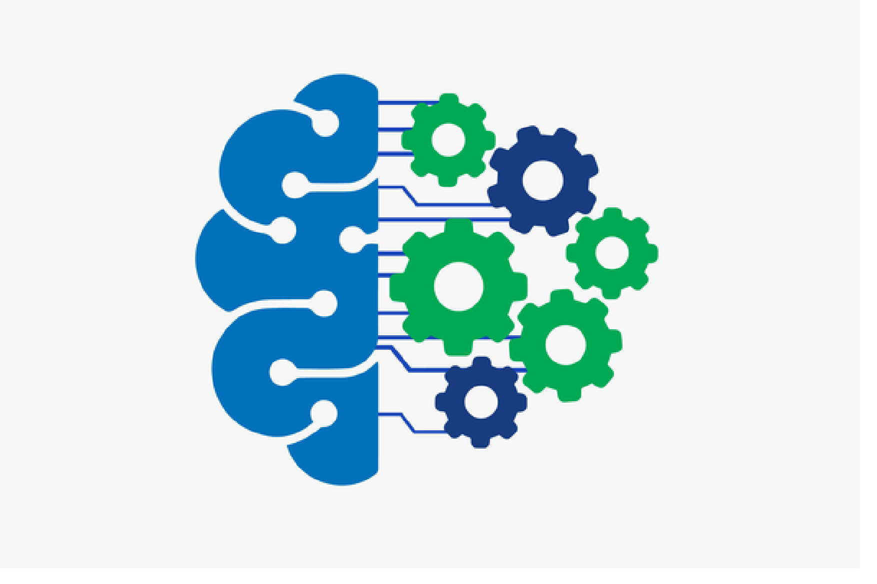
Learning Concepts

Narada Warakagoda



Introduction

- What is learning?
  - Adjust model parameters according to data
- Why is it challenging?
  - We do not have enough data
  - Learning takes time
  - Learning demands computational resources
- What is the remedy?
  - Learn efficiently with as little data as possible

Learning buzzwords

Deep Learning

Supervised learning
Unsupervised learning
Reinforcement learning

Transfer learning

Semi-supervised learning
Curriculum learning

Self-supervised learning

Self taught learning
Machine Learning
Hebbian learning

Active learning
Metric learning
Feature learning

Multi-task learning
Multi-modal learning

Zero-shot learning
Few-shot learning
One-shot learning

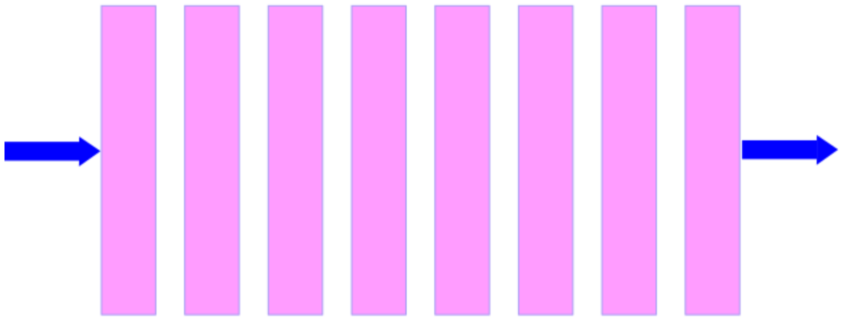
Meta learning

Transfer Learning

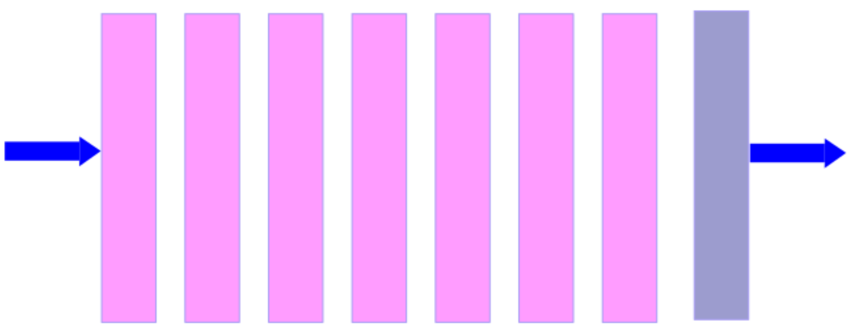
- General definition:
  - Using the knowledge gathered in one learning task in another learning task
- In deep learning:
  - Re-train a previously trained network with new data
- Advantage:
  - Can train a large network with relatively small (new) data set.

Possible Approaches

Original pretrained

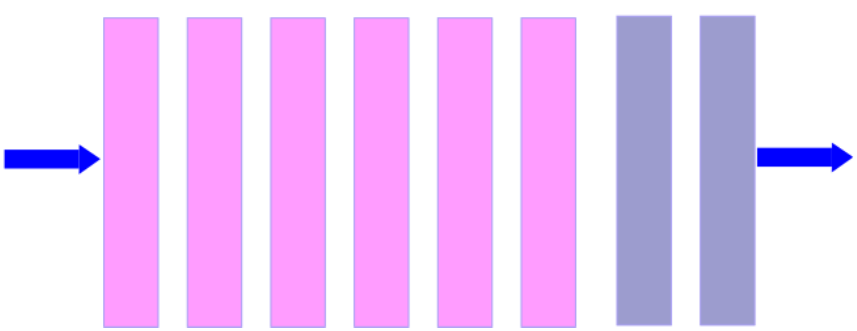


Transfer learning last layer



-New data set is very small
-New data and original data from the same domain
-Different class structure

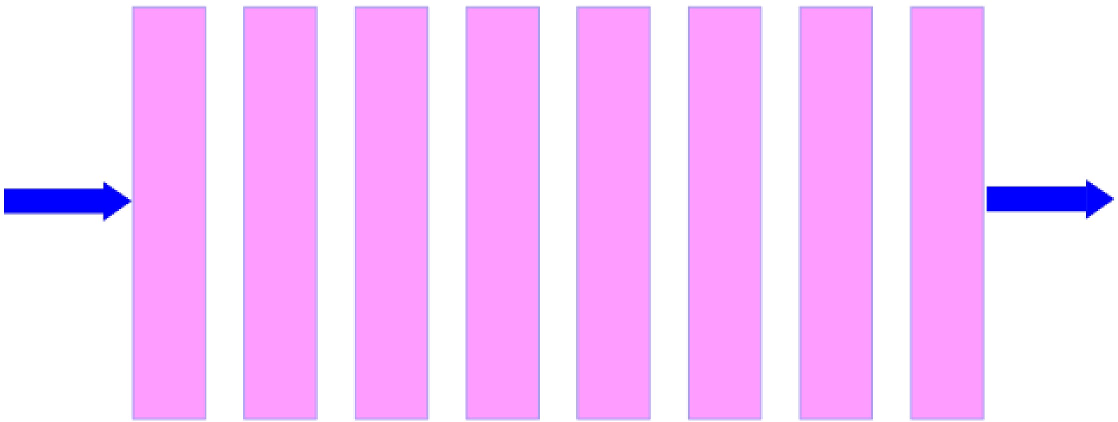
Transfer learning last two layers



-New data set is less small
-Same domain, different class structure

Possible Approaches

Original pretrained



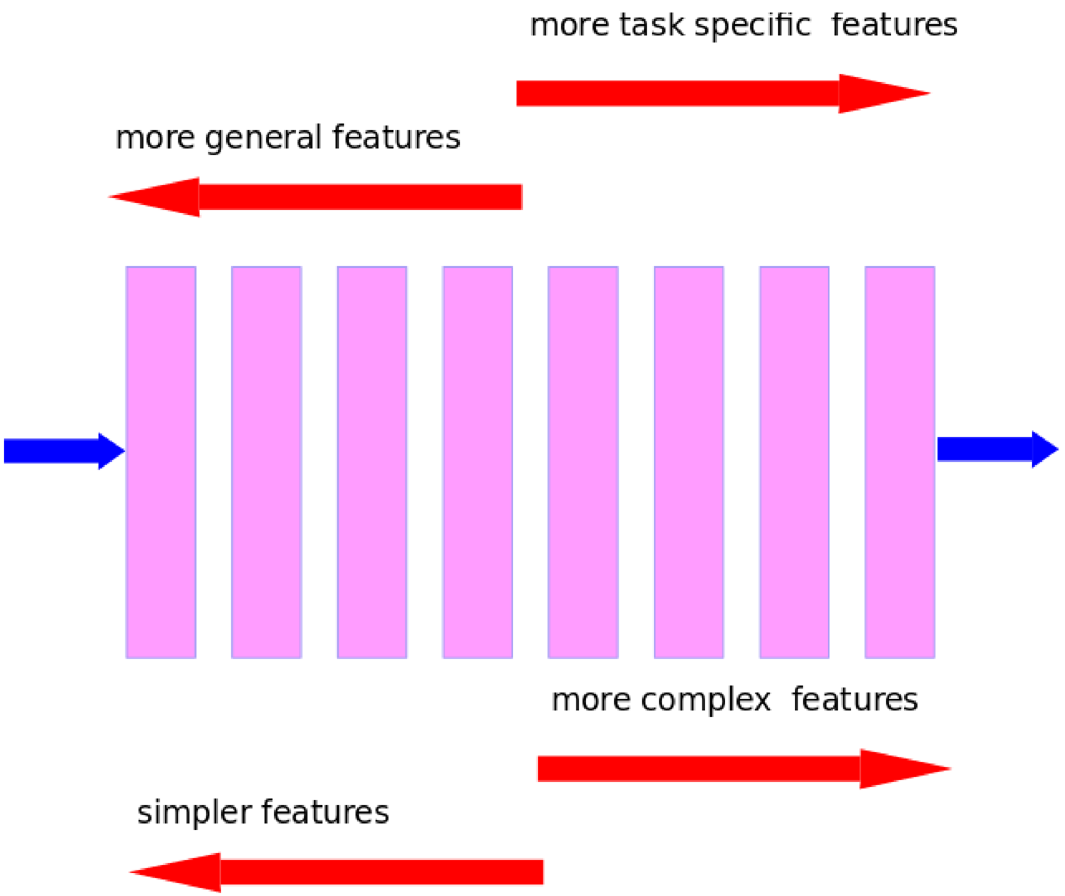
Transfer learning all layers

New data is from a different domain than original data.

Eg: Optical images and sonar images.

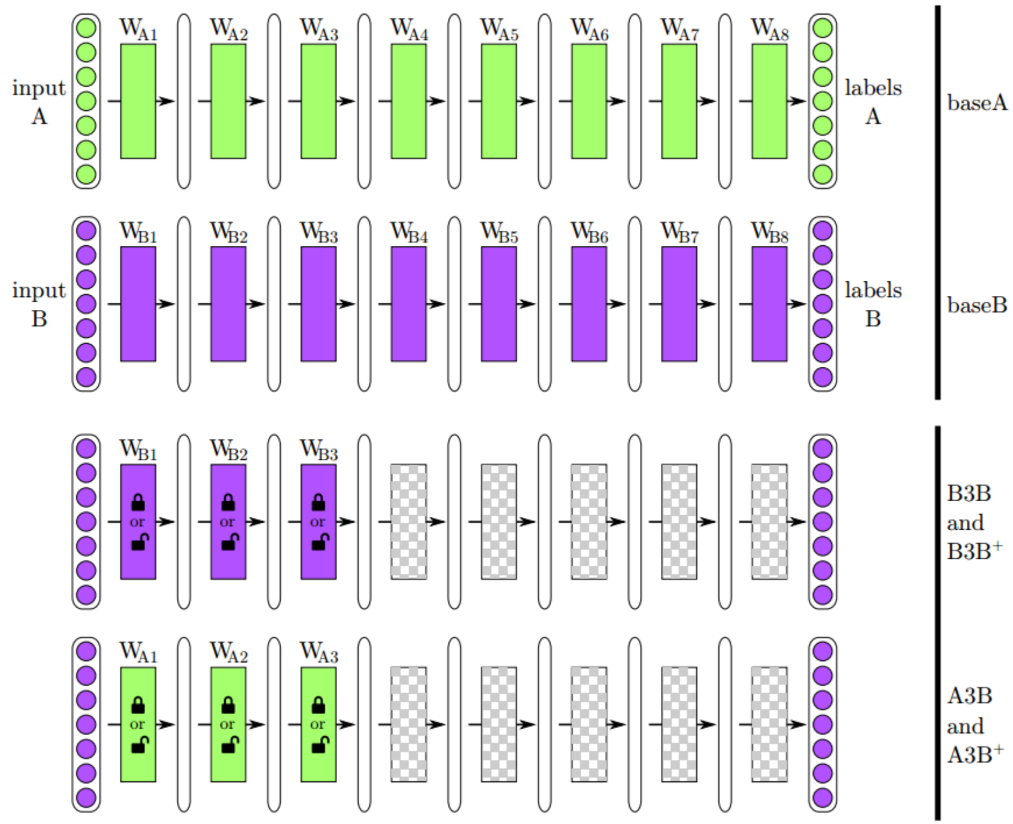
Why is Transfer Learning Possible?

- The network learns general feature representations
- More general, more transferable?

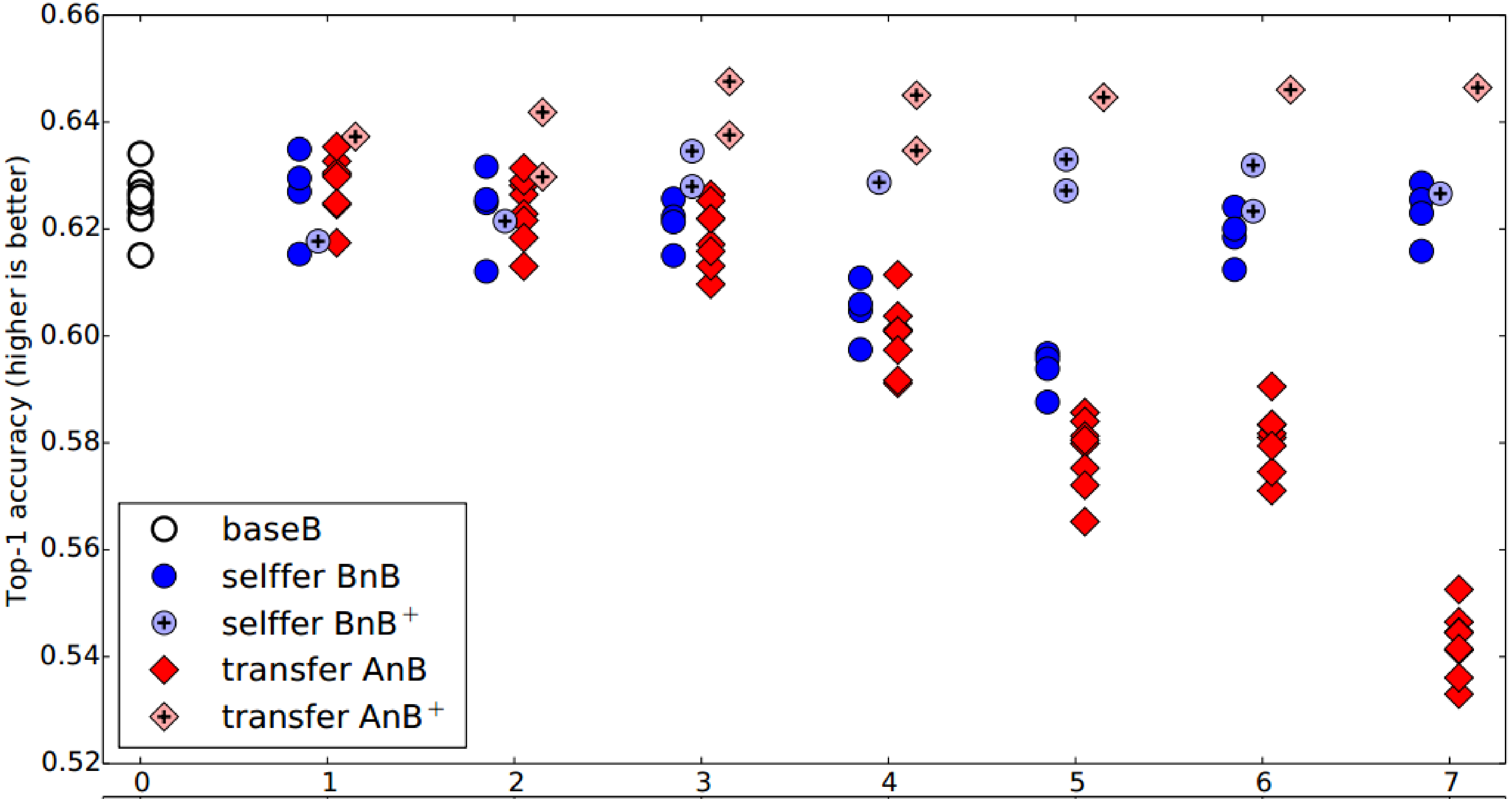


How Transferable are Features?

- Experiment reported in *How transferable are features in deep neural networks?, Yosinkski et.al 2014*
  - Two pre-trained networks baseA and baseB
  - Same architecture trained with data sets A and B
- AnB = First n layers from baseA, rest are retrained with data set B
  - AnB+ = Same approach but the whole network is re-trained
- BnB= First n layers from baseB, rest are retrained with data set B
  - BbB+ = Same approach but the whole network is retrained

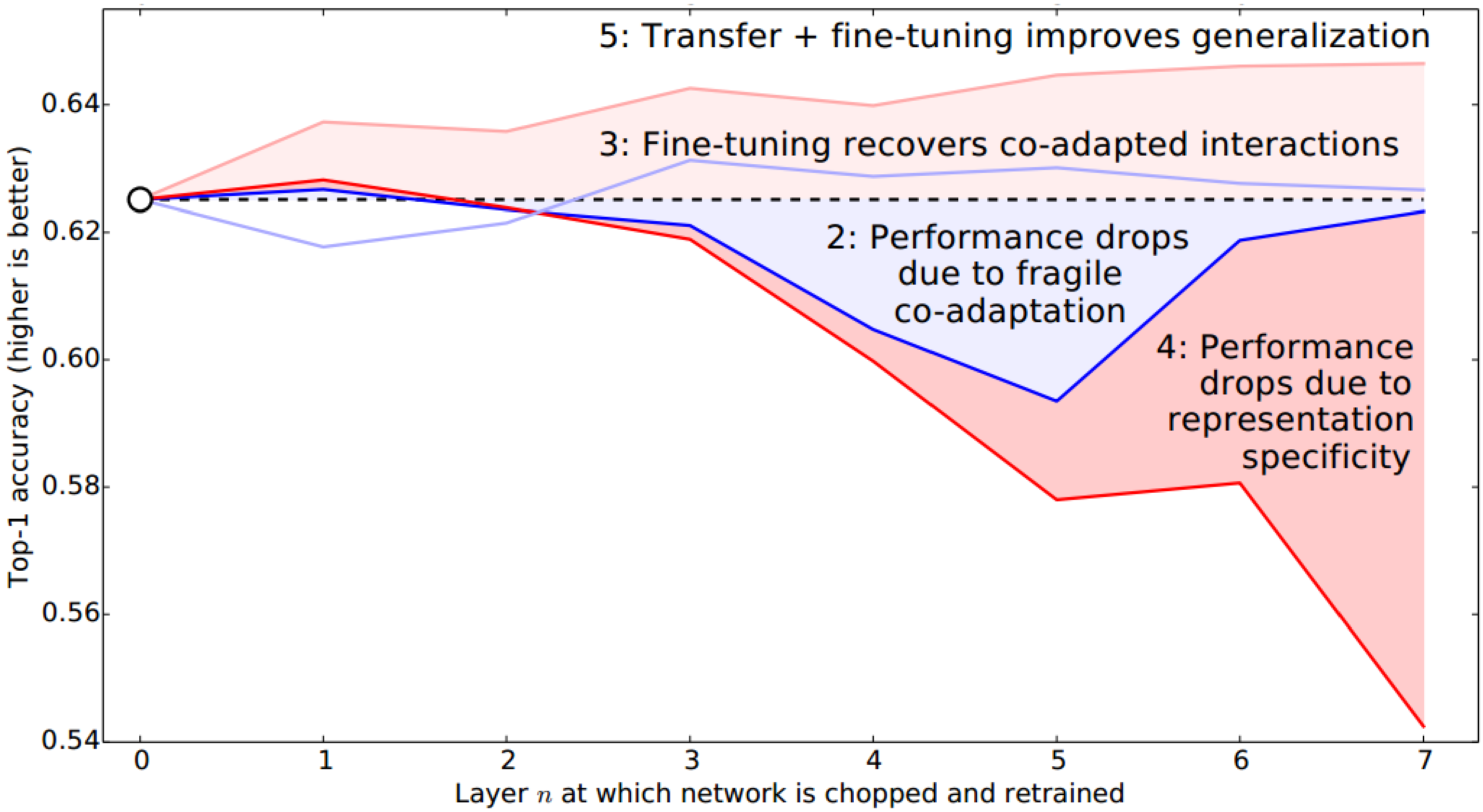


How Transferable are Features?



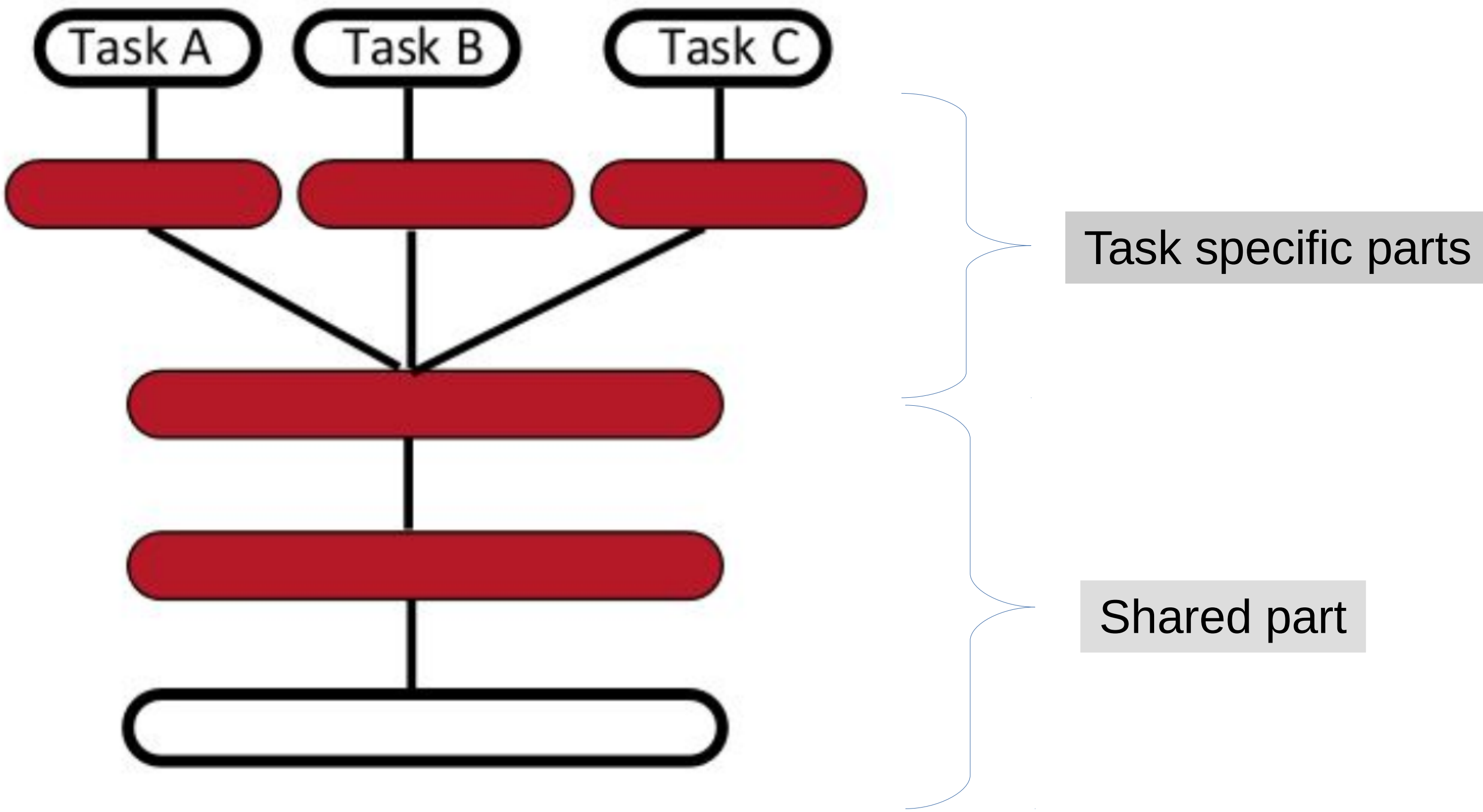
How Transferable are Features?

- C
  - –
  - –

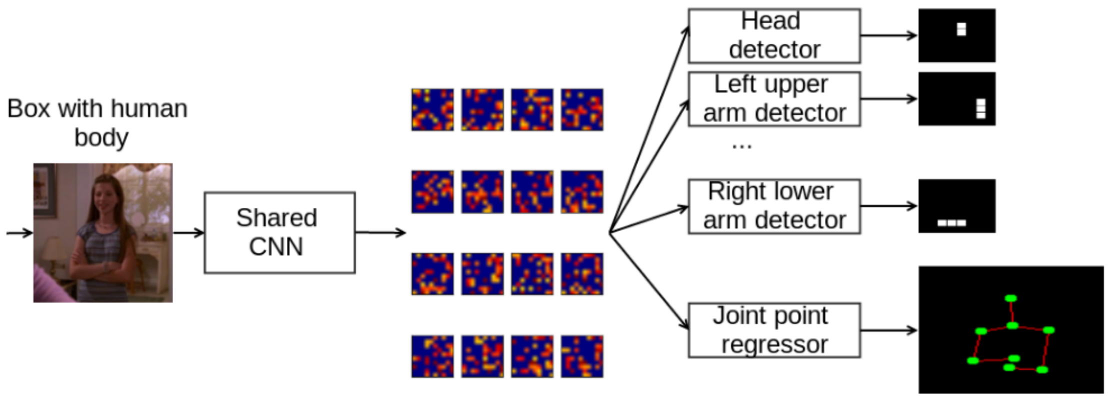


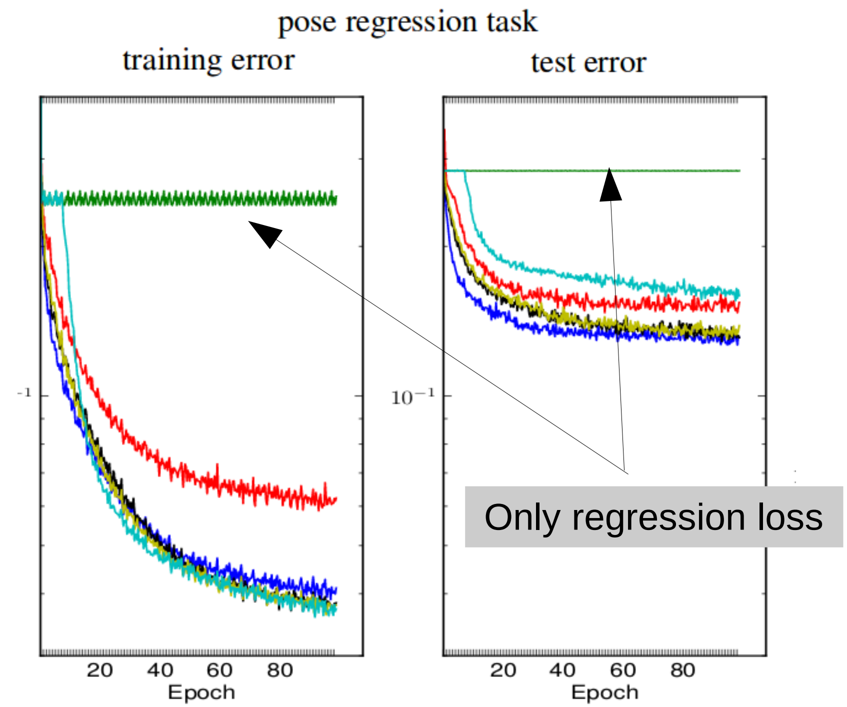
Multi-task Learning

- Sharing a single network for several different tasks
- Enhanced regularization effect



Multi-task Learning Example



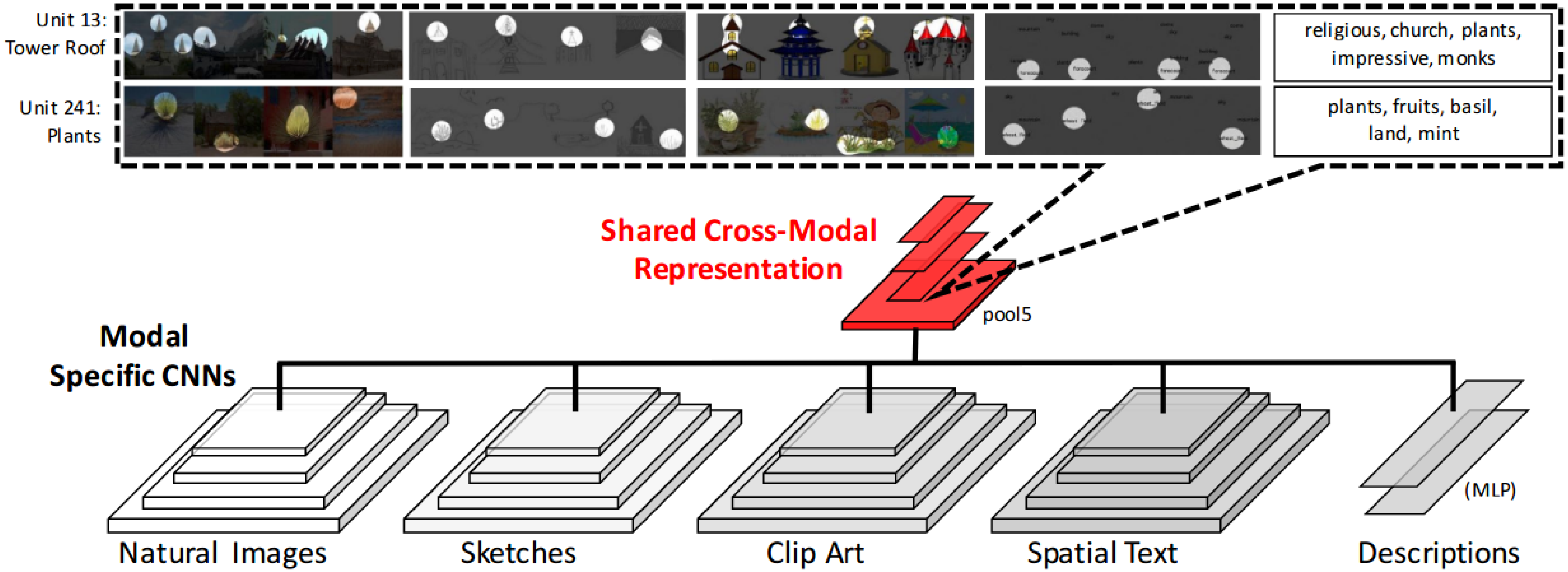


- Loss function $L_{\text{tot}} = \lambda_r L_{\text{regression}} + \lambda_d L_{\text{detection}}$
- Detection task helps regression task

Heterogeneous Multi-task Learning for Human Pose Estimation with DeepConvolutional Neural Network, S. Li et.al 2014

Multi-modal (Cross-modal) Learning

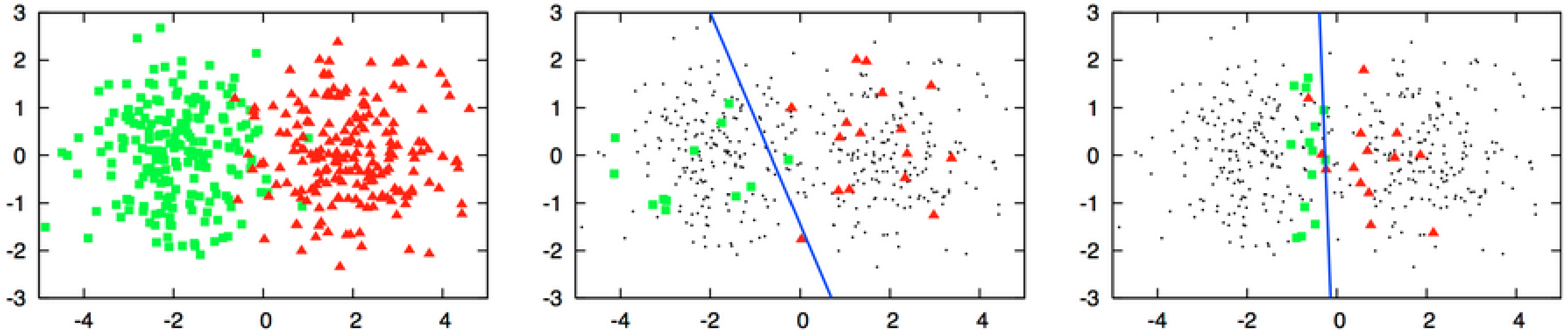
- Several input types sharing a single task



Cross-Modal Scene Networks, Aytar et.al 2016

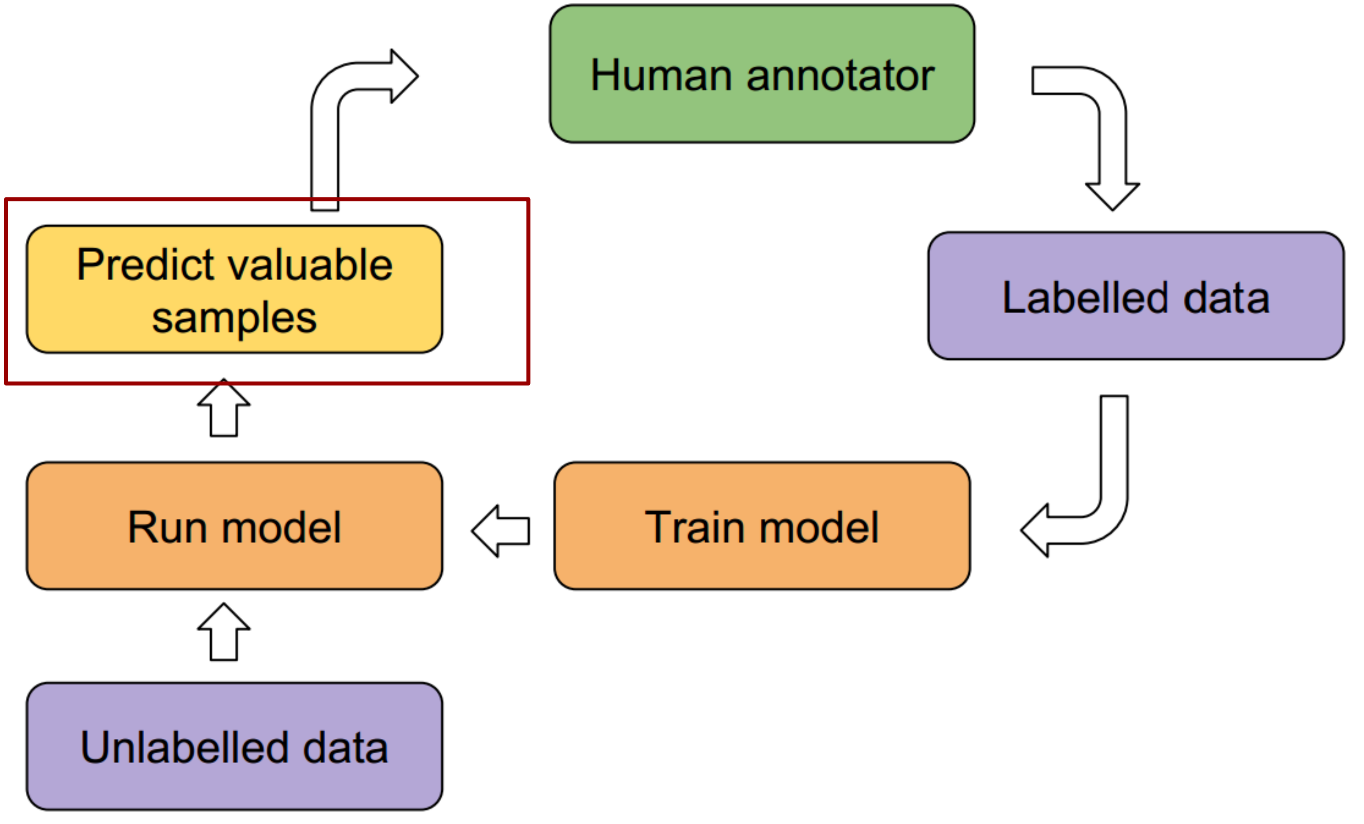
Active Learning

- Actively choose the samples in training



https://www.datacamp.com/community/tutorials/active-learning, Stefan Hosein

Active Learning Typical Procedure



- The most important element of active learning is to predict which samples are the most “valuable” ones

Predicting the "Valuable" Samples

- Learner can learn more from samples that are difficult to classify now.
- Measures to identify difficult-to-classify samples
  - Confidence $x_{\text{lowest confidence}} = \arg\min_x p(y_1|x)$
  - Classification margin $x_{\text{lowest margin}} = \arg\min_x |p(y_1|x) - p(y_2|x)|$
  - Entropy $x_{\text{highest entropy}} = \arg\max_x - \sum_i p(y_i|x) \log p(y_i|x)$
  - Variance
    - Bayesian techniques (eg: Monte-Carlo dropout, ensembles)

$$p(y_1|x) > p(y_2|x) > \cdots > p(y_N|x)$$

Self-supervised Learning

- Related to Transfer learning
- How to get a pre-trained model?
  - Alt1: Use an existing big labeled database (eg: ImageNet) and supervised learning
  - Alt2: Use own unlabeled data and self-supervised learning
- Supervised vs Self-supervised learning
  - Supervised:

Manually created labels

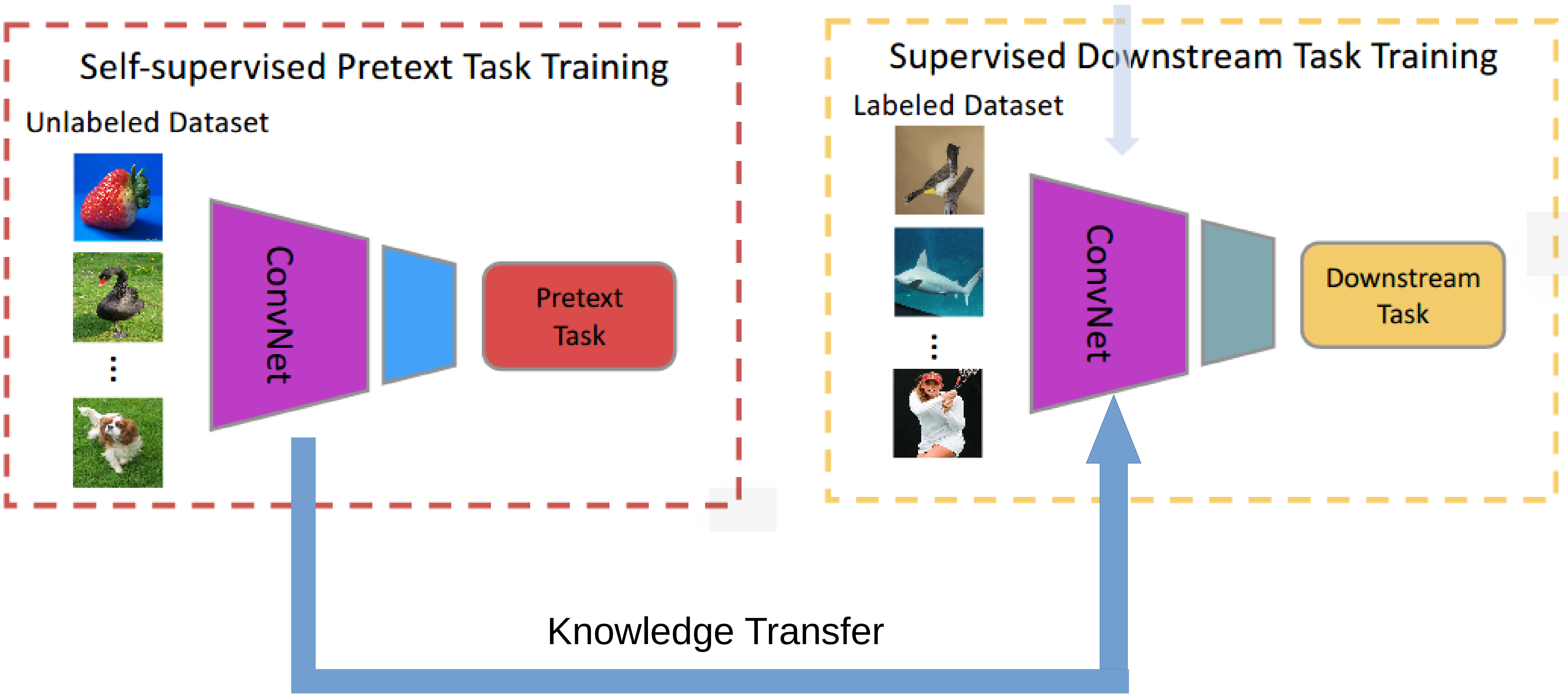
$$loss(D) = \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} loss(X_i, Y_i).$$

  - Self-Supervised

Automatically created labels

$$loss(D) = \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} loss(X_i, P_i).$$

Pretext and Downstream Tasks

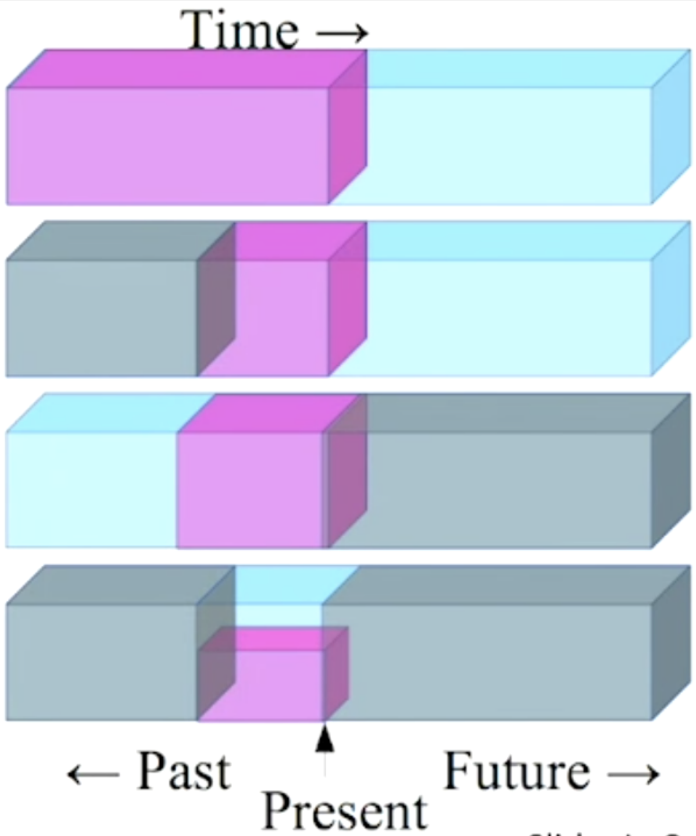


Pretext task is defined so that automatic generation of labels is possible

Self-supervised Visual Feature Learning withDeep Neural Networks: A SurveyLonglong Jing and Yingli Tian, 2019

Pretext Task Approaches

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible
- Pretend there is a part of the input you don't know and predict that.

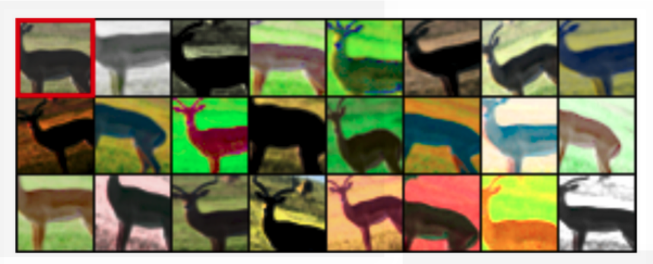


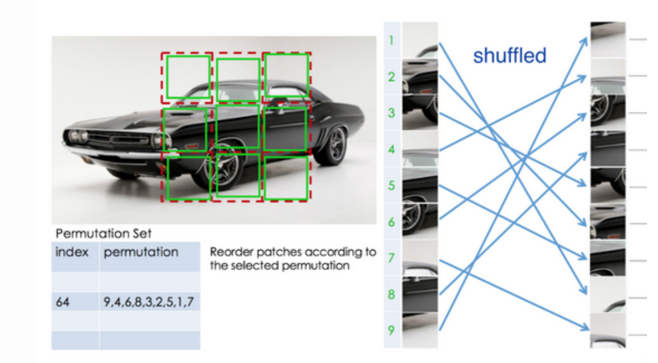
Slide: LeCun

https://www.youtube.com/watch?v=7I0Qt7GALVk

Examples

- Eg 1 (ULMFiT):
  - Predict the next word given the previous words (pretext)
  - Predict sentiments for given sentences (down-stream)
- Eg 2: (BERT)
  - Predict a randomly masked word in a sentence (Pretext)
  - Question answering, language inference etc. (down-stream)
- Eg 3: (Exemplar-CNN)
  - Predict the class of distorted images (Pretext)
  - Image classification (down-stream)
- Eg 4: (JigSaw Puzzle)
  - Predict the correct order of image patches extracted using a 3x3 grid. (Pretext)
  - Image retrieval (Down-stream)
- Eg 5: (Colorization)
  - Predict the Color given the grayscale image (Pretext)
  - Image classification, detection and segmentation (down-stream)
- And many more





https://lilianweng.github.io/lil-log/2019/11/10/self-supervised-learning.html
https://amitness.com/2020/05/self-supervised-learning-nlp/
Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks, A. Dosovitskiy et.al , 2015
Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles, Mehdi Noroozi, Paolo Favaro, 2017

Metric Learning

- Learn a distance measure (metric) which reflects the semantic similarity
- In practice, we learn a transform $\mathrm{f}(\cdot)$ such that the Euclidean distance reflects the similarity.

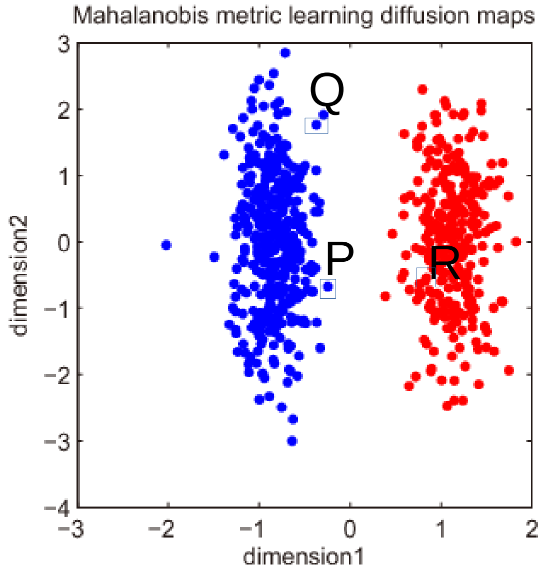
$$\|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)\|^2 = [f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)]^T [f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)]$$

- “Similarity” is expressed through side information
  - Eg1: x_i and x_j are similar if they can be linked (i.e. same class) and vice versa
  - Eg2: x_a is more similar to x_p than x_n if a and p can be linked while a and n cannot be linked
- A loss function can be defined for extracting similarity. Examples:
  - Contrastive loss:

$$L_c = \begin{cases} ||f(\mathbf{x}_i) - f(\mathbf{x}_j)||^2, & \text{if } \mathbf{x}_i, \mathbf{x}_j \in P \\ -||f(\mathbf{x}_i) - f(\mathbf{x}_j)||^2, & \text{if } \mathbf{x}_i, \mathbf{x}_j \in N \end{cases}$$

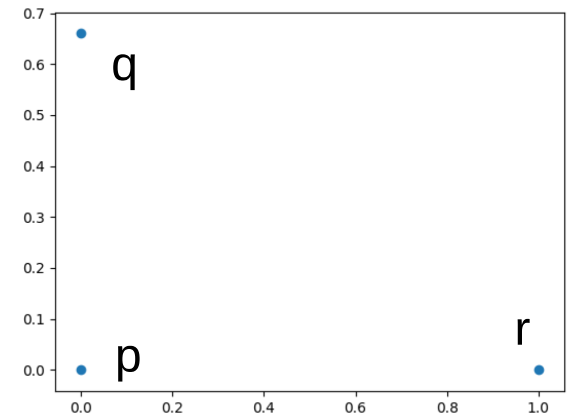
  - Triplet loss $\mathrm{L}_t = \sum_{\boldsymbol{x}_i \in A, \boldsymbol{x}_j \in P, \boldsymbol{x}_k \in N} \|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)\|^2 - \|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_k)\|^2$

Mahalanobis Metric (example)



Scaling

$$\hat{x}_1 = \frac{x_1}{\sigma_1}, \quad \hat{x}_2 = \frac{x_2}{\sigma_2}$$



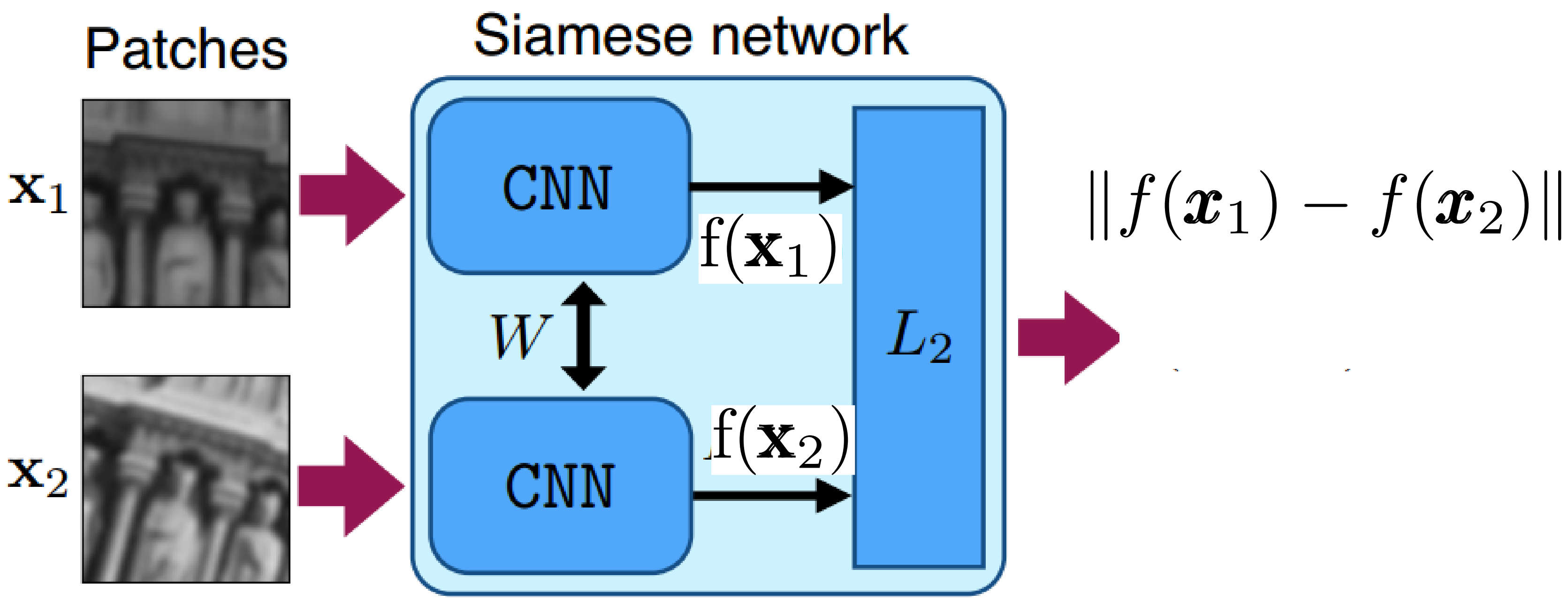
- Consider P, Q and R
  - Q should be more similar to P than R (P and Q belongs to the same class)
  - But Euclidean distance d(P,Q) > d(P,R),
  - Correct this by scaling variance along each dimension

Mahalanobis distance/metric

$$\begin{aligned} D_m(P,Q) &= \left\| \left[\frac{x_P(1)}{\sigma_1}, \frac{x_P(2)}{\sigma_2} \right] - \left[\frac{x_Q(1)}{\sigma_1}, \frac{x_Q(2)}{\sigma_2} \right] \right\| \\ &= \left\| [x_P(1), x_P(2)] \begin{bmatrix} \frac{1}{\sigma_1} & 0 \\ 0 & \frac{1}{\sigma_2} \end{bmatrix} - [x_Q(1), x_Q(2)] \begin{bmatrix} \frac{1}{\sigma_1} & 0 \\ 0 & \frac{1}{\sigma_2} \end{bmatrix} \right\| \\ &= \left\| \boldsymbol{x}_P^T \Sigma^{-\frac{1}{2}} - \boldsymbol{x}_Q^T \Sigma^{-\frac{1}{2}} \right\| \end{aligned}$$

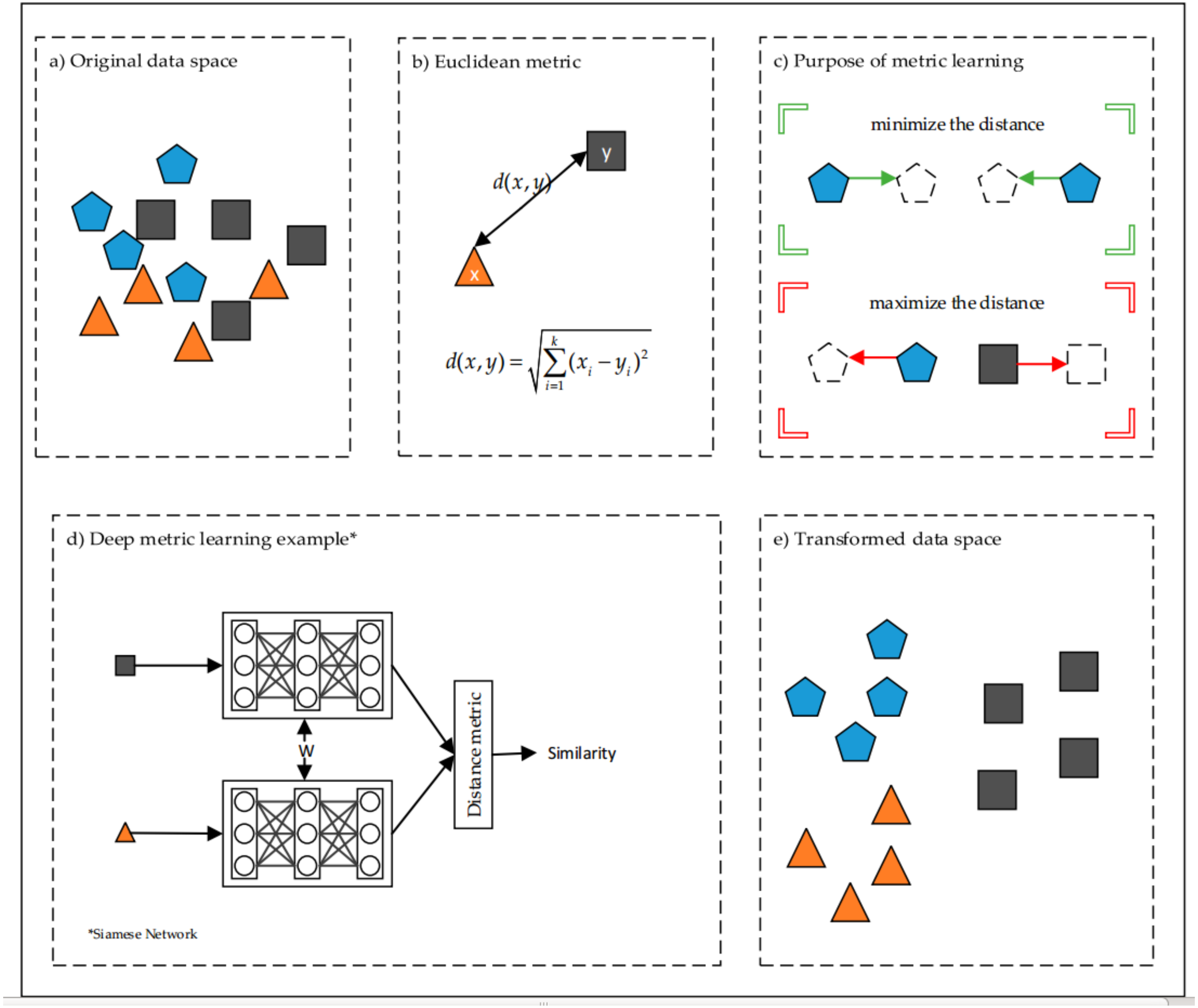
Metric Learning in Deep Learning

- Replace the linear transform with a neural net and train using a suitable loss



Discriminative Learning of Deep Convolutional Feature Point Descriptors Simo-Serra et.al 2014

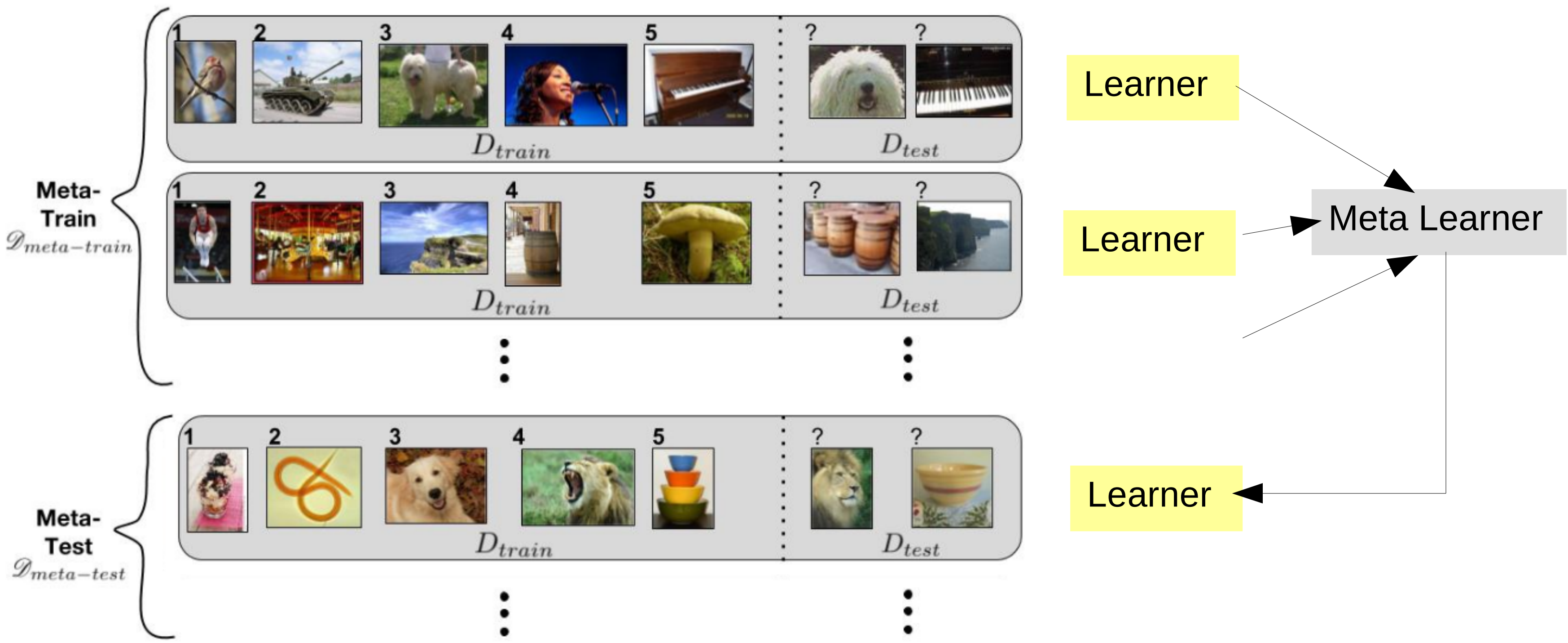
Metric Learning Summary



Deep metric learning, a survey, Kaya and Bilge, 2019

Meta Learning

- Also known as “Learning to Learn”
- It is about learning what a system learns in different tasks
- Then the meta learner can predict the system parameters for a new task (without going through full training).



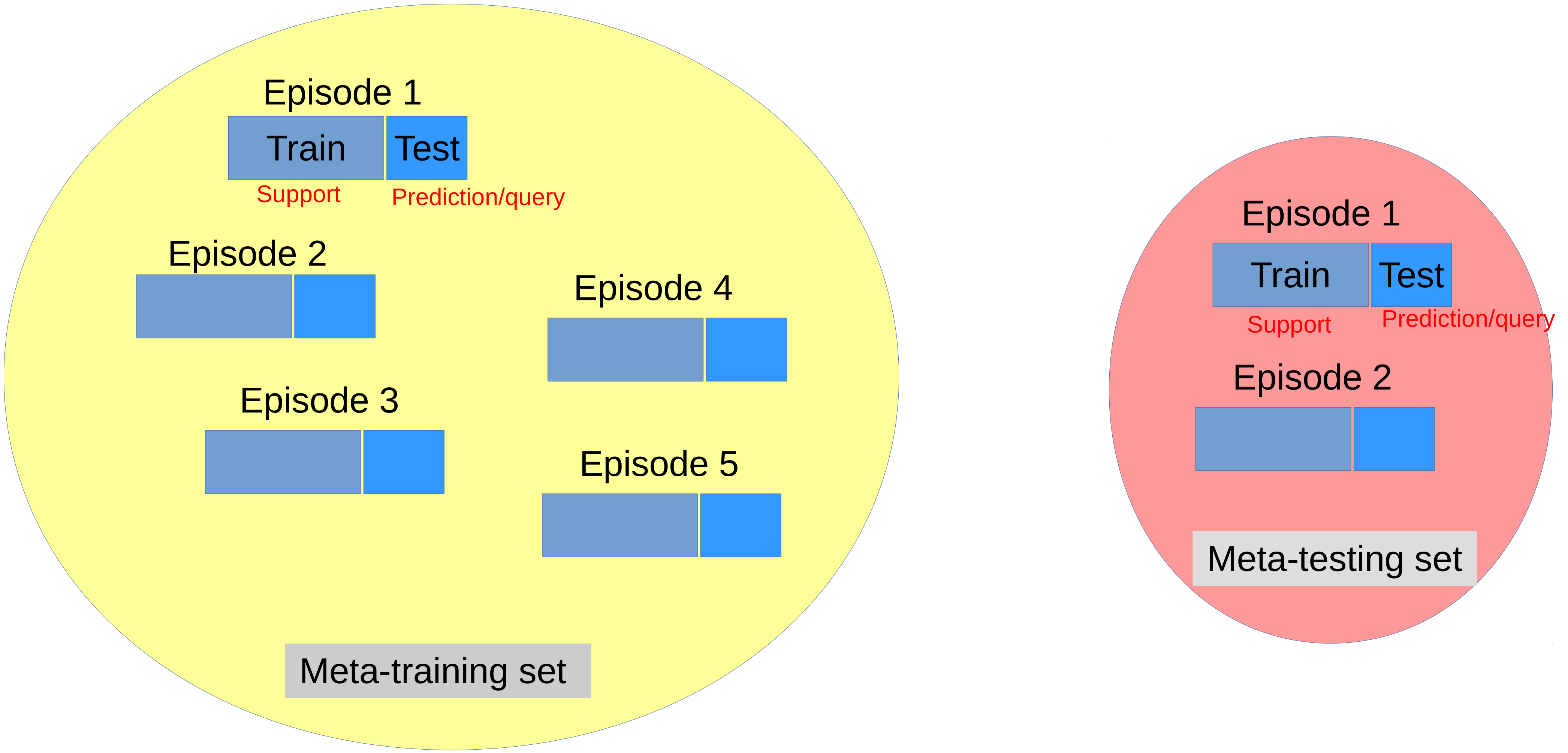
Generalizing from few examples with meta-learning Hugo Larochelle, 2018

Learning vs Meta-learning

- Learning algorithm (Learner)
  - **Input:** Training set $D_{\text{train}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$
  - **Output:** Parameters $\boldsymbol{\theta}$ of the model M
  - **Objective:** Performance on the test set $D_{\text{test}} = \{(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i)\}$

- Meta-Learning algorithm (Meta-learner)
  - **Input:** Meta-training set $D_{\text{meta-train}} = \{(D^{(n)}_{\text{train}}, D^{(n)}_{\text{test}})\}$
  - **Output:** Parameters $\boldsymbol{\Theta}$ of the learner
  - **Objective:** Performance on the meta-test set $D_{\text{meta-test}} = \{(\hat{D}^{(m)}_{\text{train}}, \hat{D}^{(m)}_{\text{test}})\}$

Terminology



Optimization Objectives

- Formulation 1

$$\theta^\star = \arg\max_\theta \frac{1}{N} \sum_{n=1}^{N} \sum_{\boldsymbol{x},\boldsymbol{y} \in D_{\text{test}}^{(n)}} p_\theta(\boldsymbol{y} | \boldsymbol{x}, D_{\text{train}}^{(n)})$$

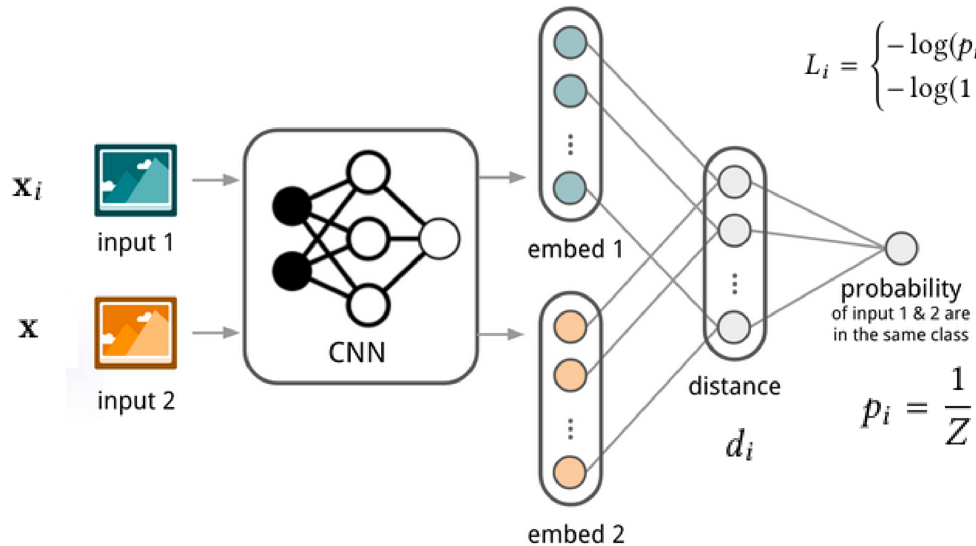
- Formulation 2

$$[\theta^\star, \phi^\star] = \arg\max_{\theta,\phi} \frac{1}{N} \sum_{n=1}^{N} \sum_{\boldsymbol{x},\boldsymbol{y} \in D_{\text{test}}^{(n)}} p_{g_\phi(\theta, D_{\text{train}}^{(n)})}(\boldsymbol{y} | \boldsymbol{x})$$

Possible Approaches

- Model dependent approaches
  - Metric based
    - Siamese networks
    - Matching network
    - Relation network
    - Prototypical network
  - RNN based
    - Memory augmented neural network (MANN)
    - LSTM meta learner
- Model independent approaches
  - Optimization based
    - Model Agnostic Meta-learning (MAML)
    - Reptile

Siamese networks



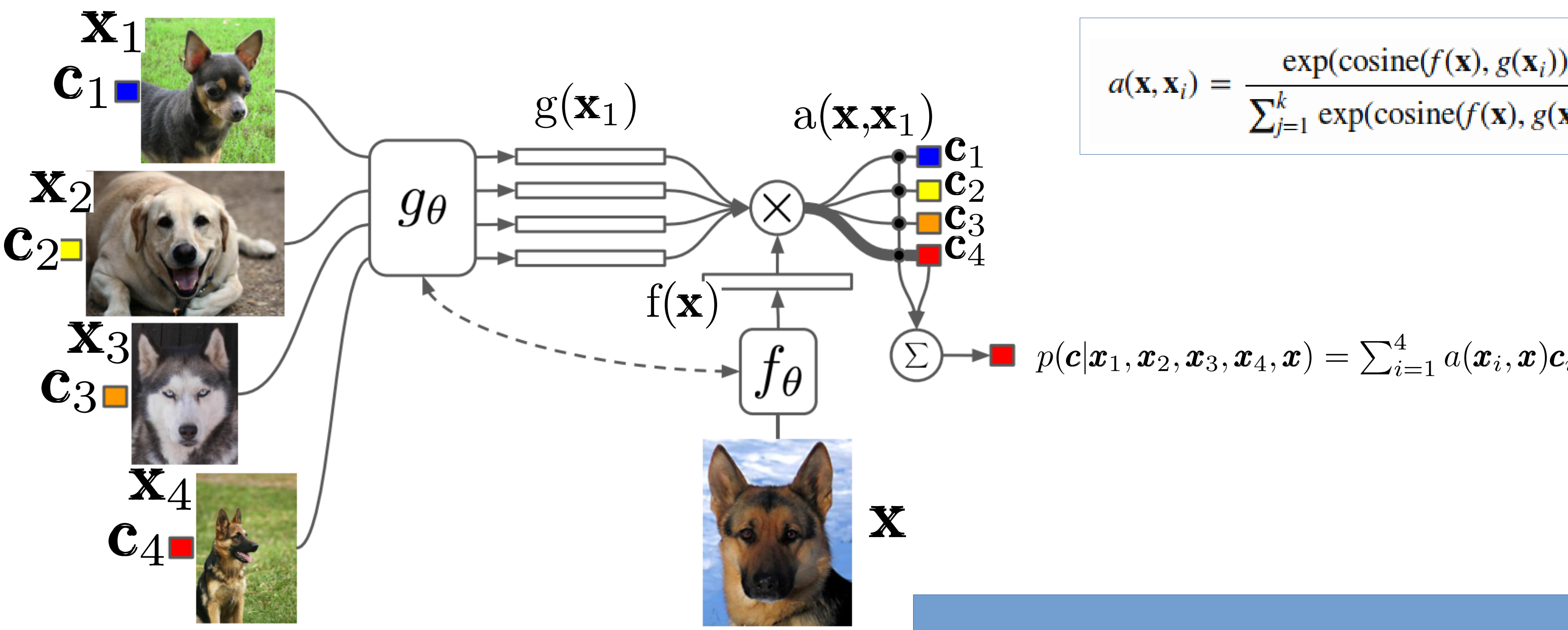
$$L_i = \begin{cases} -\log(p_i) & \text{if } (\mathbf{x}_i, \mathbf{x}) \text{ belongs to the same class} \\ -\log(1-p_i) & \text{if } (\mathbf{x}_i, \mathbf{x}) \text{ belongs to different classes} \end{cases}$$

$$p_i = \frac{1}{Z}\exp(-d_i)$$

- Training
  - Meta-training set
    - $D_{train} = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), (\mathbf{x}_3, c_3), \cdots, (\mathbf{x}_N, c_N)\}$
    - $D_{test} = \{\mathbf{x}, c\}$
  - Make image pairs $\{(\mathbf{x}_1, \mathbf{x}), (\mathbf{x}_2, \mathbf{x}), \cdots, (\mathbf{x}_N, \mathbf{x})\}$
  - Trained to optimize $L = \sum_i L_i$
- Testing
  - Meta-testing set
    - $\hat{D}_{train} = \{(\hat{\mathbf{x}}_1, \hat{c}_1), (\hat{\mathbf{x}}_2, \hat{c}_2), (\hat{\mathbf{x}}_3, \hat{c}_3), \cdots, (\hat{\mathbf{x}}_N, \hat{c}_N)\}$
    - and a test image $\hat{\mathbf{x}}$
  - Make a set of image pairs $\{(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}), (\hat{\mathbf{x}}_2, \hat{\mathbf{x}}), \cdots, (\hat{\mathbf{x}}_N, \hat{\mathbf{x}})\}$
  - Feed each image pair and register the output probability. Class of support set image which outputs the highest probability is the class of the test image. $c = \arg\max_i p(\hat{\mathbf{x}}, \hat{\mathbf{x}}_i)$

Siamese Neural Networks for One-shot Image Recognition, G. Koch et. al 2015

Matching network



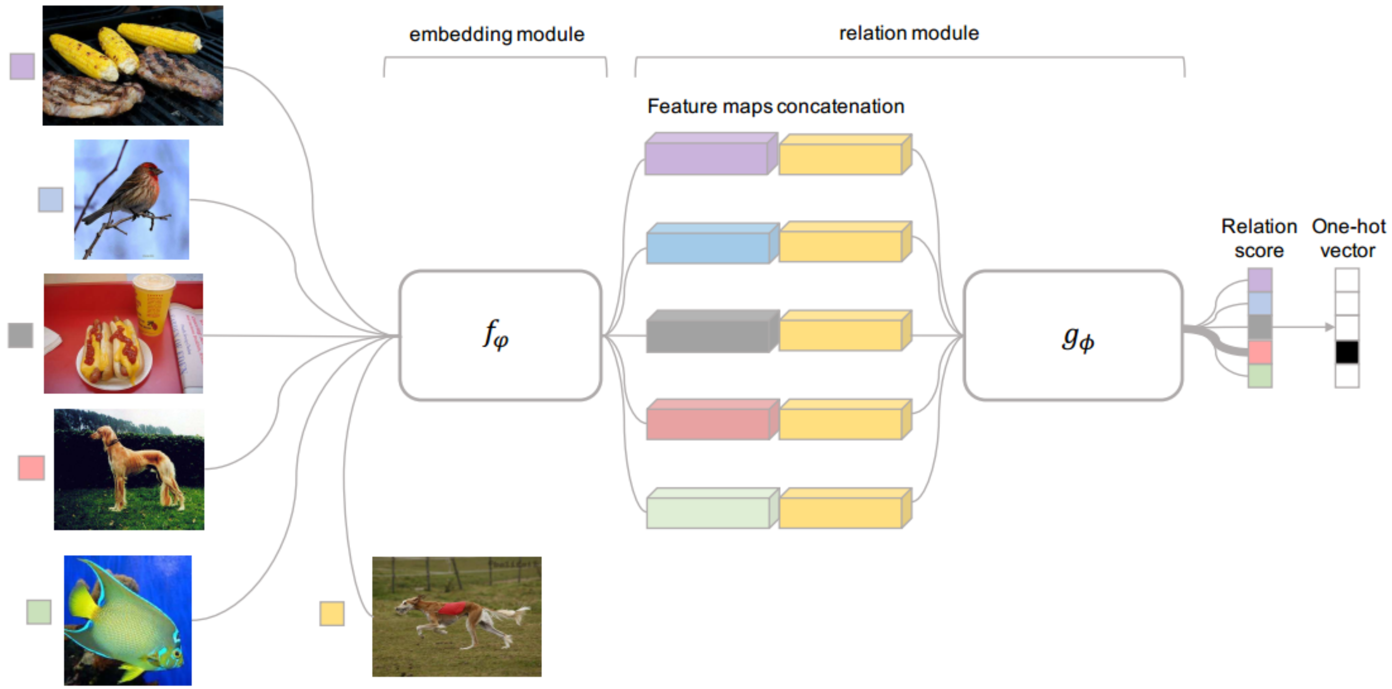
$$a(\mathbf{x}, \mathbf{x}_i) = \frac{\exp(\text{cosine}(f(\mathbf{x}), g(\mathbf{x}_i)))}{\sum_{j=1}^{k} \exp(\text{cosine}(f(\mathbf{x}), g(\mathbf{x}_j)))}$$

Note: Labels are in one-hot encoding format

- Training
  - Meta-training set
    - $D_{\text{train}} = \{(\boldsymbol{x}_1, \boldsymbol{c}_1), (\boldsymbol{x}_2, \boldsymbol{c}_2), (\boldsymbol{x}_3, \boldsymbol{c}_3), (\boldsymbol{x}_4, \boldsymbol{c}_4)\}$
    - $D_{\text{test}} = \{(\boldsymbol{x}, \boldsymbol{c})\}$
    - Train to maximize $p(\boldsymbol{c}|\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}) = \sum_{i=1}^{4} a(\boldsymbol{x}_i, \boldsymbol{x})\boldsymbol{c}_i$
- Testing
  - Meta-testing set
    - $\hat{D}_{\text{train}} = \{(\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{c}}_1), (\hat{\boldsymbol{x}}_2, \hat{\boldsymbol{c}}_2), (\hat{\boldsymbol{x}}_3, \hat{\boldsymbol{c}}_3), (\hat{\boldsymbol{x}}_4, \hat{\boldsymbol{c}}_4)\}$
    - $\hat{D}_{\text{test}} = \{(\hat{\boldsymbol{x}}, \hat{\boldsymbol{c}})\}$
    - Find $\arg\max_k c_k \in p(\hat{\boldsymbol{c}}|\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}_k \; k = 1, 2, 3, 4)$

Matching networks for one shot learning. Oriol Vinyals et .al 2016

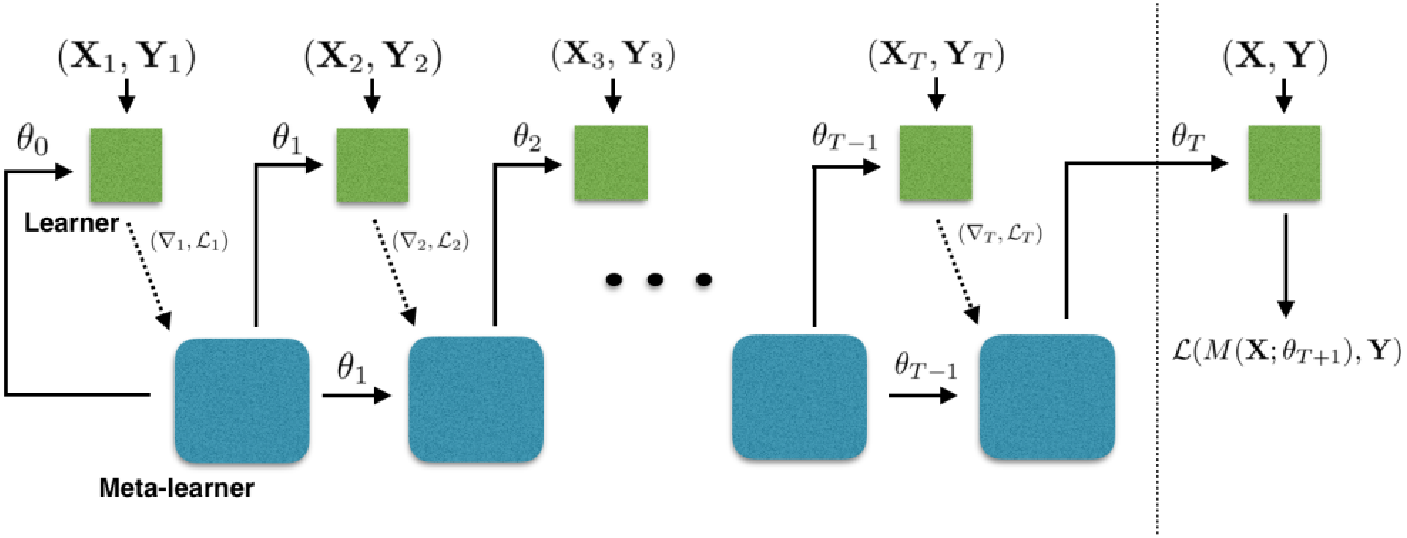
Relation network



- Idea is similar to matching network
- Similarity is learned with a different network architecture

Learning to compare: Relation network for few-shot learning, Flood Sung et al., 2018

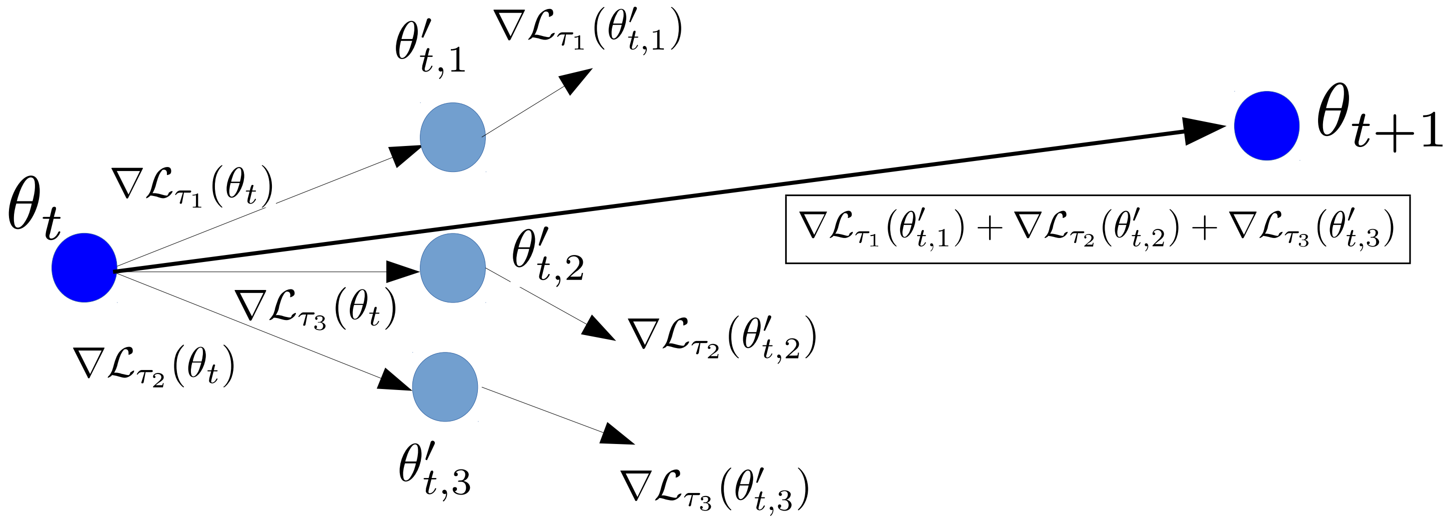
LSTM Meta-learner



- Learner is a neural network with parameters θ
- Meta-learner is an LSTM with parameters ϕ
- Data pairs from D_{train} are fed sequentially
- At each iteration, meta-learner delivers better parameter set θ_{t+1} given previous parameters θ_t, learner loss $\mathcal{L}_t$ and its gradient $\nabla\mathcal{L}_t$.
- Finally feed data from D_{test} and find a loss
- Loss is back-propagated through the both learner and meta-learner and both θ and ϕ can be updated.

Optimization as a Model for Few-Shot Learning, Sachin Ravi and Hugo Larochelle, 2017.

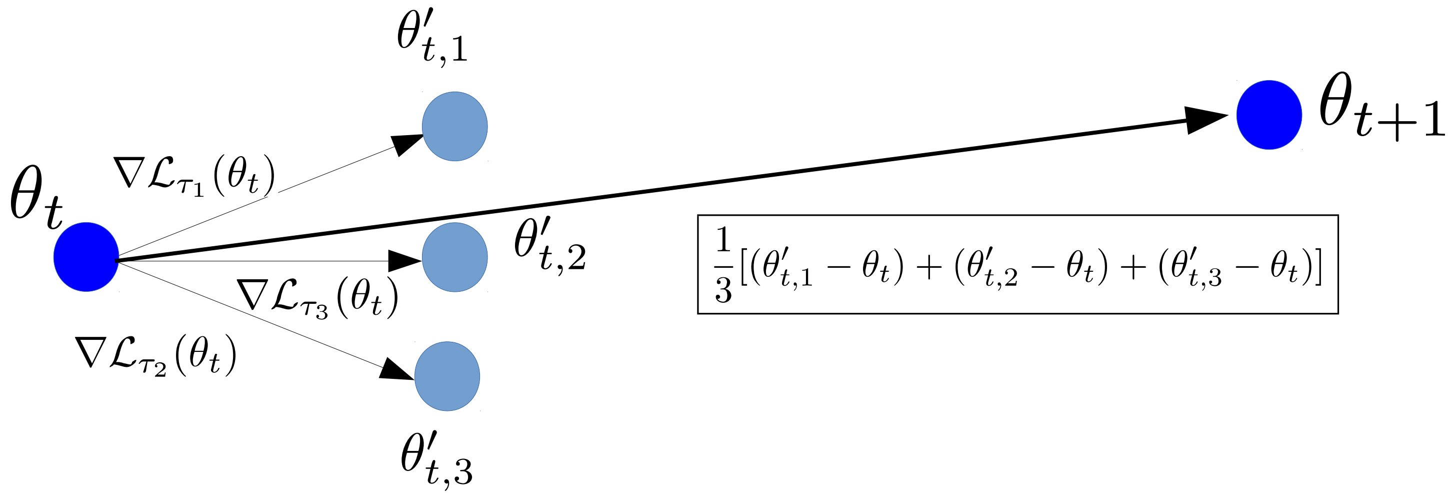
Model Agnostic Meta Learning (MAML)



- Optimization approach related to gradient descent
- Can be applied to any model
- Main idea:
  - Divide the meta-training set D_{train} into a set of tasks $\tau_i, \quad i = 1, 2, \cdots, N$
  - Update the model parameters, using the average of gradients over the tasks
  - The updated model parameter set is kept close to optimum model parameter set for each individual task.

Model-agnostic meta-learning for fast adaptation of deep networks, Chelsea Finn et al. 2017.

Reptile



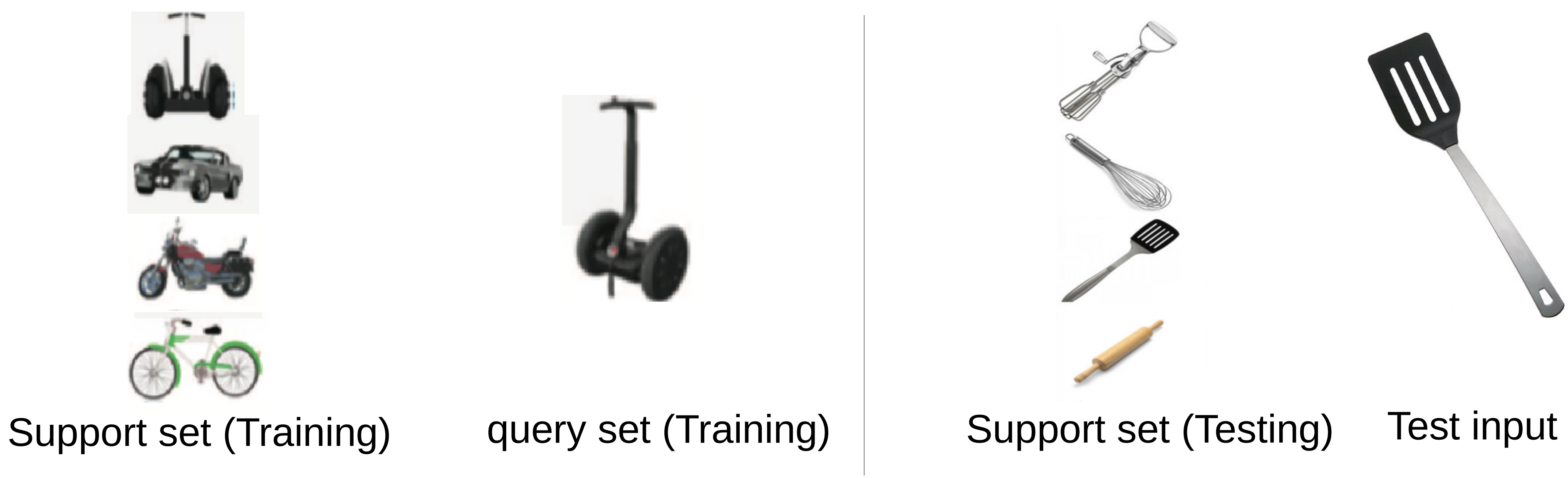
- Very similar to MAML
- Averaging strategy is different
- Idea is to minimize the Euclidean distance between θ_i and $\theta'_{t,i}$ $i = 1, 2, \cdots, N$

$$\nabla \sum_{i=i}^{N} (\theta_t - \theta'_{t,i})^2 = 2 \sum_{i=1}^{N} (\theta_t - \theta'_{t,i})$$

On First-Order Meta-Learning Algorithms. Alex Nichol et. al. 2018

One/Few Shot Learning

- Given a support set of one/few training examples of new classes, learn to classify a test input
  - K-class (k-way) one shot learning
    - one example per class and k classes
  - K-class (k-way) n-shot learning
    - n examples per class and k classes
- A straight-forward application of meta learning.



https://sorenbouma.github.io/blog/oneshot/

Zero Shot Learning

- Example: Given a semantic description, learn to classify an image

Okapi is ” zebra-striped four legged animal with a brown torso and a deer-like face”. Which of these images is of Okapi?

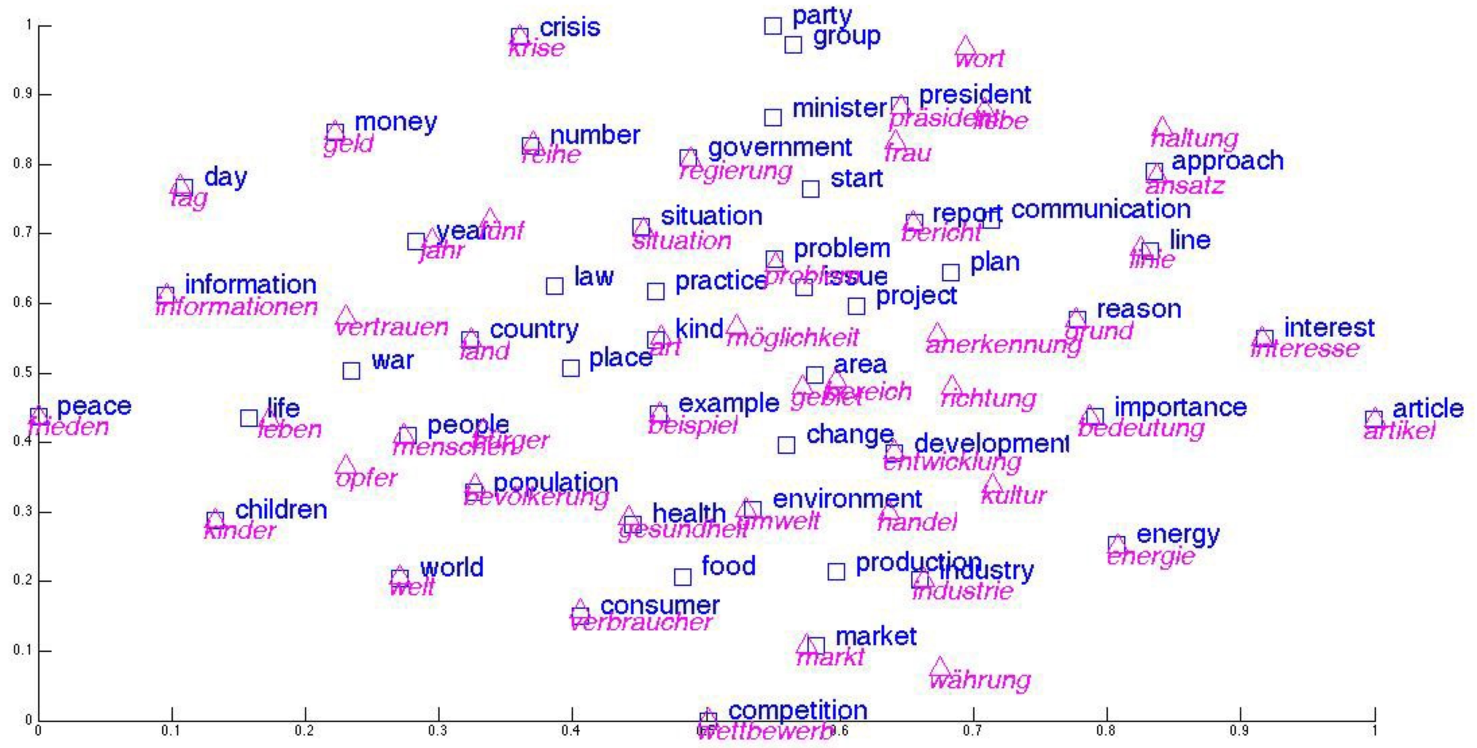
Zero-shot Recognition via Semantic Embeddings and Knowledge Graphs, Wang et.al 2018

Zero-shot Learning Problem

- Given
  - Training set $D_{tr} = \{(\boldsymbol{x}_i, c_i^{tr}), \;\; i = 1, 2, \cdots, n\}$
    - Training example is $\boldsymbol{x}_i$
    - Corresponding class label is c_i^{tr}
    - Class label is drawn from the **seen** training set classes $C_{tr} = \{c_i^{tr} \mid i = 1, 2, \cdots, n_C^{tr}\}$
  - **Unseen** test set classes $C_{te} = \{c_i^{te} \mid i = 1, 2, \cdots, n_C^{te}\}$
  - Semantic/auxiliary representation vectors for each class in both training and test set $\{\boldsymbol{v}_i^{tr} | i = 1, 2, \cdots, n_C^{tr}\} \cup \{\boldsymbol{v}_j^{te} \mid \; j = 1, 2, \cdots, n_C^{te}\}$
  - Training and testing classes are disjoint $C_{tr} \cap C_{te} = \emptyset$
- Find
  - A function which maps test input examples into the corresponding class $f : \boldsymbol{x}_i^{te} \rightarrow c_i^{te}$

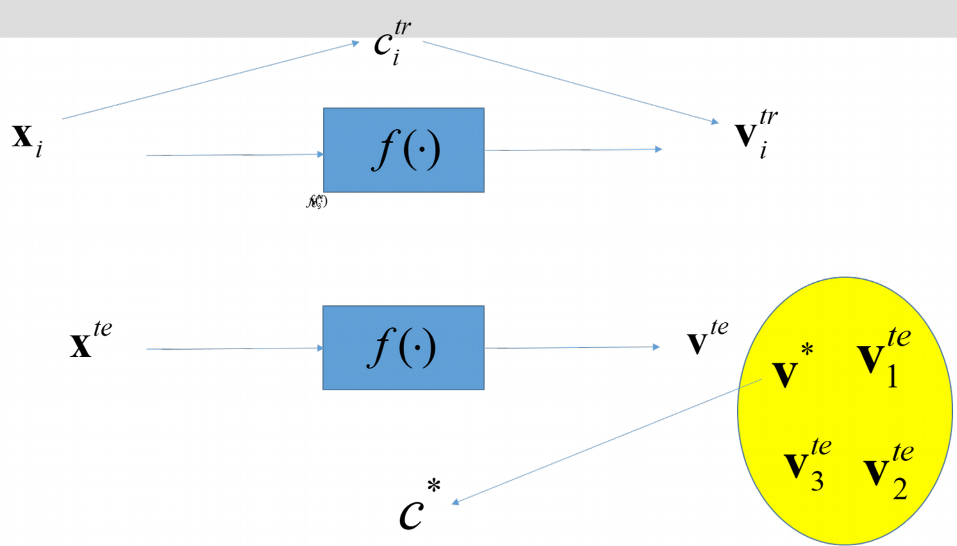
Semantic Representation

- Attribute description:
  - Example
    - Attributes: small, cute, furry, horns
    - Dog=[1,1,1,0], Bull=[0,1,1,1]
- Word co-occurrence count
- Word embedding vectors
  - Trained vector representations for each class name



Zero-shot Learning Approaches-1

- Training:
  - For each training example $(\boldsymbol{x}_i, c_i^{tr})$, get the semantic representation vector (or category vector) $\boldsymbol{v}_i^{tr}$ corresponding to the class c_i^{tr}.
  - Train a network $f(\cdot)$ to map $\boldsymbol{x}_i$ to $\boldsymbol{v}_i^{tr}$.
- Testing:
  - Get the test input $\boldsymbol{x}^{te}$ and map it to the semantic representation vector $\boldsymbol{v}^{te} = f(\boldsymbol{x}^{te})$
  - Find the nearest neighbour to $\boldsymbol{v}^{te}$ and call it $\boldsymbol{v}^{\star}$
  - Return the corresponding class $c^{\star}$ to $\boldsymbol{v}^{\star}$



Zero-shot Learning Approaches-2

- Training:
  - For each training example $(\boldsymbol{x}_i, c_i^{tr})$, get the semantic representation vector (or category vector) $\boldsymbol{v}_i^{tr}$ corresponding to the class c_i^{tr}.
  - Train a network $f(\cdot,\cdot)$ to map $(\boldsymbol{x}_i, \boldsymbol{v}_i^{tr})$ to a similarity value E .
- Testing:
  - Get the test input $\boldsymbol{x}^{te}$ and couple it with semantic representation vector $\boldsymbol{v}_i^{te}$ corresponding to each of the possible classes c_i^{te}. This will create $\{(\boldsymbol{x}^{te}, \boldsymbol{v}_i^{te}) |\ i = 1, 2, \cdots, n_C^{te}\}$
  - For each pair $(\boldsymbol{x}^{te}, \boldsymbol{v}_i^{te})$ find $E_i = f(\boldsymbol{x}^{te}, \boldsymbol{v}_i^{te})$.
  - Return the corresponding class to maximum E_i