# Neocognitron

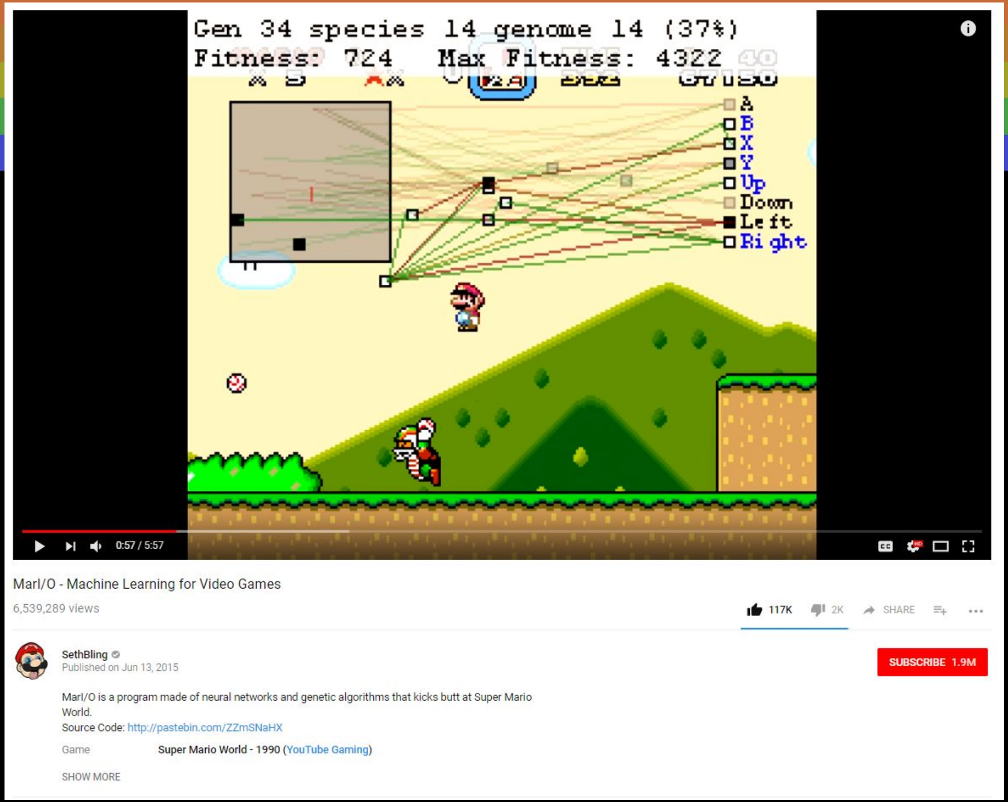
Adam Mischke, Patrick Howell, Eric Sciullo
Kevin Lutz, Benjamin H. Buehler, Garrett Coulter

Making a Neural Network that plays
Breakout


011
2
1

# Motivations

- Many students now studying CS have an appreciation of video games.
- Having a neural net play a video game is a task more concrete than analysis of data.
- Reinforcement learning can be used for video game playing as shown by Google DeepMind. [Mnih et al.]



# Influences

- SethBling taught a neural net called MarI/O to play Super Mario World and a net called MariFlow to play Mario Kart. [https://www.polygon.com/2017/11/5/16610012/mario-kart-mariflow-neural-network-video]
- Google DeepMind taught a neural net to play 49 Atari games using the same reinforcement DQN topology for each network. [Mnih et al.]
- DeepMind also created a reinforcement net to play Go at a superhuman level. [Silver et al.]
- Two students at Carnegie Mellon University taught a reinforcement net to play Doom deathmatch. [Lample and Chaplot]

# Aims

- Teach a neural net to play Breakout using pixel data;
  - Would need to
    - Interpret and store visual data
      - [memory=1,000,000, height=84, width=84, frames=4 or 5] uint8, float32
    - Determine appropriate actions
      - [no-op, fire, move left, move right] -> [no-op, move left, move right]
- Observe the learning process of the neural net;

# Code

- We used an approach similar to that of Mnih et al.'s Atari networks.
  - Preprocessing
    - Cut input resolution
      - When training: normalize frame batches, float32
      - When storing: downsample, grayscale, uint8
    - Input only every fourth frame and repeat chosen action for the next three frames
      - Deterministic/NoFrameSkip
  - State-action (Q) Function based reinforcement learning
  - 3 Convolutional layers to process input
  - Epsilon greedy exploration approach
    - Linear annealing for the first 1 million frames

# Code pt. 2

- We used an approach similar to that of Mnih et al.'s Atari networks.
    - Reward Schedule
        - Clipping the reward to (-1, 1)
            - Reward is never negative
            - Reward is reset every death
    - No-op max
        - Up to 30 frames of no-ops to diversify initial frame states
    - Loss Function
        - Huber (less resistant to outlier data), logcosh, mean squared error
    - Lots of hyperparameters:

# Hyperparameters!

**Breakout Main Loop:**

```
'GAME' : 'BreakoutDeterministic-v4', # Name of
                                     # v1-4 Det
'DISCRETE_FRAMING' : True,     # 2 discrete set
'LOAD_WEIGHTS' : '',           # Loads weights
                               # leave '' if st
'RENDER_ENV' : False,          # shows the scre
                               # massivly slows
                               # default: False
'HEIGHT' : 84,                 # height in pixe
'WIDTH'  : 84,                 # and width in p
                               # defaults: 84,
'FRAME_SKIP_SIZE' : 4,         # how many frame
                               # choose an acti
                               # default: 4
'MAX_EPISODES' : 12000,        # defined as how
                               # winning a roun
                               # default: 12,00
'MAX_FRAMES' : 50000000,       # max number of
                               # default: 50,00
'SAVE_MODEL' : 500,            # how many episo
                               # default: whene
'TARGET_UPDATE' : 10000,       # on what mod epochs should we upda
                               # default: 10000
```

**DQNAgent:**

```
'WATCH_Q' : False,             # watch the Q function and see what decision it picks
'LEARNING_RATE
'INIT_EXPLORATI
'EXPLORATION'
'MIN_EXPLORATI
'OPTIMIZER' :
'MIN_SQUARED_GF
'GRADIENT_MOMEN
'LOSS' : 'huber
'NO-OP_MAX' : 3
```

**Replay and Remember Memory:**

```
'SHOW_FIT' : 0,                # shows the fit of the model and it's work, turn to 0 for off
                               # default: 0 for off

'REPLAY_START' : 50000,        # when to start using replay to update the model
                               # default: 50000 frames

'MEMORY_SIZE' : 1000000,       # size of the memory bank
                               # default: 1,000,000

'GAMMA' : 0.99,                # integration of rewards, discount factor,
                               # preference for present rewards as opposed to future rewards
                               # default: 0.99
# 4 * 8 = 32 batch
'REPLAY_ITERATIONS' : 4,       # how many ireration of replay
                               # default: 4

'BATCH_SIZE' : 8               # batch size used to learn
                               # default: 8
```

# Results

- The first successful results came from training over 1 million frames using an NVIDIA 1080 GPU. This yielded an average reward/score  of around 40. This training took a little over 10 hours to complete.

- Training takes a while! We are still training to achieve a higher average reward. An instance is training right now that is working to train over 20 million frames rather than the previous 1 million. This training is expected to be complete in two or three more days. The current average reward is around 70.

- Even though it has not completely cleared the game yet, it has achieved a high score of 215 and on average, exceeds the playing abilities of many human players.

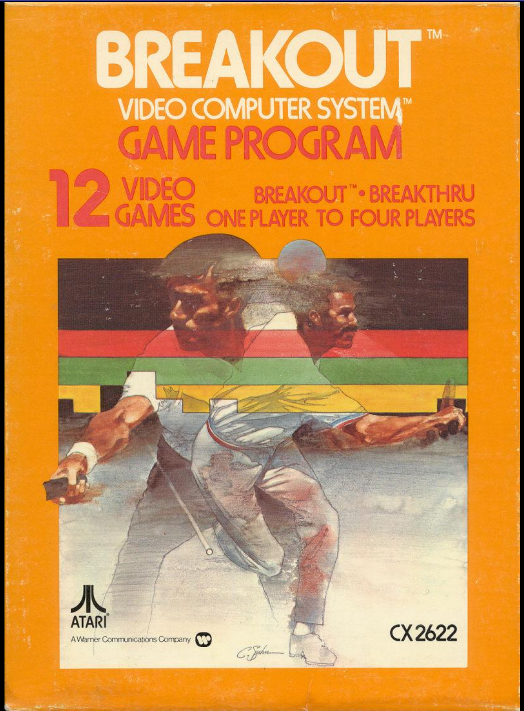
# Building the Demo

- We are using the weights from the training to play a game of breakout.

- Acknowledgement: Dr. Phillips contributed some of the code for the demo.

# Administrative Team

# To The Demo!


BREAKOUT™
VIDEO COMPUTER SYSTEM™
GAME PROGRAM
12 VIDEO GAMES
BREAKOUT™ • BREAKTHRU
ONE PLAYER TO FOUR PLAYERS
ATARI®
A Warner Communications Company
CX 2622