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)
)
```

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

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

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)
)
```

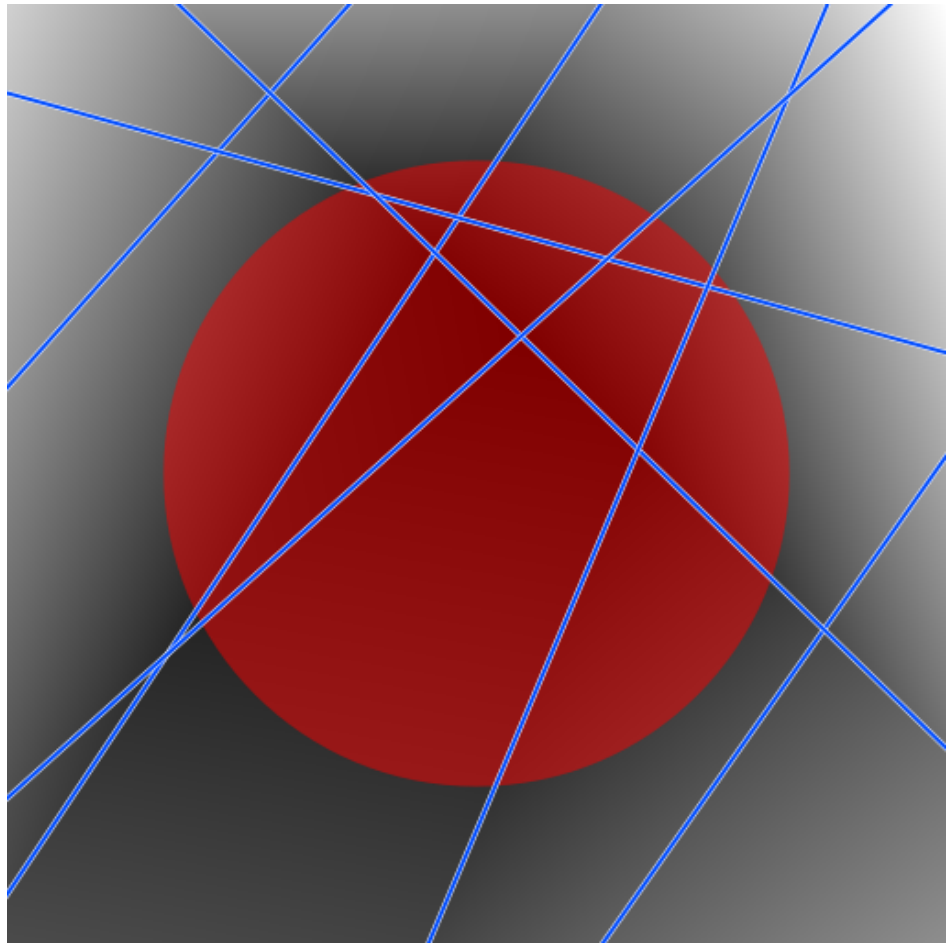
we can visit the parameters (w, b) of each hidden units with

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

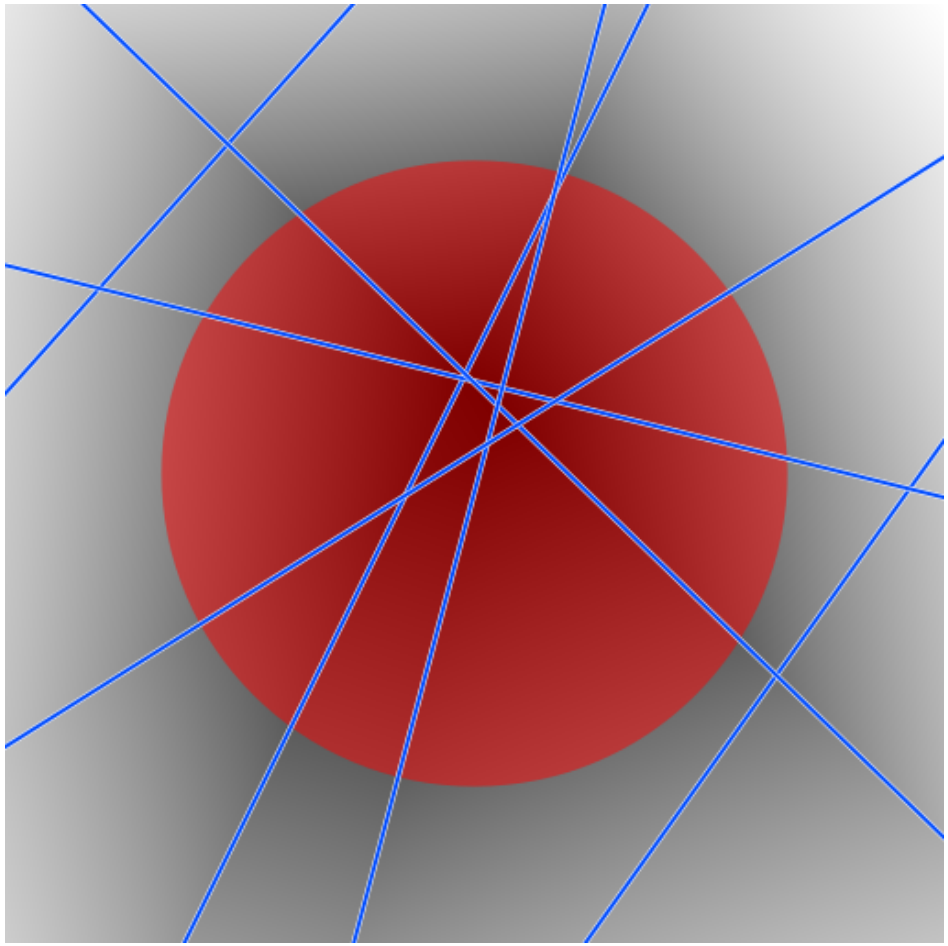
and draw for each the line

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

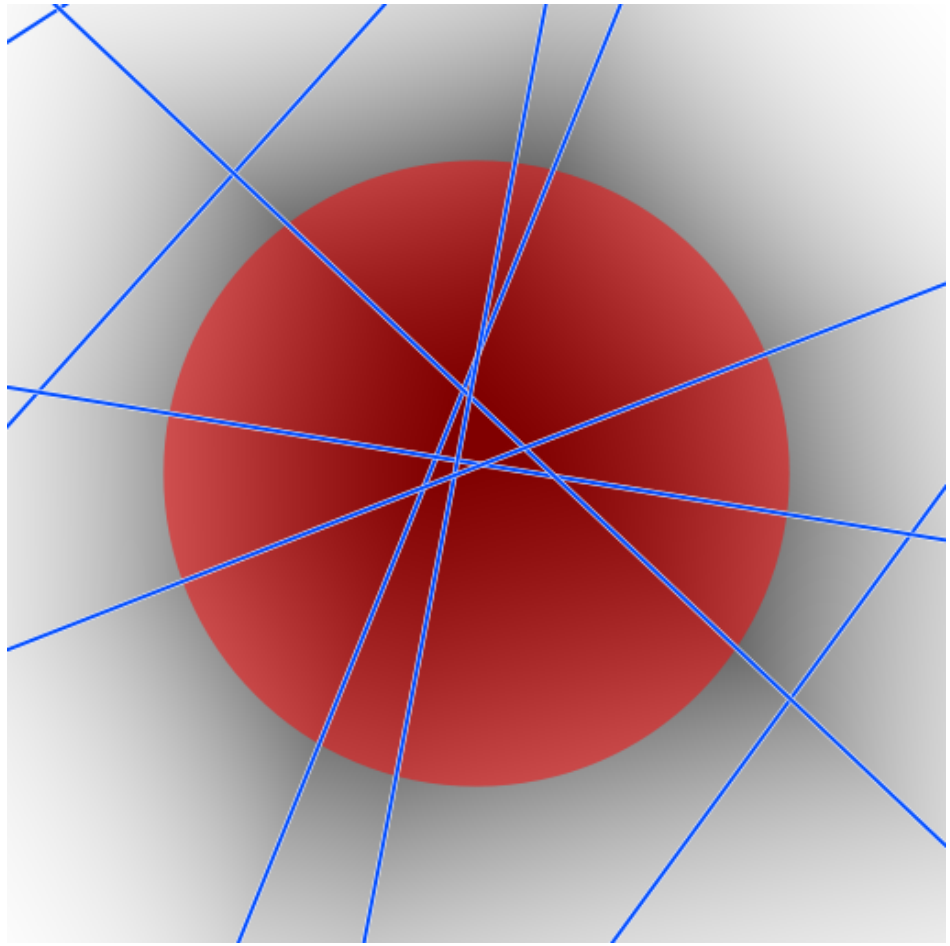
During training, these separations get organized so that their combination partitions properly the signal space.



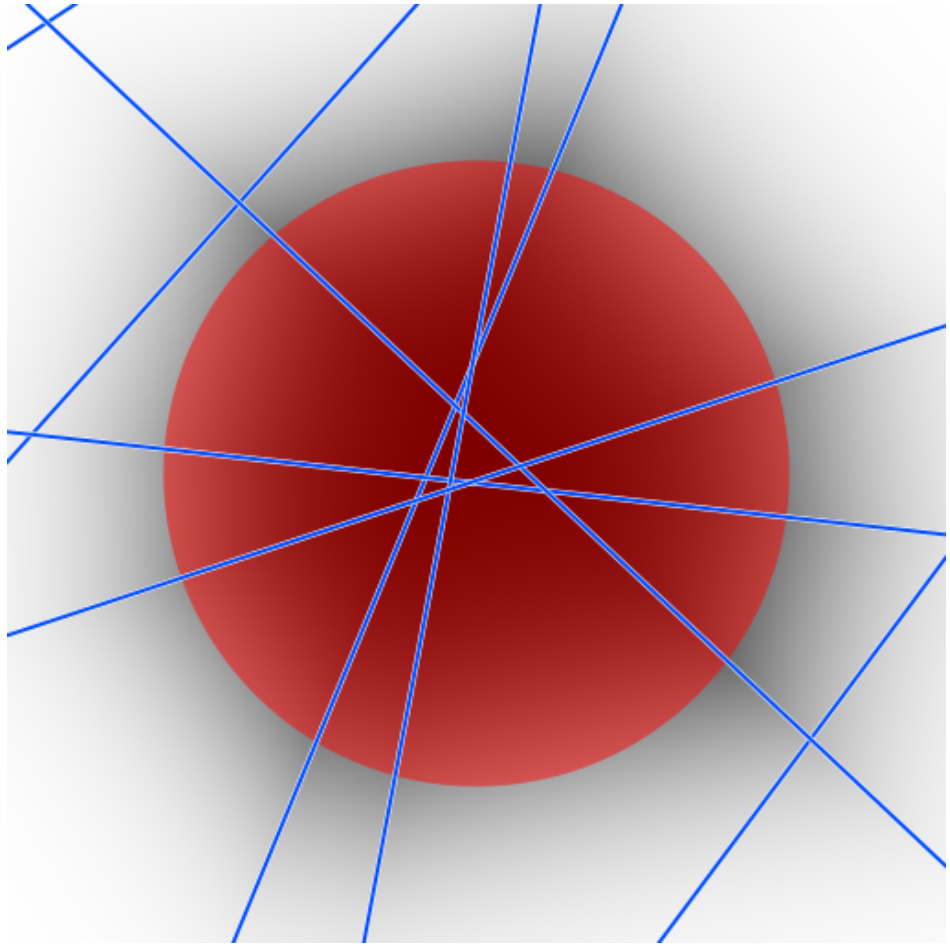
Iteration 1



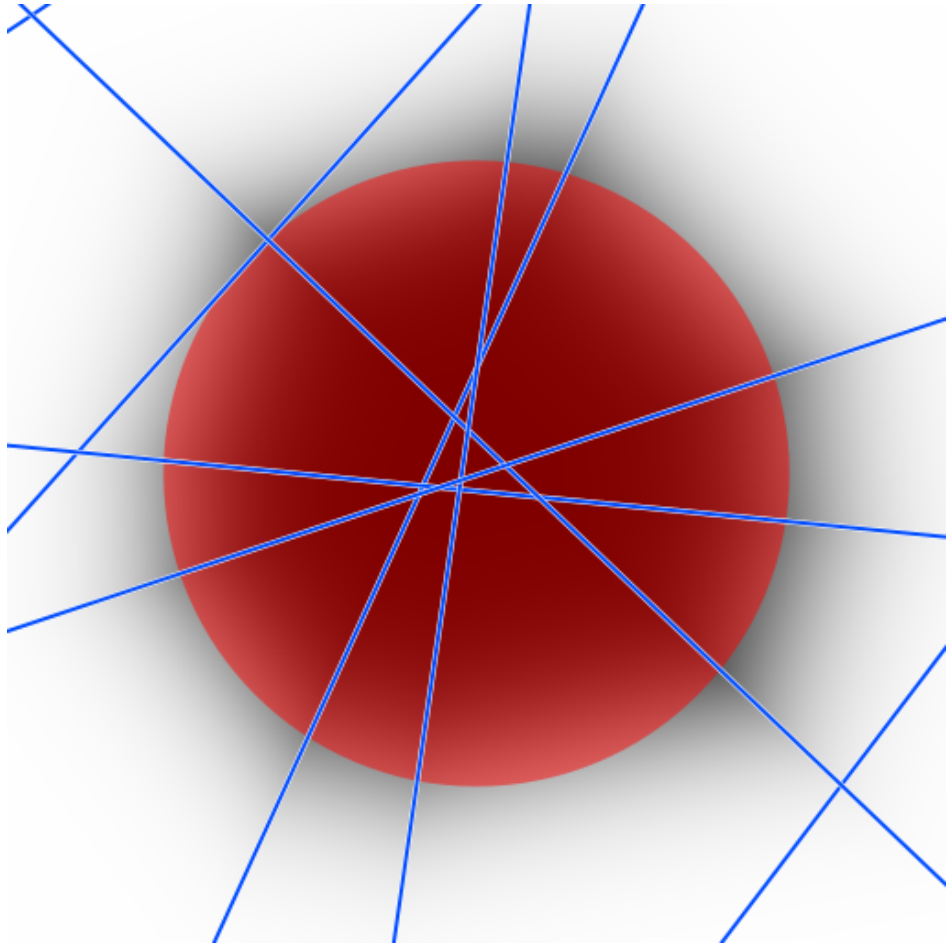
Iteration 4



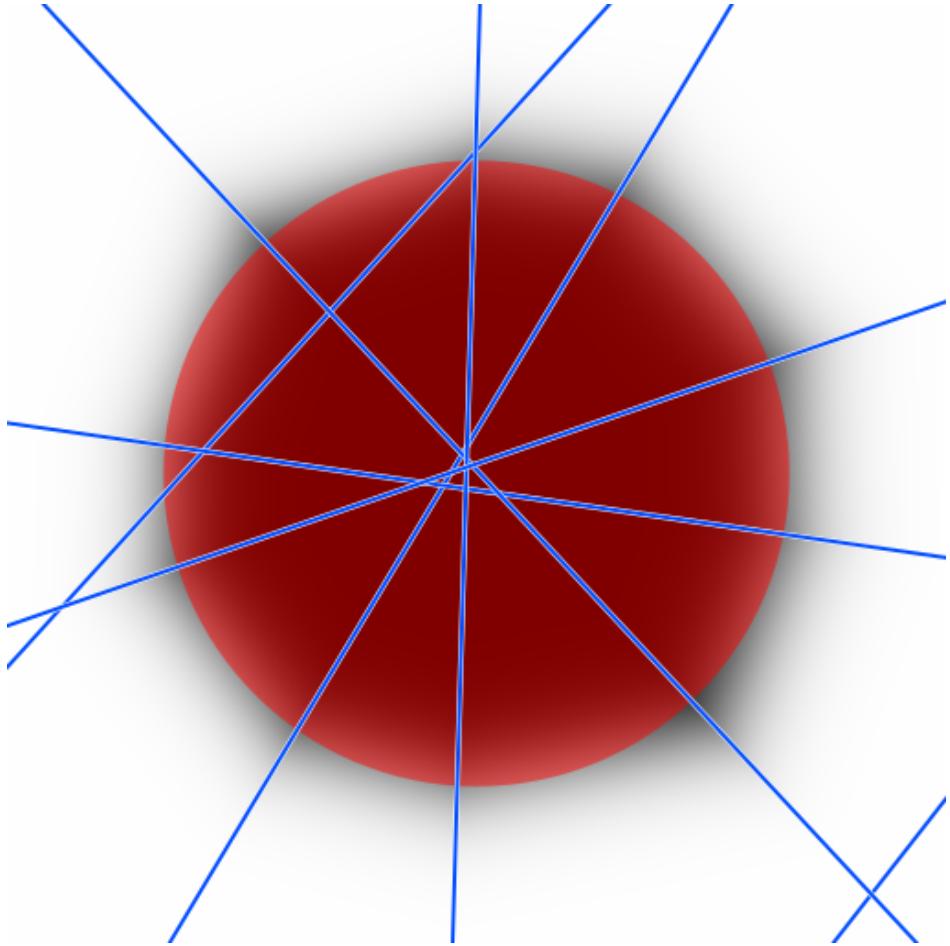
Iteration 7



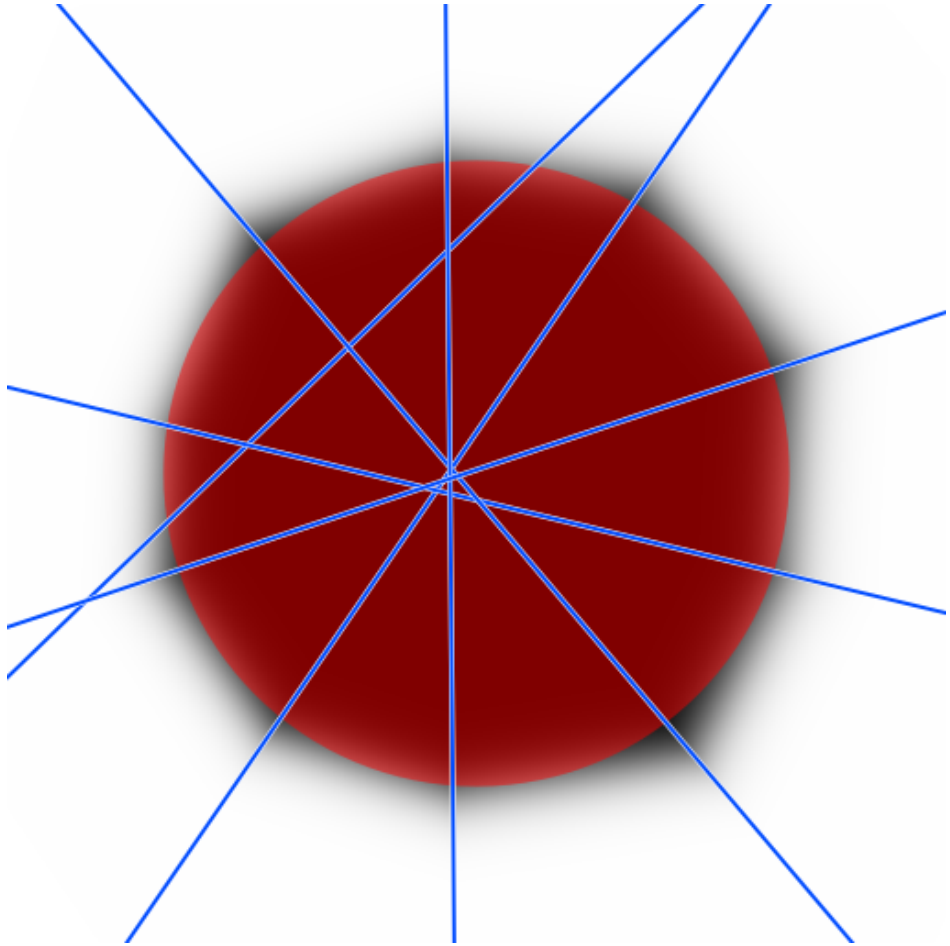
Iteration 10



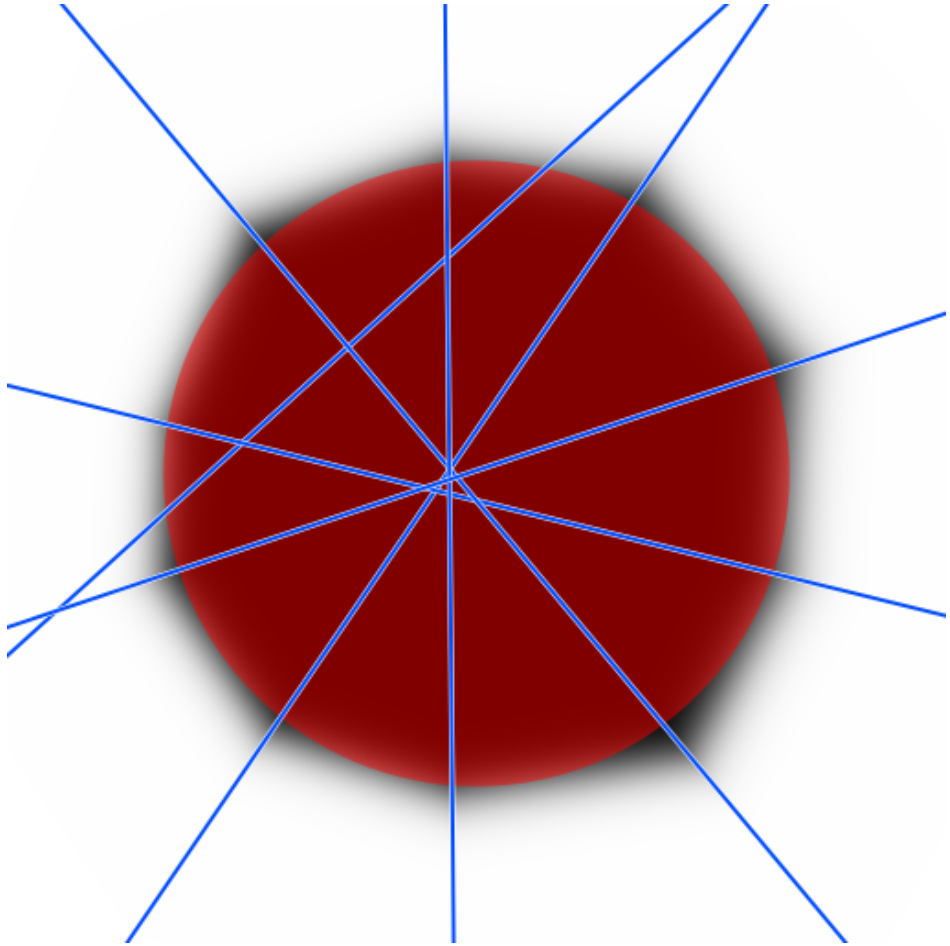
Iteration 16



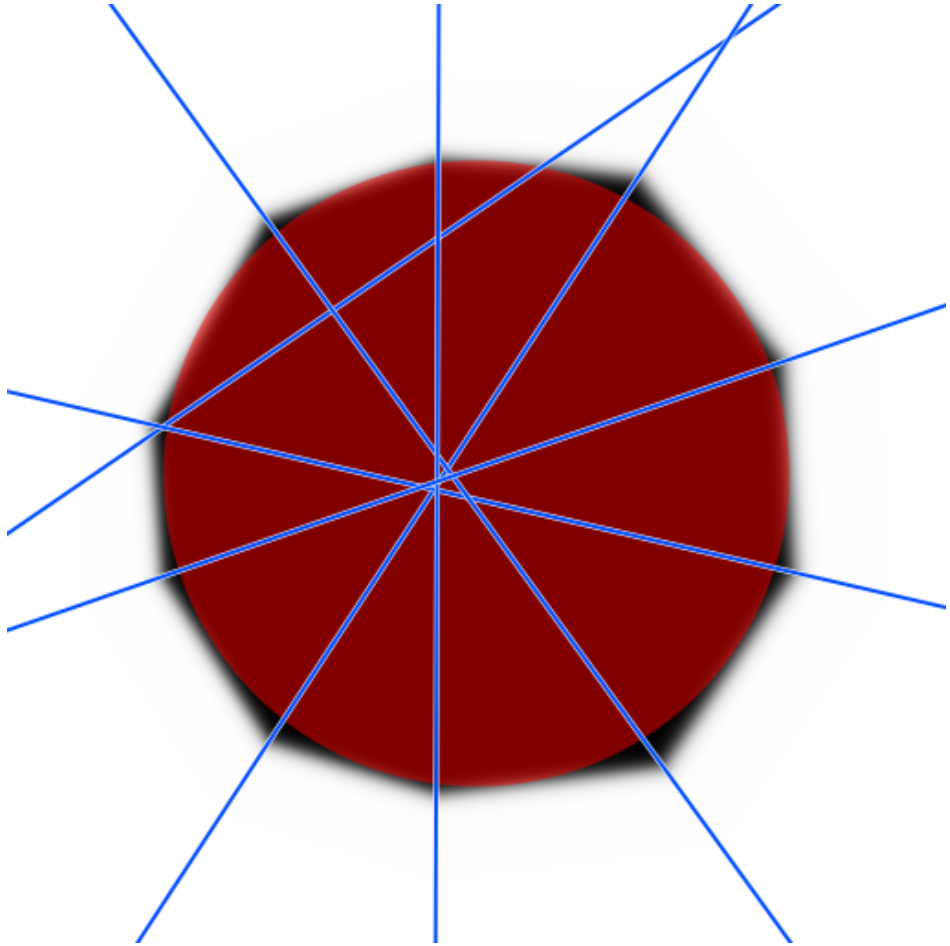
Iteration 34



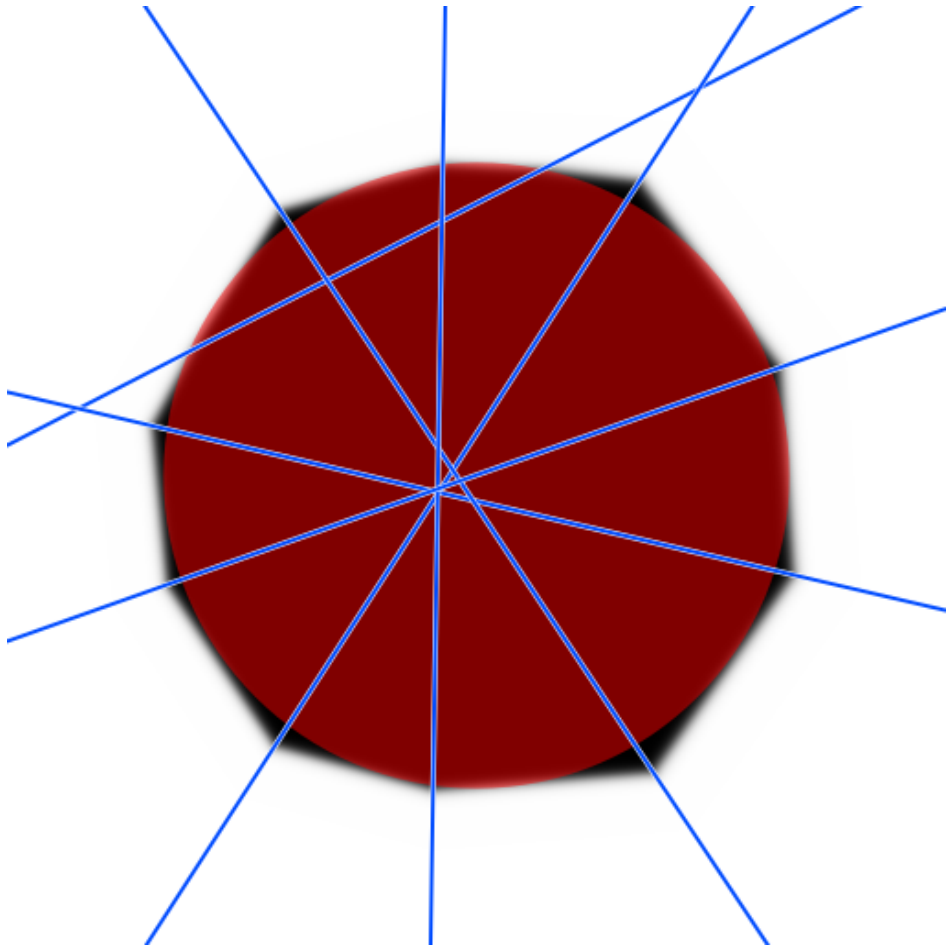
Iteration 77



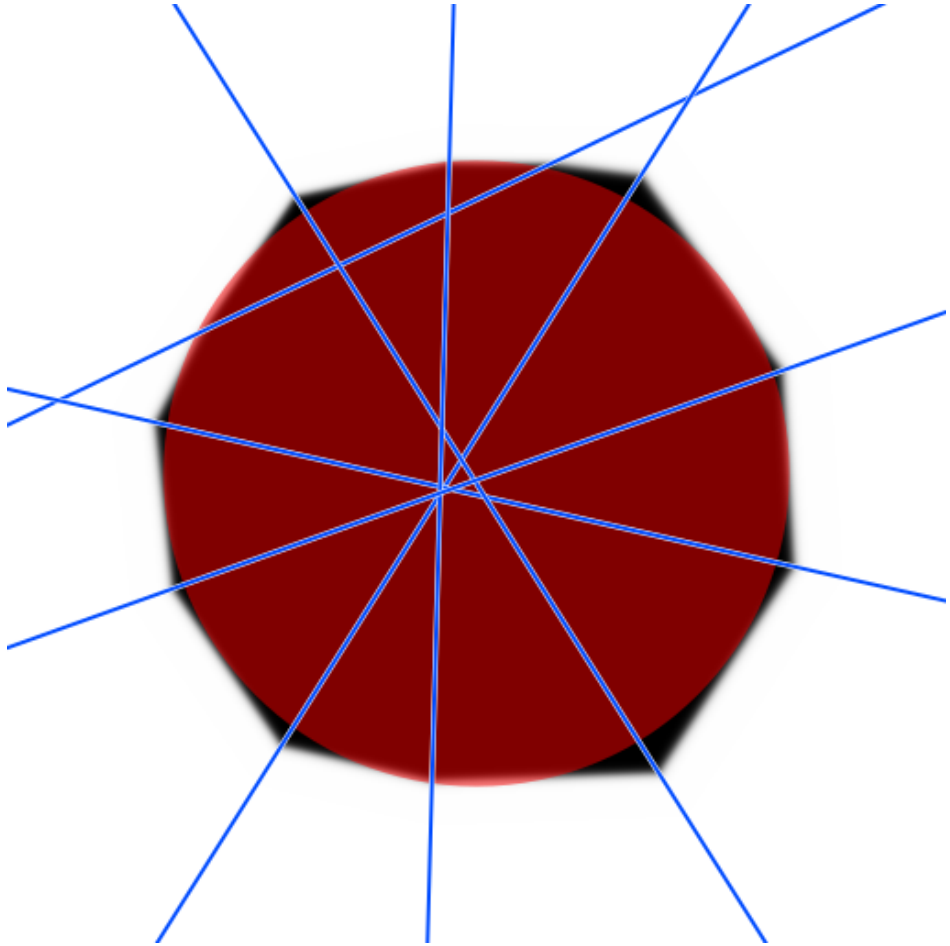
Iteration 100



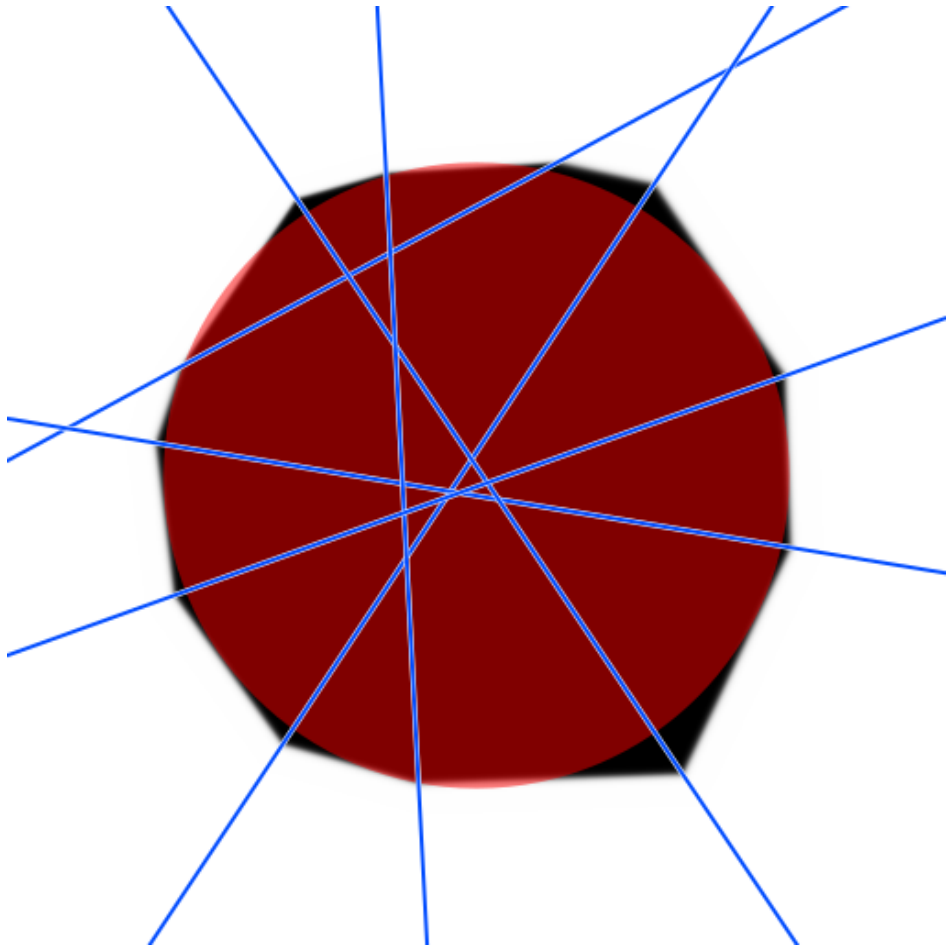
Iteration 703



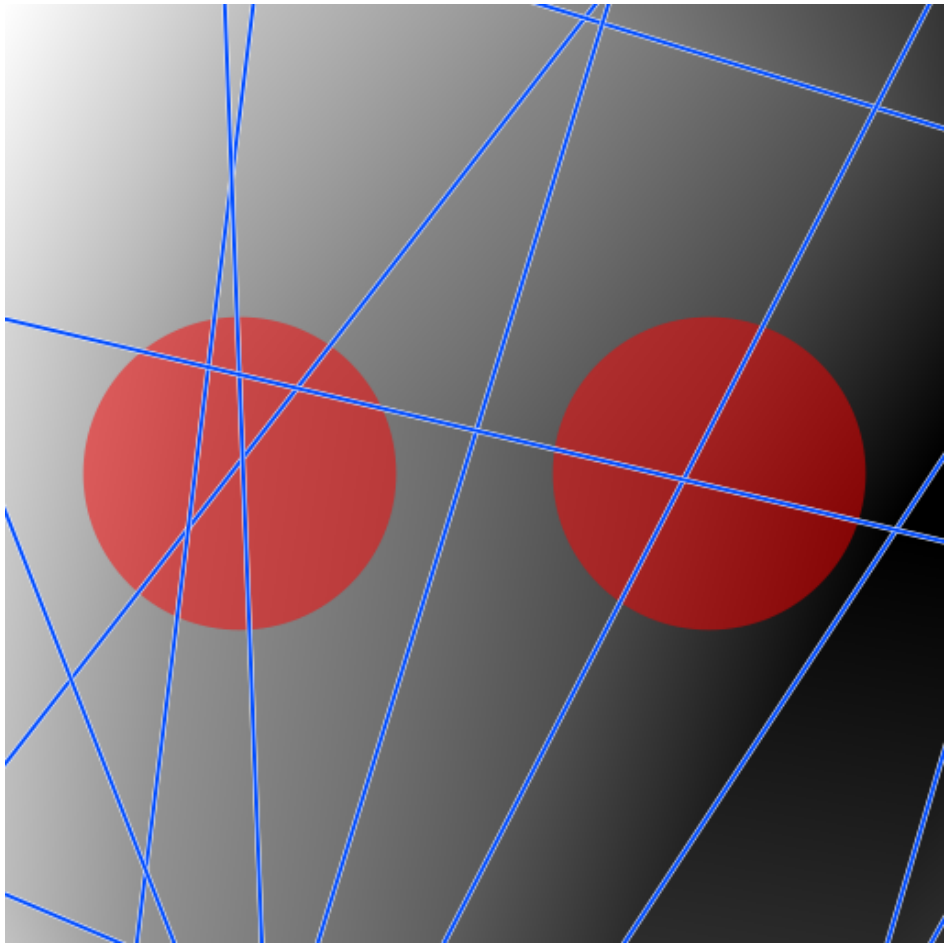
Iteration 1407



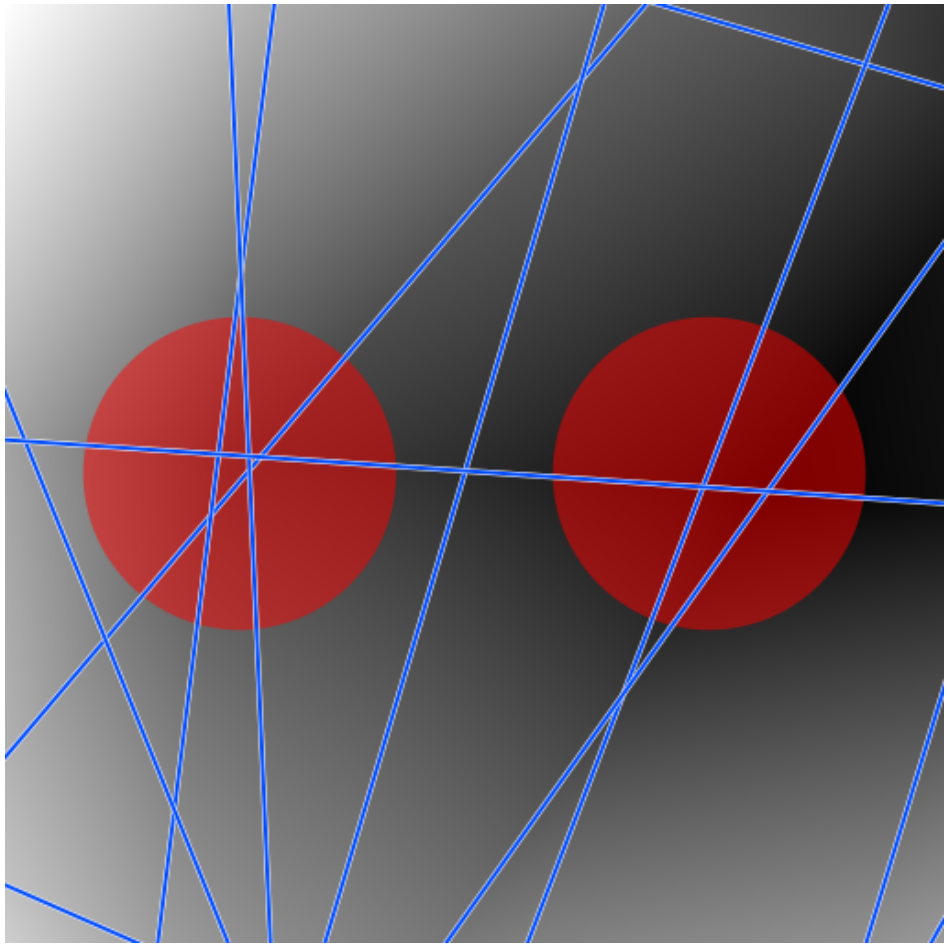
Iteration 2789



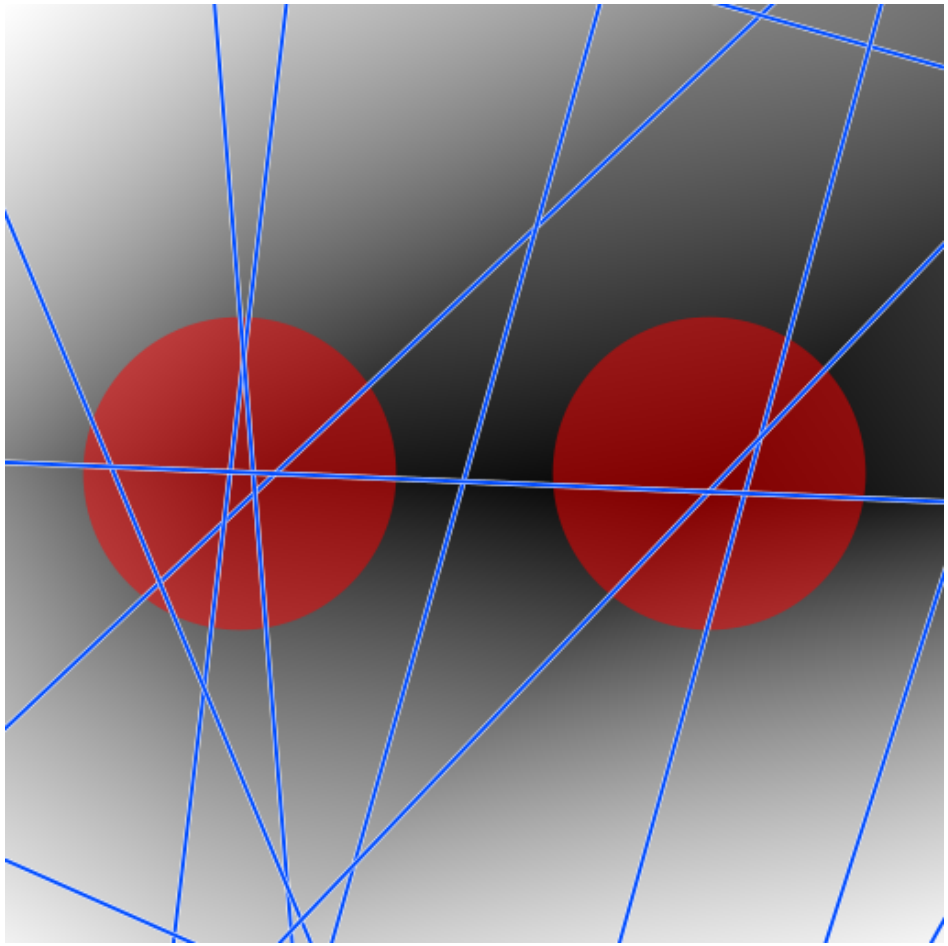
Iteration 9999



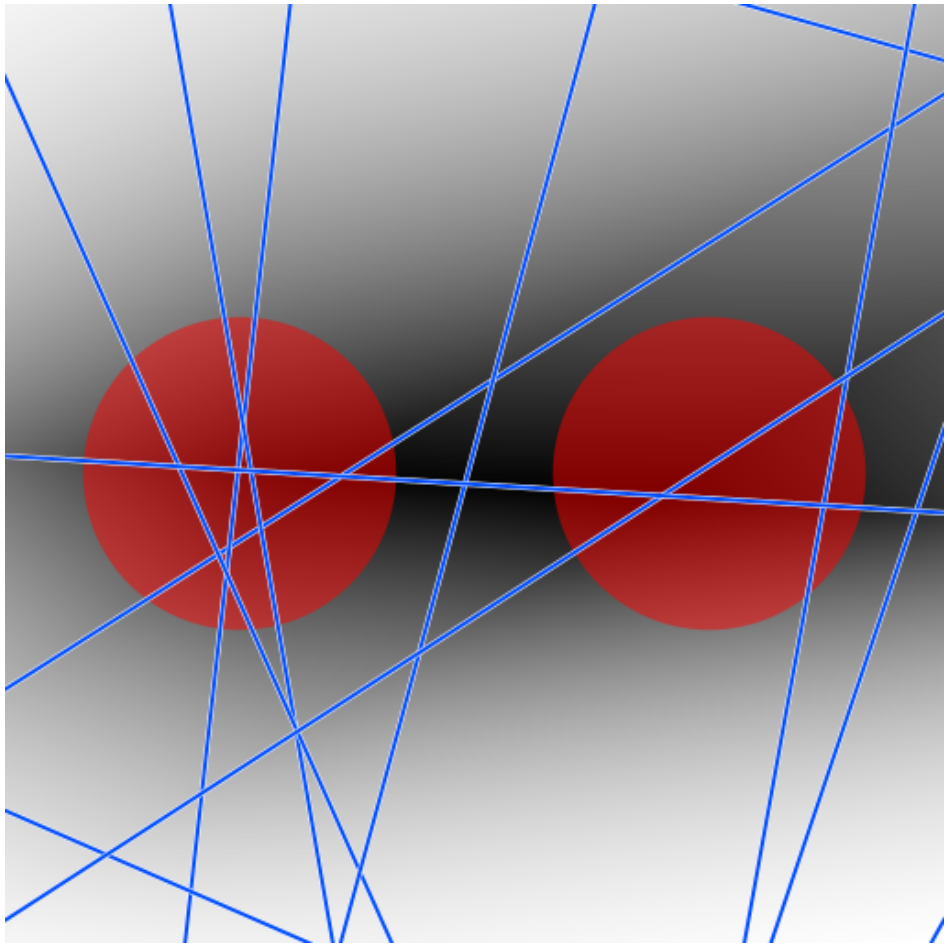
Iteration 1



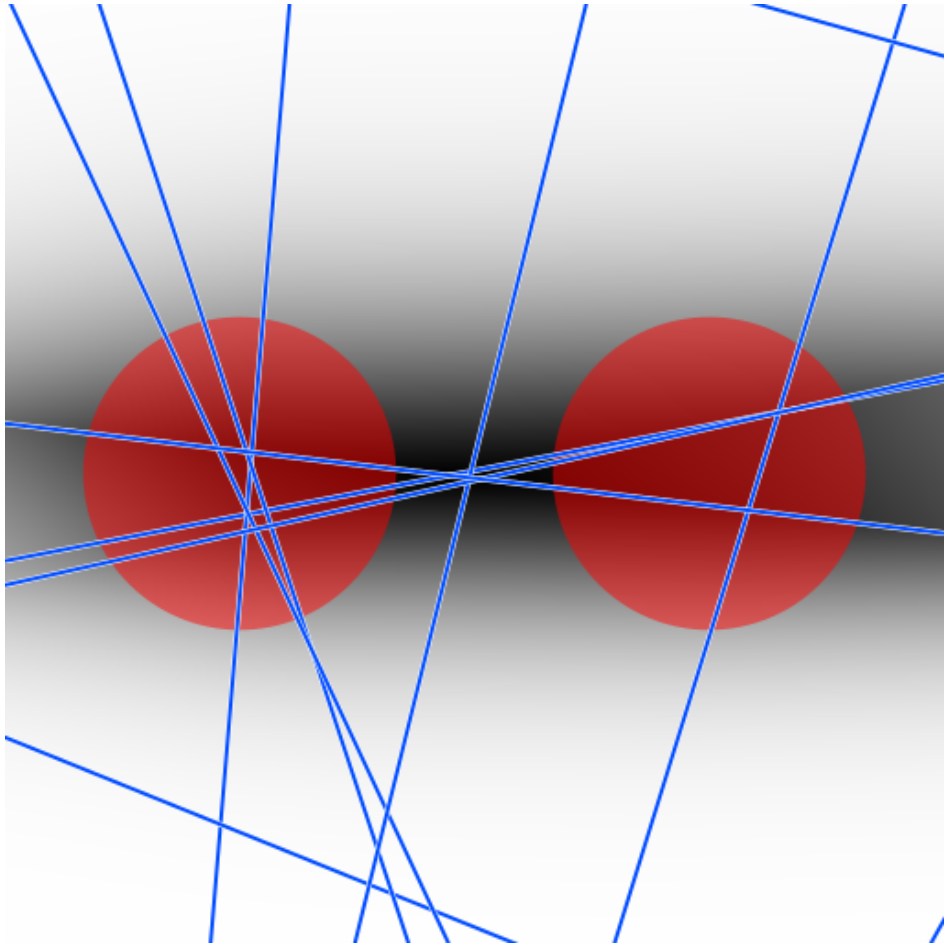
Iteration 4



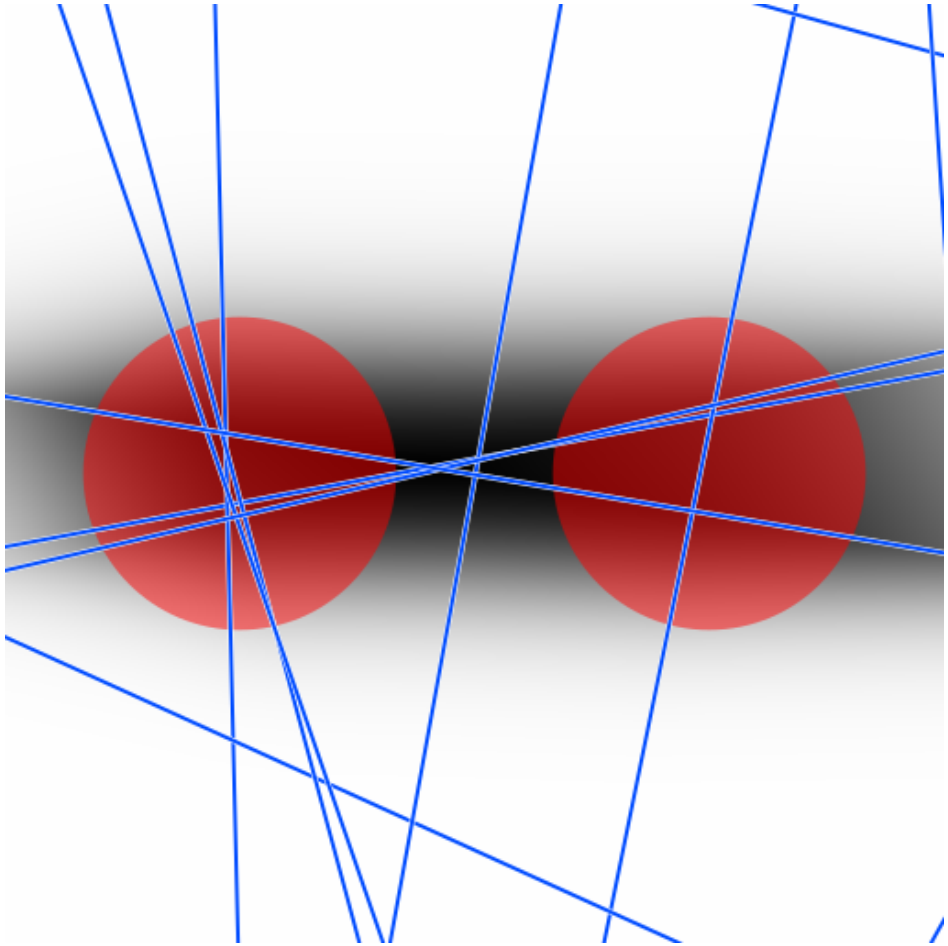
Iteration 7



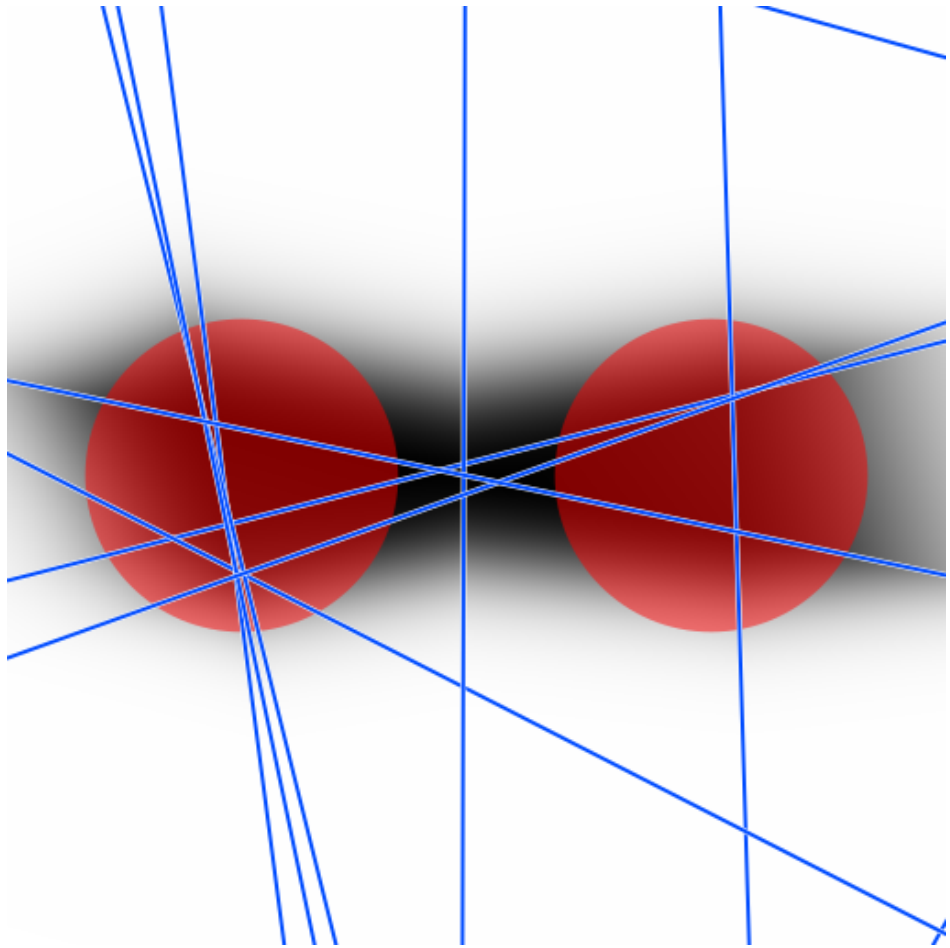
Iteration 10



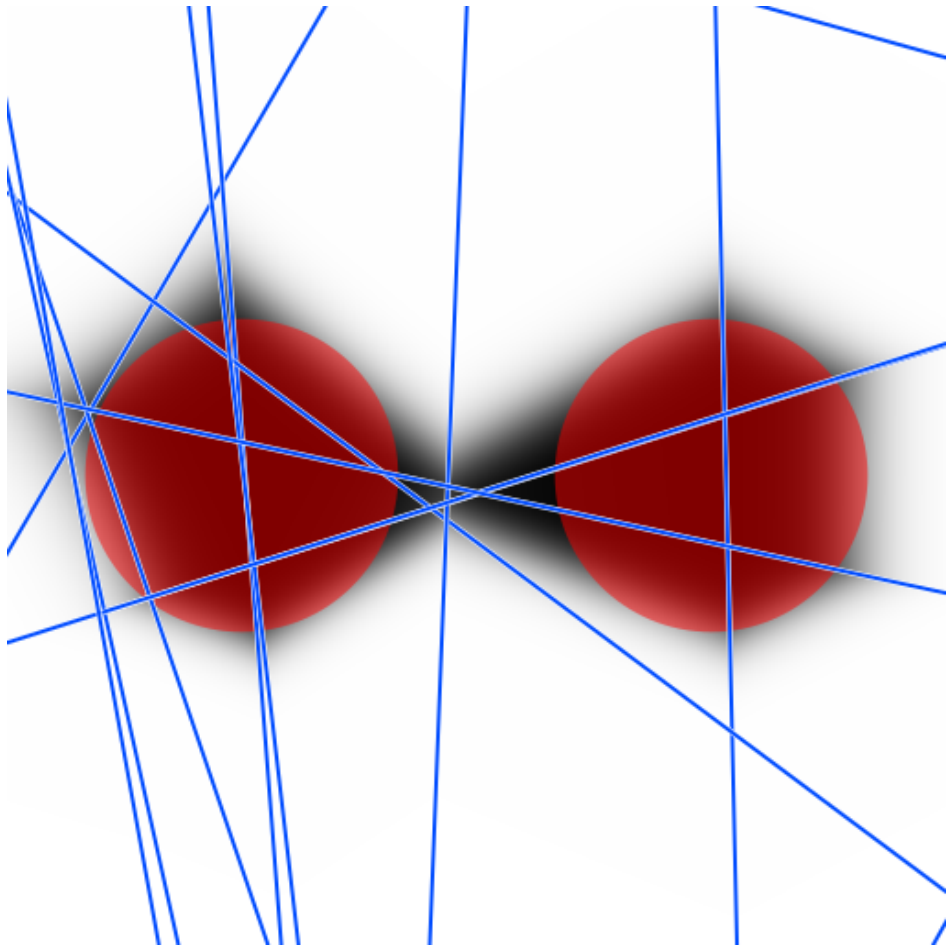
Iteration 16



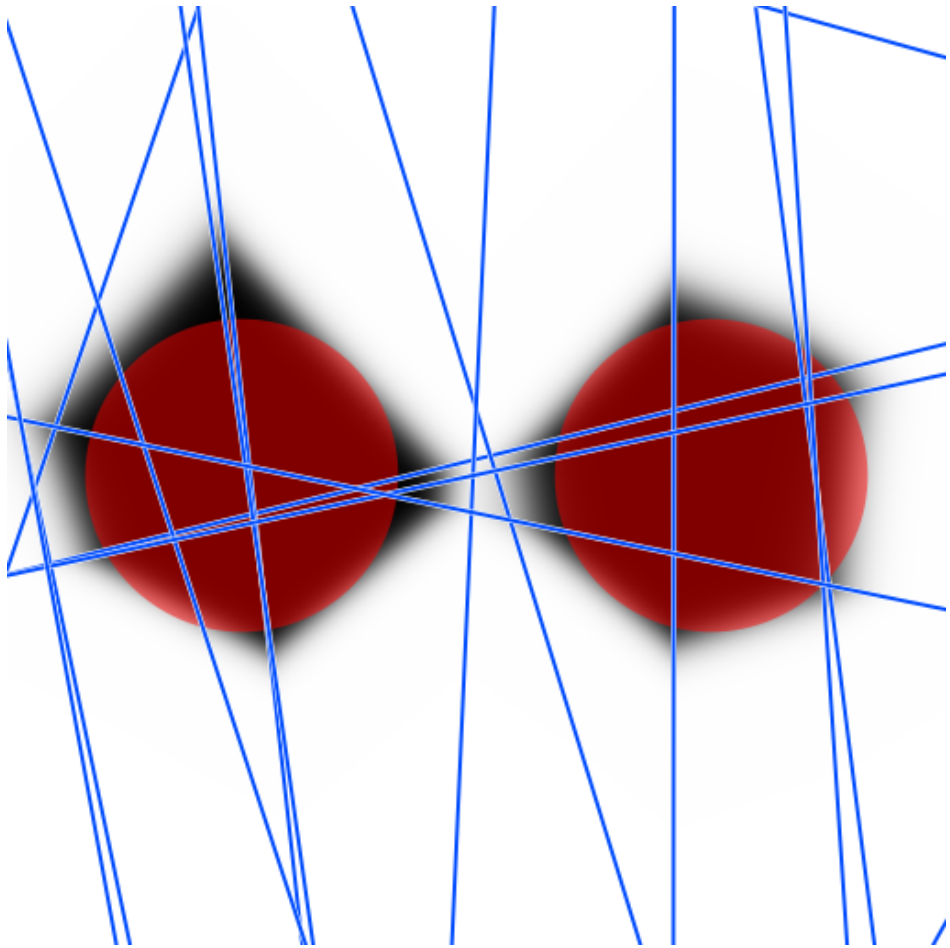
Iteration 34



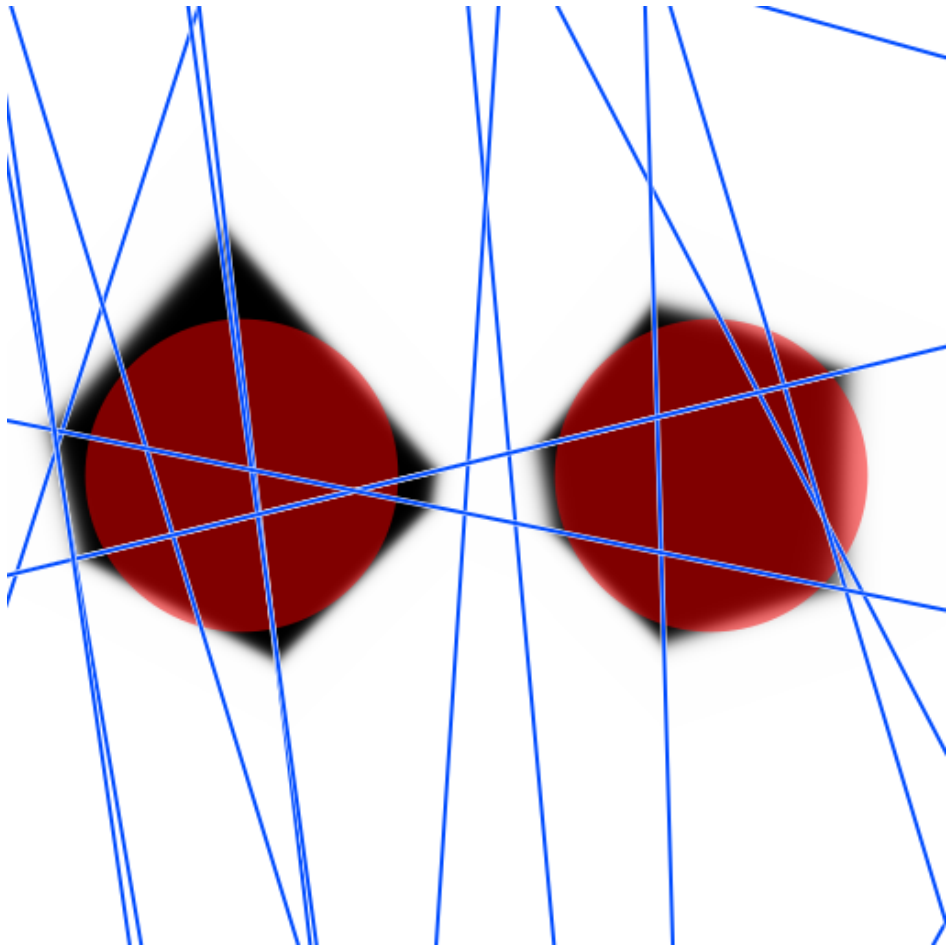
Iteration 100



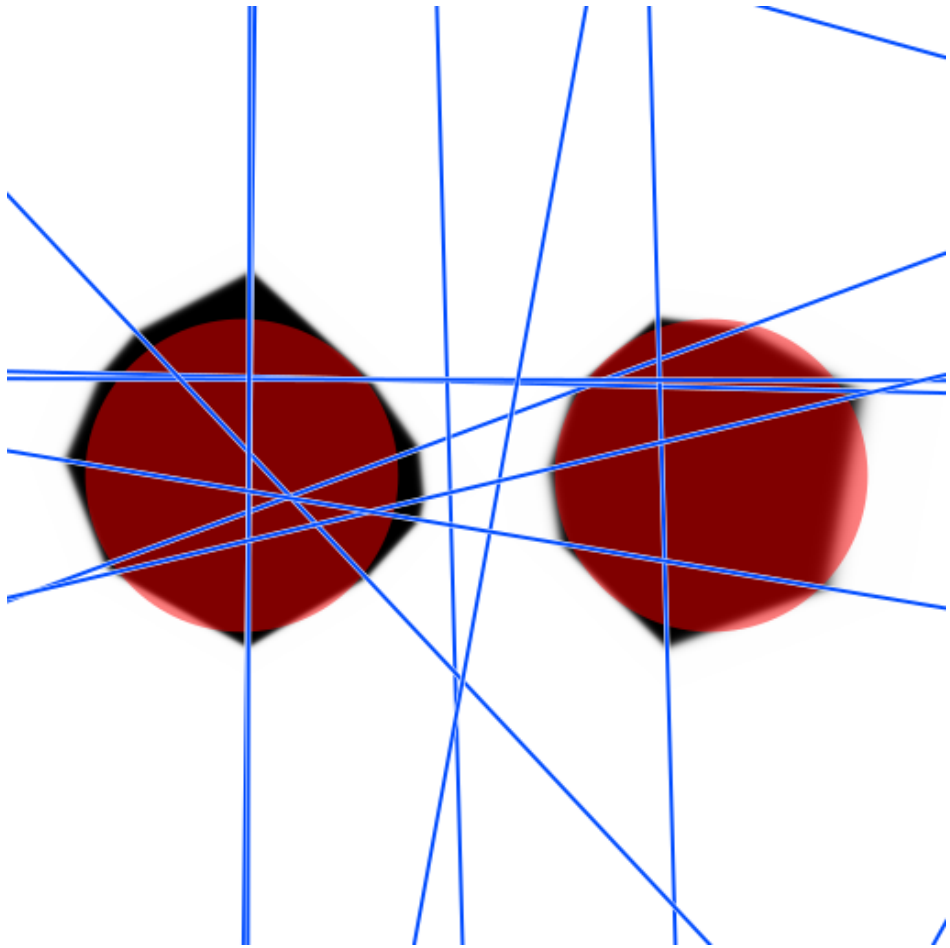
Iteration 272



Iteration 556



Iteration 2222



Iteration 9999

Convnet filters

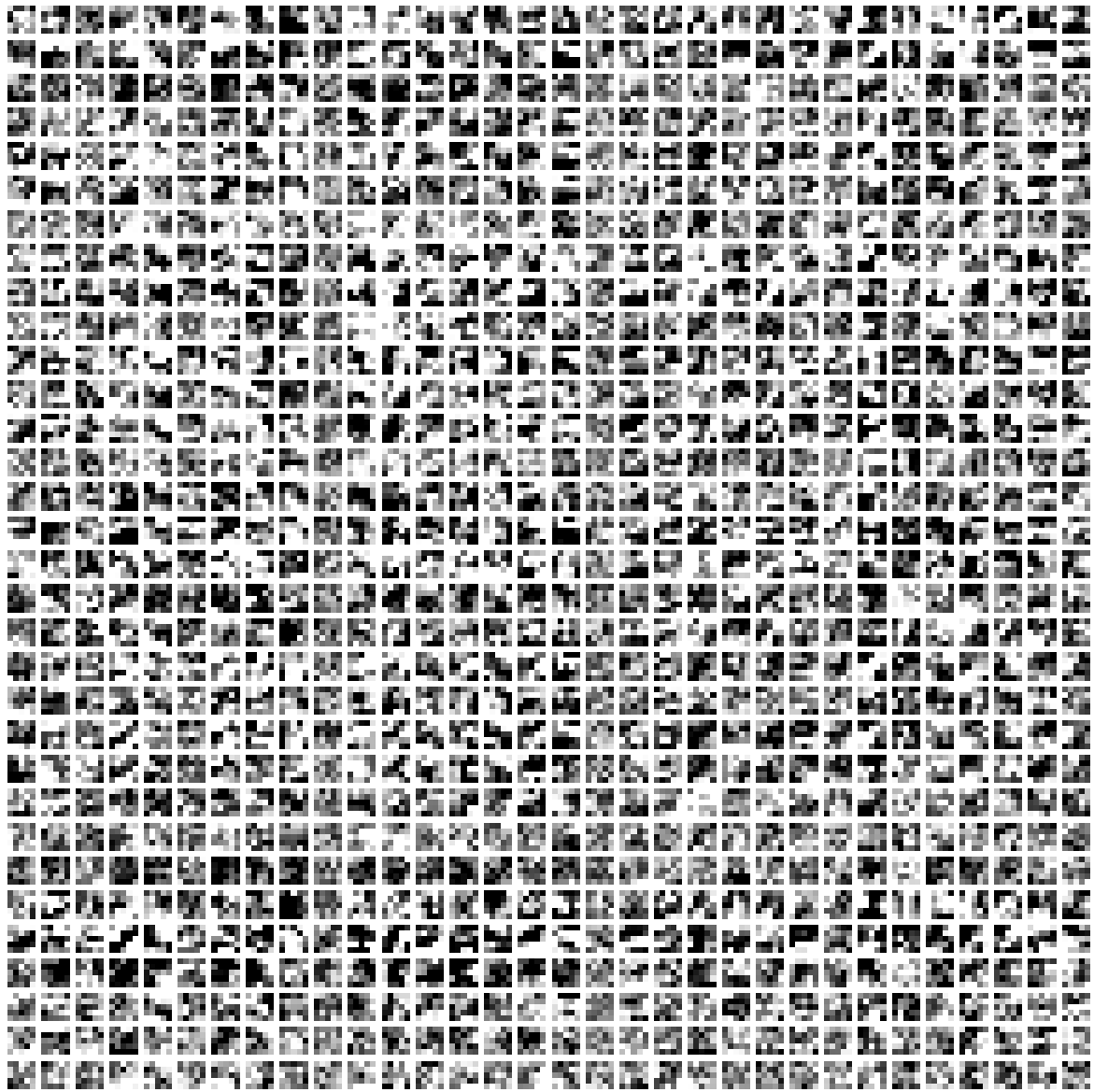
A similar analysis is complicated to conduct with real-life networks given the high dimension of the signal.

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

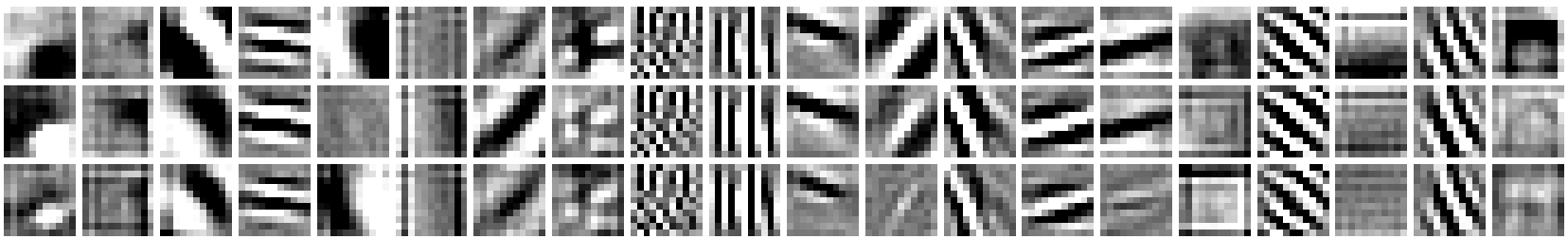
While it is quite reasonable in the first layer, since the filters are indeed consistent with the image input, it is far less so in the subsequent layers.

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

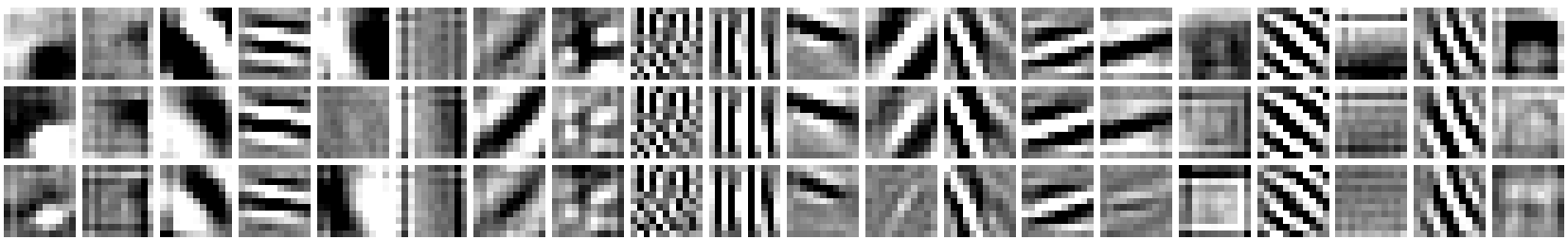
LeNet's second convolutional layer ($32 \rightarrow 64$), first 32 filters out of 64



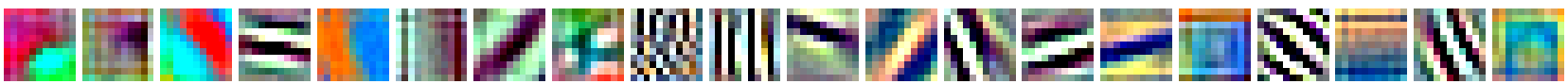
AlexNet's first convolutional layer ($3 \to 64$), first 20 filters out of 64



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



or as RGB images



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

The end