Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML

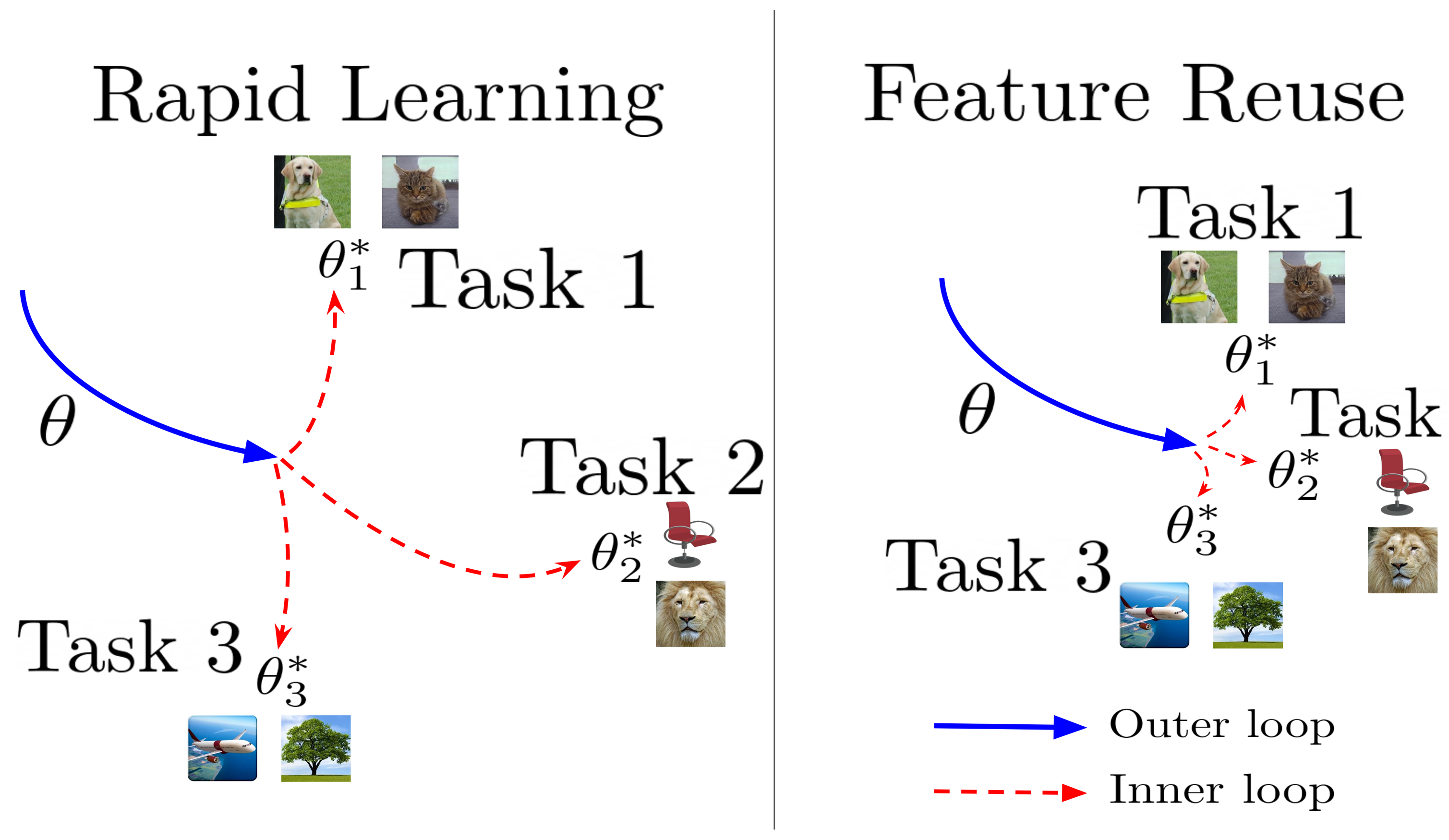
Aniruddh Raghu*, Maithra Raghu*, Samy Bengio, Oriol Vinyals

Introduction

- Model Agnostic Meta-Learning (MAML) is a highly popular and successful algorithm for few-shot learning.
- MAML algorithm has two optimisation loops:
  - *Outer loop:* Find an effective meta-initialisation
  - *Inner loop:* Using this initialisation, adapt parameters via gradient descent to solve each target task
- Despite its popularity, it is unclear whether MAML works due to :
  - **Rapid Learning:** efficient but significant representational adaptation
  - **Feature Reuse:** meta-initialisation already has high-quality representations, so can just reuse these
- We analyse MAML and find that **feature reuse** is the dominant mode of operation.
- Motivated by our analysis, we propose two simplified algorithms, with same performance

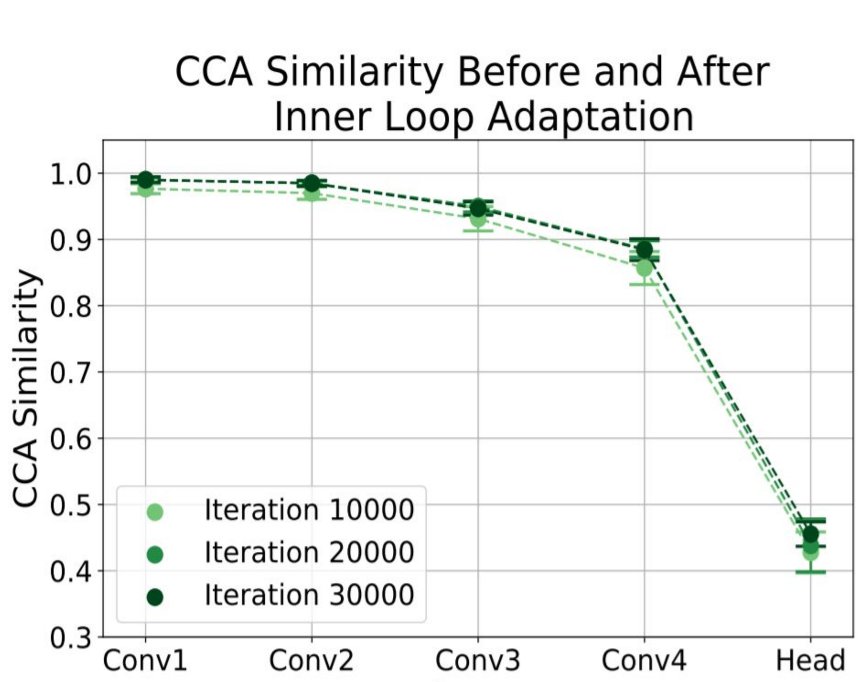
Rapid Learning and Feature Reuse

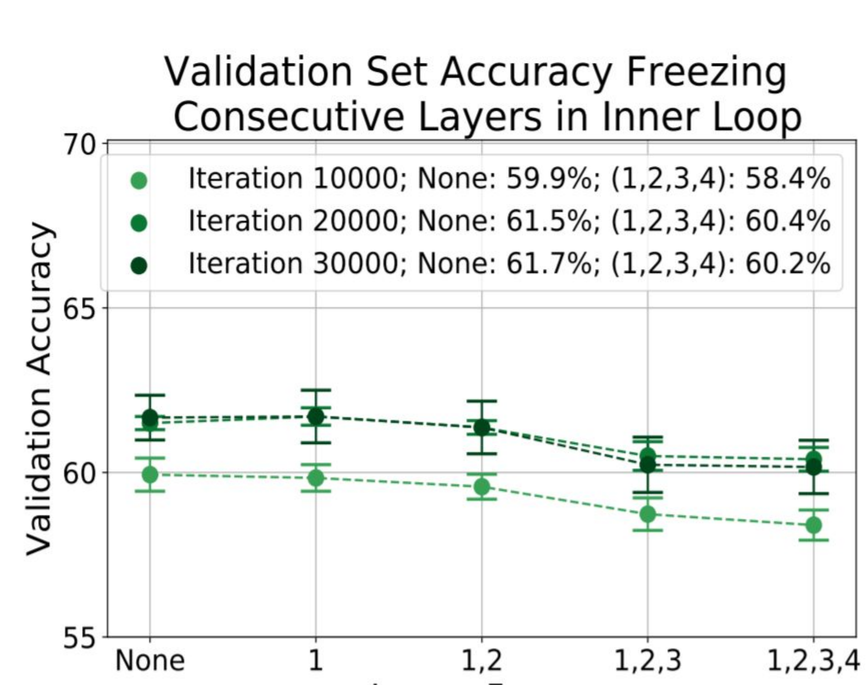
- Rapid learning involves large parameter changes on inner loop, whereas feature reuse involves little specialisation



Feature Reuse Dominates

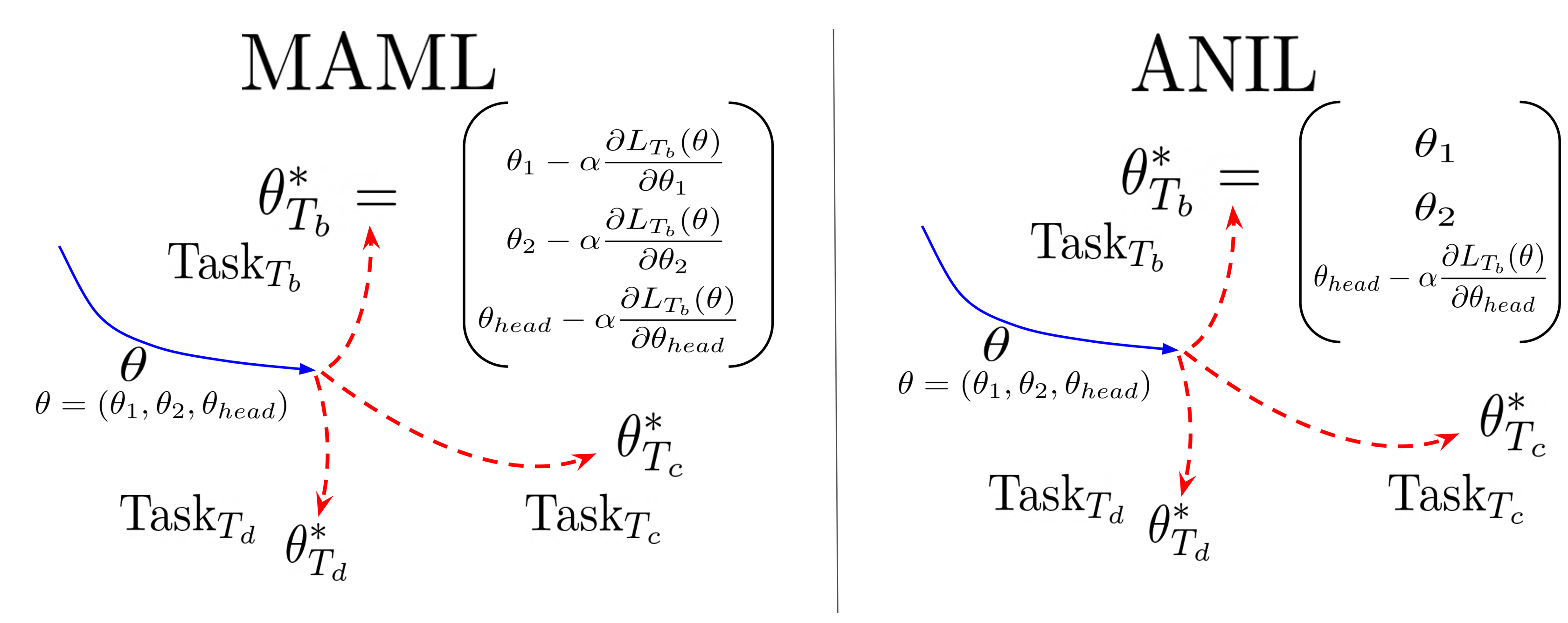
- We perform two sets of analyses:
  - **Layer Freezing:** Do not update contiguous subset of layers of the network, during the inner loop at inference time. Examine FSL performance to no freezing.
  - **Representational Similarity:** Apply Canonical Correlation Analysis (CCA) to the latent representations of the network; compare pre and post inner loop updates.
- **Results:** we see:
  (1) Freezing layers does not affect performance
  (2) Layers are highly similar pre/post inner loop updates.
  (3) Above is true from early on in training.
- **Significant features reuse is occurring!**





ANIL and NIL Algorithms

- Inner loop has little effect at inference time. But what about at training time?
- Introduce **ANIL (Almost No Inner Loop)** algorithm -- no inner loop at training time either, for network body. Keep for head to allow alignment. Pictorially:



- Further consider **NIL (No Inner Loop)** algorithm -- train with ANIL, remove network head at test time, and classify based on cosine distance nearest neighbours from support set.
- **Accuracy:** ANIL and NIL perform identically to MAML!
  - **We (mostly) don't need inner loop at training time**
  - **We can remove inner loop entirely at test time**

Method	MiniImageNet 5 way 1 shot	MiniImageNet 5 way 5 shot
MAML	46.9 ± 0.2	**63.1 ± 0.4**
ANIL	46.7 ± 0.4	61.5 ± 0.5
NIL	**48.0 ± 0.7**	62.2 ± 0.5

- **Computational benefit:** ANIL obtains:
  - **Training: ~1.7x speedup** over MAML
  - **Inference: ~4x speedup** over MAML
  - Without sacrificing performance

MAML Head Learns Better Features

- NIL removes the network head at test time: no performance drop
- **What is role of head at training time?**
  - Compare performance using ANIL (ie, MAML head) to different methods of feature learning, and assess performance with nearest nbors:

Method	MiniImageNet 5 way 1 shot	MiniImageNet 5 way 5 shot
ANIL	**46.7 ± 0.4**	**61.5 ± 0.5**
Multiclass	39.7 ± 0.3	54.4 ± 0.5
Multitask	26.5 ± 1.1	34.2 ± 3.5

- Head permits better feature learning over other baselines. Addressing alignment issue → significant improvement over multitask training.

Conclusions

- Feature reuse dominates in MAML → leads to simpler methods
- Interesting to explore more diverse tasks, datasets

Acknowledgements
The authors would like to thank Geoffrey Hinton, Chelsea Finn, Hugo Larochelle and Chiyuan Zhang for helpful comments and suggestions.