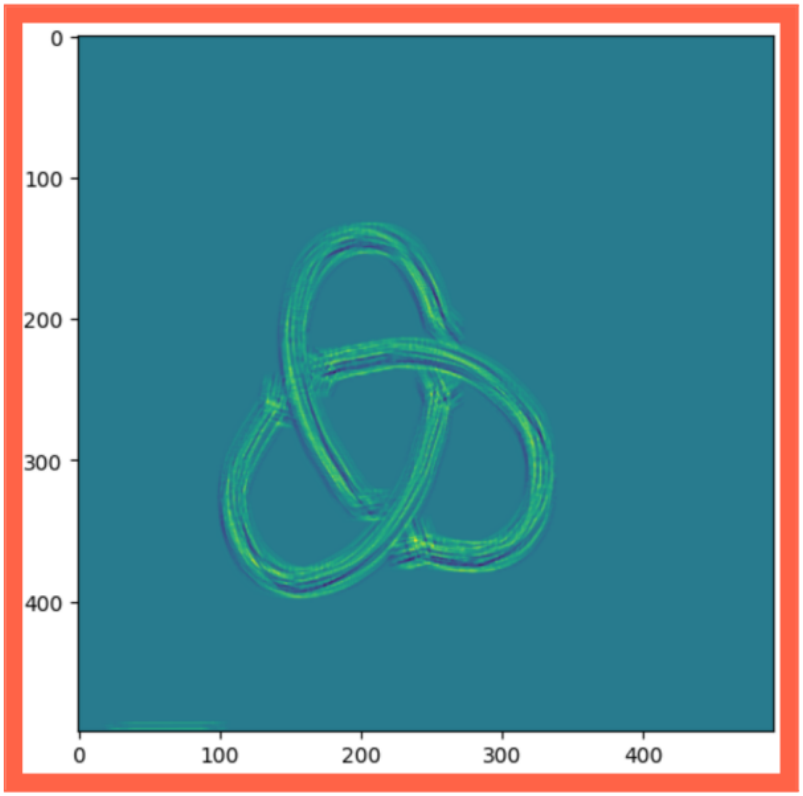
Picture recognition, knots and quantum algebra

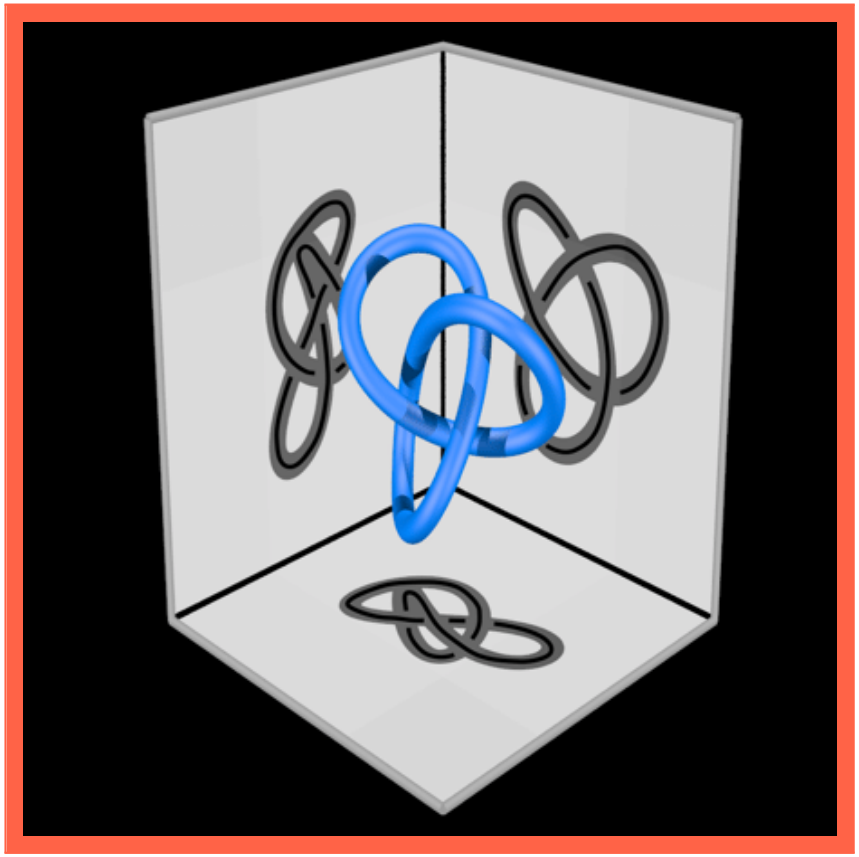
Or: Knot detector

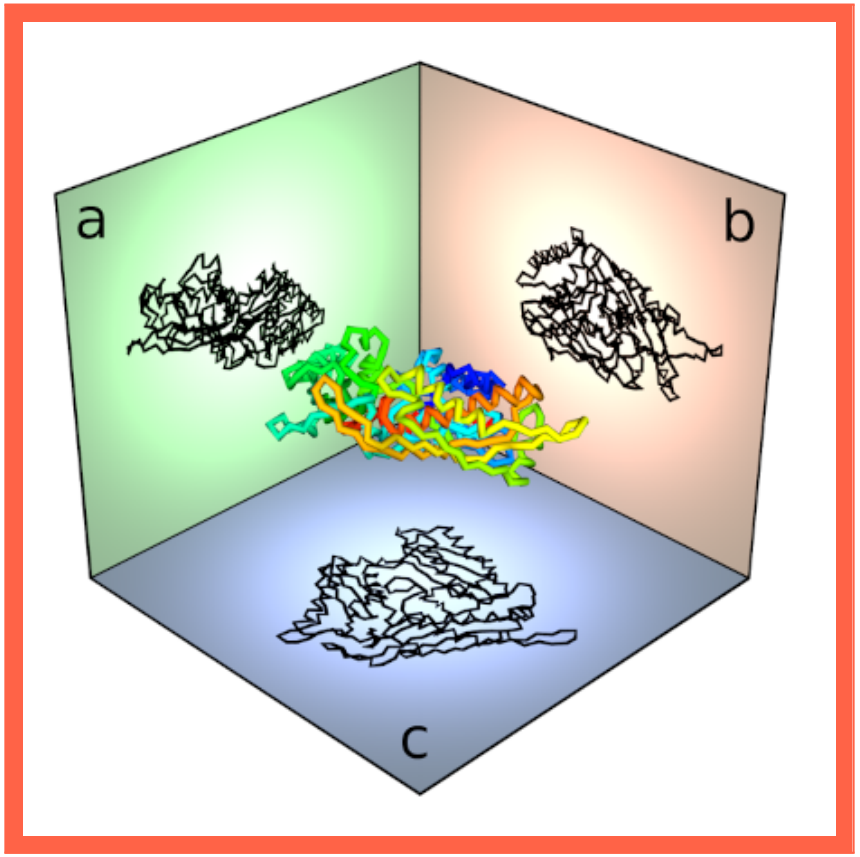
~~Accept~~Change what you cannot ~~change~~accept



I report on work of Dranowski–Kabkov–Melamud (via Yulia's dream)

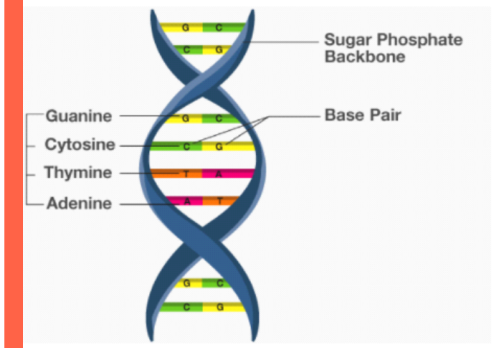
Motivation: detect knots via images





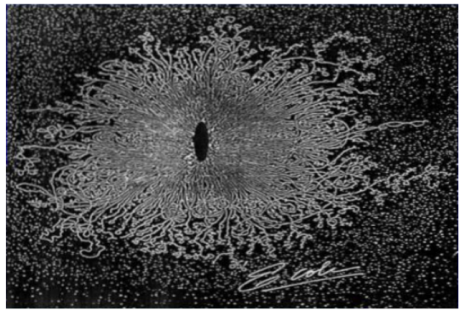
- Knot = closed string (a circle S^1) in three spaces; link = multiple components
- Knots are studied by projections to the plane Shadows
- In math knot theory started in the early 20th century

Motivation: detect knots via images

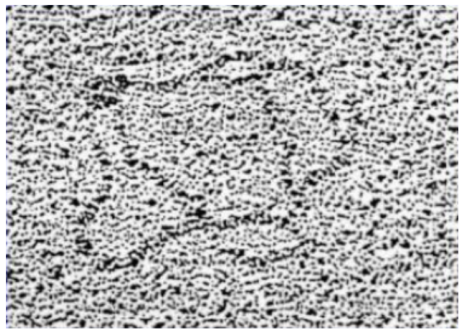


DNA double helix

DNA supercoil



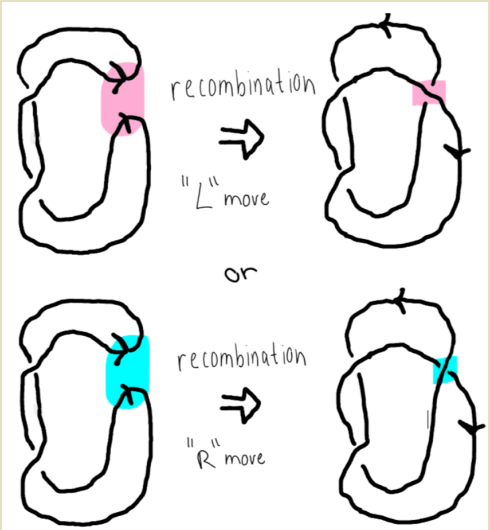
DNA is crowded in the cell



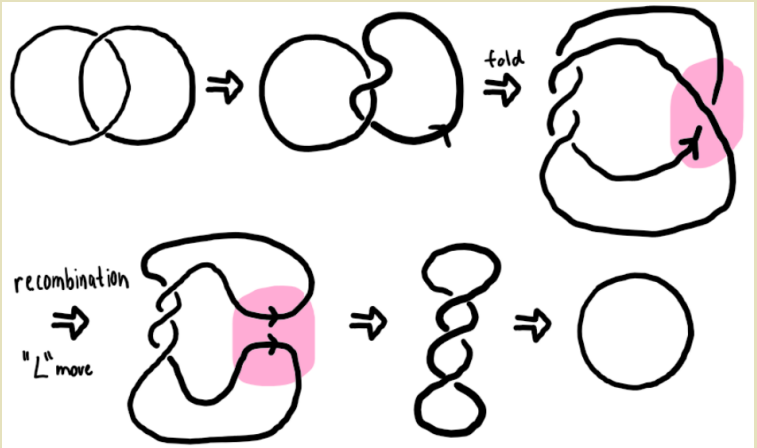
Enzyme bound to DNA

- **Outside math** knot theory started ∼1980
- **Theorem** Long strings in small boxes get knotted
- **Example** DNA is a long string in a tiny box (cell) ⇒ its often knotted

Enzymes cut DNA knots

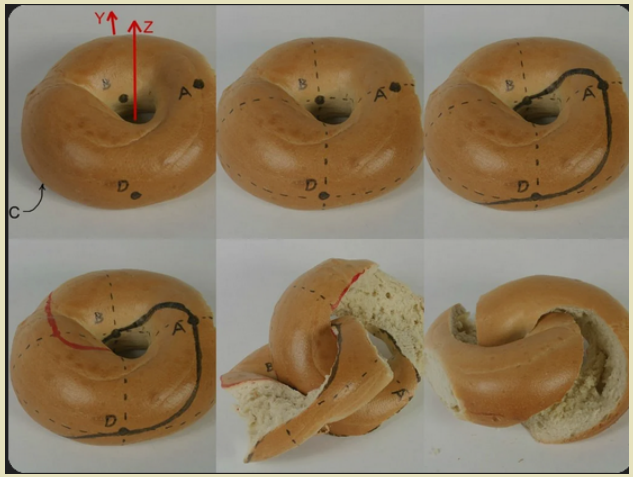


Site specific DNA recombination



Site specific DNA recombination example

DNA knots need to be undone (the body uses the moves above) before the DNA can multiply; here is why (the two parts get knotted):



Motivat

It is crucial to identify the DNA knot

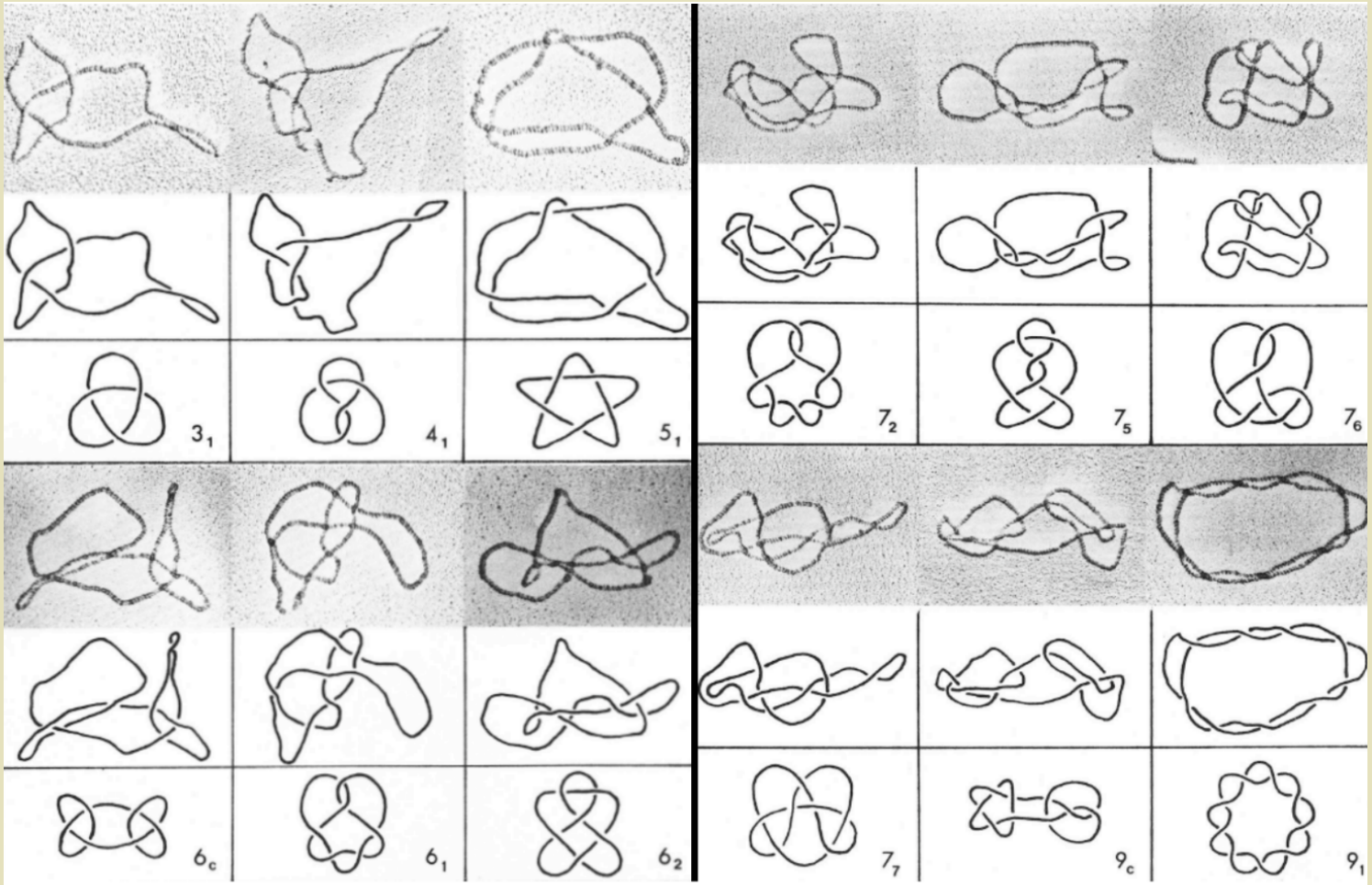


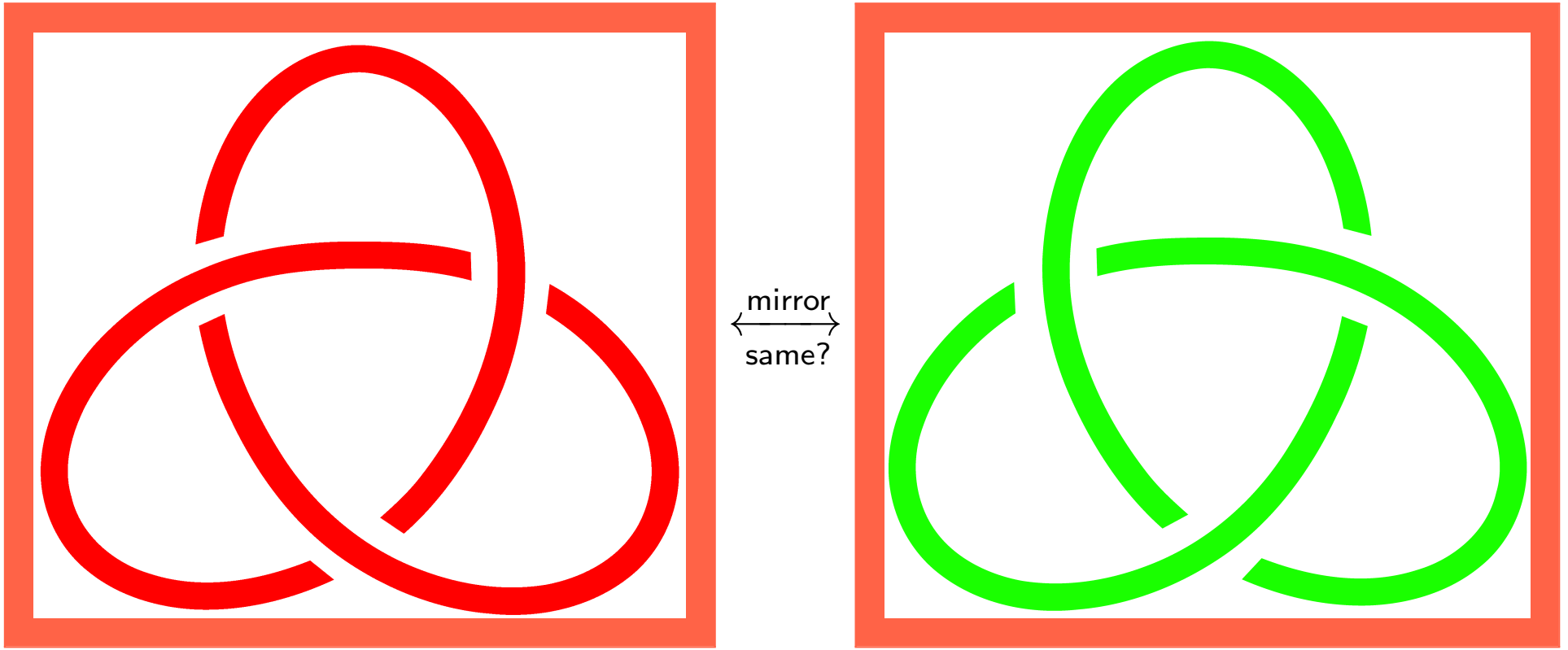
Figure: Duplex DNA knots produced by Escherichia coli Topoisomerase, via *Journal of Biological Chemistry* 1985

For example, difficult DNA knots prevent DNA from reproducing and you want that for things like viruses, or cancer cells

- Ou
- Th
- Ex

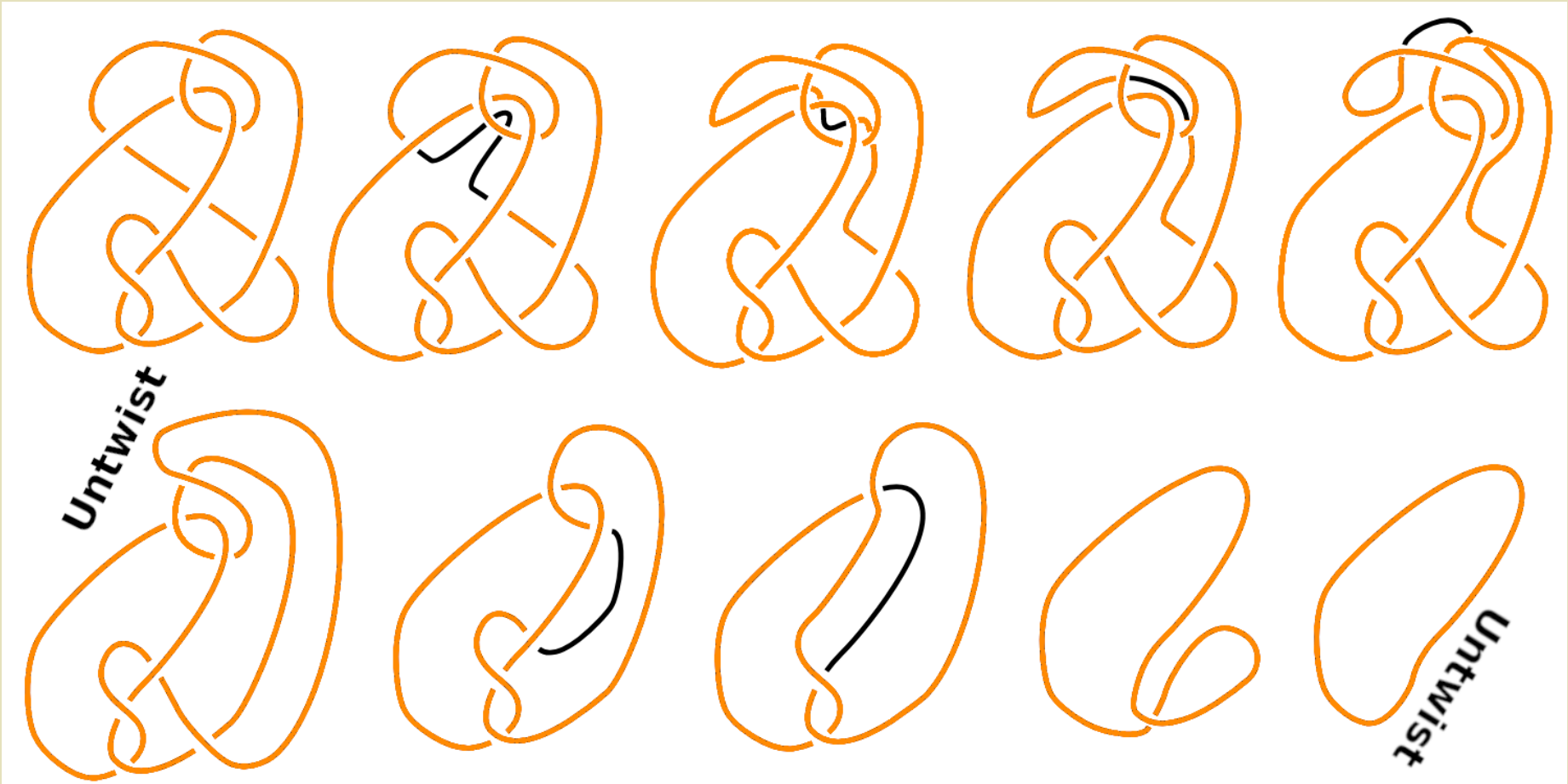
ed

Motivation: detect knots via images



- Problem Deciding whether two knot projections are the same knot is difficult
- Task Find an invariant. Sounds easy? Well, most knot invariants are pretty bad...so: find a 'good' knot invariant
- Example There was no knot invariant before 1980 that can distinguish the above knots

Even the unknotting problem is tricky

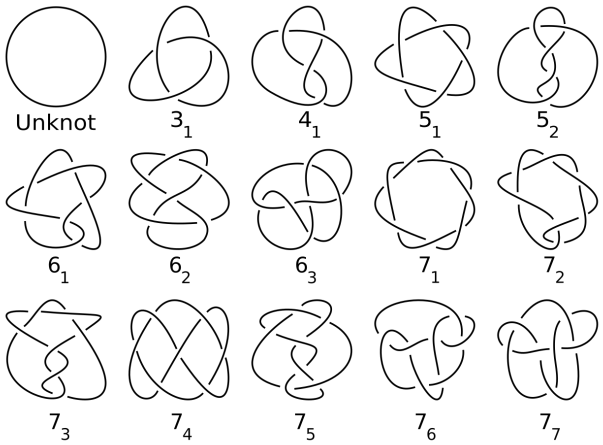


In general, the "standard invariants from algebraic topology"
(homology and friends)
are really not good for knots

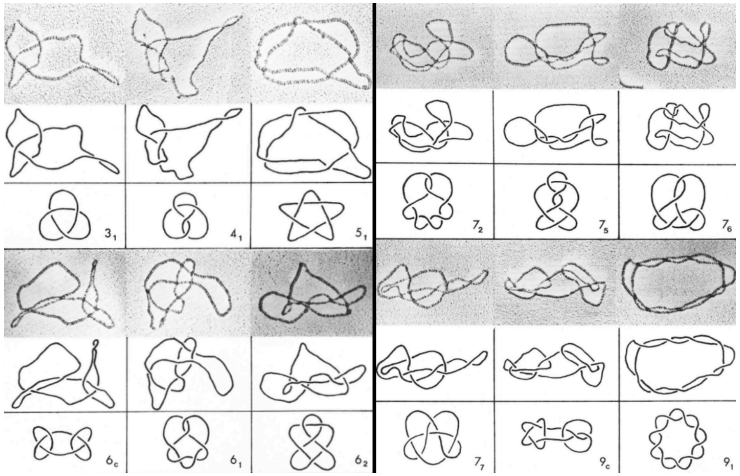
above knots

Motivation: detect knots via images

Math knots:



DNA knots:



- Knots are hard for humans/machines to recognize visually
- Developing an automated method to extract data from knot images has impact beyond mathematics (DNA, protein folding, polymer science, ...)
- Goal Develop a CNN model for knot recognition
- Aims, explained together with why this is feasible as we go:
  - Identify the number of crossings. **(Done!)**
  - Produce planar diagram (PD) presentations for computation of invariants, e.g., Jones polynomial. **(On it!)**

Motivation: detect knots via images

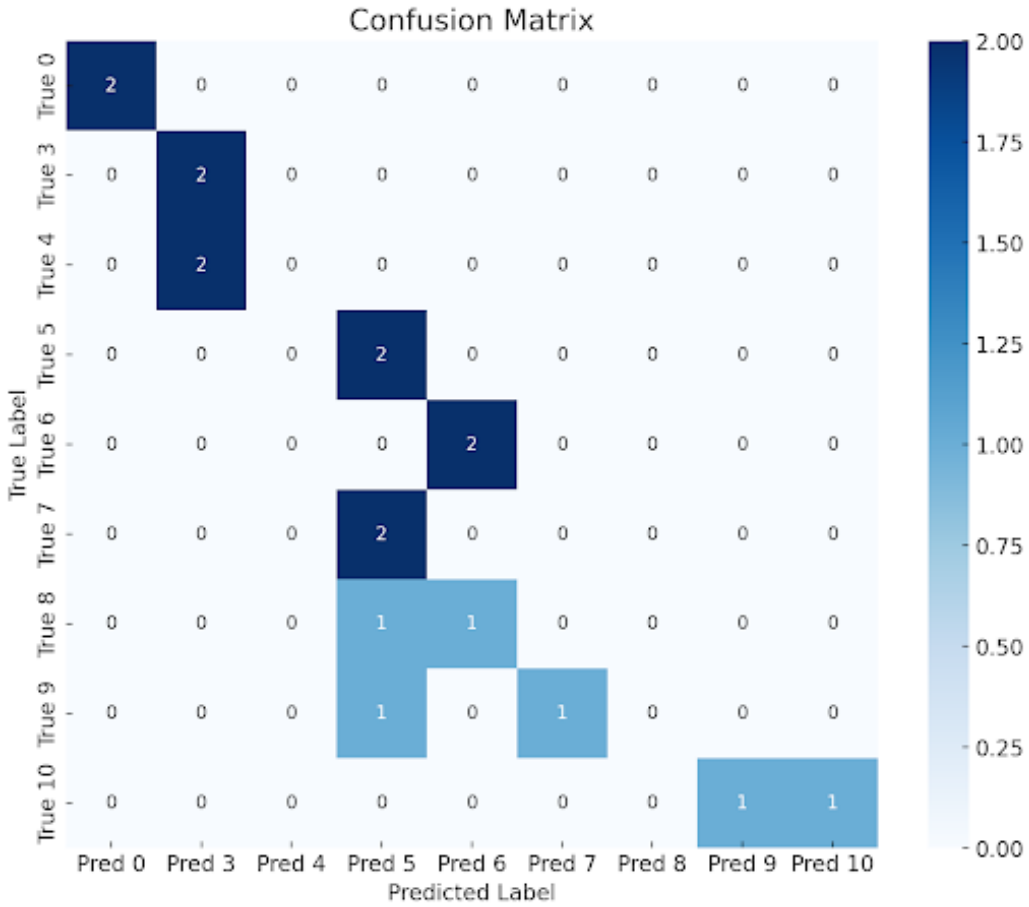


Figure: Confusion matrix of o4-mini results. Average accuracy is 50%

- ChatGPT did poorly
- Although the test set was small, there was no way to even count the number of crossings with ChatGPT

Motivation: detect knots via images

Reminder: A confusion matrix is a generalization of:

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

		Predicted condition	
	Total 8 + 4 = 12	Cancer 7	Non-cancer 5
Actual condition	Cancer 8	6	2
	Non-cancer 4	1	3

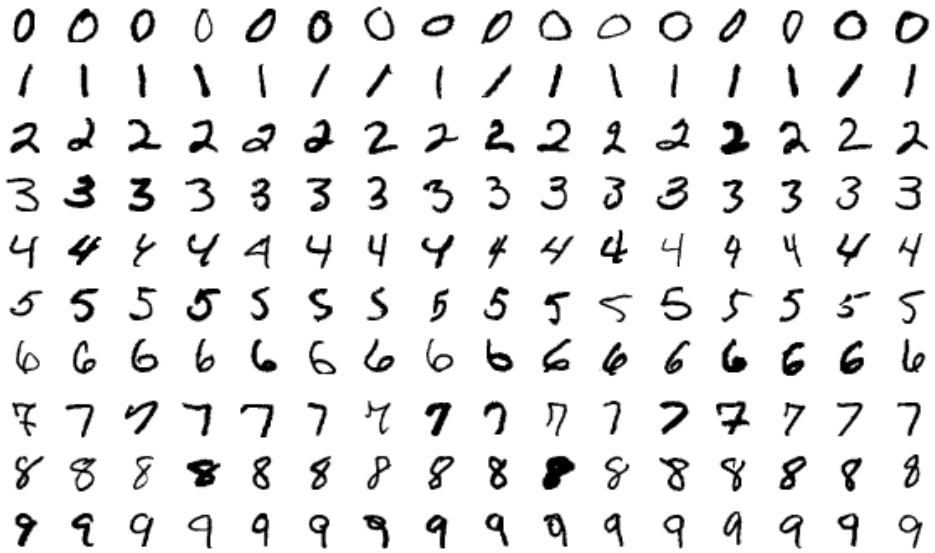
The more diagonal this matrix is the better

Figure: is 50%

- ChatGPT c
- Although t

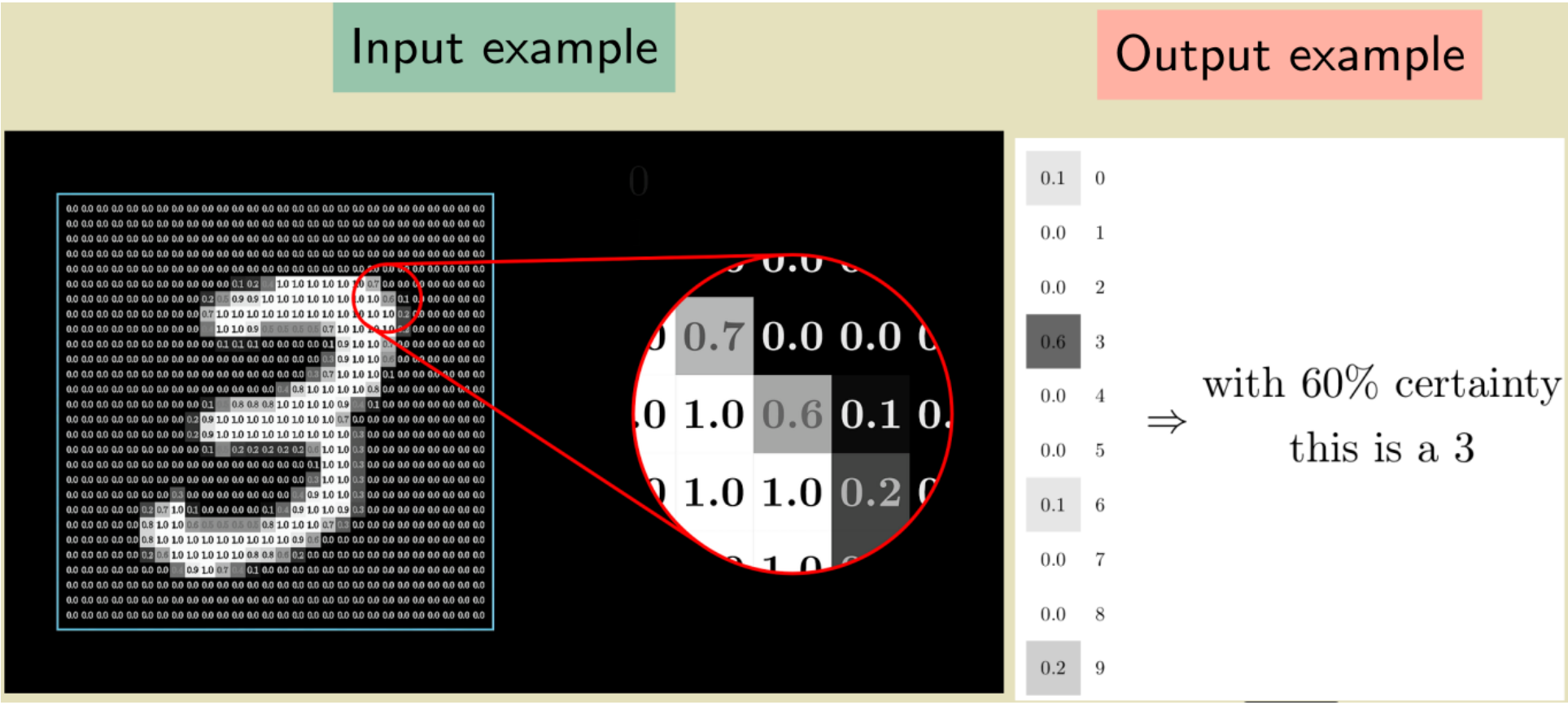
no way to even count the number of crossings with ChatGPT

Preparation: neural networks (NN) and friends



- Task: Identify handwritten digits
- We can see this as a function in the following way:
  (a) Convert the pictures into grayscale values, e.g. 28×28 grid of numbers
  (b) Flatten this into a vector, e.g. $28 \times 28 \mapsto$ a vector with $28^2 = 784$ entries
  (c) The output is a vector with 10 entries
  (d) We thus have a function $\mathbb{R}^{784} \to \mathbb{R}^{10}$

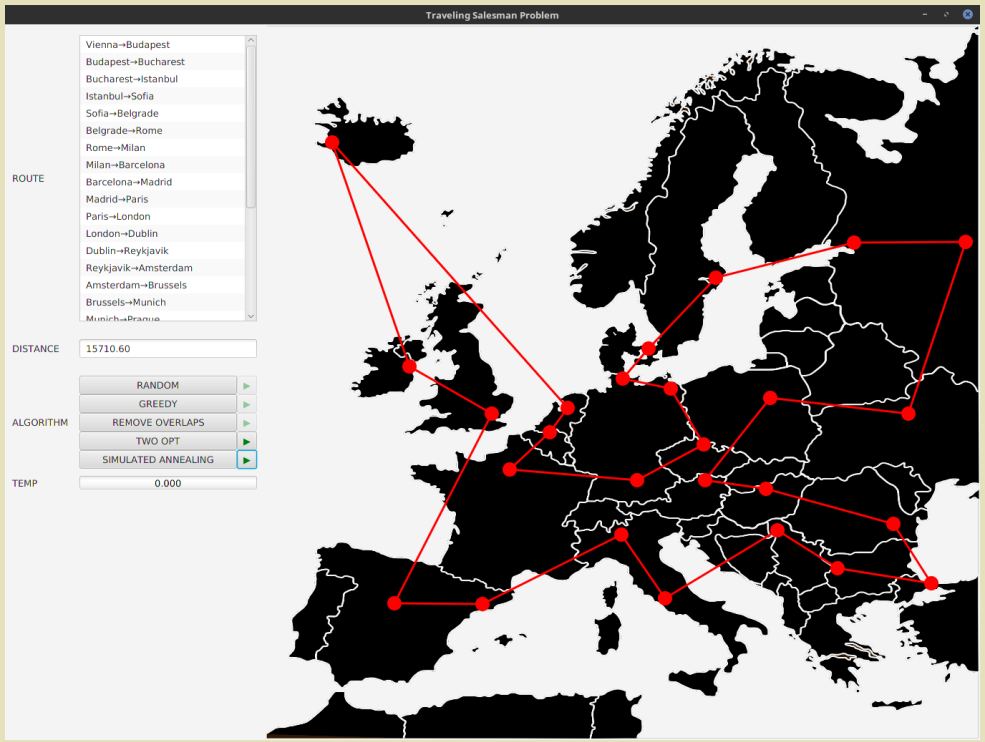
Preparation: neural networks (NN) and friends



- Task (rephrased): We have a function $\mathbb{R}^{784} \to \mathbb{R}^{10}$. How can we describe it?
- A NN is way to approximate an unknown function
- The specifics of how this works vary: there are many models and architectures

Preparation: neural networks (NN) and friends

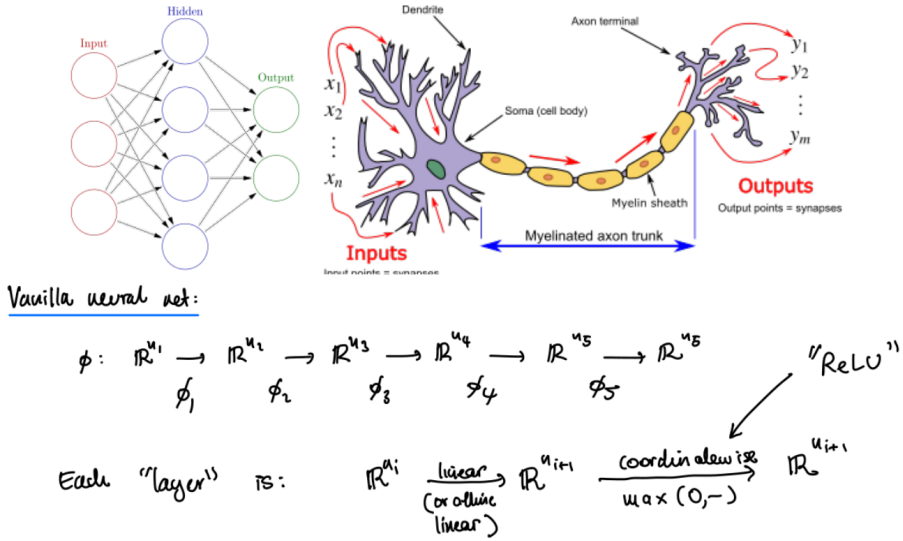
Key: Approximations to hard problems



NN are really good in finding approximate answers
for difficult problems such as traveling salesperson (TSP)

- Task (rep … e describe it?
- A NN is way to approximate an unknown function
- The specifics of how this works vary: there are many models and architectures

Preparation: neural networks (NN) and friends



A typical model architecture consists of a composition of functions, say $\phi_1, \phi_2, ..., \phi_N$. This composition is applied to the input data, and the result is processed to optimize the model parameters called weights

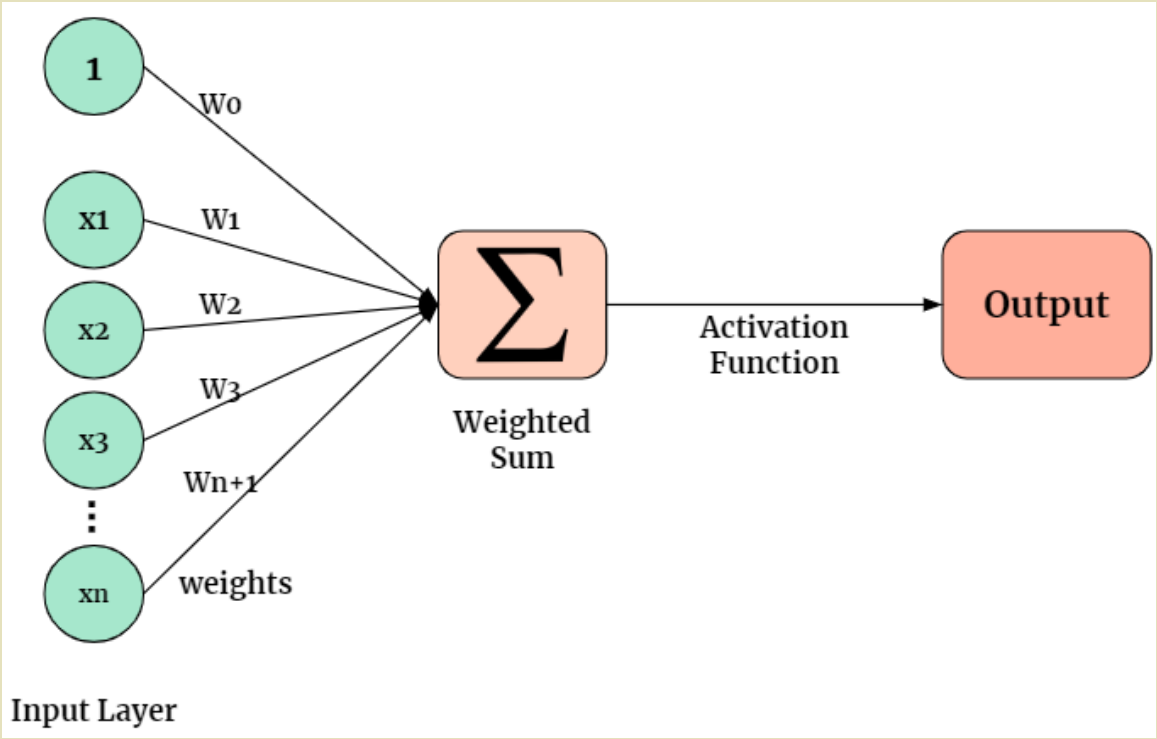
$$\phi_N \circ \phi_{N-1} \circ ... \circ \phi_1(input) = output$$

$$\phi(\mathbf{x}; \boldsymbol{\theta}) = \phi_N \circ \phi_{N-1} \circ ... \circ \phi_1(\mathbf{x}),$$

where $\mathbf{x}$ is the input and $\boldsymbol{\theta}$ is the vector of all weights

Preparation: neural networks (NN) and friends

Vanilla NN (multilayer perceptron)



When you read about NN, people almost always mean vanilla NN
Modern research usually uses beefed-up architectures
We see one, a convolutional NN (CNN) on the next slide

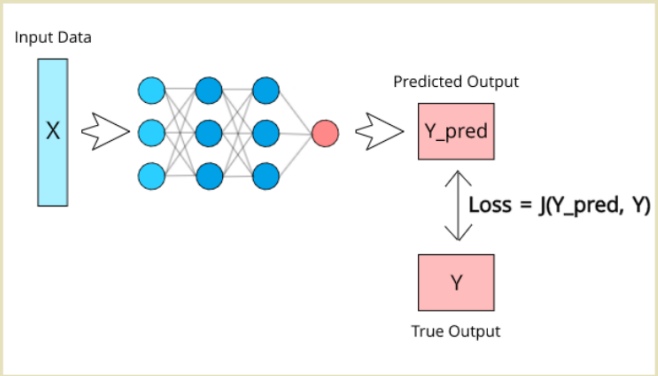
A typical r
$\phi_1, \phi_2, \ldots,$
processed
sult is

where $\mathbf{x}$ is the input and θ is the vector of all weights

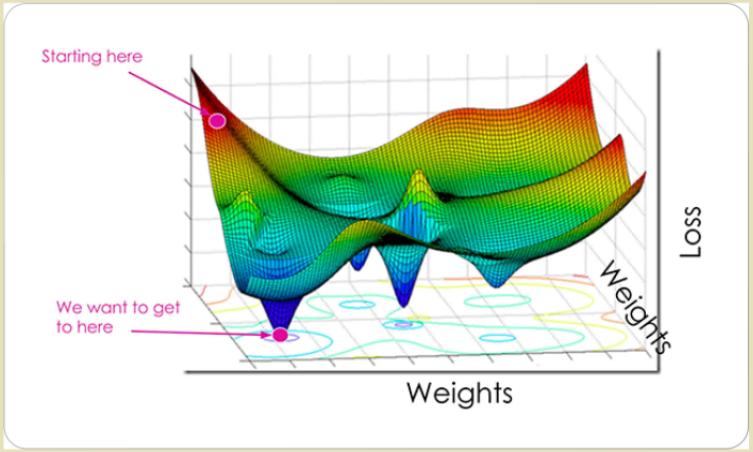
Preparation: neural networks (NN) and friends

Loss

The difference between real values and predictions



Task Minimize loss function



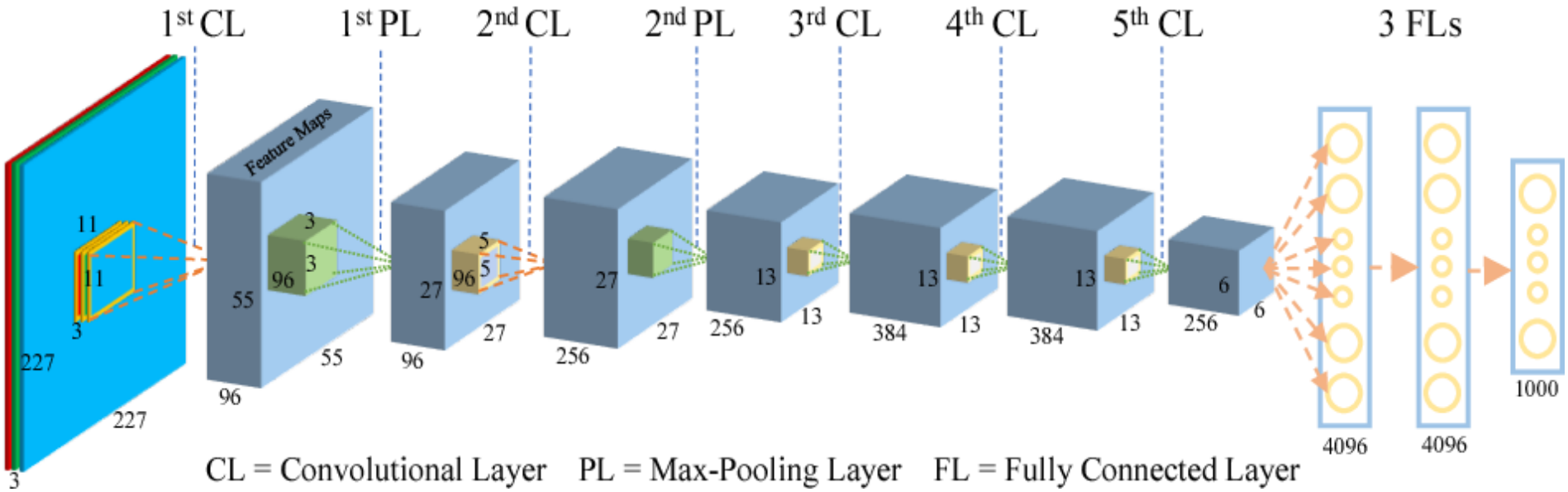
Loss function L measures the discrepancy between predictions and ground truth:

$$L = L(predictions, truths) = L(\phi(input), truths) = discrepancy$$

$$\min_{\boldsymbol{\theta} \in \text{dom } L} L \quad \Rightarrow \quad \boldsymbol{\theta}^{(l+1)} = \boldsymbol{\theta}^{(l)} - \eta \nabla L(\boldsymbol{\theta}^{(l)}), \tag{1}$$

where the superscript (l) denotes the l-th iteration, η is the **learning rate**, and ∇L is the **gradient** of the loss function with respect to the weights

Preparation: neural networks (NN) and friends

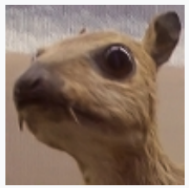
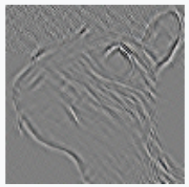
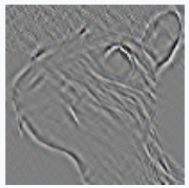
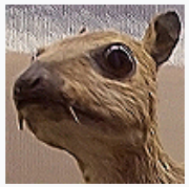


- Task Classify images by detecting local patterns
- Key idea Convolutions detect features (edges, textures)
- CNN structure:
  (a) Convolutional layers: Extract features
  (b) Pooling layers: Downsample the image Ignore for today
  (c) Fully connected layers: Classify the image Ignore for today

Preparation: neural networks (NN) and friends

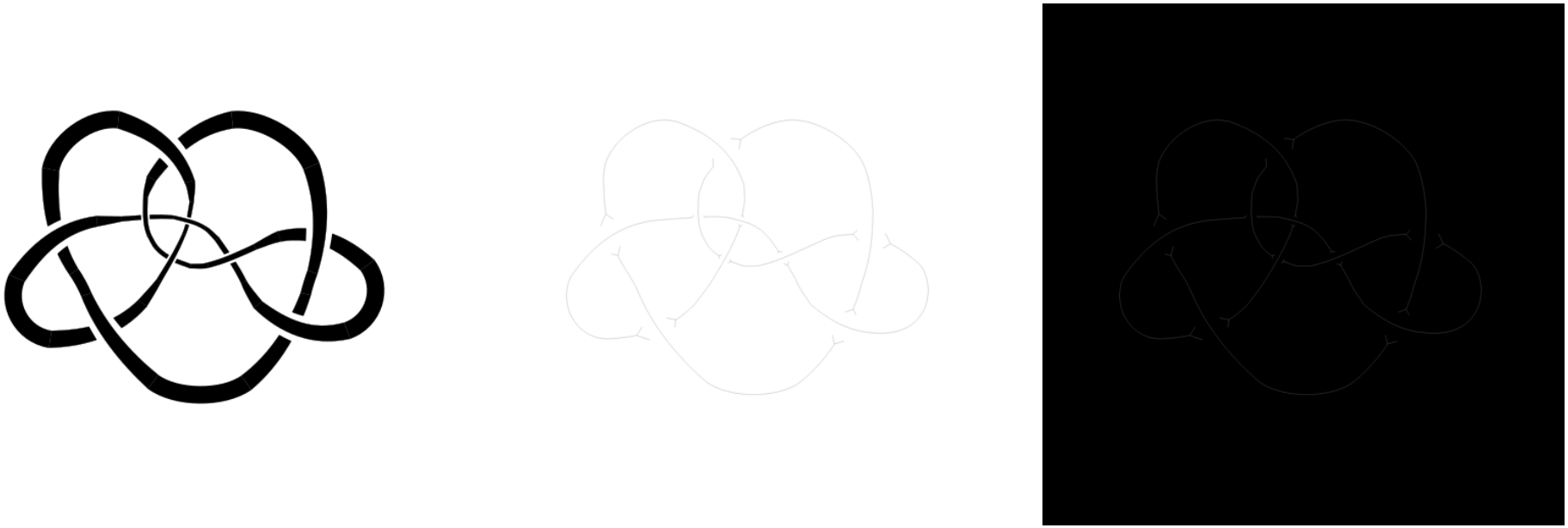
Let us open https://poloclub.github.io/cnn-explainer/

Idea (kernel, filter, etc.)

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or **edge detection**	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

In image processing, a kernel, convolution matrix, or mask is a small matrix used for blurring, sharpening, embossing, edge detection, and more
This is accomplished by doing a convolution between the kernel and an image

Preparation: neural networks (NN) and friends



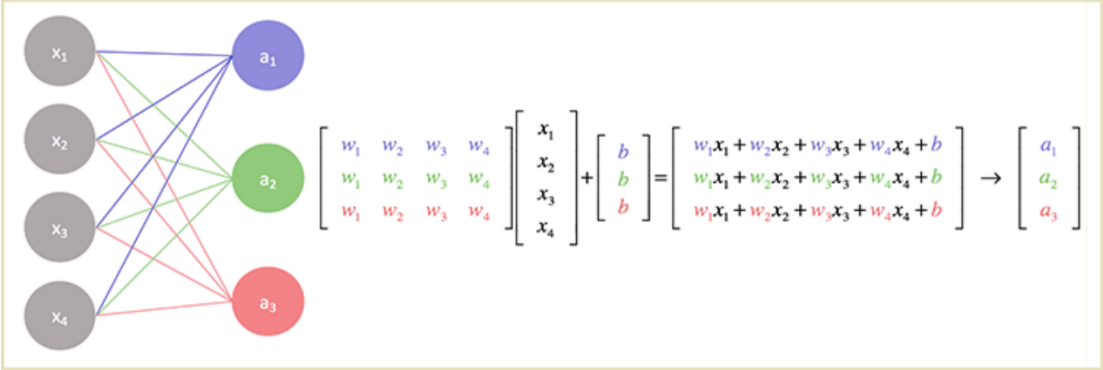
(a) Original knot **(b)** Skeletonized **(c)** Inverted

Figure: Steps of image preprocessing

- Our data source were many knot images as in (a)
- We ran a skeletonization on it

Results

Boils down to a bunch of matrix multiplications



$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \rightarrow \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

followed by the nonlinear activation e.g. ReLU

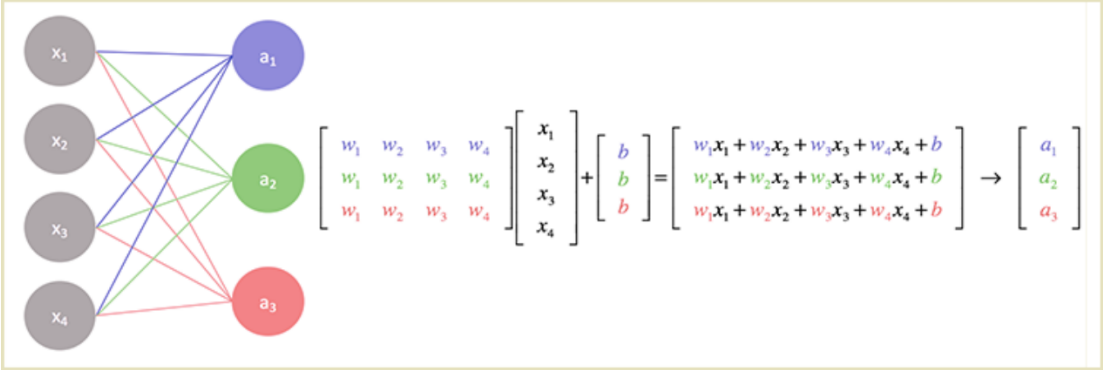
The easiest architecture is linear layer + nonlinear activation function e.g. ELU.

$$FC(\mathbf{x}) = ELU(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$Vanilla = FC^k,\ k \in \mathbb{N}$$

Results

Boils down to a bunch of matrix multiplications



$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \rightarrow \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

followed by the nonlinear activation e.g. ReLU

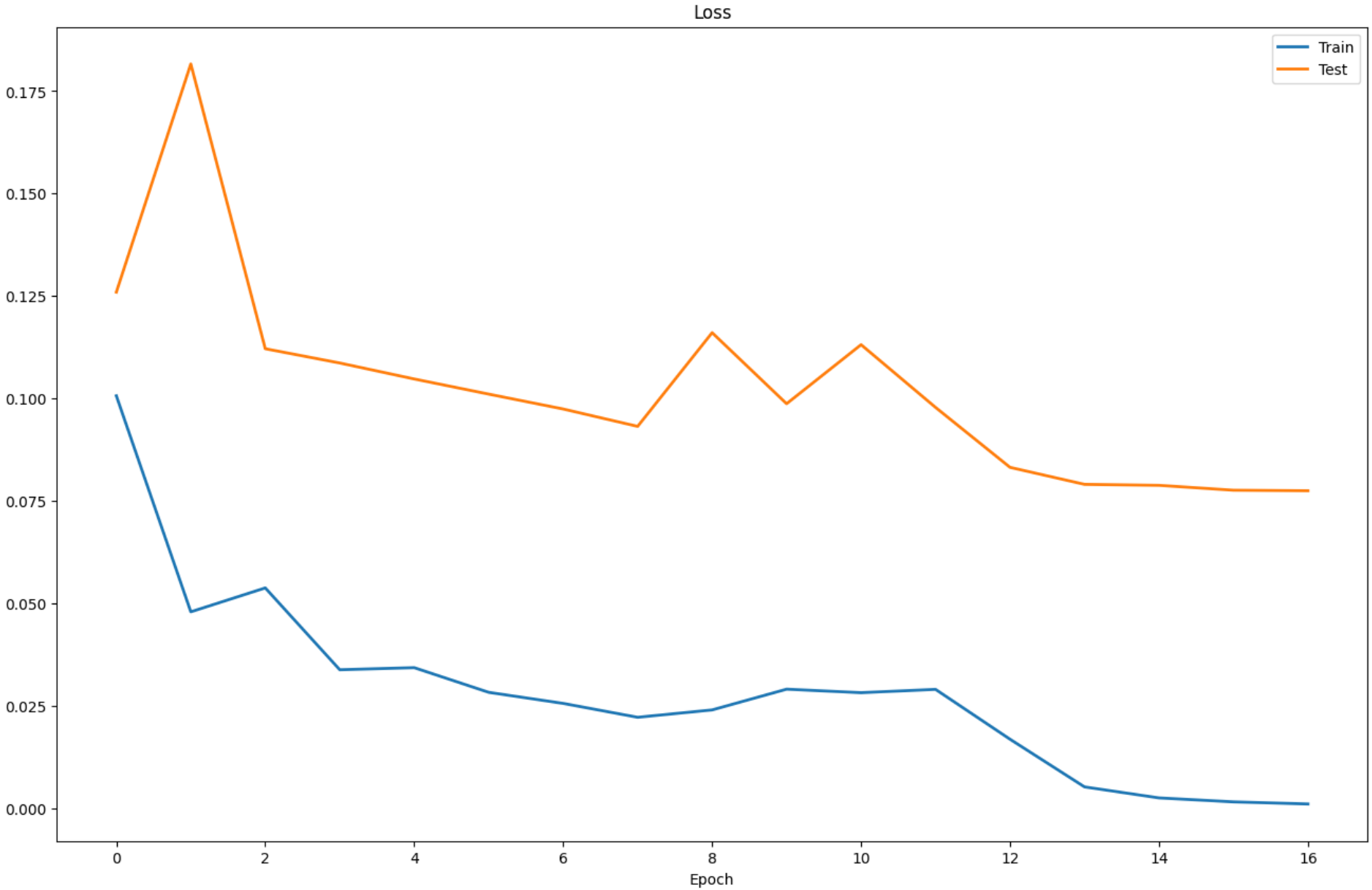
The easiest architecture is linear layer + nonlinear activation function e.g. ELU.

$$FC(\mathbf{x}) = ELU(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$Vanilla = FC^k,\ k \in \mathbb{N}$$

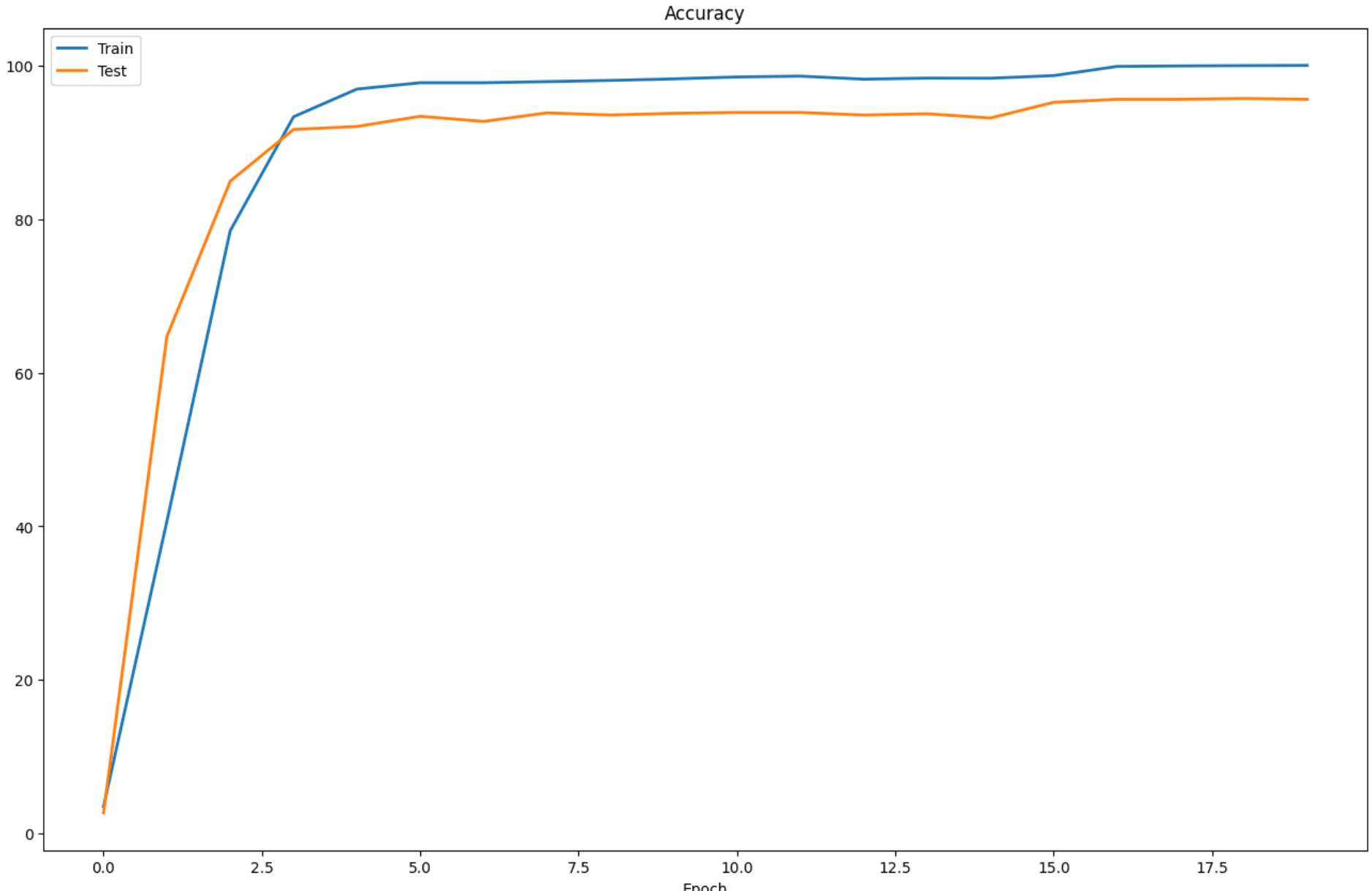
Results

Vanilla, loss



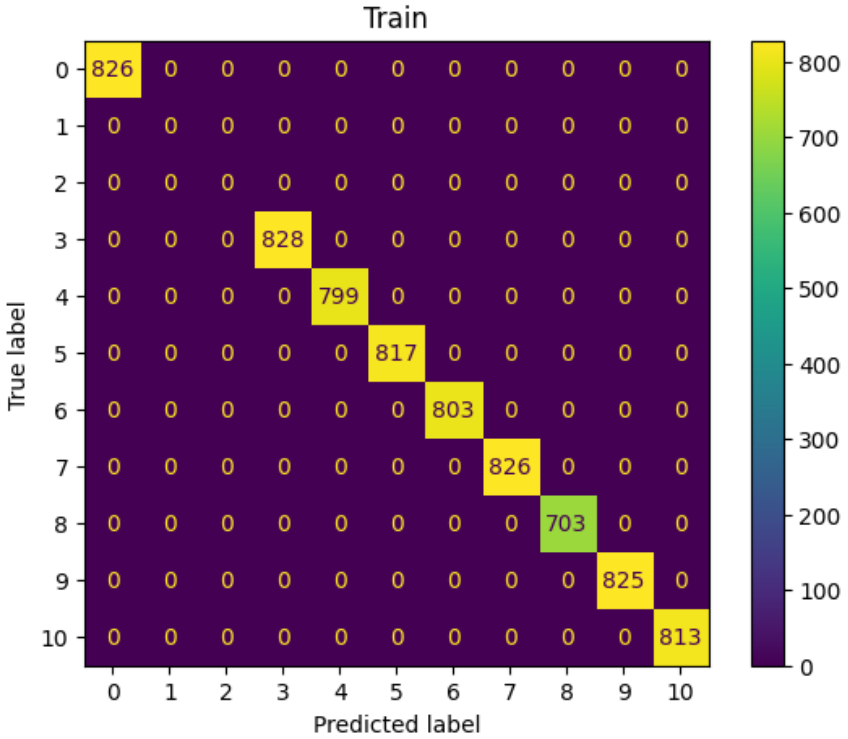
Results

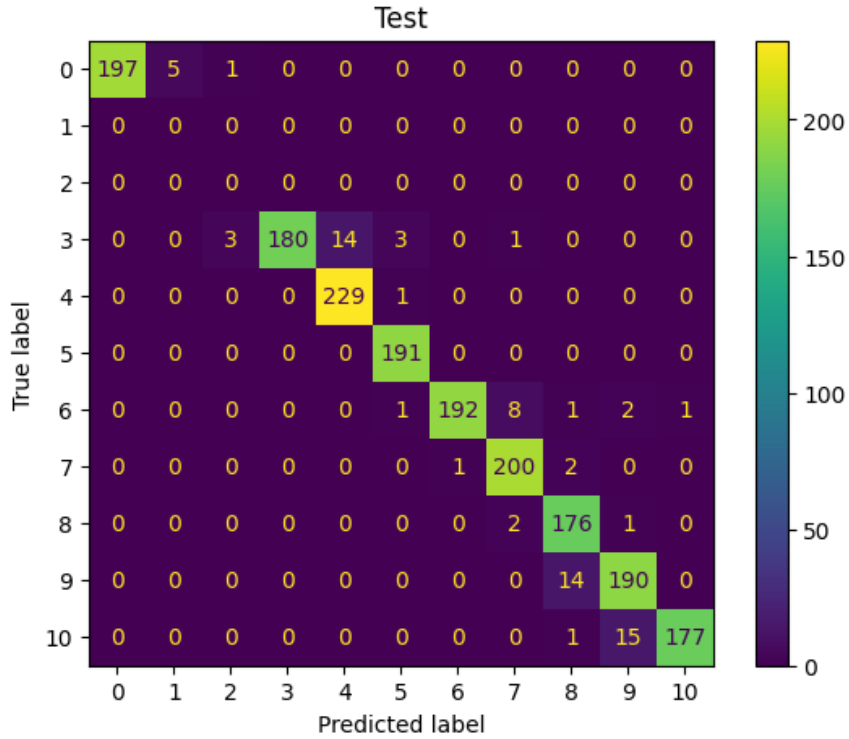
Vanilla, accuracy



Results

Vanilla, confusion matrices





Results

Vanilla, weights

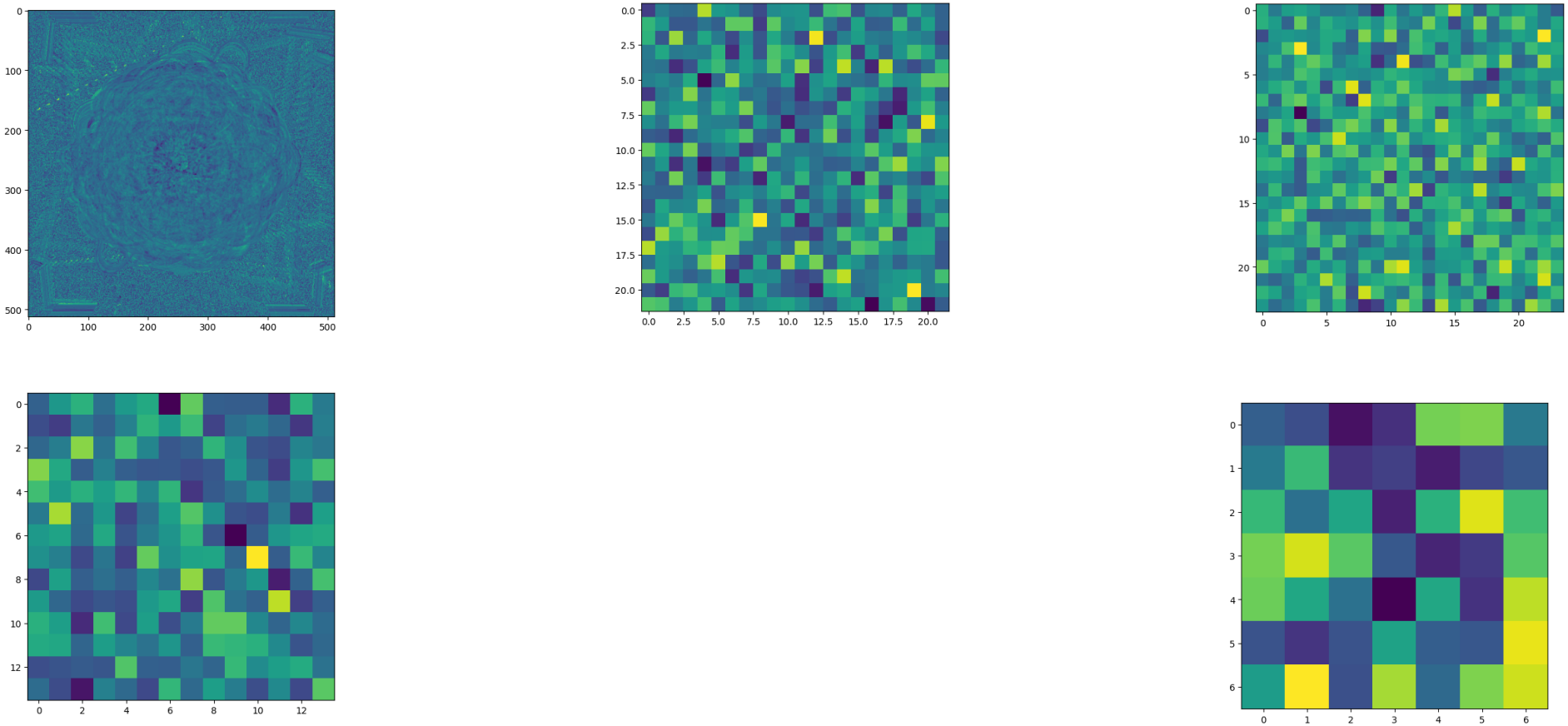
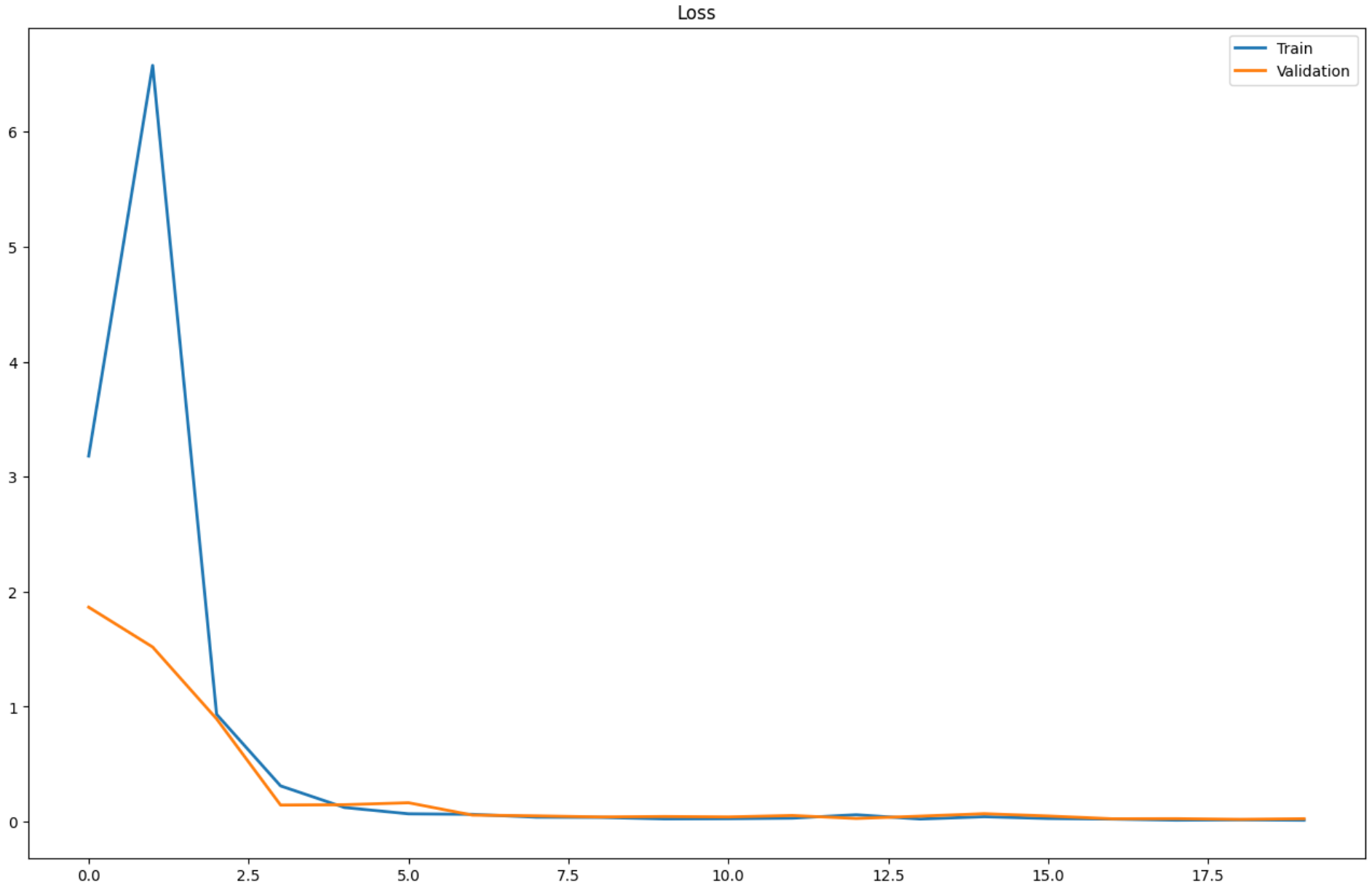


Figure: Weights of different Vanilla's layers – pretty random!

Problem: This is random, doesn't seem generalizable

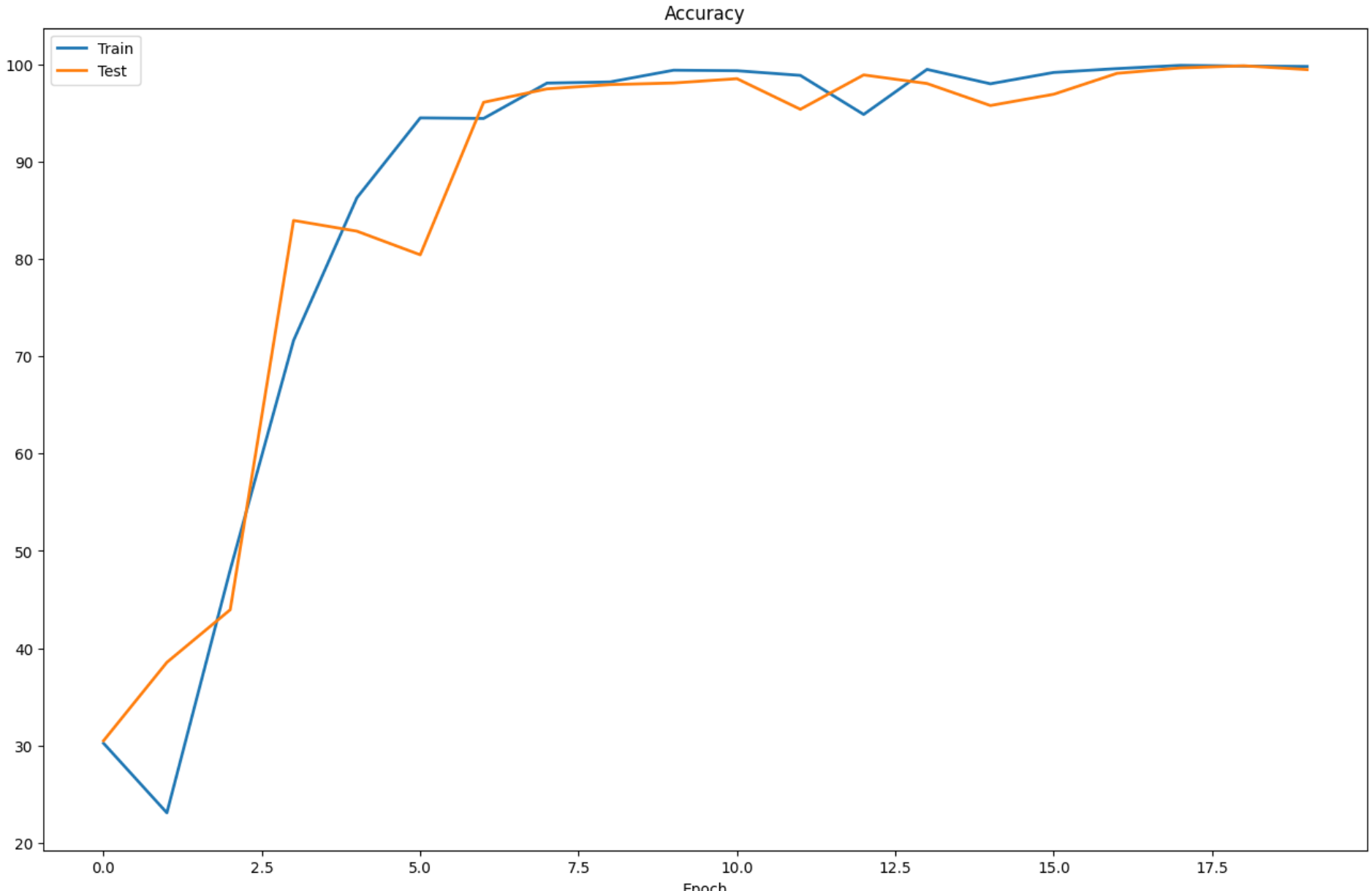
Results

CNN, loss



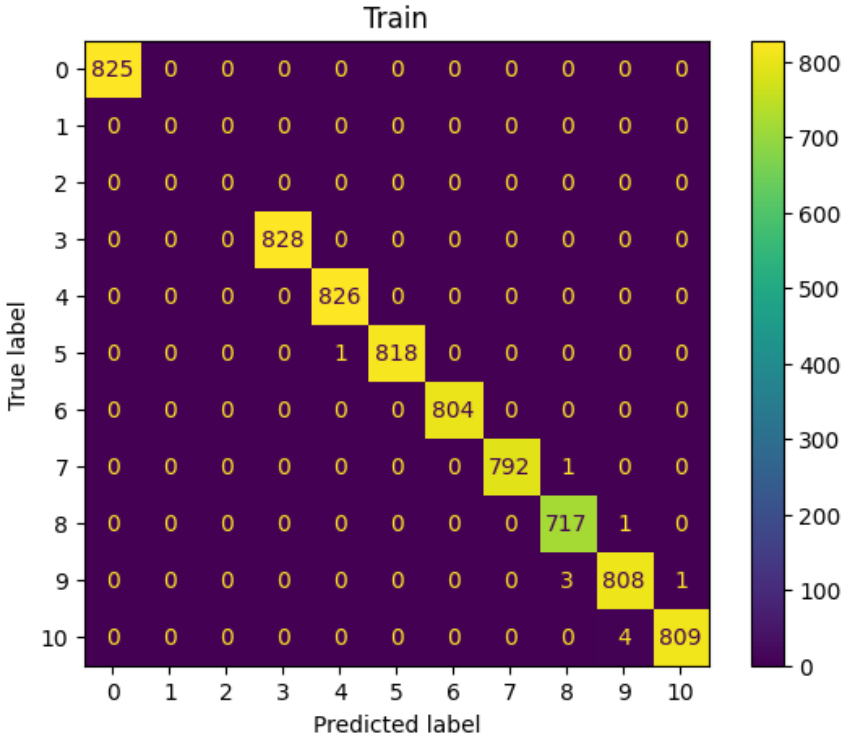
Results

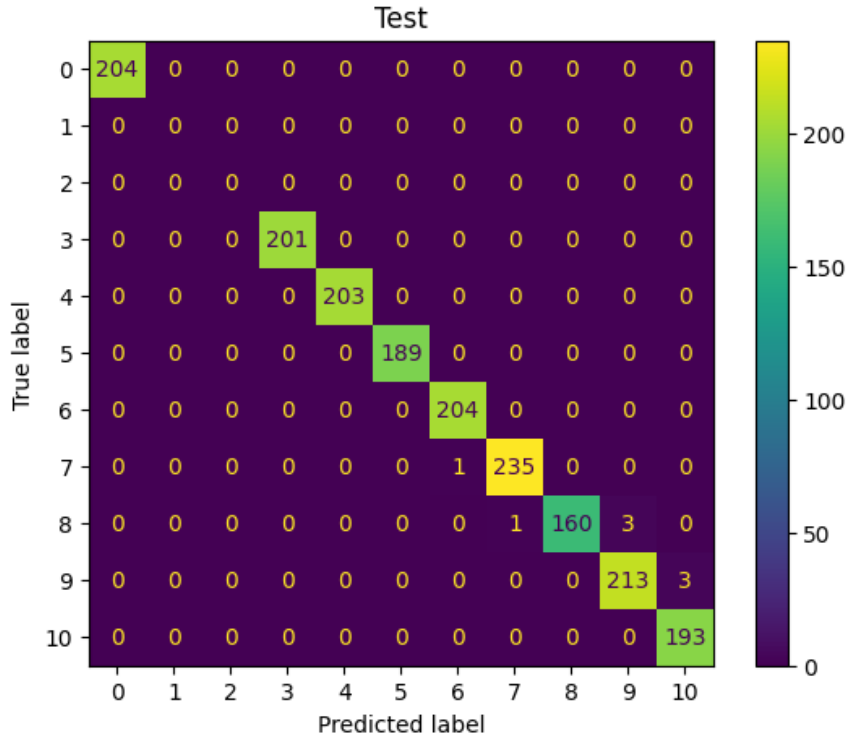
CNN, accuracy



Results

CNN, confusion matrices





Results

CNN, weights

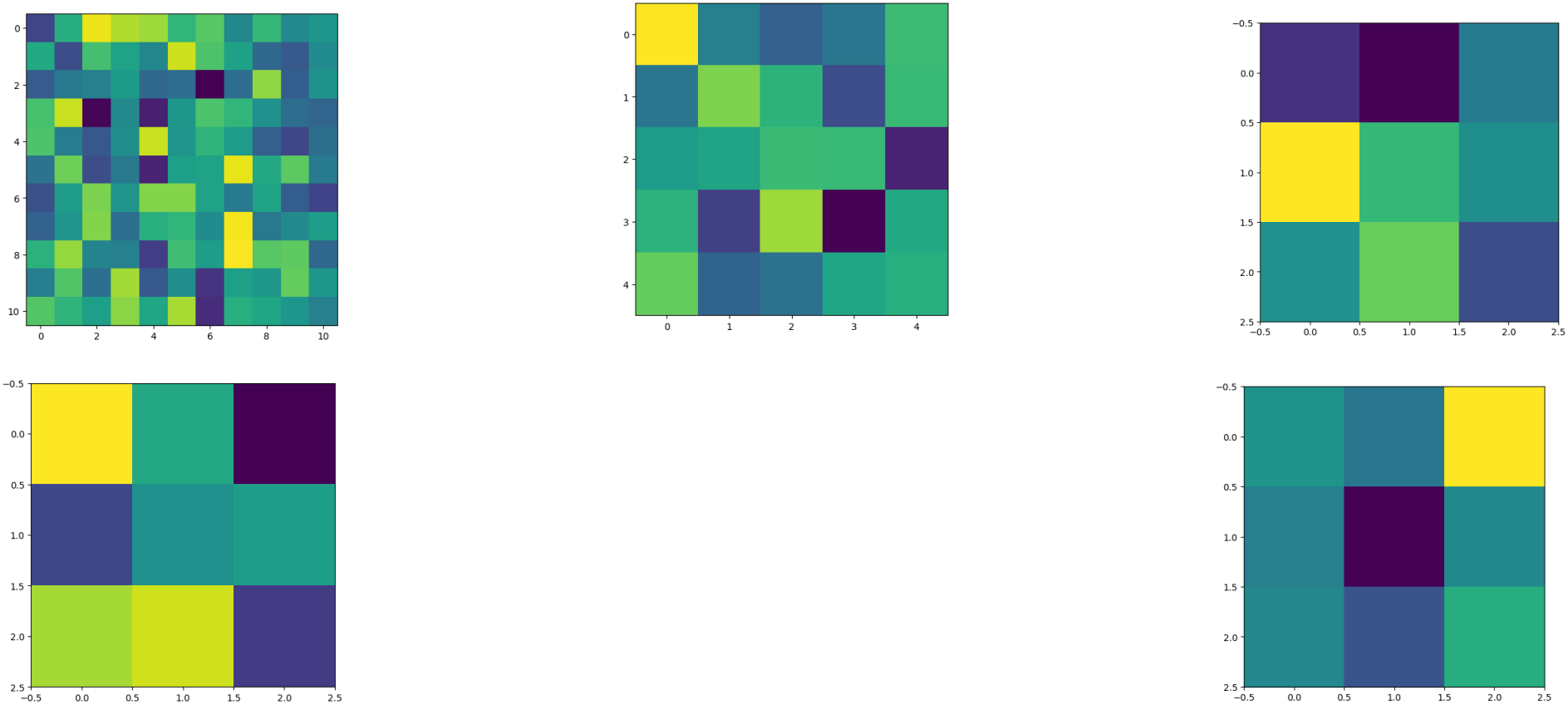
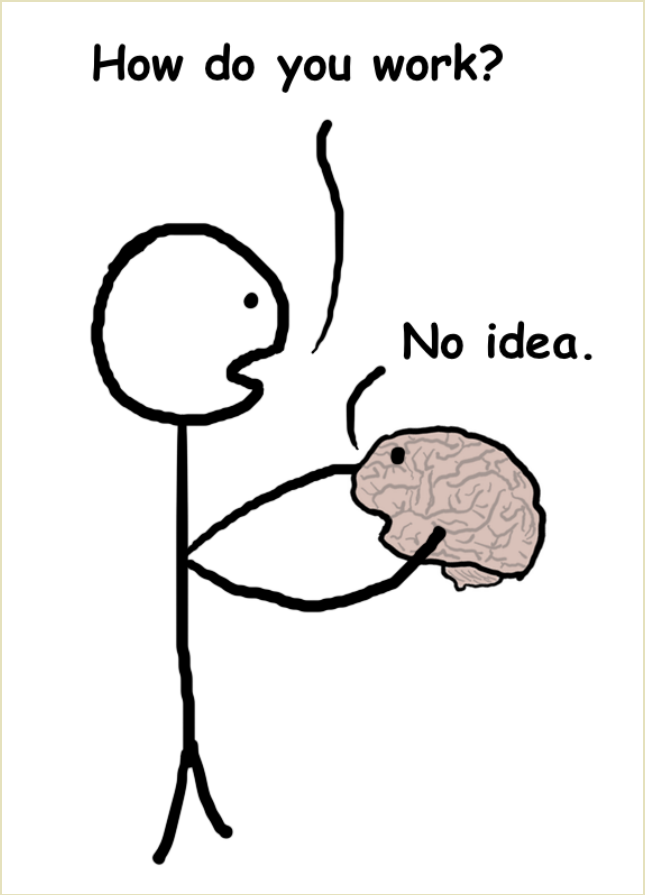


Figure: Weights of different CNN's layers

Turns out this is not random: we feed in a knot next!

Is a NN a black box? Well...yes and no!



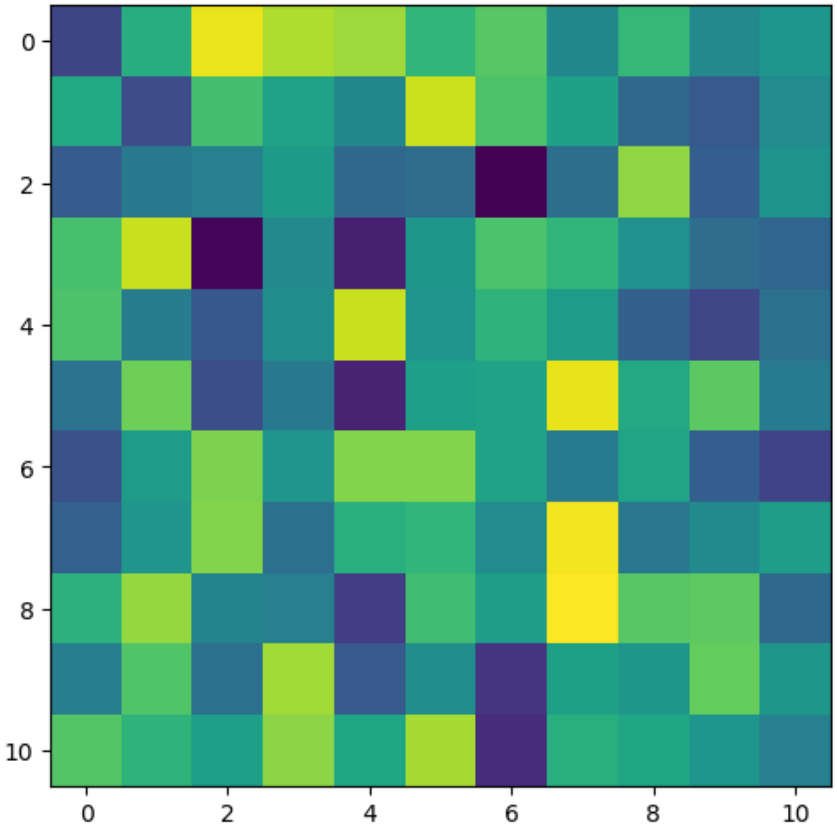
The vanilla NN learns the pixel distribution – and that is why it looks random!
Most vanilla NN do something similar – and that is why people call them black boxes

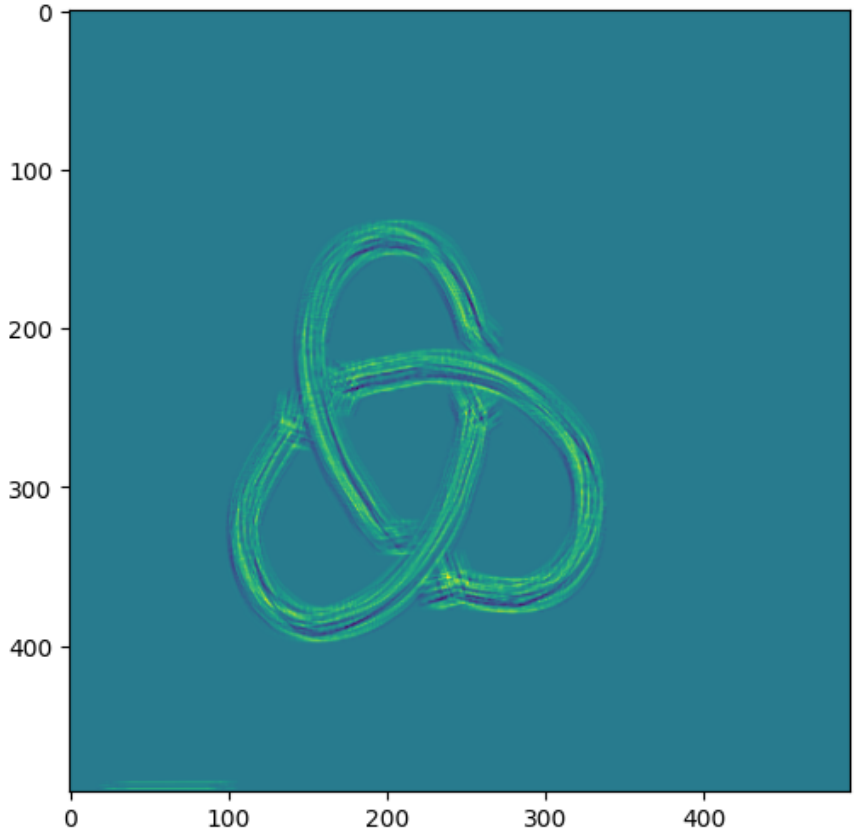
For more sophisticated architectures it is often possible to understand what is going on!

We see that next for the CNN – it learns what crossings are!

Results

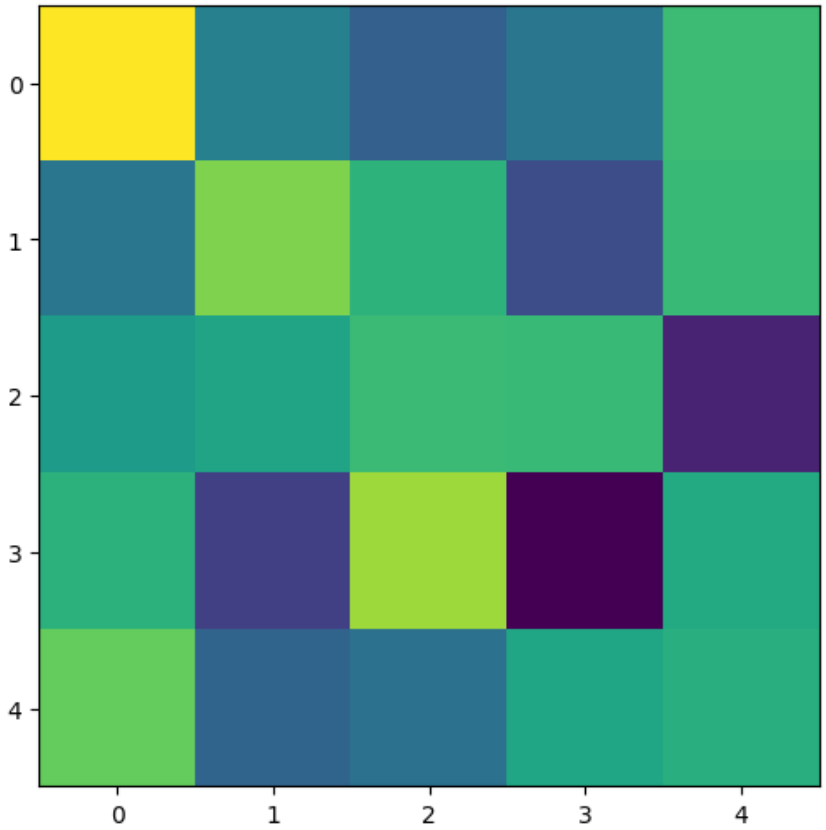
What does the CNN actually do?

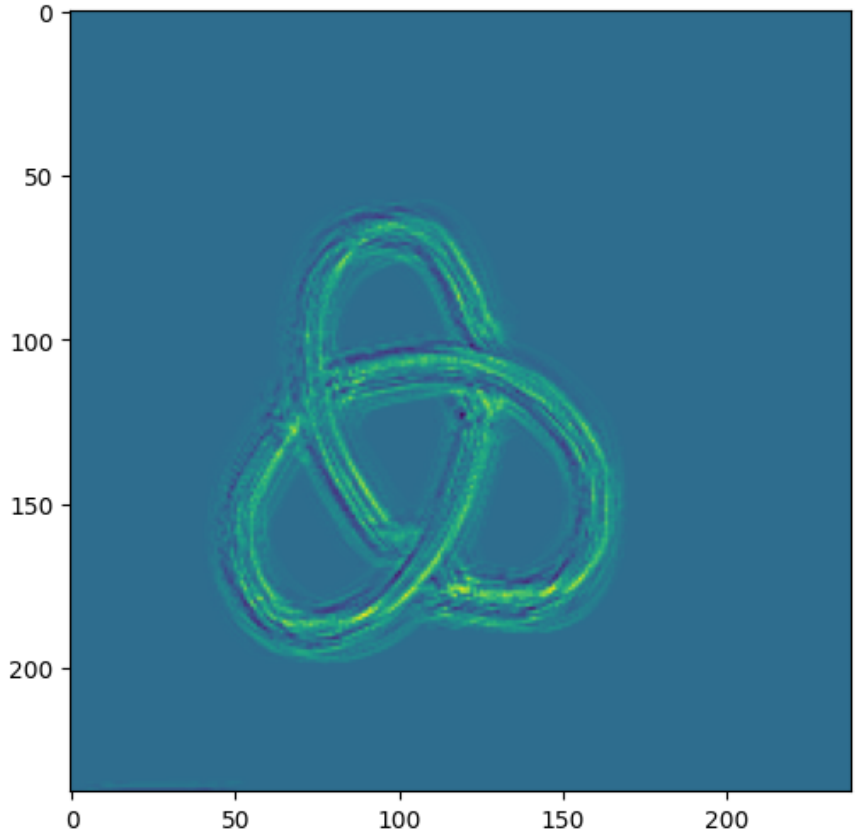




Results

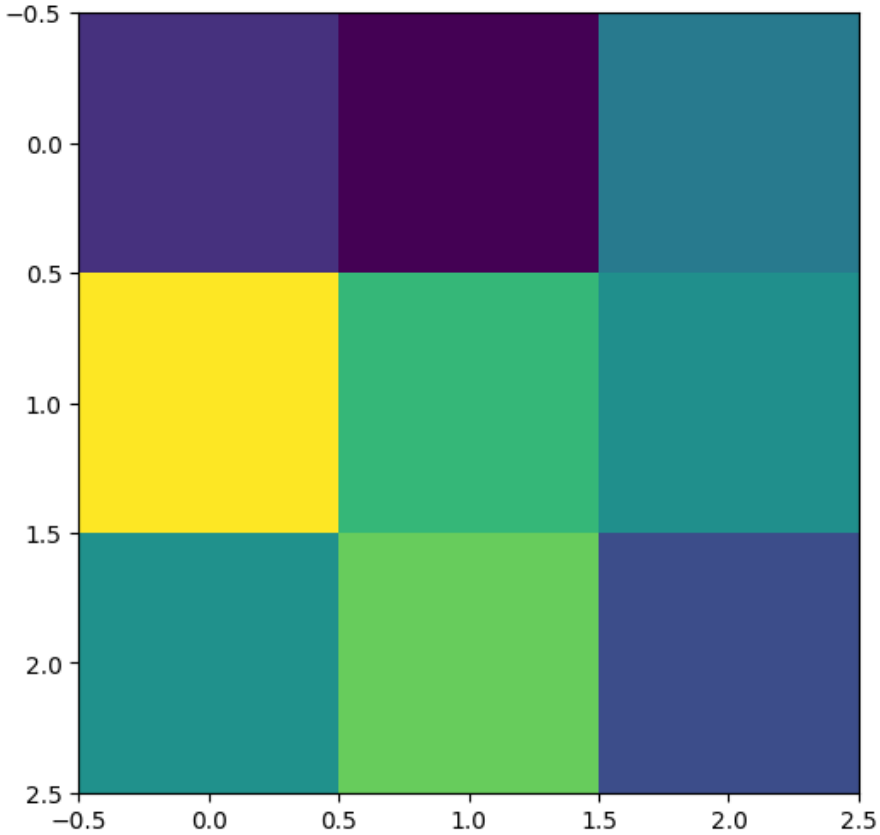
What does the CNN actually do?

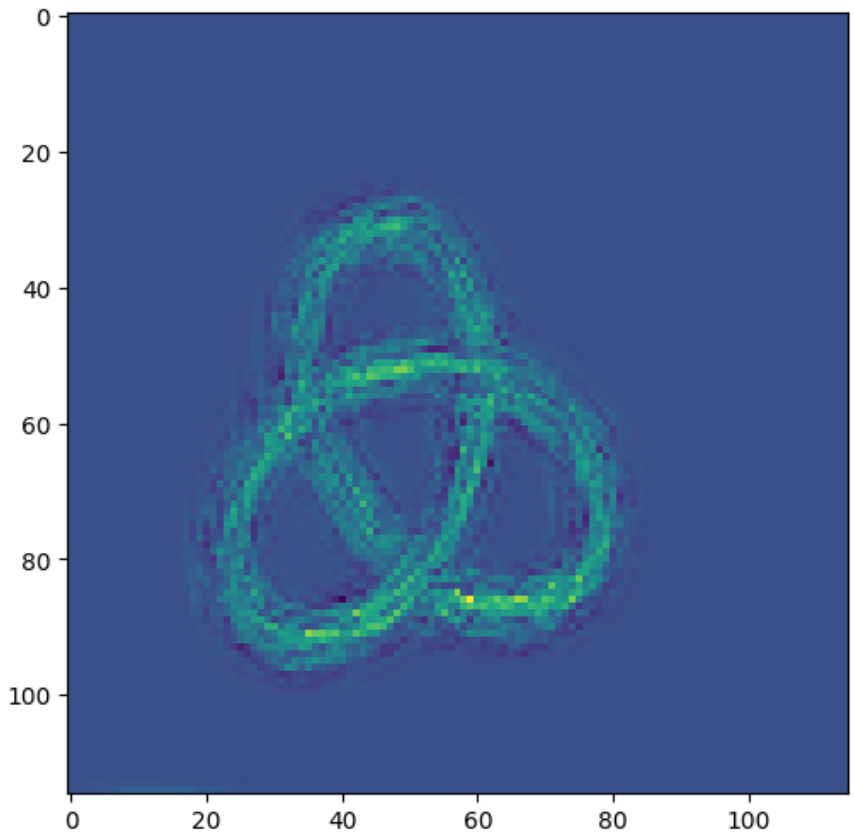




Results

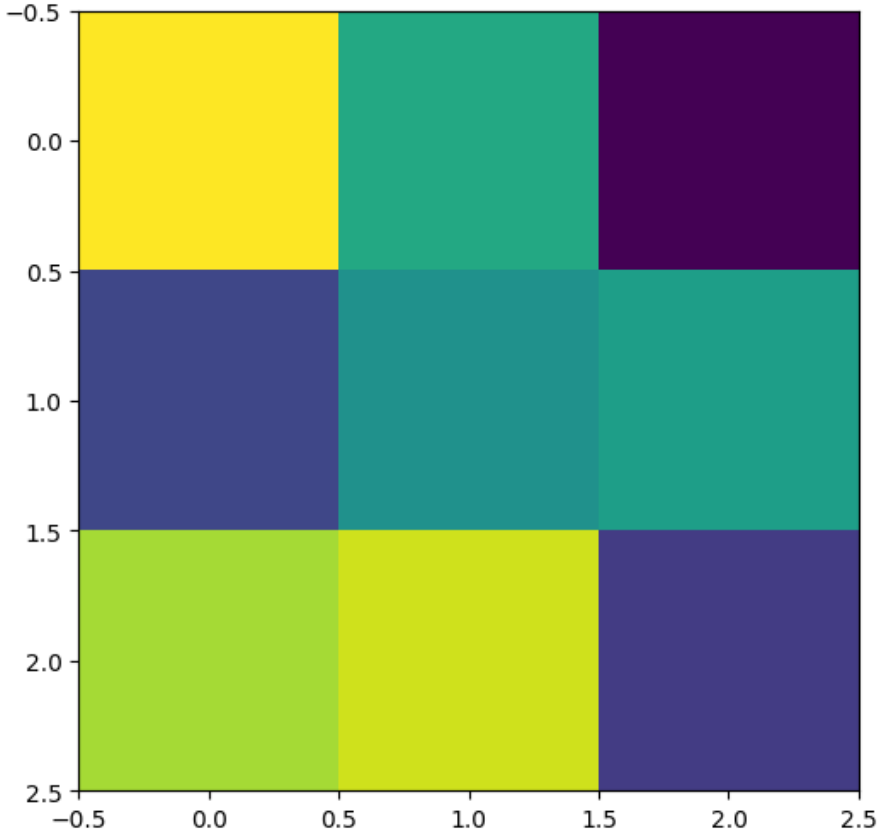
What does the CNN actually do?

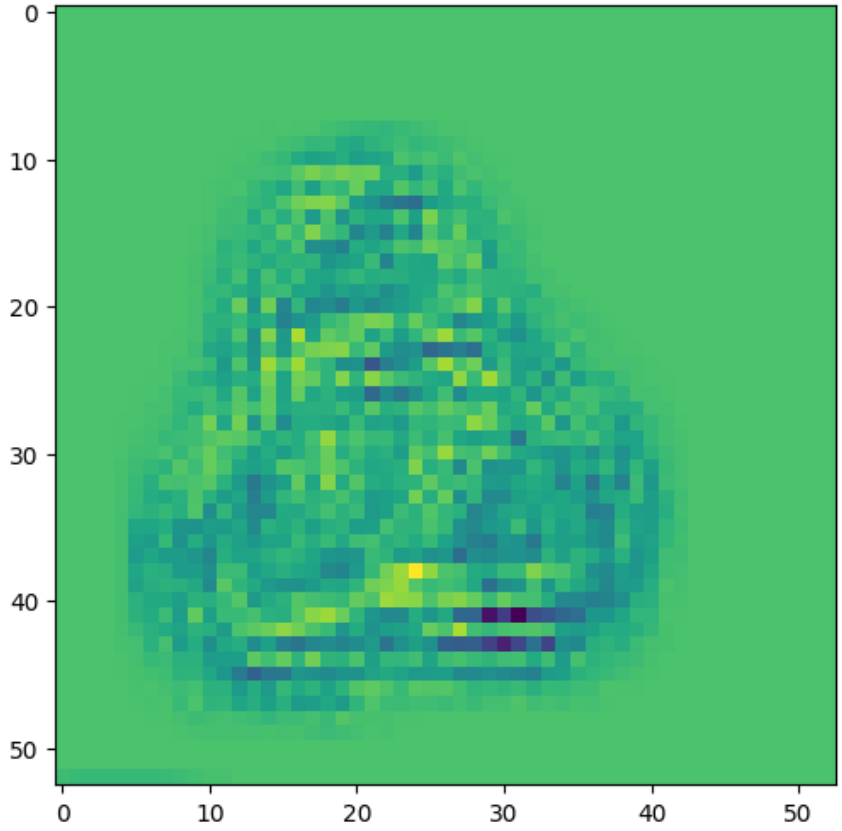




Results

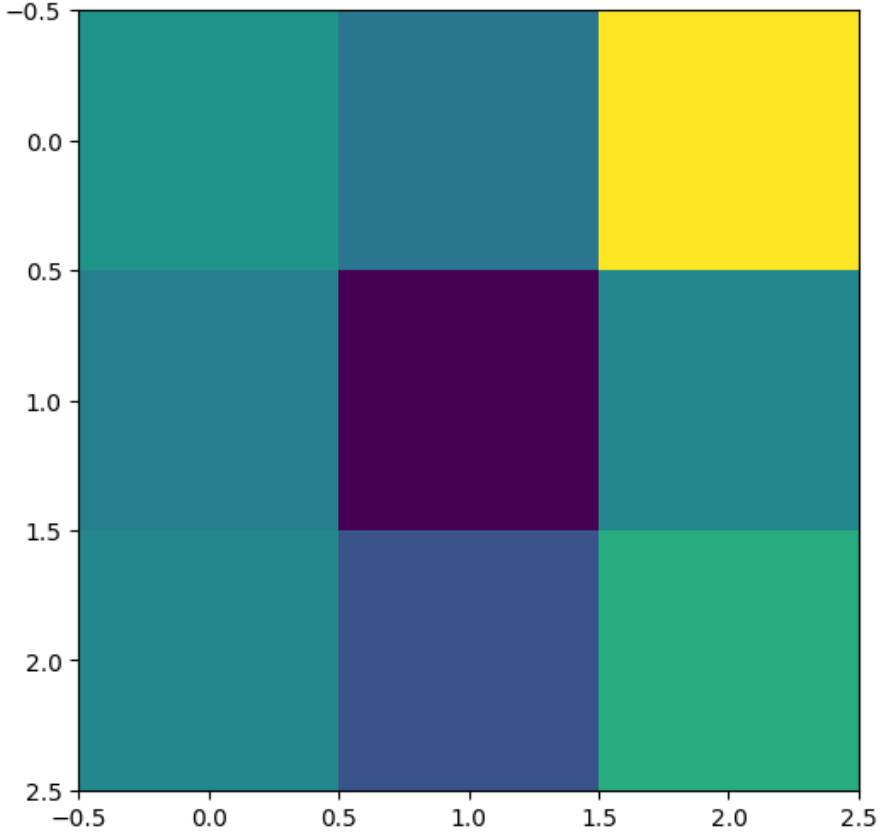
What does the CNN actually do?

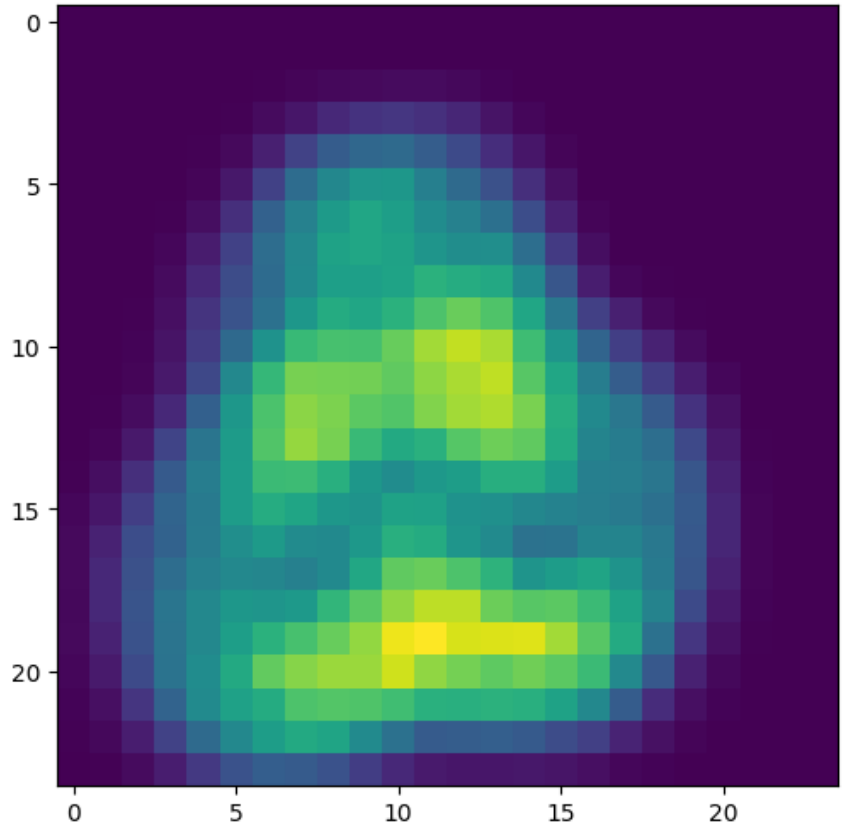




Results

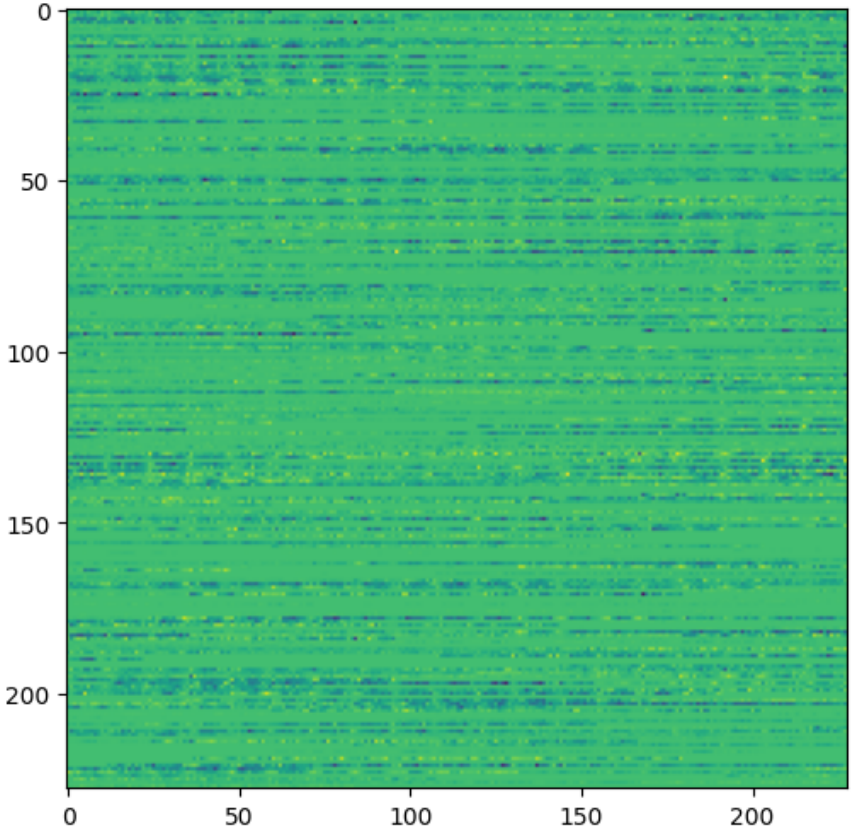
What does the CNN actually do?

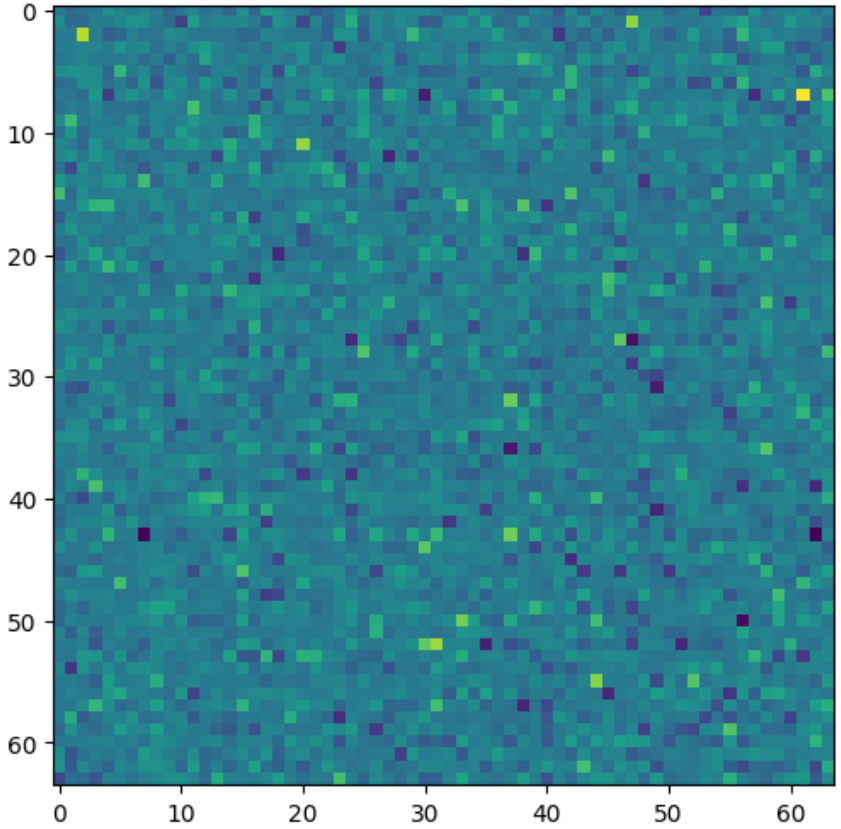




Results

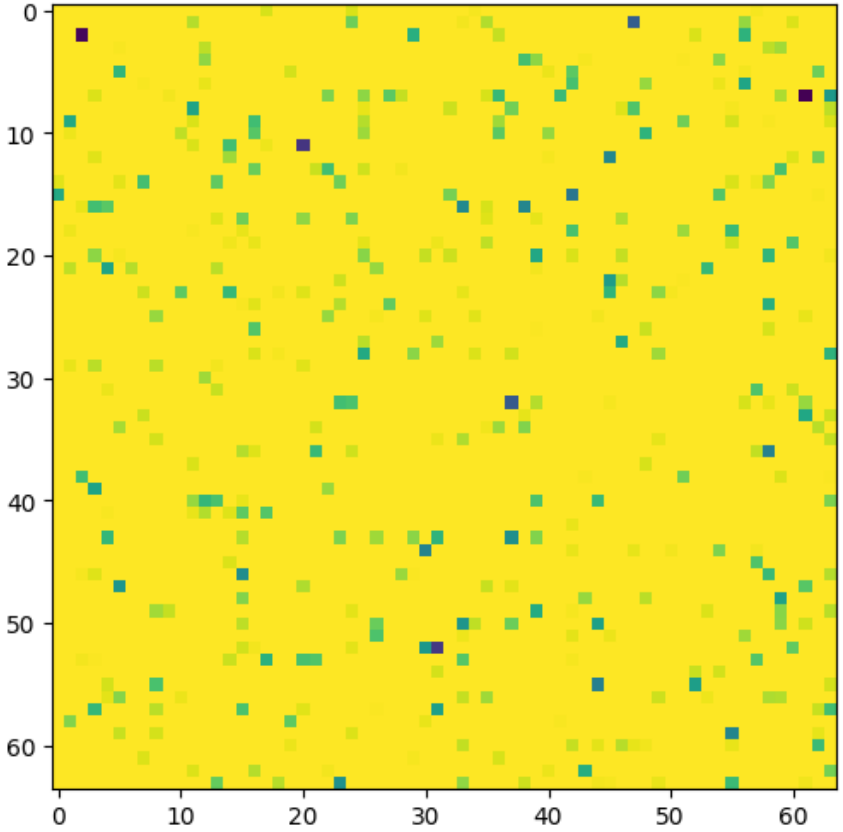
What does the CNN actually do?

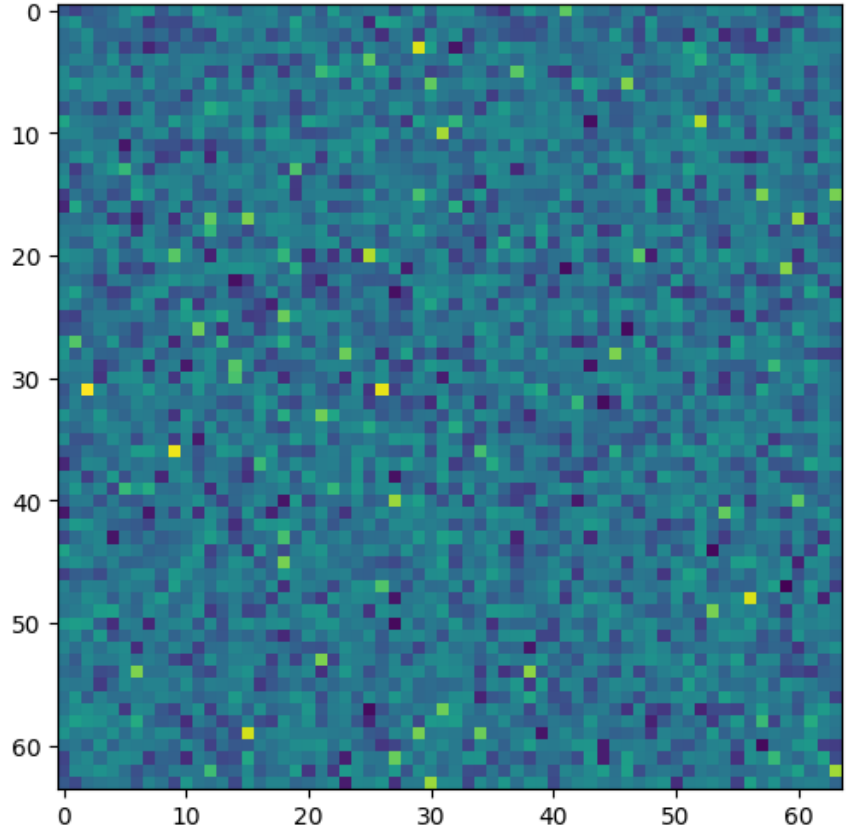




Results

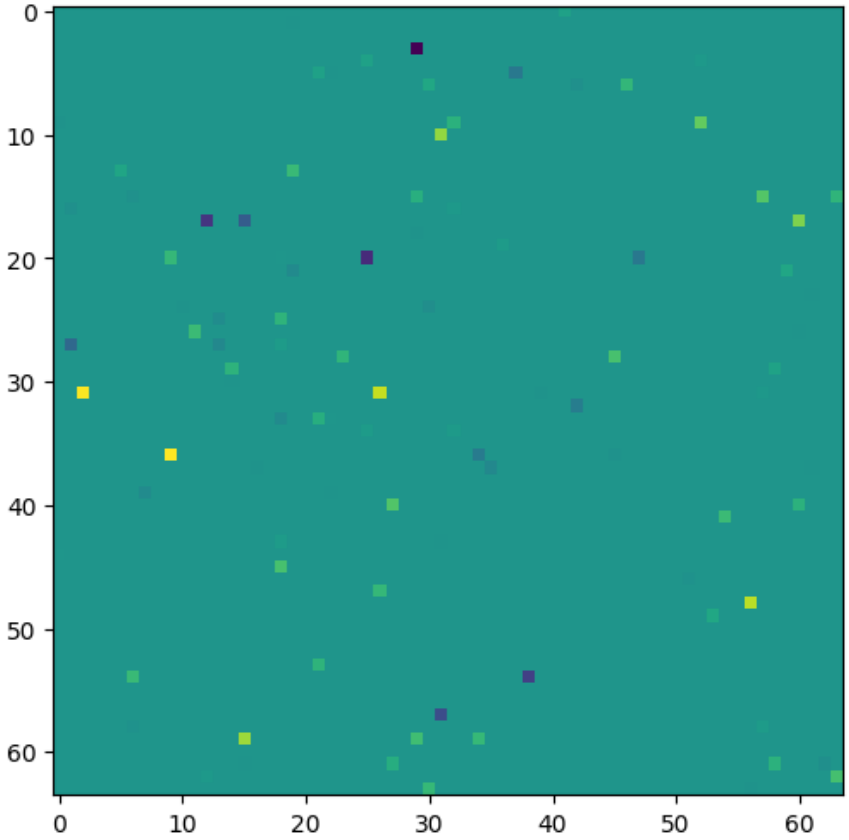
What does the CNN actually do?

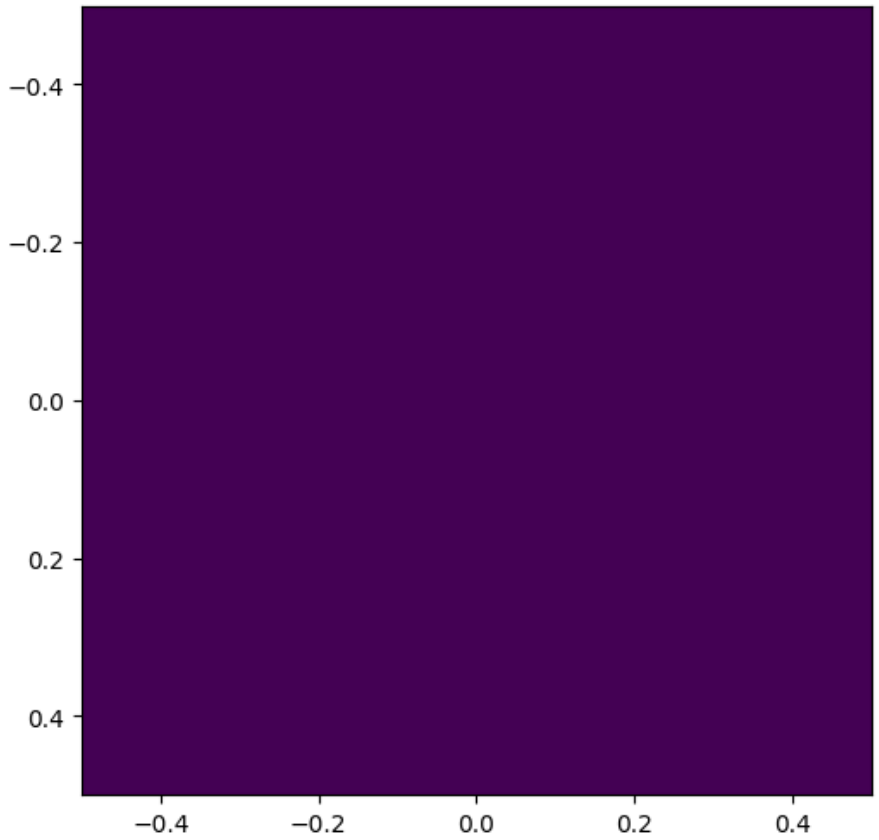




Results

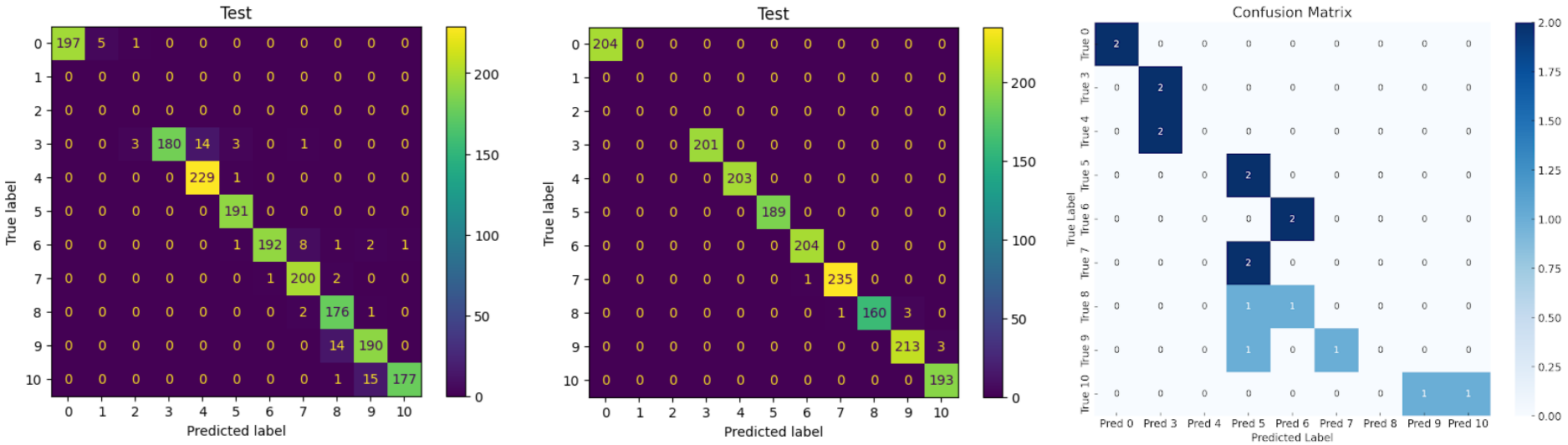
What does the CNN actually do?





Results

Comparison



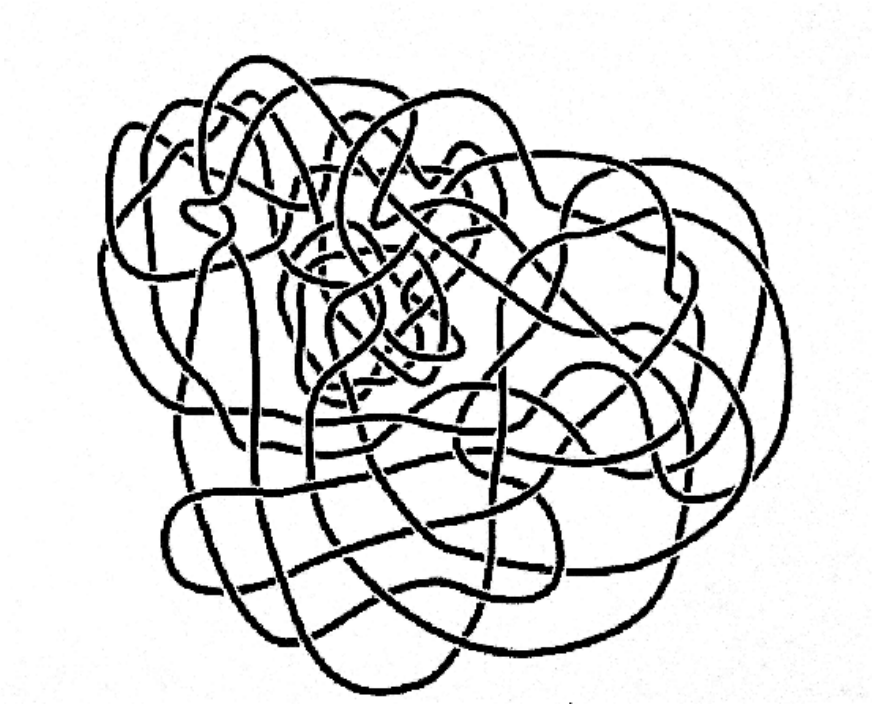
(a) Vanilla, acc.≈95.74%

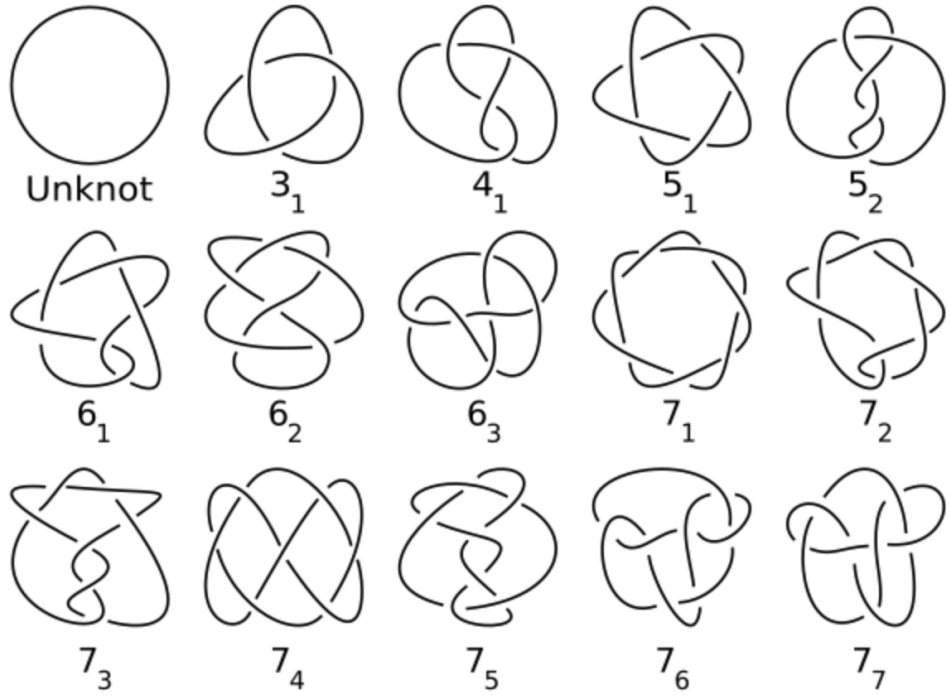
(b) CNN, acc.≈99.50%

(c) GPT, acc.≈50.00%

Figure: Confusion matrices on test dataset

Next steps? Identify knots using the quantum invariants





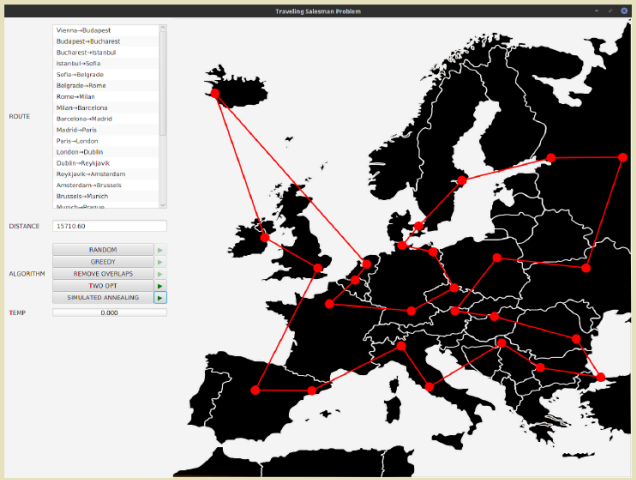
- Above, left A nasty picture of the unknot
- Above, right The first few knots (easy diagrams)
- Knot detection is not (expected to be) an easy problem: there are too many nasty diagrams – this is were NN come into the game

Next st

Recall

Preparation: neural networks (NN) and friends

Key: Approximations to hard problems



NN are really good in finding approximate answers
for difficult problems such as traveling salesperson (TSP)

xample

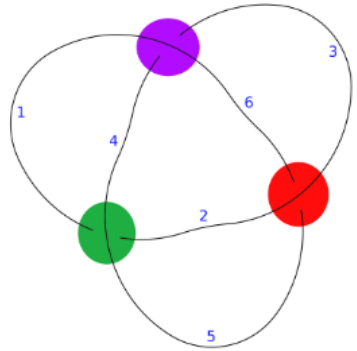
60% certainty
his is a 3

- Task (rep e describe it?
- A NN is way to approximate an unknown function
- The specifics of how this works vary: there are many models and architectures

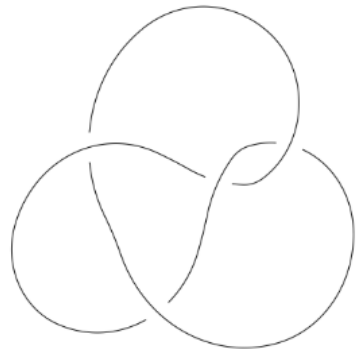
- Ab
- Ab
- Kn oo many nasty diagrams – this is were NN come into the game

NN are really good in approximating answers
And I will tell you know that one can solve the problem approximately
(This is work in progress: take it with a pinch of salt)

Next steps? Identify knots using the quantum invariants

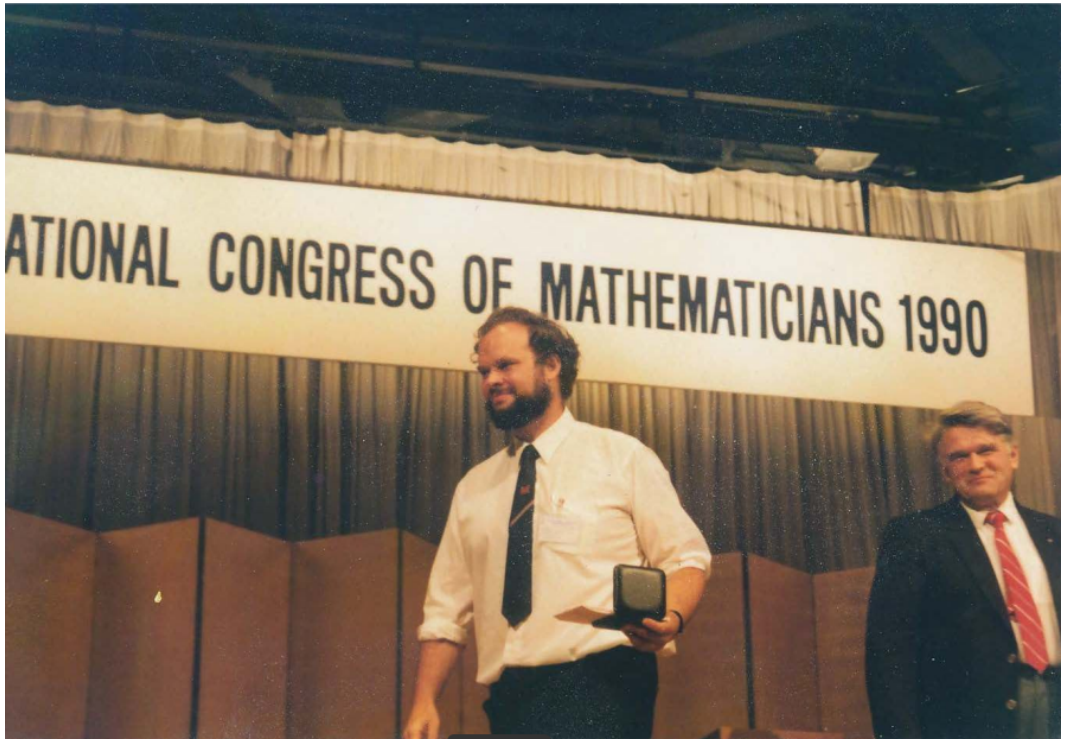
$PD[X[1,5,2,4], X[3,1,4,6], X[5,3,6,2]] \rightsquigarrow$ 

,

$PD[X[4,2,5,1], X[8,6,1,5], X[6,3,7,4], X[2,7,3,8]] \rightsquigarrow$  .

- Above A PD presentation (an algebraic way to encode knot diagrams)
- Using the same architecture as before, we can read this in from an image
- This is Step 1

Next steps? Identify knots using the quantum invariants

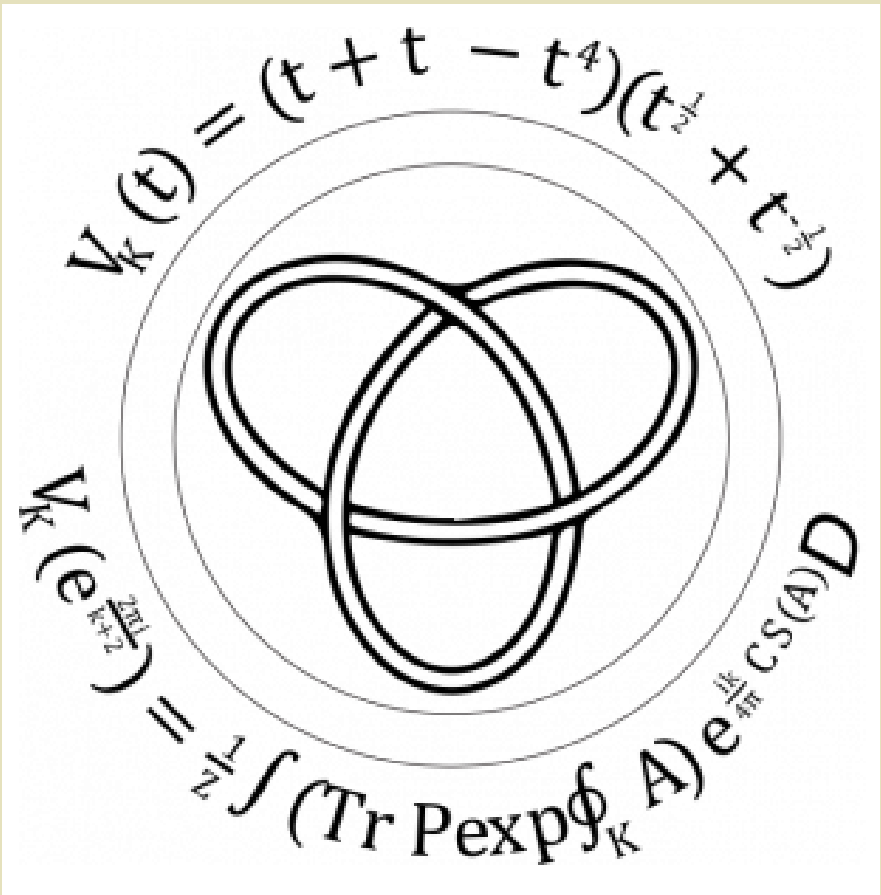


- Above Jones receives the fields medal (also for the Jones polynomial)
- From the PD presentation one can compute the Jones polynomial or friends (we have a program that can do this fast for 100 crossing knots)
- This is Step 2

Next steps? Identify knots using the quantum invariants

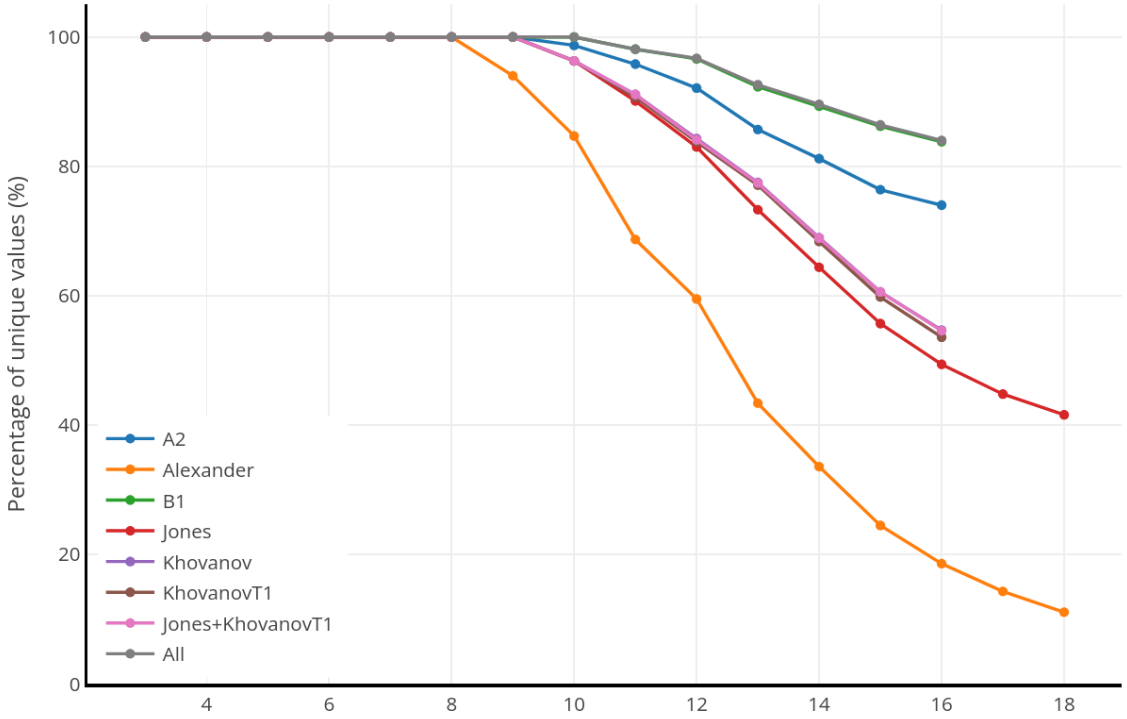
For this talk: Jones polynomial

= a number thing that you can compute from a knot diagram



- Above … mial)
- From t… l or friends (we ha…
- This is Step 2

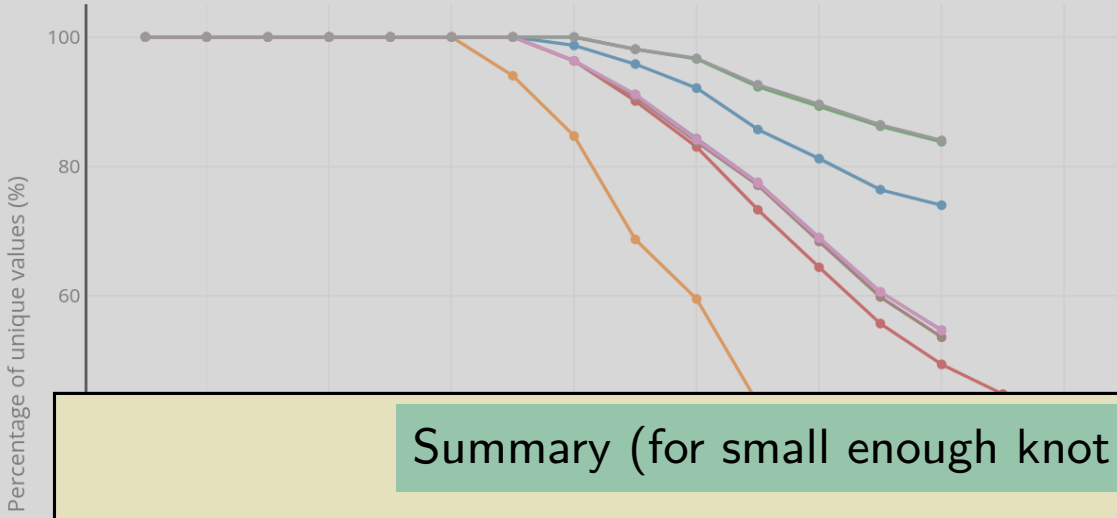
Next steps? Identify knots using the quantum invariants



Crossing number / number of diagrams	3 36	4 276	5 2936	6 35 872	7 484 088	8 6 967 942	9 105 555 336	10 1 664 142 836
0_1	34	265	2744	32 456	422 332	5 852 832	85 253 534	1 291 291 155
3_1	1	5	85	1466	25 432	440 570	7 696 083	135 702 456
3_1^m	1	5	85	1466	25 432	440 570	7 696 083	135 702 456
4_1	—	1	18	412	8450	165 791	3 175 612	60 146 706
5_2	—	—	1	24	730	18 075	415 290	9 025 926
5_2^m	—	—	1	24	730	18 075	415 290	9 025 926
5_1	—	—	1	6	213	5819	142 614	3 252 078
5_1^m	—	—	1	6	213	5819	142 614	3 252 078
$3_1\#3_1^m$	—	—	—	2	112	3953	113 684	2 923 783
6_3	—	—	—	2	106	3515	96 666	2 389 180
6_2	—	—	—	1	58	2027	58 354	1 493 624
6_2^m	—	—	—	1	58	2027	58 354	1 493 624
$3_1\#3_1$	—	—	—	2	58	2006	56 893	1 461 498
$3_1^m\#3_1^m$	—	—	—	2	58	2006	56 893	1 461 498
6_1^m	—	—	—	1	34	1267	38 199	1 015 996
6_1	—	—	—	1	34	1267	38 199	1 015 996
$3_1\#4_1$	—	—	—	—	8	516	20 458	648 362
$3_1^m\#4_1$	—	—	—	—	8	516	20 458	648 362
7_6^m	—	—	—	—	3	193	7608	240 121
7_6	—	—	—	—	3	193	7608	240 121
7_7	—	—	—	—	2	124	4709	144 455
7_7^m	—	—	—	—	2	124	4709	144 455
7_5^m	—	—	—	—	2	102	4244	138 467
7_5	—	—	—	—	2	102	4244	138 467
7_2^m	—	—	—	—	1	44	2103	74 739
7_2	—	—	—	—	1	44	2103	74 739
7_3	—	—	—	—	1	39	1793	62 059
7_3^m	—	—	—	—	1	39	1793	62 059
$4_1\#4_1$	—	—	—	—	—	20	1176	51 526
7_4^m	—	—	—	—	1	36	1516	49 731
7_4	—	—	—	—	1	36	1516	49 731
8_{20}	—	—	—	—	—	14	985	41 843
8_{20}^m	—	—	—	—	—	14	985	41 843
$3_1^m\#5_2^m$	—	—	—	—	—	10	784	36 548
$3_1\#5_2$	—	—	—	—	—	10	784	36 548
$3_1^m\#5_2$	—	—	—	—	—	10	784	36 544
$3_1\#5_2^m$	—	—	—	—	—	10	784	36 544

- "Problem" The Jones polynomial is not a very good invariant
- Solution This doesn't matter as the distribution of knot diagrams is biased towards "easy knots" – and the polynomial is good on these!
- This is Step 3

Next steps? Identify knots using the quantum invariants



Crossing number / number of diagrams	3 36	4 276	5 2936	6 35 872	7 484 088	8 6 967 942	9 105 555 336	10 1 664 142 836
0_1	34	265	2744	32 456	422 332	5 852 832	85 253 534	1 291 291 155
3_1	1	5	85	1466	25 432	440 570	7 696 083	135 702 456
3_1^m	1	5	85	1466	25 432	440 570	7 696 083	135 702 456
4_1	—	1	18	412	8450	165 791	3 175 612	60 146 706
5_2	—	—	1	24	730	18 075	415 290	9 025 926
5_2^m	—	—	1	24	730	18 075	415 290	9 025 926
5_1	—	—	1	6	213	5819	142 614	3 252 078
5_1^m	—	—	1	6	213	5819	142 614	3 252 078
$3_1\#3_1^m$	—	—	—	2	112	3953	113 684	2 923 783
6_3	—	—	—	2	106	3515	96 666	2 389 180
6_2	—	—	—	1	58	2027	58 354	1 493 624
6_2^m	—	—	—	1	58	2027	58 354	1 493 624
$3_1\#3_1$	—	—	—	2	58	2006	56 893	1 461 498
$3_1^m\#3_1^m$	—	—	—	2	58	2006	56 893	1 461 498
6_1^m	—	—	—	1	34	1267	38 199	1 015 996
6_1	—	—	—	1	34	1267	38 199	1 015 996
$3_1\#4_1$	—	—	—	—	8	516	20 458	648 362
$3_1^m\#4_1$	—	—	—	—	8	516	20 458	648 362

Summary (for small enough knot pictures)

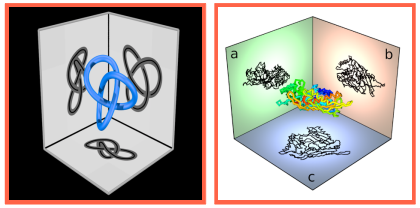
Step 1 Take a picture with your phone, a CNN gives the PD presentation

Step 2 An algorithm computes the Jones polynomial

Step 3 This determines the knot with probability 1

- **"Problem"** The Jones polynomial is not a very good invariant
- **Solution** This doesn't matter as the distribution of knot diagrams is biased towards "easy knots" – and the polynomial is good on these!
- This is **Step 3**

Motivation: detect knots via images



- Knot = closed string (a circle S^1) in three spaces; link = multiple components
- Knots are studied by projections to the plane Shadows
- In math knot theory started in the early 20th century

It is crucial to identify the DNA knot

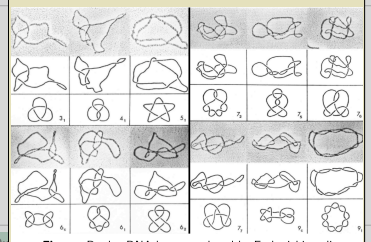
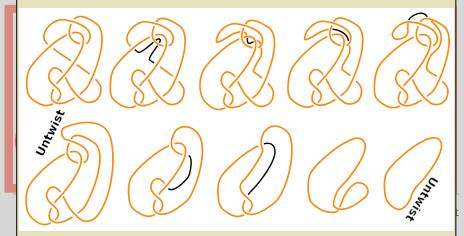


Figure: Duplex DNA knots produced by Escherichia coli Topoisomerase, via *Journal of Biological Chemistry* 1985

For example, difficult DNA knots prevent DNA from reproducing and you want that for things like viruses, or cancer cells

Even the unknotting problem is tricky

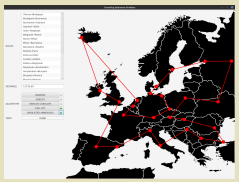


In general, the "standard invariants from algebraic topology" (homology and friends) are really not good for knots

above knots

Preparation: neural networks (NN) and friends

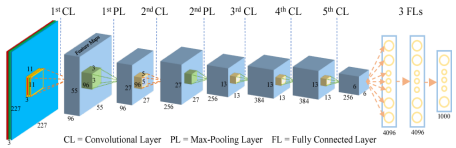
Key: Approximations to hard problems



NN are really good in finding approximate answers for difficult problems such as traveling salesperson (TSP)

- Task (rep… describe it?
- A NN is way to approximate an unknown function
- The specifics of how this works vary: there are many models and architectures

Preparation: neural networks (NN) and friends



- Task Classify images by detecting local patterns
- Key idea Convolutions detect features (edges, textures)
- CNN structure:
  (a) Convolutional layers: Extract features
  (b) Pooling layers: Downsample the image Ignore for today
  (c) Fully connected layers: Classify the image Ignore for today

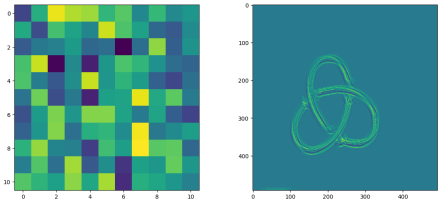
Idea (kernel, filter, etc.)

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

In image processing, a kernel, convolution matrix, or mask is a small matrix used for blurring, sharpening, embossing, edge detection, and more This is accomplished by doing a convolution between the kernel and an image

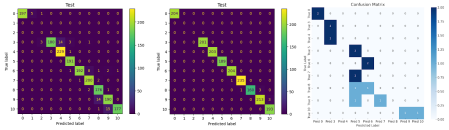
Results

What does the CNN actually do?



Results

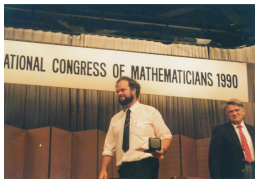
Comparison



(a) Vanilla, acc.≈95.74% (b) CNN, acc.≈99.50% (c) GPT, acc.≈50.00%

Figure: Confusion matrices on test dataset

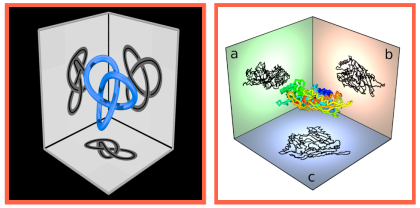
Next steps? Identify knots using the quantum invariants



- Above Jones receives the fields medal (also for the Jones polynomial)
- From the PD presentation one can compute the Jones polynomial or friends (we have a program that can do this fast for 100 crossing knots)
- This is Step 2

There is still much to do...

Motivation: detect knots via images



- Knot = closed string (a circle S^1) in three spaces; link = multiple components
- Knots are studied by projections to the plane Shadows
- In math knot theory started in the early 20th century

It is crucial to identify the DNA knot

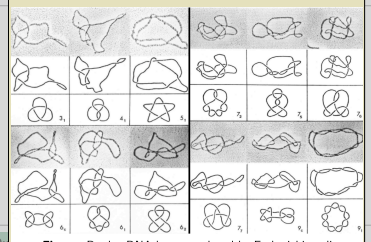
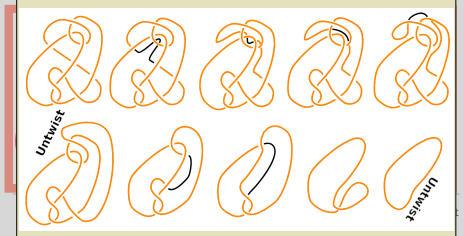


Figure: Duplex DNA knots produced by Escherichia coli Topoisomerase, via *Journal of Biological Chemistry* 1985

For example, difficult DNA knots prevent DNA from reproducing and you want that for things like viruses, or cancer cells

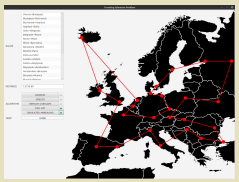
Even the unknotting problem is tricky



In general, the "standard invariants from algebraic topology" (homology and friends) are really not good for knots

Preparation: neural networks (NN) and friends

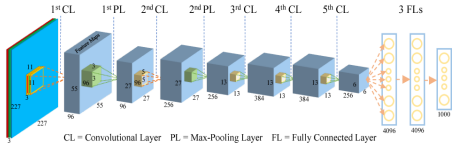
Key: Approximations to hard problems



NN are really good in finding approximate answers for difficult problems such as traveling salesperson (TSP)

- Task (rep… describe it?
- A NN is way to approximate an unknown function
- The specifics of how this works vary: there are many models and architectures

Preparation: neural networks (NN) and friends



- Task Classify images by detecting local patterns
- Key idea Convolutions detect features (edges, textures)
- CNN structure:
  - (a) Convolutional layers: Extract features
  - (b) Pooling layers: Downsample the image Ignore for today
  - (c) Fully connected layers: Classify the image Ignore for today

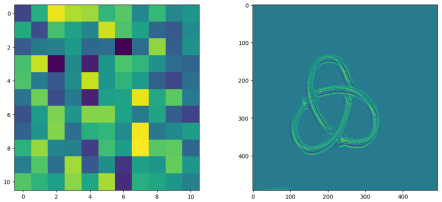
Idea (kernel, filter, etc.)

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

In image processing, a kernel, convolution matrix, or mask is a small matrix used for blurring, sharpening, embossing, edge detection, and more This is accomplished by doing a convolution between the kernel and an image

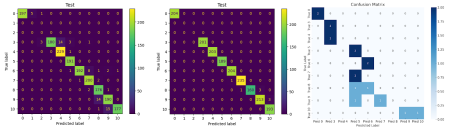
Results

What does the CNN actually do?



Results

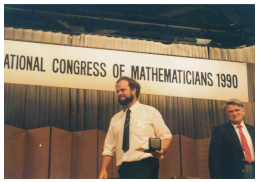
Comparison



(a) Vanilla, acc.≈95.74% (b) CNN, acc.≈99.50% (c) GPT, acc.≈50.00%

Figure: Confusion matrices on test dataset

Next steps? Identify knots using the quantum invariants



- Above Jones receives the fields medal (also for the Jones polynomial)
- From the PD presentation one can compute the Jones polynomial or friends (we have a program that can do this fast for 100 crossing knots)
- This is Step 2

Thanks for your attention!