BSC~Microsoft Research Centre

Deep Learning for Computer Vision

Tassadaq Hussain

Professor Namal University

Director Centre for AI and Big Data

Collaborations:

Barcelona Supercomputing Center Barcelona, Spain

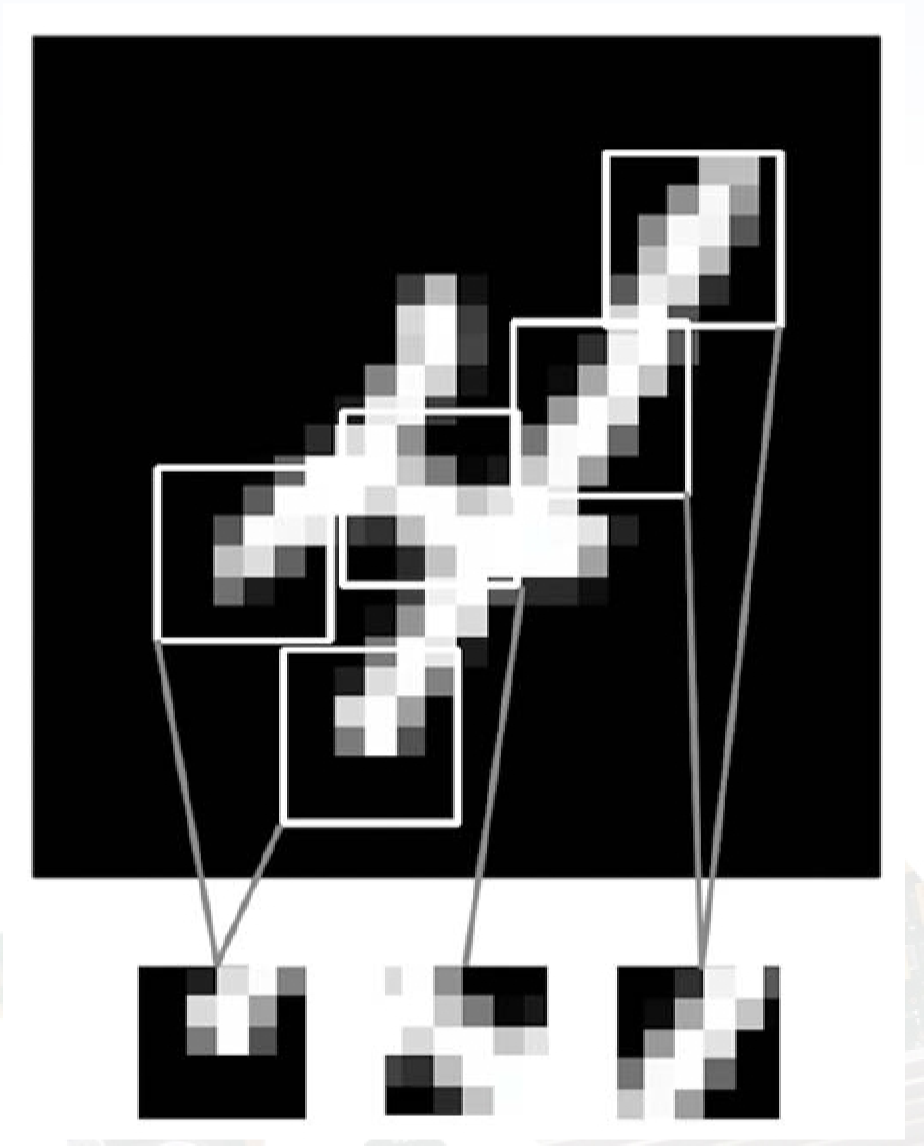
European Network on High Performance and Embedded Architecture and Compilation

Pakistan Supercomputing Center

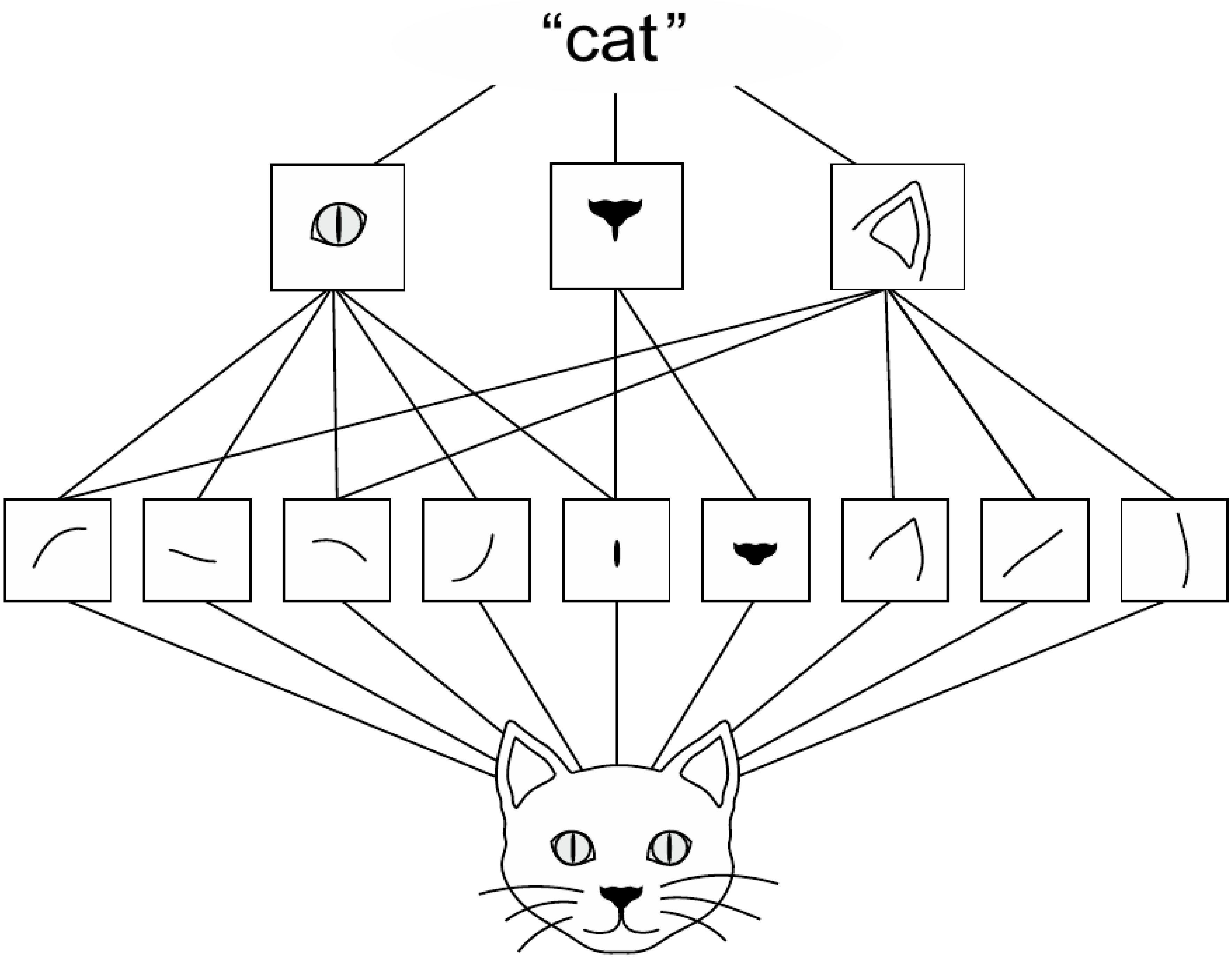
UCERD
Gathering Intellectuals
www.ucerd.com

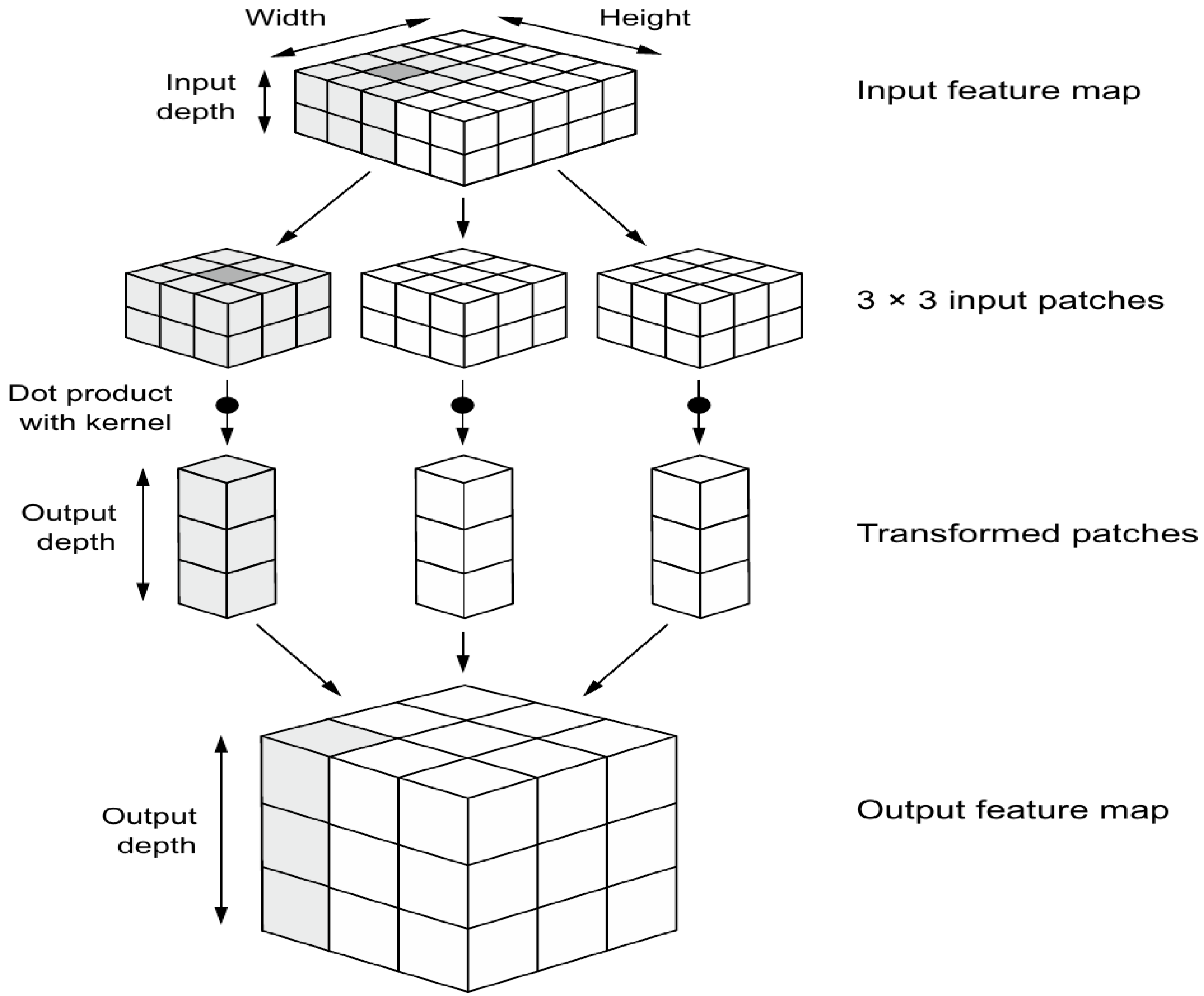
```
from tensorflow import keras
from tensorflow.keras import layers
inputs = keras.Input(shape=(28, 28, 1))
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(inputs)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(10, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

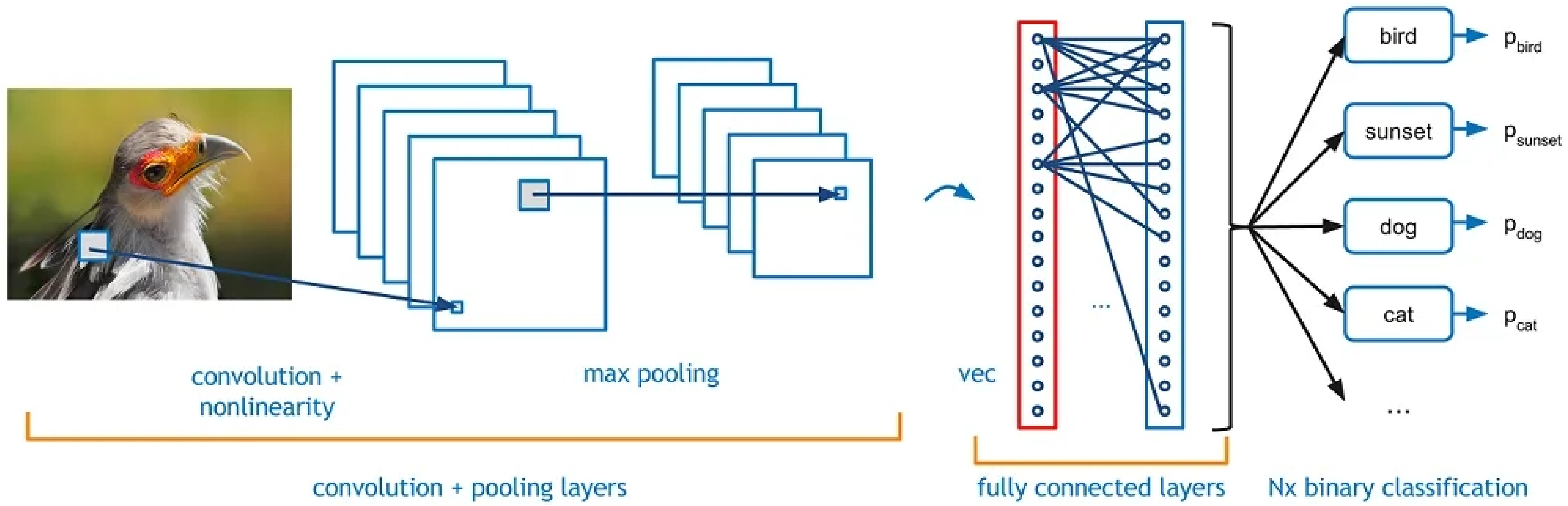
Convolution operation



UCERD


“cat”


Width
Height
Input depth
Input feature map
3 × 3 input patches
Dot product with kernel
Output depth
Transformed patches
Output depth
Output feature map



convolution +
nonlinearity
max pooling
vec
...
bird
sunset
dog
cat
...
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
convolution + pooling layers
fully connected layers
Nx binary classification

UCERD

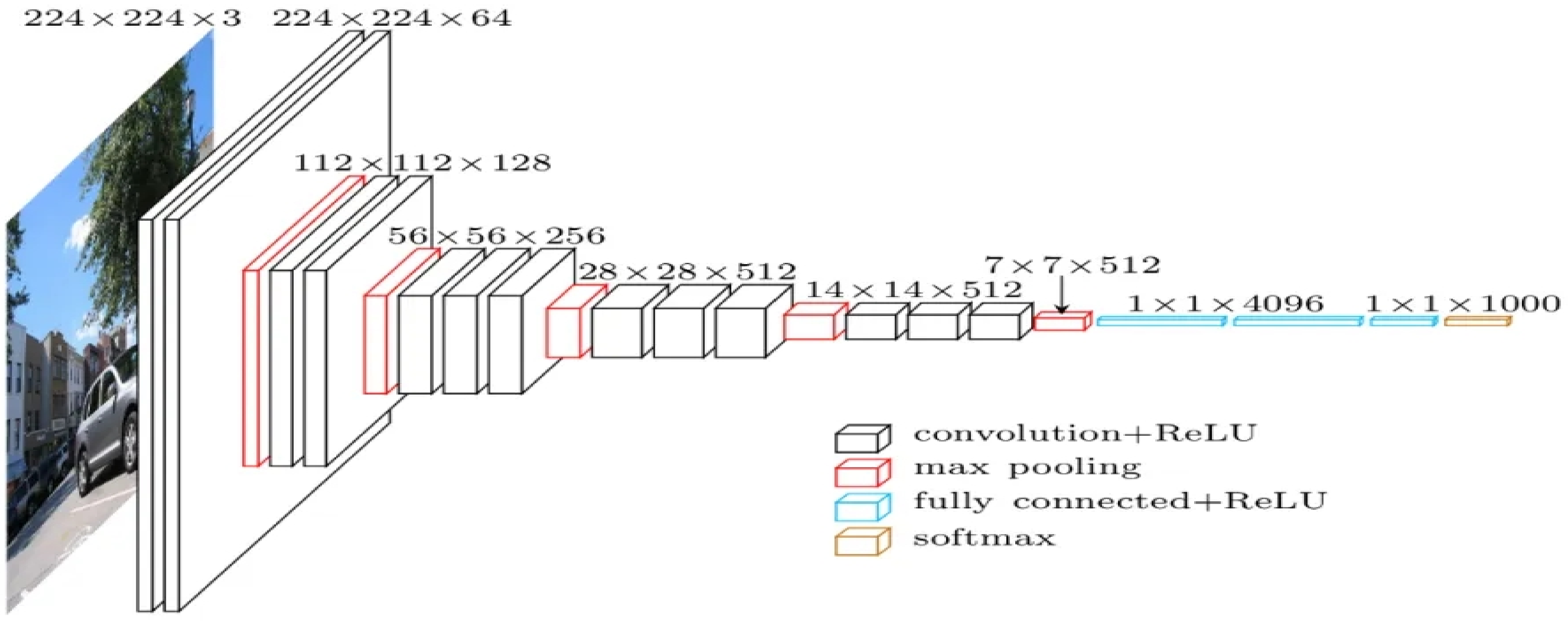
CNN?

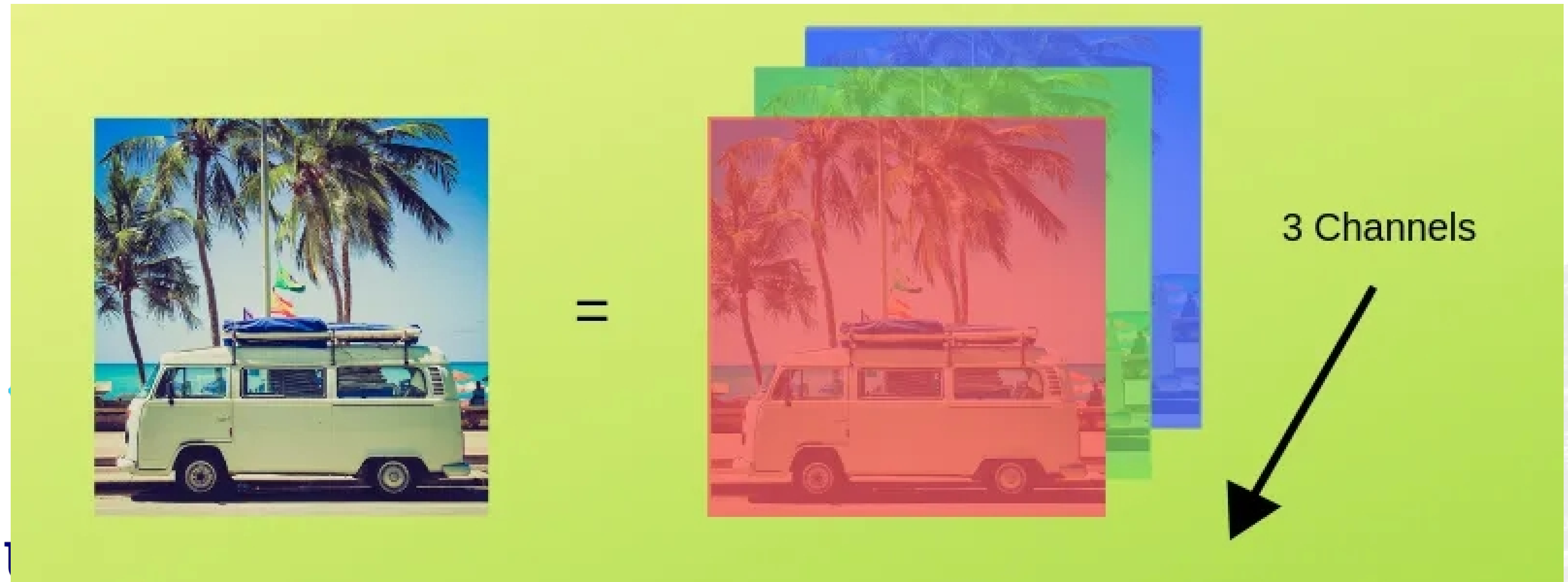
- Computer vision is evolving rapidly day-by-day. Its one of the reason is deep learning. When we talk about computer vision, a term convolutional neural network(abbreviated as CNN) comes in our mind because CNN is heavily used here. Examples of CNN in computer vision are face recognition, image classification etc. It is similar to the basic neural network. CNN also have learnable parameter like neural network i.e, weights, biases etc.

UCERD

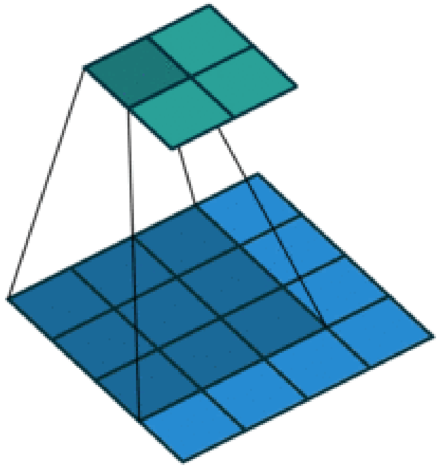
Why CNN

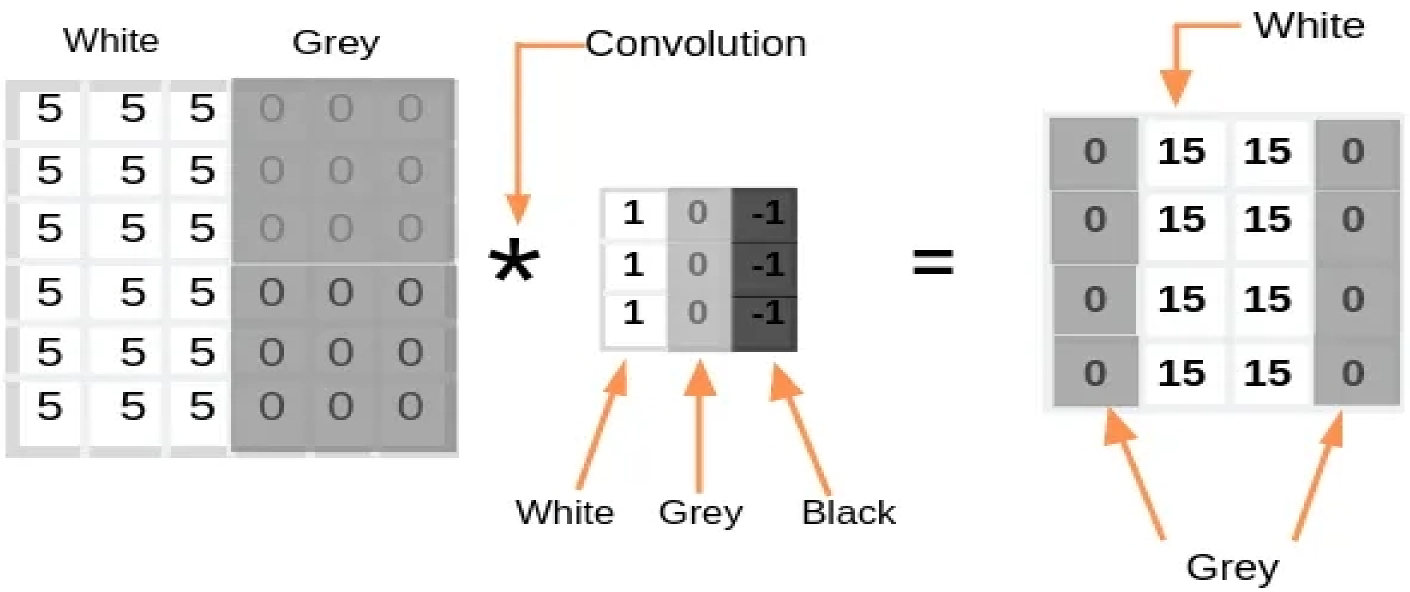
- Suppose image is 1000 x 1000 which means you need 10^6 neurons in input layer. **It is computationally ineffective right**.
- **CNN extract the feature of image and convert it into lower dimension without loosing its characteristics.**




=
3 Channels

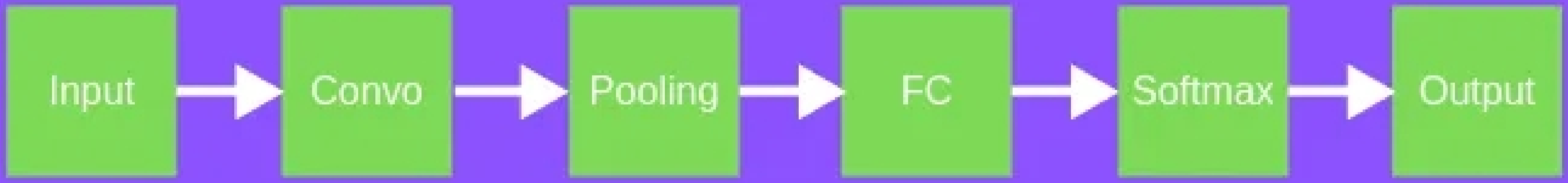
Edge Detection





$$\begin{bmatrix}5&5&5&0&0&0\\5&5&5&0&0&0\\5&5&5&0&0&0\\5&5&5&0&0&0\\5&5&5&0&0&0\\5&5&5&0&0&0\end{bmatrix} * \begin{bmatrix}1&0&-1\\1&0&-1\\1&0&-1\end{bmatrix} = \begin{bmatrix}0&15&15&0\\0&15&15&0\\0&15&15&0\\0&15&15&0\end{bmatrix}$$

Layers in CNN



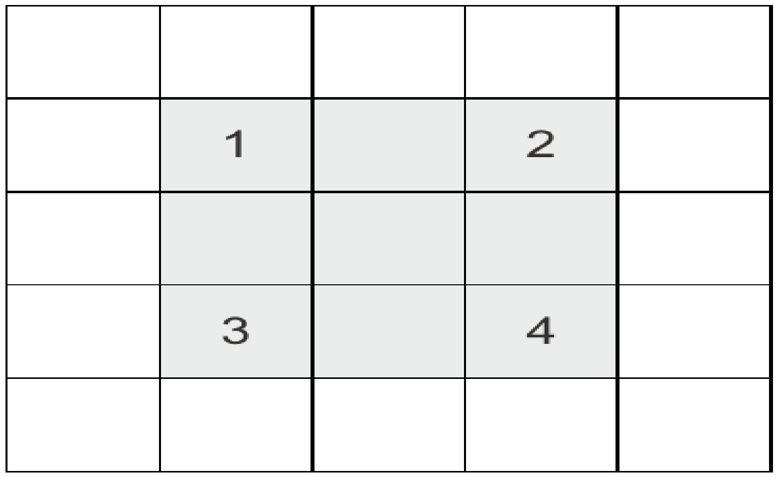
The convolution operation

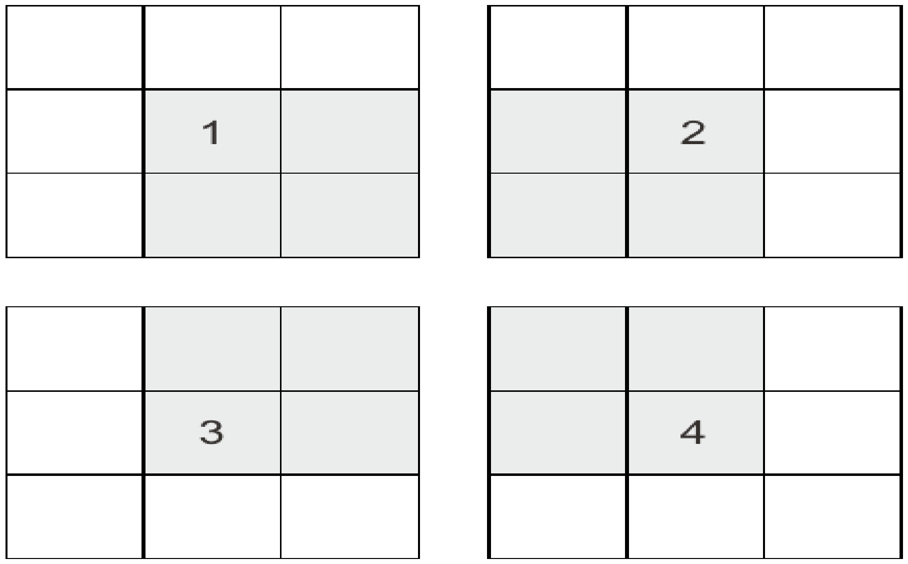
- The fundamental difference between a densely connected layer and a convolution layer is this:
- Dense layers learn global patterns in their input feature space
- Convolution layers learn local patterns in the case of images, patterns found in small 2D windows of the inputs.

UCERD

CONVOLUTION STRIDES

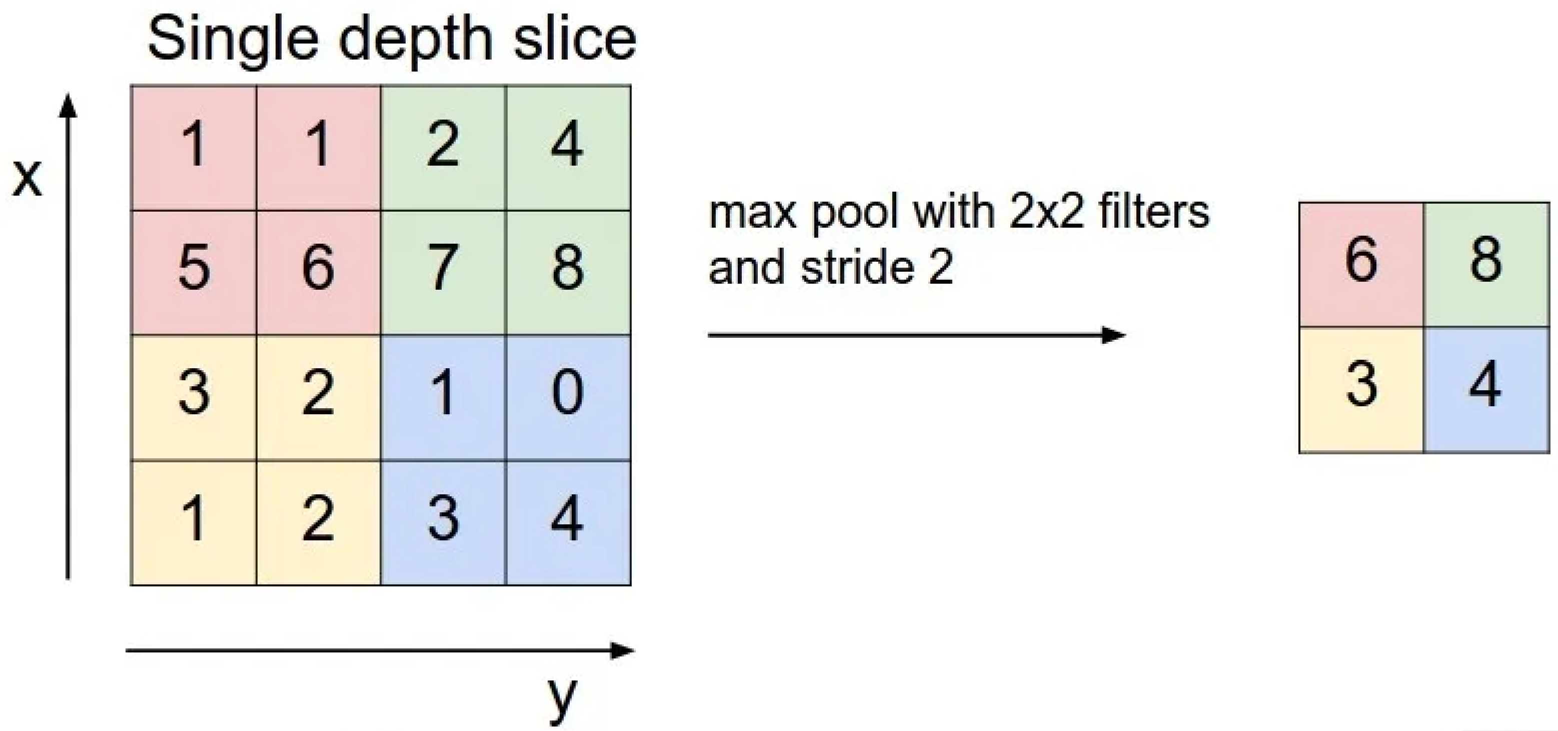
- The other factor that can influence output size is the notion of strides.
- It has been assumed that the center tiles of the convolution windows are all contiguous. But the distance between two successive windows is a parameter of the convolution, called its stride, which defaults to 1.

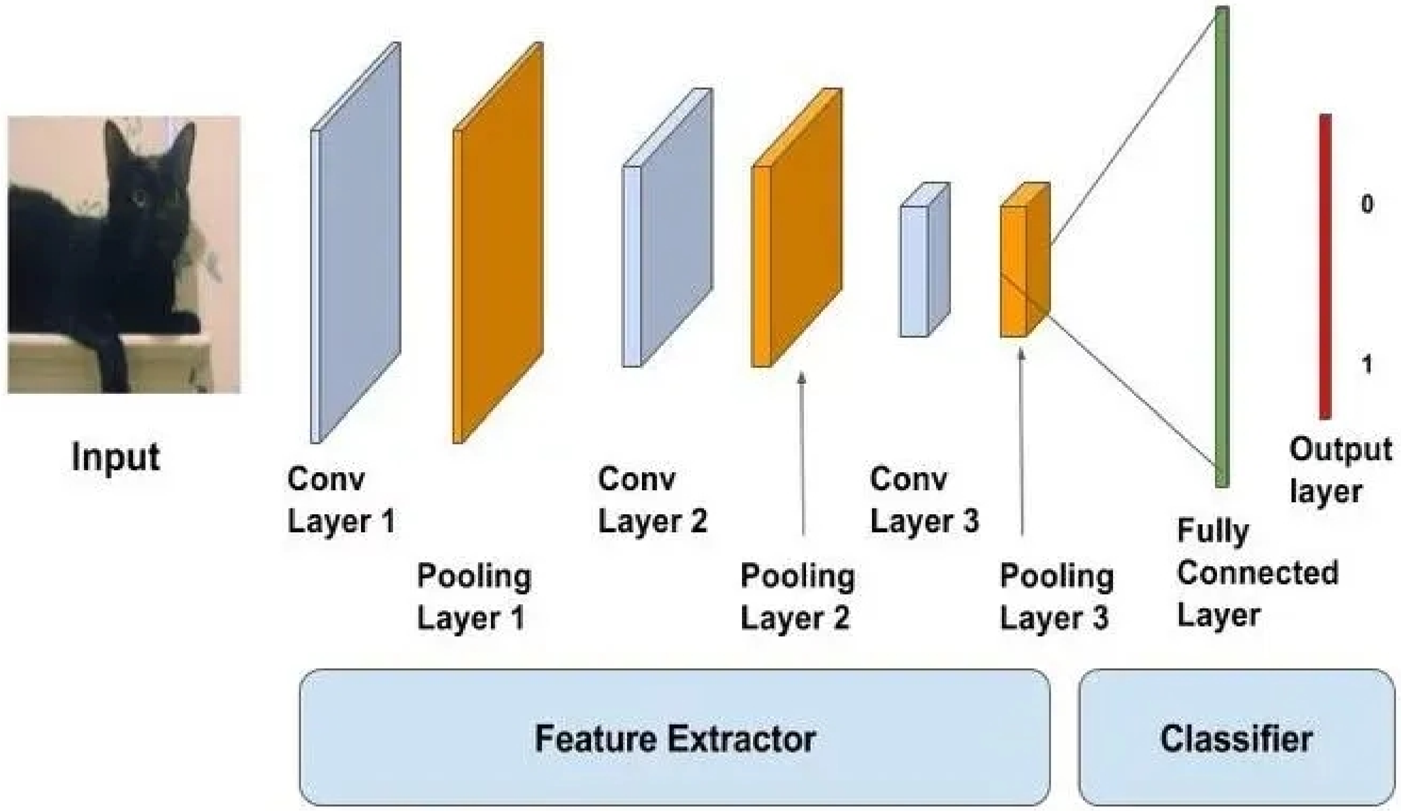




Activation Layer (ReLU):

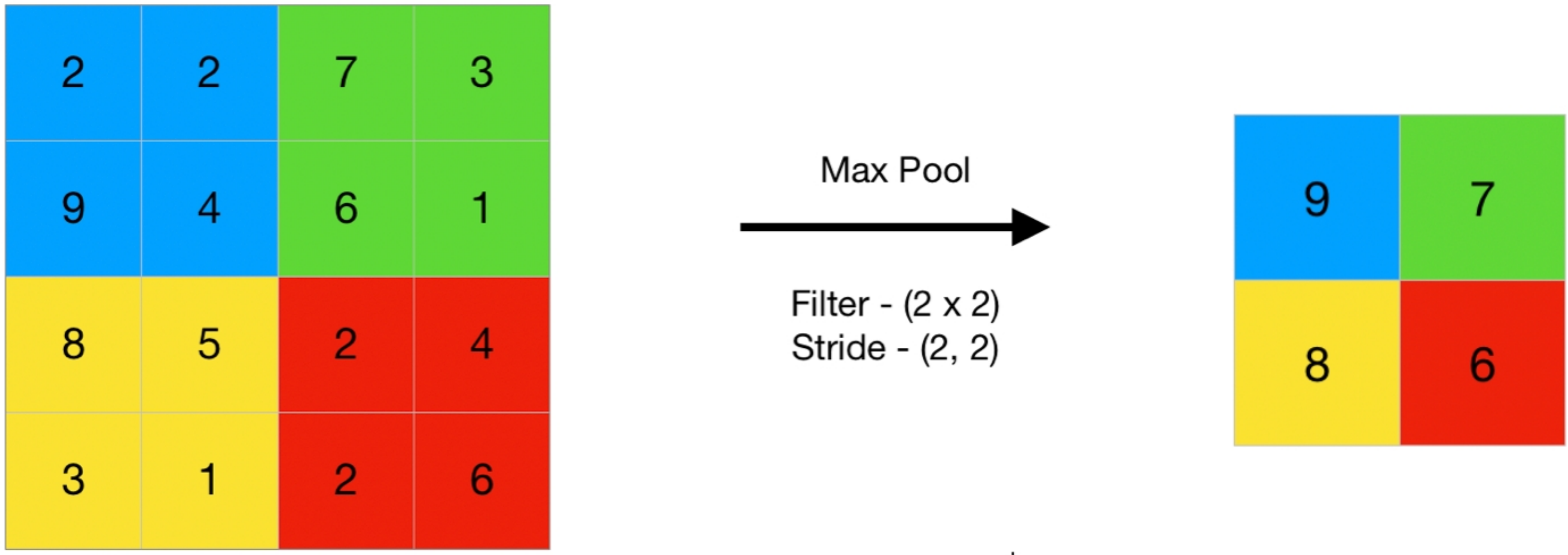
After each convolutional operation, a non-linear activation function, often ReLU (Rectified Linear Unit), is applied element-wise to introduce non-linearity into the model. ReLU helps in capturing complex patterns and relationships within the data.


Single depth slice
x
y
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


Input
Conv Layer 1
Pooling Layer 1
Conv Layer 2
Pooling Layer 2
Conv Layer 3
Pooling Layer 3
Fully Connected Layer
Output layer
0
1
Feature Extractor
Classifier

The max-pooling operation

- Pooling layers (e.g., MaxPooling or AveragePooling) are used to downsample the spatial dimensions of the feature maps generated by the convolutional layers. This reduces the computational load and helps in creating a more abstract representation of the input data.



Fully Connected Layer (Dense Layer):

Fully connected layers connect every neuron in one layer to every neuron in the next layer. They are typically placed towards the end of the CNN and are responsible for making predictions based on the high-level features extracted by the earlier layers.

Flattening Layer:

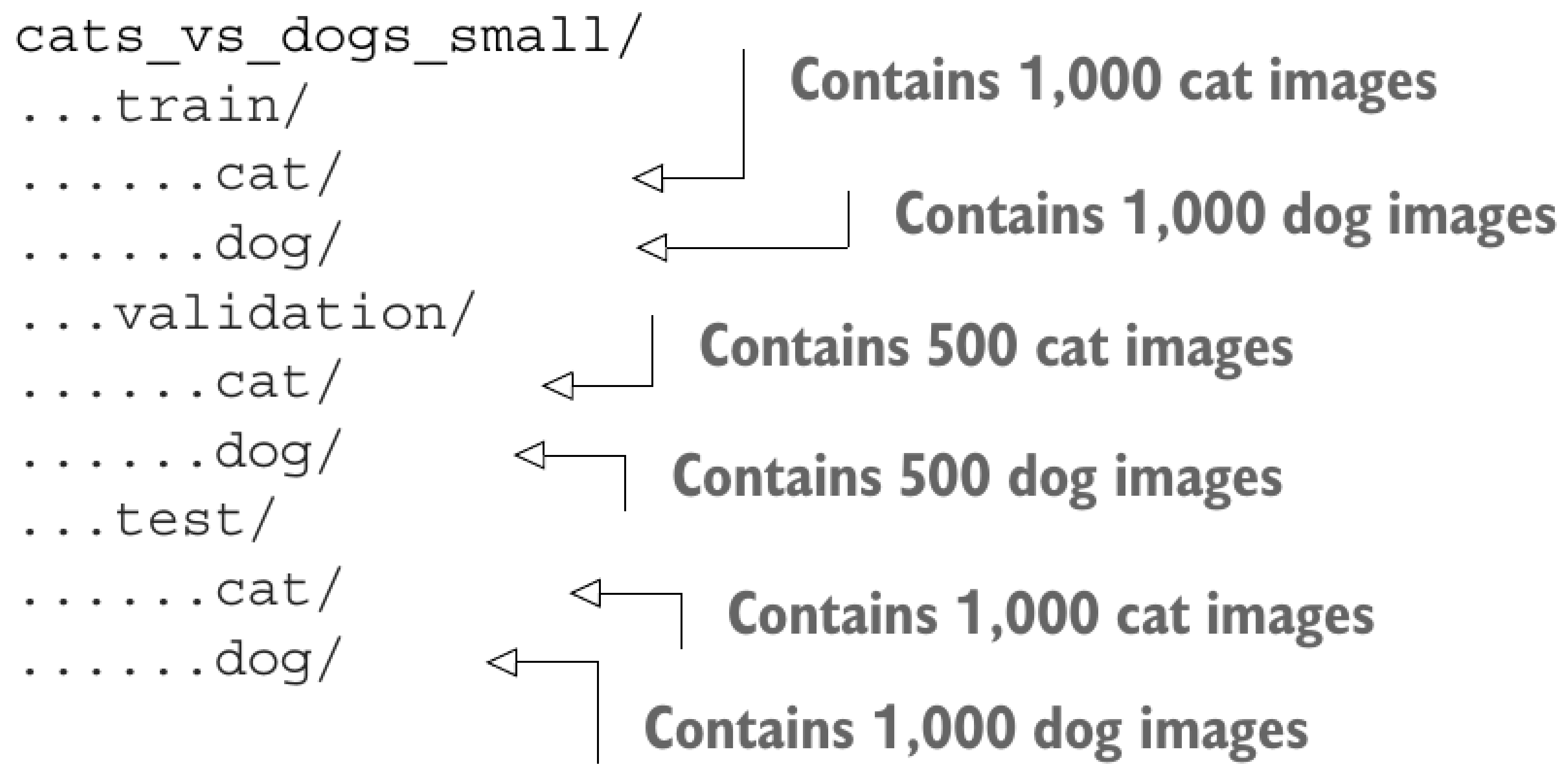
Before passing the output of the convolutional and pooling layers to the fully connected layers, a flattening layer is often used to convert the 2D matrix data into a vector. This step prepares the data for input into the fully connected layers.

UCERD

Output Layer:

The output layer produces the final predictions or classifications. The number of neurons in this layer corresponds to the number of classes in a classification task, and the activation function is often softmax for multi-class classification.

```
from tensorflow import keras
from tensorflow.keras import layers
inputs = keras.Input(shape=(180, 180, 3))
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.Flatten()(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```



```
cats_vs_dogs_small/
...train/
......cat/
......dog/
...validation/
......cat/
......dog/
...test/
......cat/
......dog/
```

Contains 1,000 cat images
Contains 1,000 dog images
Contains 500 cat images
Contains 500 dog images
Contains 1,000 cat images
Contains 1,000 dog images

```
import os, shutil, pathlib
original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small")
def make_subset(subset_name, start_index, end_index):
for category in ("cat", "dog"):
dir = new_base_dir / subset_name / category
os.makedirs(dir)
fnames = [f"{category}.{i}.jpg"
for i in range(start_index, end_index)]
for fname in fnames:
shutil.copyfile(src=original_dir / fname,
dst=dir / fname)
make_subset("train", start_index=0, end_index=1000)
make_subset("validation", start_index=1000, end_index=1500)
make_subset("test", start_index=1500, end_index=2500)
```