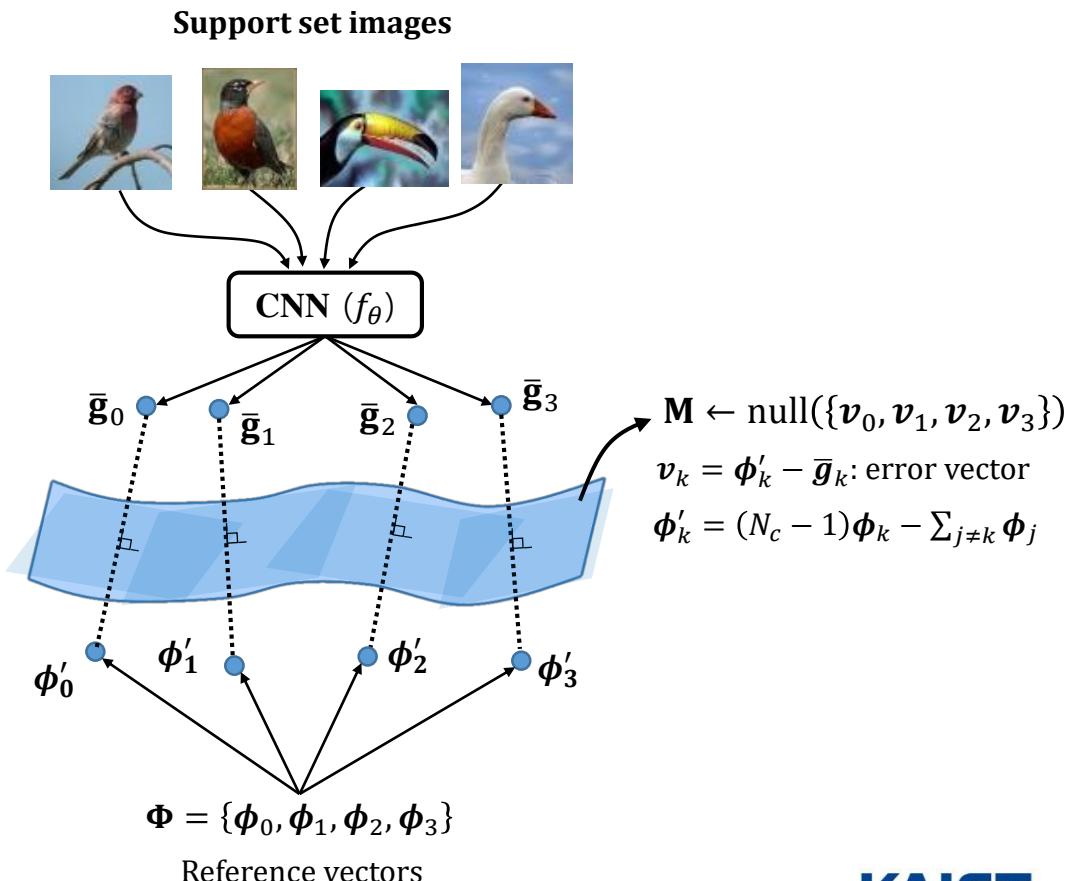
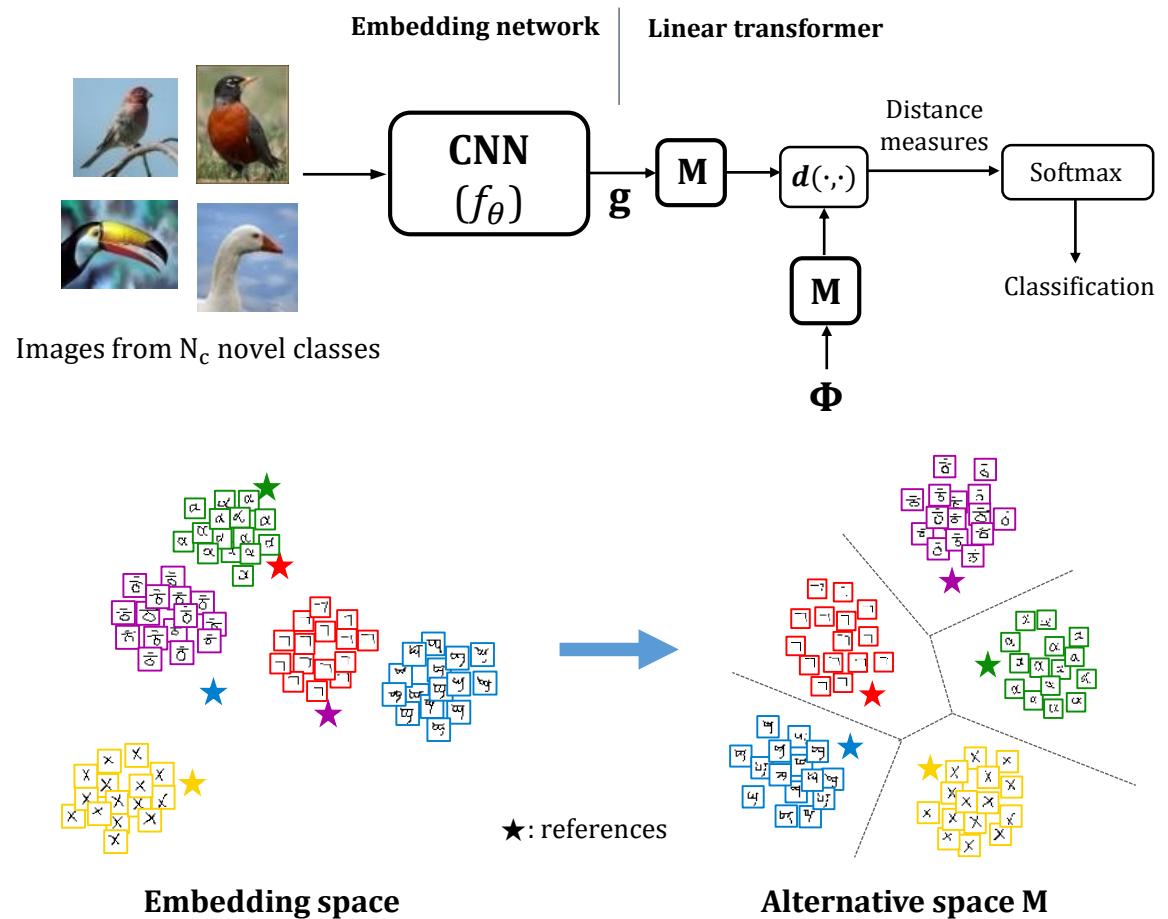


Meta-Learner with Linear Nulling

- An embedding network is combined with a linear transformer.
- The linear transformer carries out null-space projection on an alternative classification space.
- The projection space M is constructed to match the network output with a special set of reference vectors.



OBOE: Collaborative Filtering for AutoML Initialization

Chengrun Yang, Yuji Akimoto, Dae Won Kim, Madeleine Udell

Cornell University

Goal: Select models for a new dataset within time budget.

Given: Model performance and runtime on previous datasets.

Approach:

- ▶ **low rank** dataset-by-model collaborative filtering matrix
- ▶ **predict model runtime** using polynomials
- ▶ **classical experiment design** for cold-start
- ▶ missing entry imputation for model performance prediction

Performance:

- ▶ cold-start: high accuracy
- ▶ model selection: fast and perform well

Backpropamine: meta-learning with neuromodulated Hebbian plasticity

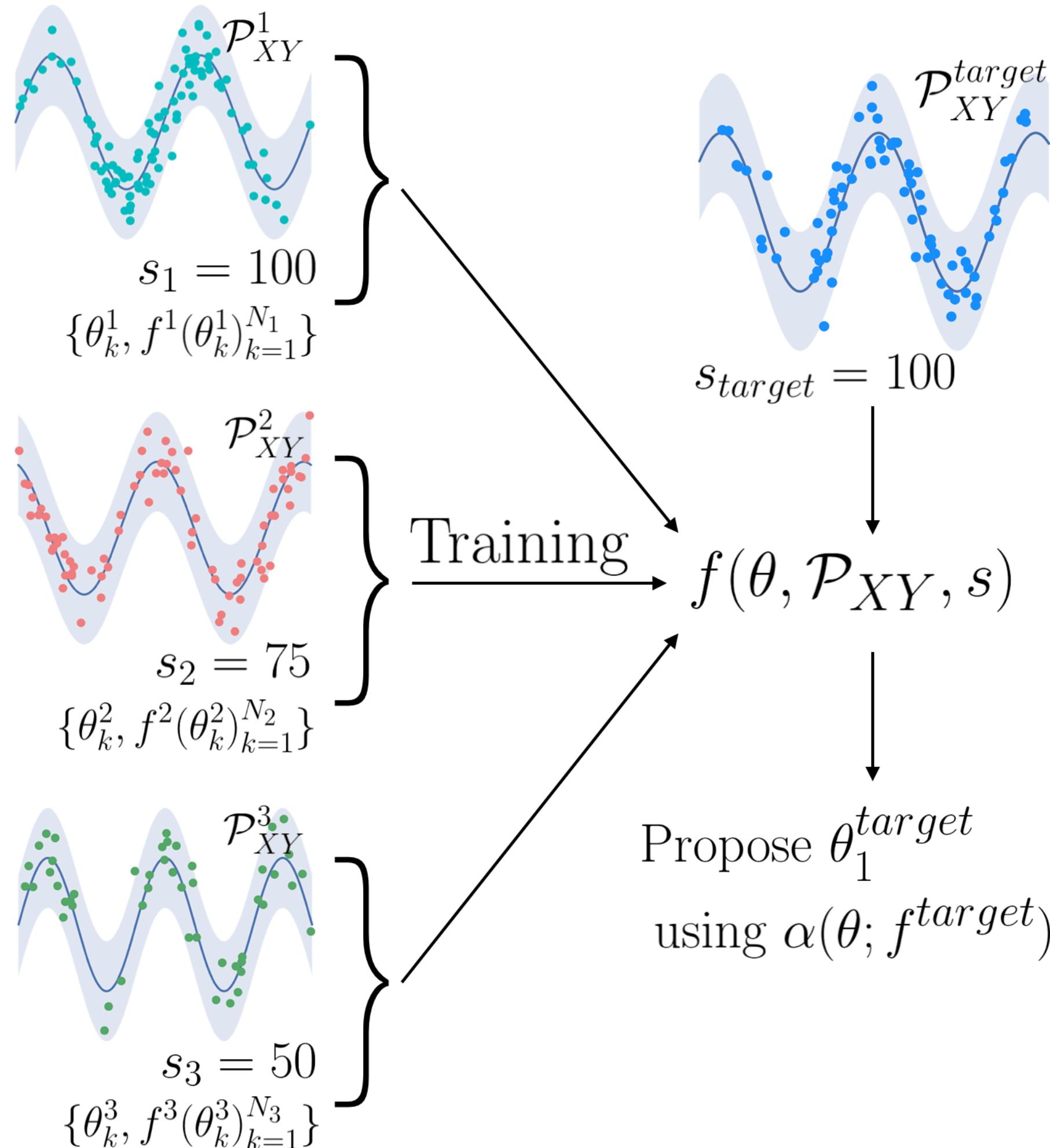
- **Differentiable plasticity:** meta-learning with Hebbian **plastic** connections
 - Meta-train both the baseline **weight** and **plasticity** of each connection to support efficient learning in any episode
- In nature, plasticity is under real-time control through **neuromodulators**
 - The brain can decide **when** and **where** to be plastic
- **Backpropamine = Differentiable plasticity + neuromodulation**
 - Make the **rate** of plasticity a real-time **output** of the network
 - During each episode, the network effectively learns by self-modification
- Results:
 - Solves tasks that non-modulated networks cannot
 - Improves LSTM performance on PTB language modeling task



Hyperparameter Learning via Distributional Transfer

Ho Chung Leon Law¹, Peilin Zhao², Junzhou Huang² and Dino Sejdinovic¹

¹University of Oxford and ²Tencent AI Lab



Goal (hyperparameter selection):

Optimise f^{target} (target objective) w.r.t θ :

$$\theta_{target}^* = \operatorname{argmax}_{\theta \in \Theta} f^{target}(\theta)$$

Scenario:

- We have n potentially related tasks f^i , $i = 1, \dots, n$
- For these tasks, we have $\{\theta_k^i, f^i(\theta_k^i)\}_{k=1}^{N_i}$ from past runs

Method:

- Assume training data D_i comes from distribution \mathcal{P}_{XY}^i
- Transfer information using embeddings of \mathcal{P}_{XY}^i
- Jointly model θ , \mathcal{P}_{XY} and sample size s



Toward Multimodal Model-Agnostic Meta-Learning

Risto Vuorio¹, Shao-Hua Sun², Hexiang Hu² & Joseph J. Lim²

University of Michigan¹

University of Southern California²

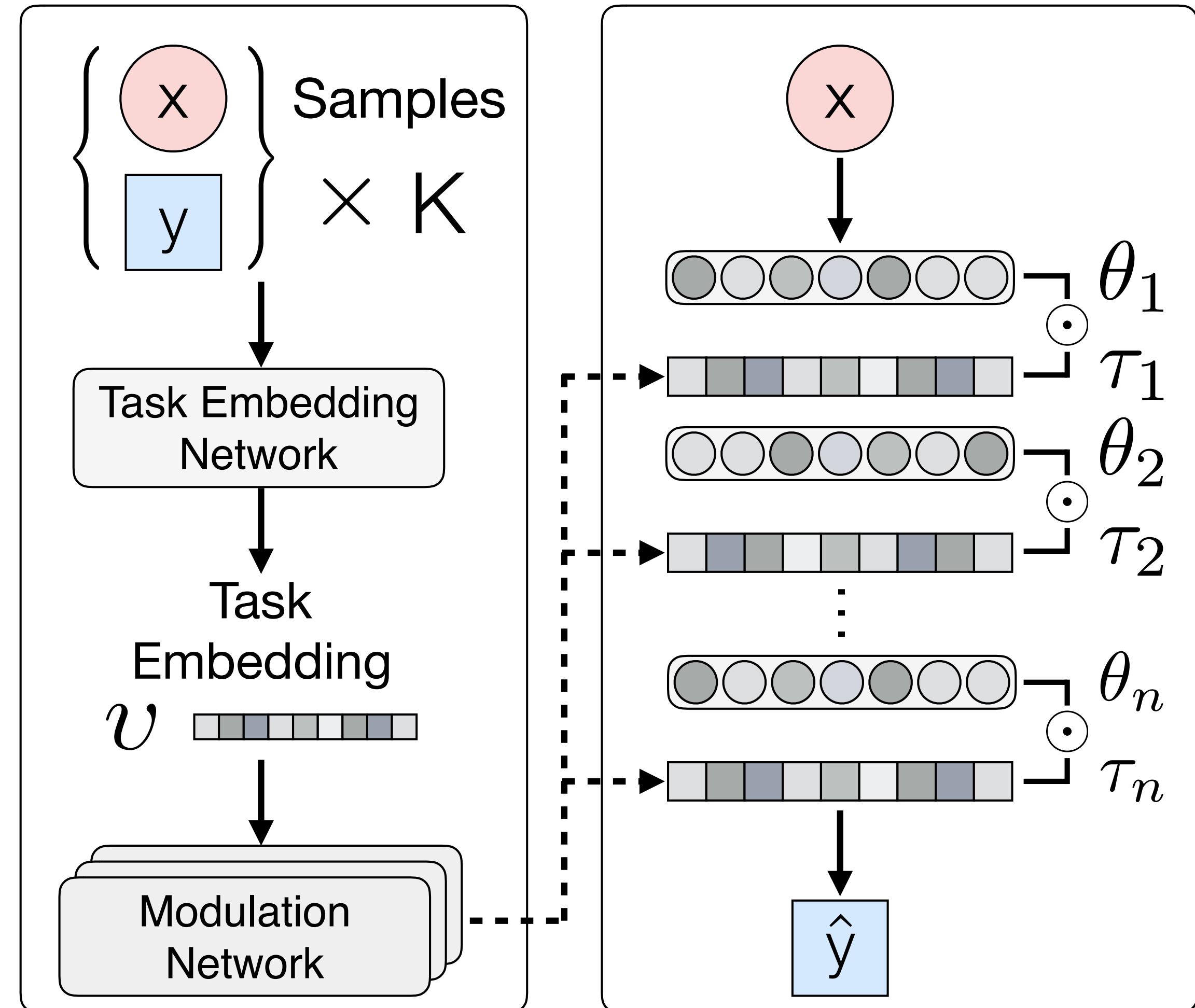


The limitation of the MAML family

- One initialization can be suboptimal for multimodal task distributions.

Multi-Modal MAML

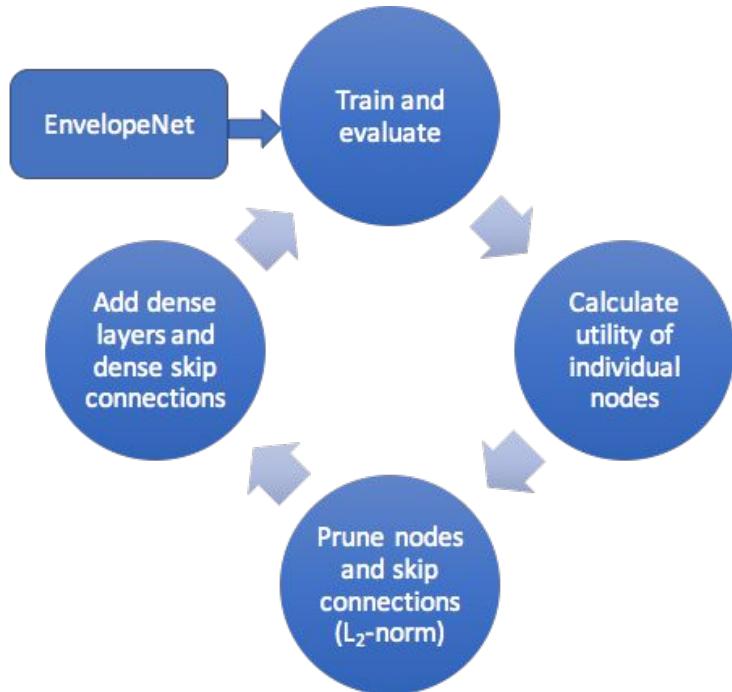
1. Model-based meta-learner computes task embeddings
2. Task embeddings are used to modulate gradient-based meta-learner
3. Gradient-based meta-learner adapts via gradient steps



Model-based
Meta-learner

Gradient-based
Meta-learner

Fast Neural Architecture Construction using EnvelopeNets

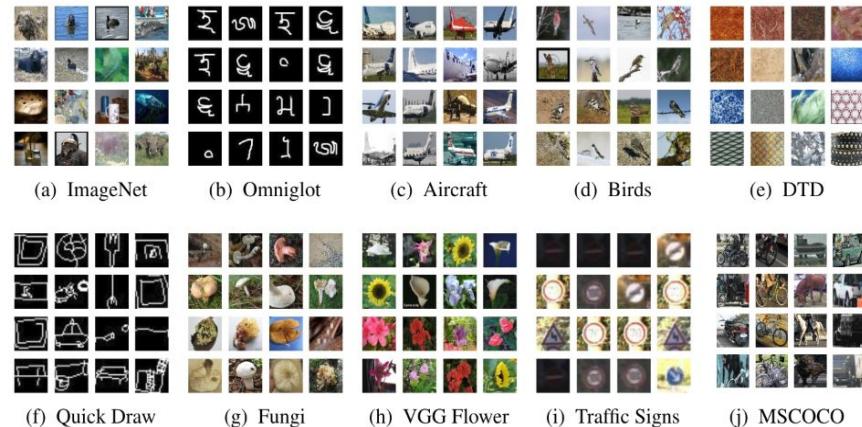


1. Finds architecture for CNNs in ~0.25 days
2. Based on the idea of utility of individual nodes.
3. Closely aligns with a theory of human brain ontogenesis.

Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, Hugo Larochelle

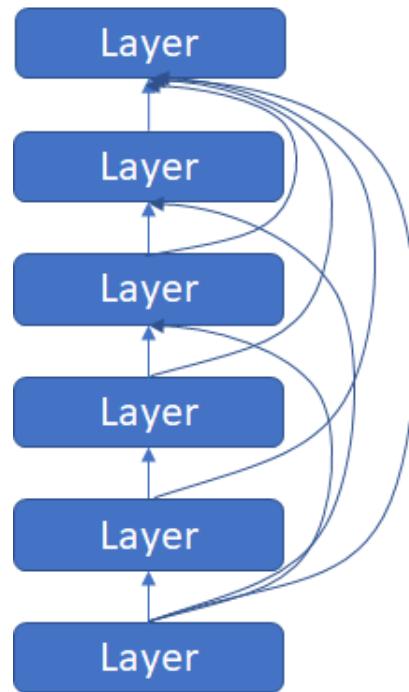
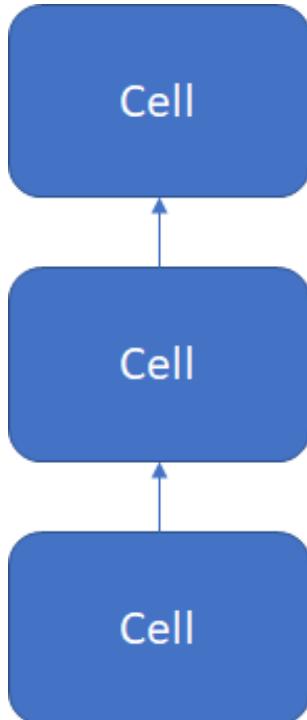
- New benchmark for **few-shot classification**
- Two-fold approach:
 1. **Change the data**
 - Large-scale
 - Diverse
 2. **Change the task creation**
 - Introduce imbalance
 - Utilize class hierarchy for ImageNet
- Preliminary results on: baselines, Prototypical Networks, Matching Networks, and MAML.
- Leveraging data of multiple sources remains an open and interesting research direction!



Macro Neural Architecture Search Revisited

Hanzhang Hu¹, John Langford², Rich Caruana², Eric Horvitz², Debadatta Dey²

¹Carnegie Mellon University, ²Microsoft Research



Cell Search: applies the found template on predefined skeleton.

Macro Search: learns all connections and layer types.

Cell Search: the predefined skeleton ensures the simplest cell search can achieve 4.6% error with 0.4M params on CIFAR 10.

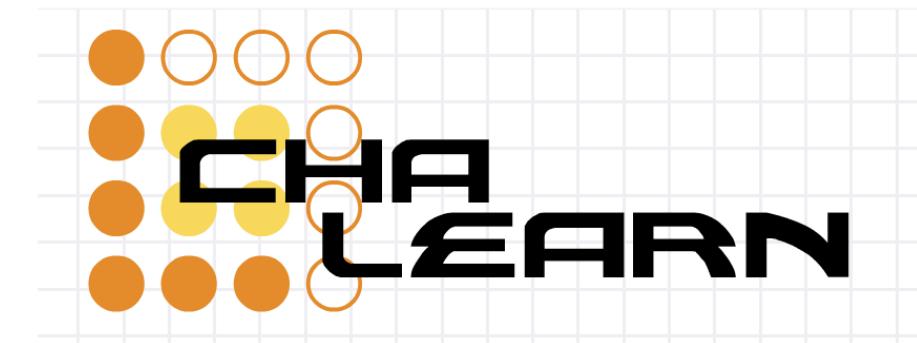
Key take-away: macro search can be competitive against cell search, even with simple random growing strategies, if the initial model is the same as cell search.

Microsoft®
Research



AutoDL 2019

Help Automating Deep Learning



Join the AutoDL challenge!

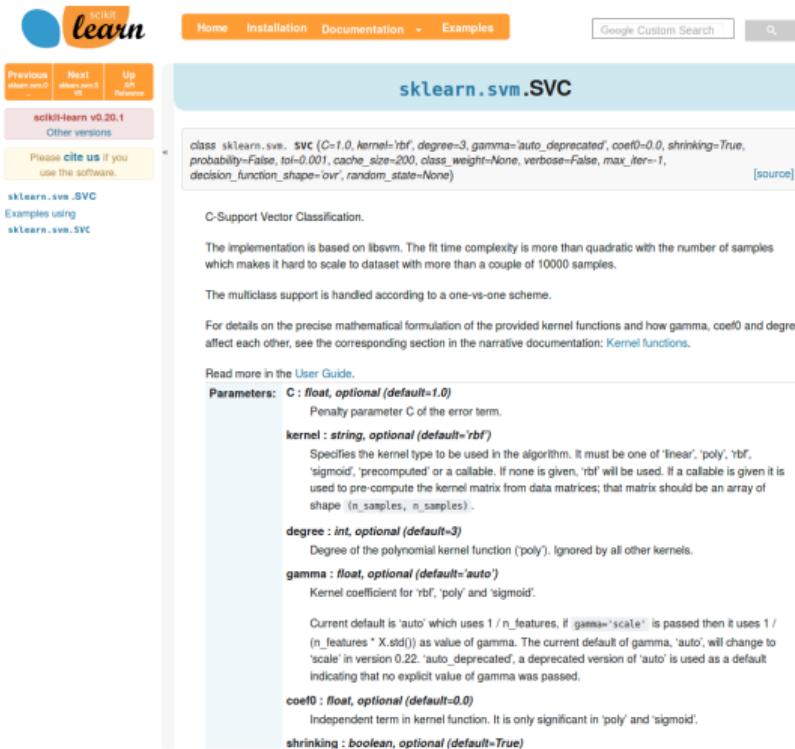
<https://autodl.chalearn.org>

AutoDL challenge design and beta tests

Zhengying Liu*, Olivier Bousquet, André Elisseeff, Sergio Escalera, Isabelle Guyon,
Julio Jacques Jr., Albert Clapés, Adrien Pavao, Michèle Sebag, Danny Silver,
Lisheng Sun-Hosoya, Sébastien Tréguer, Wei-Wei Tu, Yiqi Hu, Jingsong Wang, Quanming Yao

Meta Learning for Defaults – Symbolic Defaults

Jan N. van Rijn, Florian Pfisterer, Janek Thomas, Andreas Müller, Bernd Bischl, Joaquin Vanschoren



The screenshot shows the scikit-learn documentation for the `sklearn.svm.SVC` class. The top navigation bar includes links for Home, Installation, Documentation, Examples, Google Custom Search, and a search icon. On the left, there's a sidebar with links for Previous version (v0.20.1), Next version (v0.21), Up to Reference, and Other versions. A note encourages users to cite the software. The main content area has a light blue header for `sklearn.svm.SVC`. Below it, a code block shows the class definition:

```
class sklearn.svm. SVC (C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)
```

There is a link to [source]. The text below describes the class as C-Support Vector Classification. It notes that the implementation is based on libsvm, which makes it hard to scale to datasets with more than a couple of 10000 samples. It also mentions that multiclass support is handled according to a one-vs-one scheme. For mathematical details, it points to the Kernel functions section. A link to the User Guide is provided.

Parameters:

- C : float, optional (default=1.0)**: Penalty parameter C of the error term.
- kernel : string, optional (default='rbf')**: Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n_samples, n_samples).
- degree : int, optional (default=3)**: Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.
- gamma : float, optional (default='auto')**: Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. Current default is 'auto' which uses $1 / n_features$, if `gamma='scale'` is passed then it uses $1 / (n_features * X.std())$ as value of gamma. The current default of gamma, 'auto', will change to 'scale' in version 0.22. 'auto_deprecated', a deprecated version of 'auto' is used as a default indicating that no explicit value of gamma was passed.
- coef0 : float, optional (default=0.0)**: Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.
- shrinking : boolean, optional (default=True)**

- ▶ Defaults commonly used in Machine Learning research and practise

Meta Learning for Defaults – Symbolic Defaults

Jan N. van Rijn, Florian Pfisterer, Janek Thomas, Andreas Müller, Bernd Bischl, Joaquin Vanschoren

The screenshot shows the scikit-learn documentation for the `sklearn.svm.SVC` class. The top navigation bar includes links for Home, Installation, Documentation, Examples, Google Custom Search, and a search icon. On the left, there's a sidebar with links for Previous versions (v0.20.1), Next versions (v0.21.0), Up and Down (v0.20.1), and Other versions. Below that is a "Please cite us" section. The main content area has a title "sklearn.svm.SVC" and a code snippet showing the class definition:

```
class sklearn.svm. SVC (C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)
```

Below the code, there's a note about C-Support Vector Classification, a warning about quadratic fit time complexity, and a note about multiclass support. It also mentions that the implementation is based on libsvm. A note states that the implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples.

The parameters section lists several parameters with their types and default values:

- C : float, optional (default=1.0)**: Penalty parameter C of the error term.
- kernel : string, optional (default='rbf')**: Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n_samples, n_samples).
- degree : int, optional (default=3)**: Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.
- gamma : float, optional (default='auto')**: Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.
- coef0 : float, optional (default=0.0)**: Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.
- shrinking : boolean, optional (default=True)**: Whether to use the shrinking criterion.

- ▶ Defaults commonly used in Machine Learning research and practise
- ▶ Example: $\text{SVM}(C=1.0, \gamma=0.0125, \text{kernel}=RBF)$

Meta Learning for Defaults – Symbolic Defaults

Jan N. van Rijn, Florian Pfisterer, Janek Thomas, Andreas Müller, Bernd Bischl, Joaquin Vanschoren

The screenshot shows the scikit-learn documentation for the `sklearn.svm.SVC` class. The top navigation bar includes links for Home, Installation, Documentation, Examples, Google Custom Search, and a search icon. On the left, there are links for Previous (version 0.19), Next (version 0.20), Up (sklearn.svc.SVC), and Other versions. A sidebar on the left lists "Please cite us if you use the software.", "sklearn.svc.SVC", and "Examples using sklearn.svc.SVC". The main content area displays the code for `sklearn.svm.SVC`, followed by a detailed description of the class, parameters, and specific notes for the `kernel`, `degree`, and `gamma` parameters.

```
class sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)
```

C-Support Vector Classification.

The implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how gamma, coef0 and degree affect each other, see the corresponding section in the narrative documentation: [Kernel functions](#).

Read more in the [User Guide](#).

Parameters: `C : float, optional (default=1.0)`
Penalty parameter C of the error term.

`kernel : string, optional (default='rbf')`
Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n_samples, n_samples).

`degree : int, optional (default=3)`
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

`gamma : float, optional (default='auto')`
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.
Current default is 'auto' which uses $1/n_features$, if `gamma='scale'` is passed then it uses $1/(n_features * X.std())$ as value of gamma. The current default of gamma, 'auto', will change to 'scale' in version 0.22. 'auto_deprecated', a deprecated version of 'auto' is used as a default indicating that no explicit value of gamma was passed.

`coef0 : float, optional (default=0.0)`
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

`shrinking : boolean, optional (default=True)`

- ▶ Defaults commonly used in Machine Learning research and practise
- ▶ Example: $\text{SVM}(C=1.0, \gamma=0.0125, \text{kernel}=RBF)$
- ▶ Goal: Defaults based on meta-feature
- ▶ Example: $\text{SVM}(C=85, \gamma=0.2 / \text{num. features}, \text{kernel}=RBF)$
- ▶ Classical form of meta-learning
- ▶ Question: How to find good symbolic defaults?
- ▶ Answer: Let's discuss this at our poster!

Modular meta-learning in abstract graph networks for combinatorial generalization



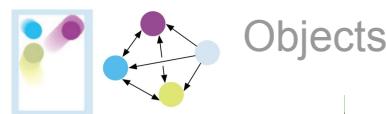
Ferran Alet, Maria Bauza, A. Rodriguez, T. Lozano-Perez, L. Kaelbling

code&pdf:alet-etal.com

Combinatorial generalization: generalizing by reusing neural modules

Graph Neural Networks

Nodes tied to entities



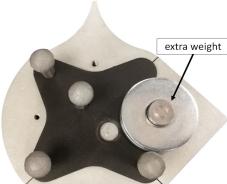
Particles



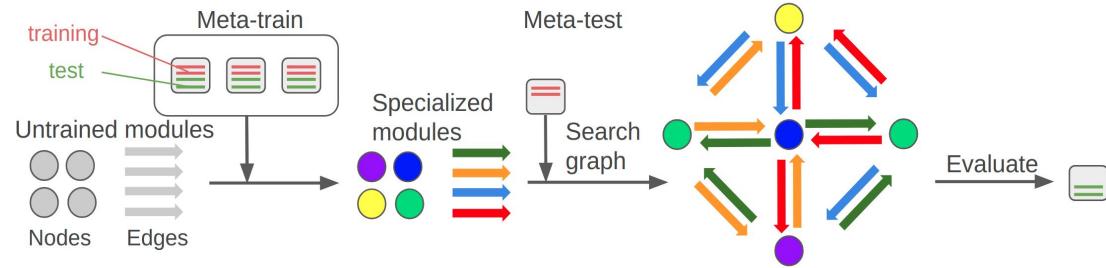
We introduce: **Abstract Graph Networks**

nodes are not tied to concrete entities

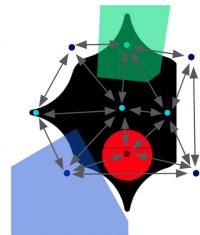
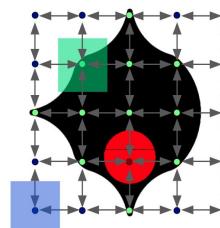
OmniPush dataset



Modular meta-learning



Graph Element Networks



Cross-Modulation Networks For Few-Shot Learning

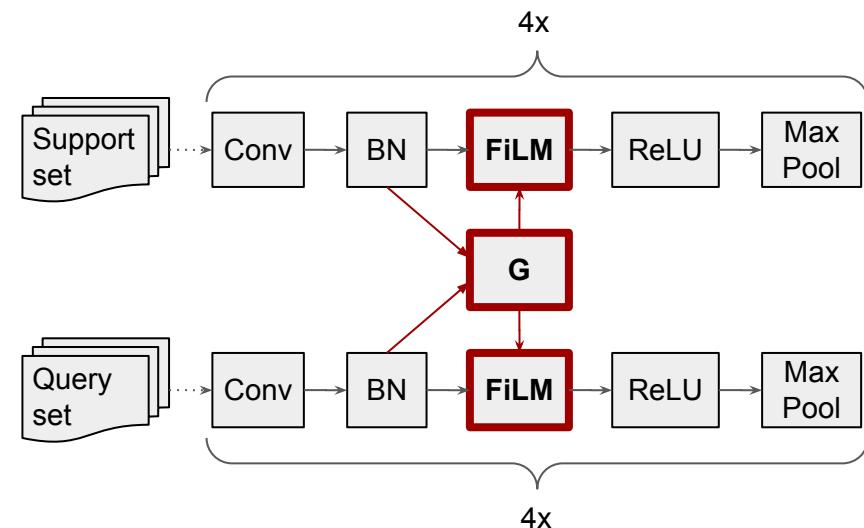
Hugo Proli[†], Vincent Dumoulin[‡],
and Luis Herranz[†]

[†] Computer Vision Center, Univ. Autònoma de Barcelona
[‡] Google Brain

Key idea: allow support and query examples to interact at each level of abstraction.

Extending the feature extraction pipeline of Matching Networks:

- ★ Channel-wise affine transformations: $\text{FiLM}(\mathbf{x}) = (1 + \gamma) \odot \mathbf{x} + \beta$
- ★ Subnetwork G predicts the affine parameters γ and β





Large Margin Meta-Learning for Few-Shot Classification

The University of Hong Kong¹, The Hong Kong Polytechnic University²



Yong Wang¹, Xiao-Ming Wu², Qimai Li², Jiatao Gu¹, Wangmeng Xiang², Lei Zhang², Victor O.K. Li¹

Large Margin Principle

$$\mathcal{L} = \mathcal{L}_{\text{softmax}} + \lambda * \mathcal{L}_{\text{large-margin}}$$

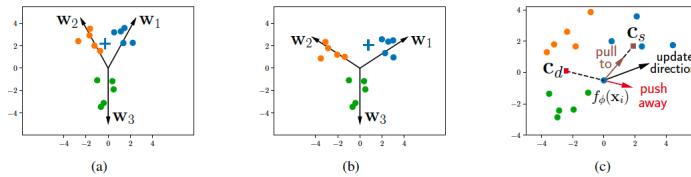


Fig. 1: Large margin meta-learning. (a) Classifier trained without the large margin constraint. (b) Classifier trained with the large margin constraint. (c) Gradient of the triplet loss.

One Implementation: Triplet Loss

$$\mathcal{L}_{\text{large-margin}} = \frac{1}{N_t} \sum_{i=1}^{N_t} [\| f_\phi(\mathbf{x}_i^a) - f_\phi(\mathbf{x}_i^p) \|_2^2 - \| f_\phi(\mathbf{x}_i^a) - f_\phi(\mathbf{x}_i^n) \|_2^2 + m]_+$$

Case study

- Graph Neural Network (GNN)
- Prototypical Network (PN)

Analysis

After rearrangement:

$$\mathcal{L}_{\text{large-margin}} = \frac{1}{N_t} \left(\sum_{\mathbf{x}_s \in S_s} \| f_\phi(\mathbf{x}_i) - f_\phi(\mathbf{x}_s) \|_2^2 - \sum_{\mathbf{x}_d \in S_d} \| f_\phi(\mathbf{x}_i) - f_\phi(\mathbf{x}_d) \|_2^2 \right) + \text{const.}$$

The gradient:

$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{large-margin}}}{\partial f_\phi(\mathbf{x}_i)} &= \frac{2}{N_t} \left(\sum_{\mathbf{x}_s \in S_s} (f_\phi(\mathbf{x}_i) - f_\phi(\mathbf{x}_s)) - \sum_{\mathbf{x}_d \in S_d} (f_\phi(\mathbf{x}_i) - f_\phi(\mathbf{x}_d)) \right) \\ &= -\frac{2|S_s|}{N_t} \left(\frac{1}{|S_s|} \sum_{\mathbf{x}_s \in S_s} f_\phi(\mathbf{x}_s) - f_\phi(\mathbf{x}_i) \right) - \frac{2|S_d|}{N_t} \left(f_\phi(\mathbf{x}_i) - \frac{1}{|S_d|} \sum_{\mathbf{x}_d \in S_d} f_\phi(\mathbf{x}_d) \right) \\ &= -\underbrace{\frac{2|S_s|}{N_t} (c_s - f_\phi(\mathbf{x}_i))}_{\text{pull towards its own class}} - \underbrace{\frac{2|S_d|}{N_t} (f_\phi(\mathbf{x}_i) - c_d)}_{\text{push away from other classes}} . \end{aligned}$$

Features

- We implement and compare several of other large margin methods for few-shot learning.
- Our framework is simple, efficient, and can be applied to improve existing and new meta-learning methods with very little overhead.

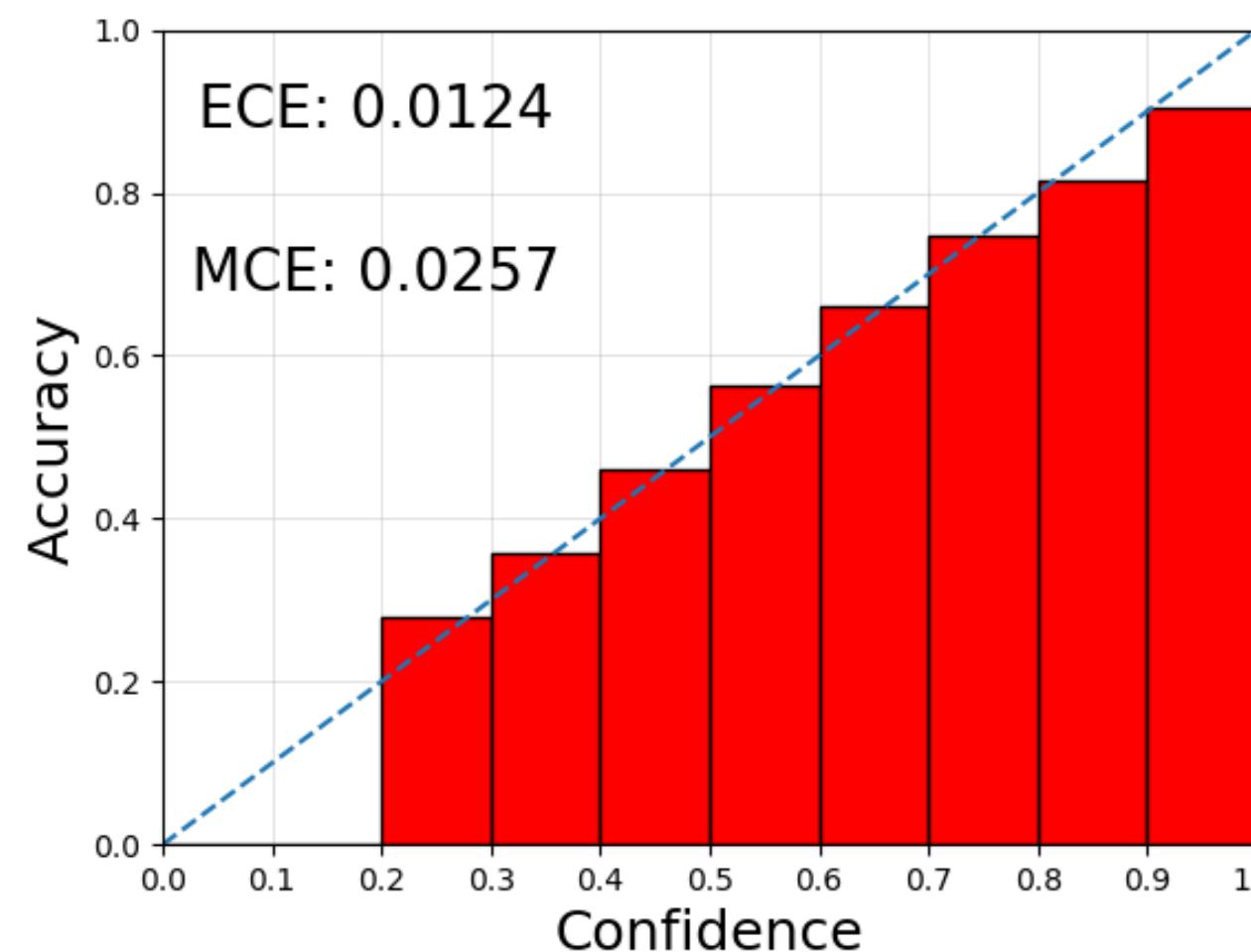


Amortized Bayesian Meta-Learning

Sachin Ravi & Alex Beatson

Department of Computer Science, Princeton University

- ▶ Lot of progress in few-shot learning but under controlled settings
- ▶ In real world, relationship between training and testing tasks can be tenuous
 - ▶ Task-specific predictive uncertainty is crucial
- ▶ We present gradient-based meta-learning method for computing task-specific approximate posterior
- ▶ Show that method displays good predictive uncertainty on contextual-bandit and few-shot learning tasks



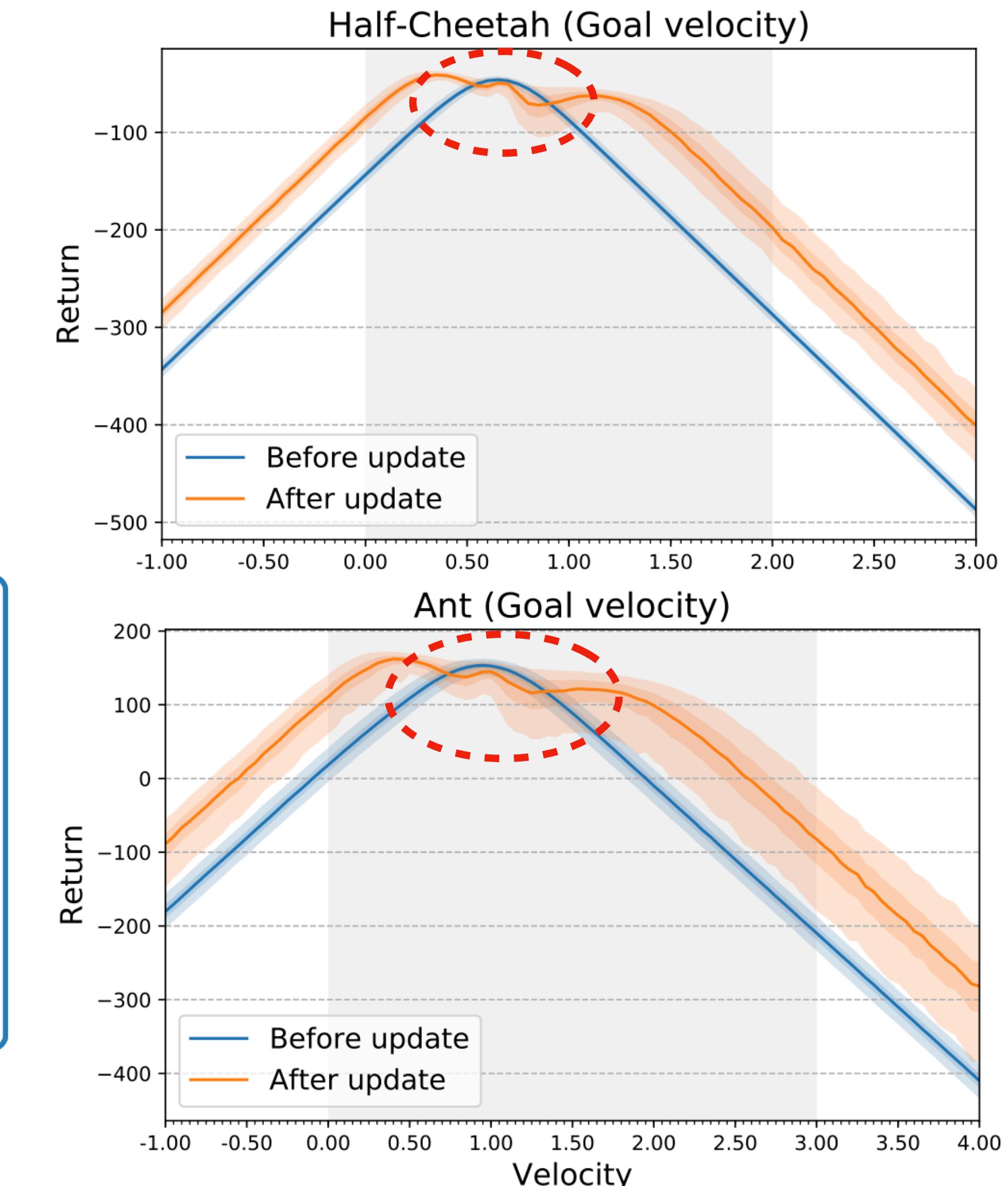
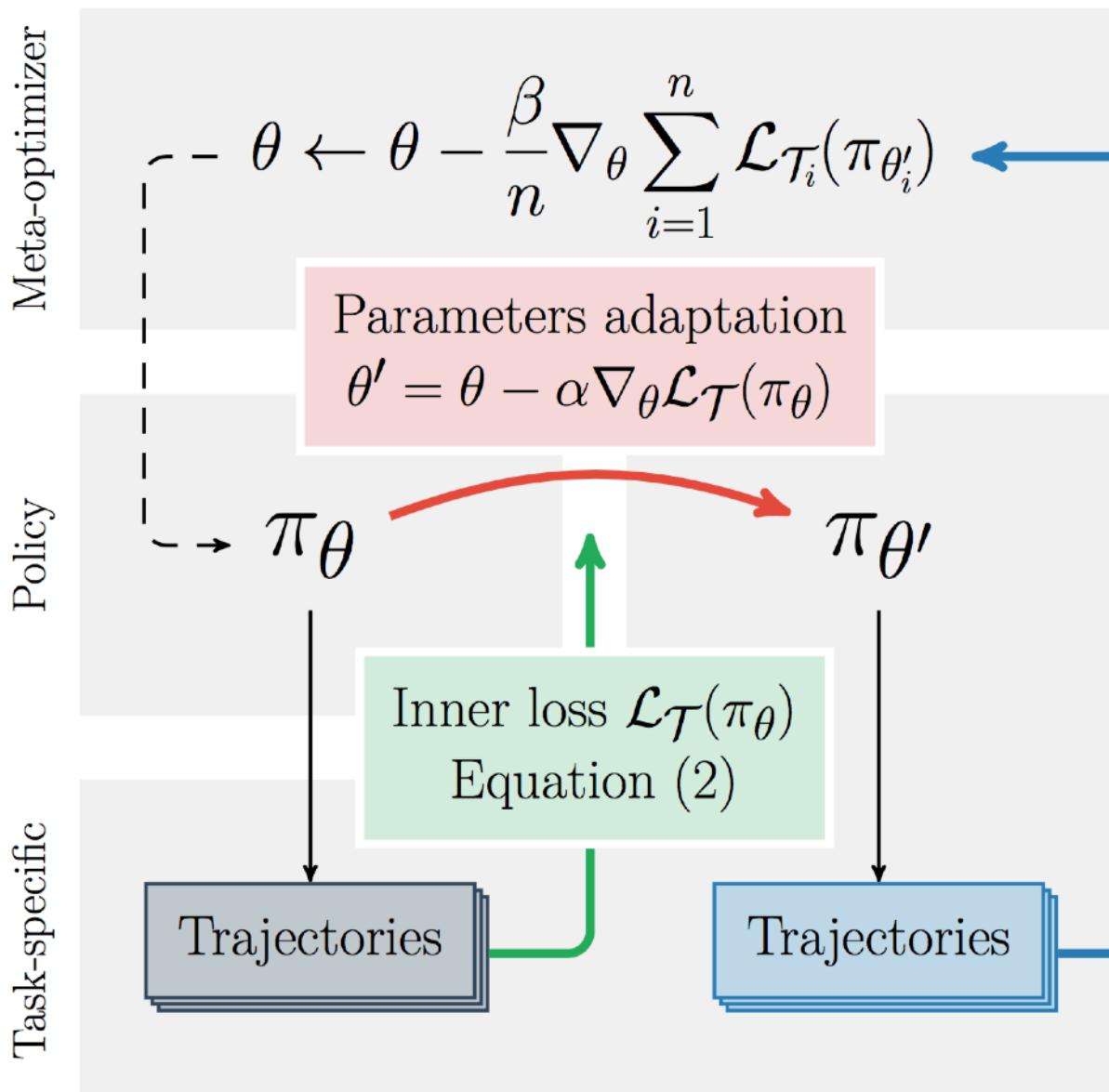
The effects of negative adaptation in Model-Agnostic Meta-Learning

Tristan Deleu, Yoshua Bengio

- The advantage of meta-learning is well-founded under the assumption that **the adaptation phase does improve the performance** of the model on the task of interest
- Optimization: maximize the performance after adaptation, **performance improvement is not explicitly enforced**

$$\min_{\theta} \mathbb{E}_{\mathcal{T} \sim p(\mathcal{T})} [\mathcal{L}(\theta'_{\mathcal{T}}; \mathcal{D}'_{\mathcal{T}})]$$

- We show empirically that performance **can decrease** after adaptation in MAML. We call this **negative adaptation**
- How to fix this issue? Ideas from **Safe Reinforcement Learning**



Mitigating Architectural Mismatch During the Evolutionary Synthesis of Deep Neural Networks

Audrey G. Chung, Paul Fieguth, Alexander Wong

- *Evolutionary deep intelligence* for increasingly efficient networks
- Preliminary study into the effects of architectural alignment
- Like-with-like mating policy via gene tagging system
- Resulting networks are comparable:
 - Restricts search space exploration?
 - Compensated with training epochs?
 - ???



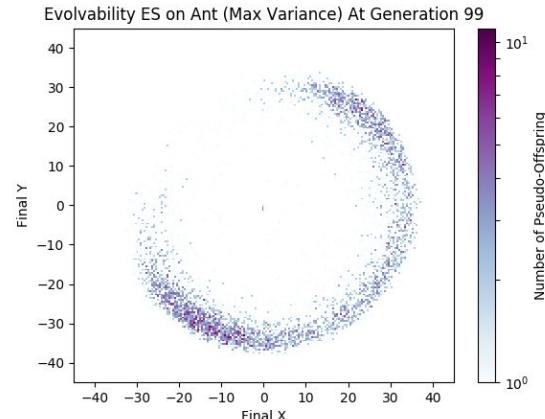
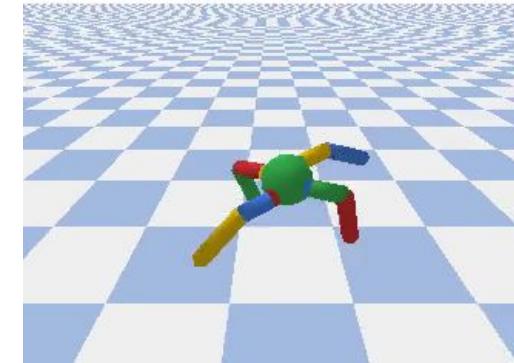
Evolvability ES: Scalable Evolutionary Meta-Learning



By Alexander Gajewski, Jeff Clune, Kenneth O. Stanley, and Joel Lehman

- Evolvability ES is a meta-learning algorithm inspired by Evolution Strategies [1]
- Surprisingly, Evolvability ES finds parameters such that at test time, **random** perturbations result in diverse behaviors
- In a simulated Ant locomotion domain, adding Gaussian noise to the parameters results in policies which move in many different directions

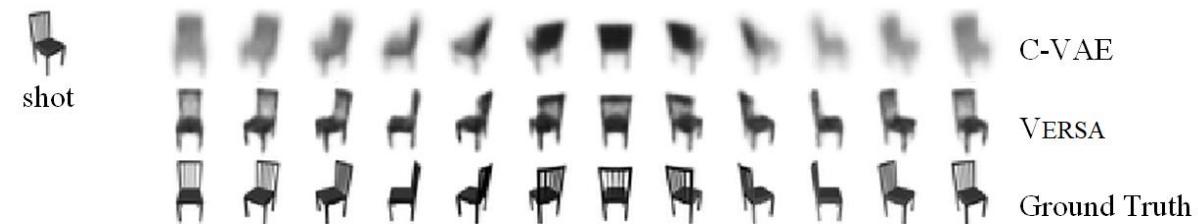
[1] Salimans et al., Evolution Strategies as a Scalable Alternative to Reinforcement Learning, 2017.



Consolidating the Meta-Learning Zoo

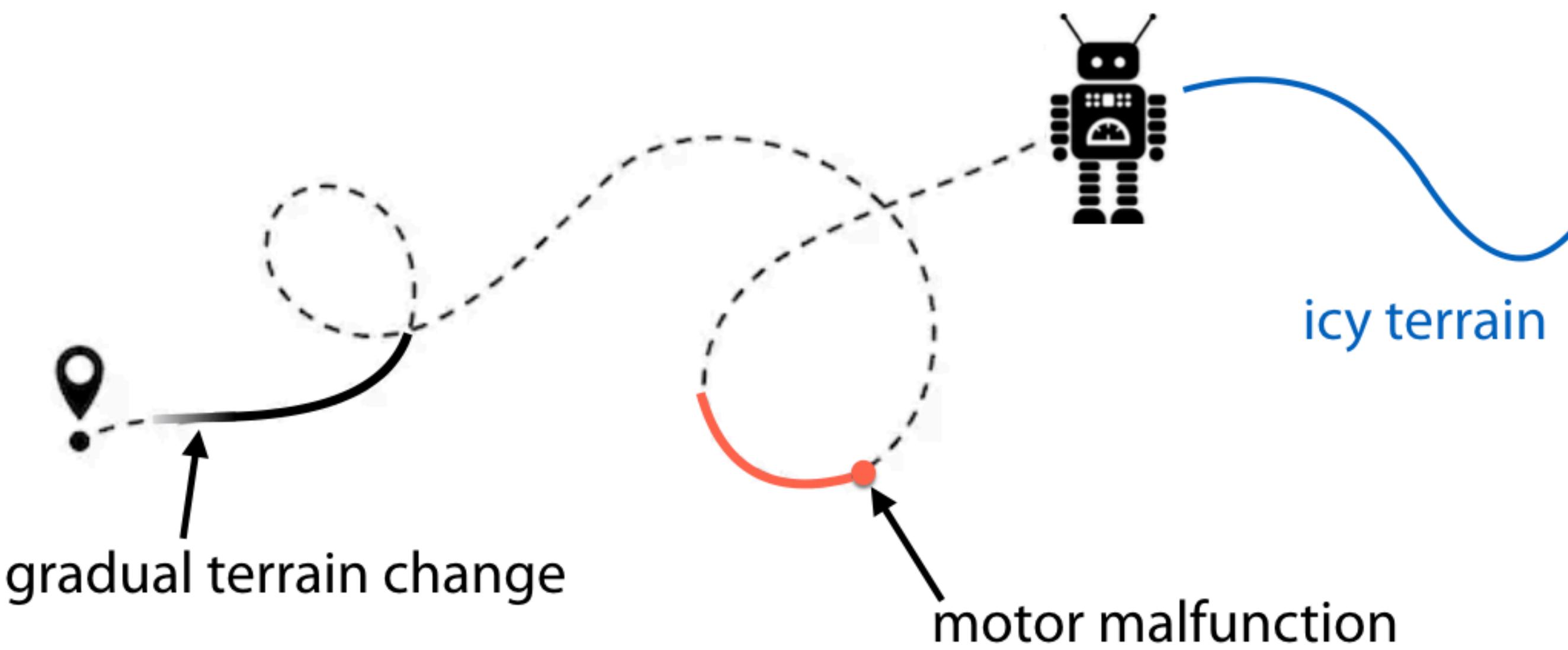
A Unifying Perspective as Posterior Predictive Inference

- **Novel:** Probabilistic, amortized, multi-task, meta-learning framework.
- **Meta-learning:** Learns how to learn a classifier or regressor for each new task.
- **Unifies:** MAML, Meta-LSTM, Prototypical networks, and Conditional Neural Processes are special cases.
- **State of the art:** Leading classification accuracy on 5 of 6 Omniglot & *minilmageNet* tasks.
- **Efficient:** Test-time requires only forward passes, no gradient steps are needed.
- **Versatile:** Robust classification accuracy as shot and way are varied at *test*-time.
- **High quality 1-shot view reconstruction:**

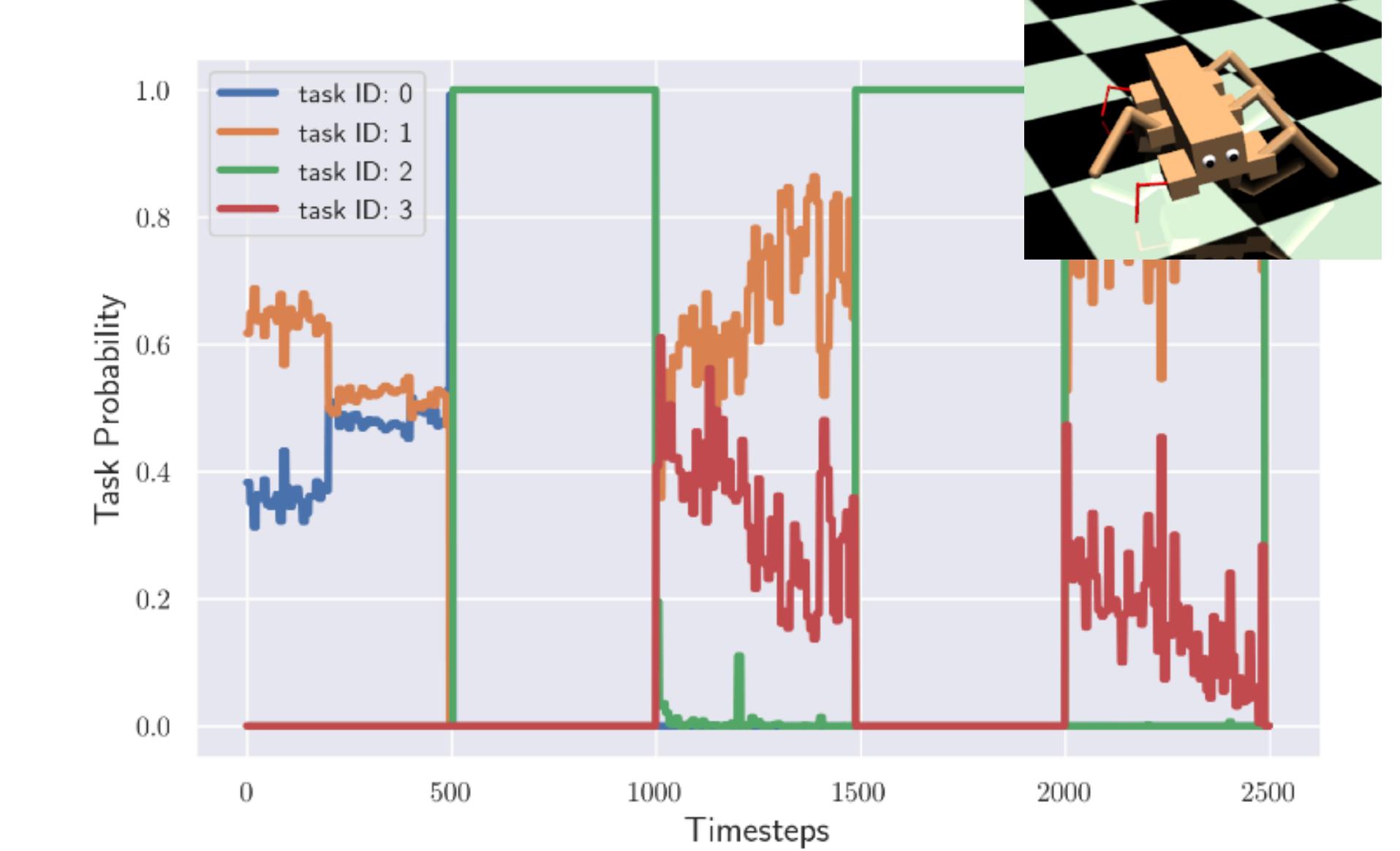


Deep Online Learning via Meta-Learning: Continual Adaptation for Model-Based RL

Anusha Nagabandi, Chelsea Finn, Sergey Levine



Can we use meta-learning for effective online learning?



Our method can:

- Reason about non-stationary latent distributions over tasks.
- Recall past tasks