

Asynchronous Parallel Learning for Neural Networks and Structured Models with Dense Features

Xu SUN (孙栩)

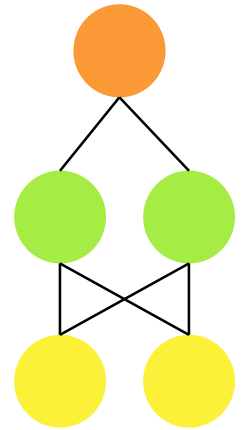
Peking University

xusun@pku.edu.cn

Motivation

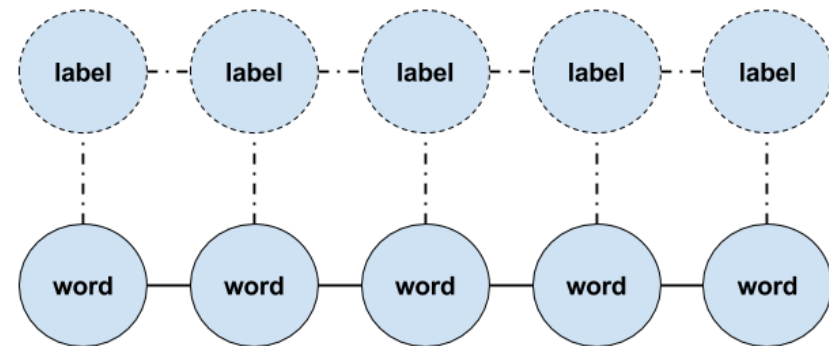
□ Neural networks -> Good Performance

- CNN, RNN, LSTM...
- Sequence labelling, parsing, machine translation...



□ Neural networks -> Slow Training

- Large parameter space
- Dense feature
- Complex computation

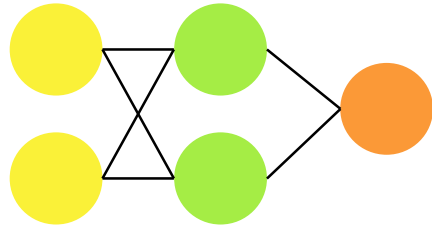


□ Faster Training ? -> Parallel Training

- Synchronous
- Asynchronous -> AsynGrad

Neural Networks

□ Many kinds

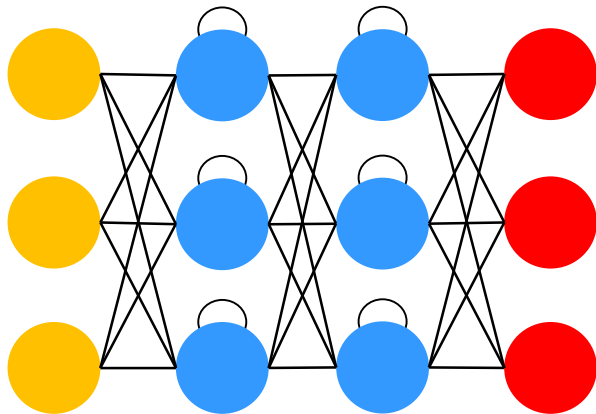
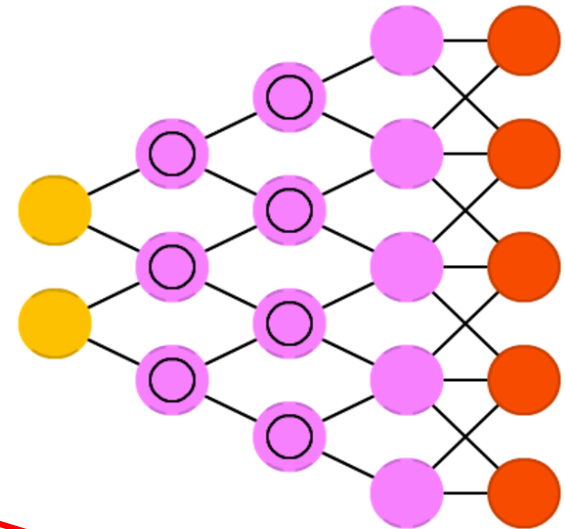


□ Feed Forward Neural Networks

- logistic regression

□ Convolutional Neural Networks

- image processing



□ Recurrent Neural Networks

- RNN, LSTM, GRU...
- structured prediction

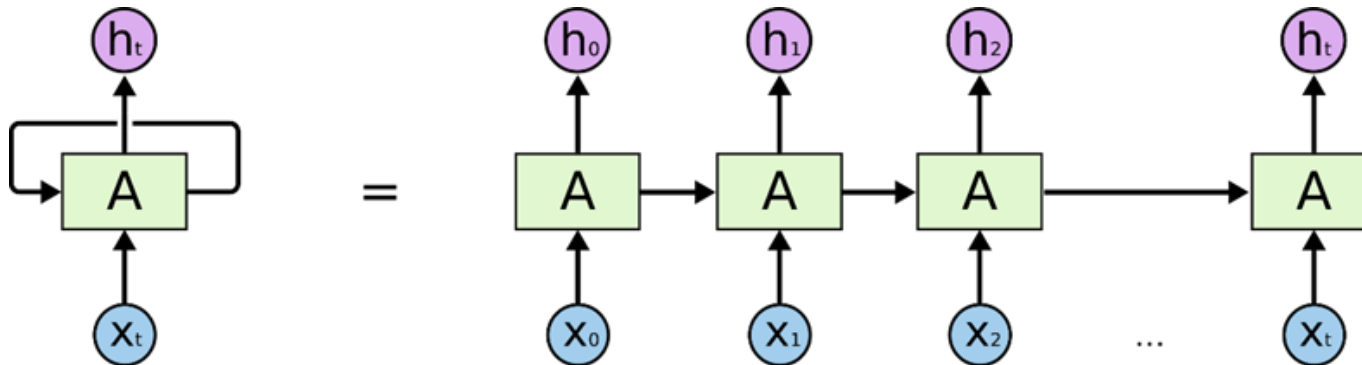
Recurrent Neural Network (RNN)

□ Recurrent neural network (Elman, Cognitive Science 1990)

- Model **time** series
- Predict **linear-chain** structures
- Conditioned on **all previous input**

$$h_t = f(Uh_{t-1} + Wx_t)$$

$$\hat{y}_t = \text{softmax}(W^{(s)}h_t)$$



Long Short-term Memory (LSTM)

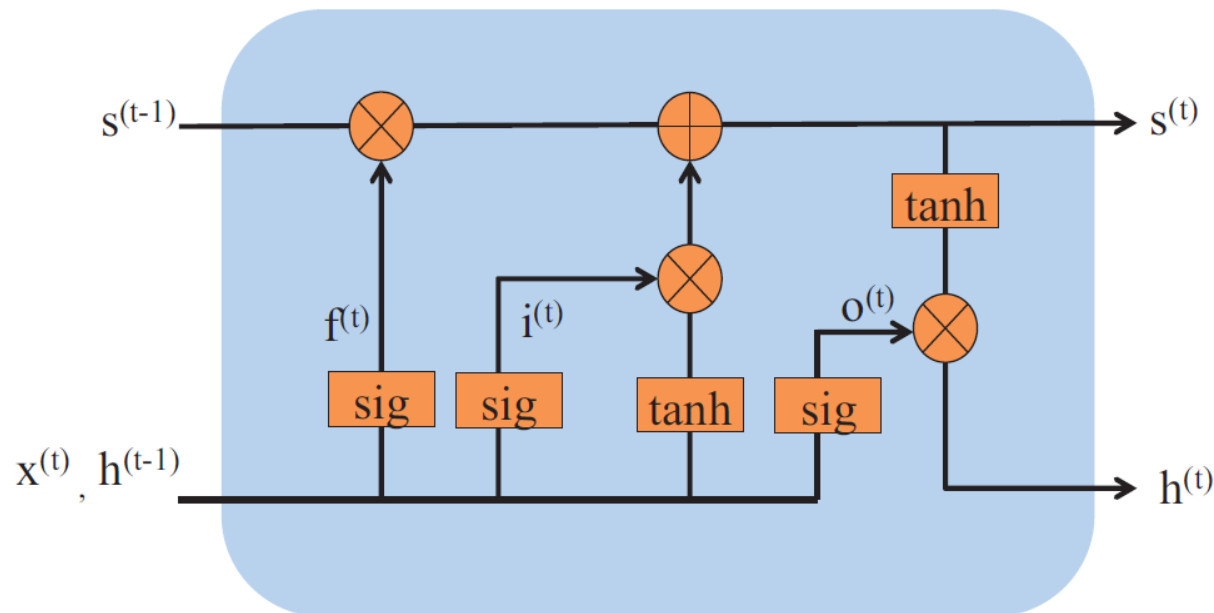
□ Long short-term memory (Hochreiter and Schmidhuber 1997)

□ A lasting linear memory

- Capture long distance dependency

□ Three gates: input, forget and output gates

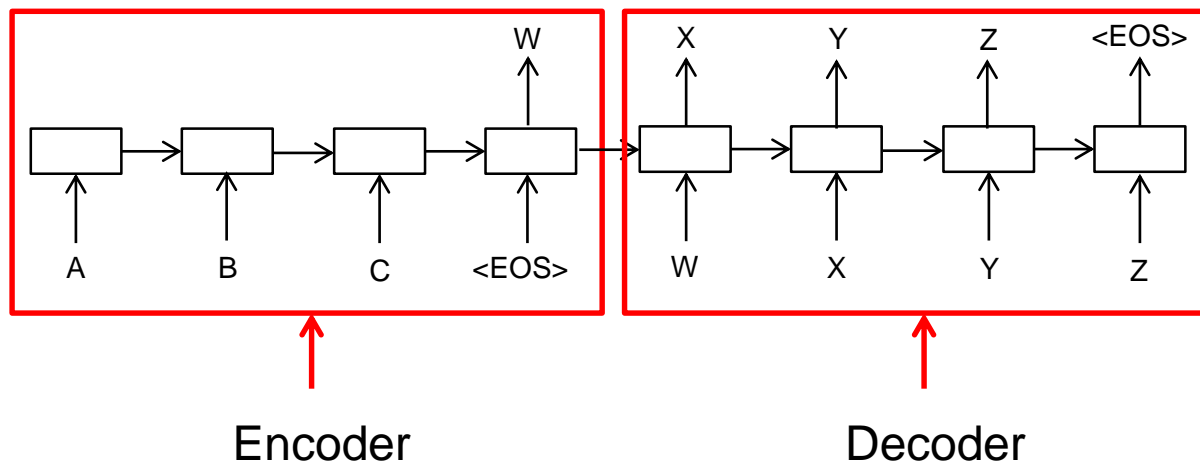
- Control modification to the memory



Sequence-to-Sequence Model

□ Sequence to sequence neural network (Sutskever et al., NIPS 2014)

- Encoder & Decoder
- The encoder information is stored in a **fixed-length vector**



□ Popular for high-level task

- Machine Translation
- Summarization
- ...

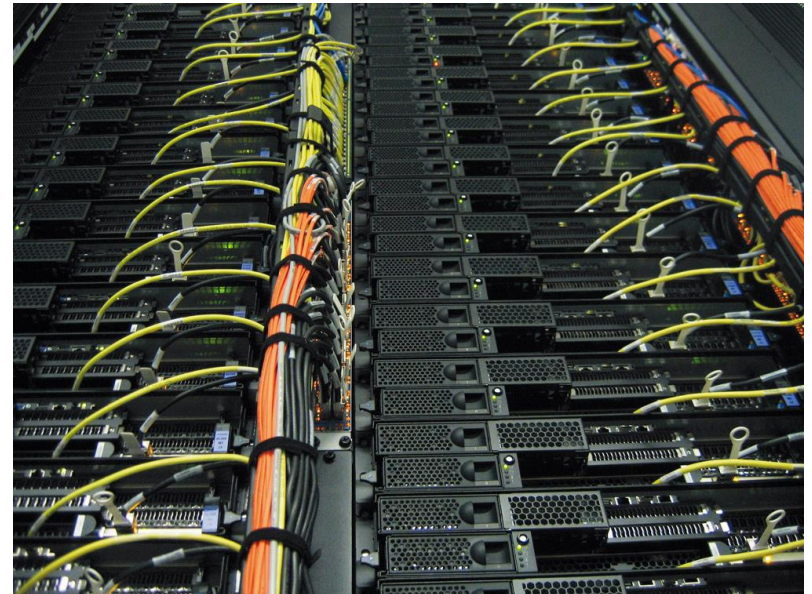
For **Large-Scale** Structured Prediction

- **Training large-scale neural networks is costly**

- Numerous parameters
- Dense Feature
- Time-consuming

- **For example**

- A NMT model may take weeks to train
- Days, even if with GPU clusters

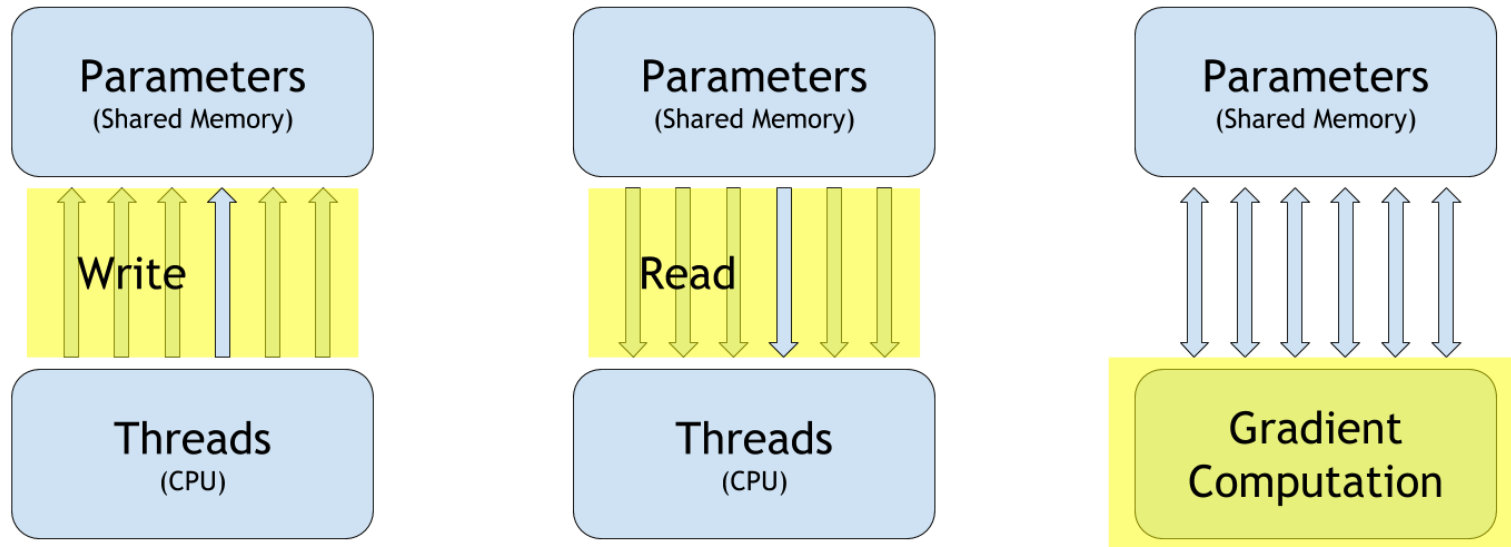


- **How to accelerate training speed?**

- **Parallel training**
- Especially, **asynchronous (lock-free) parallel training**

Problem Analysis

□ Basic operations in parallel training



□ Problem differs in

□ **Online** vs. Mini-batch vs. Batch

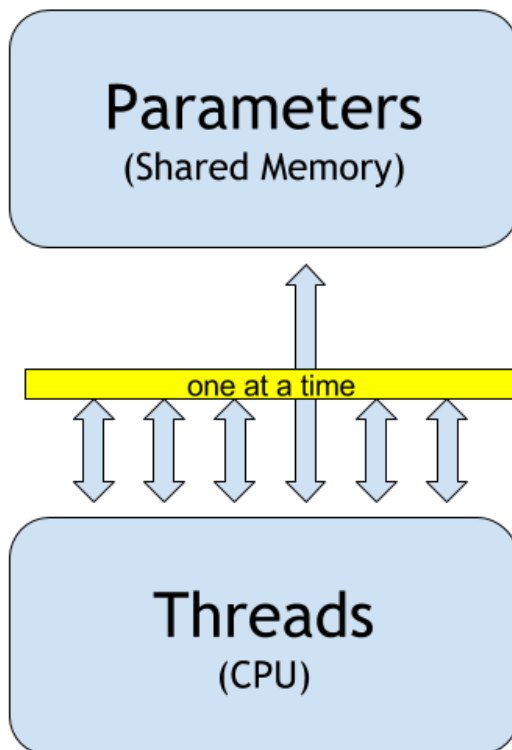
□ Synchronous parallel vs. **Asynchronous parallel**

□ **Dense feature model** vs. Sparse feature model

Parallel Training

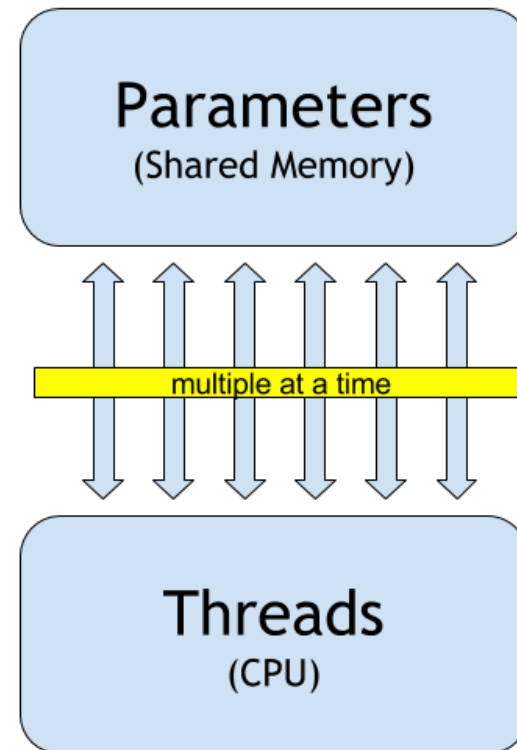
□ Synchronous (locked)

- Multiple threads
- **Only one** can modify model parameters at the same time



□ Asynchronous (lock-free)

- Multiple threads
- **Each one** can modify model parameters at the same time



Model Types

□ Sparse feature model

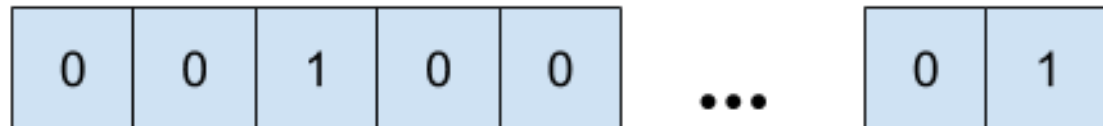
- e.g. HMM, CRF, Perceptron, MILA...
- features are sparse
- less read & write time

□ Dense feature model

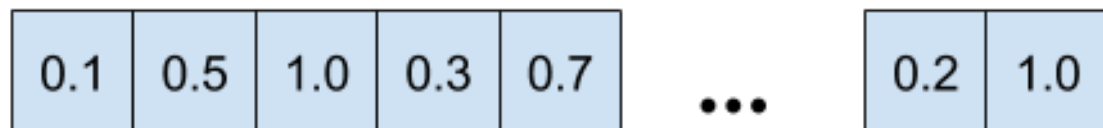
- e.g. RNN, LSTM, Sequence-to-Sequence...
- features are dense
- more read & write time

feature space

sparse feature model



dense feature model



Problem Analysis


- How threads interact with each other?

	Sparse	Dense
Sync.	?	?
Async.	?	?

Problem Analysis

- How threads interact with each other?

	Sparse	Dense
Sync.	?	?
Async.	?	?

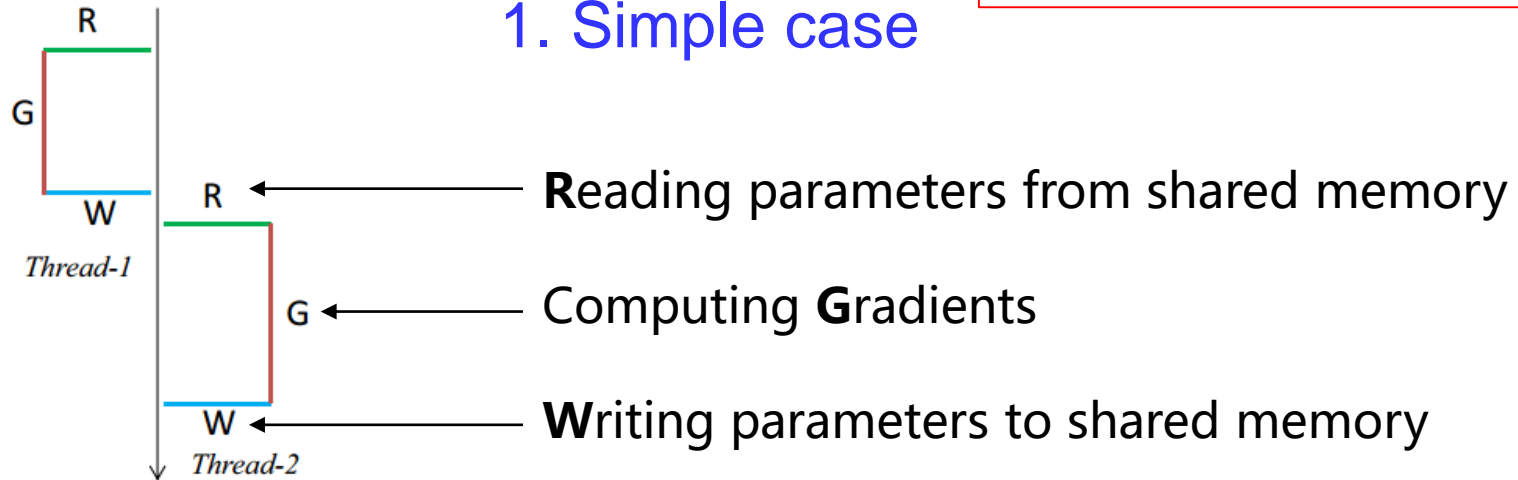


Synchronous Online Parallel Training

□ Correctness

No problem at all!

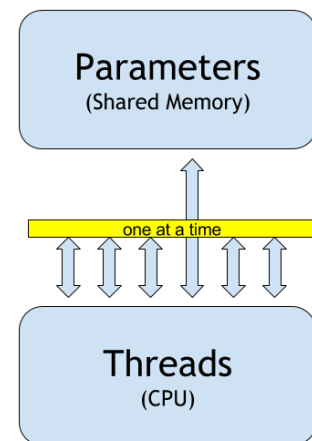
1. Simple case



(a) Simple case

□ Current approach: **DSGD** (round-robin)

- Langford et al, NIPS 2009
- Stochastic parallel learning by **locking memory**

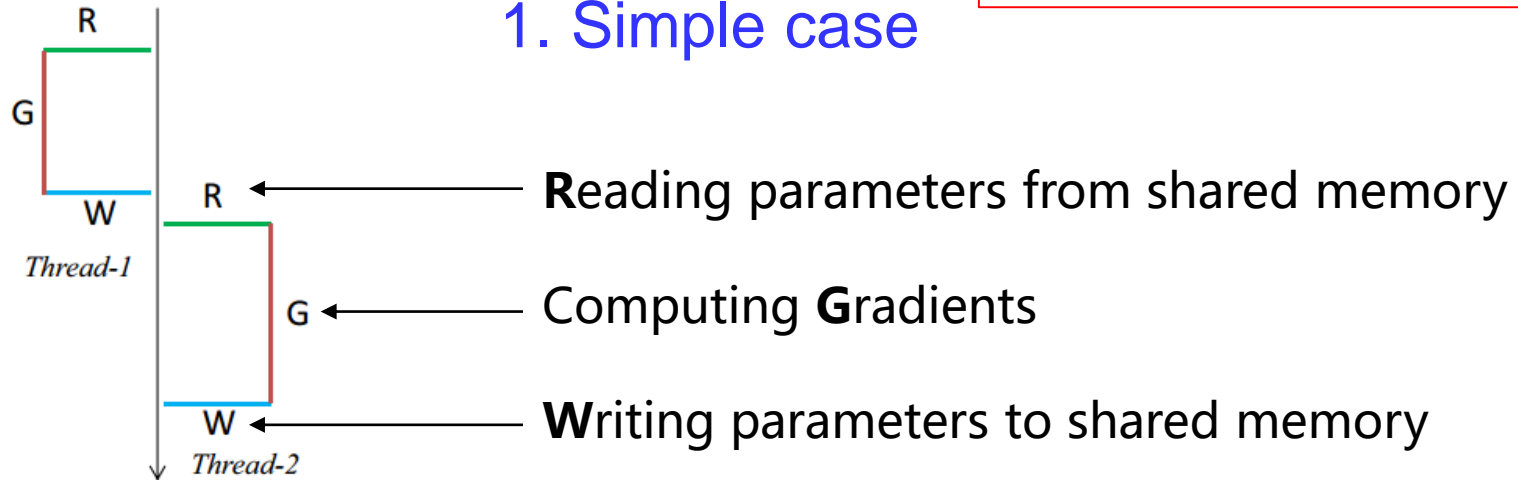


Synchronous Online Parallel Training

□ Correctness

No problem at all!

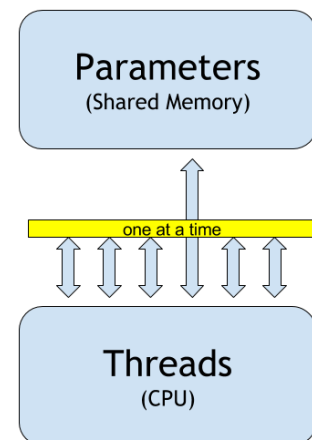
1. Simple case



(a) Simple case





□ Current approach:

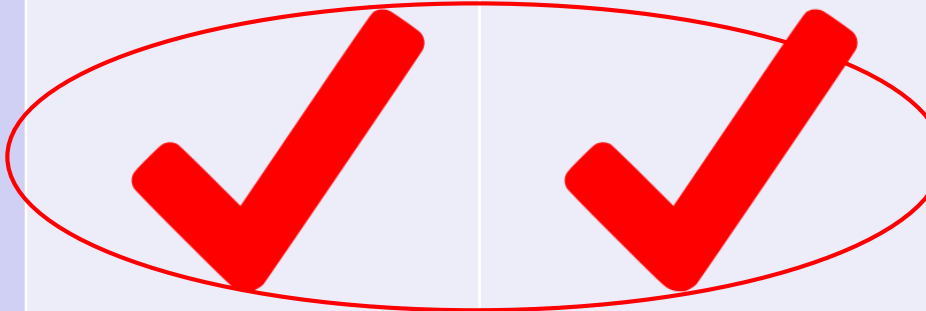
- mini-batch based method
- Computing gradients in parallel
 - such as: parallel matrix operations via GPU



Problem Analysis




- How threads interact with each other?

	Sparse	Dense
Sync.		
Async.		







Problem Analysis

- How threads interact with each other?

	Sparse	Dense
Sync.		
Async.		

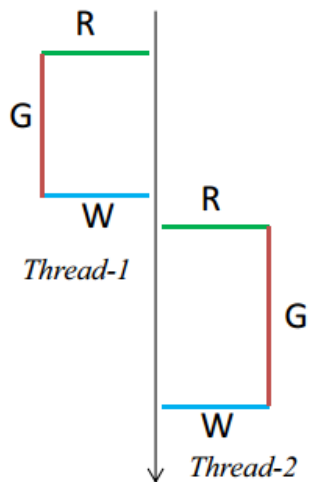
Problem Analysis

- How threads interact with each other?

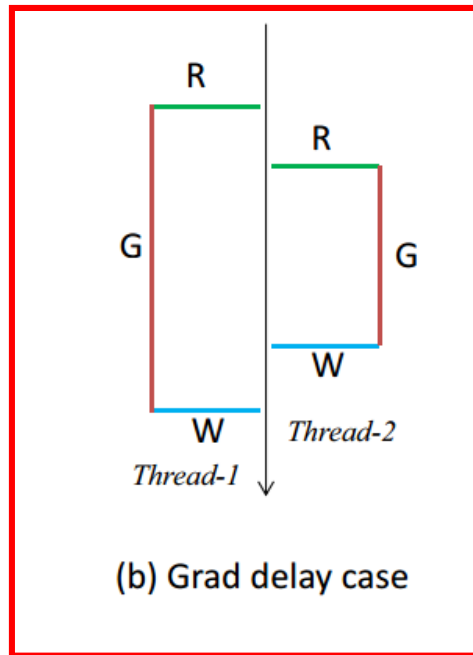
	Sparse	Dense
Sync.		
Async.		

Asynchronous Online Parallel Training

- Asynchronous parallel learning is very popular for traditional **sparse** feature models



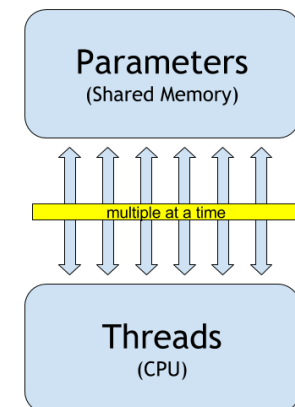
(a) Simple case



(b) Grad delay case

2. This case is called Gradient Delay case

→ More complicated, but problem solved for sparse feature models (Niu et al. NIPS 2011)



Asynchronous Online Parallel Training

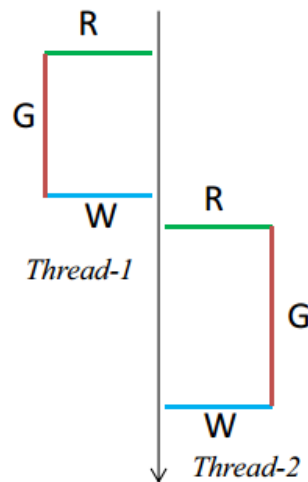
□ Current approach: **HogWild!** and variants

- Multiple threads updating parameters at the same time
- For sparse feature models

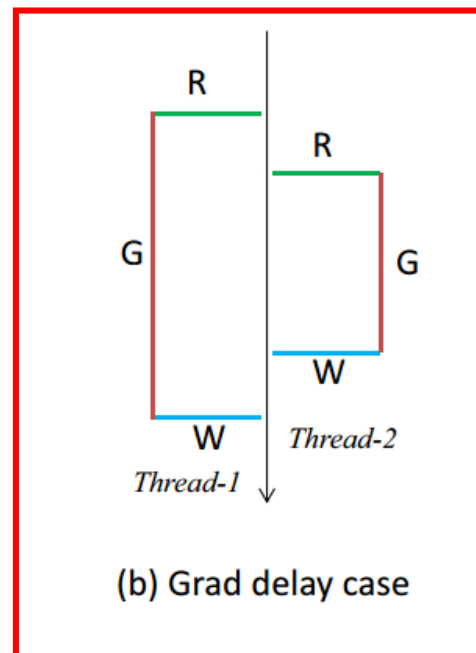
□ **Advantage**

- Actual parallel training
- Fast training speed with **no performance loss**

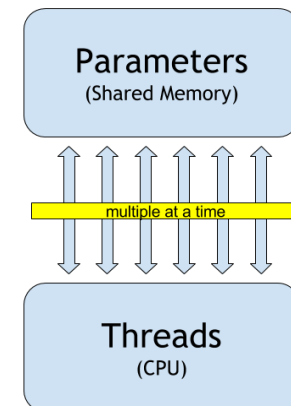
Problem also solved
(for sparse feature
models)!



(a) Simple case







(b) Grad delay case



Problem Analysis

- How threads interact with each other?

	Sparse	Dense
Sync.		
Async.		





Problem Analysis

- How threads interact with each other?

	Sparse	Dense
Sync.		
Async.		

Problem Analysis

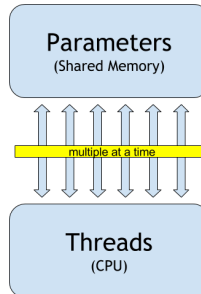
- How threads interact with each other?

	Sparse	Dense
Sync.		
Async.		

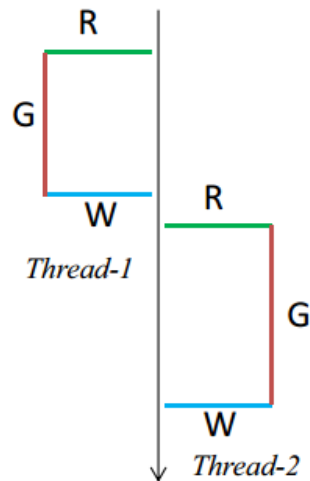
Asynchronous Online Parallel Learning

□ 3. Even more difficult case: **Gradient Error Case**

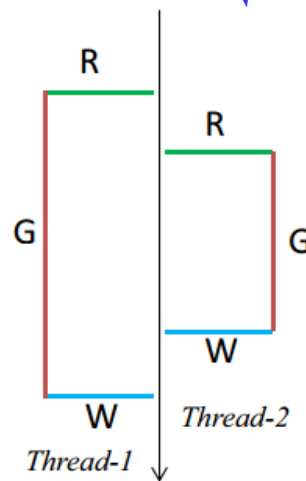
- Happens for dense feature models, like neural networks
 - Actions (R, G & W) are time-consuming
- Read-overwrite and write-overwrite problems



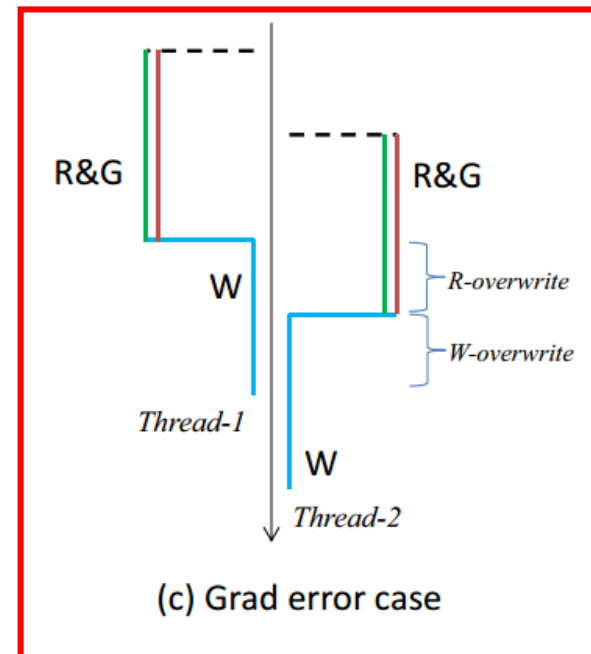
→ Not well studied before, how to deal with this problem?



(a) Simple case







(b) Grad delay case



(c) Grad error case




Problem Analysis

- How threads interact with each other?

	Sparse	Dense
Sync.		
Async.		

Problem Analysis




- How threads interact with each other?

	Sparse	Dense
Sync.		
Async.		AsynGrad

AsynGrad
aims to
solve
gradient
error case

Problem Analysis

- How threads interact with each other?

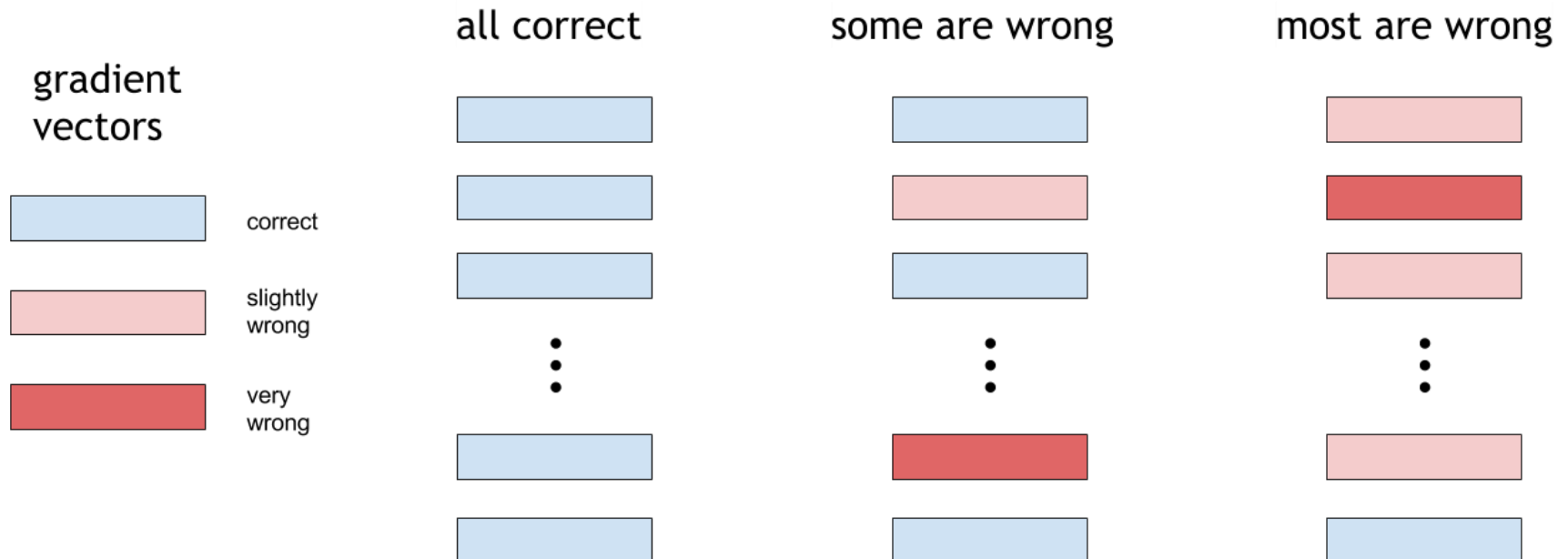
	Sparse	Dense
Sync.		
Async.		AsynGrad

This is our
proposal

Review of Gradient Error Case

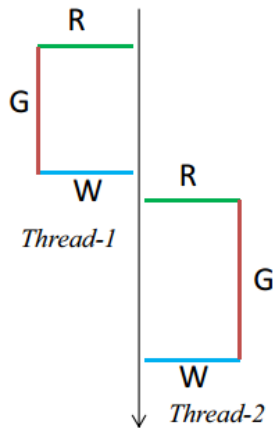
□ Gradient error has two aspects

- How many of the gradients are wrong?
- How wrong are they?

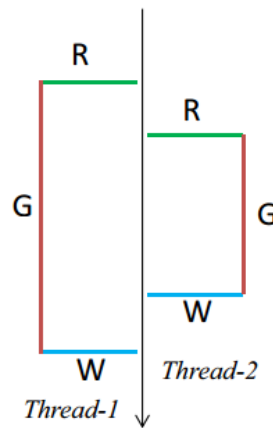


Experimental Observations

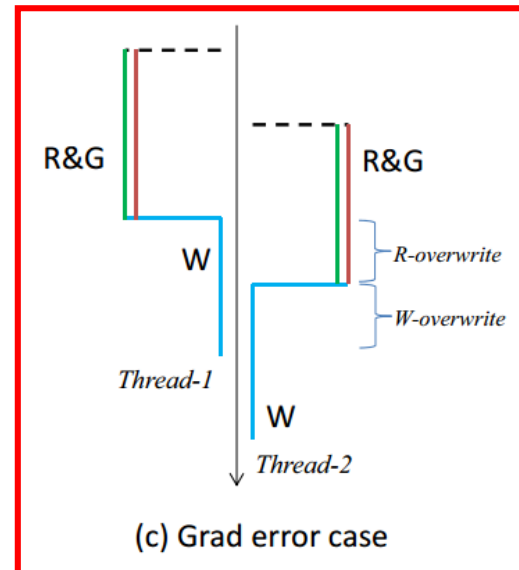
- Gradient error is **very common** in asynchronous training of neural networks in real-world tasks



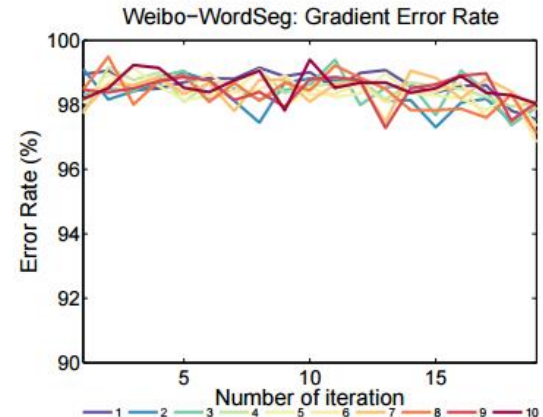
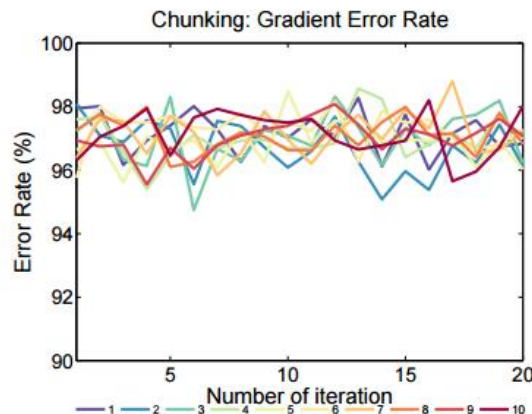
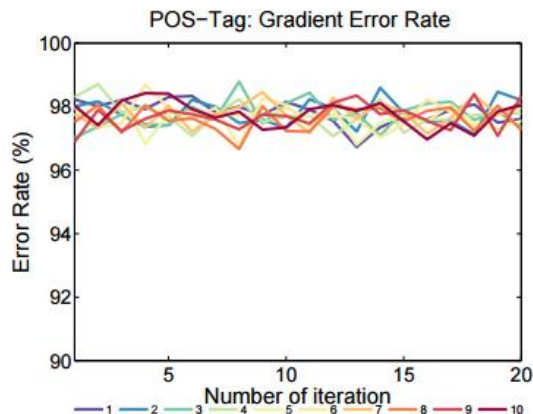
(a) Simple case



(b) Grad delay case

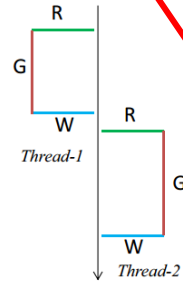


(c) Grad error case

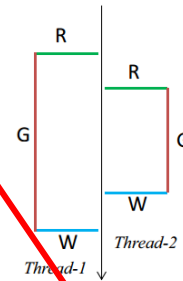


Experimental Observations

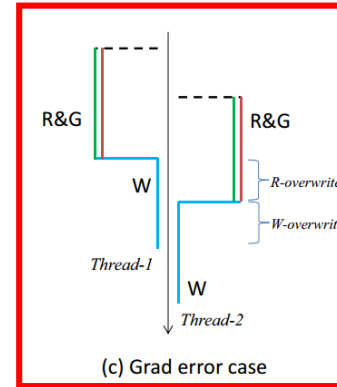
- Gradient error is **moderate** in asynchronous training of neural networks in real-world tasks



(a) Simple case



(b) Grad delay case



(c) Grad error case

gradient
vectors



correct



slightly
wrong

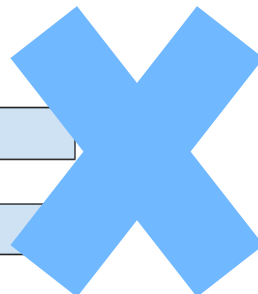


very
wrong

naïve case



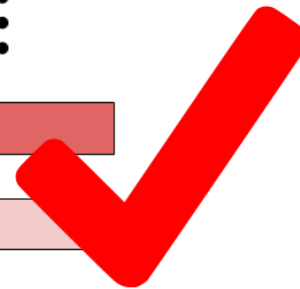
⋮



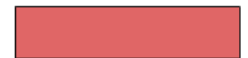
practical case



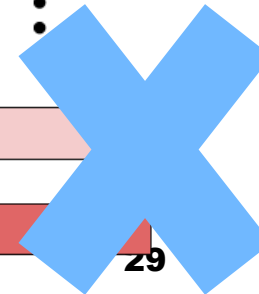
⋮



doomed case



⋮



Our Theoretical Analysis

□ Can training still converge with gradient errors?

Theorem 1 (AsynGrad convergence and convergence rate). *With the conditions (4), (5), (6), (7), let $\epsilon > 0$ be a target error, let \mathbf{w}^* denote the optimum). Let*

Even though there are gradient errors, training **still converges towards the optimum**, when the gradient errors are bounded.

where \mathbf{w} is such that $\mathbf{s}(\mathbf{w}) = \mathbb{E}_{\mathbf{z}}[\mathbf{s}_{\mathbf{z}}(\mathbf{w})]$. Let γ be a learning rate as

$$\gamma = \frac{c\epsilon - 2\tau q}{\beta q \kappa^2} \quad (9)$$

where we can set β as any value as far as $\beta \geq 1$. Let t be the number of updates as follows

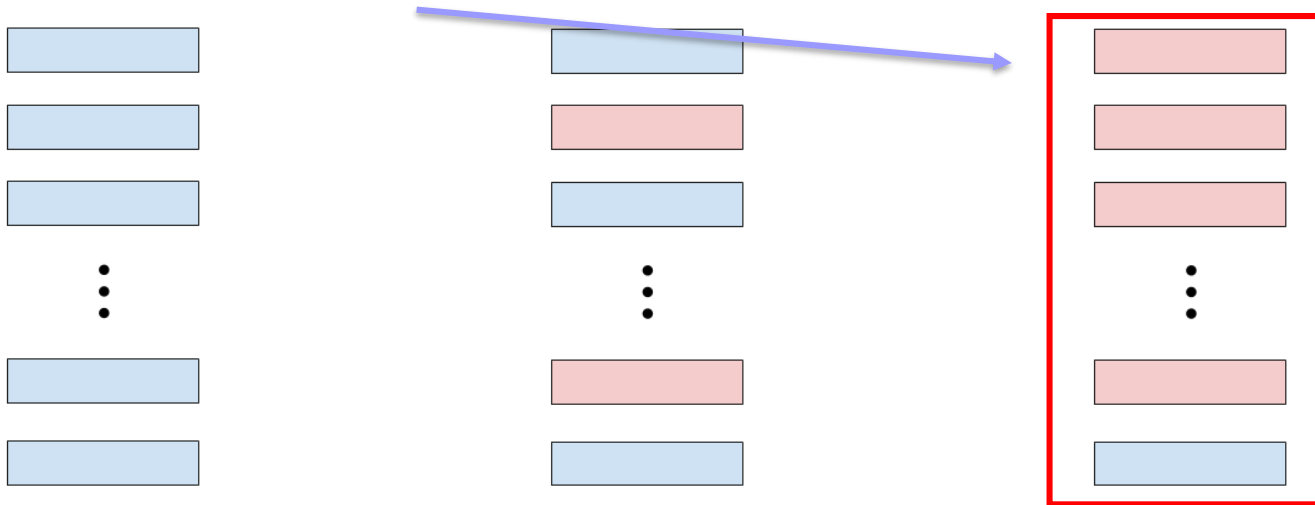
$$t \doteq \frac{\beta q \kappa^2 \log(qa_0/\epsilon)}{c(c\epsilon - 2\tau q)} \quad (10)$$

where \doteq means ceil-rounding of a real value to an integer, and a_0 is the initial distance such that $a_0 = \|\mathbf{w}_0 - \mathbf{w}^\|^2$. Then, after t updates of \mathbf{w} , AsynGrad converges towards the optimum such that $\mathbb{E}[f(\mathbf{w}_t) - f(\mathbf{w}^*)] \leq \epsilon$, as far as the gradient errors are bounded such that*

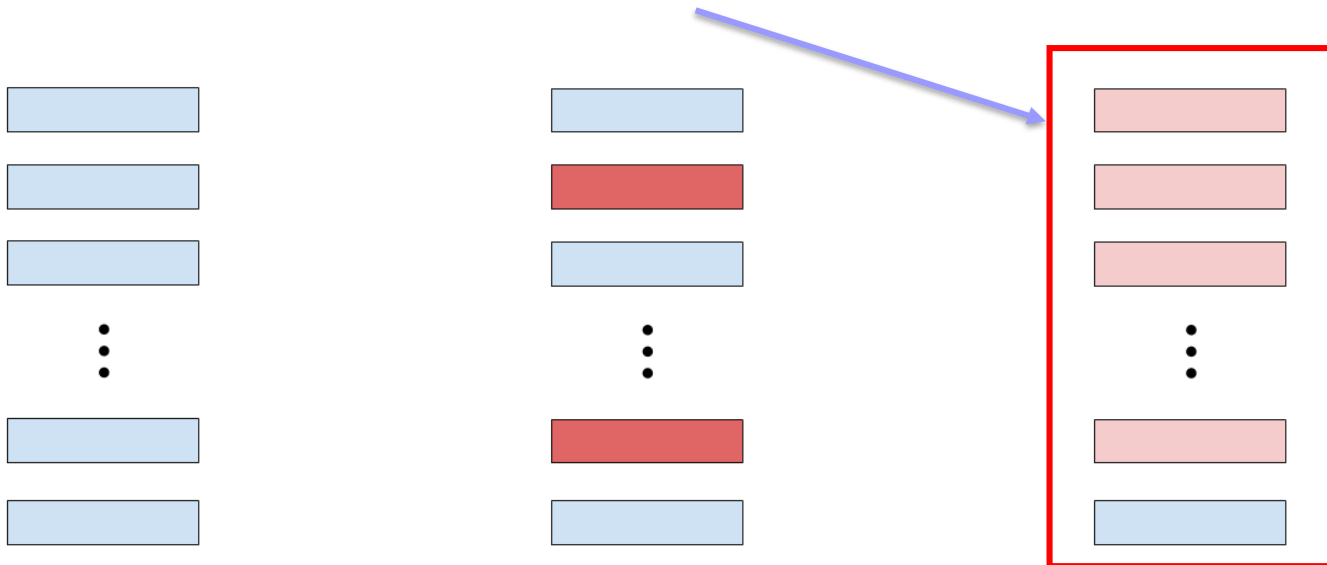
$$\tau \leq \frac{c\epsilon}{2q} \quad \text{bounded gradient errors} \quad (11)$$

Our Theoretical Analysis

- Even if **most of the gradients are wrong**



- With their **errors bounded**, training still **converge**



gradient
vectors



- An asynchronous parallel learning solution for fast training of neural networks
 - Asynchronous Parallel Learning with Gradient Error (AsynGrad)
- Algorithm

Algorithm 1 *AsynGrad*: Asynchronous Parallel Learning with Gradient Error

Input: model weights \mathbf{w} , training set S of m samples

Run k threads in parallel with share memory, and procedure of each thread is as follows:

repeat

 Get a sample \mathbf{z} uniformly at random from S

 Get the update term $\mathbf{s}_{\mathbf{z}}(\mathbf{w})$, which is computed as $\nabla f_{\mathbf{z}}(\mathbf{w})$ but usually contains error

 Update \mathbf{w} such that $\mathbf{w} \leftarrow \mathbf{w} - \gamma \mathbf{s}_{\mathbf{z}}(\mathbf{w})$

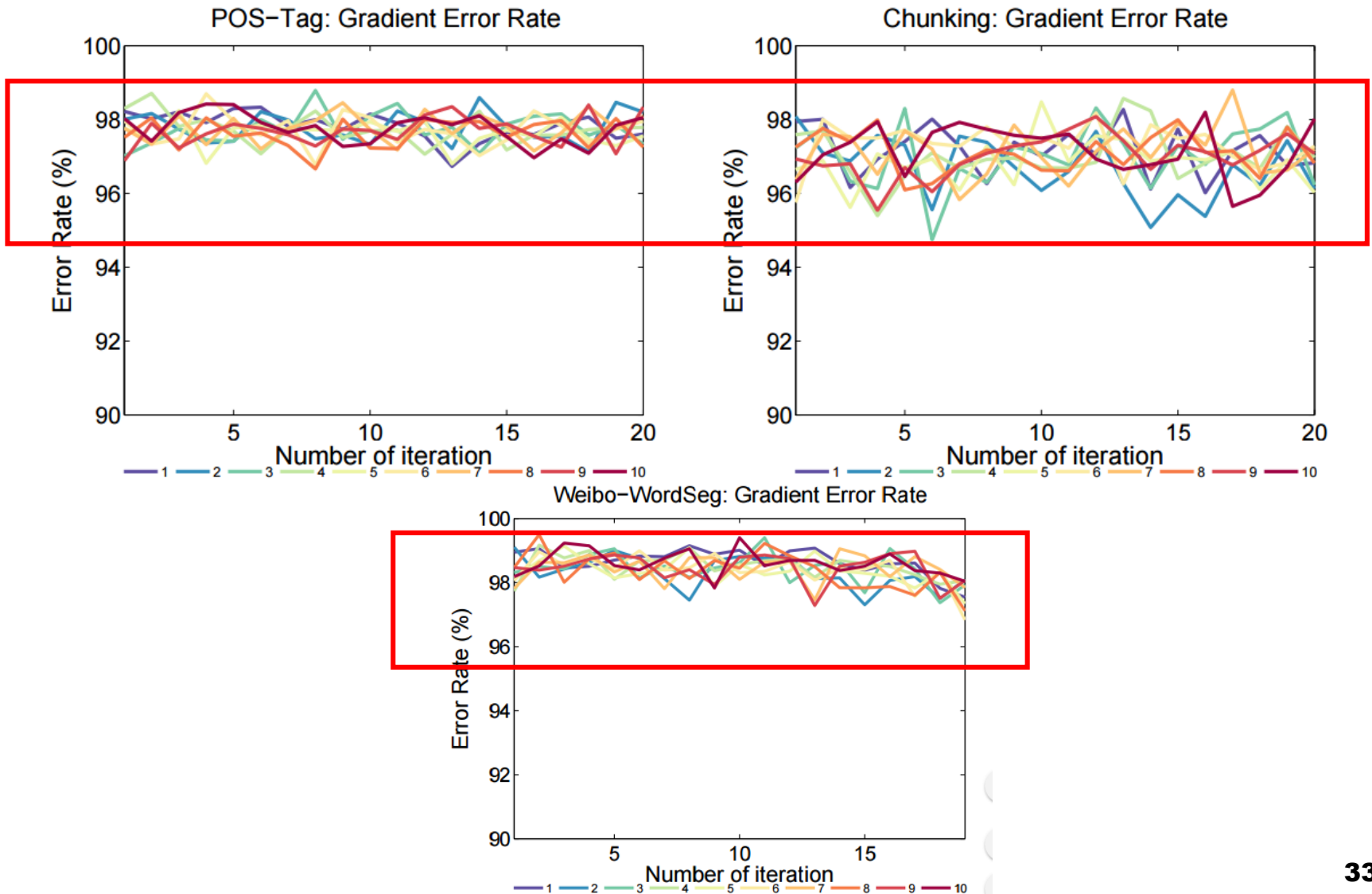
until Convergence

return \mathbf{w}

X. Sun. Asynchronous Parallel Learning for Neural Networks and Structured Models with Dense Features. COLING 2016.

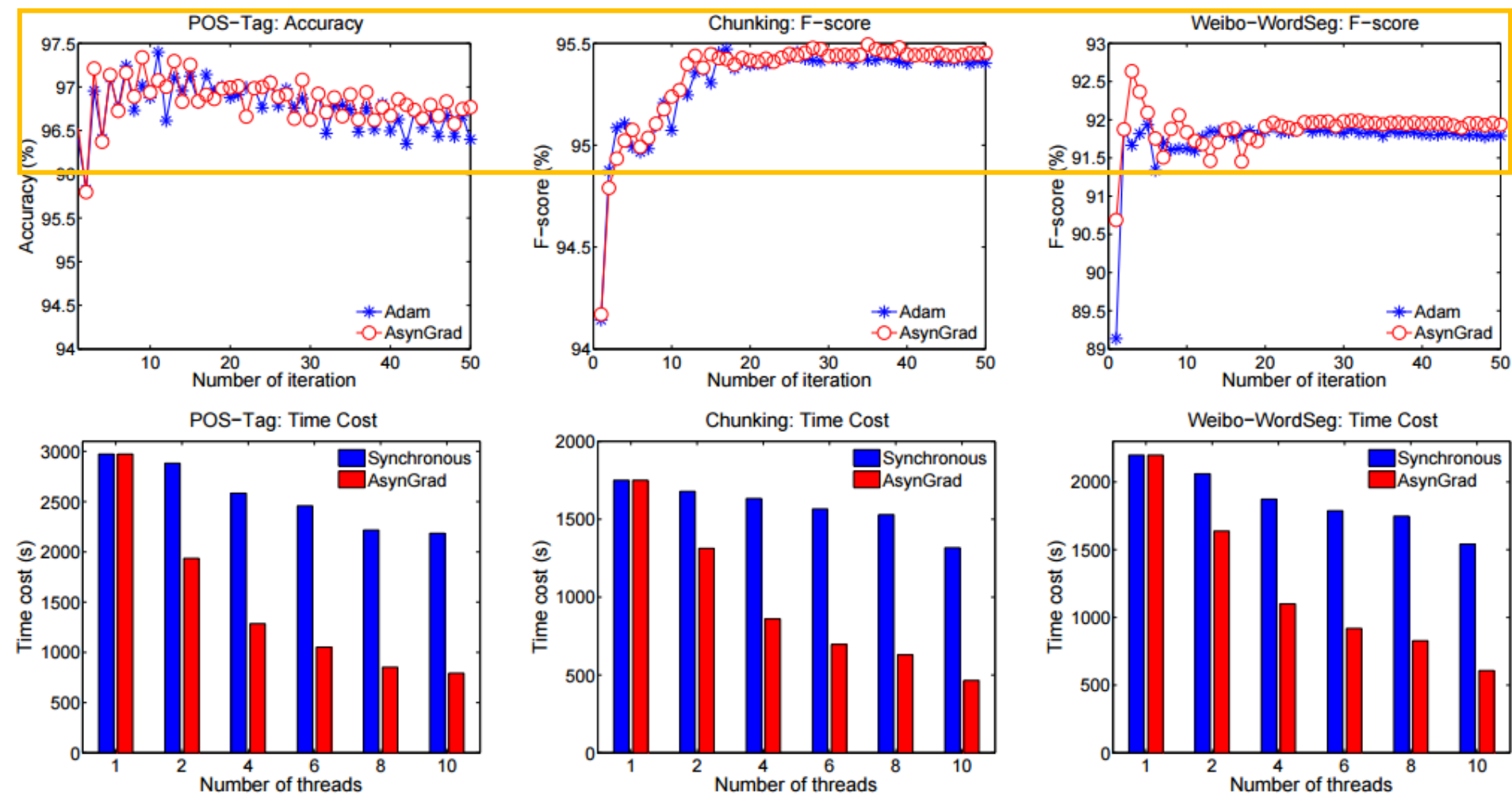
Experiments on LSTM

Experiments show a high gradient error rate



Experiments on LSTM

Experiments show that **AsynGrad** still converge even with a high gradient error rate

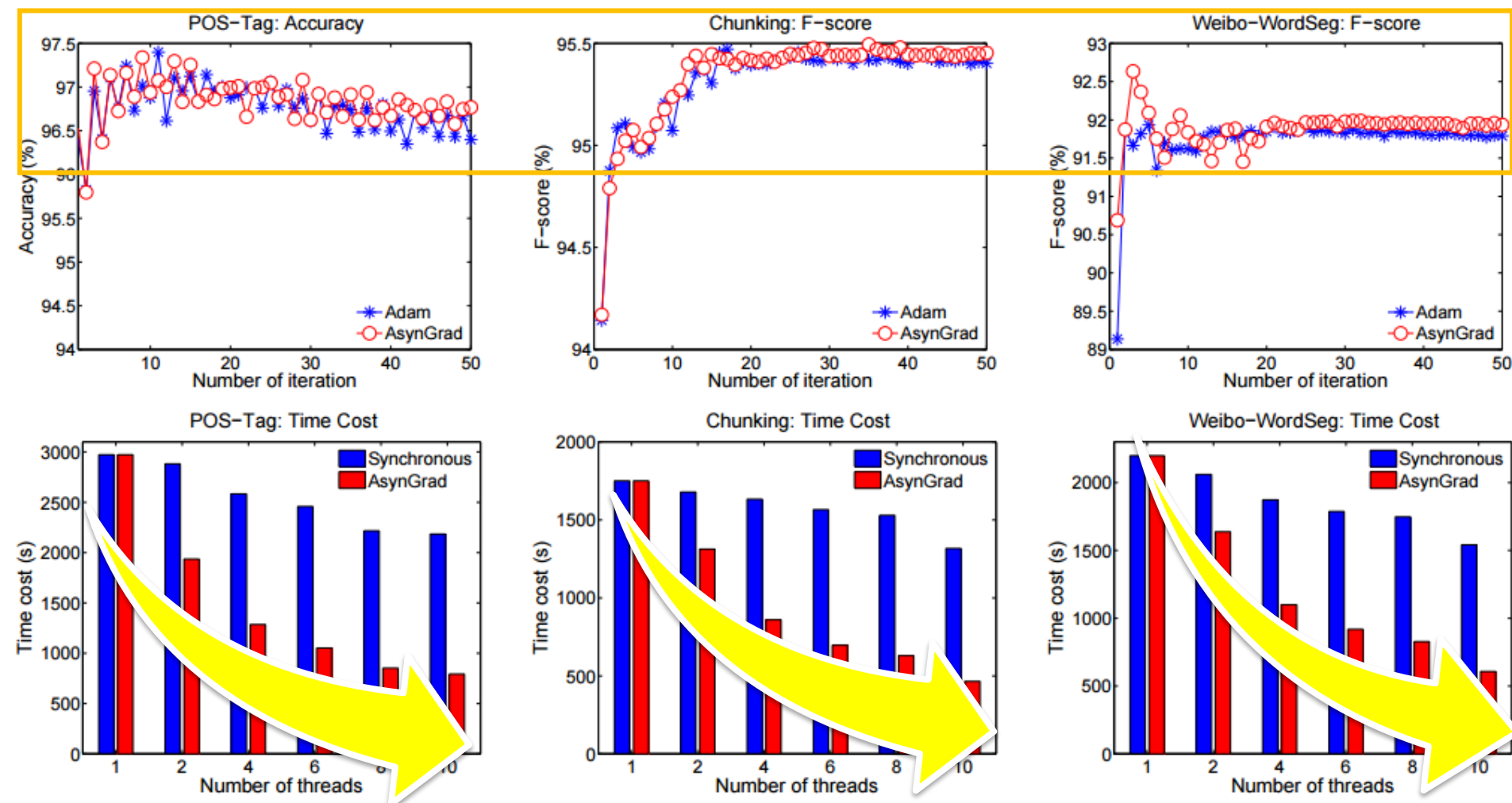


Experiments on LSTM

□ No loss on accuracy/F-score

AsynGrad

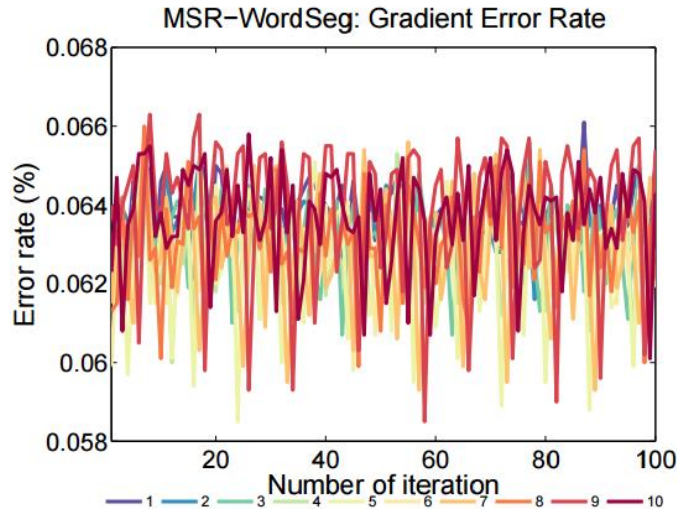
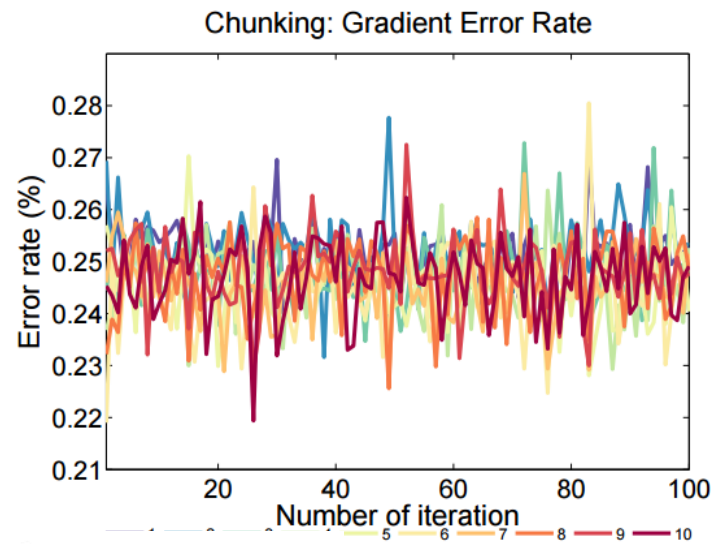
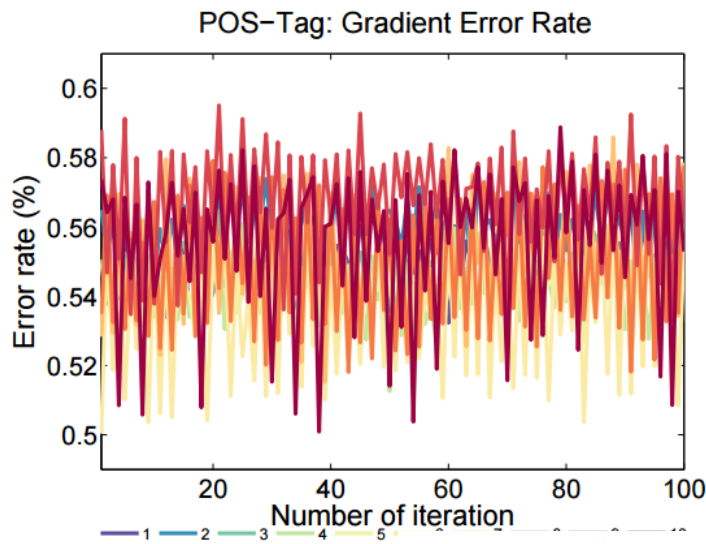
□ With **substantially faster** training speed



AsynGrad: A General-Purpose Solution

□ Also suitable for other dense feature models

□ Dense CRF -> moderate error rate



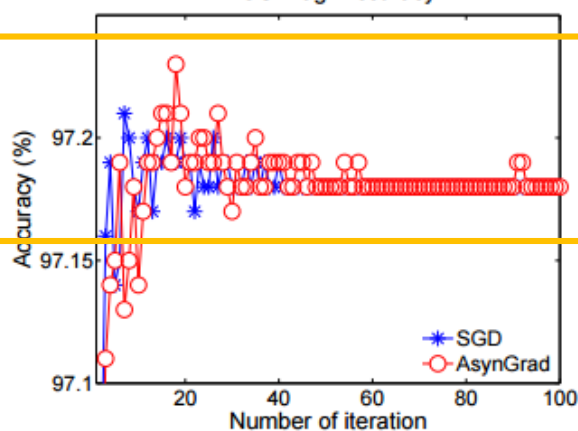
Experiments on Dense-CRF

□ No loss on accuracy/F-score

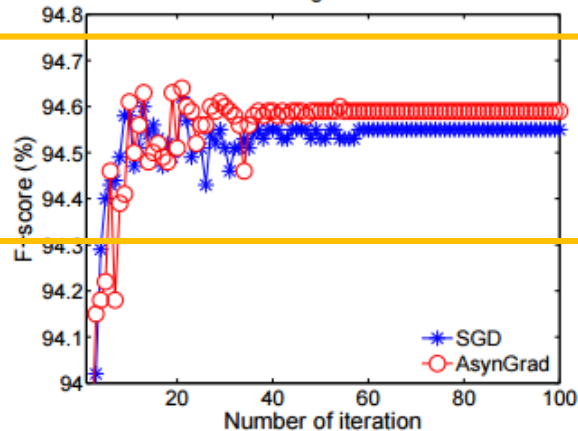
AsynGrad

□ With substantially faster training speed

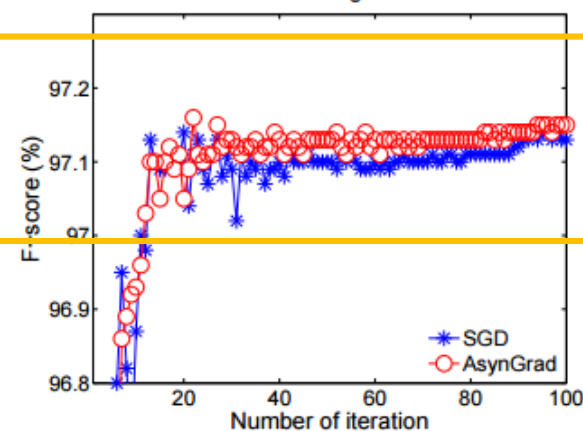
POS-Tag: Accuracy



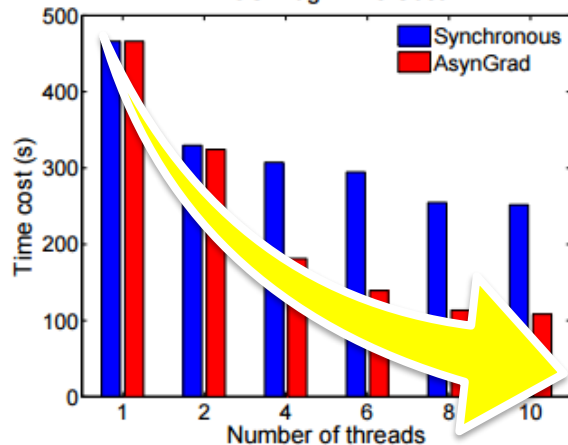
Chunking: F-score



MSR-WordSeg: F-score

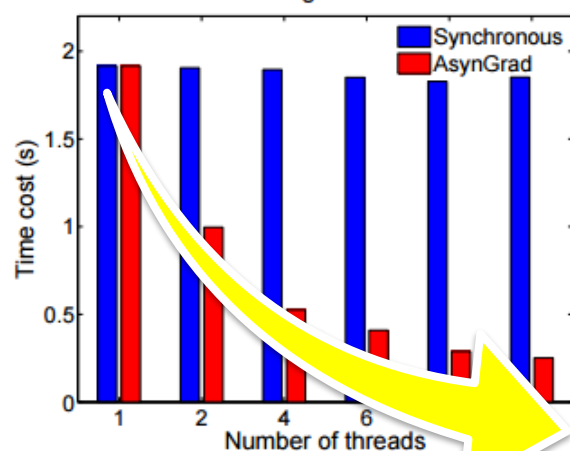


POS-Tag: Time Cost



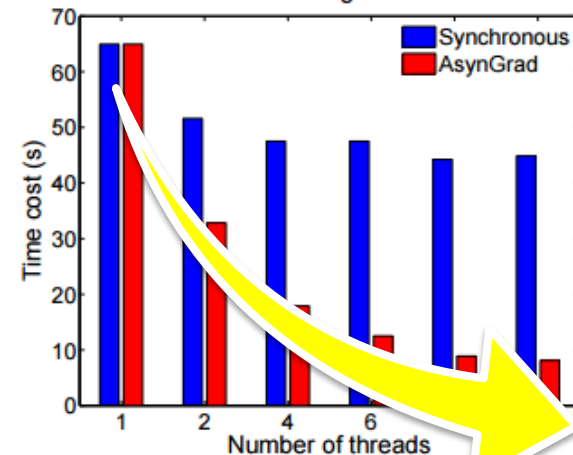
POS-Tag: Accuracy with SR

Chunking: Time Cost



Chunking: F-score with SR





MSR-WordSeg: Time Cost



MSR-WordSeg: F-score with SR

Problem Analysis

- How threads interact with each other?

	Sparse	Dense
Sync.		
Async.		

- ❑ **Gradient errors** are common and inevitable in asynchronous training of dense feature models
 - ❑ Such as neural networks
- ❑ **AsynGrad survives with gradient errors**
 - ❑ With substantial faster training speed
 - ❑ No loss at all on test accuracy
 - ❑ Theoretical justification

Thanks!