[SKT AI Course: Deep Learning Basics]

Gentle Introduction to Transfer Learning

Heung-Il Suk

hisuk@korea.ac.kr

http://www.ku-milab.org

Department of Brain and Cognitive Engineering,
Korea University

September 21, 2017

Contents

1. Transfer Learning In a Nutshell
2. Transfer Learning with Deep Models

Transfer Learning In a Nutshell

Pan and Yang, "A Survey on Transfer Learning," IEEE Transactions on Knowledge and Data Engineering, 2010

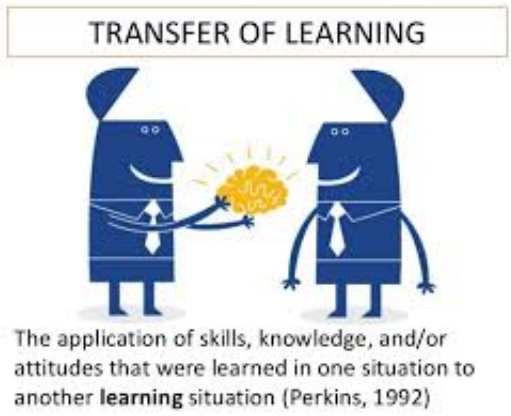
D. Gupta, "Transfer learning & The art of using Pre-trained Models in Deep Learning," Analytics Vidhya, 2017

What is Transfer Learning (TL)?

Transfer Learning

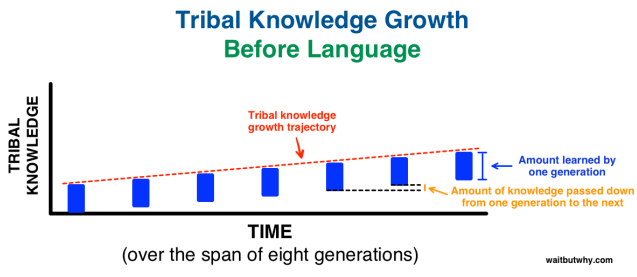
The ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks (in new domains)

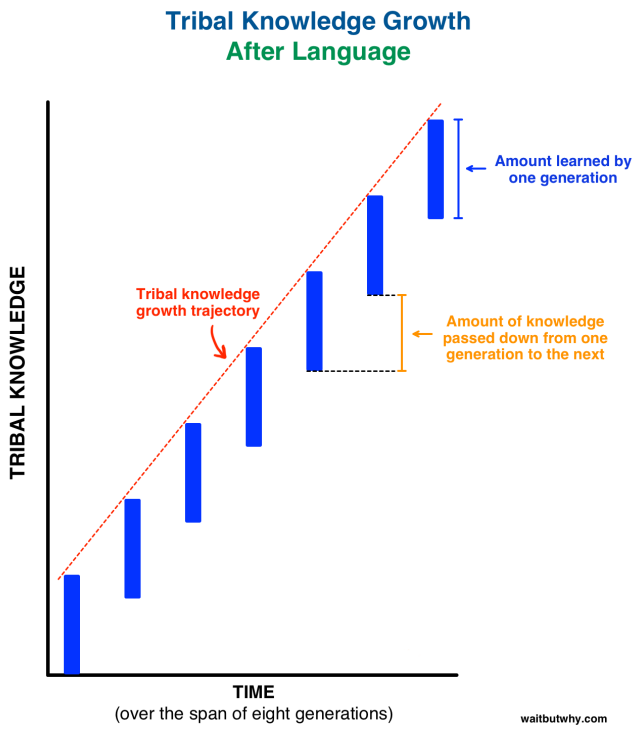
- Motivated by human learning
- People can often transfer knowledge learned previously to novel situations.
  - mathematics → computer science
  - table tennis → tennis



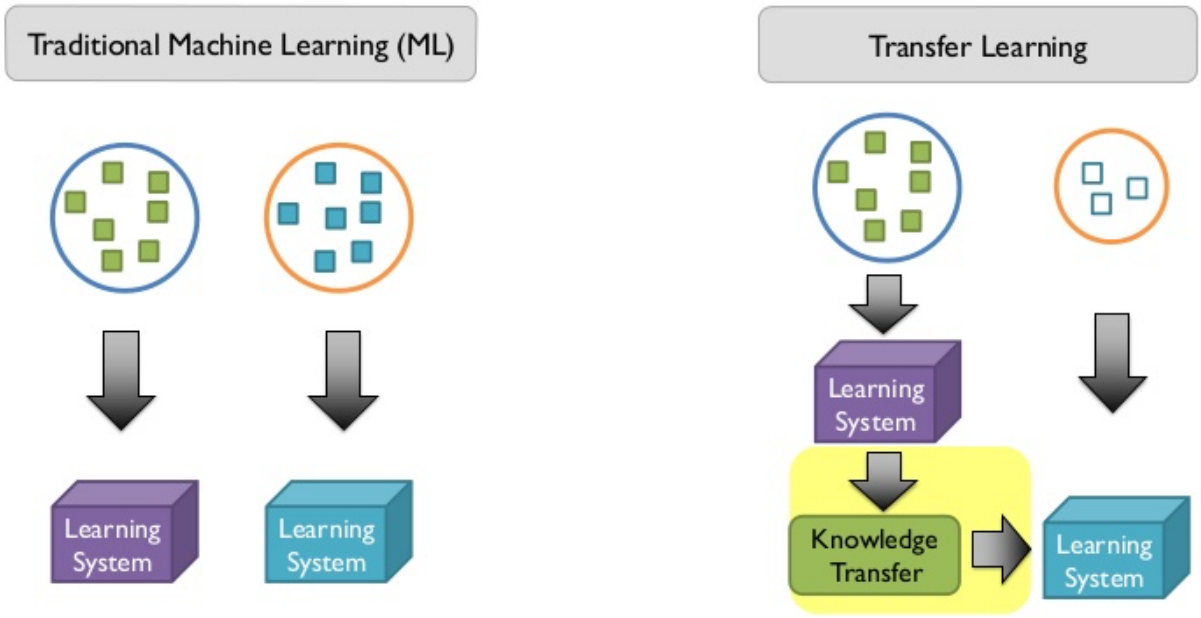
Teacher-student analogy

- accumulated information, the lectures that students get is a concise and brief overview of the topic
- "transfer" of information from the learned to a novice





Traditional ML vs. Transfer Learning



Domain $\mathcal{D} \equiv \{\mathcal{X}, P(X)\}$

$\mathcal{X}$ and $P(X)$ denote, respectively, a feature space and a marginal probability distribution of $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathcal{X}$.

- In general, if two domains are different, then they may have different feature spaces or different marginal distributions.

Task $\mathcal{T} \equiv \{\mathcal{Y}, f(\cdot)\}$

$\mathcal{Y}$ and $f(\cdot)$ denote, respectively, a label space and an objective predictive function, which is not observed but can be learned from the training data, *i.e.*, $\{\mathbf{x}_i \in X, y_i \in \mathcal{Y}\}$.

- In general, if two tasks are different, then they may have different label spaces or different conditional distributions $P(Y|X)$, where $Y = \{y_1, \ldots, y_n)$ and $y_i \in \mathcal{Y}$.

For simplicity, consider at most two domains and two tasks

- **Source domain**

$$P(X_S), \text{ where } X_S = \left\{\mathbf{x}_{S_1}, \mathbf{x}_{S_2}, \ldots, \mathbf{x}_{S_{n_S}}\right\} \in \mathcal{X}_S$$

- **Task in the source domain**

$$P(Y_S|X_S), \text{ where } Y_S = \left\{y_{S_1}, y_{S_2}, \ldots, y_{S_{n_S}}\right\} \text{ and } y_{S_i} \in \mathcal{T}_S$$

- **Target domain**

$$P(X_T), \text{ where } X_T = \left\{\mathbf{x}_{T_1}, \mathbf{x}_{T_2}, \ldots, \mathbf{x}_{T_{n_T}}\right\} \in \mathcal{X}_T$$

- **Task in the target domain**

$$P(Y_T|X_T), \text{ where } Y_T = \left\{y_{T_1}, y_{T_2}, \ldots, y_{T_{n_T}}\right\} \text{ and } y_{T_i} \in \mathcal{T}_T$$

Transfer Learning

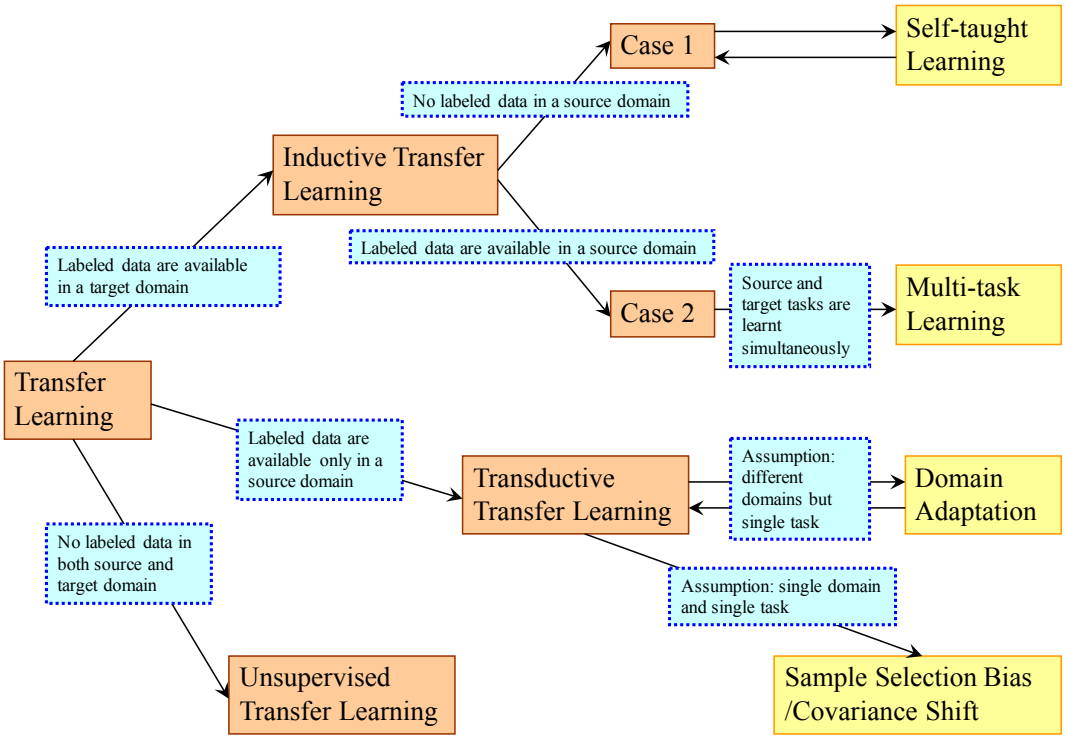
Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims *to help improve the learning of the target predictive function* $f(\cdot)_T$ *in* $\mathcal{D}_T$ *using the knowledge in* $\mathcal{D}_S$ *and* $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

- "What to transfer": which part of knowledge can be transferred across domains or tasks
- How to extract knowledge learned from related domains to help learning in a target domain with a few labeled data?
- How to extract knowledge learned from related domains to speed up learning in a target domain?
- How to avoid negative transfer is also an important issue

Different Settings of Transfer

Transfer Learning Settings	Related Areas	Source Domain Labels	Target Domain Labels	Tasks
Inductive Transfer Learning	Multi-task Learning	Available	Available	Regression, Classification
	Self-taught Learning	Unavailable	Available	Regression, Classification
Transductive Transfer Learning	Domain Adaptation, Sample Selection Bias, Co-variate Shift	Available	Unavailable	Regression, Classification
Unsupervised Transfer Learning		Unavailable	Unavailable	Clustering, Dimensionality Reduction

- Inductive TL: $\mathcal{T}_S \neq \mathcal{T}_T$
  - achieving high performance in the target task only (inductive TL) vs. both source and target task simultaneously (multi-task learning)
  - When $\mathcal{Y}_S \neq \mathcal{Y}_T$, the side information of the source domain cannot be used directly, similar to the inductive TL where the labeled data in the source domain unavailable
- Transductive TL: $\mathcal{T}_S = \mathcal{T}_T$, $\mathcal{D}_S \neq \mathcal{D}_T$
  - $\mathcal{X}_S \neq \mathcal{X}_T$
  - $\mathcal{X}_S = \mathcal{X}_T$, but $P(X_S) \neq P(X_T)$: *c.f.*, domain adaptation, sample selection bias, covariate shift
- Unsupervised TL: $\mathcal{T}_S \neq \mathcal{T}_T$, but related to each other
  - focus on solving unsupervised learning tasks in the target domain, such as clustering, dimensionality reduction, density estimation



Approaches to Transfer Learning

- **Instance-transfer**
  - To re-weight some labelled data in a source domain for use in the target domain
- **Feature-representation-transfer**
  - Find a "good" feature representation that reduces difference between domains
- **Parameter-transfer**
  - Discover shared parameters or priors of models between domains
- **Relational-knowledge-transfer**
  - Build mapping of relational knowledge between domains

	Inductive Transfer Learning	Transductive Transfer Learning	Unsupervised Transfer Learning
Instance-transfer	√	√	
Feature-representation-transfer	√	√	√
Parameter-transfer	√		
Relational-knowledge-transfer	√		

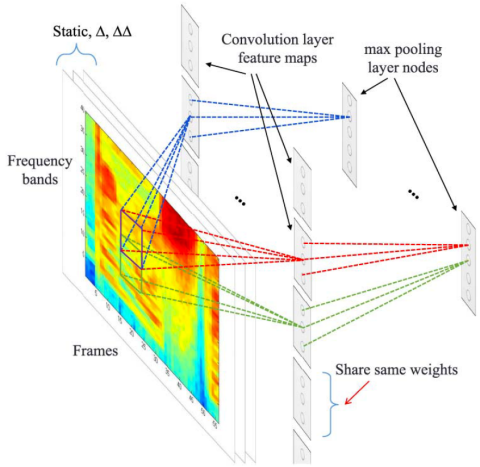
Transfer Learning with Deep Models

- A neural network is trained on a dataset.
- A trained network gains knowledge from this data, which is compiled as "**weights**" of the network.
- These weights can be extracted and then transferred to any other neural network.
- Instead of training the other neural network from scratch, we "**transfer**" the learned features.
- Transfer learning by passing on weights is equivalent of language used to disseminate knowledge over generations in human evolution.

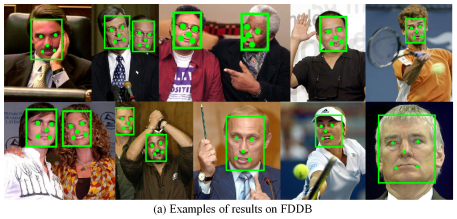
Pretrained models

- Common to see people release their final ConvNet checkpoints for the benefit of others who can use the networks for fine-tuning.
- *e.g.*, Model Zoo in Caffe library: `https://github.com/BVLC/caffe/wiki/Model-Zoo`
- By using pretrained models which have been previously trained on large datasets, we can directly use the weights and architecture obtained and apply the learning on our problem statement.
- Instead of building a model from scratch to solve a similar problem, you use the model trained on other problem as a starting point.

- A pretrained model may not be 100% accurate in your application, but it saves huge efforts required to re-invent the wheel.
- If the problem statement we have at hand is very different from the one on which the pretrained model was trained - the prediction we would get would be very inaccurate



CNN for speech recognition



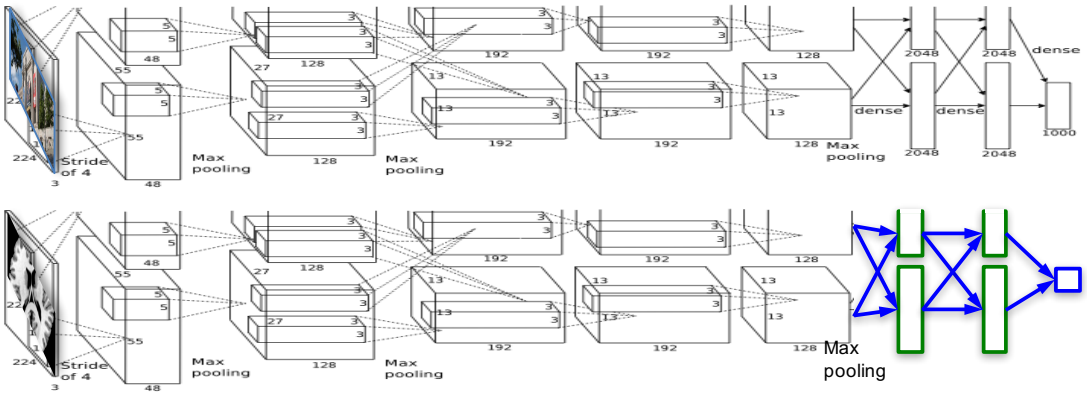
(a) Examples of results on FDDB

(b) Examples of results on WIDER FACE

A task of face detection

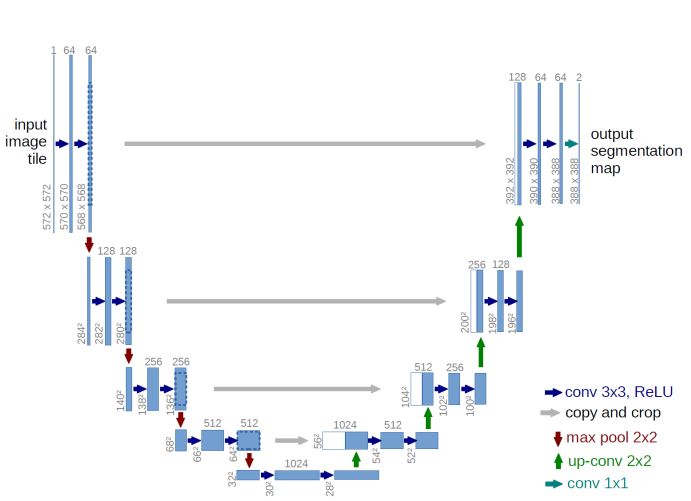
Feature extraction

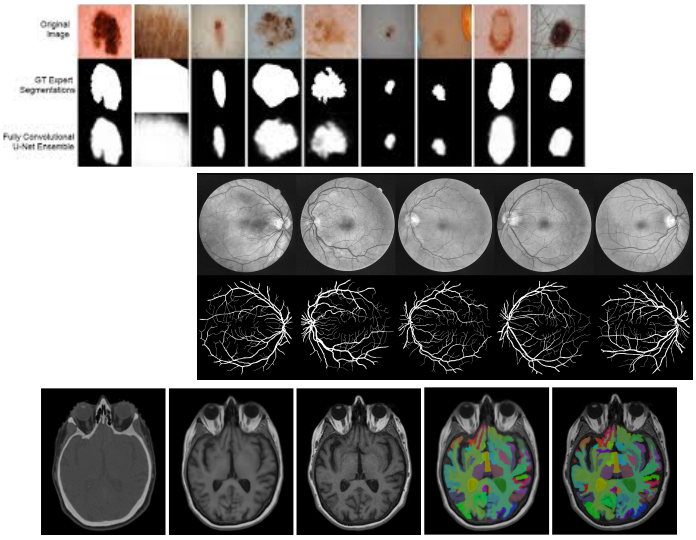
- We can use a pretrained model as a feature extraction mechanism.
- What we can do is that we can remove the output layer (the one which gives the probabilities for being in each of the 1000 classes) and then use the entire network as a fixed feature extractor for the new data set.



Use the Architecture of the pretrained model

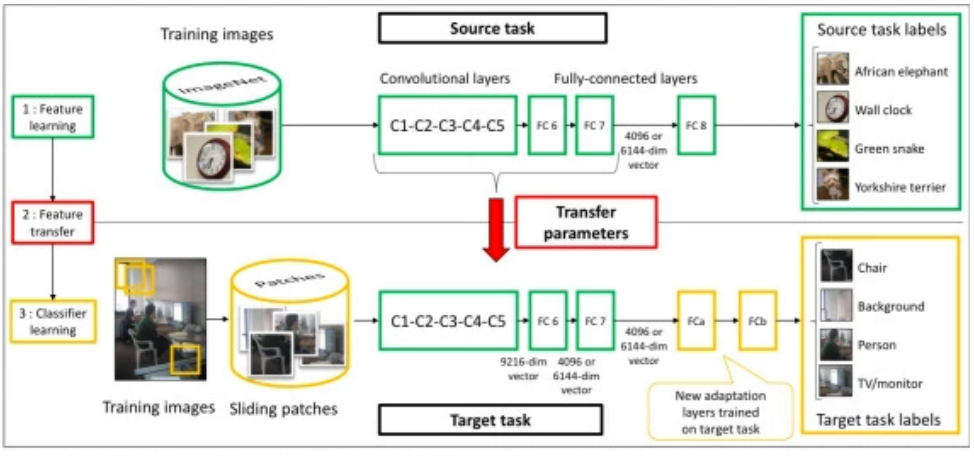
- What we can do is that we use architecture of the model while we initialize all the weights randomly and train the model according to our dataset again.



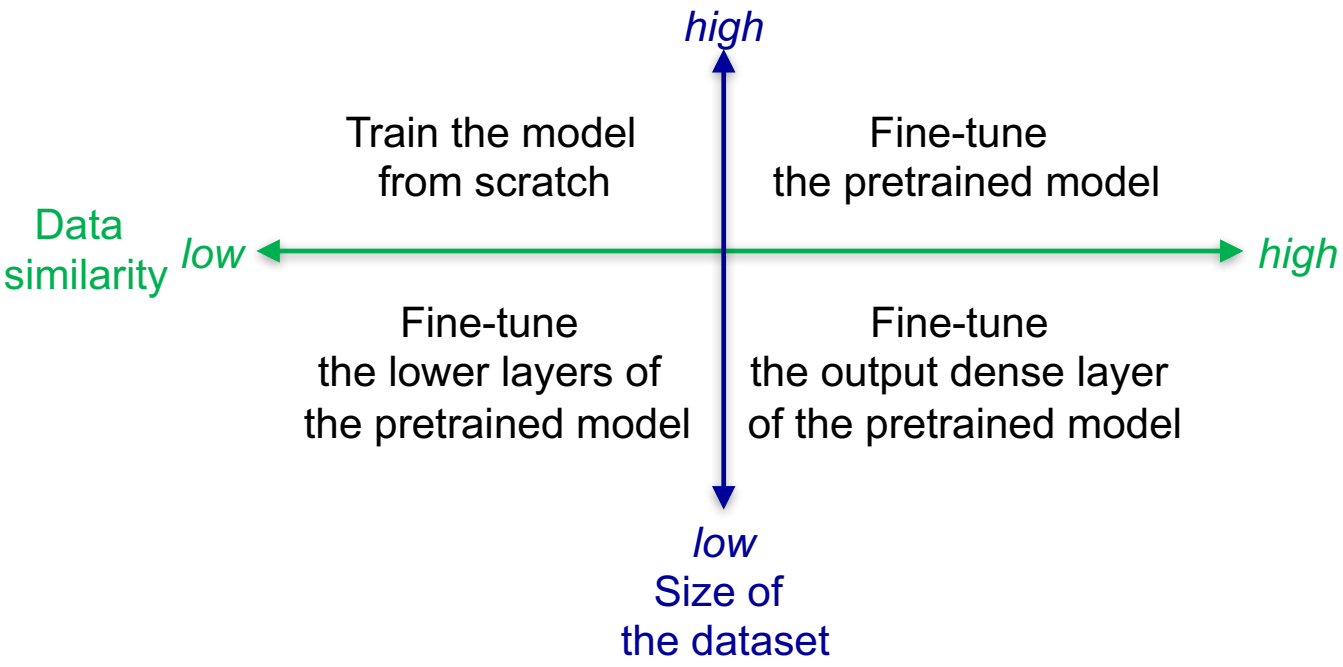


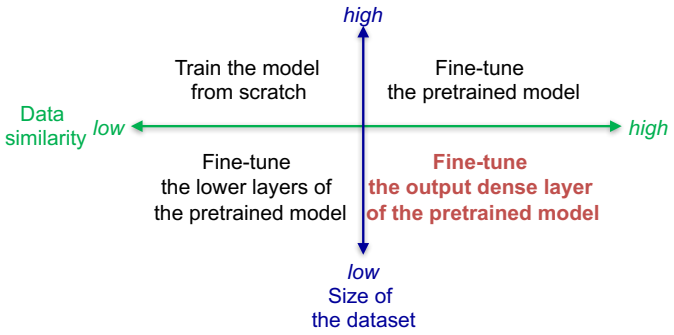
Train some layers while freeze others

- Another way to use a pretrained model is to train it partially.
- Earlier features: more generic features (*e.g.*, edge detectors or color blob detectors); later layers: more specific to the details of the classes contained in the original dataset
- We keep the weights of initial layers of the model frozen while we retrain only the higher layers.



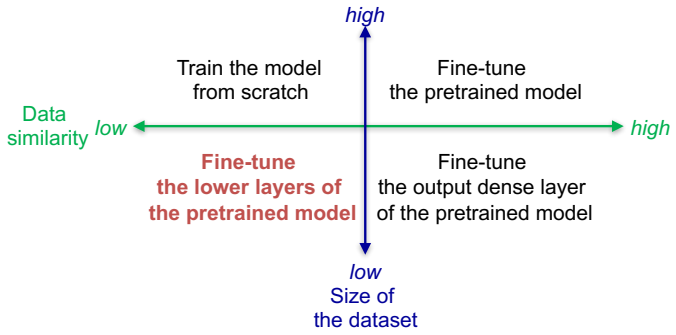
[Oquab et al., 2014]





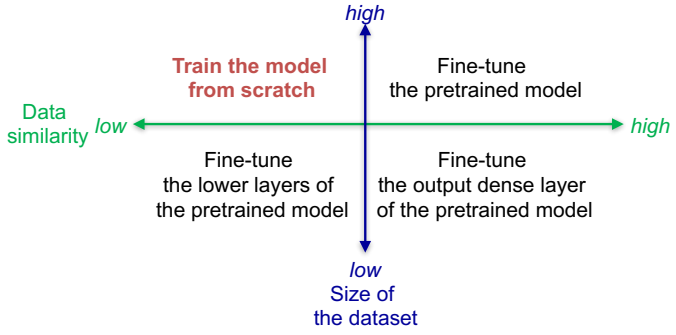
Small-sized dataset, high-data similarity

- In this case, since the data similarity is very high, we do not need to retrain the model. All we need to do is to customize and modify the output layers according to our problem statement.
- We use the pretrained model as a feature extractor.
- Suppose we decide to use models trained on ImageNet to identify if the new set of images have cats or dogs. Here the images we need to identify would be similar to imageNet, however we just need two categories as my output cats or dogs. In this case all we do is just modify the dense layers and the final softmax layer to output 2 categories instead of a 1,000.



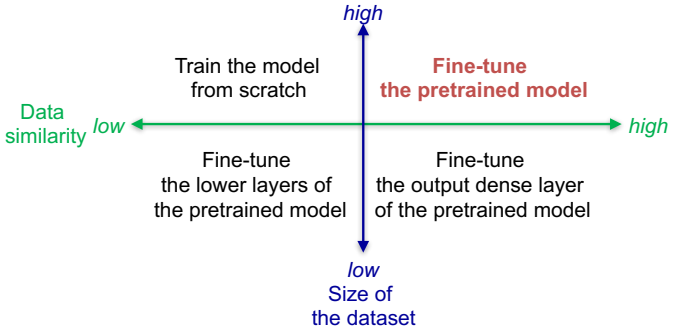
Small-sized dataset, low-data similarity

- In this case we can freeze the initial (lets say k) layers of the pretrained model and train just the remaining(n-k) layers again. The top layers would then be customized to the new data set.
- Since the new data set has low similarity it is significant to retrain and customize the higher layers according to the new dataset.
- The small size of the data set is compensated by the fact that the initial layers are kept pretrained(which have been trained on a large dataset previously) and the weights for those layers are frozen.



Large-sized dataset, low-data similarity

- In this case, since we have a large dataset, our neural network training would be effective. However, since the data we have is very different as compared to the data used for training our pretrained models.
- The predictions made using pretrained models would not be effective.
- Hence, its best to train the neural network from scratch according to your data.



Large-sized dataset, high-data similarity

- This is the ideal situation.
- In this case the pretrained model should be most effective.
- The best way to use the model is to retain the architecture of the model and the initial weights of the model. Then we can retrain this model using the weights as initialized in the pretrained model.

Hands on Programming: Transfer Learning

Thank you
for your attention!!!

(Q & A)

hisuk (AT) korea.ac.kr

http://www.ku-milab.org