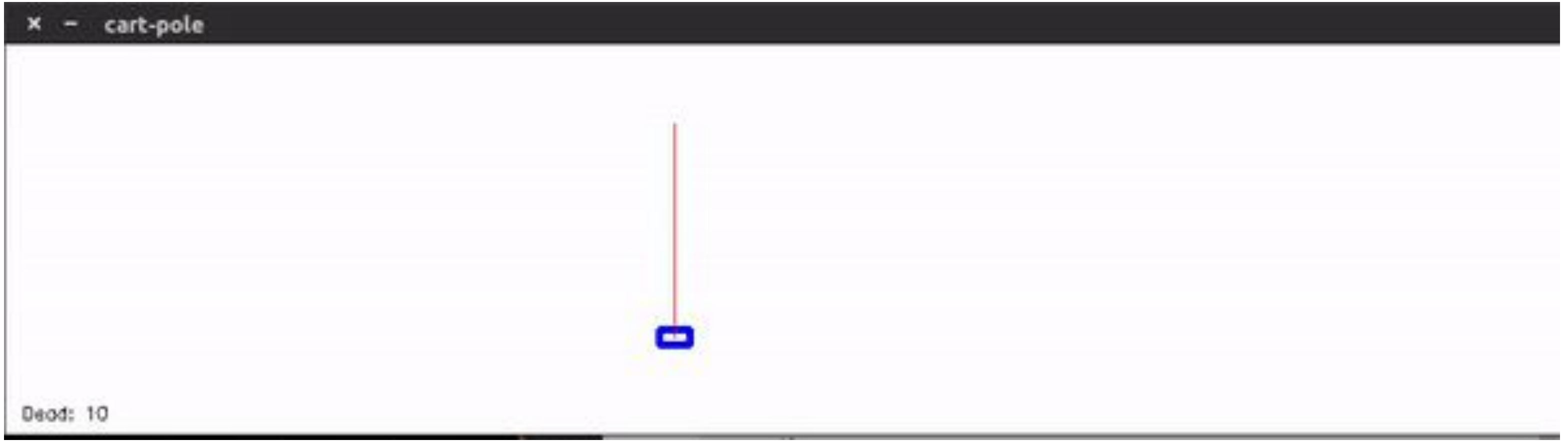
* Thanks!
* No notes needed ;)
  * Stickers

Let's build Tensorflow together! :) :) :) :)

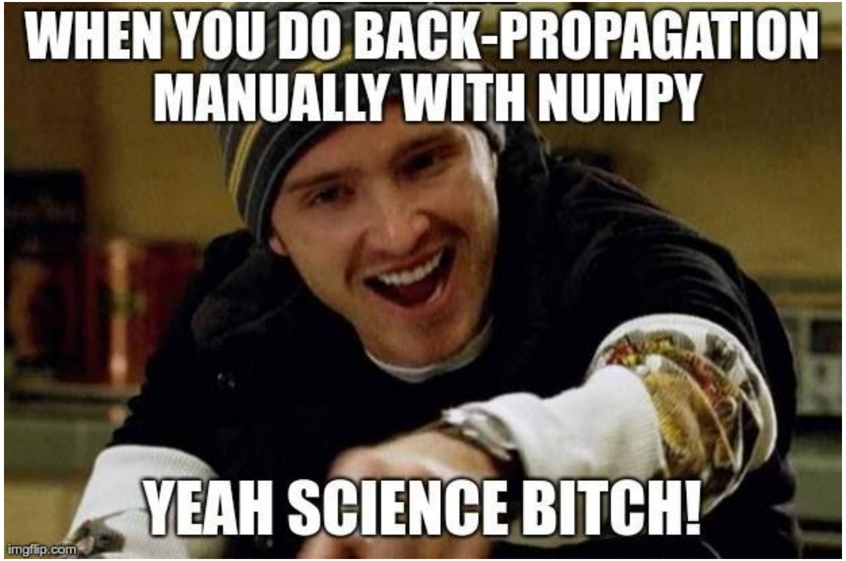
Teaser 001: VQA

```
Q: What is the animal in the picture?      . A: cat
Q: What is the cat doing?                  . A: sitting
Q: What is the cat color?                  . A: white
Q: Is the cat smiling?                     . A: yes
```

Teaser 002: Deep RL



Motivation


WHEN YOU DO BACK-PROPAGATION MANUALLY WITH NUMPY
YEAH SCIENCE BITCH!
imgflip.com

Motivation

Neural nets are

* (Computational) Graphs

Neural nets are

* (Computational) Graphs

* Directed

Neural nets are

* (Computational) Graphs

* Directed

* Acyclic

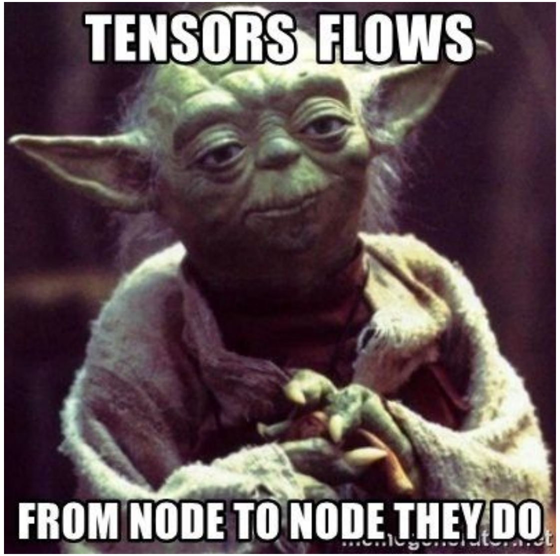
Neural nets are

* (Computational) Graphs

* Directed

* Acyclic

e.g. Conv, RNNs



The case for "Back"-prop

$$x$$

$$y = g(x)$$

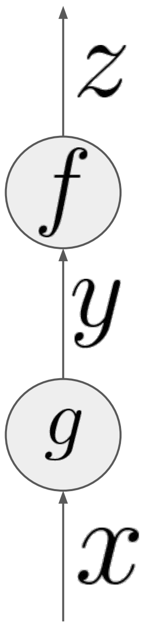
$$z = f(y)$$

The case for "Back"-prop

$$x$$

$$y = g(x)$$

$$z = f(y)$$

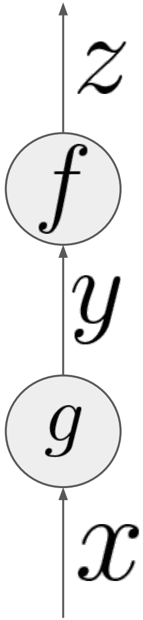


The case for "Back"-prop

$$x$$

$$y = g(x)$$

$$z = f(y)$$



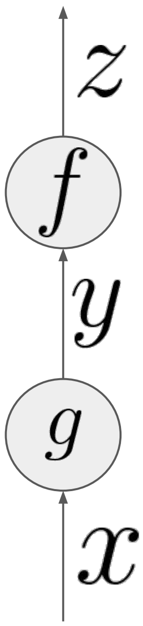
$$\frac{\partial z}{\partial x} =$$

The case for "Back"-prop

$$x$$

$$y = g(x)$$

$$z = f(y)$$



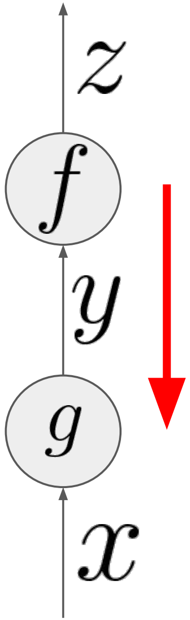
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

The case for "Back"-prop

$$x$$

$$y = g(x)$$

$$z = f(y)$$



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

The case for "Back"-prop

$$y_1 = g_1(x)$$

$$y_2 = g_2(x)$$

$$y_3 = g_3(x)$$

$$z = h(y_1, y_2, y_3)$$

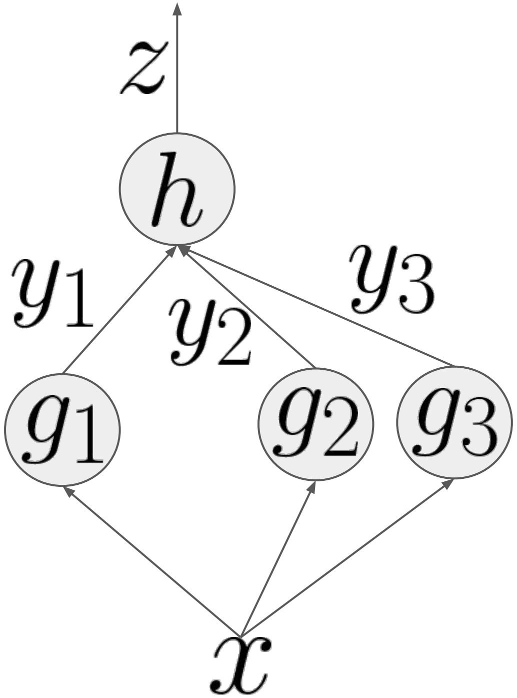
The case for "Back"-prop

$$y_1 = g_1(x)$$

$$y_2 = g_2(x)$$

$$y_3 = g_3(x)$$

$$z = h(y_1, y_2, y_3)$$



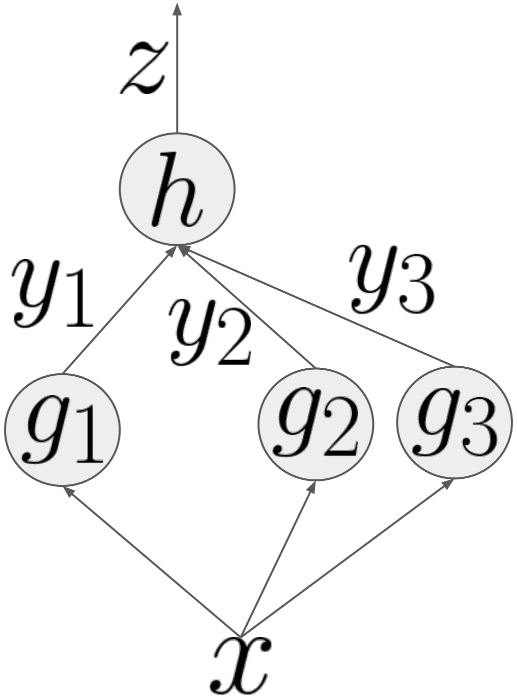
The case for "Back"-prop

$$y_1 = g_1(x)$$

$$y_2 = g_2(x)$$

$$y_3 = g_3(x)$$

$$z = h(y_1, y_2, y_3)$$



$$\frac{\partial z}{\partial x} = \quad ?$$

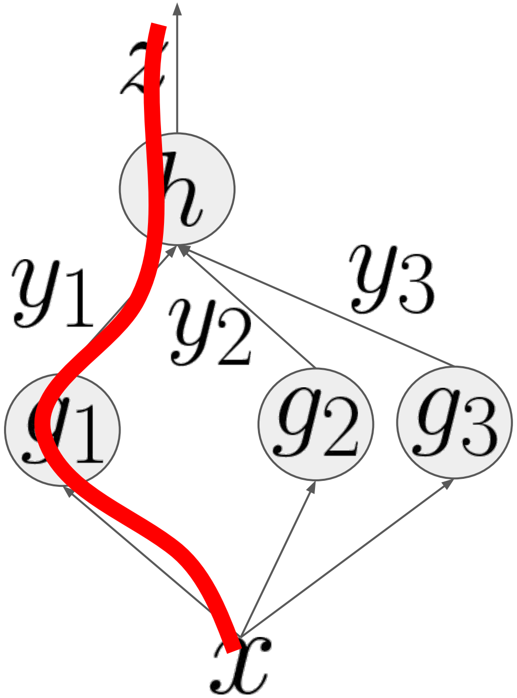
The case for "Back"-prop

$$y_1 = g_1(x)$$

$$y_2 = g_2(x)$$

$$y_3 = g_3(x)$$

$$z = h(y_1, y_2, y_3)$$



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x}$$

$$+$$

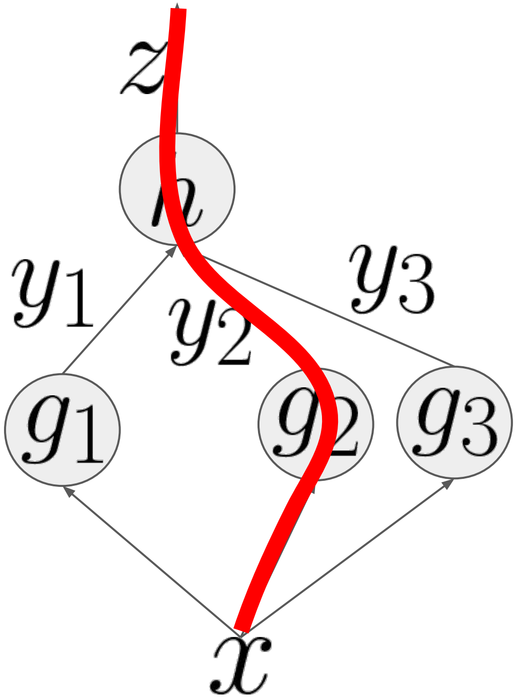
The case for "Back"-prop

$$y_1 = g_1(x)$$

$$y_2 = g_2(x)$$

$$y_3 = g_3(x)$$

$$z = h(y_1, y_2, y_3)$$



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x} +$$

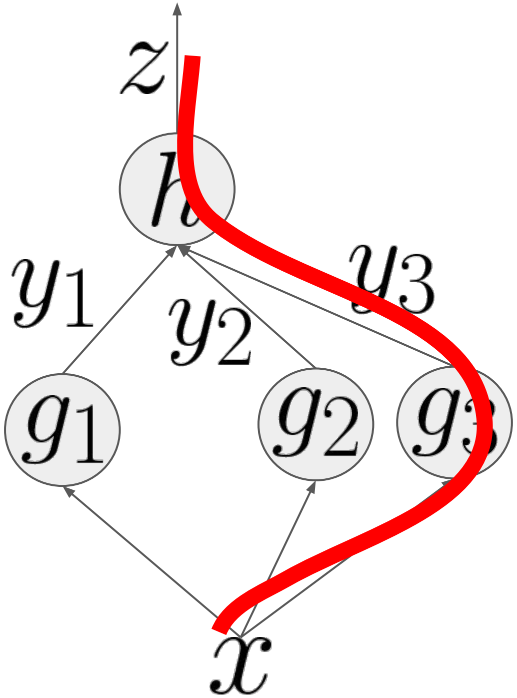
The case for "Back"-prop

$$y_1 = g_1(x)$$

$$y_2 = g_2(x)$$

$$y_3 = g_3(x)$$

$$z = h(y_1, y_2, y_3)$$



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x} + \frac{\partial z}{\partial y_3}\frac{\partial y_3}{\partial x}$$

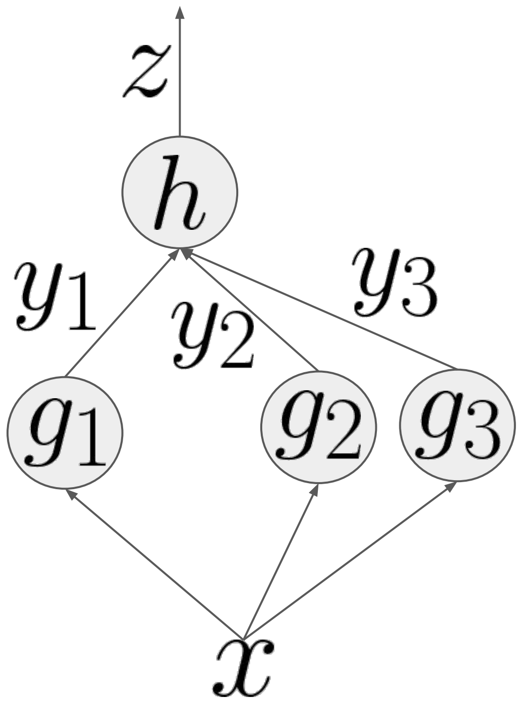
The case for "Back"-prop

$$y_1 = g_1(x)$$

$$y_2 = g_2(x)$$

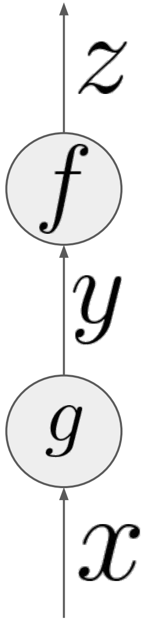
$$y_3 = g_3(x)$$

$$z = h(y_1, y_2, y_3)$$



$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial z}{\partial y_2}\frac{\partial y_2}{\partial x} + \frac{\partial z}{\partial y_3}\frac{\partial y_3}{\partial x}$$

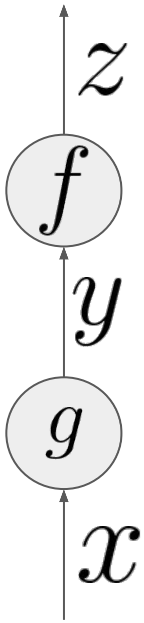
Modularity of Backprop



$$f(\cdot) \equiv softmax(\cdot)$$

$$g(\cdot) \equiv conv(\cdot)$$

Modularity of Backprop

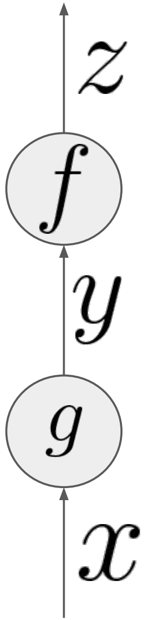


A

$$f(\cdot) \equiv softmax(\cdot)$$

$$g(\cdot) \equiv conv(\cdot)$$

Modularity of Backprop



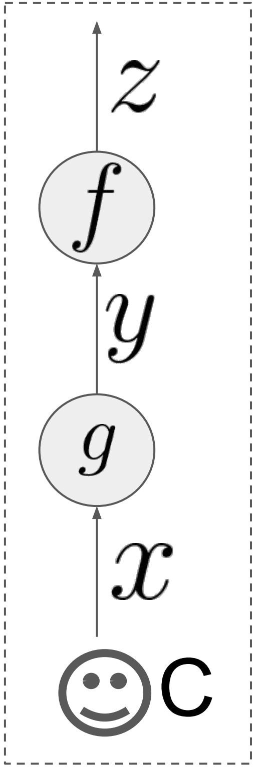
☺A

$$f(\cdot) \equiv softmax(\cdot)$$

$$g(\cdot) \equiv conv(\cdot)$$

☺B

Modularity of Backprop



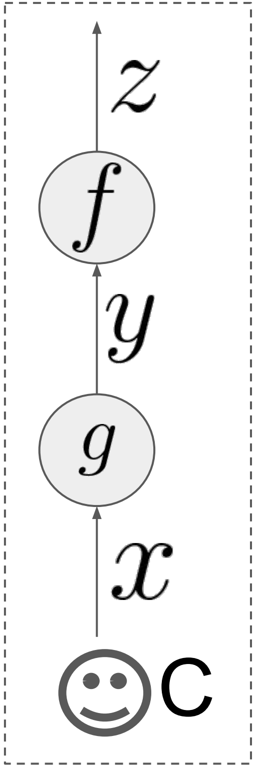
A

$$f(\cdot) \equiv softmax(\cdot)$$

$$g(\cdot) \equiv conv(\cdot)$$

B

Modularity of Backprop



A

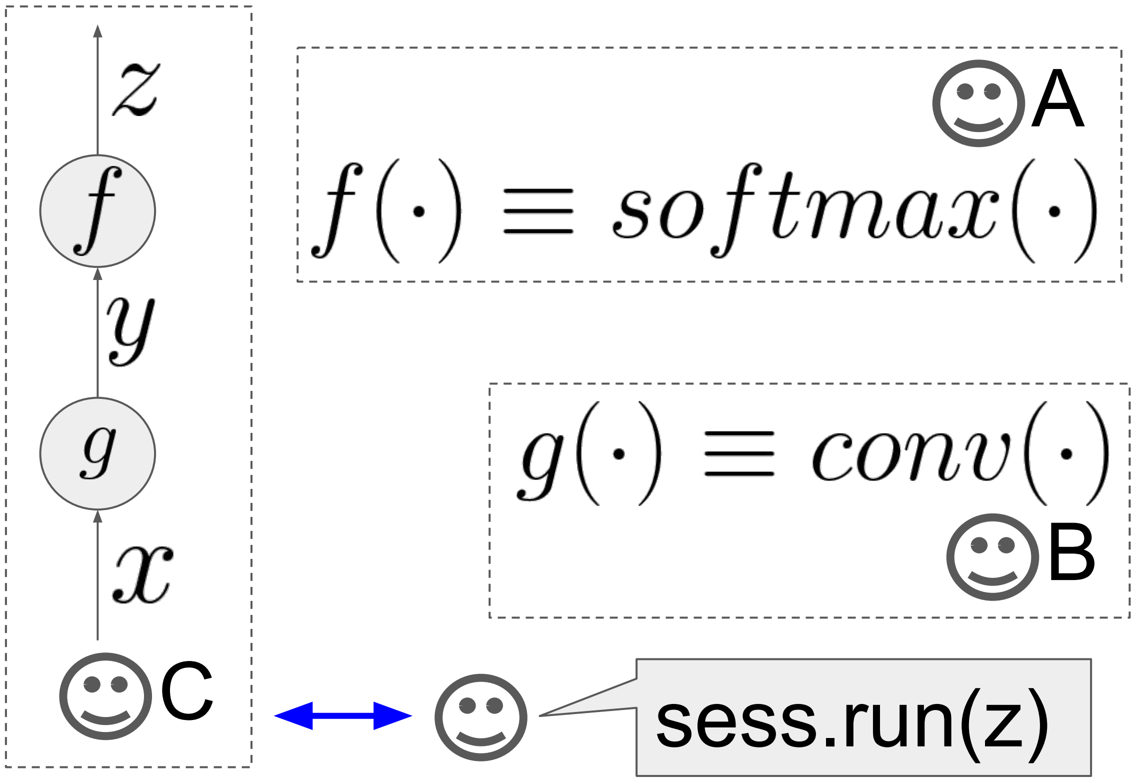
$$f(\cdot) \equiv softmax(\cdot)$$

$$g(\cdot) \equiv conv(\cdot)$$

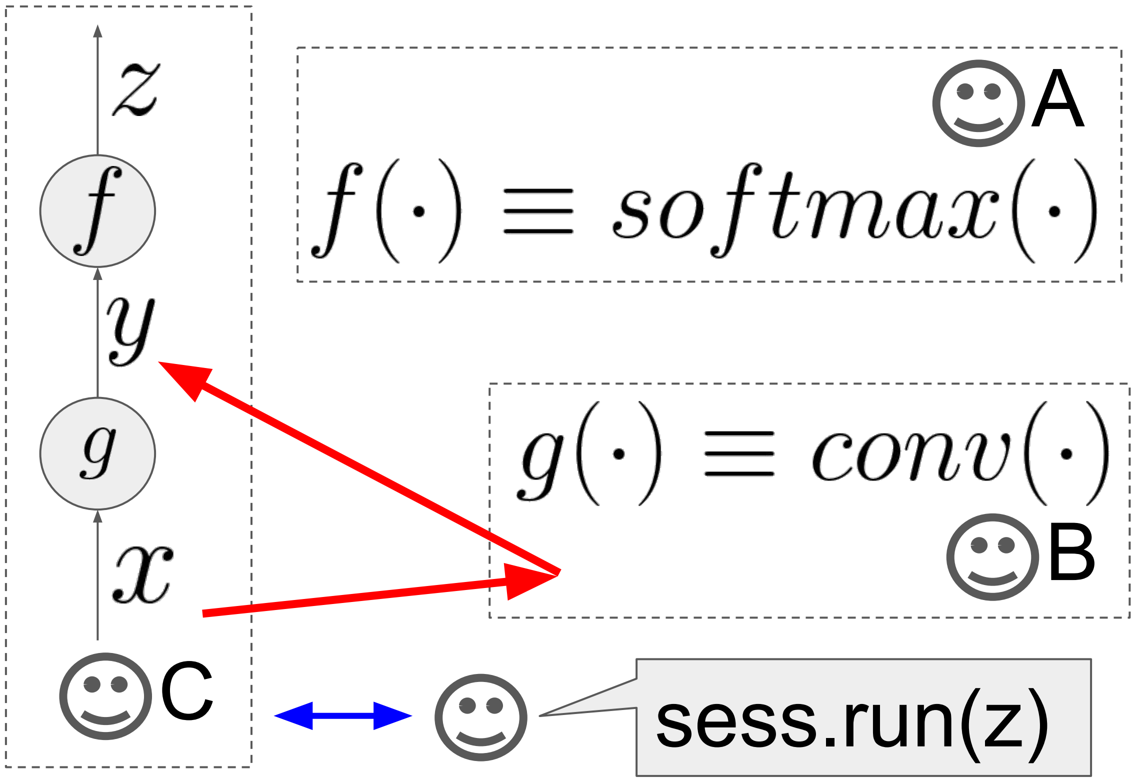
B

VietAI Student

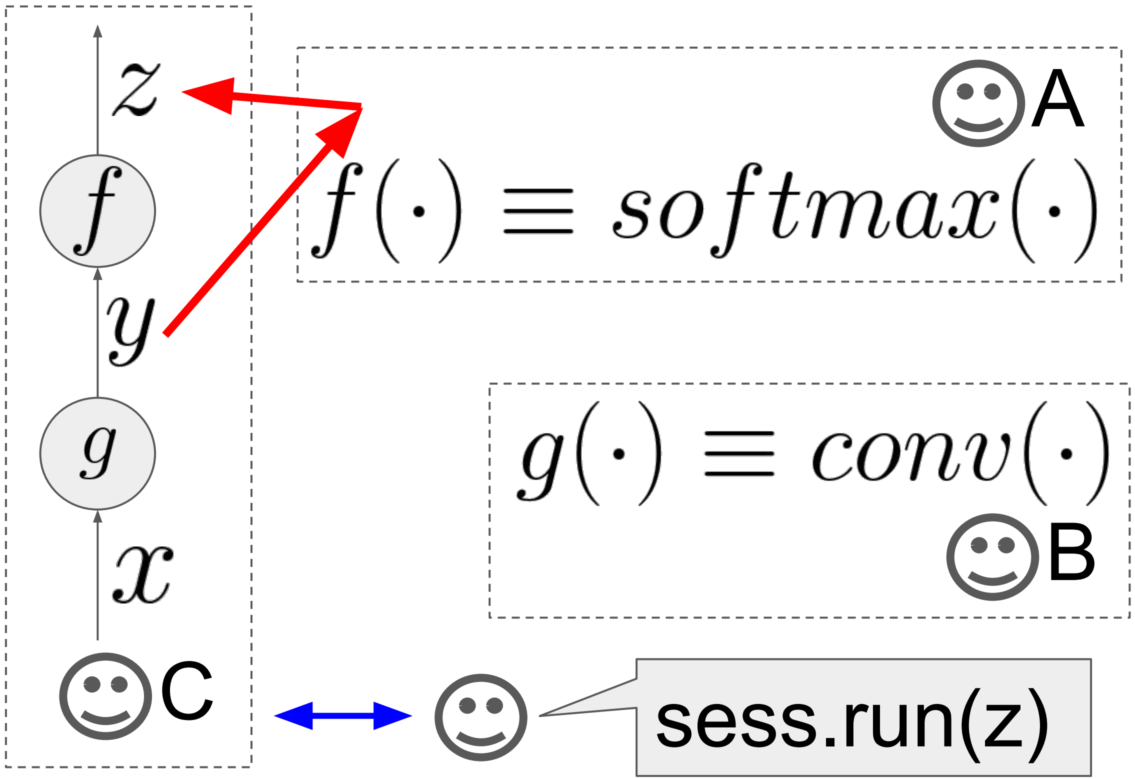
Modularity of Backprop



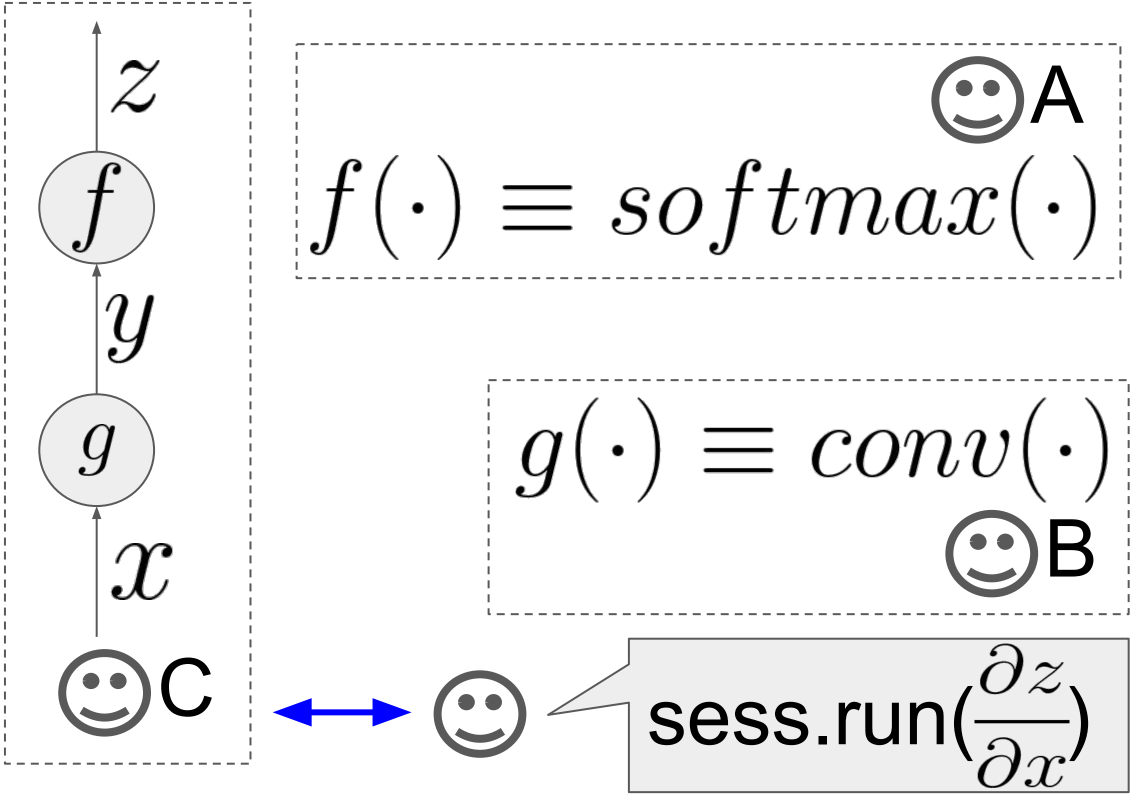
Modularity of Backprop



Modularity of Backprop

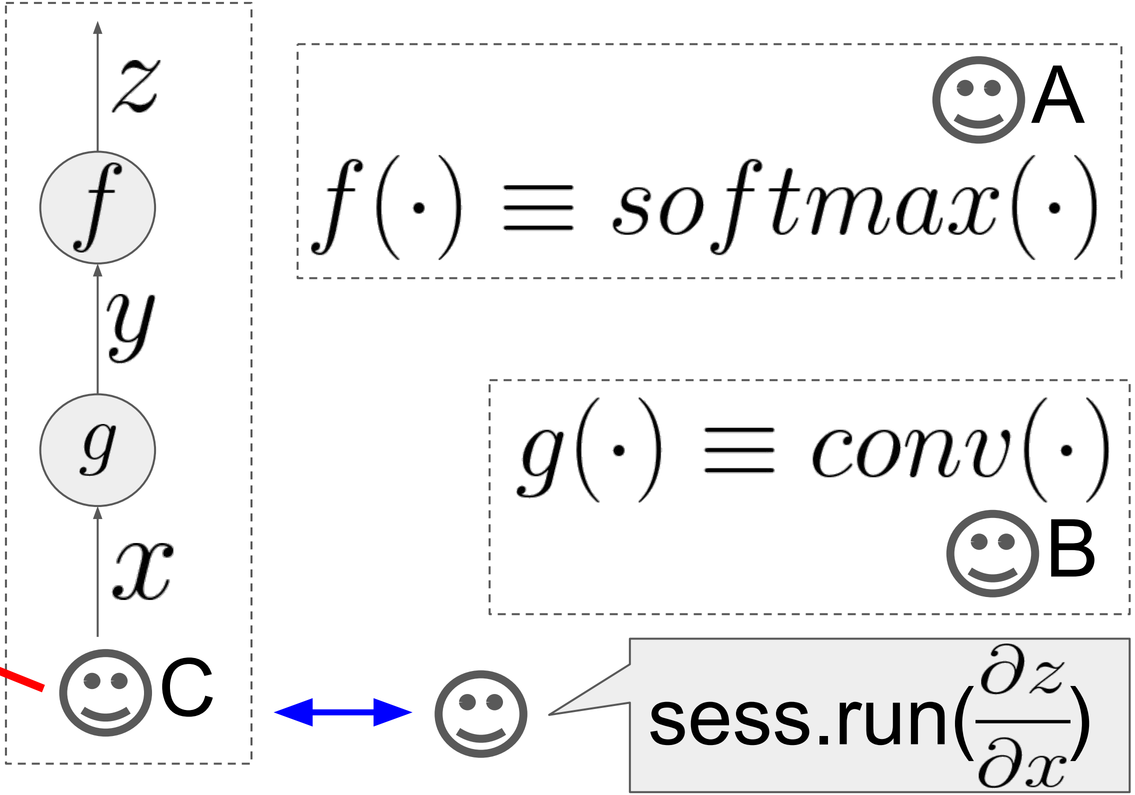


Modularity of Backprop



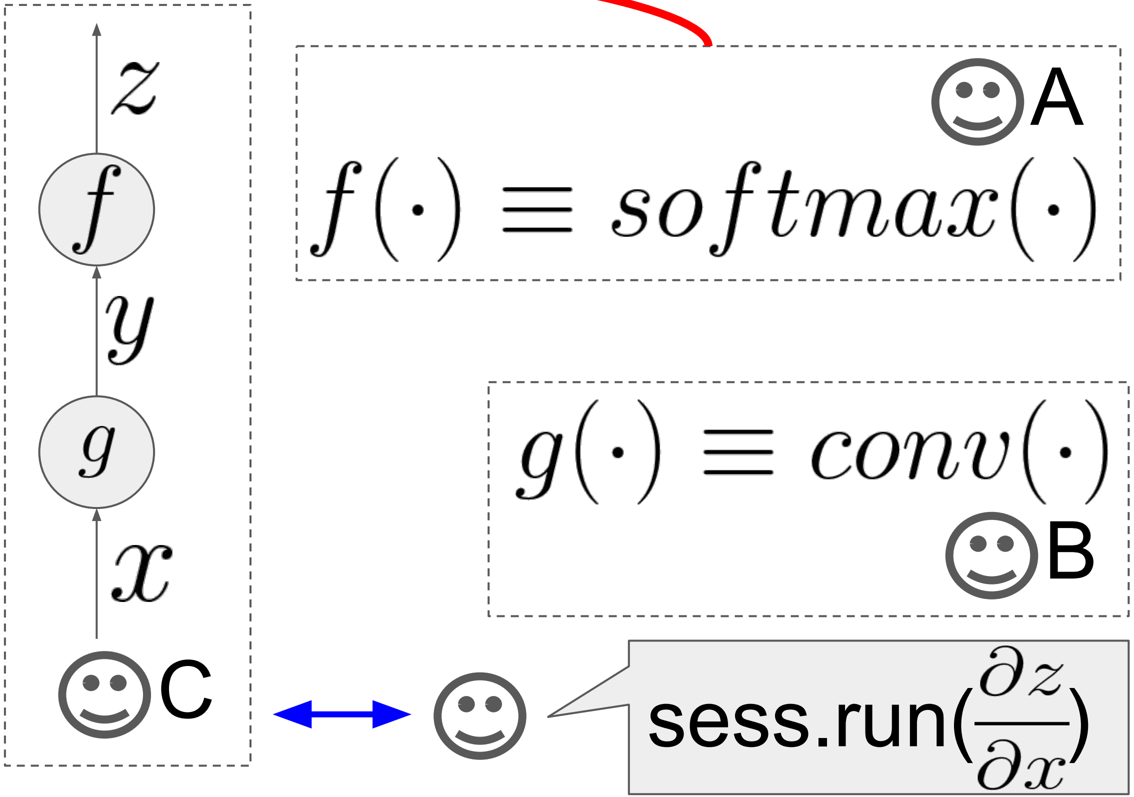
Modularity of Backprop

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$



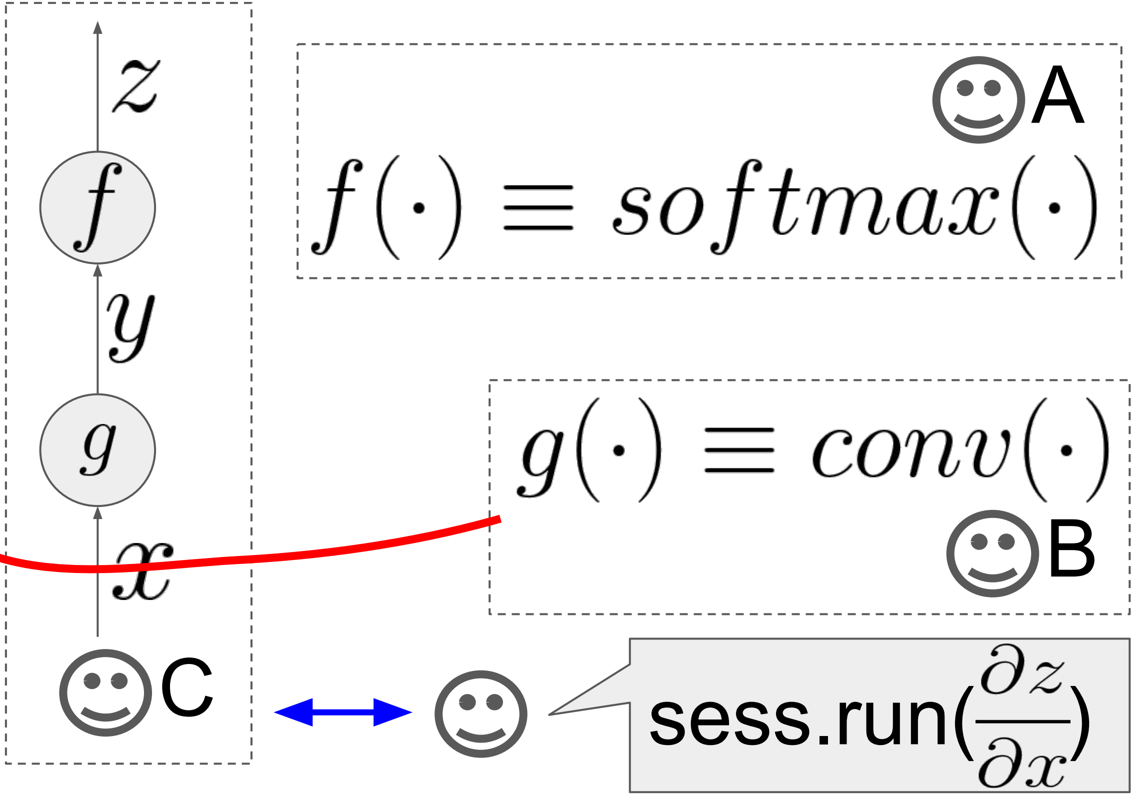
Modularity of Backprop

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$



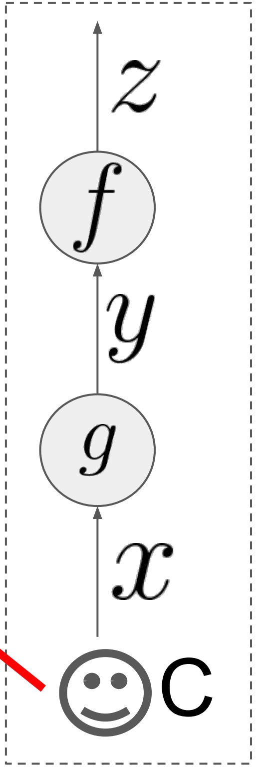
Modularity of Backprop

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$



Modularity of Backprop

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$



A

$$f(\cdot) \equiv softmax(\cdot)$$

$$g(\cdot) \equiv conv(\cdot)$$

B

sess.run($\frac{\partial z}{\partial x}$)

Modularity of Backprop

* Softmax()
* Softmax'()

A

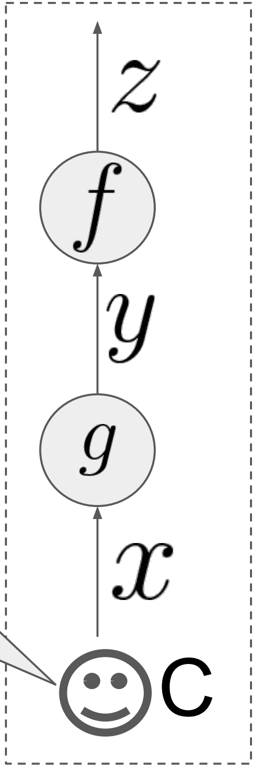
$$f(\cdot) \equiv softmax(\cdot)$$

$$g(\cdot) \equiv conv(\cdot)$$

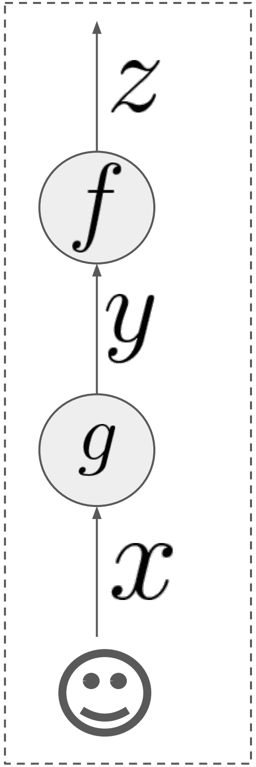
B

Conv & Conv'

* Architecture
* Chain-rule



Modularity of Backprop



VietAI Student

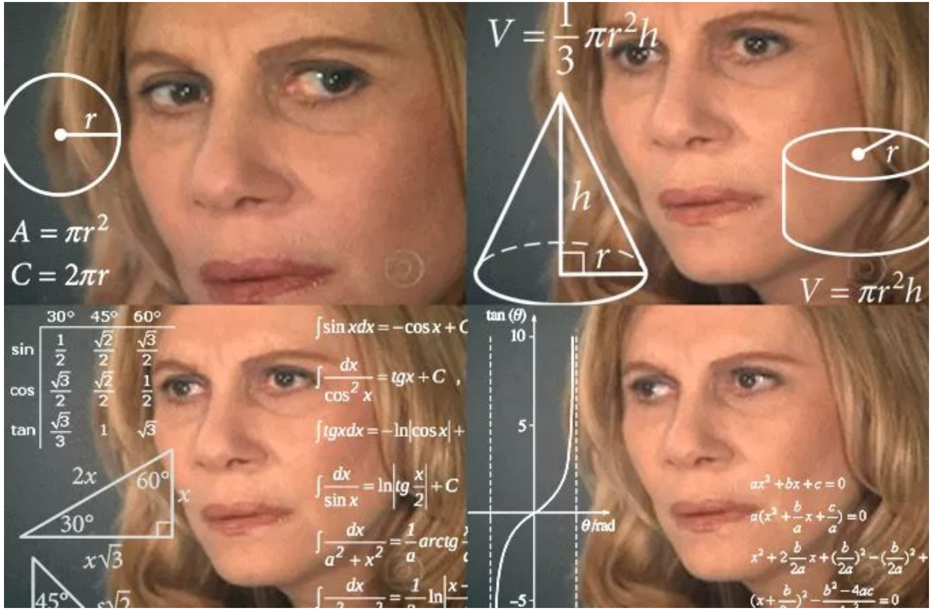
VietAI Student ☺

$$f(\cdot) \equiv softmax(\cdot)$$

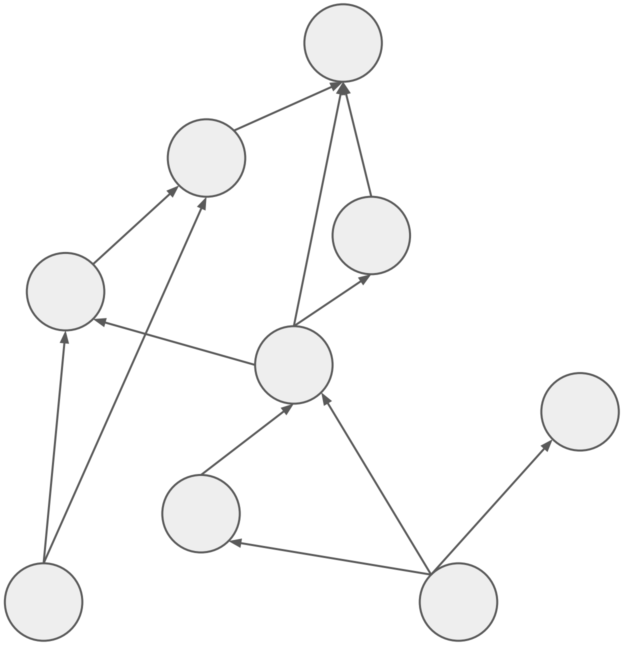
$$g(\cdot) \equiv conv(\cdot)$$

VietAI Student ☺

Part 1. The Computational Graph Expert

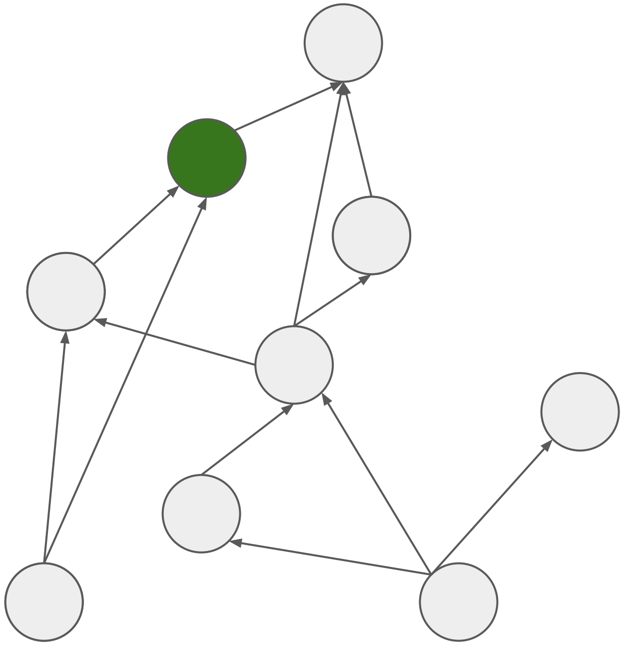


DAG Representation



Q for CS major:
"What data structure to represent this graph?"

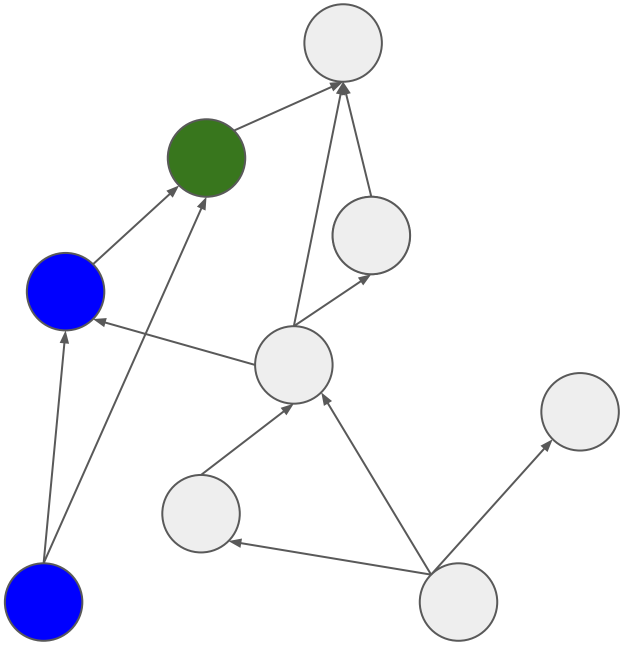
DAG Representation



Q for CS major:
"What data structure to represent this graph?"

A: I need
`"Sess.run(any_node)"`

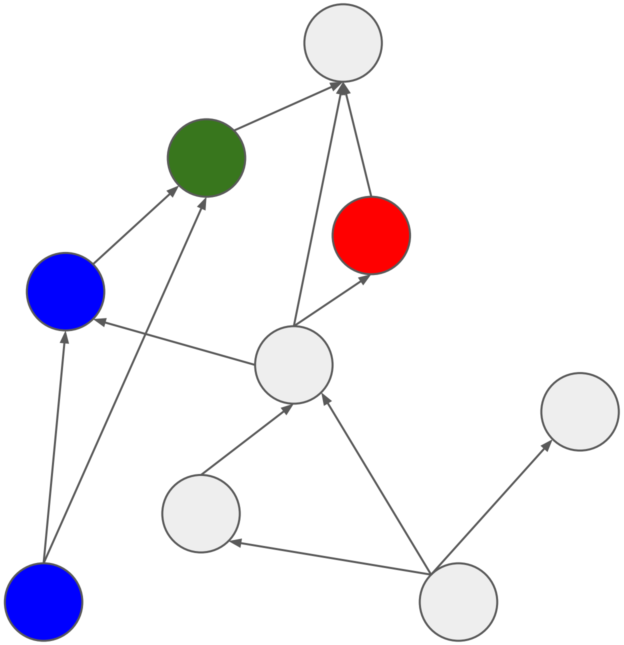
DAG Representation



Q for CS major:
"What data structure to represent this graph?"

A: I need
`"Sess.run(any_node)"`

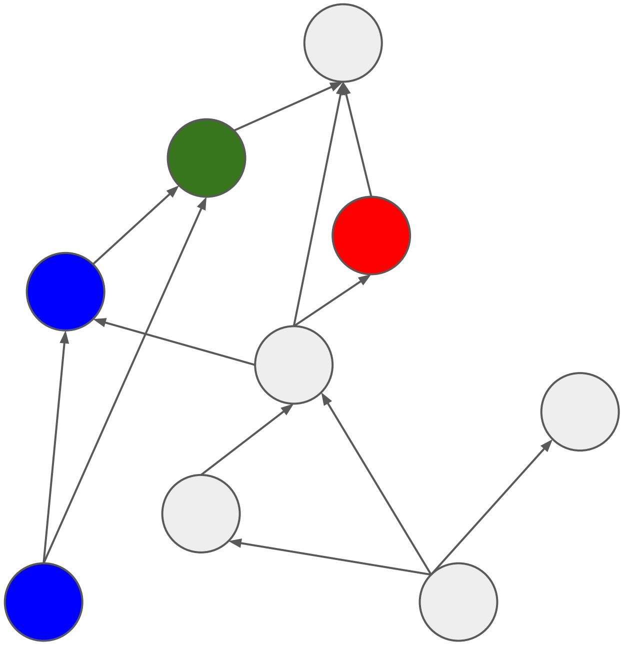
DAG Representation



Q for CS major:
"What data structure to represent this graph?"

A: I need
`"Sess.run(any_node)"`

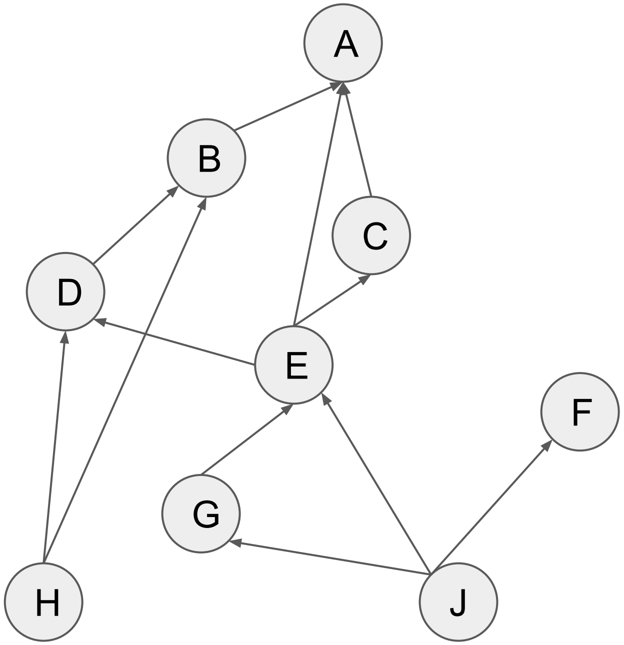
DAG Representation



Q for CS major:
"What data structure to represent this graph?"

A: I need
`"Sess.run(any_node)"`
==> **List of dependencies.**

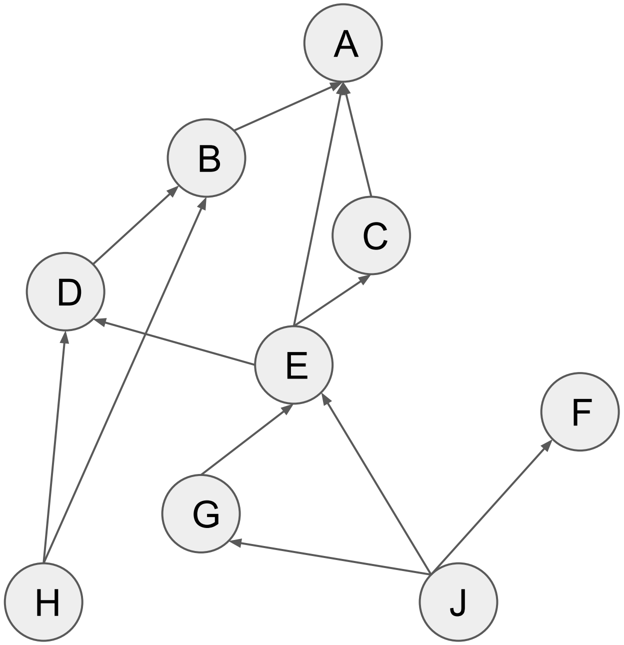
DAG Representation



Representation:

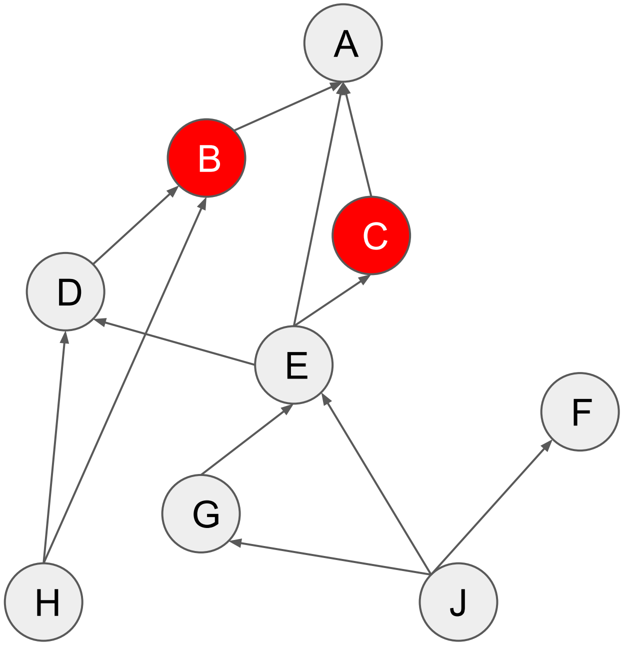
```
Graph = {
    A: [B, C, E]
    B: [D, H]
    C: [E]
    D: [E, H]
    E: [G, J]
    F: [J]
    G: [J]
    H: []
    J: []
}
```

DAG Representation



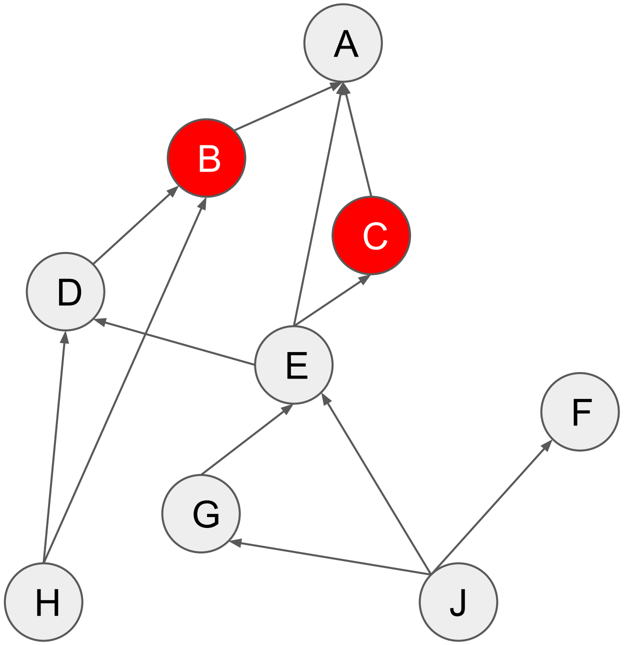
https://gist.github.com/thtrieu/c91c49599 68ef944cbecaa9bc5f287d1

Forward: caching values



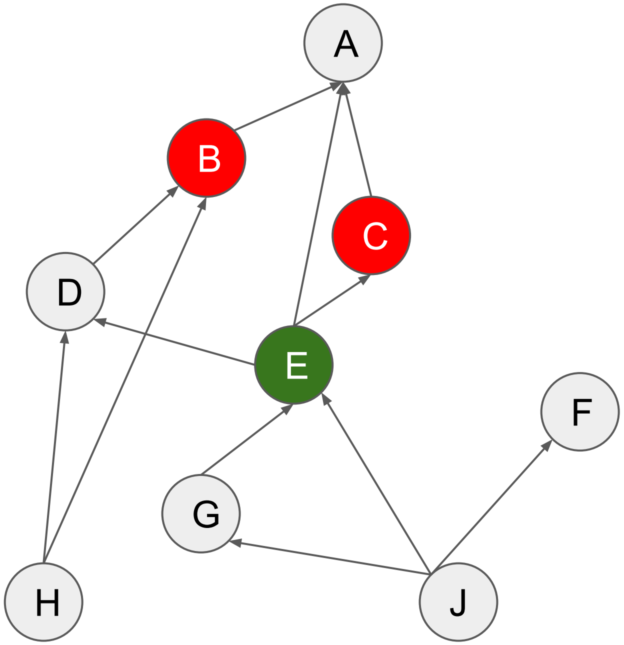

```
Session.run([B, C])
```

Forward: caching values



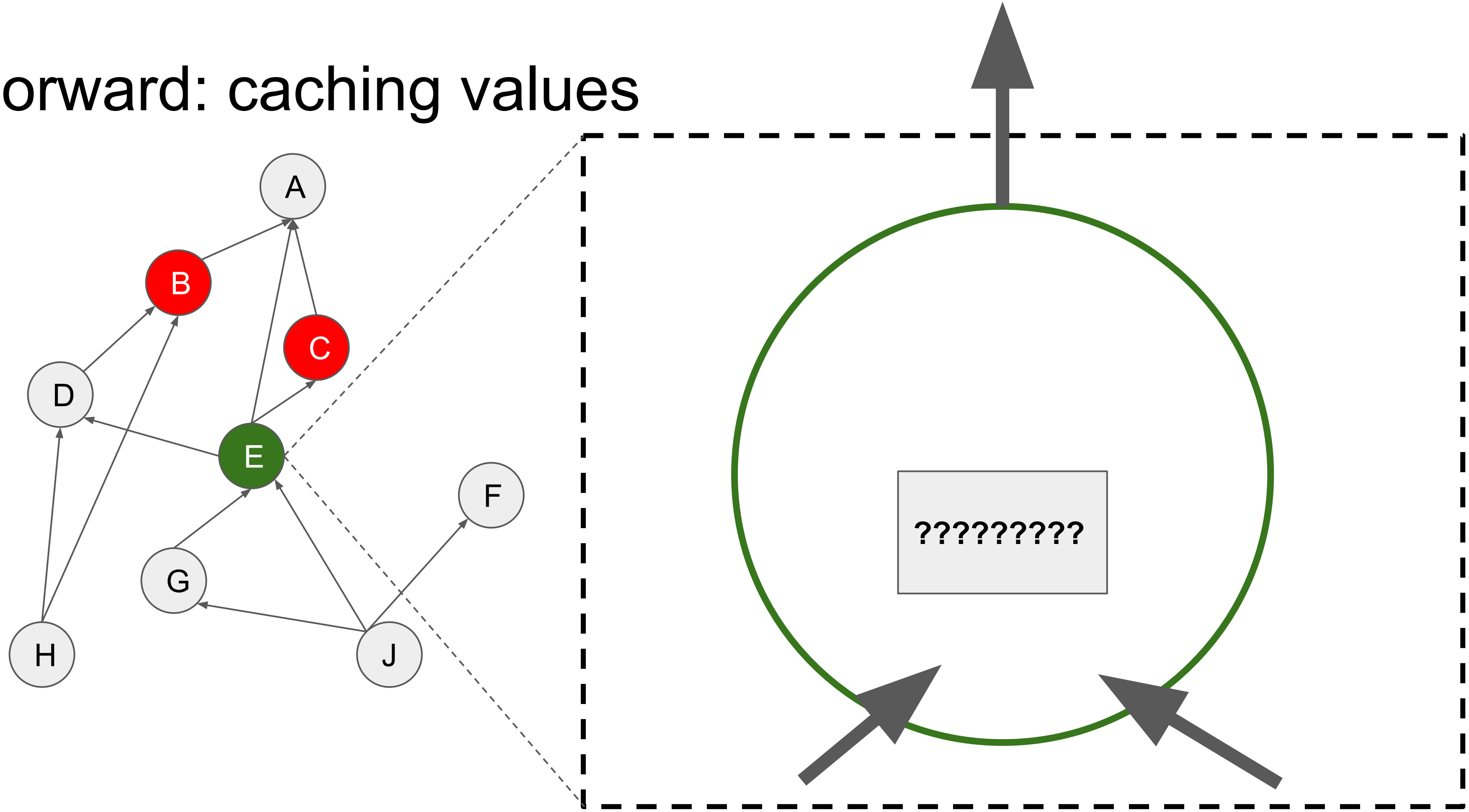

```
Session.run([B, C])

??
for node in [B, C]:
    Session.run(node)
```

Forward: caching values

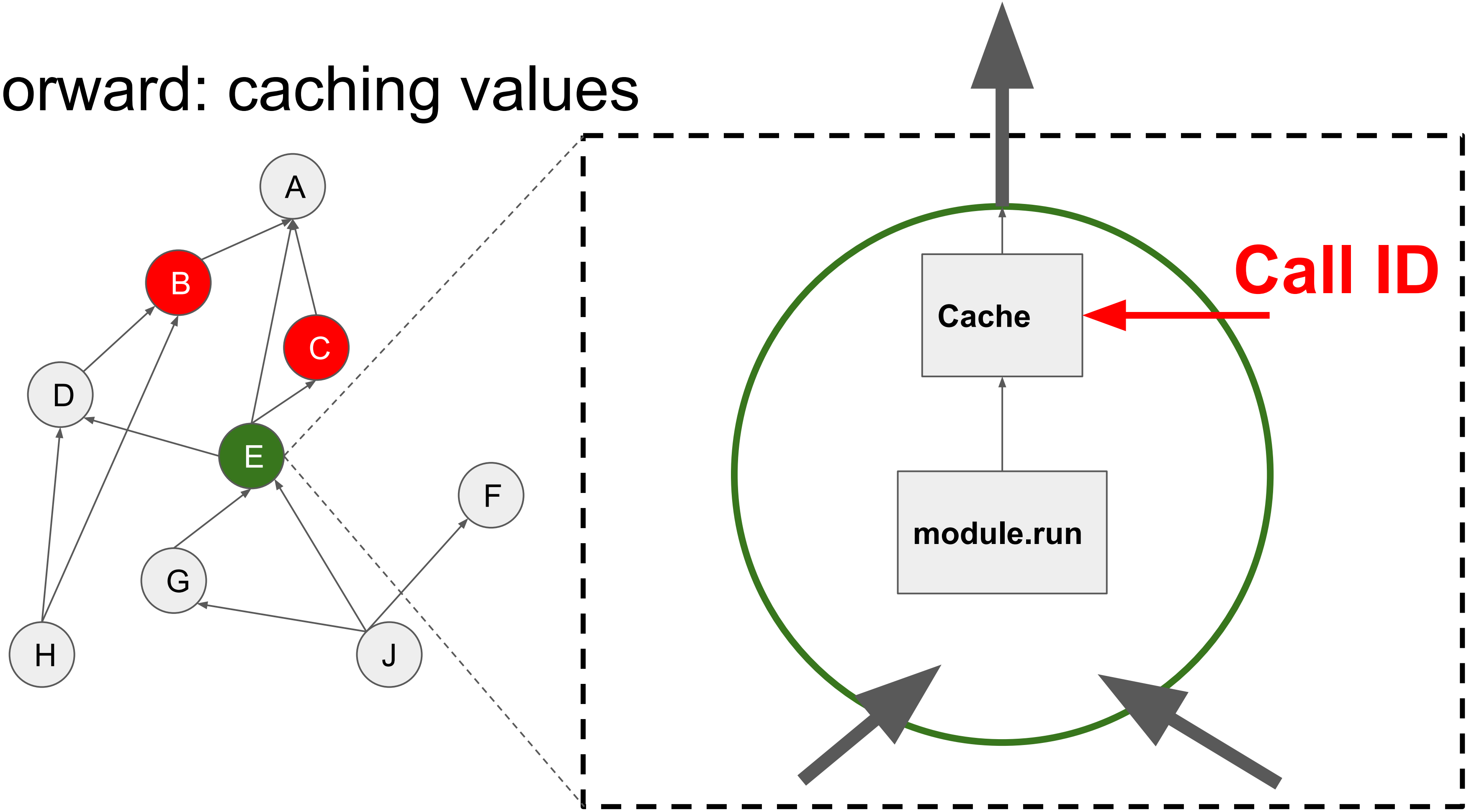



```
Session.run([B, C])

??
for node in [B, C]:
    Session.run(node)

==> E got evaluated
twice!
```

Forward: caching values

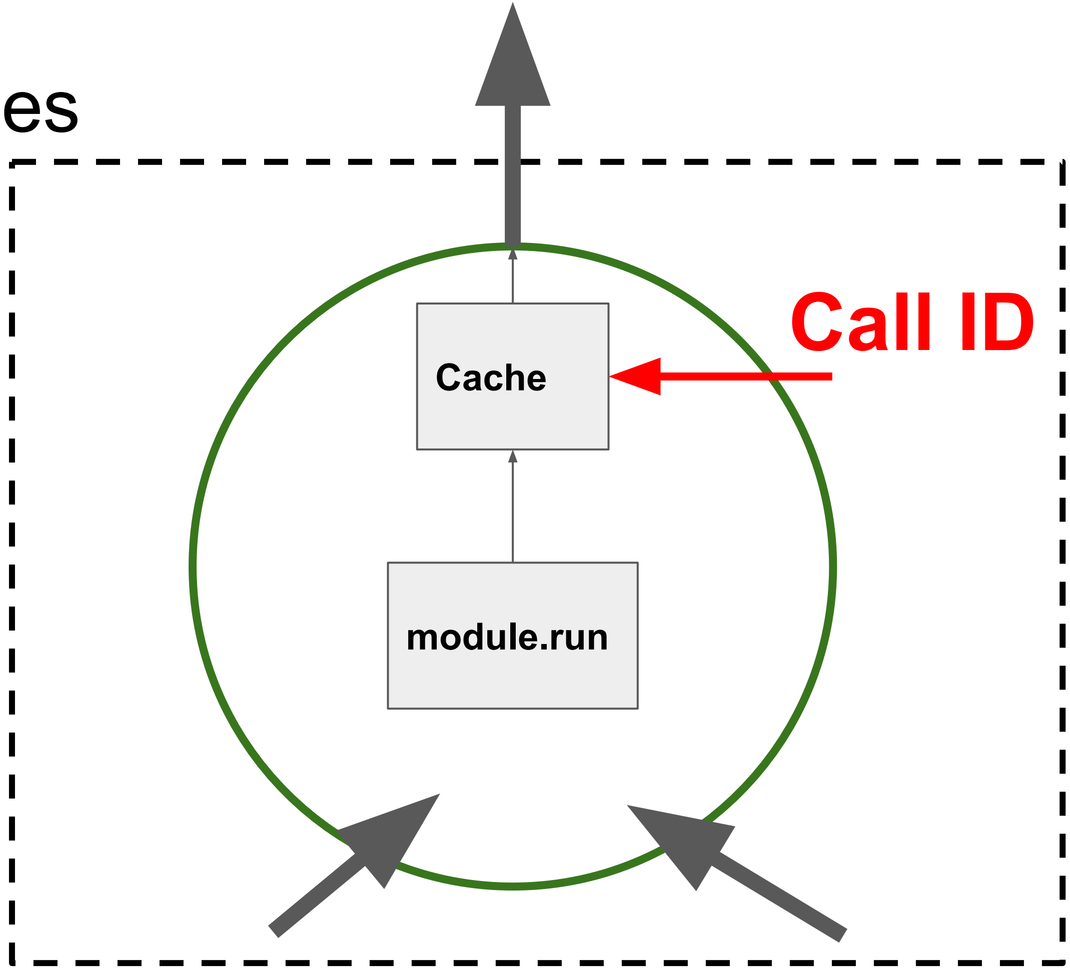


Forward: caching values

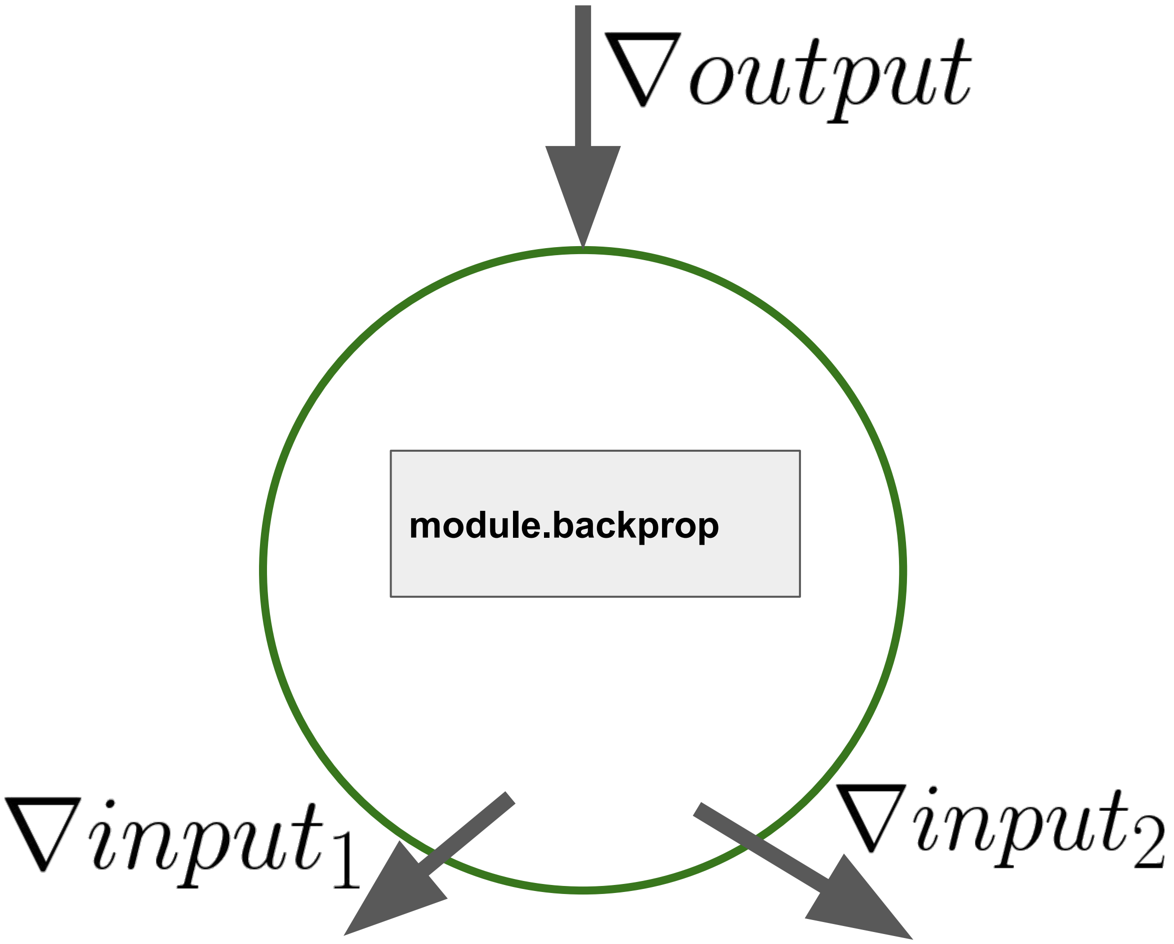


Forward: caching values

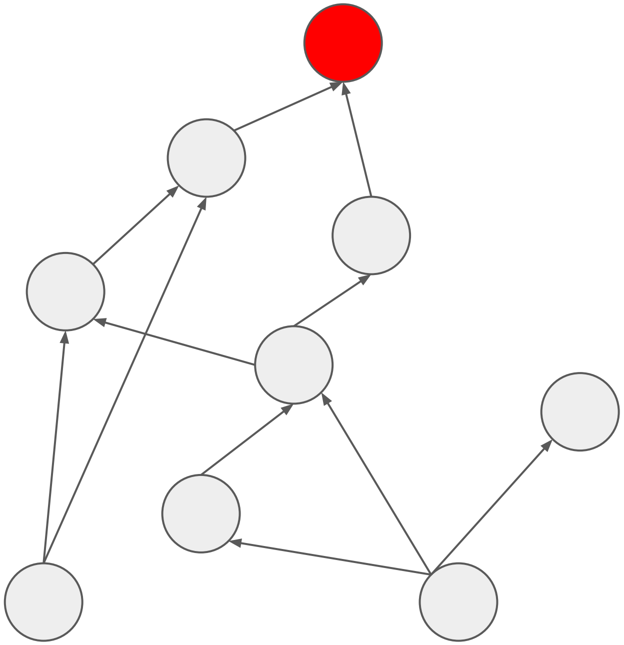
https://gist.github.com/thtrieu/3c318ed471fd827ec1c3ff774048db6e

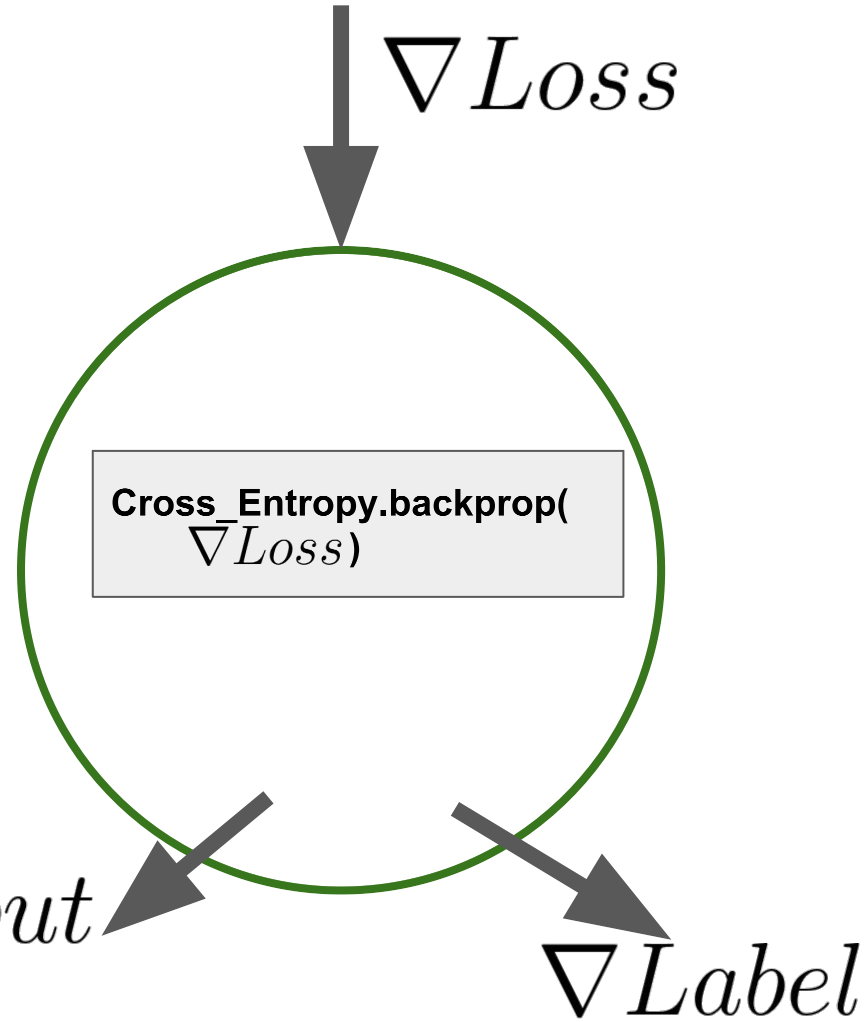


Backward

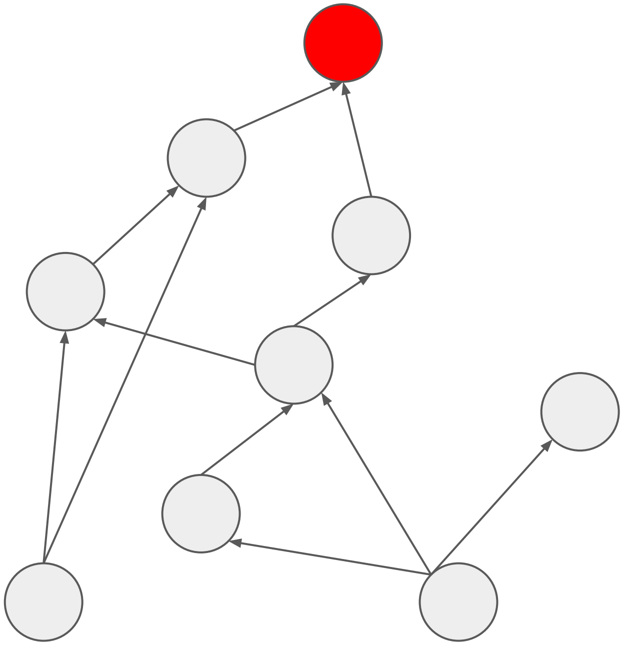


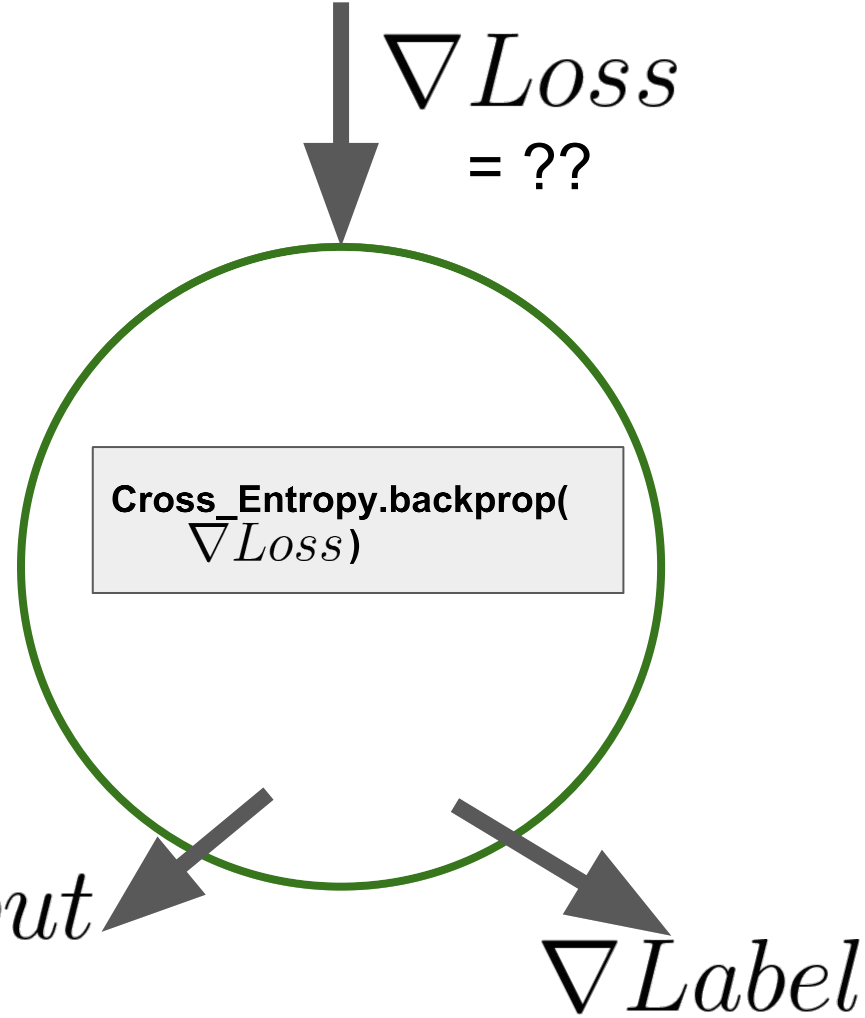
Cross Entropy Module



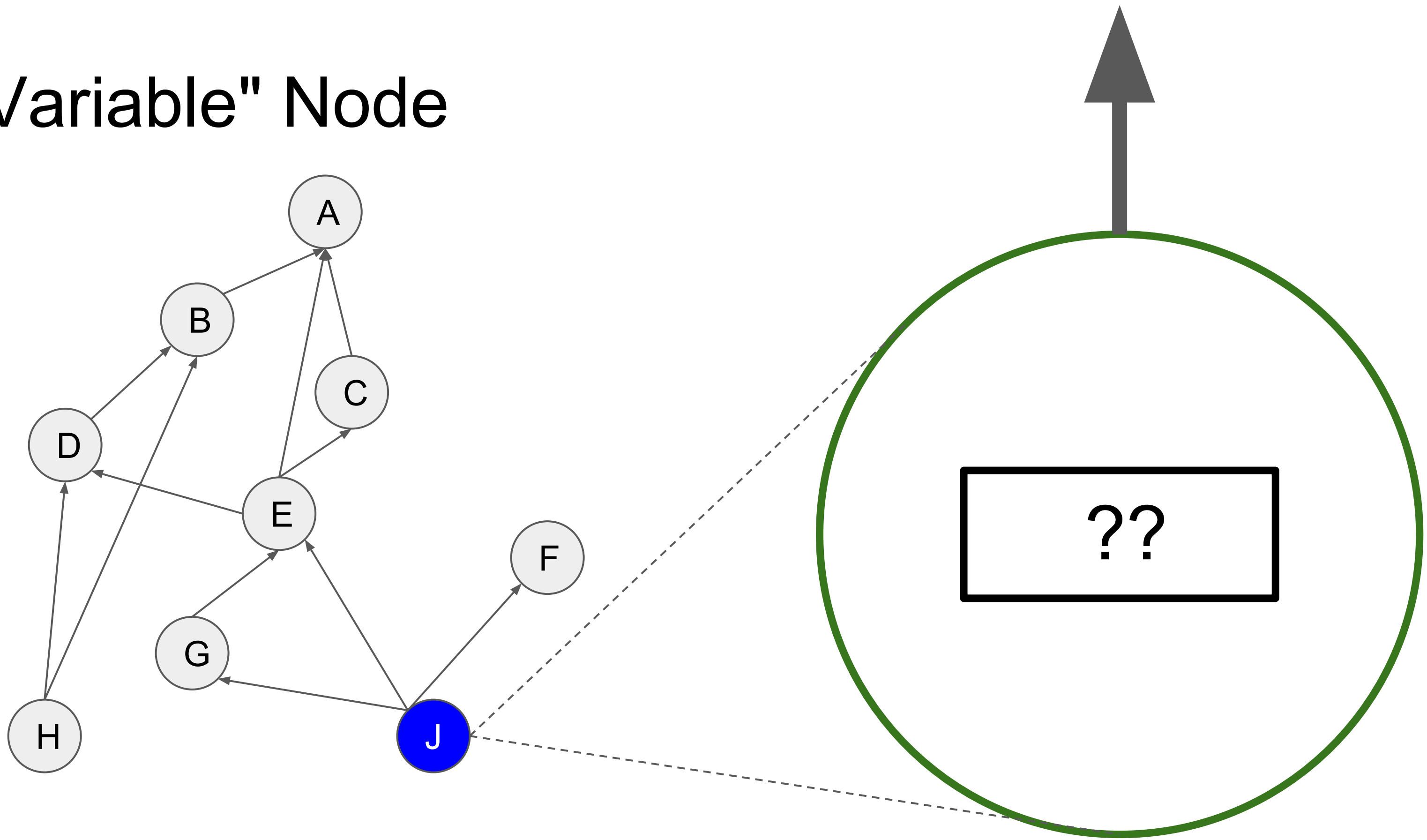


Cross Entropy Module

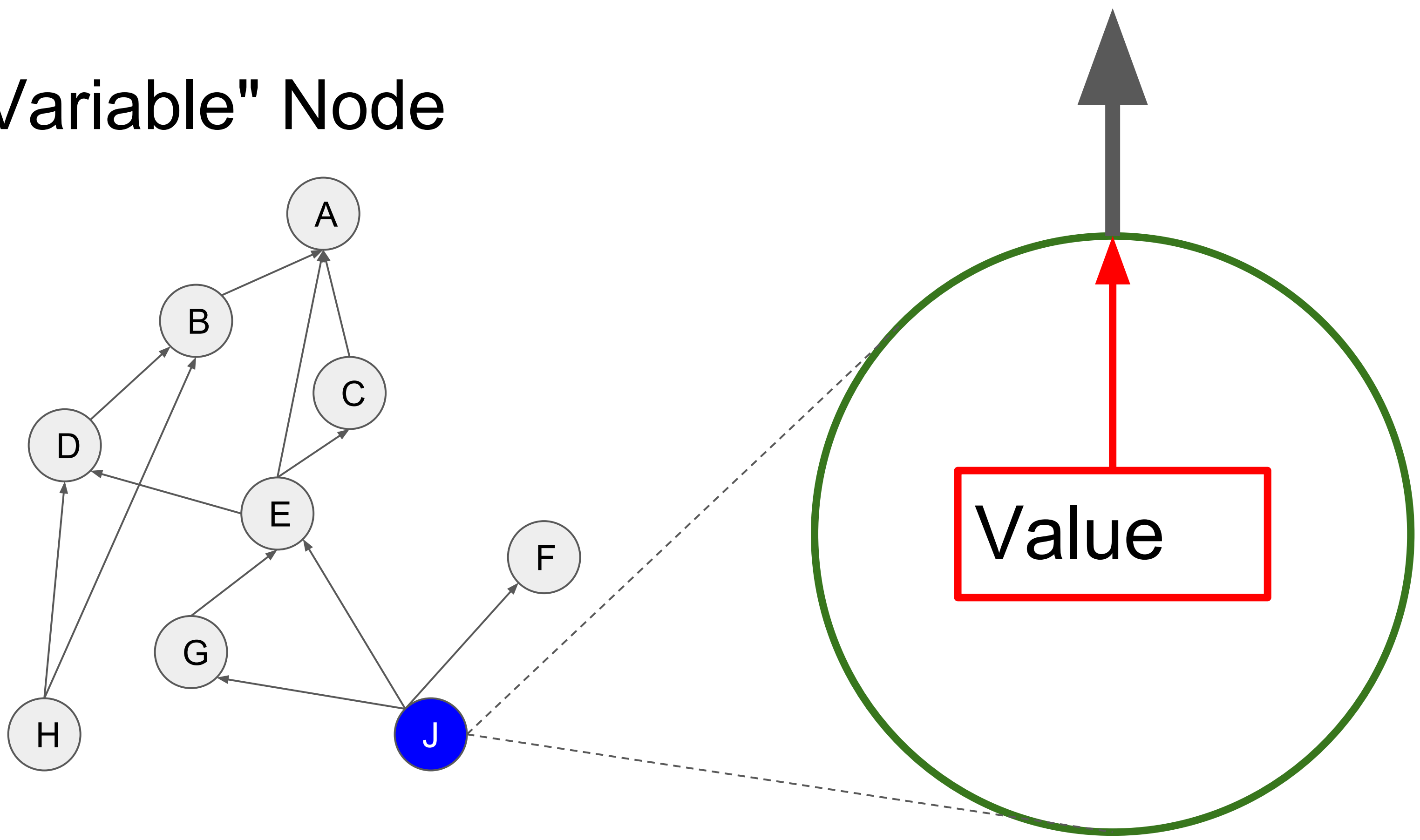




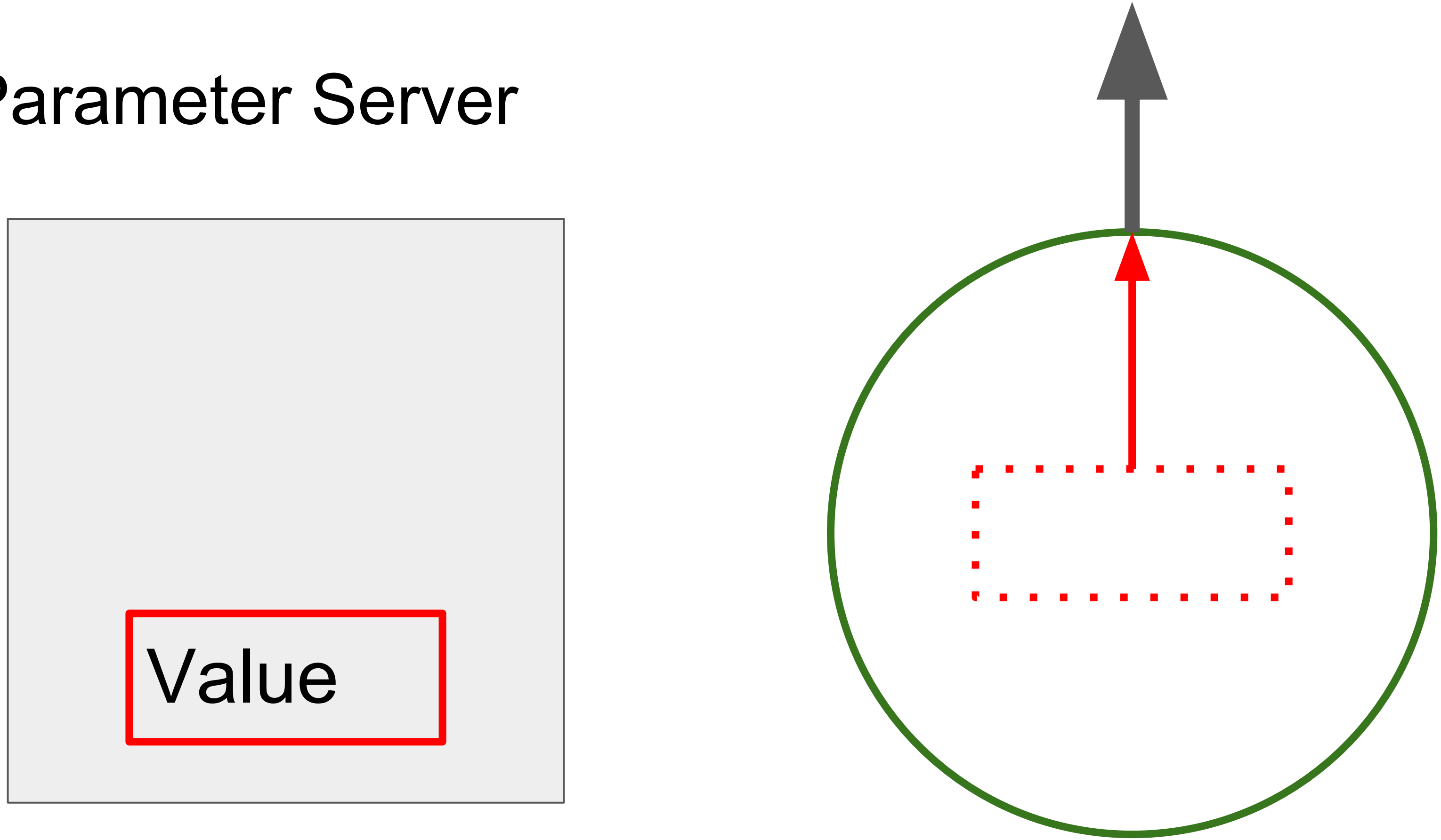
"Variable" Node

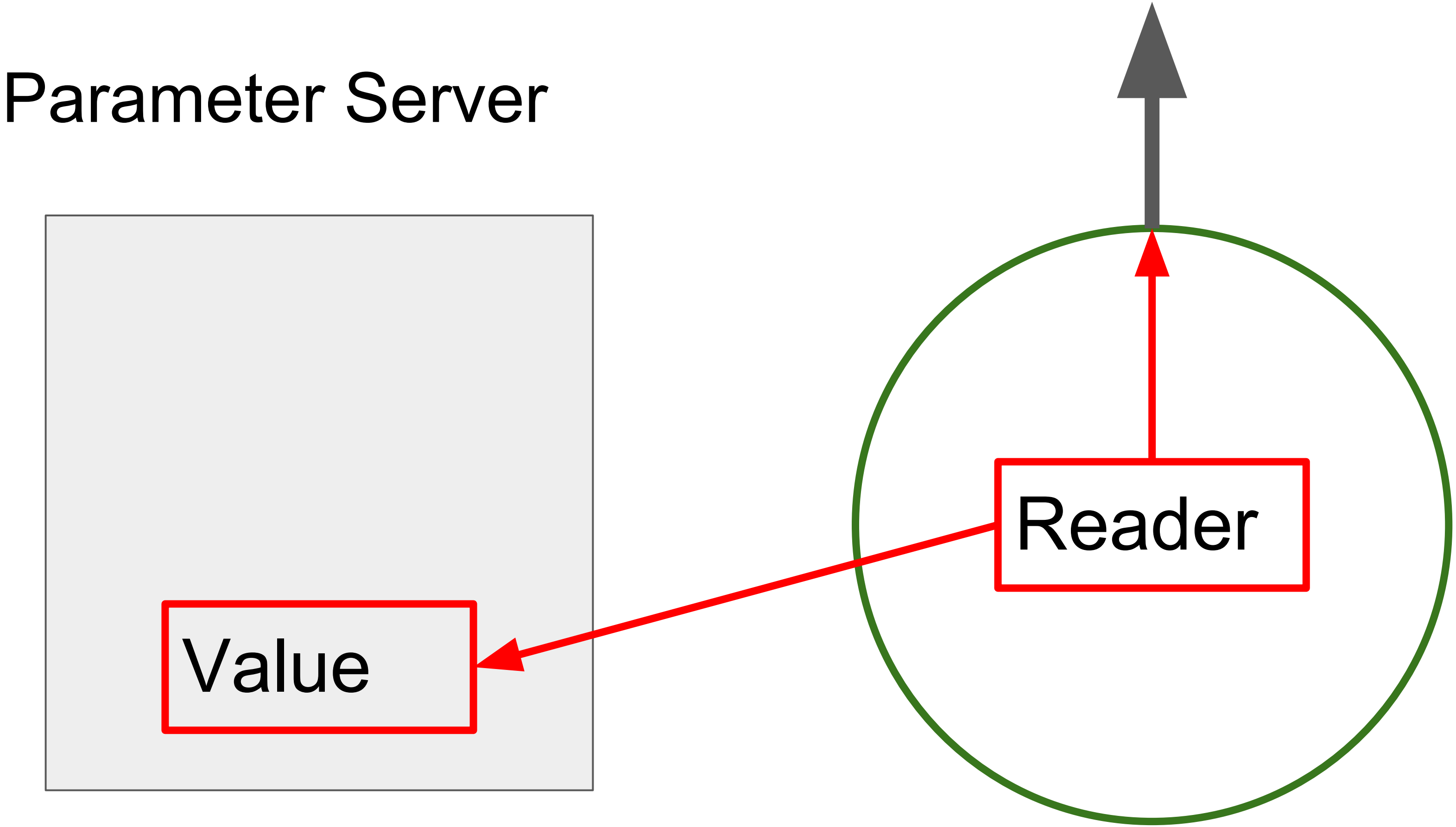


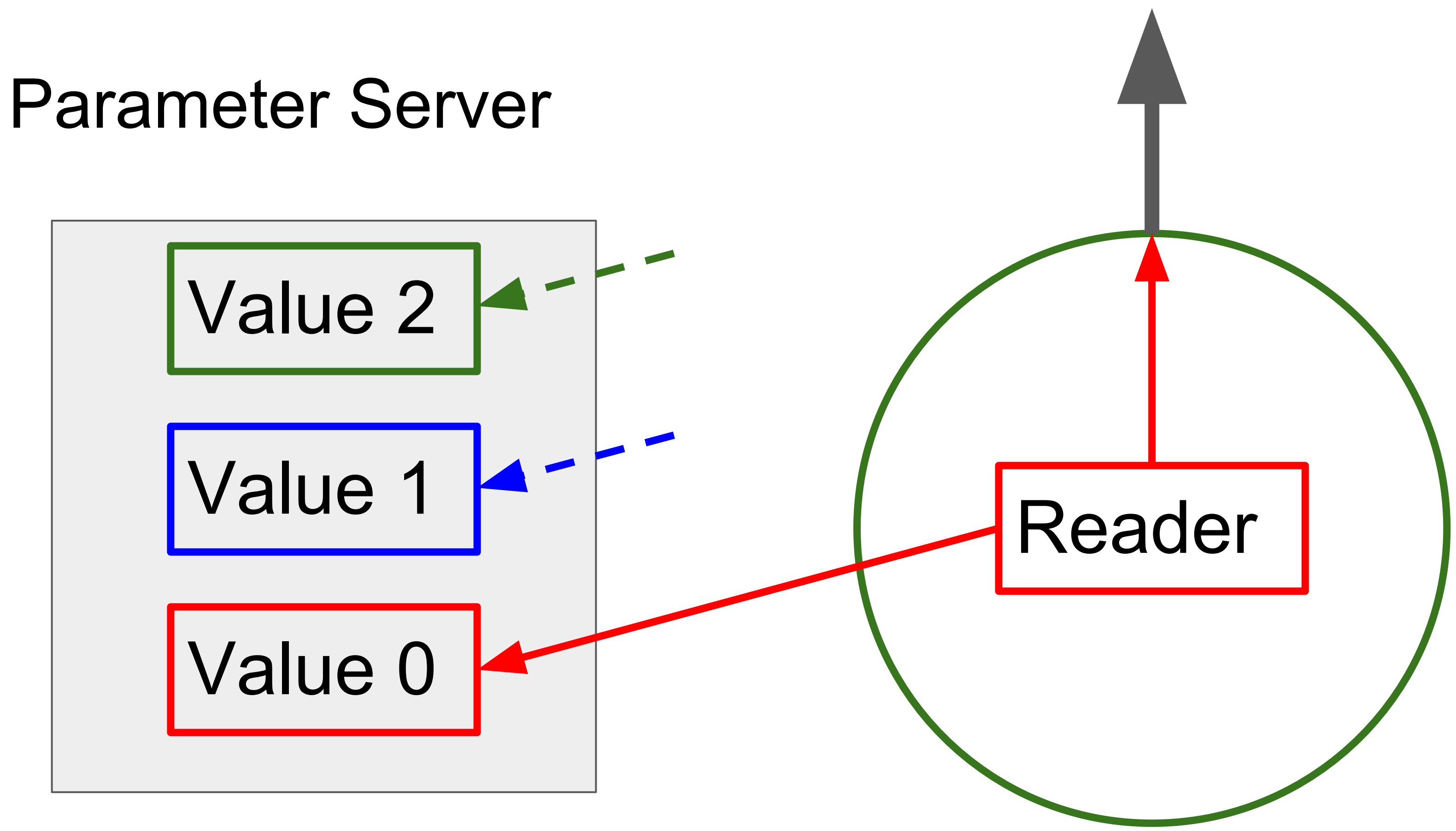
"Variable" Node

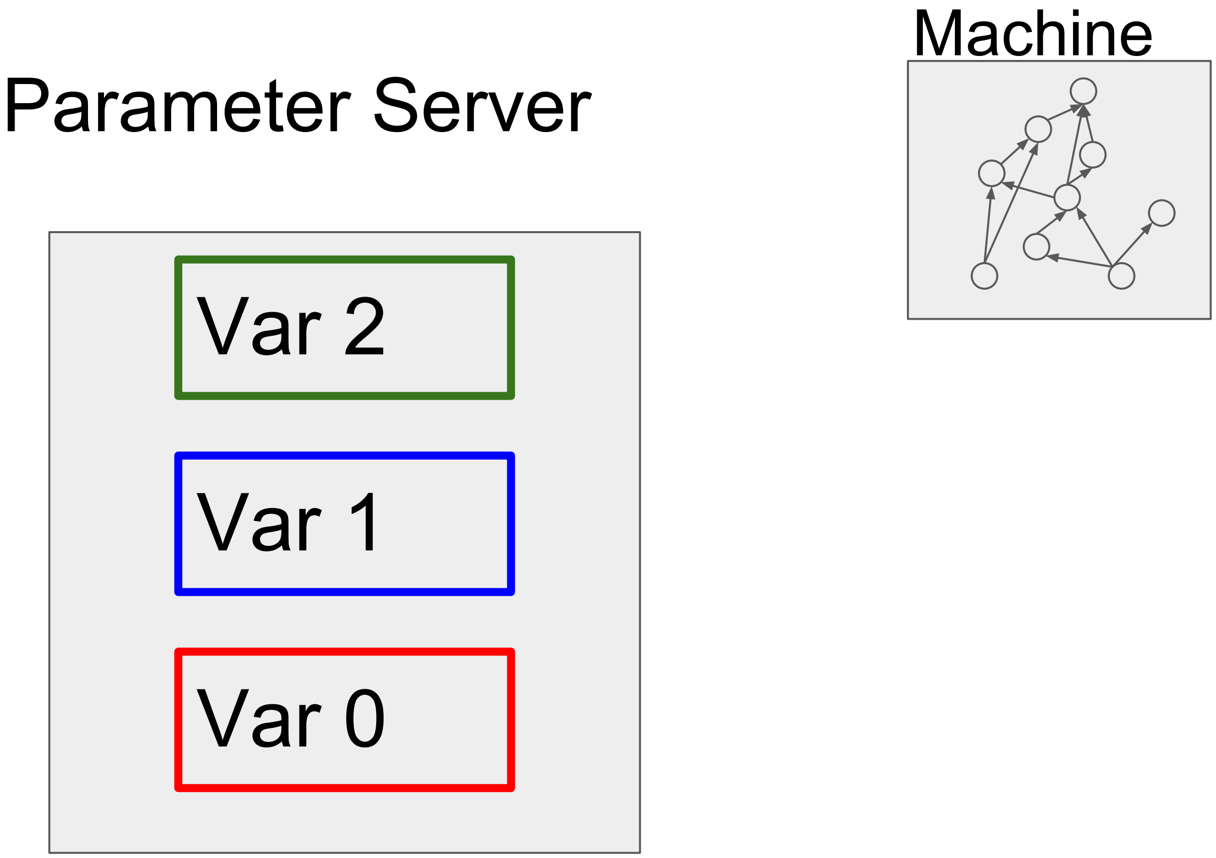


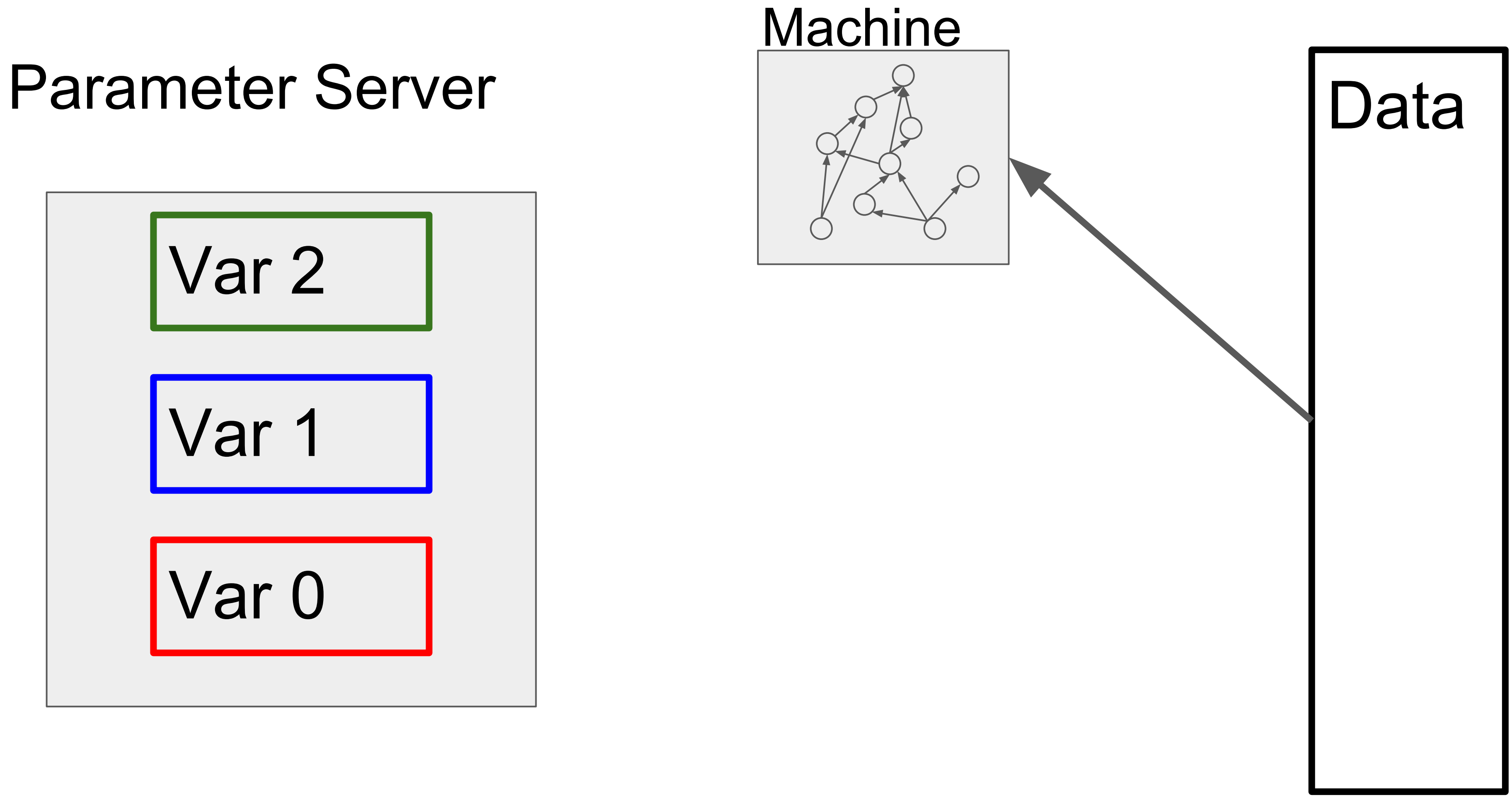
Parameter Server

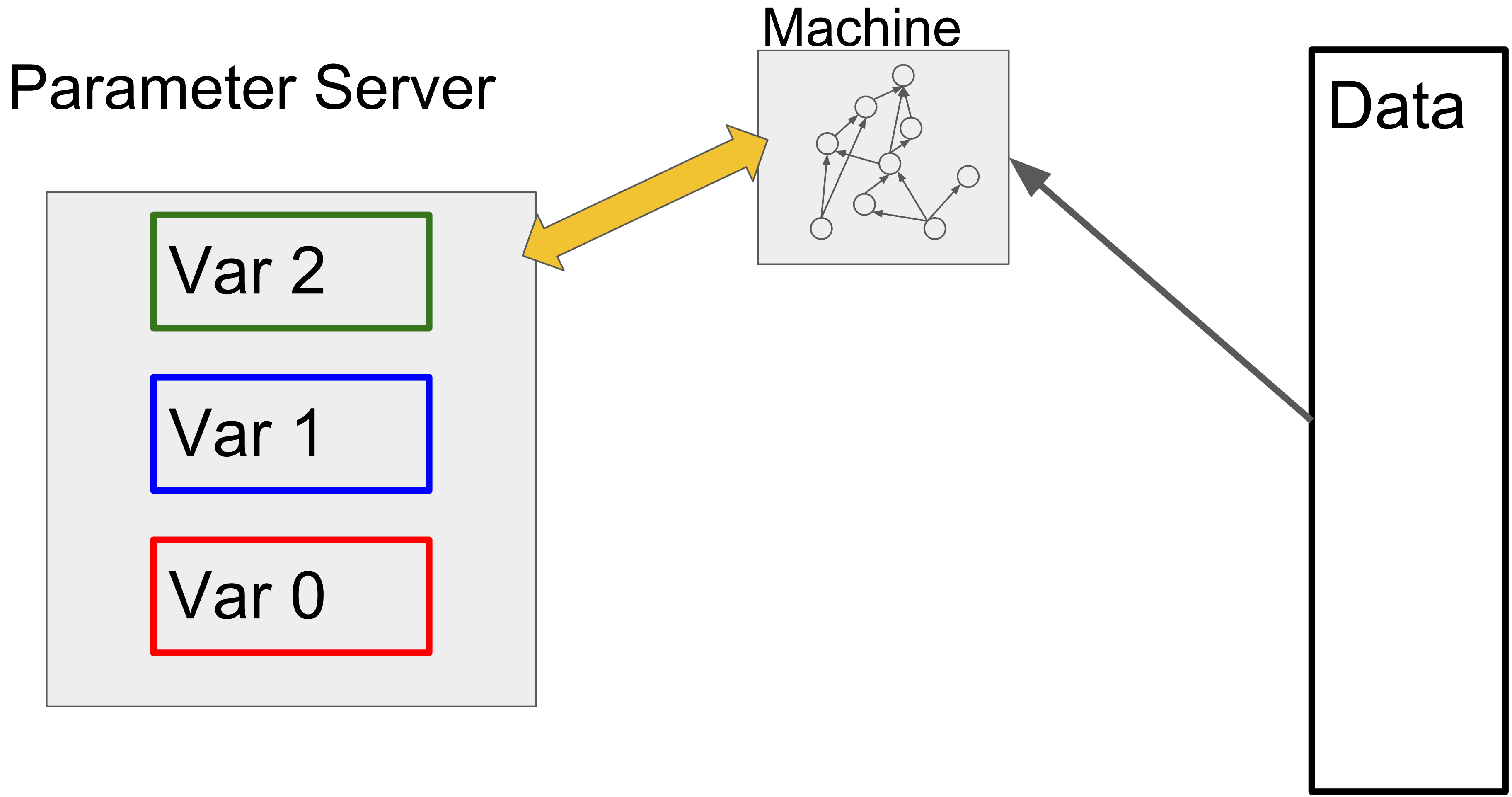


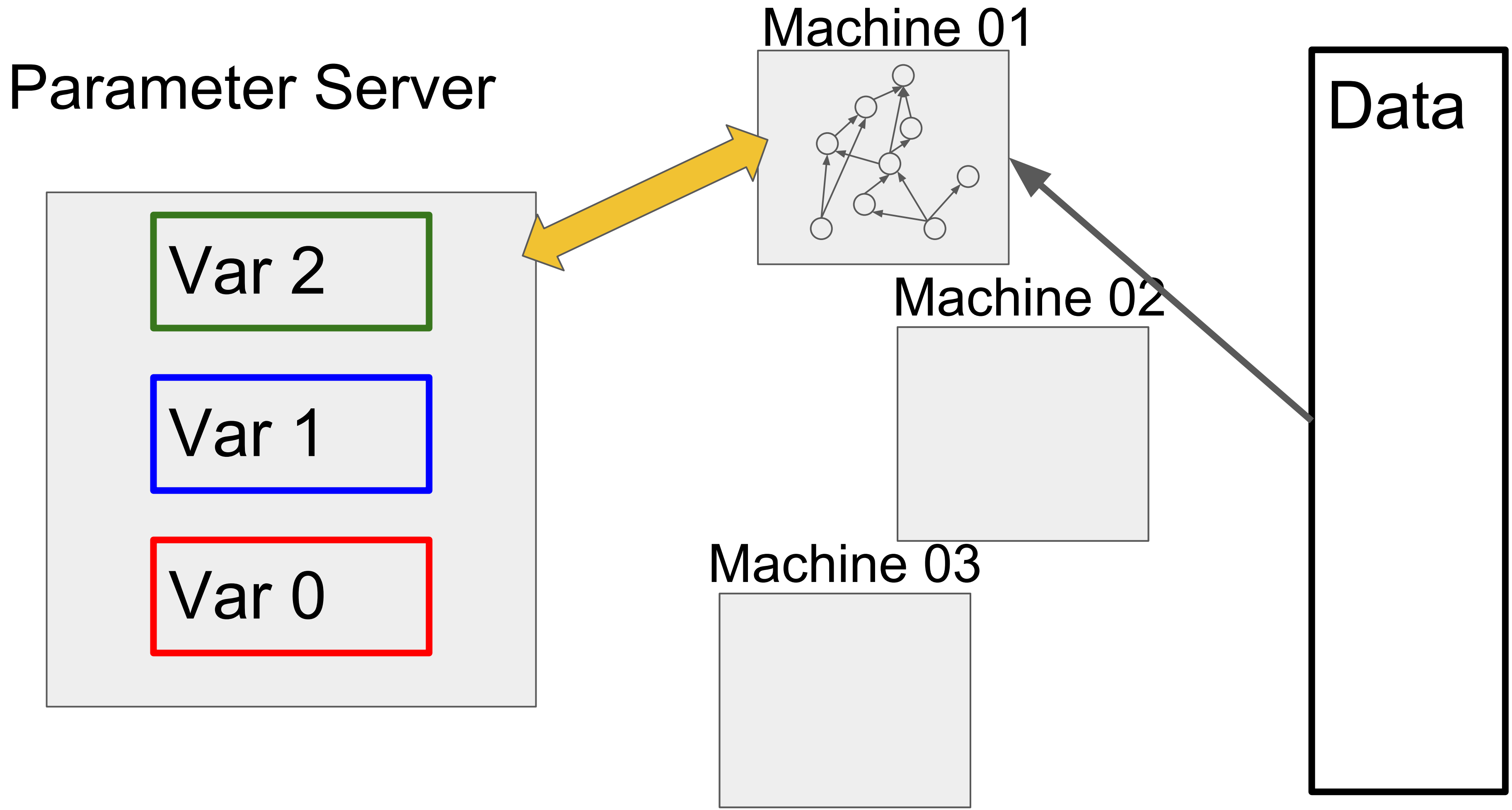

Parameter Server
Value
Reader


Parameter Server
Value 2
Value 1
Value 0
Reader

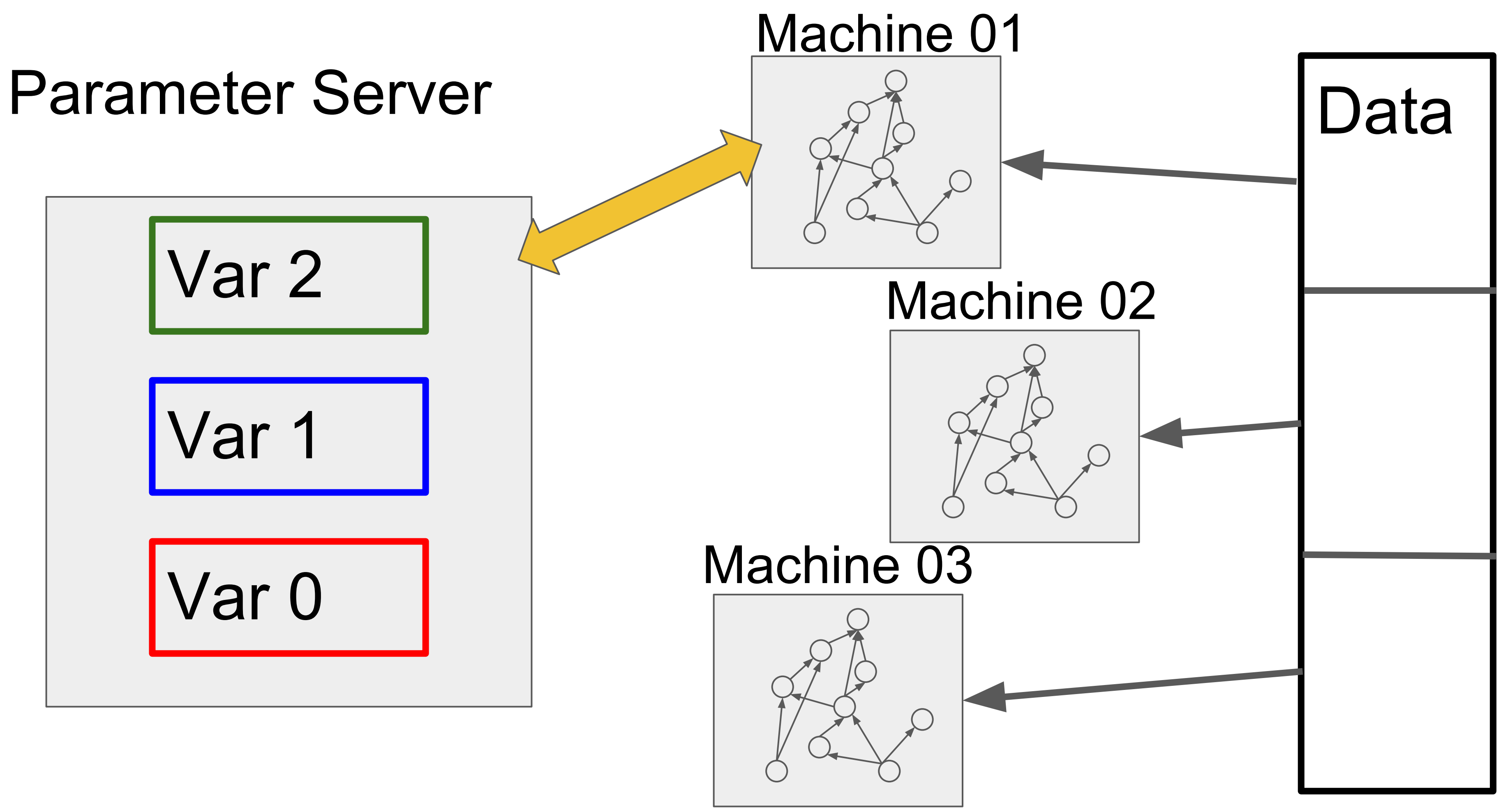

Parameter Server
Var 2
Var 1
Var 0
Machine


Parameter Server
Var 2
Var 1
Var 0
Machine
Data

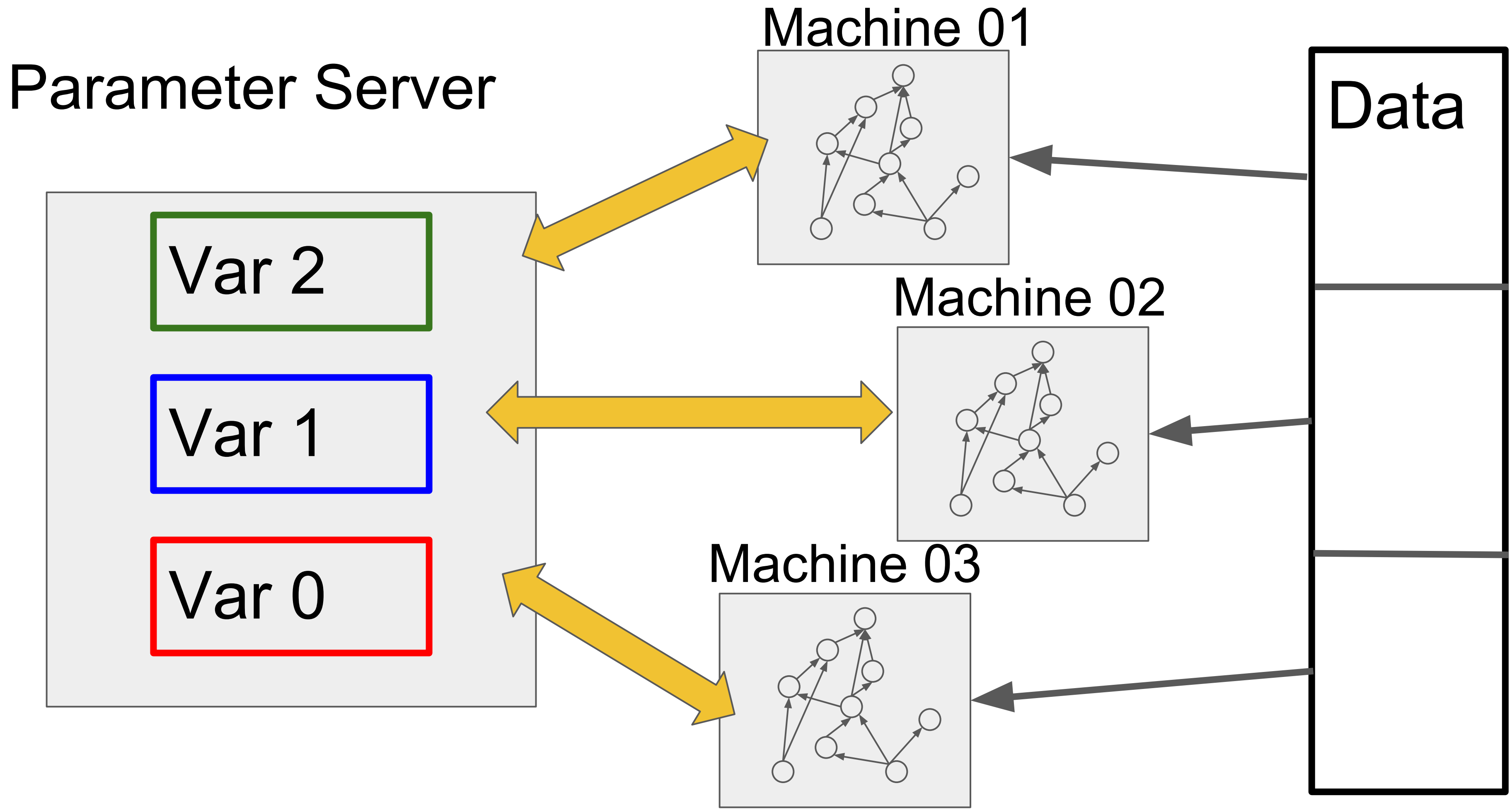

Parameter Server
Var 2
Var 1
Var 0
Machine
Data



Parameter Server
Var 2
Var 1
Var 0
Machine 01
Machine 02
Machine 03
Data



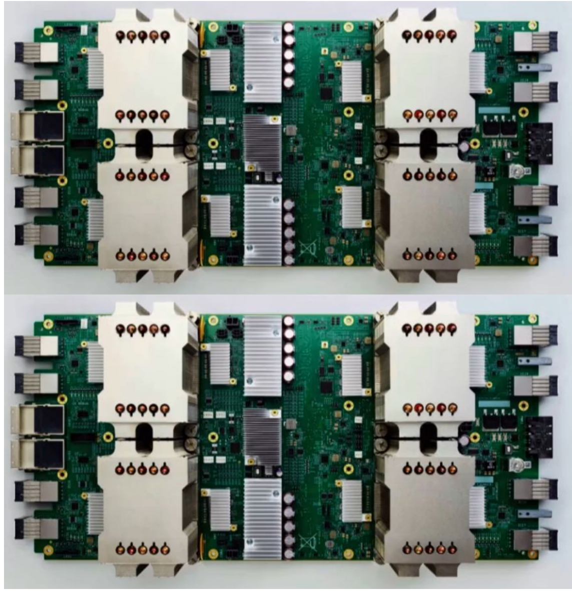
Parameter Server
Var 2
Var 1
Var 0
Machine 01
Machine 02
Machine 03
Data


Parameter Server
Var 2
Var 1
Var 0
Machine 01
Machine 02
Machine 03
Data

Stories time

Stories time

TPU

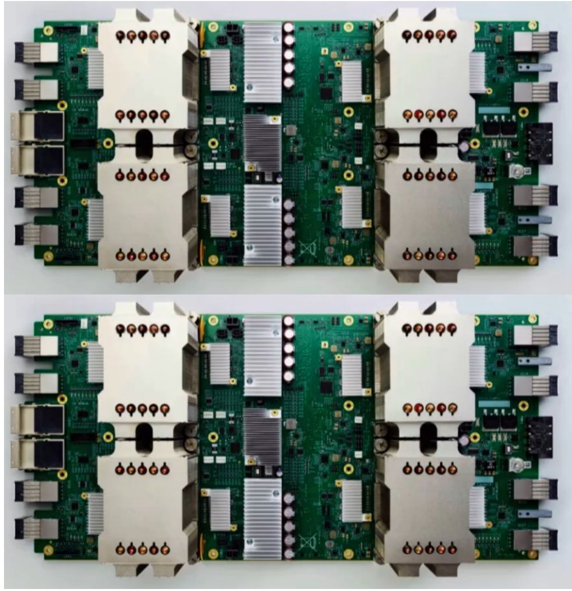


Stories time

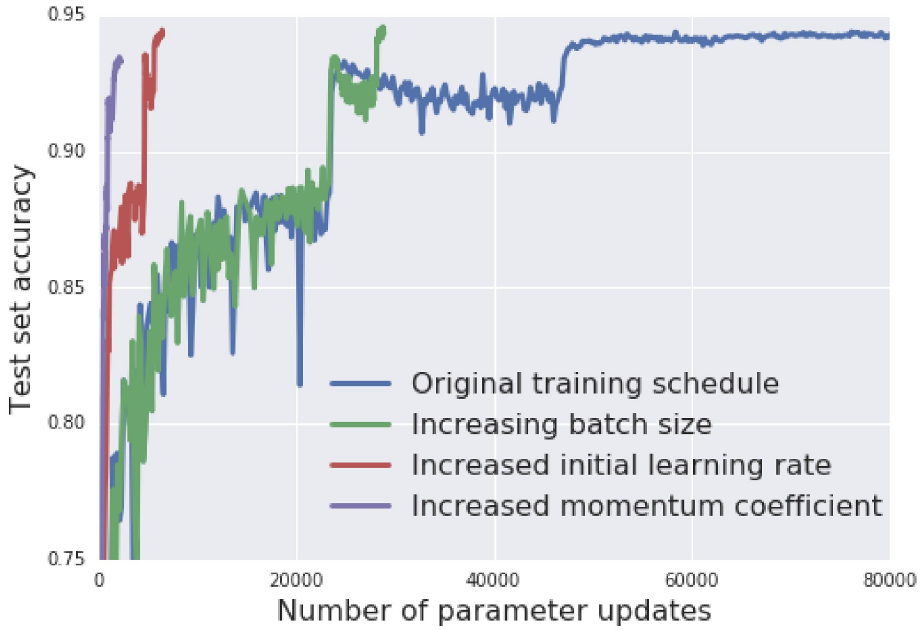
DON'T DECAY THE LEARNING RATE,
INCREASE THE BATCH SIZE

Samuel L. Smith*, Pieter-Jan Kindermans*, Chris Ying & Quoc V. Le
Google Brain

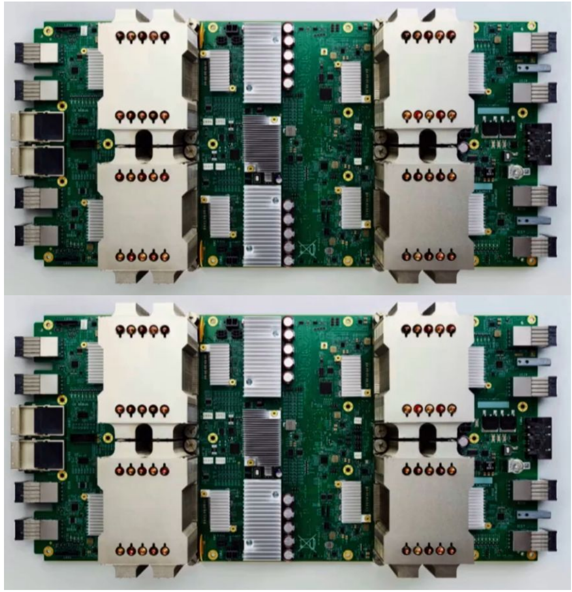
TPU



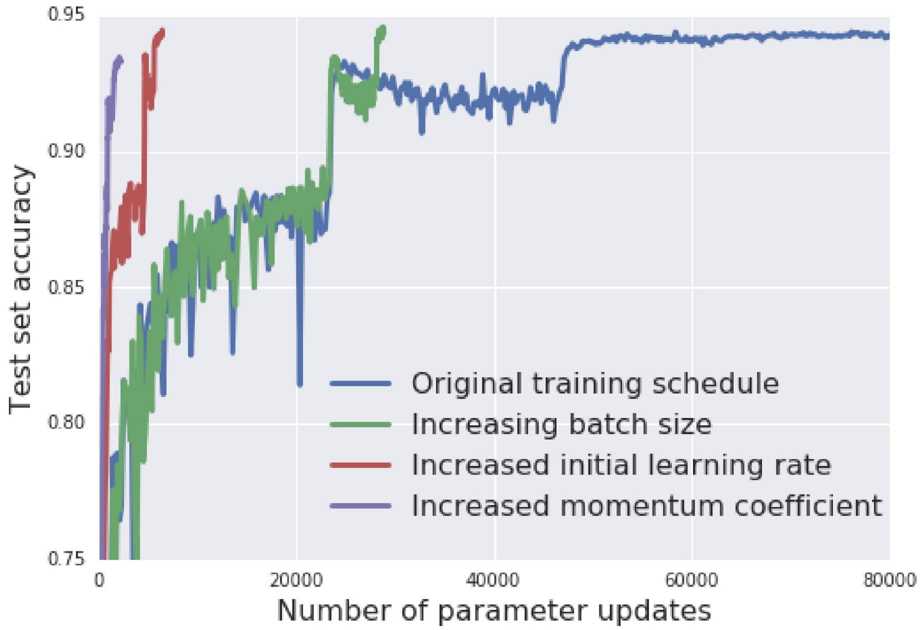
Stories time



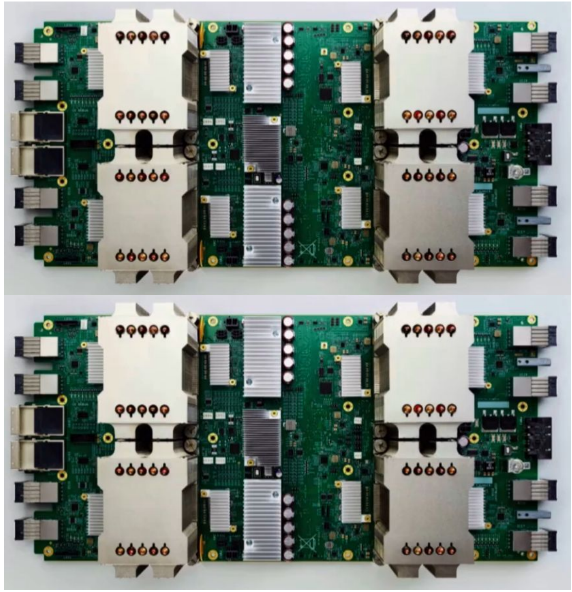
TPU

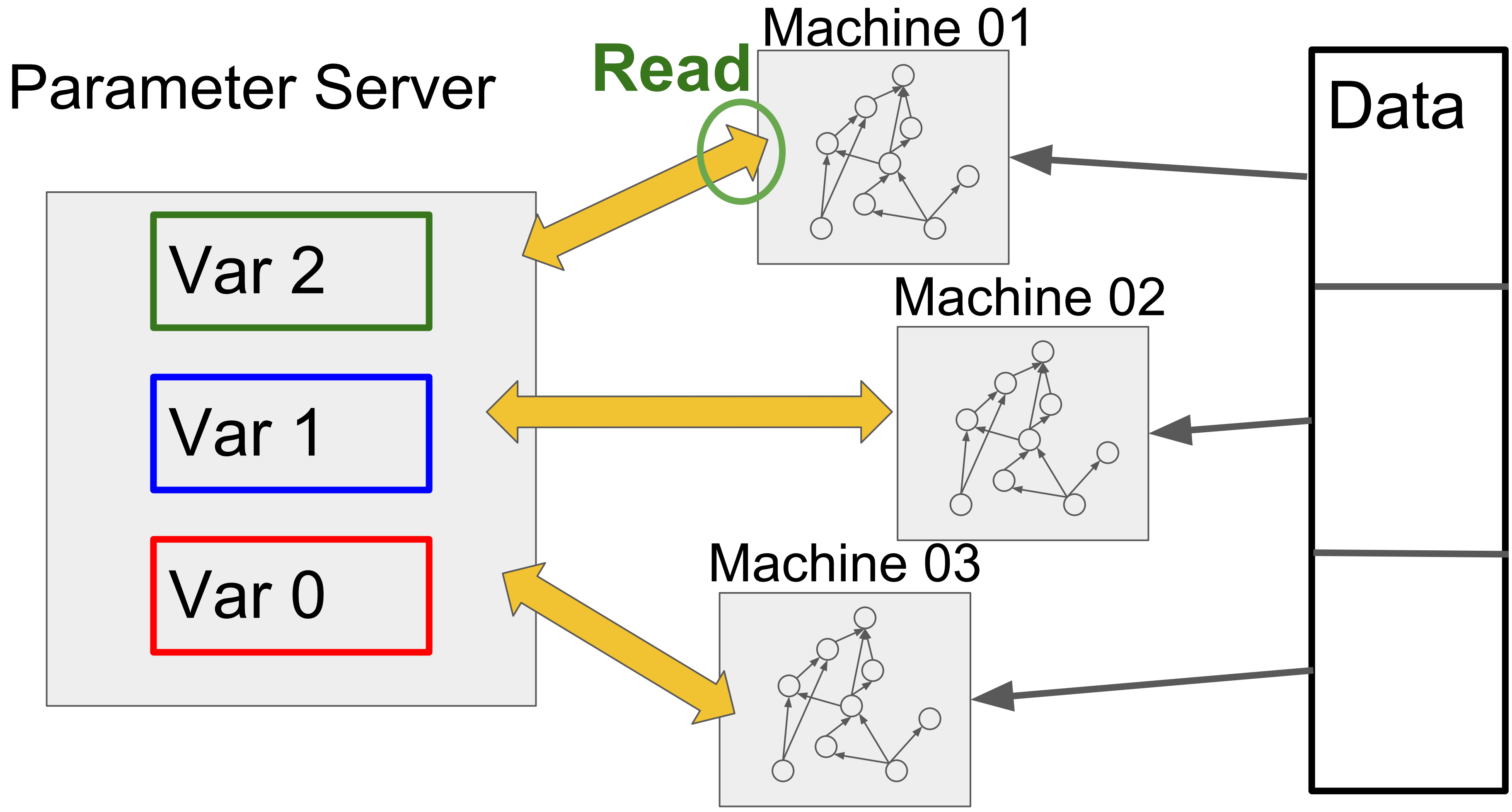


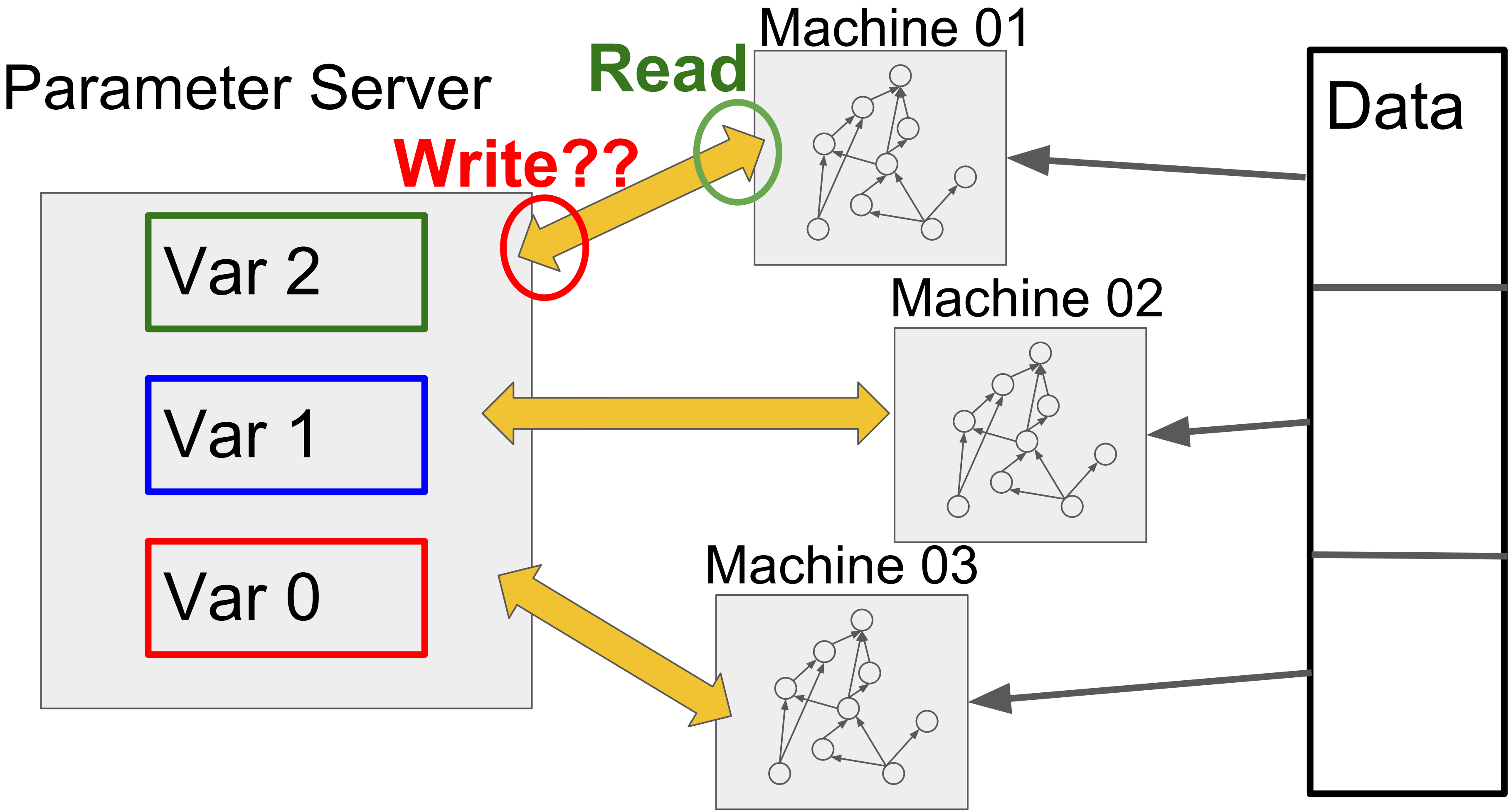
Stories time



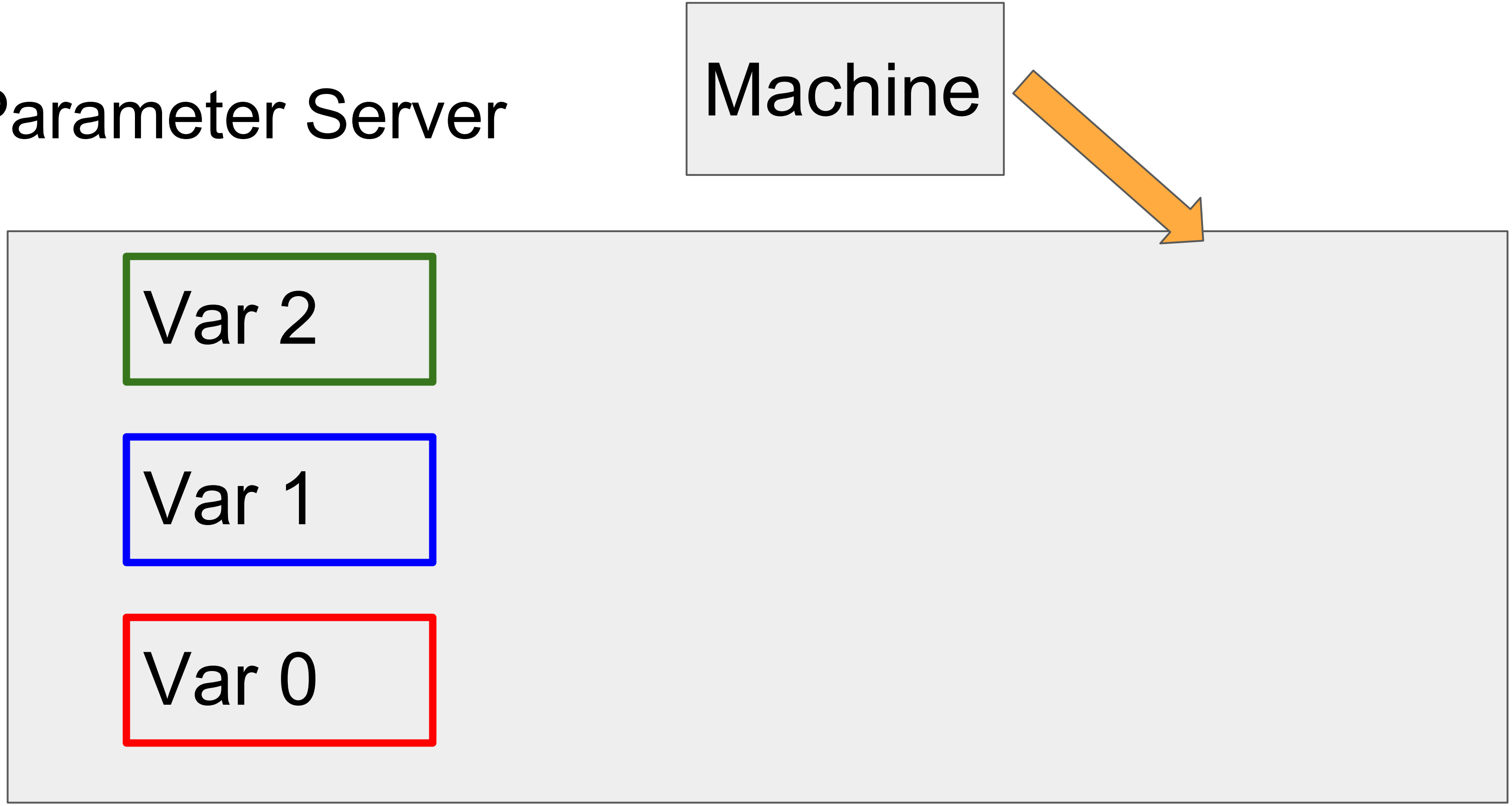
TPU

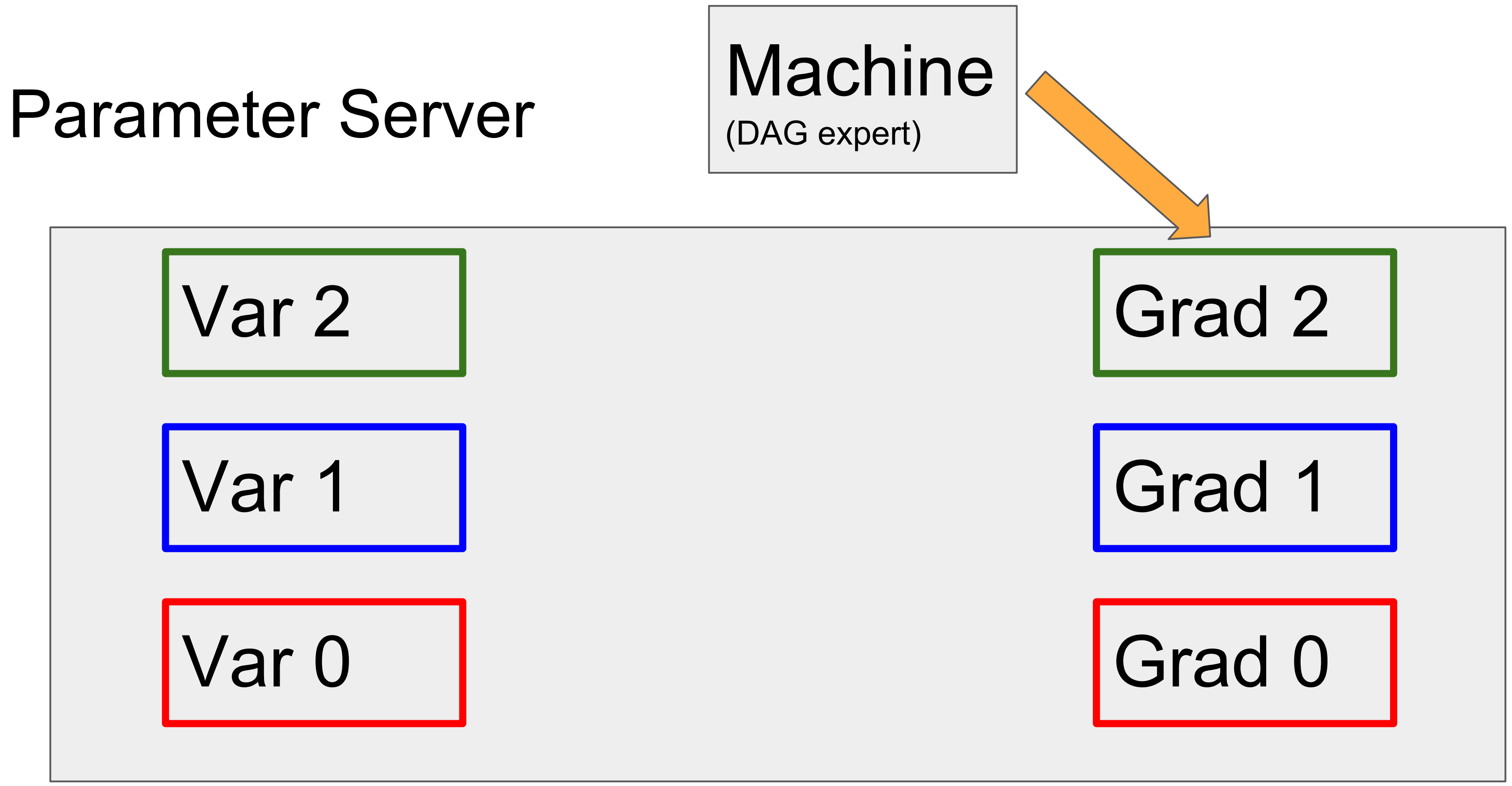



Parameter Server
Read
Machine 01
Var 2
Machine 02
Var 1
Machine 03
Var 0
Data

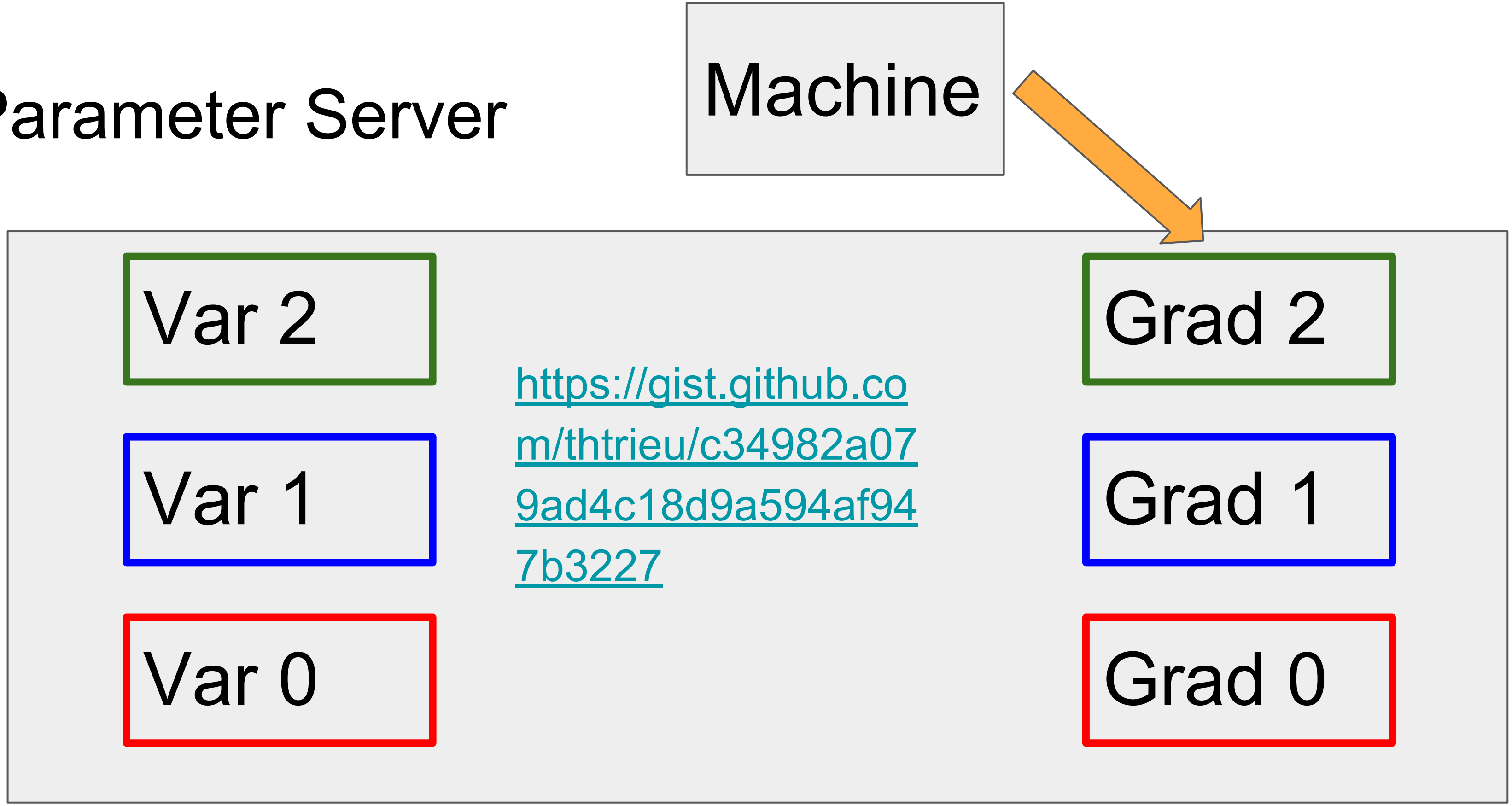

Parameter Server
Write??
Read
Machine 01
Var 2
Var 1
Var 0
Machine 02
Machine 03
Data

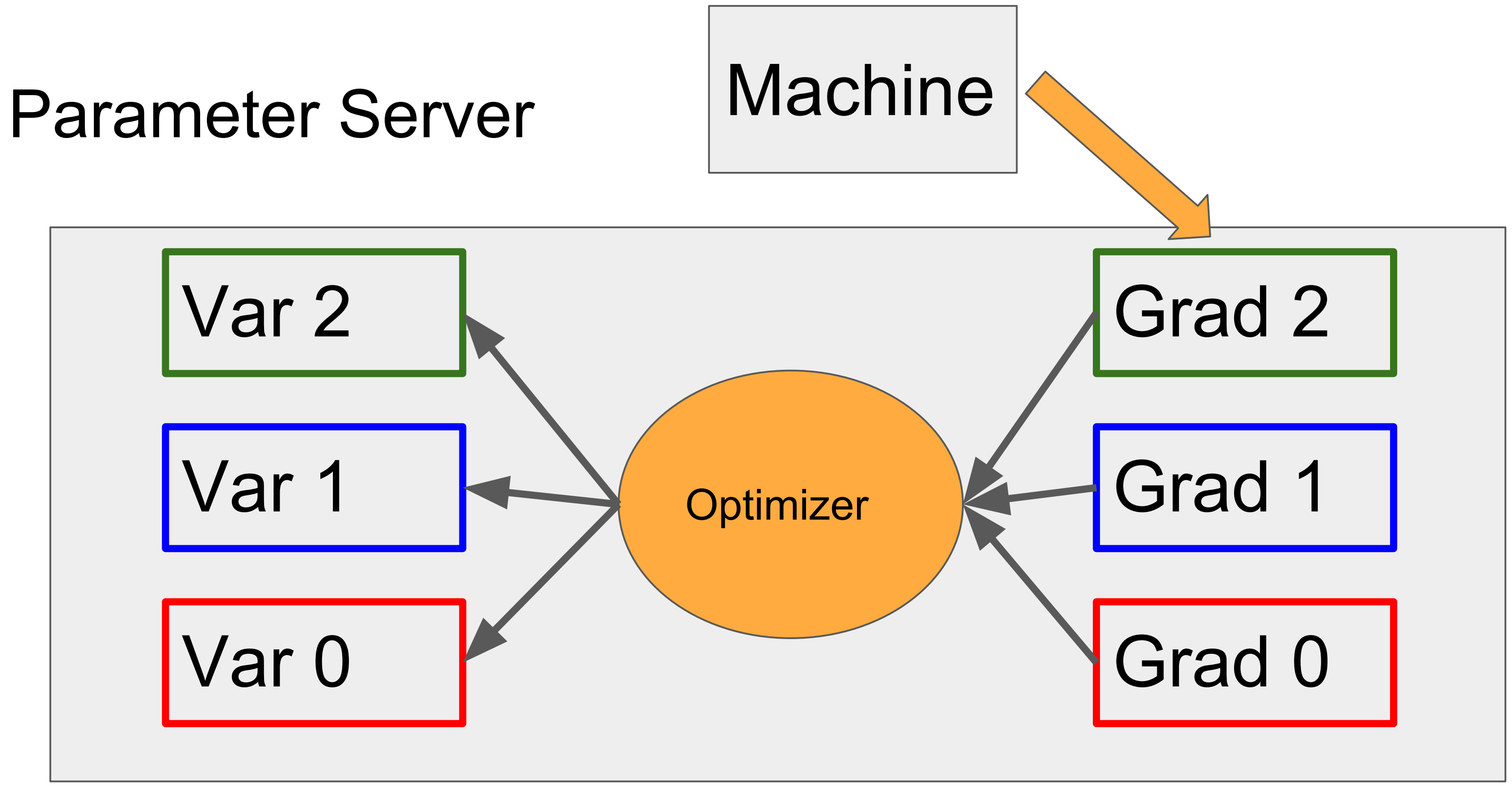
Parameter Server

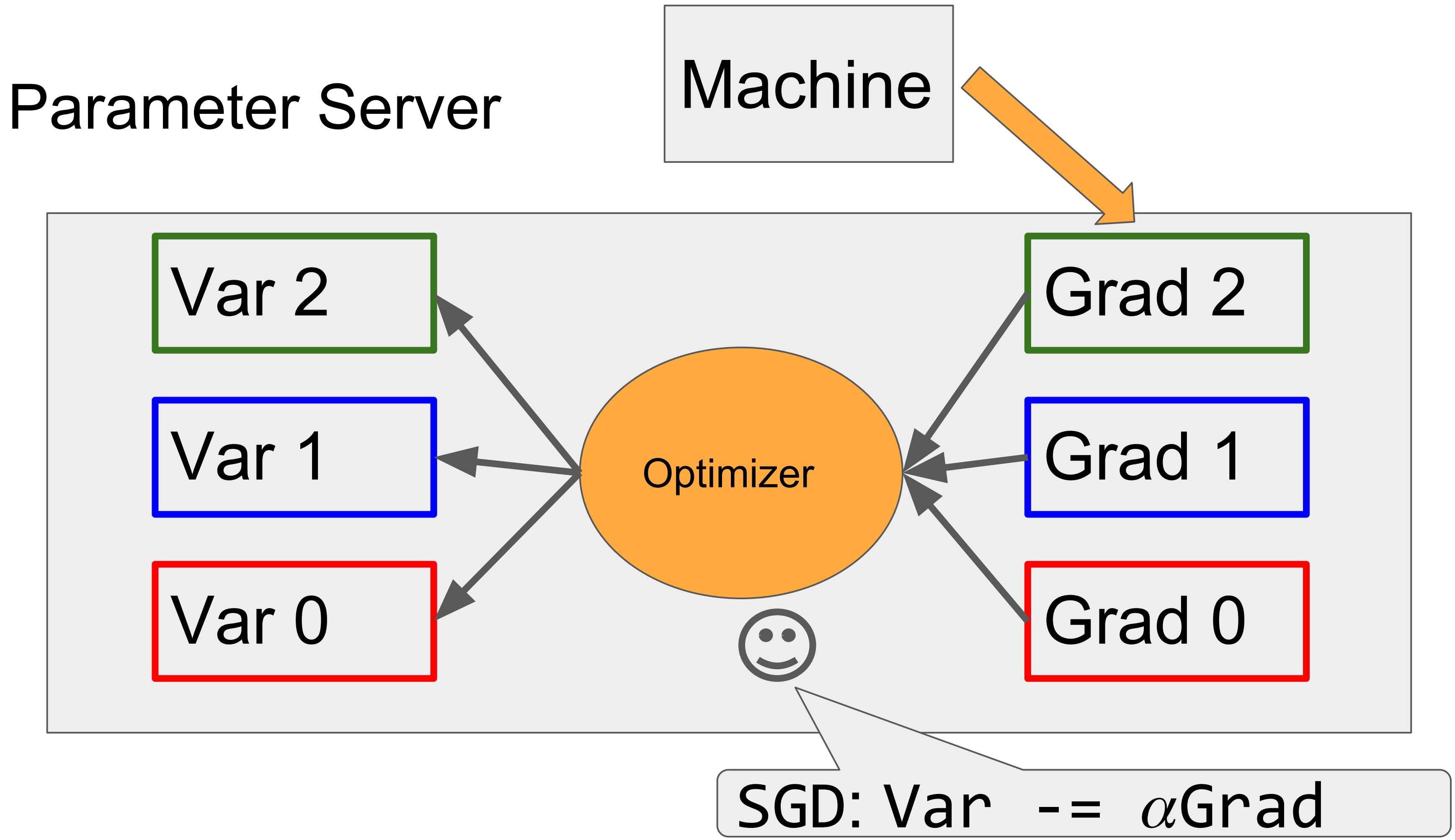



Parameter Server
Machine
(DAG expert)
Var 2
Grad 2
Var 1
Grad 1
Var 0
Grad 0

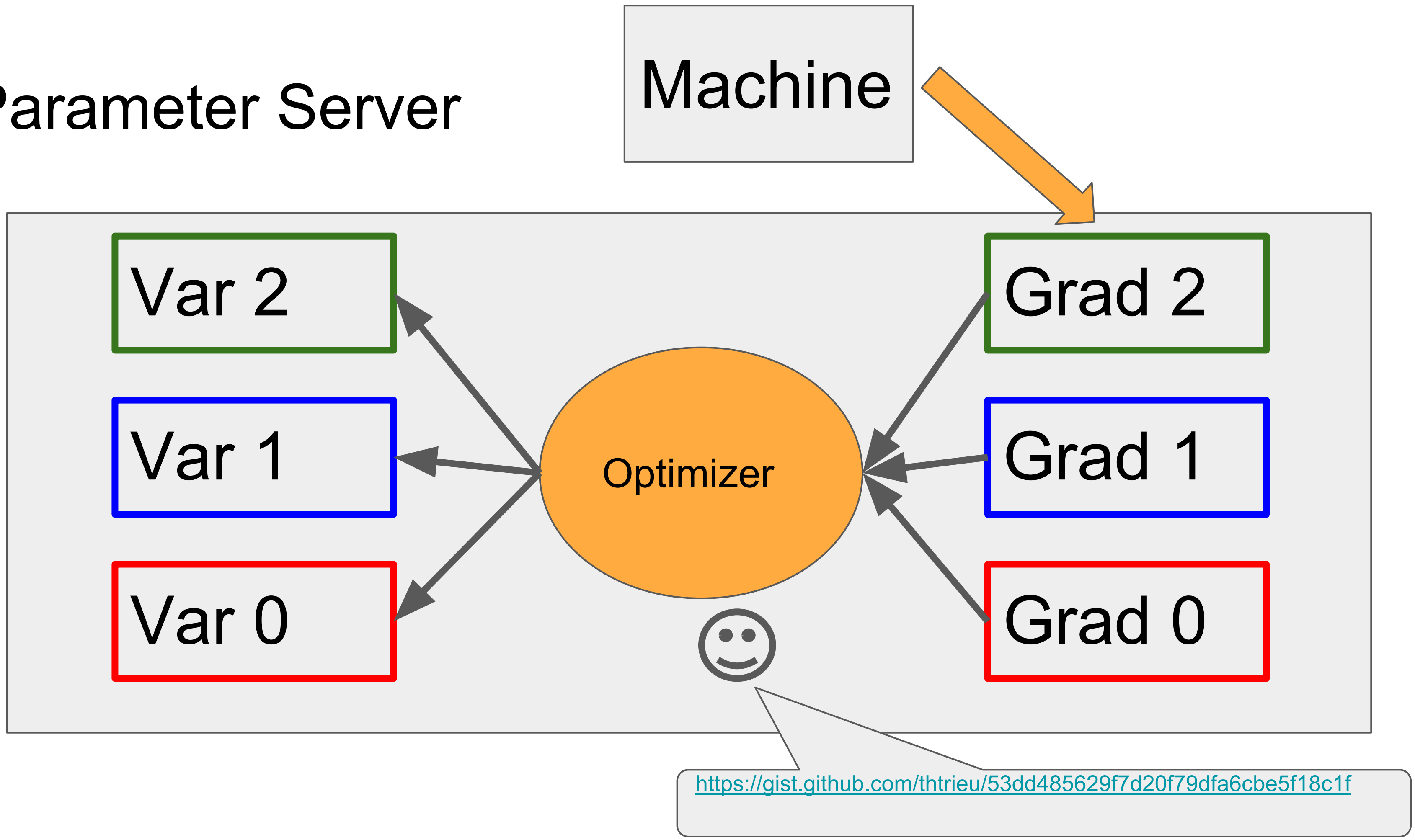
Parameter Server




Parameter Server
Machine
Var 2
Var 1
Var 0
Optimizer
Grad 2
Grad 1
Grad 0

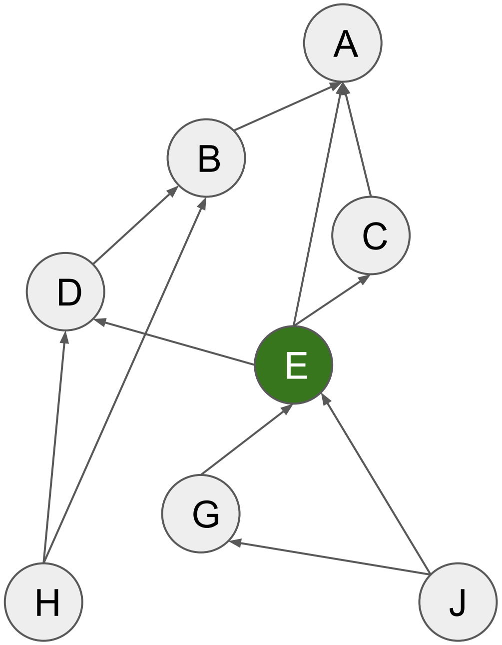

Parameter Server
Machine
Var 2
Var 1
Var 0
Optimizer
Grad 2
Grad 1
Grad 0
SGD: Var -= αGrad

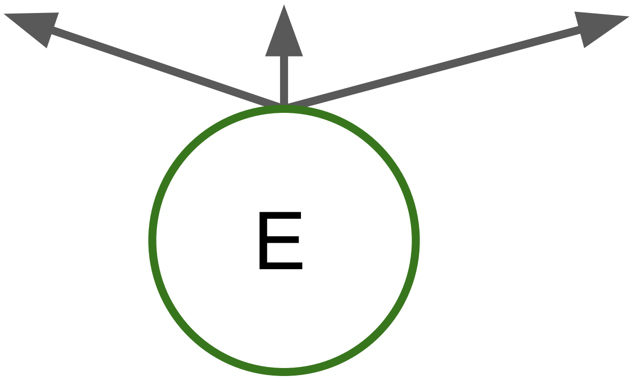
Parameter Server



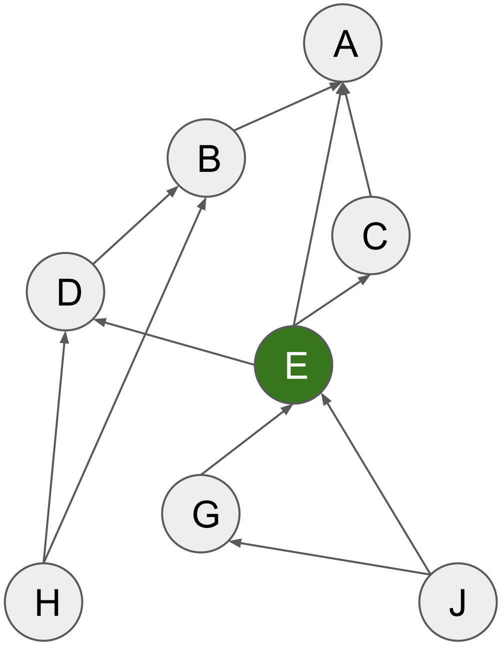
Part 2. Computation Operator Experts

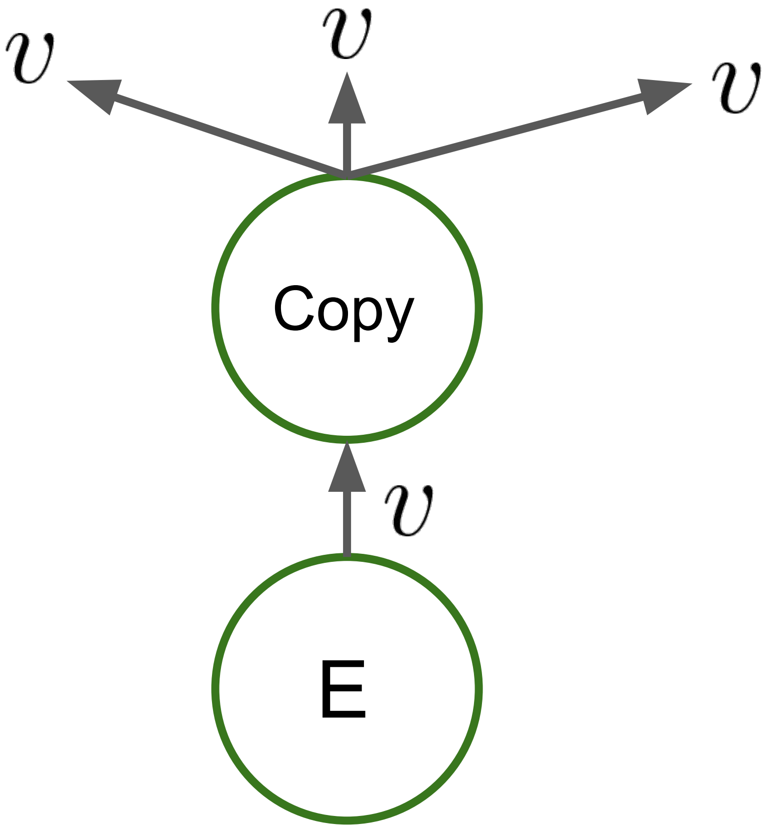
Copy module



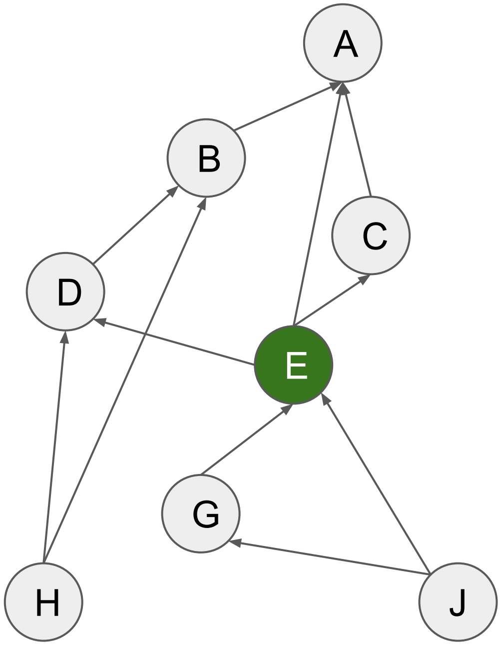


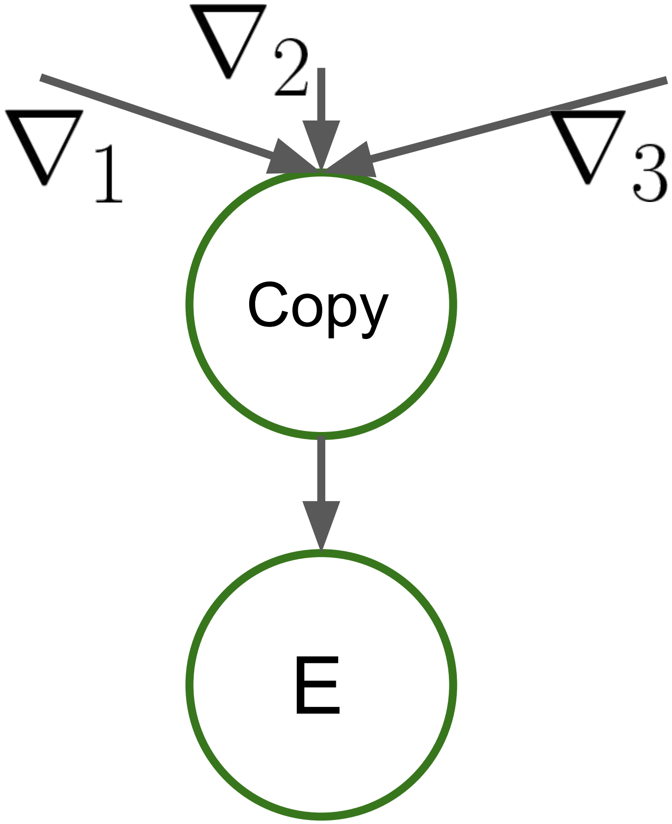
Copy module



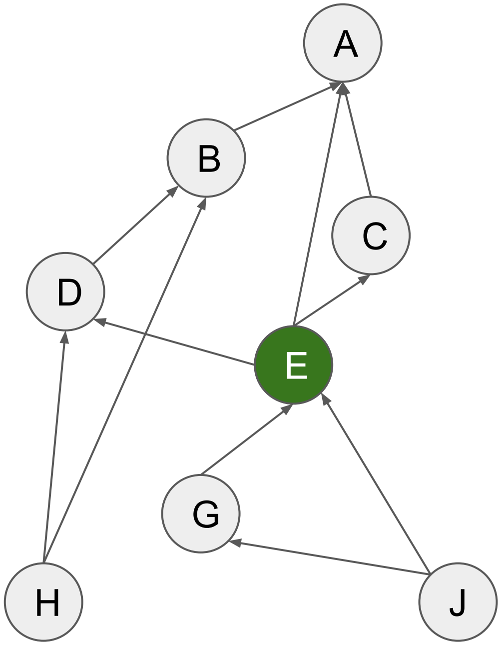


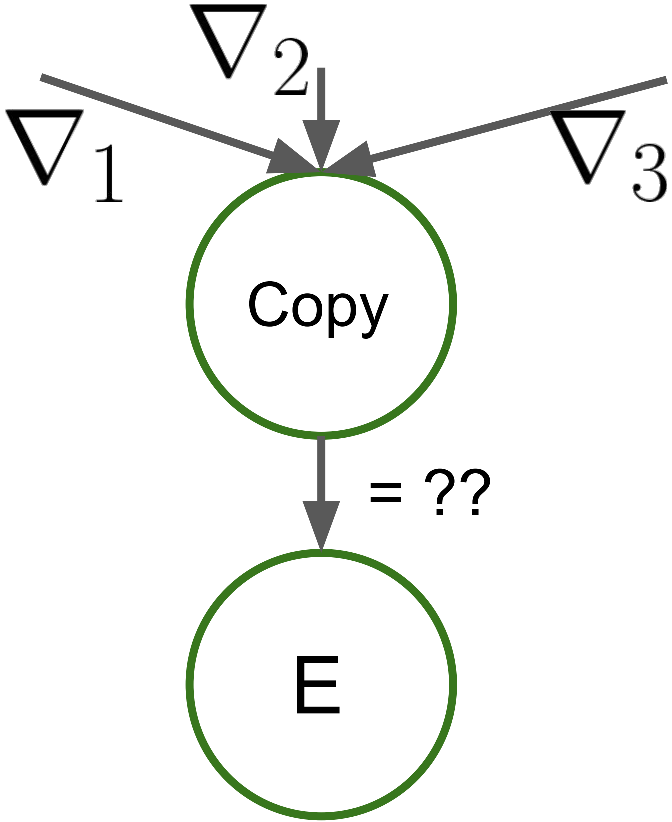
Copy module





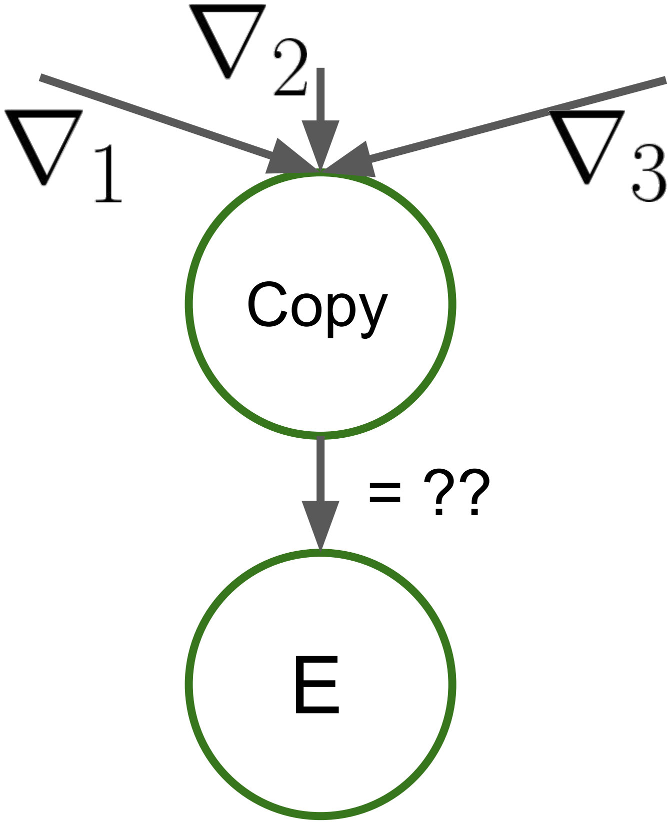
Copy module





Copy module

https://gist.github.com/thtrieu/b65ddf490d962fd14bbab70f526f167e



Backprop basics:
Plus and Element-wise Multiply

Backprop basics:
Bias-Adding (Broadcasting)

https://gist.github.com/thtrieu/c2207b1d32f91843928b40ffc3bd4a9c

Dropout

https://gist.github.com/thtrieu/18385644fe104d7dd74003804ba120a9

Dropout

https://gist.github.com/thtrieu/18385644fe104d7dd74003804ba120a9

Dropout : y = matmul(dropout(x), w)

Dropconnect: y = matmul(x, dropout(w))

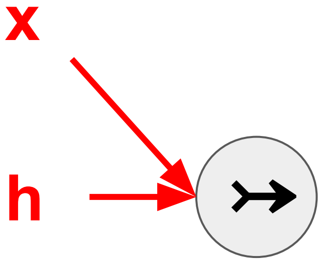
Vectorizing chain-rule case study: Fully Connected Module

https://gist.github.com/thtrieu/f4b573db29fae162c9d492682a9e1a71

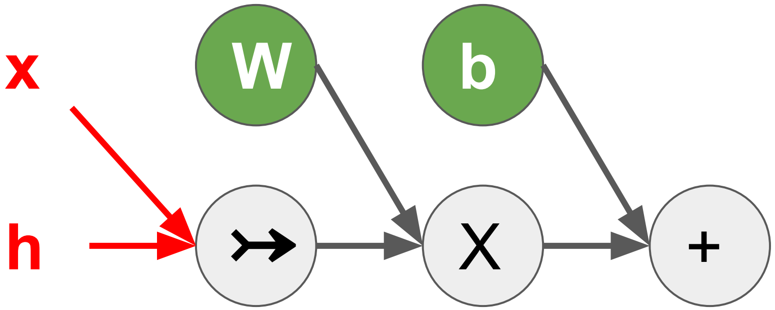
LSTM: Review

?

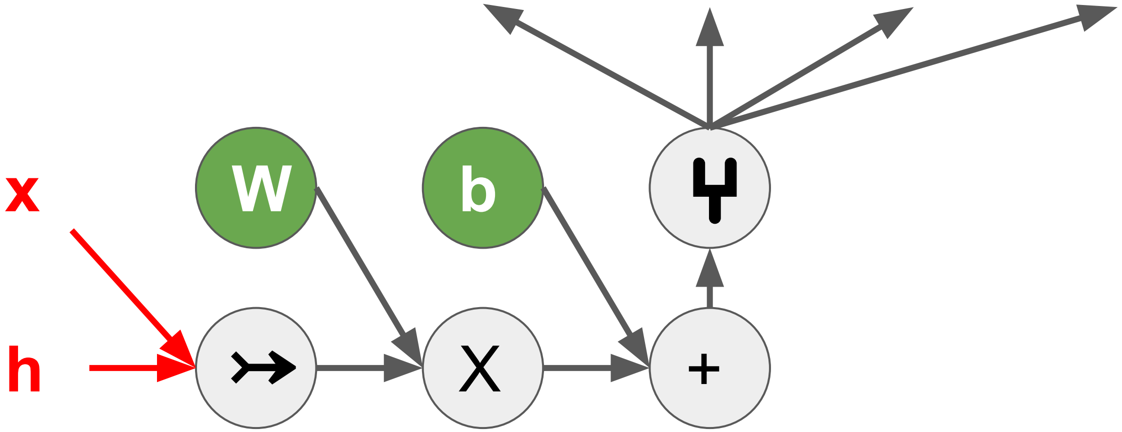
LSTM: Review



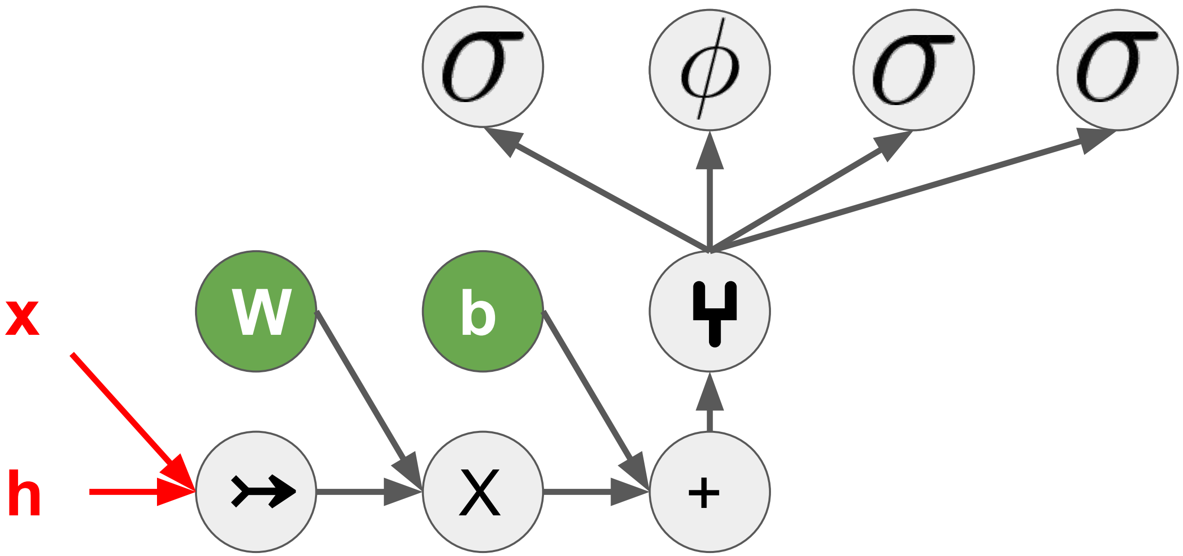
LSTM: Review



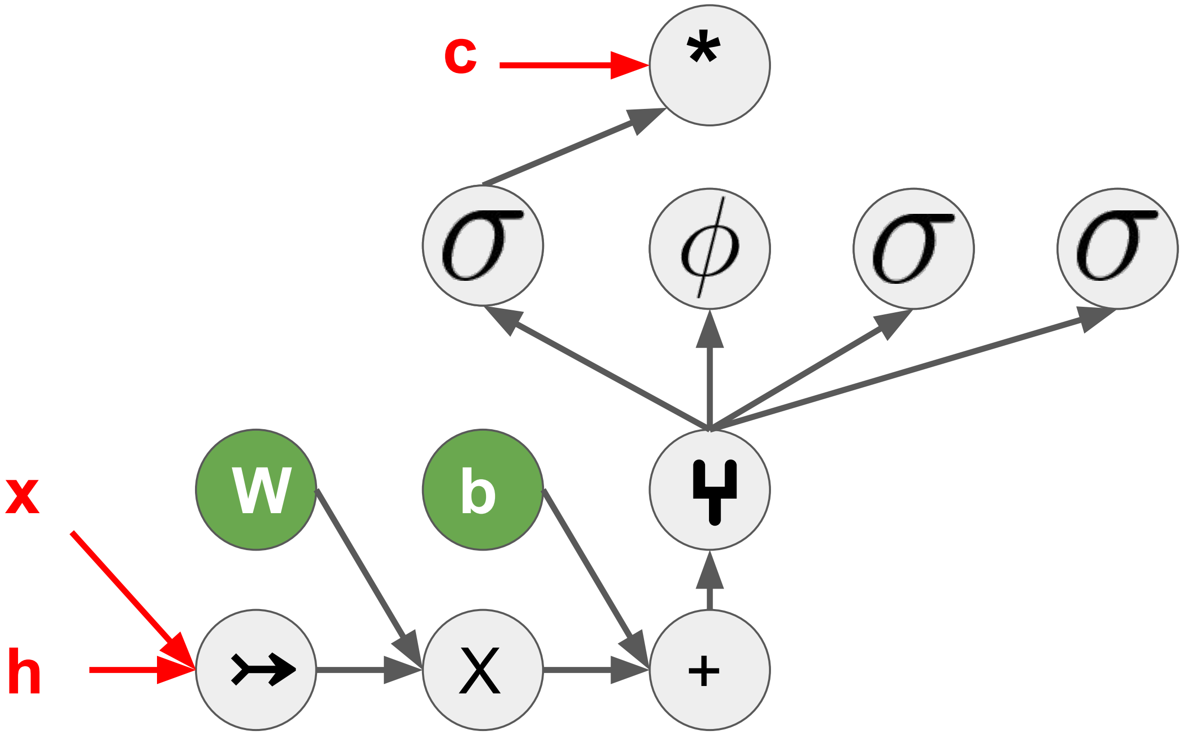
LSTM: Review



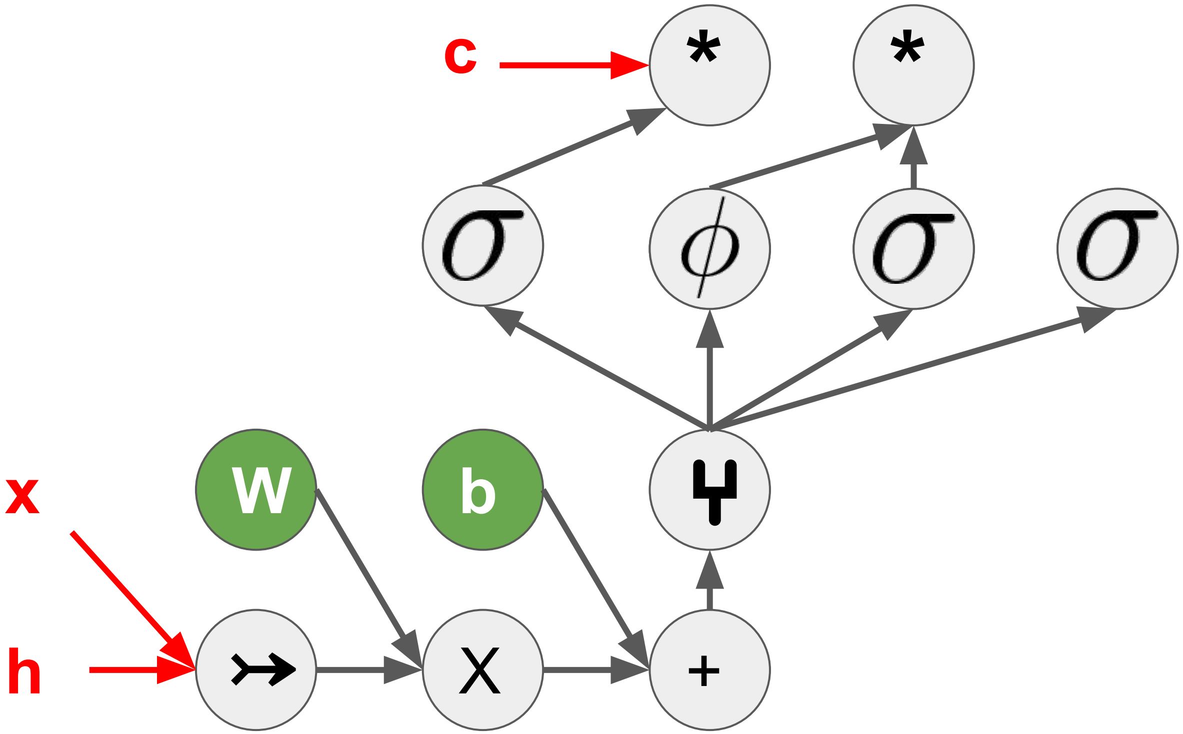
LSTM: Review



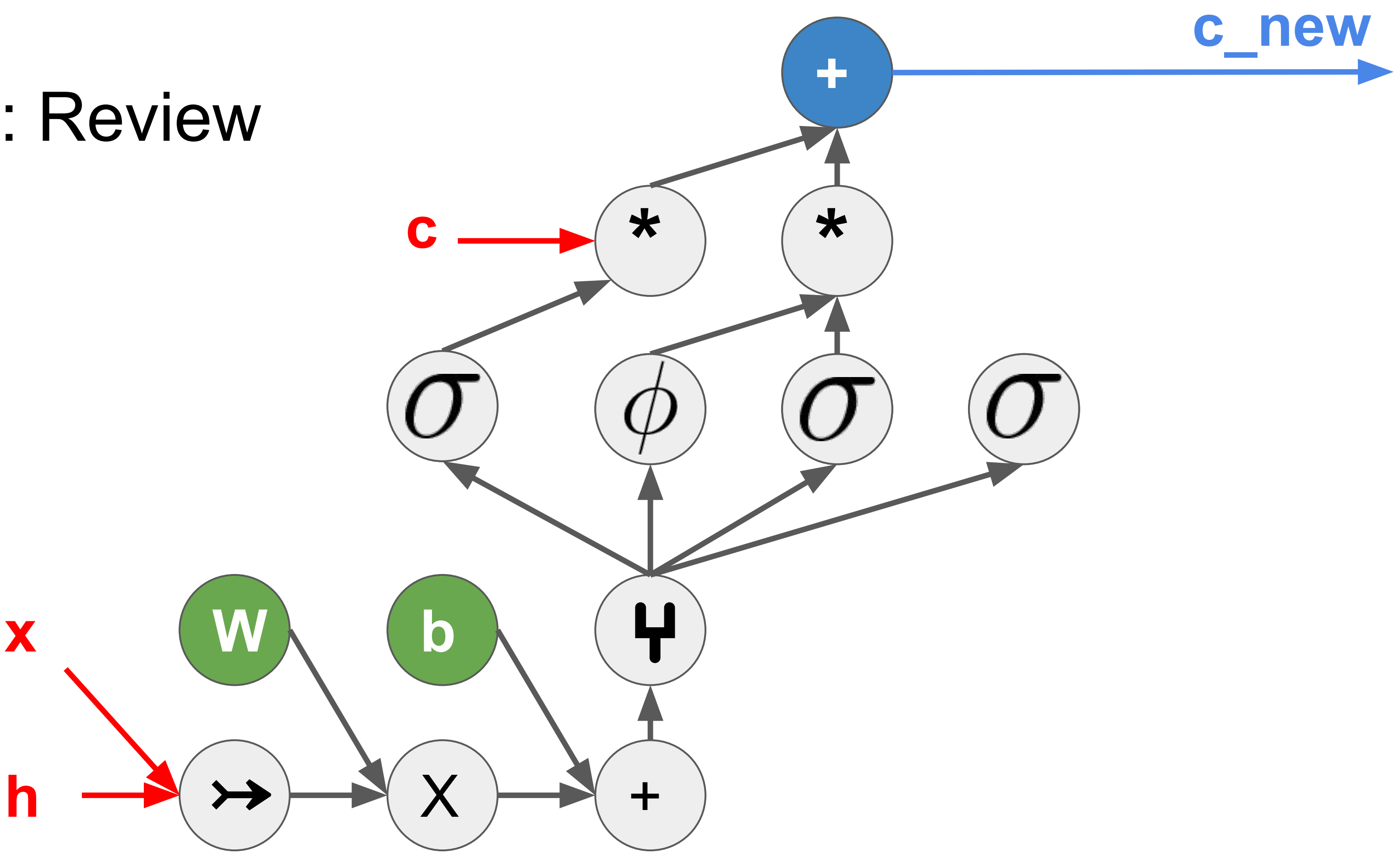
LSTM: Review



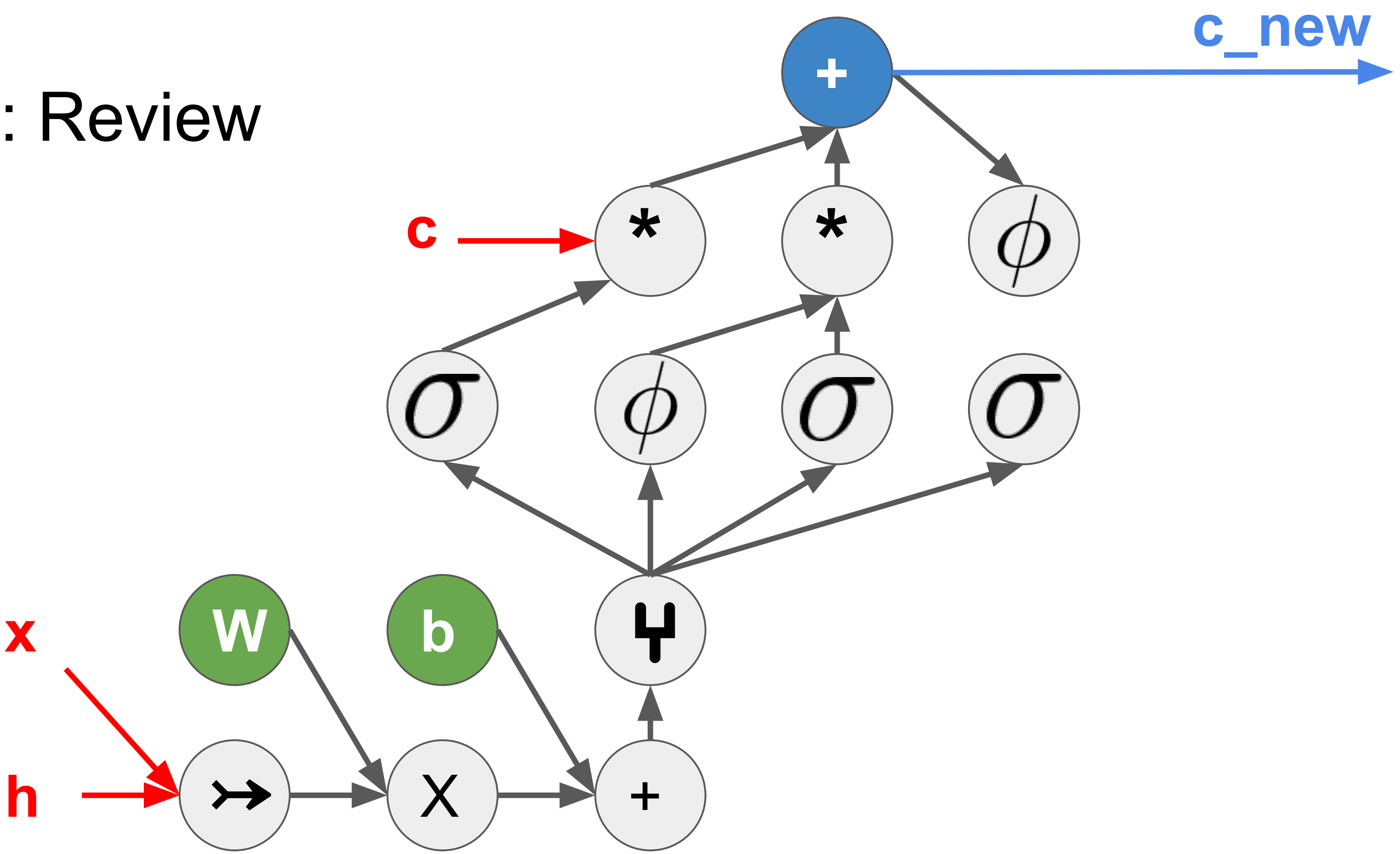
LSTM: Review



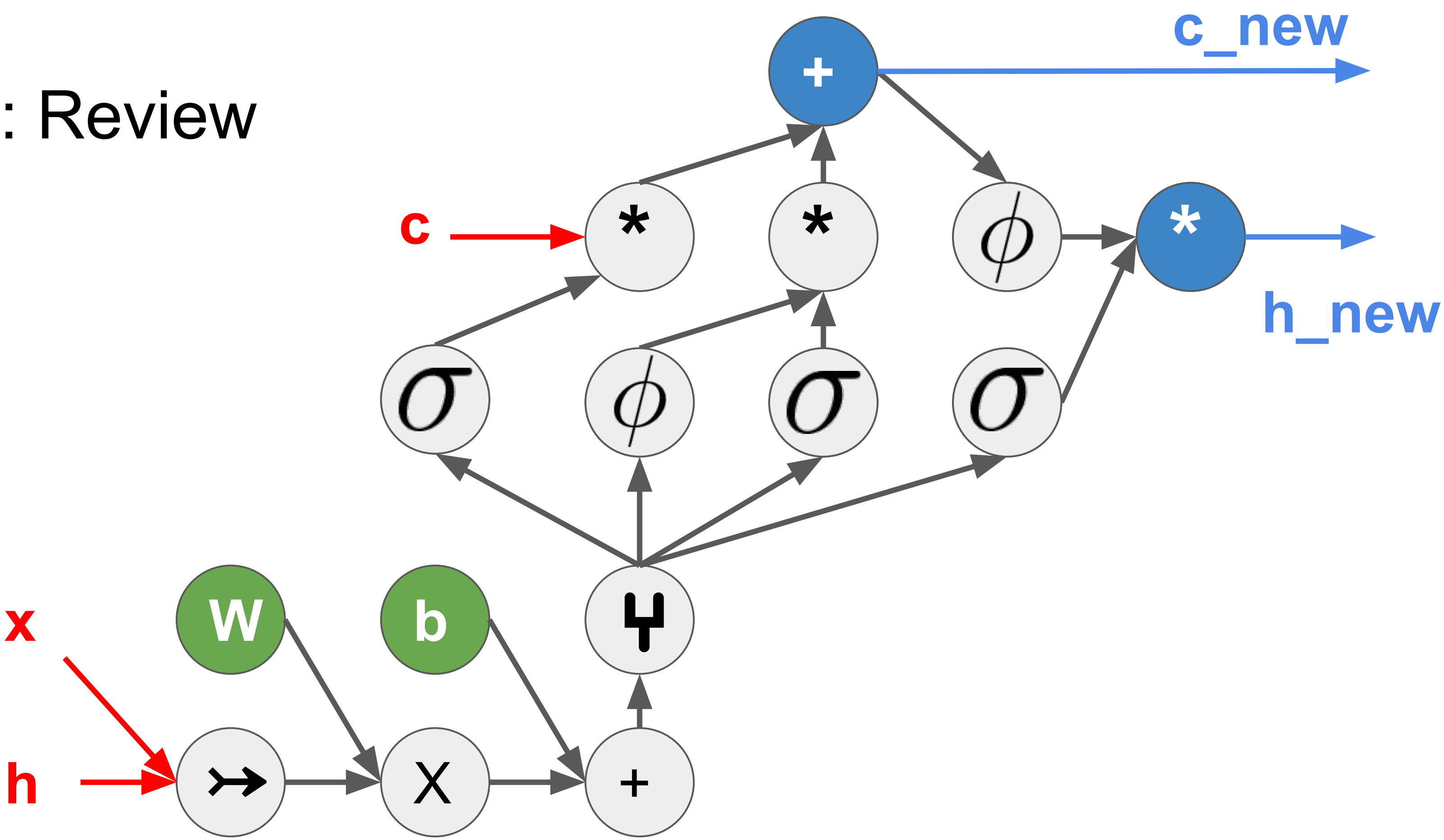
LSTM: Review



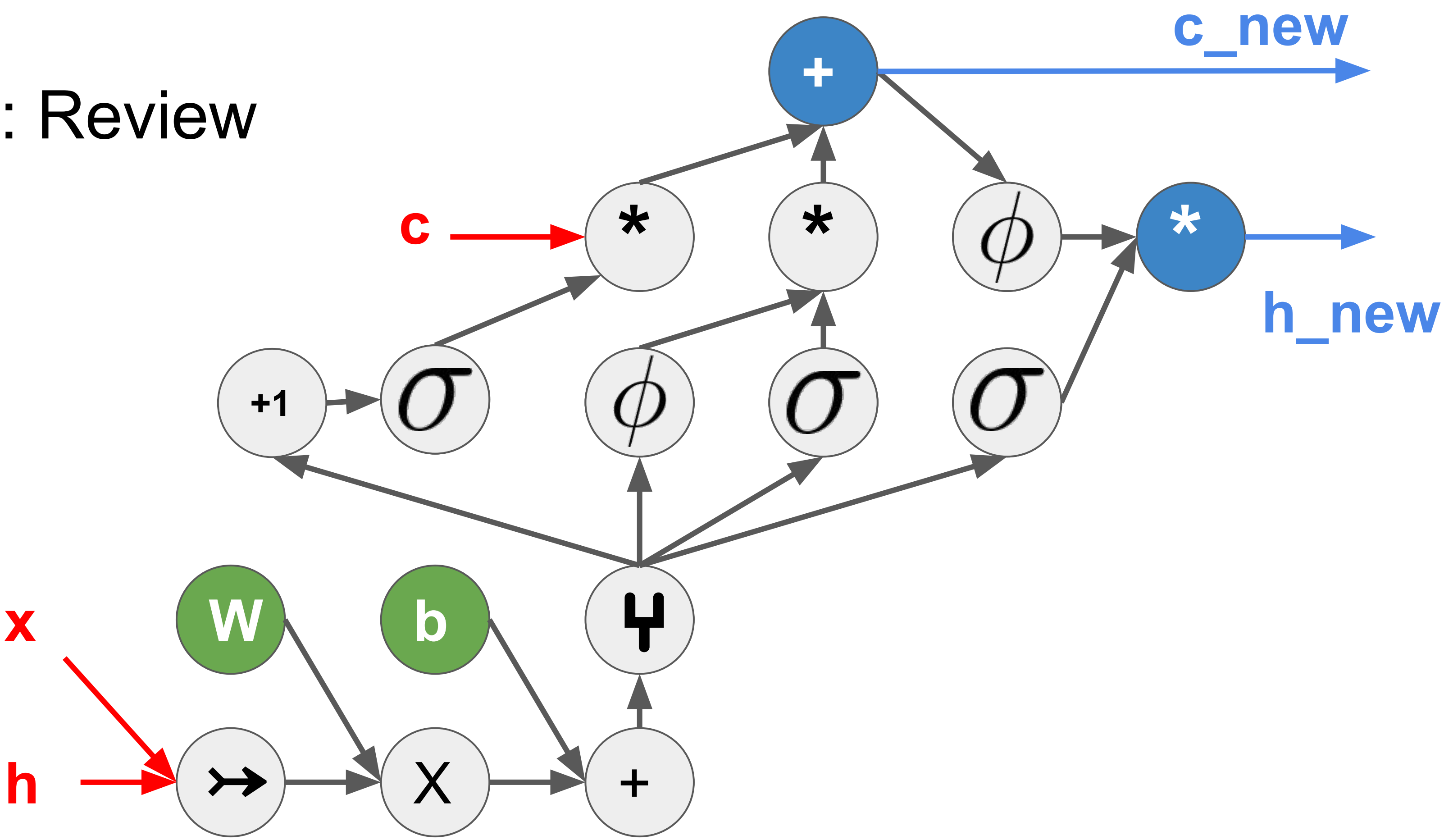
LSTM: Review



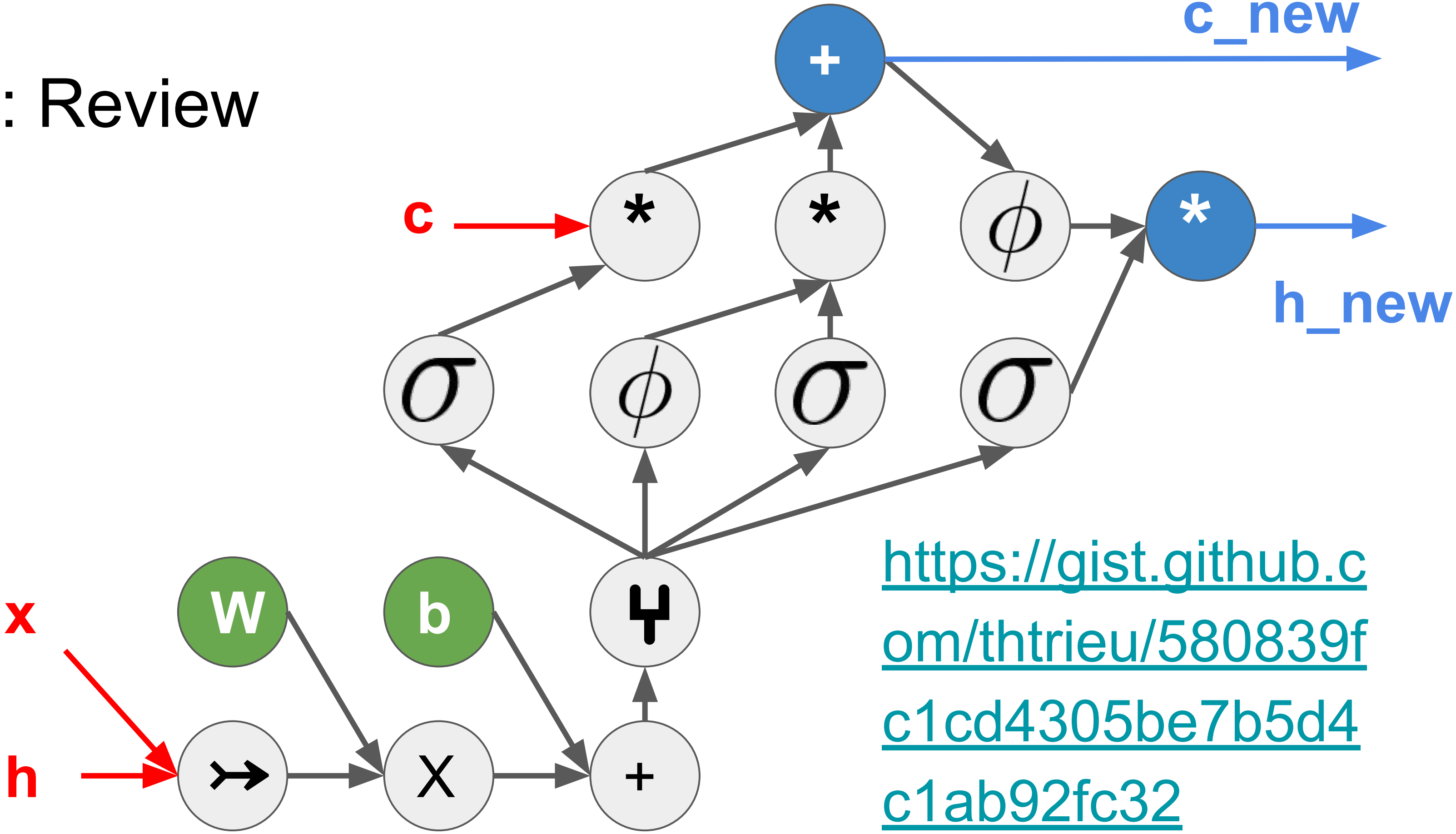
LSTM: Review



LSTM: Review



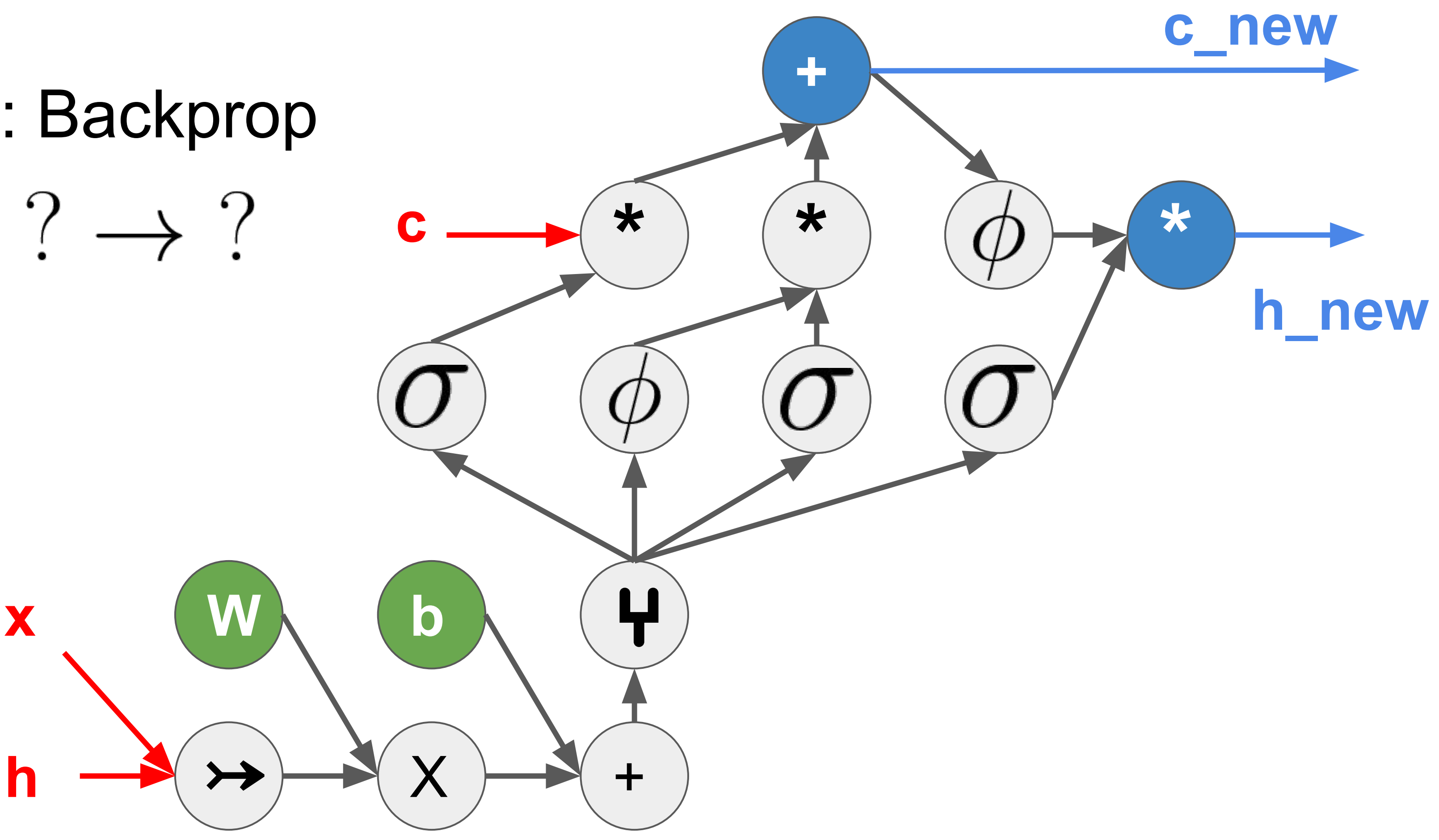
LSTM: Review



https://gist.github.com/thtrieu/580839fc1cd4305be7b5d4c1ab92fc32

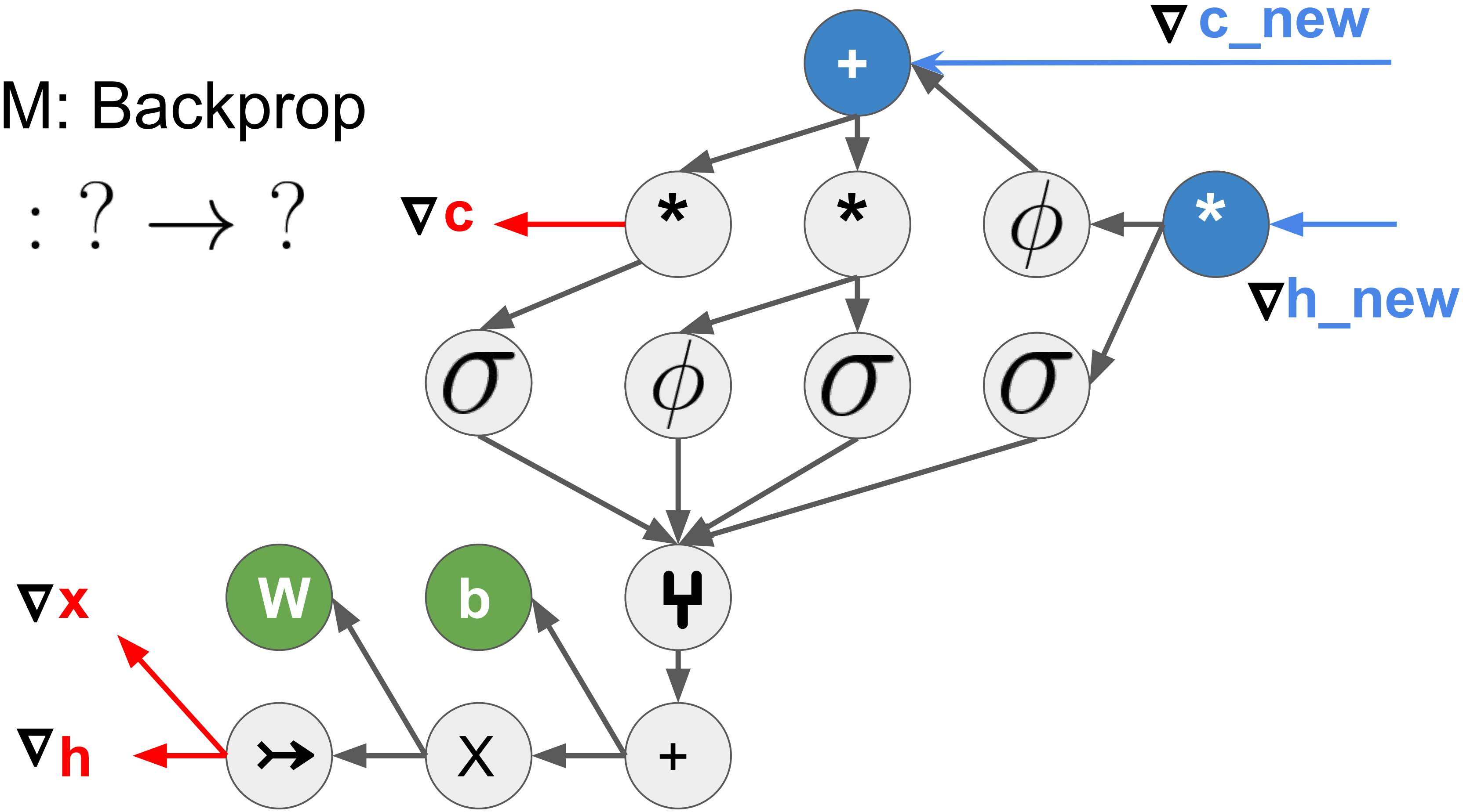
LSTM: Backprop

$$f : ? \to ?$$



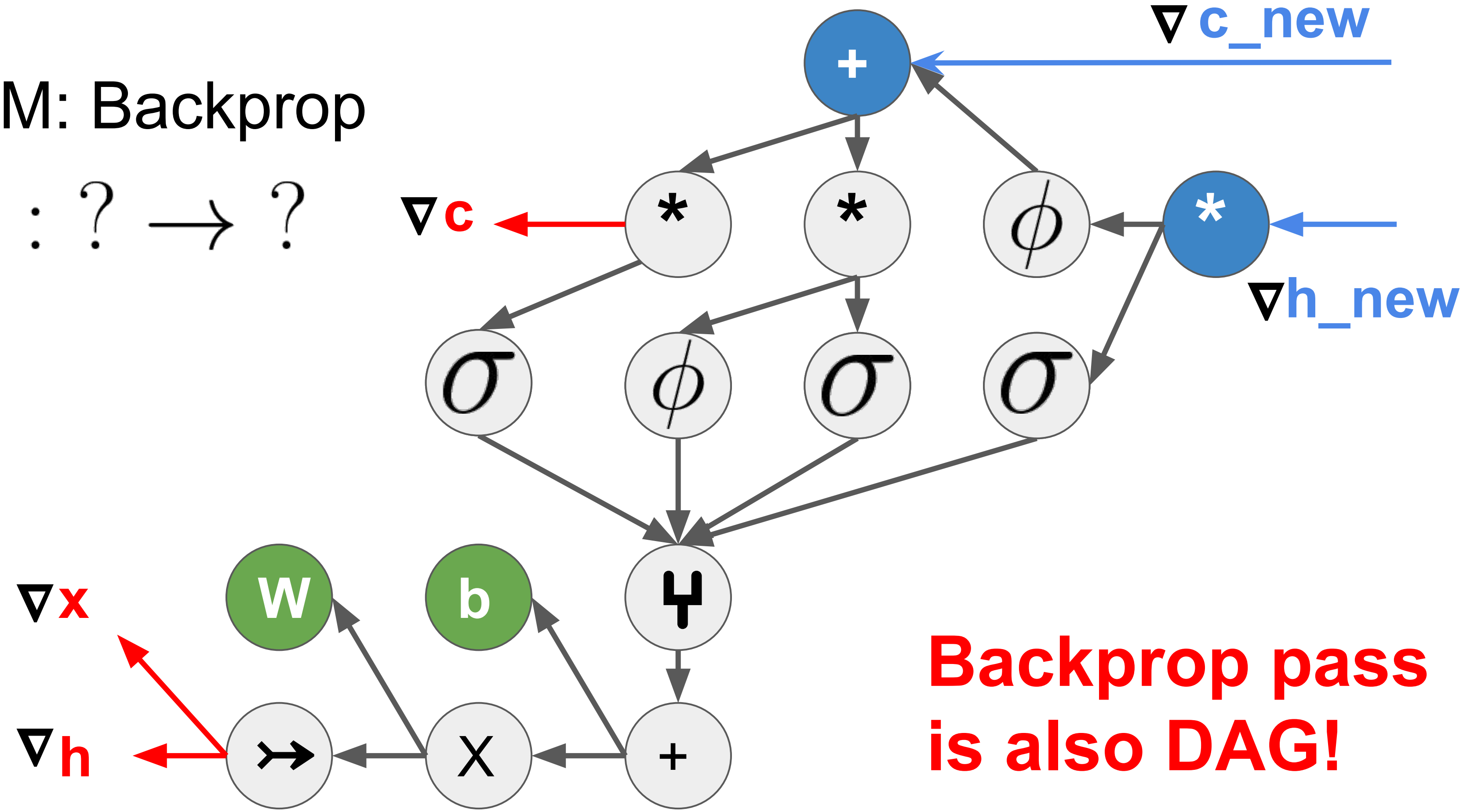
LSTM: Backprop

$$f : ? \rightarrow ?$$



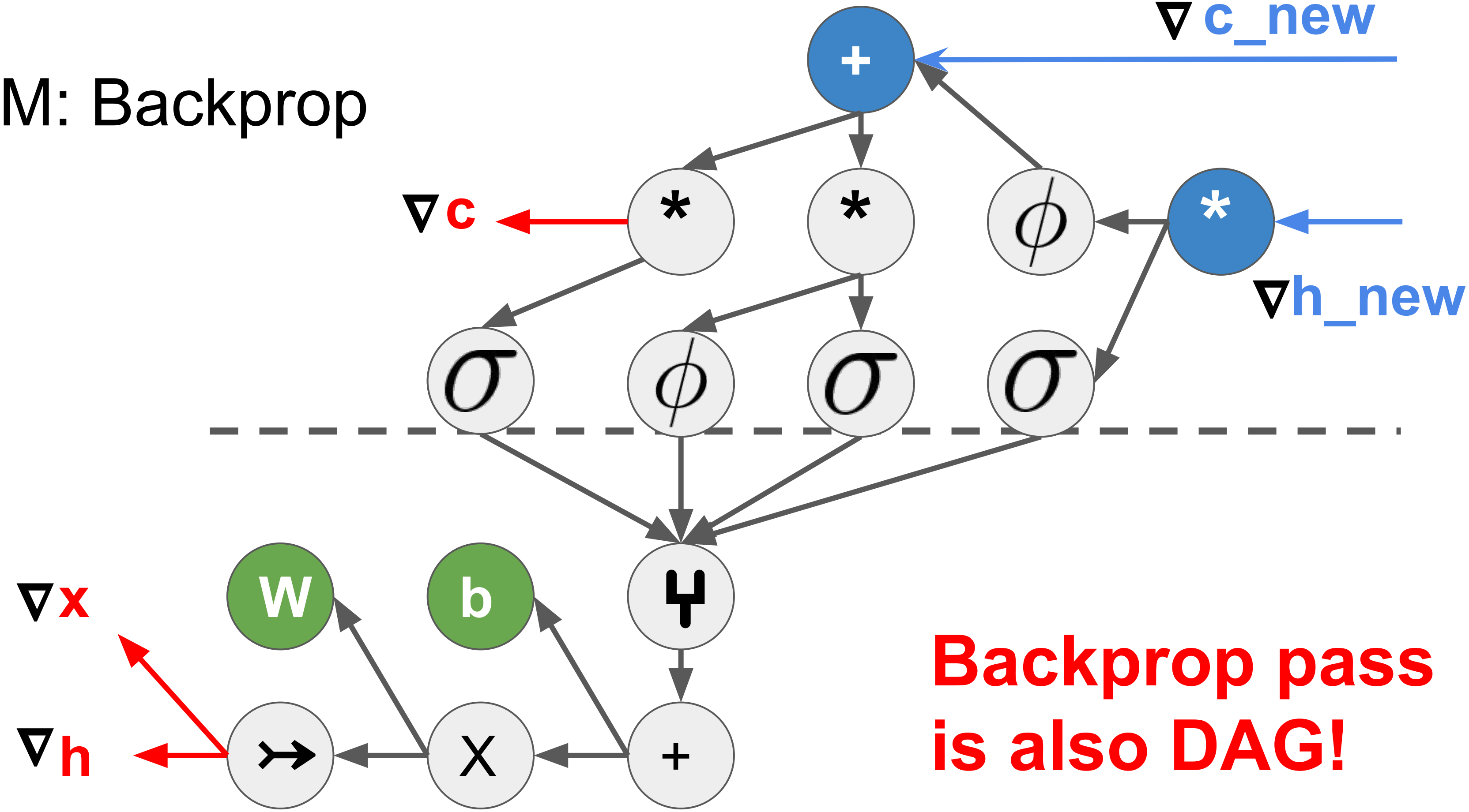
LSTM: Backprop

$$f : ? \to ?$$



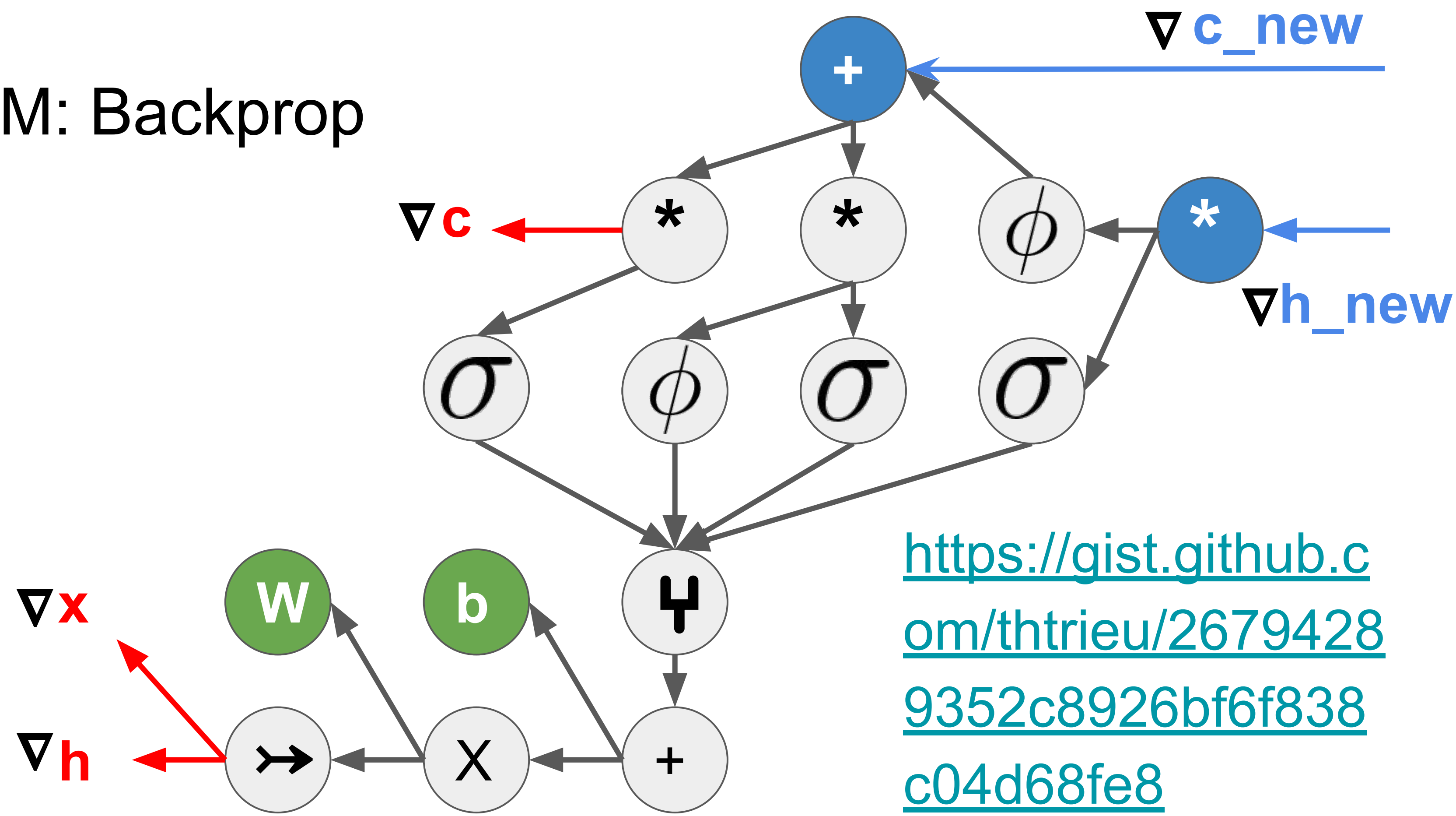
Backprop pass
is also DAG!

LSTM: Backprop



Backprop pass is also DAG!

LSTM: Backprop



https://gist.github.com/thtrieu/26794289352c8926bf6f838c04d68fe8

Teaser 004: Winograd Schema Challenge (Commonsense Reasoning)

- The **trophy** cannot fit in the **suitcase** because ***it*** is too big.

Teaser 004: Winograd Schema Challenge (Commonsense Reasoning)

??

- The **trophy** cannot fit in the **suitcase** because ***it*** is too big.

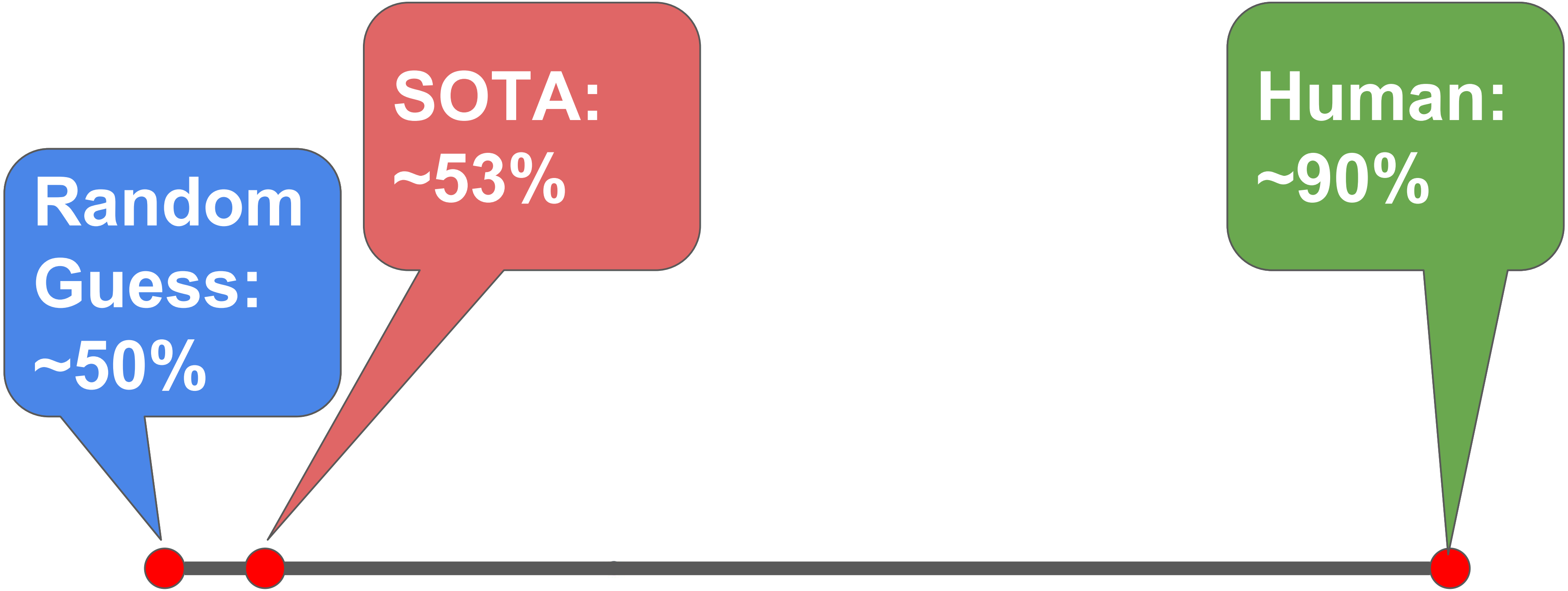
Teaser 004: Winograd Schema Challenge (Commonsense Reasoning)

??

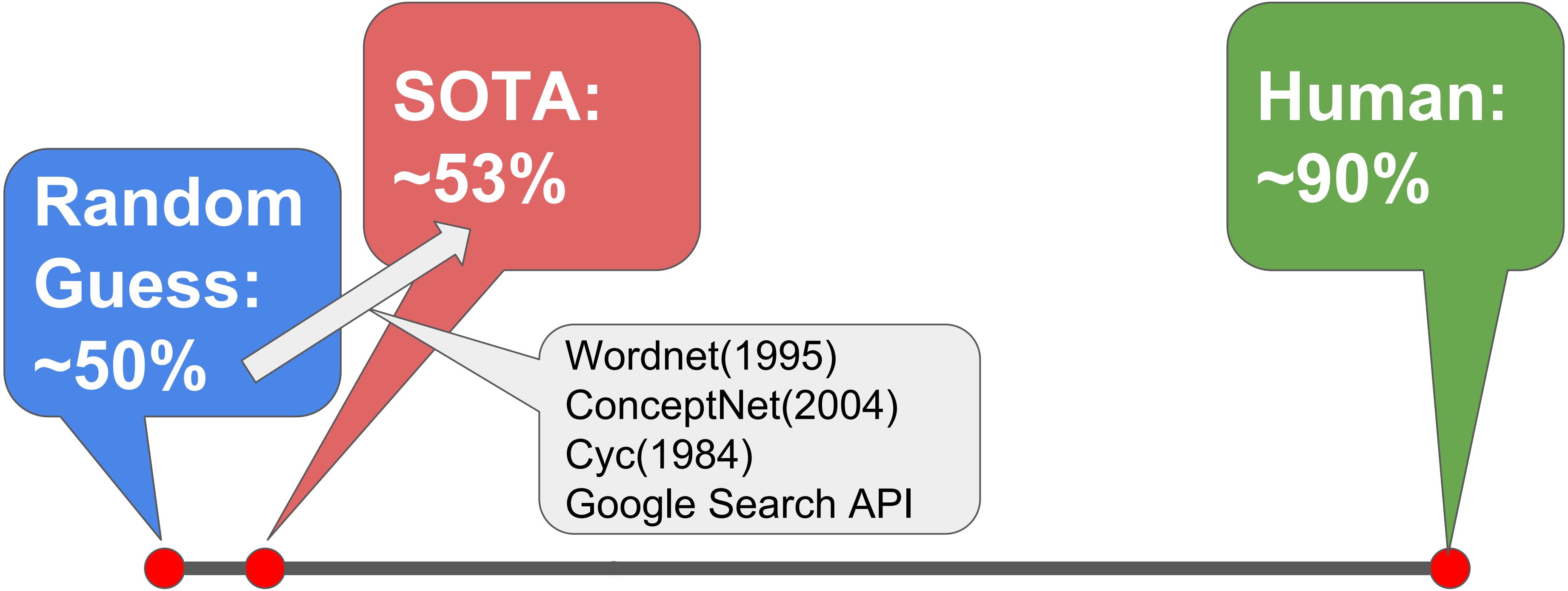
- The **trophy** cannot fit in the **suitcase** because ***it*** is too big.

Teaser 004: Winograd Schema Challenge (Commonsense Reasoning)

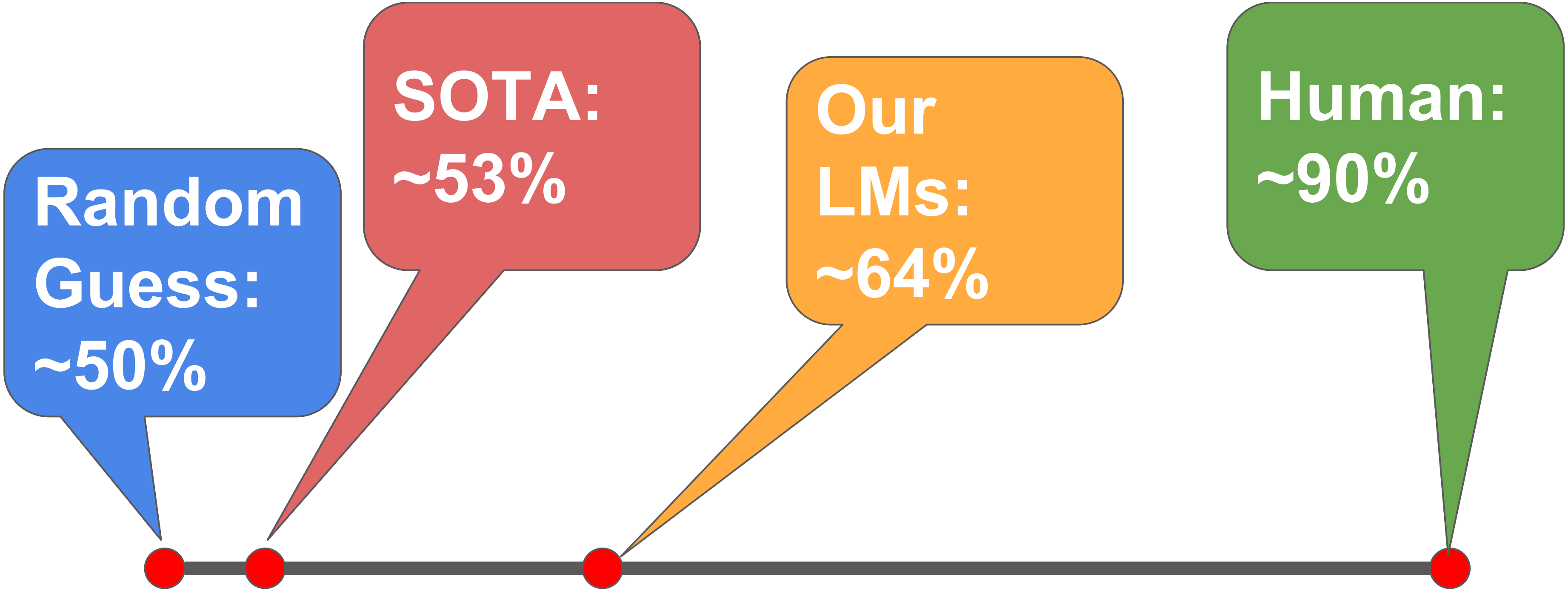
Teaser 004: Winograd Schema Challenge (Commonsense Reasoning)



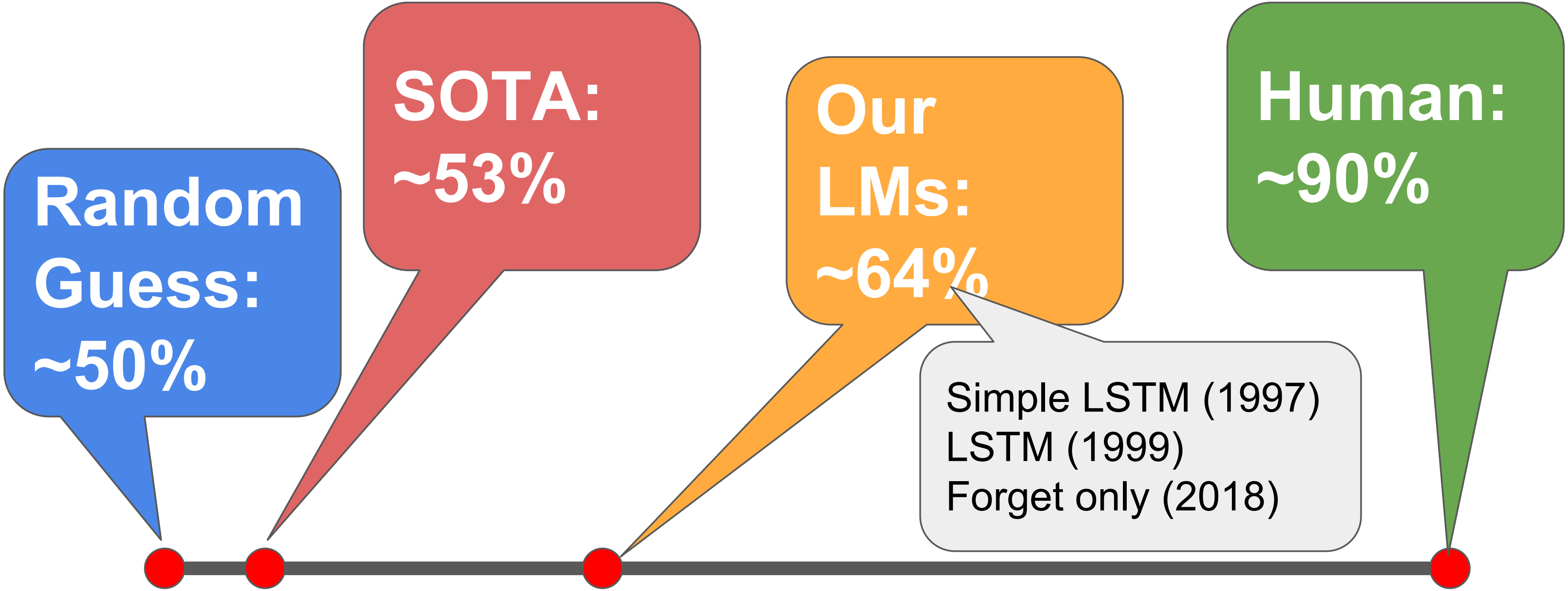
Teaser 004: Winograd Schema Challenge (Commonsense Reasoning)



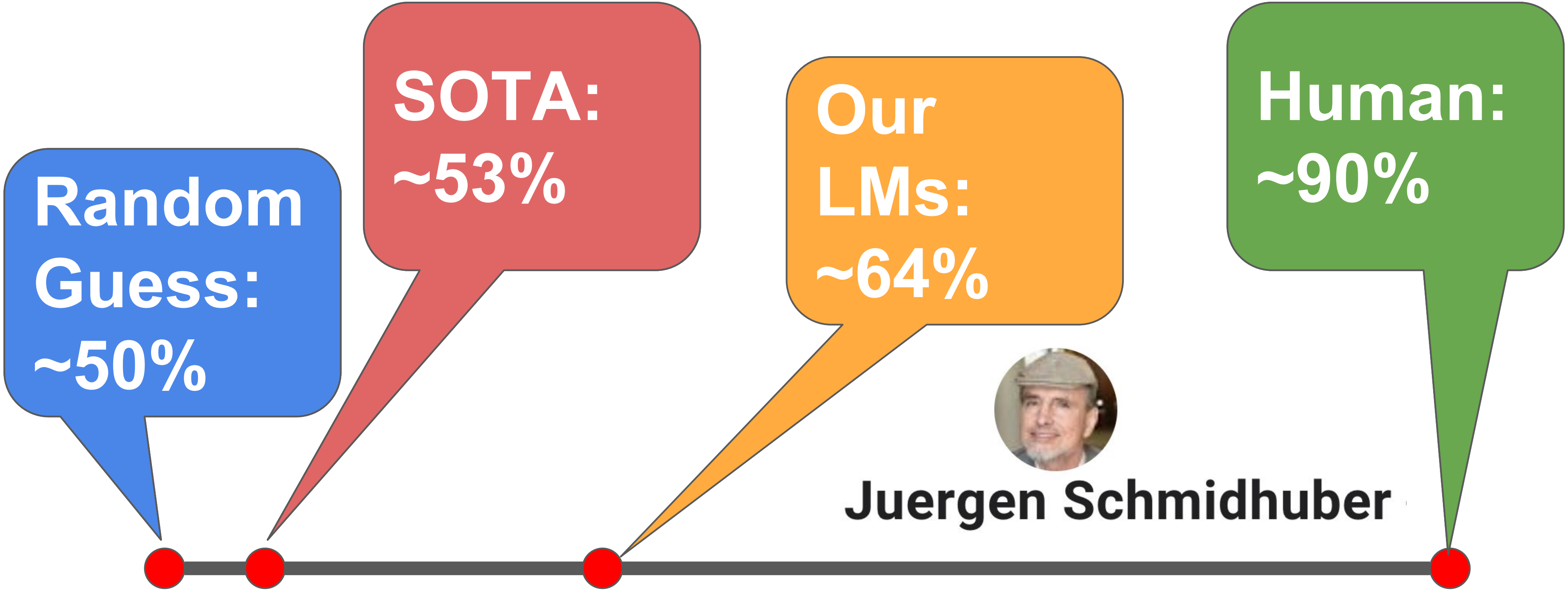
Teaser 004: Winograd Schema Challenge
(Commonsense Reasoning)



Teaser 004: Winograd Schema Challenge (Commonsense Reasoning)



Teaser 004: Winograd Schema Challenge (Commonsense Reasoning)



Stories time: **LM is magic**

Task	Previous SOTA		Our baseline	ELMo + Baseline	Increase (Absolute/Relative)
SQuAD	SAN	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al (2017)	88.6	88.0	88.7 +/- 0.17	0.7 / 5.8%
SRL	He et al (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al (2017)	91.93 +/- 0.19	90.15	92.22 +/- 0.10	2.06 / 21%
Sentiment (5-class)	McCann et al (2017)	53.7	51.4	54.7 +/- 0.5	3.3 / 6.8%

Stories time: **LM is magic**

	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

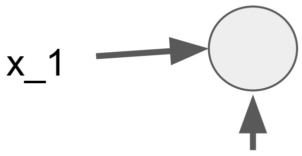
Table 2: Test error rates (%) on two text classification datasets used by McCann et al. (2017).

	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

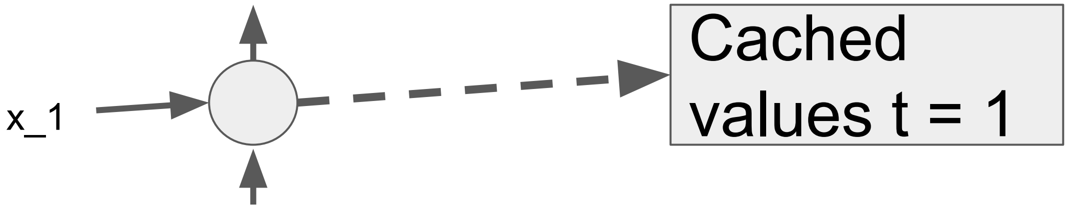
Stories time: **LM is magic**

DATASET	TASK	SOTA	OURS
SNLI	Textual Entailment	89.3	**89.9**
MNLI Matched	Textual Entailment	80.6	**82.1**
MNLI Mismatched	Textual Entailment	80.1	**81.4**
SciTail	Textual Entailment	83.3	**88.3**
QNLI	Textual Entailment	82.3	**88.1**
RTE	Textual Entailment	**61.7**	56.0
STS-B	Semantic Similarity	81.0	**82.0**
QQP	Semantic Similarity	66.1	**70.3**
MRPC	Semantic Similarity	**86.0**	82.3
RACE	Reading Comprehension	53.3	**59.0**
ROCStories	Commonsense Reasoning	77.6	**86.5**
COPA	Commonsense Reasoning	71.2	**78.6**
SST-2	Sentiment Analysis	**93.2**	91.3
CoLA	Linguistic Acceptability	35.0	**45.4**
GLUE	Multi Task Benchmark	68.9	**72.8**

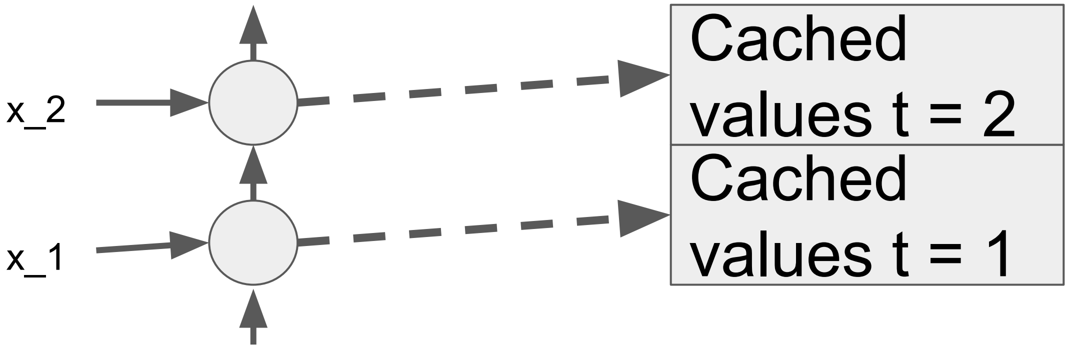
LSTM: Caching values



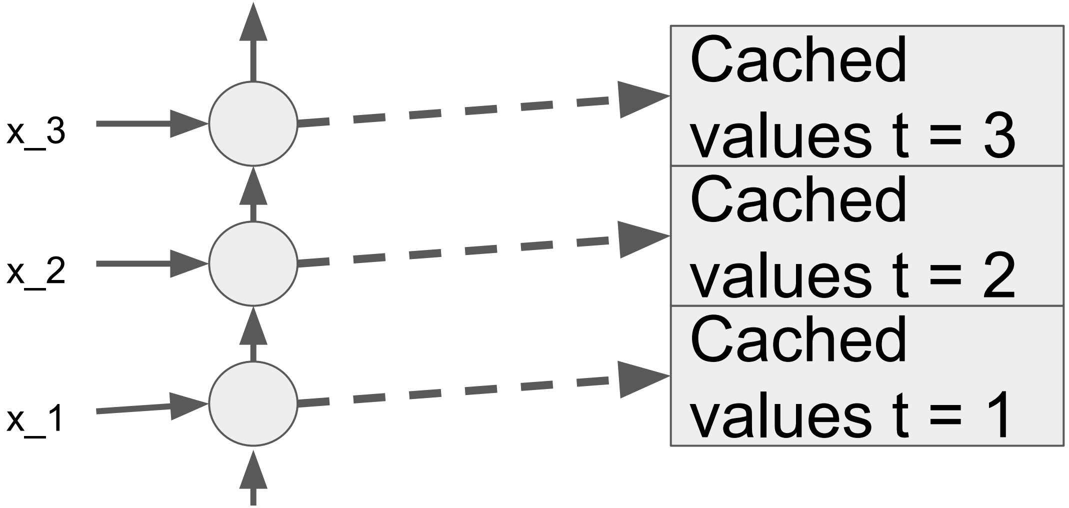
LSTM: Caching values



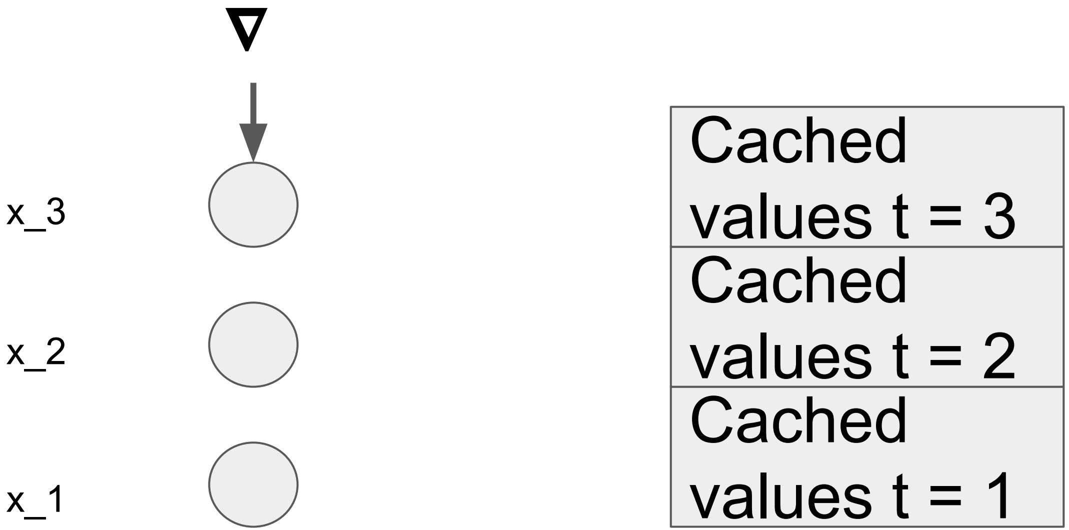
LSTM: Caching values



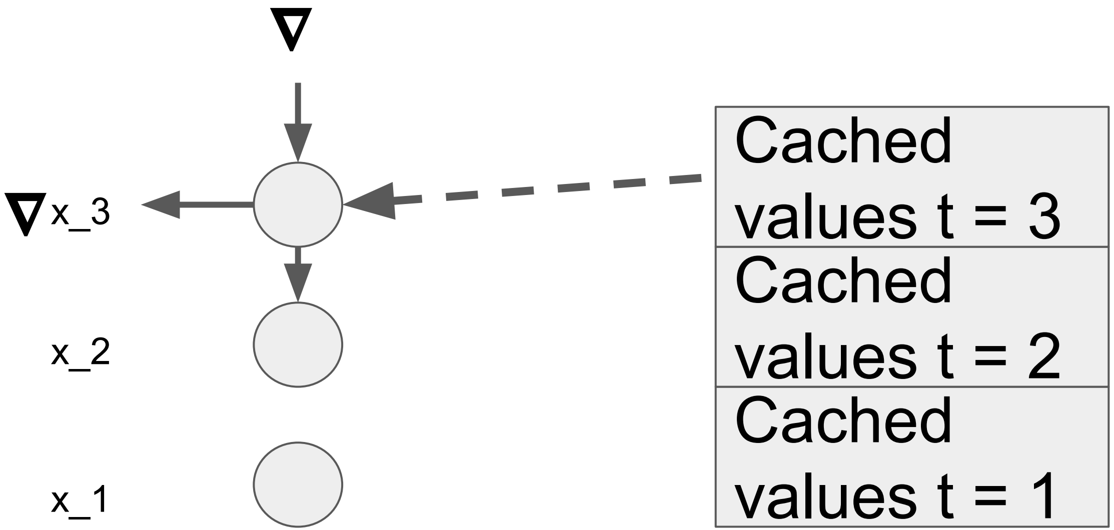
LSTM: Caching values



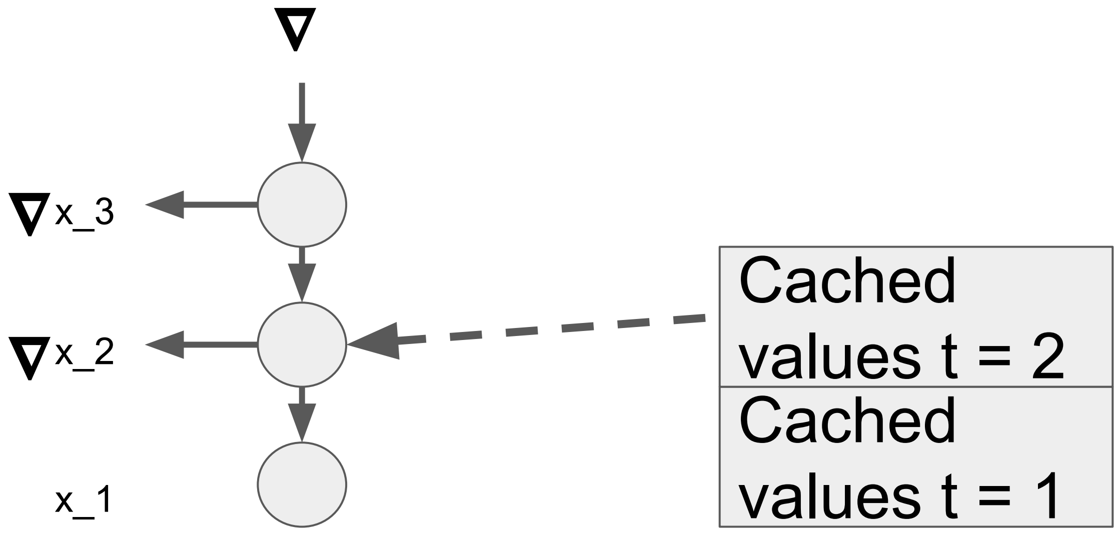
LSTM: Caching values



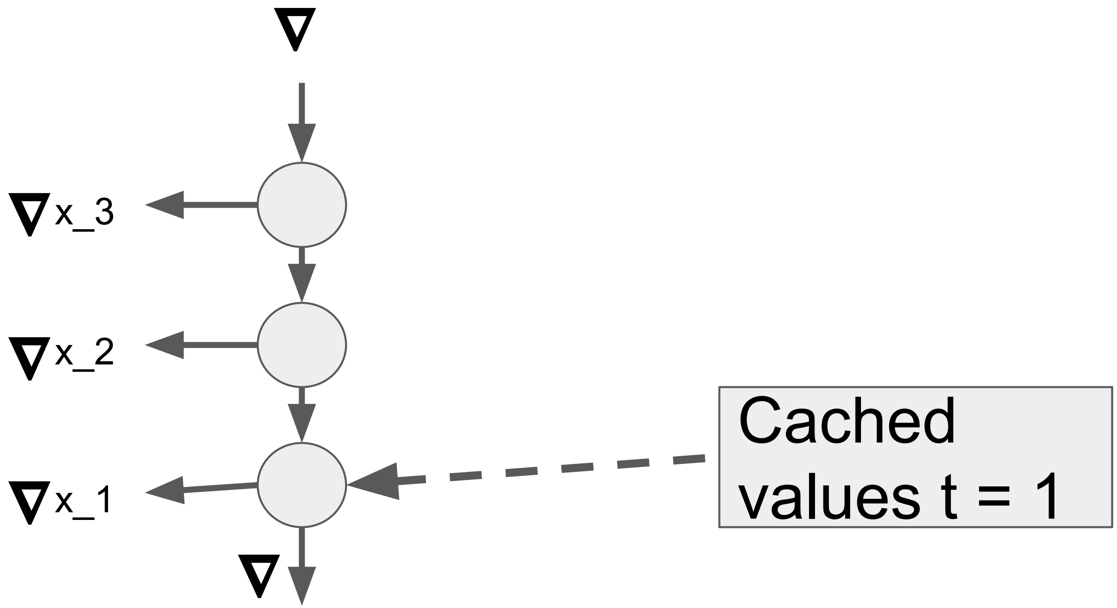
LSTM: Caching values



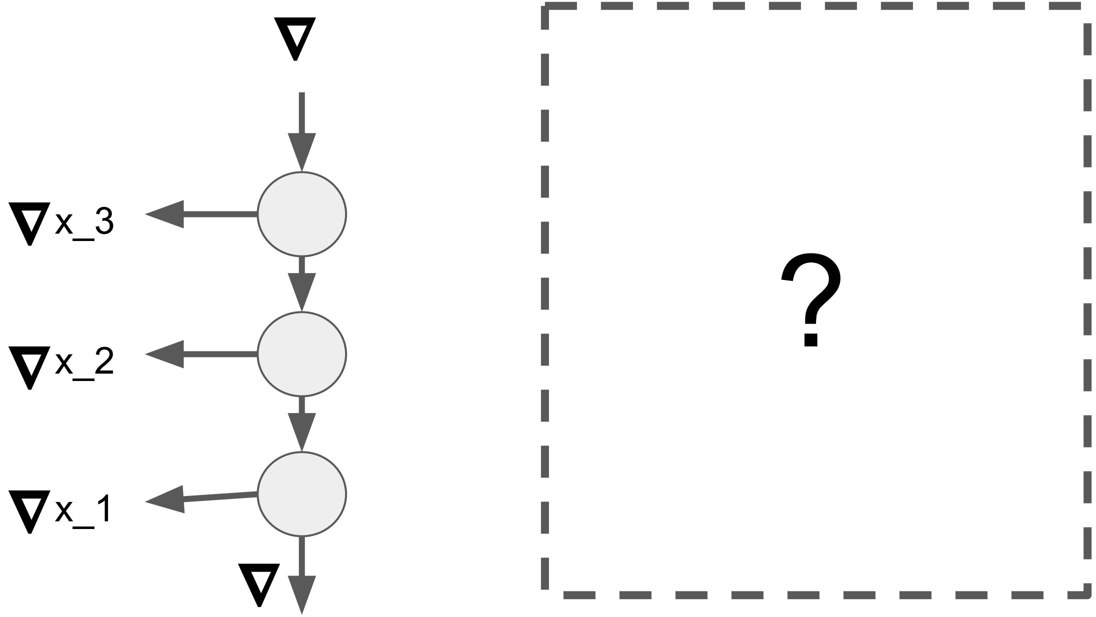
LSTM: Caching values



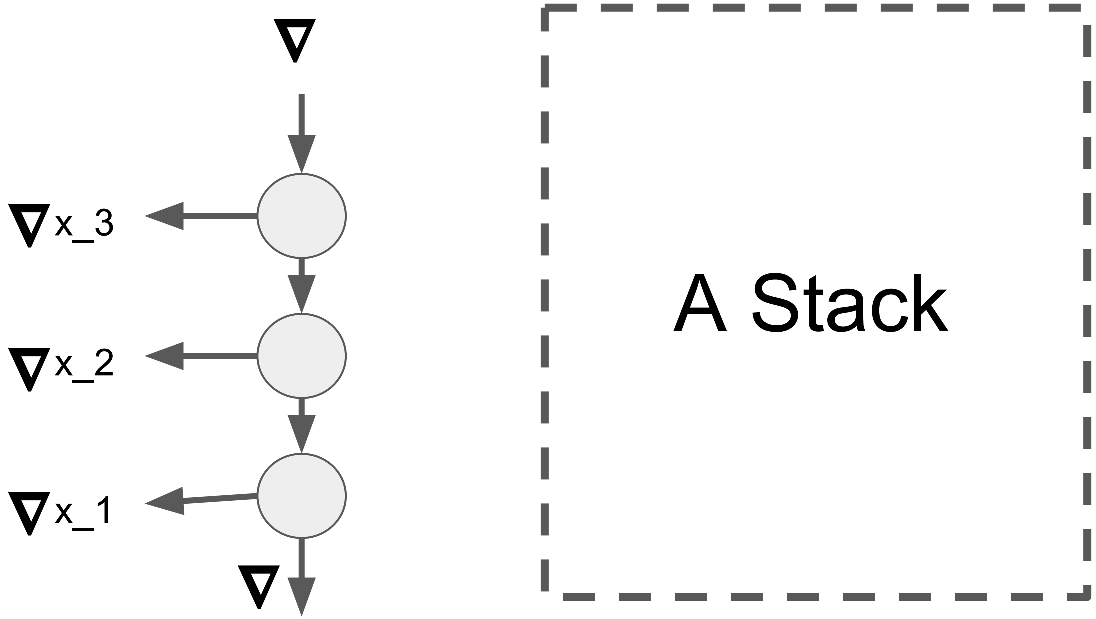
LSTM: Caching values



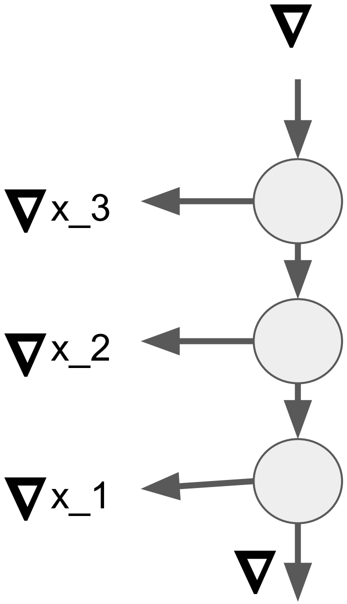
LSTM: Caching values



LSTM: Caching values

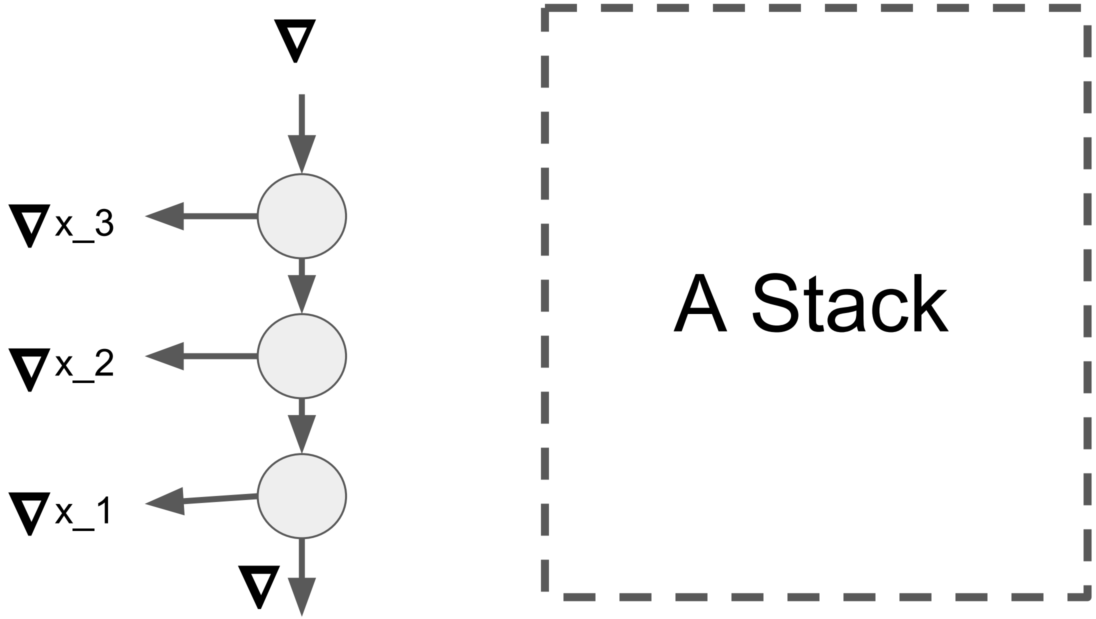


LSTM: Caching values

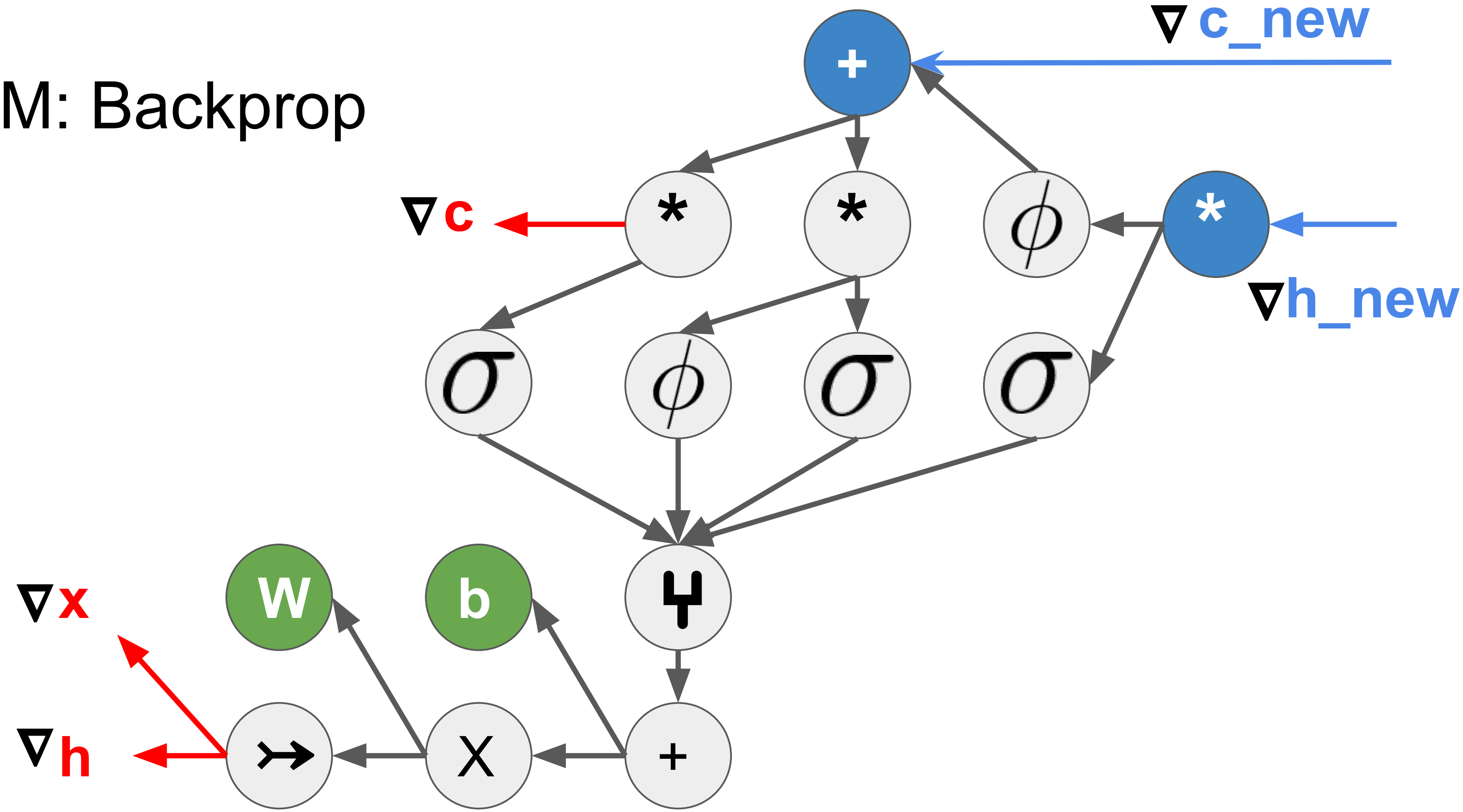


https://gist.github.com/thtrieu/62d7002476b6f40be2f61ee836b10759

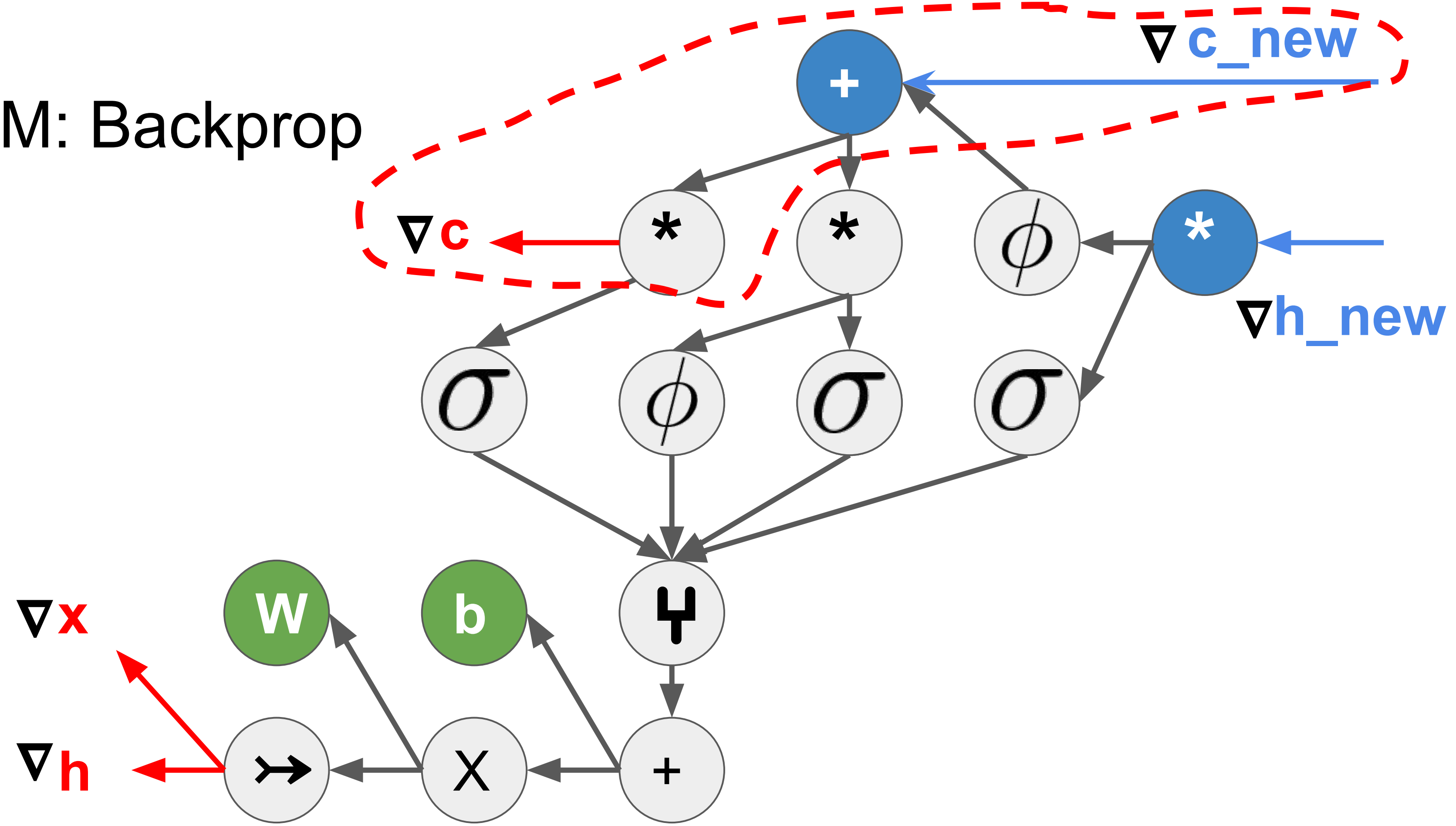
LSTM: Caching values: **Large Memory!**



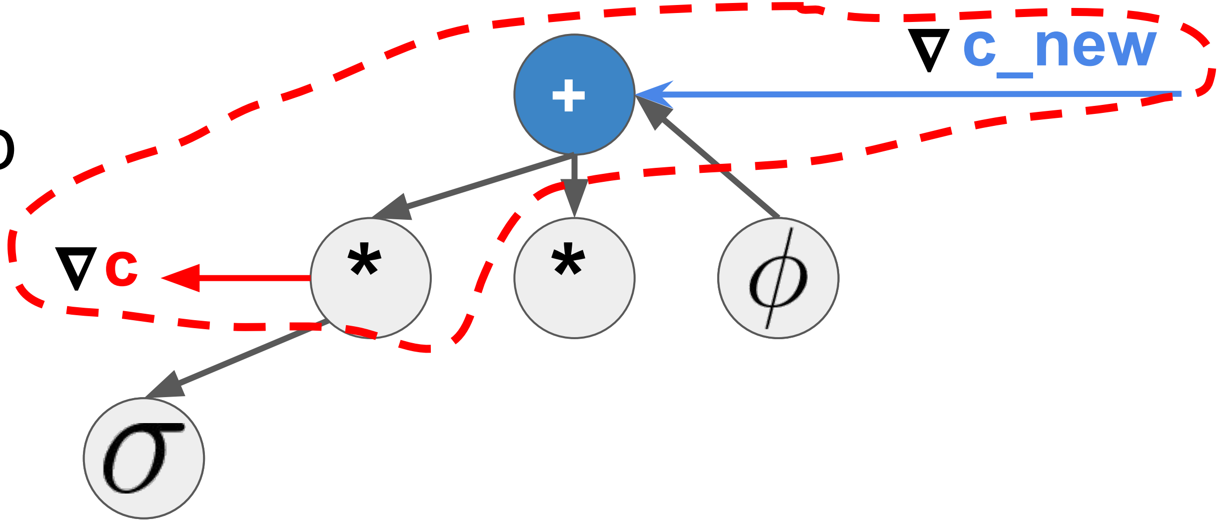
LSTM: Backprop



LSTM: Backprop

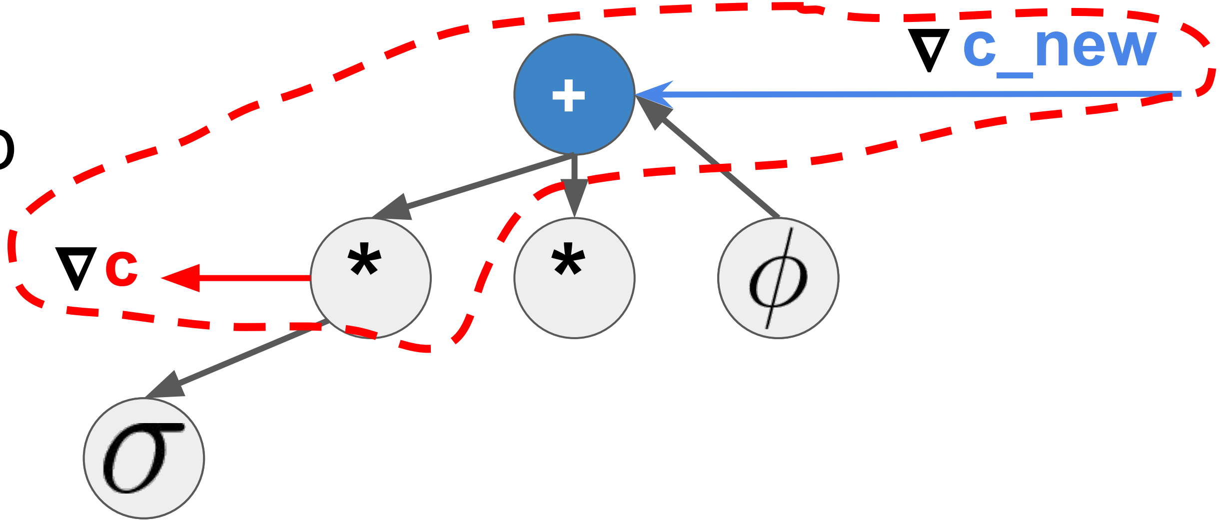


LSTM: Backprop



$$\nabla c_t = \sigma_t^f \odot \left(\nabla \phi_t (1 - \phi_t^2) + \nabla c_{t+1}\right)$$

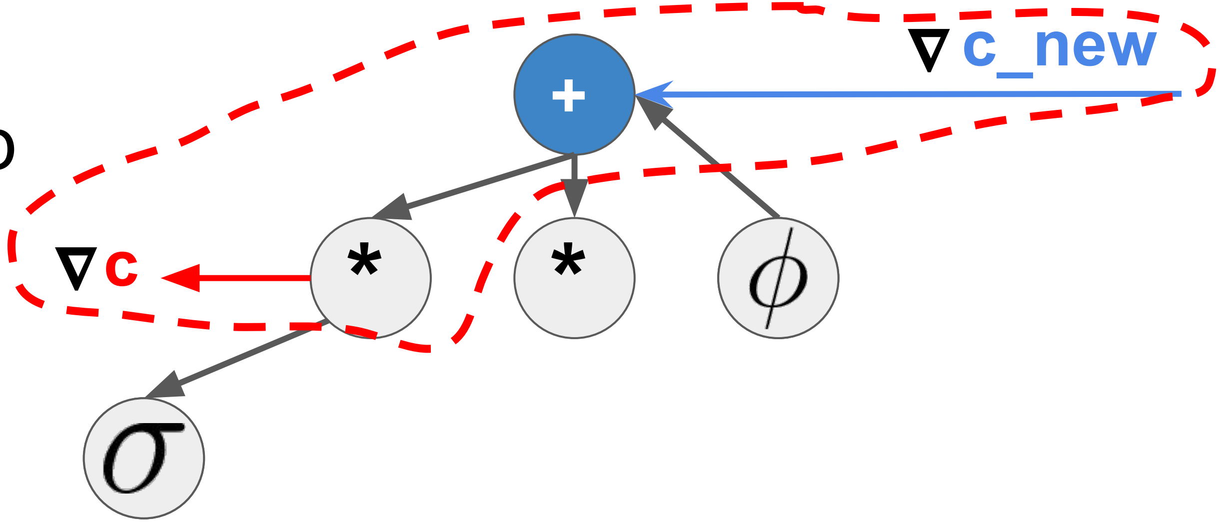
LSTM: Backprop



$$\nabla c_t = \sigma_t^f \odot (\nabla \phi_t (1 - \phi_t^2) + \nabla c_{t+1})$$

$$\nabla c_t = \sigma_t^f \odot \nabla c_{t+1} + v_t$$

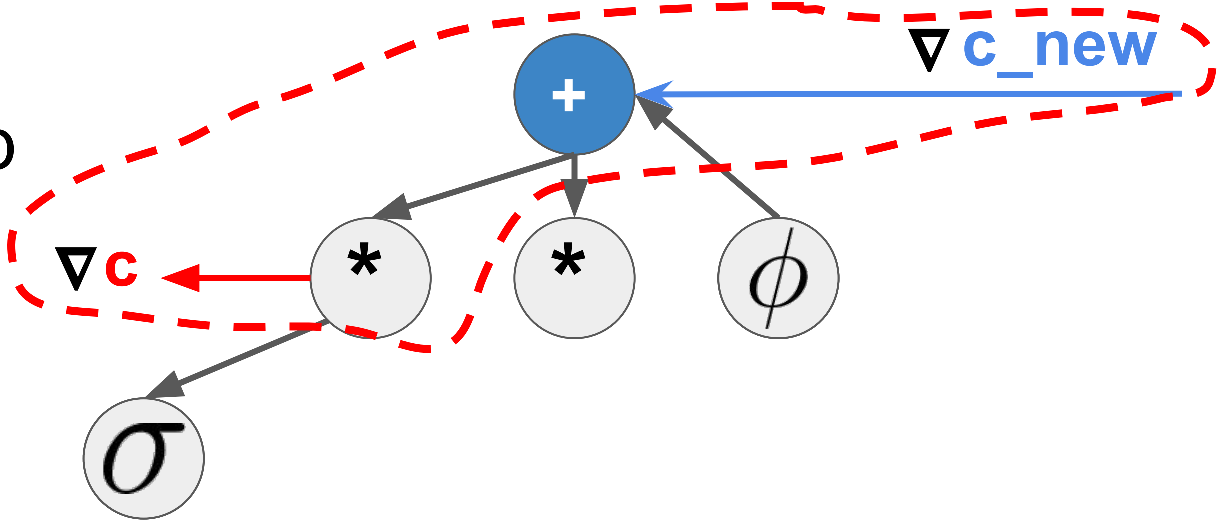
LSTM: Backprop



$$\nabla c_t = \sigma_t^f \odot (\nabla \phi_t (1 - \phi_t^2) + \nabla c_{t+1})$$

$$\nabla c_t = \boxed{\sigma_t^f} \odot \nabla c_{t+1} + v_t$$

LSTM: Backprop



$$\nabla c_0 = \boxed{\prod_{i=1}^{n} \sigma_{i-1}^{f}} \odot \nabla c_n + C$$

LSTM: Backprop

$$\nabla c_0 = \boxed{\prod_{i=1}^{n} \sigma_{i-1}^{f}} \odot \nabla c_n + C$$

Forget bias speeds up training!

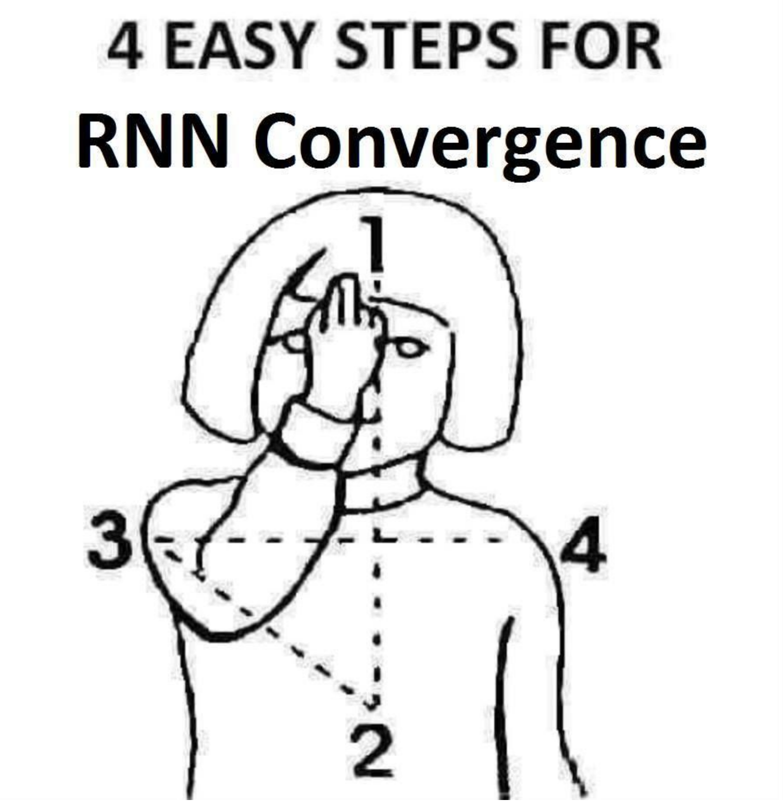
LSTM on Very-Long Sequence?

* Memory

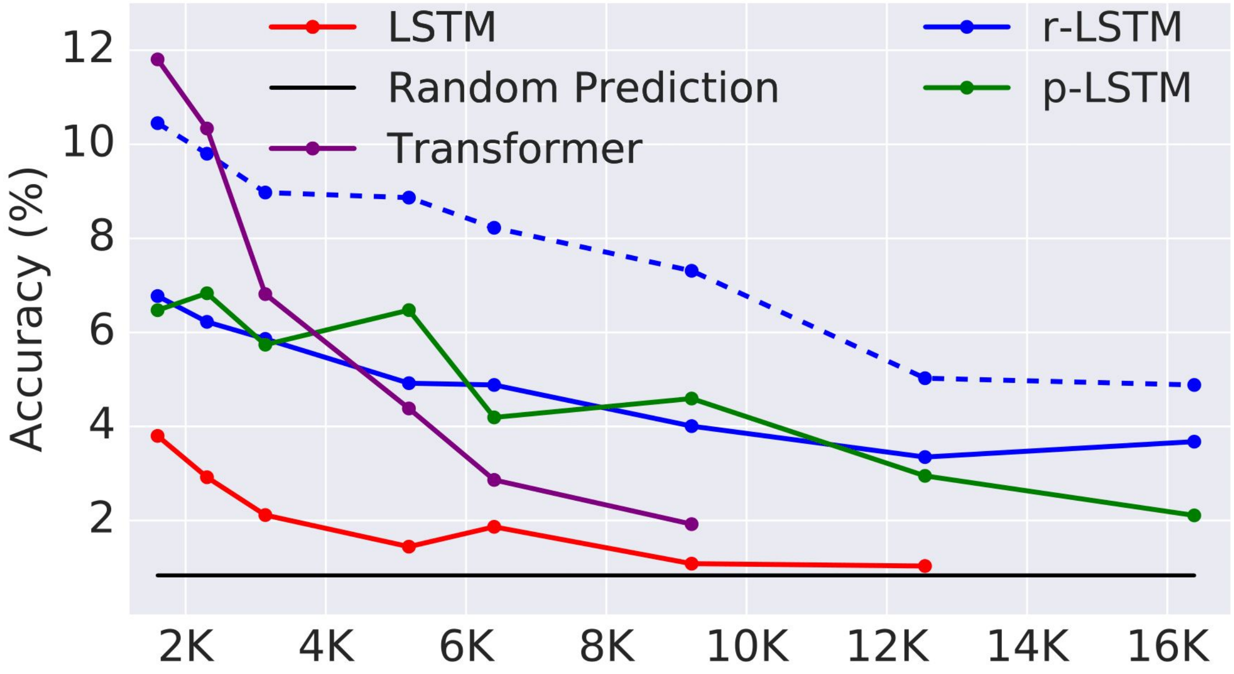
* Time

* Problem difficulty

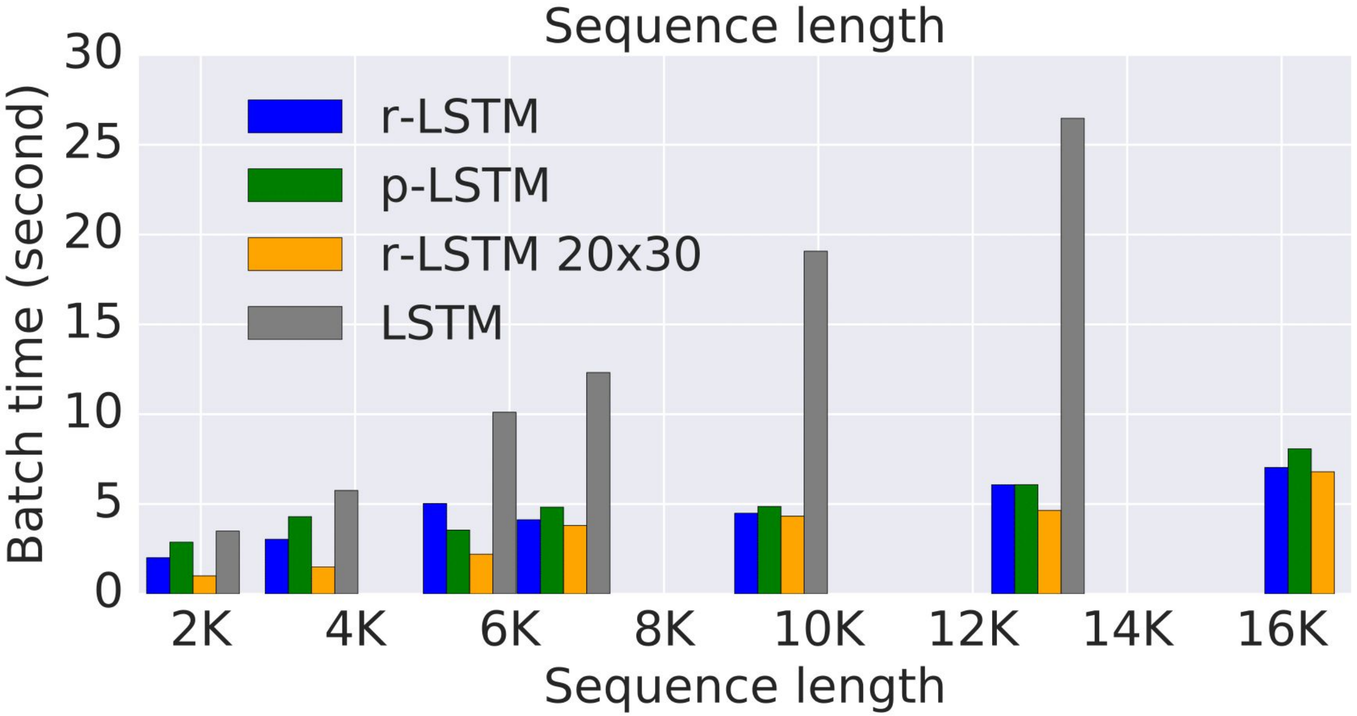
* Training difficulty



Teaser 003: Processing very-long sequence



Teaser 003: Processing very-long sequence



Teaser 003: Processing very-long sequence

Learning Longer-term Dependencies in RNNs with Auxiliary Losses

Trieu H. Trinh[1] **Andrew M. Dai** **Minh-Thang Luong** **Quoc V. Le**
{thtrieu, adai, thangluong, qvl}@google.com

[1]Work done as a member of the Google Brain Residency program (g.co/brainresidency.)

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

Stories time

Juergen Schmidhuber

to me

Sounds wonderful, Trieu!

All best,

Jürgen

Softmax Module

$$Softmax(\mathbf{x})_i = \frac{exp(\mathbf{x}_i)}{\sum_j exp(\mathbf{x}_j)}$$

Softmax Module: Overflow handling

* 1000 classes, 0 <= logits <= 300: Overflow (NaN values)

$$Softmax(\mathbf{x})_i = \frac{exp(\mathbf{x}_i)}{\sum_j exp(\mathbf{x}_j)}$$

Softmax Module: Overflow handling

* 1000 classes, 0 <= logits <= 300: Overflow (NaN values)

$$Softmax(\mathbf{x})_i = \frac{exp(\mathbf{x}_i)}{\sum_j exp(\mathbf{x}_j)}$$

$$= Softmax(\mathbf{x} - max(\mathbf{x}))$$

Softmax-Cross Entropy Module

https://gist.github.com/thtrieu/4ecf75d98fac77eb738cc8cb6ef47c81

Softmax-Cross Entropy Module

https://gist.github.com/thtrieu/4ecf75d98fac77eb738cc8cb6ef47c81

Faster than (Softmax **and then** Cross Entropy)

Quiz: Max module? (e.g. Max-pooling)

Quiz: Max module?

$$softmax(\alpha x) \overset{\alpha \to \infty}{\to} one\ hot$$

Quiz: Max module?

$$softmax(\alpha x) \overset{\alpha \to \infty}{\to} one\ hot$$

$$y = x^T softmax(\alpha x) \overset{\alpha \to \infty}{\to} \max(x)$$

Quiz: Max module?

$$softmax(\alpha x) \overset{\alpha \to \infty}{\to} one\ hot$$

$$y = x^T \boxed{\underset{s}{softmax(\alpha x)}} \overset{\alpha \to \infty}{\to} \max(x)$$

Quiz: Max module?

$$softmax(\alpha x) \overset{\alpha \to \infty}{\to} one\ hot$$

$$y = x^T \boxed{\underset{s}{\underbrace{softmax(\alpha x)}}} \overset{\alpha \to \infty}{\to} \max(x)$$

$$\nabla x = s \nabla y + \alpha(\mathbf{I}s - ss^T)\nabla y x$$

Quiz: Max module?

$$softmax(\alpha x) \overset{\alpha\to\infty}{\longrightarrow} one\ hot$$

$$y = x^T \boxed{\underset{s}{softmax(\alpha x)}} \overset{\alpha\to\infty}{\longrightarrow} \max(x)$$

$$\nabla x = s\nabla y + \boxed{\underset{\overset{\alpha\to\infty}{\longrightarrow}\ \mathbf{0}}{\alpha(\mathbf{I}s - ss^T)}}\nabla y x$$

Quiz: Max module?

$$softmax(\alpha x) \overset{\alpha\to\infty}{\longrightarrow} one\ hot$$

$$y = x^T \boxed{\underset{s}{softmax(\alpha x)}} \overset{\alpha\to\infty}{\longrightarrow} \max(x)$$

$$\boxed{\nabla x = s \nabla y} + \boxed{\underset{\alpha\to\infty \atop \longrightarrow \ \mathbf{0}}{\alpha(\mathbf{I}s - ss^T)}} \nabla y x$$

Quiz: Max module?

$$softmax(\alpha x) \xrightarrow{\alpha \to \infty} one\ hot$$

$$y = x^T \boxed{\underbrace{softmax(\alpha x)}_{s}} \xrightarrow{\alpha \to \infty} \max(x)$$

$$\boxed{\nabla x = s \nabla y \xrightarrow{\alpha \to \infty} [0..\nabla y..0]} + \boxed{\alpha(\mathbf{I}s - ss^T) \xrightarrow{\alpha \to \infty} \mathbf{0}} \nabla y x$$

Put it together!

```
c = graph_builder.add(a, b)  # tf.add(a, b)
```

Put it together!

```
c = graph_builder.add(a, b)  # tf.add(a, b)
```

1. Create the **module "Add"**

Put it together!

```
c = graph_builder.add(a, b)  # tf.add(a, b)
```

1. Create the **module "Add"**

2. Create a **Node**, containing **"Add"**

Put it together!

```
c = graph_builder.add(a, b)  # tf.add(a, b)
```

1. Create the **module "Add"**

2. Create a **Node**, containing **"Add"**

3. Set dependencies of **Node** to corresponding **Nodes of a and b**

Put it together!

```
c = graph_builder.add(a, b)  # tf.add(a, b)
```

1. Create the **module "Add"**

2. Create a **Node**, containing **"Add"**

3. Set dependencies of **Node** to corresponding **Nodes of a and b**

4. Return that **Node** to the user

Put it together!

```
c = graph_builder.add(a, b)  # tf.add(a, b)
```

1. Create the **module "Add"**

2. Create a **Node**, containing **"Add"**

3. Set dependencies of **Node** to corresponding **Nodes of a and b**

4. Return that **Node** to the user

```
d = graph_builder.square(c)
```

Put it together!

```
c = graph_builder.add(a, b)  # tf.add(a, b)
```

https://gist.github.com/thtrieu/5e02893fc6eed73046a97110fa051682

Put it together!

```
x = graph_builder.placeholder()
```

https://gist.github.com/thtrieu/7967c43809b1613e4ee1ba25033289cf

Put it together!

```
opt = graph_builder.sgd_optimizer(loss)

#  session.run(opt)
```

https://gist.github.com/thtrieu/89a849dd52806dae8cb4333fd1ca78fe

There are many more...

* Shape Inference

* Optimization for Convolution

* Closure set

Can we run something now?

Yes!

Everything so far **and beyond**

https://github.com/thtrieu/essence/tree/master/src

Can we run something now?

Yes!

Everything so far **and beyond**

https://github.com/thtrieu/essence/tree/master/src

Recommended read:

- **Gradients checking**
- **Convolution optimization**

Not quite Tensorflow

* GPU

* Low-level optimization

* Distributed training

Not quite Tensorflow

* GPU

* Low-level optimization

* Distributed training

Technical

Not quite Tensorflow

* GPU

* Low-level optimization

* Distributed training

Technical

tf.gradients

Second-order derivatives

Conceptual

Not quite Tensorflow ... (yet)

* GPU

* Low-level optimization

* Distributed training

Technical

tf.gradients

Second-order derivatives

https://gist.github.com/thtrieu/a5268745a70dabb5f413cf21df50b8c7

Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent

Marcin Andrychowicz[1], Misha Denil[1], Sergio Gómez Colmenarejo[1], Matthew W. Hoffman[1], David Pfau[1], Tom Schaul[1], Brendan Shillingford[1,2], Nando de Freitas[1,2,3]

[1]Google DeepMind [2]University of Oxford [3]Canadian Institute for Advanced Research

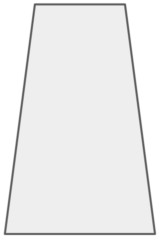
tf.gradients

Second-order derivatives

https://gist.github.com/thtrieu/a5268745a70dabb5f413cf21df50b8c7

Not quite Tensorflow ... (yet)

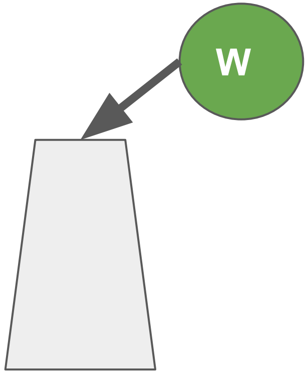
Learning to learn by gradient descent by gradient descent



Forward DAG

Not quite Tensorflow ... (yet)

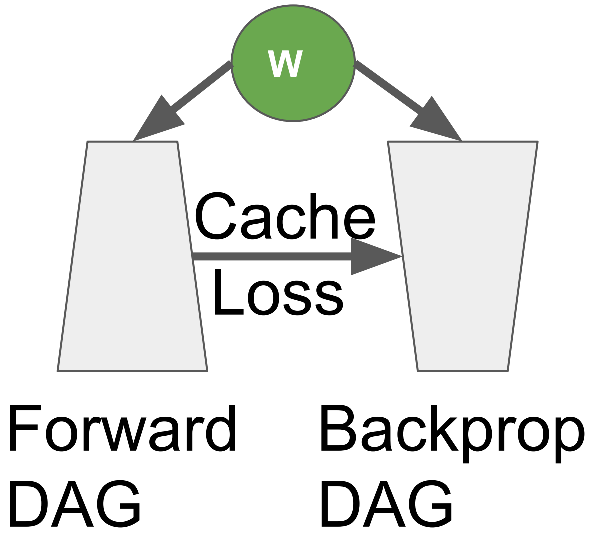
Learning to learn by gradient descent by gradient descent



Forward
DAG

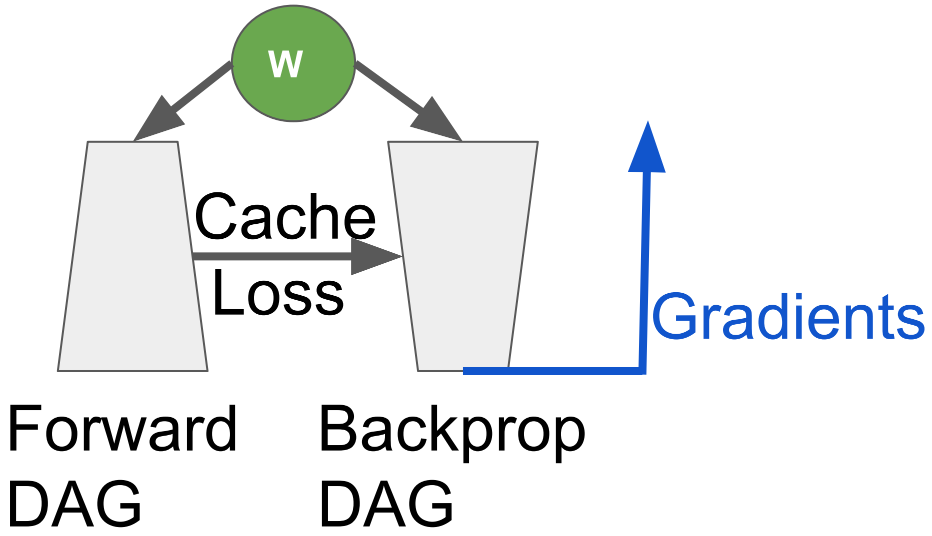
Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent



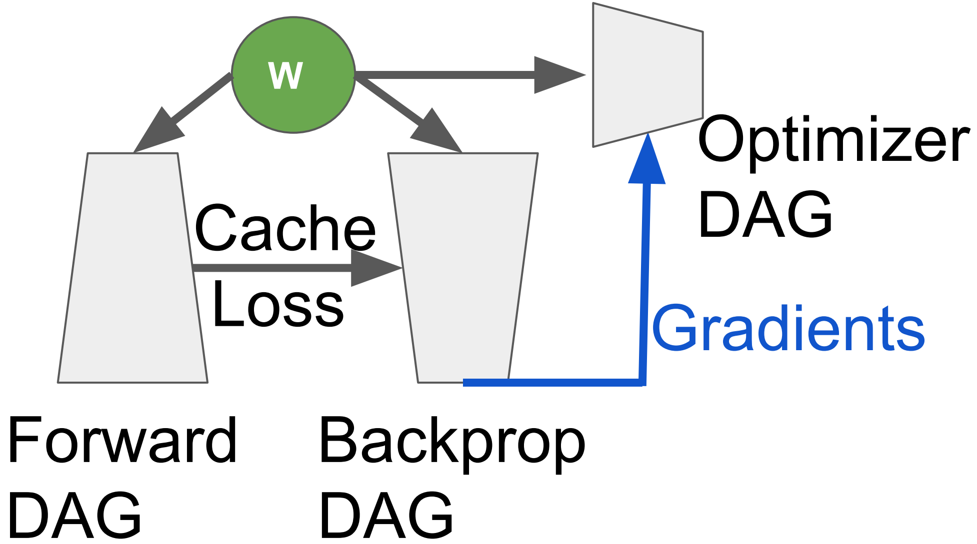
Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent



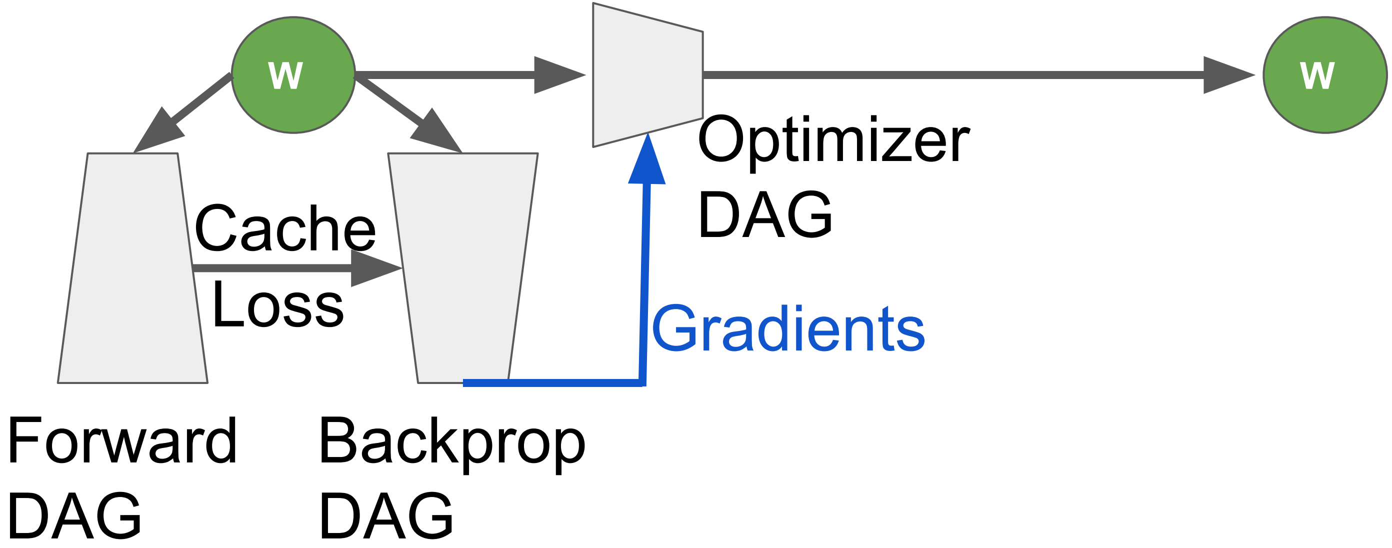
Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent



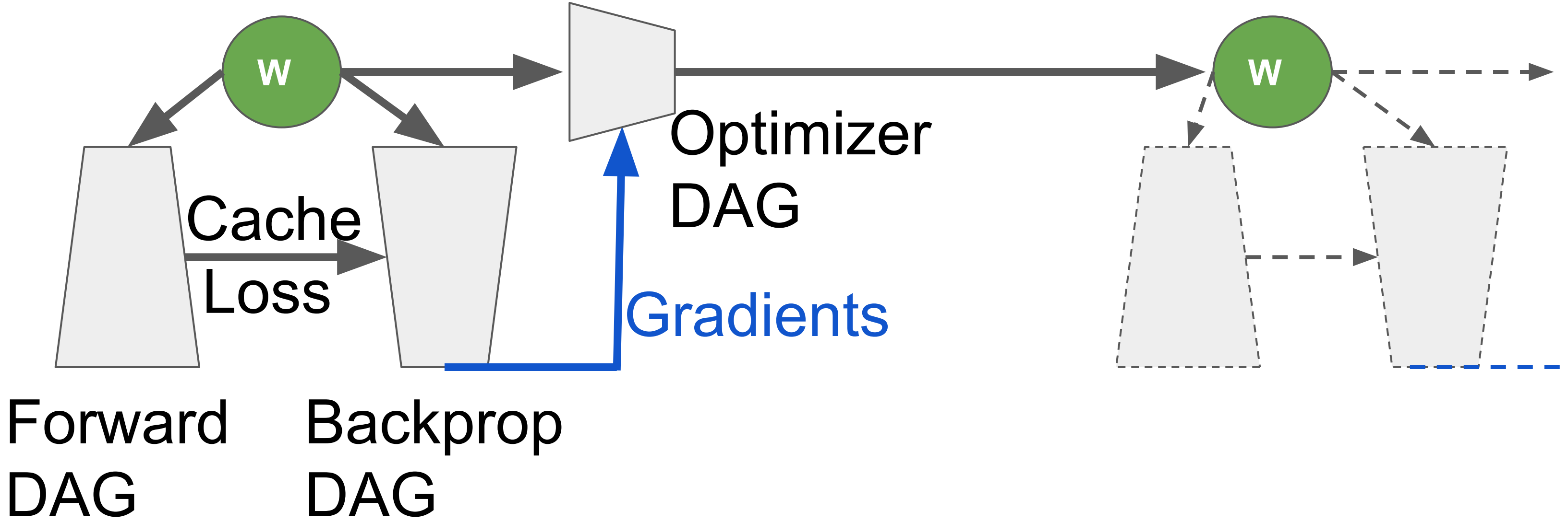
Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent



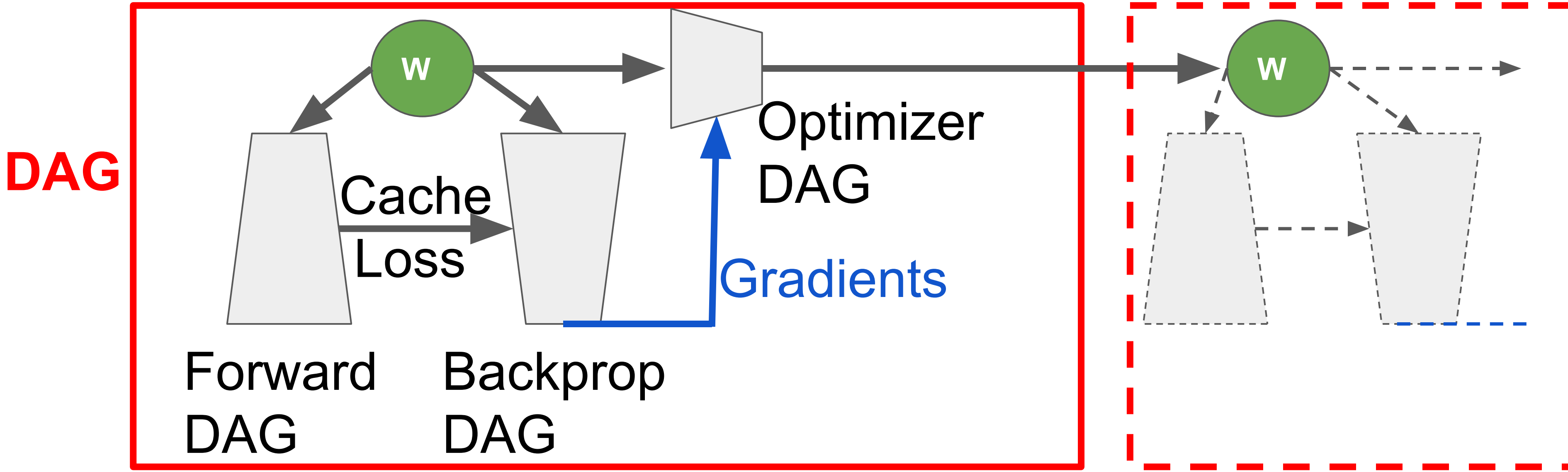
Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent



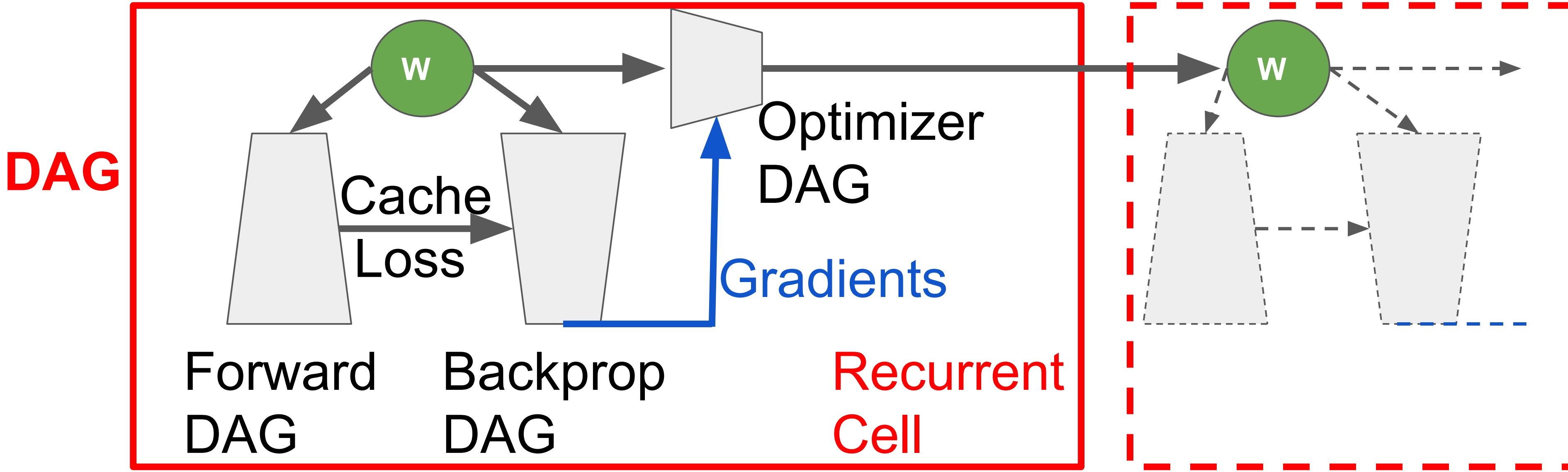
Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent



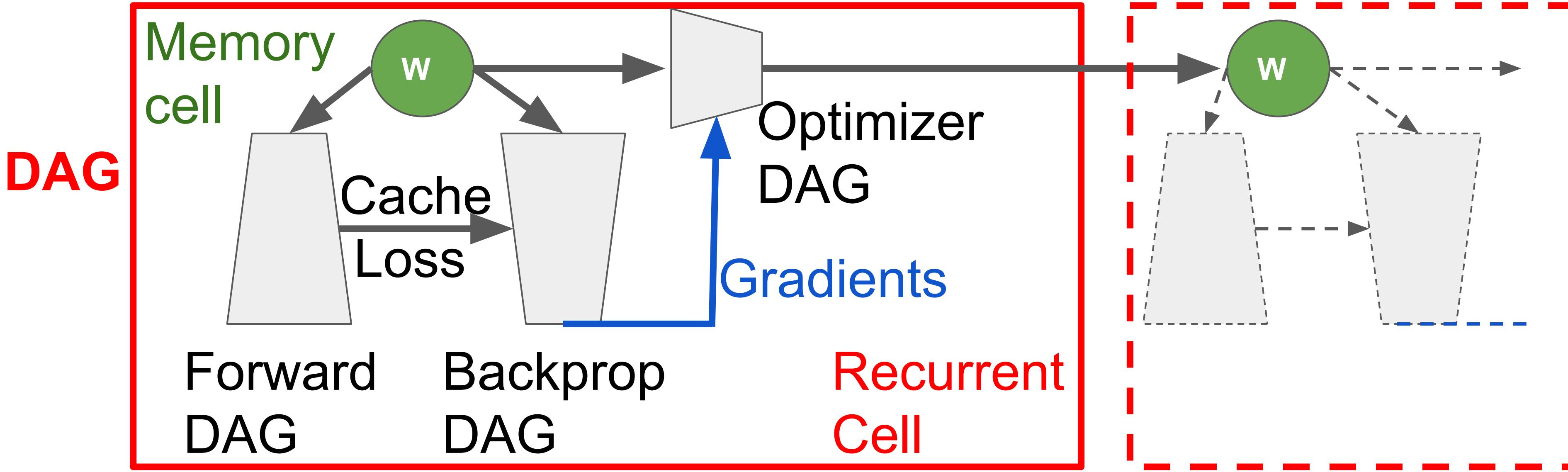
Not quite Tensorflow ... (yet)

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



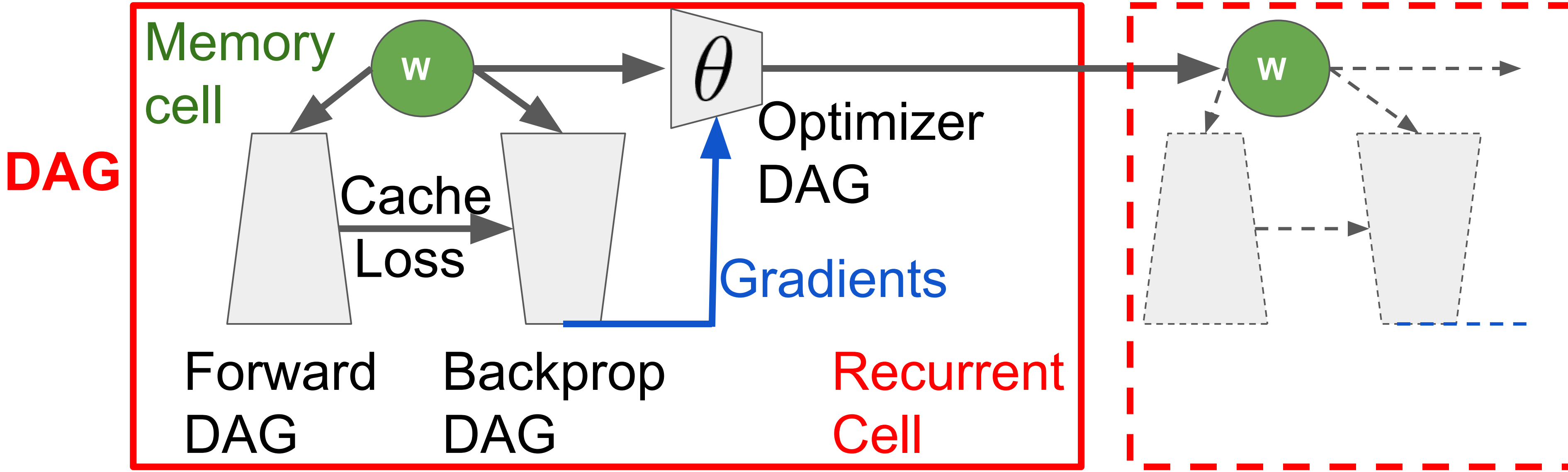
Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent



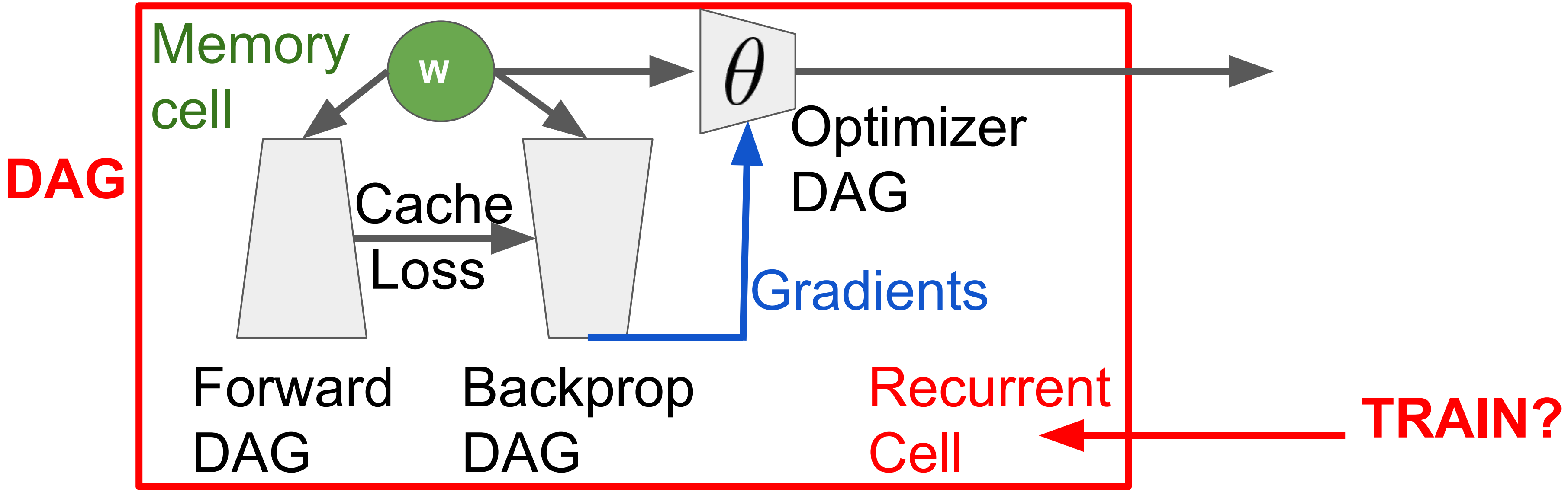
Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent



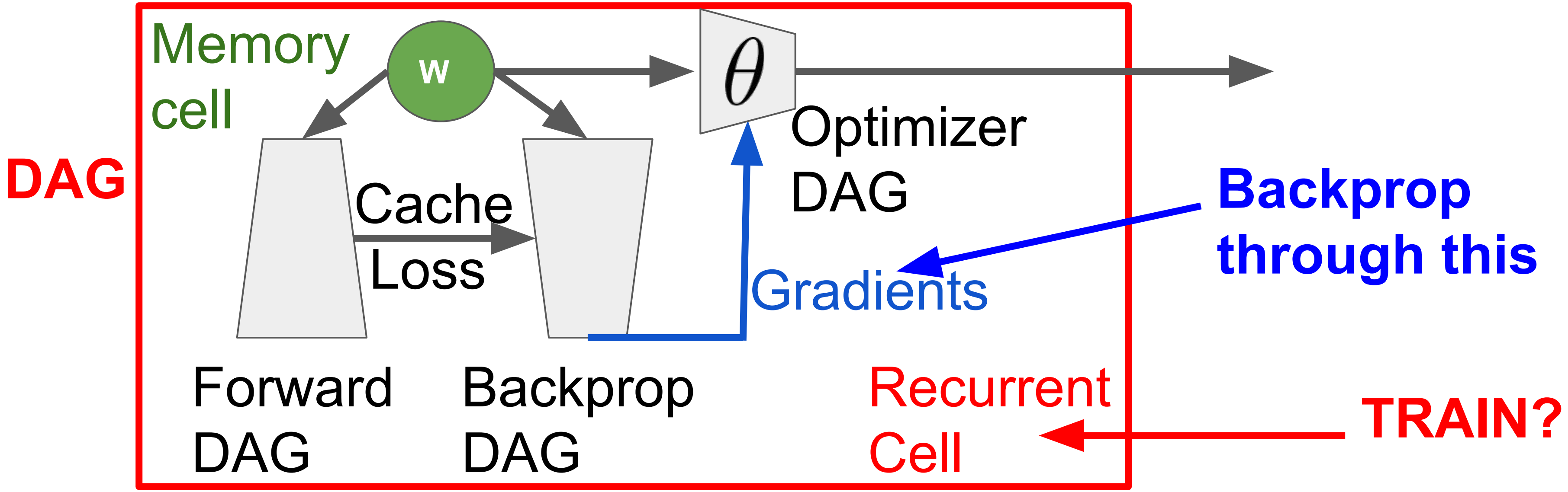
Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent



Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent



Not quite Tensorflow ... (yet)

Learning to learn by gradient descent by gradient descent

Learning Unsupervised Learning Rules

Luke Metz
Google Brain
lmetz@google.com

Niru Maheswaranathan
Google Brain
nirum@google.com

Brian Cheung
University of California, Berkeley
bcheung@berkeley.edu

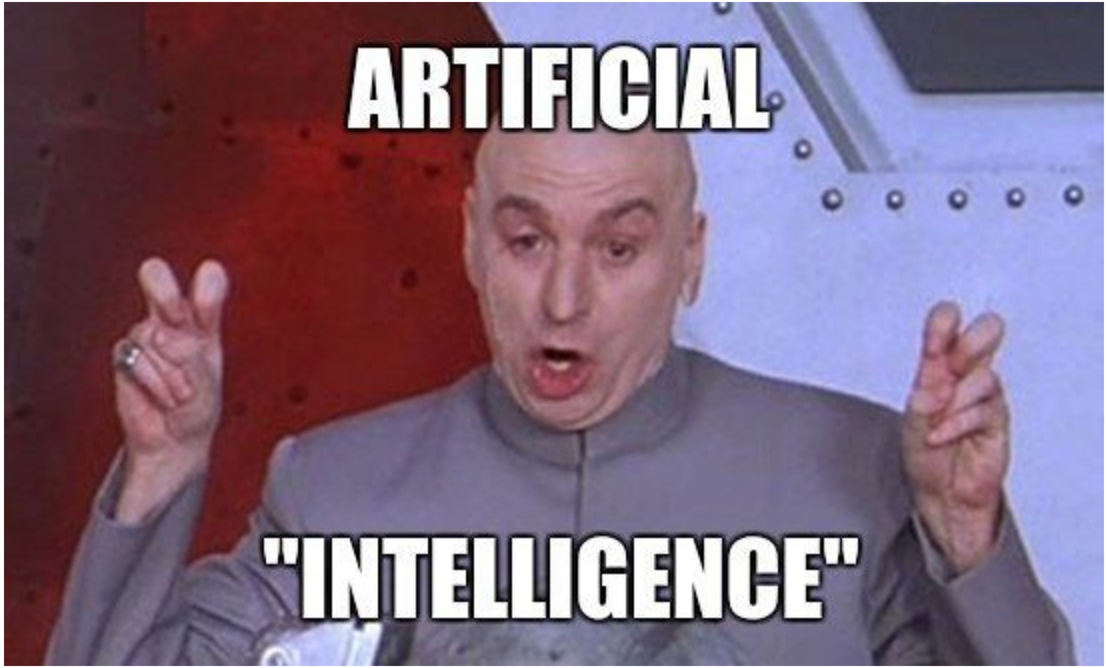
Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

And that's it!

Will be shared:

* Slides

* Gists



Please add your questions/comments **(in En)**