Knot crossings counter

Yura Kabkov

Mentors
Anne Dranowski, Daniel Tubbenhauer

Yulia's Dream program

2025

What is NN?

A typical model architecture consists of a composition of functions, say $f_1, f_2, \ldots, f_N$. This composition is applied to the input data, and the result is processed to optimize the model parameters.

$$f_N \circ f_{N-1} \circ \cdots \circ f_1(input) = output$$

$$f(\mathbf{x}; \boldsymbol{\theta}) = f_N \circ f_{N-1} \circ \cdots \circ f_1(\mathbf{x}),$$

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

How do we train it?

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

$$L = L(predictions, truths) = L(f(truths), truths) = discrepancy$$

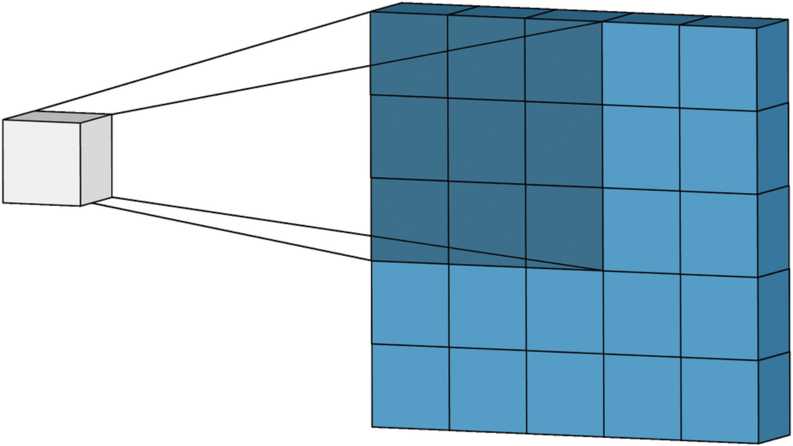
$$\min_{\boldsymbol{\theta} \in \text{dom } L} L \quad \Rightarrow \quad \boldsymbol{\theta}^{(l+1)} = \boldsymbol{\theta}^{(l)} - \eta \nabla L(\boldsymbol{\theta}^{(l)}), \qquad (1)$$

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 parameters.

Gradient Descent Algorithm

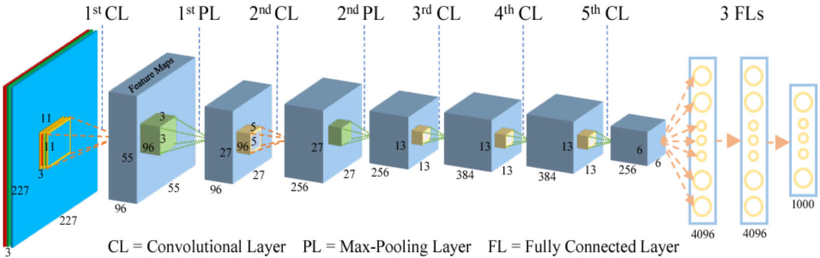
```
Algorithm 1: Gradient Descent

Input  : Training set of data points indexed by $n \in \{1, \ldots, N\}$
         Number of training steps $N$
         Loss function $L(\boldsymbol{\theta})$
         Learning rate $\eta$
         Initial parameter vector $\boldsymbol{\theta}$
Output: Final parameter vector $\boldsymbol{\theta}$

$n \leftarrow 1$;
repeat
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla L(\boldsymbol{\theta})$;
    $n \leftarrow n + 1$;
until $n > N$;
return $\boldsymbol{\theta}$
```

What is a convolution in ML?



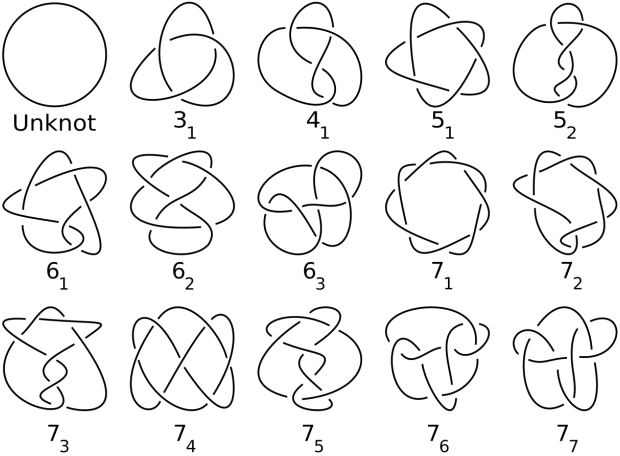
Source

Standart architecture



Source

Motivation



Source

State of the art results

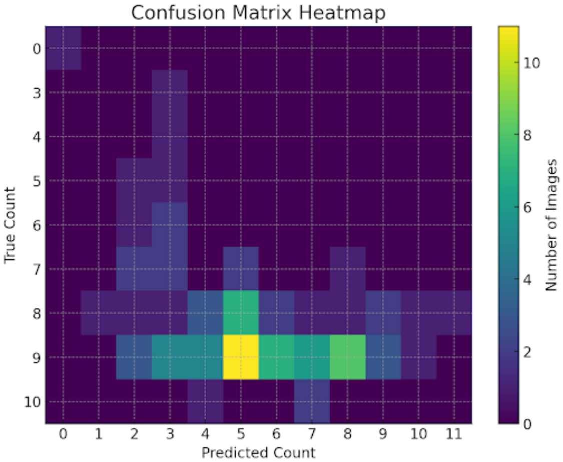
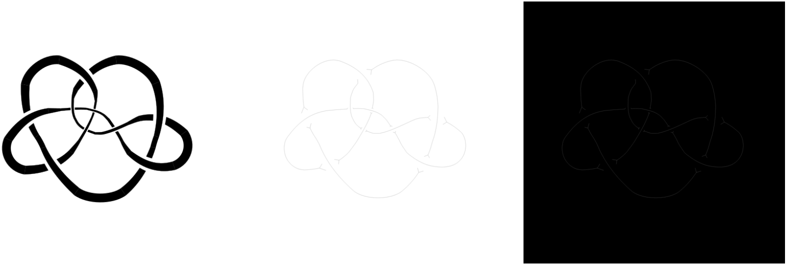


Figure: Confusion matrix of o4-mini results. Average accuracy is 5.65%

Data preparation



(a) Original knot

(b) Skeletonized

(c) Inverted

Figure: Steps of image preprocessing

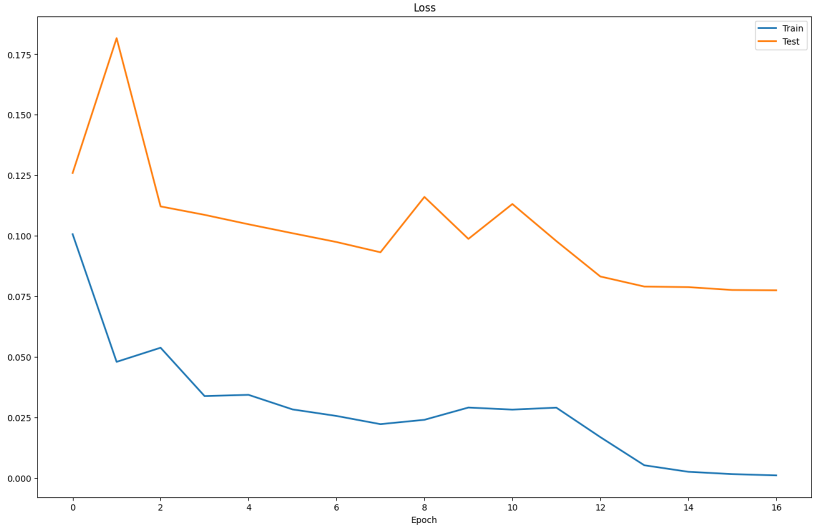
Vanilla

Easiest architecture is linear layer. But since composition of linears is also linear, one should insert non=linear function (in our case - ELU) between linear.

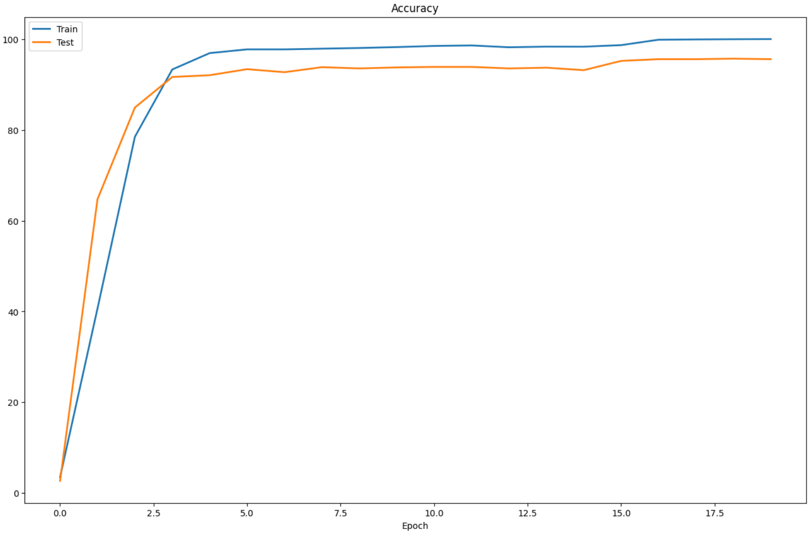
$$FC(\mathbf{x}) = ELU(\mathbf{Wx} + \mathbf{b})$$

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

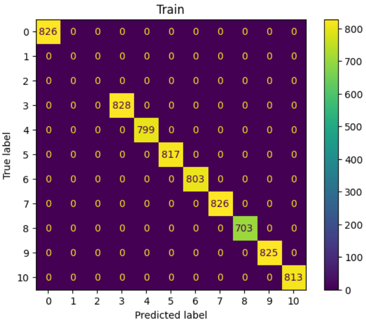
Vanilla, loss



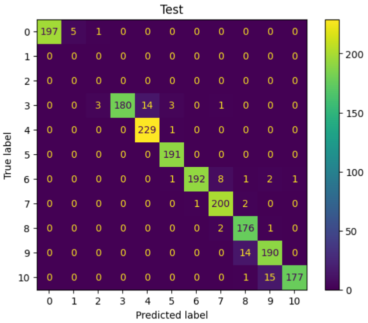
Vanilla, accuracy



Vanilla, confusion matrices



True \ Predicted	0	1	2	3	4	5	6	7	8	9	10
0	826	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	828	0	0	0	0	0	0	0
4	0	0	0	0	799	0	0	0	0	0	0
5	0	0	0	0	0	817	0	0	0	0	0
6	0	0	0	0	0	0	803	0	0	0	0
7	0	0	0	0	0	0	0	826	0	0	0
8	0	0	0	0	0	0	0	0	703	0	0
9	0	0	0	0	0	0	0	0	0	825	0
10	0	0	0	0	0	0	0	0	0	0	813



True \ Predicted	0	1	2	3	4	5	6	7	8	9	10
0	197	5	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	3	180	14	3	0	1	0	0	0
4	0	0	0	0	229	1	0	0	0	0	0
5	0	0	0	0	0	191	0	0	0	0	0
6	0	0	0	0	0	1	192	8	1	2	1
7	0	0	0	0	0	0	1	200	2	0	0
8	0	0	0	0	0	0	0	2	176	1	0
9	0	0	0	0	0	0	0	0	14	190	0
10	0	0	0	0	0	0	0	0	1	15	177

Vanilla, weights

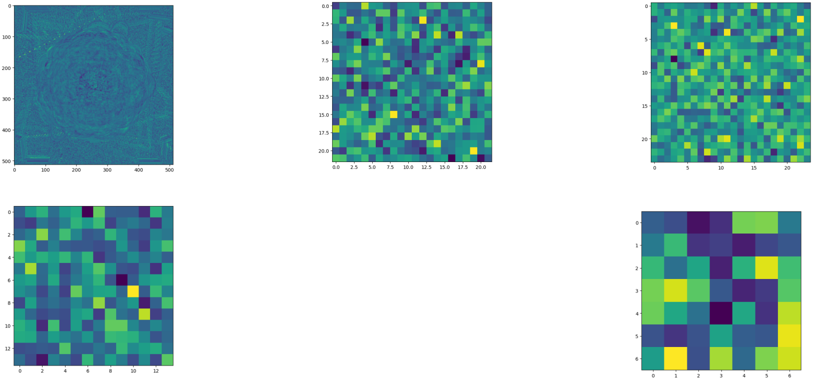
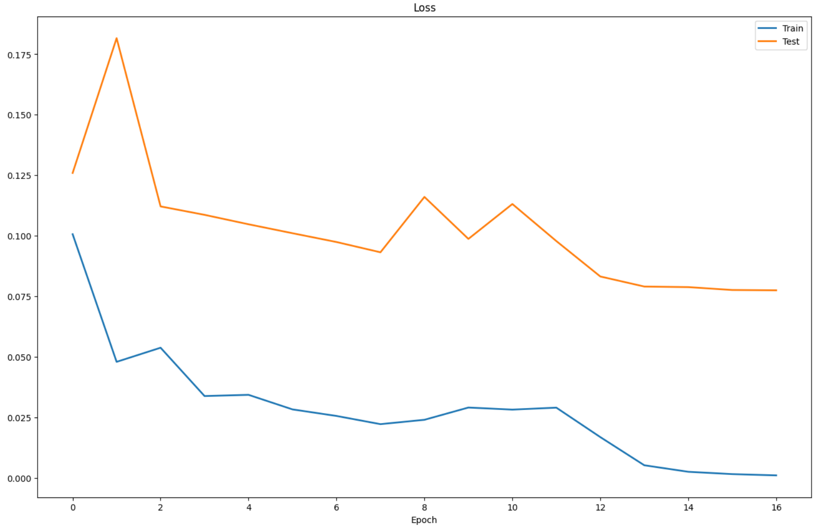
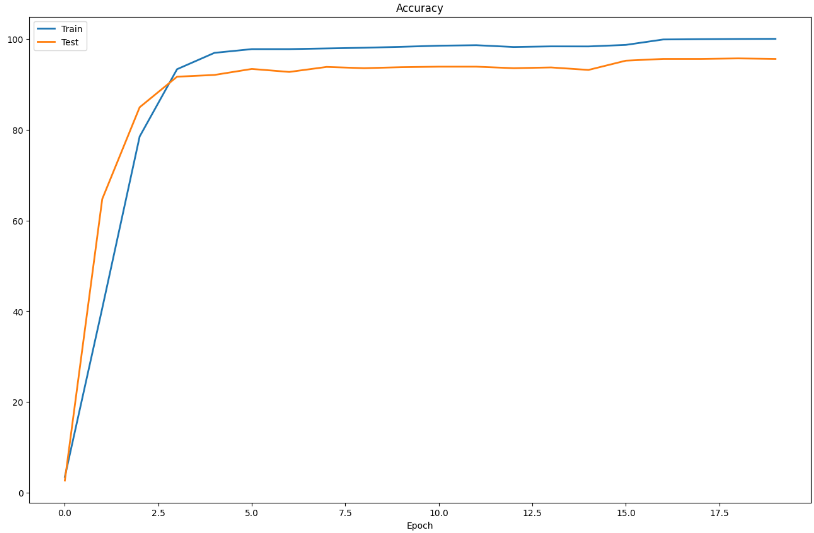


Figure: Weights of different Vanilla's layers

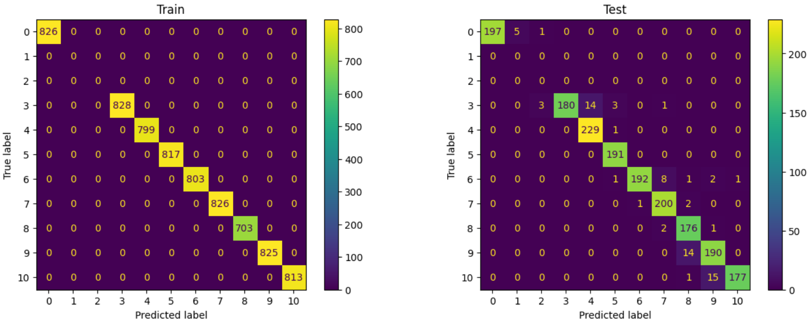
CNN, loss


Loss
Train
Test
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0
2
4
6
8
10
12
14
16
Epoch

CNN, accuracy



CNN, confusion matrices



Train

	0	1	2	3	4	5	6	7	8	9	10
0	826	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	828	0	0	0	0	0	0	0
4	0	0	0	0	799	0	0	0	0	0	0
5	0	0	0	0	0	817	0	0	0	0	0
6	0	0	0	0	0	0	803	0	0	0	0
7	0	0	0	0	0	0	0	826	0	0	0
8	0	0	0	0	0	0	0	0	703	0	0
9	0	0	0	0	0	0	0	0	0	825	0
10	0	0	0	0	0	0	0	0	0	0	813

Test

	0	1	2	3	4	5	6	7	8	9	10
0	197	5	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	3	180	14	3	0	1	0	0	0
4	0	0	0	0	229	1	0	0	0	0	0
5	0	0	0	0	0	191	0	0	0	0	0
6	0	0	0	0	0	1	192	8	1	2	1
7	0	0	0	0	0	0	1	200	2	0	0
8	0	0	0	0	0	0	0	2	176	1	0
9	0	0	0	0	0	0	0	0	14	190	0
10	0	0	0	0	0	0	0	0	1	15	177

CNN, weights

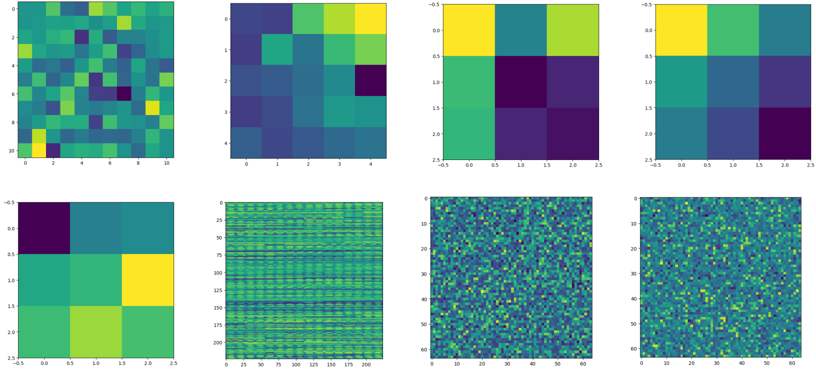
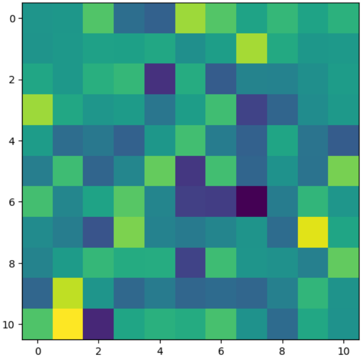
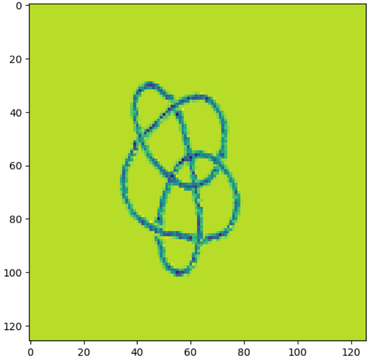


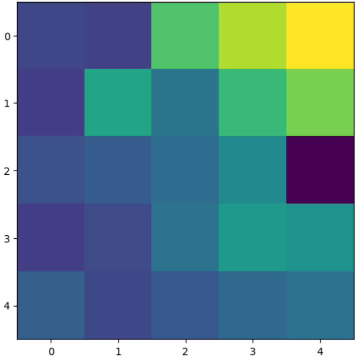
Figure: Weights of different CNN's layers

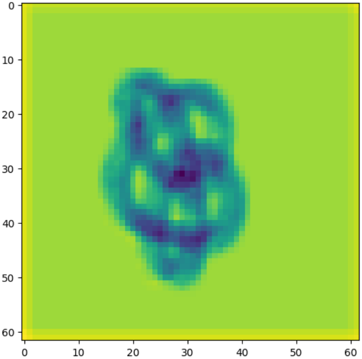
What does CNN actually do? 1



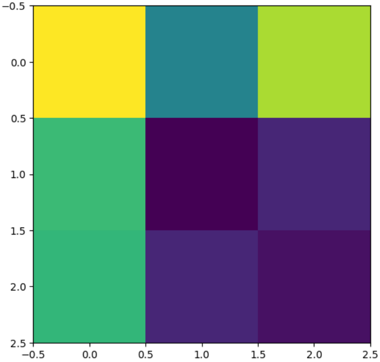


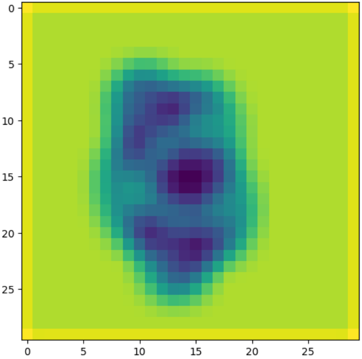
What does CNN actually do? 2



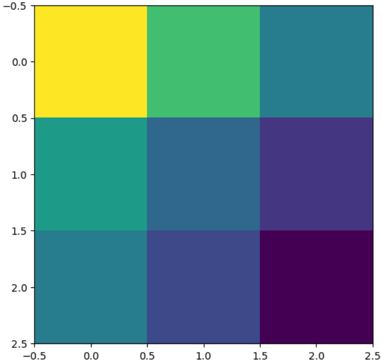


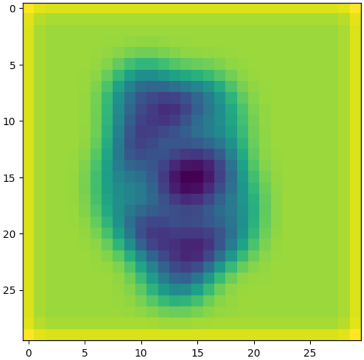
What does CNN actually do? 3



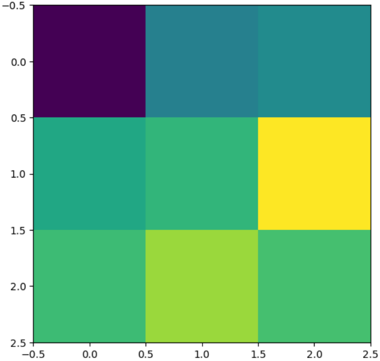


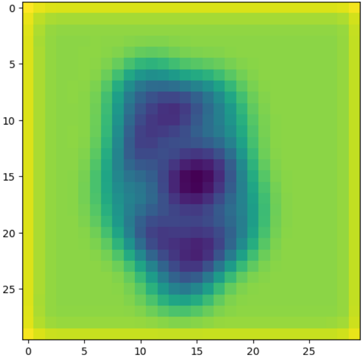
What does CNN actually do? 4



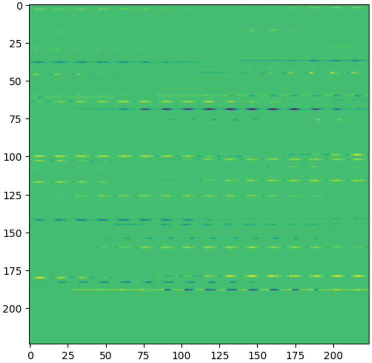


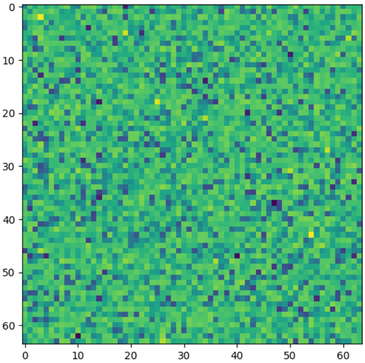
What does CNN actually do? 5



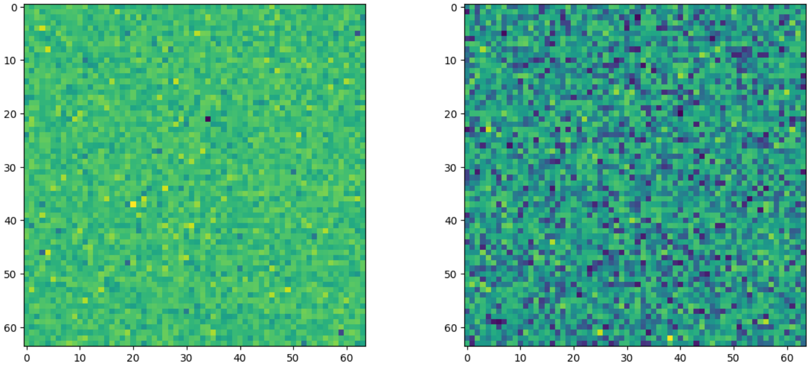


What does CNN actually do? 6

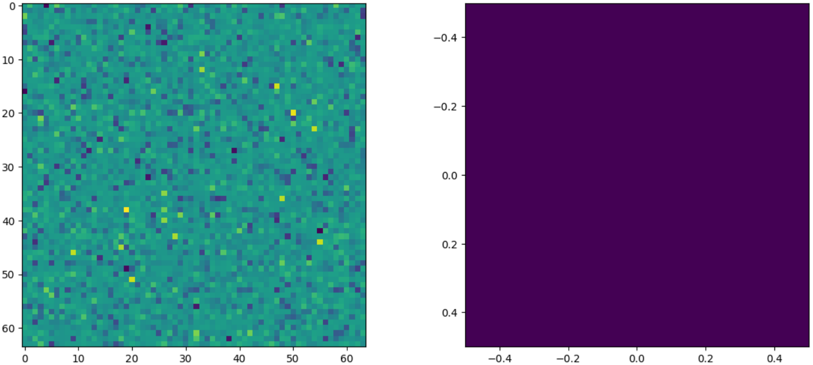




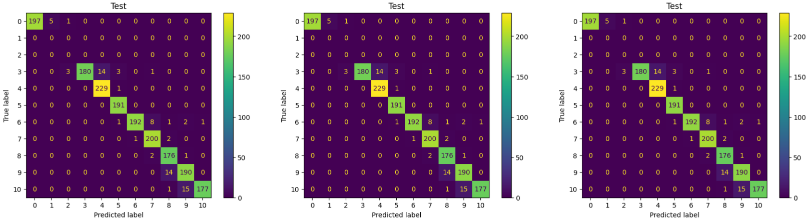
What does CNN actually do? 7



What does CNN actually do? 8



Comparison



	0	1	2	3	4	5	6	7	8	9	10
0	197	5	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	3	180	14	3	0	1	0	0	0
4	0	0	0	0	229	1	0	0	0	0	0
5	0	0	0	0	0	191	0	0	0	0	0
6	0	0	0	0	0	1	192	8	1	2	1
7	0	0	0	0	0	0	1	200	2	0	0
8	0	0	0	0	0	0	0	2	176	1	0
9	0	0	0	0	0	0	0	0	14	190	0
10	0	0	0	0	0	0	0	0	1	15	177

(a) Vanilla

	0	1	2	3	4	5	6	7	8	9	10
0	197	5	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	3	180	14	3	0	1	0	0	0
4	0	0	0	0	229	1	0	0	0	0	0
5	0	0	0	0	0	191	0	0	0	0	0
6	0	0	0	0	0	1	192	8	1	2	1
7	0	0	0	0	0	0	1	200	2	0	0
8	0	0	0	0	0	0	0	2	176	1	0
9	0	0	0	0	0	0	0	0	14	190	0
10	0	0	0	0	0	0	0	0	1	15	177

(b) CNN

	0	1	2	3	4	5	6	7	8	9	10
0	197	5	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	3	180	14	3	0	1	0	0	0
4	0	0	0	0	229	1	0	0	0	0	0
5	0	0	0	0	0	191	0	0	0	0	0
6	0	0	0	0	0	1	192	8	1	2	1
7	0	0	0	0	0	0	1	200	2	0	0
8	0	0	0	0	0	0	0	2	176	1	0
9	0	0	0	0	0	0	0	0	14	190	0
10	0	0	0	0	0	0	0	0	1	15	177

(c) GPT

Figure: Confusion matrices on test dataset

Thank you for listening!