Surveillance Problem with Deep Learning

Emily Nguyen, Rui Fang

Oden Institute

2021

Table of Contents

Table of Contents

Surveillance Problem

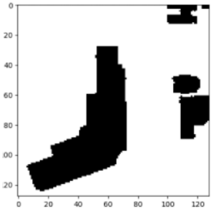
- With a known environment (comprised of free and occluded space), what is the minimum number of (and positions of the minima of) sensors to map out the environment?

Surveillance Problem

- With a known environment (comprised of free and occluded space), what is the minimum number of (and positions of the minima of) sensors to map out the environment?
- Example scenarios
  - Placing lights in a city block to make every part visible
  - The minimum set of guards required to watch over an art gallery

Surveillance Problem

- With a known environment (comprised of free and occluded space), what is the minimum number of (and positions of the minima of) sensors to map out the environment?
- Example scenarios
  - Placing lights in a city block to make every part visible
  - The minimum set of guards required to watch over an art gallery



Problem Statement

- Consider a domain $\Omega \subset \mathbb{R}^n$. Let the domain be partitioned as $\Omega = \Omega_{\text{free}} \cup \Omega_{\text{obs}}$
  - Ω_{free}: free space
  - Ω_{obs}: obstacles/occluded space with no holes

Problem Statement

- Consider a domain $\Omega \subset \mathbb{R}^n$. Let the domain be partitioned as $\Omega = \Omega_{\text{free}} \cup \Omega_{\text{obs}}$
  - Ω_{free}: free space
  - Ω_{obs}: obstacles/occluded space with no holes
- Find a sequence of vantage points x_i that maximize a gain function
  - $\mathcal{V}_{x_i}$: visibility set (points visible from x_i)
  - $\Omega_k = \bigcup_{i=0}^{k} \mathcal{V}_{x_i}$: cumulative visibility set (note: $\Omega_{k-1} \subset \Omega_k$)

Problem Statement

- Consider a domain $\Omega \subset \mathbb{R}^n$. Let the domain be partitioned as $\Omega = \Omega_{\text{free}} \cup \Omega_{\text{obs}}$
  - Ω_{free}: free space
  - Ω_{obs}: obstacles/occluded space with no holes
- Find a sequence of vantage points x_i that maximize a gain function
  - $\mathcal{V}_{x_i}$: visibility set (points visible from x_i)
  - $\Omega_k = \bigcup_{i=0}^{k} \mathcal{V}_{x_i}$: cumulative visibility set (note: $\Omega_{k-1} \subset \Omega_k$)
- The surveillance problem is: for a set of vantage points O

$$\min_{O \subseteq \Omega_{\text{free}}} |O| \text{ subject to } \Omega_{\text{free}} = \bigcup_{x \in O} \mathcal{V}_x$$

Problem Statement

- Consider a domain $\Omega \subset \mathbb{R}^n$. Let the domain be partitioned as $\Omega = \Omega_{\text{free}} \cup \Omega_{\text{obs}}$
  - Ω_{free}: free space
  - Ω_{obs}: obstacles/occluded space with no holes
- Find a sequence of vantage points x_i that maximize a gain function
  - $\mathcal{V}_{x_i}$: visibility set (points visible from x_i)
  - $\Omega_k = \bigcup_{i=0}^{k} \mathcal{V}_{x_i}$: cumulative visibility set (note: $\Omega_{k-1} \subset \Omega_k$)
- The surveillance problem is: for a set of vantage points O

$$\min_{O \subseteq \Omega_{\text{free}}} |O| \text{ subject to } \Omega_{\text{free}} = \bigcup_{x \in O} \mathcal{V}_x$$

- An implicit assumption: the sensor range is larger than the area of the environment

Greedy Algorithm

- Can use a greedy algorithm based on choosing subsequent x_{k+1} that maximize information gain. Let the gain function be

$$g(x;\Omega_k) := |\mathcal{V}_x \cup \Omega_k| - |\Omega_k|$$

i.e., the volume of region visible from x but not from previous points $x_0, x_1, \ldots, x_k$

Greedy Algorithm

- Can use a greedy algorithm based on choosing subsequent x_{k+1} that maximize information gain. Let the gain function be

$$g(x; \Omega_k) := |\mathcal{V}_x \cup \Omega_k| - |\Omega_k|$$

  i.e., the volume of region visible from x but not from previous points $x_0, x_1, \ldots, x_k$
- We choose

$$x_{k+1} = \arg\max_{x \in \Omega_{\text{free}}} g(x; \Omega_k)$$

Greedy Algorithm

- Can use a greedy algorithm based on choosing subsequent x_{k+1} that maximize information gain. Let the gain function be

$$g(x; \Omega_k) := |\mathcal{V}_x \cup \Omega_k| - |\Omega_k|$$

i.e., the volume of region visible from x but not from previous points $x_0, x_1, \ldots, x_k$

- We choose

$$x_{k+1} = \arg\max_{x \in \Omega_{\text{free}}} g(x; \Omega_k)$$

- However, this is very costly to compute

Approximating Greedy Algorithm

- Try to train a model to predict the gains

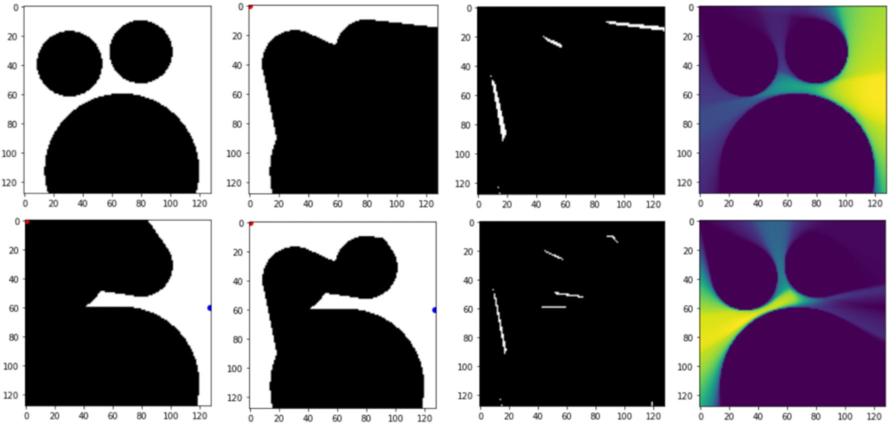
Approximating Greedy Algorithm

- Try to train a model to predict the gains
- Let the learned function be

$$g_\theta(x; \Omega_k, B_k)$$

  where B_k is the part of $\partial\Omega_k$ that might be in Ω_{free} (i.e., $B_k = \partial\Omega_k \setminus \Omega_{\text{obs}}$)

Cumulative Visibility, Frontiers, and Gain Function



Example Results

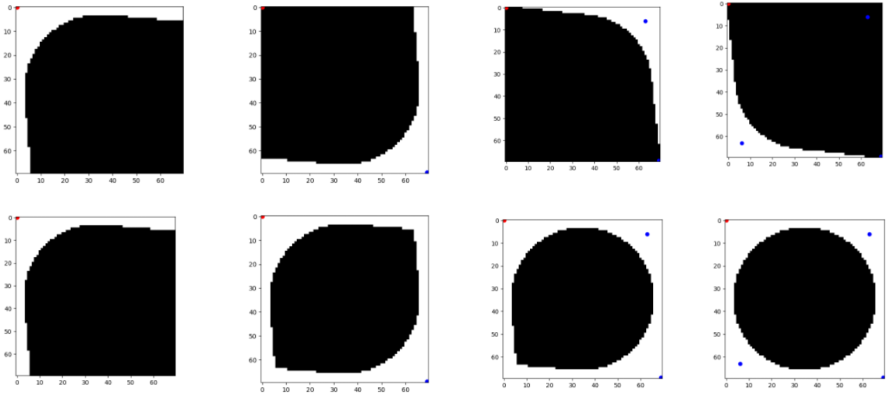
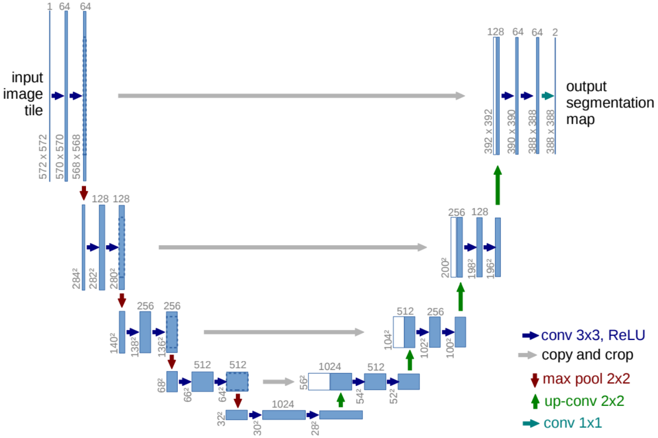


Table of Contents

U-Net

- Base the model on U-Net because we want the value predicted at each pixel to depend on a neighborhood



Source: https://arxiv.org/pdf/1505.04597.pdf

Dataset

- Used two types of environments
  - Circle: 64×64 pixel environments consisting of 2 to 6 circles
  - City: 128×128 pixel aerial city blocks cropped from INRIA Aerial Image Labeling Dataset

Dataset

- Used two types of environments
  - Circle: 64×64 pixel environments consisting of 2 to 6 circles
  - City: 128×128 pixel aerial city blocks cropped from INRIA Aerial Image Labeling Dataset
- Generated training data using the greedy algorithm

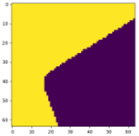
Dataset

- Used two types of environments
  - Circle: 64×64 pixel environments consisting of 2 to 6 circles
  - City: 128×128 pixel aerial city blocks cropped from INRIA Aerial Image Labeling Dataset
- Generated training data using the greedy algorithm
- Removed data with very small gain, then normalized gain such that the maximum gain is 1 for each image

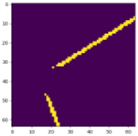
Dataset

- Used two types of environments
  - Circle: 64×64 pixel environments consisting of 2 to 6 circles
  - City: 128×128 pixel aerial city blocks cropped from INRIA Aerial Image Labeling Dataset
- Generated training data using the greedy algorithm
- Removed data with very small gain, then normalized gain such that the maximum gain is 1 for each image
- Got ∼50,000 data pairs for Circle, and ∼20,000 data pairs for City

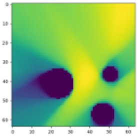
Inputs and Targets (Circle)



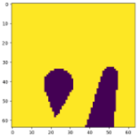
(a) Cumulative Visibility 1



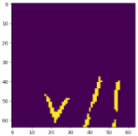
(b) Frontier 1



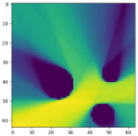
(c) Gain 1



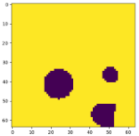
(d) Cumulative Visibility 2



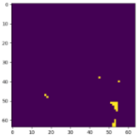
(e) Frontier 2



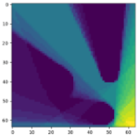
(f) Gain 2



(g) Cumulative Visibility 3

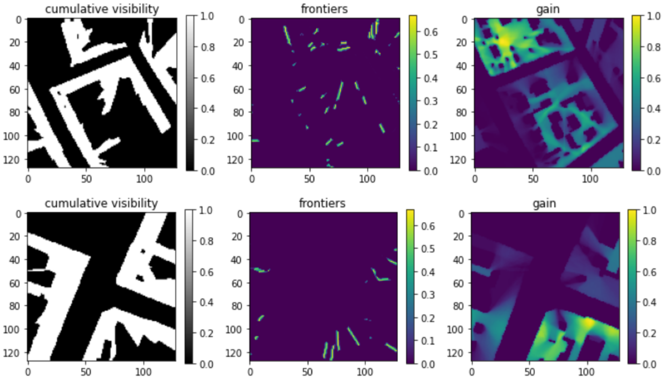


(h) Frontier 3



(i) Gain 3

Inputs and Targets (City)



Training

- Trained a model for each of Circle and City, used 90% of data for training, 10% for validation

Training

- Trained a model for each of Circle and City, used 90% of data for training, 10% for validation
- Loss function is binary cross entropy

$$L(p, q) = -\left[p \log(q) + (1 - p) \log(1 - q)\right]$$

Training

- Trained a model for each of Circle and City, used 90% of data for training, 10% for validation
- Loss function is binary cross entropy

$$L(p, q) = -\left[p \log(q) + (1 - p) \log(1 - q)\right]$$

- Optimization method is Adam
    - Initial learning rate: 10^{-3}

Training

- Trained a model for each of Circle and City, used 90% of data for training, 10% for validation
- Loss function is binary cross entropy

$$L(p, q) = -\left[p \log(q) + (1 - p) \log(1 - q)\right]$$

- Optimization method is Adam
  - Initial learning rate: 10^{-3}
- Used PyTorch and Google Colab

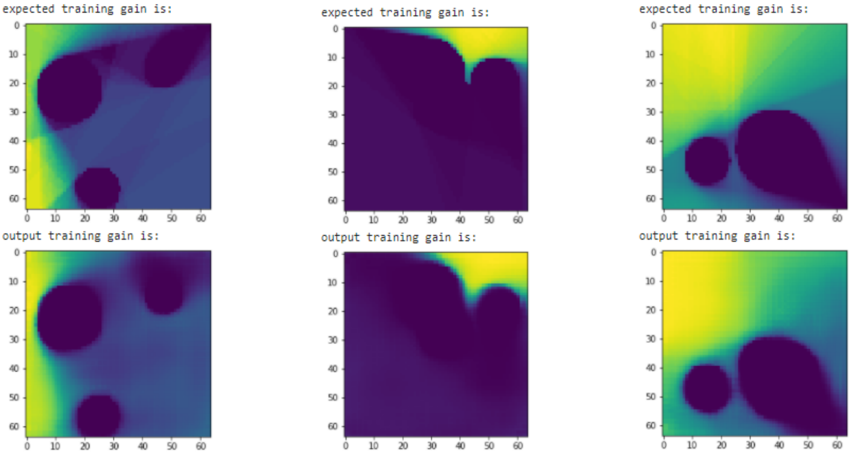
Table of Contents

Results: Circular Environments

- These training outputs are from 1770 epochs in, which is indicated to be the point where the losses start to stagnate

Results: Circular Environments

- These training outputs are from 1770 epochs in, which is indicated to be the point where the losses start to stagnate

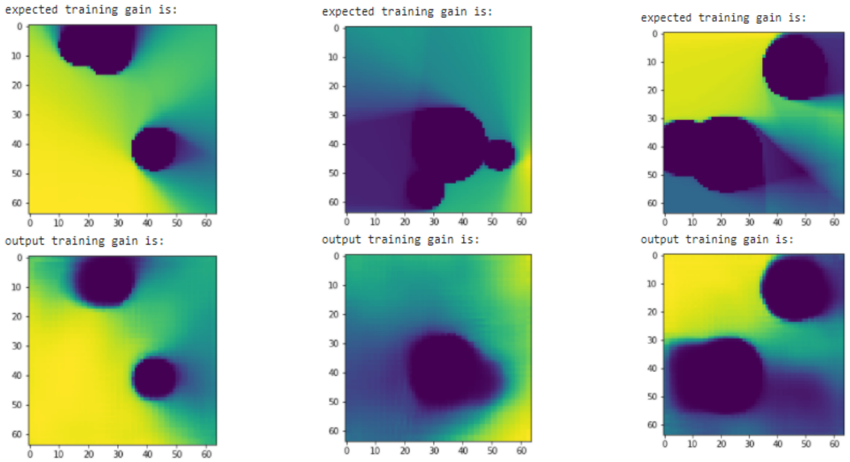


Results: Circular Environments

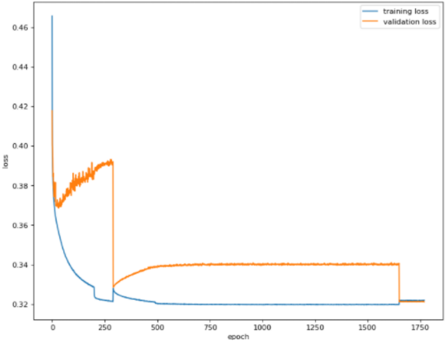
- However, there are some environments that the model does not predict as well

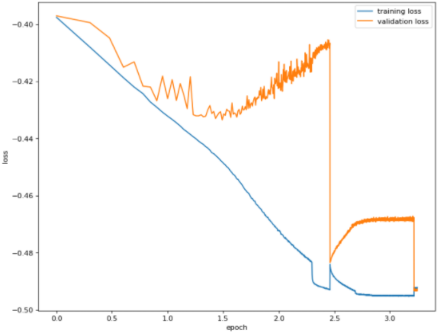
Results: Circular Environments

- However, there are some environments that the model does not predict as well



Results: Circular Environments Losses



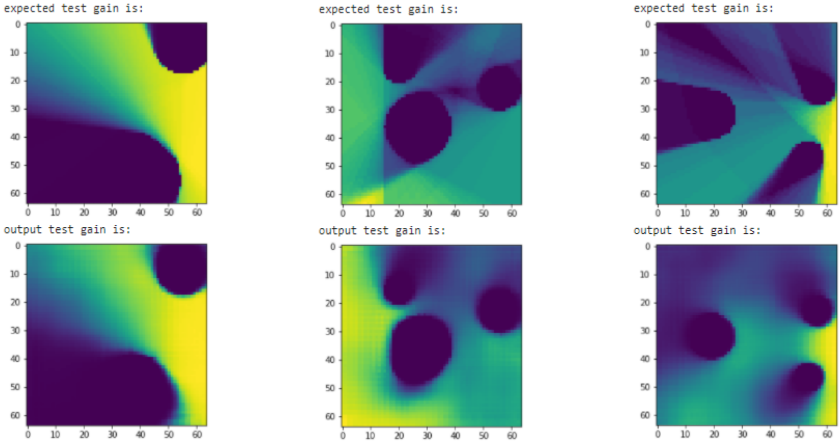


Results: Circular Environments Generalization

- The model seems to generalize well on the validation set

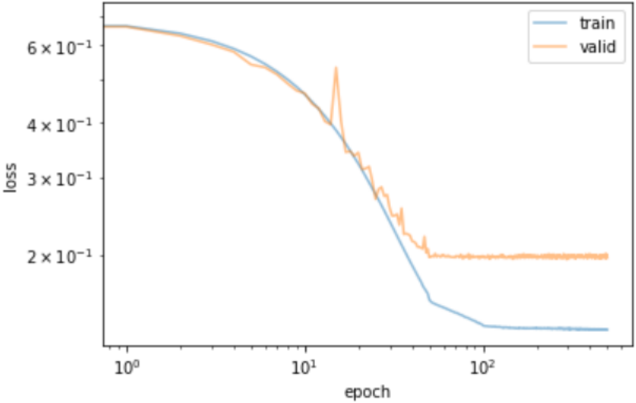
Results: Circular Environments Generalization

- The model seems to generalize well on the validation set



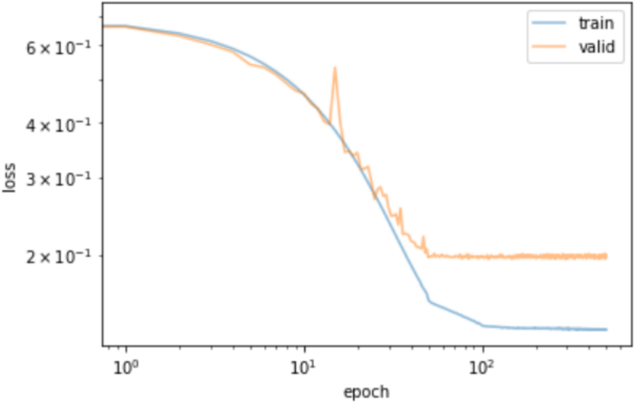
Results: City Environments

- Loss vs Epoch for 500 epochs



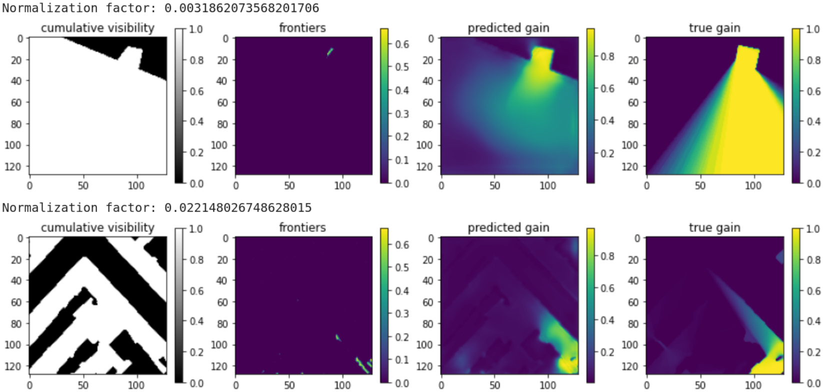
Results: City Environments

- Loss vs Epoch for 500 epochs



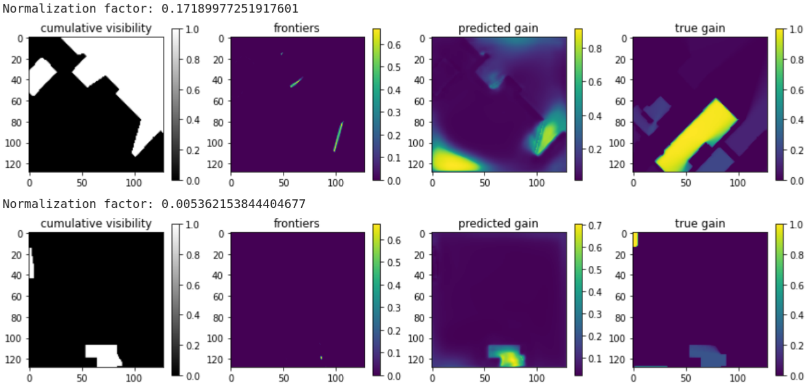
- Learning rate was decreased by a factor of 10 at epoch 50, 100, 150, resulting in slowed down decreases in losses
- The model doesn't generalize well on validation set

- Prediction on validation set
  - Good examples



Results: City Environments

- Prediction on validation set
  - Bad examples



Discussion

- For the circular environment, the model seems to combine circles if they seem too similar in shape to a bigger circle

Discussion

- For the circular environment, the model seems to combine circles if they seem too similar in shape to a bigger circle
- A disadvantage of using Google Colab is the runtime limit where the runtime disconnects after a certain number of hours. This makes restarting more difficult in terms of keeping track of the learning rate
  - The validation and training losses experience an unusual jump upon continuing
  - A fix for this will be to train locally or on TACC

Discussion

- For the circular environment, the model seems to combine circles if they seem too similar in shape to a bigger circle
- A disadvantage of using Google Colab is the runtime limit where the runtime disconnects after a certain number of hours. This makes restarting more difficult in terms of keeping track of the learning rate
  - The validation and training losses experience an unusual jump upon continuing
  - A fix for this will be to train locally or on TACC
- For the city environment, in many cases the map contains “isolated” free space, e.g., a courtyard surrounded by buildings. It is not going to be seen by any vantage point placed outside. Then it is extremely difficult for the network to predict the gain over such space.

Discussion

- For the circular environment, the model seems to combine circles if they seem too similar in shape to a bigger circle
- A disadvantage of using Google Colab is the runtime limit where the runtime disconnects after a certain number of hours. This makes restarting more difficult in terms of keeping track of the learning rate
  - The validation and training losses experience an unusual jump upon continuing
  - A fix for this will be to train locally or on TACC
- For the city environment, in many cases the map contains "isolated" free space, e.g., a courtyard surrounded by buildings. It is not going to be seen by any vantage point placed outside. Then it is extremely difficult for the network to predict the gain over such space.
  - Can improve this by giving the network more direct information of the map, e.g., using cumulative visibility and obstacles as input, rather than using cumulative visibility and frontier

Future Directions

- Tune hyperparameters of model and training procedure, in order to make the model more generalizable

Future Directions

- Tune hyperparameters of model and training procedure, in order to make the model more generalizable
- Generate sequences of vantage points based on approximated gain function, and compare against the points generated with exact gain function

Future Directions

- Tune hyperparameters of model and training procedure, in order to make the model more generalizable
- Generate sequences of vantage points based on approximated gain function, and compare against the points generated with exact gain function
- Frequency maps: for a certain map, using many sequences of vantage points generated with different initial conditions, see hot spots for where the greedy algorithm tends to put sensors/pick vantage points

Future Directions

- Tune hyperparameters of model and training procedure, in order to make the model more generalizable
- Generate sequences of vantage points based on approximated gain function, and compare against the points generated with exact gain function
- Frequency maps: for a certain map, using many sequences of vantage points generated with different initial conditions, see hot spots for where the greedy algorithm tends to put sensors/pick vantage points
- Build a model to learn the “true” (unnormalized) gain function

Future Directions

- Tune hyperparameters of model and training procedure, in order to make the model more generalizable
- Generate sequences of vantage points based on approximated gain function, and compare against the points generated with exact gain function
- Frequency maps: for a certain map, using many sequences of vantage points generated with different initial conditions, see hot spots for where the greedy algorithm tends to put sensors/pick vantage points
- Build a model to learn the “true” (unnormalized) gain function
  - Option 1: train the “true” gain function with MSE loss, with ReLU as the final layer
  - Option 2: create an additional module, MLP for example, connecting the bottom layer of the current Unet to a single output for the normalization factor

References

- Olaf Ronneberger and Philipp Fischer and Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation
- Louis Ly thesis