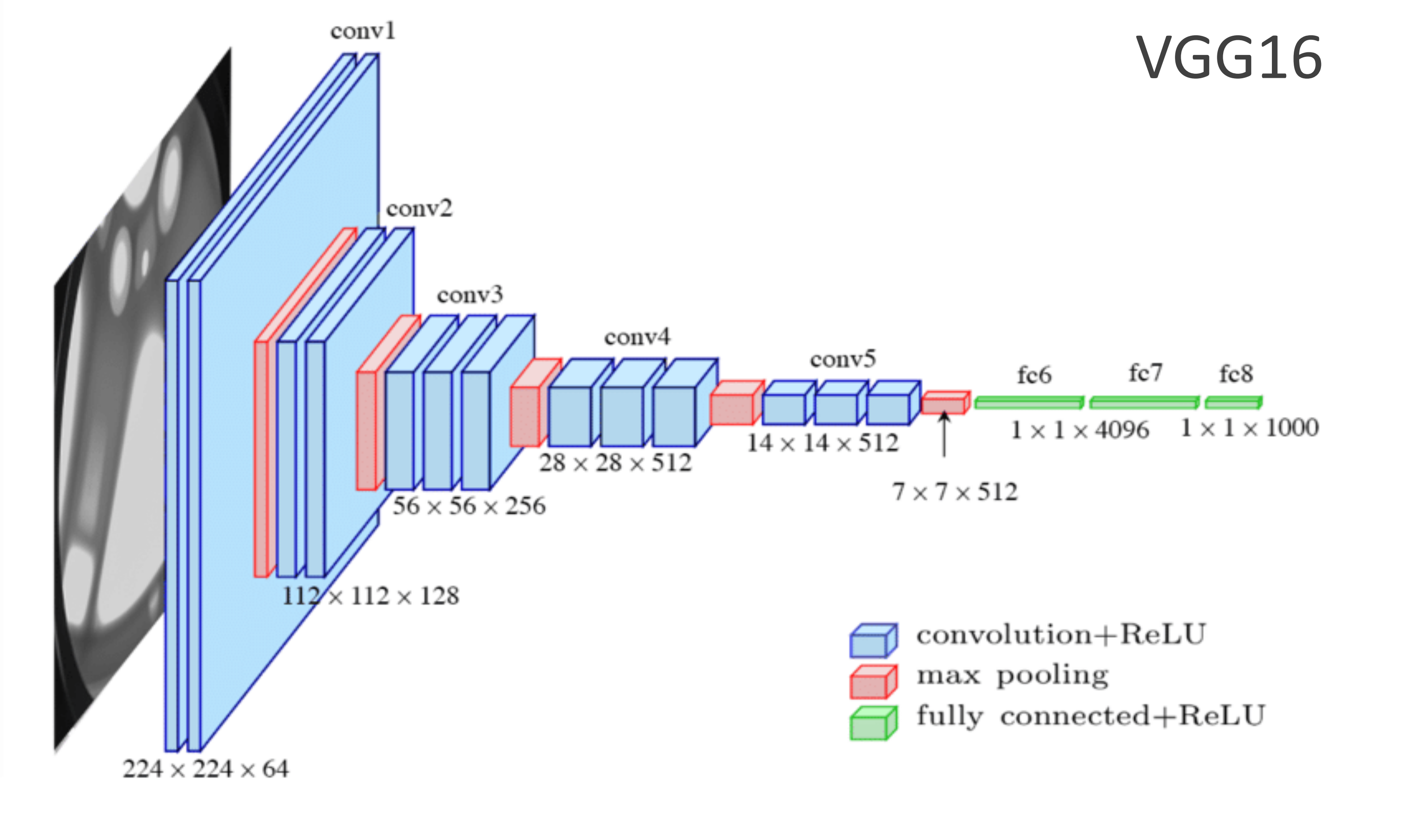
Classify Cats, Dogs, Chinchillas etc



VGG16

Classify Cats, Dogs, Chinchillas etc

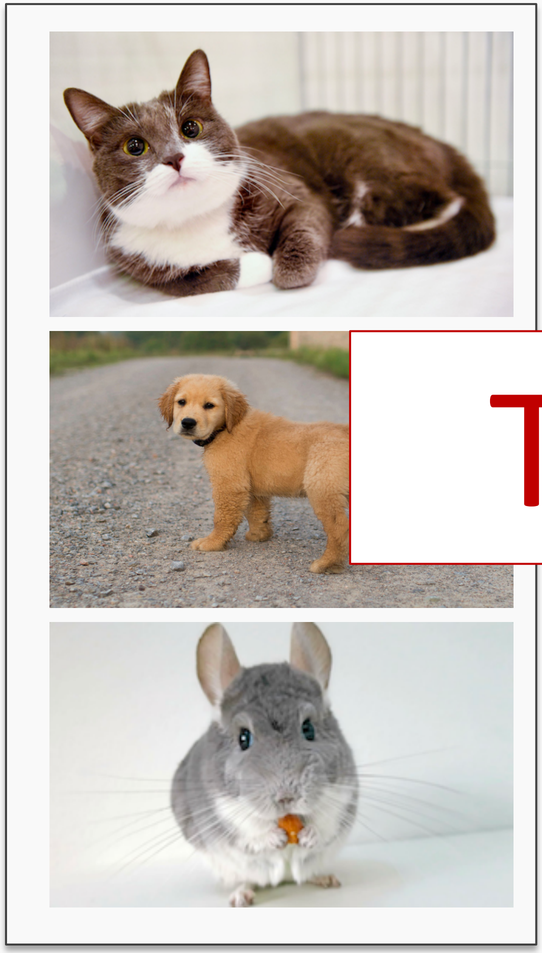


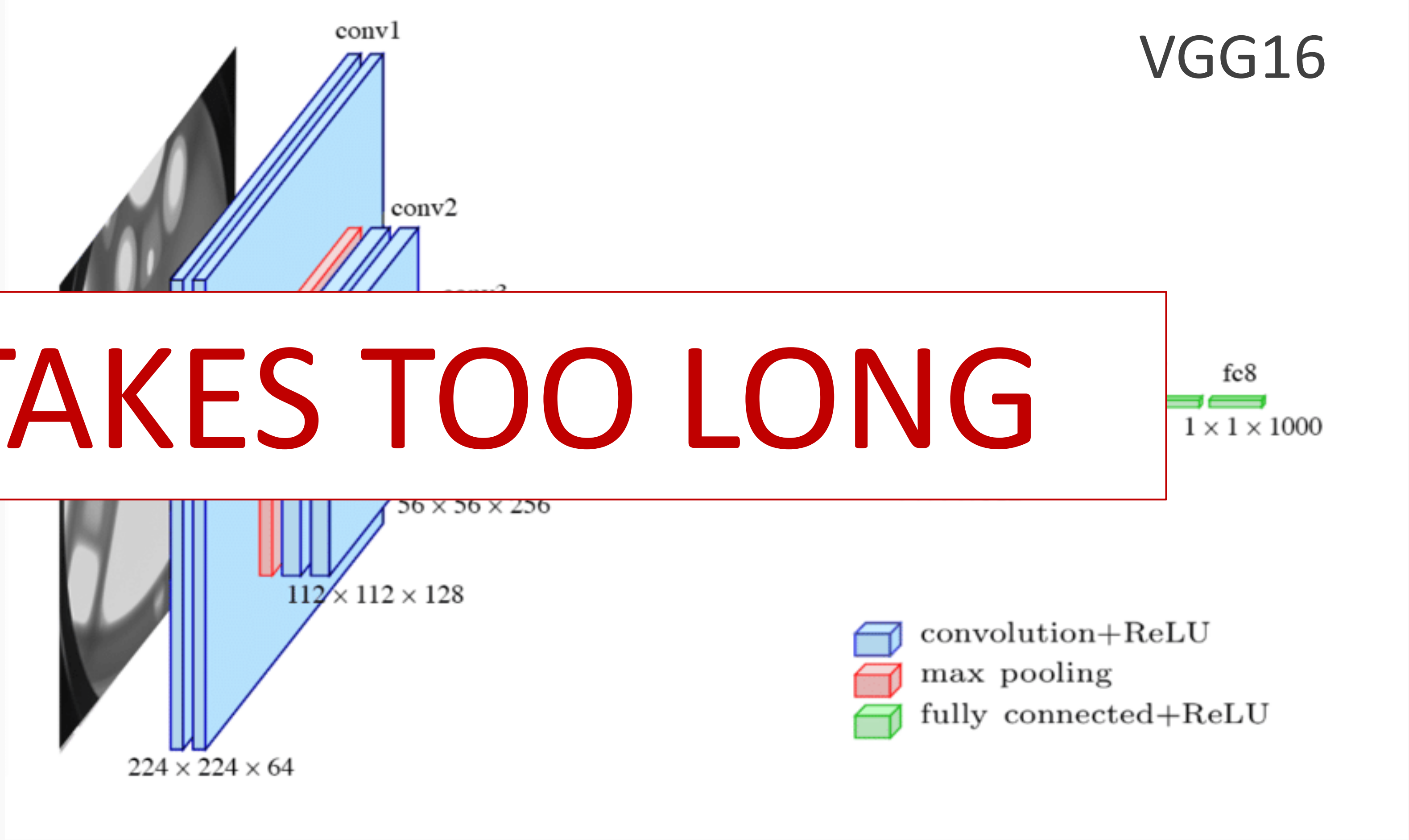


Number of parameters: 134,268,737

Enough training data. ImageNet approximate 1.2M

Classify Cats, Dogs, Chinchillas etc



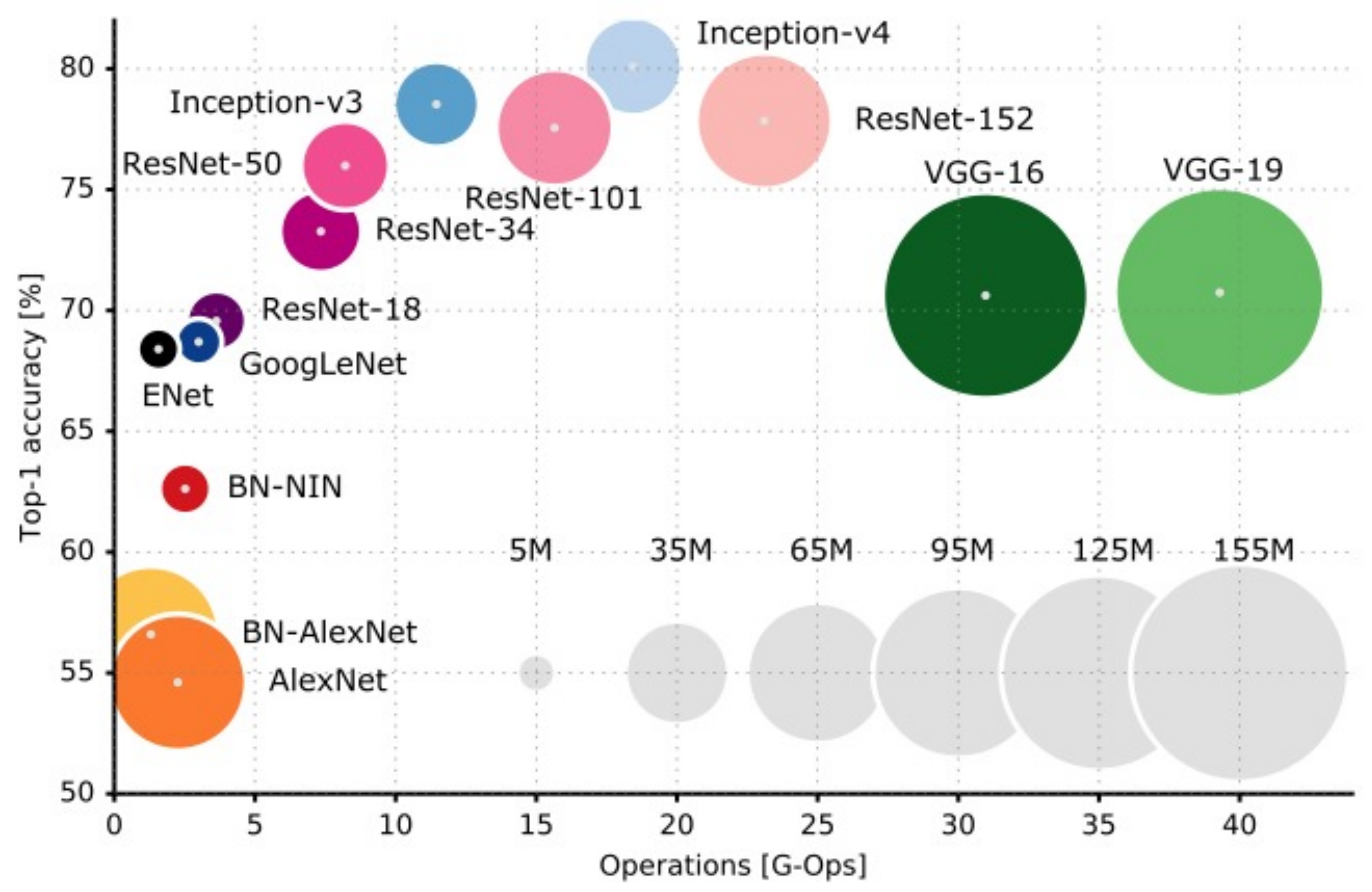


TAKES TOO LONG

Number of parameters: 134,268,737

Enough training data. ImageNet approximate 1.2M

Training time for SOTAs



Transfer Learning To The Rescue

How do you build an image classifier that can be trained in a few minutes on a CPU with very little data?

?

Basic idea of Transfer Learning

Wikipedia:
Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.[1]

Basic idea of Transfer Learning

Train a ML model M for a task T using a dataset D_s

Wikipedia:
Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.[1]

Basic idea of Transfer Learning

Train a ML model M for a task T using a dataset D_s

Use M on a new dataset D_T for the same task T

Wikipedia:
Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.[1]

Basic idea of Transfer Learning

Train a ML model M for a task T using a dataset D_s

Use M on a new dataset D_T for the same task T

Use part of M on original dataset D_s for a new task T_n

Wikipedia:
Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.[1]

Basic idea of Transfer Learning

Train a ML model M for a task T using a dataset D_s

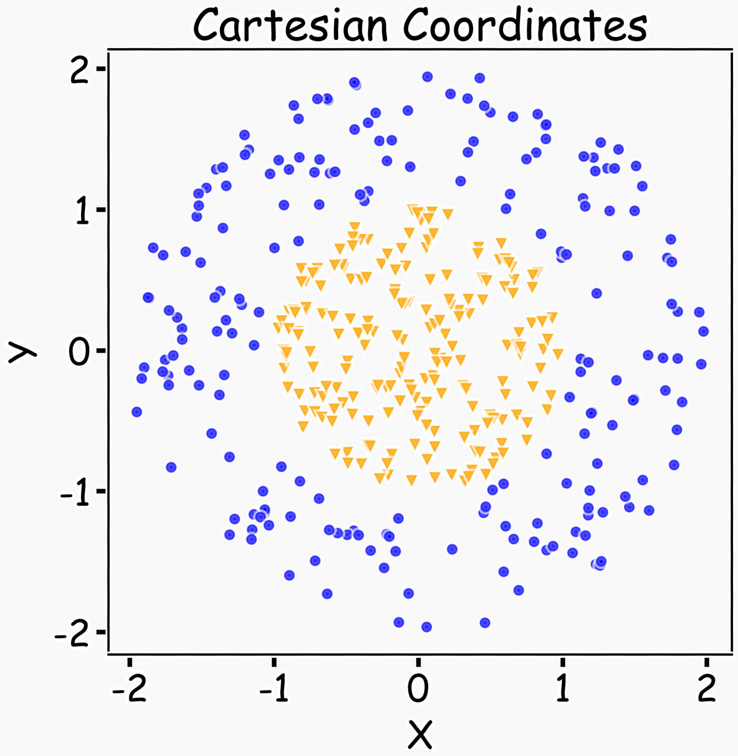
Use M on a new dataset D_T for the same task T

Use part of M on original dataset D_s for a new task T_n

Use part of M on a new dataset D_T for a new task T_n

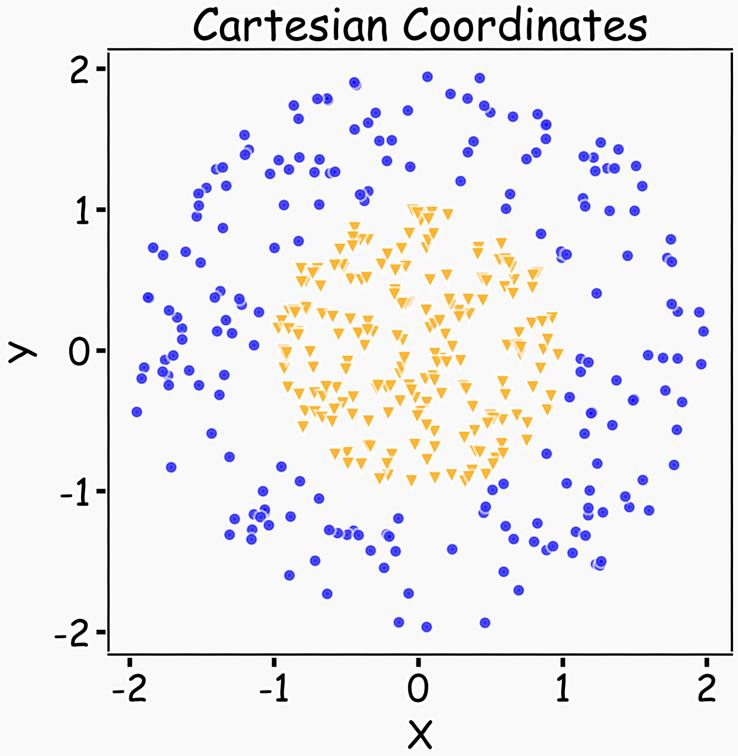
Wikipedia:
Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.[1]

Key Idea: Representation Learning



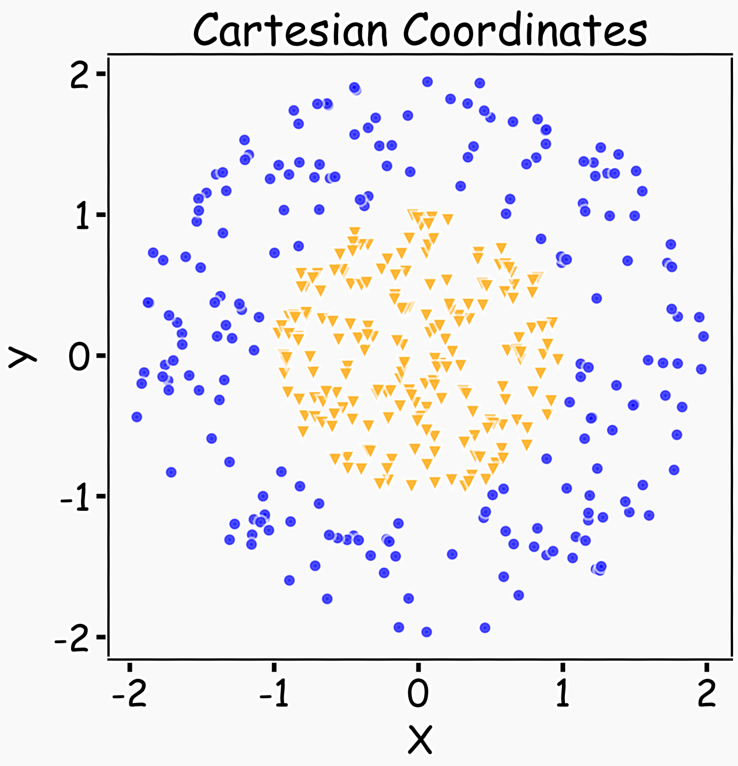
Key Idea: Representation Learning

Relatively difficult task



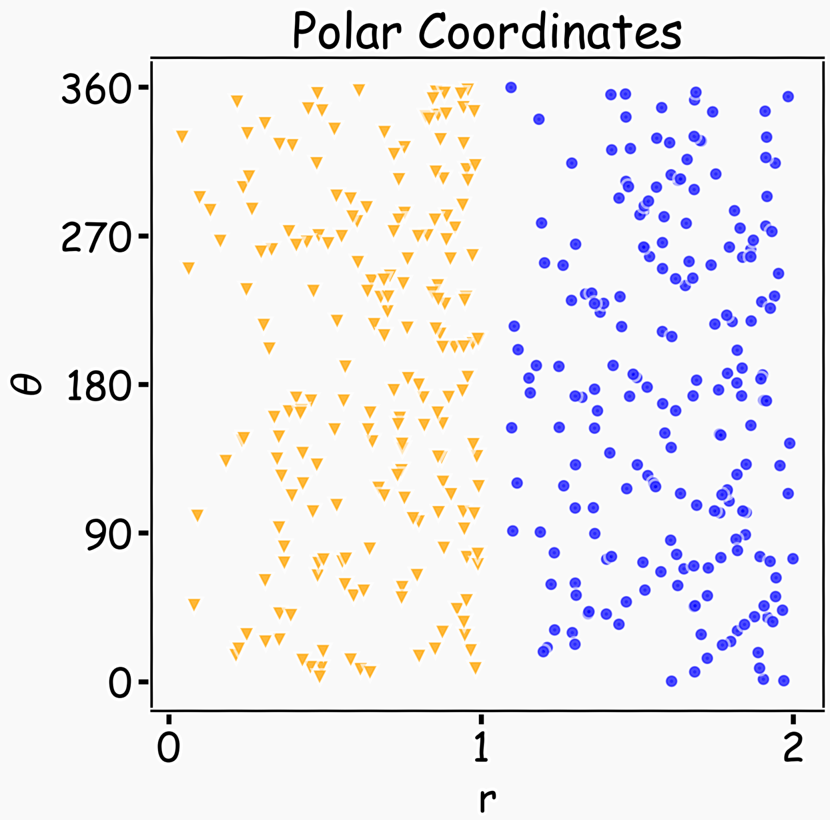
Key Idea: Representation Learning

Relatively difficult task



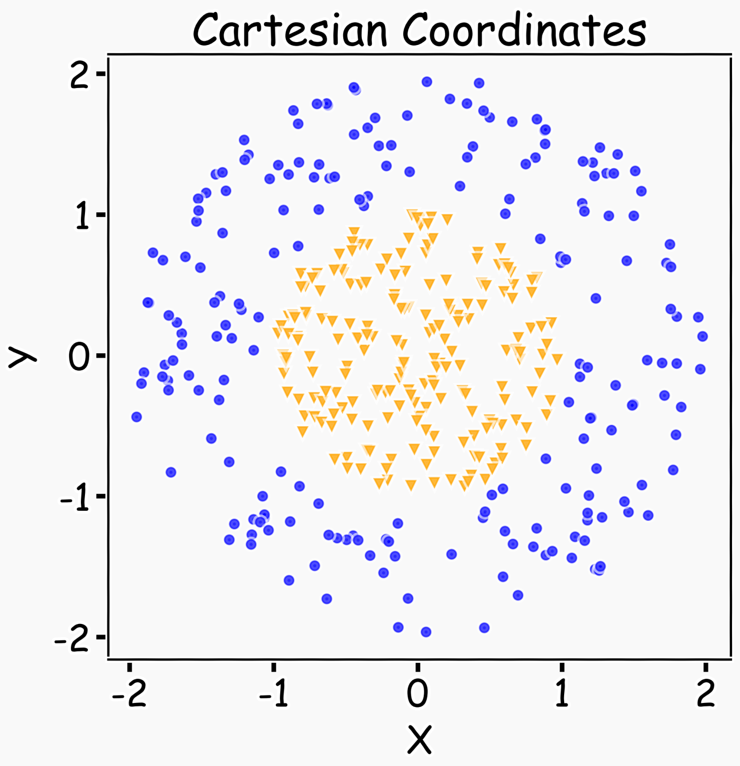
Tranform:

$$(X, Y) \rightarrow (r, \theta)$$



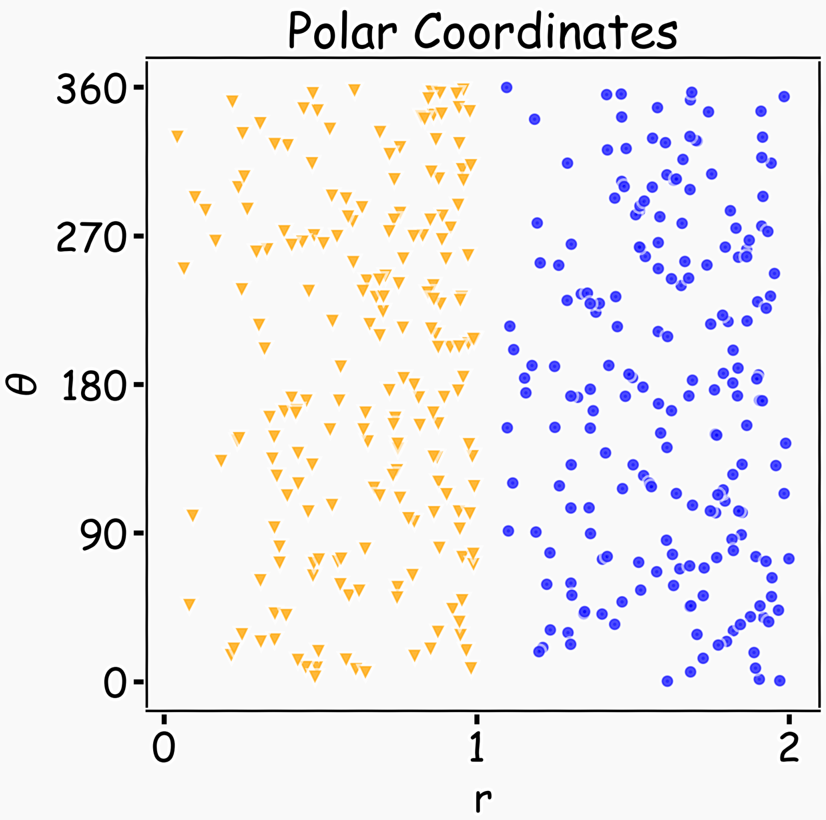
Key Idea: Representation Learning

Relatively difficult task



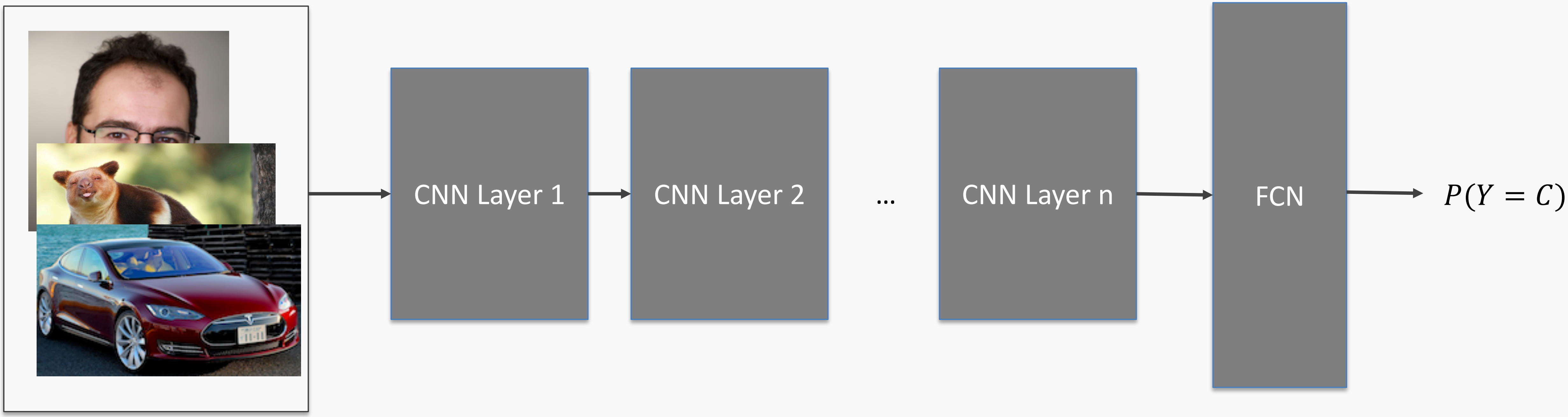
Tranform:
$$(X,Y) \rightarrow (r,\theta)$$

Easier task



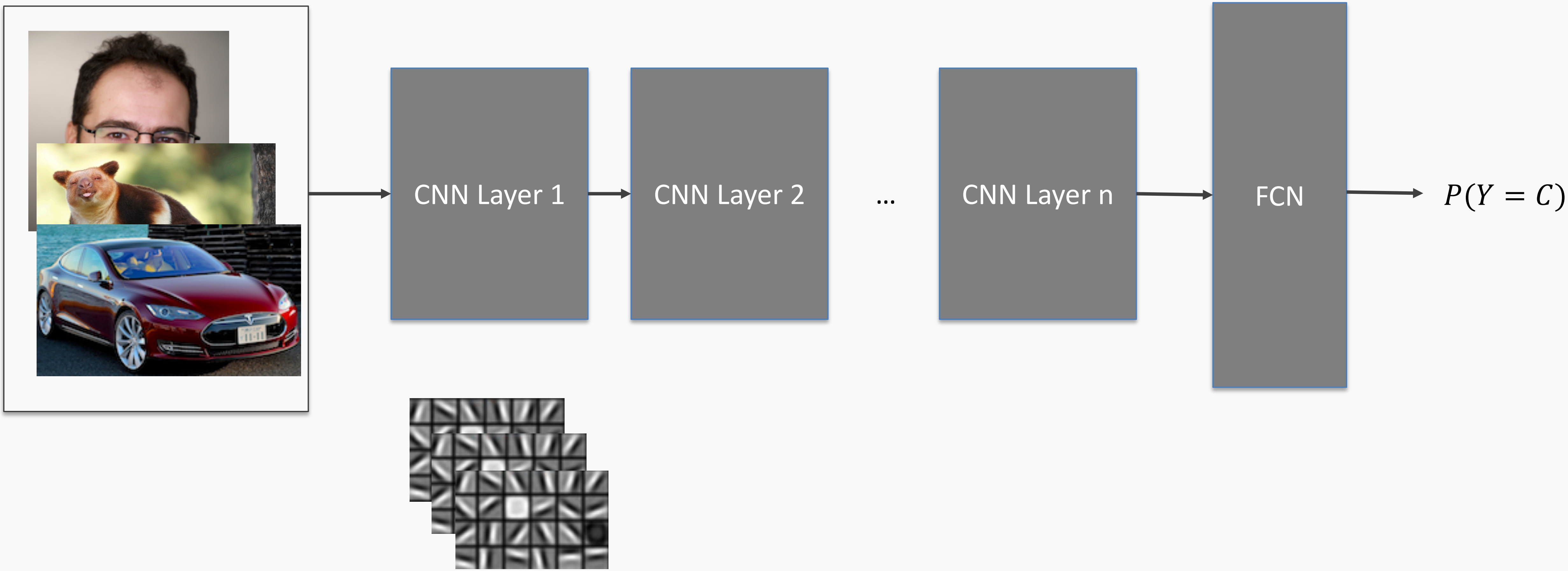
Representation Learning

Task: classify cars, people, animals and objects



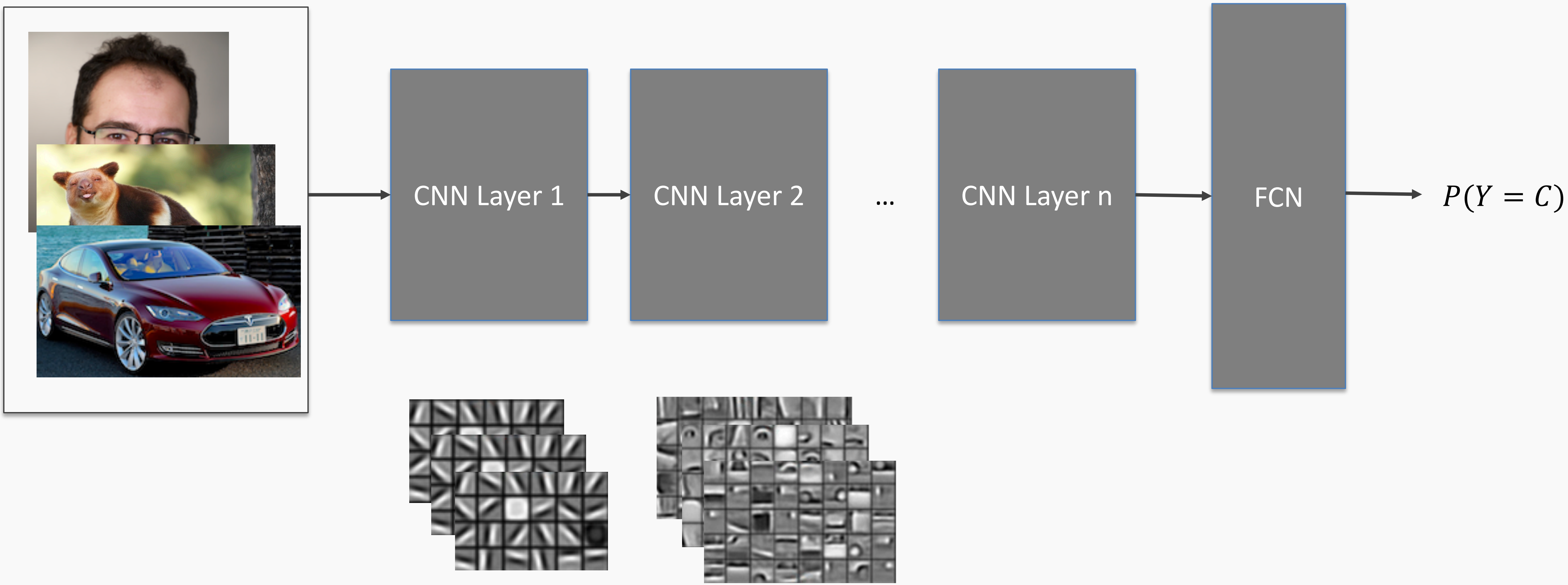
Representation Learning

Task: classify cars, people, animals and objects



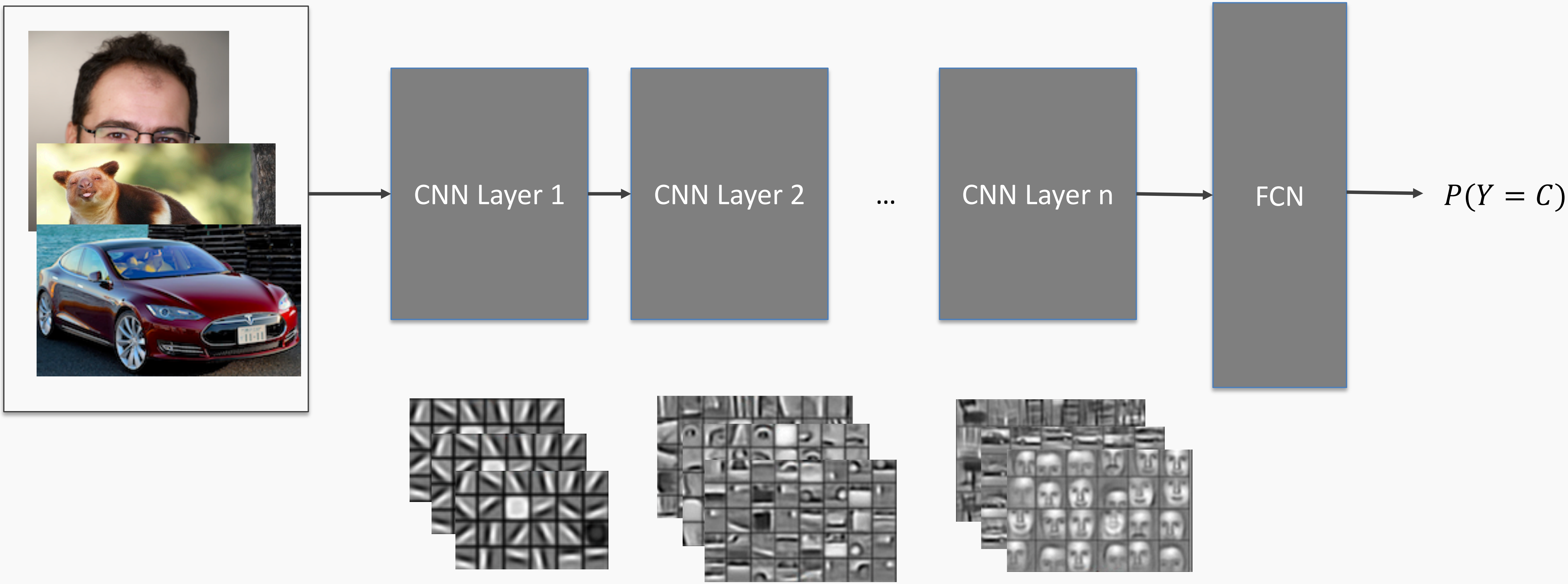
Representation Learning

Task: classify cars, people, animals and objects



Representation Learning

Task: classify cars, people, animals and objects



Transfer Learning To The Rescue

How do you make an image classifier that can be trained in a few minutes on a CPU with very little data?

Transfer Learning To The Rescue

How do you make an image classifier that can be trained in a few minutes on a CPU with very little data?

Use pre-trained models, i.e., models with known weights.

Transfer Learning To The Rescue

How do you make an image classifier that can be trained in a few minutes on a CPU with very little data?

Use pre-trained models, i.e., models with known weights.

Main Idea: earlier layers of a network learn low level features, which can be adapted to new domains by changing weights at later and fully-connected layers.

Transfer Learning To The Rescue

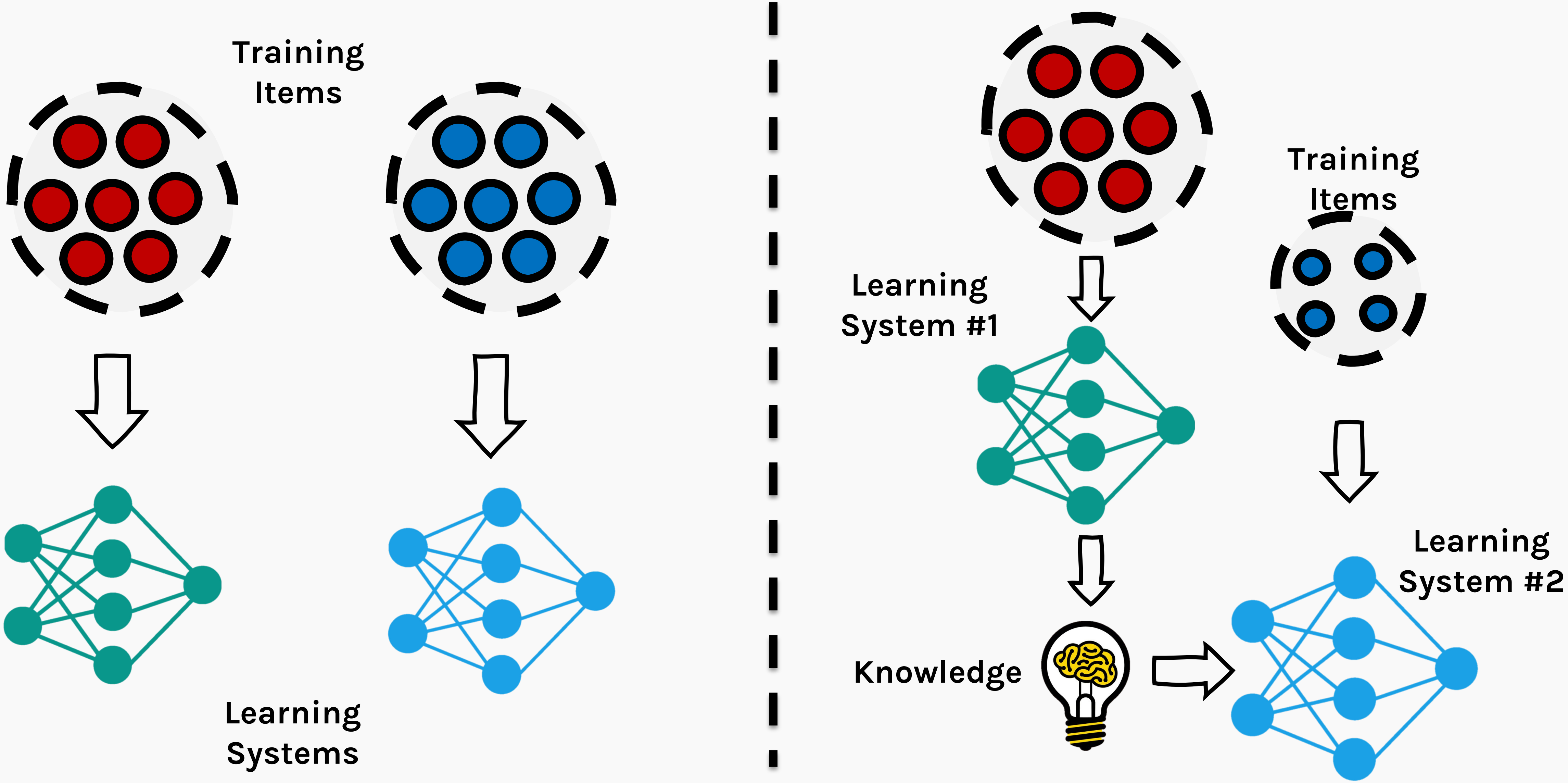
How do you make an image classifier that can be trained in a few minutes on a CPU with very little data?

Use pre-trained models, i.e., models with known weights.

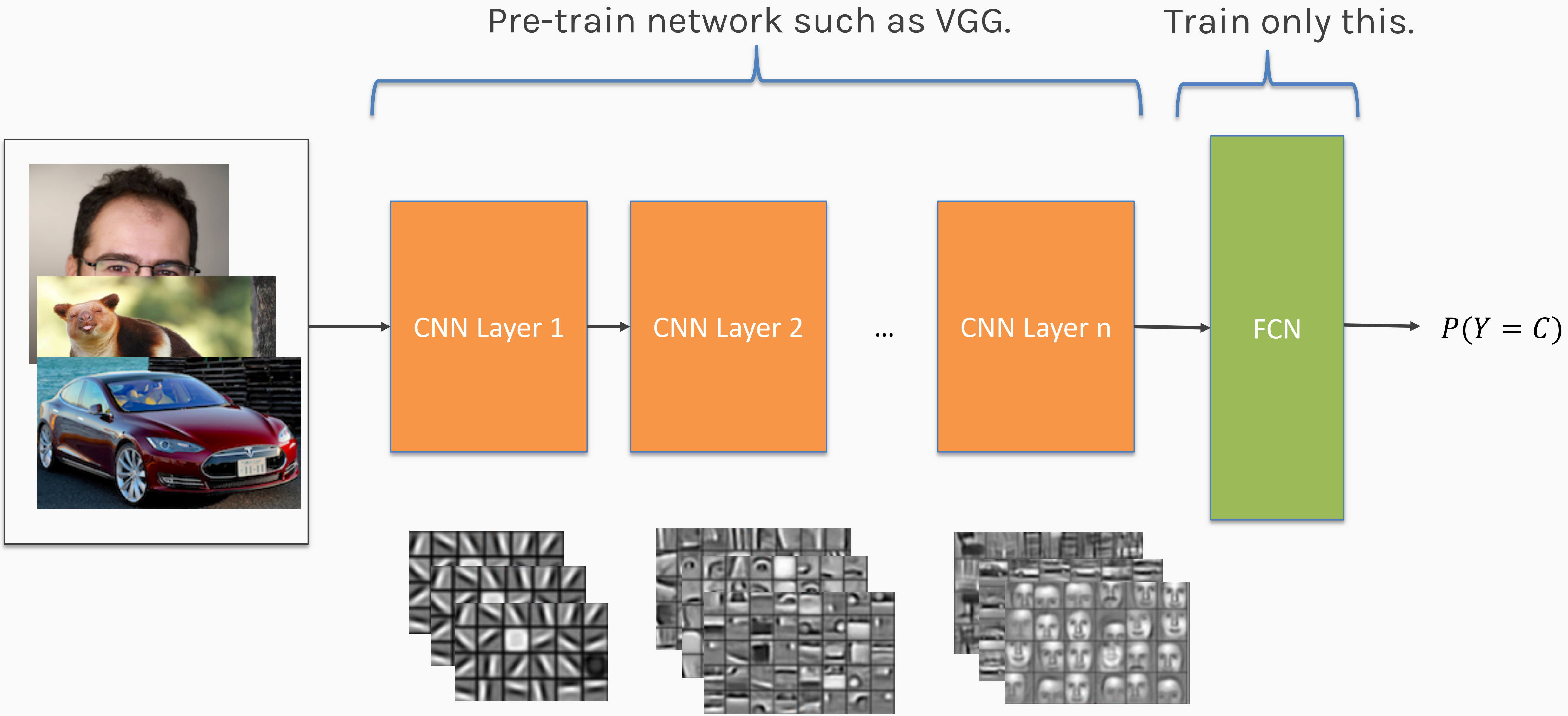
Main Idea: earlier layers of a network learn low level features, which can be adapted to new domains by changing weights at later and fully-connected layers.

Example: use ImageNet trained with any sophisticated huge network. Then retrain it on a few images.

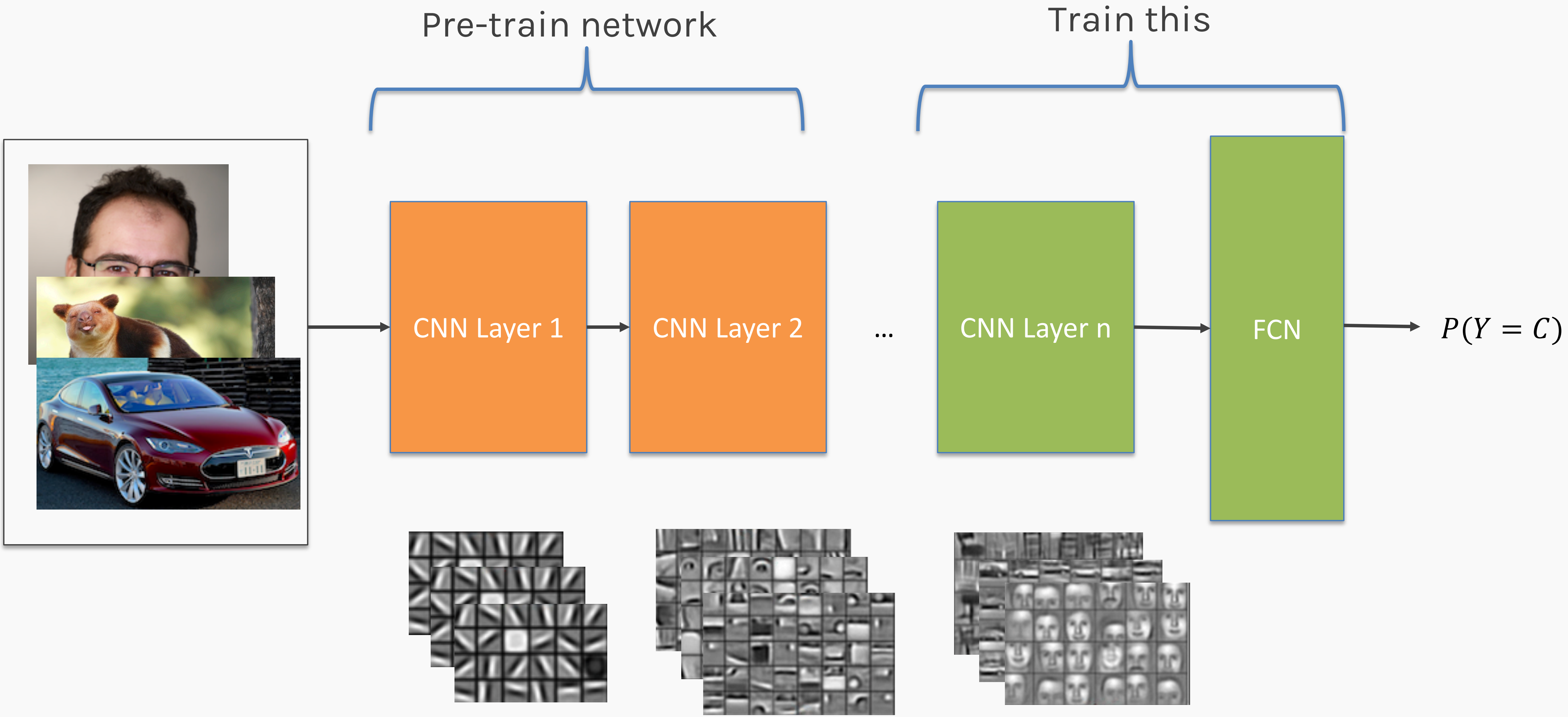
Traditional Machine Learning vs Transfer Learning



Transfer Learning: **Only train dense layers**

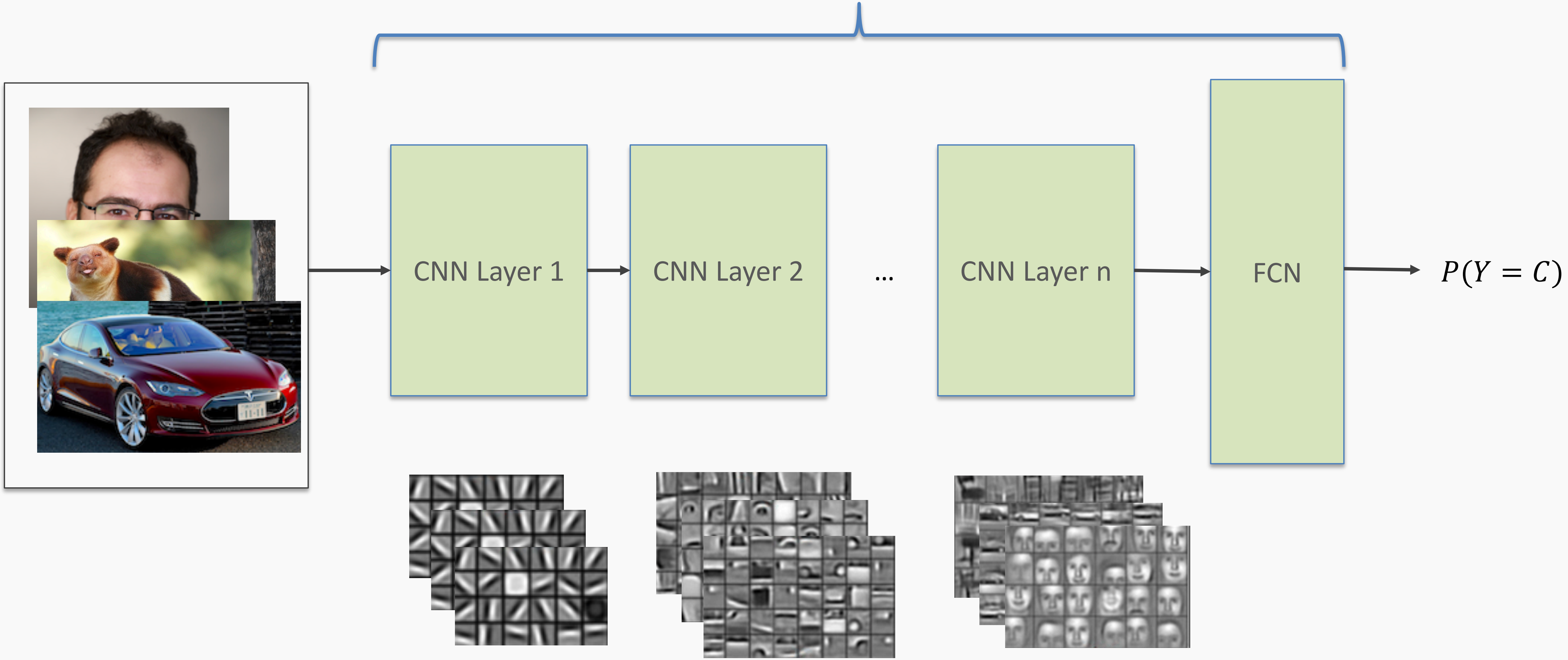


Transfer Learning: **Train last few Conv layers and dense layers**



Transfer Learning: **Fine tuning**

Train everything but start with weights that are trained already



Transfer Learning for Deep Learning

What people think:

- You can't do deep learning unless you have a million labeled examples.

What people do:

- You can train on a nearby objective for which is easy to generate labels (ImageNet).
- You can transfer learned representations from a relate task.
- You can learn representations from unlabeled data.

Transfer Learning for Deep Learning

What people think:

- You can't do deep learning unless you have a millio

Indeed, you can train a network for classification and use it for segmentation.

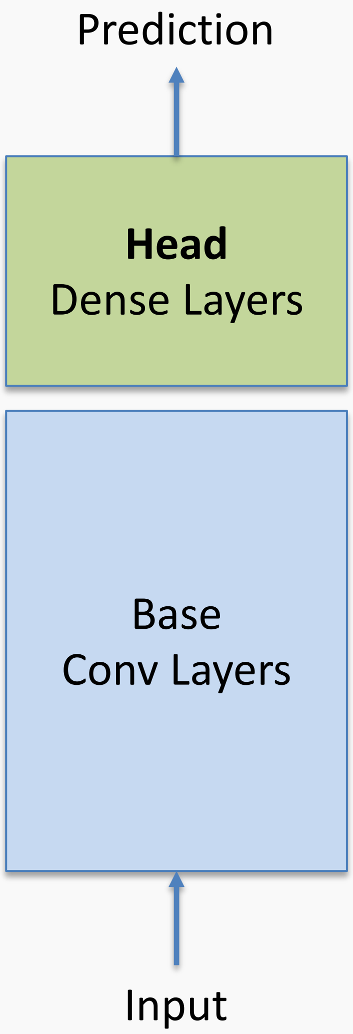
What people do:

- You can train on a nearby objective for which is eas to generate labels (ImageNet).
- You can transfer learned representations from a relate task.
- You can learn representations from unlabeled data.

We will revisit this idea when we will talk about autoencoders and language model.

Transfer learning – **Feature-Representation Extraction**

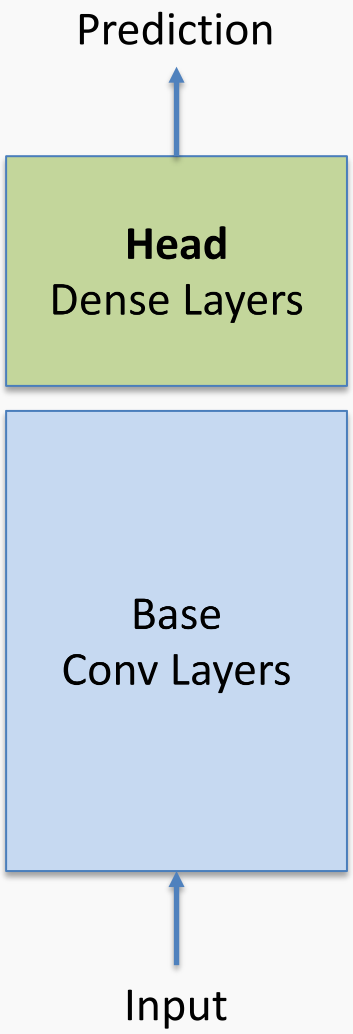
Use representations learned by big net to extract features from new samples, which are then fed to a new classifier.



Transfer learning – **Feature-Representation Extraction**

Use representations learned by big net to extract features from new samples, which are then fed to a new classifier.

Example: Classify cats and dogs from your own images:



Transfer learning – **Feature-Representation Extraction**

Use representations learned by big net to extract features from new samples, which are then fed to a new classifier.

Example: Classify cats and dogs from your own images:

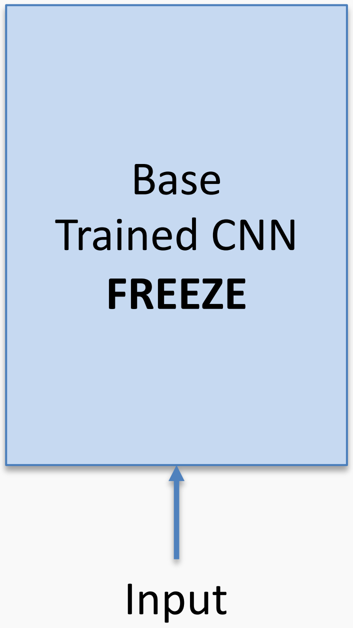
- Keep (frozen) convolutional **base** from big model, since convolutional base is "generic".

Transfer learning – **Feature-Representation Extraction**

Use representations learned by big net to extract features from new samples, which are then fed to a new classifier.

Example: Classify cats and dogs from your own images:

- Keep (frozen) convolutional **base** from big model, since convolutional base is “generic”.
- Throw away **the fully connected** layers (head) since these layers have no notion of space.



Transfer learning – **Feature-Representation Extraction**

Use representations learned by big net to extract features from new samples, which are then fed to a new classifier.

Example: Classify cats and dogs from your own images:

- Keep (frozen) convolutional **base** from big model, since convolutional base is “generic”.
- Throw away **the fully connected** layers (head) since these layers have no notion of space.
- Introduce new dense layers. It can be new architecture or the same as the original model.

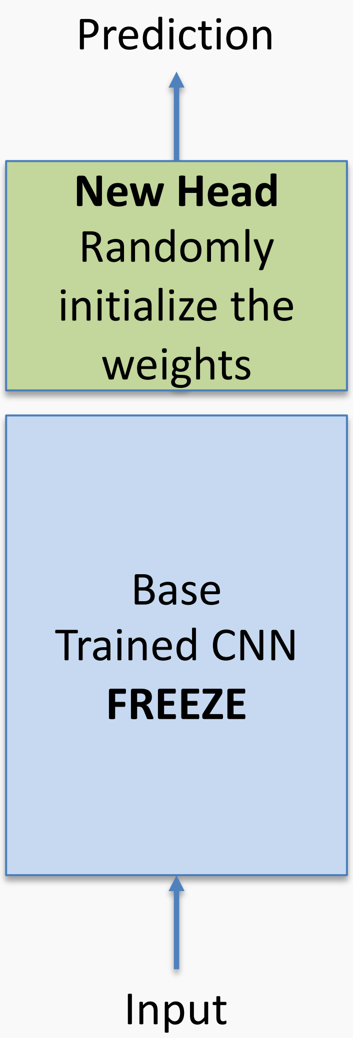
Transfer learning – **Feature-Representation Extraction**

Use representations learned by big net to extract features from new samples, which are then fed to a new classifier.

Example: Classify cats and dogs from your own images:

- Keep (frozen) convolutional **base** from big model, since convolutional base is “generic”.
- Throw away **the fully connected** layers (head) since these layers have no notion of space.
- Introduce new dense layers. It can be new architecture or the same as the original model.

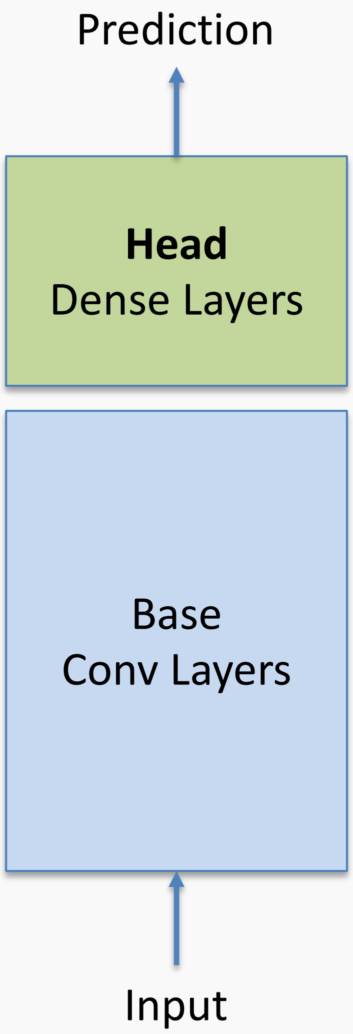
Note: Classify cats and dogs from your own images. Since there are both dogs and cats in *ImageNet* you could get **away** with using the head FC layers as well (instance TL). But by throwing it away you can learn more from other dog/cat images.



Transfer learning – **Fine-tuning**

Up to now we have frozen the entire convolutional base.

Remember that earlier layers learn highly generic feature maps (edges, colors, textures).

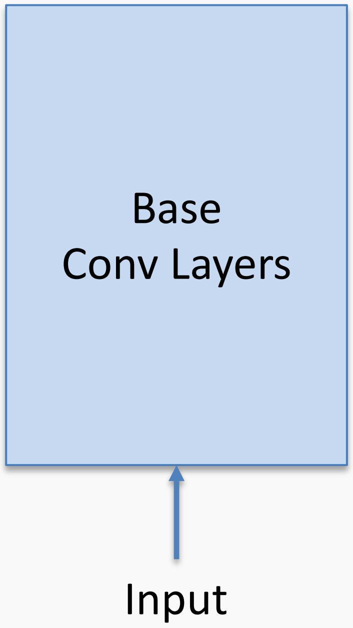


Transfer learning – **Fine-tuning**

Up to now we have frozen the entire convolutional base.

Remember that earlier layers learn highly generic feature maps (edges, colors, textures).

- Throw away **the fully connected** layers (head) since these layers have no notion of space.

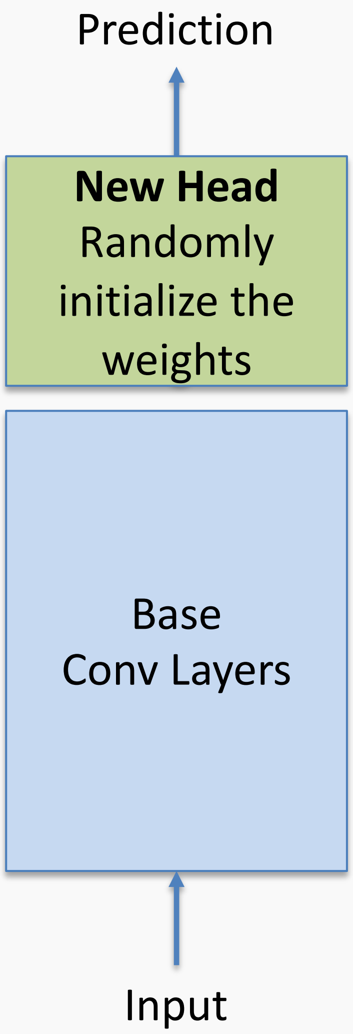


Transfer learning – **Fine-tuning**

Up to now we have frozen the entire convolutional base.

Remember that earlier layers learn highly generic feature maps (edges, colors, textures).

- Throw away **the fully connected** layers (head) since these layers have no notion of space.
- Introduce new dense layers. It can be new architecture or the same as the original model. Train the new head while keeping the base fixed.



Transfer learning – **Fine-tuning**

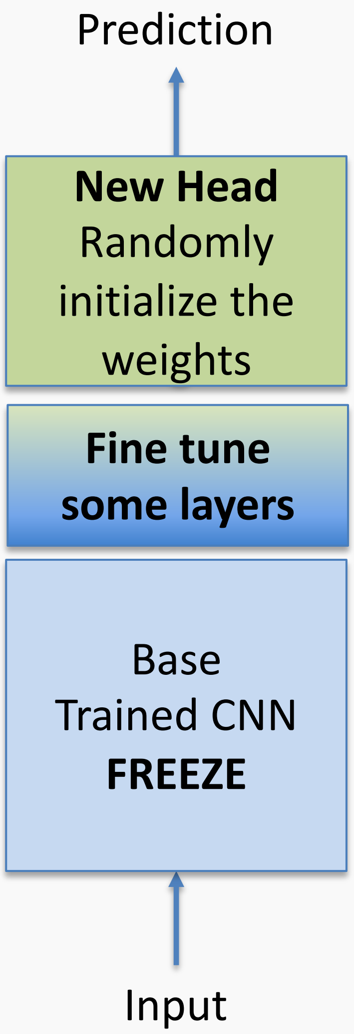
Up to now we have frozen the entire conv

Remember that earlier layers learn highl
(edges, colors, textures).

Since you are now in a better part of the loss surface already, gradients won't be terribly high, but we still need to be careful. Thus, often we use a **very low learning rate**.

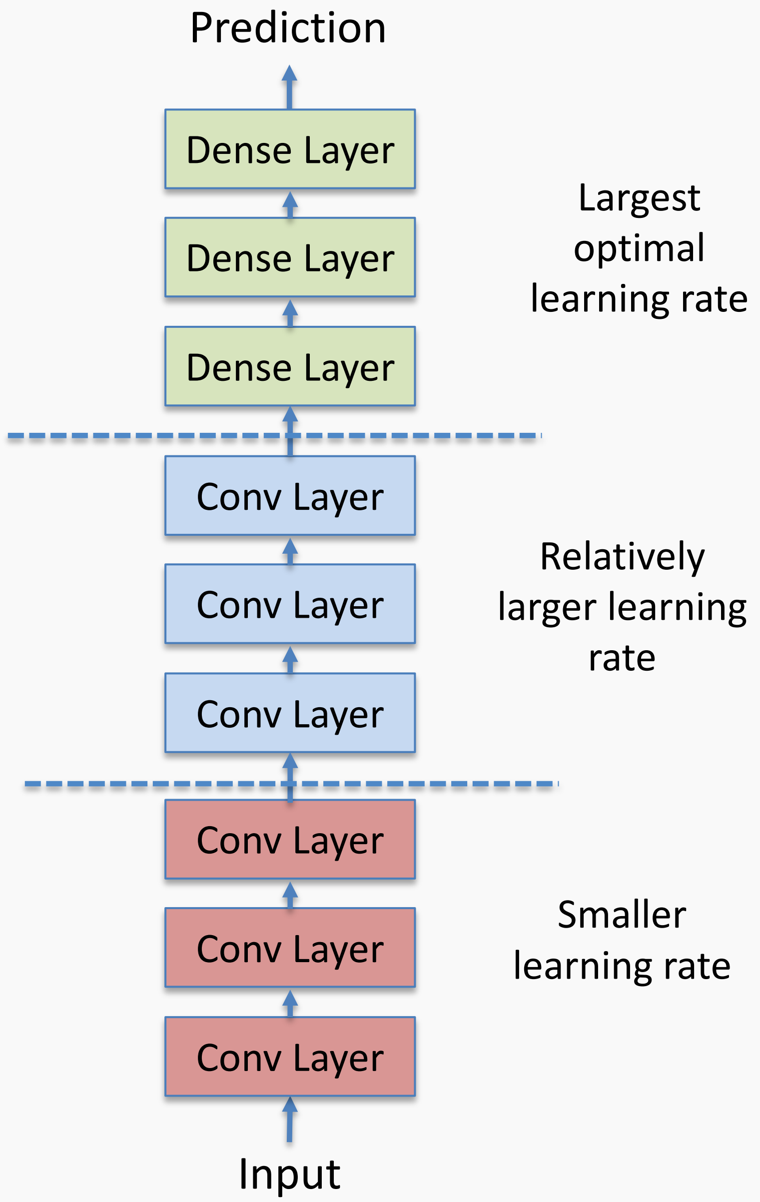
- Throw away **the fully connected** layers (head) since these layers have no notion of space.
- Introduce new dense layers. It can be new architecture or the same as the original model. Train the new head while keeping the base fixed.
- To particularize the model to our task, its often worth tuning the later layers as well. Unfreeze some "later" layers in the base net and now train the base net and FC net together.

Note: But we must be very careful not to have big gradient updates.



Transfer Learning for Deep Learning: Differential Learning Rates

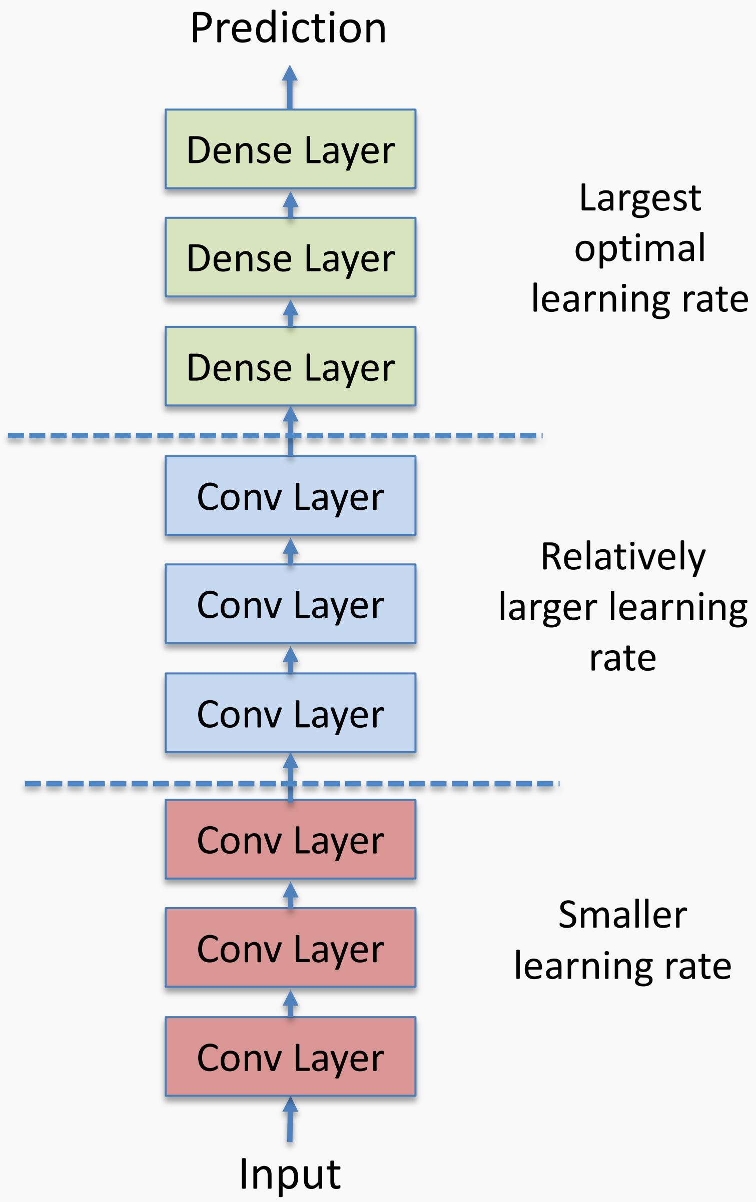
- A low learning rate can take a lot of time to train on the "later" layers. Since we trained the FC head earlier, we could probably retrain them at a higher learning rate.



Transfer Learning for Deep Learning: Differential Learning Rates

- A low learning rate can take a lot of time to train on the "later" layers. Since we trained the FC head earlier, we could probably retrain them at a higher learning rate.

- General Idea: **Train different layers at different rates**.

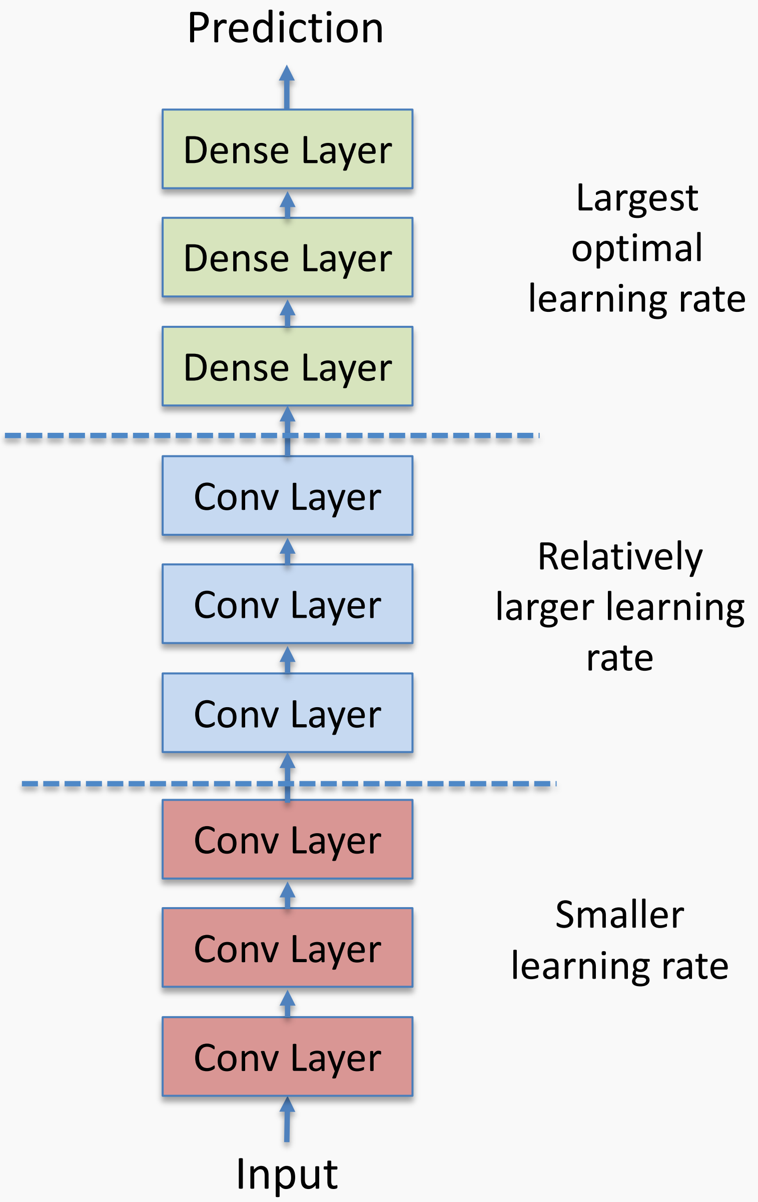


Transfer Learning for Deep Learning: Differential Learning Rates

- A low learning rate can take a lot of time to train on the "later" layers. Since we trained the FC head earlier, we could probably retrain them at a higher learning rate.

- General Idea: **Train different layers at different rates**.

- Each "earlier" layer or layer group (the color-coded layers in the image) can be trained at 3x-10x smaller learning rate than the next "later" one.



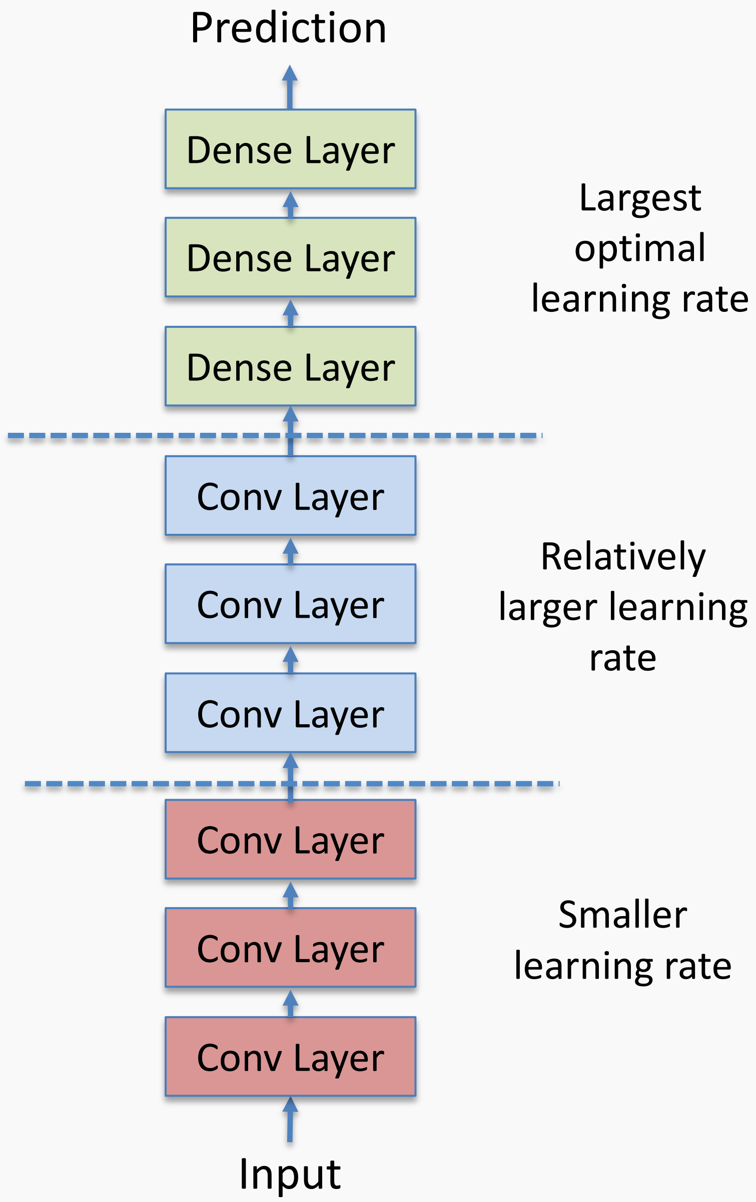
Transfer Learning for Deep Learning: Differential Learning Rates

- A low learning rate can take a lot of time to train on the "later" layers. Since we trained the FC head earlier, we could probably retrain them at a higher learning rate.

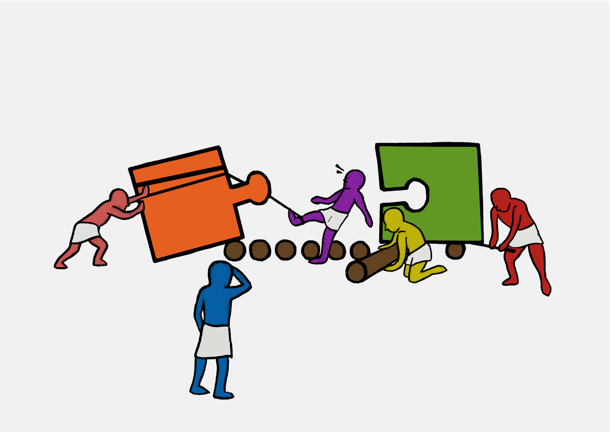
- General Idea: **Train different layers at different rates**.

- Each "earlier" layer or layer group (the color-coded layers in the image) can be trained at 3x-10x smaller learning rate than the next "later" one.

- One could even train the entire network again this way until we overfit and then step back some epochs.



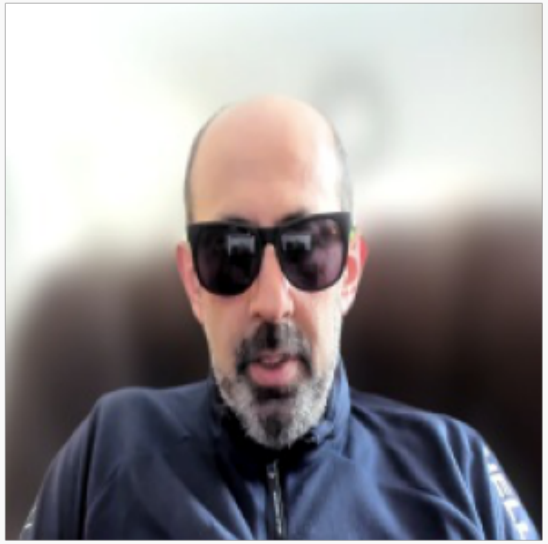
Exercise: Transfer Learning



The goal of this exercise is to use Transfer Learning to achieve near-perfect accuracy for a highly customized task. The task at hand is to distinguish images of people with Sunglasses or Hat.

Prediction: people_with_sunglasses
(with confidence: 0.99)

Prediction: people_with_sunglasses
(with confidence: 0.99)



Prediction: people_with_sunglasses
(with confidence: 0.99)