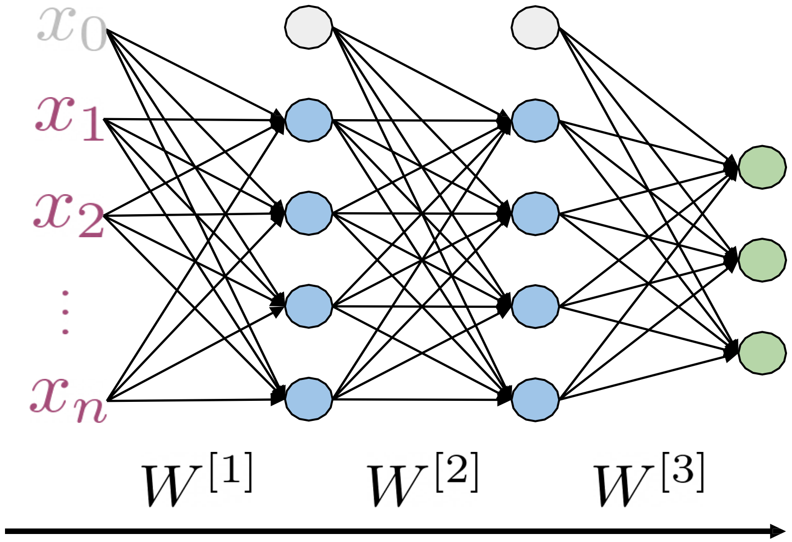
Sequence Models

- Neural Networks for sentiment analysis
- Recurrent Neural Networks for Language Modeling
- LSTMs and Named Entity Recognition
- Siamese Networks

- Neural Networks for Sentiment Analysis
  - Structure for sentiment analysis
  - Neural Networks in Trax
  - Classes, Subclasses and Inheritance

Forward propagation

- A NN with n input parameters to hidden layers and 3 output units as input.
- A data representation x with n features superscript 0 to be the input vector x.
- Superscript i, which depends on both the weights matrix for that layer and the activations A from the previous layer.



$a^{[i]}$ Activations ith layer

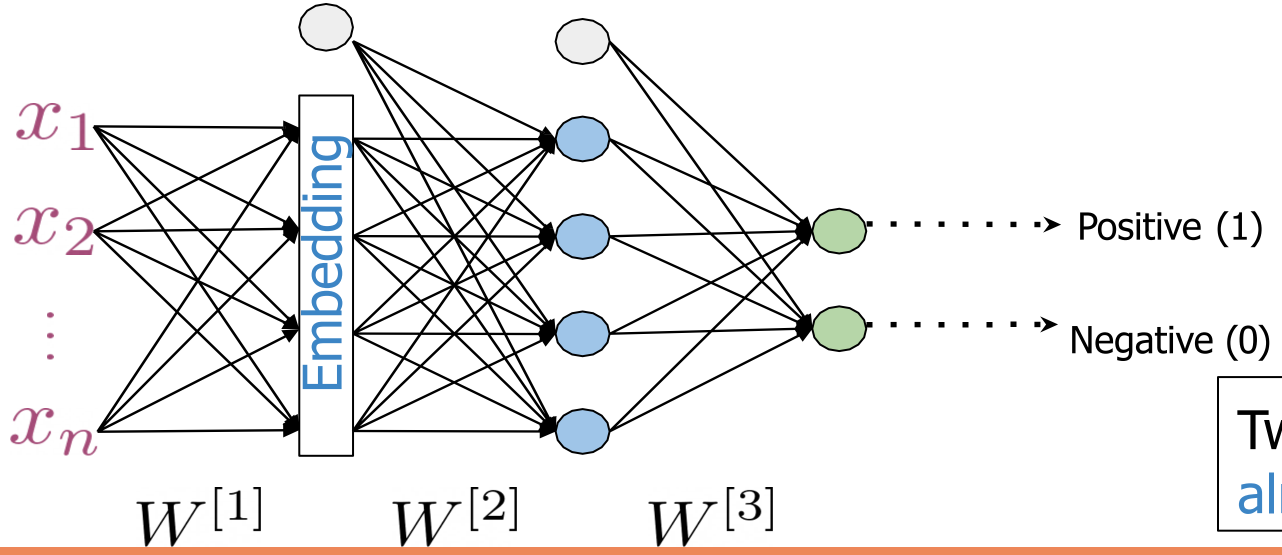
$$a^{[0]} = X$$

$$z^{[i]} = W^{[i]} a^{[i-1]}$$

$$a^{[i]} = g^{[i]}(z^{[i]})$$

Neural Networks for sentiment analysis

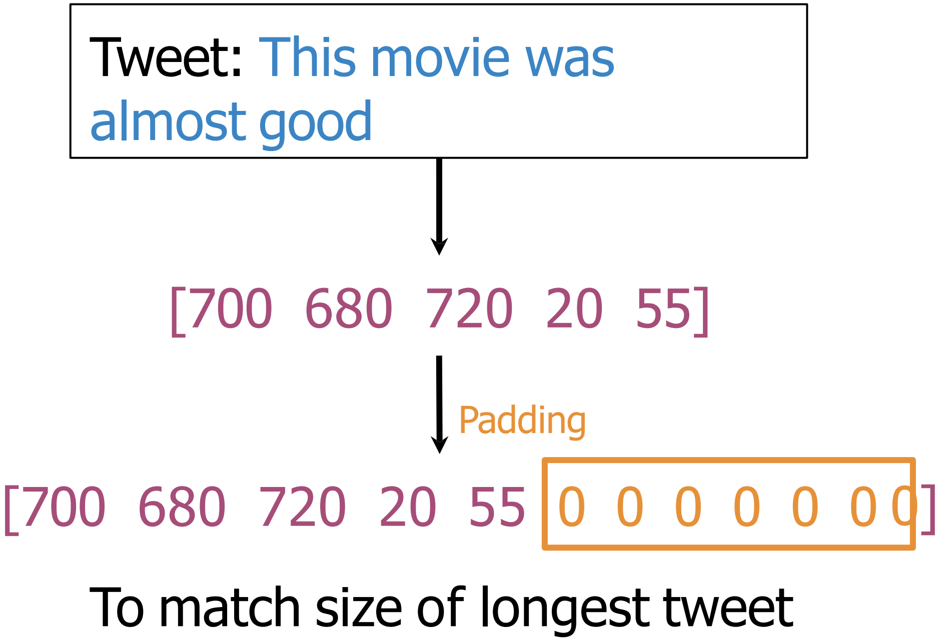
- A simple vector presentation of your tweets
- An embedding layer that will transform your representation
- A hidden layer with relu activation function and
- An output layer with a softmax function that will give you the probabilities for whether a tweet has a positive or negative sentiment.



Tweet: This movie was almost good

Initial Representation

Word	Number
a	1
able	2
about	3
…	…
hand	615
…	…
happy	621
…	…
zebra	1000



Tweet: This movie was almost good

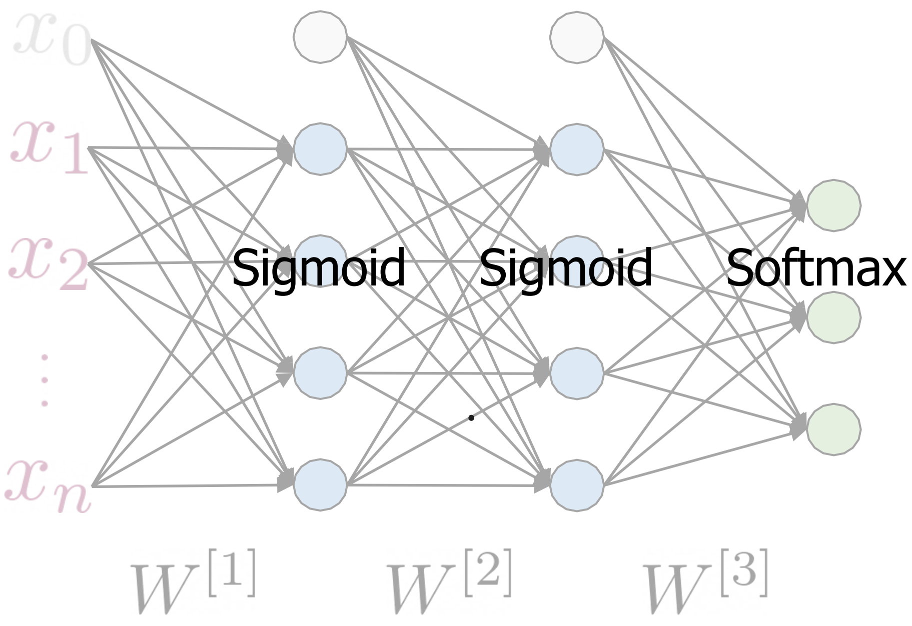
[700 680 720 20 55]

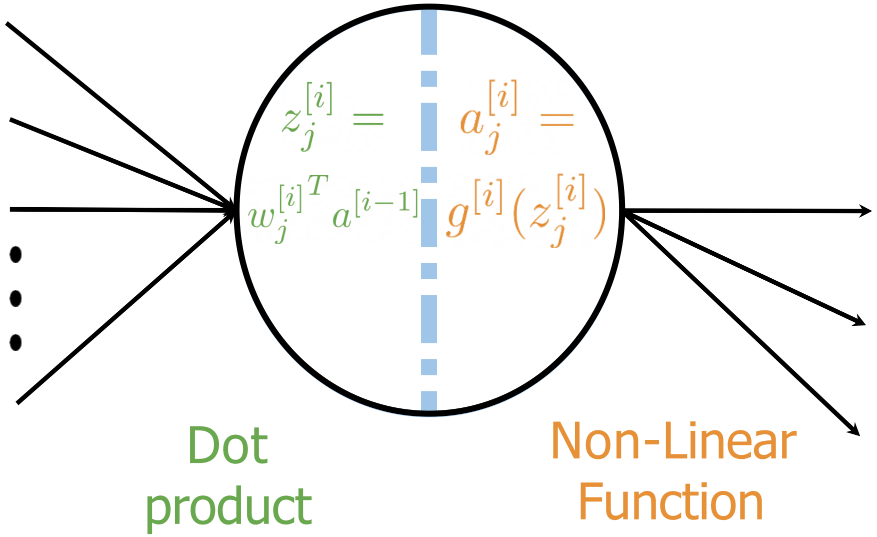
Padding

[700 680 720 20 55 0 0 0 0 0 0 0]

To match size of longest tweet

Neural Networks in Trax

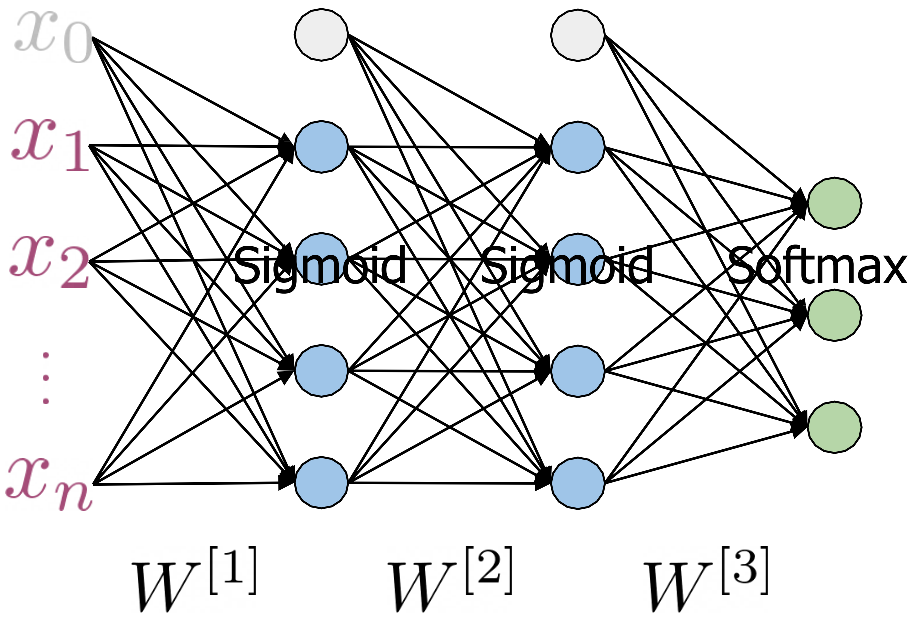




$z_j^{[i]} = w_j^{[i]^T} a^{[i-1]}$
$a_j^{[i]} = g^{[i]}(z_j^{[i]})$
Dot product
Non-Linear Function

- Let's take this network architecture as an example.
- In this model you have to hidden layers with sigmoid activation functions and an output layer with softmax activation

Neural Networks in Trax



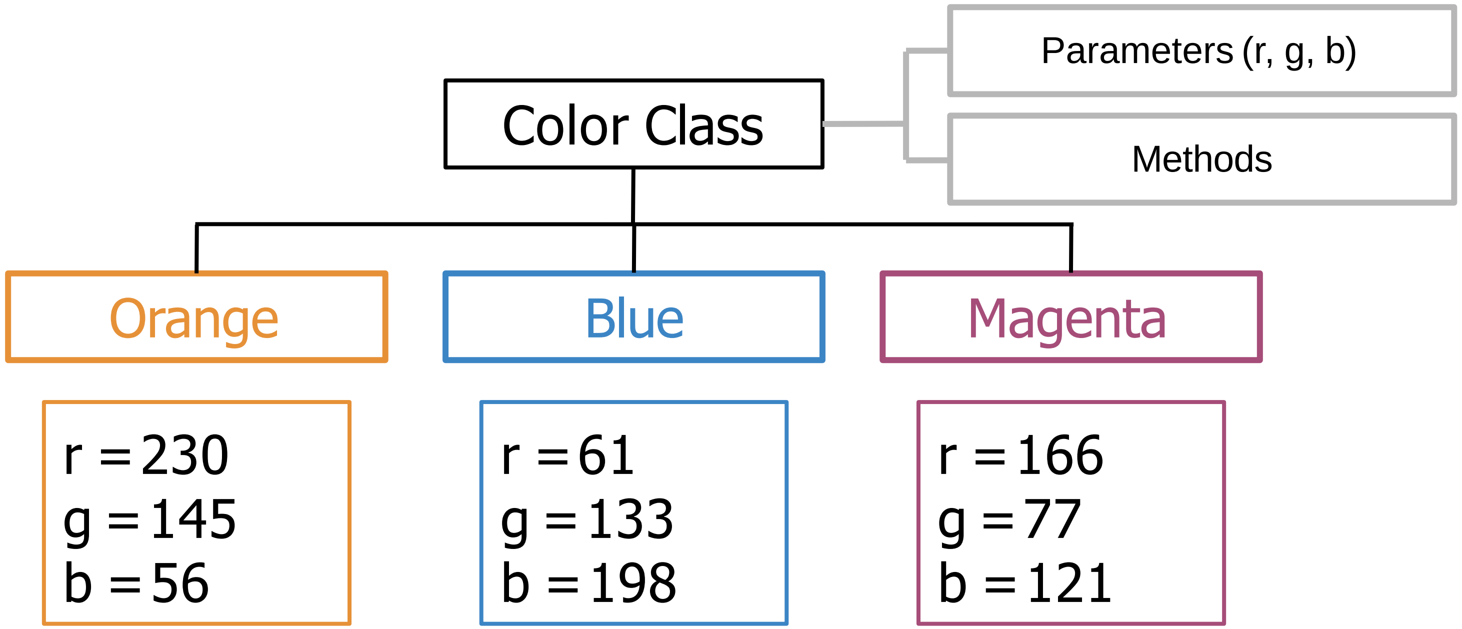
```
from trax import layers as tl
Model = tl.Serial(
           tl.Dense(4),
           tl.Sigmoid(),
           tl.Dense(4),
           tl.Sigmoid(),
           tl.Dense(3),
           tl.Softmax())
```

Advantages of using frameworks

- Run fast on CPUs, GPUs and TPUs
- Parallel computing
- Record algebraic computations for gradient evaluation Tensorflow Pytorch TRAX
- Order of computation ⟶ Model in Trax
- Benefits from using frameworks

Classes, Subclasses and Inheritance

- Classes
  - How classes work and their implementation
  - Subclasses and inheritance



Color Class

Parameters (r, g, b)

Methods

Orange

r = 230
g = 145
b = 56

Blue

r = 61
g = 133
b = 198

Magenta

r = 166
g = 77
b = 121

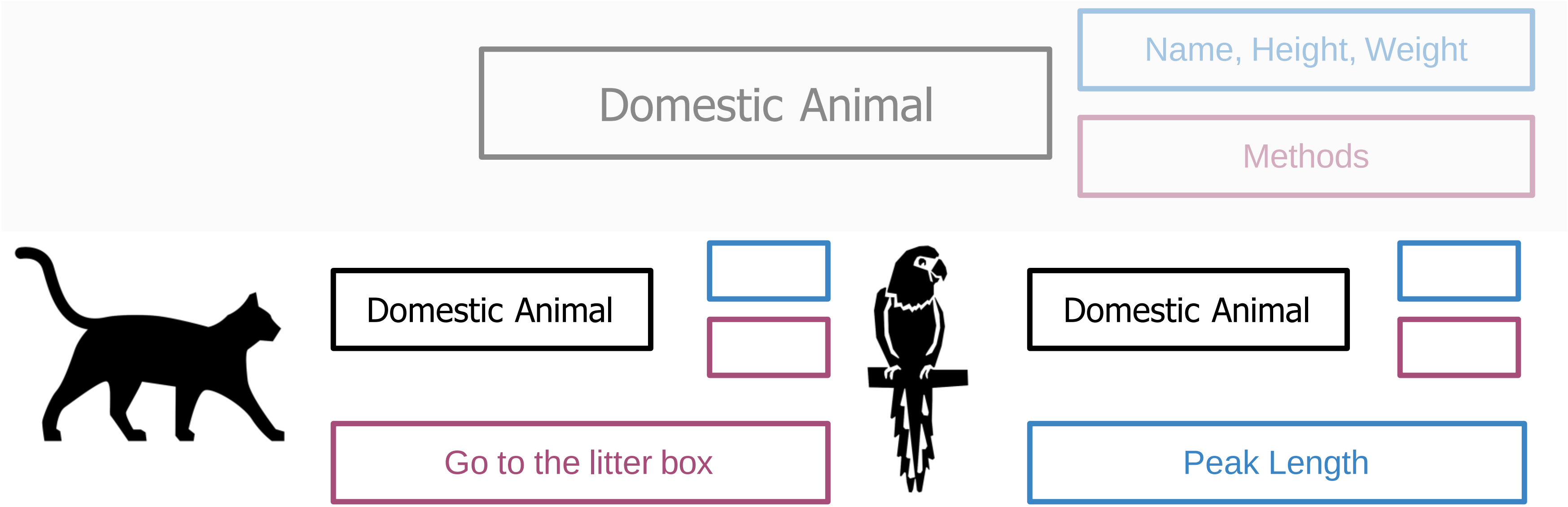
Classes in Python

```
class MyClass:
    def __init__(self, y):
            self.y = y
    def my_method(self,x):
            return x + self.y
    def __call__(self, x):
            return
self.my_method(x)
```

```
f = MyClass(7)
print(f(3))
```

```
10
```

Subclasses and Inheritance



Convenient when classes share common parameters and methods.

Subclasses

```
class MyClass:
    def __init__(self,y):
            self.y = y
    def my_method(self,x):
            return x +
f.y
    def __call__(self,x):
            Return
self.my_method(x)
```

```
class SubClass(MyClass):
    def my_method(self,x):
            return x +
self.y**2
```

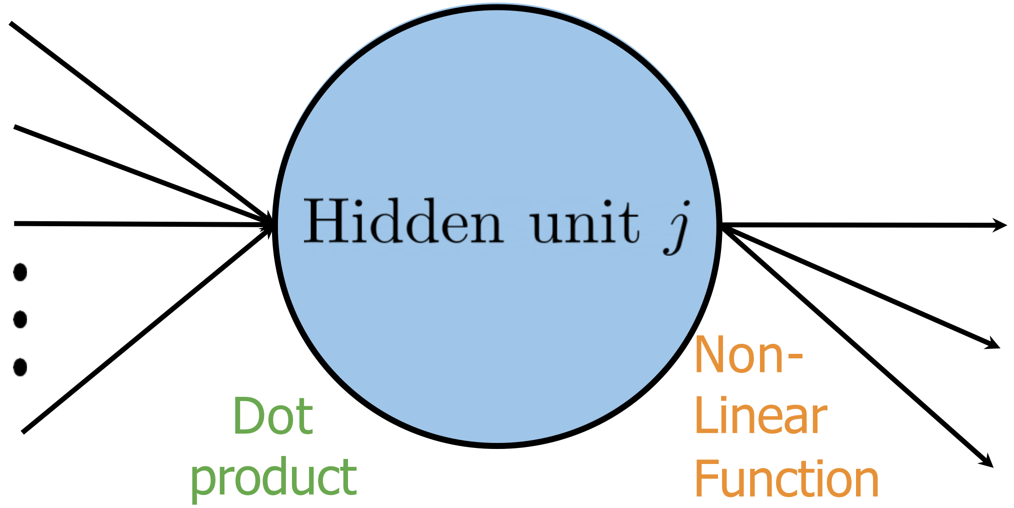
```
f = SubClass(7)
print(f(3))
```

```
52
```

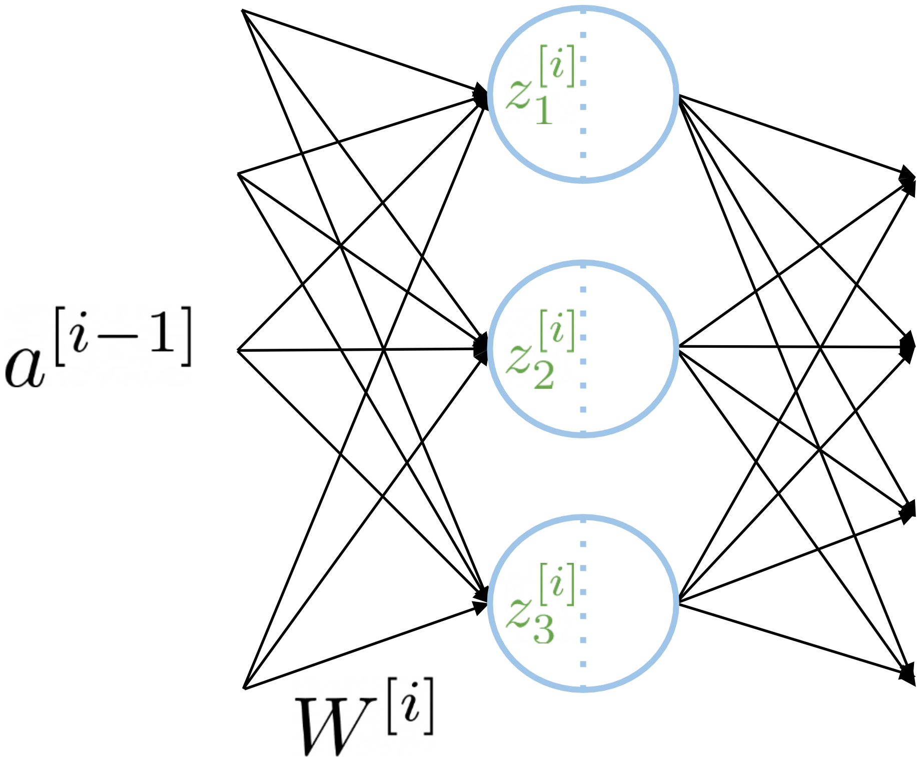
Dense and ReLU Layers

- Dense layer in detail
- ReLU function

- Neural networks in Trax



Dense Layer



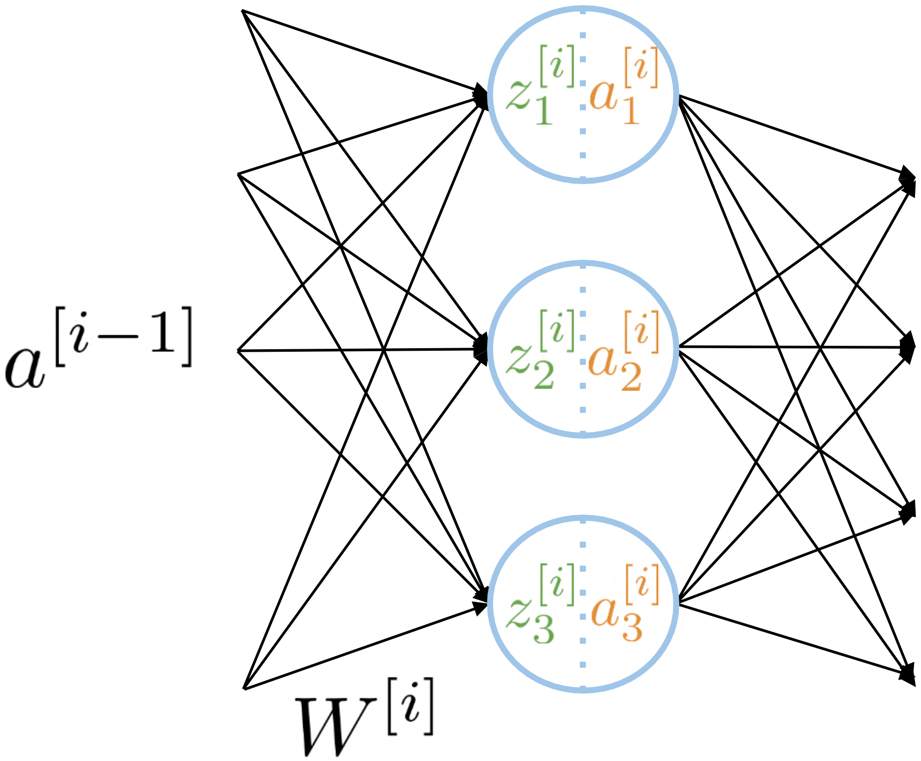
$$z_j^{[i]} = w_j^{[i]T} a^{[i-1]}$$

Dense layer

$$z^{[i]} = \boxed{W^{[i]}} a^{[i-1]}$$

Trainable parameters

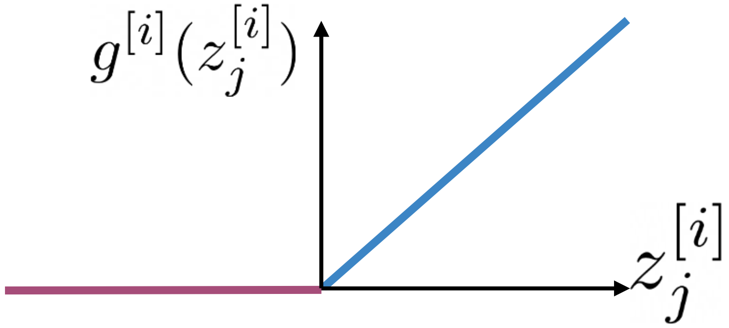
ReLU Layer



$$a_j^{[i]} = g^{[i]}(z_j^{[i]})$$

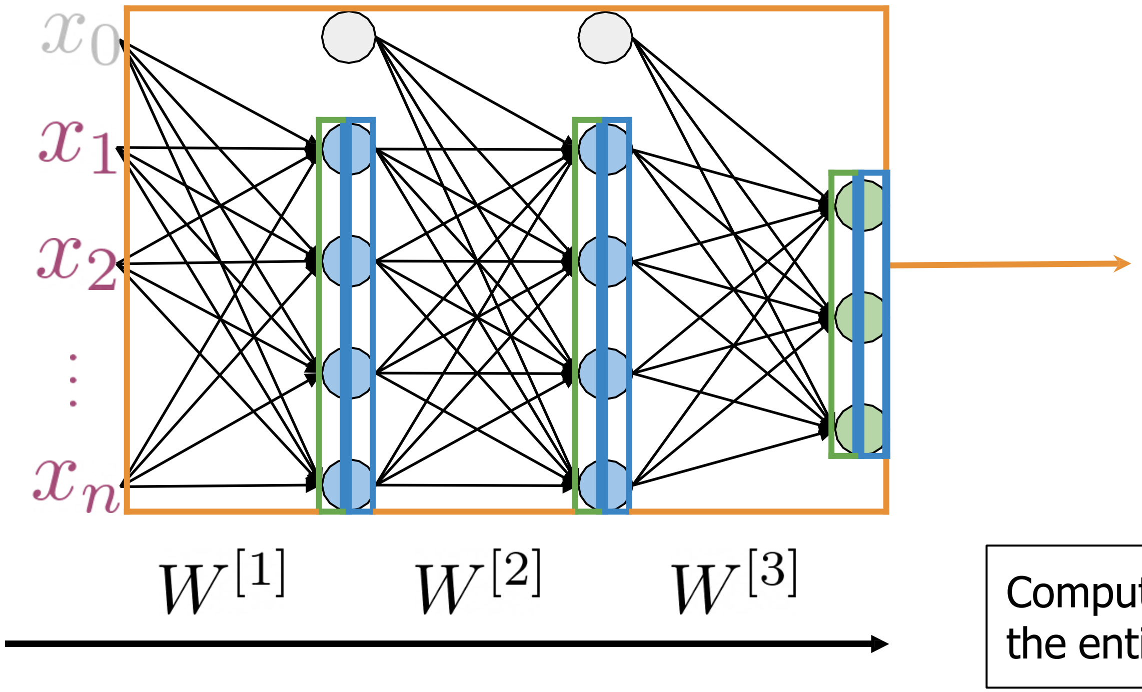
ReLU = Rectified linear unit

$$g(z^{[i]}) = \max(\underline{0}, \underline{z^{[i]}})$$



- Dense Layer ⟶ $z^{[i]} = W^{[i]}a^{[i-1]}$
- ReLU Layer ⟶ $g(z^{[i]}) = \max(0, z^{[i]})$

Serial Layer



Composition of layers in *serial* arrangement

- Dense Layers
- Activation Layers

Compute forward propagation of the entire model

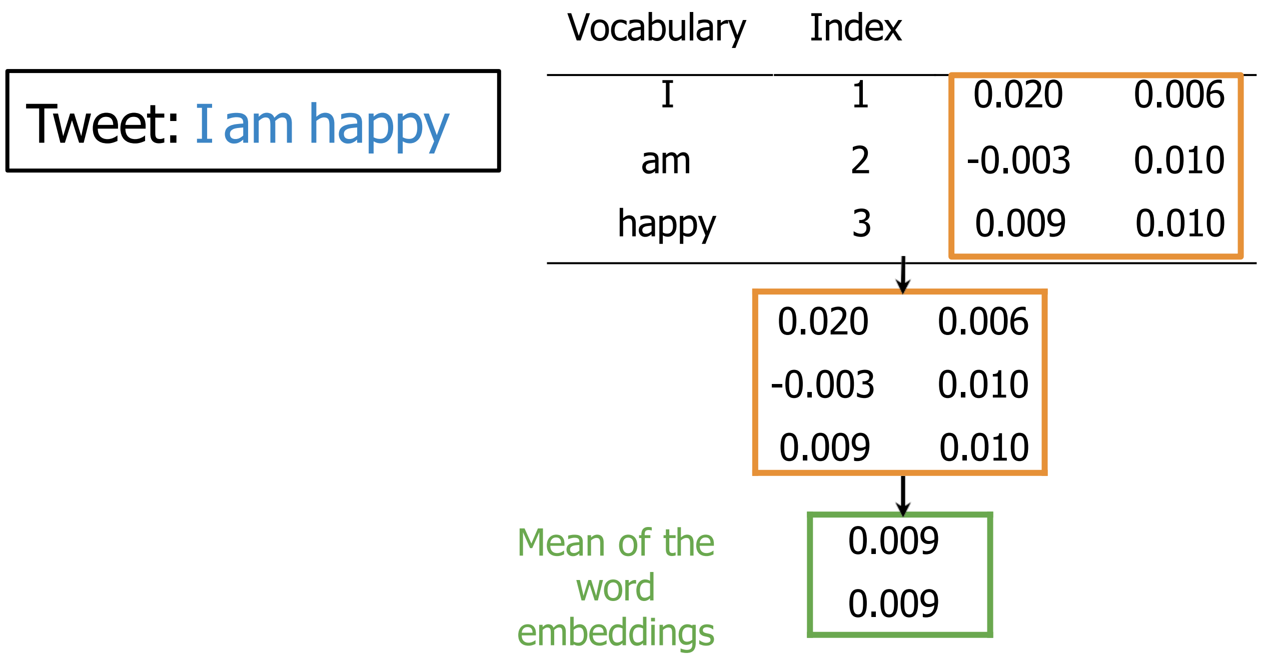
Embedding Layer

Vocabulary	Index		
I	1	0.020	0.006
am	2	-0.003	0.010
happy	3	0.009	0.010
because	4	-0.011	-0.018
learning	5	-0.040	-0.047
NLP	6	0.009	0.050
sad	7	-0.044	0.001
not	8	0.011	-0.022

Trainable weights

Vocabulary
x
Embedding

Mean Layer



Tweet: I am happy

Vocabulary	Index		
I	1	0.020	0.006
am	2	-0.003	0.010
happy	3	0.009	0.010

0.020	0.006
-0.003	0.010
0.009	0.010

Mean of the word embeddings

0.009
0.009

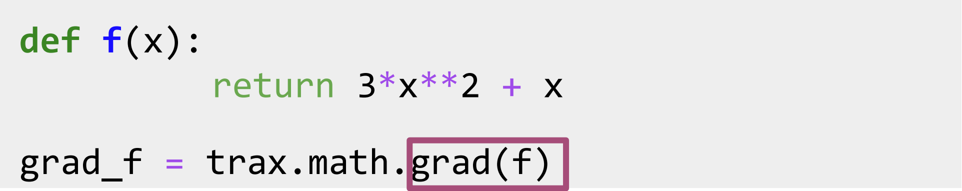
No trainable parameters

Computing gradients in Trax

$$f(x) = 3x^2 + x$$

$$\boxed{\frac{\delta f(x)}{\delta x}} = 6x + 1$$

Gradient

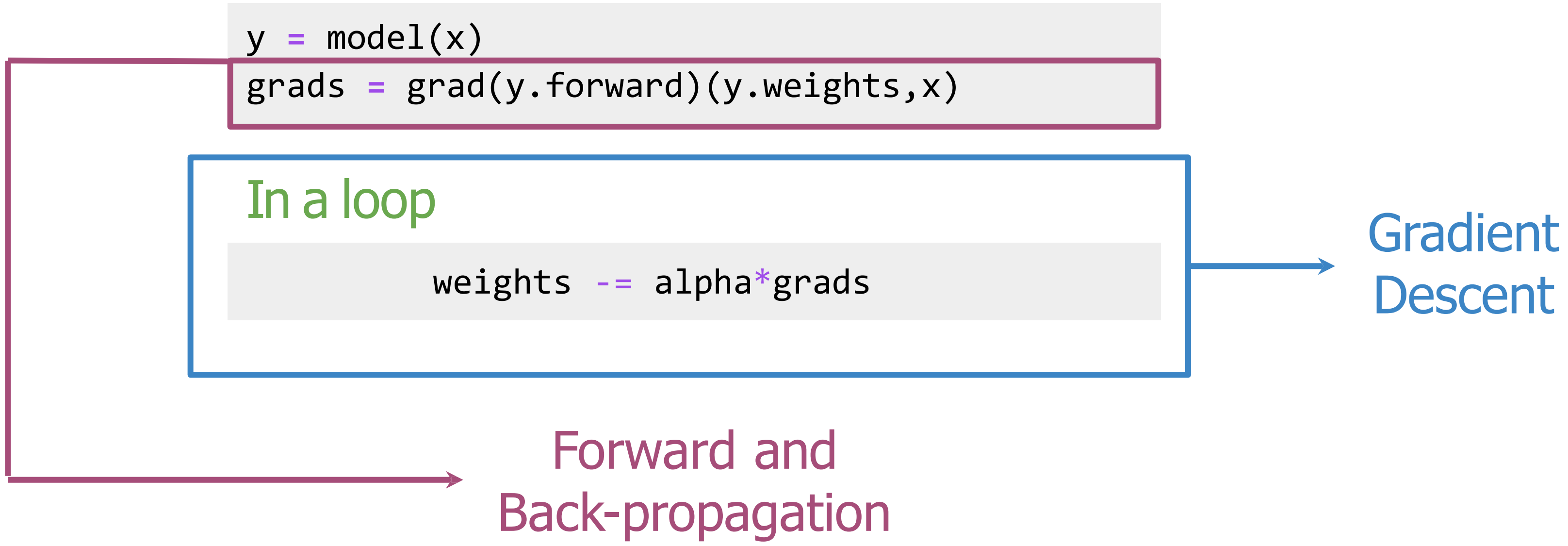


```
def f(x):
        return 3*x**2 + x

grad_f = trax.math.grad(f)
```

Returns a function

Training with grad()



```
y = model(x)
grads = grad(y.forward)(y.weights,x)
```

In a loop

```
        weights -= alpha*grads
```

Gradient Descent

Forward and Back-propagation