Deep learning

9.1. Looking at parameters

François Fleuret

https://fleuret.org/dlc/

UNIVERSITÉ
DE GENÈVE

Understanding what is happening in a deep architectures after training is complex and the tools we have at our disposal are limited.

In the case of convolutional feed-forward networks, we can look at

- the network's parameters, filters as images,
- internal activations on a single sample as images,
- derivatives of the response(s) w.r.t. the input,
- maximum-response synthetic samples,
- adversarial samples.

We can also look at distributions of activations on a population of samples at different stages in a model.

Hidden units of a perceptron

Given a one-hidden layer fully connected network $\mathbb{R}^2 \to \mathbb{R}^2$

```
nb_hidden = 20

model = nn.Sequential(
    nn.Linear(2, nb_hidden),
    nn.ReLU(),
    nn.Linear(nb_hidden, 2)
)
```

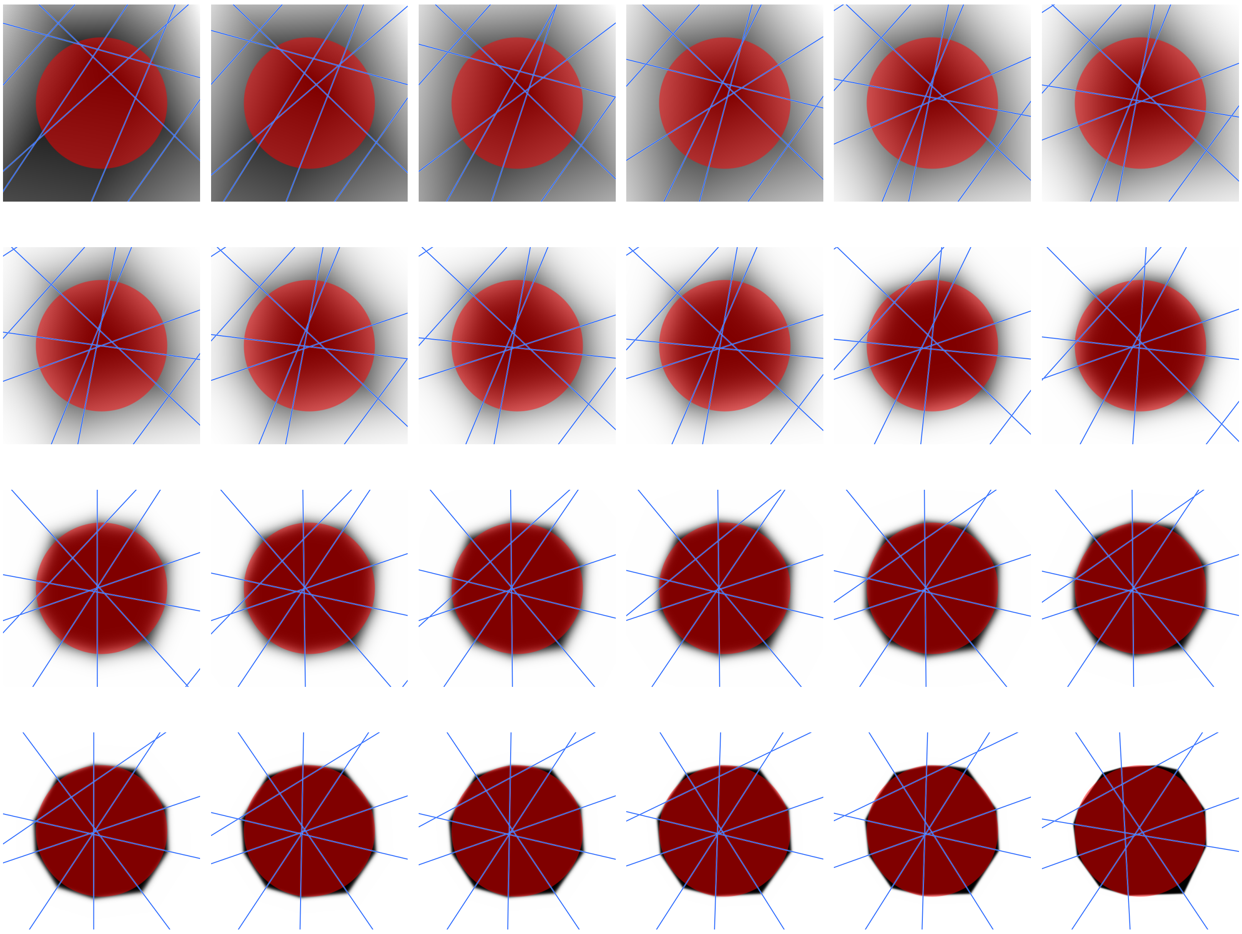
we can visit the parameters (w, b) of each hidden units with

```
for k in range(model[0].weight.size(0)):
    w = model[0].weight[k]
    b = model[0].bias[k]
```

and draw for each the line

$$\{\, x : w \cdot x + b = 0 \,\}.$$

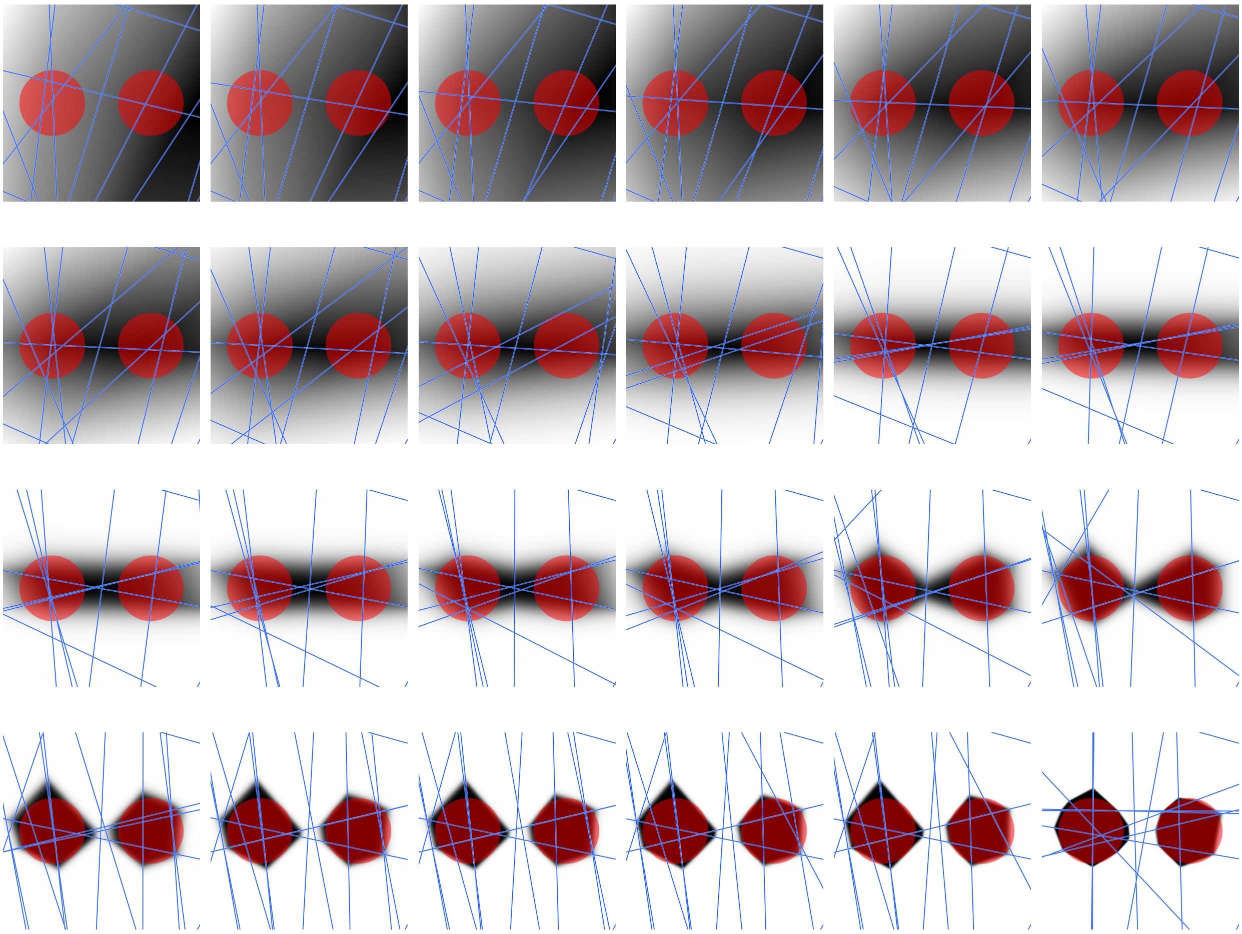
During training, these separations get organized so that their combination partitions properly the signal space.



Notes

- Each blue line corresponds to one hidden unit and depicts the hyperplane described in the previous slide.
- The red shape corresponds to the ground truth: the samples have a label 1 inside, and 0 outside.
- The shades of gray correspond to the response of the network, that is the difference between the two output units: the darker the shade of gray and the higher the prediction for class 1.

At first, the units are randomly dispatched by PyTorch's initialization of the linear layers.
As the training goes on, the hidden units are progressively organized in a way that partitions radially the disk



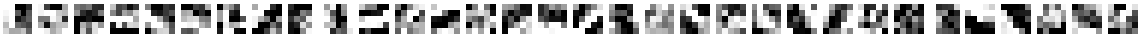
Convnet filters

A similar analysis is complicated to conduct with real-life networks given the high dimension of the signal.

The simplest approach for convnets consists of looking at the filters as images.

While it is quite reasonable in the first layer, since the filters are indeed consistent with the image input, it is far less so in the subsequent layers.

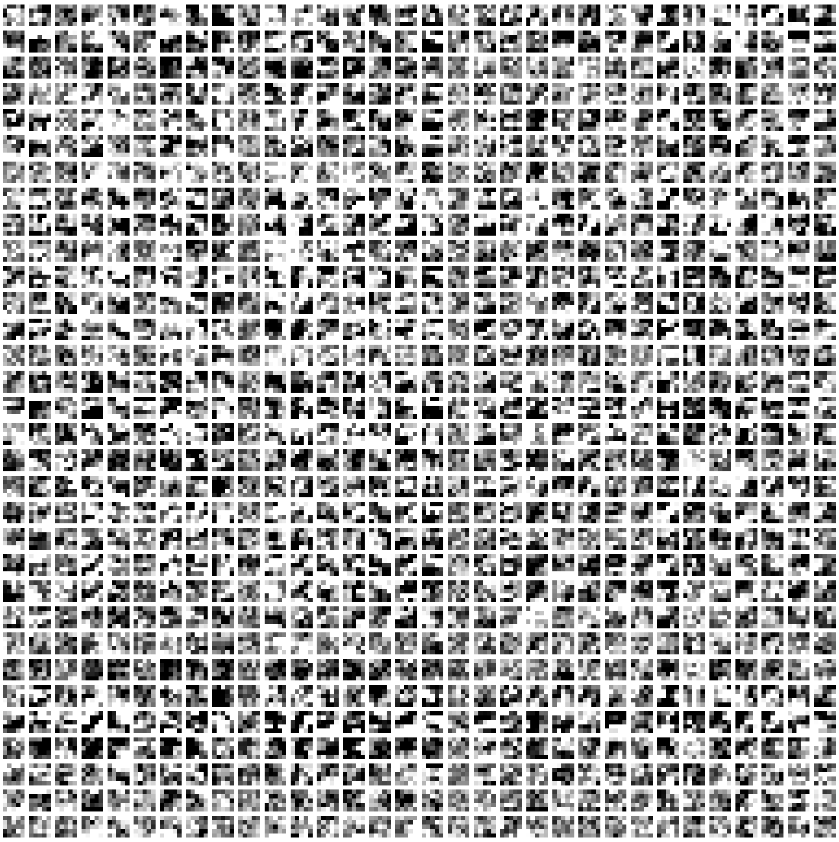
LeNet's first convolutional layer ($1 \rightarrow 32$), all filters



Notes

This images are the 32 filters of size 5×5 from the first hidden layer of a LeNet convnet trained on MNIST, represented as gray-scale images. These early filters look like edge-like patterns in different orientations which are reasonable structures for digit classification.

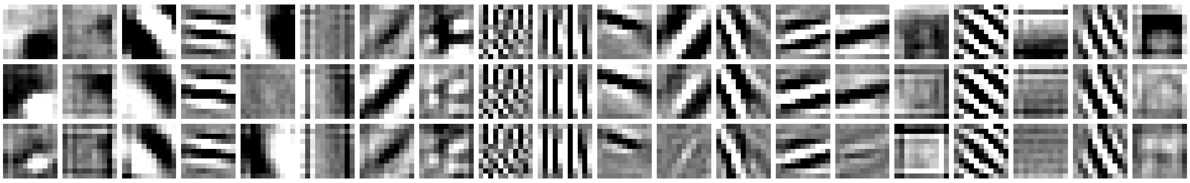
LeNet's second convolutional layer ($32 \rightarrow 64$), first 32 filters out of 64



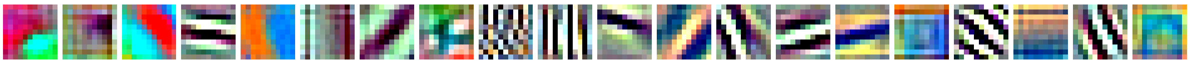
Notes

Each row correspond to one $32 \times 5 \times 5$ filter of the second convolutional layer. Each channel of a filter is represented as a 5×5 gray-scale image. This is harder to interpret than in the previous slide, as there are more channels and the input being the activations of the first layer after a non-linearity.

AlexNet's first convolutional layer ($3 \rightarrow 64$), first 20 filters out of 64

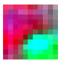


or as RGB images

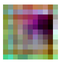


Notes

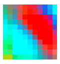
For AlexNet (pretrained model on ImageNet from PyTorch hub), the filters of the first layer take as input an RGB image and are of size $3 \times 11 \times 11$. Each column of the top image represents the three channels of one filter, which can also be represented as a RGB image on the bottom image.



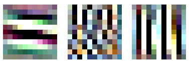
The first filter responds to a round green spot on red



This filter detects a dark spot in the top right corner



This filter corresponds to a diagonal red strip on blue



These filters respond to strip patterns

These filters correspond to filter banks, color configurations, and texture patterns which can be expected as low-level image parts.

AlexNet's second convolutional layer (64 → 192). First 15 channels (out of 64) of the first 20 filters (out of 192).

Notes

In the second layer of AlexNet, there are 192 filters of size $64 \times 5 \times 5$. We represent here the first 15 channels of the first 20 filters.
As before with LeNet, it is difficult to understand exactly the function of the resulting hidden units. A better way as we will see later is to engineer the input signal to maximize the response of a unit, and understand its role.