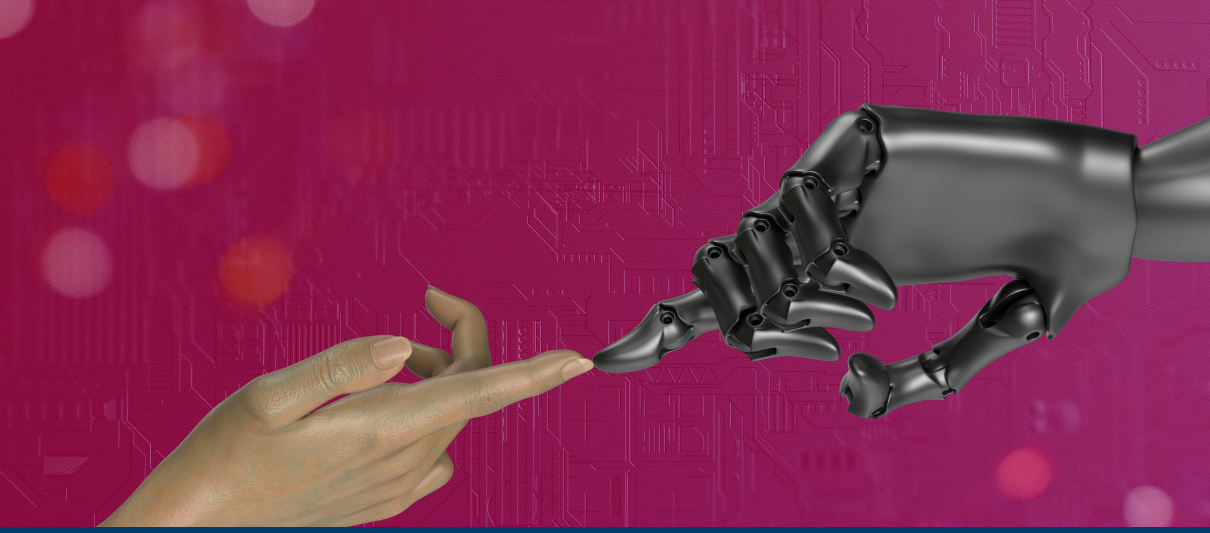


Hands-On Machine Learning

Dr. Derek Bridge | 2026/2027

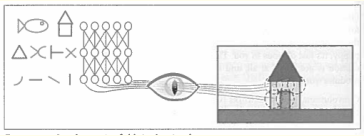
UCC Coláiste na hOllscoile Corcaigh
University College Cork, Ireland

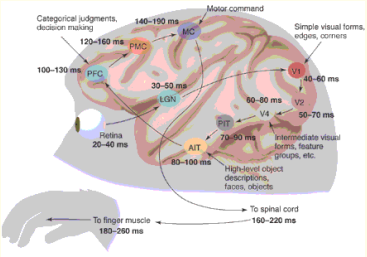
School of Computer Science and Information Technology
Scoil na Ríomheolaíochta agus na Teicneolaíochta Faisnéise

17. Convolutional Neural Networks

In this lecture, we use the MNIST dataset, which we saw in the previous lecture.

Primate Vision





- In the primate vision system, there seems to be a hierarchy of neurons within the visual cortex.
- In the lowest layers,
  - neurons have small local receptive fields, i.e. they respond to stimuli in a limited region of the visual field; and
  - they respond to, e.g., spots of light.
- In higher layers,
  - they combine the outputs of neurons in the lower layers;
  - they have larger receptive fields; and
  - they respond to, e.g., lines at particular orientations.
- In the highest layers,
  - they respond to ever more complex combinations, such as shapes and objects.
- There are perhaps as many as 8 layers in the visual cortex alone.

Convolutional Neural Networks

- **Convolutional Neural Networks** (**convnets** or **CNNs**) are widely used in computer vision and in other perceptual problems including speech recognition and natural language processing.
- We will use 2D CNNs, which are widely used for datasets of images.
- They have nice properties, some of which resemble the visual cortex in primates.

The Properties of CNNs

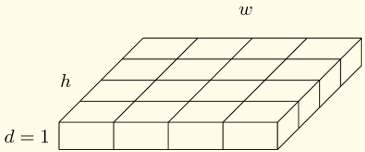
- They learn features that are **translation invariant**. In other words, a filter in a convolutional layer will recognize a feature anywhere in the image: bottom-left, top-right, ...
- They learn **spatial hierarchies** of features: from small local features such as lines in lower layers up to larger shapes in higher layers.

Grayscale Images

- A **grayscale image** has a certain height h and width w.
- Therefore, it makes sense to represent each one as an h by w **matrix** of integers $[0, 255]$.
- However, up to now, we have reshaped them into **vectors**.

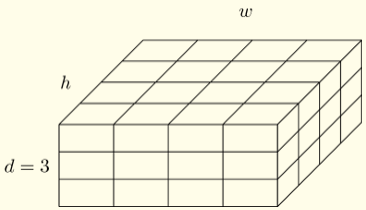
- Question: What is the disadvantage of this? What information gets destroyed?

- So, henceforth, we will not flatten them in this way.
- In fact, for consistency with colour images, we will treat grayscale images as three-dimensional objects with a height h, width w and depth d, but with $d = 1$.



Colour Images

- A **colour image** has a height h, width w and depth d, sometimes called the number of **channels**.
- Question: $d = 3$. Why?



Datasets of images

Datasets of images will have four dimensions: m, h, w, d. What is m?

Datasets of videos

Datasets of videos will have five dimensions. Why?

Tensors

Scalars

A quantity (a number), often referred to in this context as a **scalar**, has no dimensions.

Vectors

A **vector** has one dimension, n.

Matrices

A **matrix** has two dimensions, m and n.

Higher dimensions

We can also have objects that have three or more dimensions.

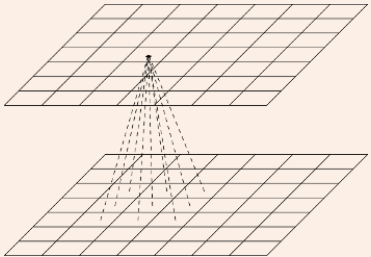
Tensors

We refer to all of these objects as **tensors** and we refer to the number of dimensions as the **rank** of the tensor.

- A scalar is a rank 0 tensor.
- A vector is a rank 1 tensor.
- A matrix is a rank 2 tensor.
- And we can have rank 3 tensors, rank 4 tensors, and so on.

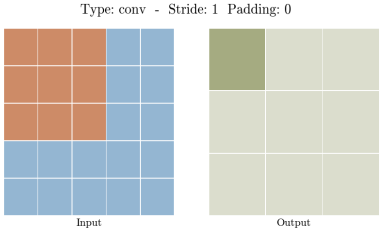
Convolutional Layers

- A **2D Convolutional Layer** outputs a **feature map**, which is a rank 3 tensor of neurons, whose shape is (h, w, d), where d, the depth, is the number of **filters**.
- For simplicity to begin with, let's assume $d = 1$.
- Connections:
  - In the case of a *dense layer*, we saw that every neuron in that layer has connections from every neuron in the preceding layer.
  - But in the case of a *convolutional layer*, every neuron in that layer has connections from only a small rectangular **patch** of neurons in the preceding layer, typically 3×3 or 5×5.



Height and Width of a Feature Map

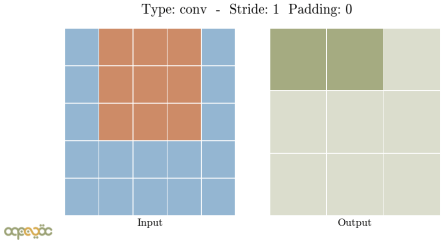
- Suppose the shape of the input is $(28, 28, 1)$ and the patches in the convolutional layer are (3×3).
- The output of the convolutional layer has height of 26 and width of 26. Why?



- In fact, if we wish, we can make the output feature map have the same height and width as the input feature map by using **padding**. Padding adds a border of zeros around the previous layer.
- And, if we wish, we can make the output feature map have even smaller height and width than the input feature map by setting the **stride**. Instead of patches that overlap by 1, we can introduce a distance between successive patches.

Height and Width of a Feature Map

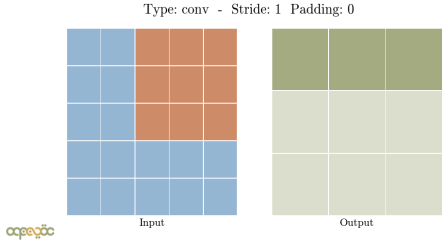
- Suppose the shape of the input is $(28, 28, 1)$ and the patches in the convolutional layer are (3×3).
- The output of the convolutional layer has height of 26 and width of 26. Why?



- In fact, if we wish, we can make the output feature map have the same height and width as the input feature map by using **padding**. Padding adds a border of zeros around the previous layer.
- And, if we wish, we can make the output feature map have even smaller height and width than the input feature map by setting the **stride**. Instead of patches that overlap by 1, we can introduce a distance between successive patches.

Height and Width of a Feature Map

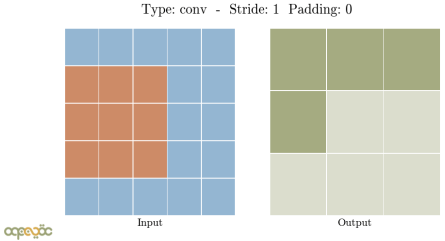
- Suppose the shape of the input is $(28, 28, 1)$ and the patches in the convolutional layer are (3×3).
- The output of the convolutional layer has height of 26 and width of 26. Why?



- In fact, if we wish, we can make the output feature map have the same height and width as the input feature map by using **padding**. Padding adds a border of zeros around the previous layer.
- And, if we wish, we can make the output feature map have even smaller height and width than the input feature map by setting the **stride**. Instead of patches that overlap by 1, we can introduce a distance between successive patches.

Height and Width of a Feature Map

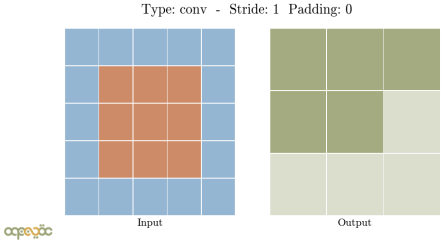
- Suppose the shape of the input is $(28, 28, 1)$ and the patches in the convolutional layer are (3×3).
- The output of the convolutional layer has height of 26 and width of 26. Why?



- In fact, if we wish, we can make the output feature map have the same height and width as the input feature map by using **padding**. Padding adds a border of zeros around the previous layer.
- And, if we wish, we can make the output feature map have even smaller height and width than the input feature map by setting the **stride**. Instead of patches that overlap by 1, we can introduce a distance between successive patches.

Height and Width of a Feature Map

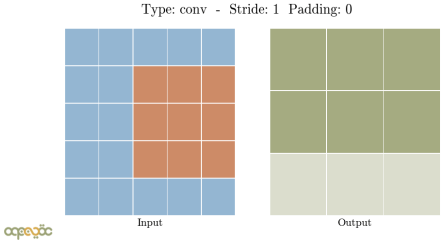
- Suppose the shape of the input is $(28, 28, 1)$ and the patches in the convolutional layer are (3×3).
- The output of the convolutional layer has height of 26 and width of 26. Why?



- In fact, if we wish, we can make the output feature map have the same height and width as the input feature map by using **padding**. Padding adds a border of zeros around the previous layer.
- And, if we wish, we can make the output feature map have even smaller height and width than the input feature map by setting the **stride**. Instead of patches that overlap by 1, we can introduce a distance between successive patches.

Height and Width of a Feature Map

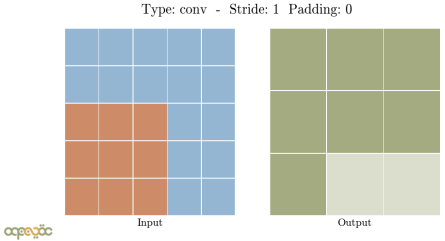
- Suppose the shape of the input is $(28, 28, 1)$ and the patches in the convolutional layer are (3×3).
- The output of the convolutional layer has height of 26 and width of 26. Why?



- In fact, if we wish, we can make the output feature map have the same height and width as the input feature map by using **padding**. Padding adds a border of zeros around the previous layer.
- And, if we wish, we can make the output feature map have even smaller height and width than the input feature map by setting the **stride**. Instead of patches that overlap by 1, we can introduce a distance between successive patches.

Height and Width of a Feature Map

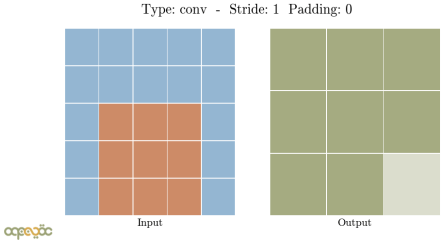
- Suppose the shape of the input is $(28, 28, 1)$ and the patches in the convolutional layer are (3×3).
- The output of the convolutional layer has height of 26 and width of 26. Why?



- In fact, if we wish, we can make the output feature map have the same height and width as the input feature map by using **padding**. Padding adds a border of zeros around the previous layer.
- And, if we wish, we can make the output feature map have even smaller height and width than the input feature map by setting the **stride**. Instead of patches that overlap by 1, we can introduce a distance between successive patches.

Height and Width of a Feature Map

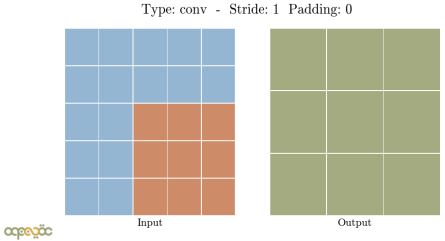
- Suppose the shape of the input is $(28, 28, 1)$ and the patches in the convolutional layer are (3×3).
- The output of the convolutional layer has height of 26 and width of 26. Why?



- In fact, if we wish, we can make the output feature map have the same height and width as the input feature map by using **padding**. Padding adds a border of zeros around the previous layer.
- And, if we wish, we can make the output feature map have even smaller height and width than the input feature map by setting the **stride**. Instead of patches that overlap by 1, we can introduce a distance between successive patches.

Height and Width of a Feature Map

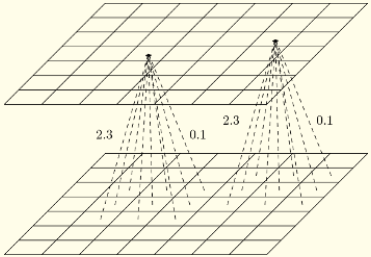
- Suppose the shape of the input is $(28, 28, 1)$ and the patches in the convolutional layer are (3×3).
- The output of the convolutional layer has height of 26 and width of 26. Why?



- In fact, if we wish, we can make the output feature map have the same height and width as the input feature map by using **padding**. Padding adds a border of zeros around the previous layer.
- And, if we wish, we can make the output feature map have even smaller height and width than the input feature map by setting the **stride**. Instead of patches that overlap by 1, we can introduce a distance between successive patches.

The Weights of a Filter in a Convolutional Layer

- The idea of a filter is that it will learn a specific aspect (feature) of its input:
  - e.g. the presence of a vertical line;
  - e.g. the presence of a pair of eyes.
- Within one filter, all neurons share the same weights!

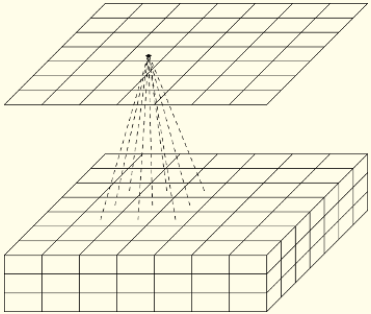


Advantages

- This reduces the number of parameters that must be learned.
- More importantly, it means that the filter will respond to the presence of that feature no matter where it is in the input (the translational invariance, mentioned earlier).

Convolutional Layers use Stacks of Filters

- Now consider the case where $d > 1$: the convolutional layer comprises a stack of d filters.
- A neuron in a filter in a convolutional layer is connected to a patch of neurons in each of the filters of the previous layer (or, in the case of the first layer, in each of the channels of the input).
- Note how this means that a filter in one layer combines several filters (or channels) of the previous layer (the spatial hierarchy, mentioned earlier).

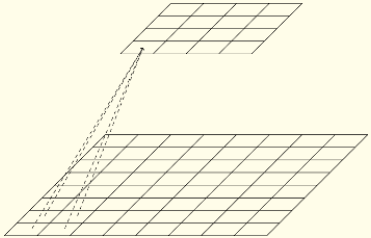


Pooling Layers

- Convolutional Layers are often followed by **Pooling Layers**.
- The goal is to have a layer that shrinks the number of neurons in higher layers:
  - to reduce the amount of computation;
  - to reduce memory usage;
  - to reduce the number of parameters to be learned, thus reducing the risk of overfitting; and
  - to create a hierarchy in which higher convolutional layers contain information about the totality of the original input image.

Height and Width of a Pooling Layer

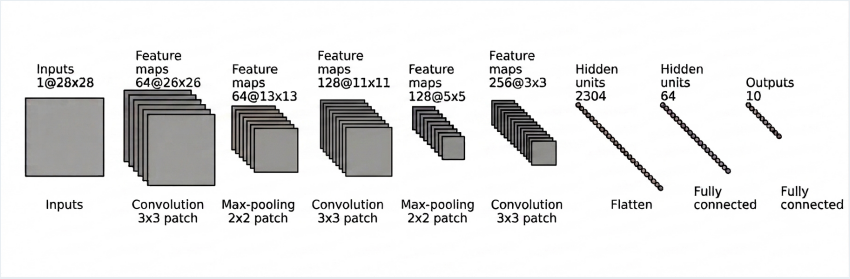
- Like Convolutional Layers, a Pooling Layer works on rectangular patches: neurons in the pooling layer are connected to patches of neurons in the previous layer
  - typically 2×2;
  - typically adjacent rather than overlapping.
- E.g. if the previous layer has height h and width w, and the pooling layer uses adjacent 2×2 pooling patches, then the pooling layer will have height $h/2$ and width $w/2$.
- The depth of the pooling layer is the same as the depth of the previous layer.



Max Pooling Layers

- Pooling layers have no weights: nothing to learn.
- In a **Max Pooling Layer**, a neuron in the pooling layer receives the outputs of the neurons in the patch in the previous layer and outputs only the *largest* of them.
- Pooling layers work on the filters independently, which is why they have the same depth as the previous layer.

Examples


Inputs
1@28x28
Feature
maps
64@26x26
Feature
maps
64@13x13
Feature
maps
128@11x11
Feature
maps
128@5x5
Feature
maps
256@3x3
Hidden
units
2304
Hidden
units
64
Outputs
10
Inputs
Convolution
3x3 patch
Max-pooling
2x2 patch
Convolution
3x3 patch
Max-pooling
2x2 patch
Convolution
3x3 patch
Flatten
Fully
connected
Fully
connected

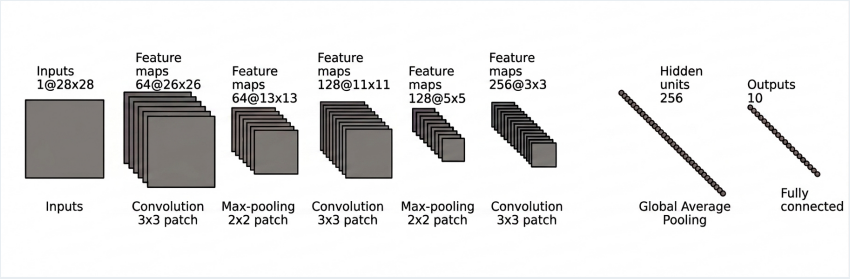

Inputs
1@28x28
Feature
maps
64@26x26
Feature
maps
64@13x13
Feature
maps
128@11x11
Feature
maps
128@5x5
Feature
maps
256@3x3
Hidden
units
256
Outputs
10
Inputs
Convolution
3x3 patch
Max-pooling
2x2 patch
Convolution
3x3 patch
Max-pooling
2x2 patch
Convolution
3x3 patch
Global Average
Pooling
Fully
connected

Image credits

Diagram taken from A. Géron: Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (2nd edn), O'Reilly, 2019

Image from Simon J. Thorpe & Michèle Fabre-Thorpe: Seeking Categories in the Brain. Science, vol.291, pp.260–263 (2001).

Image from Machine Learning Animations