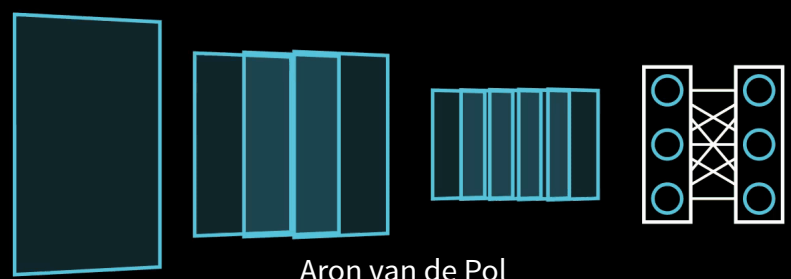
🔥 Deep Learning Workshop



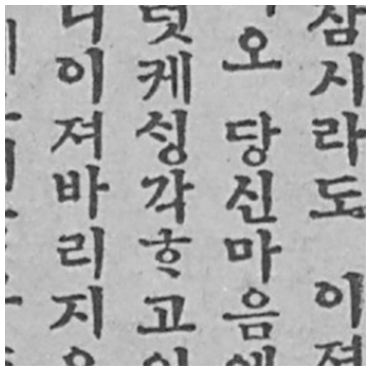
Aron van de Pol

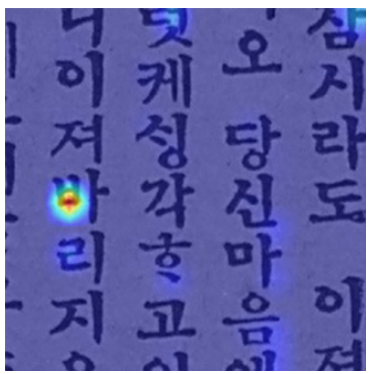
Leiden University Centre For Digital Humanities

2024-02-06

Introduction

- Aron van de Pol
- Korean Studies & Digital Humanities minor
- Focus on computer vision applications in Colonial Korean printing





Workshop Website

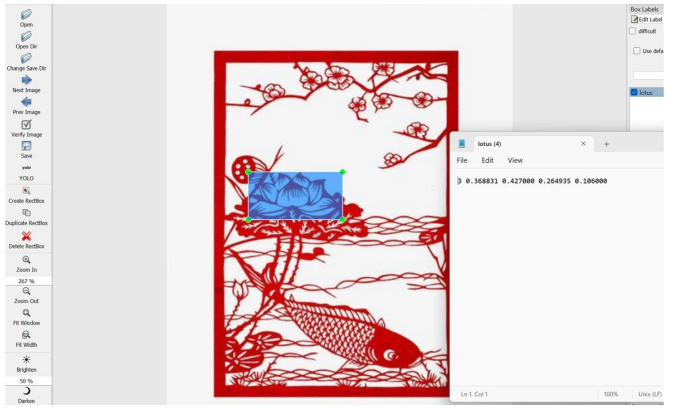
- Find all the materials and other information here: www.dlfh.aronvandepol.com

Planning

- **Session 1.** Tensors (6 February - Today…)
- **Session 2.** Linear regression (Training Pt 1.) (20 February)
- **Session 2.** Non-Linear Data (Training Pt 2. & Evaluation) (5 March)
- **Session 3.** Images as Data (19 March)
- **Session 4.** Image classification (2 April)
- **Session 5.** Transfer Learning (16 April)
- **Session 6.** Spill-over Session (30 April)

 Computer Vision?

Computer Vision?



- OCR/HTR
- Large Scale pattern recognition
- 3D Modelling

Top ten nearest neighbors

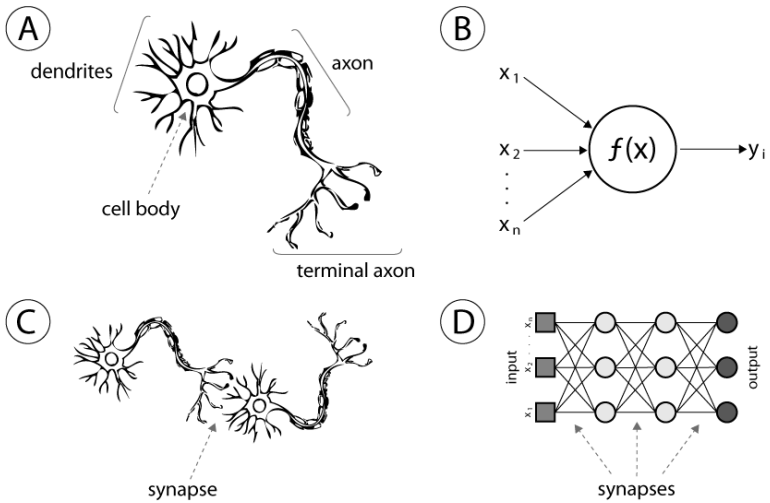
For further readings on computer vision in the Humanities see: Wevers and Smits[1]

🧠 Logic Networks?

- Logic pattern approach: $\mathrm{A} \rightarrow \mathrm{B}$
- *if it rains (A), I am inside (B).*
- Good for simple rules but hard for complex questions.
- How do you know that this is a cat? 🐈
  - Whiskers?
  - Tail?
  - Ears?

Neural Networks?

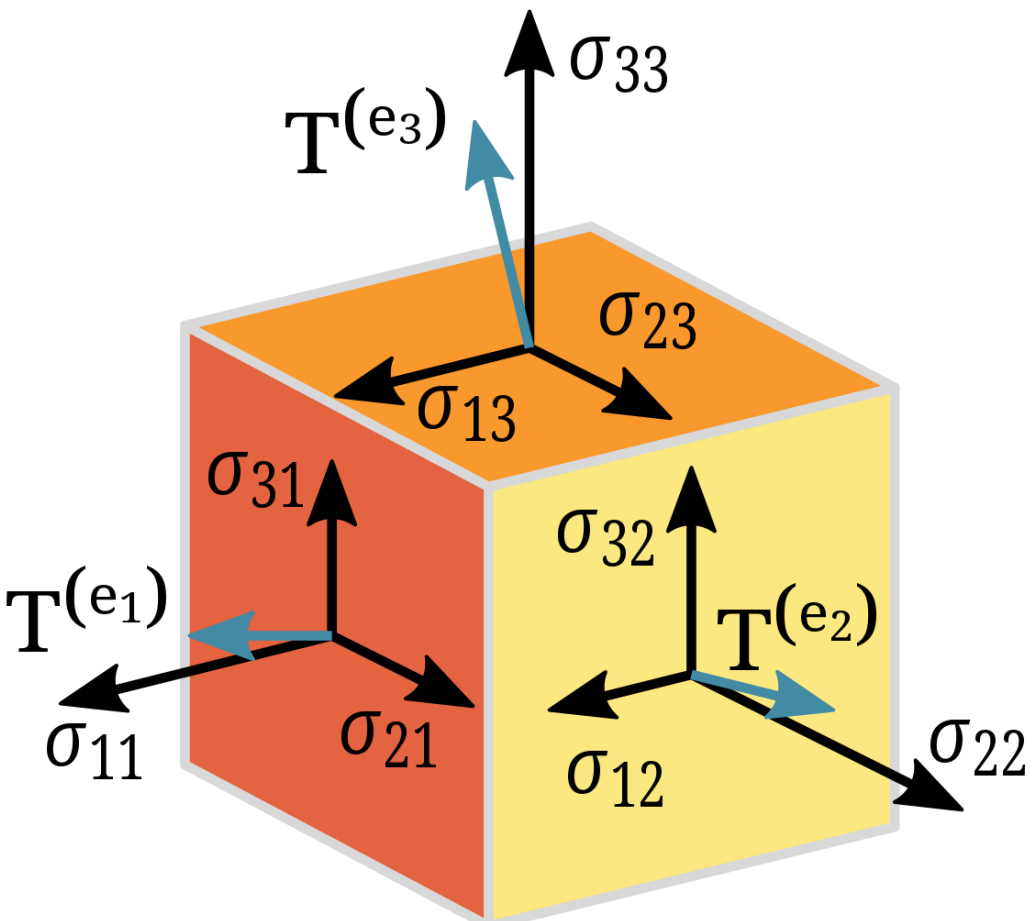
- Sometimes describing the logic is difficult
- Solution in Neural Networks
- Modeled after the human brain
- Figures out the logic by learning from examples



A Neuron & A Neuron Networks[1]

Tensors

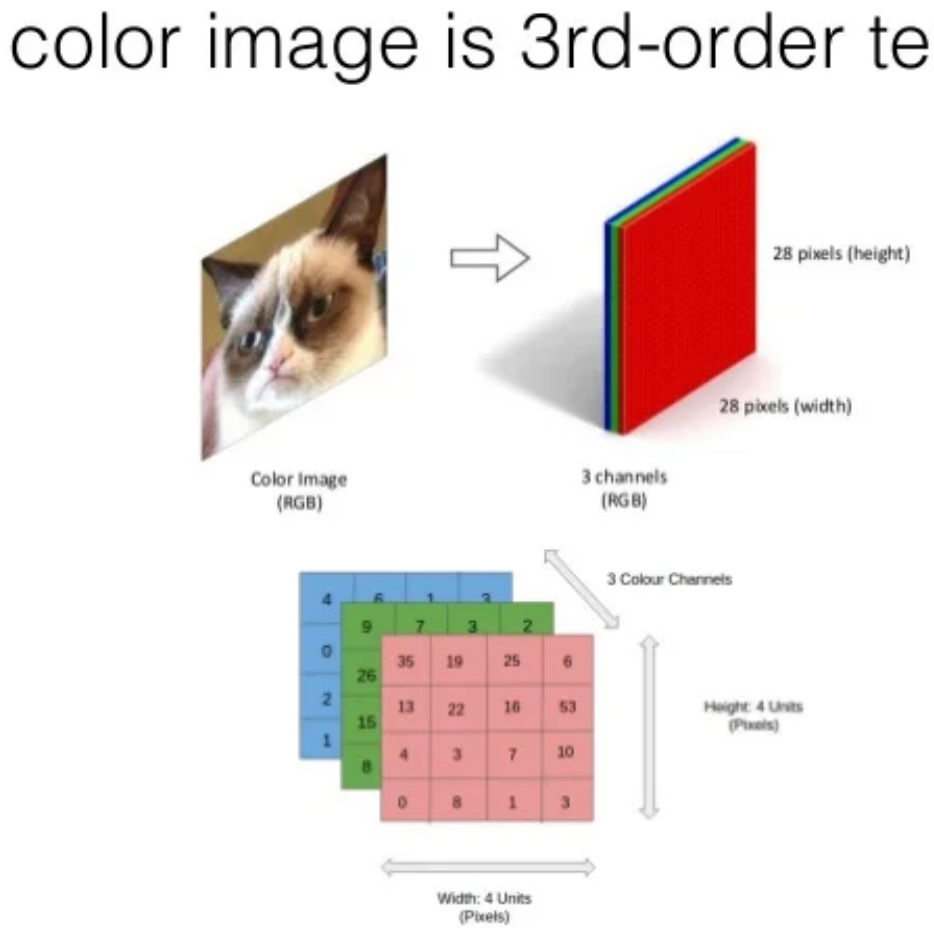
The powerhouse of the Neural Network



Wait… Tensors?

- Tensors are great at storing many types of data *efficiently*.
- Images, text, audio etc.
- Also computionally efficient.

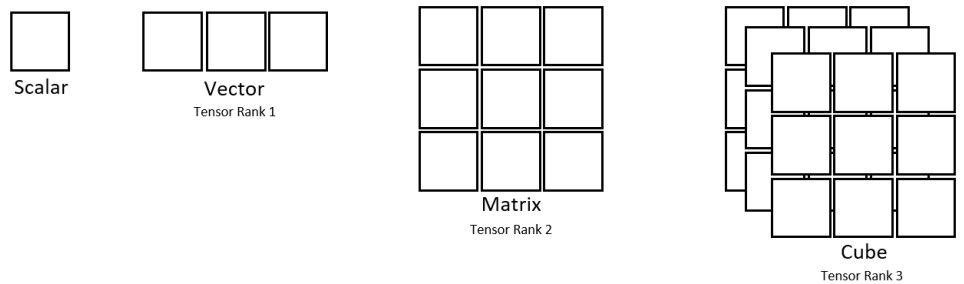
Okay, but what is a Tensor?



Images from: Wikipedia contributors[3], Earnshaw[4], Wevers and Smits[5], Wikimedia Commons contributors[6]

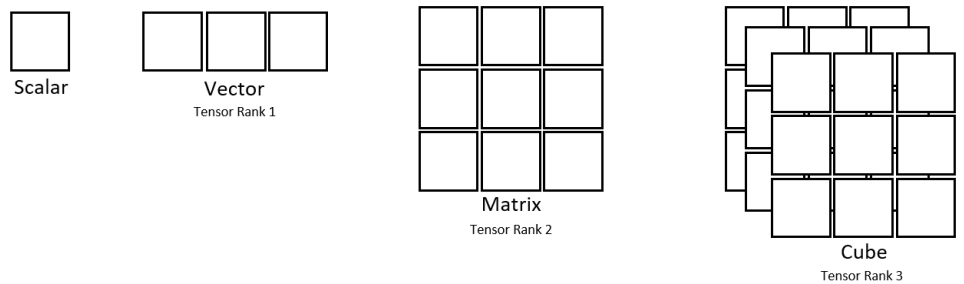
A Scalar...

```
1 # Scalar
2 1
```



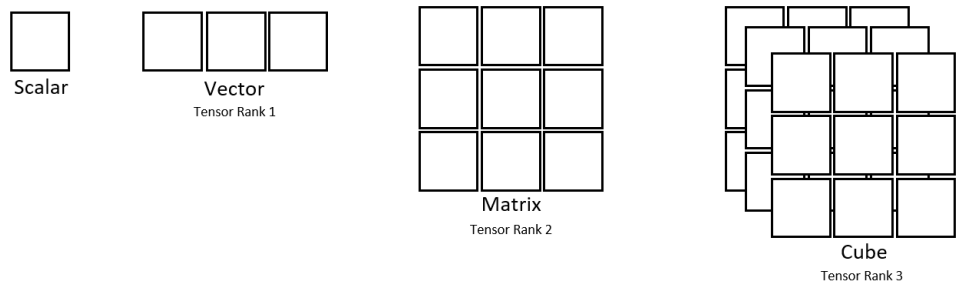
A Vector...

```
1 # Scalar
2 1
3
4 # Vector
5 [1,2,3]
```



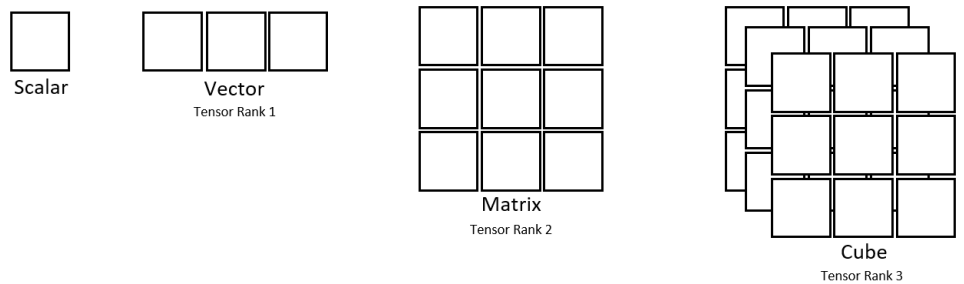
A Matrix...

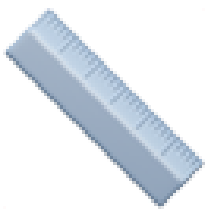
```
1  # Scalar
2  1
3
4  # Vector
5  [1,2,3]
6
7  # Matrix
8  [[1,2,3],
9   [4,5,6],
10  [7,8,9]]
```



A Tensor (Cube)...

```
1  # Scalar
2  1
3
4  # Vector
5  [1,2,3]
6
7  # Matrix
8  [[1,2,3],
9   [4,5,6],
10  [7,8,9]]
11
12  # Cube
13 [[[1,2,3],
14   [4,5,6],
15   [7,8,9]],
16  [[1,2,3],
17   [4,5,6],
18   [7,8,9]]]
```





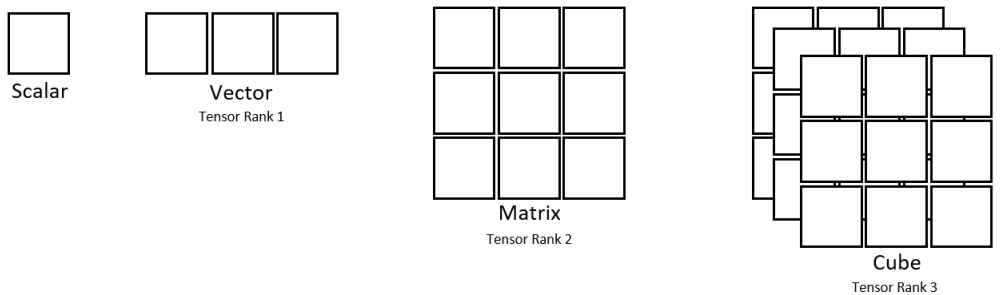
Dimensions

Important attributes of Tensors

📏 Dimensions

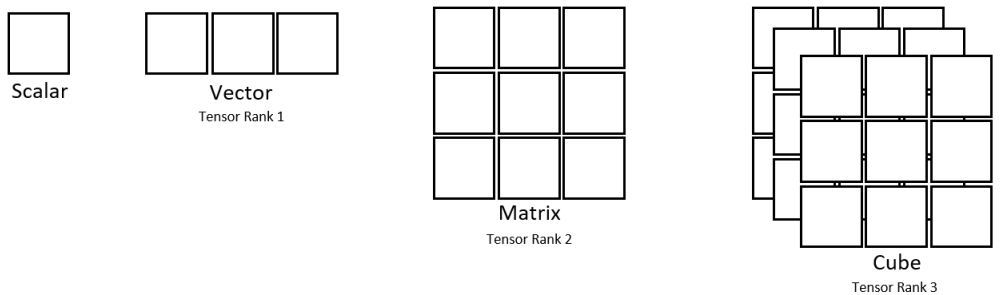
- Tensors have a so-called dimension
- *A dimension can be thought of as how many numbers are required to locate a point in a given tensor.*

Dimensions



- A scalar has 0 dimensions (No coordinates - only one location is possible)
- A vector has 1 dimension (One point needed)
- A matrix has 2 dimensions (Two points needed)
- A Cube-Tensor has 3 dimensions (Three points needed)

📏 Dimensions



Sometimes you'll encounter something like:

N-th Dimensional Tensor

- The N refers to the amount of dimensions of a tensor.
- An example of a useuable 4th dimensional Tensor would be colored video data. (RGB + Frame)

📏 Dimensions

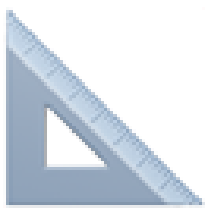
Counting the number of brackets at the start will tell you the number of dimensions.

```
# Scalar - 0
1

# Vector - 1
[1,2,3]

# Matrix - 2
[[1,2,3],
 [4,5,6],
 [7,8,9]]

 # Cube - 3
[[[1,2,3],
  [4,5,6],
  [7,8,9]],
 [[1,2,3],
  [4,5,6],
  [7,8,9]]]
```



Shapes

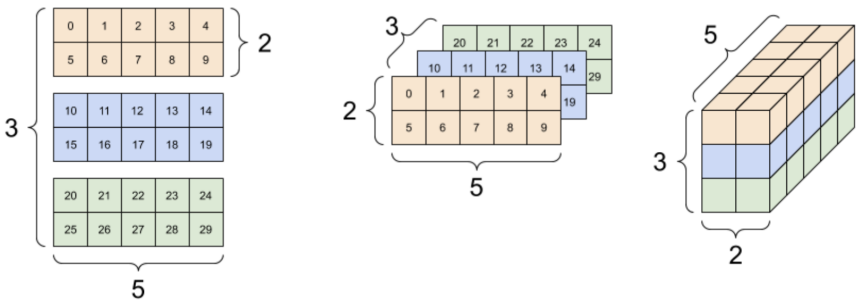
Important attributes of Tensors

Shapes

- Tensors also have shapes
- *The shape is the amount of elements in a tensor per dimension.*
- This means that:
  - A 1-D Tensor has shape (x)
  - A 2-D Tensor has shape (x, y)
  - A 3-D Tensor has shape (x, y, z)
  - etc…

Shapes

Take this representation of a single 3D Tensor




```
[[[ 0,  1,  2,  3,  4],
  [ 5,  6,  7,  8,  9]],

 [[10, 11, 12, 13, 14],
  [15, 16, 17, 18, 19]],

 [[20, 21, 22, 23, 24],
  [25, 26, 27, 28, 29]]]
```

Its shape is (3,2,5).

Working from outer bracket to inner bracket.

Image sourced from TensorFlow Documentation[7]

🤦 Why learn this?

Most issues you will encounter in while working with Neural Networks are `shape errors`.

```
RuntimeError: mat1 and mat2 shapes cannot be multiplied (3x4 and 3x4)
```

- Without knowing dimensions and shapes, you can not solve errors like this.
- Often as researchers we only care about the dimension and shapes of Tensors.

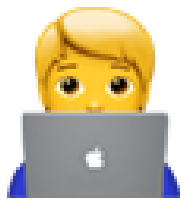
🧪 Brief Test

What are the # of dimensions and the shape of the following tensor?

```
[[[1, 2],
  [3, 4],
  [5, 6]],
 [[7, 8],
  [9, 10],
  [11, 12]]]
```

dimensions: 3

shape: (2,3,2)



Let’s Code

See the website for a link to the Notebooks

Tensor Datatypes

- Tensors have various datatypes like other Python integers, floats or Numpy arrays.
- Some examples of common datatypes are:
  - `torch.float32`
  - `torch.float16` (half precision)
- we can find the datatype of a created `torch.tensor` using `.dtype`

```
1  X = torch.rand(1,2)
2  print(X.dtype)
3  > torch.float32
```

Tensor Devices

- Tensors also have another attribute called `device`.
- GPUs devices are great for running many calculations in parallel.
- The possible devices are:
  - `cpu` (the processor on a computer)
  - `cuda` (The GPU on a computer)
  - `mps` (GPU on Mac computers)
  - `tpu` (Tensor Processing Units - GPUs made for tensors)

Tensor Devices

- we can find the device of a created `torch.tensor` using `.device`

```
1  X = torch.rand(1,2)
2  print(X.device)
3  > device(type='cpu')
```

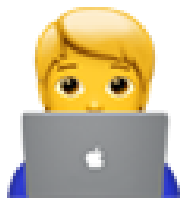
- we send tensors to devices like so:

```
1  X = torch.rand(1,2)
2  X.to('cuda')
3  print(X.device)
4  > device(type='cuda:0')
```

❌ Datatype & Device Errors

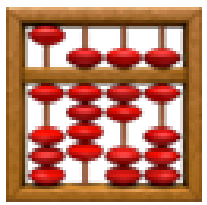
Why learn this?

- Besides the `shape error` the second most common errors are:
  - `dtype errors`
  - `device errors`
- These occur when mixing devices or types.
  - combining `torch.float32` and `torch.float16`.
  - combining `device='cpu'` and `device='gpu'`.



Let’s Code

See the website for a link to the Notebooks



Tensor Operations

Simple Operations

- Addition

```
1 [1,2,3] + 2         = [3,4,5]
2
3 [1,2,3] + [3,2,1] = [4,4,4]
```

- Subtraction

```
1 [1,2,3] - 2         = [-1,0,1]
2
3 [1,2,3] - [3,2,3] = [-2,0,0]
```

Simple Operations

- Multiplication (element-wise)

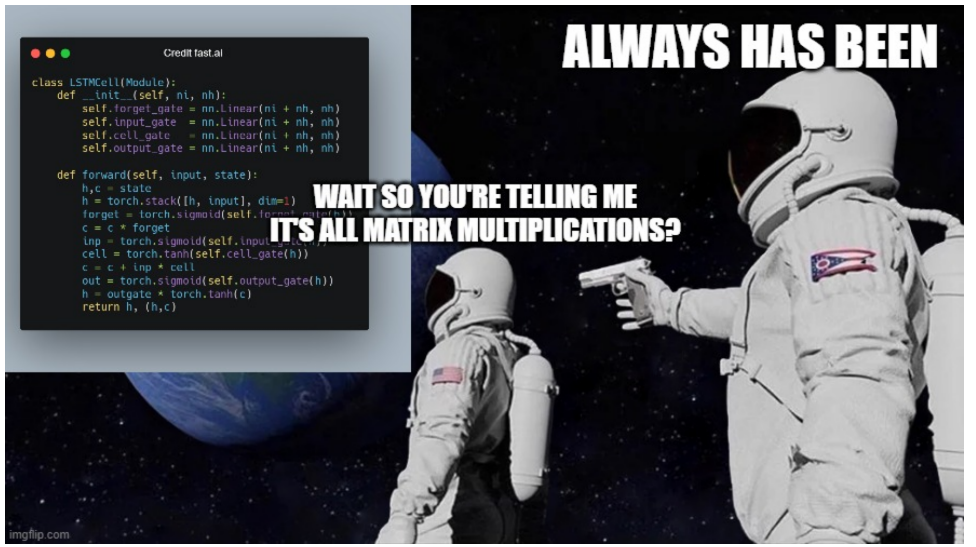
```
1 [1,2,3] * 2         = [2,4,6]
2
3 [1,2,3] * [3,2,3] = [3,4,9]
```

- Division

```
1 [1,2,3] / 2         = [0.5000,1.0000,1.5000]
2
3 [1,2,3] / [3,2,3] = [0.3333,1.0000,1.0000]
```

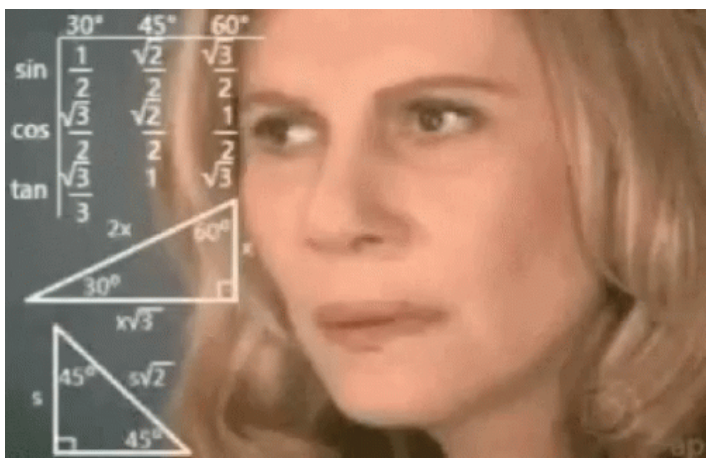
Matrix Multiplication

- The core of what happens inside Neural Networks
- We don’t handle these ourselves and rarely manually do MatMuls
- It is important to understand the principles



Matrix Multiplication

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix} \times \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \\ b_5 & b_6 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix}$$



Matrix Multiplication

Let’s see an animation.

$$\begin{bmatrix} -1 & 5 & 3 \\ 3 & 7 & 11 \end{bmatrix} \cdot \begin{bmatrix} 2 & 3 \\ -8 & 0 \\ 3 & 3 \end{bmatrix} = \begin{bmatrix} & \end{bmatrix}$$

Matrix Multiplication

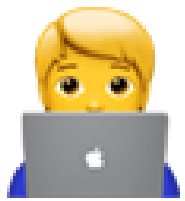
$$\begin{bmatrix} 1 & 3 & 0 \\ 4 & 4 & -1 \end{bmatrix} \cdot \begin{bmatrix} 4 & 4 & 2 & 2 \\ 1 & 2 & 1 & 2 \\ 0 & 4 & 3 & -1 \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \end{bmatrix}$$

Matrix Multiplication

Some rules given a two tensors with shape (2x3) * (3x2) = (2x2)

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \end{bmatrix} \times \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \\ b_5 & b_6 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix}$$

- Inner dimensions mutch match.
  - (2x3) (3x2)
- Resulting tensor has the shape of the outer:
  - (2x3) (3x2)



Let’s Code

See the website for a link to the Notebooks

Reshaping Tensors

Transposing, Squeezing and Unsqueezing

🔧 Transposing

- Transposing a tensor flips the rows and columns.
- A tensor with shape $(4, 5)$ becomes shape $(5, 4)$
- Useful when you want to MatMul two tensors with the same shape.
- In PyTorch you can transpose a tensor with the `.T` method.

```
1 X = torch.rand(4,8)
2 X_transposed = X.T
3 print(X_transposed.shape)
4 > torch.Size([8, 4])
```

Squeezing

- Squeezing is the process of removing dimensions from a tensor.
- Removing any 1's (singleton dimensions) from a tensors shape. `(2,1,2)` becomes `(2,2)`
- In PyTorch this can be seen with the removal of brackets [ ]
- This is done with the `.squeeze()` method.

Squeezing

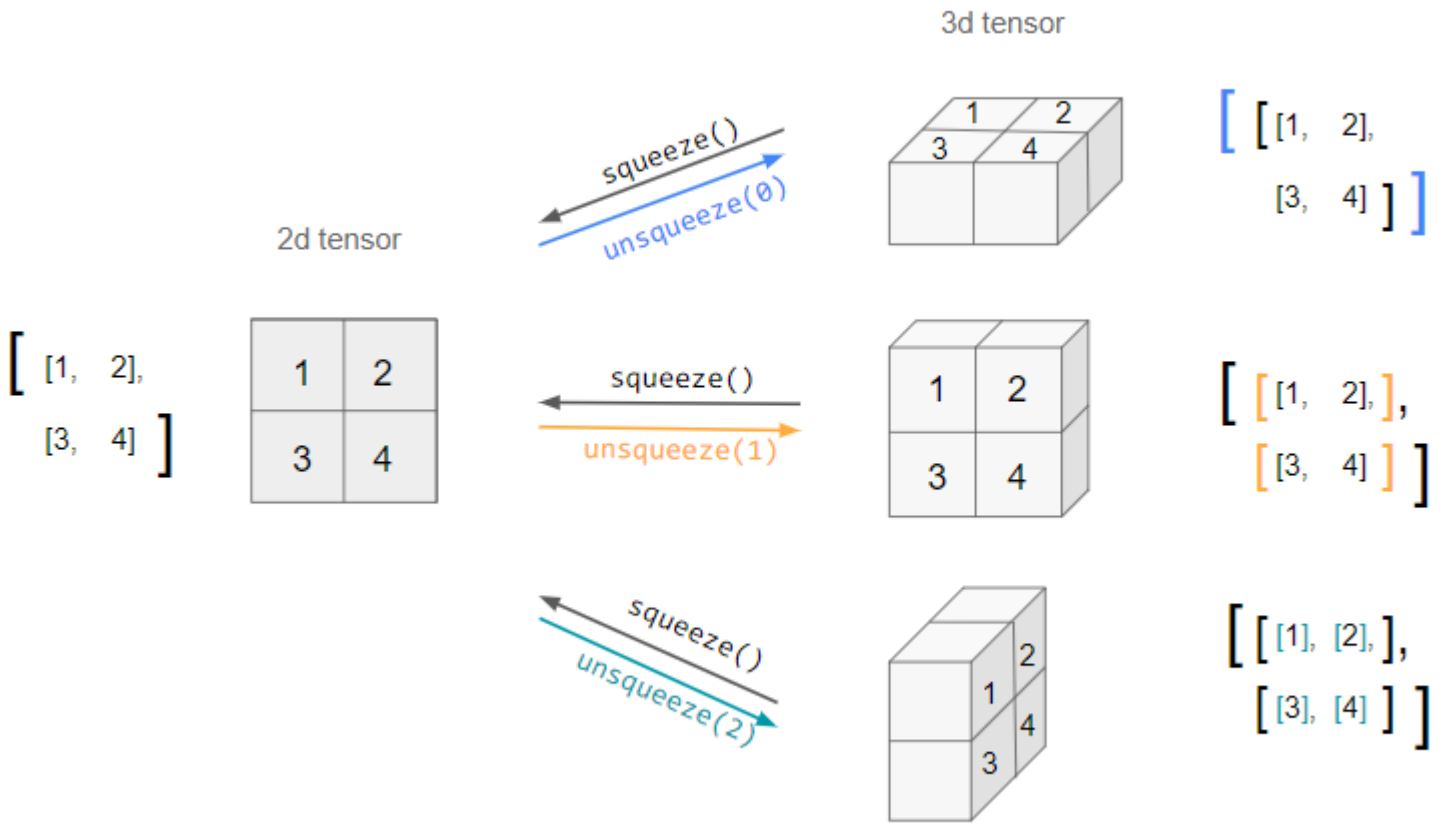
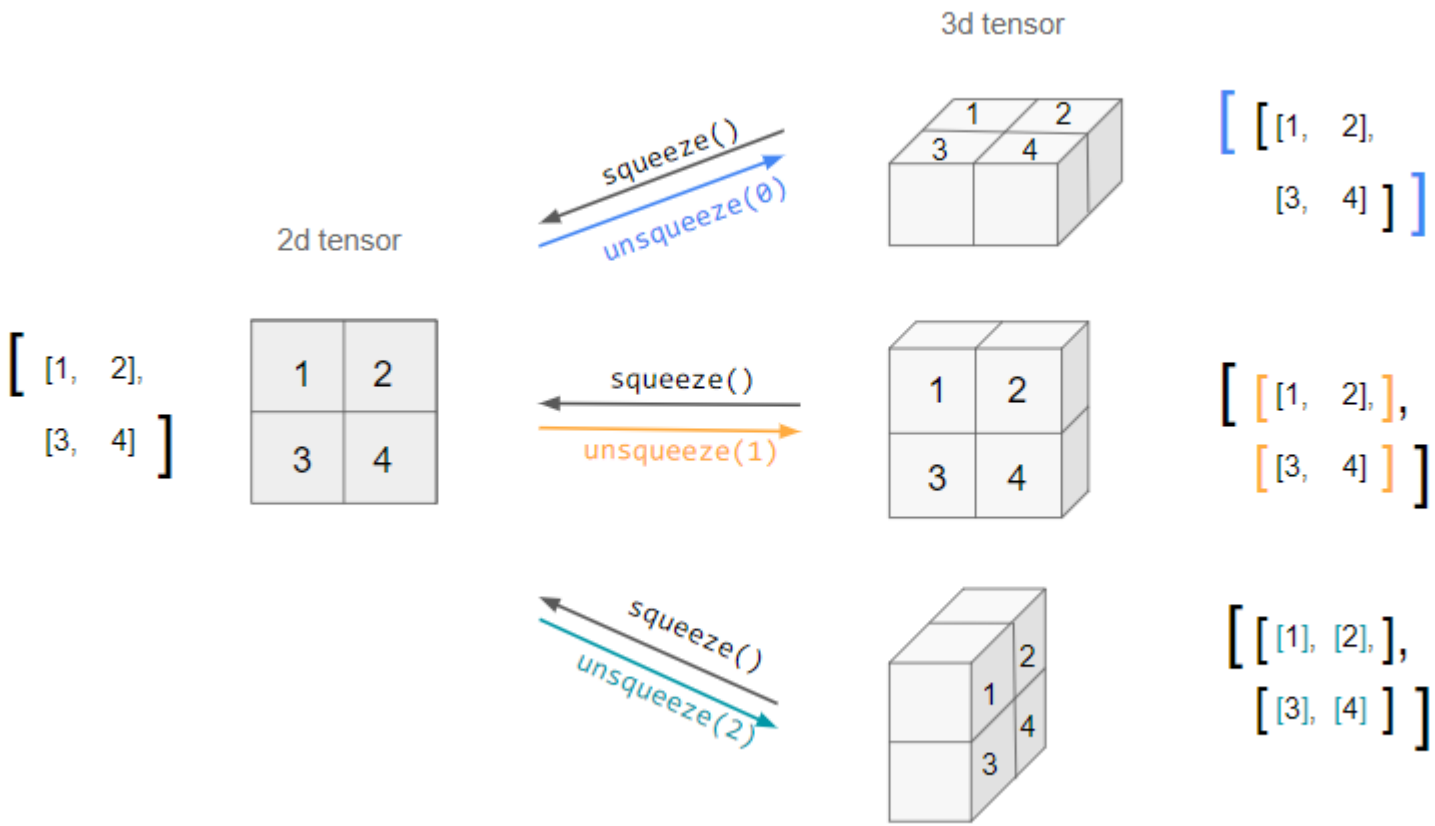


Image source: CopyProgramming[8]

🔨 Unsqueezing

- Unsqueezing is then the opposite of squeezing
- Adding any 1 to a tensors shape, but needs a dimension specified.
- `(2,2)` becomes:
  - `(1,2,2)` when we select `dim=0`
  - `(2,1,2)` when we select `dim=1`
  - `(2,2,1)` when we select `dim=2`

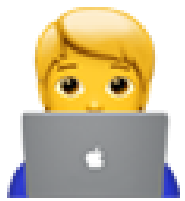
🔨 Unsqueezing



🔨 Unsqueezing

- In PyTorch this can be seen with the addition of brackets [ ]
- This is done with the `.unsqueeze(dim)` method.
  - where `dim` would be the index of where you want to dimension to be added.

```
1  X = torch.rand(2, 2)
2  X_unsqueezed = X.unsqueeze(0)
3  print(X_unsqueezed.shape)
4  > torch.Size([1, 2, 2])
```

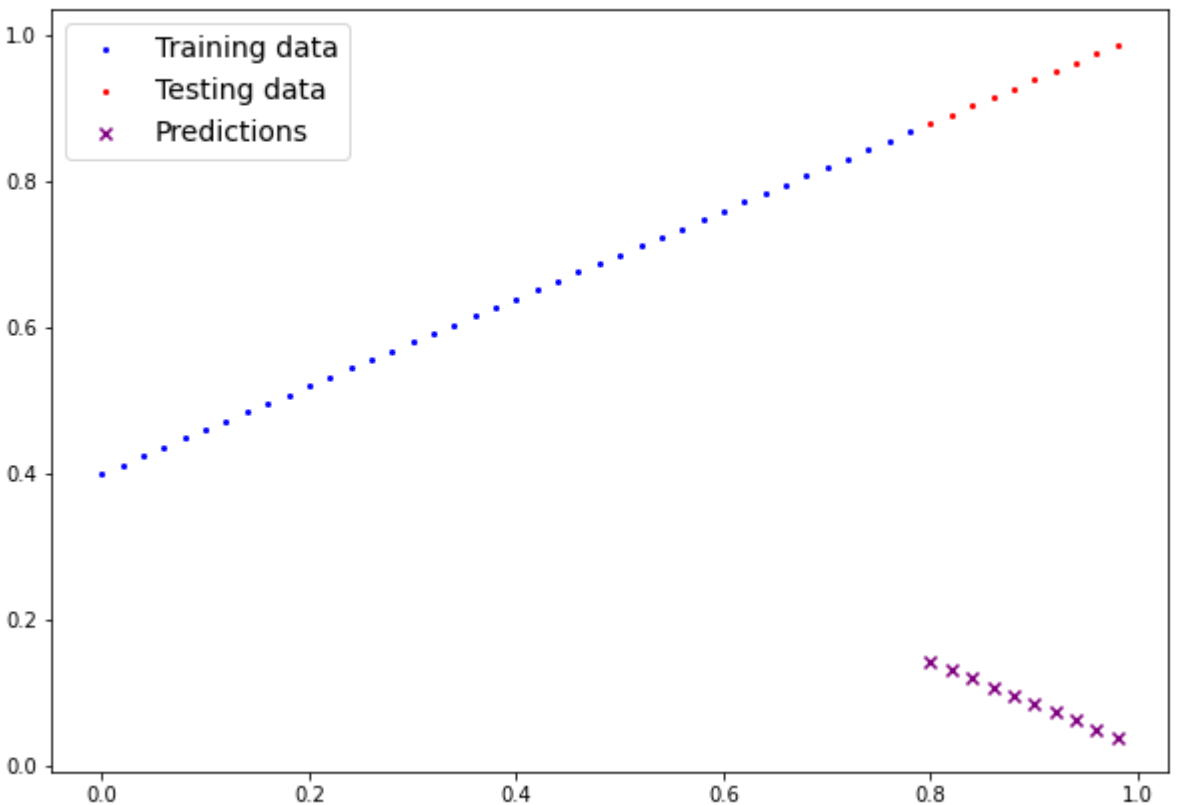


Let’s Code

See the website for a link to the Notebooks

Round-up

In two week: Linear Regression



Works Cited

CopyProgramming. “Torch Squeeze and the Batch Dimension,” Accessed: February 1, 2024. https://copyprogramming.com/howto/torch-squeeze-and-the-batch-dimension#torch-squeeze-and-the-batch-dimension.

Earnshaw, Berton. “A Brief Survey of Tensors.” SlideShare, 2024. https://www.slideshare.net/BertonEarnshaw/a-brief-survey-of-tensors.

Kim, Yungju. “Debate on the Relationship Between Neural Network and the Brain,” March 15, 2020. https://wp.nyu.edu/yungjurick/2020/03/15/debate-on-the-relationship-between-neural-network-and-the-brain/.

TensorFlow Documentation. “TensorFlow - Tensor,” Accessed: February 1, 2024. https://www.tensorflow.org/guide/tensor.

Wevers, Melvin, and Thomas Smits. “The Visual Digital Turn: Using Neural Networks to Study Historical Images.” *Digital Scholarship in the Humanities* 35, no. 1 (April 1, 2020): 194–207. https://doi.org/10.1093/llc/fqy085.

Wikimedia Commons contributors. “Scalar, Vector, Matrix, and Cube Using Squares,” Accessed: February 1, 2024. https://commons.wikimedia.org/wiki/File:Scalar,_Vector_,Matrix_and_Cube_using_squares.png.

Wikipedia contributors. “Tensor,” Accessed: February 1, 2024. https://en.wikipedia.org/wiki/Tensor.