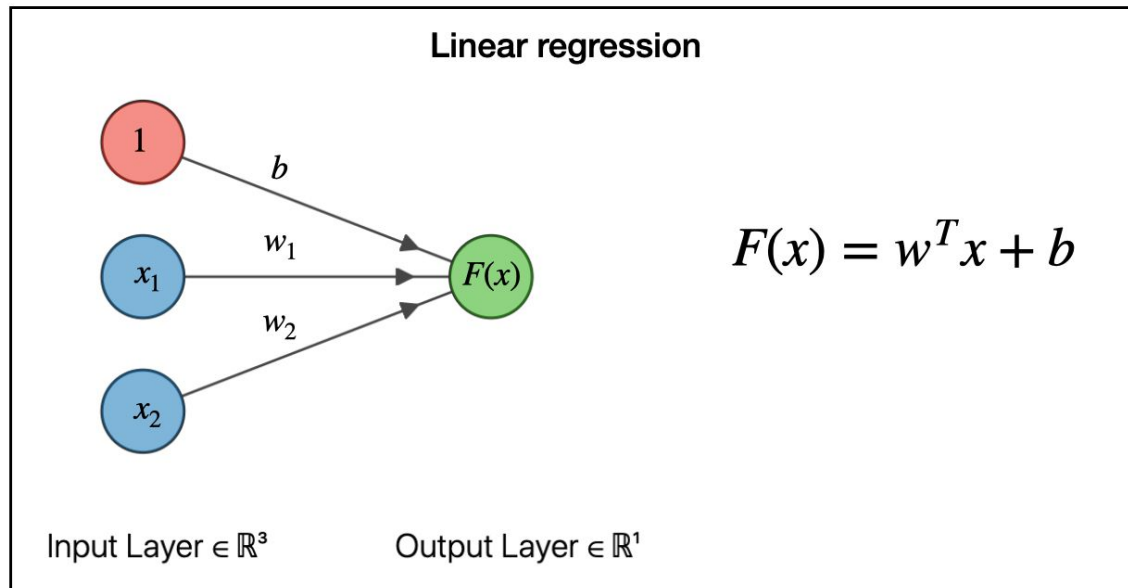


# Deep Learning

# Objetivos

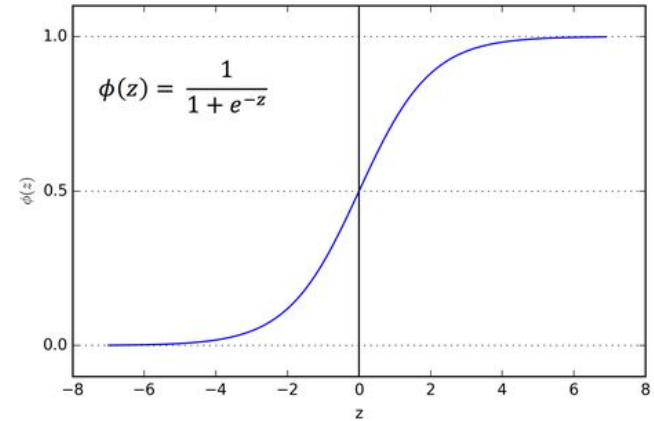
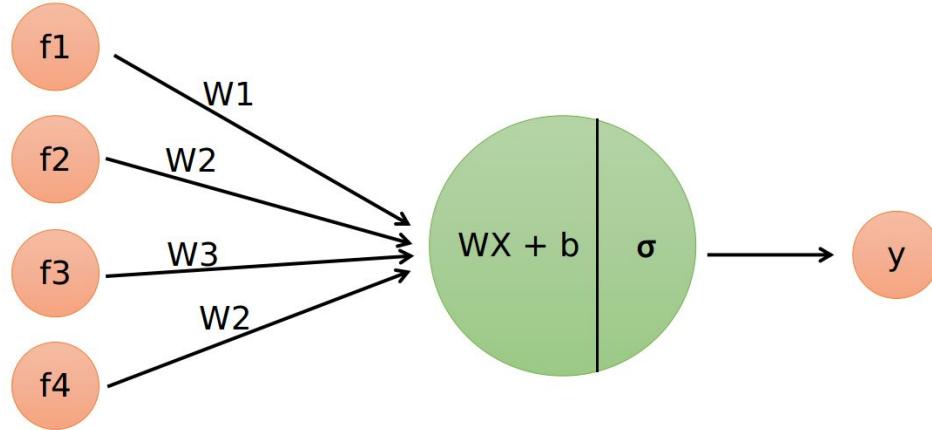
- Keras + Tensorflow
- Conceptos de ANN
- Imágenes
- Google Colab
- Kaggle
- CNN

# Regresión Lineal como Neurona

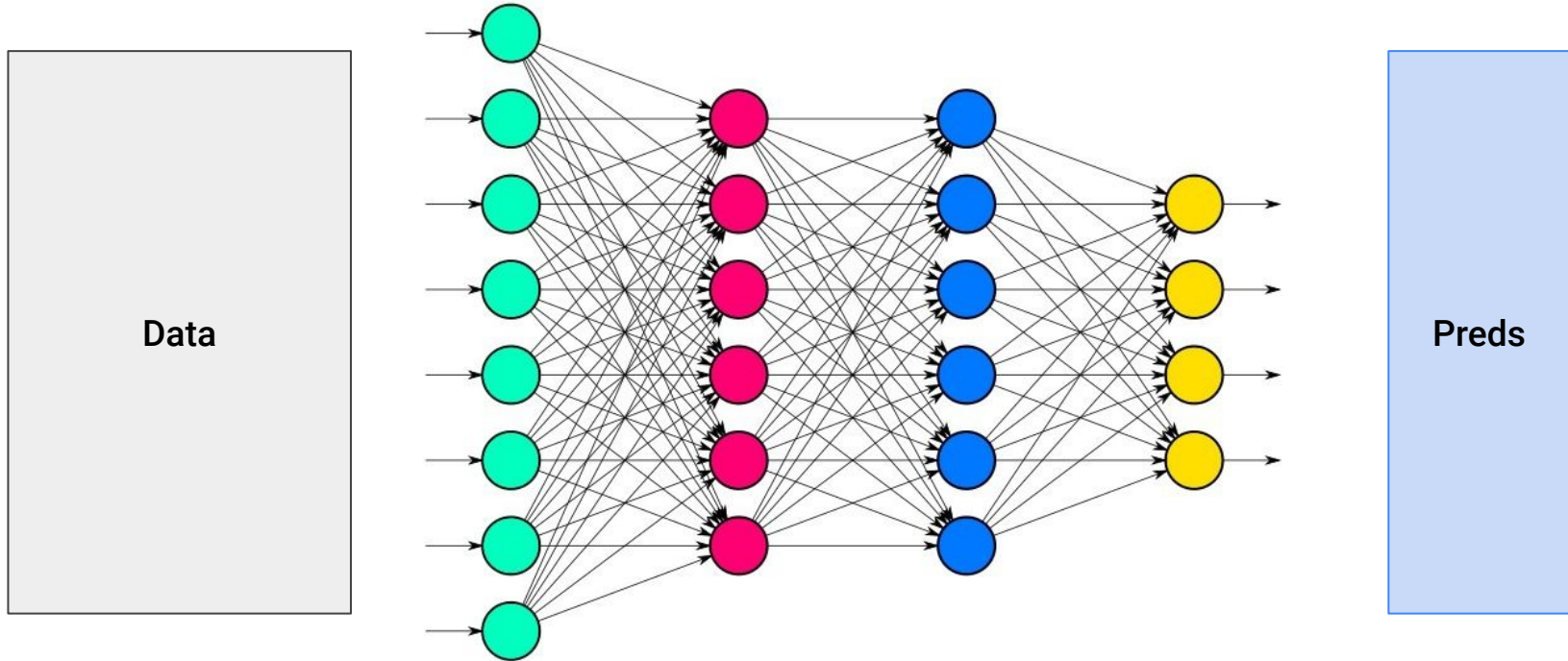


$$F(x) = w_1 \cdot x_1 + w_2 \cdot x_2 + 1 \cdot b$$

# Regresión Logística como Neurona



# Redes Neuronales Densas

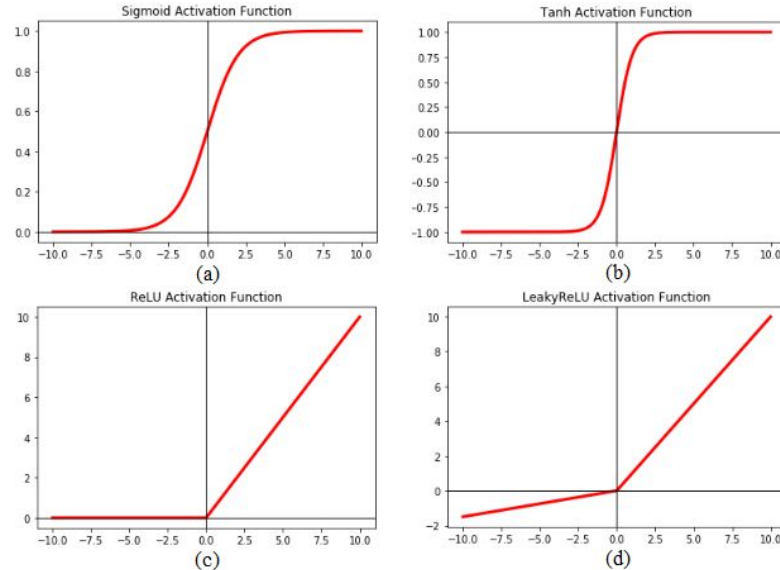


# ANN

- Aproximadores universales.
- Transformación sucesiva de los datos para generar representaciones que facilitan la resolución del problema.
- Importancia de activaciones No Lineales.

# Activaciones

- Le dan mayor flexibilidad a las representaciones creadas por la red en las capas ocultas.

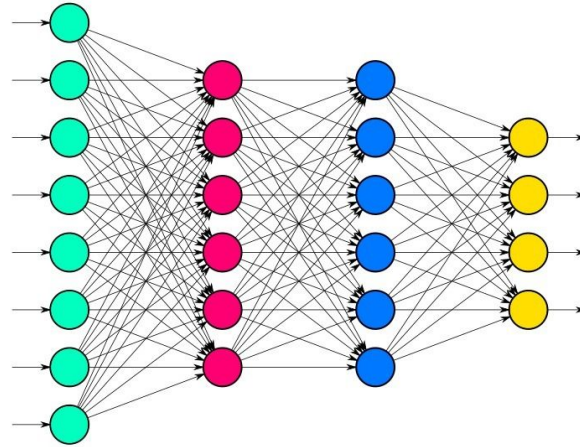


# Conceptos ANN

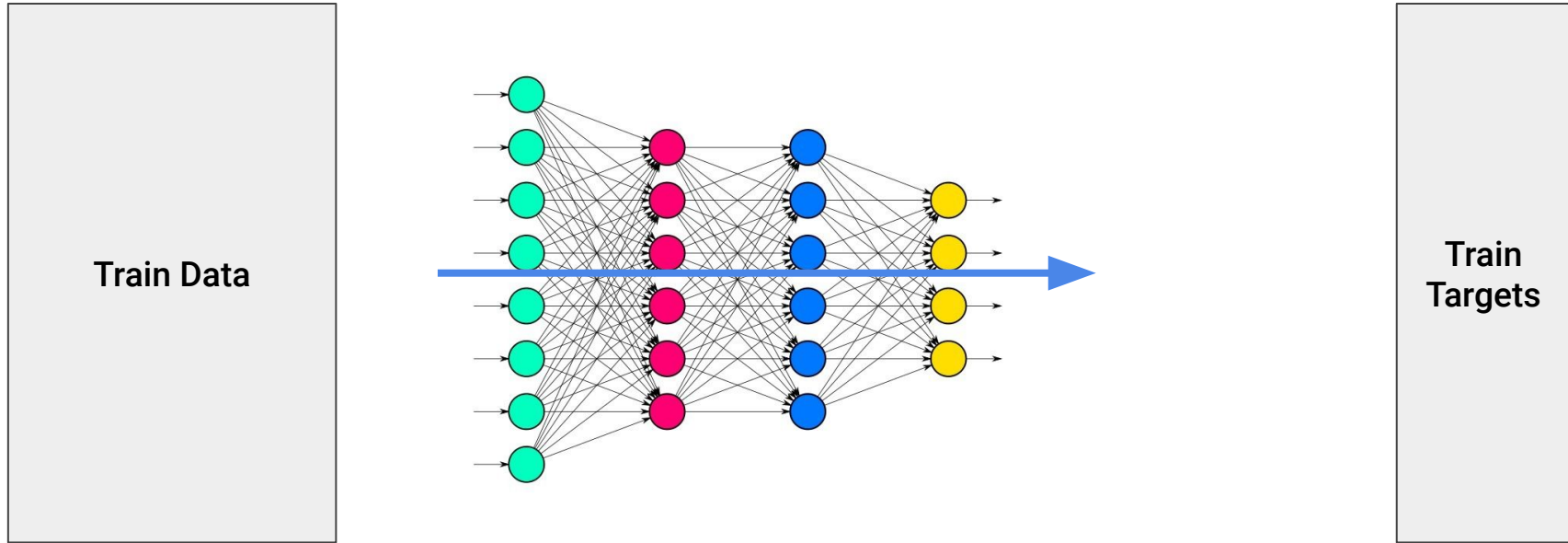
- Datos Normalizados
- Entrenamiento - Backpropagation
- Loss Function
- Learning Rate
- Batch Size
- Epochs
- Activaciones



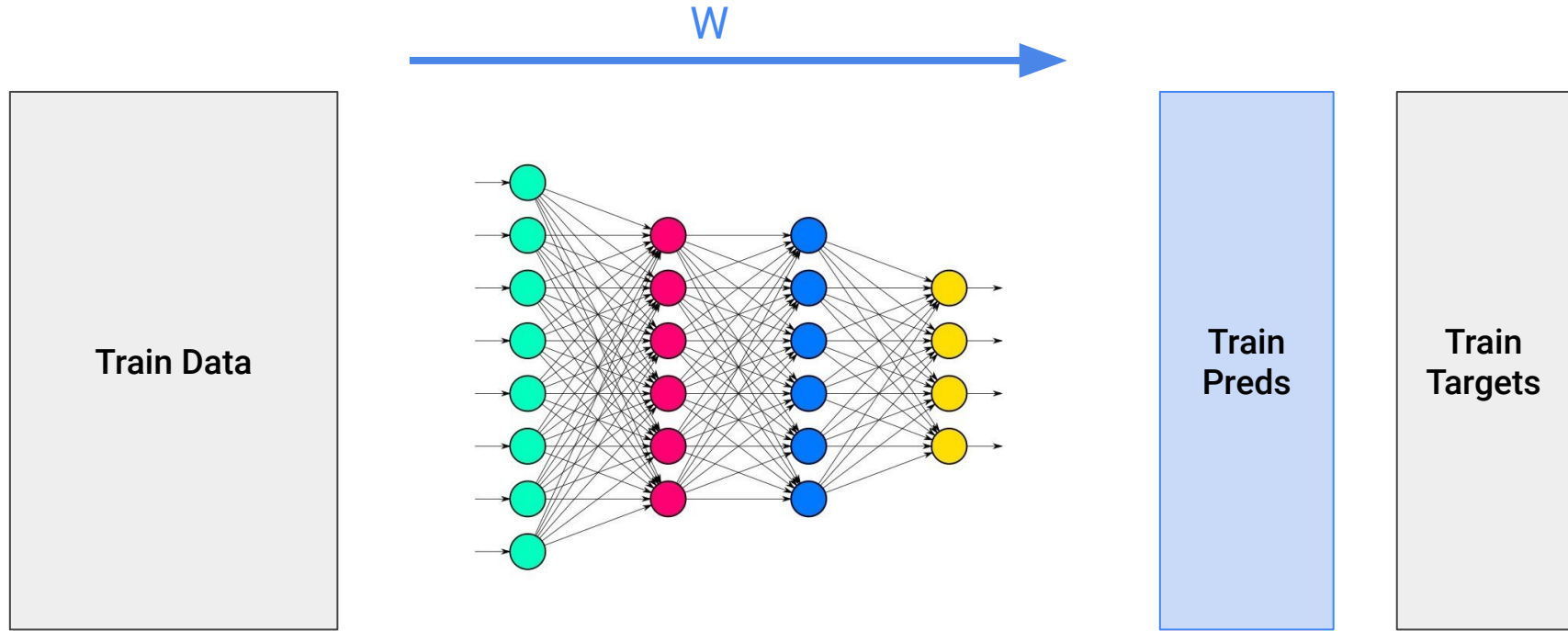
# Entrenamiento



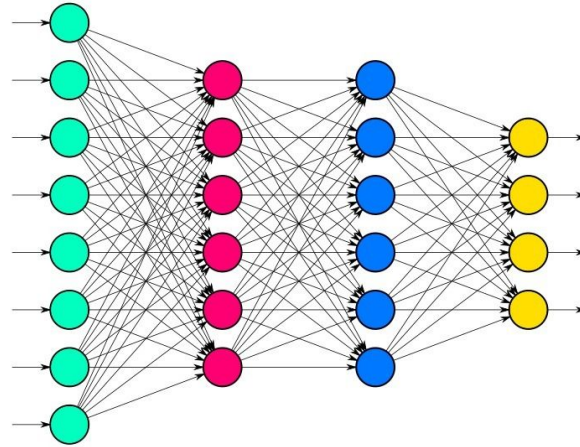
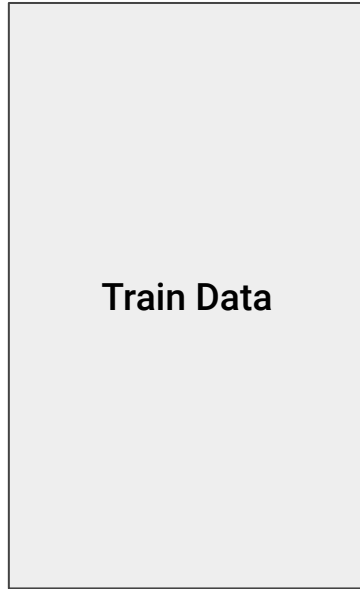
# Entrenamiento



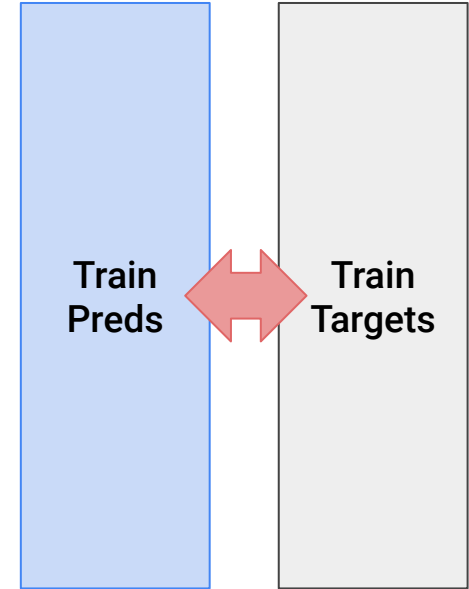
# Entrenamiento



# Entrenamiento



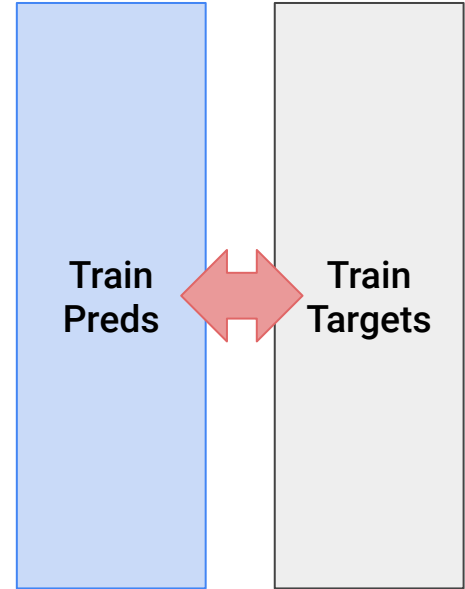
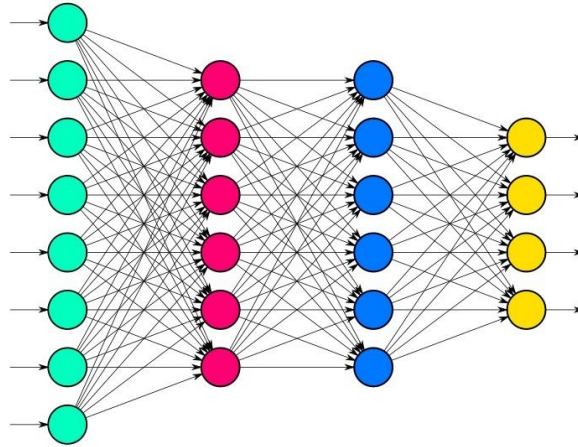
Loss



# Entrenamiento

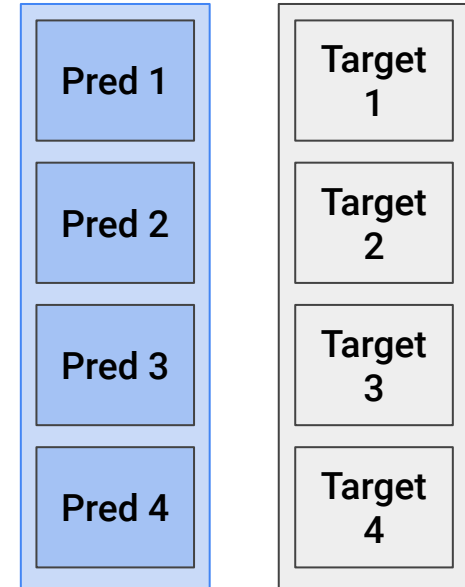
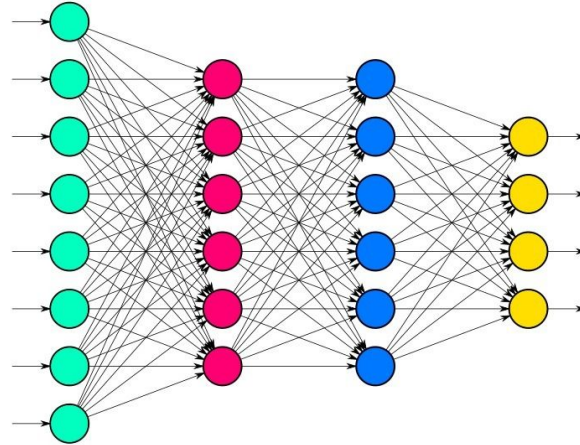
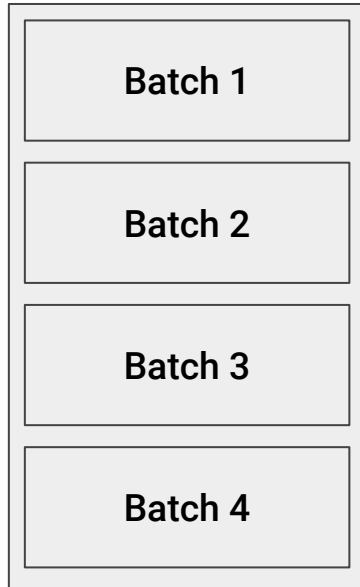
Optimización = Update W

Min(Loss)



# Entrenamiento

- Batch = Lote ~ N Observaciones para actualizar W.
- Epoch = Iteración ~ N Veces que se repite el proceso.
- Learning Rate = Magnitud de actualización de W.



# Keras

- Layers - <https://keras.io/api/layers/>
- Loss Function - <https://keras.io/api/losses/>
- Metrics - <https://keras.io/api/metrics/>
- Optimizers - <https://keras.io/api/optimizers/>

## Colab + Kaggle

kaggle

<https://www.kaggle.com/>



<https://colab.research.google.com/>



# Scenes Dataset

Usamos un subset de datos de **5000** observaciones y predecimos **4** de las 6 clases presentes.



# Computer Vision Tasks

Single-label multi-class classification



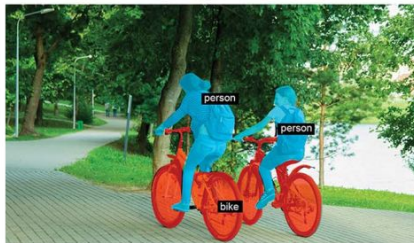
- ☒ Biking
- ☐ Running
- ☐ Swimming

Multi-label classification



- |  |  |
|--|--|
| <input checked="" type="checkbox"/> Bike   | <input checked="" type="checkbox"/> Tree |
| <input checked="" type="checkbox"/> Person | <input type="checkbox"/> Car             |
| <input type="checkbox"/> Boat              | <input type="checkbox"/> House           |

Image segmentation



Object detection



# Convoluciones

- En Imágenes tenemos una alta dimensionalidad (gran cantidad de pixeles) que se traducen a grandes cantidad de parámetros en DNN.
- En Imágenes hay patrones que se repiten que son invariantes a traslación.
- Buscamos una mejor manera de detectar variables en imágenes, donde es importante la espacialidad pero no la posición.

# Convolución

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

\*

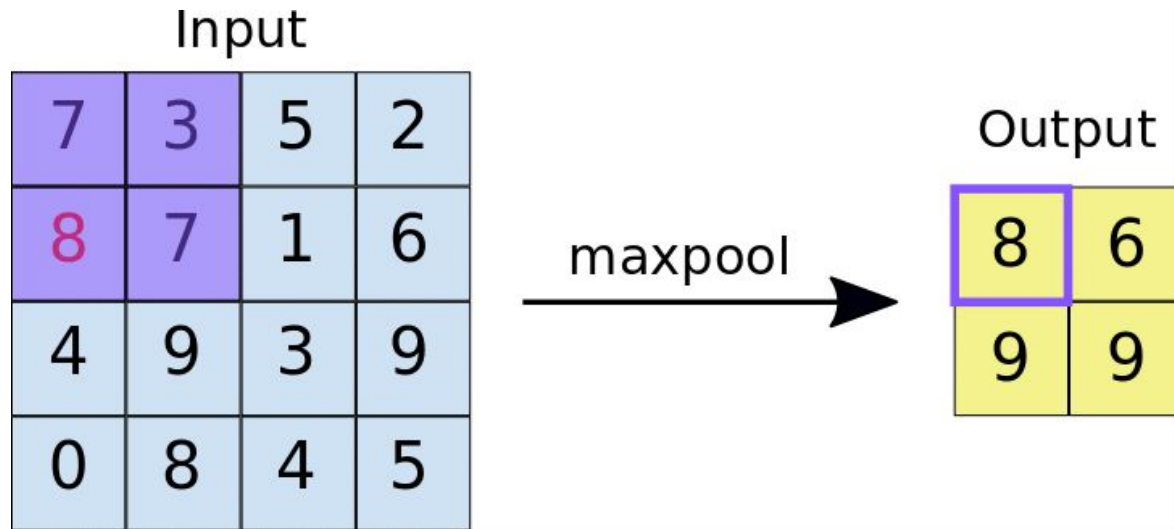
1	0	-1
1	0	-1
1	0	-1

=

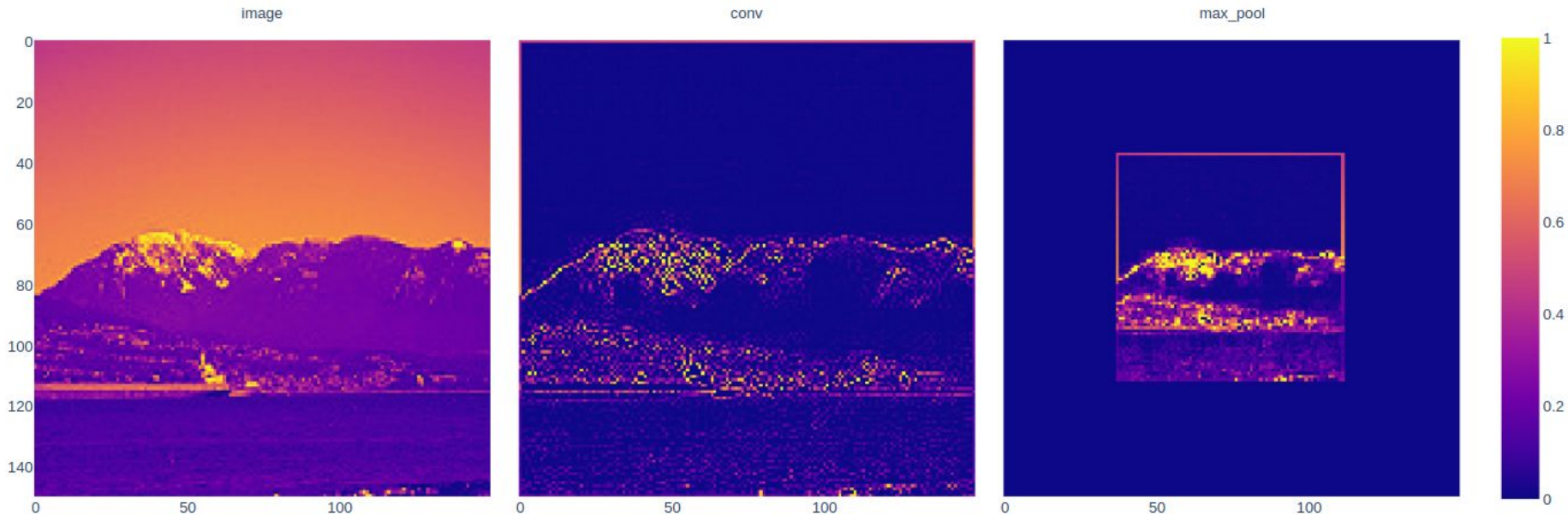
6		

$$\begin{aligned} &7 \times 1 + 4 \times 1 + 3 \times 1 + \\ &2 \times 0 + 5 \times 0 + 3 \times 0 + \\ &3 \times -1 + 3 \times -1 + 2 \times -1 \\ &= 6 \end{aligned}$$

# Max Pooling

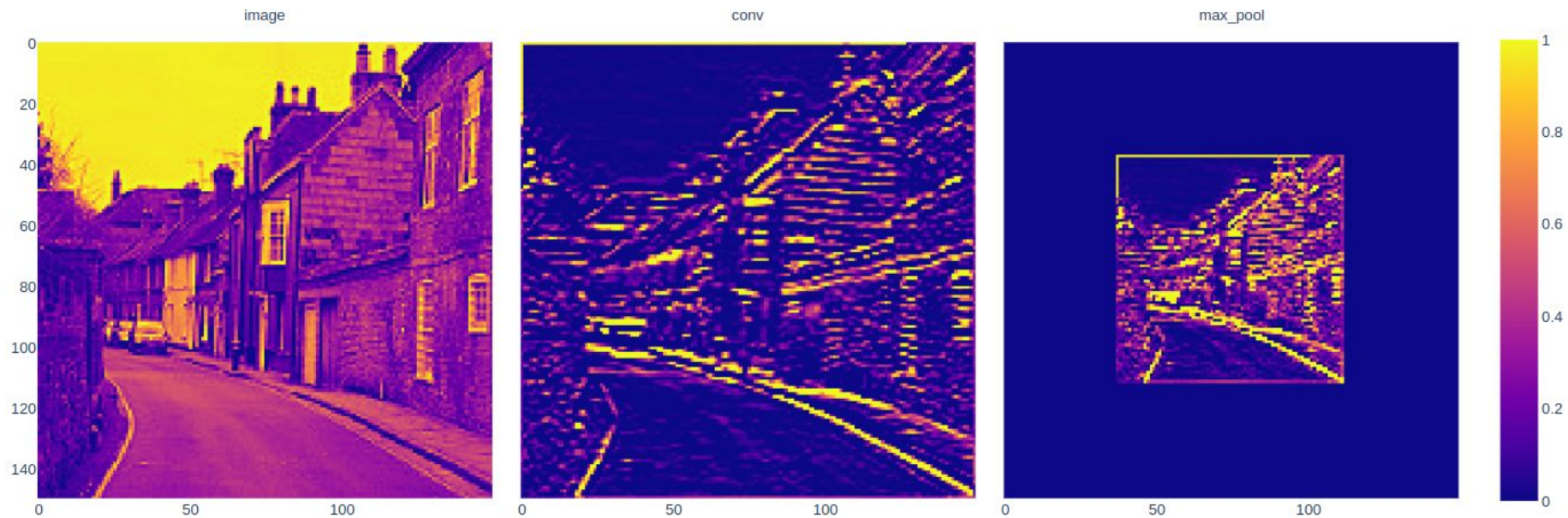


# Convolución + Max Pooling





# Convolución + Max Pooling

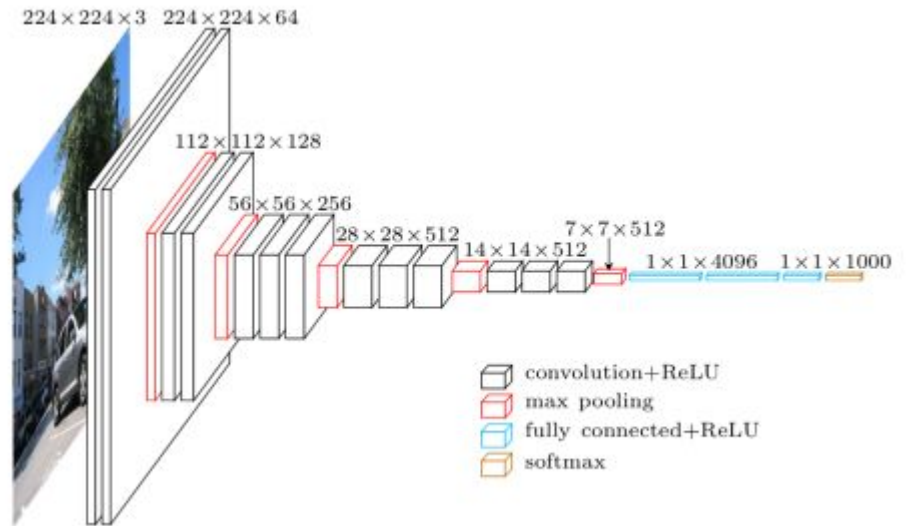
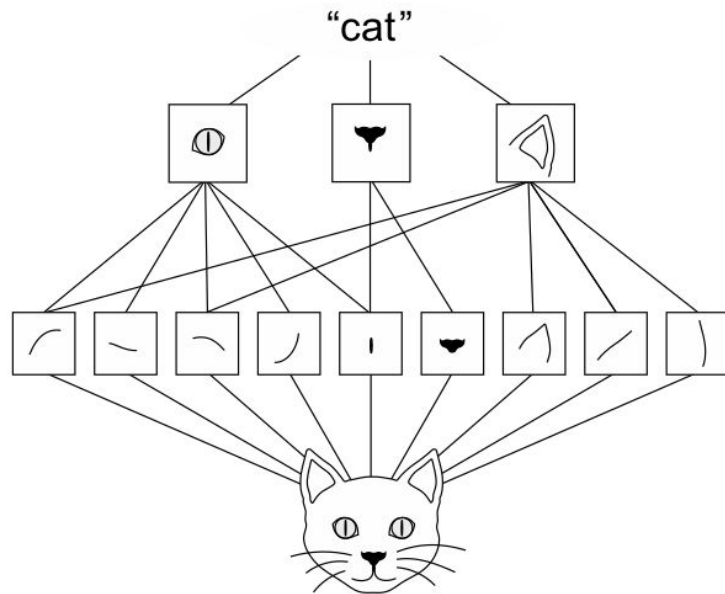


# CNN

- **Aplicando Convoluciones:**
  - Detectamos patrones locales (las DNN aprenden patrones globales)
  - Detectamos patrones invariantes a traslación: Si el mismo patrón aparece en otro lugar de la imagen será detectado.
  - Aprendemos patrones jerárquicos.
  - Requiere menor cantidad de parámetros.

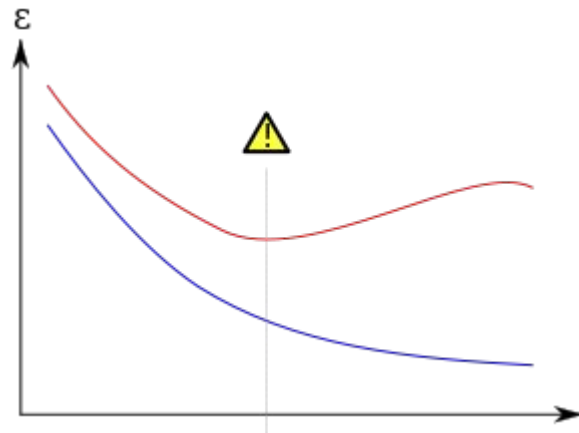


# CNN



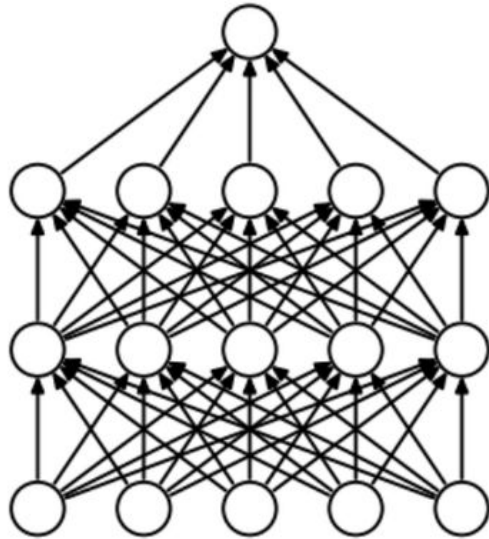
# Overfitting

- Cuando las redes neuronales tienen grandes cantidades de parámetros y datos de entrenamiento limitados pueden ajustarse demasiado a esos datos, memorizando el entrenamiento.
- Tendrán menor performance en datos nunca vistos.
- Soluciones:
  - Regularización (Dropout)
  - Data Augmentation
  - Early Stopping

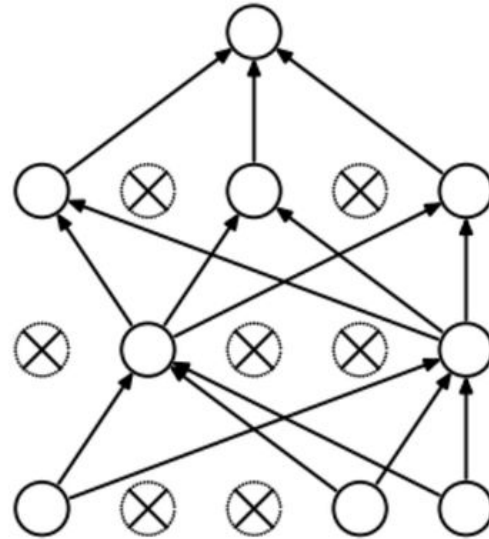


# Regularization - Dropout

- Se apagan neuronas de una capa de manera aleatoria durante el entrenamiento.



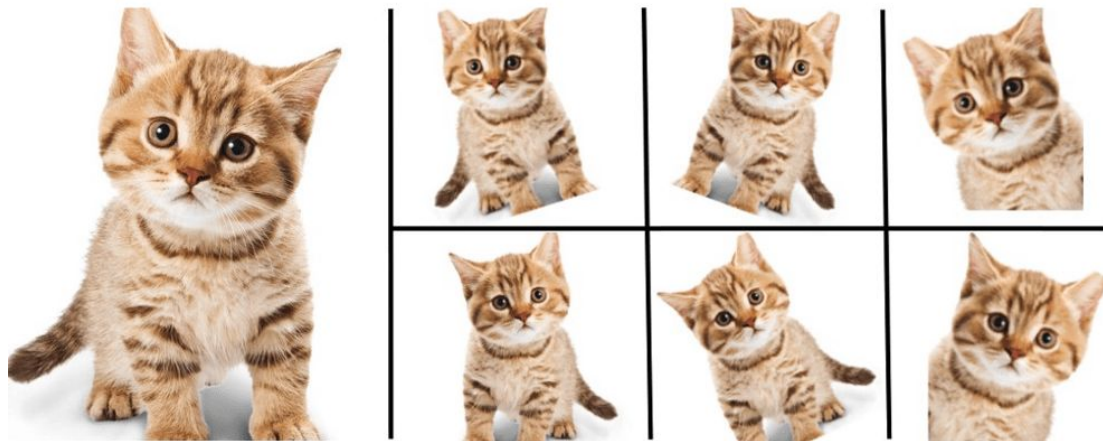
(a) Standard Neural Net



(b) After applying dropout.

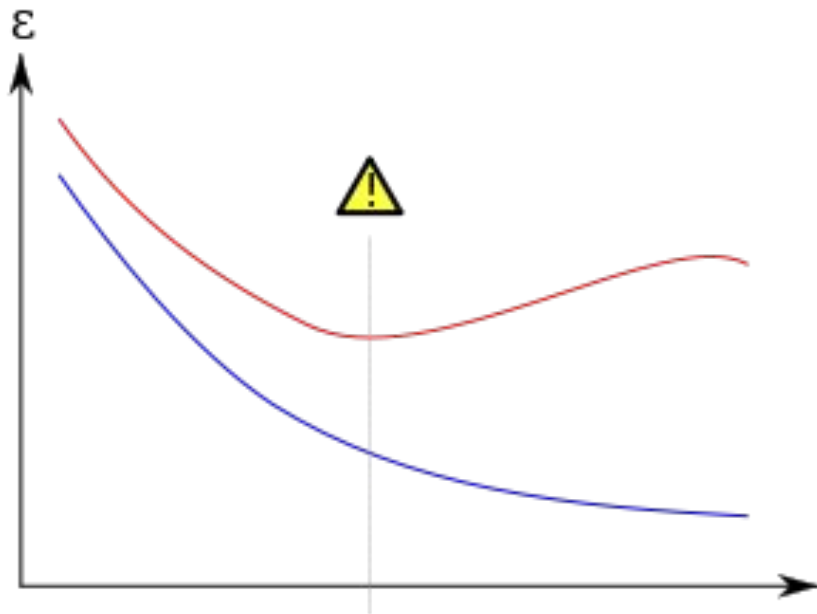
# Data Augmentation

- En imágenes, hacer cambios aleatorios no altera su sentido. Podemos introducir cambios aleatorios en el input del modelo durante el **entrenamiento** para que en cada iteración el modelo nunca vea 2 veces la misma imagen.



# Early Stopping

- Frenamos el entrenamiento en la iteración de mejor resultado en validación.

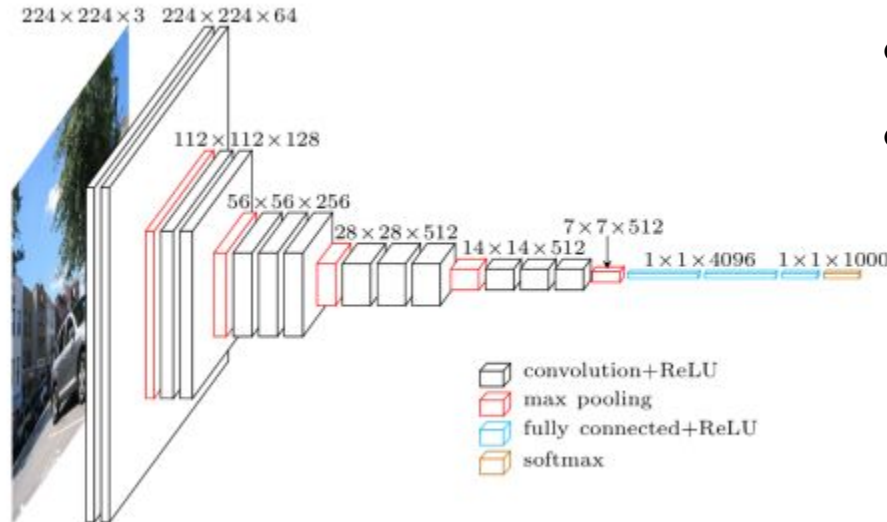


Medimos el error de entrenamiento (azul) y de validación (rojo) durante el entrenamiento del modelo.

Frenamos el entrenamiento cuando el error de validación deja de mejorar.

# Transfer Learning

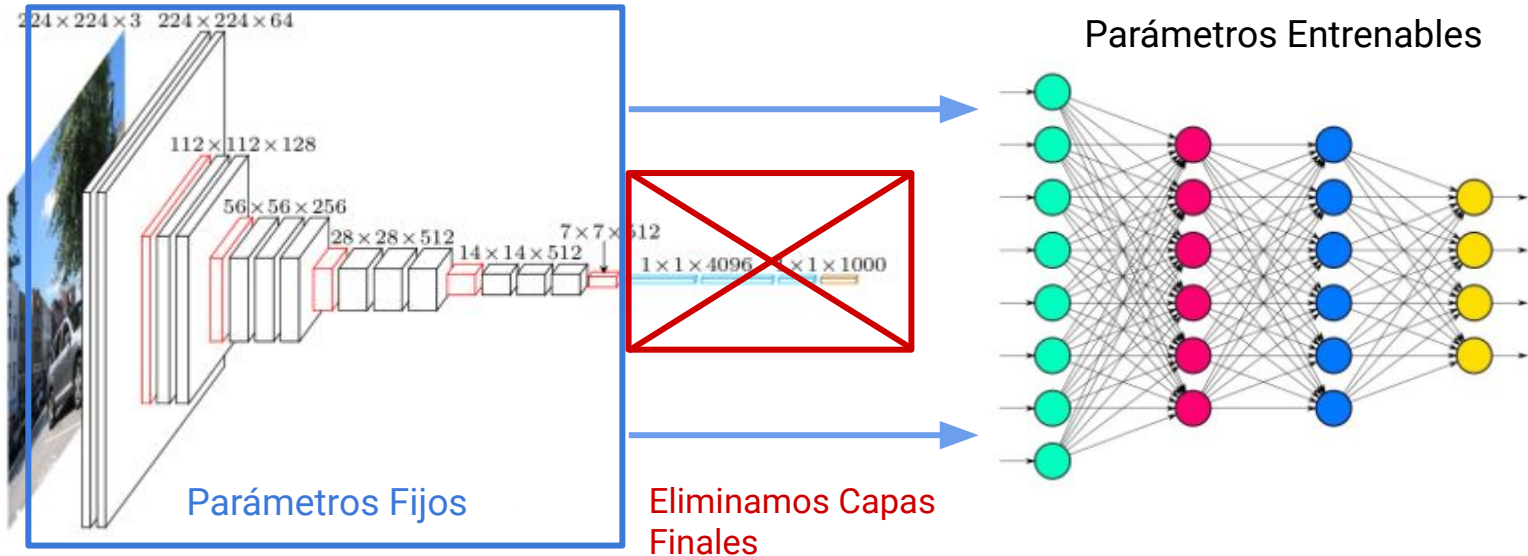
- Es el método de aprovechar los parámetros entrenados de una red y reutilizarlos en un problema diferente, cambiando las últimas capas.



- VGG16
- ImageNet

# Transfer Learning

- Podemos aprovechar los parámetros de las capas convolucionales de redes ya entrenadas.
- Estas capas ya deberían captar características y patrones básicos en las imágenes.



# Control de Predicciones

- Al ser modelos menos interpretables, es importante controlar los datos de entrenamiento y evaluar las predicciones para detectar sesgos.





# Control de Predicciones



(a) Husky classified as wolf



(b) Explanation