[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

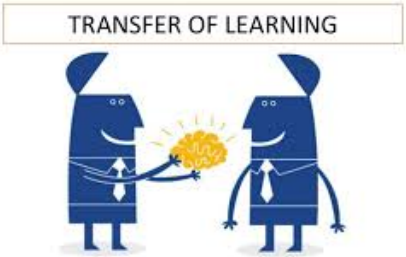
October 26, 2017

Contents

Transfer Learning In a Nutshell

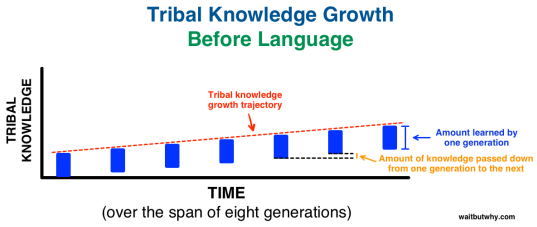
[Reference: D. Gupta, "Transfer learning & The art of using Pre-trained Models in Deep Learning," Analytics Vidhya, 2017]

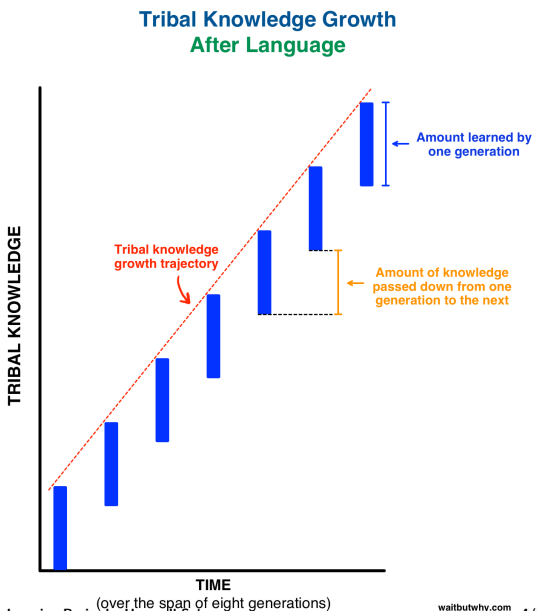
What is Transfer Learning?



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


Tribal Knowledge Growth
Before Language
TRIBAL KNOWLEDGE
Tribal knowledge growth trajectory
Amount learned by one generation
Amount of knowledge passed down from one generation to the next
TIME
(over the span of eight generations)
waitbutwhy.com


Tribal Knowledge Growth
After Language
TRIBAL KNOWLEDGE
Tribal knowledge growth trajectory
Amount learned by one generation
Amount of knowledge passed down from one generation to the next
TIME
(over the span of eight generations)
waitbutwhy.com

- 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

Transfer Learning with Pretrained Models

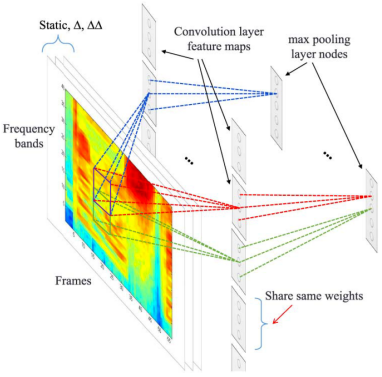
Pretrained models

- 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.

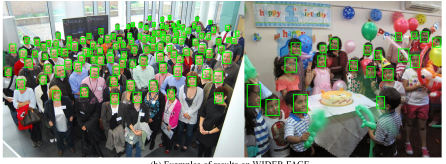
- A pretrained model may not be 100% accurate in your application, but it saves huge efforts required to re-invent the wheel.

- 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

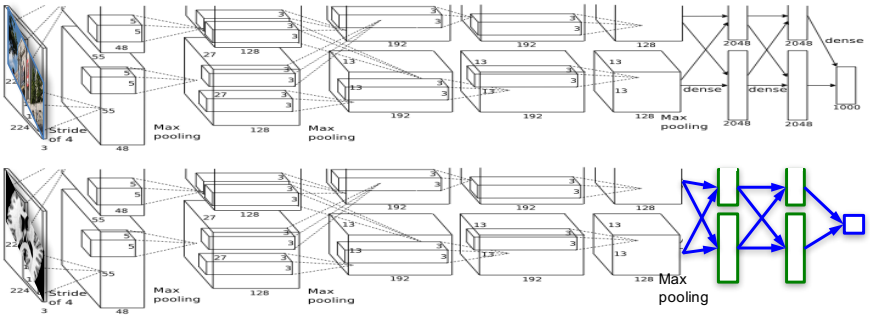


(b) Examples of results on WIDER FACE

A task of face detection

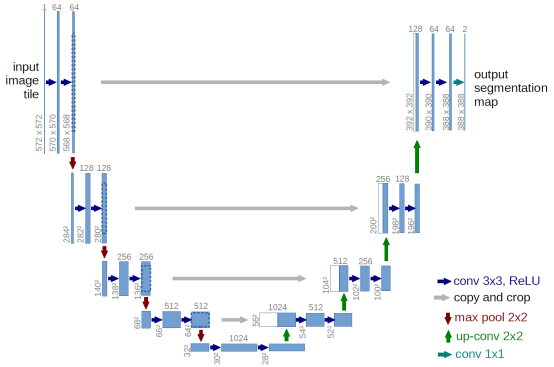
Feature extraction

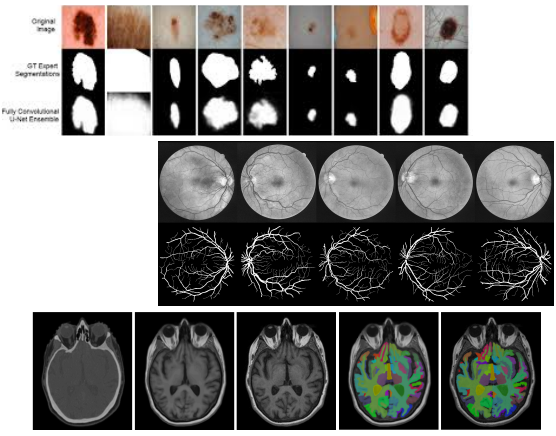
- We can use a pretrained model as a feature extraction mechanism.
- What we can do is that we can remove the output layer (the one which gives the probabilities for being in each of the 1000 classes) and then use the entire network as a fixed feature extractor for the new data set.



Use the Architecture of the pretrained model

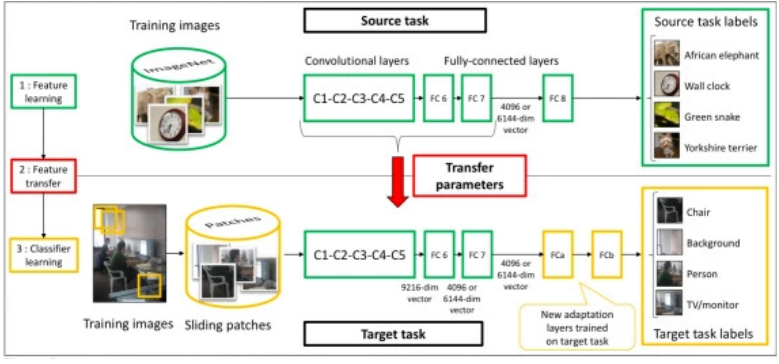
- What we can do is that we use architecture of the model while we initialize all the weights randomly and train the model according to our dataset again.



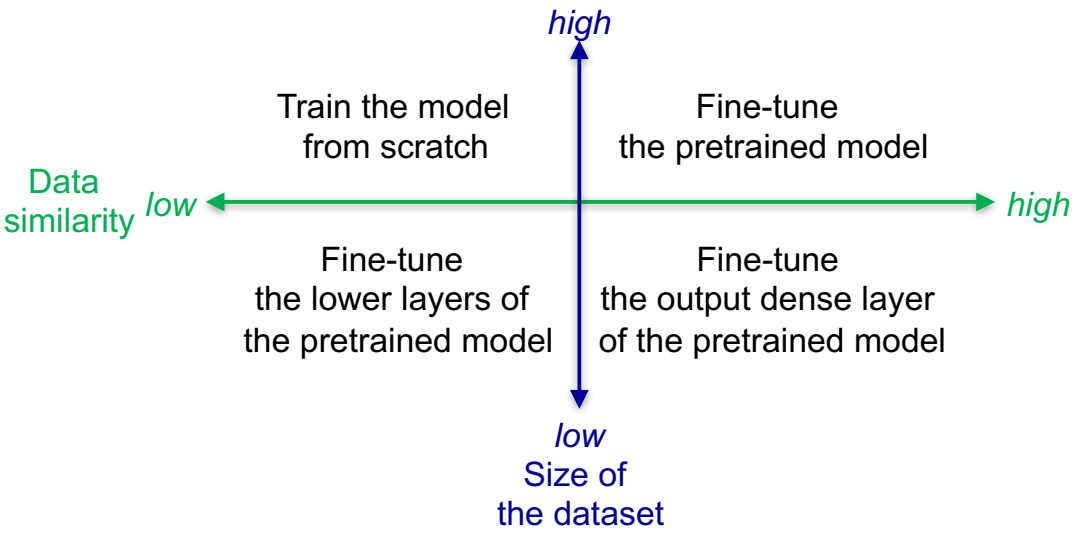


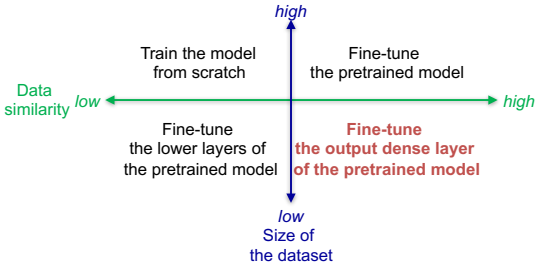
Train some layers while freeze others

- 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.



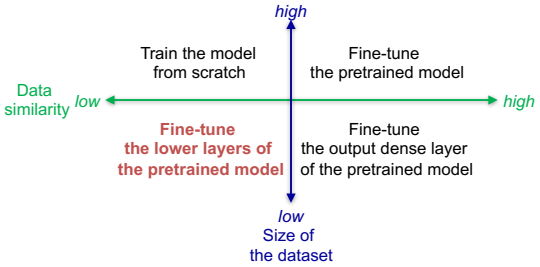
[Oquab et al., 2014]


high
Train the model
from scratch
Fine-tune
the pretrained model
Data
similarity
low
high
Fine-tune
the lower layers of
the pretrained model
Fine-tune
the output dense layer
of the pretrained model
low
Size of
the dataset



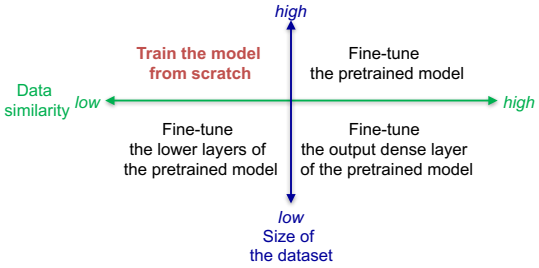
Small-sized dataset, high-data similarity

- 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.



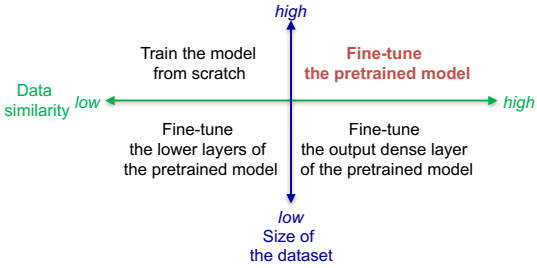
Small-sized dataset, low-data similarity

- 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

Machine Intelligence Lab