



# Deep Learning con Pytorch

---

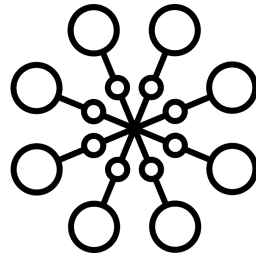
Juan Pablo Morales  
@juanpamf

---

# Regularización de Redes Neuronales profundas

---

# Construir un buen modelo de deep learning



## **La receta para construir una red neuronal de buena performance es similar a los modelos de machine learning:**

1. Construir un primer modelo simple que logre aprender sobre los datos.
2. Este modelo probablemente a pesar de haber aprendido sea muy simple y tenga cesgo (underfitting).
3. Aumentar la capacidad del modelo hasta overfittear.
4. Regularizar para encontrar el buen punto medio entre underfitting y overfitting.

# Cómo controlar la capacidad de mi modelo

## Para aumentar la capacidad:

- Agregar capas
- Agregar cantidad de neuronas por capa
- Aumentar el número de epochs
- Elegir un método de optimización más complejo (adaptividad, momentum, etc.)

## Para disminuir la capacidad:

- Entrenar con mayor cantidad de datos
- Utilizar técnicas de regularización

# L2 o weight decay

- Penalizar parámetros grandes:

$$L_{reg} = L + \frac{\lambda}{2 \cdot N} \cdot \sum w_{li}^2$$

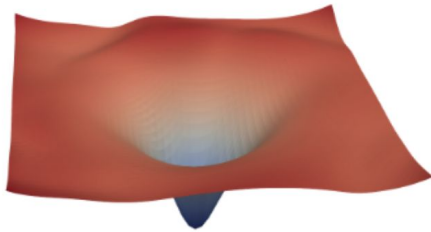
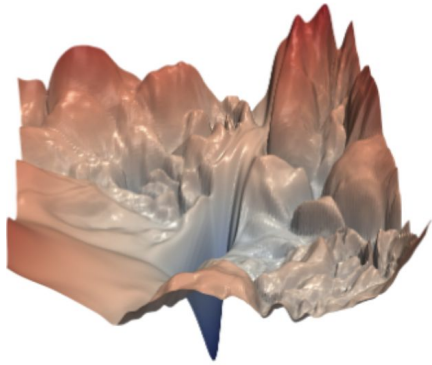
- Update con un término adicional de decaimiento:

$$w_{reg} = \left(1 - \frac{l_r \cdot \lambda}{m}\right) \cdot w - l_r \cdot dw$$

# Batch Norm

- Innovación clave en el deep learning contemporáneo (2016).
- Similar a normalizar los inputs (buena práctica), pero en capas intermedias.
  - Para un batch de datos calcular media y varianza.
  - Actualizar datos restando media y dividiendo por varianza más epsilon
  - Transformar con:  $f(x) = \gamma \cdot x + \beta$

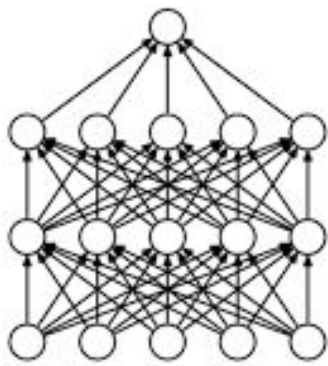
# ¿Por qué batch norm es eficiente?



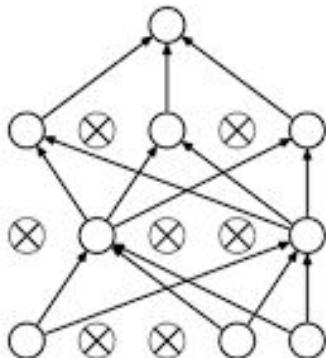
- Alisa la función de pérdida (MIT, 2019):
- Los mínimos a los que lleva son planos, y se sabe que tienen buena generalización
- Controla el internal covariate shift (ICS)



# Dropout



(a) Standard Neural Net







(b) After applying dropout.





## Algoritmo

- Con cierta probabilidad  $p$  "desactivaremos" ciertas neuronas ( $= 0$ )
- En cada iteración la capa recalculara si "desactiva" una neurona.
- Durante la fase de entrenamiento dropout actúa, durante la fase de evaluación las capas de dropout no tienen efecto.

# Data augmentation

Original	Flip	Rotation	Random crop
			
<ul style="list-style-type: none"><li>- Image without any modification</li></ul>	<ul style="list-style-type: none"><li>- Flipped with respect to an axis for which the meaning of the image is preserved</li></ul>	<ul style="list-style-type: none"><li>- Rotation with a slight angle</li><li>- Simulates incorrect horizon calibration</li></ul>	<ul style="list-style-type: none"><li>- Random focus on one part of the image</li><li>- Several random crops can be done in a row</li></ul>

Color shift	Noise addition	Information loss	Contrast change
			
<ul style="list-style-type: none"><li>- Nuances of RGB is slightly changed</li><li>- Captures noise that can occur with light exposure</li></ul>	<ul style="list-style-type: none"><li>- Addition of noise</li><li>- More tolerance to quality variation of inputs</li></ul>	<ul style="list-style-type: none"><li>- Parts of image ignored</li><li>- Mimics potential loss of parts of image</li></ul>	<ul style="list-style-type: none"><li>- Luminosity changes</li><li>- Controls difference in exposition due to time of day</li></ul>