MULTI-LAYER NETWORKS AND BACK PROPAGATION

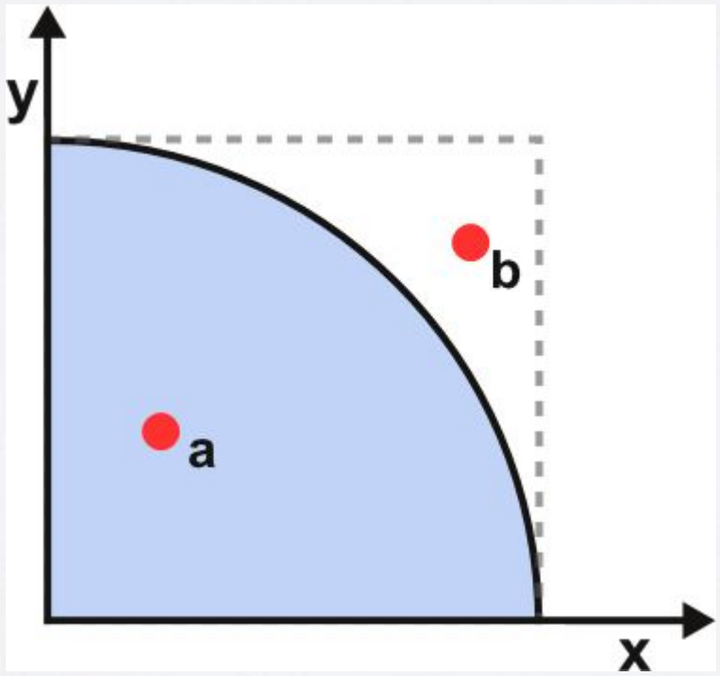
Philipp Krähenbühl

ADMINISTRATIVE

- Prof. Krähenbühl out of town this week
- HW 1
- HW 2

HW 1

- Given k $\{x_i\} \sim U[0,1]^2$, estimate π



HW 1

- Naive estimator
- $U(x,y) = 1(x^2+y^2 < 1)$, using k samples
- $E(U) = \pi/4$, $E(\bar{U})=\pi/4$
- ```
  def forward(self, x):
      return 4*torch.mean((torch.sum(x**2, dim=1)<1).float(), dim=0)
  ```
  ```

HW 1

- Better estimator
  - U(x) = sqrt(1-x^2), using 2k samples
- Still does monte-carlo estimate, but integrates one dim in close form
-
```
def forward(self, x):
    return 4*torch.mean(torch.sqrt(1-x**2))
```

HW 1

- Another interpretation
- $E_{xy}(1\{x^2+y^2<1\}) = E_x E_y(1\{x^2+y^2<1\} \mid x)) = \pi/4$
  - $1\{x^2+y^2<1\} \Leftrightarrow 1\{y<\text{sqrt}(1-x^2)\}$
  - $E_y(1\{y<\text{sqrt}(1-x^2)\}|x) = \text{sqrt}(1-x^2)$

HW 1

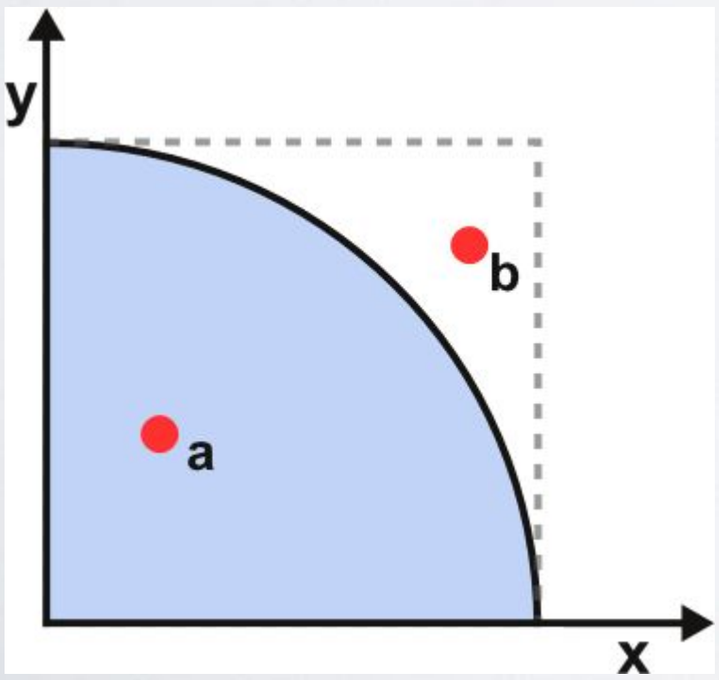
- Another interpretation
- $E_x E_y(1\{x^2+y^2<1\} \mid x)) = E_x(\text{sqrt}(1-x^2)) = \pi/4$
  - $E(U) = \pi/4$

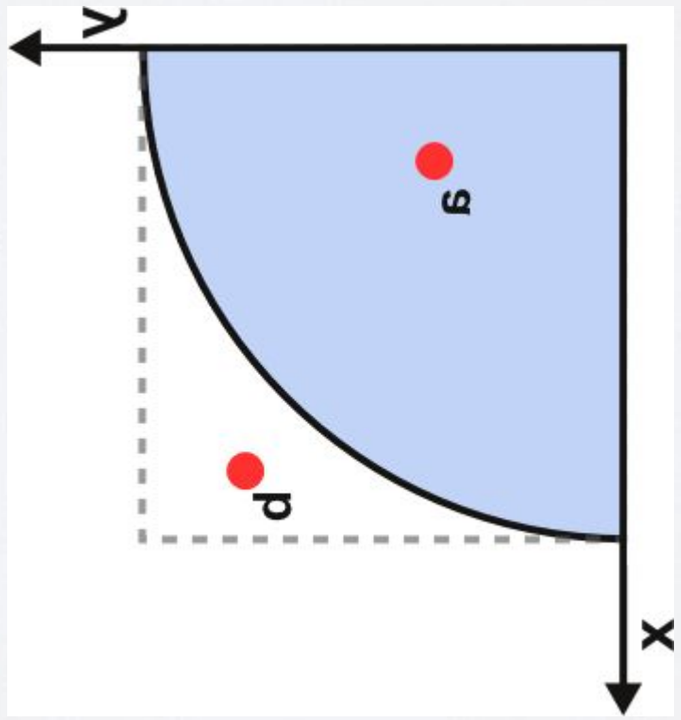
HW 1

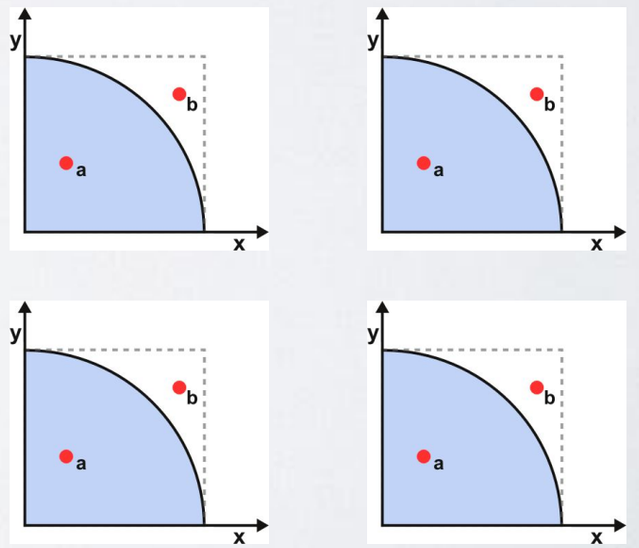
- Better estimator
  - It uses more samples

HW 1

- Can we do even better?
  - Squeeze more information from data
- x = x*2 - torch.floor(x*2)
- x = 1-x







RECAP

- Loss function

  - How well are fitting the data?

- Gradient computation

  - What **local** change to the weights decreases the loss?

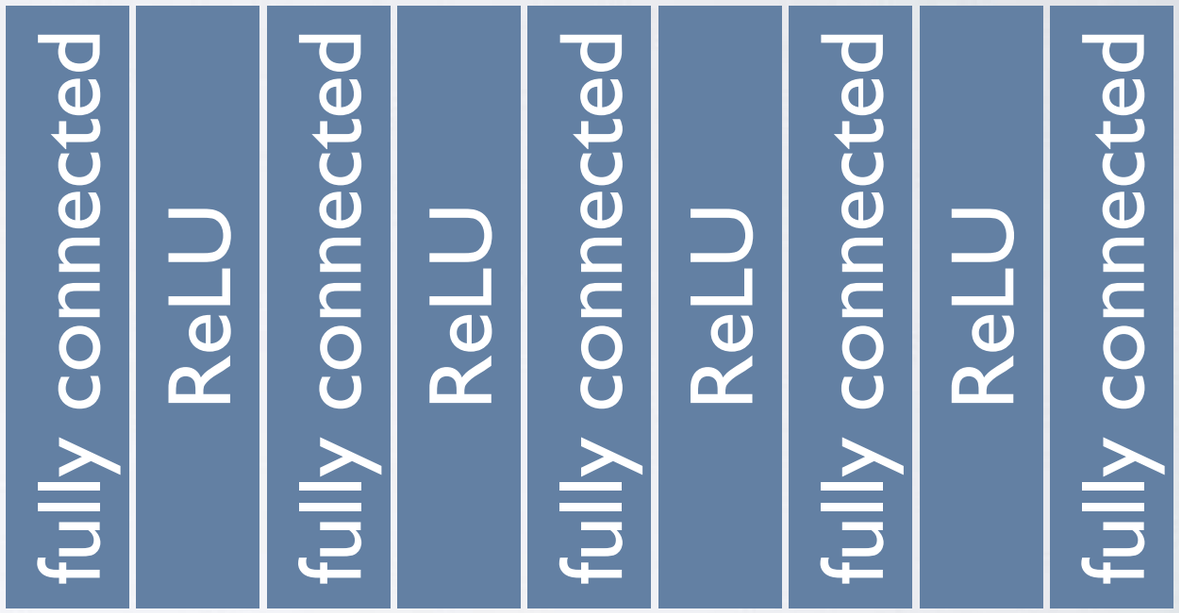
- Optimization algorithm

  - How do we change the weights to **globally** minimize the loss?

Last Thu

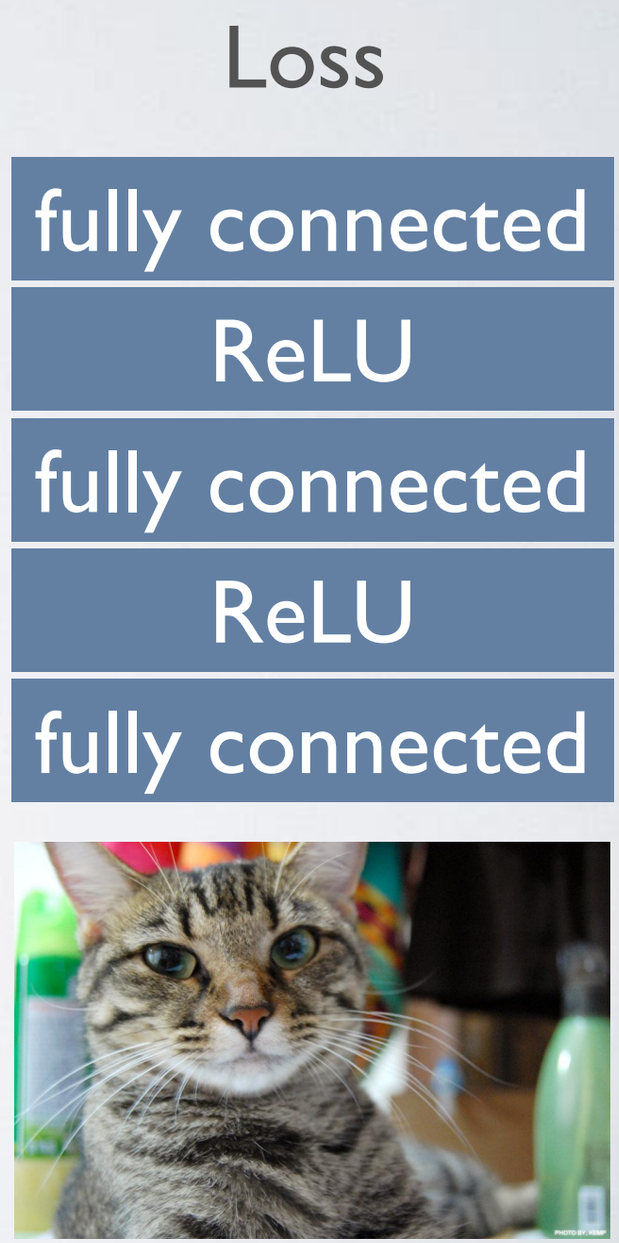
Today

Th



GRADIENT

- Forward pass
  - Compute all latent activations z
  - Compute the loss
- How do we compute the gradient of l w.r.t. W_1, W_3, W_5?
  - l and all f are differentiable



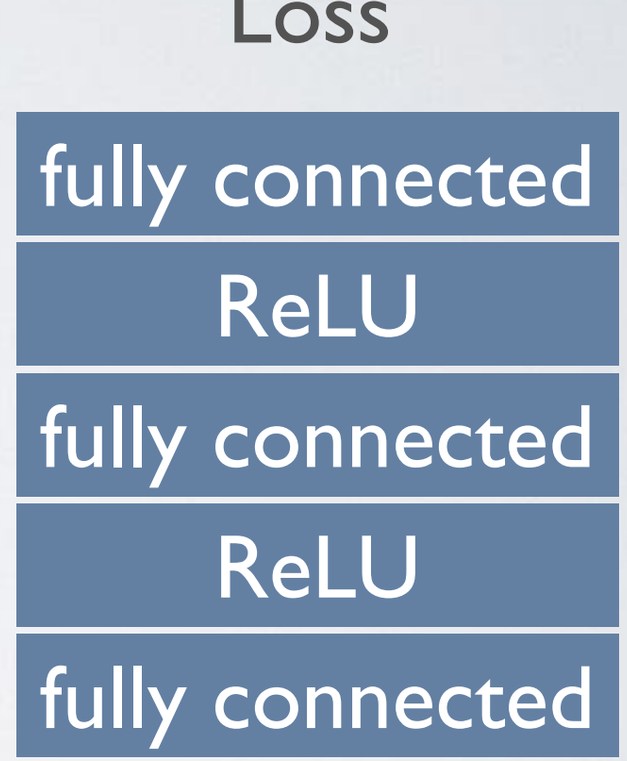
GRADIENT

- Forward is a gigantic nested function?

$$l(f_5(f_4(f_3(f_2(f_1(x, W_1)), W3)), W5)$$

- Compute the gradient using chain-rule

$$l(z_5)$$
$$z_5 = f_5(z_4, W_5)$$
$$z_4 = f_4(z_3)$$
$$z_3 = f_3(z_2, W_3)$$
$$z_2 = f_2(z_1)$$
$$z_1 = f_1(x, W_1)$$



GRADIENT

$$\frac{d}{dW_5} l(f_5(f_4(f_3(f_2(f_1(x, W_1)), W3)), W5)) = \frac{d}{dz_5} l(z_5) \frac{d}{dW_5} f_5(z_4, W_5)$$

$$\frac{d}{dW_5} l(z_5) = \frac{d}{dz_5} l(z_5) \frac{d}{dz_4} f_5(z_4, W_5) \frac{d}{dz_3} f_4(z_3) \frac{d}{dW_3} f_3(z_2, W_3)$$

$\dots$

$$l(z_5)$$

$$z_5 = f_5(z_4, W_5)$$

$$z_4 = f_4(z_3)$$

$$z_3 = f_3(z_2, W_3)$$

$$z_2 = f_2(z_1)$$

$$z_1 = f_1(x, W_1)$$

GRADIENT + CHAIN RULE

$$\ell(z_4)$$
$$z_4 = f_4(z_3)$$
$$z_3 = f_3(z_2, W_3)$$
$$z_2 = f_2(z_1)$$
$$z_1 = f_1(x, W_1)$$

$$\frac{\partial}{\partial W_3}\ell(z_4)$$
$$= \frac{\partial}{\partial z_4}\ell(z_4)\frac{\partial}{\partial W_3}z_4$$
$$= \frac{\partial}{\partial z_4}\ell(z_4)\frac{\partial}{\partial z_3}z_4\frac{\partial}{\partial W_3}z_3$$

EVALUATION

$$\frac{\partial}{\partial W_3}\ell(z_4) = \frac{\partial}{\partial z_4}\ell(z_4)\frac{\partial}{\partial z_3}z_4\frac{\partial}{\partial W_3}z_3$$

scalar

1d

EVALUATION

$$\frac{\partial}{\partial W_3}\ell(z_4) = \frac{\partial}{\partial z_4}\ell(z_4)\frac{\partial}{\partial z_3}z_4\frac{\partial}{\partial W_3}z_3$$

$$1 \times |W_3|$$

EVALUATION

$$\frac{\partial}{\partial W_3}\ell(z_4) = \frac{\partial}{\partial z_4}\ell(z_4)\frac{\partial}{\partial z_3}z_4\frac{\partial}{\partial W_3}z_3$$

$$\left|z_3\right|$$

EVALUATION

$$\frac{\partial}{\partial W_3}\ell(z_4) = \frac{\partial}{\partial z_4}\ell(z_4)\frac{\partial}{\partial z_3}z_4\frac{\partial}{\partial W_3}z_3$$

$$|z_3| \times |W_3|$$

EVALUATION

$$\frac{\partial}{\partial W_3}\ell(z_4) = \frac{\partial}{\partial z_4}\ell(z_4)\frac{\partial}{\partial z_3}z_4\frac{\partial}{\partial W_3}z_3$$

$$|z_4| \times |z_3|$$

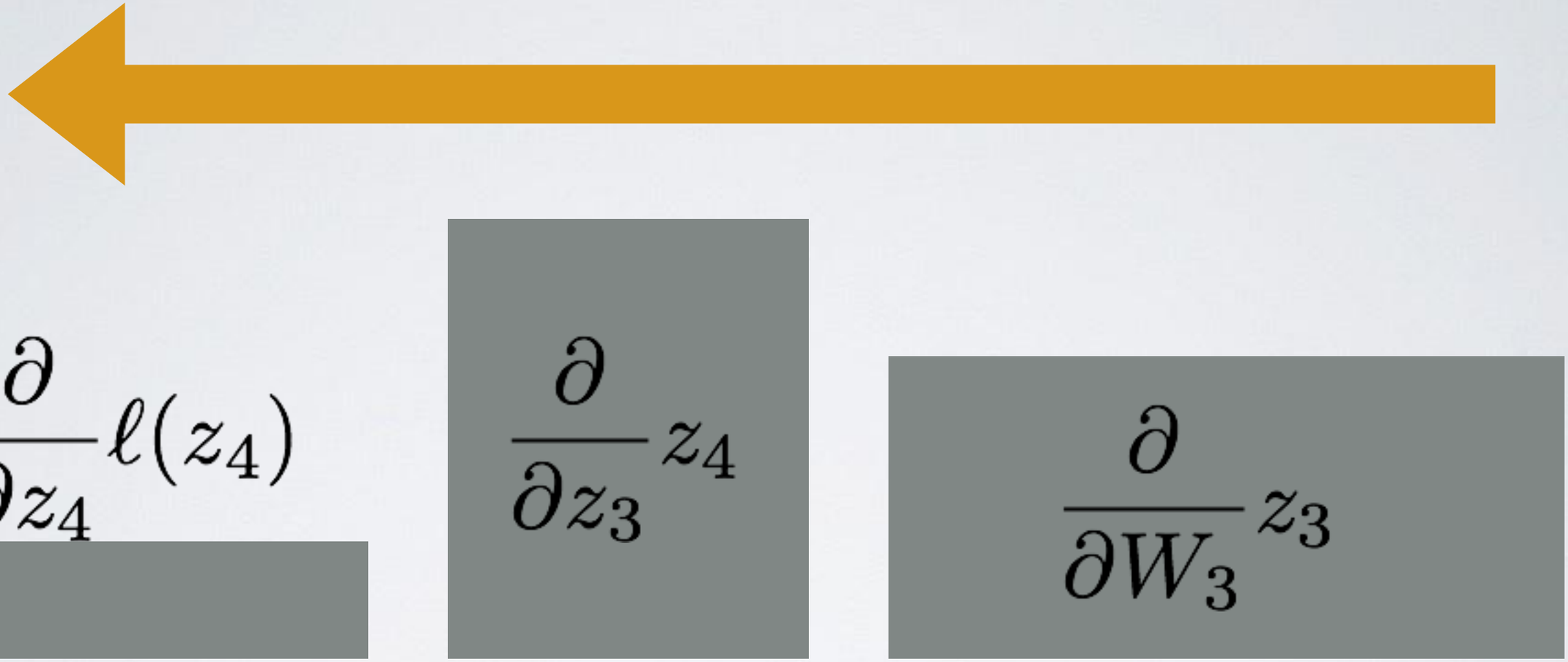
$$|z_3| \times |W_3|$$

EVALUATION

$$\frac{\partial}{\partial W_3}\ell(z_4) = \frac{\partial}{\partial z_4}\ell(z_4 \quad \frac{\partial}{\partial z_3} z_4 \frac{\partial}{\partial W_3} z_3$$

$$1 \times |z_4| \qquad |z_4| \times |z_3| \qquad |z_3| \times |W_3|$$

EVALUATION

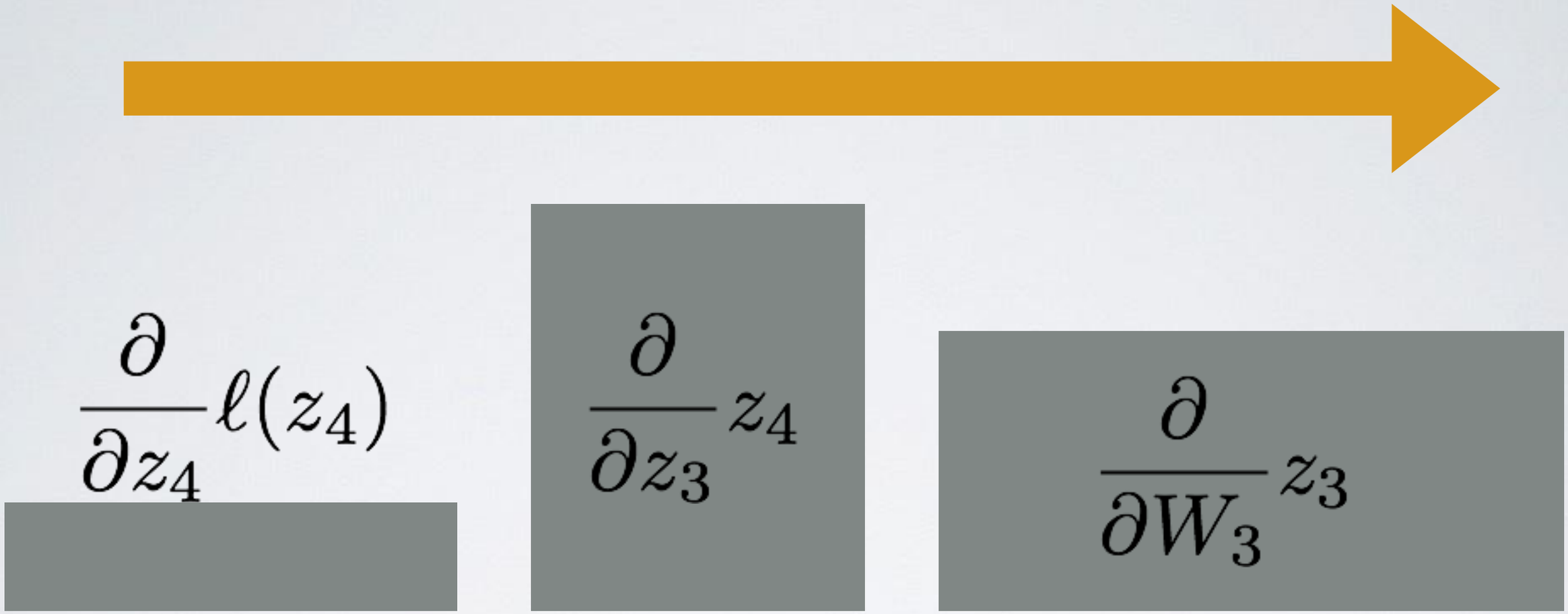


$$\frac{\partial}{\partial z_4}\ell(z_4) \qquad \frac{\partial}{\partial z_3}z_4 \qquad \frac{\partial}{\partial W_3}z_3$$

$$1 \times |z_4| \qquad |z_4| \times |z_3| \qquad |z_3| \times |W_3|$$

$$O(1|z_4||W_3|) \quad O(|z_4||z_3||W_3|)$$

EVALUATION



$$\frac{\partial}{\partial z_4}\ell(z_4) \qquad \frac{\partial}{\partial z_3}z_4 \qquad \frac{\partial}{\partial W_3}z_3$$

$$1 \times |z_4| \qquad |z_4| \times |z_3| \qquad |z_3| \times |W_3|$$

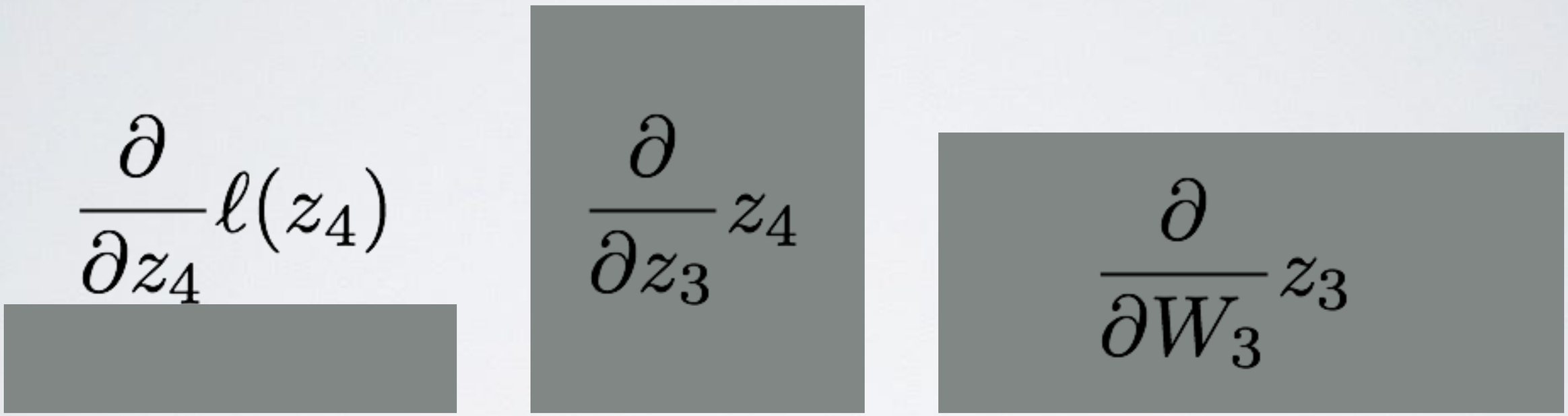
$$O(1|z_4||z_3|) \qquad O(1|z_3||W_3|)$$

EVALUATION

$$O(|z_4||W_3| + |z_4||z_3||W_3|)$$

$$O(|z_4||z_3| + |z_3||W_3|)$$

BACK PROPAGATION



DEEP NETWORK

dog vs **cat**

$$\ell(z_4)$$

Non-linearity

$$z_4 = f_4(z_3)$$

Linear transformation W

$$z_3 = f_3(z_2, W_3)$$

Non-linearity

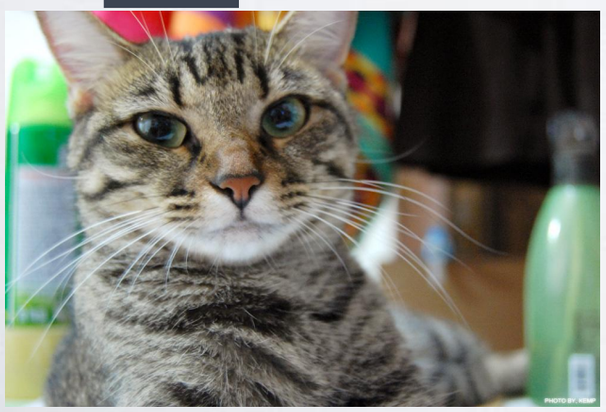
$$z_2 = f_2(z_1)$$

Linear transformation W

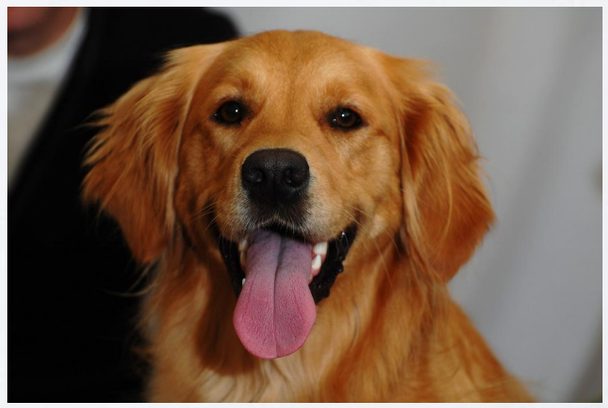
$$z_1 = f_1(x, W_1)$$

forward

backward



/



EFFICIENT COMPUTATION?

$$\frac{\partial}{\partial W_3}\ell(z_4) = \frac{\partial}{\partial z_4}\ell(z_4)\frac{\partial}{\partial z_3}z_4\frac{\partial}{\partial W_3}z_3$$

$$\frac{\partial}{\partial W_1}\ell(z_4) = \frac{\partial}{\partial z_4}\ell(z_4)\frac{\partial}{\partial z_3}z_4\frac{\partial}{\partial z_2}z_3\frac{\partial}{\partial z_1}z_2\frac{\partial}{\partial W_1}z_1$$

DEEP NETWORK

dog vs **cat**

$$\ell(z_4)$$

Non-linearity

$$z_4 = f_4(z_3)$$

Linear transformation W

$$z_3 = f_3(z_2, W_3)$$

Non-linearity

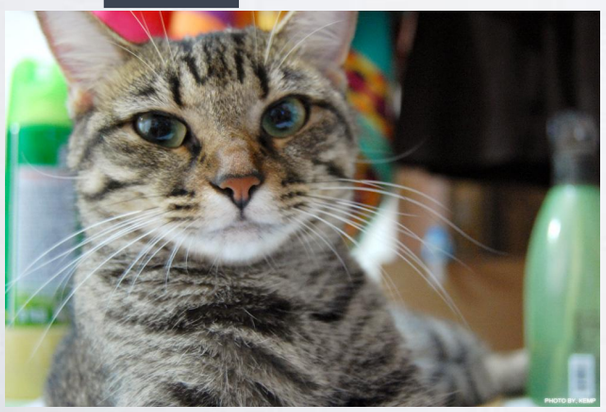
$$z_2 = f_2(z_1)$$

Linear transformation W

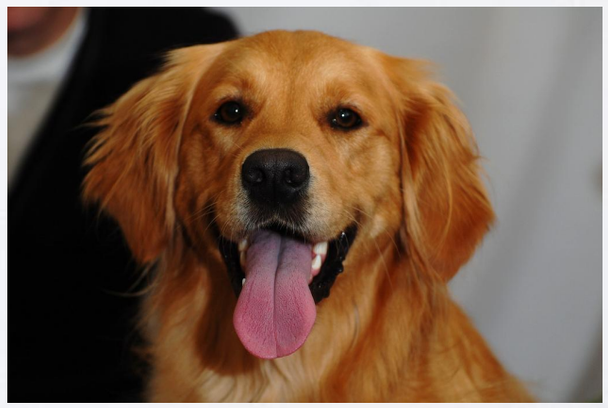
$$z_1 = f_1(x, W_1)$$

forward

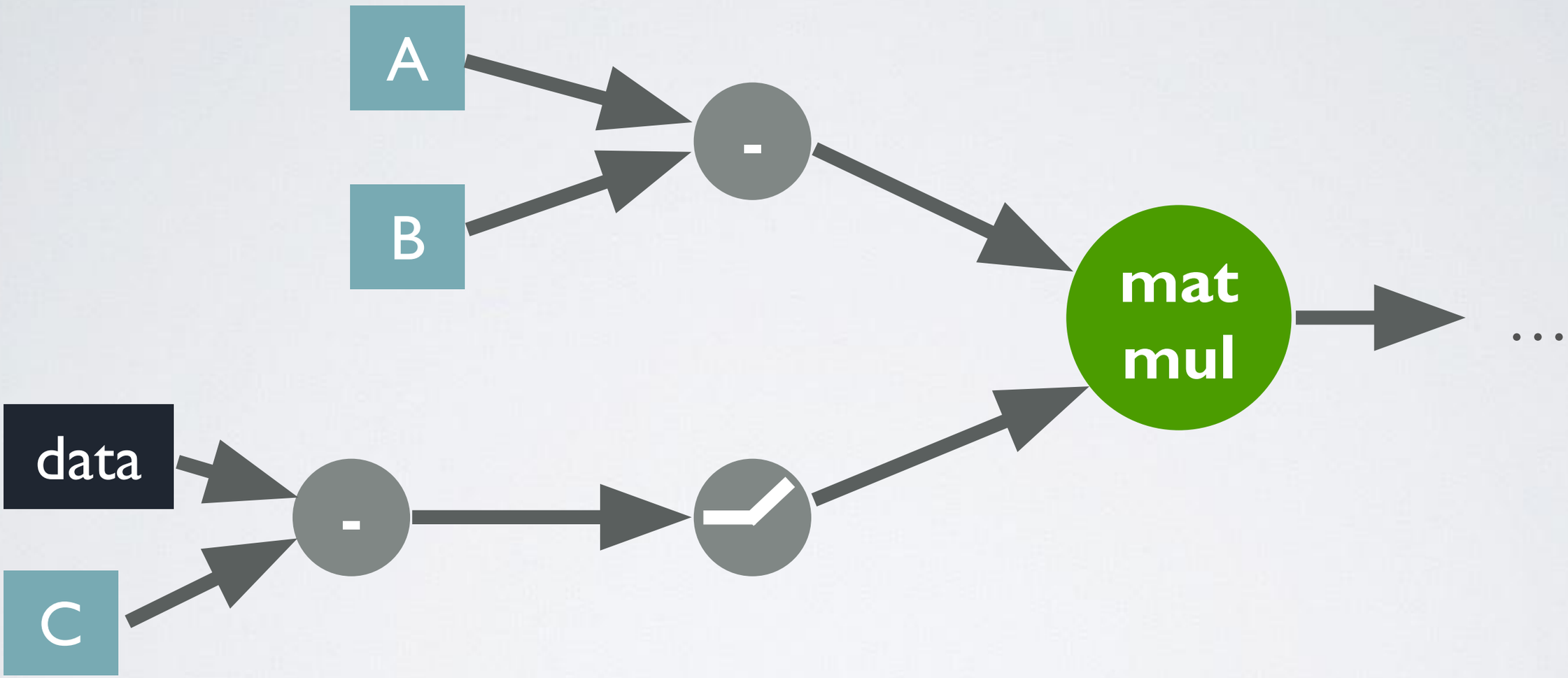
backward



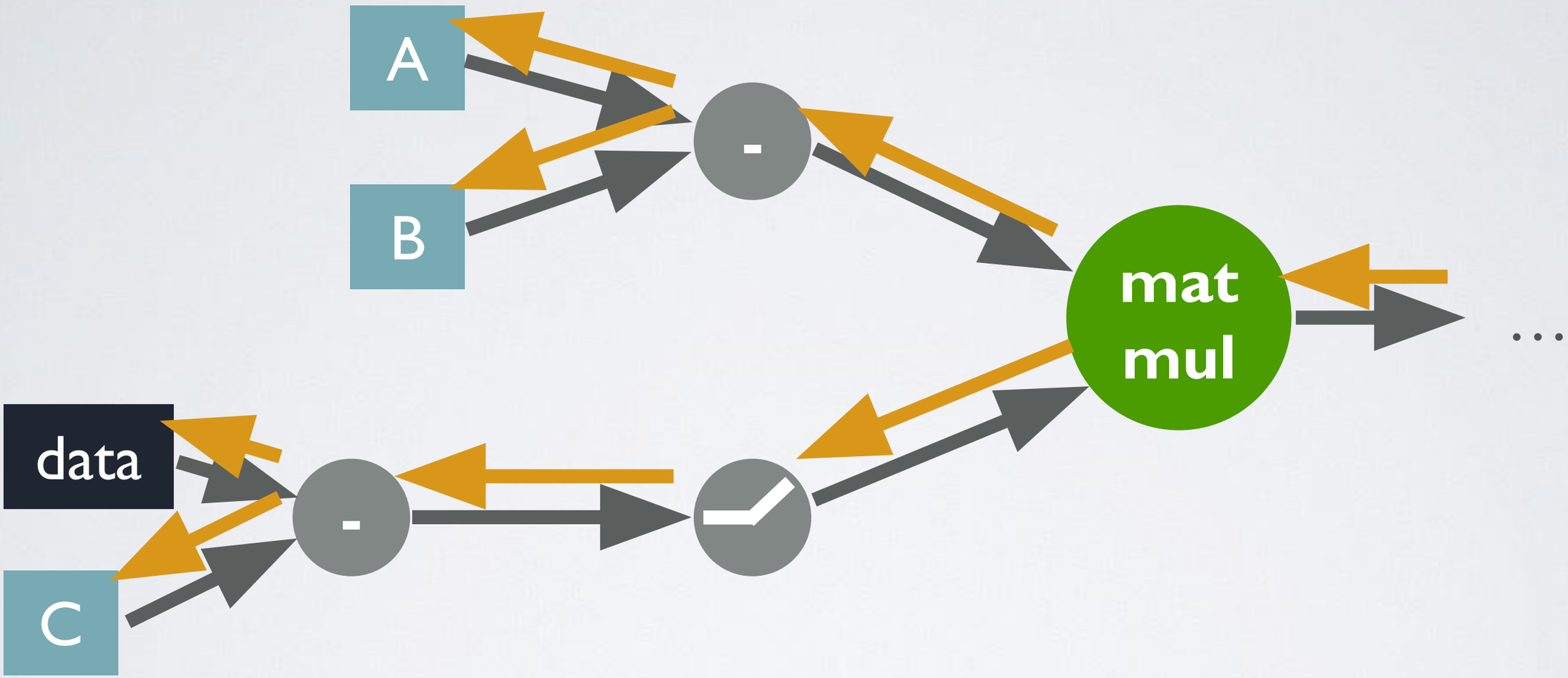
/



DEEP NETWORK IN GENERAL



DEEP NETWORK IN GENERAL



A

B

-

mat
mul

…

data

C

-

in
PyTorch
autograd

∇
A

∇B

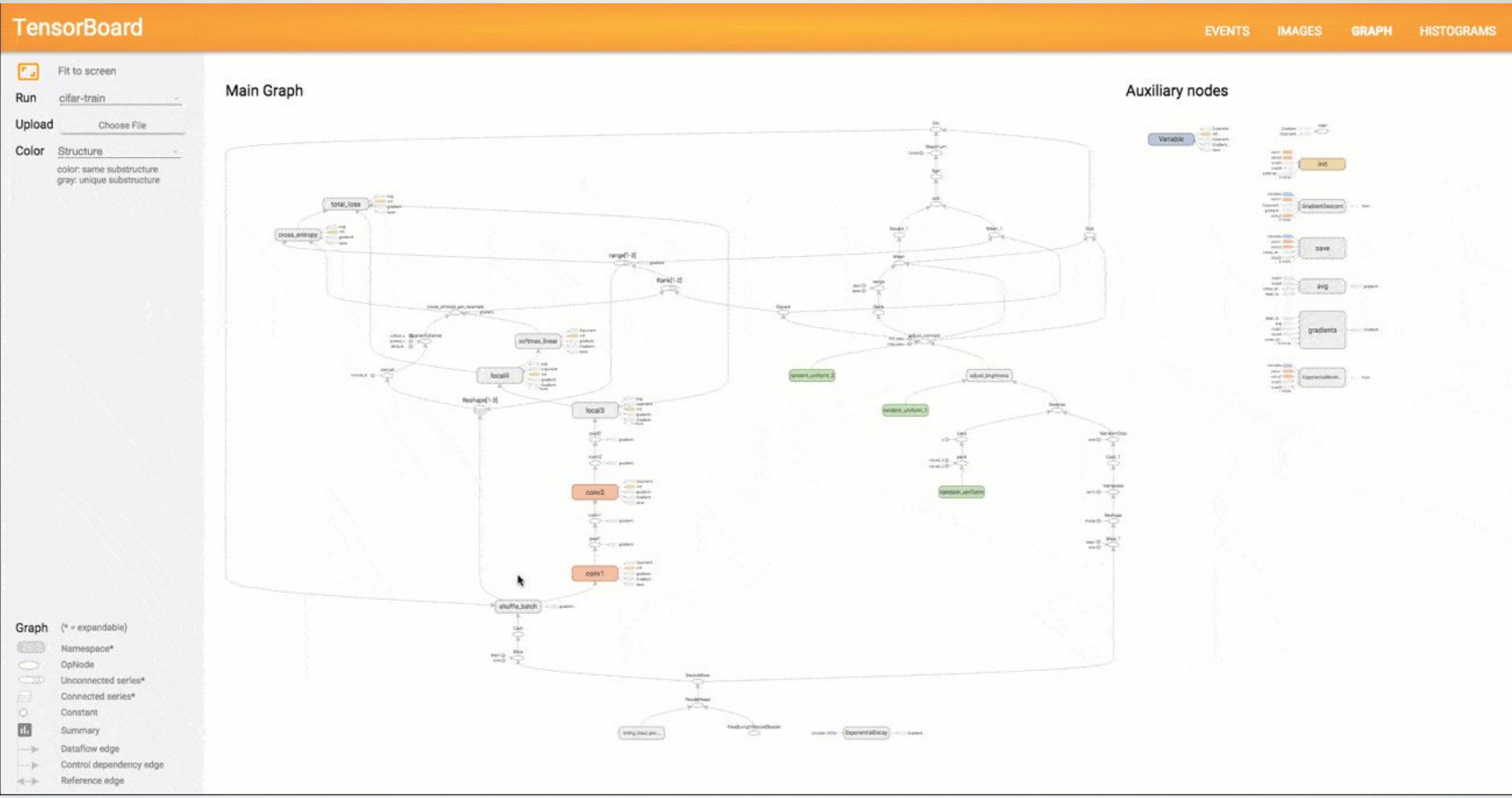
∇
-

∇ma
tmul

∇dat
a

∇
C

∇
-



In PyTorch, need additional plugins to visualize compute graph

credit:
tensorflow.org

DEEP NETWORK

dog vs **cat**

$$\ell(z_4)$$

Non-linearity

$$z_4 = f_4(z_3)$$

Linear transformation W

$$z_3 = f_3(z_2, W_3)$$

Non-linearity

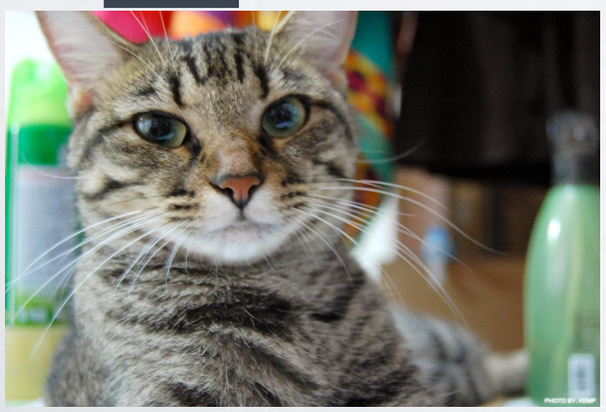
$$z_2 = f_2(z_1)$$

Linear transformation W

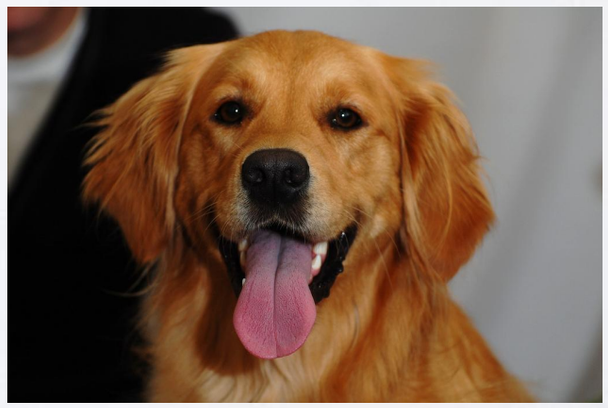
$$z_1 = f_1(x, W_1)$$

forward

backward



/



RECAP

- Loss function

  - How well are fitting the data?

- Gradient computation

  - What **local** change to the weights decreases the loss?

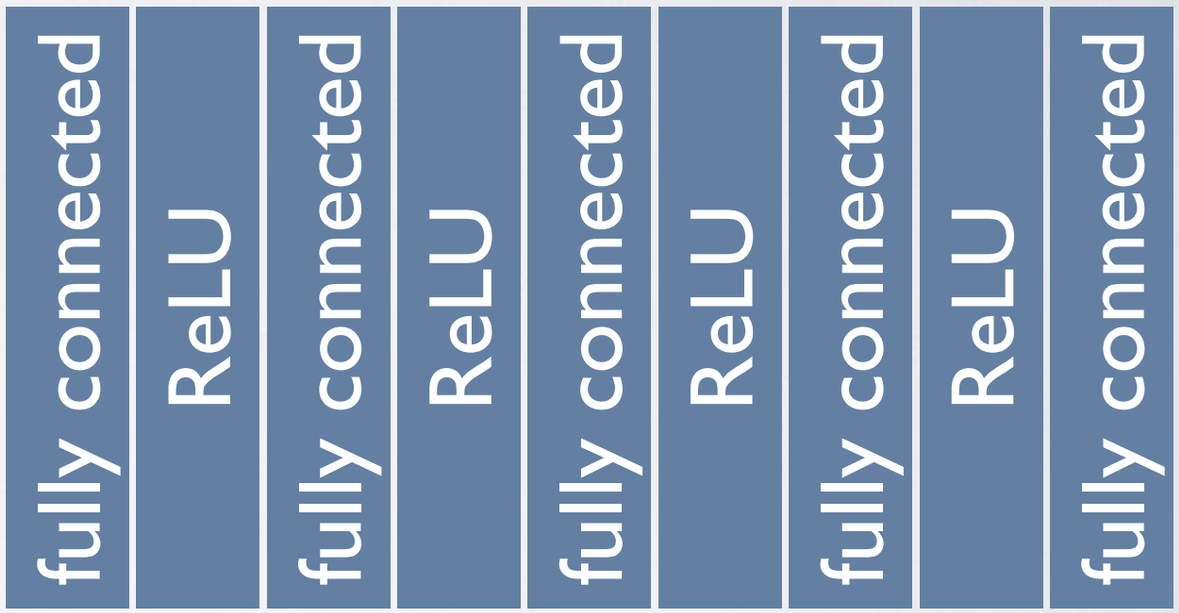
- Optimization algorithm

  - How do we change the weights to **globally** minimize the loss?

Last Thu

Today

Th



NEXT CLASS

dog vs **cat**

$\ell(z_4)$

Non-linearity $z_4 = f_4(z_3)$

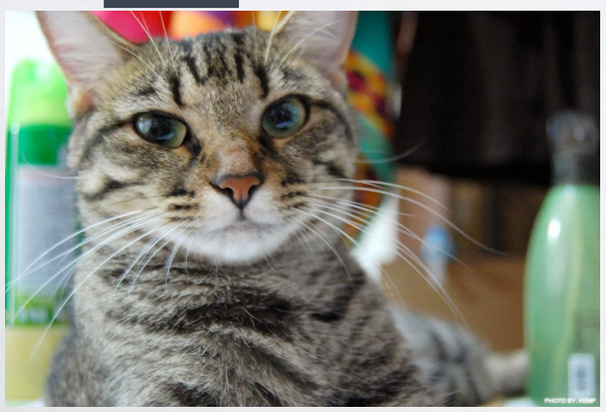
Linear transformation W $z_3 = f_3(z_2, W_3)$

Non-linearity $z_2 = f_2(z_1)$

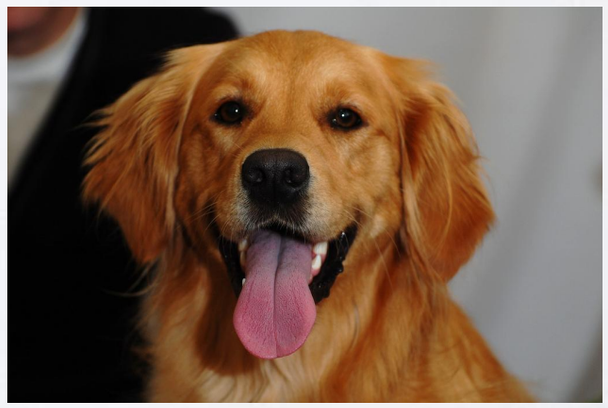
Linear transformation W $z_1 = f_1(x, W_1)$

forward

backward



/



NEXT WEEK

dog vs **cat**

backward

forward

Non-linearity

Linear transformation W

Non-linearity

Linear transformation W

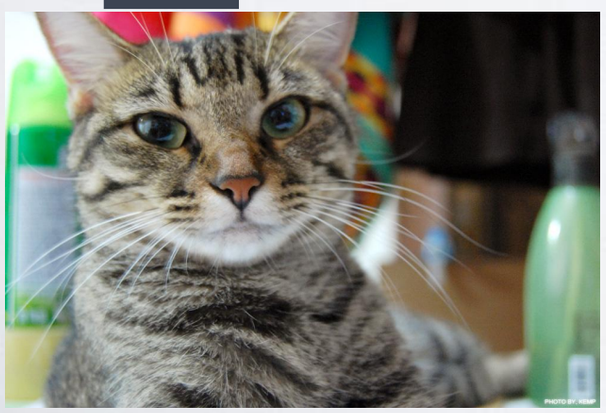
$$\ell(z_4)$$

$$z_4 = f_4(z_3)$$

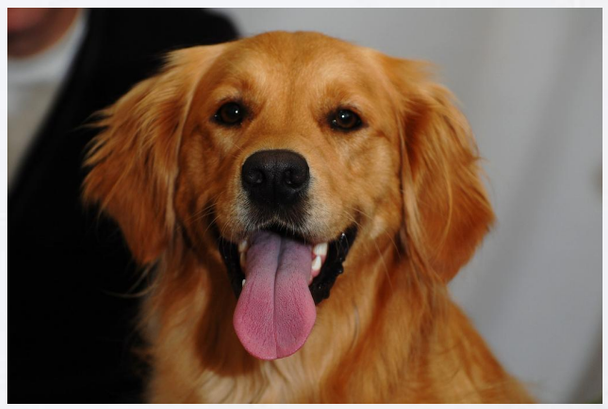
$$z_3 = f_3(z_2, W_3)$$

$$z_2 = f_2(z_1)$$

$$z_1 = f_1(x, W_1)$$



/



NEXT WEEK

dog vs **cat**

$$\ell(z_4)$$

Non-linearity $z_4 = f_4(z_3)$

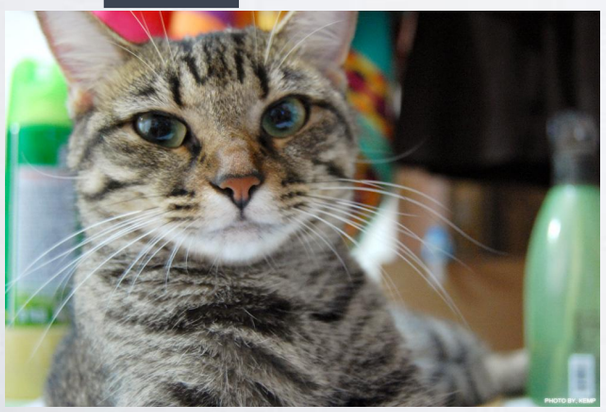
Linear transformation W $z_3 = f_3(z_2, W_3)$

Non-linearity $z_2 = f_2(z_1)$

Linear transformation W $z_1 = f_1(x, W_1)$

forward

backward



/