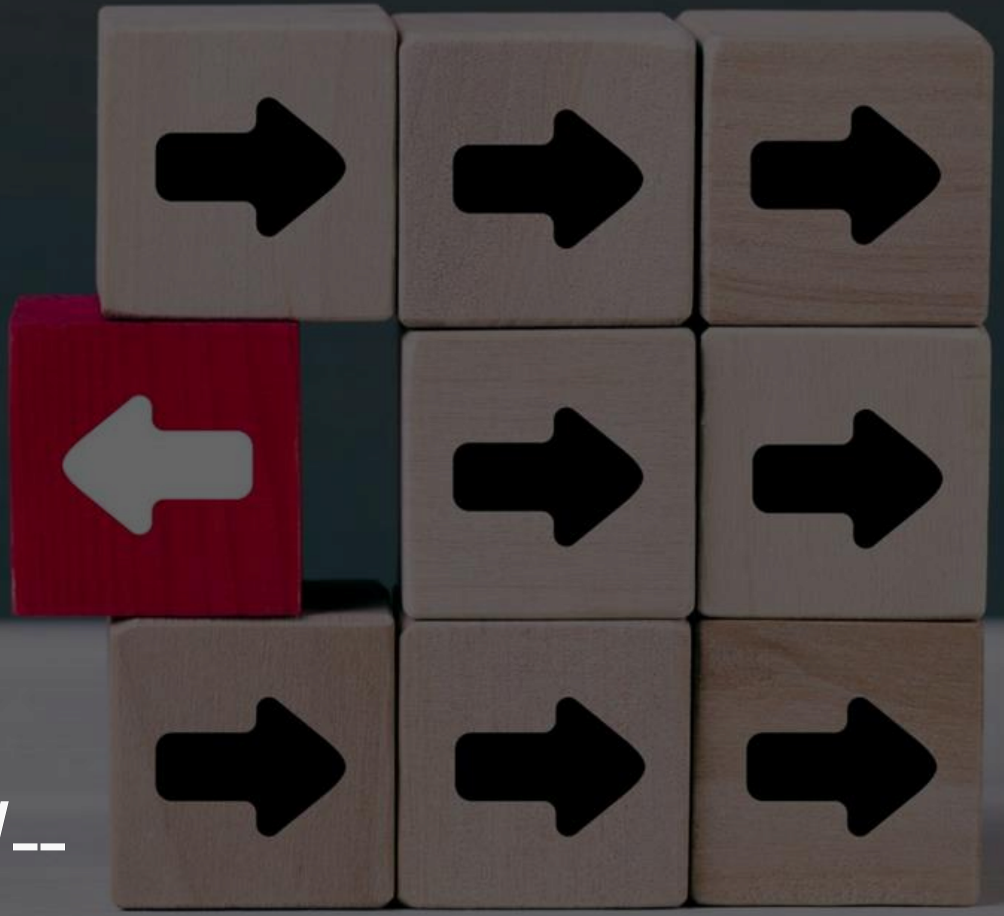
WARSZTATY BADAWCZE: „METODY PRZENOSZENIA WIEDZY W SIECIACH NEURONOWYCH”



-- PRZEGLĄD ALGORYTMÓW--

Paulina Tomaszewska

Metody przenoszenia wiedzy vs. transfer learning

Metody przenoszenia wiedzy*:

- *transfer learning***
- *multi-task learning*
- *continual learning*
- *knowledge distillation*
- *meta-learning*
- *domain adaptation*
- *etc.*

Metody przenoszenia wiedzy

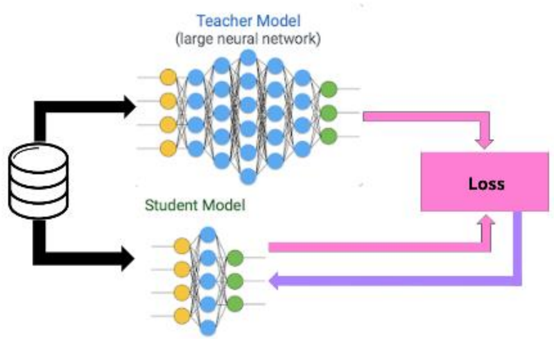
bezpośrednie

pośrednie

*definicja własna, ** w j. polskim określenie „transfer learning" jest używane w wąskim kontekście

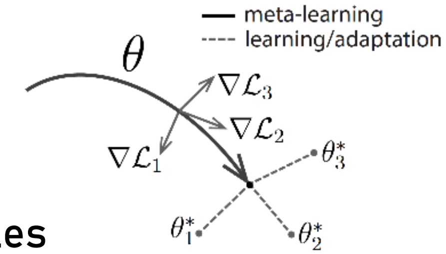
Knowledge distillation

Goal: distil knowledge from big neural network to small one to achieve similar performance



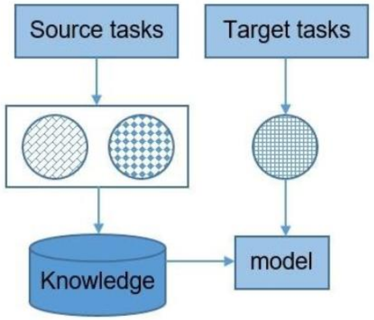
Meta Learning (learning to learn)

Example: MAML
Goal: learn weights that can be easily (by only few gradient updates) adjusted to new scenarios using only few samples



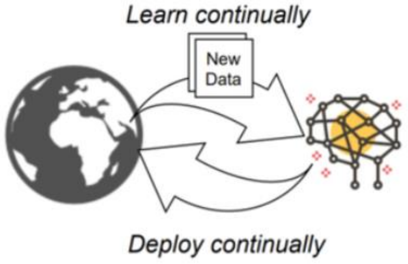
Transfer Learning

Goal: train one model on a big dataset so that the model can be adjusted to the new dataset after less computationally demanding fine-tuning (during pretraining downstream data is not used)



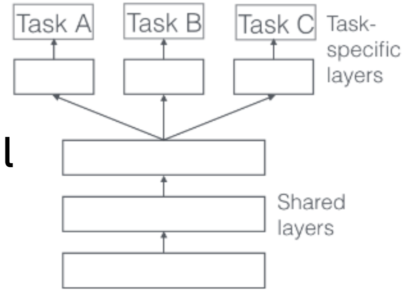
Continual Learning

Goal: train model in an incremental way using non-stationary data sequence
Challenge: catastrophical forgetting due to concept drifts



Multi-task Learning

Goal: share the common knowledge between different tasks that leads to more universal representations



KNOWLEDGE TRANSFER – ARTIFICIAL NEURAL NETWORKS

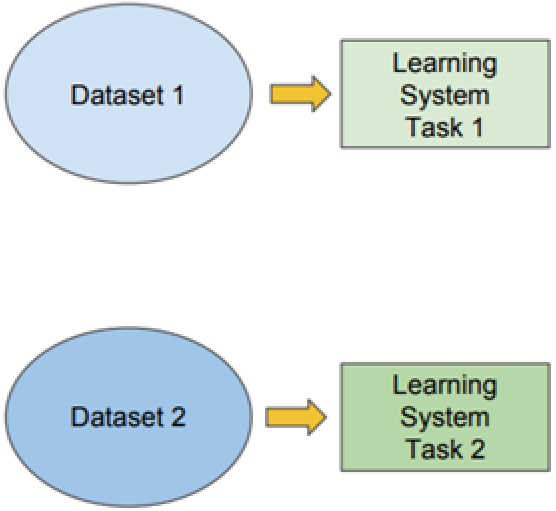
TRANSFER LEARNING

Transfer learning

Traditional ML vs Transfer Learning

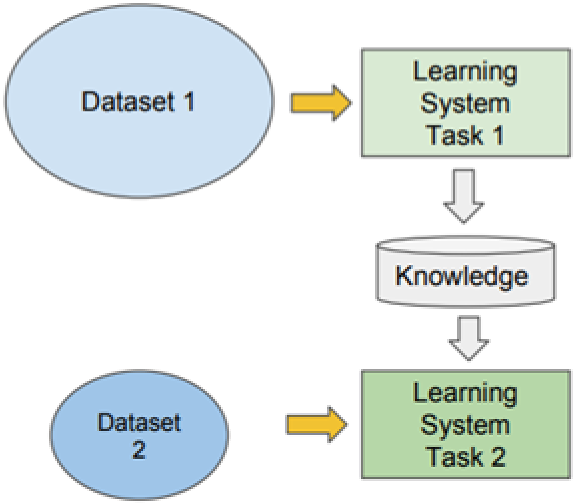
Traditional ML

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



Transfer Learning

- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data



https://medium.com/@agarwallashivam/face-recognition-using-transfer-learning-b8e36e6a15f1

Freeze or fine-tune?

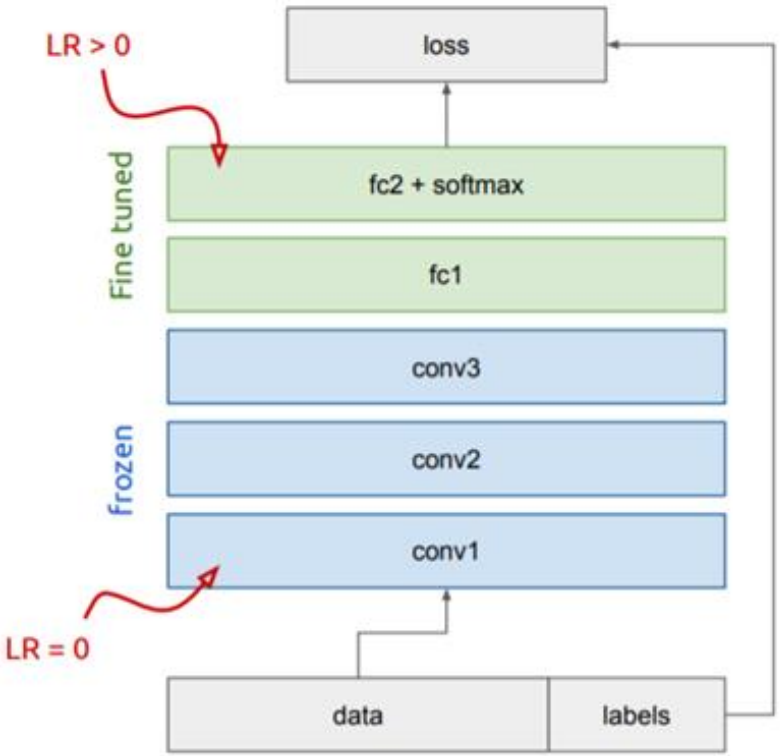
Bottom *n* layers can be frozen or fine tuned.

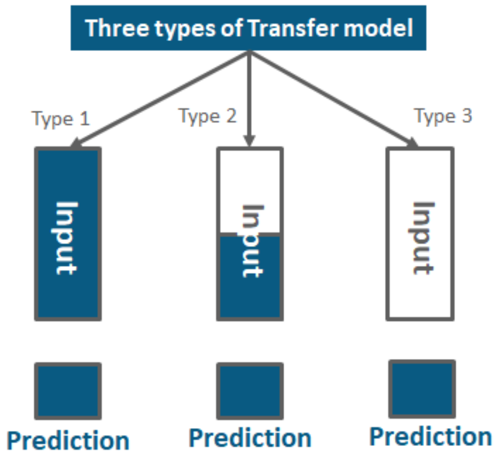
- **Frozen**: not updated during backprop
- **Fine-tuned**: updated during backprop

Which to do depends on target task:

- **Freeze**: target task labels are scarce, and we want to avoid overfitting
- **Fine-tune**: target task labels are more plentiful

In general, we can set learning rates to be different for each layer to find a tradeoff between freezing and fine tuning



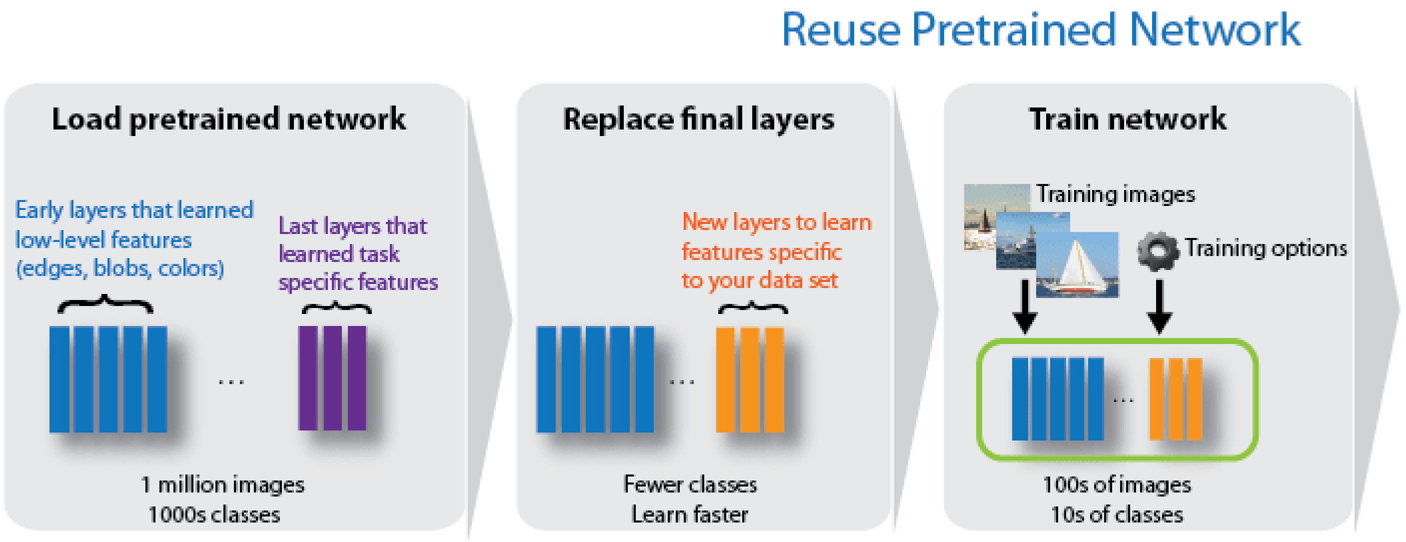


https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a?gi=f7cc6a12ff3a
https://www.quora.com/What-is-Transfer-Learning

Transfer learning in practice

Pretrained networks for Computer Vision:

- *ResNet*
- *VGG*
- *DenseNet*



https://se.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html

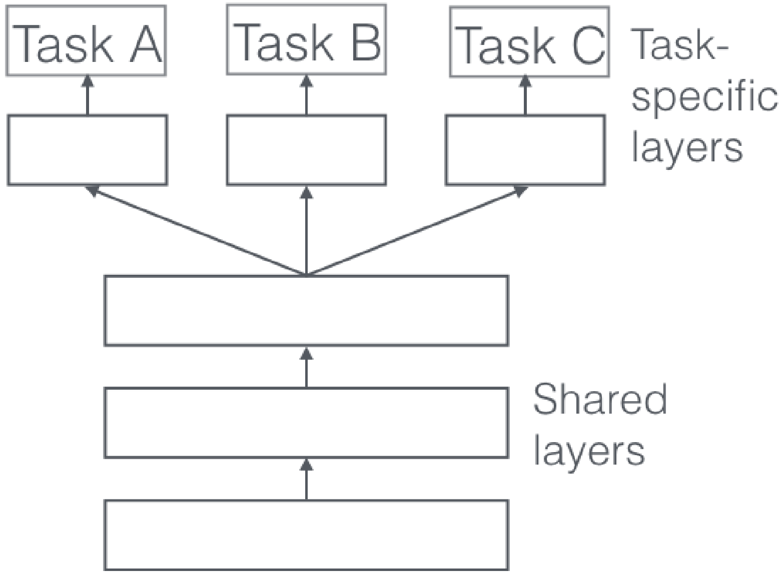
MULTI-TASK LEARNING (MTL)

Motivation

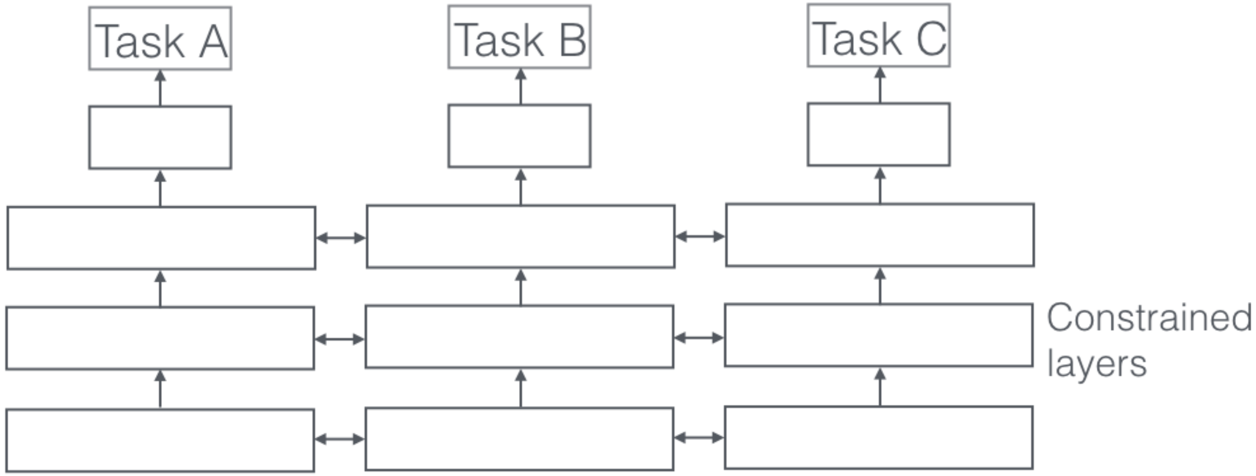
- compact models inspired by human ability of multi-tasking
- better representations (more universal)
- increased performance
- decreased inference time (autonomous cars)

Main problem: negative transfer

MTL architectures – *shared trunk*



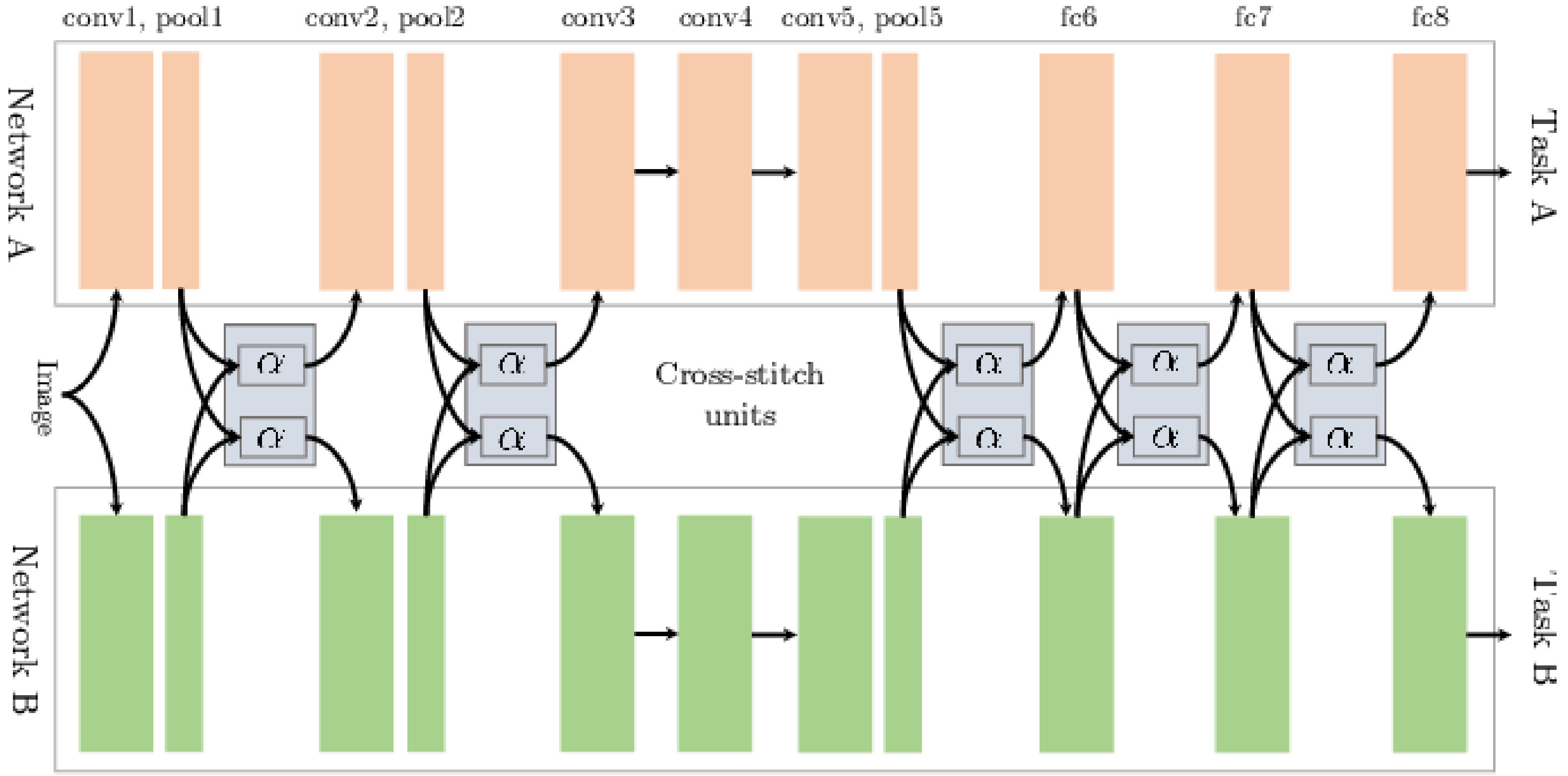
Hard parameter sharing



Soft parameter sharing

https://ruder.io/multi-task/

MTL architectures – *cross talk*



Misra, Ishan et al. “Cross-Stitch Networks for Multi-task Learning.” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016): 3994-4003.

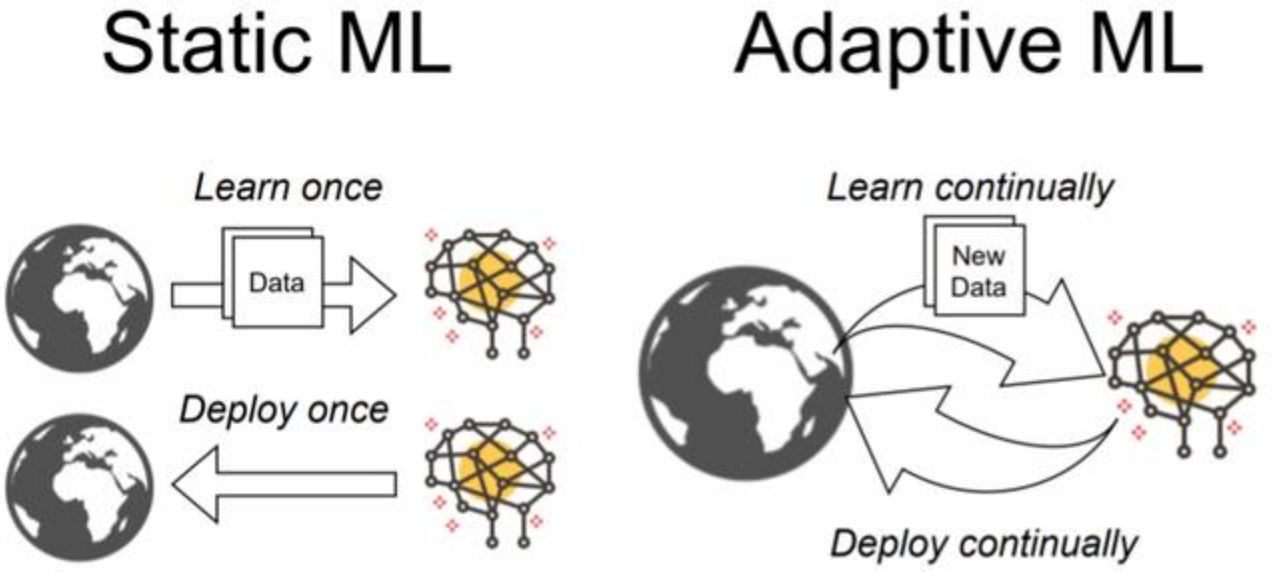
Other research topics in MTL

- Optimization (e.g. GradNorm)
- Dataset (which tasks should be learnt together)

CONTINUAL LEARNING

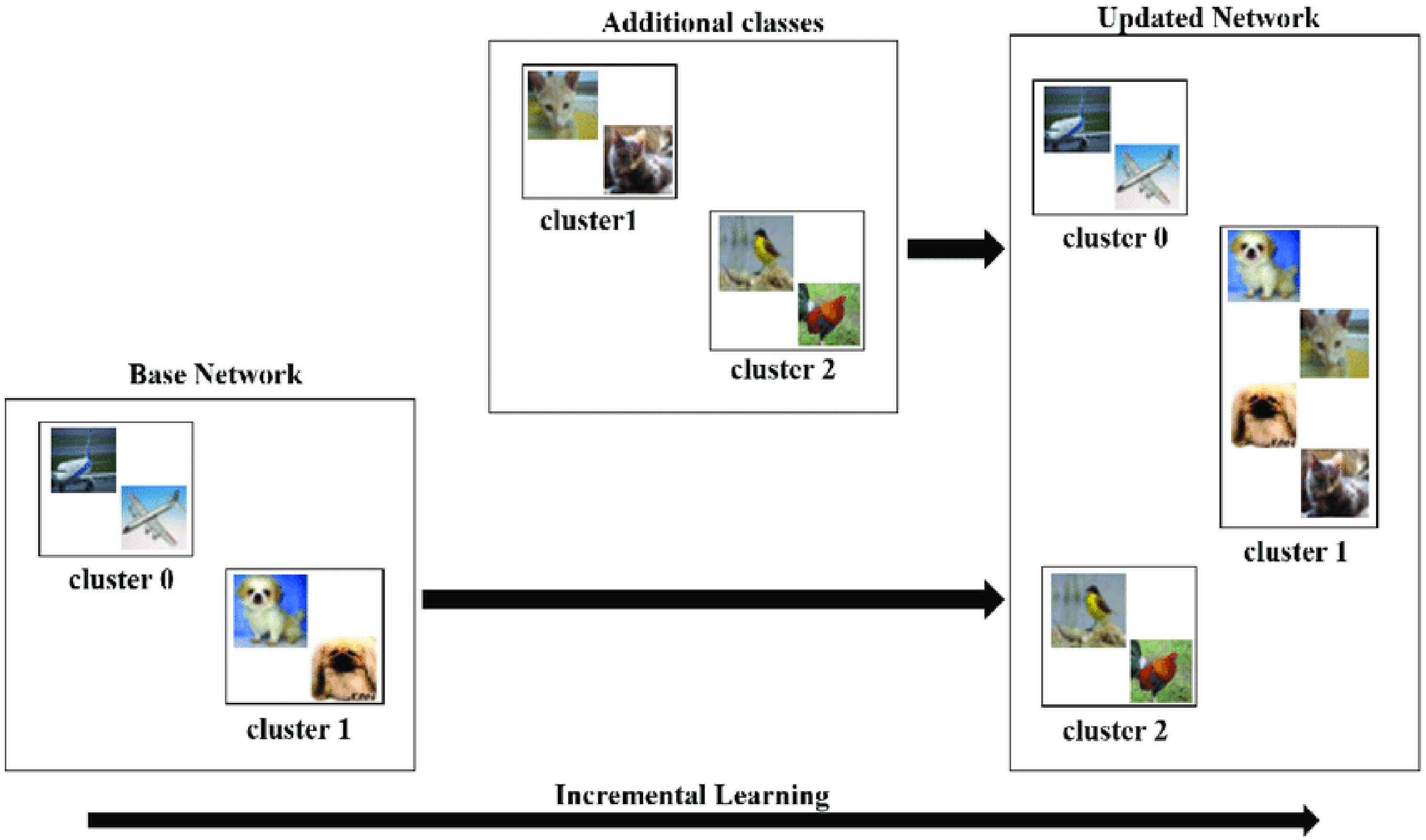
Applications

- Used in case of nonstationary (=changing statistics over time) streaming data
- Similar terms: *Incremental learning, Life-long learning*



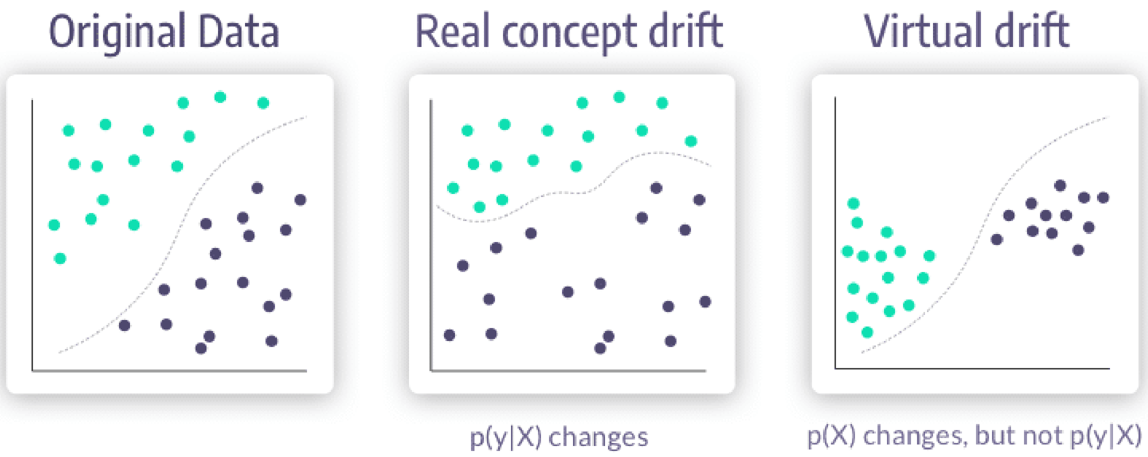
https://ai.kuleuven.be/stories/post/2021-05-10-continual-learning/

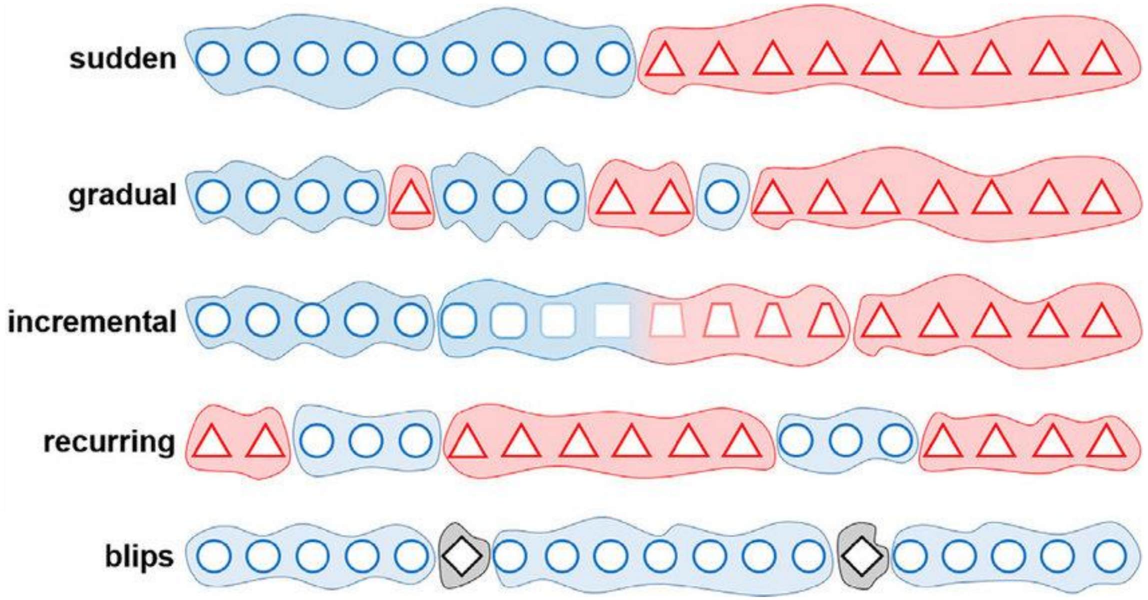
Challenges – additional classes



Madhusudhanan, Sathya & Jaganathan, Suresh & S, Jayashree. (2018). Incremental Learning for Classification of Unstructured Data Using Extreme Learning Machine. Algorithms. 11. 158. 10.3390/a11100158.

Challenges - data drift

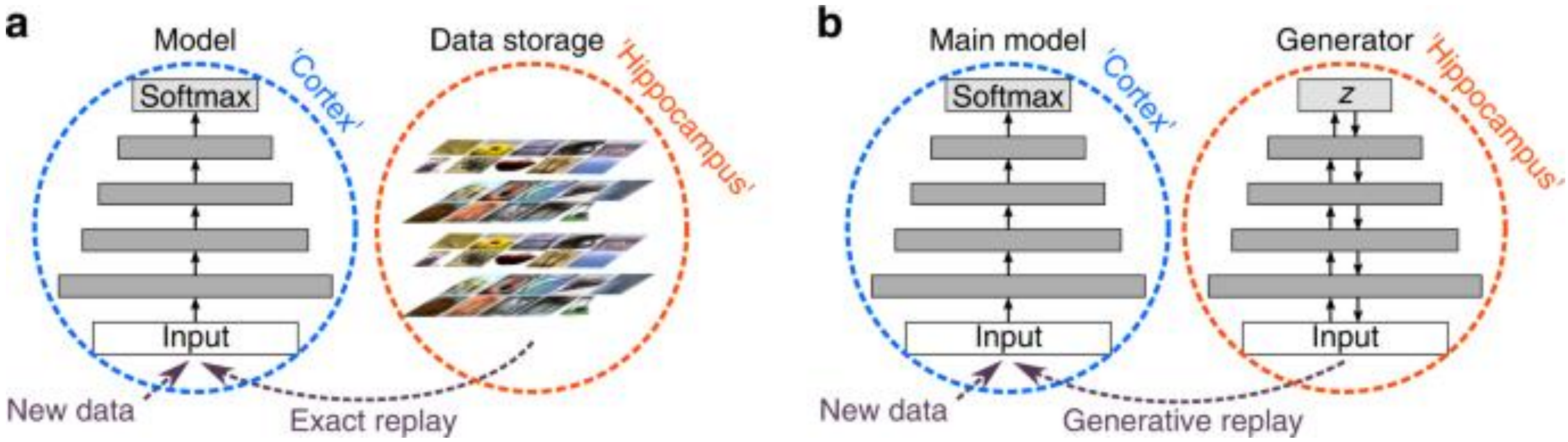




https://www.iguazio.com/blog/concept-drift-deep-dive-how-to-build-a-drift-aware-ml-system/

Krawczyk, Bartosz & Cano, Alberto. (2018). Online Ensemble Learning with Abstaining Classifiers for Drifting and Noisy Data Streams. Applied Soft Computing. 68. 677-692. 10.1016/j.asoc.2017.12.008.

Possible remedy

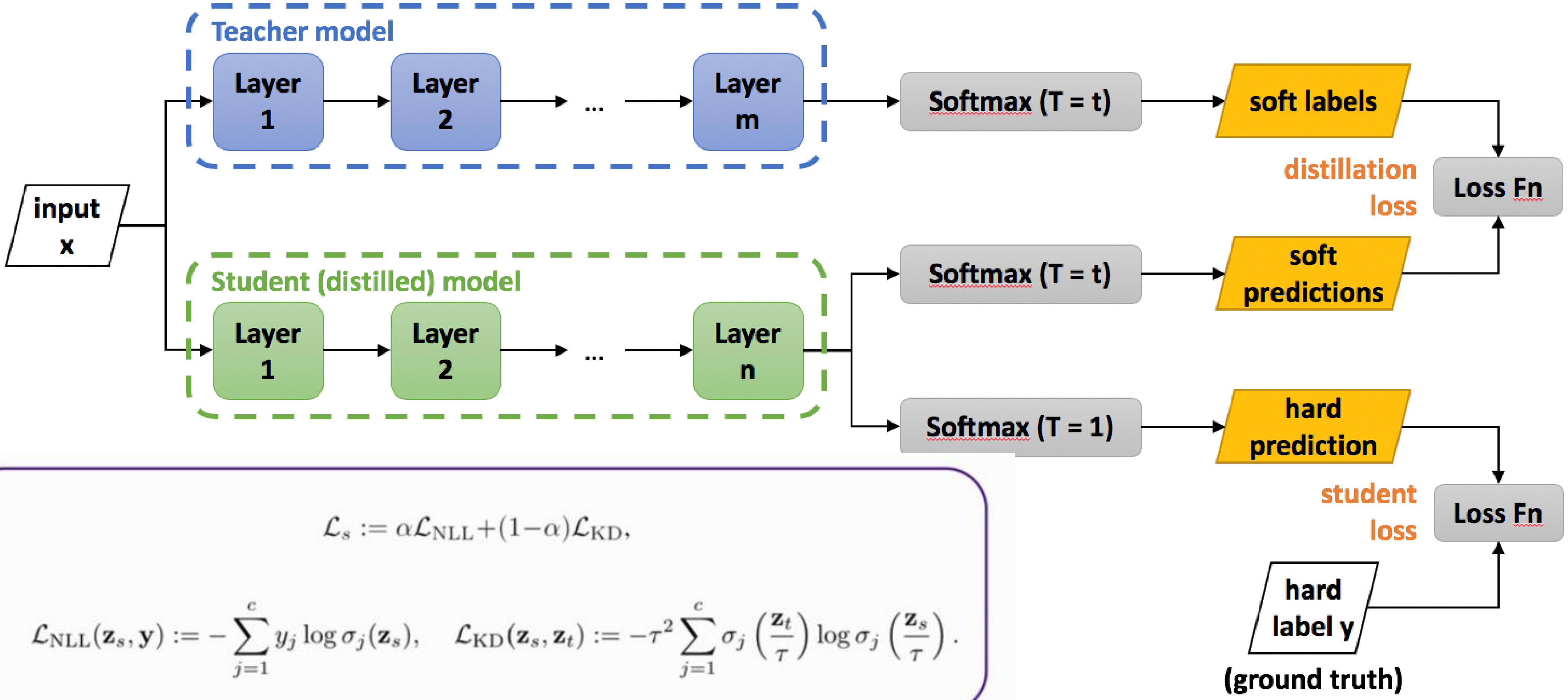


van de Ven, G.M., Siegelmann, H.T. & Tolias, A.S. Brain-inspired replay for continual learning with artificial neural networks. *Nat Commun* **11,** 4069 (2020).
https://doi.org/10.1038/s41467-020-17866-2

KNOWLEDGE DISTILLATION

Architecture

Compress big models so that they can be used at the edge devices (IoT)



$$\mathcal{L}_s := \alpha\mathcal{L}_{\mathrm{NLL}} + (1-\alpha)\mathcal{L}_{\mathrm{KD}},$$

$$\mathcal{L}_{\mathrm{NLL}}(\mathbf{z}_s,\mathbf{y}) := -\sum_{j=1}^{c} y_j \log\sigma_j(\mathbf{z}_s), \quad \mathcal{L}_{\mathrm{KD}}(\mathbf{z}_s,\mathbf{z}_t) := -\tau^2\sum_{j=1}^{c}\sigma_j\left(\frac{\mathbf{z}_t}{\tau}\right)\log\sigma_j\left(\frac{\mathbf{z}_s}{\tau}\right).$$

https://devopedia.org/knowledge-distillation

META-LEARNING

Motivation

- Other term: Learn to learn
- Goals:
  - *learning new concepts and skills fast with a few training examples*
  - *well adapting or generalizing to new tasks that have never been encountered during training time*
- The adaptation process (a mini learning session), happens during test but with a limited exposure to the new task configurations.
- Examples:
  - *A classifier trained on non-cat images can tell whether a given image contains a cat after seeing a handful of cat pictures*
  - *A game bot is able to quickly master a new game (meta reinforcement learning)*
  - *A mini robot completes the desired task on an uphill surface during test even through it was only trained in a flat surface environment (meta reinforcement learning)*

https://lilianweng.github.io/posts/2018-11-30-meta-learning/
https://sites.google.com/mit.edu/aaai2021metalearningtutorial/home

Problem statement

A good meta-learning model should be trained over a variety of learning tasks and optimized for the best performance on a distribution of tasks, including potentially unseen tasks. Each task is associated with a dataset $\mathcal{D}$, containing both feature vectors and true labels. The optimal model parameters are:

$$\theta^* = \arg\min_\theta \mathbb{E}_{\mathcal{D}\sim p(\mathcal{D})}[\mathcal{L}_\theta(\mathcal{D})]$$

It looks very similar to a normal learning task, but *one dataset* is considered as *one data sample*.

https://lilianweng.github.io/posts/2018-11-30-meta-learning/

Few-shot classification

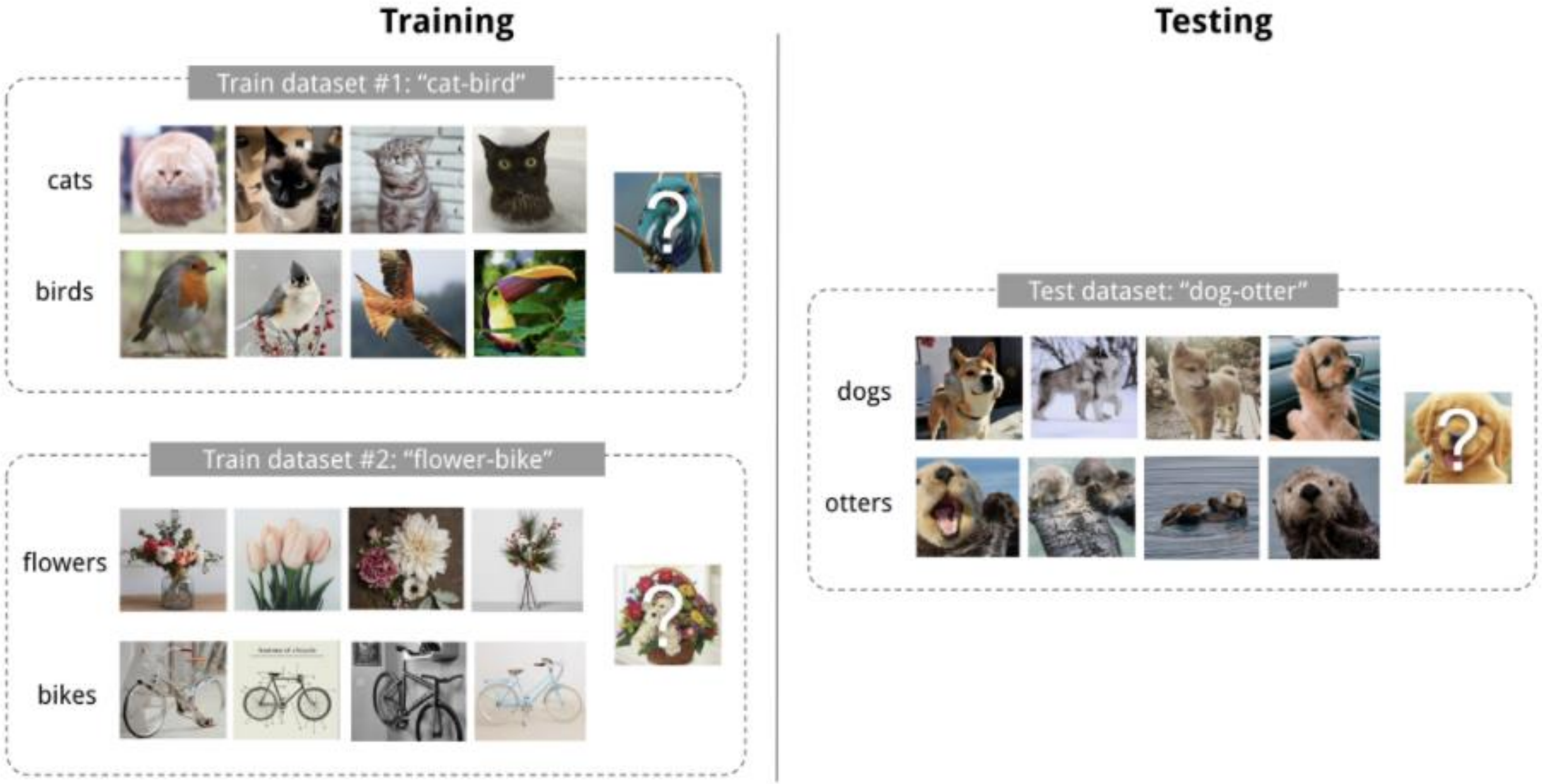


Fig. 1. An example of 4-shot 2-class image classification. (Image thumbnails are from Pinterest)

https://lilianweng.github.io/posts/2018-11-30-meta-learning/

Learner and Meta-Learner

Another popular view of meta-learning decomposes the model update into two stages:

- A classifier f_θ is the "learner" model, trained for operating a given task;
- In the meantime, a optimizer g_ϕ learns how to update the learner model's parameters via the support set S, $\theta' = g_\phi(\theta, S)$.

Then in final optimization step, we need to update both θ and ϕ to maximize:

$$\mathbb{E}_{L\subset\mathcal{L}}[\mathbb{E}_{S^L\subset\mathcal{D},B^L\subset\mathcal{D}}[\sum_{(\mathbf{x},y)\in B^L} P_{g_\phi(\theta,S^L)}(y|\mathbf{x})]]$$

https://lilianweng.github.io/posts/2018-11-30-meta-learning/

Common approaches in meta-learning

- Model-based
- Metric-based
- Optimization-based

https://lilianweng.github.io/posts/2018-11-30-meta-learning/

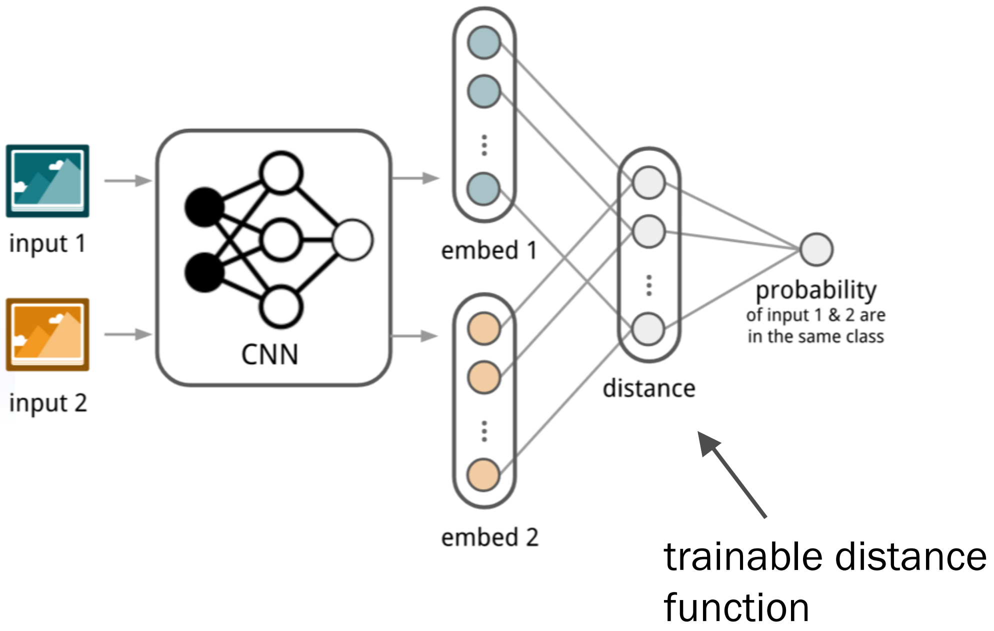
METRIC-BASED META-LEARNING

Analogy to k-nearest neighbours

Siemese networks

Applications:

- recognizing a single person at a train station or airport.
- verifying whether the photo in a pass is the same as the person claiming he or she is the same person.

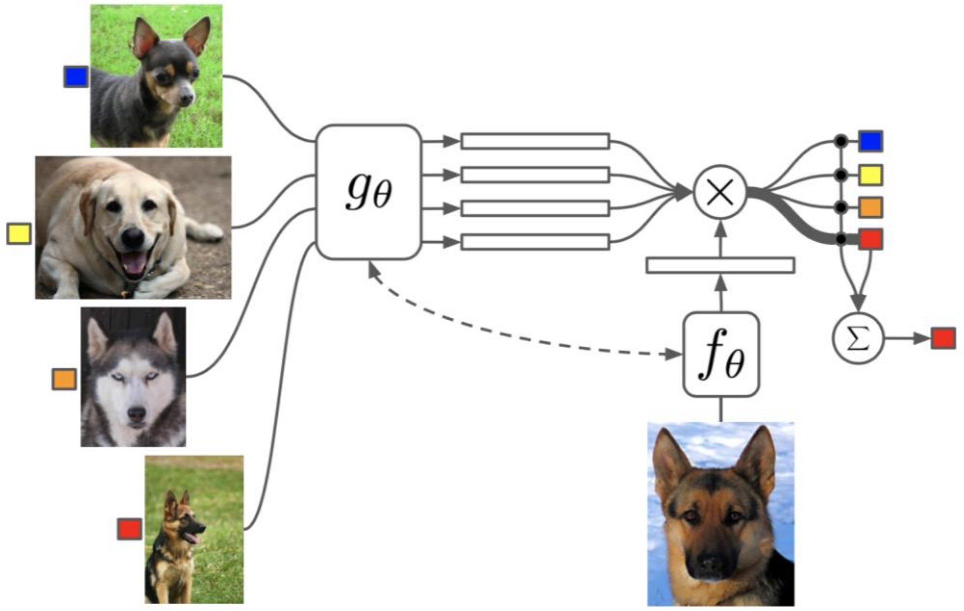


1. First, convolutional siamese network learns to encode two images into feature vectors via a embedding function f_θ which contains a couple of convolutional layers.
2. The L1-distance between two embeddings is $|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)|$.
3. The distance is converted to a probability p by a linear feedforward layer and sigmoid. It is the probability of whether two images are drawn from the same class.
4. Intuitively the loss is cross entropy because the label is binary.

$$p(\mathbf{x}_i, \mathbf{x}_j) = \sigma(\mathbf{W}|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)|)$$
$$\mathcal{L}(B) = \sum_{(\mathbf{x}_i, \mathbf{x}_j, y_i, y_j) \in B} \mathbf{1}_{y_i=y_j} \log p(\mathbf{x}_i, \mathbf{x}_j) + (1 - \mathbf{1}_{y_i=y_j}) \log(1 - p(\mathbf{x}_i, \mathbf{x}_j))$$

https://lilianweng.github.io/posts/2018-11-30-meta-learning/

Matching networks



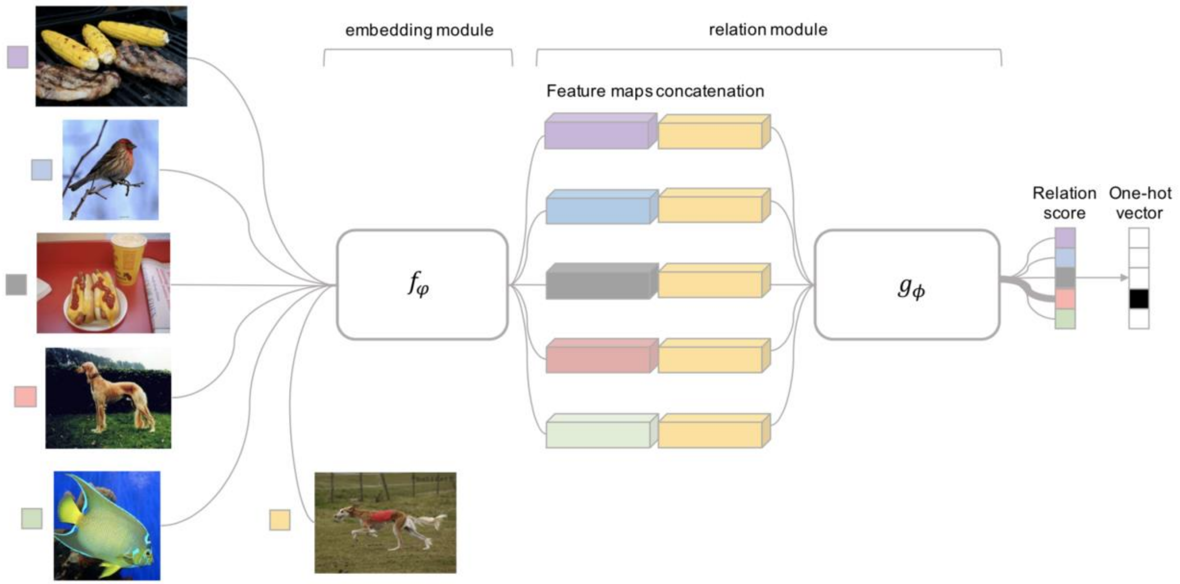
$$c_S(\mathbf{x}) = P(y|\mathbf{x}, S) = \sum_{i=1}^{k} a(\mathbf{x}, \mathbf{x}_i) y_i, \text{ where } S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{k}$$

The attention kernel depends on two embedding functions, f and g, for encoding the test sample and the support set samples respectively. The attention weight between two data points is the cosine similarity, $\text{cosine}(.)$, between their embedding vectors, normalized by softmax:

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

https://lilianweng.github.io/posts/2018-11-30-meta-learning/

Relation Network

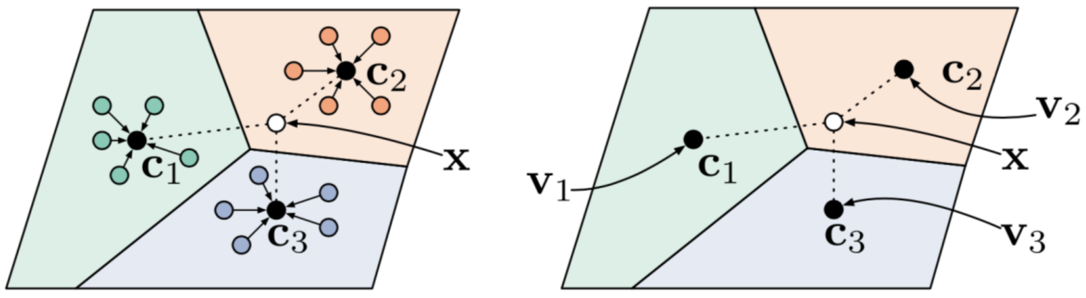


Relation Network (RN) (Sung et al., 2018) is similar to siamese network but with a few differences:

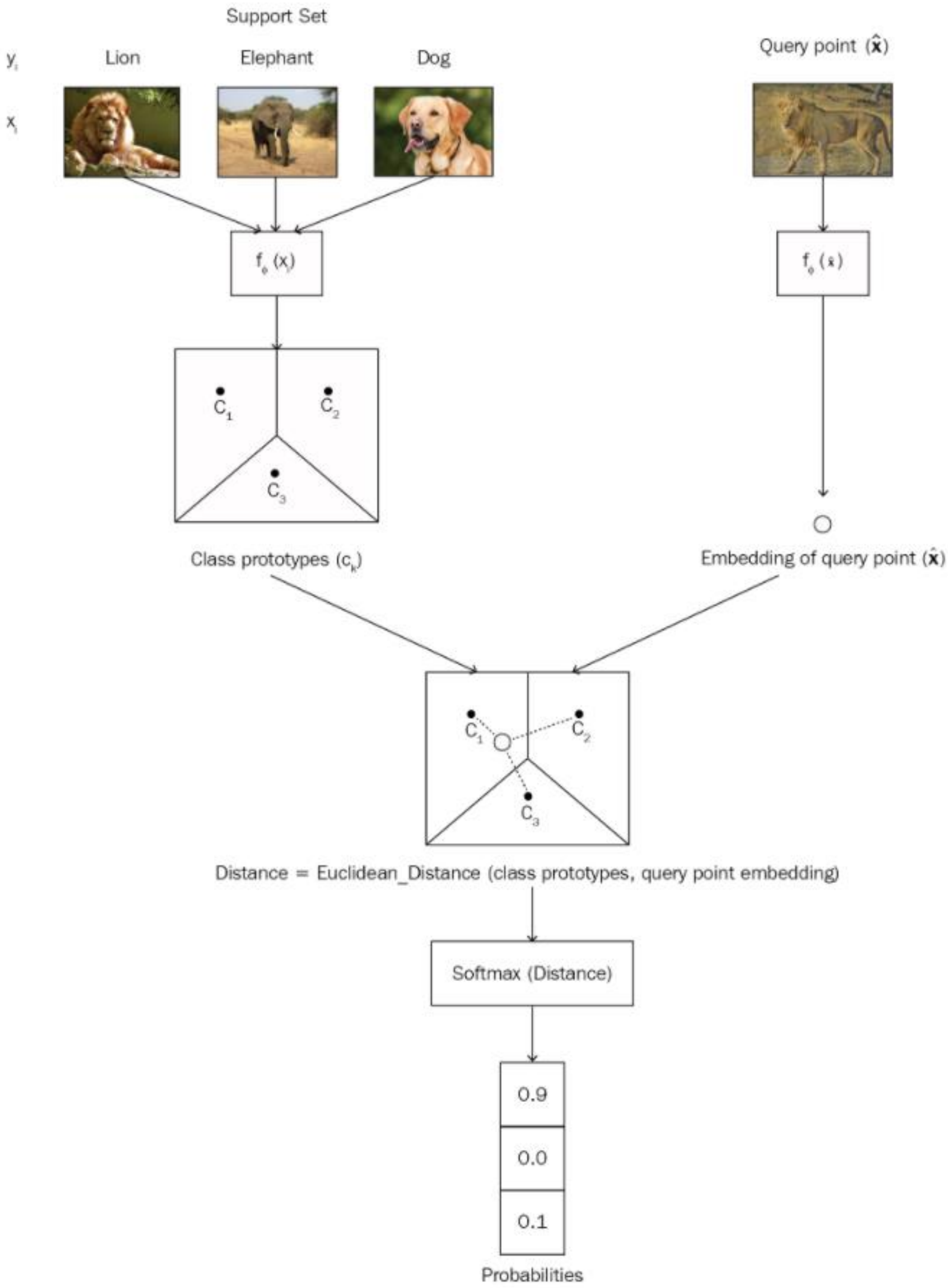
1. The relationship is not captured by a simple L1 distance in the feature space, but predicted by a CNN classifier g_ϕ. The relation score between a pair of inputs, $\mathbf{x}_i$ and $\mathbf{x}_j$, is $r_{ij} = g_\phi([\mathbf{x}_i, \mathbf{x}_j])$ where $[.,.]$ is concatenation.
2. The objective function is MSE loss instead of cross-entropy, because conceptually RN focuses more on predicting relation scores which is more like regression, rather than binary classification, $\mathcal{L}(B) = \sum_{(\mathbf{x}_i, \mathbf{x}_j, y_i, y_j) \in B} (r_{ij} - \mathbf{1}_{y_i=y_j})^2$.

https://lilianweng.github.io/posts/2018-11-30-meta-learning/

Prototypical networks



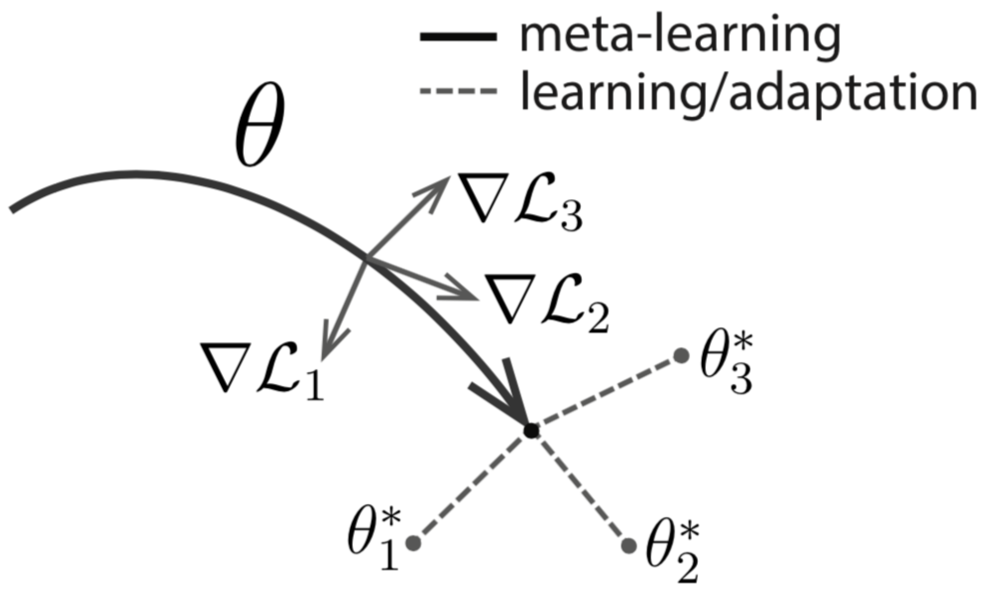
(a) Few-shot (b) Zero-shot



https://lilianweng.github.io/posts/2018-11-30-meta-learning/

OPTIMIZATION-BASED META-LEARNING

Model-Agnostic Meta-Learning (MAML)



Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters

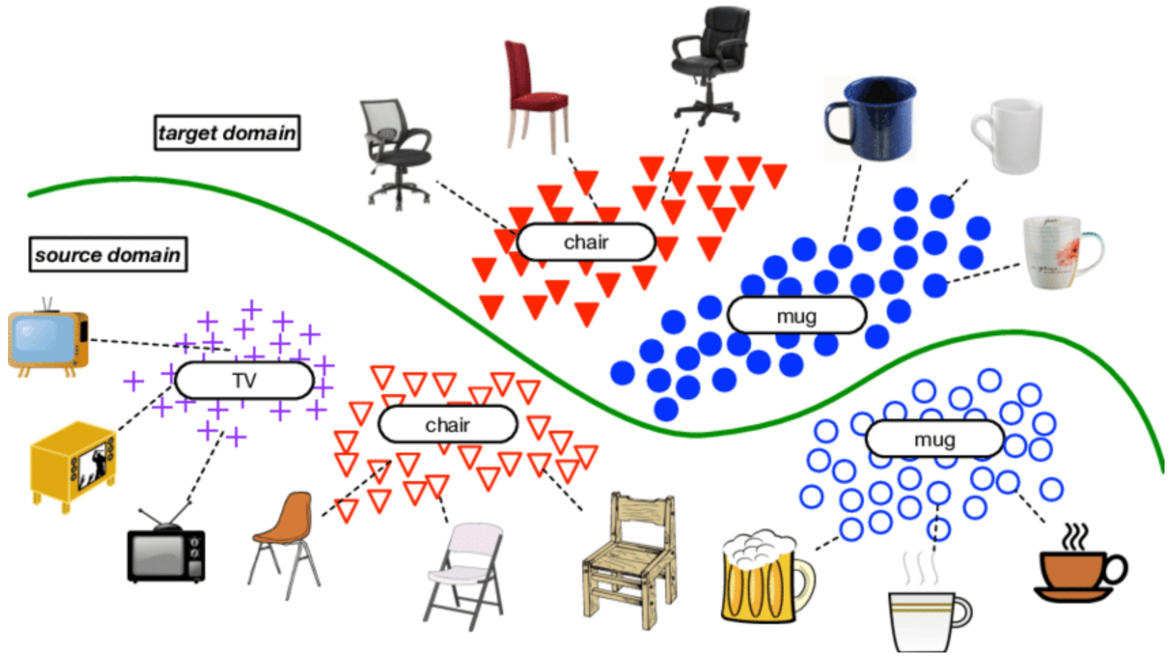
```
randomly initialize θ
while not done do
    Sample batch of tasks T_i ~ p(T)
    for all T_i do
        Evaluate ∇_θ L_{T_i}(f_θ) with respect to K examples
        Compute adapted parameters with gradient de-
        scent: θ'_i = θ − α∇_θ L_{T_i}(f_θ)
    end for   Note: the meta-update is using different set of data.
    Update θ ← θ − β∇_θ Σ_{T_i~p(T)} L_{T_i}(f_{θ'_i})
end while
```

- randomly initialize θ
- **while** not done **do**
  - Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  - **for all** $\mathcal{T}_i$ **do**
    - Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
    - Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  - **end for** Note: the meta-update is using different set of data.
  - Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
- **end while**

https://lilianweng.github.io/posts/2018-11-30-meta-learning/

DOMAIN ADAPTATION

Motivation



- learning **fair representations** (insensitive to some given attributes), while retaining enough information to solve the task

More information at: https://ameroyer.github.io/reading-notes/representation%20learning/2019/04/26/gradient_reversal_against_discrimination.html

The proposed model builds on the Domain Adversarial Network (DANN) [1], originally introduced for unsupervised domain adaptation. Given some labeled data $(x, y) \sim \mathcal{X} \times \mathcal{Y}$, and some unlabeled data $\tilde{x} \sim \tilde{\mathcal{X}}$, the goal is to learn a network that solves both classification tasks $\mathcal{X} \to \mathcal{Y}$ and $\tilde{\mathcal{X}} \to \mathcal{Y}$ while learning a shared representation between $\mathcal{X}$ and $\tilde{\mathcal{X}}$.

https://www.researchgate.net/figure/The-partial-domain-adaptation-scenario-introduced-in-this-paper-where-source-label-space_fig1_327010582
https://ameroyer.github.io/reading-notes/representation%20learning/2019/04/26/gradient_reversal_against_discrimination.html

Omówienie kodu – rozwiązań z kanonu

- Siemese networks: https://github.com/sudharsan13296/Hands-On-Meta-Learning-With-Python/blob/master/02.%20Face%20and%20Audio%20Recognition%20using%20Siamese%20Networks/2.4%20Face%20Recognition%20Using%20Siamese%20Network.ipynb
- Prototypical networks: https://github.com/PacktPublishing/Hands-On-Meta-Learning-with-Python/tree/master/Chapter03
- Relation networks, Matching networks: https://github.com/PacktPublishing/Hands-On-Meta-Learning-with-Python/tree/master/Chapter04
- MAML: https://github.com/sudharsan13296/Hands-On-Meta-Learning-With-Python/tree/master/06.%20MAML%20and%20it's%20Variants