

# Active Learning for Pixel-wise Prediction

February 28th, 2020

Email: [abdill@andrew.cmu.edu](mailto:abdill@andrew.cmu.edu)

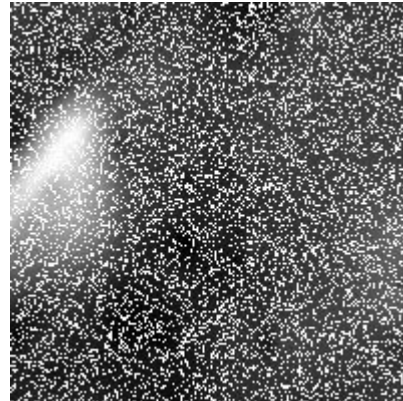


# Overview of Presentation

1. Depth Estimation Task
2. KITTI Depth Estimation Dataset
3. Overview of Problem Set-up
4. Baseline Results on New Dataset
5. Naive Model Results on New Dataset
6. Gumbel Softmax Results
7. Reinforcement Learning Techniques
8. Ising Model Approach
9. Plan Moving Forward

# Depth Completion Task

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



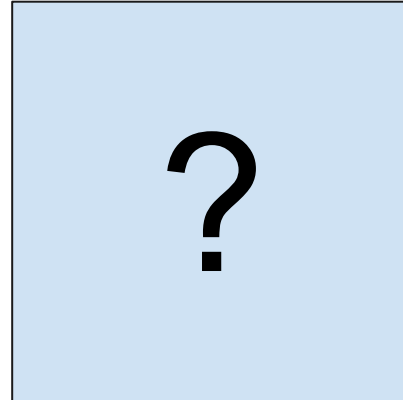
# Depth Completion Task

- 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

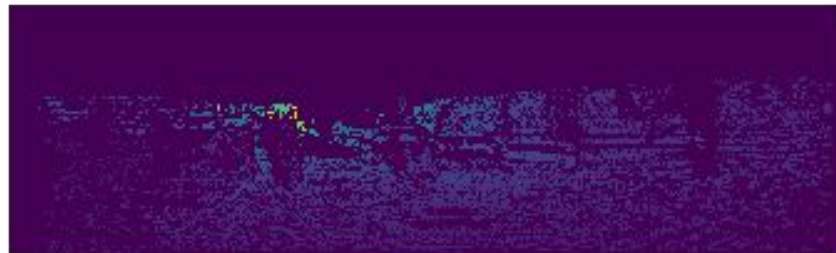


## Moving to KITTI Depth Prediction Dataset

Most of our work so far has been with a small segmentation dataset. Our experiments have been successful enough that we've moved to a true depth prediction task.



RGB Image



Ground Truth Depth



## 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.

# Dataset Selections after Processing



RGB Image



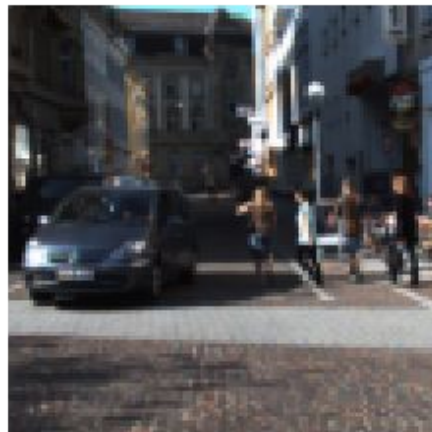
Ground Truth Depth



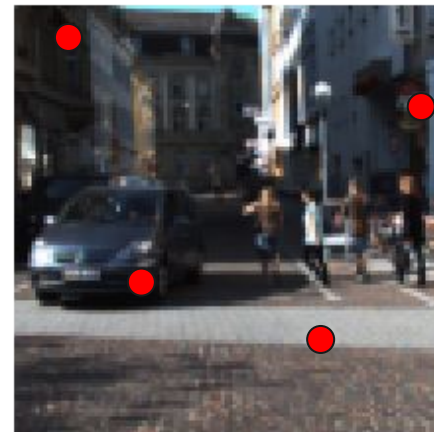
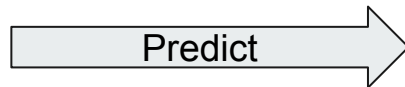
# General Problem Set-Up

---

# Masking Network Goal



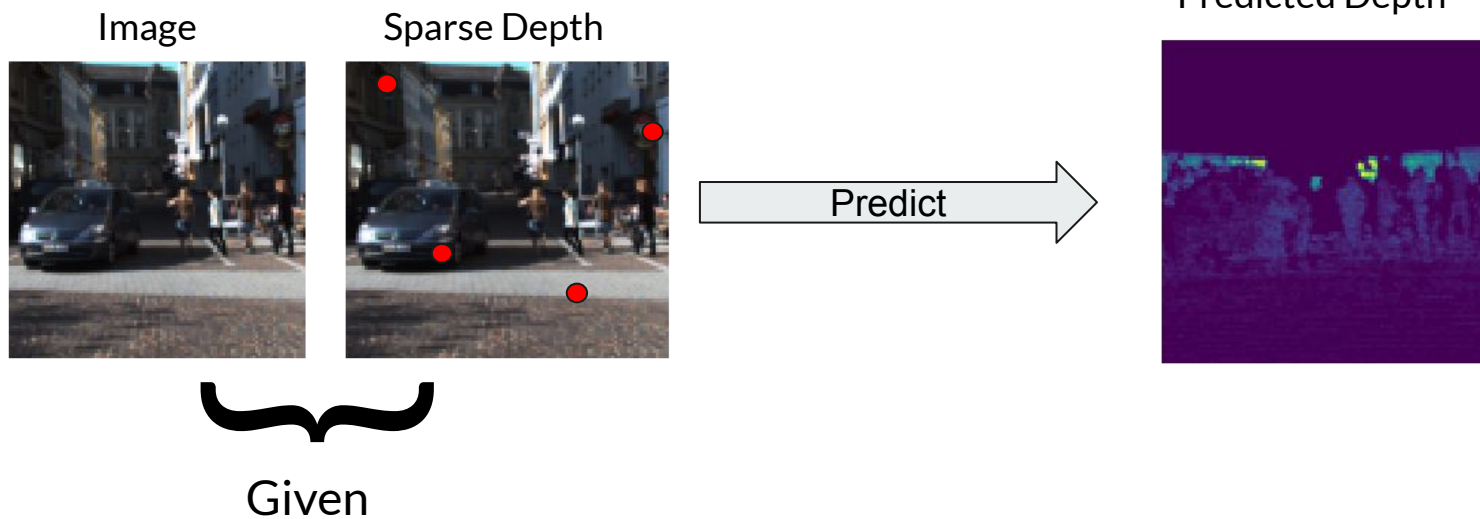
Input Image



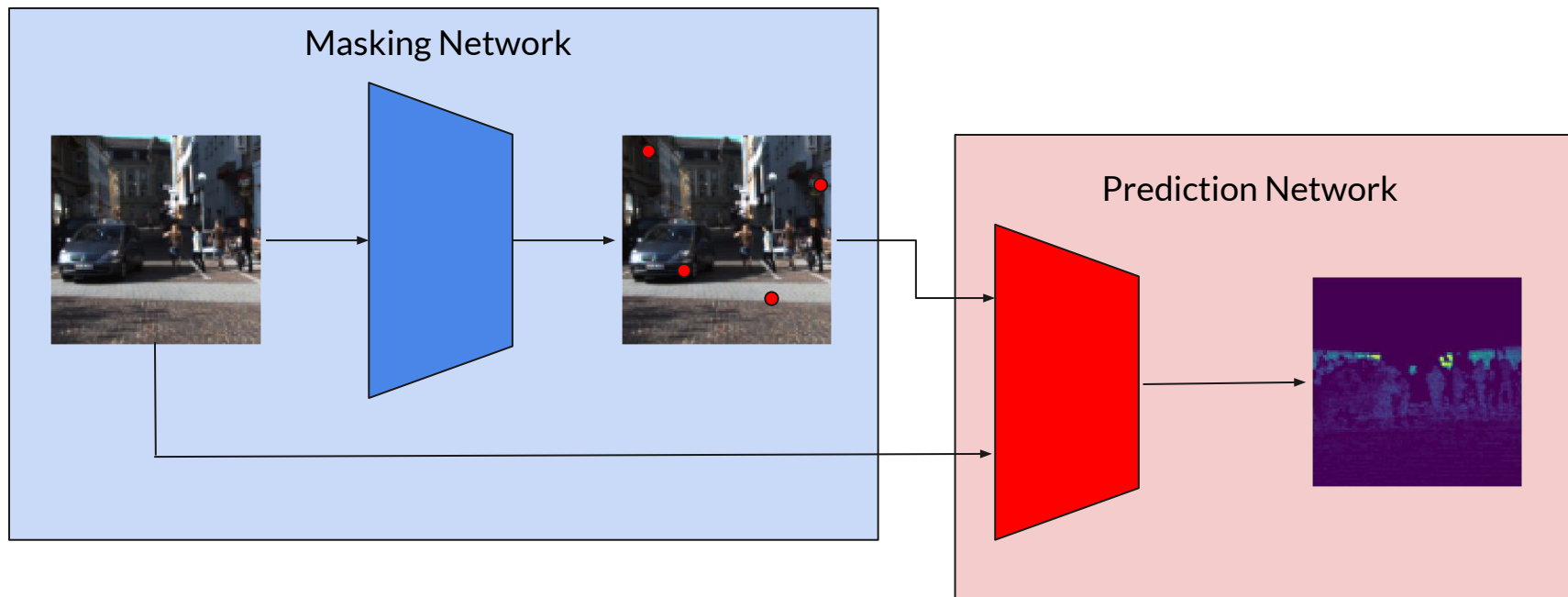
Suggested Annotation Locations

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: 
$$\sum_{i=0}^n \sum_{j=0}^m \mathbb{I}\{M_{ij} \neq 0\}$$



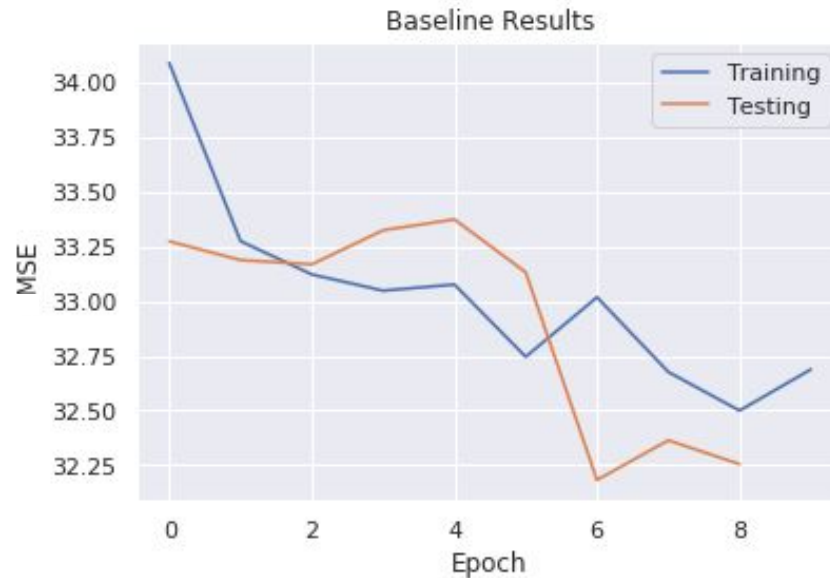
## Two Essential Questions for this Approach

- How do we define the masking network?
- How do we train this model jointly?

# Baseline Results

---

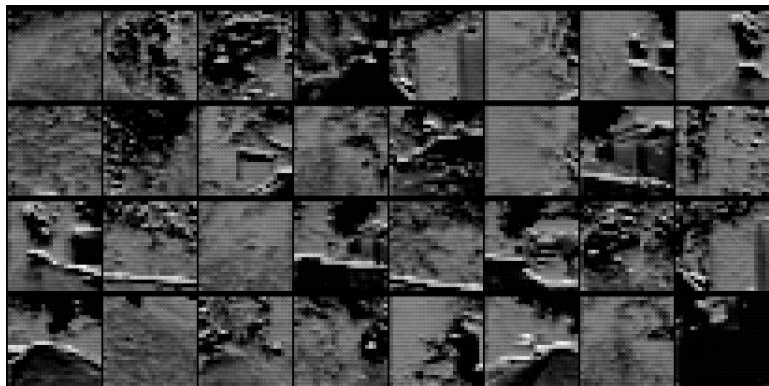
# Results with no Sparse Input





# Results with no Sparse Input

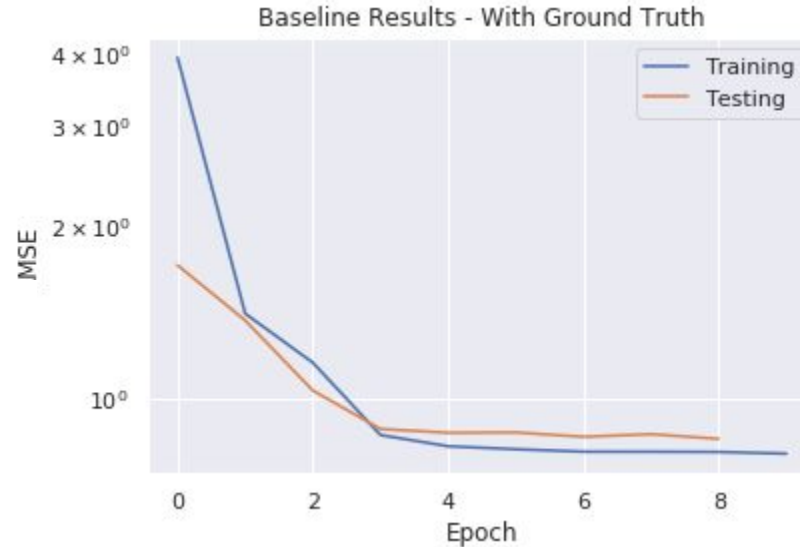
Model Prediction



Ground Truth



# Results with Ground Truth Provided



# Results with Ground Truth Provided

Model Prediction



Ground Truth



# Naive Model

---

# Naive Masking

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

Output from  
masking network

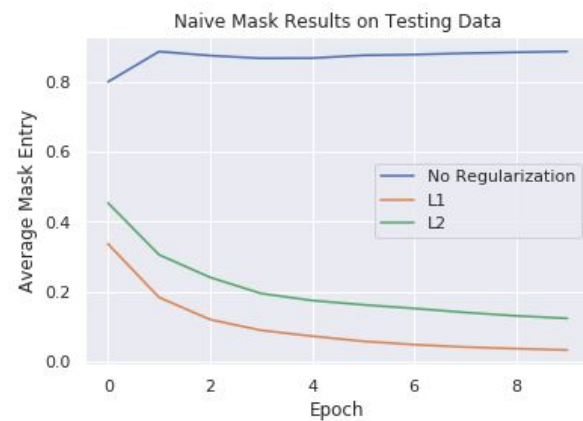
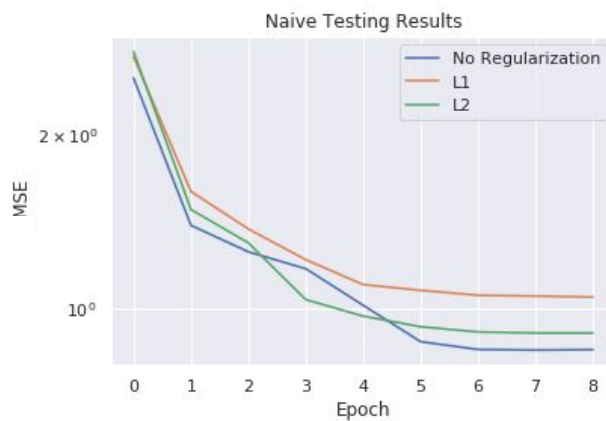
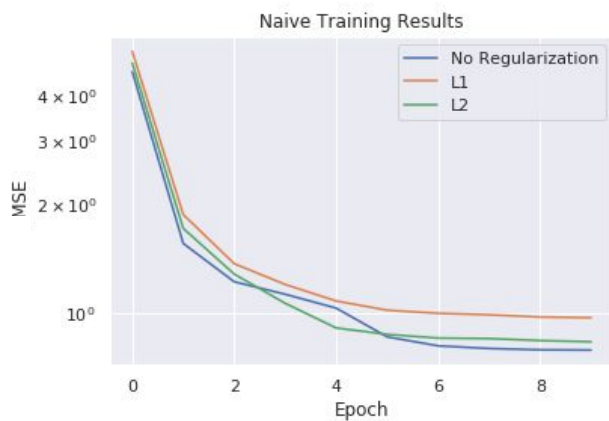
Ground truth  
segmentation

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

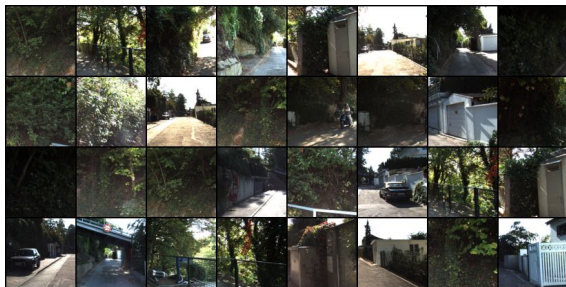
One way to encourage sparsity is through regularization.

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

# Naive Model Results



# Naive Model Results



Model Prediction



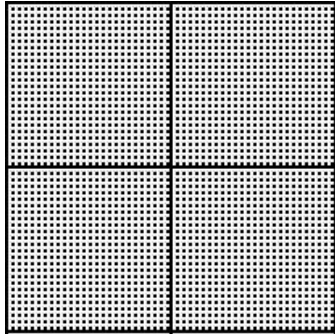
Ground Truth



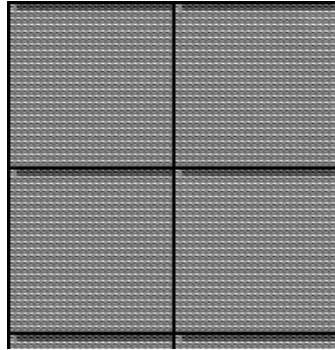




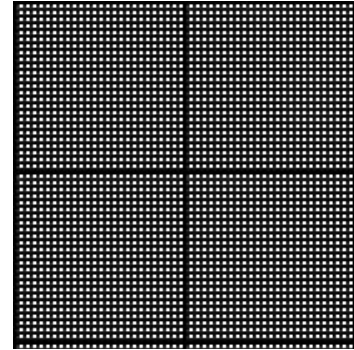
## Naive Model Results - Masks



No Regularization



L2 Penalty



L1 Penalty

# Gumbel Softmax Approach

---

# Recent relevant work suggests part of the path forward.

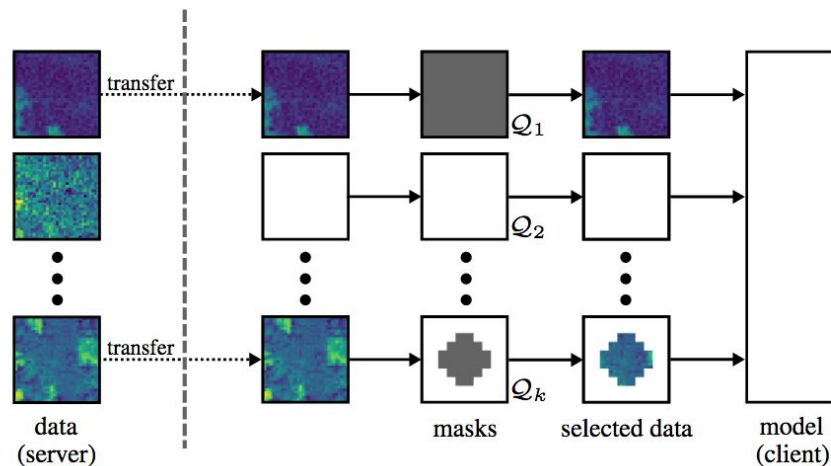
## Learning Selection Masks for Deep Neural Networks

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

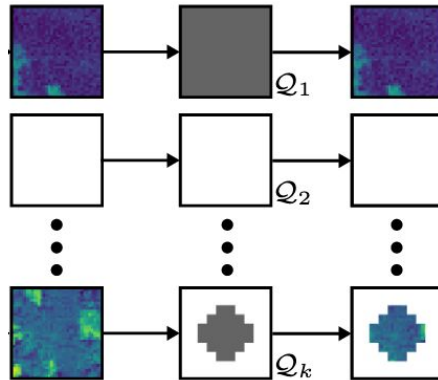
### Abstract

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.

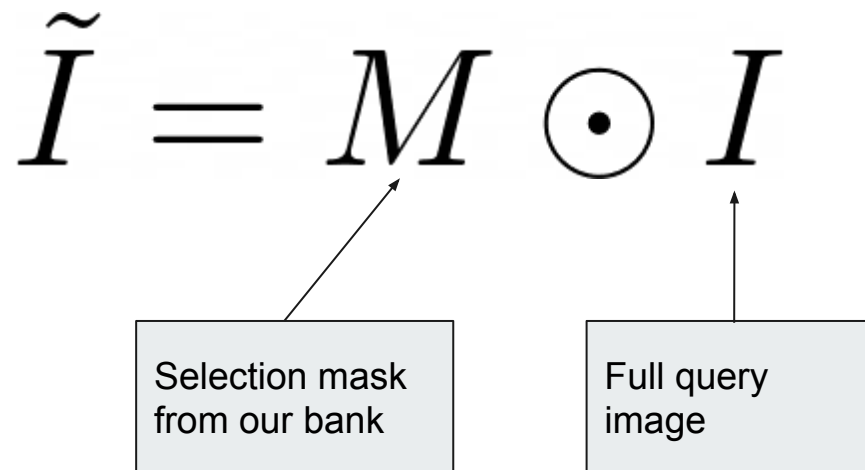


## What is their task?

- 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

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



**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 Masking

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

Output from a neural  
net parameterized  
Gumbel Softmax

Ground truth  
segmentation

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

# Gumbel Masking Results



L1 Penalty


L2 Penalty






**Why isn't this working so far?**

# 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.



## REINFORCE: Let's try Monte Carlo Sampling

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

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



## REINFORCE: Let's try Monte Carlo Sampling

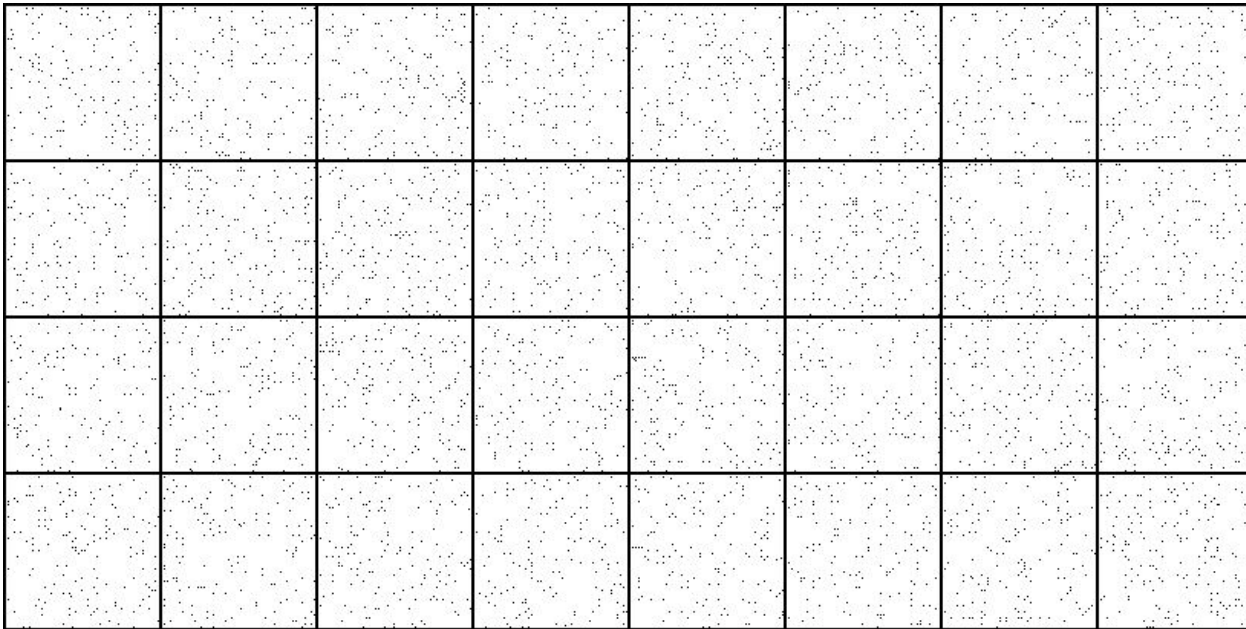
We can optimize this function with the following gradient:

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

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.



## Initial Attempt - Mask Samples



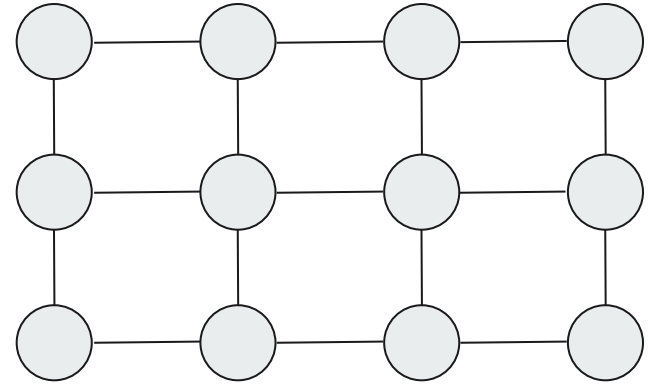
# Ising Model Approach

---



## Motivation for Ising Model

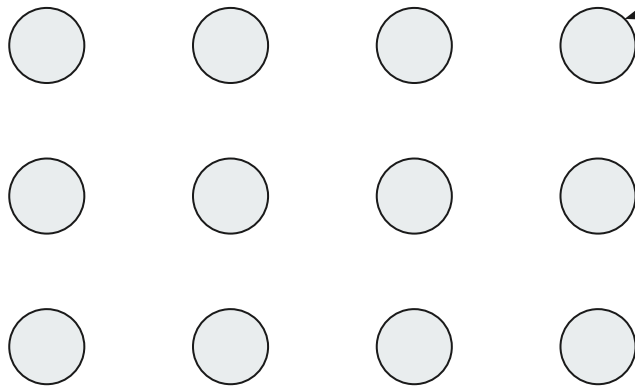
0	1	0	0
0	0	0	1
0	0	0	0



$$x_i \in \{-1, +1\}$$

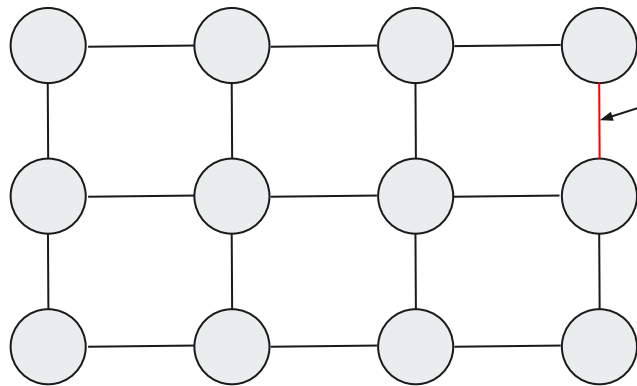
# Unary Potentials

How much do we like  
this node in each  
state?



$$\sum_{i=0}^{nm} \phi_i(x_i)$$

# Pairwise Potentials



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

$$\sum_{i=0}^n \sum_{j=0}^m \phi_{ij}(x_i, x_j)$$



## Probability in the Resulting MRF

$$P(\mathbf{x}) = \exp \left( - \sum_{i=0}^{nm} \phi_i(x_i) - \sum_{j=0}^n \sum_{k=0}^m \phi_{jk}(x_j, x_k) \right)$$



## Prediction Amounts to Inference over this MRF

If we have definitions for the potential functions we simply need to find the most likely setting of variables. Unfortunately even with this simple of a model, this essentially becomes a problem of Integer Linear Programming.

Similar work exists in the paper “Learning Associative Markov Networks”, where they use ILP to solve a maximal margin program over grid-like MRFs.



## Could this be used to train our model?

Not exactly. While we won't be able to train a neural network to predict the mask, we could use a similar trick to the AMN paper with a Structured SVM if we had ground truth masks. Defining these masks would be the essential problem of this approach.



## How could we come up with “Ground Truth” labelings?

- One possibility is to add an additional “target” network like is done with Deep Q Learning to serve as the ground truth for the masking network.
- Another possibility is to find some measure of “usefulness” of a picture such as Mutual Information or sensitivity of gradient of prediction to a pixel’s value.



## Plans Moving Forward

1. Continue to experiment with Gumbel Softmax / RL approaches.
2. Establish stronger relationship between regularization penalty and number of pixels provided to the model.
3. Incorporate Ising Model with different potential functions to encode preference for pixels' inclusion or exclusion.



# 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](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>



## Github References

- PyTorch Straight Through Gumbel Softmax Implementation:

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