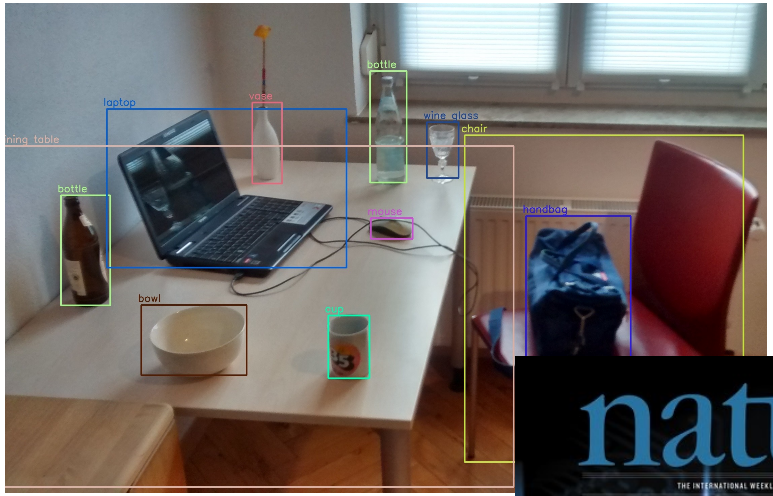
# Guarantees for Tuning the Step Size using a Learning-to-Learn Approach

Xiang Wang, Shuai Yuan, Chenwei Wu, Rong Ge

Duke University

# Optimization for neural networks





How to train the nets?

Just use SGD/Adam!

Step size, momentum,
weight decay,
……

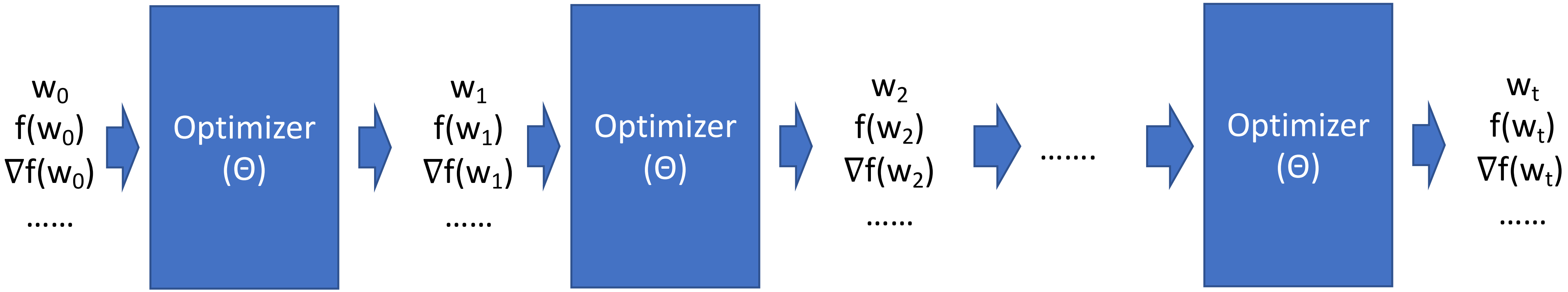
# Learning to Learn

**Learning to learn by gradient descent by gradient descent**

- Idea: use a meta-learning approach to tune hyper-parameters or learn a new optimizer!
  [Andrychowicz et al. 2016, Wichrowska et al. 2017, Metz et al. 2019]
- Goal: optimize objective function f(w) for a distribution of tasks.
- Idea: Abstract the optimization algorithm as an optimizer with parameter Θ. Optimize the parameter Θ for the distribution of task.


- Optimizer can be simple but can even be a neural network.

# How to train an optimizer?




- Unroll the optimizer for t steps.
- Define a meta-objective over the trajectory.
- Do (meta-)gradient descent on optimizer parameter Θ.
- No theoretical guarantees on training process or the learned optimizer

This work: Analyze step size tuning in GD/SGD for simple quadratic objectives.

# Optimizing the step size for a simple quadratic objective

- Naïve meta-objective: loss at last step

Point w at T-th iteration with step size $\eta$

$$F(\eta) = f(w_{\eta,T})$$

- **Theorem**: For almost all values of $\eta$, the meta-gradient $F'(\eta)$ is either exponentially large or exponentially small in T.

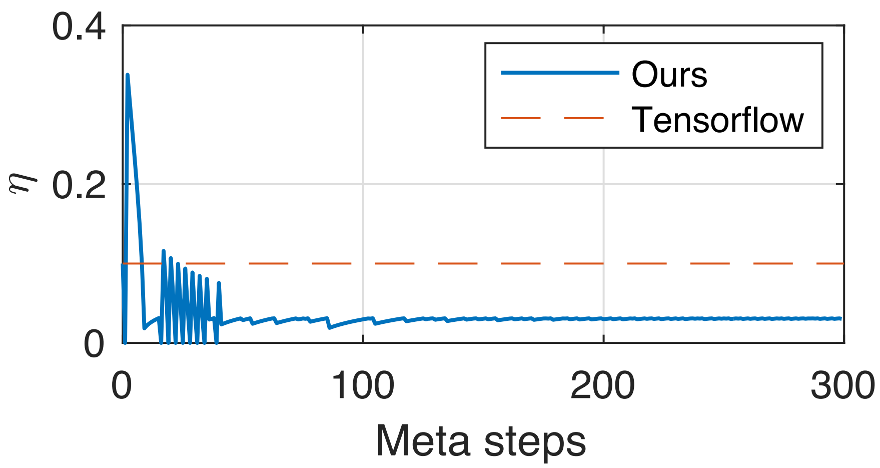
- Idea: meta-gradient is exponentially large (small) because the meta-objective is exponentially large (small) in T.

- New objective: $G(\eta) = \frac{1}{T}\log f\left(w_{\eta,T}\right) = \frac{1}{T}\log F(\eta)$

- **Theorem**: For the new objective, the meta-gradient $G'(\eta)$ is always polynomial in all relevant parameters.

# Numerical Issues in Computing Meta-gradient

- $\mathrm{G}'(\eta) = \frac{dG}{dF} \cdot \mathrm{F}'(\eta)$, both terms are exponentially large or small, but they cancel each other.
- This is exactly how one would compute $\mathrm{G}'(\eta)$ using backpropagation ➔ numerical issues!




Training trajectory for the actual meta-gradient vs. meta-gradient computed by TensorFlow

# Generalization of trained optimizer

- Recall that $w_{\eta,T}$ is the weight $w$ at the T-th iteration with step size $\eta$
- Two ways to define the meta-objective:

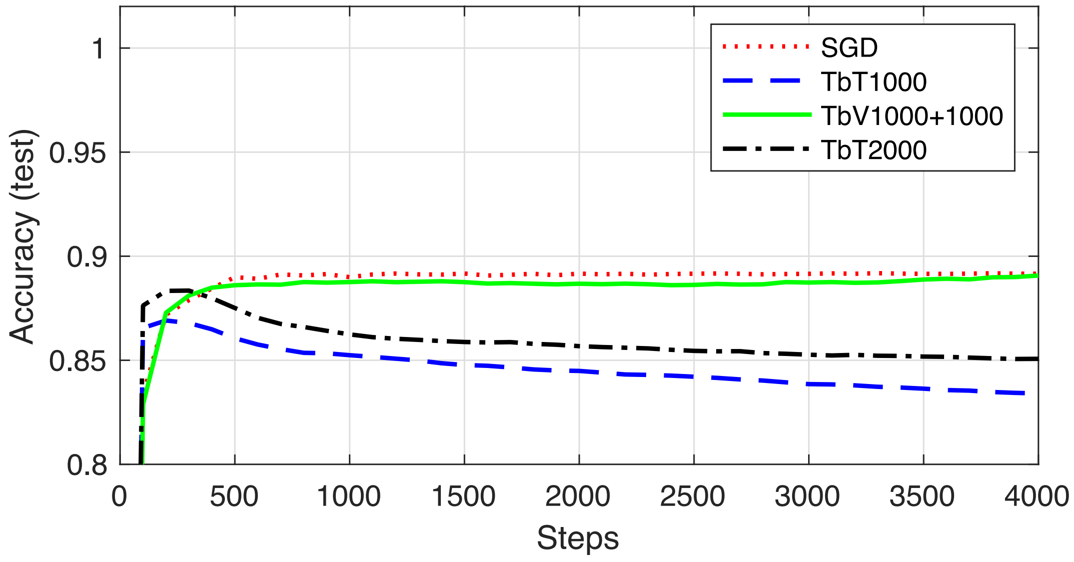
1. Train-by-train (original approach used in [Andrychowicz et al. 2016])
    - Define meta-objective on training set
    - e.g., simply choose $F(\eta) = f(w_{\eta,T})$
2. Train-by-validation [Metz et al. 2019]
    - Define meta-objective on a validation set (evaluate $w_{\eta,T}$ on a validation set)

# When do we need train-by-validation?

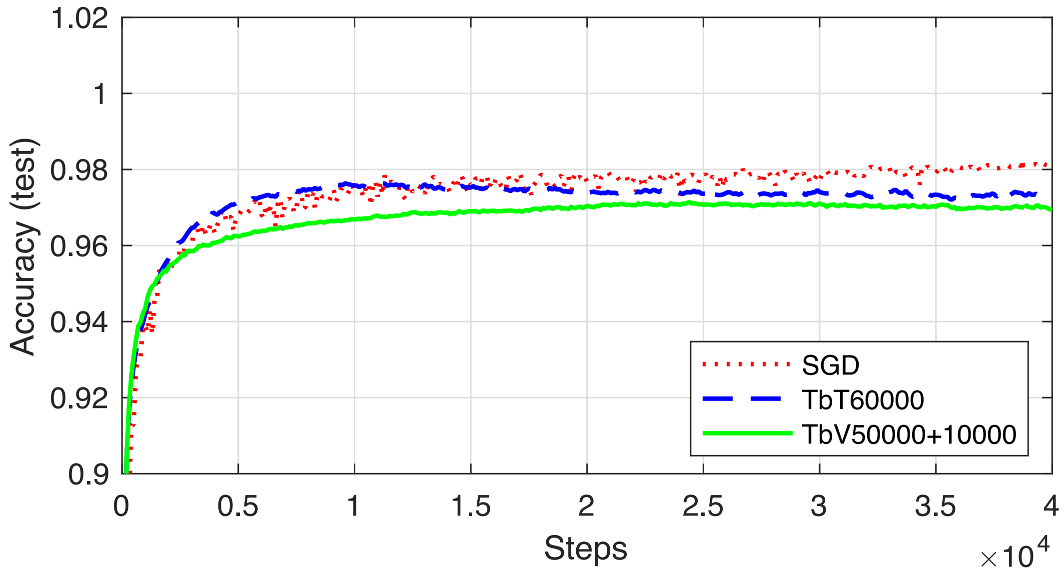
**Theorem**:

- 1. when noise $\sigma$ is large, and n (#samples) is a constant fraction of d (#dimension), then train-by-validation is better.
- 2. When n (#samples) is much larger than d (#dimension), then train-by-train is close to optimal.

# Empirical observation on neural net optimizers




1000 samples (MNIST)




All samples (MNIST)

# Conclusion

- Choosing meta-objective carefully may alleviate gradient explosion/vanishing problem; needs to be careful with backprop.
- When there are fewer samples/more noise, need to define meta-objective on a separate validation set.

Paper link:

Thank You!