Dan Beninson
Instituto de Tecnología Nuclear

Universidad Nacional de San Martín

Comisión Nacional de Energía Atómica

Nociones de Aprendizaje Automático para Aplicaciones Nucleares

Fuera de programa...

Extras...

- Lo que se mostró en el curso les da una buena base para empezar a ver si alguna herramienta de aprendizaje automático les puede servir para atacar el problema con el que estén trabajando.
- Queríamos agregar un par de cosas prácticas adicionales que les pueden ser útiles cuando se pongan a trabajar.

Fine-tuning

- Las redes neuronales aprenden representaciones complejas que pueden generalizar a muchas tareas.
- Sin embargo, cuando los nuevos datos tienen una distribución distinta (otro dominio, sensor, idioma, etc.) el modelo puede perder rendimiento.
- En lugar de entrenar desde cero, podemos reutilizar un modelo ya entrenado y ajustarlo para el nuevo conjunto de datos.
- Este proceso se denomina *fine-tuning* o ajuste fino y permite adaptar un modelo *preentrenado* a una nueva tarea preservando el conocimiento previo y reduciendo el costo de entrenamiento.

Fine-tuning

- Etapas del Fine-Tuning
  - Se toma un modelo pre entrenado (por ejemplo, en ImageNet o un corpus general de texto).
  - Se congelan las primeras capas, que capturan rasgos generales (bordes, texturas, sintaxis...).
  - Se entrenan sólo las últimas capas (o agregar una salida nueva) sobre los datos específicos del nuevo dominio.
  - Se usa un learning rate bajo para no destruir el conocimiento ya adquirido (*catastrophic forgetting*)(*).
  - Cuanto más parecido sea el nuevo dominio al original, menos capas deben reentrenarse.
  - Si las tareas son más diferentes, se pueden descongelar más capas progresivamente.

(*) Otra opción puede ser entrenar toda la red, pero siempre con un *LR* pequeño. A modo de referencia el valor por default de Adam es 0.001

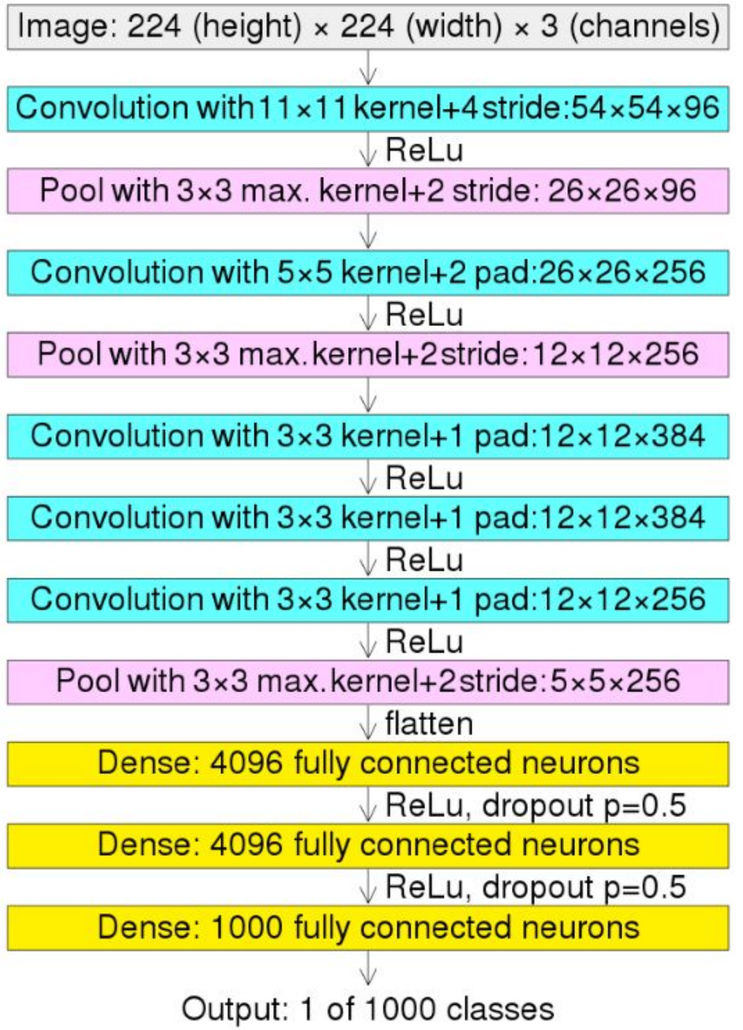
Transfer Learning

- Cuando el cambio en los datos y objetivos es más significativo, pero sigue existiendo una relación con la tarea original, se puede emplear *transfer learning*.
- Esto es particularmente útil en redes convolucionales, donde las primeras capas capturan características generales que pueden ser útiles en múltiples dominios.
- Cuando hablamos de redes convolucionales dijimos cosas como:
  - “Las primeras capas de la red detectan patrones simples”.
  - “Los filtros se combinan y las capas más profundas detectan patrones más complejos”.
- En el notebook pueden hacer ejemplos y ver que funcionan, pero medio que sigue siendo una cuestión de fe.
- Vamos a mostrar primero un trabajo muy conocido que justamente ataca este problema, ¿cómo interpretar lo que está pasando dentro de una *CNN*.

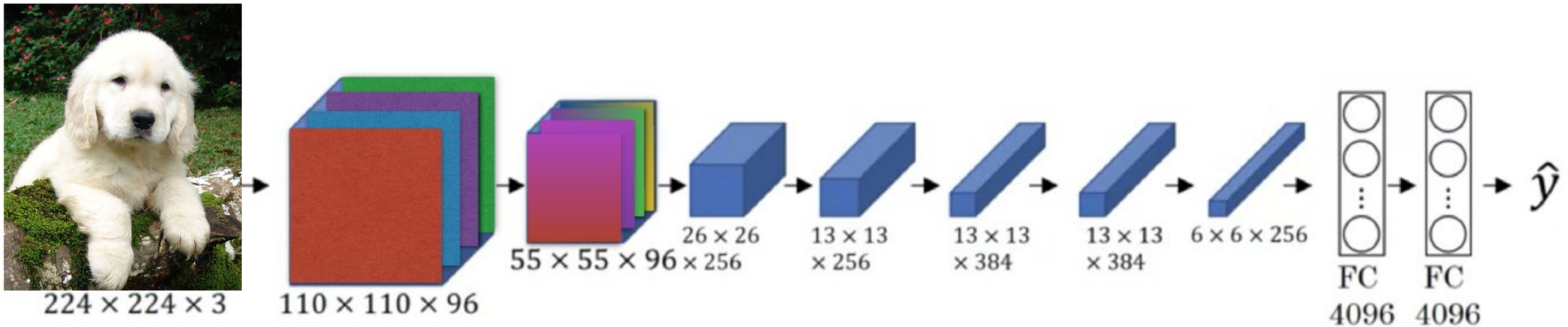
AlexNet

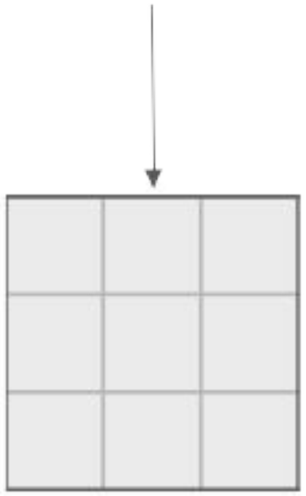
- Entrenada con más de un millón de imágenes.
- Clasifica 1000 categorías de objetos (como teclado, taza de café, lápiz y muchos animales).
- Compitió en ImageNet Large Scale Visual Recognition Challenge en 2012 y alcanzo un error de 15.3%, más de 10.8 puntos menos que el segundo.
- Una de las primeras en usar GPUs para la competencia.
- Superada en 2015 por una red de Microsoft que tenía más de 100 capas.
- En 2013, Visualizing and Understanding Convolutional Networks (https://arxiv.org/abs/1311.2901).

Fuente: Wikipedia



AlexNet





Toman los 9 parches de pixeles que más activan esta neurona (filtro)

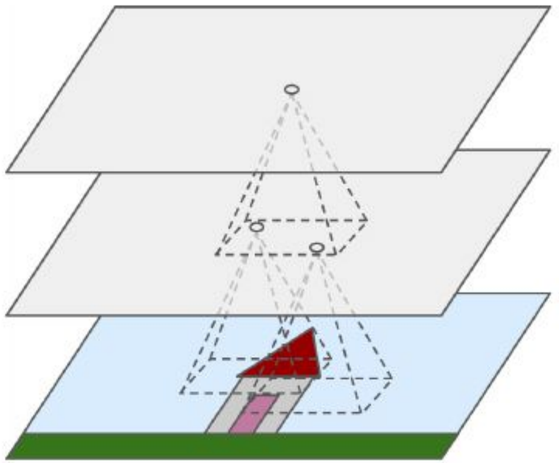


Imagen del Gerón

AlexNet

Neurona
(filtro 1)

AlexNet

Neurona (filtro 1)

Neurona (filtro 2)

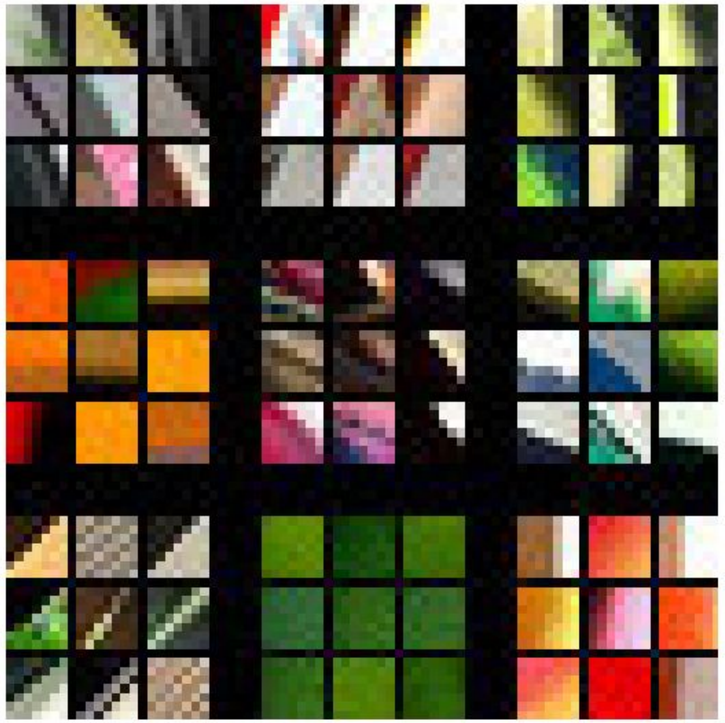
AlexNet

Neurona (filtro 1)

Neurona (filtro 2)

Neurona (filtro 3)

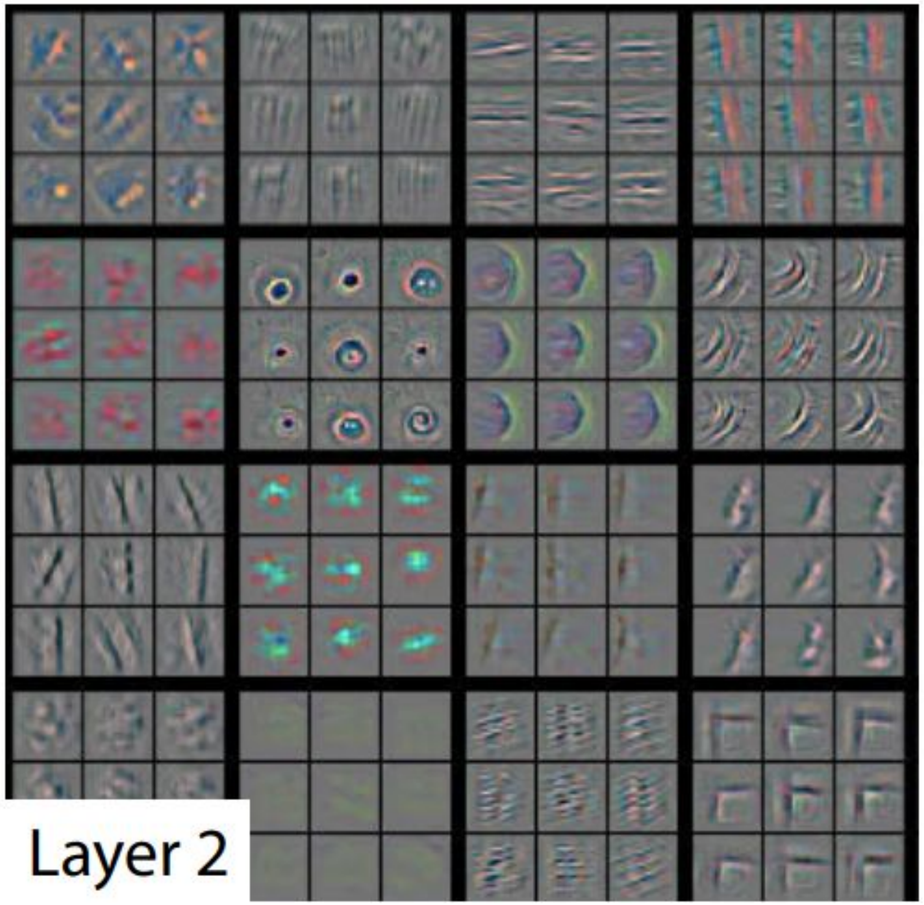
...



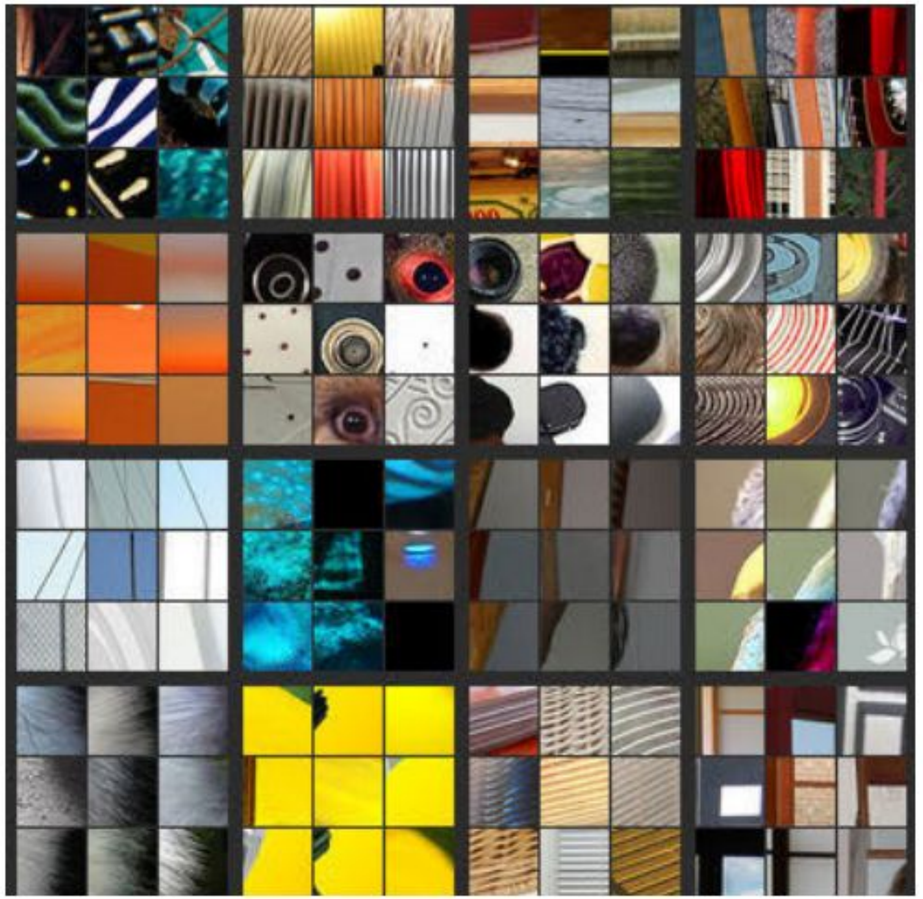
...

Neurona (filtro 9)

AlexNet

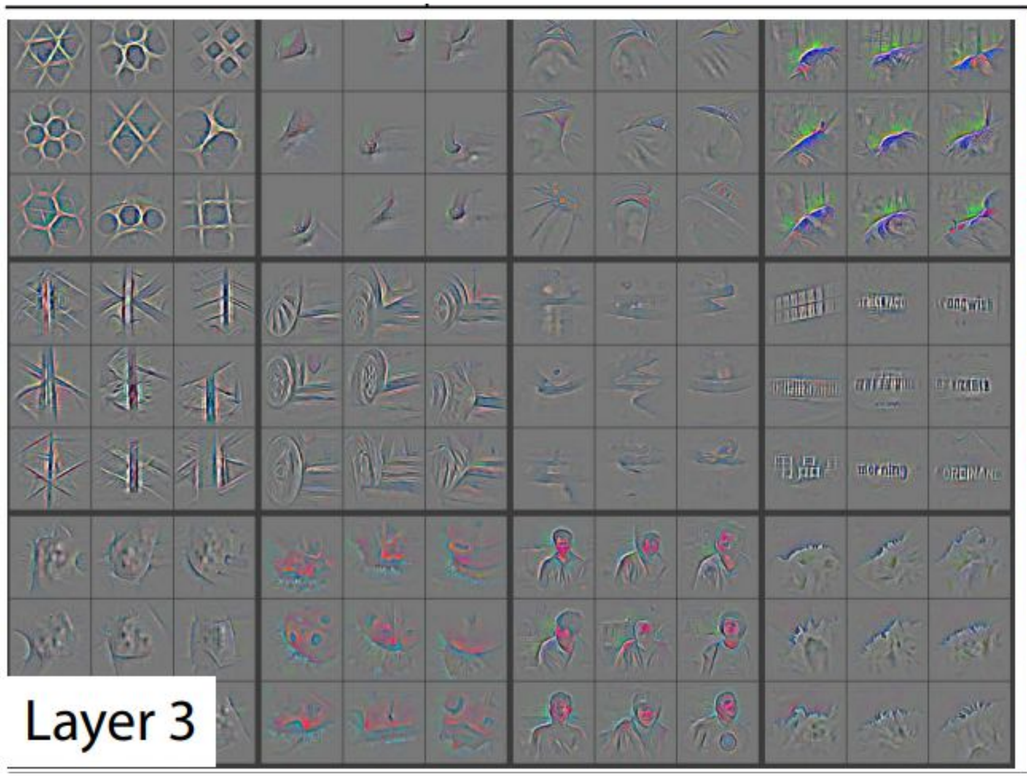


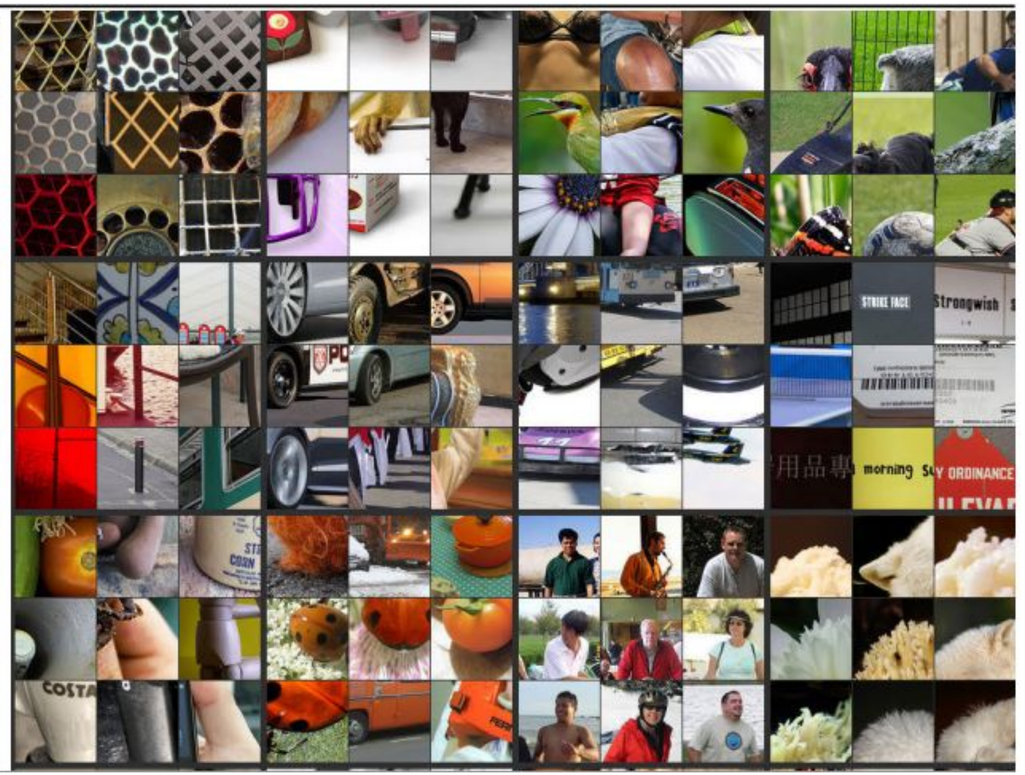
Patrones que maximizan la activación



Ejemplos de imágenes que maximizan la activación

AlexNet

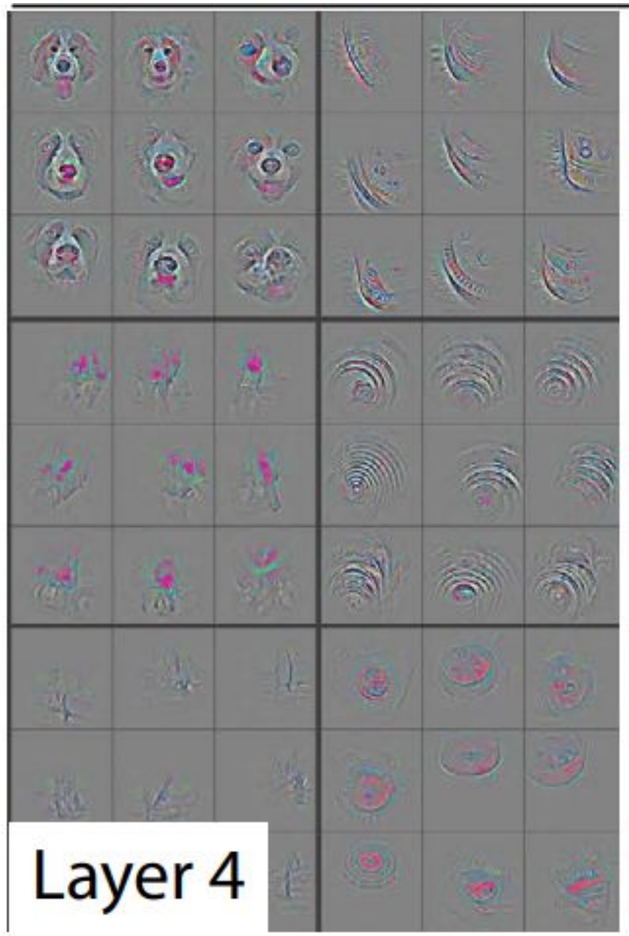




Patrones que maximizan la activación

Ejemplos de imágenes que maximizan la activación

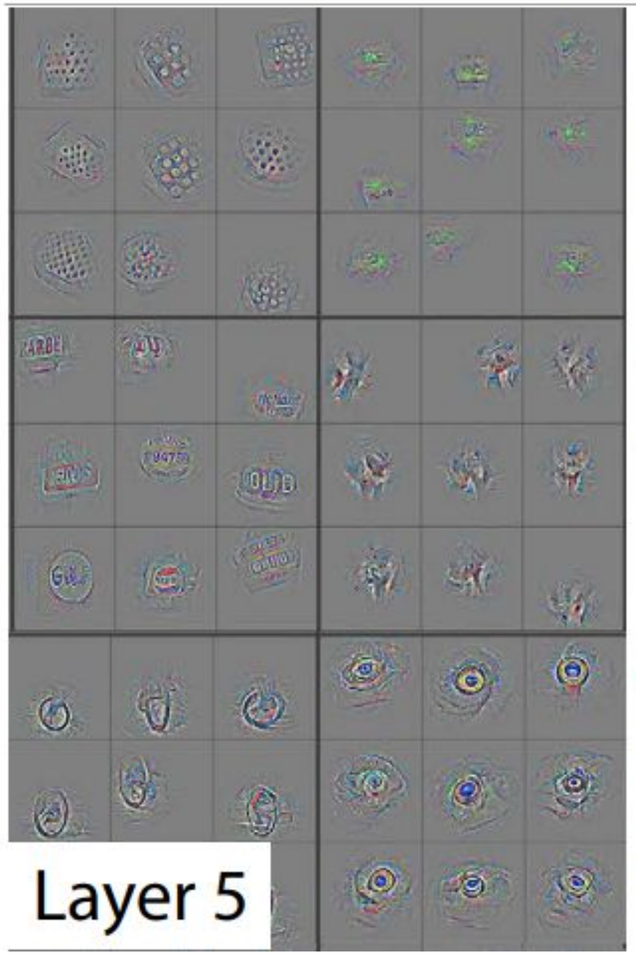
AlexNet



Patrones que maximizan la activación

Ejemplos de imágenes que maximizan la activación

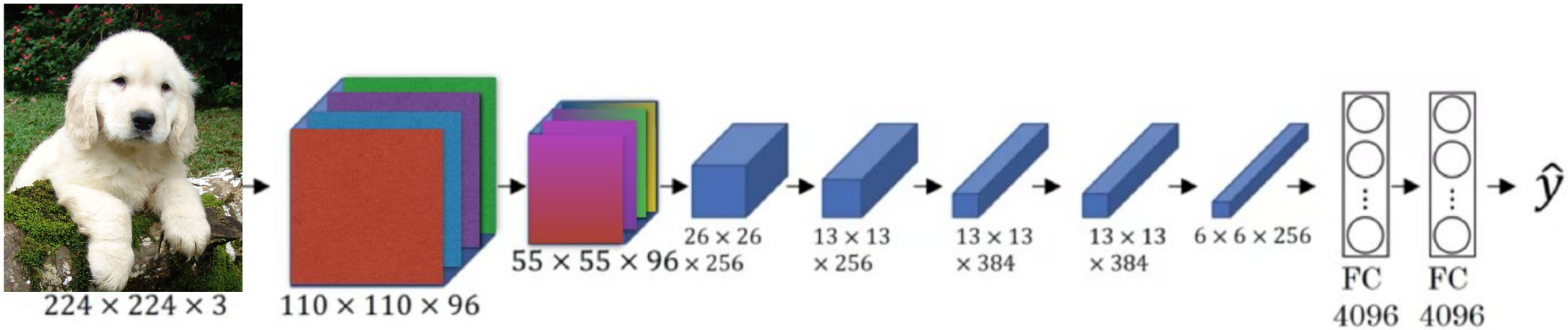
AlexNet



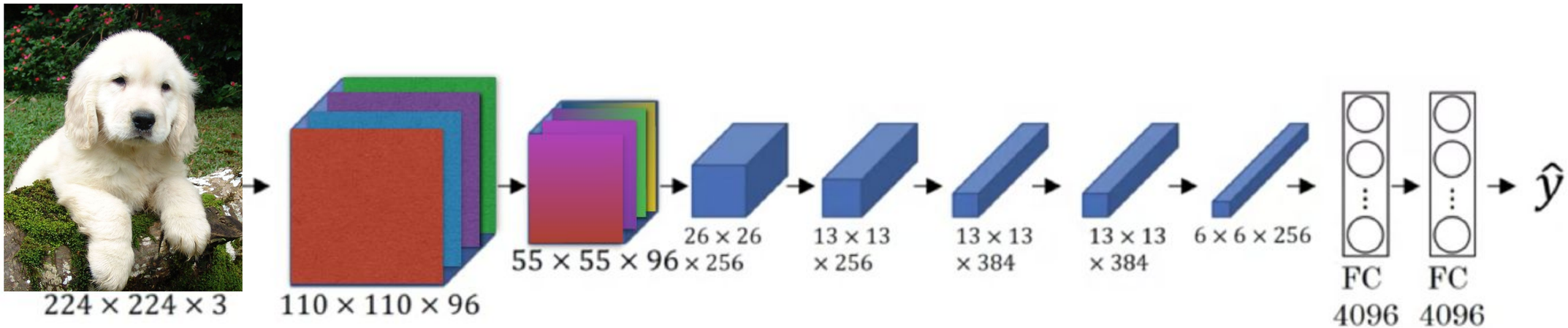
Patrones que maximizan la activación

Ejemplos de imágenes que maximizan la activación

AlexNet


224 × 224 × 3
110 × 110 × 96
55 × 55 × 96
26 × 26 × 256
13 × 13 × 256
13 × 13 × 384
13 × 13 × 384
6 × 6 × 256
FC 4096
FC 4096
$\hat{y}$

AlexNet



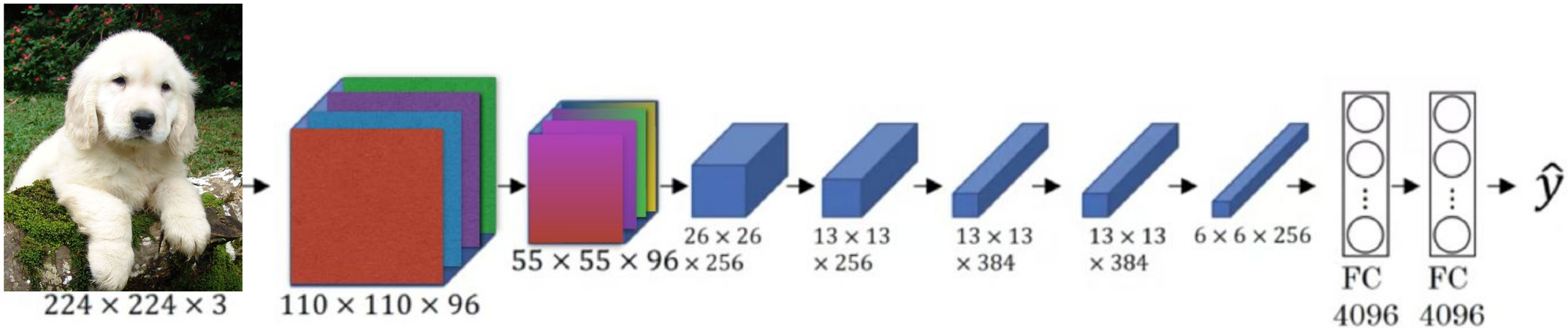
Bordes / Tonos

Formas / Texturas

Objetos / Atributos

Clasificación

AlexNet



Bordes / Tonos

Formas / Texturas

Objetos / Atributos

Clasificación

¿Qué partes les parece que pueden servir si queremos, por ejemplo, armar un clasificador que diferencia nuestra fauna autóctona de otros tipos de Caranchos?

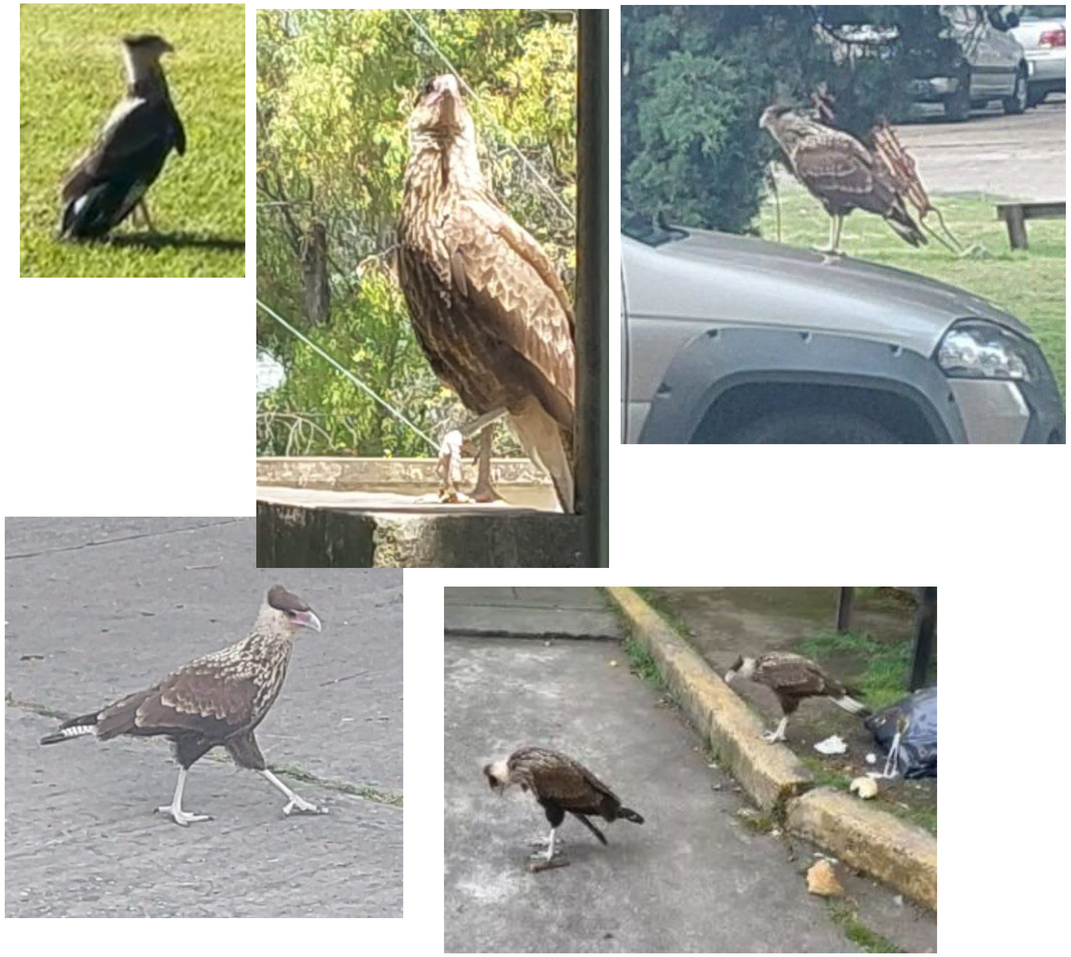
Caranchus Constituyentis (Fito)

Caranchus Chetus

Transfer learning

- Las primeras capas de la red funcionan como detectores de características simples que son generales a todas las imágenes.
- Las más profundas ya son de objetos y atributos particulares de las imágenes con las que fue entrenada.
- Veamos como podemos aprovechar todo esto para hacer un clasificador de nuestros caranchos.
- Primero: nos armamos un dataset de imágenes para nuestro problema en particular.
- Segundo: nos conseguimos una AlexNet entrenada.

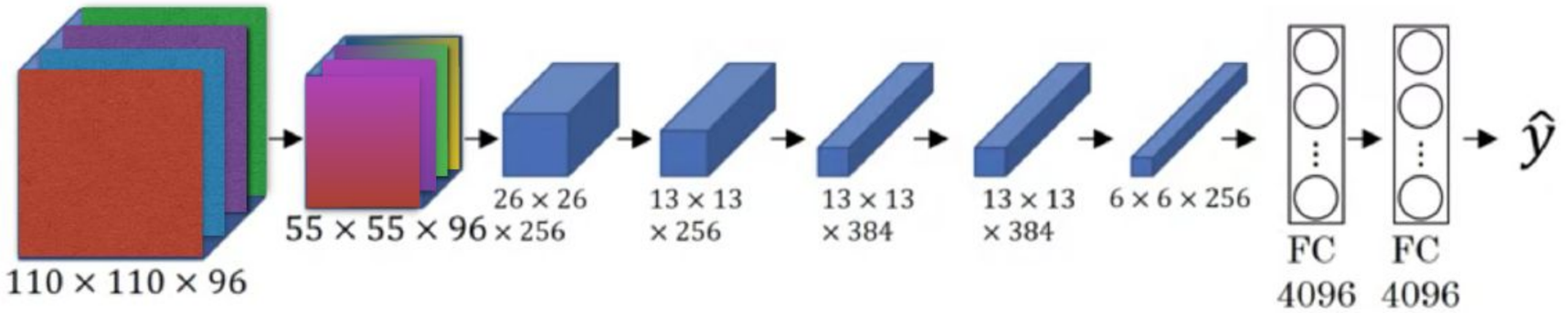
Transfer learning



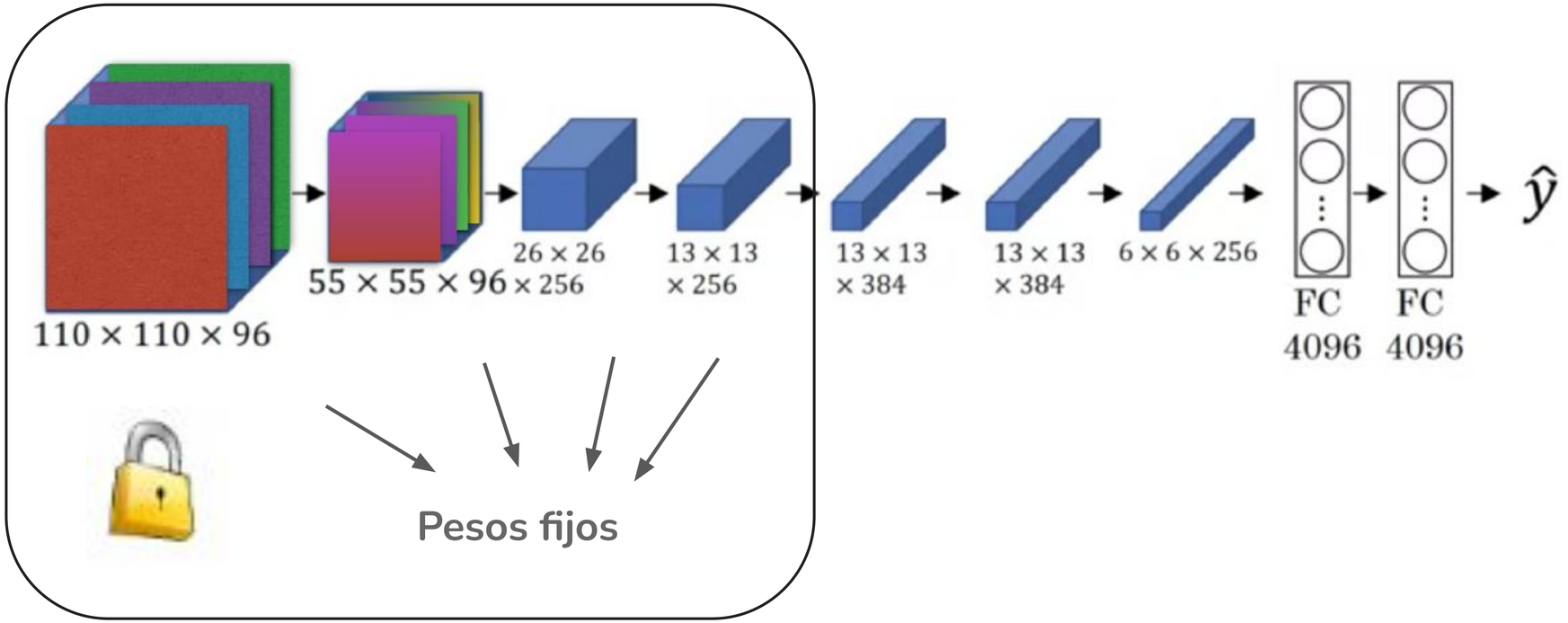
Label = 1

Label = 0

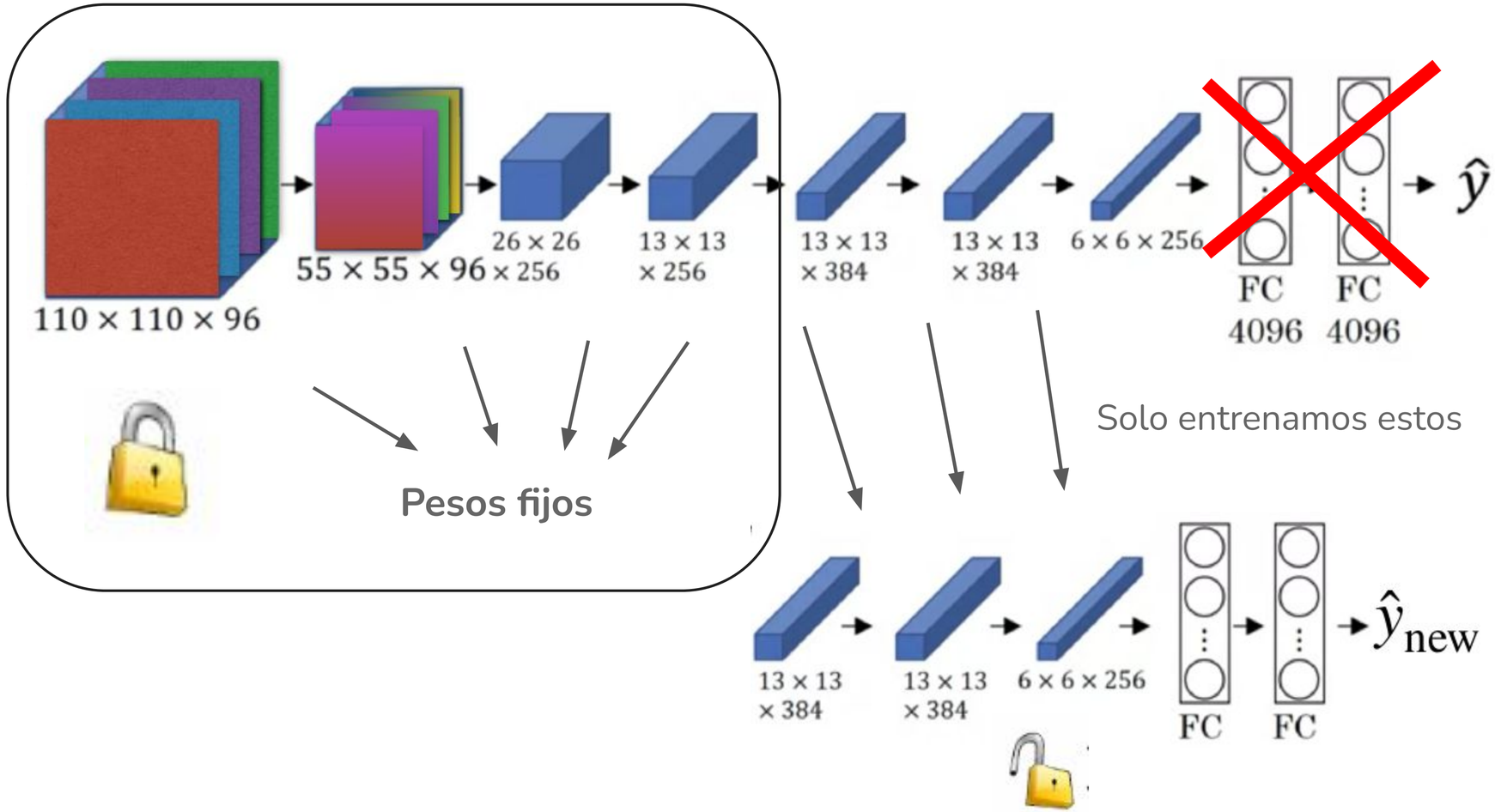
Transfer learning



Transfer learning

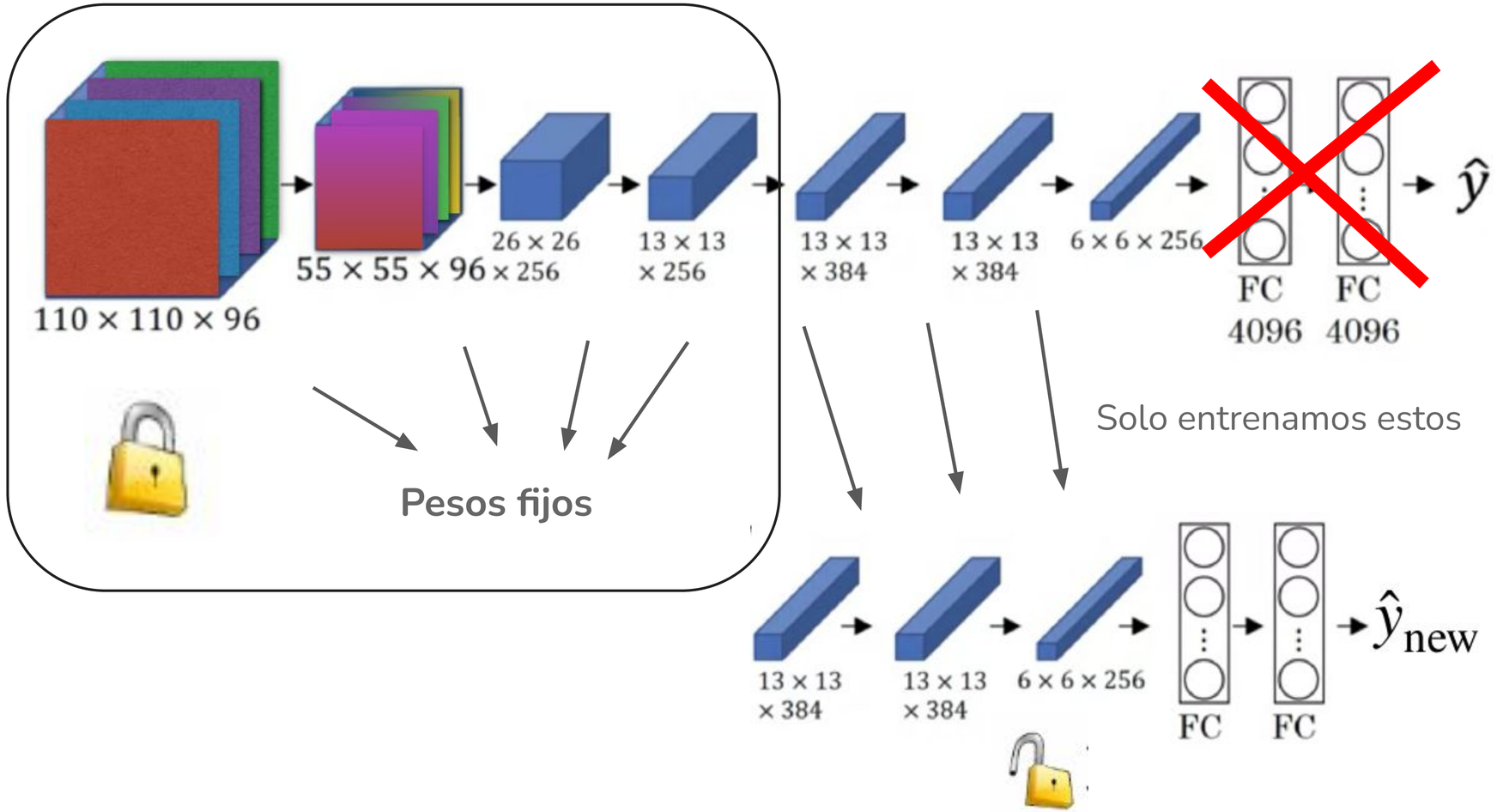


Transfer learning



Transfer learning

...



Transfer learning

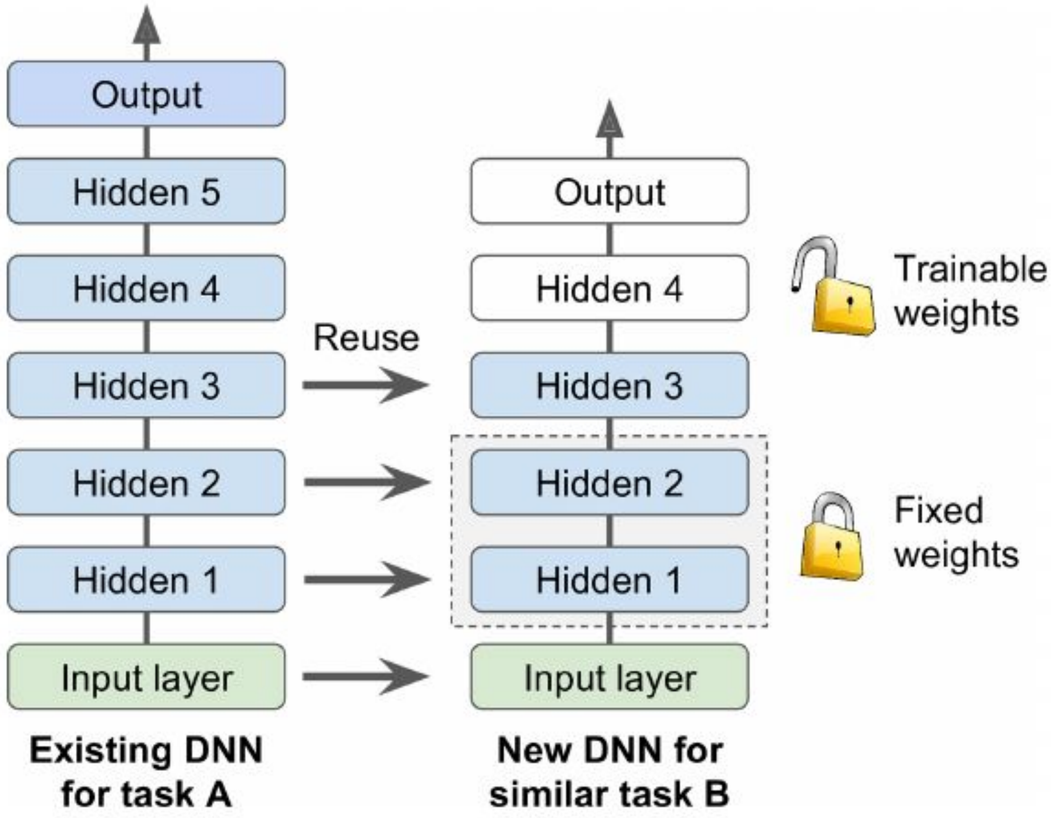


Imagen del Gerón

- La utilización de modelos pre entrenados para tareas más específicas es una gran forma de ahorrar datos y recursos.
- No solo se puede cambiar la capa de salida, se puede hacer todo un modelo nuevo reutilizando lo que nos sirva.
- Visión por computadora multipropósito: ResNet-50, VGG-16, VGG-19, Xception, Inception (v3, v4), MobileNet
- Object Detection: COCO, YOLO
- Procesamiento del lenguaje natural
- Multipropósito: ULMFiT, Transformer, BERT, GPT-2,3,...
- Word-Embeddings: Word2Vec, ELMo, Flair, ...

Transfer learning en Keras

- Usar capas (y pesos) de otro modelo descartando la capa final y agregando otra/

```
model_A = keras.models.load_model("my_model_A.h5")
model_B_on_A = keras.models.Sequential(model_A.layers[:-1])
model_B_on_A.add(keras.layers.Dense(1, activation="sigmoid"))
```

- Fijar los pesos de algunas de las capas (acá, todas, menos la última) antes de compilar/

```
for layer in model_B_on_A.layers[:-1]:
    layer.trainable = False
model_B_on_A.compile(loss="binary_crossentropy", optimizer="sgd",
                     metrics=["accuracy"])
```

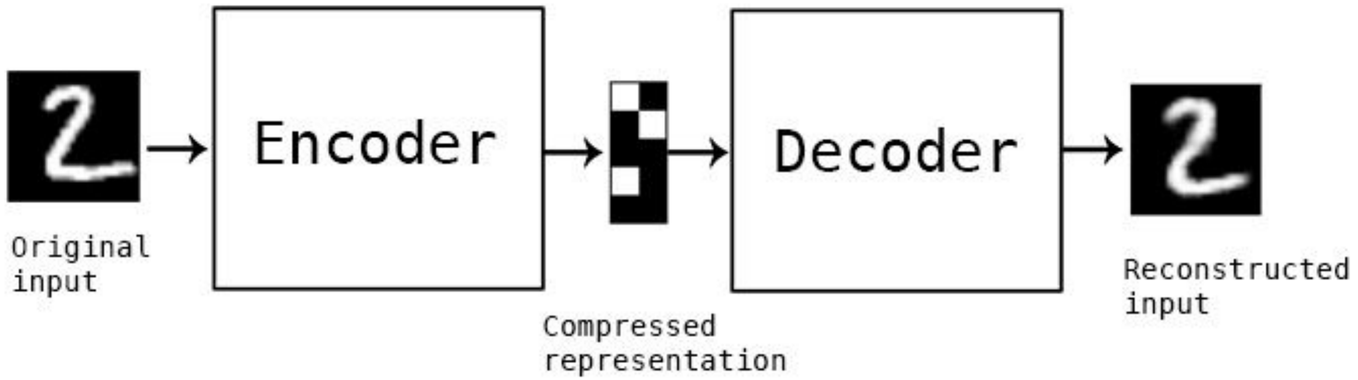
- Entrenar, descongelar algunas capas, entrenar, etc.
- Hay cientos de opciones ya cerradas que hacen todo esto aún más transparente, por ejemplo https://github.com/qubvel/segmentation_models.

Autoencoders

- A lo largo del curso exploramos distintos algoritmos y técnicas de aprendizaje.
- En particular, las redes neuronales que vimos hasta ahora se basaron en aprendizaje supervisado: su objetivo era mapear una entrada hacia una variable “target” conocida.
- No vamos a mencionar un nuevo tipo de red (o por lo menos en el sentido más estricto del término), vamos a ver una arquitectura junto a una forma de entrenamiento no supervisado (en realidad auto-supervisado). Los autoencoders

Autoencoders

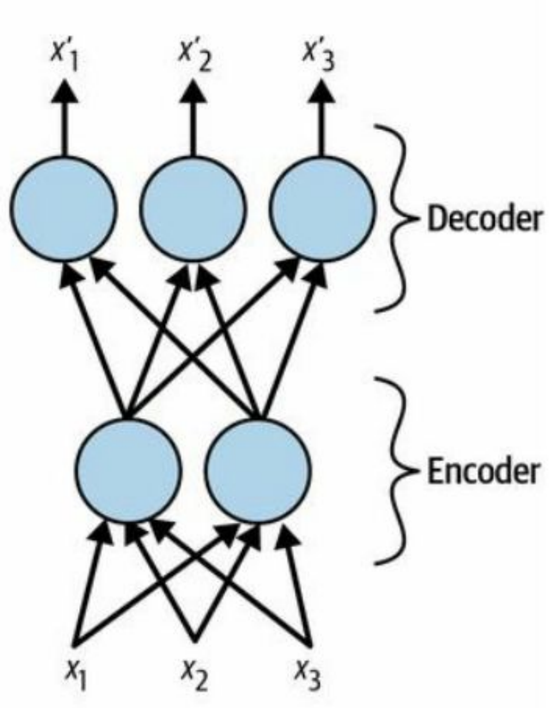
- Lo que busca un autoencoder es reproducir el dato de entrada.
- El truco está en que aprende a hacerlo pasando por un espacio de dimensión menor, o sea, codifica los datos en un **espacio latente** (que se puede lograr forzando un cuello de botella en la arquitectura) para luego decodificarlos en el espacio original buscando reconstruir el dato de entrada.



Fuente

Autoencoders

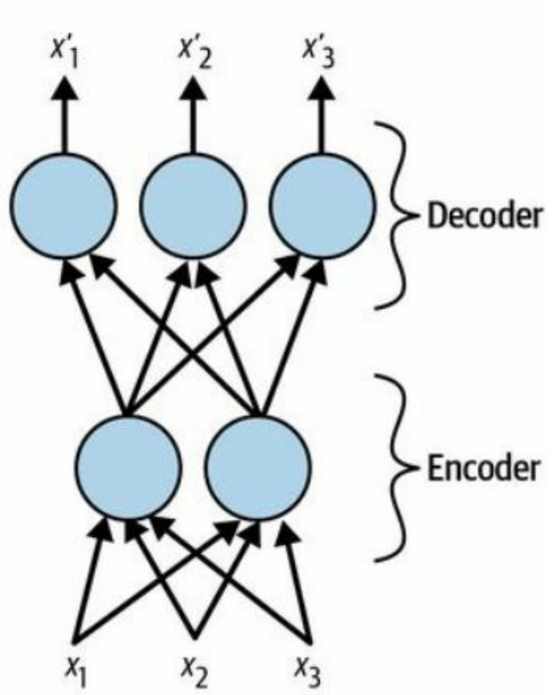
- En su forma más sencilla es una red totalmente conectada con una capa oculta:
  - La cantidad de neuronas de salida tienen que ser igual al tamaño del input
  - La cantidad de neuronas de la capa oculta tiene que ser menor que las de la salida.
- En el entrenamiento se pone como target a aprender al mismo dato que se le presentó a la red.
- De esta forma, cuando el autoencoder busca reproducir la entrada lo forzamos a comprimir la información aprendiendo sus características más relevantes.



Imágen del Géron

Autoencoders

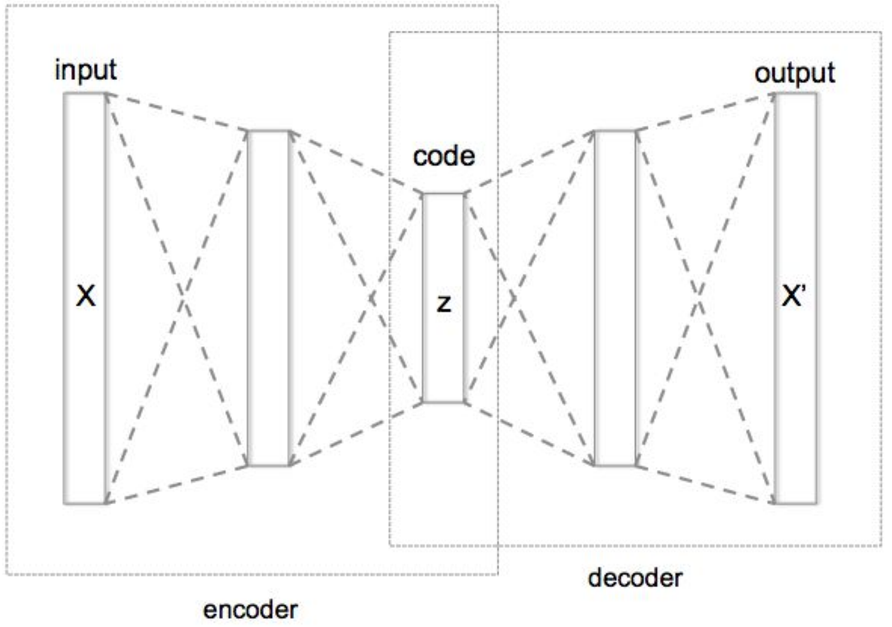
- Se van a distinguir dos partes de la red, la parte que efectivamente comprime la información llevándola al espacio latente y la parte que toma una representación en este espacio y busca regenerar el dato.
- La primera parte es el **encoder** y la segunda el **decoder**



Imágen del Géron

Autoencoders

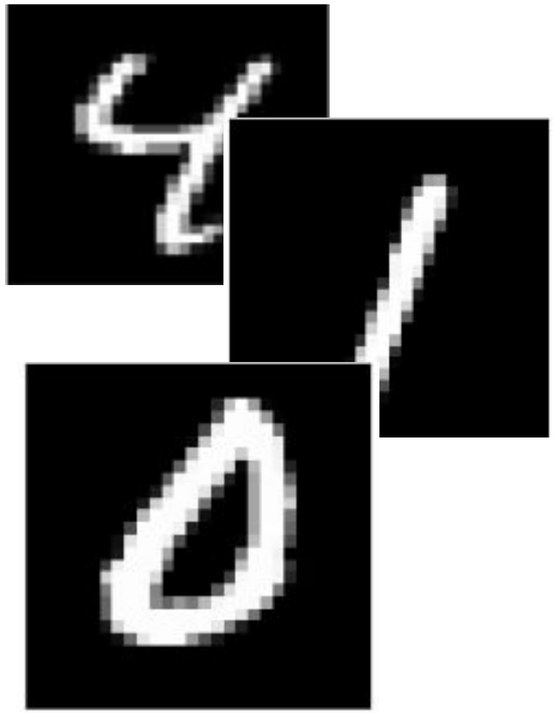
- Por supuesto, esto se puede generalizar haciendo arquitecturas más profundas.
- Esto se suele llamar Stacked Autoencoders o Deep Autoencoders.
- La idea es que añadir más capas permite aprender codificaciones más complejas.
- Ojo, hay que ser cuidadoso y no hacer al AE demasiado poderoso, no queremos que simplemente “mapee una entrada a una salida” queremos forzarlo a que **codifique** y **decodifique**.

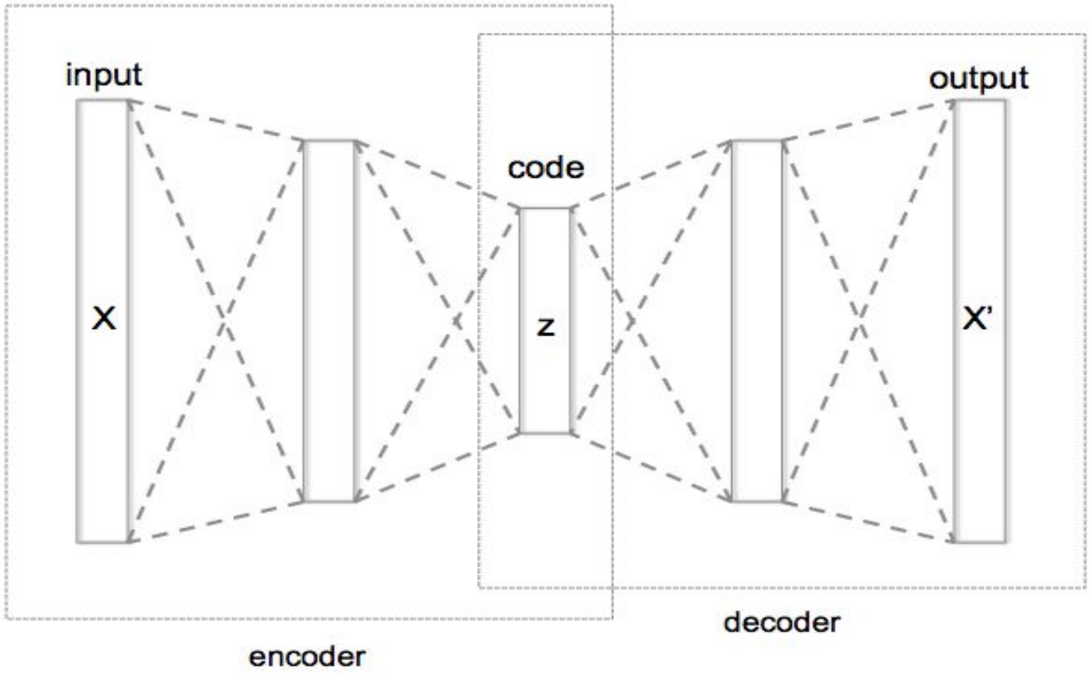


Imágen de wikipedia

Autoencoders

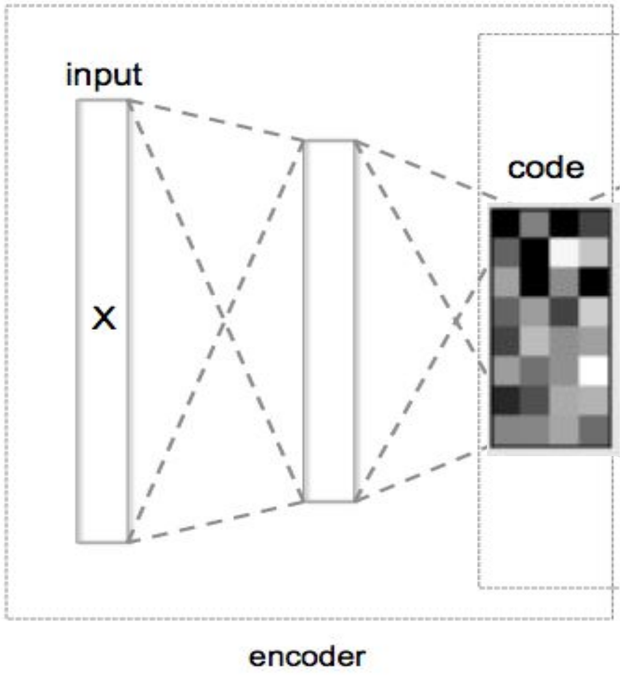
Entrenamiento...





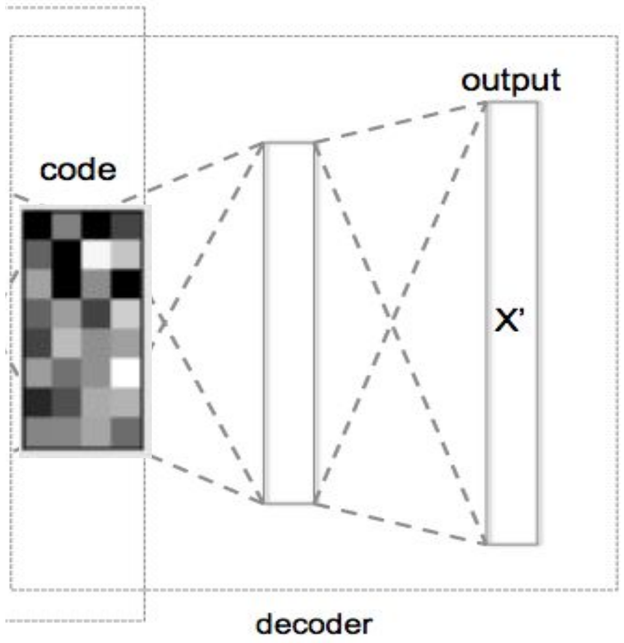
Autoencoders

Uso...



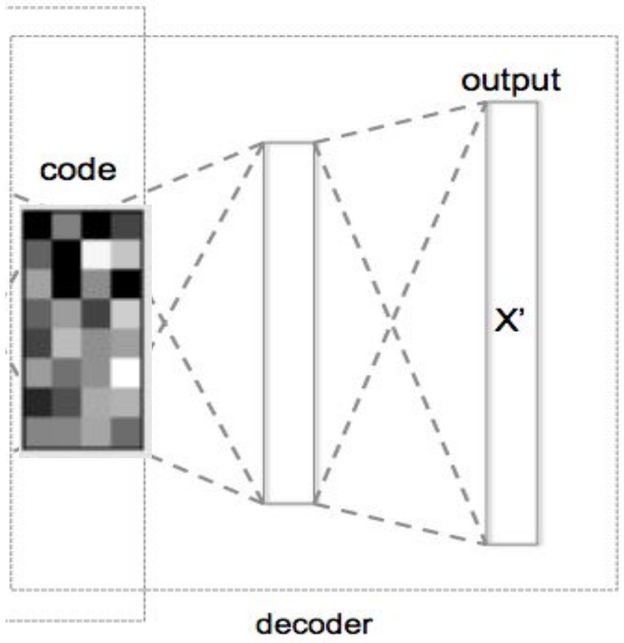
Autoencoders

Uso...



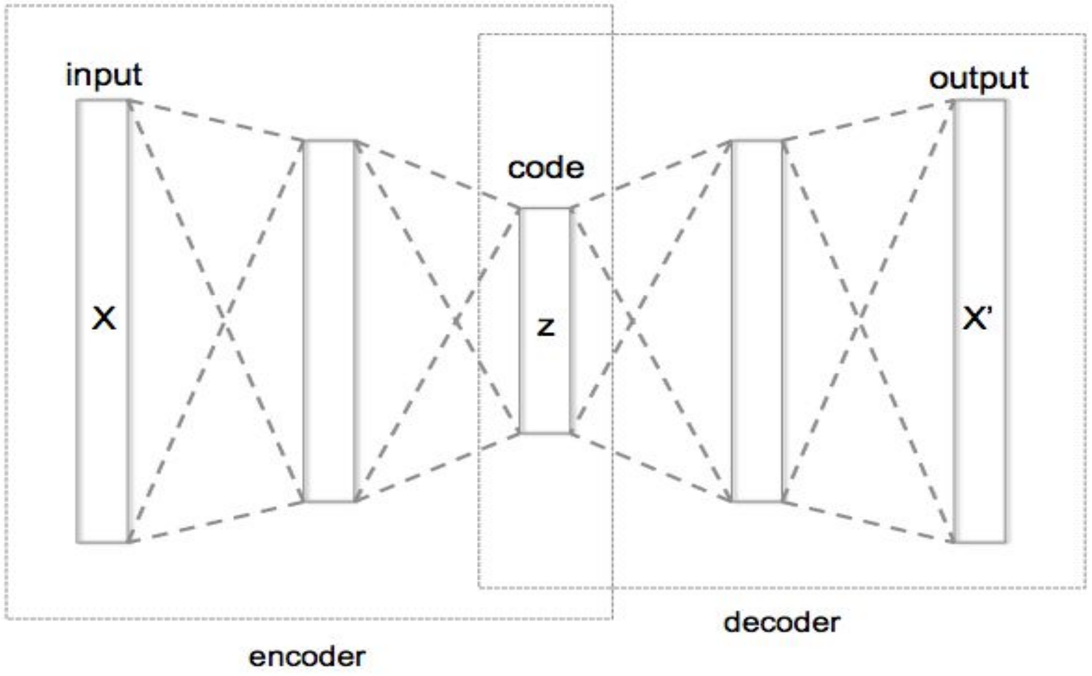
Autoencoders

Uso...



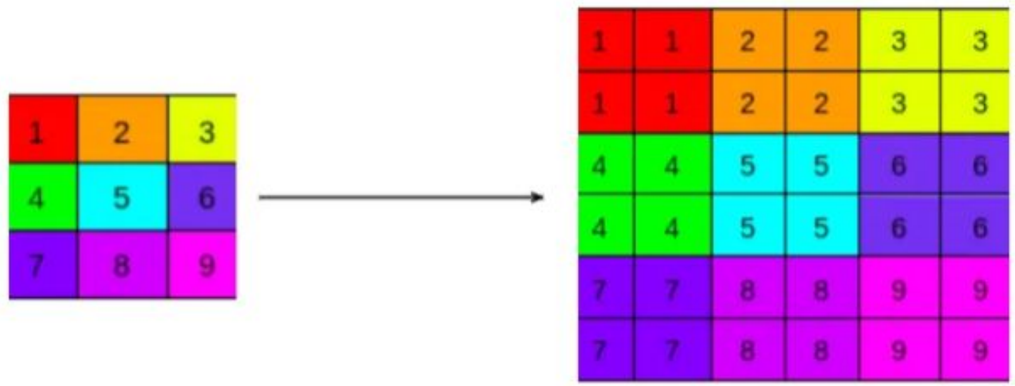
Autoencoders

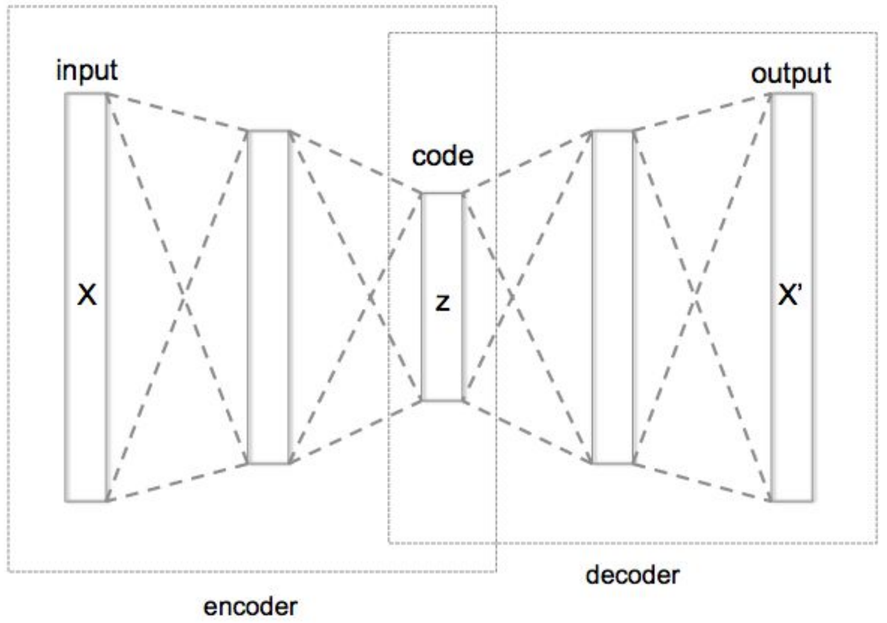
Uso...



Autoencoders

- También se pueden usar capas convolucionales, en este caso estaríamos hablando de un Convolutional Autoencoder.
- La sutileza que aparece al incorporar capas convolucionales es que necesitamos regresar al tamaño original usando capas de up-sampling.





Fuente

Autoencoders

- Aplicaciones varias:
  - Reducción de la dimensionalidad
  - Detección de características
  - Pre-entrenamiento no supervisado
  - Reducción de ruido
  - Detección de anomalias
  - Generación de datos

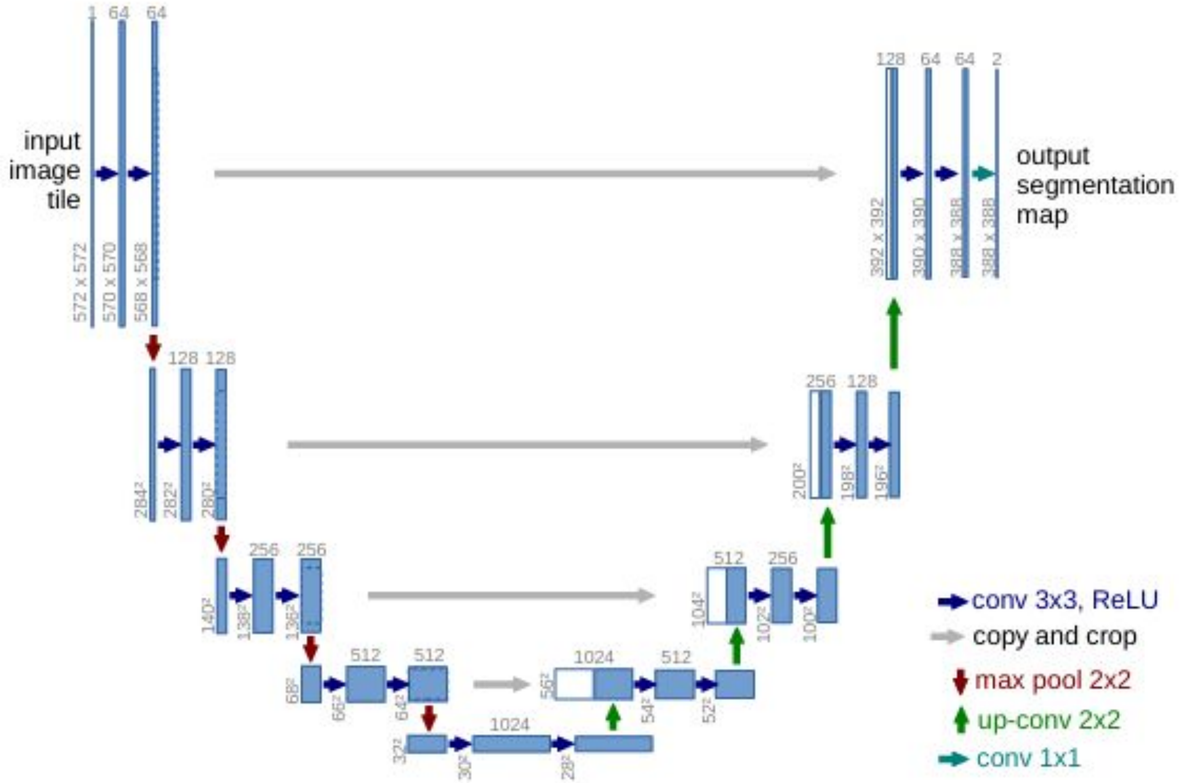
Autoencoders | Deep Learning Animated

Autoencoders

- Hay algunas aplicaciones y variantes en el notebook, lo importante es quedarse con la idea que una vez entrenado tenemos:
  - **Una red que aprendió características de los datos de forma tal que puede codificarlos (o comprimirlos, si quieren) en un espacio de menor dimensión.**
  - **Una red que toma un dato en este espacio latente y lo decodifica para llegar a un elemento en el espacio original que se parece mucho al dato de entrada.**

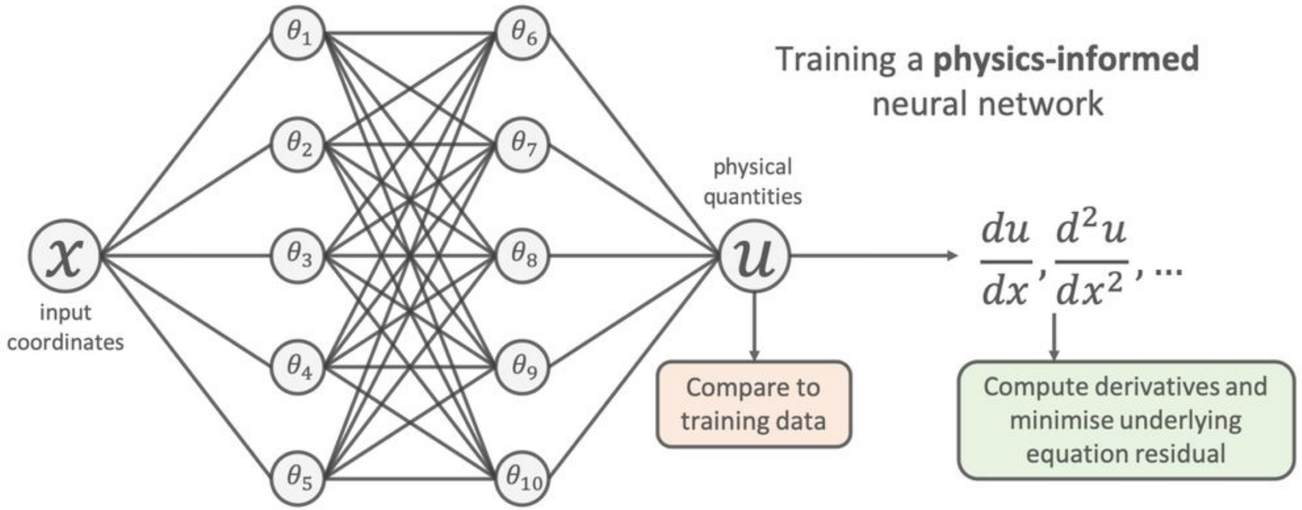
U-Net

- Arquitectura para segmentación semántica.
- Totalmente convolucional:
  - Permite cualquier tamaño de entrada.
  - La salida es del mismo tamaño que la entrada.
- Encoder/Decoder.
- Skip connections entre capas.
  - Se transfieren características de alta resolución del encoder al decoder, ayudando a recuperar detalles espaciales perdidos durante el downsampling.
- Puede ser entrenado con pocas imágenes.
- U-Net: Convolutional Networks for Biomedical Image Segmentation (https://arxiv.org/abs/1505.04597)



Physics-Informed neural networks

- Combina principios físicos y técnicas de aprendizaje automático para crear modelos más precisos, robustos y explicables. Se utiliza cuando hay sistemas físicos bien entendidos pero complejos, donde las leyes de la física ayudan a guiar el aprendizaje.
- Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. https://arxiv.org/abs/1711.10561.
- Physics Informed Machine Learning: High Level Overview of AI and ML in Science and Engineering (link).
- Physics-Informed Neural Networks (PINNs) - The Synergy of Data & Physics in Deep Learning (link).



ICTP

- Seminario ICTP 2024 - Escuela de Herramientas Colaborativas
- Buenas prácticas de desarrollo colaborativo y manejo reproducible de datos científicos.
  - Control de versiones con GIT y gestión de proyectos.
  - Visualización y representación de la información.
  - Python: diseño, testing, integración continua, documentación y extensión con C++.
  - Reproductibilidad y flujos de trabajo colaborativos.
  - Repositorio y cronograma.

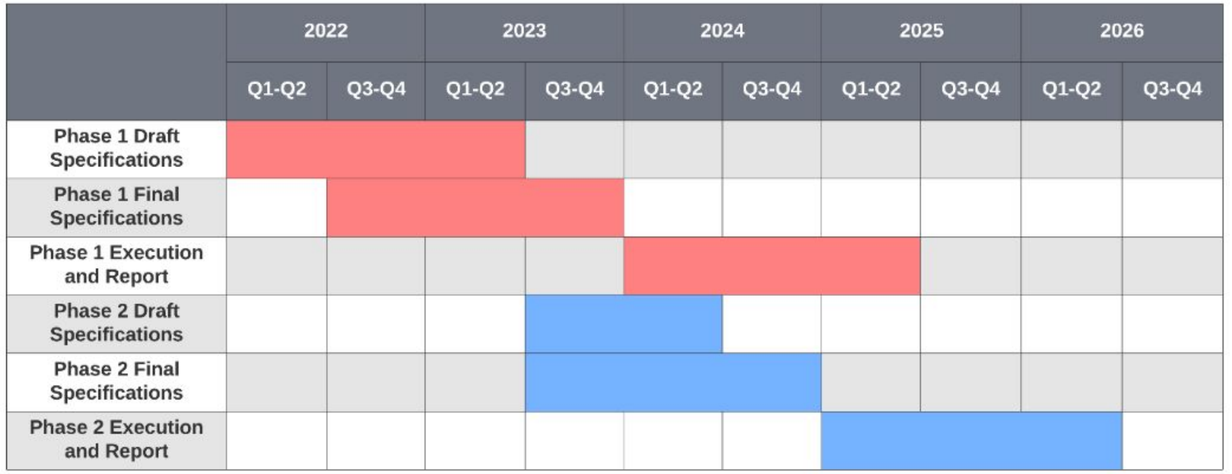
The Abdus Salam International Centre for Theoretical Physics

Oficina Internacional de Energia Atómica (*IAEA*)

- **AI for Atoms** (link).
  - Sitio principal, plataforma de intercambio de conocimientos de la OIEA para la colaboración en aplicaciones de IA en el ámbito nuclear. Eventos, documentos guias y workshops (pocos...). Revisar el Github que tiene datos y código que les puede llegar a servir.
- **Artificial Intelligence for Accelerating Nuclear Applications, Science and Technology** (link)
  - Detalla el estado del arte, los desafíos e identifica oportunidades para acelerar las aplicaciones nucleares, la ciencia y la tecnología usando inteligencia artificial.
- **Deployment of Artificial Intelligence Applications for the Nuclear Power Industry Considerations and Guidance** (link).
  - Consideraciones y guías para los estados miembros para el desarrollo de tecnologías de IA en el ámbito nuclear.

Agencia para la Energía Nuclear (*NEA*)

- **Task Force on Artificial Intelligence and Machine Learning for Scientific Computing in Nuclear Engineering** (link).
  - Diseño de un benchmark para evaluar el desempeño de la IA/ML en la simulación multifísica de sistemas de reactores nucleares.
  - El objetivo es mejorar la capacidad predictiva de la IA/ML mediante una verificación y validación rigurosa, centrada en precisión, robustez y transparencia, para fomentar la aceptación de estas tecnologías por parte de los reguladores y las partes interesadas en el sector nuclear.



The end