Active Learning for Pixel-wise Prediction

June 26th, 2020

Email: abdill@andrew.cmu.edu

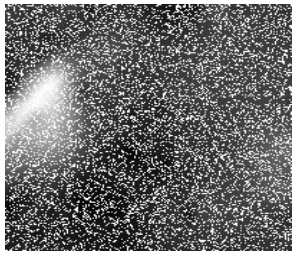
Overview of Presentation

1. Depth Estimation Task
2. Overview of Problem Set-up
3. DDPG Approach
4. Effect of Image Scale
5. Mask Plausibility
6. Takeaways and Plans Moving Forward

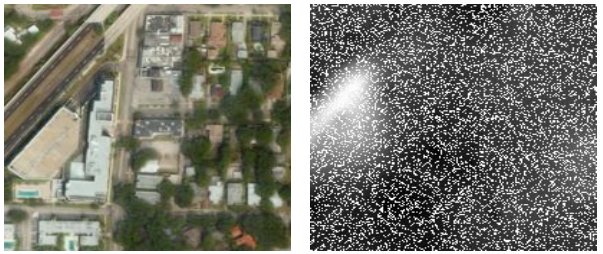
Depth Estimation Task with Active Learning

Input Image

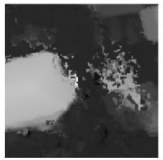
Masking



Concat



Prediction

Error = ( - )^2

Prediction

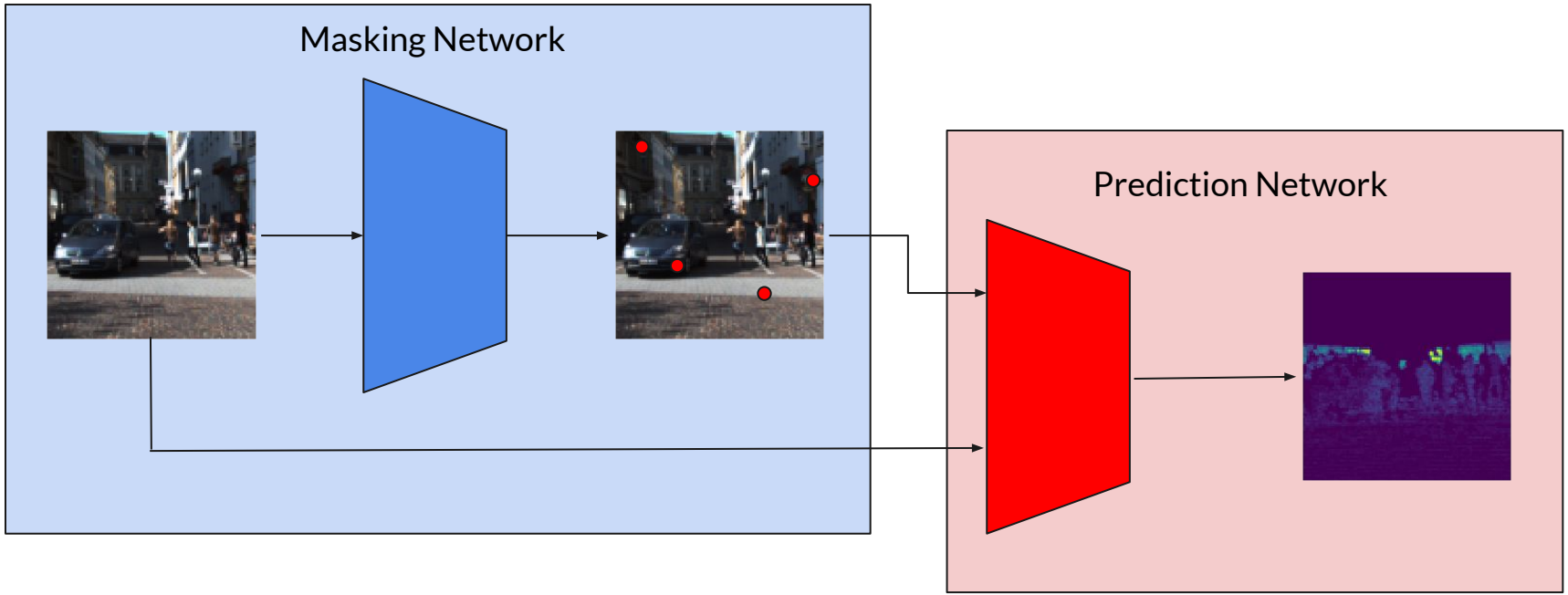
Ground Truth

KITTI Dataset Details and Preprocessing [1]

- 38,323 training images and 3,347 testing images.
- Initial images are 374 x 1238 pixels and have 3 channels intended for training self-driving cars.
- Square subregions are cropped out.
- These square regions are downsampled to be 100 by 100 pixels in order to facilitate faster training.

General Problem Set-Up

Two Networks Cooperate to Predict the Output



Measures of Success

- MSE for per pixel depth prediction: $\frac{1}{nm}\sum_{i=0}^{n}\sum_{j=0}^{m}(D_{ij}-\hat{D}_{ij})^2$
- Count of nonzero elements in the mask: $\sum_{i=0}^{n}\sum_{j=0}^{m}\mathbb{I}\{M_{ij}\neq 0\}$

Three Essential Questions for this Approach

- How do we define the masking network?
- How do we train this model jointly?
- Can we use this leverage this approach to gradually use no sparse depth information?

DDPG for Approximation of Action Space

Deep RL in Large Discrete Action Spaces [10]

Following Dulac-Arnold et al, we will *embed* our discrete action space into a continuous action space to use algorithms built for large action space exploration, such as Deep Deterministic Policy Gradient [9].

Algorithm 1 Wolpertinger Policy

State $\mathbf{s}$ previously received from environment.
$\hat{\mathbf{a}} = f_{\theta^\pi}(\mathbf{s})$ {Receive proto-action from actor.}
$\mathcal{A}_k = g_k(\hat{\mathbf{a}})$ {Retrieve k approximately closest actions.}
$\mathbf{a} = \arg\max_{\mathbf{a}_j \in \mathcal{A}_k} Q_{\theta^Q}(\mathbf{s}, \mathbf{a}_j)$
Apply $\mathbf{a}$ to environment; receive $r, \mathbf{s}'$.

Deep RL in Large Discrete Action Spaces

We must answer the follow questions for this technique:

1. **How do we embed our action space?**

   Our probability matrix is a continuous version of our action space.

2. **How do we find “nearby” actions for our selected continuous action?**

   One possibility is to simply sample from the mask distribution

How do we adapt the DDPG algorithm?

The central question for this algorithm is what loss the critic network predicts.

- Sum of weighted mask loss and prediction loss?
- Separately predict weighted mask loss and prediction loss?
- Separately predict unweighted mask loss and prediction loss?

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Critic Loss Variations

Our options for adapting DDPG include the following choices.

1. Naive Loss:

$$\nabla Q = \mathcal{L}_p + \lambda \mathcal{L}_m - Q$$

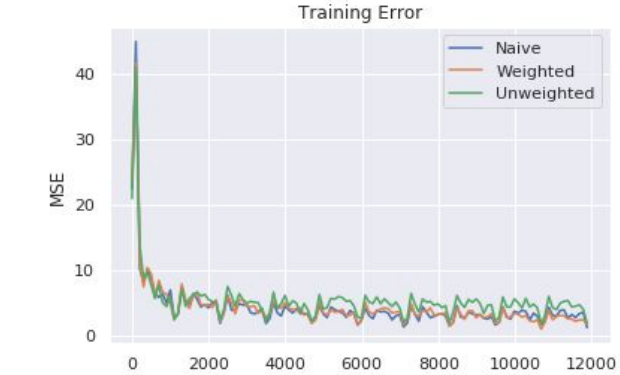
2. Weighted and Separated Loss:

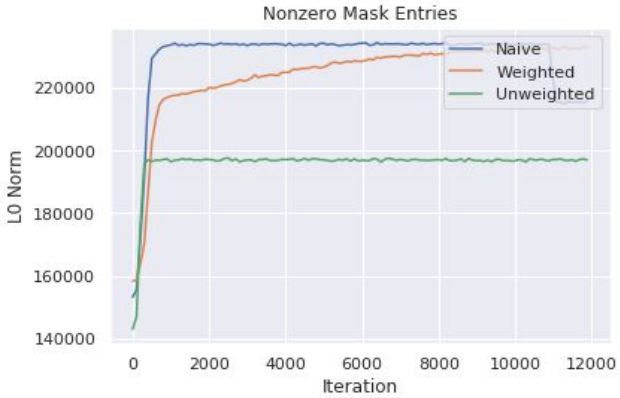
$$\nabla Q_p = \mathcal{L}_p - Q_p \qquad \nabla Q_m = \lambda \mathcal{L}_m - Q_m$$

3. Unweighted and Separated Loss:

$$\nabla Q_p = \mathcal{L}_p - Q_p \qquad \nabla Q_m = \mathcal{L}_m - Q_m$$

DDPG Results





DDPG Takeaways

From these results we can see that though the unweighted and separated loss version did not decrease the critic loss in a noticeable way over the course of training, it did result in a lower number of nonzero mask entries.

Effect of Image Scale

Scales for Comparison

100 x 100 subregion

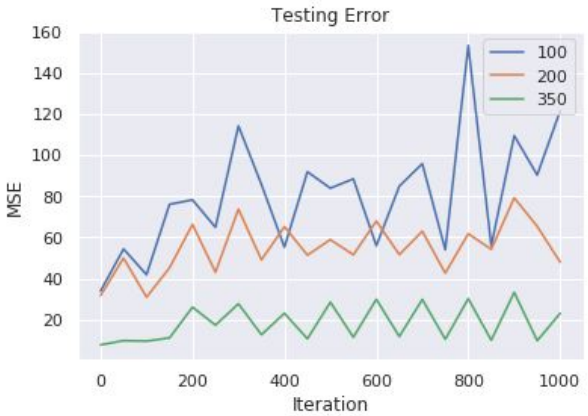
200 x 200 subregion

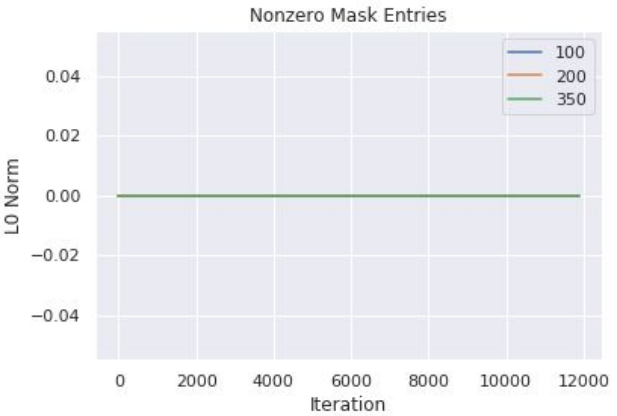
350 x 350 subregion

Homogeneity of Datasets

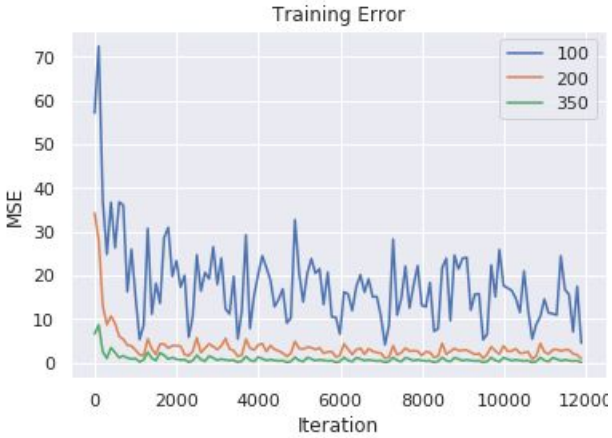
Dataset	Mean	Standard Deviation	Range
100x100 KITTI	1.4802	4.9628	23.4656
200x200 KITTI	1.7833	4.4191	32.3827
350x350 KITTI	1.3478	3.7994	32.2584
Full-Sized KITTI	0.8368	5.5685	72.6899

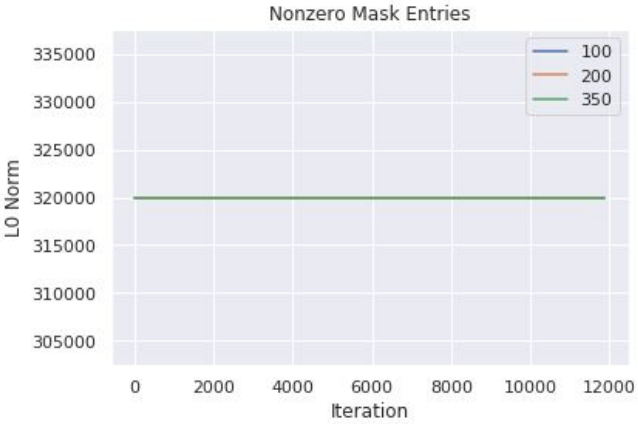
Baseline Performance



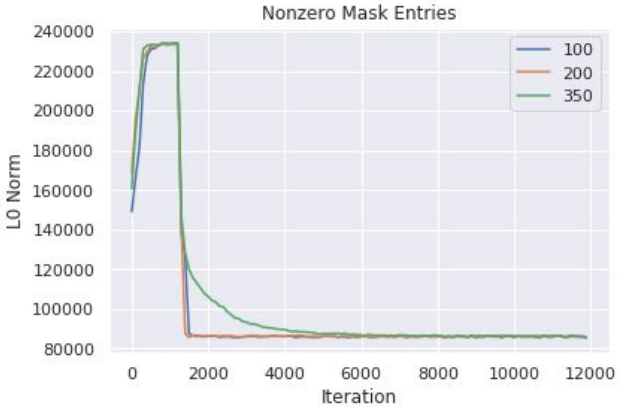


Vanilla Performance

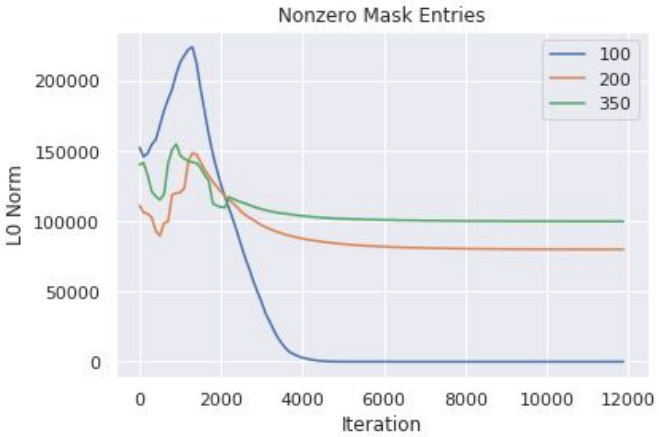




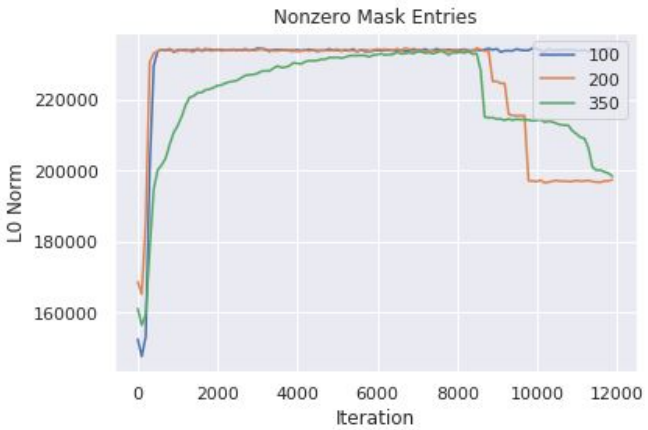
Gumbel Performance



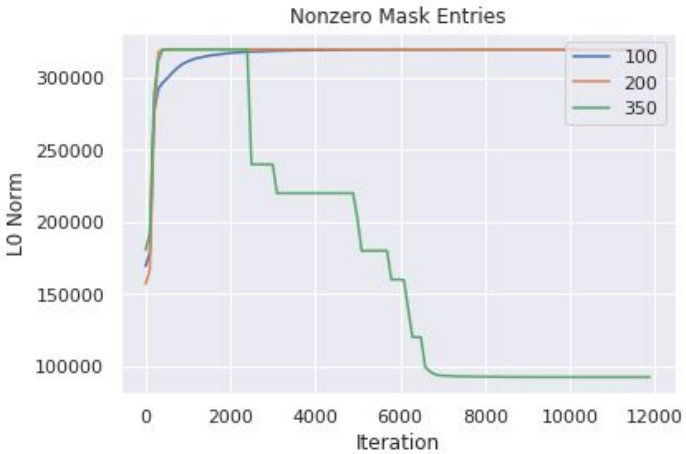
A2C Performance



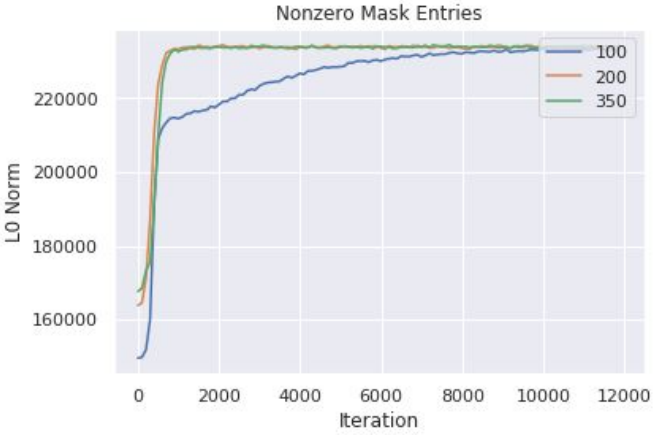
Target Performance



REINFORCE Performance



DDPG Performance



Homogeneity Takeaways

Counterintuitively, models performed better in terms of testing and training loss on larger image subregions, even though the range of their depths is much higher on average. Furthermore, the models frequently achieved greater sparsity than on the smaller subregions.

Plausibility of Masks

How can we tell if the masks are effective?

So far we've been measuring effectiveness with low L0 norm and low prediction error. Alternatives to this are:

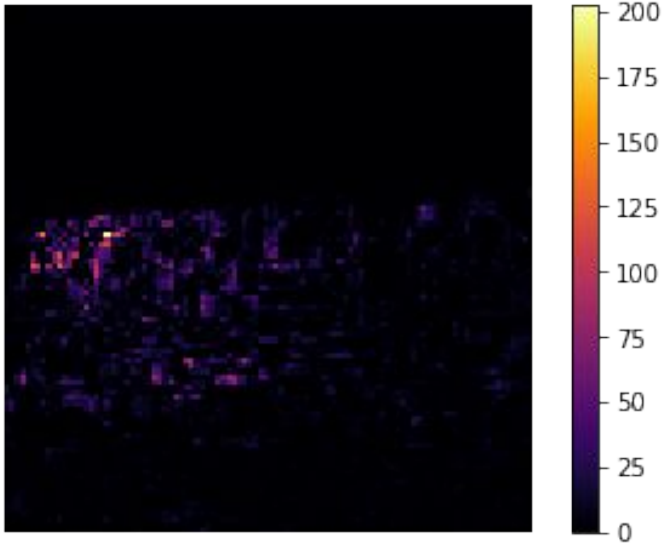
- Decline in performance when using mask from uniform distribution.
- Correlation between mask entries and error surface.

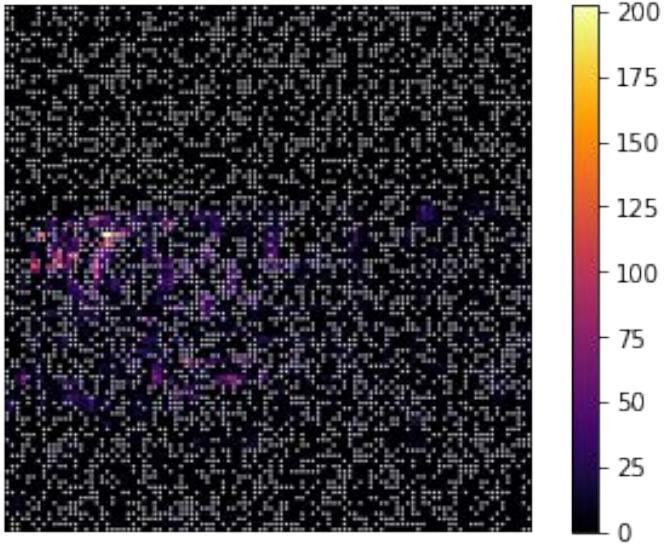
Experimental Set-up

To limit ourselves in the initial steps of this goal, I've focused on using models trained by Gumbel for a variety of steps on 350 x 350 subregions of KITTI scaled to 100 x 100 pixel RGB images.

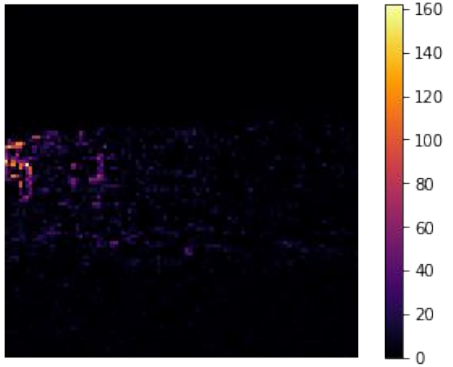
This is because Gumbel still achieves the greatest sparsity most consistently.

Error Surface and Selected Pixels

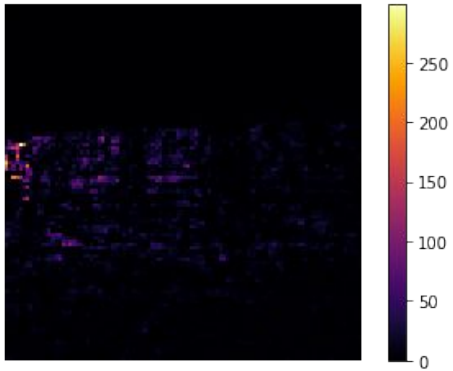




Utility of Real vs Random Masks



Masking Network



Random Mask

Conclusion

Takeaways

- DDPG approach has a lot of potential but many versions to explore.
- Paradoxically, almost all methods perform better on the larger image subregions, even when the number of pixels is held equal.
- Our masking network and prediction network learn to work together in a way that doesn't improve if we directly give the prediction network more ground truth...

Plans Moving Forward

- Continue to explore methods for training the masking/prediction system with continuous action-space RL techniques.
- Strip down existing methods with the lessons learn to see what is essential to the relative success of these methods.
- Establish more connections between the plausibility of masks and techniques for training models.

Github Repositories

Public Github Links

- Sparse Aerial Depth Completion:

  https://github.com/austinbdill/sadc

- Active Learning for Pixelwise Prediction:

  https://github.com/austinbdill/pixelwise_al

References

References

1. KITTI Depth Completion Dataset

   http://www.cvlibs.net/datasets/kitti/index.php

2. Playing Atari with Deep Reinforcement Learning

   https://arxiv.org/abs/1312.5602

References

3. Categorical Reparameterization with Gumbel Softmax

   https://arxiv.org/pdf/1611.01144.pdf

4. Learning Selection Masks for Deep Neural Networks

   https://arxiv.org/pdf/1906.04673.pdf

References

5. Learning Associative Markov Networks

   http://robotics.stanford.edu/~koller/Papers/Taskar+al:ICML04.pdf

6. Undirected Graphical Models - Probabilistic Machine Learning

   https://www.cs.ubc.ca/~murphyk/MLbook/pml-print3-ch19.pdf

References

7. Policy Gradient Methods for Reinforcement Learning with Function Approximation

   https://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf

8. Asynchronous Methods for Deep Reinforcement Learning

   https://arxiv.org/abs/1602.01783

References

9. Continuous Control with Deep Reinforcement Learning

   https://arxiv.org/abs/1509.02971

10. Deep Reinforcement Learning in Large Discrete Action Spaces

    https://arxiv.org/abs/1512.07679

Github References

11. PyTorch Straight Through Gumbel Softmax Implementation:

https://gist.github.com/yzh119/fd2146d2aeb329d067568a493b20172f