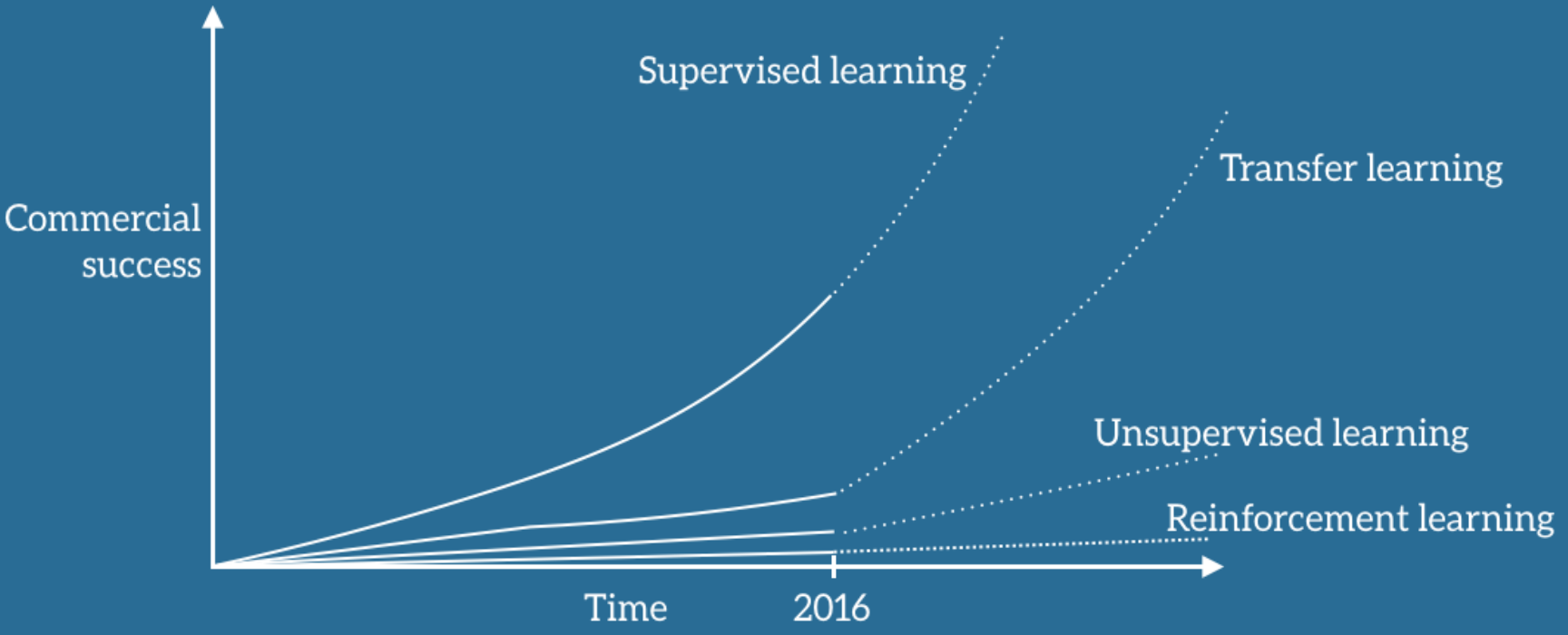
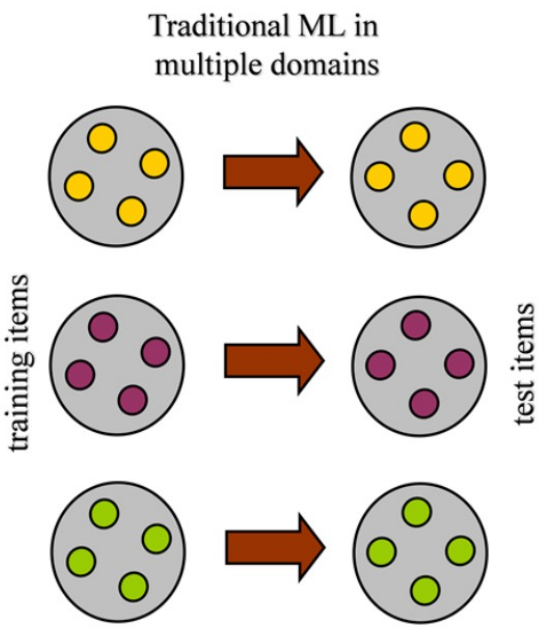
Transfer- and Meta-Learning

Drivers of ML success in industry

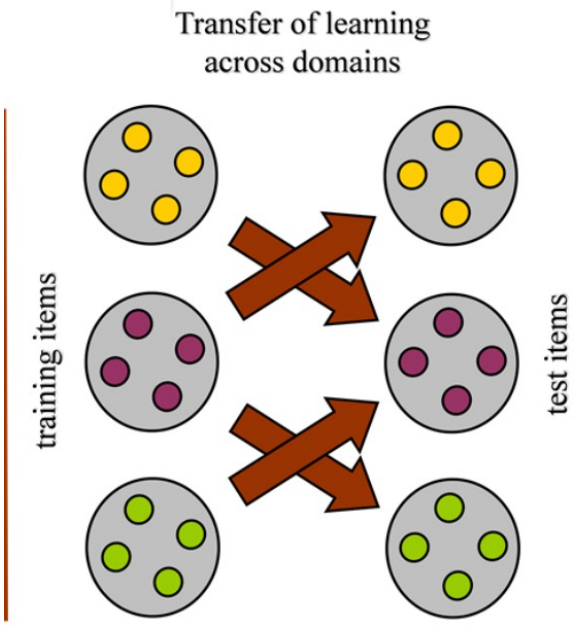


- Andrew Ng, NIPS 2016 tutorial

Transfer Learning

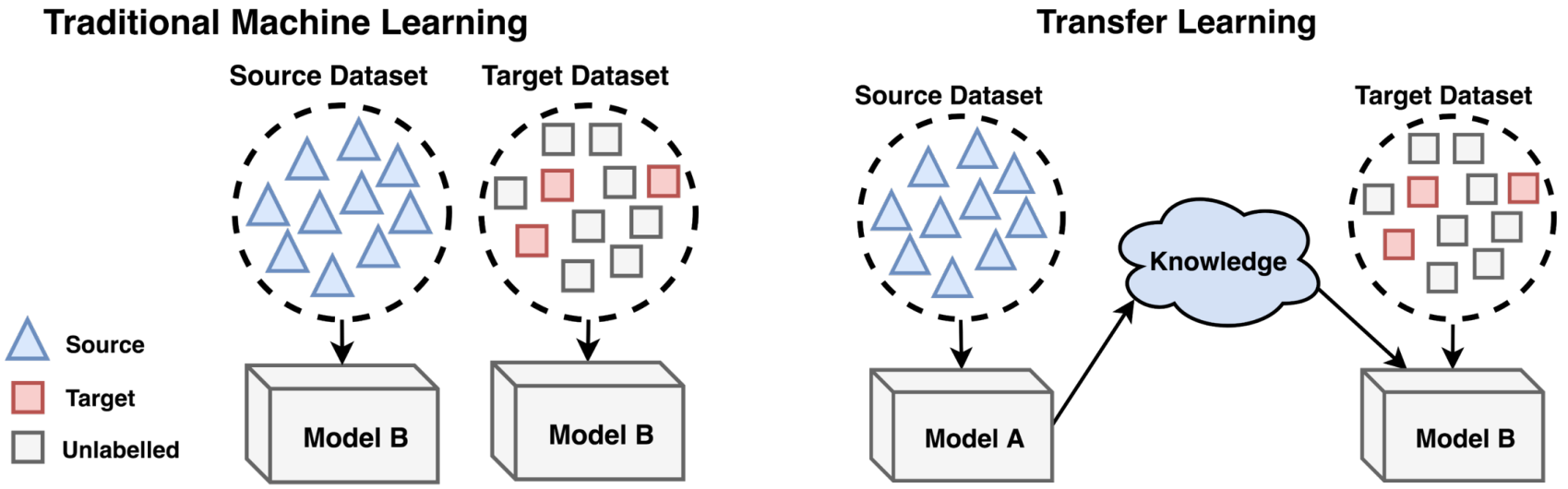


Humans can learn in many domains.



Humans can also transfer from one domain to other domains.

Transfer Learning



Why transfer learning?

- Labeled data might be limited, *data-poor*
- Calibration – training – can be very *expensive*
- The learning process can be (too) *long*

Objectives of transfer learning:

- Extract knowledge learned from related domains to help training in a target domain with a few labeled data
- Extract knowledge learned from related domains to speed up learning in a target domain

For simplicity, we only consider at most two domains and two tasks.

Source domain:

$$\mathcal{P}(X_S), \text{where } X_S = \{x_{S_1}, x_{S_2}, ..., x_{S_{n_S}}\} \in \mathcal{X}_S$$

Task in the source domain:

$$\mathcal{P}(Y_S|X_S), \text{where } Y_S = \{y_{S_1}, y_{S_2}, ..., y_{S_{n_S}}\} \text{ and } y_{S_i} \in \mathcal{Y}_S$$

Target domain:

$$\mathcal{P}(X_T), \text{where } X_T = \{x_{T_1}, x_{T_2}, ..., x_{T_{n_T}}\} \in \mathcal{X}_T$$

Task in the target domain

$$\mathcal{P}(Y_T|X_T), \text{where } Y_T = \{y_{T_1}, y_{T_2}, ..., y_{T_{n_T}}\} \text{ and } y_{T_i} \in \mathcal{Y}_T$$

Approaches to Transfer Learning

Transfer learning approaches	Description
Instance-transfer	To re-weight some labeled data in a source domain for use in the target domain
Feature-representation-transfer	Find a "good" feature representation that reduces difference between a source and a target domain or minimizes error of models
Model-transfer	Discover shared parameters or priors of models between a source domain and a target domain
Relational-knowledge-transfer	Build mapping of relational knowledge between a source domain and a target domain.

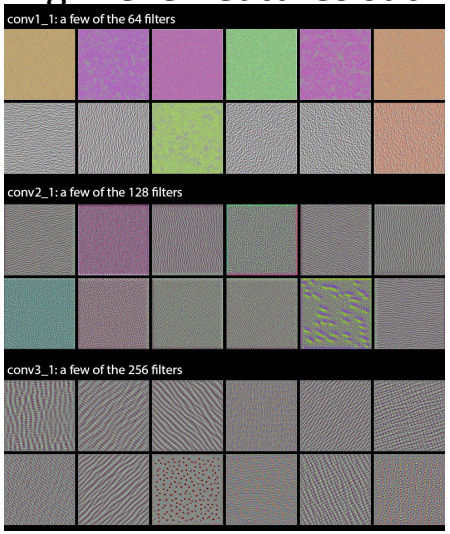
Transfer learning types (inductive model transfer)

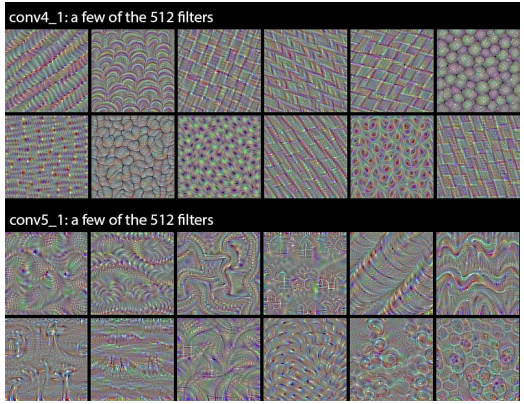
Type	Description	Examples
Inductive	Adapt existing **supervised** training model on new **labeled** dataset	Classification, Regression
Transductive	Adapt existing **supervised** training model on new **unlabeled** dataset	Classification, Regression
Unsupervised	Adapt existing **unsupervised** training model on new **unlabeled** dataset	Clustering, Dimensionality Reduction

From generic to specific

Neural Network Layers: General to Specific

- Bottom/first/earlier layers: general learners
  - Low-level notions of edges, visual shapes
- Top/last/later layers: specific learners
  - High-level features such as eyes, nose

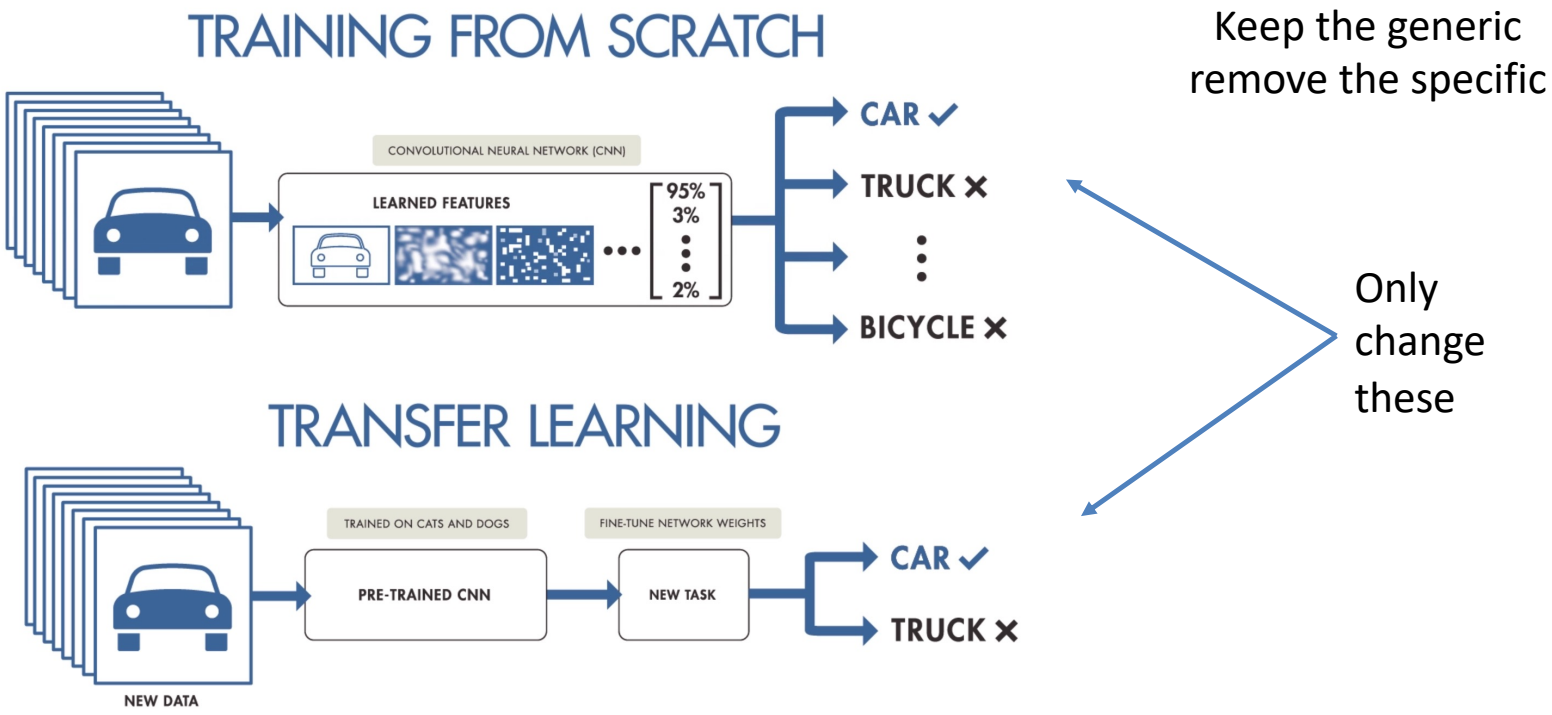




Keep the generic
adjust the specific

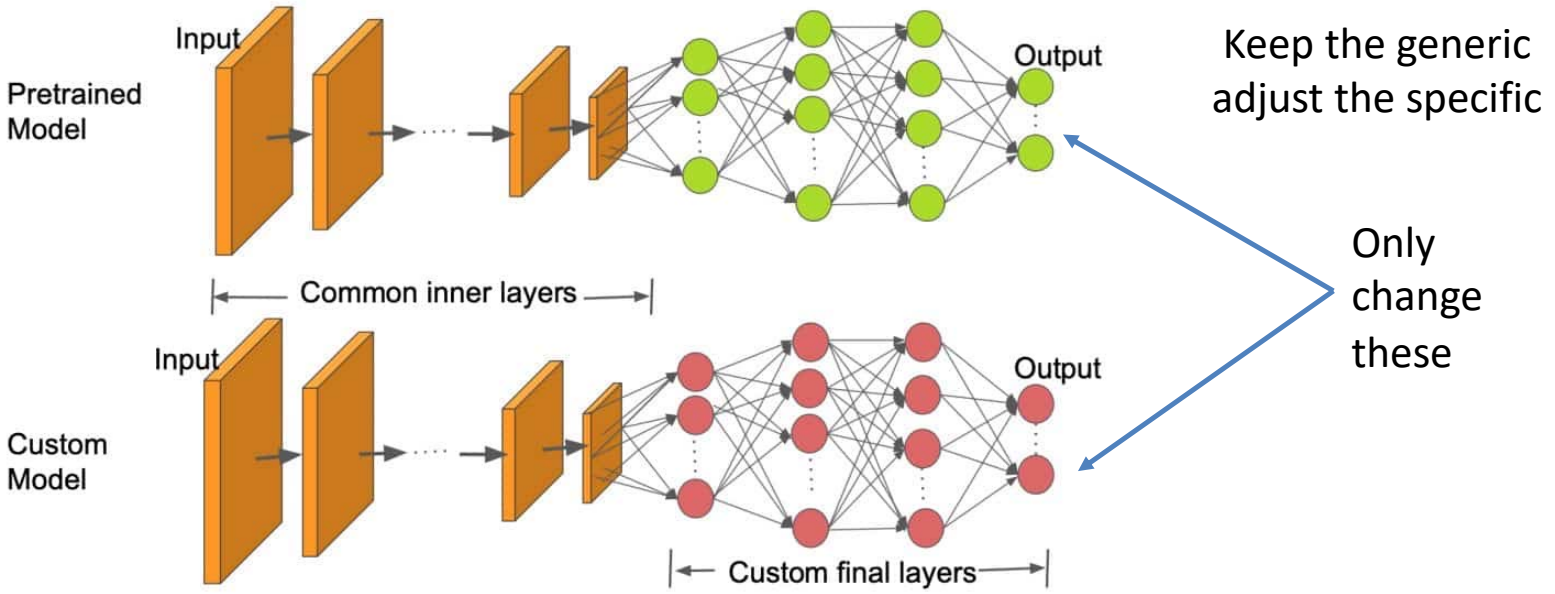
From generic to specific

Neural Network Layers: General to Specific



From generic to specific

Neural Network Layers: General to Specific



Current example

ChatGPT fine tuning
https://platform.openai.com/docs/guides/fine-tuning/when-to-use-fine-tuning

Some common use cases where fine-tuning can improve results:

- Setting the style, tone, format, or other qualitative aspects
- Improving reliability at producing a desired output
- Correcting failures to follow complex prompts
- Handling many new cases in specific ways (e.g., new type of data)
- Performing a new skill or task that's hard to articulate in a prompt

Current example

Fine Tuning for El Niño: climate prediction

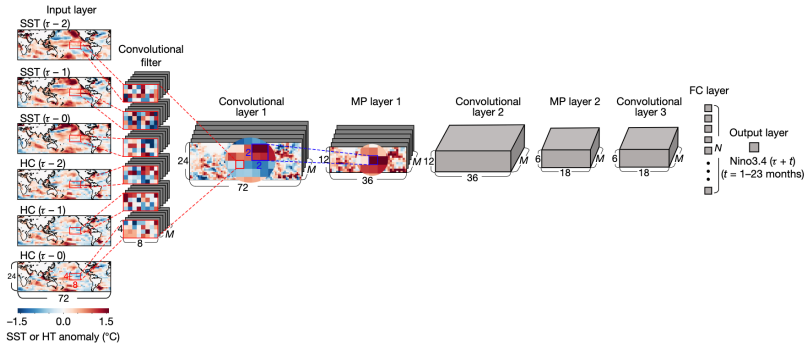
Model trained on large ensemble of climate models, fine-tuned on observations

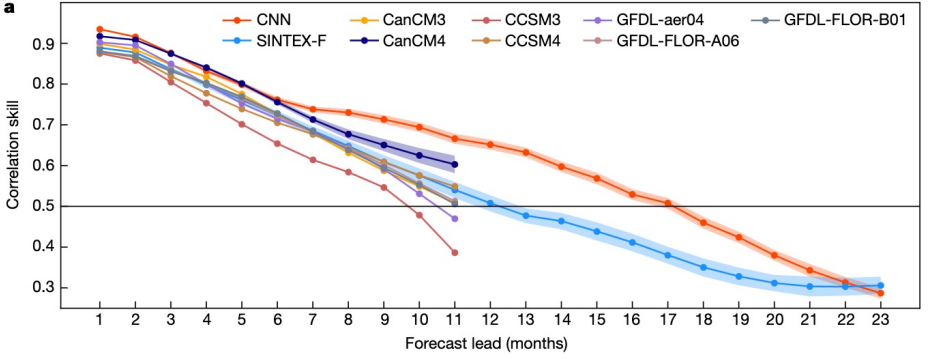
LETTER

https://doi.org/10.1038/s41586-019-1559-7

Deep learning for multi-year ENSO forecasts

Yoo-Geun Ham[1]*, Jeong-Hwan Kim[1] & Jing-Jia Luo[2,3]



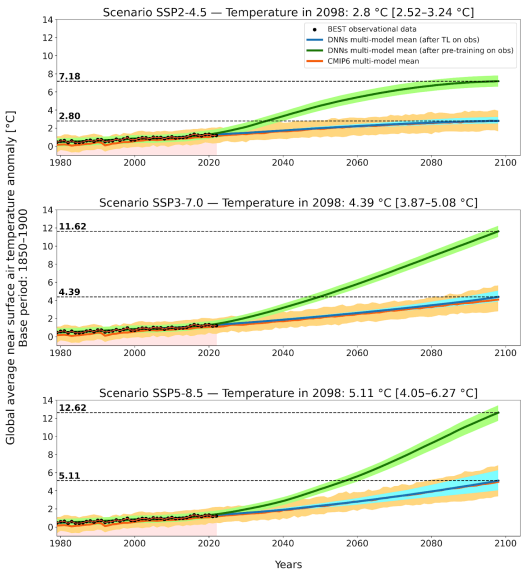


Current example

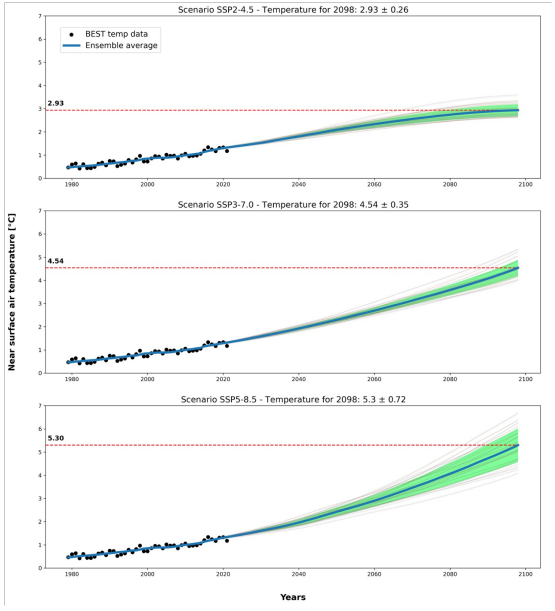
Fine Tuning for climate projections

Model trained on large ensemble of climate models, fine-tuned on observations

Without pretrained network

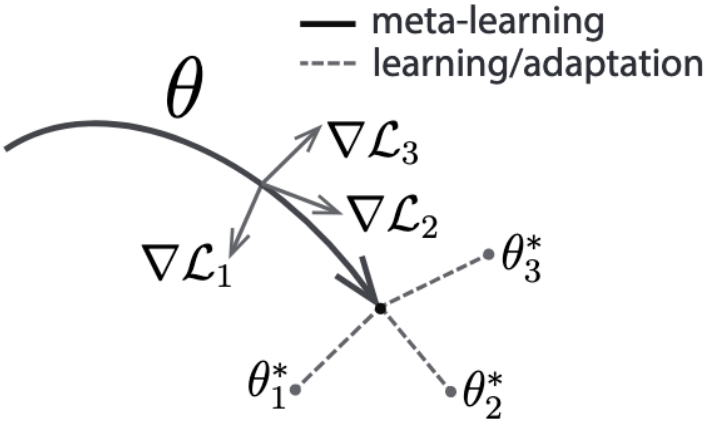


After transfer learning+ pretrained network



MetaLearning: learning to learn

Learning across tasks for rapid adaptation,
a problem setting that is often formalized as few-shot learning



Finn et al. 2017 MAML

MetaLearning: learning to learn

Model-Agnostic Meta-Learning MAML over different **tasks**

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks

Require: α, β: step size hyperparameters

1: randomly initialize θ

2: **while** not done **do**

3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$

4: **for all** $\mathcal{T}_i$ **do**

5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples

6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$

7: **end for**

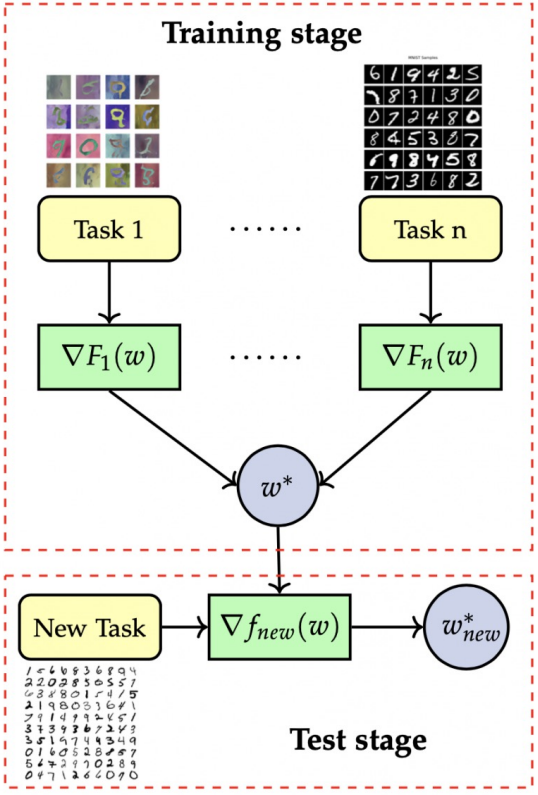
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$

9: **end while**

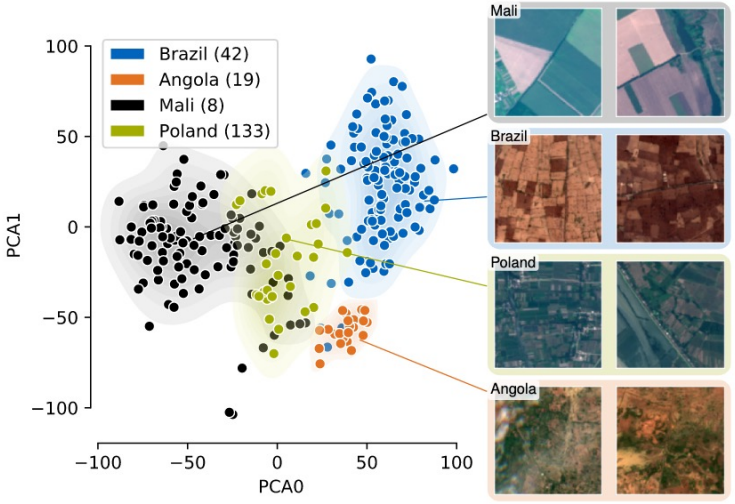
β is the meta step size.

Finn et al. 2017 MAML

Model-Agnostic Meta-Learning MAML



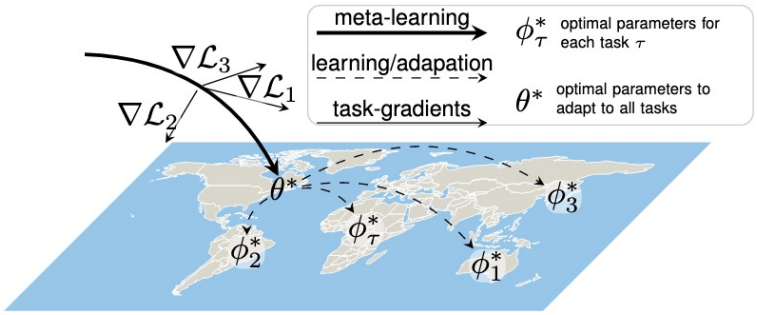
Example Meta-Learning for Few-Shot Land Cover Classification

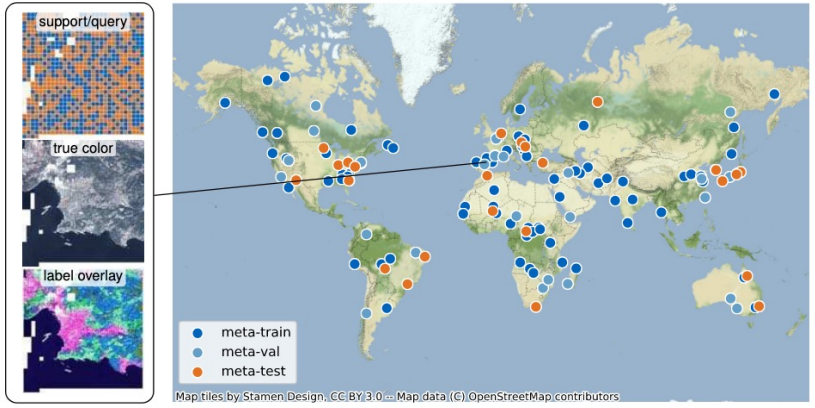


Principal component analysis (PCA) on VGG-16 features of cropland images from different countries. Representations of the same class vary geographically; applying models trained on one geography to another would violate the assumption in traditional supervised learning that train and test distributions are the same.

Rußwurm et al. 2020

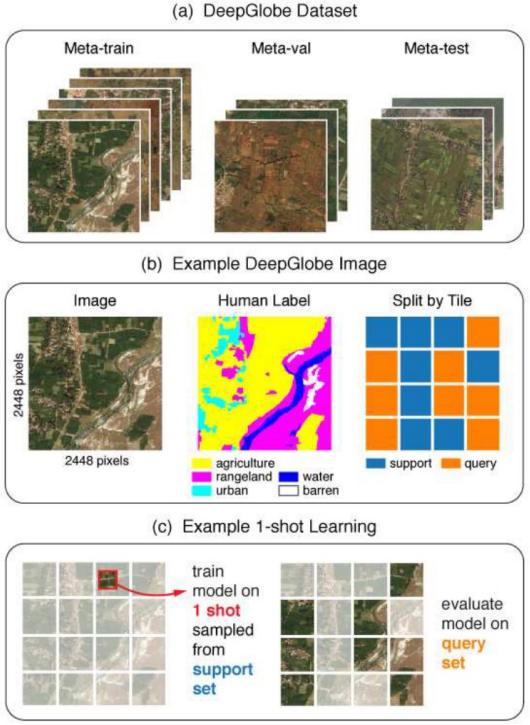
Example Meta-Learning for Few-Shot Land Cover Classification

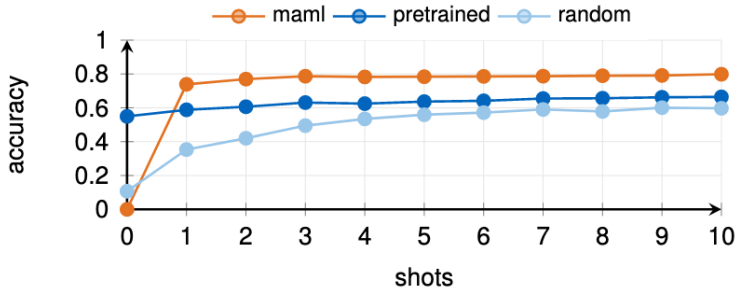




Rußwurm et al. 2020

Example Meta-Learning for Few-Shot Land Cover Classification





Rußwurm et al. 2020

Example Meta-Learning for Carbon Fluxes

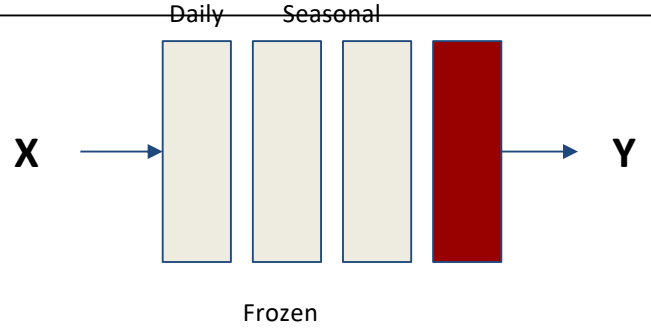
Nathaniel et al. 2022 Sci Data

Example Meta-Learning for Carbon Fluxes

Data processing
Balancing the data

Idea: get the same number of data across classes

Problem: reduce sample size significantly



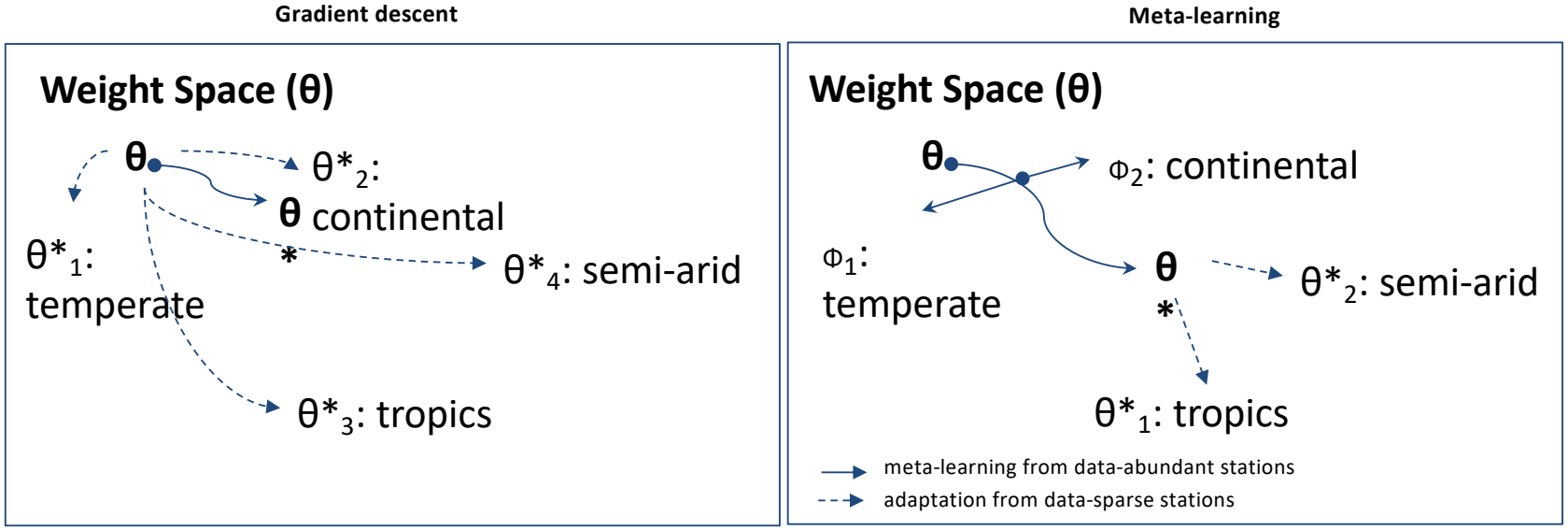
Modeling
Transfer learning

Idea: train a model on classes with a lot of data, and transfer the learned knowledge to new domains (probably with few data)

Problem(s):
(i) freeze/unfreeze which part of the model?
(ii) assumes frozen parts of the model are not changing across tasks → not capturing the right representation

Nathaniel et al. 2022 Sci Data

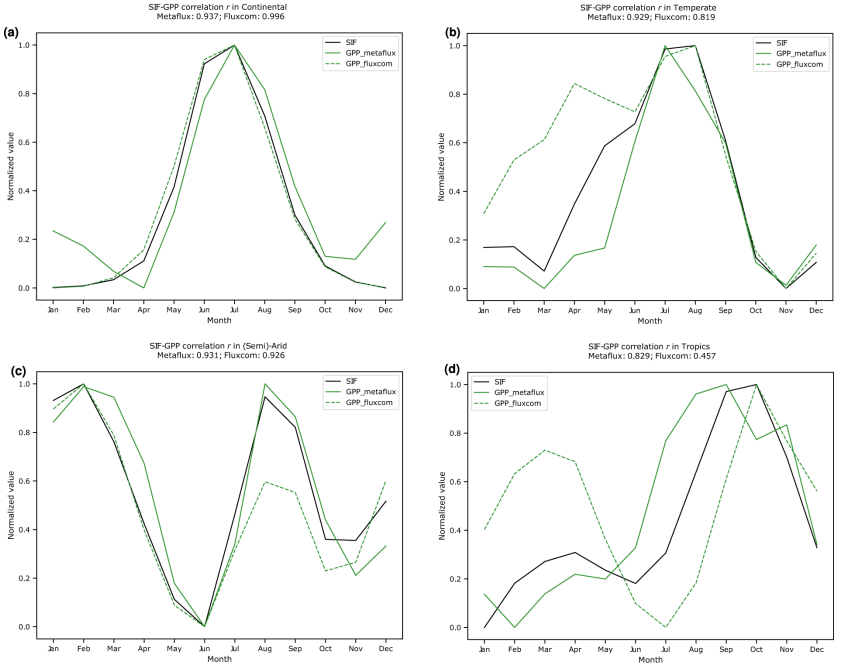
Example Meta-Learning for Carbon Fluxes



Nathaniel et al. 2022 Sci Data

Example Meta-Learning for Carbon Fluxes

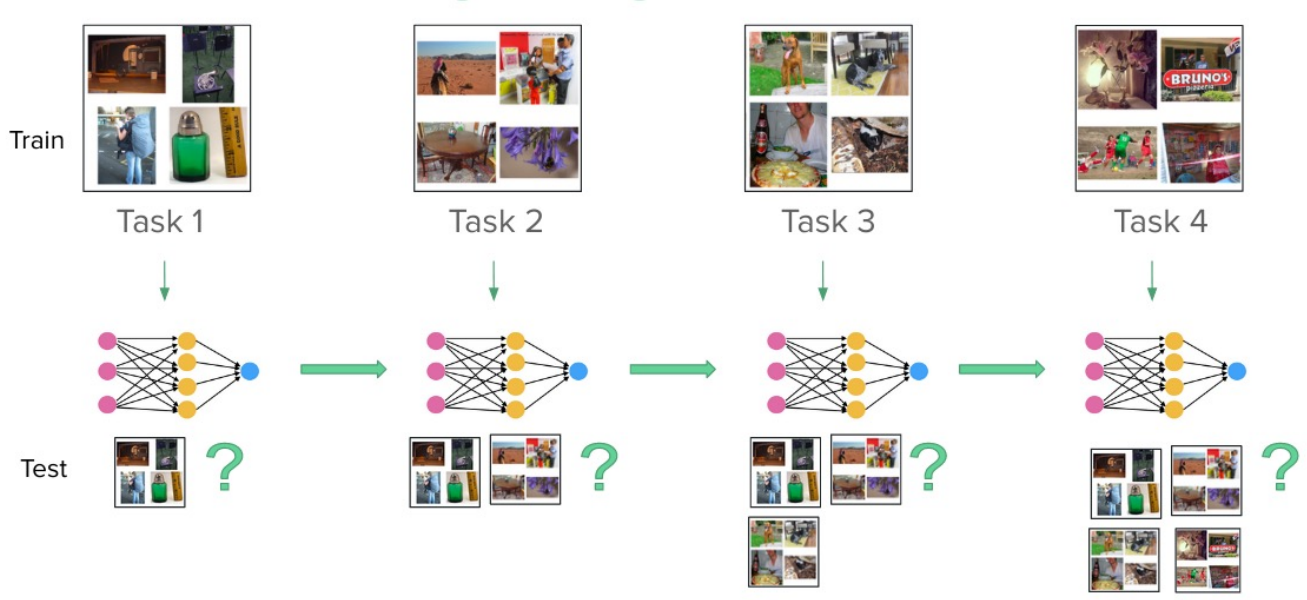
Climate zones	r($GPP_{fluxcom}$, SIF)	r($GPP_{metaflux}$, SIF)
Semi-arid	0.726 ± 0.165	$\mathbf{0.856 \pm 0.083}$
Continental	$\mathbf{0.965 \pm 0.002}$	0.937 ± 0.012
Temperate	0.826 ± 0.021	$\mathbf{0.919 \pm 0.002}$
Tropics	0.343 ± 0.164	$\mathbf{0.546 \pm 0.299}$



Nathaniel et al. 2022 Sci Data

La MAML

Continual learning (CL) problem involves training models with limited capacity to perform well on a set of an unknown number of *sequentially arriving tasks*. Data re not i.i.d when they arrive sequentially.

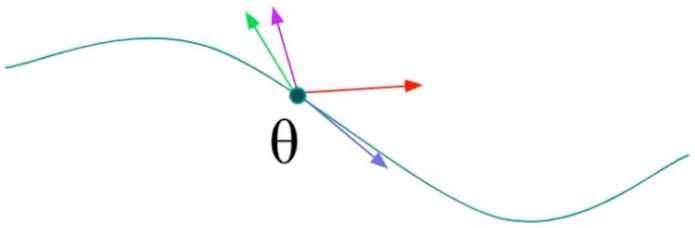


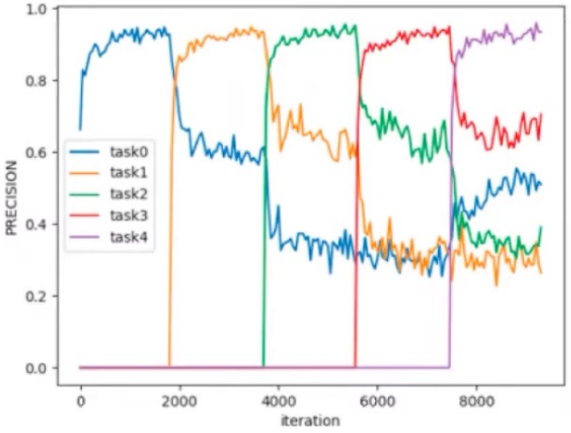
Gupta et al. 2020

Potential issue: forget all tasks (intefreference)

Catastrophic Forgetting

Gradient directions for various tasks at any time *t*

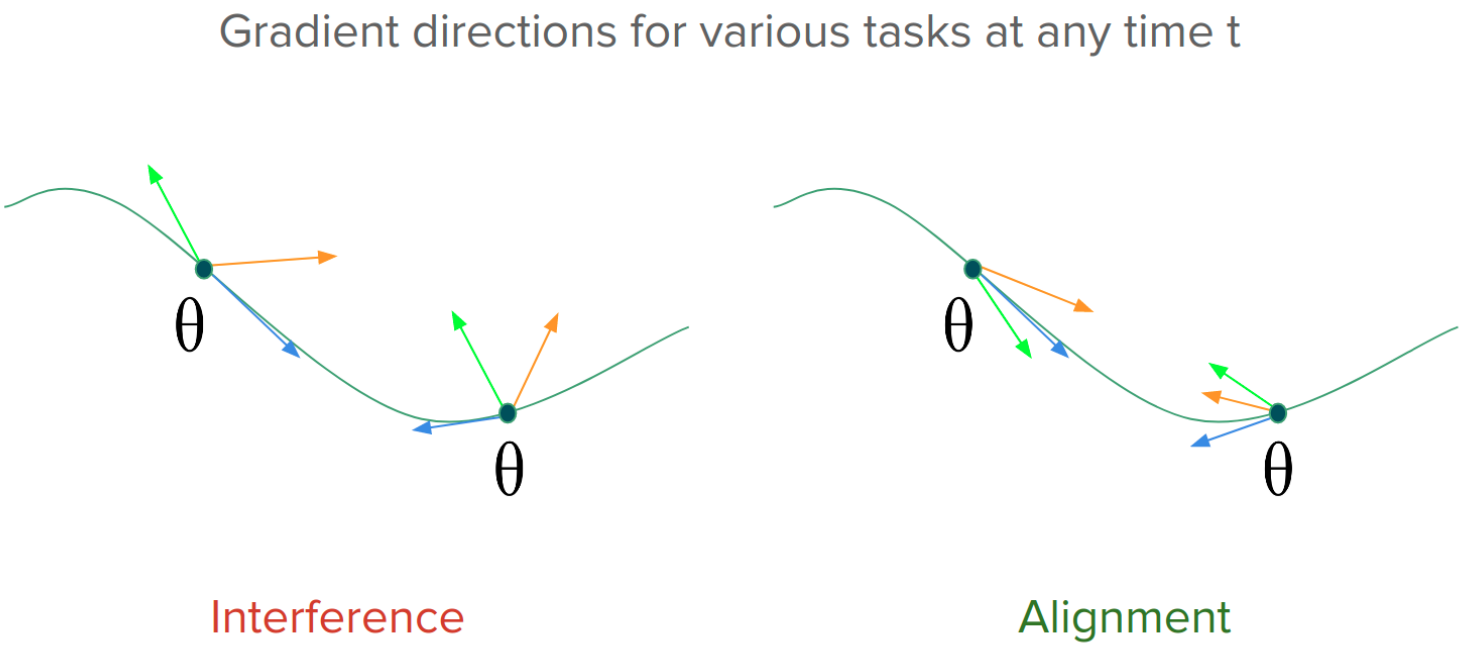




Rußwurm et al. 2020

MetaLearning

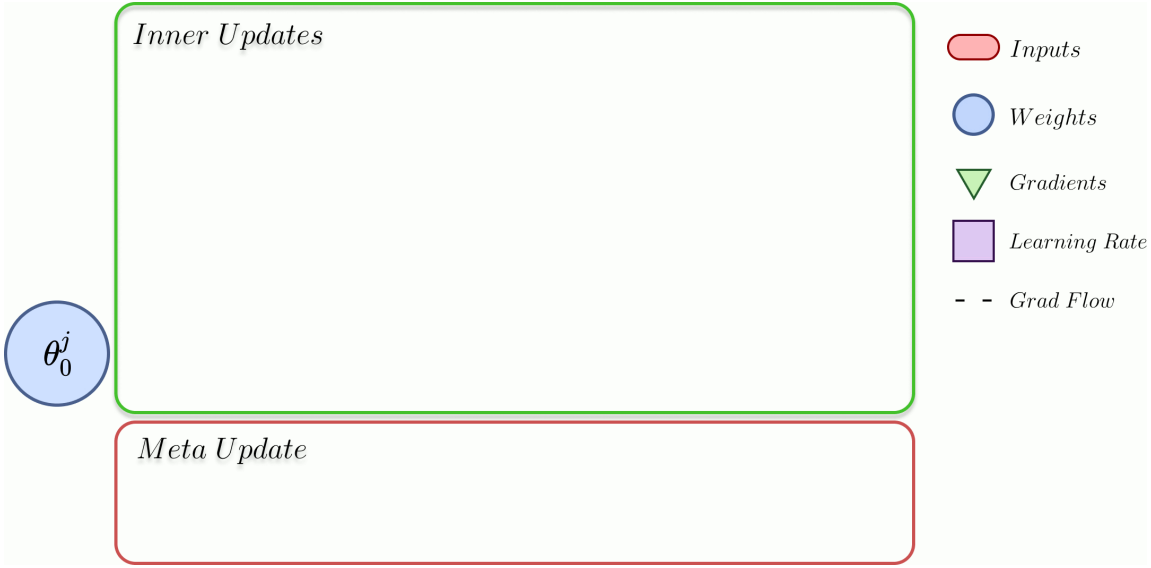
Potential issue: forget all tasks (interference), gradient-alignment is essential to make shared progress on task-wise objectives under limited availability of training-data.



Rußwurm et al. 2020

La MAML

La-MAML algorithm: For every batch of data, the initial weights undergo a series of k fast updates to obtain θjk (here $j = 0$), which is evaluated against a meta-loss to backpropagate gradients with respect to the weights $\theta 00$ and learning rates $\alpha 0$. First $\alpha 0$ is updated to $\alpha 1$ which is then used to update $\theta 00$ to $\theta 10$. The blue boxes indicate fast weights while the green boxes indicate gradients for the slow updates. LRs and weights are updated in an asynchronous manner.



Gupta et al. 2020

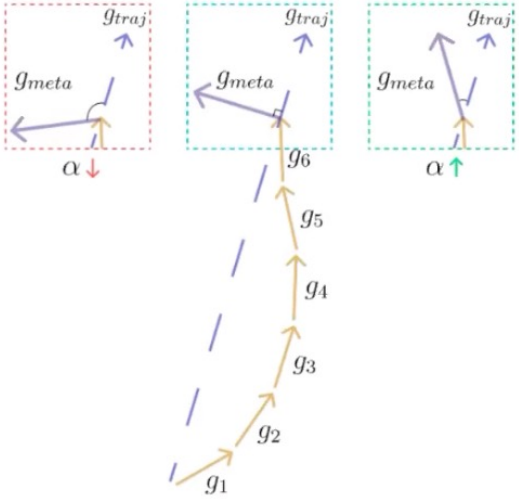
La MAML

Gradient of Learning rates (alpha) :

$$g_{\mathrm{MAML}}(\alpha) = \frac{\partial}{\partial \alpha} L_m(\theta_k)$$
$$= \left(\frac{\partial}{\partial \theta_k} L_m(\theta_k)\right)\left(-\sum_{i=0}^{k-1} \frac{\partial}{\partial \theta_i} L_i(\theta_i)\right)$$

Outer Gradient — Inner Gradient

Old + New Tasks — New Task



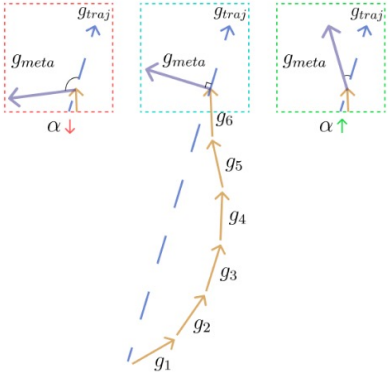
Gupta et al. 2020

La MAML

Updating the network weights and LRs asynchronously in the meta-update
Two losses to simulate continual learning in the inner loop and test forgetting in the outer loop is referred to as the Online-Aware Meta Learning objective.

Algorithm 1 La-MAML : Look-ahead MAML

Input: Network weights θ, LRs α, inner objective ℓ, meta objective L, learning rate for $\alpha : \eta$
$j \leftarrow 0$, $R \leftarrow \{\}$ ▷ Initialise Replay Buffer
for $t := 1$ **to** T **do**
  for $ep := 1$ **to** num_{epochs} **do**
    for batch b **in** $(X^t, Y^t) \sim D_t$ **do**
      $k \leftarrow sizeof(b)$
      $b_m \leftarrow Sample(R) \cup b$
      for $n = 0$ **to** $k - 1$ **do**
        Push $b[k']$ to R with reservoir sampling
        $\theta^j_{k'+1} \leftarrow \theta^j_{k'} - \alpha^j \cdot \nabla_{\theta^j_{k'}}$
      end for
      $\alpha^{j+1} \leftarrow \alpha^j - \eta \nabla_{\alpha^j} L_t(\theta^j_k, b_m)$
      $\theta^{j+1}_0 \leftarrow \theta^j_0 - max(0, \alpha^{j+1}) \cdot \nabla_{\theta^j_0} L_t(\theta^j_k, b_m)$
      $j \leftarrow j + 1$
    end for
  end for
end for



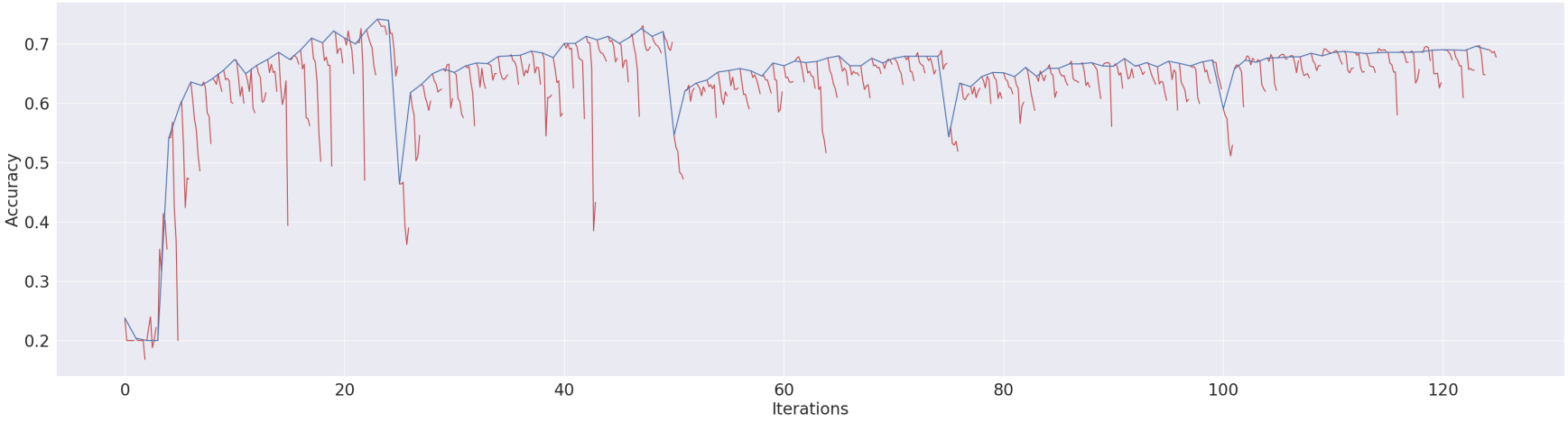
(a)
(b)

Figure 2: Different scenarios for the alignment of g_{traj} (blue dashed line) and g_{meta}, going from interference (left) to alignment (right). Yellow arrows denote the *inner updates*. The LR α increases (decreases) when gradients align (interfere).

Gupta et al. 2020

La MAML

Updating the network weights and LRs asynchronously in the meta-update
Two losses to simulate continual learning in the inner loop and test forgetting in the outer loop is referred to as the Online-Aware Meta Learning objective.



Retained Accuracy (RA) for La-MAML plotted every 25 meta-updates

Gupta et al. 2020

Reinforcement-Learning

Reinforcement learning

Example for video games

Gupta et al. 2020

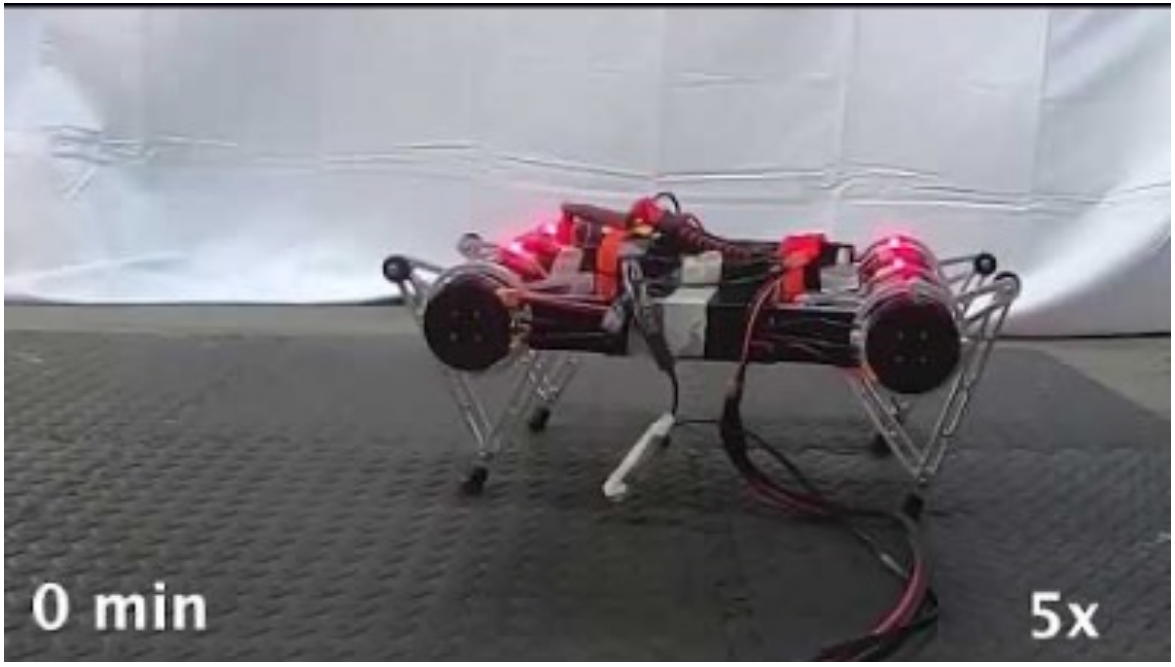
Reinforcement learning

Example for walking robots

Gupta et al. 2020

Reinforcement learning

Example for walking robots



Gupta et al. 2020

Reinforcement learning: learning from rewards

- The **Reward** Rt defines the goal of an RL problem.
- Gives the **agent** a sense of what is good and bad.
- A reward of a higher magnitude is better.
- Usually a function of **environment** state (situation).

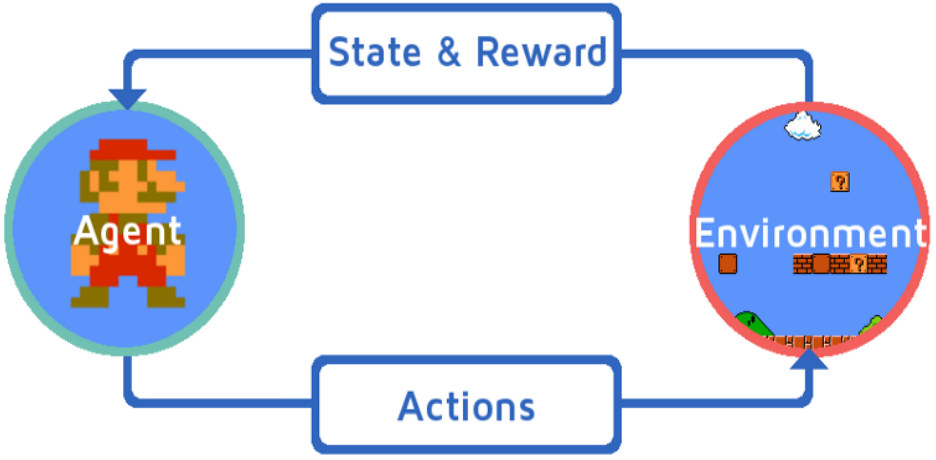
- Explore and exploit:

  Reinforcement Learning is like trial and error
- The agent should **explore** the environment to search for better policies.
- After the selection of the optimal **policy**, the agent maximizes the rewards.
  - Exploration finds more information about the environment.
  - Exploitation uses known information to maximize reward.

Gupta et al. 2020

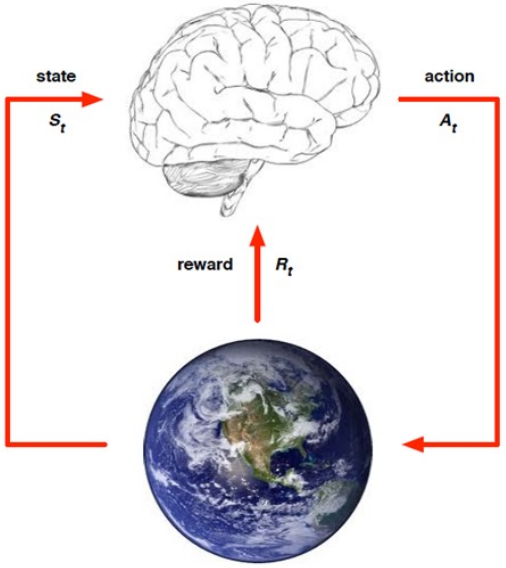
Reinforcement learning: learning from rewards

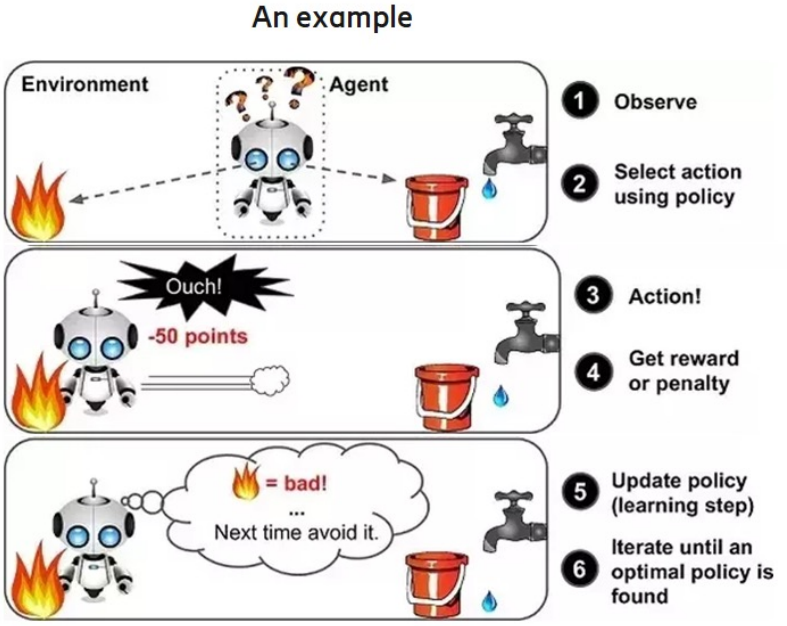
- The RL Agent receives **state S^0** from the **environment** i.e. Mario
- Based on that **state S^0,** the RL agent takes an **action A^0**, say — our RL agent moves right. Initially, this is random.
- Now, the environment is in a new state **S^1** (new frame from Mario or the game engine)
- Environment gives some **reward R^1** to the RL agent. It probably gives a +1 because the agent is not dead yet.
- This RL loop continues until we are dead or we reach our destination, and it continuously outputs a sequence of **state, action and reward.**
- Aim of our RL agent is to maximize the reward



Gupta et al. 2020

Example

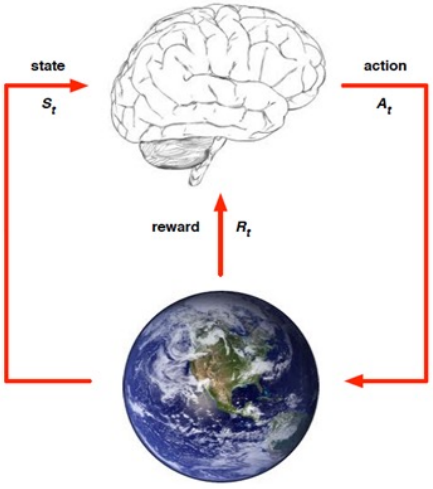




Gupta et al. 2020

RL Agent Basics

- An RL agent may include one or more of these components
  - Policy: agent's behaviour function
  - Value function: how good is each state and/or action
  - Model: agent's representation of the environment



Characteristics

What makes reinforcement learning different from other machine learning paradigms?

- There is no supervisor, only a *reward* signal
- Feedback is delayed, not instantaneous
- Time really matters (sequential, non i.i.d data)
- Agent's actions affect the subsequent data it receives

Gupta et al. 2020

Reward maximization

•The cumulative rewards at each time step with the respective action is written as:

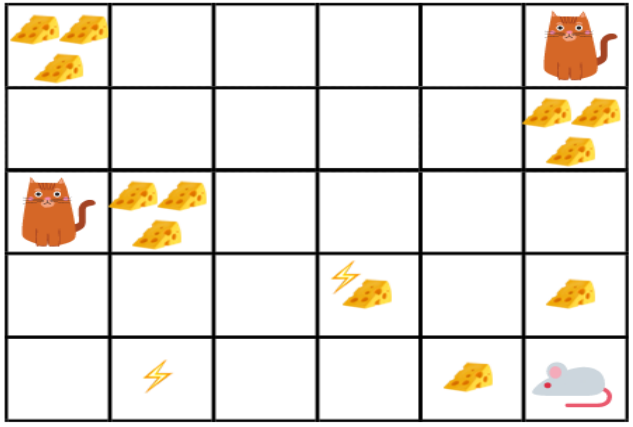
$$G_t = \sum_{k=0}^{T} R_{t+k+1}$$

•But discounting might be important: reward "closer" might be discounted (like yield/rate)

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} where \gamma \in [0,1)$$

$$R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots$$

- Sigma (Sum up)
- Discount rate
- Rewards received at each state
- Expanded form of the Equation



Better eat everything before being eaten by cat

Gupta et al. 2020

Reinforcement learning approaches

1. Policy-based approach

   Mapping each state to the best action (deterministic or stochastic- gives a distribution)

   $$a = \pi(s)$$

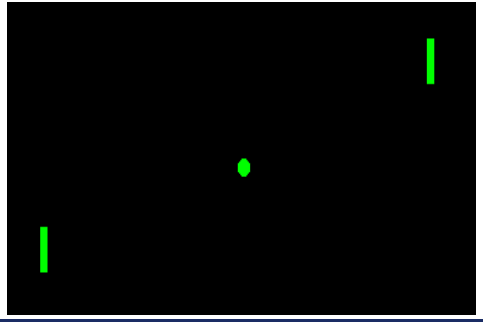
   a: actions. S: state, π : Policy function

2. Value-based

   optimize the value function: maximum expected future reward

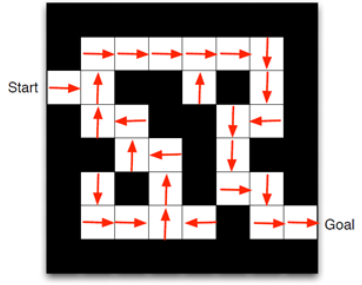
$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

Expected | Reward discounted | Given that state



Policy, Value Function & Model

- A policy is the agent's behaviour
- It is a map from state to action, e.g.
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$

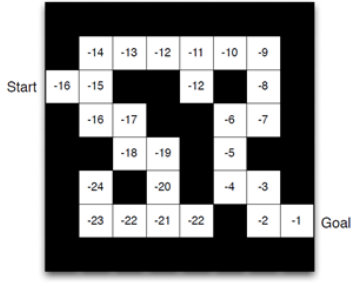


Policy

- Arrows represent policy $\pi(s)$ for each state s

- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s \right]$$

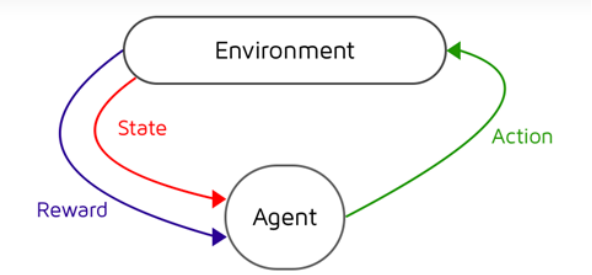


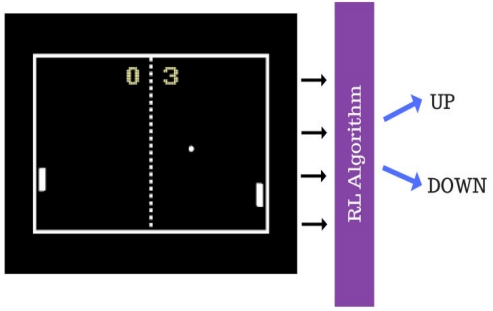
Value Function

- Numbers represent value $v_\pi(s)$ of each state s

Reinforcement learning approaches

Not all actions were bad, maybe only last moves





Do not discard entire sequence by only weight last ones negatively (due to sparsity in reward setting).