

Visualizando lo que aprenden los convnets



Las representaciones aprendidas
por los convnets corresponden a
conceptos visuales

Técnicas de visualización útiles

- Visualizar salidas de convnets intermedios:
 1. ¿Cómo convnets sucesivos modifican sus entradas?
 2. ¿Permiten ver los filtros convnets individuales?
- Visualizar filtros convnets
 1. Entender patrones visuales a los que responde cada filtro.
- Visualizar **mapas de calor** (*heatmaps*) de activación de clase en una imagen.
 1. Identificar qué partes de una imagen son cruciales para asignarle una clase dada.

Visualizar activaciones intermedias

- Utilizamos *convnet* entrenado desde cero.
- Queremos visualizar mapas 3D: ancho, alto, profundidad (canales)
- Cada canal codifica características relativamente independientes:
 - Dibujaremos el contenido de cada canal como image 2D.

Convolución:

- Extrae cuadros de los FM de entrada
- Transformaciones y FM de salida con tamaño FM arbitrarios
 - Primera capa: Profundidad son Canales de colores
 - Capas sucesivas: Profundidad son *filtros* que codifican aspectos de los datos⁴

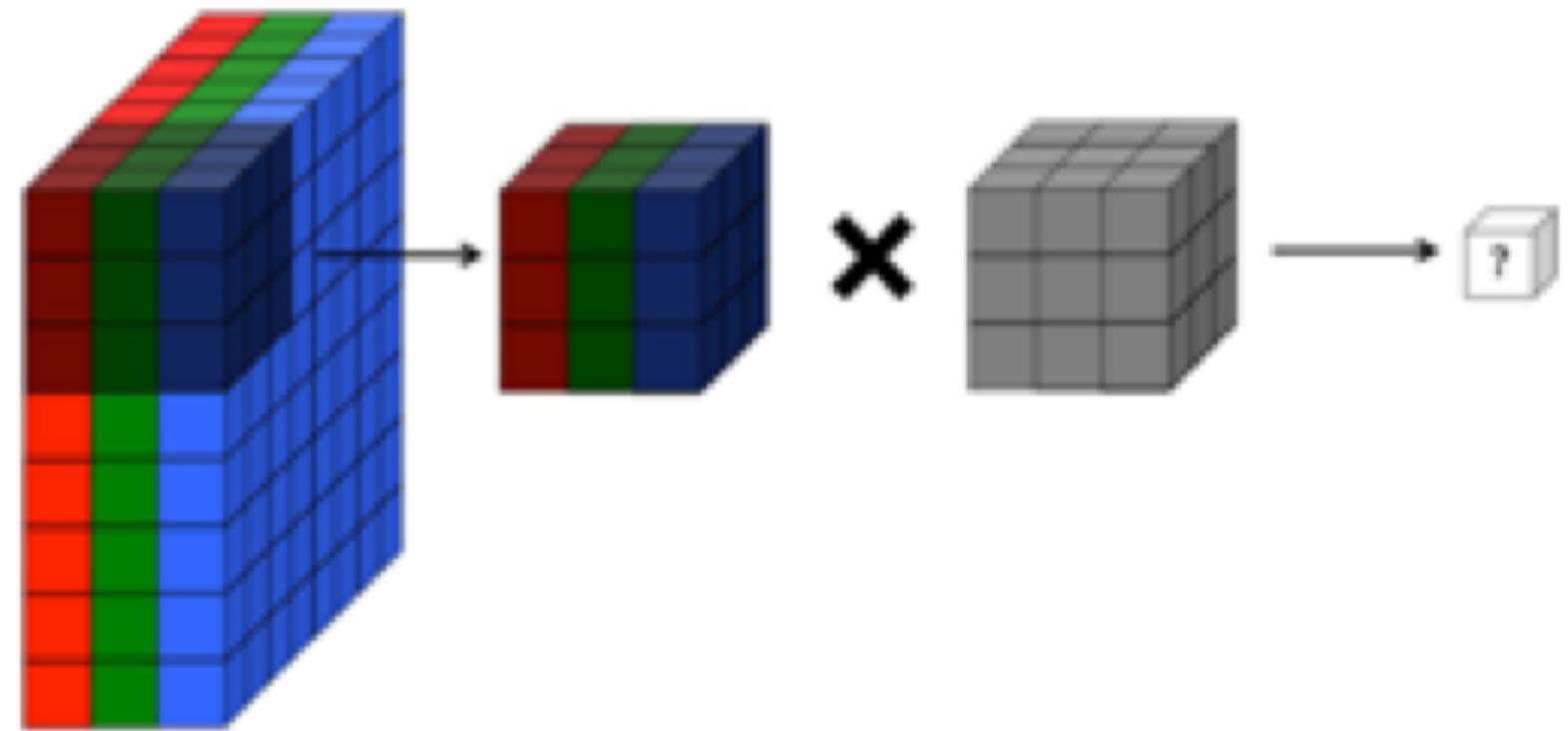
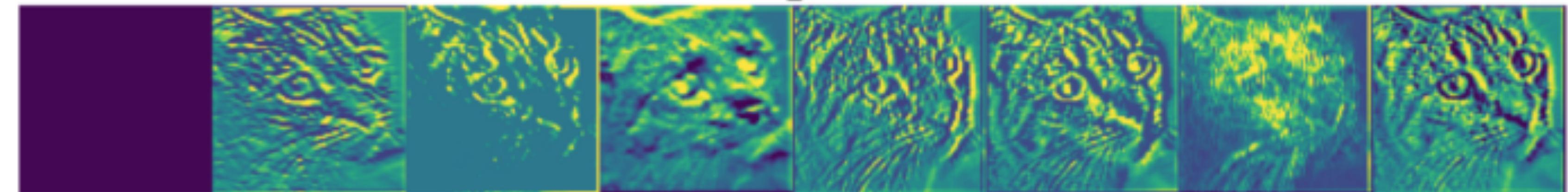


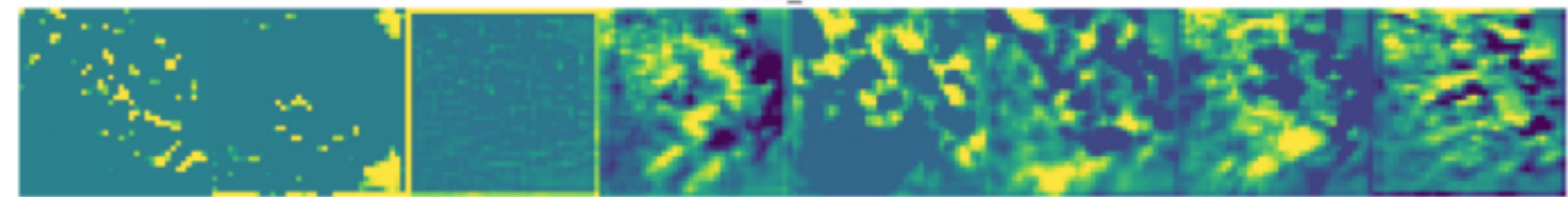
Figure 5-8. Representing a full-color RGB image as a volume and applying a volumetric convolutional filter

⁴ StackExchange, *In a CNN, does each new filter have different weights for each input channel, or are the same weights of each filter used across input channels?*

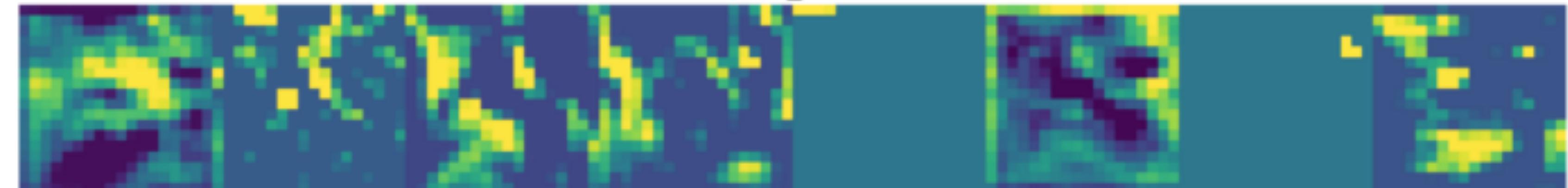
block2_conv1



block3_conv1



block4_conv1



Pasos iniciales

- Cargar el modelo de scratch.
- Obtener una imagen no utilizada de gato para entrenar modelo.
- Preprocesar la imagen.
 - Tamaño 150 x 150-
 - Convertir a tensor (4D).
 - Estandarizar por /=255.
- Mostrar la imagen.



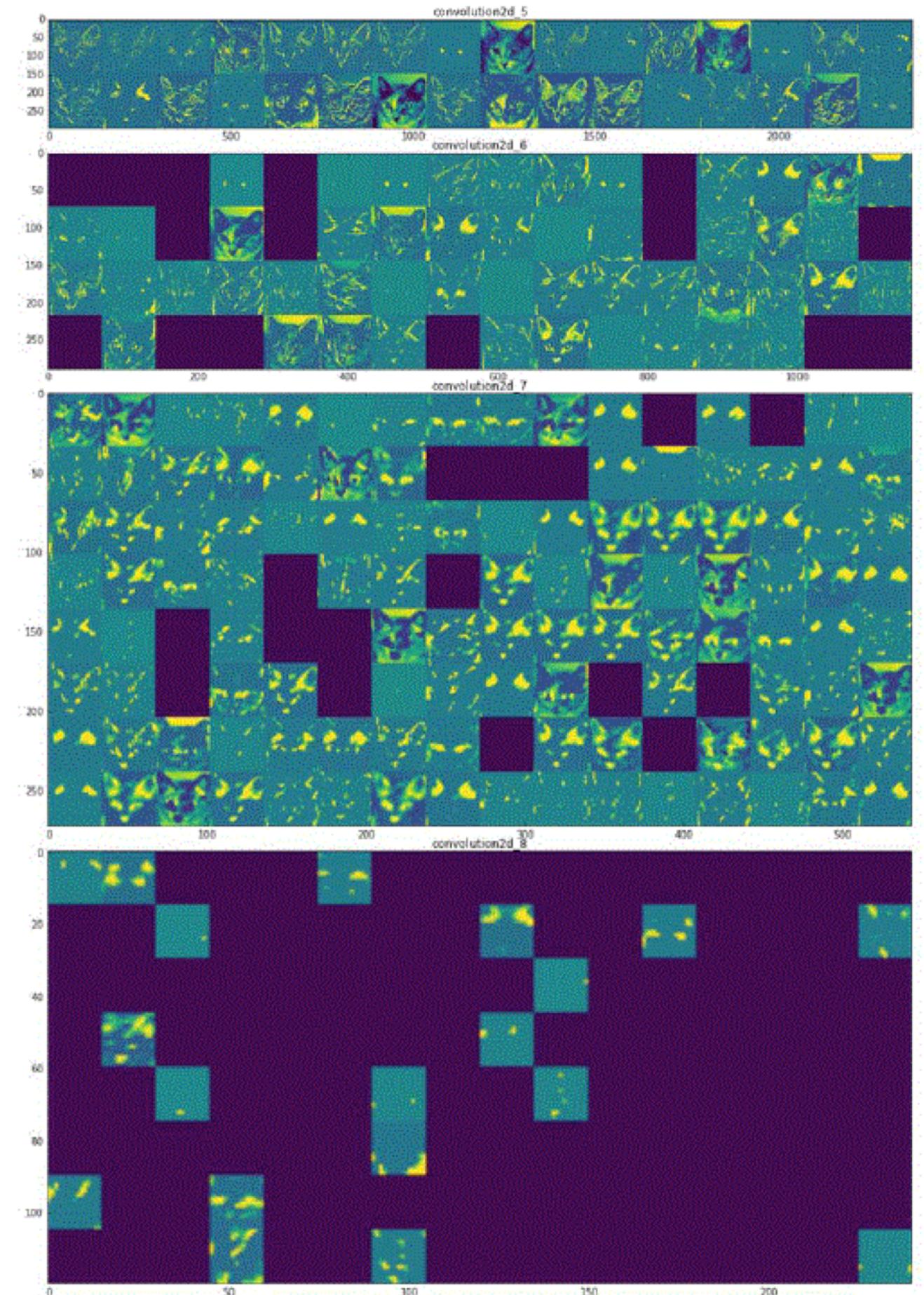
Extraer Feature Maps

- Usar la clase `model` de Keras
 - Más sobre la clase `model` al final del curso.
 - Modelo con dos argumentos:
 1. Tensor o lista de tensores de entrada.
 2. Tensor o lista de tensores de salida.
- La clase resultante es un modelo Keras.
 - En vez de un modelo secuencial.
 - La clase `model` permite multiples outputs.
 - El modelo tiene 1-input y 8-outputs.
 - Devuelve valor de activación de las capas del modelo secuencial original.

- Primera capa convolutiva (imagen gato)
 - *Feature map* $148 \times 148 \times 32$
- Visualizar canal 4
 - Codifica un detector de ejes diagonales
- Visualizar canal 7
 - Codifica *manchas verdes brillantes* (realza ojos)
- Visualizar todas las activaciones de la red
 - Extraer cada canal en cada uno de los 8 mapas de activación.

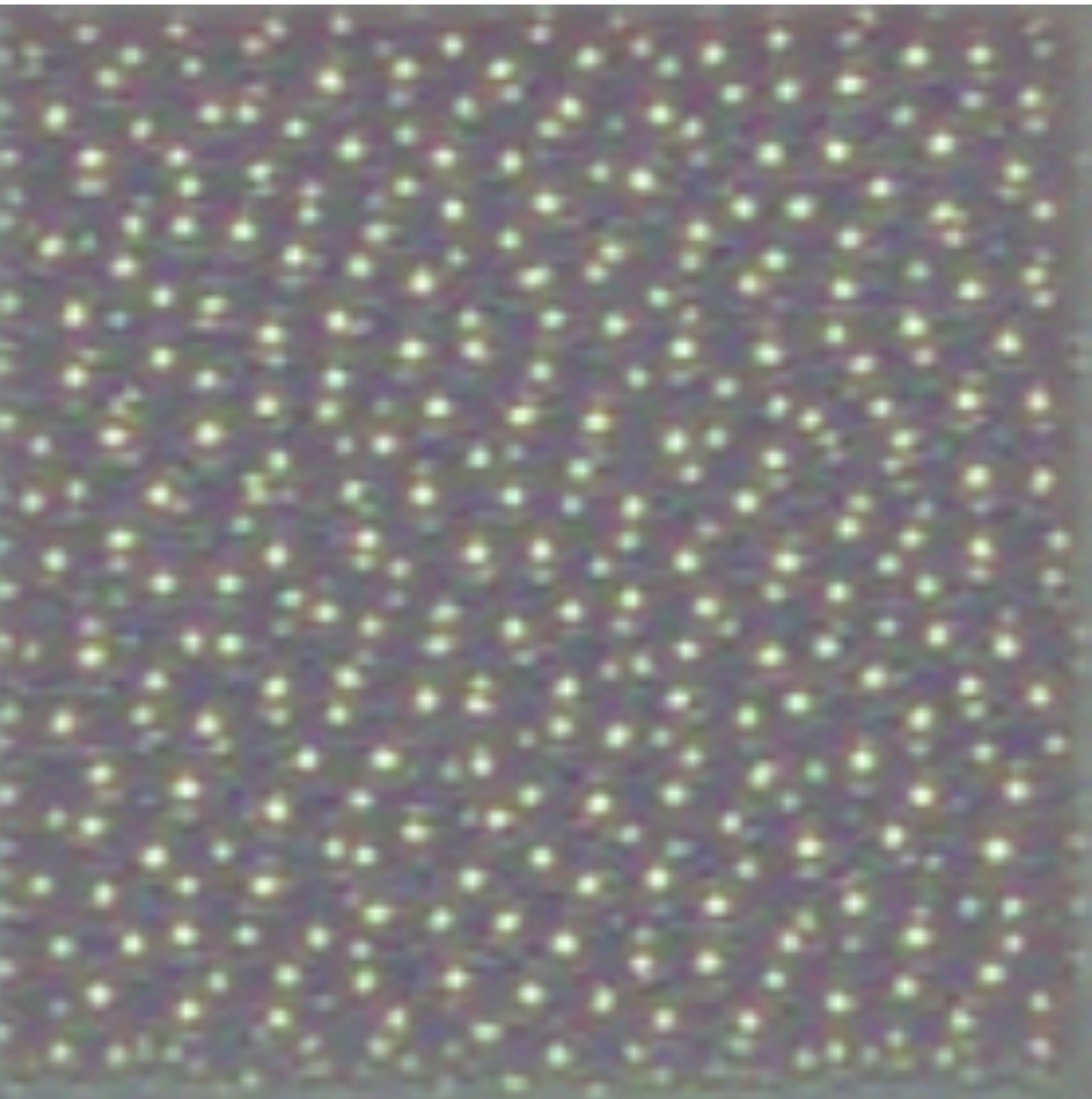
Nótese

- Capa 1: colección de detectores debordes.
 - Activación retiene casi toda la información de la imagen original
- Al profundizar en la red:
 - Activaciones más abstractas.
 - Oreja, ojo, nariz de gato...
 - Outputs llevan menos información sobre el contenido visual, pero más información relacionada con la clase a la que se asigna la imagen.
- Activaciones disminuyen con profundidad de la capa.
 - Más y más filtros *en blanco*.
 - La característica codificada en el filtro **no se encuentra** en la imagen.

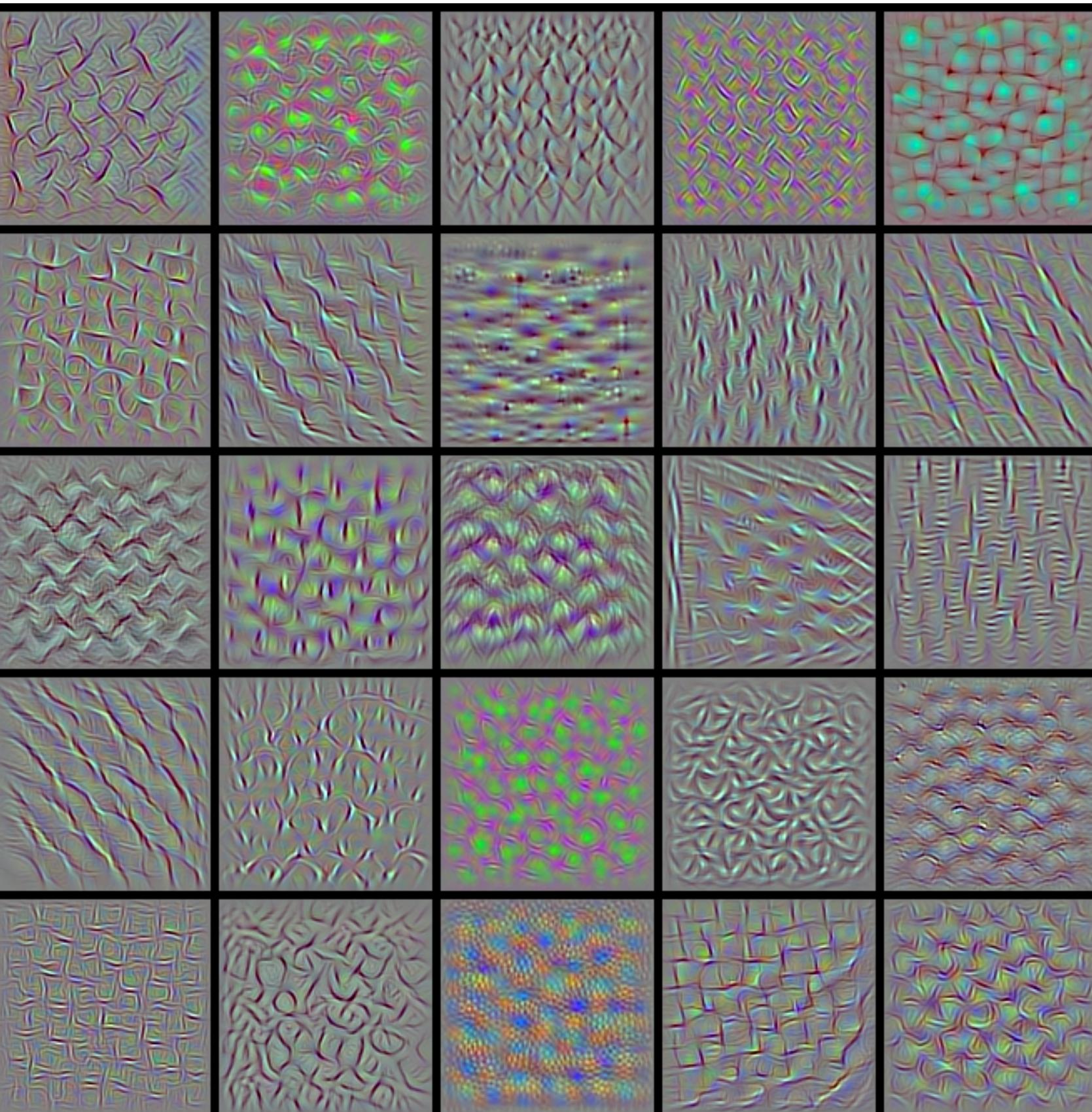


Visualización filtros convnet

- Mostrar el patrón visual al que responde cada filtro.
- Aplicar **gradiente ascendente** en el espacio de entrada.
 - Maximizar respuesta de un filtro, empezando con una imagen de entrada en blanco.
- Construimos una función de pérdida que maximice el valor de un filtro determinado en una capa dada.
- Aplicamos el **gradiente descendiente estocástico** para ajustar los valores de la imagen que maximizan el valor de activación.
 - Ejemplo: la función de pérdida para la activación del filtro 0 (cero) en la capa block3_conv1 de la red VGG16 preeentrenada en ImageNet.



- Definir una función que haga todo este trabajo:
 - `generate_pattern`
 - *Input:* Nombre de una capa e índice de filtro.
 - *Output:* Imagen del tensor que representa el patrón que maximiza la respuesta de un filtro determinado.
 - Aplicar la función sobre una selección de filtros y capas.
 - Cada capa *aprende* una colección de filtros que combinados reproducen el input.



Visualización de mapas de calor de activación de clase

- Entender qué partes de un imagen dada determinan la clasificación final.
- Útil para depurar el proceso de decisión.
 - Casos de errores de clasificación.
 - Útil para localizar objetos específicos en una imagen.

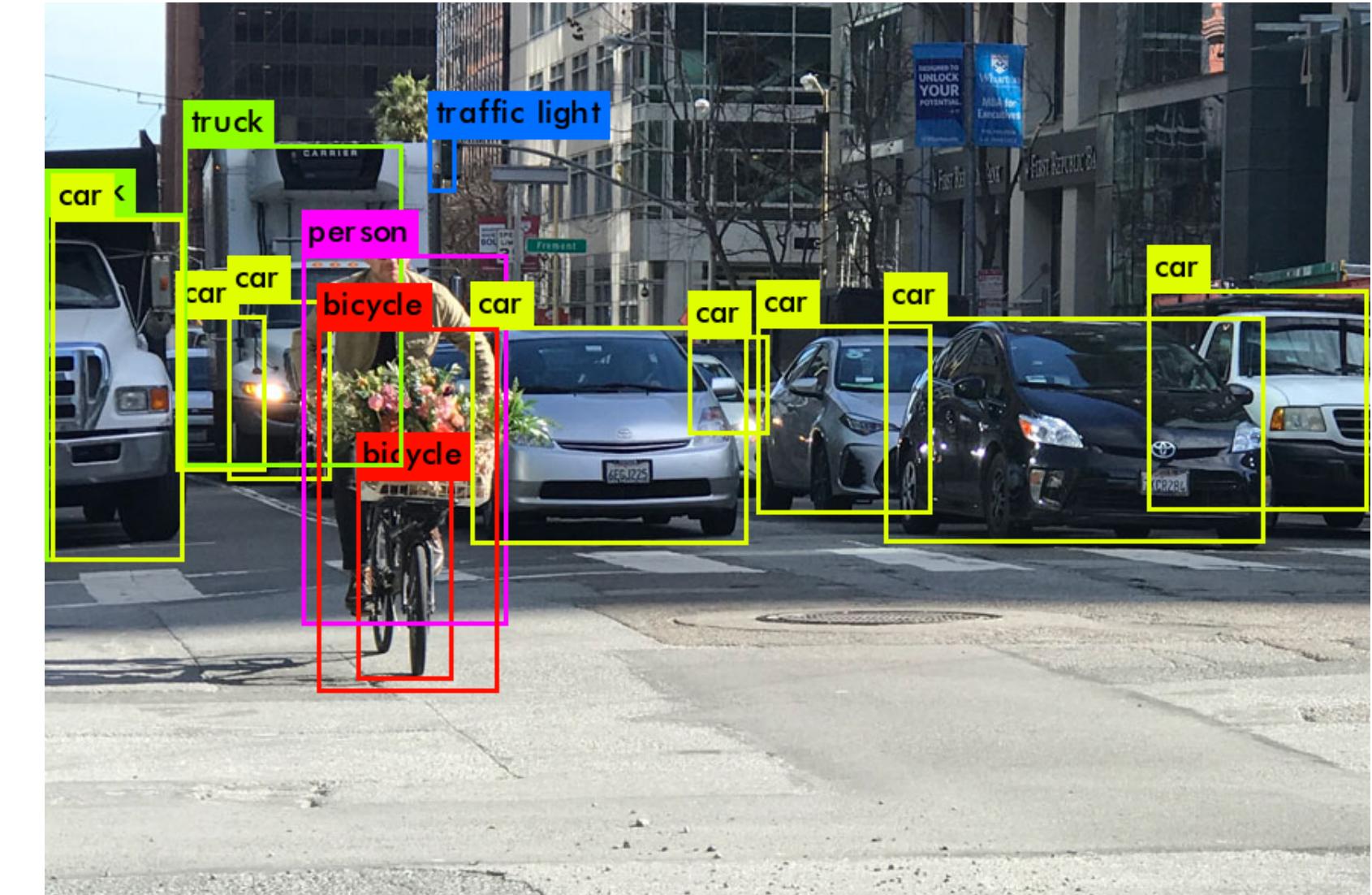
Estas técnicas se llaman en general **Class Activation Map** (CAM).

Class Activation Map

- Dada una imagen:
 - Se toma el *Feature Map* (FM) de salida de una capa convolutiva.
 - Se pesa cada capa del FM por el gradiente de la clase respecto al canal.
- Ejemplo:
 - Cambiar tamaño de imagen a 224 x 224.
 - Convertir en tensor Numpy 'float32'.
 - Aplicar preprocessamiento.



You Only Look Once (YOLO)²



² Jonathan Hui, 2018, *Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3*



Resumen

- Convnets son actualmente la mejor herramienta para problemas de clasificación de imágenes.
- Convnets aprenden una jerarquía de patrones modulares y conceptos para representar el mundo visual.
- Convnets son «fáciles» de visualizar.
- Hemos aprendido a entrenar convnets a) desde cero, b) con *data augmentation* para evitar sobreajuste, y c) preentrenados.

Lecturas adicionales



- Areeb Gani, 2020, Analytics Vidhya: [Visualizing Activation Heatmaps using TensorFlow](#)

Utiliza la función
`tf.GradientTape`

- Veröffentlicht, 2020, Linux-Blog: [A simple CNN for the MNIST dataset – IV – Visualizing the activation output of convolutional layers and maps](#)



Palabras y
caracteres:

- *One-hot encoding*
- *Word embeddings*

El próximo día

KEEP
CALM
AND
SEE YOU
NEXT CLASS