Transfer Learning
Multi-task Learning

Jian Li

IIIS, Tsinghua

Transfer Learning

Data used in training a classifier must be properly chosen to be representative

If not? Accuracy will be worse than expected

But suppose we want to apply a classifier to a new or shifting domain? Retrain!

But that’s expensive.

Can we somehow use our existing classifier as a starting point to give us a shortcut?

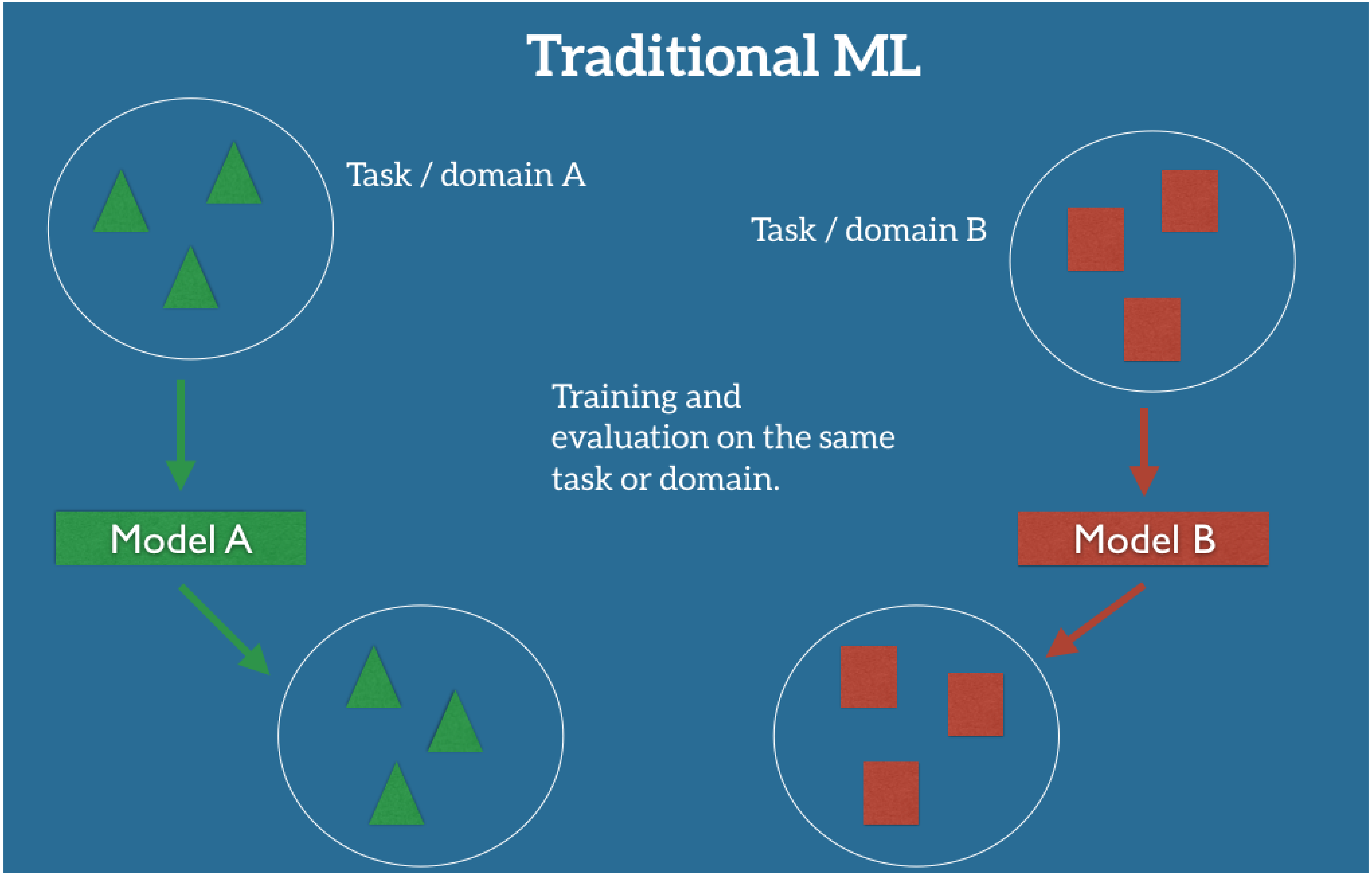
This is Transfer Learning

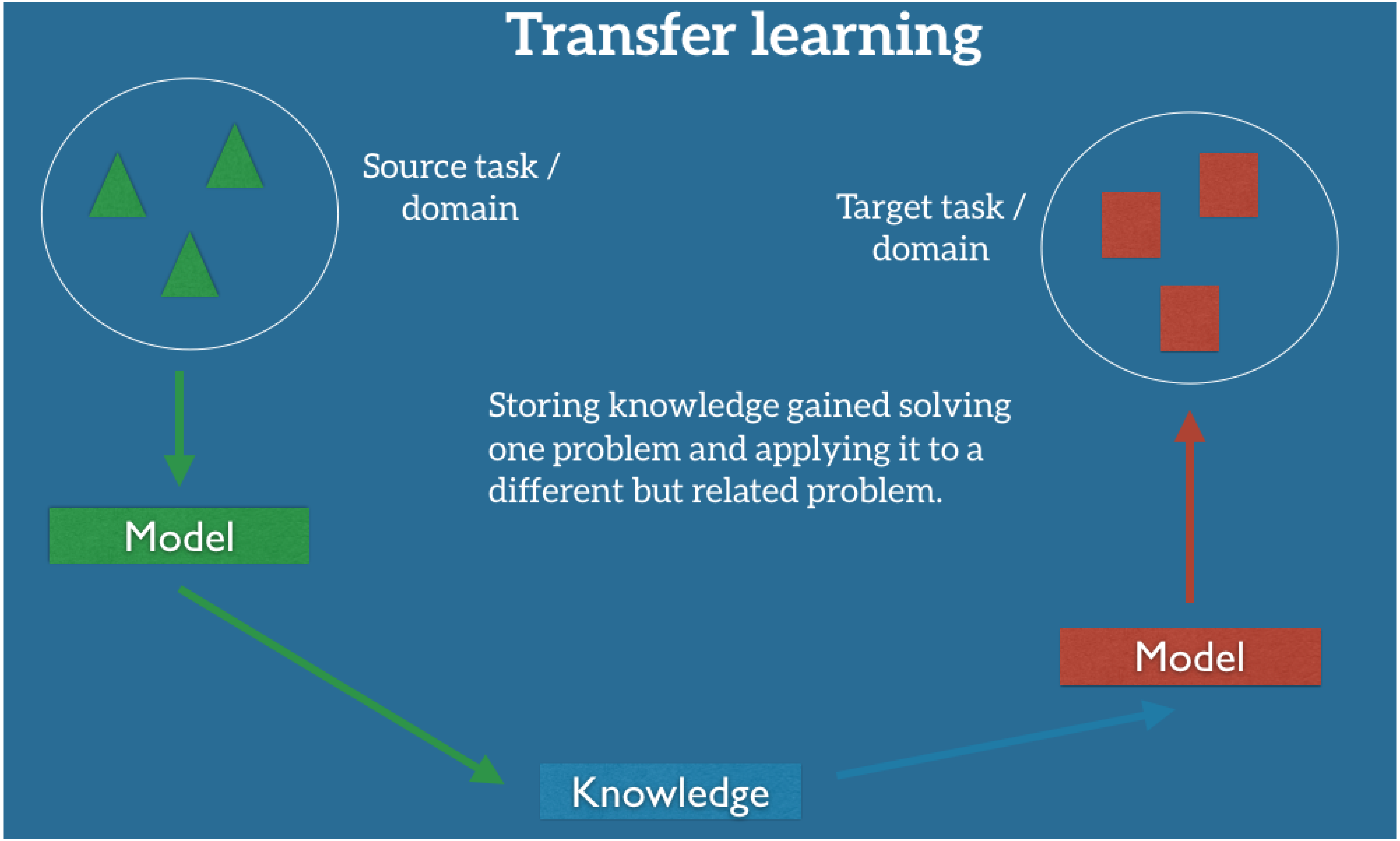
Transfer of Learning
A psychological point of view

- The study of dependency of human conduct, learning or performance on prior experience.

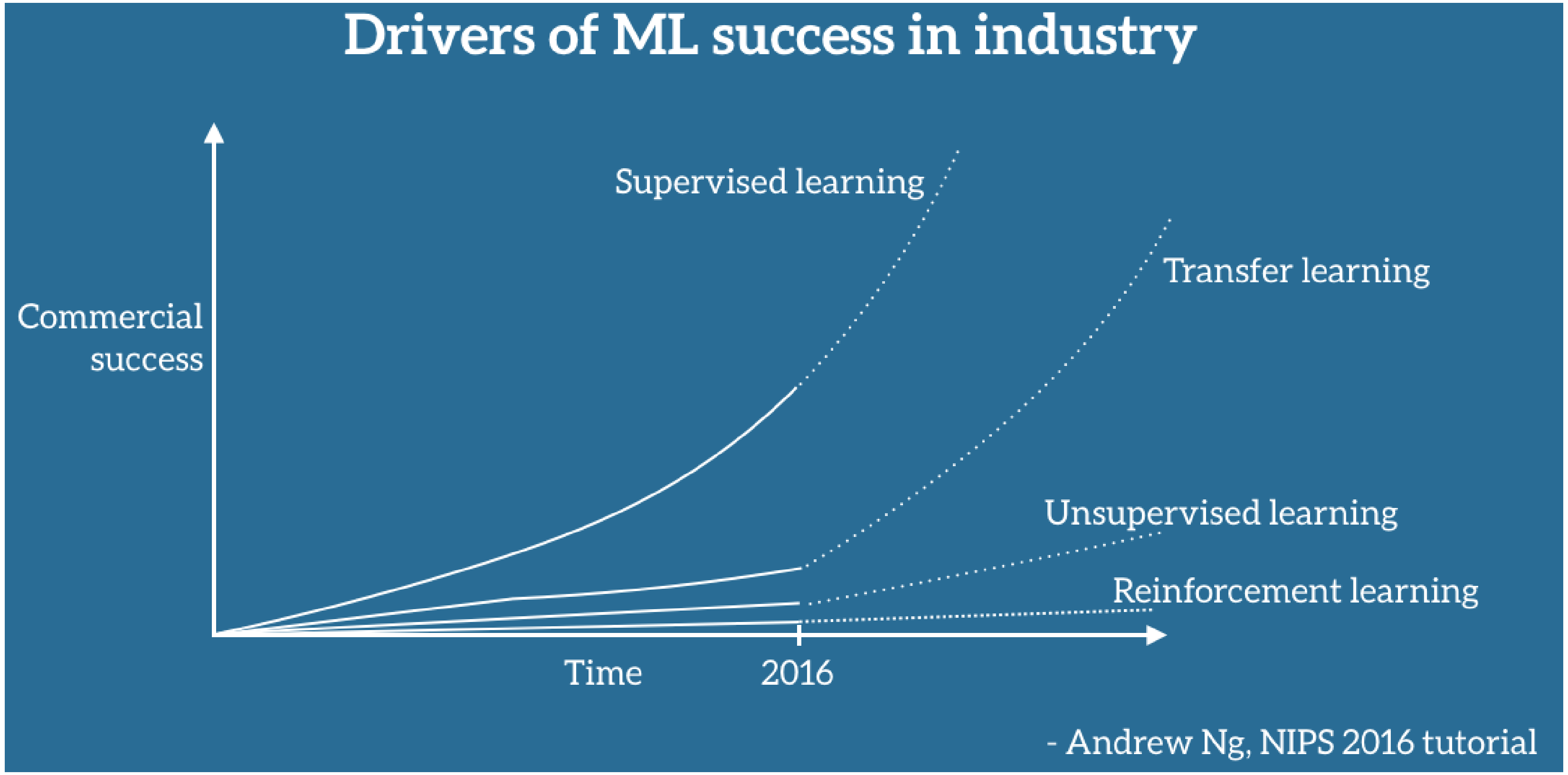
- [Thorndike and Woodworth, 1901] explored how individuals would transfer in one context to another context that share similar characteristics.

- C++ → Java
- Maths/Physics → Computer Science/Economics

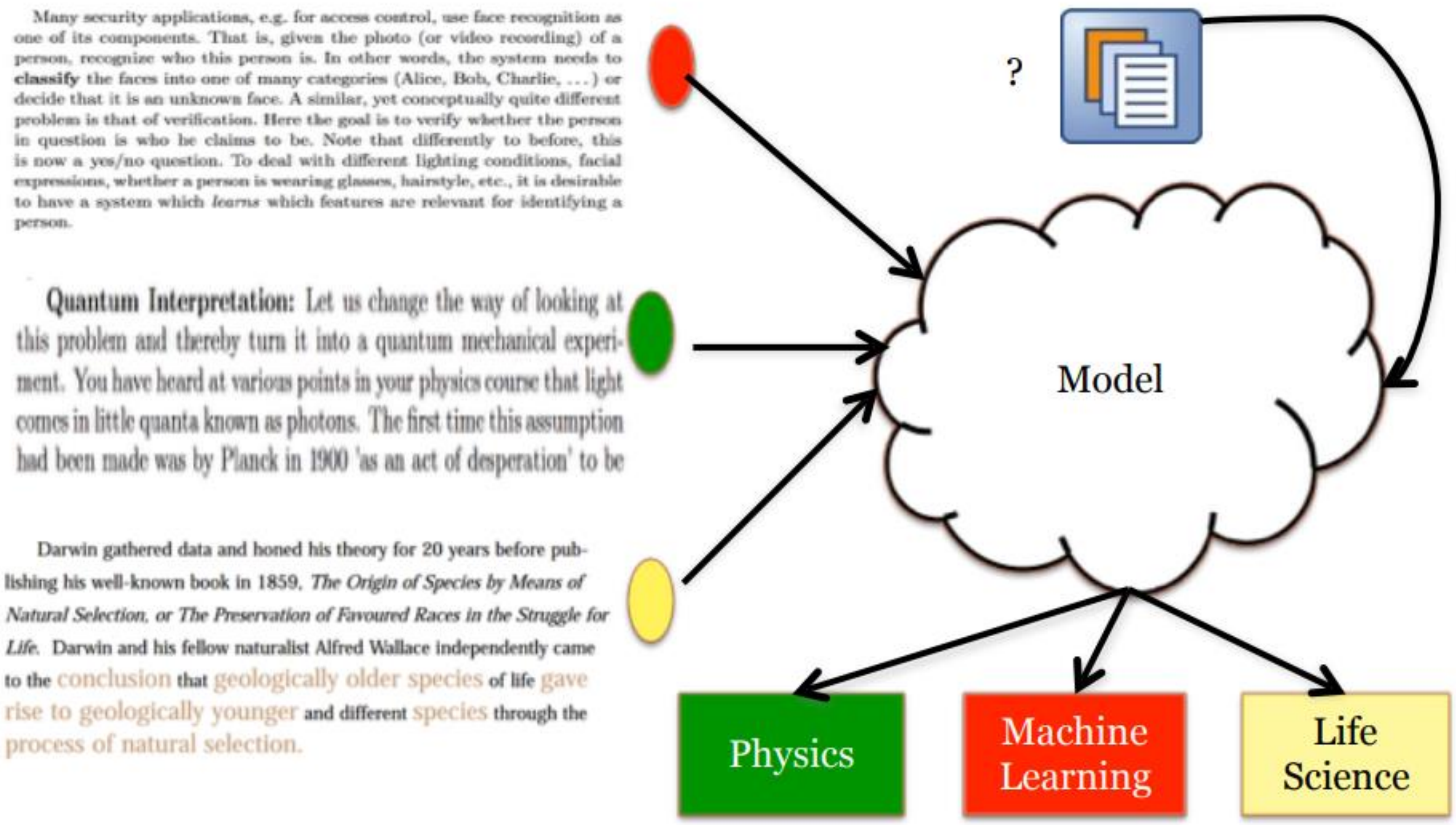

Traditional ML
Task / domain A
Task / domain B
Training and evaluation on the same task or domain.
Model A
Model B



In practice, we seek to transfer as much knowledge as we can from the source setting to our target task or domain. This knowledge can take on various forms transfer learning will be -- after supervised learning -- the next driver of ML commercial success.   -----Ng

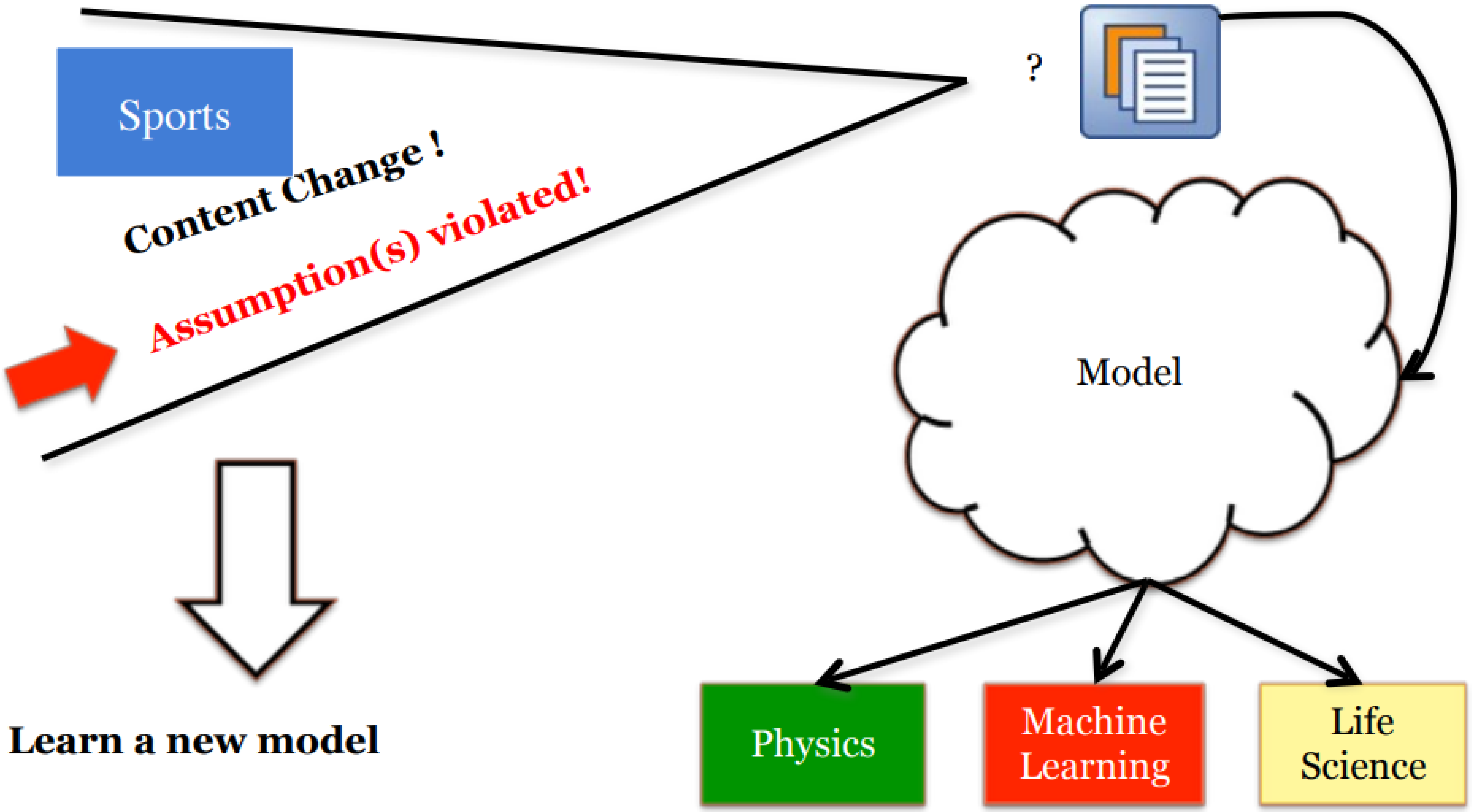


Example: document classification



Based on https://project.dke.maastrichtuniversity.nl/datamining/2013-Slides/transfer-01.ppt

Example: document classification

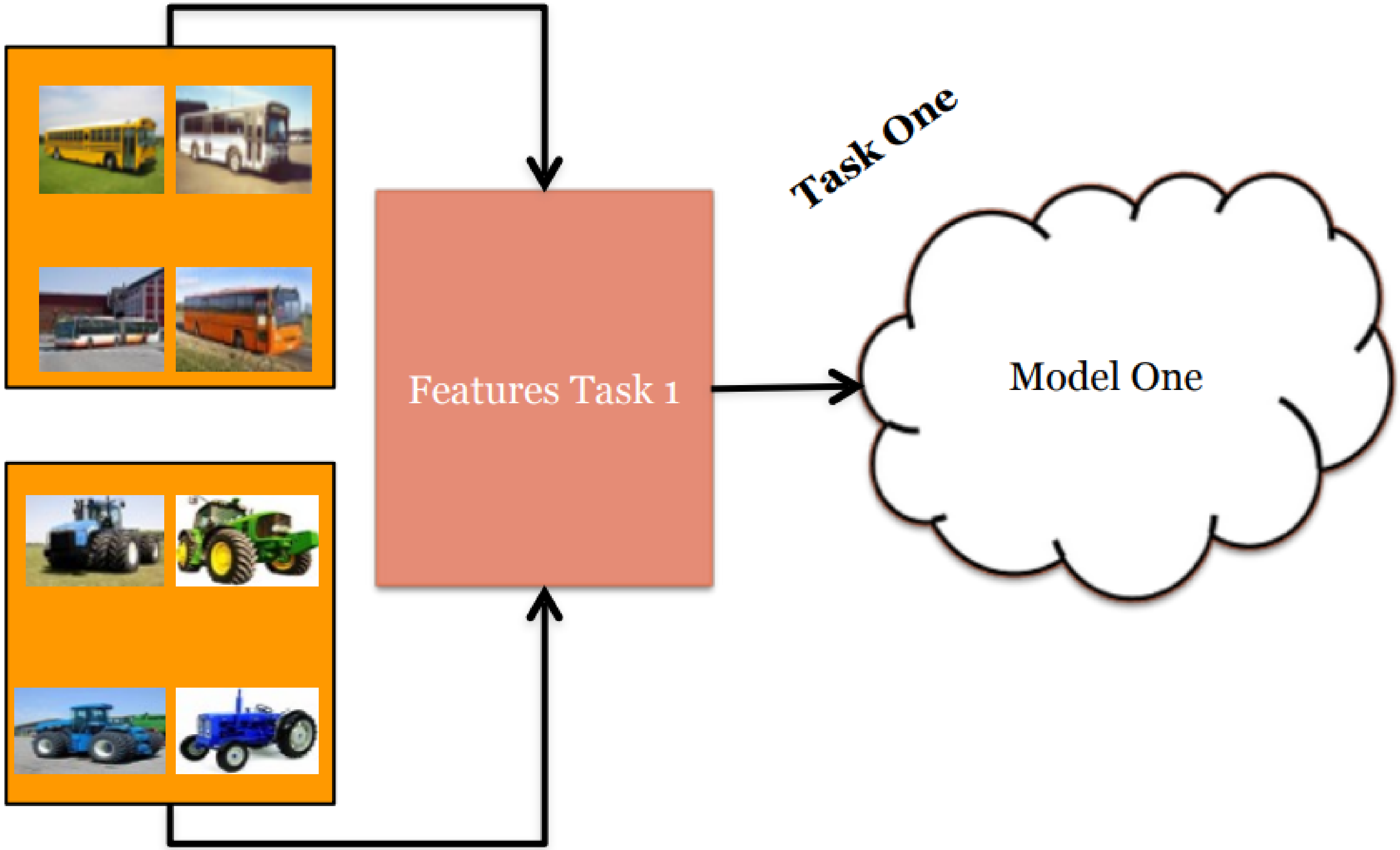


- Two possible violations:
- Same distribution? •
- Different proportions of physics, machine learning, life science

- Same feature space/task
- Sports pages: different vocabulary,
- BOW features change

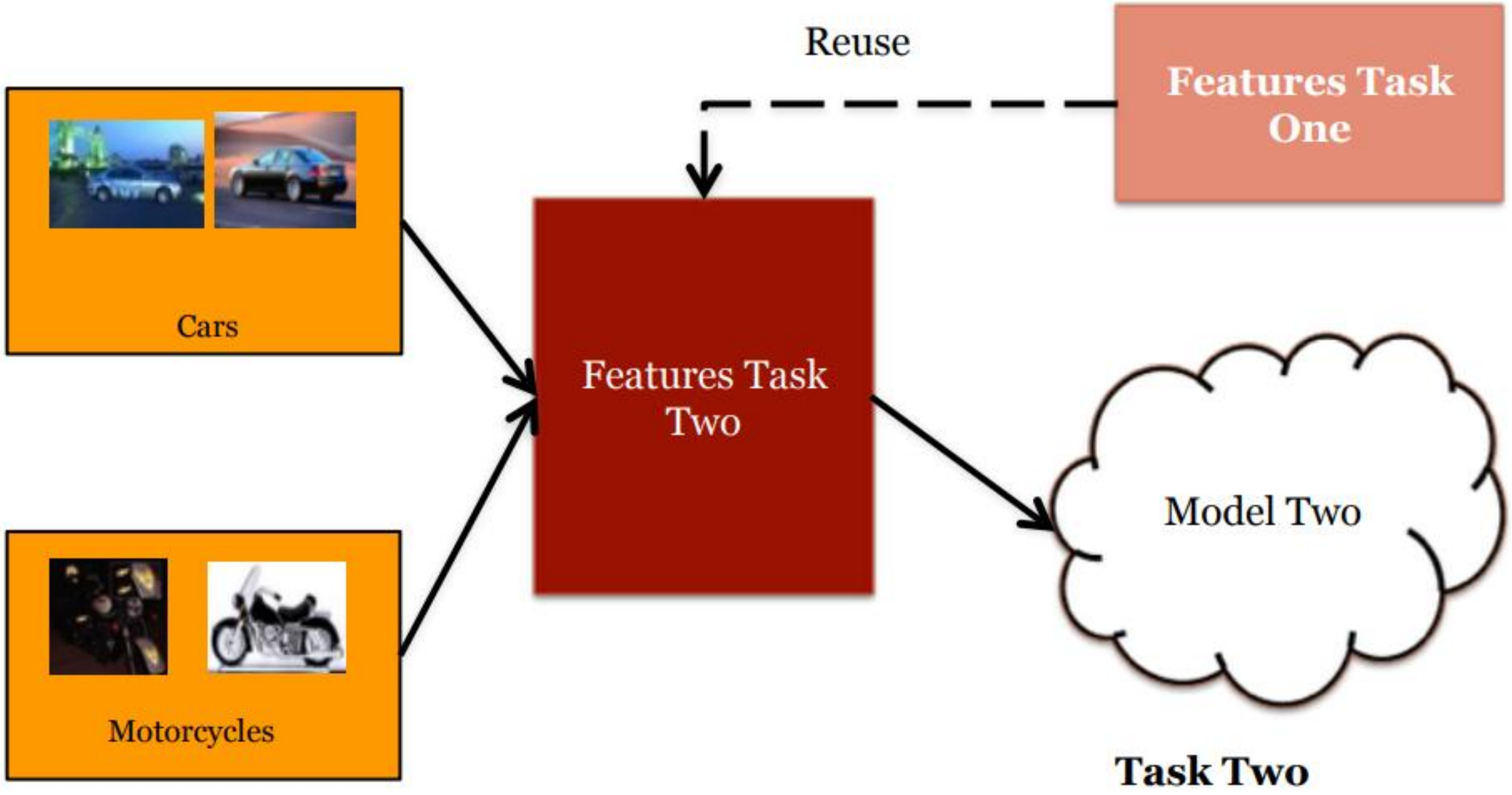
What to do: image classification

- Task 1: Buses vs tractors



Based on https://project.dke.maastrichtuniversity.nl/datamining/2013-Slides/transfer-01.ppt

- Task 2: Cars vs Motorcycles



- Domain Differences

$$\boxed{\mathcal{X}_S \neq \mathcal{X}_T \qquad \mathcal{P}_S(X) \neq \mathcal{P}_T(X)}$$

- Task differences

$$\boxed{\mathcal{Y}_S \neq \mathcal{Y}_T \qquad P(Y_S|X_S) \neq P(Y_T|X_T)}$$

Different distribution

- Example: classify documents from the web into important or not important
- Documents in different domains have the same feature space: Bag of words with frequency of each term
- However, the words have different frequencies in the different domains
- The distribution of features is different

Adaboost

- Assumptions:
  - Source and Target task have same feature space:

$$\mathcal{X}_S = \mathcal{X}_T$$

  - Marginal distributions are different:

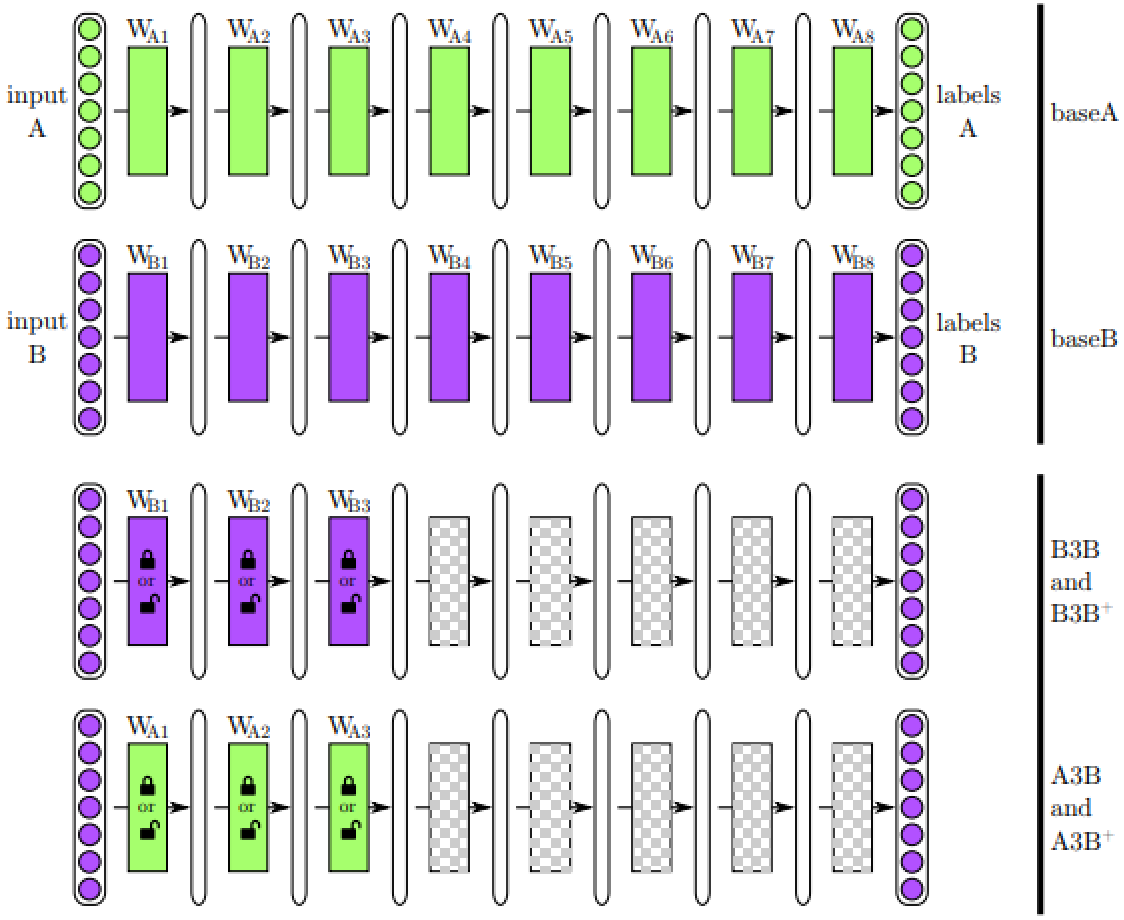
$$P_S(X) \neq P_T(X)$$

$\Rightarrow$ **Not all source data might be helpful !**

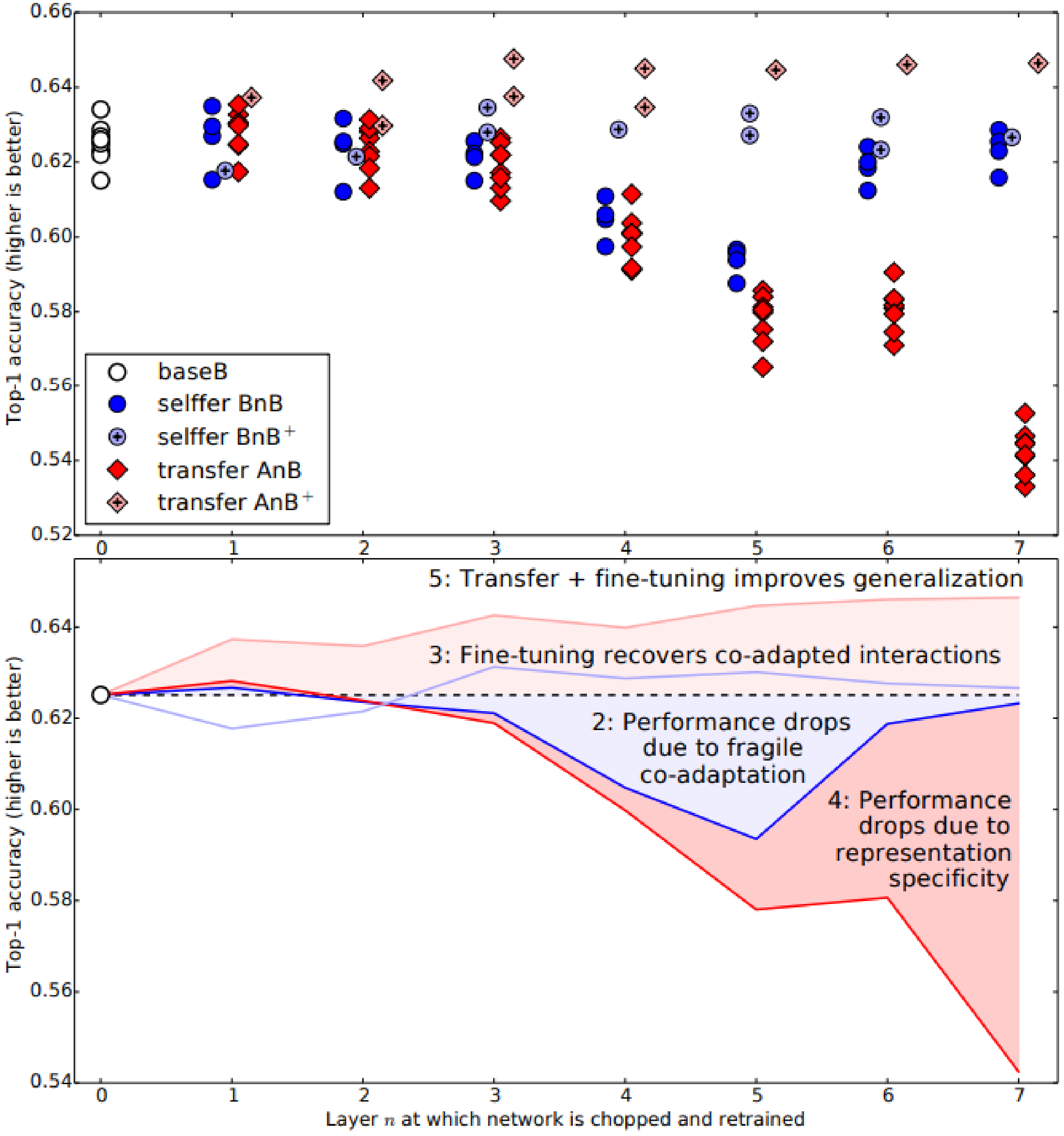
Adaboost

- Idea:
- Iteratively reweight source samples such that:
  - reduce effect of “bad” source instances
  - encourage effect of “good” source instances

How transferable are features in deep neural networks?



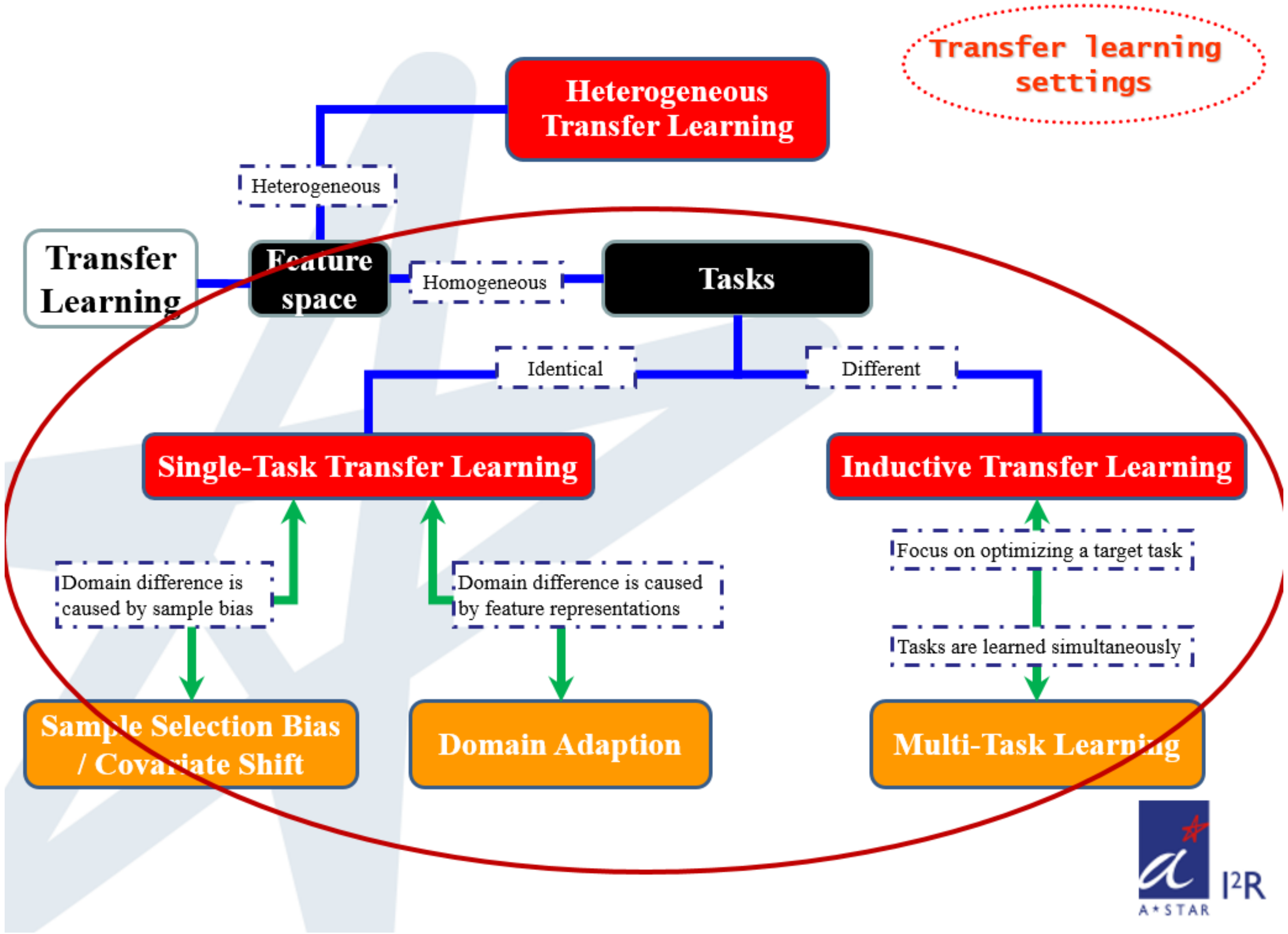
- Third row: In the selffer network control, the first n weight layers of the network (in this example, n = 3) are copied from a base network (e.g. one trained on dataset B), the upper 8 − n layers are randomly initialized, and then the entire network is trained on that same dataset (in this example, dataset B). The first n layers are either locked during training ("frozen" selffer treatment B3B) or allowed to learn ("fine-tuned" selffer treatment B3B+).
- This treatment reveals the occurrence of fragile coadaptation, when neurons on neighboring layers co-adapt during training in such a way that cannot be rediscovered when one layer is frozen.
- Fourth row: The transfer network experimental treatment is the same as the selffer treatment, except that the first n layers are copied from a network trained on one dataset (e.g. A) and then the entire network is trained on the other dataset (e.g. B). This treatment tests the extent to which the features on layer n are general or specific.



- Top: Each marker in the figure represents the average accuracy over the validation set for a trained network.
- The white circles above n =0 represent the accuracy of baseB.
- Each dark blue dot represents a BnB network.
- Light blue points represent BnB+ networks, or fine-tuned versions of BnB.
- Dark red diamonds are AnB networks
- Light red diamonds are the fine-tuned AnB+ versions.

- Bottom: Lines connecting the means of each treatment.

Overview of transfer learning



Multi-task learning

- two main components:
  - a) The architecture used for learning
  - b) the auxiliary task
  - (s) that are trained jointly.

- Auxiliary task helps?
  - Why? Additional supervised information and additional structural knowledge
  - Typically homogeneous tasks, i.e. tasks that are variations of the same classification or regression problem
  - heterogeneous tasks: somewhat more challenging

Artificial auxiliary objectives

- Sometimes, you can create additional objective to learn

Example: Travel time estimation

When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks.

Introduction

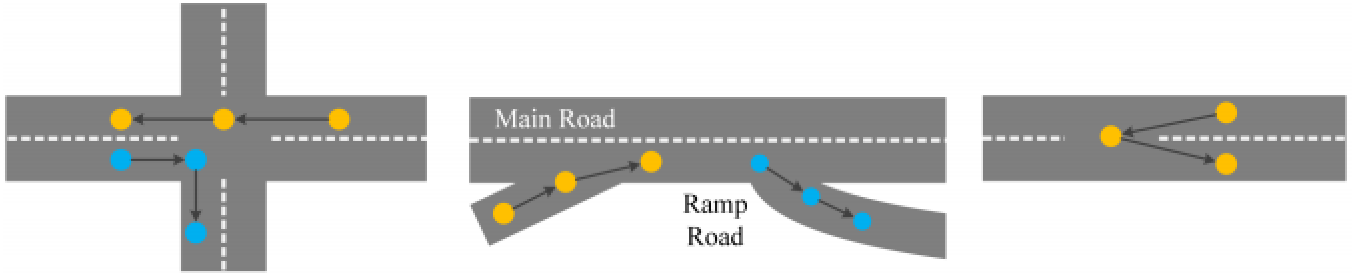
Objective:

Given: path (sequence of locations), start time, driver(optional)

Estimate: the travel time

Background:

Estimating the travel time in a city is of great importance to *traffic monitoring*, *route planning*, *ridesharing*, *taxi/Uber dispatching*, etc.



(a) Go straight (yellow) or   (b) Drive into main road (yellow)   (c) Turn around

Data:

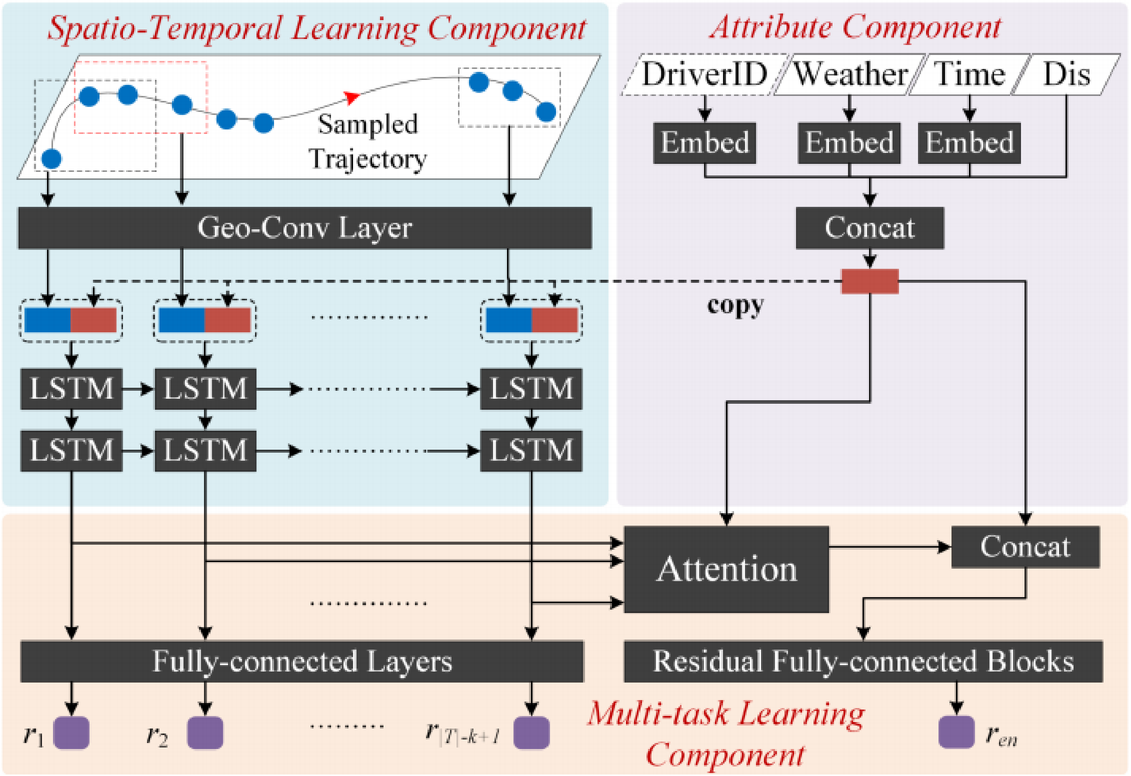
GPS trajectory: a sequence of GPS points

GPS point: latitude, longitude, timestamp, driver ID(optional).

Sample a GPS point each 200m ~ 400m

We use the timestamp as the **ground truth**.

DeepTTE Architecture



1. Spatio-temporal Learning Component:
   - Geo-Conv Layer: Capture spatial correlations
   - Recurrent Layer: Learn temporal dependencies
2. Attribute Component:
   - Handle external factors & share similar pattern
3. Multi-task Learning Component:
   - Address data sparsity problem
   - Get a better estimation result

Multi-task Learning Component

✓ Estimate the local path:

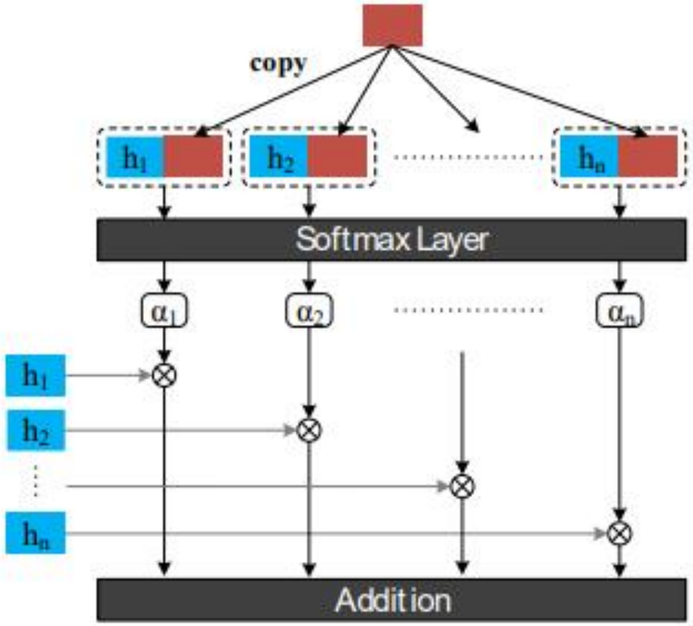
Spatio-temporal feature sequence of local paths: $\{h_1, h_2, \ldots, h_{|T|-k+1}\}$.

✓ Estimate the entire path:

$$h_{att} = \sum_{i=1}^{|T|-k+1} \alpha_i \cdot h_i$$

Attention Mechanism:

- Local path with more intersections or in extremely congested need more attention.
- Learn weights for different local path

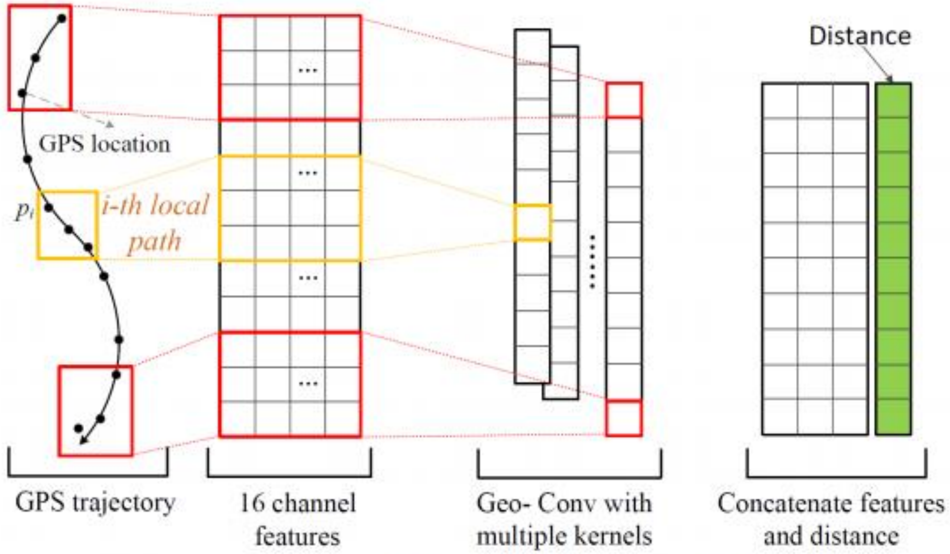


Training Loss: $\beta \cdot L_{local} + (1-\beta) \cdot L_{en}$

Spatio-temporal Learning Component

Geo-Conv Layer

Transforms the raw GPS sequence to a series of feature maps for each local paths.



$$loc_i = \tanh(W_{loc} \cdot [p_i.lat \circ p_i.lng]) \qquad (1)$$

$$loc_i^{conv} = \sigma_{cnn}(W_{conv} * loc_{i:i+k-1} + b) \qquad (2)$$

Recurrent Layer

To further capture the temporal dependencies among these local paths, we introduce the recurrent layers in our model.

$$h_i = \sigma_{rnn}(W_x \cdot loc_i^f + W_h \cdot h_{i-1} + W_a \cdot attr) \qquad (3)$$

Chengdu dataset: consists of *9,737,557 trajectories* (1.4 billion GPS records) of 14,864 taxis in August 2014 in Chengdu, China.
Beijing dataset: consists of 3,149,023 trajectories (0.45 billion GPS records) of 20,442 taxis in April 2015 in Beijing, China.
* *For Beijing Dataset, we further collected the corresponding weather conditions (16 types including sunny, rainy, cloudy etc.) as well as the road ID of each GPS point.*
* We use the last 7 days in each datasets as the test set.

	Chengdu		Beijing	
	MAPE (%)	RMSE (sec)	MAPE (%)	RMSE (sec)
AVG	28.1	533.57	24.78	703.17
D-TEMP	22.82	441.50	19.63	606.76
GBDT	19.32 ± 0.04	357.09 ± 2.44	19.98 ± 0.02	512.96 ± 3.96
MlpTTE	16.90 ± 0.06	379.39 ± 1.94	23.73 ± 0.14	701.61 ± 1.82
RnnTTE	15.65 ± 0.06	358.74 ± 2.02	13.73 ± 0.05	408.33 ± 1.83
DeepTTE	$\mathbf{11.89 \pm 0.04}$	$\mathbf{282.55 \pm 1.32}$	$\mathbf{10.92 \pm 0.06}$	$\mathbf{329.65 \pm 2.17}$

Tab: Performance Comparison

TEMP [1] is the state-of-art collective estimation method. However, there are about 10% paths that the original TEMP method can not estimate due to the lack of neighbor trajectories. We refine it as D-TEMP.

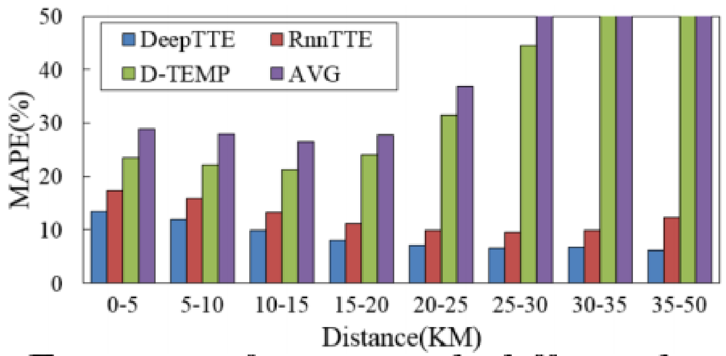


Fig: Error rates for trajs with different lengths

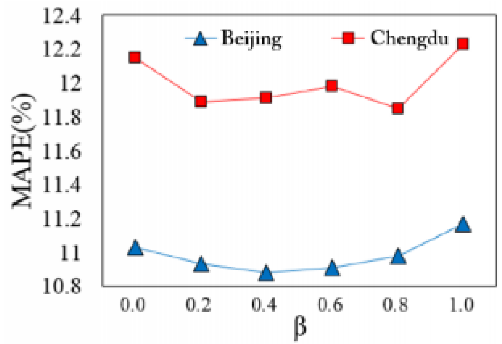


Fig: Error rates for different β

Thanks

Some materials from http://ruder.io/multi-task-learning-nlp/
Some from Sinno Jialin Pan's slides