REDES NEURALES

ESTRUCTURA DE LA CLASE

INTRODUCCIÓN

Presentación del tema de la clase

Introducción al Deep Learning

REDES DENSAS

Perceptrón multicapa

Funciones de activación

Aprendizaje de redes neurales densas

Algoritmos de optimización

Desvanecimiento y explosión del gradiente

Implementación de una red neural

REDES CONVOLUCIONALES

Red neural convolucional

Componentes básicos de la red

Kernels y convoluciones

Implementación

REGULARIZACIÓN

Sobreajuste y subajuste de modelos

Técnica de detención temprana

Técnica de abandono

Técnica por Norma Límite

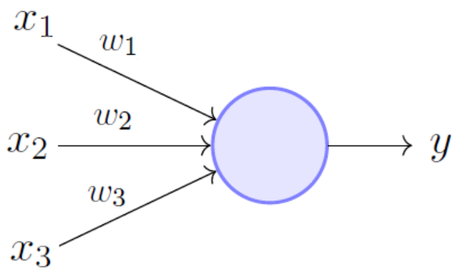
ACTIVIDAD PRÁCTICA

Alexnet

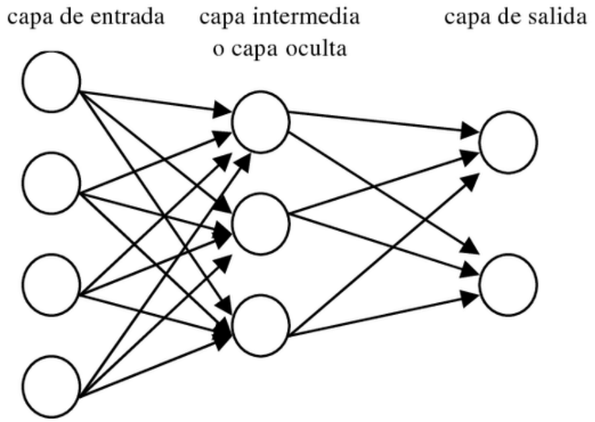
CONCLUSIONES

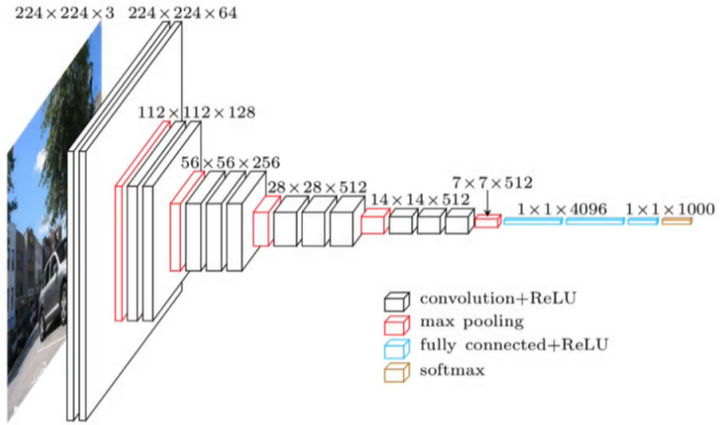
Recapitulación de los puntos clave de la clase

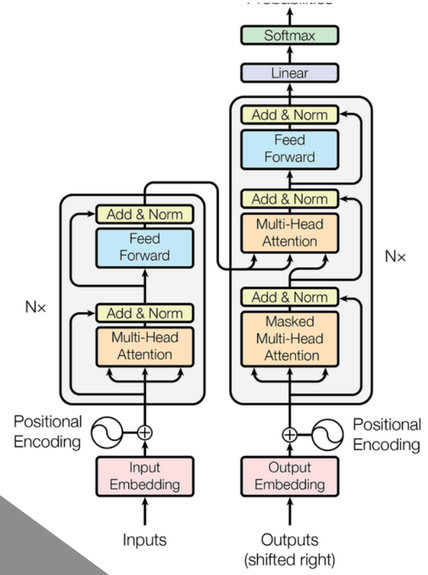
EVOLUCIÓN DE LAS REDES NEURALES

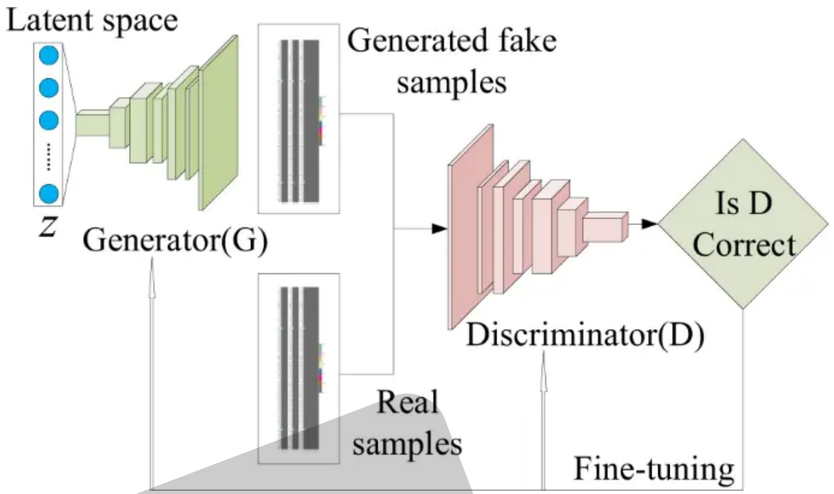


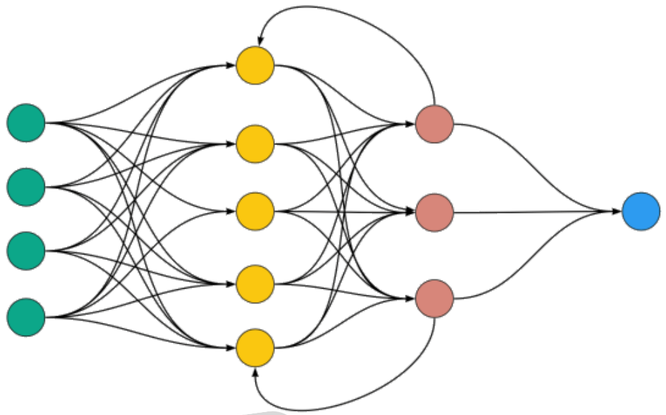
Perceptron Model (Minsky-Papert in 1969)



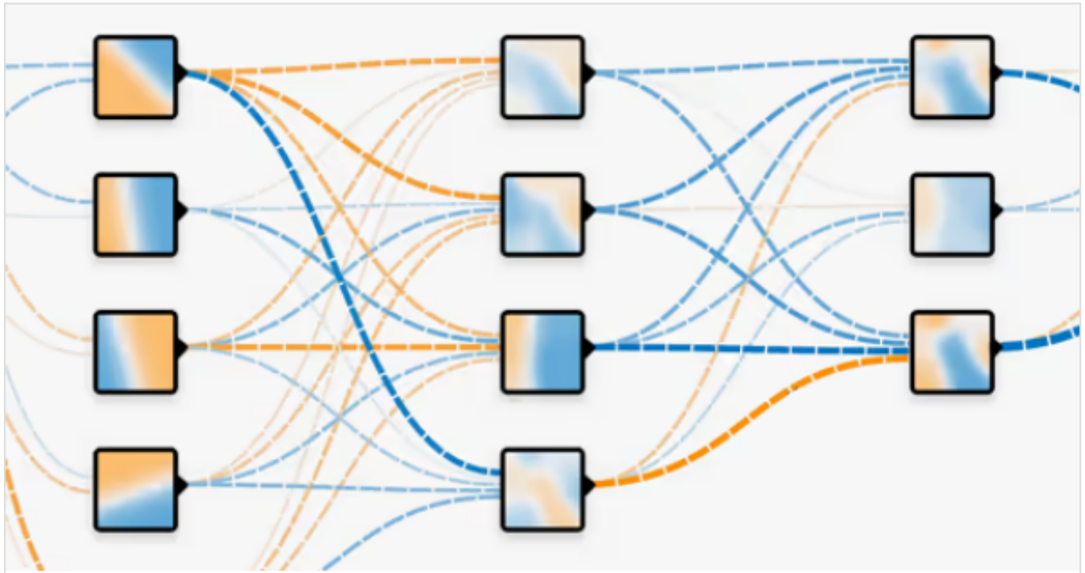








PERCEPTRÓN MULTICAPA



Tensorflow — Neural Network Playground

Tinker with a real neural network right here in your browser.

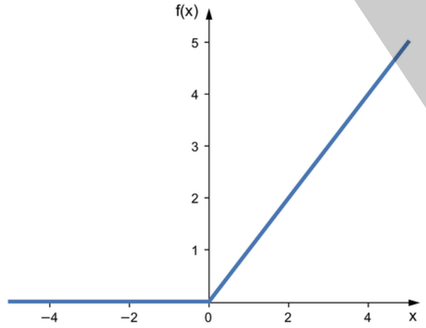
tensorflow.org /

- Problema del XOR
- Necesidad de función de activación no lineal

FUNCIONES DE ACTIVACIÓN

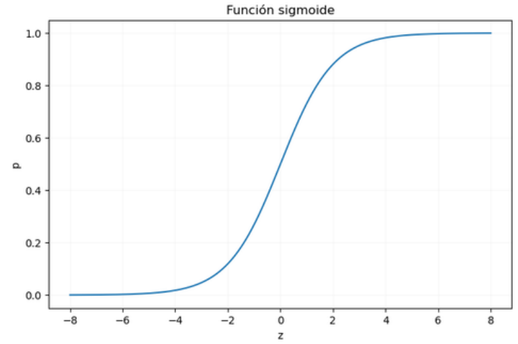
RELU

Lineal para valores positivos, 0 para negativos



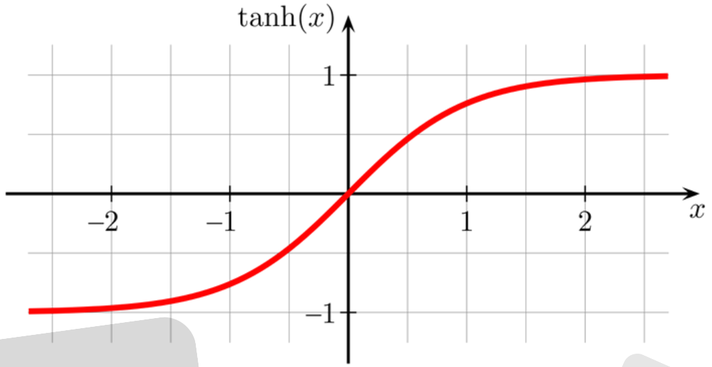
SIGMOIDE

Rango (0, 1) con crecimiento explosivo a partir de X=0.5



TANGENTE HIPERBÓLICA

Rango (-1, 1) con crecimiento a partir de X=0



APRENDIZAJE DE REDES NEURALES

CÁLCULO DEL ERROR

Calcula el error dada la diferencia entre el valor dado y el predicho

CULPABILIDAD DEL ERROR

Divide el error de acuerdo a las neuronas culpables del mismo

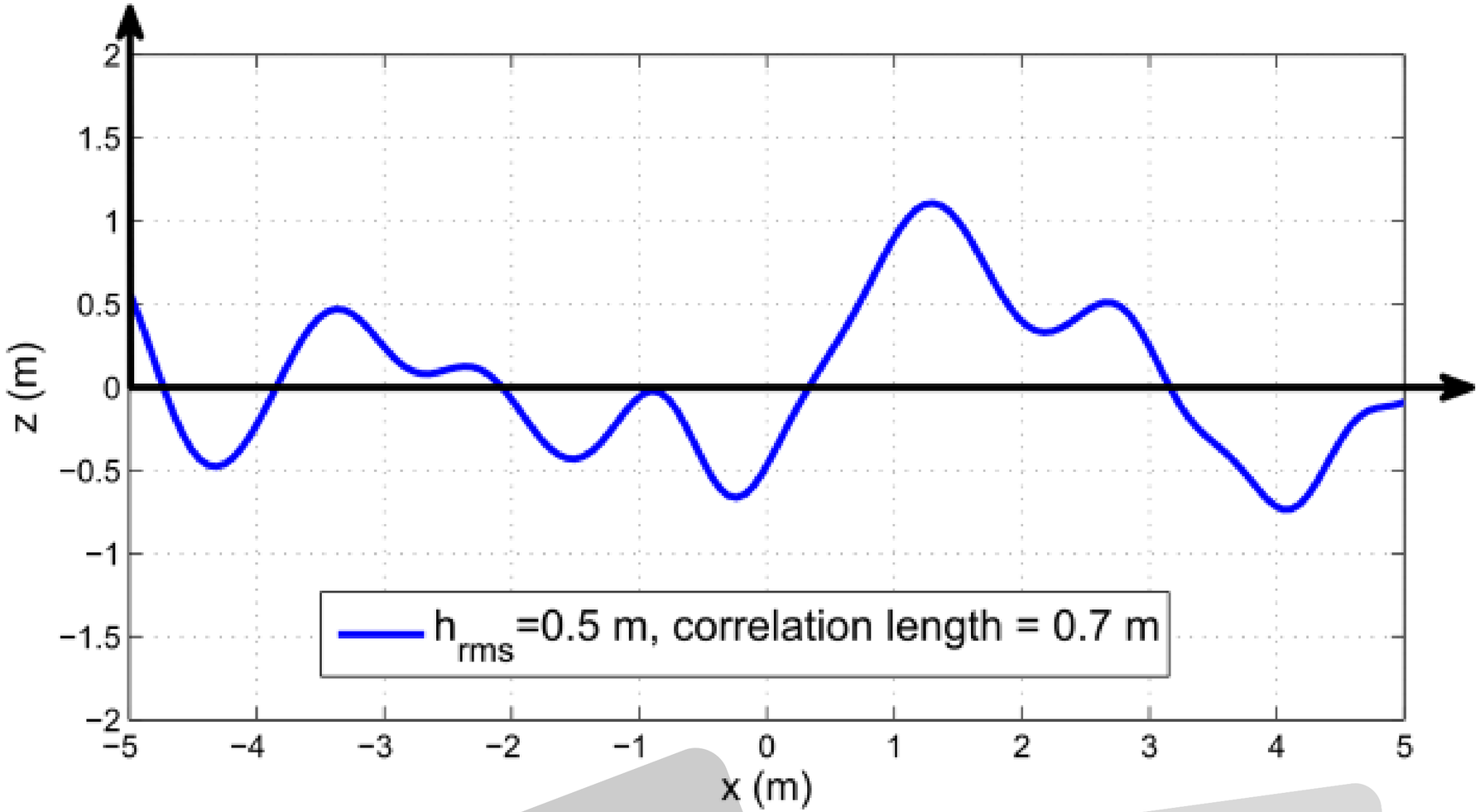
AJUSTE DE PARÁMETRO

Cada neurona asume su error, se ajusta usando un algoritmo de optimización

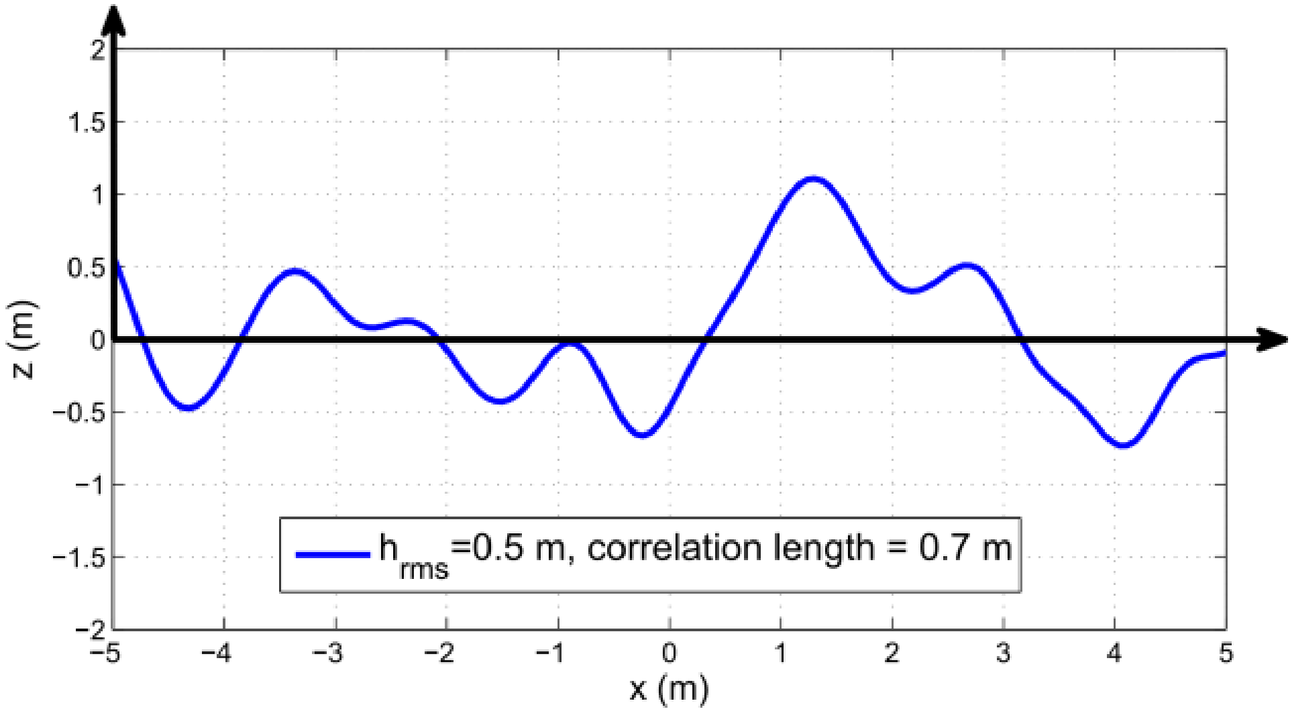
RETROPROPAGACIÓN

Cada neurona hace los pasos anteriores hasta llegar de nuevo al principio de la red

MINIMIZACIÓN DEL ERROR



ALGORITMOS DE OPTIMIZACIÓN



- RMS
- SGD
- ADAM

PROBLEMAS DEL GRADIENTE

DESVENECIMIENTO

La derivada es muy pequeña

Sucede en funciones saturadas como sigmoide.

La red se estanca en aprenzidaje

EXPLOSIÓN

La derivada es muy grande

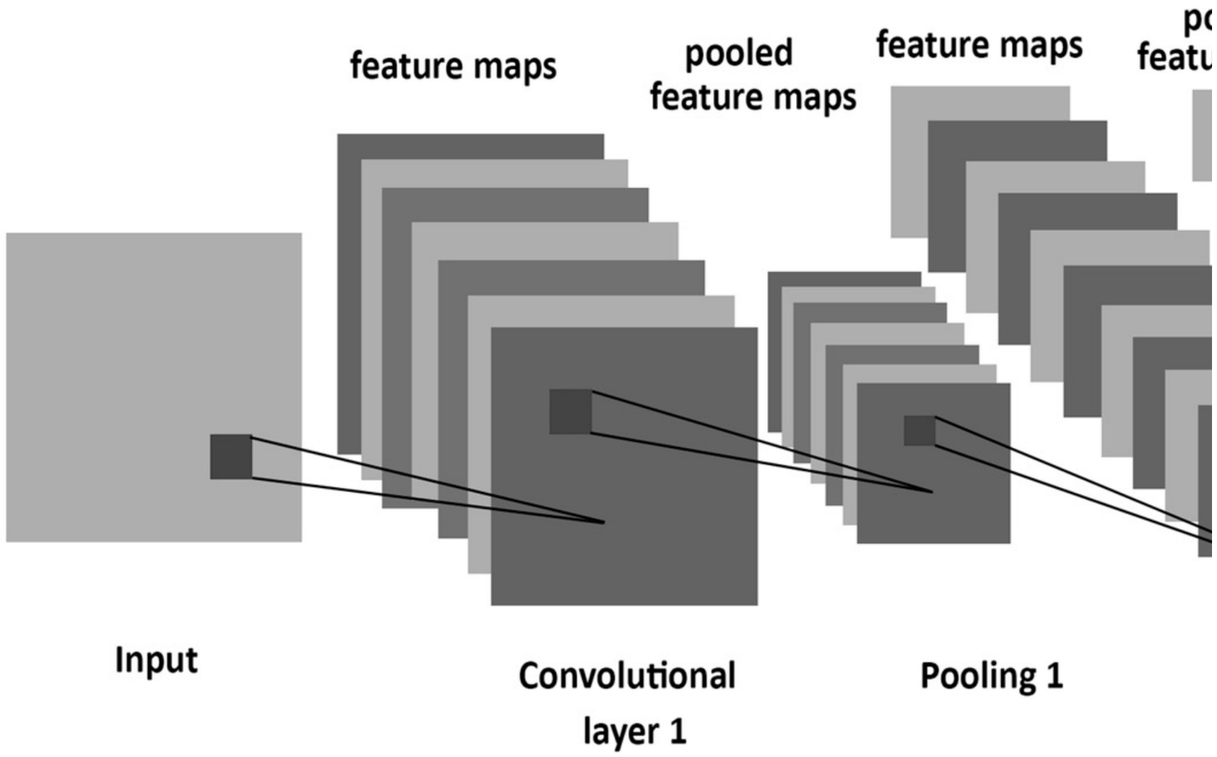
Sucede en funciones de derivada grande como ReLU.

El aprendizaje se vuelve inestable

Se solventan con la buena selección de hiperparámetros y el uso de métodos de regularización

```python
from tensorflow import keras

# Creamos un modelo secuencial
model = keras.Sequential()

# Añadimos una capa con neuronas y activación relu
model.add(keras.layers.Dense(100, activation="relu"))

# Añadimos una capa de salida con 10 categorías
model.add(keras.layers.Dense(10, activation="softmax"))

# Compilamos el modelo
model.compile(optimizer="adam",
loss="sparse_categorical_crossentropy", metrics=["accuracy"])

# Entrenamos el modelo
model.fit(X_train, y_train, epochs=100)

# Evaluamos el modelo
model.evaluate(X_test, y_test)
```

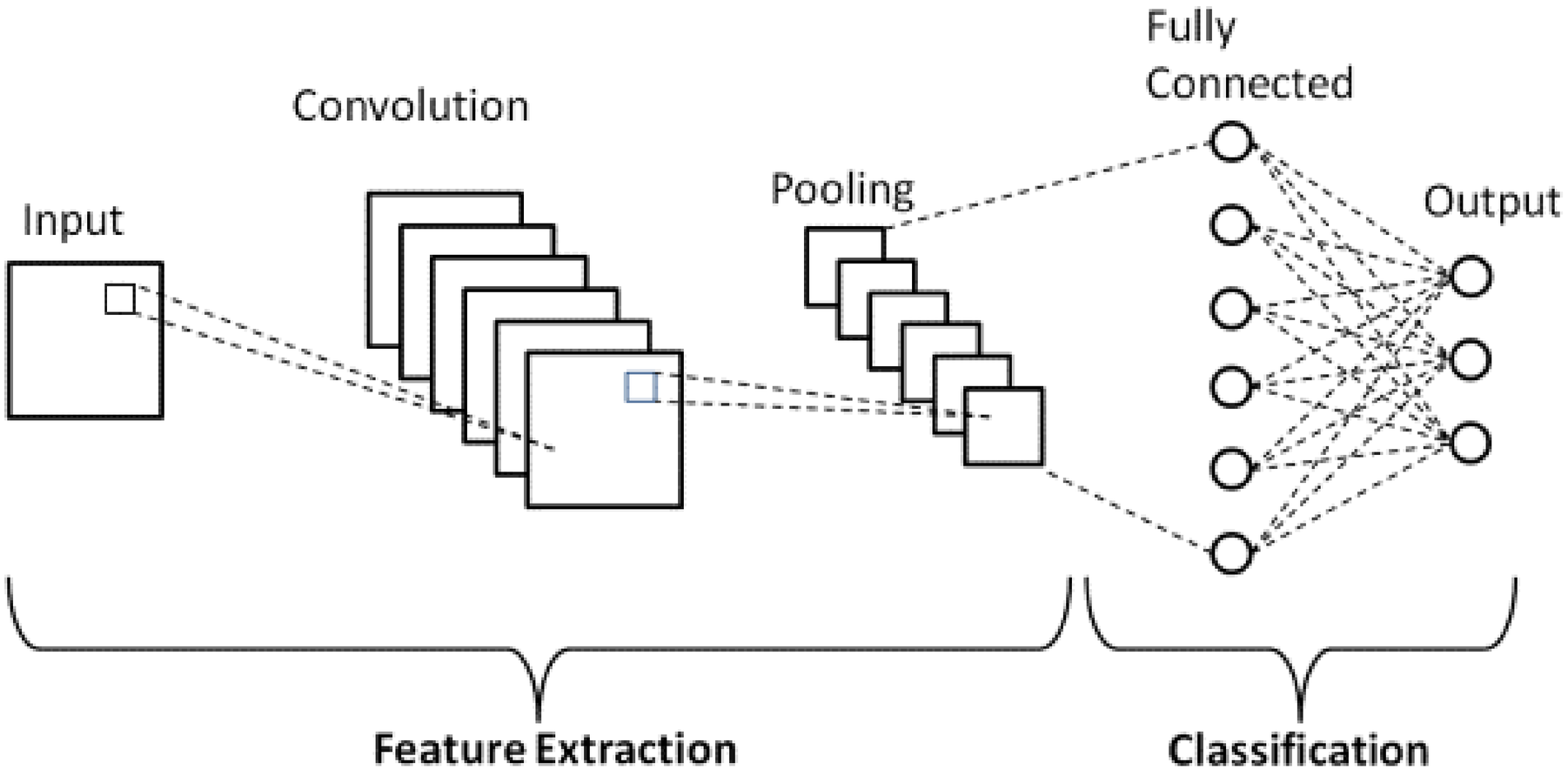
RED NEURONAL CONVOLUCIONAL



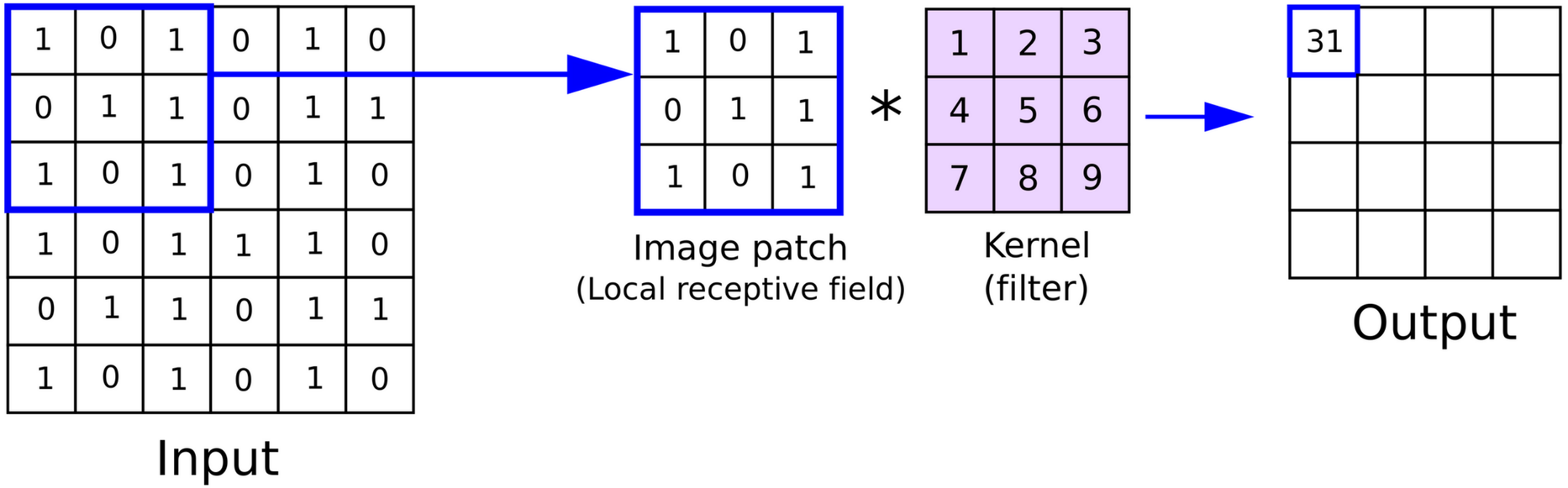
Variación del perceptrón multicapa donde se considera la posición en el espacio

La salida de cada capa aplica la función convolución

ARQUITECTURA DE LA RED



KERNELS Y CONVOLUCIONES

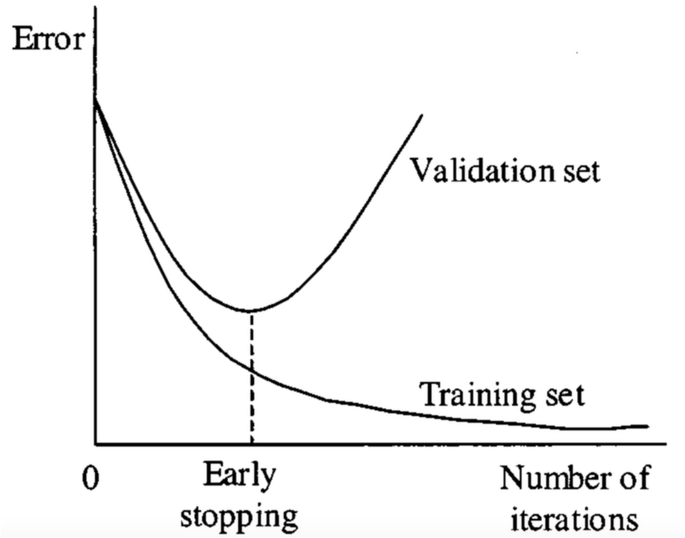


```python
from tensorflow import keras

# Crea ub modelo secuencial
model = keras.models.Sequential()

# Crea una capa convolucional de 32 filtros de 3x3 con una entrada
directa de 28x28x1 (foto blanco y negro)
model.add(keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(28, 28, 1)))

# Toma los valores máximos por cada grid 2x2
model.add(keras.layers.MaxPooling2D((2, 2)))

# Aplana las features
model.add(keras.layers.Flatten())

# Red neural densa
model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dense(10, activation='softmax'))

# Compilar el modelo
model.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
```

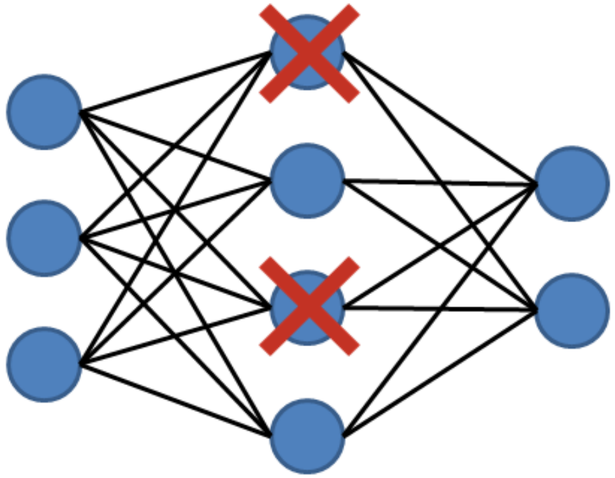
OVERFITTING Y UNDERFITTING

MÉTODOS DE REGULARIZACIÓN

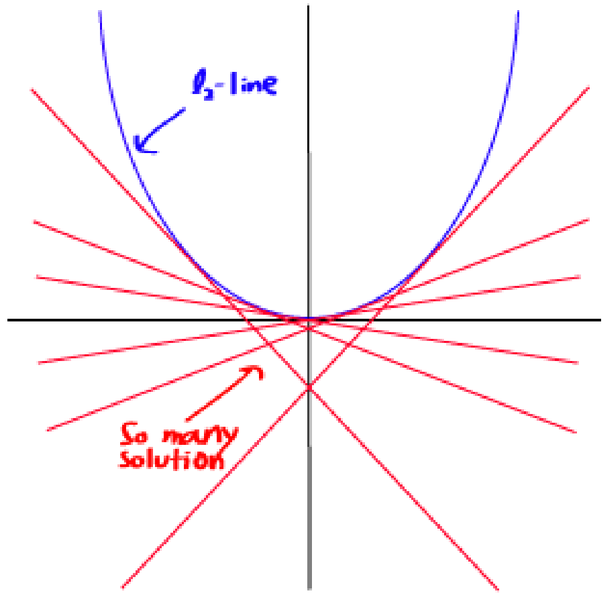
DETENCIÓN



DROPOUT



NORMA LÍMITE



CONCLUSIONES

REDES NEURALES

Generan salidas multiplicando y sumandos resultados de una capa a otra hacia adelante y aprenden retroprapagando el error

REDES CONVOLUCIONALES

Los filtros obtienen las features a través de la convolución con las entradas, luego las features son pasadas a una red densa

REGULARIZACIÓN

Para evitar sobreajuste usamos técnicas de regularización como dropout y la detención temprana