Title: Machine Learning Lecture - 230314

Speakers: Lauren Hayward

Collection: Machine Learning for Many-Body Physics (2022/2023)

Date: March 14, 2023 - 9:00 AM

URL: https://pirsa.org/23030032

Today: convolutional neural networks (CNNs)

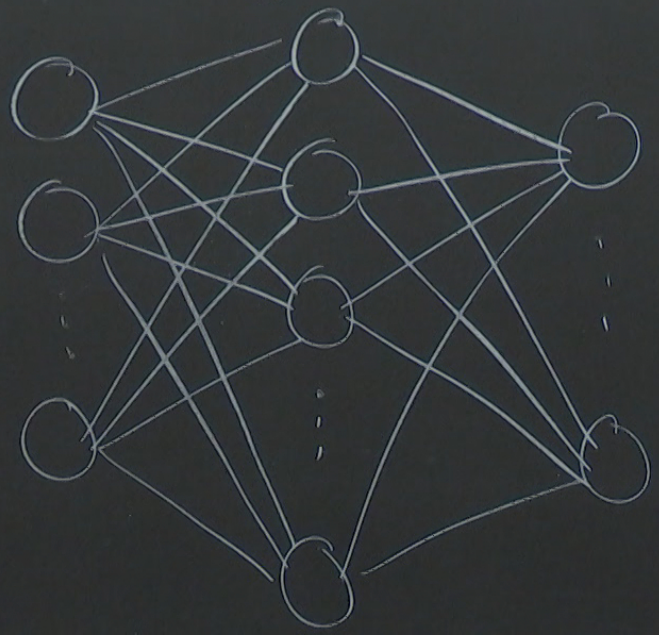
↳ local receptive fields

↳ channels

↳ weight and bias sharing

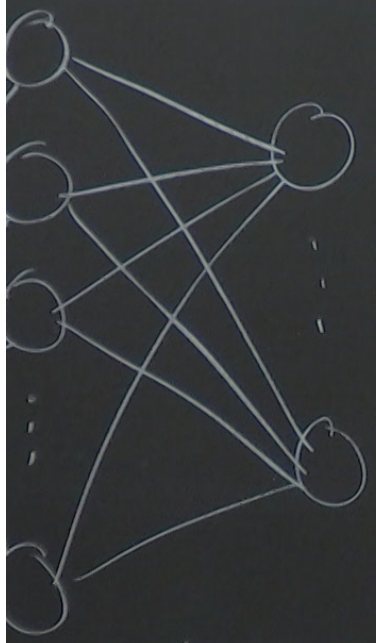
↳ coarse-graining through pooling

Lectures 3-6: fully-connected feedforward NNs (FFNNs)



Here every in layer l is connected to every neuron in layer $l+1$

-connected feedforward NNs (FFNNs)



nnected to every neuron
in layer l+1

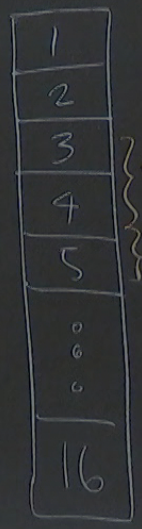
Potential issue: FFNNs do not exploit properties such as locality and translational invariance that are inherent to many physical systems and images

⇒ a lot of spatial info is lost or must be learned

Consider a 4×4 image (picture of a dog, Ising spin config, etc.). Let us label pixels (sites) as:

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

As input to a FFNN, image gets flattened:

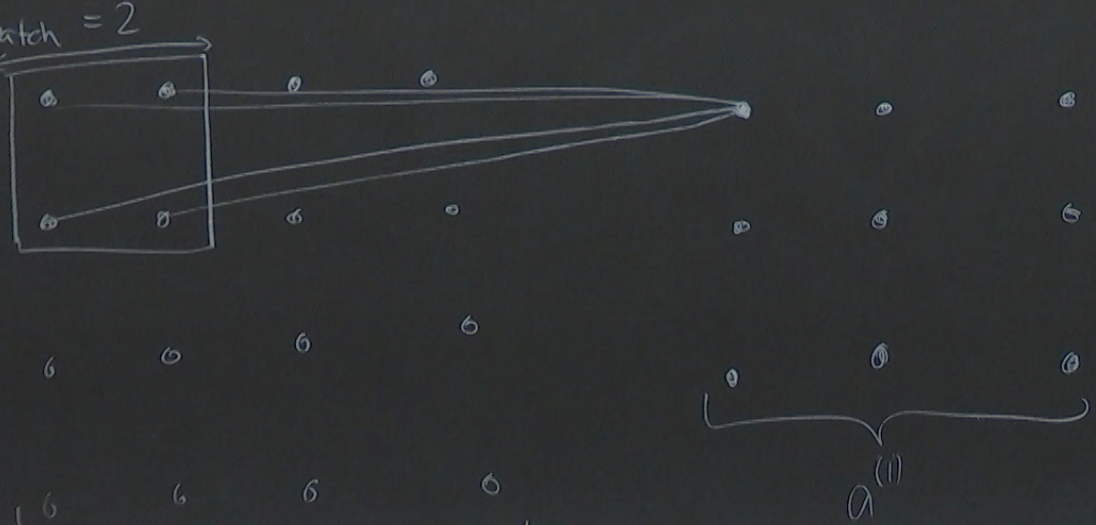


locality is easy lost

Solution: local receptive fields (LRFs)

Idea: process the data in local "patches" that keep the inherent dimensional structure

eg)

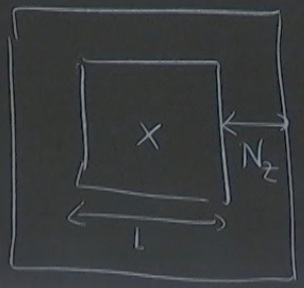


Here we start with a 4x4 input, apply filters (components of a weight matrix) of patch size 2x2, and get a 3x3 set of neurons on the next layer

16

Generalizations:

- patch size: can be a square of size $L_{patch} \times L_{patch}$ or a general shape
- padding: We can pad the image with N_z zeros around each edge

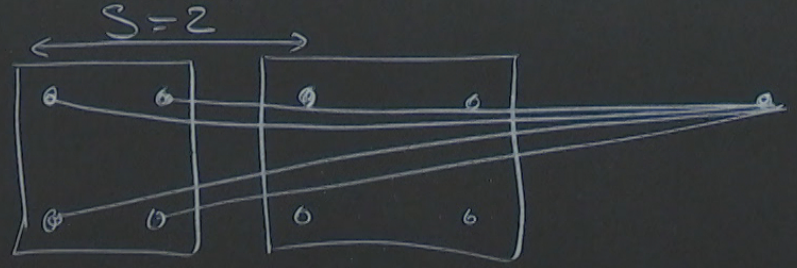


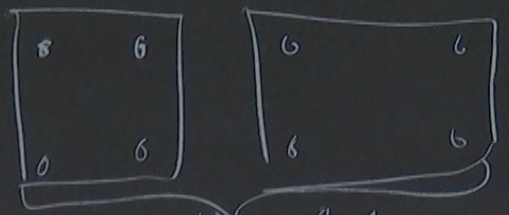
input $x = a^{(0)}$

$a^{(1)}$ neurons

- Stride: we can take a step of size S before applying the next filter

eg)





$a^{(0)}$ is 4x4

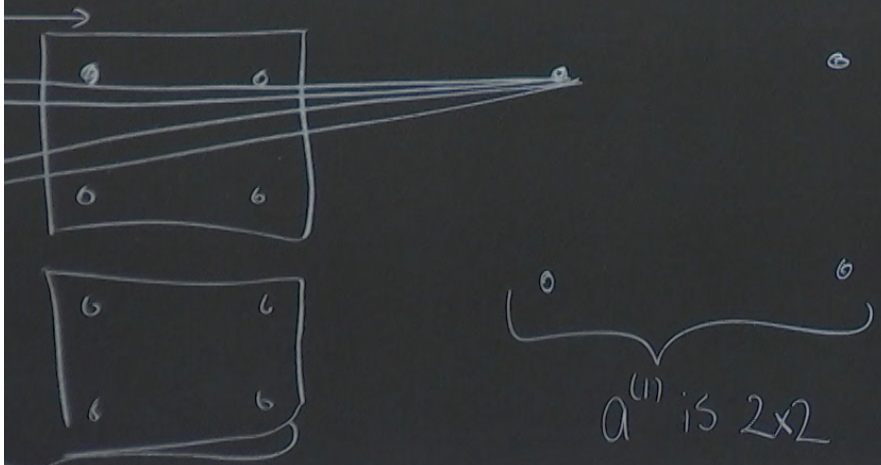


$a^{(1)}$ is 2x2

edge

$= a^{(0)}$

we can take a step of S before applying the lter

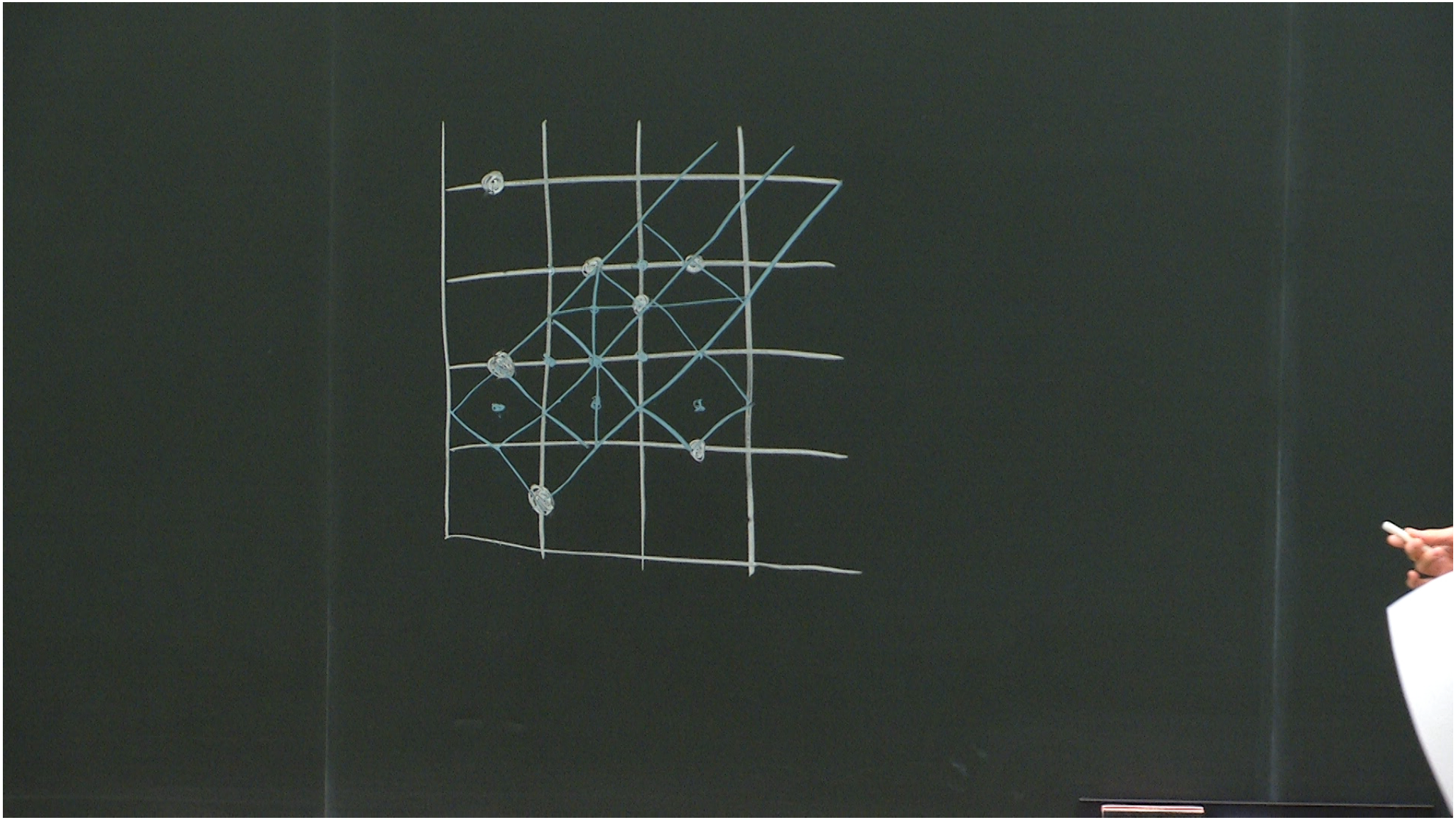


If $a^{(0)}$ has $n_0 = L \times L$ neurons then $a^{(1)}$ has $n_1 = L_1 \times L_1$, where

$$L_1 = \frac{L - L_{patch} + 2N_z}{S} + 1$$

If we apply periodic BC instead of padding:

$$L_1 = \frac{L}{S}$$



For a 2d image, in addition to spatial x and y coords., we can have different "channels"

input $a^{(0)}_{x,y,c}$

x, y: spatial indices

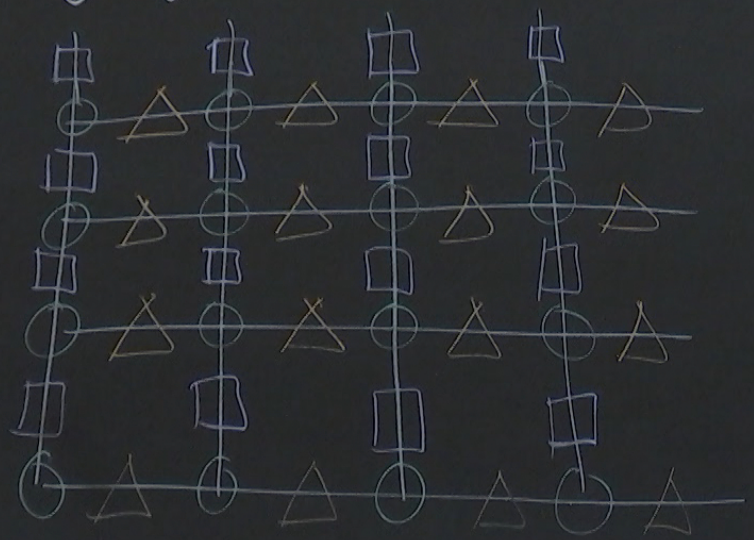
c: channel index

Channels:

- For colour images, c can label R, G, B channels
- For spin configs on lattices, c can label the sublattices

Each filter is then of size $L_{patch} \times L_{patch} \times \underbrace{N_c}_{\#\text{ of channels}}$

For example, recall the lattice we used for the Ising Z_2 gauge theory

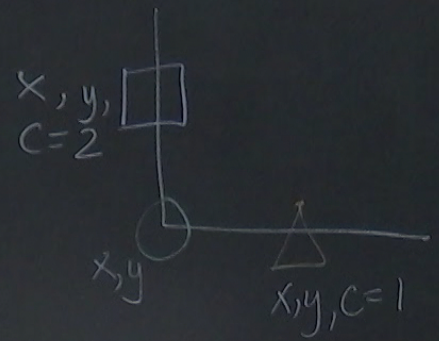


○ : lattice sites

△ : sublattice 1

□ : sublattice 2

For each lattice site (x,y) we must consider two points (one on each sublattice)



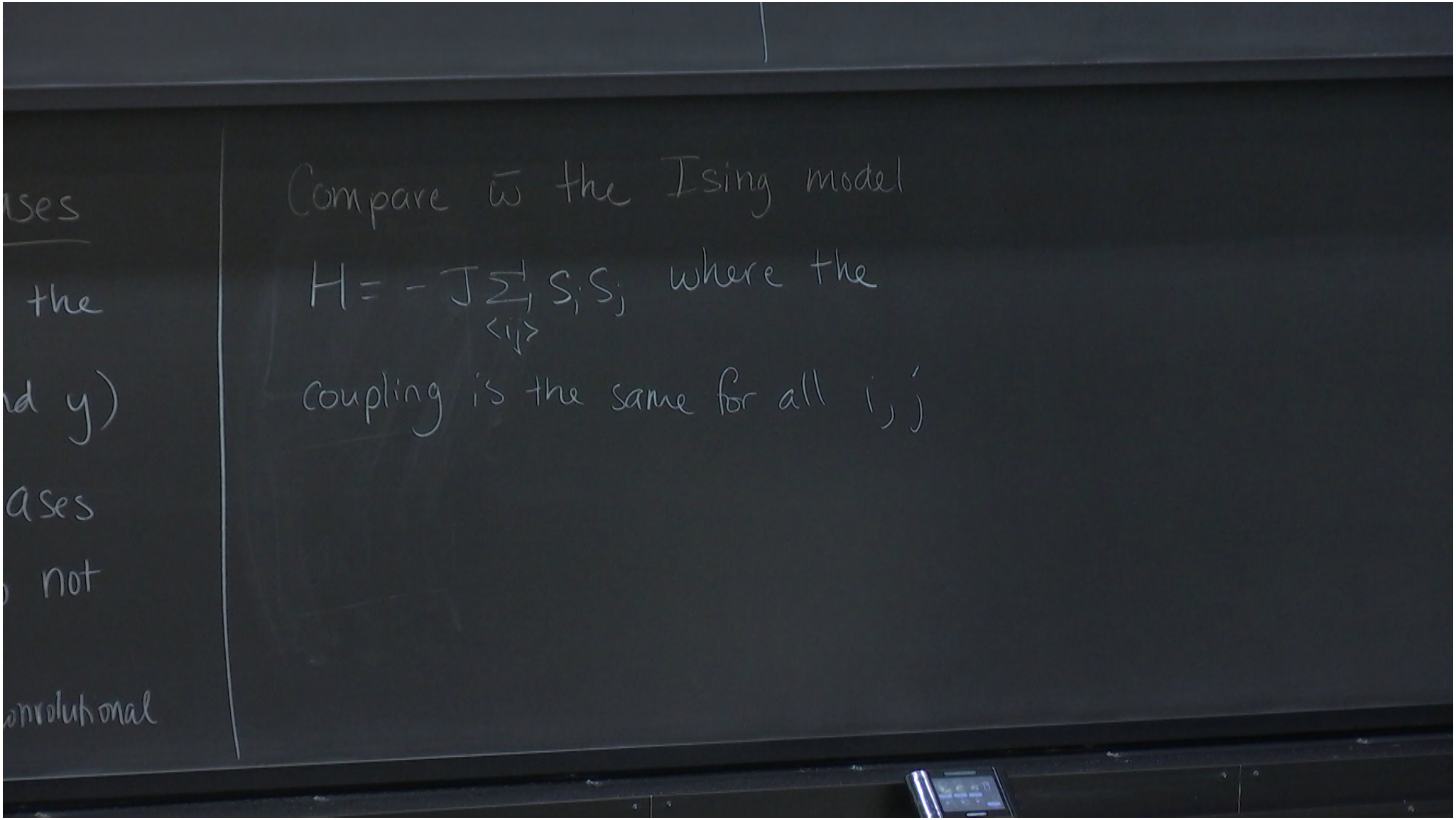


Shared weights and biases

Idea: the LRF moves across the input data (along x and y)

but the weights and biases associated to the filters do not change $\bar{w}$ x and y

$\Rightarrow$ the transformation applied by the convolutional layer is translationally invariant


ases
the
nd y)
ases
not
onvolutional
Compare ω the Ising model
$H = -J \sum_{\langle i,j \rangle} S_i S_j$ where the
coupling is the same for all i, j

At layer ℓ, we can apply N_f different filters of size $L_{patch} \times L_{patch} \times N_c$ to the output $a^{(\ell-1)}$ so that:

$$a^{(\ell)}_{x,y,k} = g_\ell\left(\sum_{m=1}^{L_{patch}} \sum_{n=1}^{L_{patch}} \sum_{c=1}^{N_c} a^{(\ell-1)}_{x+m-1,\, y+n-1,\, c} W^{(\ell)}_{mnck} + b^{(\ell)}_k\right)$$

activation function

convolution

bias

Here stride $S=1$ for simplicity.

The weight indices are:

$$W_{m\ n\ c\ k}$$

- m: slides along x
- n: slides along y
- c: channel/sublattice index
- k: labels the filters ($1 \le k \le N_f$)

With shared weights & biases, CNNs tend to have far fewer free params. to learn during training

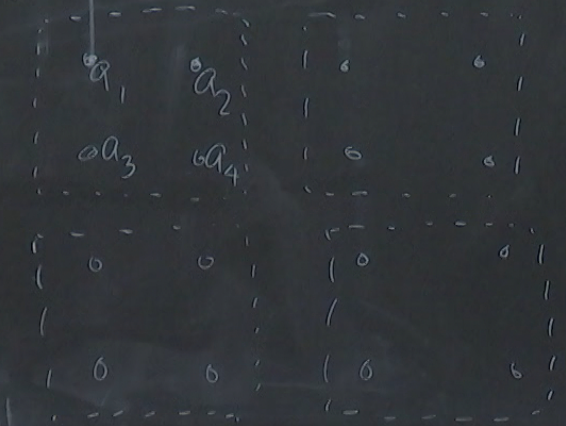
$\Rightarrow$ can often train CNNs for larger input and deeper layers compared to FFNNs

(assuming the data translationally inv. and local)

Pooling:

- Idea: perform coarse-graining or "down-sampling"

- Pooling especially useful for data w/ high correlations (redundant info)

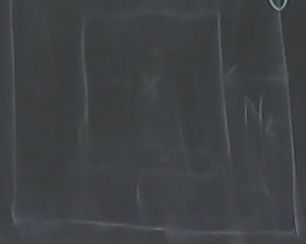
eg) 

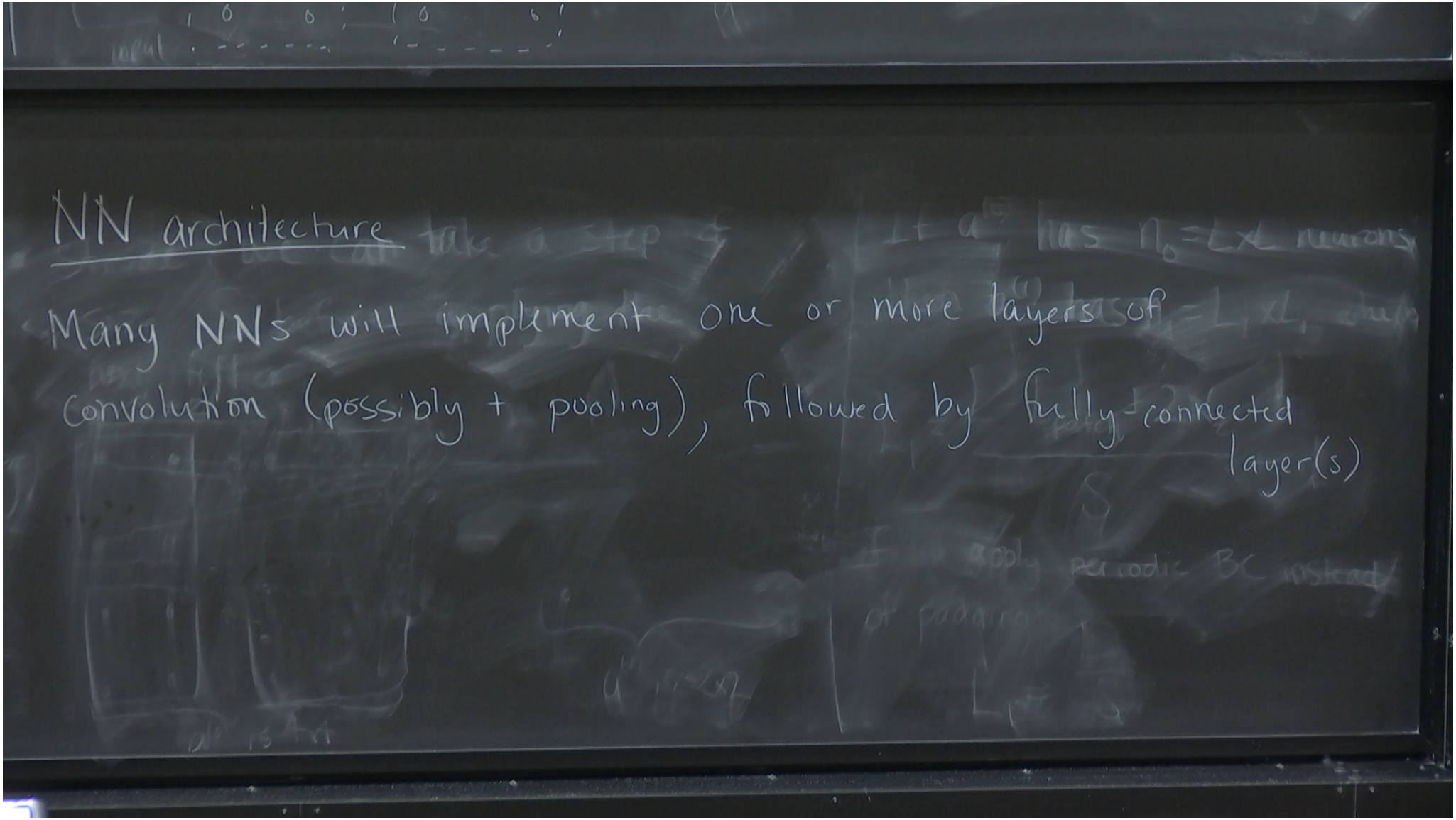
where □ can be
- $\max(a_1, a_2, a_3, a_4)$
- average $\frac{1}{4}(a_1 + a_2 + a_3 + a_4)$
- majority rule (if 'a' discrete)

Pooling notes

- information is lost but training can be faster

- usually pooling is performed separately for each channel (no coarse-graining across channels)




NN architecture
Many NNs will implement one or more layers of convolution (possibly + pooling), followed by fully-connected layer(s)