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

Konstantinos Kalais [1] **Sotirios Chatzis** [1]

Abstract

This work addresses meta-learning (ML) by considering deep networks with stochastic local winner-takes-all (LWTA) activations. This type of network units results in sparse representations from each model layer, as the units are organized into blocks where only one unit generates a non-zero output. The main operating principle of the introduced units rely on stochastic principles, as the network performs posterior sampling over competing units to select the winner. Therefore, the proposed networks are explicitly designed to extract input data representations of sparse stochastic nature, as opposed to the currently standard deterministic representation paradigm. Our approach produces state-of-the-art predictive accuracy on few-shot image classification and regression experiments, as well as reduced predictive error on an active learning setting; these improvements come with an immensely reduced computational cost. Code is available at: https://github.com/Kkalais/StochLWTA-ML

1. Introduction

When we train machine learning models on problems with limited amounts of training data, we cannot usually get good predictive performance (Sculley et al., 2015; Lai, 2019). This comes in contrast to the human ability to quickly derive information from a range of different tasks, and then adapt to a new task with limited available new examples (Castro et al., 2008; Kühl et al., 2020). In essence, this capability of the mind of learning how to learn (Black et al., 2006) has inspired researchers to investigate the concept of Meta-Learning (ML) (Lake et al., 2017; Buysse et al., 2019).

There is a large variety of deep learning methods for ML (Andrychowicz et al., 2016; Finn et al., 2019; Baik et al., 2020). Specifically, Finn et al. (2017) presented the *Model-Agnostic Meta-Learning* (MAML) algorithm that enables tuning the parameters of a trained network to quickly learn a new task with only a few gradient updates. To get away with the entailed second-order computations, that are expensive, several researchers proposed appropriate first-order approximations for MAML; such works are the *First-Order MAML* (FOMAML) of Finn et al. (2017) and the *Reptile* algorithm of Nichol et al. (2018). Recently, several researchers have also considered Bayesian inference-driven methods for deep learning ML, extending upon older works built for conventional machine learning approaches (Grant et al., 2018; Yoon et al., 2018; Finn et al., 2018; Ravi & Beatson, 2019; Patacchiola et al., 2020; Zou & Lu, 2020; Chen et al., 2020).

This work proposes a different regard toward improving generalization capacity for deep networks in the context of ML. Specifically, our proposed approach relies on the following main concepts:

- **The concept of sparse learned representations.** For the first time in the literature of deep network-driven ML, we employ a mechanism that inherently learns to extract sparse data representations. This consists in replacing standard unit nonlinearities (e.g., ReLU) with a unit competition mechanism. Specifically, (linear) units are organized into blocks. Presented with some input, the units within a block engage in a competition process with only one winner. The outputs of all units except for the winner are zeroed out; the output of the winner retains its computed value (local winner takes-all, LWTA, architecture).
- **The concept of stochastic representations.** We establish a stochastic formulation for the previously described competition process. Specifically, we postulate that, within a block of competing units, winner is selected via sampling from an appropriate Categorical posterior. The corresponding winning probability of each unit is proportional to its linear computation (thus depending on the layer input). Via this competition

[1]Dept. of Electrical Eng., Computer Eng., and Informatics, Cyprus University of Technology, Limassol, Cyprus. Correspondence to: Konstantinos Kalais <ki.kalais@cut.ac.cy>, Sotirios Chatzis <sotirios.chatzis@cut.ac.cy>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

process, we yield stochastic representations from network layers, that is representations that may change each time we present to the network layer exactly the same input.

Based on the results from existing approaches, we posit that the proposed treatment of the ML problem, which combines learned representation sparsity and stochasticity, will be extremely beneficial to the deep learning community. We dub our approach Stochastic LWTA for ML (StochLWTA-ML).

We perform a variational Bayes treatment of the proposed model. We opt for a full Bayesian treatment, by also handling network weights as latent variables. That is, we elect to impose an appropriate prior over the network weights and fit approximate (variational) posteriors. As we evaluate our approach on image classification, sinusoidal regression and active learning problems, we show that StochLWTA-ML offers a variety of advantages over the current state-of-the-art methods, namely: (i) incurring reduced predictive error rate compared to the currently state-of-the-art methods in the field; (ii) obtaining this performance with networks that comprise *an immensely reduced* number of trainable parameters, and therefore give rise to better computational efficiency and imposed memory footprint.

The remainder of this paper is organized as follows: In Section 2, we briefly review related work. Section 3 introduces our approach and provides the related training and prediction algorithms. In Section 4, we perform a thorough experimental evaluation of StochLWTA-ML, and compare our findings to the current state-of-the-art. In the final Section 5, we end up with the conclusions of our work, and suggest lines of further research.

2. Related Work

2.1. LWTA layers in Deep Learning

LWTA layers are not new in the field of deep learning; see, e.g., Srivastava et al. (2013). Although not much work has been pursued along these lines, the recent works of Panousis et al. (2019; 2021) and Voskou et al. (2021) have spurred some fresh interest in the field. These works have presented alternative implementations of the basic concepts of LWTA units in the context of diverse deep network architectures. Specifically, Panousis et al. (2019) propose a stochastic LWTA formulation which is founded upon the Indian Buffet process (IBP) prior, borrowed from nonparametric statistics; they use this architecture to effect data-driven network compression. Panousis et al. (2021) exploit the same technique to train adversarially-robust deep networks. On the other hand, Voskou et al. (2021) consider a different incarnation of stochastic LWTA architectures, which relies on sampling the winner from a Categorical posterior, driven from the layer input. This layer architecture is used to replace dense ReLU layers in Transformer networks; it is then shown to yield important benefits in a Sign-Language Translation benchmark.

This paper is different from the previous works in various ways: (i) stochasticity does not stem from utilization of the IBP; we rather adopt an approach similar to Voskou et al. (2021); (ii) we do not use the proposed architecture as a replacement for a specific type of layer in a greater network architecture (Transformer) that remains largely unchanged; instead, we build completely new networks using these layers; (iii) we perform a full Bayesian treatment, by treating network weights as random variables; we do not employ this construction as a means of compressing the weights at prediction time, contrary to Panousis et al. (2019), Voskou et al. (2021); instead, we perform weight sampling at prediction time as a means of improving accuracy; and (iv) for the first time, we examine how these principles perform in the context of deep network-driven ML. Note that, apart from LWTA architectures, other data-driven sparsity models have also been proposed recently, e.g. Lee et al. (2018), Kessler et al. (2021). However, none of these have been developed or evaluated in the context of an ML setting.

2.2. Model-Agnostic Meta-Learning

Finn et al. (2017) suggested a *model-agnostic* algorithm for ML, that can be applied to any model trained via gradient descent. The introduced MAML algorithm initializes model parameters in a way that can be quickly adapted to several types of new tasks.

Let us consider a model with parameters $\boldsymbol{\theta}$ and a parametric form $f_{\boldsymbol{\theta}}$. When the model is adapting to an unseen task T_i, MAML runs few steps of gradient descent that yields a task-specific parameter set, $\boldsymbol{\theta}_i' = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L_{T_i}(f_{\boldsymbol{\theta}})$; α is the step size hyperparameter and L_{T_i} denotes the loss on the task T_i. Subsequently, training proceeds to optimize the function $f_{\boldsymbol{\theta}_i'}$ with respect to the model parameters $\boldsymbol{\theta}$. Assuming that the batch of tasks has size M, we can define the targeted *meta-objective* as:

$$L_{meta}(\boldsymbol{\theta}) = \sum_{i=1}^{M} L_{T_i}(f_{\boldsymbol{\theta}_i'}) = \sum_{i=1}^{M} L_{T_i}(f_{\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} L_{T_i}(f_{\boldsymbol{\theta}})}). \tag{1}$$

Optimization of this meta-objective over $\boldsymbol{\theta}$ yields the *outer-loop* update:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{M} L_{T_i}(f_{\boldsymbol{\theta}_i'}) \tag{2}$$

where β stands for the *outer-loop* learning rate.

Finally, as *MAML* involves expensive computations stem-

ming from the second-order updates of Eqs. (1) and (2), Finn et al. (2017) and Nichol et al. (2018) have developed more efficient first-order approximations.

3. Proposed Approach

3.1. Architecture

Let us denote as $\boldsymbol{x} \in \mathbb{R}^I$ an input vector presented to a dense ReLU layer of a conventional deep neural network, with corresponding weights matrix $\boldsymbol{W} \in \mathbb{R}^{I \times O}$. The output of the layer is the vector $\boldsymbol{y} \in \mathbb{R}^O$ and is fed to the subsequent layer. In our approach, a ReLU unit is replaced by J competing linear units, organized in one block; in the following, we denote with R the number of blocks in a layer. The input $\boldsymbol{x}$ is now presented to each block through weights that are organized into a three-dimensional matrix $\boldsymbol{W} \in \mathbb{R}^{I \times R \times J}$. Then, the j-th competing unit within r-th block computes the sum $\sum_{i=1}^{I}(w_{i,r,j}) \cdot x_i$. Competition means that, of the J units in the block, one unit (the "winner") will present its linear computation to the next layer; the rest will present zero values. Traditionally in the literature, the winner unit is selected to be the unit with greatest linear computation. Recently, stochastic competition principles have been considered, e.g. Panousis et al. (2019; 2021), Voskou et al. (2021).

Let us denote as $\boldsymbol{y} \in \mathbb{R}^{R \cdot J}$ the output of an LWTA layer; this is composed of R subvectors $\boldsymbol{y}_r \in \mathbb{R}^J$ and is sparse, since all units except for one, in each block, yield zero values. Let us introduce the discrete latent indicator vector $\boldsymbol{\xi} \in \text{one\_hot}(J)^R$ to denote the winner units in the R blocks that constitute a considered stochastic LWTA layer. This vector comprises R component subvectors, where each component entails one non-zero value at the index position that corresponds to the winner unit of the respective LWTA block. On this basis, the output $\boldsymbol{y}$ of the stochastic LWTA layer's $(r \cdot j)$-th component $\boldsymbol{y}_{r,j}$ is defined as:

$$\boldsymbol{y}_{r,j} = \boldsymbol{\xi}_{r,j} \sum_{i=1}^{I}(w_{i,r,j}) \cdot x_i \in \mathbb{R} \tag{3}$$

where we denote as $\boldsymbol{\xi}_{r,j}$ the j-th component of $\boldsymbol{\xi}_r$, and $\boldsymbol{\xi}_r \in \text{one\_hot}(J)$ holds the r-th subvector of $\boldsymbol{\xi}$.

In Eq. (3), we postulate that the latent winner indicator variables are drawn from a Categorical distribution which is proportional to the intermediate linear computation that each unit performs. Therefore, the stronger the linearity the higher the chance of the unit winning the stochastic competition within its block. In detail, we postulate that, a posteriori, the winner distributions yield:

$$q(\boldsymbol{\xi}_r) = \text{Categorical}\left(\boldsymbol{\xi}_r \middle| \text{softmax}(\sum_{i=1}^{I}[w_{i,r,j}]_{j=1}^{J} \cdot x_i)\right) \tag{4}$$

where $[w_{i,r,j}]_{j=1}^{J}$ denotes the vector concatenation of the set $\{w_{i,r,j}\}_{j=1}^{J}$. A graphical illustration of the proposed stochastic architecture is provided in Fig. 1.

As a network composed of such (StochLWTA) layers entails latent variables $\boldsymbol{\xi}$, we need to perform a Bayesian network treatment to perform effective parameter training. We opt for a (approximate) stochastic gradient variational Bayes treatment (Kingma & Welling, 2014), for scalability purposes. This means that the used objective function takes the form of an evidence lower-bound (ELBO) objective, as we describe next. In our work, we take one step further: we also elect to infer a posterior density over the network weights $\boldsymbol{W}$, as opposed to obtaining point-estimates. This results in a second source of stochasticity for our approach, which may further increase its generalization capacity under uncertain conditions arising from limited training data availability. Note that the use of these posteriors is totally different from Panousis et al. (2019) and Voskou et al. (2021): therein, posterior variance is used for compressing posterior mean bit-precision; then, predictions are performed using only the compressed posterior mean. Instead, in our work we sample multiple times from the trained weight posterior and perform model averaging (in a Bayesian sense), as a means of increasing generalization capacity.

We postulate:

$$q(\text{vec}(\boldsymbol{W})) = N(\text{vec}(\boldsymbol{W})|\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)) \tag{5}$$

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

3.2. A Model-Agnostic ML Algorithm

Let us define an ML problem where we are given a training dataset D of tasks, T, that are governed by a distribution $P(T)$. We consider an N-way, K-shot learning setting, where each task contains K labelled examples for each of N available classes.

Our approach entails parameter sets $\boldsymbol{\theta}$ which coincide with the hyperparameter sets $\boldsymbol{\psi} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}^2\}$ of the weights $\boldsymbol{W} \in \mathbb{R}^{I \times R \times J}$; these sets $\boldsymbol{\psi} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}^2\}$ are the target of our MAML-type training algorithm.

Therefore, to perform model training, we have to first initialize the trainable parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ across layers. To this end, one can appropriately exploit popular initialization schemes, such as Glorot Uniform (Glorot & Bengio, 2010). Then, our approach entails an inner-outer loop scheme, in a vein similar to existing approaches, but with some crucial differences:

1. The stochastic nature of the postulated weights, $\boldsymbol{W}$, results in the updates taking place over the posterior means, $\boldsymbol{\mu}$ and variances, $\boldsymbol{\sigma}^2$.

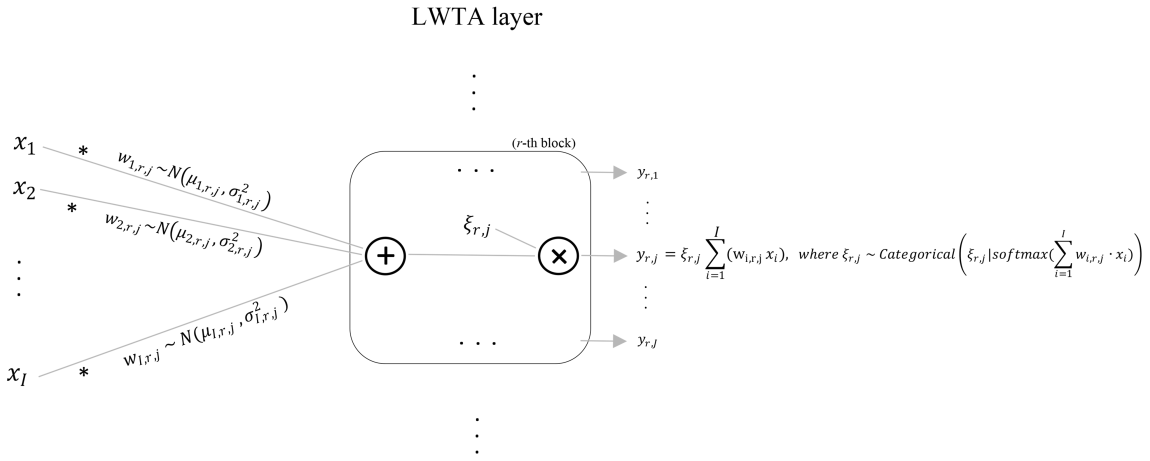


Figure 1. A zoomed-in graphical illustration of the r-th block of a stochastic LWTA layer. Input $\boldsymbol{x} = [x_1, x_2, \ldots, x_I]$ is presented to each unit in the block. The j-th component $\boldsymbol{y}_{r,j}$ of the block's output is obtained after computing latent variable $\boldsymbol{\xi}_{r,j}$.

2. The stochastic nature of both the inferred representations and the network weights themselves implies that proper training must rely on optimization of the ELBO function of the network. Let us consider an inner-loop dealing with task $T_i \sim P(T)$, with data $D_i = (X_i, Y_i) \subset D$. Let $\mathrm{CE}(Y_i, f_{\boldsymbol{\psi}}(X_i; \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{W}}))$ be the categorical cross-entropy between the data labels Y_i and the class probabilities $f_{\boldsymbol{\psi}}(X_i; \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{W}})$ generated by the Softmax layer. Then, we yield:

$$L_{T_i}(\boldsymbol{\psi}) = -\mathrm{CE}(Y_i, f_{\boldsymbol{\psi}}(X_i; \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{W}})) - KL[\, q(\boldsymbol{\xi}) \,||\, p(\boldsymbol{\xi}) \,] - KL[\, q(\boldsymbol{W}) \,||\, p(\boldsymbol{W}) \,] \tag{6}$$

for task T_i with dataset D_i which is dealt with on the i-th training iteration.

Here, for simplicity and without harming generality, we consider that the weights prior $p(\mathrm{vec}(\boldsymbol{W}))$ is a Gaussian distribution $N(\boldsymbol{0}, \boldsymbol{I})$ and the latent variables prior $p(\boldsymbol{\xi})$ is a symmetric Categorical distribution $\mathrm{Categorical}(1/J)$. In addition, in our notation we stress that the output $f_{\boldsymbol{\psi}}(X_i; \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{W}})$ depends on the winner selection process, which is stochastic, and the outcomes of sampling the network weights. Specifically, in our work we perform Monte-Carlo (MC) sampling using one reparameterized sample of the corresponding latent variables.

Let $\bar{\boldsymbol{\xi}}$ be the unnormalized probabilities of the Categorical distribution $q(\boldsymbol{\xi})$ (Eq. (4)). The sampled instances of $\boldsymbol{\xi}$, $\hat{\boldsymbol{\xi}}_{r,j}$, are expressed as (Maddison et al., 2017):

$$\hat{\boldsymbol{\xi}}_{r,j} = \mathrm{Softmax}((\log \bar{\boldsymbol{\xi}}_{r,j} + g_{r,j})/\tau), \quad \forall r = 1, \ldots, R,\ j = 1, \ldots, J \tag{7}$$

where $g_{r,j} = -\log(-\log U_{r,j})$, $U_{r,j} \sim \mathrm{Uniform}(0, 1)$, and $\tau \in (0, \infty)$ is a temperature factor that controls how closely the Categorical distribution is approximated by this continuous relaxation.

Similarly, the Gaussian weights yield: $\hat{w}_{t,r,j} = \mu_{t,r,j} + \sigma_{t,r,j}\hat{\epsilon}$, and $\hat{\epsilon} \sim N(0, 1)$.

On this basis, the KL divergences in Eq. (6) become:

$$KL[\, q(\boldsymbol{\xi}_{r,j} \,||\, p(\boldsymbol{\xi}_{r,j}) \,] = \mathbb{E}_{q(\boldsymbol{\xi}_{r,j})}[\log q(\boldsymbol{\xi}_{r,j}) - \log p(\boldsymbol{\xi}_{r,j})] \approx \log q(\hat{\boldsymbol{\xi}}_{r,j}) - \log p(\hat{\boldsymbol{\xi}}_{r,j}),\ \forall r,\ j \tag{8}$$

and

$$KL[\, q(w_{t,r,j}) \,||\, p(w_{t,r,j}) \,] = \mathbb{E}_{q(w_{t,r,j})}[\log q(w_{t,r,j}) - \log p(w_{t,r,j})] \approx \log q(\hat{w}_{t,r,j}) - \log p(\hat{w}_{t,r,j}),\ \forall t = 1, \ldots, I,\ r,\ j \tag{9}$$

Hence, the ELBO becomes:

$$L_{T_i}(\boldsymbol{\psi}) = -\mathrm{CE}(Y_i, f_{\boldsymbol{\psi}}(X_i; \hat{\boldsymbol{\xi}}, \hat{\boldsymbol{W}})) - \sum_{r,j} (\log q(\hat{\boldsymbol{\xi}}_{r,j}) - \log p(\hat{\boldsymbol{\xi}}_{r,j})) - \sum_{t,r,j} (\log q(\hat{w}_{t,r,j}) - \log p(\hat{w}_{t,r,j})) \tag{10}$$

Therefore, we establish a MAML-type algorithm, where: (i) the used networks comprise blocks of stochastic LWTA units visually depicted in Fig. 1; (ii) the trainable parameters are the means $\boldsymbol{\mu}$ and variances $\boldsymbol{\sigma}^2$ of the synaptic weights; and (iii) the objective function of the inner-loop process is given in Eq. (10).

We summarize our training algorithm in Alg. 1.

3.3. Prediction Algorithm

We start our prediction algorithm (Alg. 2) by sampling a task $T' \sim P(T)$, with data $D' = (X', Y') \subset D$. Then,

Algorithm 1 Model training with StochLWTA-ML

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

we draw a set of B samples of the Gaussian connection weights from the trained posteriors $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, and 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 of the network, which we average to obtain the final predictive outcome:

$$f_{\boldsymbol{\psi}}(X'; \tilde{\boldsymbol{\xi}}, \tilde{\boldsymbol{W}}) \approx \frac{1}{B} \sum_{s=1}^{B} f_{\boldsymbol{\psi}}(X'; \tilde{\boldsymbol{\xi}}_s, \tilde{\boldsymbol{W}}_s) \qquad (11)$$

where $\tilde{\boldsymbol{\xi}}_s$ and $\tilde{\boldsymbol{W}}_s$ are sampled directly from the posteriors $q(\boldsymbol{\xi})$ and $q(\boldsymbol{W})$, respectively.

This concludes the formulation of the proposed model-agnostic ML approach.

Algorithm 2 Prediction with StochLWTA-ML

Require: Learned parameters $\boldsymbol{\psi} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}^2\}$
Sample task $T' \sim P(T)$
for $s = 1$ **to** B **do**
  Sample $\tilde{\boldsymbol{W}}_s \sim q(\boldsymbol{W})$ defined in Eq. (5)
  Sample $\tilde{\boldsymbol{\xi}}_s \sim q(\boldsymbol{\xi})$ defined in Eq. (4), for $\boldsymbol{W} = \tilde{\boldsymbol{W}}$ and $(x_i = x_i') \in X'$
  Compute output logits $f_{\boldsymbol{\psi}}(X'; \tilde{\boldsymbol{\xi}}_s, \tilde{\boldsymbol{W}}_s)$, given the sampled values $\tilde{\boldsymbol{\xi}}_s$ and $\tilde{\boldsymbol{W}}_s$
end for
Use Eq. (11) to average over the resulting set of B logits and derive the final prediction.

4. Experiments

We evaluate our proposed model in various few-shot learning tasks: image classification, sinusoidal regression and active learning. After thorough exploration on the number of LWTA layers as well as the number of blocks for each layer and the competing units per block, we end up with using networks comprising 2 layers with 16 blocks and 2 competing units per block on the former layer, and 8 blocks with 2 units per block on the latter. The last network layer is a Softmax. Weight mean initialization, as well as point-estimate initialization for our competitors, is performed via Glorot Uniform. Weight log-variance initialization is performed via Glorot Normal, by sampling from $N(0.0005, 0.01)$. The Gumbel-Softmax relaxation temperature is set to $\tau = 0.67$.

In the inner-loop updates, we use the Stochastic Gradient Descent (SGD) (Robbins, 2007) optimizer with a learning rate of 0.003. For the outer-loop, we use SGD with a linear annealed outer step size to 0, and an initial value of 0.25. Additionally, all the experiments were ran with task batch size of 50 for both training and testing mode. Prediction is carried out averaging over $B = 4$ output logits. The code was implemented in Tensorflow (Abadi et al., 2016).

4.1. Classification

We first evaluate StochLWTA-ML on popular few-shot image classification datasets, and compare its performance to state-of-the-art prior results. In Table 1, we show how StochLWTA-ML performs on Omniglot 20-way (Lake et al., 2017), Mini-Imagenet 5-way (Vinyals et al., 2016) and CIFAR-100 5-way (Krizhevsky, 2009) few-shot settings. We compare our findings to state-of-the-art ML algorithms such as LLAMA and PLATIPUS as reported in Gordon et al. (2018), Amortized Bayesian Meta-Learning (ABML), MAML, FOMAML, Reptile and others. Using the original architectures with the same hyperparameters and data preprocessing as in Finn et al. (2017), we have also locally reproduced ABML, BMAML (with 5 particles), PLATIPUS, MAML, FOMAML and Reptile (dubbed "local" in Table 1).

For the local replicates, the results in Table 1 constitute average performance statistics and corresponding standard deviations (std's) over three runs, using different random seeds. For completeness sake, we also compare our findings to other state-of-the-art ML models as reported in Finn et al. (2017), including Matching Nets and LSTM Meta-Learner. As we observe, our method outperforms the existing state-of-the-art in both the 1-shot and 5-shot settings. Besides, the reported std statistics on the locally reproduced experiments show consistency across all methods; thus, our stochastic approach is as resilient to seed initialization as the existing approaches.

In the following, we perform a diverse set of ablations that shed light to the characteristics and capabilities of our approach. Few more ablations are deferred to Appendix D.

4.1.1. Does stochastic competition contribute to classification accuracy?

To check whether the accuracy improvements stem from the LWTA-induced sparsity or the proposed stochastic com-

Table 1. N-way K-shot (%) classification accuracies on Omniglot, Mini-Imagenet and CIFAR-100

	Omniglot 20-way		**Mini-Imagenet 5-way**		**CIFAR-100 5-way**	
Algorithm	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Matching Nets (Santoro et al., 2016)	93.80	98.50	43.56	55.31	-	-
LSTM Meta-Learner (Ravi & Larochelle, 2017)	-	-	43.44	60.60	-	-
MAML	95.80	98.90	48.70	63.11	-	-
FOMAML	-	-	48.07	63.15	-	-
Reptile	88.14	96.65	47.07	62.74	-	-
PredCP (Nalisnick et al., 2021)	-	-	49.30	61.90	-	-
Neural Statistician (Edwards & Storkey, 2016)	93.20	98.10	-	-	-	-
mAP-SSVM (Triantafillou et al., 2017)	95.20	98.60	50.32	63.94	-	-
LLAMA (Grant et al., 2018)	-	-	49.40	-	-	-
PLATIPUS (Finn et al., 2018)	-	-	50.13	-	-	-
GEM-BML+ (Zou & Lu, 2020)	96.24	98.94	50.03	-	-	-
DKT (Patacchiola et al., 2020)	-	-	49.73	64.00	-	-
ABML (Ravi & Beatson, 2019)	-	-	45.00	-	49.50	-
BMAML (with 5 particles) (Yoon et al., 2018)	-	-	53.80	-		-
ABML (local)	90.21 ± 0.34	93.39 ± 0.09	44.23 ± 0.81	52.12 ± 1.01	49.23 ± 0.23	53.60 ± 0.39
BMAML (local)	96.92 ± 0.58	98.11 ± 2.03	53.10 ± 1.05	64.80 ± 0.93	52.60 ± 1.40	65.80 ± 0.04
PLATIPUS (local)	94.35 ± 0.87	98.30 ± 0.44	49.97 ± 0.97	63.13 ± 1.18	51.14 ± 0.48	63.61 ± 2.16
MAML (local)	95.48 ± 0.81	98.61 ± 0.49	48.60 ± 1.23	63.01 ± 1.28	50.67 ± 0.93	62.89 ± 0.77
FOMAML (local)	94.92 ± 0.71	98.12 ± 0.94	47.93 ± 0.67	63.10 ± 0.58	49.13 ± 0.61	63.80 ± 0.83
Reptile (local)	87.98 ± 1.18	96.36 ± 1.54	46.97 ± 0.95	62.53 ± 0.61	48.19 ± 0.74	63.45 ± 1.63
StochLWTA-ML	**97.79 ± 0.48**	**98.97 ± 0.61**	**54.11 ± 0.82**	**66.70 ± 0.41**	**54.60 ± 0.39**	**66.73 ± 0.06**

petition concept, we evaluate both our approach as well as MAML, FOMAML, ABML, BMAML and PLATIPUS, considering both "deterministic LWTA" and "stochastic LWTA" setups; deterministic LWTA networks have been adopted from Srivastava et al. (2013). As we see in Table 2, replacing ReLU with deterministic LWTA yields negligible differences. On the other hand, stochastic LWTA units yield a clear improvement in all cases. This improvement becomes even more important in the case of our approach, where we sample from stochastic weights.

Table 2. Ablation study (% classification accuracy)

		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**

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

It is also important to investigate whether our approach represents a trade-off between accuracy and computational time compared to our competitors. To facilitate this investigation, in Table 3 we provide training iteration wall-clock times for our approach and the existing locally reproduced state-of-the-art, as well as the total number of iterations each model needs to achieve the reported performance of Table 1. It appears that our methodology takes 77% *less* training time than the less efficient algorithms ABML, BMAML, PLATIPUS, and is comparable to other approaches. This happens because our approach yields the reported state-of-the-art performance by employing a network architecture (that is, number of LWTA layers, as well as number of blocks and block size on each layer) that result in a total number of trainable parameters that is *one order of magnitude less* on average than the best performing baseline methods. This can be seen in the last three columns of Table 3 (dubbed D_A, D_B and D_C for Omniglot, Mini-Imagenet and CIFAR-100 respectively). In addition, training for our approach converges fast.

The situation changes when it comes to prediction: our approach imposes a slight computational time overhead compared to MAML, FOMAML and Reptile, but still *much less* than the time-consuming PLATIPUS. BMAML and ABML. This is a rather negligible increase when we are dealing with a low number of drawn samples, B. More information on the effect of sample size in our approach's performance are provided in the Supplementary.

Finally, we provide an example of how training for our approach converges, and how this compares to the alternatives.

Table 3. Performance comparison: average wall-clock time (in msecs), training iterations for each locally reproduced method and number of baselines' trainable parameters over the considered datasets of Table 1

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**

We illustrate our outcomes on the Omniglot 20-way 1-shot benchmark; similar outcomes have been observed in the rest of the considered datasets. Fig. 2(a) compares StochLWTA-ML with prior traditional ML methods: MAML, FOMAML and Reptile. It becomes apparent that our approach converges equally fast to these competitors. Further, Fig. 2(b) compares StochLWTA-ML with the probabilistic ML models ABML, BMAML, PLATIPUS. Since these methods are quite time-consuming and less efficient regarding to memory consumption, StochLWTA-ML gives rise to an easier time training MAML based probabilistic model.

4.1.3. Effect of block size J

As it is presented in Table 4, 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. This corroborates our initial choice of using $J = 2$ competing units per block in our approach.

Table 4. Effect of block size J in StochLWTA-ML's classification (%) accuracy

	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

4.1.4. How does the task batch size affect StochLWTA-ML's performance?

For demonstration purposes, in Fig. 3(a) and 3(b) we illustrate the performance of our model on the Mini-Imagenet 5-way 1-shot setting with different task batch sizes. As we see, 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.

4.1.5. How does the sample size B at prediction time affect StochLWTA-ML's accuracy?

In Fig. 3(c), we illustrate how sample size B affects predictive accuracy on the Omniglot 20-way 1-shot, Mini-Imagenet 5-way 1-shot and CIFAR-100 5-way 5-shot settings. As we observe, an increase in sample size, B, does not always yield an accuracy increase. It seems that selecting $B = 4$ allows for the best predictive accuracy/computational complexity trade-off. We have obtained similar findings for the remainder of the considered experimental settings as well. Thus, we finally choose $B = 4$ throughout our experiments.

4.1.6. How important are the stochastic weights?

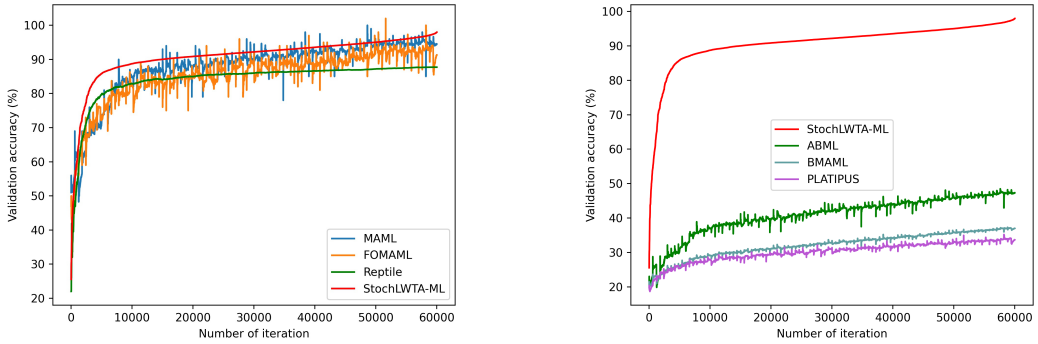
We perform an ablation study on Omniglot 20-way; the goal is to discern how much of an extra improvement Gaussian weights offer. Our results are shown in Table 5. Apparently, stochastic LWTA units offer the greatest fraction of the accuracy gains, but the Gaussian weights are still indispensable.

Table 5. Omniglot 20-way ablation study: 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**

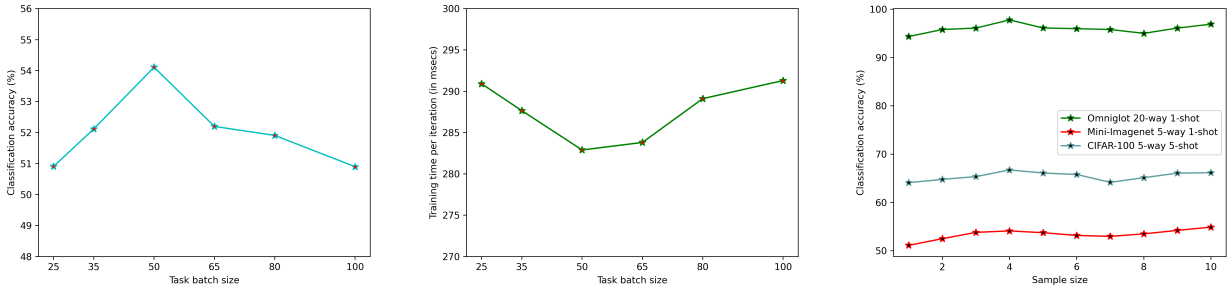
4.2. Regression

In this section, we compare our approach StochLWTA-ML with the locally reproduced baselines of Section 4.1, on two sinusoidal function regression problems. In the former case, we apply the default setting used in Finn et al. (2017); in the latter, we use a more challenging setting as proposed in Yoon et al. (2018), containing more uncertainty than the setting used in Finn et al. (2017). Specifically, the



(a) (b)

Figure 2. ML algorithms' training convergence comparison



(a) (b) (c)

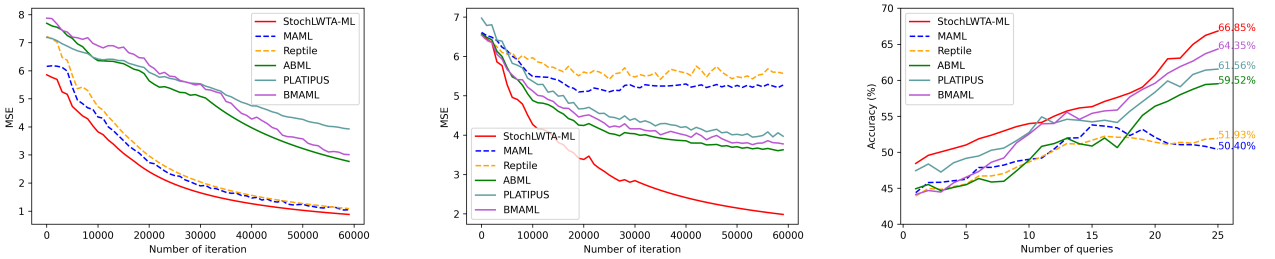
Figure 3. (a) The effect of task batch size in StochLWTA's predictive accuracy, (b) the effect of task batch size in StochLWTA's training time per iteration (in msecs), and (c) the effect of sample size B in StochLWTA-ML's classification (%) accuracy

tasks distribution $P(T)$ is defined by a sinusoidal function $y = A\sin(\omega x + b) + \epsilon$, with amplitude A, frequency ω, phase b and observation noise ϵ. The parameters of each task are sampled from Uniform distributions $A \in [0.1, 5.0]$, $b \in [0.0, 2\pi]$, $\omega \in [0.5, 2.0]$, and observation noise ϵ from Normal distribution $N(0, (0.01A)^2)$. For each task, 10 input instances x are sampled from $[-5.0, 5.0]$. The network architecture employed for the baseline experiments consists of 2 hidden layers of size 64 with tanh activation. In our StochLWTA-ML case, we replace each hidden layer with a proposed stochastic LWTA one. Both architectures end up with a Softmax layer.

In Fig. 4, we illustrate the Mean Squared Error (MSE) performance on test tasks for both settings, after training all the models for 60000 iterations. Specifically, in Fig. 4(a) we observe that StochLWTA-ML, in the default setting, adapts equally fast as MAML and Reptile; thus, its convergence speed is much higher than the other time-consuming probabilistic methods ABML, BMAML and PLATIPUS. In the challenging setting of Fig. 4(b), the Bayesian methods StochLWTA-ML, ABML, BMAML and PLATIPUS can still perform well in a high uncertainty setting while non-Bayesian models MAML and Reptile fail to converge; this outcome is achieved due to the ability of Bayesian methods to reduce overfitting and generalize better. It is clear that our approach yields the most time-efficient method among the baselines.

4.2.1. Is there a computational time trade-off for the reduced MSE?

In Table 6, we report the numbers of trainable parameters as an average over the default and challenging settings of the regression experiments. In a similar vein with the classification outcomes of Section 4.1.2, it appears that our methodology takes 77% *less* training time than the less efficient algorithms ABML, BMAML, PLATIPUS, and is comparable to other approaches. This demonstrates the effect of parameters' number reduction to the training process of a MAML based probabilistic model.



(a) (b) (c)

Figure 4. Sinusoidal regression results: (a) MSE of default setting after 60000 training iterations, (b) MSE of challenging setting after 60000 training iterations, and (c) active learning setting

Table 6. Performance comparison: average wall-clock time (in msecs), and average number of baselines' trainable parameters over the two settings of regression experiments

Algorithm	Training	Prediction	Parameters
PLATIPUS (local)	359.23	135.74	21541
BMAML (local)	323.76	115.90	21541
ABML (local)	150.96	59.75	8622
MAML (local)	65.54	23.91	4315
FOMAML (local)	64.80	**23.02**	4315
Reptile (local)	64.63	23.30	4353
StochLWTA-ML	**63.11**	25.45	**2092**

4.3. Active learning with regression

To further evaluate the effectiveness of the proposed stochastic LWTA network, we now consider an active learning experiment, in the challenging setting described in Section 4.2. Specifically, we provide the models with 5 input observations x sampled from $[-5.0, 5.0]$. Then, each model queries to label 5 extra instances. The Bayesian methods StochLWTA-ML, ABML, BMAML and PLATIPUS choose an item x^* that has the maximum variance across the sampled regressors. However, the non-Bayesian models MAML and Reptile, choose points randomly, since they do not entail a utility that handles uncertainty; as we demonstrate in Fig. 4(c), they fail to converge due to their inability to adapt to a more ambiguous training environment. Considering the Bayesian methods, our approach achieves the better predictive performance compared to the other baselines. This reaffirms the prevalence of our proposed network to another experimental scenario.

5. Conclusion

In this paper, we proposed a sparse and stochastic network paradigm for ML, with novel network design principles compared to currently used model-agnostic ML models. We introduced *stochastic LWTA activations* in the context of a *variational Bayesian treatment* that gave rise to a doubly-stochastic ML framework, bearing the promise of stronger generalization capacity. We evaluated our approach using standard image classification benchmarks in the field, and showed that it outperformed the state-of-the-art in terms of both predictive accuracy, error rate and computational costs. The results have provided strong empirical evidence supporting our claims. In the future, we plan to study the effect of StochLWTA-ML in areas related to ML, such as Continual Learning (Javed & White, 2019) and Reinforcement Learning (Zhu et al., 2020).

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 872139, project aiD.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., and Devin, M. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. pp. 265–283. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, 2016.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In *Neural Information Processing Systems*, 2016.

Baik, S., Choi, M., Choi, J., Kim, H., and Lee, K. M. Meta-learning with adaptive hyperparameters. In *Neural Information Processing Systems*, 2020.

Black, P., McCormick, R., James, M., and Pedderd, D. Learning how to learn and assessment for learning: a theoretical inquiry. *Research Papers in Education*, 21: 119–132, 2006.

Buysse, H. M., Mets, K., and Latré, S. Fast task-adaptation for tasks labeled using natural language in reinforcement learning. 2019. URL https://arxiv.org/abs/1910.04040.

Castro, R., Kalish, C., Nowak, R., Qian, R., Rogers, T., and Zhu, J. Human active learning. In *Neural Information Processing Systems*, 2008.

Chen, Y., Friesen, A. L., Behbahani, F., Doucet, A., Budden, D., Hoffman, M., and de Freitas, N. Modular meta-learning with shrinkage. In *Neural Information Processing Systems*, 2020.

Edwards, H. and Storkey, A. Towards a neural statistician. 2016. URL https://arxiv.org/abs/1606.02185.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Finn, C., Xu, K., and Levine, S. Probabilistic model-agnostic meta-learning. In *Neural Information Processing Systems*, 2018.

Finn, C., Rajeswaran, A., Kakade, S., and Levine, S. Online meta-learning. In *International Conference on Machine Learning*, 2019.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research 9*, pp. 249–256, 2010.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. E. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*, 2018.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.

Javed, K. and White, M. Meta-learning representations for continual learning. pp. 1818–1828. In *Neural Information Processing Systems*, 2019.

Kessler, S., Nguyen, V., Zohren, S., and Roberts, S. Hierarchical indian buffet neural networks for bayesian continual learning. In *UAI*, 2021.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Kühl, N., Goutier, M., Baier, L., Wolff, C., and Martin, D. Human vs. supervised machine learning: Who learns patterns faster? 2020. URL https://arxiv.org/abs/2012.03661.

Lai, Y. A comparison of traditional machine learning and deep learning in image recognition. *Journal of Physics: Conference Series*, 1314, 2019.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

Lee, J., Kim, S., Yoon, J., Lee, H. B., Yang, E., and Hwang, S. J. Adaptive network sparsification with dependent variational beta-bernoulli dropout. 2018. URL https://arxiv.org/abs/1805.10896.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

Nalisnick, E., Gordon, J., and Hernández-Lobato, J. M. Predictive complexity priors. In *Neural Information Processing Systems*, 2021.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. 2018. URL https://arxiv.org/abs/1803.02999.

Panousis, K., Chatzis, S., Alexos, A., and Theodoridis, S. Local competition and stochasticity for adversarial robustness in deep learning. In *International Conference on Artificial Intelligence and Statistics*, 2021.

Panousis, K. P., Chatzis, S., and Theodoridis, S. Nonparametric bayesian deep networks with local competition. In *International Conference on Machine Learning*, 2019.

Patacchiola, M., Turner, J., Crowley, E. J., O'Boyle, M., and Storkey, A. Bayesian meta-learning for the few-shot setting via deep kernels. In *Neural Information Processing Systems*, 2020.

Ravi, S. and Beatson, A. Amortized bayesian meta-learning. In *International Conference on Learning Representations*, 2019.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

Robbins, H. A stochastic approximation method. *Annals of Mathematical Statistics*, 2007.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016.

Sculley, D., Holt, G., Golovin, D., Davydov, E., and Phillips, T. Hidden technical debt in machine learning systems. In *Neural Information Processing Systems*, 2015.

Srivastava, R. K., Masci, J., Kazerounian, S., Gomez, F., and Schmidhube, J. Compete to compute. In *Neural Information Processing Systems*, 2013.

Triantafillou, E., Zemel, R., and Urtasun, R. Few-shot learning through an information retrieval lens. In *Neural Information Processing Systems*, 2017.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. In *Neural Information Processing Systems*, 2016.

Voskou, A., Panousis, K., Kosmopoulos, D., Metaxas, D., and Chatzis, S. Stochastic transformer networks with linear competing units: Application to end-to-end sl translation. In *International Conference on Computer Vision*, 2021.

Yoon, J., Kim, T., Dia, O., Kim, O., Bengio, Y., and Ahn, S. Bayesian model-agnostic meta-learning. In *Neural Information Processing Systems*, 2018.

Zhu, H., Yu, J., Gupta, A., Shah, D., Hartikainen, K., Singh, A., Kumar, V., and Levine, S. The ingredients of real-world robotic reinforcement learning. In *International Conference on Learning Representations*, 2020.

Zou, Y. and Lu, X. Gradient-em bayesian meta-learning. In *Neural Information Processing Systems*, 2020.

A. Further details on the used datasets

Omniglot is a dataset of 1623 characters from different alphabets, containing 20 examples per character scaled down to 28x28 grayscale pixels. The ratio between training and testings sets is 3:2, so after shuffling the character classes we randomly choose the first 974 classes for training and the remaining are left for testing. As for the Mini-Imagenet dataset, it has color images of size 84x84 and contains 100 classes with 600 examples from the ImageNet dataset. We randomly choose 45000 examples for the training phase and the rest constitute the testing population. The CIFAR-100 dataset consists of color images of size 32x32 and contains 100 classes with 600 images per class. We randomly choose 500 images per class for training and the rest 100 images per class constitute the testing population.

B. Few-Shot Classification Network Architectures

For the local replicates of prior ML algorithms in the experiments of our work, we follow the same architecture for the deep neural network as the one used by Vinyals et al. (2016). For Omniglot, the network is composed of 4 convolutional layers with 64 filters, 3 x 3 convolutions and 2 x 2 strides, followed by a Batch Normalization layer (Ioffe & Szegedy, 2015) and the final values of each layer are processed by an activation function. For both Mini-Imagenet and CIFAR-100, we use 4 convolutional layers with 32 filters to reduce overfitting like Ravi & Larochelle (2017), 3 x 3 convolutions followed by Batch Normalization layer and 2×2 max-pooling layer with the values of each layer finally passed again through an activation block. The activation function used for the baseline experiments is: ReLU for Tables 1, 3, 6 and D1, and LWTA for Table 2.

C. What parameters do we count for the outcomes of Tables 3, 6 and D1?

The included parameters of each baseline for the outcomes of Tables 3, 6 and D1 are:

- PLATIPUS: $\Theta = \{\boldsymbol{\mu}_\theta, \boldsymbol{\sigma}_\theta^2, \boldsymbol{v}_\theta, \boldsymbol{\gamma}_p, \boldsymbol{\gamma}_q\}$
- BMAML: $\Theta = \{\theta^m\}_{m=1}^5$, for using 5 particles
- ABML: $\theta = \{\boldsymbol{\mu}_\theta, \boldsymbol{\sigma}_\theta^2\}$
- MAML: $\theta = \{\boldsymbol{\mu}_\theta\}$
- FOMAML: $\theta = \{\boldsymbol{\mu}_\theta\}$
- Reptile: $\theta = \{\boldsymbol{\mu}_\theta\}$
- StochLWTA-ML: $\theta = \{\boldsymbol{\mu}_\theta, \boldsymbol{\sigma}_\theta^2\}$

Table D1. Performance comparison: wall-clock time (in msecs), training iterations for each locally reproduced method, classification accuracy and number of baselines' trainable parameters over the Omniglot 20-way 1-shot benchmark

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**

D. More Ablations

D.1. How do the state-of-the-art methods perform with a parameter count reduced to be about the same as StochLWTA-ML?

We repeated Omniglot 20-way 1-shot experiments to evaluate all state-of-the-art, using a deep network of the same number of parameters and similar architecture as the proposed StochLWTA-ML model. To this end, we simply replaced the 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, as we show in Table D1.

D.2. How does StochLWTA-ML perform with more parameters?

We repeated a classification experiment on Omniglot 20-way 1-shot by using a stochastic LWTA architecture of 4 convolutional layers. In this context, in each layer competition is performed among the feature maps of a convolutional kernel; this proceeds on a position-wise basis. The so-obtained convolutional deep network yields the same number of parameters as in the state-of-the-art. Our experimental outcomes are conspicuous: our approach lost accuracy (the reported 97.79% accuracy reduced to 96.50%), while training time increased by four and inference time almost doubled; similar outcomes have been observed in the rest of the considered datasets.