Stochastic Deep Networks with Linear Competing Units for Model-Agnostic Meta-Learning

Sotirios Chatzis

Cyprus University of Technology

sotirios.chatzis@cut.ac.cy

Thursday 21st July, 2022

Overview

Stochastic Deep Networks with Linear Competing Units for Model-Agnostic Meta-Learning

- Meta-Learning (ML): Challenge of generalizing learned representations to new tasks, where only limited data is available.
- Model-Agnostic Meta-Learning (MAML): Tunes the parameters of a trained model to learn a new task with few gradient updates.
- *Contribution*: A novel MAML framework with: (i) improved generalization capacity; and (ii) immensely decreased trainable parameters and imposed memory footprint.
- Stochastic Local Competition (LWTA) Units + Stochastic Weights $\rightarrow$ Completely outperform the SOTA in few-shot image classification and regression benchmarks.

Inspiration

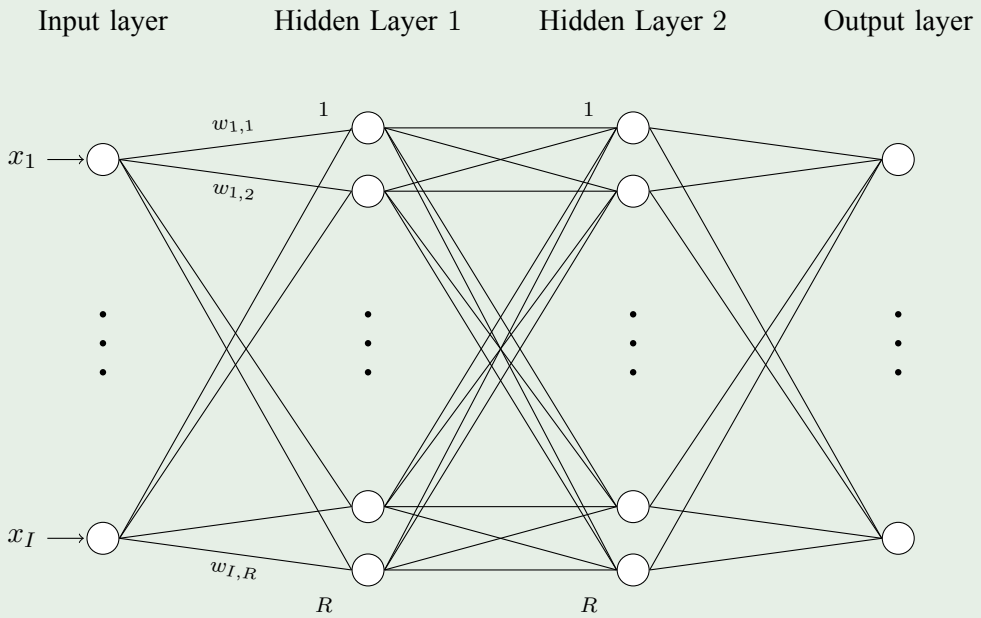
- Brain structure: *Lateral* "connections" among *blocks* of neurons which *compete* for their firing →Local Winner-Takes-All (LWTA).
- It appears that this mechanism is of *stochastic* nature:
  - The same system may produce different firing outcomes when presented with exactly the same stimulus at multiple times.
- *Stochasticity* may be crucial for endowing the learned representations with the capacity to generalize better, within a task and across tasks.

Contribution

- A novel deep network paradigm founded upon the concepts of *Local Competition* and *Stochasticity*.
- *Local Competition* yields a sparse output in each layer; one unit with non-zero output in each block, and the rest zeroed-out.
- This is determined by the outcome of local competition among linear units within each block; multiple blocks per layer.
- Main *Source of Stochasticity*: Competition is stochastic; winner is determined through sampling from an appropriate Categorical posterior.
  - *Stochastic LWTA Layers*.

Conventional Network Architectures

Dense-Layer Model

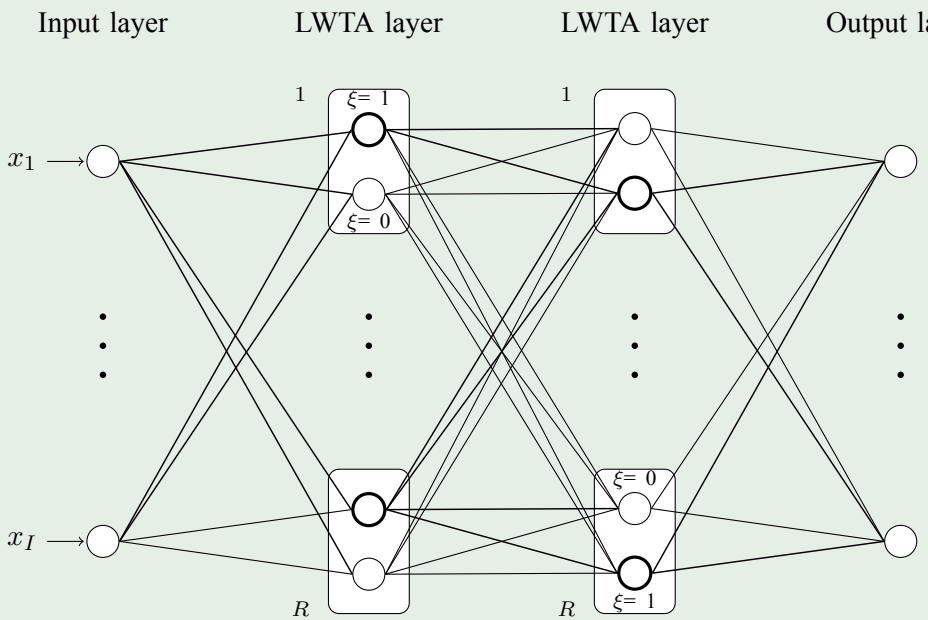


- Observations: $\boldsymbol{x} \in \mathbb{R}^I$.
- R nonlinear units in a considered layer.
- The weights of all synapses: $\boldsymbol{W}: \ w_{i,r}, \ i|_1^I, r|_1^R$.
- Standard inner-product computation: $h_r = \boldsymbol{W}_r^T \boldsymbol{x}, r|_1^R$.
- Then, application of a nonlinear activation $\sigma(\cdot)$, e.g. ReLU.
- Layer Output:

$$\boldsymbol{y} = \sigma(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}) \in \mathbb{R}^R$$

Stochastic LWTA Networks

Dense-Layer Model



- Observations: $\boldsymbol{x} \in \mathbb{R}^I$.
- The weights of all synapses: $\boldsymbol{W}: \; w_{i,r,j}, \; i|_1^I, r|_1^R, j|_1^J$.
- Units compute inner-products: $\sum_{i=1}^{I} w_{i,r,j} \cdot \boldsymbol{x}_i, \forall r, j$.

- We introduce the *winner indicator* latent variables $\boldsymbol{\xi} \in \text{one\_hot}(J)^R$.
- Layer Output: $\boldsymbol{y}$ with $(r \cdot j)$-th component $\boldsymbol{y}_{r,j}$:

$$\boldsymbol{y}_{r,j} = \boldsymbol{\xi}_{r,j} \sum_{i=1}^{I} w_{i,r,j} \cdot x_i \in \mathbb{R}$$

Stochastic LWTA Networks

Priors

- Weights: $\text{vec}(\boldsymbol{W}) \sim N(\boldsymbol{0}, \boldsymbol{I})$ and latent variables: $\boldsymbol{\xi} \sim \text{Categorical}(1/J)$.

Categorical Posterior for Winner Sampling

- We consider a block-wise posterior of the form:

$$q(\boldsymbol{\xi}_r) = \text{Categorical}\left(\boldsymbol{\xi}_r \middle| \text{softmax}(\sum_{i=1}^{I} [w_{i,r,j}]_{j=1}^{J} \cdot \boldsymbol{x}_i)\right)$$

Gaussian Posterior for Weights

- We consider an approximate *(variational)* posterior of the form:

$$q(\text{vec}(\boldsymbol{W})) = N(\text{vec}(\boldsymbol{W})|\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$$

where $\boldsymbol{\psi} \triangleq \{\boldsymbol{\mu}, \boldsymbol{\sigma}^2\}$ are the means and variances of the Gaussian weight posteriors, respectively.

Training

- Let $\mathrm{CE}(Y_i, f_{\boldsymbol{\psi}}(X_i; \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{W}}))$ be the categorical cross-entropy between data labels Y_i and class probabilities $f_{\boldsymbol{\psi}}(X_i; \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{W}})$ obtained by the network.
- We resort to ELBO maximization for training; the ELBO yields the expression:

$$L_{T_i}(\boldsymbol{\psi}) = -\mathrm{CE}(Y_i, f_{\boldsymbol{\psi}}(X_i; \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{W}})) - \sum_{r,j} (\log q(\hat{\boldsymbol{\xi}}_{r,j}) - \log p(\hat{\boldsymbol{\xi}}_{r,j})) - \sum_{t,r,j} (\log q(\hat{w}_{t,r,j}) - \log p(\hat{w}_{t,r,j}))$$

- We use the following reparameterization tricks: $\hat{w}_{t,r,j} = \mu_{t,r,j} + \sigma_{t,r,j}\hat{\epsilon}$, with $\hat{\epsilon} \sim N(0,1)$; and:

$$\hat{\boldsymbol{\xi}}_{r,j} = \mathrm{softmax}((\log \bar{\boldsymbol{\xi}}_{r,j} + g_{r,j})/\tau)$$

where $\bar{\boldsymbol{\xi}}_{r,j} \triangleq q(\boldsymbol{\xi}_{r,j})$, $g_{r,j} = -\log(-\log \mathcal{U}(0,1))$, and τ regulates relaxation.

Stochastic LWTA Networks

Inference

- We draw a set of B samples of the Gaussian connection weights from the trained posteriors $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$.
- In addition, we select the winning units in each block of the network by similarly sampling from the posteriors $q(\boldsymbol{\xi})$.
- This results in a set of B output logits from the network, which we average to obtain the final predictive outcome:

$$f_{\boldsymbol{\psi}}(X'; \tilde{\boldsymbol{\xi}}, \tilde{\boldsymbol{W}}) \approx \frac{1}{B} \sum_{s=1}^{B} f_{\boldsymbol{\psi}}(X'; \tilde{\boldsymbol{\xi}}_s, \tilde{\boldsymbol{W}}_s)$$

  where $\tilde{\boldsymbol{\xi}}_s$ and $\tilde{\boldsymbol{W}}_s$ are sampled directly from the posteriors $q(\boldsymbol{\xi})$ and $q(\boldsymbol{W})$, respectively.
- $B = 4$ drawn samples are sufficient (favorable accuracy/complexity trade-off).

Training

1: **Require:** $P(T)$: distribution over tasks
2: Initialize $\boldsymbol{\psi} := \{\boldsymbol{\mu}, \boldsymbol{\sigma}^2\}$
3: Define outer-step size β and inner learning rate α
4: **for** i = 1,2, . . . **do**
5: **Inner training loop**:
6: Sample task $T_i \sim P(T)$
7: Use the ELBO to compute loss $L_{T_i}(\boldsymbol{\psi})$
8: Compute adapted parameters with SGD: $\boldsymbol{\psi}_i' = \boldsymbol{\psi} - \alpha \nabla_{\boldsymbol{\psi}} L_{T_i}(f_{\boldsymbol{\psi}})$
9: **Outer training loop**:
10: Derive $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} + \beta(\boldsymbol{\psi}_i' - \boldsymbol{\psi})$
11: **end for**

- The stochastic nature of the weights, $\boldsymbol{W}$, results in the updates taking place over the posterior means, $\boldsymbol{\mu}$ and variances, $\boldsymbol{\sigma}^2$.
- The stochastic nature of both the representations (stemming from $\boldsymbol{\xi}$), and the network weights themselves, implies that proper training must rely on optimization of the ELBO function of the network.

Classification experiments

Defining a new SOTA paradigm

- Networks comprising 2 layers.
- 16 blocks and 2 competing units per block on the former layer.
- 8 blocks with 2 neurons per block on the latter.

	Omniglot 20-way		Mini-ImageNet 5-way	
Algorithm	1-shot	5-shot	1-shot	5-shot
Matching Nets	93.8	98.50	43.56	55.31
LSTM Meta-Learner	-	-	43.44	60.60
MAML (original)	95.80	98.90	48.70	63.11
MAML (locally-reproduced)	95.47	98.60	48.57	62.87
Reptile (original)	88.14	96.65	47.07	62.74
Reptile (locally-reproduced)	87.98	96.35	46.83	62.40
ABML (locally-reproduced)	90.21	93.39	44.23	52.12
BMAML (locally-reproduced)	96.92	98.11	53.10	64.80
PLATIPUS (locally-reproduced)	94.35	98.30	49.97	63.13
StochLWTA-ML	**97.79**	**98.97**	**54.11**	**66.70**

Is there a computational time trade-off for the increased accuracy?

Performance comparison: Average wall-clock time (in msecs), training iterations and number of baselines' trainable parameters

- Our methodology takes 77% less training time than the less efficient algorithms ABML, BMAML, PLATIPUS, and is comparable to other approaches.
- Contains a total number of trainable parameters that is *one order of magnitude less* on average than the best performing baseline methods.

Algorithm	Training	Prediction	Training iterations	D_A parameters	D_B parameters	D_C parameters
PLATIPUS (local)	1603.39	602.77	333600	560025	615395	580440
BMAML (local)	1450.31	514.43	301800	560025	615395	580440
ABML (local)	678.48	265.78	138000	224010	246158	232176
MAML (local)	288.25	103.28	60000	112005	123079	116088
FOMAML (local)	284.49	102.34	60000	112005	123079	116088
Reptile (local)	284.30	**102.27**	60000	113221	124613	117463
StochLWTA-ML	**282.90**	113.44	60000	**54549**	**60112**	**56745**

How does stochastic competition contribute to classification accuracy?

Ablation Study: Few-shot classification accuracy (%)

- **Replacing** ReLU with **deterministic** LWTA yields negligible gains.
- **Stochastic** LWTA units yield a clear improvement in all cases.

		Omniglot 20-way		**Mini-Imagenet 5-way**	
Algorithm	**Network type**	1-shot	5-shot	1-shot	5-shot
MAML (local)	deterministic LWTA	95.52	98.15	48.88	63.15
	stochastic LWTA	95.91	98.78	49.61	64.03
FOMAML (local)	deterministic LWTA	95.01	98.18	48.11	63.54
	stochastic LWTA	95.80	98.41	49.24	64.54
ABML (local)	deterministic LWTA	90.30	93.64	44.31	52.27
	stochastic LWTA	91.21	93.91	45.11	53.31
BMAML (local)	deterministic LWTA	96.96	98.21	53.12	64.84
	stochastic LWTA	97.11	98.30	53.50	65.31
PLATIPUS (local)	deterministic LWTA	94.48	98.31	49.99	63.21
	stochastic LWTA	95.13	98.56	51.06	64.18
StochLWTA-ML	deterministic LWTA	96.95	98.63	53.12	64.93
	stochastic LWTA	**97.79**	**98.97**	**54.11**	**66.70**

Effect of block size J in predictive (%) accuracy.

Few-shot classification benchmarks

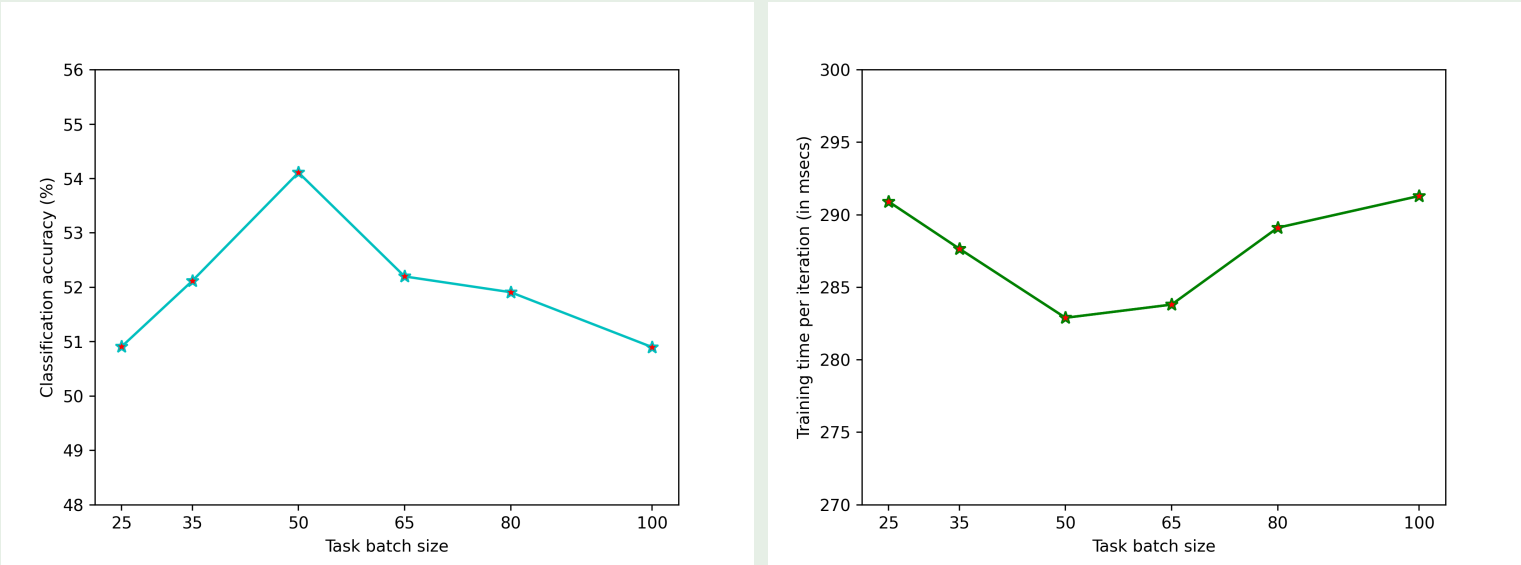
- Increasing the number of competing units per block to $J = 4$ or $J = 8$ does not notably improve the results of our approach.
- On the contrary, it increases the number of trained parameters, thus leading to higher network computational complexity.

	Omniglot 20-way		**Mini-Imagenet 5-way**		**CIFAR-100 5-way**	
Number of units	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
$J = 2$	97.79	98.97	54.11	66.70	54.60	66.73
$J = 4$	96.33	98.55	53.99	66.65	54.51	66.13
$J = 8$	95.38	98.83	53.70	67.08	54.45	66.18

How does task batch size affect the performance?

Mini-Imagenet 5-way 1-shot setting

- Our model performs optimally with task batch size of 50 in terms of both training time and predictive accuracy.
- We have obtained similar results for all other considered datasets.



(a) (b)

Figure: (a) Predictive accuracy (b) Training time per iteration (msecs)

How does sample size B at prediction time affect predictive accuracy?

Few-shot classification benchmarks

- An increase in sample size, B, does not always yield an accuracy increase.
- $B = 4$ allows for the best predictive accuracy/computational complexity trade-off

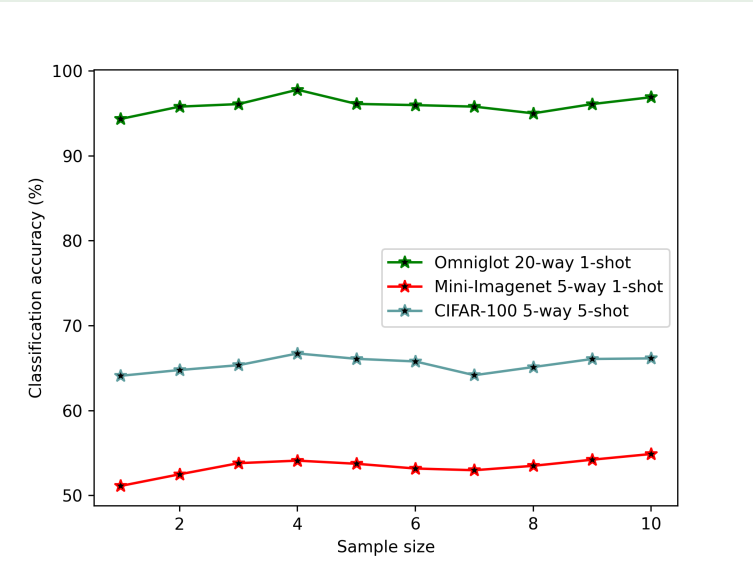


Figure: Predictive (%) accuracy

How important are the stochastic weights?

Omniglot 20-way ablation study

- Stochastic LWTA units offer the greatest fraction of the accuracy gains, but the Gaussian weights are still indispensable.

Gausssian vs deterministic weights (point estimates).

Network type	**1-shot**	**5-shot**
ReLU (point estimates)	95.63	96.17
ReLU (Gaussians)	95.80	96.48
stochastic LWTA (point estimates)	97.68	98.85
stochastic LWTA (Gaussians)	**97.79**	**98.97**

How do SOTA perform with a parameter count reduced to be about the same as StochLWTA-ML?

Omniglot 20-way 1-shot setting

- Replaced stochastic LWTA layers with dense ReLU layers of the same size, and dropped the Gaussians from the weights.
- This yielded between 2.2% and 3.5% reduction in classification accuracy.

Wall-clock time (in msecs), training iterations, predictive accuracy and trainable parameters.

Algorithm	Training	Prediction	Training iterations	Accuracy (%)	Parameters
PLATIPUS (local)	490.67	221.91	107100	91.57	55817
BMAML (local)	479.03	200.68	103560	94.11	56321
ABML (local)	402.38	170.32	87000	87.92	55312
MAML (local)	272.89	91.24	60000	92.10	55917
FOMAML (local)	269.01	91.12	60000	92.83	55917
Reptile (local)	**268.72**	**90.83**	60000	85.60	56525
StochLWTA-ML	272.14	102.56	60000	**97.79**	**54549**

Regression experiments

Sinusoidal regression results

- (a): Equally fast convergence as MAML, Reptile; convergence speed is much higher than the time-consuming ABML, BMAML and PLATIPUS methods.
- (b): Yields the most time-efficient method among the baselines.

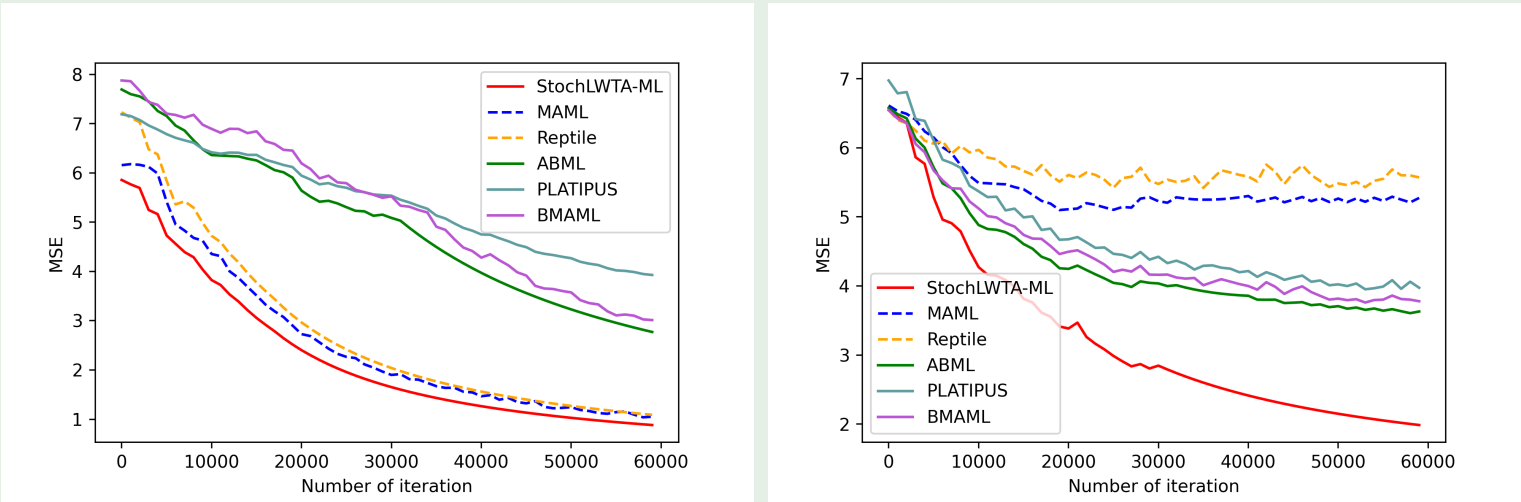


Figure: (a) MSE of default setting [Finn et al., 2017], (b) MSE of challenging setting [Yoon et al., 2018].

Thank you!