Computer Vision

Synthesis Questions

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are based on the brain's visual cortex.

Convolutional layer takes a 3d tensor (channels, height, width) as an input, and determines the weight of each pixel by going through the image section by section. This is done by applying a "filter" called ***kernel*** to each section, which is a matrix containing scalars. When determining the section of the image, ***stride*** refers to the the shift in pixels to the next *kernel*, and ***padding*** refers to the extra pixels around the **input** borders. After going through all the sections, the convolutional layer returns an array output called ***feature map***, representing the feature of the original image.

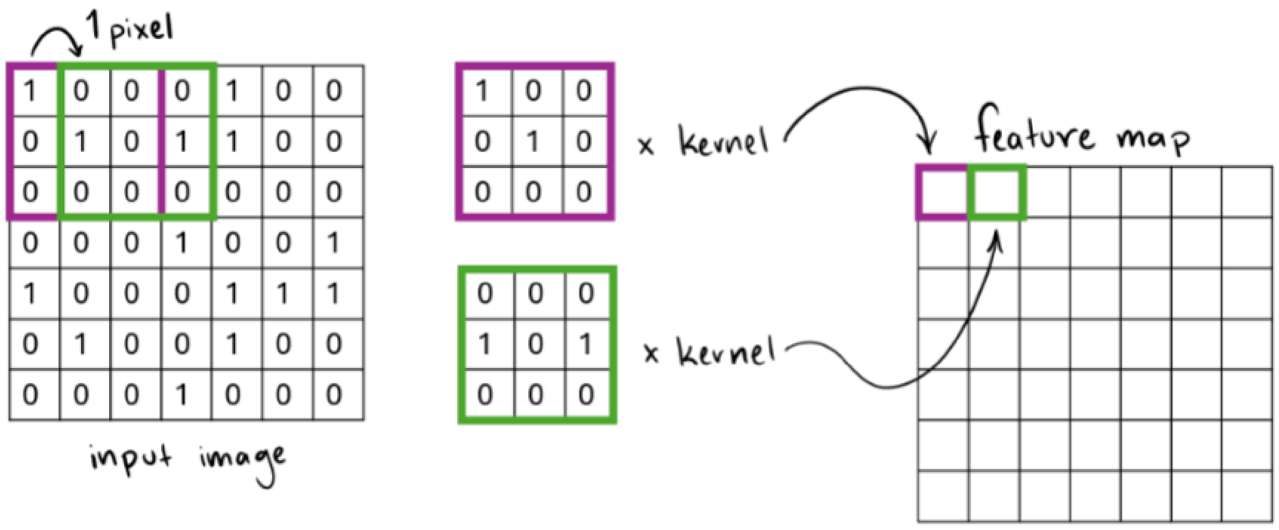

Figure 3: Illustration depicting a stride of 1 pixel

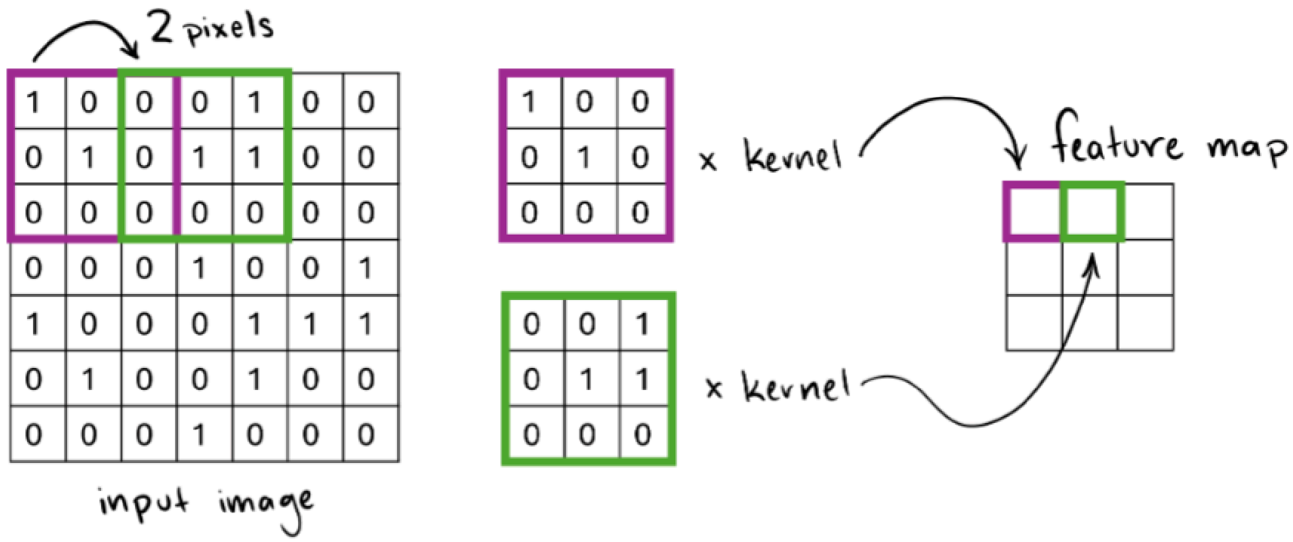

Figure 4: Illustration depicting a stride of 2 pixels

image-131.png

- Here, the number of the stride and padding is responsible for the size of the output feature map. The values populating a kernel determines the value going in the output feature map.

After the convolutional layer outputs the feature map, an activation function is applied for more complex feature representation. Early layers learn simple features like edges and textures, while deeper layers combine these to detect more abstract features such as objects or shapes. Usually, a **ReLU** is used.

Then, a pooling layer comes in, which reduces the amount of parameters that move on to the next layer and the amount of computation necessary, making the model more efficient. The most common technique is called **max pooling**.

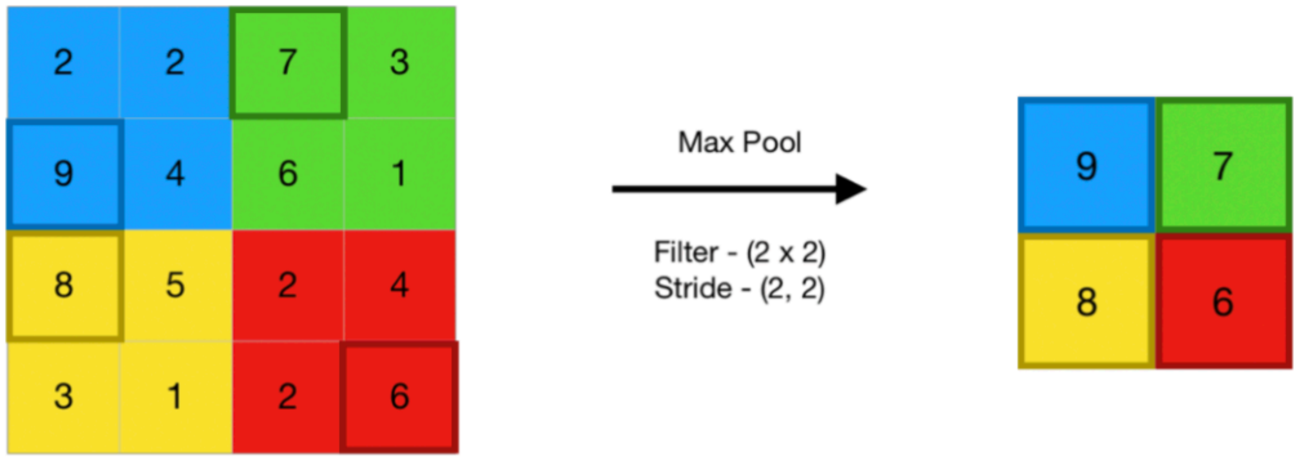

Figure 5: Illustration of the max pooling operation

image-132.png

There are usually multiple convolutional and pooling layers in CNNs. This is because depth builds a feature hierarchy:

- Early convolutional layers: edges/gradients/textures
- Middle convolutional layers: corners/motifs/parts
- Deeper convolutional layers: object-level patterns

At the end, after a multiple convolutional and pooling layers, the final output is then flattened into a one-dimensional vector. It is then put into a dense layer, which performs either a regression or classification operations. When doing a multi-level classification, it is common to use a softmax activation function.

Self-Supervised Learning

1. How does self-supervised learning mitigate the shortcomings of supervised learning?

Supervised learning uses labels, which are slow, expensive, and often biased. However, self-supervised learning makes the models to understand the data first, so it becomes cheaper, faster, and sturdier.

2. Can you think of any ways in which self-supervised learning is similar to human learning?

Most of the experiences are self-supervised. Especially after you move out from your parents' house and become independent in college. There is no one beside you to teach you about everything you see, feel and think. You just have to face it first and learn from it. All those self-supervised learning makes us to have more efficient and flexible decision-making process.

3. What are some specific fields that would benefit from using self-supervised learning for computer vision?

Medical imaging. Because there are just too many data. Hundreds and thousands of them are sitting on a table, and it is nearly impossible to go through each of them manually to label.

Image Segmentation

1. How do superpixels enhance the efficiency of image segmentation algorithms?

They turn a huge, noisy pixel problem into a small region problem, cutting compute and memory while often improving final segmentation quality.

2. In what scenarios might an entity be classified as both a "thing" and "stuff" in image segmentation? How might this dual classification impact the effectiveness of semantic segmentation methods?

When materials form discrete objects, it can be both "stuff" and "thing", like a crowd or people. That ambiguity injects label noise and blurs boundaries for plain semantic segmentation.