PyData 2017 Delhi, India

Transfer Learning Using Keras and Tensorflow

Dr Amita Kapoor, Associate Professor,
Shaheed Rajguru College of Applied Sciences for Women,
University of Delhi- India
Email: dr.amita.Kapoor@ieee.org
Website: http://www.dramitakapoor.com

Traditional Deep Neural Networks

DNNs are able to learn effectively from large amounts of data.
Data is presented in input-output pairs (supervised learning).
Requires enormous computing power

Traditional Deep Neural Networks

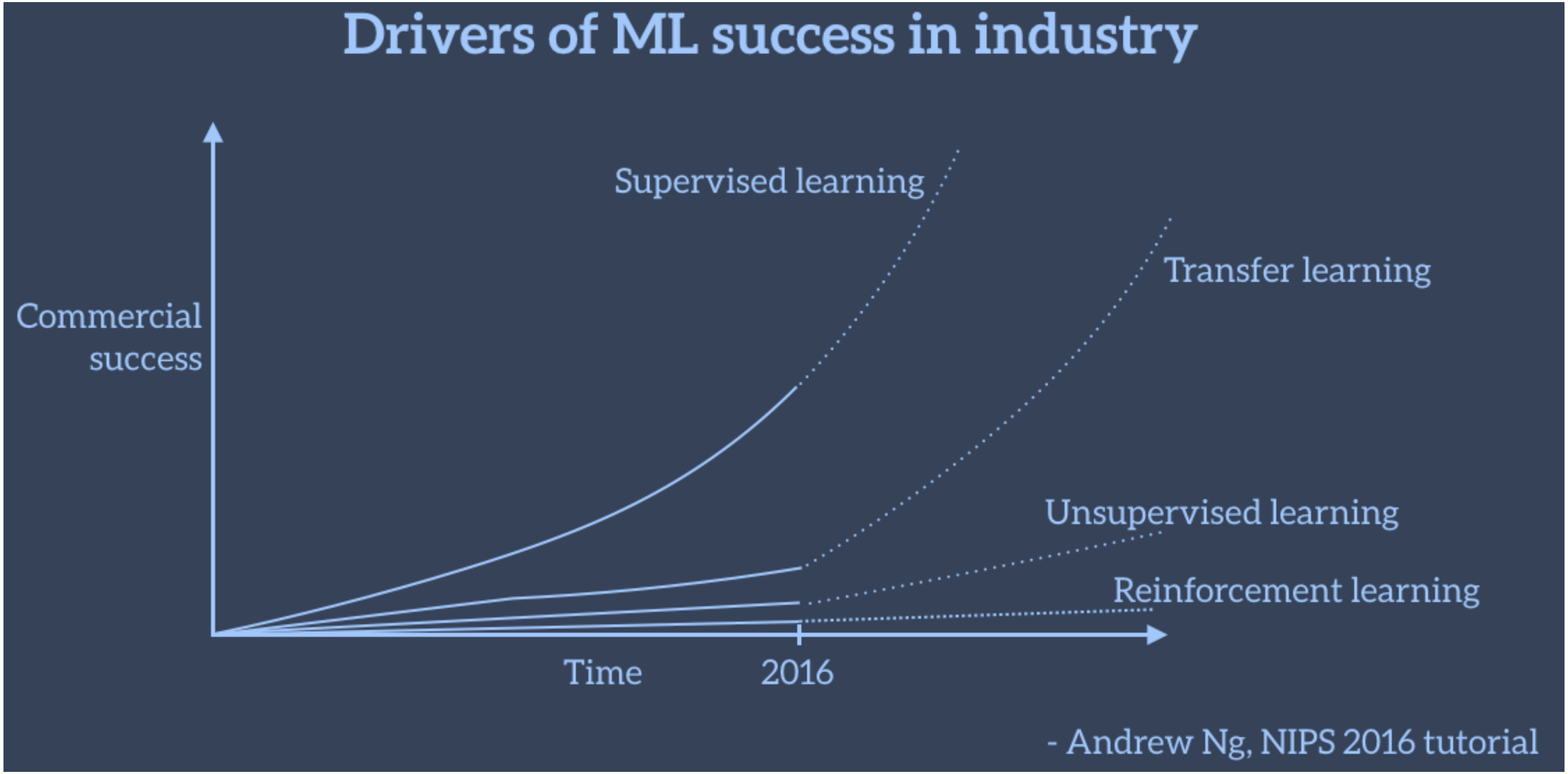
DNNs are able to learn effectively from large amounts of data.	Large Data is not always available
Data is presented in input-output pairs (supervised learning).	Labelled Data is not always available
Requires enormous computing power	Is expensive and Time consuming

Solution

Transfer Learning

Transfer learning will become a key driver of Machine Learning success in industry.

-Andrew Ng (NIPS 2016)



Taken From: http://ruder.io/transfer-learning/

What is Transfer Learning?

- People can intelligently apply knowledge learned previously to solve new problems.
- Not a new concept:
  - NIPS-95 Learning to Learn: *Need for lifelong machine learning methods that retain and reuse previously learned knowledge*.
  - DARPA 2005: *The ability of a system to recognize and apply knowledge and skills learned in previous task to novel tasks*.

What is Transfer Learning?

- Transfer knowledge to new conditions.
- Reuse of some or all of the training (data) of a prior model:
    - feature representations
    - neural-node layering
    - Weights
    - training method
    - loss function
    - learning rate etc.
- Tap into the knowledge gained on prior projects: Supervised, Unsupervised, Reinforcement Learning
- Extracts knowledge from one or more source tasks and apply the knowledge to a target task.

Transfer Learning- ML Commercial Success- Key?

- Traditional Models have reported Super human performance in certain tasks.
- Yet, when they are used in production, the performance deteriorates.
- Real world is very different from the structured data used for training and testing.
- Individual users can have slightly different preferences.
- Transfer Learning can help deal with these and allow us to use ML beyond tasks and domains where
  - labelled data is plentiful,
  - data is outdated
- Boost productivity by reducing time to implement new projects

Transfer Learning: Formal Definition[1]

- **Domain** *D* consists of two components: Feature Space χ and marginal Probability Distribution *P(X)* where $X = \{x_1, \ldots, x_n\} \in \chi$

$$D = \{\chi, P(X)\}$$

- **Task** *T* also consists of two components: a Label Space *Y* and an objective predictive function *f*(.)

$$T = \{Y, f(\cdot)\}$$

- In terms of probability the objective predictive function *f(x)* can be written as *P(y|x)*.
- Consider one source domain D_s and corresponding source task T_s and one target domain D_T and target task T_T

Transfer Learning: Formal Definition[1]

- Consider one source domain D_s and corresponding source task T_s and one target domain D_T and target task T_T
- Traditional Machine Learning*:*

$$D\downarrow s = D\downarrow T \quad and \quad T\downarrow s = T\downarrow T$$

- Transfer Learning*:*

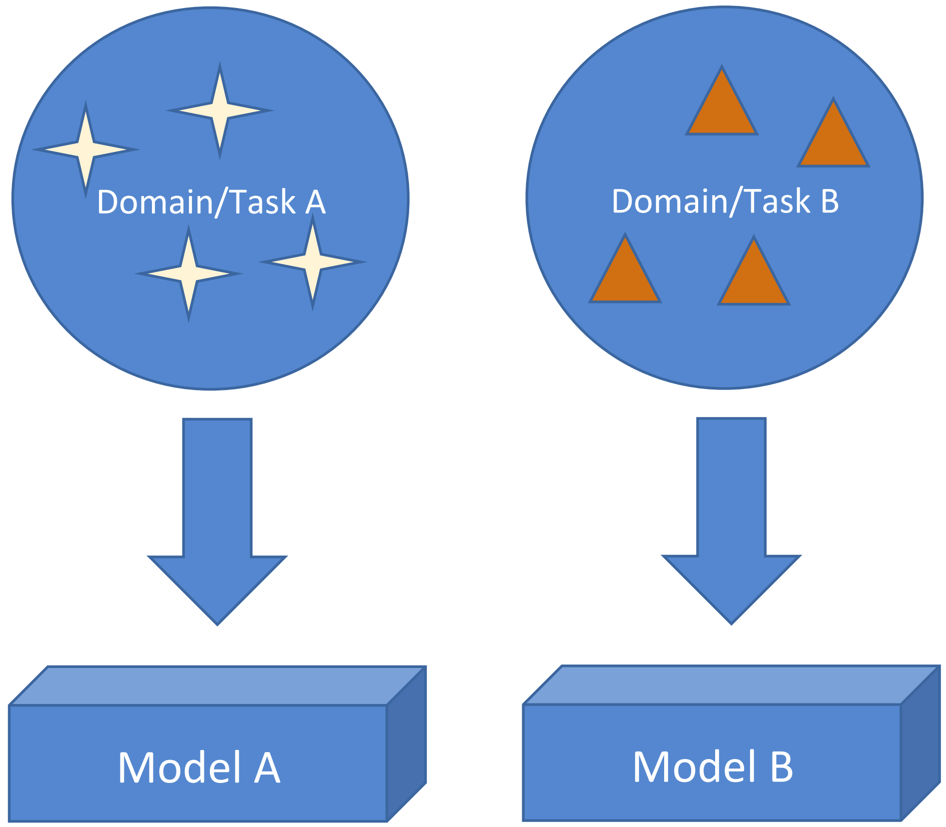
$$D\downarrow s \neq D\downarrow T \quad or \quad T\downarrow s \neq T\downarrow T$$

Source and target conditions can vary in four ways =>

Four Scenarios

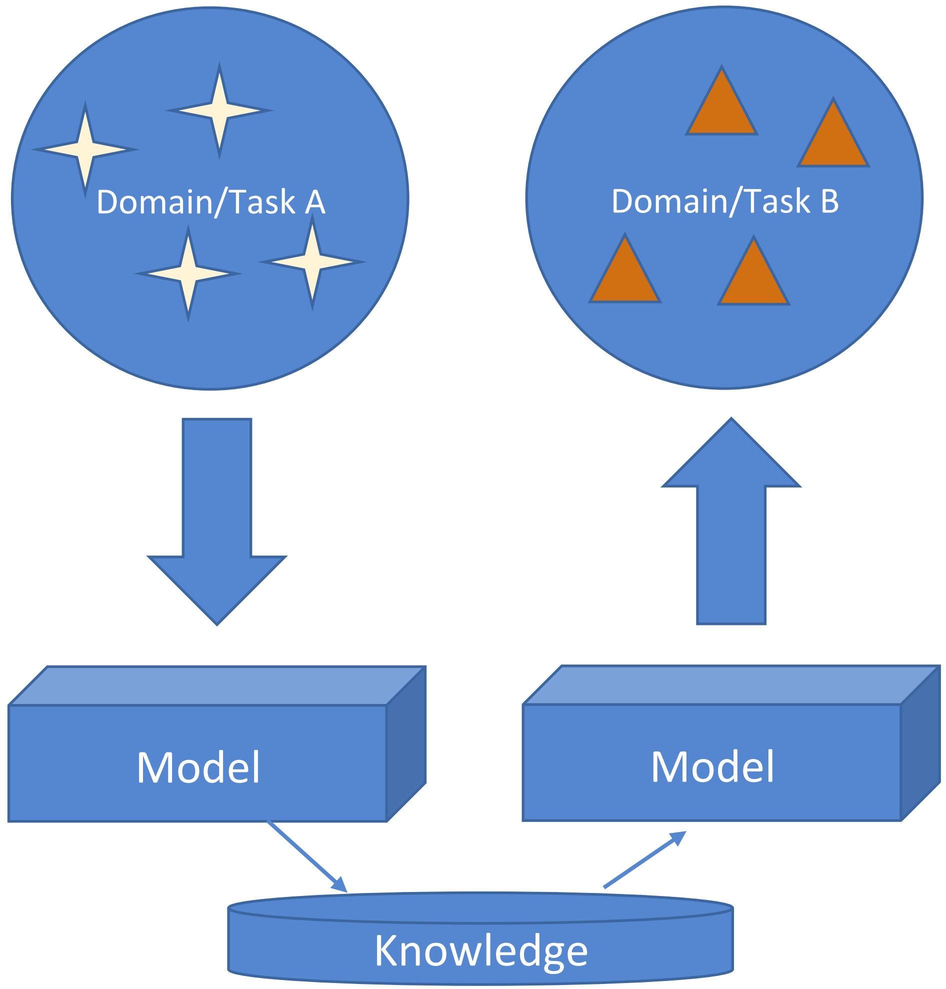
Traditional vs Transfer Learning

$D_s = D_T$ and $T_s = T_T$



Training and Evaluation on same task/domain

$D_s \neq D_T$ or $T_s \neq T_T$



Training and Evaluation on same task/domain

Transductive Transfer Learning

Source Domain and Target Domain different: $D_s \neq D_T$

- $\chi_s \neq \chi_T$: Heterogenous Transfer Learning
  - E.g. Source and Target languages are different
- $P(X_S) \neq P(X_T)$: Frequency Feature Bias/Domain Adaption
  - E.g. Source and target documents are on different topics.

Inductive Transfer Learning

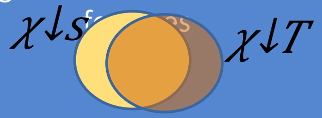
Source Task and Target Task different: $T_s \neq T_T$

- $Y_s \neq Y_T$: **Main focus of this talk**
  - E.g. Source documents had with binary classification and Target documents have multiple classification.
- $P(Y_S | X_S) \neq P(Y_T | X_T)$: Context Feature Bias:

Transfer Learning Approach

Instance based approach:

Source and target domains have lot of overlapping features



Feature-based approach:

Source and target have some overlapping features

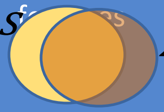
Parameter-based approach

Relational Approach

Transfer Learning Approaches

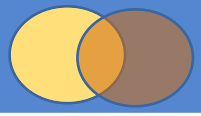
Instance based approach:

Source and target domains have lot of overlapping features

$\chi{\downarrow}s$  $\chi{\downarrow}T$

Feature-based approach:

Source and target have some overlapping features

$\chi{\downarrow}s$  $\chi{\downarrow}T$

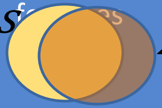
Parameter-based approach

Relational Approach

Transfer Learning Approaches

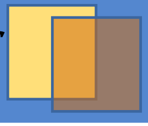
Instance based approach:

Source and target domains have lot of overlapping features

χ_S  χ_T

Feature-based approach:

Source and target have some overlapping features

$\varphi(\chi_S)$  $\varphi(\chi_T)$

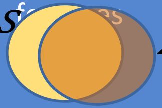
Parameter-based approach

Relational Approach

Transfer Learning Approaches

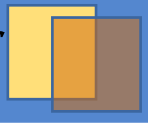
Instance based approach:

Source and target domains have lot of overlapping

χ_s  χ_T

Feature-based approach:

Source and target have some overlapping features

$\varphi(\chi_s)$  $\varphi(\chi_T)$

Parameter-based approach:

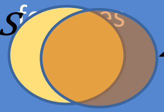
Source and target tasks are related, and so what has been learned from source can be transferred to target.

Relational Approach

Transfer Learning Approaches

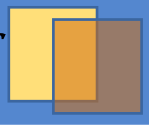
Instance based approach:

Source and target domains have lot of overlapping features

$\chi\downarrow s$  $\chi\downarrow T$

Feature-based approach:

Source and target have some overlapping features

$\varphi(\chi\downarrow s$  $\varphi(\chi\downarrow T)$

Parameter-based approach:

Source and target tasks are related, and so what has been learned from source can be transferred to target.

Relational Approach:

If two relational domains are related, they may share similar relations among Objects.

Transfer Learning Techniques

Using pre-trained models: Useful when new task has different label space.	• Training an entire convolutional neural network from scratch requires a very large dataset and time, instead use a pretrained CNN
Learn domain-invariant representations: Useful for Heterogenous transfer learning and Domain Adaptation.	• Use models to learn representations that do not change based on the domain: Find a Common Latent Feature Space. Use Denoised Autoencoders
Make representations more similar	• Pre-process such that representations of both domains become more similar to e/o
Domain Confusion	• Add an objective function to existing model that confuses the two domains.

Pre-trained Models

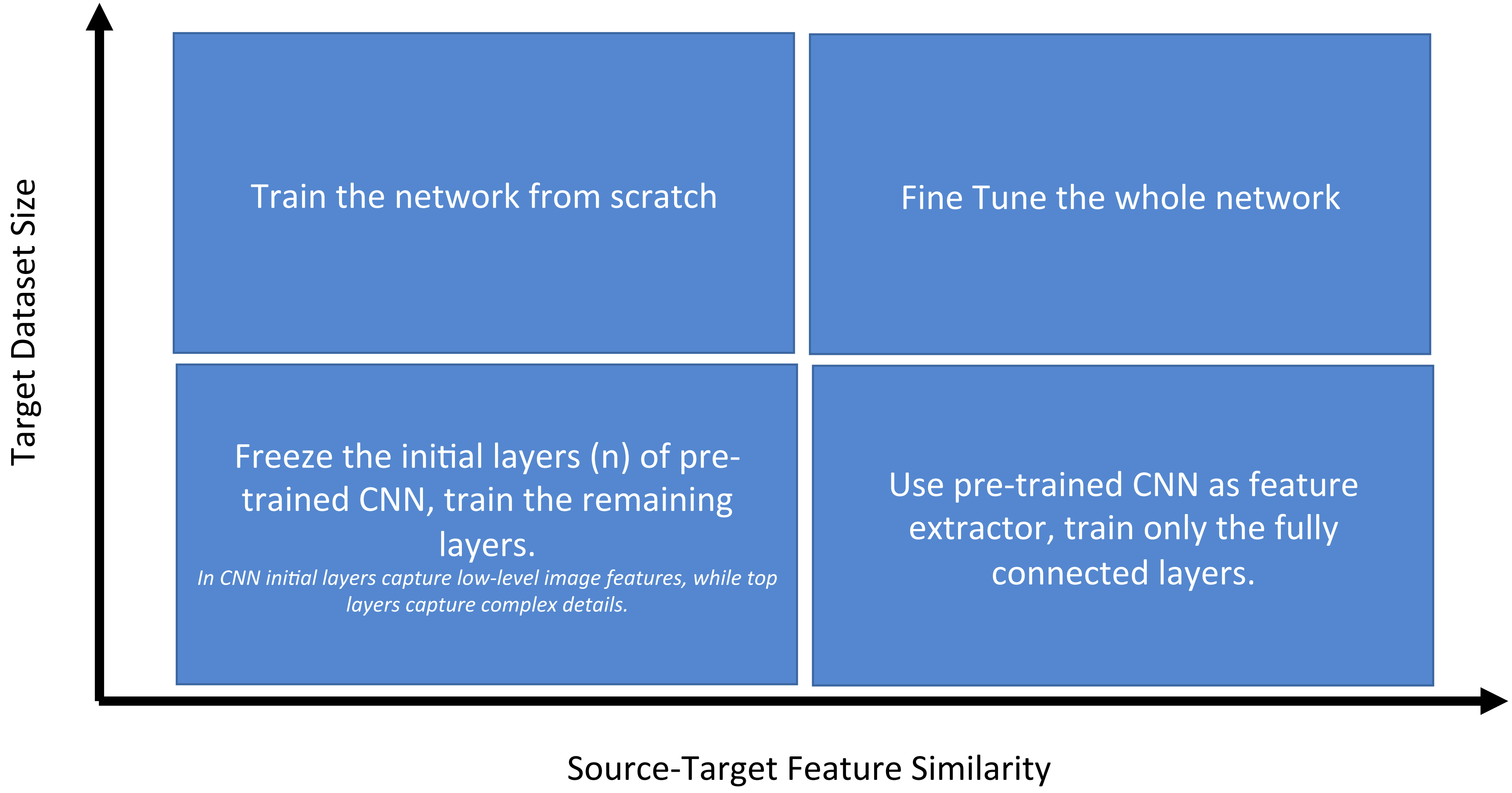
Pre-trained CNN as feature extractor:

Take a pre-trained CNN model, remove the last fully connected layers, use the remaining CNN as feature extractor for the new dataset. The output of this CNN feature extractor is fed to a classifier. The classifier is trained for the new dataset.

Fine tune pre-trained CNN:

The weights of both the pre-trained CNN layers (all or some) and fully connected classifier layer/s are fine tuned using backpropagation.

Which method to employ?



Pre-trained Models in Keras/Tensorflow

Models for Image classification
with weights trained on ImageNet

- Xception
- VGG16
- VGG19
- ResNet50
- InceptionV3
- MobileNet

Import Models:
from keras.applications.model_name import model_name
From keras.applications.model_name import preprocess_input, decode_predictions

from keras.applications.vgg16 import VGG16

Predicting Dog Breed Using Xception

- Detect whether given image is a dog or not
  - We use Resnet60 trained on Imagenet

```
ResNet50_model = ResNet50(weights='imagenet‘)

def ResNet50_predict_labels(img_path):
    # returns prediction vector for image located at img_path
    img = preprocess_input(path_to_tensor(img_path))
    return np.argmax(ResNet50_model.predict(img))

def dog_detector(img_path):
    prediction = ResNet50_predict_labels(img_path)
    return ((prediction <= 268) & (prediction >= 151))
```

Predicting Dog Breed Using Xception

- Get Bottleneck features (The output of last CNN layer for the training dataset)

```
# Generated using predict_generator method
bottleneck_features = np.load('bottleneck_features/DogXceptionData.npz')

train_VGG16 = bottleneck_features['train']
valid_VGG16 = bottleneck_features['valid']
test_VGG16 = bottleneck_features['test']
```

Predicting Dog Breed Using Xception

- Define the classifier to be used over pre-trained Xception Model

```
TLmodel_model = Sequential()
TLmodel_model.add(GlobalAveragePooling2D(input_shape=train_new.shape[1:]))
#TLmodel_model.add(Dense(400, activation='relu'))
#TLmodel_model.add(Dropout(0.2))
TLmodel_model.add(Dense(200, activation='relu'))
TLmodel_model.add(BatchNormalization())
TLmodel_model.add(Dropout(0.4))
TLmodel_model.add(Dense(133, activation='softmax'))

TLmodel_model.summary()
```

Predicting Dog Breed Using Xception

- Define the classifier to be used over pre-trained Xception Model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_1 ( (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 200)               409800
_________________________________________________________________
batch_normalization_1 (Batch (None, 200)               800
_________________________________________________________________
dropout_1 (Dropout)          (None, 200)               0
_________________________________________________________________
dense_2 (Dense)              (None, 133)               26733
=================================================================
Total params: 437,333.0
Trainable params: 436,933.0
Non-trainable params: 400.0
_________________________________________________________________
```

Predicting Dog Breed Using Xception

- Define the optimizer and train the model

```
### Compile the model.
TLmodel_model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy‘])

### Train the model.
from keras.callbacks import ModelCheckpoint
checkpointer = ModelCheckpoint(filepath='saved_models/weights.best.TLmodel_InceptionV3.hdf5',
                               verbose=1, save_best_only=True)

TLmodel_model.fit(train_new, train_targets,
        validation_data=(valid_new, valid_targets),
        epochs=20, batch_size=20, callbacks=[checkpointer], verbose=2, shuffle=True)
```

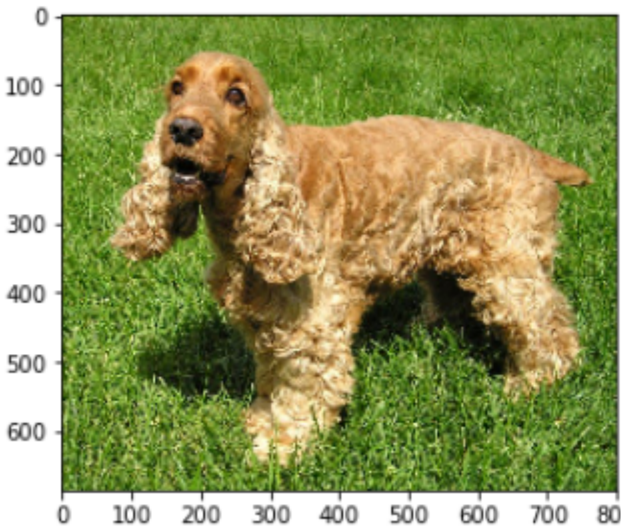
Predicting Dog Breed Using Xception

- Prediction Using the Model

```
Wow, Wow you are a Dog!
And your breed is
Labrador_retriever
```


```
Correct breed is
Chesapeake_bay_retriever
```

```
Wow, Wow you are a Dog!
And your breed is
English_cocker_spaniel
```

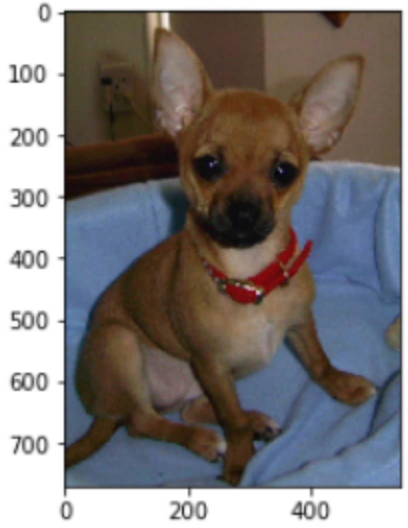



```
Correct breed is
English_cocker_spaniel
```

```
Wow, Wow you are a Dog!
And your breed is
Greater_swiss_mountain_dog
```


```
Correct breed is
Greater_swiss_mountain_dog
```

```
Wow, Wow you are a Dog!
And your breed is
Chihuahua
```




```
Correct breed is
Chihuahua
```

Applications of Transfer Learning

- Learning from simulations and then applying to real world:
  - Real world data is hard to come by. Generate data using simulator: Data has similar feature space, slightly different in marginal probability distributions, and different in conditional probability distributions.
  - E.g.: Self driving Car, Robots, AGI Agents
- Data becoming outdated:
  - Wi-Fi Localization: Locating a mobile device in an indoor environment, position of device changes
- Sentiment Classification:
  - Learn sentiment classification on one topic and apply the model learned on other topics
- Cross Domain Activity Recognition
  - Knowledge about activity learned in one domain (Cleaning Indoor) can be applied to other domain (Doing Laundry).

Further Research

One/Zero shot learning

Aim to learn from only a few/one/zero shot learning.

Multi Task learning

Learn more than one task. Use knowledge acquired by learning from related tasks to do well on target. Source and Target are jointly trained.

References

- Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." *IEEE Transactions on knowledge and data engineering* 22.10 (2010): 1345-1359.
- Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang. "A survey of transfer learning." *Journal of Big Data* 3.1 (2016): 9.
- Ruder, Sebastian. “Transfer Learning – Machine Learning’s Next Frontier”. Medium. March 2017. Web. 13 Aug 2017
- Chen, Minmin, et al. "Marginalized denoising autoencoders for domain adaptation." *arXiv preprint arXiv:1206.4683* (2012).
- https://keras.io/applications/
- Hu, Derek Hao, and Qiang Yang. "Transfer learning for activity recognition via sensor mapping." *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. Vol. 22. No. 3. 2011.
- Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

References

- Glorot, X., Bordes, A., & Bengio, Y. (2011). Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. Proceedings of the 28th International Conference on Machine Learning, 513–520.
- Daumé III, H. (2007). Frustratingly Easy Domain Adaptation. Association for Computational Linguistic (ACL), (June), 256–263.
- Thrun, S., & Pratt, L. (1998). Learning to learn. Springer Science & Business Media
- Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., & Wierstra, D. (2016). Matching Networks for One Shot Learning. NIPS 2016.
- Ravi, S., & Larochelle, H. (2017). Optimization as a Model for Few-Shot Learning.
- Yu, J., & Jiang, J. (2016). Learning Sentence Embeddings with Auxiliary Tasks for Cross-Domain Sentiment Classification. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP2016), 236–246.
- Udacity Artificial Intelligence Nanodegree