ML Reproducibility Challenge 2021 [1]
[Re] Differentiable Spatial Planning using Transformers

Anonymous Author(s)
Affiliation
Address
`email`

Reproducibility Summary

Scope of Reproducibility

This report covers our reproduction effort of the paper 'Differentiable Spatial Planning using Transformers' by Chaplot et al. In this paper, the problem of spatial path planning in a differentiable way is considered. They show that their proposed method of using Spatial Planning Transformers outperforms prior data-driven models and leverages differentiable structure to simultaneously learn to map in the absence of ground truth map. We verify these claims by reproducing their experiments and testing their method on new data. We also investigate the stability of planning accuracy with maps approaching mazes in obstacle complexity. Efforts to investigate and verify the learnings of the mapper module were met with failure stemming from paucity of compute resources.

Methodology

The authors' source code and datasets are not open-source yet. We reproduce the original experiments using source code written from scratch. We generate all synthetic datasets ourselves following similar parameters as described in the paper. Training the mapper module required loading our synthetic dataset over 1.6 TB in size on GPU, which could not be completed.

Results

We reproduced the accuracy for the SPT module to within 14.7% of reported value, which supports the paper's conclusion that it outperforms the baselines but fails to provide weight to stated figures. The Mapper module's accuracy could not be tested.

What was easy

The easiest part of the reproduction effort was getting the Spatial Planning Transformer model up and ready from scratch. The authors' instructions regarding the layer parameters and encoder-decoder structure was abundantly clear. Furthermore, even though initialisation information was missing, the model was robust enough to learn under various settings.

What was difficult

We lost significant time in generating all synthetic datasets, especially the dataset for the mapper module that required us to set up the Habitat Simulator and API. The ImageExtractor API was broken and workarounds had to be implemented. The final dataset approached 1.6 TB in size and we were unable to arrange enough compute resources and expertise to handle the GPU training. Furthermore, the description of the action prediction accuracy metric used is vague and could be one of the possible reasons behind non-reproducibility of the results.

Communication with original authors

Authors of the paper could not be reached even after multiple attempts.

[1] Equal Contribution.

1 Introduction

In the paper 'Differentiable Spatial Planning using Transformers' by Chaplot et al., the problem of spatial path planning in a differentiable way is considered. The authors show that their proposed method of using Spatial Planning Transformers outperforms prior data-driven models that propagate information locally via convolutional structure in an iterative manner, allows seamless generalization to out-of-distribution maps and goals and leverages differentiable structure to simultaneously learn to map in the absence of ground truth map.

2 Scope of reproducibility

We seek to investigate the following major claims made in the paper:

- Claim 1:
  Their proposed SPT planner module provides an absolute improvement of 7-19% over state-of-the-art CNN based planning baselines in average action prediction accuracy.
- Claim 3:
  Their proposed SPT planner module maintains stability in accuracy as complexity and number of obstacles is increased.
- Claim 3:
  Their proposed SPT module outperforms classical mapping and planning baselines under an end-to-end mapping and planning setting.

3 Methodology

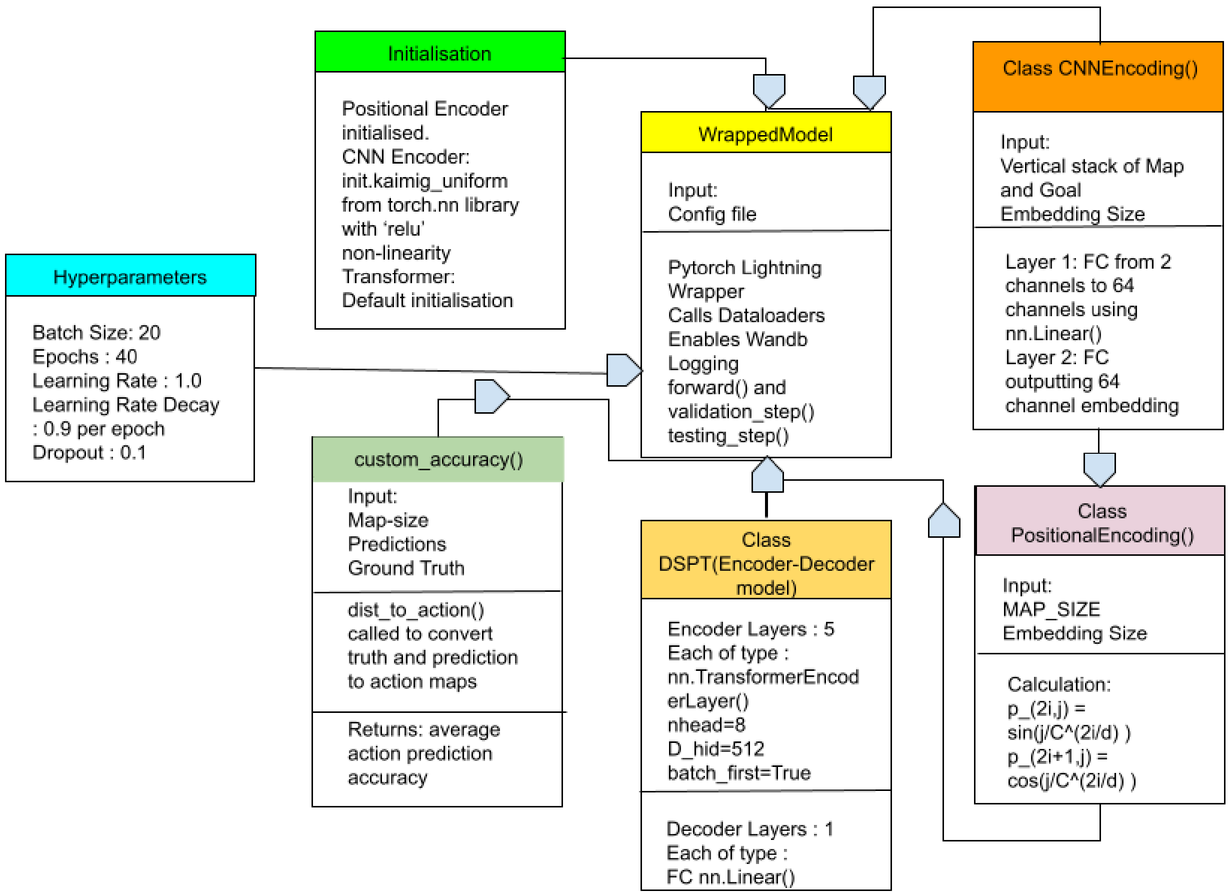
All code was written from scratch for both the SPT modules as well as the synthetic dataset generation in Python 3.6. Pytorch Lightning was used for the SPT modules. For dataset generation, similar parameters were used as mentioned in the paper to the maximum extent. The vagueness of parameters in terms of obstacle size provided us the opportunity to test out a range of obstacle sizes and the accuracy of the model on them. Training was done on NVIDIA Tesla T4 GPU on Google Colaboratory Pro.

3.1 Model descriptions

Our implementation of the model follows the description provided in the paper taking liberties where details are vague. Further investigations confirmed that liberties taken in specifying the CNN Encoder layers for feeding the Transformer does not have any significant effect on the final trained model.
The input map and the goal are stacked vertically and then fed into a CNN Encoder. The Encoder has 2 fully connected layers with a kernel size=1 and ReLU activation function. The first layer increases the number of channels from 2 to 64 while second layer maintains the number of channels and outputs a 64 channel encoded input.
Positional encoding as described in the original paper is added to the encoded input, which is then reshaped and fed into the Encoder part. Their are five encoder layers, each with nheads=8, hidden dimension=512 and dropout=0.1. This output is fed into a Decoder made of a fully connected layer. The Decoder gives ones output for each cell. The output is then reshaped to regain its original map shape.

3.2 Datasets

We create 3 datasets for the SPT planner module, each with a map size = {15,30,50} and upto 5 randomly generated obstacles. The position of the goal is randomly chosen from a free-space cell. A further dataset is generated at map size = 15 with upto 10 obstacles. Each of these datasets has 100,000 maps for training, 5,000 for validation and 5,000 for testing. We further used the Habitat Simulator and Habitat API to generate 36000 maps for training the end-to-end model. 72 scenes from the Gibson dataset from Stanford are loaded onto the simulator and 500 maps with a grid cell dimension of 0.5 meter and map size of 15, are rendered from each scene. Ground truths for all datasets were generated



Initialisation
Positional Encoder initialised.
CNN Encoder: init.kaimig_uniform from torch.nn library with 'relu' non-linearity
Transformer: Default initialisation
Hyperparameters
Batch Size: 20
Epochs : 40
Learning Rate : 1.0
Learning Rate Decay : 0.9 per epoch
Dropout : 0.1
WrappedModel
Input: Config file
Pytorch Lightning Wrapper
Calls Dataloaders
Enables Wandb Logging
forward() and validation_step() testing_step()
Class CNNEncoding()
Input: Vertical stack of Map and Goal
Embedding Size
Layer 1: FC from 2 channels to 64 channels using nn.Linear()
Layer 2: FC outputting 64 channel embedding
custom_accuracy()
Input: Map-size Predictions Ground Truth
dist_to_action() called to convert truth and prediction to action maps
Returns: average action prediction accuracy
Class DSPT(Encoder-Decoder model)
Encoder Layers : 5
Each of type : nn.TransformerEncoderLayer()
nhead=8
D_hid=512
batch_first=True
Decoder Layers : 1
Each of type : FC nn.Linear()
Class PositionalEncoding()
Input: MAP_SIZE Embedding Size
Calculation:
p_(2i,j) = sin(j/C^(2i/d))
p_(2i+1,j) = cos(j/C^(2i/d))

Figure 1: Code-flow diagram for our implementation.

using the classical Dijkstra's Algorithm. All datasets generated and used have been released for open-source and can be found on the project's github page.

3.3 Hyperparameters

An extensive hyperparameter grid search led us back to the same hyperparameters cited in the paper. The model is trained for 40 epochs with a learning rate decay of 0.9 per epoch, a starting learning rate of 1.0 and batch size of 20. The model is separately trained for each of the map distributions using mean squared error loss.

4 Results

We reproduced the accuracy for the SPT planner module to within 14.7% of reported value, which supports the paper's conclusion that it outperforms the baselines but fails to provide weight to stated figures. The Mapper module's accuracy could not be tested.

5 Discussion

5.1 What was easy

The easiest part of the reproduction effort was getting the Spatial Planning Transformer model up and ready from scratch. The authors' instructions regarding the layer parameters and encoder-decoder structure was abundantly clear.

	Navigation			Manipulation		Overall
Method	M=15	M=30	M=50	M=18	M=36	
VIN (Paper)	86.19	83.62	80.84	75.06	74.27	80.00
GPPN (Paper)	97.10	96.17	91.97	89.06	87.23	92.31
SPT (Paper)	99.07	99.56	99.42	99.24	99.78	99.41
SPT (Ours)	**84.40**	**83.33**	*	**86.49**	*	**84.74**

Table 1: Results.

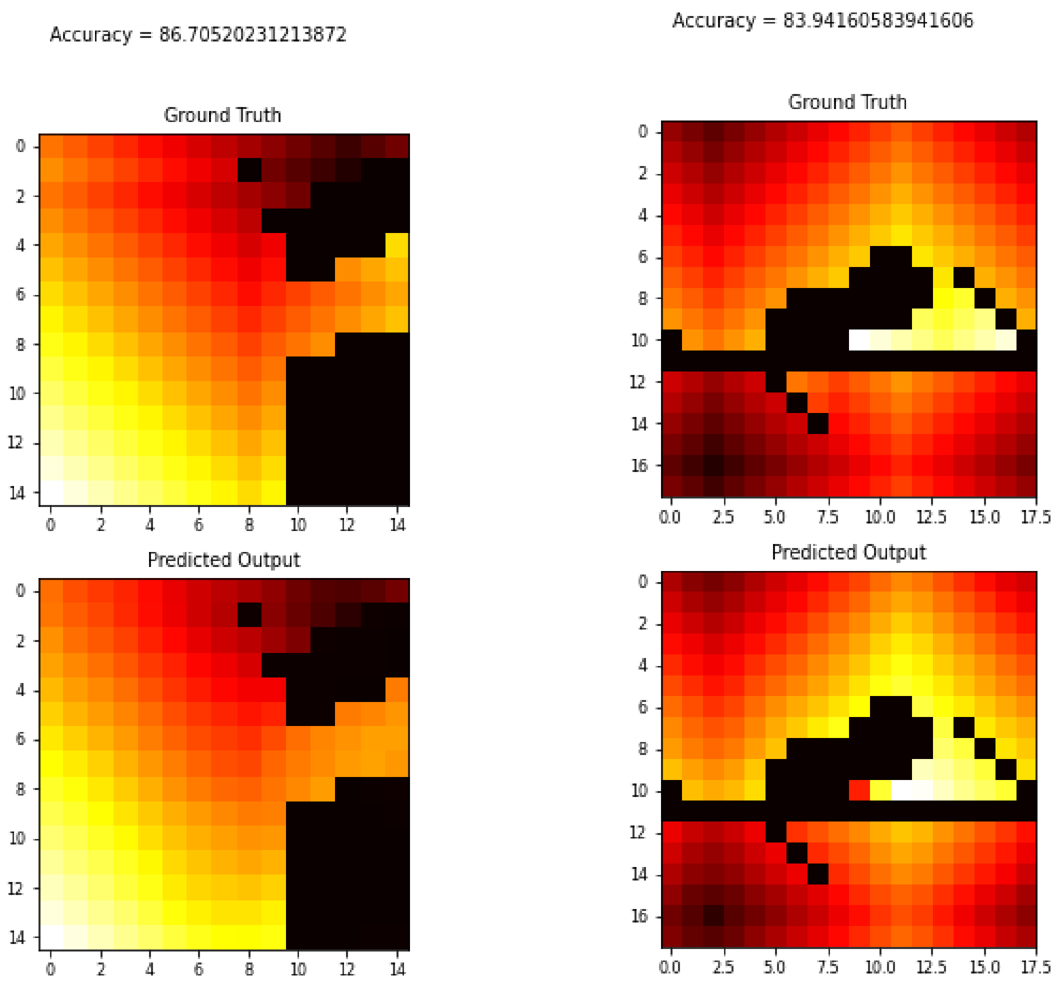


Figure 2: Sample output for Navigation Task (left) and Manipulation Task (right) visualised.

Furthermore, even though initialisation information was missing, the model was robust enough to learn under various settings.

5.2 What was difficult

We lost significant time in generating all synthetic datasets, especially the dataset for the mapper module that required us to set up the Habitat Simulator and API. The ImageExtractor API was broken and workarounds had to be implemented. The final dataset approached 1.6 TB in size and we were unable to arrange enough compute resources and expertise to handle the GPU training. The SPT Planner Module also could not be trained on M=50 dataset following the same issue.

5.3 Reproducibility of results of SPT Planner Module

Our results lag those mentioned in the paper by a margin of over 10% which makes us believe that the paper is not reproducible in its exact form. We suspect this to be the result of a variety of reasons which we discuss now.

The lack of openly available standard datasets in the domain presents a challenge. Different papers have to report results on datasets of their choice using a metric they design themselves. Our paper's authors also do this with their own synthetic datasets and a novel action prediction accuracy metric. Furthermore, these datasets are not open-sourced and generation parameters in the paper are vague in terms of obstacle complexity and size.

Our experiments with maps of increasing obstacle complexity shows a further steep drop in accuracy by 10-15%. This points to a plausible explanation for non-reproducibility. The non-uniformity of dataset-generation guidelines could have resulted in obstacles of greater size in our synthetic dataset.

5.4 Stability of the SPT Planner Module

Our results show comprehensively that the SPT Planner Module suffers from stability in accuracy as obstacle complexity is increased. This provides room for further research if we wish to apply SPTs to bigger maps.

5.5 Communication with original authors

Authors of the paper could not be reached even after multiple attempts.

6 Conclusion

We have tried to reproduce the paper to the best of our abilities, following the textual descriptions for source code and dataset generation to the maximum extent. The results that couldn't be reproduced are so prohibitively expensive that only the authors themselves can afford it, which is why they should release the code and datasets.

References

Chaplot, D.S., Pathak, D. and Malik, J., 2021, July. Differentiable spatial planning using transformers. In International Conference on Machine Learning (pp. 1484-1495). PMLR.

Tamar, A., Wu, Y., Thomas, G., Levine, S. and Abbeel, P., 2016. Value iteration networks. arXiv preprint arXiv:1602.02867.

Lee, L., Parisotto, E., Chaplot, D.S., Xing, E. and Salakhutdinov, R., 2018, July. Gated path planning networks. In International Conference on Machine Learning (pp. 2947-2955). PMLR.