

Machine Learning II

“Classic” Deep Learning & Images – Convolutional Neural Networks



Dr. Elena Gavagnin

elena.gavagnin@zhaw.ch

FS2025

Machine Learning II – Images

Plan:

1. Artificial Neural Networks – Introduction, implementation with TF/Keras and Training
2. "Classic" Deep Learning & Images – Convolutional Neural Networks and Transfer Learning
3. NEW: Multimodal and generative AI

"Classic" Deep Learning & Images (W8 and W9):

1. Working with visual data
2. What is a convolution?
3. Convolutional Neural Networks (CNN)
4. Famous CNN Architectures
5. Reusing Pre-Trained Layers: Transfer Learning and Fine Tuning
6. Visual Data preparation

From Last week:

The Neural Network Real World with TF/Keras Summary

Create a NN Model

```
model = tf.keras.models.Sequential([  
    tf.keras.layers.Dense(N_Neurons_H1, activation= activ_f1, input_dim = num_features),  
    tf.keras.layers.Dense(N_Neurons_H2, activation= activ_f2),  
    tf.keras.layers.Dense(N_Neurons_H3, activation= activ_f3),  
    tf.keras.layers.Dense(N_Neurons_Out, activation= activ_f_out)  
])
```

Configure Model's losses & evaluation metrics

```
model.compile(loss=loss_function,optimizer=how_to_descend,  
metrics=what_to_check)
```

Train the model

```
model.fit(X_train, y_train, epochs=N_epochs,  
          validation_data=(X_valid, y_valid))
```

Evaluate the model

```
model.evaluate(X_test, y_test)
```

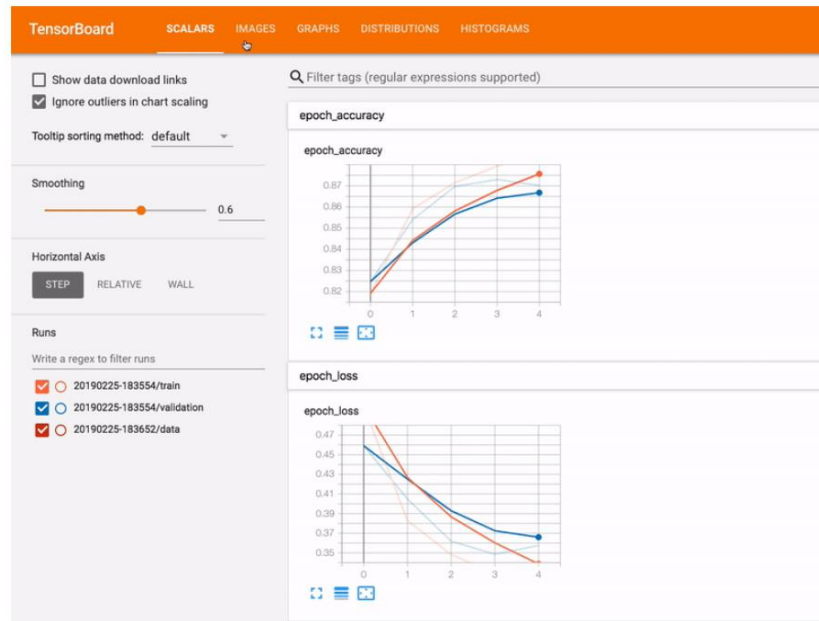
TensorBoard

Live visualisation tool for your model

- Tracking and visualizing metrics such as loss and accuracy
- Visualizing the model graph
- Viewing histograms of weights, biases, or other tensors as they change over time
- Displaying images, text, and audio data

```
tensorboard_cb =  
tf.keras.callbacks.TensorBoard(run_logdir)  
  
model.fit(x=x_train,  
          y=y_train,  
          epochs=5,  
          validation_data=(x_test, y_test),  
          callbacks=[tensorboard_cb])
```

<https://www.tensorflow.org/tensorboard>



Over SGD: Faster Optimizers

KERAS:

```
tf.  
optimizer = keras.optimizers.SGD(lr=0.001, momentum=0.9)
```

```
tf.  
optimizer = keras.optimizers.RMSprop(lr=0.001, rho=0.9)
```

```
tf.  
optimizer = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999)
```

Big Choice: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers

Must be passed in
model.compile()



```
model.compile(loss="sparse_categorical_crossentropy",  
              optimizer="sgd",  
              metrics=["accuracy"])
```

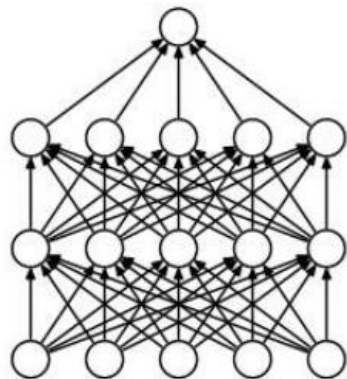
Regularization against Overfitting

L2 (and L1) Regularization:

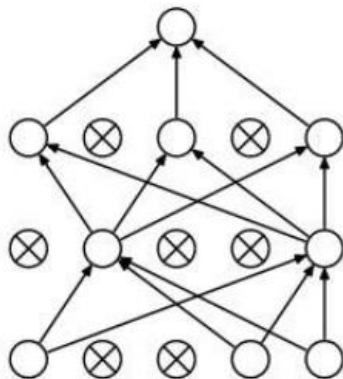
```
tf.keras.layers.Dense(64, activation='relu', kernel_regularizer = tf.keras.regularizers.L2(0.01))
```

https://www.tensorflow.org/api_docs/python/tf/keras/regularizers

Dropout



Standard Neural Net



After applying dropout.

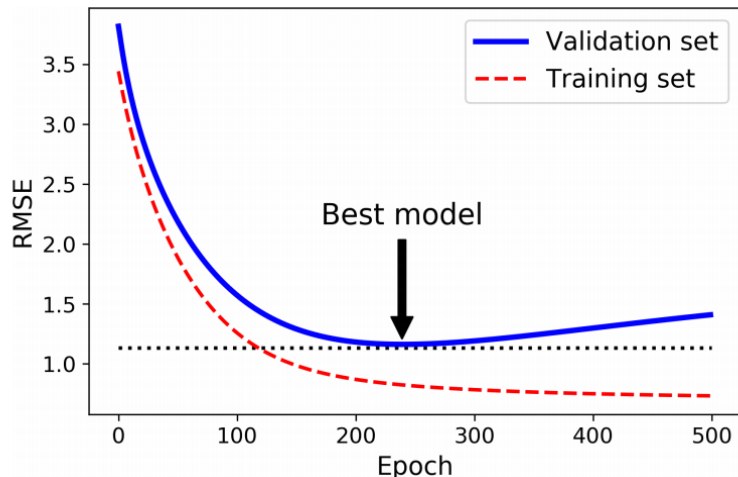
To every node is associated a probability to “go off” or stay on

Training on a randomly smaller network

```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    tf.keras.layers.Dropout(rate=0.2),
    keras.layers.Dense(300, activation="elu", kernel_initializer="he_normal"),
    keras.layers.Dropout(rate=0.2),
    keras.layers.Dense(100, activation="elu", kernel_initializer="he_normal"),
    keras.layers.Dropout(rate=0.2),
    keras.layers.Dense(10, activation="softmax")
])
```

*= only during training

Early Stopping



Idea: Stop training as soon as validation reaches the minimum

Are we really really sure it is the minimum??
Have some **patience!**

```
tf.  
early_stopping_cb = keras.callbacks.EarlyStopping(patience=10,  
                                                    restore_best_weights=True)  
history = model.fit(X_train, y_train, epochs=100,  
                    validation_data=(X_valid, y_valid),  
                    callbacks=[checkpoint_cb, early_stopping_cb])
```

Hands-on Session

Hands-on Session

Exercise: Application to a tiny bit more difficult case

Resume your notebook from last week “1.MNIST_first_steps.ipynb”.

Apply the same code in the notebook to a new dataset, the CIFAR10 Dataset.

(<https://www.cs.toronto.edu/~kriz/cifar.html>).

This also contains 10 categories of different objects (RGB images). This dataset is available to be loaded directly from Tensorflow: https://www.tensorflow.org/api_docs/python/tf/keras/datasets/cifar10

Run the same architecture with this new dataset and check the performance respect to MNIST.

To do this step you need though to adapt the input to the network. In this case the pictures have 32x32 pixels and 3 color channels (R,G,B). Take into consideration for the input layer.

How is the performance?

Deep Computer Vision: Convolutional Neural Networks

Image Classification & other stories



What the computer sees

image classification →
82% cat
15% dog
2% hat
1% mug

Credits: A. Karpathy



Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

Classification + Localization



CAT

Single Object

Object Detection



DOG, DOG, CAT

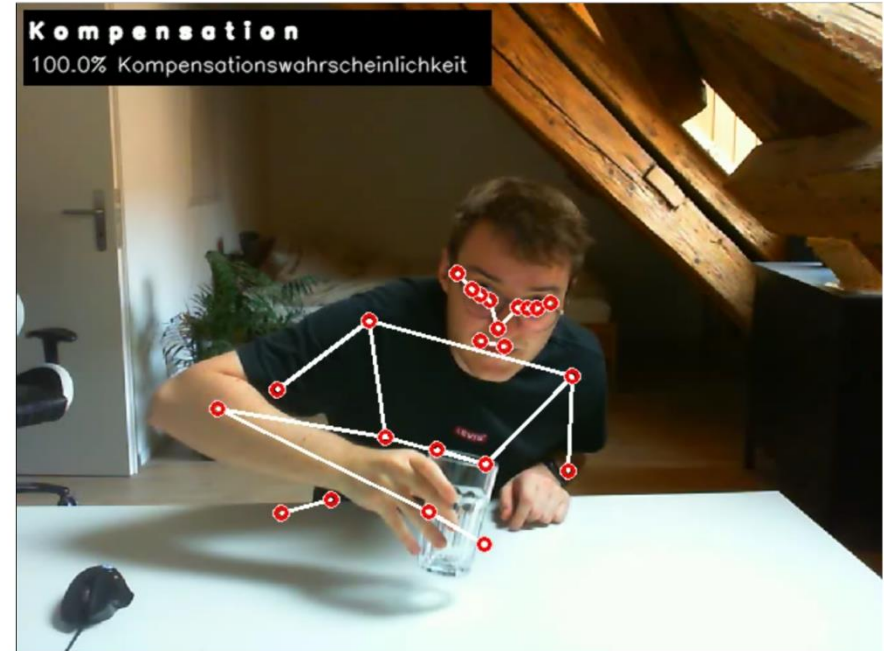
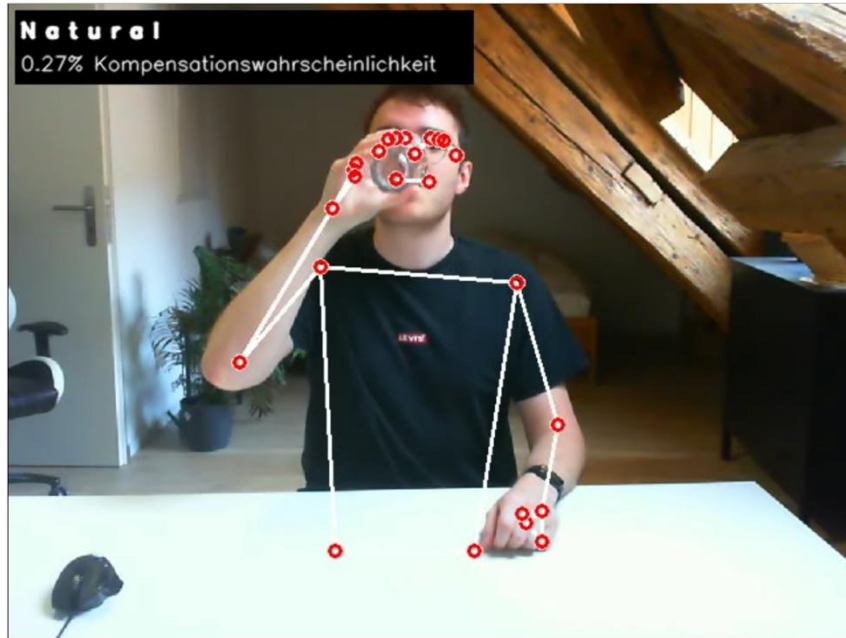
Instance Segmentation



DOG, DOG, CAT

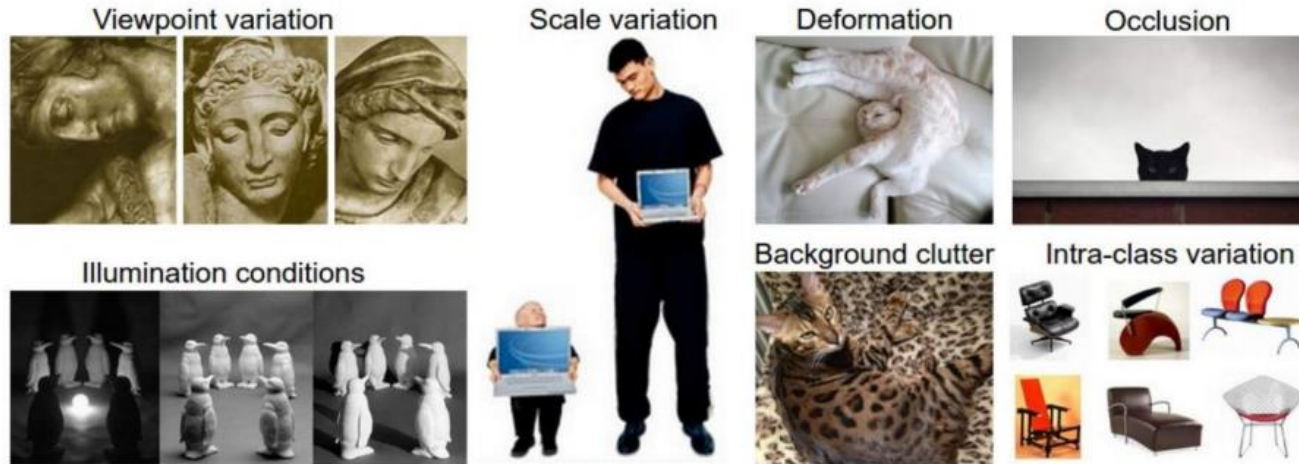
Multiple Object

Image Classification & other stories



Credits: Bühner Nils

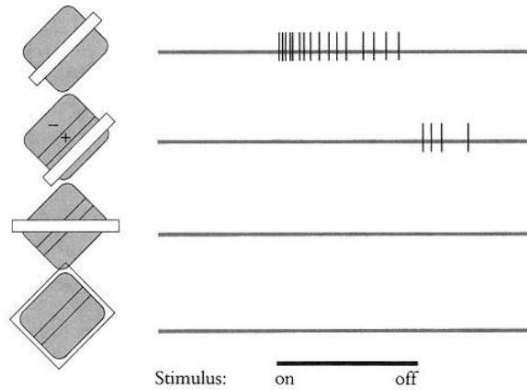
Not so easy...



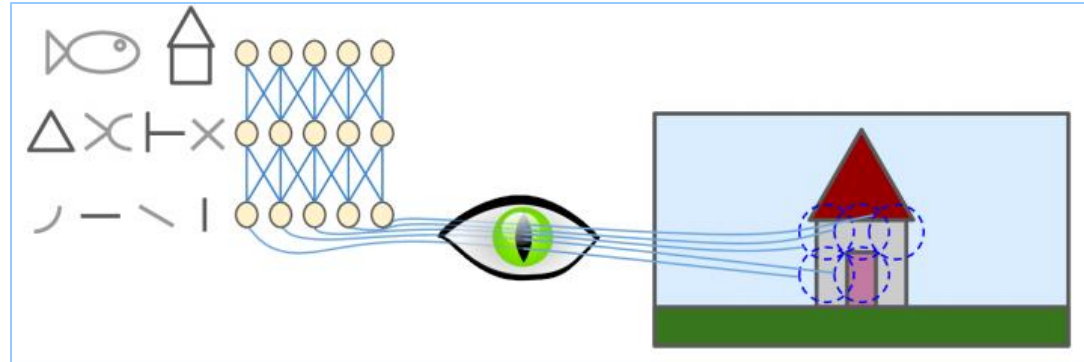
Credits: A. Karpathy

How do we do it..?

Take-Home Message: Visual System as a Hierarchy of Feature Detectors



Orientation selectivity: Most V1 neurons are orientation selective meaning that they respond strongly to lines, bars, or edges of a particular orientation (e.g., vertical) but not to the orthogonal orientation (e.g., horizontal).



Many neurons in the visual cortex have a small local receptive field, -> they react only to visual stimuli located in a limited region of the visual field

- “Single Unit Activity in Striate Cortex of Unrestrained Cats,” D. Hubel and T. Wiesel (1958).
- “Receptive Fields of Single Neurones in the Cat’s Striate Cortex,” D. Hubel and T. Wiesel (1959).
- “Receptive Fields and Functional Architecture of Monkey Striate Cortex,” D. Hubel and T. Wiesel (1968).

Yann LeCun (1998)

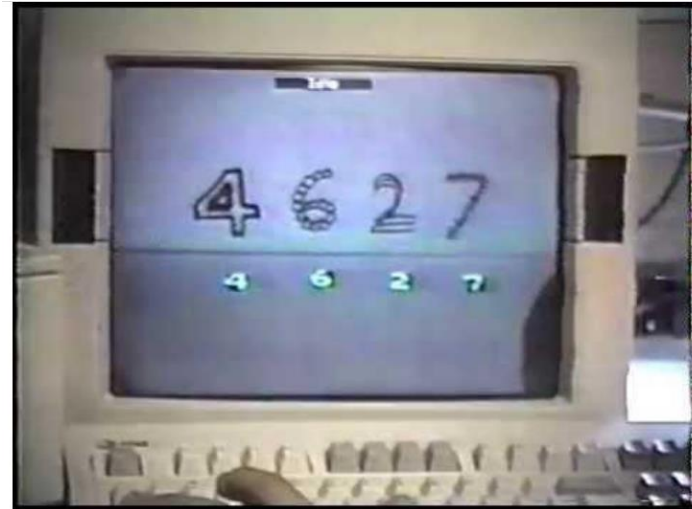
LeNet-5

Training: 60,000 examples

Test: 10,000 examples

Test error rates (no pre-proc)

- Linear Classifier: 8.4 %
- Non-linear Classifier: 3.3 %
- K-Nearest Neighbors: 3.1 %
- SVM: 1.4 %
- **Neural Nets (NN): 0.83 %**
- **Convolutional NN: 0.35 %**



<http://yann.lecun.com/exdb/mnist/>

ImageNet: the turning point

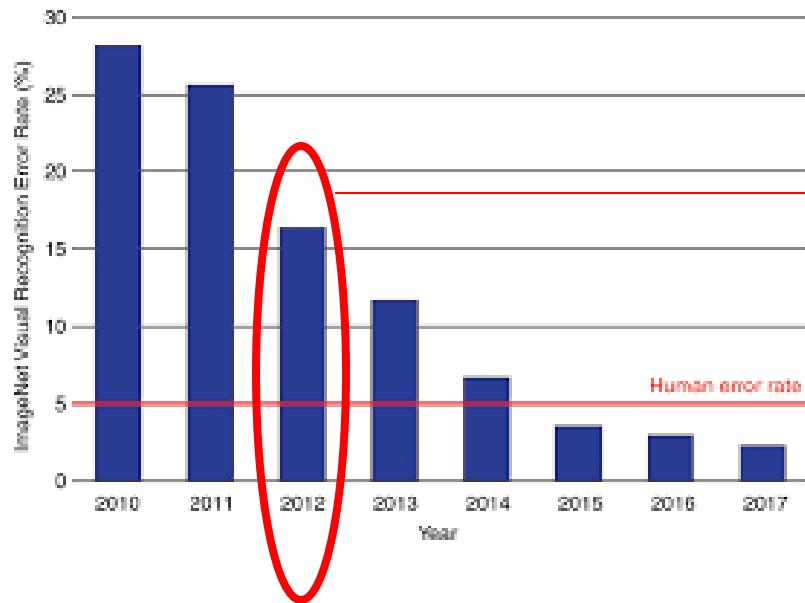
IMAGENET



[Deng et al. CVPR 2009]

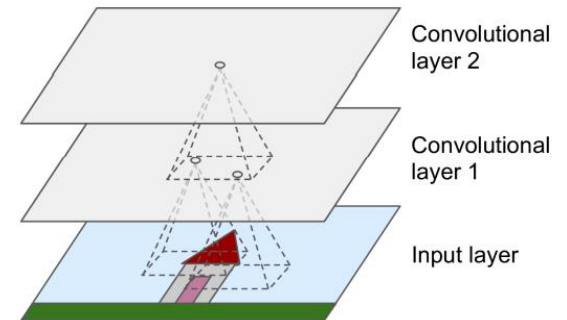
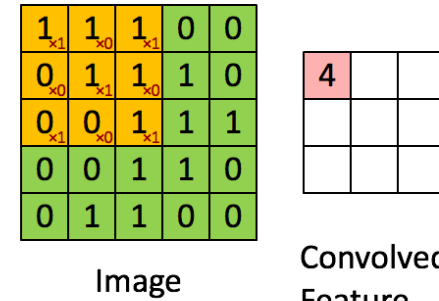
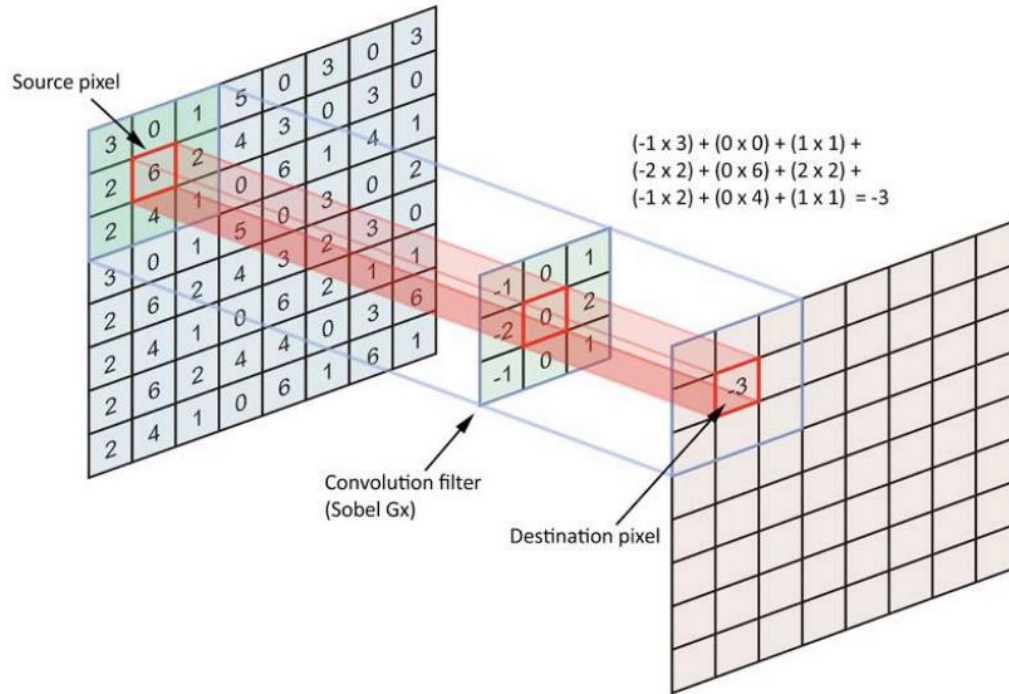
- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon Turk
- Challenge: 1.2 million training images, 1000 classes

ImageNet: the turning point

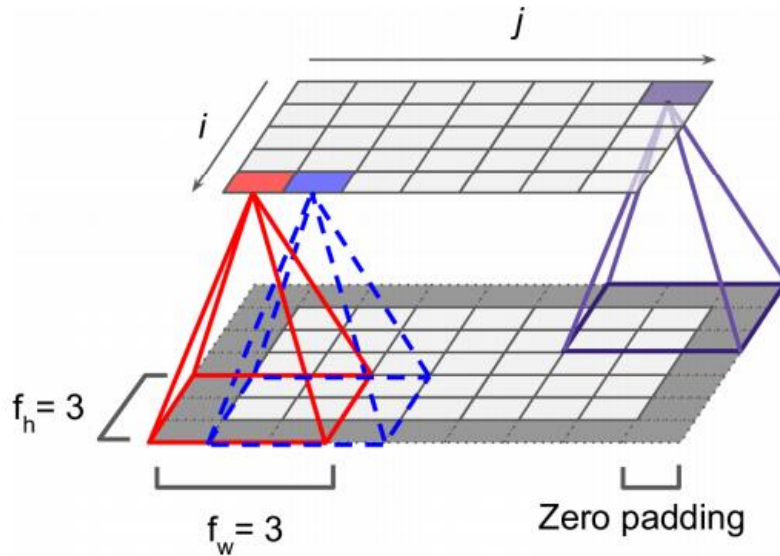


In 2012, a convolutional neural network (CNN) called AlexNet achieved a top-5 error of 15.3% in the ImageNet 2012 Challenge using GPUs, more than 10.8 percentage points lower than that of the runner up (non-ConvNet).

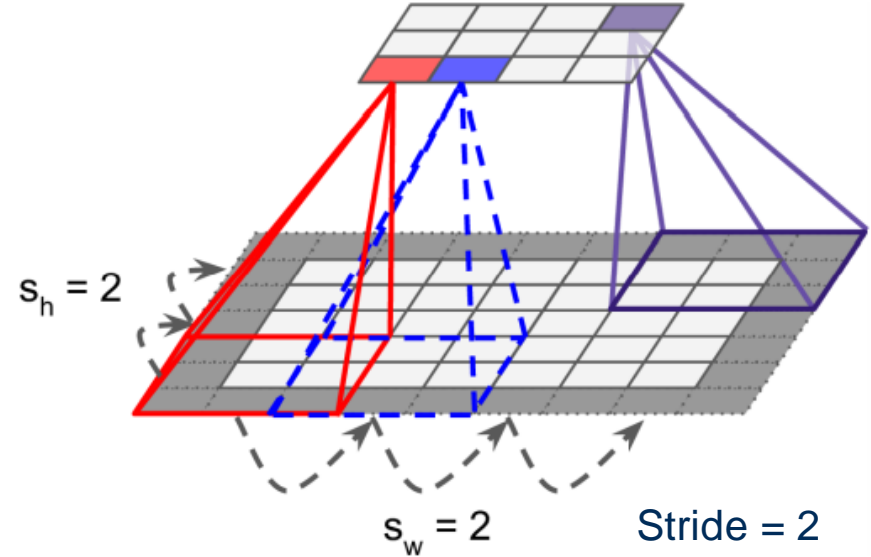
Convolution



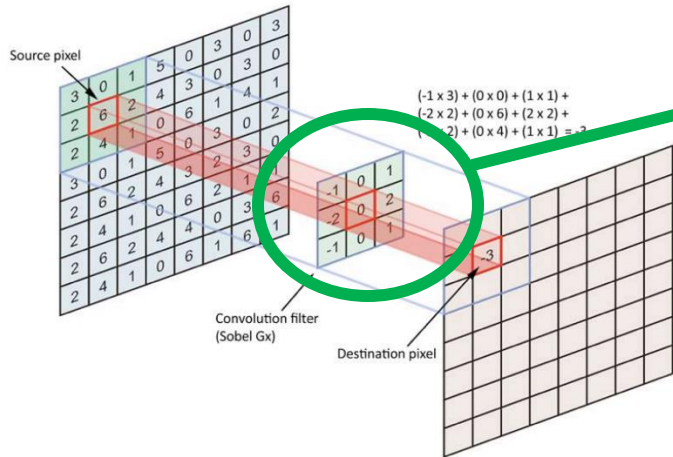
Convolution



We can reduce the dimension of the original image

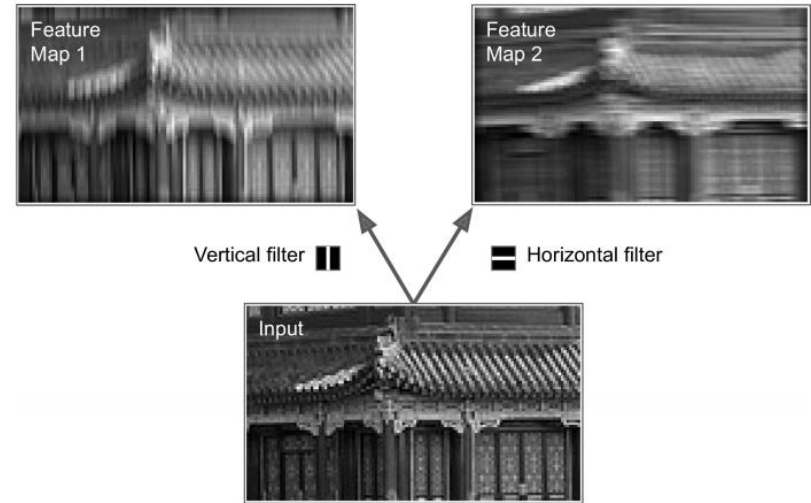


Filters/Convolution Kernels

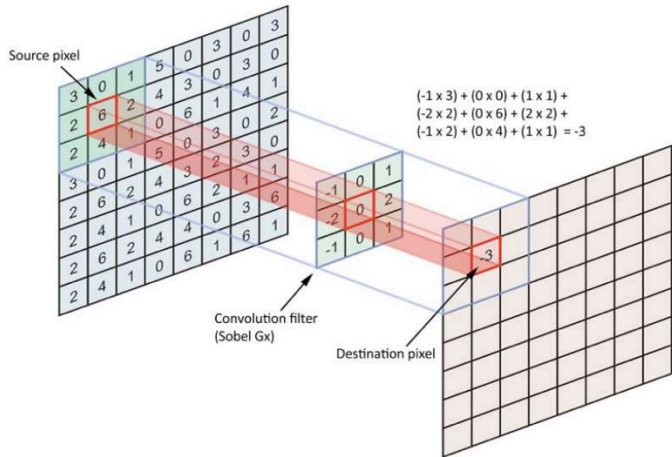


Filter
(Need to be learned)

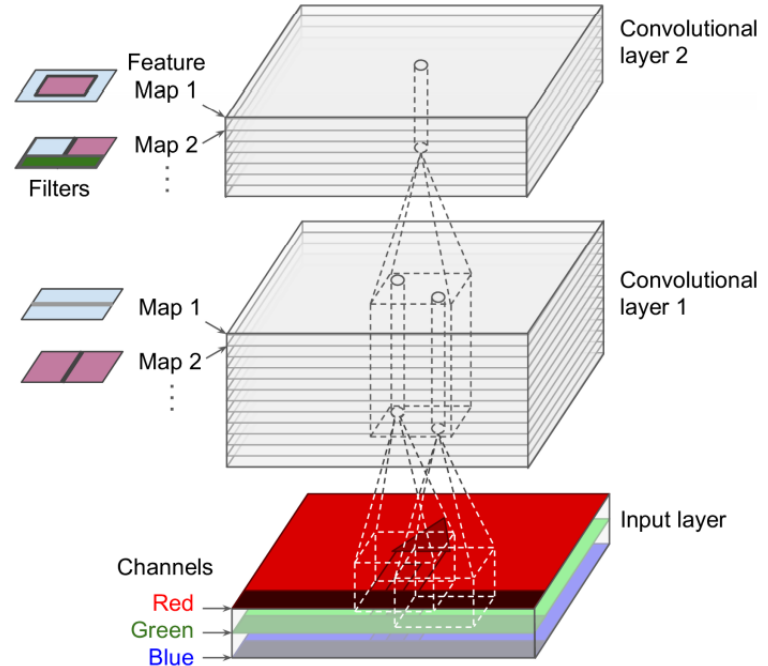
Single filters produce specific *feature maps*



Filters/Convolution Kernels



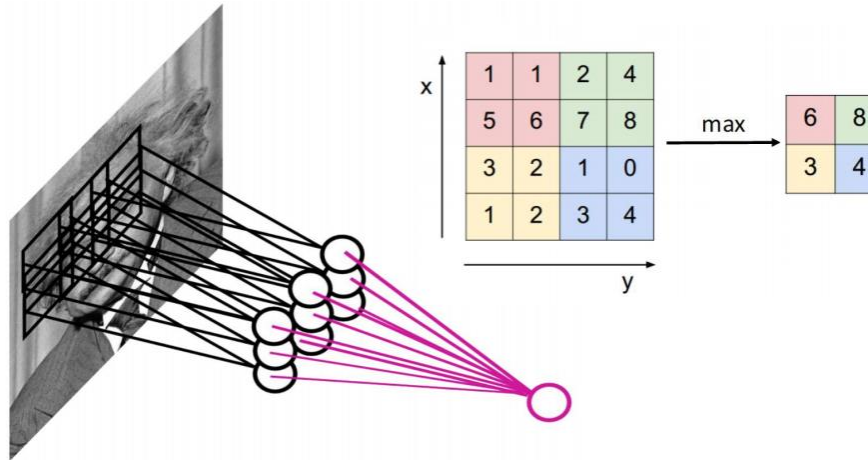
Every convolutional layer has many filters



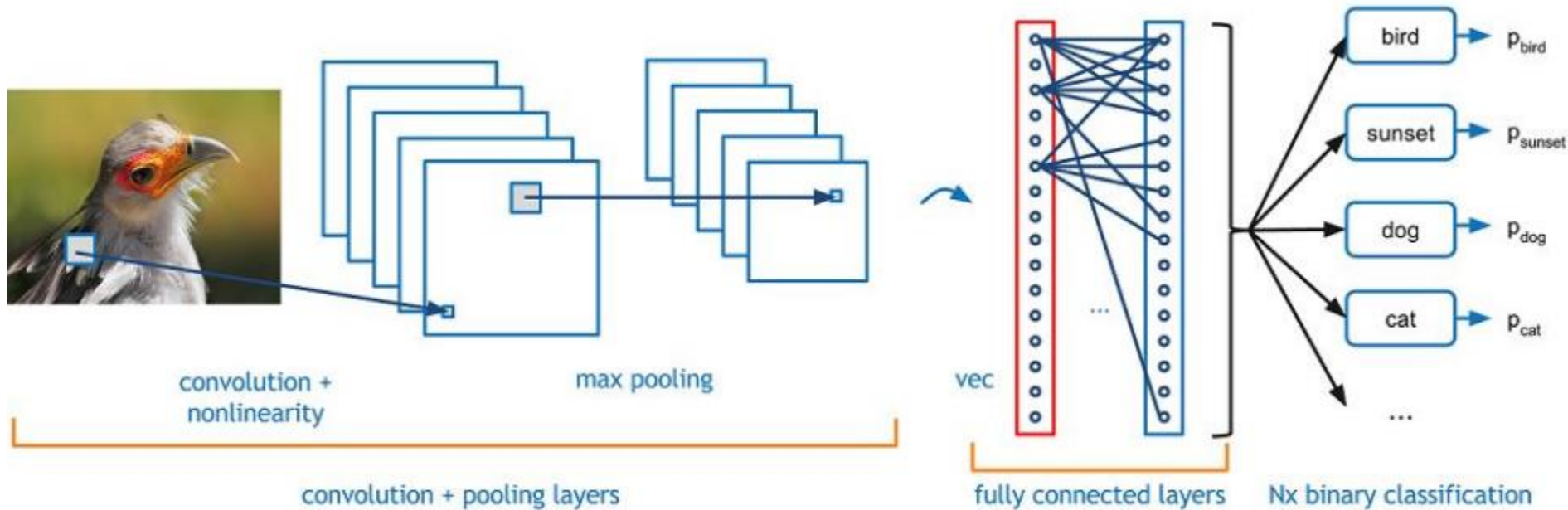
Pooling Layers

Goal: To subsample (i.e., shrink) the input image in order to reduce the computational load, the memory usage, and the number of parameters

However, a pooling neuron has no weights; all it does is aggregate the inputs using an aggregation function such as the **max** or **mean**.



Typical Convolution Neural Network (CNN)



Feature extraction

Classification

TF/Keras Implementation



So far, we flattened the images before feeding them as input to a NN. Now, CNN take a tensor of shape = [image_h, image_w, color_channels]

```
model2 = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(32, 32, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation = 'softmax')
])
```

After you can attach a “classic” MLP/NN, with Flatten and Dense layers

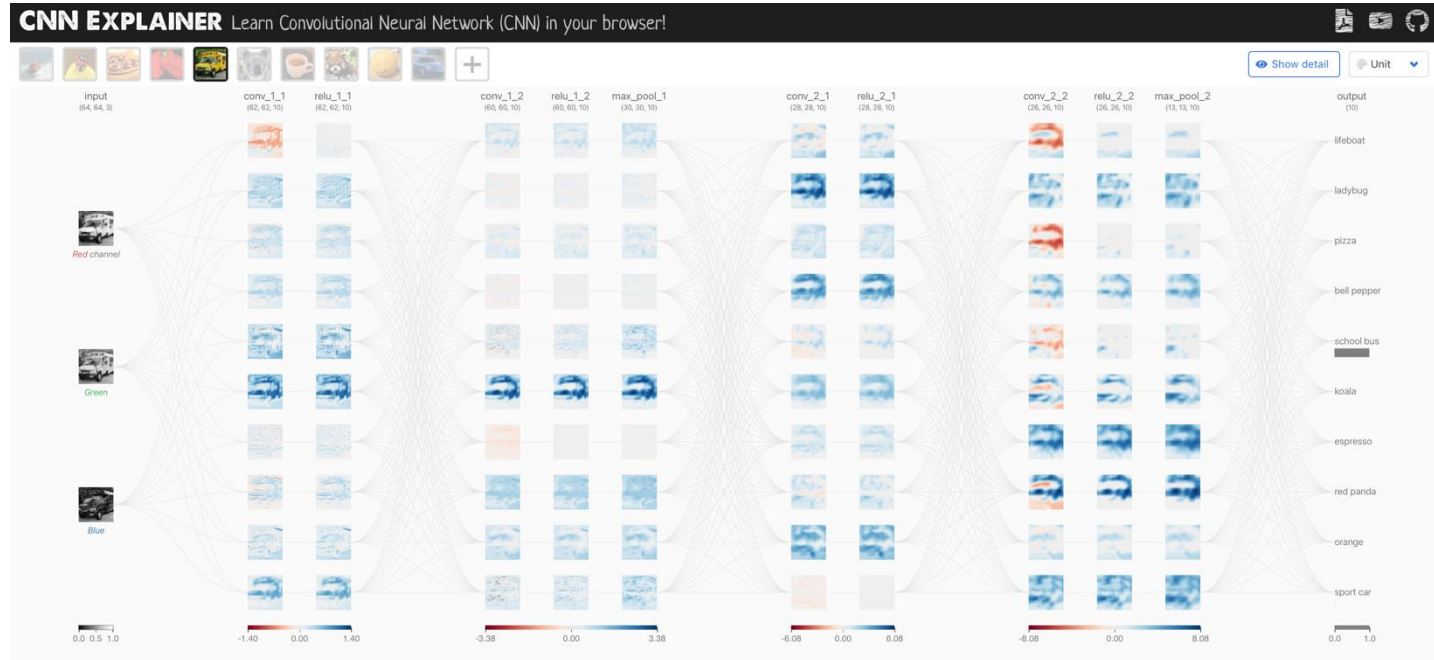
Example in action



Stanford University CS231n: Convolutional Neural Networks for Visual Recognition

Example in action -2-

Try yourself: <https://poloclub.github.io/cnn-explainer/>



Example in action -3-


<https://setosa.io/ev/image-kernels/>



input image

$$\begin{pmatrix} \begin{matrix} ? & + & 243 & + & 160 \\ \times 0 & \times -1 & \times 0 \end{matrix} \\ + \\ \begin{matrix} ? & + & 192 & + & 154 \\ \times -1 & \times 5 & \times -1 \end{matrix} \\ + \\ \begin{matrix} ? & + & 90 & + & 109 \\ \times 0 & \times -1 & \times 0 \end{matrix} \end{pmatrix} = \begin{matrix} ? \end{matrix}$$

kernel:





output image

Useful Reads:

A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, Tensorflow*, O'Reilly
[Chapter 11 and part of 14]

Thank you.

