Learning to Learn Second-Order Back-Propagation for CNNs Using LSTMs

Anirban Roy[*] and Sinisa Todorovic[#]
[*]SRI International
[#]Oregon State University

ICPR 2018

Problem: Learning CNN Parameters

Iterative updates: $$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} + \Delta\boldsymbol{\theta}^{t}$$

Current parameters   Parameter update

Learning a CNN:

$\boldsymbol{\theta}$

Loss on the output

Backpropagate gradients of

the loss function to estimate $\Delta\boldsymbol{\theta}^{t}$

Problem: Learning CNN Parameters

Iterative updates: $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \Delta\boldsymbol{\theta}^t$

Current parameters Parameter update

<u>Standard approach</u>:

Given the loss function , estimate $\Delta\boldsymbol{\theta}^t$ using:

- Stochastic Gradient Descent (SGD)
- Other heuristics, e.g., learning rate, momentum.

Problem: Learning CNN Parameters

Iterative updates: $$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} + \Delta\boldsymbol{\theta}^{t}$$

Current parameters

Parameter update

<u>Our goal</u>:

- Improve the convergence rate
- Eliminate hand-tuning of heuristics parameters

Little Theoretical Understanding

- The loss function is **highly non-convex**
- Why does SGD even converge?
- SGD $\Leftrightarrow$ Regular gradient descent on a convolved smoothed loss function. [Kleinberg 2018, Chaudhari & Soatto 2018]
- Convolution kernel size grows with $\dfrac{\textbf{mini-batch size}}{\textbf{learning rate}}$

Prior Work: Heuristics for Faster Convergence

Learning rate is adaptively adjusted based on:

- **RMSProp**: Current magnitude of gradients [Tieleman & Hinton 2012]
- **ADAM**: Magnitude of current and past gradients [Kingma & Ba 2014]

Prior work: Learning the Gradients

- **Learning gradient descent**:

Updates $\Delta\boldsymbol{\theta}^t$ are estimated from a history of the gradients using an LSTM

[Schmidhuber 1993; Thrun & Pratt 2012; Andrychowicz et. al. 2016]

- **Learning an update policy**:

Updates $\Delta\boldsymbol{\theta}^t$ are estimated via reinforcement learning

[Li & Malik 2016]

Our Motivation 1: Recent Findings

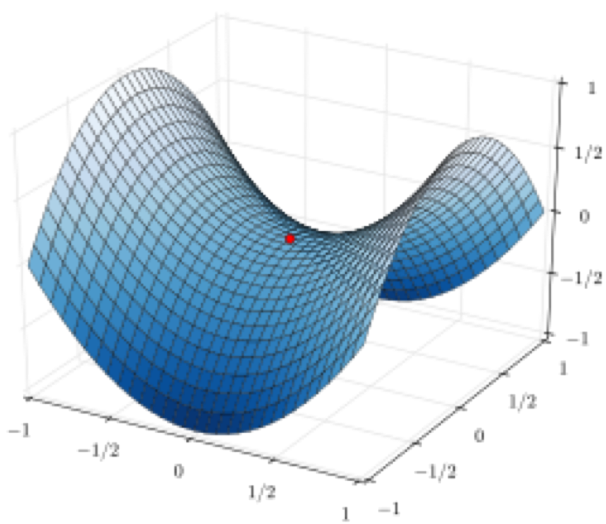
- Loss function has numerous “plateau” regions with near-zero gradient values.
  [Dauphin et al. 2014; Choromanska et al. 2015; Kawaguchi 2016]

- $\Rightarrow$ We need second-order optimization

Our Motivation 2: Recent Findings

- A zero-gradient point is more likely to be a saddle point than a local minimum.

  [Dauphin et al. 2014; Choromanska et al. 2015; Kawaguchi 2016]



Our Motivation 2: Recent Findings

- $\Rightarrow$ SGD has slow convergence due to frequent "passes" through the **"plateau" regions**

- $\Rightarrow$ We need second-order optimization

Our Goal

Iterative updates: $$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} + \Delta\boldsymbol{\theta}^{t}$$

Current parameters Parameter update

Estimate $\Delta\boldsymbol{\theta}^{t}$ based on the gradient + Hessian

<u>Challenge</u>: Computing the Hessian !

Prior work: Second-order Optimization

Approximate the Hessian to compute the updates $\Delta\boldsymbol{\theta}^t$

⇒ Helps navigate faster through the ``plateaus”

[Buntine 1994; Martens 2010; Chapelle & Erhan 2011; Dauphin 2014; Choromanska 2015; Kawaguchi 2016, Henriques 2018]

Our Approach: Second-order Updates

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \alpha^{t} (\mathbf{H}^{t})^{-1} \boldsymbol{g}^{t}$$

Learning rate Hessian gradient

Our Key Idea

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - \boxed{\alpha^{t}(\mathbf{H}^{t})^{-1}\boldsymbol{g}^{t}} = u^{t}$$

Update vector

Learning rate

Hessian

gradient

Our Key Idea

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} + u^{t}$$

Estimate the update vector using LSTM

$$[u^{t}, \boldsymbol{x}^{t+1}] = LSTM(\boldsymbol{g}^{t}, \boldsymbol{h}^{t}, \boldsymbol{x}^{t})$$

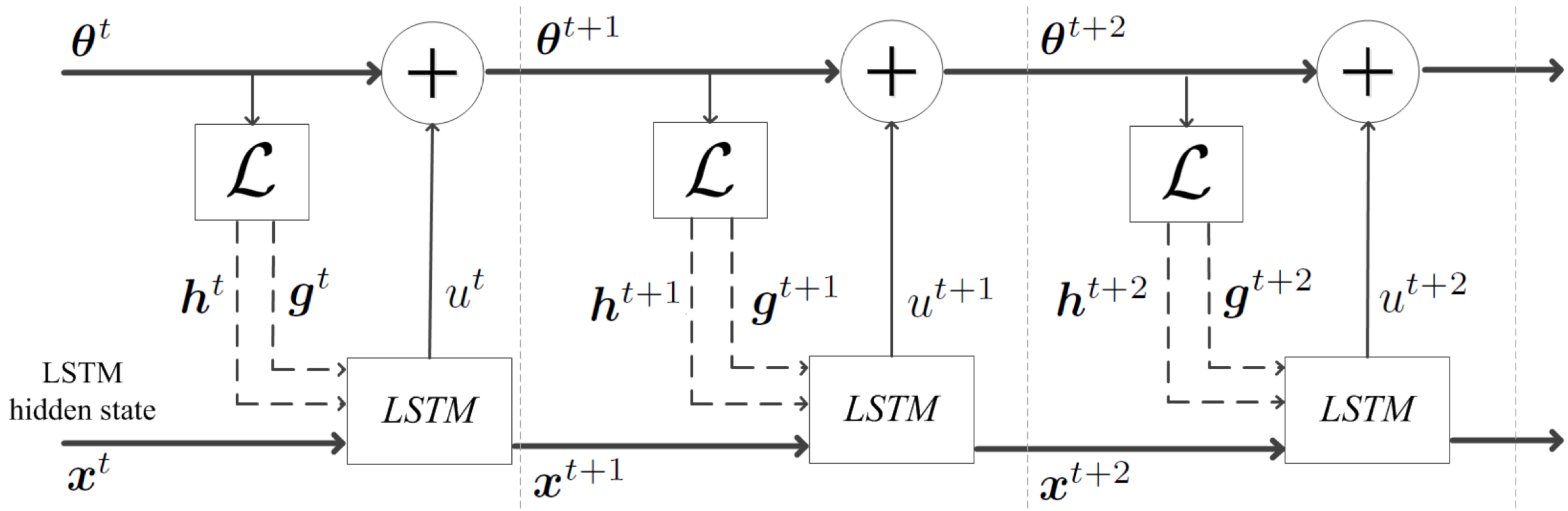
LSTM hidden state

Gradients

Second derivatives

LSTM hidden state

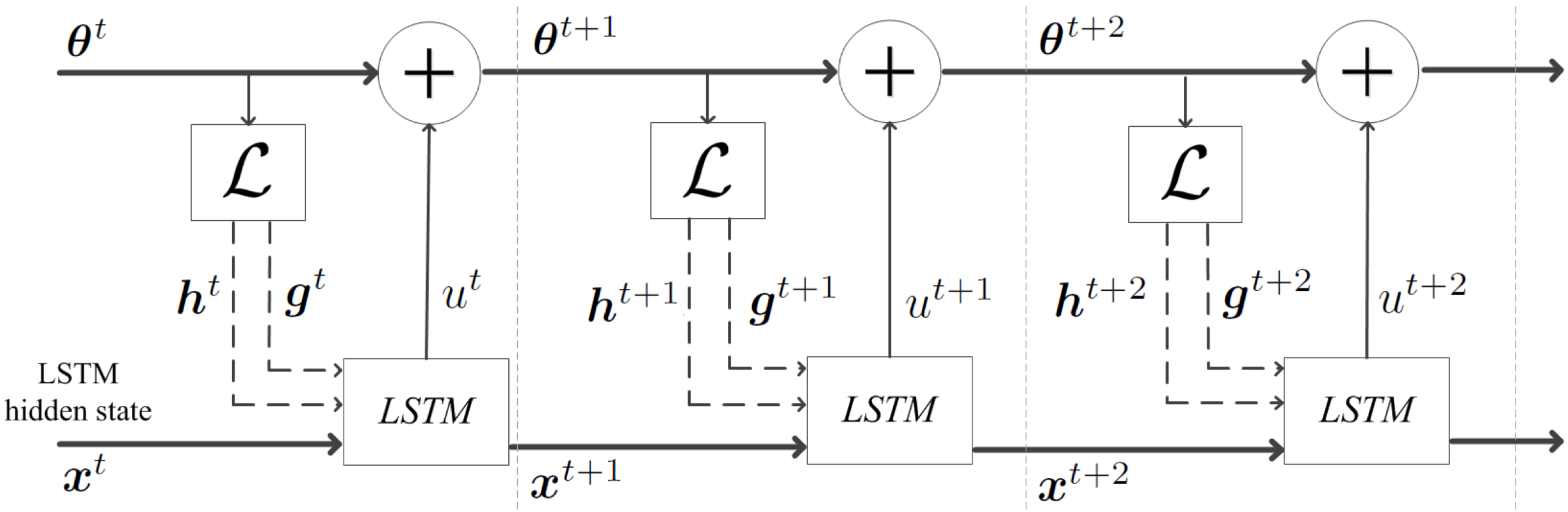
Our Approach



<u>Input to LSTM</u>: Current gradient and second derivatives

<u>Output of LSTM</u>: Update vector

Joint Learning of CNN and LSTM



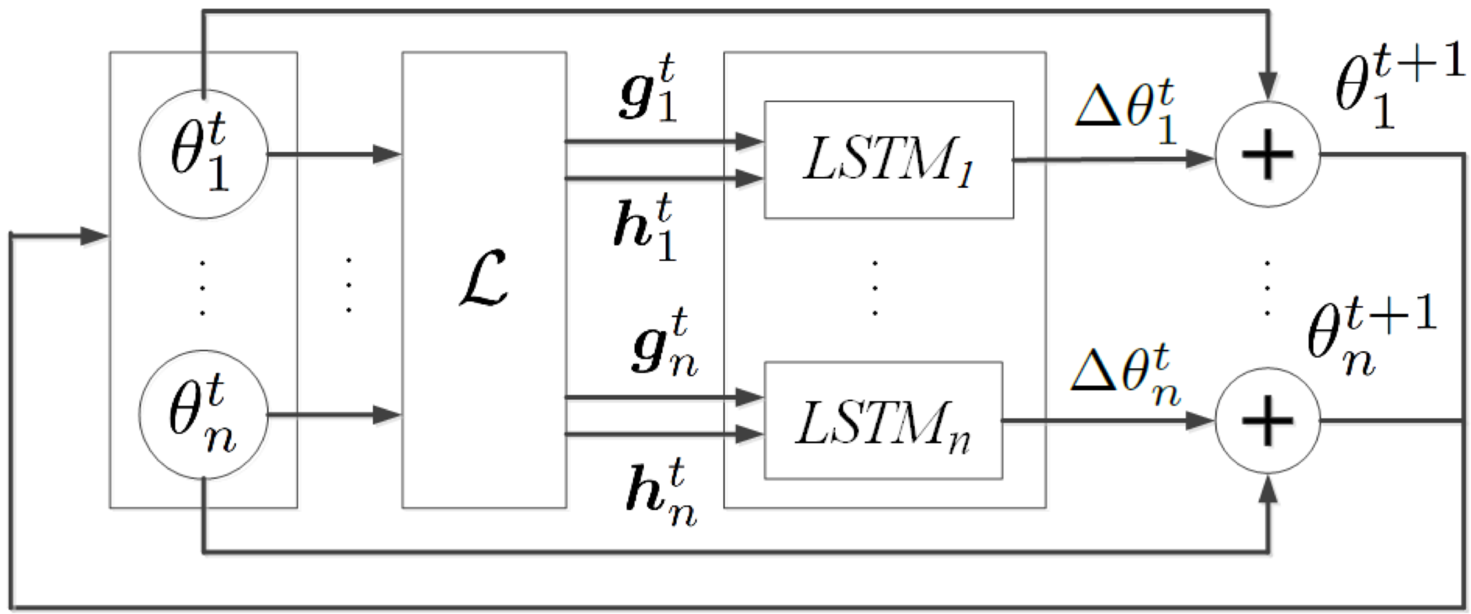
- Both LSTM and CNN are learned via back-propagation through time (BPTT).

- Details are presented in the paper.

Details

- Learn a separate LSTM for each layer of CNN
- Separate LSTMs for convolutional and fully connected layers
- Each LSTM has two hidden layers with 20 hidden units in each layer

Our approach



Learning step: $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \Delta\boldsymbol{\theta}^t(\boldsymbol{g}^t, \boldsymbol{h}^t)$

Updated parameters Current parameters Gradients Hessian

Results

Datasets

- MNIST [LeCun et al. 1998]:
  - 28x28 images of the 10 handwritten digits
  - 60,000 training images and 10,000 test images
- CIFAR-10 [Krizhevsky & Hinton 2009]:
  - 32x32 color images of 10 classes
  - 50,000 training images and 10,000 test images
- ImageNet [Deng et al. 2009]:
  - Color images of 1000 object classes
  - 1.2 million training images and 10,000 test images

Baselines

- **SGD** : a vanilla SGD to update parameters.

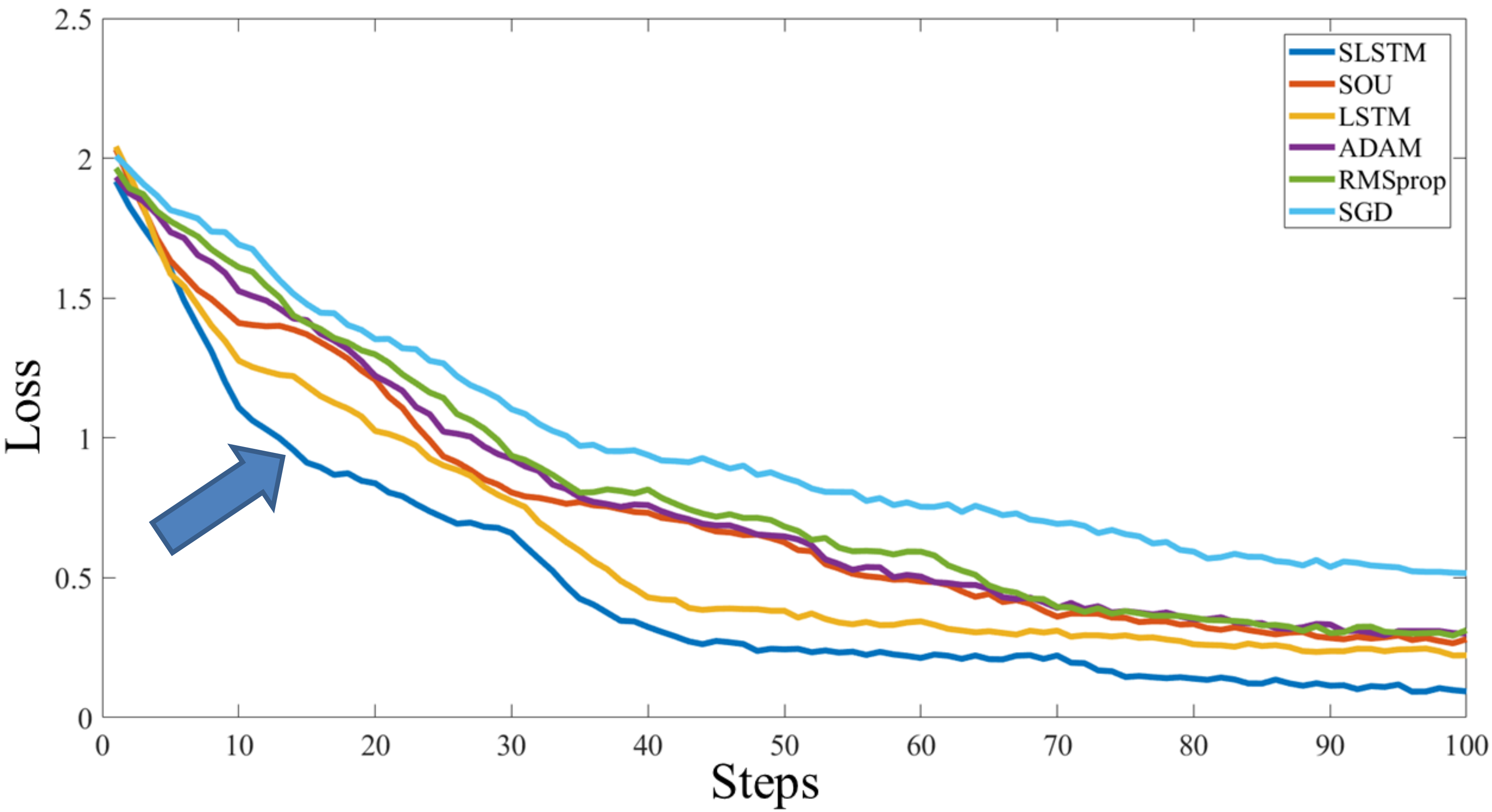
- **RMSprop** [Tieleman & Hinton, 2012]:
  Estimate the learning rate using current gradients.

- **ADAM** [Kingma & Ba, 2014]:
  Estimate the learning rate using current & past gradients.

- **LSTM** [Andrychowicz et. al., 2016]:
  LSTM estimates the update u^t from gradients only.

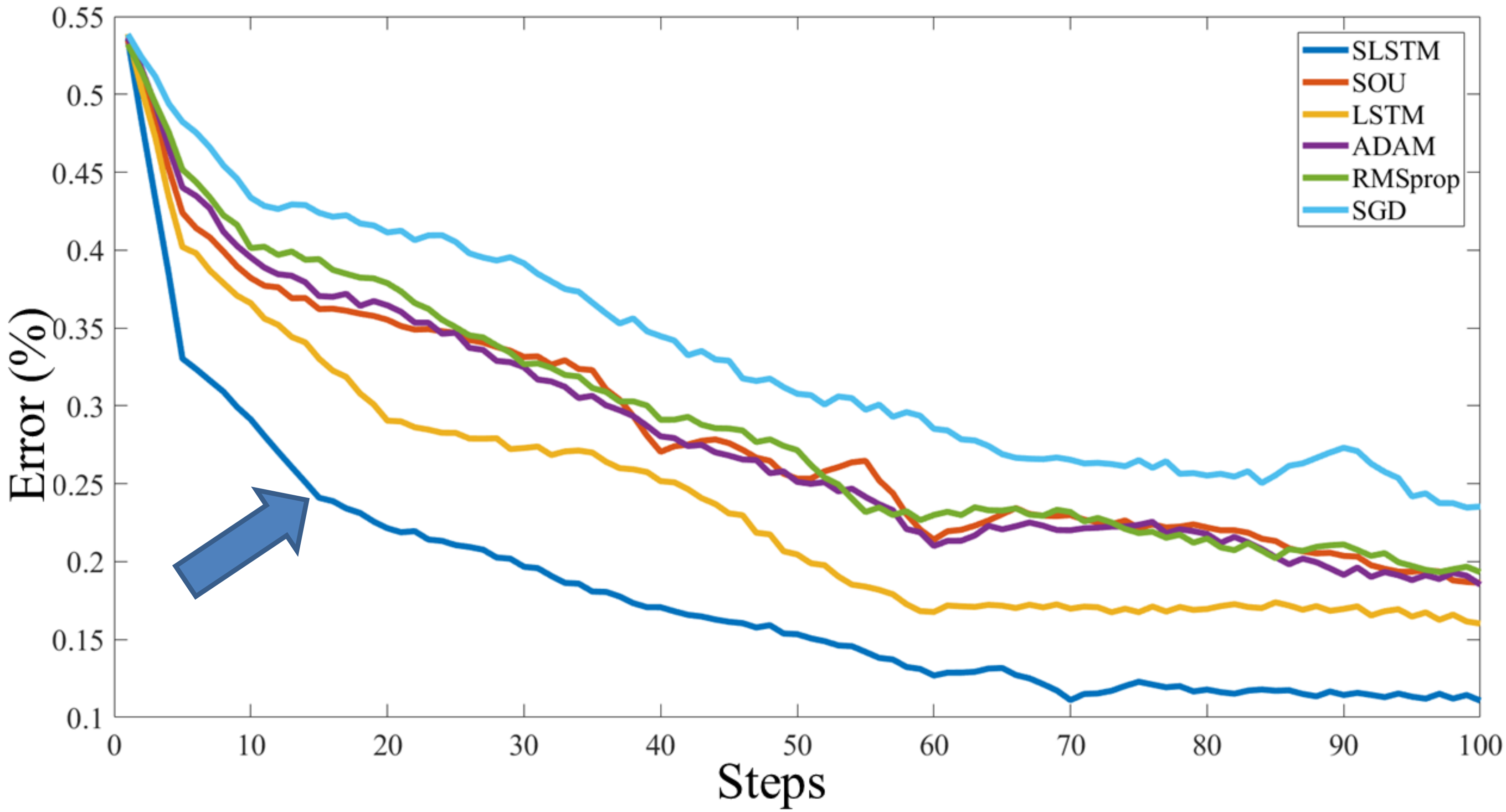
- **Second order updates (SOU)**: $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha^t (\mathbf{H}^t)^{-1} \boldsymbol{g}^t$
  Using the diagonal Hessian.

Comparison with baselines on MNIST



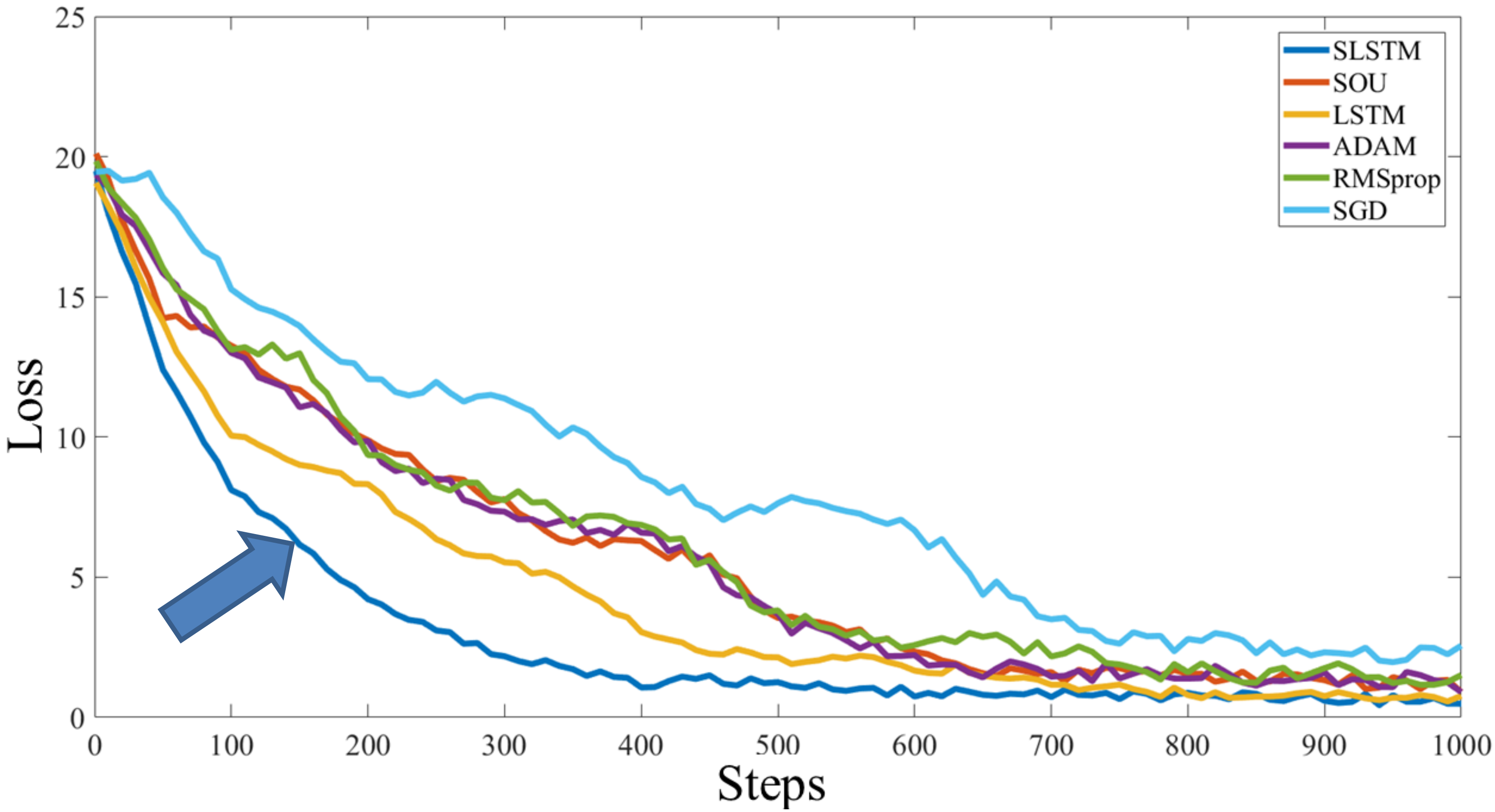
Faster convergence of our SLSTM on MNIST

Comparison with baselines on MNIST



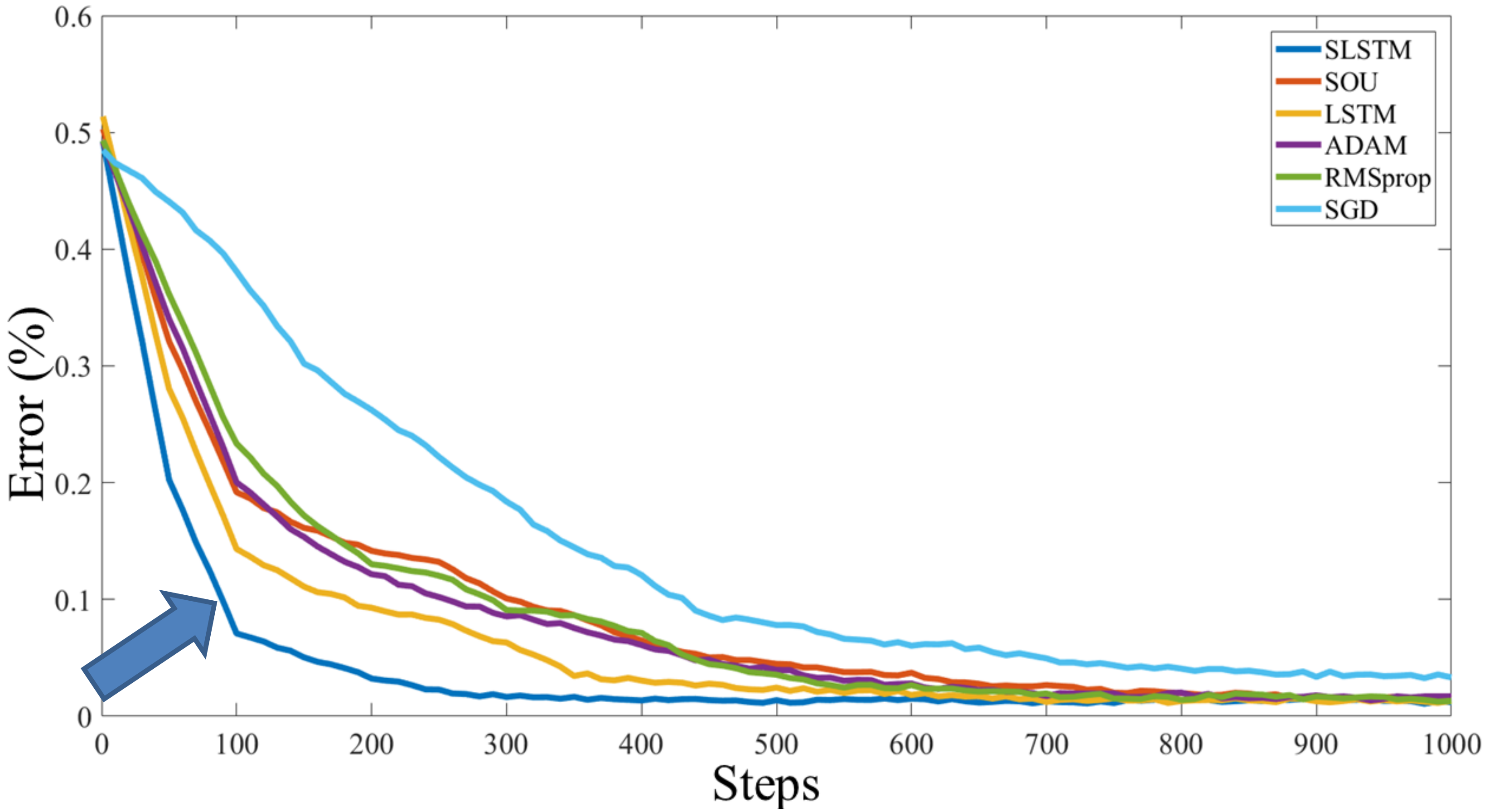
Faster decrease of test error of our SLSTM on MNIST

Comparison with baselines on CIFAR



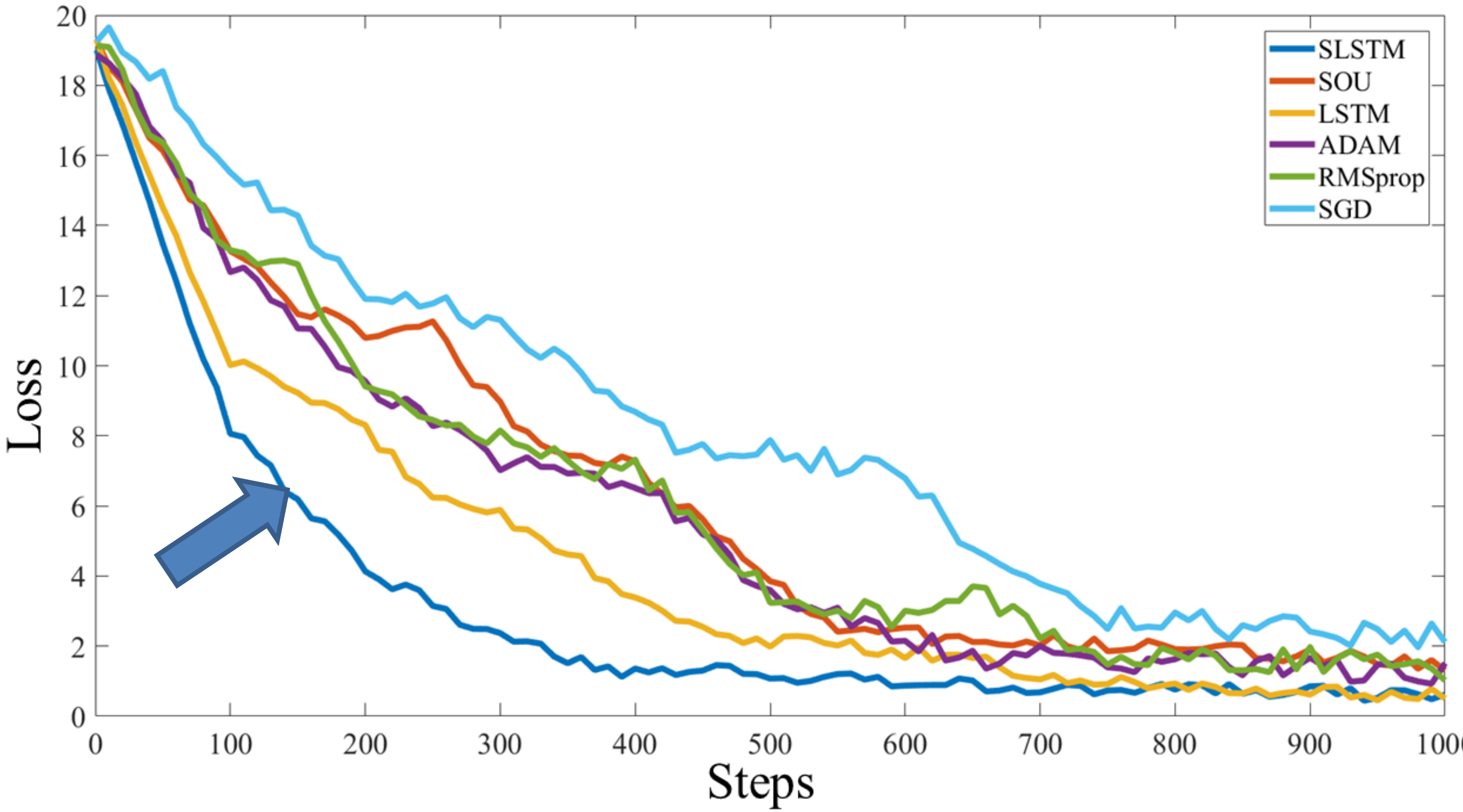
Faster convergence of our SLSTM on CIFAR

Comparison with baselines on CIFAR



Faster decrease of test error of our SLSTM on CIFAR

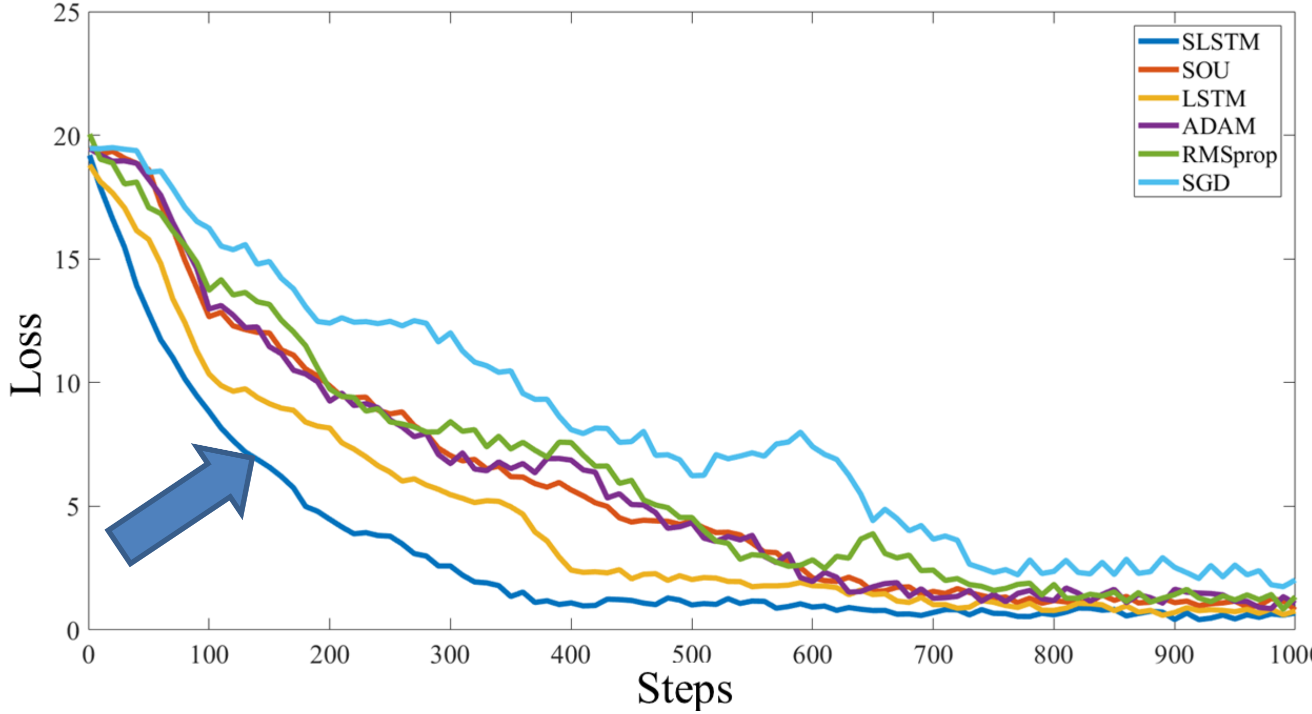
Comparison with baselines on ImageNet



Faster convergence of our SLSTM on ImageNet

Transfer learning

- Small network 1 convolutional layer and 1 fully connected layer
- Bigger network 3 convolutional layer and 2 fully connected layer



We learn LSTM with a smaller network and then use it to train a bigger network.

Conclusion

- A new meta-learning for CNNs using gradients and Hessian.
- We get faster convergence wrt heuristic optimizations on the benchmark datasets
- Learning LSTM on a small network has been successfully transferred to a learn bigger network with more layers