Where the Heck Am I?
Few Shot Location Estimation on Synthetic Environments

Nicholas Samoray
University of Massachusetts at Amherst
nsamoray@umass.edu

Abstract

Traditional interior location estimation often requires sensor beacons or other costly hardware solutions tailored to each individual building. Deep learning approaches promise the ability to estimate position with just standard imagery, but requires hundreds of images of an environment in order to estimate an agents position accurately. The advent of meta-learning has opened an alternative approach that requires few images and is generalizable to a multitude of environments. In this paper, we demonstrate the applicability of using meta-learning to estimate an agents position within a simulated environment given only a few samples of labelled imagery. We show that meta-learning accomplishes this task with far greater sample efficiency than standard deep learning methods.

1. Introduction

The ability to learn from only a few examples has been a hallmark of human intelligence. Show an individual a handful of images of an elephant, and they will be able to recognize one in the flesh. Machine learning and artificial intelligence systems however require magnitudes more data, with standard image classification tasks requiring hundreds of samples of each object if not more. This holds especially true for navigation and estimating ones own position, even in a completely new environment. Introduce a friend to your new house, walk them through the rooms just once, point out notable features, and that friend will be able to navigate your house using those landmarks and their own situational awareness. While the problem of exterior location estimation has been essentially solved through technologies such as GPS, interior location estimation has remained a challenge, especially for environments that are novel or without navigational aids.

Through the use of meta-learning, we aim to introduce a new possible avenue for approaching this problem. Meta-learning is the process of learning how to best approach new data and process it in order to extract the maximal value and information from it. Put simply, it is learning how to learn. With enough samples, the goal is to be able to produce models that can process new information in a way as to require as little of it as possible, for example, only requiring a dozen images of an elephant in order to classify it properly rather than hundreds of images. This is referred to as a few shot problem, wherein each training example is a "shot". There also exist variants such as one shot wherein only one example is provided for training.

In this paper, we aim to test the applicability of meta-learning algorithms to the problem of few shot location estimation. We also introduce a synthetically generated dataset for regression that we believe is more challenging and practically applicable than the sine-wave regression used in most meta-learning papers.

2. Background/Related Work

There have been broad location estimation approaches with deep learning, such has Weyand *et al*.'s PlaNet [7], but their estimation encompasses the entire planet and is very broad in its estimation, aiming for accuracy that is measured in dozens of miles. Work on interior location estimation with deep-learning networks does not tend to make a lot of sense as traditional methods require hundreds of labelled images just for a single space.

In regards to meta-learning, there have existed several different approaches, including using Santoro *et al*.'s RNN approach [5] or Koch's Siamese model approach [2]. More recently, Finn *et al*. introduced Model-Agnostic Meta-Learning [1], or MAML, a general algorithm for approaching meta-learning. Following this was an adaptation introduced by Nichol *et al*. entitled Reptile [4], a simpler version that claims to have similar effectiveness to MAML. Finally there was also Meta-SGD by Li *et al*. [3], which is more analogous to traditional stochastic gradient descent but applied in a meta-learning context.

However, these tend to tackle a small subset of problems, almost entirely centered around image classification

on datasets such as Omniglot and ImageNet, and they have done well. Modern methods are able to obtain over 95% accuracy or more Omniglot, with as few as a single sample for each class. For regression however, the standard metric has been matching an arbitrary sine wave with as few data samples as possible. While modern methods have also done well on this task, we believe it is not a fair representation of the difficulty of some regression tasks as it can be solved with a fairly simple network and does not require more advanced neural networks such as convolutional networks.

For our synthetic data generation, papers such as Tremblay *et al*. [6] demonstrate that synthetic data generated via 3D modelling software can be used as a substitute for real image data when training deep learning models. They demonstrate this by creating synthetic outdoor environments and placing 3D models of cars in order to train their model, and their quantitative results indicate that this approach is successful at creating deep learning models that can transfer to real world examples with no real world data.

3. Approach

Our approach can be broken down into two chief components, the synthetic data generation application and the meta-learning model.

3.1. Dataset Generation

Due to the lack of a public dataset that would be sufficient for our goals, we generate the data ourselves utilizing the 3D Unity Engine. Our application was built to generate custom interior designs using a variety of 3D models of indoor "props" such as couches, shelves, and other furniture. These 3D models are randomly placed and oriented within the environment, with colors and textures varied randomly. We currently utilize over 65 different unique 3D models with 12 different textures. These textures can be set to any number of the millions of colors possible, giving us a practically unlimited range of possible arrangements and colors for our simulated interior. A sample interior can be seen in Figure 1.

The application is also able to randomly generate and place walls throughout the entire room, up to any number and density in order to occlude certain objects or portions of the interior. These walls can be given random colors as well, or remain the same color as the primary walls and floor. Various placements of walls and objects with random colors can be seen in Table 1.

For the virtual camera, we are able to set the field of view to any value we desire, though for this paper we stick to a field of view angle of 90. We have also built the option to generate a 360 field of view image of the cameras current location to simulate a 360 camera. We also implemented settings to ensure that the camera is not simply facing a blank wall in order to ensure it has a view of the room that can give it pertinent information. The resolution can be set to whatever is desired, though for the purposes of this paper we stuck to the resolution of 56x56 to accurately compare results.

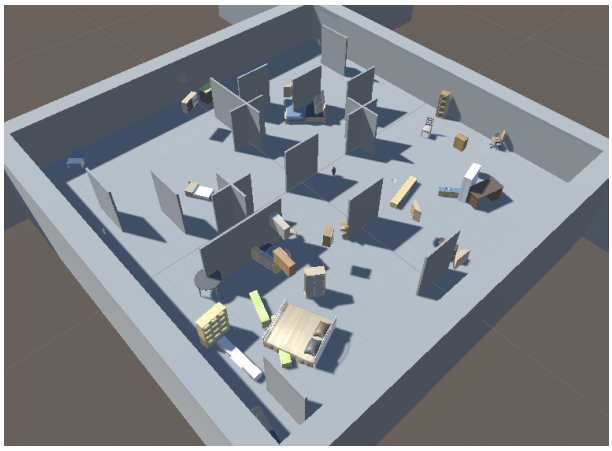


Figure 1: 3D view of a generated environment

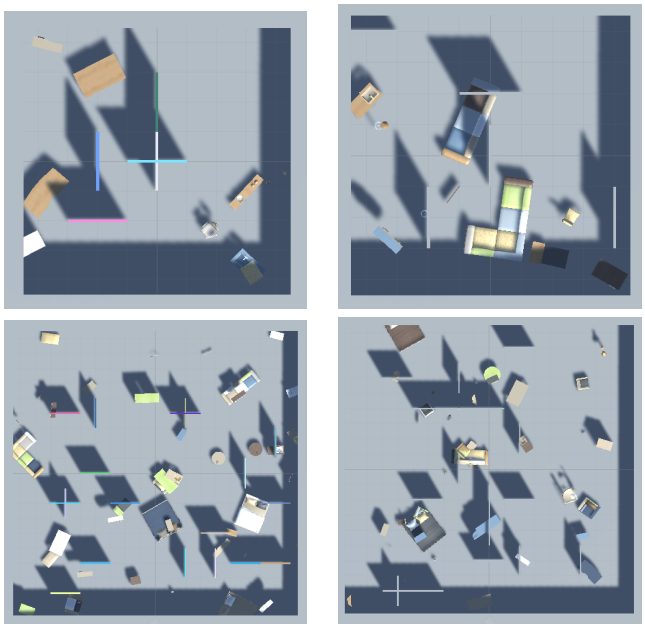


Table 1: Overhead views of several generated rooms of varying size and colors

The application automatically generates a room, fills it with props and walls, and generates as many images as desired for the given environment. For each image, it moves the virtual camera to a random position within the room, and orients it in a random direction. It then empties the room, and repeats the process for as many separate simulated environments are desired.

The final output for a single environment is a dataset con-

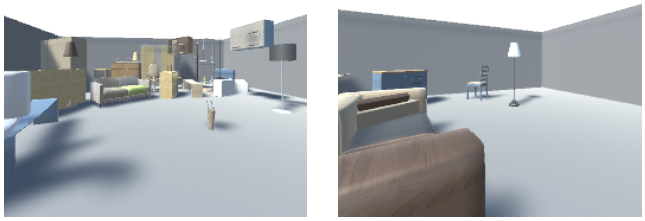


Table 2: Samples of generated perspective images that will be fed to the meta-learning model

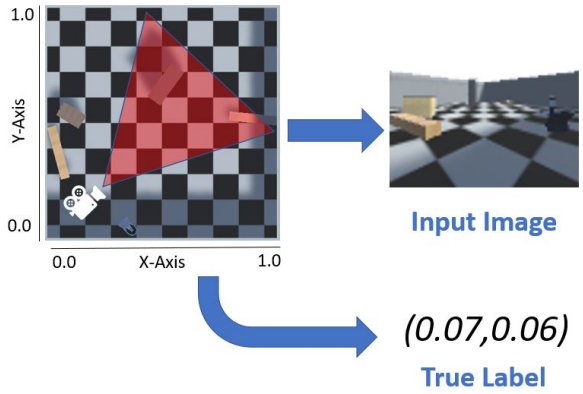


Figure 2: The agents position, represented by the camera icon, is place within the environment (Shown in an overhead view). The view from the agents position, represented by the shaded red area, is rendered into an input image. The agent's (X,Y) position within the environment is normalized to be between 0 and 1, and output as the true label for the rendered input image.

taining as many images as were specified by the generator settings, each one in a unique location. Along with the images is a comma delimited text file giving the normalized location within the environment, with an x value and a y value ranging from 0 to 1. An overview of the whole data generation process can be seen in Figure 2.

3.2. Meta-Learning

3.2.1 Meta Learning Base

For meta-learning, we explored a number of approaches, focusing primarily on Meta-SGD. Reptile and MAML were initially considered, however due to poor results in early experimentation were discarded as viable approaches given our time frame.

Meta-SGD, focuses on creating an optimizer that learns not only the ideal starting weights, but also tunes the learning rates of each parameter in order to best digest new information. This way, it can find not only the optimal weights that require minimal tuning for each new task, but the amount each parameter needs to change by in order to best tackle a new task. For example, an edge filter would be a filter that is very general and would require no changes regardless of task indicating a learning rate at zero or close to it. On the other hand, the fully connected layer at the end may need to change drastically for each new task, necessitating a higher learning weight for those parameters. With this approach, the data is only seen a single time by the model, and it is then adjusted. This is in contrast to traditional training that will undergo dozens or hundreds of epochs.

We decided against the approach of Siamese networks due to their design. Since they are geared towards classification primarily, it was determined that they would likely be ill suited for a regression task such as this. RNN based approaches were also decided against due to their greater sensitivity to hyperparameters and initialization. The approach we settled upon, Meta-SGD, claims to be both simpler and more effective than these methods as well.

3.2.2 Alterations for Location Approximation

Due to these algorithms being primarily tested and trained on classification tasks, a wide range of hyperparameter tuning was taken into account as well as modification of the models themselves. The first key change was changing the model from a classification to a regression model. This meant changing the output from softmax to linear, and changing the loss function. We tweaked the loss function, trying modifications such as euclidean distance between the two points, but we found standard mean squared difference provided comparable or better performance as well as being readily supported by deep learning libraries.

We found that otherwise the model structure could remain unaltered from the architecture used for conducting meta-learning on the ImageNet dataset. The few changes that were made included reducing the number of channels from 64 to 32, and trying various input sizes. The original input size was 84x84, but through experimentation we were able to go as small as 56x56 and achieve results while drastically speeding up the training time. The entire architecture used can be seen in Figure 3.

4. Experiments

To test our approach, we first established baselines against which to compare. Following that, we have tested our model against a variety of simulated environments, varying both size and number of visible objects within the room. We also have quantitative results for the effect that changing the input resolution has on our model and the

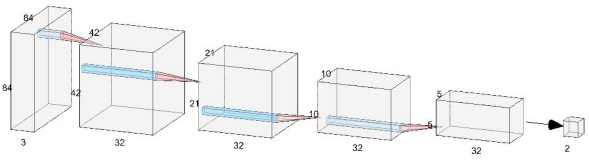


Figure 3: Overview of the architecture used. All kernels are 3x3.

number of samples processed. For each meta-learning experiment, evaluation is performed by measuring the error after a single training step is performed on new data in the train or test set.

4.1. Baselines

In order to test our dataset and general feasibility, we have run baselines utilizing our generated data in order to show that it is a solvable problem through standard deep learning regression. We have run these baselines with various conditions such as the number of available samples and the resolution of the imagery available. We found that we can predict a general location, with accuracy increasing the more samples that are available, with a resolution as low as 56x56. Typically, around 100-200 images will be enough to get accuracy within a simulated meter, this can be seen in Figures 4a and 4b. The value is the average per coordinate value error distance, that is, it is the absolute value in difference between the true coordinate values and our estimated values, averaged over both the x and y values.

It is important to note that while the normalized error may be similar between the two baseline figures, the absolute distance error is larger due to the larger area of the 20x20 meter environment. A relative error of 0.1 corresponds to an absolute distance error of 1 meter in a 10x10 meter room, but 2 meters in a 20x20 meter room.

For the results of the baseline, an error distance of 0.25 per point is considered the absolute minimum necessary for indicating that anything was learned at all. This is the same average error if $(0.5, 0.5)$ is guessed for every single output.

With this done, we know that general location can be estimated through deep learning regression on images of my simulated 3D environment and have a baseline with which to compare against.

4.2. Single Prop Room

We began our experiments with the simplest input we could generate, a single 3D prop in the center of the room. The camera was confined to a smaller space, about 5x5, in order to ensure that even with the low resolution that the prop was clearly visible. An example of what the environment would look like can be seen in Figure 5.

The purpose was to ensure that the meta-learning strategy was feasible given as simple of a room as possible. With a single prop, the task was akin to learning the single 3D model and how its features relate to the camera's position. Also important was seeing if the model would be able to learn how lighting and shadows relate to the camera's position. This was critical for situations such as in Figure 5 wherein the model is symmetrical on its vertical axis, the only difference being the lighting.

We chose a low resolution, only 56x56, in order to facilitate faster training times. This was decided upon due to the baseline results indicating it would be a sufficient resolution to see if the model is able to capture elements of the environment while minimizing the amount of necessary training time.

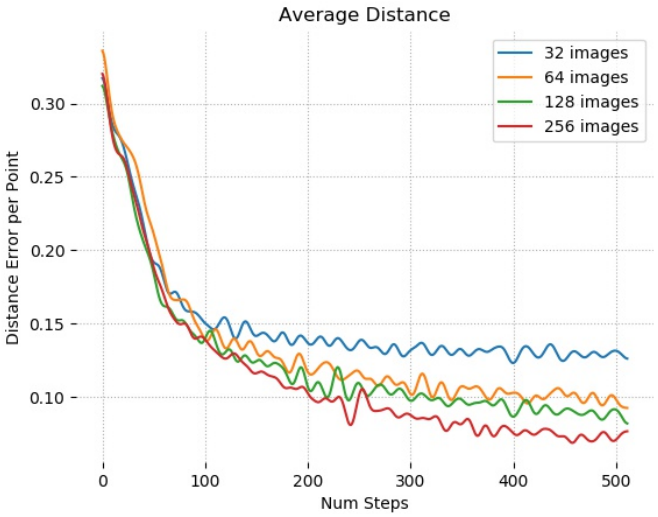
The results can be seen in Figure 6a. There is a clear correlation between the test and train sets, indicating that the Meta-SGD algorithm is indeed learning a generalizable approach for single prop environments. One thing that is noticeable however, is that the model's ability to generalize drops off past a certain point and it simply remains steady. We believe that this is due to the lower number of environment used for these particular experiments. Due to the lower number of individual prop models, around 65, we felt that 1024 environments would be a sufficient number for this task. However it is clear that the model begins to overfit on the training set without any additional performance on the test set.

4.3. Five Prop Room

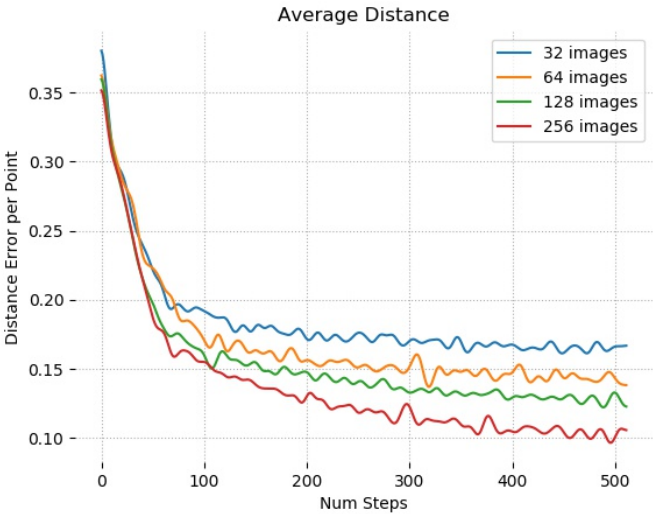
Given the results of the single prop room, we decided to generate a dataset consisting of a slightly larger room, but this time with up to five props scattered throughout the room as well as increasing the number of novel environments to 2048. This would prove to be a more rigorous test of the actual meta-learning approach due to the greater variety of possible layouts. A sample of the type the models inputs can be seen in Table 3. The results of this environment can be seen in Figure 6b.

With an environment with 5 props, the model seems to be able to generalize better than the single prop environment. We believe that due to the greater noise and diversity of environments, the Meta-SGD model is forced to learn a more general optimization strategy that lends itself to not overfitting. As a result, as seen in Figure 6b, the test results track very closely with the training results. This indicates that in a 5 prop environment, a modified version of Meta-SGD is able to generalize to similar environments effectively.

The increased number of samples also plays a large role. We found that fewer samples than 32 of an environment were insufficient for the model to learn past guessing randomly. With 64 samples, the results were further improved,



(a) Average distance error for a regression model trained on a 10x10 room with 5 props and 56x56 resolution. Each step is 1 batches, with each batch containing the entire training dataset. The lower the error, the closer the estimated location is to the true location for a given input image.



(b) Average distance error for a regression model trained on a 20x20 room with 40 props and 56x56 resolution. Each step is 1 batches, with each batch containing the entire training dataset. The lower the error, the closer the estimated location is to the true location for a given input image.

Figure 4

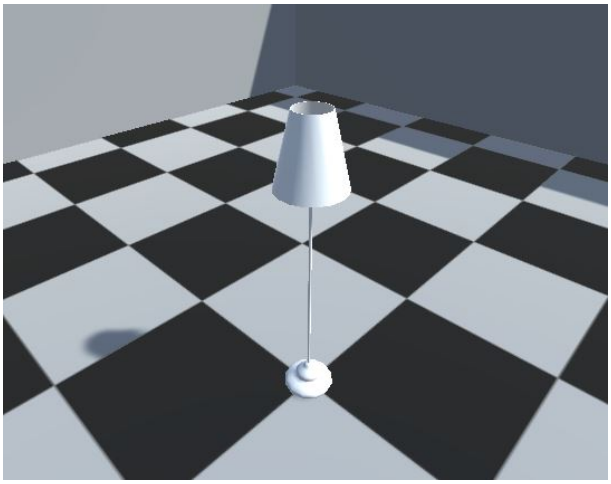


Figure 5: An example single prop environment

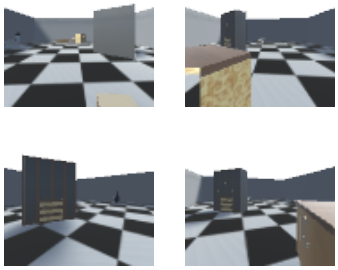


Table 3: Sample generated inputs for a 5 prop 10x10m room at 56x56 resolution.

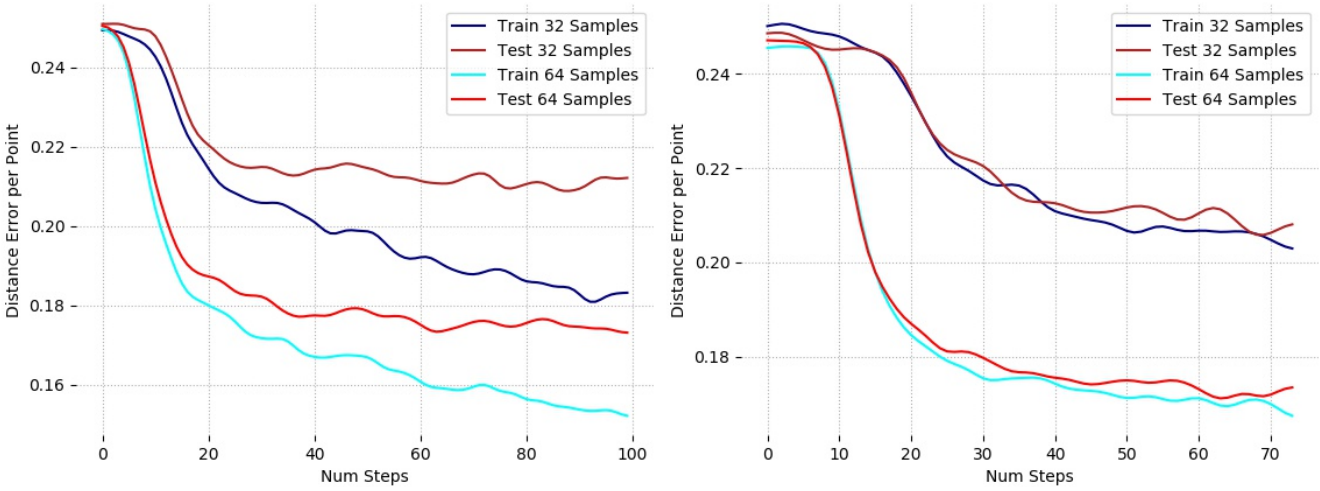
with an approximately 50% decrease in average error from a baseline of 0.25, from 0.21 decrease to 0.17.

When compared to the baseline, we are able to achieve results similar to around 50-75 steps of standard regression training, with our fully trained meta learning model taking only a single training step on new data. This is a marked improvement on our baseline in terms of sample efficiency, and the consistent results indicating that this model generalizes this sample efficiency to a variety of similar sized environments with different objects and placements.

4.4. Scaling to Larger Environments

We also tested our approach on a larger, busier room. We generated a dataset for a 20x20 meter simulated room with 40 props randomly placed, 4 times the square footage of the 10x10 meter rooms used in our other tests. The results can be seen in Figure 7a.

We found that with approximately twice the number of samples, we got very similar errors to our smaller environment even though the area is actually 4 times larger. However with the same number of samples, 64, the model was unable to converge effectively, indicating that it is necessary



(a) Average distance error for a single prop 5x5m room. Each step in this graph is 500 Meta-SGD updates. Lower is better.

(b) Average distance error for a five prop 10x10m room. Each step in this graph is 500 Meta-SGD updates. Lower is better.

Figure 6

to scale the number of samples with the size of the environment.

When compared to the baseline, our fully trained meta learning model is able to achieve results in one training step on new data that takes the standard regression baseline almost 100 steps to achieve. This is indicative of the model's ability to generalize and efficiently extract the necessary information from a limited set of samples in just a single step even in larger environments.

4.5. Impact of Resolution

To test the impact that the input resolution has on the results, we ran the same 5 prop environment dataset through the same architecture, but the images scaled down to 70x70 rather than 56x56. The results of which can be seen in Figure 7b.

The results are similar, but counter-intuitively the results for the smaller resolution are better than those of the larger resolution. Our belief is that the Meta-SGD model is able to find better parameters when there are fewer of them, and the few extra parameters of the larger model make it more difficult to converge. It may however also be down to a poor initialization. Further tests are needed in order to confirm the confidence intervals.

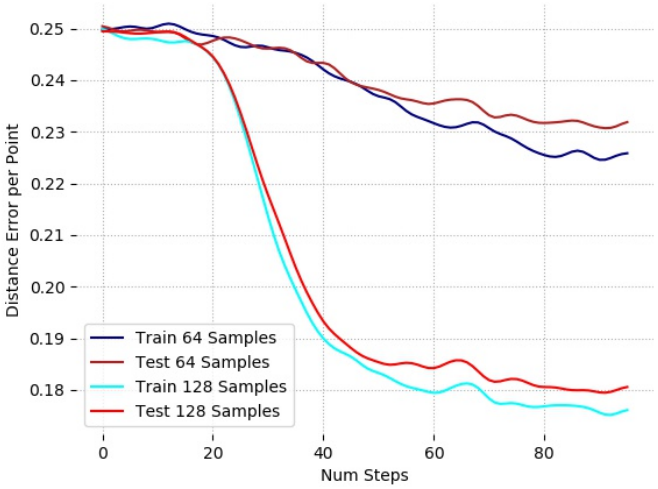
5. Conclusion

In this paper, we have shown that meta-learning is an approach that is viable for more complex regression tasks, specifically estimating an agent's position in an environment. While this paper was limited in scope, we believe that it hints at several promising directions. We believe that higher resolution and deeper architectures hold the key to more accurate results regardless of environment.

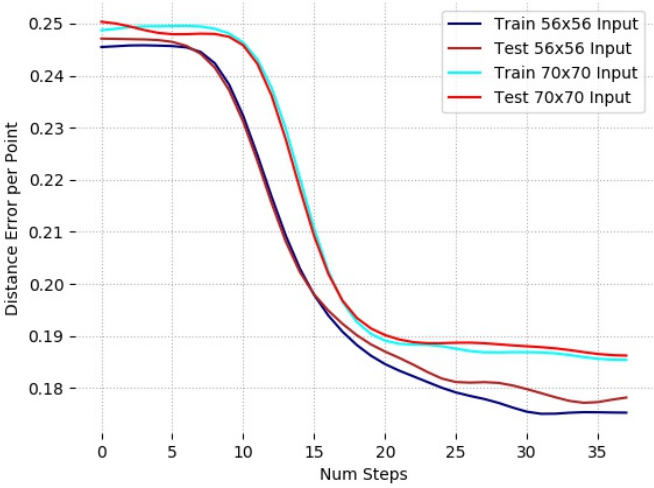
Other directions to explore include more complex synthetic environments, such as more realistic layouts, multiple rooms, multiple floors, greater diversity of objects and textures, and possibly even synthetic environments that are recreated from real ones. While we do not currently believe that a model with this low of a resolution would be able to be applied in a real life complex environment, we believe that proving out a deeper, larger, model on more complex synthetic environments is a fitting next step.

One more factor that could be significant is the number of environments available to the meta-learning model to train on. As seen in Figure 5, 1024 environments was not enough to prevent a divergence between the training and test environments. 2048 environments created a stronger correlation between the test and train sets, as seen in Figures 6b and 7a, but it may be worth if even more environments lead to better results, even as many as 10k+ environments.

Finally, approaches such as Reptile and MAML gave poor initial results in our experimentation. Further work should be done exploring if they simply require greater hyper-parameter tuning, different model architectures, or other modifications to enable them to work effectively. If they are unable to work regardless of any modifications, it should be worth exploring why Reptile and MAML fail where Meta-SGD succeeds.



(a) Average distance error for a 40 prop 20x20m room. Each step in this graph is 500 Meta-SGD updates. Lower is better.



(b) Results for a five prop room with 56x56 input vs. 70x70 input. Each step in this graph is 500 Meta-SGD updates. Lower is better.

Figure 7

References

[1] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[2] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

[3] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few-shot learning, 2017.

[4] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms, 2018.

[5] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. P. Lillicrap. One-shot learning with memory-augmented neural networks. *CoRR*, abs/1605.06065, 2016.

[6] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization, 2018.

[7] T. Weyand, I. Kostrikov, and J. Philbin. Planet - photo geolocation with convolutional neural networks. In *European Conference on Computer Vision (ECCV)*, 2016.