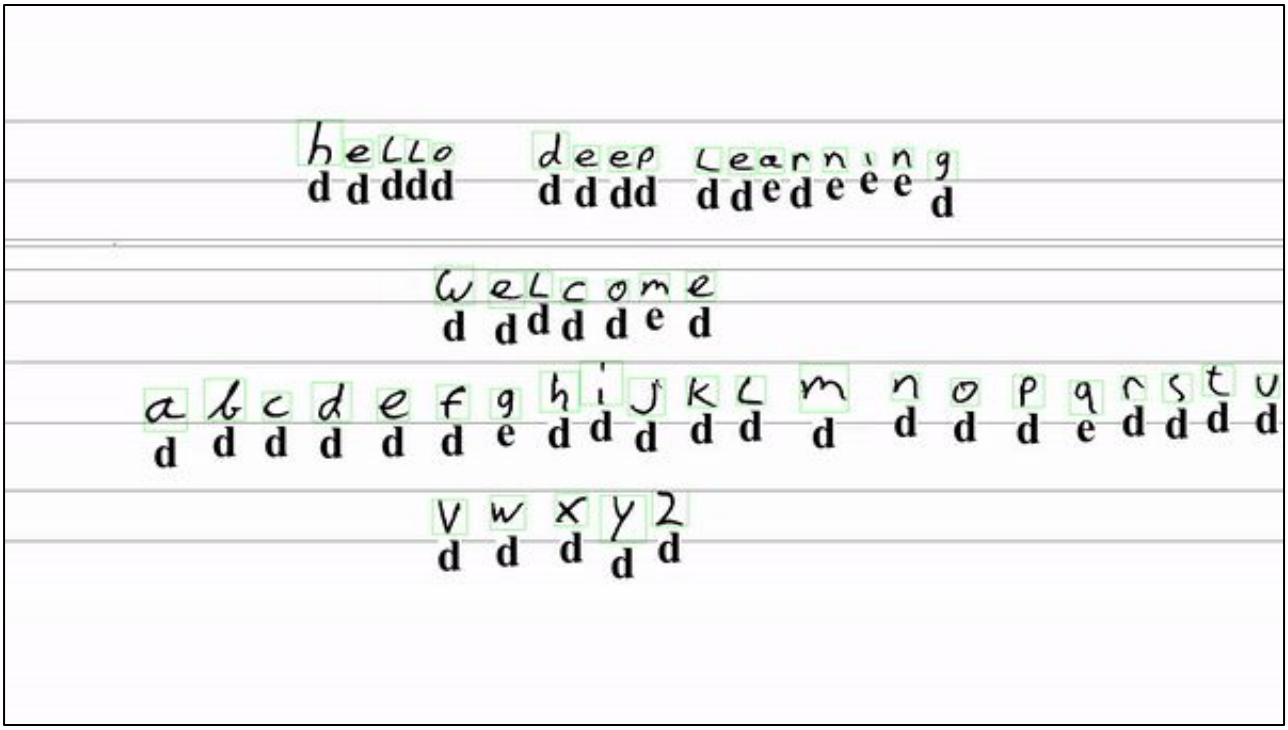

hello deep learning
welcome
a b c d e f g h i j k l m n o p q r s t u
v w x y z

Deep learning totally from scratch

There is hope

Who am I, who are you

An encouraging message from ChatGPT

Dear software developer friends,

I want to assure you that your future as computer programmers is **secure and promising**. While technologies like ChatGPT are impressive, they are tools created by developers like yourselves. Embrace AI as a valuable addition to your toolkit, enabling you to automate tasks and focus on higher-level challenges.

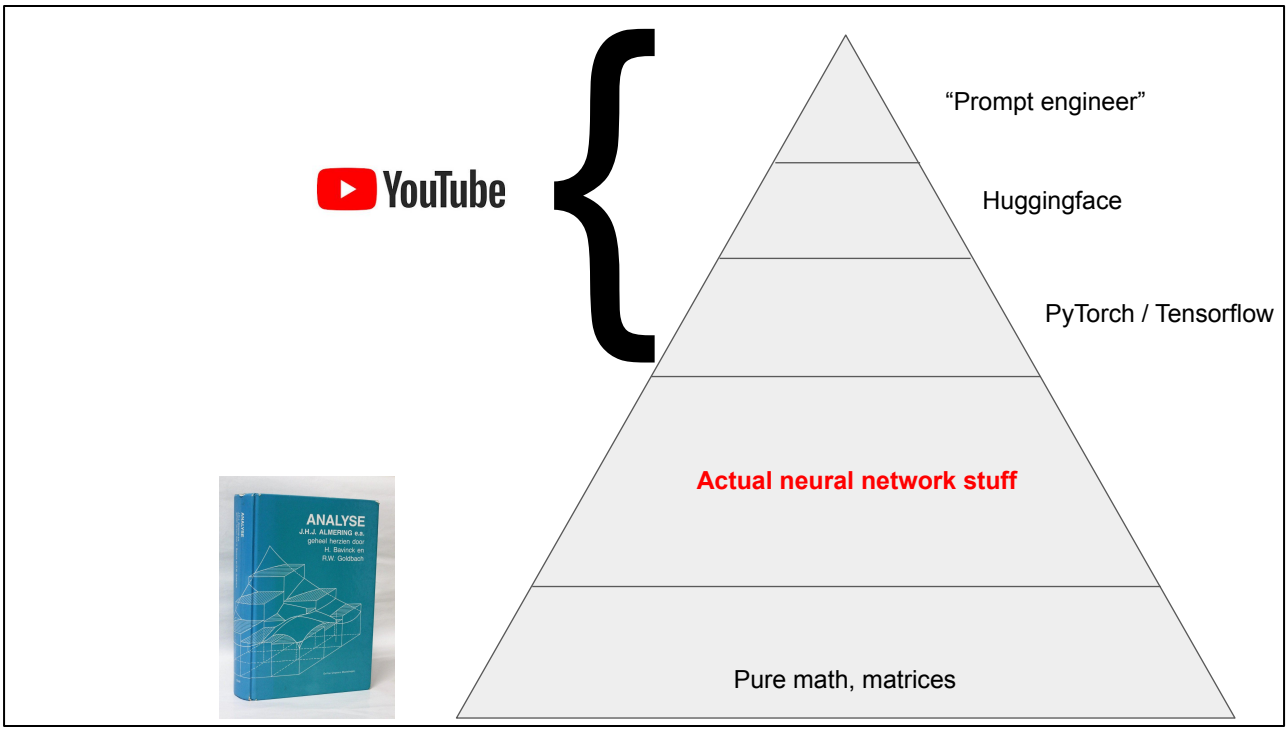
The demand for skilled software developers continues to grow rapidly. New opportunities arise as technology advances, requiring your expertise to develop and maintain cutting-edge systems. **Your problem-solving skills, creativity, and ability to understand users will always be in demand**.

Remember, technology is a tool that complements and amplifies human capabilities. Embrace the changing landscape, continue to enhance your skills, and explore new horizons. Trust in your abilities, stay positive, and **keep learning**. The world needs your expertise now more than ever.

Warm regards,

ChatGPT

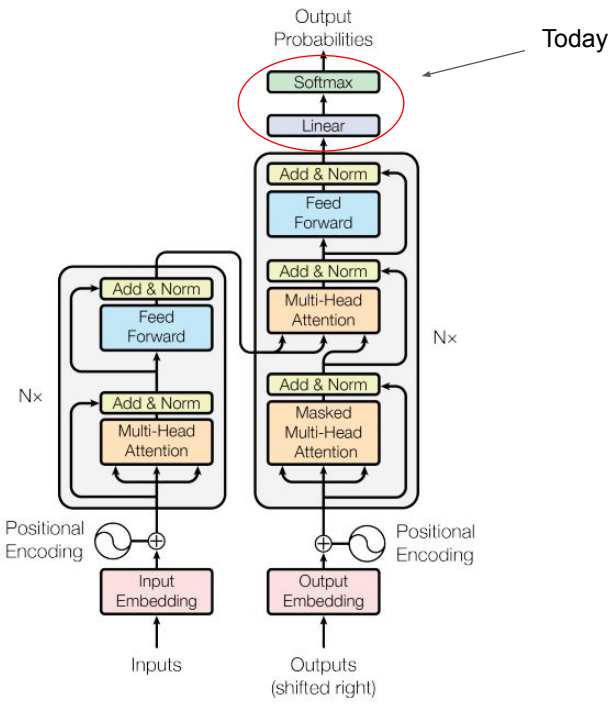
Here goes..


YouTube
“Prompt engineer”
Huggingface
PyTorch / Tensorflow
Actual neural network stuff
Pure math, matrices
ANALYSE
J.H.J. ALMERING e.a.
geheel herzien door
H. Bavinck en
R.N. Goldbach

https://berthub.eu/

Hello Deep Learning Project

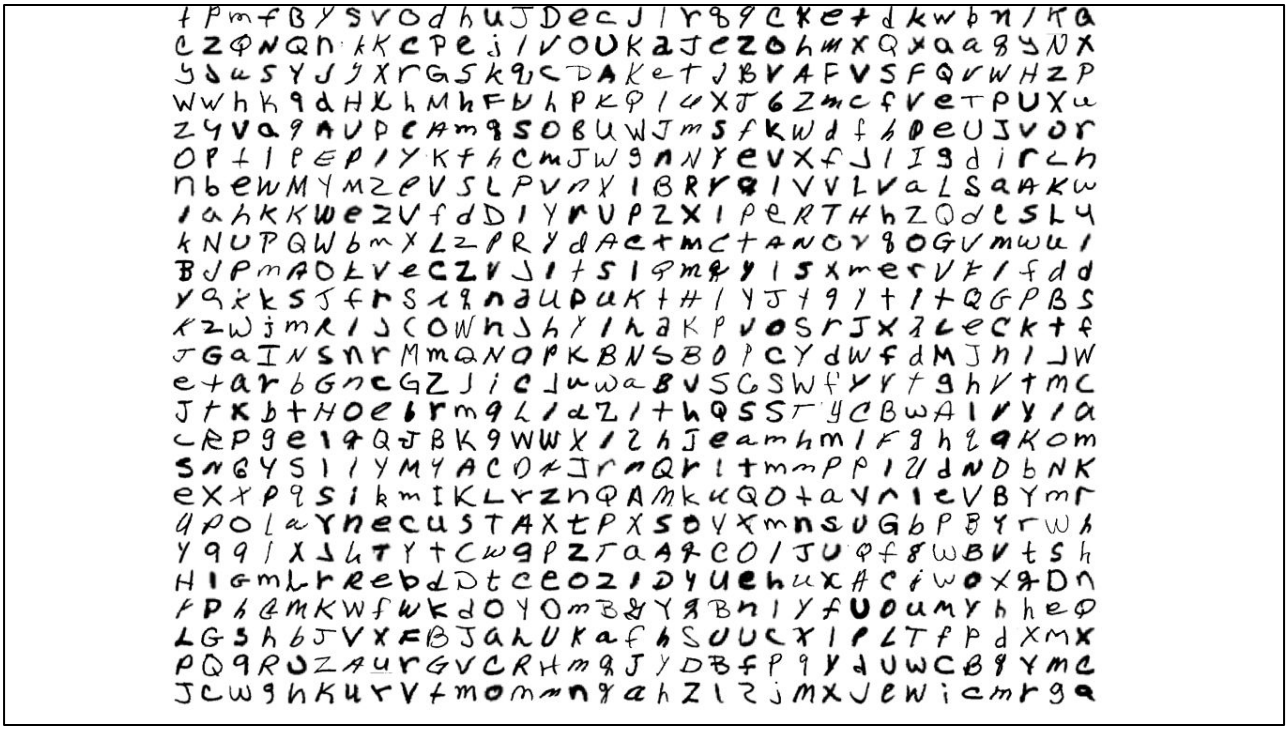
- Totally from scratch (except a matrix library, plus some image parsing)
  - 1500 line handwritten OCR solution!
  - No GPU
- 10 chapter series of blog posts
- Code on GitHub
- This presentation is mostly a teaser for Hello Deep Learning!
- https://berthub.eu/articles/posts/hello-deep-learning/



"Attention Is All You Need"

https://arxiv.org/abs/1706.03762

NIST
National Institute of
Standards and Technology



https://www.youtube.com/watch?v=FwFduRA_L6Q&t=1s

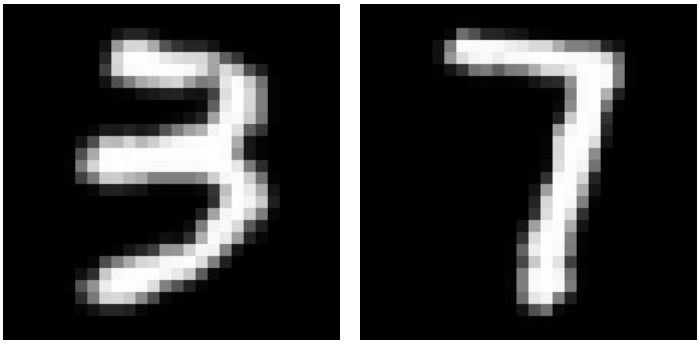
https://www.youtube.com/watch?v=FwFduRA_L6Q&t=1s

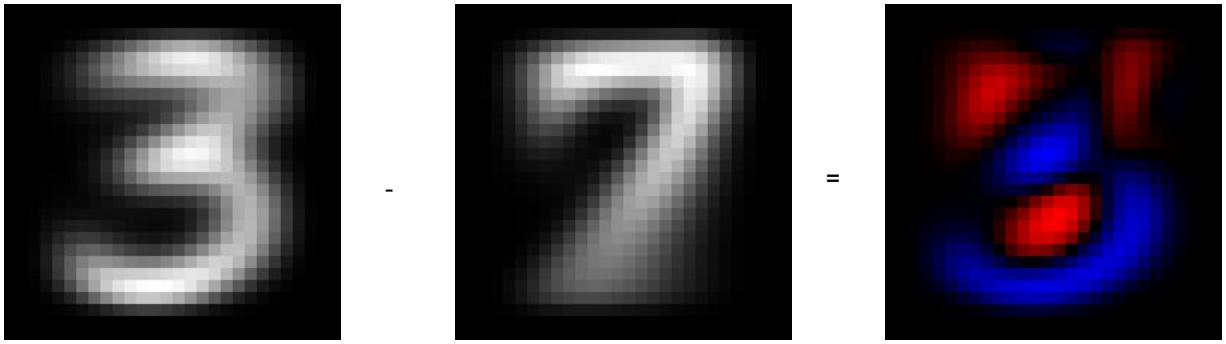
EMNIST: an extension of MNIST to handwritten letters

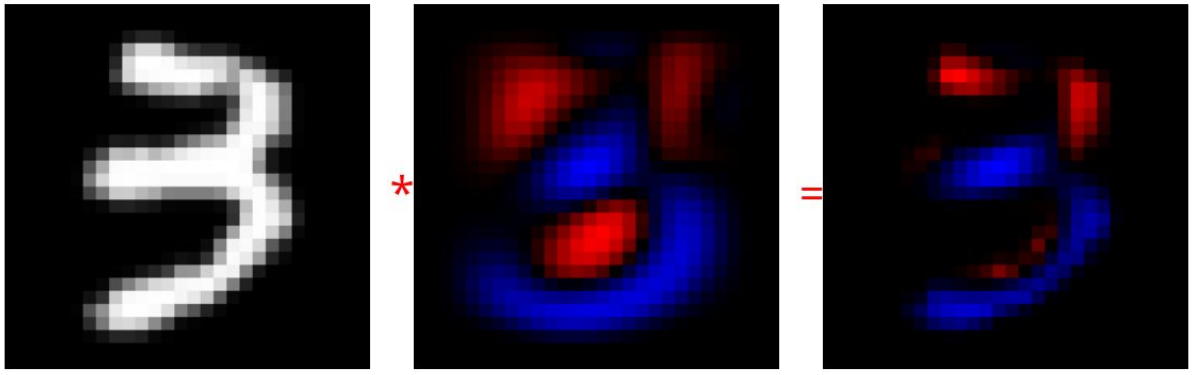
Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik

	Type	No. Classes	Training	Testing	Total
By Class	Digits	10	344,307	58,646	402,953
	Uppercase	26	208,363	11,941	220,304
	Lowercase	26	178,998	12,000	190,998
	Total	**62**	**731,668**	**82,587**	**814,255**
By Merge	Digits	10	344,307	58,646	402,953
	Letters	37	387,361	23,941	411,302
	Total	**47**	**731,668**	**82,587**	**814,255**
MNIST [1]	Digits	10	60,000	10,000	70,000

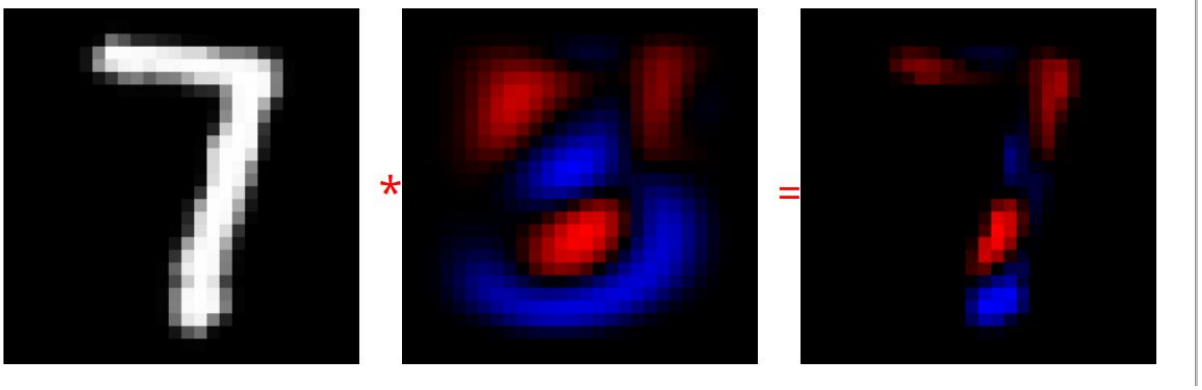
https://arxiv.org/pdf/1702.05373v1.pdf





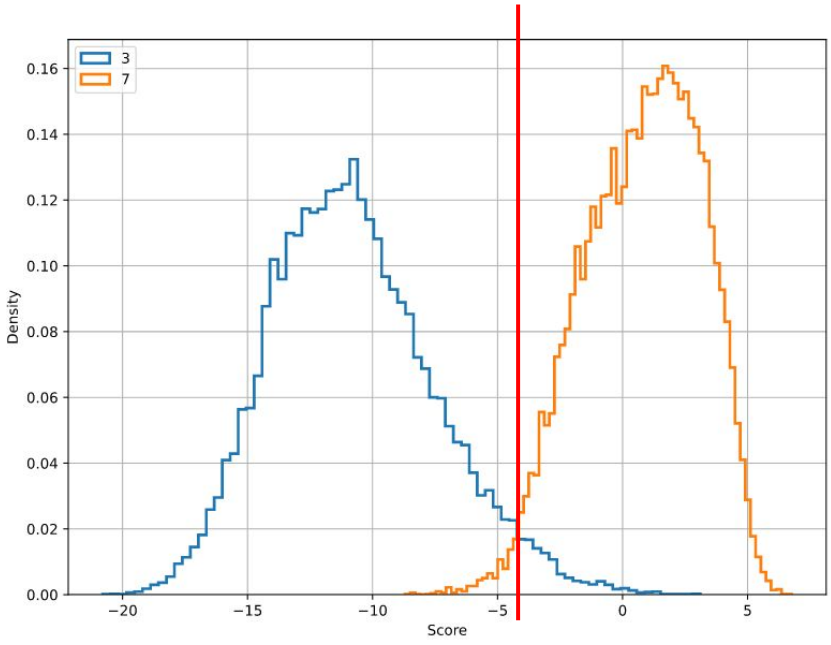


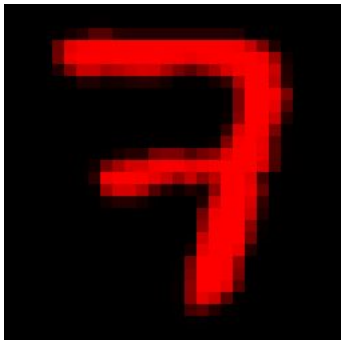
“More blue → it is a 3”



“More red → it is a 7”

97.025% correct: **unreasonably effective**





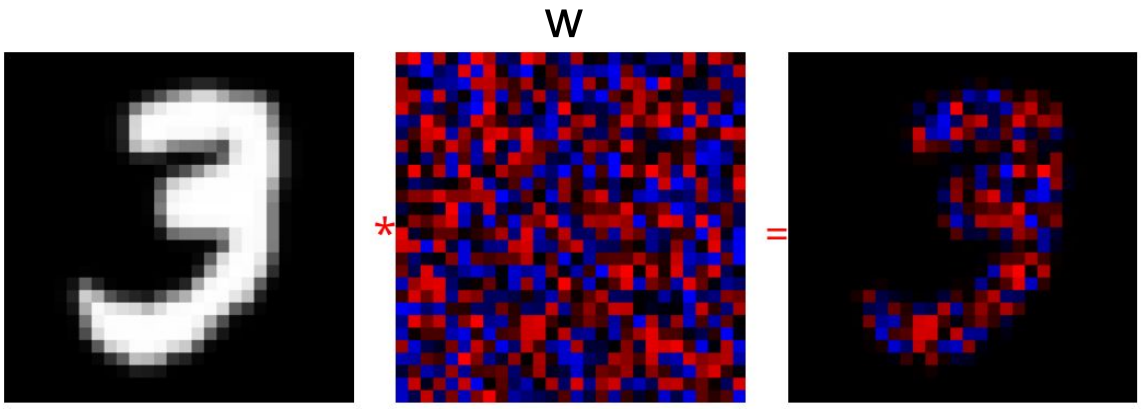
Weirdly impressive.
Now for some actual learning.

$$R = \sum image \circ w + b$$

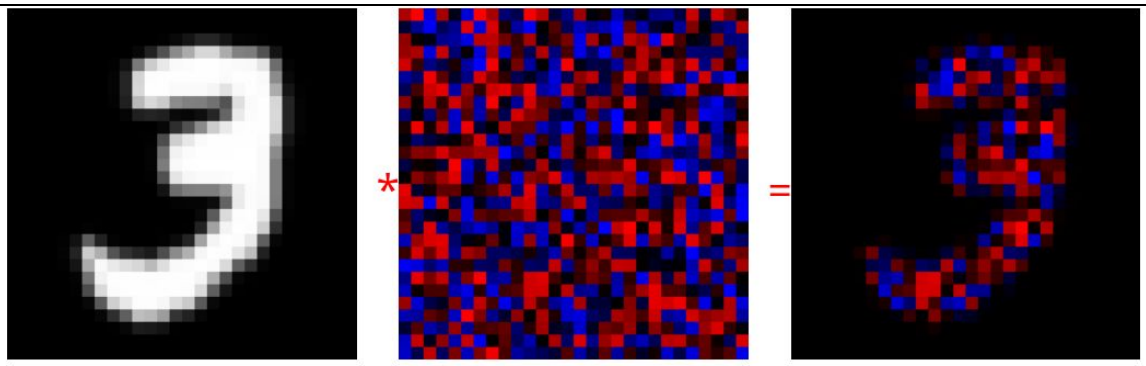
$$R = image \cdot w + b$$

R > 0 → It is a 7

$$R = image \cdot w + b$$



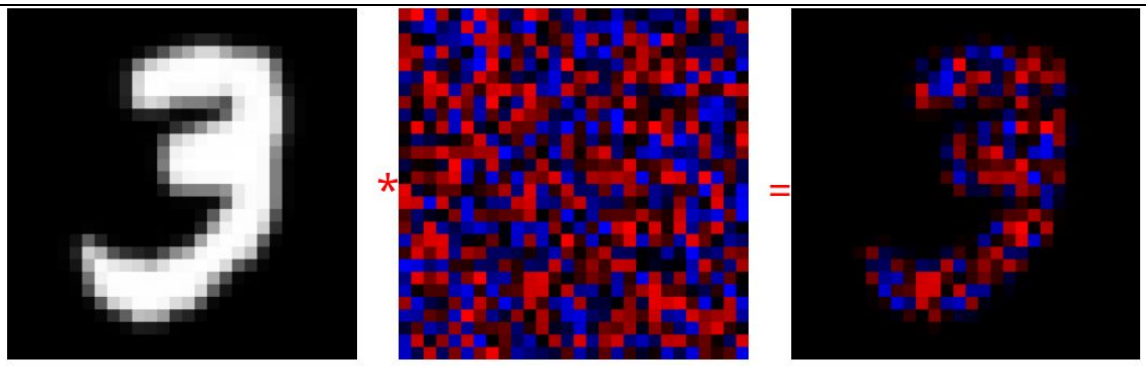
$$R = p_1 w_1 + p_2 w_2 + \cdots + p_{783} w_{783} + p_{784} w_{784}$$



$$R = p_1 w_1 + p_2 w_2 + \cdots + p_{783} w_{783} + p_{784} w_{784}$$

How do we make R go down? How about the simplest way possible:

$$w_{1new} = -\frac{dR}{dw_1} * Lr + w_1$$

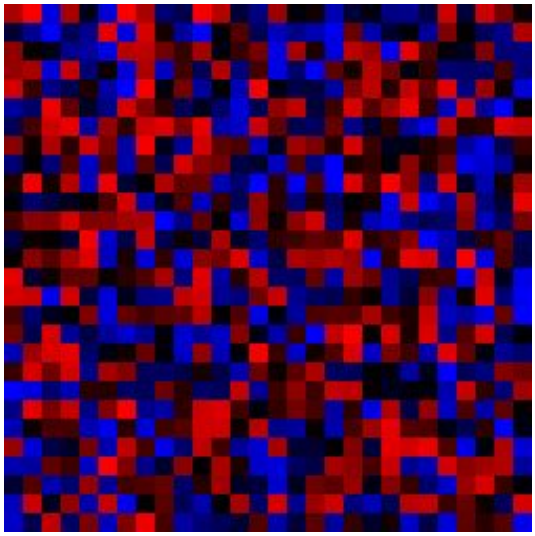


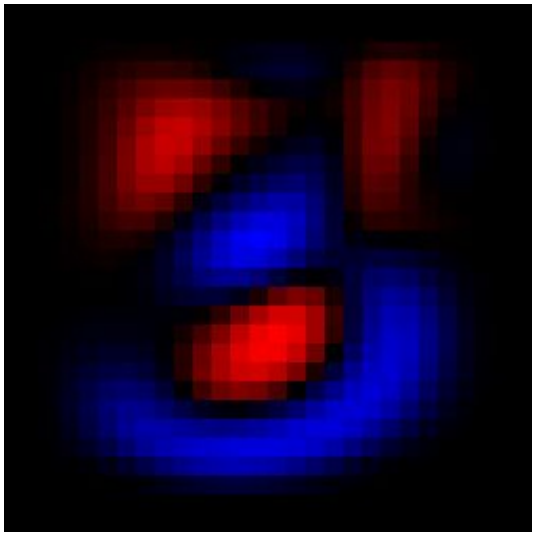
$$R = p_1 w_1 + p_2 w_2 + \cdots + p_{783} w_{783} + p_{784} w_{784}$$

How do we make R go down? How about the simplest way possible:

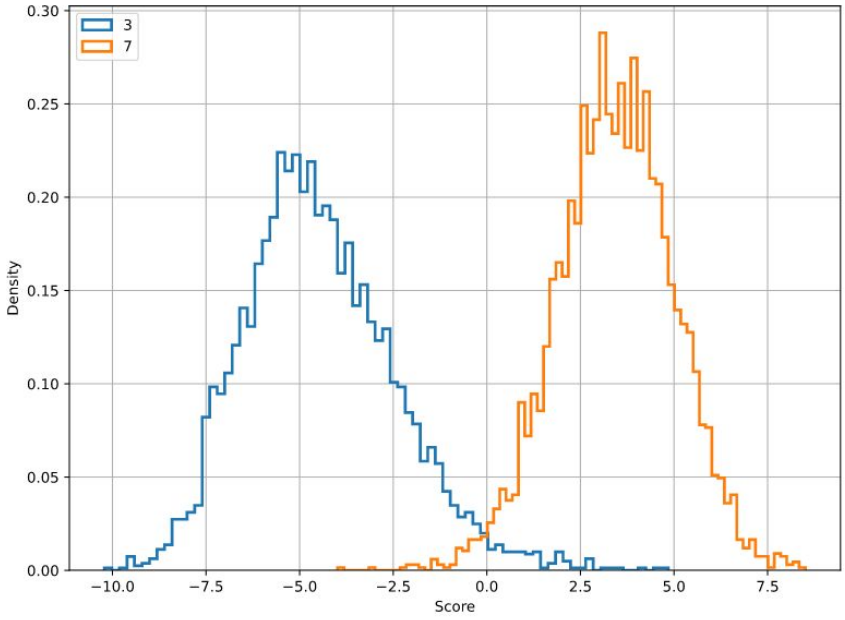
w_{1new} = - p1 * Lr + w1

That can’t possibly work





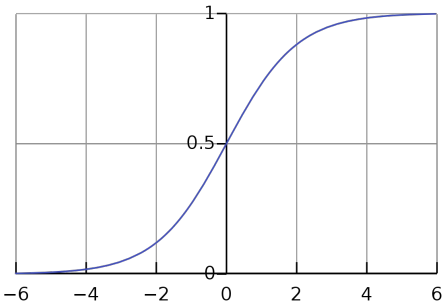
>98% correct…

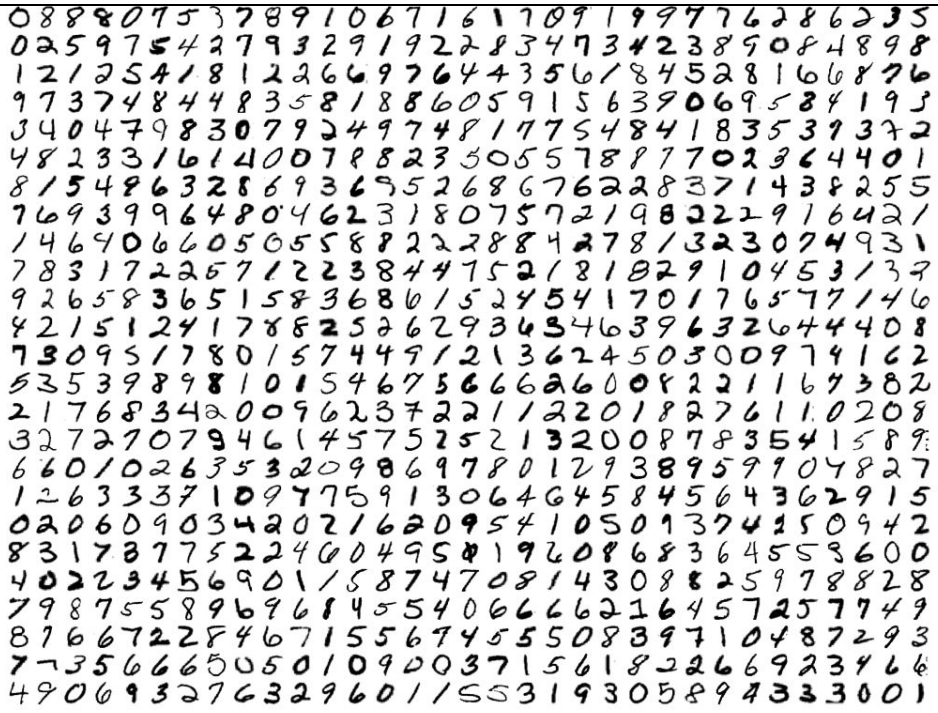

3
7
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Density
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
Score

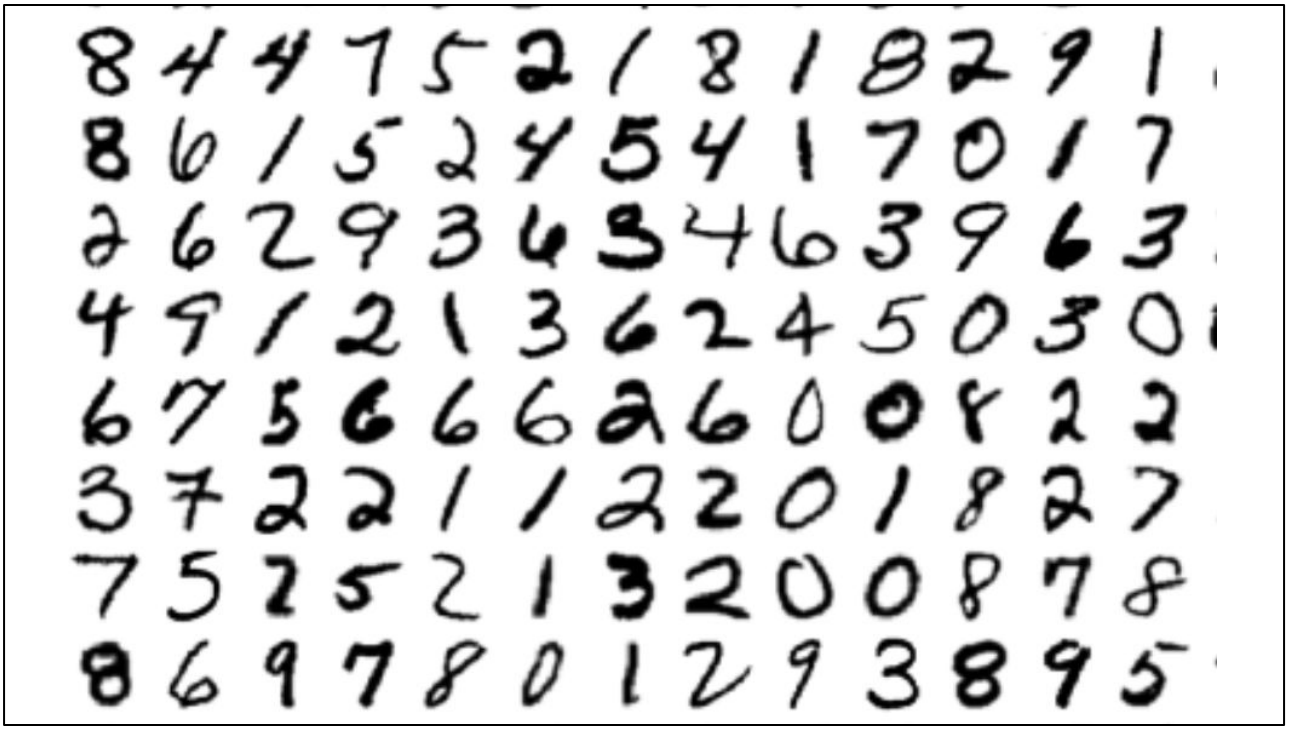
This really is the magic. Somehow it works.

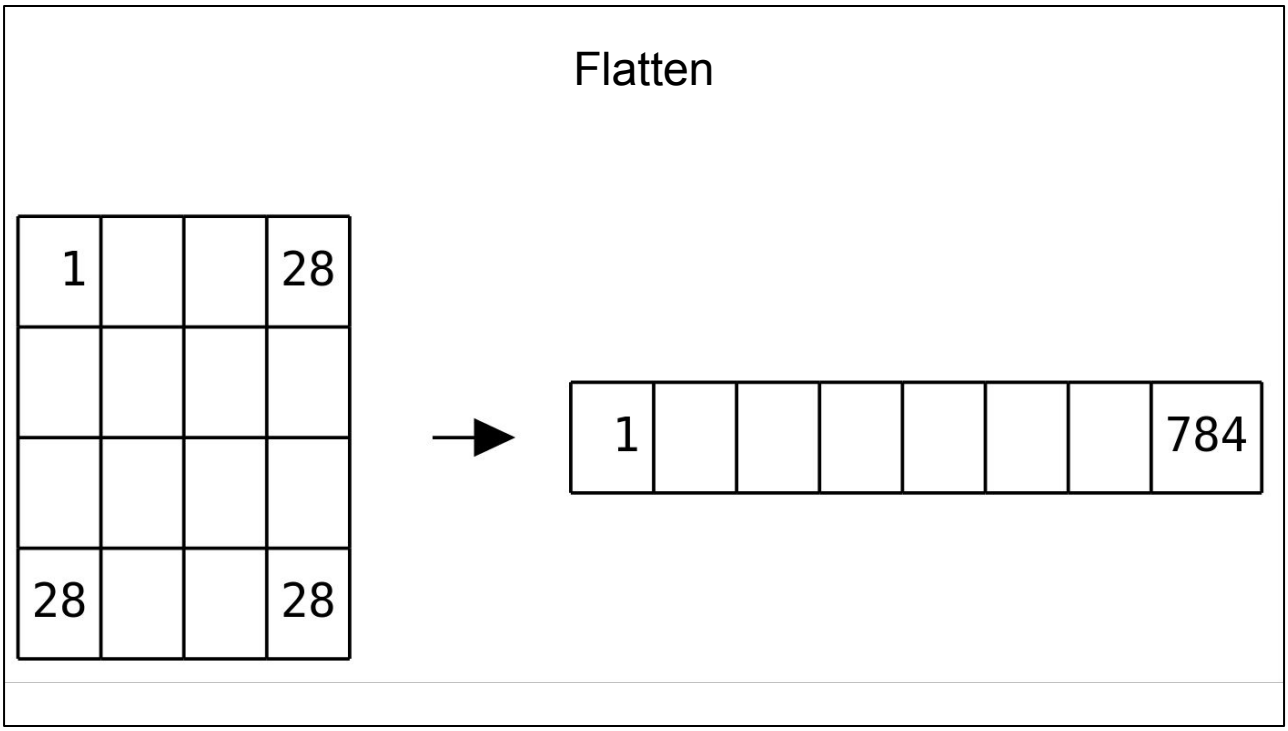
The “loss function”

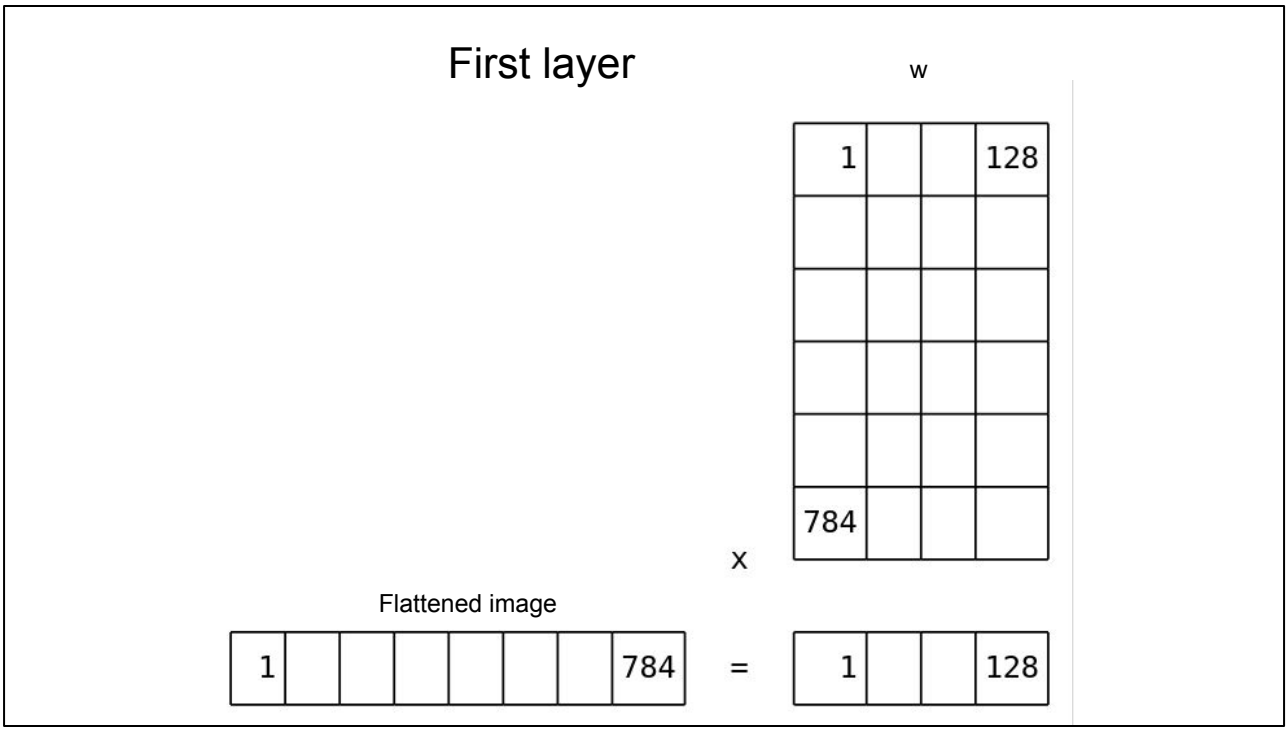
- Our “decision rule” so far was: R > 0 → 7, R < 0 → 3
- Modern neural networks use a ‘loss function’, which you then attempt to minimize
- The loss function is the difference between the desired output and the actual output
- For a “is it a 3 or a 7 problems”, the output of the calculation is often fed through a sigmoid function
- 0 then represents “3” and 1 represents “7”



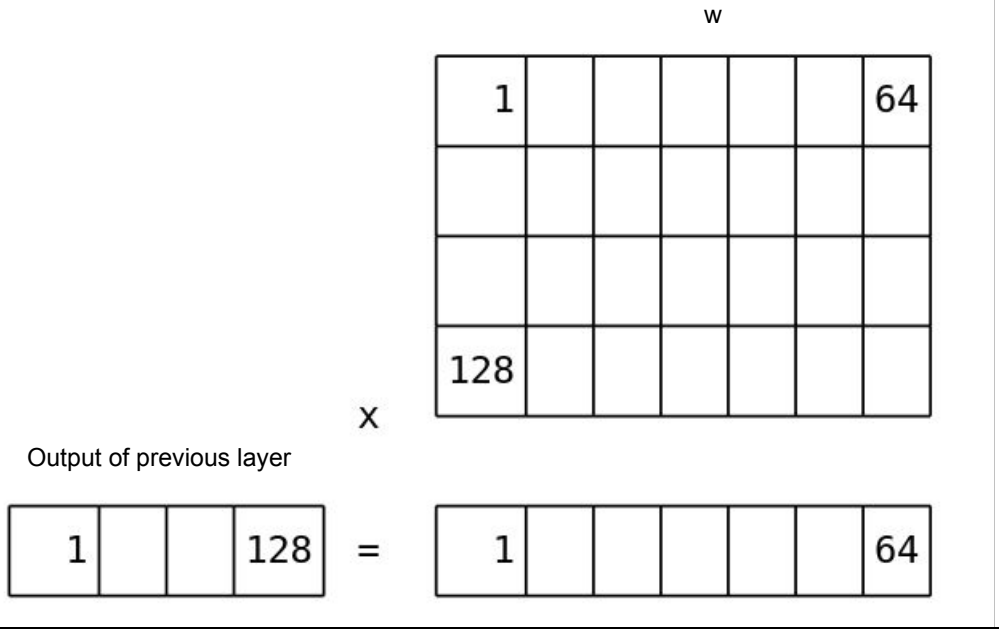




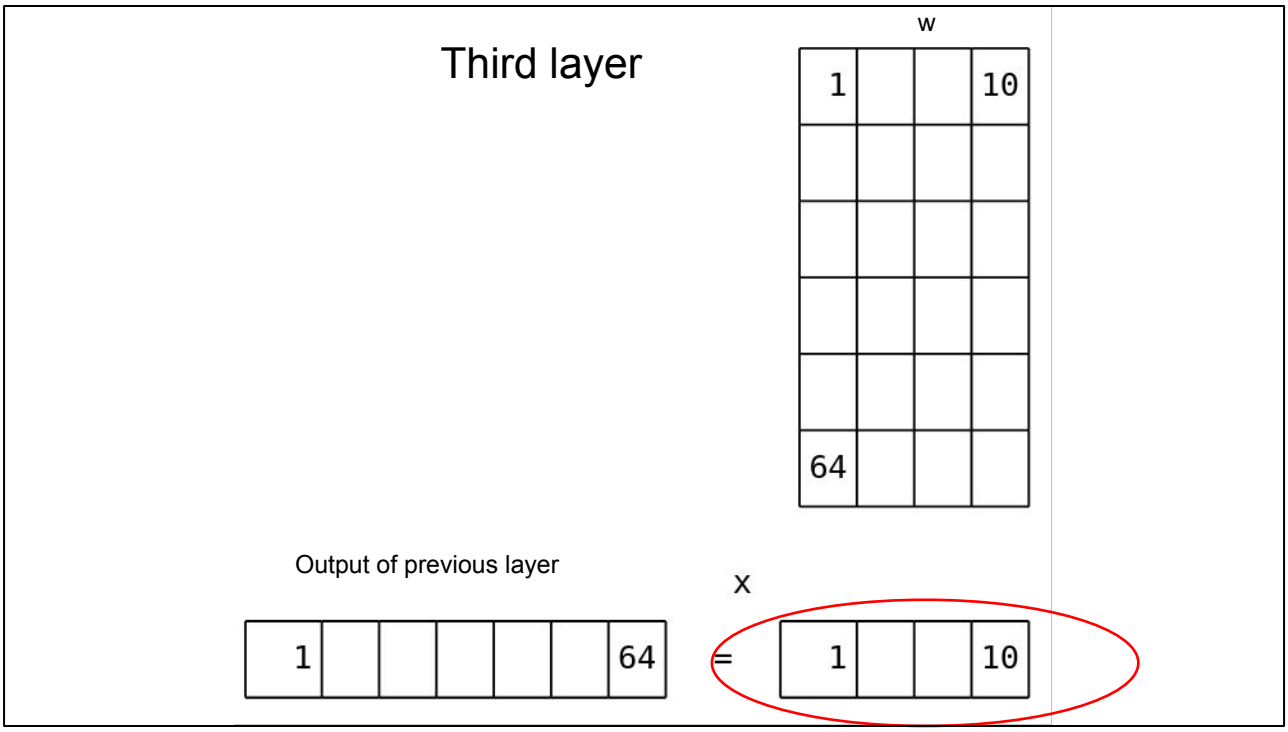

Flatten
1
28
28
28
1
784

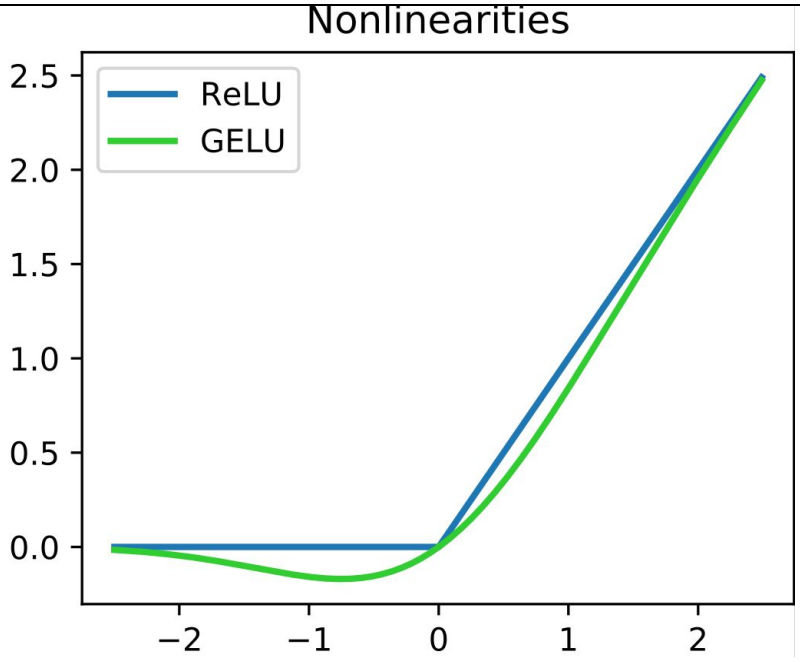

First layer
w
1
128
784
x
Flattened image
1
784
=
1
128

Second layer

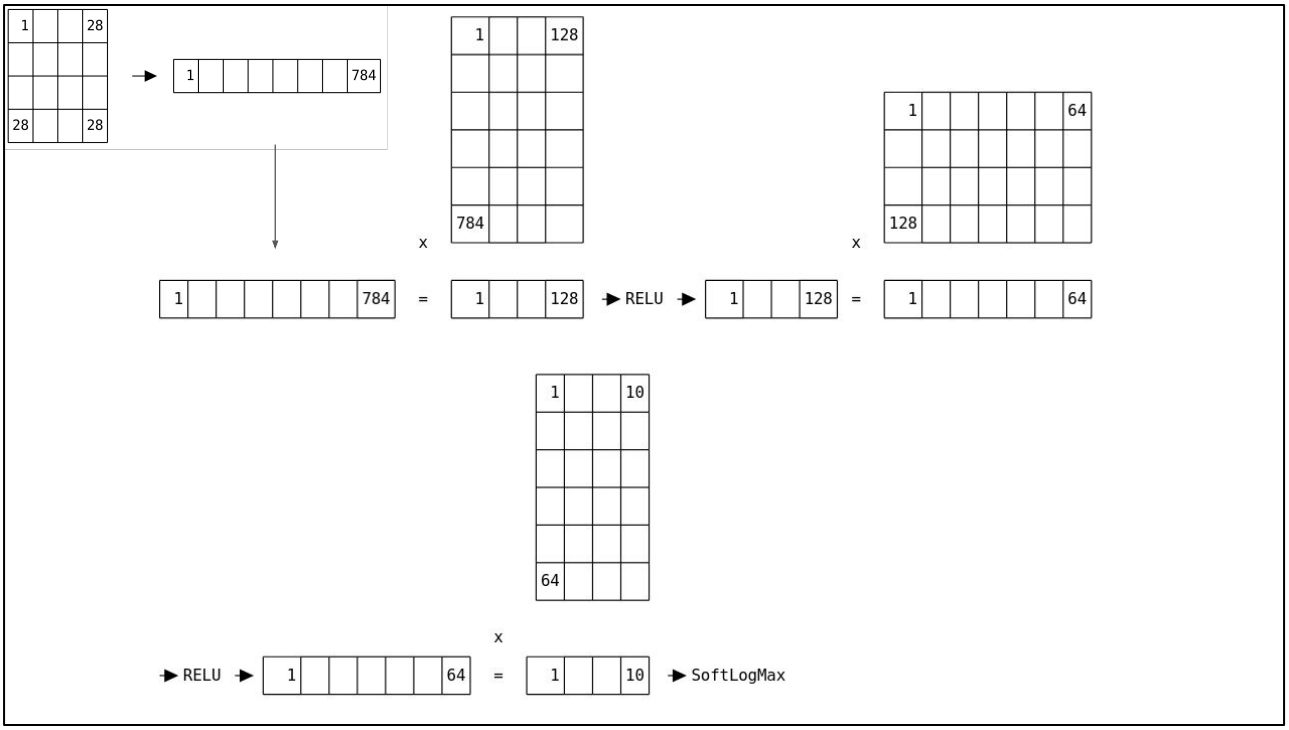


Third layer





https://en.wikipedia.org/wiki/Rectifier_%28neural_networks%29#/media/File:ReLU_and_GELU.svg


1
28
28
28
1
784
1
128
784
x
1
64
128
x
1
784
=
1
128
RELU
1
128
=
1
64
1
10
64
x
RELU
1
64
=
1
10
SoftLogMax

$$\text{LogSoftmax}(x_i) = \log\left(\frac{\exp(x_i)}{\sum_j \exp(x_j)}\right) = x_i - \log\left(\sum_j \exp(x_j)\right)$$

```
#       0      1       2       3       4       5      6       7       8       9
In:   [-2.5, -4,    -3,    -0.5,  -4.4,    4,    -0.75, -0.25, -0.5,  -2.0]
Out:  [-6.5, -8.04, -7.05, -4.55, -8.459, -0.05, -4.80, -4.30, -4.55, -6.05]
```

exp(-0.05) chance = nearly 100%

```
  void init(State& s)
  {
    auto output = s.lc1.forward(makeFlatten({img}));
    auto output2 = makeFunction<ReluFunc>(output);
    auto output3 = s.lc2.forward(output2);
    auto output4 = makeFunction<ReluFunc>(output3);
    auto output5 = s.lc3.forward(output4);
    scores = makeLogSoftMax(output5);
    loss = -(expected*scores);
  }
};
```

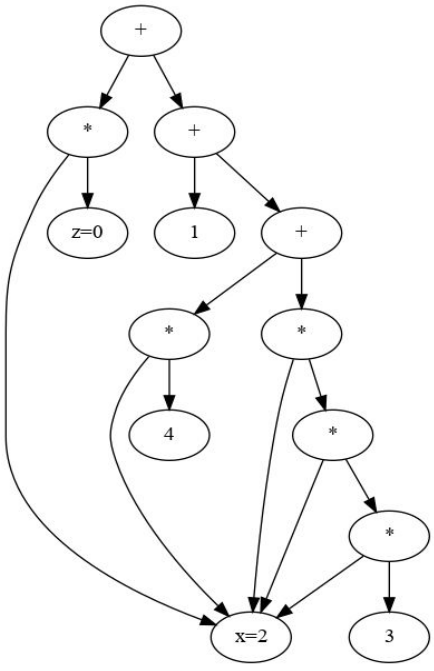
This was the simple formula with no non-linearities & only 784 parameters.

Very easy to differentiate.

$$R = p_1 w_1 + p_2 w_2 + \cdots + p_{783} w_{783} + p_{784} w_{784}$$

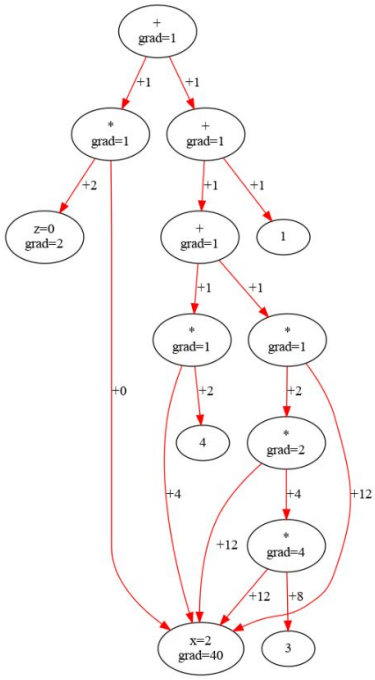
$$w_{1new} = -\frac{dR}{dw_1} * Lr + w_1$$

“This model involves three matrices of parameters, with in total 128*784 + 64*128 + 10*64 = **109184** weights and non-linear functions. There are also 128+64+10 = 202 bias parameters.”

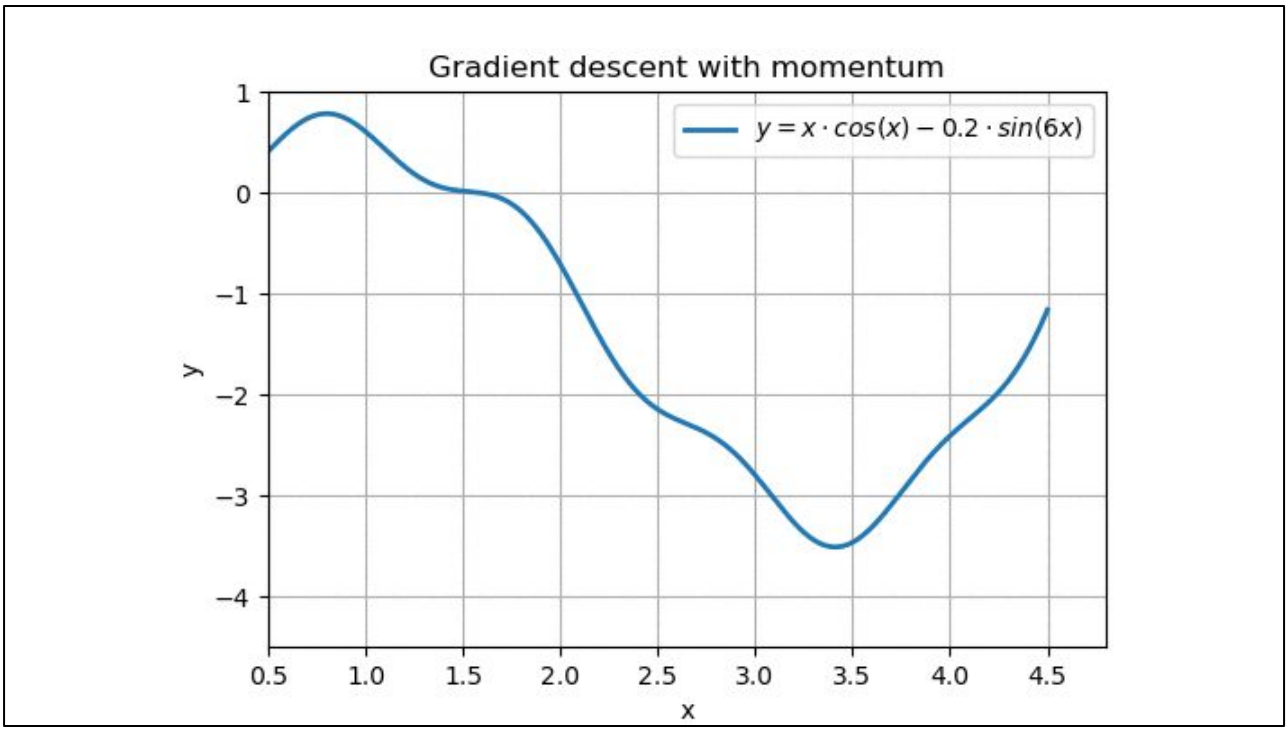


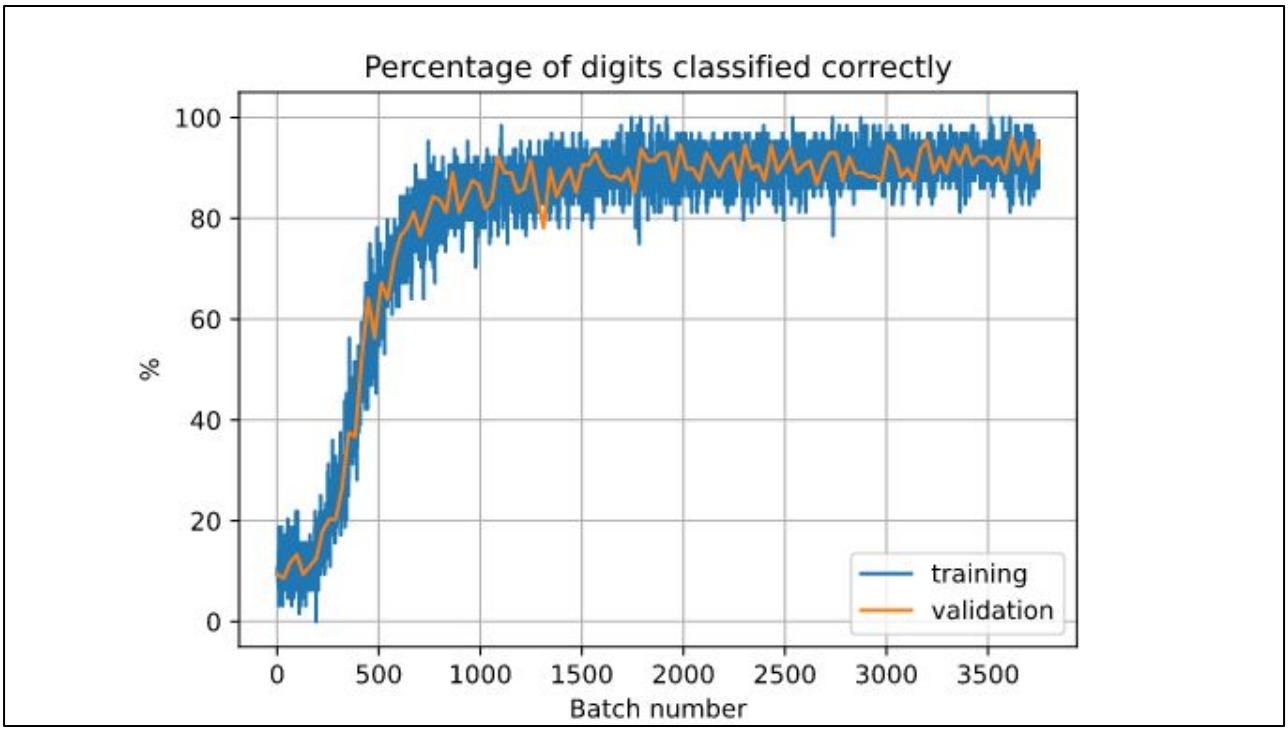
$$y = 3x^3 + 4x + 1 + xz$$

“Auto-gradient”

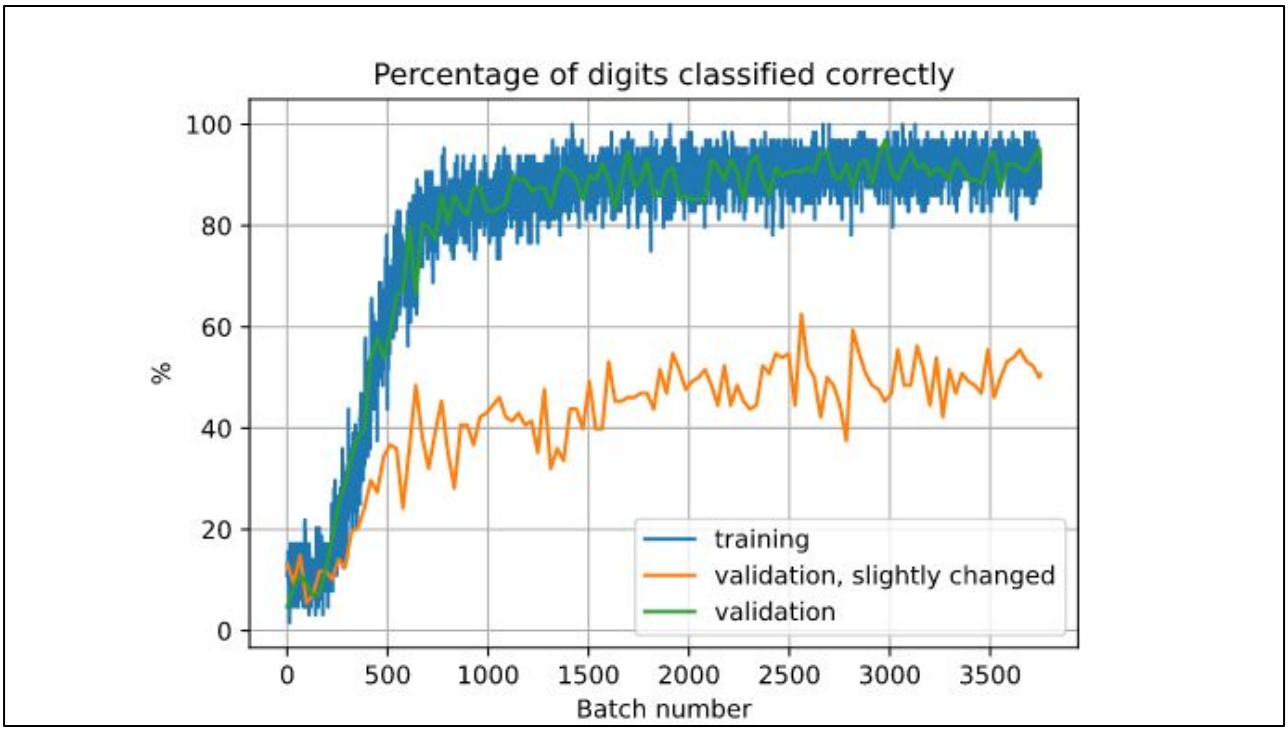


$y = 3x^3 + 4x + 1 + xz$


Gradient descent with momentum
$y = x \cdot cos(x) - 0.2 \cdot sin(6x)$
1
0
−1
−2
−3
−4
y
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
x


Percentage of digits classified correctly
100
80
60
40
20
0
%
0
500
1000
1500
2000
2500
3000
3500
Batch number
training
validation

	0	1	2	3	4	5	6	7	8	9
0	3750	3	27	25	12	42	16	5	10	10
1	4	3793	25	5	13	24	18	16	87	12
2	32	9	3665	74	22	40	24	12	44	3
3	7	31	32	3581	0	110	0	10	53	34
4	70	34	68	3	3766	86	32	17	63	123
5	48	32	22	143	3	3548	23	3	120	15
6	38	18	79	7	52	48	3865	0	8	0
7	3	3	26	40	2	12	0	3716	11	171
8	46	73	55	83	21	75	22	22	3556	37
9	2	4	1	39	109	15	0	199	48	3595


Percentage of digits classified correctly
100
80
60
40
20
0
%
0
500
1000
1500
2000
2500
3000
3500
Batch number
training
validation, slightly changed
validation

The rules

If you take away one thing from this series of posts, please let it be that production use of a neural network tends to go through these four phases (if you are lucky):

1. It works on the training data
2. It also works on the validation data
3. After a lot of disappointment, we get it to work on other people's real life data too
4. Other people can get it to work on their own data as well

Almost all demos declare victory after phase 2.

Parting thoughts

- Deep learning is real
- Deep learning achieves magical things
- Deep learning however is not magic & deceptively simple
- You can still get on board!

- **What I'm hoping for: bring deep learning to our own technology!**. Scripts that help us (say) fight spam and abuse in a way we are used to:
  - Modular ON SITE solutions
  - Reliable, robust
  - With metrics & logging & insight -> TO US
- "SQLite for deep learning"
- Please help!

Useful links

- Whisper.cpp: state of the art voice transcription in dozens of languages, entirely self-contained on your own computer/phone: https://github.com/ggerganov/whisper.cpp
- LlaMA “GPT-like”, self-contained, own computer etc: https://github.com/ggerganov/llama.cpp
- https://berthub.eu/articles/posts/hello-deep-learning/ - the series behind this presentation, https://github.com/berthubert/hello-dl
- https://berthub.eu/articles/posts/ai-is-guaranteed-to-disrupt-us/

Deep learning totally from scratch

There is hope