Gated Path Planning Networks

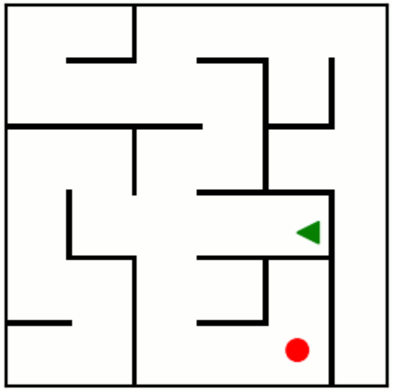
Lisa Lee

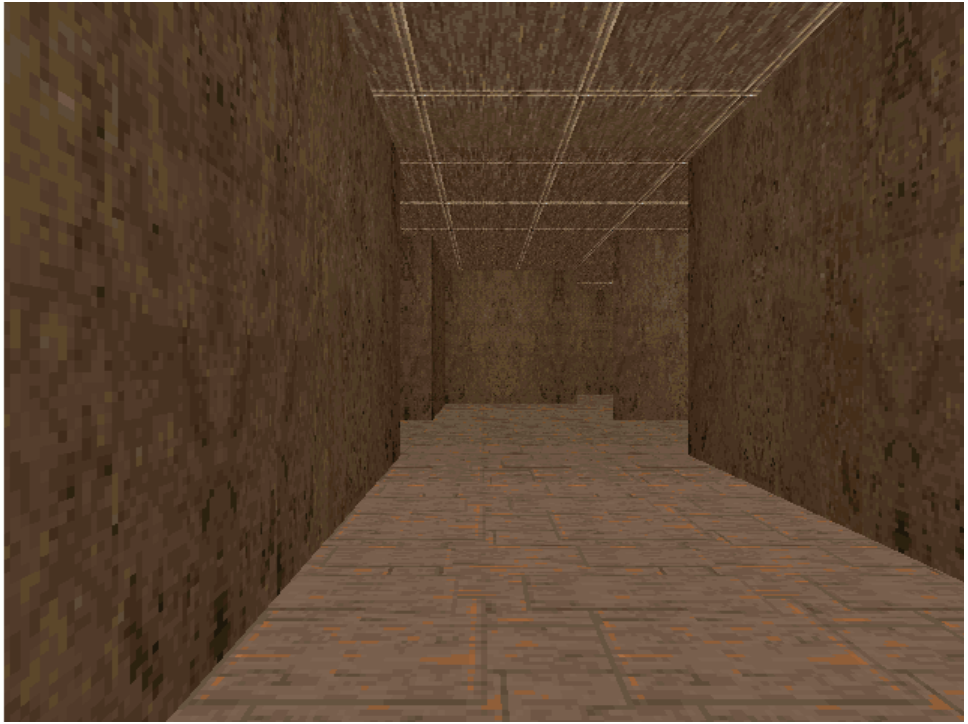
Machine Learning Department
Carnegie Mellon University

Joint work with Emilio Parisotto, Devendra Chaplot, Eric Xing, & Ruslan Salakhutdinov

ICML 2018

Path Planning





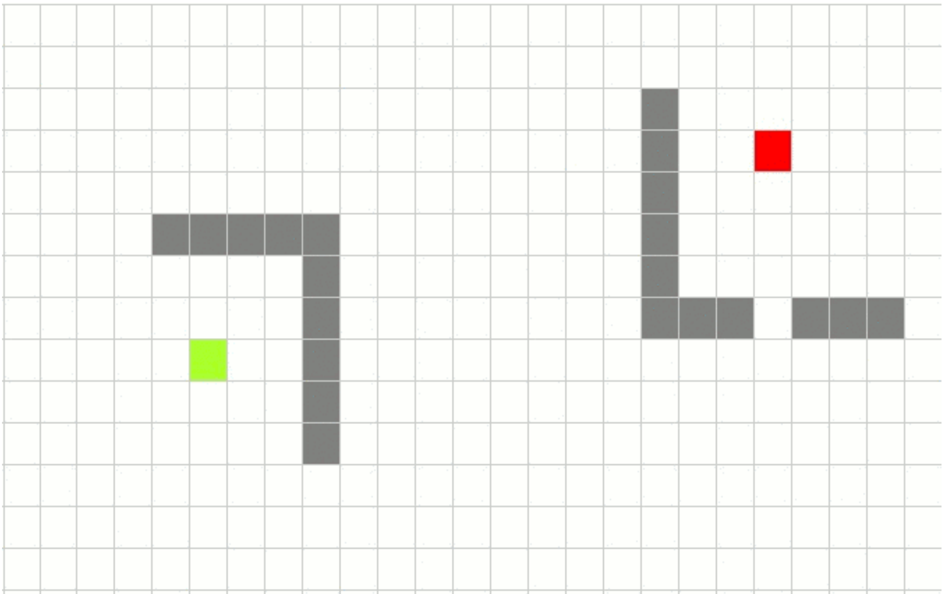
Path Planning is a fundamental part of any application that requires navigation.

- Autonomous vehicles
- Drones
- Factory robots
- Household robots

https://giphy.com/gifs/battlefield-navigate-selfdriving-AmqDSvVwywm7m

Path Planning

A* search (popular heuristic algorithm) ⇒ Not differentiable



Path Planning

Value Iteration Networks (Tamar et al., 2016) ⇒ Fully differentiable!

- Can be used as a path planner module in neural architectures while maintaining end-to-end differentiability.
- VINs have become an important path planner component used in many recent works:
  - QMDP-Net: Deep learning for planning under partial observability (Karkus et al., 2017)
  - Cognitive mapping and planning for visual navigation (Gupta et al., 2017)
  - Unifying map and landmark based representations for visual navigation (Gupta et al., 2017)
  - Memory Augmented Control Networks (Khan et al., 2018)
  - Deep Transfer in RL by Language Grounding (Narasimhan 2017)

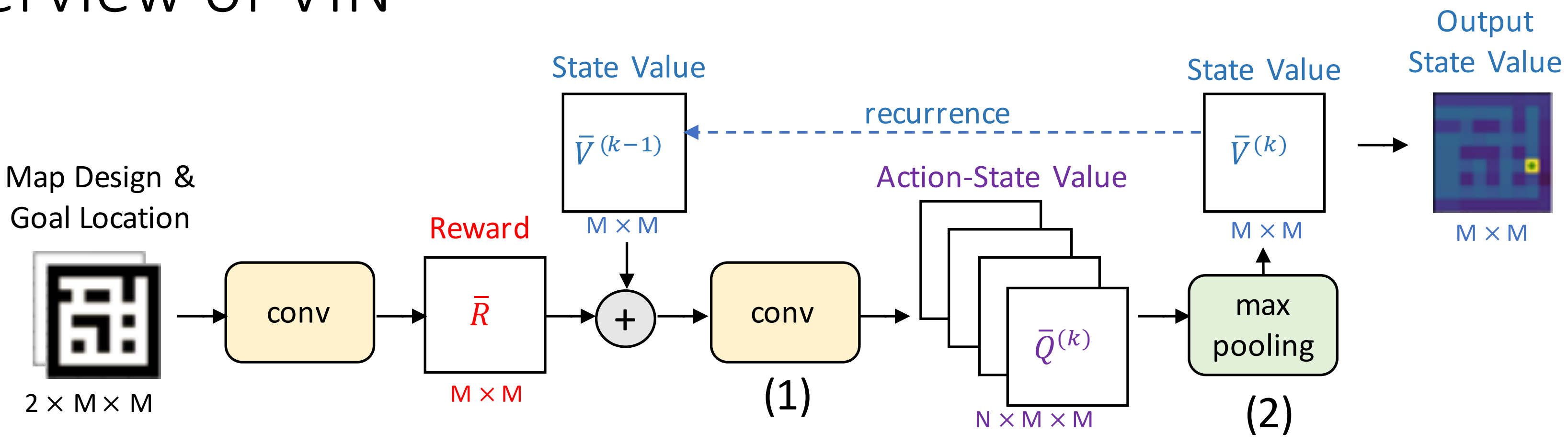
Outline of this talk

Problem: VINs are difficult to optimize.

1. Overview of VIN
2. We reframe VIN as a recurrent-convolutional network.
3. From this perspective, we propose architectural improvements to VIN. ⇒ Gated Path Planning Networks (GPPN)
4. We show that GPPN performs better & alleviates many optimization issues of VIN.

Methods

Overview of VIN



(1) $$\bar{Q}^{(k)}_{\bar{a}} = \overbrace{W^R_{\bar{a}}\,\bar{R}_{[3]} + W^V_{\bar{a}}\,\bar{V}^{(k-1)}_{[3]}}^{\text{Convolution with kernel size 3}}$$

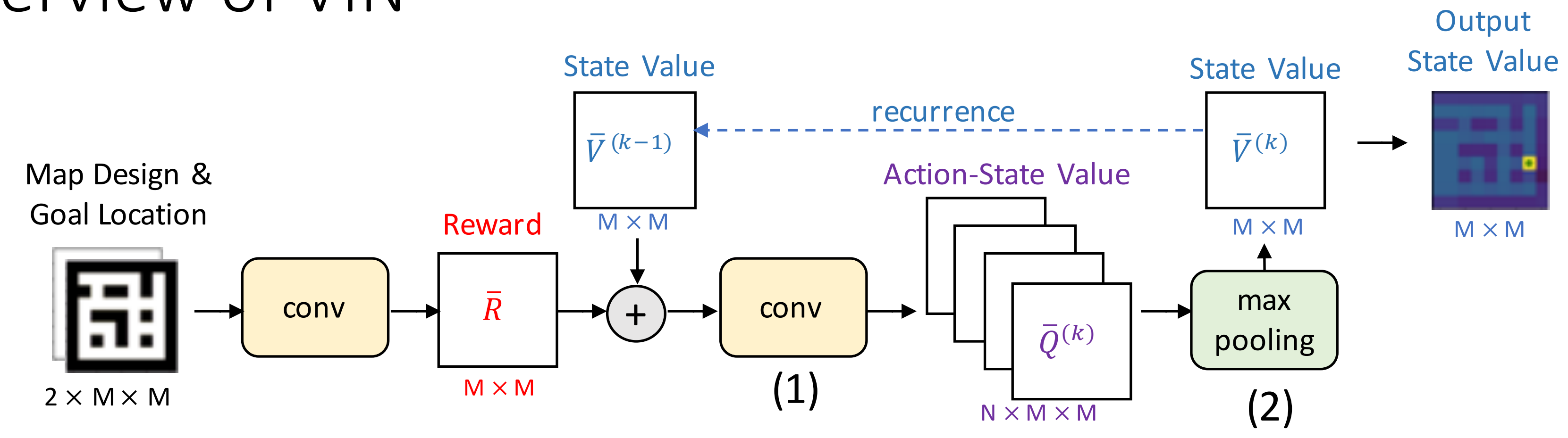
(2) $$\bar{V}^{(k)} = \max_{\bar{a}} \bar{Q}^{(k)}_{\bar{a}}$$

Value Iteration (Bellman, 1957)

$$Q^{(k)}(s,a) = \sum_{i',j'} P(s'|s,a)\big(R(s,a,s') + \gamma V^{(k-1)}(s')\big)$$

$$V^{(k)}(s) = \max_a Q^{(k)}(s,a)$$

Overview of VIN



Recurrent-Convolutional Network with:

- An unconventional nonlinearity (max-pooling)
- Restriction of kernel sizes to 3
- A hidden dimension of 1

Non-gated RNNs are known to be difficult to optimize.

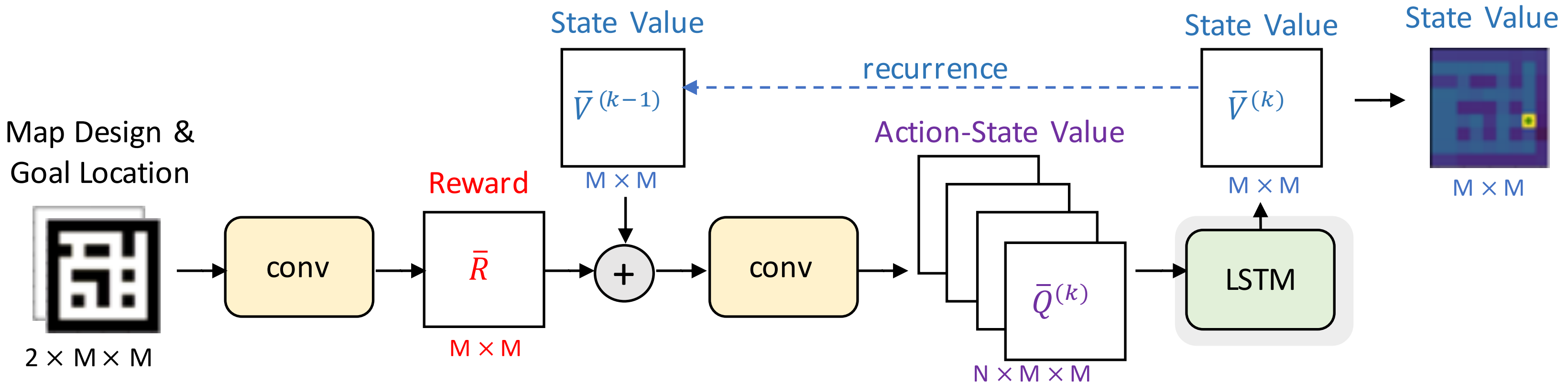
recurrence

nonlinearity

$$\bar{V}^{(k)} = \max_{\bar{a}} \Big(\overbrace{W_{\bar{a}}^{R} \bar{R}_{[3]} + W_{\bar{a}}^{V} \bar{V}_{[3]}^{(k-1)}}^{\text{convolution}} \Big)$$

kernel size 3

Gated Path Planning Networks (GPPN)



GPPN:
- Replace max-pooling activation with a well-established gated recurrent operator (e.g., LSTM).
- Allow kernel size F > 3.

recurrence

nonlinearity

convolution

$$\bar{V}^{(k)} = \mathrm{LSTM}\left(\sum_{\bar{a}} \left(W_{\bar{a}}^{R}\, \bar{R}_{[F]} + W_{\bar{a}}^{V}\, \bar{V}_{[F]}^{(k-1)}\right)\right)$$

kernel size

Gated Path Planning Networks (GPPN)

The gated LSTM update is well-known to alleviate many of the optimization problems with standard recurrent networks.

VIN update: $$V^{(k)} = \max_a \left(W_a^R R_{[3]} + W_a^V V_{[3]}^{(k-1)} \right)$$

GPPN update: $$V^{(k)} = \mathrm{LSTM}\left(\sum_a \left(W_a^R R_{[F]} + W_a^V V_{[F]}^{(k-1)} \right) \right)$$

Experimental Setup

Maze environments

Test VIN & GPPN on a variety of settings such as:

- Training dataset size
- Maze size
- Maze Transition Models

Maze environments

Test VIN & GPPN on a variety of settings such as:

- Training dataset size
- Maze size
- Maze Transition Models
  - **NEWS**

Maze environments

Test VIN & GPPN on a variety of settings such as:

- Training dataset size
- Maze size
- Maze Transition Models
  - NEWS
  - **Moore**

Maze environments

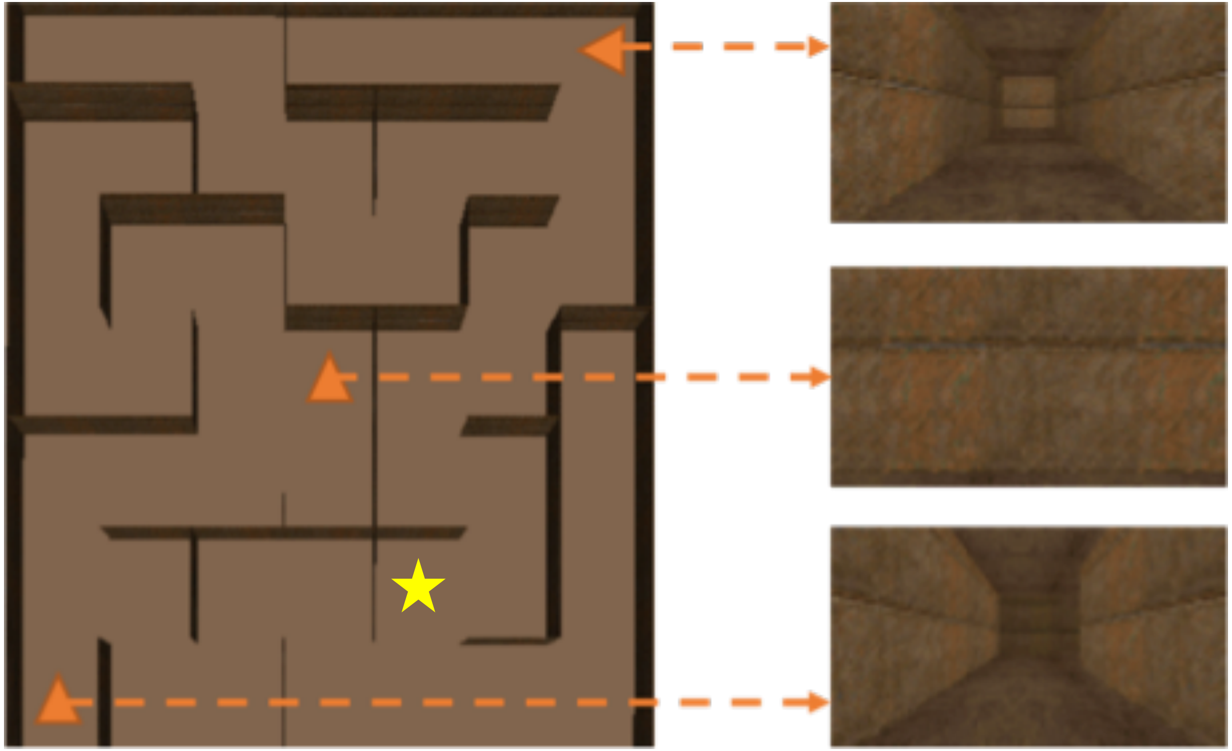
Test VIN & GPPN on a variety of settings such as:

- Training dataset size
- Maze size
- Maze Transition Models
  - NEWS
  - Moore
  - **Differential Drive**

Maze environments

3D ViZDoom Environment

First-person RGB images



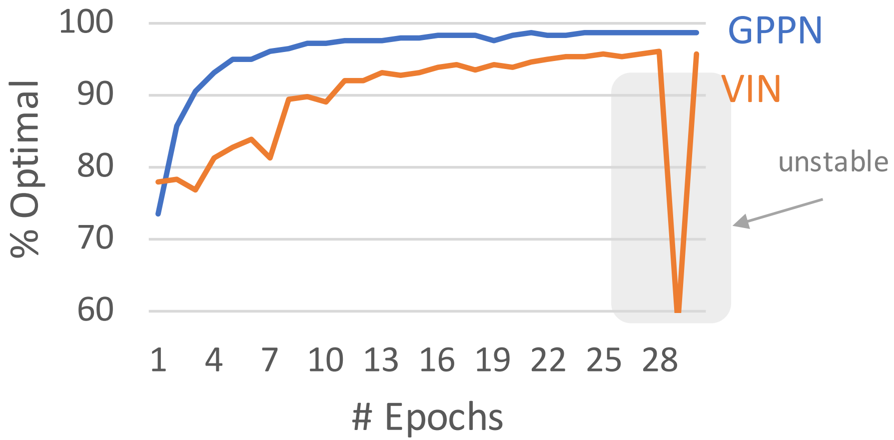
Experimental Results

Our GPPN outperforms VIN in a variety of metrics:

- **Learning speed**

GPPN learns faster.

% Optimal: Percentage of states whose predicted paths have optimal length.

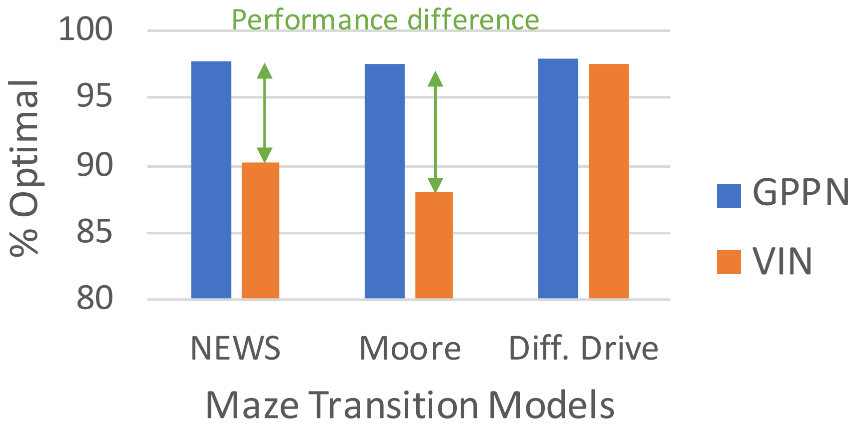


Test performance on 15 × 15 mazes with NEWS mechanism, dataset size 25k, and best (K, F) settings for each model.

Our GPPN outperforms VIN in a variety of metrics:

- Learning speed
- **Performance**

GPPN performs better.

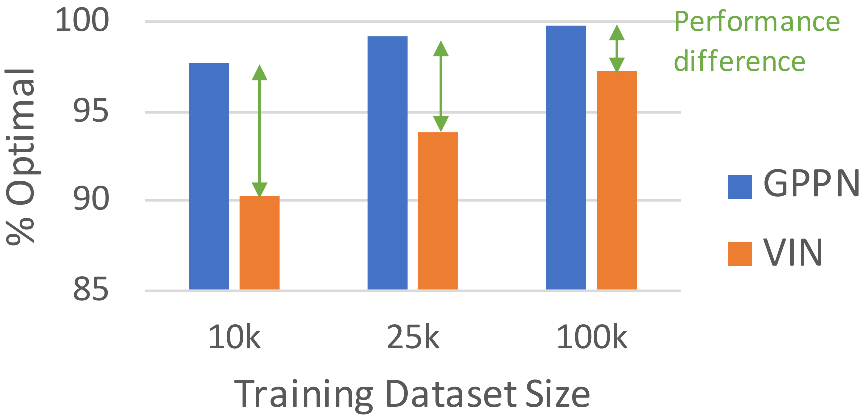


Test performance on 15 × 15 mazes with dataset size 10k and best (K, F) settings for each model.

Our GPPN outperforms VIN in a variety of metrics:

- Learning speed
- Performance
- **Generalization**

GPPN generalizes better with less data.

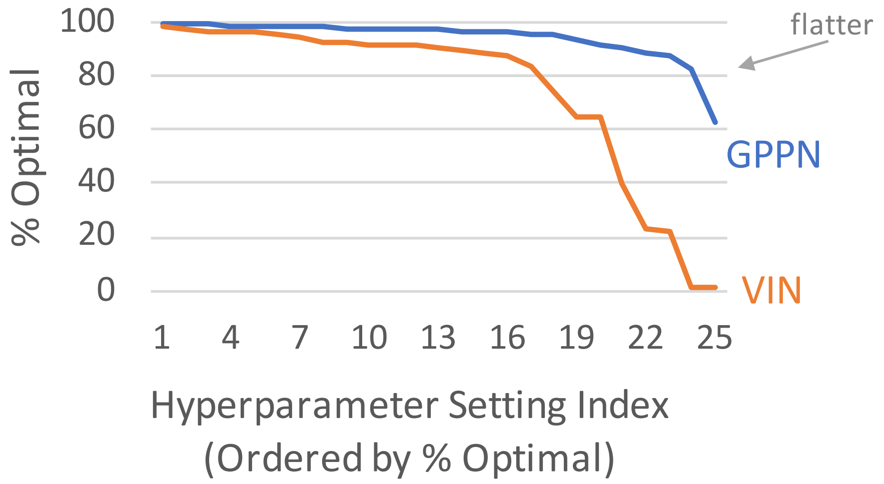


Test performance on 15 × 15 mazes with NEWS mechanism and best (K, F) settings for each model.

Our GPPN outperforms VIN in a variety of metrics:

- Learning speed
- Performance
- Generalization
- **Hyperparameter sensitivity**

GPPN is more stable to hyperparameter changes.

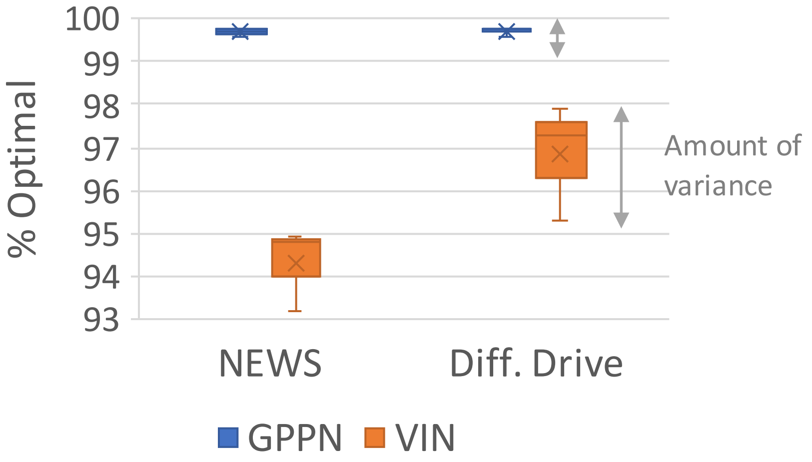


Test performance on 15 × 15 mazes with Differential Drive mechanism, dataset size 100k, and best (K, F) settings for each model.

Our GPPN outperforms VIN in a variety of metrics:

- Learning speed
- Performance
- Generalization
- Hyperparameter sensitivity
- **Random seed sensitivity**

GPPN exhibits less variance.



Test performance on 15 × 15 mazes with dataset size 100k and best (K, F) settings for each model.

Conclusion

- GPPN is a more general architecture that relaxes the architectural inductive bias of VIN.
  - Performs better & alleviates many optimization issues of VIN.
  - Our results suggest that path planning architectures need not strictly resemble path-finding algorithms like value iteration.

VIN: $$V^{(k)} = \max_a \left(W_a^R R_{[3]} + W_a^V V_{[3]}^{(k-1)} \right)$$

GPPN: $$V^{(k)} = \mathrm{LSTM}\left(\sum_a \left(W_a^R R_{[F]} + W_a^V V_{[F]}^{(k-1)} \right) \right)$$

Conclusion

- GPPN is a more general architecture that relaxes the architectural inductive bias of VIN.
  - Performs better & alleviates many optimization issues.
  - Our results suggest that path planning architectures need not strictly resemble path-finding algorithms like value iteration.
- By looking at VIN as a recurrent-convolutional network, we can explore other RNN architectural improvements:
  - Gated recurrent operators (Our work)
  - Multiplicative Integration (Wu et al., 2016)
  - Orthogonality constraints (Vorontsov et al., 2017)

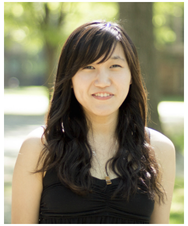
Future directions

Check out our poster!

Today 18:15 - 21:00 @ Hall B (#134)

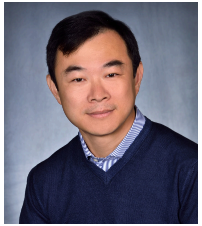
Code available on GitHub:

https://github.com/lileee/gated-path-planning-networks


Lisa Lee


Emilio Parisotto


Devendra Chaplot


Eric Xing


Russ Salakhutdinov