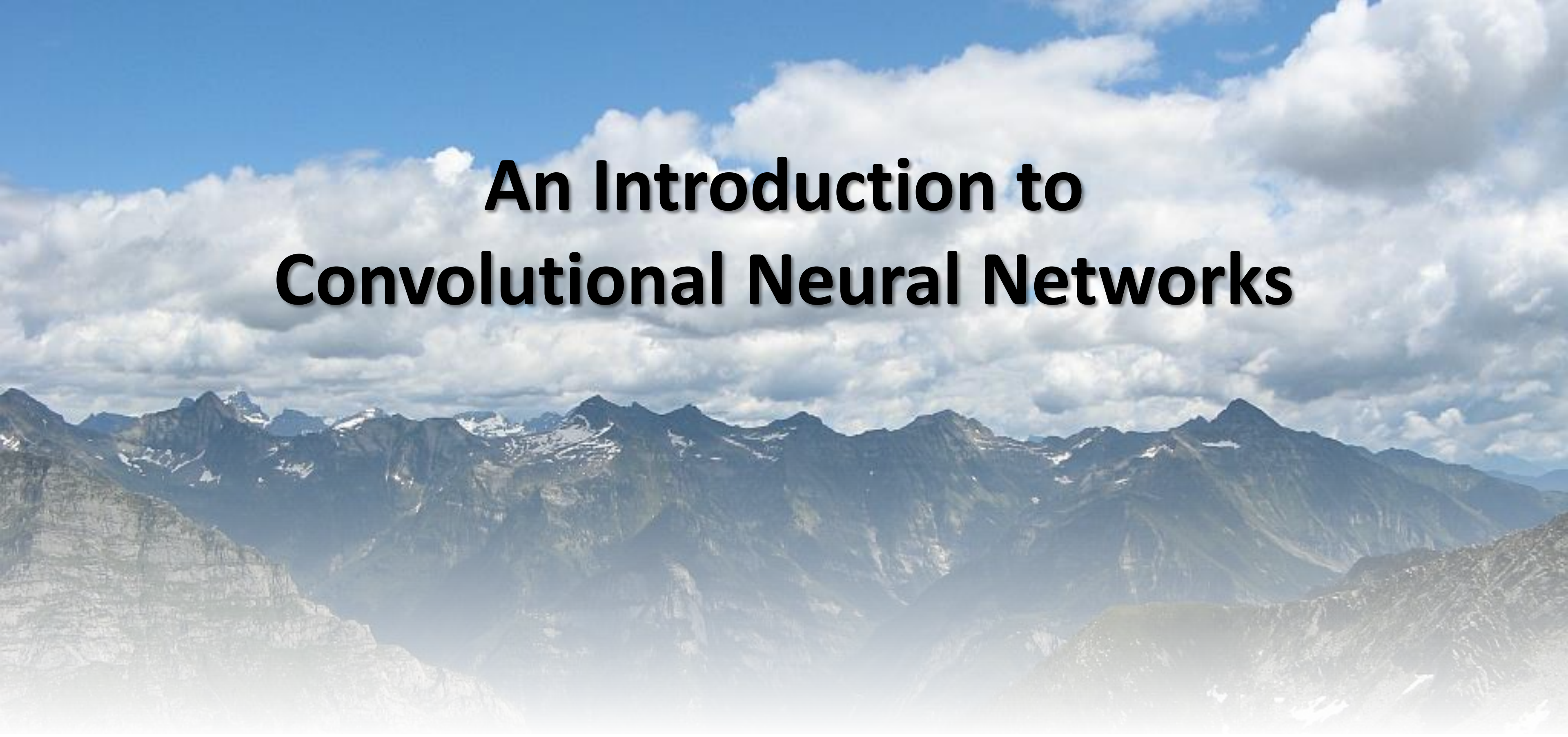


An Introduction to Convolutional Neural Networks

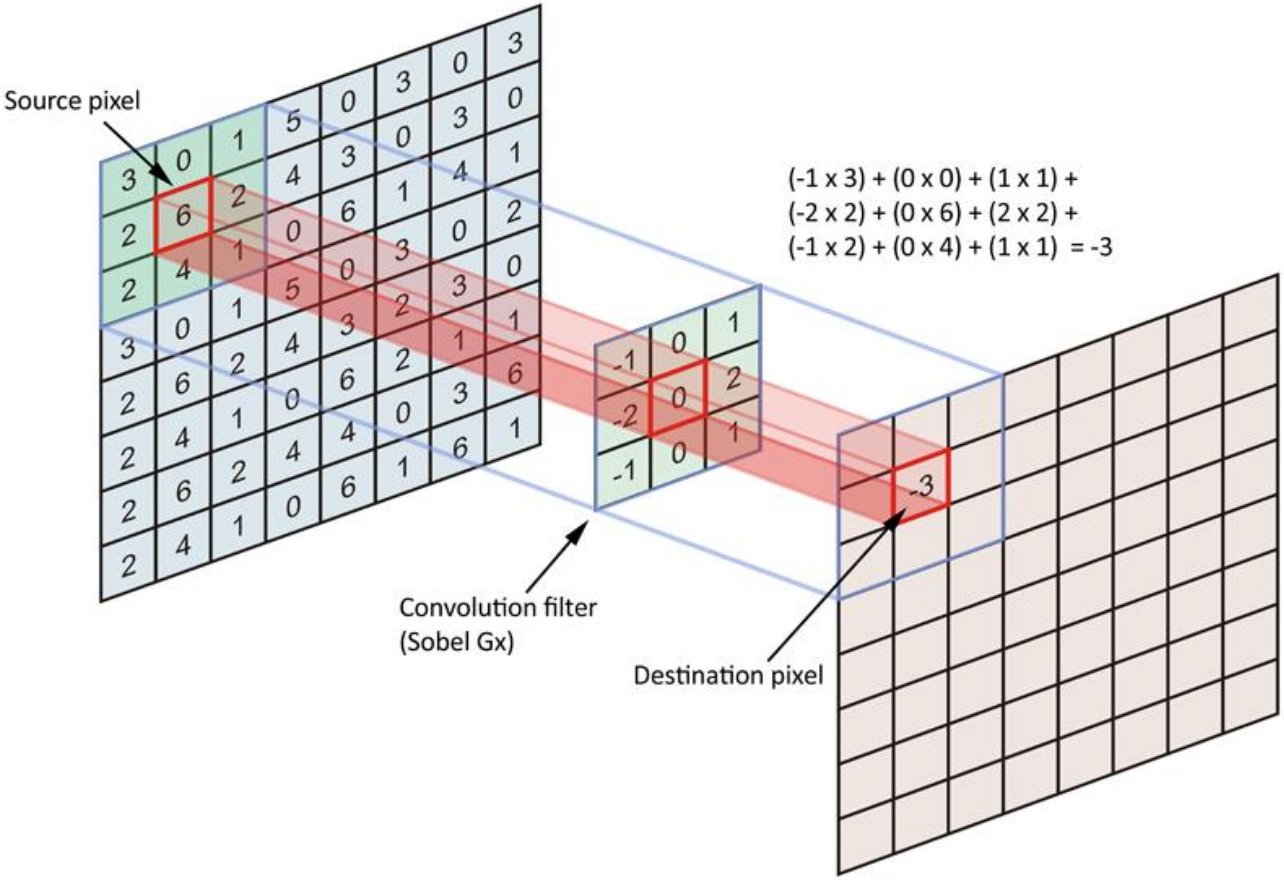
Alessandro Giusti
Dalle Molle Institute for Artificial Intelligence
Lugano, Switzerland

Sources & Resources

- Andrej Karpathy, CS231n
  http://cs231n.github.io/convolutional-networks/
- Ian Goodfellow et al., Deep Learning
  http://www.deeplearningbook.org/
- F. Chollet, Deep Learning with Python
  https://www.manning.com/books/deep-learning-with-python
- Jonathan Hui, CNN Tutorial
  https://jhui.github.io/2017/03/16/CNN-Convolutional-neural-network/
- Tim Demetters, Understanding Convolutions
  http://timdettmers.com/2015/03/26/convolution-deep-learning/

Some images in this presentation are extracted from the sources listed above

What is a convolution?

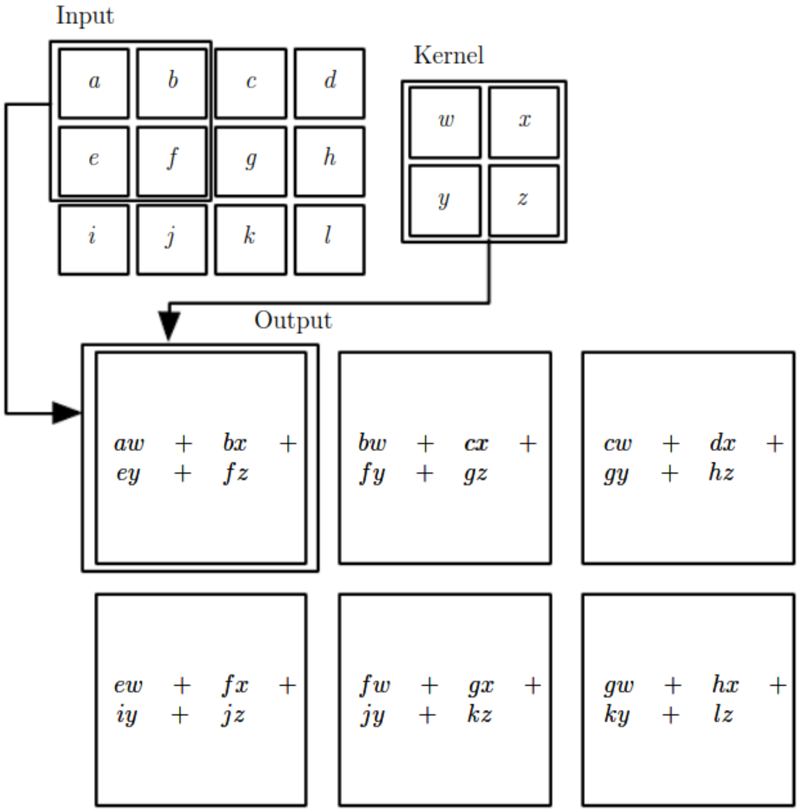


3	0	1	5	0	3	0	3
2	6	2	4	3	0	3	0
2	4	1	0	6	1	4	1
3	0	1	5	0	3	0	2
2	6	2	4	3	2	3	0
2	4	1	0	6	2	1	1
2	6	2	4	4	0	3	6
2	4	1	0	6	1	6	1

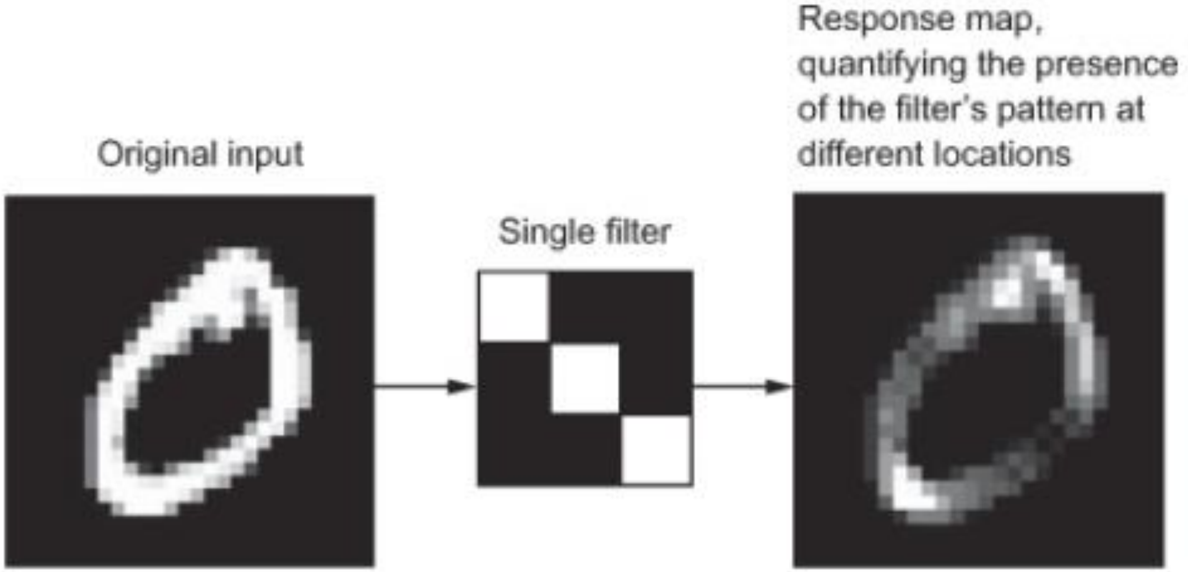
-1	0	1
-2	0	2
-1	0	1

(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3

What is a convolution?



How does a convolution look like?

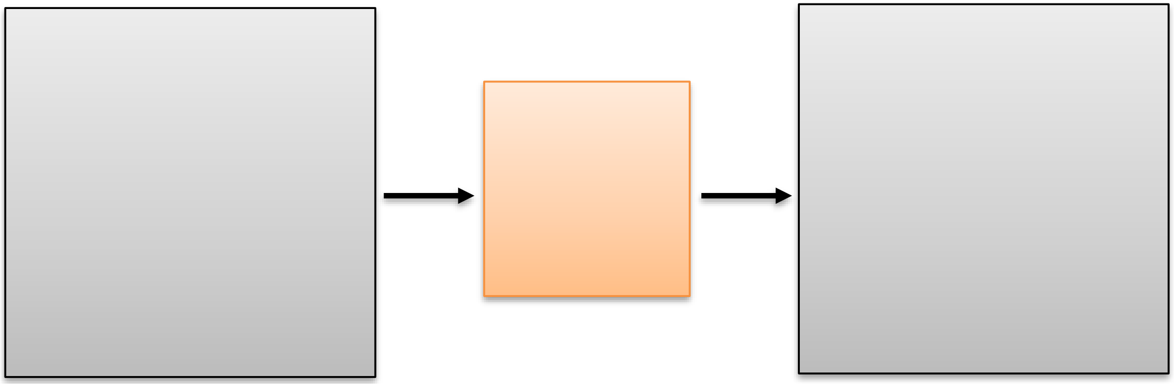

Original input
Single filter
Response map, quantifying the presence of the filter's pattern at different locations

1 input map

1 3x3 kernel

1 output map

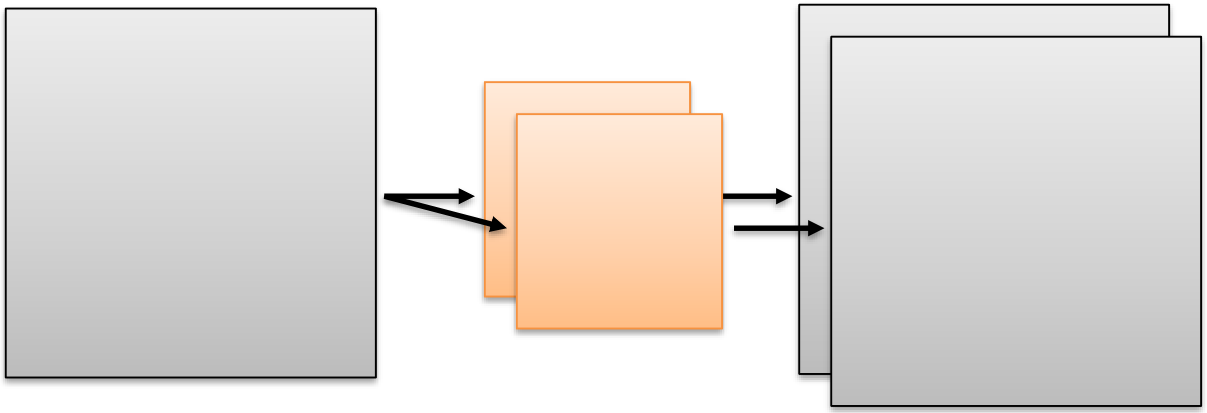
What about multiple maps?



1 input map

1 3x3 kernel

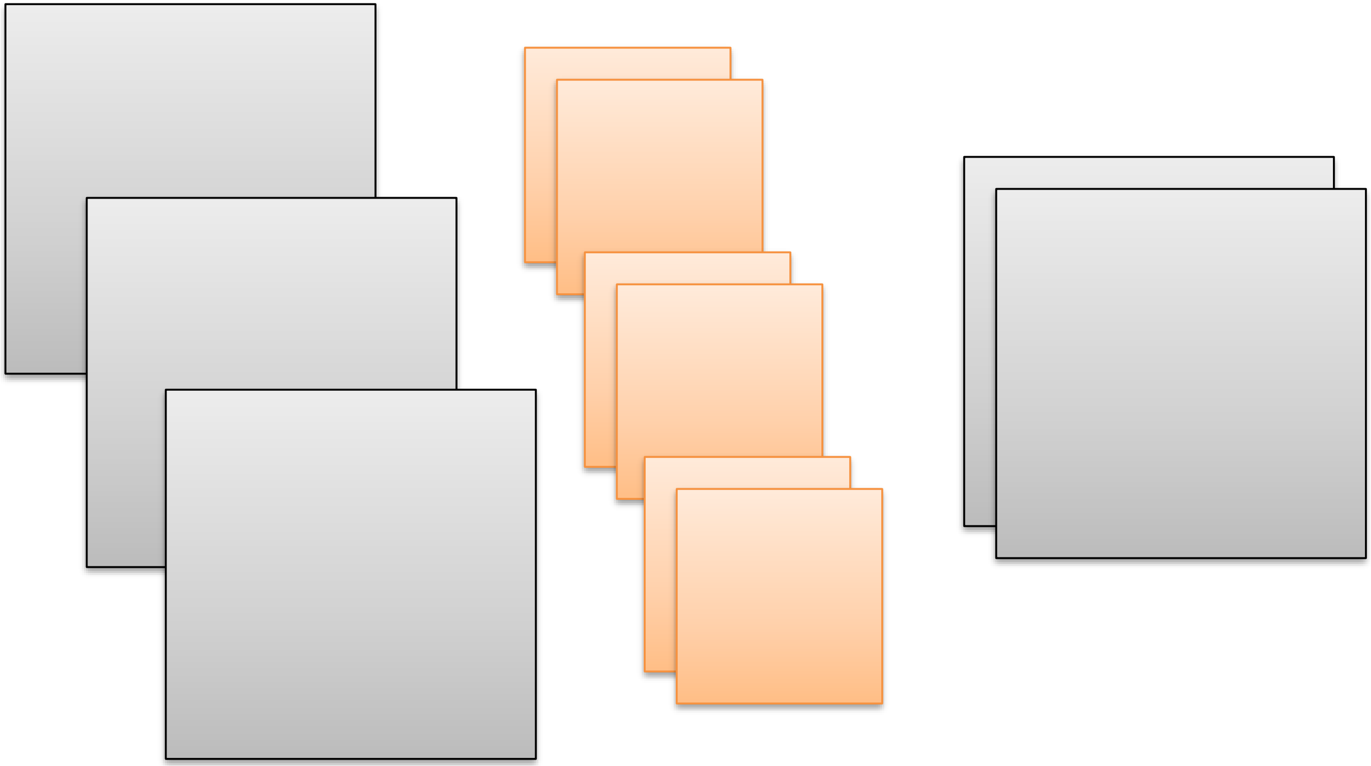
1 output map



1 input map

2 3x3 kernels

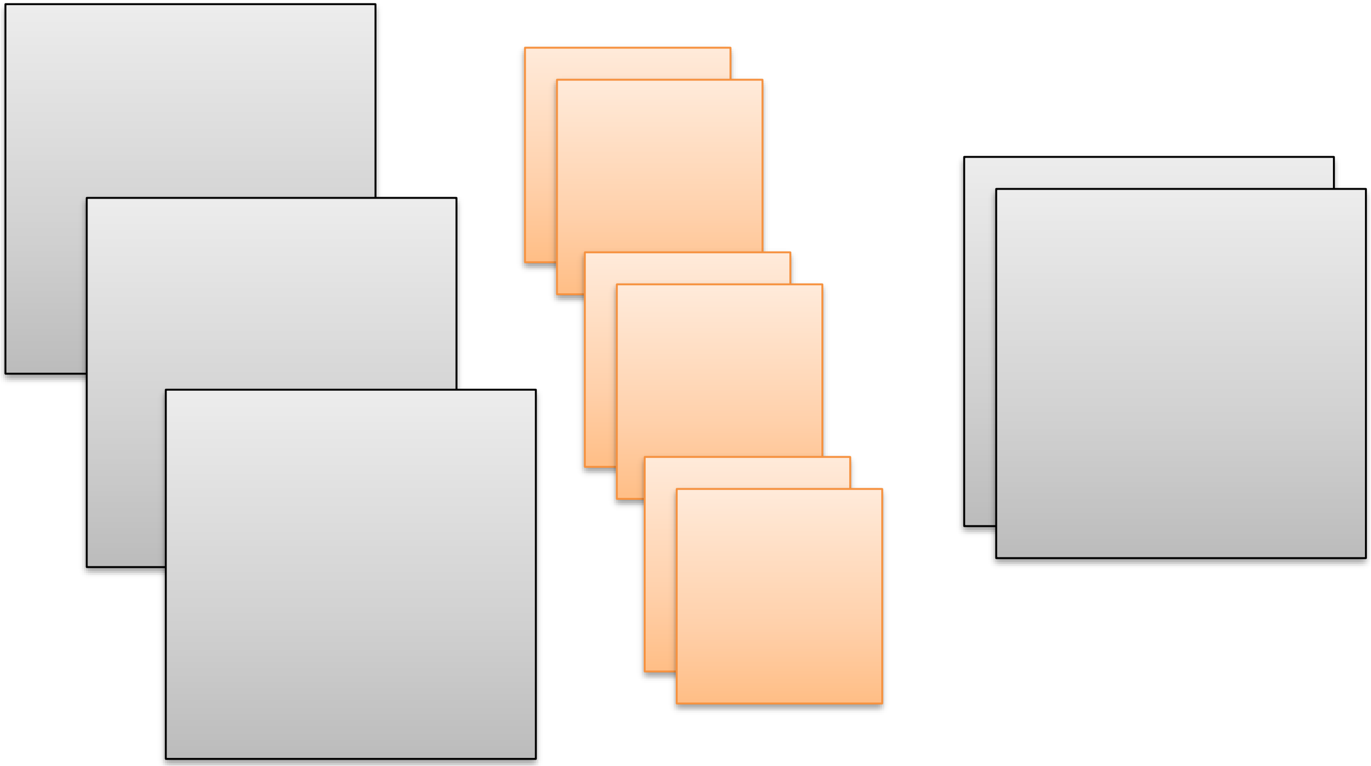
2 output maps



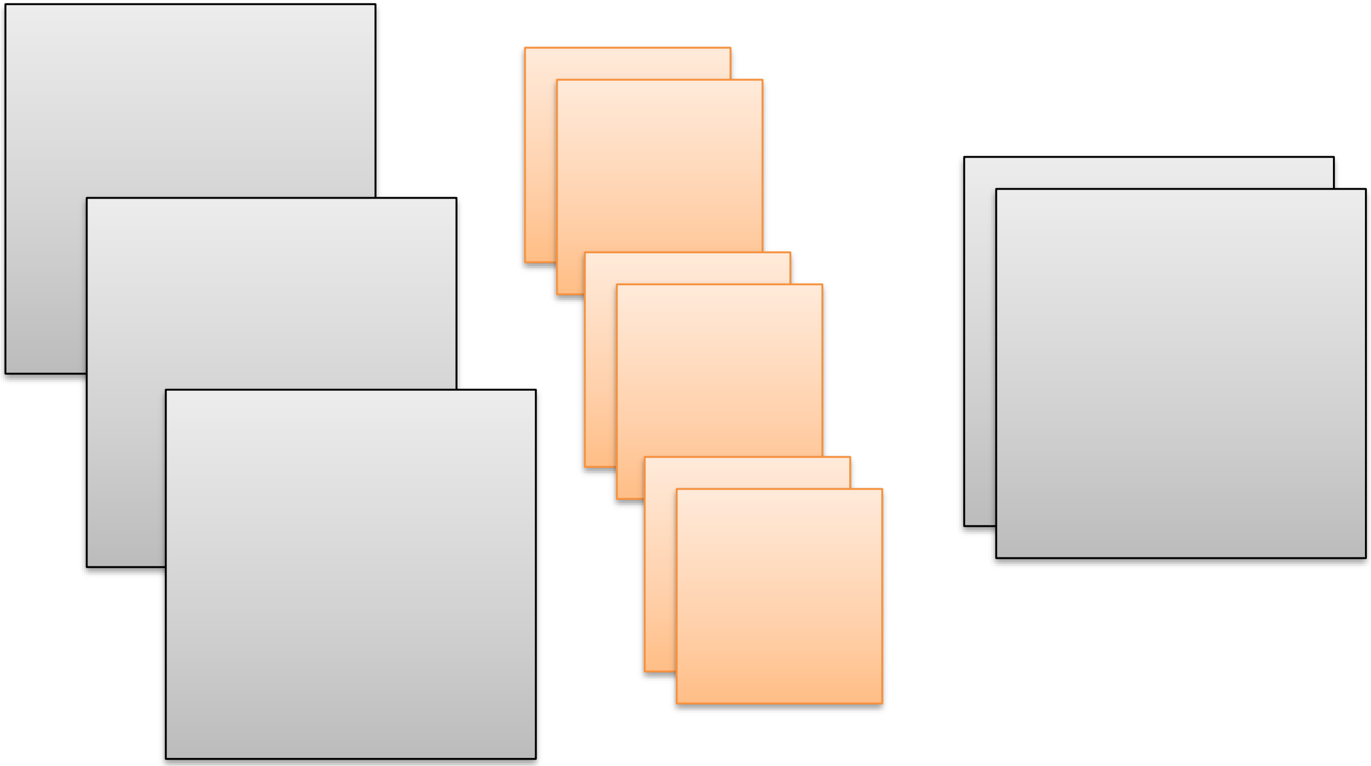
3 input maps

3x2 3x3 kernels

2 output maps



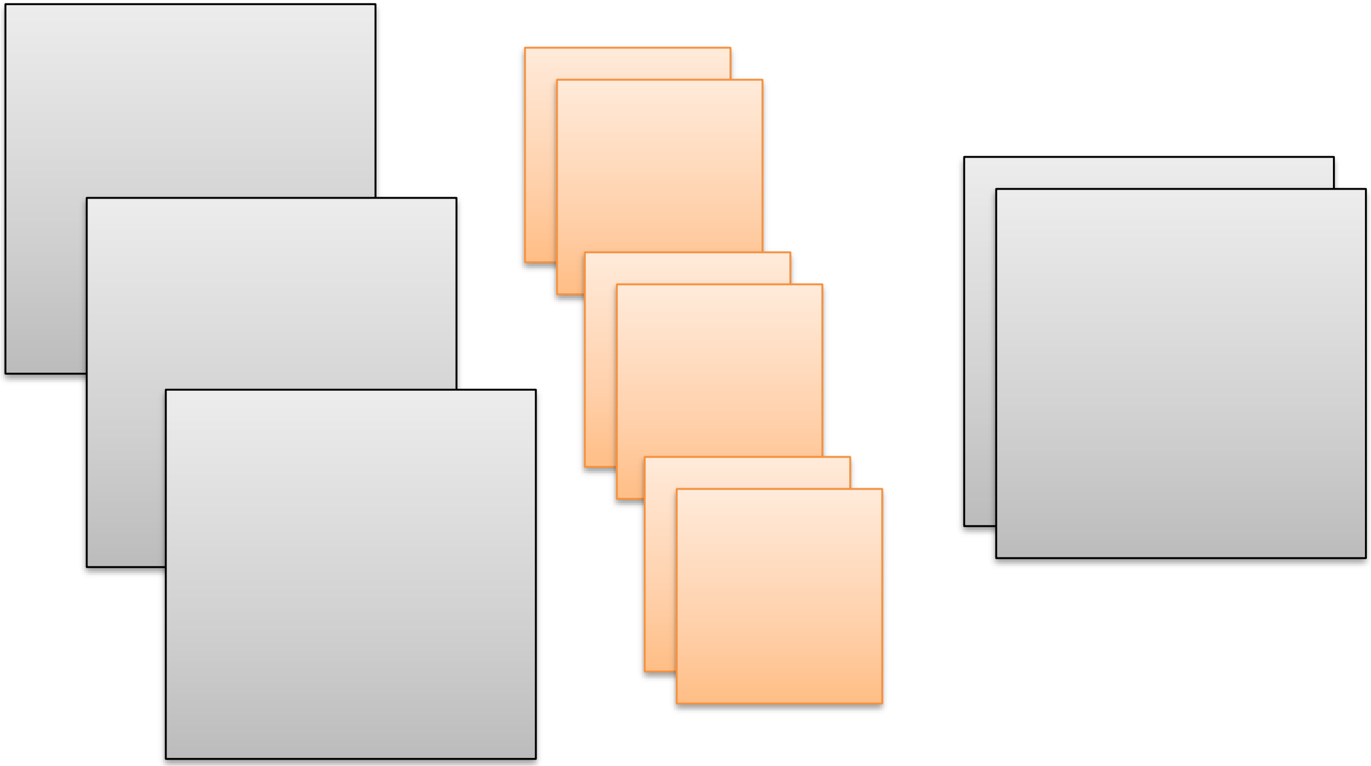
3 input maps 3x2 3x3 kernels 2 output maps



3 input maps

3x2 3x3 kernels

2 output maps

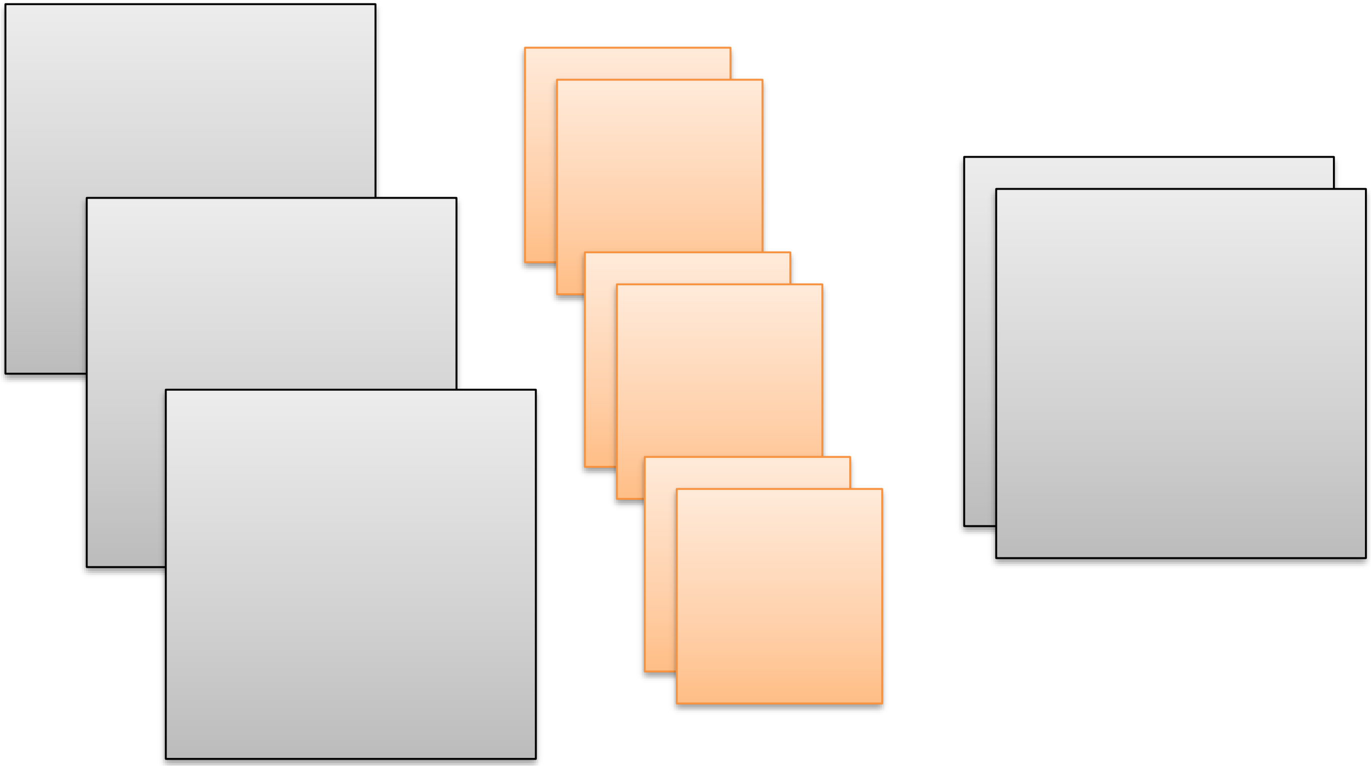


3 input maps

3x2 3x3 kernels

2 output maps

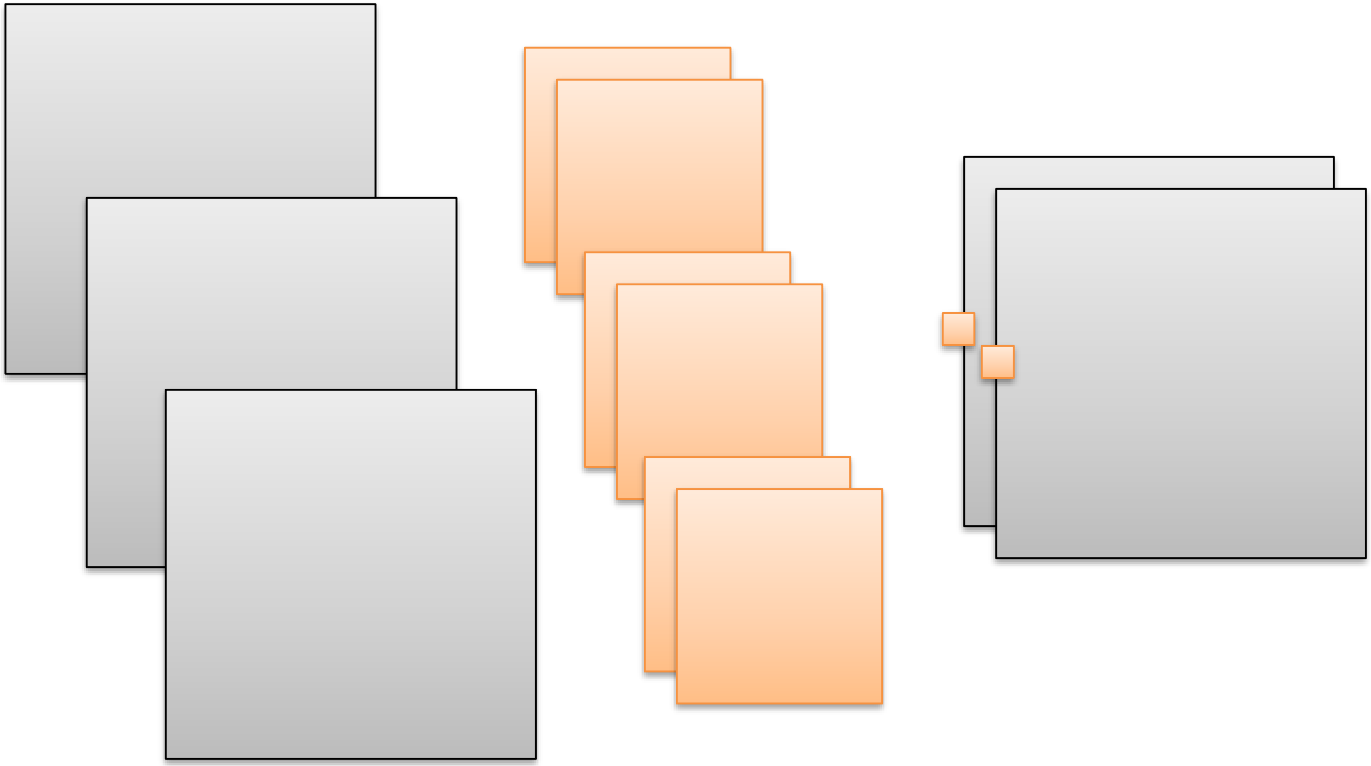
Quiz: how many parameters does this layer have?



3 input maps

3x2 3x3 kernels
= 54 ...

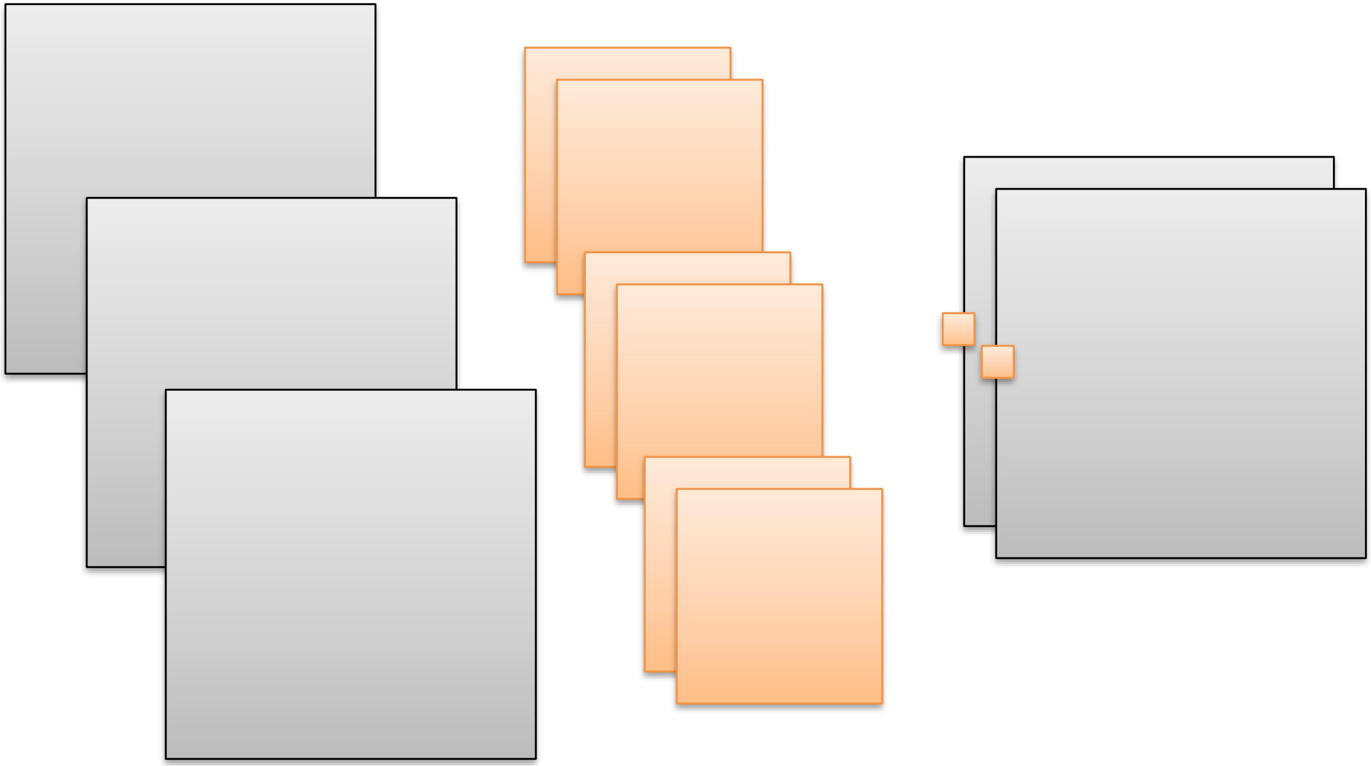
2 output maps



3 input maps

3x2 3x3 kernels
= 54 ...

2 output maps
+ 2 biases



3 input maps

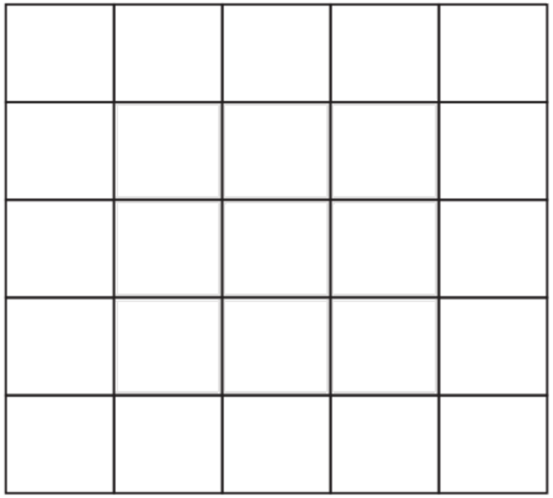
3x2 3x3 kernels

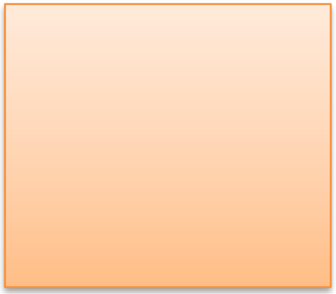
2 output maps

= 54 ... + 2 biases

= 56 trainable parameters (weights)

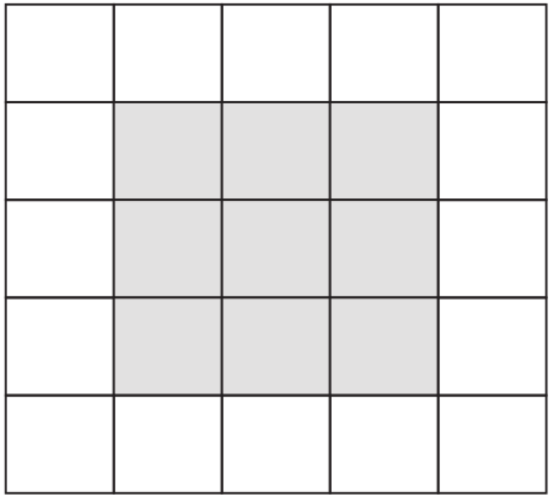
Details: padding

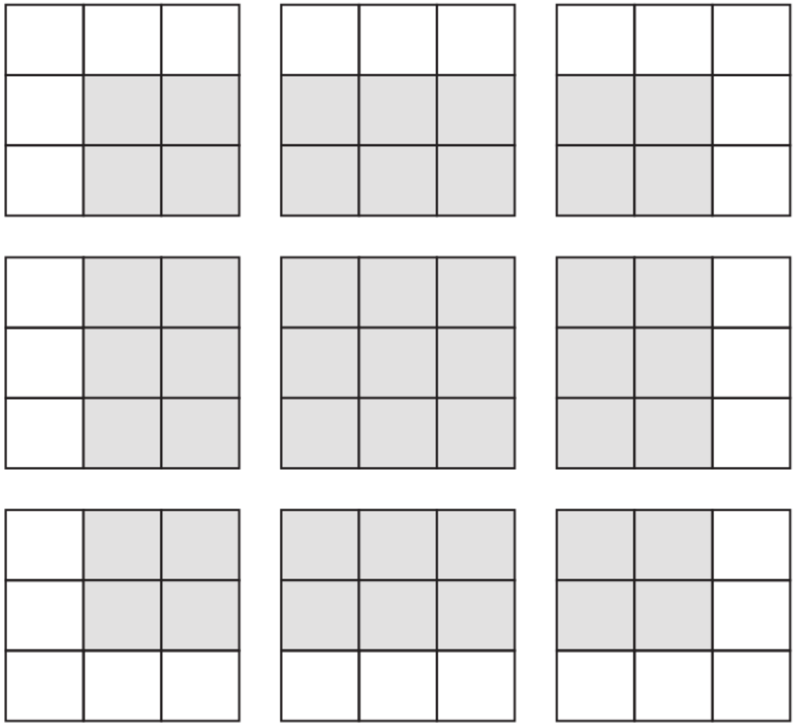




How many 3x3 patches are fully contained in a 5x5 map?

Details: padding

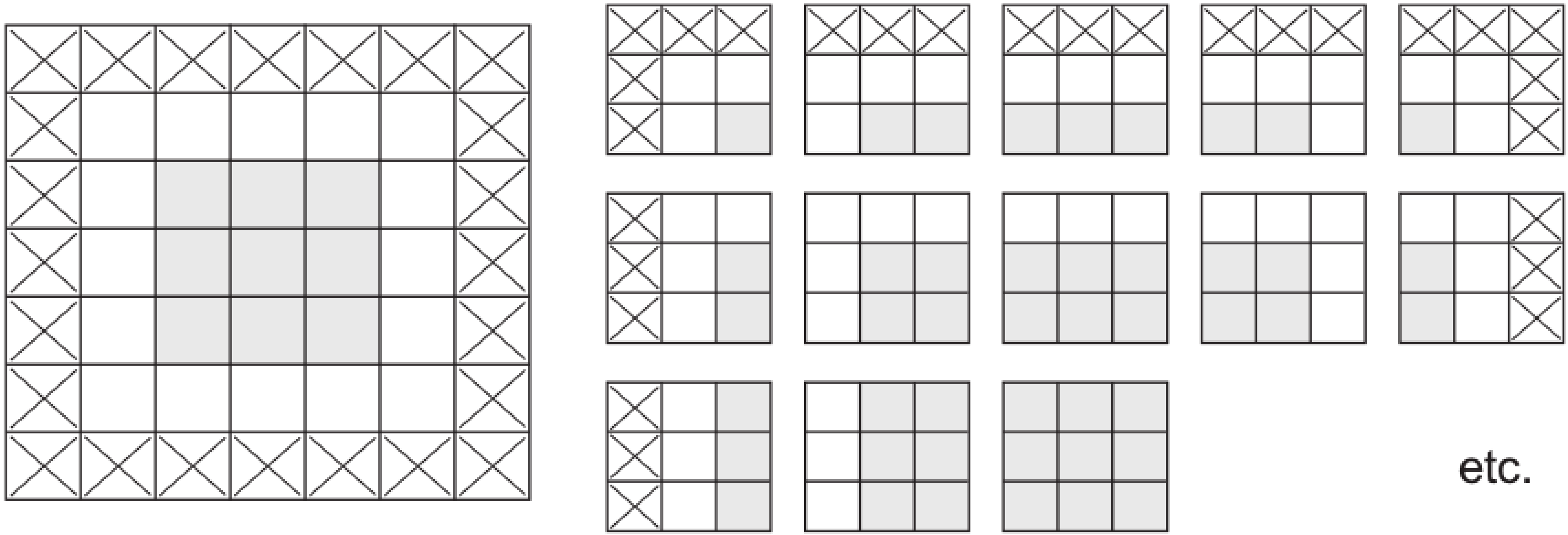




9: the output map is 3x3

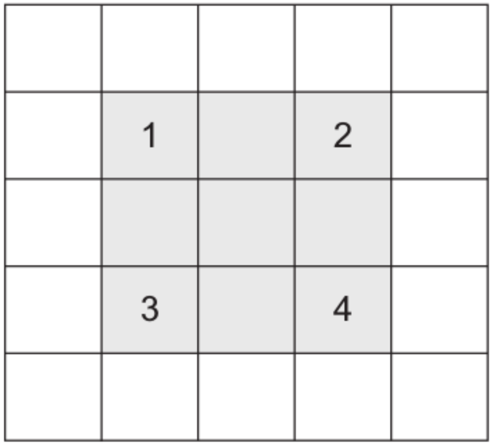
Details: padding

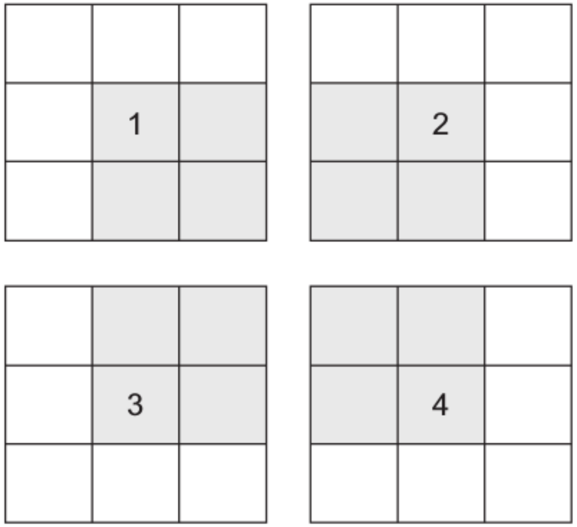
- This is known as “valid” padding mode (default)
- An alternative pads the input map with zeros to yield a same-sized map



Details: striding

- Stride 1x1 is most frequently used: shift 1 pixel at a time → patches are heavily overlapping
- Stride 2x2 skips one patch horizontally and vertically



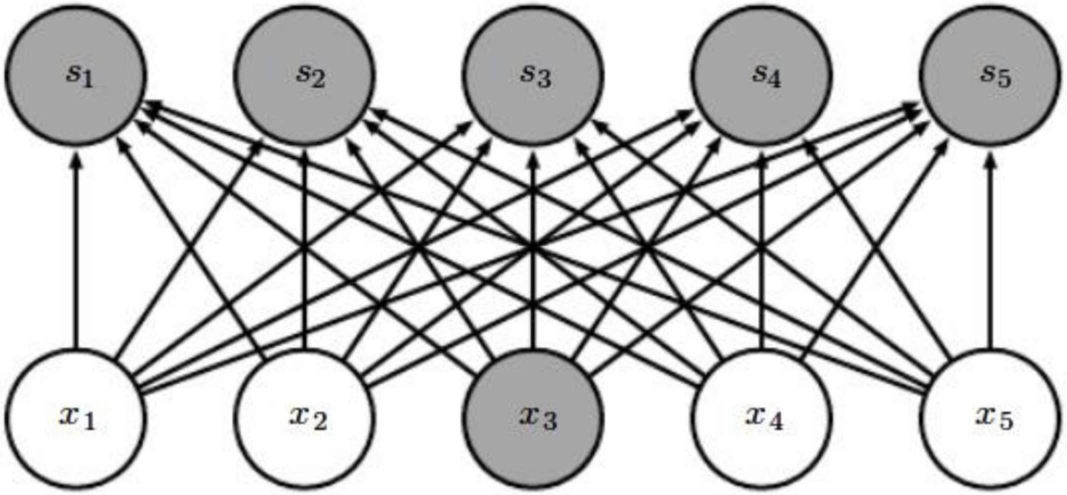


Why convolutional layers?

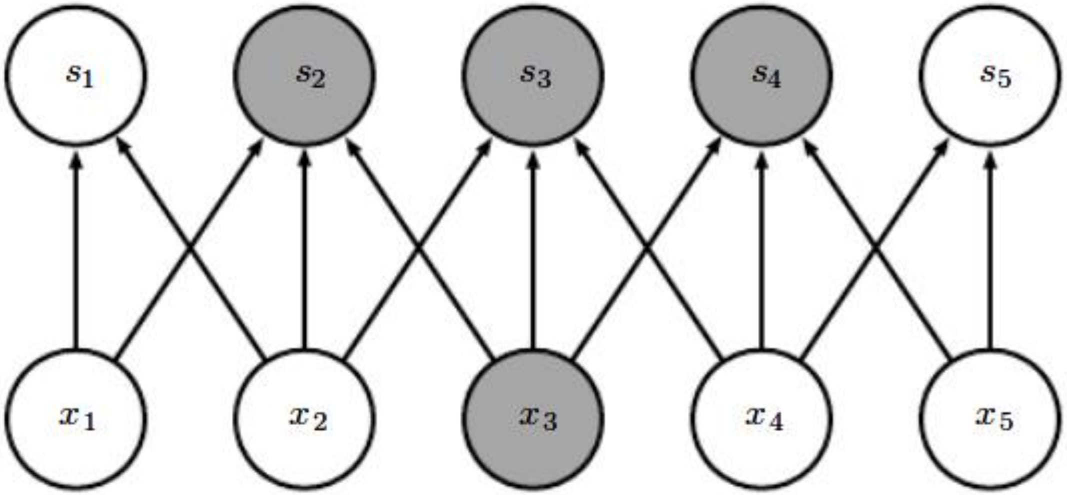
- Sparse connectivity
- Parameter sharing
- Translation invariance

Sparse connectivity

Fully connected

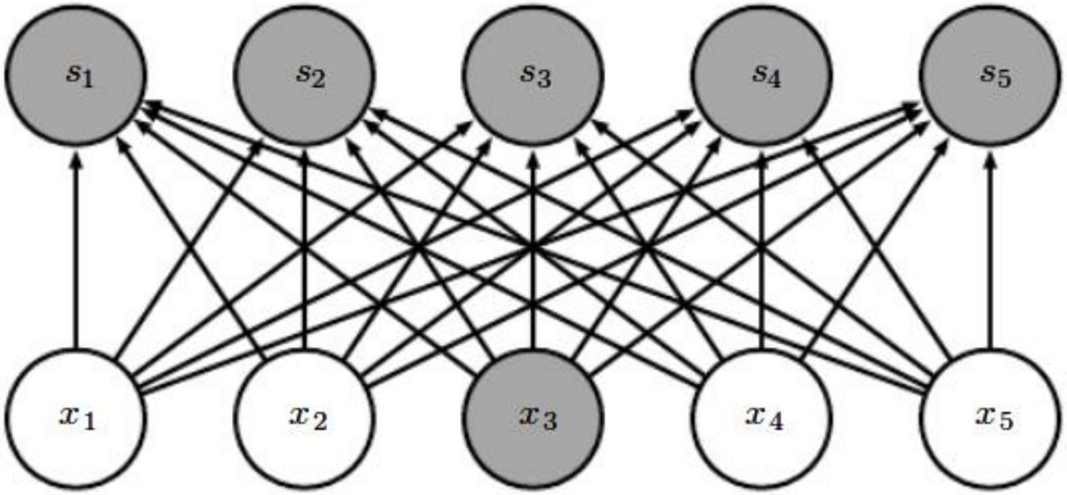


3x1 convolutional

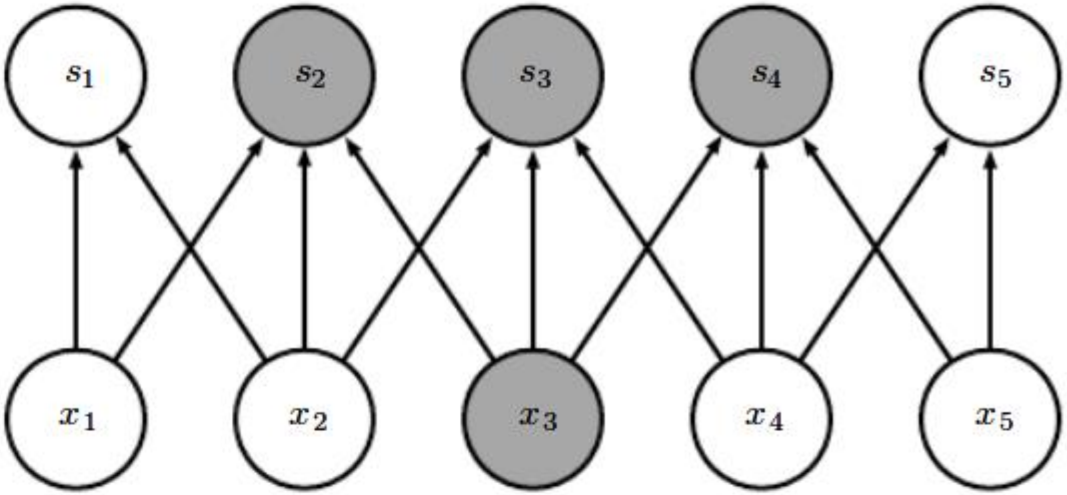


Sparse connectivity

Fully connected

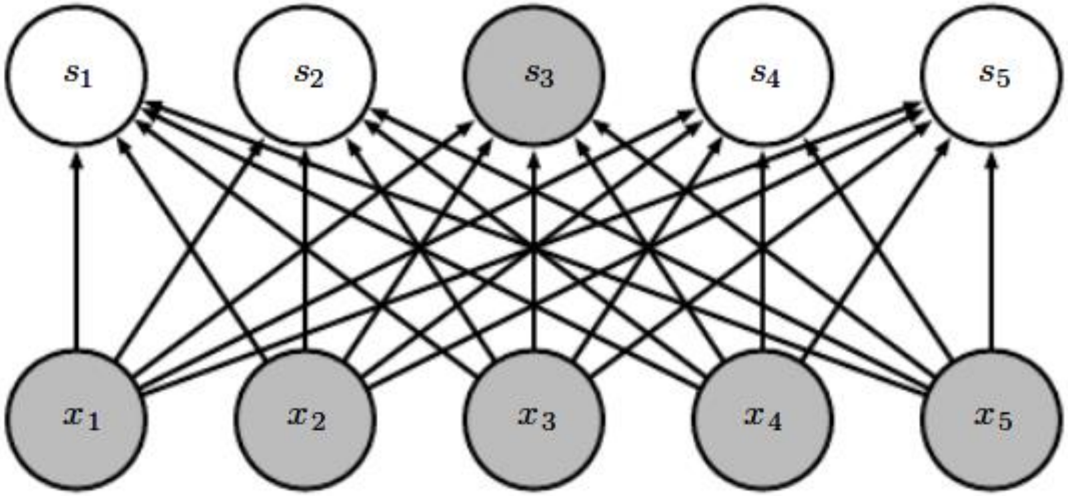


3x1 convolutional

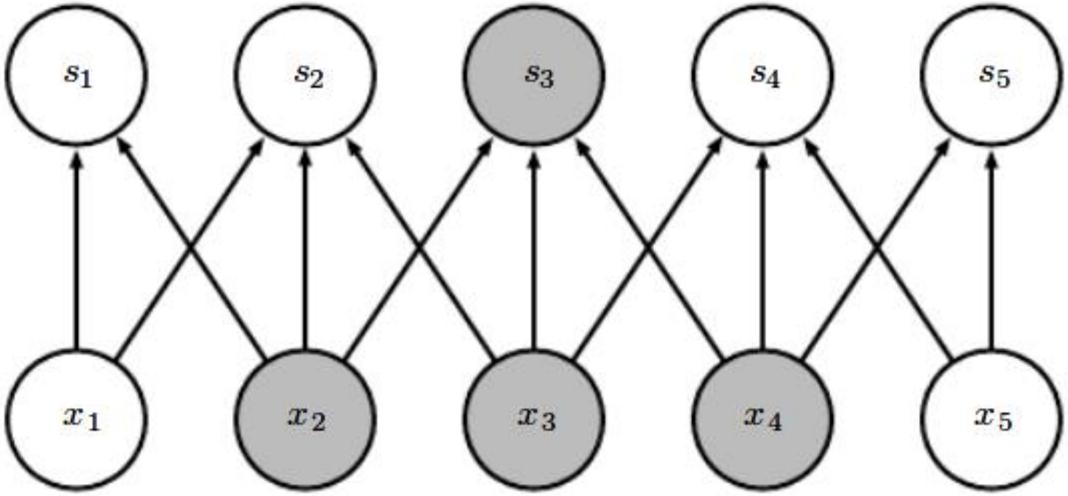


Receptive fields

Fully connected



3x1 convolutional

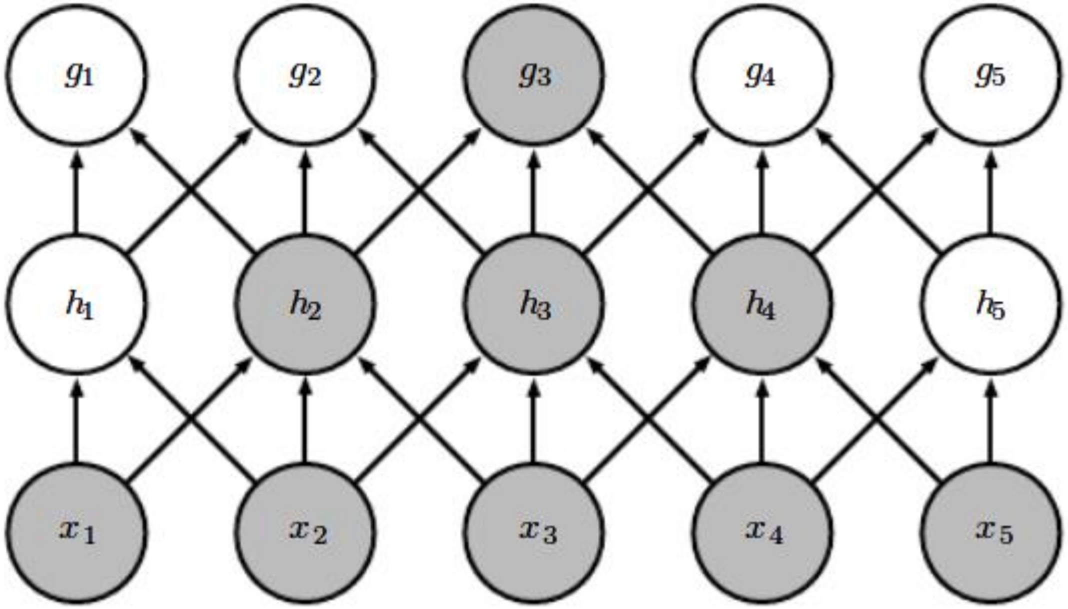


Receptive fields

Deeper neurons depend on wider patches of the input

3x1 convolutional

3x1 convolutional

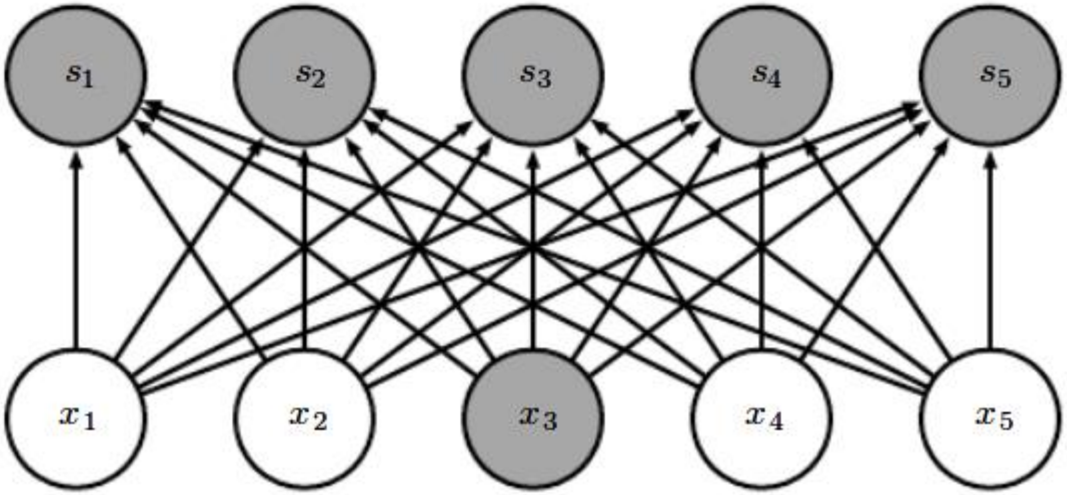


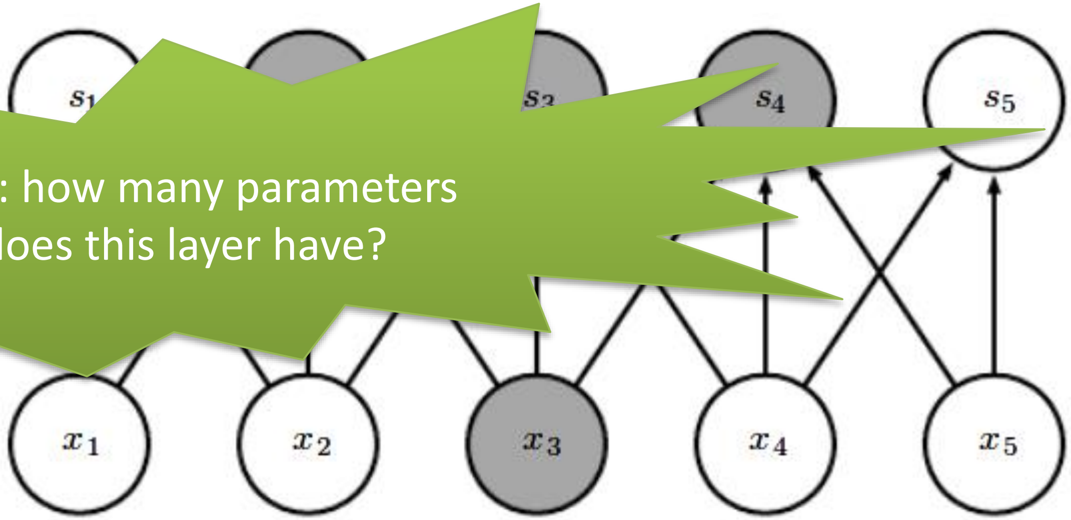
Parameter sharing

Fully connected

5x5 = 25 weights

(+ 5 bias)





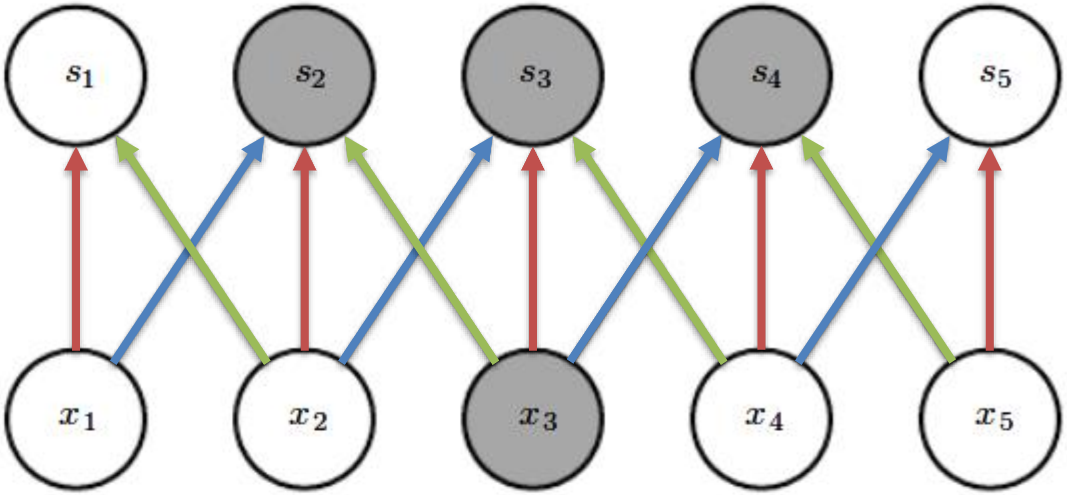
Quiz: how many parameters does this layer have?

3x1 conv

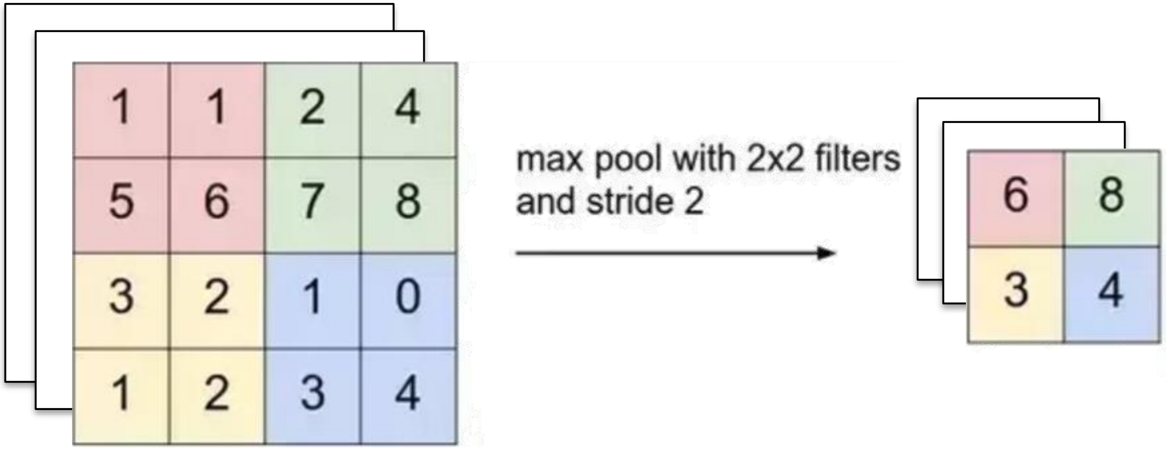
3 weights!

(+ 1 bias)

Translational invariance



Max pooling layers ... on many maps?



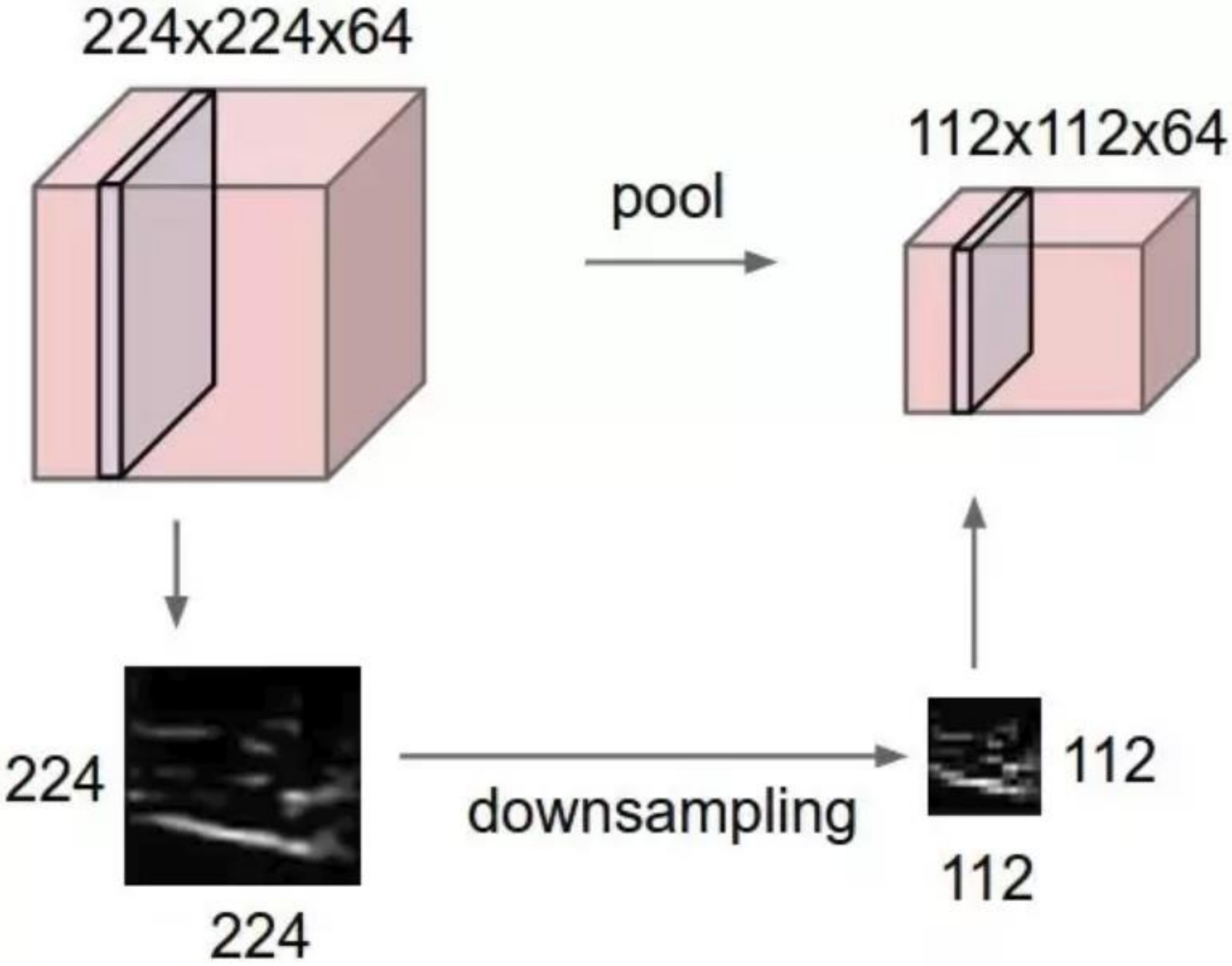
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2 filters and stride 2

6	8
3	4

Quiz: how many parameters does this layer have?

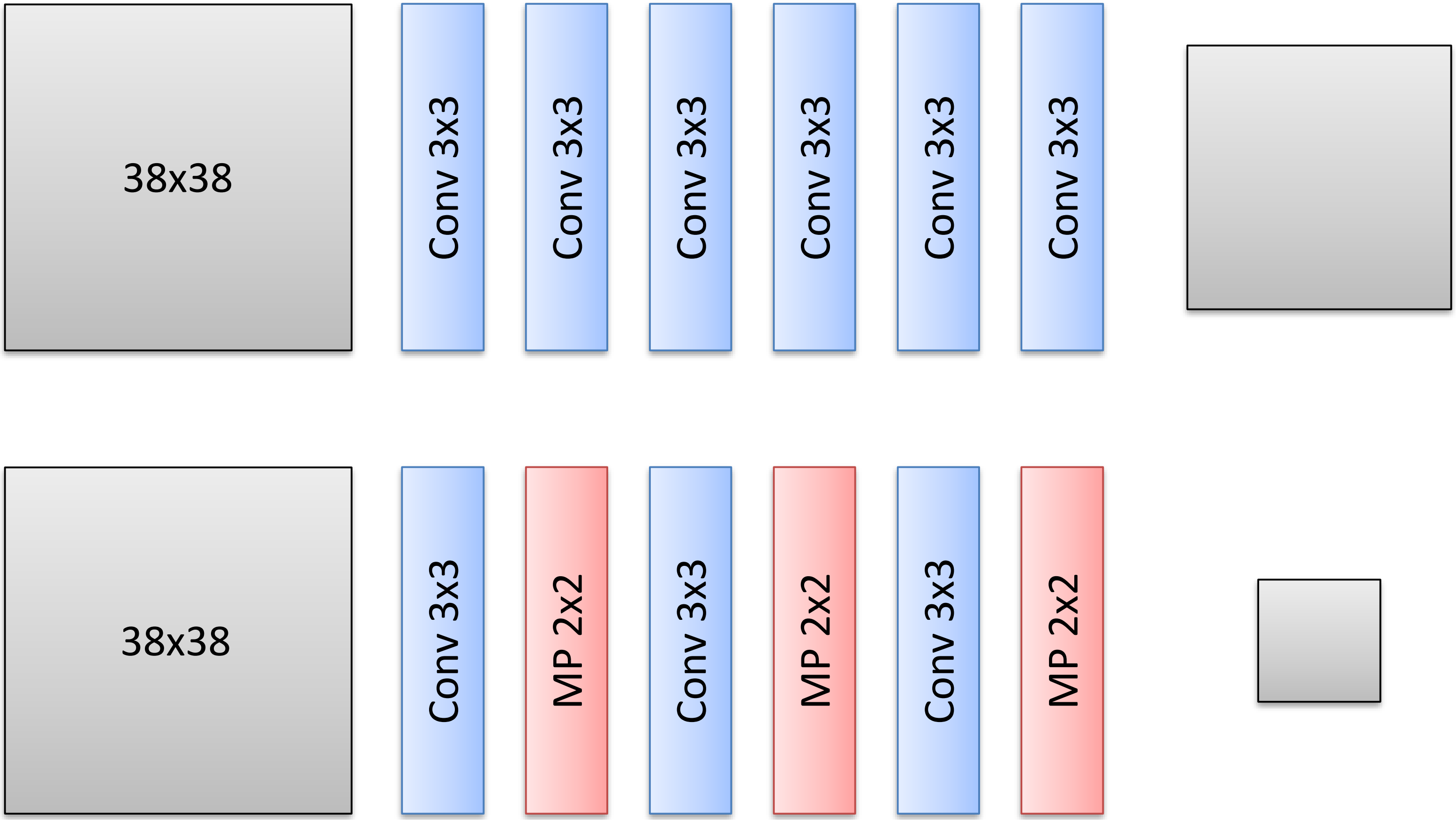
Max pooling downsamples activation maps



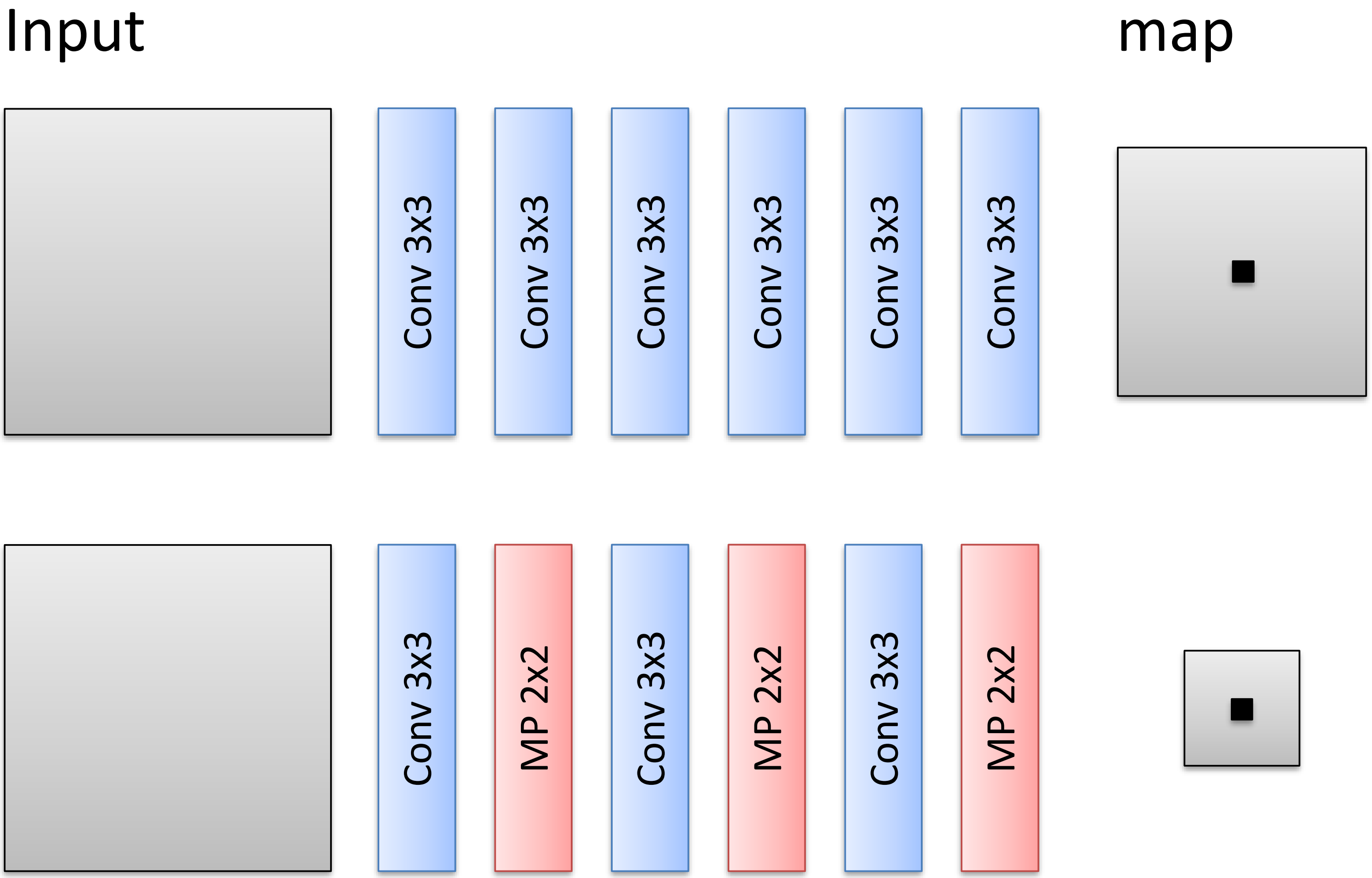
Exercise

Input:

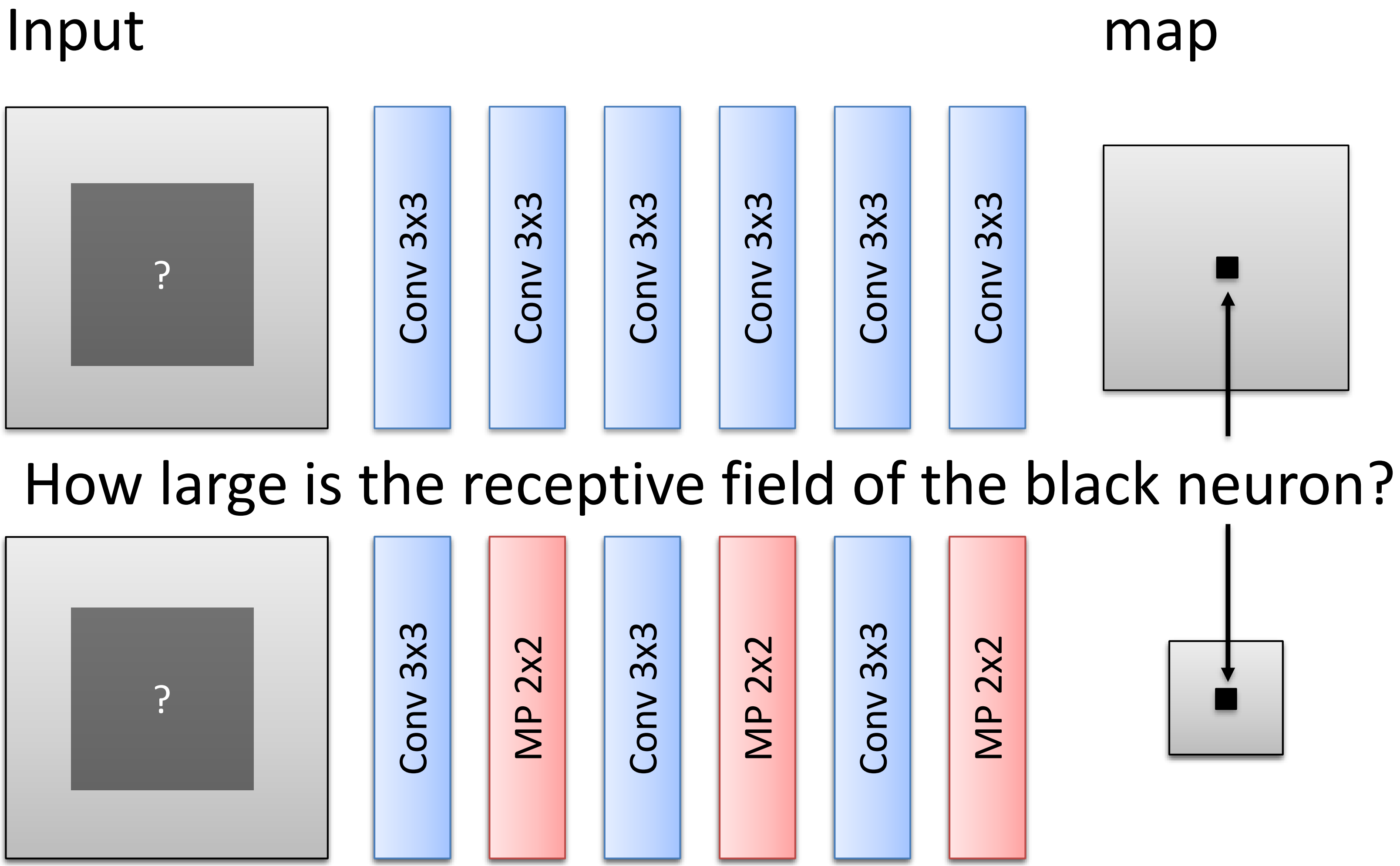
map



Receptive fields



Receptive fields

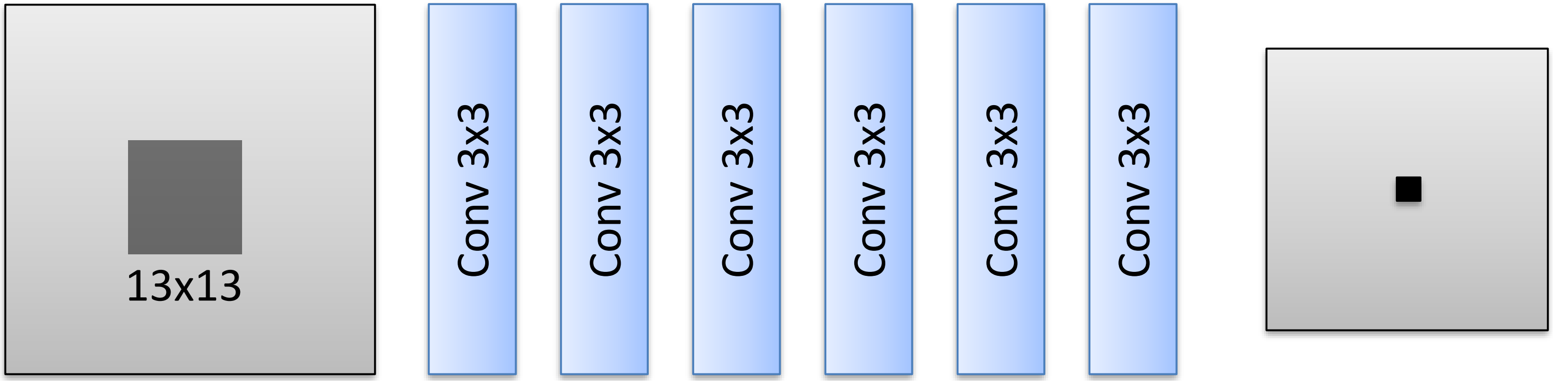


How large is the receptive field of the black neuron?

Receptive fields

Input

map



13x13

Conv 3x3

Conv 3x3

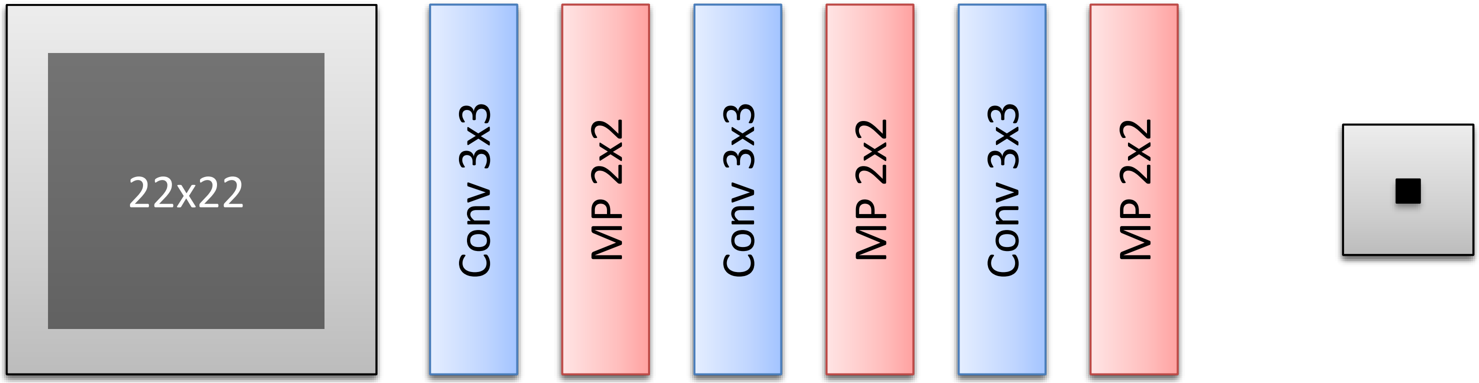
Conv 3x3

Conv 3x3

Conv 3x3

Conv 3x3

How large is the receptive field of the black neuron?



22x22

Conv 3x3

MP 2x2

Conv 3x3

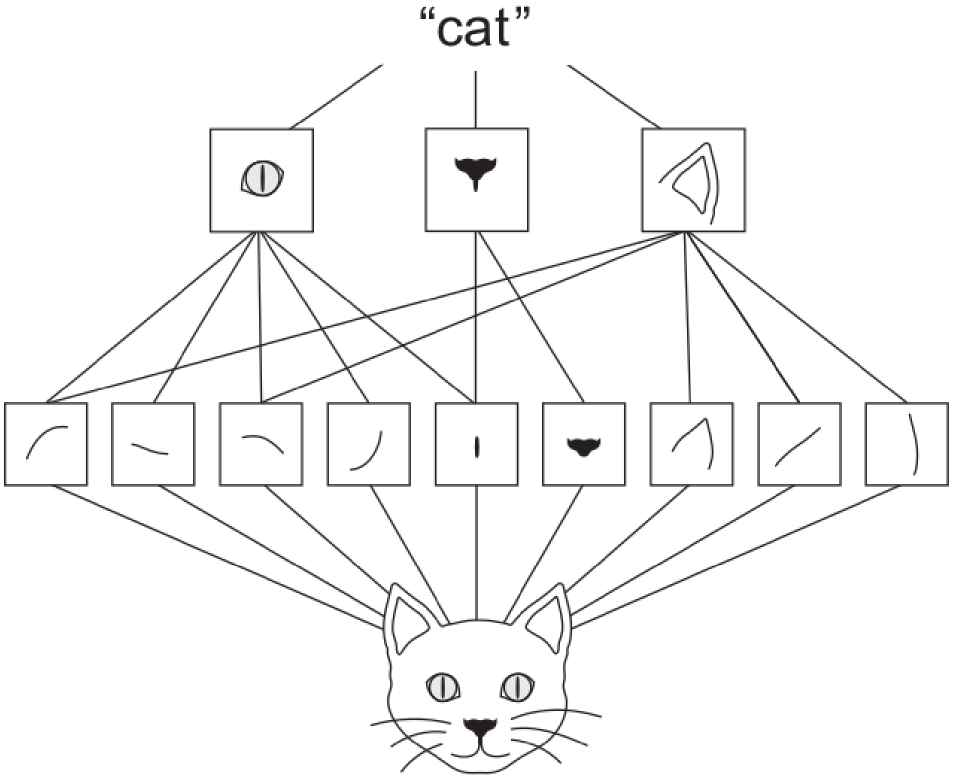
MP 2x2

Conv 3x3

MP 2x2

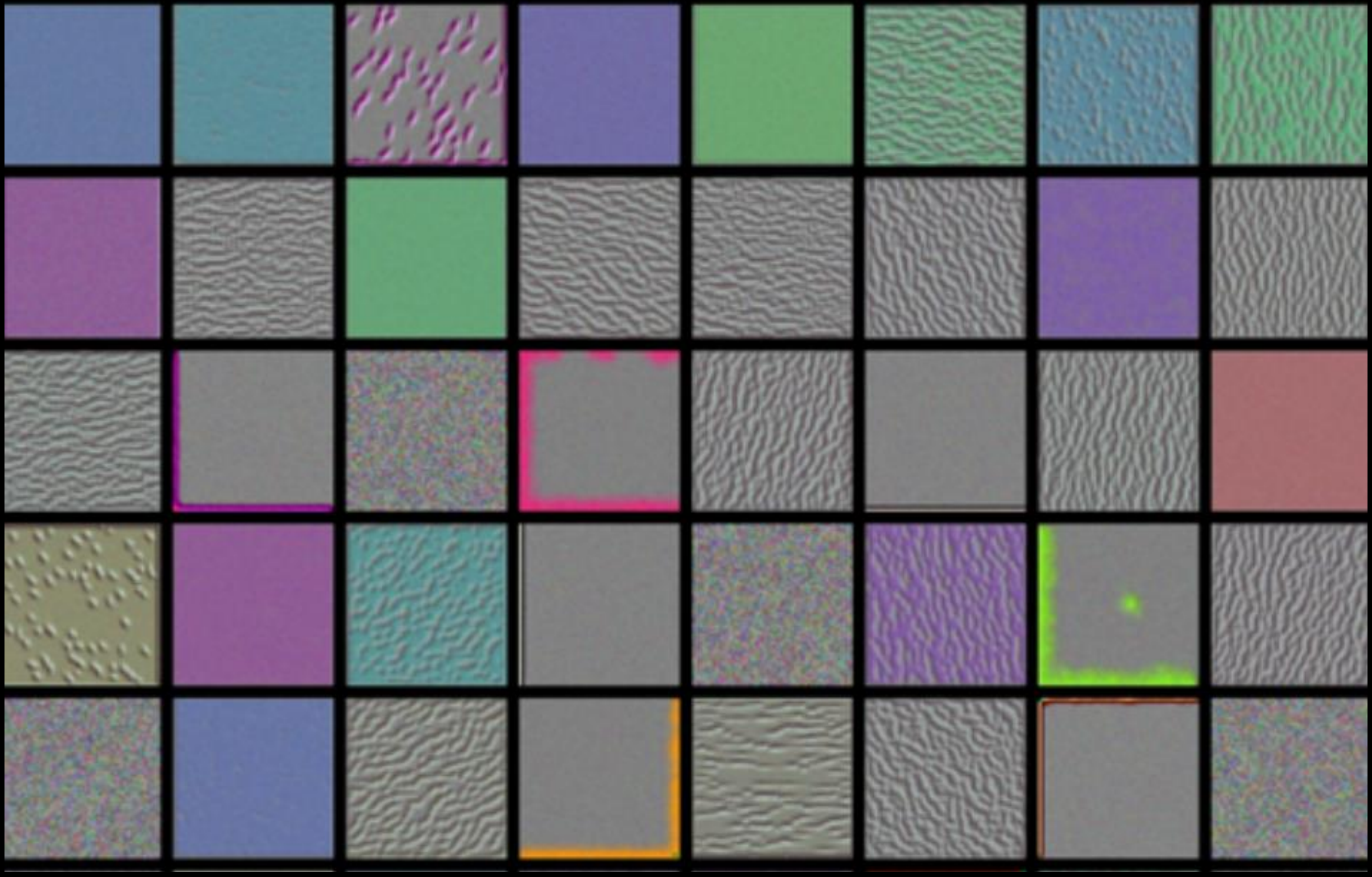
Why convnets work

Convnets learn a **hierarchy** of **translation-invariant** spatial pattern detectors

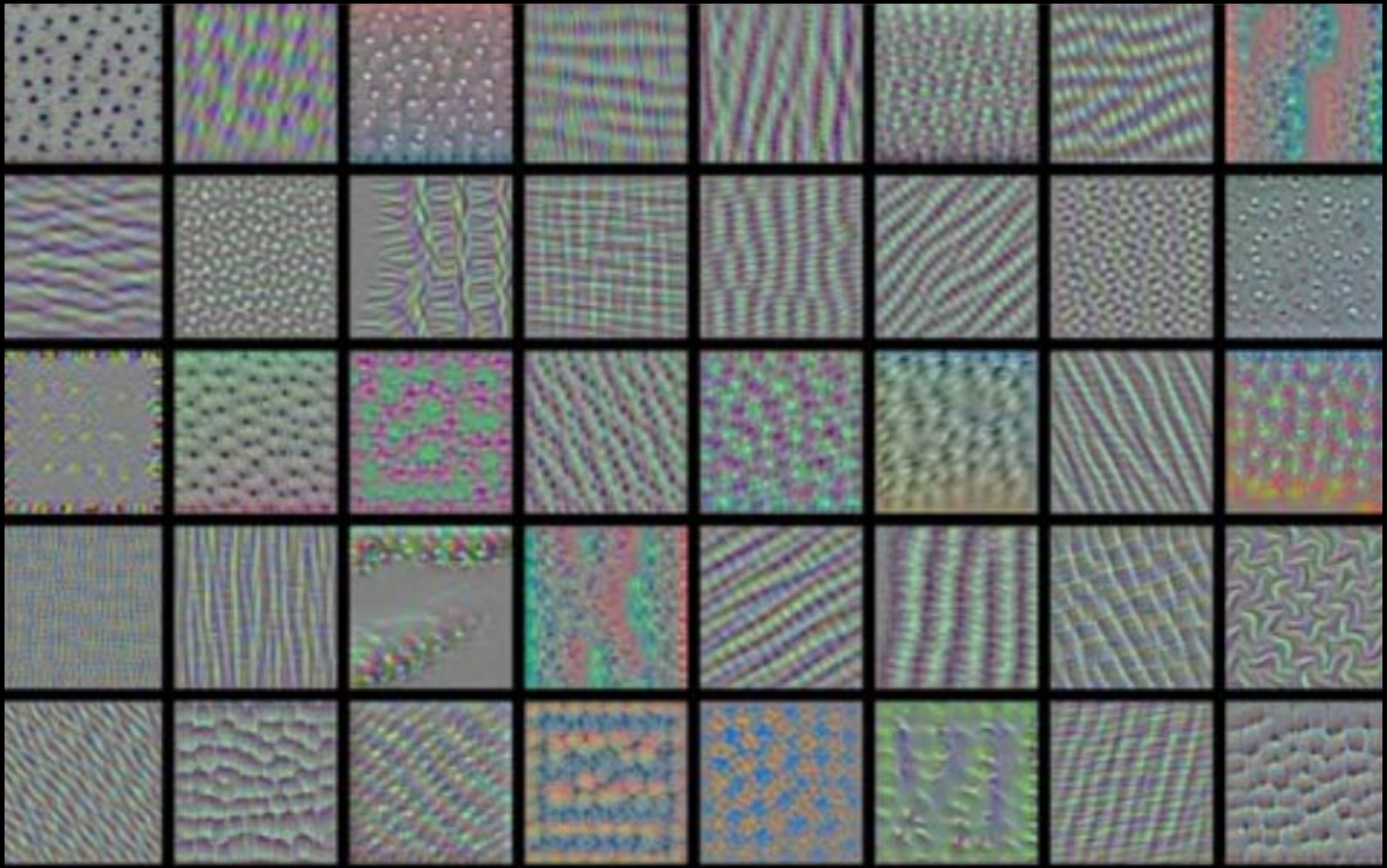


What are layers looking for? Data from a convnet trained on ImageNet

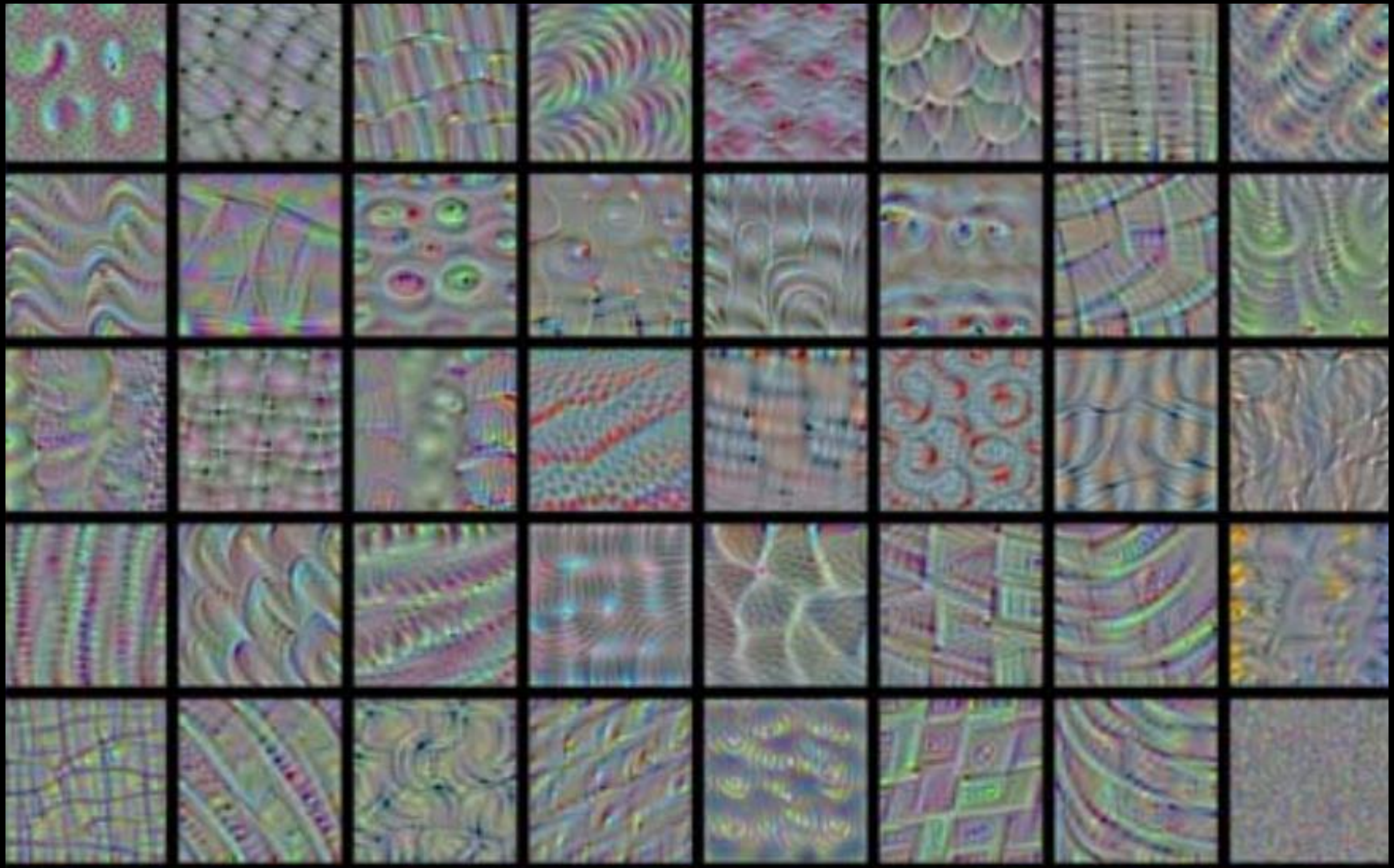
Shallow layers respond to fine, **low-level** patterns



Intermediate layers ...



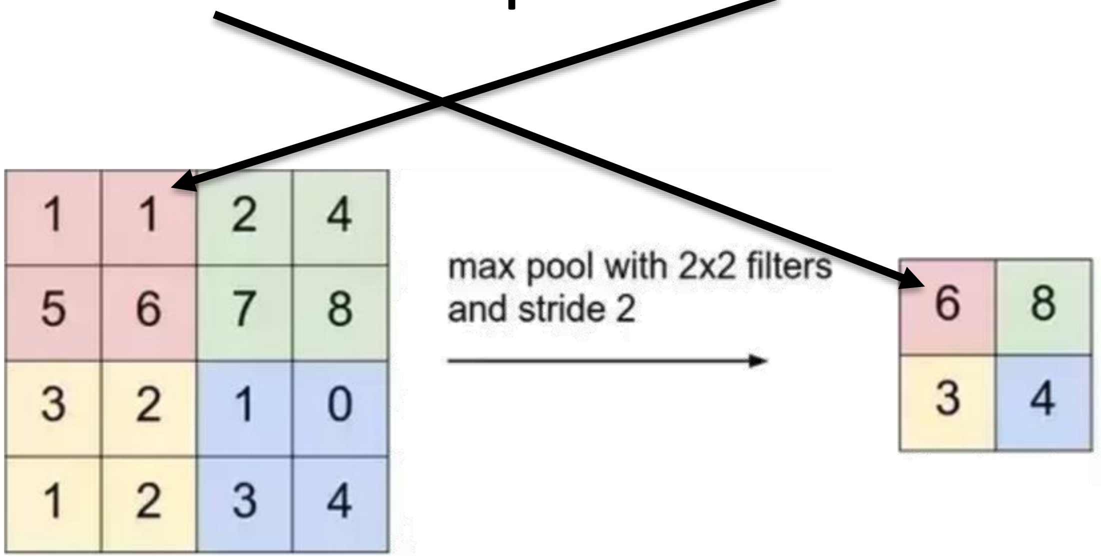
Deep layers respond to complex, **high-level** patterns



Detail: backprop with max pooling

The gradient is only routed through the input pixel that contributes to the output value; e.g.:

Gradient of • with respect to • = 0



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

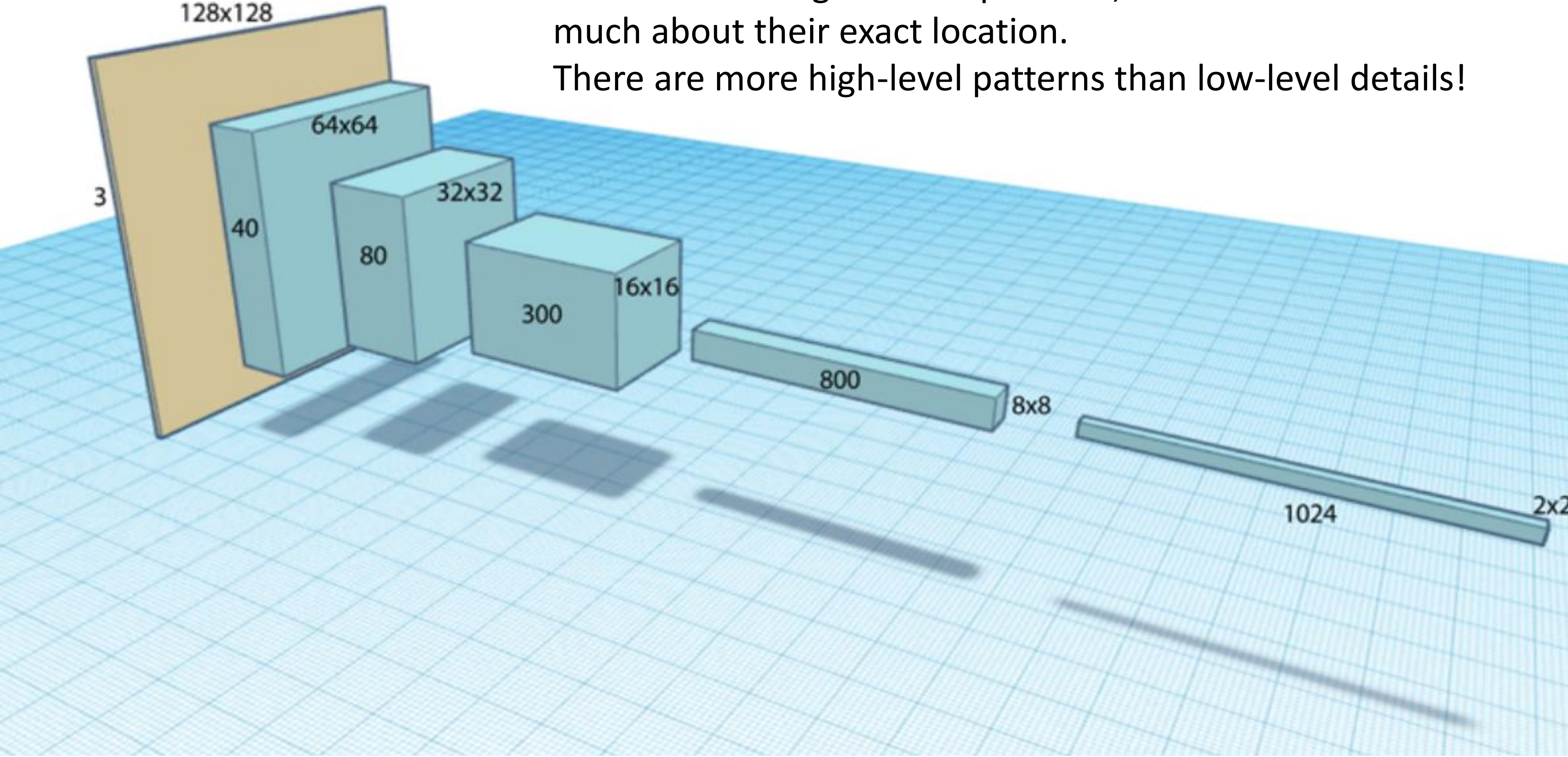
A typical architecture

As we move to deeper layers:

- spatial resolution is reduced
- the number of maps increases

We search for higher-level patterns, and don't care too much about their exact location.

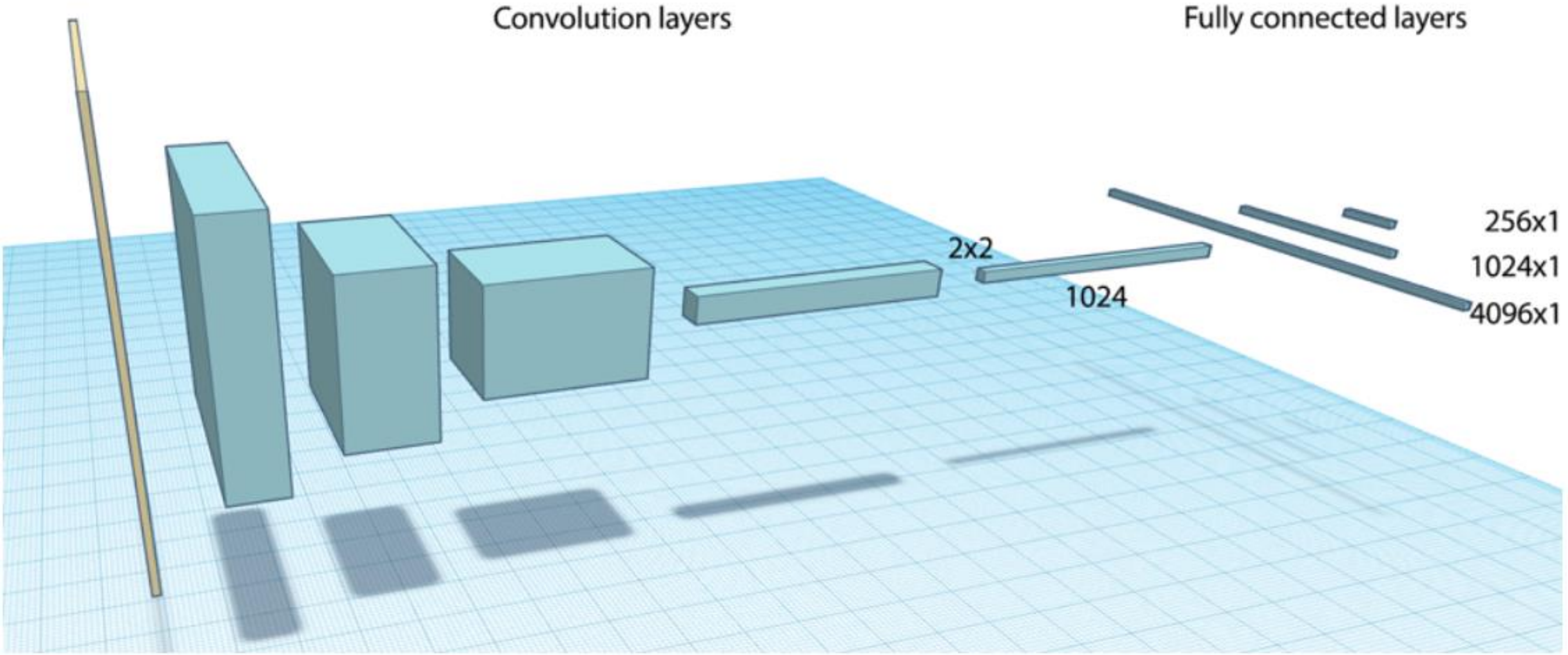
There are more high-level patterns than low-level details!



A typical architecture

Extract high-level features from pixel data

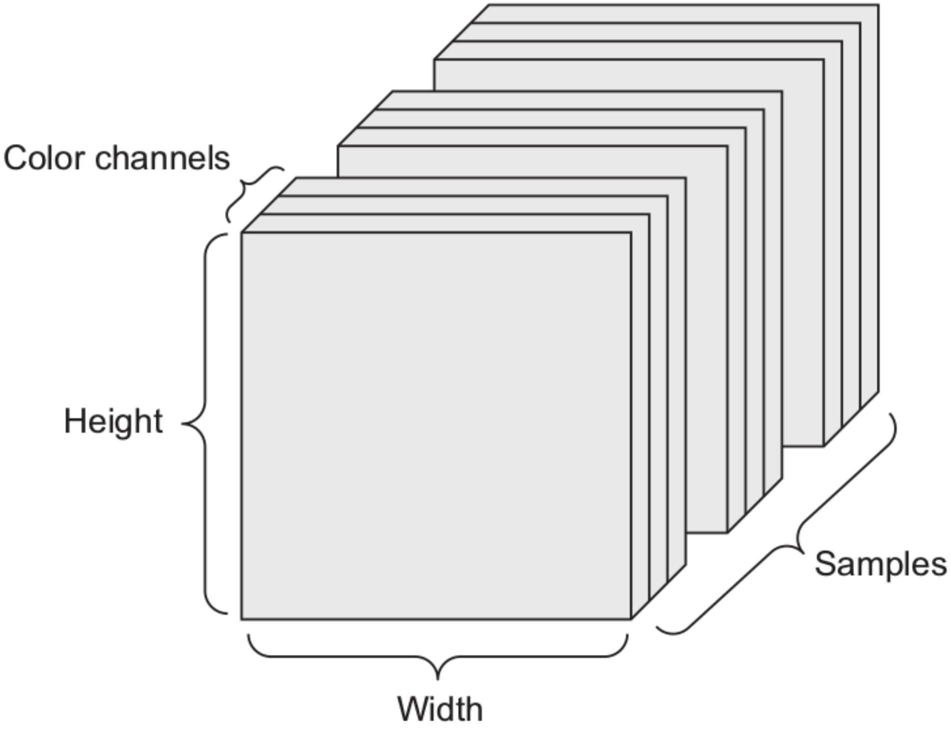
Classify



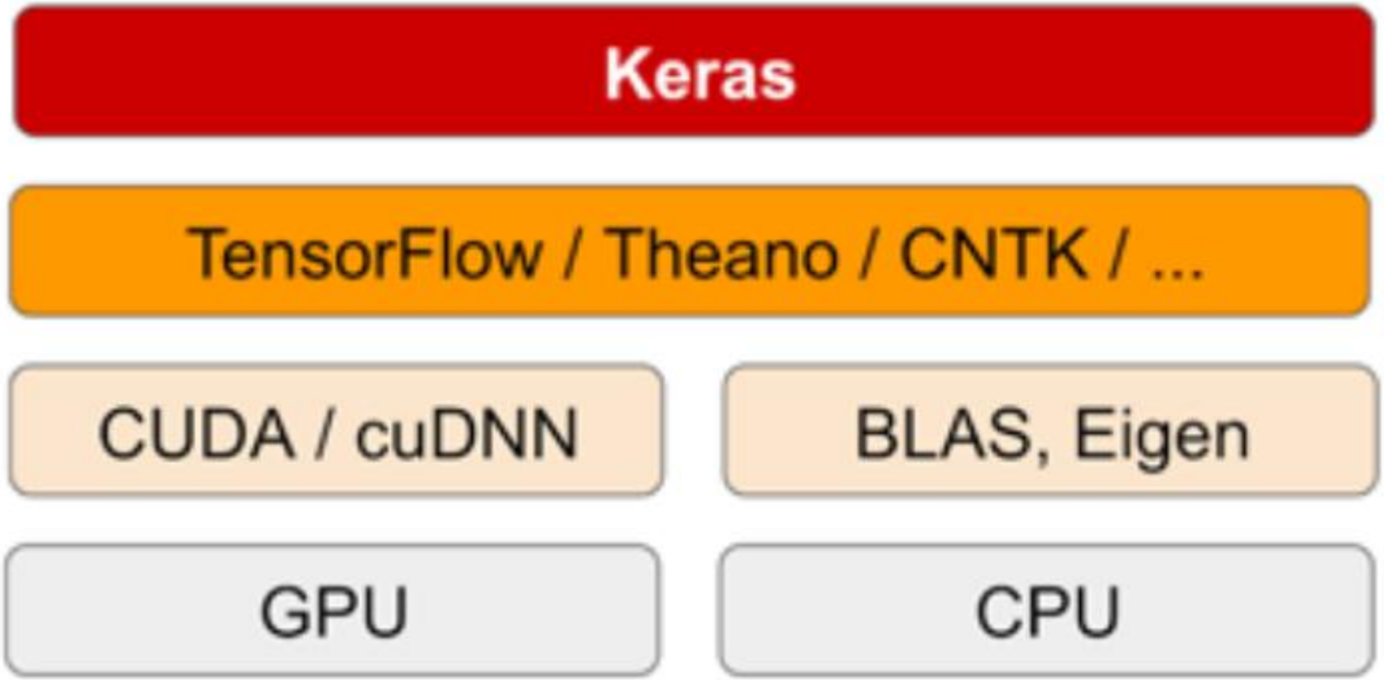
We will manipulate 4D tensors

Images are represented in 4D tensors:

Tensorflow convention:
(samples, height, width, channels)



The software stack



Keras

TensorFlow / Theano / CNTK / ...

CUDA / cuDNN

BLAS, Eigen

GPU

CPU

What is Keras?

- A model-level library, providing high-level building blocks for developing deep-learning models.
- Doesn’t handle low-level operations such as tensor manipulation and differentiation.
- Relies on backends (such as Tensorflow)
- Allows full access to the backend

Why Keras?

Pros:

- Higher level → fewer lines of code
- Modular backend → not tied to tensorflow
- Way to go if you focus on applications

Cons:

- Not as flexible
- Need more flexibility? Access the backend directly!

More about ConvNets

Alessandro Giusti

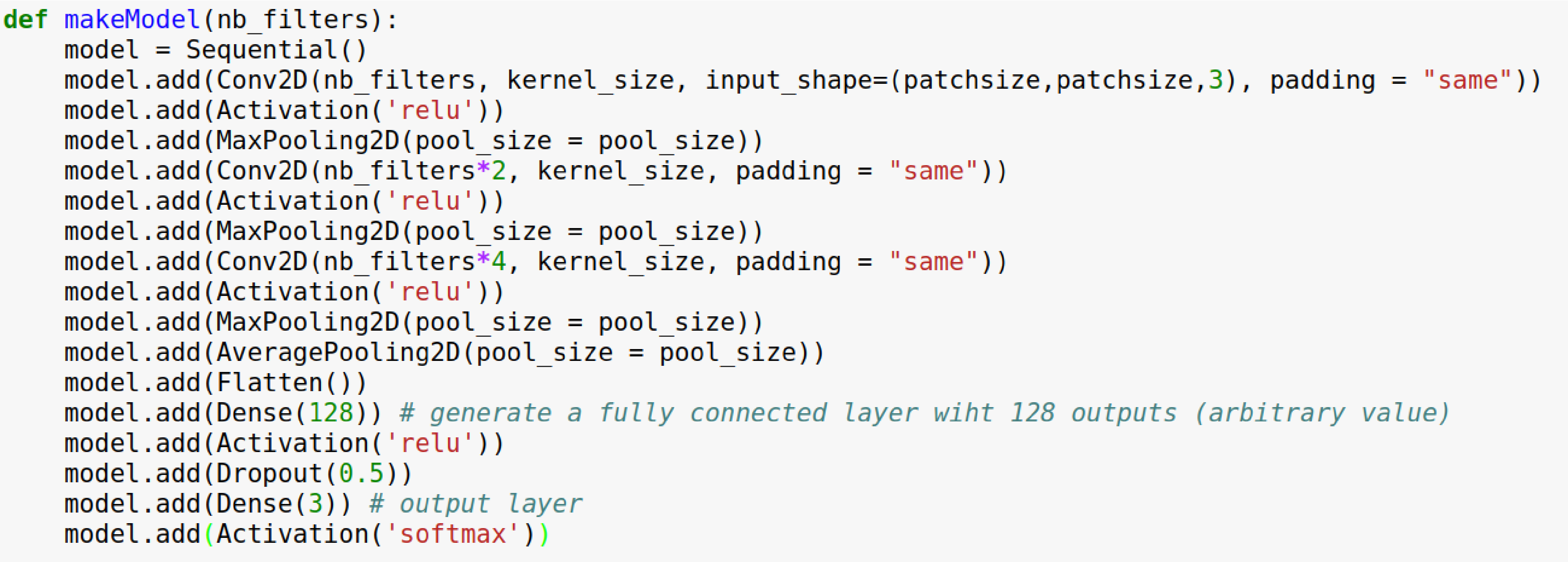
Dalle Molle Institute for Artificial Intelligence

Lugano, Switzerland

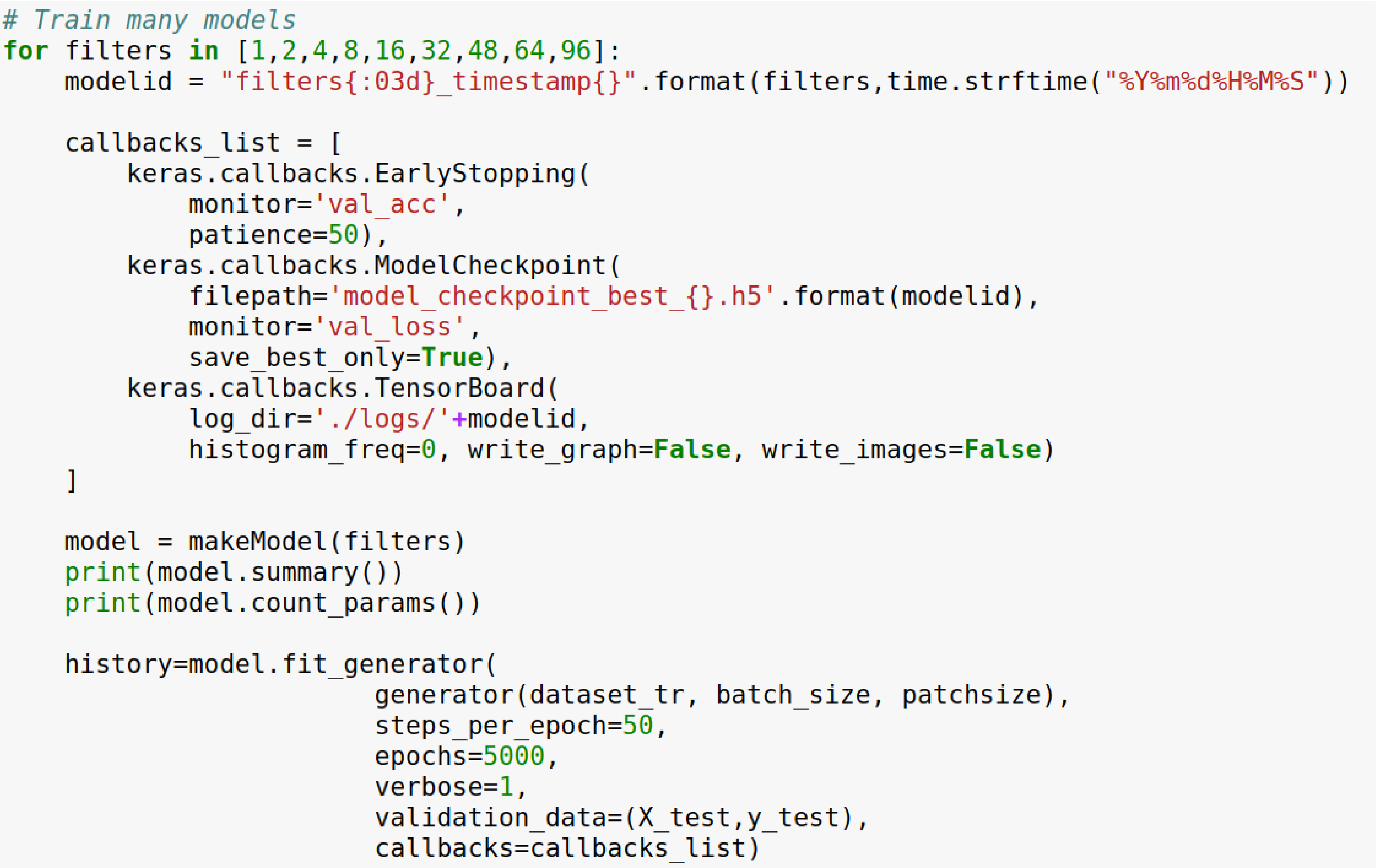
IDSIA

Rock Paper Scissors

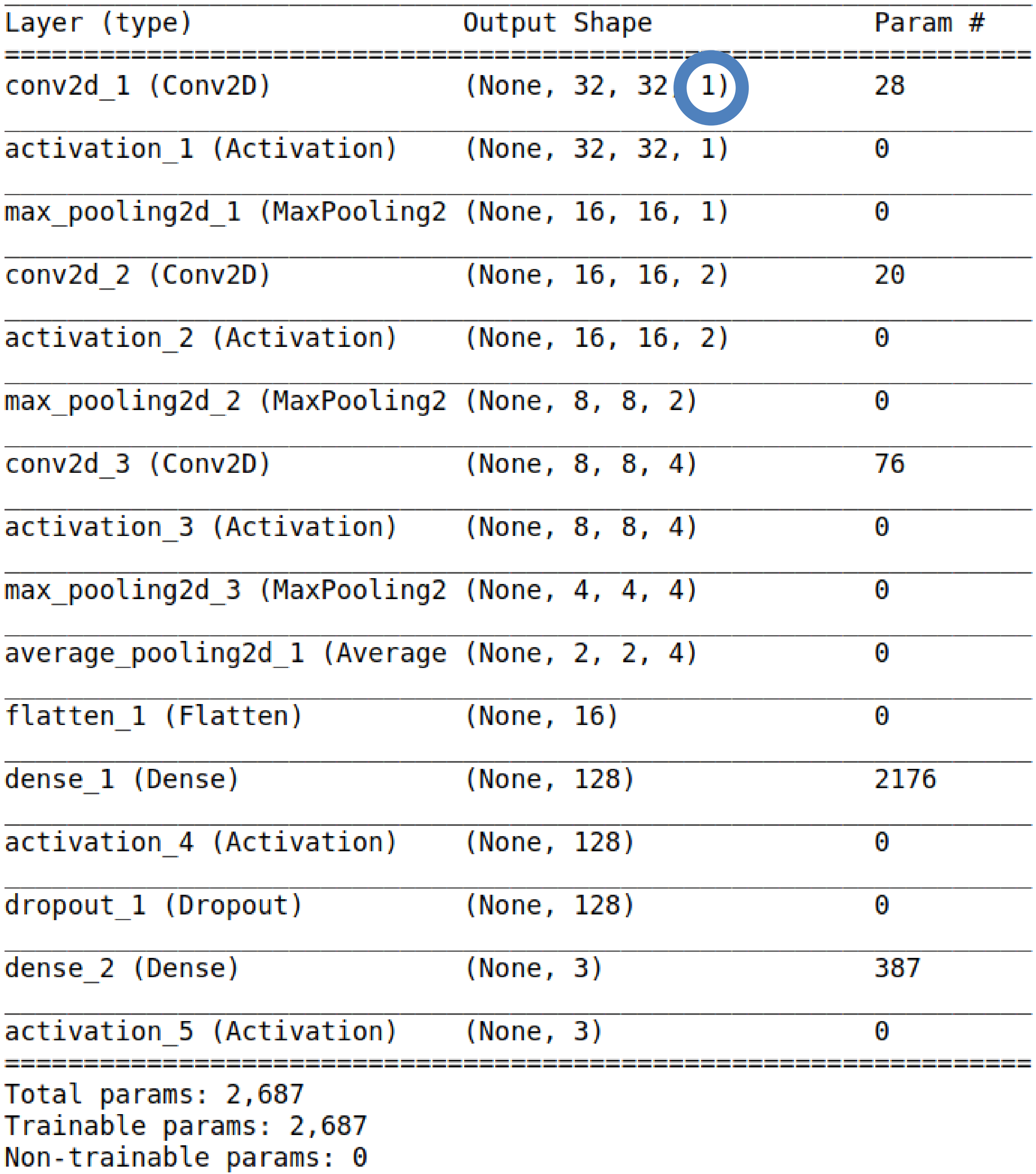
1. UNDERFITTING & OVERFITTING ON OUR ROCK PAPER SCISSORS NET



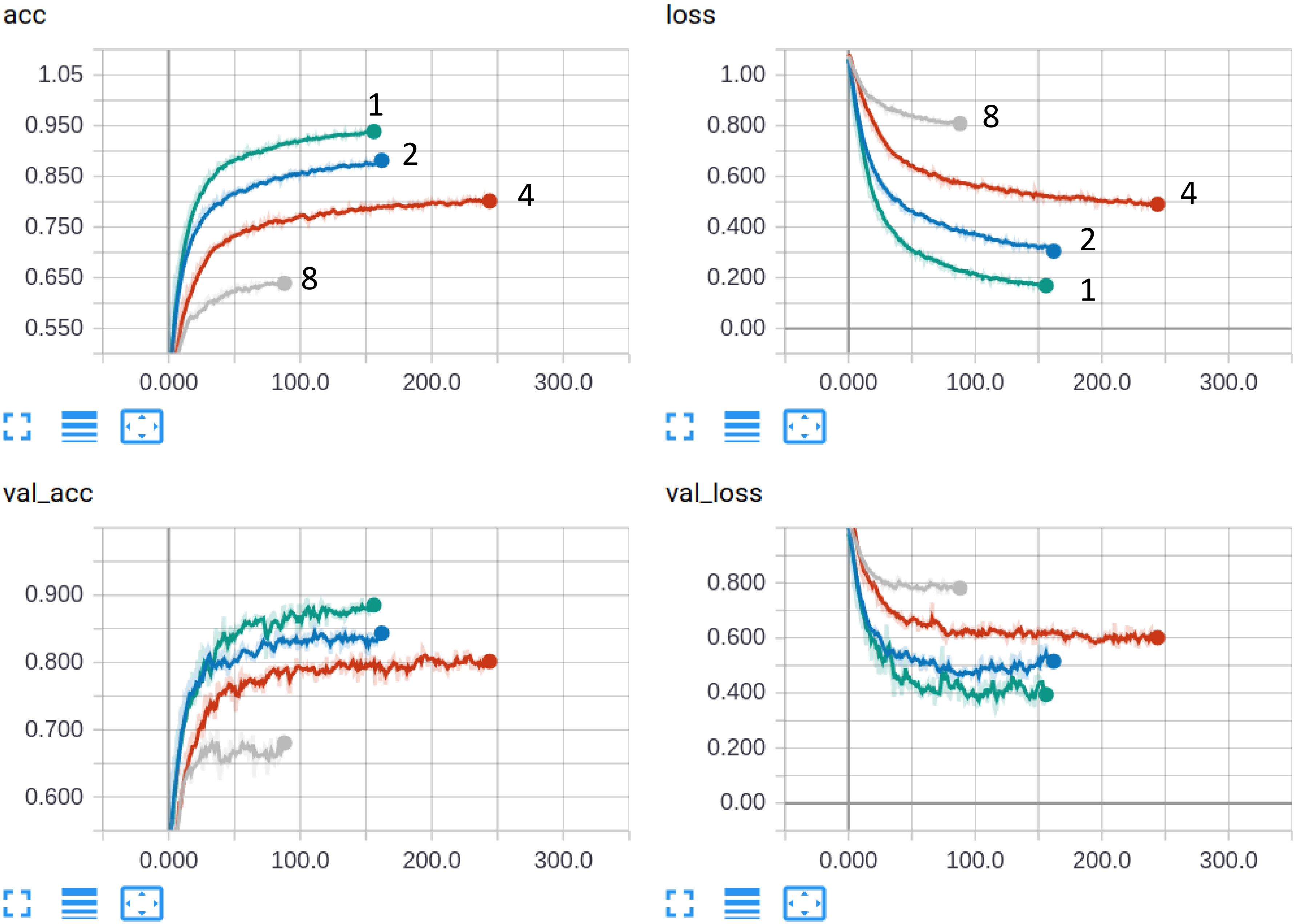
```python
def makeModel(nb_filters):
    model = Sequential()
    model.add(Conv2D(nb_filters, kernel_size, input_shape=(patchsize,patchsize,3), padding = "same"))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size = pool_size))
    model.add(Conv2D(nb_filters*2, kernel_size, padding = "same"))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size = pool_size))
    model.add(Conv2D(nb_filters*4, kernel_size, padding = "same"))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size = pool_size))
    model.add(AveragePooling2D(pool_size = pool_size))
    model.add(Flatten())
    model.add(Dense(128)) # generate a fully connected layer wiht 128 outputs (arbitrary value)
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(3)) # output layer
    model.add(Activation('softmax'))
```



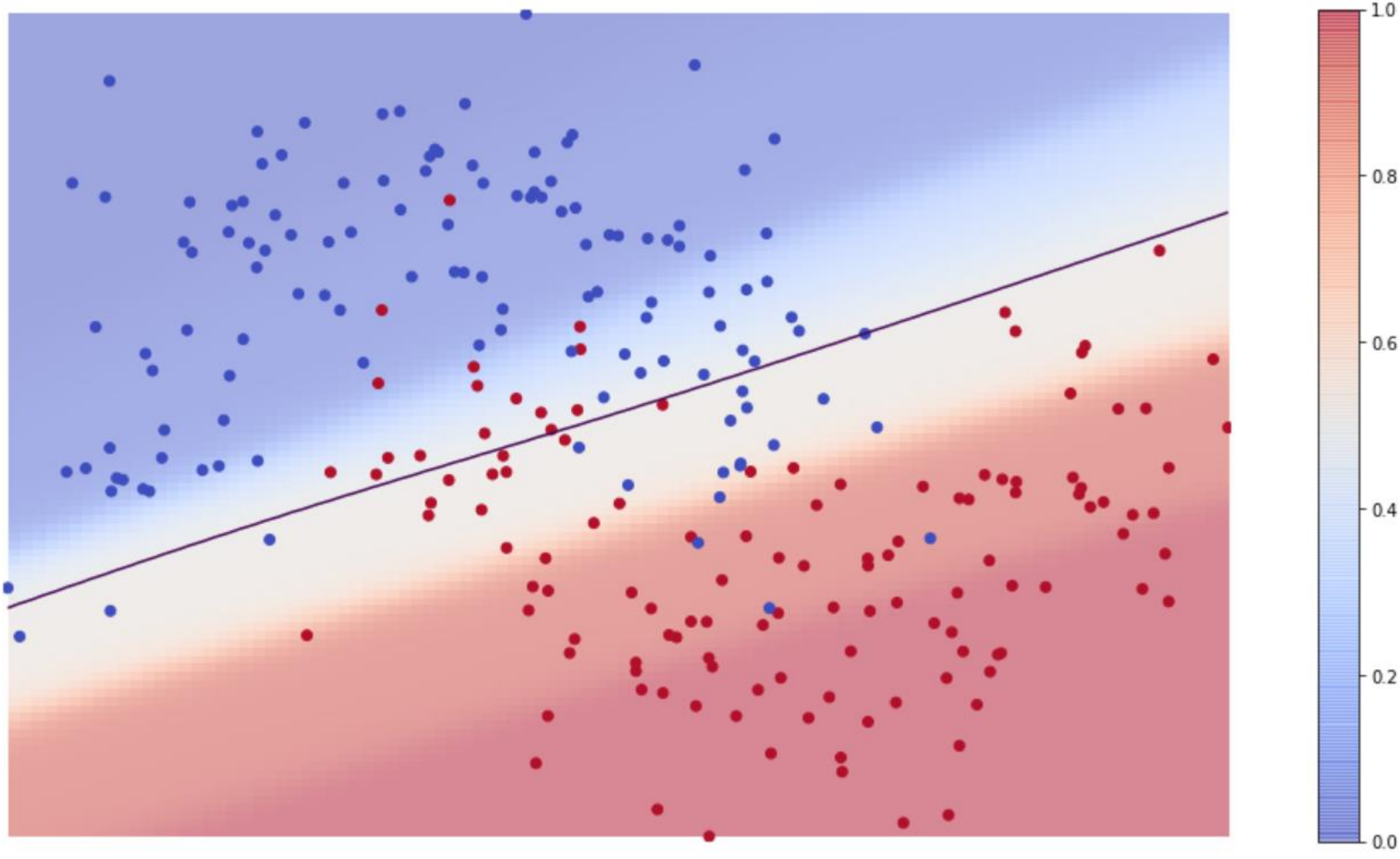
```python
# Train many models
for filters in [1,2,4,8,16,32,48,64,96]:
    modelid = "filters{:03d}_timestamp{}".format(filters,time.strftime("%Y%m%d%H%M%S"))

    callbacks_list = [
        keras.callbacks.EarlyStopping(
            monitor='val_acc',
            patience=50),
        keras.callbacks.ModelCheckpoint(
            filepath='model_checkpoint_best_{}.h5'.format(modelid),
            monitor='val_loss',
            save_best_only=True),
        keras.callbacks.TensorBoard(
            log_dir='./logs/'+modelid,
            histogram_freq=0, write_graph=False, write_images=False)
    ]

    model = makeModel(filters)
    print(model.summary())
    print(model.count_params())

    history=model.fit_generator(
                    generator(dataset_tr, batch_size, patchsize),
                    steps_per_epoch=50,
                    epochs=5000,
                    verbose=1,
                    validation_data=(X_test,y_test),
                    callbacks=callbacks_list)
```

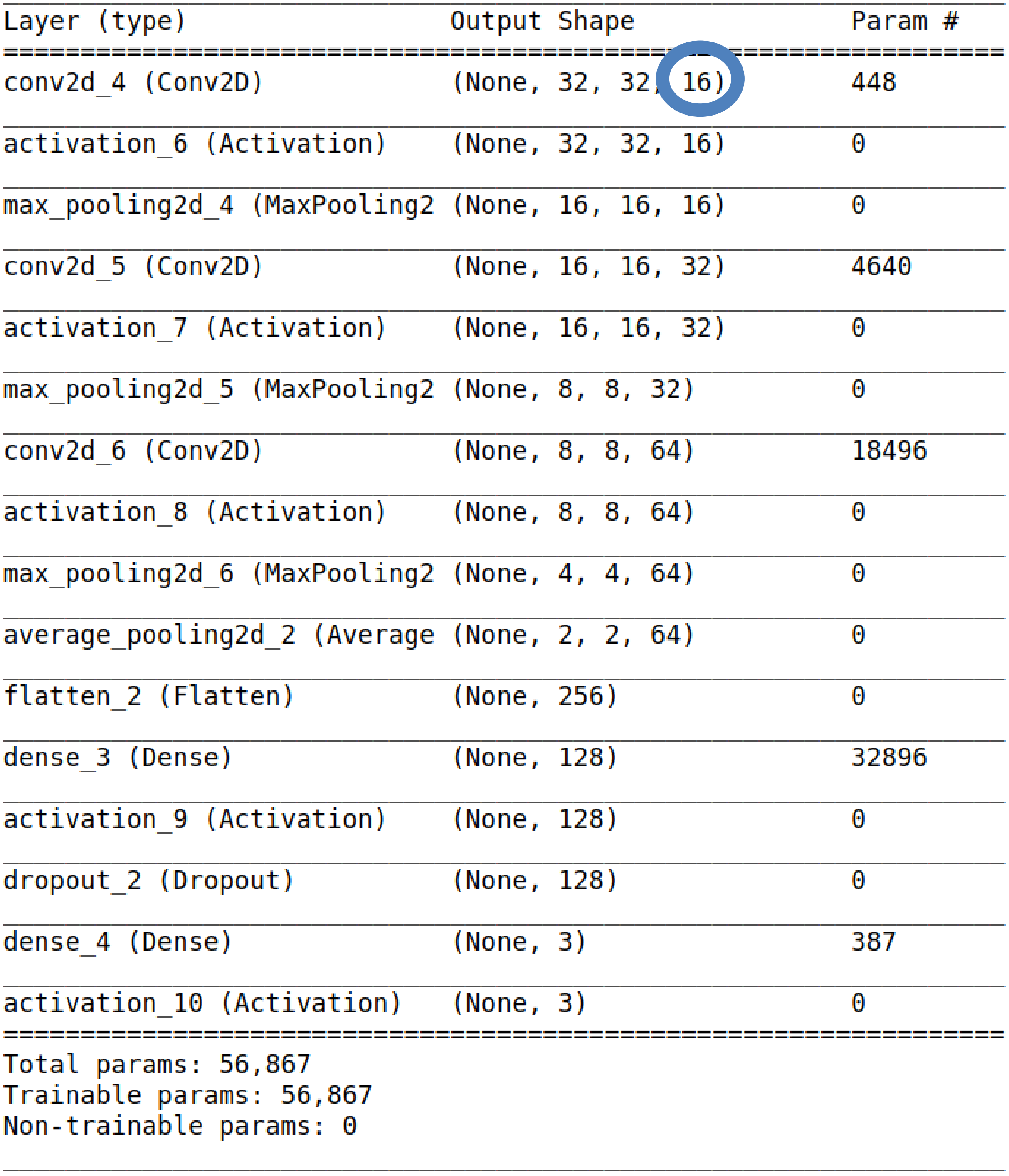


```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 1)         28
_________________________________________________________________
activation_1 (Activation)    (None, 32, 32, 1)         0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 1)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 2)         20
_________________________________________________________________
activation_2 (Activation)    (None, 16, 16, 2)         0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 2)           0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 4)           76
_________________________________________________________________
activation_3 (Activation)    (None, 8, 8, 4)           0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 4)           0
_________________________________________________________________
average_pooling2d_1 (Average (None, 2, 2, 4)           0
_________________________________________________________________
flatten_1 (Flatten)          (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               2176
_________________________________________________________________
activation_4 (Activation)    (None, 128)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 387
_________________________________________________________________
activation_5 (Activation)    (None, 3)                 0
=================================================================
Total params: 2,687
Trainable params: 2,687
Non-trainable params: 0
```

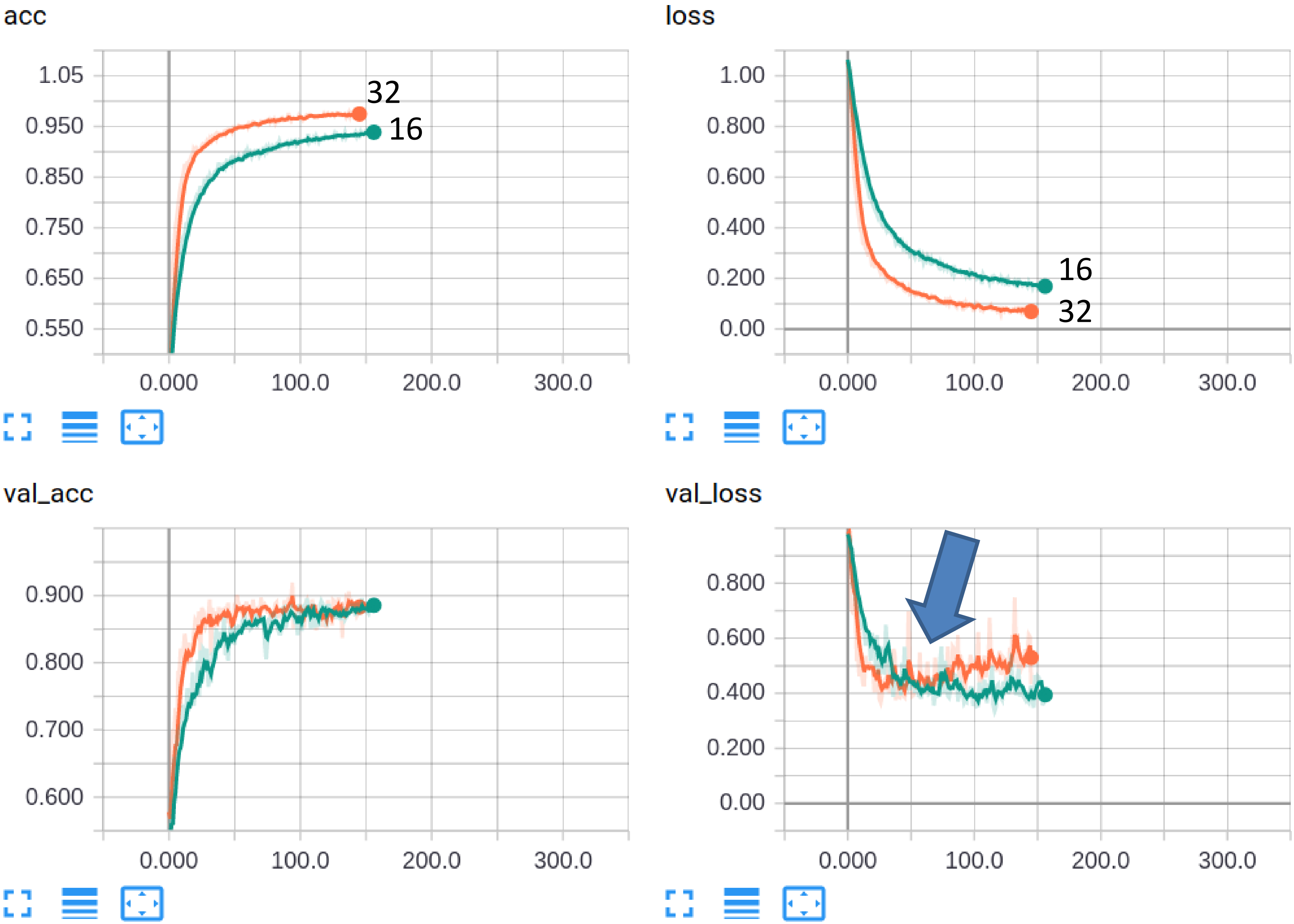

acc
1.05
0.950
0.850
0.750
0.650
0.550
0.000
100.0
200.0
300.0
1
2
4
8
loss
1.00
0.800
0.600
0.400
0.200
0.00
0.000
100.0
200.0
300.0
8
4
2
1
val_acc
0.900
0.800
0.700
0.600
0.000
100.0
200.0
300.0
val_loss
0.800
0.600
0.400
0.200
0.00
0.000
100.0
200.0
300.0

1, 2, 4, 8

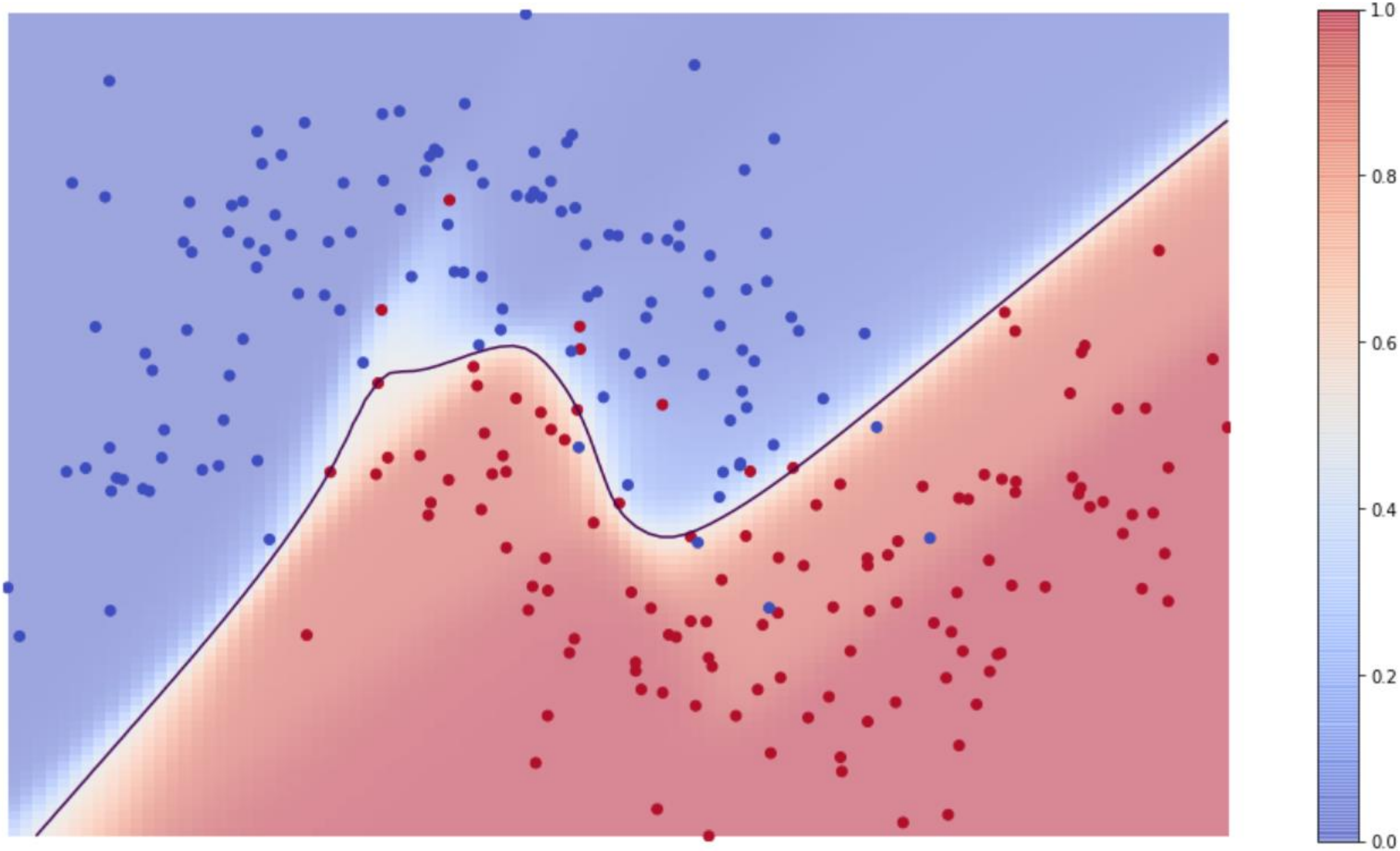

1.0
0.8
0.6
0.4
0.2
0.0

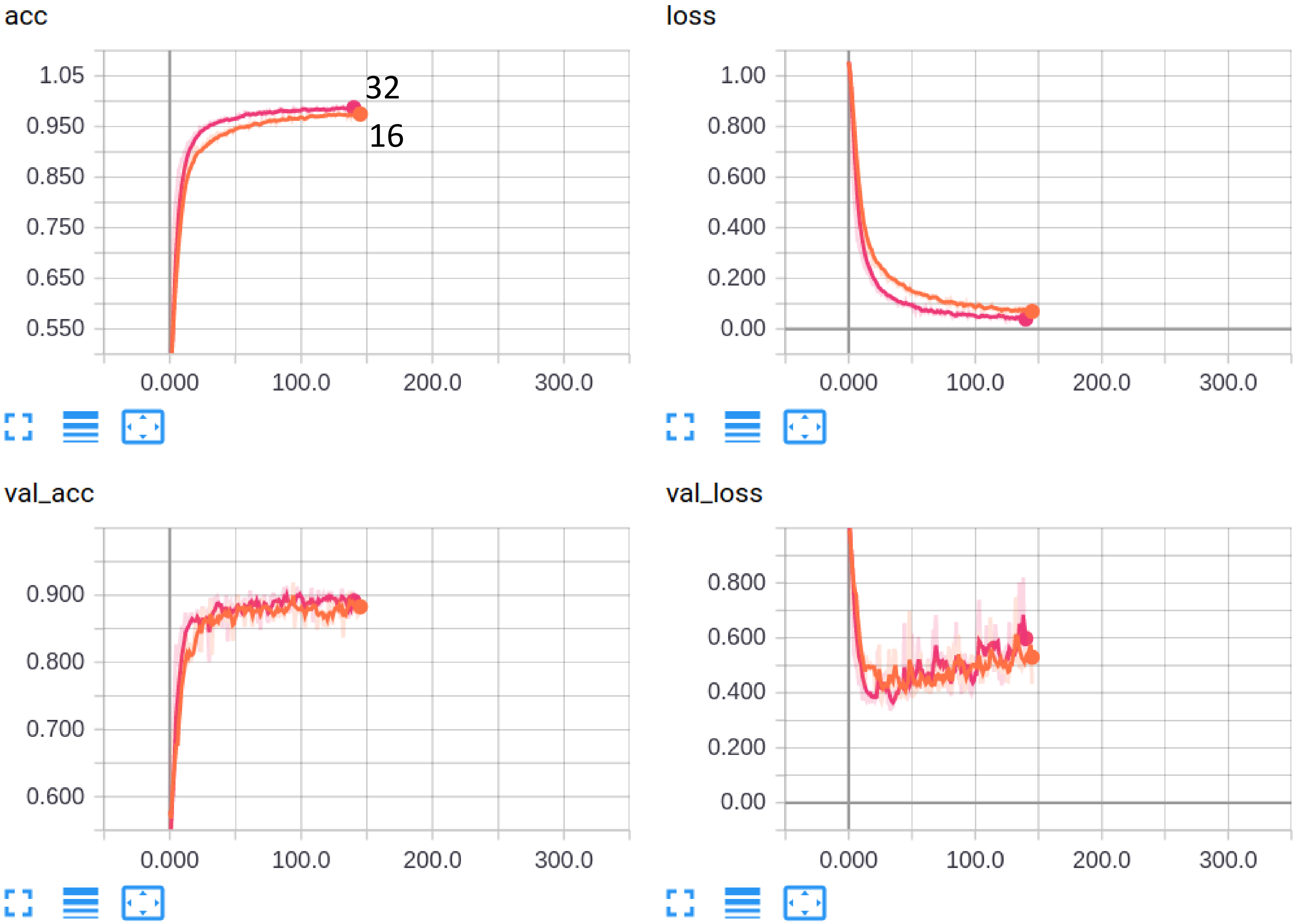


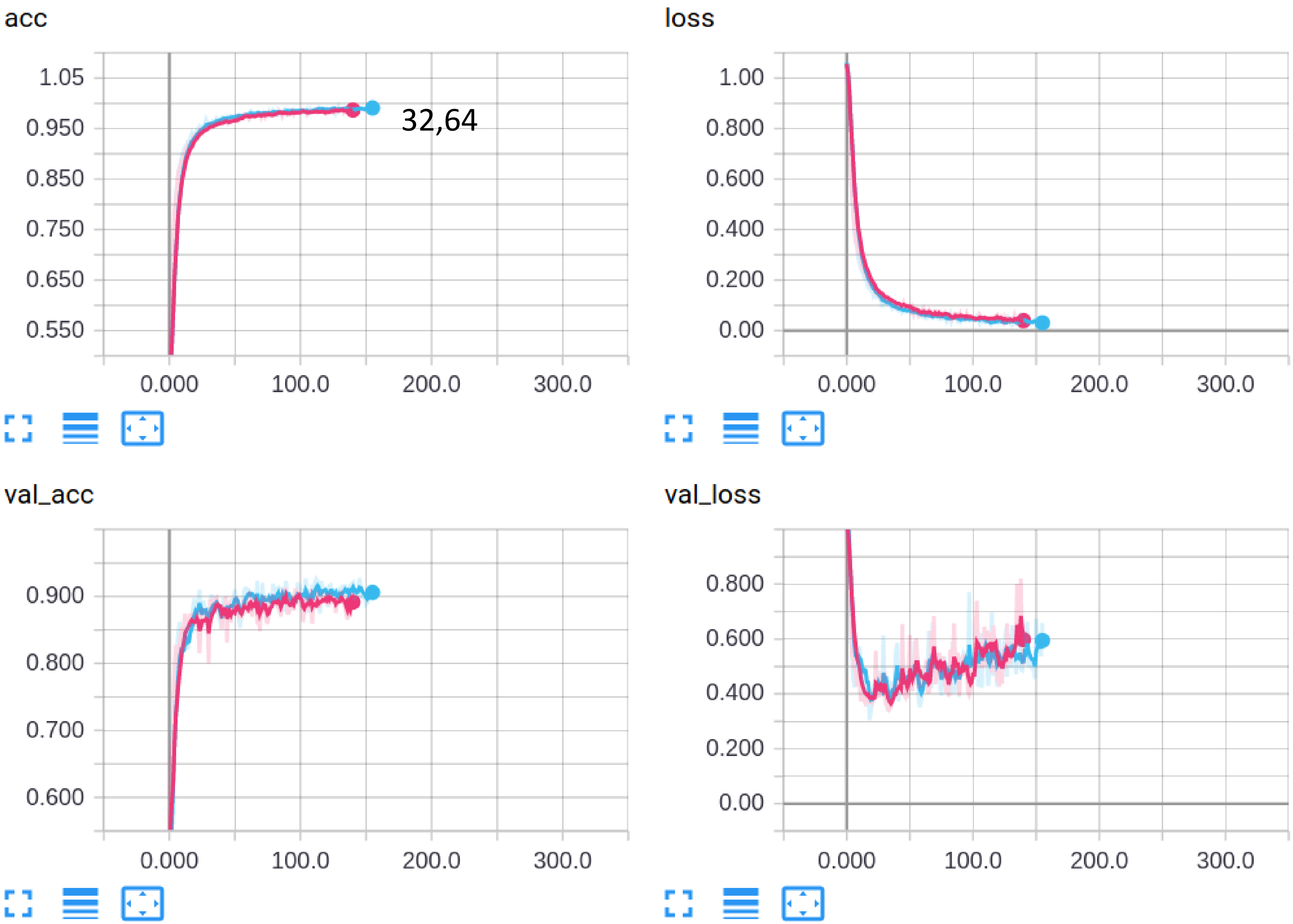
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_4 (Conv2D)            (None, 32, 32, 16)        448       
_________________________________________________________________
activation_6 (Activation)    (None, 32, 32, 16)        0         
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 16, 16, 16)        0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 16, 16, 32)        4640      
_________________________________________________________________
activation_7 (Activation)    (None, 16, 16, 32)        0         
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 8, 8, 32)          0         
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 8, 8, 64)          18496     
_________________________________________________________________
activation_8 (Activation)    (None, 8, 8, 64)          0         
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 4, 4, 64)          0         
_________________________________________________________________
average_pooling2d_2 (Average (None, 2, 2, 64)          0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 256)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896     
_________________________________________________________________
activation_9 (Activation)    (None, 128)               0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 3)                 387       
_________________________________________________________________
activation_10 (Activation)   (None, 3)                 0         
=================================================================
Total params: 56,867
Trainable params: 56,867
Non-trainable params: 0
_________________________________________________________________
```

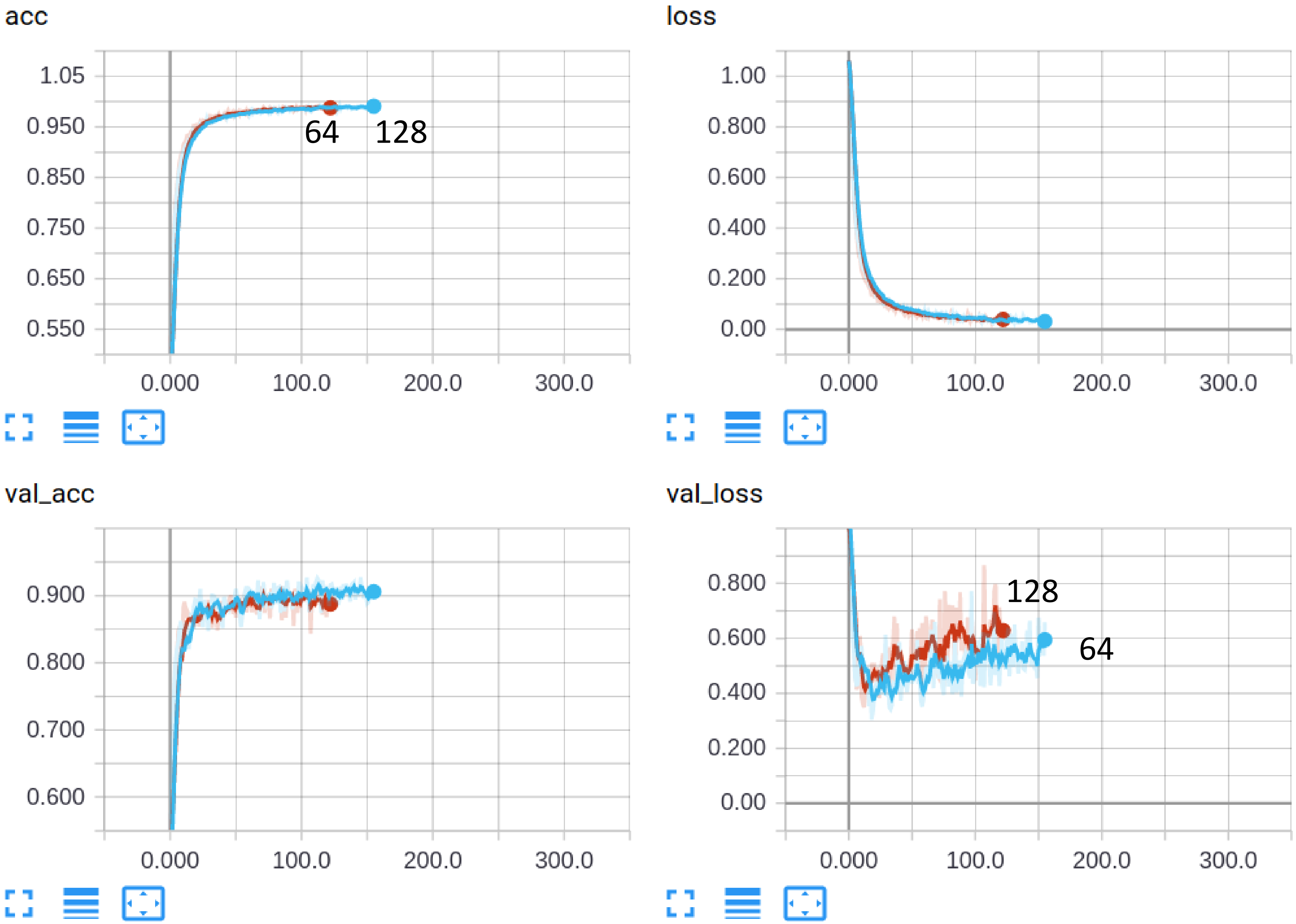

acc
1.05
0.950
0.850
0.750
0.650
0.550
0.000
100.0
200.0
300.0
32
16
loss
1.00
0.800
0.600
0.400
0.200
0.00
0.000
100.0
200.0
300.0
16
32
val_acc
0.900
0.800
0.700
0.600
0.000
100.0
200.0
300.0
val_loss
0.800
0.600
0.400
0.200
0.00
0.000
100.0
200.0
300.0

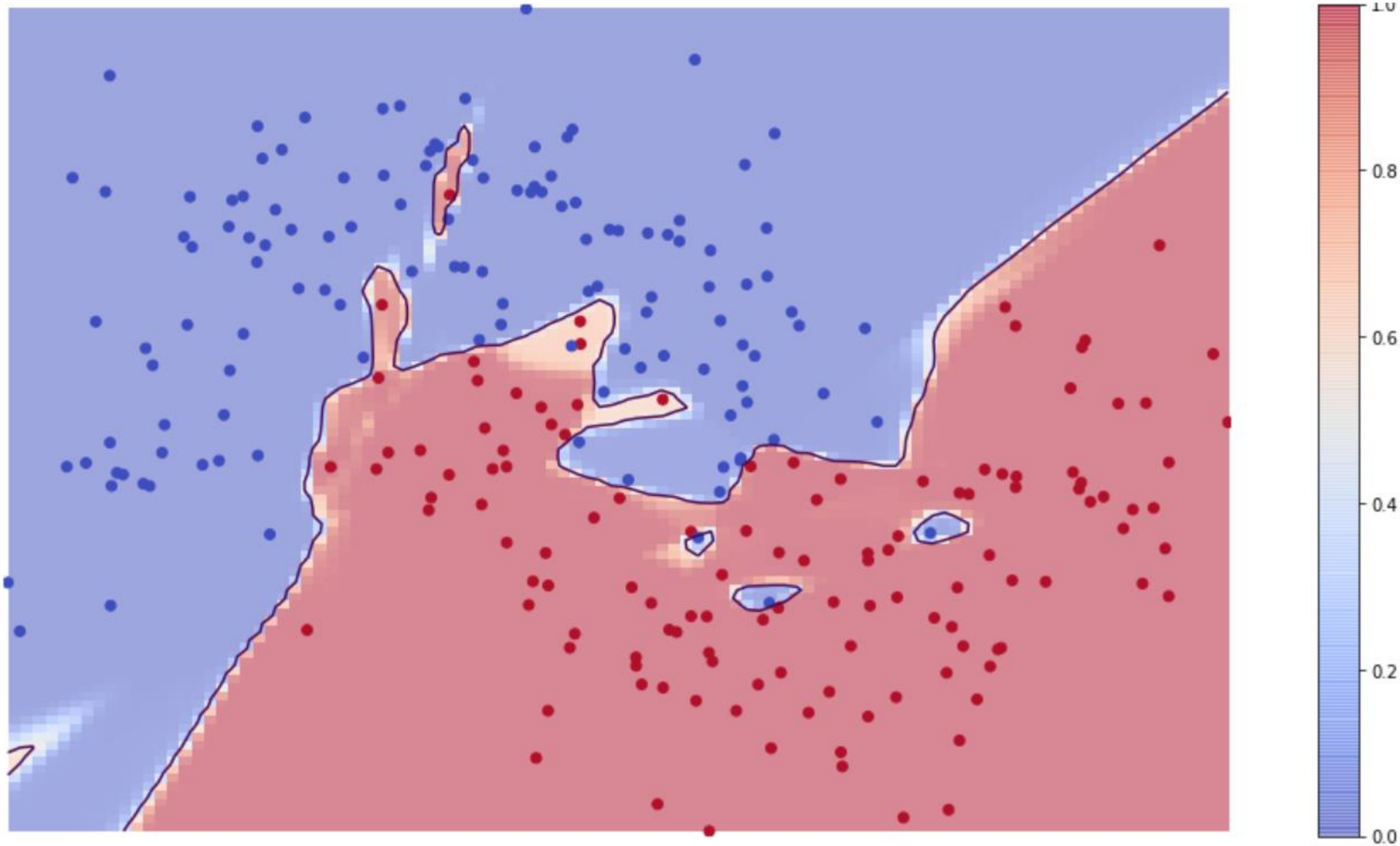
8, 16

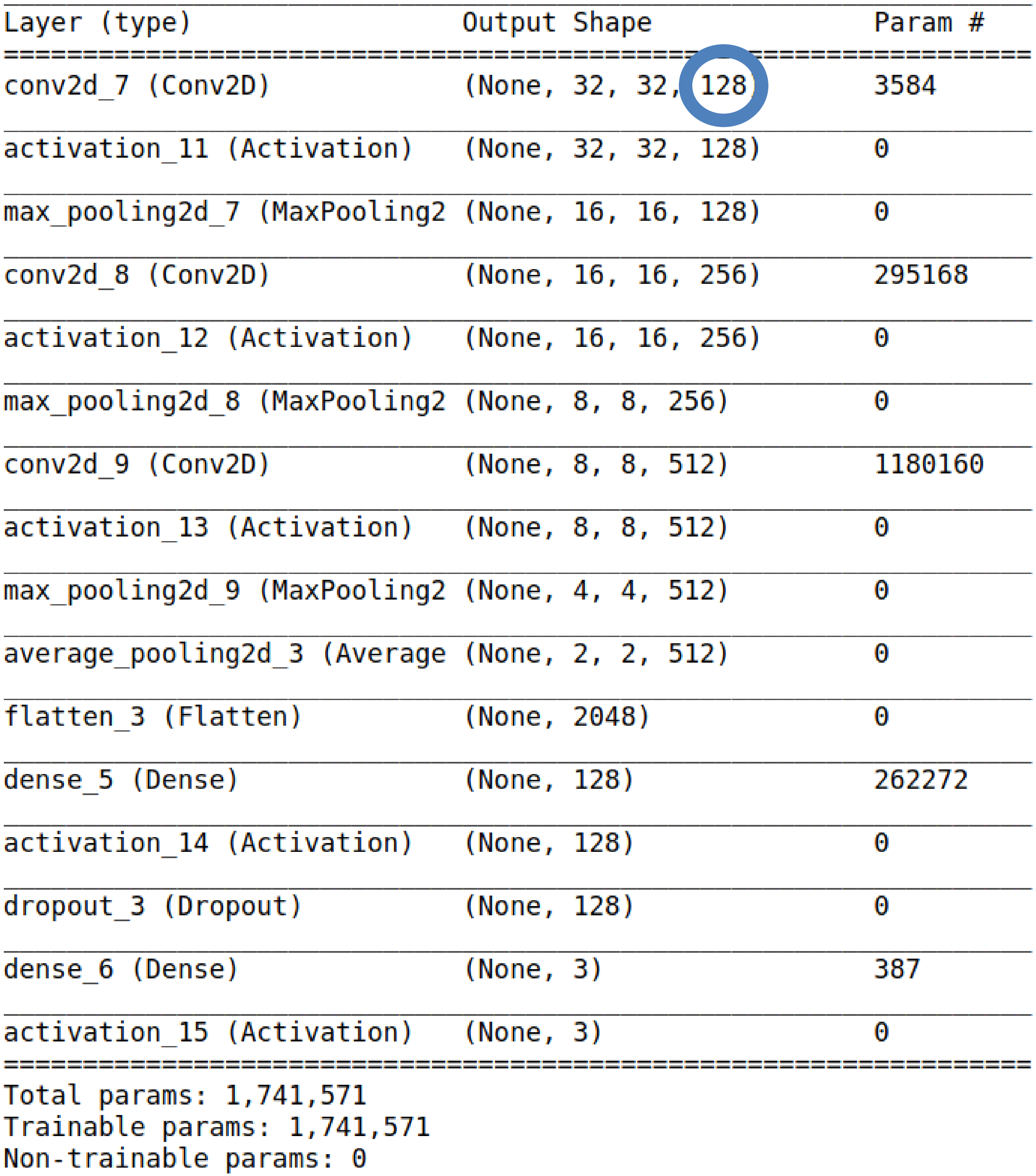

1.0
0.8
0.6
0.4
0.2
0.0


acc
1.05
0.950
0.850
0.750
0.650
0.550
0.000
100.0
200.0
300.0
32
16
loss
1.00
0.800
0.600
0.400
0.200
0.00
0.000
100.0
200.0
300.0
val_acc
0.900
0.800
0.700
0.600
0.000
100.0
200.0
300.0
val_loss
0.800
0.600
0.400
0.200
0.00
0.000
100.0
200.0
300.0

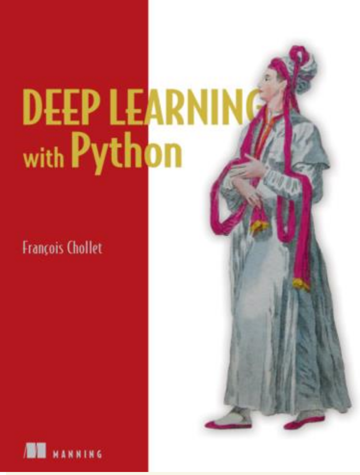

acc
1.05
0.950
0.850
0.750
0.650
0.550
0.000
100.0
200.0
300.0
32,64
loss
1.00
0.800
0.600
0.400
0.200
0.00
0.000
100.0
200.0
300.0
val_acc
0.900
0.800
0.700
0.600
0.000
100.0
200.0
300.0
val_loss
0.800
0.600
0.400
0.200
0.00
0.000
100.0
200.0
300.0


acc
1.05
0.950
0.850
0.750
0.650
0.550
0.000
100.0
200.0
300.0
64
128
loss
1.00
0.800
0.600
0.400
0.200
0.00
0.000
100.0
200.0
300.0
val_acc
0.900
0.800
0.700
0.600
0.000
100.0
200.0
300.0
val_loss
0.800
0.600
0.400
0.200
0.00
0.000
100.0
200.0
300.0
128
64

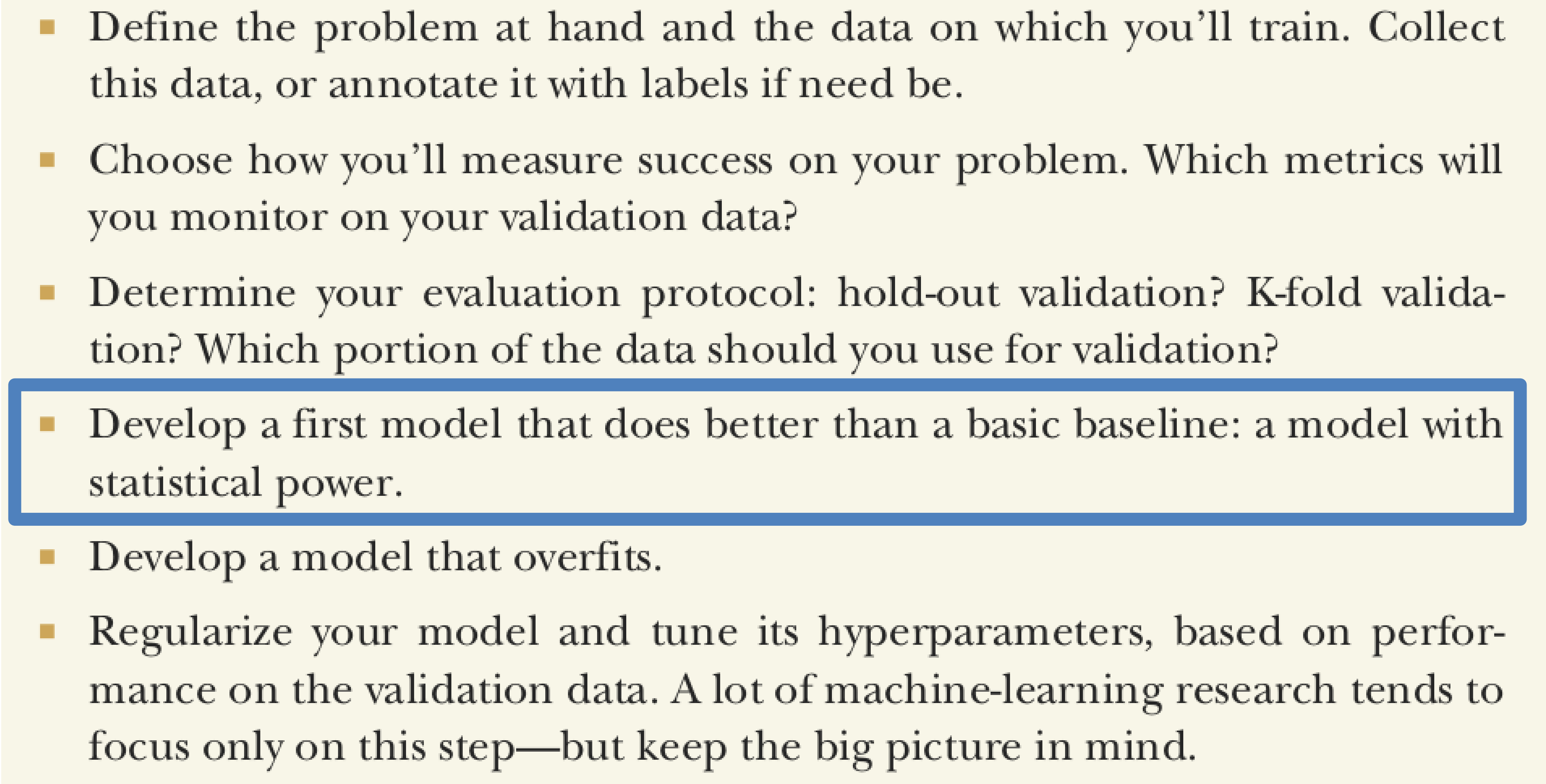

1.0
0.8
0.6
0.4
0.2
0.0



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_7 (Conv2D)            (None, 32, 32, 128)       3584      
_________________________________________________________________
activation_11 (Activation)   (None, 32, 32, 128)       0         
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 16, 16, 128)       0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 16, 16, 256)       295168    
_________________________________________________________________
activation_12 (Activation)   (None, 16, 16, 256)       0         
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 8, 8, 256)         0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 8, 8, 512)         1180160   
_________________________________________________________________
activation_13 (Activation)   (None, 8, 8, 512)         0         
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 4, 4, 512)         0         
_________________________________________________________________
average_pooling2d_3 (Average (None, 2, 2, 512)         0         
_________________________________________________________________
flatten_3 (Flatten)          (None, 2048)              0         
_________________________________________________________________
dense_5 (Dense)              (None, 128)               262272    
_________________________________________________________________
activation_14 (Activation)   (None, 128)               0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_6 (Dense)              (None, 3)                 387       
_________________________________________________________________
activation_15 (Activation)   (None, 3)                 0         
=================================================================
Total params: 1,741,571
Trainable params: 1,741,571
Non-trainable params: 0
```



The ML Pipeline (Chollet)

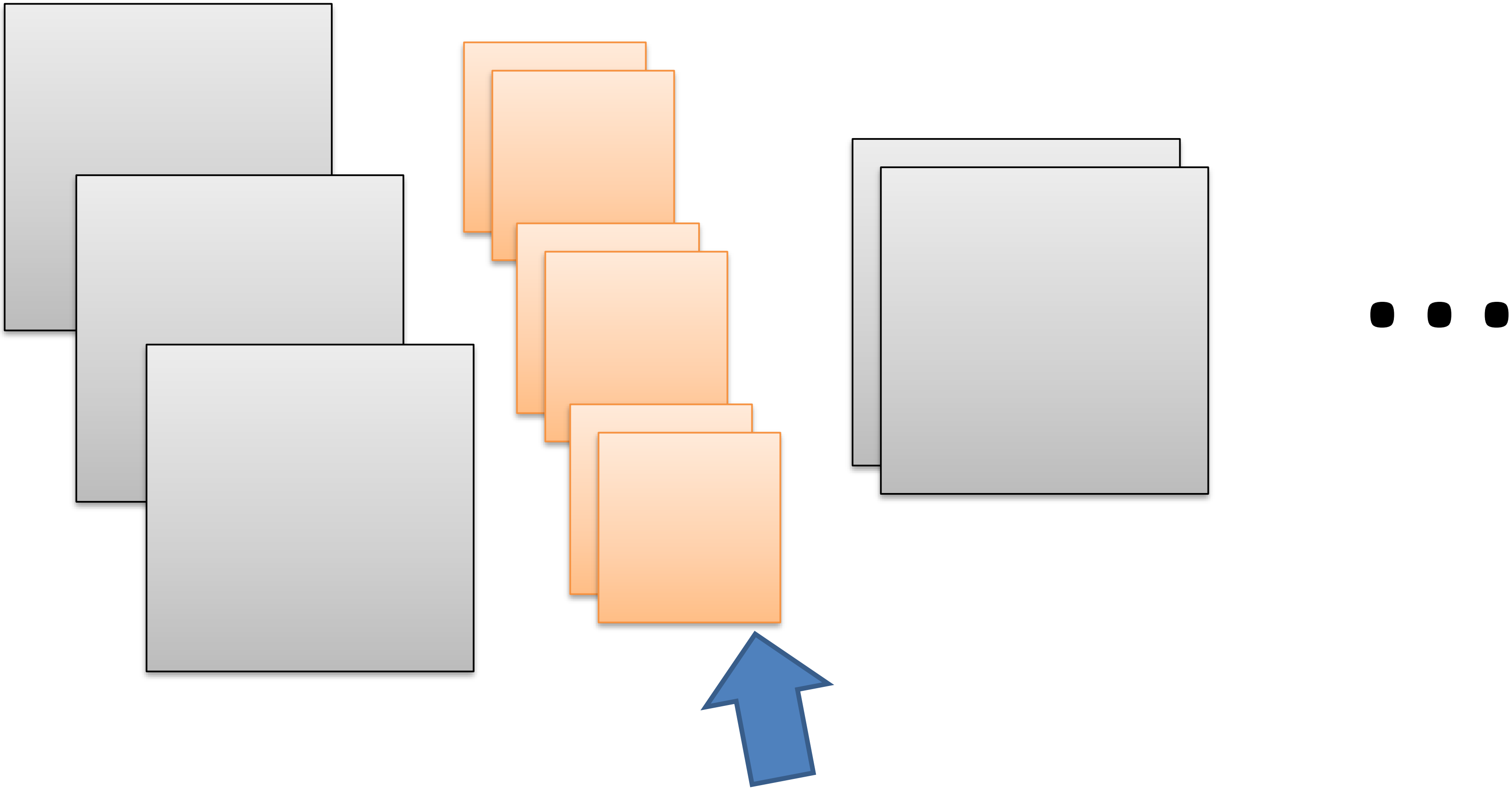


- Define the problem at hand and the data on which you'll train. Collect this data, or annotate it with labels if need be.
- Choose how you'll measure success on your problem. Which metrics will you monitor on your validation data?
- Determine your evaluation protocol: hold-out validation? K-fold validation? Which portion of the data should you use for validation?
- Develop a first model that does better than a basic baseline: a model with statistical power.
- Develop a model that overfits.
- Regularize your model and tune its hyperparameters, based on performance on the validation data. A lot of machine-learning research tends to focus only on this step—but keep the big picture in mind.

Rock Paper Scissors

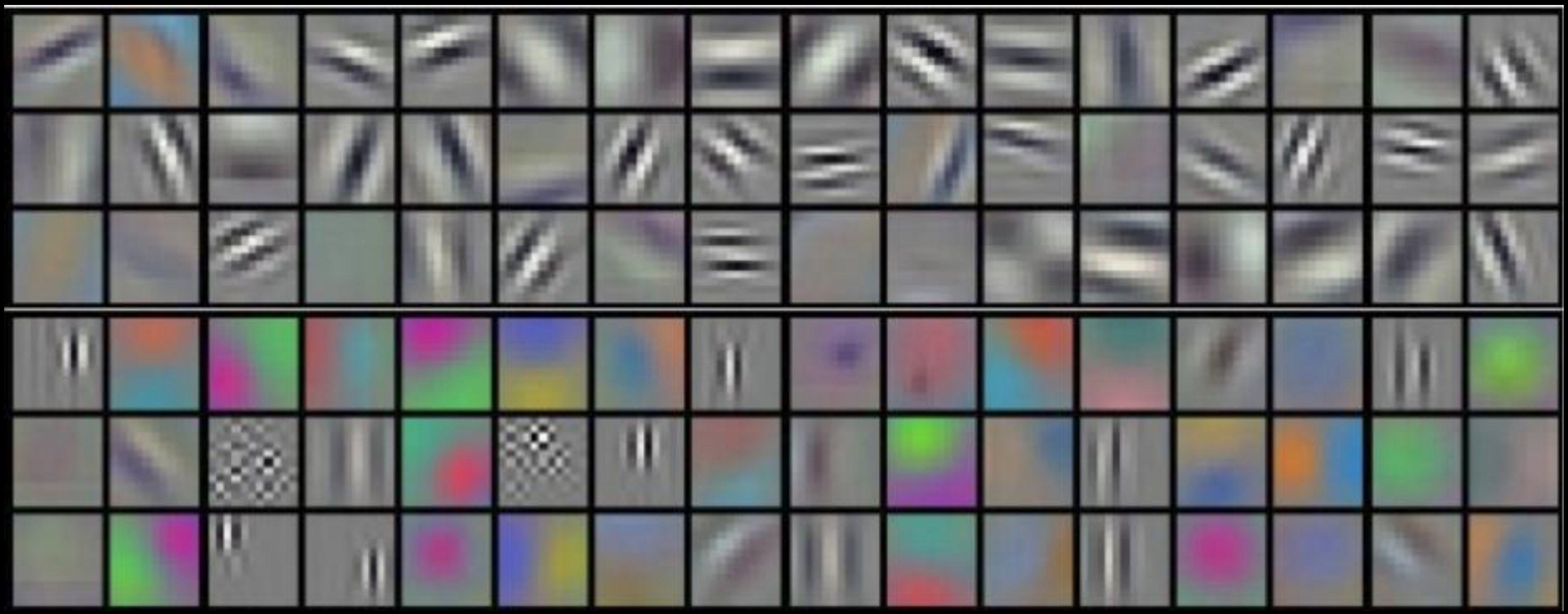
2. VISUALIZATION TECHNIQUES

Visualizing the weights of the net



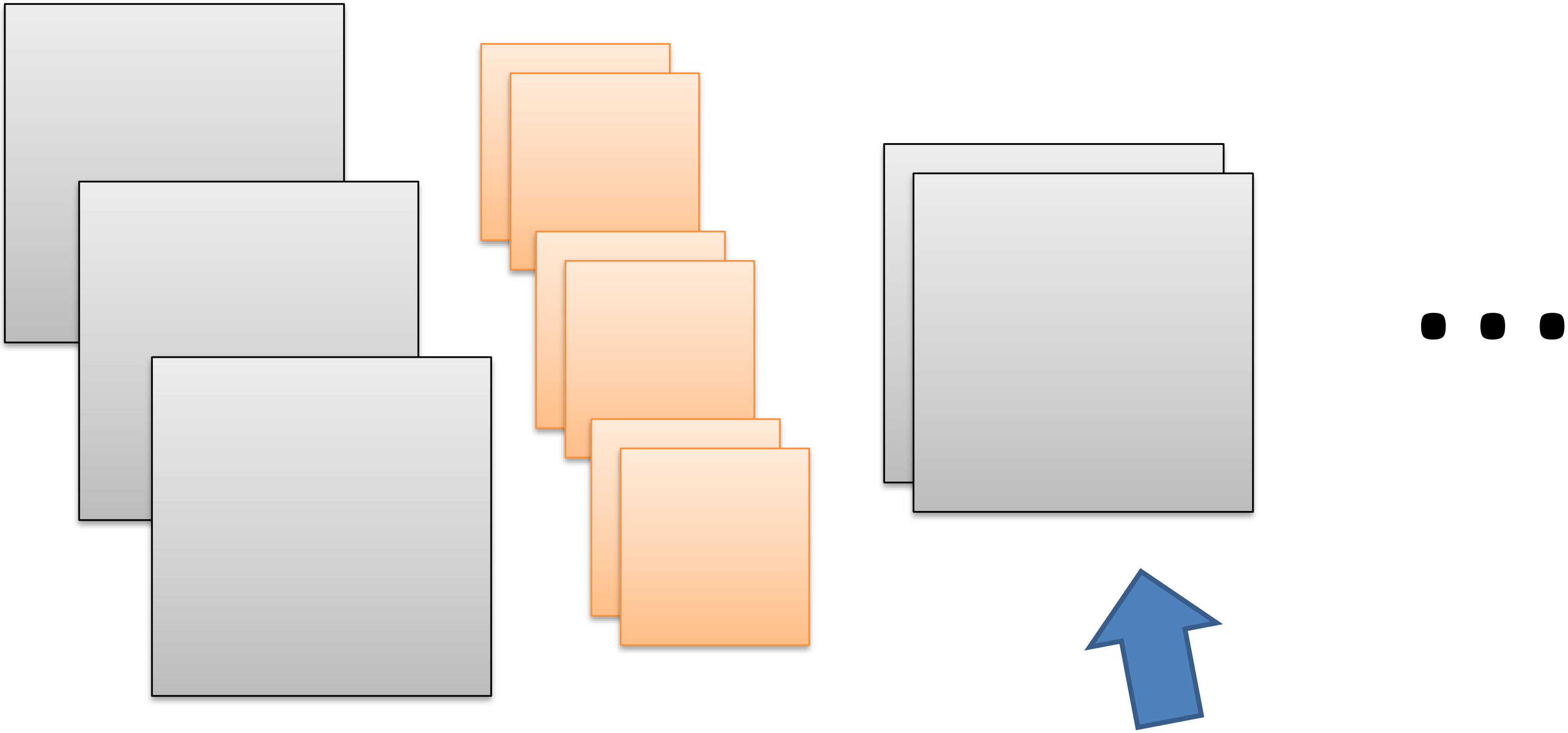
We want to see this

Visualizing the weights of the net



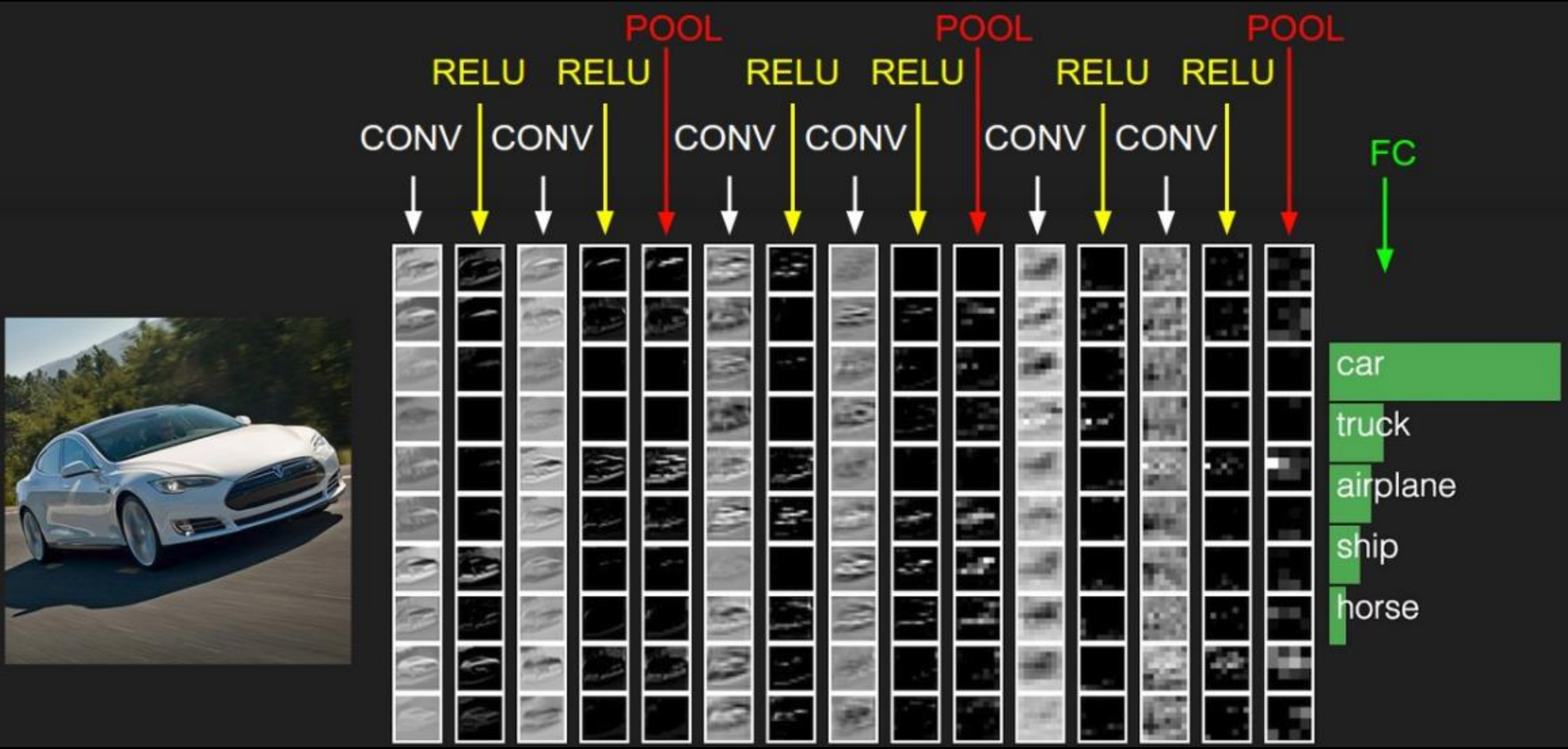
11x11x3 filters (visualized in RGB) in the first convolutional layers

Visualizing the activations of intermediate layers

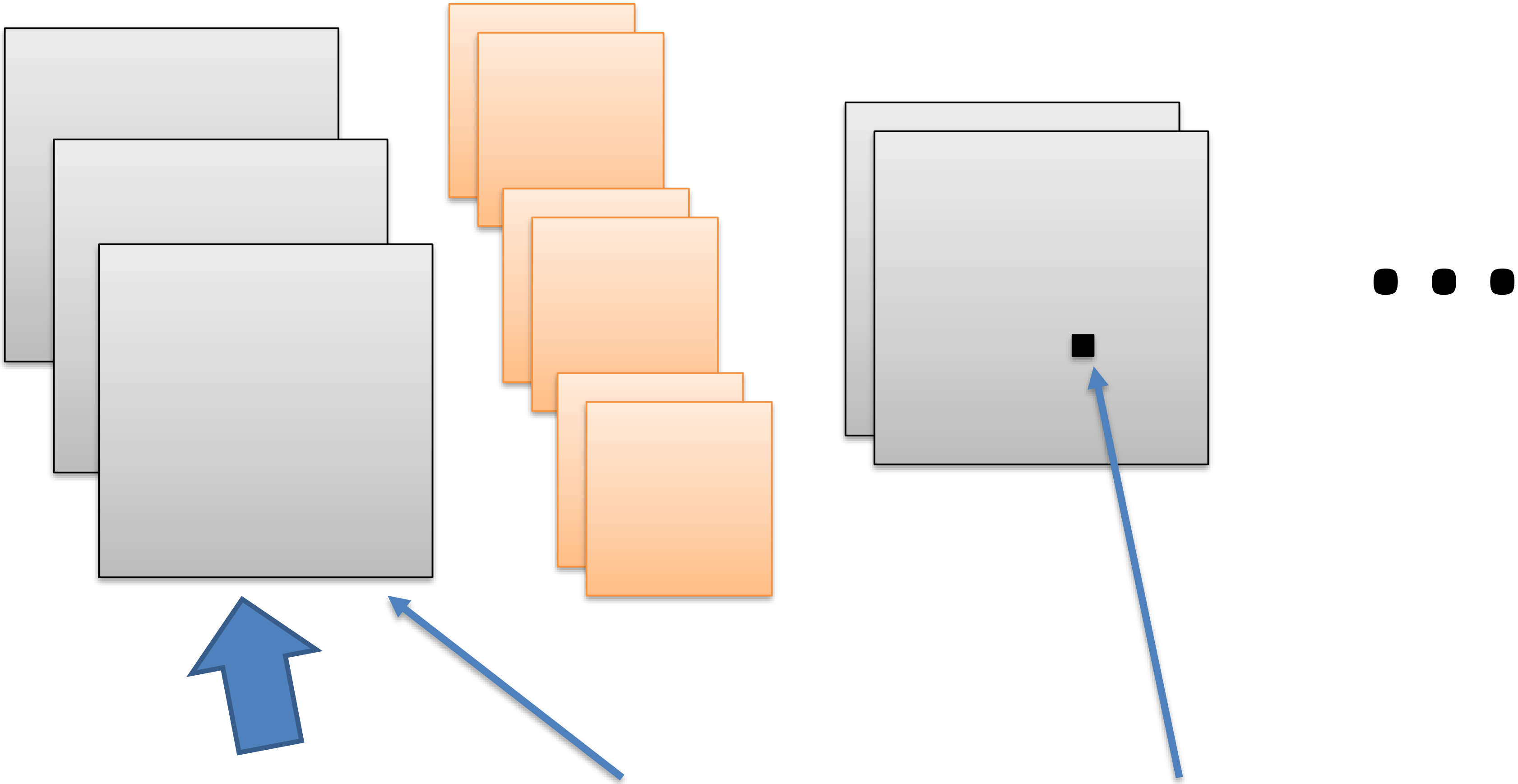


We want to see this

Visualizing the activations of intermediate layers

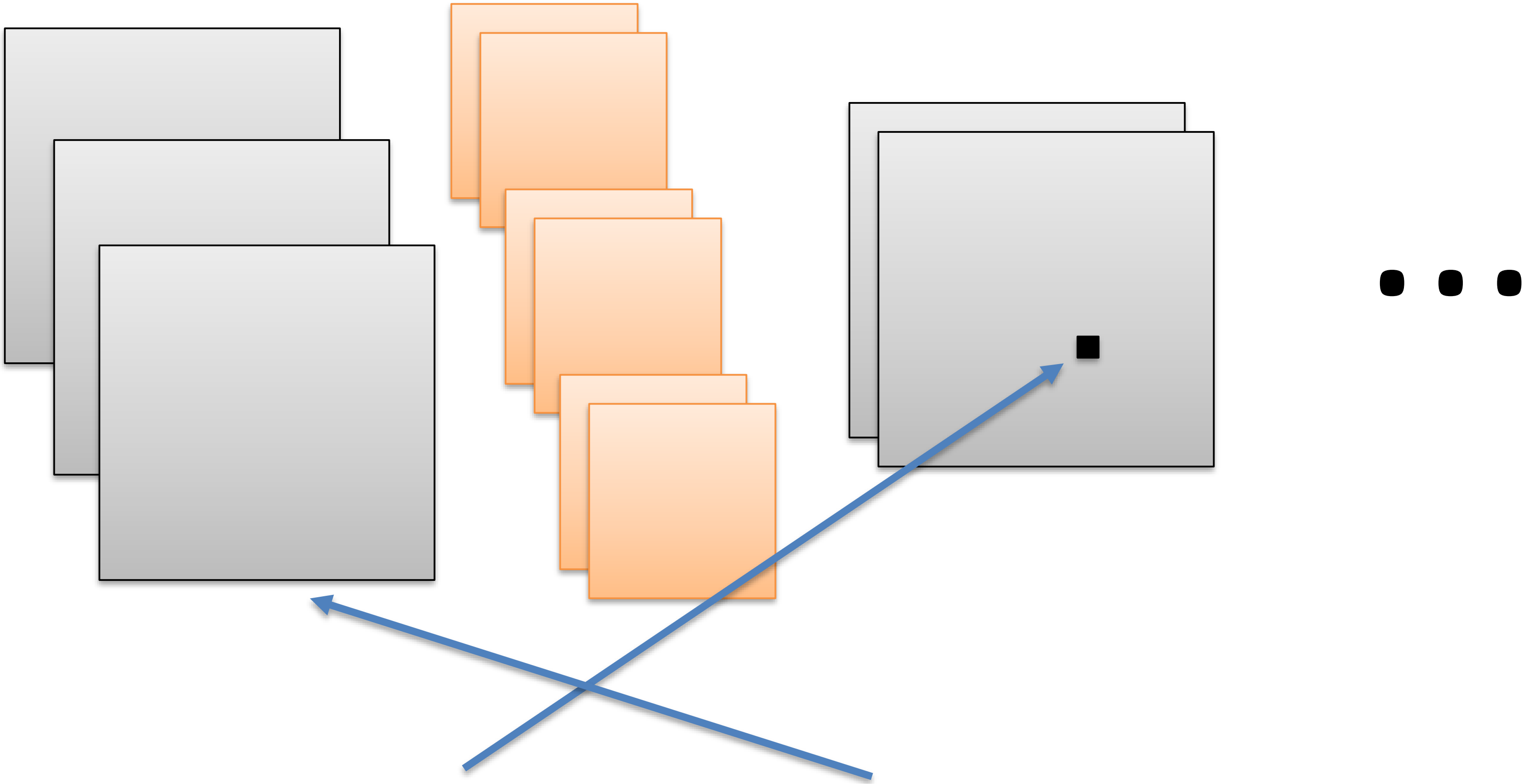


Visualizing the input that maximally activates some neurons



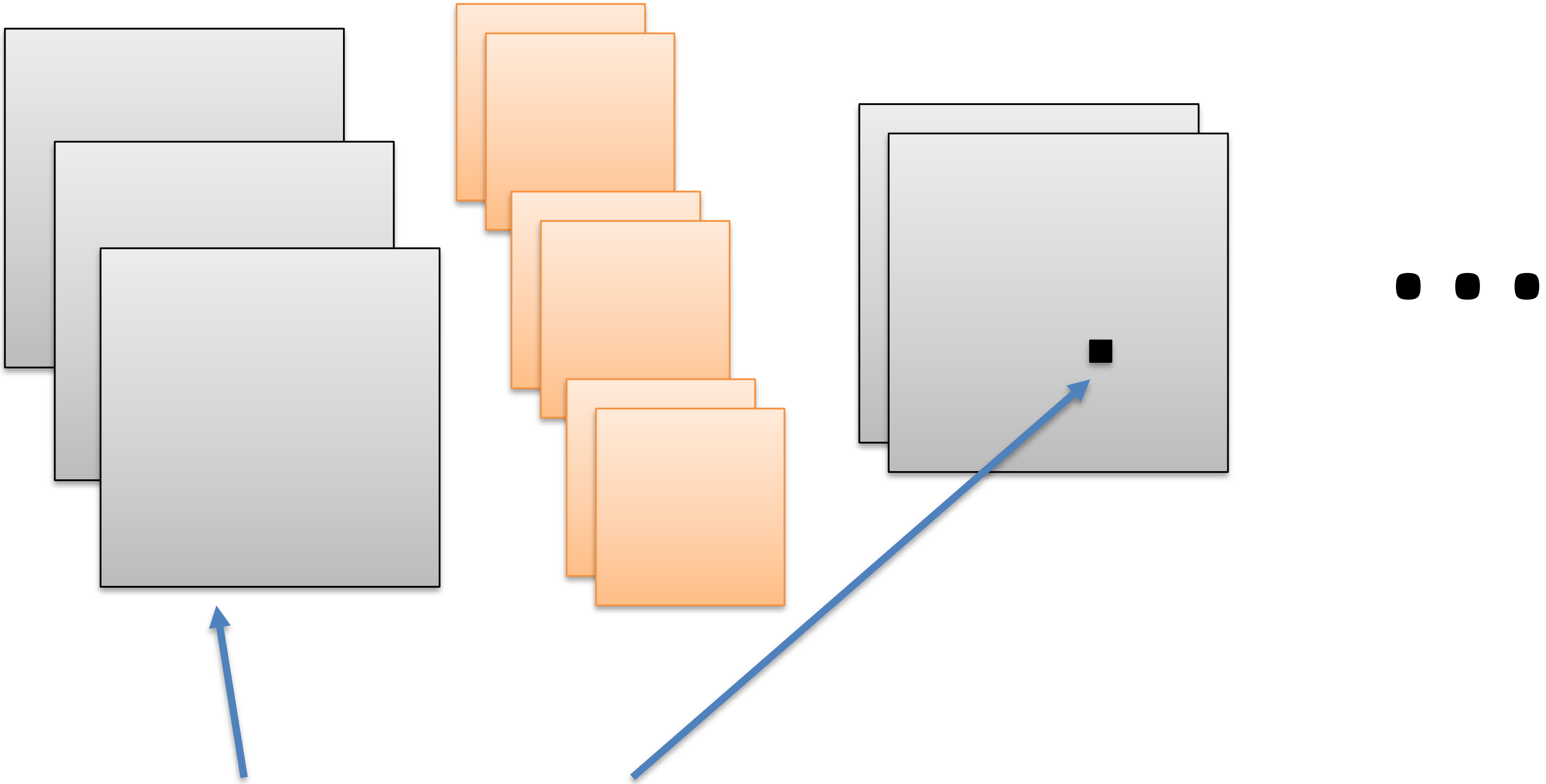
We want to compute (and see) the input that maximally activates this guy

Step 1



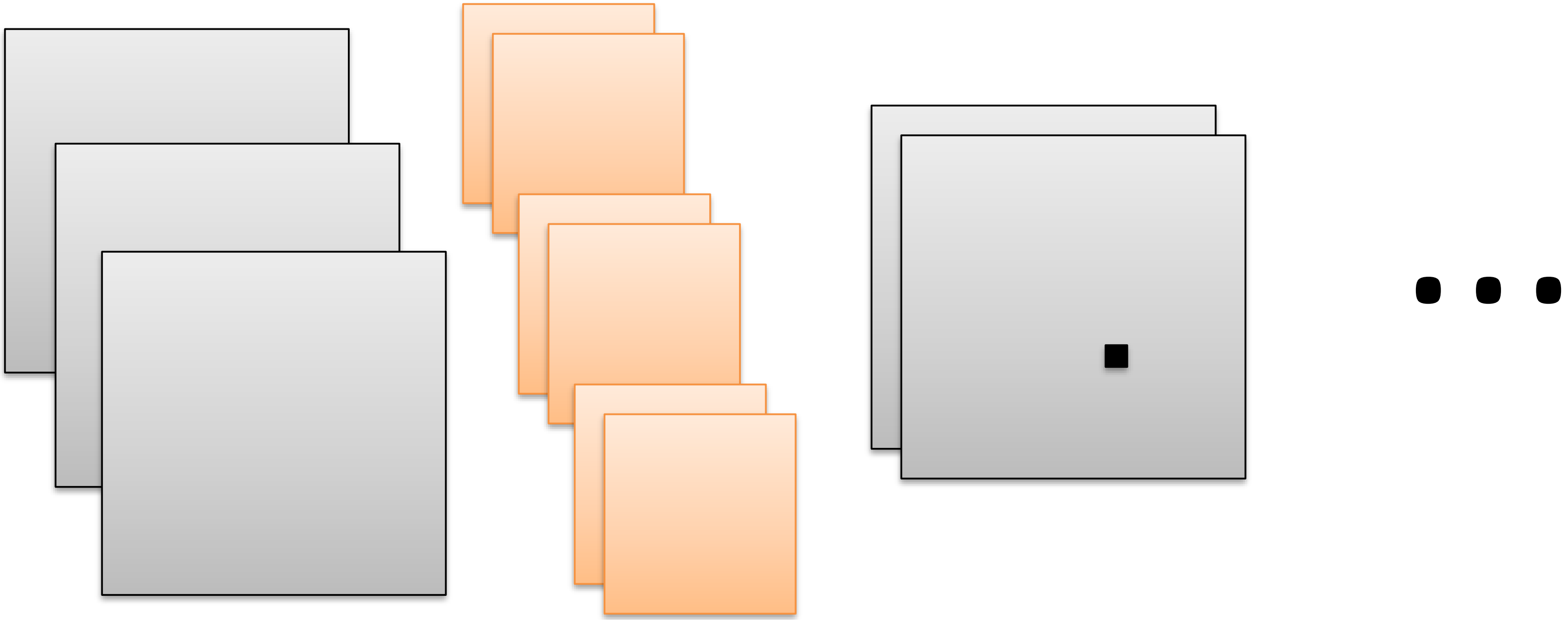
Compute the gradient of this with respect to the input

Step 2

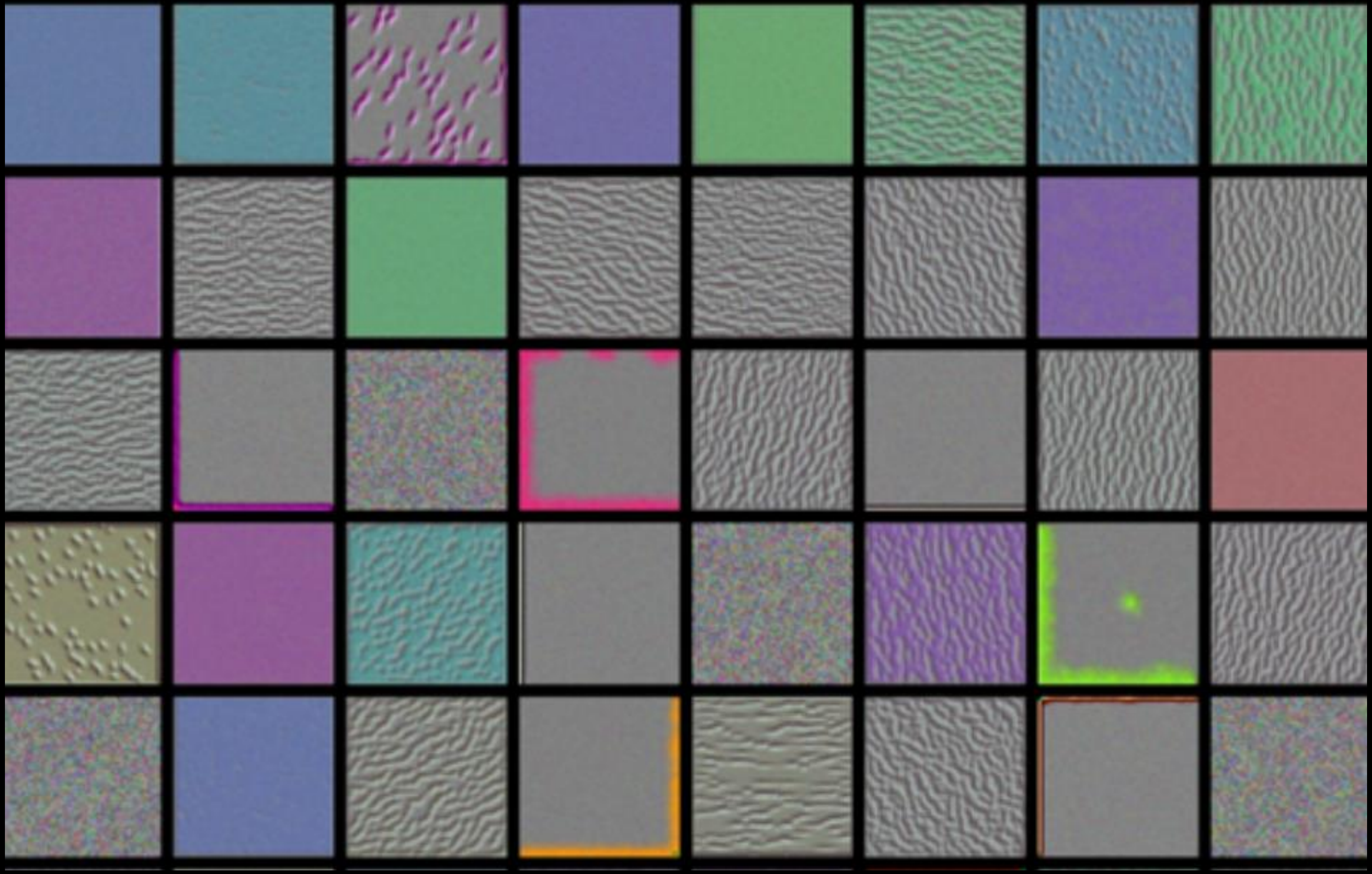


Nudge the input accordingly: our guy will increase its activation

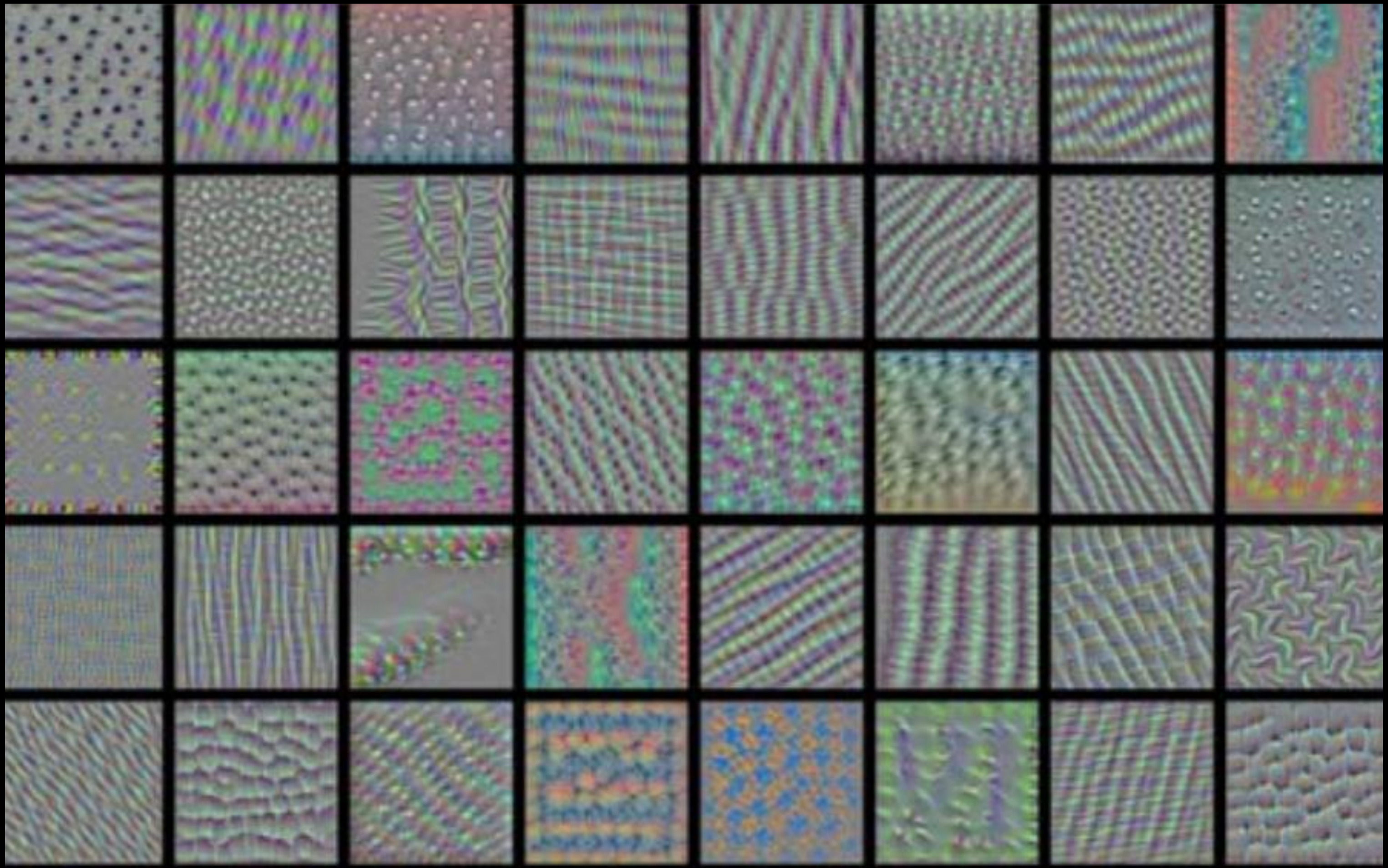
Goto step 1



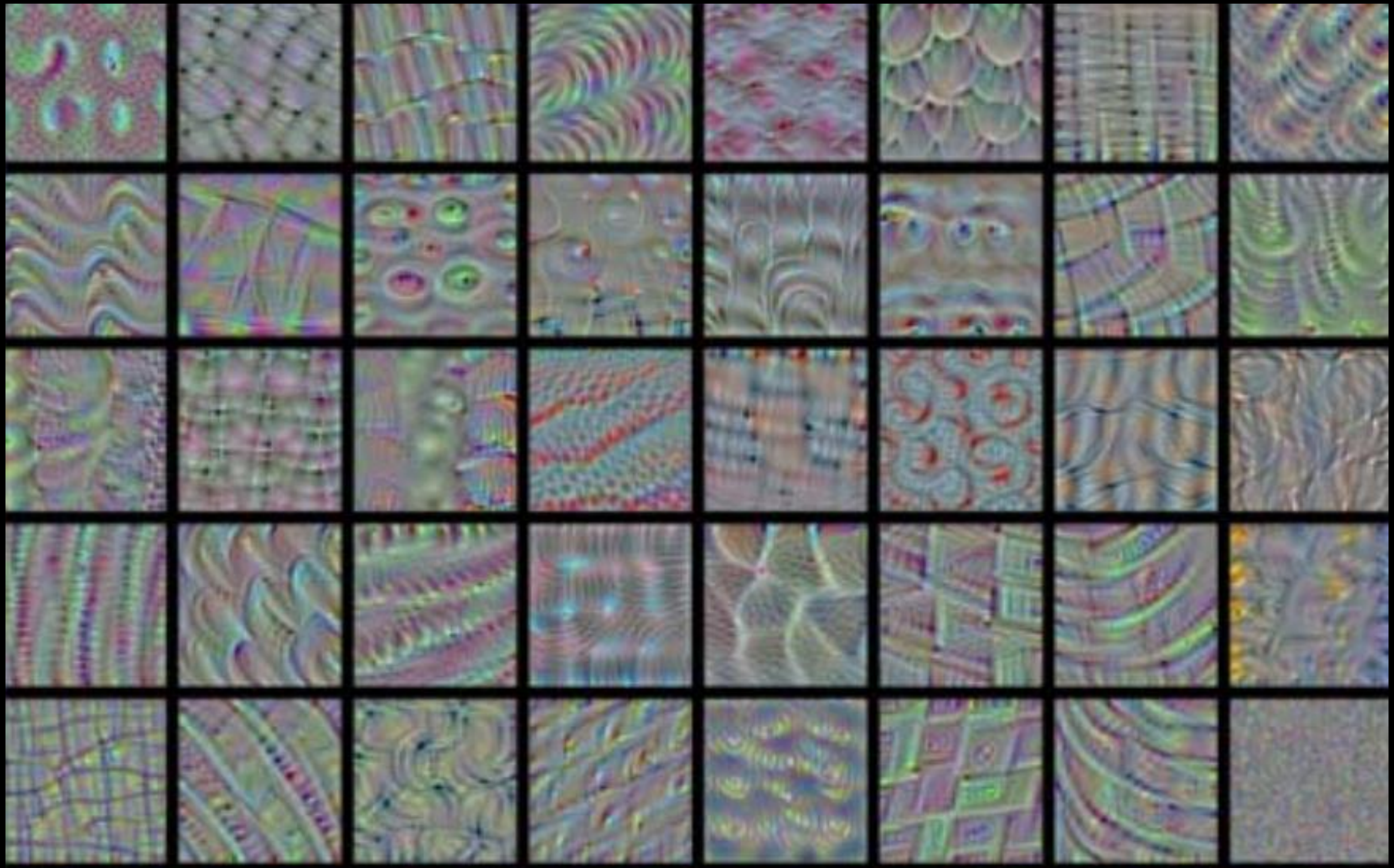
Shallow layers respond to fine, **low-level** patterns

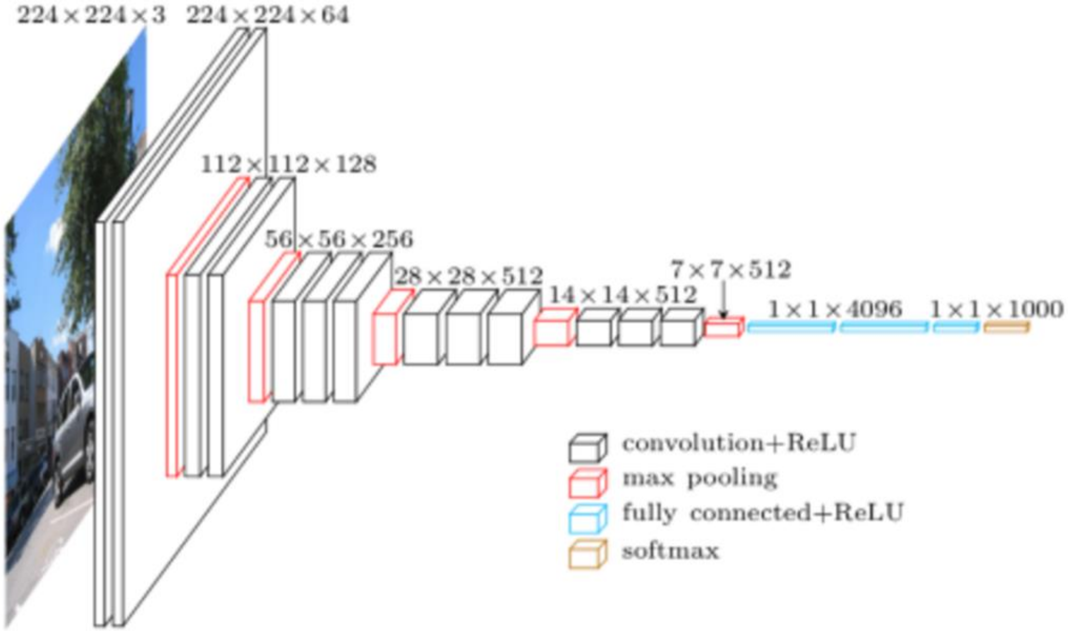


Intermediate layers ...



Deep layers respond to complex, **high-level** patterns



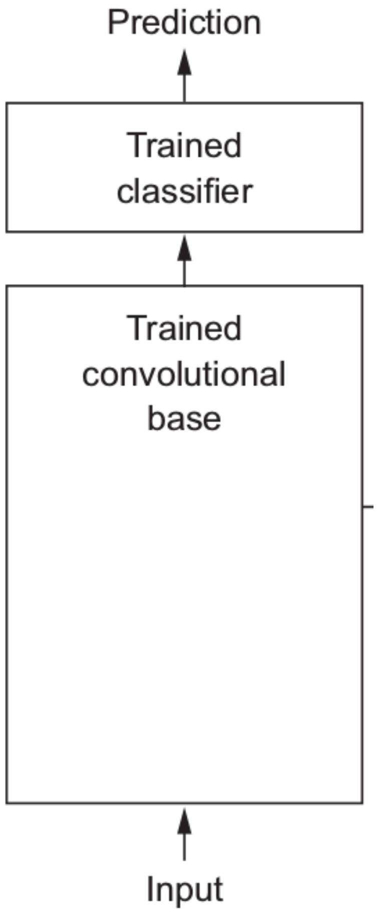


Stand on the shoulder of giants

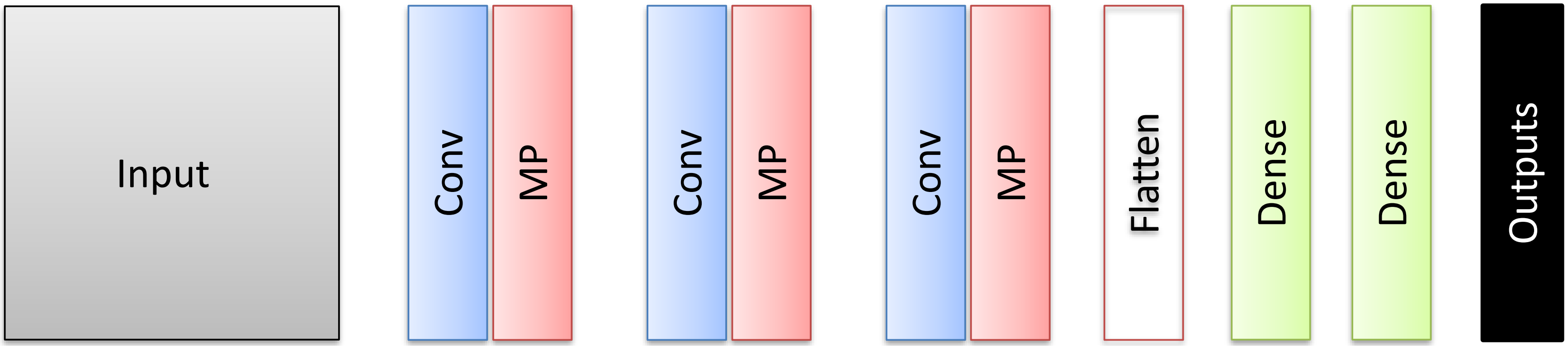
USING PRETRAINED WEIGHTS

Using pretrained weights

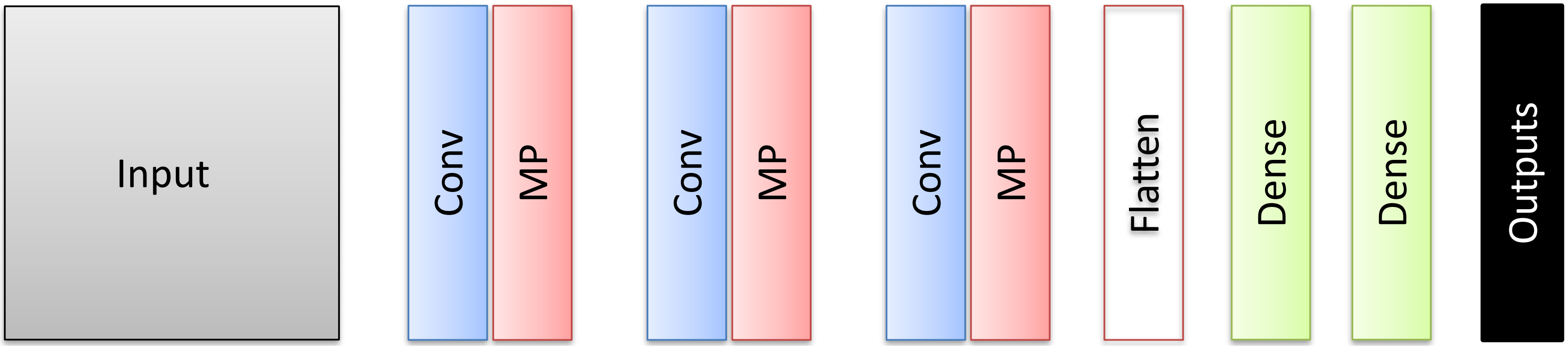
Step 1



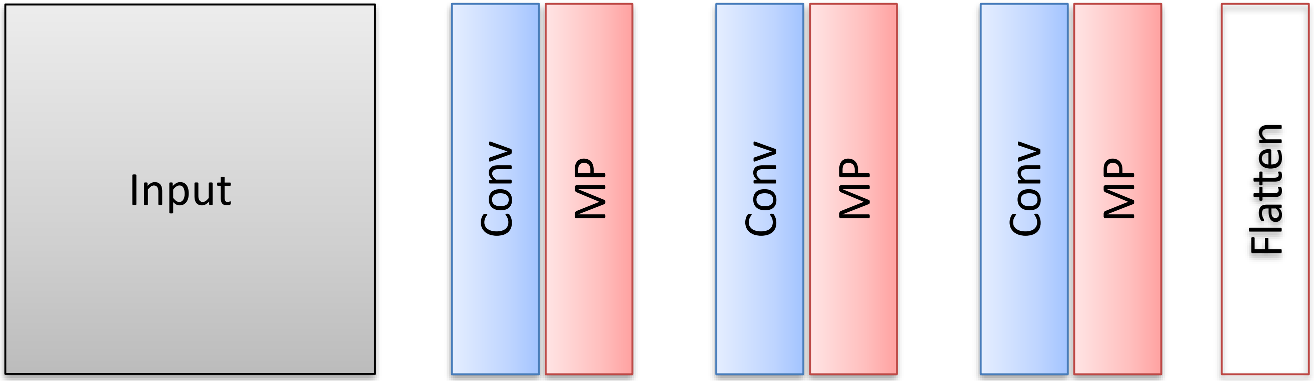
Option 1



Option 1



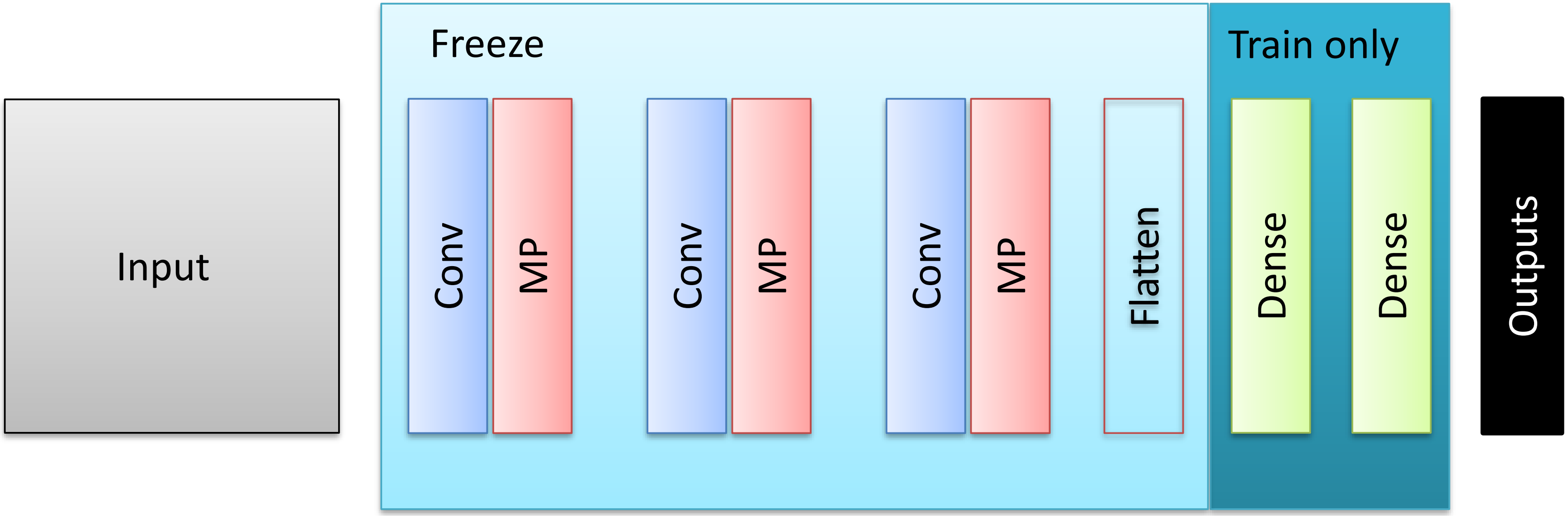
Option 1



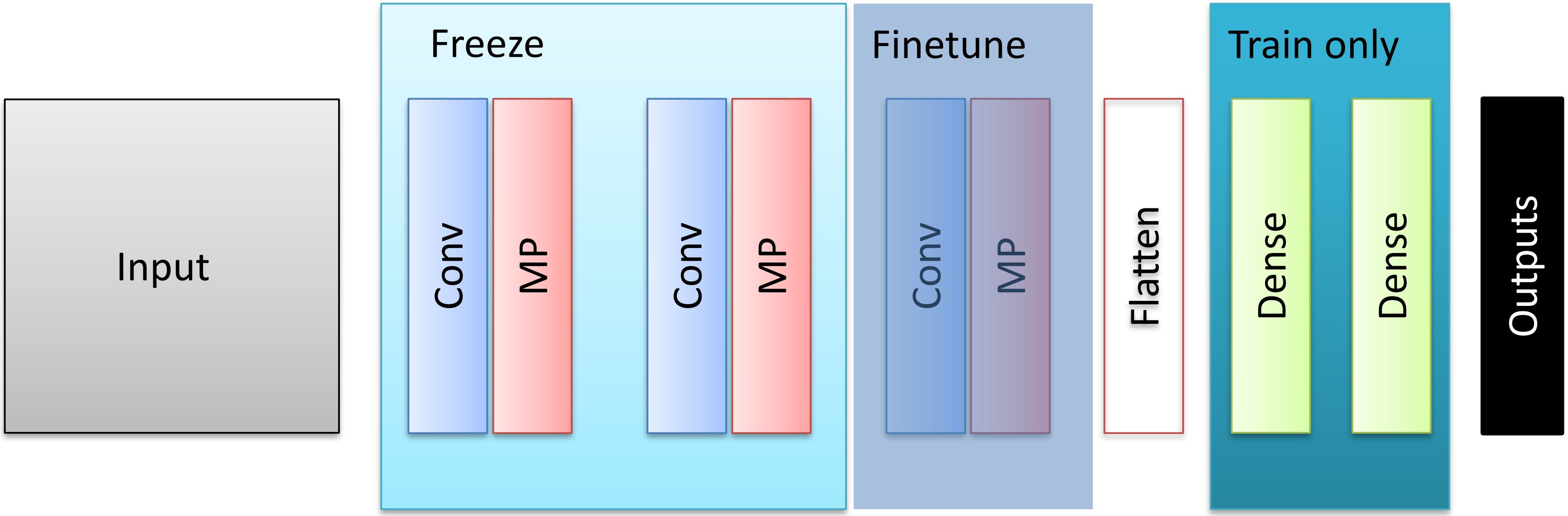
Save these features for the whole training and testing datasets.

Then, train a new classifier that uses these features as input

Option 2

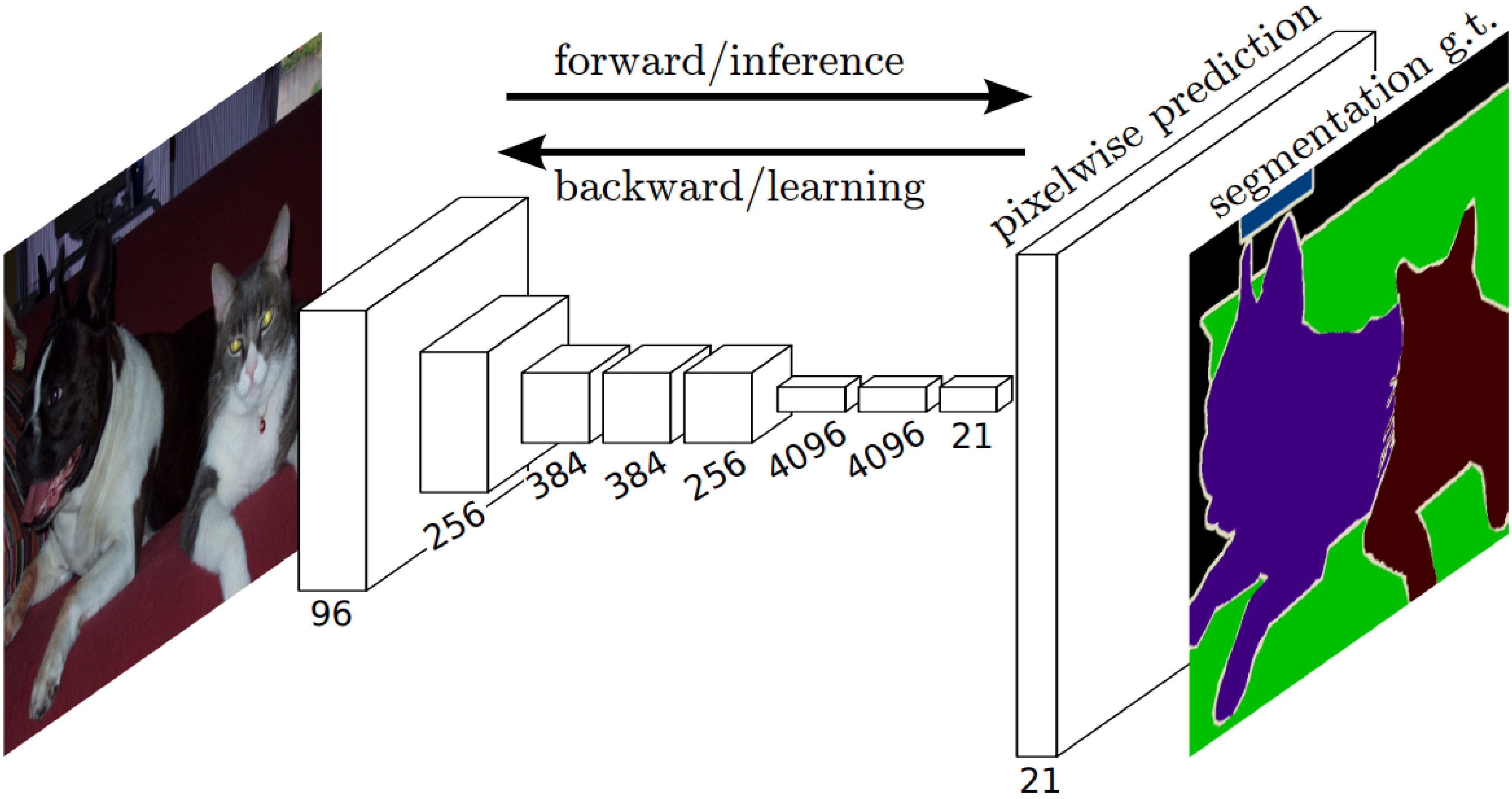

Input
Freeze
Conv
MP
Conv
MP
Conv
MP
Flatten
Train only
Dense
Dense
Outputs

Option 3

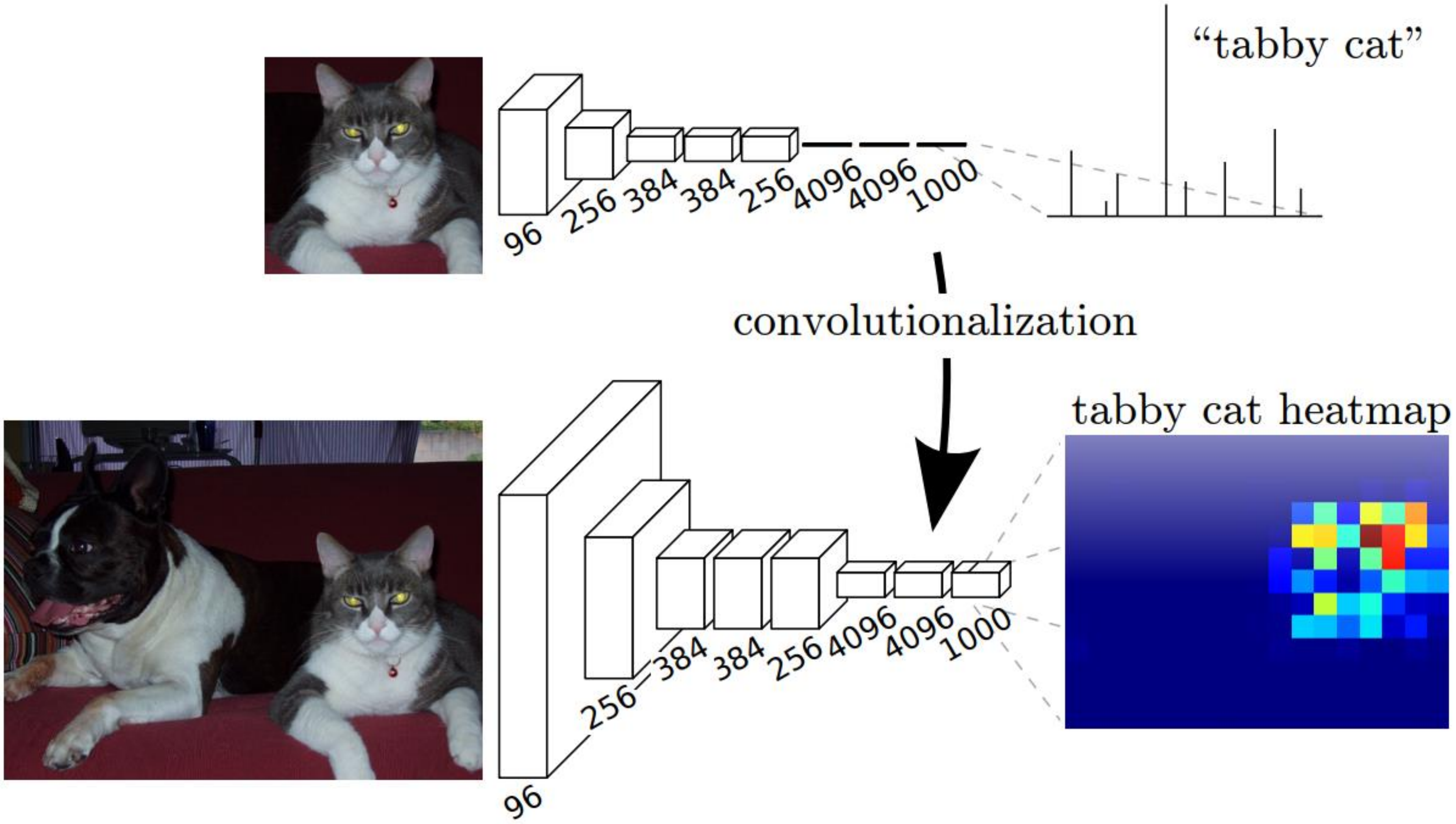

Input
Freeze
Conv
MP
Conv
MP
Finetune
Conv
MP
Flatten
Train only
Dense
Dense
Outputs

A MILE-HIGH OVERVIEW OF FULLY CONVOLUTIONAL NETWORKS FOR SEGMENTATION

Overall idea

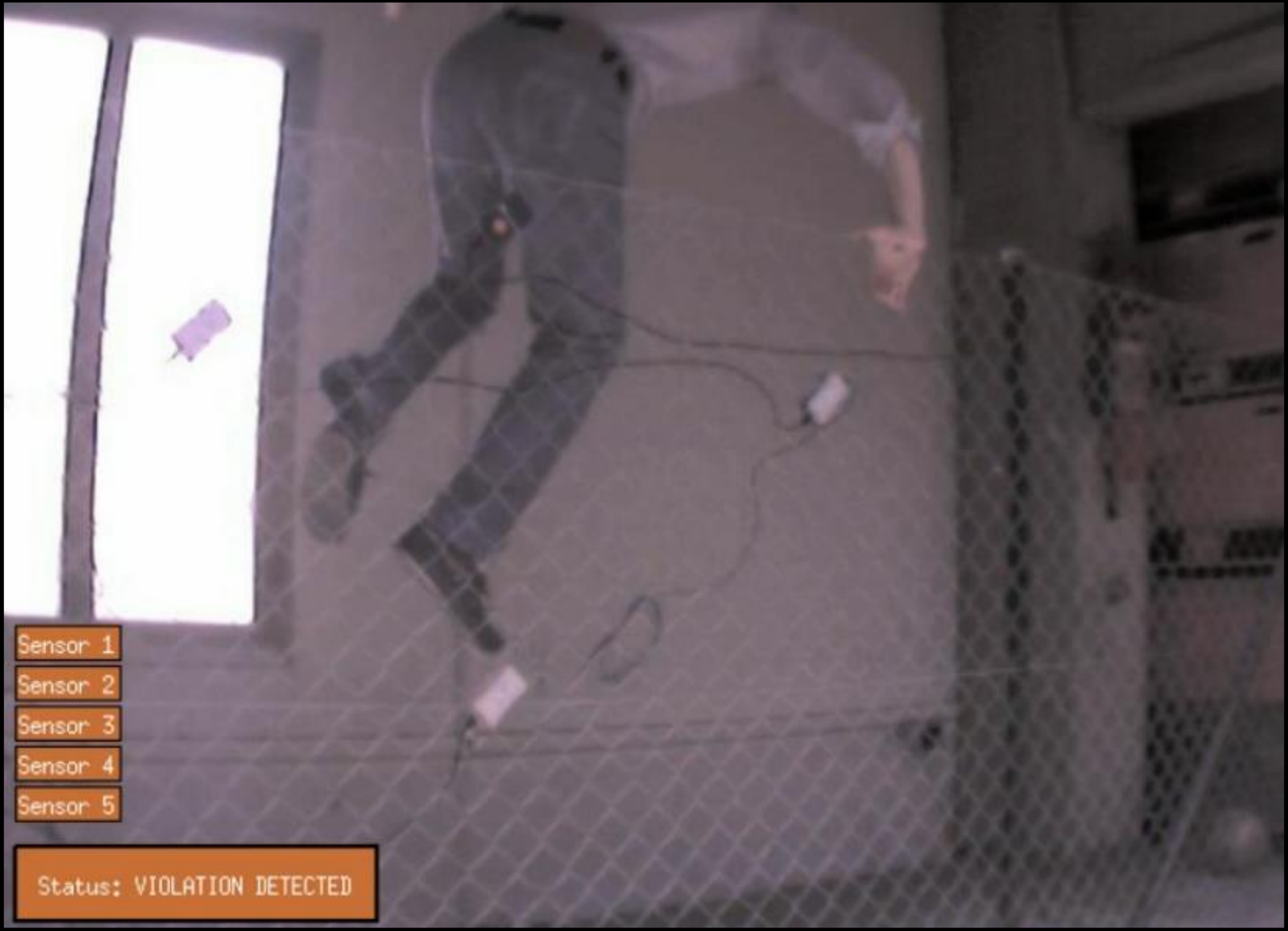


“Convolutionalization” of a dense layer



SOME POSSIBLE PROJECTS

Deep Learning on vibration data for detecting fence violations



Deep Learning on wearable sensor data for robot control

USI/SUPSI
IDSIA

Landing a Drone with Pointing Gestures

Boris Gromov, Luca M. Gambardella, Alessandro Giusti

Dalle Molle Institute for Artificial Intelligence (IDSIA)
Lugano, Switzerland

Learning to predict errors in weather forecasts