Nanodegree - Lesson 2

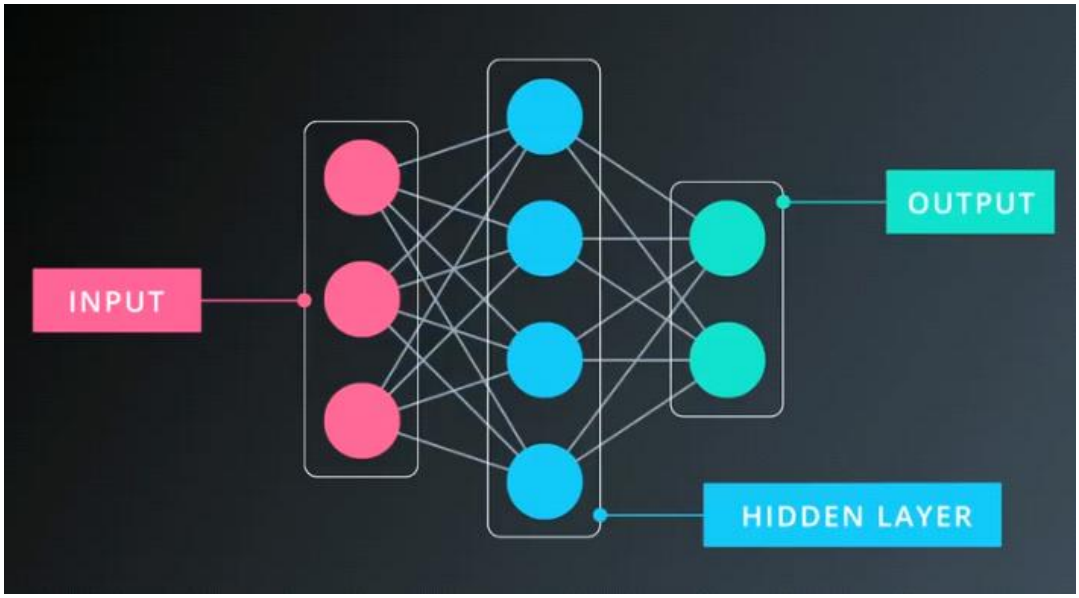
Saturday 16 January 2021 12:18

Style transfer

Screen clipping taken: 16/01/2021 12:18

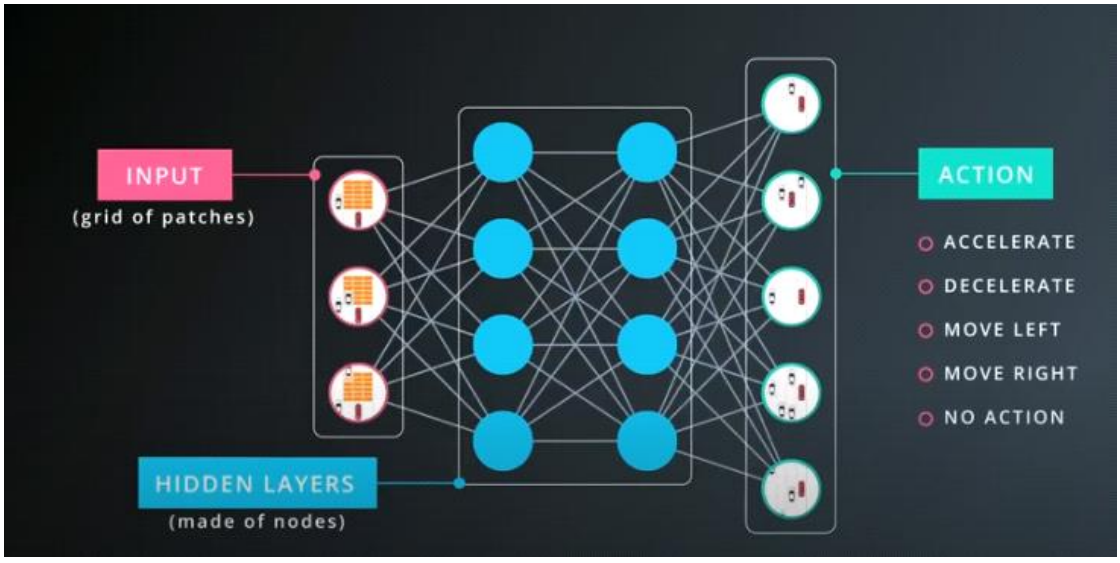
Reinforcement learning

The basic concept behind this is a neural network made of layers.
There are: the input layer, the output layer and the hidden layers. The input layer are the input, the output layer are the decision to take. Then there are some hidden layers that help the network to learn. Each layer learns from the previous layer.



Screen clipping taken: 17/01/2021 18:01

This can be used to self-drive a car. The car needs to be in a line and needs to go on a certain speed, moving without hitting the other cars. The input is a grid ahead of the car and the output are the decision that need to be taken.

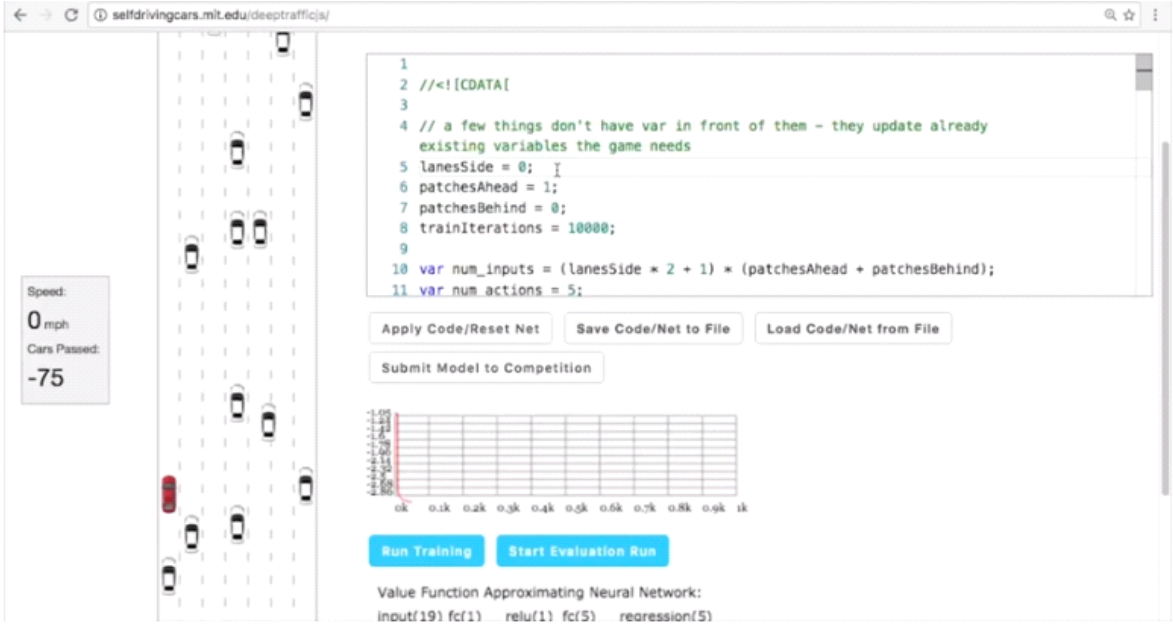


Screen clipping taken: 17/01/2021 18:04

Hi, I'm Cezanne Camacho, I have a Masters degree in electrical engineering from Stanford, which is where I first got into machine and deep learning. I love to think about how humans reason, and how we might replicate that reasoning in algorithms. I'm inspired by those with the curiosity and drive to learn something new! To stay up-to-date with my work, consider following me on Twitter.

The instructor

This is the problem. We can modify the code, especially the input, like the broadness of the grid, then we train the network and evaluate the result.

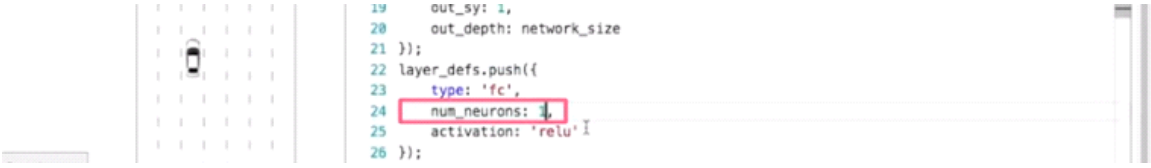


Screen clipping taken: 17/01/2021 18:02

These are the part of the code with the inputs. There is at the moment only one hidden layer, she suggests to increase the number of hidden layers.

Screen clipping taken: 17/01/2021 18:03

This is the part linked to the hidden layers.



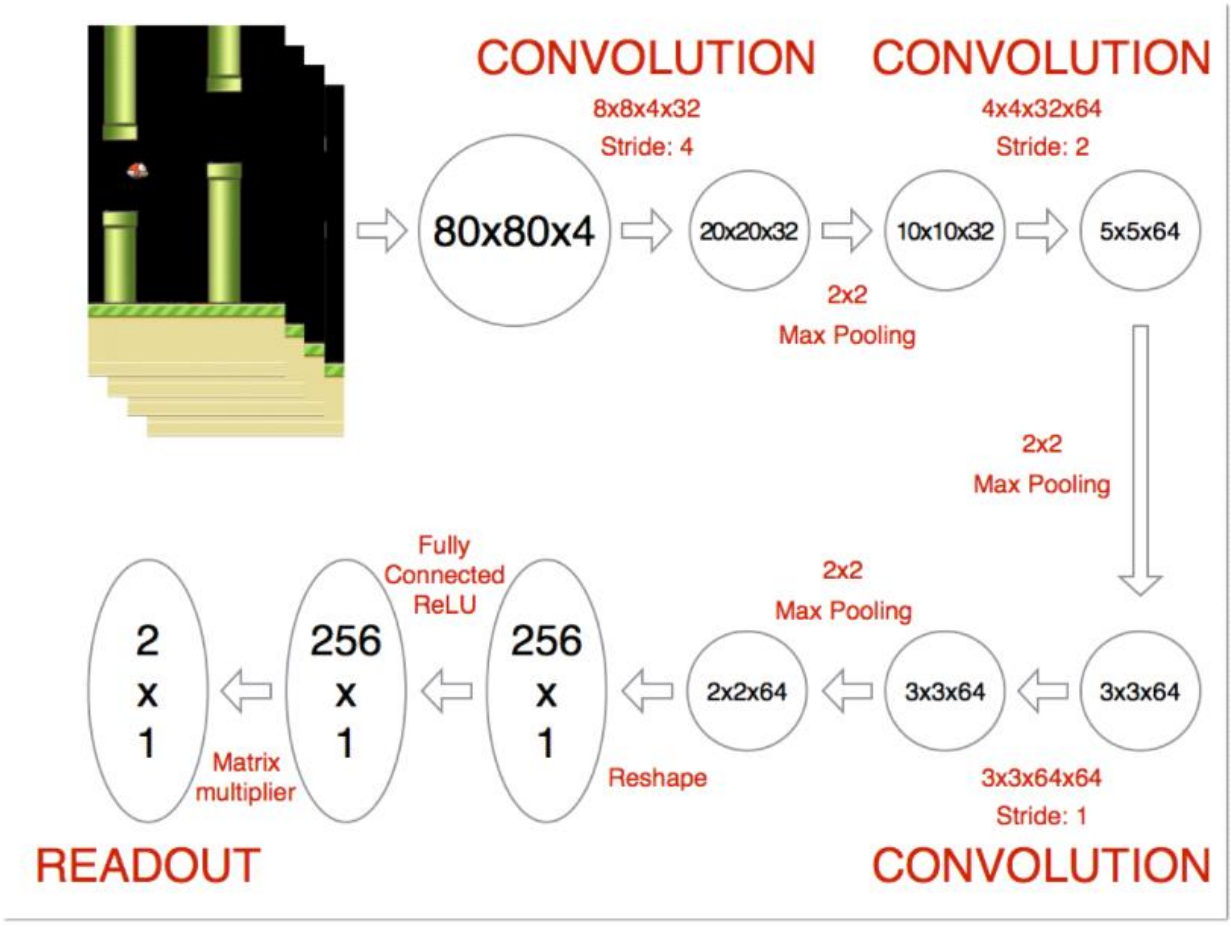
Increase the number of layers

How to build an environment in CONDA

Instructions

1. Install miniconda or anaconda if you have not already.
2. Create an environment for flappybird
   - Mac/Linux: `conda create --name=flappybird python=2.7`
   - Windows: `conda create --name=flappybird python=3.5`
3. Enter your conda environment: `conda activate flappybird`
4. `conda install opencv`
   - If you encounter an error here, you may try an **alternate** download path and *instead* type `conda install --channel https://conda.anaconda.org/menpo opencv3`
5. `pip install pygame`
6. `pip install tensorflow==0.12`
7. `git clone https://github.com/yenchenlin/DeepLearningFlappyBird.git`
   - If you don't have git installed, you can download and extract the zip archive directly from the repository
8. `cd DeepLearningFlappyBird`
   - If you downloaded the archive, you will need to navigate to the extracted folder **DeepLearningFlappyBird-master** instead
9. `python deep_q_network.py`

Screen clipping taken: 17/01/2021 18:34



Screen clipping taken: 17/01/2021 18:38

Sharing Environments

When sharing your code on GitHub, it's good practice to make an environment file and include it in the repository. You can do this using `conda` as:

```
conda env export > environment.yaml
```

Screen clipping taken: 17/01/2021 19:13

Share the List of Dependencies

For users not using conda, you may want to share the list of packages installed in the current environment. You can use `pip` to generate such a list as `requirements.txt` file using:

```
pip freeze > requirements.txt
```

Later, you can share this `requirements.txt` file with other users over Github. Once a user (or yourself) switches to another environment, you can install all the packages mentioned in the `requirements.txt` file using:

```
pip install -r requirements.txt
```

Screen clipping taken: 17/01/2021 19:13

The main breakage between Python 2 and 3

For the most part, the Python 2 code will work with Python 3. Of course, most new features introduced with Python 3 versions won't be backward compatible. The place where your Python 2 code will fail most often is the `print` statement. See the change in the syntax below:

```
# Print statement in Python 2
print "Hello", "world!"

# Print statement in Python 3
print("Hello", "world!")
```

If you want your `print()` function to work in both Python 2 and 3 versions, you'll need to import the `print_function` in your Python 2.6+ code. The `print()` function was backported to Python 2.6+ through the `__future__` module:

```
# Python 3 `print()` function can run in Python 2.6+ after an `import` statement.
# In Python 2.6+
from __future__ import print_function
print("Hello", "world!")

# Python 2 `print` statement cannot run in Python 3.
# The following line of code will NOT work in Python 3
print "Hello", "world!"
```

Screen clipping taken: 17/01/2021 19:15