

Active Learning for Pixel-wise Prediction

April 2nd, 2020

Email: abdill@andrew.cmu.edu

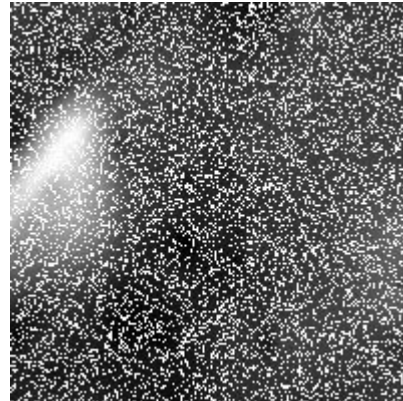


Overview of Presentation

1. Depth Estimation Task
2. Overview of Problem Set-up
3. Gumbel Softmax
4. Reinforcement Learning Techniques
5. Ising Model Approach
6. Takeaways and Plans Moving Forward

Depth Completion Task

- Input: RGB aerial image and corresponding depth map with some percentage replaced with zeros.



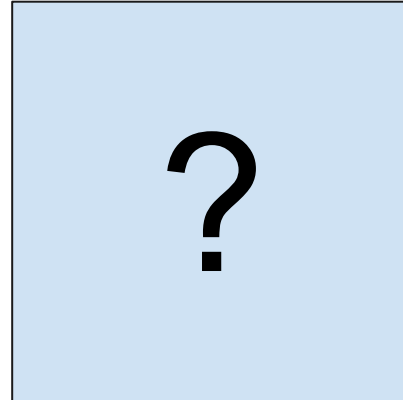
Depth Completion Task

- Output: Predicted depth map

$$\text{Error} = \left(\underbrace{\text{Prediction}} - \underbrace{\text{Ground Truth}} \right)^2$$

Depth Estimation Task with Active Learning

- Input: RGB aerial image



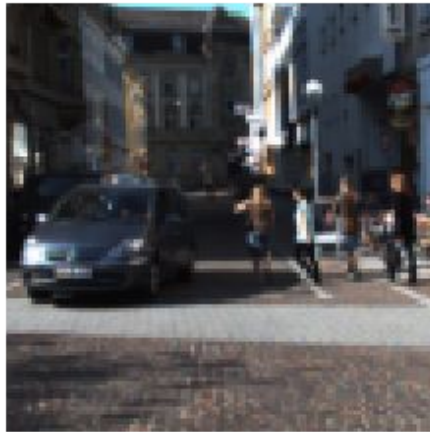


KITTI Dataset Details and Preprocessing

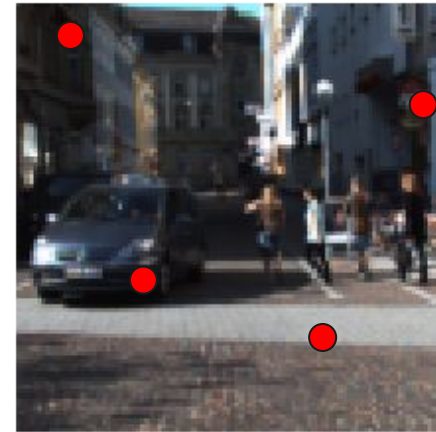
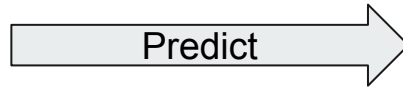
- 38,323 training images and 3,347 testing images.
- Initial images are 374 x 1238 pixels and have 3 channels intended for training self-driving cars.
- 350 by 350 subregions are cropped out.
- These square regions are downsampled to be 100 by 100 pixels in order to facilitate faster training.

General Problem Set-Up

Masking Network Goal



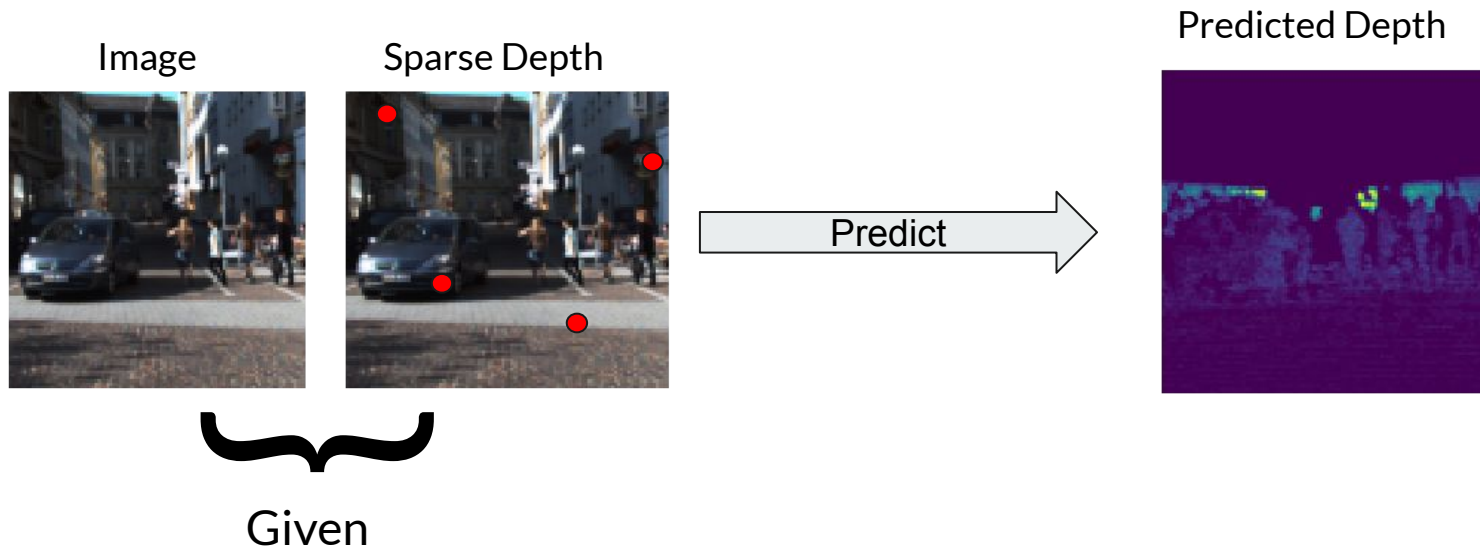
Input Image



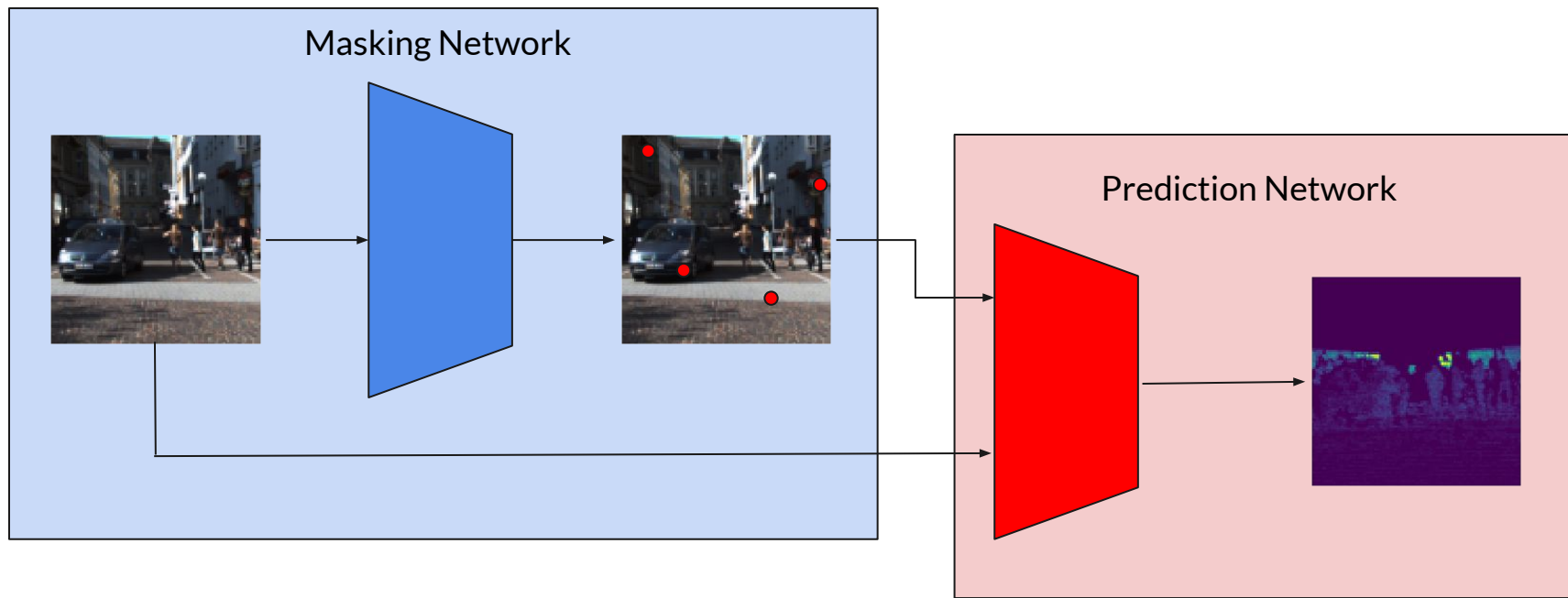
Suggested Annotation
Locations

Represented as
binary mask

Prediction Network Goal



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?

Gumbel Softmax Approach

Recent relevant work suggests part of the path forward.

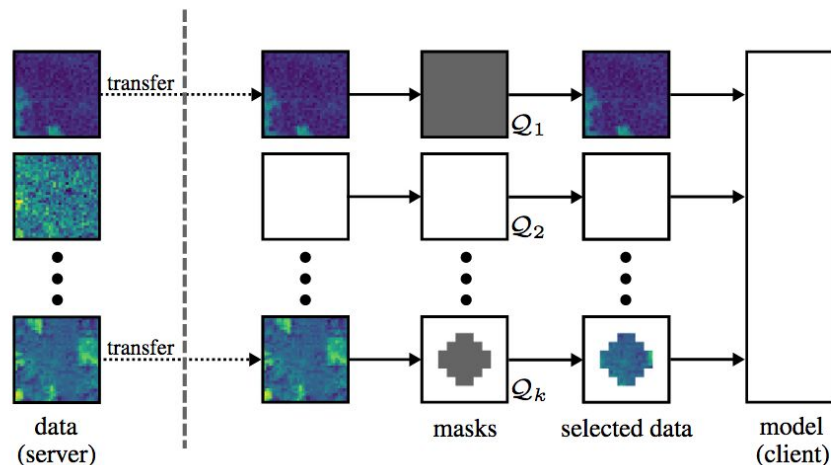
Learning Selection Masks for Deep Neural Networks

Stefan Oehmcke
Department of Computer Science
University of Copenhagen
stefan.oehmcke@di.ku.dk

Fabian Gieseke
Department of Computer Science
University of Copenhagen
fabian.gieseke@di.ku.dk

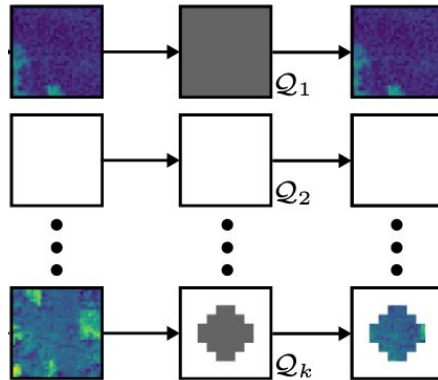
Abstract

Data have often to be moved between servers and clients during the inference phase. For instance, modern virtual assistants collect data on mobile devices and the data are sent to remote servers for the analysis. A related scenario is that clients have to access and download large amounts of data stored on servers in order to apply machine learning models. Depending on the available bandwidth, this data transfer can be a serious bottleneck, which can significantly limit the application machine learning models. In this work, we propose a simple yet effective framework that allows to select certain parts of the input data needed for the subsequent application of a given neural network. Both the masks as well as the neural network are trained simultaneously such that a good model performance is achieved while, at the same time, only a minimal amount of data is selected by the masks. During the inference phase, only the parts selected by the masks have to be transferred between the server and the client. Our experimental evaluation indicates that it is, for certain learning tasks, possible to significantly reduce the amount of data needed to be transferred without affecting the model performance much.

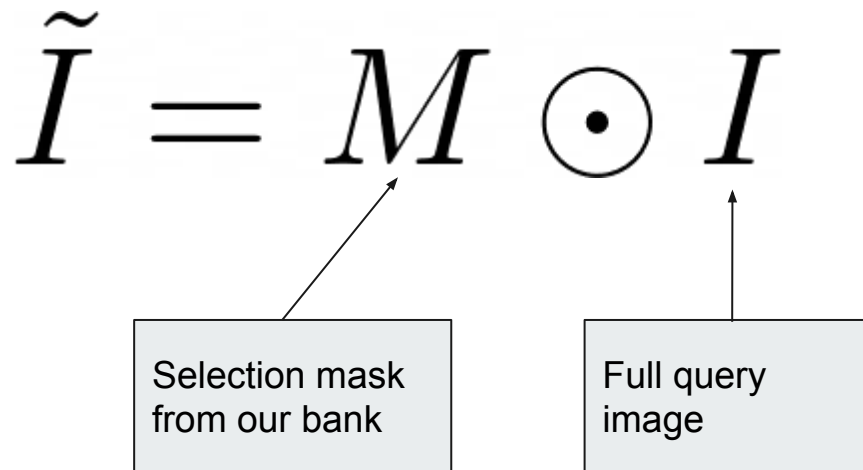


What is their task?

- Start with a set of arrays of the same size as their images initialized with random values between $(0+\epsilon, 1-\epsilon)$.
- Want to update this arrays to learn “selection” masks to trade of the success of their ML task and the sparsity of their masks.



Rephrased with our Notation


$$\tilde{I} = M \odot I$$

Selection mask from our bank

Full query image

$$\hat{S} = h_{\theta}(\tilde{I})$$

One way to encourage sparsity is through regularization.

$$\text{Error} = \left(\underbrace{\text{Prediction}} - \underbrace{\text{Ground Truth}} \right)^2 + \lambda ||M||_p$$

Gumbel Masking

$$\tilde{S} = M \odot S$$

Output from a neural
net parameterized
Gumbel Softmax

Ground truth
segmentation

$$\hat{S} = h_{\theta}(I, \tilde{S})$$



How do they get binary masks?

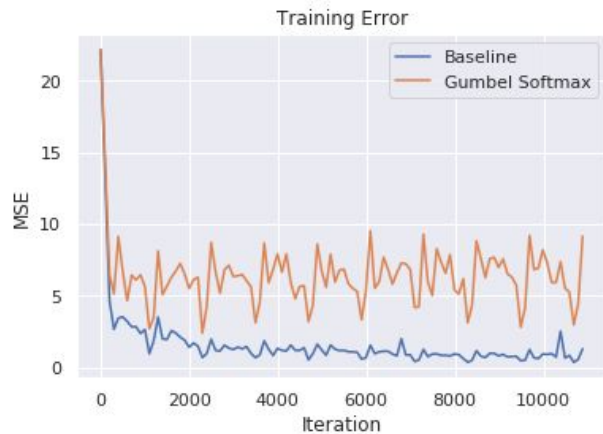
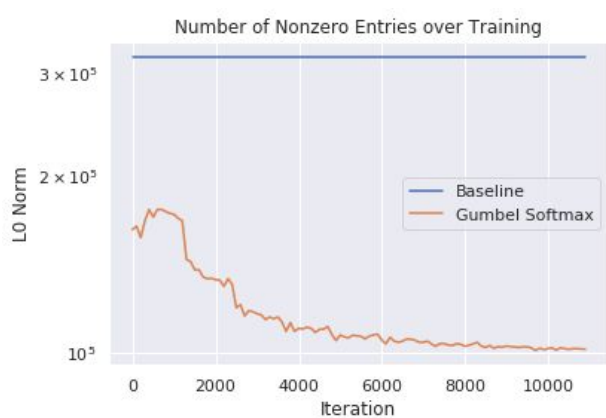
$$M_i = \text{softmax} \left(\frac{g_0 + \log \pi_i}{\tau}, \frac{g_1 + \log(1 - \pi_i)}{\tau} \right)$$



Gumbel Softmax Training Parameters

- Temperature: Gradually decrease the temperature to more closely approximate the categorical distribution
- Noisiness: Trade-off between exploration and exploitation by switching between noisy and noiseless samples.
- Regularization Term: Gradually increase to prevent lack of decrease in sparsity over training.

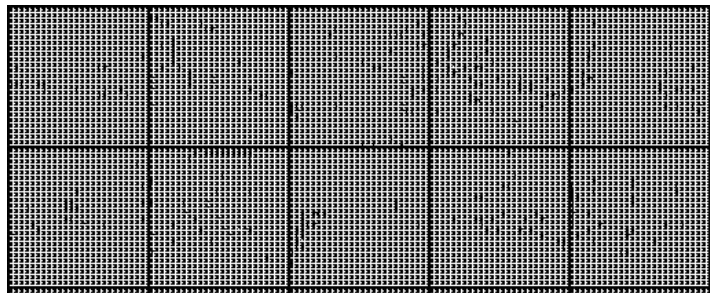
Gumbel Masking Results



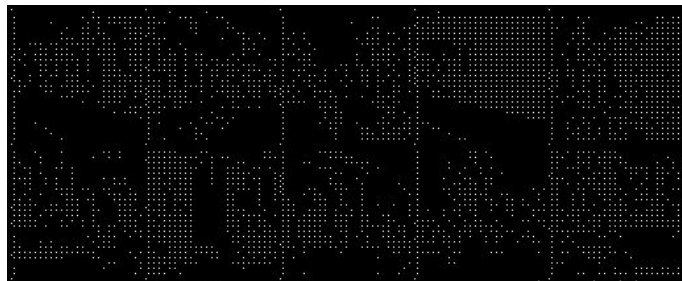
Gumbel Masking Results



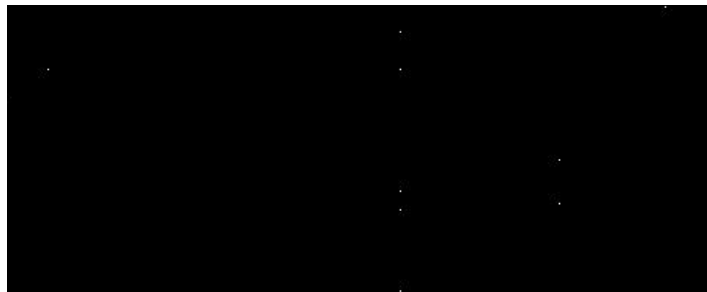
Epoch 1



Epoch 5



Epoch 10



Reinforcement Learning Approaches



Decouple Training Cooperative Networks

One approach to this problem is to notice that if we had a perfect masking network we could fix its weights and easily train the prediction network. The same is true if we had a perfect prediction network and wanted to train the masking network.

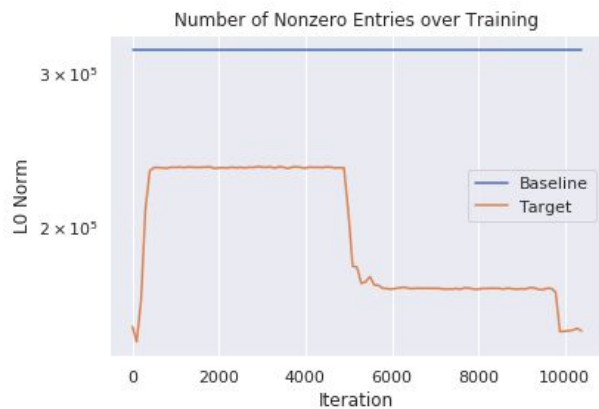
This is similar to the essential idea that made Deep Q-Learning possible by having “target” networks that change more gradually over time.



Decouple Training Cooperative Networks

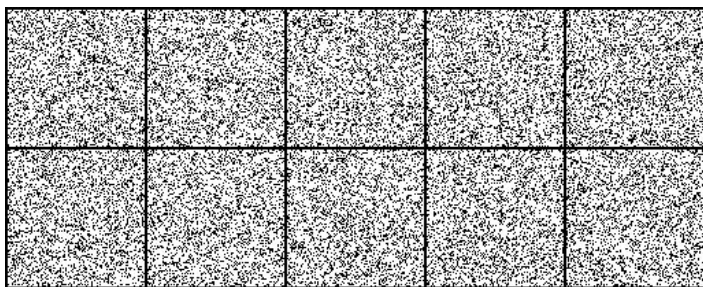
- Initialize masking network, prediction network, and copies of both as our “targets”.
- Train prediction network with output from target masking network for a while.
- Train masking network with output from target prediction network for a while.
- Repeat until convergence.

Moving Target Results

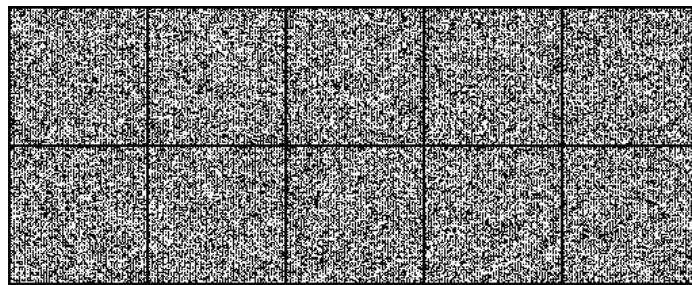


Moving Target Results

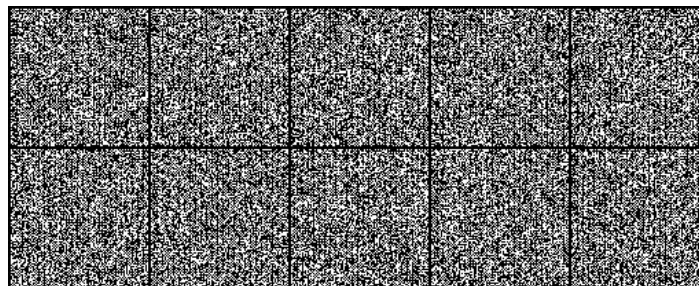
Epoch 1



Epoch 5



Epoch 10





REINFORCE: Let's try Monte Carlo Sampling

- Measure of performance: $\mathcal{L}(M)$
- Probability distribution: $P_{\theta}(M)$
- Parameters we're optimizing over: θ
- Loss function we want to optimize with gradient descent:

$$\mathbb{E}_{M \sim P_{\theta}}[\mathcal{L}(M)]$$



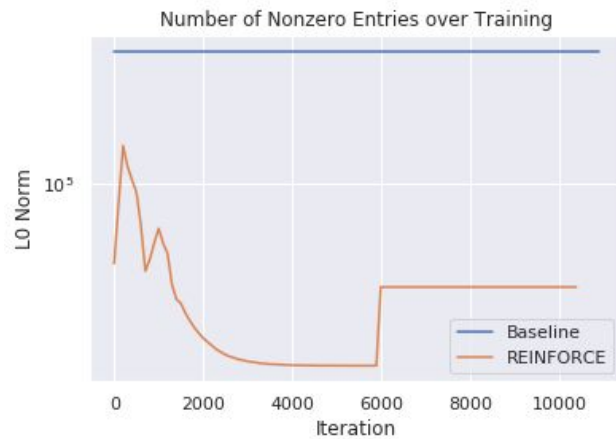
REINFORCE: Let's try Monte Carlo Sampling

We can optimize this function with the following gradient:

$$\mathbb{E}_{M \sim P_{\theta}} \nabla_{\theta} \log P_{\theta}(M) [\mathcal{L}(M)]$$

Note that the measure of success is never differentiated. We can therefore include any regularizer and error function on the training data without backpropping through the prediction network.

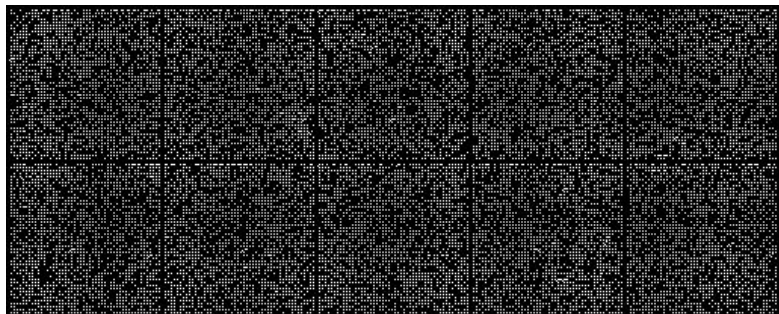
REINFORCE Results



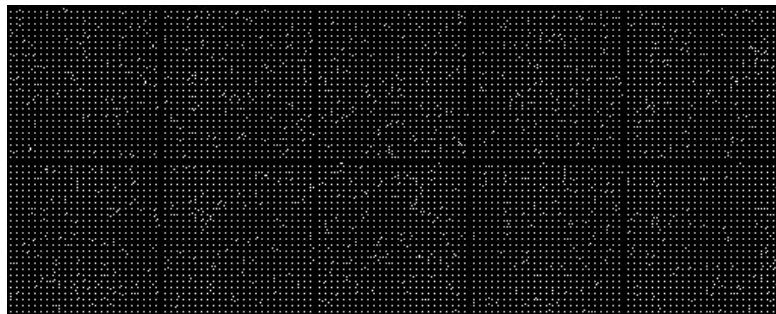
REINFORCE Results



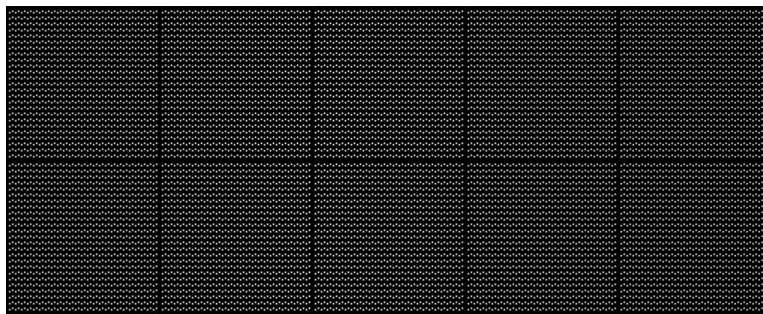
Epoch 1



Epoch 5



Epoch 10



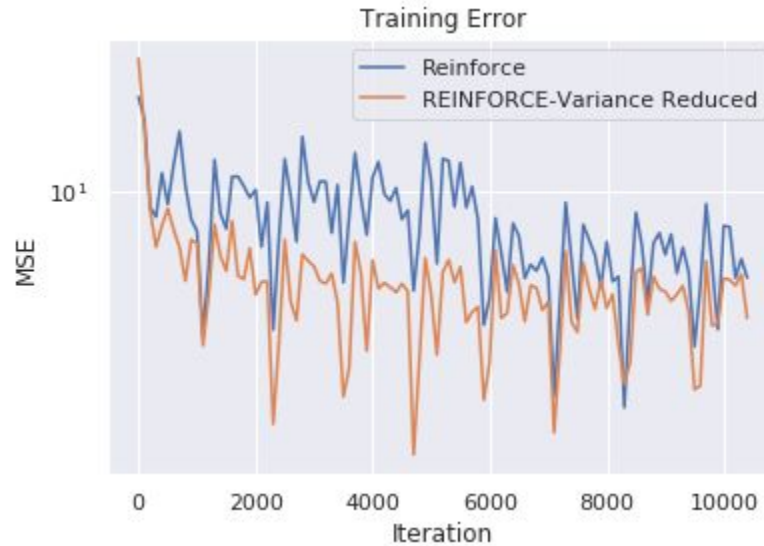


Decreasing REINFORCE Variance

We can decrease the variance of the score gradient estimator by subtracting a constant value from the loss function. Frequently, this constant is the batch mean loss.

$$\mathbb{E}_{M \sim P_{\theta}(M)} \nabla_{\theta} \log P_{\theta}(M) [\mathcal{L}(M) - \mathbb{E}_M[\mathcal{L}M]]$$

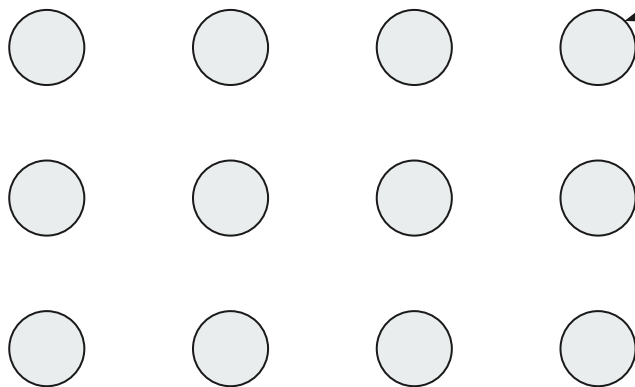
REINFORCE Results - Decrease Variance



Ising Model Approach

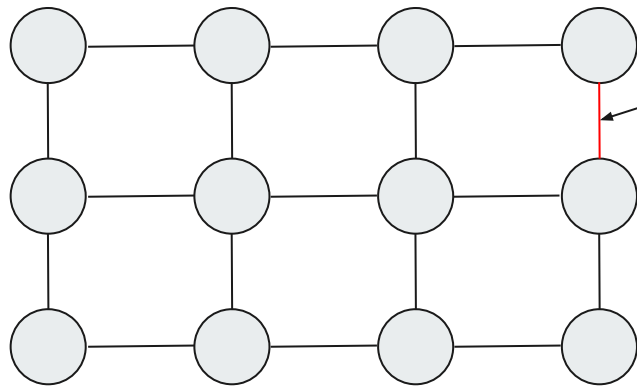
Unary Potentials

How much do we like
this node in each
state?



$$\sum_{i=0}^{nm} \theta_u^T \phi(x_i)$$

Pairwise Potentials



How much do we like
this pair of nodes to
be in the same state

$$\sum_{j,k \in \mathcal{E}} \theta_p^T \phi_{jk}(x_j, x_k)$$



Probability in the Resulting MRF Random Field

$$P(\mathbf{x}; \theta) \propto \exp \left(- \sum_{i=0}^{nm} \theta_u^T \phi(x_i) - \sum_{j,k \in \mathcal{E}} \theta_p^T \phi_{jk}(x_j, x_k) \right)$$

Probability in the CRF

$$\underline{y} \triangleq \text{vect}(M)$$

$$y[y = 0] \leftarrow -1$$

$$\log P_{\theta}(y|x) \propto -\frac{1}{2}y^T W y - b^T y$$

Probability in the CRF



$$W \triangleq h_{\theta}(x) + h_{\theta}(x)^T$$

$$h_{\theta}(x) \in \mathbb{R}^{D \times D}$$

Calculating the gradient of the log-probability



$$\begin{aligned}\nabla_w \log P_\theta(y|x) &= -yy^T - \mathbb{E}_y[-yy^T] \\ \nabla_b \log P_\theta(y|x) &= -y - \mathbb{E}_y[-y]\end{aligned}$$



Using REINFORCE with a CRF

- Sampling from this pairwise CRF is less direct than in the vanilla REINFORCE algorithm discussed before. In other words, it's easy to evaluate the probability up to normalization but difficult to sample.
- A possible solution to this is using Gibbs sampling to generate samples from the Monte Carlo gradient estimate.

$$\mathbb{E}_{M \sim P_{\theta}(M)} \nabla_{\theta} \log P_{\theta}(M) [\mathcal{L}(M)]$$

Conclusion

Performance Comparison





Takeaways

- Adjustments to these methods over the last month have led to much better performance when compared to their naive application.
- These methods are high variance and can produce dramatically different results between similar runs, with REINFORCE having the highest variance.
- More advanced probability models can be use but with the cost of additional Monte Carlo sampling to deal with high dimensional distributions.



Plans Moving Forward

1. Continue to combine techniques to yield more stable results.
2. Get Deep Ising Model up and running with REINFORCE training.
3. Research use of hierarchical/overlapping group LASSO with deep learning models to overcome our sparsity barriers.

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

- KITTI Depth Completion Dataset

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

- Playing Atari with Deep Reinforcement Learning

<https://arxiv.org/abs/1312.5602>



References

- Categorical Reparameterization with Gumbel Softmax

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

- Learning Selection Masks for Deep Neural Networks

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



References

- Learning Associative Markov Networks

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

- Undirected Graphical Models - Probabilistic Machine Learning

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



Github References

- PyTorch Straight Through Gumbel Softmax Implementation:

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