
Introducción a las Redes Neuronales con
PYTORCH

HELLO!

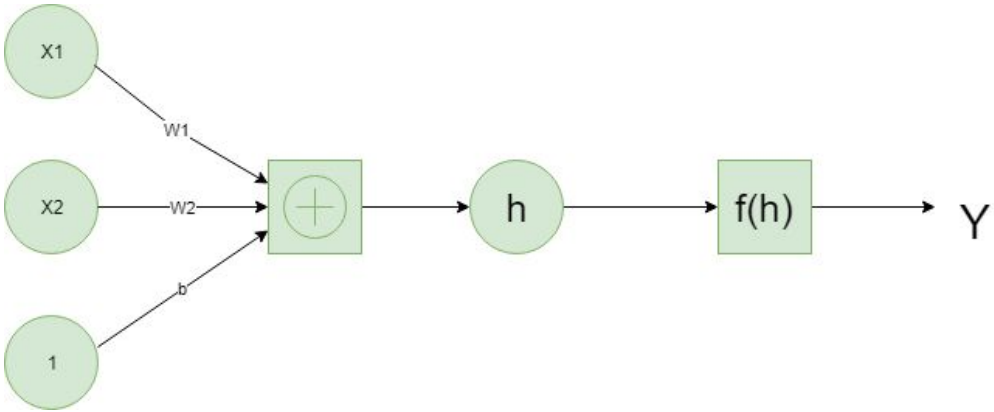
Linkedin

calasius@gmail.com

Agenda

- MLP multiplicando matrices
- Clasificador dígitos MNIST
- Inferencia y validación
- Leer y salvar modelos
- Transfer learning convolucionales
- redes neuronales en pytorch
- Clasificador Fashion MNIST
- Dropout
- carga de imágenes
- Que aprenden los filtros de las redes

Simple neuron



$$y = f(w_1 x_1 + w_2 x_2 + b)$$

$$y = f\left(\sum_i w_i x_i + b\right)$$

$$h = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

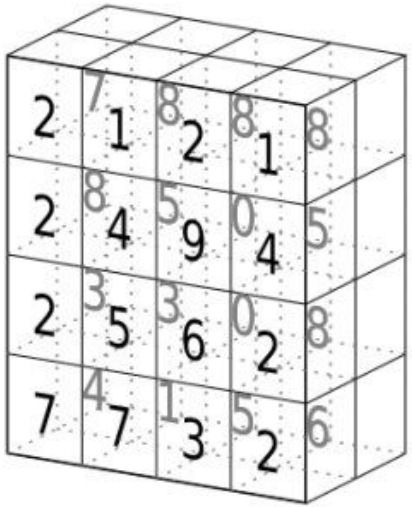
Tensores

't'
'e'
'n'
's'
'o'
'r'

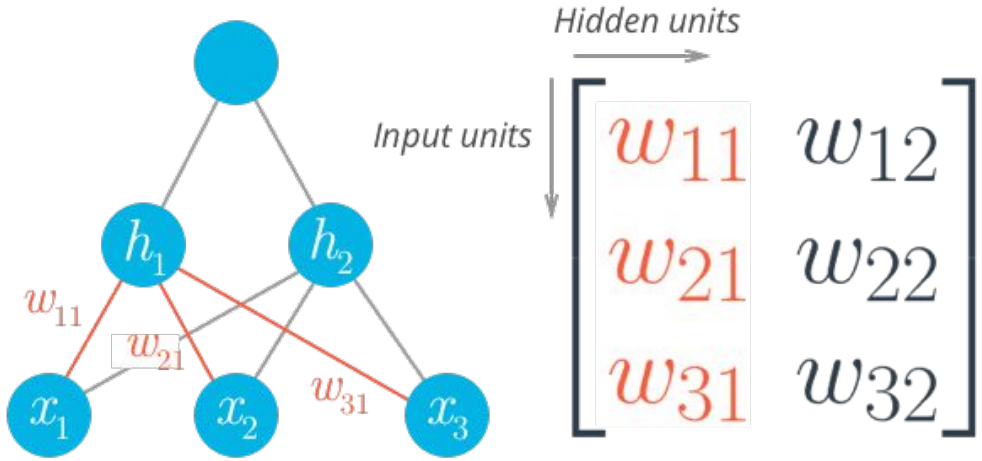
tensor of dimensions [6]
(vector of dimension 6)

3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3
2	3	8	4
6	2	6	4

tensor of dimensions [6,4]
(matrix 6 by 4)



tensor of dimensions [4,4,2]



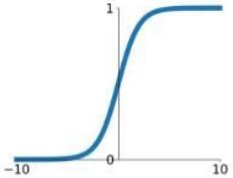
$$\vec{h} = [h_1\ h_2] = \begin{bmatrix} x_1\ x_2 \cdots\ x_n \end{bmatrix} \cdot \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ \vdots & \vdots \\ w_{n1} & w_{n2} \end{bmatrix}$$

$$y = f_2\big(\, f_1\big(\vec{x}\,\mathbf{W_1}\big)\,\mathbf{W_2}\big)$$

Activation Functions

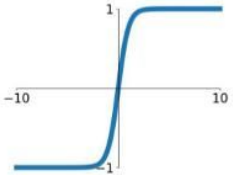
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



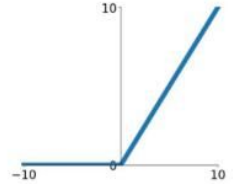
tanh

$$\tanh(x)$$



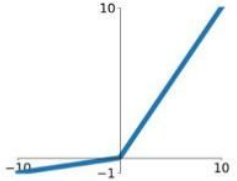
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

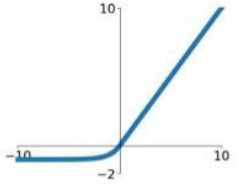


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

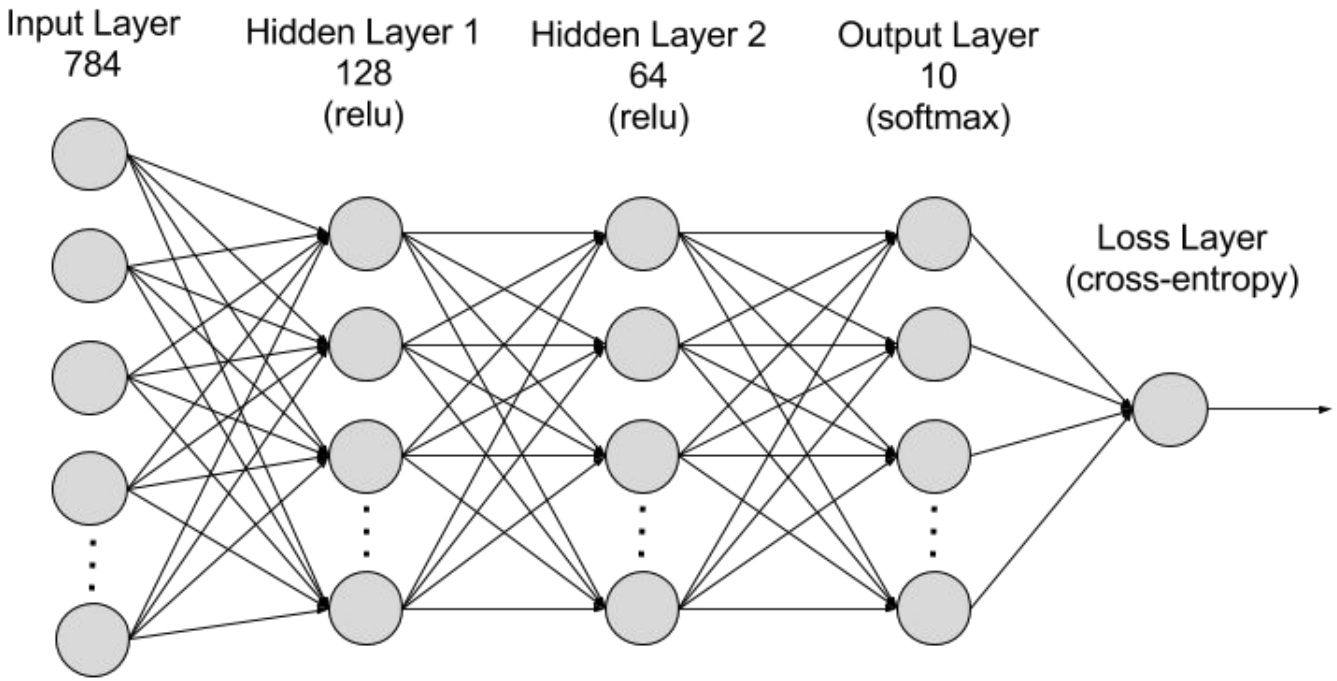
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

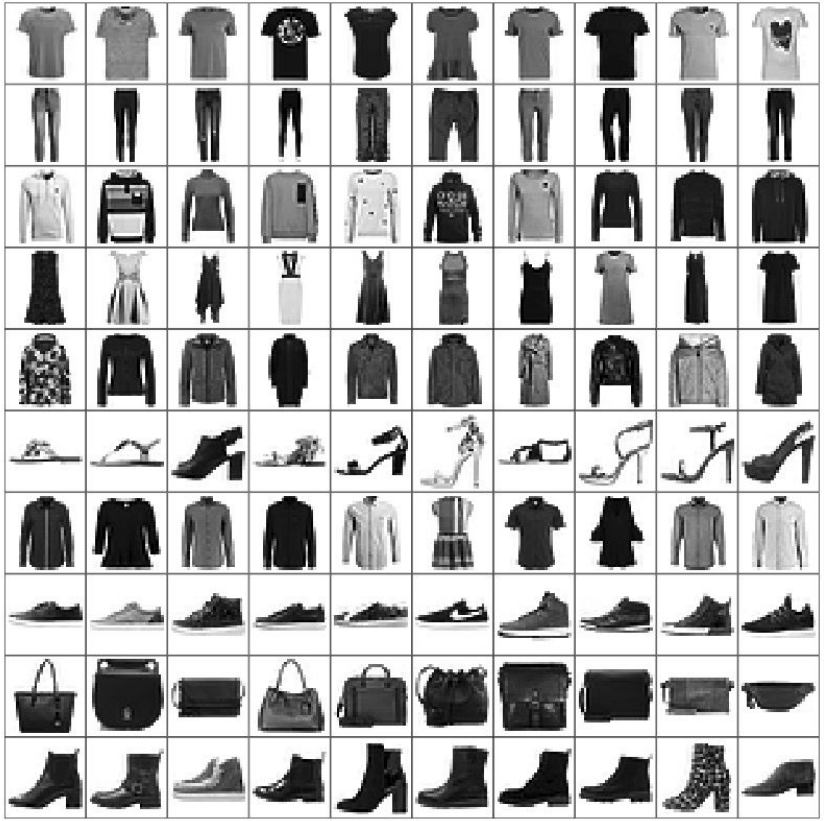


MNIST dataset

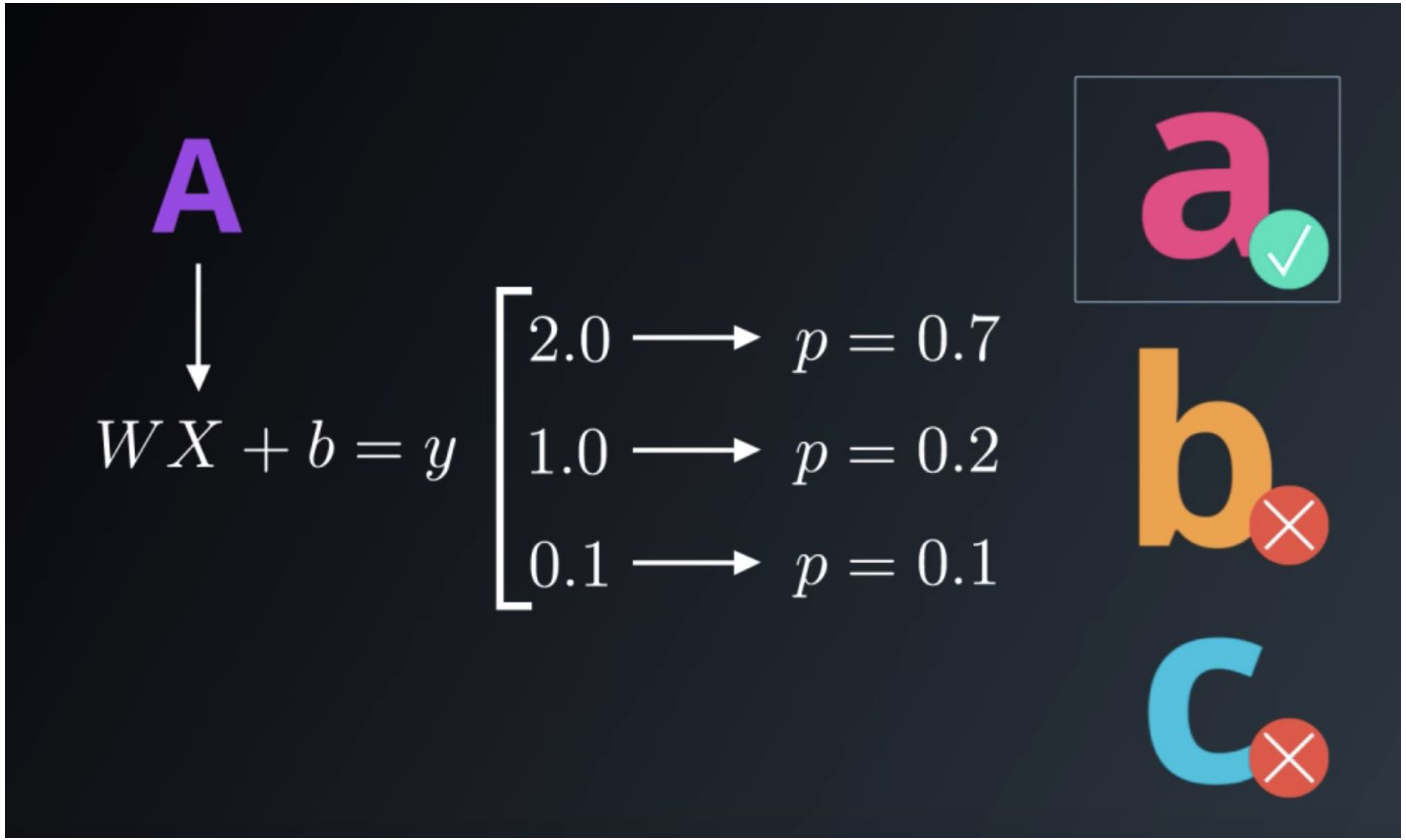
MLP MNIST


Input Layer
784
Hidden Layer 1
128
(relu)
Hidden Layer 2
64
(relu)
Output Layer
10
(softmax)
Loss Layer
(cross-entropy)

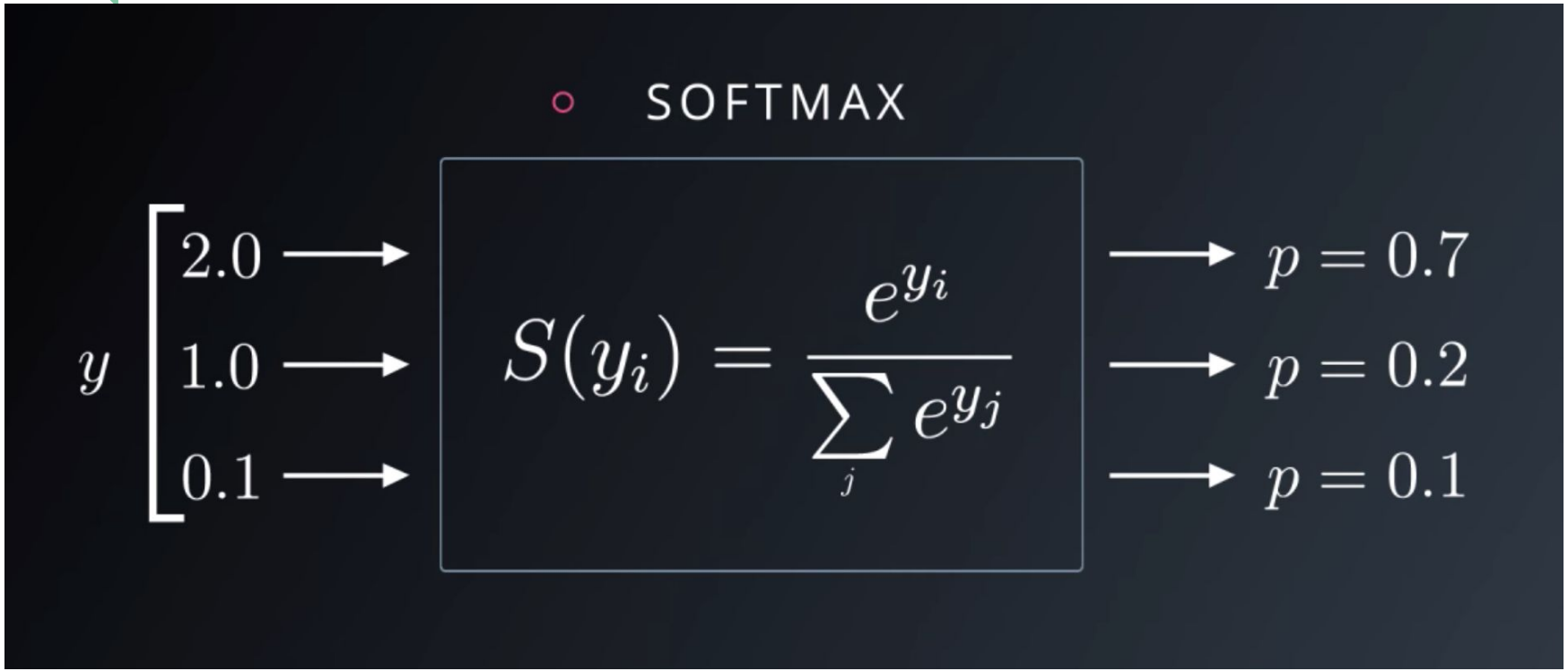
Fashion MNIST dataset

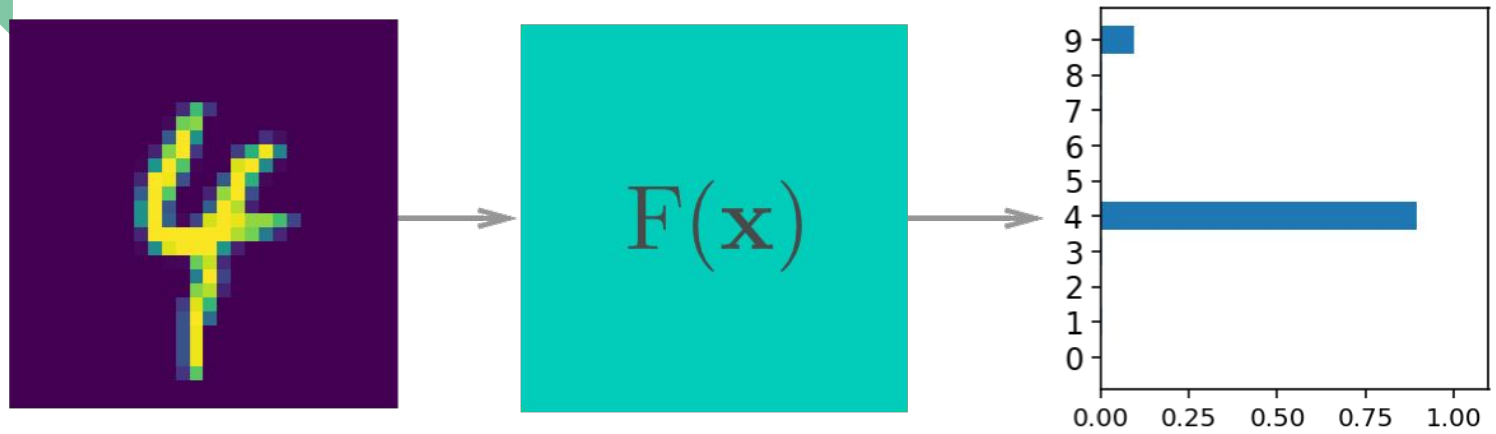


Softmax function



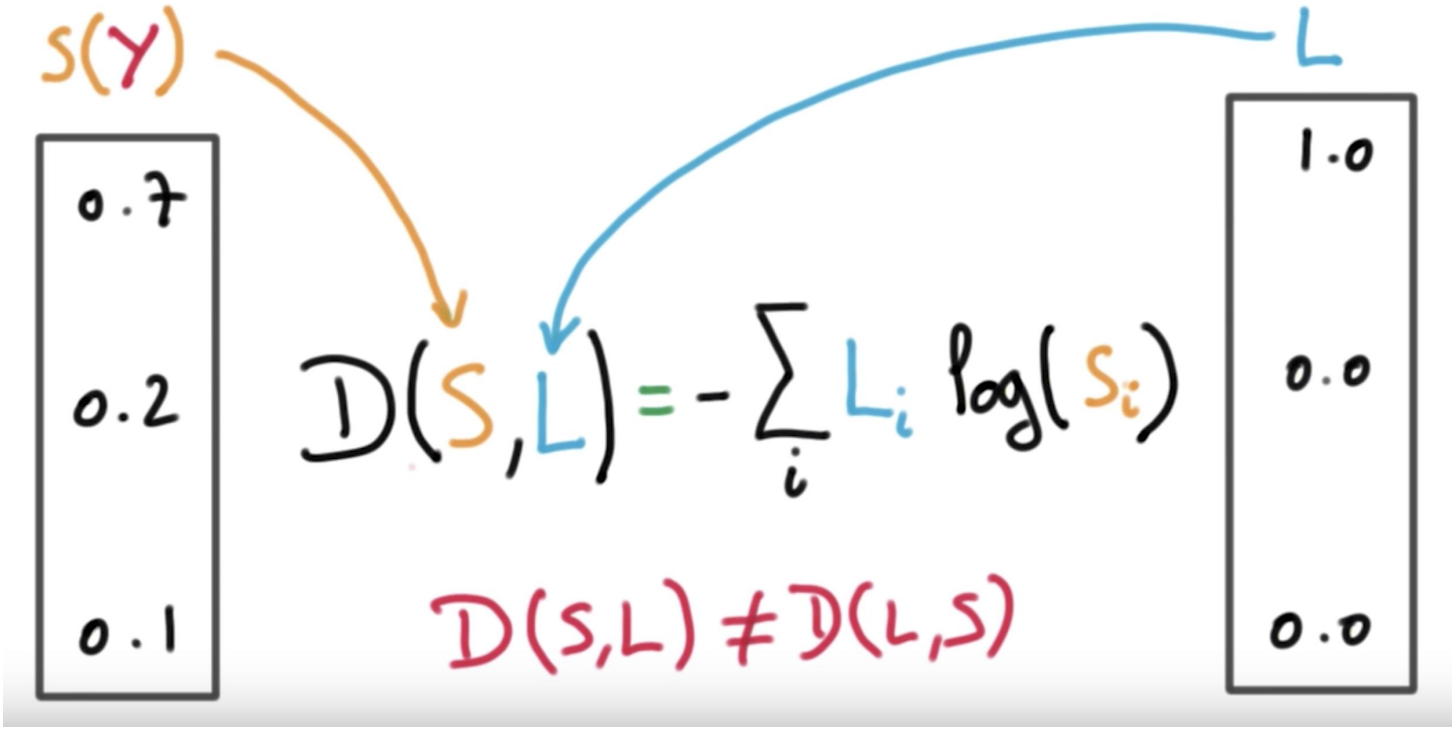
Softmax function

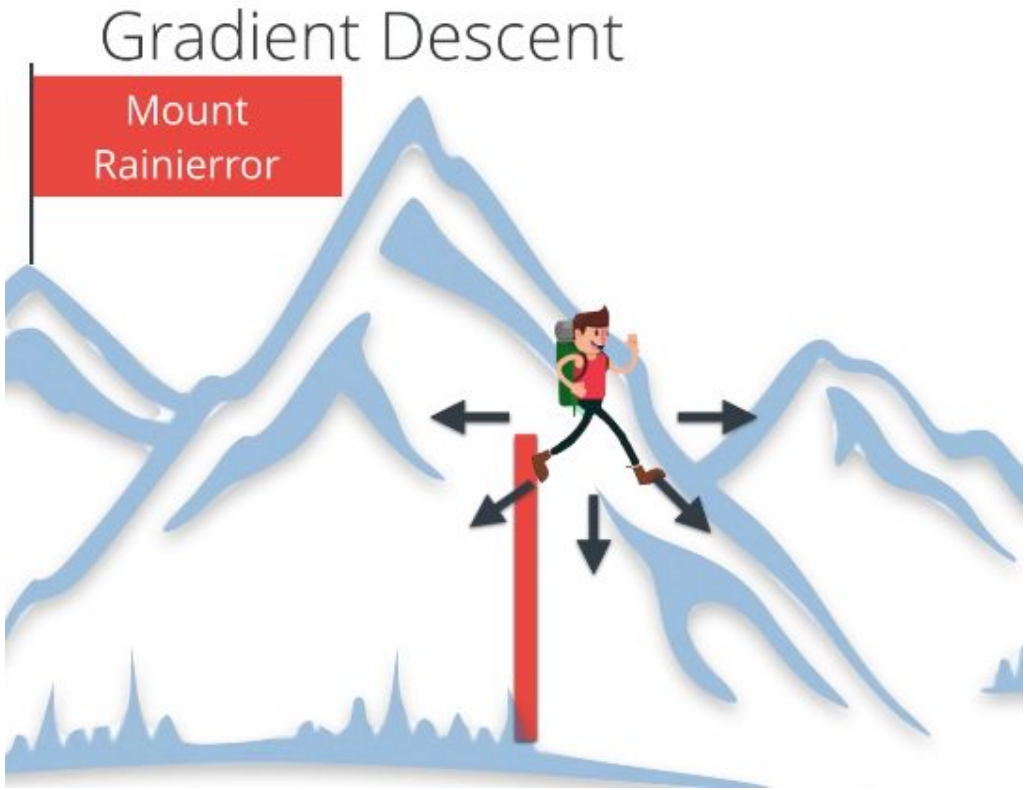


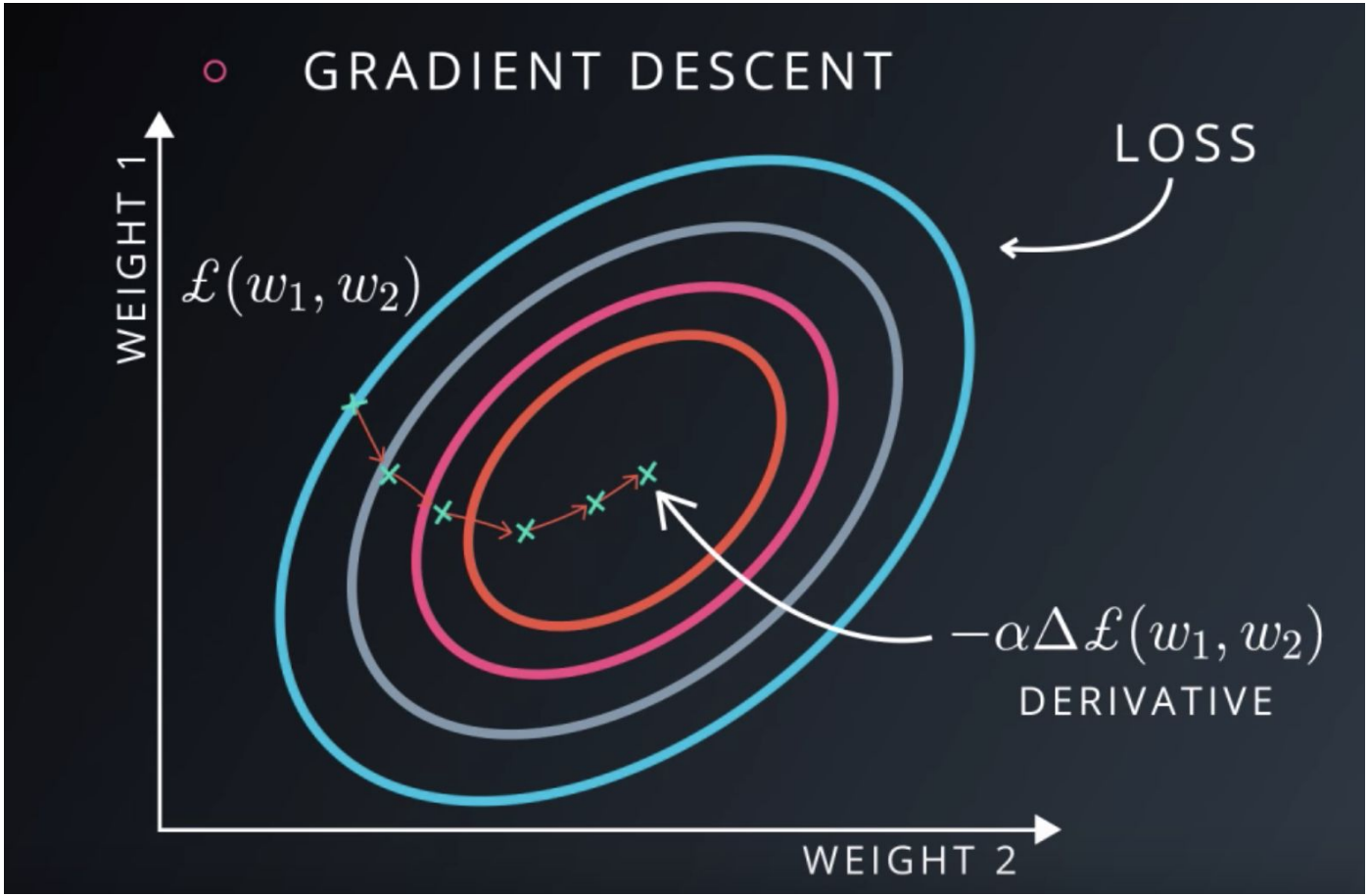


$$\ell = \frac{1}{2n} \sum_{i}^{n} \left(y_i - \hat{y}_i \right)^2$$

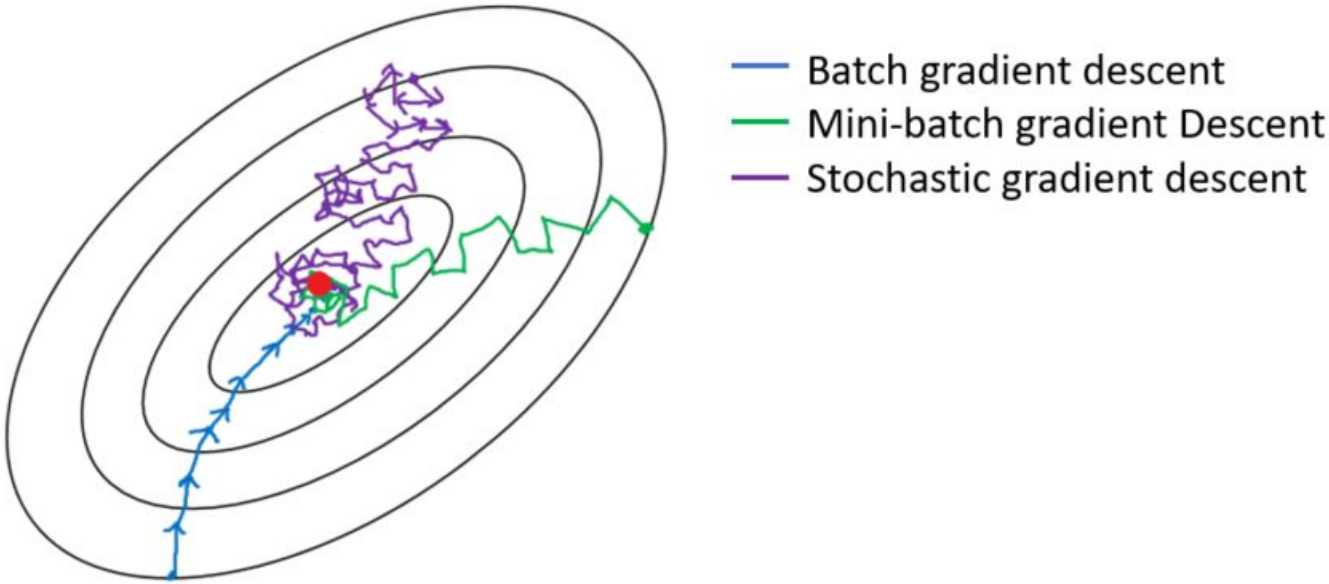
Cross entropy



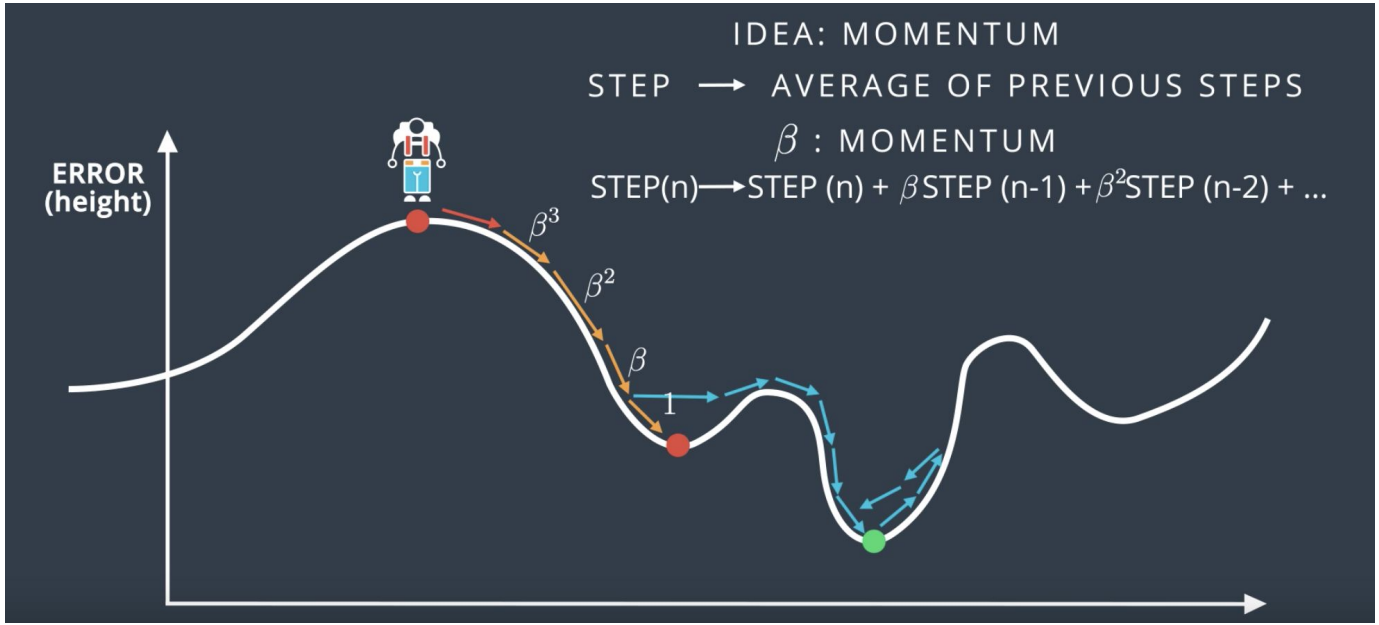

Gradient Descent
Mount
Rainierror


GRADIENT DESCENT
LOSS
WEIGHT 1
$\mathcal{L}(w_1, w_2)$
$-\alpha\Delta\mathcal{L}(w_1, w_2)$
DERIVATIVE
WEIGHT 2

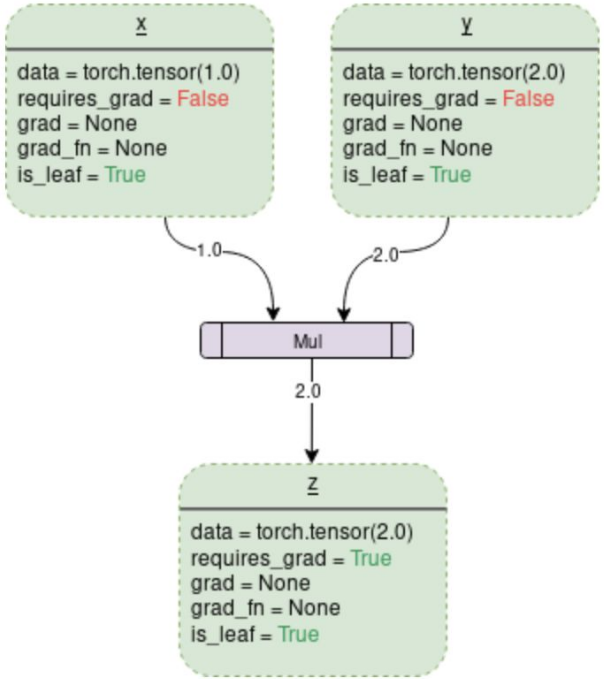
Comparación de los métodos

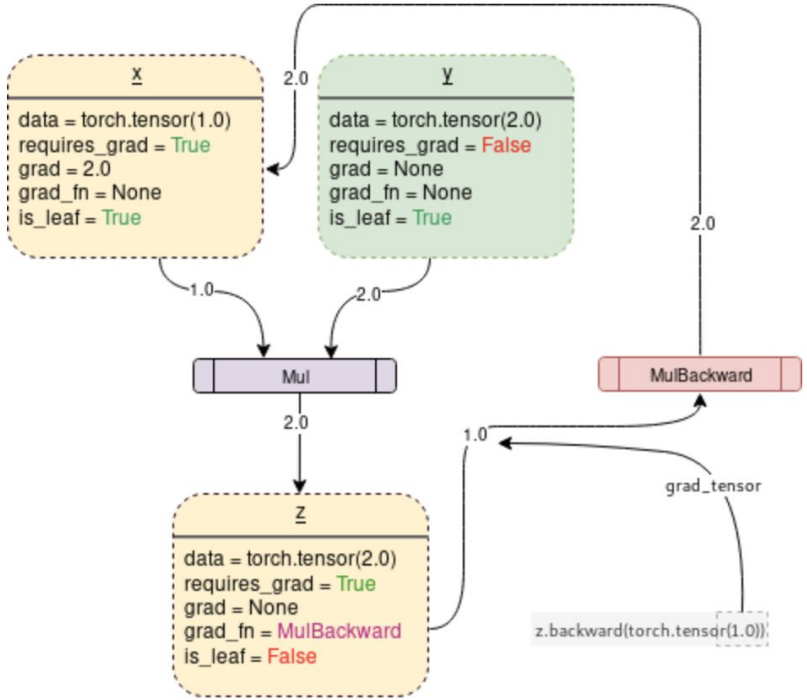


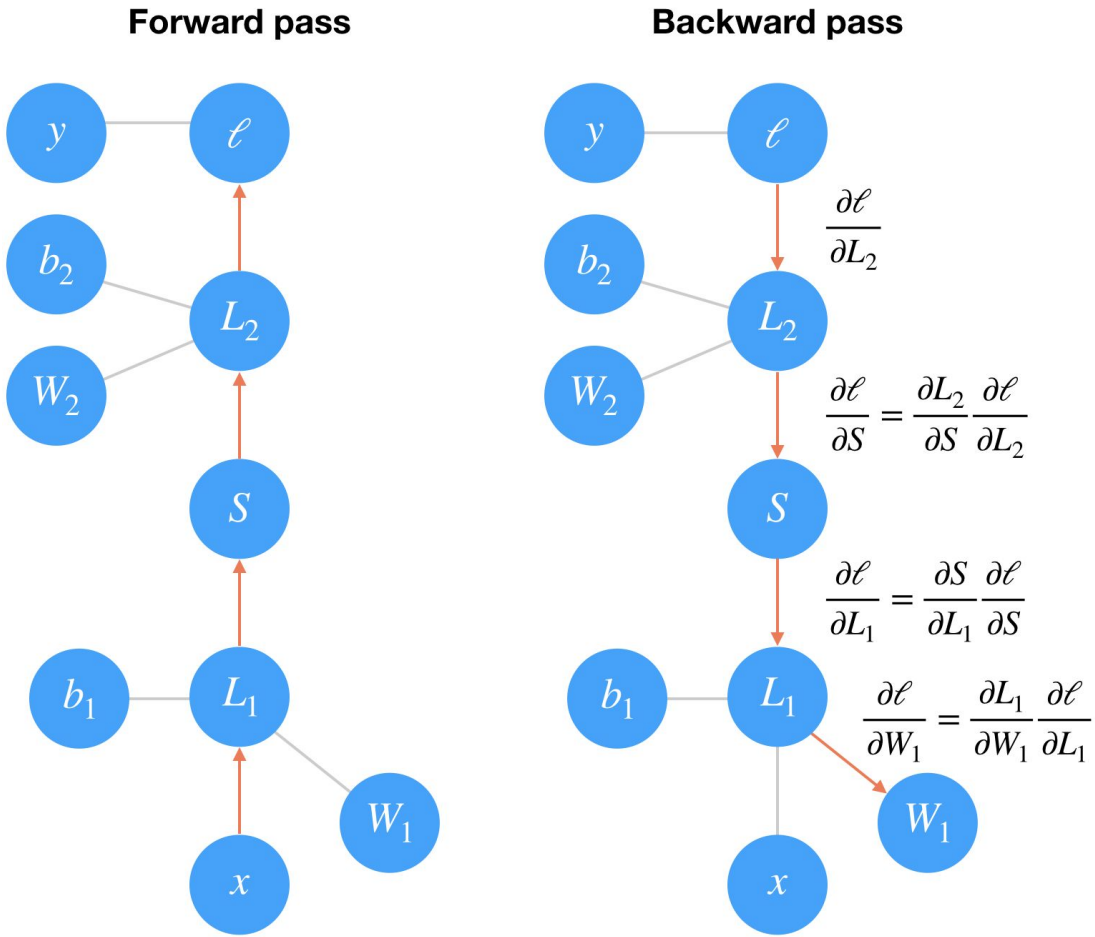
Momentum


IDEA: MOMENTUM
STEP ⟶ AVERAGE OF PREVIOUS STEPS
β : MOMENTUM
STEP(n)⟶STEP (n) + βSTEP (n-1) +β²STEP (n-2) + ...
ERROR
(height)
β³
β²
β
1

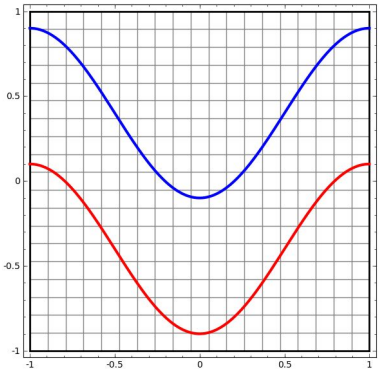
Autograd

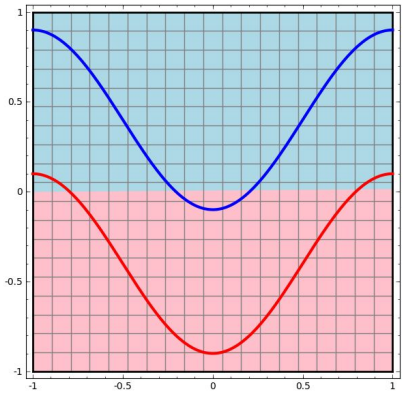

x
data = torch.tensor(1.0)
requires_grad = False
grad = None
grad_fn = None
is_leaf = True
y
data = torch.tensor(2.0)
requires_grad = False
grad = None
grad_fn = None
is_leaf = True
1.0
2.0
Mul
2.0
z
data = torch.tensor(2.0)
requires_grad = True
grad = None
grad_fn = None
is_leaf = True

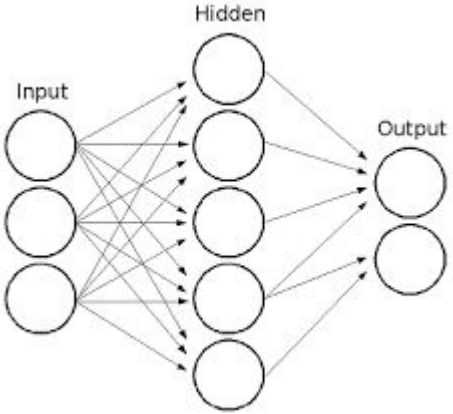

x
data = torch.tensor(1.0)
requires_grad = True
grad = 2.0
grad_fn = None
is_leaf = True
2.0
y
data = torch.tensor(2.0)
requires_grad = False
grad = None
grad_fn = None
is_leaf = True
2.0
1.0
2.0
Mul
2.0
MulBackward
1.0
grad_tensor
z
data = torch.tensor(2.0)
requires_grad = True
grad = None
grad_fn = MulBackward
is_leaf = False
z.backward(torch.tensor(1.0))

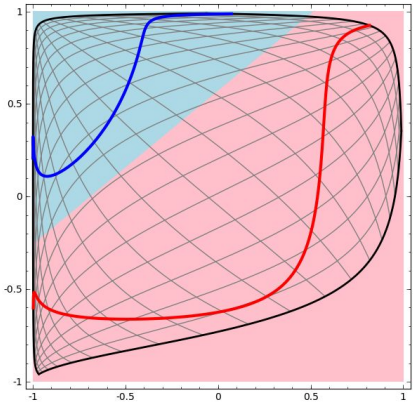

Forward pass
Backward pass
y
ℓ
b2
L2
W2
S
b1
L1
W1
x
∂ℓ/∂L2
∂ℓ/∂S = ∂L2/∂S ∂ℓ/∂L2
∂ℓ/∂L1 = ∂S/∂L1 ∂ℓ/∂S
∂ℓ/∂W1 = ∂L1/∂W1 ∂ℓ/∂L1

Que aprenden las capas ocultas

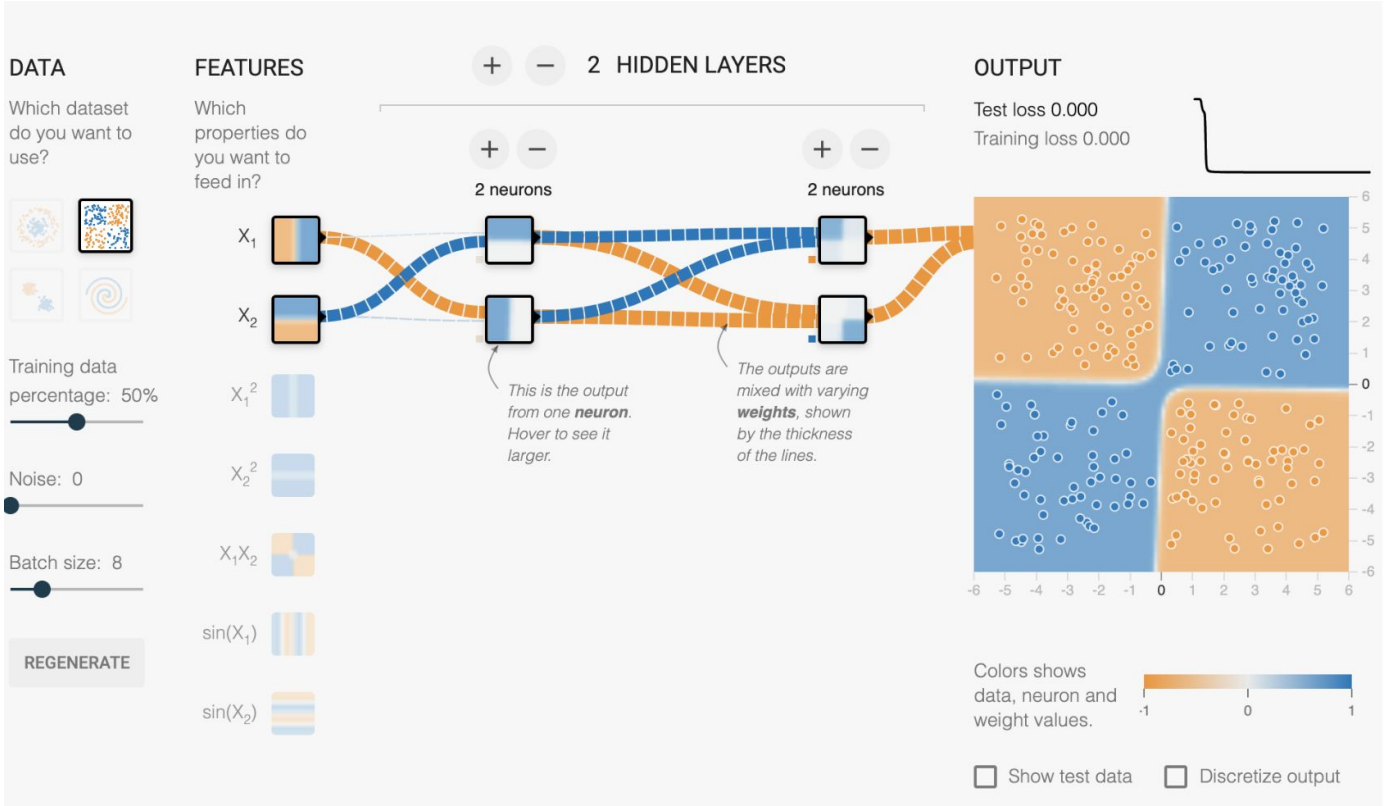




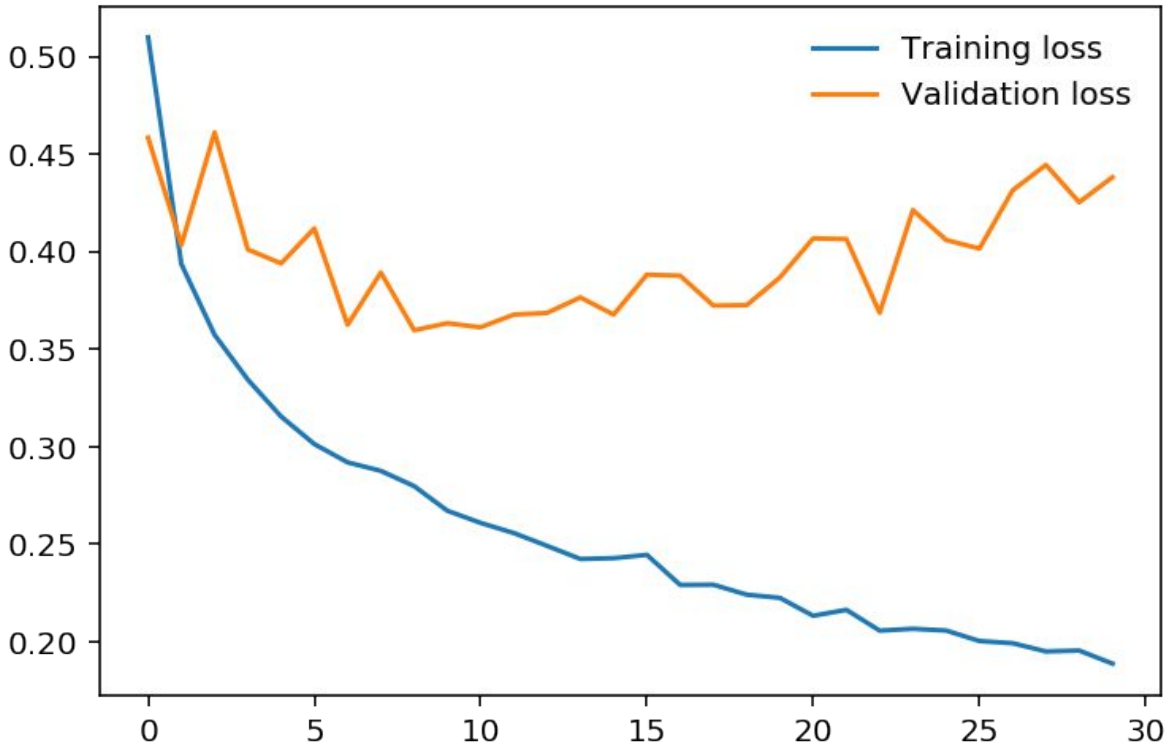




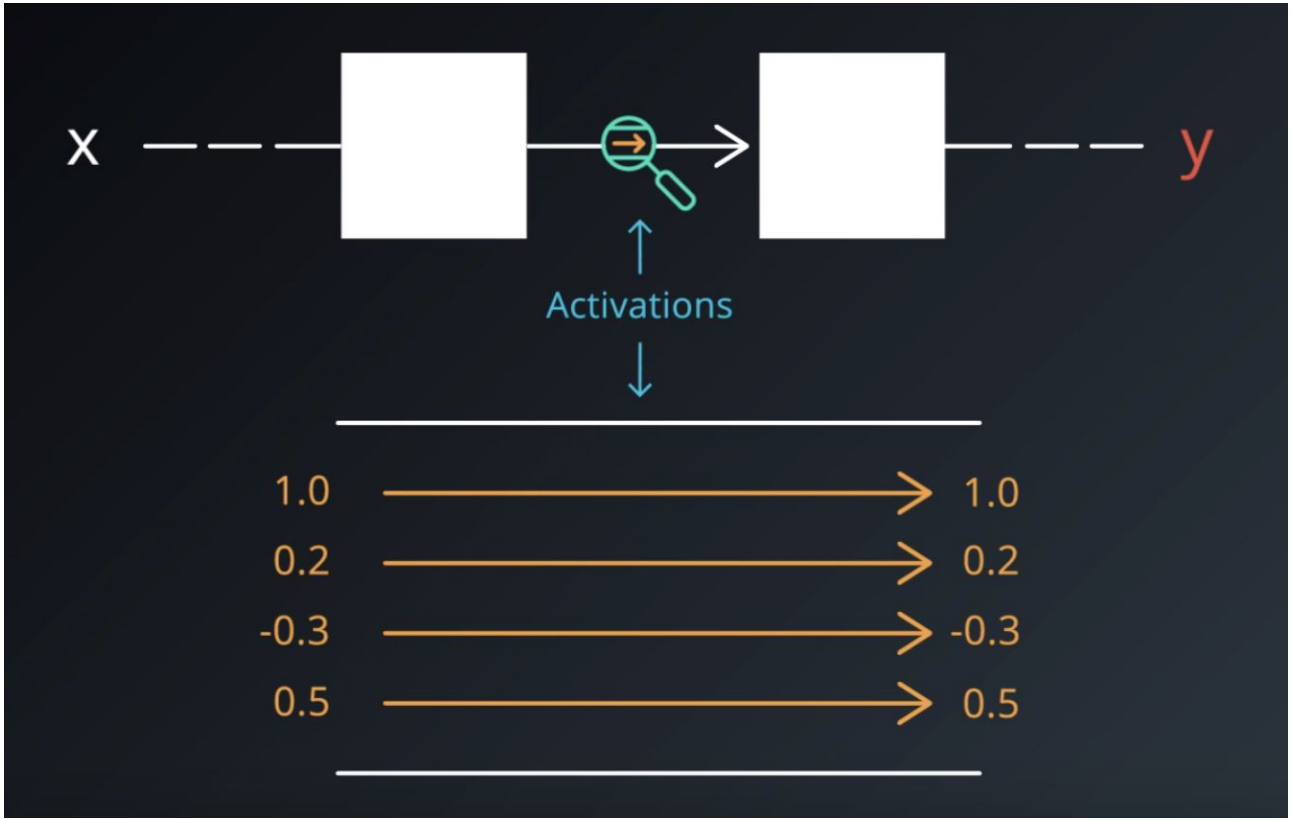
Neural network playground [link]



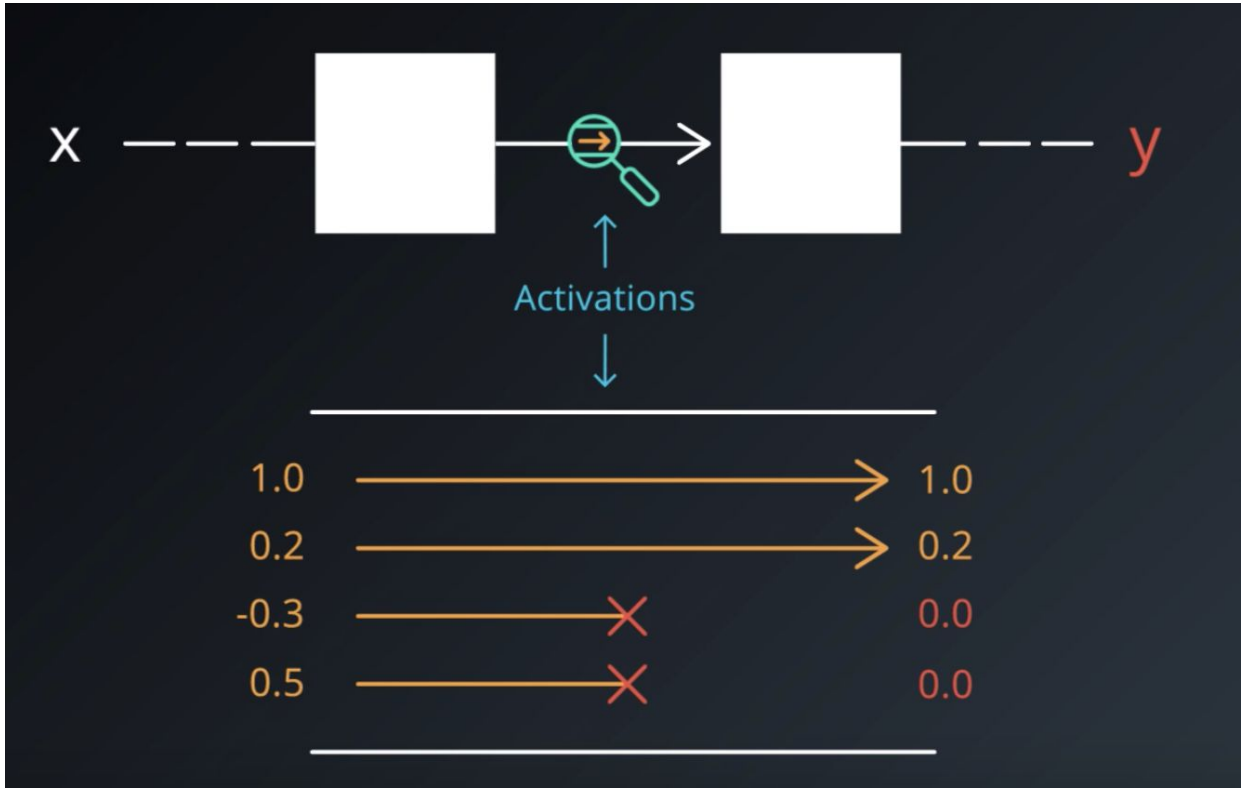
Overfitting



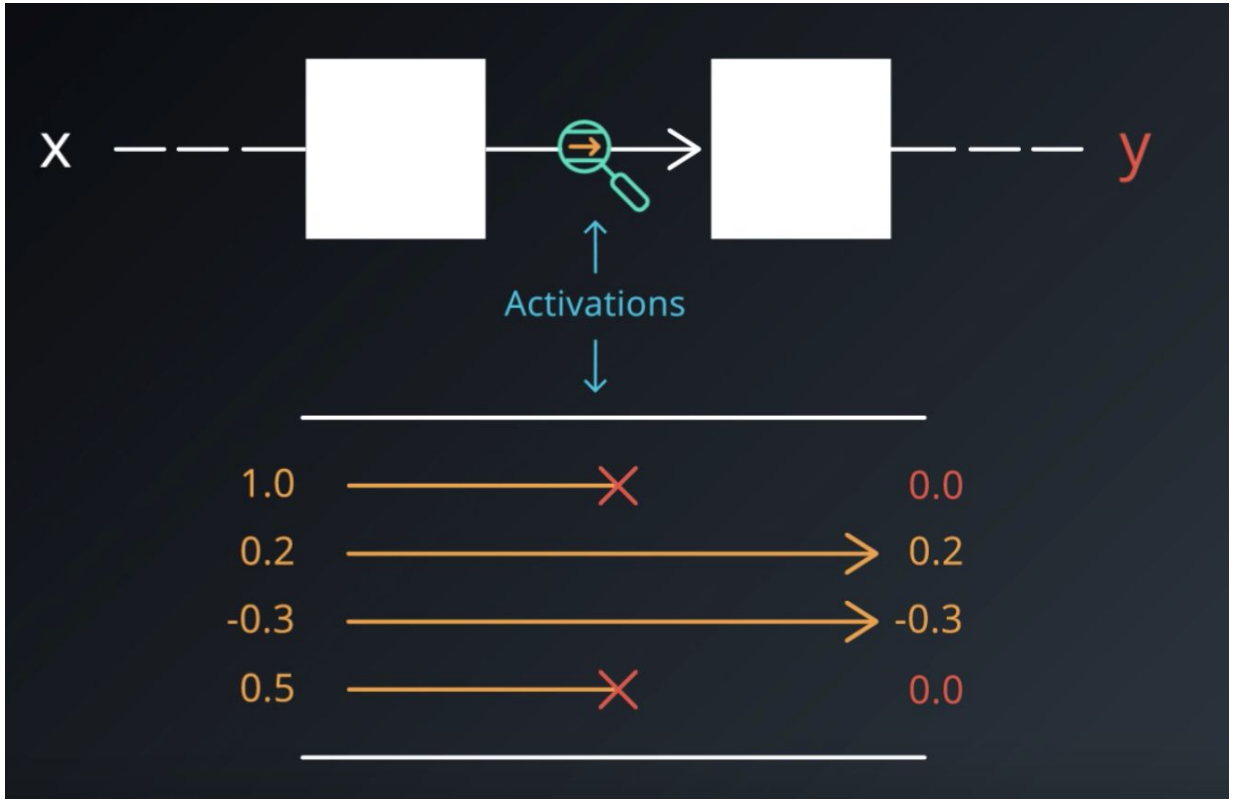
Regularización (Dropout)



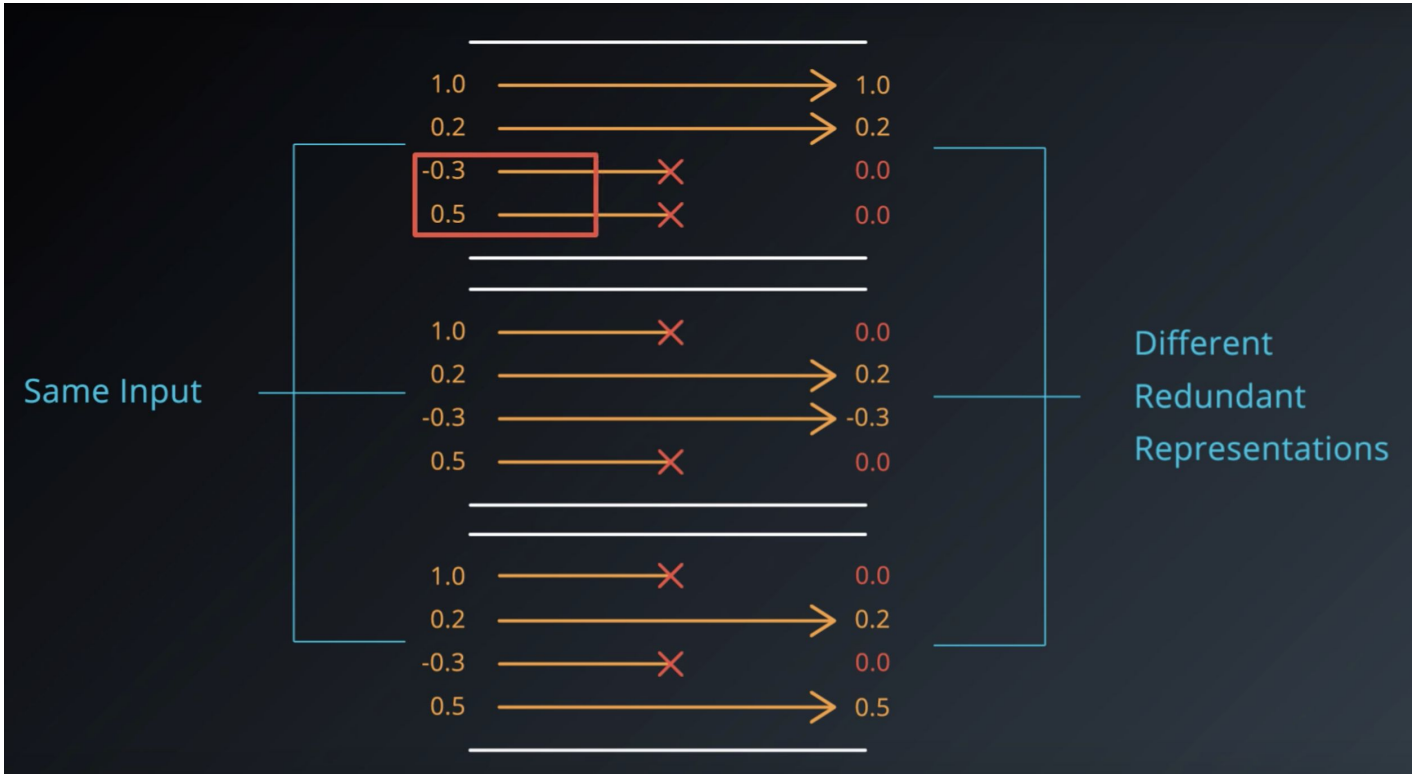
Dropout



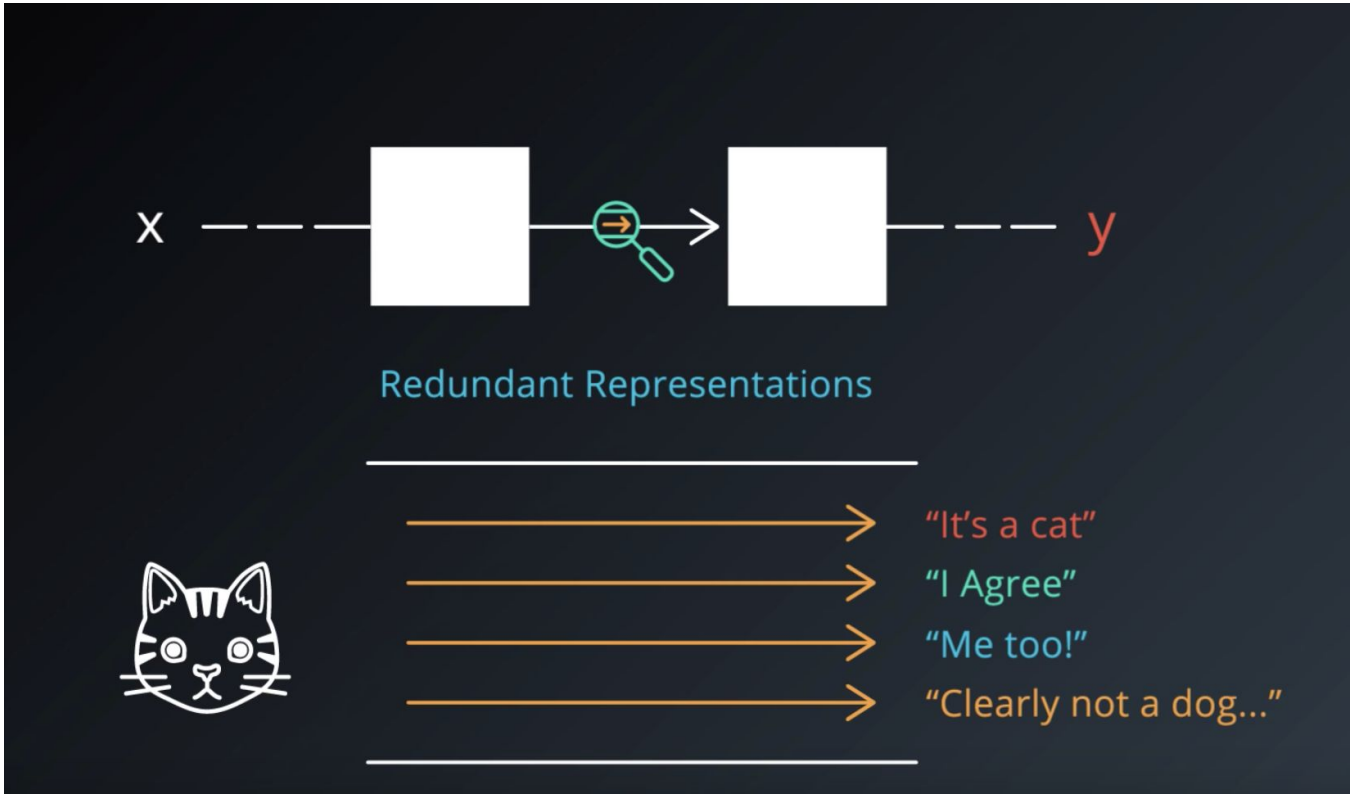
Dropout



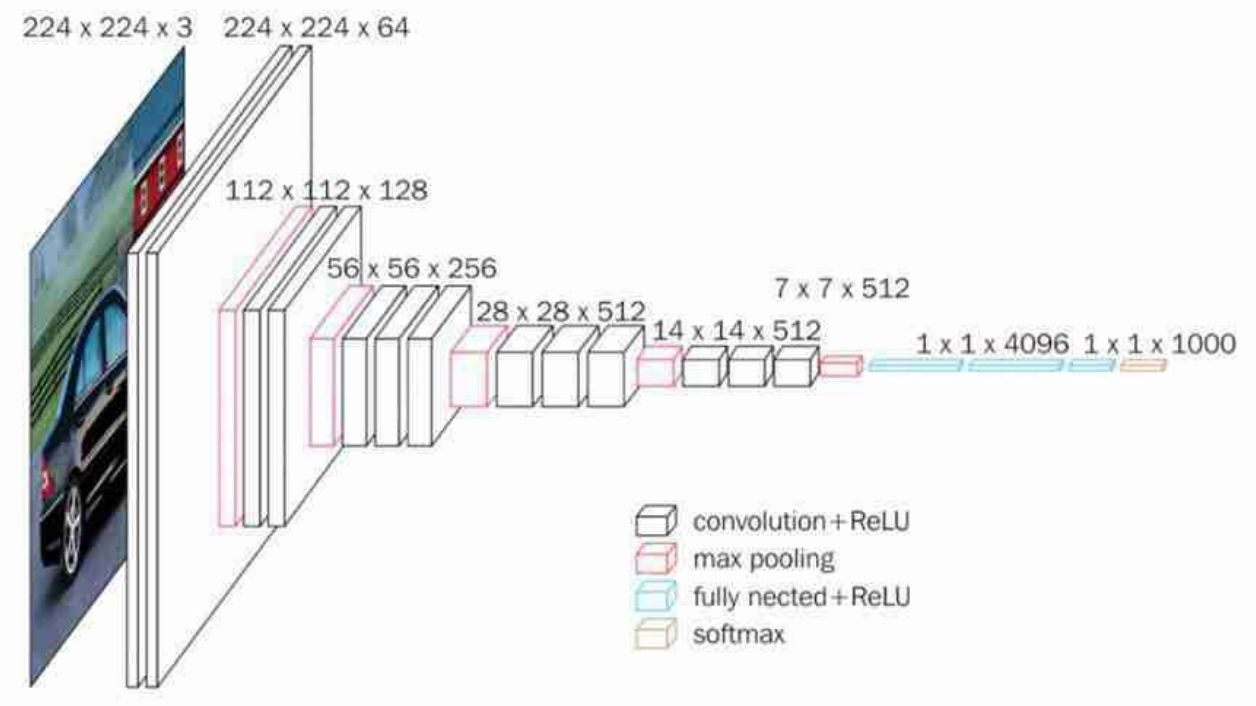
Dropout


1.0
0.2
-0.3
0.5
1.0
0.2
0.0
0.0
1.0
0.2
-0.3
0.5
0.0
0.2
-0.3
0.0
1.0
0.2
-0.3
0.5
0.0
0.2
0.0
0.5
Same Input
Different
Redundant
Representations

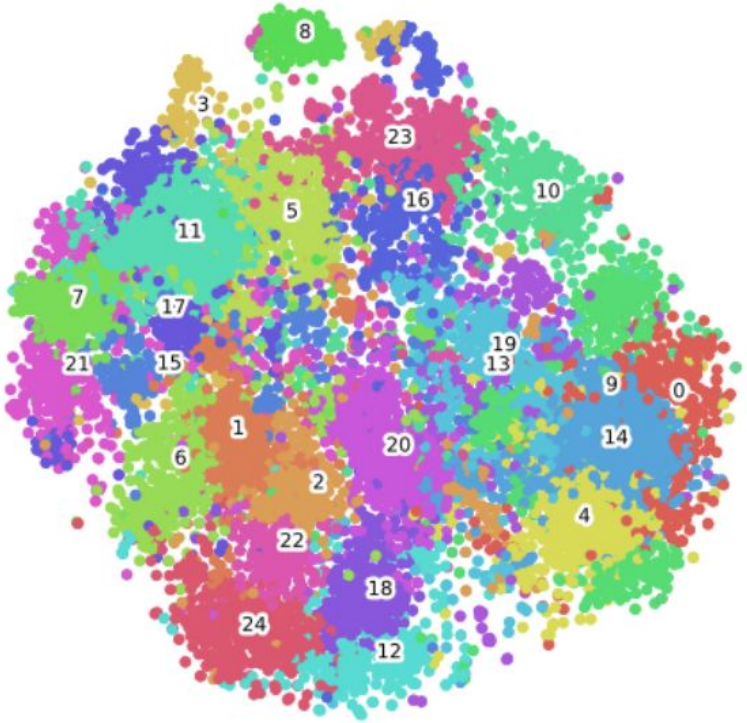
Dropout


x
y
Redundant Representations
"It's a cat"
"I Agree"
"Me too!"
"Clearly not a dog..."

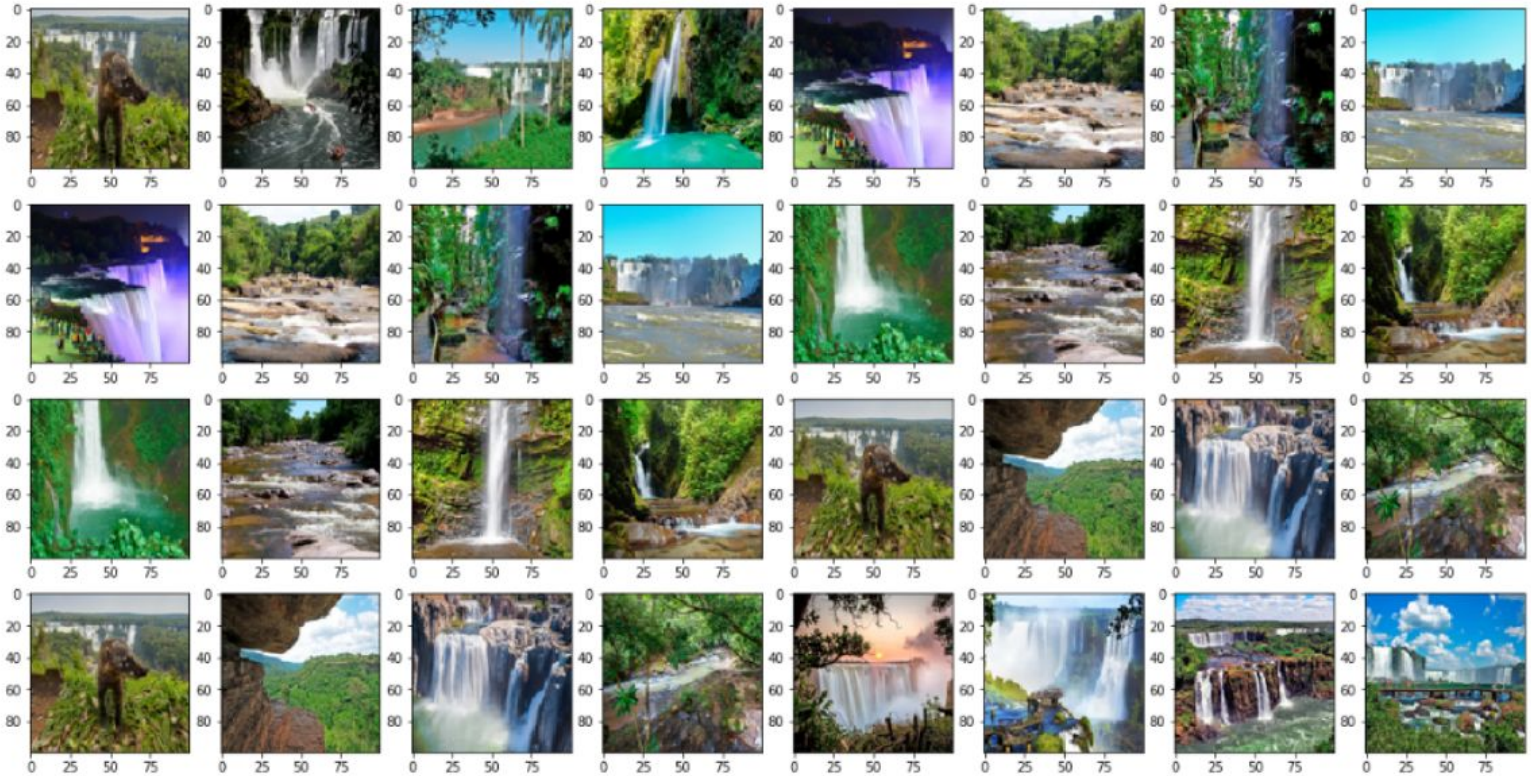
Transfer Learning



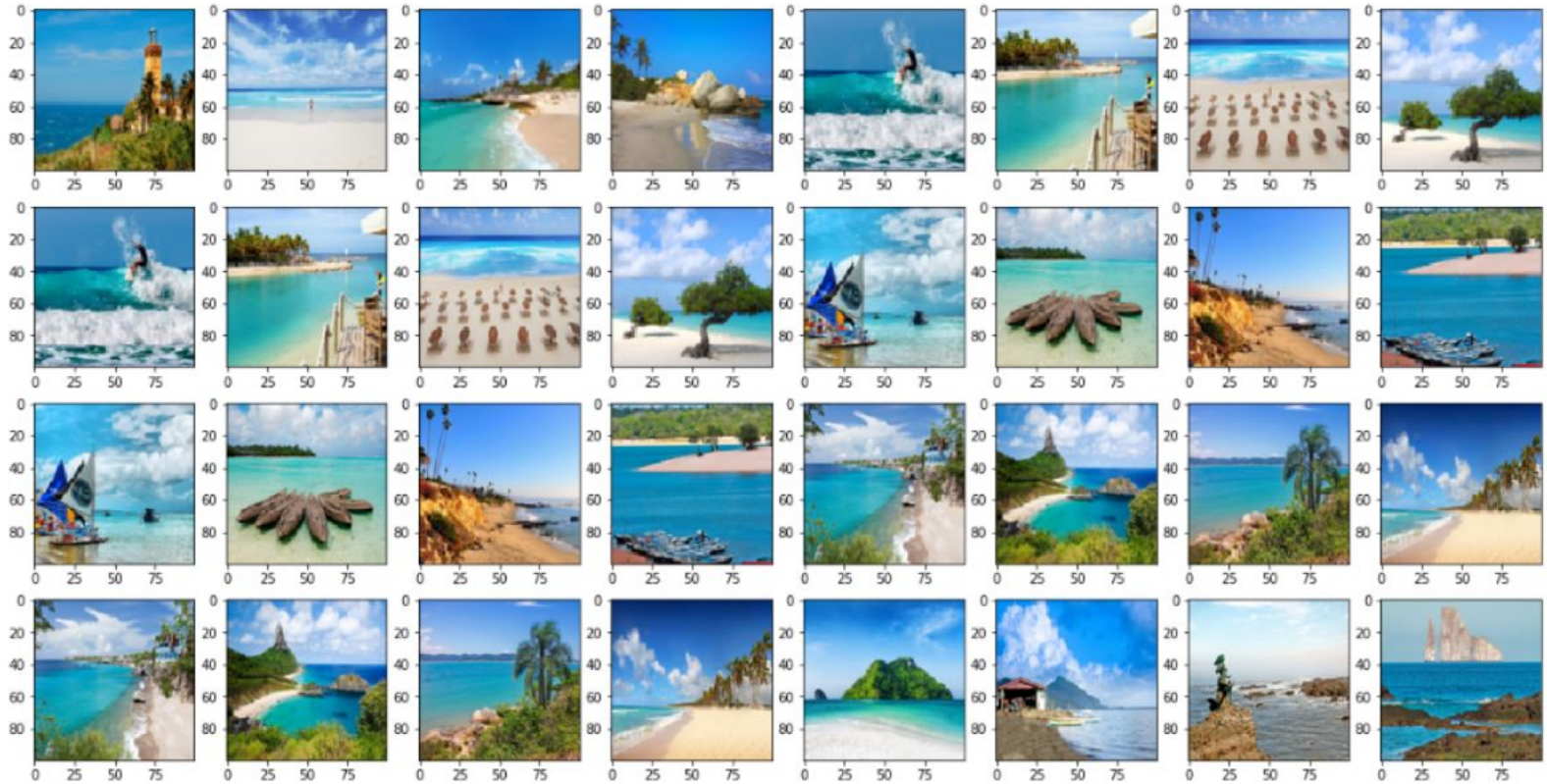
Embeddings de imágenes + K-means



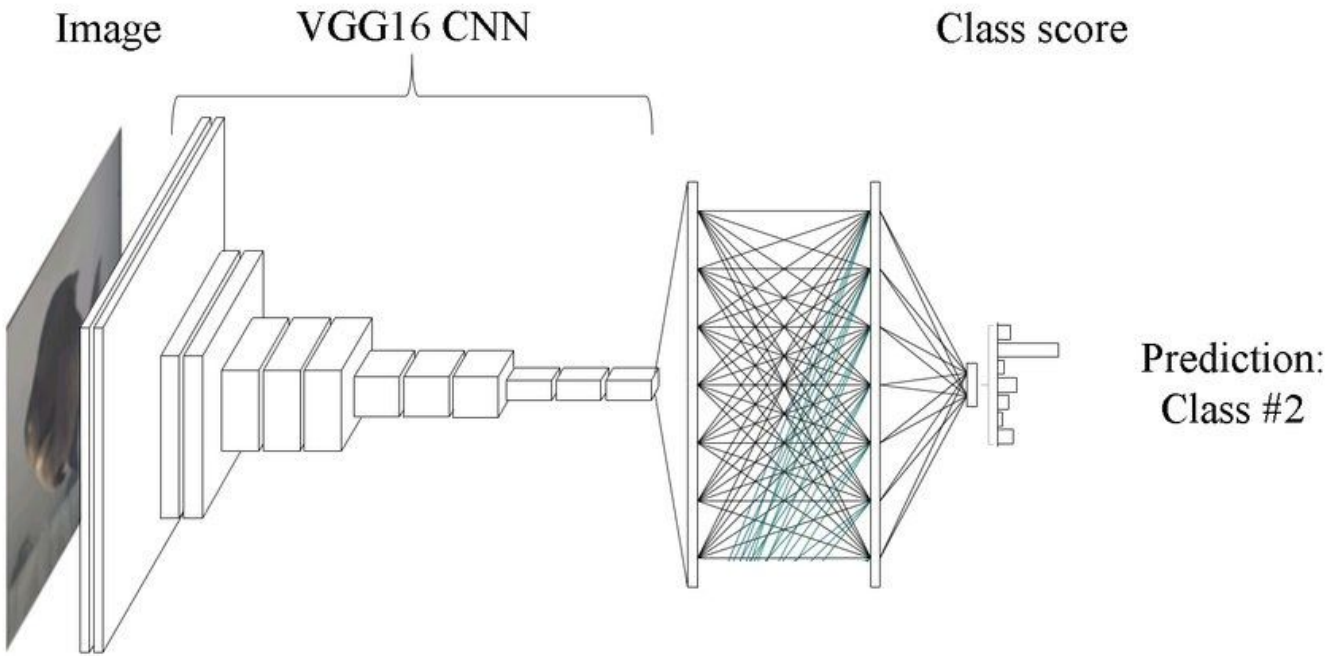
Cluster 8



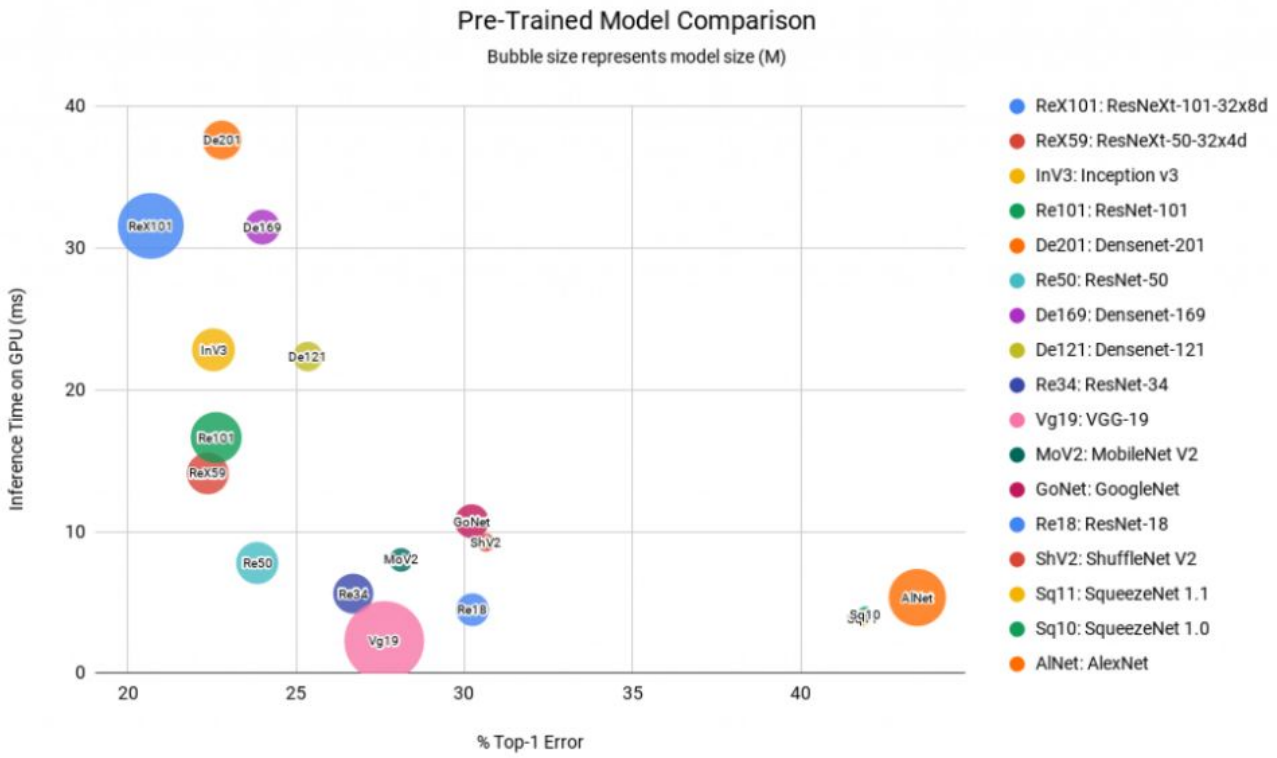
Cluster 11



Transferir conocimiento a nuestro nuevo clasificador



Comparación de los modelos pre entrenados de pytorch



Bibliografía

- Deep learning book -> https://github.com/janishar/mit-deep-learning-book-pdf
- Aplicacion para probar redes neuronales -> https://playground.tensorflow.org/
- Un buen blog que explica conceptos de redes neuronales -> https://colah.github.io/