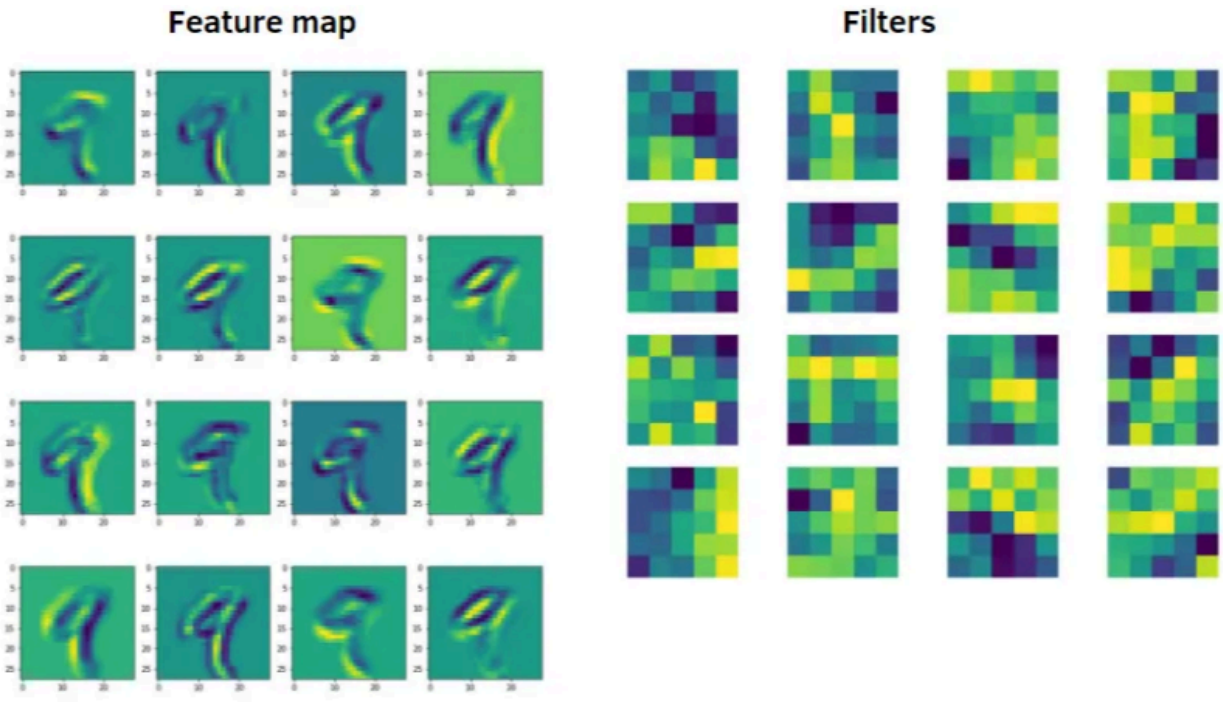
Pooling Layer in CNN

`from tensorflow.keras.layers import MaxPooling2D, AveragePooling2D`

`model.add(MaxPooling2D(pool_size=(2, 2), strides=2, padding='valid'))`

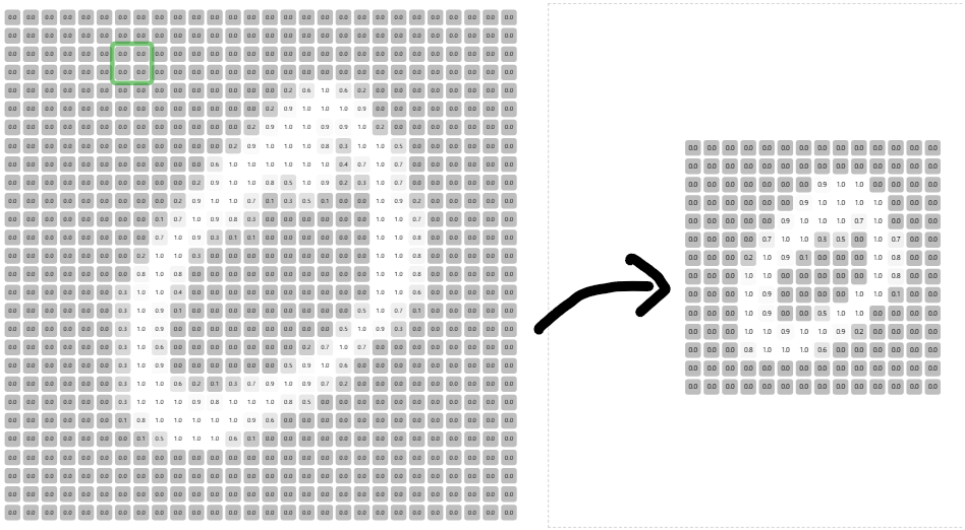
- **Reduce spatial dimensions** (width & height) while retaining important features
- **Reduce the size of the feature maps**.



This helps:

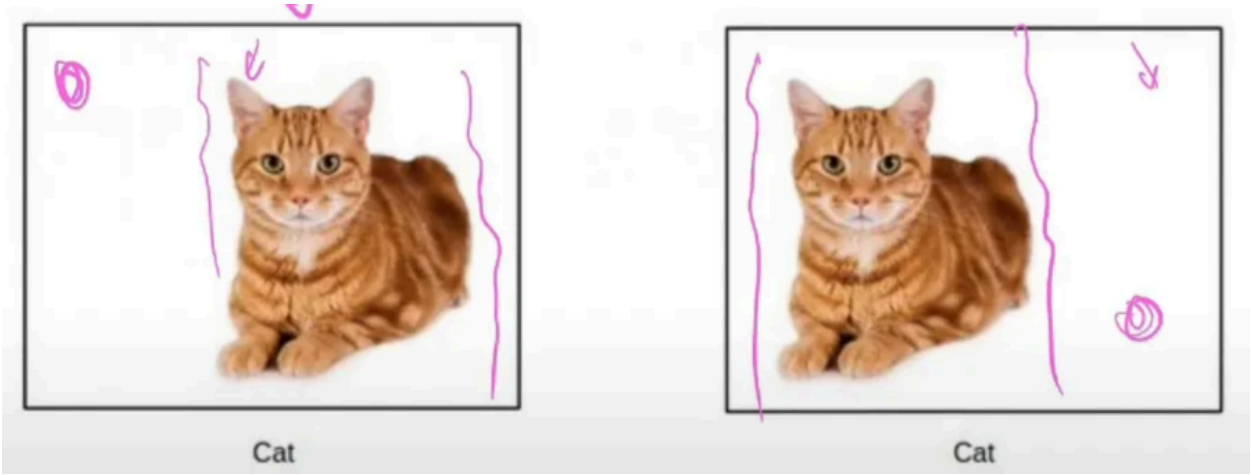
- 🧠 **Retain the most important information**
- **Reduce low level details.**
- 📉 **Reduce computation**
- 🔁 **Prevent overfitting** (too much memorization)
- **Reduce translation variance**

- **Enhance features**
- No need for training



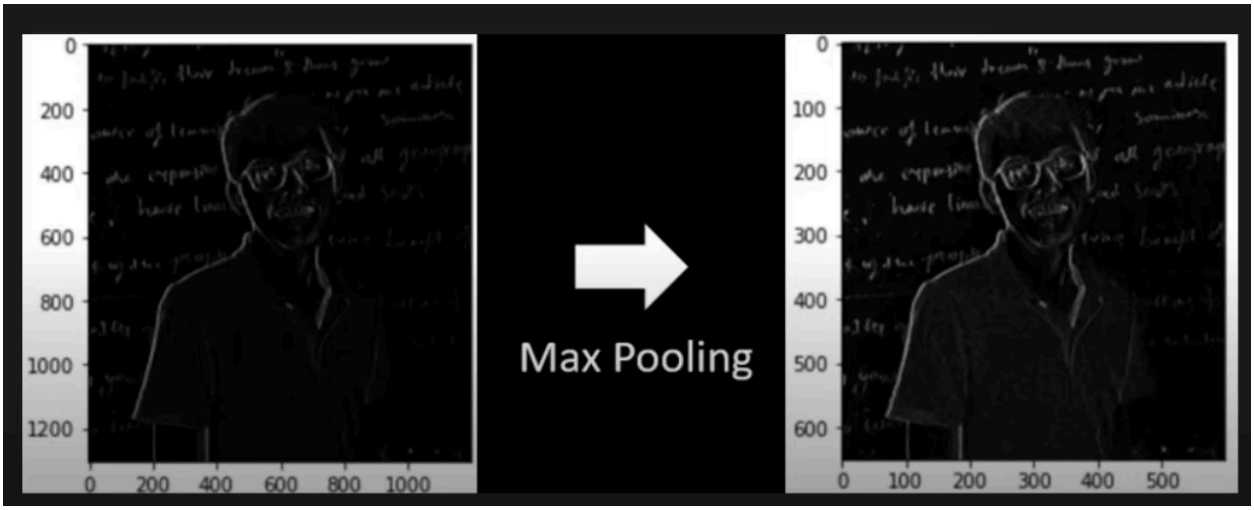
Translation variance:

- Translation variance refers to the phenomenon where small translations (shifts) in the position of an object or feature in an image can cause large changes in the output of a mode



- **Translational invariance** refers to **the property where an operator remains the same regardless of the shift in position**, ensuring that the representation of a visual object remains consistent irrespective of its location in the image plane.

Enhanced Features: (Only works with *MaxPooling*)



The two most common types:

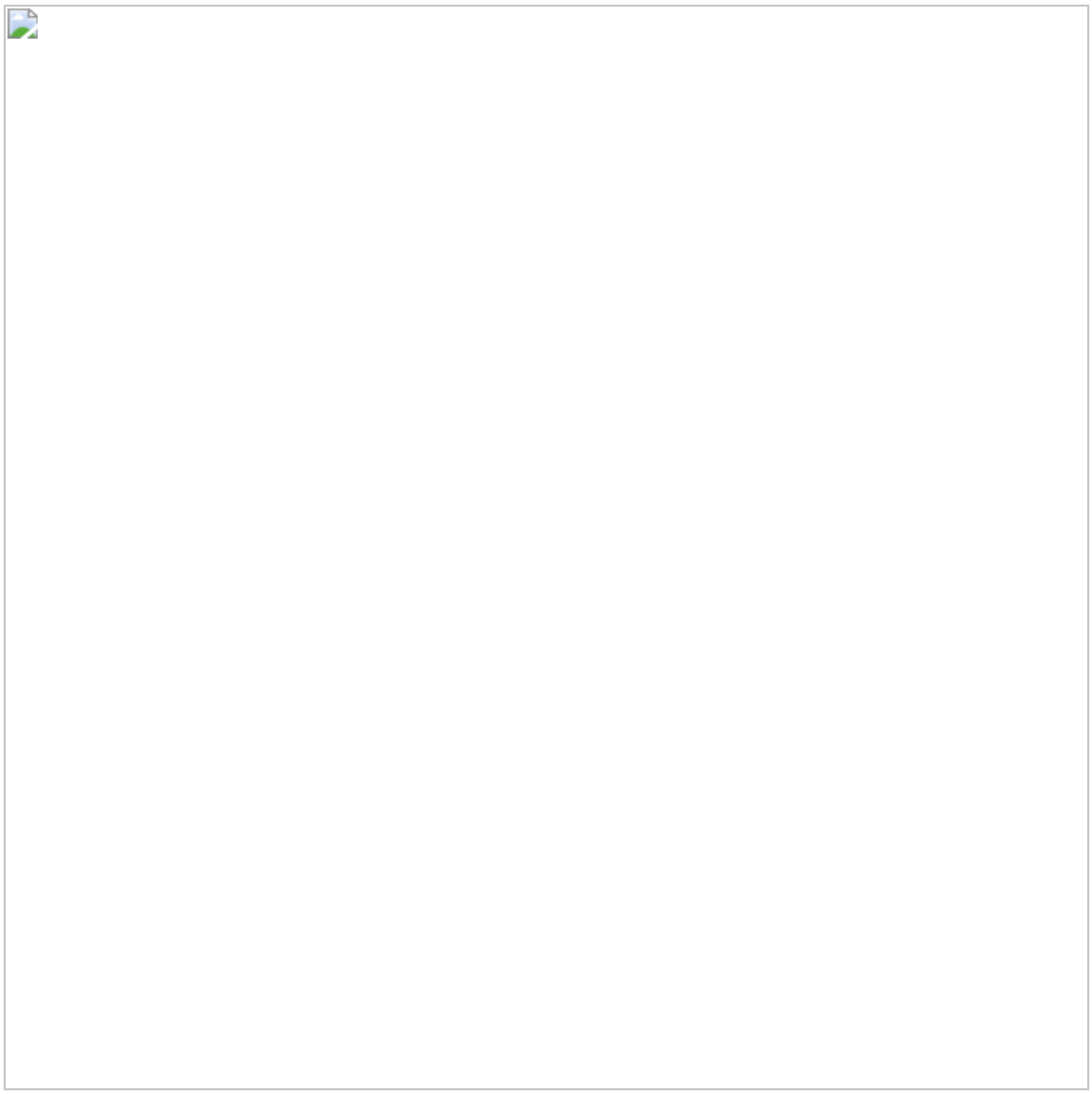
- **Max Pooling** (keeps the largest number)
- **Average Pooling** (keeps the average)

1. Max Pooling (`MaxPooling2D`)

- **Operation**: Takes the **maximum value** from each window.
- **Use Case**: **Most common**; preserves **sharpest features** (e.g., edges).

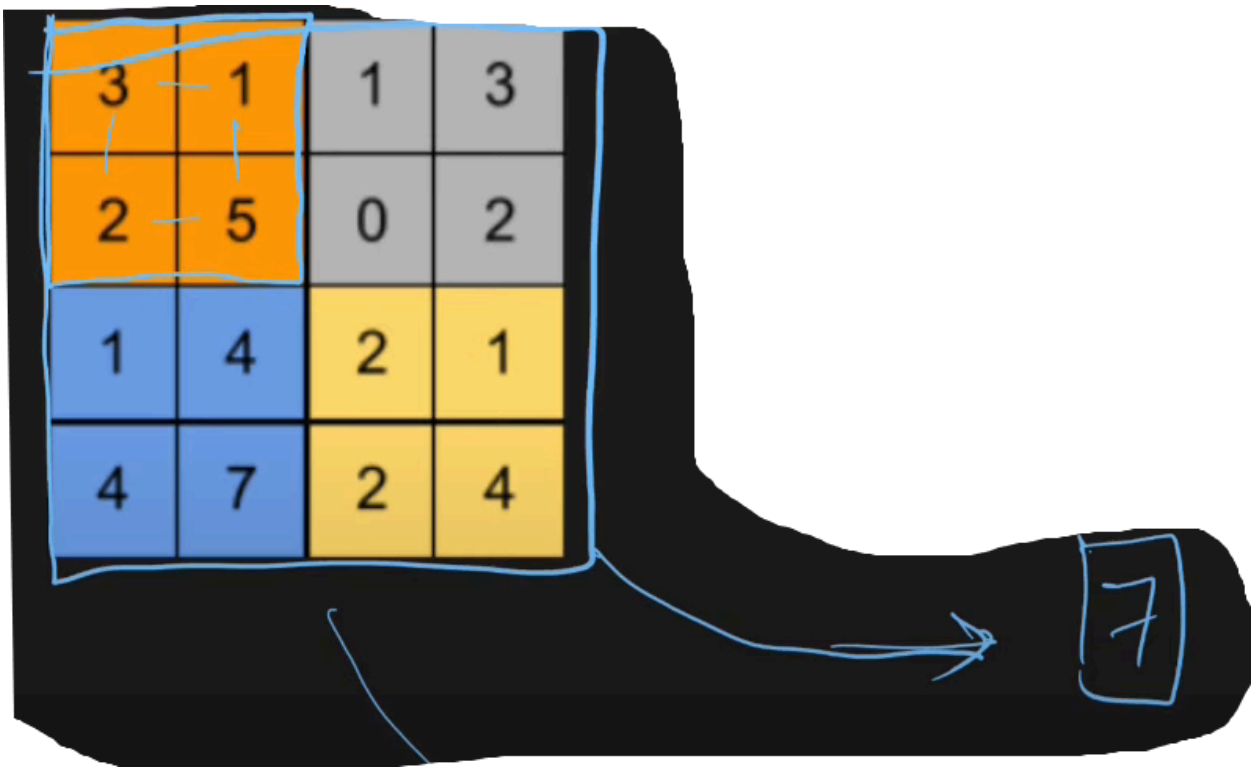
2. Average Pooling (`AveragePooling2D`)

- **Operation**: Takes the **average value** from each window.
- **Use Case**: Smooths features (e.g., in older architectures like LeNet).
- **Keras Example**:



3. Global Pooling (`GlobalMaxPooling2D`, `GlobalAveragePooling2D`)

- **Operation**: Reduces **each channel** to a **single value** (max or avg).
- **Use Case**: Replaces `Flatten()` in classification heads (e.g., ResNet).



Why Do We Use Pooling?

Imagine the CNN extracts hundreds of features (edges, textures, shapes) from an image. Without pooling:

- The data becomes **too big**.
- It will take **too much time to compute**.
- The network may focus on **too many unimportant details**.

Pooling solves this by:

- Compressing the image
- Keeping only the **strongest features**
- Making the model **faster, smaller, and more robust**

What Does Pooling Do?

Pooling uses a small window (like 2×2 or 3×3) and slides it over the input, just like a filter. Instead of multiplying, it **summarizes** the values inside the window using a

rule:

Pooling Type	Rule Applied	Example
Max Pooling	Keeps the **maximum** value	[2, 8, 1, 4] → 8
Average Pooling	Takes the **average** of values	[2, 8, 1, 4] → (2+8+1+4)/4 = 3.75

📏 Parameters in Pooling Layers

Parameter	What It Does	Default / Common Values
`pool_size`	Size of the window (like 2×2, 3×3)	2×2
`stride`	Steps the window moves (usually same as pool size)	2
`padding`	If needed to preserve size	Usually not used
`type`	max or average pooling	max

Keras Code: Full CNN with Pooling

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential([
    # Conv → ReLU → Pool
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),  # Output: (14,14,32)

    # Repeat
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),  # Output: (7,7,64)

    # Classifier
    Flatten(),
    Dense(10, activation='softmax')
```

```
])

model.summary()
```

Default values:

```
pool_size=(2, 2), strides=None, padding="valid"
```

`strides:`

- If `None`, the stride is set to the same value as `pool_size`. In this case, the stride will be `(2, 2)`.
- If you want to control the stride separately, you can provide a tuple `(stride_height, stride_width)`.

`padding` (Default: `'valid'`):

- Determines the padding method for the pooling operation:
  - `'valid'`: No padding. The pooling window only slides over valid regions, reducing the output size.
  - `'same'`: Padding is added so that the output has the same spatial dimensions (height and width) as the input

Disadvantage of Pooling

- **Loss of Spatial Information**
- Not helpful in image segmentation
  - **Aggressive downsampling** may discard useful details (e.g., precise object boundaries).
  - Harms tasks requiring **pixel-level accuracy** (e.g., segmentation).

When to Avoid Pooling?

1. **Low-Resolution Images** (e.g., 32×32):

- Pooling may destroy critical details.

2. **Pixel-Level Tasks** (e.g., Segmentation):
   - Use **transposed convolutions** or **dilated convolutions** instead.
3. **GANs/Image Generation**:
   - Pooling causes artifacts; prefer **strided convs** or **pixel shuffle**.

📚 Summary Table

Feature	Max Pooling	Average Pooling	Global Pooling
Output Size	Smaller	Smaller	1 value per channel
Learnable?	❌ No	❌ No	❌ No
Keeps Location?	❌ No	❌ No	❌ No
Best For	Classification	General smoothing	Final CNN layers