Title: Lecture - Machine Learning, PHYS 777

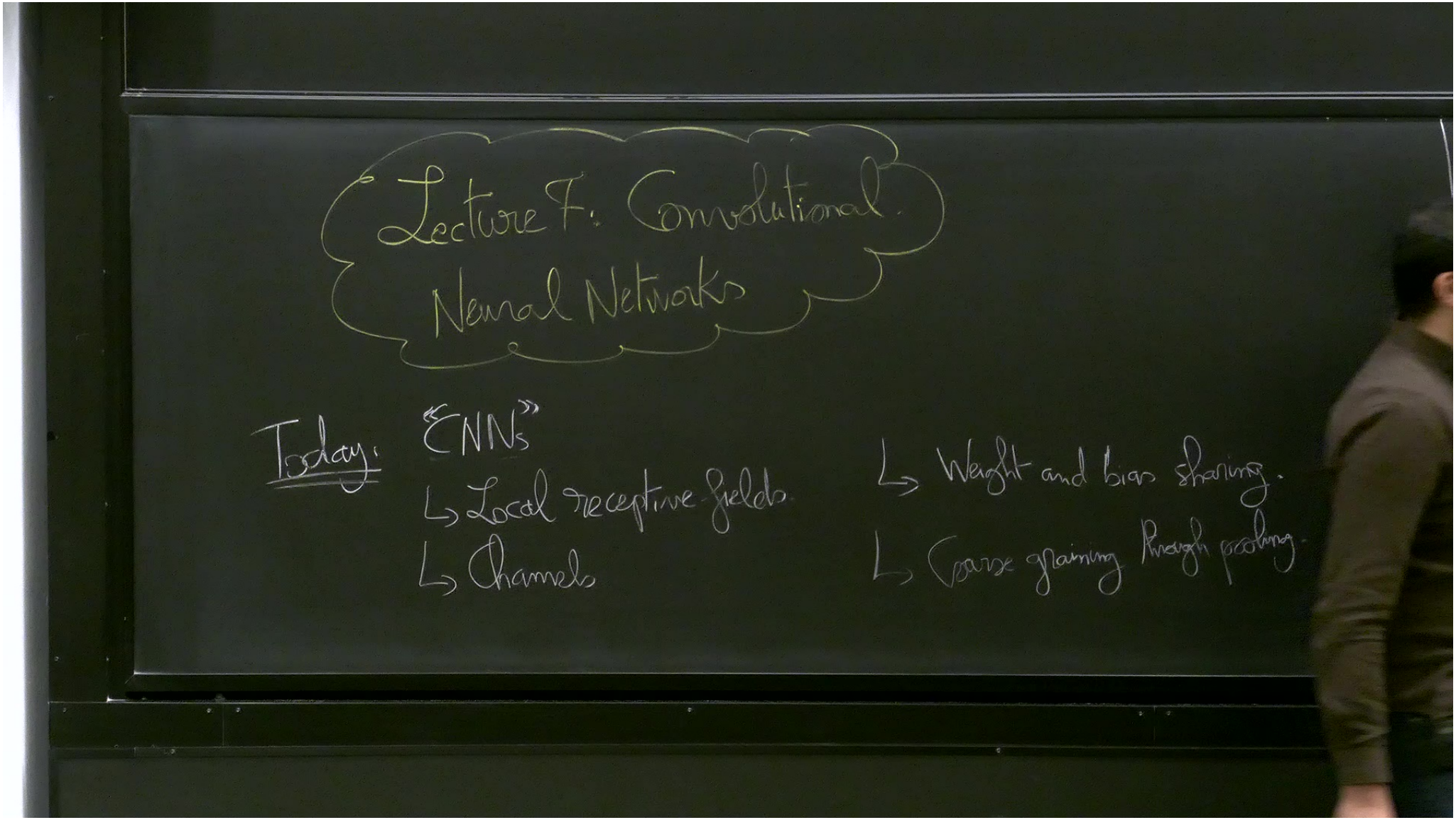
Speakers: Mohamed Hibat Allah

Collection/Series: Machine Learning (Elective), PHYS 777, February 24 - March 28, 2025

Subject: Condensed Matter, Other

Date: March 11, 2025 - 9:00 AM

URL: https://pirsa.org/25030038

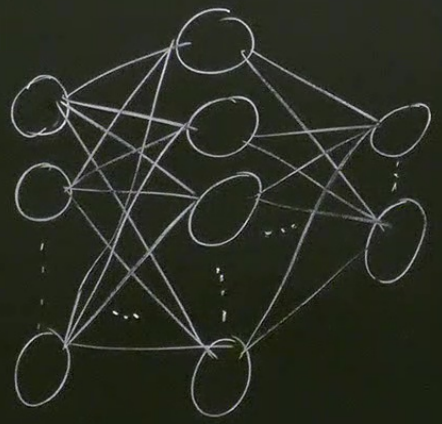

Lecture 7: Convolutional Neural Networks
Today: "CNNs"
↳ Local receptive fields
↳ Channels
↳ Weight and bias sharing.
↳ Coarse graining through pooling.

Lecture 7: Convolutional Neural Networks

Today: "CNNs"

- Local receptive fields
  - Weight and bias sharing.
- Channels
- Coarse graining through pooling.

Lectures 3-6. Fully-connected feed-forward neural networks (FFNNs)

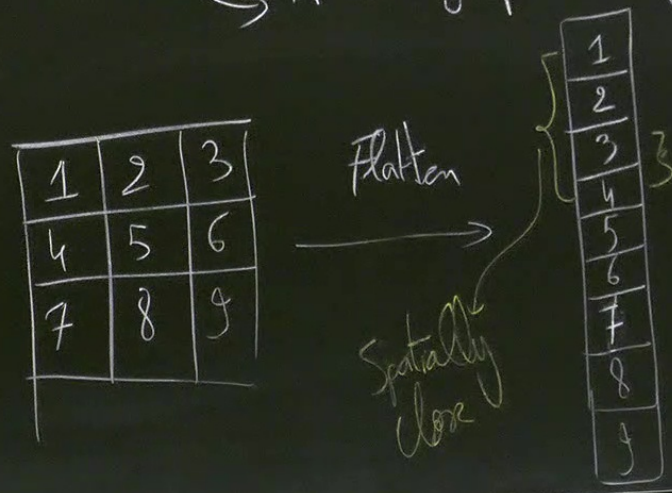


Every neuron in layer l is connected

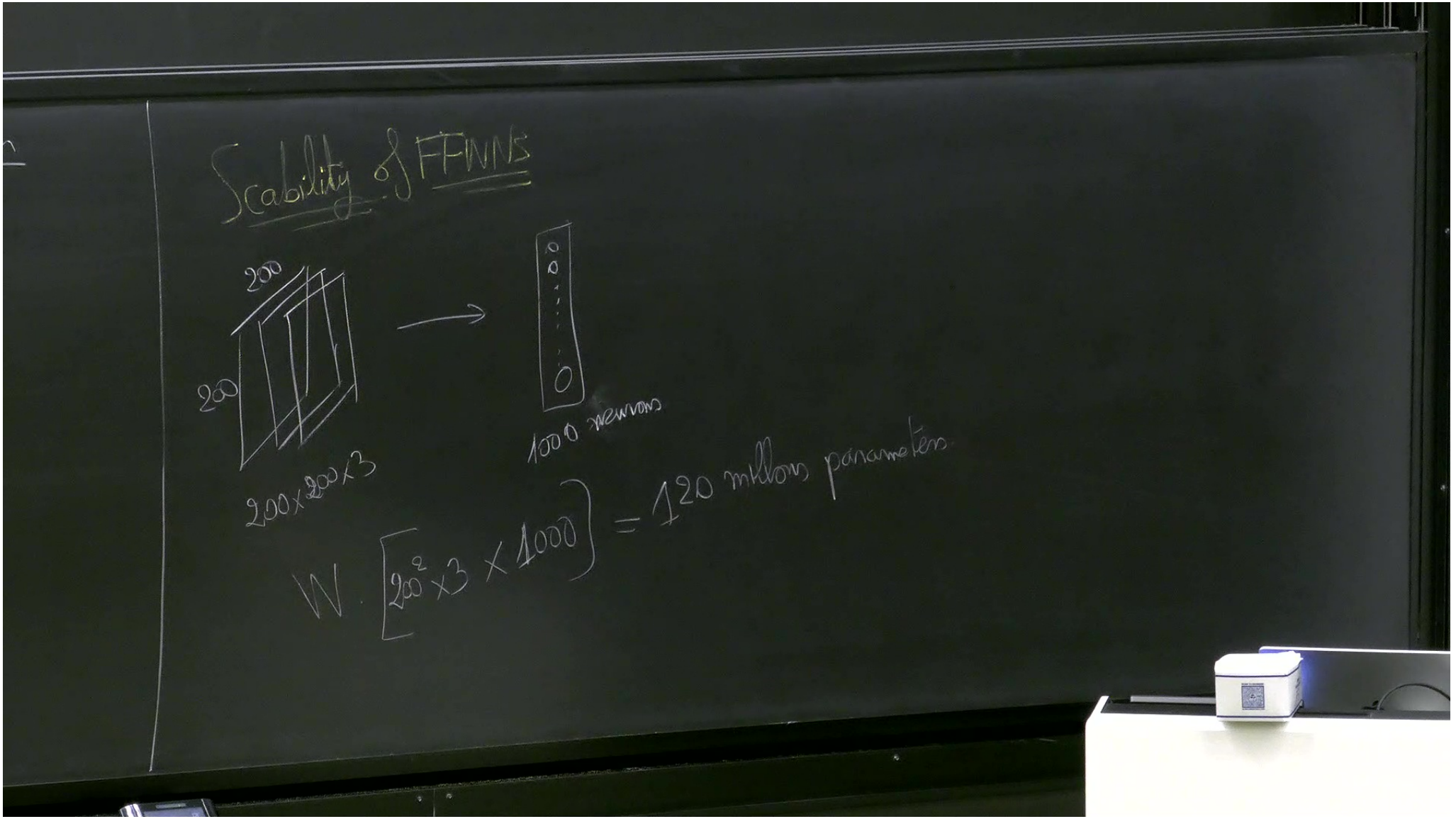
to every neuron in layer $l+1$

Issue: FNNs do not exploit properties such as locality and translation invariance

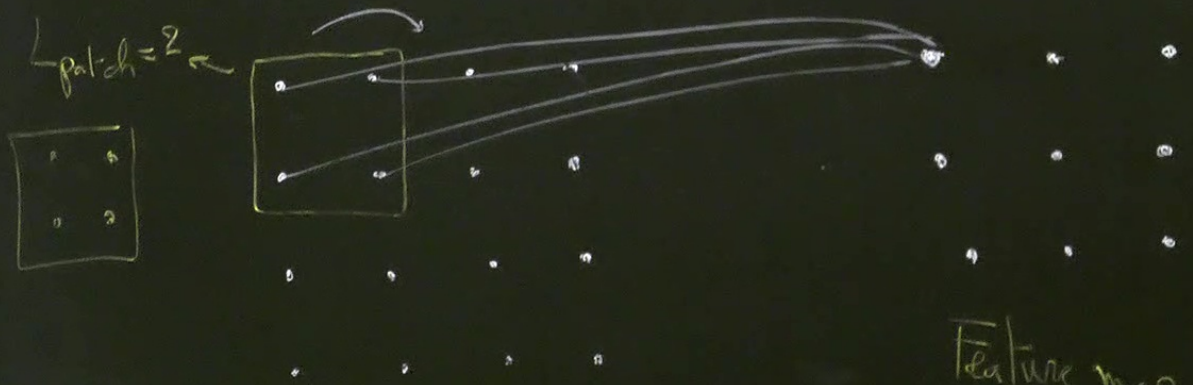
↳ A lot of spatial information is lost or must be learned.



« Locality is lost »


Scability of FFNNs
200
200
200×200×3
1000 neurons
W · [200² × 3 × 1000] = 120 million parameters.

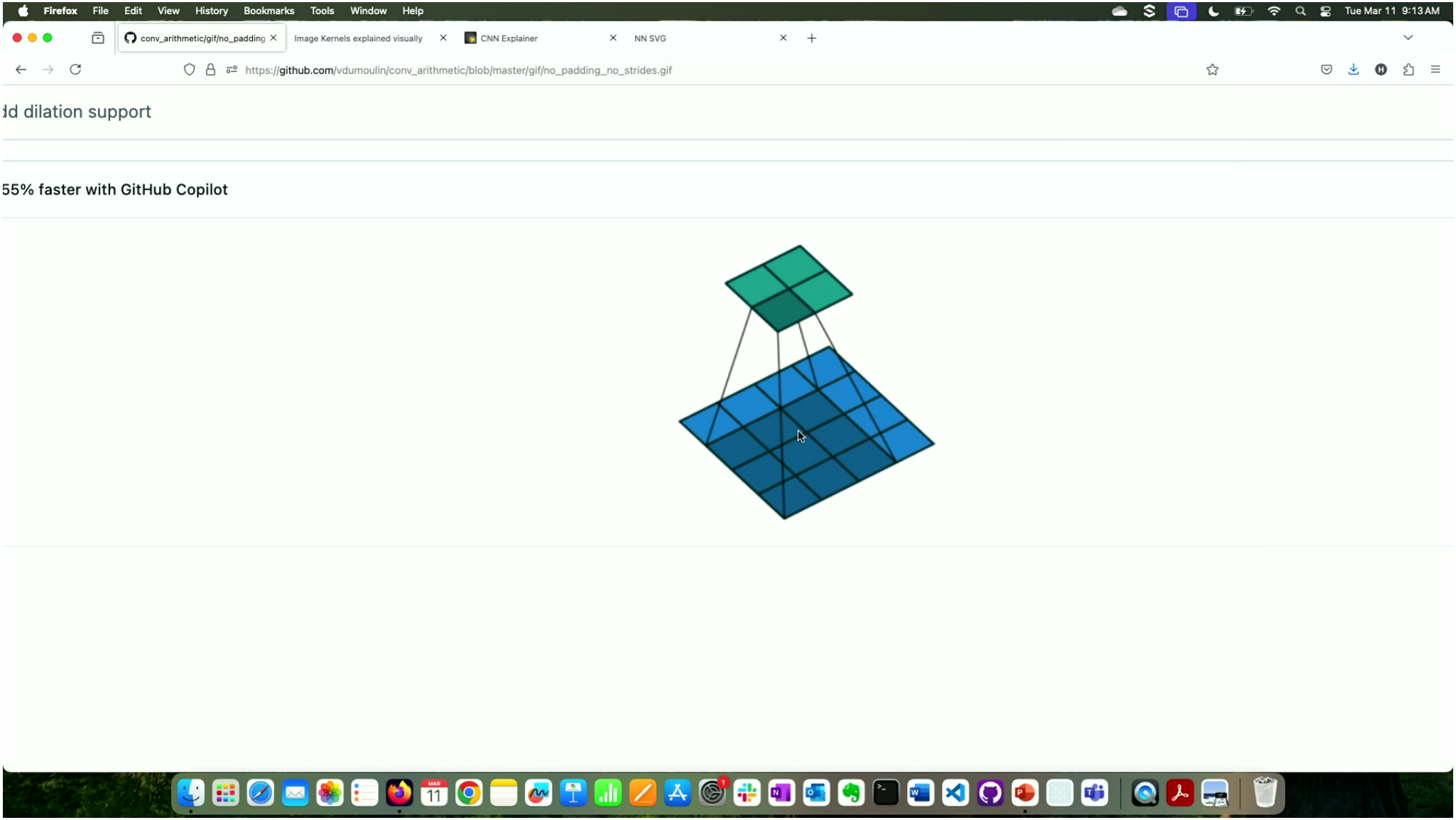
Solution: Convolution with local receptive fields / Filters / Patches...



$x = a^{(0)}$

Feature map.

Convolution operation $a^{(1)}$

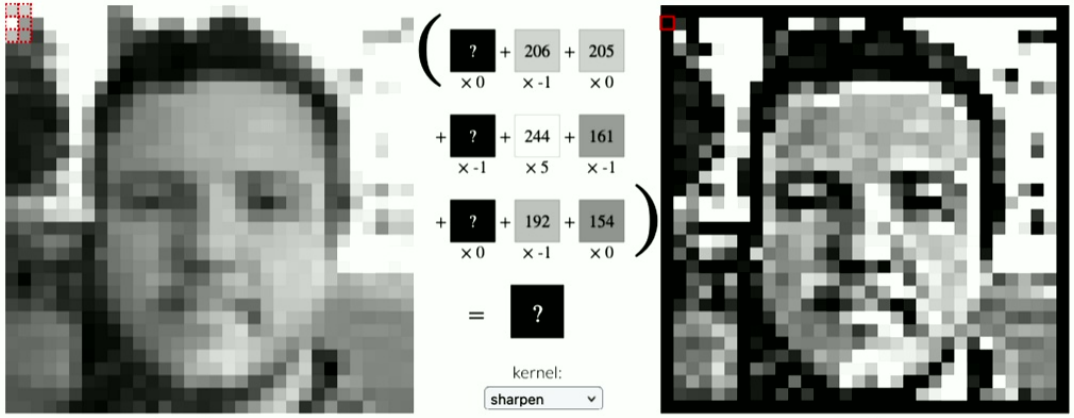

Firefox File Edit View History Bookmarks Tools Window Help
Tue Mar 11 9:13 AM
conv_arithmetic/gif/no_padding
Image Kernels explained visually
CNN Explainer
NN SVG
https://github.com/vdumoulin/conv_arithmetic/blob/master/gif/no_padding_no_strides.gif
dd dilation support
55% faster with GitHub Copilot

Let's walk through applying the following 3x3 **sharpen** kernel to the image of a face from above.

sharpen

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.



input image output image

One subtlety of this process is what to do along the edges of the image. For example, the top left corner of the input image only has three neighbors. One way to fix this is to extend the edge values out by one in the original image while keeping our new image the same size. In this demo, we've instead ignored those values by making them black.

Here's a playground were you can select different kernel matrices and see how they effect the original image or build your own kernel. You can also upload your own image or use live video if your browser supports it.

Files
master
Go to file t
arbitrary_padding_no_strides.gif
arbitrary_padding_no_strides_tr...
dilation.gif
full_padding_no_strides.gif
full_padding_no_strides_transp...
no_padding_no_strides.gif
no_padding_no_strides_transpo...
no_padding_strides.gif
no_padding_strides_transposed...
padding_strides.gif
padding_strides_odd.gif
padding_strides_odd_transpose...
padding_strides_transposed.gif
same_padding_no_strides.gif
same_padding_no_strides_trans...
natbib
pdf
png
templates
.gitignore
LICENSE
README.md
bibliography.bib
conv_arithmetic.tex

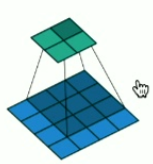
Convolution arithmetic

A technical report on convolution arithmetic in the context of deep learning.

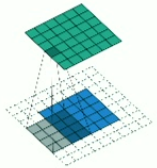
- [1] Vincent Dumoulin, Francesco Visin - A guide to convolution arithmetic for deep learning (BibTeX)

Convolution animations

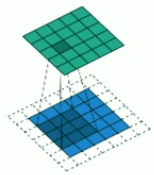
N.B.: Blue maps are inputs, and cyan maps are outputs.



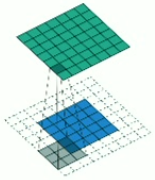
No padding, no strides



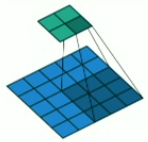
Arbitrary padding, no strides



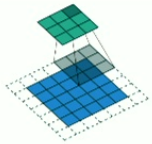
Half padding, no strides



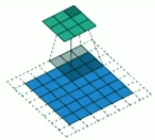
Full padding, no strides



No padding, strides



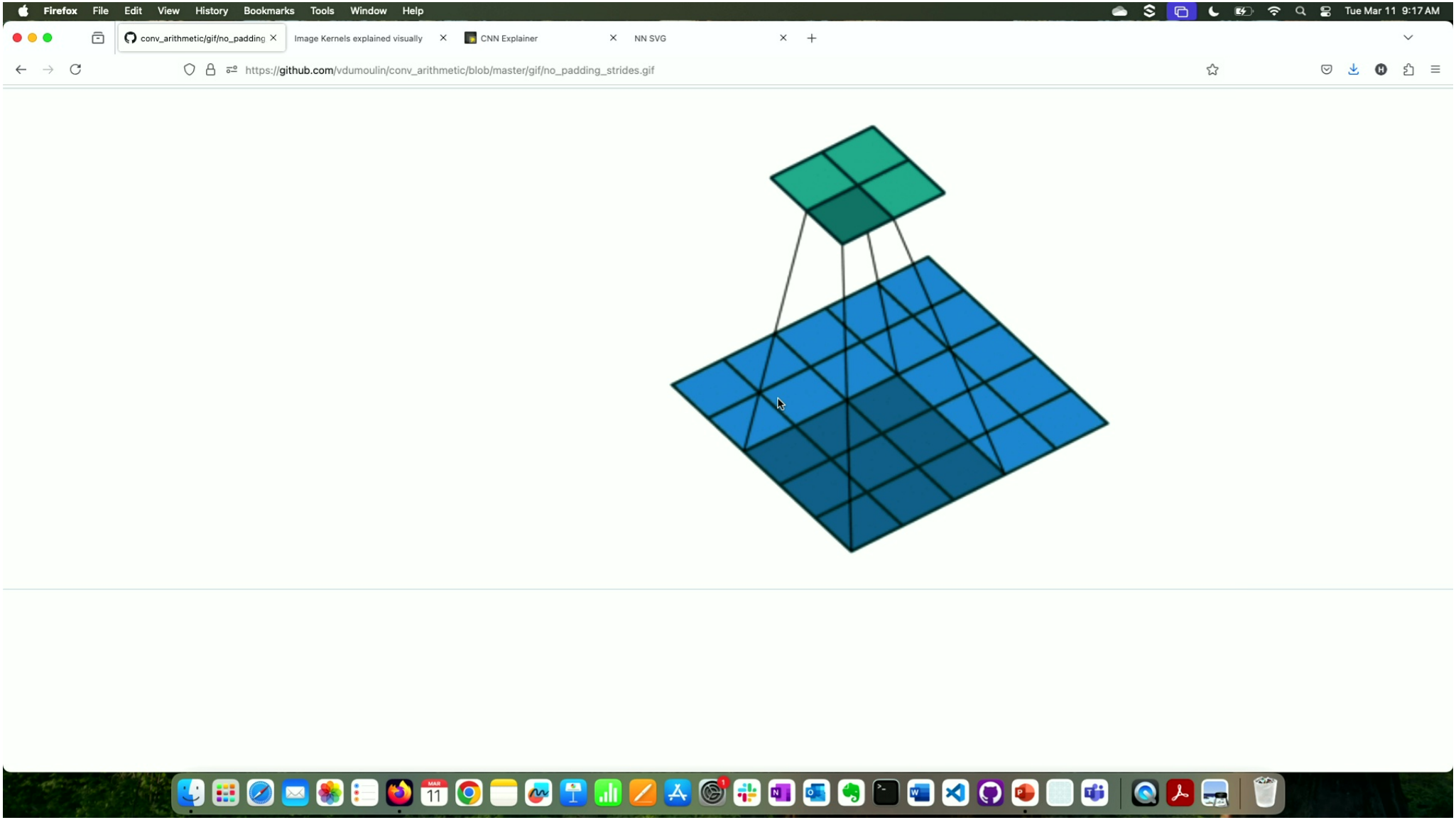
Padding, strides



Padding, strides (odd)

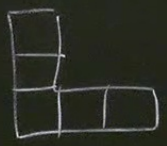
Transposed convolution animations

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

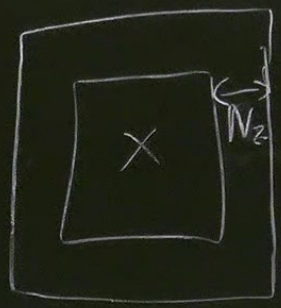

Firefox File Edit View History Bookmarks Tools Window Help
Tue Mar 11 9:17 AM
conv_arithmetic/gif/no_padding
Image Kernels explained visually
CNN Explainer
NN SVG
https://github.com/vdumoulin/conv_arithmetic/blob/master/gif/no_padding_strides.gif

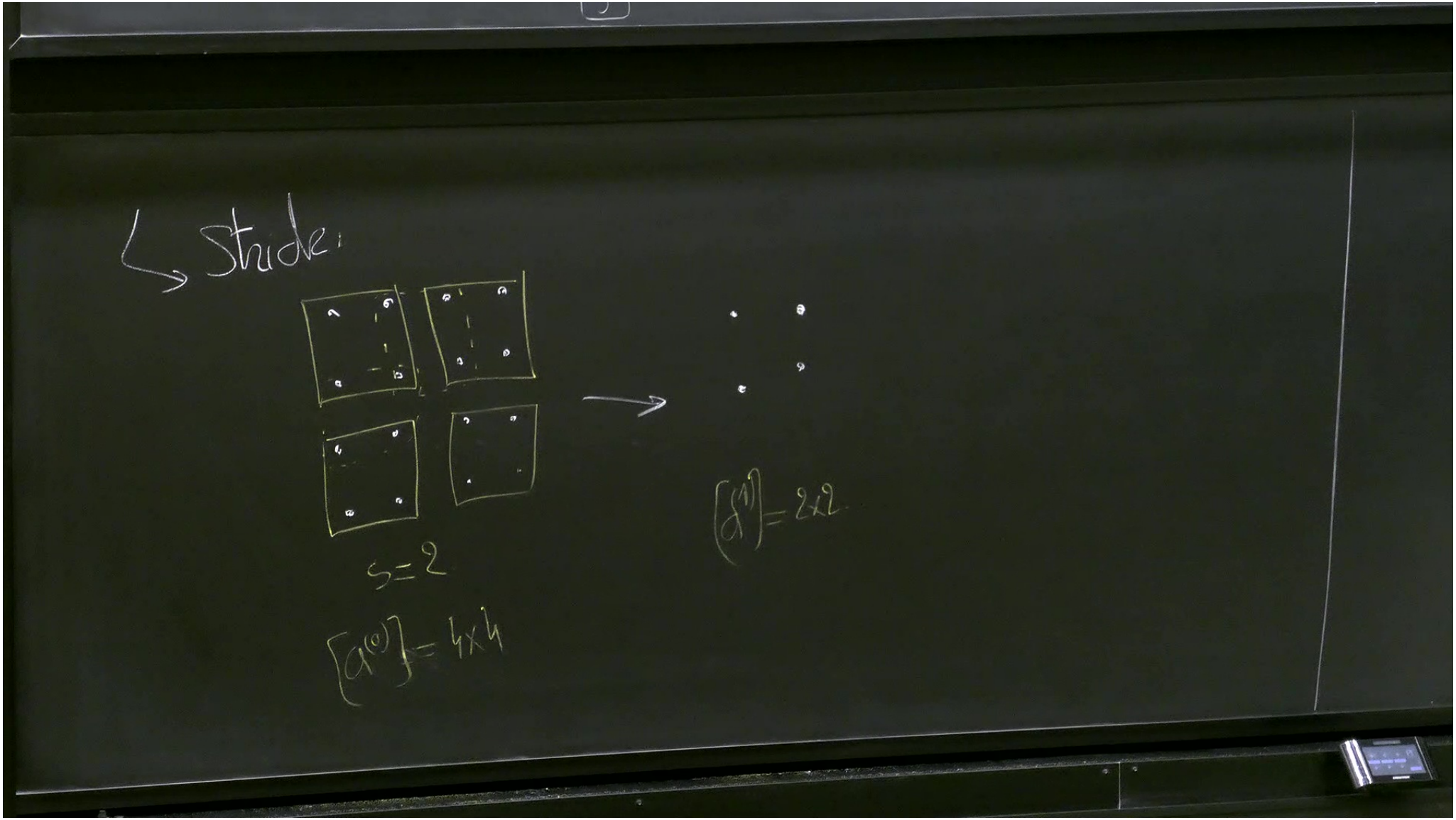
Generalizations

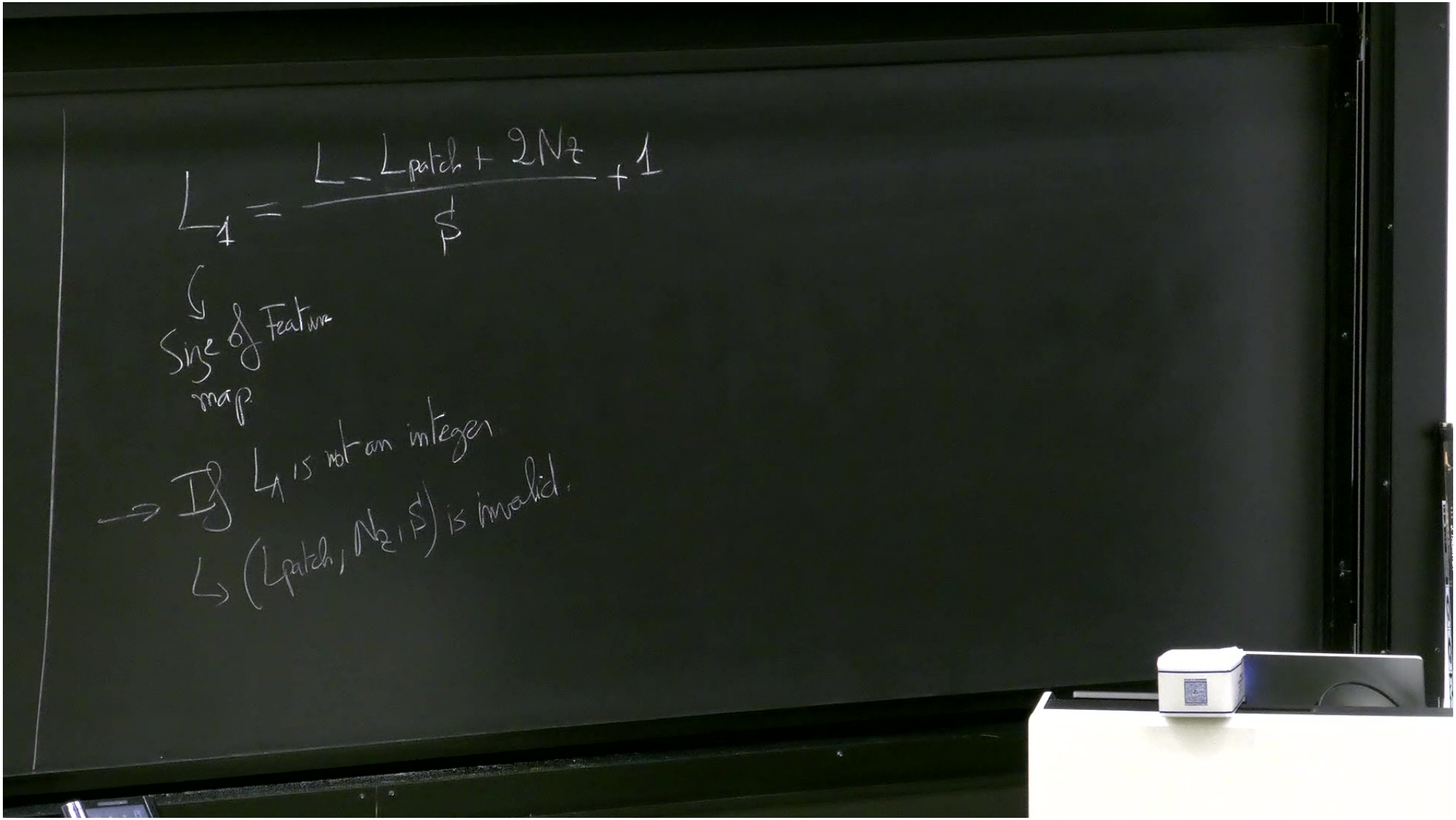
↳ Patch size: can be a square of size $L_{patch} \times L_{patch}$ or a more general shape.



↳ Padding:




↳ Stride:
S=2
[a(0)] = 4x4
[a(1)] = 2x2


$L_1 = \frac{L - L_{patch} + 2N_z}{S} + 1$
Size of Feature map
→ If L_1 is not an integer
↳ (L_{patch}, N_z, S) is invalid.

Channels

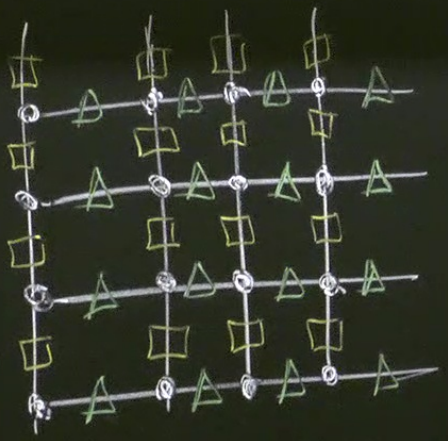
→ For colour images, c can label R, G, B channels

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

spatial indices

channel index

→ For spin configuration on lattices, c can label sublattices.

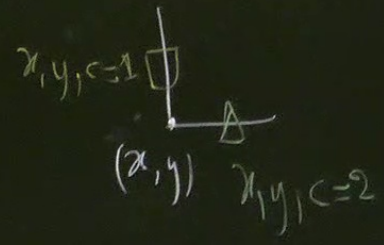


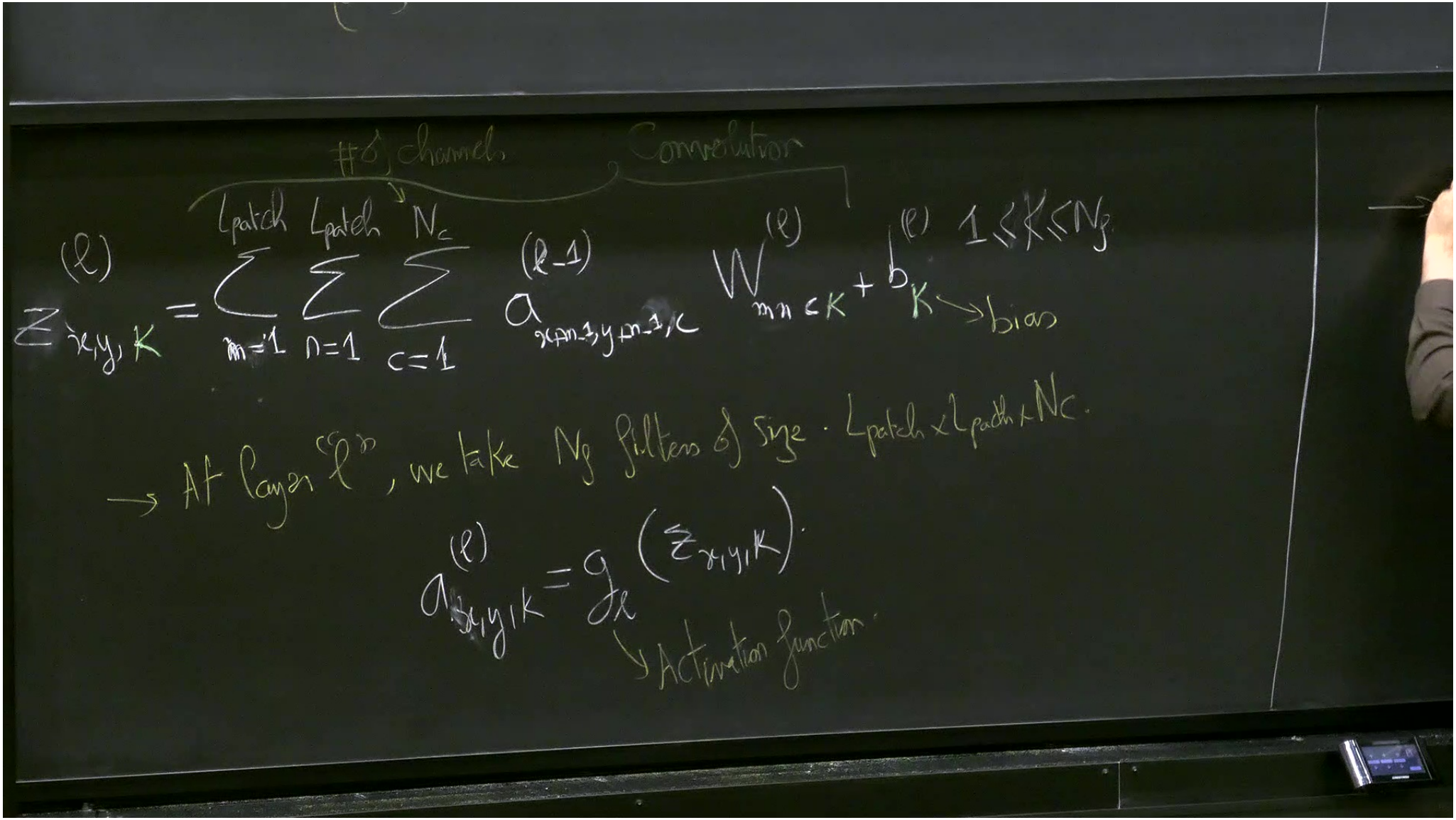
⊗ : Lattice site.

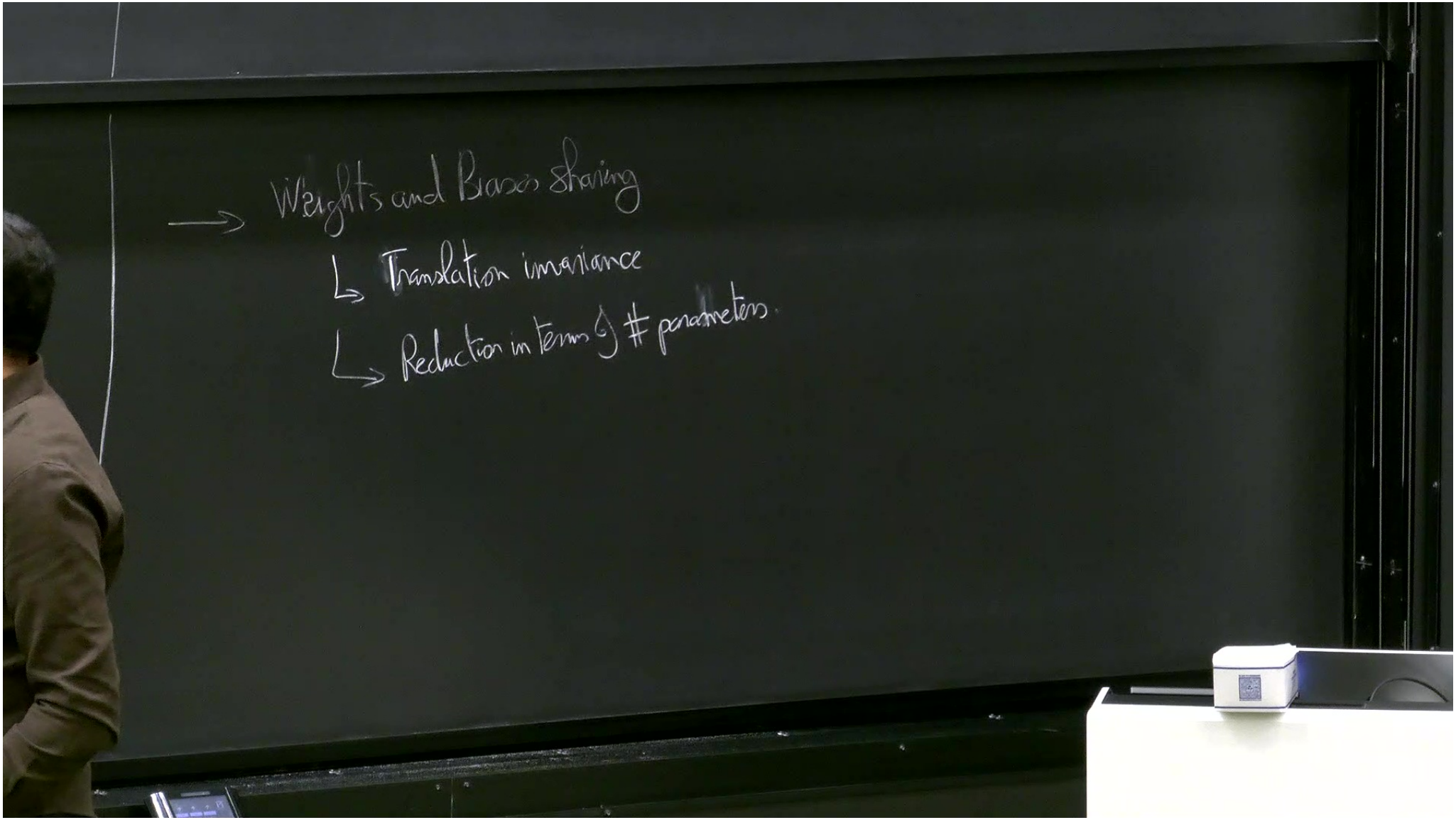
□ : sublattice 1

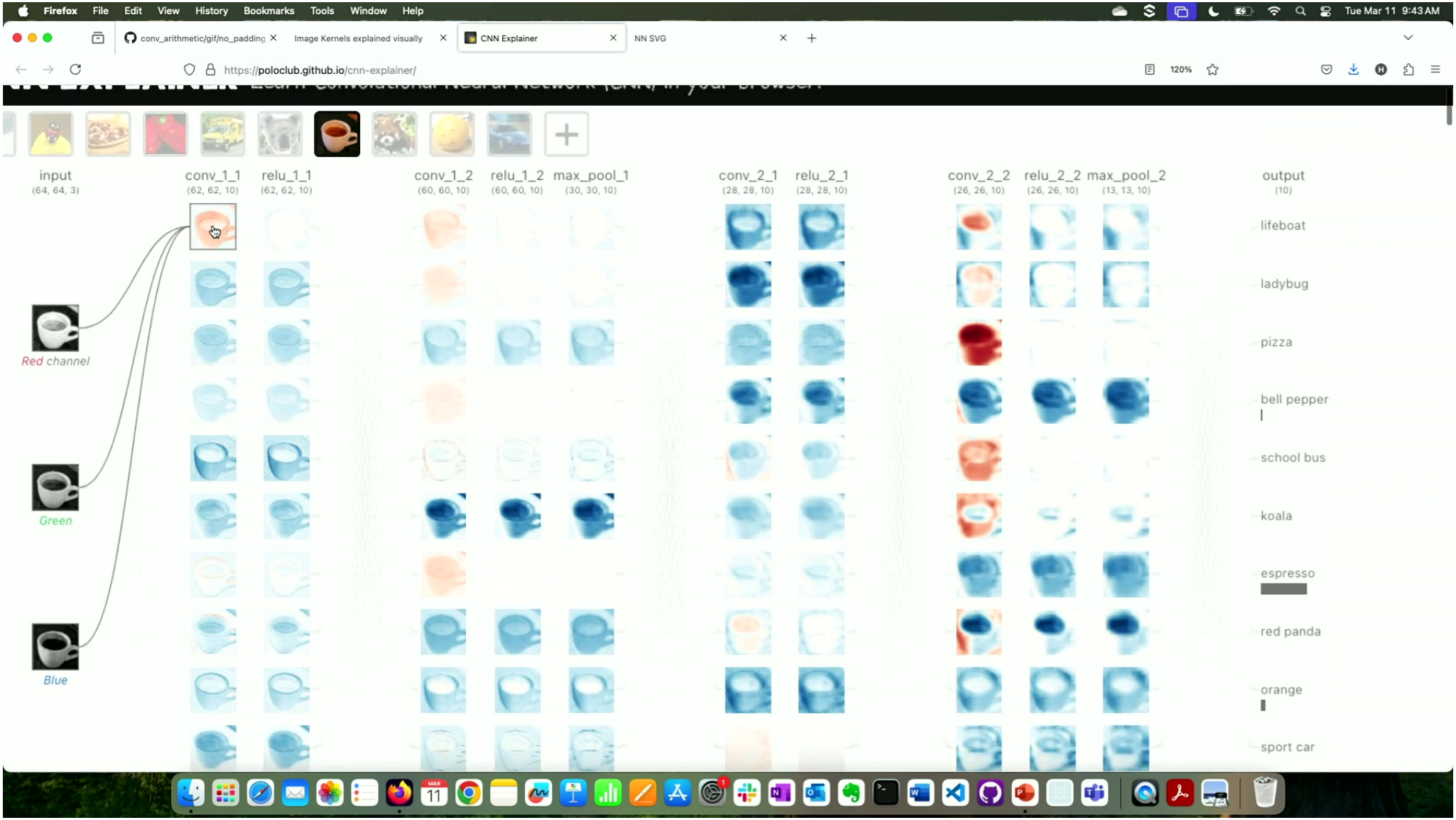
△ : sublattice 2

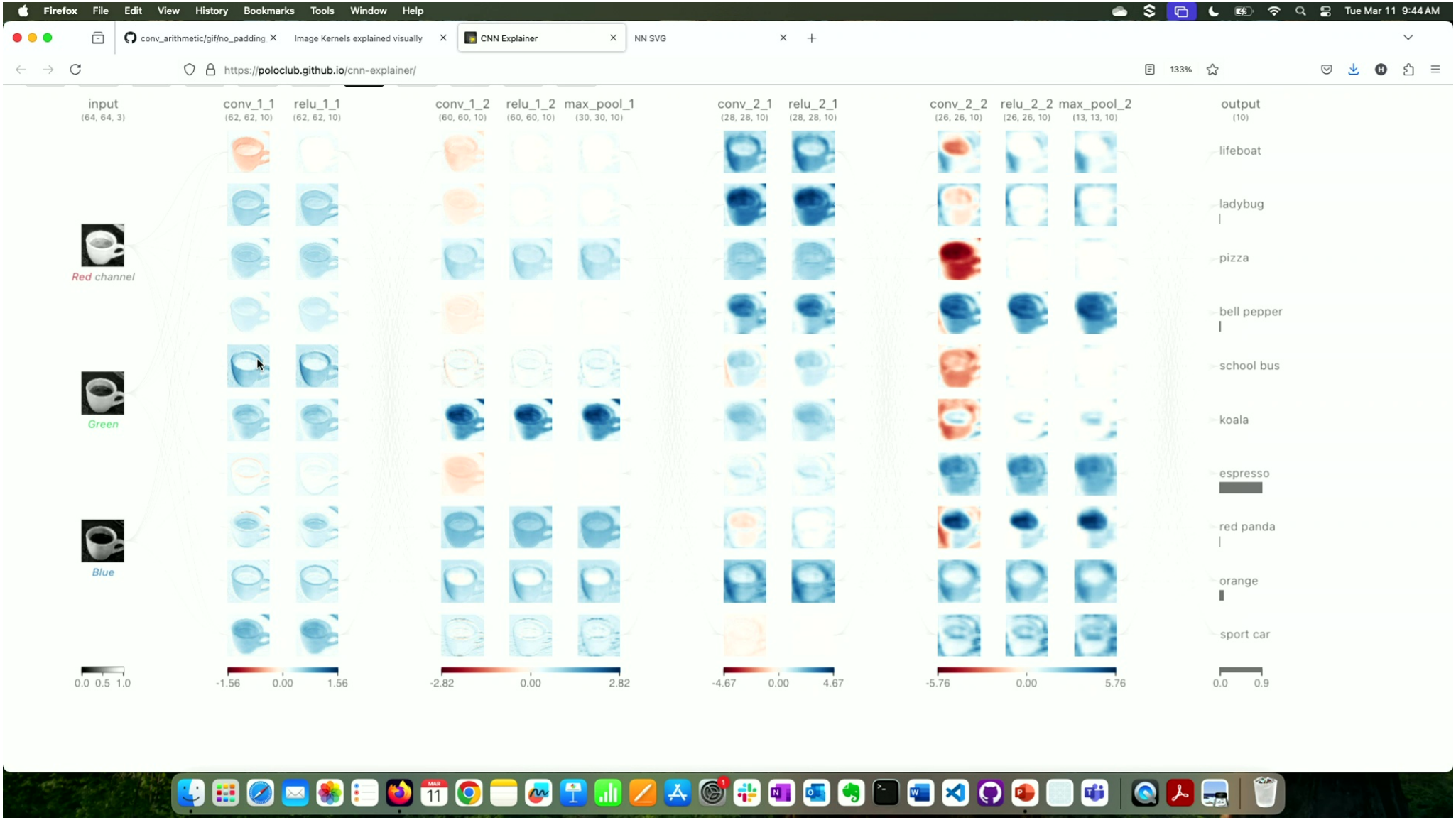
$\{q_i^{(\alpha)}\} = L \times L \times 2$

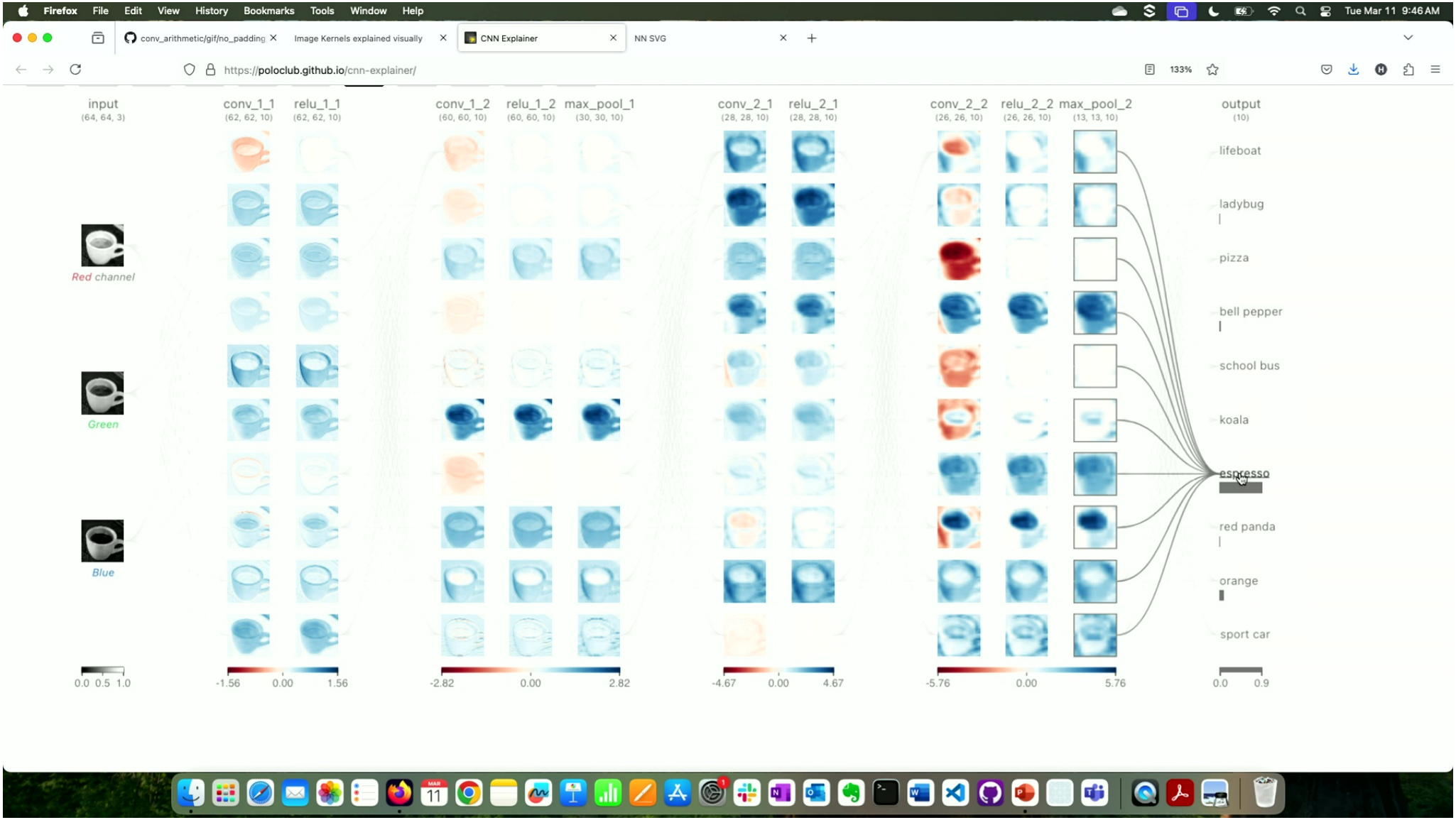


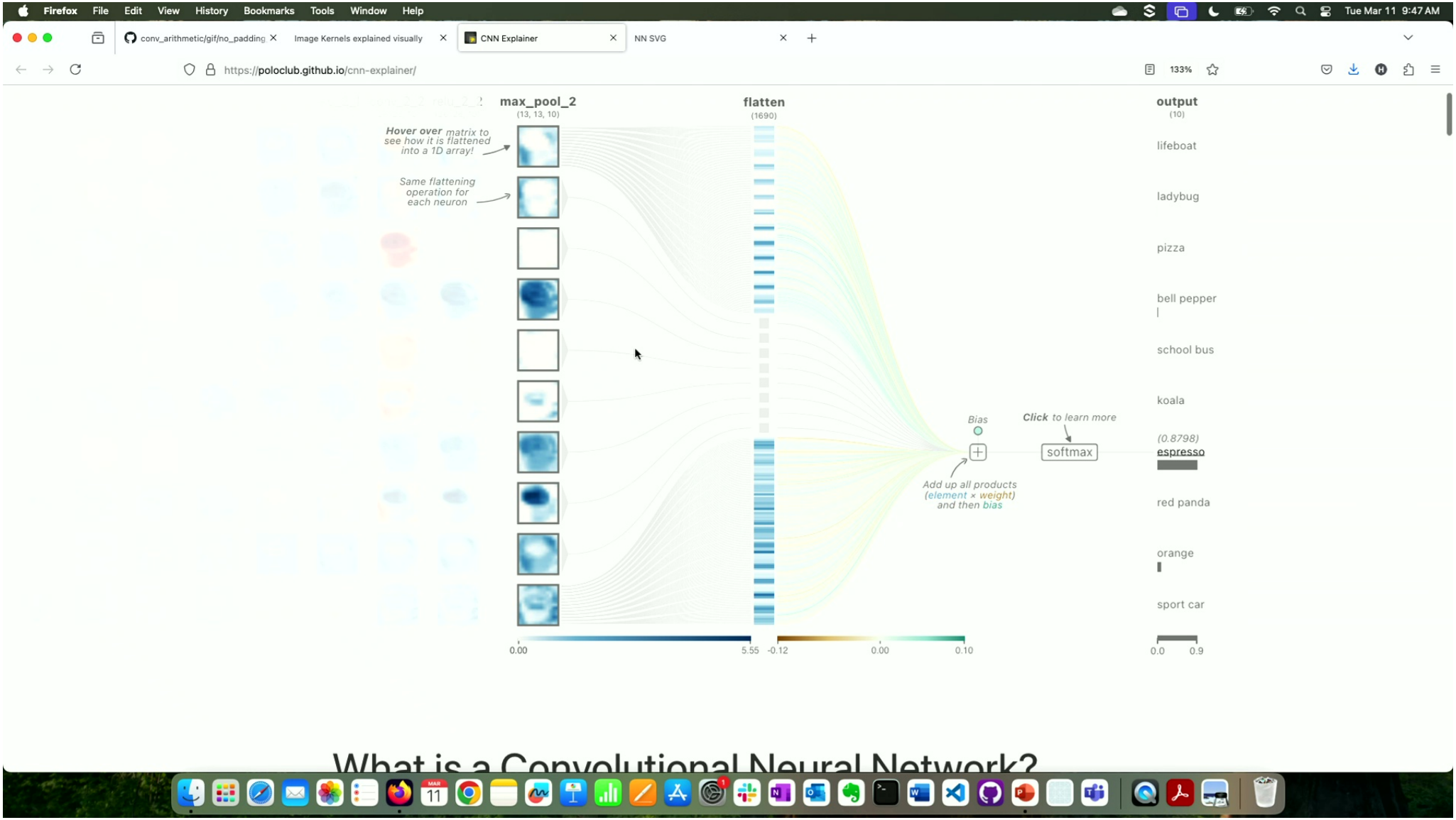

#δ channels
Convolution
$z^{(\ell)}_{x,y,K} = \sum_{m=1}^{L_{patch}} \sum_{n=1}^{L_{patch}} \sum_{c=1}^{N_c} a^{(\ell-1)}_{x+m-1,\, y+n-1,\, c}\, W^{(\ell)}_{mnck} + b^{(\ell)}_K \quad 1 \le K \le N_\ell$
b_K → bias
→ At layer "ℓ", we take N_ℓ filters of size $\cdot L_{patch} \times L_{patch} \times N_c$.
$a^{(\ell)}_{x,y,K} = g_\ell\left(z_{x,y,K}\right)$.
↳ Activation function.


Weights and Biases sharing
Translation invariance
Reduction in terms of # parameters.


Firefox File Edit View History Bookmarks Tools Window Help
Tue Mar 11 9:43 AM
conv_arithmetic/gif/no_padding
Image Kernels explained visually
CNN Explainer
NN SVG
https://poloclub.github.io/cnn-explainer/
120%
input
(64, 64, 3)
conv_1_1
(62, 62, 10)
relu_1_1
(62, 62, 10)
conv_1_2
(60, 60, 10)
relu_1_2
(60, 60, 10)
max_pool_1
(30, 30, 10)
conv_2_1
(28, 28, 10)
relu_2_1
(28, 28, 10)
conv_2_2
(26, 26, 10)
relu_2_2
(26, 26, 10)
max_pool_2
(13, 13, 10)
output
(10)
Red channel
Green
Blue
lifeboat
ladybug
pizza
bell pepper
school bus
koala
espresso
red panda
orange
sport car


Firefox File Edit View History Bookmarks Tools Window Help
Tue Mar 11 9:44 AM
conv_arithmetic/gif/no_padding
Image Kernels explained visually
CNN Explainer
NN SVG
https://poloclub.github.io/cnn-explainer/
133%
input
(64, 64, 3)
conv_1_1
(62, 62, 10)
relu_1_1
(62, 62, 10)
conv_1_2
(60, 60, 10)
relu_1_2
(60, 60, 10)
max_pool_1
(30, 30, 10)
conv_2_1
(28, 28, 10)
relu_2_1
(28, 28, 10)
conv_2_2
(26, 26, 10)
relu_2_2
(26, 26, 10)
max_pool_2
(13, 13, 10)
output
(10)
Red channel
Green
Blue
lifeboat
ladybug
pizza
bell pepper
school bus
koala
espresso
red panda
orange
sport car
0.0 0.5 1.0
-1.56 0.00 1.56
-2.82 0.00 2.82
-4.67 0.00 4.67
-5.76 0.00 5.76
0.0 0.9


Firefox File Edit View History Bookmarks Tools Window Help
Tue Mar 11 9:46 AM
conv_arithmetic/gif/no_padding
Image Kernels explained visually
CNN Explainer
NN SVG
https://poloclub.github.io/cnn-explainer/
133%
input
(64, 64, 3)
conv_1_1
(62, 62, 10)
relu_1_1
(62, 62, 10)
conv_1_2
(60, 60, 10)
relu_1_2
(60, 60, 10)
max_pool_1
(30, 30, 10)
conv_2_1
(28, 28, 10)
relu_2_1
(28, 28, 10)
conv_2_2
(26, 26, 10)
relu_2_2
(26, 26, 10)
max_pool_2
(13, 13, 10)
output
(10)
Red channel
Green
Blue
lifeboat
ladybug
pizza
bell pepper
school bus
koala
espresso
red panda
orange
sport car
0.0 0.5 1.0
-1.56 0.00 1.56
-2.82 0.00 2.82
-4.67 0.00 4.67
-5.76 0.00 5.76
0.0 0.9

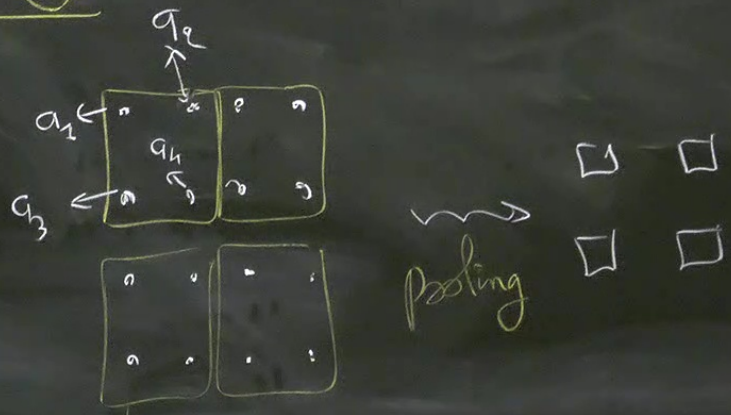

Firefox File Edit View History Bookmarks Tools Window Help
Tue Mar 11 9:47 AM
conv_arithmetic/gif/no_padding
Image Kernels explained visually
CNN Explainer
NN SVG
https://poloclub.github.io/cnn-explainer/
133%
max_pool_2
(13, 13, 10)
flatten
(1690)
output
(10)
Hover over matrix to see how it is flattened into a 1D array!
Same flattening operation for each neuron
lifeboat
ladybug
pizza
bell pepper
school bus
koala
(0.8798)
espresso
red panda
orange
sport car
Bias
Click to learn more
softmax
Add up all products (element × weight) and then bias
0.00
5.55
-0.12
0.00
0.10
0.0
0.9
What is a Convolutional Neural Network?

Lecture 7: Convolutional Neural Networks

Today: "CNNs"

- Local receptive fields
  - Weight and bias sharing.
- Channels
- Coarse graining through pooling.

Pooling:

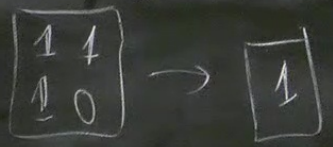


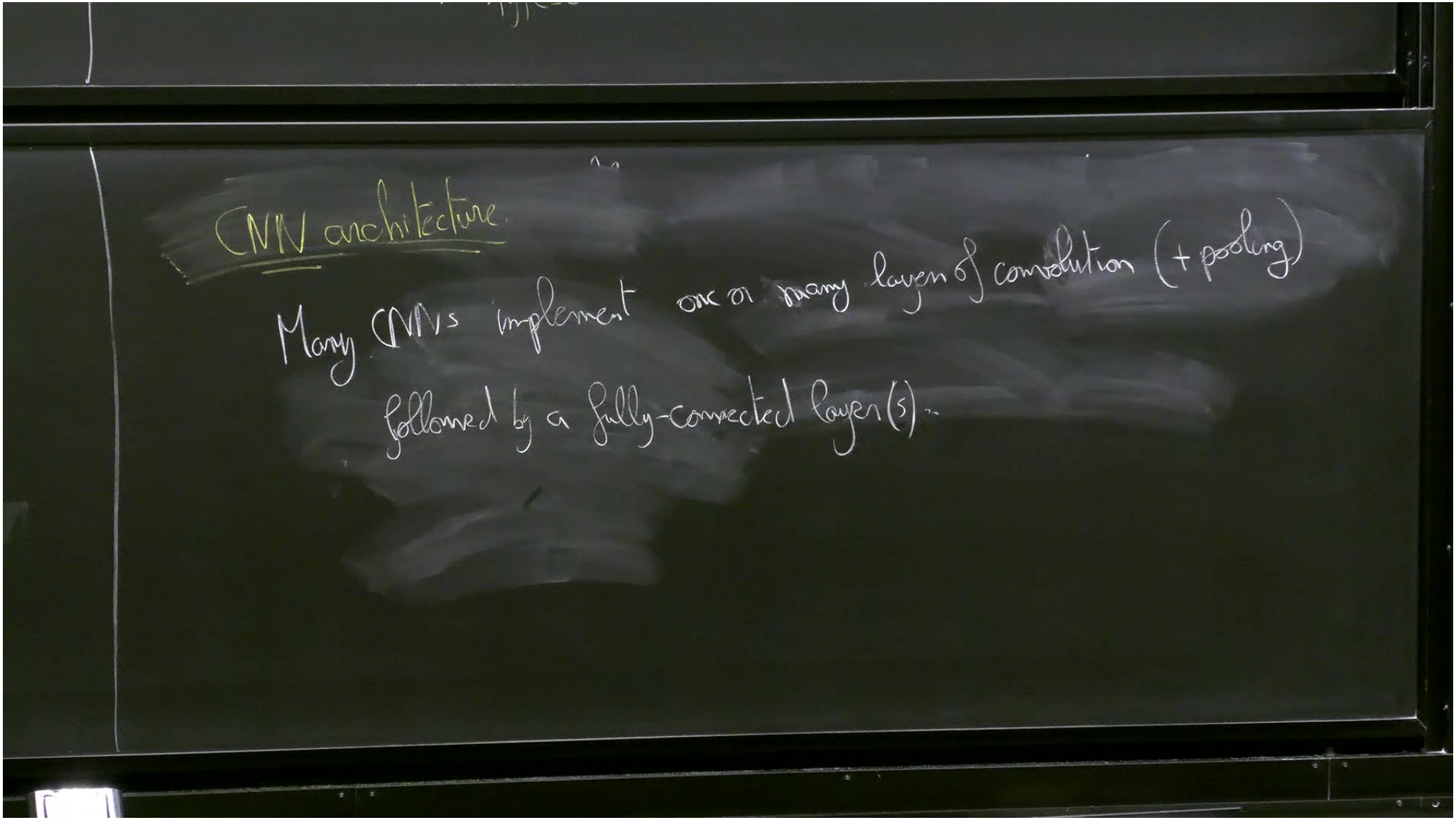
where □ can be:

→ $\max(a_1, a_2, a_3, a_4)$.

→ $\frac{a_1 + a_2 + a_3 + a_4}{4}$

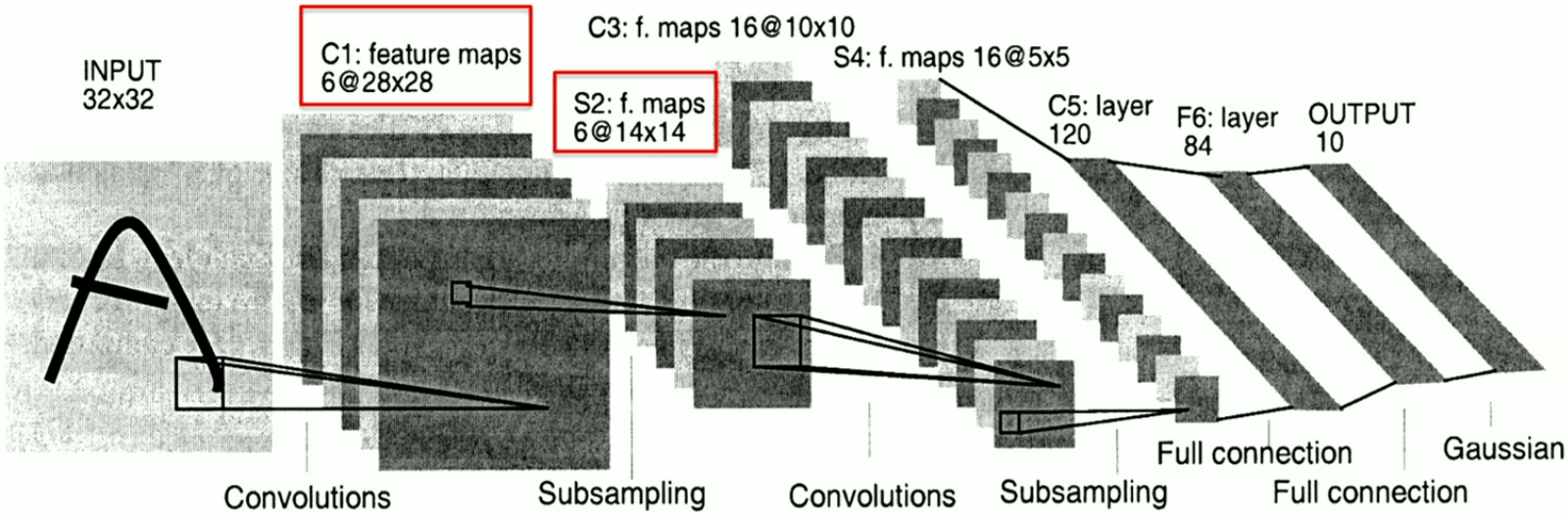
→ Majority rule.




CNN architecture
Many CNNs implement one or many layers of convolution (+ pooling)
followed by a fully-connected layer(s)...

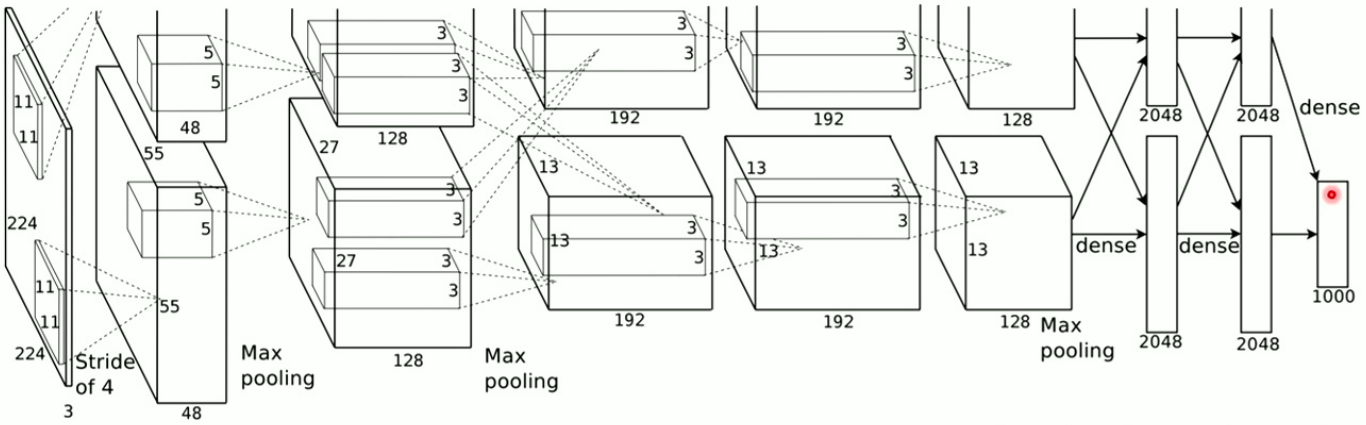
LeNet

- Here's the LeNet architecture, which was applied to handwritten digit recognition on MNIST in 1998:



AlexNet

- AlexNet, essentially like LeNet but scaled up in every way (more layers, more units, more connections, etc.):



(Krizhevsky et al., 2012)

- AlexNet's performance on the ImageNet competition got everyone excited about deep learning in 2012.

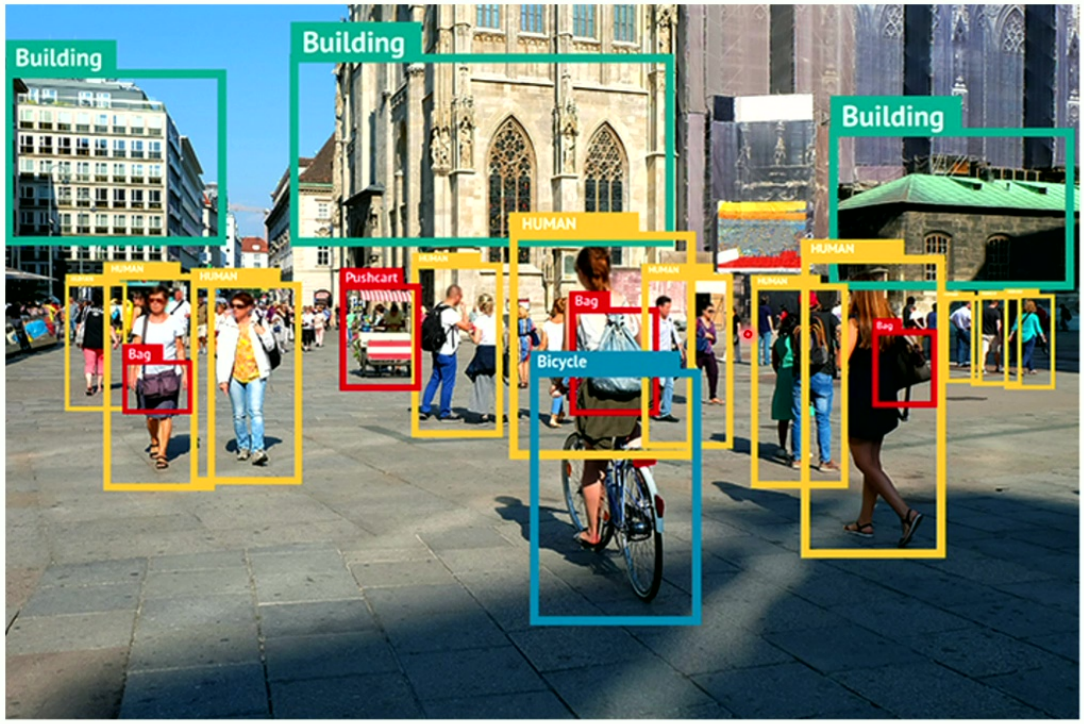
Classification

- ImageNet results over the years. There are 1000 classes. Note that errors are top-5 errors (the network gets to make 5 guesses).

Year	Model	Top-5 error
2010	Hand-designed descriptors + SVM	28.2%
2011	Compressed Fisher Vectors + SVM	25.8%
2012	AlexNet	16.4%
2013	a variant of AlexNet	11.7%
2014	GoogLeNet	6.6%
2015	deep residual nets	4.5%

- Human-level performance is around 5.1%.
- They stopped running the object recognition competition because the performance is already so good.

Applications of CNNs



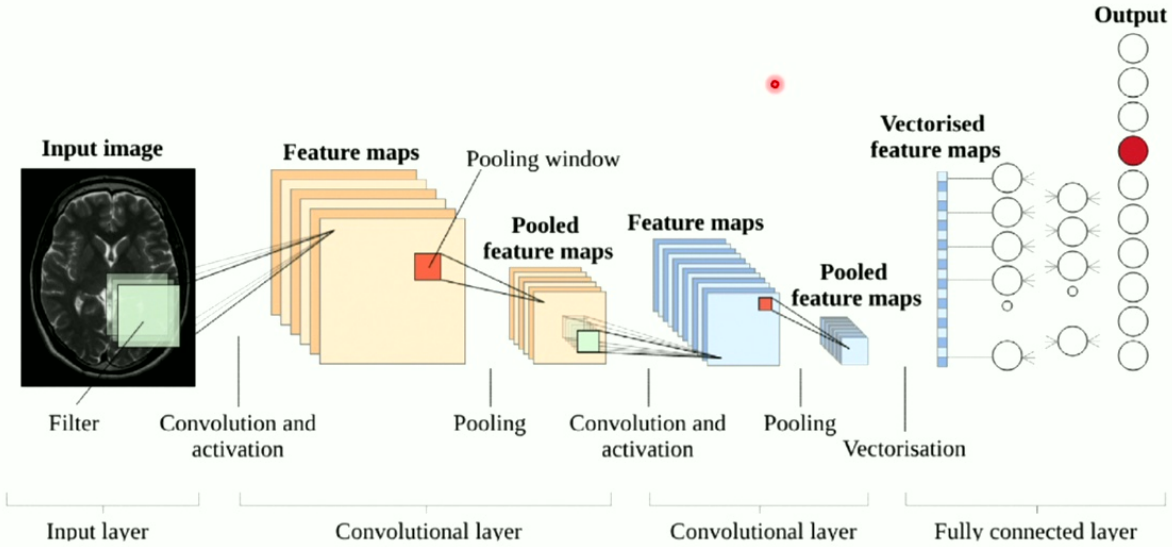
Credit: www.datasciencecentral.com

Applications of CNNs

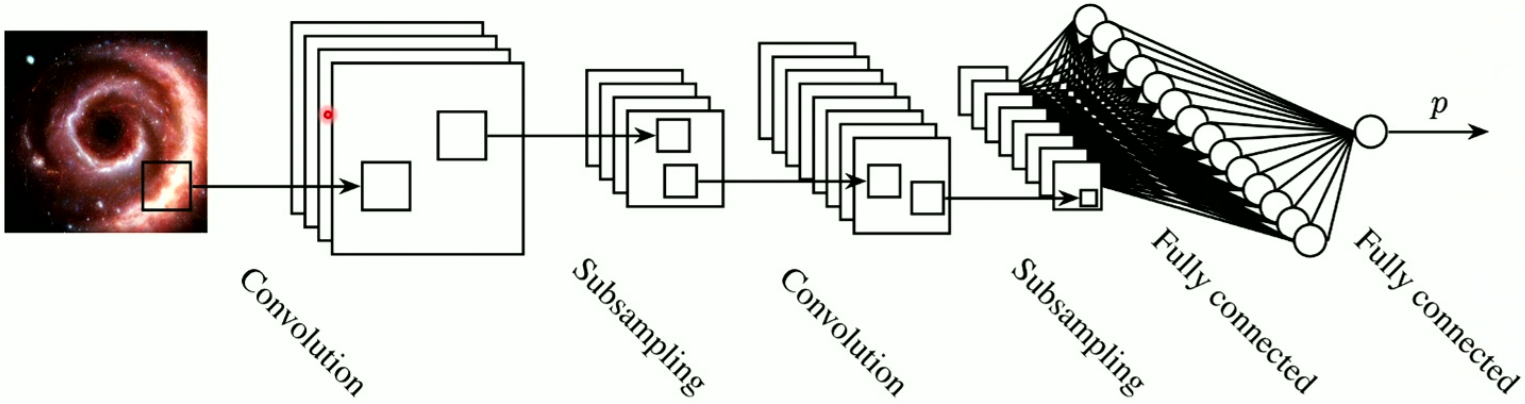
Review

An overview of deep learning in medical imaging focusing on MRI

Alexander Selvikvåg Lundervold [a b], Arvid Lundervold [a c d]



Applications of CNNs



Astronomia ex machina: a history, primer, and outlook on neural networks in astronomy

Michael J. Smith and James E. Geach

Department of Physics, Astronomy & Mathematics, School of Physics, Engineering and Computer Science, University of Hertfordshire, Hatfield, AL10 9AB

Applications of CNNs

Two-dimensional frustrated J_1-J_2 model studied with neural network quantum states

Kenny Choo,[1] Titus Neupert,[1] and Giuseppe Carleo[2]
[1]*Department of Physics, University of Zurich, Winterthurerstrasse 190, 8057 Zurich, Switzerland*
[2]*Center for Computational Quantum Physics, Flatiron Institute, 162 5th Avenue, New York, New York 10010, USA*

(Received 23 March 2019; revised manuscript received 17 July 2019; published 11 September 2019)

The use of artificial neural networks to represent quantum wave functions has recently attracted interest as a way to solve complex many-body problems. The potential of these variational parametrizations has been supported by analytical and numerical evidence in controlled benchmarks. While approaching the end of the early research phase in this field, it becomes increasingly important to show how neural-network states perform for models and physical problems that constitute a clear open challenge for other many-body computational methods. In this paper, we start addressing this aspect, concentrating on a presently unsolved model describing two-dimensional frustrated magnets. Using a fully convolutional neural network model as a variational ansätz, we study the frustrated spin-1/2 J_1-J_2 Heisenberg model on the square lattice. We demonstrate that the resulting predictions for both ground-state energies and properties are competitive with, and often improve upon, existing state-of-the-art methods. In a relatively small region in the parameter space, corresponding to the maximally frustrated regime, our ansätz exhibits comparatively good but not the best performance. The gap between the complexity of the models adopted here and those routinely adopted in deep-learning applications is, however, still substantial, such that further improvements in future generations of neural-network quantum states are likely to be expected.