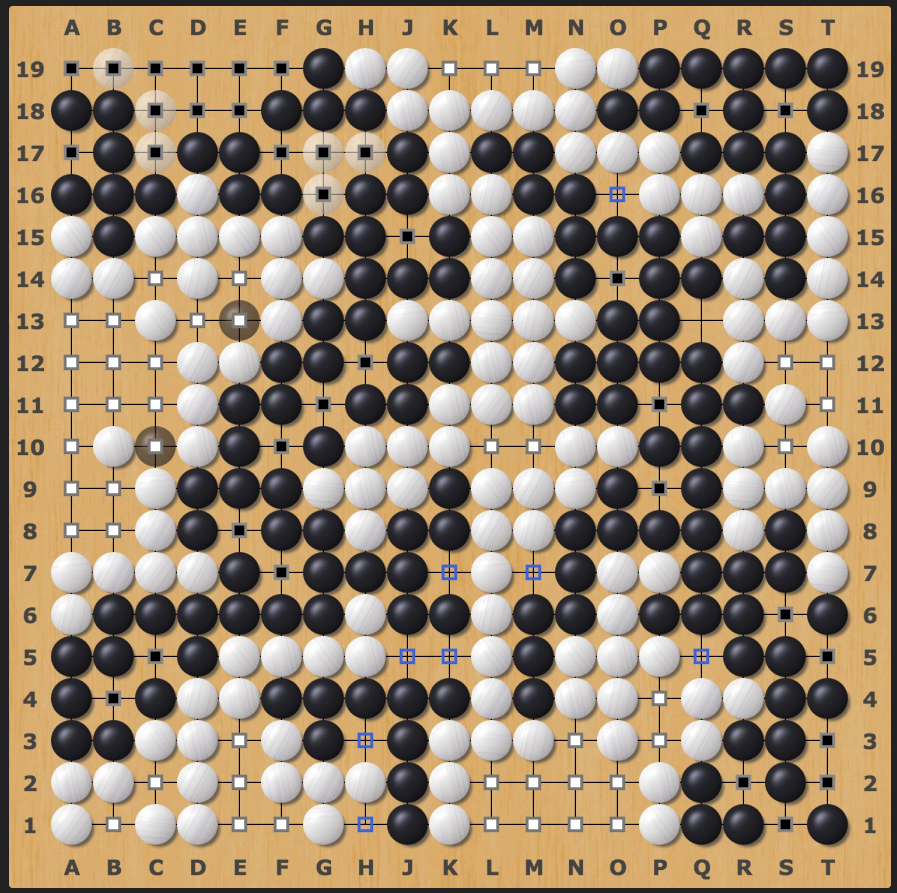
They Grow Up So Fast

a technical story about how robots learned to play

Go, Baduk, Weiqi



source: online-go.com

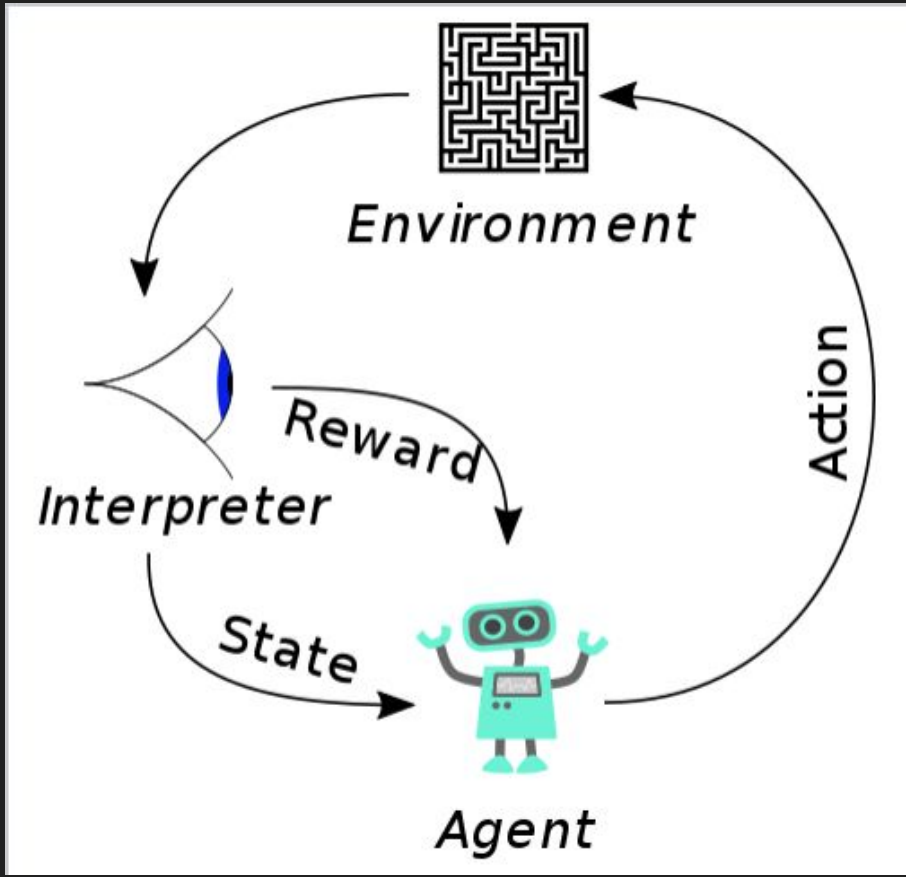
- What is it?
  - A perfect information, deterministic, discrete combinatorial game
  - An ancient cultural relic from one of the cradles of modern civilization
- Why is it the center of attention in AI?
  - It is easily formalised: Tromp-Taylor rules
  - The search space is way-way-way too big to do an exhaustive search
    - Possible games: ~2.082 × 10^170
    - Chess: 2 × 10^120
  - Fitness function readily available - just play games

Dota

- What is it?
  - A non-perfect information, stochastic, continuous game
  - A popular e-sport with many million dollars in prizes, also a cultural icon
- How did this also become an AI game?
  - Evolution in AI is extremely fast
  - AI is being rapidly generalized
  - Learning methods more and more focused

Reinforcement Learning

- What is it?
  - A wide field of machine learning concerned with maximizing a reward in a dynamic environment
  - Reward is automatically gained by doing the right actions
  - Action results can be observed
- Where is it used?
  - Robotics
  - Chemistry
  - Civil engineering
  - Competitive games

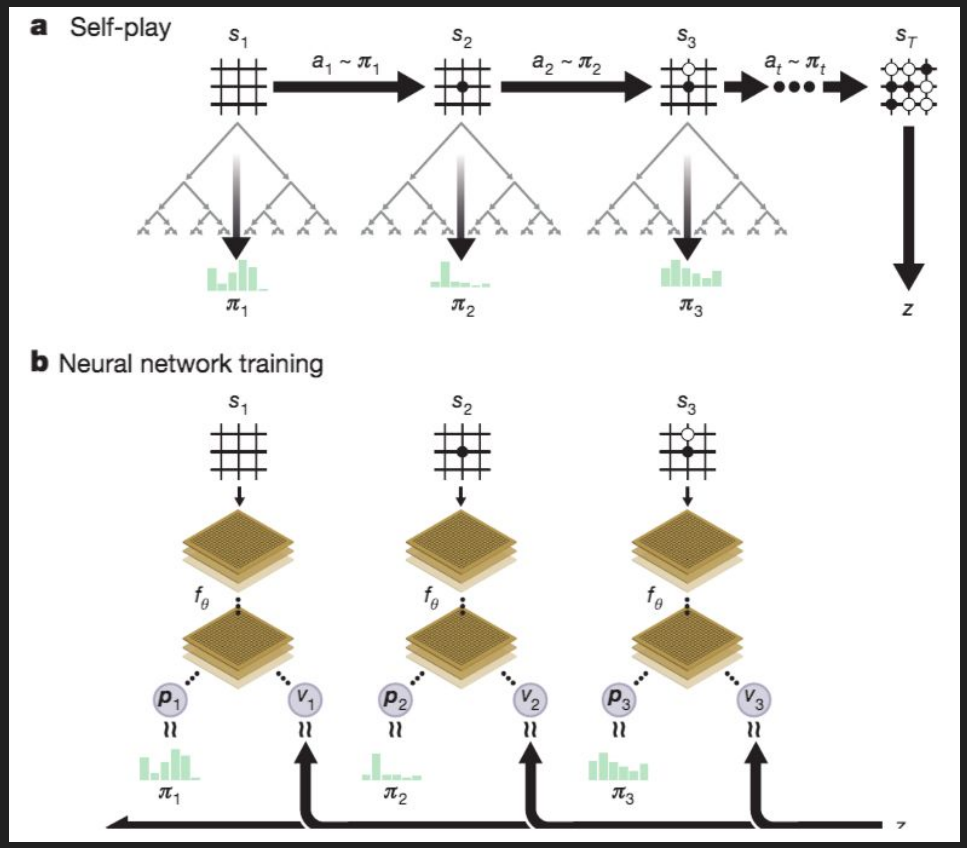


source: Wikipedia

A Story of RL Evolution

AlphaGo	AlphaGo Zero	AlphaZero	OpenAI Five
Go	Go	Go, Chess, Shogi	Dota
Initial supervision	No supervision	No supervision	Proximal Policy Optimization
Reinforcement	Gated Reinforcement	Ungated Reinforcement	Engineered features
Engineered features	Raw features	Raw features	Raw network
MC rollouts	MCTS	MCTS	

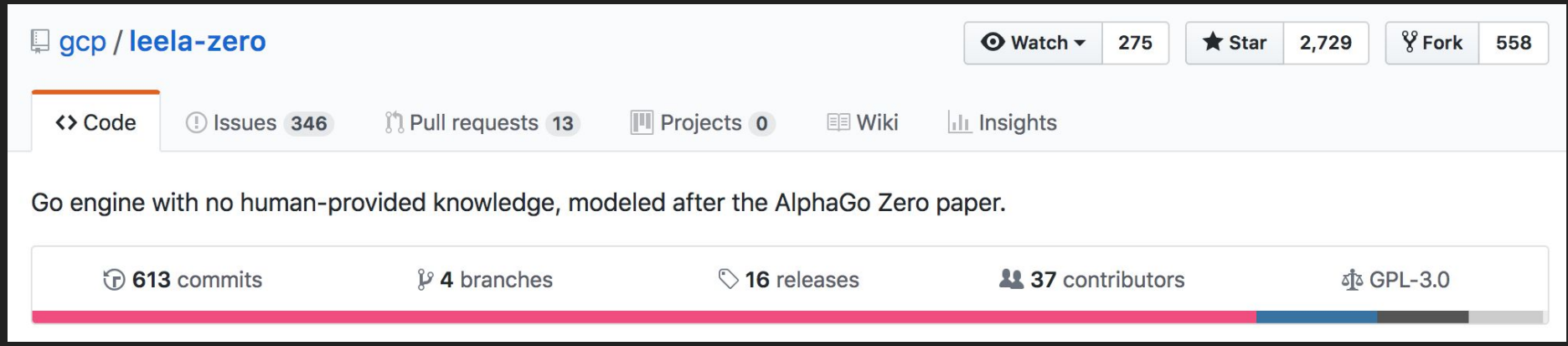
AlphaGo Zero



source: AlphaGo Zero paper

- What is it?
  - An implementation of RL Go
- Who made it?
  - Deepmind, acquired by Google
  - Paper by Demis Hassabis et al.
- How was it made?
  - Deep residual neural network utilising special hardware (TPUs)
  - Learned from self-play only
  - Utilizes a cycle of policy improvement and policy evaluation operators

Leela Zero



- What is it?
  - An open source recreation of the AlphaGo Zero experiment
  - A community resourced* RL project
- What techniques does it display?
  - Reinforcement learning
  - Transfer learning
  - Community resourcing

Transfer learning

- Transfer learning is a way to train a large neural network efficiently
- Net2net explores this idea
- Original paper by T. Chen, I. Goodfellow, and J. Shlens (Google)
- Naive approach: take slices of the teacher (small) network as input
  - Experimentally doesn't fulfil the goal
  - Idea: Function preserving initialization. New, larger network should have same initial output
- Net2WiderNet
  - Split individual neurons into two, halve their weights, introduce noise
- Net2DeeperNet
  - Introduce matching size layers that are trained to act as identity function
- Both provide learning speedups, initially W is better, then later D

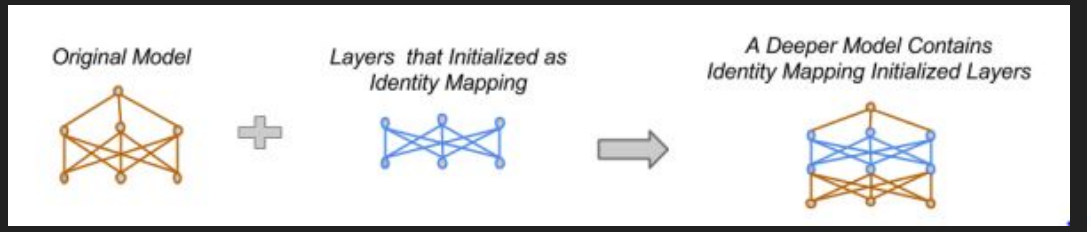
Transfer learning

https://github.com/gcp/leela-zero/blob/master/training/tf/net2net.py

```
parser = argparse.ArgumentParser(description='Add filters/blocks to existing network such that the output is preserved.')
parser.add_argument("blocks", help="Residual blocks to add", type=int)
parser.add_argument("filters", help="Filters to add", type=int)
parser.add_argument("network", help="Input network", type=str)
parser.add_argument("--noise", nargs='?', help="Standard deviation of noise to add to new filters/blocks. Default: 5e-3",
        default=5e-3, type=float)
parser.add_argument("--dir_alpha", nargs='?', help=\
    """Dirichlet distribution parameter for input weight distribution for replicated channels. """\
    """Larger values divide input values more equally. """\
    """Smaller ones give one large input weight while others are very small. """\
    """You probably want this to be at least 1 to avoid near zero weights. """\
    """Set to 0 to divide input weights equally. Default: 10""",
    default=10, type=float)
parser.add_argument("--verify", help="Verify that output matches. Noise must be disabled.",
        default=False, action='store_true')
parser.add_argument("--add_inputs", help="Adds input planes to network",
        default=0, type=int)
```

Transfer learning

Deepening the network

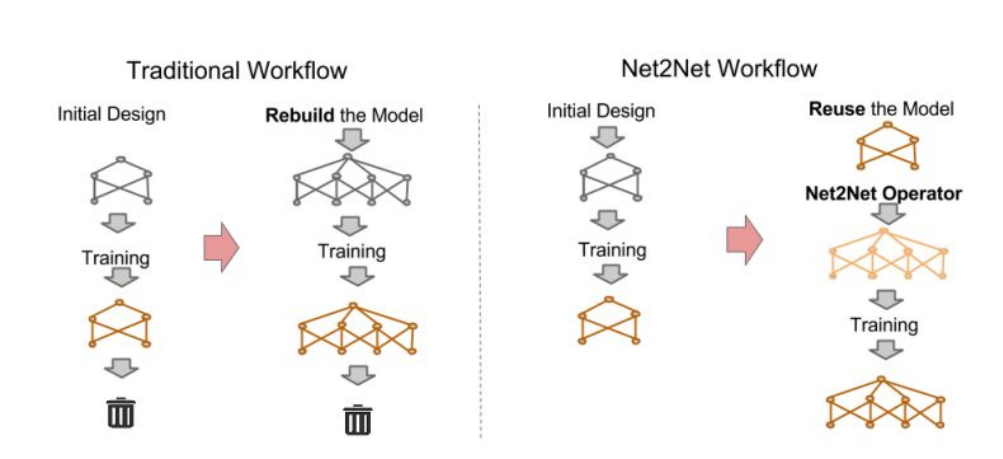


source: Net2net paper

```
print("Output will have {} blocks and {} channels.".format(
    blocks+new_blocks, channels+new_channels))

input_planes = 18

#Input convolution, bias, batch norm means, batch norm variances
w_input = weights[:4]

#Residual block convolution + batch norm
w_convs = []
for b in range(2*blocks):
    w_convs.append(weights[4 + b*4: 4 + (b+1)*4])

i = ((b+1)*4) + 4
w_pol = weights[i:i+6]
w_val = weights[i+6:]

if new_blocks > 0:
    #New blocks must have zero output due to the residual connection
    new_block_conv = np.random.normal(0, noise_std, 9*(channels)**2)
    new_block_bias = np.zeros(channels)
    new_block_bn_mean = new_block_bias.copy()
    new_block_bn_variances = np.ones(channels)
    new_block = [new_block_conv, new_block_bias, new_block_bn_mean, new_block_bn_variances]

    for i in range(2*new_blocks):
        w_convs.append(deepcopy(new_block))

    blocks += new_blocks
```

Transfer learning

Widening the network



source: Net2net paper

```
def conv_bn_wider(weights, next_weights, inputs, channels,
                  new_channels, noise_std=0, last_block=False,
                  rand=None, dir_alpha=None, verify=False):

    if new_channels == 0:
        return weights, next_weights

    if rand == None:
        rand = list(range(channels))
        rand.extend(np.random.randint(0, channels, new_channels))
    rep_factor = np.bincount(rand)

    factor = np.zeros(len(rand))

    #In the net2net paper every input weight was weighted equally,
    #but in general we can have unequal division of the weights
    if dir_alpha == None:
        #Equal division
        for i in range(len(rand)):
            factor[i] = 1.0/rep_factor[rand[i]]
    else:
        #Unequal input weighting determined by dirichlet distribution
        for i in range(channels):
            x = np.random.dirichlet([dir_alpha]*rep_factor[i])
            e = 0
            for j in range(channels + new_channels):
                if rand[j] == i:
                    factor[j] = x[e]
                    e += 1
```

Transfer learning

Tying it all up

```
for e, w in enumerate(w_convs[:-1]):
    r = rand
    if e % 2 == 0:
        print("Processing block", 1 + e//2)
        #First convolution in residual block can be widened randomly
        r = None

    w_wider, conv_next = conv_bn_wider(w, [w_convs[e+1][0]], channels + new_channels,
            channels, new_channels, noise_std, rand=r, dir_alpha=dir_alpha, verify=verify)
    w_convs[e+1][0] = conv_next[0]
    write_layer(w_wider, out_file)

#The last block is special case because of policy and value heads
w_wider, w_next = conv_bn_wider(w_convs[-1], [w_pol[0], w_val[0]], channels + new_channels,
        channels, new_channels, noise_std, last_block=True, rand=rand, dir_alpha=dir_alpha, verify=verify)
w_pol[0] = w_next[0]
w_val[0] = w_next[1]

write_layer(w_wider, out_file)

write_layer(w_pol, out_file)
write_layer(w_val, out_file)

out_file.close()
```

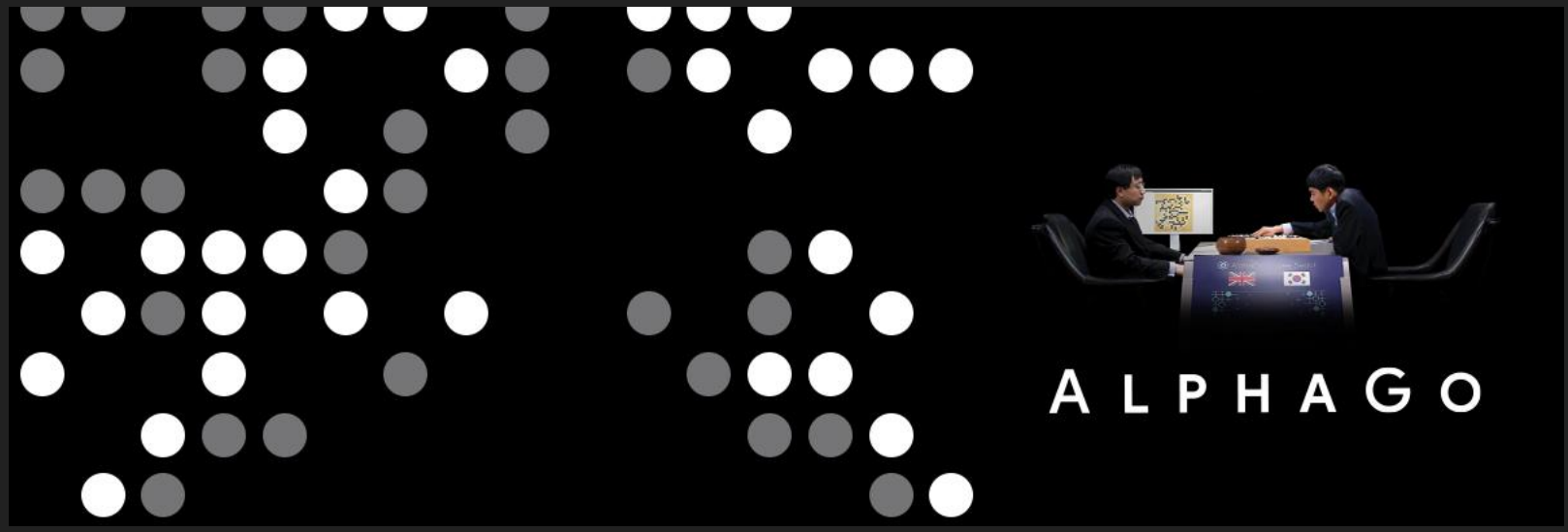
Pointers

- AlphaGo Zero paper
- Net2net paper
- Leela Zero on GitHub
- AlphaZero paper
- OpenAI Five

source: xkcd.com

Debate starters

- Google didn't release the source code of the AlphaGo family of experiments because this way they have generated more hype.



source: ualberta.ca

Debate starters

- AI will outperform humans in all competitive fields because it can train against itself much faster and more efficiently than humans can.

source: deeplearningskysthelimit.blogspot.com

Debate starters

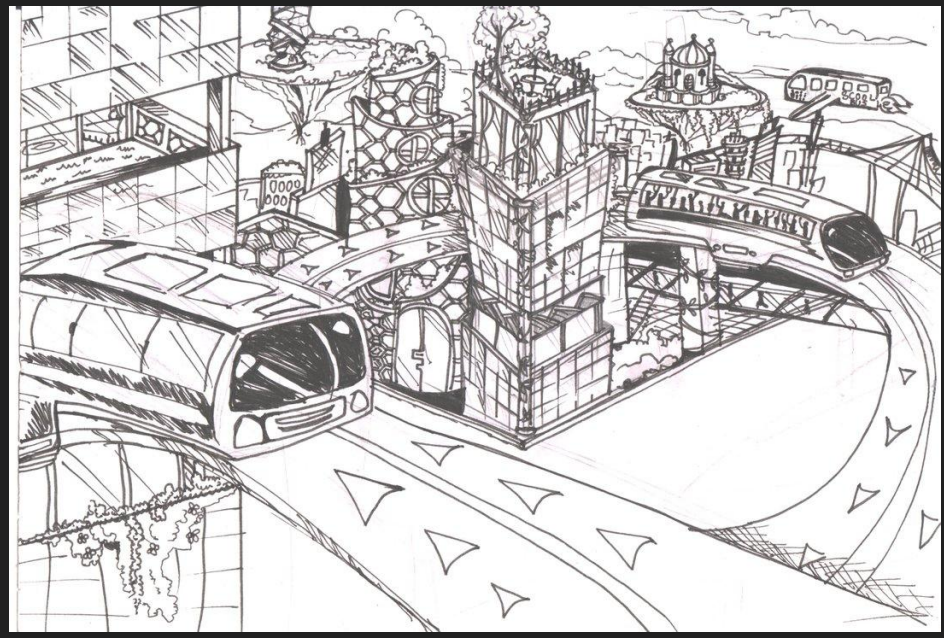
- We should not be worried of robots taking our jobs; we should be automating all jobs and come up with new ideas to redistribute wealth.

source: thelocal.se

The show is only starting now...

Peter Ferenczy

peter@peoplelostin.space



source: nanciro on DeviantArt