Discovering Weight Initializers with Meta Learning

Dmitry Baranchuk and Artem Babenko

Yandex

Goal

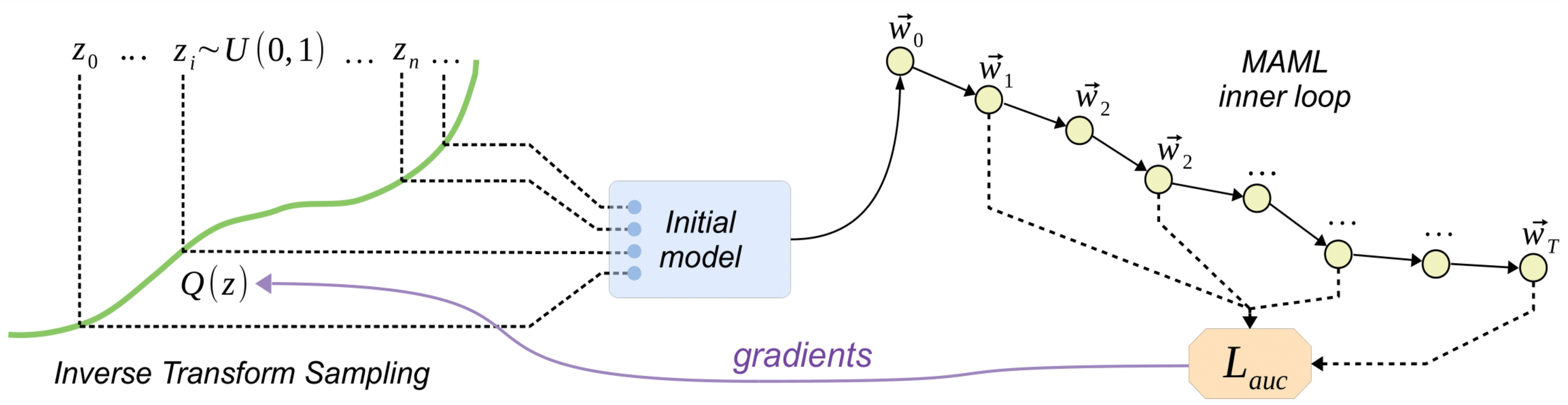
Develop an automatic method for discovering optimal initial weight distributions for arbitrary neural networks.

- In practice, finding effective model-specific initializers is an arduous task that requires human intuition and theoretical insights;
- Many sophisticated architectures and training protocols appear regularly.

Contributions

- Propose **DIMAML** – an automated approach that learns initial parameter distributions for a given architecture by directly optimizing its training performance;
- Evaluate DIMAML on several architectures and demonstrate that the learned initialization strategies match or outperform existing alternatives;
- Examine universality of the learned initializers by measuring their ability to generalize to unseen tasks, including different data distributions and training protocols;
- The code is available at *https://github.com/yandex-research/learnable-init*.

General Approach



DIMAML parameterizes weight initializers and exploits the ideas from Model-Agnostic Meta-Learning [1] to learn them by backpropagating through the training loop:

1. Define T_{train} tasks to learn initializers and different T_{test} tasks for evaluation;
2. Define initial weight distributions $p_\psi(\theta_{init})$ with trainable parameters ψ;
3. Sample i.i.d. initial weights $\theta_{init} \sim p_\psi(\theta_{init})$;
4. Train the model for a problem-specific objective using gradient descent and calculate validation loss on intermediate training steps;
5. Backpropagate through the entire training procedure and update the meta-parameters ψ.

Method

Initializer Parameterizations

We consider two parameterizations for initial distributions $p_\psi(\theta_{init})$.

Normal Initializers

- $p_\psi(\theta_{init})$ – a Normal distribution with parameters ψ = {μ, σ};
- Sample initial weights for layer l: $\theta_l = \mu_l + \sigma_l \cdot z,\ z \sim N(0,1)$.

PLIF Initializers

- Represent $p_\psi(\theta_{init})$ in the form of its quantile function Q_ψ, which is modeled as **PLIF** – a **p**iecewise **l**inear **i**ncreasing **f**unction [2] defined in [0, 1] with trainable slope and bias parameters;
- Inverse transform sampling – sample initial weights for layer l: $\theta_l = F_\psi^{-1}(z) = Q_\psi(z)\ ,\ z \sim \mathrm{U}(0,1)$.

Training Algorithm

1. Define the initial distribution for the i-th weight tensor - $p^i_\psi(\theta_i)$;
2. Sample $\theta_i \sim p^i_\psi(\theta_i)$ for each weight tensor in the model;
3. Train the model with the gradient descent algorithm (e.g., SGD or Adam) for N training steps;
4. Compute the average validation loss measured on intermediate training steps: $L_{auc} = \sum_{i=1}^{N} L_{step_i}(x_{val}, y_{val})$;
5. Compute $\partial L / \partial \psi$ by backpropagating through the training loop;
6. Update meta-parameters ψ by meta-optimizer.

Memory Efficient MAML

- **Limitation:** large memory footprint when applying MAML to commonly used neural network architectures and datasets.
- **Solution:** Gradient checkpointing [4] stores only 1 in m optimizer states in device memory, recomputing intermediate steps on the fly. Thus, it allows us to fit about 10-100x more optimizer steps in the same GPU memory.

Experiments

Autoencoders

$\mathcal{T}_{train}$	Tiny ImageNet			Tiny ImageNet		
$\mathcal{T}_{test}$	AnimeFaces			AnimeFaces Shuffled Pixels		
Epoch	10	50	100*	10	50	100*
Kaiming	0.471	0.383	0.369	0.823	0.643	0.590
DeltaOrthogonal	0.496	0.392	0.377	0.842	0.743	0.652
MetaInit [4]	0.552	0.437	0.398	0.859	0.792	0.708
DIMAML-Normal	**0.386**	**0.352**	**0.344**	**0.814**	**0.570**	**0.546**
DIMAML-PLIF	**0.386**	**0.350**	**0.344**	**0.812**	**0.567**	**0.545**

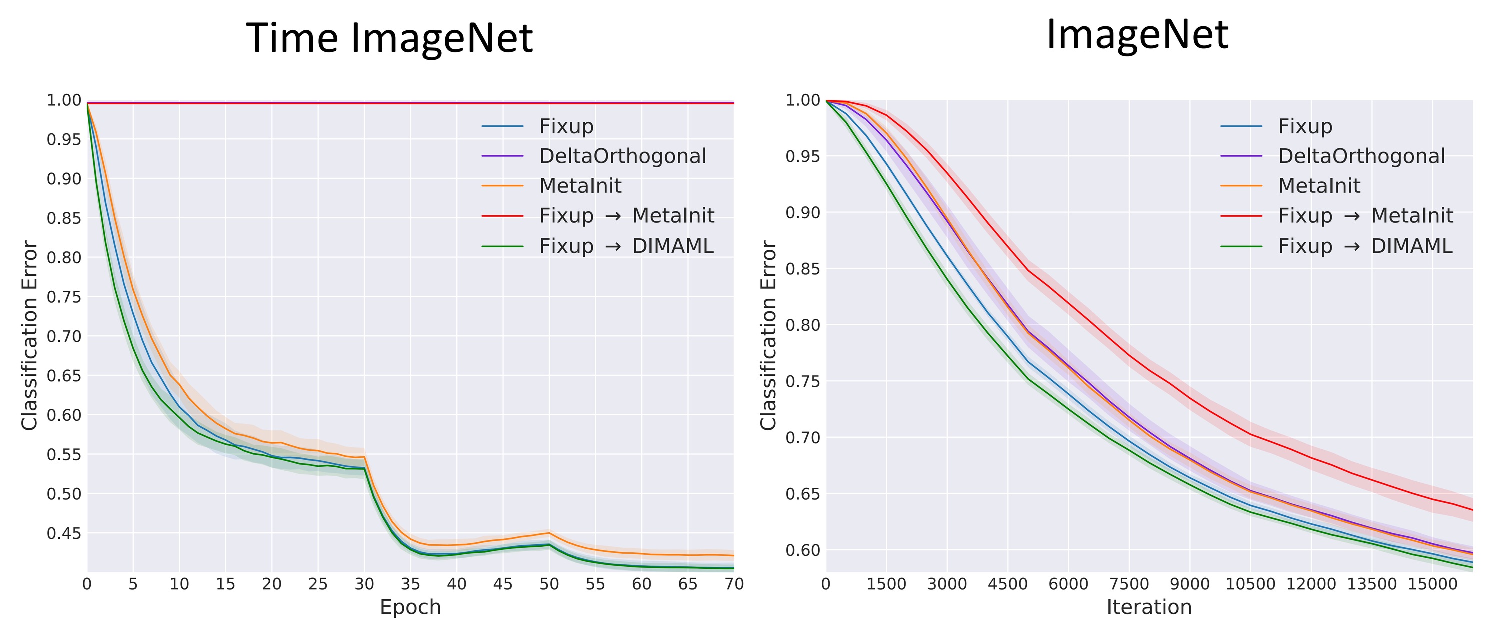
Comparison of autoencoder models with different initializers in terms of mean squared error. (*) corresponds to convergence.

Language models

$\mathcal{T}_{train}$	PennTreebank			Wikitext2		
$\mathcal{T}_{test}$	Wikitext2			PennTreebank		
Epoch	10	50	100*	10	50	100*
Kaiming	1.983	1.843	1.801	1.492	1.377	1.352
Orthogonal	2.017	1.849	1.806	1.528	1.383	1.354
MetaInit [4]	1.907	1.819	1.792	1.477	1.382	1.359
DIMAML-Normal	**1.879**	**1.812**	**1.782**	**1.454**	**1.372**	**1.345**
DIMAML-PLIF	**1.876**	**1.810**	**1.784**	**1.452**	**1.369**	**1.344**

Performance of the character-level language model in bits-per-character. DIMAML initial distributions speedup the training and converges to better optima for the same number of epochs. (*) corresponds to convergence.

Residual Networks



Classification performance on Tiny ImageNet (Left) and ImageNet (Right) for ResNet-18. During the first epochs, DIMAML is superior over all baselines and then converges similar to top-performing Fixup reaching the same optima.

References

[1] C. Finn, P. Abbeel, and S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, ICML2017.

[2] O. Ganea et al., Breaking the Softmax Bottleneck via Learnable Monotonic Pointwise Non-linearities, ICML 2019.

[3] T. Chen et al., Training deep nets with sublinear memory cost, arXiv 2016.

[4] Y. Dauphin, S. Schoenholz, MetaInit: Initializing learning by learning to initialize, NeurIPS 2019.