Psych 253
Advanced Statistical Modeling

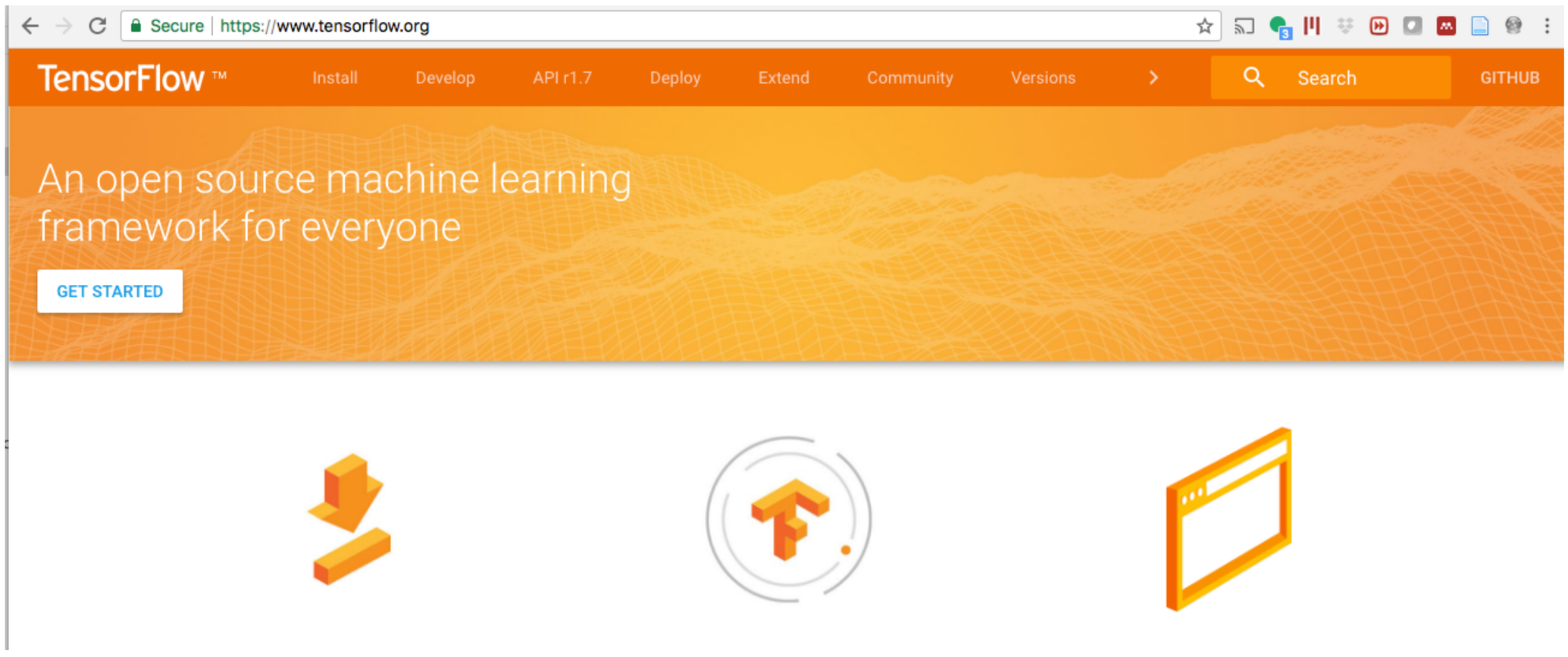
Intro to Tensorflow

Daniel Yamins
Wu Tsai Neurosciences Institute
Departments of Psychology and Computer Science
Stanford Artificial Intelligence Laboratory
Stanford University

Russ Poldrack
Department of Psychology
Stanford University

Graph Model of Computation

Tensorflow is a graph computation language.



Graph Model of Computation

$$e=(a+b)*(b+1)$$

Graph Model of Computation

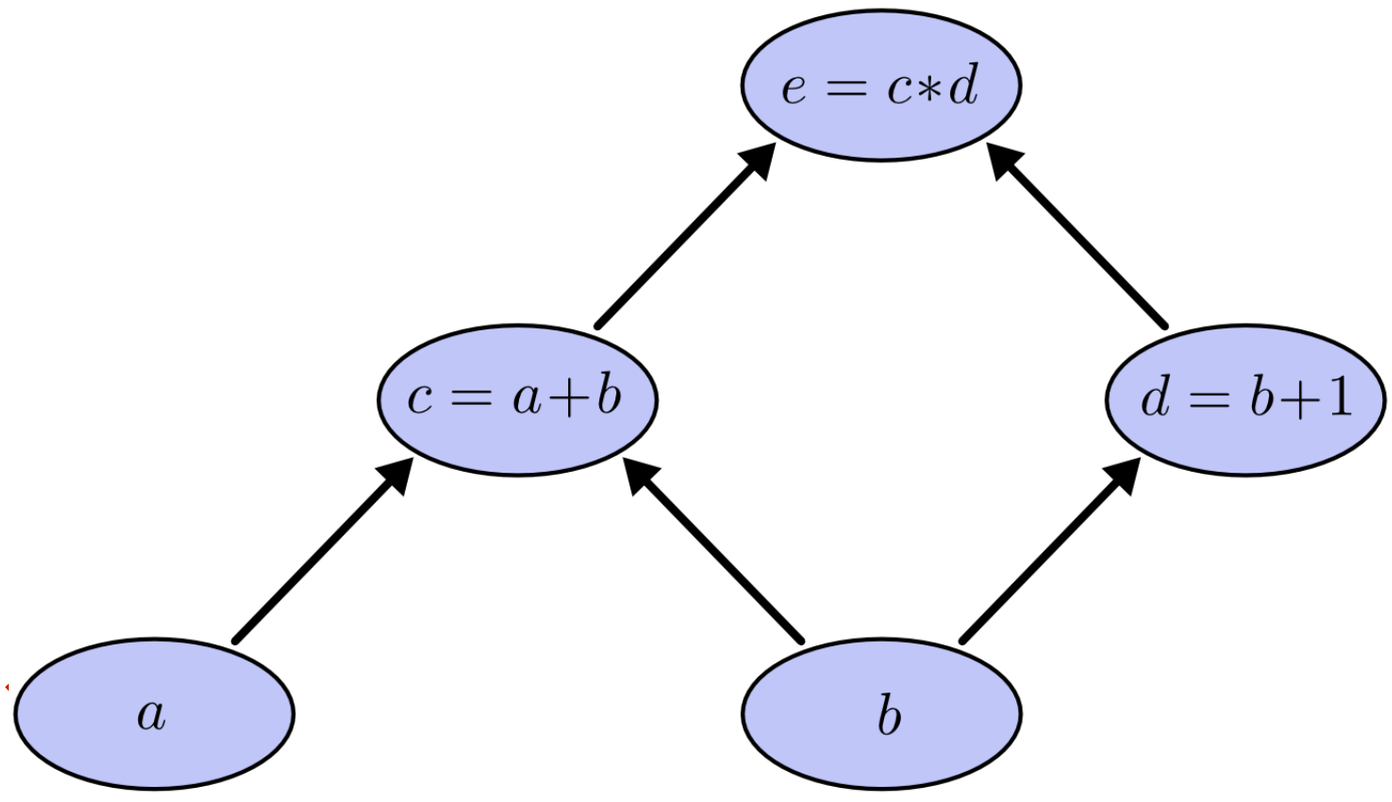
e=(a+b)*(b+1)

decomposes into 5 "nodes":

c=a+b

d=b+1

e=c*d



It's a Directed Acyclic Graph (DAG)

Graph Model of Computation

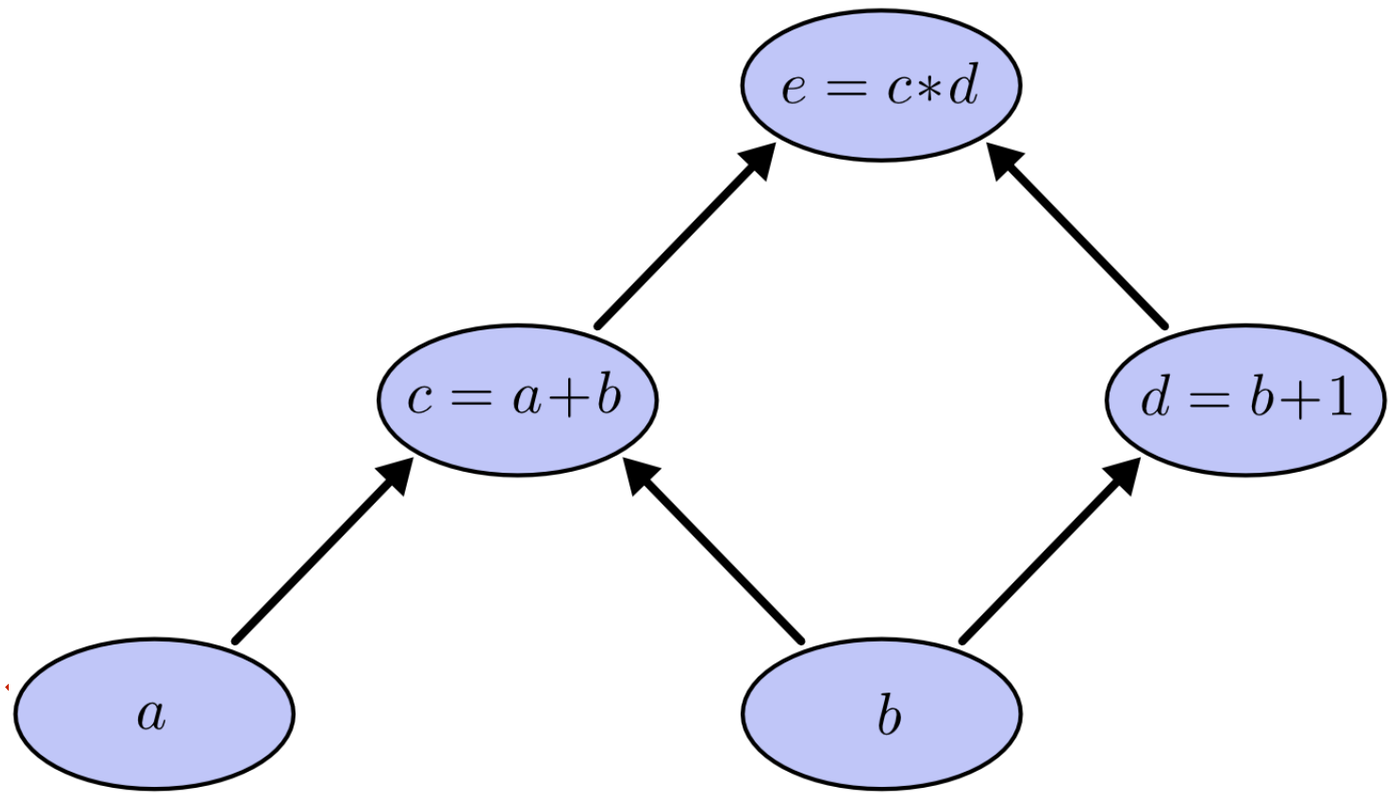
$$e=(a+b)*(b+1)$$

decomposes into 5 "nodes":

c=a+b

d=b+1

e=c*d



Ok let's see what happens if a=2 and b = 1.

Graph Model of Computation

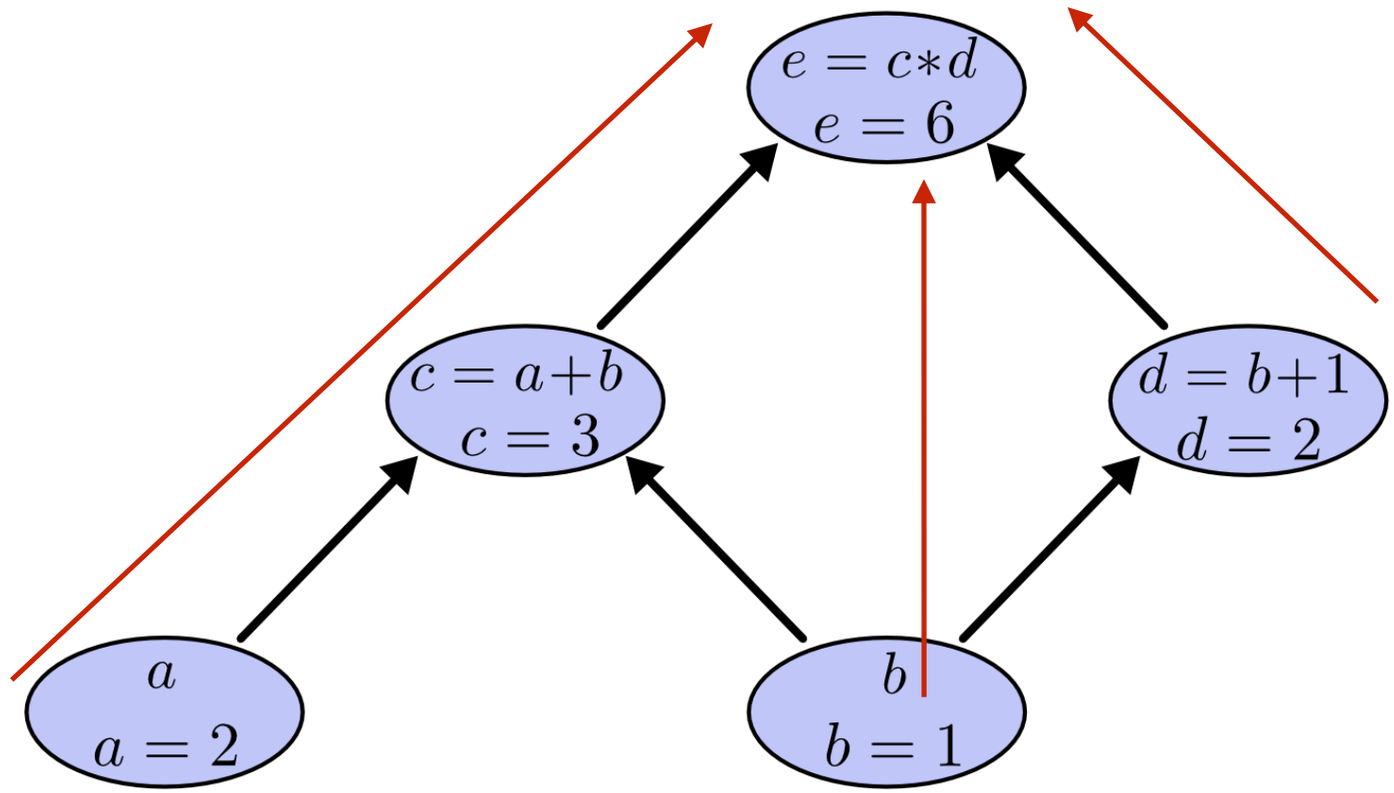
e=(a+b)*(b+1)

decomposes into 5 "nodes":

c=a+b

d=b+1

e=c*d



Ok let's see what happens if a=2 and b = 1.

Tensorflow

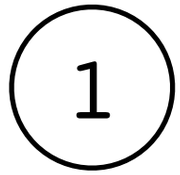
Agenda:

- Tensor constants
- Tensor operations
- Tensor variables
- Automatic Gradients

[ipynb]

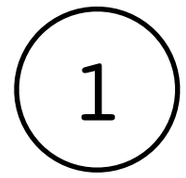
Tensor Constants

```
one = tf.constant(1)
```

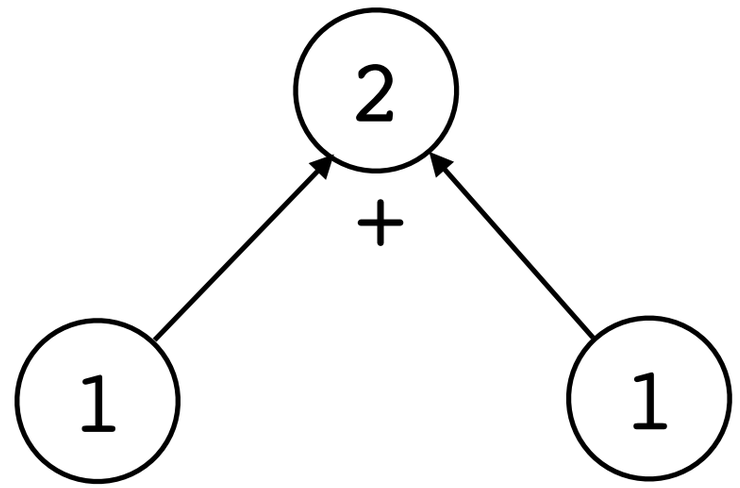


Tensor Constants

```
one = tf.constant(1)
```



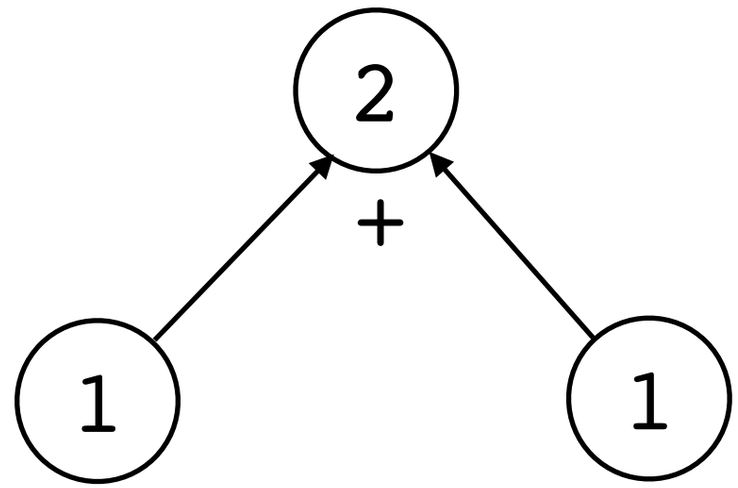

```
two = one + one
```



Tensor Constants

```
one = tf.constant(1)
```

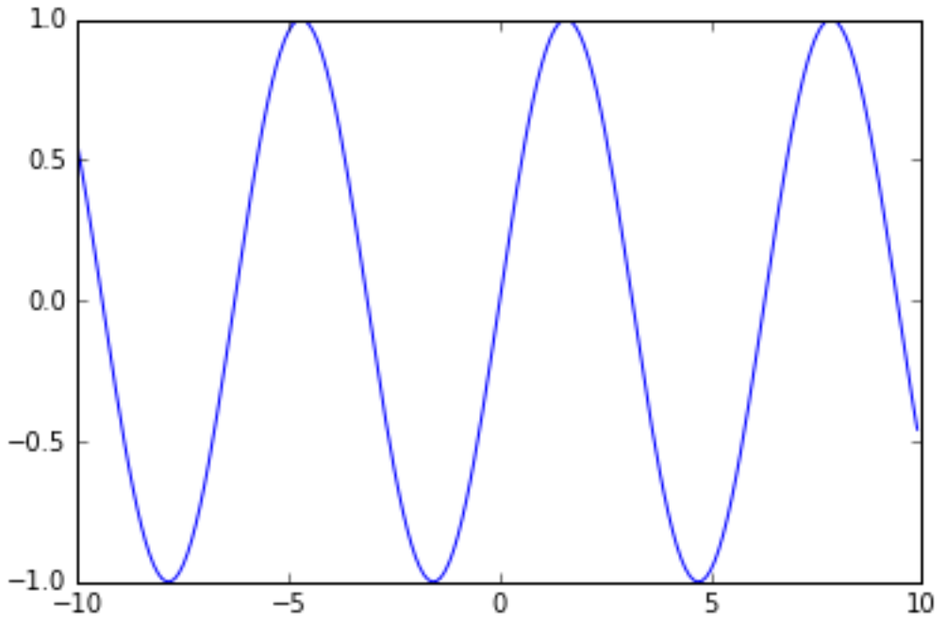

```
two = one + one
```



+ matrices and tensors with shapes just like in NumPy.

Tensor Operations

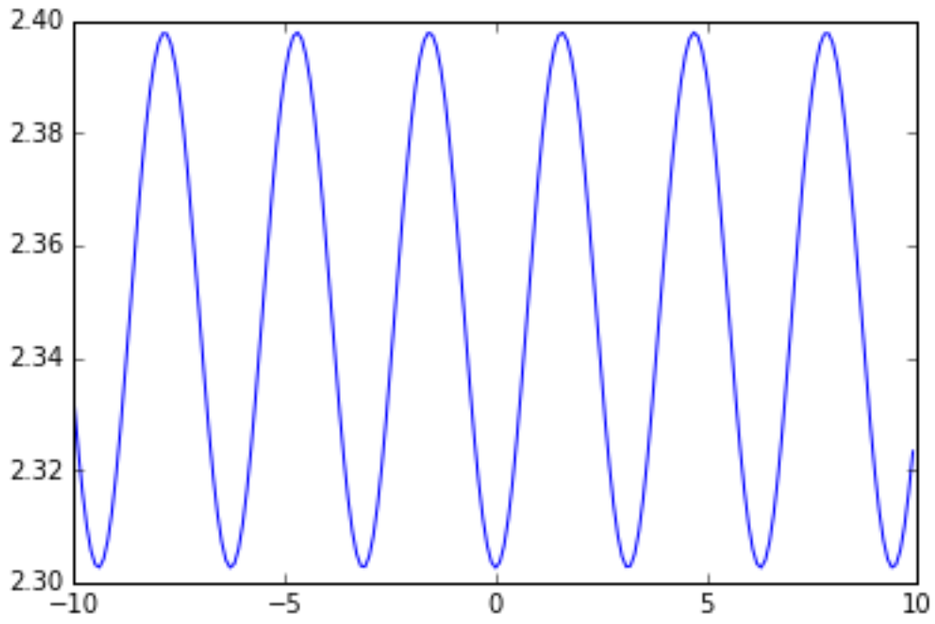
```
x = tf.range(-10, 10, .1)
y = tf.sin(x)

plt.plot(x, y)
```



Tensor Operations

```
x = tf.range(-10, 10, .1)
y = tf.sin(x)
z = y**2 + 10
w = tf.log(z)
```

```
plt.plot(x, y)
```



Tensor Operations

Element-wise multiplication:

```
m1 * m2
```

```
m1.shape == m2.shape
```

(NumPy broadcasting rules apply)

Matrix multiplication:

```
tf.matmul(m1, m2)
```

```
m1.shape[1] == m2.shape[0]
```

Tensor Operations

Convolution:

constant filter

does blurring

```
blurred = tf.nn.conv2d(imtensor, filterarray,
    strides=[1, 1, 1, 1], padding='VALID')
```

Having constructed variables

```
neural_data.shape —> (batch_size, num_neurons)

category_labels.shape —> (batch_size, num_categories)

weights.shape —> (num_neurons, num_categories)

bias.shape -> (num_categories,)
```

We then make hinge loss like so:

```
margins = tf.matmul(neural_data, weights) + bias

hinge_loss = tf.maximum(0., 1. - category_labels * margins)
```

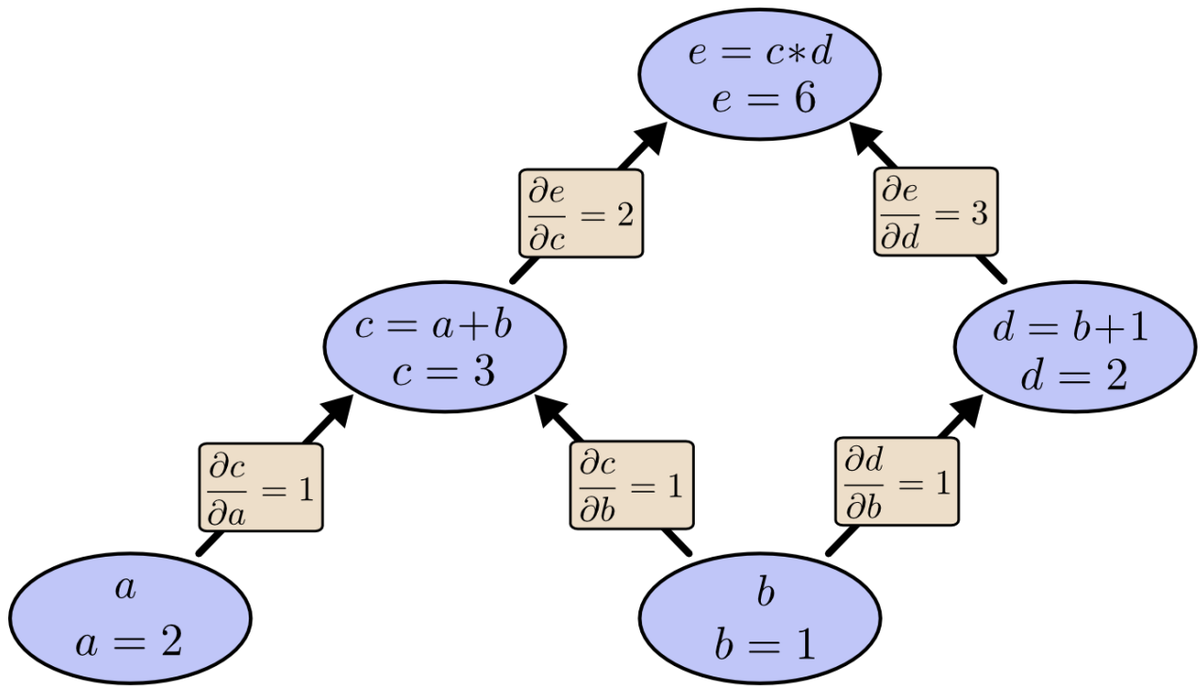
Graph Model of Computation

$$\frac{\partial}{\partial a}(a+b) = \frac{\partial a}{\partial a} + \frac{\partial b}{\partial a}$$

$$\frac{\partial}{\partial u} u \cdot v = u \cdot \frac{\partial v}{\partial u} + \frac{\partial u}{\partial u} \cdot v = v$$

a changes at rate 1

=> c changes at rate 1



Graph Model of Computation

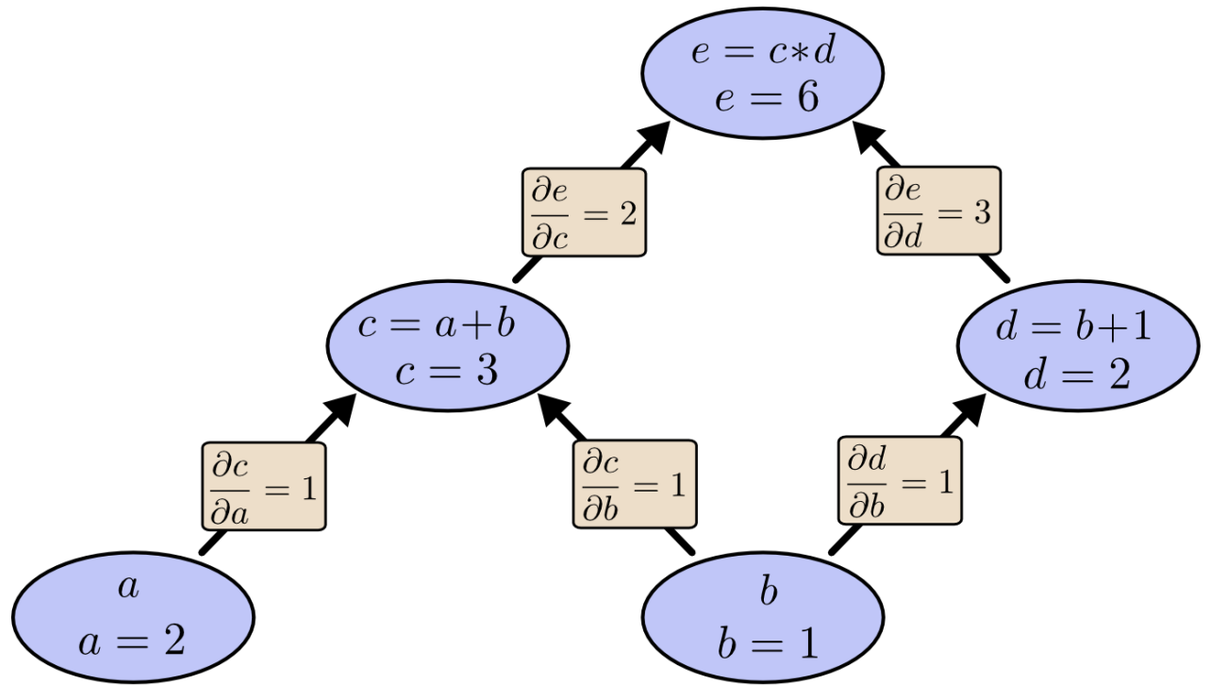
$$\frac{\partial}{\partial a}(a+b) = \frac{\partial a}{\partial a} + \frac{\partial b}{\partial a}$$

$$\frac{\partial}{\partial u} u \cdot v = u \cdot \frac{\partial v}{\partial u} + \frac{\partial u}{\partial u} \cdot v = v$$

a changes at rate 1
=> c changes at rate 1

c changes at rate 1
=> e changes at rate 2

$$\frac{\partial e}{\partial a} = 1 \cdot 2 = 2$$



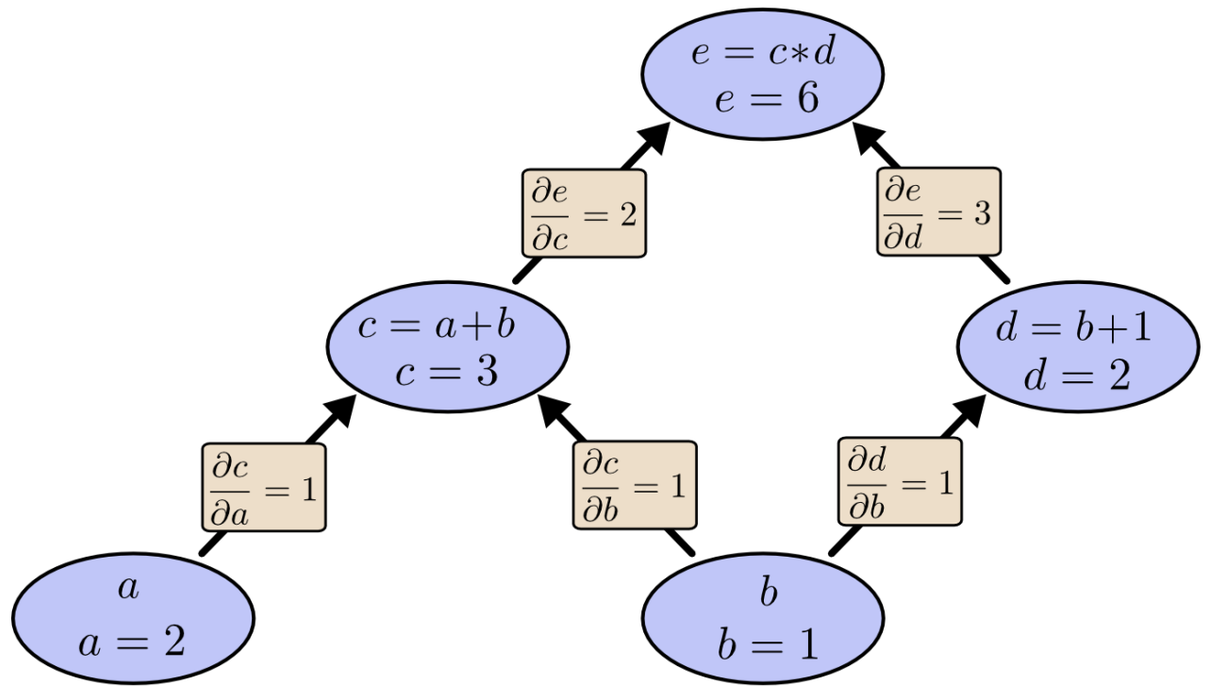
Graph Model of Computation

$$\frac{\partial}{\partial a}(a+b) = \frac{\partial a}{\partial a} + \frac{\partial b}{\partial a} \qquad \frac{\partial}{\partial u} u \cdot v = u \cdot \frac{\partial v}{\partial u} + \frac{\partial u}{\partial u} \cdot v = v$$

a changes at rate 1
=> c changes at rate 1

c changes at rate 1
=> e changes at rate 2

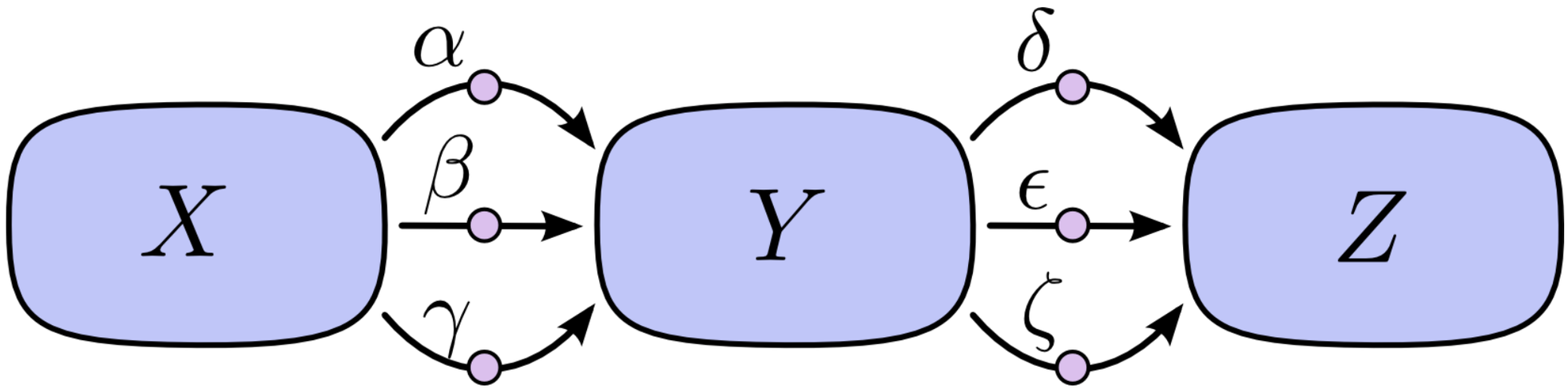
$$\frac{\partial e}{\partial a} = 1 \cdot 2 = 2$$



effect of n1 on n2: sum over all paths between n1 and n2, multiply across edges

$$\frac{\partial e}{\partial b} = 1 \cdot 2 + 1 \cdot 3 = 6$$

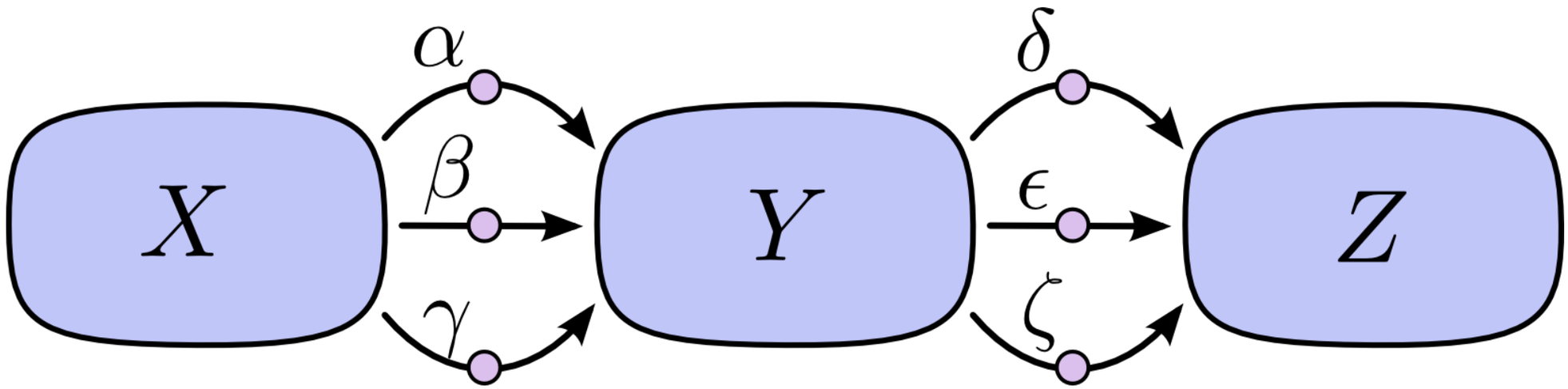
Graph Model of Computation



$$\frac{\partial Z}{\partial X} = \alpha\delta + \alpha\epsilon + \alpha\zeta + \beta\delta + \beta\epsilon + \beta\zeta + \gamma\delta + \gamma\epsilon + \gamma\zeta$$

How to prevent combinatorial explosion?

Graph Model of Computation



$$\frac{\partial Z}{\partial X} = \alpha\delta + \alpha\epsilon + \alpha\zeta + \beta\delta + \beta\epsilon + \beta\zeta + \gamma\delta + \gamma\epsilon + \gamma\zeta$$

How to prevent combinatorial explosion? **Factor and merge.**

$$\frac{\partial Z}{\partial X} = (\alpha + \beta + \gamma) \cdot (\delta + \epsilon + \zeta)$$

Graph Model of Computation

How to prevent combinatorial explosion? **Factor and merge.**

$$\frac{\partial Z}{\partial X} = (\alpha + \beta + \gamma) \cdot (\delta + \epsilon + \zeta)$$

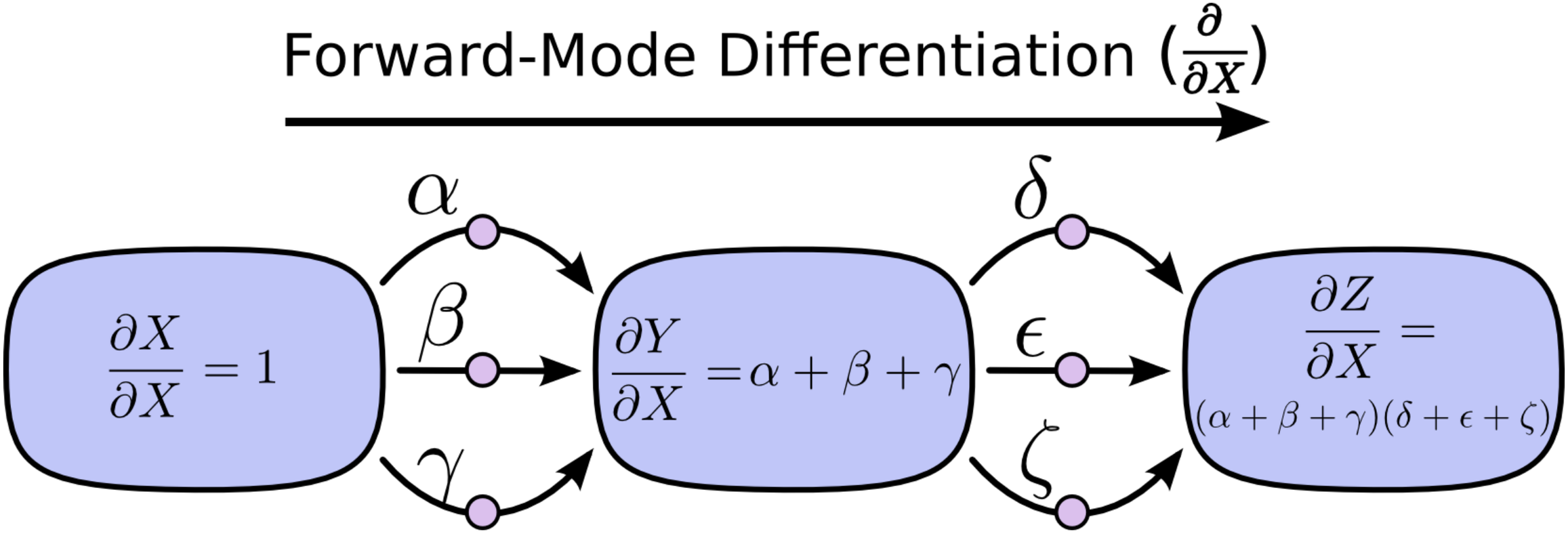
Method 1: start at an input, move up; sum all paths as you go.

Graph Model of Computation

How to prevent combinatorial explosion? **Factor and merge.**

$$\frac{\partial Z}{\partial X} = (\alpha + \beta + \gamma) \cdot (\delta + \epsilon + \zeta)$$

Method 1: start at an input, move up; sum all paths as you go.



Graph Model of Computation

How to prevent combinatorial explosion? **Factor and merge.**

$$\frac{\partial Z}{\partial X} = (\alpha + \beta + \gamma) \cdot (\delta + \epsilon + \zeta)$$

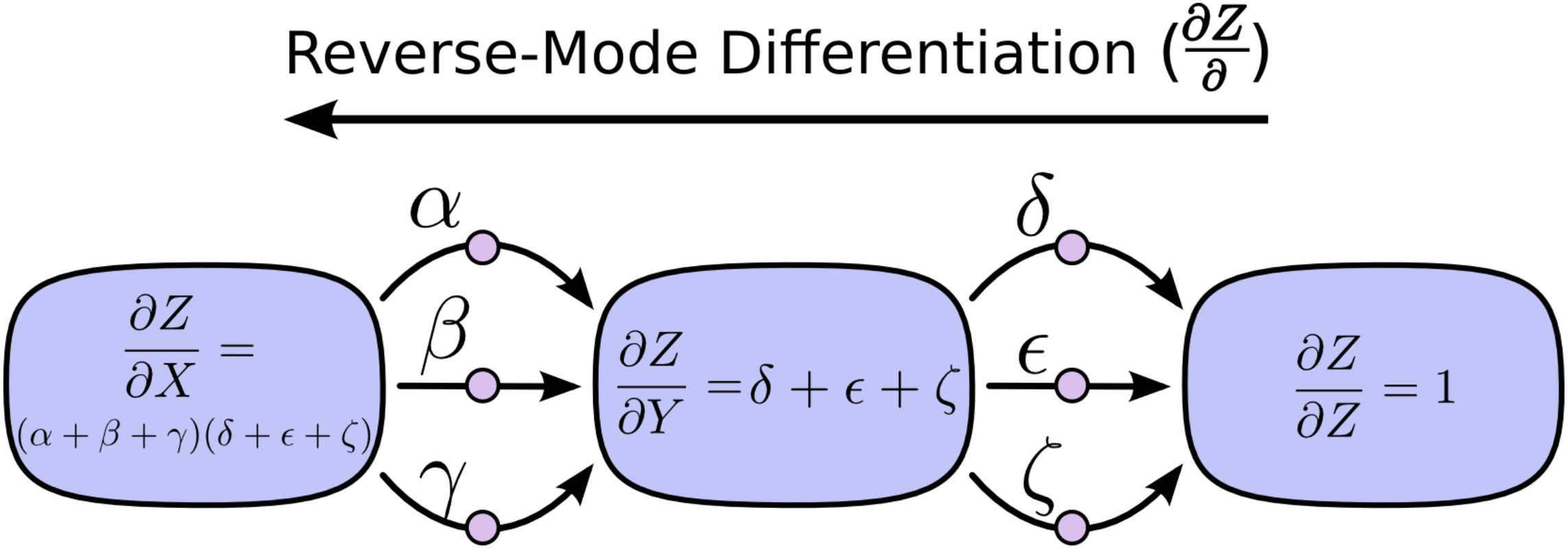
Method 2: start at the output, move down; sum all paths as you go.

Graph Model of Computation

How to prevent combinatorial explosion? **Factor and merge.**

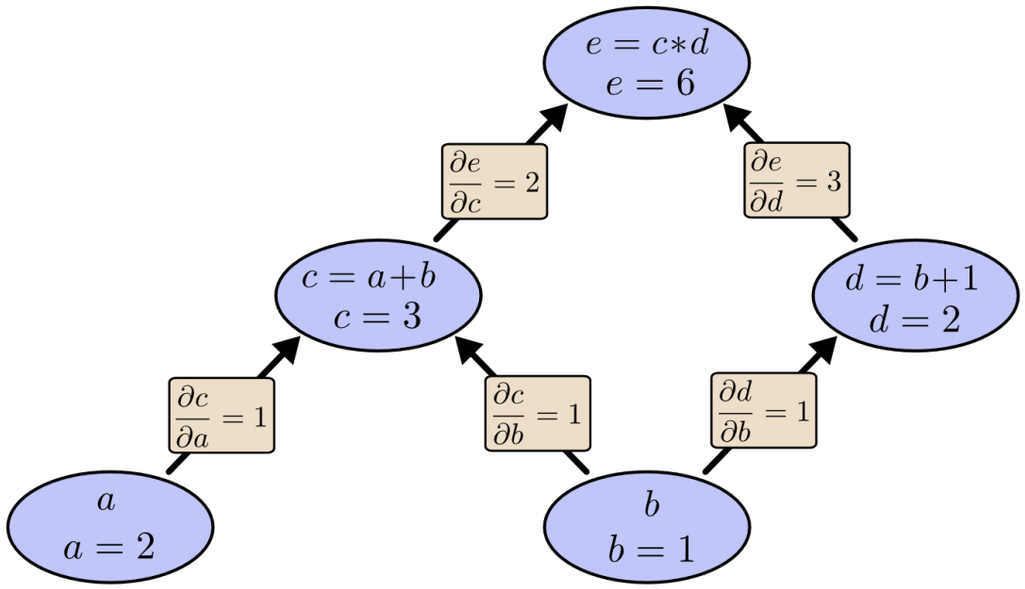
$$\frac{\partial Z}{\partial X} = (\alpha + \beta + \gamma) \cdot (\delta + \epsilon + \zeta)$$

Method 2: start at the output, move down; sum all paths as you go.



Graph Model of Computation

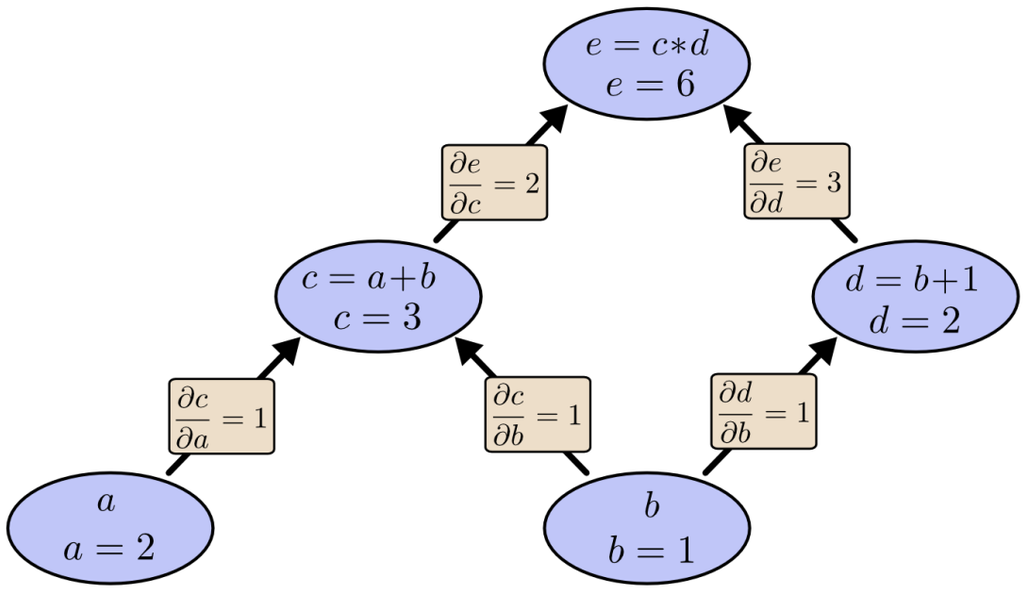
Goal: calculate $\dfrac{\partial e}{\partial b}$

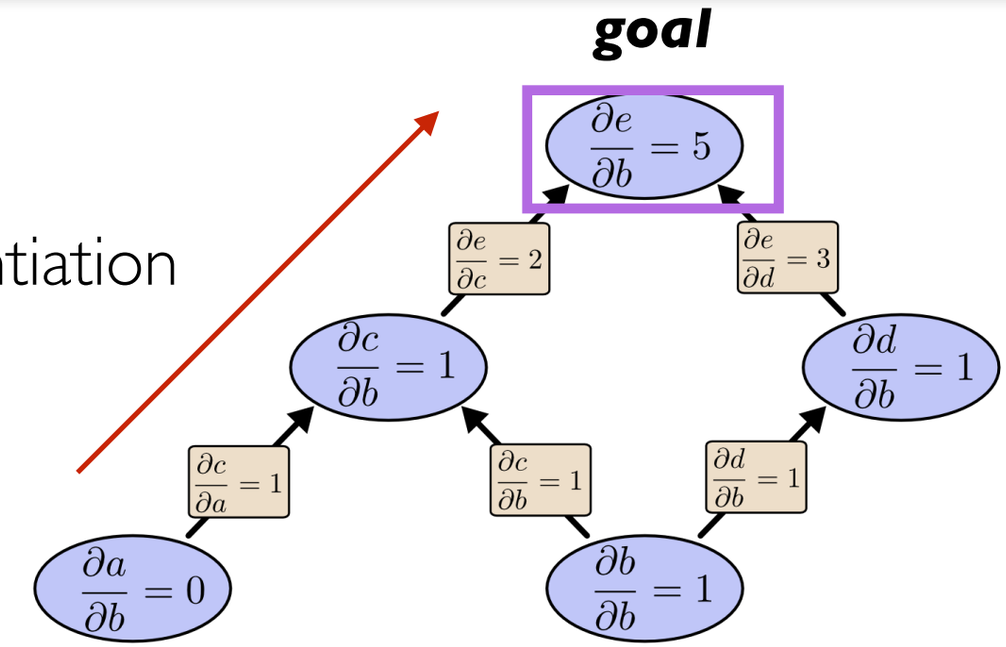


Graph Model of Computation

Goal: calculate $\frac{\partial e}{\partial b}$

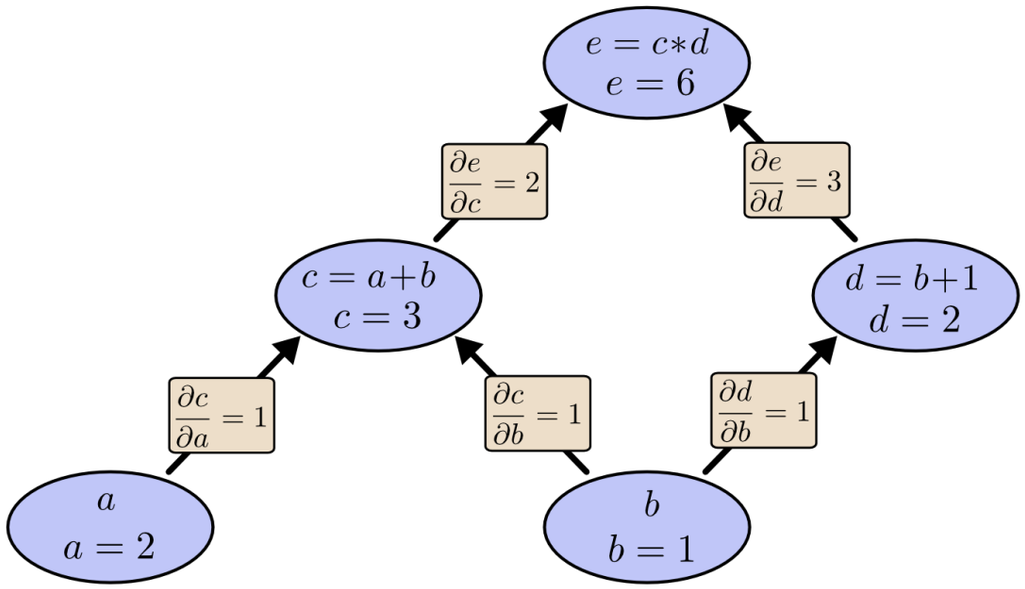
Forward differentiation



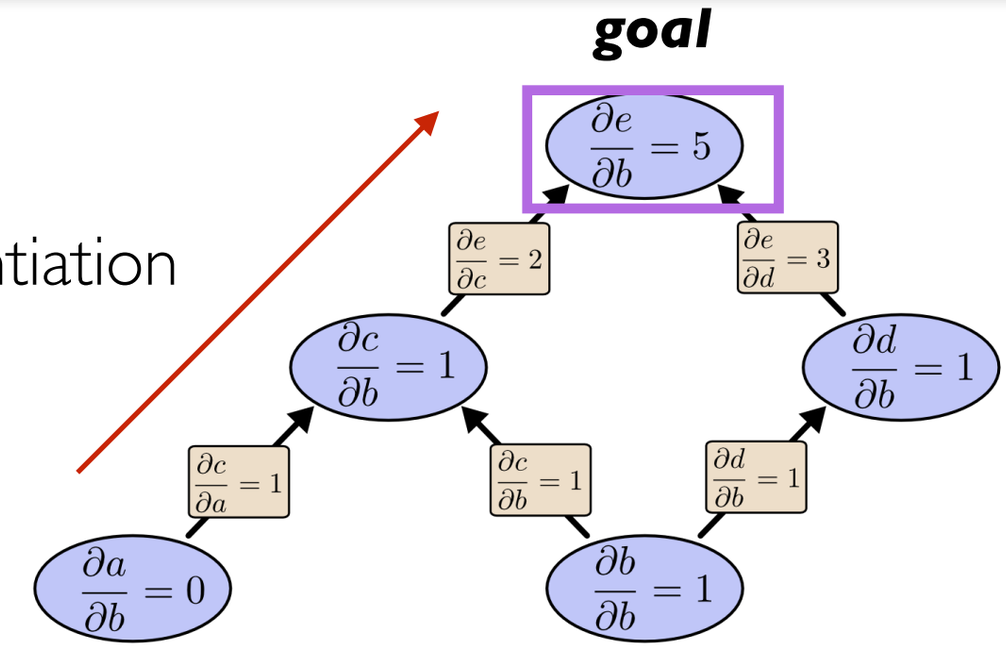


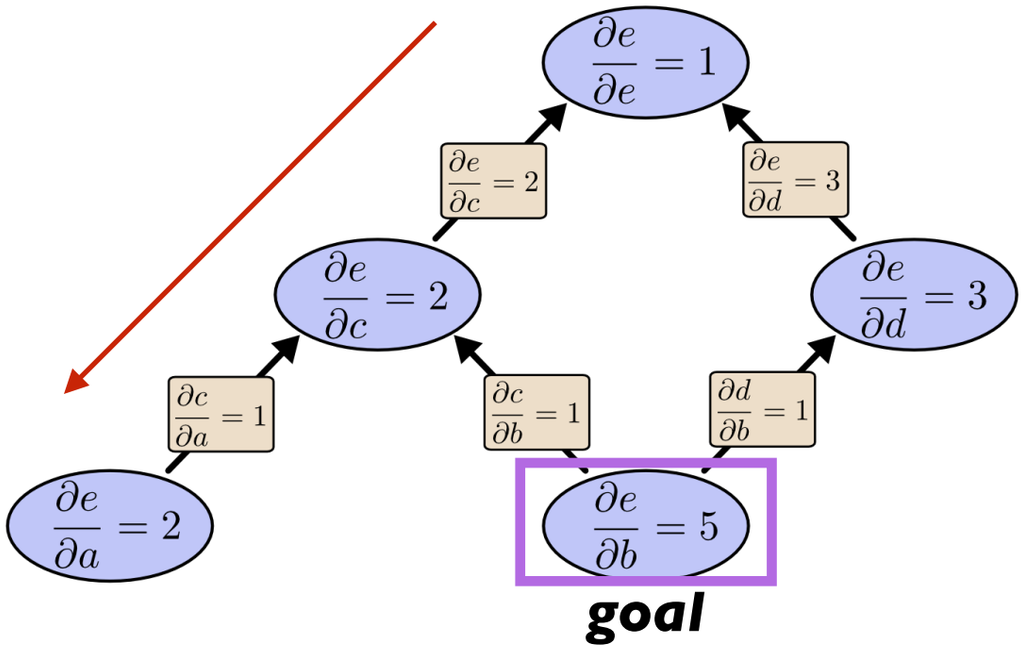
Graph Model of Computation

Goal: calculate $\frac{\partial e}{\partial b}$



Forward differentiation

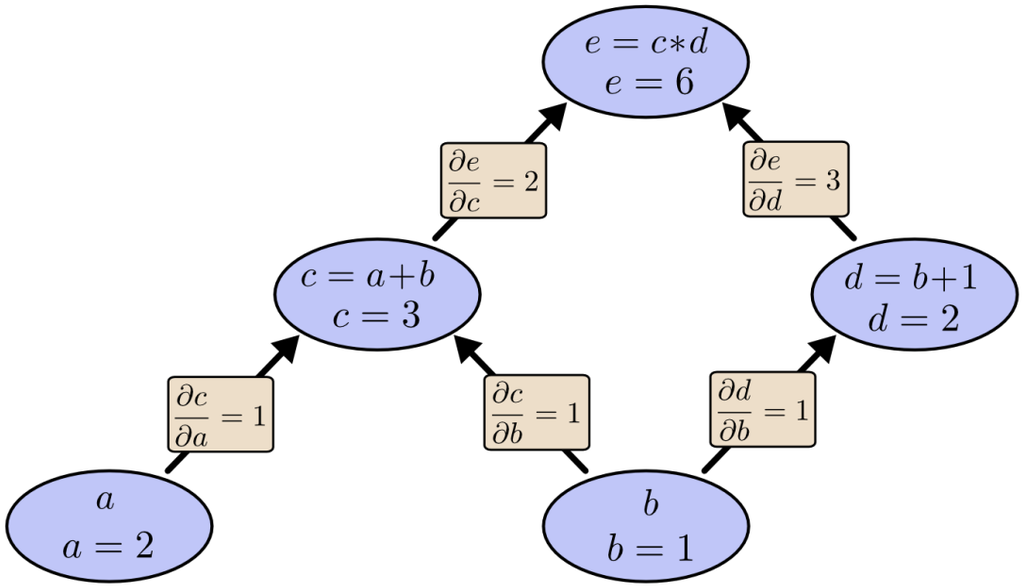




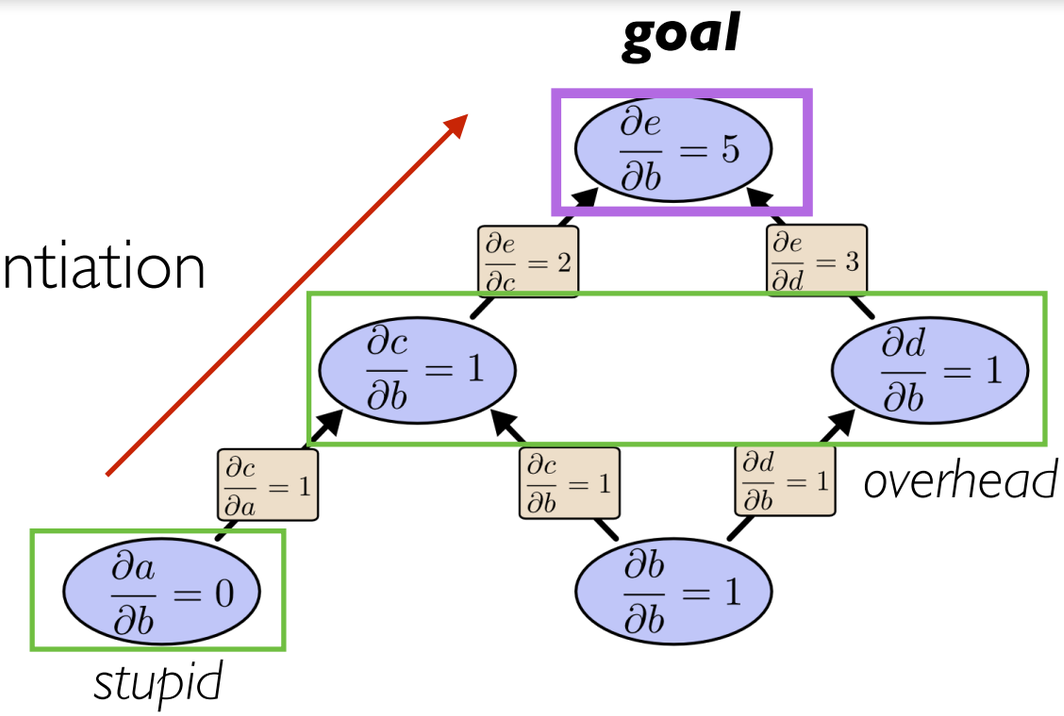
Reverse Differentiation

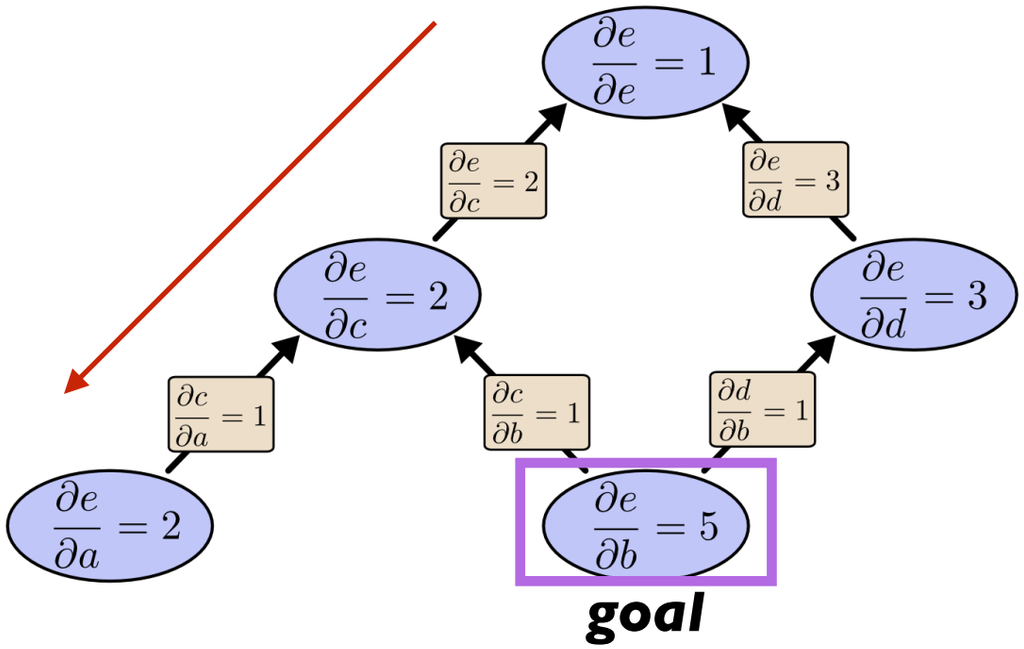
Graph Model of Computation

Goal: calculate $\frac{\partial e}{\partial b}$



Forward differentiation



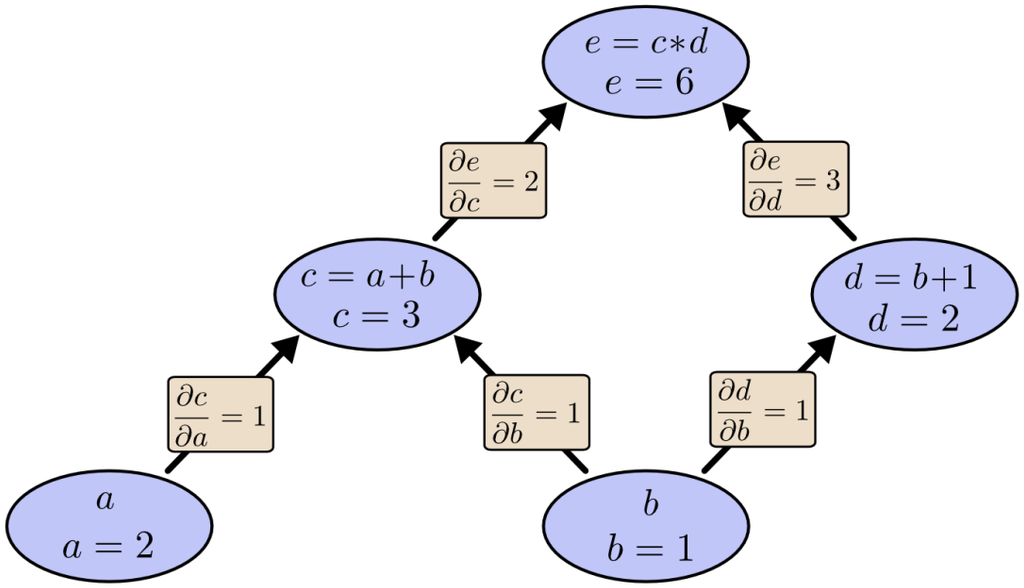


Reverse Differentiation

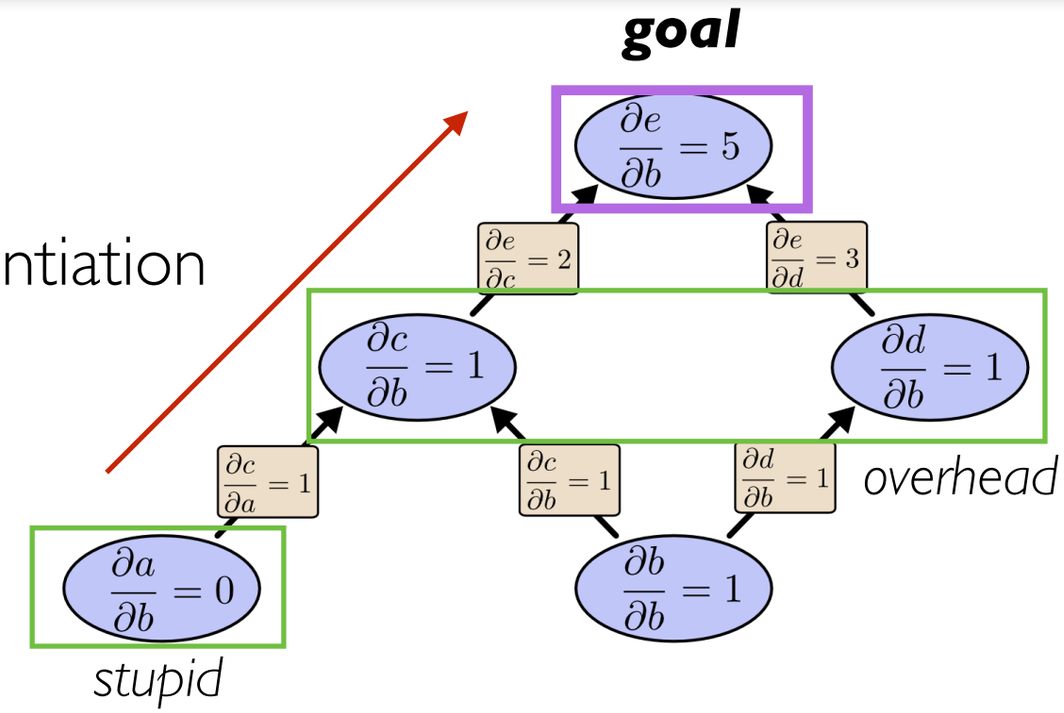
Graph Model of Computation

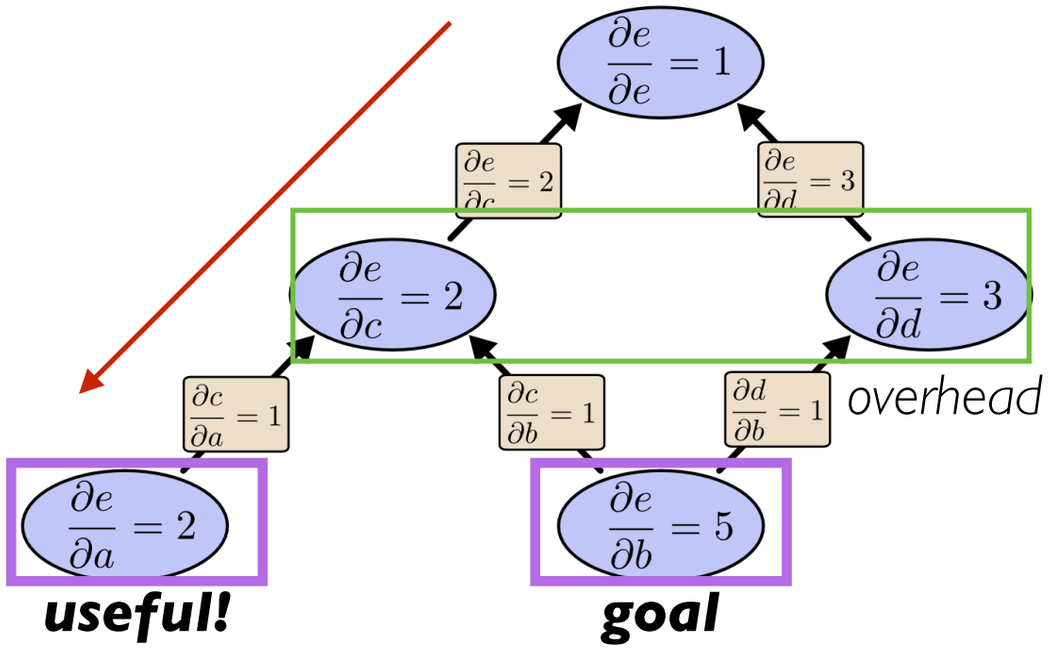
Goal: calculate $\frac{\partial e}{\partial b}$

Get: $\frac{\partial e}{\partial b}$ and $\frac{\partial e}{\partial a}$



Forward differentiation





Reverse Differentiation

Graph Model of Computation

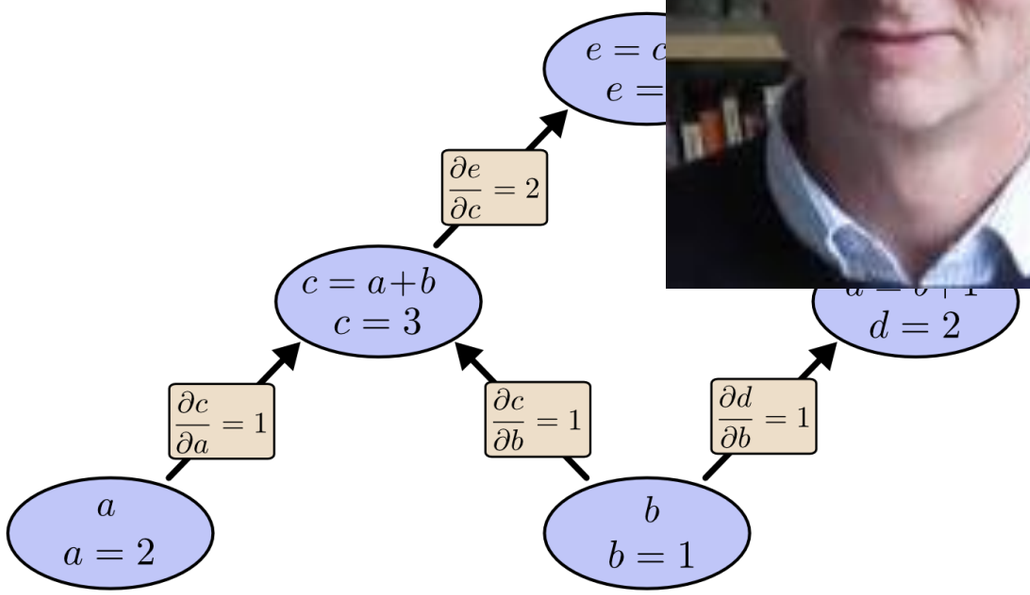
Goal: calculate $\frac{\partial e}{\partial b}$

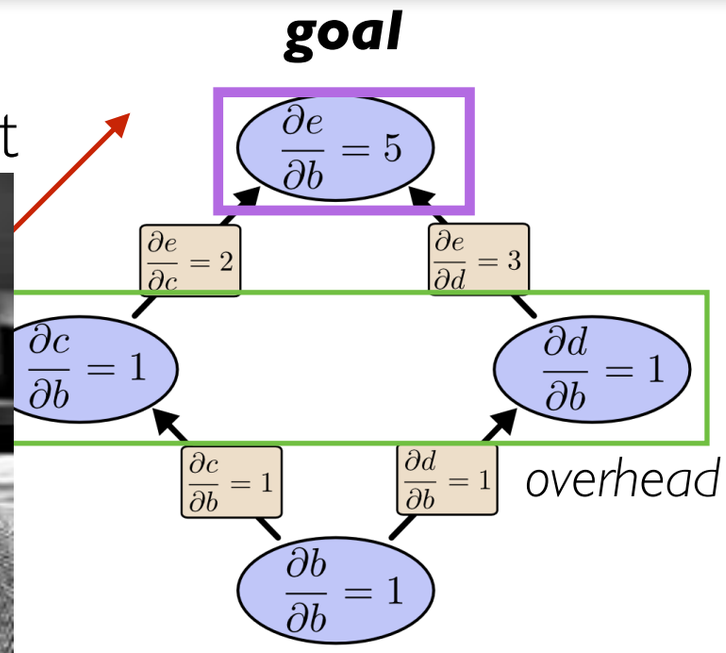
Get: $\frac{\partial e}{\partial b}$ and $\frac{\partial e}{\partial a}$

Geoff Hinton

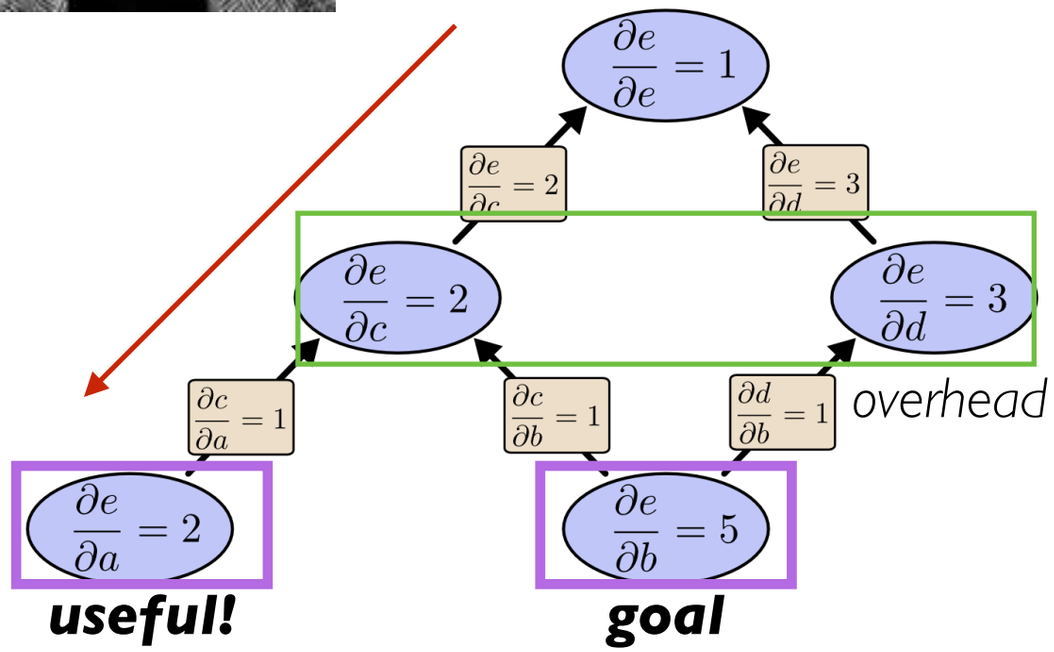
David Rumelhart

dif





The utility of *backpropagation*.

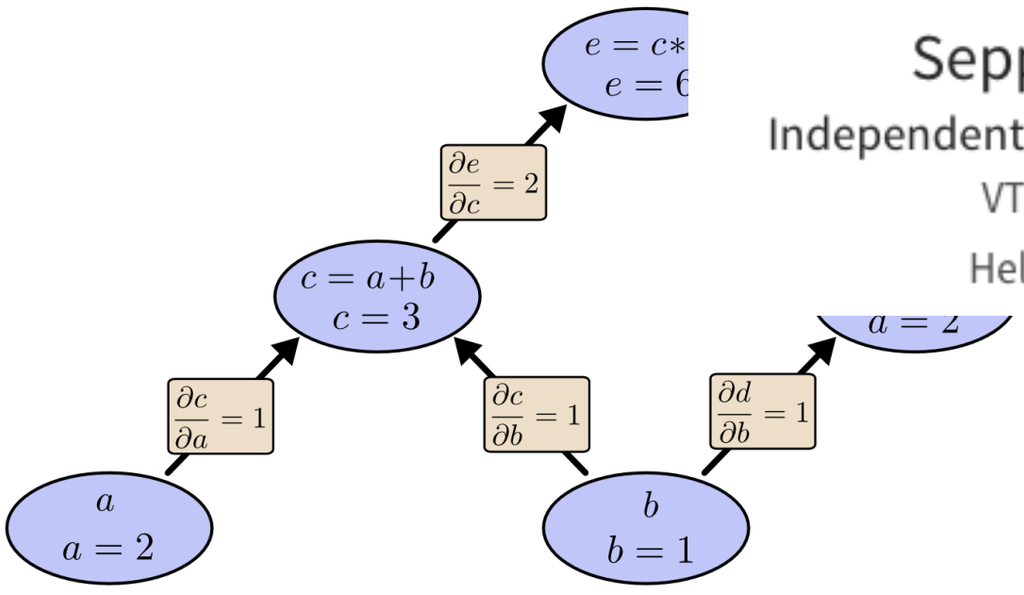


Reverse Differentiation

Graph Model of Computation

Goal: calculate $\frac{\partial e}{\partial b}$

Get: $\frac{\partial e}{\partial b}$ and $\frac{\partial e}{\partial a}$



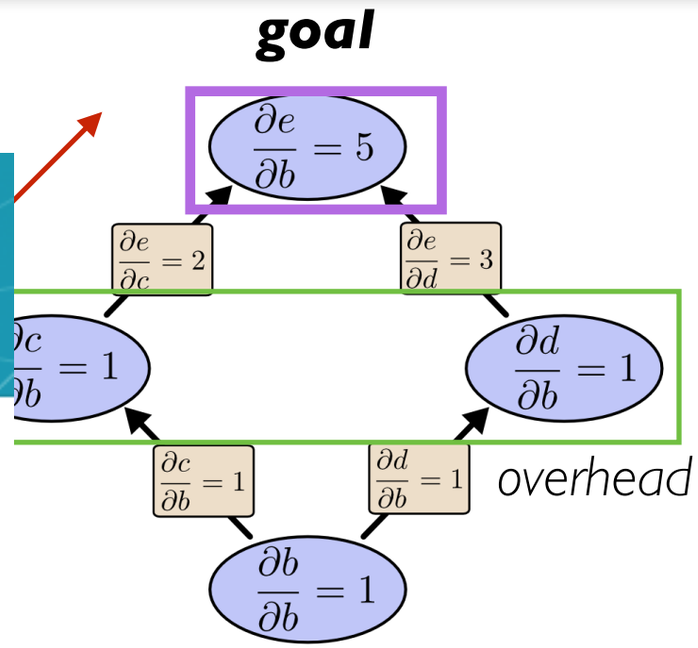
Reverse Mode AD in 1970 MA thesis

Seppo Linnainmaa • 3rd

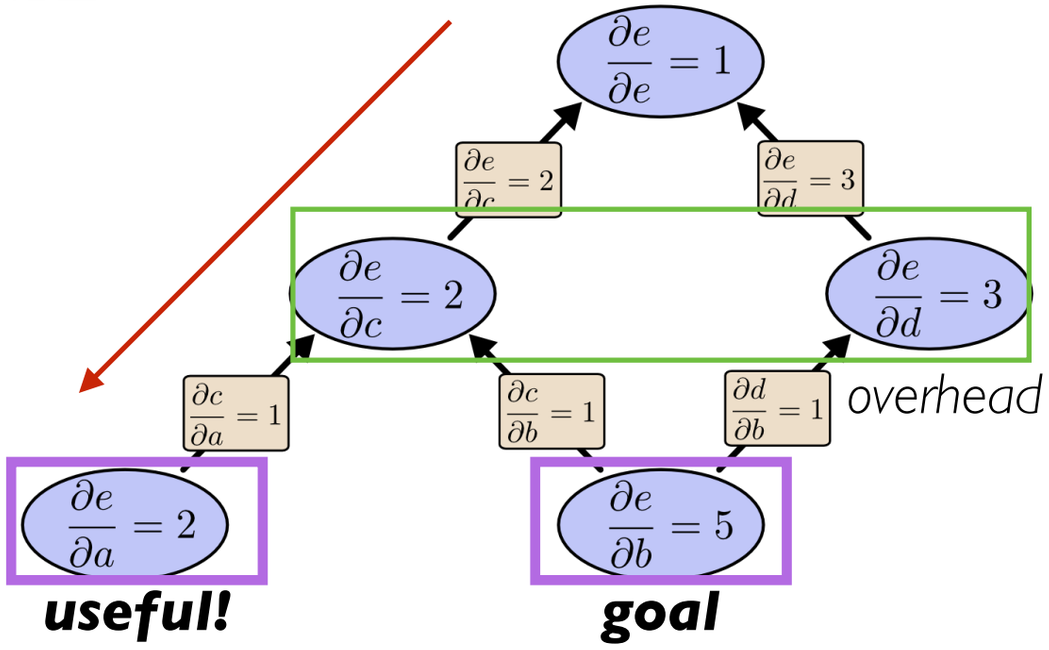
Independent Computer Software Professional

VTT • University of Helsinki

Helsinki Area, Finland • 72



The utility of *backpropagation.*



Reverse Differentiation

Graph Model of Computation

tf.Graph

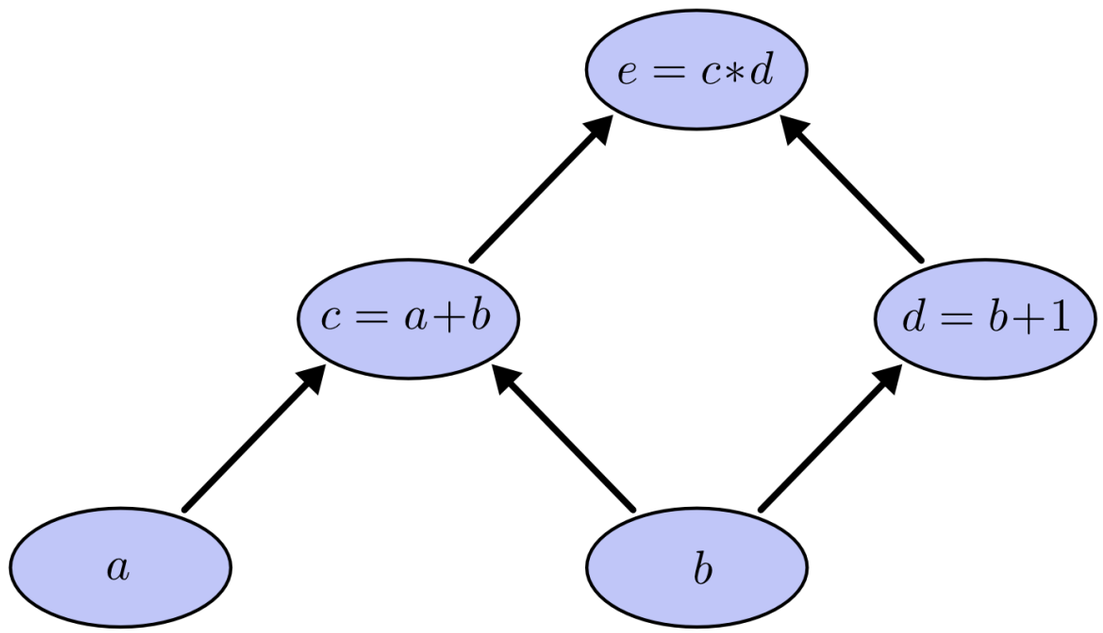
Class **Graph**

Defined in `tensorflow/python/framework/ops.py`.

See the guide: Building Graphs > Core graph data structures

A TensorFlow computation, represented as a dataflow graph.

A `Graph` contains a set of `tf.Operation` objects, which represent units of computation; and `tf.Tensor` objects, which represent the units of data that flow between operations.



Graph Model of Computation

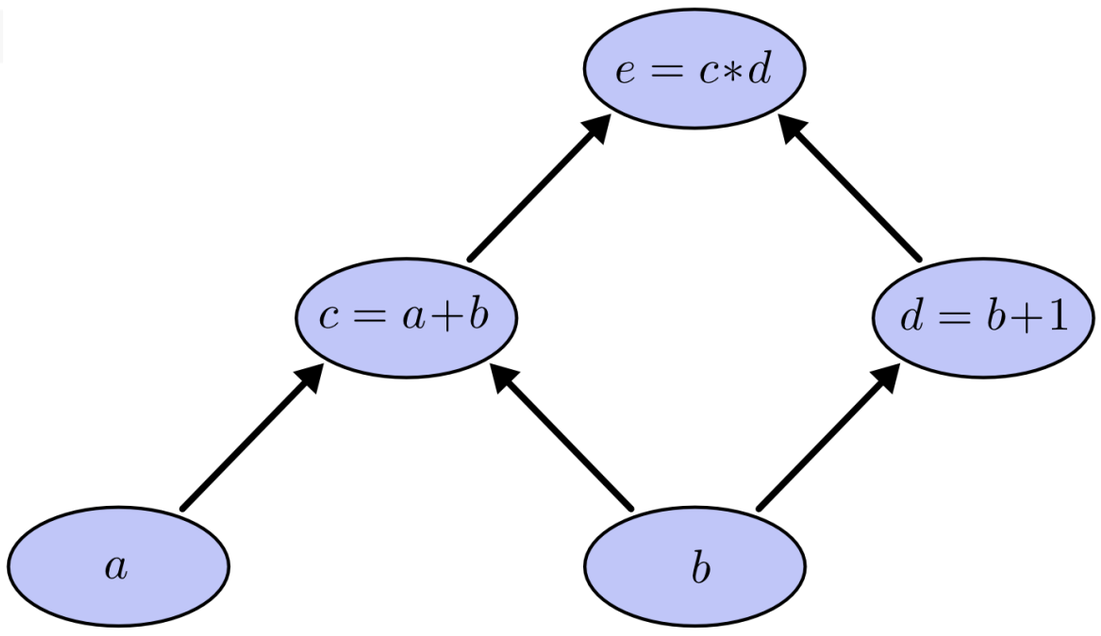
tf.gradients

```
gradients(
    ys,
    xs,
    grad_ys=None,
    name='gradients',
    colocate_gradients_with_ops=False,
    gate_gradients=False,
    aggregation_method=None
)
```

Defined in `tensorflow/python/ops/gradients_impl.py`.

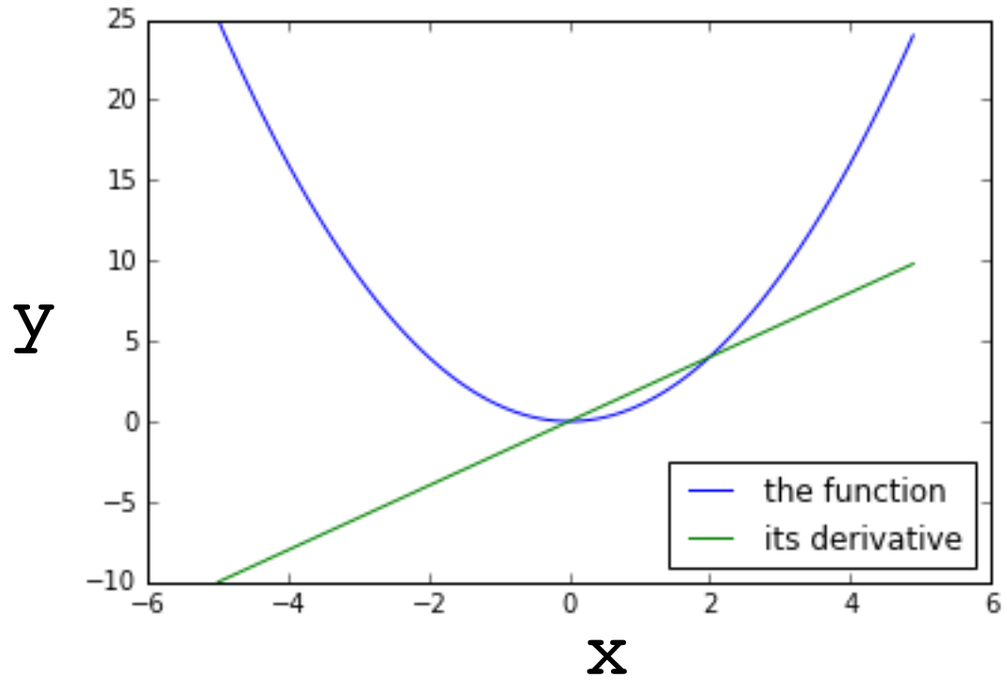
See the guide: Training > Gradient Computation

Constructs symbolic partial derivatives of sum of `ys` w.r.t. x in `xs`



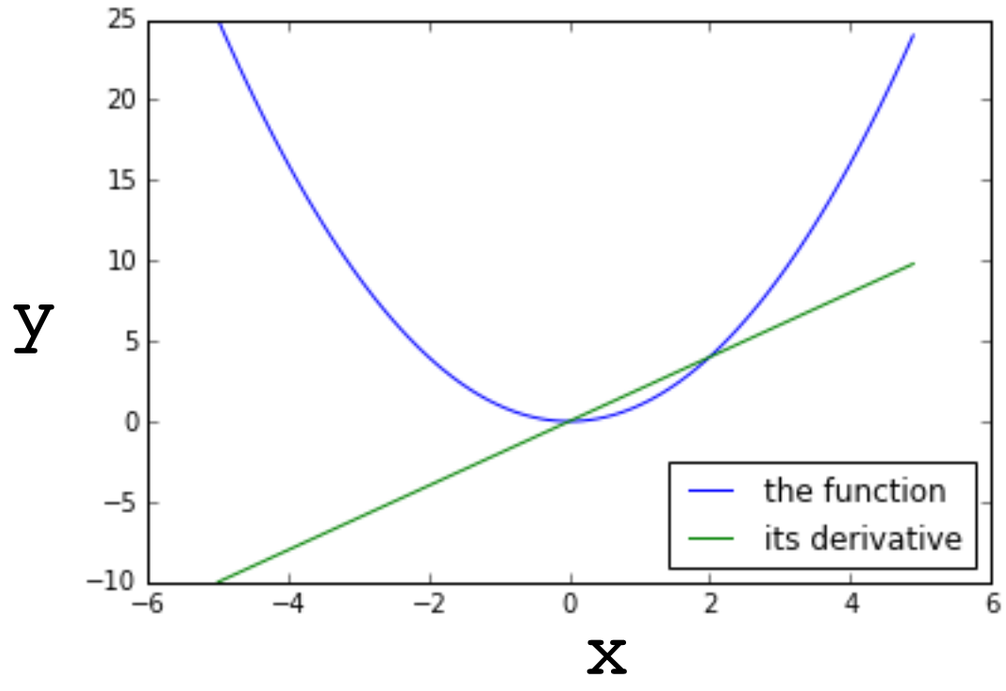
Derivatives

```
y = x**2

d = tf.gradients(y, x)
```

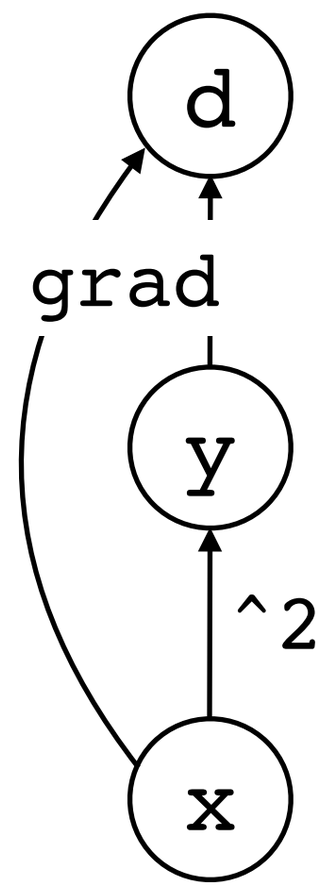


Derivatives

```
y = x**2

d = tf.gradients(y, x)
```

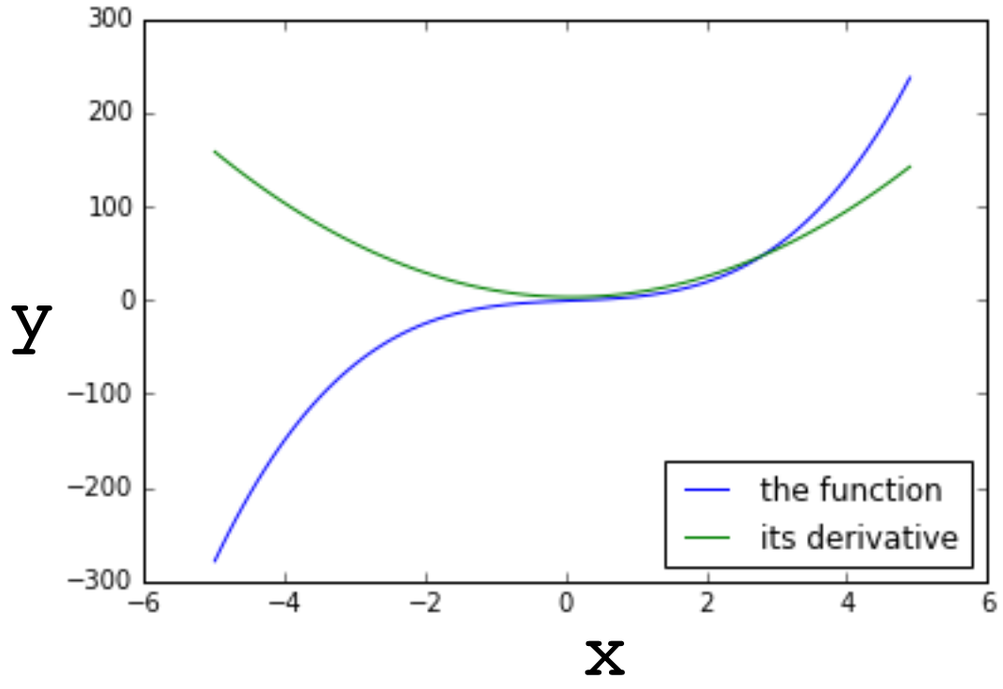


computation graph:



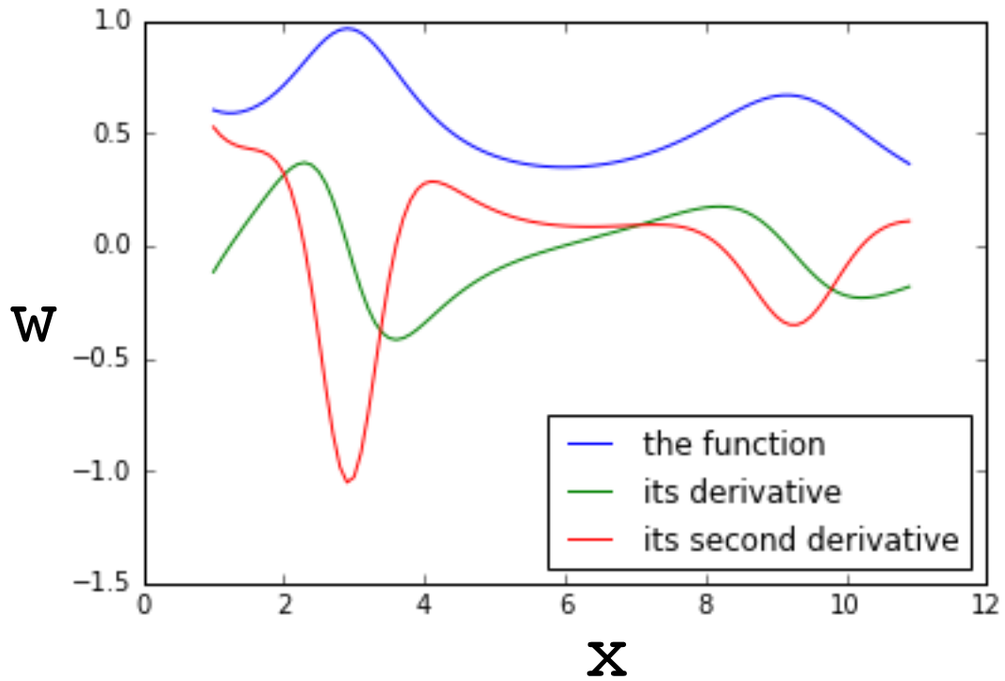
Derivatives

```
y = 2*x**3 - 5*x**2 + 3*x - 1
```



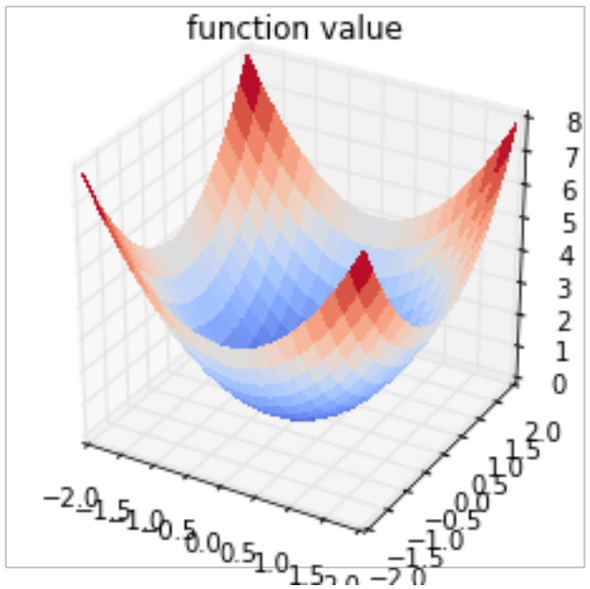
Derivatives

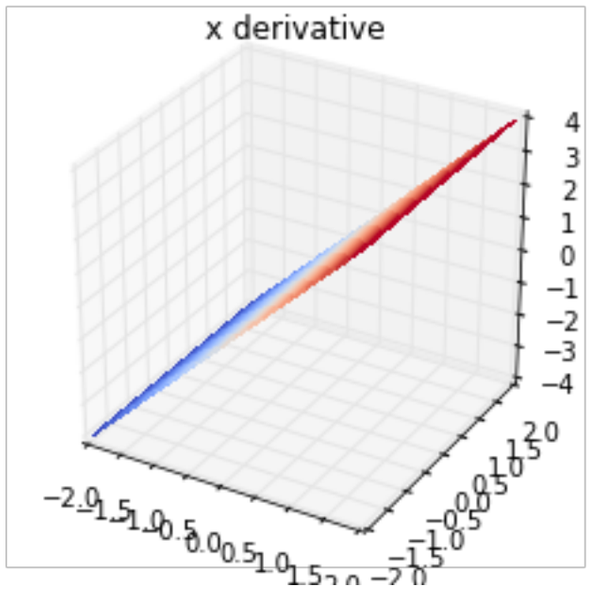
```
y = x**.5
z = tf.log(tf.exp(tf.sin(y) + tf.cos(2*y)) + 1)
w = z / (y + tf.cos(x_arr))
```

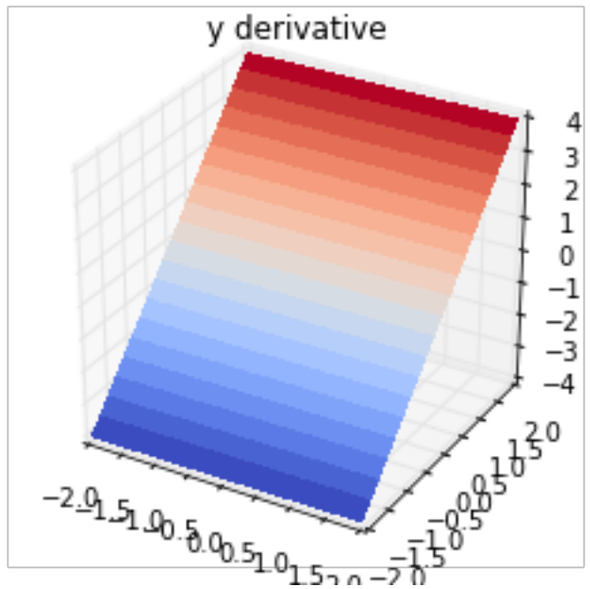


Derivatives

```
z = x**2 + y**2
```

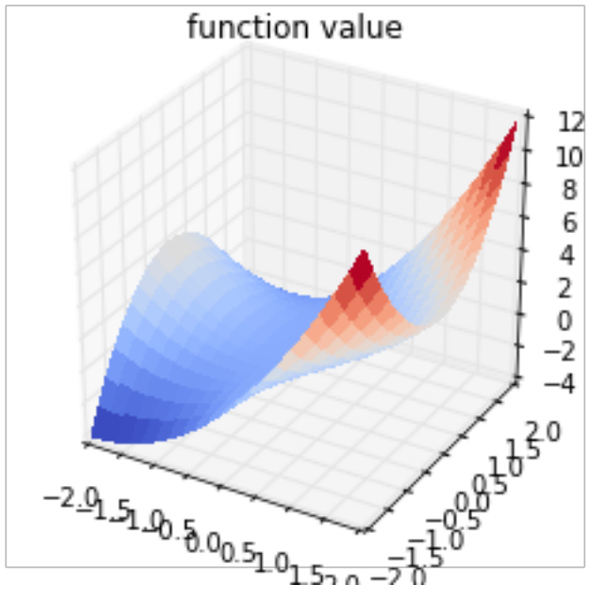


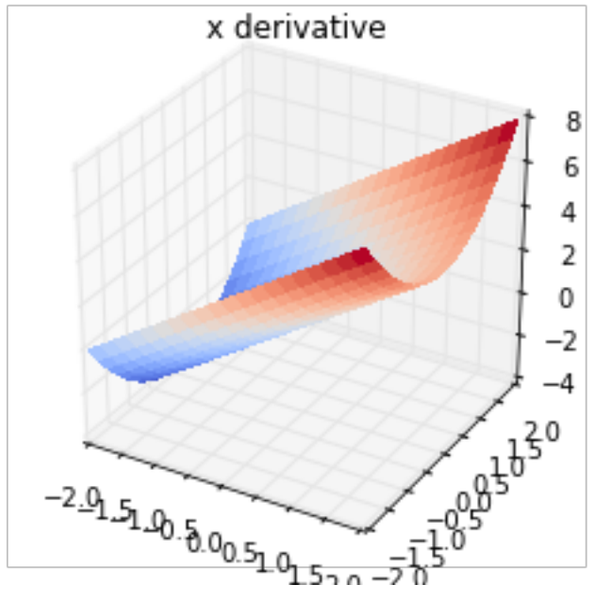


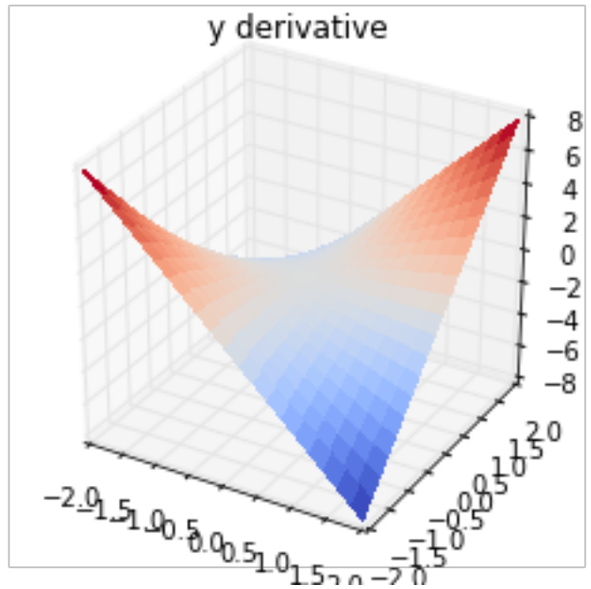


Derivatives

```
z = x**2 + y**2 * x
```







Derivatives

```
z = tf.cosh((3 + tf.cos(x+2*y)**3)**.5 * tf.sin(y-x))
```