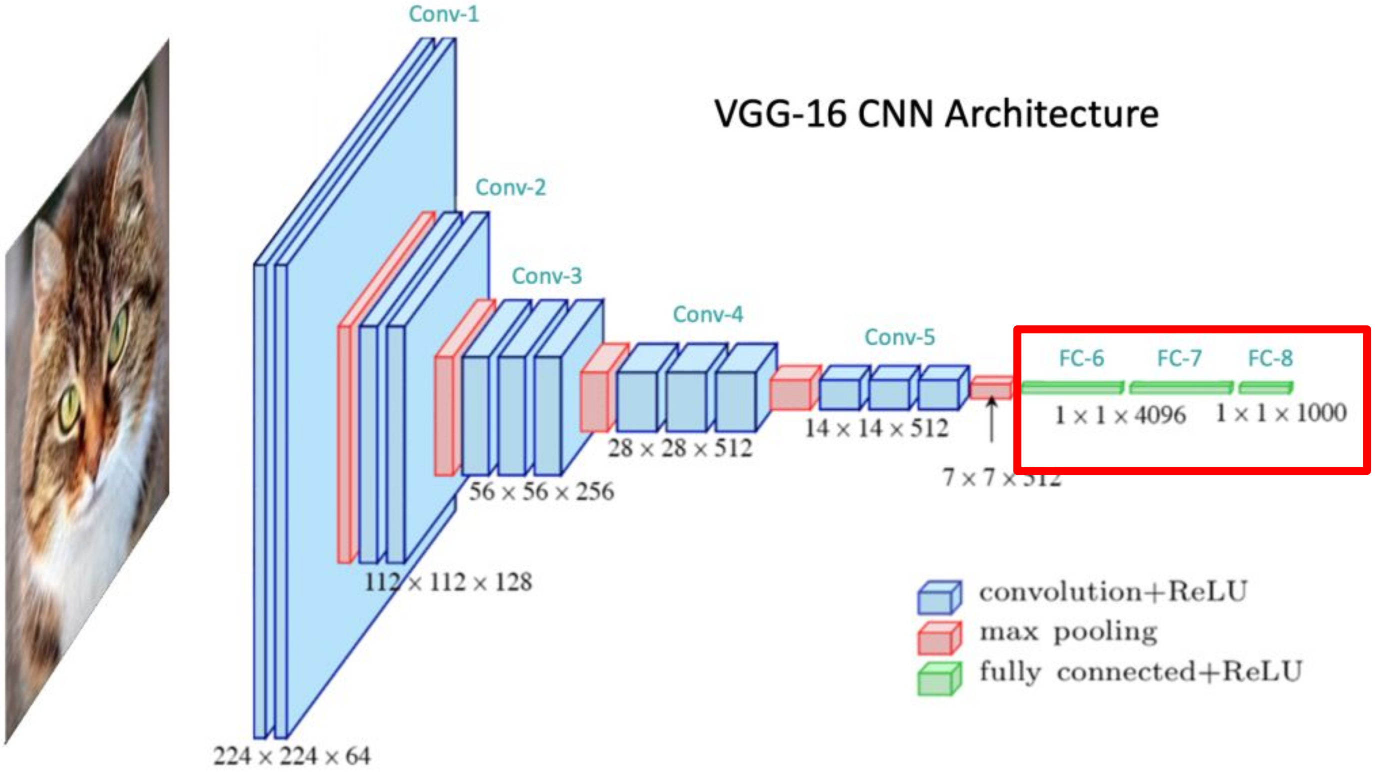
# Convolutional Neural Networks II

Dr. Chelsea Parlett-Pelleriti

# Convolutional Neural Networks



# Data Augmentation

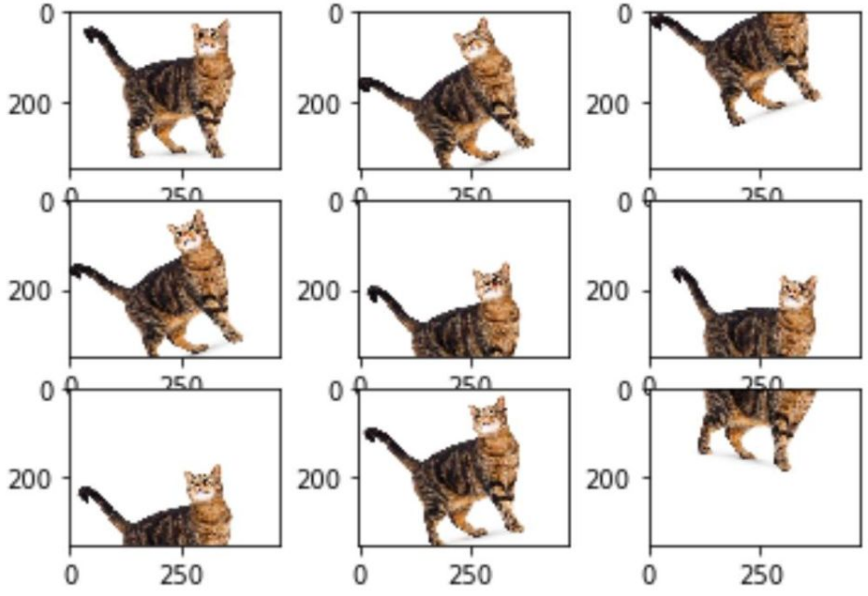



Image from: https://analyticsindiamag.com/image-data-augmentation-impacts-performance-of-image-classification-with-codes/

# Data Augmentation

- Crop
- Flip
- Translation
- Rotation
- Zoom
- Contrast
- Brightness

```
plt.figure(figsize=(10, 10))
for images, _ in train_dataset.take(1):
    for i in range(9):
        augmented_images = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented_images[0].numpy().astype("uint8"))
        plt.axis("off")
```

We can use take(N) to only sample *N* batches from the dataset. This is equivalent to inserting a break in the loop after the *N*th batch.

Apply the augmentation stage to the batch of images.

Display the first image in the output batch. For each of the nine iterations, this is a different augmentation of the same image.

**Figure 8.10** Generating variations of a very good boy via random data augmentation

Image from: Deep Learning with Python (Chollet)

# Residual Connections

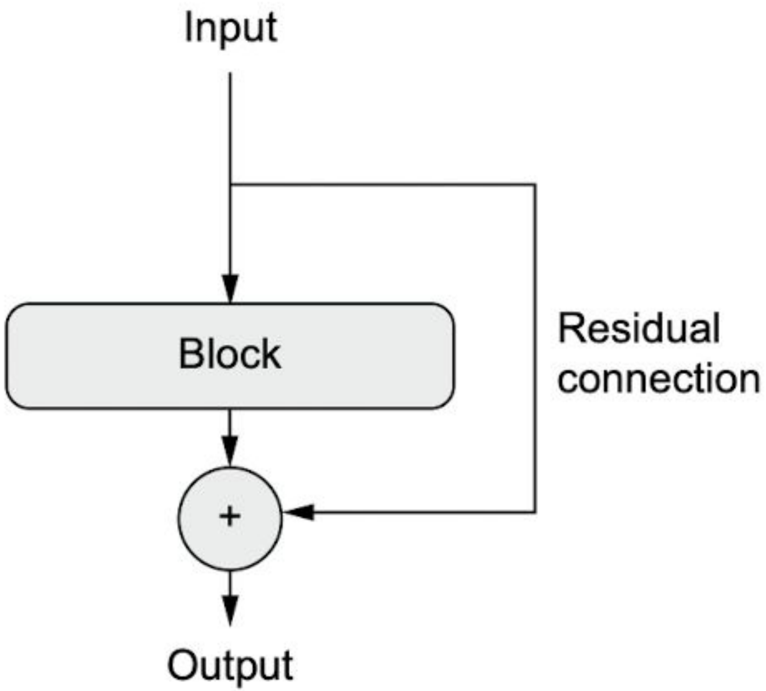



Image from: Deep Learning with Python (Chollet)

# Batch Normalization

**Input:** Values of $x$ over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: $\gamma$, $\beta$

**Output:** $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

**Algorithm 1:** Batch Normalizing Transform, applied to activation $x$ over a mini-batch.

# Depthwise Separable Convolutions

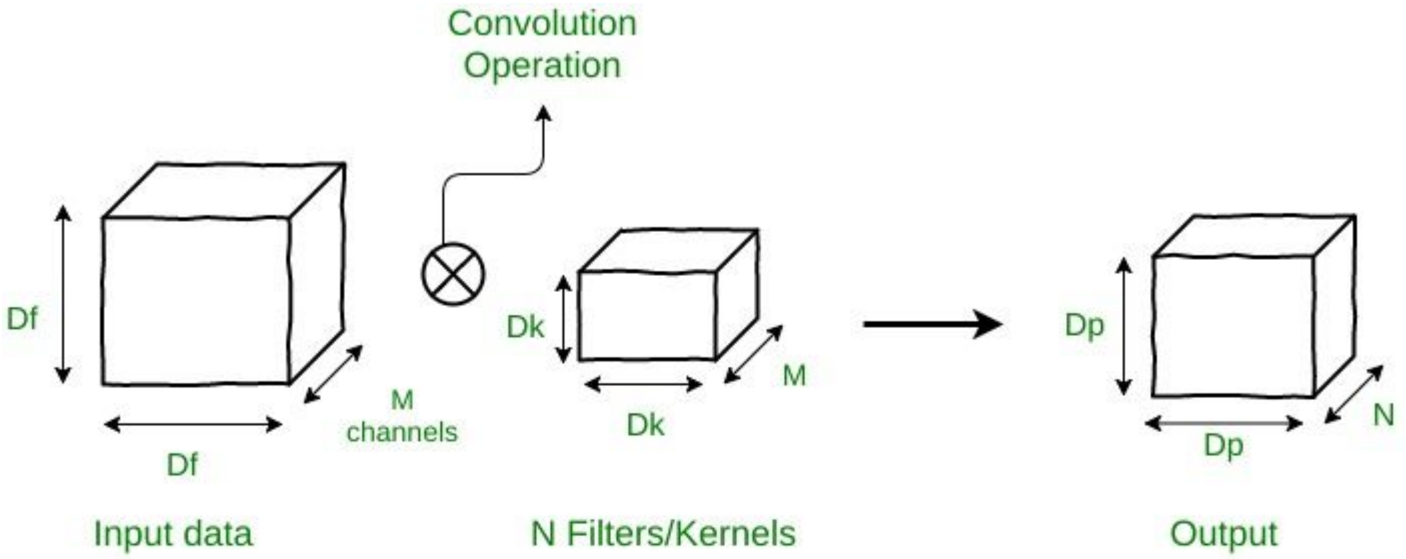



Image from: https://www.geeksforgeeks.org/depth-wise-separable-convolutional-neural-networks/

# Depthwise Separable Convolutions

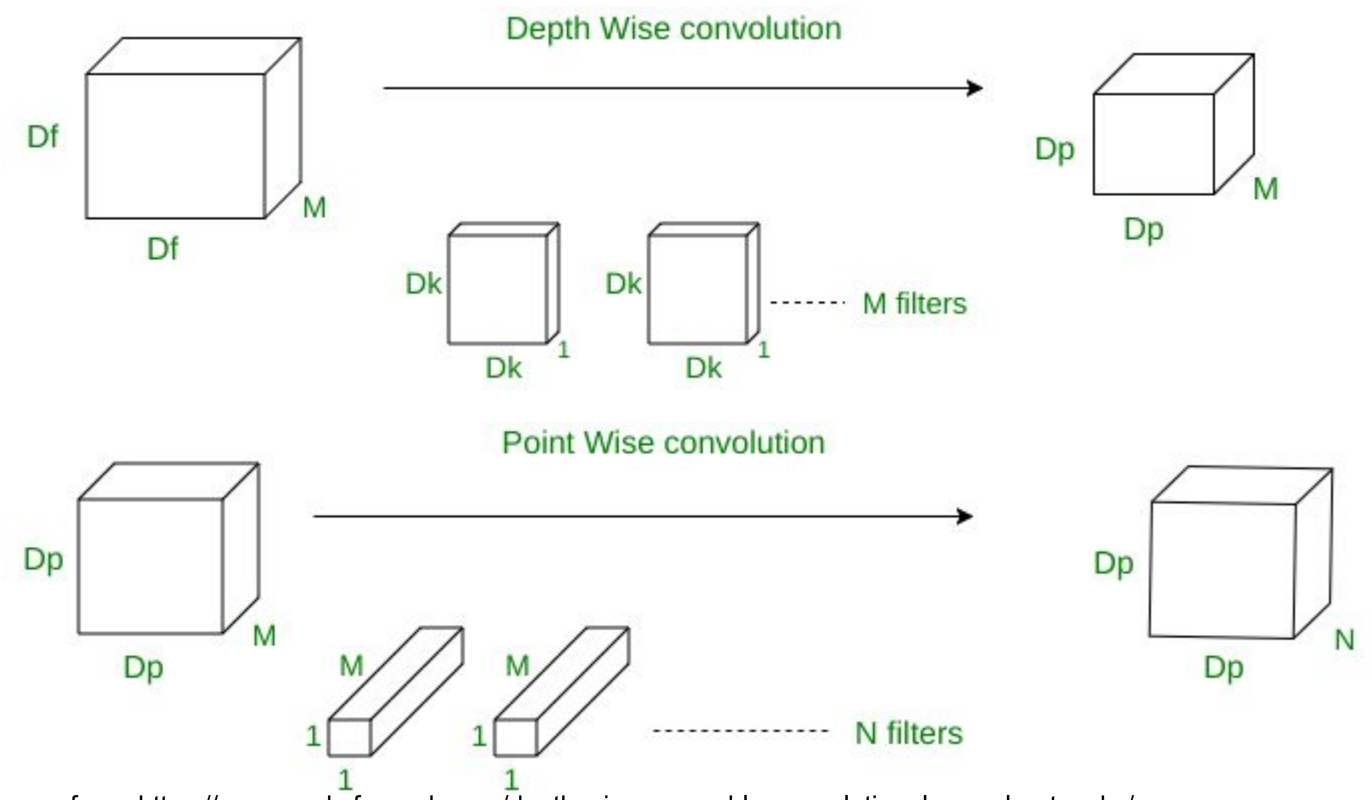


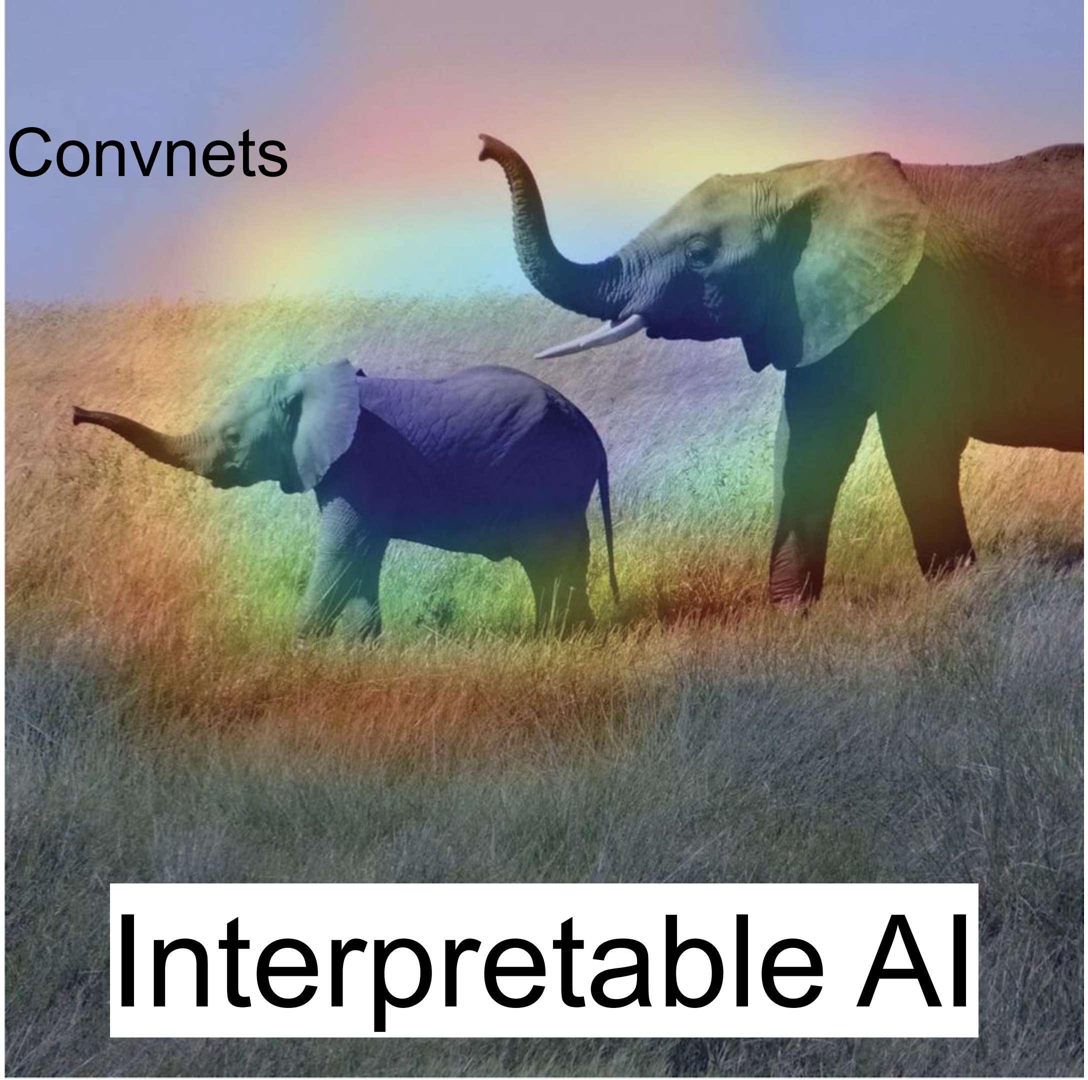

Image from: https://www.geeksforgeeks.org/depth-wise-separable-convolutional-neural-networks/

# Depthwise Separable Convolutions

| Type of Convolution | Complexity |
|---|---|
| Standard | $N \times Dp^2 \times Dg^2 \times M$ |
| Depth wise separable | $M \times Dp^2 \times (Dk^2 + N)$ |

Image from: https://www.geeksforgeeks.org/depth-wise-separable-convolutional-neural-networks/

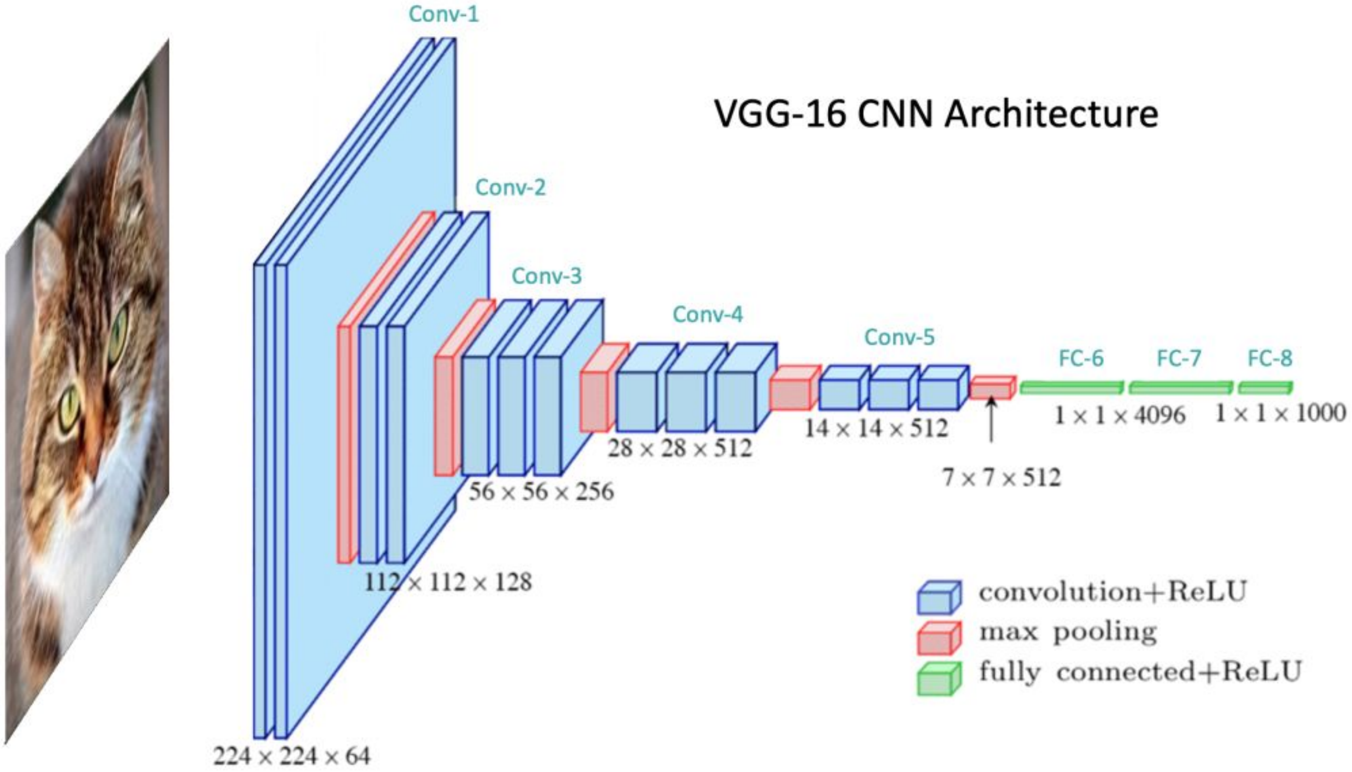
# Visualizing Convnets



# Visualizing ConvNets

- Visualizing Layer Activations
- Visualizing Filters
- Visualizing Class Activation Heatmaps

# Visualizing Layer Activations



# Visualizing Layer Activations

1. Take a test image
2. Feed it through each layer and save the output/activations
3. Plot

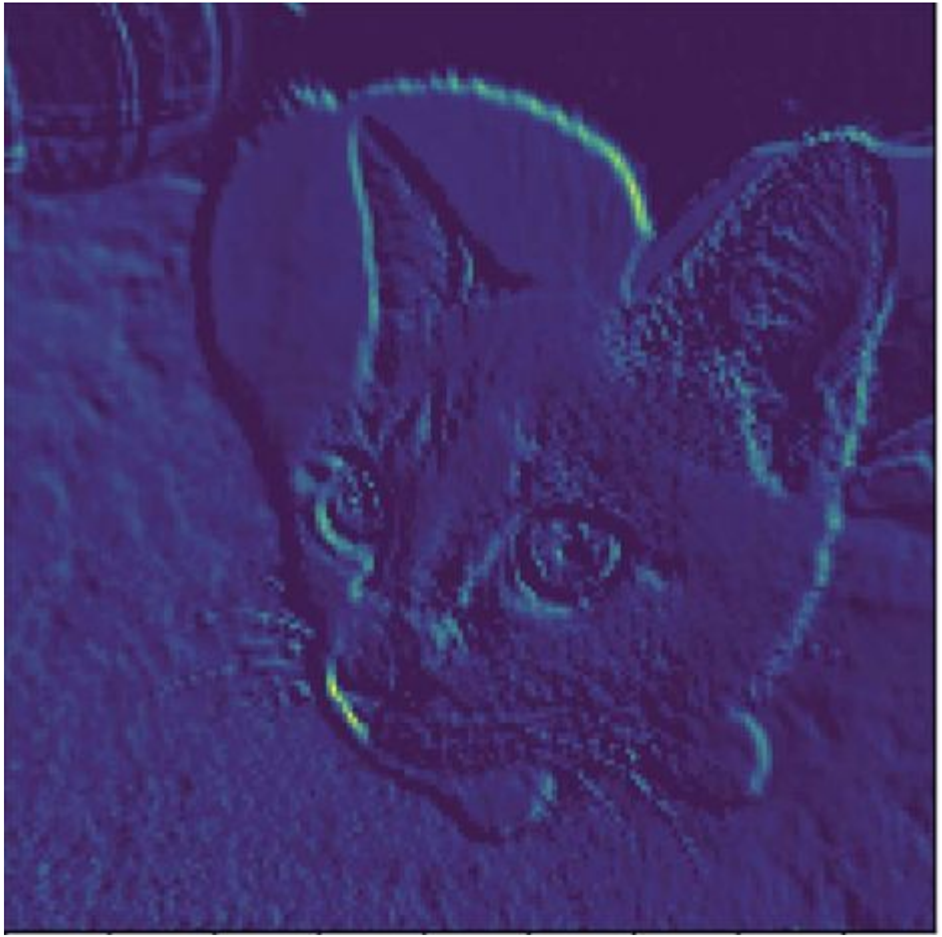


Image from: Deep Learning with Python (Chollet)

# Visualizing Layer Activations

Activations from Conv Layer 1

Activations from MaxPool Layer 1

Activations from Conv Layer 2

Activations from MaxPool Layer 2

...

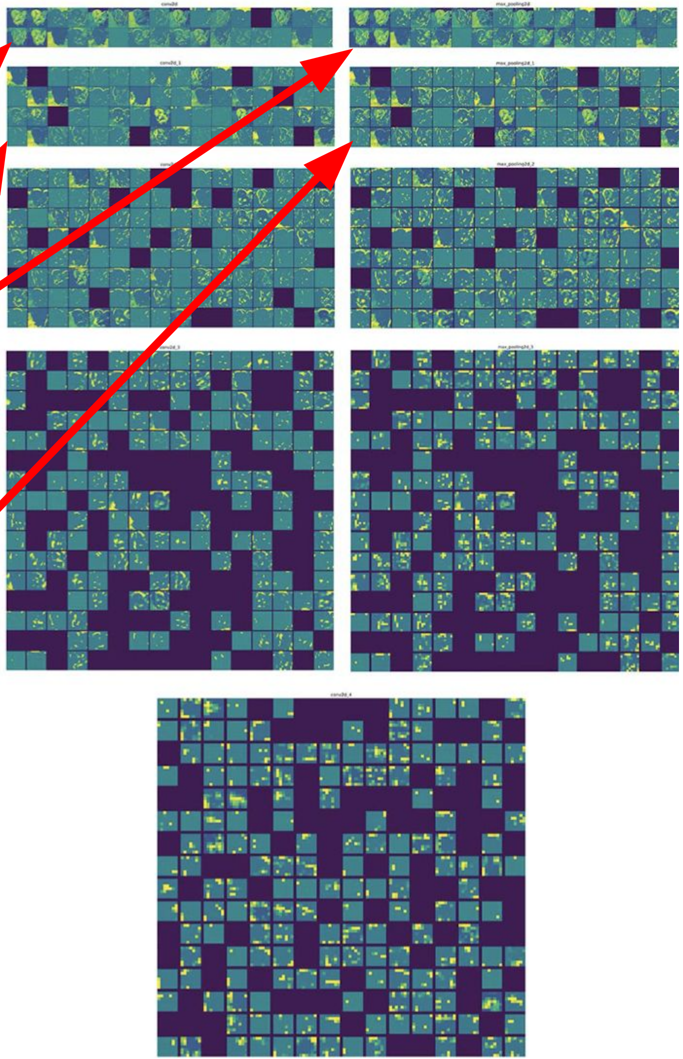


Image from: Deep Learning with Python (Chollet)

# Visualizing Layer Activations

You can see:

- The sparsity of higher layers (due to ReLu)
- The increasing abstraction of higher layers

# Visualizing Filters

- What image would the filter *maximally* respond to? What activates it the most?
- Start with a **blank image** and use **gradient descent** to change it until the filter responds maximally

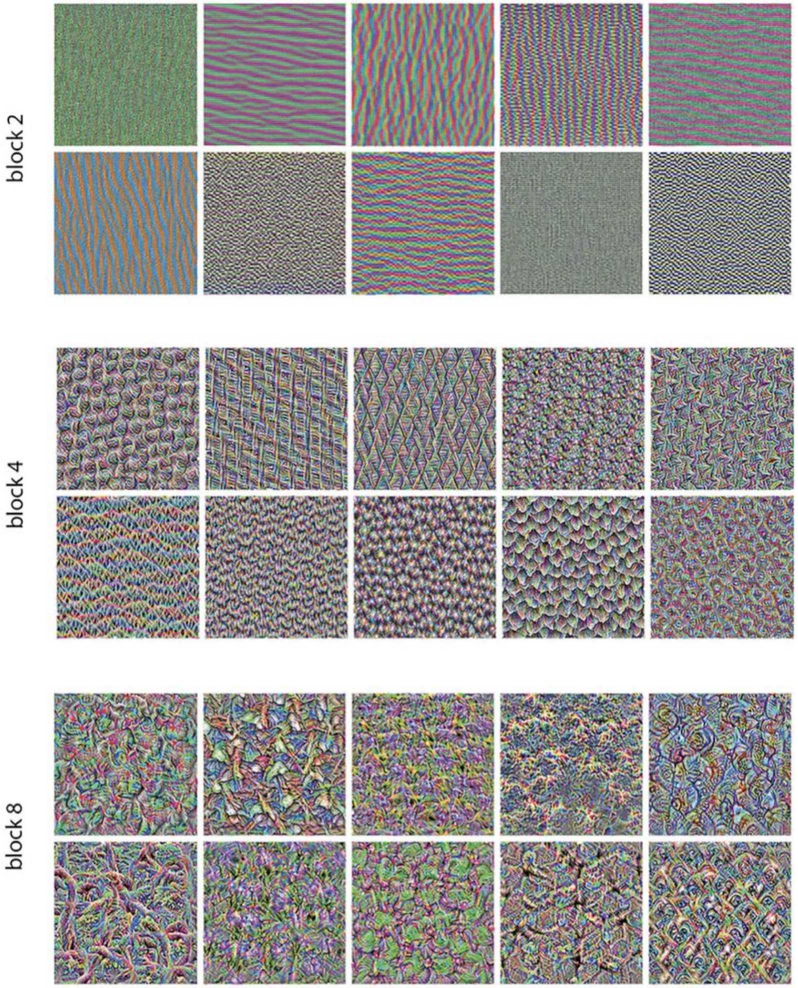



**Figure 9.17** **Some filter patterns for layers** `block2_sepconv1`, `block4_sepconv1`, **and** `block8_sepconv1`

# Visualizing Filters

- Filter seems to respond to horizontal lines

# Visualizing Class Activation Heatmaps

**Grad-CAM:** creates heatmaps of how intensely the input images activates the class

# Visualizing Class Activation Heatmaps

*How **intensely** the input image **activated different channels** in the last layer*

- Send image through network and get activations of final conv layer (before classification layer)

*How **important** each **channel** is with regard to that class*

- calculate the gradient for a specific class with respect to the activations of the final conv layer

# Visualizing Class Activation Heatmaps

Heatmap of how intensely the input images activates the class

# Visualizing Class Activation Heatmaps

Heatmap of how
intens
input

NOTE: We usually create these heatmaps for the top/predicted class but we CAN create it for any class

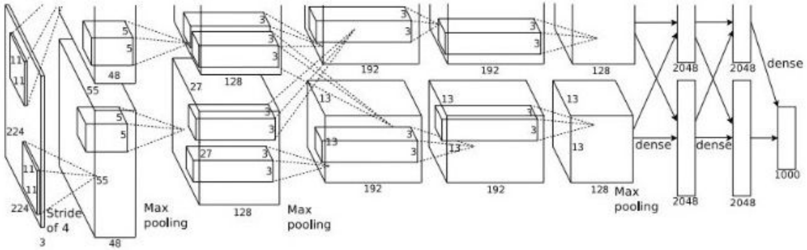
# Visualizing Class Activation Heatmaps


CAM would be helpful in situations like this!

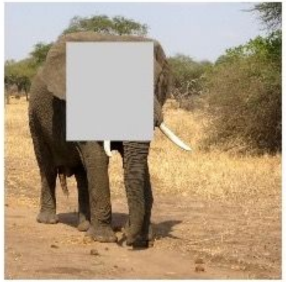
Image from: Besse, Philippe & Castets-Renard, Céline & Garivier, Aurélien & Loubes, Jean-Michel. (2018). Can Everyday AI be Ethical? Machine Learning Algorithm Fairness (english version). 10.13140/RG.2.2.22973.31207.

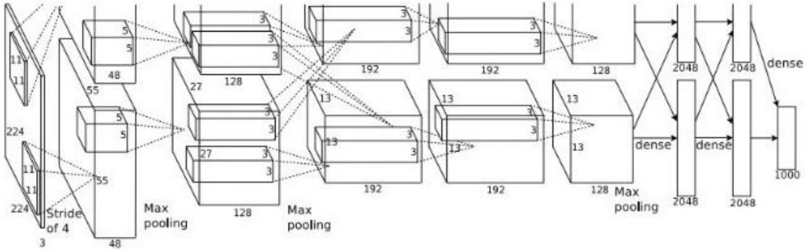
# Bonus: Occlusion

**Mask part of the image before feeding to CNN, check how much predicted probabilities change**




P(elephant) = 0.95






P(elephant) = 0.75

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

# Bonus: Tricking Convnets

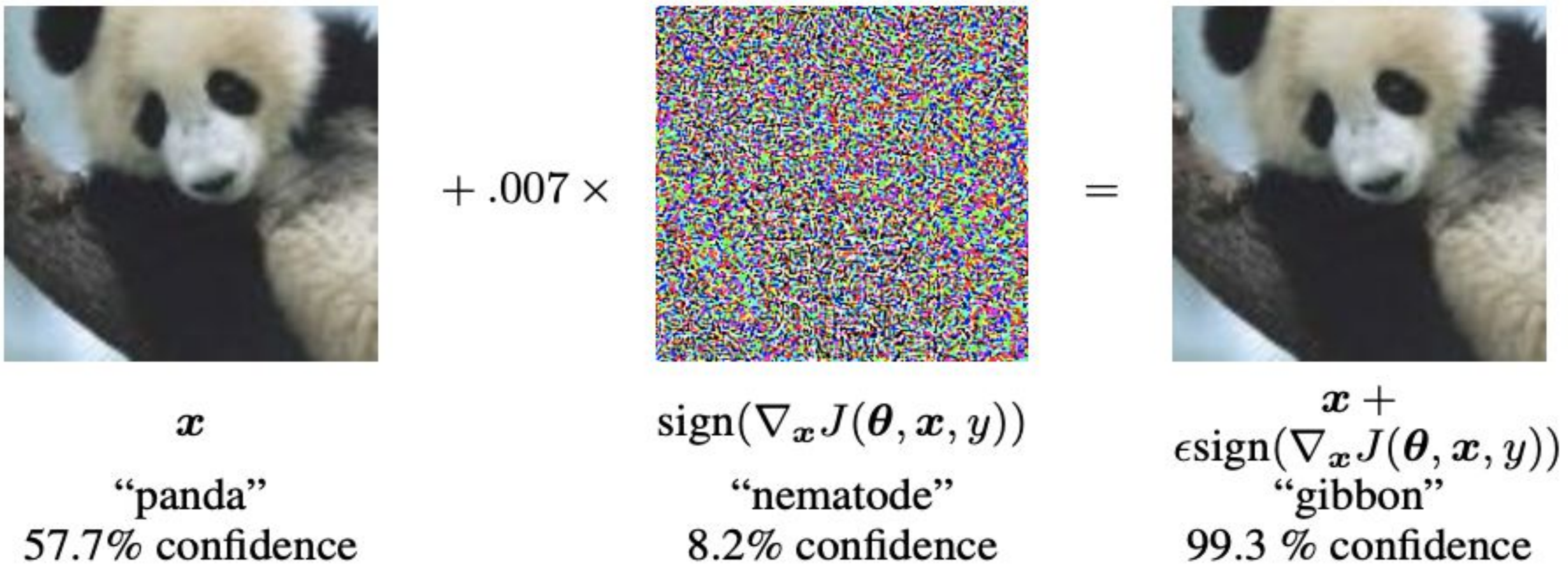



Image from: Explaining And Harnessing Adversarial Examples Goodfellow (2015)

# Types of Computer Vision Tasks

- Image Classification (done)
- Image Segmentation
- Object Detection
- Inpainting

# Image Segmentation

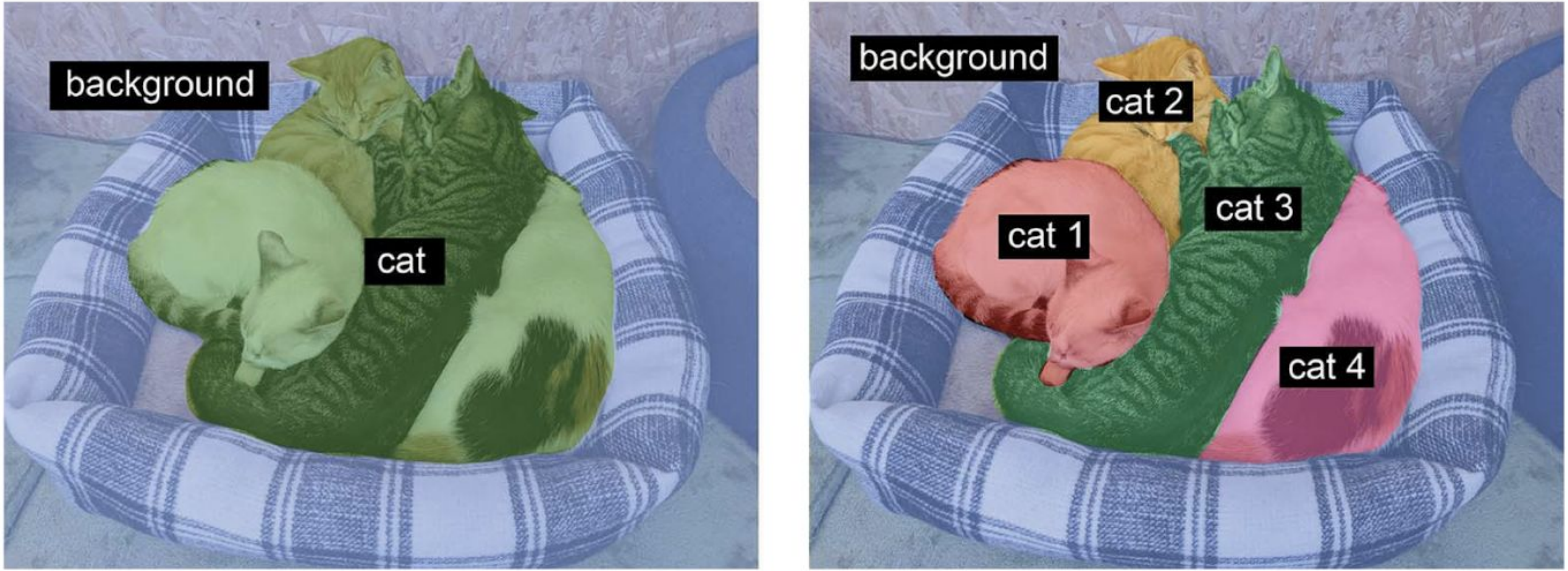
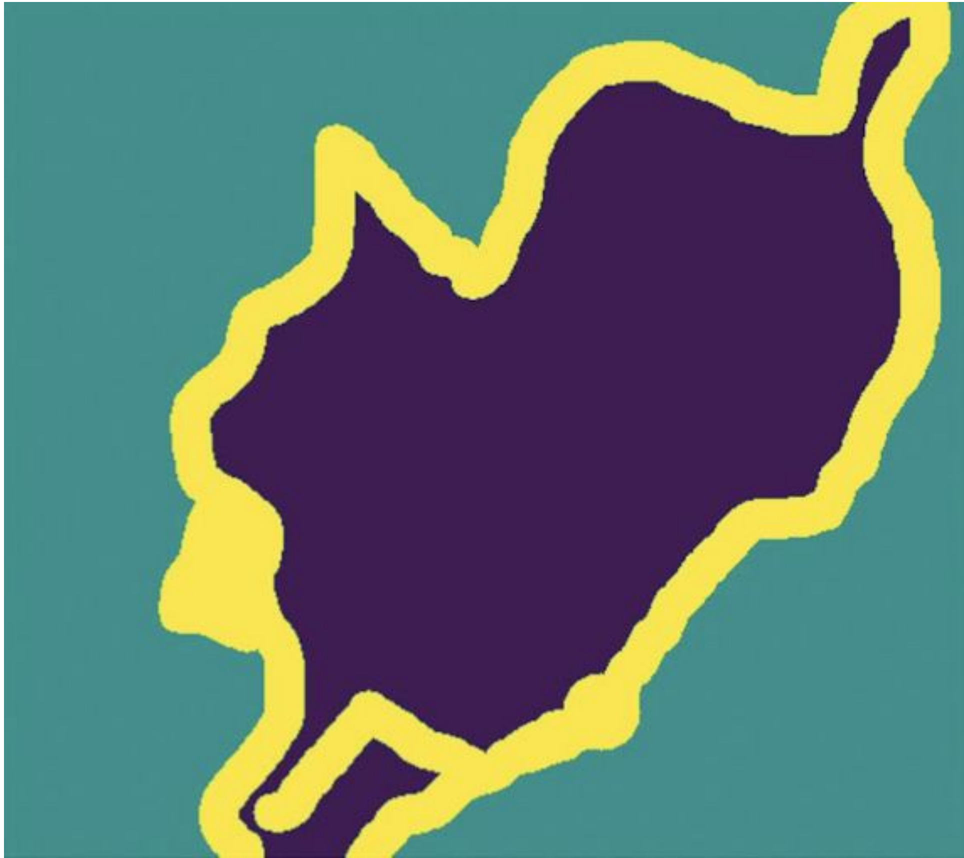


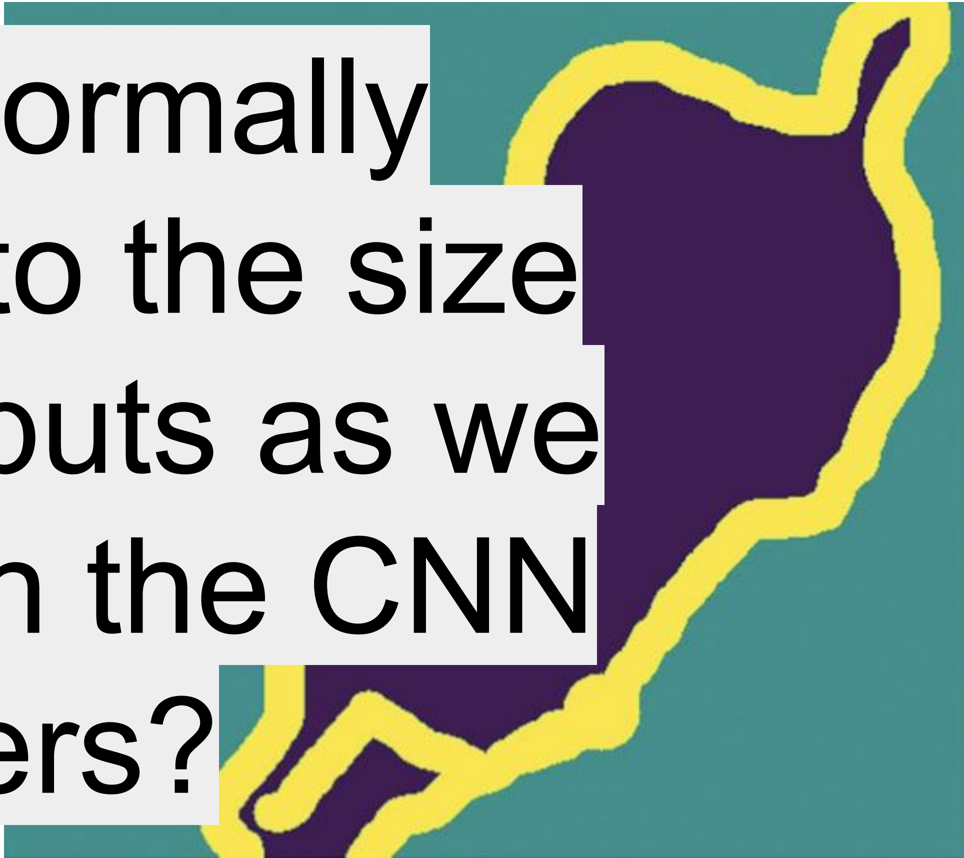

**Figure 9.2 Semantic segmentation vs. instance segmentation**

Image from: Deep Learning with Python (Chollet)

# Image Segmentation



# Image Segmentation



What normally happens to the size of our outputs as we go through the CNN layers?

# Object Detection

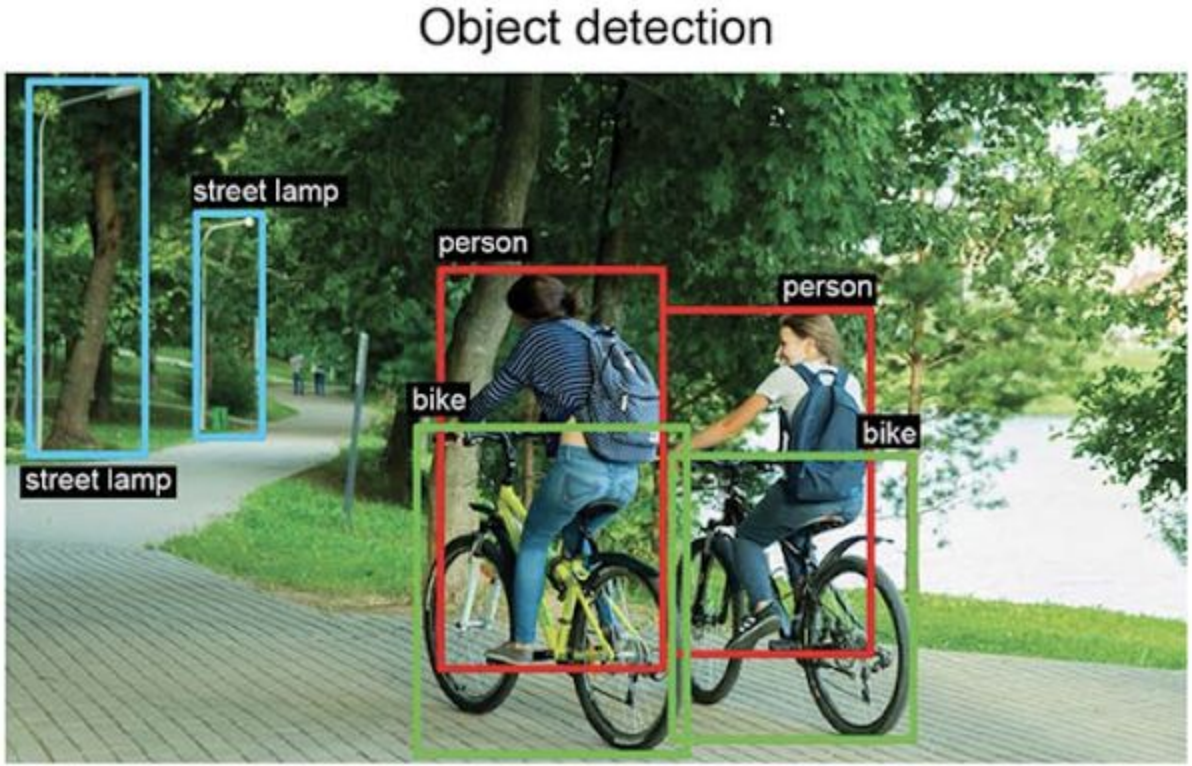



Image from: Deep Learning with Python (Chollet)

# Neural Inpainting

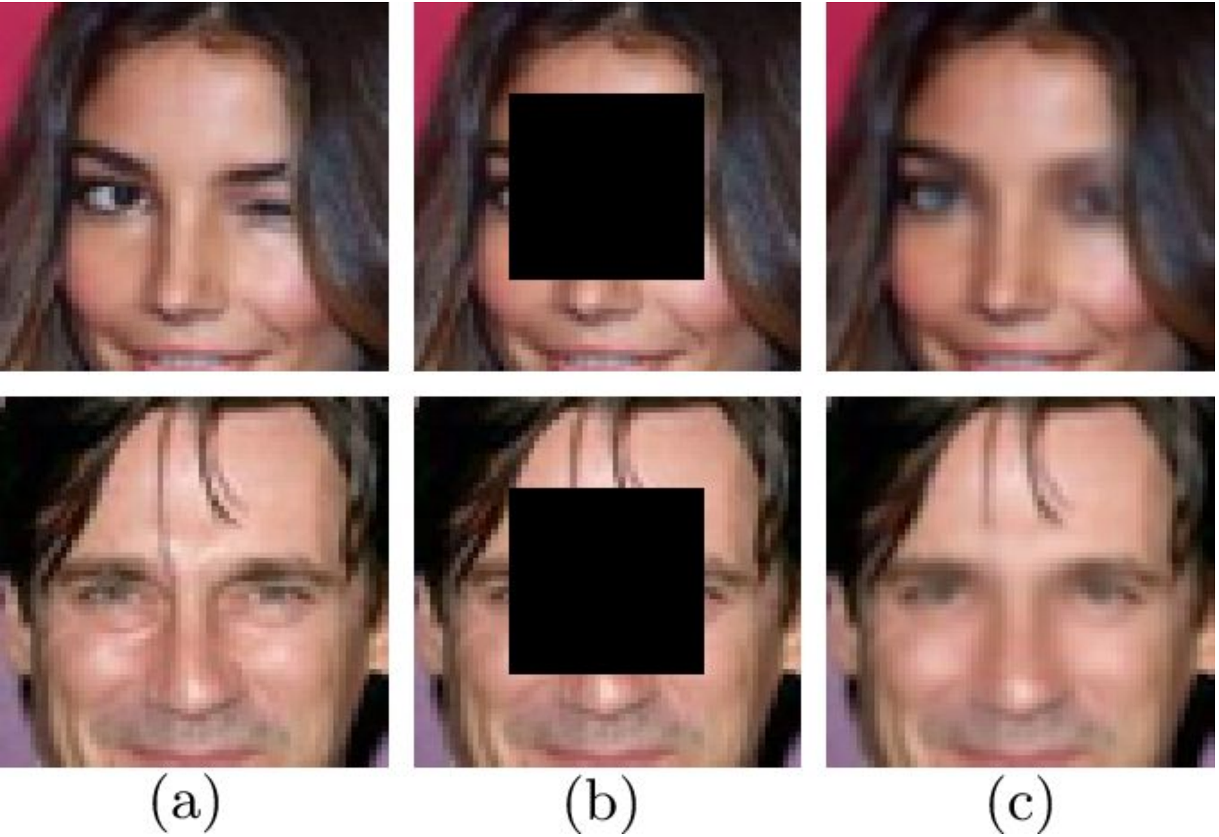



Image from: Altinel, Fazil & Ozay, Mete & Okatani, Takayuki. (2018). Deep Structured Energy-Based Image Inpainting.

# Pre-trained Models