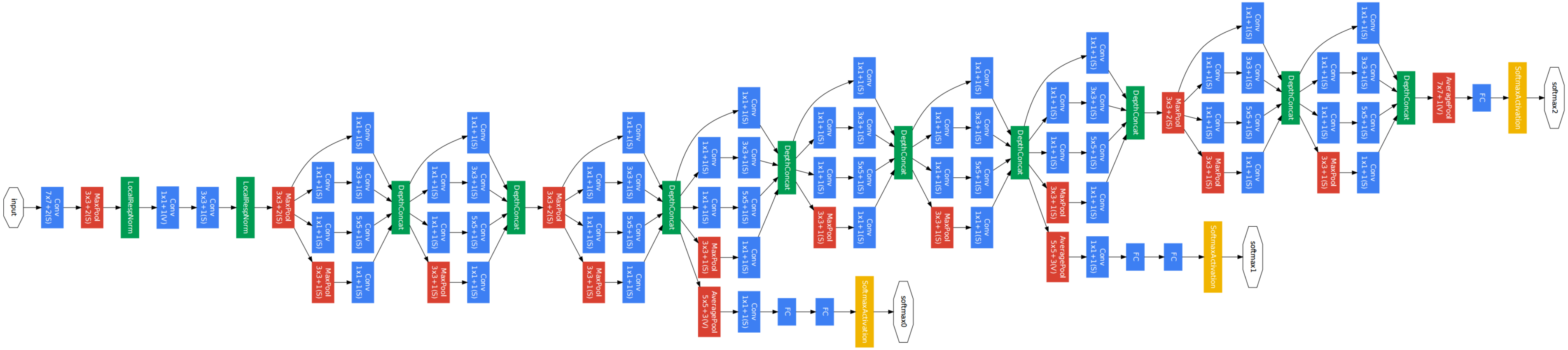
Theano & Tensorflow

Symbolic Compute Graphs

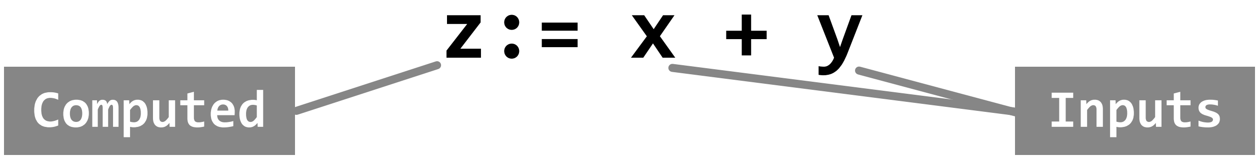
GoogLeNet


input
Conv 7x7+2(S)
MaxPool 3x3+2(S)
LocalRespNorm
Conv 1x1+1(V)
Conv 3x3+1(S)
LocalRespNorm
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
AveragePool 5x5+3(V)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
FC
FC
SoftmaxActivation
softmax0
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
AveragePool 5x5+3(V)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
FC
FC
SoftmaxActivation
softmax1
DepthConcat
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
Conv 1x1+1(S)
Conv 1x1+1(S)
Conv 1x1+1(S)
MaxPool 3x3+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
Conv 1x1+1(S)
DepthConcat
AveragePool 7x7+1(V)
FC
SoftmaxActivation
softmax2

Compute Graphs

- Deep networks are static
- Math can be represented as a graph

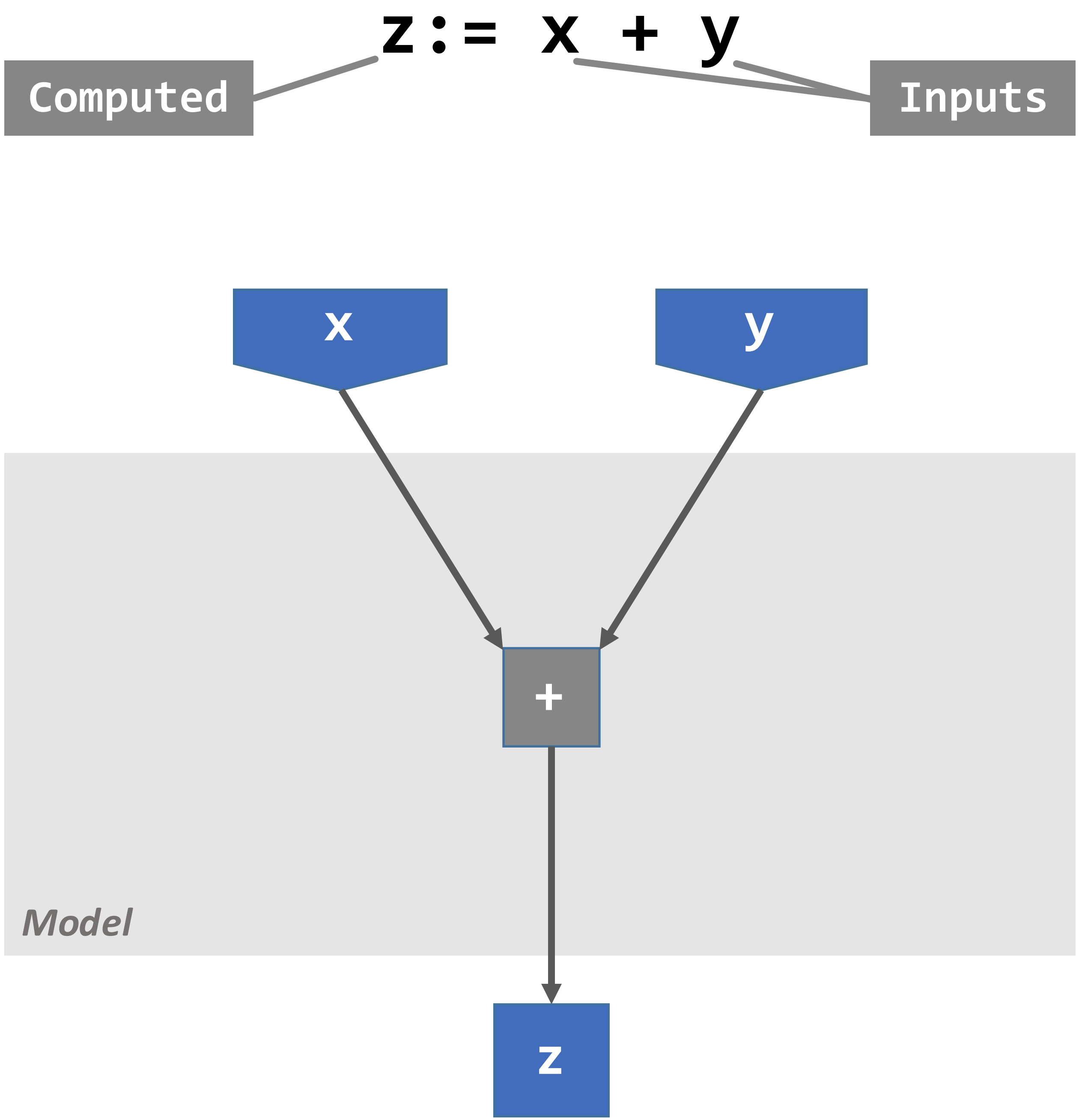
```
z:= x + y
```



z:= x + y

Computed

Inputs

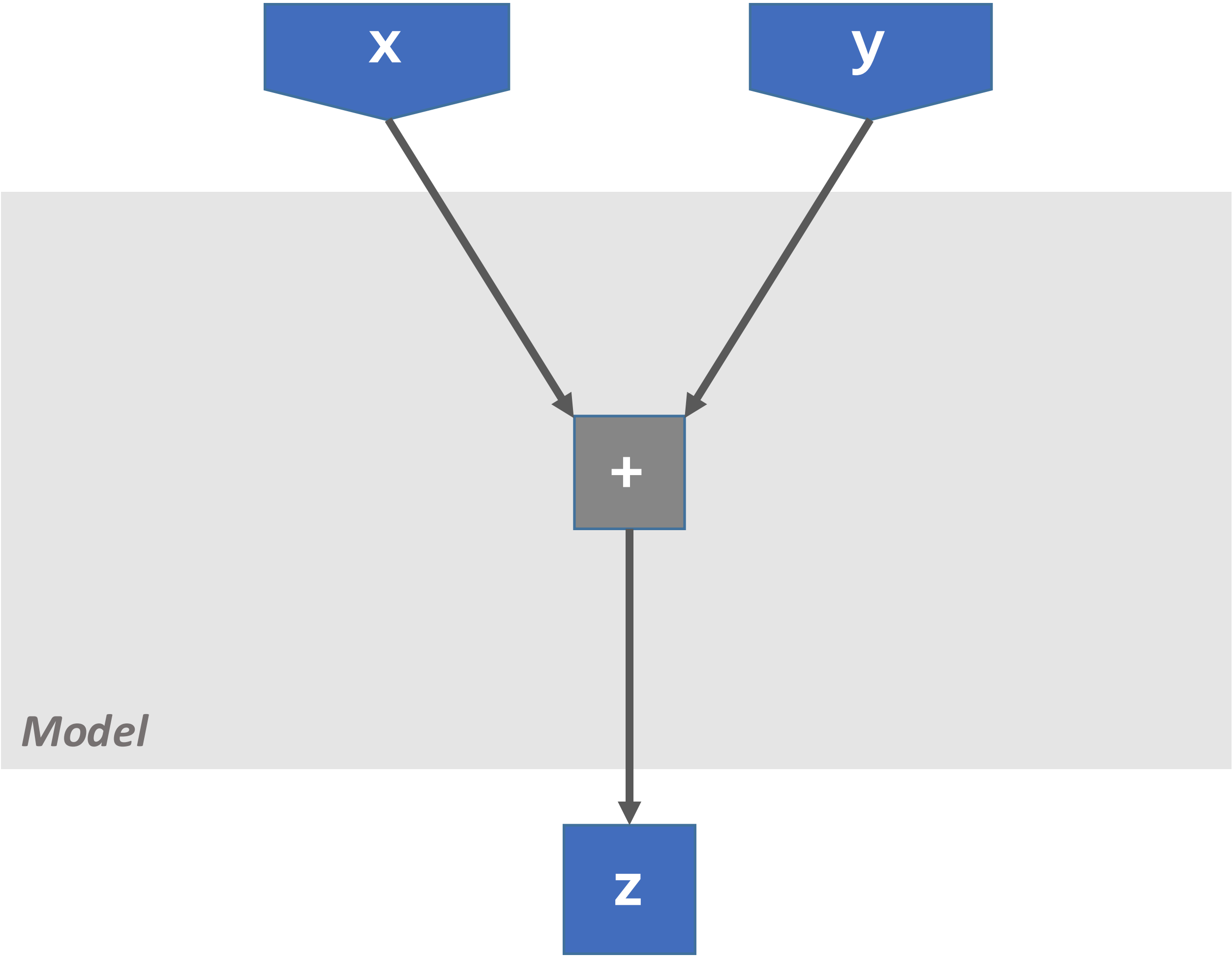


```
z:= x + y
```

Computed

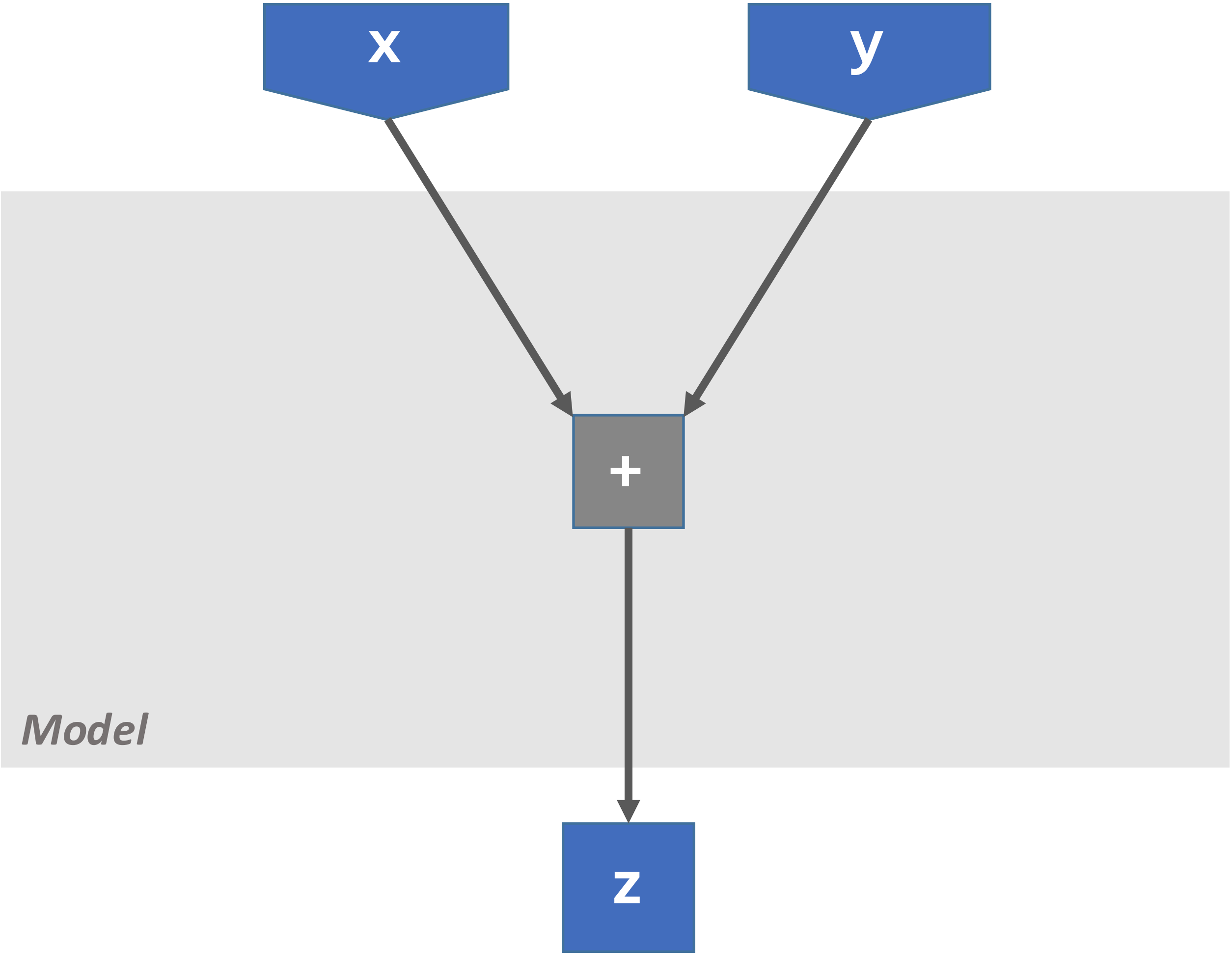
Inputs

```
z := a*x + b*y + c
```



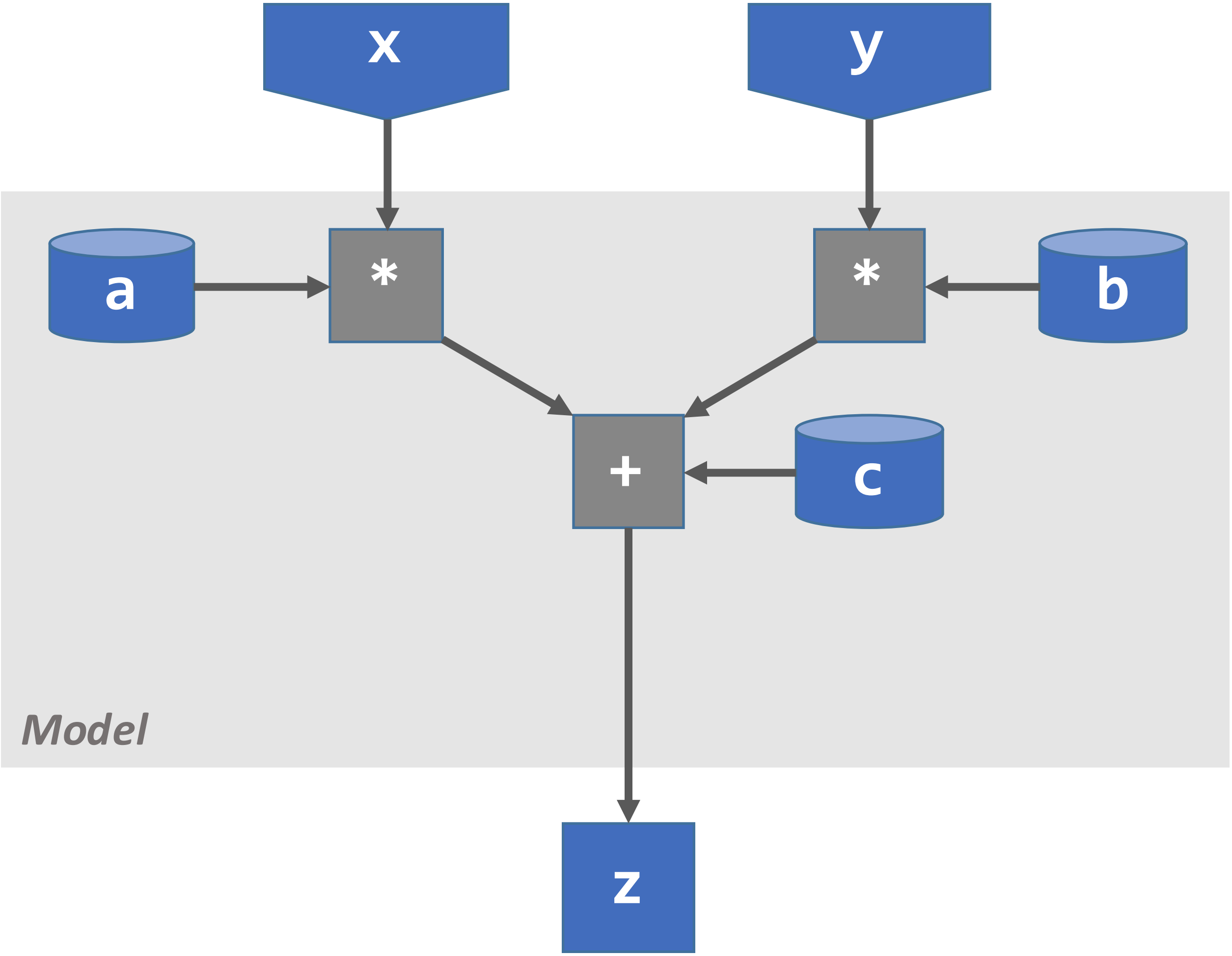
```
z:= a*x + b*y + c
```

Parameters

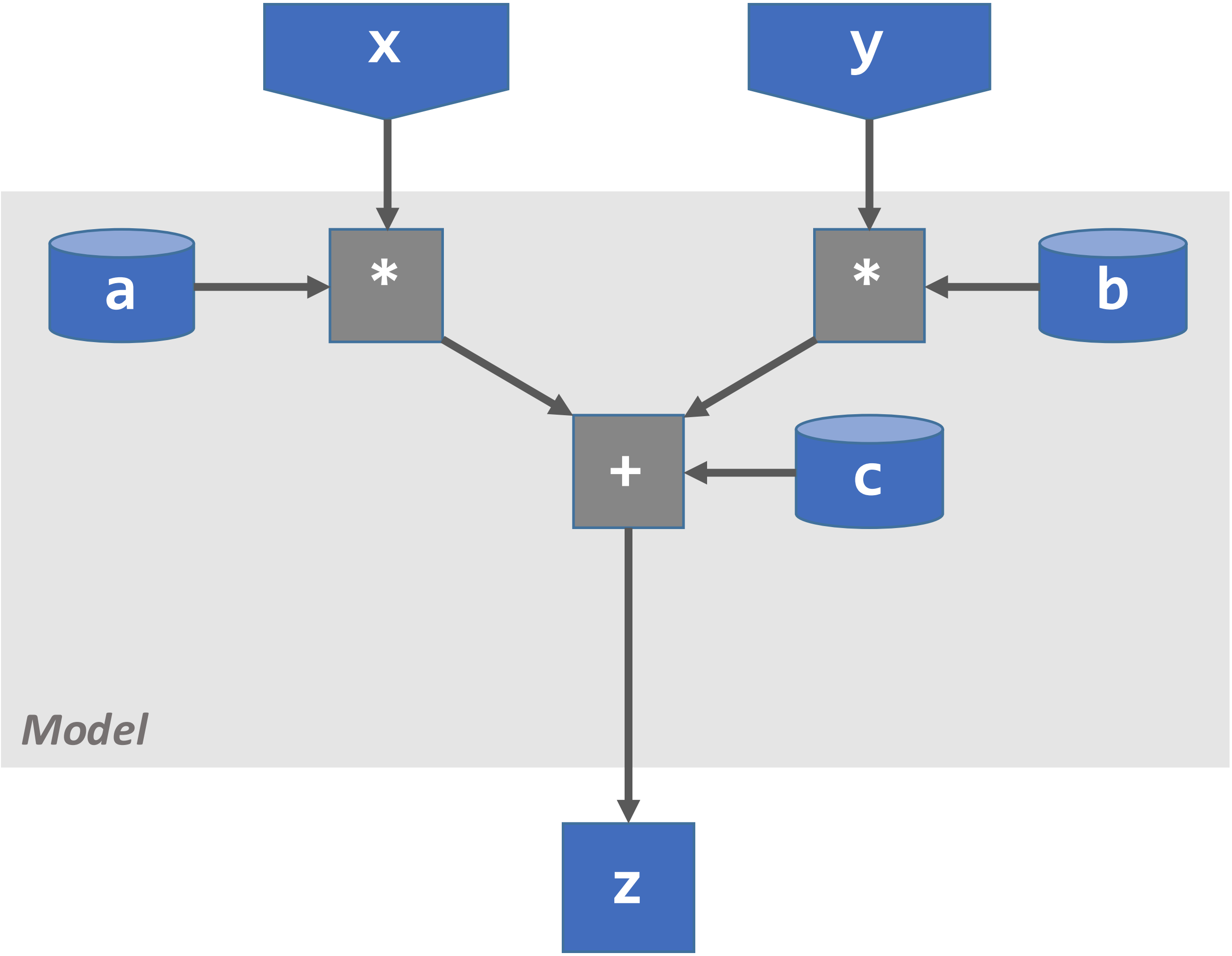


```
z:= a*x + b*y + c
```

Parameters

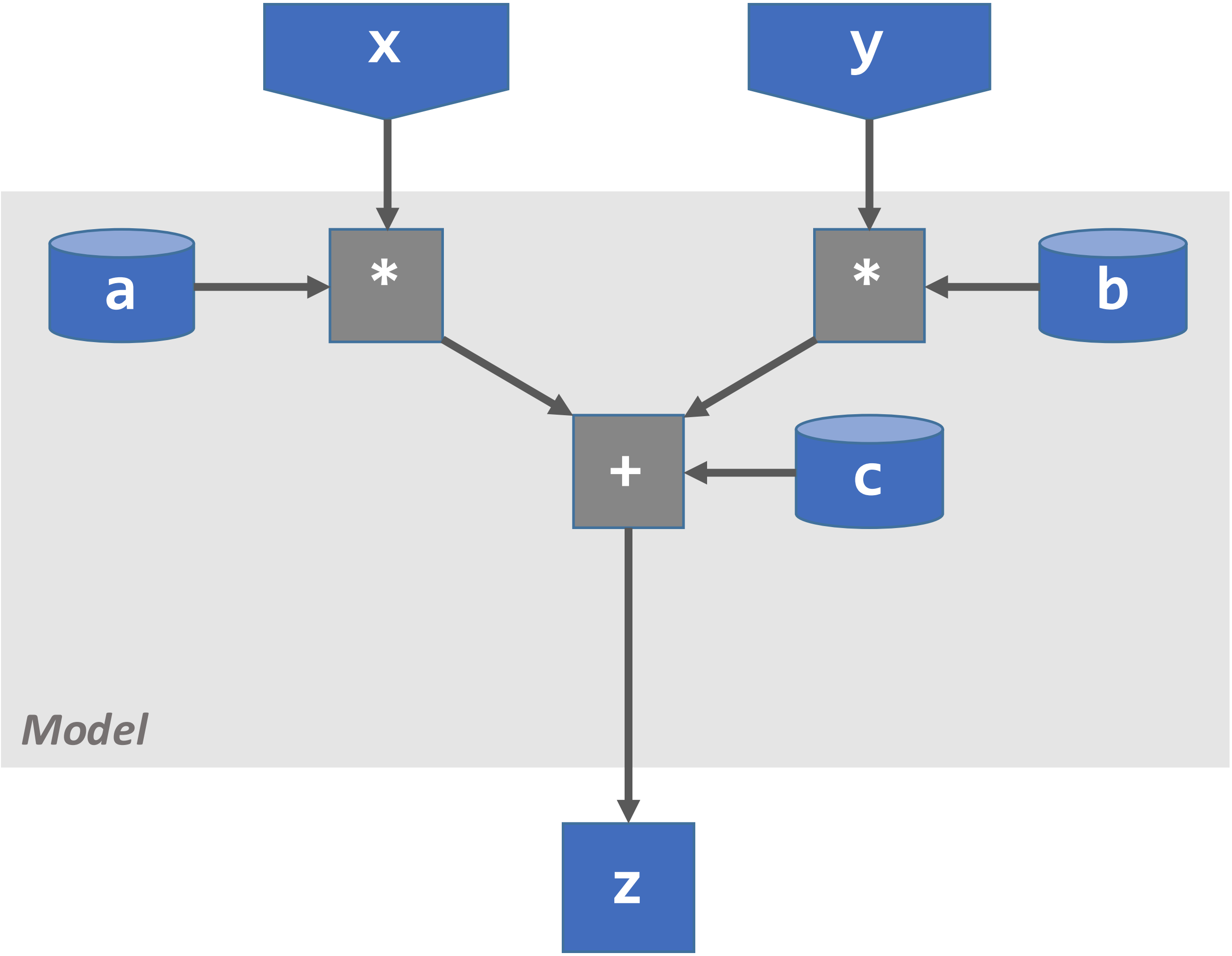


```
z:= Exp(a*x + b*y + c)
```



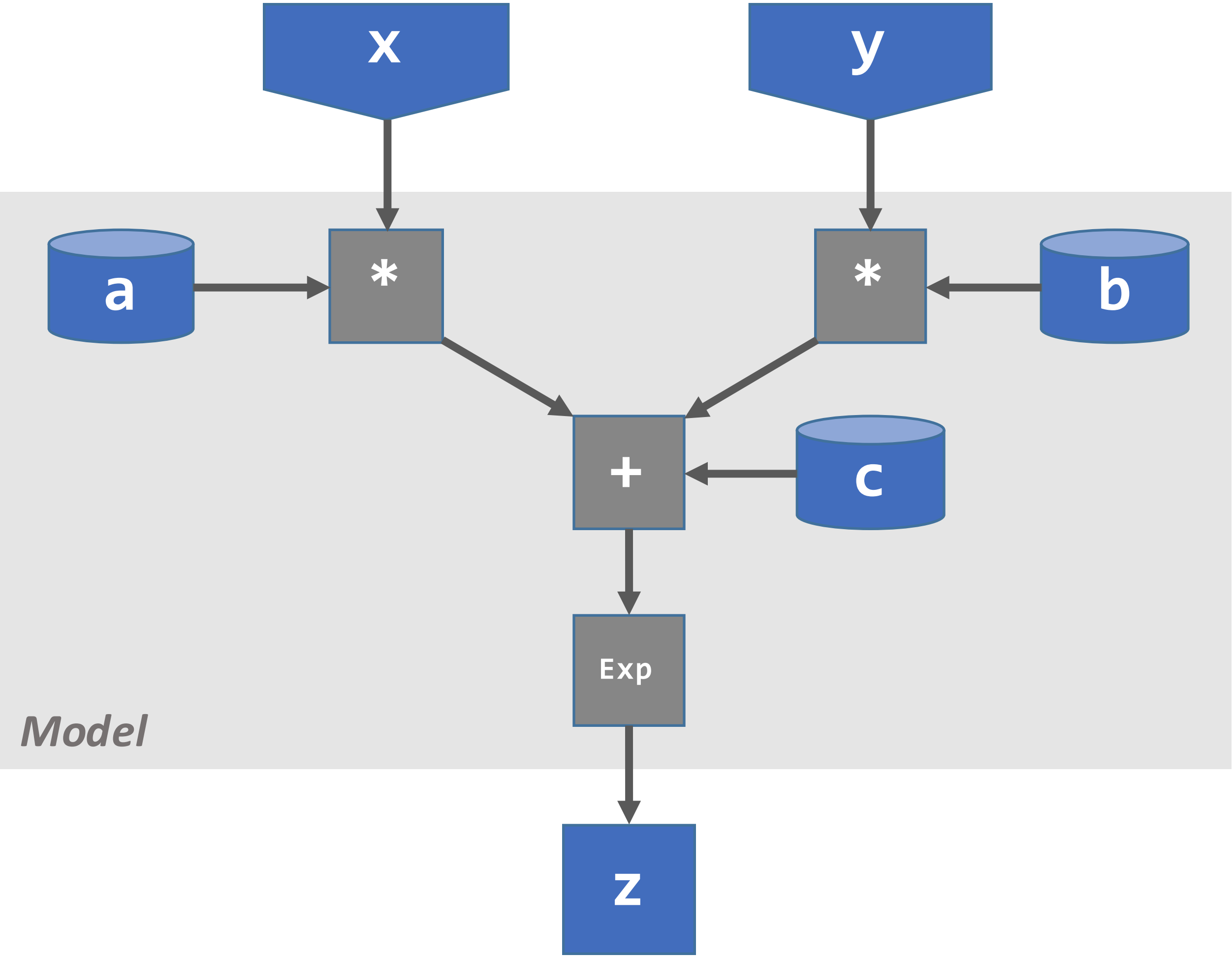
```
z:= Exp(a*x + b*y + c)
```

Function

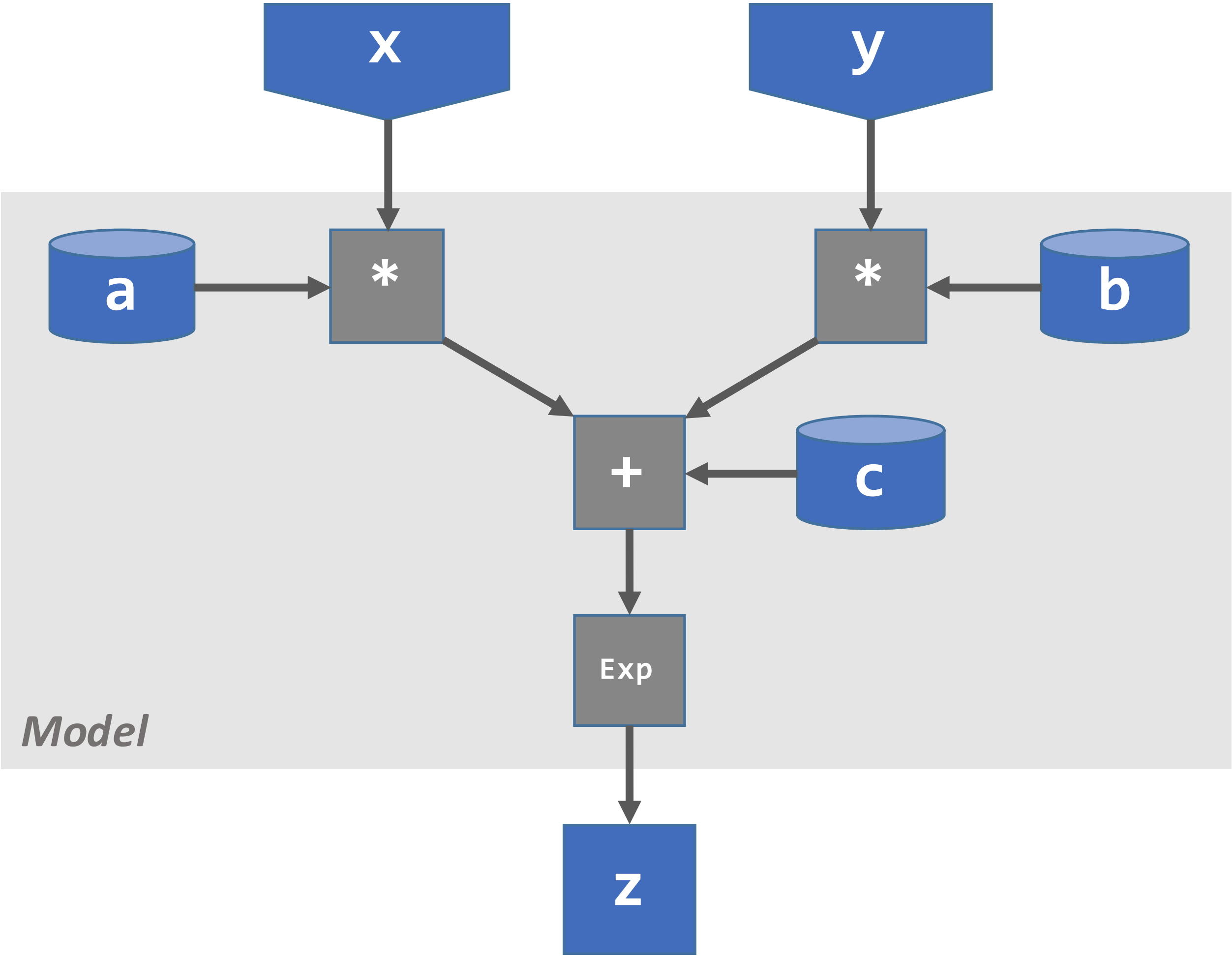


```
z:= Exp(a*x + b*y + c)
```

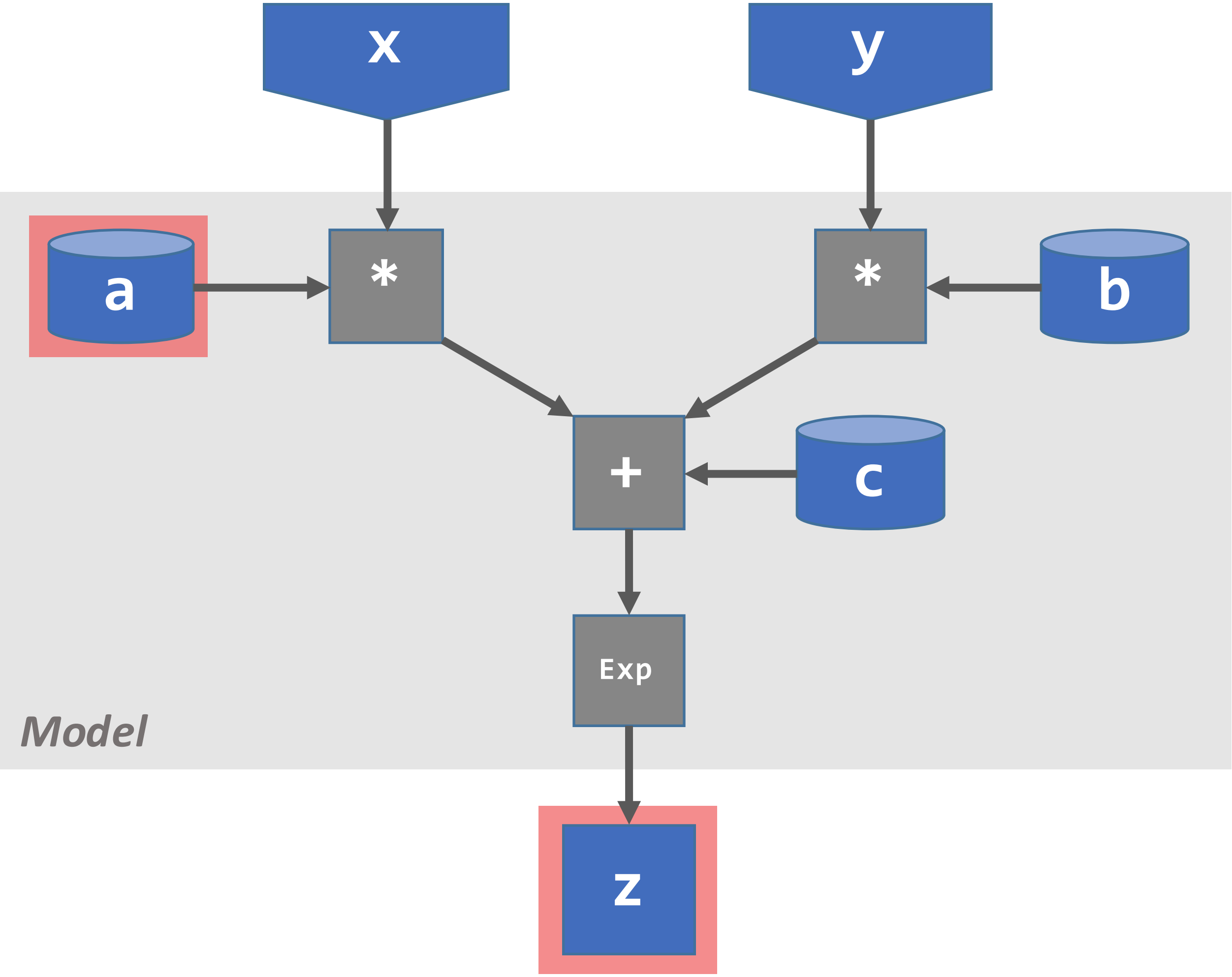
Function



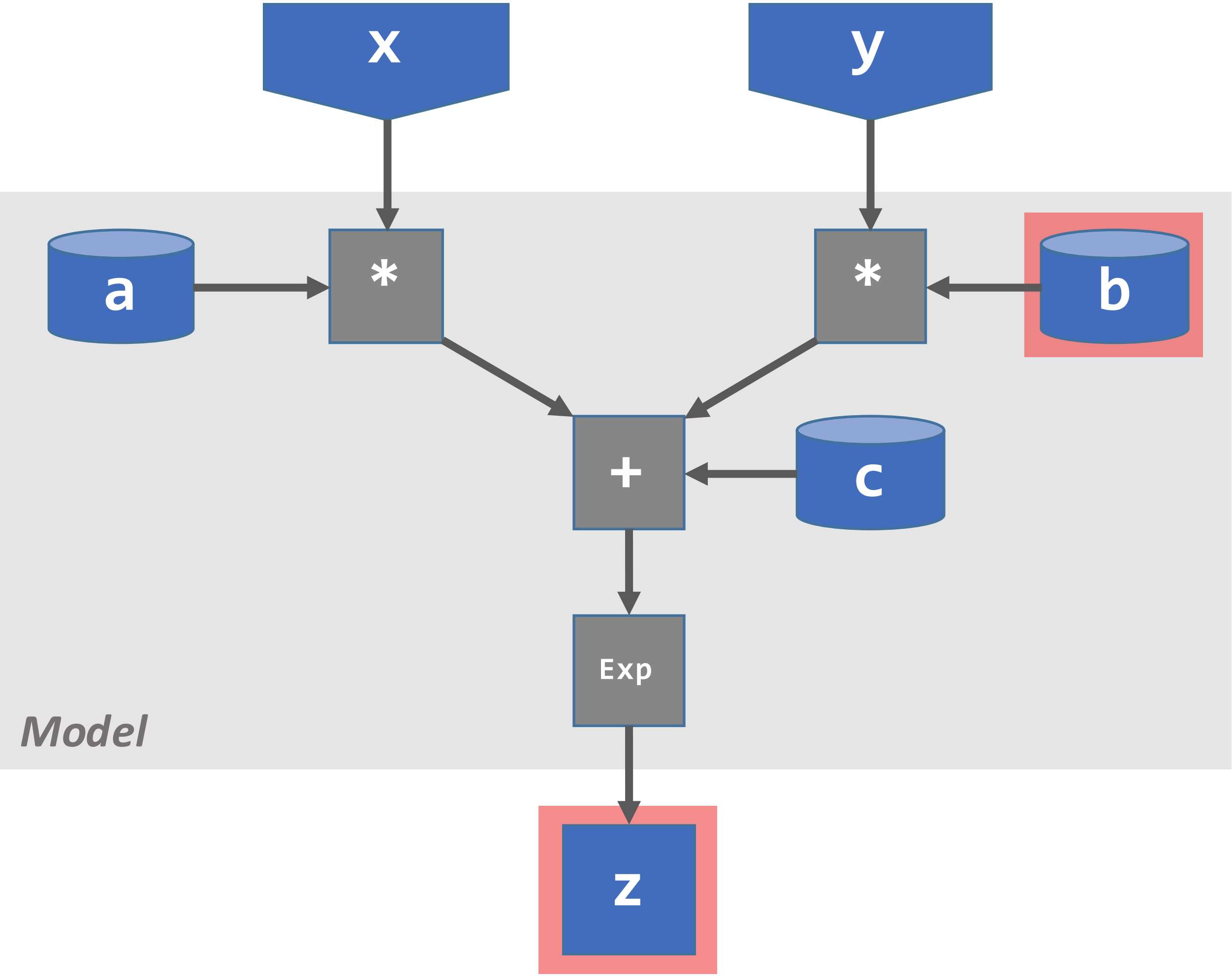
What is ∂z/∂a ?



What is ∂z/∂a ?

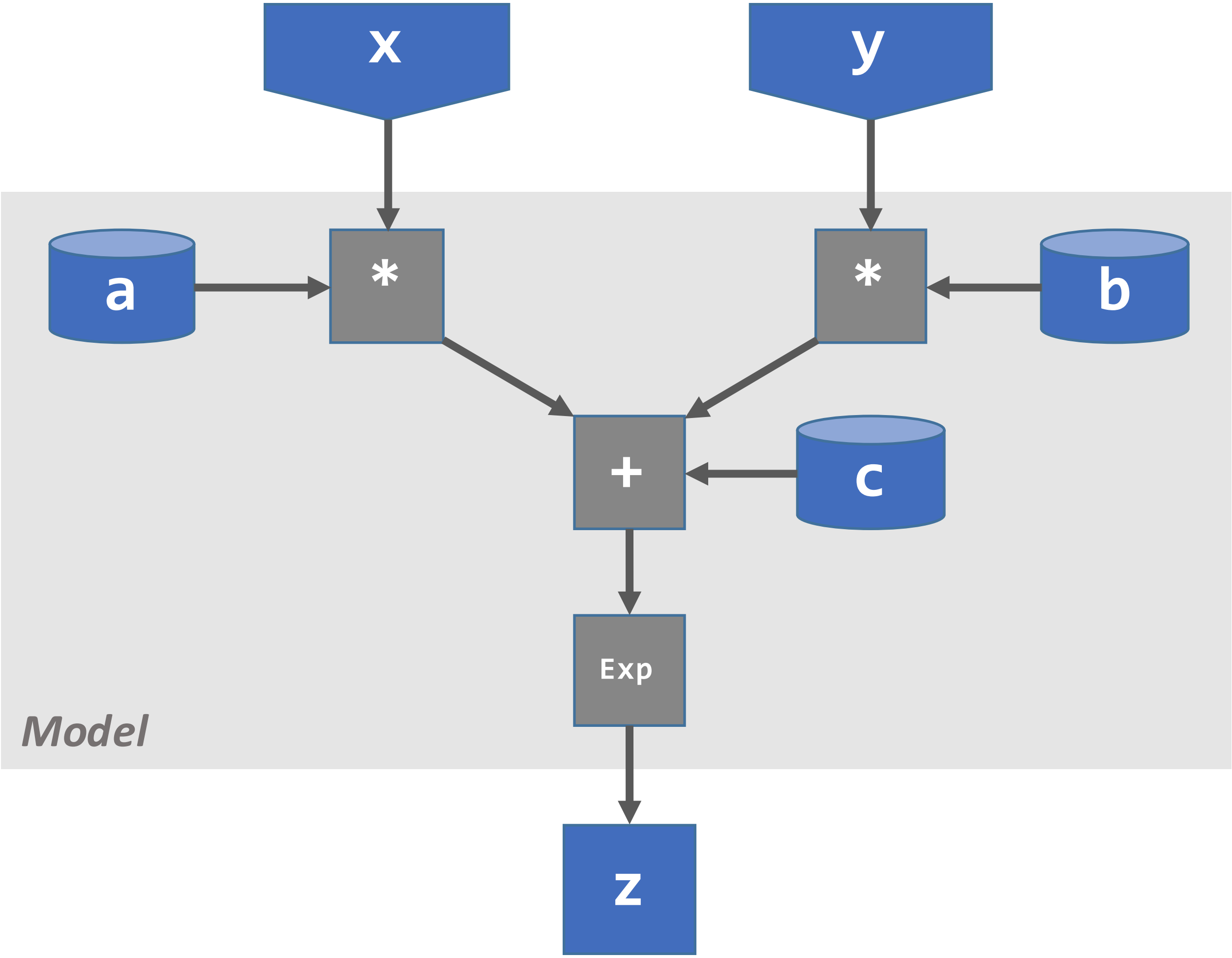


What is ∂z/∂a ?   ∂z/∂y ?

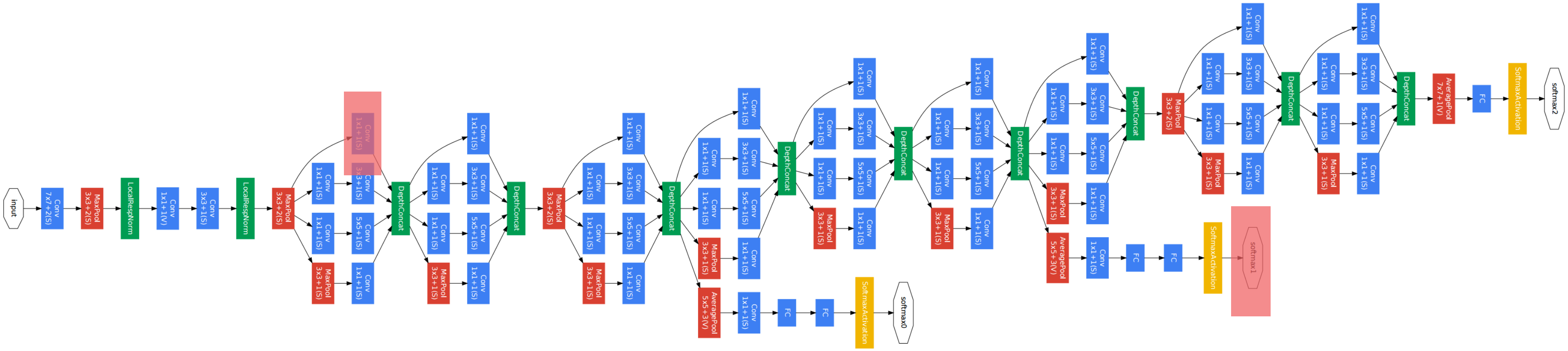


What is $\partial z/\partial a$? $\partial z/\partial y$?

How about $\partial \boxed{+}/\partial b$?

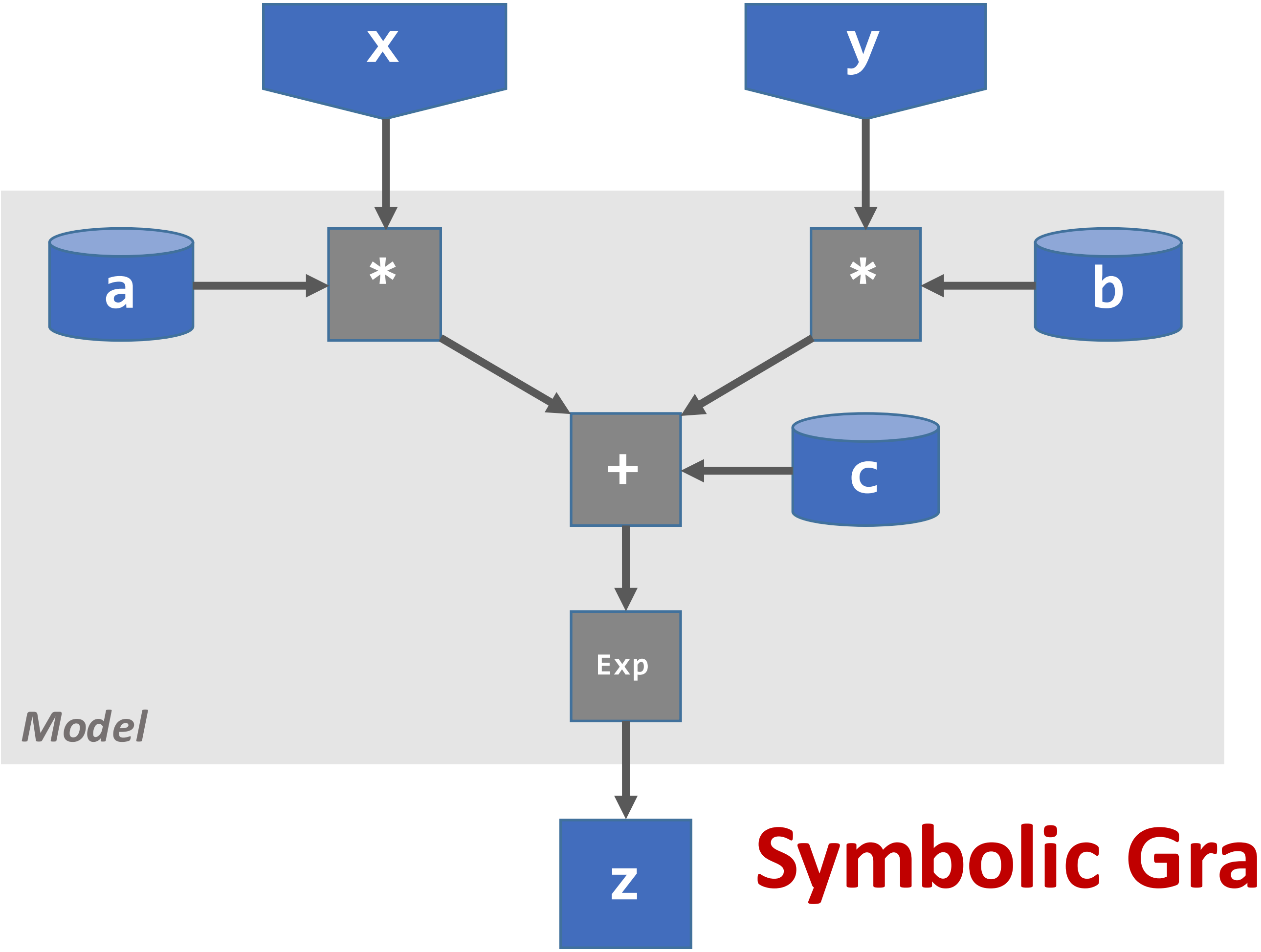


How about here?



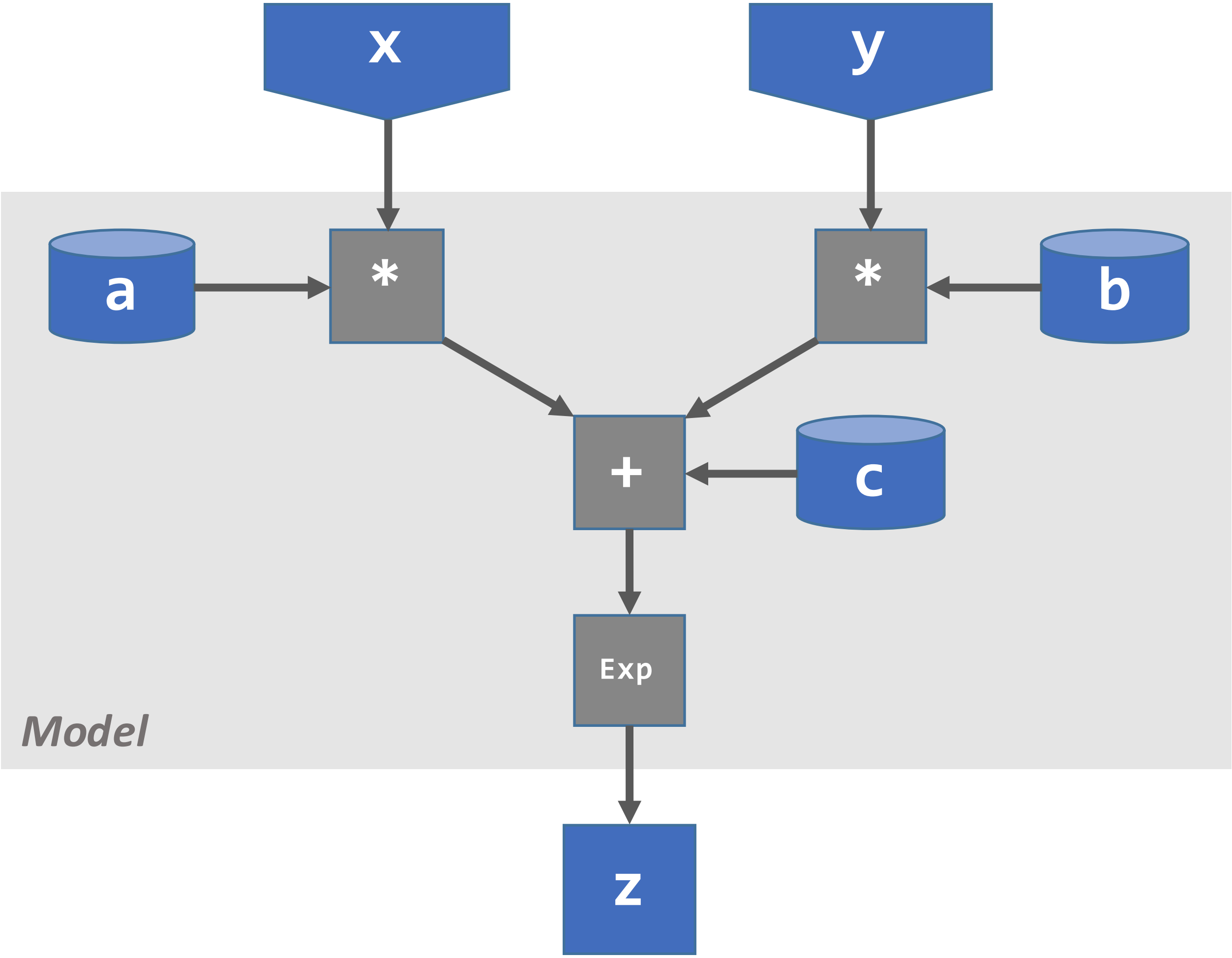
What is $\partial z/\partial a$? $\partial z/\partial y$?

How about ∂+$/\partial b$?

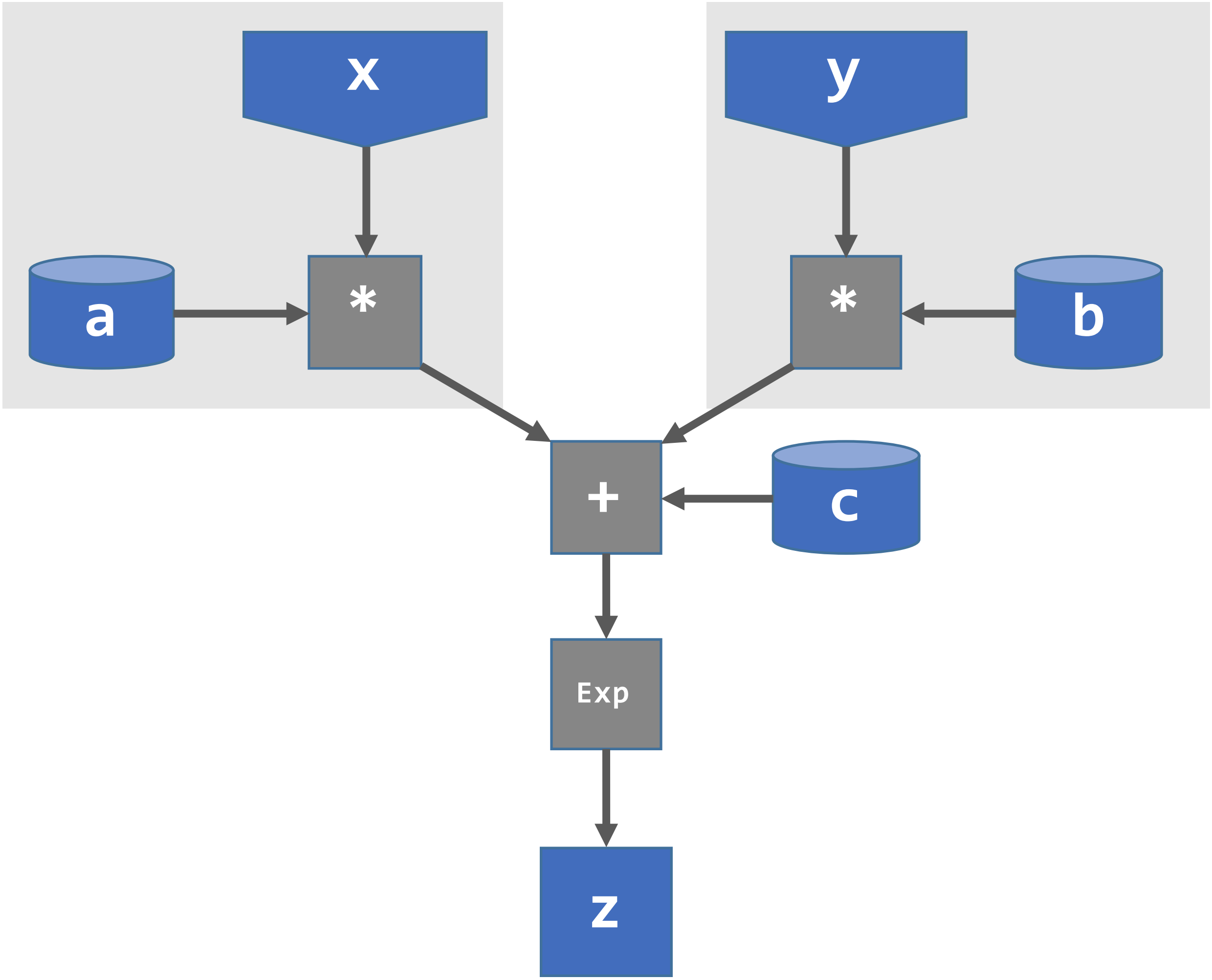


Symbolic Gradients!

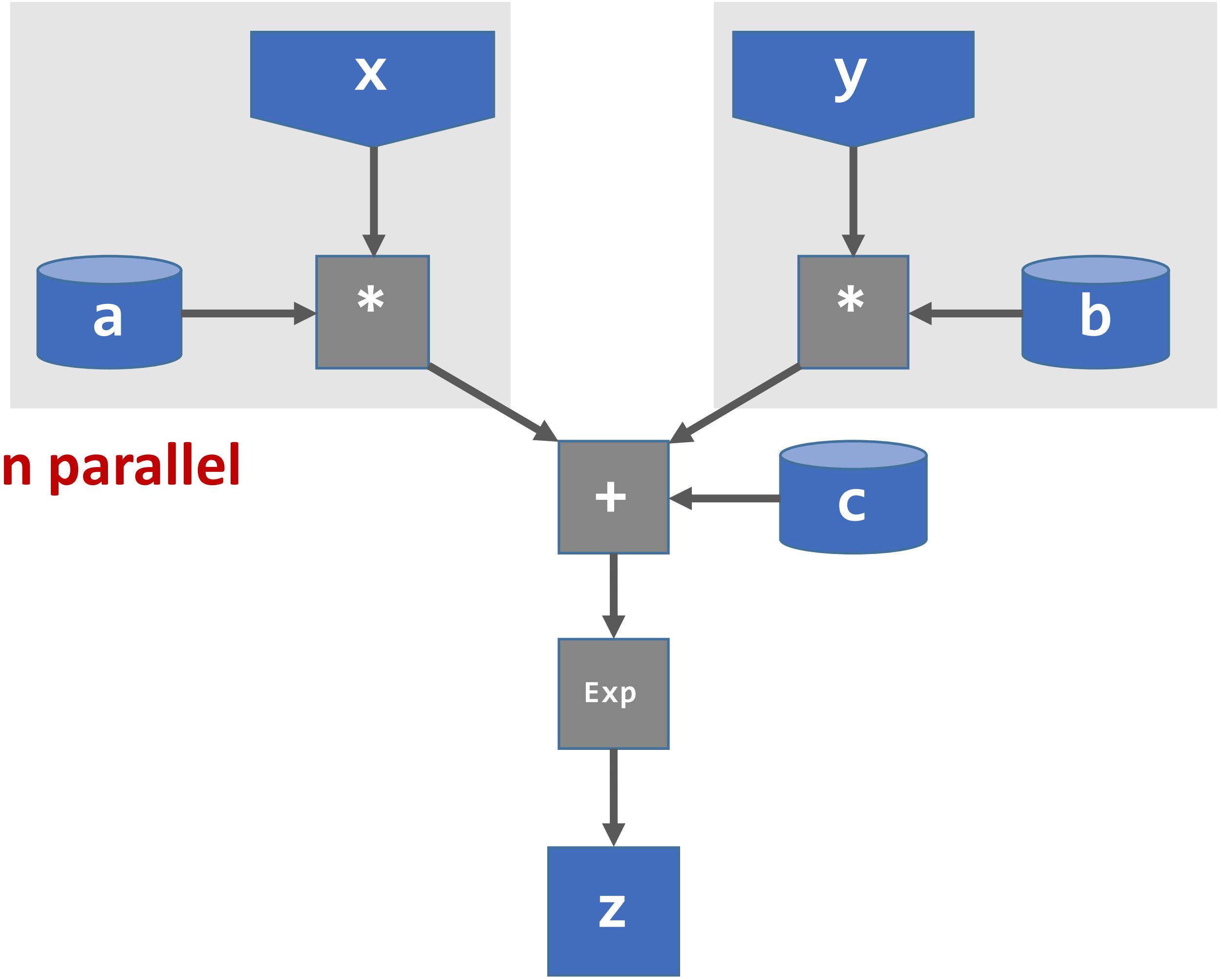
What about optimization?



What about optimization?

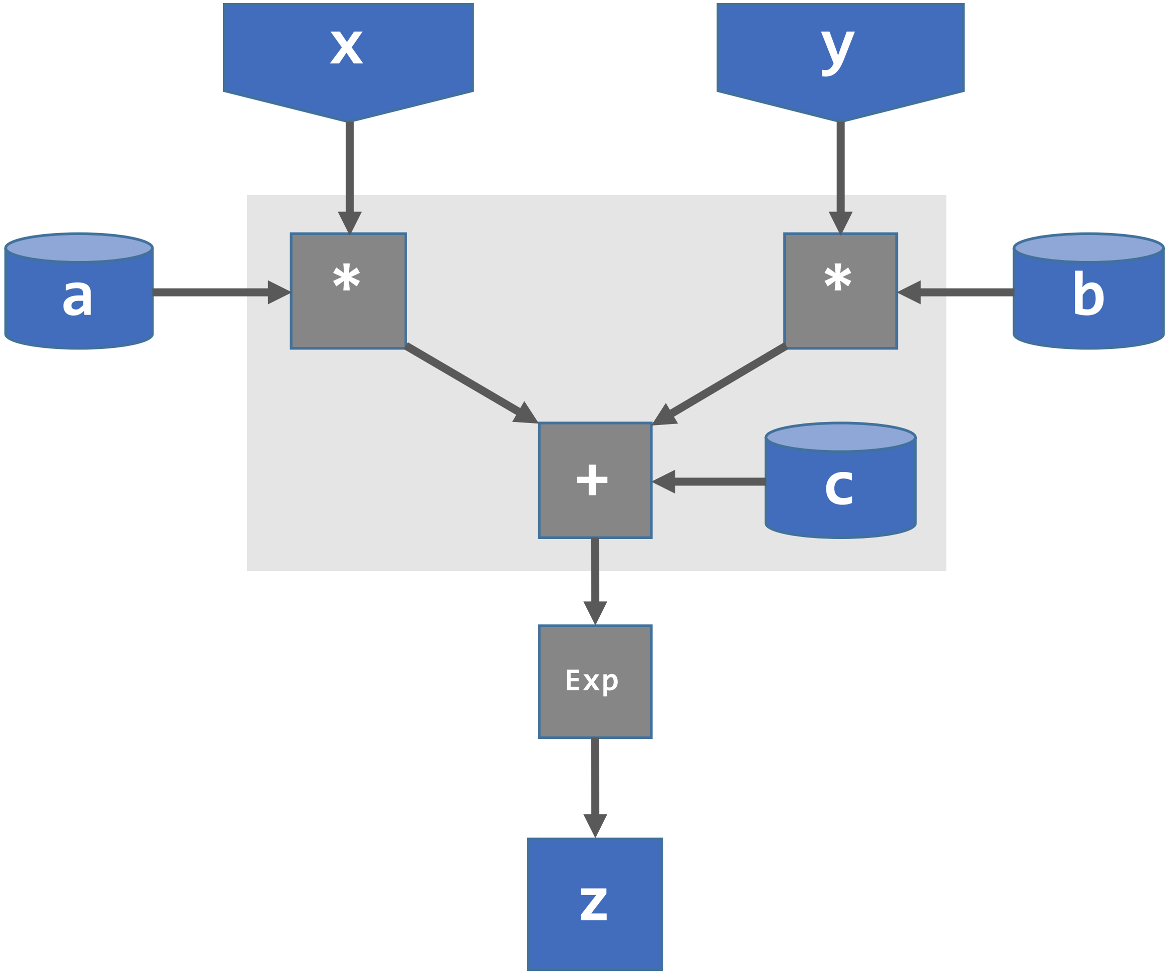


What about optimization?

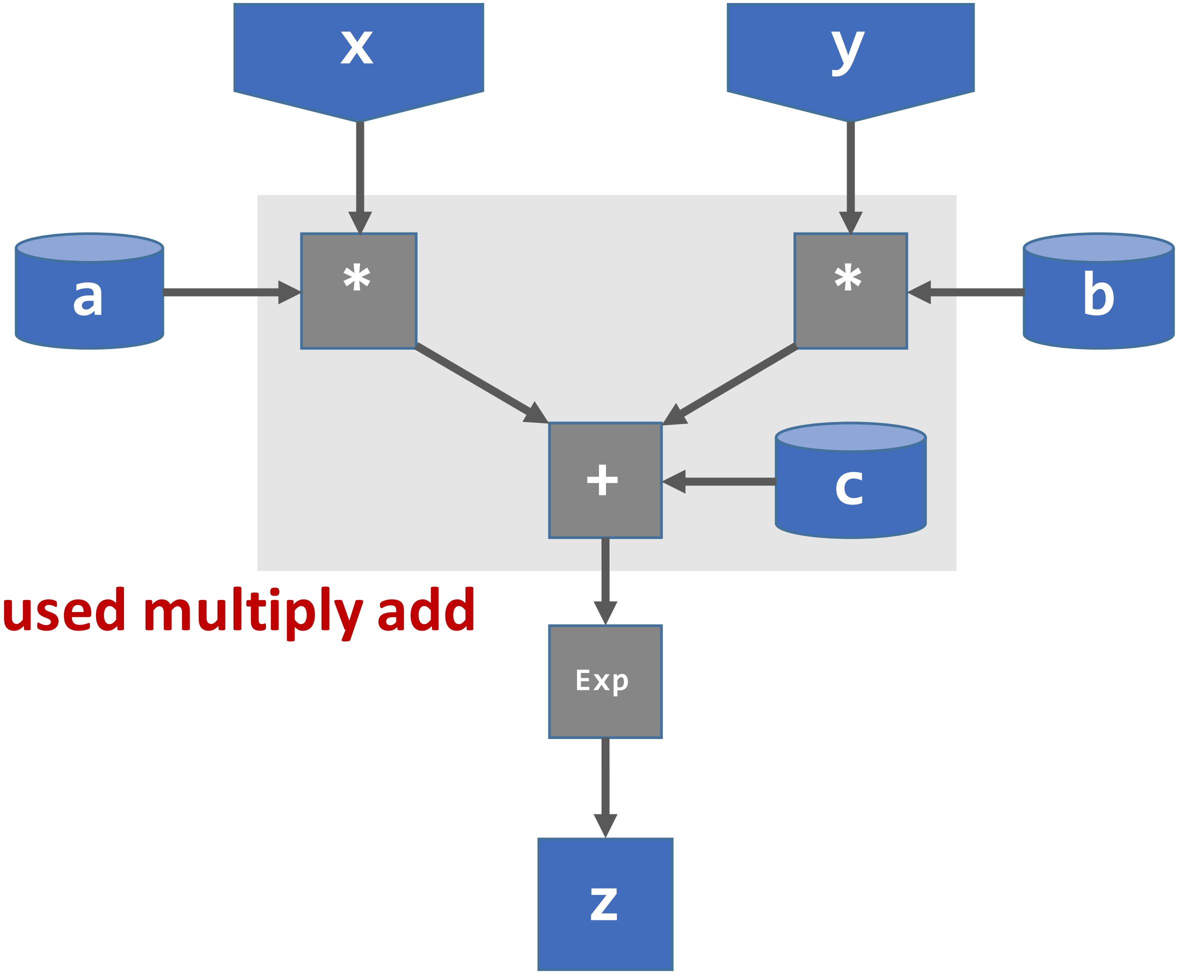


run in parallel

What about optimization?



What about optimization?



fused multiply add

Compute Graphs

- Deep networks are static
- Math can be represented as a graph

Compute Graphs

- **Deep networks are static**
- **Math can be represented as a graph**
- **This allows to automate**
  - Differentiation
  - Optimization
  - Parallelization

How do I define a graph?

How do I define a graph?

- You don’t have to worry about this!
- Mostly transparent to the user
- Math abstracted into network layers

```
X, Y = matrix(), vector()
```

Inputs

```
X, Y = matrix(), vector()
```

```
net = InputLayer(X)
net = DenseLayer(net, num_units=64)
net = DenseLayer(net, num_units=10,
                    nonlinearity=softmax)

prediction = get_output(net)
parameters = get_all_params(net)
```

Network

```
X, Y = matrix(), vector()

net = InputLayer(X)
net = DenseLayer(net, num_units=64)
net = DenseLayer(net, num_units=10,
                    nonlinearity=softmax)

prediction = get_output(net)
parameters = get_all_params(net)
```

```
loss = mean(categorical_crossentropy(prediction, Y)
accuracy = mean(eq(prediction, Y))
```

Cost

```
X, Y = matrix(), vector()

net = InputLayer(X)
net = DenseLayer(net, num_units=64)
net = DenseLayer(net, num_units=10,
                       nonlinearity=softmax)

prediction = get_output(net)
parameters = get_all_params(net)

loss = mean(categorical_crossentropy(prediction, Y))
accuracy = mean(eq(prediction, Y))

gradient = grad(loss, parameters)
updates = sgd(grad, parameters)
```

Optimization

```
X, Y = matrix(), vector()

net = InputLayer(X)
net = DenseLayer(net, num_units=64)
net = DenseLayer(net, num_units=10,
                    nonlinearity=softmax)

prediction = get_output(net)
parameters = get_all_params(net)

loss = mean(categorical_crossentropy(prediction, Y))
accuracy = mean(eq(prediction, Y))

gradient = grad(loss, parameters)
updates = sgd(grad, parameters)
```

```
f_train = theano.function([X, Y], loss, updates=updates)
f_valid = theano.function([X, Y], [loss, accuracy])
f_predict = theano.function([X], prediction)
```

Compile

```
f_train = theano.function([X, Y], loss, updates=updates)
f_valid = theano.function([X, Y], [loss, accuracy])
f_predict = theano.function([X], prediction)
```

```
{{ train loop }}

x_mb, y_mb = get_minibatch()
f_train(x_mb, y_mb)

{{ end }}
```

Numpy Arrays

Train

```
f_train = theano.function([X, Y], loss, updates=updates)
f_valid = theano.function([X, Y], [loss, accuracy])
f_predict = theano.function([X], prediction)

{{ train loop }}

x_mb, y_mb = get_minibatch()
f_train(x_mb, y_mb)

{{ end }}

x_v, y_v = get_validation_set()
valid_loss, valid_acc = f_valid(x_v, y_v)
```

Validation

```
f_train = theano.function([X, Y], loss, updates=updates)
f_valid = theano.function([X, Y], [loss, accuracy])
f_predict = theano.function([X], prediction)

{{ train loop }}

x_mb, y_mb = get_minibatch()
f_train(x_mb, y_mb)

{{ end }}

x_v, y_v = get_validation_set()
valid_loss, valid_acc = f_valid(x_v, y_v)

x_new = get_unlabeled_data()
y_new = f_pred(x_new)
```

Prediction

```
f_train = theano.function([X, Y], loss, updates=updates)
f_valid = theano.function([X, Y], [loss, accuracy])
f_predict = theano.function([X], prediction)

{{ train loop }}

x_mb, y_mb = get_minibatch()
f_train(x_mb, y_mb)

{{ end }}

x_v, y_v = get_validation_set()
valid_loss, valid_acc = f_valid(x_v, y_v)

x_new = get_unlabeled_data()
y_new = f_pred(x_new)
```

Prediction

It’s just python!

Pros & Cons

+ Automatic differentiation & optimization

+ Transparent GPU usage

+ Programming in Python

– Only Static Graphs

– Learning Curve

Theano or Tensorflow?

- **Theano**
  - Made by researchers, for researchers
  - Mature
  - Compiles C on the fly
- **Tensorflow**
  - Made by Google
  - Still in Beta
  - Focus on large scale learning & production

Links

- **Theano + Lasagne**
  - `github.com/theano/theano`
  - `github.com/lasagne/lasagne`
- **Tensorflow**
  - `github.com/tensorflow/tensorflow`
- **Keras**
  - `github.com/fchollet/keras`