

Deep Learning: Percepción Computacional, Foundation Models y Transfer Learning

Inteligencia Artificial e Ingeniería del Conocimiento

Constantino Antonio García Martínez

Universidad San Pablo Ceu

Maneras de percibir el entorno, tema concentrado en imágenes

Introducción a las ConvNets

Introducción a las ConvNets

Aplicaciones Actuales

Aplicaciones Actuales de las ConvNets

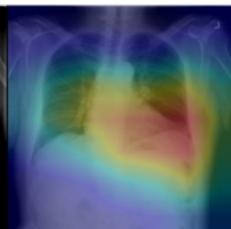
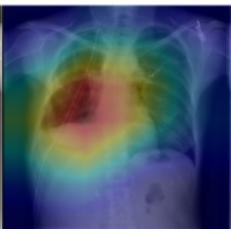
Las **ConvNets** son fundamentales para las tareas modernas de visión por computador:

- **Imagen Médica:** Detección de enfermedades en rayos X, resonancias magnéticas y TACs
- **Vehículos Autónomos:** Detección de objetos, detección de carriles y comprensión de escenas
- **Control de Calidad:** Detección de defectos de fabricación e inspección de productos
- **Seguridad:** Reconocimiento facial, sistemas de vigilancia y detección de anomalías
- **Aplicaciones Móviles:** Modo retrato, filtros faciales y mejora de imágenes
- **Foundation Models:** Componente clave en transformers de visión (ViT) y modelos multimodales

Aplicaciones en Detalle

- Imagen Médica

Saliency maps
Negative  Positive



(a) True positive, $p=0.9977$

(b) False positive, $p=0.9040$



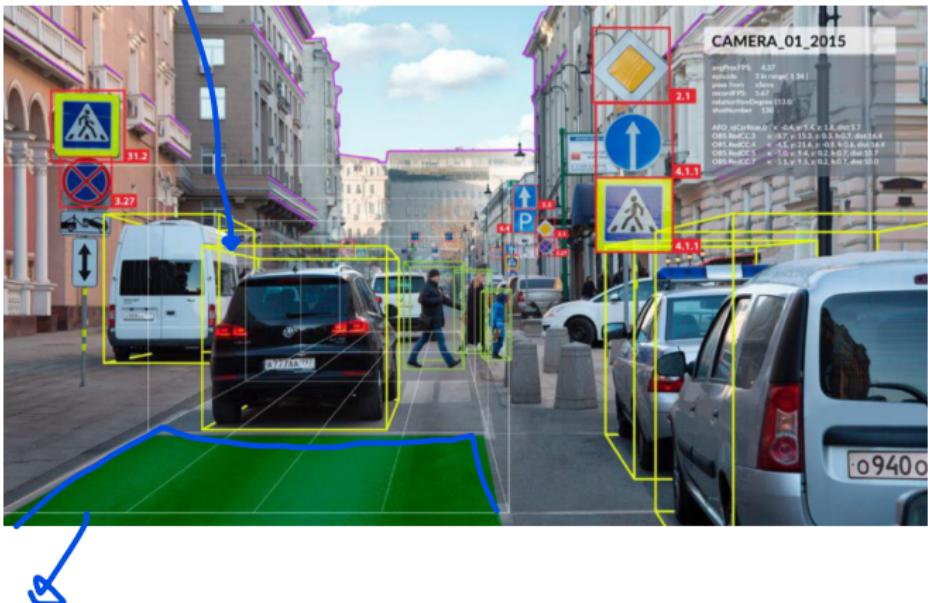
(c) True negative, $p=0.0506$

(d) False negative, $p=0.1538$

Aplicaciones en Detalle

Cuadrados: object detection

- Vehículos Autónomos



Zonas de la imagen:
segmentacion

Introducción a las ConvNets

Entendiendo las Imágenes

¿Qué es una imagen?

Numeros altos ==> indican zonas de blanco, numeros bajos ==> zonas de negros

Las imágenes en escala de grises son simplemente matrices $H \times W$ de valores en el rango [0, 255]. Se codifican usando 8 bits



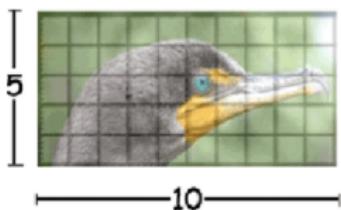
197	182	174	168	160	152	138	151	172	161	180	166	156				
156	180	169	74	76	62	99	37	116	210	180	166	154				
180	186	90	14	54	6	10	33	48	186	186	181					
206	103	6	122	191	111	120	204	161	15	56	180					
194	64	197	28	237	249	239	238	227	87	73	201					
172	126	267	23	239	214	220	239	238	16	74	206					
188	84	179	209	185	216	211	158	136	75	20	169					
189	37	164	94	10	158	134	11	31	47	32	148					
179	166	191	193	158	237	178	143	182	136	36	190					
205	174	166	282	236	237	149	178	226	43	35	234					
190	214	216	149	236	187	89	150	79	39	238	241					
190	224	147	104	237	212	127	102	36	101	208	224					
190	214	173	54	103	143	94	90	2	199	249	215					
187	196	236	75	1	81	47	0	6	217	250	211					
183	202	237	145	0	8	12	108	209	136	240	236					
195	206	123	209	177	121	125	200	176	19	96	218					

157	183	174	168	160	152	138	151	172	161	180	166	156				
195	182	163	74	76	62	33	17	110	210	180	181					
180	180	66	14	34	6	16	33	49	106	189	181					
206	103	6	122	191	111	120	204	161	15	56	180					
194	64	197	28	237	249	239	238	227	87	73	201					
172	106	207	239	239	230	214	230	230	238	98	74	206				
188	84	179	209	185	216	211	158	136	75	20	169					
189	37	164	94	10	158	134	11	31	47	32	148					
179	166	191	193	158	237	178	143	182	136	36	190					
205	174	166	282	236	237	149	178	226	43	35	234					
190	214	216	149	236	187	89	150	79	39	238	241					
190	224	147	104	237	212	127	102	36	101	208	224					
190	214	173	54	103	143	94	90	2	199	249	215					
187	196	236	75	1	81	47	0	6	217	250	211					
183	202	237	145	0	8	12	108	209	136	240	236					
195	206	123	209	177	121	125	200	176	19	96	218					

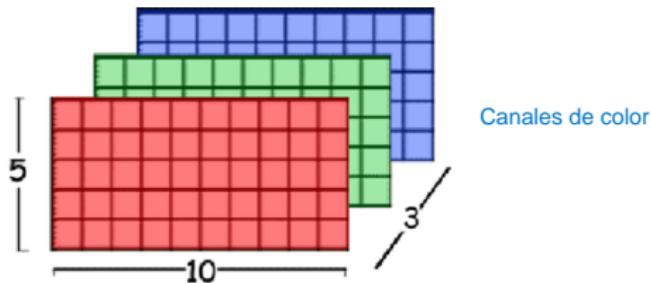
¿Qué es una imagen?

Imagen a color son tres matrices

Las imágenes en color son 3 matrices $H \times W$ apiladas con valores en el rango $[0, 255]$. Cada matriz es un **canal de color**. Alternativamente, las imágenes en color pueden verse como un **tensor** de tamaño $H \times W \times 3$.



Original Color Image



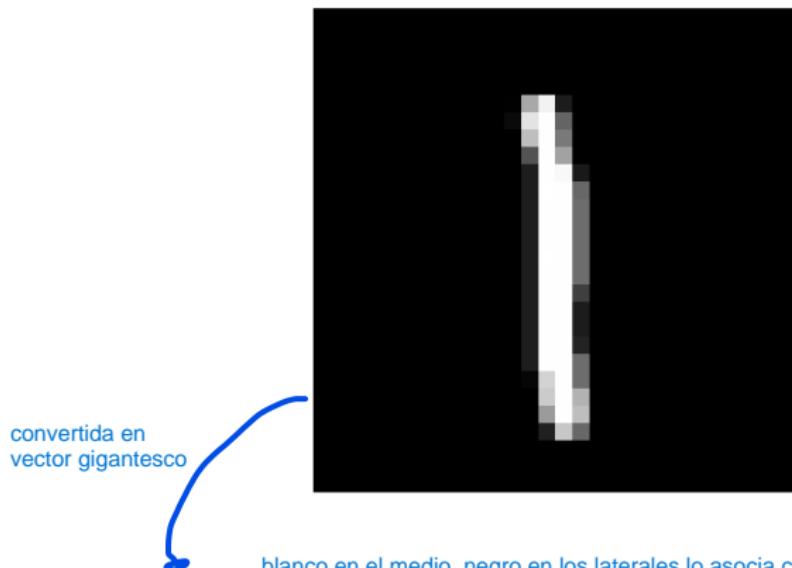
Matlab RGB Matrix

Fundamentos de las Convoluciones

Fundamentos de las Convoluciones

Conceptos Básicos

¿Por qué no usar redes densas?



Pero esto no funciona del todo bien

blanco en el medio, negro en los laterales lo asocia con un 1.



$$xw \downarrow$$

pesos mas bajos

negativos

$$xw \uparrow$$

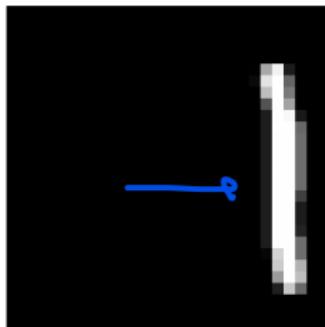
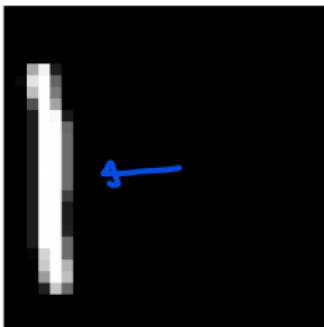
pesos altos

positivos

$$xw \rightarrow$$

por los pesos, sale una probabilidad muy cercana a 1 de que sea un 1

¿Por qué no usar redes densas?



acaba con un numero negativo para detectar un 1 porque no encaja con los pesos para la detección de 1



-20

Nos gustaría que la clasificación fuera invariante a la translación, rotación y escala.

Si la franja se mueve, los pesos asignados a detectar esos numeros no valen

Entendiendo las **Convoluciones** de Forma Intuitiva

es un gif...

Con la transformada de furier,
se puede hacer muy poco
costoso

Detective que busca patrones
buscando por toda la imagen,
hasta que encuentra un match.



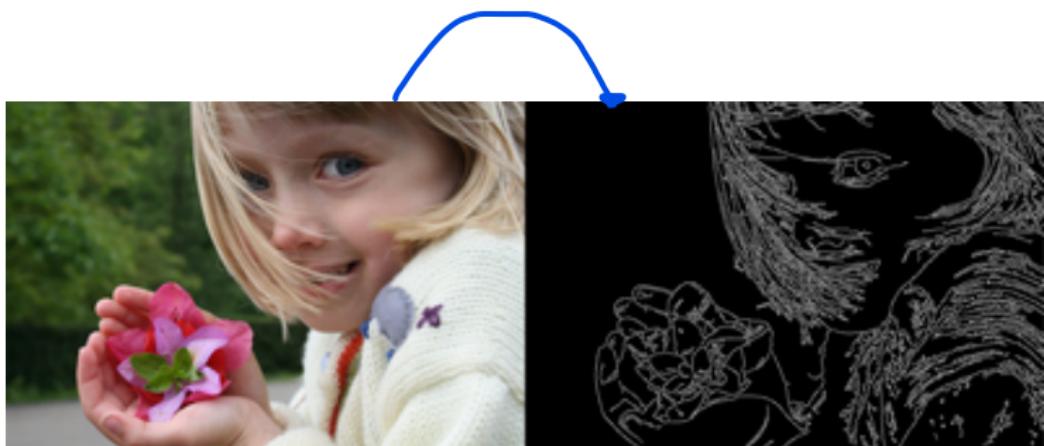
Outputea un num alto cuando
detecta un patron, cuando no
son numeros bajos

Piensa en la **convolución** como una **forma sistemática de buscar patrones**:

- El **filtro (kernel)** es como una **plantilla del patrón que buscamos**
- **Valores altos en la salida** = **fuerte presencia del patrón**

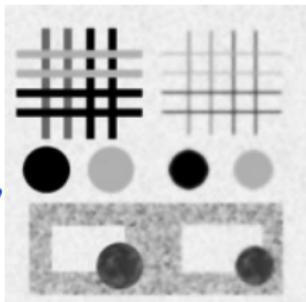
Convoluciones

Muy estudiado previo al deep learning

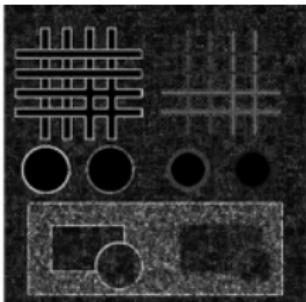


Convoluciones

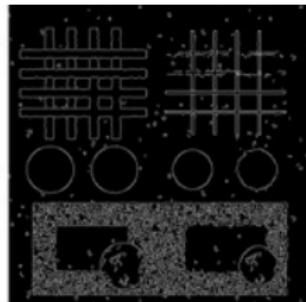
Los filtros han sido estudiados y diseñados manualmente durante mucho tiempo para detectar patrones en imágenes:



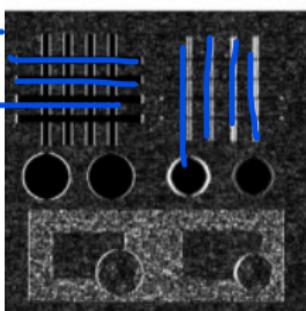
Original



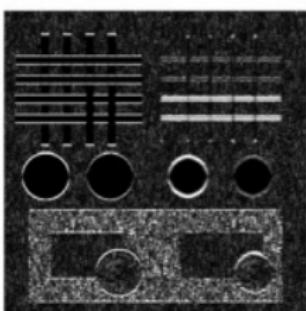
Laplacian



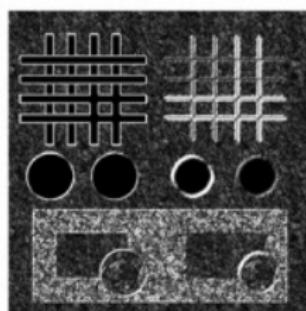
Canny



Sobel X
especializado en detectar
bordes en eje x



Sobel Y



Sobel X+Y

Convoluciones

Por ejemplo, el filtro de Prewitt para detectar bordes verticales es:

Los filtros son matrices

$$W = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

aprendemos los pesos de nuestros filtros

Con redes neuronales, la idea es que el filtro se aprende

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

Capas convolucionales de los filtros,

Las redes neuronales que usan convoluciones se denominan **redes neuronales convolucionales (ConvNets o CNNs)**.

Fundamentos de las Convoluciones

Capas Convolucionales

Con las convoluciones, se pierde size en la imagen original.

Si se le aplica a una img 5x5 un filtro 3x3 deberia salir mucho mas peq la salida, pero usamos 0-pairing para llenar y hacer la img original mas grande en prevision de que se va a perder size

Cada uno de los filtros tiene el mismo num de canales que la img original.

Ademas de las matrices ponemos el bias (ordenado en el origen)

2 Filtros

b

2 Canales

La red neuronal decide los pesos

La zona roja representa que estoy aprendiendo dos filtros

representa la salida de la capa convulacional, pero como son dos filtros la salida va a tener dos canales/activaciones, uno por cada filtro

Capas Convolucionales

otro gif...

el gif es util...si
cargara

otro gif...

Capas Convolucionales

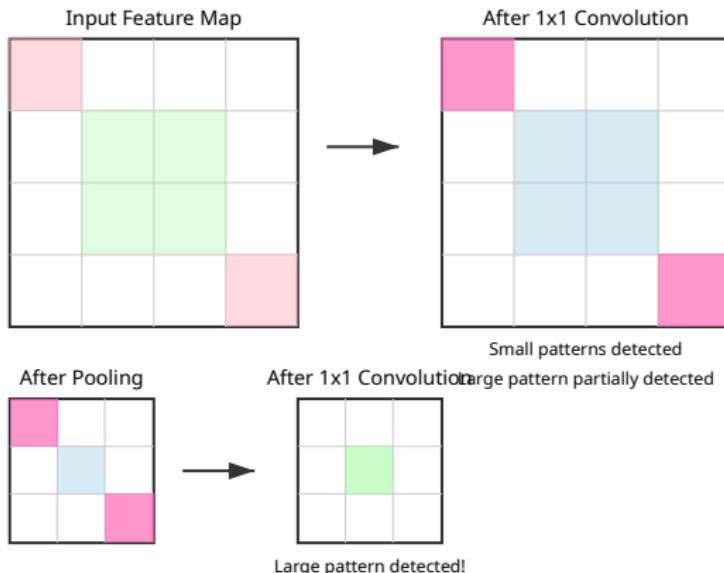
- Una capa convolucional acepta una “imagen” de tamaño $W_1 \times H_1 \times C_1$
- Requiere cuatro hiperparámetros:
 - Número de filtros F 32,64,128,...
 - Tamaño del kernel K (3x3)
 - Stride S → decide cuanto me muevo en el x e y (ej anterior, y lo mas comun, de uno en uno)
 - Cantidad de padding P 1
- Produce una nueva “imagen” de tamaño $W_2 \times H_2 \times C_2$ donde:
 - $W_2 = (W_1 - K + 2P)/S + 1$
 - $H_2 = (H_1 - K + 2P)/S + 1$
 - $C_2 = F$
- El canal d-ésimo de salida es el resultado de convolucionar el d-ésimo filtro sobre la “imagen” de entrada
- El número total de parámetros es $K \cdot K \cdot D_1 \cdot F$ pesos y F bias

Construcción de Arquitecturas ConvNet

Construcción de Arquitecturas ConvNet

Componentes Esenciales

¿Por qué Necesitamos Pooling?



Pooling Enables Multi-Scale Pattern Detection

Through hierarchical feature processing

¿Por qué Necesitamos Pooling?

Las capas de pooling sirven para múltiples propósitos:

- **Reducción de Dimensionalidad:**

- Reduce la carga computacional
- Disminuye el uso de memoria
- Hace los mapas de características más manejables

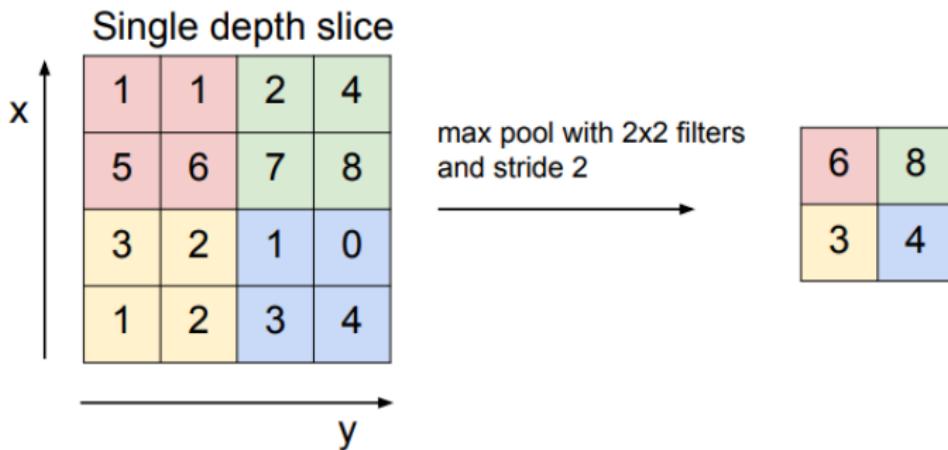
- **Invarianza a la Traslación:**

- Pequeños desplazamientos en la entrada producen la misma salida
- Ayuda a reconocer objetos independientemente de su posición exacta

- **Procesamiento Jerárquico:**

- Ayuda a capturar características a diferentes escalas
- Crea una jerarquía implícita de características

Capa de Pooling



Construcción de Arquitecturas ConvNet

Implementación Práctica

Implementación Práctica de ConvNets

Las ConvNets modernas típicamente siguen un patrón repetitivo:

1. Bloque Convolucional:

- Capa Conv2D (aprende características)
- BatchNormalization (estabiliza el entrenamiento)
- Activación ReLU (añade no-linealidad)
- MaxPooling (reduce dimensiones)
- Opcional: Dropout (previene sobreajuste)

2. Repetir bloques varias veces

3. Cabeza de Clasificación (Classification Head):

- Capa Flatten (convierte 2D a 1D)
- Capas Densas (clasificación final)
- Activación final (depende del tipo de problema)

Implementación Práctica de ConvNets

```
model = models.Sequential()

# First Convolutional Block
model.add(layers.Conv2D(32, (3, 3), padding='same', input_shape=(28, 28, 1)))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.25))

# Second Convolutional Block
model.add(layers.Conv2D(64, (3, 3), padding='same'))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.25))

# Other Convolutional Block
# ....

# Classification Head
model.add(layers.Flatten())
model.add(layers.Dense(128))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(10, activation='softmax'))
```

Implementación Práctica de ConvNets

Code Example: MNIST con ConvNets

Técnicas Avanzadas

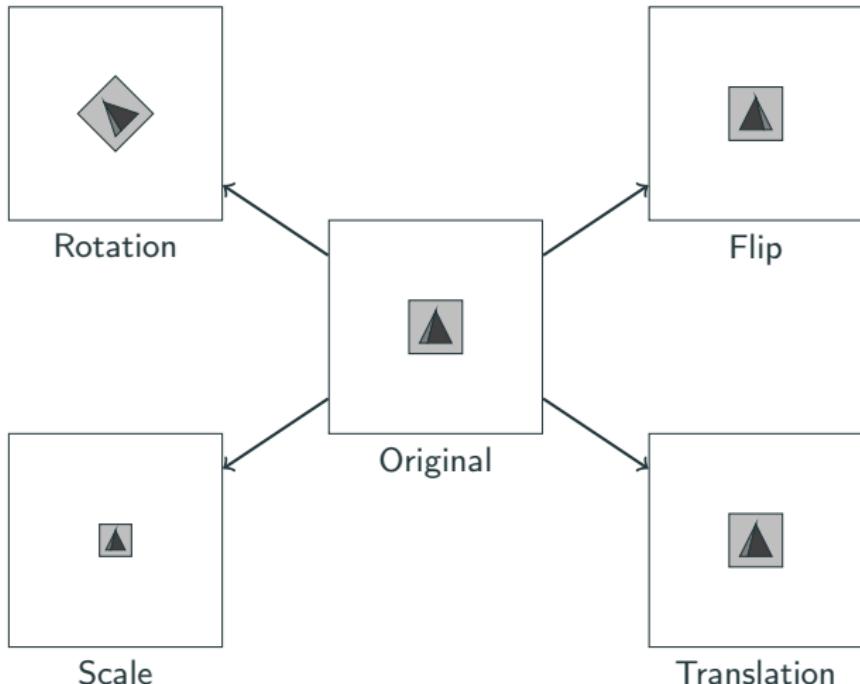
Técnicas Avanzadas

Data Augmentation

Data Augmentation

¿Por qué?

- Más datos de entrenamiento
- Mejor generalización
- Previene el sobreajuste



Técnicas de Data Augmentation

CIFAR10



IMAGENET



SVHN



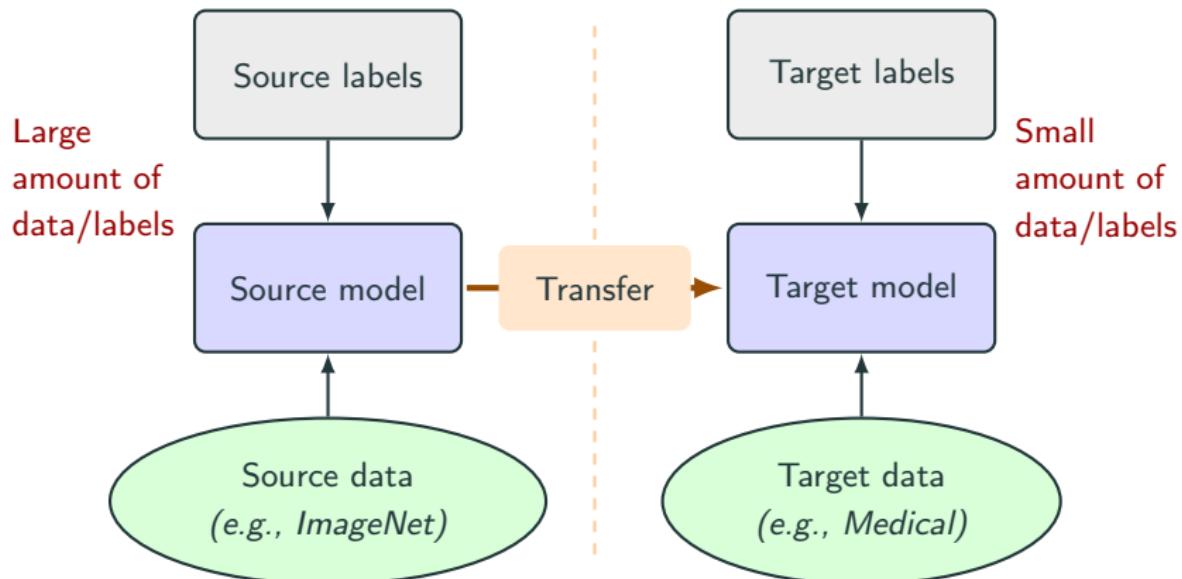
Data Augmentation

Code Example: Data Augmentation

Técnicas Avanzadas

Transfer Learning y Foundation Models

Transfer Learning



Tres estrategias:

Feature Extraction

Fine-tuning

Progressive Fine-tuning

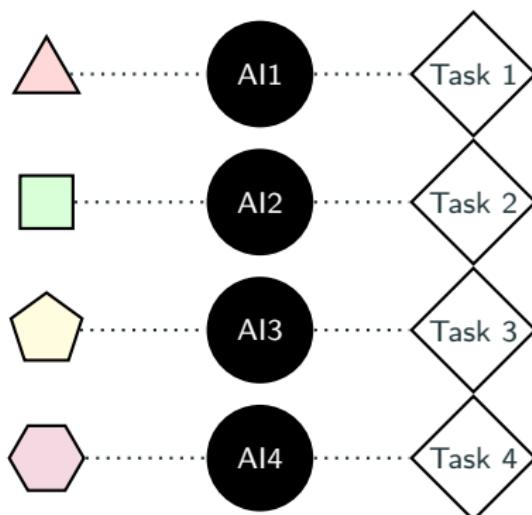
Foundation Models

Entrenados con conjuntos
de datos masivos

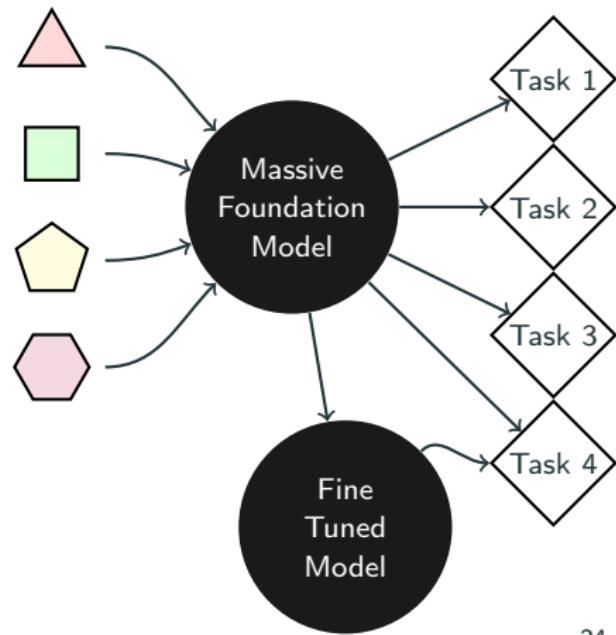
Entrenados en múltiples
tareas

Base para transfer
learning

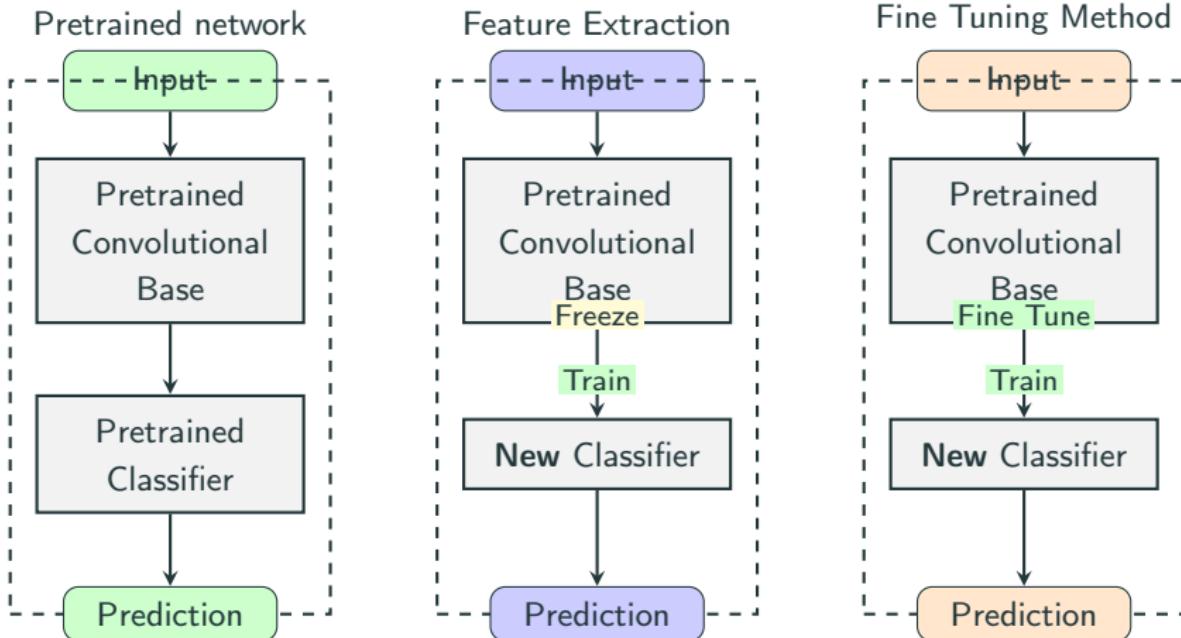
Traditional ML



Foundation Models



Estrategias de Fine Tuning



¿Cuándo Usar Cada Estrategia?

- **Feature Extraction:**
 - Dataset pequeño
 - Tarea similar
- **Fine-tuning:**
 - Más datos
 - Tarea diferente
- **Desde Cero:**
 - Dominio único
 - Dataset grande

Transfer Learning

Code Example: Transfer Learning

Code Exercise: ConvNets Prácticas