Numerical Simulation Laboratory

LECTURE 12

MACHINE LEARNING
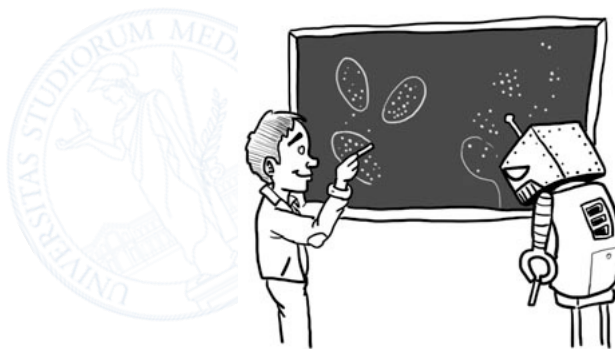
1

# Outline

- Introduction to Convolutional Neural Networks

2

Locality and symmetries in the dataset

- One of the core lessons of physics is that we should exploit symmetries and invariances when analyzing physical systems.

- Properties such as locality and translational invariance are often built directly into the physical laws. For example, we know that in many cases it is sufficient to consider only local couplings in our Hamiltonians, or that a model, in which local couplings depend only on the distances among the degrees of freedom, is translationally invariant and thus the total momentum is conserved, etc.

- Like physical systems, many datasets (especially datasets from Physics) and supervised learning tasks also possess additional symmetries and structure.

- For instance, consider a supervised learning task where we want to label images from some dataset as being pictures of cats or not. Our statistical procedure must first learn features associated with cats.
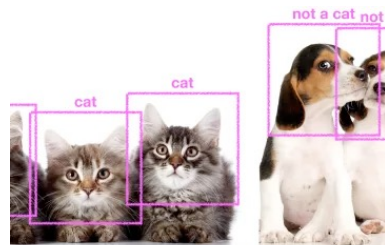
Cat   Cat   Representation

- Because a cat is a physical object, we know that these features are likely to be local (whiskers, tails, eyes, etc).

- We also know that the cat can be anywhere in the image. Thus, it does not really matter where in the picture these features occur (though relative positions of features likely do matter). This is a manifestation of translational invariance that is built into our supervised learning task.

- This example makes clear that, like many physical systems, many ML tasks (especially in the context of image processing) also possess additional structure, such as locality and translation invariance.

- The all-to-all coupled neural networks in the previous section typically fail to exploit this additional structure

Convolutional Neural Networks (CNNs)

- Not surprisingly, the neural networks community realized these problems and designed a class of neural network architectures, convolutional neural networks or CNNs, that take advantage of this additional structure (locality and translational invariance)

- A convolutional neural network is a "translationally invariant" neural network that respects locality of the input data.

- There are two kinds of basic layers that make up a CNN: a convolution layer that computes the convolution of the input with a series of filters, and pooling layers that coarse-grain the input while maintaining locality and spatial structure
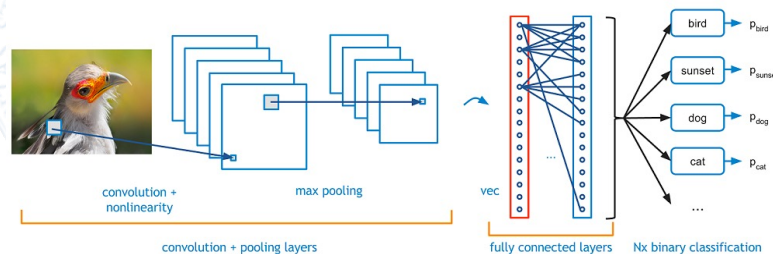
5

- For 2D data, a layer l is characterized by three numbers: height H_l, width W_l, and the "number of channels" D_l

- The height and width correspond to the sizes of the two-dimensional spatial $(W_l; H_l)$-plane (in neurons) (e.g. if l is the input layer, $W_l \times H_l$ is the number of pixels of the image), and the depth D_l to the "number of channels" in that layer (e.g. 3 channels for an RGB image).

- In general, we will be concerned with local spatial filters (often called a receptive field in analogy with neuroscience) that take as inputs a small spatial patch of the previous layer at all depths. The convolution consists of running this filter over all locations in the spatial plane

- To demonstrate how this works in practice, let us a consider the simple example consisting of a one-dimensional input of depth 1 (1 channel), shown in figure

- In this case, a filter of size Fx1x1 can be specified by a vector of weights w of length F
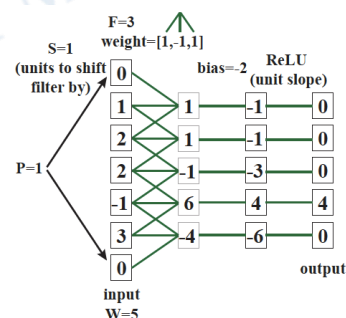
6

- The stride, S, encodes by how many neurons we translate the filter by when performing the convolution.

- In addition, it is common to pad the input with P zeros. For an input of width W, the number of neurons (outputs) in the layer is given by 1+(W–F+2P)/S.

- Below I show you some convolution procedures, for a square input of unit depth:

   N.B.: Blue maps are inputs, and cyan maps are outputs.

| No padding, no strides | Arbitrary padding, no strides | No padding, strides | Padding, strides |

- In a CNN, the convolutional layer applies a set of filters to the input layer to extract features. Each filter performs a convolution operation by sliding over the input layer and computing a dot product between the filter weights and the corresponding region of the input.

- If you apply a convolutional layer with, for example, 2 filters on an input layer that contains 3 channels, it means that each filter will have a depth of 3. The depth of a filter corresponds to the number of channels it operates on.

- During the convolution operation, each filter will independently convolve with the input across all three channels. As a result, for each filter, you obtain a single channel output.

- The reason for having multiple channels in the output is to capture different types of features. Each filter learns to detect specific patterns or features in the input.

- These convolutional layers are interspersed with pooling layers that coarse-grain spatial information by performing a subsampling at each depth. One common pooling operation is the max pool. A simple example of a max-pooling operation is shown in figure

- In a max pool, the spatial dimensions are coarse-grained by replacing a small region (say 2x2 neurons) by a single neuron whose output is the maximum value of the output in the region. In physics, this pooling step is very similar to the decimation step in the Renormalization Group

- This reduces the dimension of outputs. For example, if the region we pool over is 2x2, then both the height and the width of the output layer will be halved.

- Generally, pooling operations do not reduce the "number of channels" of the convolutional layers because pooling is performed separately on each channel.
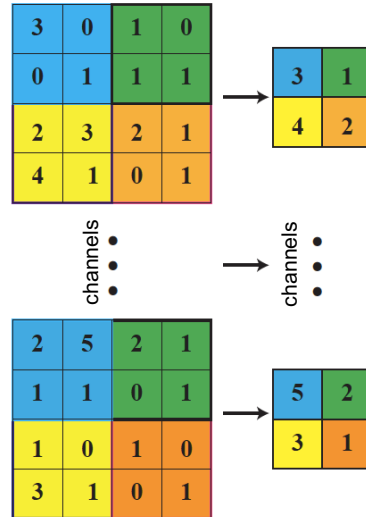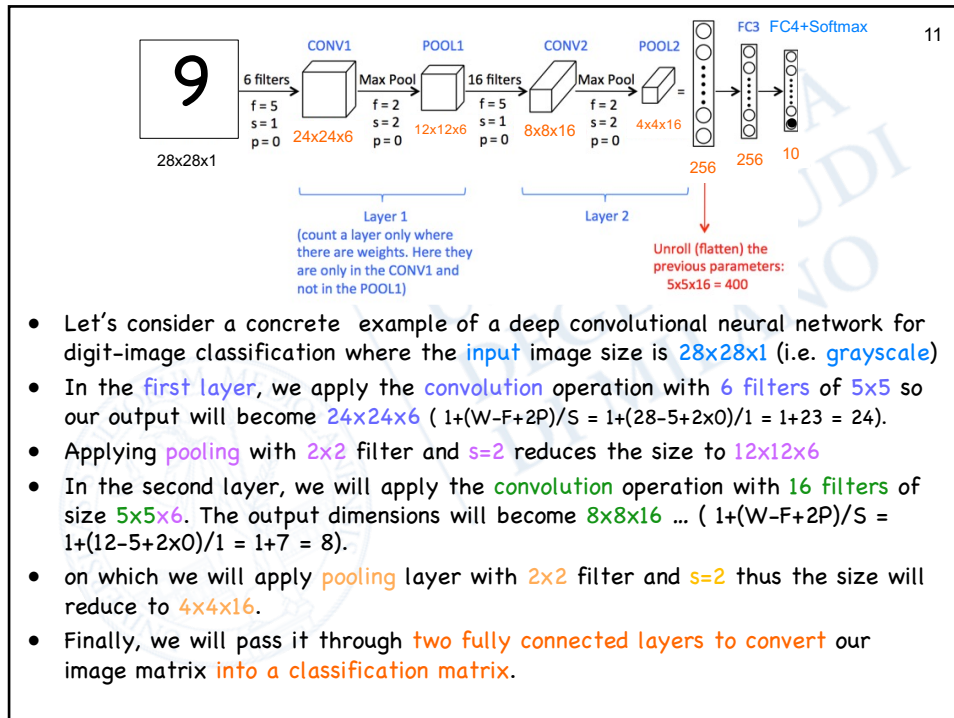

- In a CNN, the convolution and max-pool layers are generally followed by an all-to-all connected layer and a classifier such as a soft-max.

- CNNs are trained using the backpropagation algorithm. From a backpropagation perspective, CNNs are almost identical to fully connected NN architectures except with tied parameters …

- In fact, all neurons at a given convolutional layer represent the same filter, and hence can all be described by a single set of weights and biases. This reduces the number of free parameters by a factor of HxW at each layer. For example, for a layer with $D=10^2$ and $H=W=10^2$, this gives a reduction in parameters of nearly 10^6 … this allows for the training of much larger models!

- The immense success of CNNs for image recognition resides in the fact that filters (receptive fields) learned by the convolution layers should work well on similar tasks, not just on the ones they were originally trained for. We expect that, since images reflect the natural world, the filters learned by CNNs should transfer over to new tasks with only slight modifications and fine-tuning.

Layer 1
(count a layer only where there are weights. Here they are only in the CONV1 and not in the POOL1)

Layer 2

Unroll (flatten) the previous parameters:
5x5x16 = 400

- Let's consider a concrete example of a deep convolutional neural network for digit-image classification where the input image size is 28x28x1 (i.e. grayscale)
- In the first layer, we apply the convolution operation with 6 filters of 5x5 so our output will become 24x24x6 (1+(W-F+2P)/S = 1+(28-5+2x0)/1 = 1+23 = 24).
- Applying pooling with 2x2 filter and s=2 reduces the size to 12x12x6
- In the second layer, we will apply the convolution operation with 16 filters of size 5x5x6. The output dimensions will become 8x8x16 … (1+(W-F+2P)/S = 1+(12-5+2x0)/1 = 1+7 = 8).
- on which we will apply pooling layer with 2x2 filter and s=2 thus the size will reduce to 4x4x16.
- Finally, we will pass it through two fully connected layers to convert our image matrix into a classification matrix.

11

DNNs vs CNNs in action!

- Some fun with CNNs at:

  https://adamharley.com/nn_vis/cnn/2d.html

  Compare with a fully connected DNN at:

  https://adamharley.com/nn_vis/mlp/2d.html

12

Lecture 12: Suggested books

- P. Mehta, C.H. Wang, A.G.R. Day, and C. Richardson, *A high-bias, low-variance introduction to Machine Learning for physicists*, arXiv:1803.08823 (Feb 2019)

- S. Haykin, *Neural Networks and Learning Machines*, Pearson (2009)

- S. Theodoridis, *Machine Learning. A Bayesian and Optimization Perspective*, Elsevier (2015)

- K.P. Murphy, *Machine Learning. A Probabilistic Perspective*, MIT press (2012)

13

13

14