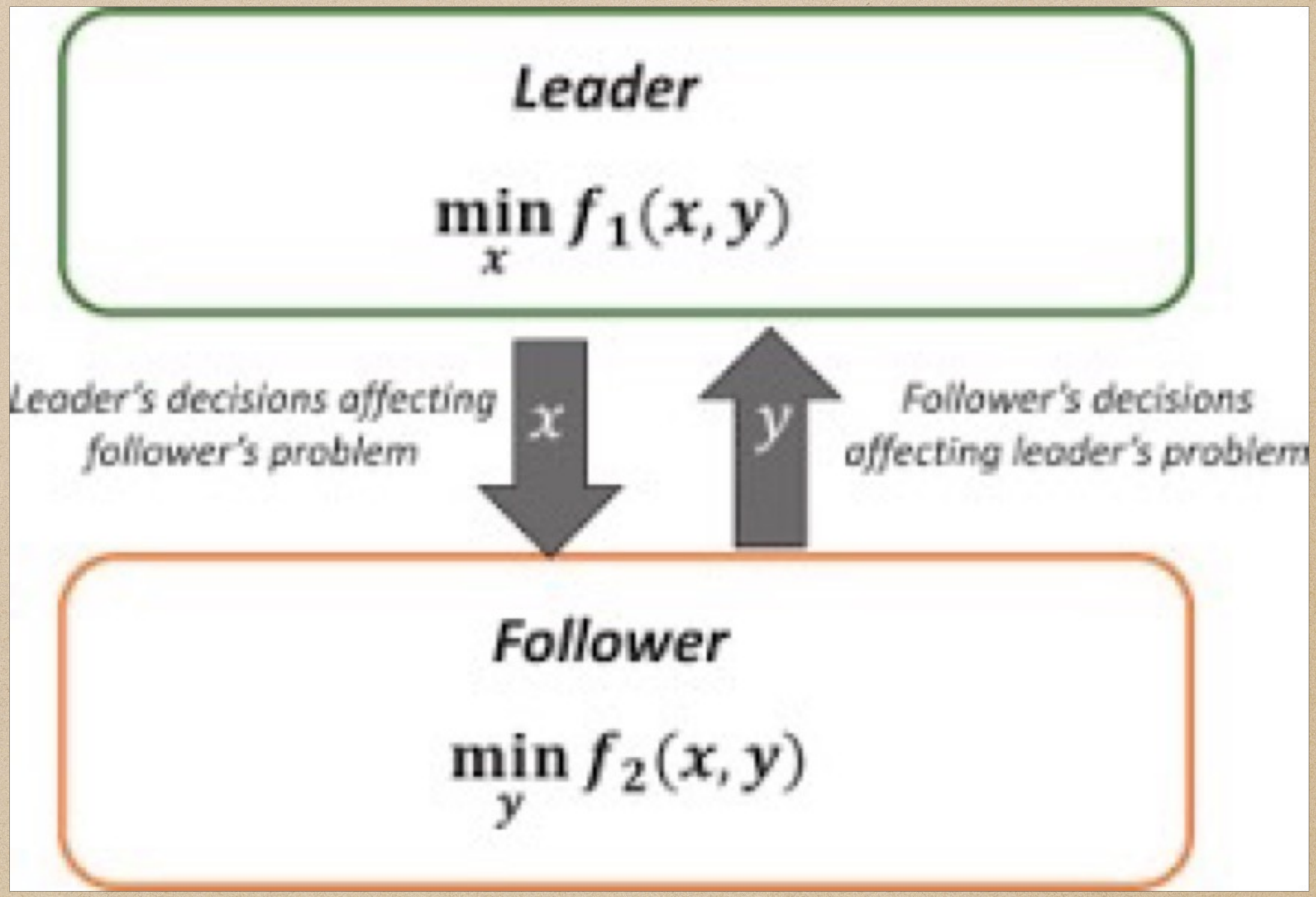
BILEVEL PROGRAMMING FOR HYPERPARAMETER OPTIMIZATION AND META-LEARNING

Luca Franceschi, Paolo Frasconi, Saverio Sales, Riccardo Grazzi, Massimiliano Pontil

KOTHA MEHER PREETHI

What is Bilevel Programming?

- Mathematical programs that have problems with optimization
- Has two levels- upper and lower
- Originated from Stackelberg game and Bracken & McGill



HYPERPARAMETERS

- Values which help in training a model
- Very significant while trying to learn optimal parameters
- The prefix "hyper" imply that these are top-level parameters
- These are not a part of the final model
- Examples - Activation functions, number of hidden layers, batch size, train-test split ratio and so on.

HYPERPARAMETER OPTIMIZATION

- Also called tuning
- The method of finding an optimal combination of hyperparameters
- Techniques - random search, bayesian optimisation , grid search and so on.

META-LEARNING

- Usually refers to ensemble learning algorithms
- A machine learns from the predictions of other learning algorithms
- Meta-Learning executes a level above machine learning

ABSTRACT

- An approach which combines hyperparameter optimization and meta-learning
- Use bilevel programming to achieve this
- The notion of outer and inner variables
- Approximate problem converges to the exact problem

INTRODUCTION

- The goal in Hyperparameter Optimization(HO) and Meta-Learning(ML) is to find a configuration that makes the learning algorithm work well for new data
- Goal of HO and ML - finding a good hypothesis at the inner level and a good configuration at the outer level

- Previous studies on HO were only able to find a few dozen hyper parameters
- Recently developed gradient-based techniques were able to tune hyperparameters , as required
- Technique works well for ML
- HO and ML do not have many differences

- Various techniques evolved in the past few years to tackle ML
- Include metric strategy, memorization strategy, initialization strategy and optimization strategy
- Outer optimization problem is solved based on Inner optimization problem

- HO - outer problem : hyperparameters , inner problem: minimizing the empirical loss
- ML - outer problem : common representation among tasks, inner problem : classifiers for individual tasks

- In this paper, problems related to bilevel optimisation take the form : $\min\{f(\lambda) : \lambda \in \Lambda\}$, where function $f : \Lambda \to R$ is defined at $\lambda \in \Lambda$ as $f(\lambda) = \inf\{E(w,\lambda) : w \in \arg\min L(u)\}$, where $E : R \times \Lambda \to R$ is the outer objective and $L : R \to R$ is the inner objective.

HO- Inner and Outer Objective

- In HO, the focus is on reducing the validation error of a model by finding suitable hyper parameters.
- The validation error is measured on a validation dataset which is different from the training dataset.
- Machine Learning models are highly sensitive to hyper parameters.
- Here, we regularise the empirical error for the inner objective and the outer objective acts as a proxy for generalisation error.

Hyperparameter Optimization

Inner Objective

$$L_\lambda(w) = \sum_{(x,y)\in D_{\mathrm{tr}}} \ell(g_w(x), y) + \Omega_\lambda(w),$$

Outer Objective

$$E(w, \lambda) = \sum_{(x,y)\in D_{\mathrm{val}}} \ell(g_w(x), y).$$

ML - Inner and Outer Objectives

- In ML, the inner and outer objectives are calculated by computing the average of training and validation errors across various tasks.
- We consider a meta-training dataset which is a collection of datasets, each of which is related to a specific task.
- The hyperparameters here are shared between tasks.

Meta-Learning

Inner Objective

$$L_\lambda(w) = \sum_{j=1}^{N} L^j(w^j, \lambda, D_{\text{tr}}^j),$$

$$E(w, \lambda) = \sum_{j=1}^{N} L^j(w^j, \lambda, D_{\text{val}}^j)$$

Outer Objective

GRADIENT-BASED APPROACH

- This approach is taken into consideration when the hyper parameter vector has real numbers.
- Here we consider the inner objective to have a unique minimizer w and introduce $T = \{1,2,\ldots T\}$ which is a positive integer and the approximation becomes $\min f_T(\lambda) = E(w_{T,\lambda}, \lambda)$ where E is a smooth scalar function.

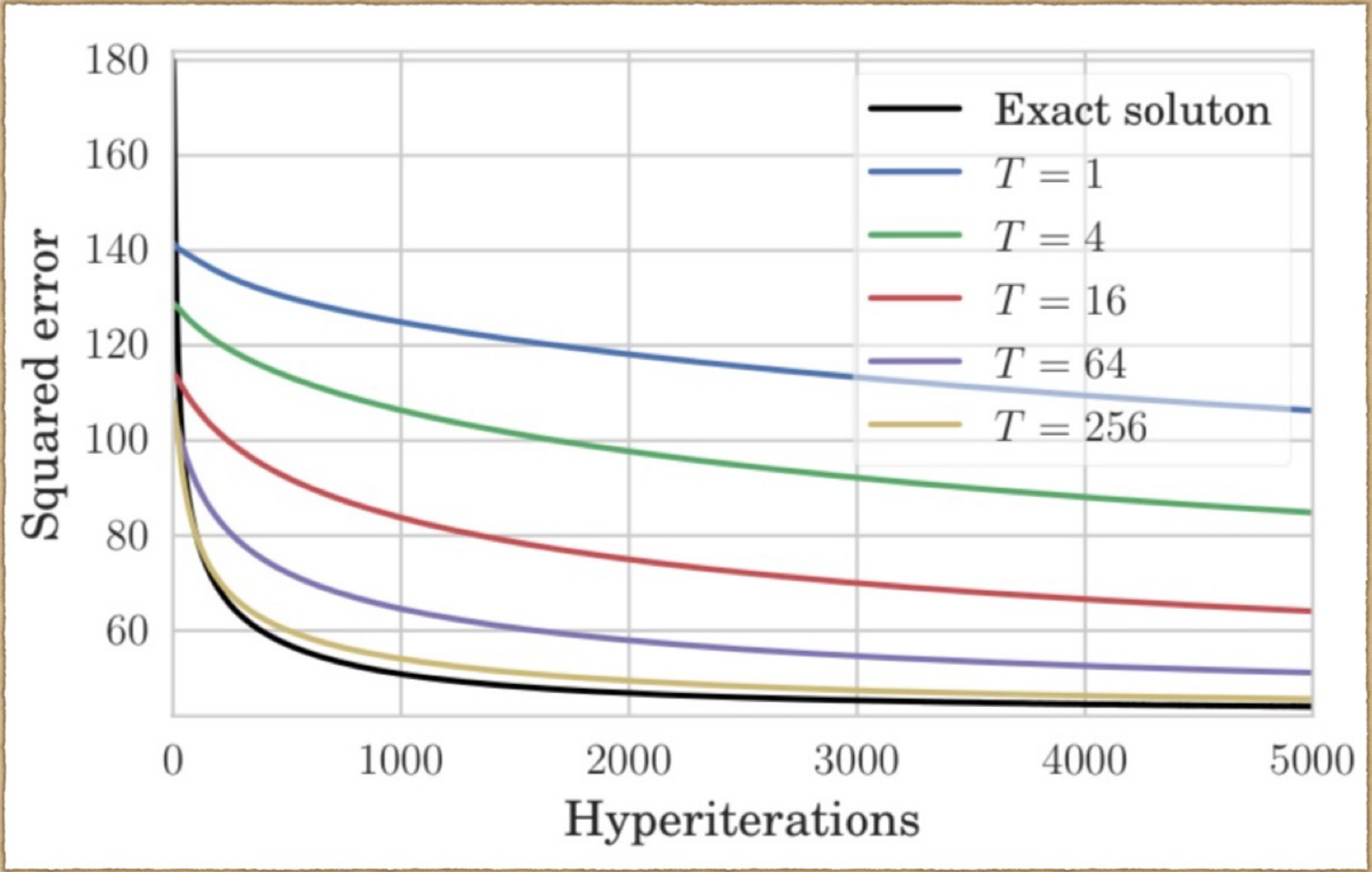
HYPER-REPRESENTATIONS

- Good data representations are very crucial.
- In a bilevel approach, we divide each dataset into training and validation sets.
- Weights of the specific tasks are learned from T iterations of the gradient descent.
- We focus on hypothesis space and aim to maximise the generalisation to new data while training with respect to the hyperparameters.

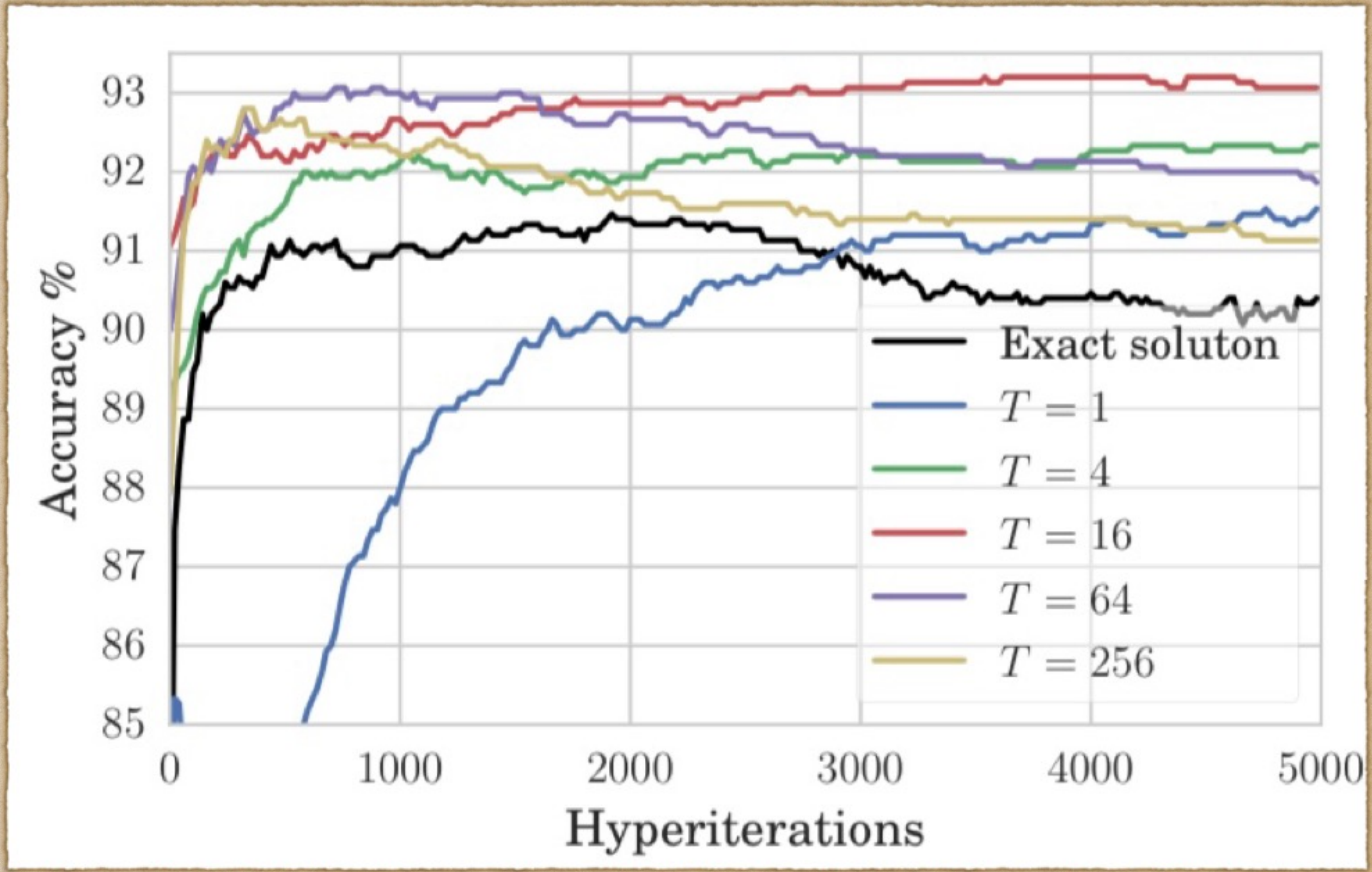
EFFECT OF ITERATIONS(T)

- Investigate how performing a small number of iterations affects the performance and running time.
- Omniglot dataset is used, from which 100 classes are extracted and a HO problem is constructed in order to tune a hyperparameter H.
- A training set and validation set- consisting of 3 randomly drawn examples per class.
- A testing dataset containing 15 examples per class.

Effect of T on Training Dataset and Validation Dataset



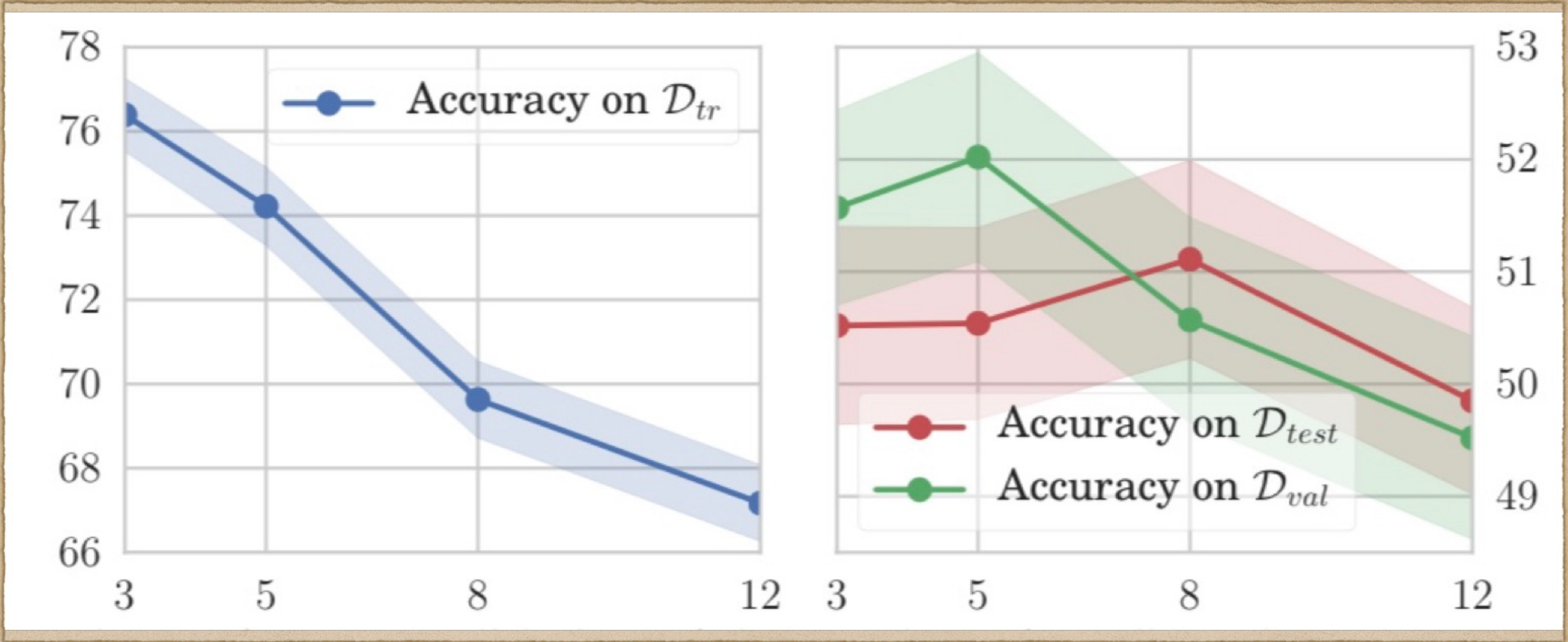
Effect of T on the testing dataset



FEW-SHOT LEARNING

- Used in Meta-Learning
- Uses two benchmark datasets
- OMNIGLOT : 1623 different handwritten characters using 50 alphabets.
- MINIIMAGENET : Subset of ImageNet, contains 60,000 downsampled images from 100 different classes

- Classify the datasets to meta-training set , meta-validation set and a meta-testing set
- meta-training set - sampling datasets
- meta-validation set - tuning Meta-Learning Hyperparameters
- meta-testing set - estimate accuracy
- Every meta-dataset contain samples that belong to different classes



EFFECT OF T IN META-LEARNING

Confidence intervals of other methods

- Advantages of believe framework over other approaches
- One-shot, five-shot on Omniglot and MiniImagenet
- Confidence intervals of both the datasets are calculated

Method	OMNIGLOT 5 classes 1-shot	OMNIGLOT 5 classes 5-shot	OMNIGLOT 20 classes 1-shot	OMNIGLOT 20 classes 5-shot	MINIIMAGENET 5 classes 1-shot	MINIIMAGENET 5 classes 5-shot
Siamese nets (Koch et al., 2015)	97.3	98.4	88.2	97.0	–	–
Matching nets (Vinyals et al., 2016)	98.1	98.9	93.8	98.5	43.44 ± 0.77	55.31 ± 0.73
Neural stat. (Edwards and Storkey, 2016)	98.1	99.5	93.2	98.1	–	–
Memory mod. (Kaiser et al., 2017)	98.4	99.6	95.0	98.6	–	–
Meta-LSTM (Ravi and Larochelle, 2017)	–	–	–	–	43.56 ± 0.84	60.60 ± 0.71
MAML (Finn et al., 2017)	98.7	99.9	95.8	98.9	48.70 ± 1.75	63.11 ± 0.92
Meta-networks (Munkhdalai and Yu, 2017)	98.9	–	97.0	–	49.21 ± 0.96	–
Prototypical Net. (Snell et al., 2017)	98.8	99.7	96.0	98.9	49.42 ± 0.78	68.20 ± 0.66
SNAIL (Mishra et al., 2018)	99.1	99.8	97.6	99.4	55.71 ± 0.99	68.88 ± 0.92
Hyper-representation	98.6	99.5	95.5	98.4	50.54 ± 0.85	64.53 ± 0.68

Performance of Various Methods

CONCLUSIONS

- HO AND ML, both can be evaluated using bilevel programming.
- An iterative strategy is followed.
- Inner problem containing a unique solution, display convergence guarantee.

- Classical strategies while working with ML have proved to be efficient.
- Hyper- representations play a vital role.
- Different inner learning algorithms' designs might be worth exploring in the future.

References

- https://arxiv.org/pdf/1806.04910.pdf
- https://machinelearningmastery.com/category/ensemble-learning/
- https://serokell.io/blog/nn-and-one-shot-learning
- http://link.springer.com/10.1007/BF00935665
- https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21

THANK YOU