Argentina programa 4.0

Universidad Nacional de San Martín

Módulo 3
Aprendizaje Automático

Argentina programa 4.0

Universidad Nacional de San Martín

Módulo 3
Aprendizaje Automático

Semana 8 AlexNet / Interpretabilidad / *Transfer Learning*

Contenidos del módulo

ML Clásico

- Árboles de Decisión
- Métodos de Ensemble
  - Bagging / Pasting —> Random Forests
  - Boosting
- Support Vector Machines

Deep Learning

- Redes Neuronales
- Redes Neuronales Convolucionales
- Auto-Encoders / Auto-Encoders Variacionales
- Redes Neuronales Recurrentes (LSTM, otras)
- Extras:
  - Generative Adversarial Networks (GAN)
  - Reinforcement Learning

Contenidos del módulo

ML Clásico

- Árboles de Decisión
- Métodos de Ensemble
  - Bagging / Pasting —> Random Forests
  - Boosting
- Support Vector Machines

Deep Learning

- Redes Neuronales
- Redes Neuronales Convolucionales
- Auto-Encoders / Auto-Encoders Variacionales
- Redes Neuronales Recurrentes (LSTM, otras)
- Extras:
  - Generative Adversarial Networks (GAN)
  - Reinforcement Learning

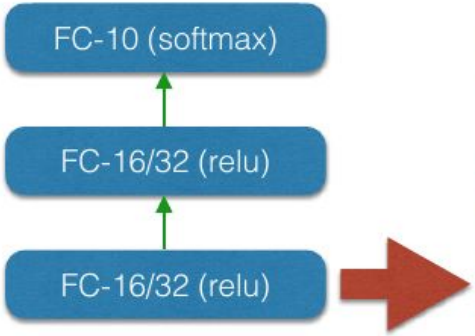
Interpretabilidad

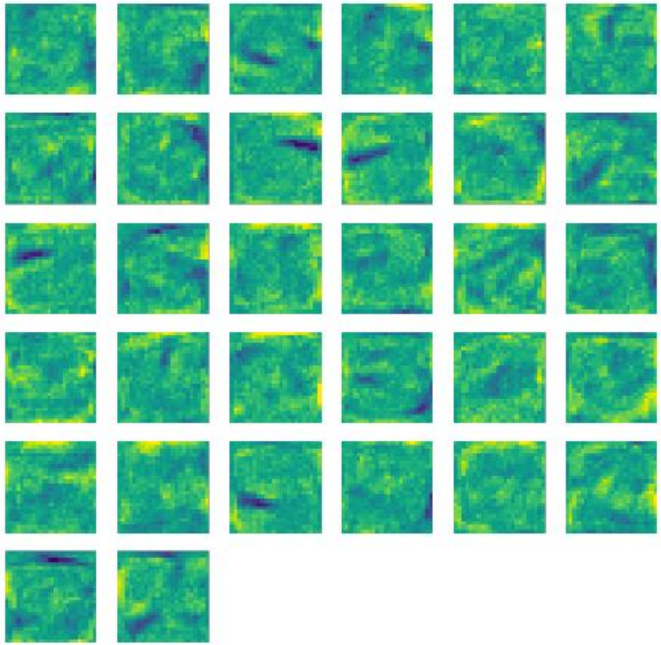
- Estuvimos una semana viendo redes neuronales convolucionales y nos creyeron (quizás) cosas como:
  - “Las primeras capas de la red detectan patrones simples”
  - “Los filtros se combinan y las capas más profundas detectan patrones más complejos”
  - “El lunes les vamos a mostrar pruebas”
  - etc.
- Entrenamos algunas redes y funcionaron, a pesar de que en realidad no sabemos muy bien qué está pasando dentro
- Vamos a mostrar un trabajo muy conocido que justamente ataca este problema, como interpretar lo que está pasando dentro de una CNN

¿Por qué?

- Porque es interesante
- Nos puede ayudar a mejorar nuestras arquitecturas
- Nos va a servir para reutilizar redes que ya sabemos que funcionan bien y adaptarlas a una nueva tarea

Totalmente conectada

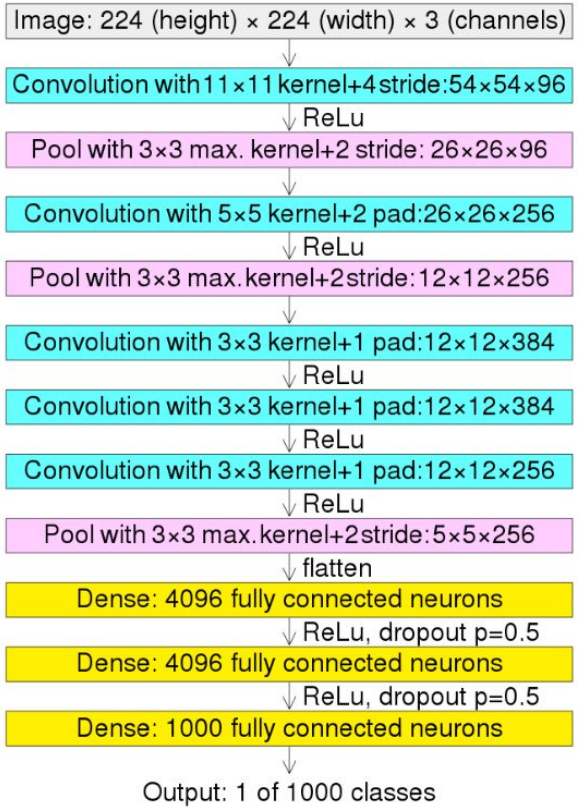




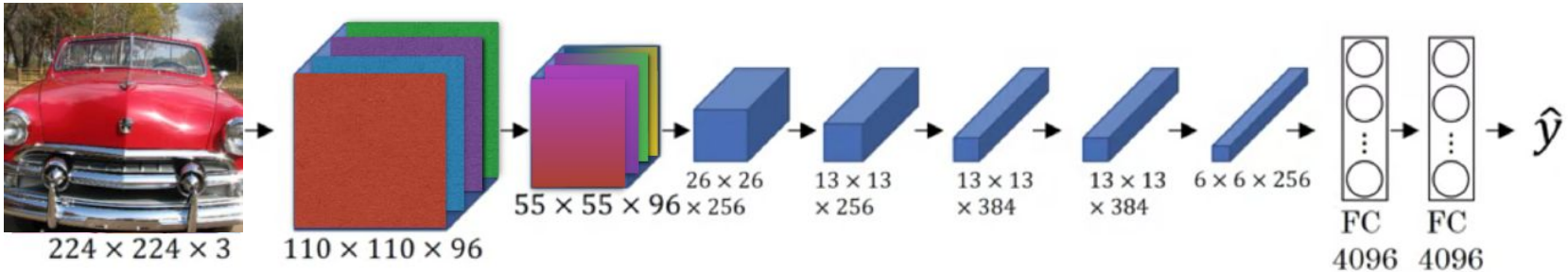
AlexNet

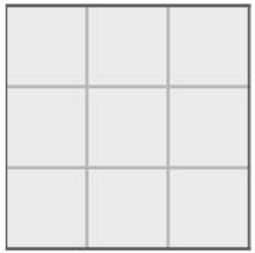
- Entrenada con más de un millón de imágenes
- Clasifica 1000 categorías de objetos (como teclado, taza de café, lápiz y muchos animales)
- Compitió en *ImageNet Large Scale Visual Recognition Challenge* en 2012 y alcanzo un error de 15.3%, más de 10.8 puntos menos que el segundo
- Una de las primeras 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

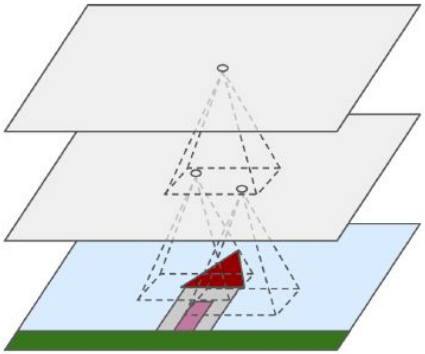


AlexNet





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



AlexNet

Neurona
(filtro 1)

AlexNet

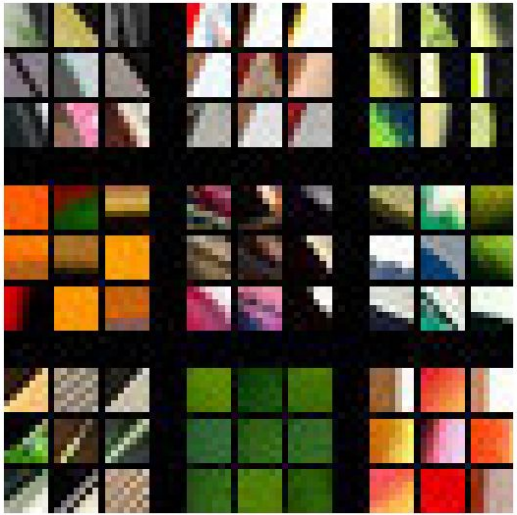
Neurona (filtro 1)

Neurona (filtro 2)

AlexNet

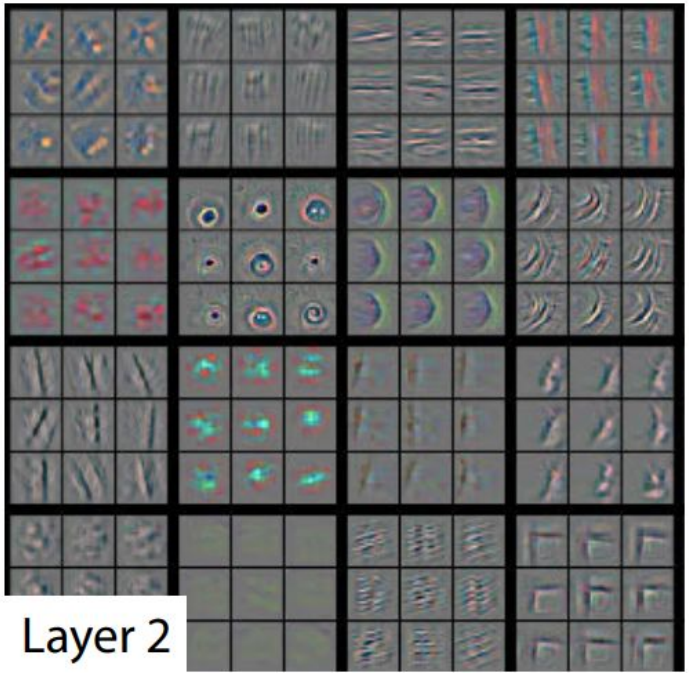
Neurona (filtro 1) Neurona (filtro 2) Neurona (filtro 3)

...

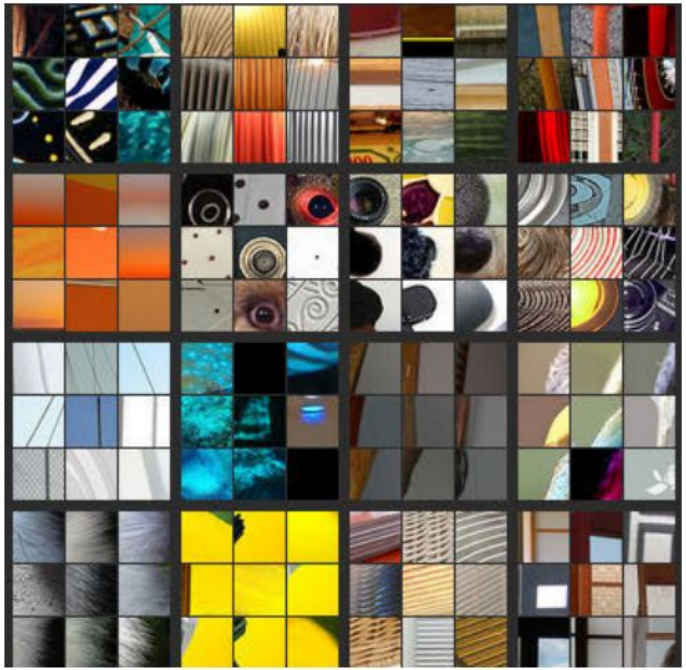


Neurona (filtro 9)

AlexNet

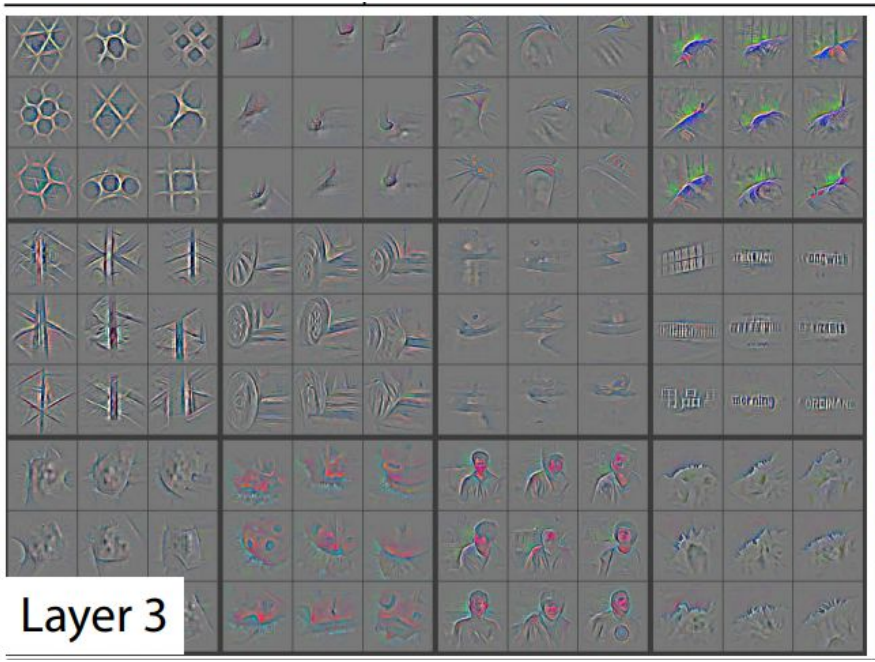


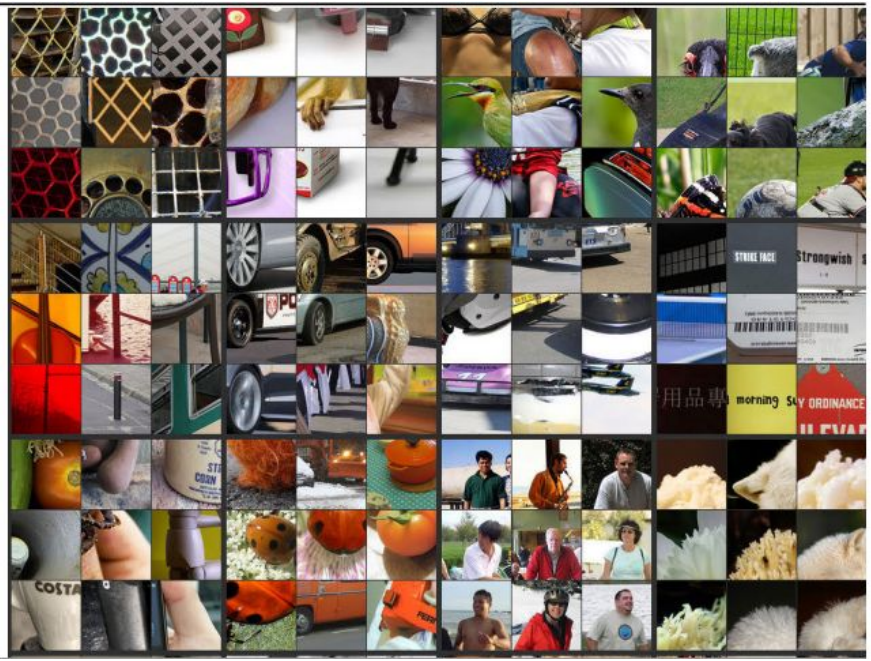
Patrones que maximizan la activación



Ejemplos de imágenes que maximizan la activación

AlexNet



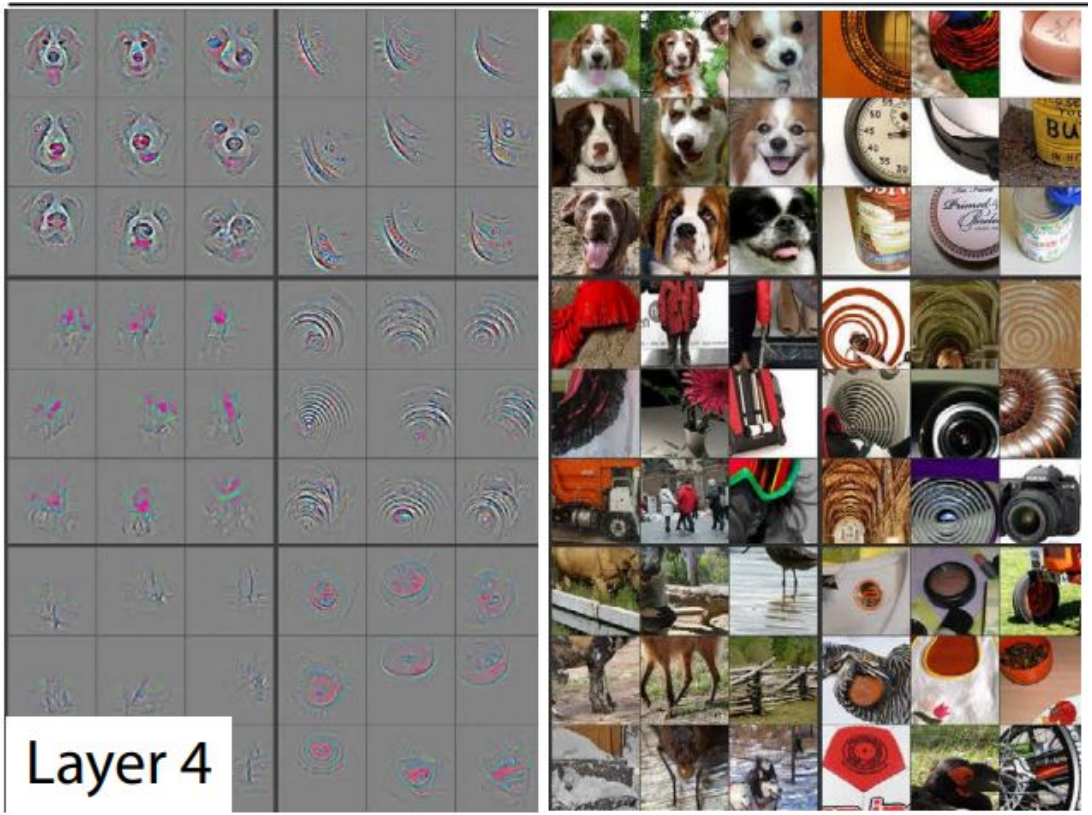


Patrones que maximizan la activación

Ejemplos de imágenes que maximizan la activación

AlexNet

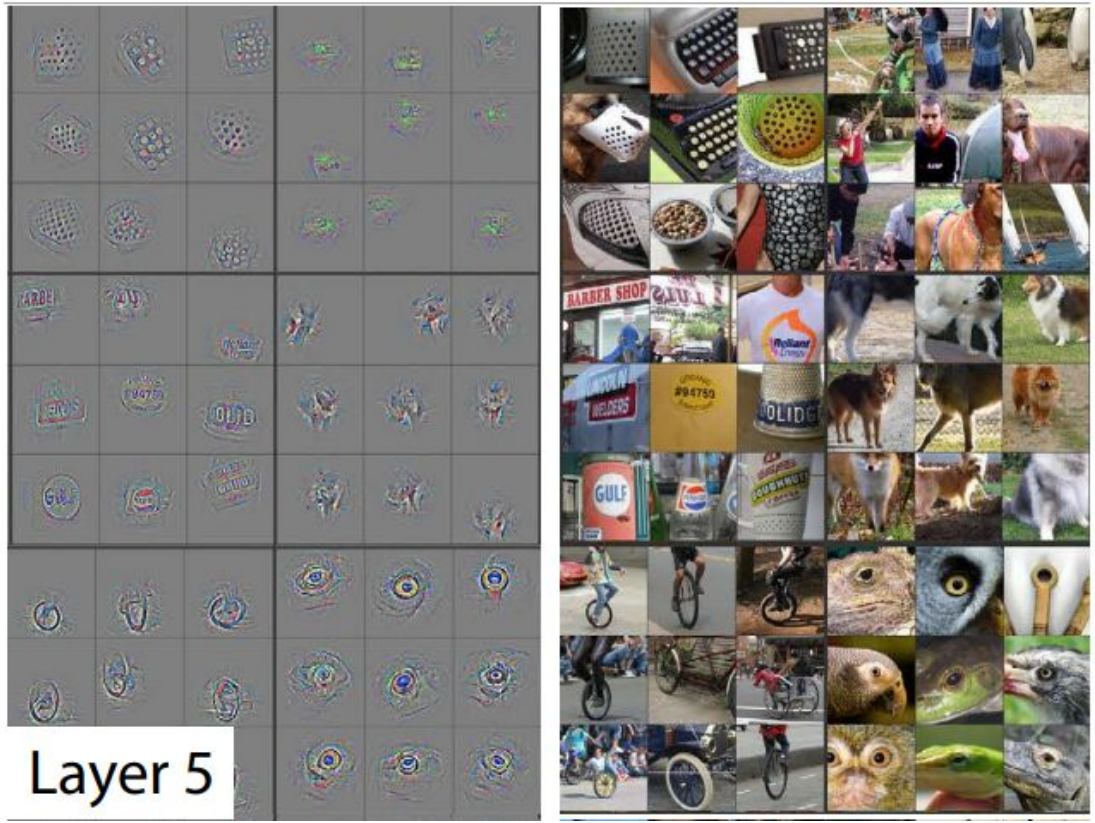
Patrones que maximizan la activación



Ejemplos de imágenes que maximizan la activación

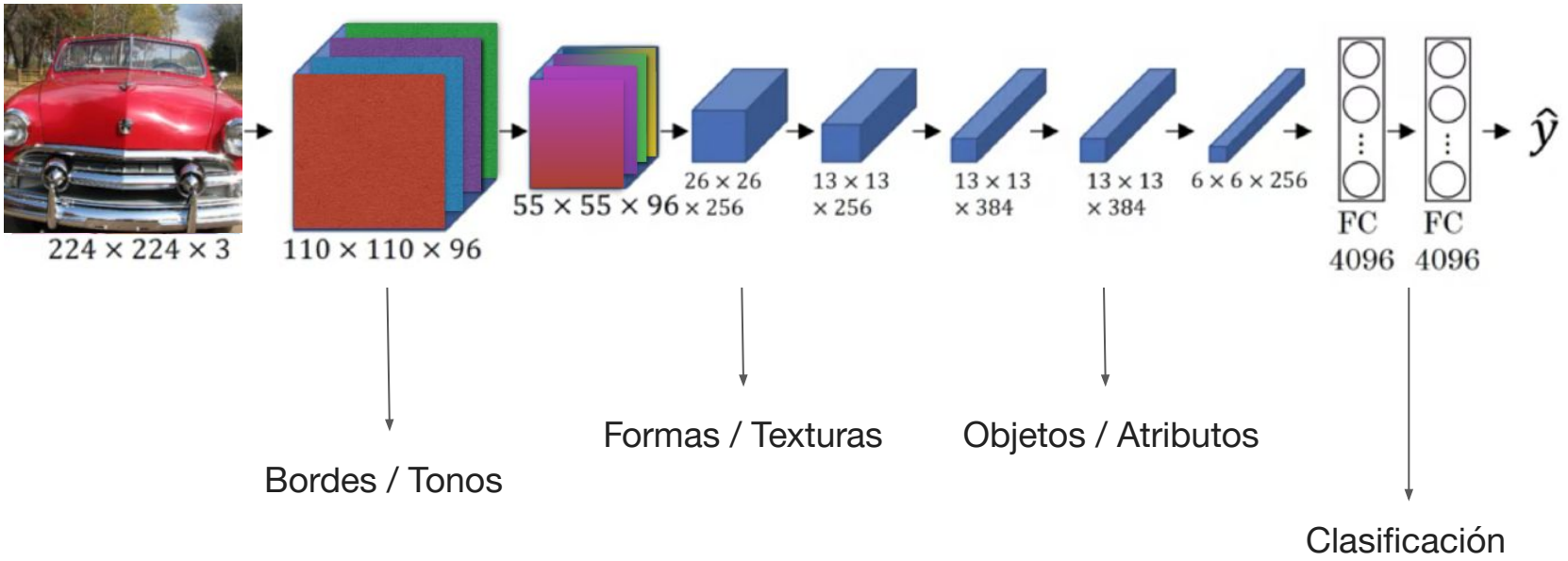
AlexNet

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
$\hat{y}$
Bordes / Tonos
Formas / Texturas
Objetos / Atributos
Clasificación

Transfer Learning

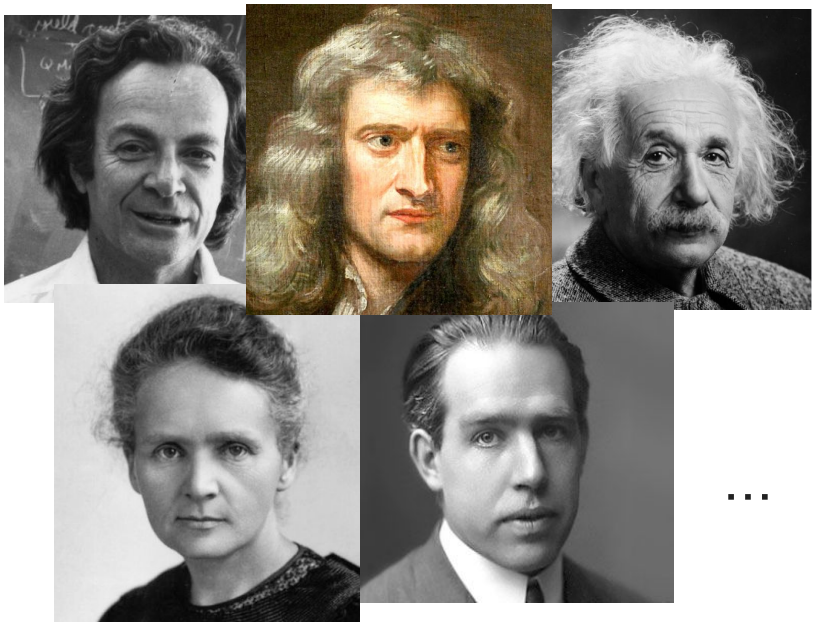
- Las primeras capas de una red profunda funcionan como detectores de características simples que son generales a todas las imágenes (ahora lo vimos, basta de fe)
- Las más profundas ya son de objetos y atributos particulares de las imágenes con las que fue entrenado
- Veamos como podemos aprovechar todo esto para hacer un clasificador de De Florianes…
- Primero, nos armamos un dataset de imágenes para nuestro problema en particular
- Segundo, nos conseguimos una AlexNet entrenada

Imágen del Géron

DeFloSet

...

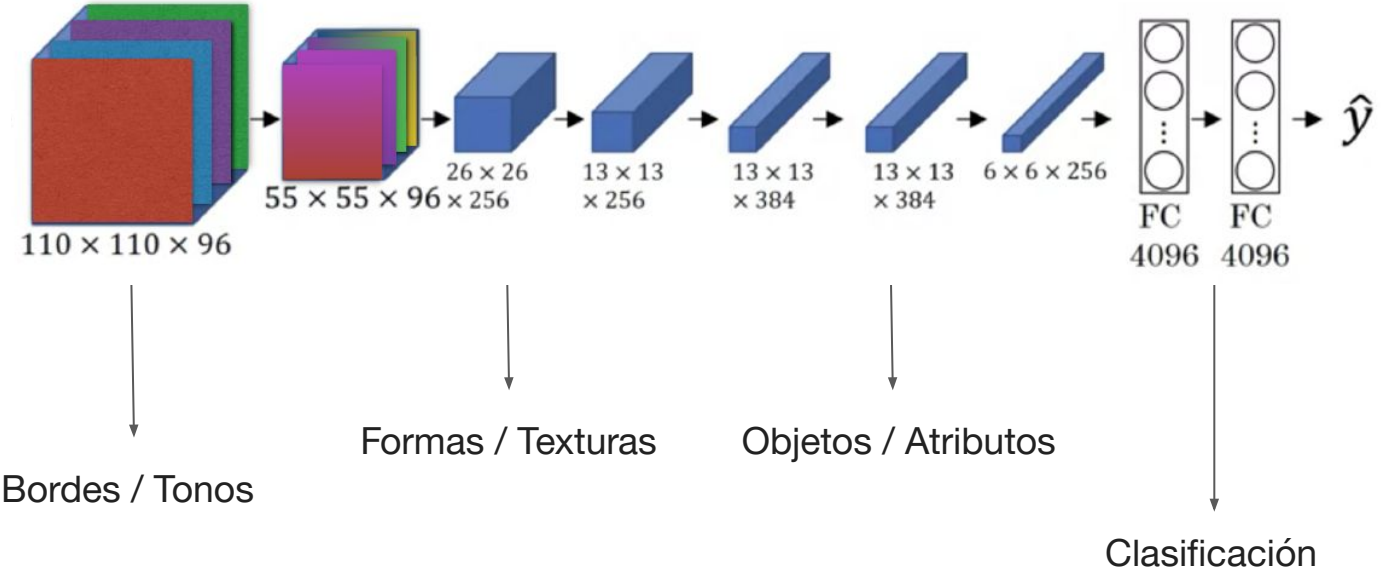
Label = 1



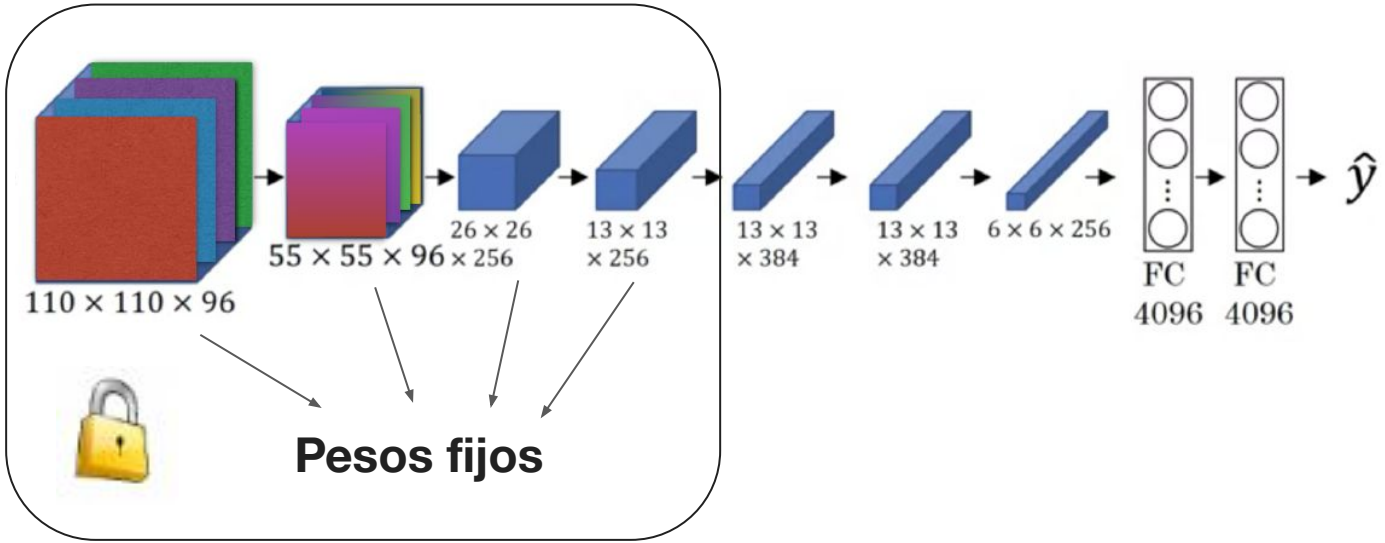
...

Label = 0

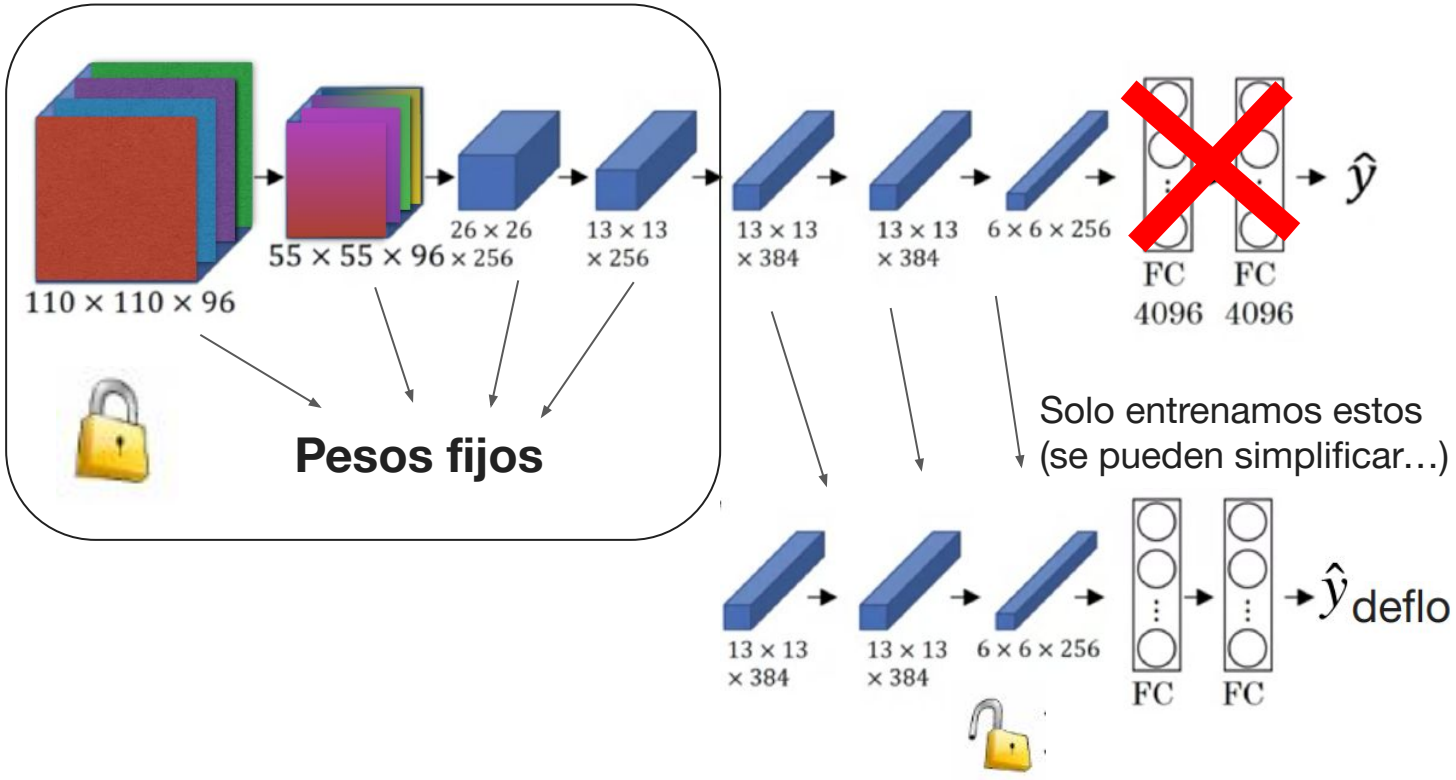
Modelo base


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
$\hat{y}$
Bordes / Tonos
Formas / Texturas
Objetos / Atributos
Clasificación

Fijamos pesos

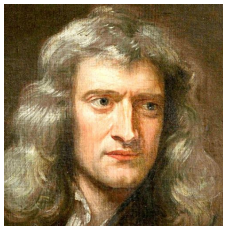


Elegimos que capas entrenar

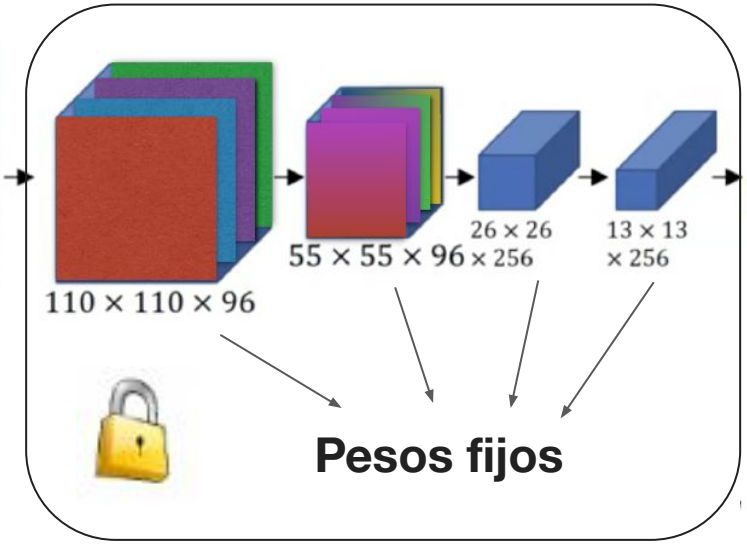


Entrenamos...

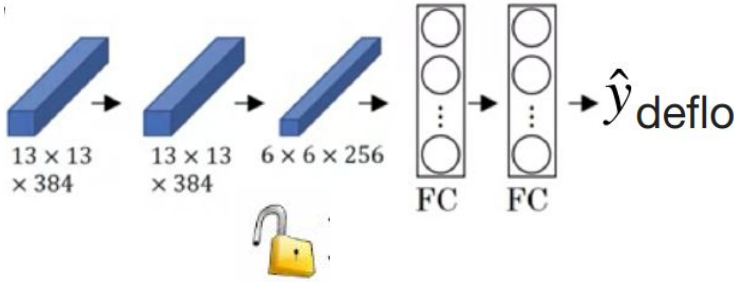
224 × 224 × 3



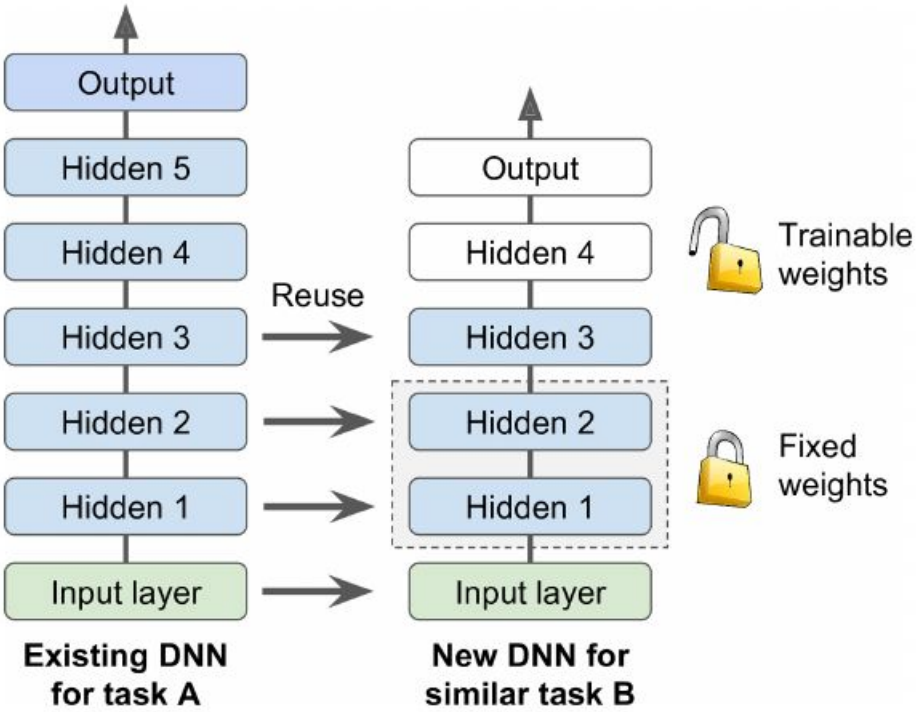
...



Solo entrenamos estos
(se pueden simplificar…)



Transfer Learning



Imágen del Géron

- 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.

Notebook

Vamos al notebook!

`Notebook_Semana_8_CNN_Transfer_Learning.ipynb`