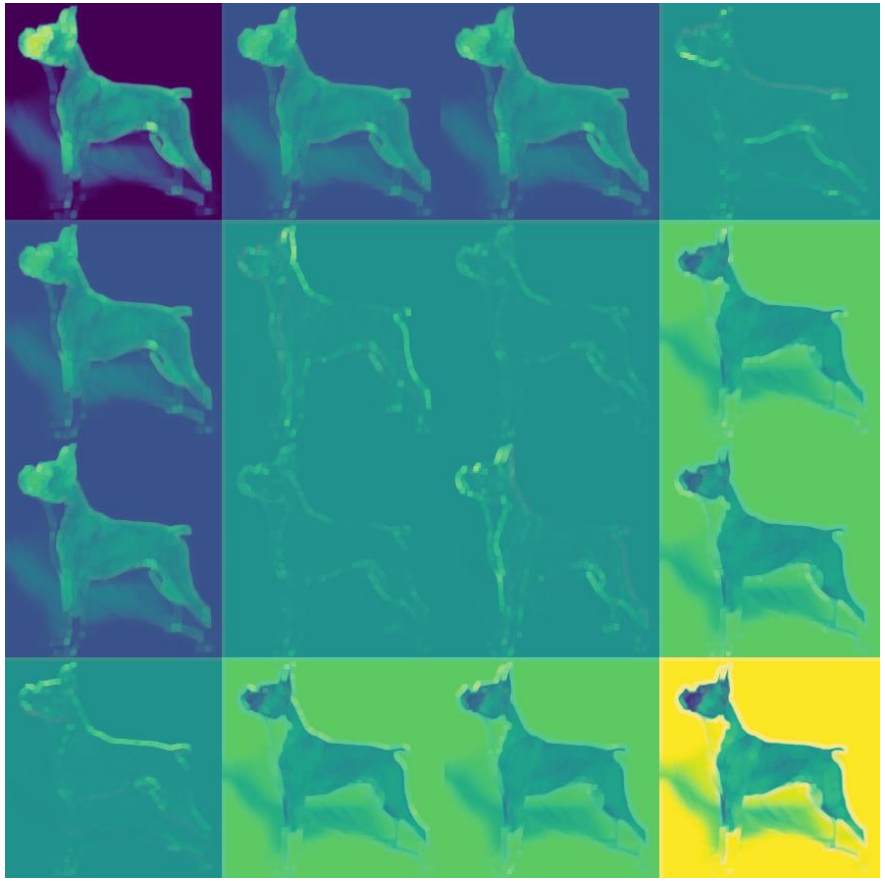


# Neural Networks for Computer Vision

Dmitry Ryabokon, github.com/dryabokon

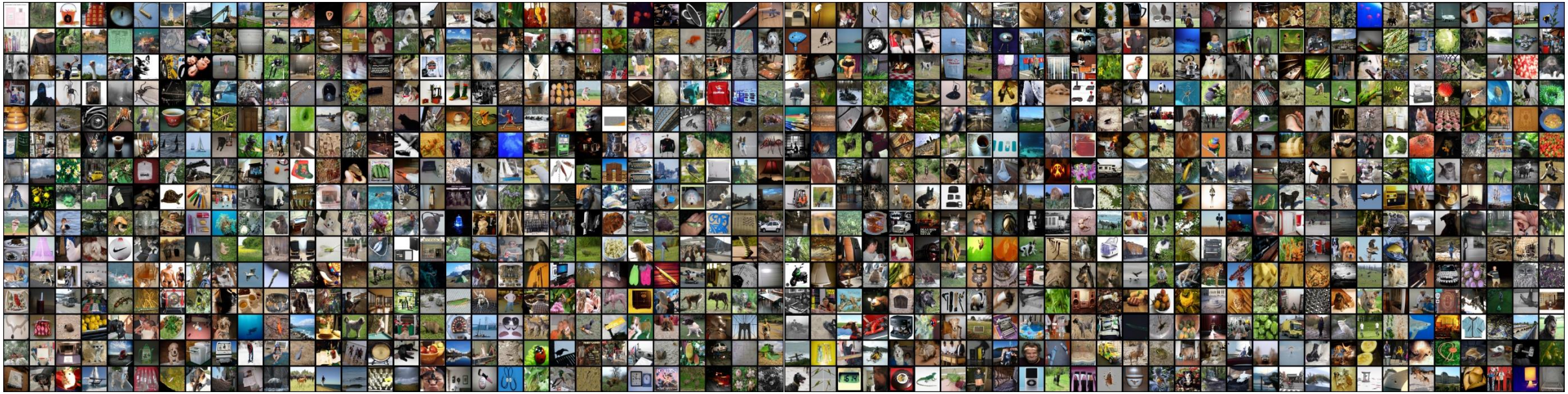
# Content

1. Intro
2. CNNs out of box
3. Basic operations
4. Feature extraction
5. Feature visualization
6. Visualization of layers and filters
7. Transfer learning
8. Fine Tuning
9. Architecture of the popular networks
10. Build, Train and Test own CNN
11. Benchmarking the CNNs
12. Autoencoders
12. Object detection
13. Style transfer
    - Colorization
    - CAM
    - Texture analysis
    - Texture synthesis
    - PixelRNN, GAN

# Intro

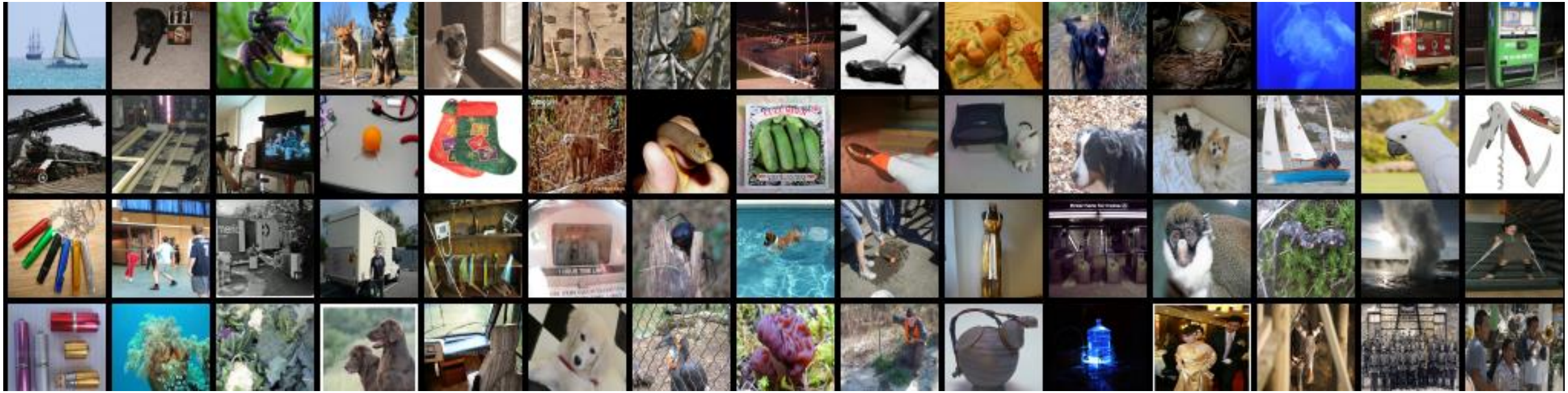
# Intro

**ImageNet database:** 1000 classes, ~100 samples for each class



# Intro

## ImageNet database

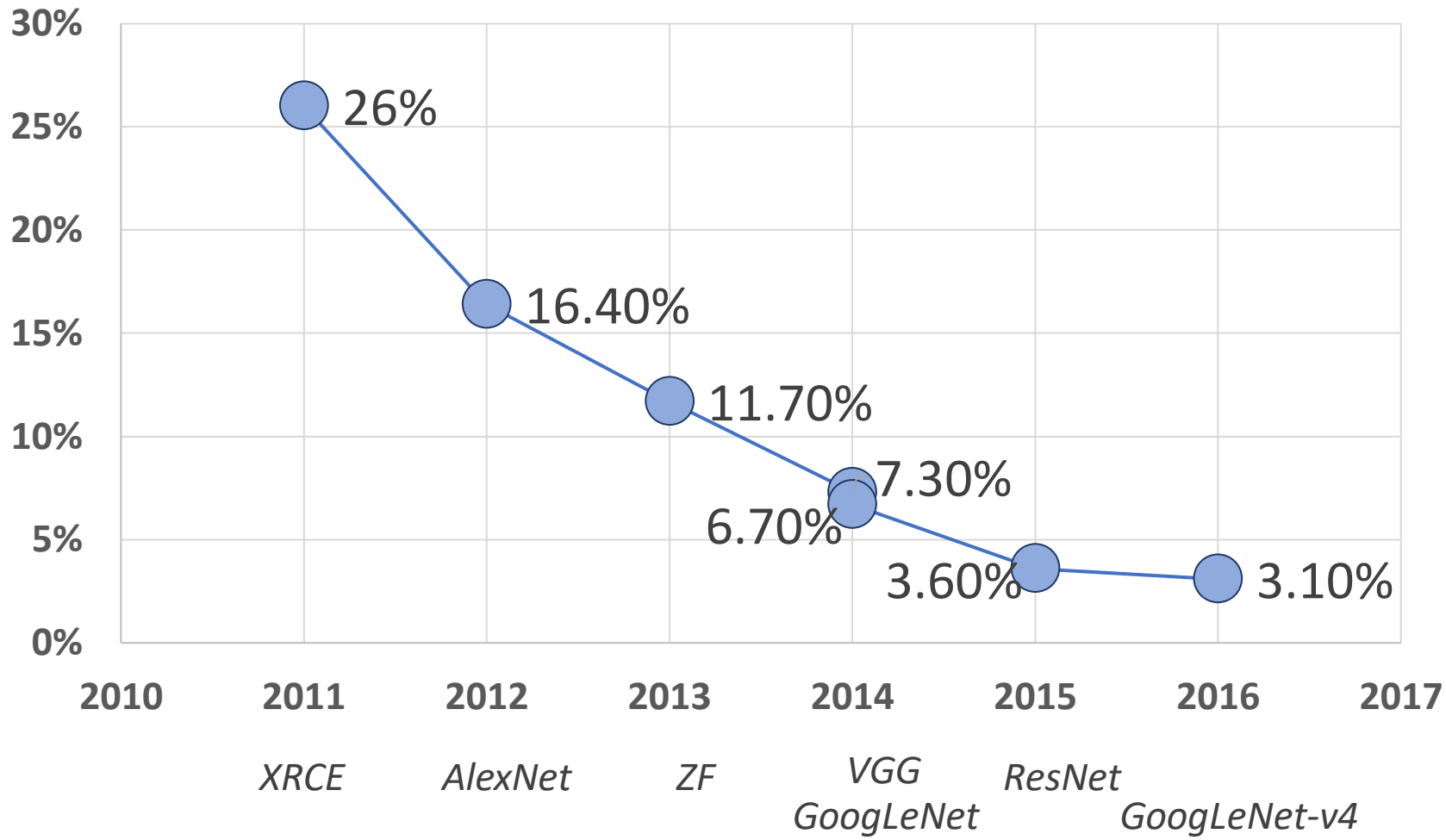


# Intro

## ImageNet database

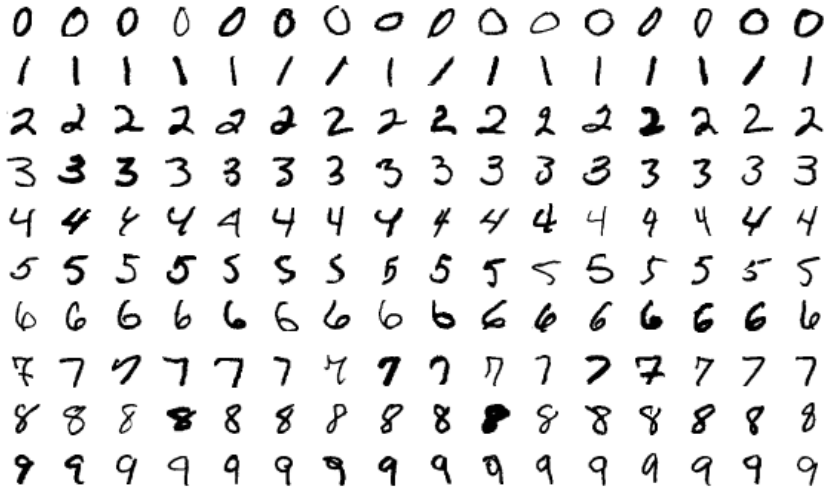
# Intro

## ImageNet Classification error

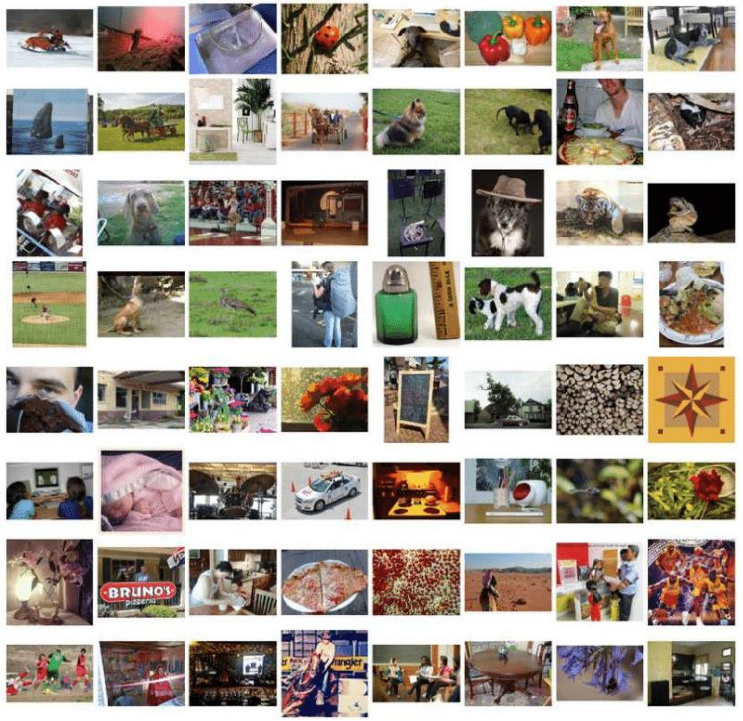


# Intro

## Some datasets available for research



**MNIST**: 10 classes, ~7000 ex. per class

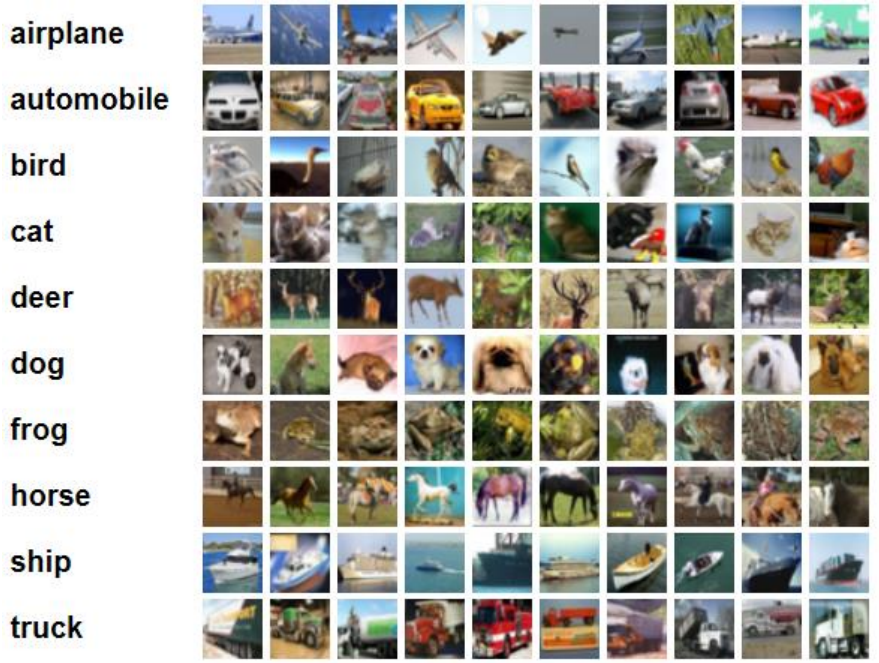


**ImageNet**: 1000 classes, ~100 ex per class

# Intro

## Some datasets available for research

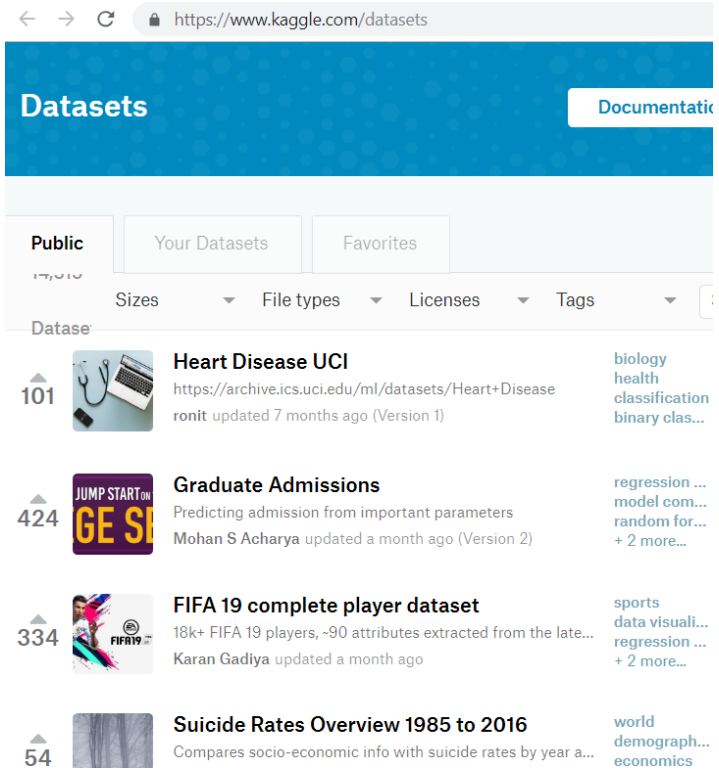
**The Street View House Numbers**
10 classes, ~2000 ex. per class




**CIFAR**: 10 classes, 6000 ex. per class
100 classes, 600 ex per class

# Some datasets available for research

**Olivetti database:** 40 classes




**.. and much more @ kaggle**

# Usage of the neural networks

identify the name of a street (in France) from an image

compressing and decompressing images

semantic image segmentation

real-world image text extraction.

image classification

image-to-text

unsupervised learning

3D object reconstruction

image matching and retrieval

automatic speech recognition

localizing and identifying multiple objects in a single image

computer vision

discovery of latent 3D keypoints

predicting future video frames

*https://github.com/keras-team/keras/tree/master/examples*
*https://github.com/tensorflow/models/tree/master/research*

# The popular networks

**Classification**

- LeNet Model
- AlexNet Model
- VGG Model
- ResNet Paper
- YOLO9000 Paper
- DenseNet Paper

**Segmentation**

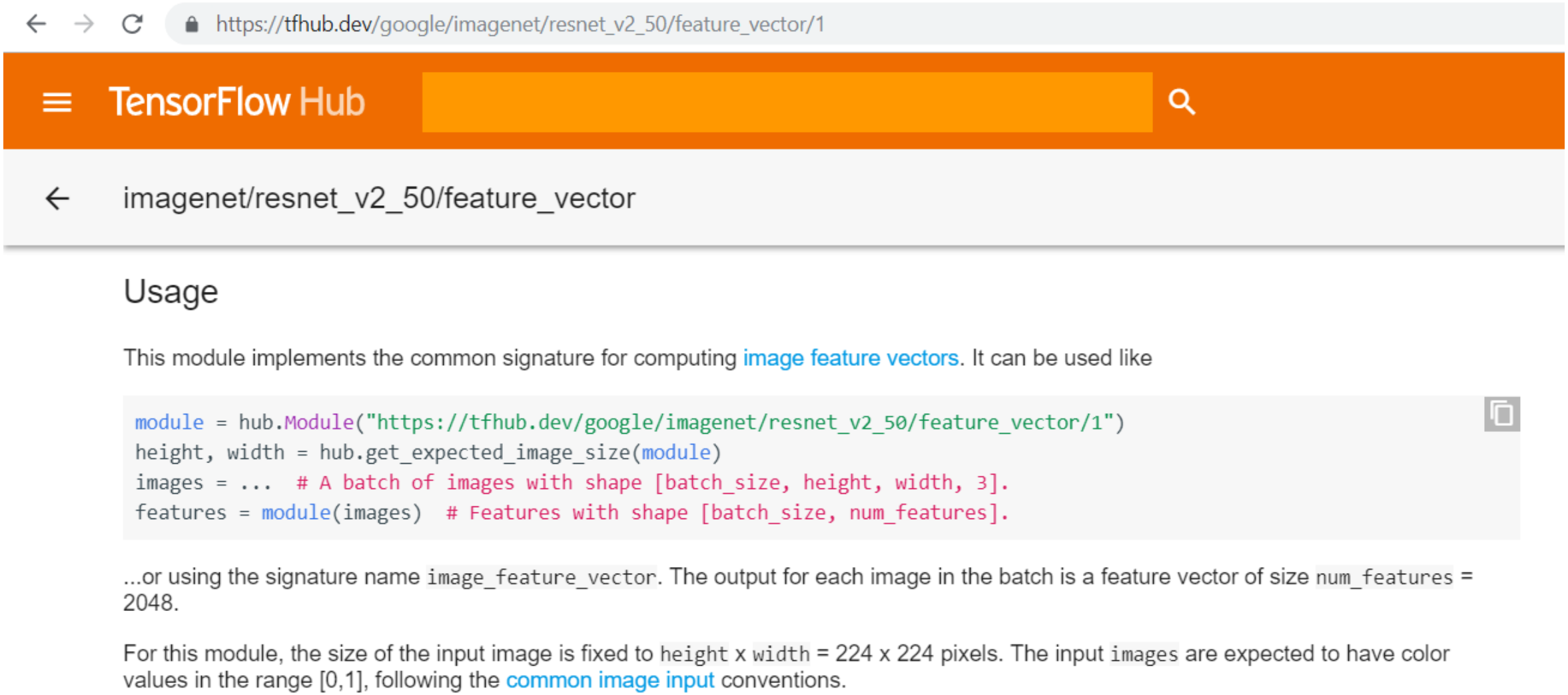
- FCN8 Paper
- SegNet Paper
- U-Net Paper
- E-Net Paper
- ResNetFCN Paper
- PSPNet Paper
- Mask RCNN Paper

**Detection**

- Faster RCNN Paper
- SSD Paper
- YOLOv2 Paper
- R-FCN Paper

# CNNs out of box

# CNNs out of box



# CNNs out of box

```
import tensorflow_hub as hub
import urllib.request
import cv2
import numpy
import json
import tensorflow as tf
# ----------------------------------------------------------------------------------------------------------------
URL = 'https://s3.amazonaws.com/deep-learning-models/image-models/imagenet_class_index.json'
data = json.loads(urllib.request.urlopen(URL).read().decode())
class_names = [data['%d'%i][1] for i in range(0,999)]
# ----------------------------------------------------------------------------------------------------------------
def example_predict():

    module = hub.Module("https://tfhub.dev/google/imagenet/resnet_v2_50/classification/1")
    img = cv2.imread('data/ex-natural/dog/dog_0000.jpg')
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (224, 224)).astype(numpy.float32)
    img = numpy.array([img]).astype(numpy.float32) / 255.0
    sess = tf.Session()
    sess.run(tf.global_variables_initializer())
    sess.run(tf.tables_initializer())

    outputs = module(dict(images=img), signature="image_classification", as_dict=True)
    prob = outputs["default"].eval(session=sess)[0]
    idx = numpy.argsort(-prob)[0]

    print(class_names[idx], prob[idx])
    sess.close()

    return
# ----------------------------------------------------------------------------------------------------------------
if __name__ == '__main__':
    example_predict()
```

# CNNs out of box

https://keras.io/getting-started/faq/#how-can-i-use-pre-trained-models-in-keras

Keras Documentation

Search docs

Home

Why use Keras

GETTING STARTED

Guide to the Sequential model

Guide to the Functional API

FAQ

How should I cite Keras?

How can I run Keras on GPU?

How can I run a Keras model on multiple GPUs?

What does "sample", "batch", "epoch" mean?

How can I save a Keras model?

Why is the training loss much higher than the testing loss?

How can I obtain the output of an intermediate layer?

How can I use Keras with datasets that don't fit in memory?

How can I interrupt training when the validation loss isn't decreasing

## How can I use pre-trained models in Keras?

Code and pre-trained weights are available for the following image classification models:

- Xception
- VGG16
- VGG19
- ResNet50
- Inception v3
- Inception-ResNet v2
- MobileNet v1
- DenseNet
- NASNet
- MobileNet v2

They can be imported from the module `keras.applications`:

```
from keras.applications.xception import Xception
from keras.applications.vgg16 import VGG16
from keras.applications.vgg19 import VGG19
from keras.applications.resnet50 import ResNet50
from keras.applications.inception_v3 import InceptionV3
from keras.applications.inception_resnet_v2 import InceptionResNetV2
from keras.applications.mobilenet import MobileNet
from keras.applications.densenet import DenseNet121
from keras.applications.densenet import DenseNet169
from keras.applications.densenet import DenseNet201
from keras.applications.nasnet import NASNetLarge
from keras.applications.nasnet import NASNetMobile
from keras.applications.mobilenet_v2 import MobileNetV2

model = VGG16(weights='imagenet', include_top=True)
```

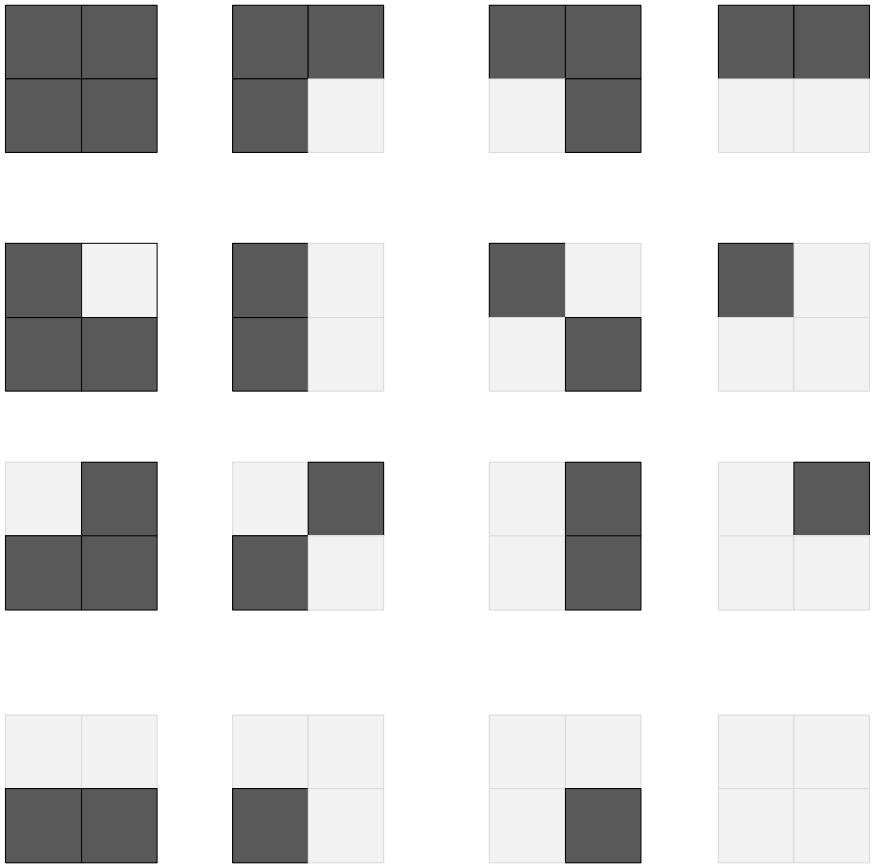
# CNNs out of box

```
from keras.applications import MobileNet
from keras.applications.xception import Xception
from keras.applications.mobilenet import preprocess_input
import urllib.request
import cv2
import numpy
import json
from keras import backend as K
K.set_image_dim_ordering('tf')
# ------------------------------------------------------------------------------------------------------------
-------
URL = 'https://s3.amazonaws.com/deep-learning-models/image-models/imagenet_class_index.json'
data = json.loads(urllib.request.urlopen(URL).read().decode())
class_names = [data['%d'%i][1] for i in range(0,999)]
# ------------------------------------------------------------------------------------------------------------
-------
def example_predict():

    CNN = MobileNet()
    #CNN = Xception()

    img = cv2.imread('data/ex-natural/dog/dog_0000.jpg')
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (224, 224)).astype(numpy.float32)

    prob = CNN.predict(preprocess_input(numpy.array([img])))
    idx = numpy.argsort(-prob[0])[0]
    print(class_names[idx], prob[0, idx])

    return
# ------------------------------------------------------------------------------------------------------------
-------
if __name__ == '__main__':
    example_predict()
```

# Basic operations

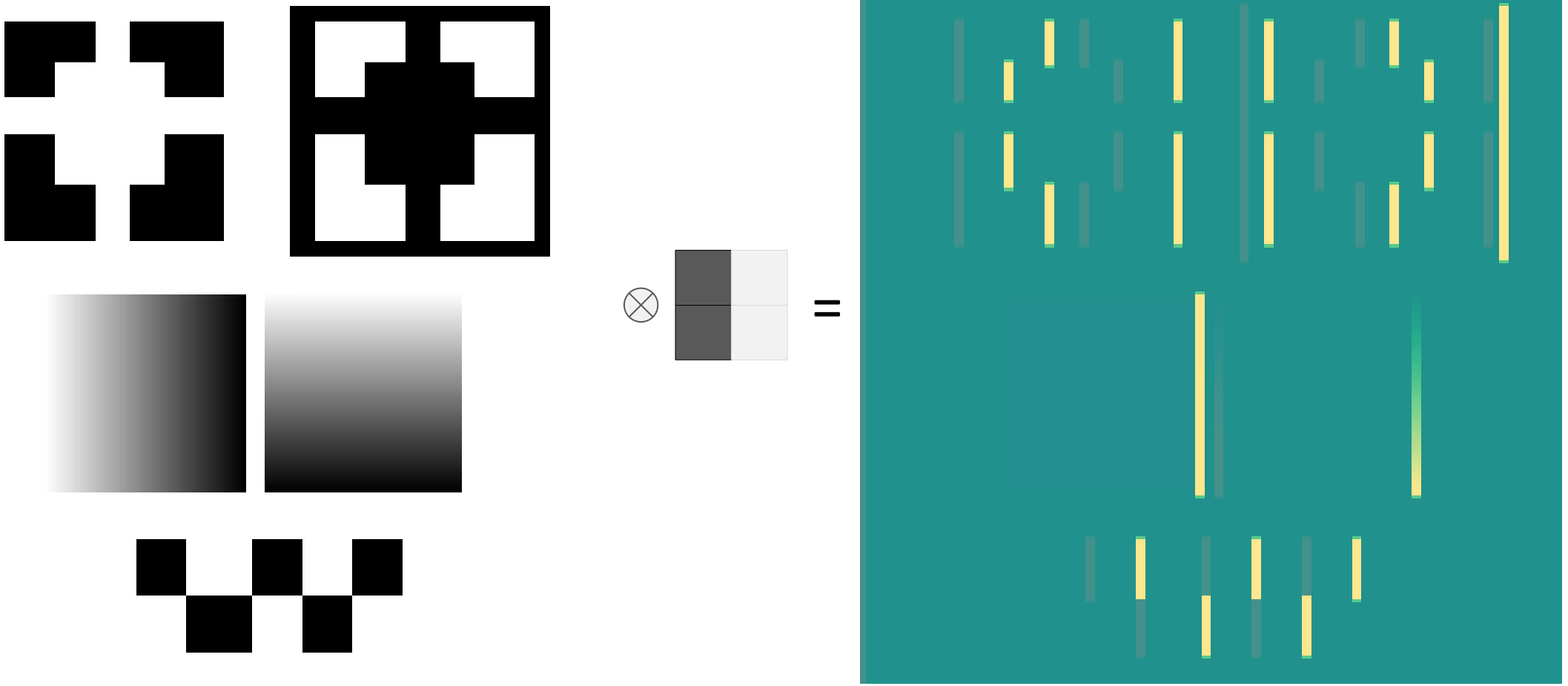
# Basic operations

## Convolution



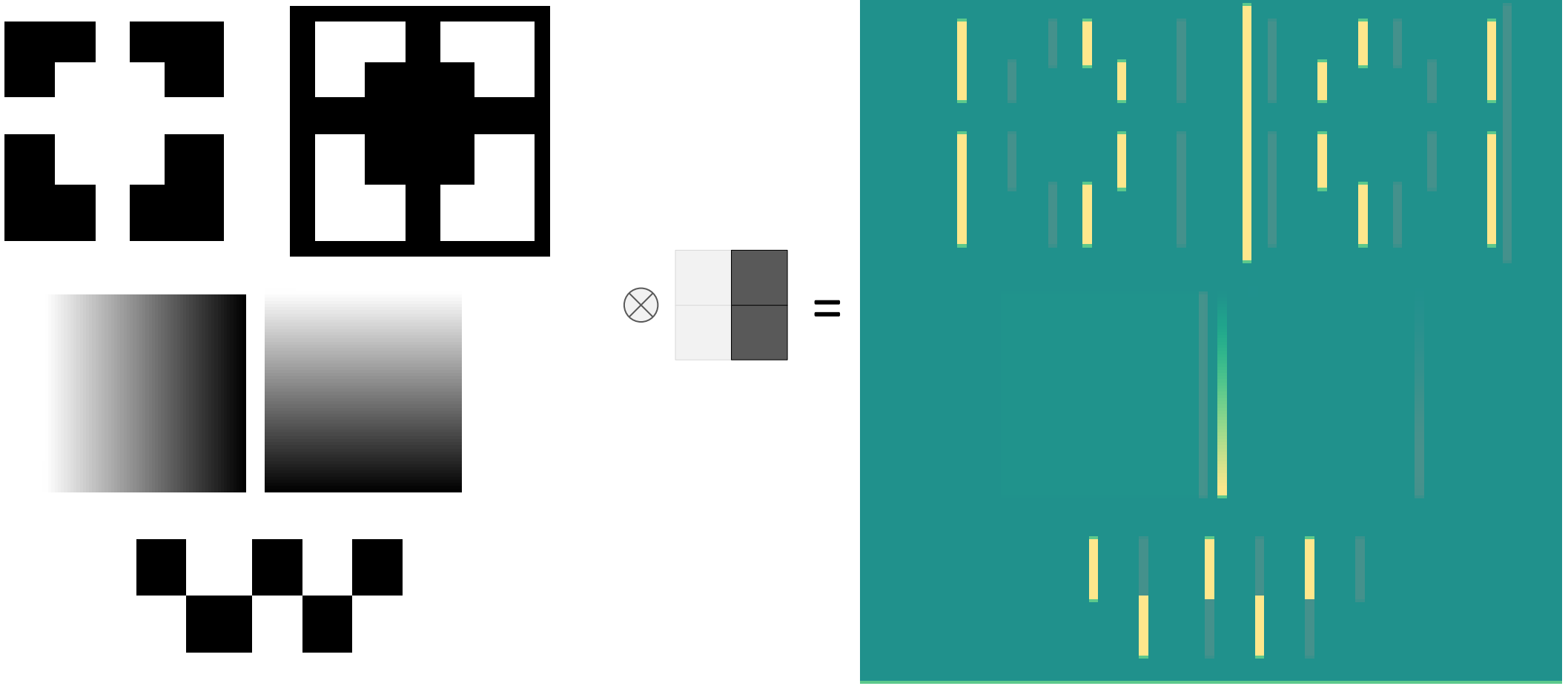
# Basic operations

## Convolution



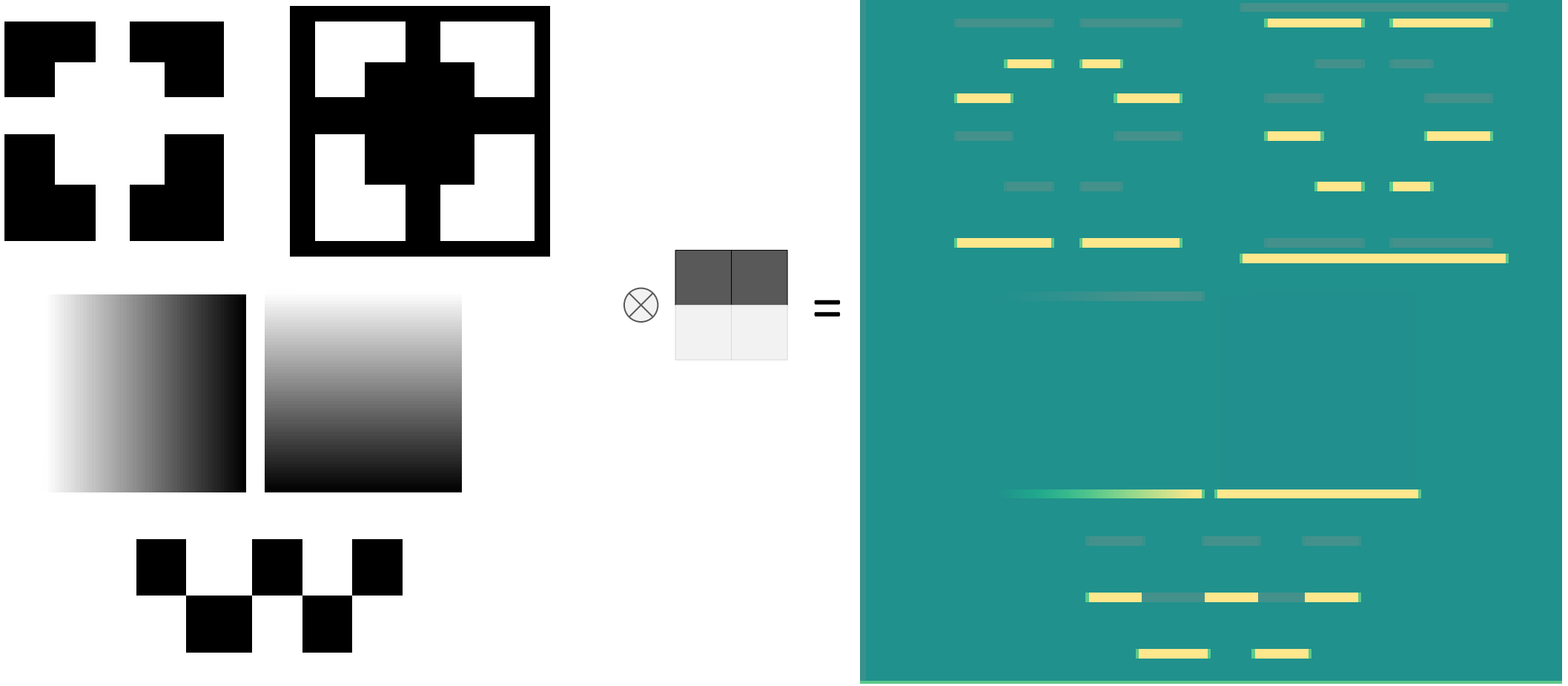
# Basic operations

## Convolution



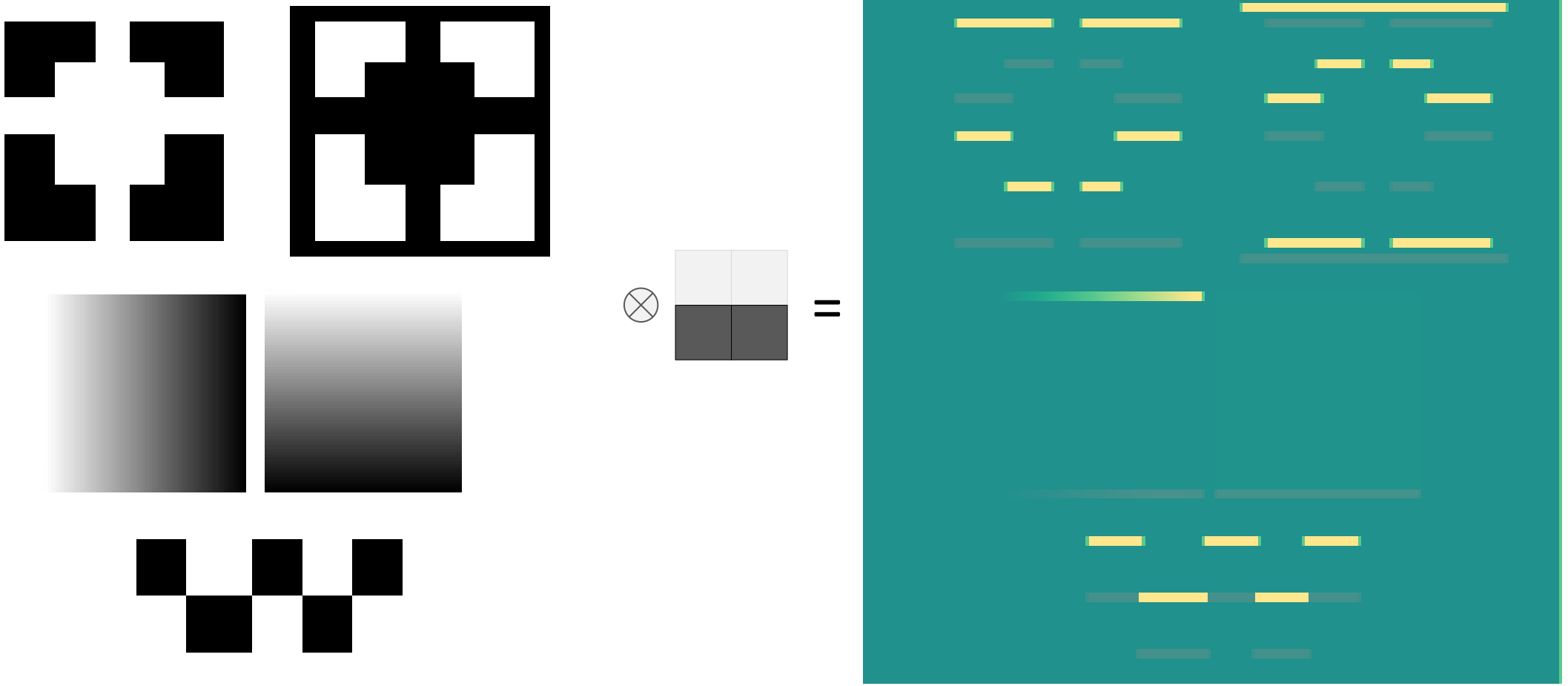
# Basic operations

## Convolution



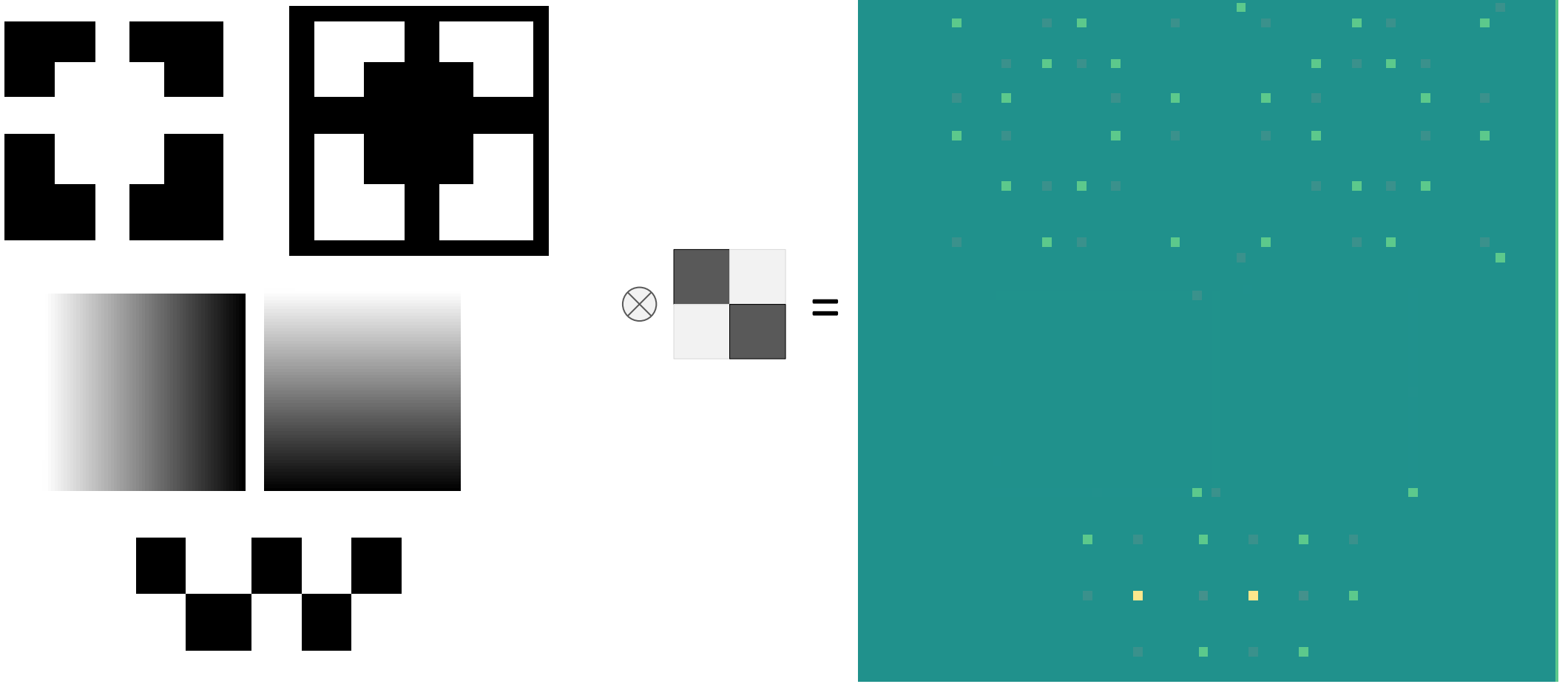
# Basic operations

## Convolution



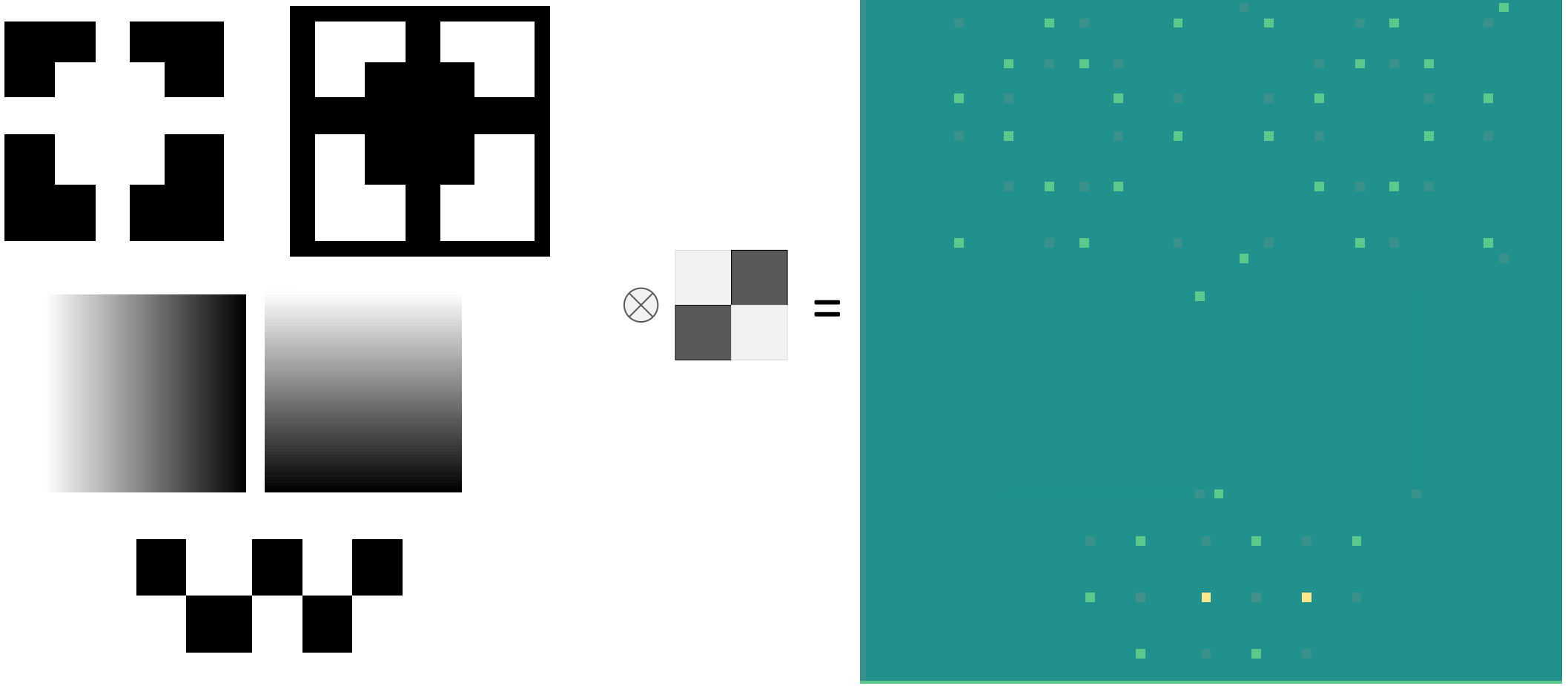
# Basic operations

## Convolution



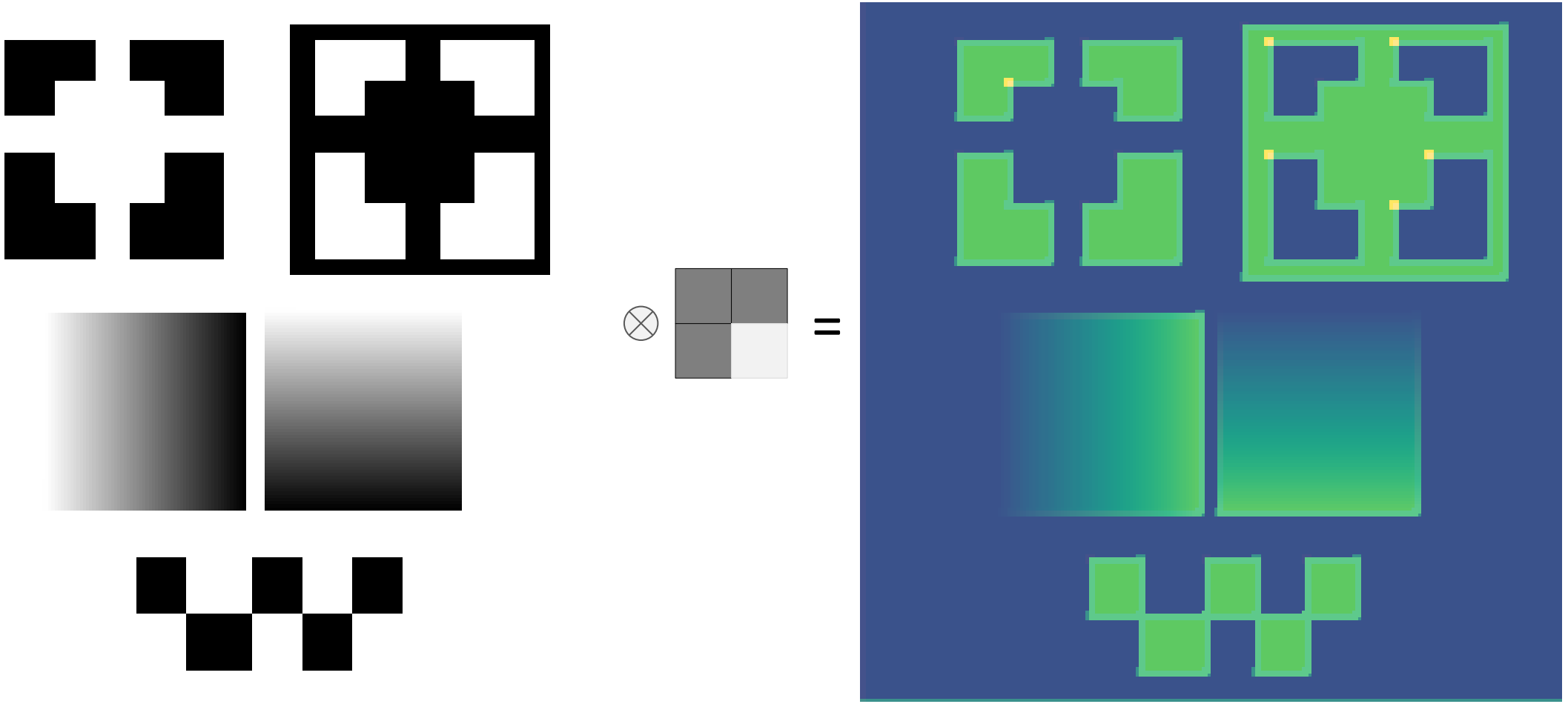
# Basic operations

## Convolution



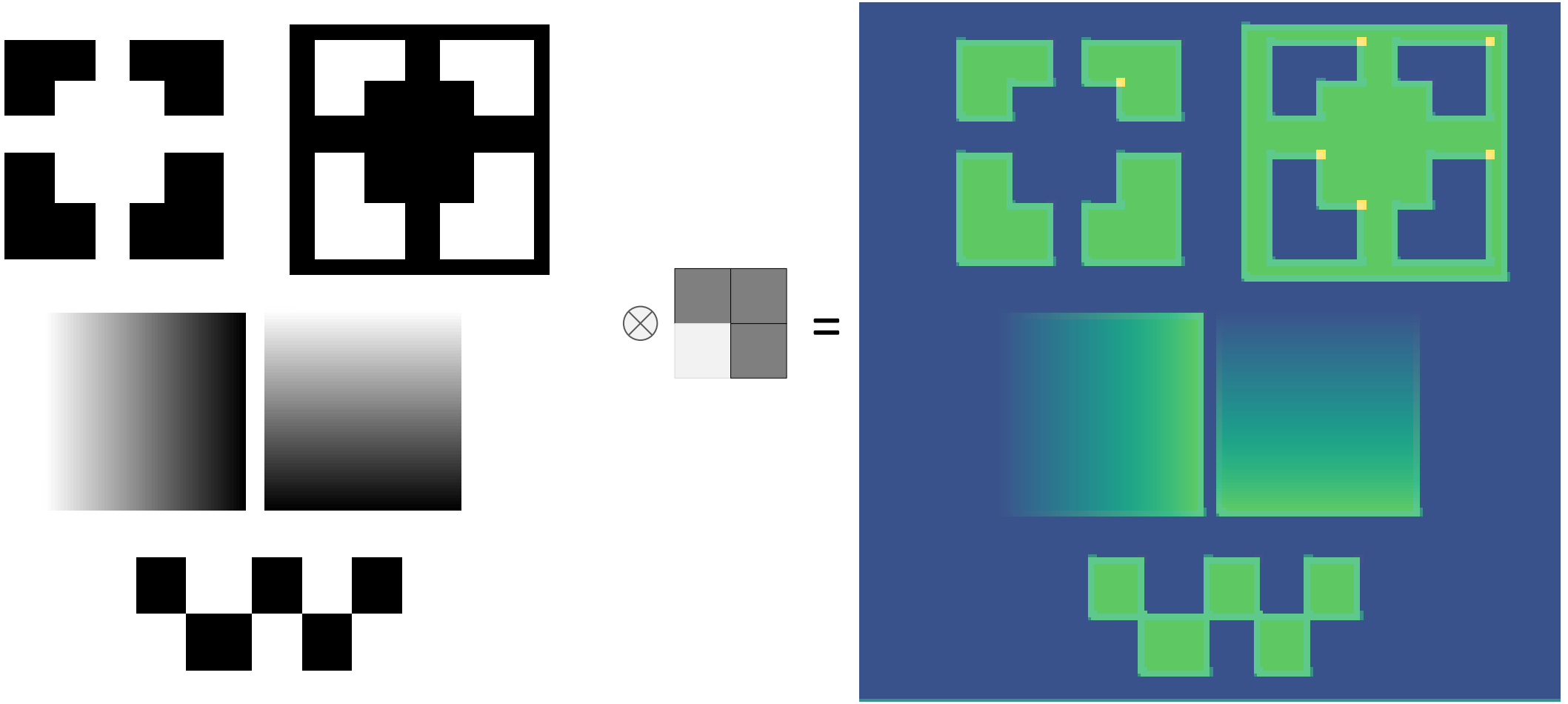
# Basic operations

## Convolution



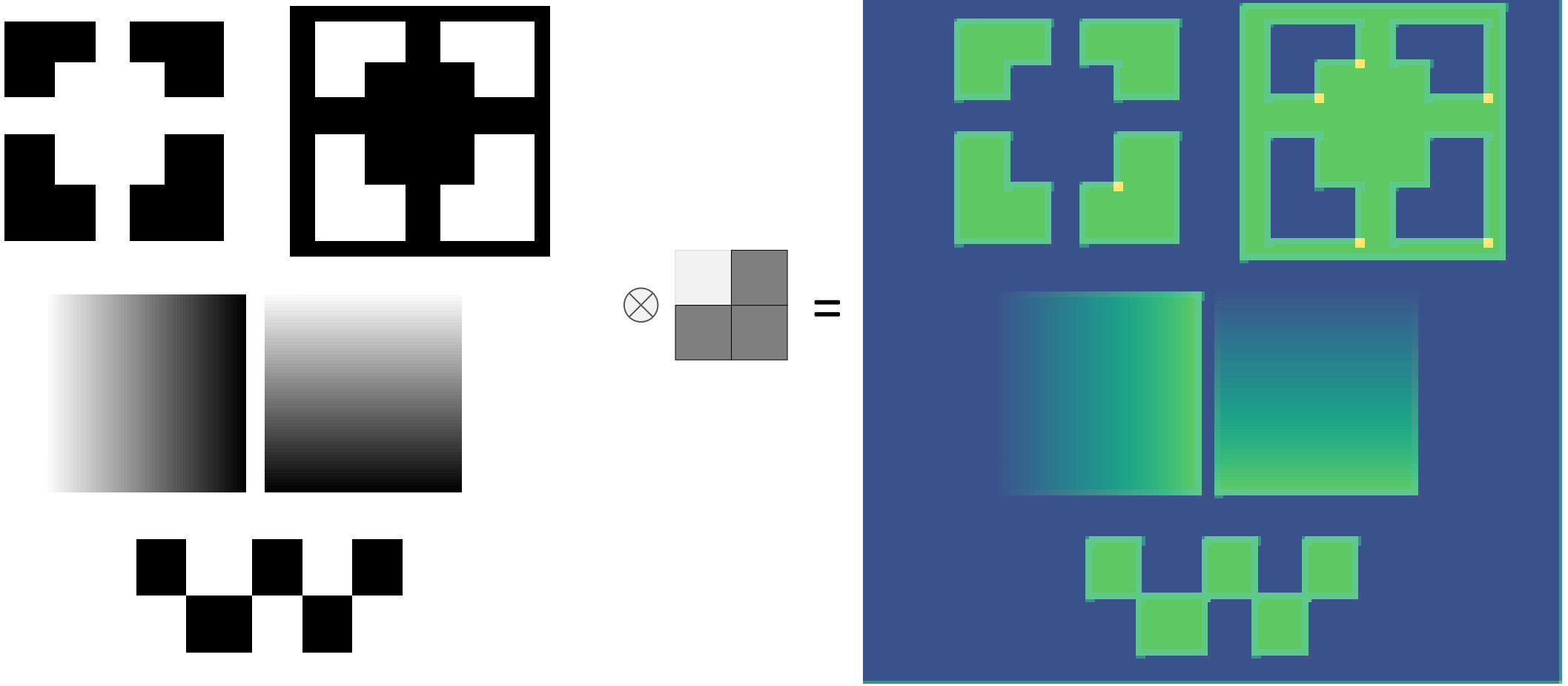
# Basic operations

## Convolution



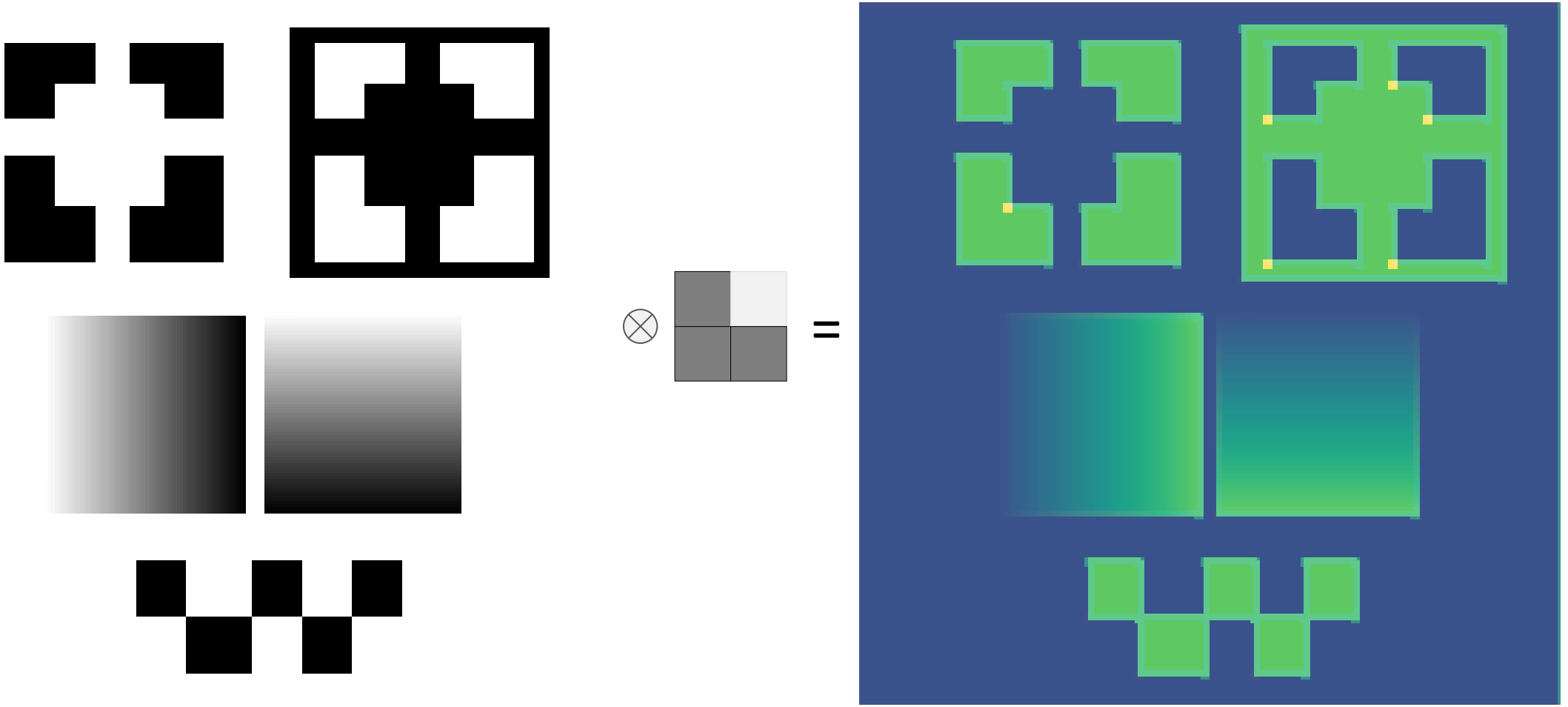
# Basic operations

## Convolution



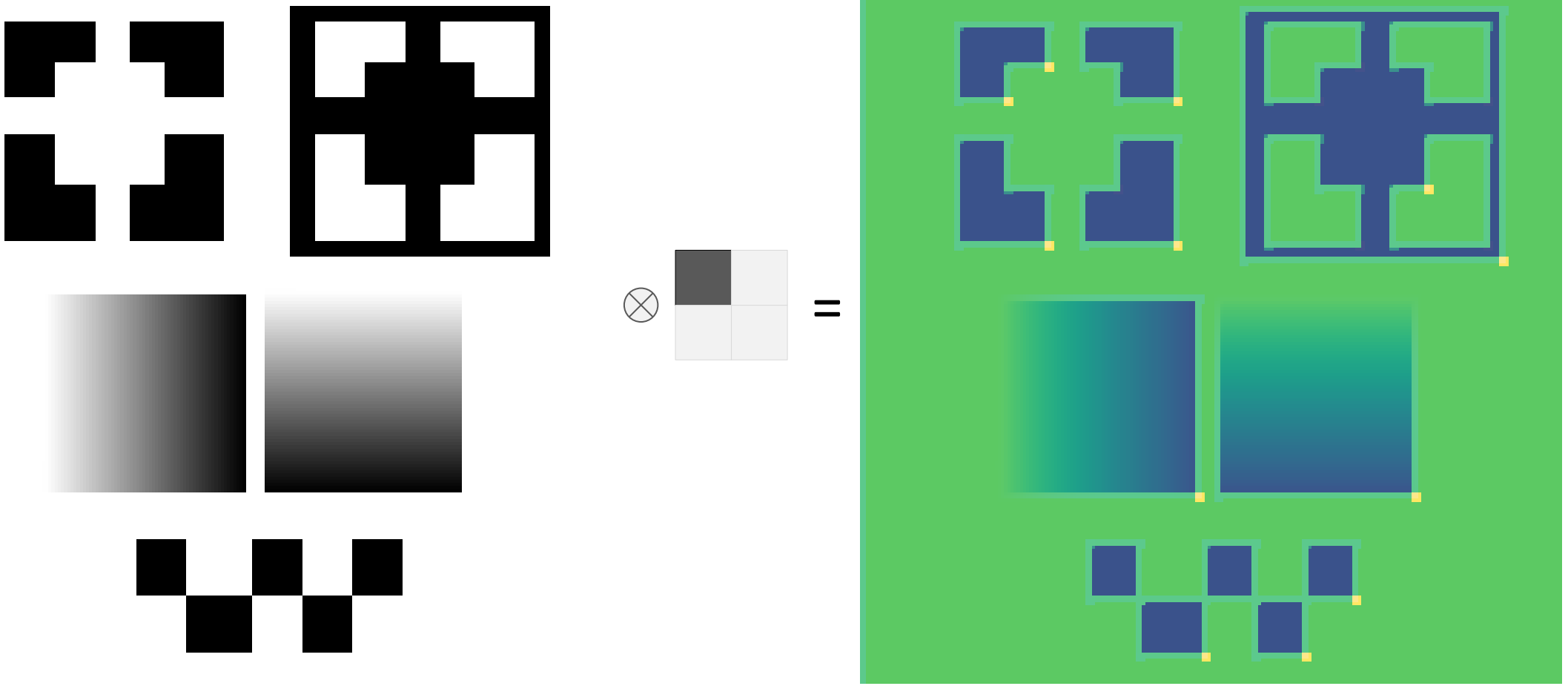
# Basic operations

## Convolution



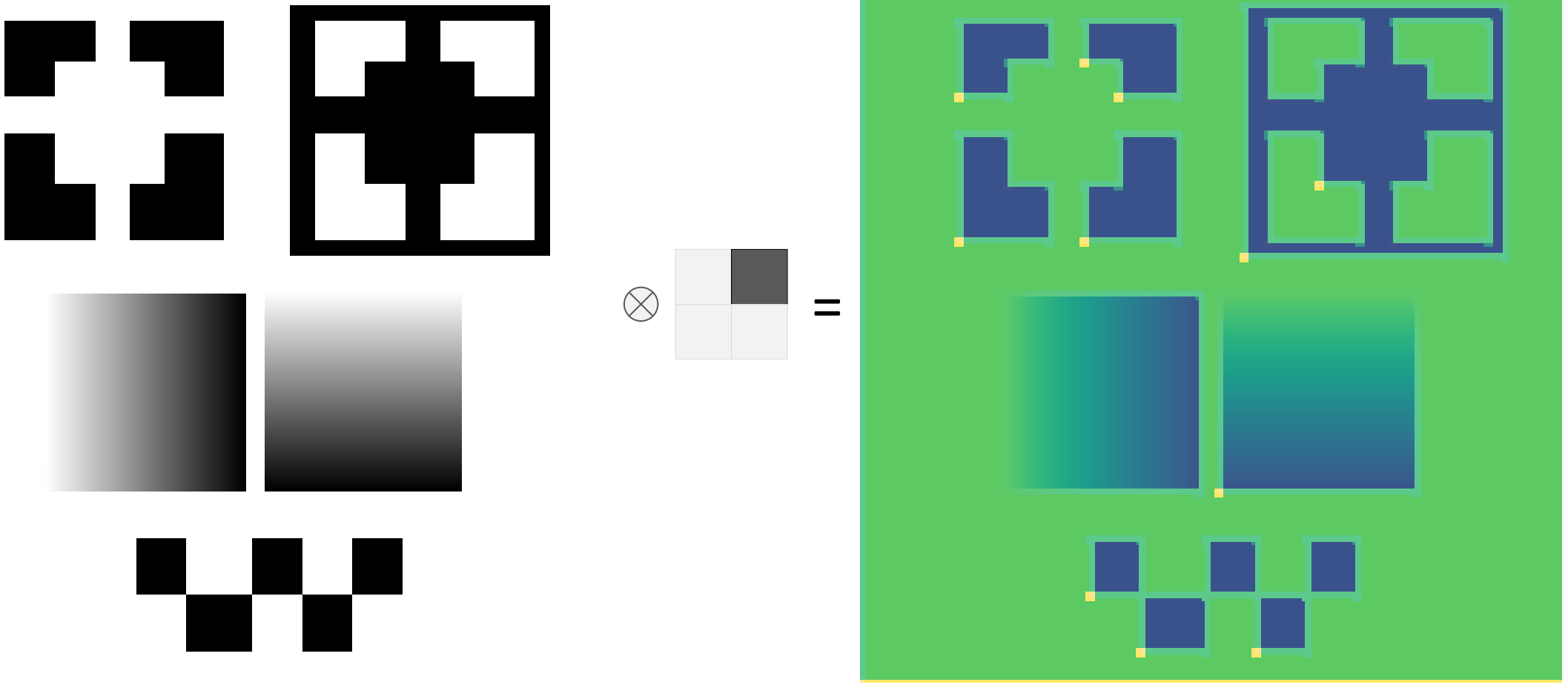
# Basic operations

## Convolution



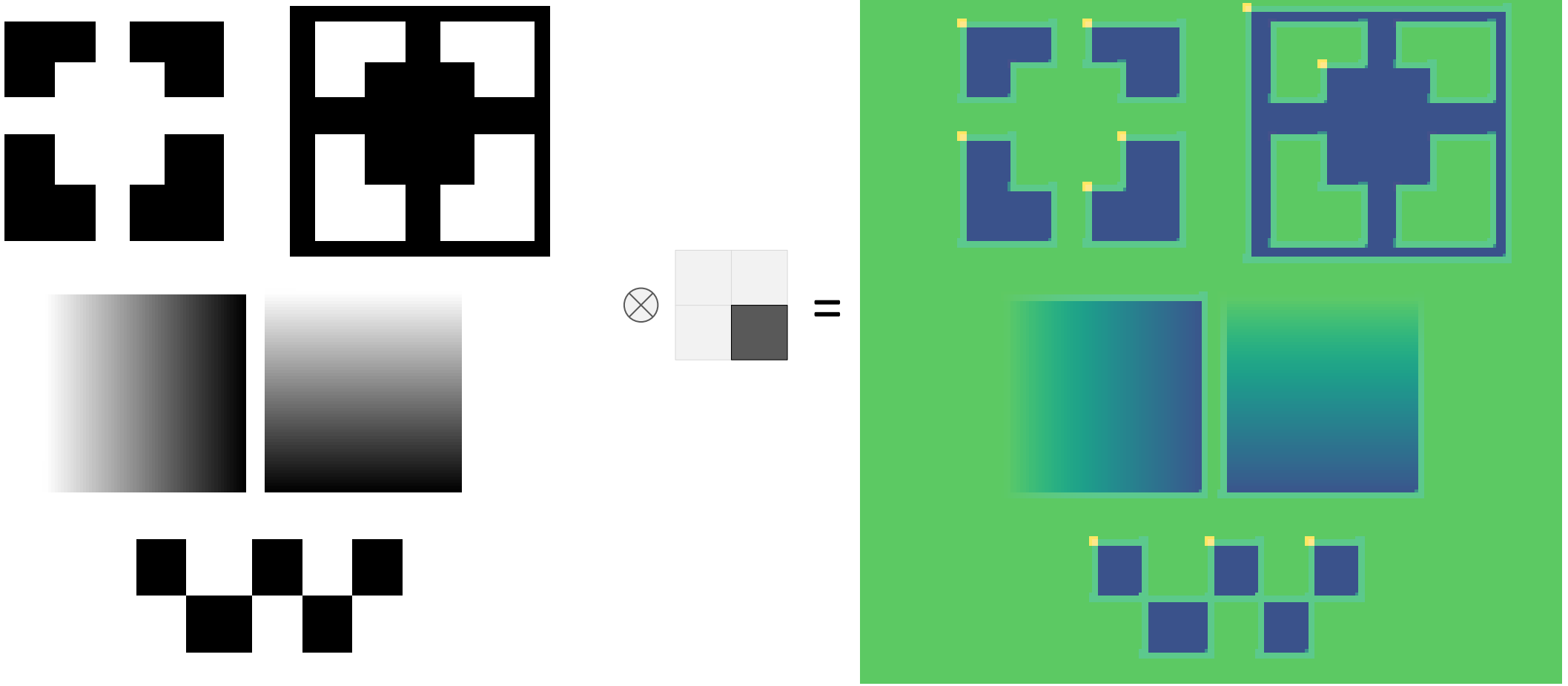
# Basic operations

## Convolution



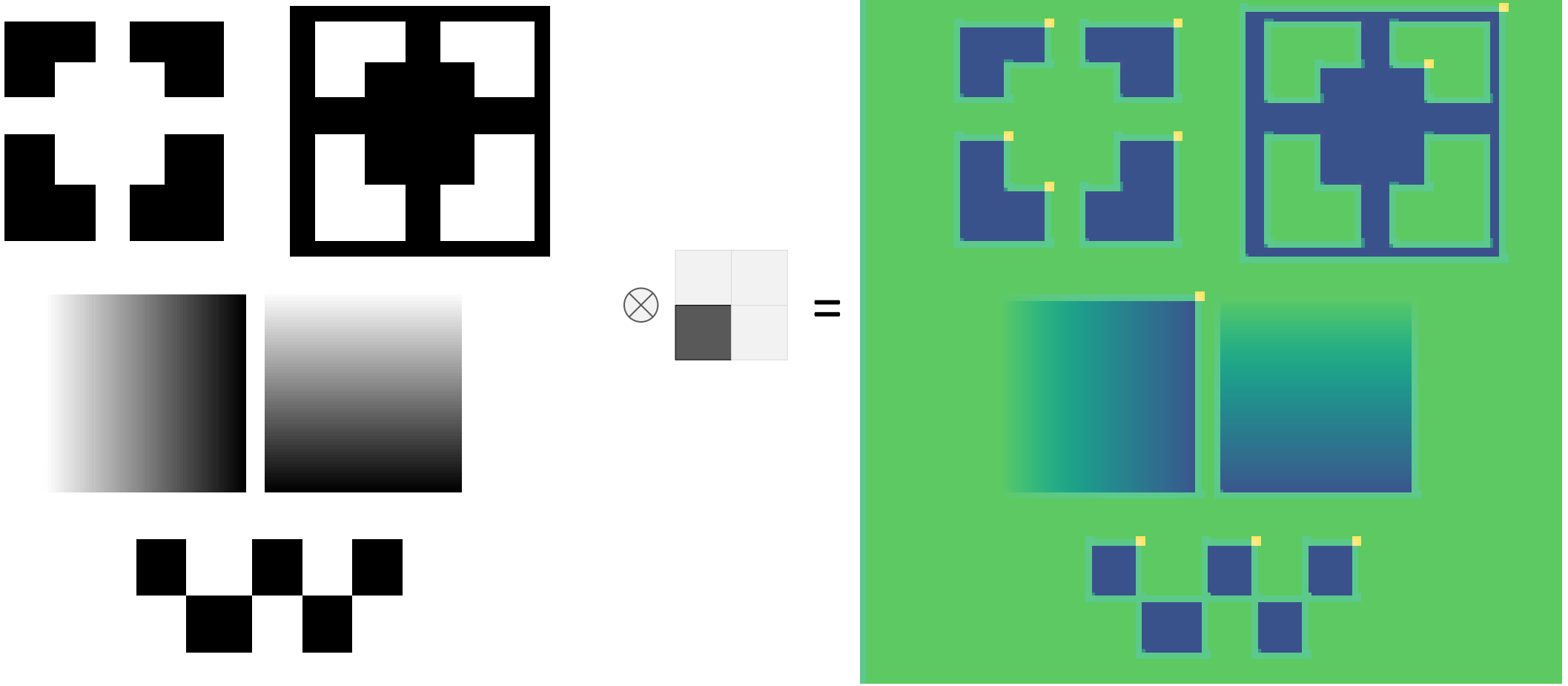
# Basic operations

## Convolution



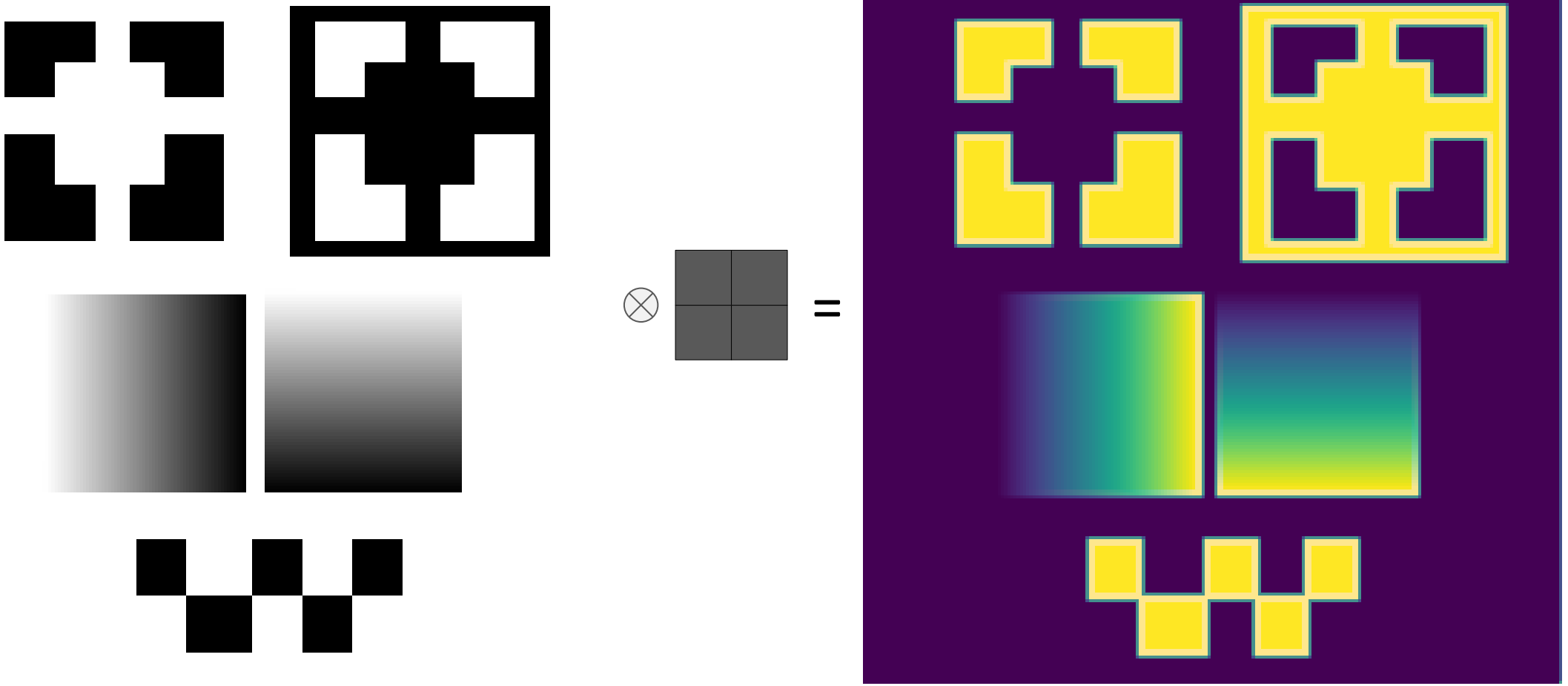
# Basic operations

## Convolution



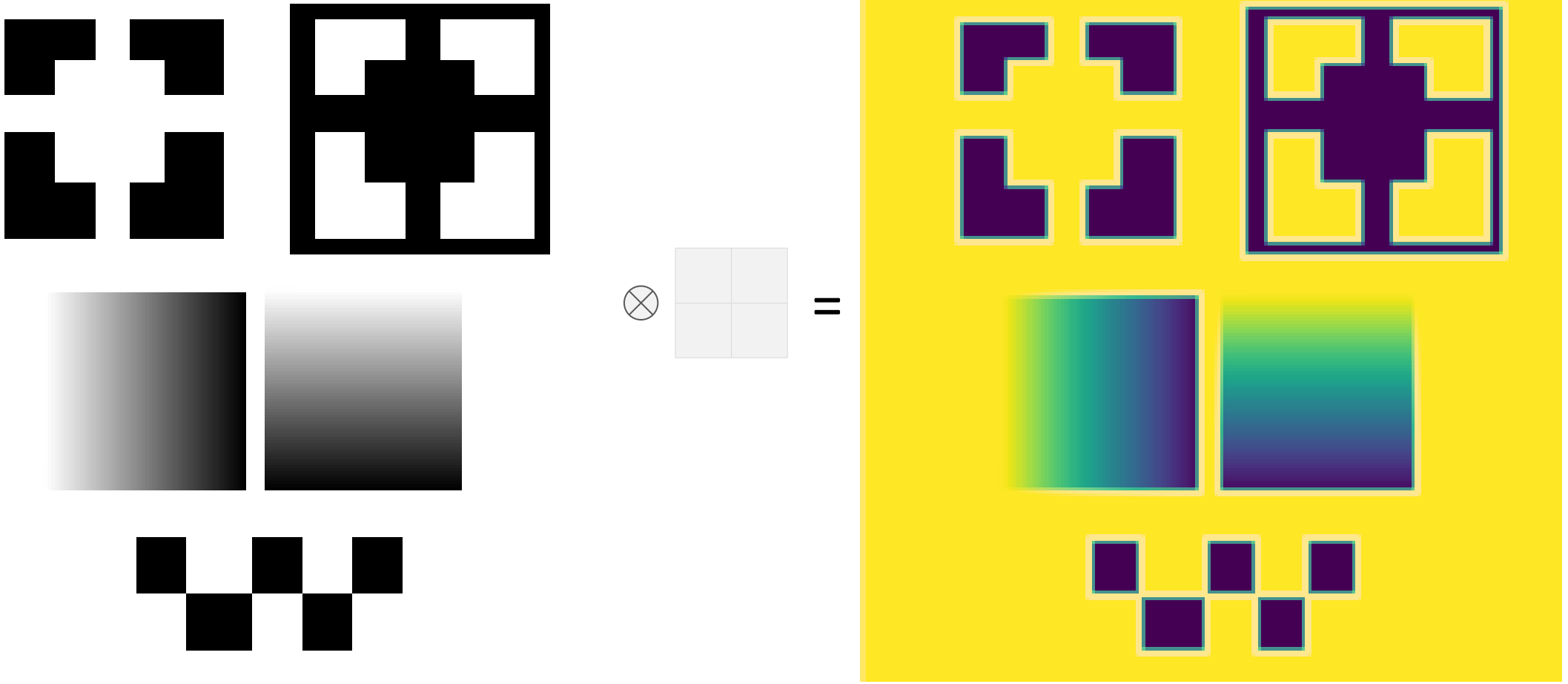
# Basic operations

## Convolution



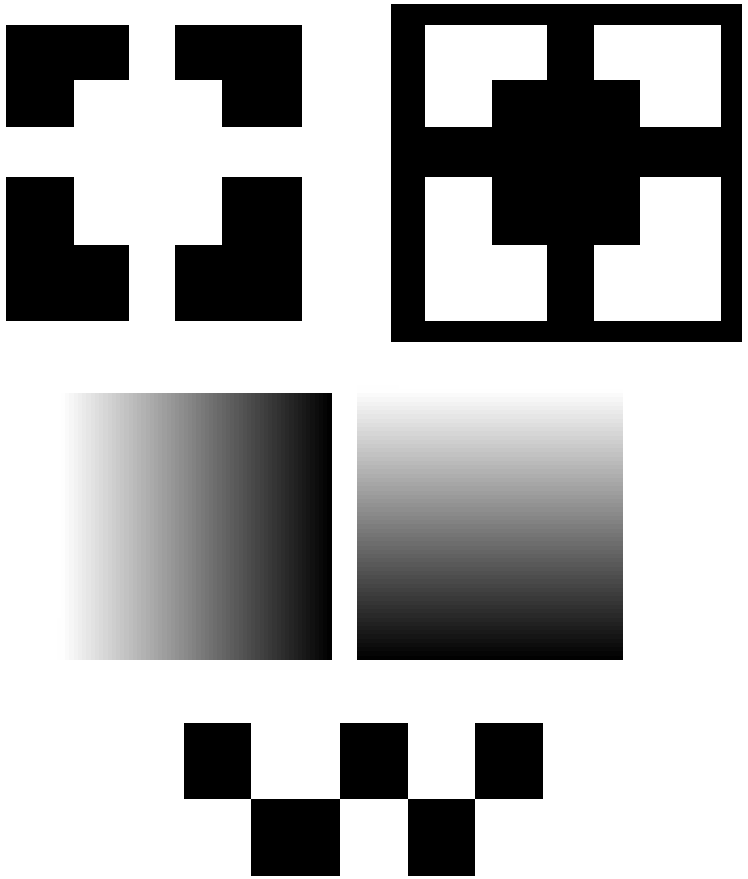
# Basic operations

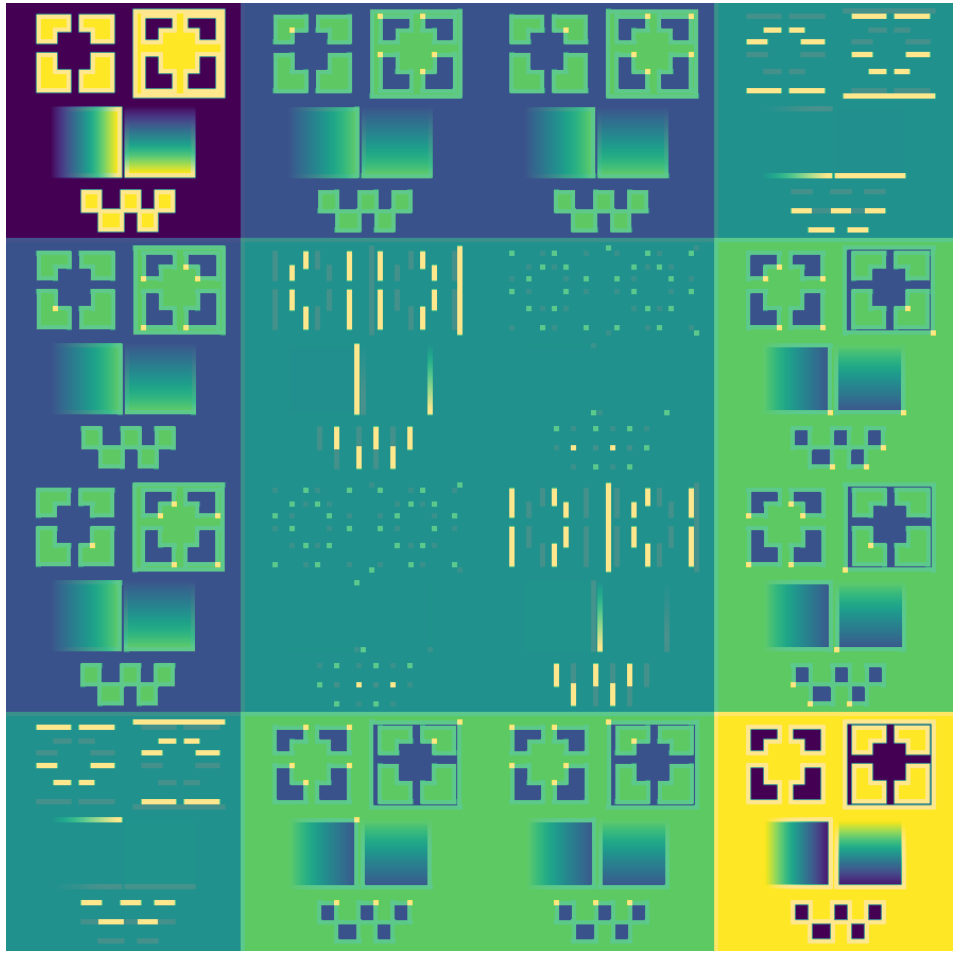
## Convolution



# Basic operations

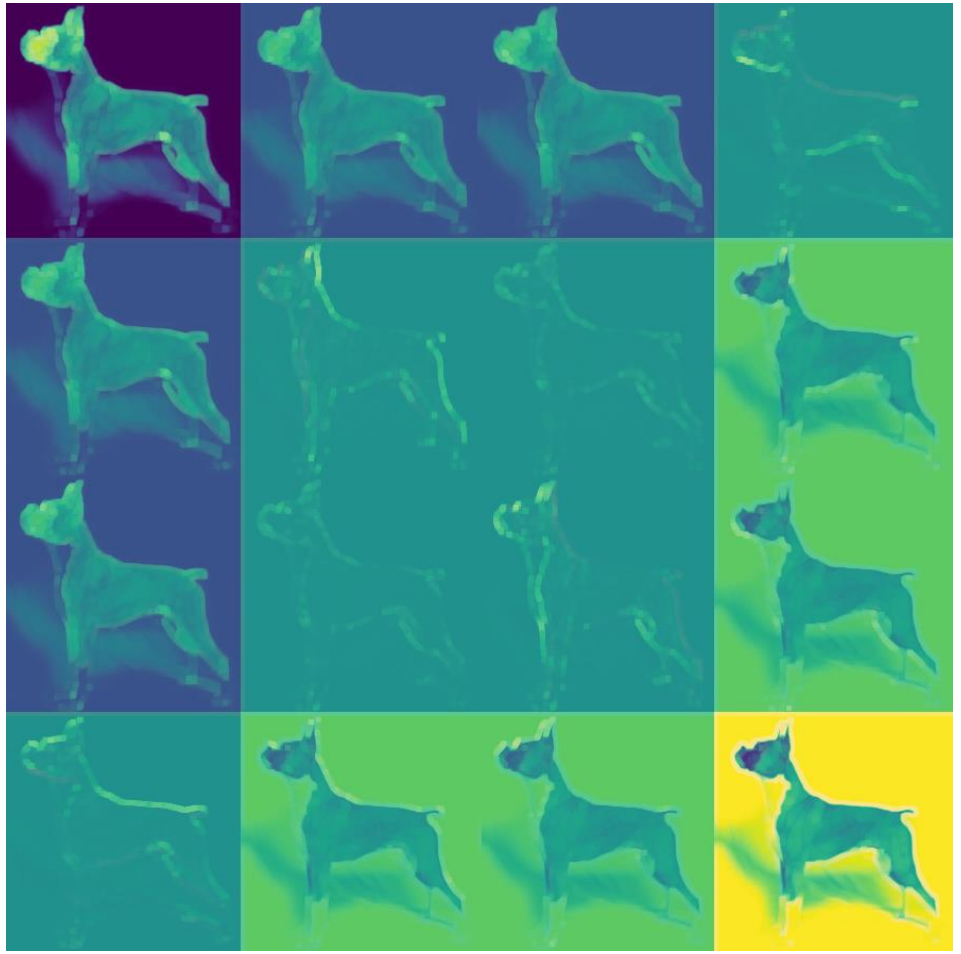
## Convolution





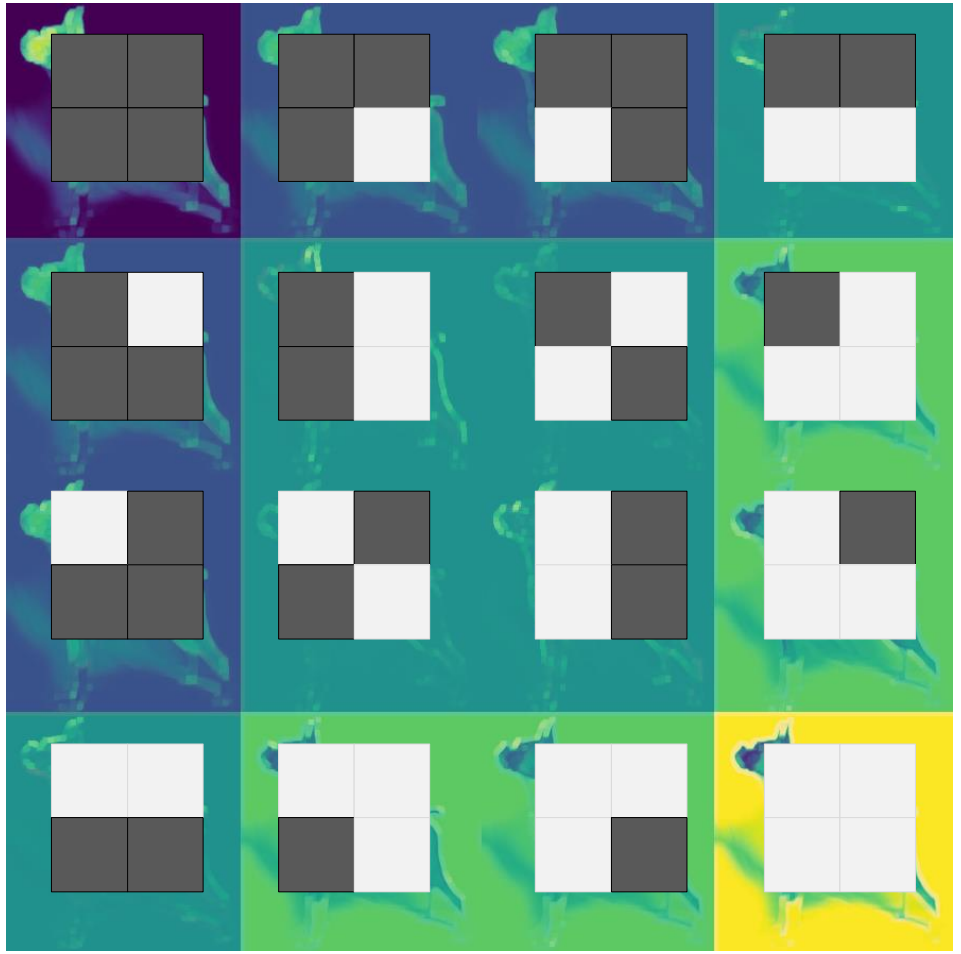
# Basic operations

## Convolution



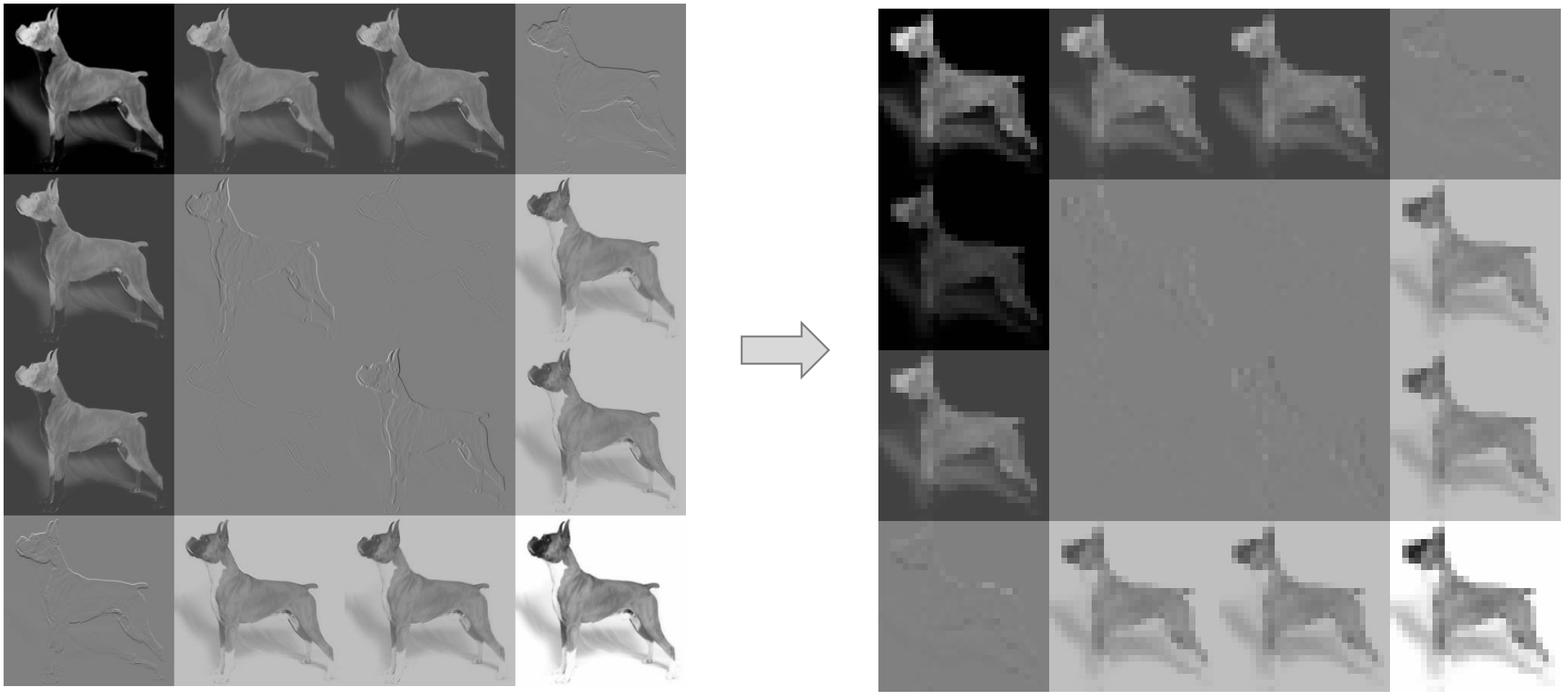
# Basic operations

## De-Convolution



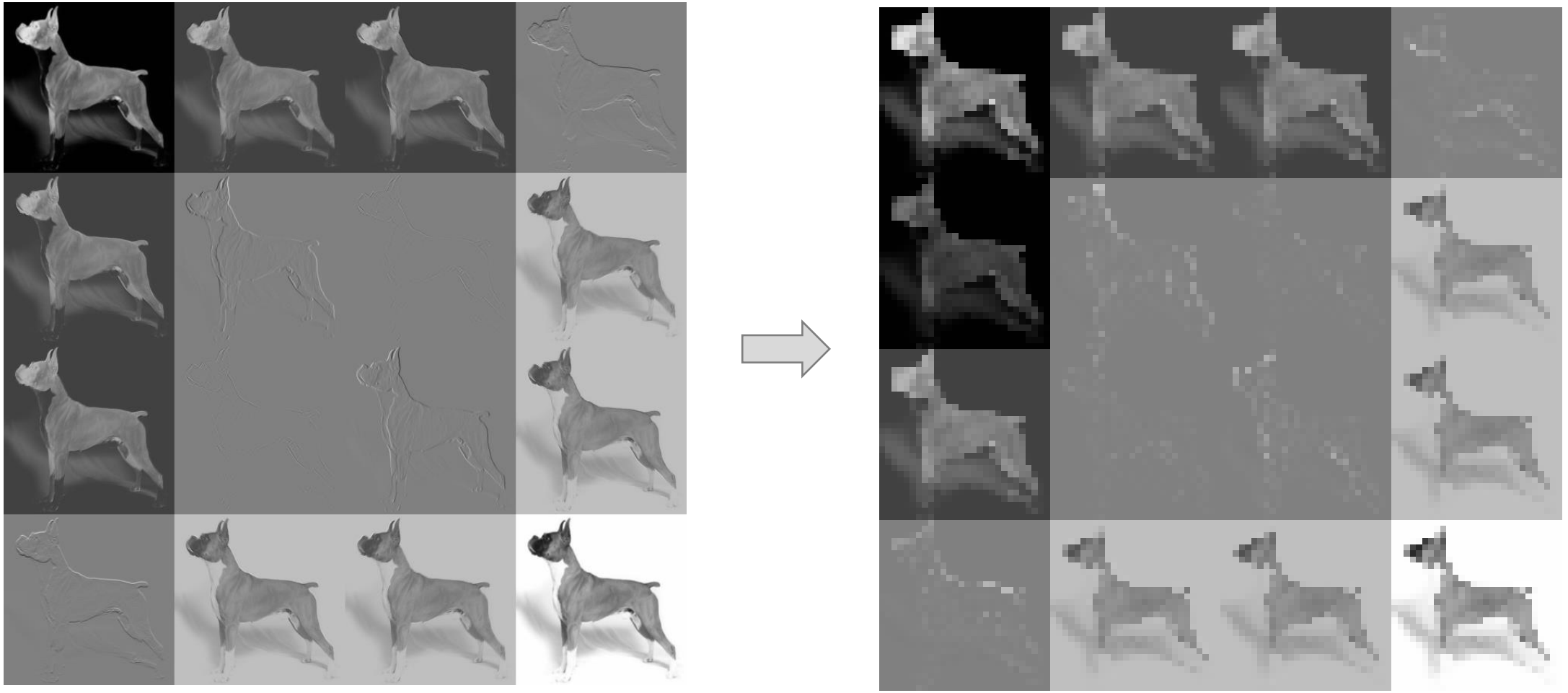
# Basic operations

## Pool: avg



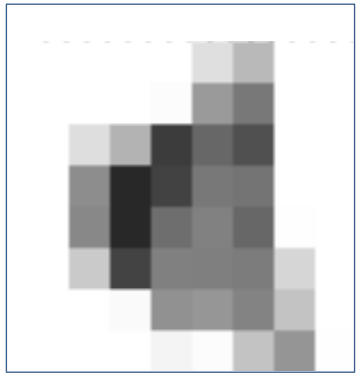
# Basic operations

## Pool: max



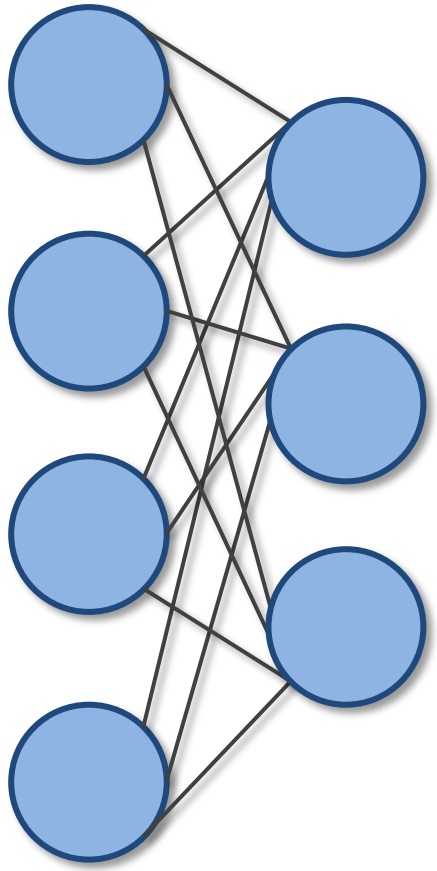
# Basic operations

## Flatten



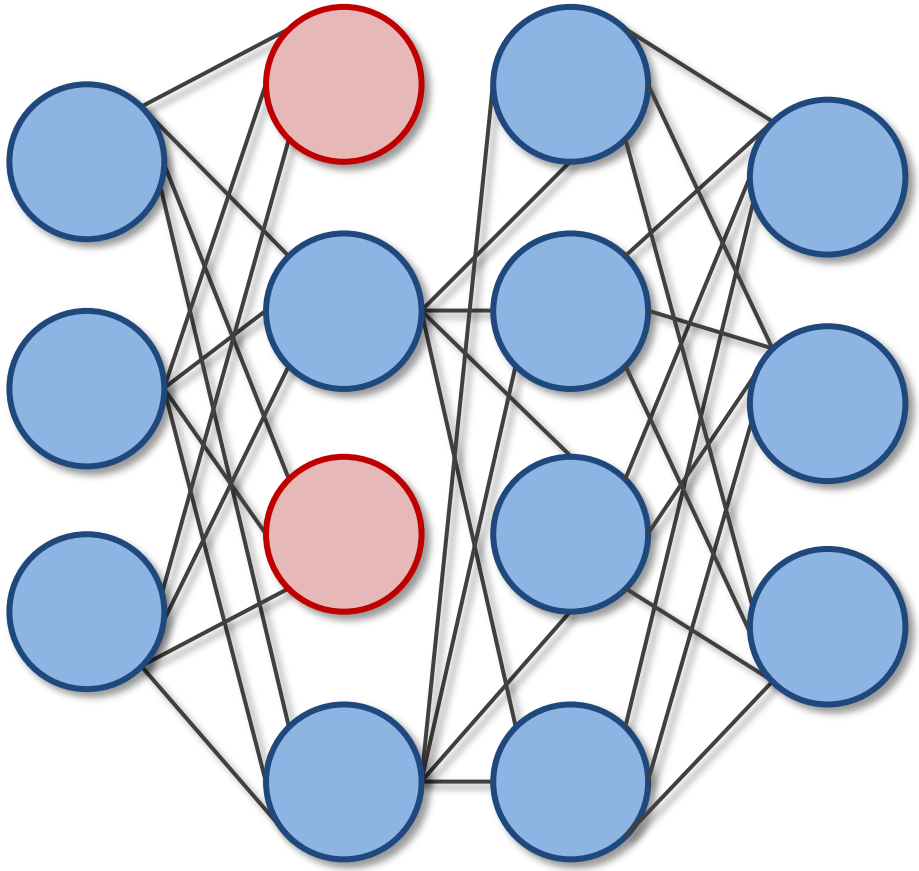
# Basic operations

## Fully Connected



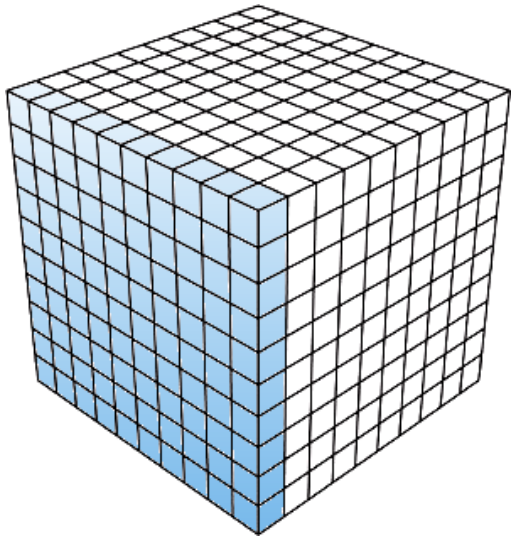
# Basic operations

## Dropout

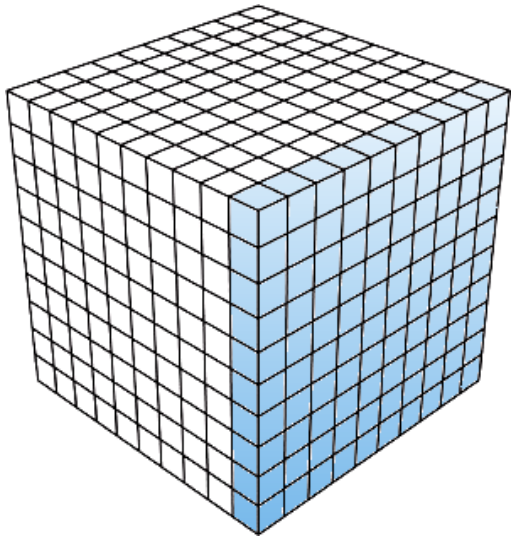


# Basic operations

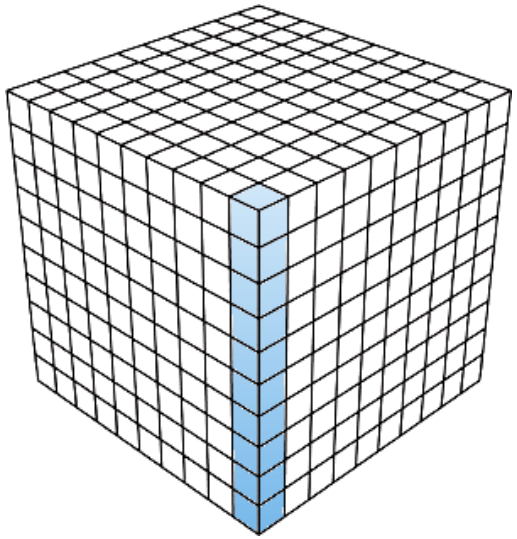
## Normalization



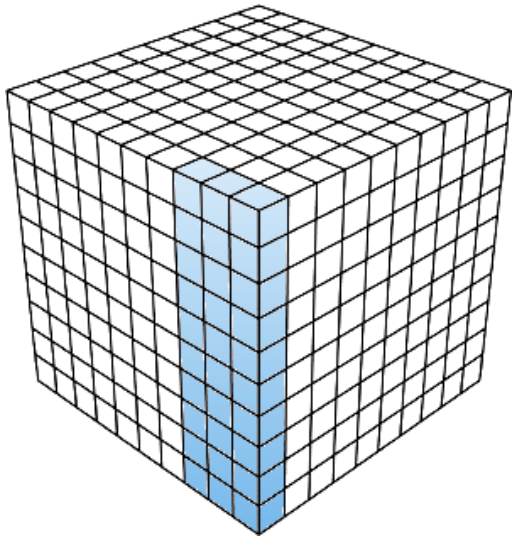
layer



batch



instance



group

# Feature extraction

# Feature extraction

**Illustration**

# Feature extraction: Keras

```
from keras.applications.xception import Xception
from keras.applications.mobilenet import preprocess_input
from keras.models import Model
import cv2
import numpy
from keras import backend as K
K.set_image_dim_ordering('tf')
# --------------------------------------------------------------------------------------------------------------
-------
def example_feature_extract_Keras():

    CNN = Xception()

    img = cv2.imread('data/ex-natural/dog/dog_0000.jpg')
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (224, 224)).astype(numpy.float32)

    model = Model(inputs=CNN.input, outputs=CNN.get_layer('predictions').output)
    prob  = model.predict(preprocess_input(numpy.array([img])))

    model = Model(inputs=CNN.input, outputs=CNN.output)
    prob  = model.predict(preprocess_input(numpy.array([img])))


    model = Model(inputs=CNN.input, outputs=CNN.get_layer('avg_pool').output)
    feature = model.predict(preprocess_input(numpy.array([img])))

    return
# --------------------------------------------------------------------------------------------------------------
-------
if __name__ == '__main__':
    example_feature_extract_Keras()
```

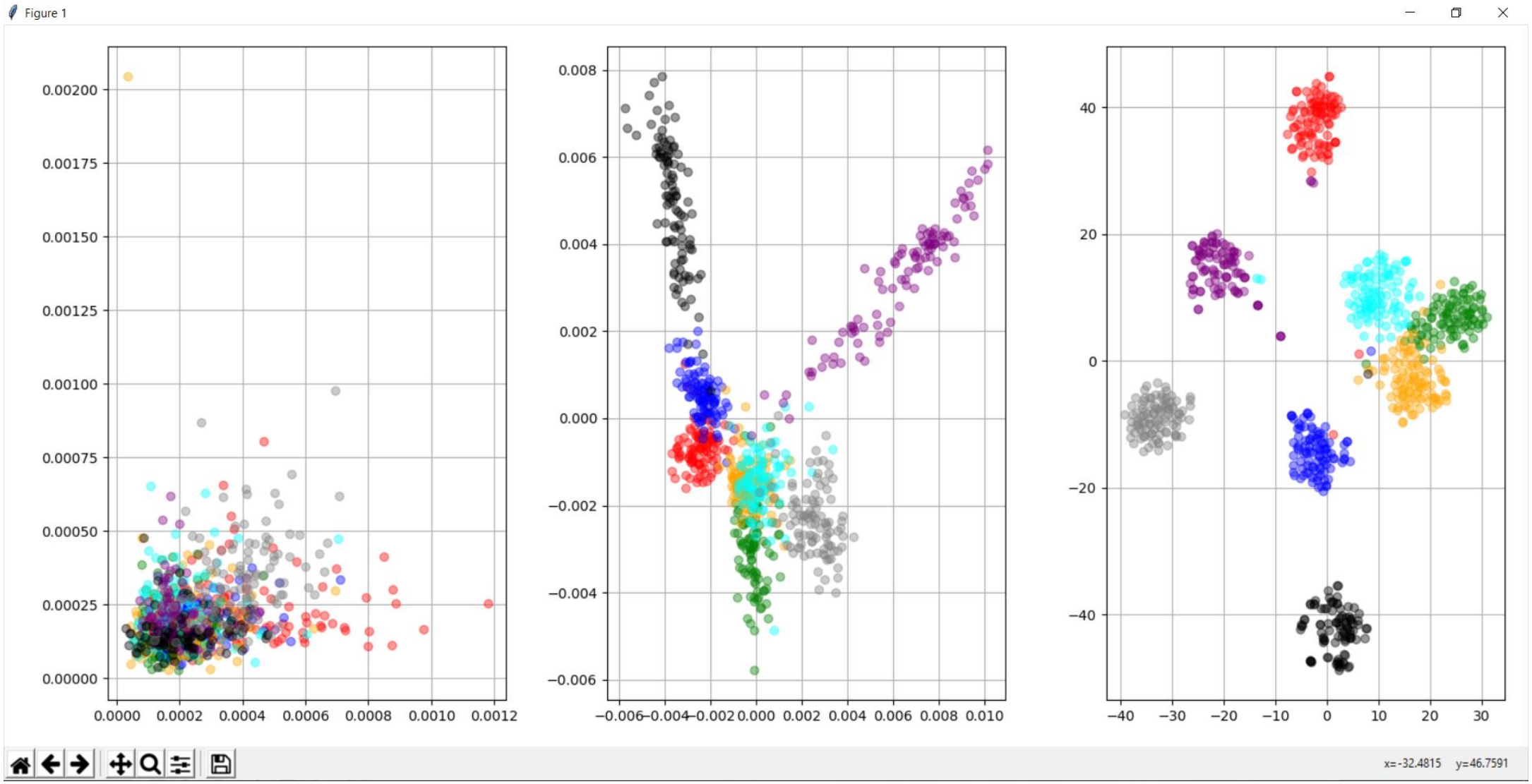
# Feature extraction: Tensorflow

```
import tensorflow_hub as hub
import cv2
import numpy
import tensorflow as tf
# ----------------------------------------------------------------------------------------------------
def example_feature_extract_TF():

    module = hub.Module("https://tfhub.dev/google/imagenet/resnet_v2_50/classification/1")
    img = cv2.imread('data/ex-natural/dog/dog_0000.jpg')
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (224, 224)).astype(numpy.float32)
    img = numpy.array([img]).astype(numpy.float32) / 255.0
    sess = tf.Session()
    sess.run(tf.global_variables_initializer())
    sess.run(tf.tables_initializer())

    outputs = module(dict(images=img), signature="image_classification", as_dict=True)
    prob = outputs["default"].eval(session=sess)[0]

    feature = module(img).eval(session=sess)

    sess.close()

    return
# ----------------------------------------------------------------------------------------------------
if __name__ == '__main__':
    example_feature_extract_TF()
```

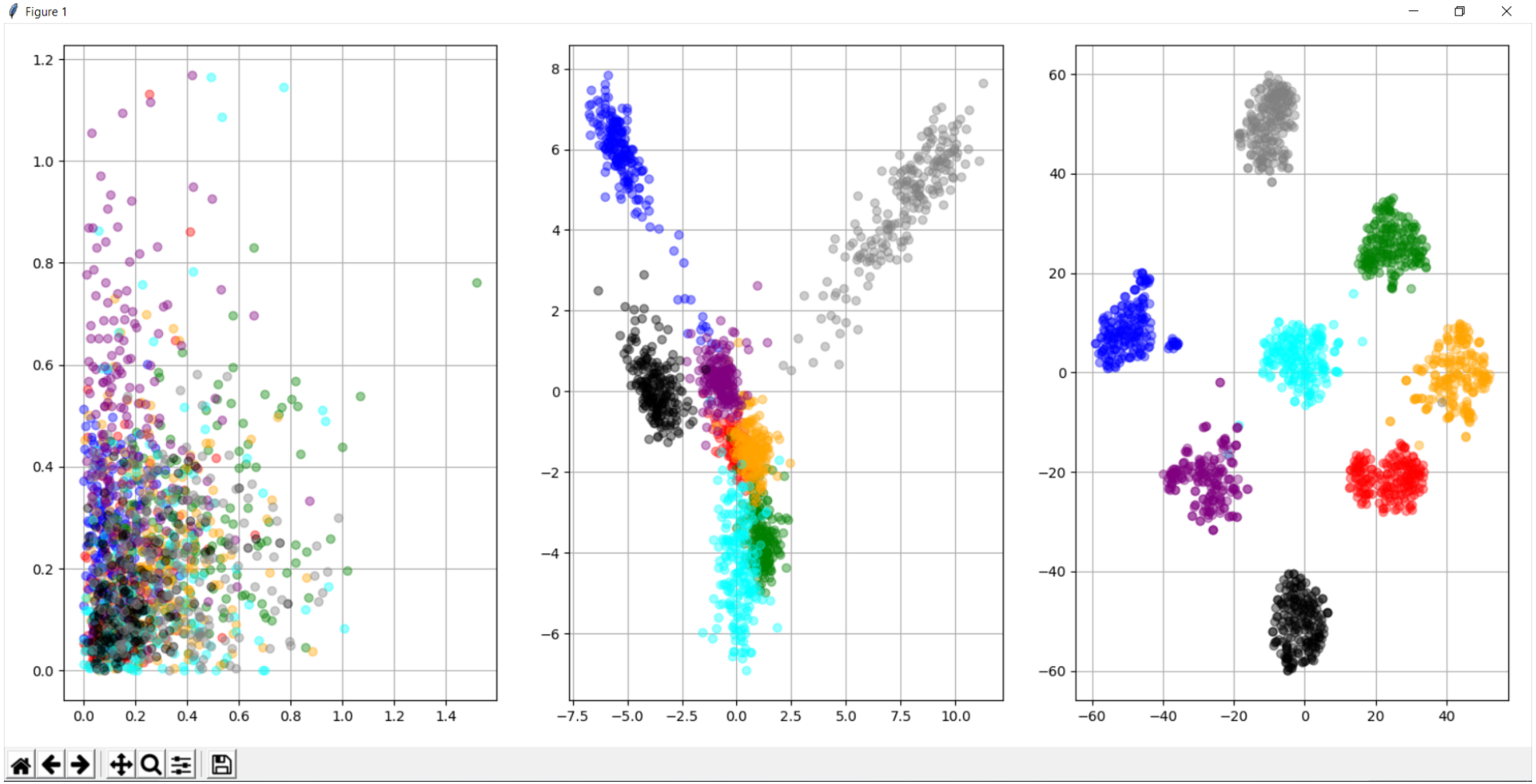
# Feature visualization

https://github.com/keras-team/keras/blob/master/examples/conv_filter_visualization.py

# Feature visualization



# Feature visualization



# Feature extraction: Tensorflow

```
# ----------------------------------------------------------------------------------------------------------------------
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import fnmatch
from os import listdir
from sklearn import decomposition
import numpy
# ----------------------------------------------------------------------------------------------------------------------
import tools_IO
import tools_ML
# ----------------------------------------------------------------------------------------------------------------------
def example_visualize_features():

    path_input= 'data/features/CNN_Inception_TF/'

    patterns = fnmatch.filter(listdir(path_input),'*.txt')
    for i in range (0,len(patterns)):
        patterns[i]=patterns[i].split('.')[0]

    patterns = numpy.array(patterns)
    ML = tools_ML.tools_ML(None)

    X,Y,  filenames = ML.prepare_arrays_from_feature_files(path_input,patterns=patterns,feature_mask='.txt')
    X_PC = decomposition.PCA(n_components=2).fit_transform(X)
    X_TSNE = TSNE(n_components=2).fit_transform(X)


    tools_IO.plot_2D_scores_multi_Y(plt.subplot(1,3,1),X,      Y)
    tools_IO.plot_2D_scores_multi_Y(plt.subplot(1,3,2),X_PC,   Y)
    tools_IO.plot_2D_scores_multi_Y(plt.subplot(1,3,3),X_TSNE, Y)
    plt.tight_layout()
    plt.show()

    return
# ----------------------------------------------------------------------------------------------------------------------
if __name__ == '__main__':

    example_visualize_features()
```

# Visualization of layers and filters

# Visualization of filters in Keras

```
for i in range (0, len(keras_model.layers)):
    tensor = keras_model.layers[i]
    if isinstance(tensor, Conv2D):
        tensor = tensor.get_weights()[0]
        tensor -= tensor.min()
        tensor *= 255.0 / tensor.max() #(W,H,3,N)

        if tensor.shape[2]==3:
            cv2.imwrite(path_out + 'filter_%03d.png'%i, tensor_color_4D_to_image(tensor))
        else:
            cv2.imwrite(path_out + 'filter_%03d.png'%i, tensor_gray_4D_to_image(tensor,do_colorize=True))
```

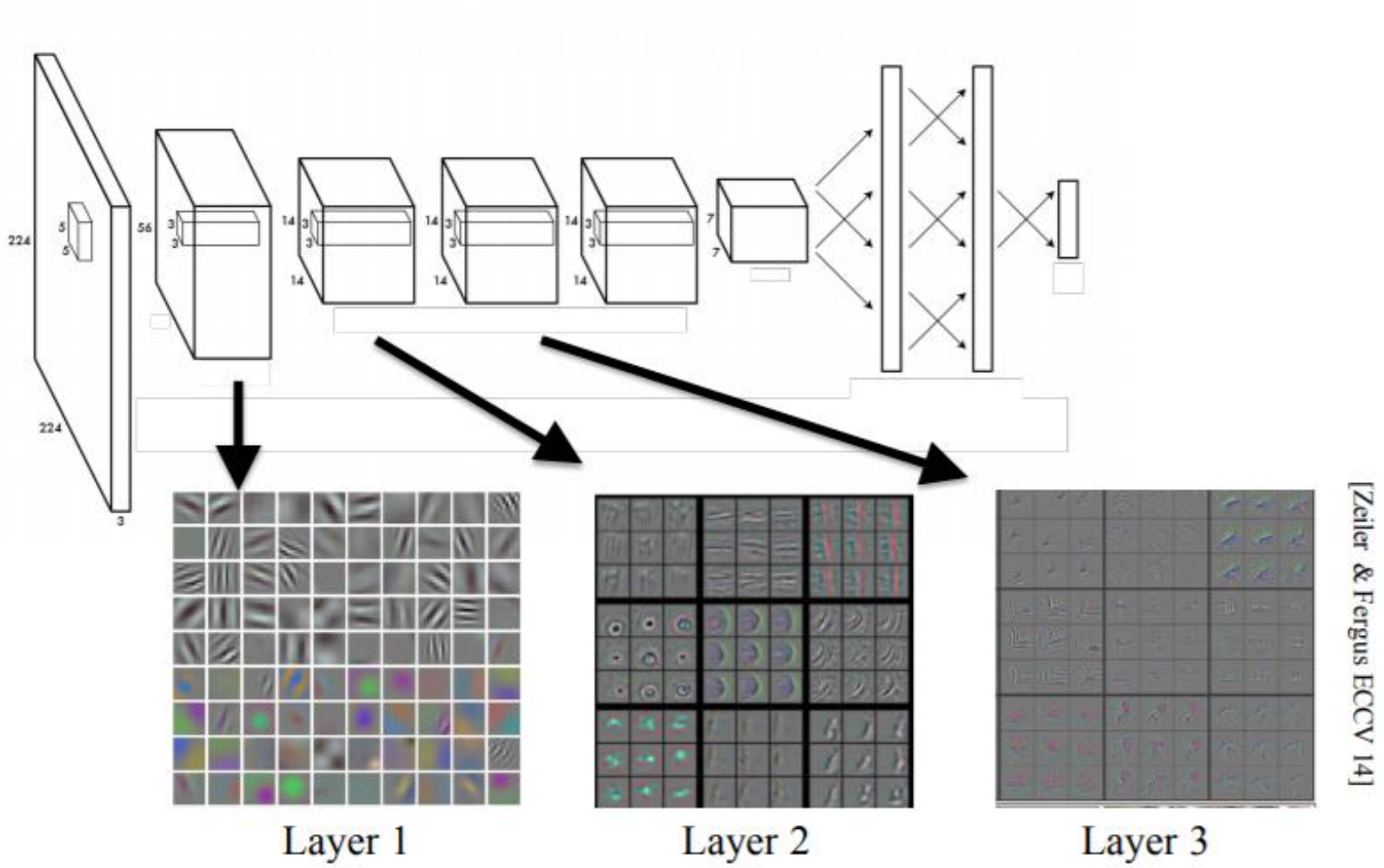
# Visualization of layers in Keras

```
layer_outputs = [layer.output for layer in keras_model.layers]
activation_model = Model(inputs=keras_model.input, outputs=layer_outputs)
activations = activation_model.predict(numpy.expand_dims(image, axis=0))


for i in range(0,len(activations)):
    tensor = activations[i][0]
    tensor -= tensor.min()
    tensor *= 255.0 / tensor.max()

    if len(tensor.shape) == 3:
        if need_transpose:
            tensor = tensor.transpose((1, 2, 0))

        if tensor.shape[2] == 3:
            cv2.imwrite(path_out + 'layer_%03d.png' % i, tensor)
        elif tensor.shape[2] != 3:
            cv2.imwrite(path_out + 'layer_%03d.png' % i, tensor_gray_3D_to_image(tensor,do_colorize=True))
    elif len(tensor.shape) == 1:
        cv2.imwrite(path_out + 'layer_%03d.png' % i,tools_image.hitmap2d_to_viridis(tensor_gray_1D_to_image(tensor)))
```

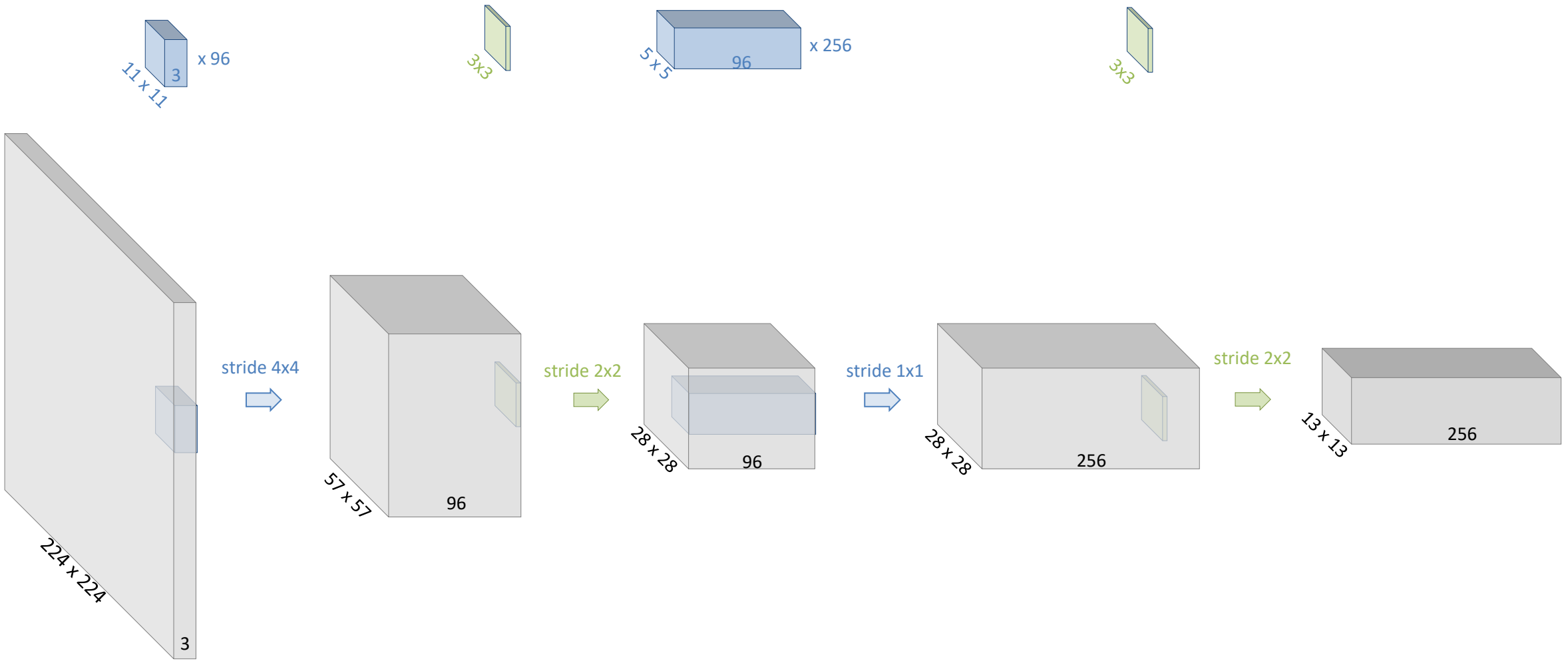
# Visualization of layers in Tensorflow

```
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

image = cv2.resize(cv2.imread(filename_input), (self.input_shape[0], self.input_shape[1]))

output = sess.run(self.layer_conv1, feed_dict={self.x: [image]})[0] #(57,57,96)
cv2.imwrite(path_output + 'layer01_conv1.png',tools_CNN_view.tensor_gray_3D_to_image(output,do_colorize=True))
```

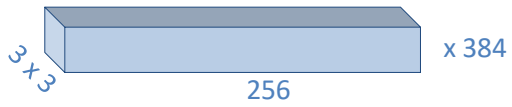
# Visualization of layers and filters

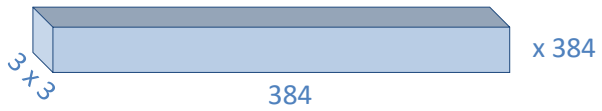


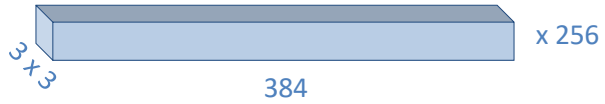
# Visualization of layers and filters

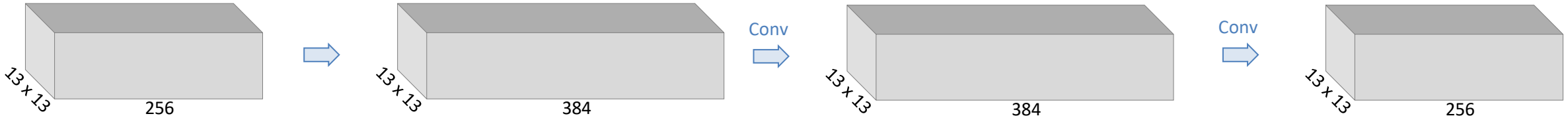


# Visualization of layers and filters

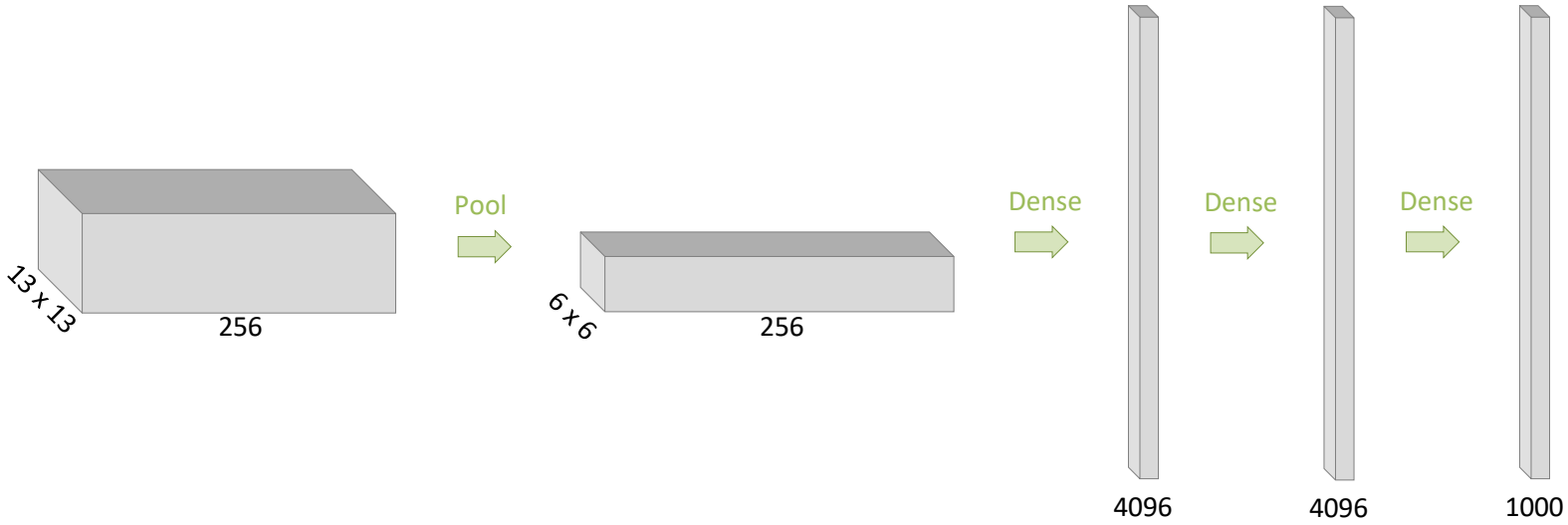


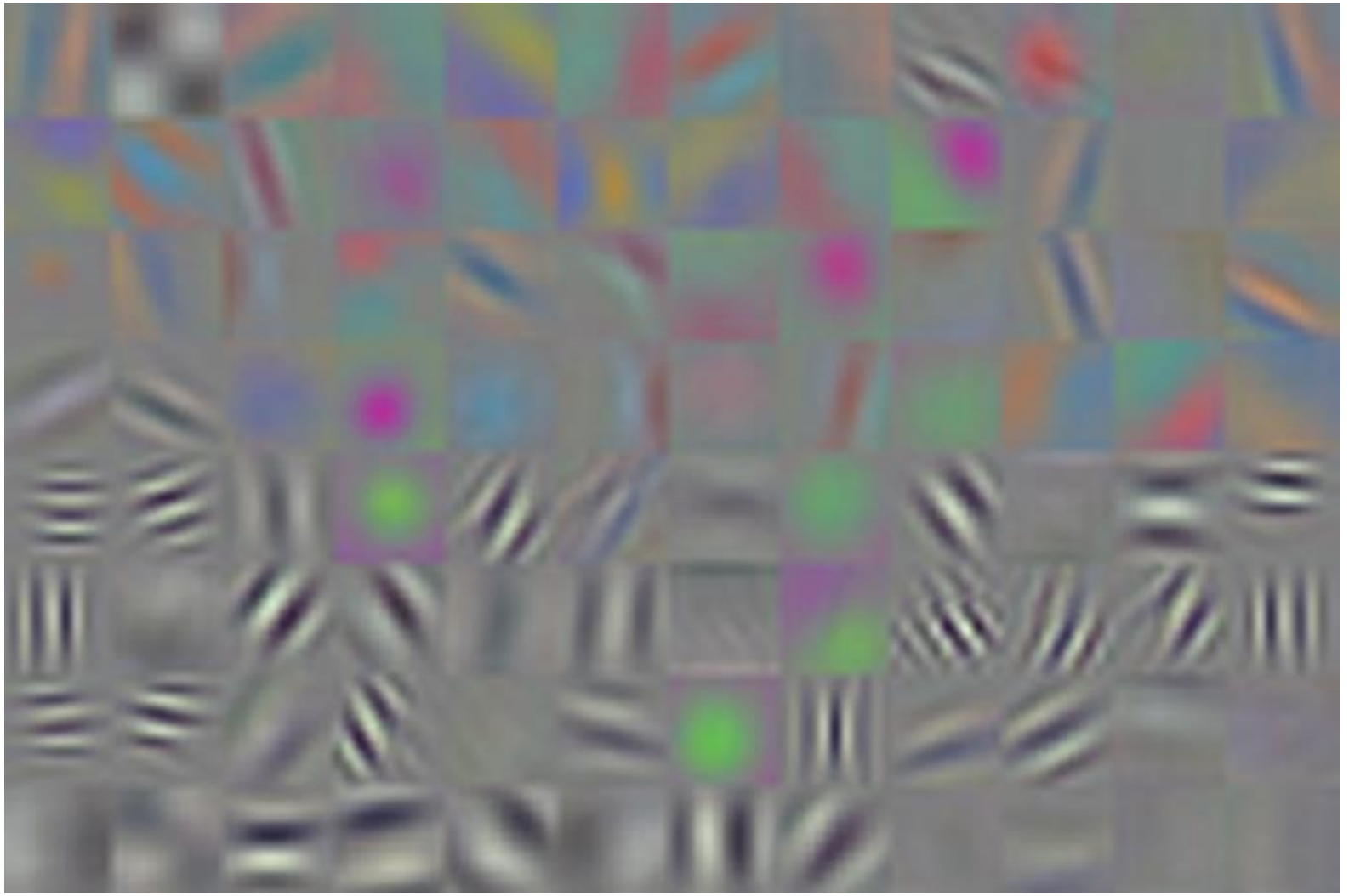



# Visualization of layers and filters

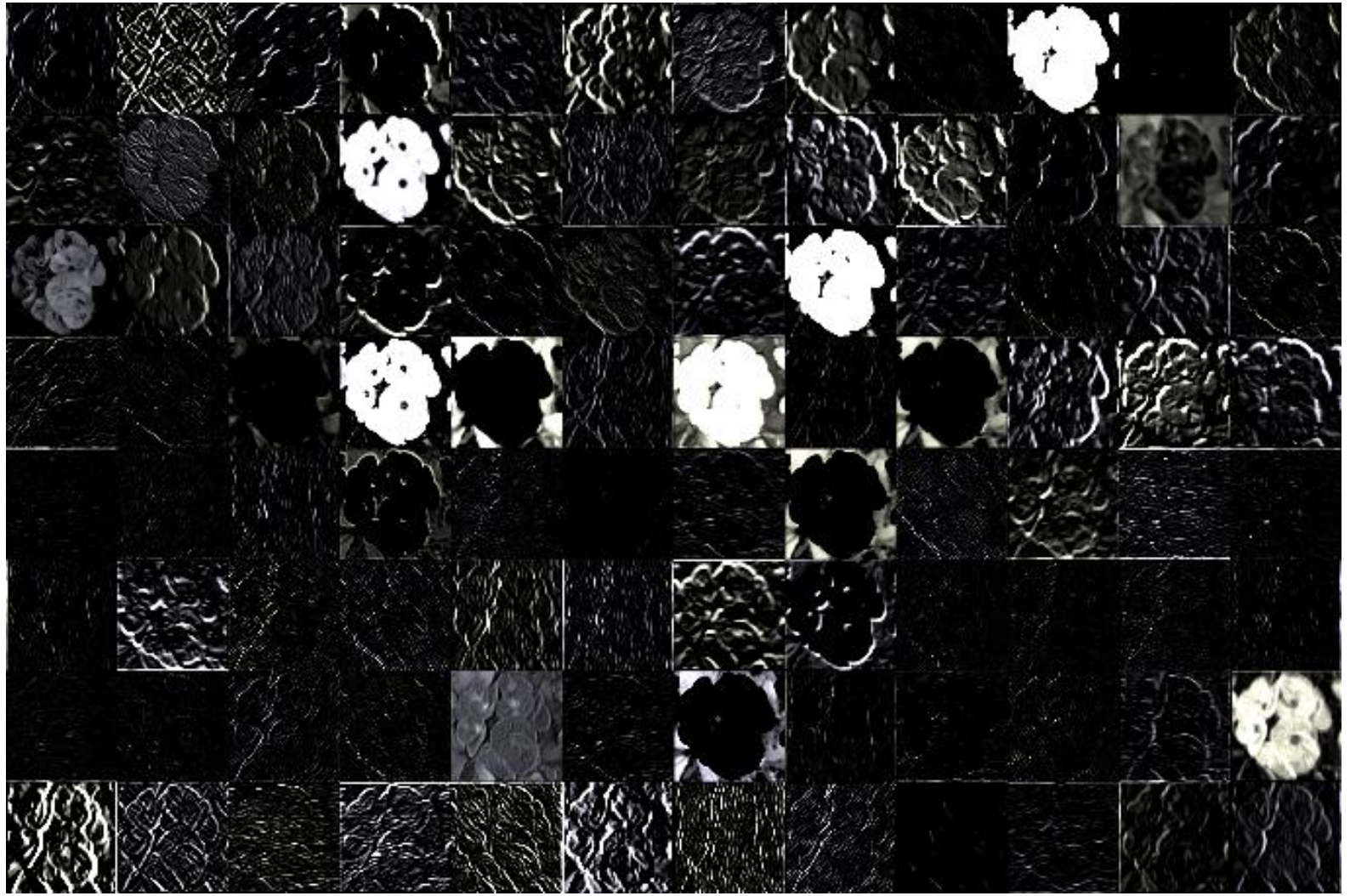


# Visualization of layers and filters

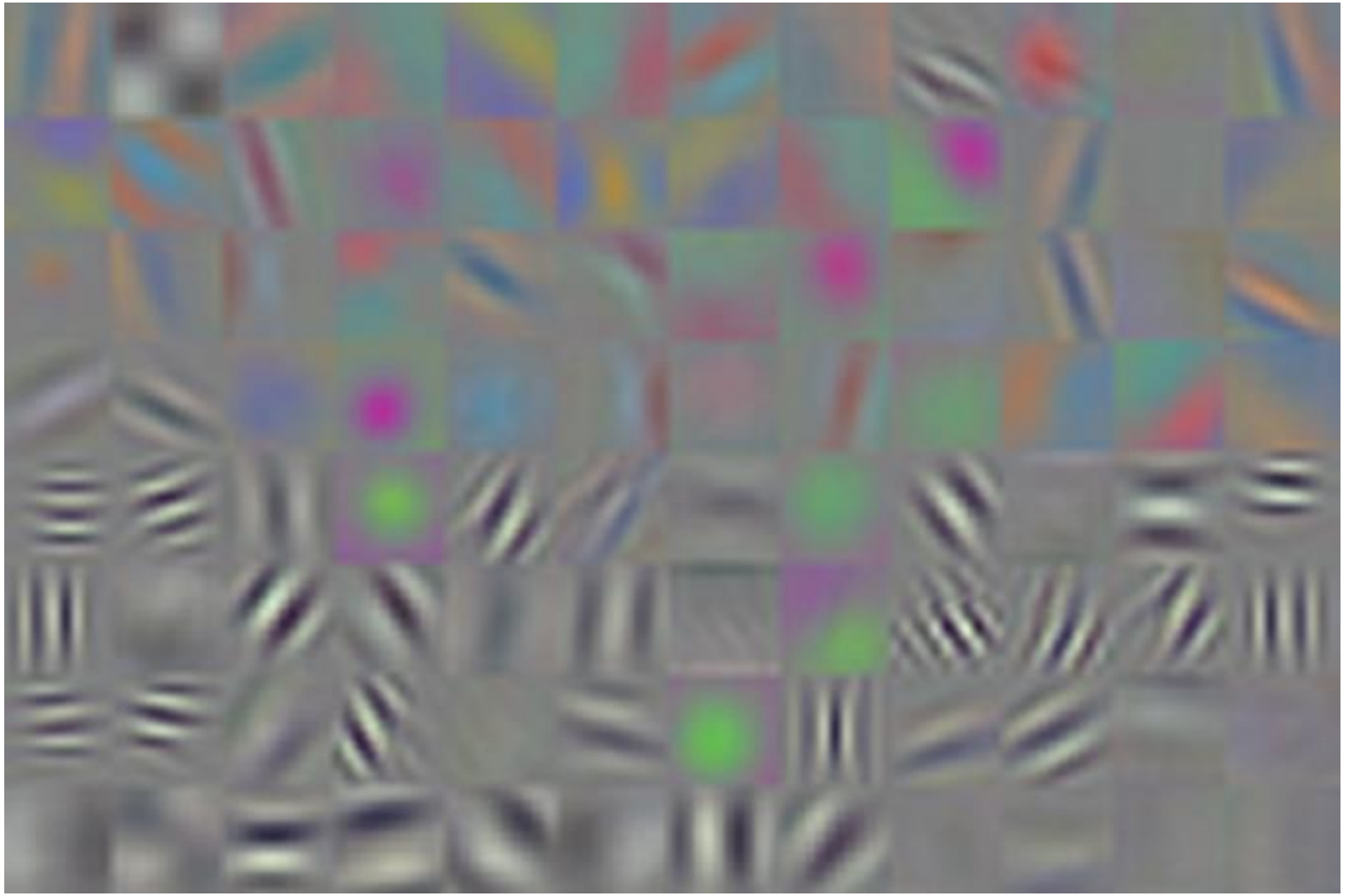
# Visualization of layers and filters



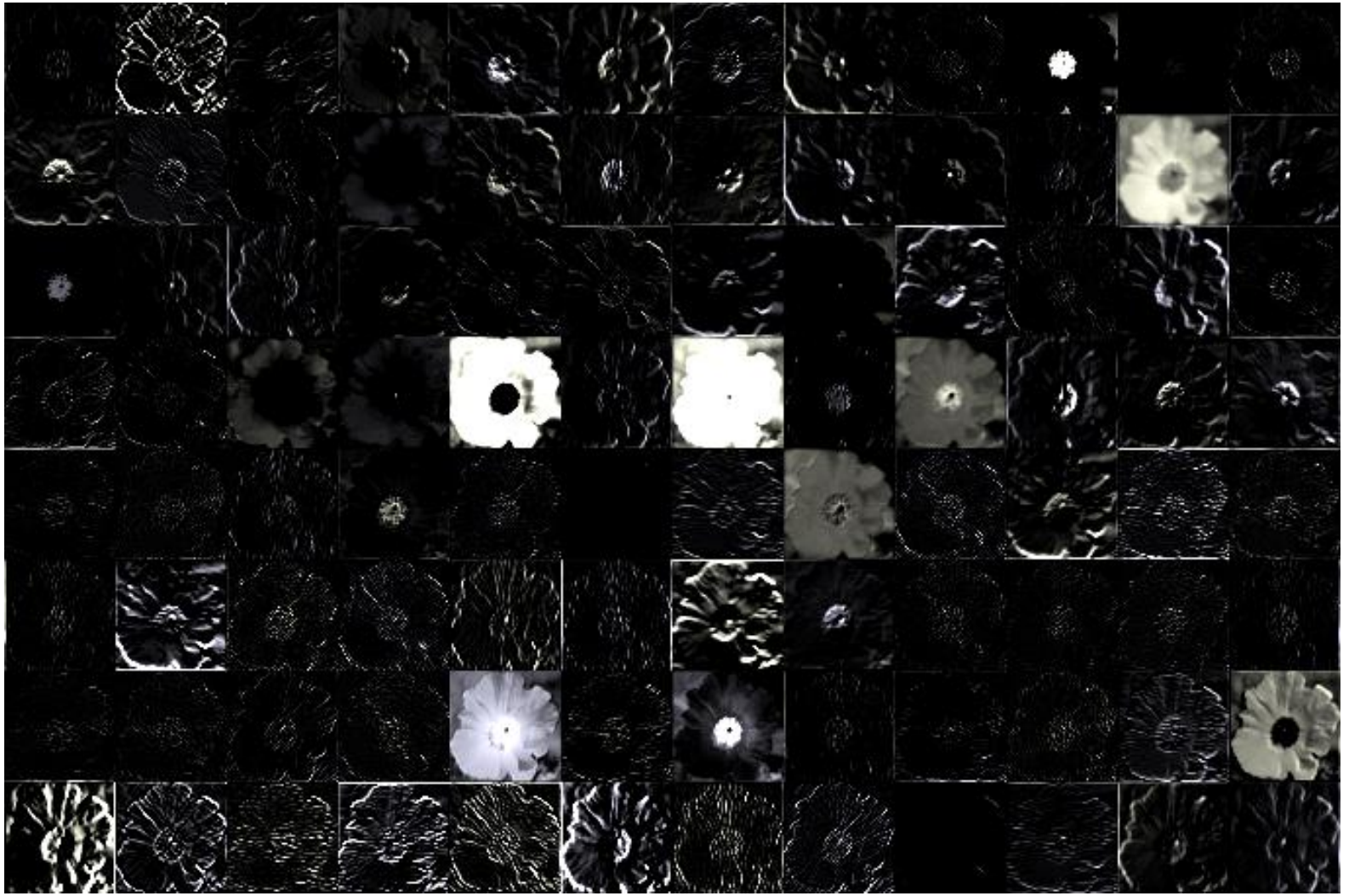
# Visualization of layers and filters



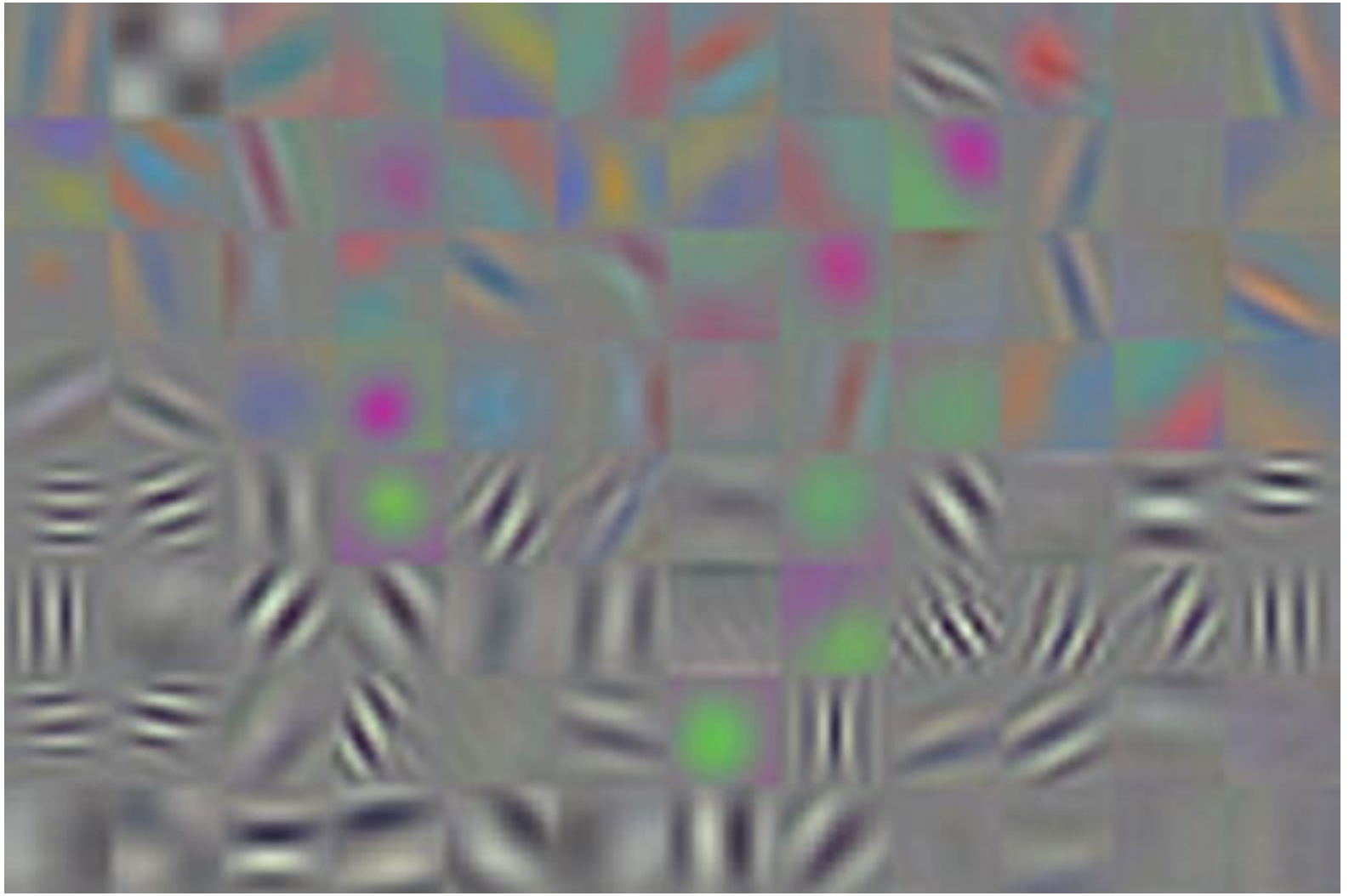
# Visualization of layers and filters



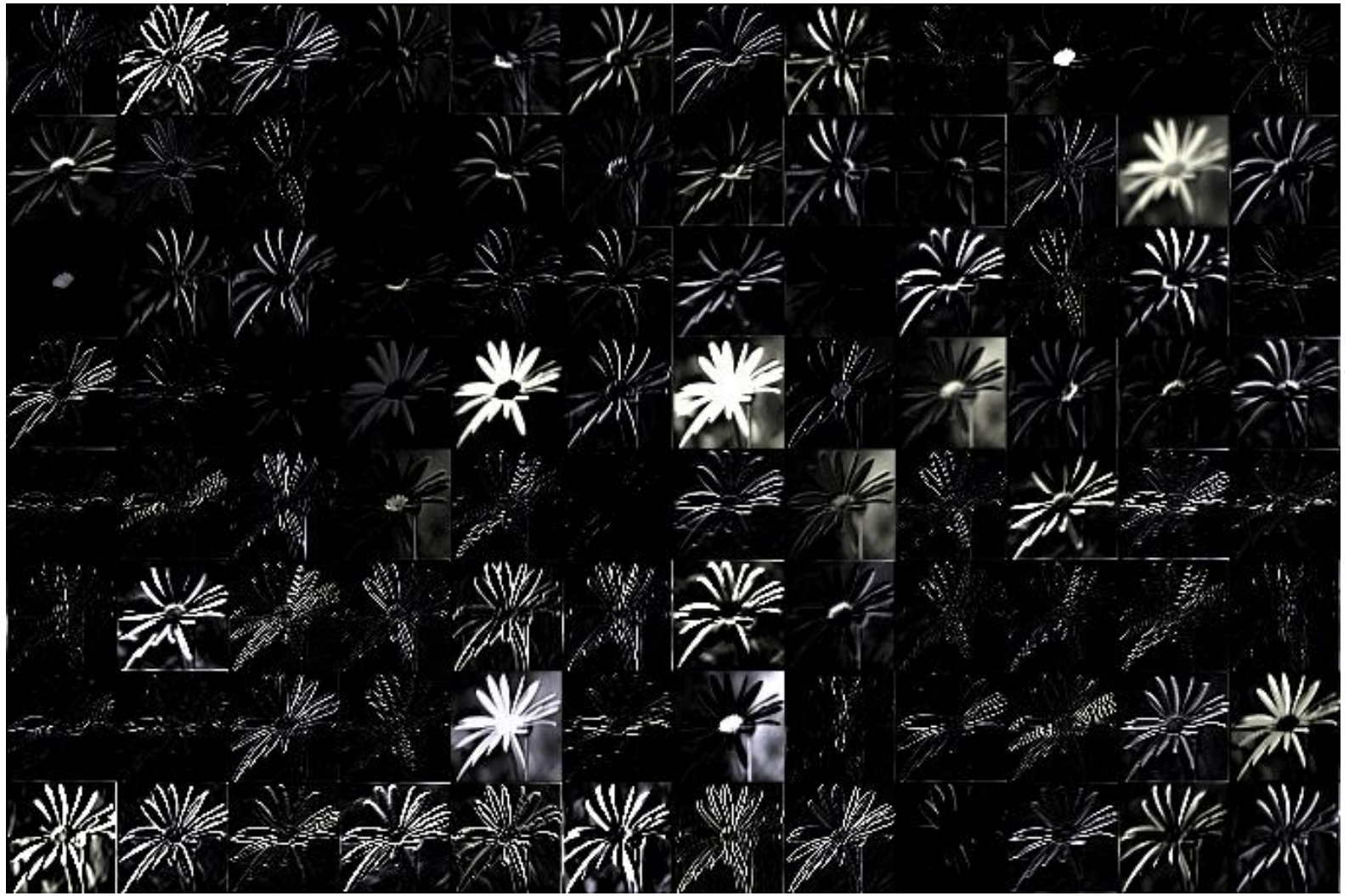
# Visualization of layers and filters



# Visualization of layers and filters



# Visualization of layers and filters



# Transfer learning

# Training the models

**Fine-tuning** is the process in which the parameters of a trained model must be adjusted very precisely while we are trying to validate that model taking into account a small data set that does not belong to the train set. That small validation data set comes from the same distribution as the data set used for the training of the model. The split of the available data to train and validation set is random.

https://stats.stackexchange.com/questions/343763/fine-tuning-vs-transferlearning-vs-learning-from-scratch

https://datascience.stackexchange.com/questions/22302/what-is-the-different-between-fine-tuning-and-transfer-learning

**Transfer Learning** or Domain Adaptation is related to the difference in the distribution of the train and test set. So it is something broader than Fine tuning, which means that we know a priori that the train and test come from different distribution and we are trying to tackle this problem with several techniques depending on the kind of difference, instead of just trying to adjust some parameters (usually we are doing this for reasons as preventing overfitting etc.)

# Training the models

https://www.tensorflow.org/hub/tutorials/image_retraining

https://github.com/tensorflow/hub/blob/master/examples/image_retraining/retrain.py

https://medium.com/tensorflow/introducing-tensorflow-hub-a-library-for-reusable-machine-learning-modules-in-tensorflow-cdee41fa18f9

https://github.com/tensorflow/hub/tree/master/examples/colab

# Fine tuning

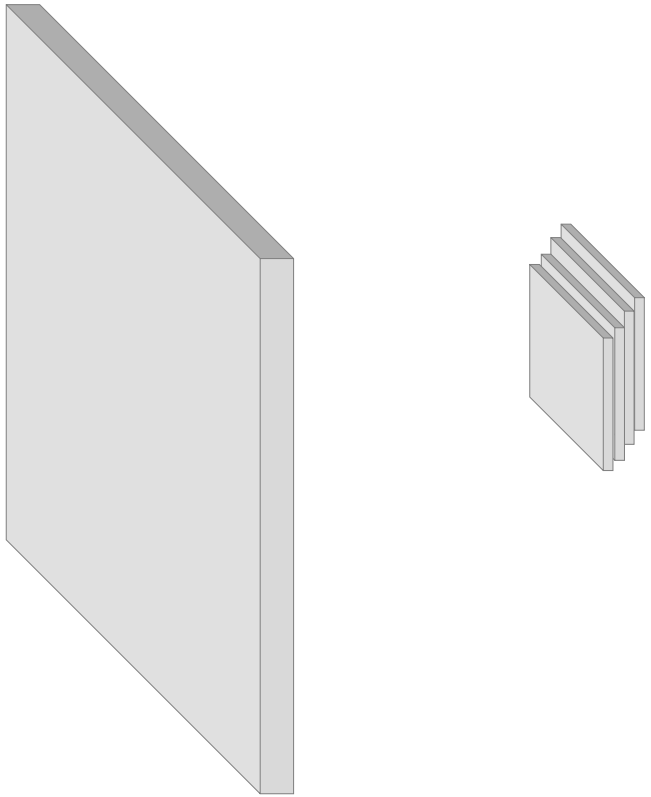
https://www.tensorflow.org/hub/fine_tuning

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

https://keras.io/applications/#fine-tune-inceptionv3-on-a-new-set-of-classes

# Architecture of the popular networks

# Architecture of the popular networks: AlexNet



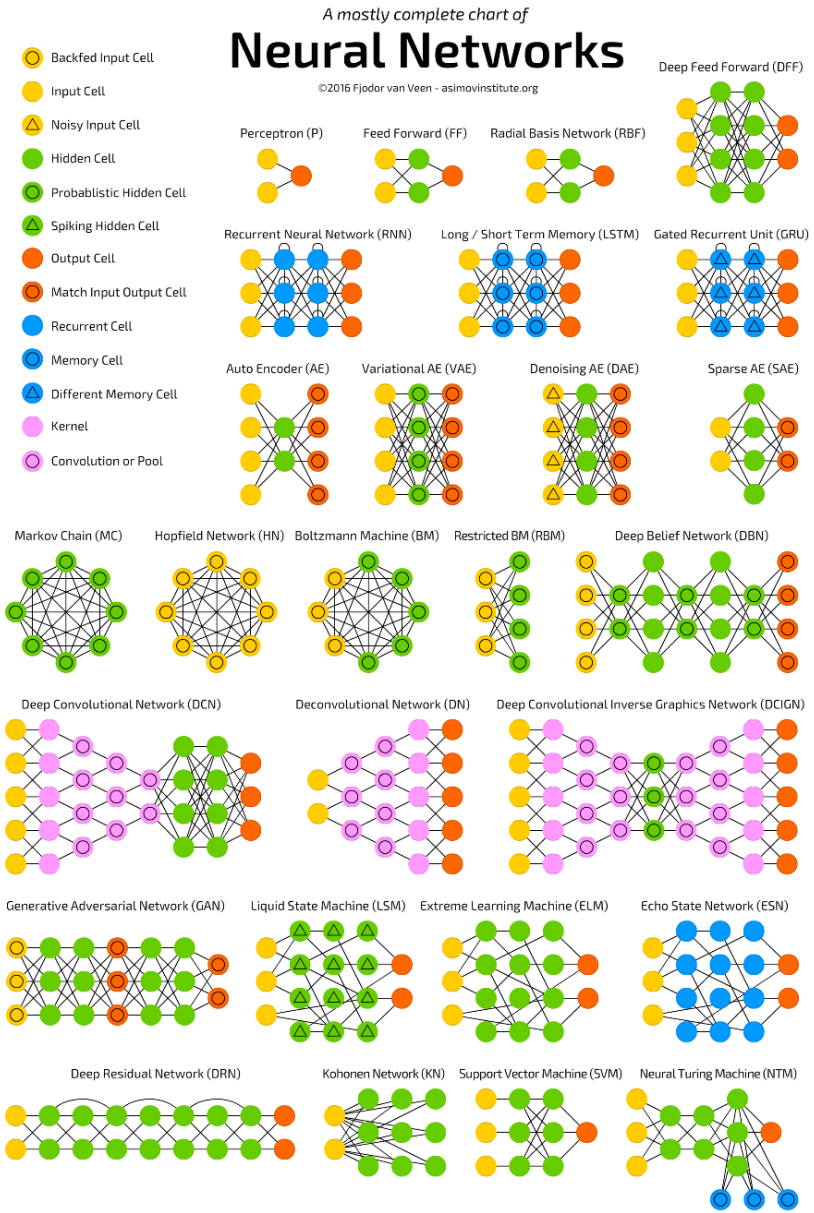
224 x 224 x 3



11 x 11 x 96

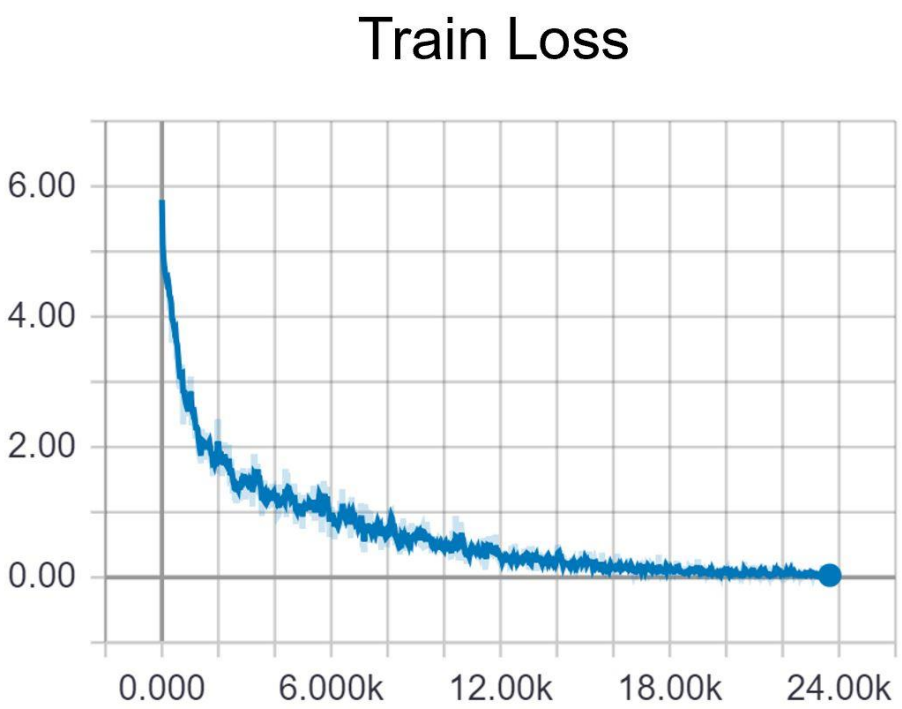
https://lutzroeder.github.io/netron/

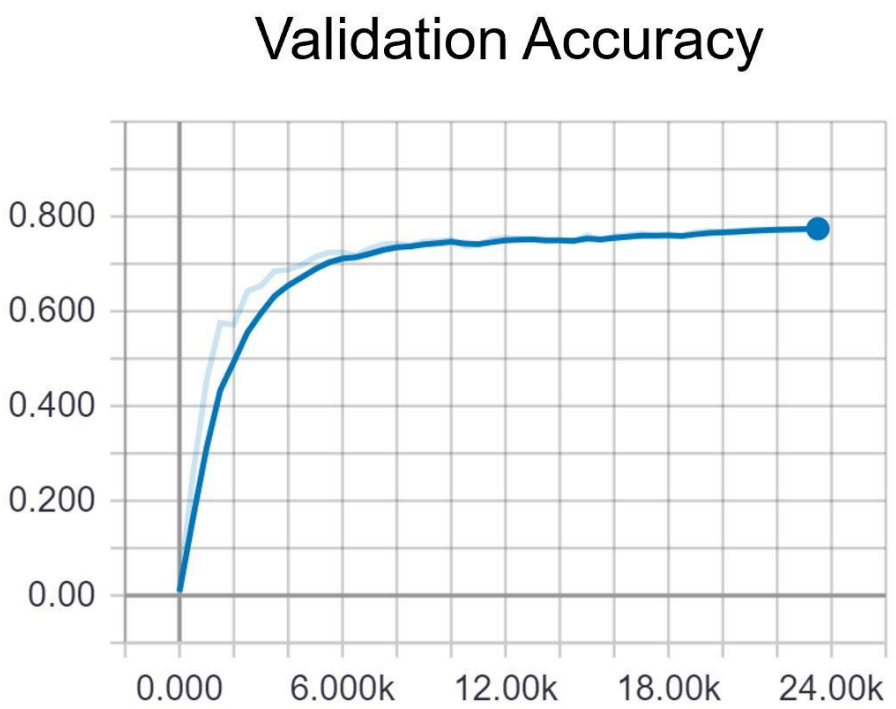
# Architecture of the popular networks: AlexNet



# Build, Train and Test own CNN

# Build, Train and Test own CNN




Epoch

# Build, Train and Test own CNN

graph

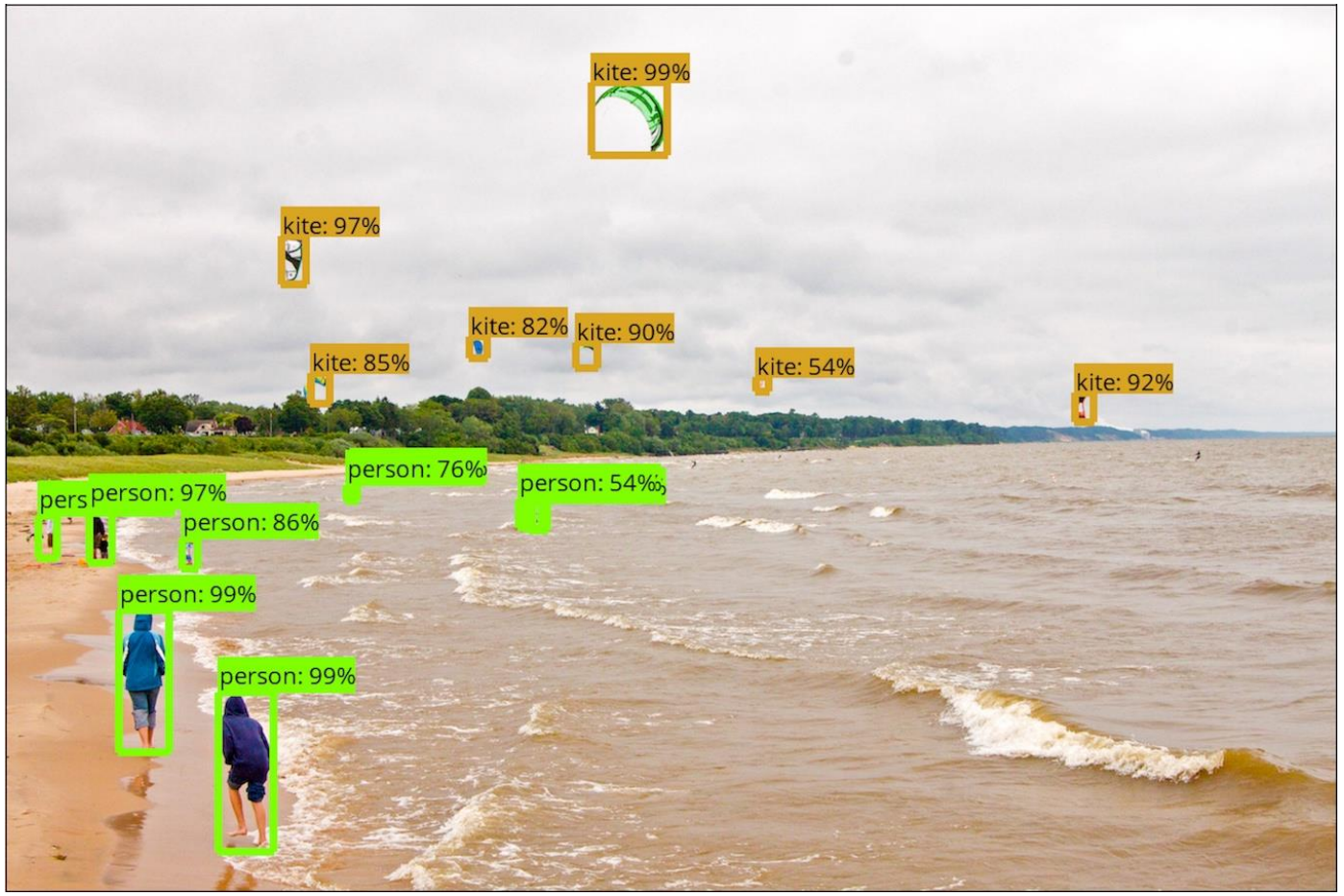
# Build, Train and Test own CNN

Tensorflow vs Keras vs PyTorch vs Caffe vs Teano

# Benchmarking the CNNs

# Object detection

# Object detection




https://github.com/tensorflow/models/tree/master/research/object_detection

# Auto Encoders

https://blog.keras.io/building-autoencoders-in-keras.html

# Style transfer

# CAM

Class Activation Mapping

# CAM

•The class activation map highlights the most informative image regions relevant to the predicted class. This map can be obtained by adding a global average pooling layer at the end of convolutional layers.
•Details of the implementation and more results can be found here. Some results:

# References


https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv

http://cs231n.stanford.edu/2017/syllabus.html

https://www.jeremyjordan.me/convnet-architectures/

https://medium.com/activewizards-machine-learning-company/data-science-for-managers-mindmap-bb95bedc3154

https://habr.com/ru/company/cloud4y/blog/346968/

https://dou.ua/lenta/digests/ai-ml-digest-9/

https://ai.stackexchange.com/questions/5546/what-is-the-difference-between-a-convolutional-neural-network-and-a-regular-neur

# GAN

https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

https://www.lyrn.ai/2018/12/26/a-style-based-generator-architecture-for-generative-adversarial-networks/