

# Transfer Learning

Advanced Deep Learning

# Outline

---

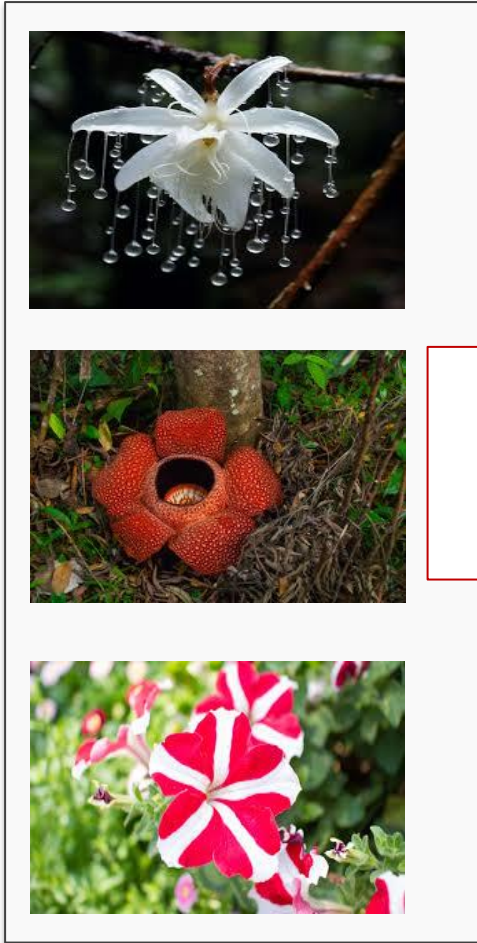
1. Communications
2. Motivation
3. The Basics idea for Transfer Learning
4. Formal Definition
5. Transfer Learning Strategies
6. Demo: How Transfer Learning Works in Practice

# Outline

---

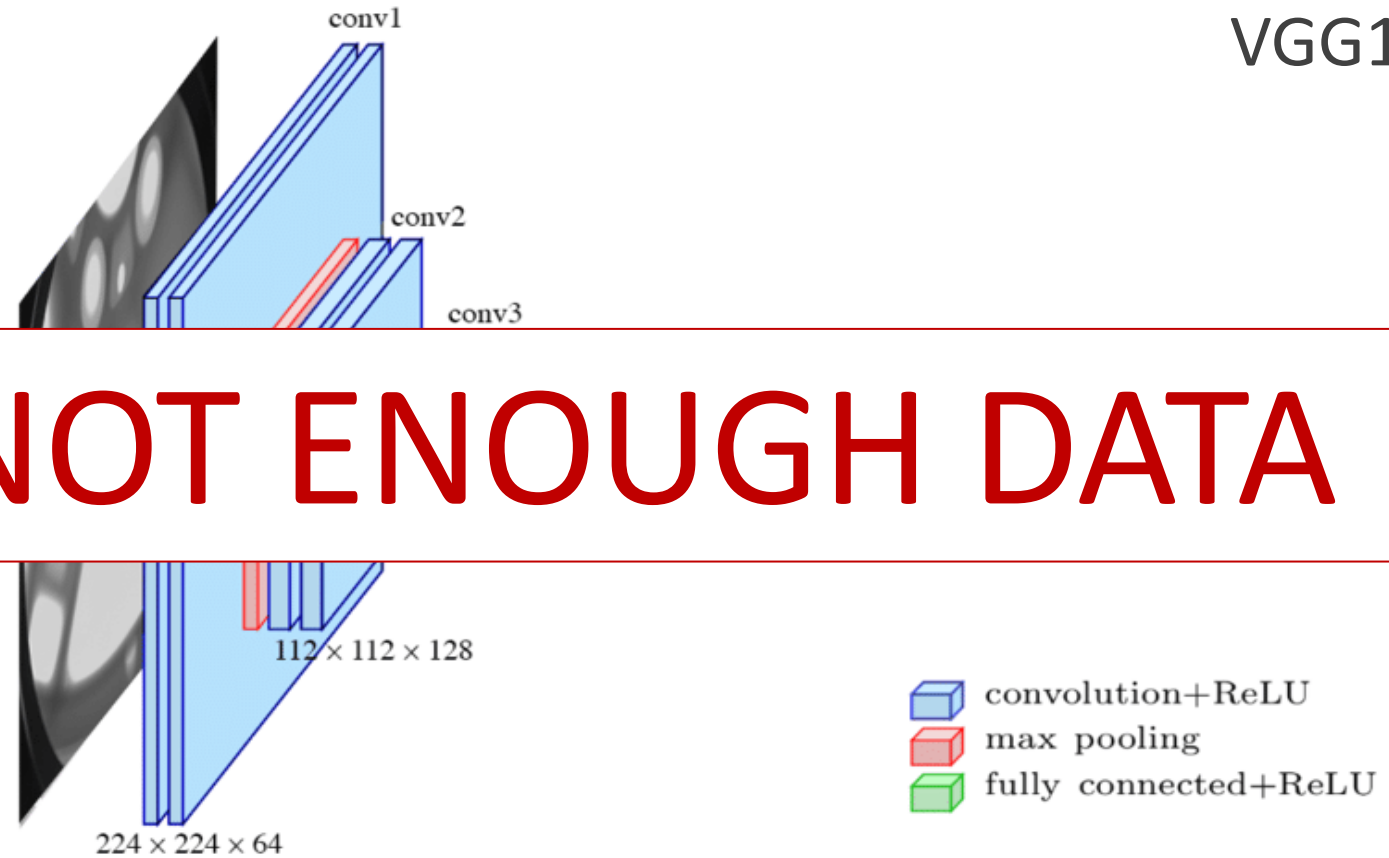
1. **Motivation**
2. The Basics idea for Transfer Learning
3. Formal Definition
4. Transfer Learning Strategies
5. Demo: How Transfer Learning Works in Practice

# Classify Rarest Plants



VGG16

**NOT ENOUGH DATA**

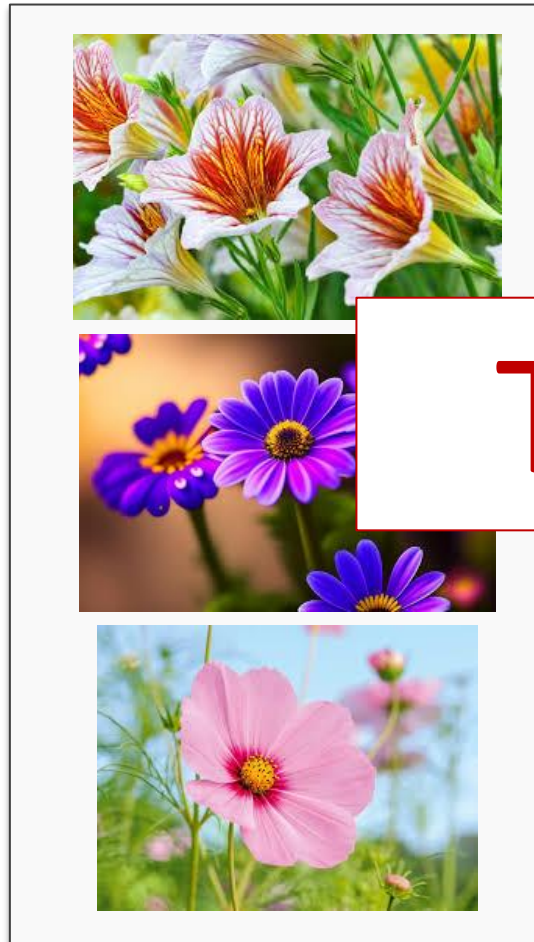


Number of parameters: 134,268,737

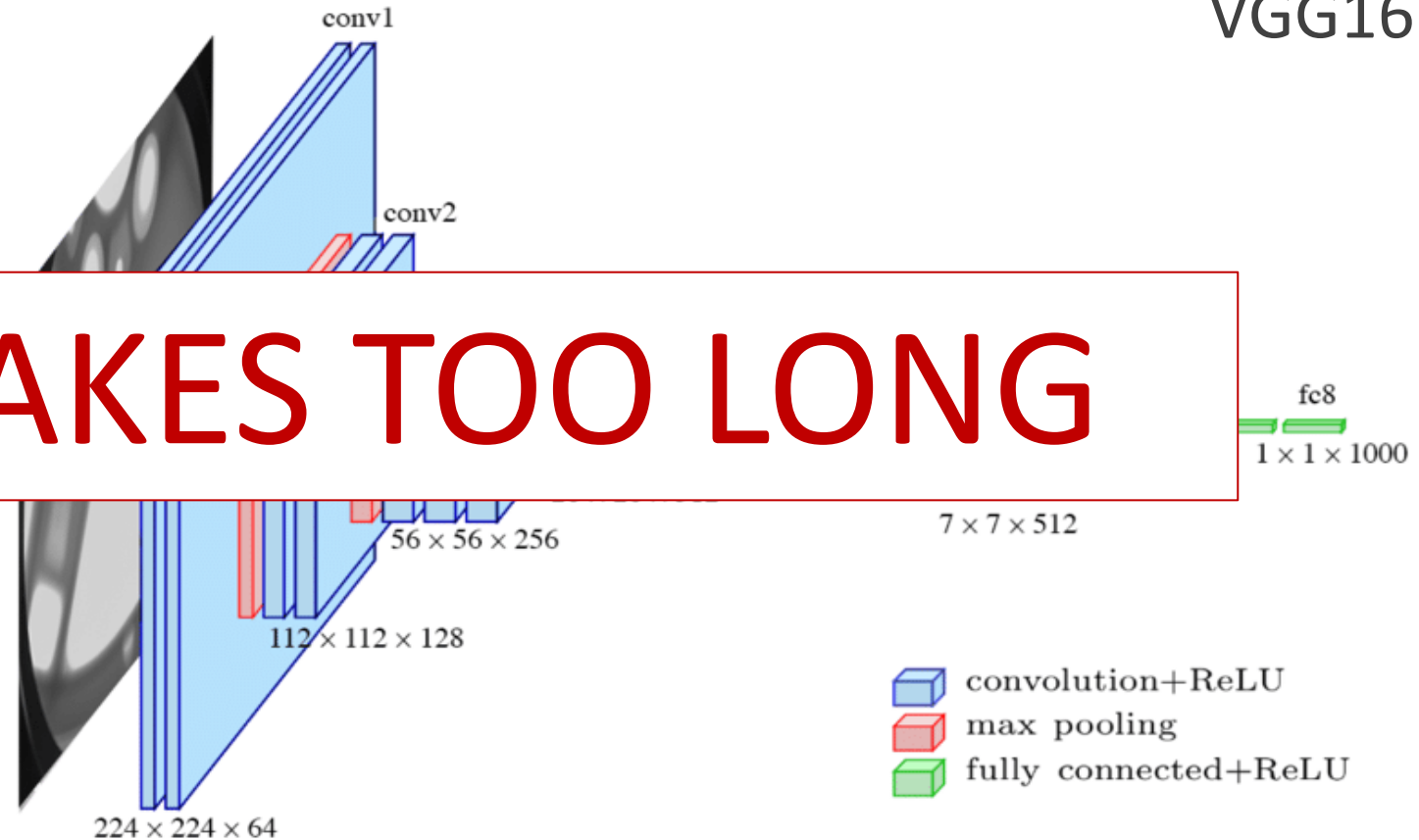
Data Set: Few hundred images

# Classify Cats, Dogs, Chinchillas etc

VGG16

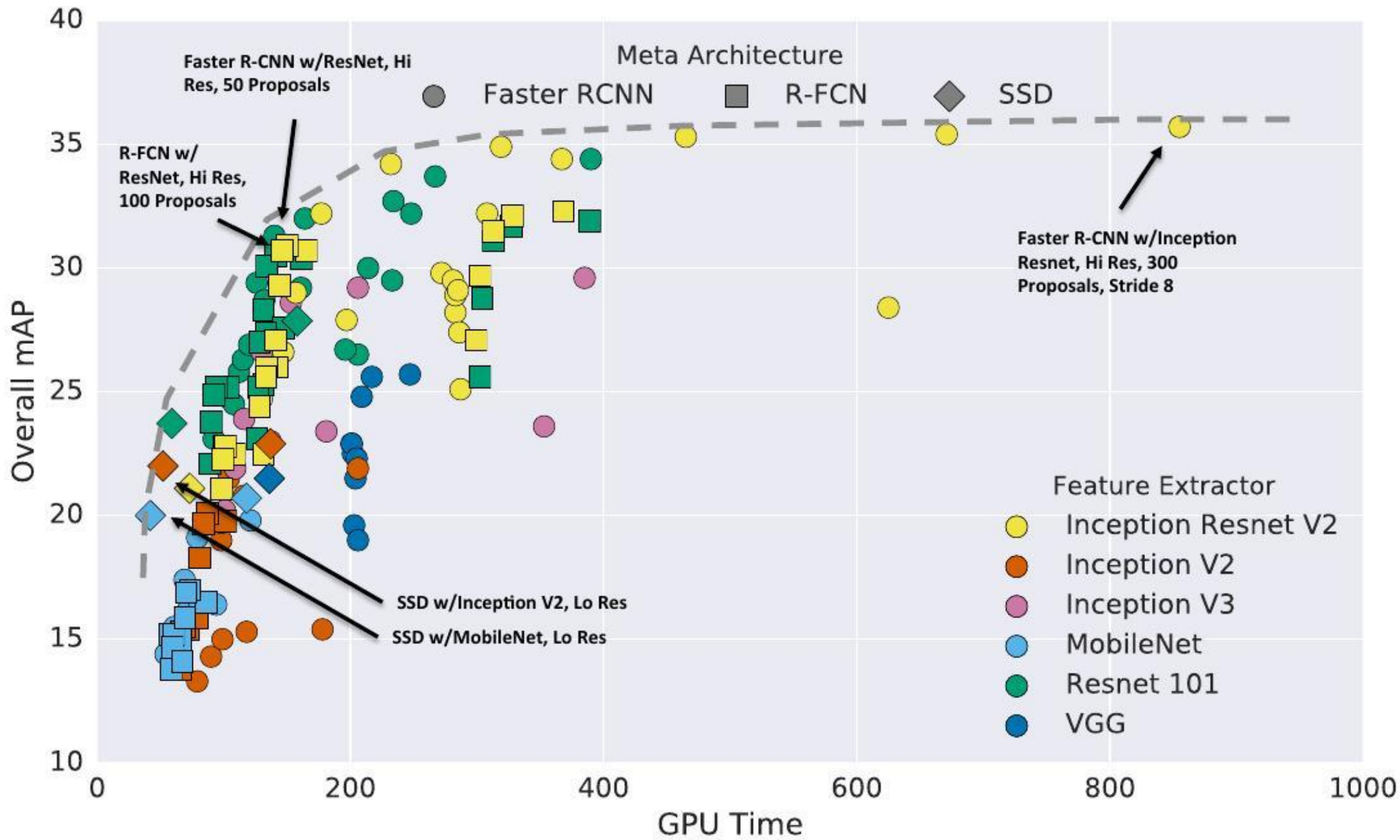


**TAKES TOO LONG**



Number of parameters: 134,268,737

Enough training data. ImageNet approximate 1.2M



# 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?



# Outline

---

1. Motivation
- 2. The Basics idea for Transfer Learning**
3. Formal Definition
4. Transfer Learning Strategies
5. Demo: How Transfer Learning Works in Practice



# 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.<sup>[1]</sup>

# Basic idea of Transfer Learning (cont)

---

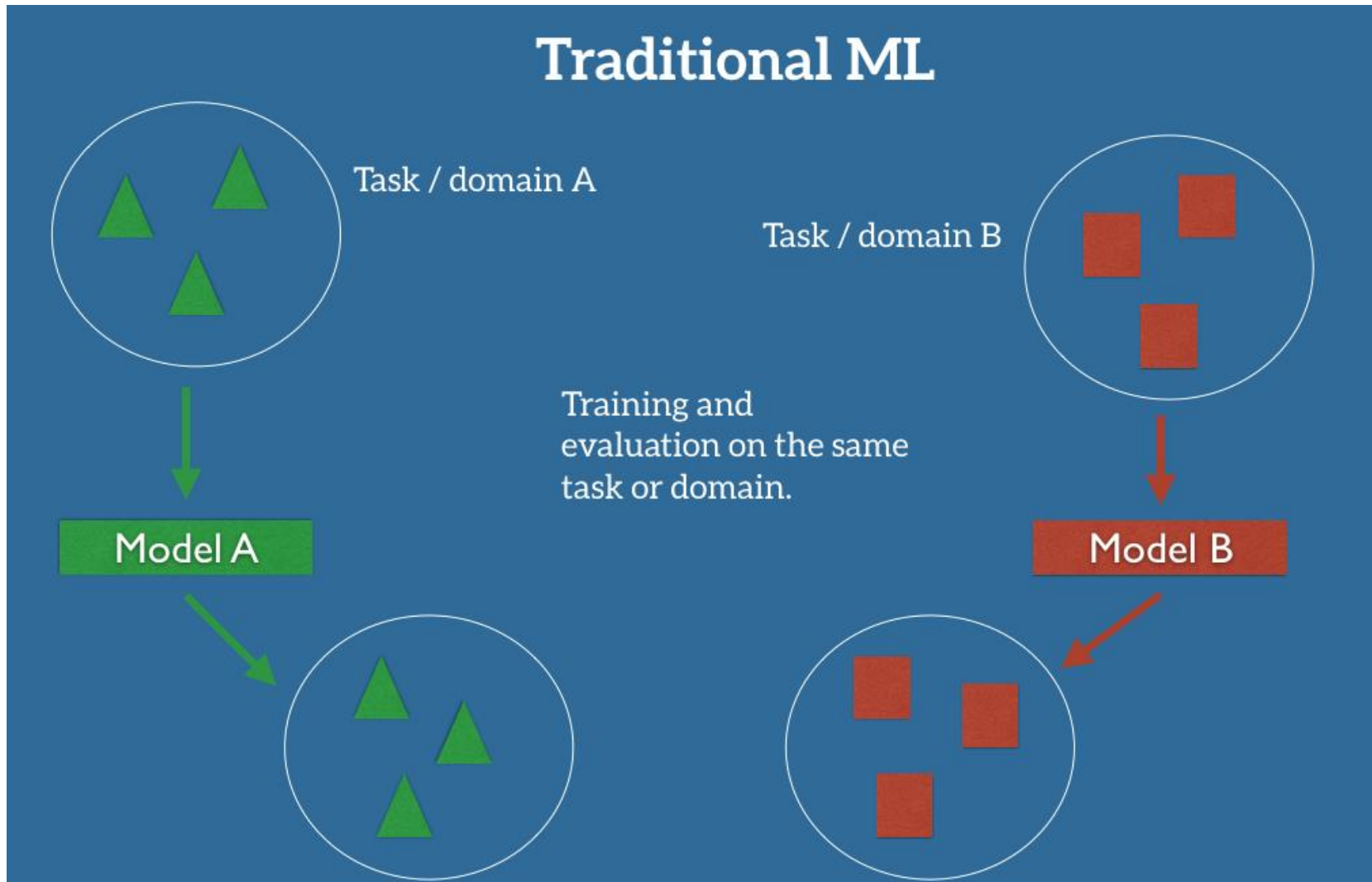
How do you make **an image classifier** that can be **trained** in a few minutes on a CPU with very little data?

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

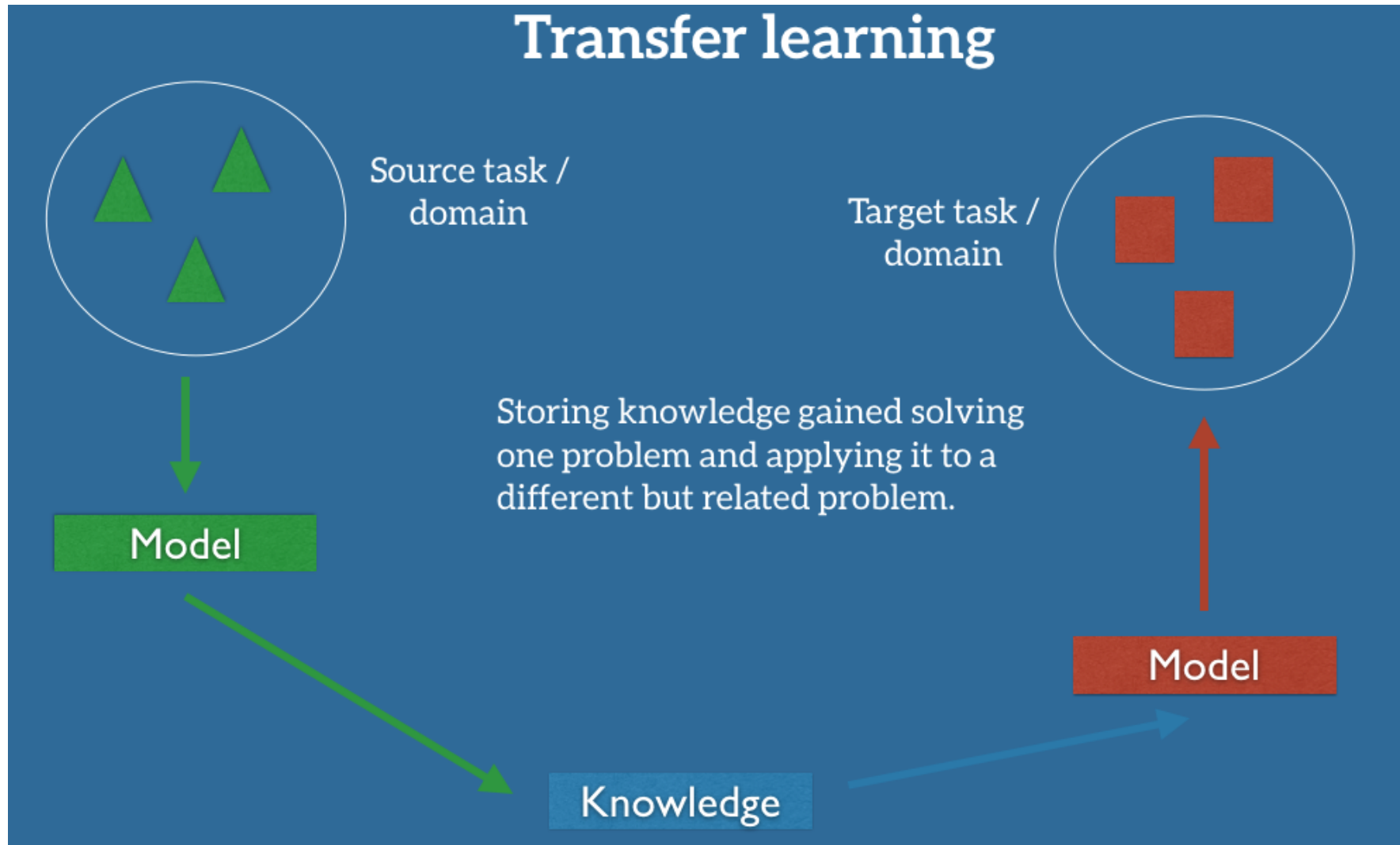
**Main Idea:** Earlier layers of a network learn low level features, which can be adapted to new domains by changing weights at later and fully-connected layers.

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

# Basic idea of Transfer Learning (cont.)

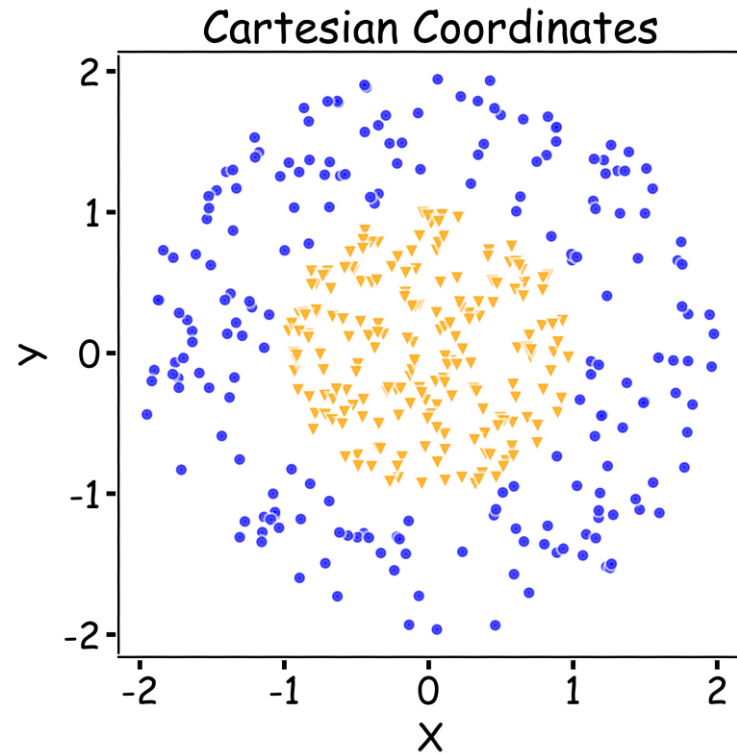


# Basic idea of Transfer Learning (cont.)



# Key Idea: Representation Learning

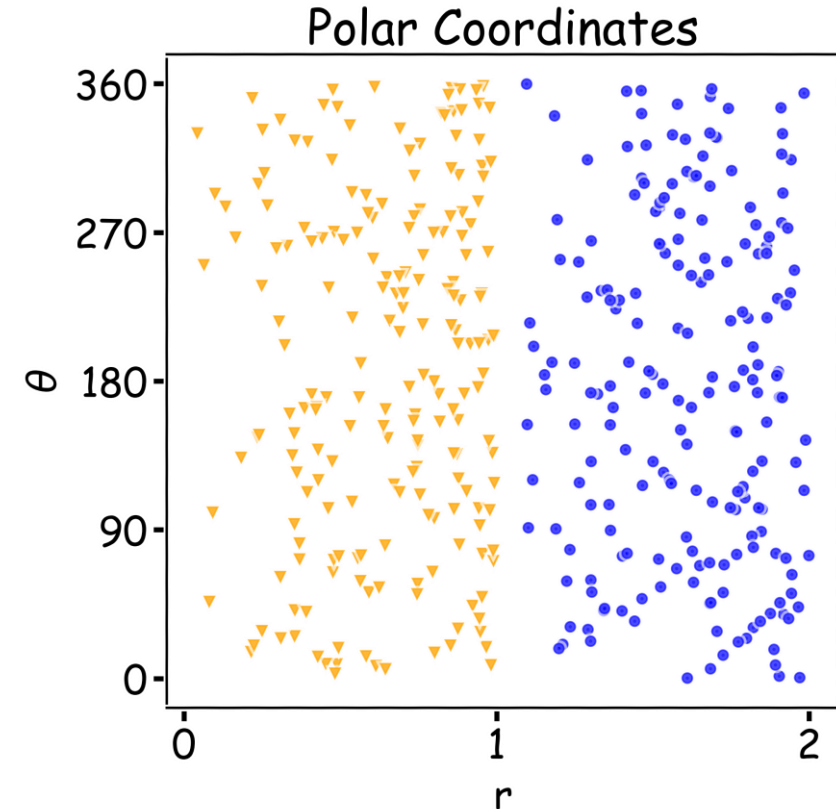
Relatively difficult task



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

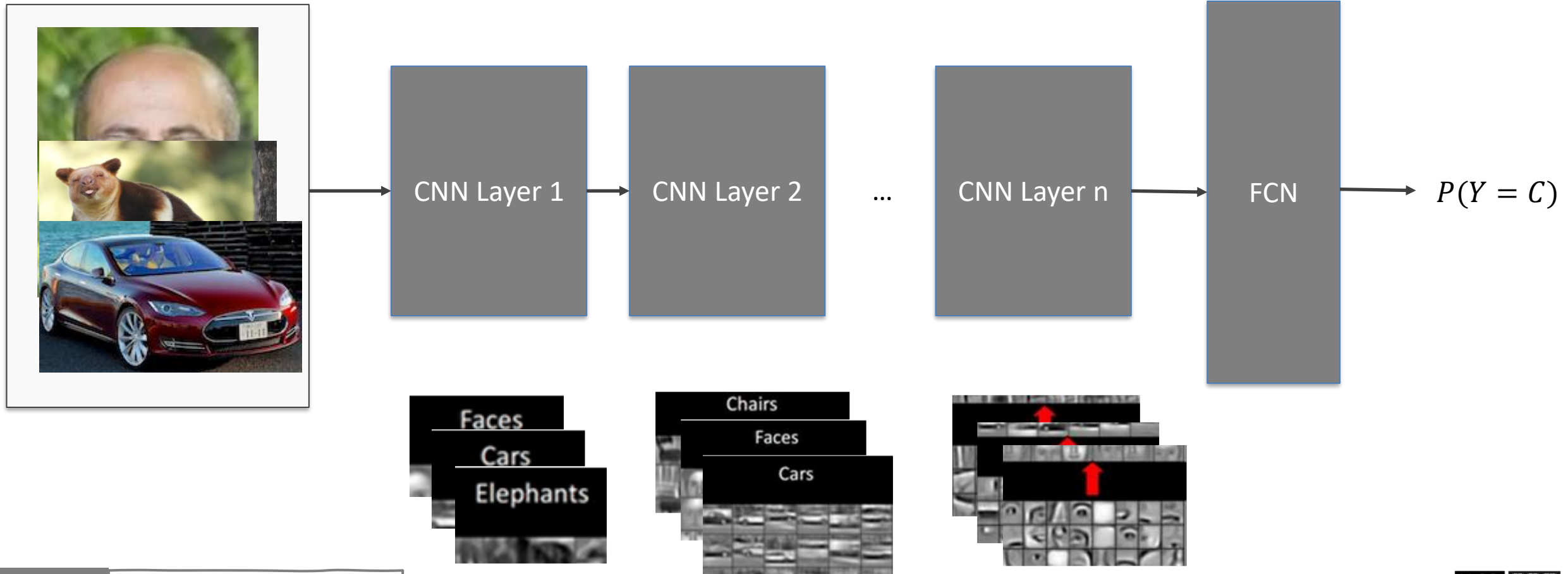


Easier task



# Representation Learning

Task: classify cars, people, animals and objects



# Basic idea of Transfer Learning (cont)

---

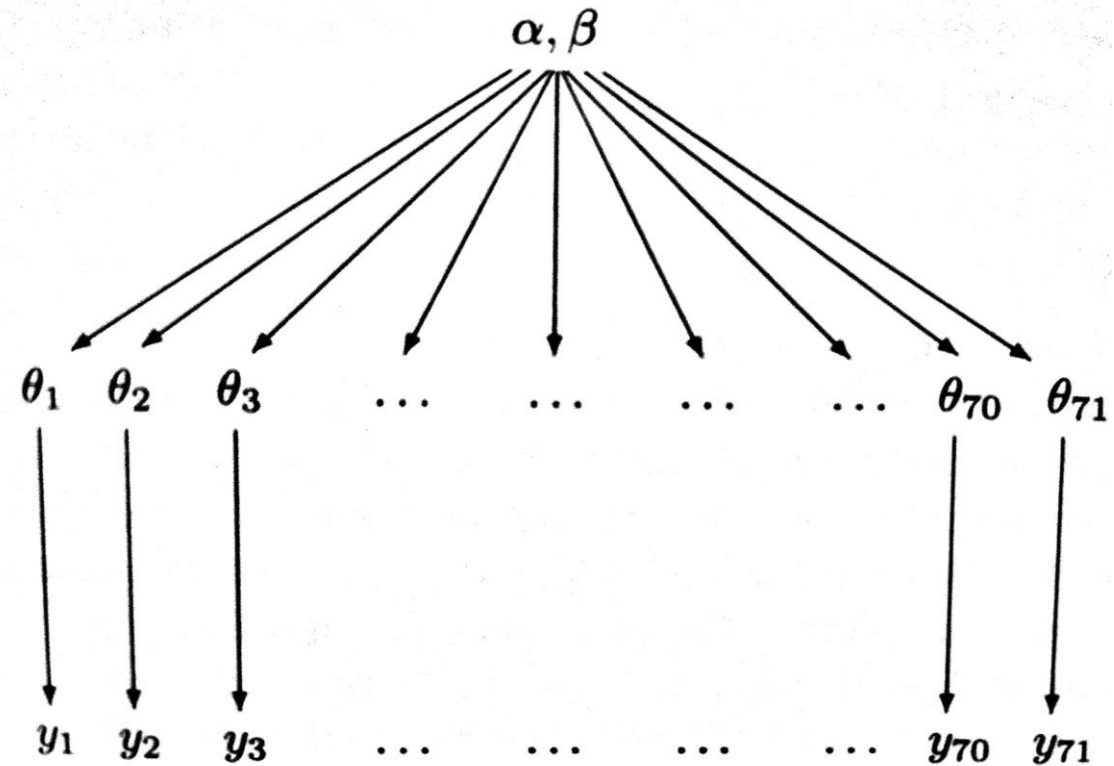
- Train on a big "**source**" data set, with a big model, on one downstream tasks (say classification). Do it once and save the parameters. This is called a **pre-trained model**.
- Use these parameters for other smaller "**target**" datasets, say, for classification on new images (possibly different **domain**, or training distribution), or for image segmentation on old images (new **task**), or new images (new task and new domain).
- Less helpful if you have a large target dataset with many labels.
- Will fail if source domain (where you trained big model) has nothing in common with target domain (that you want to train on smaller data set).

# Transfer Learning

## Not a new idea!

It has been there in the ML and stats literature for a while.

- An exemplar is hierarchical **glm models (Generalized Linear Models)** in stats, where information flows from higher data units to lower data units to the lower data.
- Neural networks **learn hierarchical representations** and thus are particularly suited to this kind of learning. Furthermore, since we learn representations, we can deal with domain adaptation/covariate shift.





# Applications of Transfer Learning

---

- Learn from simulations (self-driving cars)
- Domain adaptation: bikes -> bikes with backgrounds, bikes at night, etc
- Speech recognition for immigrants and minorities
- Cross-lingual adaptation for few shot learning of resource-poor languages (English->Nepalis, for example)

# Applications of Transfer Learning

---

- Create a classifier to distinguish dogs and cats.
- Use a convnet previously trained (expensive to learn)
  - e.g. Imagenet (1.4 M images and 1000 classes).
- In NLP, you might use a language model trained on Wikipedia and reddit, and then look at legal documents.

# Outline

---

1. Motivation
2. The Basics idea for Transfer Learning
- 3. Formal Definition**
4. Transfer Learning Strategies
5. Demo: How Transfer Learning Works in Practice

# A Formal Definition

A **Domain** consists of two components:  $D = \{\mathcal{X}, P(X)\}$

- Feature space:  $\mathcal{X}$
- Marginal Distribution:  $P(X); X = \{x_1, \dots, x_n\}, x_i \in \mathcal{X}$

For a given Domain, a **Task** is defined by two components:

$$T = \{\mathcal{Y}, P(Y|X)\} = \{\mathcal{Y}, \eta\}; Y = \{y_1, \dots, y_n\}, y_i \in \mathcal{Y}$$

- Label space:  $\mathcal{Y}$
- A predictive function  $\eta$ , learned from feature vector/label pairs  $(x_i, y_i), x_i \in \mathcal{X}, y_i \in \mathcal{Y}$ .
- For each feature vector in the domain,  $\eta$  predicts its corresponding label  $\eta(x_1) = y_1$ .

# Scenarios

---

Different features spaces among source and target

$$\chi_{source} \neq \chi_{target}$$

**Scenario:** document A - the source - is written in one language while document B - the target - is written in a different language

**Task:** can we use the weights learned training a model that distinguish phonemes on the source to distinguish those in the target written in another language? In NLP this method is called cross lingual adaptation.

# Scenarios

---

Different marginal probabilities among source and target

$$P_s(x) \neq P_t(X)$$

**Scenario:** Consider two different telescopes, in which one is equipped with a sensor with higher sensitivity than the other.

**Task:** can we use the weights learned training a model learned training a model on the source data that distinguish topics in another target document?

# Scenarios

---

Different labels among source and target

$$Y_{source} \neq Y_{target}$$

**Scenario:** farm animals versus wild forest animals or different level of classification, e.g. {dogs, cats} different breeds {Retriever, Bulldog, ..., Persian, Siamese}.

# Scenarios

---

Different conditional probabilities distribution among source and target task

$$P_s(Y|X) \neq P_t(Y|X)$$

**Scenario:** source and target are documents are unbalanced regarding their labels. Common scenario in practice, approachable with sampling techniques (e.g. under, over).



# Key Takeaways

---

During the process of transfer learning, the following three important questions must be answered:

- **What to transfer?** Identify which portion of knowledge is source-specific and what is common between the source and the target.
- **When to transfer?** Aim at utilizing transfer learning to improve target task performance/results and not degrade them. We need to be careful about when to transfer and when not to.
- **How to transfer?** Changes to existing algorithms and different techniques, which we will cover in later sections of this article.

# Transfer Learning Strategies

There are different transfer learning strategies and techniques, which can be applied **based on the domain, task at hand, and the availability of data:**

1. **Inductive Transfer learning:** the source and target have **same domains**, yet they have **different tasks**. The algorithms utilize the inductive biases of the source domain to help improve the target task.
  - **Example:** Suppose you have a model trained to classify different types of animals (e.g., dogs, cats, birds). You can use this pre-trained model to help develop a new model to classify specific breeds of dogs. Even though the tasks are different (general animal classification vs. specific breed classification), the knowledge from the first task can be transferred to **improve performance on the second task.**

# Transfer Learning Strategies

There are different transfer learning strategies and techniques, which can be applied **based on the domain, task at hand, and the availability of data:**

- 2. Unsupervised Transfer Learning:** the source and target have **same domains**, with a focus on **unsupervised tasks in the target** domain. The source and target domains are similar, but the tasks are different. In this scenario, labeled data is unavailable in either of the domains.
  - **Example:** Imagine you have a large dataset of text documents in English and you want to develop a model to cluster similar documents in a new language, say Spanish. Here, you can use the knowledge from the English text clustering model (e.g., understanding of language structure, common topics) to help with clustering Spanish texts. You don't have labeled data for the clustering task in Spanish, but the unsupervised learning from English can guide the process.

# Transfer Learning Strategies

There are different transfer learning strategies and techniques, which can be applied **based on the domain, task at hand, and the availability of data:**

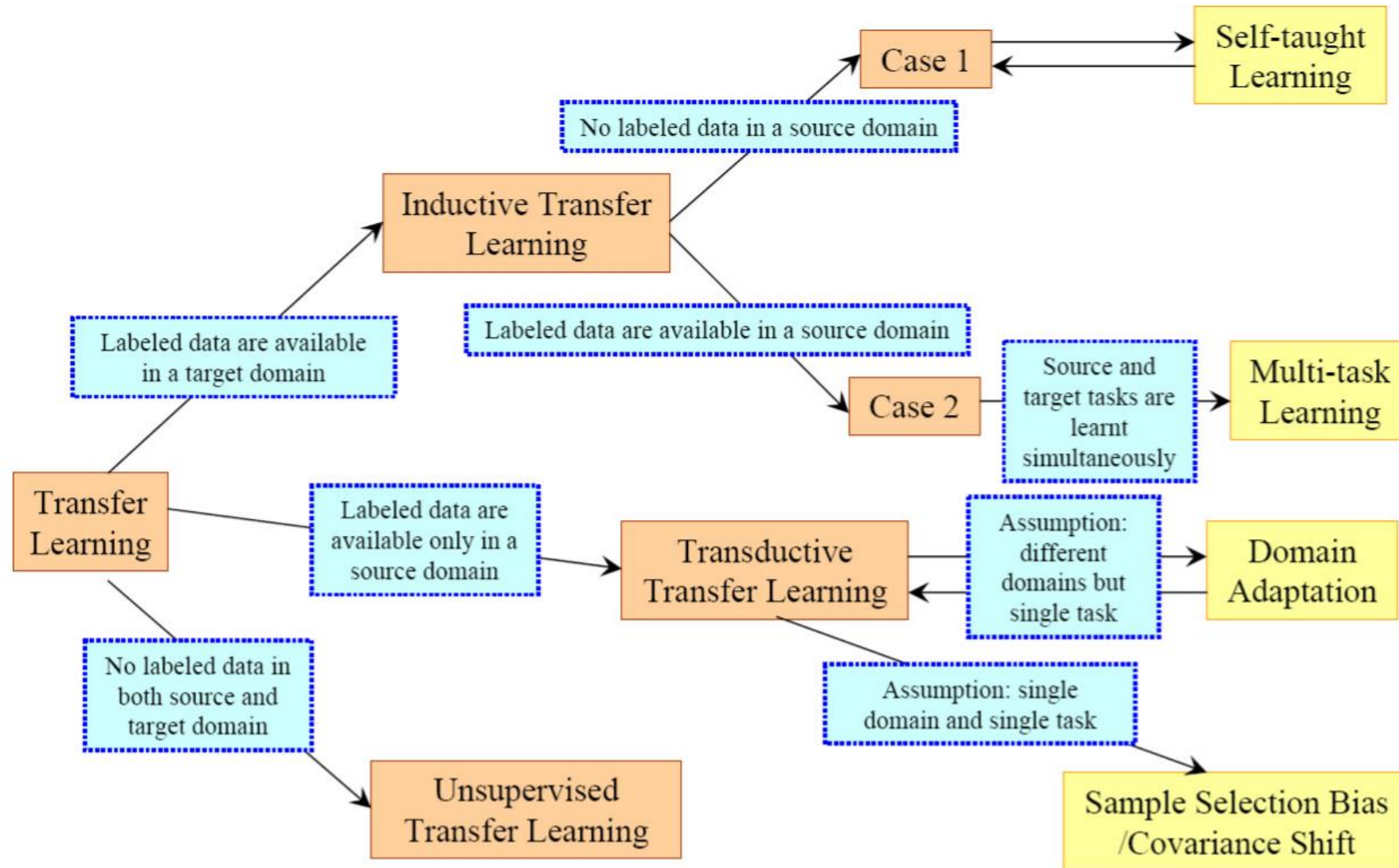
- 3. Transductive Transfer Learning:** In this scenario, there are similarities between the source and target tasks, but the corresponding domains are different. In this setting, the source domain has a lot of labeled data, while the target domain has none.
  - **Example:** Suppose you have a sentiment analysis model trained on English movie reviews (source domain). You want to apply this model to analyze sentiment in Spanish product reviews (target domain).

# Transfer Learning Strategies

The approaches for Transfer Learning can be defined in few categories:

- **Instance transfer:** Reusing knowledge from the source domain to the target task (ideal scenario). In most cases, the source domain data cannot be reused directly.
- **Feature-representation transfer:** This approach aims to minimize domain divergence and reduce error rates by identifying good feature representations that can be utilized from the source to target domains.
- **Parameter transfer:** This approach works on the assumption that the models for related tasks share some parameters or prior distribution of hyperparameters.
- **Relational-knowledge transfer** attempts to handle non-IID data, such as data that is not independent and identically distributed.

# Transfer Learning Strategies



# Outline

---

1. Motivation
2. The Basics idea for Transfer Learning
3. Formal Definition
- 4. Transfer Learning Strategies**
5. Demo: How Transfer Learning Works in Practice

# Transfer Learning for Deep Learning

## What people think

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

## What people can do, instead

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



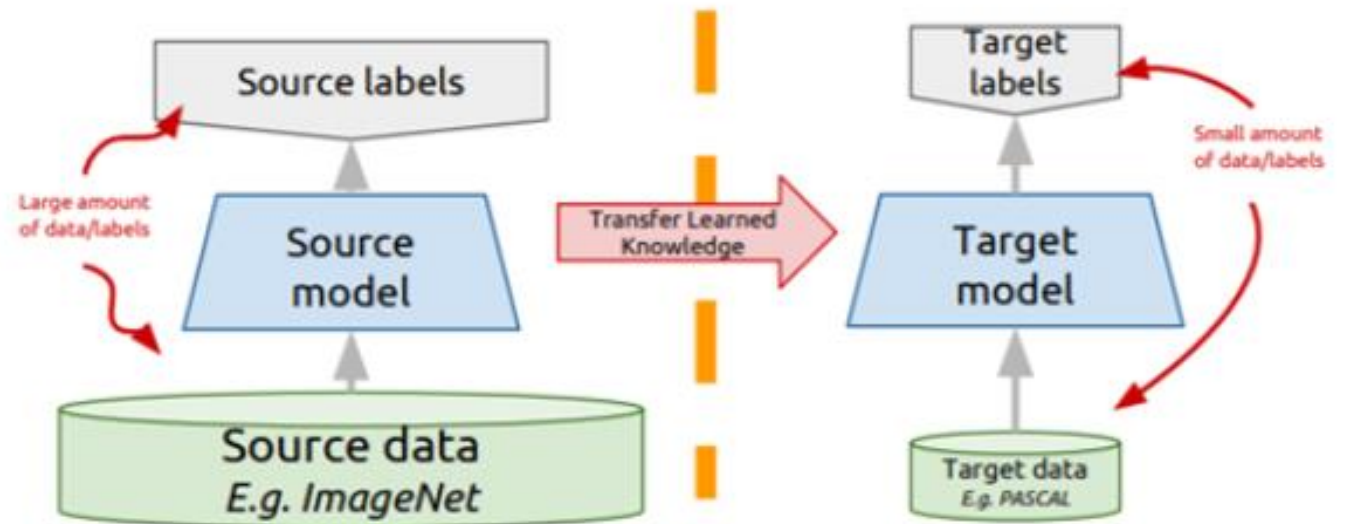
# Transfer Learning for Deep Learning

**Instead of training a network from scratch:**

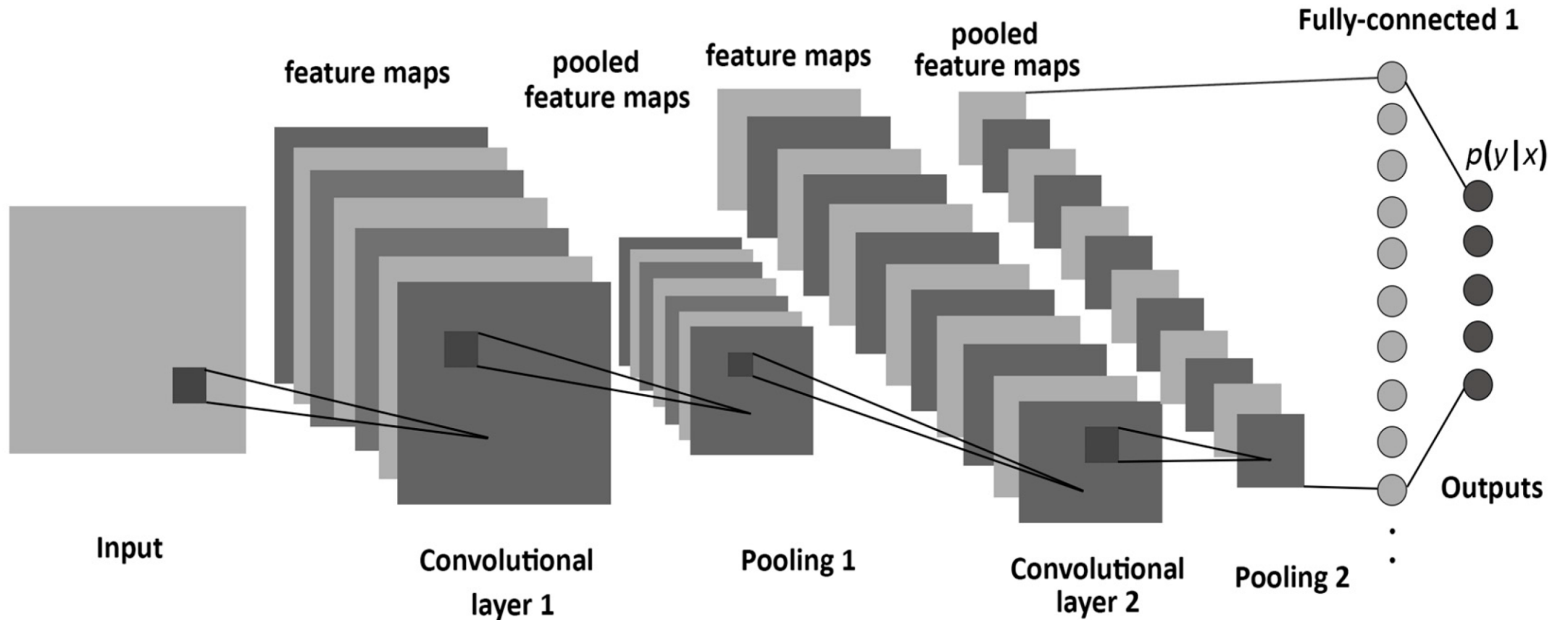
- Take a network trained on a different domain for a different **source task**
- Adapt it for your domain and your **target task**

## Variations

- Same domain, different task.
- Different domain, same task.



# Extremely short review of CNN



# Example

---

- Let C be a CNN with the following disposition:
  - Input: 32x32x3 images
  - Conv1: 8 3x3 filters, stride 1, padding=same
  - Conv2: 16 5x5 filters, stride 2, padding=same
  - Flatten layer
  - Dense1: 512 nodes
  - Dense2: 4 nodes

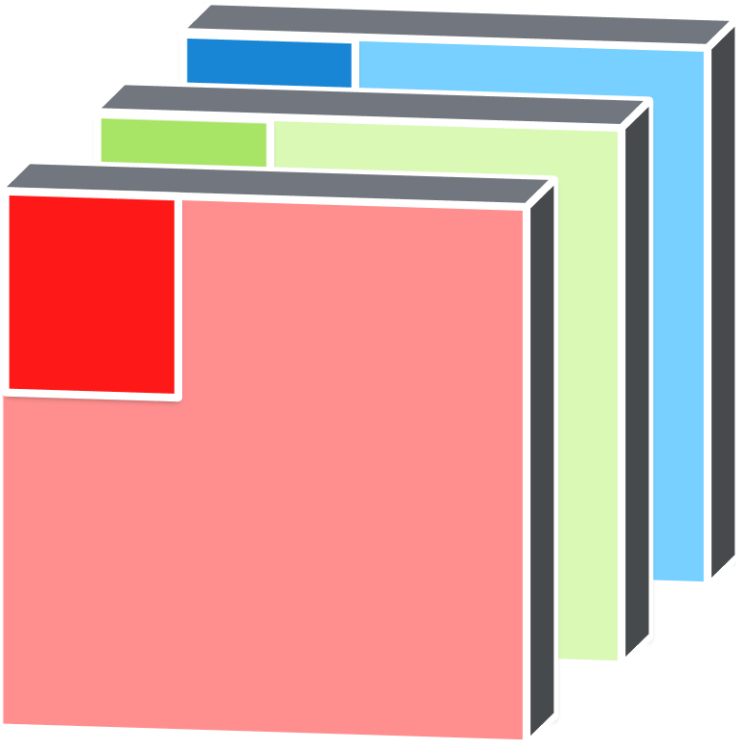
- How many parameters does this network have?

$$(8 \times 3 \times 3 \times 3 + 8) + (16 \times 5 \times 5 \times 8 + 16) + (16 \times 16 \times 16 \times 512 + 512) + (512 \times 4 + 4)$$

Conv1Conv2Dense1Dense2

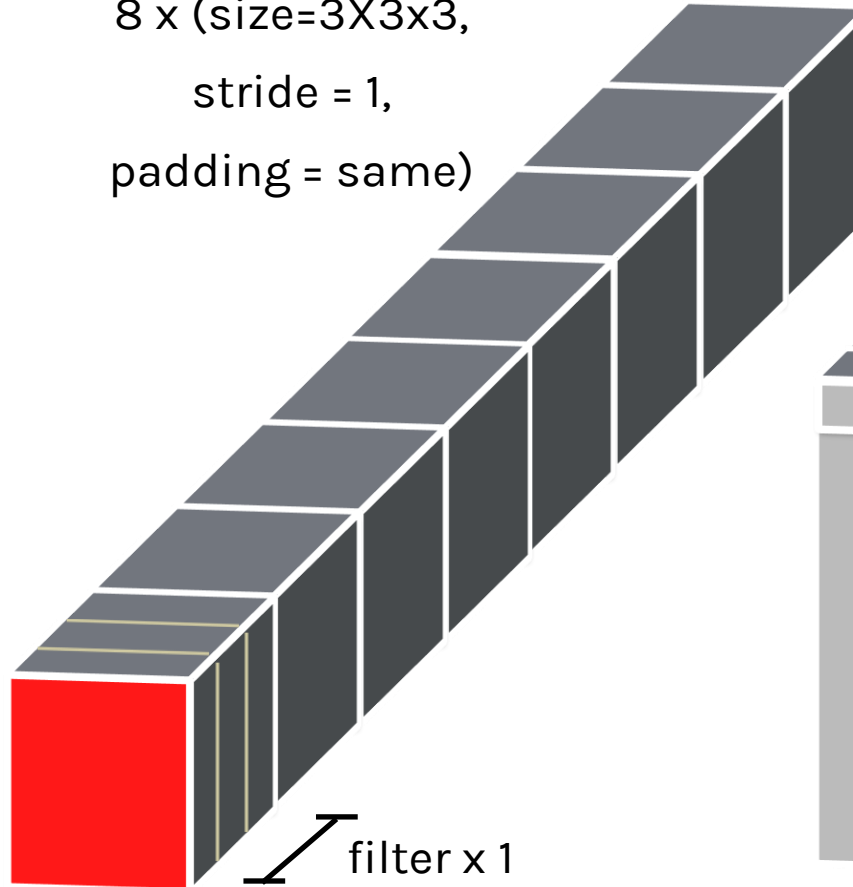
## Input

(size=32X32,  
channels=3)



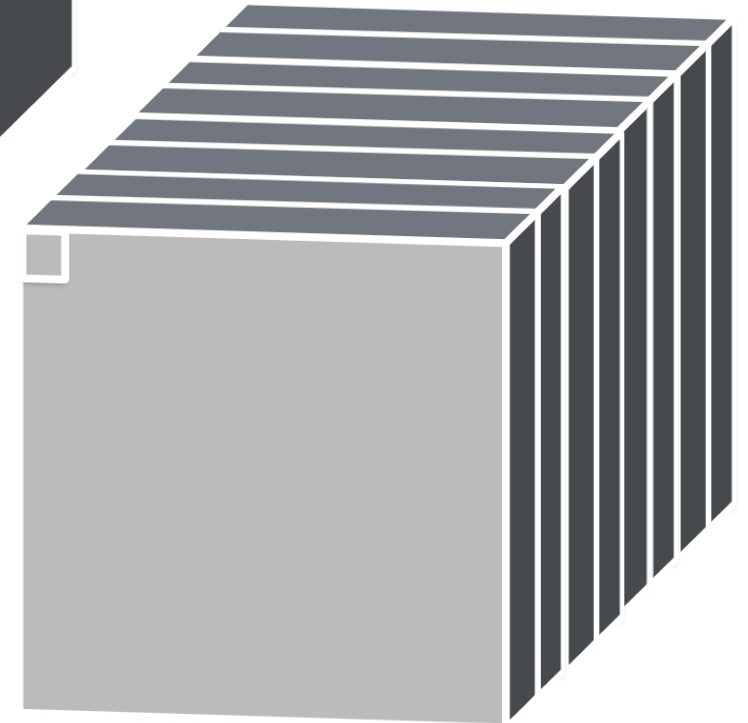
## Filter

8 x (size=3X3x3,  
stride = 1,  
padding = same)



## Output

(size=32X32,  
channels = 8)



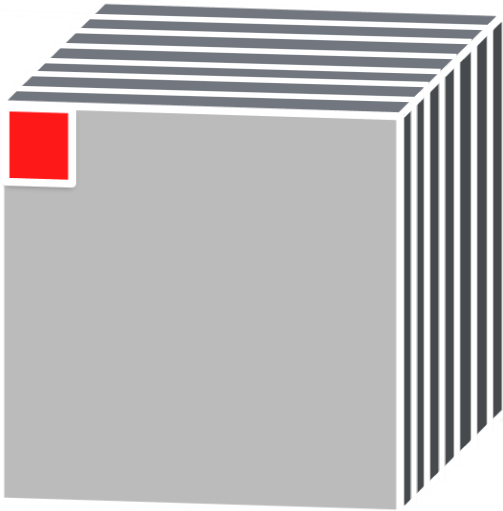
**How many parameters does the layer have if I want to use 8 filters?**

$n\_filters \times filter\_volume + biases = \text{total number of params}$

$$8 \times (3 \times 3 \times 3) + 8 = 224$$

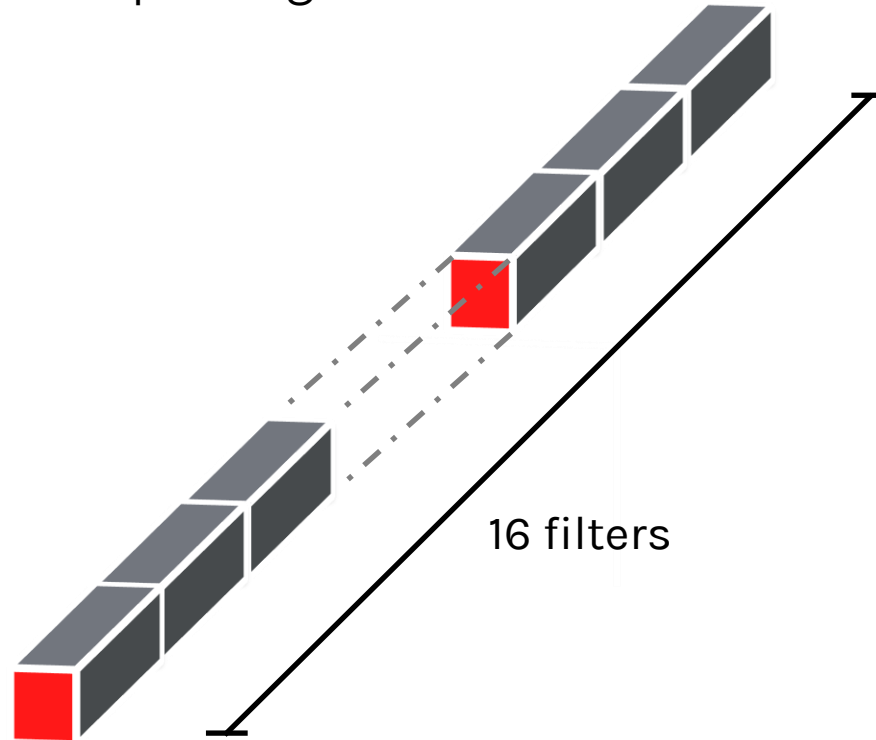
## Input

(size=32X32,  
channels=8)



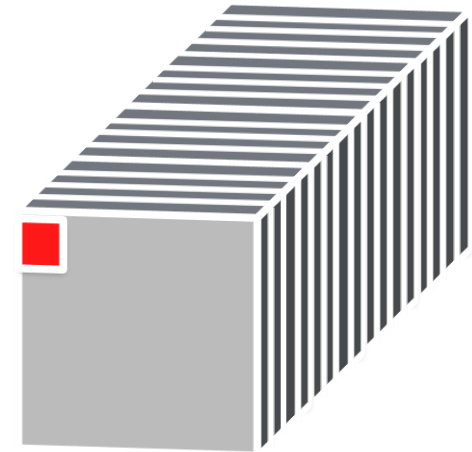
## Filter

16 x (size=5X5X8,  
stride = 2,  
padding = same)



## Output

(size=16X16,  
channels=16)



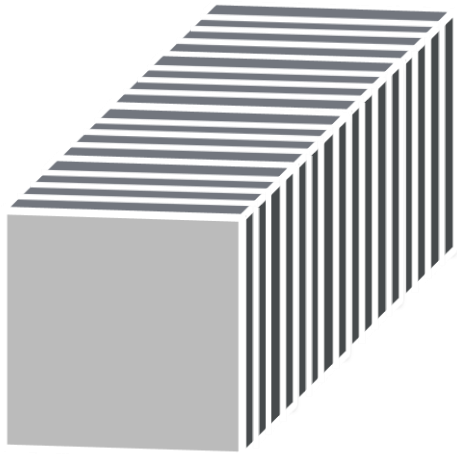
**How many parameters does the layer have if I want to use 16 filters?**

$n\_filters \times filter\_volume + biases = \text{total number of params}$

$$16 \times (5 \times 5 \times 8) + 16 = 3216$$

## Input

(size=16X16,  
channels=16)



## Flatten

(size= 16X16X16)



## Fully Connected

(n\_nodes=512)



## Fully Connected

(n\_nodes=4)



### How many parameters ... ?

input x FC1\_nodes + FC2\_nodes = total number of params

$$(16 \times 16 \times 16) \times 512 + 512 + 512 \times 4 + 4 = 2,099,716$$

# Outline

---

1. Communications
2. Motivation
3. The Basics idea for Transfer Learning
4. Formal Definition
- 5. Transfer Learning Strategies**
6. Demo: How Transfer Learning Works in Practice

# Representation Extraction

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

- Keep (frozen) convolutional **base** from big model.
- Generally throw away **head** FC layers since these have no notion of space, and convolutional base is more generic.
- Since there are both dogs and cats in *ImageNet* you could get **away** with using the head FC layers as well. But by throwing it away you can learn more from other dog/cat images.



$$P_s(Y|X) \neq P_t(Y|X)$$

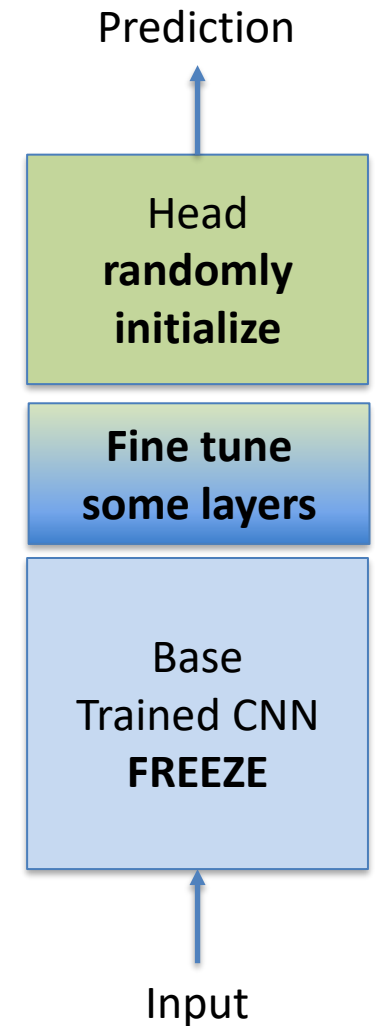


# Fine-tuning

---

- Up to now we have frozen the entire convolutional base.
- Remember that earlier layers learn highly generic feature maps (edges, colors, textures).
- Later layers learn abstract concepts (dog's ear).
- To particularize the model to our task, its often worth tuning the later layers as well.
- But we must be very careful not to have big gradient updates.

$$P_s(x) \neq P_t(X)$$



# Procedure for Fine-tuning

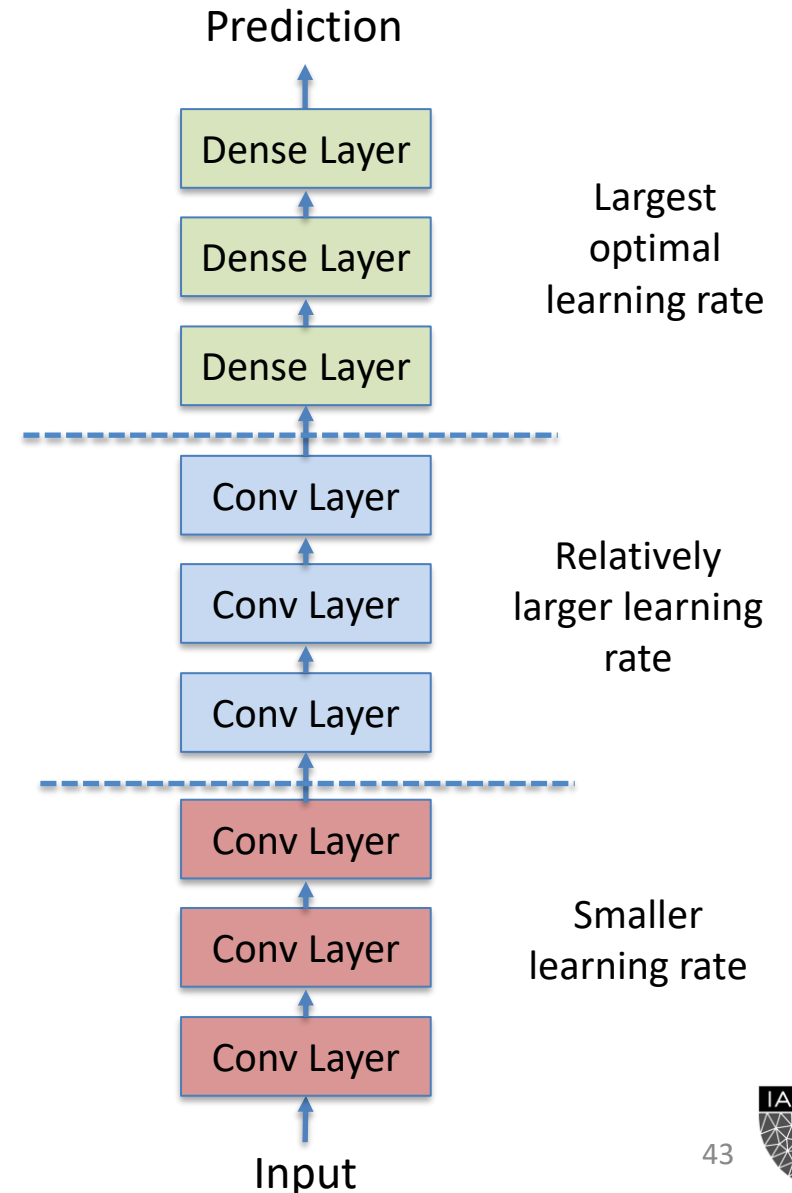
---

1. Freeze the convolutional base.
2. First train the fully connected head you added, keeping the convolutional base fixed.
3. Unfreeze some "later" layers in the base net and now train the base net and FC net together.

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**.

# 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.



# Outline

---

1. Communications
2. Motivation
3. The Basics idea for Transfer Learning
4. Formal Definition
5. Transfer Learning Strategies
- 6. Demo: How Transfer Learning Works in Practice**  
**TF Classification**

THANK YOU

**AC295**

**Advanced Practical Data Science**  
Pavlos Protopapas