Convolutional Neural Nets

Introduction

1. Convolutional neural nets, also called convnets or CNNs, are a neural net architecture especially suited to the structure in visual signals
2. The key idea of CNNs is to chop up the input image into little patches, and then process each patch independently and identically

Introduction

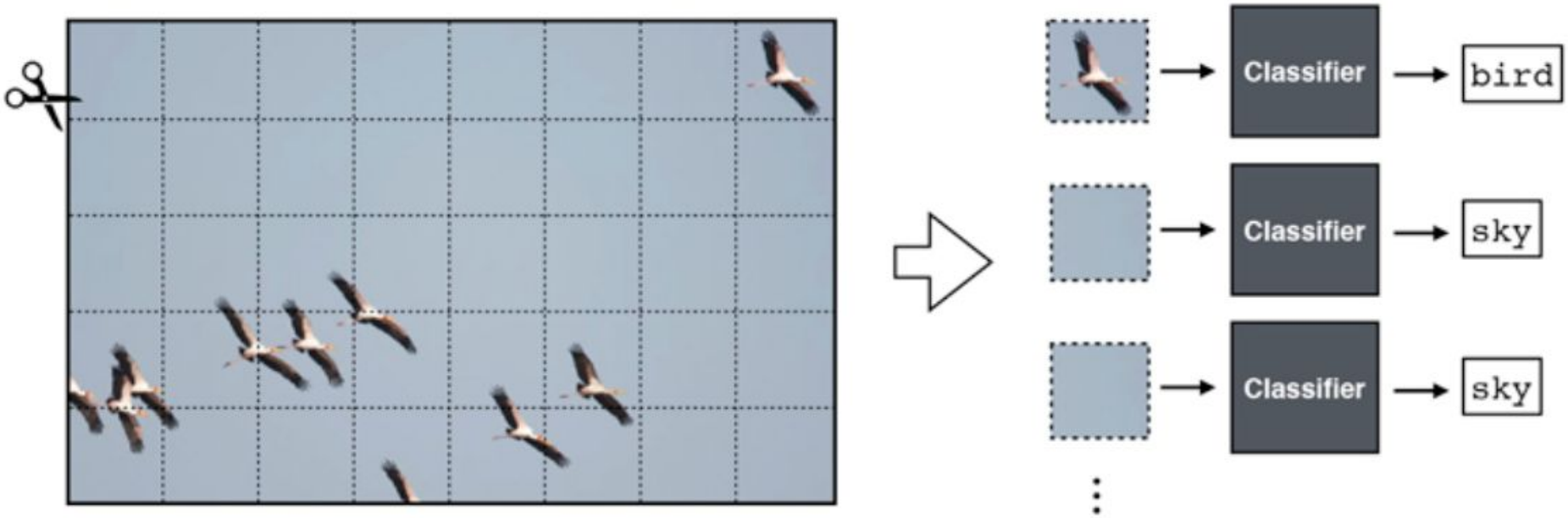


Figure 24.1: CNNs as patch processing. *Photo source*: Fredo Durand.

Introduction

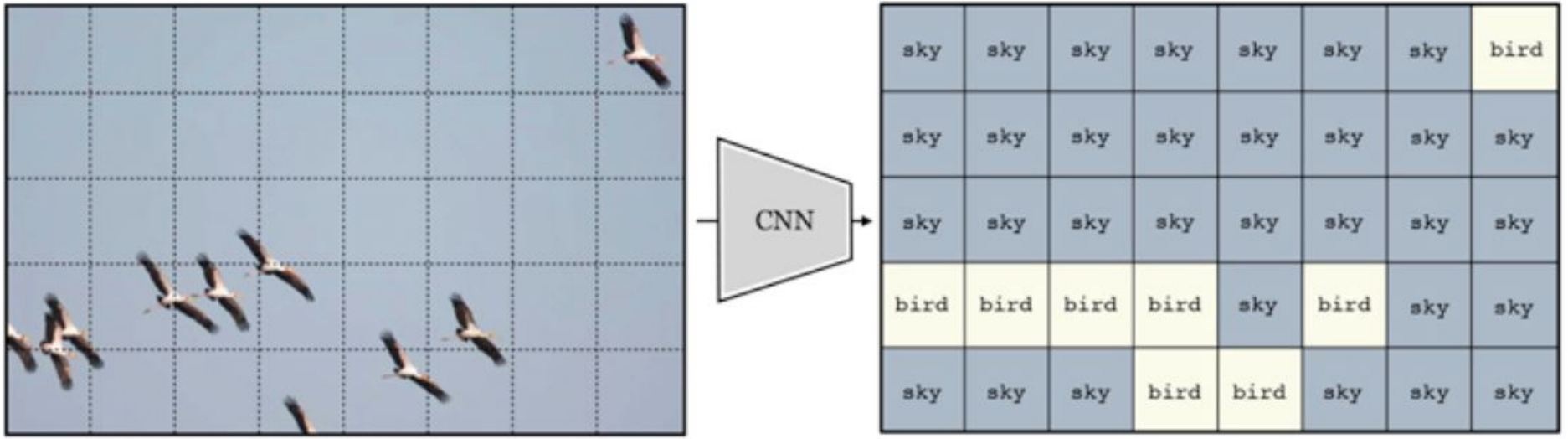


Figure 24.2: Input-output mapping of a CNN.

Introduction

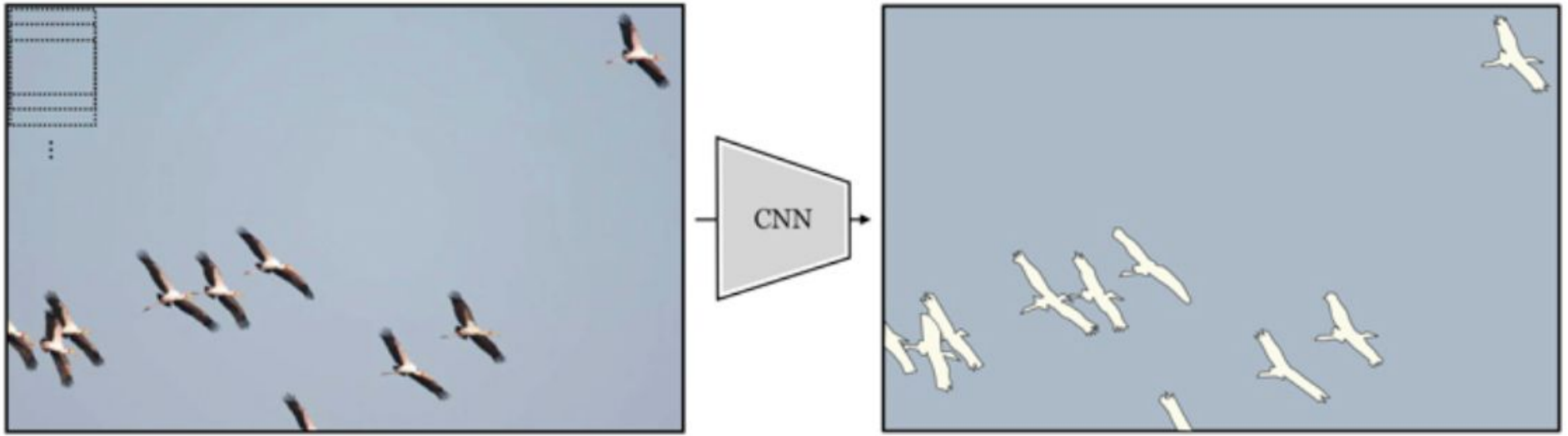


Figure 24.3: Dense input-output mapping.

Convolutional Layers

"Convolutional" layers in deep nets are typically actually defined as cross-correlations. We need not worry about the misnomer because whether you implement the layers with convolution or cross-correlation usually makes no difference for learning.

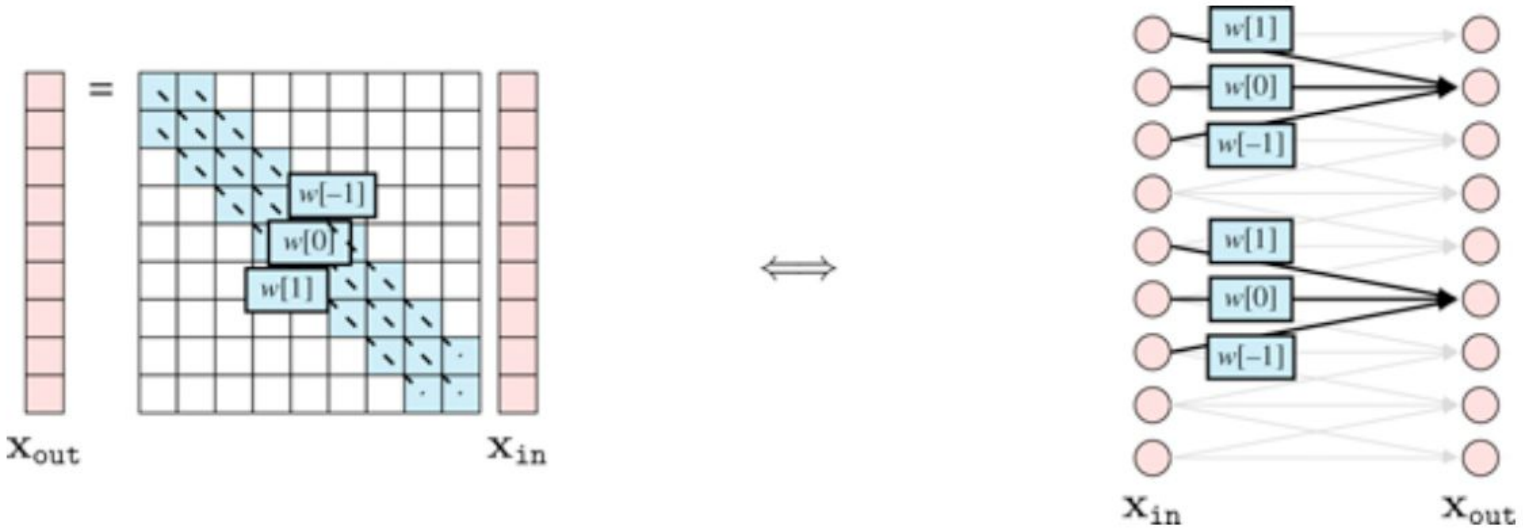


Figure 24.4: Two equivalent visualizations of a convolutional layer.

Multi-Input, Multi-Output Convolutional Layers

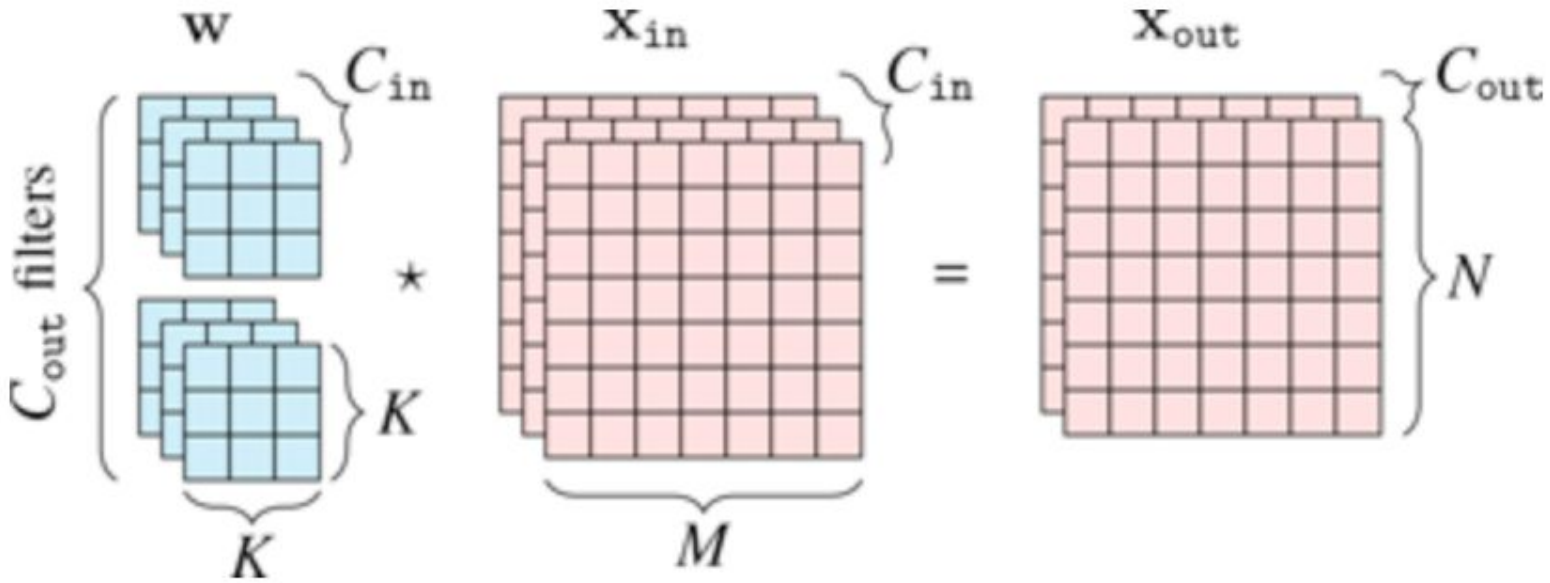


Figure 24.5: Multichannel convolution.

Multi-Input, Multi-Output Convolutional Layers

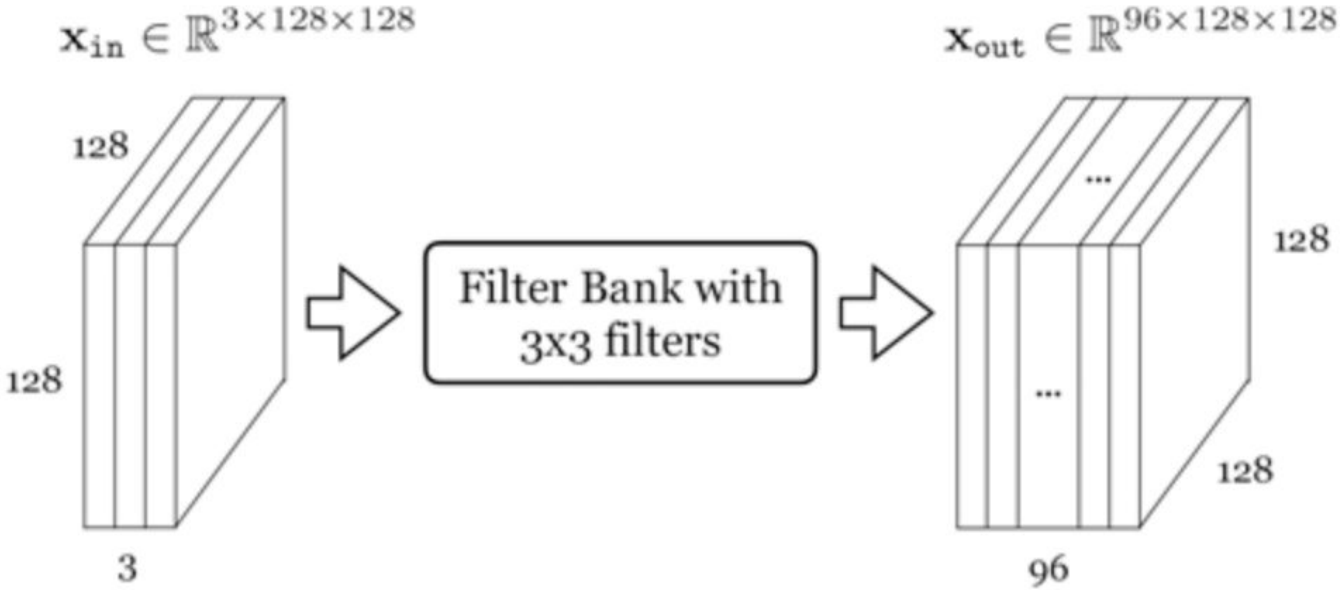


Figure 24.6: A convolutional layer that applies a bank of 3×3 filters. How many parameters does each filter have? How many filters are in the filter bank? *Source*: created by Jonas Wulff.

Separable Filters in CNNs

1. We can create a convolutional layer with separable filters by simply stacking two convolutional layers in sequence, with no other layers in between.
2. The first layer is a filter bank with K × 1 kernels and the second uses 1 × K kernels.
3. The composition of these layers is equivalent to a single convolutional layer with K × K separable filters.
4. In a neural network, one could use only separable filters for all the units and the learning could discover ways of combining them in order to build more complex, nonseparable kernels.

Downsampling and Upsampling Layers

1. Image pyramids can be used for analysis and synthesis of images
2. In CNNs this is done with downsampling and upsampling layers
3. Downsampling
    a. Strided convolution
    b. Pooling
4. Upsampling
    a. Dilates the signal then convoles
    b. It involves inserting zeros between the input pixels and then performing a standard convolution.

Nonlinear Filtering Layers - Pooling Layers

1. Like all downsampling layers, pooling layers can be used to reduce the resolution of the input tensor, removing high-frequency information in the signal.
2. Pooling is also particularly useful as a way to achieve invariance.
3. Convolutional layers produce outputs that are equivariant to translations of their input.
4. Pooling is a way to convert equivariance into invariance.

Normalization Layers

1. Another kind of non-linear filter is local normalization layer
2. Local normalization normalize within some neighbourhood
3. Although local normalization is a common structure within the brain, it is not very frequently used in current neural networks, which more often use global normalization layers like **batchnorm** or **layernorm**

A Simple CNN Classifier

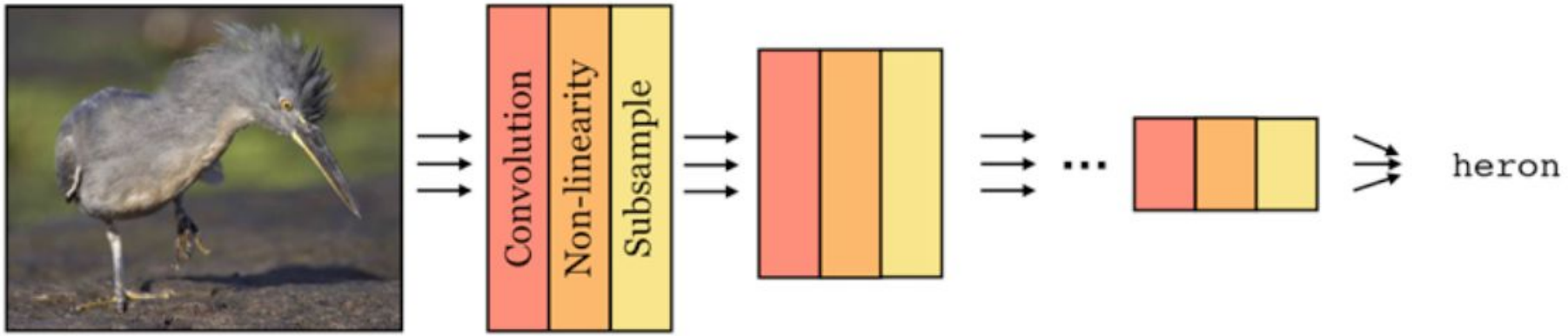
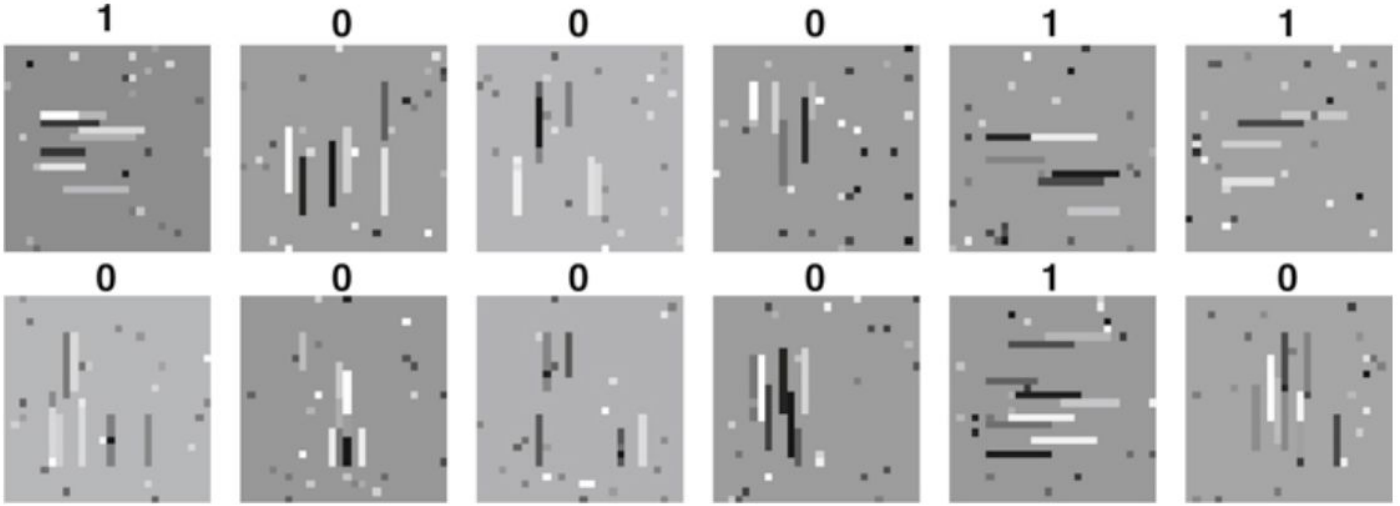


Figure 24.16: A CNN architecture for image classification. *Photo source*: Fredo Durand.

A Worked Example

1. We want to design a CNN that will classify the image according to the orientation of the lines that it contains. We define the two output classes as: 0 (vertical) and 1 (horizontal)



A Worked Example

1. To solve this problem we use the CNN with two convolutional channels C = 2 in the first layer.
2. 1 Conv layer → relu → gap → fc → softmax

Network Visualization

1. One important part of developing a system is to have tools to **prove**, **understand**, and **debug** it.
2. To understand, we need to visualize

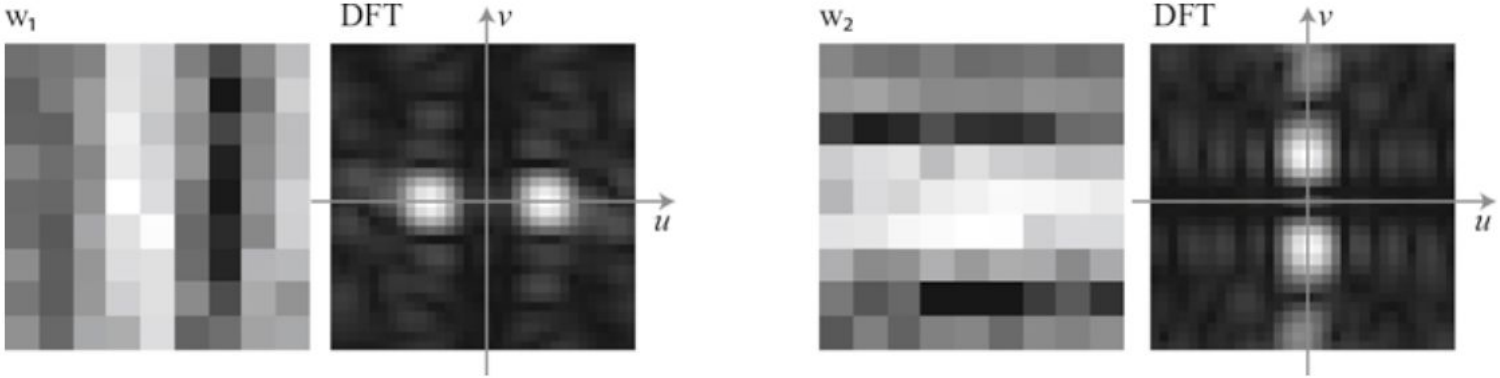


Figure 24.19: Visualization of the learned kernels.

Out-of-Domain Generalization

1. Our CNN works well for following examples

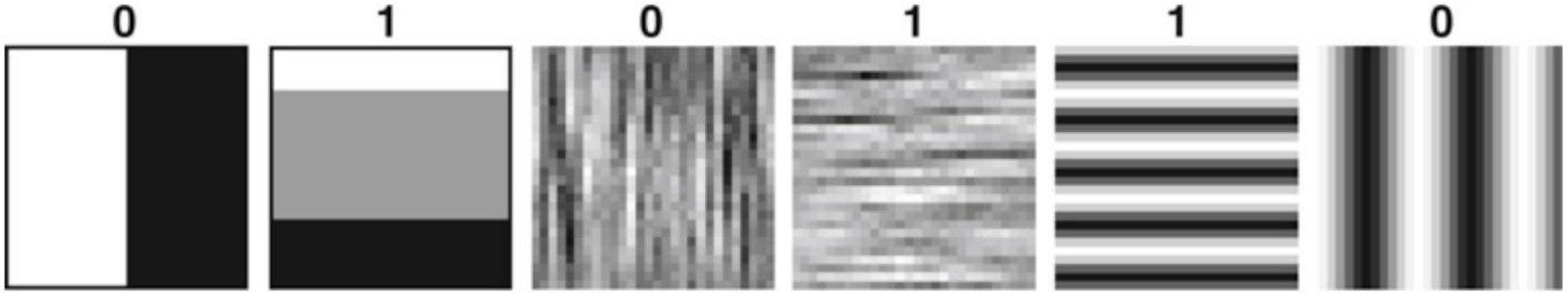


Figure 24.20: Out-of-domain test examples. The predicted label is shown at the top.

Identifying Vulnerabilities

1. Does the network solve the task that we had in mind?
2. Can we predict which inputs will make the output fail?
3. Can we produce test examples that to us look right but for which the network produces the wrong classification output?
4. The goal of this analysis is to identify weaknesses in the learned representation and in our training set (missing training examples, biases in our data, limitations of the architecture, etc.)

Identifying Vulnerabilities

1. We saw that the output of the first layer does not really look for lines, instead it looks at where the energy is in the Fourier domain.
2. So, we could fool the classifier by creating lines that for us look like vertical lines, but that have the energy content in the wrong side of the Fourier domain
3. We can use modulation

Identifying Vulnerabilities

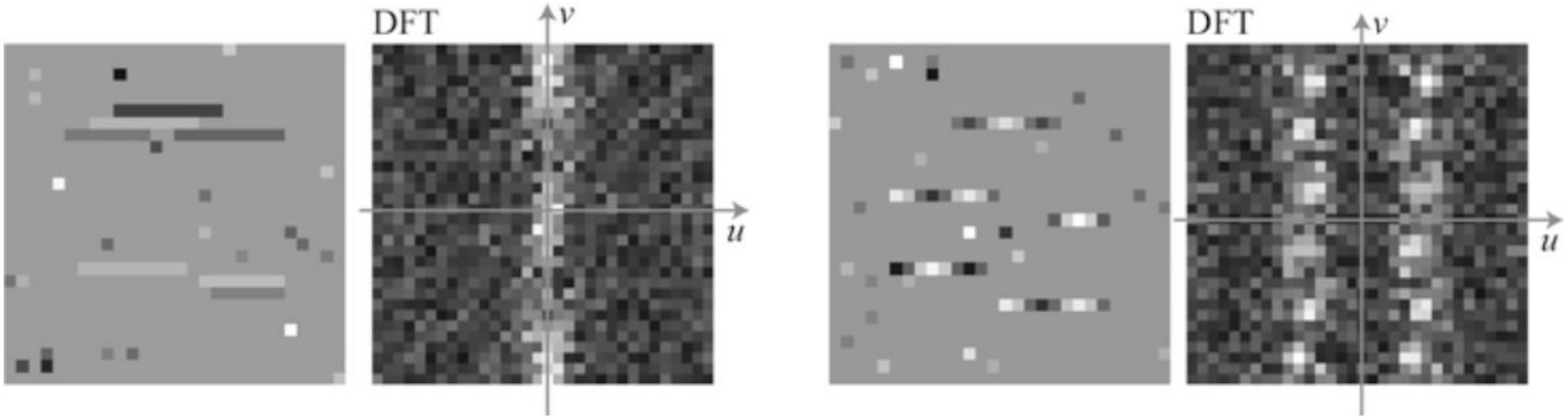


Figure 24.21: Creating a new out-of-domain test image obtained by multiplying an in-domain test image with horizontal lines (left image its DFT) with a sinusoidal wave. The resulting image and its DFT are shown on the right).

Identifying Vulnerabilities

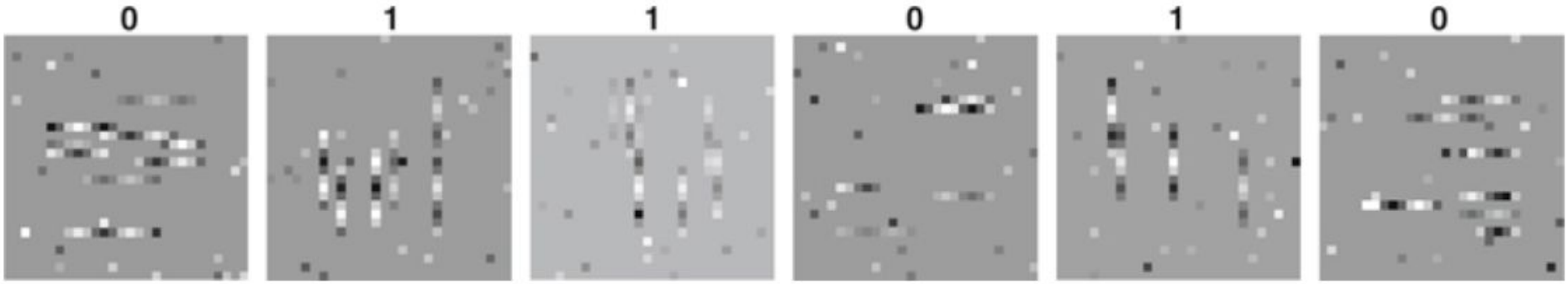


Figure 24.22: Classification results on out-of-domain test images created by modulation.

1. For complex architectures, adversarial examples are obtained as an optimization problem

Feature Maps

1.

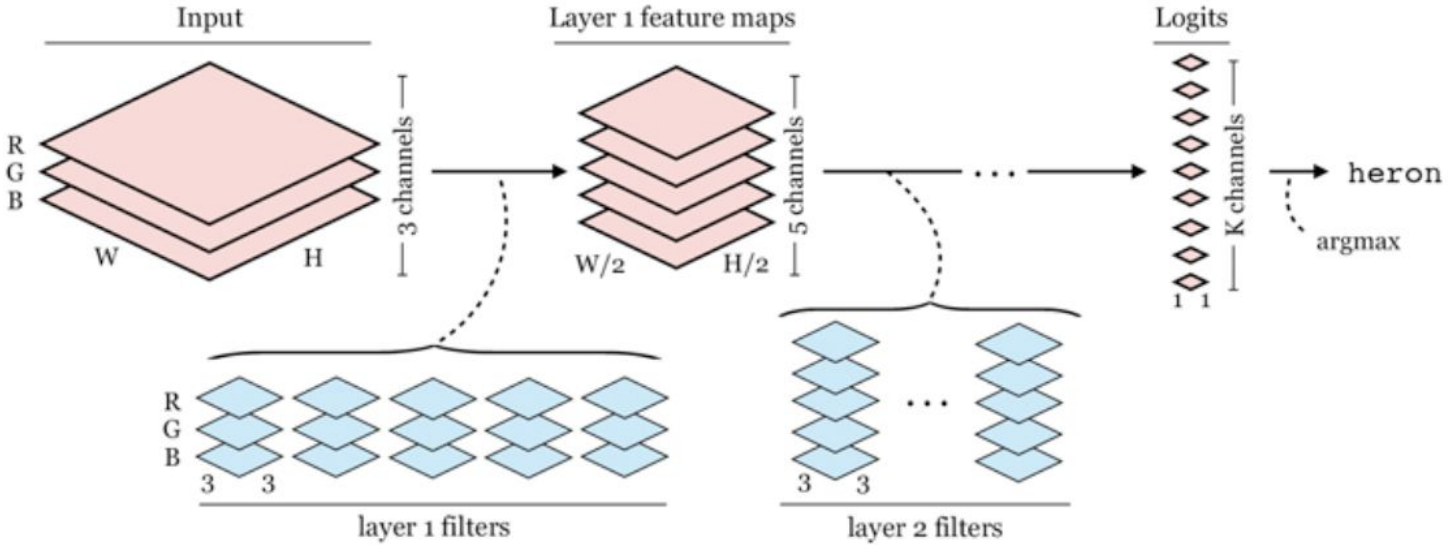


Figure 24.23: The interplay between feature maps and filters banks in a CNN. You can think of the input image itself as an R, G, B feature map and the output logits as a 1x1 resolution feature map with class logits as the channels.

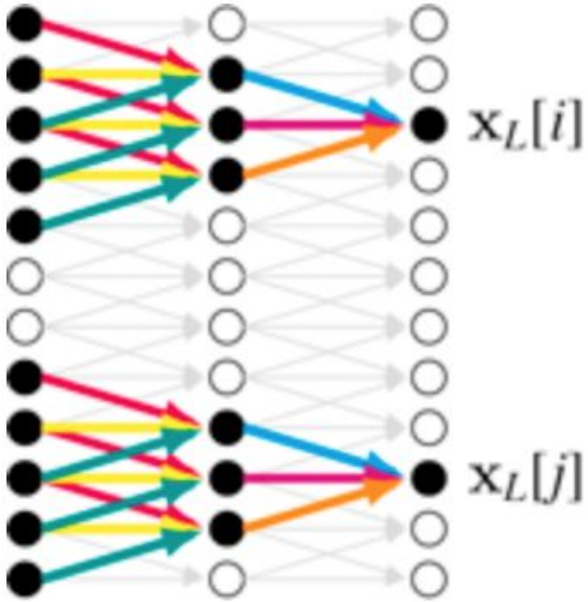
Feature Maps

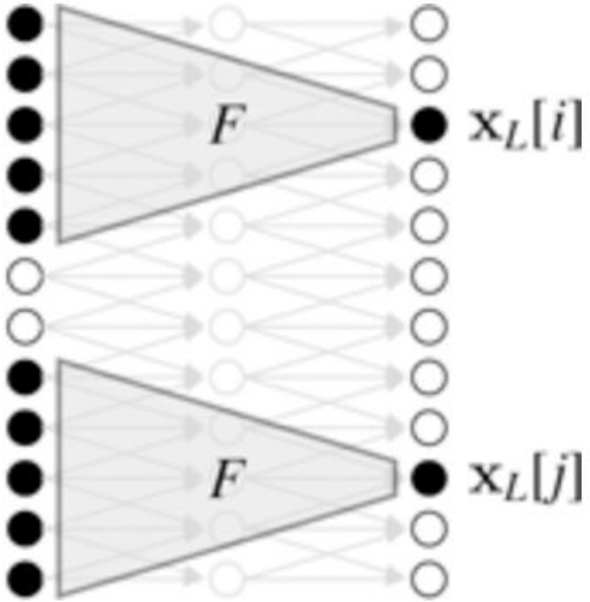
1. each layer downsamples and adds channels to partially compensate for the reduction in resolution.
2. While on the first layer the feature maps are sensitive to **basic patterns** in the input image – edges, lines, etc – the maps become more **abstracted** as we go deeper.

CNN as a Sliding Filter

1. Imagine the CNN has no pointwise nonlinearities.
2. Then the entire CNN is just the composition of a sequence of convolutions, which itself is a convolution
3. Now notice that this key property is unchanged when we add pointwise nonlinearities, because they introduce no interaction between neurons or pixels (they are pointwise after all)
4. “Hence it follows that a complete CNN, made up only of convolutional layers and pointwies nonlinearities, is itself a nonlinear operator that applies the same transformation independently and identically to each patch of the input signal, i.e. a nonlinear filter!”

CNN as a Sliding Filter





Why Process Images Patch by Patch?

1. Property #1: Treating Patches as Independent. This is a divide-and-conquer strategy.
    a. “The small problems are easier to solve than the original problem.”
    b. “The small problems can all be solved in parallel.”
    c. “This approach is agnostic to signal length”
    d. Chopping up into small patches like this is sufficient for many vision problems because the world exhibits locality: related things clump together, that is, within a single patch; far apart things can often be safely assumed to be independent.

Why Process Images Patch by Patch?

1. Property #2: Processing Each Patch Identically. For images, convolution is an especially suitable strategy because visual content tends to be translation invariant, and, the convolution operator is also translation invariant.
2.

Encoder and Decoder

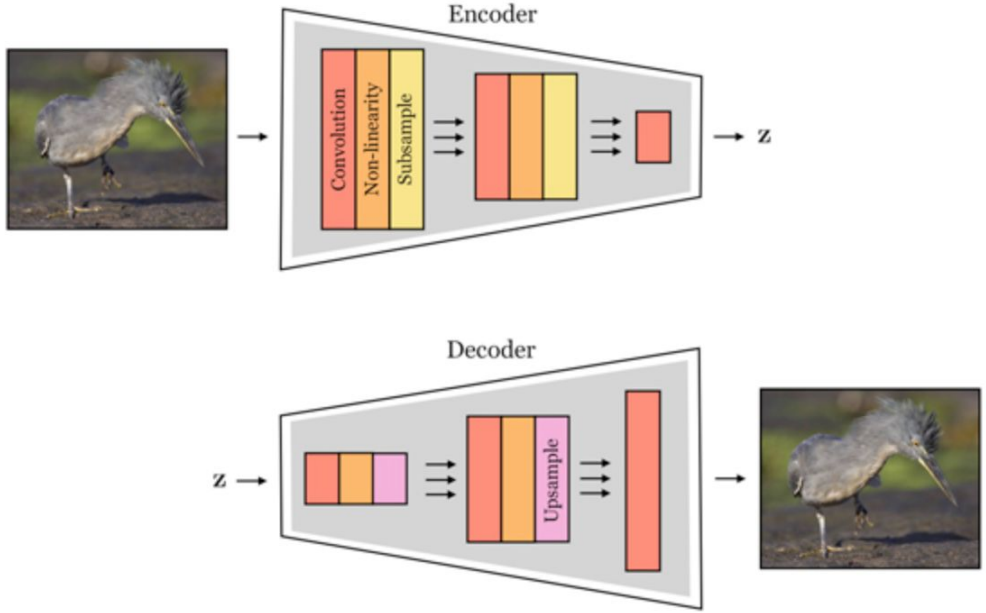


Figure 24.29: A convolutional encoder (top) and a convolutional decoder (bottom). The exact ordering of the operations (e.g., downsample before or after the non-linearity) is just an example and may vary in different encoder and decoder models. The **z** is a feature map or vector of neural activations, and is sometimes called an **embedding** (see chapter 30).

Encoder and Decoder

1. By downsampling, the internal feature maps are smaller, using less memory and computation
2. Encoder-decoders introduce an information bottleneck – i.e. the representation between the encoder and decoder is low-dimensional and can only transmit so much information – and this **forces abstraction**.

Encoder and Decoder

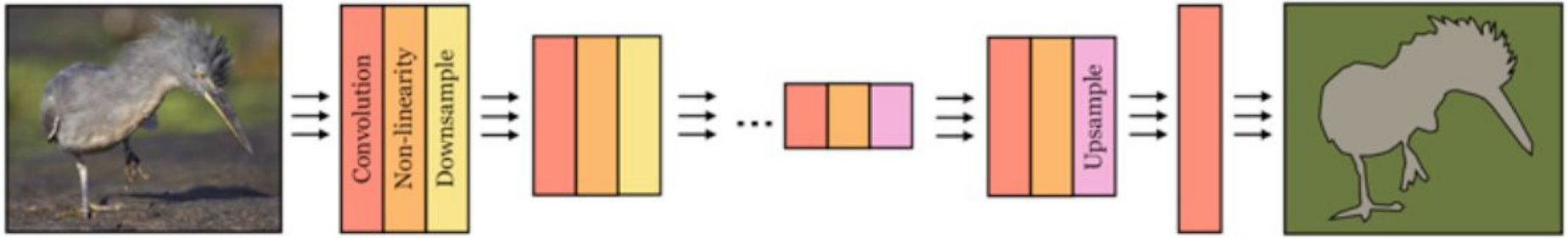


Figure 24.30: Encoder-decoder architecture. In this example, the input is an image and the output is a segmentation map.

U-Nets

1. “Adding skip connections to an encoder-decoder results in an architecture known as a U-Net”
2. “U-nets do this by concatenating the activations from the prior layer to the activations on the later layer, along the channel dimension”
3. “This architecture can maintain the information-bottleneck of the encoder-decoder, with its incumbent benefits in terms of memory and compute efficiency and forced abstraction, while also allowing residual information to flow through the skip connections, thereby not sacrificing the ability to output high-frequency spatial predictions

U-Nets

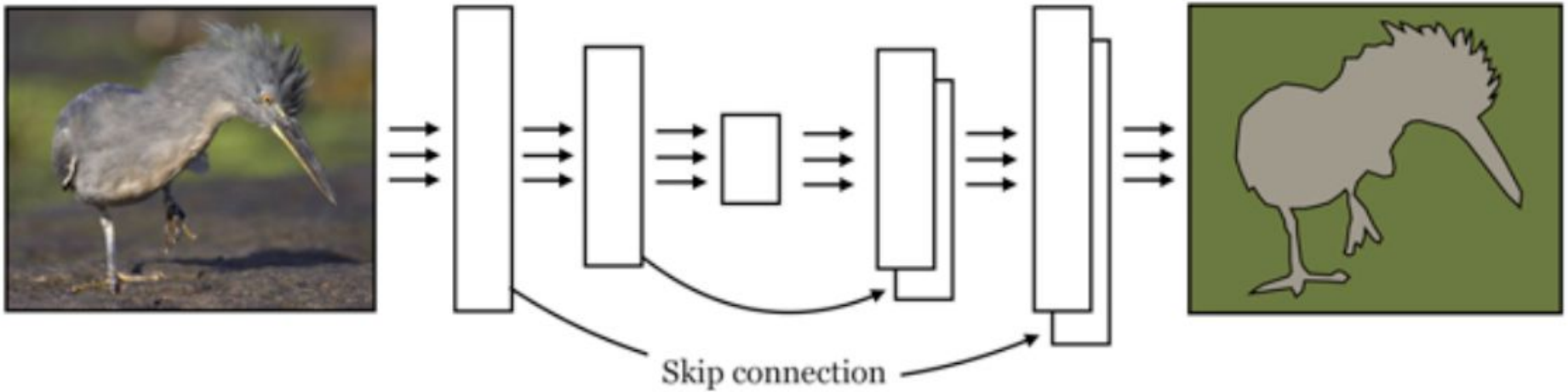


Figure 24.31: U-net architecture. Each block contains a series of layers. The skip connections concatenate activations.

Concluding Remarks

1. The underlying principles CNN embodies are ubiquitous in sensory processing
2. There are better models for solving image related problems that we will look at e.g. transformers

References

1. Foundations of Computer Vision - Chapter 24