General overview of the methodology
Transfer learning

• Training a new model from scratch for each problem is wasteful.

• Much better to train a model on one task, and then reuse this knowledge on other related tasks as well.

• Allows neural networks to learn well and faster on very small datasets (even on a couple of hundred samples).
Usual way of doing transfer learning

1. Train on Imagenet

2. Small dataset: **feature extractor**

   - Freeze these

3. Medium dataset: **finetuning**

   - more data = retrain more of the network (or all of it)
   - Freeze these

   - Train this
PathNet’s way of doing transfer learning

- The authors propose a giant neural network that has many possible paths from input to output.
- The network can choose which path to use on any given task.
- For example, if it first learns to recognize dogs, it uses one path. Then, when learning to recognize cats, it can use another path, which may be partially overlapping (as opposed to fully overlapping).
PathNet’s architecture

- Modular deep neural network with $L$ layers, each layer consisting of $M$ modules.
- Each module is a small neural network.
- For each layer, the outputs of the modules are summed before being passed into the active modules of the next layer.
- A module is active if it is present in the path currently being evaluated.
- Each path can have at most $N$ modules per layer.
- The final layer is unique and unshared for each task being learned.
How are the paths computed?

1. $P$ paths are initialized randomly, where each path is a $N \times L$ matrix of integers.
2. A random path is chosen and trained for $T$ epochs. During training, we **calculate the fitness of the path**.
3. Then another random path is chosen and again trained for $T$ epochs and its fitness evaluated.
4. The 2 paths are compared and the **least fit path is overwritten** by the winning path.
5. Go over each element in the winning path’s matrix and with probability of $1/(N \times L)$, add an integer in the range of $[-2, 2]$ to it.
Videos

- **Normal:**
  https://www.youtube.com/watch?v=o9r-Z-sibS0&index=1&list=PLKhLdFXp1JN5sHZ0vJEu0jWF2Rsb20NBv

- **No convergence:**
  https://www.youtube.com/watch?v=Wkz4bG_JlcU&list=PLKhLdFXp1JN5sHZ0vJEu0jWF2Rsb20NBv&index=2

- **Fast convergence:**
  https://www.youtube.com/watch?v=7fHN5zA7R3o&index=3&list=PLKhLdFXp1JN5sHZ0vJEu0jWF2Rsb20NBv
Specific experiments
Binary MNIST classification
(Supervised learning)
Parameters for the MNIST classification

- Paths are **evolved until near perfect classification** (99.8%) on the training set is achieved.
- \( L = 3 \) layers.
- Each layer contains \( M = 10 \) modules.
- Each module contains 20 ReLU units.
- Maximum of 3 modules per layer may be chosen.
- A population of 64 paths were generated randomly at the start of both tasks.
- The evaluation of one path involves training 50 mini-batches of size 16.
- The total number of maximum parameters in one path is:
  \[
  (28 \times 28) \times 20 \times 3 + 20 \times 20 \times 3 + 20 \times 20 \times 3 + 20 \times 2 = 49 480
  \]
PathNet learns faster than regular networks

PathNet

Generations to achieve 0.998 accuracy
Correlation between speedup and path overlap

- Speedup Ratio - independent control training time / PathNet training time.
- Overlap measure - the number of modules in the original optimal path that were present in the population of paths at the end of the second task.
CIFAR and SVHN classification

(Supervised learning)
PathNet learns better than regular networks

- $L = 3$ layers.
- $M = 20$ modules per layer.
- 20 neurons in each module.
- Maximum of 5 modules per layer may be chosen.
- Networks were trained for a fixed period of 500 generations, where one generation was the evaluation of 2 pathways.
Games

(Reinforcement learning)
PathNet in reinforcement learning

1. 64 paths are initialised randomly and their fitness of $-\infty$ stored in a central parameter server.
2. All paths are evaluated \textit{in parallel}, the fitness of a path is the reward accumulated over $T (=10)$ episodes using the given path.
3. Once a path is evaluated, it chooses $B (=20)$ random paths from the central server and compares its fitness to each of the $B$ paths. If it finds a path that is more fit, it overwrites itself with this path with some probability of mutation.
PathNet architecture for RL

- 4 layers.
- 10 or 15 modules per layer.
- 8 rectangular kernels per CNN module.
- Fully connected ReLU layers of 50 neurons each in the final layer.
- Typically a maximum of 4 modules per layer are permitted to be included in a path.
Atari games

- The plots show learning curves of models which have already learned RiverRaid for 80M timesteps.
  - Blue - PathNet
  - Red - independent learning
  - Green - fine-tuning
- Results from the best five hyperparameter settings are shown.
Atari transfer matrix

<table>
<thead>
<tr>
<th></th>
<th>Alien</th>
<th>Asterix</th>
<th>Boxing</th>
<th>Centipede</th>
<th>Gopher</th>
<th>Hero</th>
<th>JamesBond</th>
<th>Krull</th>
<th>Road_Runner</th>
<th>Robotank</th>
<th>Star_gunner</th>
<th>wizard_of_wor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Control</strong></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Pong</strong></td>
<td>0.83</td>
<td>1.48</td>
<td>1.09</td>
<td>1.23</td>
<td>1.07</td>
<td>1.49</td>
<td>1.01</td>
<td>1.09</td>
<td>1.03</td>
<td>0.85</td>
<td>0.95</td>
<td>0.90</td>
</tr>
<tr>
<td><strong>Riverraid</strong></td>
<td>0.89</td>
<td>0.80</td>
<td>1.57</td>
<td>1.36</td>
<td>1.31</td>
<td>0.85</td>
<td>2.59</td>
<td>1.15</td>
<td>0.90</td>
<td>4.75</td>
<td>0.46</td>
<td>1.07</td>
</tr>
<tr>
<td><strong>Seaquest</strong></td>
<td>0.87</td>
<td>1.23</td>
<td>1.23</td>
<td>1.37</td>
<td>1.00</td>
<td>0.87</td>
<td>0.95</td>
<td>1.08</td>
<td>0.60</td>
<td>0.79</td>
<td>0.24</td>
<td>0.56</td>
</tr>
<tr>
<td><strong>Pong</strong></td>
<td>1.14</td>
<td>0.58</td>
<td>1.35</td>
<td>1.43</td>
<td>1.21</td>
<td>1.14</td>
<td>1.93</td>
<td>1.12</td>
<td>0.96</td>
<td>1.07</td>
<td>0.50</td>
<td>0.56</td>
</tr>
<tr>
<td><strong>Sea Quest</strong></td>
<td>1.15</td>
<td>1.17</td>
<td>1.80</td>
<td>2.22</td>
<td>1.17</td>
<td>1.03</td>
<td>2.04</td>
<td>1.13</td>
<td>0.96</td>
<td>1.22</td>
<td>0.29</td>
<td>0.73</td>
</tr>
<tr>
<td><strong>Riverraid</strong></td>
<td>1.00</td>
<td>0.79</td>
<td>1.71</td>
<td>2.15</td>
<td>1.04</td>
<td>1.14</td>
<td>2.25</td>
<td>1.11</td>
<td>0.90</td>
<td>6.86</td>
<td>0.25</td>
<td>0.84</td>
</tr>
</tbody>
</table>
Labyrinth games

- A 3D first person game environment.
- PathNet architecture and setup was almost the same as for the Atari games, except for the module duplication mechanism.
- Allows PathNet to copy the weights of the modules to other modules within the same layer.
- Sliding mean over the fitness of any path which contains the module to measure the usefulness of a module.
Labyrinth transfer matrix

- The average performance ratio for fine-tuning across all games is 1.00.
- Avg. performance ratio for PathNet is 1.26.

<table>
<thead>
<tr>
<th></th>
<th>Fixed Path</th>
<th>PathNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>de novo</td>
<td>de novo</td>
</tr>
<tr>
<td></td>
<td>from scratch</td>
<td>from scratch</td>
</tr>
<tr>
<td></td>
<td>lt_chasm</td>
<td>lt_chasm</td>
</tr>
<tr>
<td></td>
<td>stairway_to_melon</td>
<td>stairway_to_melon</td>
</tr>
<tr>
<td></td>
<td>seekavoid_arena</td>
<td>seekavoid_arena</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lt_chasm</td>
<td>1.13</td>
<td>1.05</td>
</tr>
<tr>
<td>stairway_to_melon</td>
<td>0.63</td>
<td>0.85</td>
</tr>
<tr>
<td>seekavoid_arena</td>
<td>1.23</td>
<td>1.11</td>
</tr>
</tbody>
</table>
Conclusion

- PathNet is **capable of transfer learning** in both supervised and reinforcement learning tasks.
- PathNet is **scalable**.
Further work

- Applying PathNet to other RL tasks, e.g. continuous robotic control problems.
- Genetic programming can be replaced with reinforcement learning to evolve the paths.
Thank you.