Beyond Backprop:

Different Approaches to Credit Assignment in Neural Nets

Irina Rish (Mila & UdeM)

NYU


Anna Choromanska


Benjamin Cowen

IBM Research


Sadhana Kumaravel


Ronny Luss


Mattia Rigotti


Irina Rish

MIT


Ravi Tejwani


Brian Kingsbury


Paolo Di Achillele

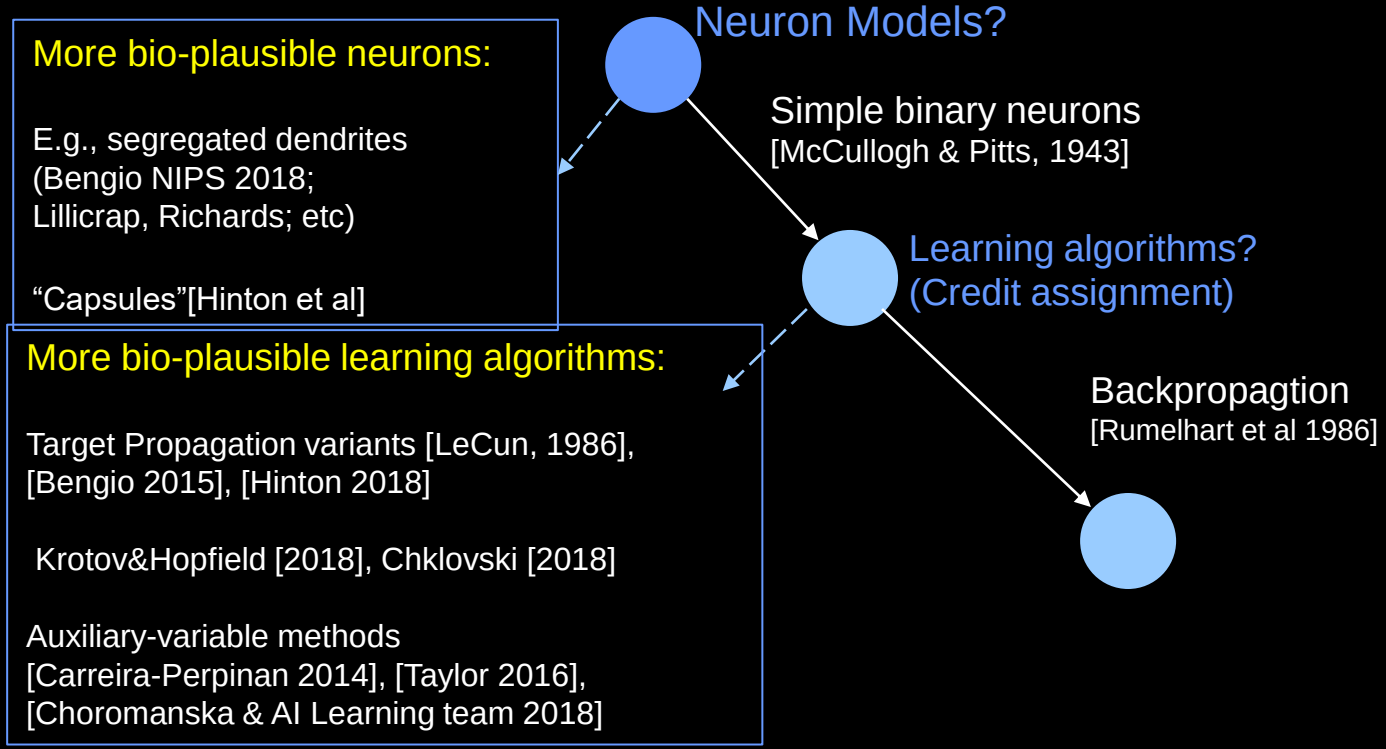

Viatcheslav Gurev


Djallel Bouneffouf

RETHINKING DEEP LEARNING?

Let's "backtrack" in the decision-tree of AI history:

welcome to the "garden of forking path"(a la Borges ☺)



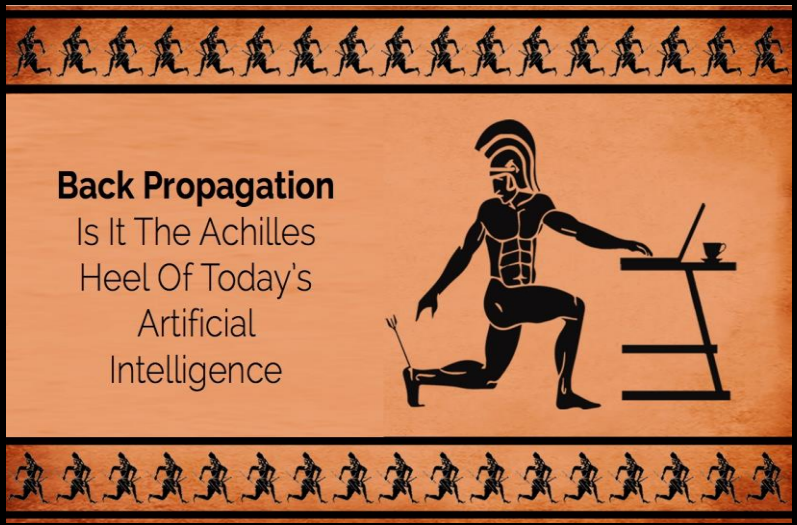
RETHINKING DEEP LEARNING?

"Artificial intelligence pioneer says we need to start over"

interview with Geoff Hinton, 2017

Geoff Hinton is "deeply suspicious of back-propagation'', workhorse method that underlies most of the advances we are seeing in the AI field today…

"My view is throw it all away and start again."



WHAT'S WRONG WITH BACKPROP?

Biologically implausibility:

- Error feedback does not influence neural activity, and hence does not conform to known biological feedback mechanisms underlying neural communication
- Weight transport problem: symmetric weight connectivity for feedforward and feedback directions
- Many other issues (precise clocking between feedforward and backprop phases, violation of Dale's law, etc)

Computational Issues:

- Vanishing gradients (due to chain of derivatives)
- Difficulty handling non-differentiable nonlinearities (e.g., binary spikes)
- Lack of cross-layer parallelism

ALTERNATIVES: PRIOR WORK

LeCun, Yann. Learning process in an asymmetric threshold network.
In Disordered systems and biological organization, 1986.

LeCun, Yann. Modeles connexionnistes de lapprentissage. PhD thesis, Universite Paris 6, 1987.

Lee, Zhang, Fischer and Bengio. Difference target propagation. ECML-2015

Bengio, Y. How auto-encoders could provide credit assignment in deep networks via target propagation. arXiv:1407.7906, 2014.

Bartunov, S.; Santoro, A.; Richards, B. A.; Hinton, G. E.; and Lillicrap, T.
Assessing the scalability of biologically-motivated deep learning algorithms and architectures, NeurIPS 2018.

ALTERNATIVES: PRIOR WORK

- **Offline Auxiliary-variable methods**
  - MAC (Carreira-Perpiñán & Wang, 2014) and other BCD methods (Zhang & Brand, 2017; Zhang & Kleijn, 2017; Askari et al., 2018; Zeng et al., 2018; Lau et al., 2018; Gotmare et al., 2018)
  - ADMM (Taylor et al., 2016; Zhang et al., 2016)
  - offline (batch) is not scalable to large data and continual learning

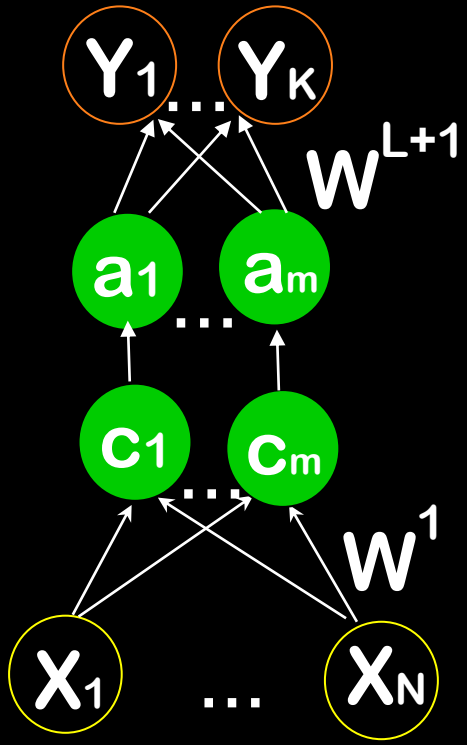
- **Target propagation methods**
  - [LeCun 1986] [Lee, Fisher, Bengio 2015] [Bartunov et al, 2018]
  - Below backprop-SGD performance levels on standard benchmarks

- **Proposed method:**
  - Online (mini-batch, stochastic) auxiliary-variable alternating-minimization

OUR APPROACH

Breaking gradient chains with auxiliary activation variables:

- Relaxing nonlinear activations to noisy (Gaussian) linear activations followed by nonlinearity (e.g., ReLU)
- Alternating minimization over activations and weights: explicit activation propagation
- Weight updates are layer-local, and thus can be parallel (distributed, asynchronous)



NEURAL NETWORK FORMULATIONS

Standard neural network objective function:

Nested

$$\min_{W} \quad \mathcal{L}(y, f(\boldsymbol{W}, \boldsymbol{x}_L))$$
$$\text{where} \quad f(\boldsymbol{W}, \boldsymbol{x}_L) = f_{L+1}(\boldsymbol{W}_{L+1}, f_L(\boldsymbol{W}_L, f_{L-1}(\boldsymbol{W}_{L-1}, ...f_1(\boldsymbol{W}_1, \boldsymbol{x})...)$$

Add auxiliary activation variables (hard constrained problem)

Constrained

$$\min_{\boldsymbol{W},C} \quad \sum_{t=1}^{n} \mathcal{L}(\boldsymbol{y}_t, \boldsymbol{a}_t^L, \boldsymbol{W}^{L+1}), \text{ where } \boldsymbol{a}_t^l = \sigma_l(\boldsymbol{c}_t^l),$$
$$\text{s.t.} \quad \boldsymbol{c}_t^l = \boldsymbol{W}^l \boldsymbol{a}_t^{l-1}, \; l = 1, ..., L, \text{ and } \boldsymbol{a}_t{}^0 = \boldsymbol{x}_t$$

Relax constraints and now amenable to alternating minimization

Relaxed

$$\min_{\boldsymbol{W},C} \quad \sum_{t=1}^{n} \mathcal{L}(y_t, \sigma_L(\boldsymbol{c}_t^L), \boldsymbol{W}^{L+1}) \quad + \quad \mu \sum_{t=1}^{n} \sum_{l=1}^{L} \|\boldsymbol{c}_t^l - \boldsymbol{W}^l \sigma_{l-1}(\boldsymbol{c}_t^{l-1})\|_2^2$$

ONLINE ALTERNATING MINIMIZATION

Offline algorithms of prior works are not scalable to extremely large datasets and not suitable for incremental, continual/lifelong learning, hence …

1: while more samples do
2: Input (x_t, y_t)
3: $C \leftarrow \text{encodeInput}(x_t, W_{t-1})$ **Forward: compute linear activations at layers 1,…,L**
4: $C \leftarrow \text{updateCodes}(C, y_t, W_{t-1}, \mu)$ **Backward: error propagation by code changes**
5: $W_t \leftarrow \text{updateWeights}(W_{t-1}, x_t, y_t, C, \mu, \eta, Mem)$ **Parallelizable**
6: end while
7: return W_t

Note: **updateWeights** has two options: Apply SGD to the current mini-batch or apply BCD to version that includes memory of previous samples using the following (via Mairal et al., 2009):

$$\sum_{i=1}^{t} ||c_i^l - W a_i^l||_2^2 = Tr(W^T W A_t^l) - 2Tr(W^T B_t^l)$$

Algorithm 2.1 Online Alternating Minimization (AM)

Require: $(\boldsymbol{x}, \boldsymbol{y}) \sim p(\boldsymbol{x}, y)$ (data stream sampled from distribution $p(\boldsymbol{x}, \boldsymbol{y})$; initial weights $\boldsymbol{W}_0$; $\mu \in \mathbb{R}^+$ (quadratic penalty weight); $\eta \in \mathbb{R}^+$ (top-layer weight update step size); Mem (indicates the type of optimization method for **updateWeights**; if "yes", input initial memory matrices $\boldsymbol{A_0}$ and $\boldsymbol{B_0}$).

1: **while** more samples **do**
2: Input $(\boldsymbol{x}_t, y_t)$
3: $\boldsymbol{C} \leftarrow$ **encodeInput**($\boldsymbol{x}_t$, $\boldsymbol{W}_{t-1}$) % forward: compute linear activations at layers 1, ..., L
4: $\boldsymbol{C} \leftarrow$ **updateCodes**($\boldsymbol{C}$, y_t, $\boldsymbol{W}_{t-1}$, μ) % backward: error propagation by activation (code) changes
5: $\boldsymbol{W}_t \leftarrow$ **updateWeights**($\boldsymbol{W}_{t-1}$, $\boldsymbol{x}_t$, y_t, $\boldsymbol{C}$, μ, η, Mem)
6: **end while**
7: **return** $\boldsymbol{W}_t$

Algorithm 2.2 Activation Propagation (Code Update) Steps

encodeInput($\boldsymbol{x}$, $\boldsymbol{W}$)

1: $\boldsymbol{c}^0 = \boldsymbol{x}$
2: **for** $l = 1$ to L **do**
3: $\quad \boldsymbol{c}^l = \boldsymbol{W}^l \sigma_{l-1}(\boldsymbol{c}^{l-1})$
$\quad \%\ \sigma_0(\boldsymbol{x}) = \boldsymbol{x}, \sigma_l(\boldsymbol{x}) = ReLU(\boldsymbol{x})\ for\ l = 1, ..., L$
4: **end for**
5: **return** $\boldsymbol{C}$

updateCodes($\boldsymbol{C}$, $\boldsymbol{y}$, $\boldsymbol{W}$, λ_C, μ)

1: $\boldsymbol{c}^L \leftarrow$ Solve Problem (4), $\quad \boldsymbol{c}^0 = \boldsymbol{x}$
2: **for** $l = L - 1$ to 1 **do**
3: $\quad \boldsymbol{c}^l \leftarrow$ Solve Problem (5)
4: **end for**
5: **return** $\boldsymbol{C}$

Algorithm 2.3 Weight and Memory Update Steps

updateWeights($\boldsymbol{W}$, $\boldsymbol{x}$, y, $\boldsymbol{C}$, μ, η, Mem)

$\boldsymbol{W}^{L+1} = \boldsymbol{W}^{\boldsymbol{L+1}} - \eta \nabla_{\boldsymbol{W}} \mathcal{L}(y, \sigma_L(\boldsymbol{c}^L), \boldsymbol{W}^{\boldsymbol{L+1}})$
for $l = 1$ to L **do**
  if Mem **then**
    $(\boldsymbol{A^l}_t, \boldsymbol{B^l}_t) \leftarrow$ **updateMemory**($\boldsymbol{A^l}_{t-1}$, $\boldsymbol{B^l}_{t-1}$, $\boldsymbol{C^l}$)
    % $\hat{f}^l(\boldsymbol{W^l}) \equiv Tr(\boldsymbol{W}^T \boldsymbol{W} \boldsymbol{A}^l) - 2Tr(\boldsymbol{W}^T \boldsymbol{B}^l)$
    $\boldsymbol{W}^l = \arg\min_W \hat{f}^l(\boldsymbol{W})$
  else
    %(parallel) local update of each layer weights,
    %(*independently of other layers (unlike backprop)*
    $\boldsymbol{W}^l \leftarrow$ **SGD**($\boldsymbol{W^l}$, $\boldsymbol{x}$, y, $\boldsymbol{C^l}$, μ, η)
  end if
end for
return $\boldsymbol{W}$

updateMemory($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$)

for $l = 1$ to L **do**
  $\boldsymbol{a} = \sigma_{l-1}(\boldsymbol{c}^{l-1})$, $\boldsymbol{A}^l \leftarrow \boldsymbol{A}^l + \boldsymbol{a}\boldsymbol{a}^T$, $\boldsymbol{B}^l \leftarrow \boldsymbol{B}^l + \boldsymbol{c}^l \boldsymbol{a}^T$
end for
return $\boldsymbol{A}, \boldsymbol{B}$

NOVEL THEORETIC RESULT FOR STOCHASTIC ALTMIN

Theorem 3.2. *Given the stochastic AM gradient iterates of the version of Algorithm 2.1 given in eq. 8 with decaying step size* $\eta^t = \frac{3/2}{[2\xi - 3\gamma(K-1)](t+2) + \frac{3}{2}(K-1)\gamma}$ *and assuming that* $\gamma < \frac{2\xi}{3(K-1)}$, *the error at iteration* $t+1$ *satisfies*

$$\mathbb{E}\left[\sum_{d=1}^{K} \|\boldsymbol{\Delta}_d^{t+1}\|_2^2\right] \leq \mathbb{E}\left[\sum_{d=1}^{K} \|\boldsymbol{\Delta}_d^{0}\|_2^2\right] \left(\frac{2}{t+3}\right)^{\frac{3}{2}} + \sigma^2 \frac{9}{[2\xi - 3\gamma(K-1)]^2 (t+3)}, \quad (10)$$

where $\boldsymbol{\Delta}_d^{t+1} := \boldsymbol{\theta}_d^{t+1} - \boldsymbol{\theta}_d^{*}$ *for* $d = 1, 2, \ldots, K$, $\gamma := \max_{i=1,2,\ldots,K} \gamma_i$, *and* $\xi := \min_{i=1,2,\ldots,K} \frac{2\mu_i \lambda_i}{\mu_i + \lambda_i}$.

FULLY-CONNECTED NETS

AM greatly **outperforms all off-line methods** (ADMM of Taylor et al, and offline AM), and often matches Adam and SGD (50 epochs)

MNIST

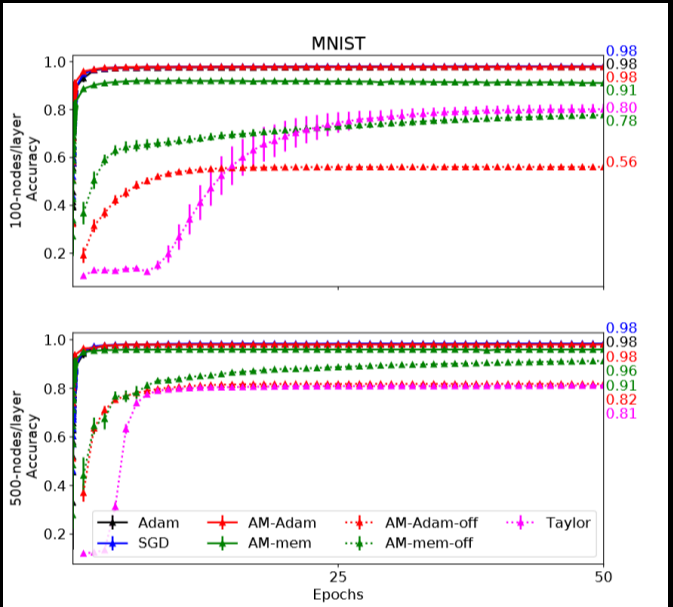


Figure 2. MNIST (fully-connected nets, 2 layers): online vs. offline methods vs. Taylor's ADMM, 50 epochs.

CIFAR-10

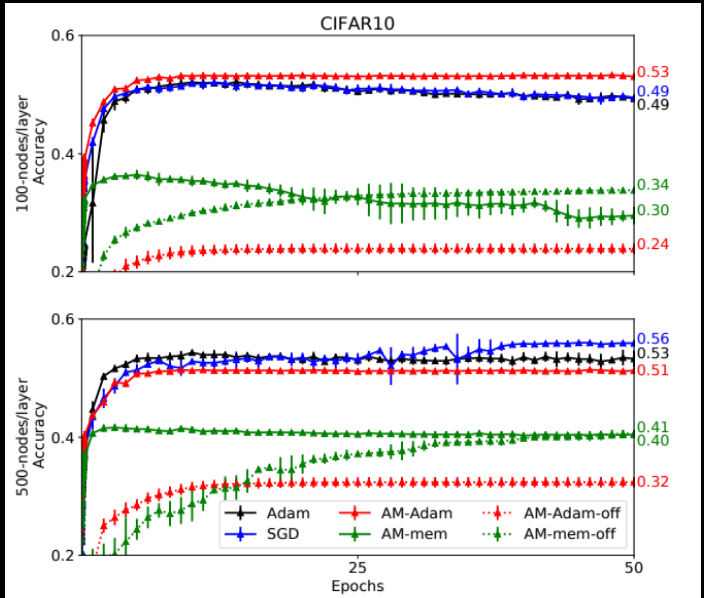


Figure 4. CIFAR10 (fully-connected networks): online vs. offline, 50 epochs. Similar experiments to Figure 2.

FASTER INITIAL LEARNING: POTENTIAL USE AS A GOOD INIT?

- AM often learns faster than SGD & Adam (backprop-based) in the 1st epoch, then matches their performance

MNIST

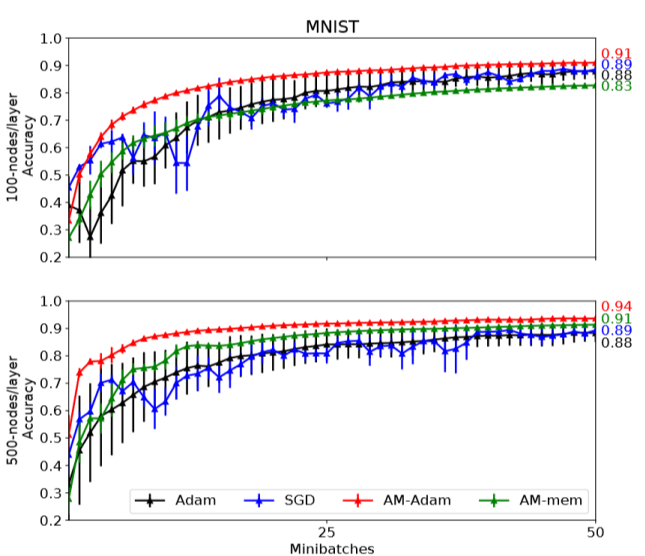


Figure 1. MNIST (fully-connected nets, 2 layers): online methods, first epoch; 50 mini-batches, 200 samples each.

CIFAR-10

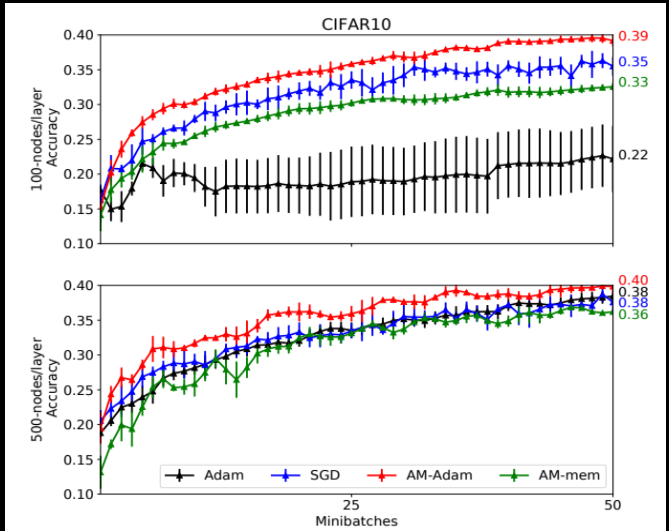


Figure 3. CIFAR10 (fully-connected nets): online methods, 1st epoch. 2 hidden layers with 100 (top) and 500 (bottom) units each; 250 mini-batches, 200 samples each.

CONVNETS: LENET5, MNIST

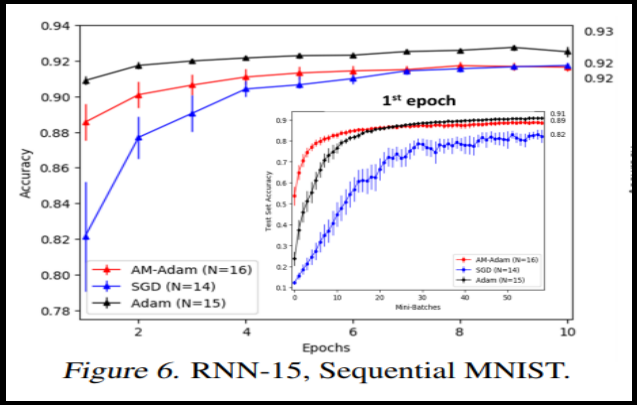


Figure 6. RNN-15, Sequential MNIST.

RNN: SEQUENTIAL MNIST

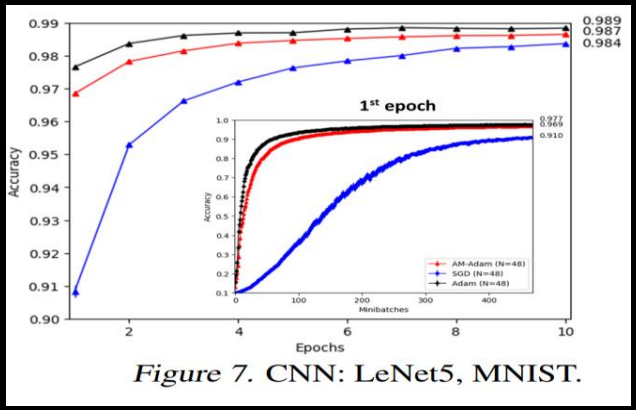


Figure 7. CNN: LeNet5, MNIST.

HIGGS DATASET, FULLY-CONNECTED

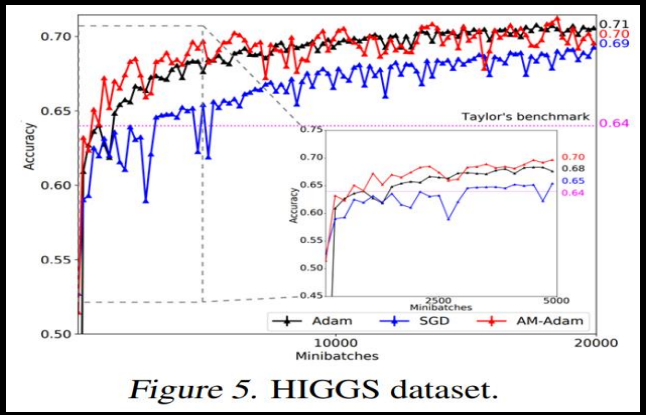


Figure 5. HIGGS dataset.

- AM performs similarly to Adam, outperforms SGD
- All methods greatly outperform offline ADMM (Taylor's 0.64 benchmark) using less than 0.01% of 10.5M-sample HIGGS data

NONDIFFERENTIABLE (BINARY) NETS

- Backprop replaced by Straight-Through Estimator (STE)
- Comparing with Difference Target Propagation (DTP)
- DTP took about 200 epochs to reach 0.2 error, matching the STE performance (Lee et al., 2015)
- AM-Adam with binary activations reaches same error in < than 20 epochs

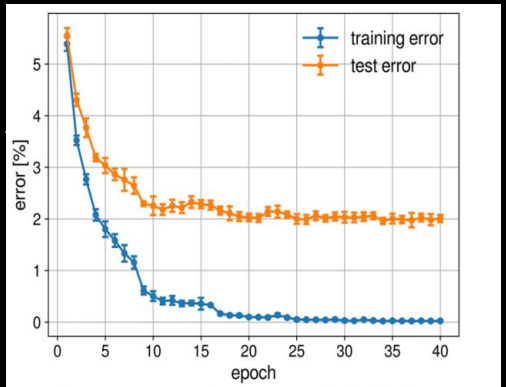


Figure 8. Binary net, MNIST.

SUMMARY: CONTRIBUTIONS

- **Algorithm(s):** novel online (stochastic) auxiliary-variable approach for training neural networks (prior methods are offline/batch); two versions of the approach (memory-based and local-SGD-based)

- **Theory:** first general theoretical convergence guarantees for alternating minimization in the stochastic setting: the error decays at the sub-linear rate $O((1/t)^{3/2} + 1/t)$ in t iterations

- **Extensive Evaluations:** variety of architectures and datasets demonstrating advantages of online vs offline approaches and performance similar to SGD (Adam), with faster initial convergence