How to Simulate a Self-Driving Car

Inspired by YouTuber Siraj Raval.

Initialize

- This workshop is for aspiring and active developers
- Please let’s help each other out

- Ask who is a Python beginner/comfortable/pro? (show hands)
- Ask minority to disperse

- Feel free to ask questions at any time.
- Including, perhaps even especially, among yourselves.

Start with a demo

Three step process

1) Generate Data
2) Train model
3) Test

1) Data Generation

NVIDIA trained an autonomous car in real life.

Udacity built a self-driving car simulator to perform the same task.

1) Data Generation

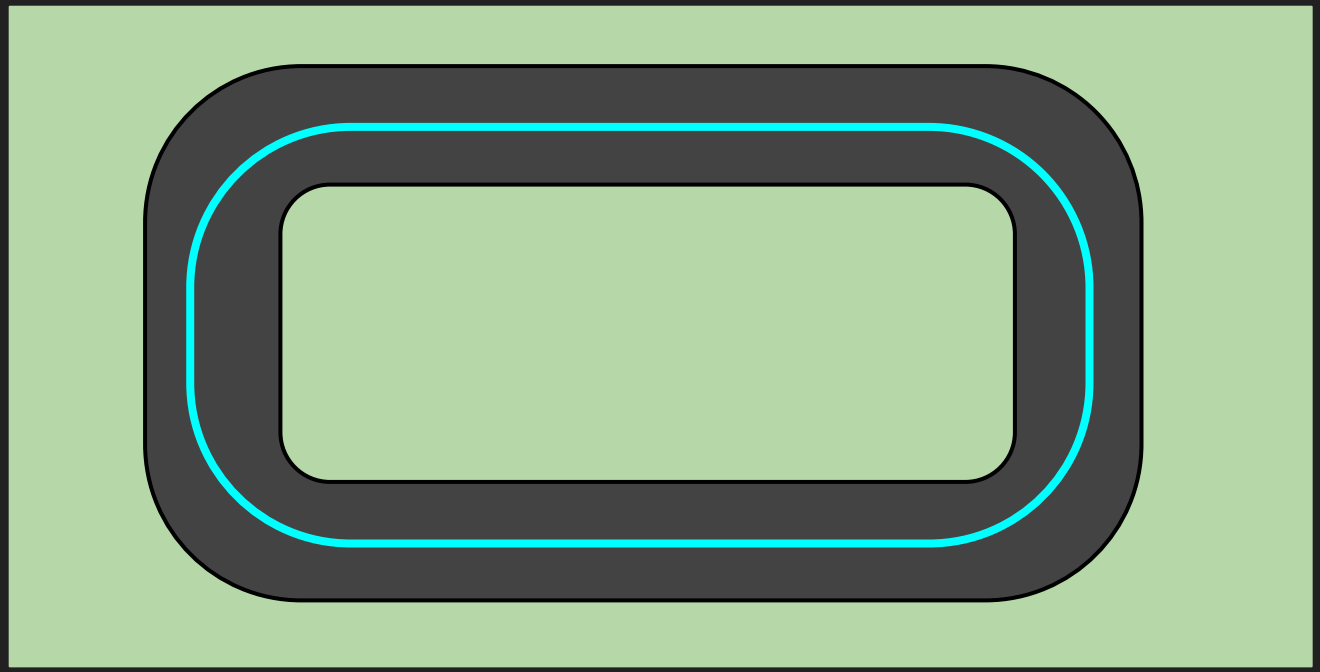
NVIDIA trained an autonomous car in real life.

Udacity built a self-driving car simulator to perform the same task.

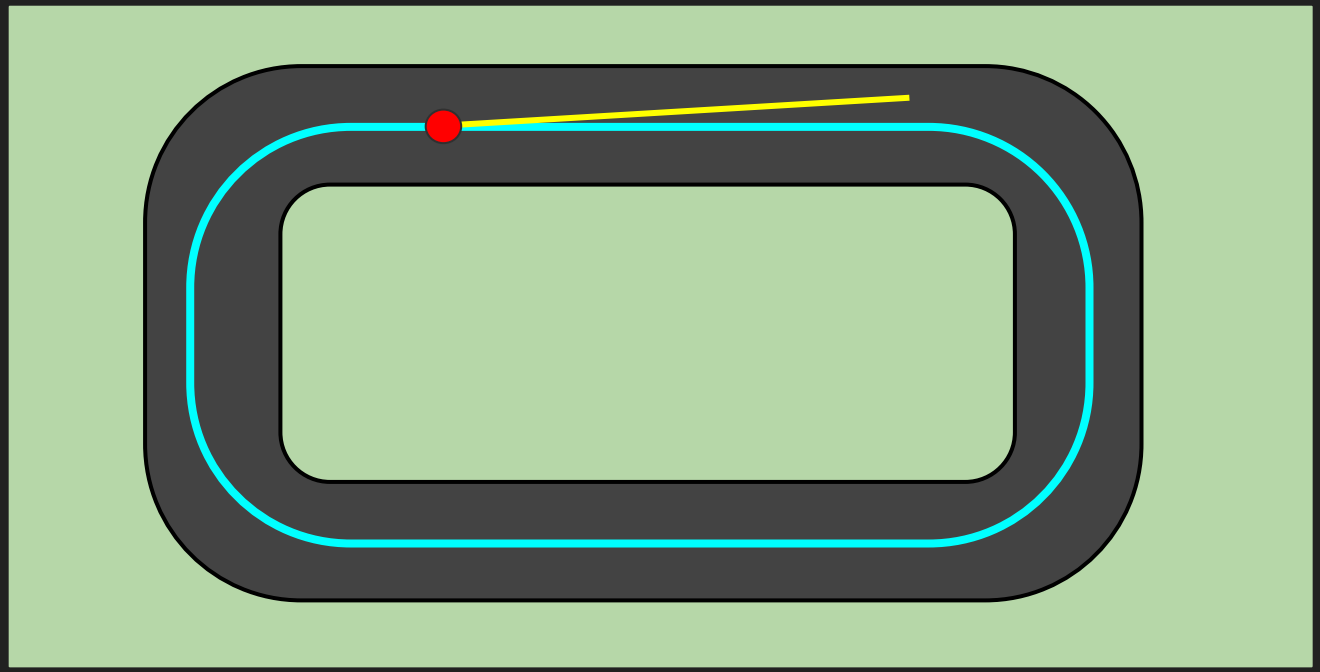
They did the exact same thing we are going to do.

- Three cameras to the head of the car
- Human driver
- Collect data (i.e. steering angle, speed, throttle, brake and images)
- Train a neural net
- Drive car using a single camera

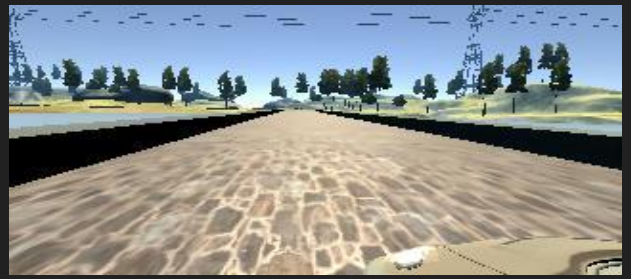
The three cameras



The three cameras



The three cameras



left

center

right

...steers you back on course after minor model errors

2) Training

machine learning approach = behavioural cloning.

- NVIDIA = 72 hours all kinds of conditions
- We will do a few laps on a single circuit.

2) Training

- steering angle
- throttle
- brake
- speed

- ‘how to drive the car’

2) Training

- steering angle
- throttle
- brake
- speed

- ‘how to drive the car’

This requires a lot of training data so we are going to simplify..

.. and just train the steering angle.

2) Training

label images kinda like this..

Left a bit

Straight on

Right

2) Training

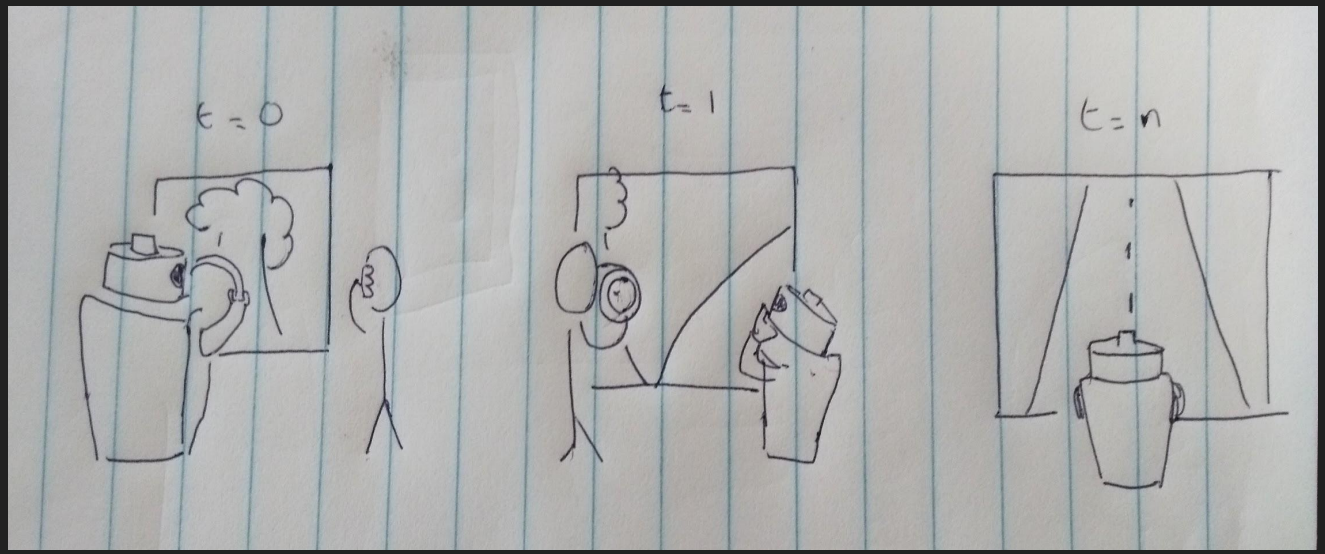
This is an example of a supervised learning algorithm

Reduce the difference between:

1) what human did
2) what the computer thinks it should do

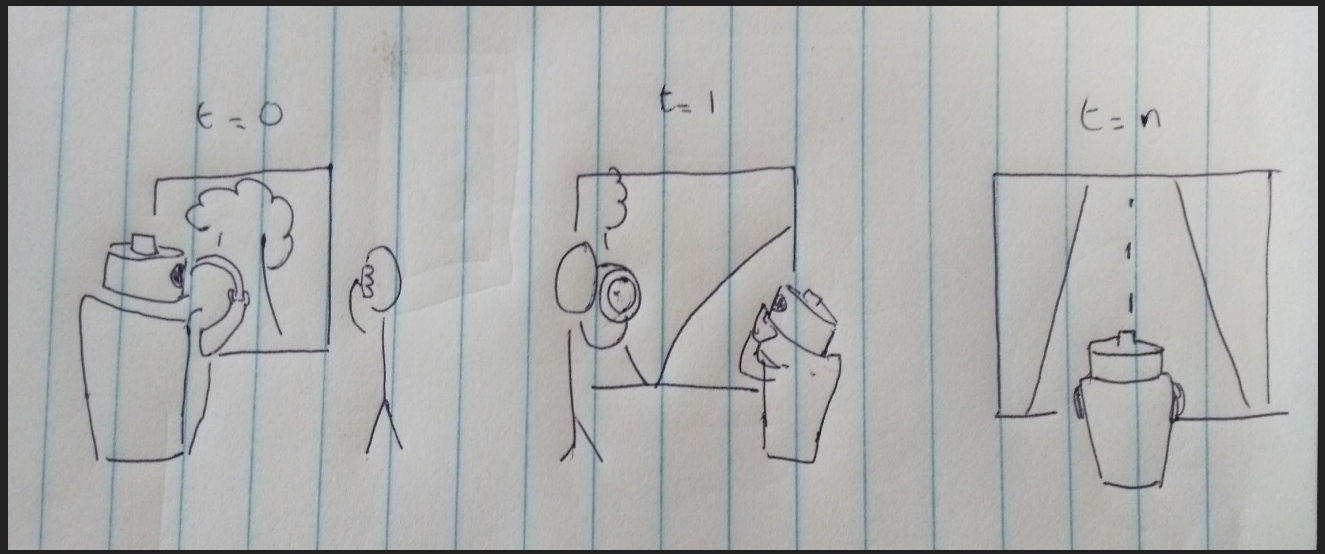
2) Training

This is an example of a supervised learning algorithm



2) Training

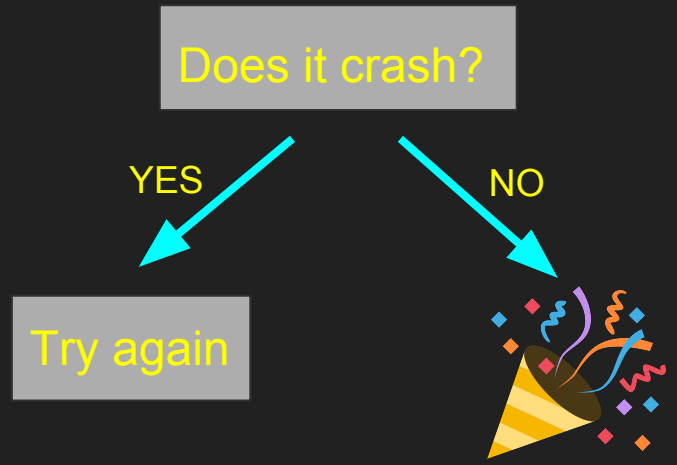
This is an example of a supervised learning algorithm



GIVEN A NEW IMAGE I HAVEN'T SEEN YET, HOW SHOULD I DRIVE?

3) Testing

Drive the car using the neural net.



3) Testing

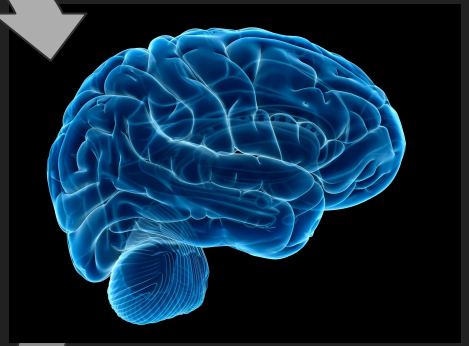
We will communicate using a server-client architecture

3) Testing

server-client architecture

Server
(the simulator)

Client
(car brains)



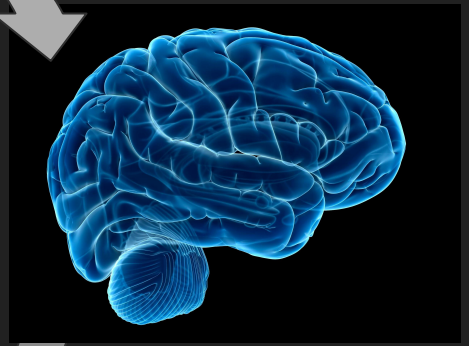
Turn left a bit

3) Testing

server-client architecture

Server
(the simulator)

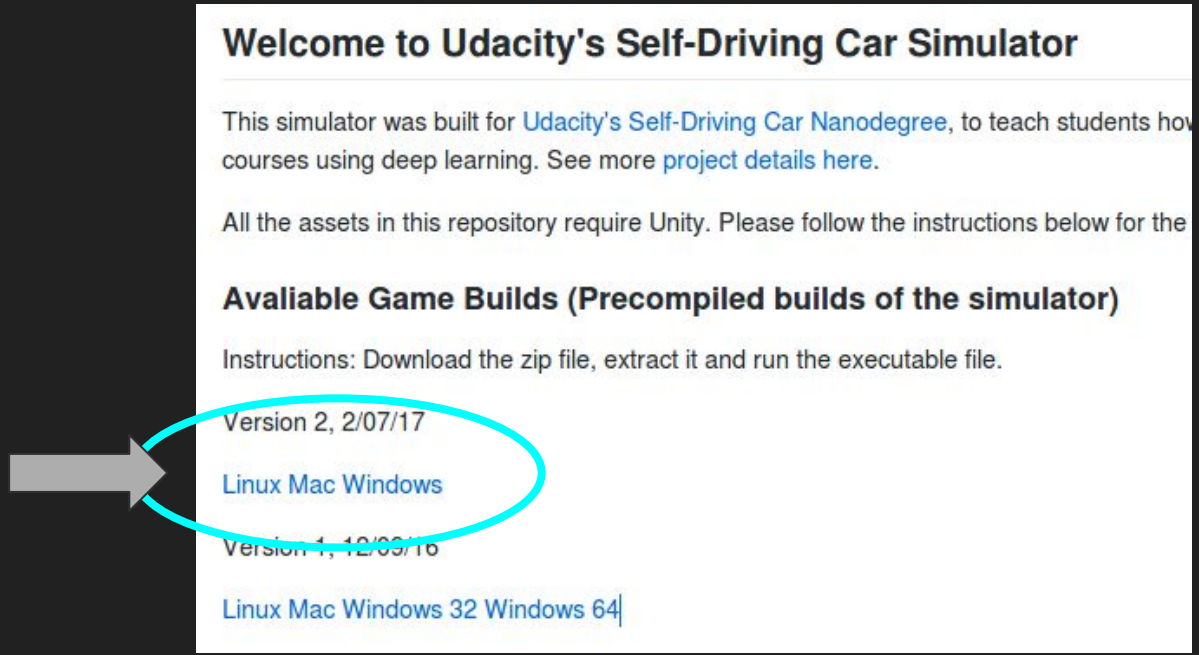
Client
(car brains)



Go forwards

Fork, clone and cd into source code repo

Download Self-Driving Car Simulator



Install miniconda

```
# Install dependencies

conda env create -f environments.yml
source activate car-behavioral-cloning

# Install TensorFlow

conda install tensorflow
```

ALVIN (1992)

NVIDIA (2016)

Generate Data

- Run car simulator in TRAINING MODE
- Press ‘r’ key to select fixed output directory
- Press ‘r’ again to record driving skillz
- Complete between 3 to 5 laps.

Generate Data

Inspect

- Images (IMG/)
- driving_log.csv

Train the model

```
python model.py --help
```

The Model

y = brains(X)

The Model

y = brains(X)

LABEL

INPUT

(ALL MACHINE LEARNING MODELS EVER)

The Model

y = brains(X)

(ALL MACHINE LEARNING MODELS EVER)

LABEL — HOW TO DRIVE

INPUT — IMAGES

(OUR MODEL)

The Model

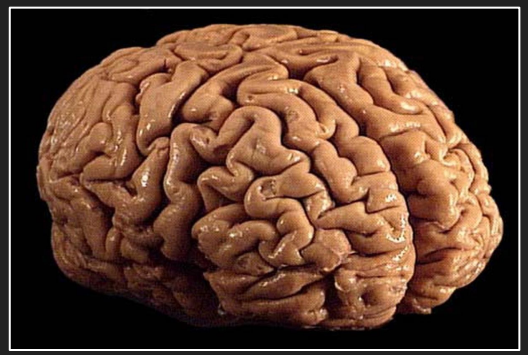
y = brains(X)

(ALL MACHINE LEARNING MODELS EVER)

LABEL — HOW TO DRIVE

INPUT — IMAGES

(OUR MODEL)



BUT WHAT ARE BRAINS?

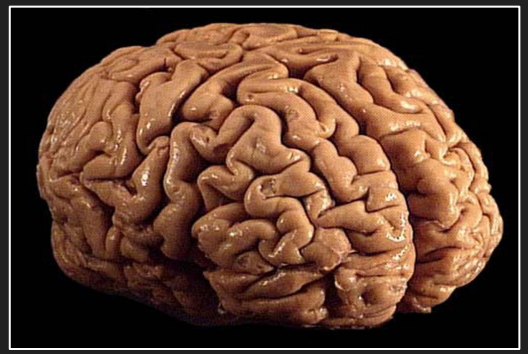
The Model

y = brains(X)

LABEL — INPUT

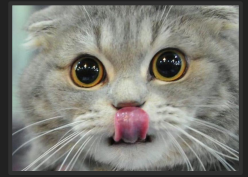
(ALL MACHINE LEARNING MODELS EVER)

HOW TO DRIVE — IMAGES — (OUR MODEL)



BUT WHAT ARE BRAINS?

a way of converting some input into something else that has meaning for humans



= cat

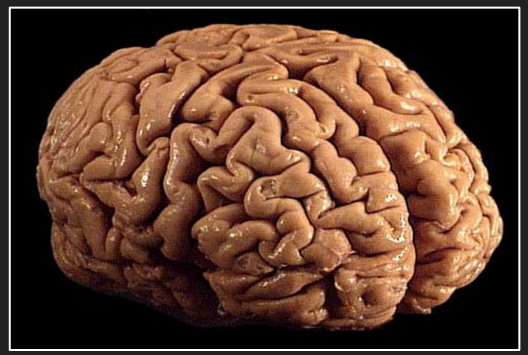
The Model

y = brains(X)

LABEL — INPUT

HOW TO DRIVE — IMAGES — (OUR MODEL)

(ALL MACHINE LEARNING MODELS EVER)



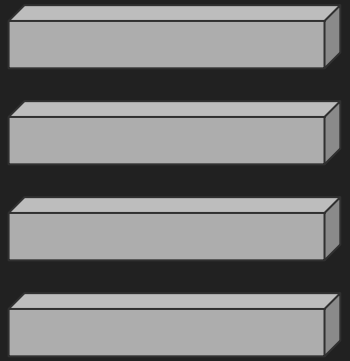
BUT WHAT ARE BRAINS?

a way of converting some input into something else that has meaning for humans

also = arrangement of mathematical functions

Convolutional Neural Network (CNN)

- Keras
  - python machine learning library
- everything arranged in layers
  - input is first layer
  - output is last layer
- Model specifics
  - 2D Convolutional layers
  - Dropout layer
  - Fully connected layers



What is a 2D convolution layer?

- image input
- filter (a.k.a. kernel)
- dot product
- move along
- repeat
- outputs a new layer

1	0	2	1	2	0
0	2	1	2	2	1
1	1	0	2	3	4
4	5	2	1	2	2
1	0	0	0	1	2
3	2	1	0	4	2

What is a 2D convolution layer?

- image input
- filter (a.k.a. kernel)
- dot product
- move along
- repeat
- outputs a new layer

1	0	2	1	2	0
0	2	1	2	2	1
1	1	0	2	3	4
4	5	2	1	2	2
1	0	0	0	1	2
3	2	1	0	4	2

0	1	0
1	1	1
0	1	0

What is a 2D convolution layer?

- image input
- filter (a.k.a. kernel)
- dot product
- move along
- repeat
- outputs a new layer

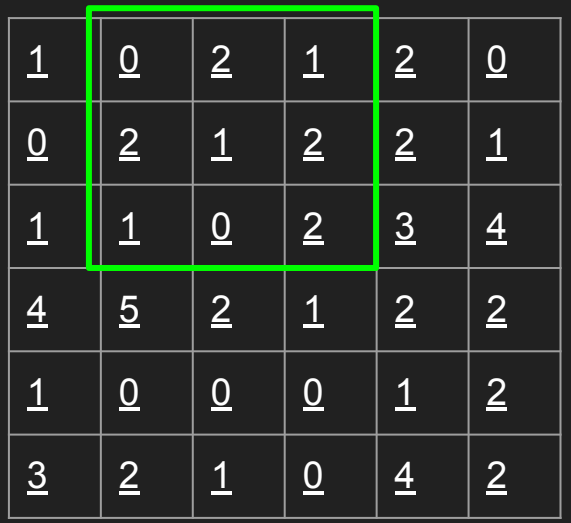
1	0	2	1	2	0
0	2	1	2	2	1
1	1	0	2	3	4
4	5	2	1	2	2
1	0	0	0	1	2
3	2	1	0	4	2

0	1	0
1	1	1
0	1	0

= 4

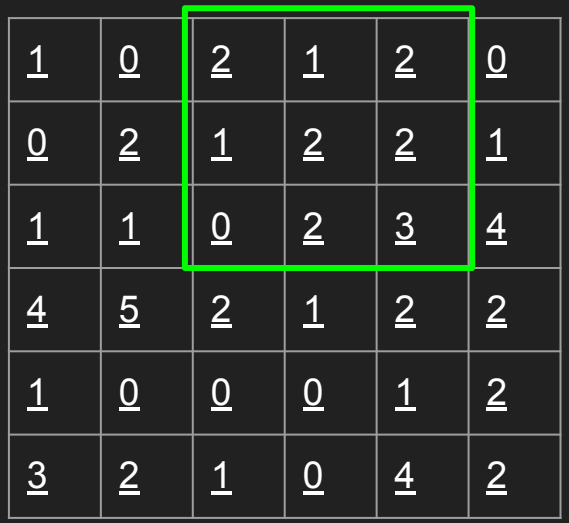
What is a 2D convolution layer?

- image input
- filter (a.k.a. kernel)
- dot product
- move along
- repeat
- outputs a new layer



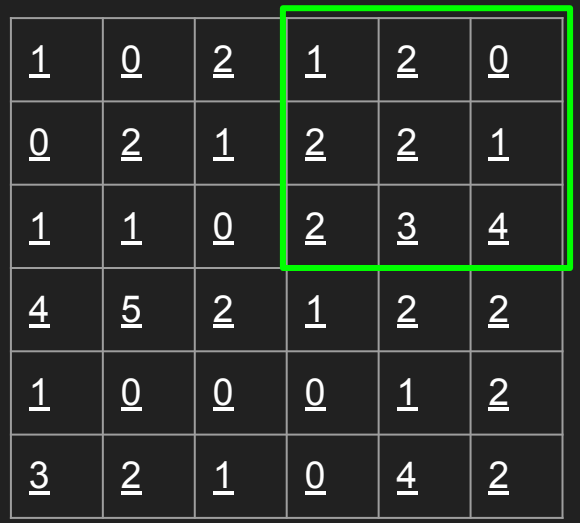
What is a 2D convolution layer?

- image input
- filter (a.k.a. kernel)
- dot product
- move along
- repeat
- outputs a new layer



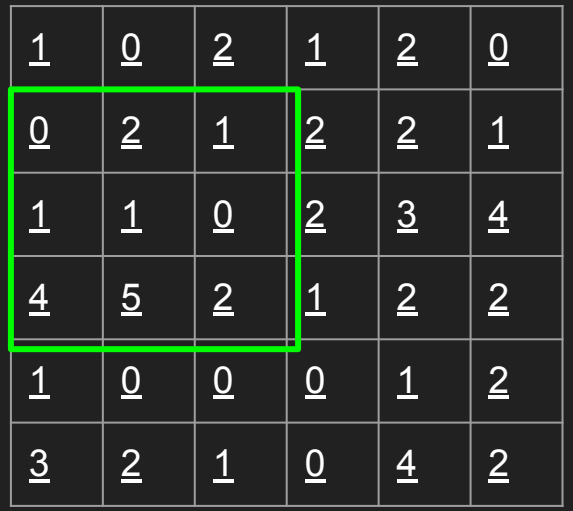
What is a 2D convolution layer?

- image input
- filter (a.k.a. kernel)
- dot product
- move along
- repeat
- outputs a new layer



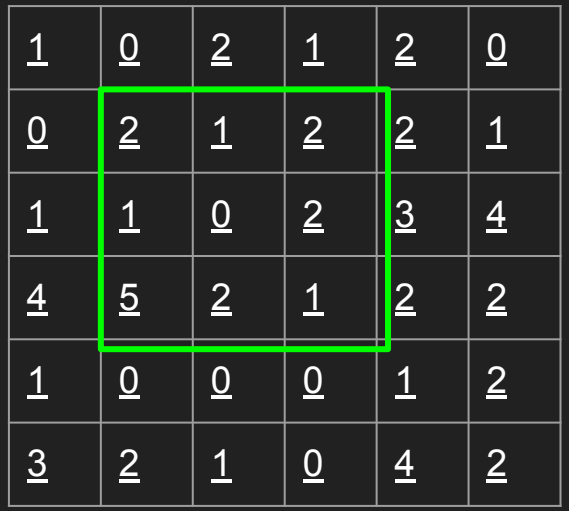
What is a 2D convolution layer?

- image input
- filter (a.k.a. kernel)
- dot product
- move along
- repeat
- outputs a new layer



What is a 2D convolution layer?

- image input
- filter (a.k.a. kernel)
- dot product
- move along
- repeat
- outputs a new layer



What is a 2D convolution layer?

- image input
- filter (a.k.a. kernel)
- dot product
- move along
- repeat
- outputs a new layer

1	0	2	1	2	0
0	2	1	2	2	1
1	1	0	2	3	4
4	5	2	1	2	2
1	0	0	0	1	2
3	2	1	0	4	2

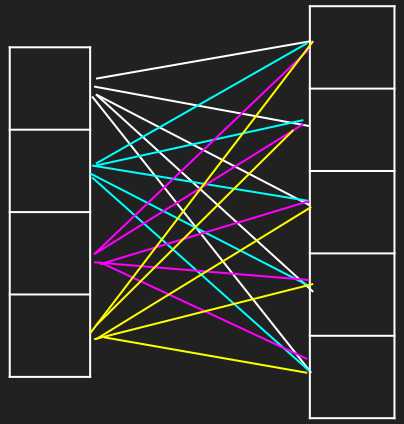
4	7	8	10
9	6	8	13
12	8	7	9
8	3	2	9

What is a 2D convolution layer?

- Neurons are only locally connected (i.e. in width and height)
- Less computationally expensive (images too large for full connections)
- Can find spatial patterns
- Much better at generalized learning (fully connected overfit)

What is a fully connected layer?

- More traditional neural network
- Every single neuron in a layer is connected to every other neuron



Code time!

Code time!

```
# Sequential is a linear stack of layers
from keras.models import Sequential

# this creates a model instance
model = Sequential()

# For an 8-bit image (i.e. from 0 to 255) this will normalize values between -1 to 1.
model.add(Lambda(lambda x: x/127.5-1.0, input_shape=INPUT_SHAPE))
```

Code time!

```
# Sequential is a linear stack of layers
from keras.models import Sequential

# this creates a model instance
model = Sequential()

# For an 8-bit image (i.e. from 0 to 255) this will normalize values between -1 to 1.
model.add(Lambda(lambda x: x/127.5-1.0, input_shape=INPUT_SHAPE))
```

must define input shape
in the first layer

Code time!

```
model.add(Conv2D(24, 5, 5, activation='elu', subsample=(2, 2)))
```

Translation to English..

A 2D convolutional layer with 24 sets of random weights and a 5 by 5 pixel kernel.
Neurons will fire based on an exponential linear unit activation function.
The kernel will skip 2 pixels vertically and horizontally during the convolution.

Code time!

```
model.add(Conv2D(24, 5, 5, activation='elu', subsample=(2, 2)))
model.add(Conv2D(36, 5, 5, activation='elu', subsample=(2, 2)))
model.add(Conv2D(48, 5, 5, activation='elu', subsample=(2, 2)))
model.add(Conv2D(64, 3, 3, activation='elu'))
model.add(Conv2D(64, 3, 3, activation='elu'))
```

A bunch of convolutional layers

Code time!

```
model.add(Dropout(args.keep_prob))
```

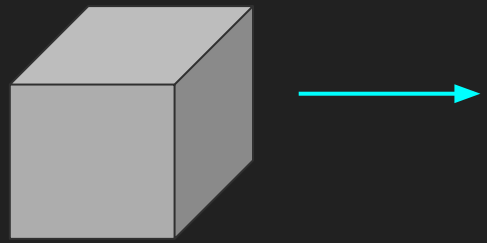
The dropout rate (e.g. 50%) will define how many nodes are randomly ignored

- Helps to prevent over-fitting
- Speeds up training

Essentially it forces the network to not really too heavily on a single set of neurons.

Code time!

```
# Flatten the data the dropout rate
model.add(Flatten())
```



..because we are going to feed in a series of fully connected layers which expect a 1D vector

Code time!

```
model.add(Dense(100, activation='elu'))
model.add(Dense(50, activation='elu'))
model.add(Dense(10, activation='elu'))
model.add(Dense(1))
```

Dense = fully connected layer

Series of fully connected layers condensing the output into a single value, i.e. the steering angle.

Training the model

- Minimize root mean squared error (RMSE)
  - Between model output and human output

- Adam optimizer (gradient descent)
  - Adaptive moment estimation.

Self Driving

Self Driving

Start the 3D simulation

(double click using mouse)

Ready for instructions

(run in autonomous mode)

Self Driving

```
# Move into source code repository

cd {path_to_your_local_repo}


# Start the car brain

python drive.py {model_name.h5}
```

Well Done!

NASCAR Bonus Challenge

Who can drive the fastest?

max speed on line 41 of drive.py