Neural Networks with keras

by: Seth Temple

UNIVERSITY of WASHINGTON

keras: “easy-to-use” NN library for python

UNIVERSITY *of* WASHINGTON

We will use keras over pytorch

- keras
  - built by a Google engineer
  - on top of existing TensorFlow
  - Easy to quickly develop a working model
  - Better commercial support

- PyTorch
  - built by Facebook's AI Research lab
  - Easier to debug
  - Better release management

https://www.reddit.com/r/MachineLearning/comments/6bicfo/d_keras_vs_pytorch/

Sequential()

- For “simple” perceptrons
- Object-oriented
- Define the network topology
  - Use .add() to include layers in model
  - Use .pop() to exclude layers in model
  - Debug using .add() combined with .summary()

https://keras.io/guides/sequential_model/

Layers

- Specify how many nodes are in a layer
- Specify the activation function
  - ReLU
  - Sigmoid
  - Softmax

- Types
  - Dense()
  - dropout layers (can help with overfitting)
  - pooling layers
  - convolutional layers

https://keras.io/api/layers/

Dense()

- This layer is the most basic layer
- Dense(32, activation = ‘relu’)
  - contains 32 nodes
  - uses the rectified linear unit function

.summary()

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Dense_1 (Dense)              (None, 10)                50
_________________________________________________________________
Dense_2 (Dense)              (None, 10)                110
_________________________________________________________________
logits (Dense)               (None, 3)                 33
_________________________________________________________________
softmax (Dense)              (None, 3)                 12
=================================================================
Total params: 205
Trainable params: 205
Non-trainable params: 0
_________________________________________________________________
```

.compile()

- Run this method after setting up the network!
- Parameters chosen here influence results of the backpropagation algorithm
- Specify
  - an optimizer (e.g. Adam)
  - a loss function
  - metrics

.fit()

- Pass X and Y data
- Specify
  - epochs
    - how many times we train our network
    - can be an important parameter to tune
  - batch_size
    - a subset of the data
    - we make gradient updates after each batch
    - can be an important parameter to tune
  - steps_per_epoch
    - related to batch_size
    - # of sample / batch_size

A procedure

1. Instantiate a Sequential() model
2. Define the network by adding layers
   a. Verify with .summary()
3. Train on local CPU
   a. .compile()
   b. .fit()
4. Predict using .predict()
   a. Check on a small sample
   b. Make plots

A better procedure

1. Instantiate a Sequential() model
2. Define the network by adding layers
3. Train on cloud TPUs or GPUs
    a. .compile()
    b. .fit()
4. Save model to a .h5 file using .save_weights()
    a. A neural network is really just a topology with weights
5. Load model to local CPU using .load_weights()
6. Predict on local CPU using .predict()

This is too easy . . .

Really ?!?

- passing inputs of the right dimension can be tricky, particularly for CNNs
- can be difficult to use GPU or TPU accelerators for the first time
- there are many moving pieces to keep track of
- also, we dare you to read a stats paper
  - https://arxiv.org/abs/1412.6980
  - https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf
- and other open questions . . .

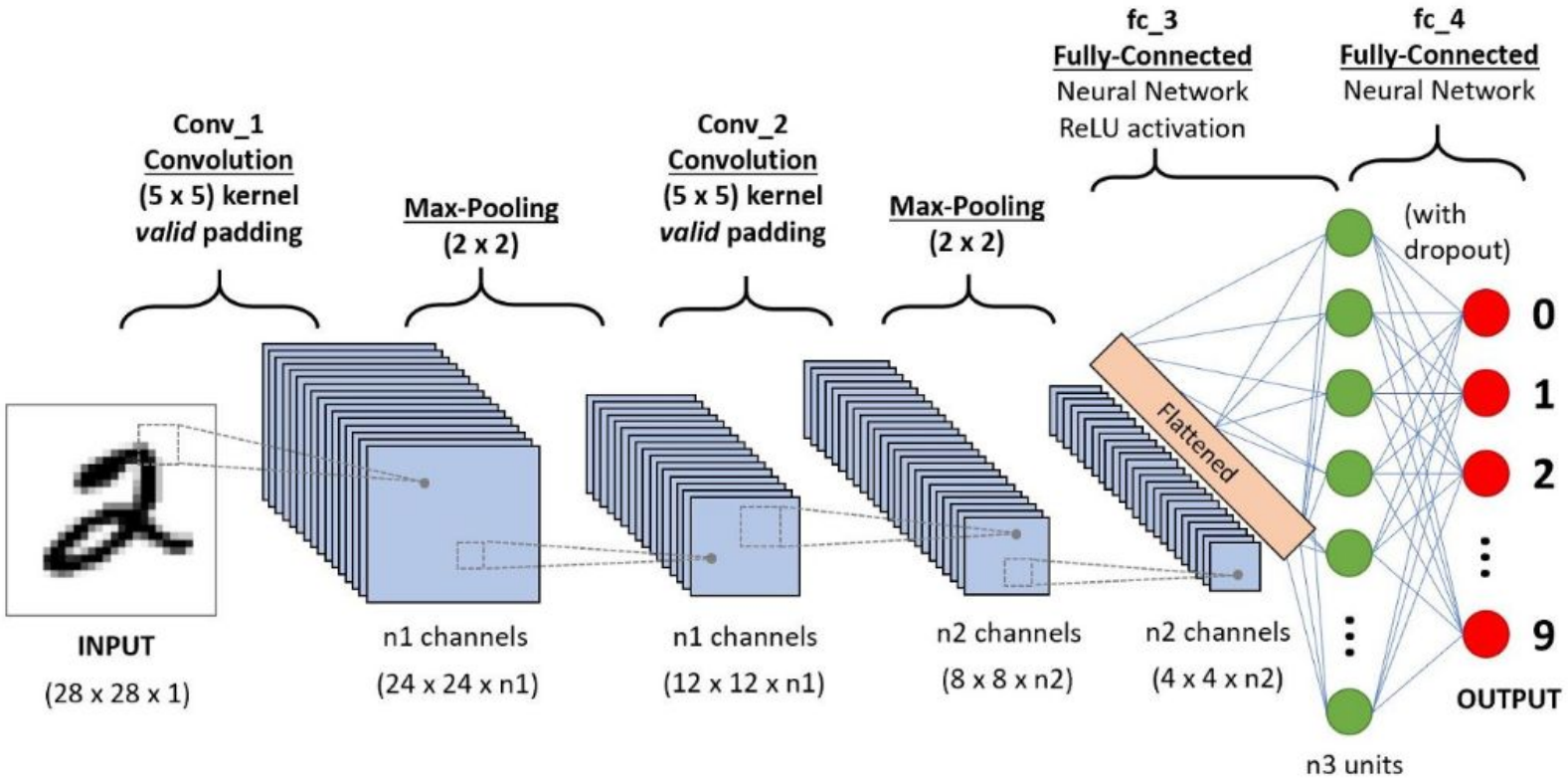
Open questions

- How do we select a network topology?
  - Includes layer types, # of nodes, activations
- Which optimizer should we use?
  - Also, optimizers have other important parameters to select like the learning rate
- How long should we train for?

Grid search may be computationally infeasible

Convolutional Neural Networks (CNNs)

A high-level description



Convolutions

Yuck! We have to do math . . .

Given two functions *f* and *g*, we define the convolution to be as the following integral transform

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau.$$

Refer to the following blog post for visualizations

https://colah.github.io/posts/2014-07-Understanding-Convolutions/

A convolutional layer

- **Filters**
  - a matrix transformation
  - define things like edges or colors in an image
- **Stride**
  - how we skip across the image applying filters
- **Pooling**
  - Dimensionality reduction
  - Combine some cells in matrix together by averaging or taking the max

Refer to these amazing animations !!!

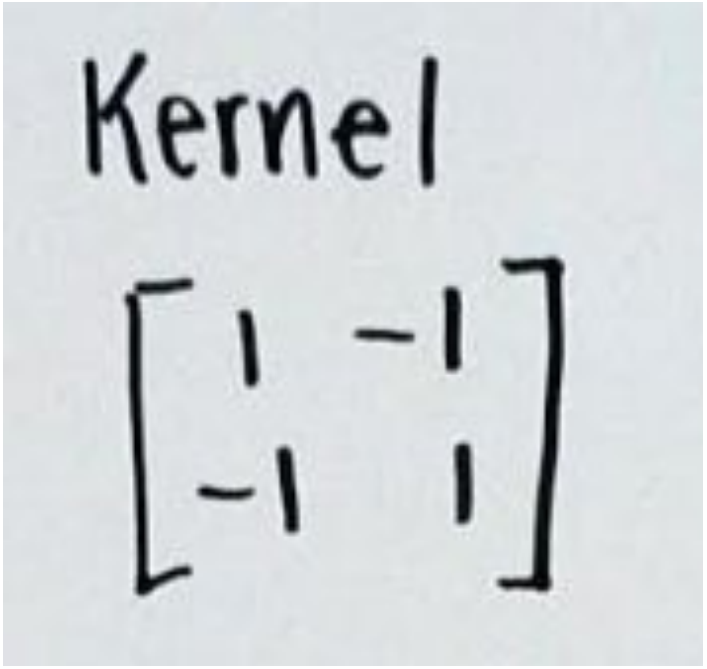
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Conv2D()

- There are 1D, 2D, and 3D convolutional layers
- Specify the following
  - filters
    - the number of filters
  - kernel_size
    - how big the filter is
  - strides
    - how we move the filter around
  - padding
    - handles output dimensionality

Convolution Exercise

We will use this kernel and stride 1



(1) Convolve this matrix

Kernel

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

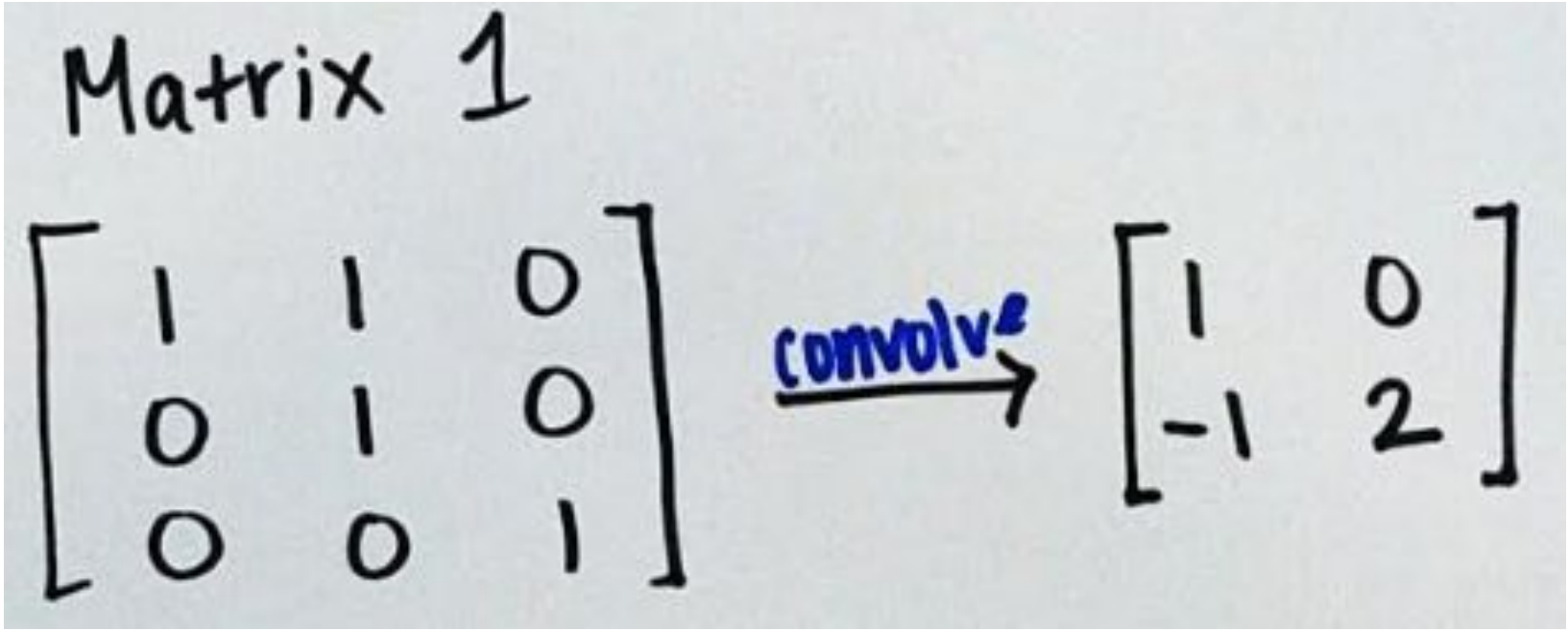
Matrix 1

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(1) Convolve this matrix

Kernel

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$



(2) Convolve this matrix

Kernel

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

- Tell us in chat
  - dimension of the matrix after *
  - value in cell (2, 2) after *

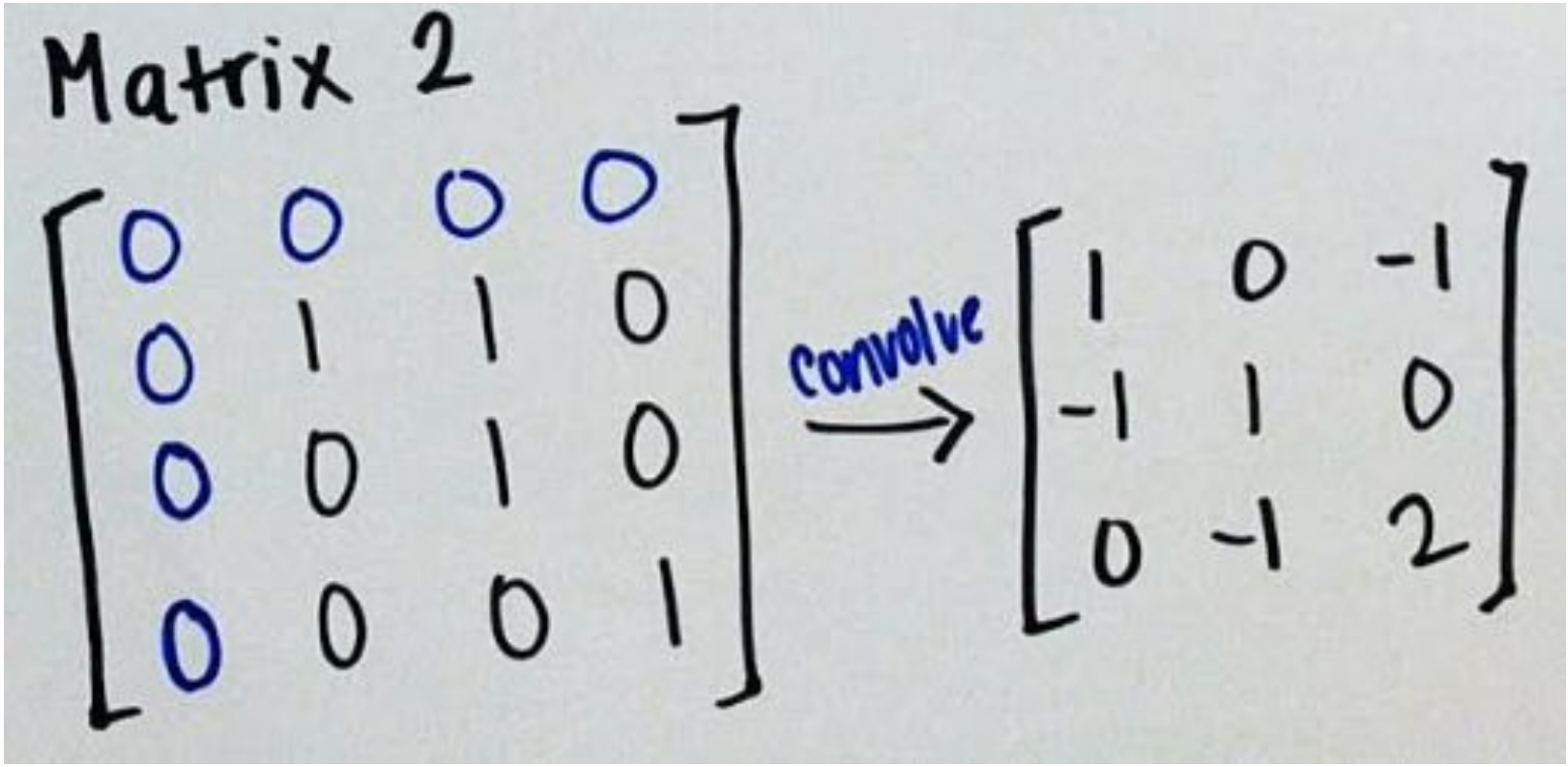
Matrix 2

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(2) Convolve this matrix

Kernel

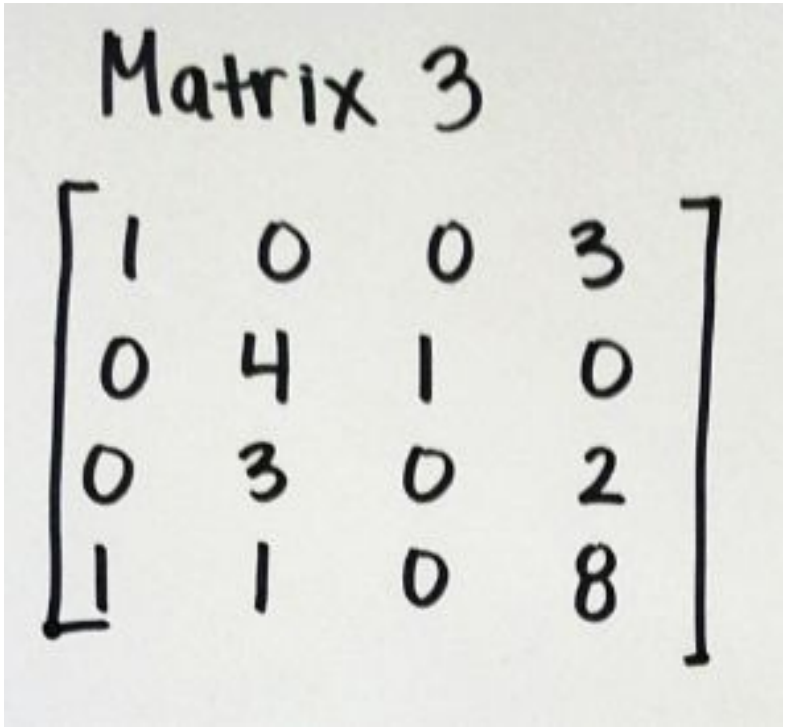
$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$



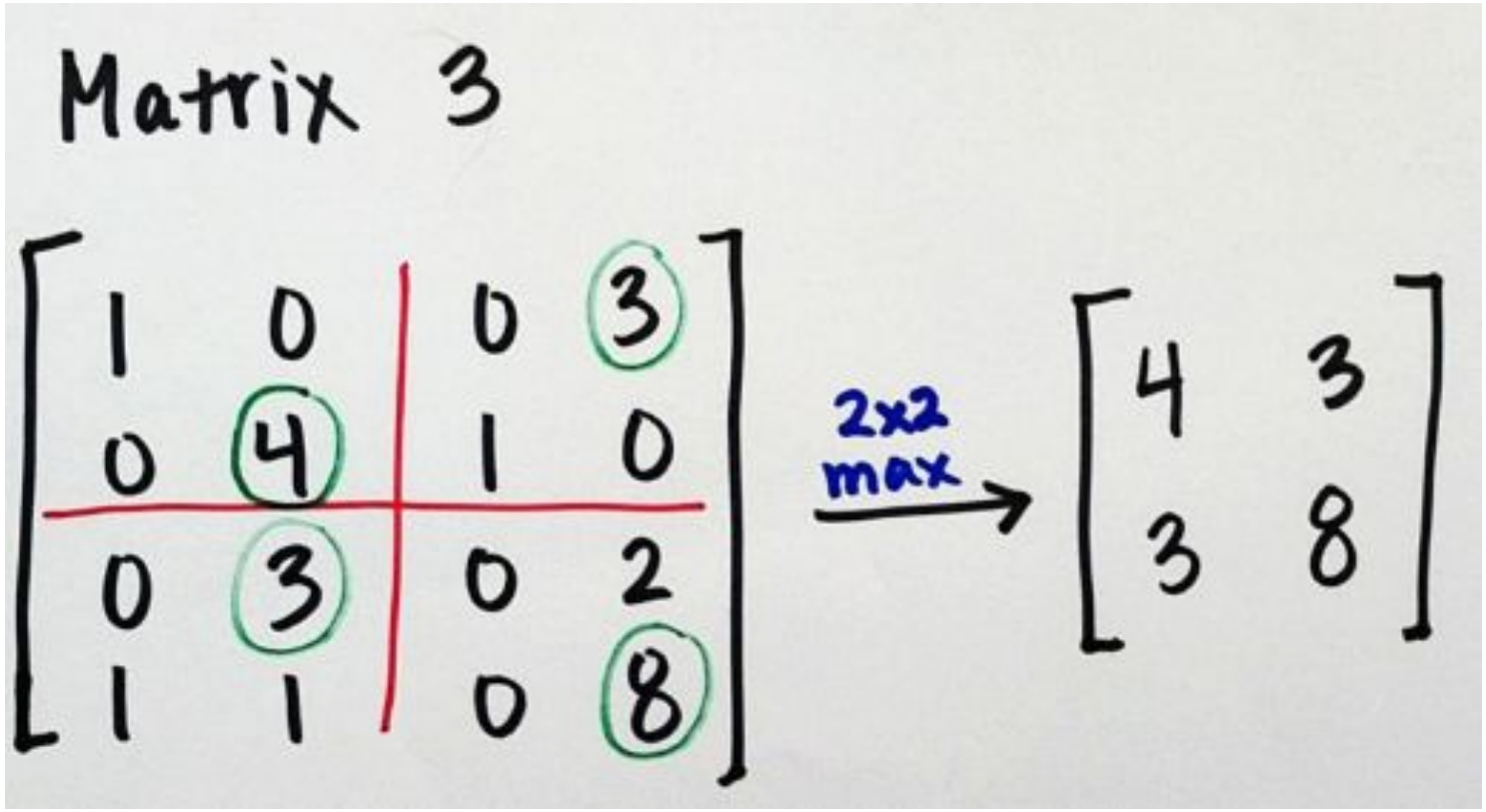
$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{convolve}} \begin{bmatrix} 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 2 \end{bmatrix}$$

(3) Perform max pooling on this matrix

- Use 2 x 2 max pooling
- Tell us in the chat
  - dimensions of the matrix after max pooling
  - value in cell (2, 2) after max pooling



(3) Perform max pooling on this matrix



Some other fun references

> Data augmentation

- https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/
- https://www.tensorflow.org/tutorials/images/data_augmentation

> Valid vs same padding

- https://stackoverflow.com/questions/37674306/what-is-the-difference-between-same-and-valid-padding-in-tf-nn-max-pool-of-t

> Backpropagation

- **See deeplizard YouTube video series**
- https://www.youtube.com/watch?v=XE3krf3CQls&list=PLZbbT5o_s2xq7LwI2y8_QtvuXZedL6tQU&index=23

> Convolution vs cross-correlation

- https://en.wikipedia.org/wiki/Convolution
- https://en.wikipedia.org/wiki/Cross-correlation
- https://www.quora.com/Why-do-CNNs-use-convolution-instead-of-cross-correlation
- https://glassboxmedicine.com/2019/07/26/convolution-vs-cross-correlation/

Copyright