CNAPs

Fast and Flexible Multi-Task Classification Using Conditional Neural Adaptive Processes[1]

James Requeima*
University of Cambridge,
Invenia Labs

Jonathan Gordon*
University of Cambridge

John Bronskill*
University of Cambridge

Sebastian Nowozin
Google Research Berlin

Richard E. Turner
University of Cambridge,
Microsoft Research

Department of Engineering

[1] *Requeima, J., *Gordon, J., and *Bronskill, J., Nowozin, S. and Turner, R.E. " Fast and Flexible Multi-Task Classification Using Conditional Neural Adaptive Processes." arXiv preprint arXiv:1906.07697 (2019). **Authors contributed equally*.

Multi-task, Few-shot Learning

Multi-task, Few-shot Learning

Motivation

- Humans can learn a new concept from just a few examples.[1]

[1] Lake, Brenden, et al. "One shot learning of simple visual concepts.“ *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 33. No. 33. 2011.

Multi-task, Few-shot Learning

Motivation

- Humans can learn a new concept from just a few examples.[1]

Goal:

- Quickly *adapt* and learn to make predictions from a small number of training examples or *shots* at *test time* on *diverse* tasks.

[1] Lake, Brenden, et al. "One shot learning of simple visual concepts.“ *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 33. No. 33. 2011.

Multi-task, Few-shot Learning

Motivation

- Humans can learn a new concept from just a few examples.[1]

Goal:

- Quickly *adapt* and learn to make predictions from a small number of training examples or *shots* at *test time* on *diverse* tasks.

Strategy:

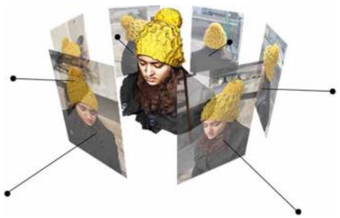
- Learn across *many tasks* in order to generalize to a new unseen task.

[1] Lake, Brenden, et al. "One shot learning of simple visual concepts.“ *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 33. No. 33. 2011.

Scenarios:
Multiple tasks, few training examples

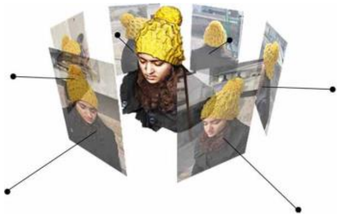
Scenarios:
Multiple tasks, few training examples

- 3D reconstruction of objects from a small number of views.

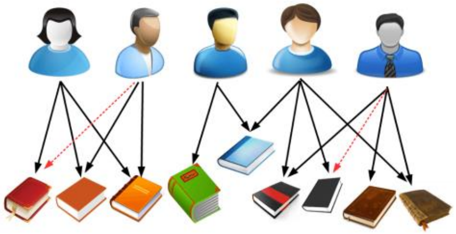


Scenarios:
Multiple tasks, few training examples

- 3D reconstruction of objects from a small number of views.

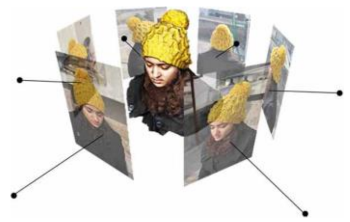


- Recommendation/personalization:
  - many users, few ratings/likes per user.

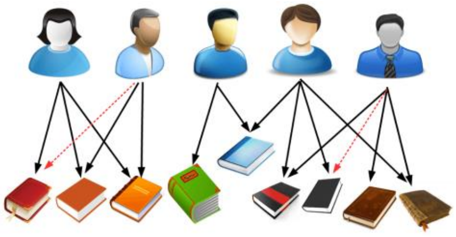


Scenarios:
Multiple tasks, few training examples

- 3D reconstruction of objects from a small number of views.



- Recommendation/personalization:
  - many users, few ratings/likes per user.



- Face recognition/tagging within groups:
  - Many groups of different sizes, with varying number of photos per person.

Training Images (*Context Set*)

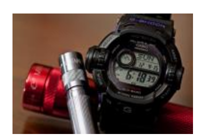
Stop Watch


Digital Clock

Digital Watch


Wrist Computer Tells Time Without Hands


Parking Meter


Test Images (*Target Set*)

Training Images (*Context Set*)

Predictions

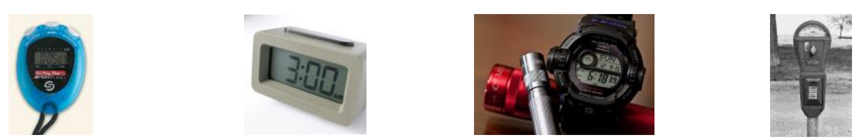
Stop Watch

Digital Clock

Digital Watch

Parking Meter

0.69	0.02	0.07	0.24
0.04	0.13	0.11	0.32
0.27	0.85	0.80	0.35
0.00	0.01	0.02	0.10



Test Images (*Target Set*)

Training Images (*Context Set*)

Stop Watch

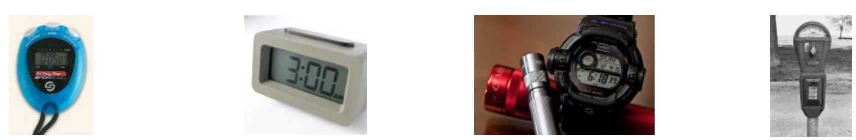
Digital Clock

Digital Watch

Parking Meter

Predictions

0.69	0.02	0.07	0.24
0.04	0.13	0.11	0.32
0.27	0.85	0.80	0.35
0.00	0.01	0.02	0.10



Test Images (*Target Set*)

Training Images (*Context Set*)

Stop Watch

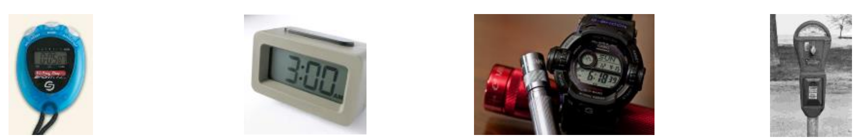
Digital Clock

Digital Watch

Parking Meter

Predictions

0.69	0.02	0.07	0.24
0.04	0.13	0.11	0.32
0.27	0.85	0.80	0.35
0.00	0.01	0.02	0.10



Test Images (*Target Set*)

2/4 correct

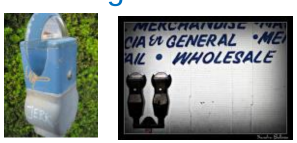
Training Images (*Context Set*)

Stop Watch

Digital Clock

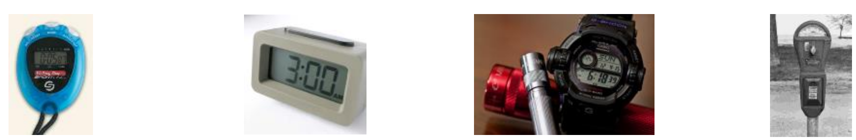
Digital Watch

Parking Meter



Predictions

0.63	0.02	0.04	0.11
0.04	0.09	0.13	0.11
0.29	0.70	0.61	0.24
0.04	0.19	0.22	0.54



Test Images (*Target Set*)

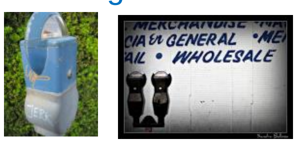
Training Images (*Context Set*)

Stop Watch

Digital Clock

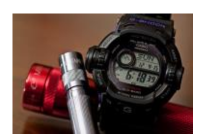
Digital Watch

Parking Meter



Predictions

$\boxed{0.63}$	0.02	0.04	0.11
0.04	0.09	0.13	0.11
0.29	$\boxed{0.70}$	$\boxed{0.61}$	0.24
0.04	0.19	0.22	$\boxed{0.54}$


Test Images (*Target Set*)

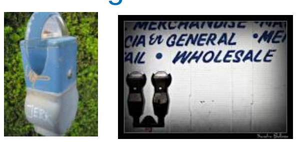
Training Images (*Context Set*)

Stop Watch

Digital Clock

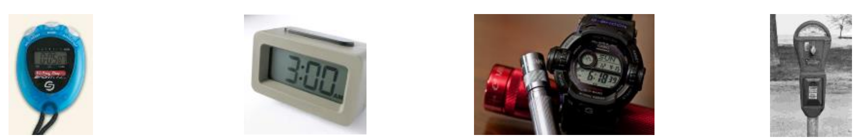
Digital Watch

Parking Meter



Predictions

$\boxed{0.63}$	0.02	0.04	0.11
0.04	0.09	0.13	0.11
0.29	$\boxed{0.70}$	$\boxed{0.61}$	0.24
0.04	0.19	0.22	$\boxed{0.54}$



Test Images (*Target Set*)

3/4 correct

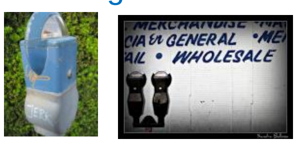
Training Images (*Context Set*)

Stop Watch

Digital Clock

Digital Watch

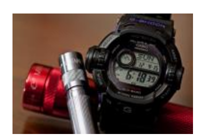
Parking Meter



Sundial

Predictions

0.81	0.04	0.07	0.11	0.11
0.01	0.12	0.09	0.12	0.16
0.08	0.68	0.72	0.12	0.13
0.02	0.16	0.12	0.54	0.12
0.07	0.00	0.00	0.12	0.47



Test Images (*Target Set*)

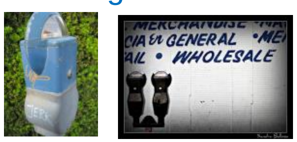
Training Images (*Context Set*)

Stop Watch

Digital Clock

Digital Watch

Parking Meter



Sundial

Predictions

$\boxed{0.81}$	0.04	0.07	0.11	0.11
0.01	0.12	0.09	0.12	0.16
0.08	$\boxed{0.68}$	$\boxed{0.72}$	0.12	0.13
0.02	0.16	0.12	$\boxed{0.54}$	0.12
0.07	0.00	0.00	0.12	$\boxed{0.47}$

Test Images (*Target Set*)

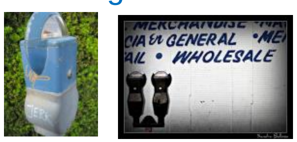
Training Images (*Context Set*)

Stop Watch

Digital Clock

Digital Watch

Parking Meter



Sundial

Predictions

0.81	0.04	0.07	0.11	0.11
0.01	0.12	0.09	0.12	0.16
0.08	0.68	0.72	0.12	0.13
0.02	0.16	0.12	0.54	0.12
0.07	0.00	0.00	0.12	0.47

Test Images (*Target Set*)

4/5 correct

Probabilistic Models: old and new

Probabilistic Models: old and new

- **Traditional multi-task modelling approach:**

$$p(y^\tau | x^\tau, \theta, \psi^\tau)$$

τ = task

θ = global parameters

ψ^τ = task specific params

Probabilistic Models: old and new

- **Traditional multi-task modelling approach:**

$$p(y^\tau | x^\tau, \theta, \psi^\tau)$$

τ = task
θ = global parameters
ψ^τ = task specific params

Train task specific parameters on task data $\mathcal{D}^\tau = \{x_n^\tau, y_n^\tau\}_{n=1}^N$ giving $\hat{\psi}^\tau$

Probabilistic Models: old and new

- **Traditional multi-task modelling approach:**

τ = task
θ = global parameters
ψ^τ = task specific params

$$p(y^\tau | x^\tau, \theta, \psi^\tau)$$

Train task specific parameters on task data $\mathcal{D}^\tau = \{x_n^\tau, y_n^\tau\}_{n=1}^N$ giving $\hat{\psi}^\tau$

Predict using: $p(y^* | x^*, \theta, \hat{\psi}^\tau)$

Probabilistic Models: old and new

- **Traditional multi-task modelling approach:**

τ = task
θ = global parameters
ψ^τ = task specific params

$$p(y^\tau | x^\tau, \theta, \psi^\tau)$$

Train task specific parameters on task data $\mathcal{D}^\tau = \{x_n^\tau, y_n^\tau\}_{n=1}^N$ giving $\hat{\psi}^\tau$

Predict using: $p(y^* | x^*, \theta, \hat{\psi}^\tau)$

- **New approach: directly specify the predictive in terms of the task data**

$$p(y^* | x^*, \theta, \psi_\phi(\mathcal{D}^\tau))$$

Probabilistic Models: old and new

- **Traditional multi-task modelling approach:**

τ = task
θ = global parameters
ψ^{τ} = task specific params

$$p(y^{\tau}|x^{\tau}, \theta, \psi^{\tau})$$

Train task specific parameters on task data $\mathcal{D}^{\tau} = \{x_n^{\tau}, y_n^{\tau}\}_{n=1}^{N}$ giving $\hat{\psi}^{\tau}$

Predict using: $p(y^{*}|x^{*}, \theta, \hat{\psi}^{\tau})$

- **New approach: directly specify the predictive in terms of the task data**

$$p(y^{*}|x^{*}, \theta, \psi_{\phi}(\mathcal{D}^{\tau}))$$

Relates to Conditional Neural Processes [1] and auto-regressive models.

UNIVERSITY OF CAMBRIDGE

[1] Garnelo, Marta, et al. "Conditional neural processes." arXiv preprint arXiv:1807.01613 (2018).

Probabilistic Models: old and new

- **Traditional multi-task modelling approach:**

τ = task
θ = global parameters
ψ^τ = task specific params

$$p(y^\tau | x^\tau, \theta, \psi^\tau)$$

Train task specific parameters on task data $\mathcal{D}^\tau = \{x_n^\tau, y_n^\tau\}_{n=1}^N$ giving $\hat{\psi}^\tau$

Predict using: $p(y^* | x^*, \theta, \hat{\psi}^\tau)$

- **New approach: directly specify the predictive in terms of the task data**

$$p(y^* | x^*, \theta, \psi_\phi(\mathcal{D}^\tau))$$

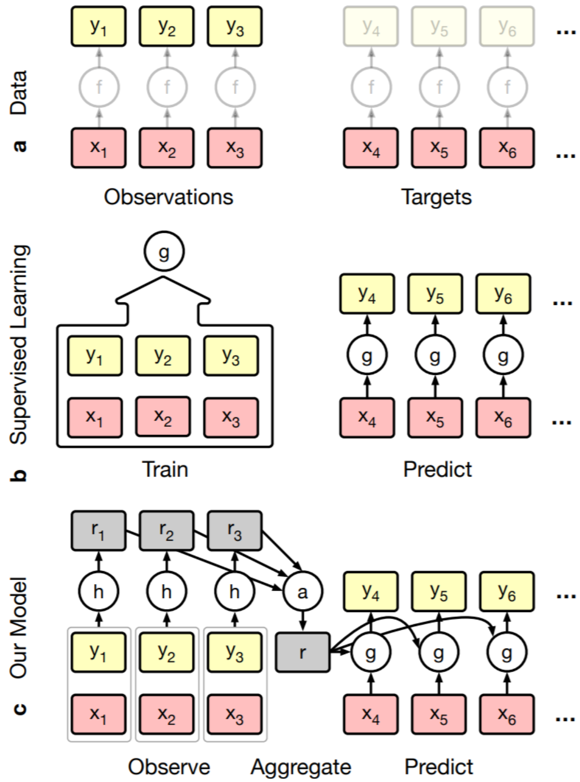
Relates to Conditional Neural Processes [1] and auto-regressive models.

ϕ is trained via maximum likelihood.

UNIVERSITY OF CAMBRIDGE

[1] Garnelo, Marta, et al. "Conditional neural processes." arXiv preprint arXiv:1807.01613 (2018).

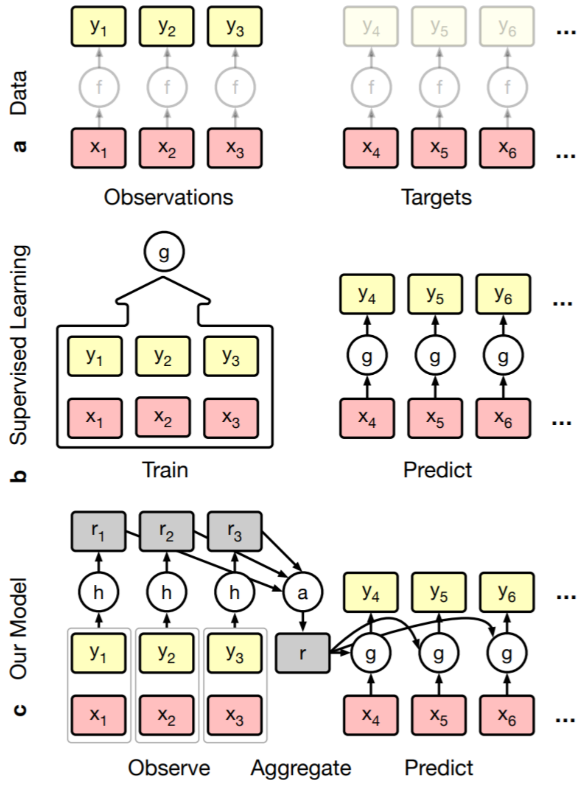
Conditional Neural Processes



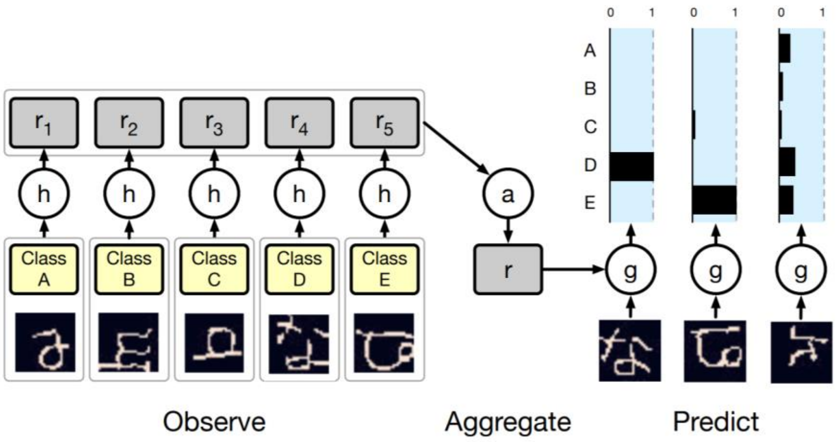
Regression

[1] Garnelo, Marta, et al. "Conditional neural processes." arXiv preprint arXiv:1807.01613 (2018).

Conditional Neural Processes



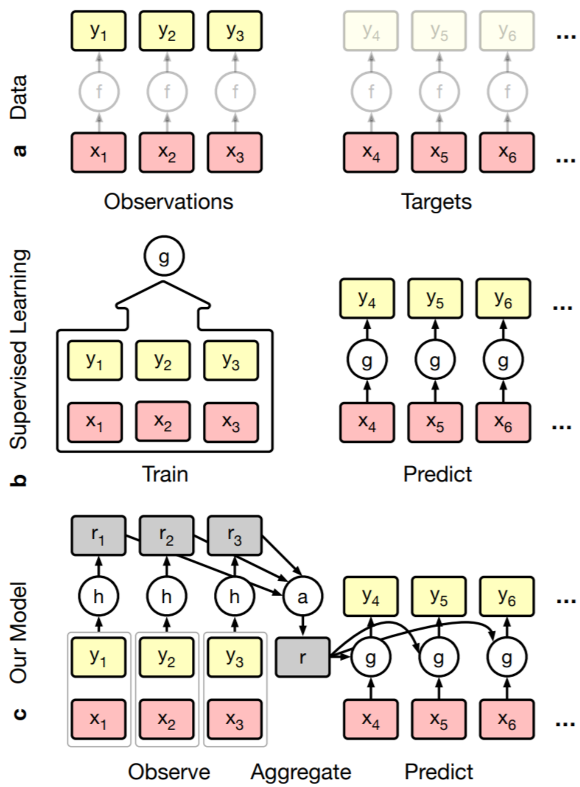
Regression



Classification

[1] Garnelo, Marta, et al. "Conditional neural processes." arXiv preprint arXiv:1807.01613 (2018).

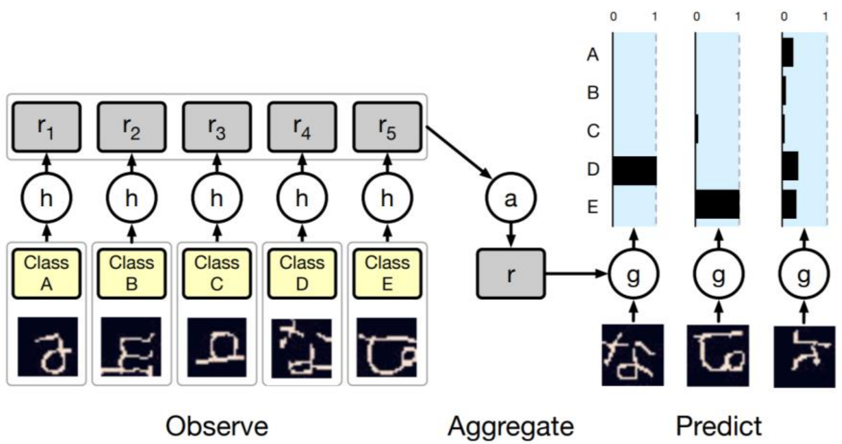
Conditional Neural Processes



Regression

For classification:

- Aggregation is class by class.
- 'r' grows with number of classes
- Parameters in 'g' grow with number of classes
- CNP limited to fixed way classification



Classification

[1] Garnelo, Marta, et al. "Conditional neural processes." arXiv preprint arXiv:1807.01613 (2018).

CNAPs vs CNPs

CNAPs *specialize* CNPs to the multi-task classification setting:

CNAPs vs CNPs

CNAPs *specialize* CNPs to the multi-task classification setting:

- The task-specific parameters ψ^{τ} are parameters of the model itself versus CNPs fixed dimensional vector r as a model input.

CNAPs vs CNPs

CNAPs *specialize* CNPs to the multi-task classification setting:

- The task-specific parameters ψ^τ are parameters of the model itself versus CNPs fixed dimensional vector r as a model input.
    - Allows for varying-way classification with a fixed number of parameters. *CNPs have pre-specified number of classes and the parameters grow (at least) linearly with the number of classes.*

CNAPs vs CNPs

CNAPs *specialize* CNPs to the multi-task classification setting:

- The task-specific parameters ψ^{τ} are parameters of the model itself versus CNPs fixed dimensional vector r as a model input.
  - Allows for varying-way classification with a fixed number of parameters. *CNPs have pre-specified number of classes and the parameters grow (at least) linearly with the number of classes.*
  - Allows modeling of explicit hierarchy in the data (task vs class vs global). *CNPs share all parameters across tasks and use r to adapt.*

CNAPs vs CNPs

CNAPs *specialize* CNPs to the multi-task classification setting:

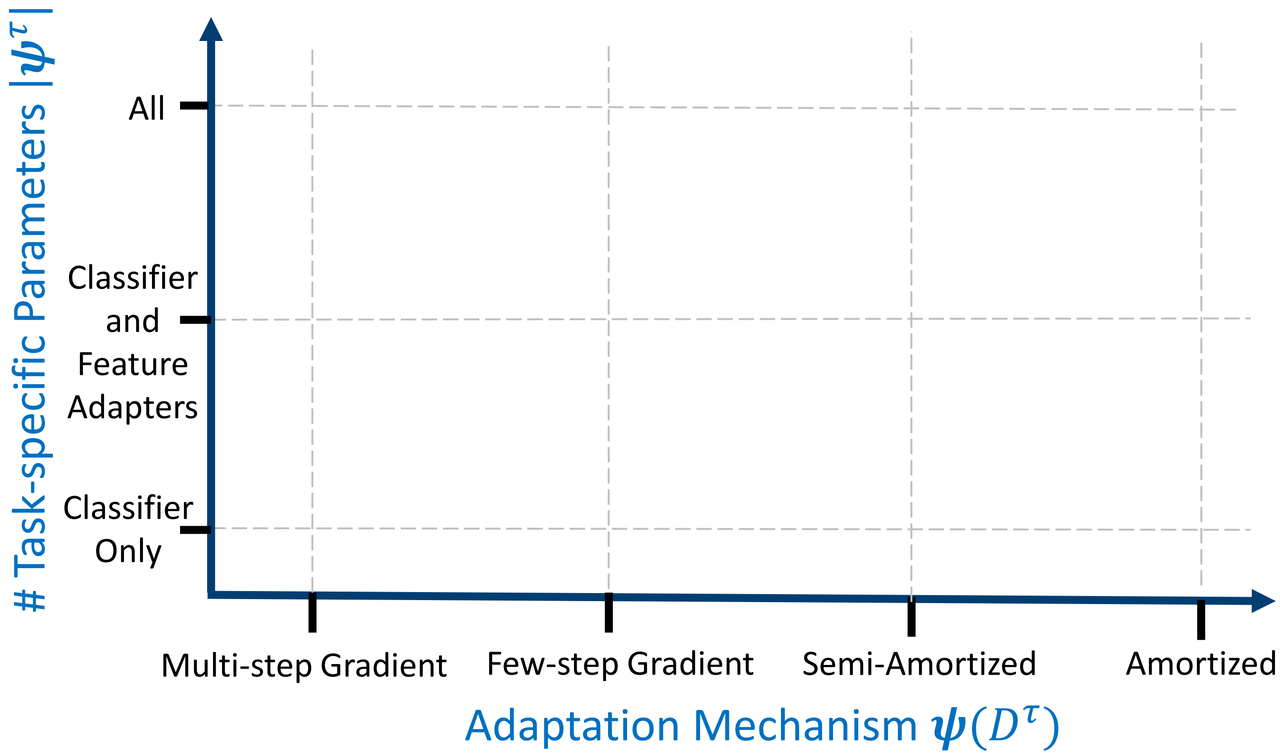
- The task-specific parameters ψ^{τ} are parameters of the model itself versus CNPs fixed dimensional vector r as a model input.
  - Allows for varying-way classification with a fixed number of parameters. *CNPs have pre-specified number of classes and the parameters grow (at least) linearly with the number of classes.*
  - Allows modeling of explicit hierarchy in the data (task vs class vs global). *CNPs share all parameters across tasks and use r to adapt.*
  - Allows support for continual learning in a natural way (more on this later)

CNAPs vs CNPs

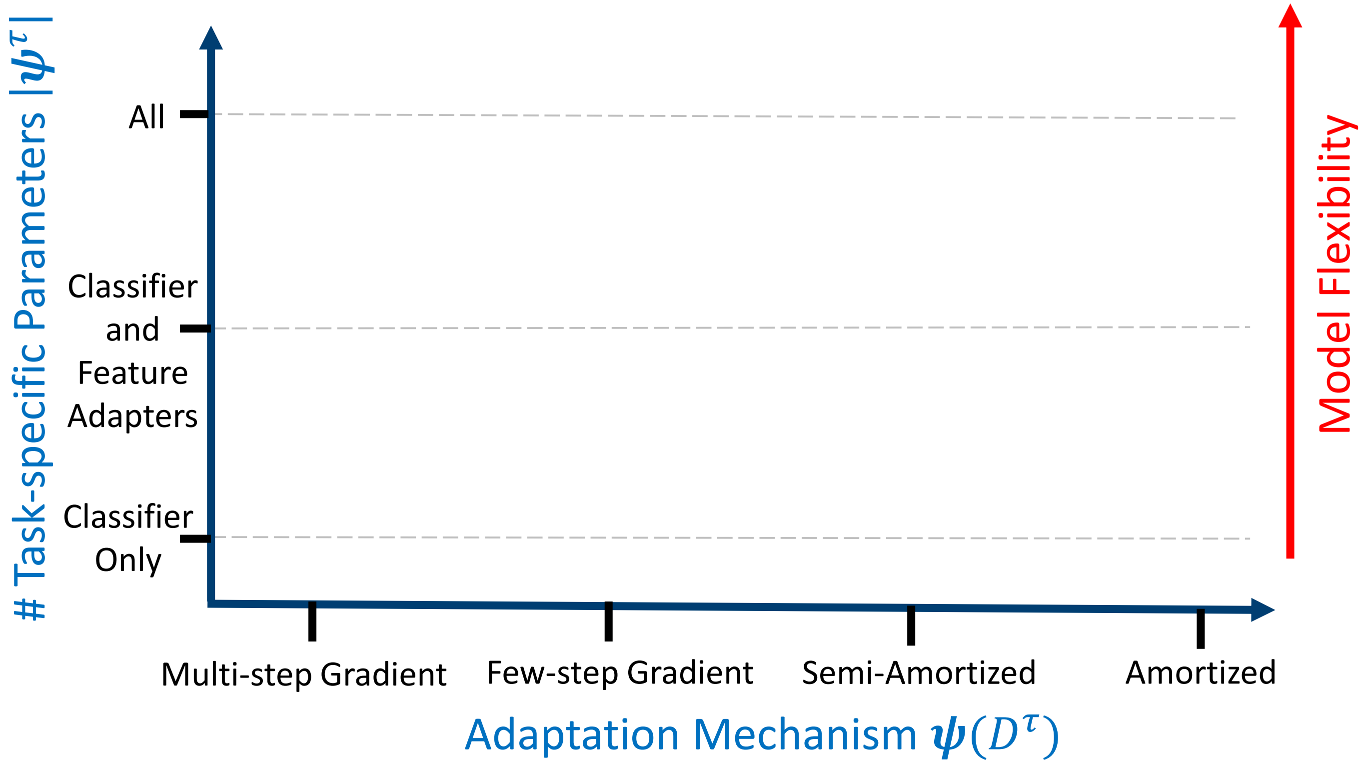
CNAPs *specialize* CNPs to the multi-task classification setting:

- The task-specific parameters ψ^{τ} are parameters of the model itself versus CNPs fixed dimensional vector r as a model input.
    - Allows for varying-way classification with a fixed number of parameters. *CNPs have pre-specified number of classes and the parameters grow (at least) linearly with the number of classes.*
    - Allows modeling of explicit hierarchy in the data (task vs class vs global). *CNPs share all parameters across tasks and use r to adapt.*
    - Allows support for continual learning in a natural way (more on this later)
- Two-step training procedure facilitates multi-task and transfer learning geared toward to diverse tasks.

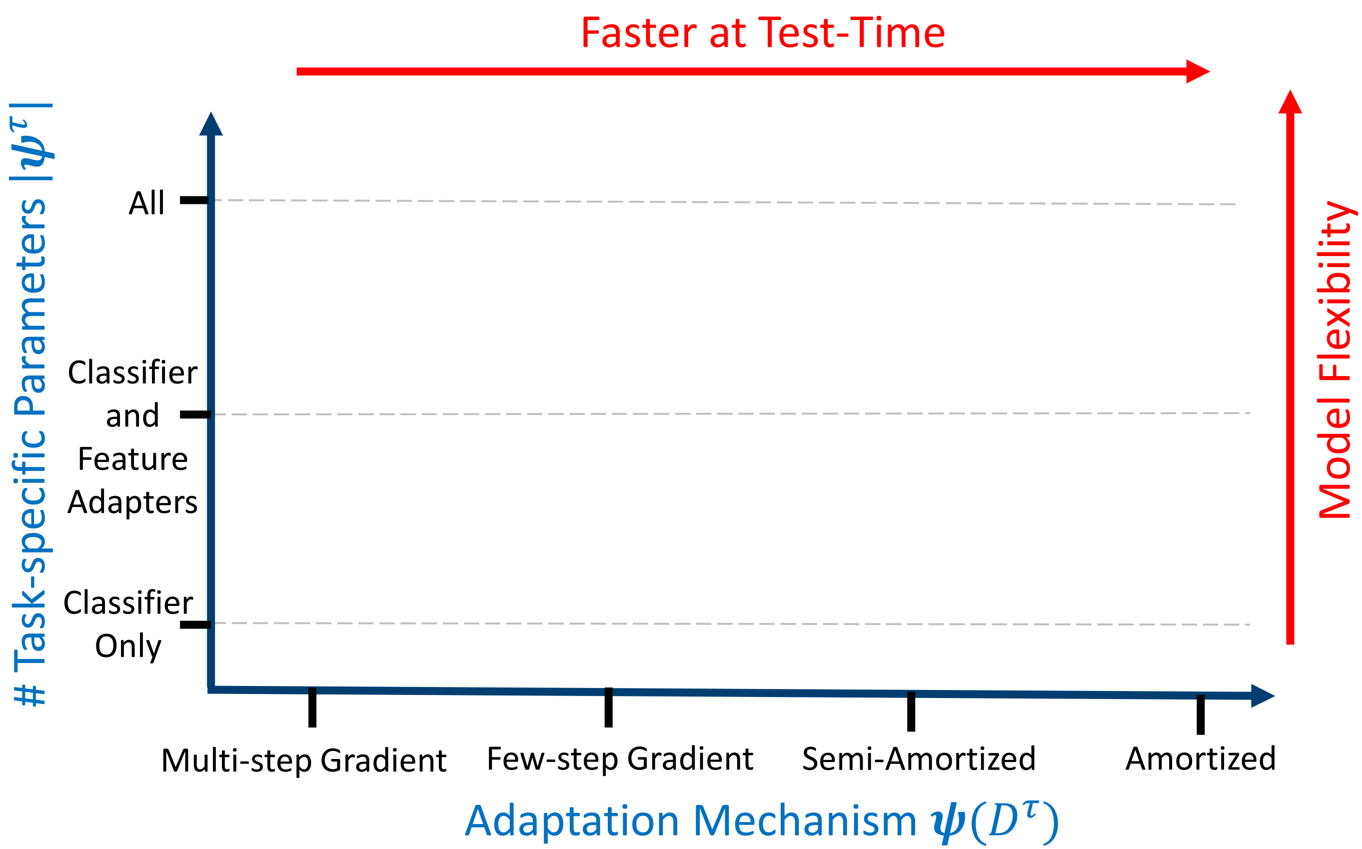
Competitive Landscape



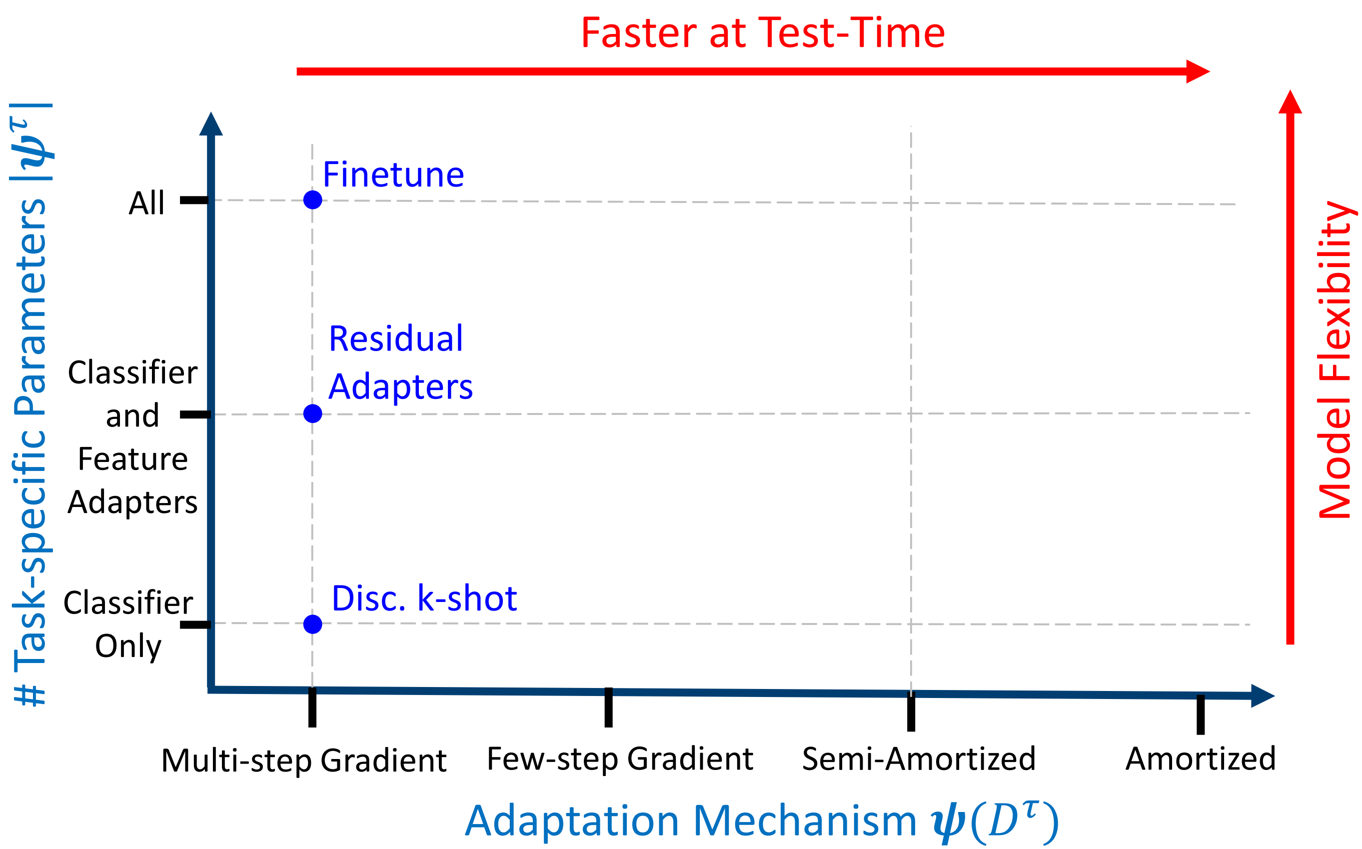
Competitive Landscape



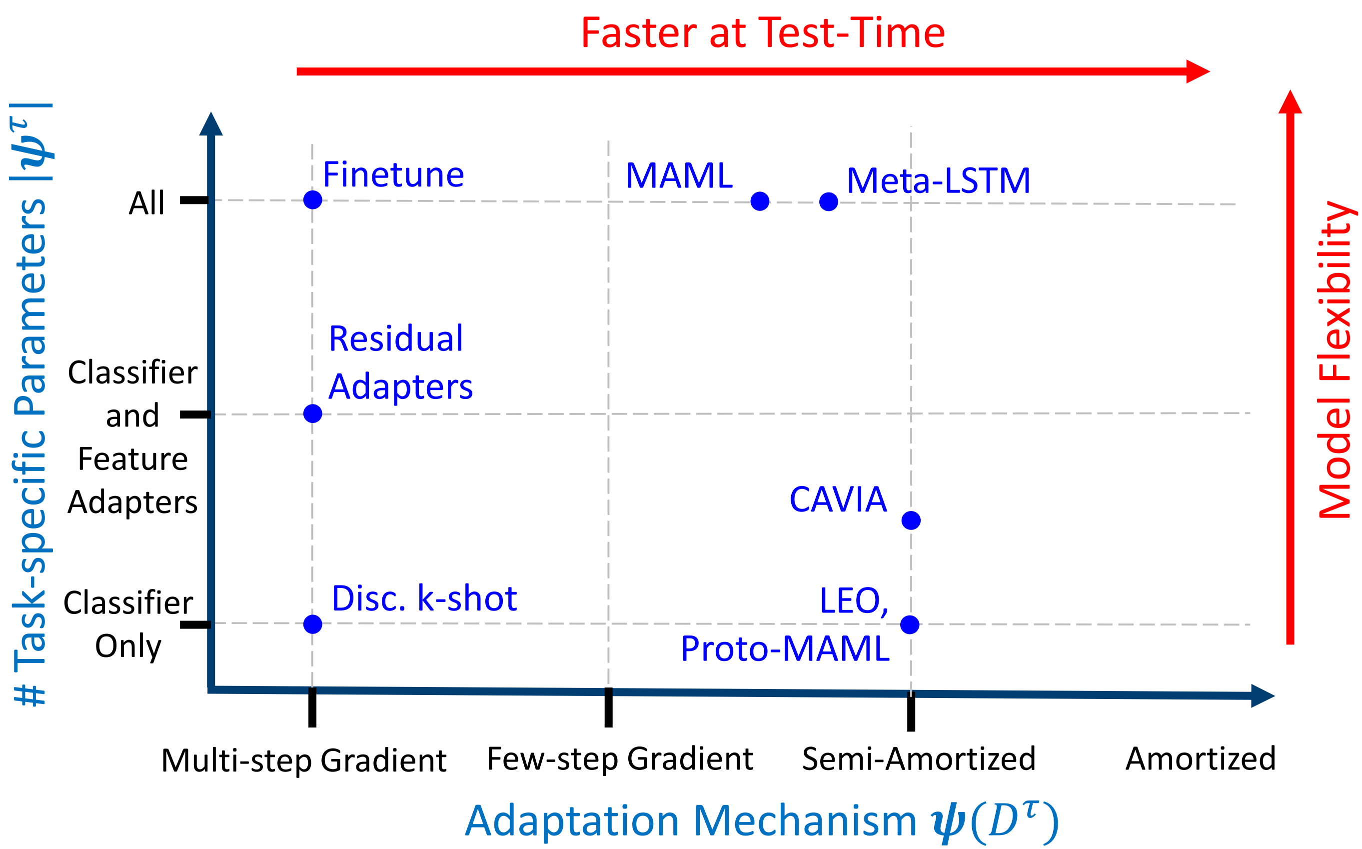
Competitive Landscape



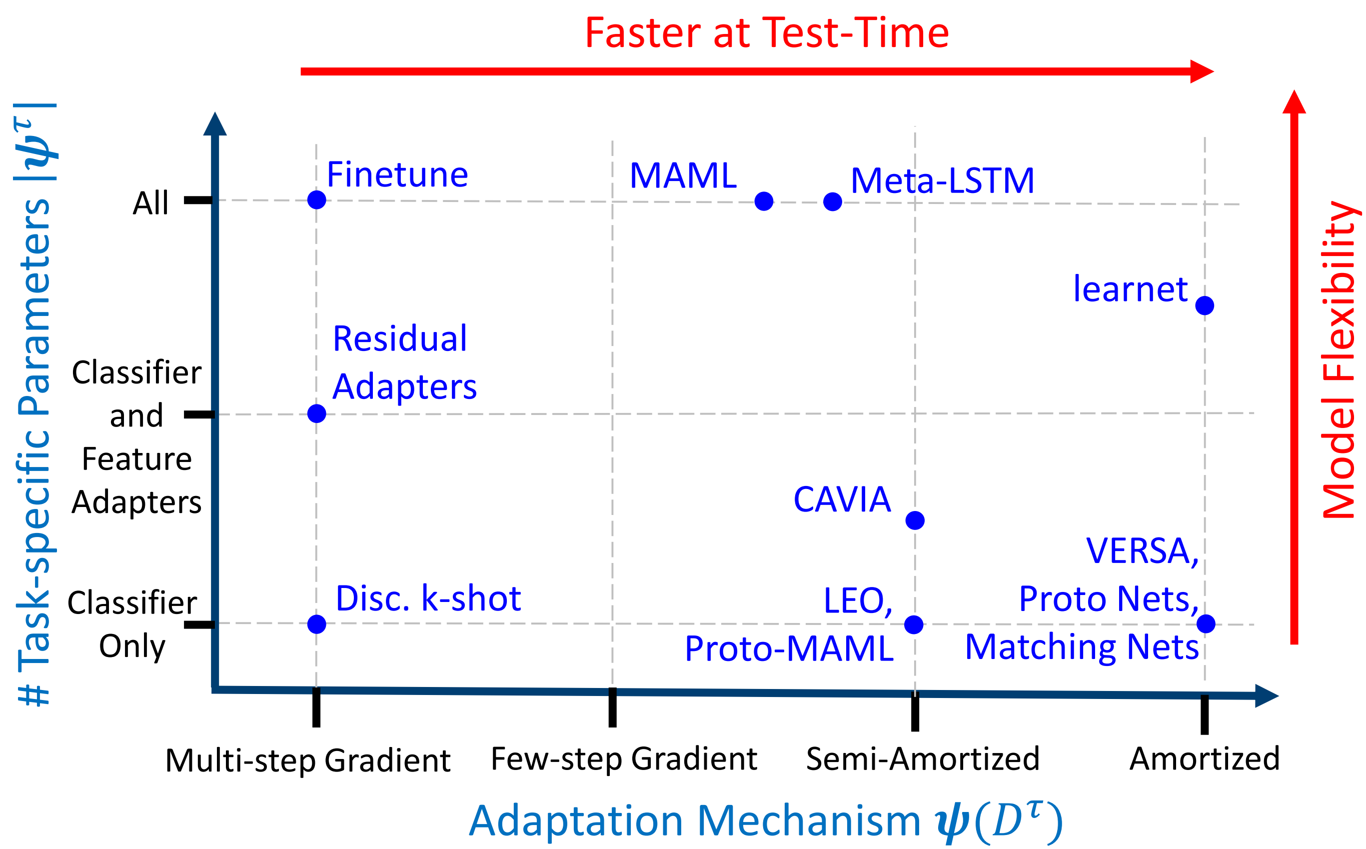
Competitive Landscape



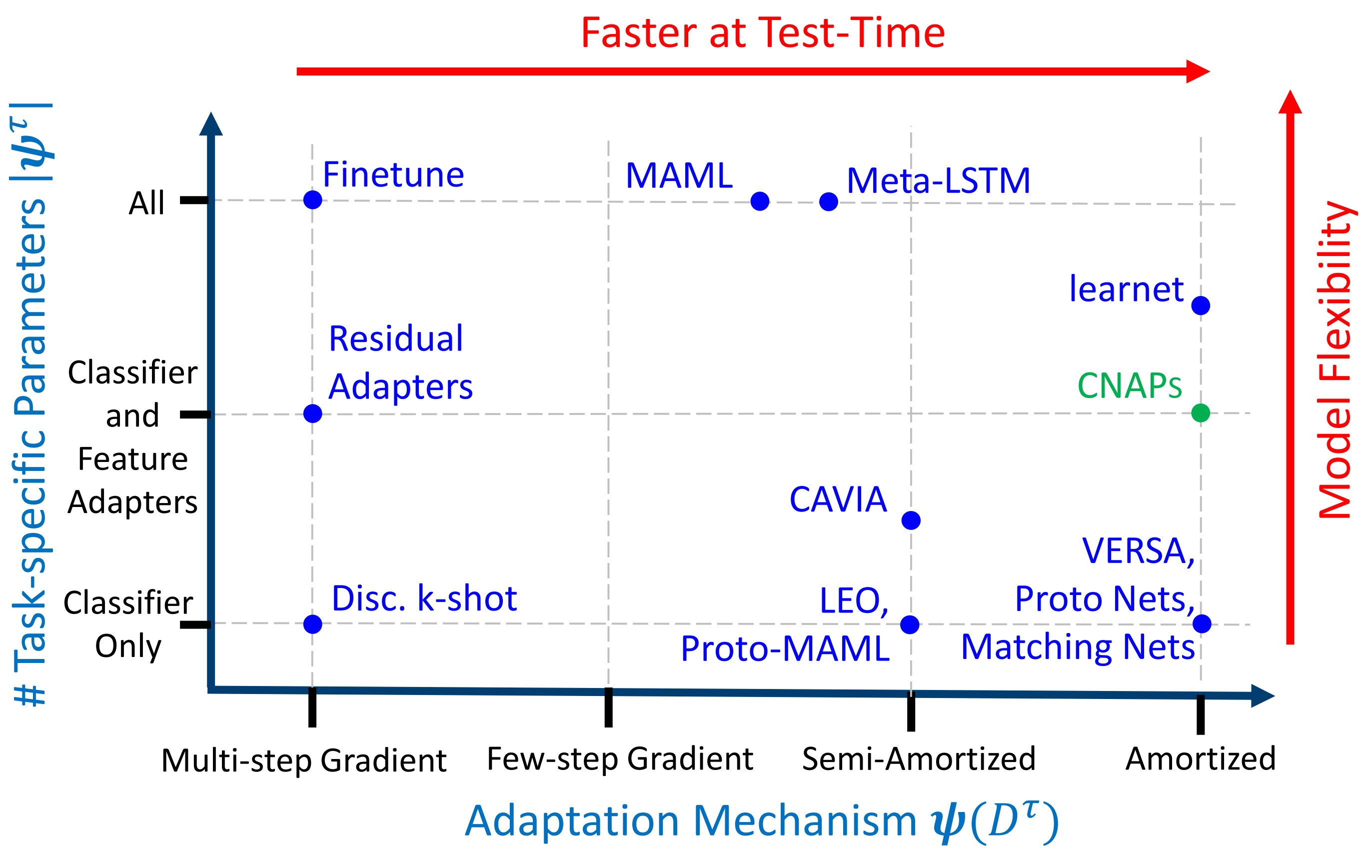
Competitive Landscape



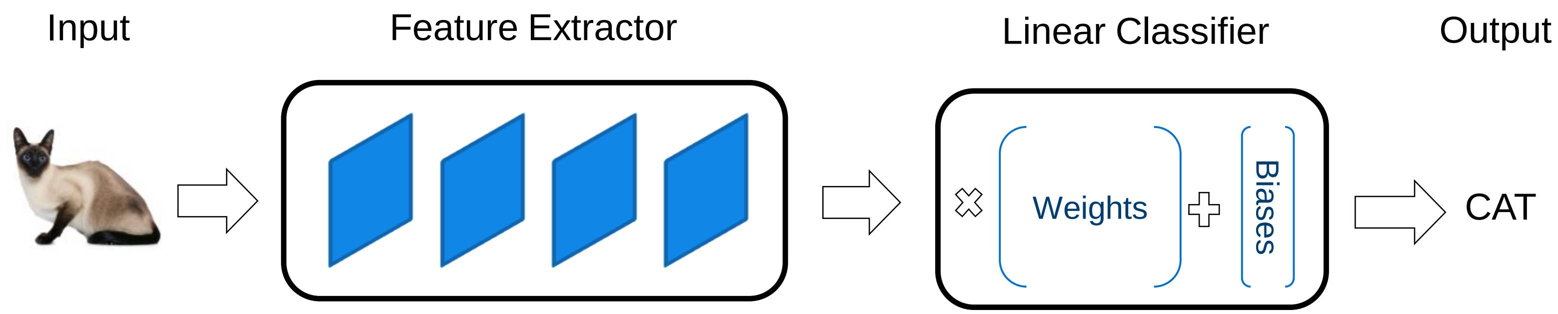
Competitive Landscape



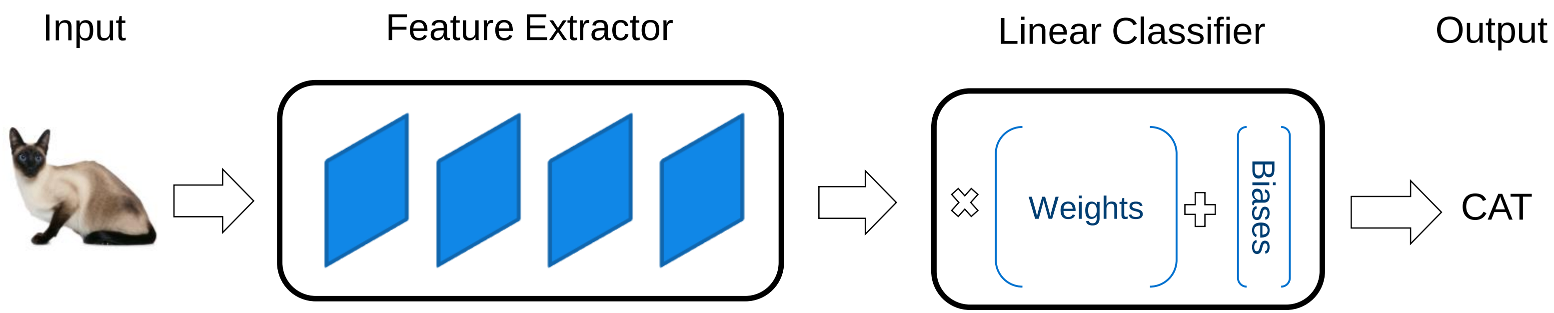
Competitive Landscape



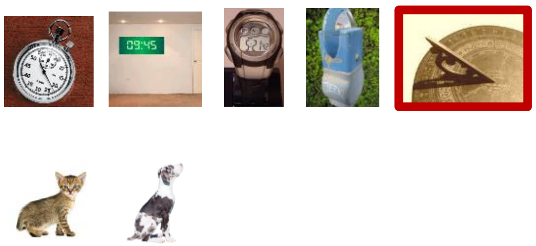
Task specific parameters: adapting a classifier



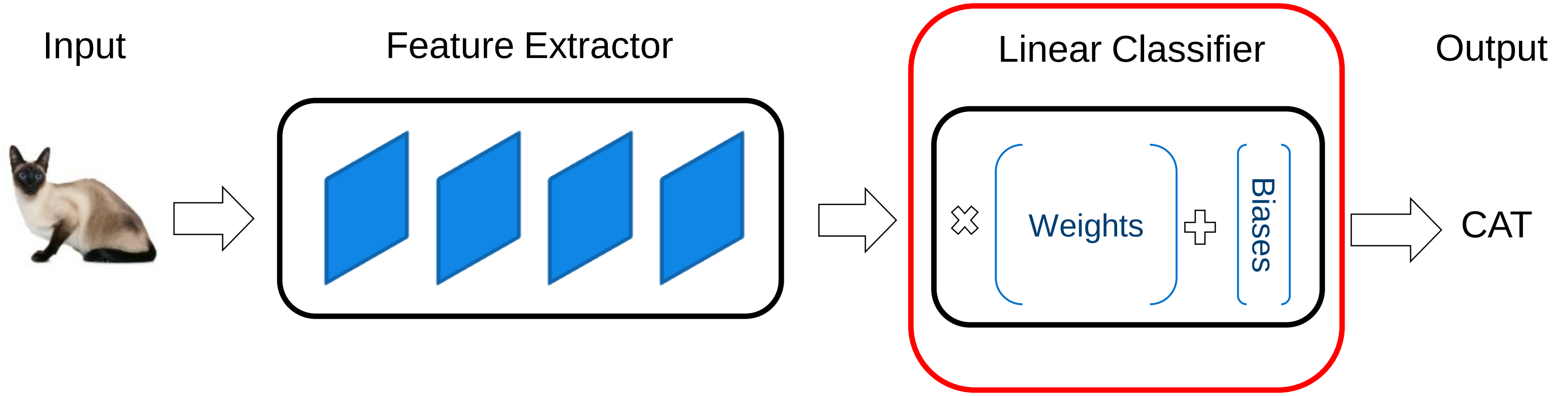
Task specific parameters: adapting a classifier



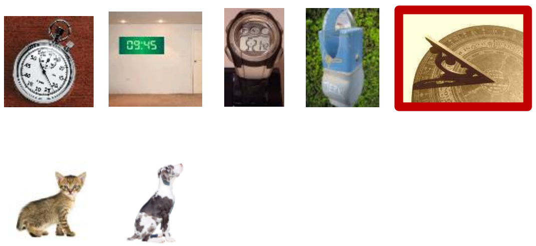
different head for each task
(different classes and different numbers of classes in each task)



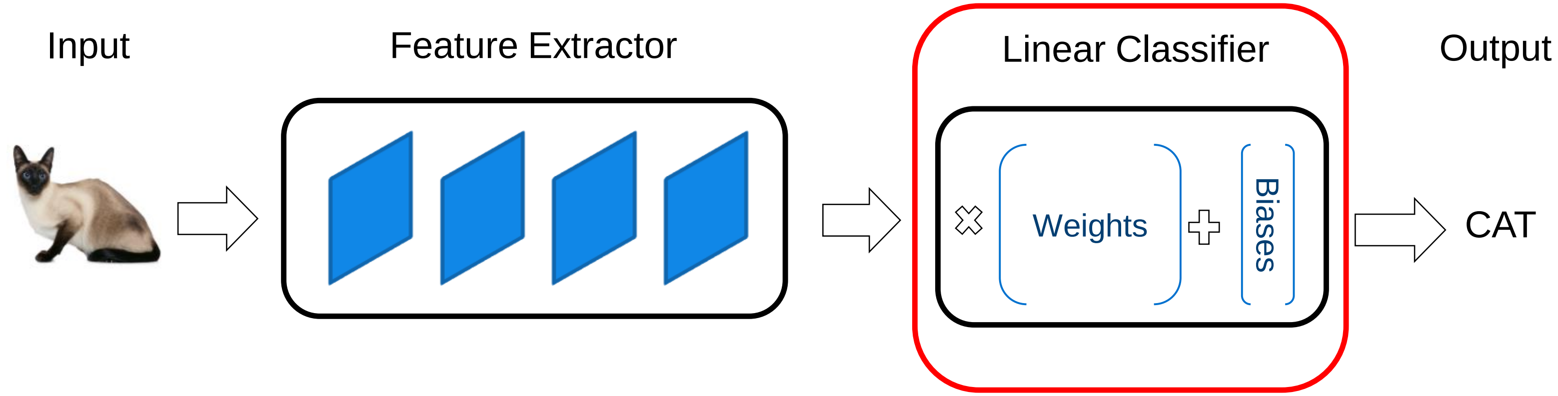
Task specific parameters: adapting a classifier



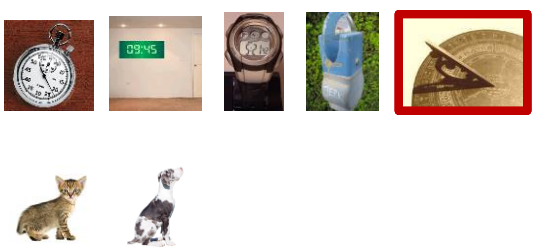
different head for each task
(different classes and different numbers of classes in each task)



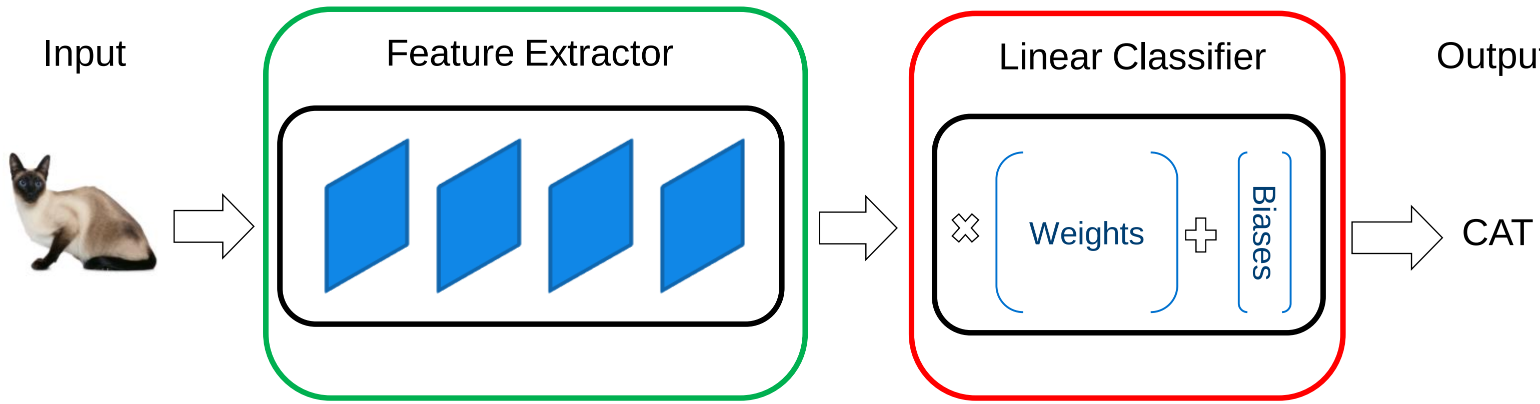
Task specific parameters: adapting a classifier



different head for each task
(different classes and different numbers of classes in each task)

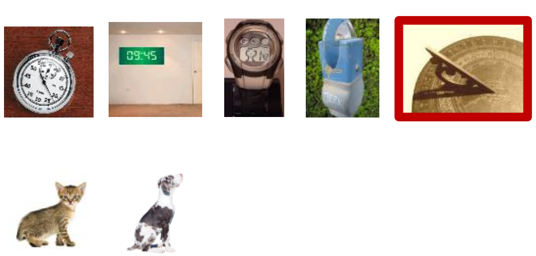


Task specific parameters: adapting a classifier



adapt feature extractor
(very different types of input image)

different head for each task
(different classes and different numbers of classes in each task)



Adapting the feature extractor: FiLM layers[1]

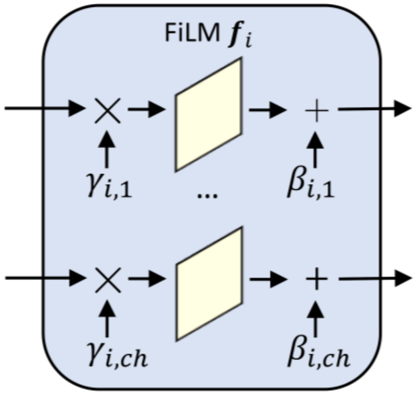
UNIVERSITY OF CAMBRIDGE

[1] Perez, Ethan, et al. "Film: Visual reasoning with a general conditioning layer." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

Adapting the feature extractor: FiLM layers[1]

- FiLM (Feature-wise Linear Modulation) layer:
    - scales (by factor γ); and
    - shifts (with offset β)

  convolutional layer feature maps.

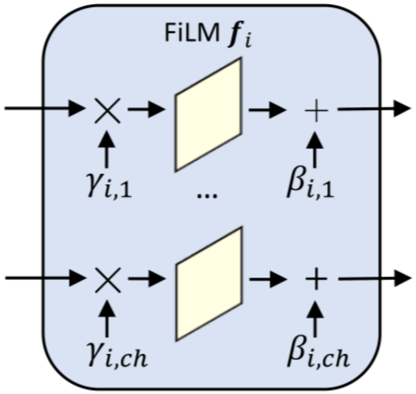


[1] Perez, Ethan, et al. "Film: Visual reasoning with a general conditioning layer." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

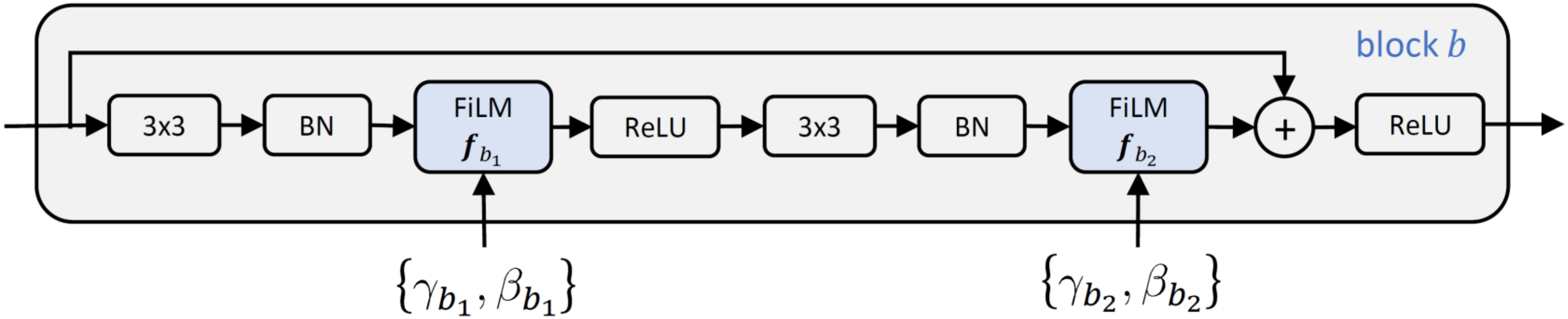
Adapting the feature extractor: FiLM layers[1]

- FiLM (Feature-wise Linear Modulation) layer:
  - scales (by factor γ); and
  - shifts (with offset β)

  convolutional layer feature maps.



- When added to a ResNet block:
  - enable expressive feature adaptation
  - add a small number of parameters (< 1% of the system).

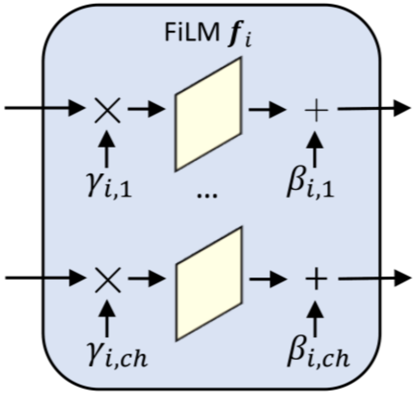


[1] Perez, Ethan, et al. "Film: Visual reasoning with a general conditioning layer." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

Adapting the feature extractor: FiLM layers[1]

- FiLM (Feature-wise Linear Modulation) layer:
  - scales (by factor γ); and
  - shifts (with offset β)

  convolutional layer feature maps.



- When added to a ResNet block:
  - enable expressive feature adaptation
  - add a small number of parameters (< 1% of the system).

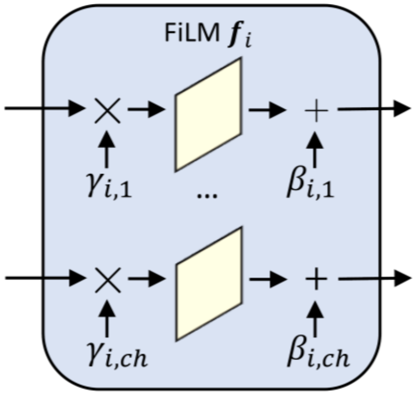
UNIVERSITY OF CAMBRIDGE

[1] Perez, Ethan, et al. "Film: Visual reasoning with a general conditioning layer." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

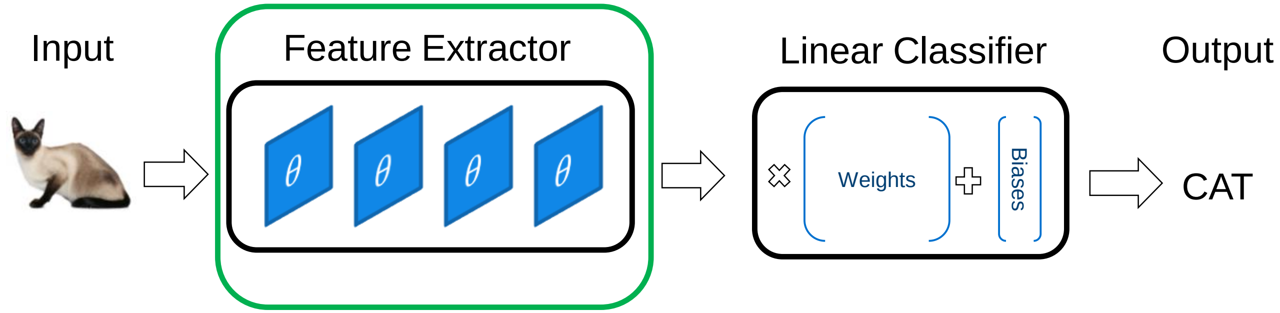
Adapting the feature extractor: FiLM layers[1]

- FiLM (Feature-wise Linear Modulation) layer:
  - scales (by factor γ); and
  - shifts (with offset β)

  convolutional layer feature maps.



- When added to a ResNet block:
  - enable expressive feature adaptation
  - add a small number of parameters (< 1% of the system).



- θ parameters are learned via pre-training (e.g. on ImageNet)

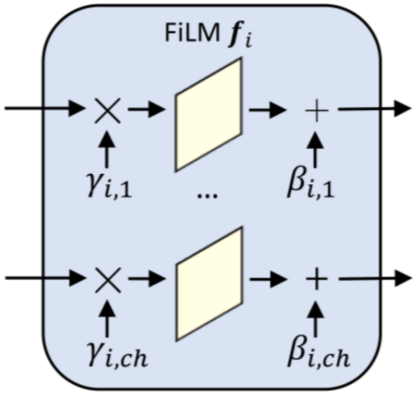
UNIVERSITY OF CAMBRIDGE

[1] Perez, Ethan, et al. "Film: Visual reasoning with a general conditioning layer." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

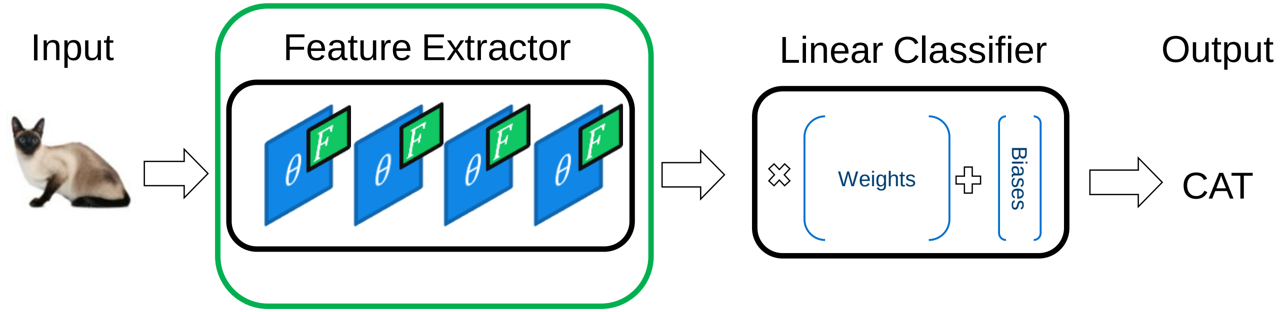
Adapting the feature extractor: FiLM layers[1]

- FiLM (Feature-wise Linear Modulation) layer:
  - scales (by factor γ); and
  - shifts (with offset β)

  convolutional layer feature maps.



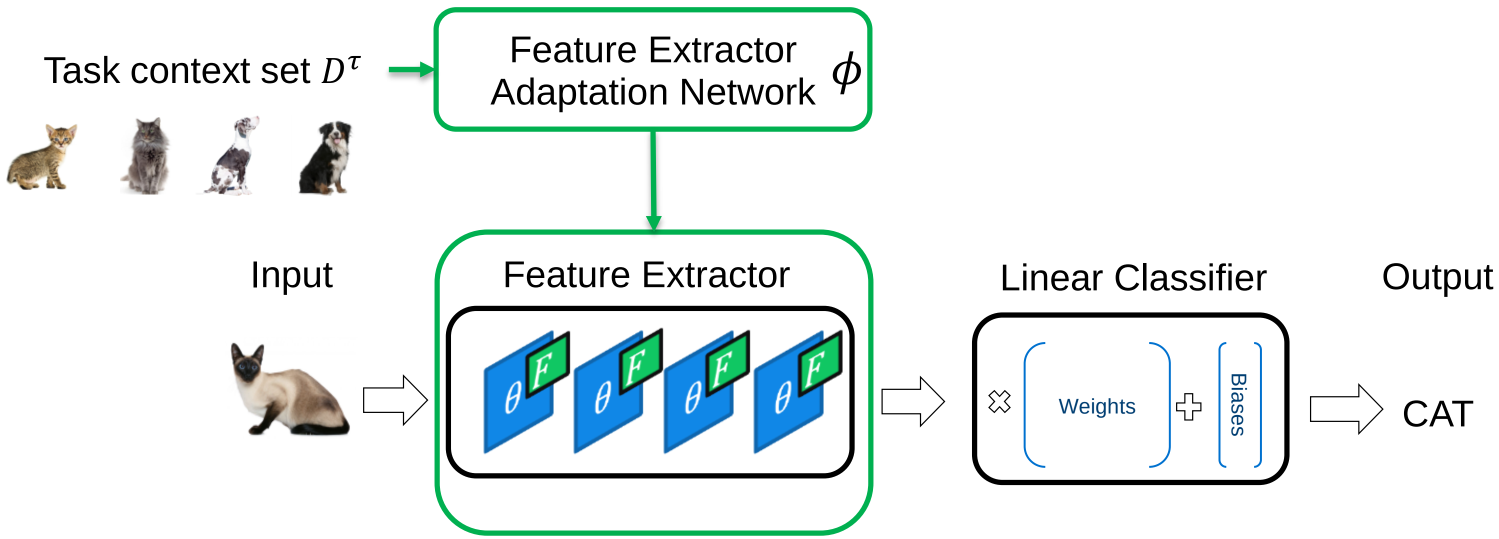
- When added to a ResNet block:
  - enable expressive feature adaptation
  - add a small number of parameters (< 1% of the system).



- θ parameters are learned via pre-training (e.g. on ImageNet)

[1] Perez, Ethan, et al. "Film: Visual reasoning with a general conditioning layer." Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

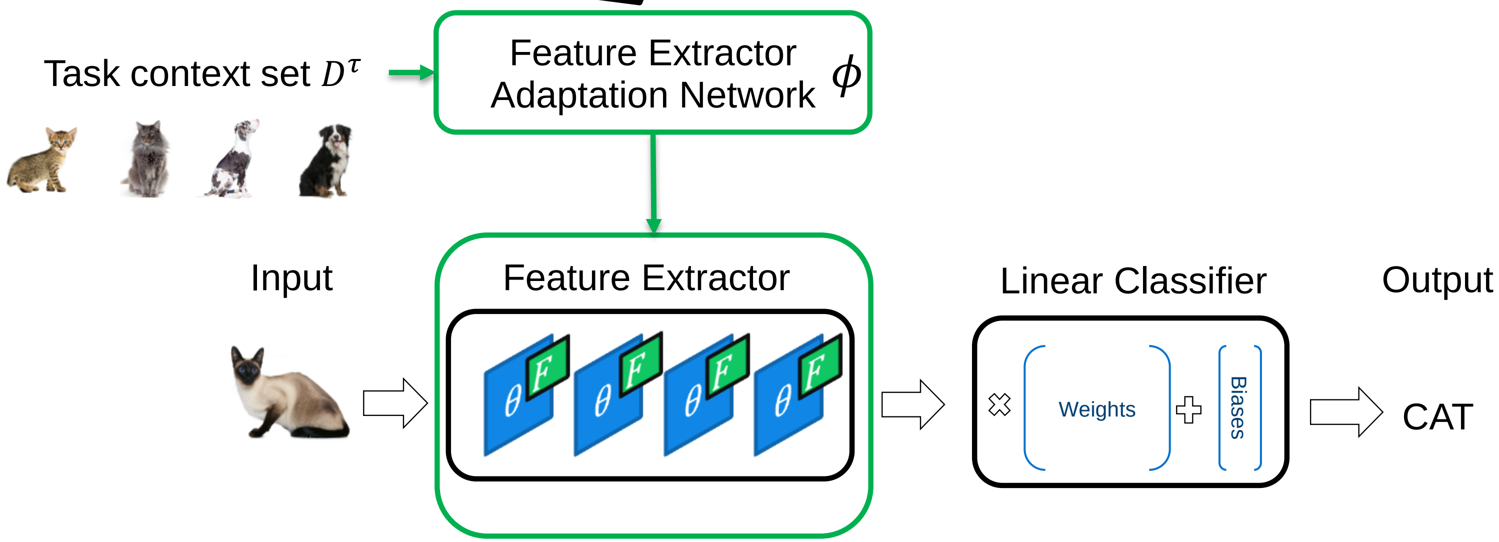
Feature Extractor Adaptation Network



Feature Extractor Adaptation Network

- PointNet[1] / Deep Sets[2] / Neural Statistician[3] statistics network → permutation invariance.

$$\psi(\mathcal{D}^\tau) = f\left(\sum_{n=1}^{N} g(x_n^\tau)\right)$$

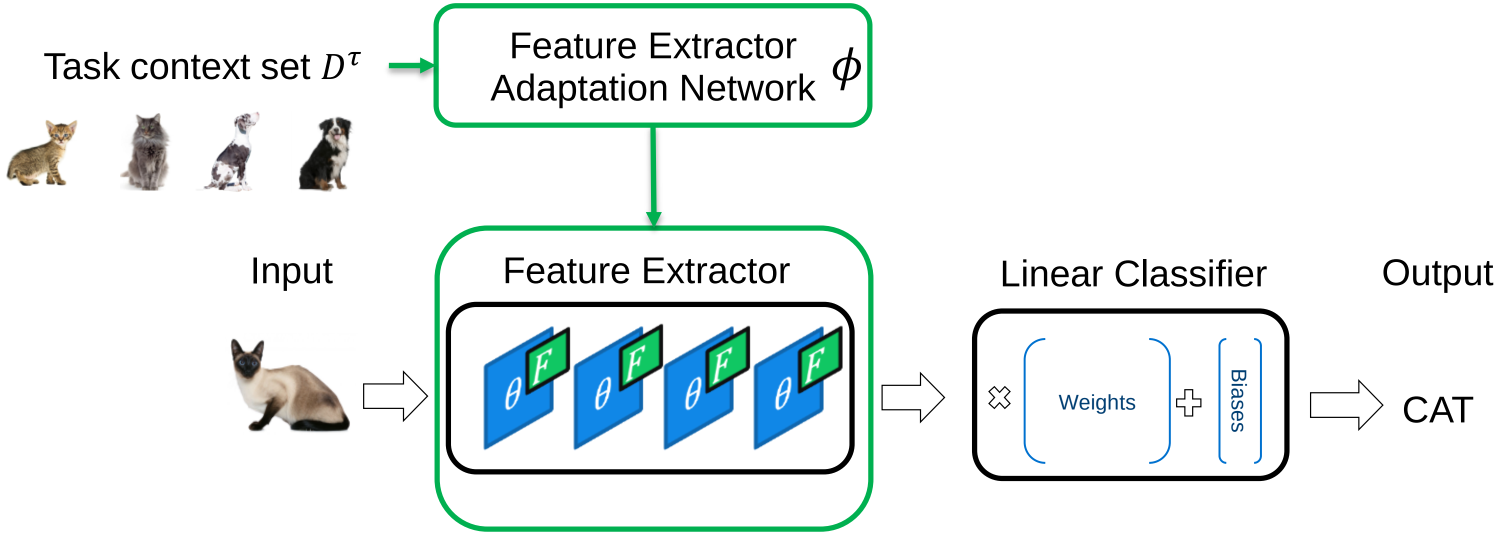


[1] Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017
[2] Zaheer, Manzil, et al. "Deep sets." *Advances in neural information processing systems*. 2017.
[3] Edwards, Harrison, and Amos Storkey. "Towards a neural statistician." *arXiv preprint arXiv:1606.02185* (2016).

Feature Extractor Adaptation Network

- PointNet[1] / Deep Sets[2] / Neural Statistician[3] statistics network → permutation invariance.
- MLP network maps set encoding to FiLM weights.

$$\psi(\mathcal{D}^\tau) = f\left(\sum_{n=1}^{N} g(x_n^\tau)\right)$$



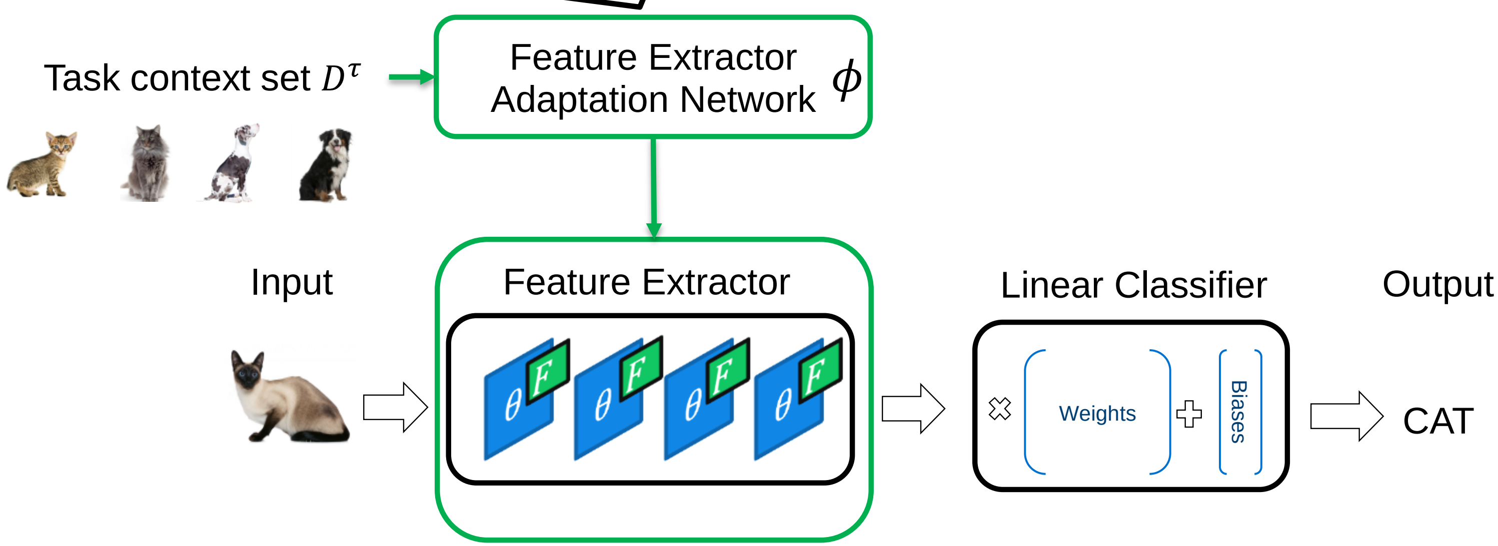
UNIVERSITY OF CAMBRIDGE

[1] Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017
[2] Zaheer, Manzil, et al. "Deep sets." *Advances in neural information processing systems*. 2017.
[3] Edwards, Harrison, and Amos Storkey. "Towards a neural statistician." *arXiv preprint arXiv:1606.02185* (2016).

Feature Extractor Adaptation Network

- PointNet[1] / Deep Sets[2] / Neural Statistician[3] statistics network → permutation invariance.
- MLP network maps set encoding to FiLM weights.
- Continual learning supported by running averages of task set encoding.

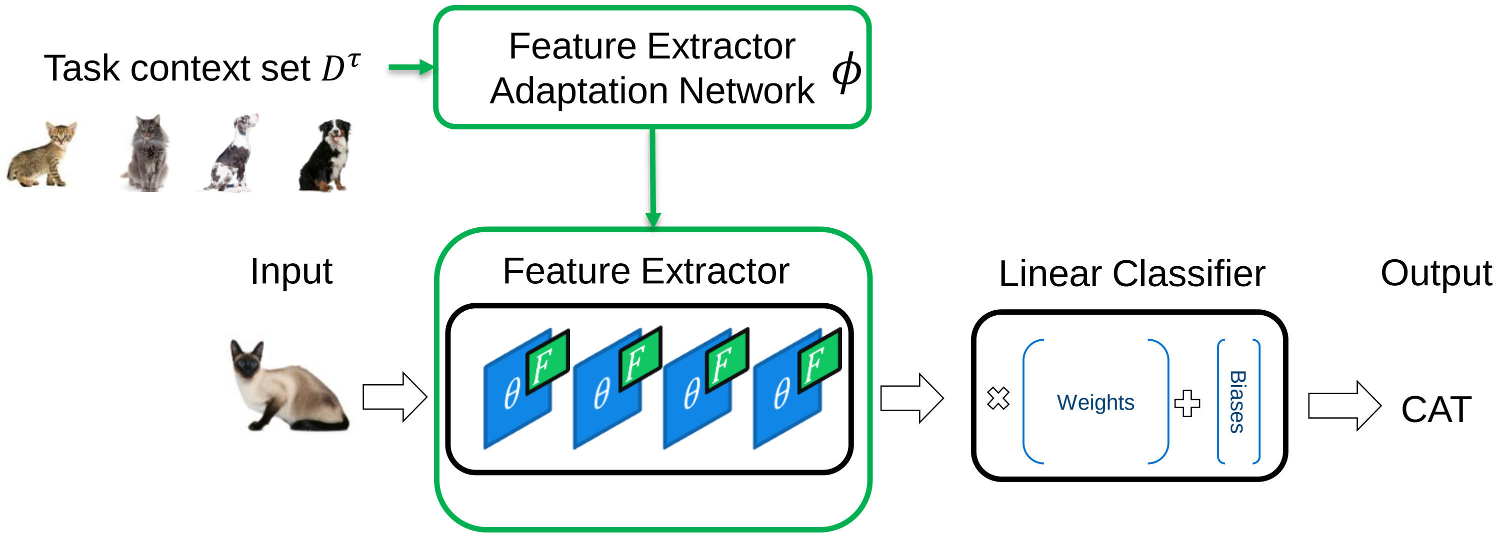
$$\psi(\mathcal{D}^\tau) = f\left(\sum_{n=1}^{N} g(x_n^\tau)\right)$$



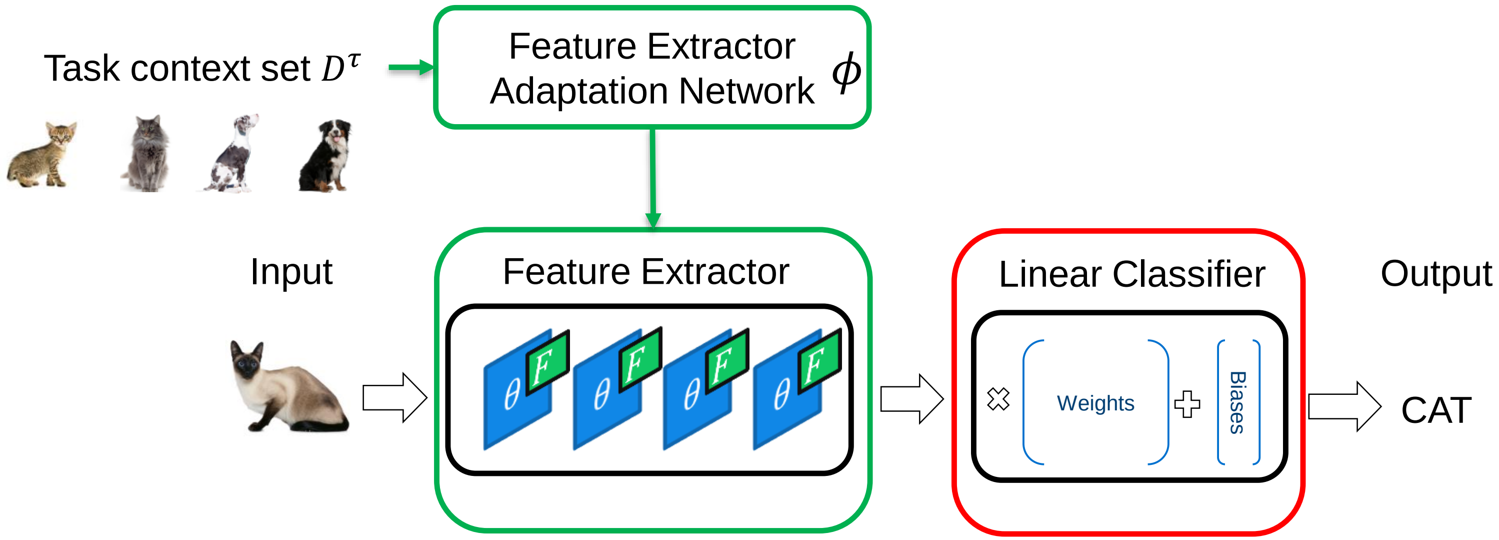
UNIVERSITY OF CAMBRIDGE

[1] Qi, Charles R., et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017
[2] Zaheer, Manzil, et al. "Deep sets." *Advances in neural information processing systems*. 2017.
[3] Edwards, Harrison, and Amos Storkey. "Towards a neural statistician." *arXiv preprint arXiv:1606.02185* (2016).

Classifier Head Adaptation Network

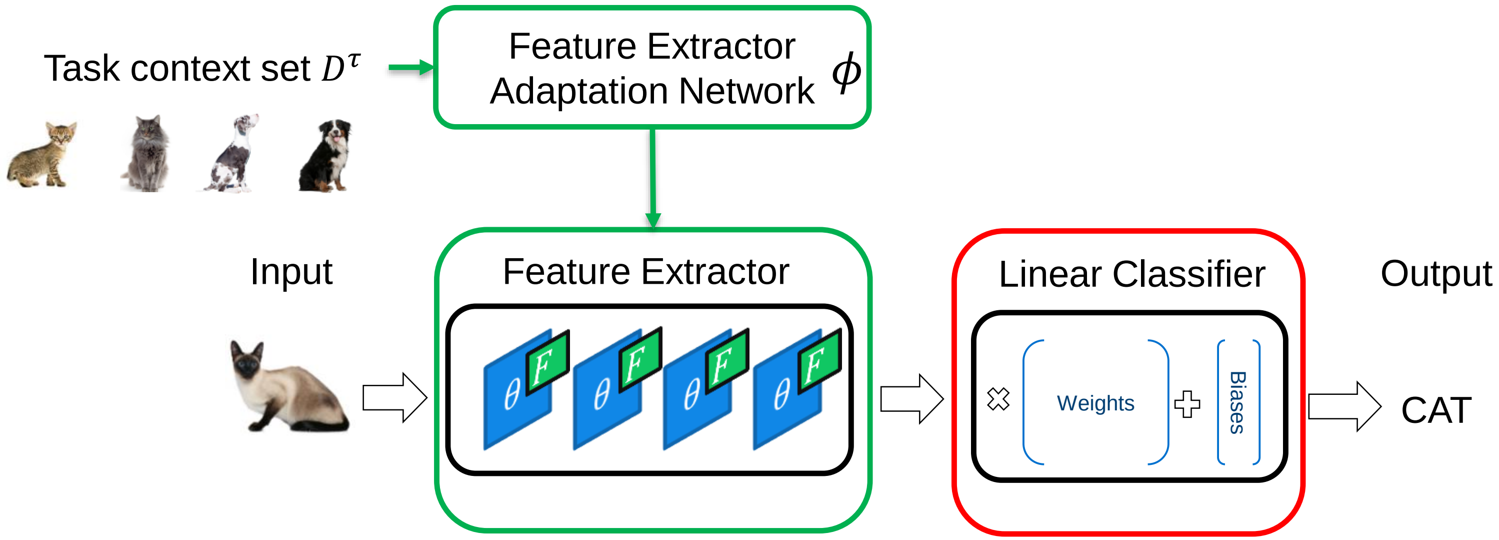


Classifier Head Adaptation Network



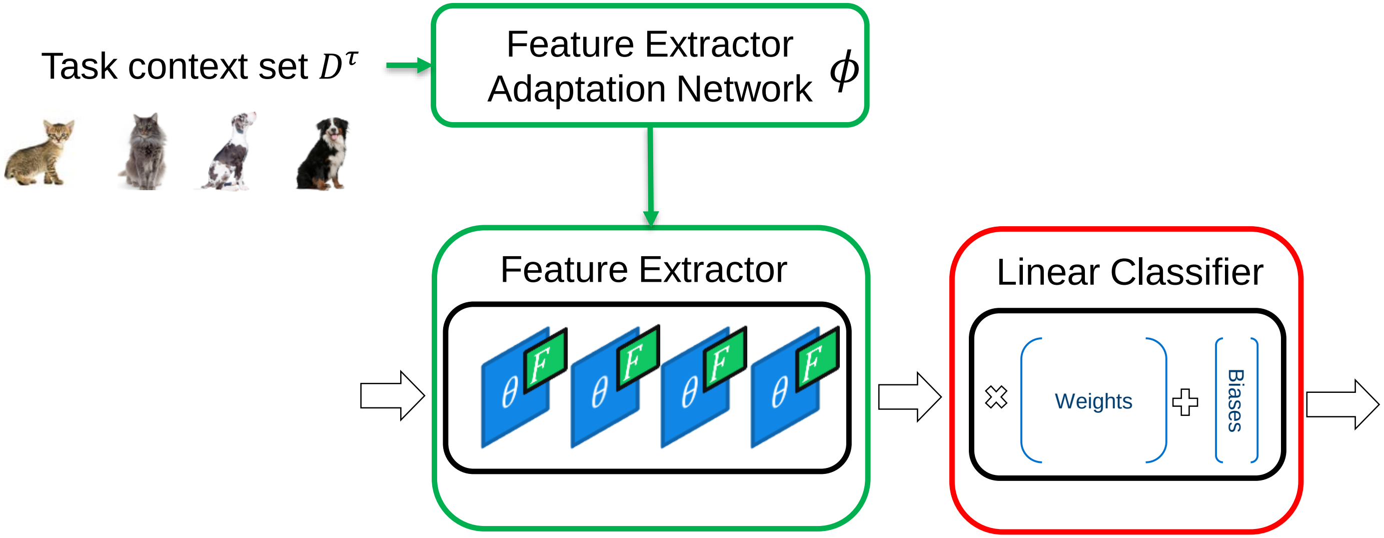
Classifier Head Adaptation Network

- Pass context set through feature extractor class by class.



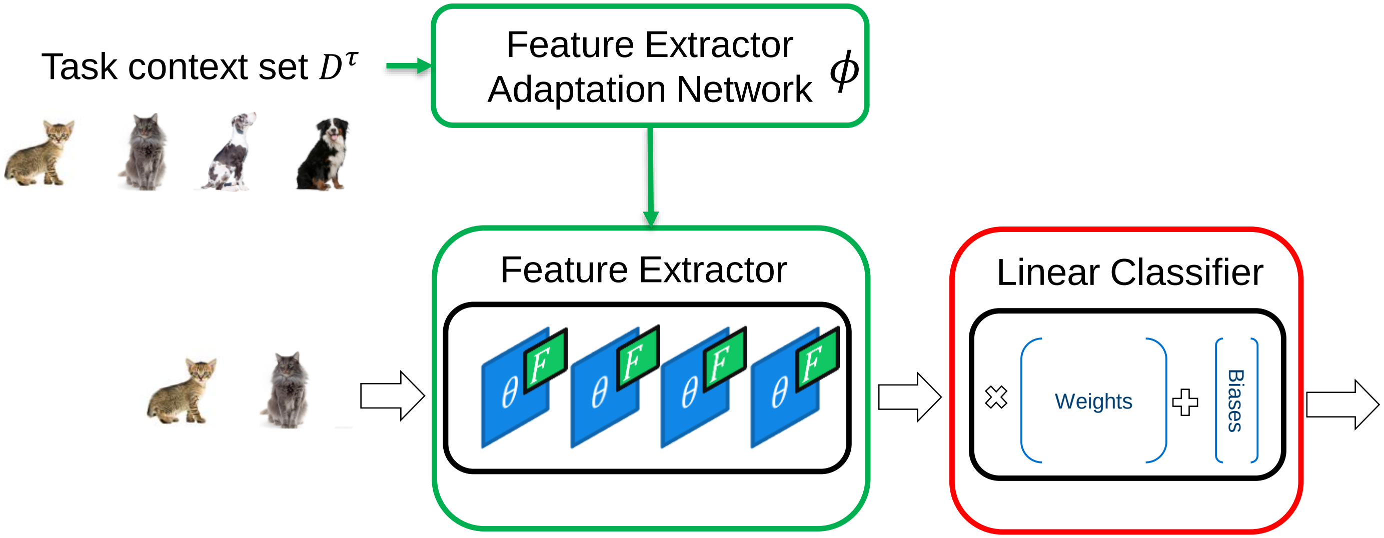
Classifier Head Adaptation Network

- Pass context set through feature extractor class by class.



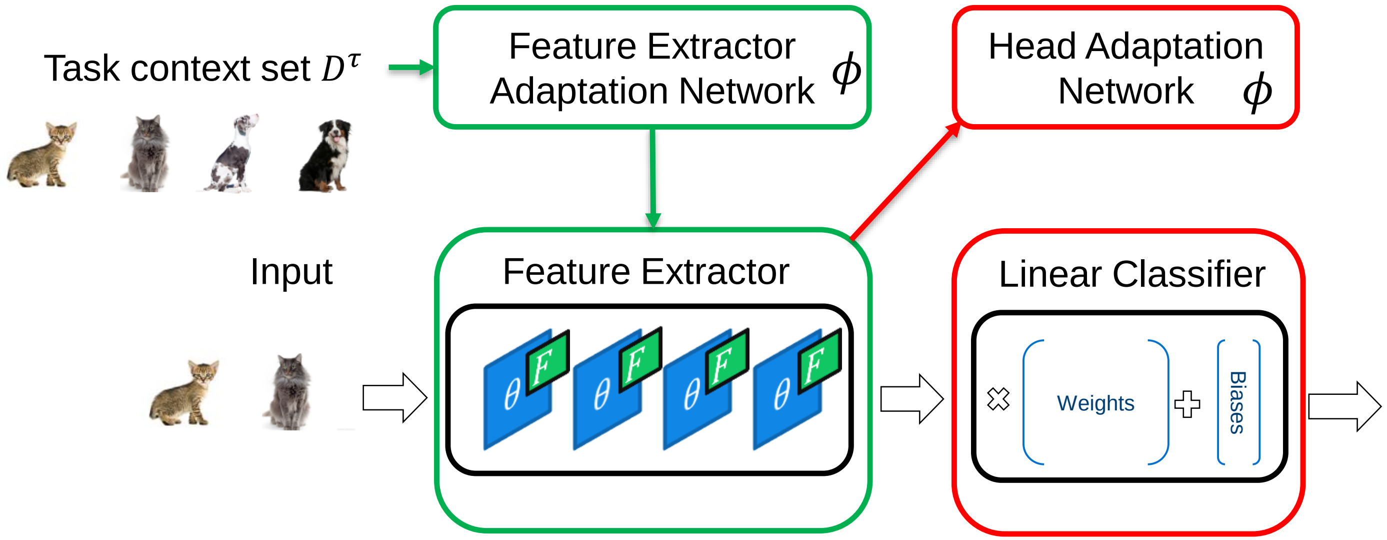
Classifier Head Adaptation Network

- Pass context set through feature extractor class by class.



Classifier Head Adaptation Network

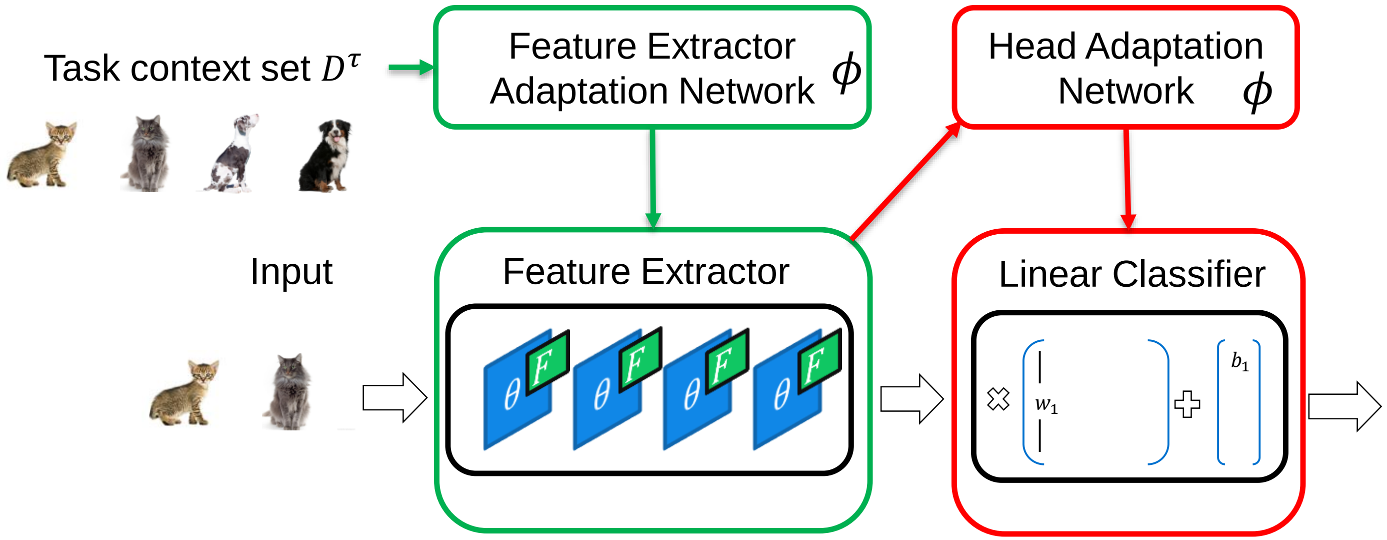
- Pass context set through feature extractor class by class.
- Features for each class c are averaged (*a la* PointNet / Deep Sets).



[1] Gordon, J. et al, "Meta-learning probabilistic inference for prediction". In *International Conference on Learning Representations*, 2019.

Classifier Head Adaptation Network

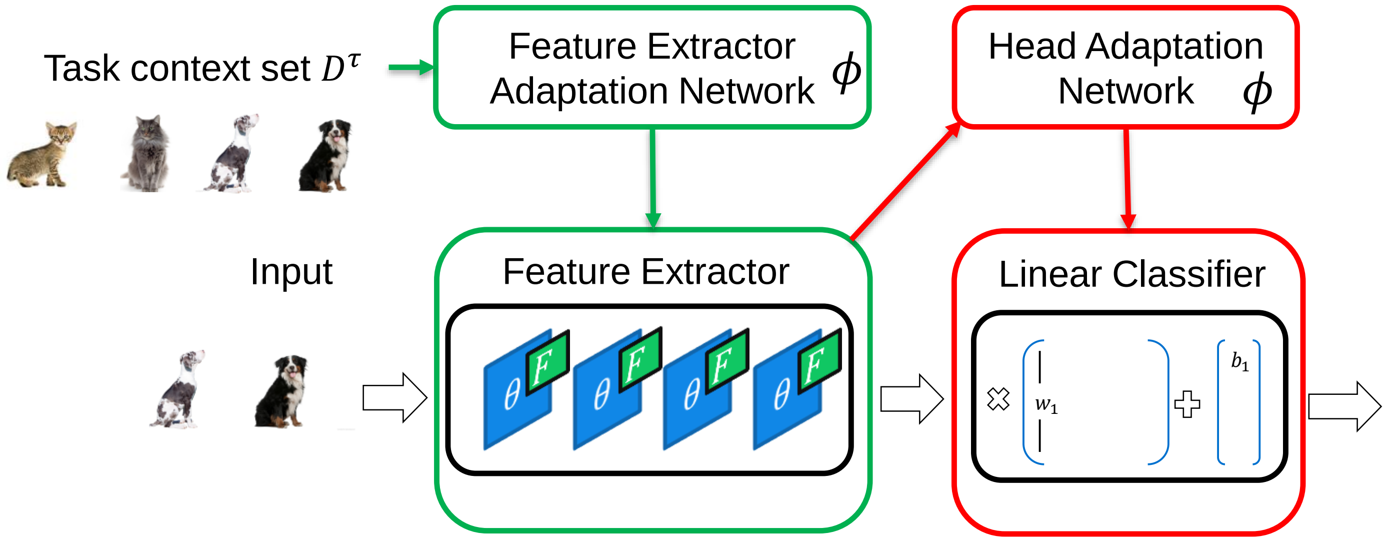
- Pass context set through feature extractor class by class.
- Features for each class c are averaged (*a la* PointNet / Deep Sets).
- Head adaptation network generates the weights and biases for class c.



[1] Gordon, J. et al, "Meta-learning probabilistic inference for prediction". In *International Conference on Learning Representations*, 2019.

Classifier Head Adaptation Network

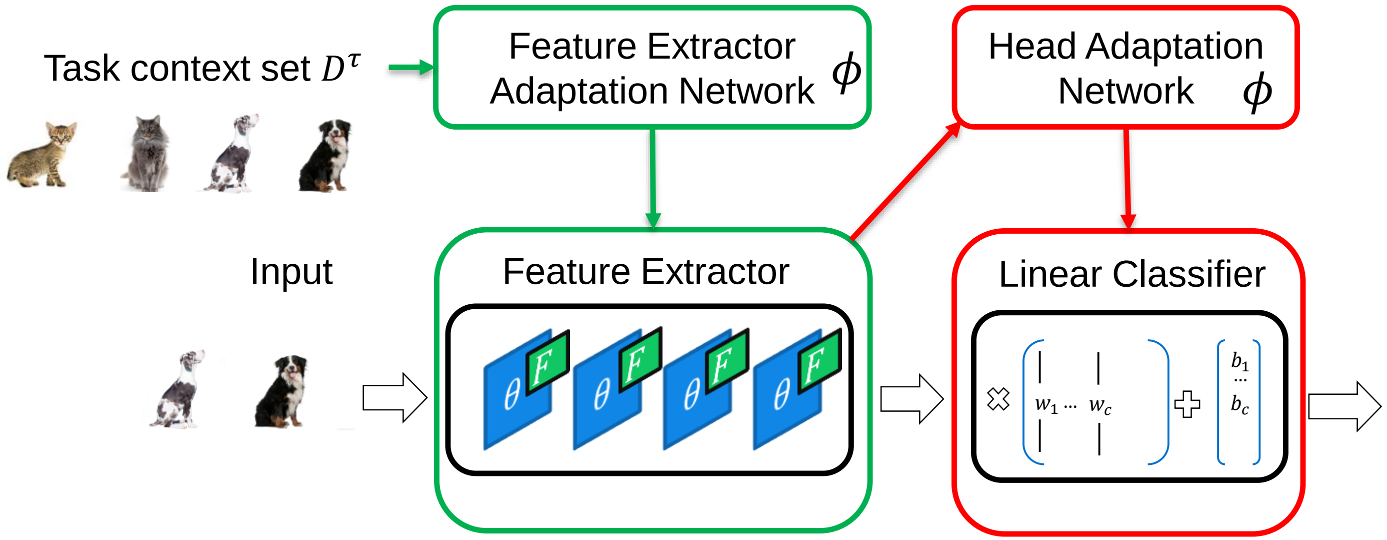
- Pass context set through feature extractor class by class.
- Features for each class c are averaged (*a la* PointNet / Deep Sets).
- Head adaptation network generates the weights and biases for class c.



[1] Gordon, J. et al, "Meta-learning probabilistic inference for prediction". In *International Conference on Learning Representations*, 2019.

Classifier Head Adaptation Network

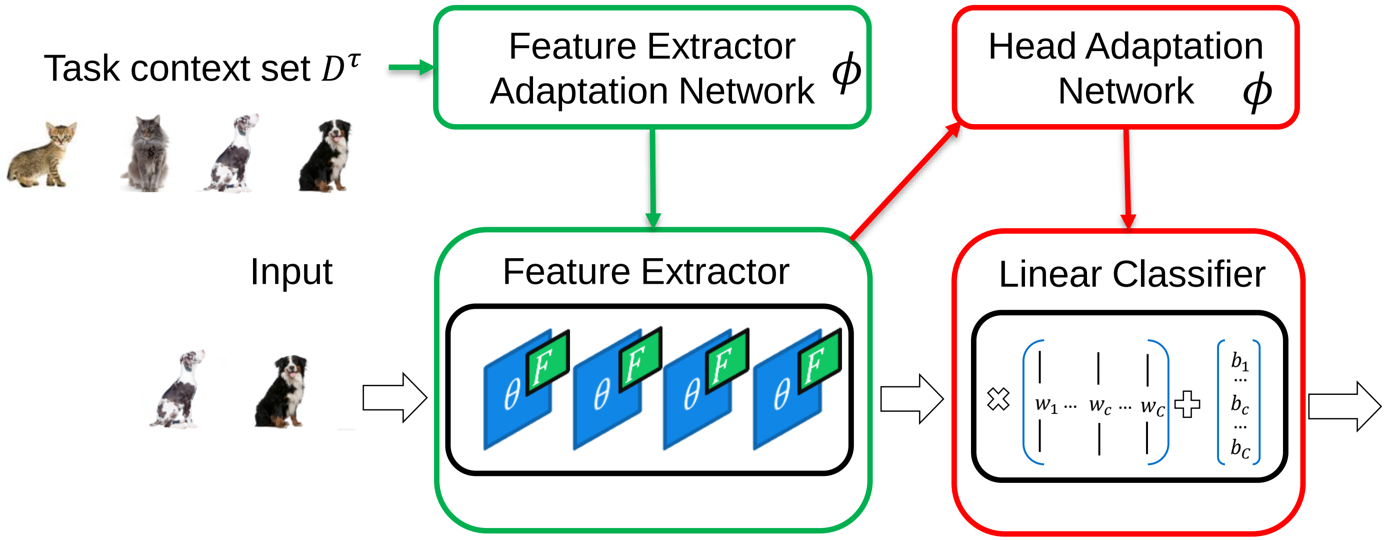
- Pass context set through feature extractor class by class.
- Features for each class c are averaged (*a la* PointNet / Deep Sets).
- Head adaptation network generates the weights and biases for class c.



[1] Gordon, J. et al, “Meta-learning probabilistic inference for prediction”. In *International Conference on Learning Representations*, 2019.

Classifier Head Adaptation Network

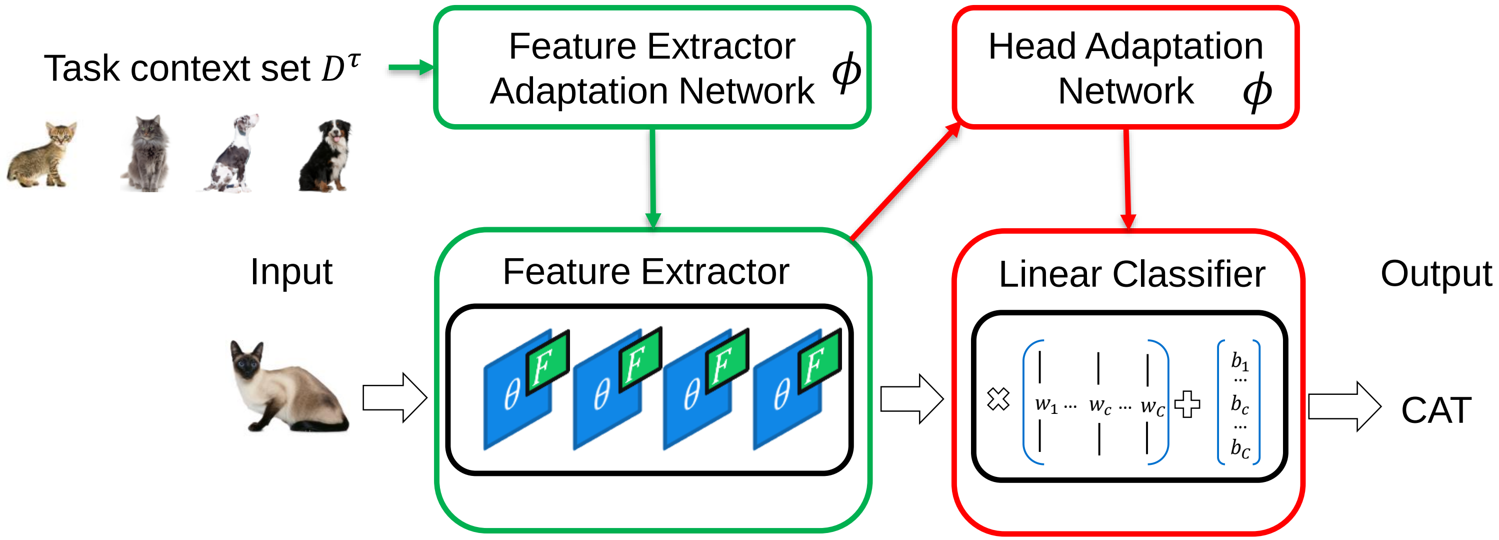
- Pass context set through feature extractor class by class.
- Features for each class c are averaged (*a la* PointNet / Deep Sets).
- Head adaptation network generates the weights and biases for class c.



[1] Gordon, J. et al, "Meta-learning probabilistic inference for prediction". In *International Conference on Learning Representations*, 2019.

Classifier Head Adaptation Network

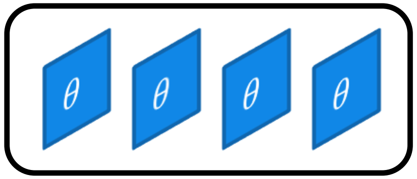
- Pass context set through feature extractor class by class.
- Features for each class c are averaged (*a la* PointNet / Deep Sets).
- Head adaptation network generates the weights and biases for class c.



[1] Gordon, J. et al, “Meta-learning probabilistic inference for prediction”. In *International Conference on Learning Representations*, 2019.

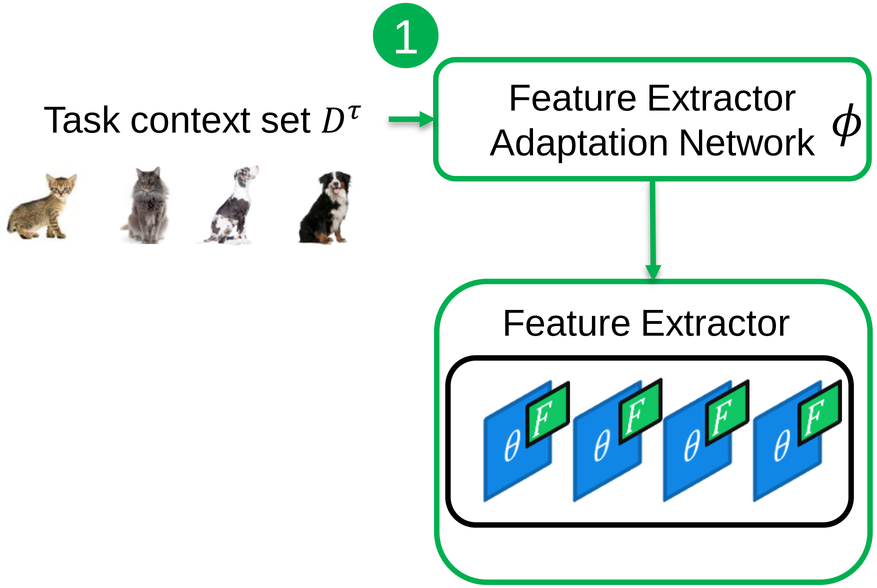
Summary of Adaptation

Feature Extractor



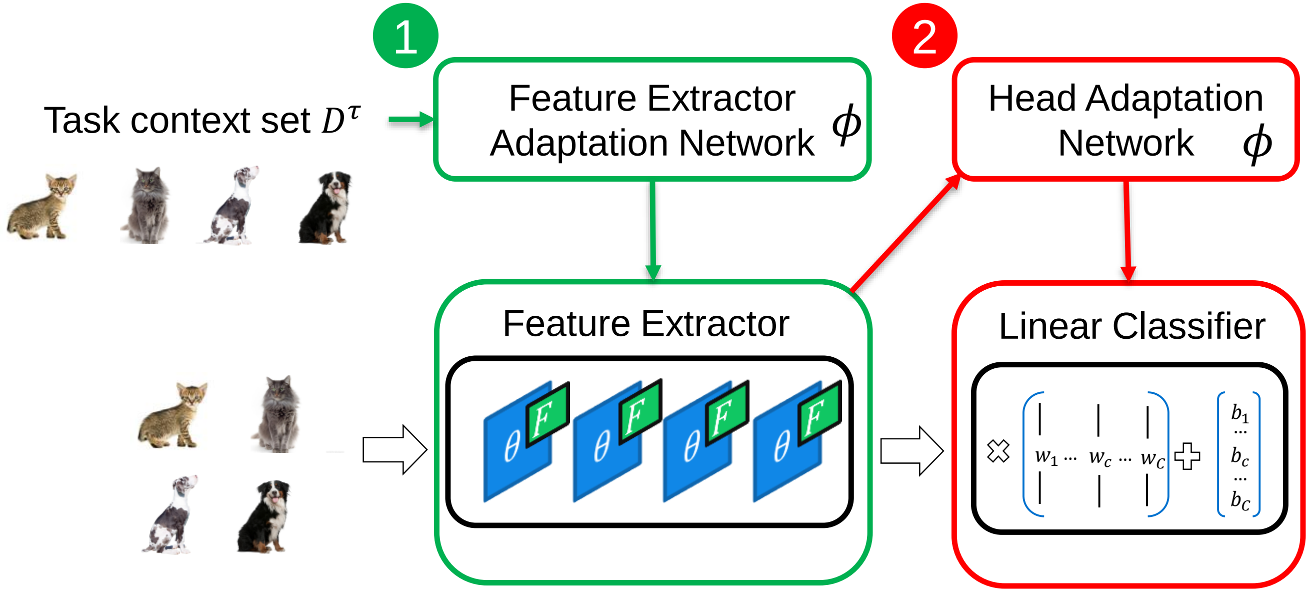
Summary of Adaptation

1. Use context set to adapt feature extractor FiLM layers



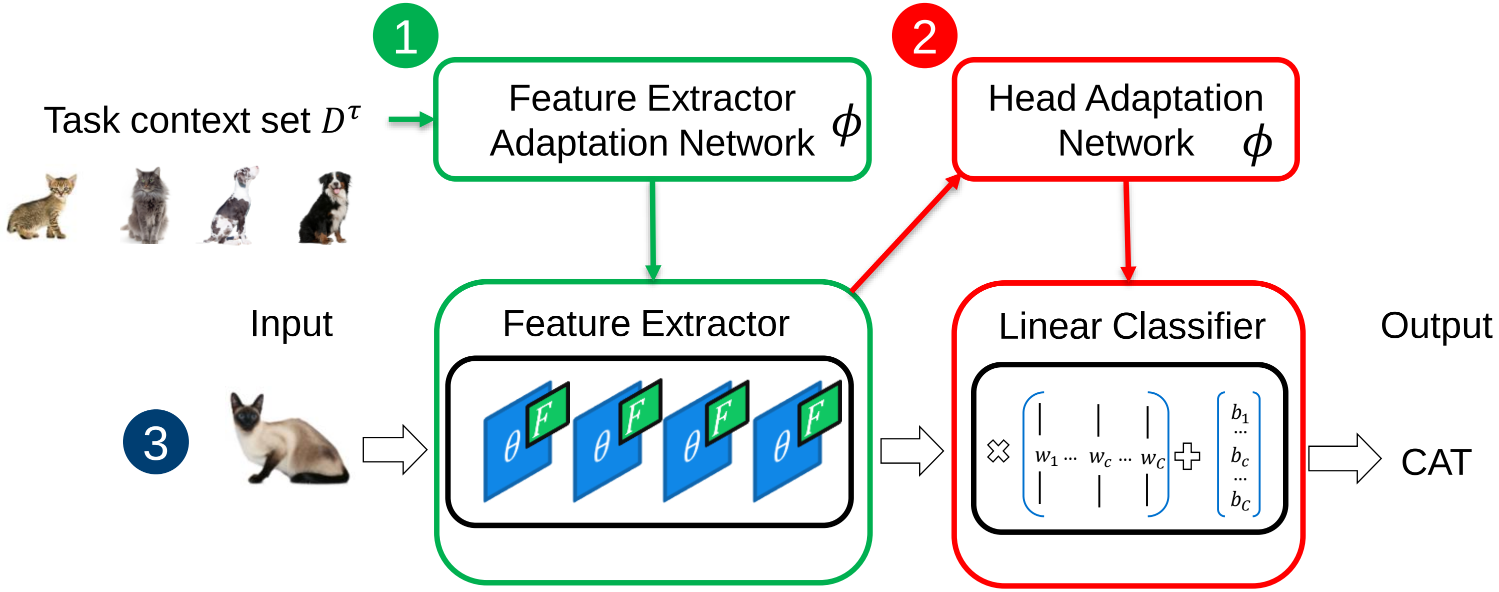
Summary of Adaptation

1. Use context set to adapt feature extractor FiLM layers
2. Pass context set class by class through adapted feature extractor and generate head



Summary of Adaptation

1. Use context set to adapt feature extractor FiLM layers.
2. Pass context set class by class through adapted feature extractor and generate head.
3. Apply adapted network to test examples.



Training θ and ϕ

Training θ and ϕ

1. Pretraining θ
   - Feature extractor parameters θ are pre-trained on a large dataset (e.g. ImageNet) in a full-way classification procedure and then *frozen*.

Training θ and ϕ

1. Pretraining θ
    - Feature extractor parameters θ are pre-trained on a large dataset (e.g. ImageNet) in a full-way classification procedure and then *frozen*.
2. Meta-training ϕ
    - The adaptation parameters ϕ with are trained via maximum likelihood:

Training θ and ϕ

1. Pretraining θ
    - Feature extractor parameters θ are pre-trained on a large dataset (e.g. ImageNet) in a full-way classification procedure and then *frozen*.
2. Meta-training ϕ
    - The adaptation parameters ϕ with are trained via maximum likelihood:

① Sample task $\tau \sim U$.

Training θ and ϕ

1. Pretraining θ
   - Feature extractor parameters θ are pre-trained on a large dataset (e.g. ImageNet) in a full-way classification procedure and then *frozen*.
2. Meta-training ϕ
   - The adaptation parameters ϕ with are trained via maximum likelihood:

① Sample task $\tau \sim U$.

② Randomly split data into D^{τ}(context) and $\{(x_m^{*\tau}, y_m^{*\tau})\}_{m=1}^{M_\tau}$(target).

Training θ and ϕ

1. Pretraining θ
   - Feature extractor parameters θ are pre-trained on a large dataset (e.g. ImageNet) in a full-way classification procedure and then *frozen*.
2. Meta-training ϕ
   - The adaptation parameters ϕ with are trained via maximum likelihood:

① Sample task $\tau \sim U$.

② Randomly split data into D^τ(context) and $\{(x_m^{*\tau}, y_m^{*\tau})\}_{m=1}^{M_\tau}$(target).

③ Evaluate $L^\tau = \frac{1}{M_\tau}\sum_{m=1}^{M_\tau} \log p(y_m^{*\tau} | x_m^{*\tau}, \theta, \psi_\phi(D^\tau, \theta))$.

Training θ and ϕ

1. Pretraining θ
   - Feature extractor parameters θ are pre-trained on a large dataset (e.g. ImageNet) in a full-way classification procedure and then *frozen*.
2. Meta-training ϕ
   - The adaptation parameters ϕ with are trained via maximum likelihood:

① Sample task $\tau \sim U$.

② Randomly split data into D^τ(context) and $\{(x_m^{*\tau}, y_m^{*\tau})\}_{m=1}^{M_\tau}$(target).

③ Evaluate $L^\tau = \frac{1}{M_\tau} \sum_{m=1}^{M_\tau} \log p(y_m^{*\tau} | x_m^{*\tau}, \theta, \psi_\phi(D^\tau, \theta))$.

④ Update $\phi_{new} = \phi_{old} + \eta \nabla_\phi L^\tau$ with learning rate η.

Training θ and ϕ (con't)

- Freezing θ is essential for good feature extractor adaptation.

Training θ and ϕ (con't)

- Freezing θ is essential for good feature extractor adaptation.

  a. The small number of generated FiLM-layer parameters cannot *"compete"* with the large number of parameters θ when adapting to a new context set.

Training θ and ϕ (con't)

- Freezing θ is essential for good feature extractor adaptation.

  a. The small number of generated FiLM-layer parameters cannot *"compete"* with the large number of parameters θ when adapting to a new context set.

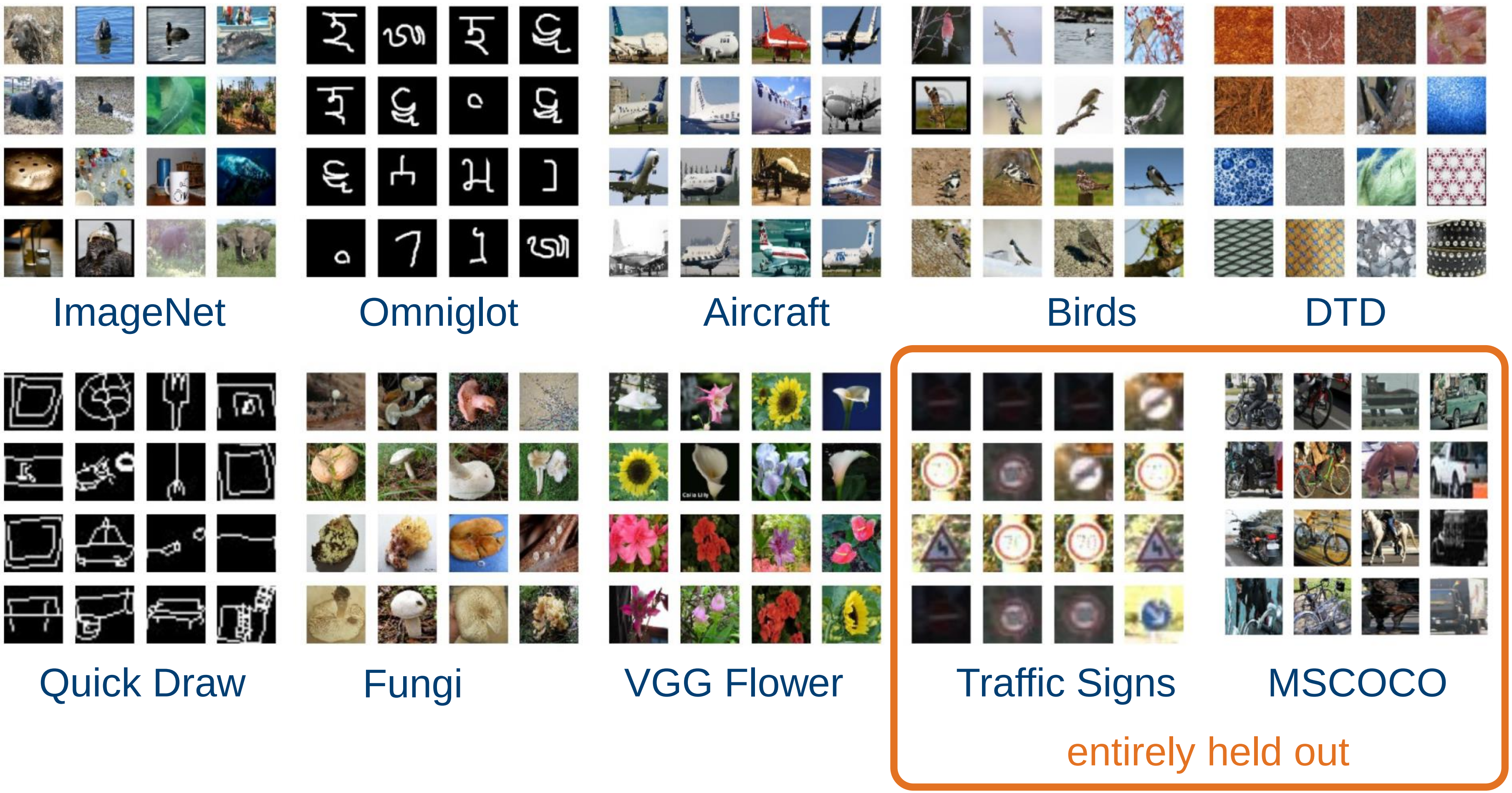
  b. We want to "train as we test", and only the FiLM parameters will update during test.

Meta-Dataset[1] Multi-task, Few-shot Benchmark

UNIVERSITY OF CAMBRIDGE

[1] Triantafillou, Eleni, et al. "Meta-dataset: A dataset of datasets for learning to learn from few examples.“ *arXiv preprint arXiv:1903.03096* (2019).

Meta-Dataset[1] Multi-task, Few-shot Benchmark



[1] Triantafillou, Eleni, et al. "Meta-dataset: A dataset of datasets for learning to learn from few examples." *arXiv preprint arXiv:1903.03096* (2019).

Meta-Dataset Task Sampling

- Task (τ) sampling procedure:
  - Choose a dataset uniformly at random.
  - Choose a random number of classes (between 5 and 50) from the dataset.
  - Choose a random number of shots per class (minimum of 1, maximum of 500 across all classes).
  - Dataset hierarchy is respected in Omniglot and ImageNet
- Adaptation networks are trained on 100,000 of these tasks.

Meta-Dataset Few-Shot Classification Results

Dataset	Finetune	MatchingNet	ProtoNet	fo-MAML	Proto-MAML	CNAPs
ILSVRC [21]	43.1 ± 1.1	36.1 ± 1.0	44.5 ± 1.1	32.4 ± 1.0	47.9 ± 1.1	**52.3 ± 1.0**
Omniglot [31]	71.1 ± 1.4	78.3 ± 1.0	79.6 ± 1.1	71.9 ± 1.2	82.9 ± 0.9	**88.4 ± 0.7**
Aircraft [32]	72.0 ± 1.1	69.2 ± 1.0	71.1 ± 0.9	52.8 ± 0.9	74.2 ± 0.8	**80.5 ± 0.6**
Birds [33]	59.8 ± 1.2	56.4 ± 1.0	67.0 ± 1.0	47.2 ± 1.1	70.0 ± 1.0	**72.2 ± 0.9**
Textures [34]	**69.1 ± 0.9**	61.8 ± 0.7	65.2 ± 0.8	56.7 ± 0.7	67.9 ± 0.8	58.3 ± 0.7
Quick Draw [35]	47.0 ± 1.2	60.8 ± 1.0	64.9 ± 0.9	50.5 ± 1.2	66.6 ± 0.9	**72.5 ± 0.8**
Fungi [36]	38.2 ± 1.0	33.7 ± 1.0	40.3 ± 1.1	21.0 ± 1.0	42.0 ± 1.1	**47.4 ± 1.0**
VGG Flower [37]	85.3 ± 0.7	81.9 ± 0.7	86.9 ± 0.7	70.9 ± 1.0	**88.5 ± 0.7**	86.0 ± 0.5
Traffic Signs [38]	**66.7 ± 1.2**	55.6 ± 1.1	46.5 ± 1.0	34.2 ± 1.3	52.3 ± 1.1	60.2 ± 0.9
MSCOCO [39]	35.2 ± 1.1	28.8 ± 1.0	39.9 ± 1.1	24.1 ± 1.1	**41.3 ± 1.0**	**42.6 ± 1.1**
MNIST [29]						**92.7 ± 0.4**
CIFAR10 [30]						**61.5 ± 0.7**
CIFAR100 [30]						**50.1 ± 1.0**

held out classes

entirely held out

FiLM Parameter Learning: SGD vs Adaptation Nets

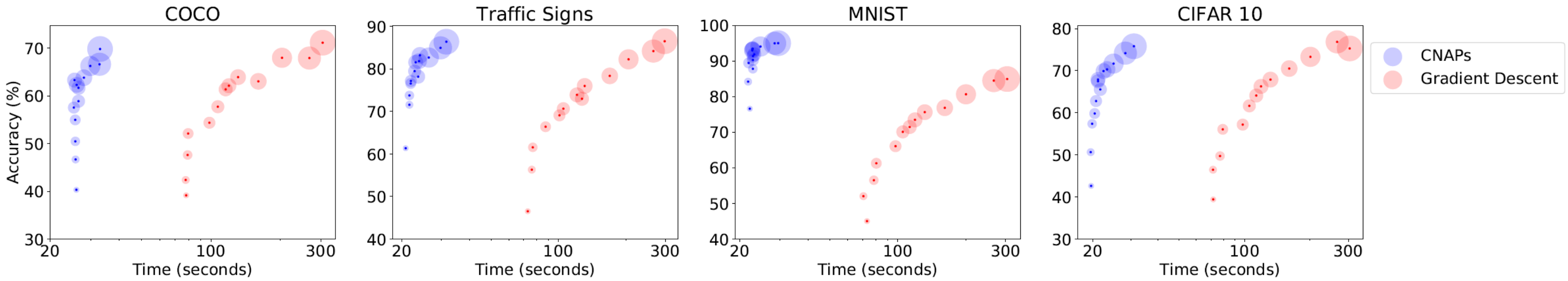


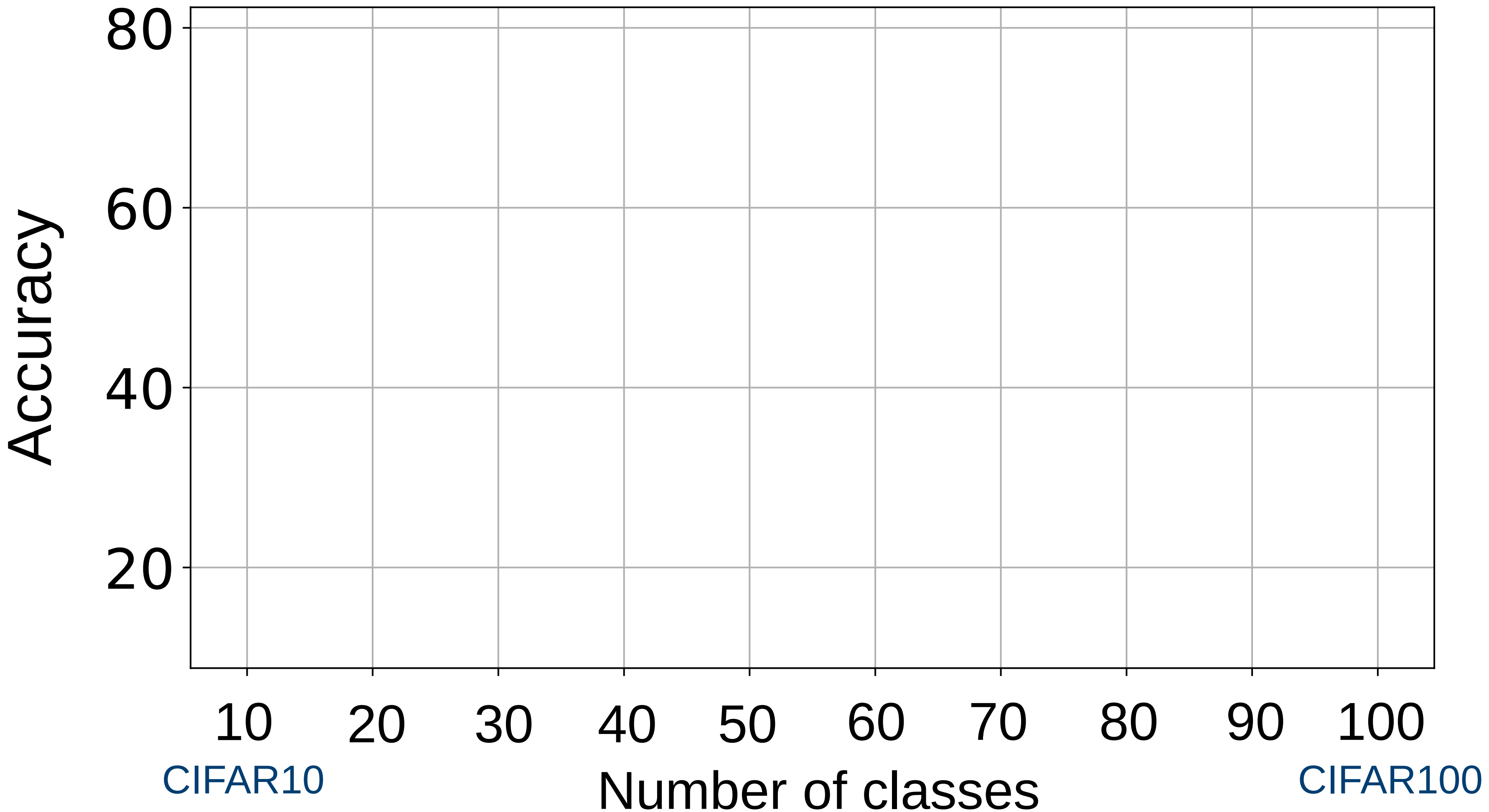
Figure 5: Comparing CNAPs to gradient based feature extractor adaptation: accuracy on 5-way classification tasks from withheld datasets as a function of processing time. Dot size reflects shot number (1 to 25 shots).

- Given sufficient data and processing time, SGD will yield higher accuracy, but:
  - CNAPs is at least 5x faster at test time as it only requires a *single forward pass* through the network while SGD needs multiple forward + backward passes.
  - CNAPs has higher accuracy at low shot, as it is resistant to overfitting as a result of the adaptation network parameters being shared and trained on diverse tasks.

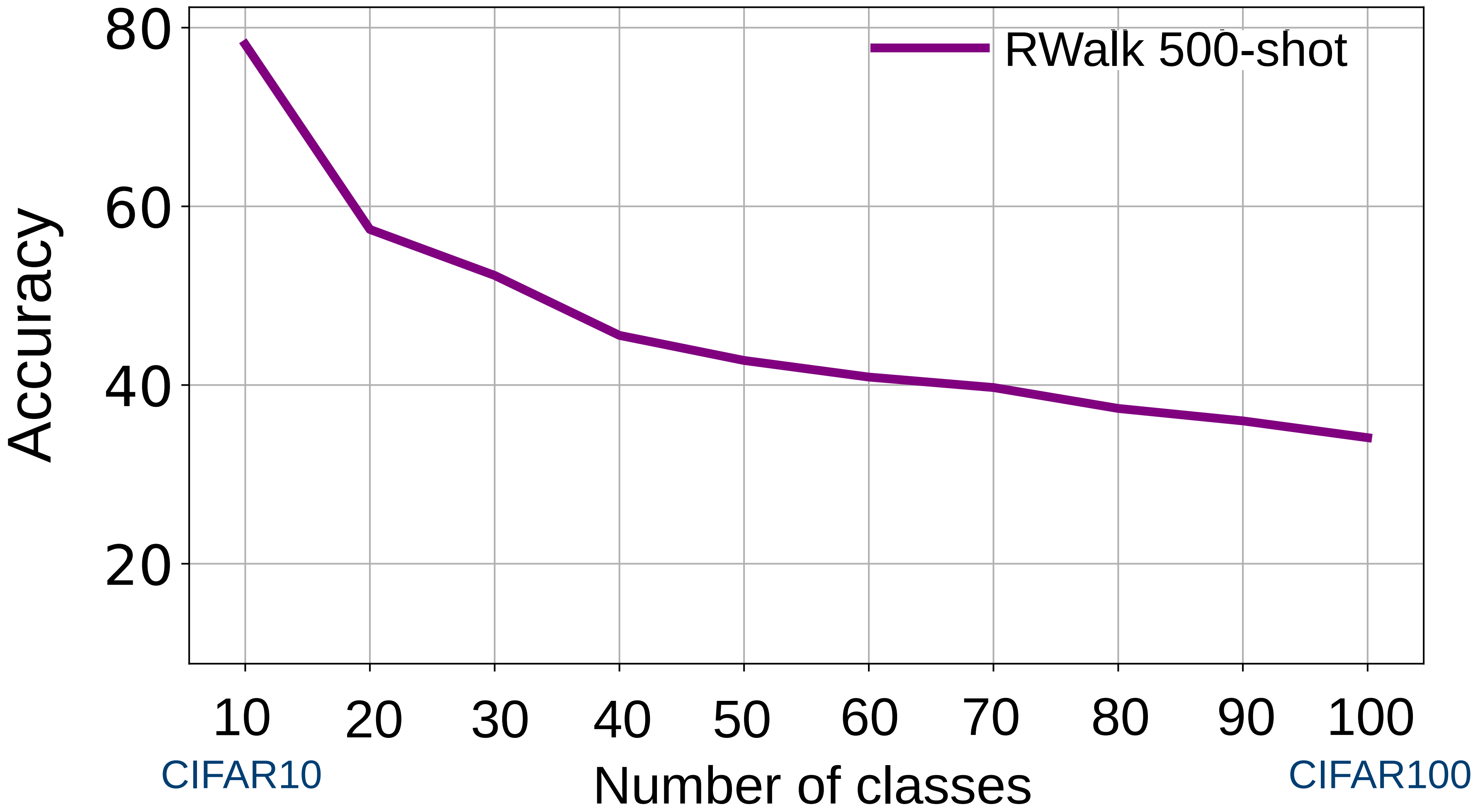
Continual Learning

- Continual (or lifelong) learning involves learning from a sequence of new tasks without forgetting earlier ones.
- We employ the same meta-trained model that was use in the few-shot, multi-task classification experiments.
- Memory:
  - We allow our model to “remember” by storing running averages of the adaptation networks set encodings as new tasks are encountered.
- The following results are an “apples to oranges” comparison:
  - Model has been meta-trained on many datasets and has learned to adapt!

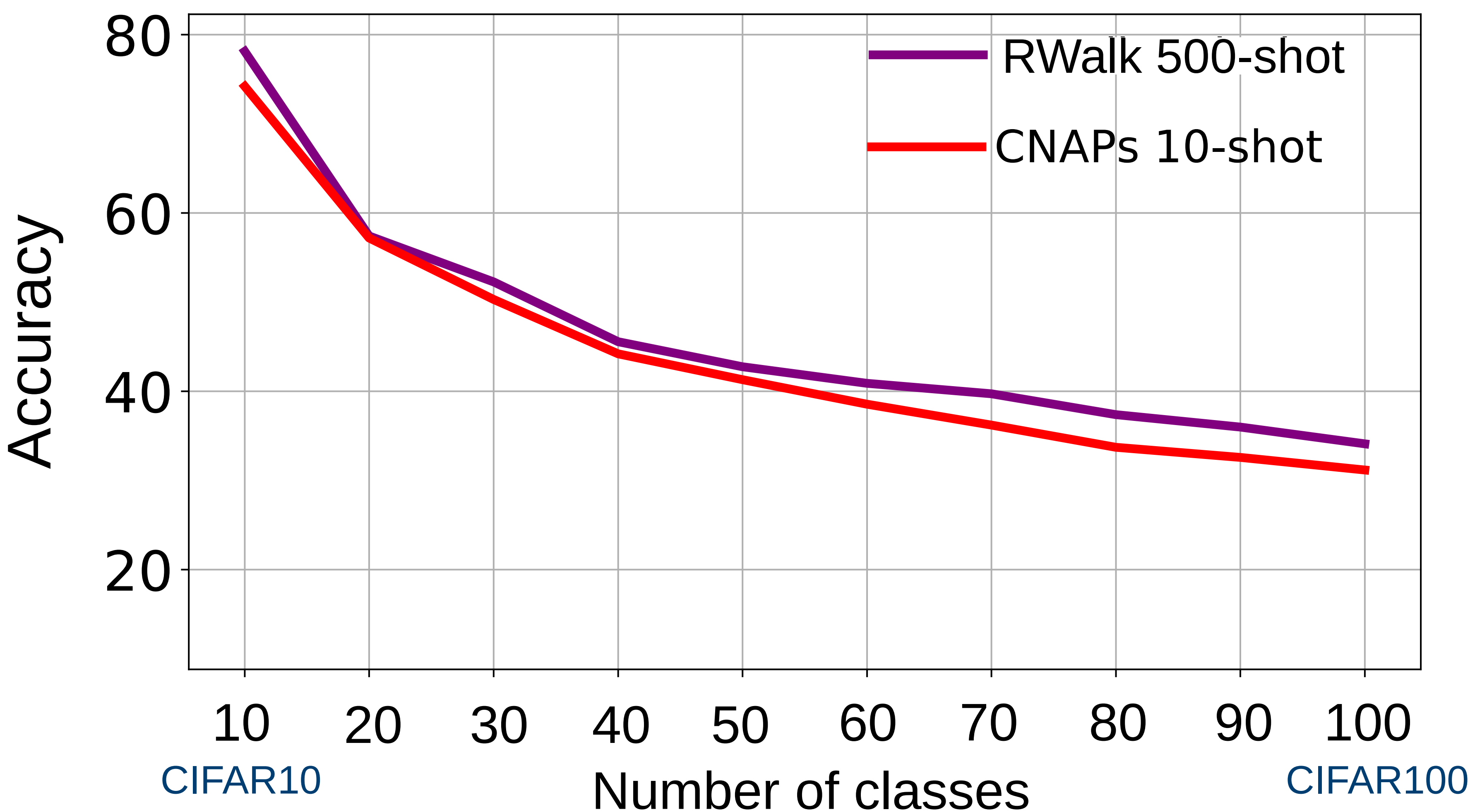
Continual Learning Results: Single head, CIFAR 100



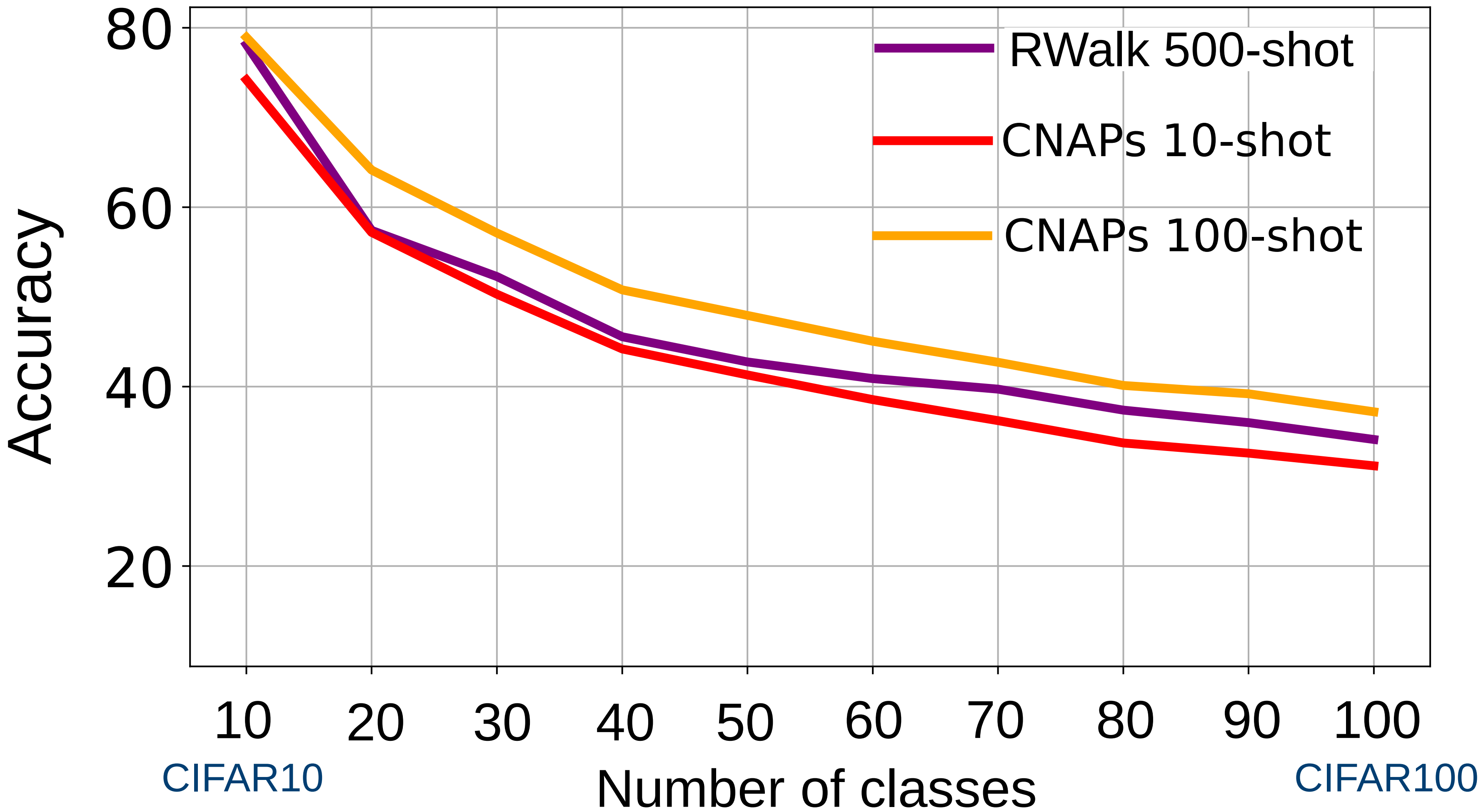
Continual Learning Results: Single head, CIFAR 100



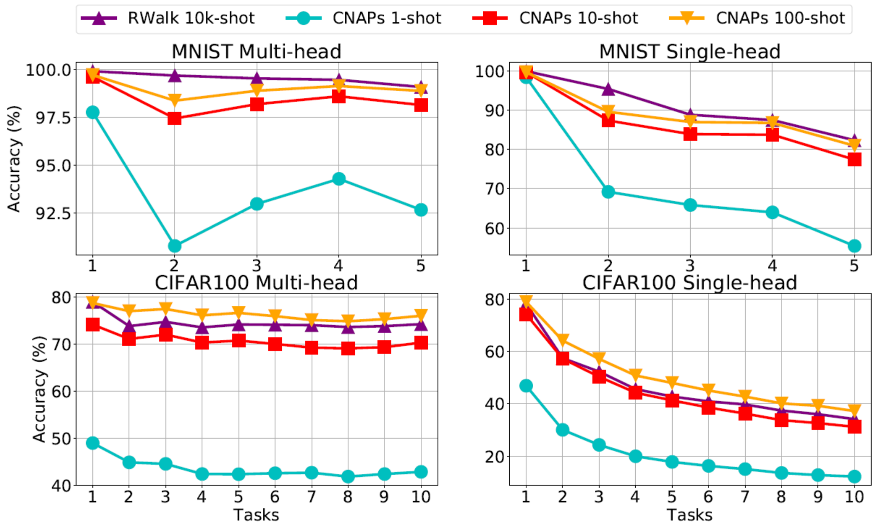
Continual Learning Results: Single head, CIFAR 100



Continual Learning Results: Single head, CIFAR 100



Continual Learning Results



	MNIST		*CIFAR100*	
Method	Multi	Single	Multi	Single
SI [41]	99.3	57.6	73.2	22.8
EWC [43]	99.3	55.8	72.8	23.1
VCL [44]	98.5 $\pm$ 0.4	-	-	-
RWalk [42]	99.3	82.5	74.2	34.0
CNAPs	98.9 $\pm$ 0.2	80.9 $\pm$ 0.9	76.0 $\pm$ 0.5	37.2 $\pm$ 0.6

- Results are competitive with or better than state of the art baselines. Despite:
  - Not being trained to do continual learning.
  - Not exposed to these datasets during meta-training.
  - Observing orders of magnitude fewer examples.

Thanks for listening!

- Any questions?
- Paper: https://arxiv.org/pdf/1906.07697.pdf
- Code: https://github.com/cambridge-mlg/cnaps