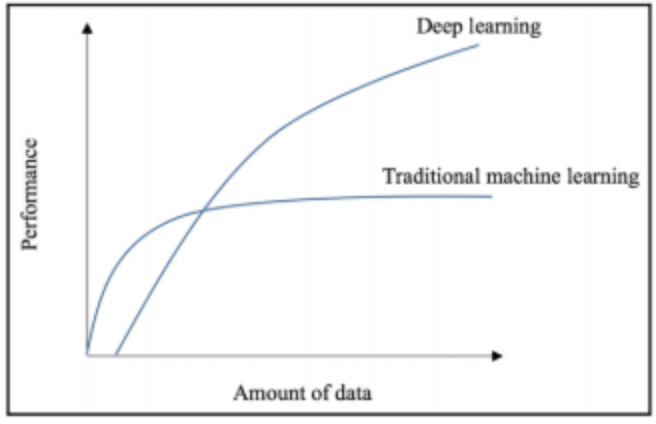
Meta-Learning

Motivation

- Humans do not learn from scratch anytime!
- Fewer samples are needed to master a new skill.
- **Goal:** Can we have a machine do the same?
- Very important for deep learning systems.



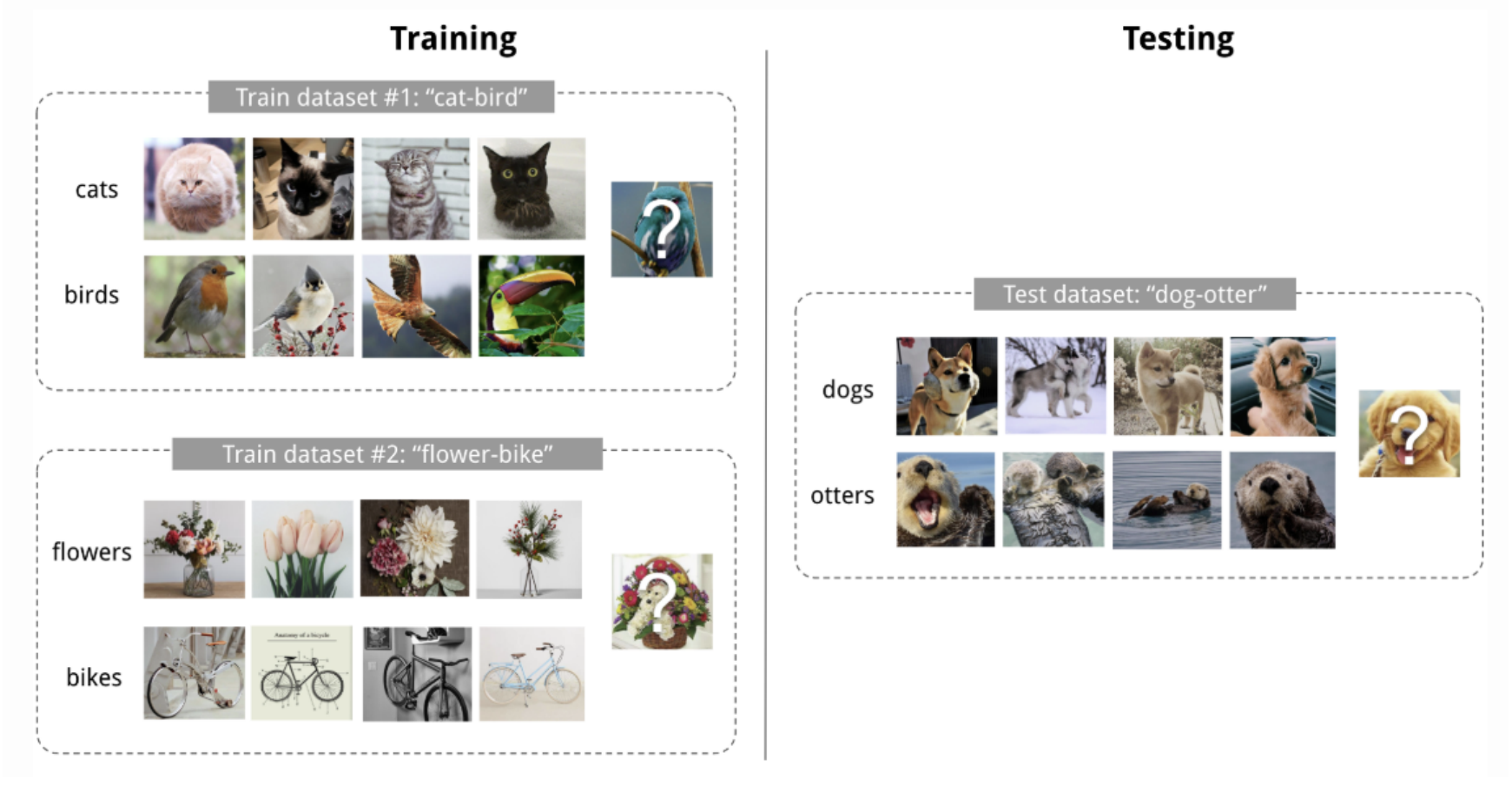
Human vs Machine

- **Sample efficiency:** Humans need few samples, even relatively simple systems like handwritten digit recognition need thousands of samples.
- **Transferability:** Less samples required, but still critical parts or novel components might be hard to tweak.

Meta-Learning: Learning to learn

- A model is trained over a variety of tasks.
- Each task is associated with a dataset that contains input features and a target variable (supervised learning).
- The **model** on a meta-learning problem is a high-level optimizer that updates a low-level model, which is specialized for the task.

Example: 4-shot 2-class image classification

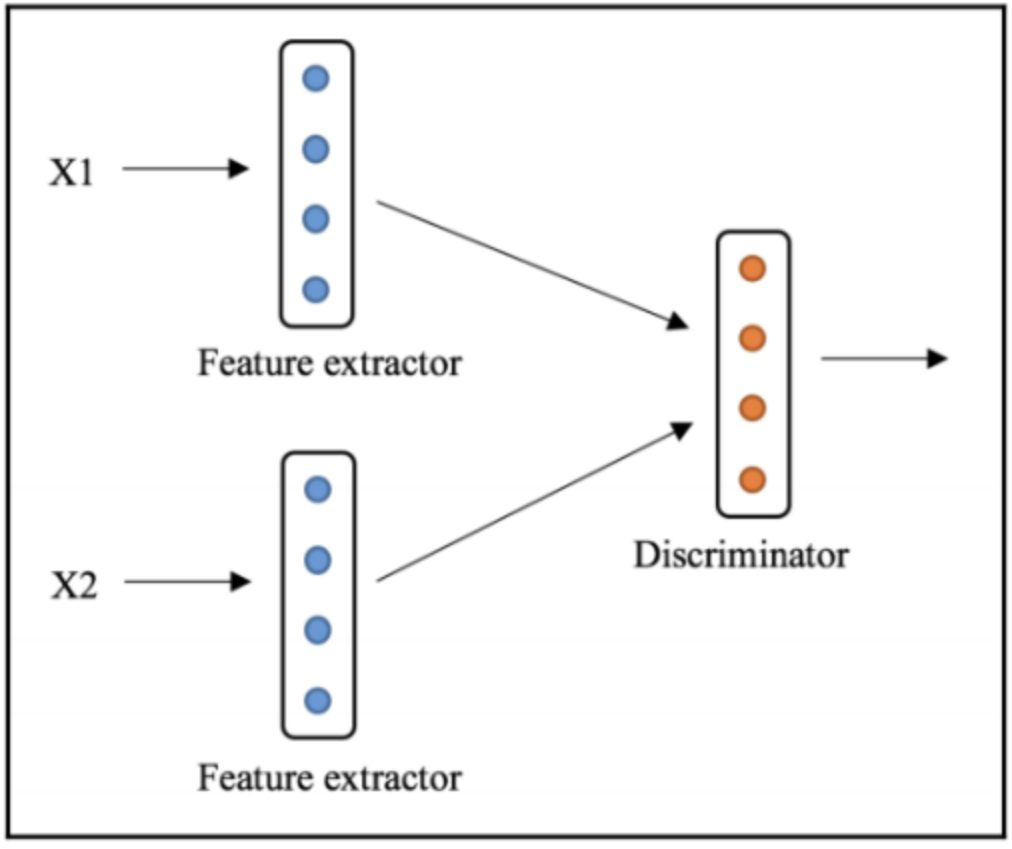


Common approaches

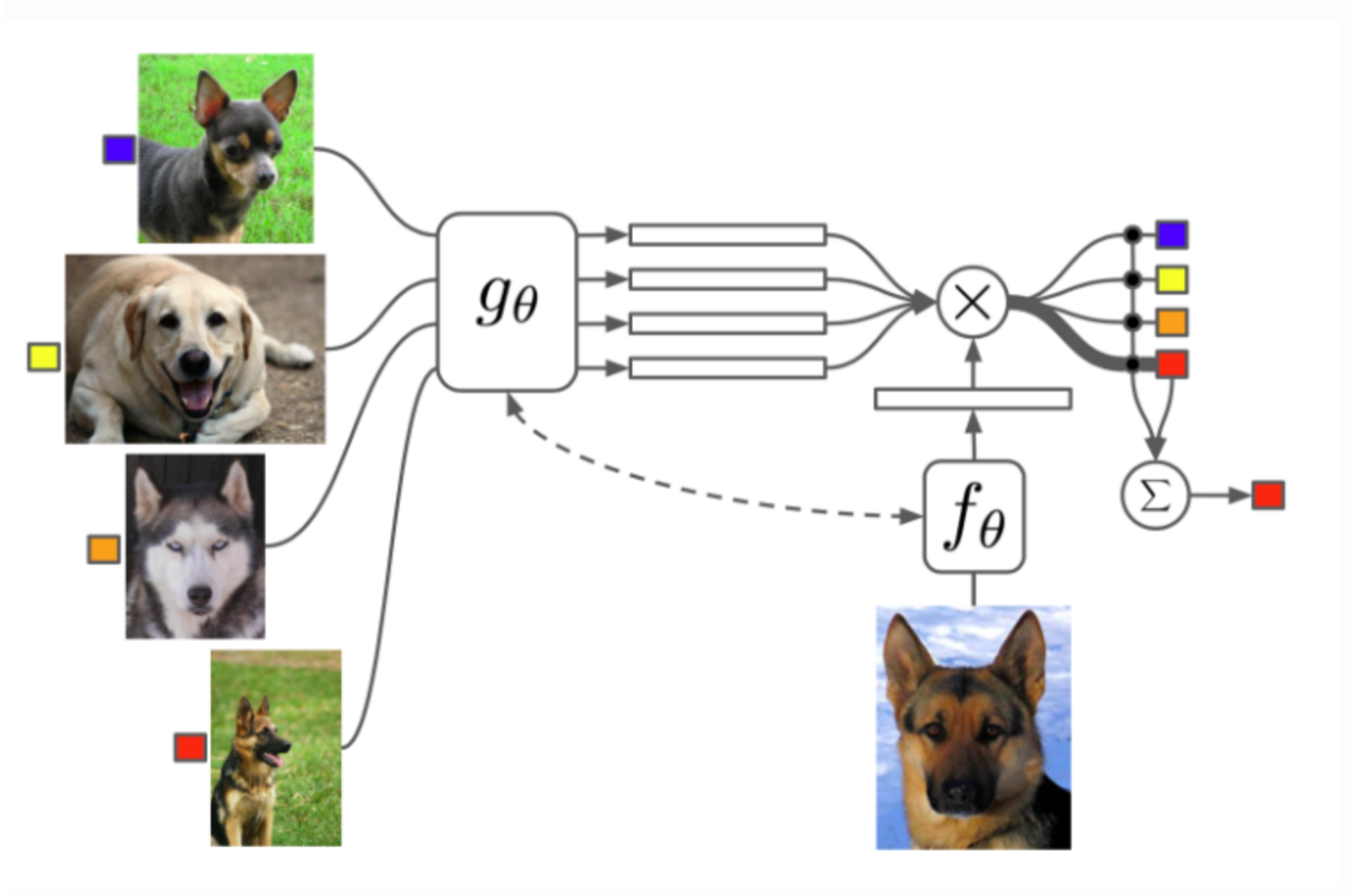
- **Model-based:** Similar to RNN.
- **Metric-based:** Learn embedding and distance function to separate classes.
- **Optimization-based:** Gradient descent.

Metric-Based Meta-Learning: Siamese networks

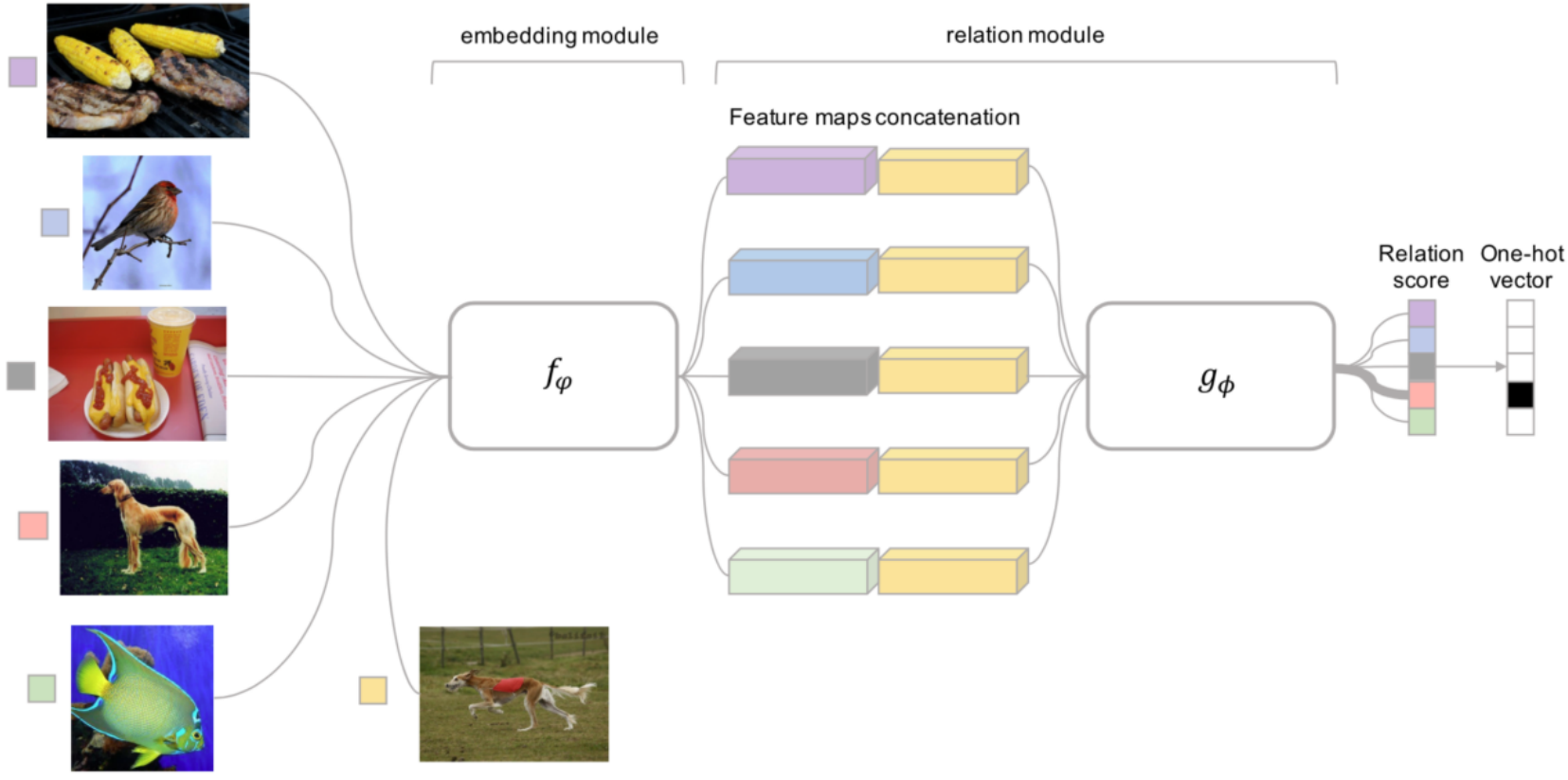
- Twin networks that act as feature extractors.
- The discriminator calculates the distance between embeddings and issues a class probability.



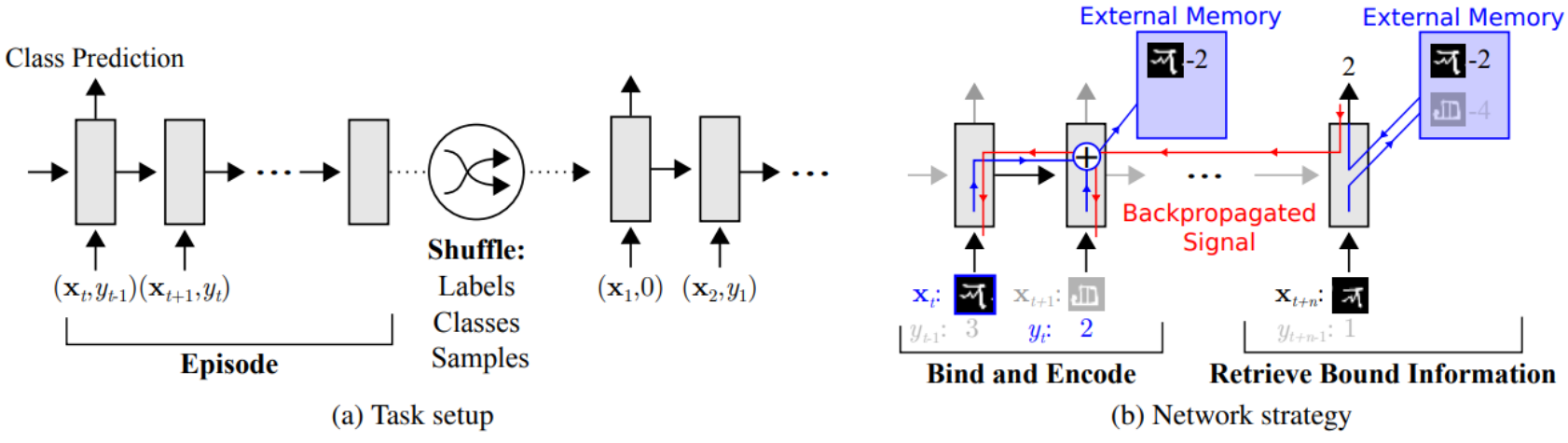
Metric-Based Meta-Learning: Matching network



Metric-Based Meta-Learning: Relation network



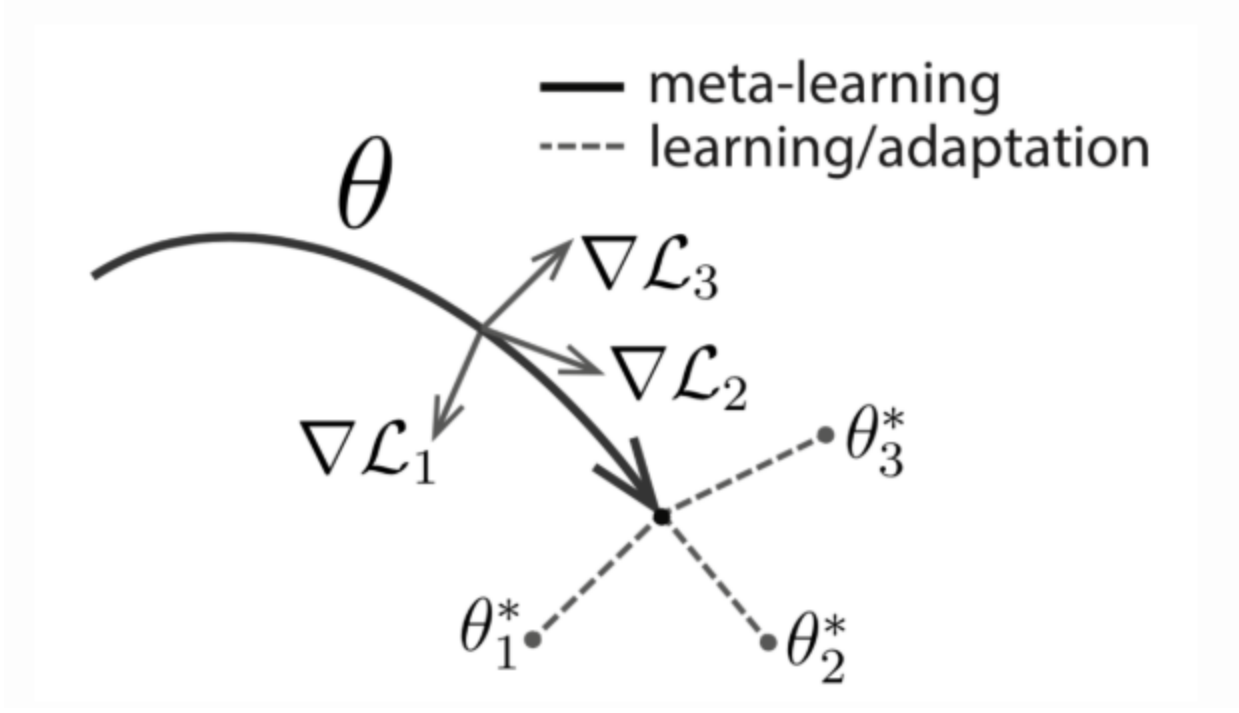
Model-Based Meta-Learning: MANN



- Present data to the network with time-offset labels.
- Focus of the training is on a) learning the embedding, b) retrieving on memory.

Optimization-Based

- **Idea:** We love gradients. Can we make them work in this few-data setting too?



Optimization-Based: MAML (model-agnostic)

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: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$

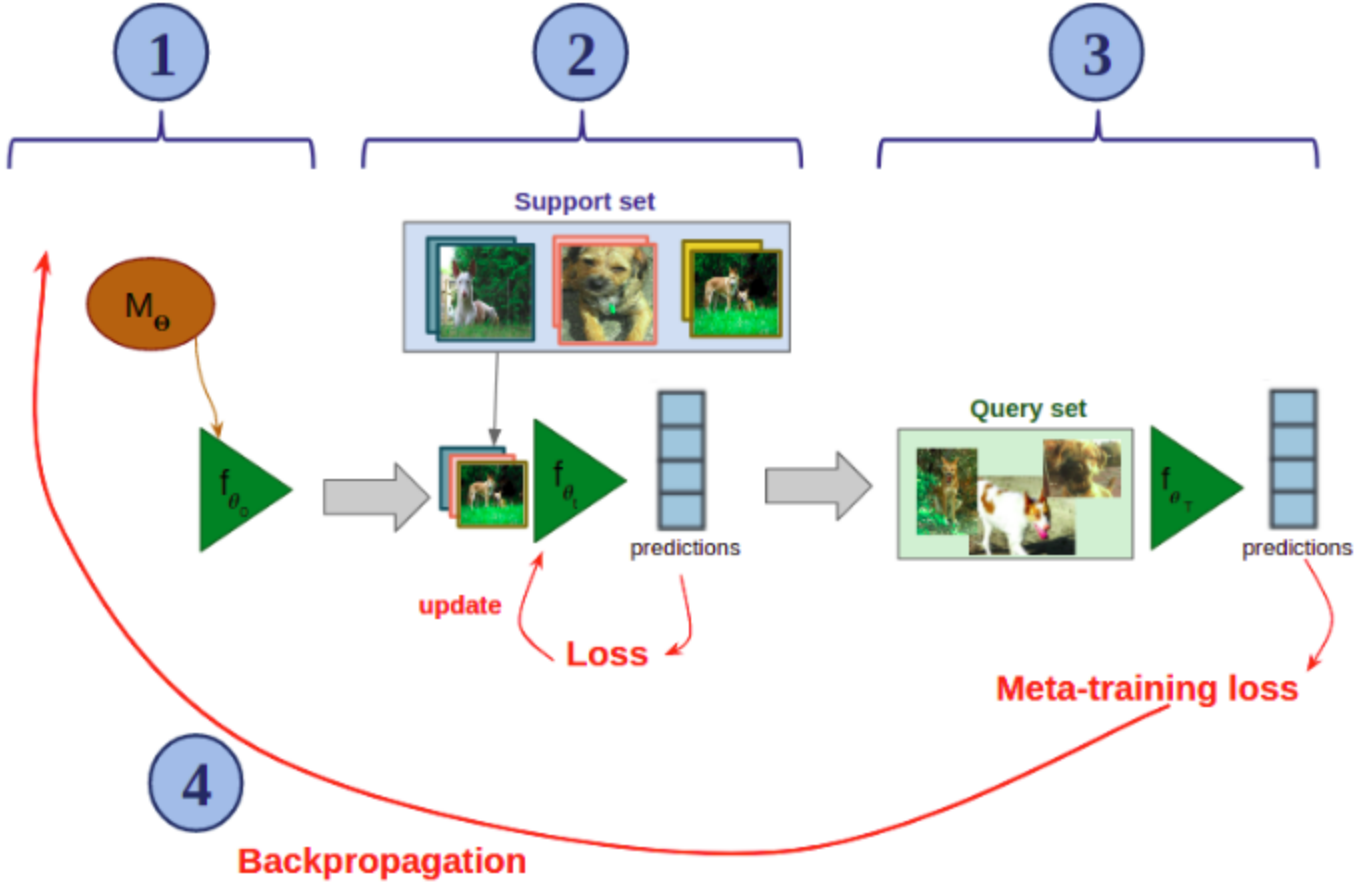
7: **end for** Note: the meta-update is using different set of data.

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**

- Summary article from the authors.

Optimization-Based: MAML (cont.)



- **Support set:** a classification task on which we will train.

Optimization-Based: Reptile

Algorithm 2 Reptile, batched version

Initialize θ

for iteration $= 1, 2, \ldots$ **do**

  Sample tasks $\tau_1, \tau_2, \ldots, \tau_n$

  for $i = 1, 2, \ldots, n$ **do**

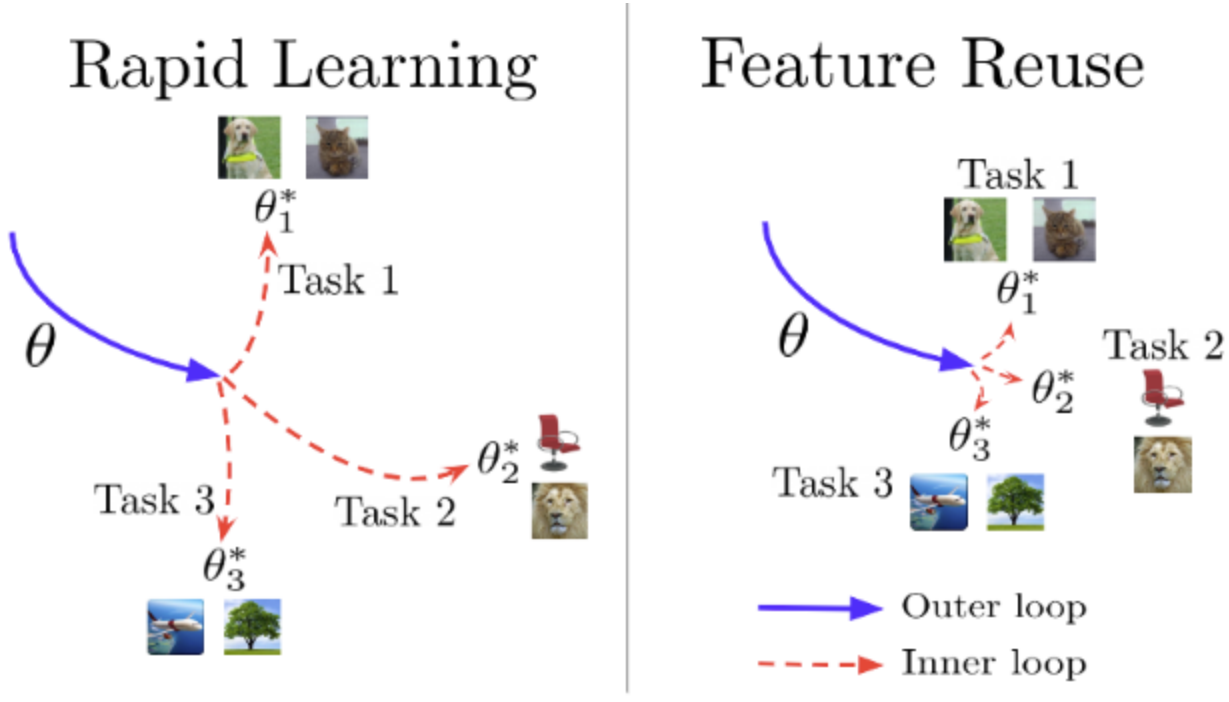
    Compute $W_i = \text{SGD}(L_{\tau_i}, \theta, k)$

  end for

  Update $\theta \leftarrow \theta + \beta \frac{1}{n} \sum_{i=1}^{n} (W_i - \theta)$

end for

Beyond MAML: ANIL (Almost no inner loop)



Visualizations of rapid learning and feature reuse. Diagram from Raghu et al., 2019.

- Same as in MAML, but take the gradient only of the head of the model, keeping the feature extractor part.

This is all very nice, but where is AutoML?

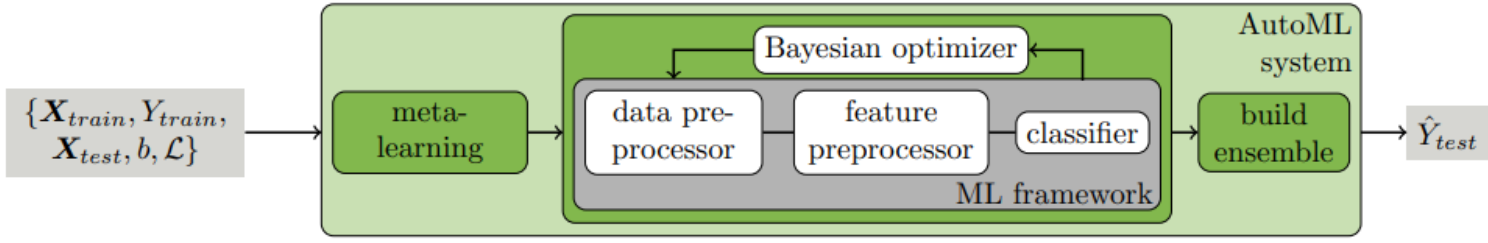


Figure 1: Our improved AutoML approach. We add two components to Bayesian hyperparameter optimization of an ML framework: meta-learning for initializing the Bayesian optimizer and automated ensemble construction from configurations evaluated during optimization.

- Meta-learning helps determine which algorithm to use.
  - Based on 140 datasets from OpenML repository and meta-features (statistics, skewness, entropy of the targets).
- BO then kicks in to solve the HPO part.
- Create an ensemble model in the end.

Auto-sklearn: results

	Abalone	Amazon	Car	Cifar-10	Cifar-10 Small	Convex	Dexter	Dorothea	German Credit	Gisette	KDD09 Appetency	KR-vs-KP	Madelon	MNIST Basic	MRBI	Secom	Semeion	Shuttle	Waveform	Wine Quality	Yeast
AS	**73.50**	**16.00**	0.39	**51.70**	**54.81**	**17.53**	**5.56**	**5.51**	**27.00**	**1.62**	**1.74**	0.42	**12.44**	2.84	**46.92**	**7.87**	**5.24**	**0.01**	14.93	33.76	40.67
AW	**73.50**	30.00	**0.00**	56.95	56.20	21.80	8.33	6.38	28.33	2.29	**1.74**	**0.31**	18.21	2.84	60.34	8.09	5.24	**0.01**	14.13	**33.36**	**37.75**
HS	76.21	16.22	0.39	-	57.95	19.18	-	-	27.67	2.29	-	0.42	14.74	**2.82**	55.79	-	5.87	0.05	**14.07**	34.72	38.45

Table 2: Test set classification error of AUTO-WEKA (AW), vanilla AUTO-SKLEARN (AS) and HYPEROPT-SKLEARN (HS), as in the original evaluation of AUTO-WEKA [2]. We show median percent error across 100 000 bootstrap samples (based on 10 runs), simulating 4 parallel runs. Bold numbers indicate the best result. Underlined results are not statistically significantly different from the best according to a bootstrap test with $p = 0.05$.

Auto-sklearn vs sklearn

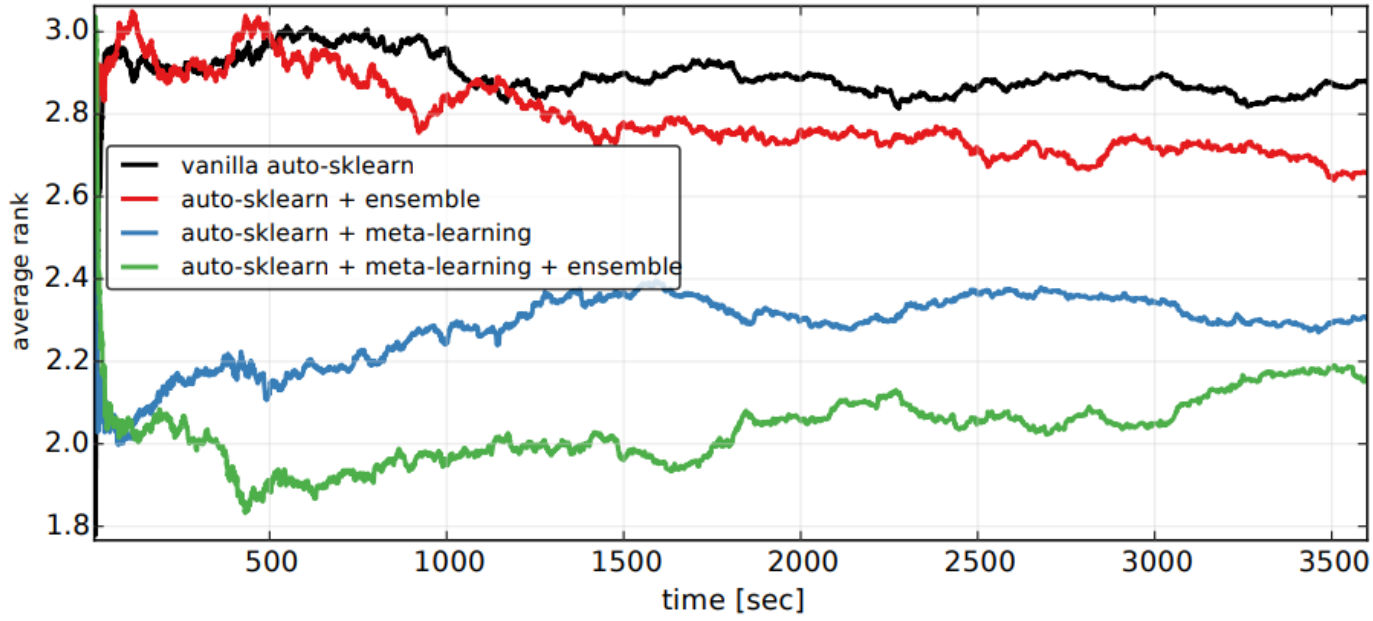


Figure 3: Average rank of all four AUTO-SKLEARN variants (ranked by balanced test error rate (BER)) across 140 datasets. Note that ranks are a relative measure of performance (here, the rank of all methods has to add up to 10), and hence an improvement in BER of one method can worsen the rank of another. The supplementary material shows the same plot on a log-scale to show the time overhead of meta-feature and ensemble computation.

References

- Meta-learning survey (mostly deep learning).
- MANN original article
- Learning to learn (MAML).
- Reptile, with implementation.
- Meta-Learning with Python book, with code
- Auto-sklearn paper