Learning Interpretable Representations with Deep Networks
(Structure to the rescue)

Arunkumar Byravan
UW CSE

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Motivation

- Deep networks are great:
  - Huge success in Computer Vision, Robotics, NLP etc
- But, hard to interpret:
  - Intermediate representations not meaningful
  - Weights can be visualized after training (edge filters, patterns etc)
  - Almost black box function approximators

Motivation

- In practice, we know a lot about the problem domain:
  - **Ex:** Scenes have objects (not pixels)
  - Physics can model the world (reasonably well), etc.

- Make use of prior knowledge about the problem, data distribution etc:
  - Structural priors for deep networks
  - **Pros:** Better interpretability, (maybe) improved generalization & data-efficiency
  - **Cons:** Bias-Variance tradeoff, Problem specific

Outline

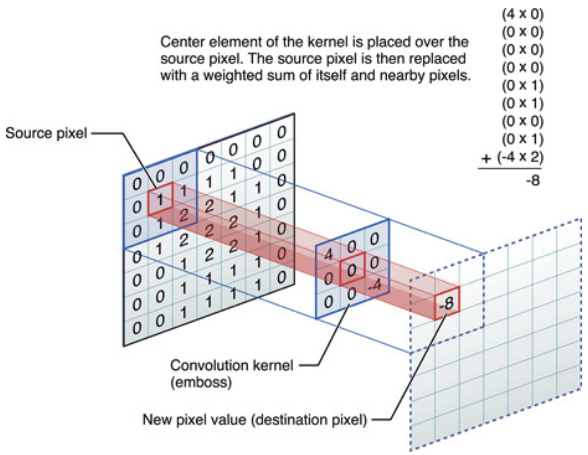
- Structure in the construction:
  - Convolution (LeNet, ImageNet etc.)
  - Affine transforms of data (Transforming auto-encoders, Spatial Transformer Nets)
  - Rigid body dynamics (SE3-Nets)
  - Integrating graphics engines (Neural Scene de-rendering)
- Structured training (Inverse Graphics Net)

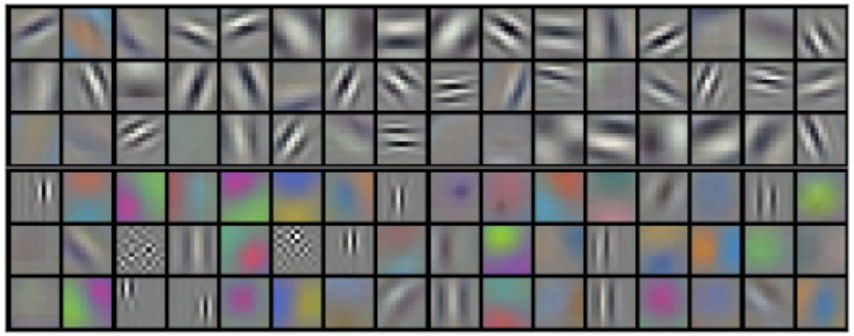
Structure in the construction

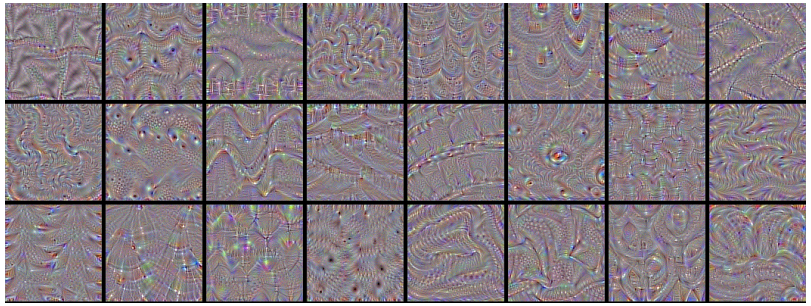
- Allow for specific operations on data:
  - Convolution
  - Transformations of data etc.
- Explicitly assign meaning to intermediate representations:
  - Object segments, transforms, poses, viewpoint, object identity etc.
- Output predictions based on learned representations and defined operations:
  - Trained end to end
  - No explicit supervision on intermediate representations

Convolution

- Early networks used fully-connected layers:
  - Large parameter space
  - Global receptive field
- Local operation on image/feature patches
  - Significant reductions in parameters
  - Translation invariance
- Trained filters detect edges, contours, shapes etc



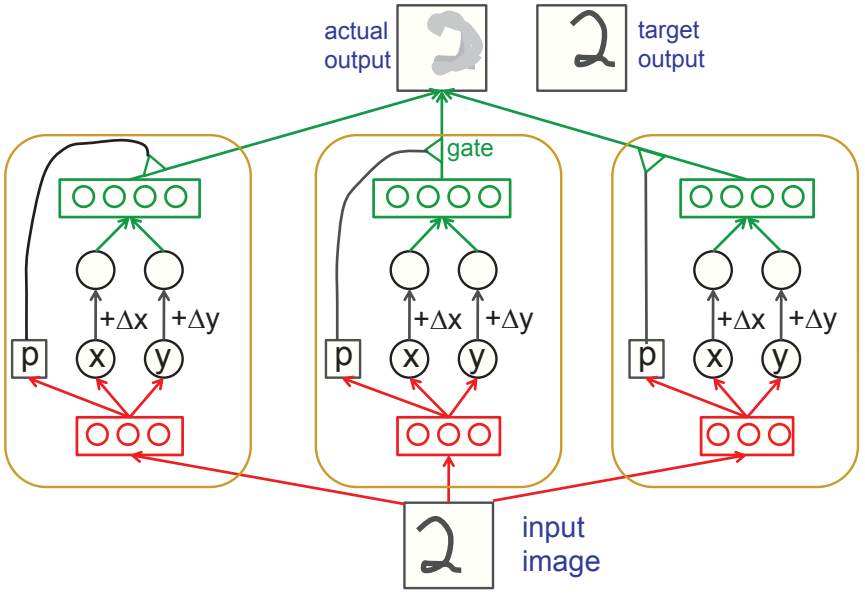




Transforming auto-encoders

(Hinton et al., 2011)

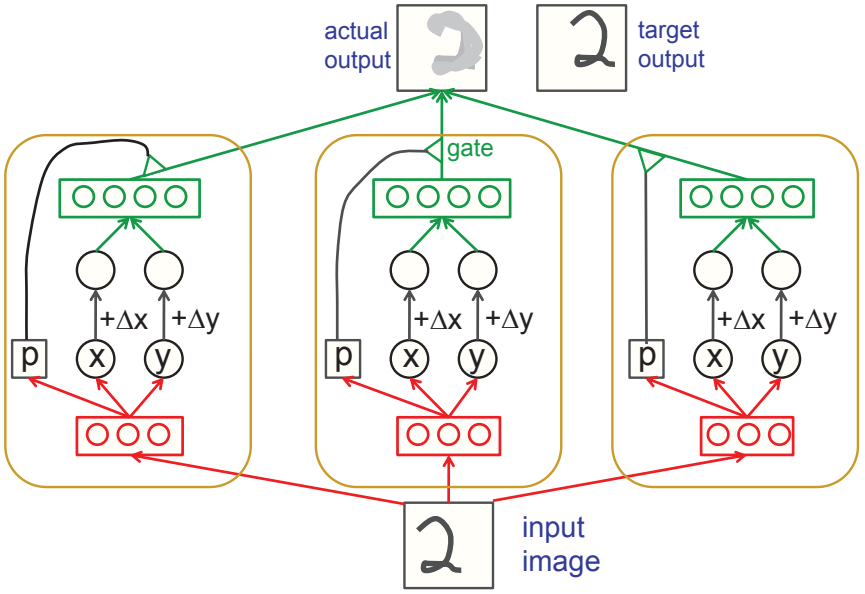
- **Domain:** Scene understanding, Image generation etc
- Network composed of capsules:
  - Learns to identify / reconstruct a visual entity locally
- Recognition units:
  - Predict confidence & instantiation parameters (ex: pose, lighting)
- Generative units:
  - Generates images (patches) based on instantiation parameters
- Combine capsule generations to predict final output



Transforming auto-encoders

(Hinton et al., 2011)

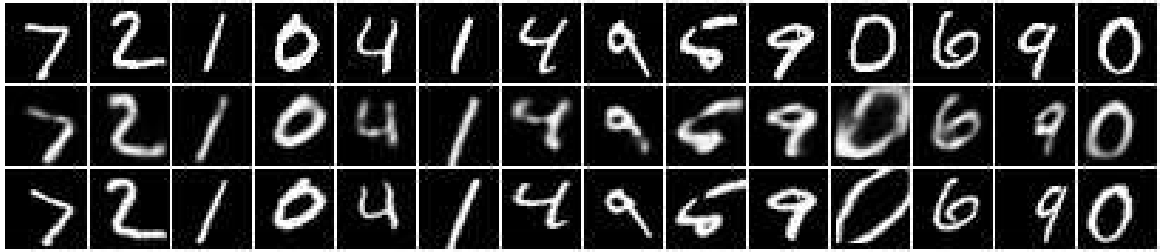
- **Input:** Image of digit, desired pixel shift (dx, dy)
- **Target:** Shifted image of digit
- **Capsule predicts:**
  - Recognition unit predicts position of digit (x,y) & probability it sees the digit (p)
  - Generation unit predicts image shifted by (x + dx, y+dx)
- **Output:** Sum capsule predicted images weighted by p



- Trained end to end
- No supervision on capsule outputs

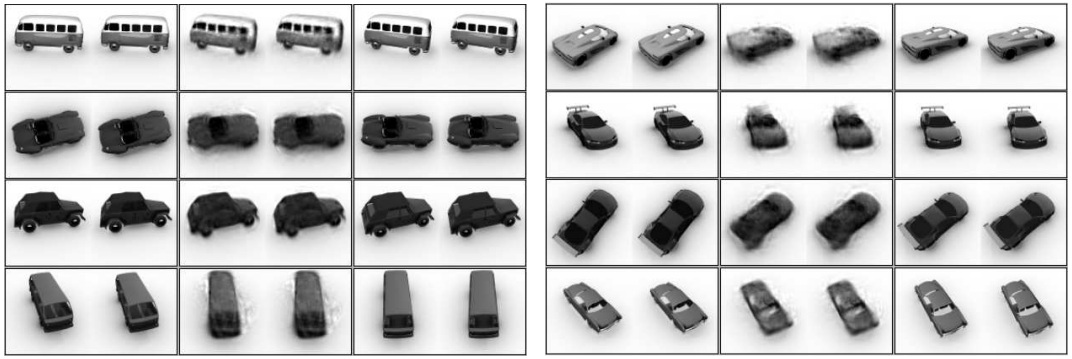
Transforming auto-encoders
(Hinton et al., 2011)

- Affine transforming MNIST digits:



Rows, from top: Input, Output, Target

- 3D rotations of stereo pairs:



Columns, from left: Input, Output, Target

Structure

- Capsules detect objects locally and predict instantiation parameters
- Generation units predict images (patches) based on transformed parameters

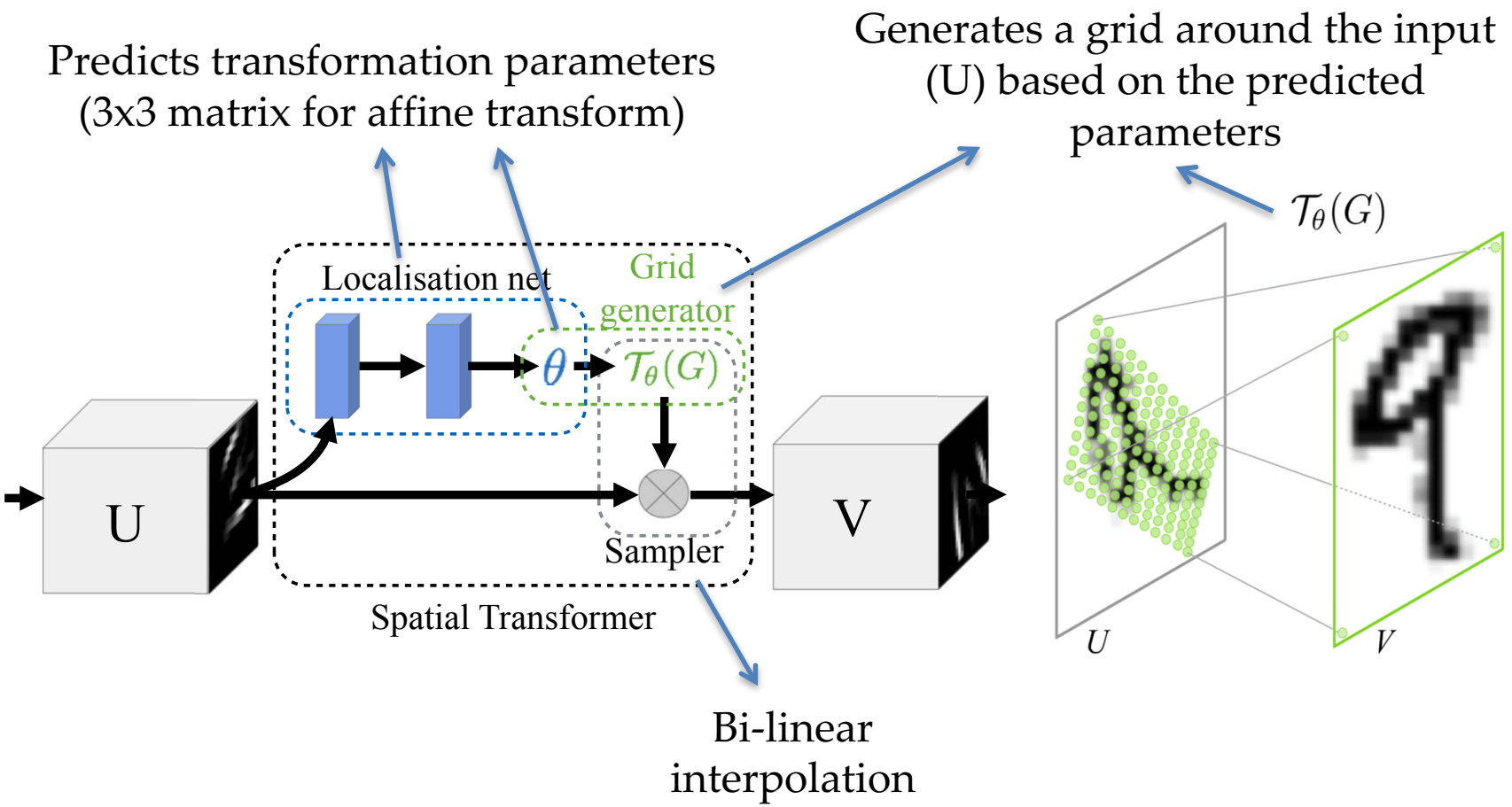
Spatial Transformer Nets
(Jaderberg et al., 2015)

- **Domain:** Classification, modeling dynamics etc.
- Idea: Conditional on input feature map, spatially warp image
  - Transforms data to a space expected by subsequent layers
  - Intelligently select features of interest (attention)
  - Invariant to more generic warping

Slide courtesy of Jaderberg et al.

Spatial Transformer Nets

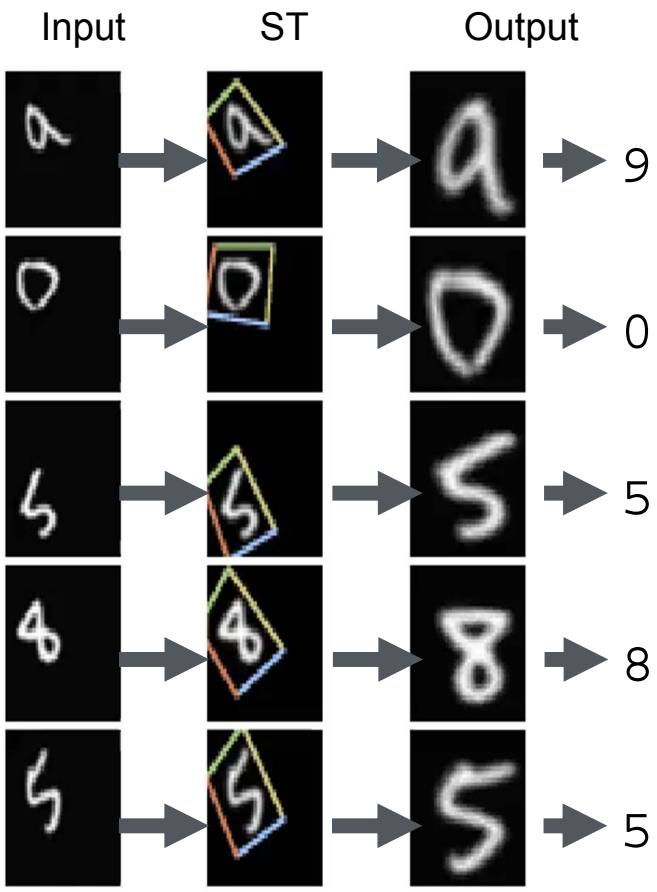
(Jaderberg et al., 2015)



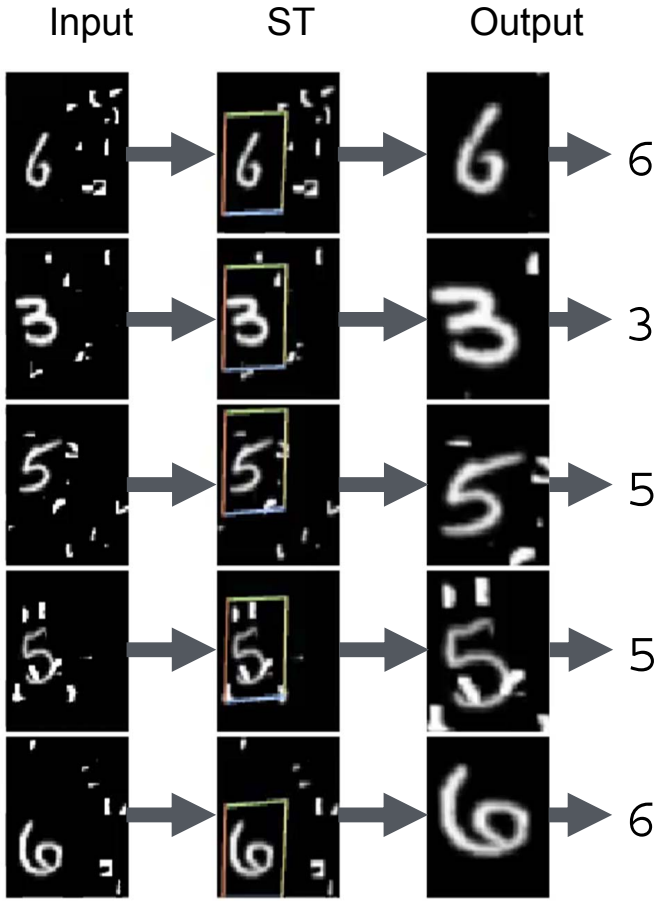
- End-to-End differentiable, no intermediated supervision
- Trained through loss based on the output
  - Ex: Output (V) can be used for MNIST digit classification

Spatial Transformer Nets

(Jaderberg et al., 2015)



Translated, Rotated, Scaled MNIST



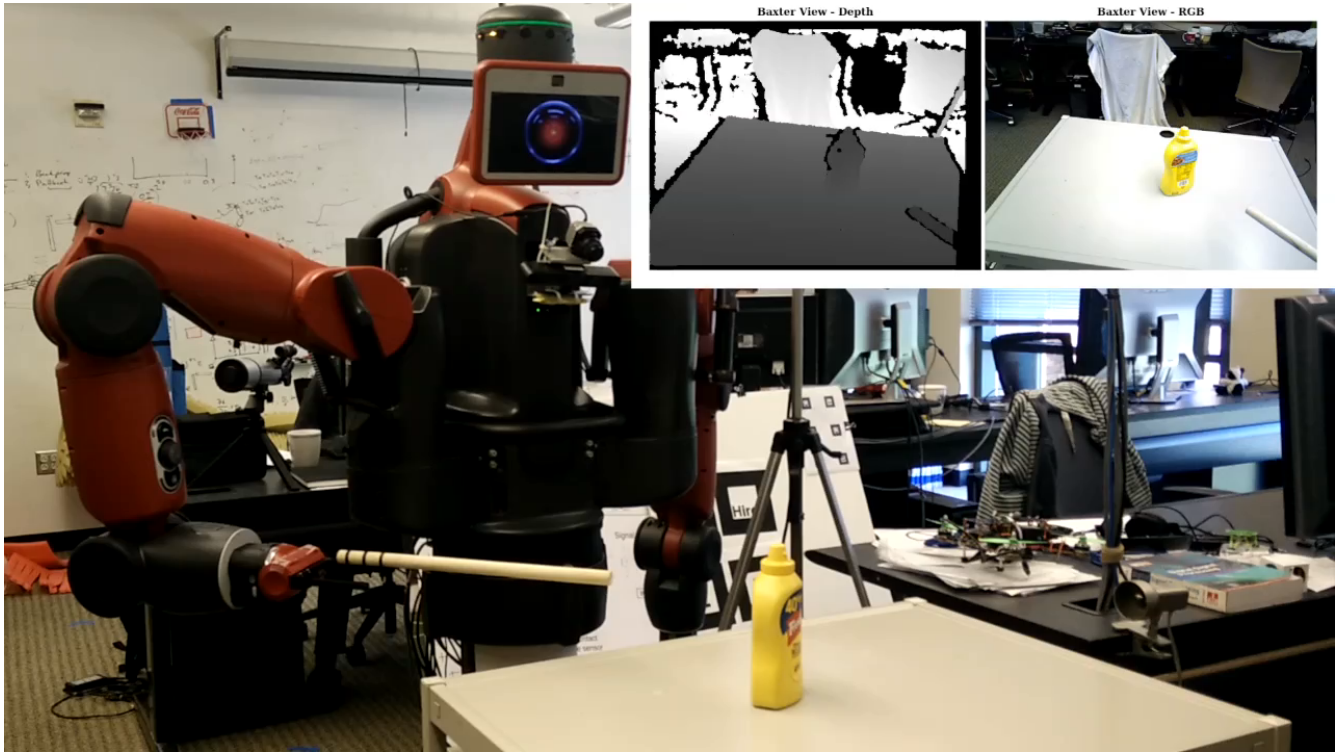
Translated, Cluttered MNIST

Structure

- Predicts affine transform parameters based on input/ features
- Affine transforms input/features for use in later tasks

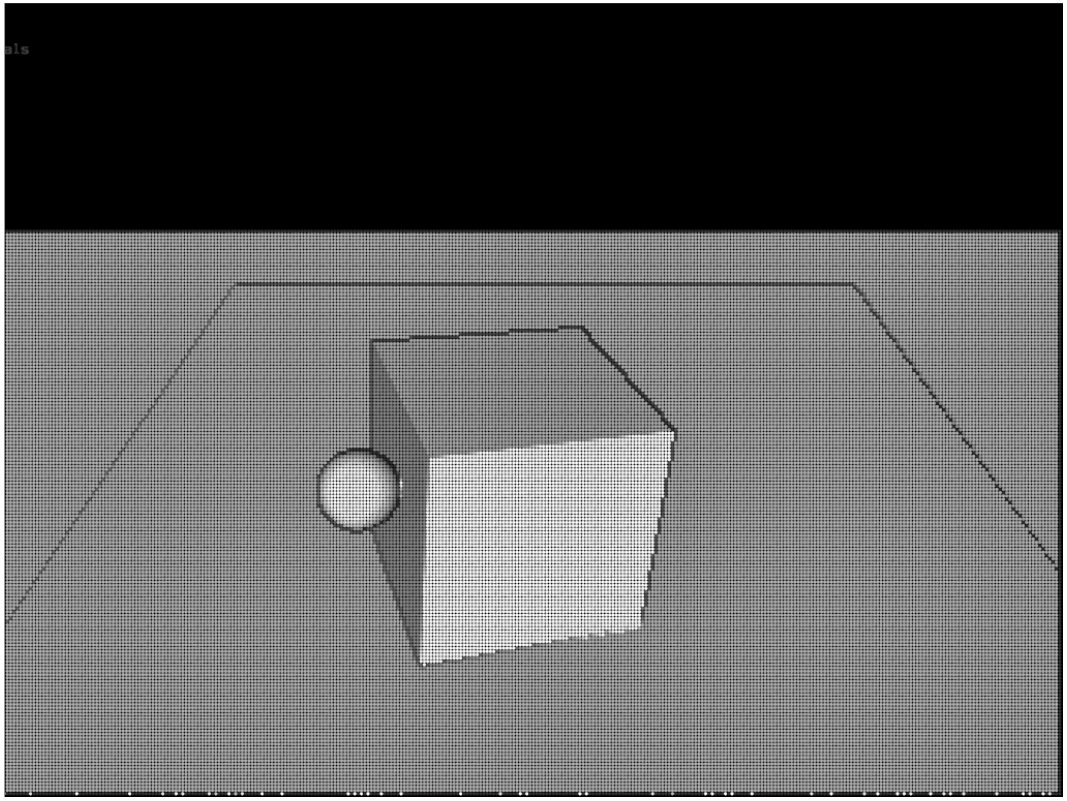
SE3-Nets
(Byravan et al., 2017)

- **Domain:** Modeling scene dynamics, intuitive physics
  - Predict how the scene changes as the robot interacts with objects



Problem definition

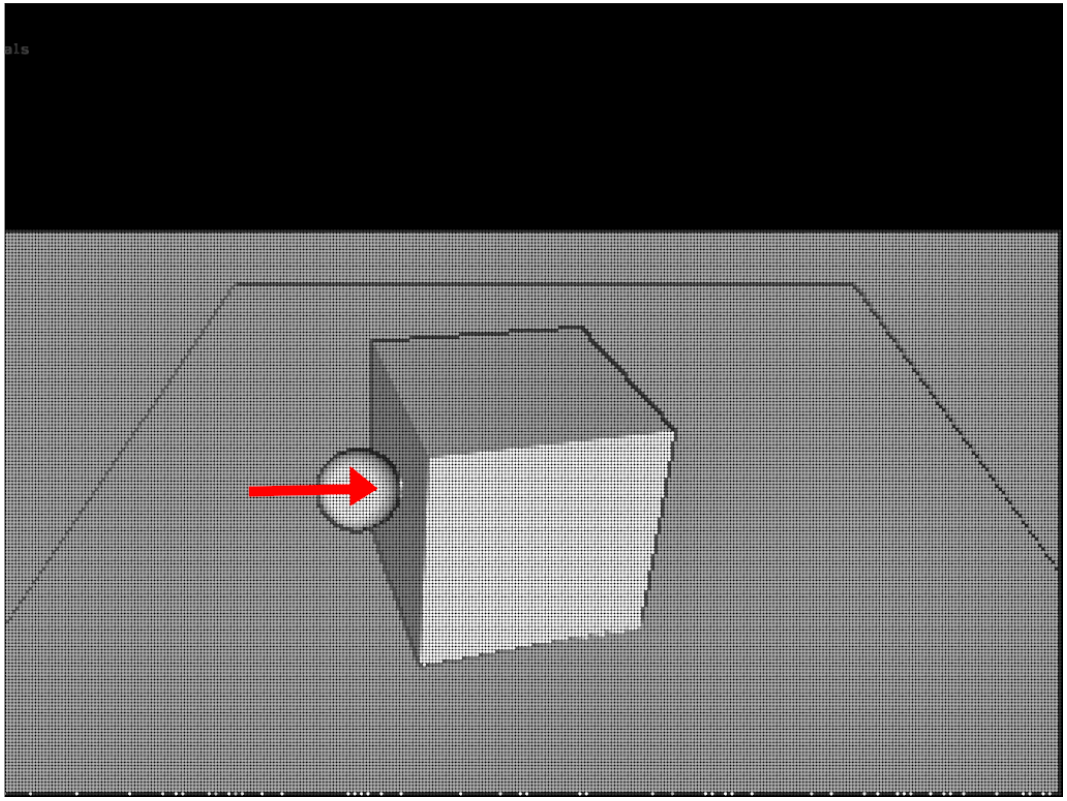
Scene
(Point cloud)



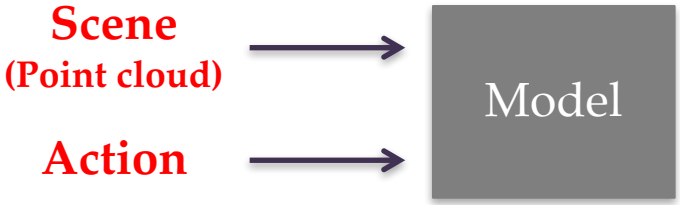
Problem definition

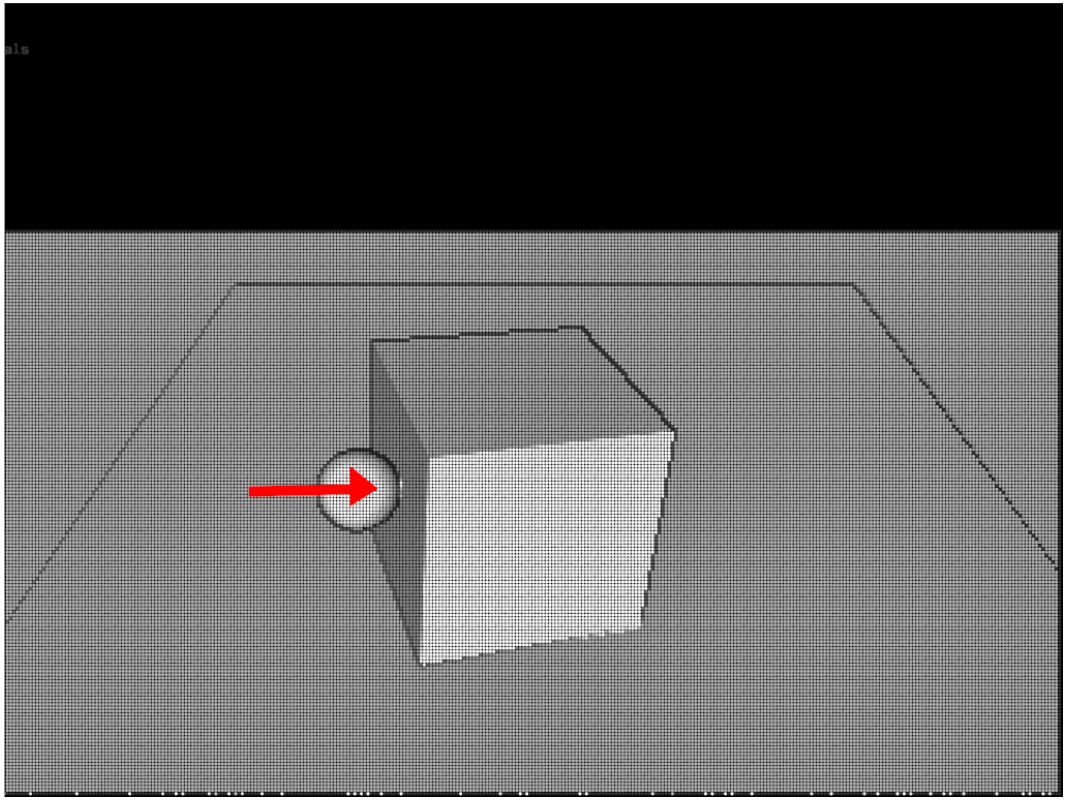
Scene
(Point cloud)

Action

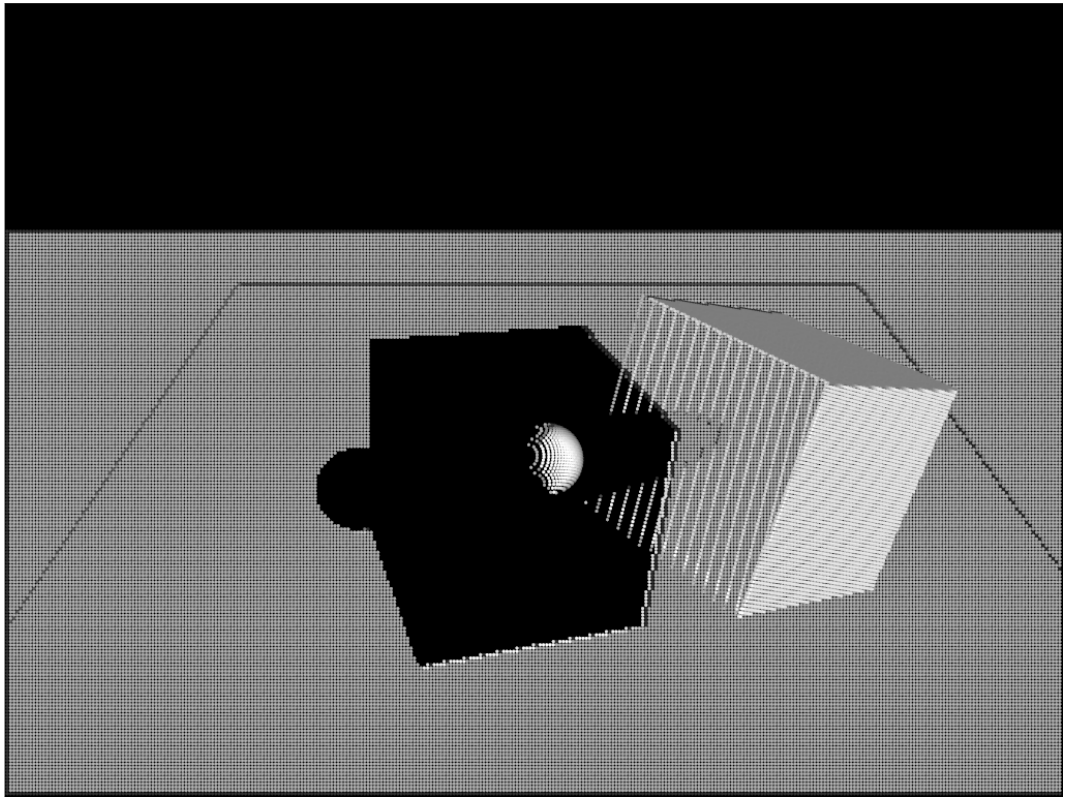


Problem definition

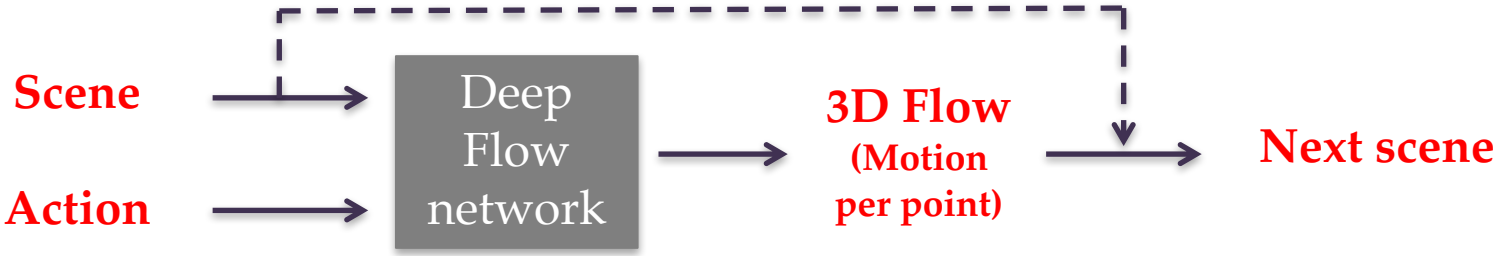


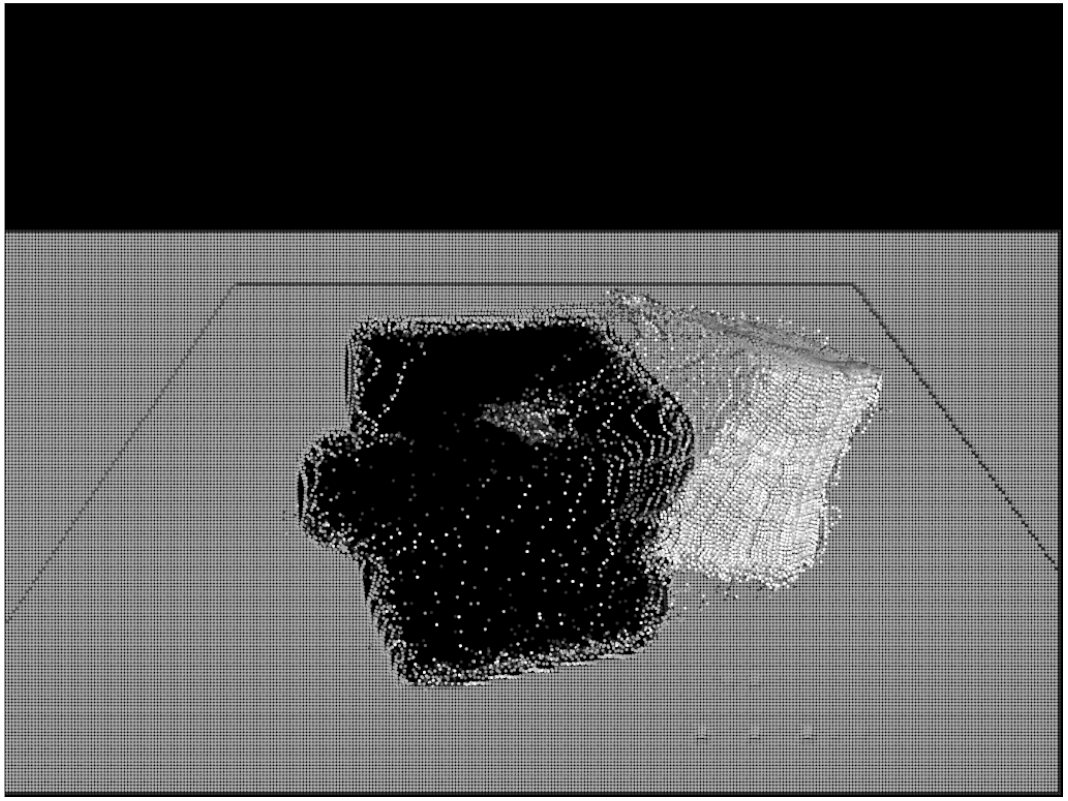


Problem definition



Baseline deep model





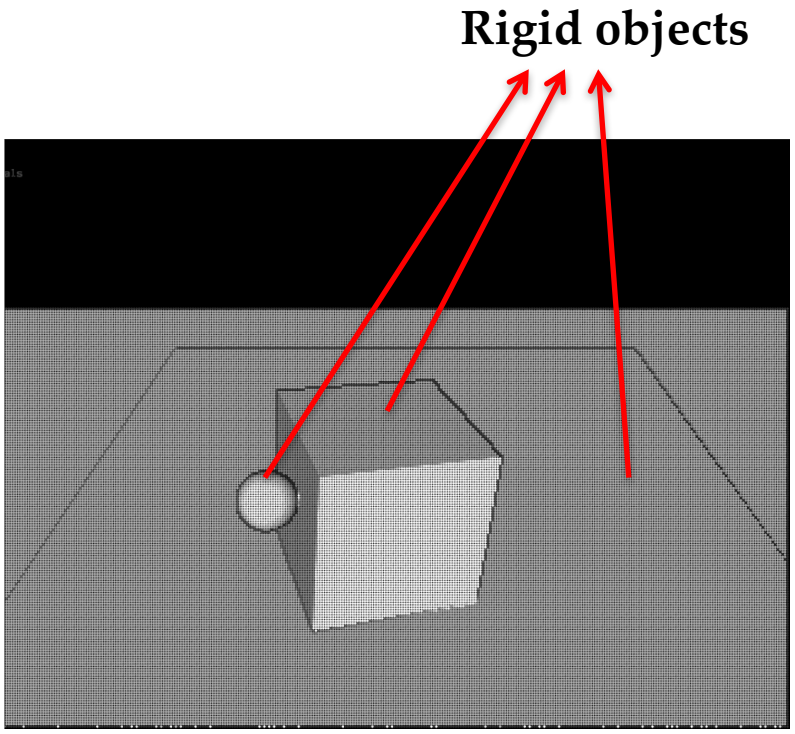
Noisy predictions. Objects distorted

Key Idea: Physical priors for learning

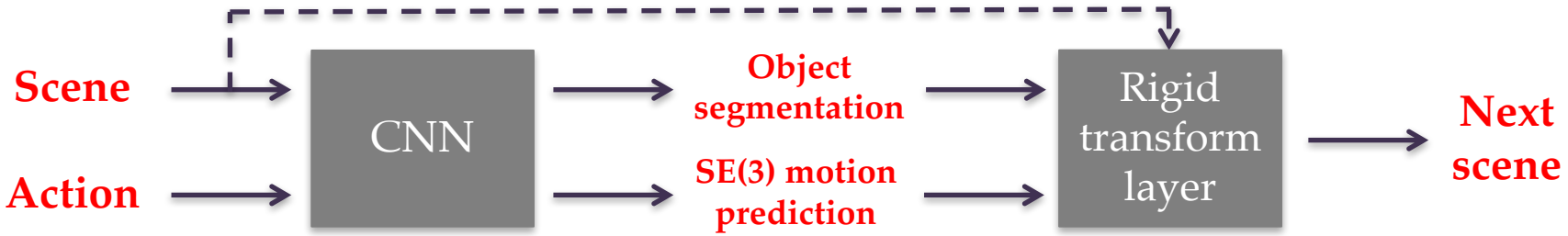
- Many real world interactions involve rigid objects

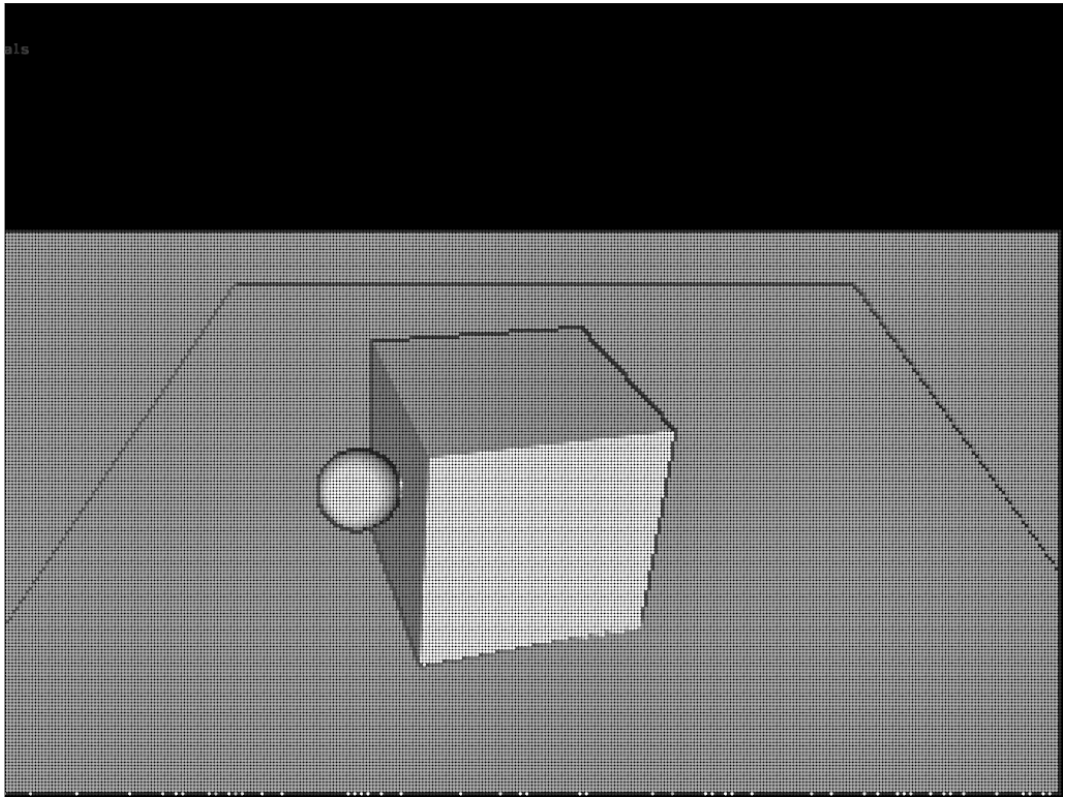
- Rigid body motion can be modeled by SE(3) transformations (Rotations, Translations in 3D)

- Use deep models to identify and predict motion of rigid objects: **SE3-Nets**

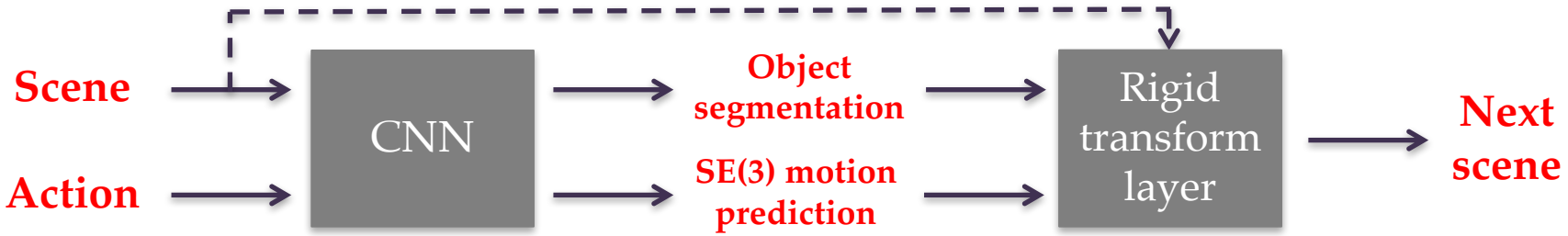


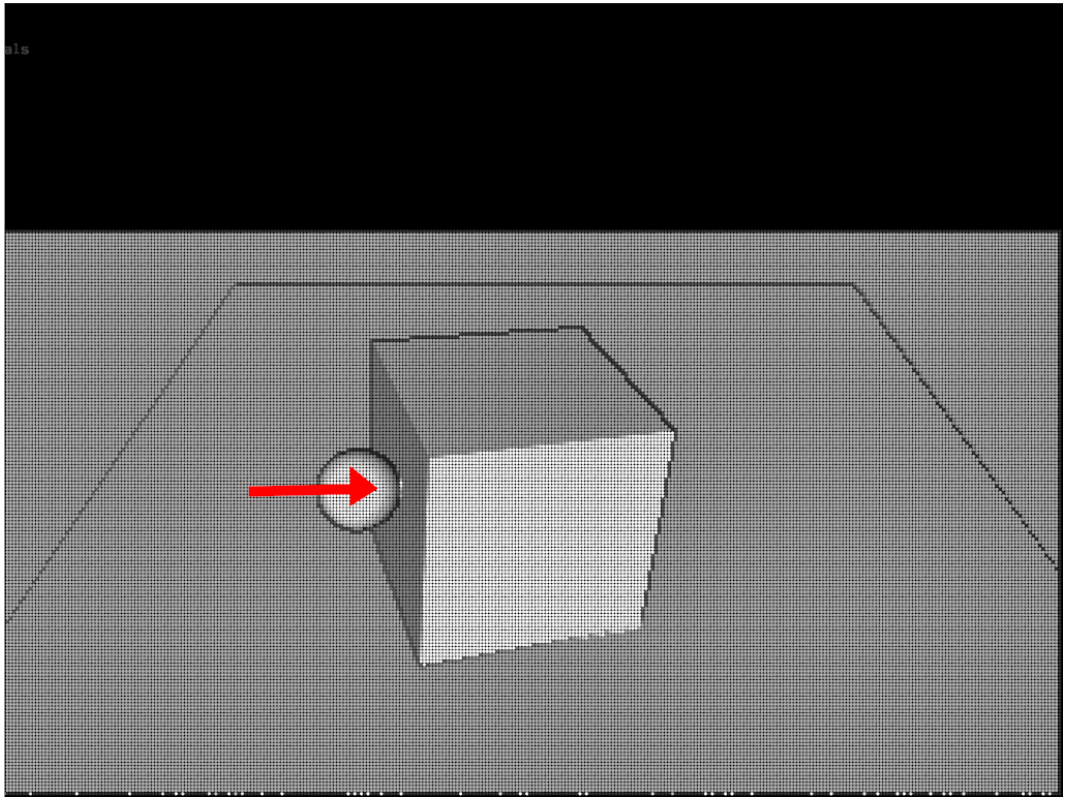
SE3-Net


Scene
Action
CNN
Object segmentation
SE(3) motion prediction
Rigid transform layer
Next scene

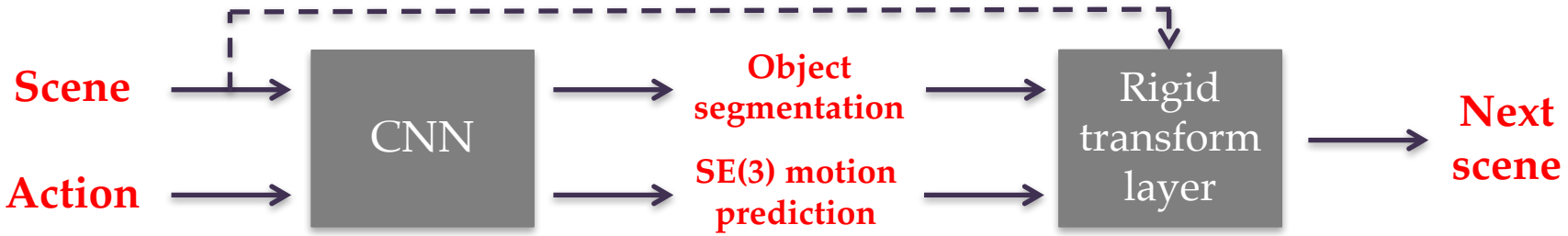


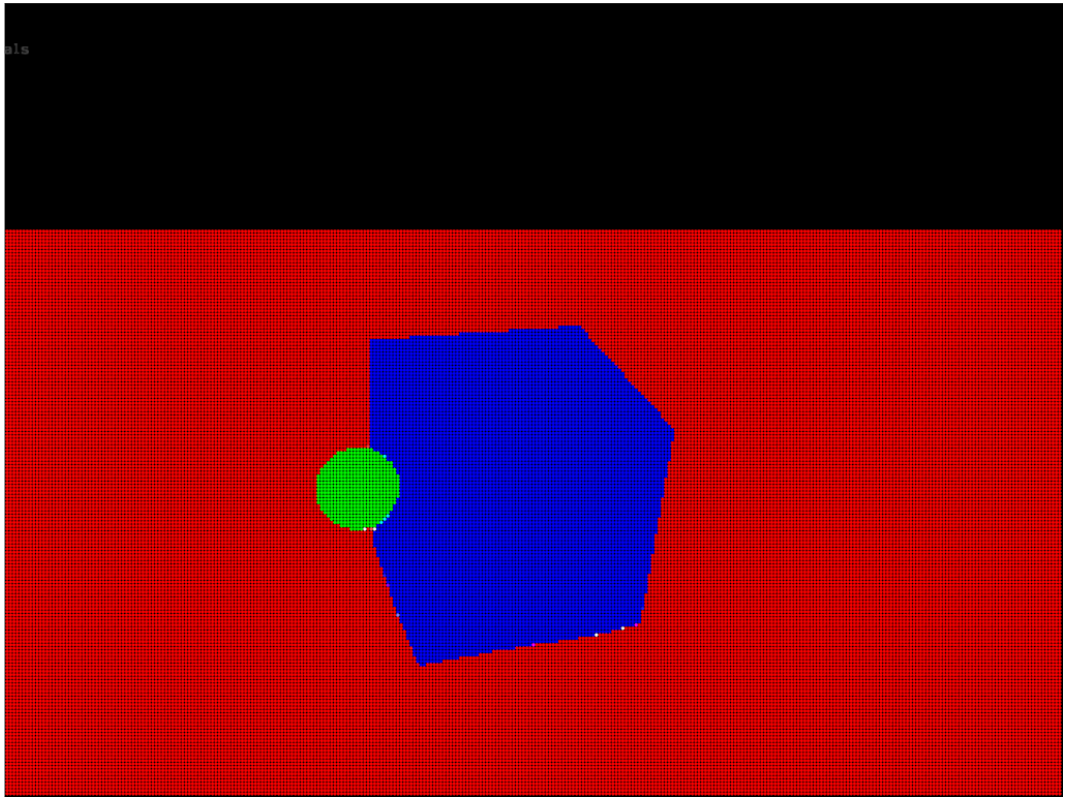
SE3-Net


Scene
Action
CNN
Object segmentation
SE(3) motion prediction
Rigid transform layer
Next scene

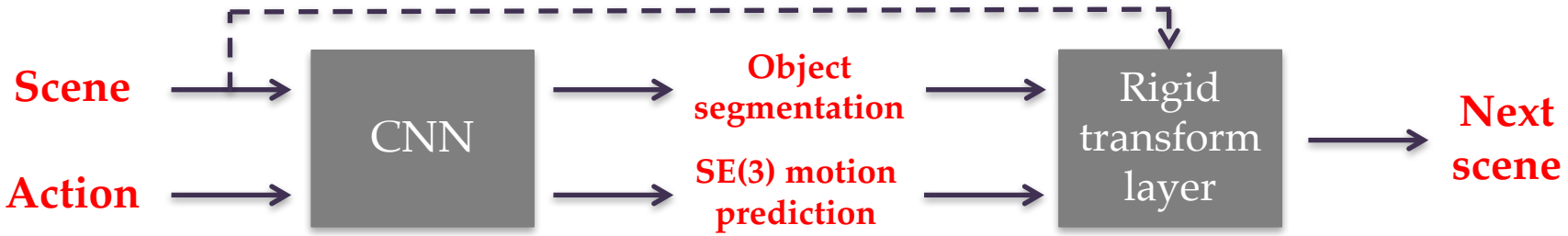


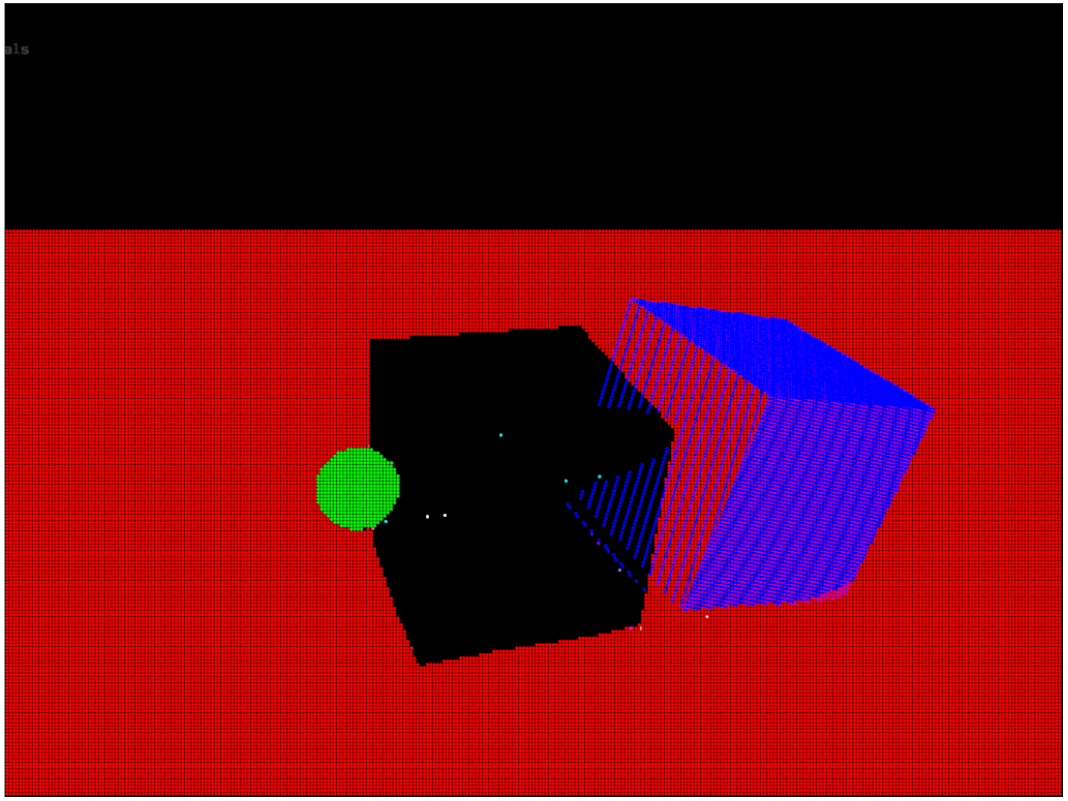
SE3-Net


Scene
Action
CNN
Object segmentation
SE(3) motion prediction
Rigid transform layer
Next scene

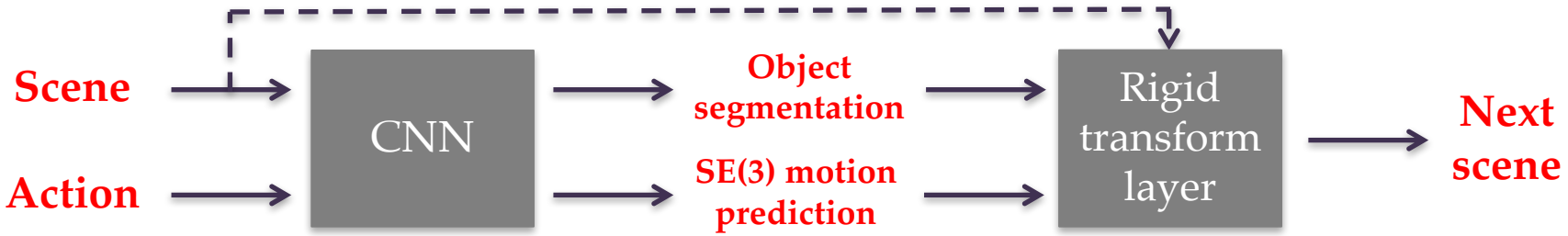


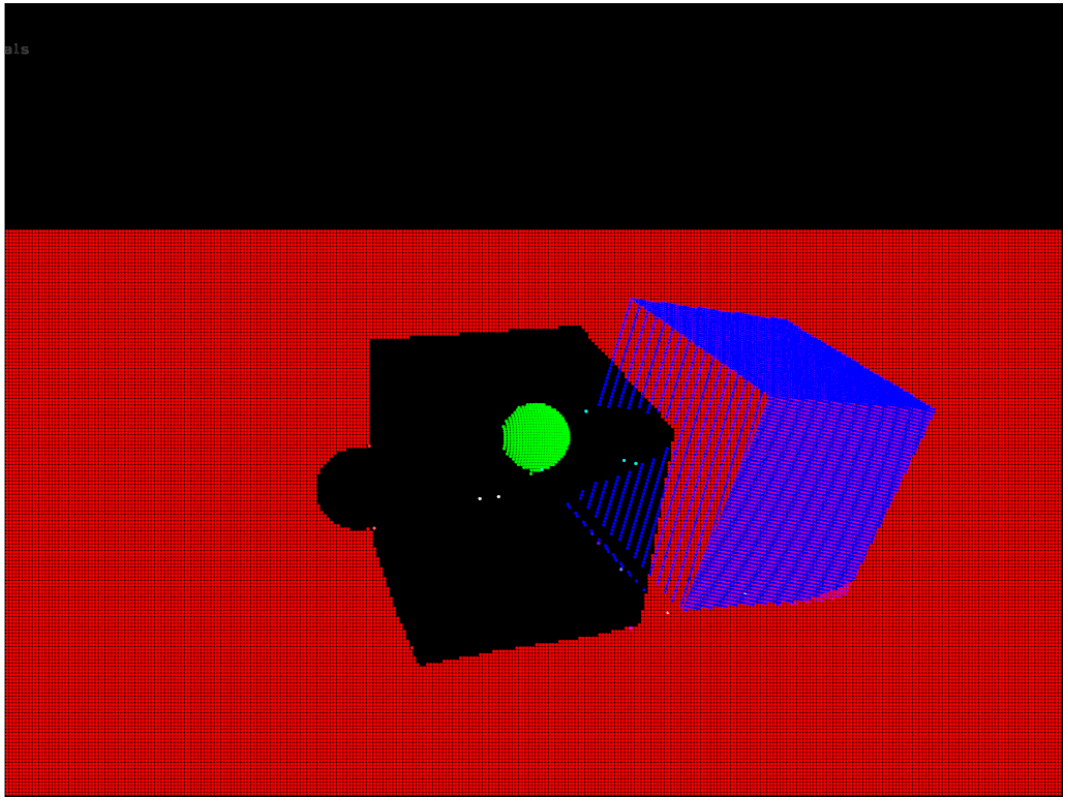
SE3-Net


Scene
Action
CNN
Object segmentation
SE(3) motion prediction
Rigid transform layer
Next scene

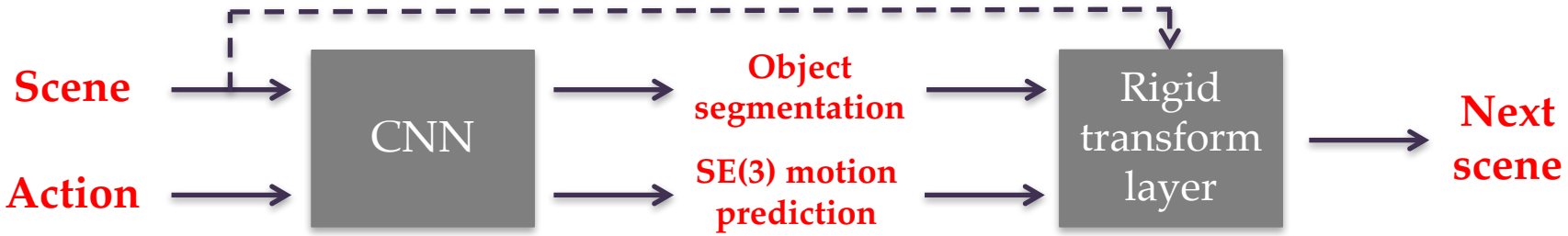


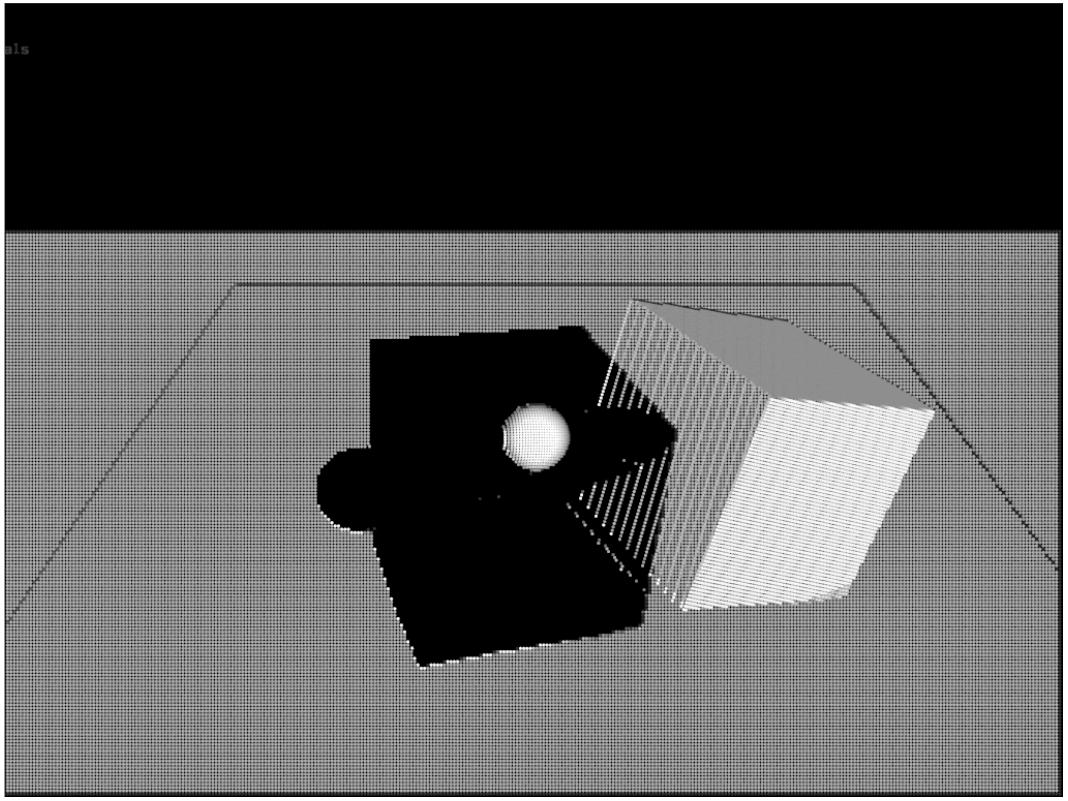
SE3-Net


Scene
Action
CNN
Object segmentation
SE(3) motion prediction
Rigid transform layer
Next scene



SE3-Net



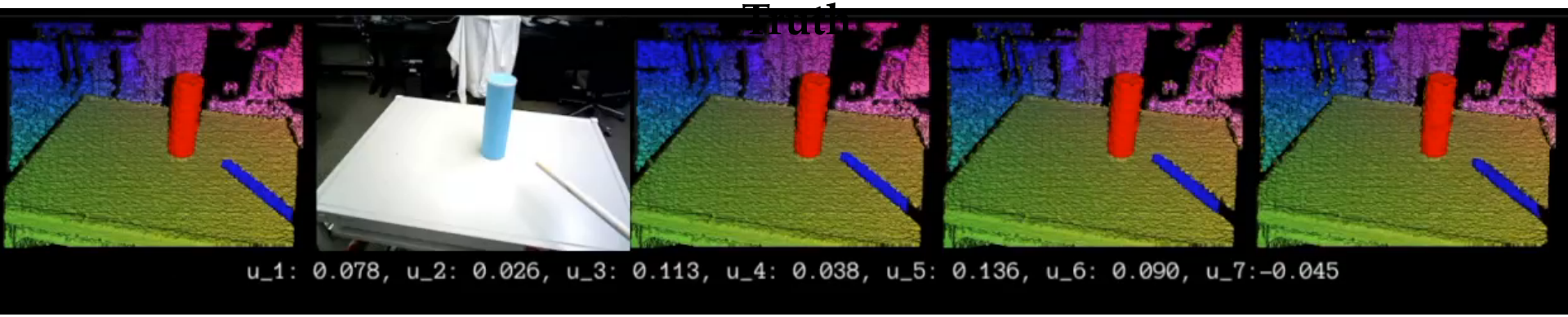


Sharp predictions. Object shapes preserved

SE3-Nets
(Byravan et al., 2017)

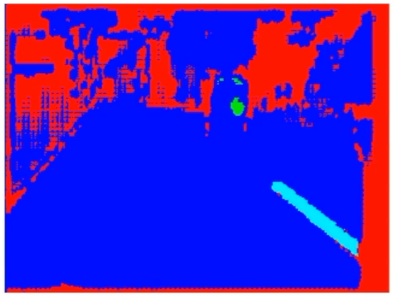
- Trained end-to-end
- No supervision for segmentation or pose prediction

Input Depth **RGB** **Ground Truth** **SE3-Net** **Baseline**



Structure

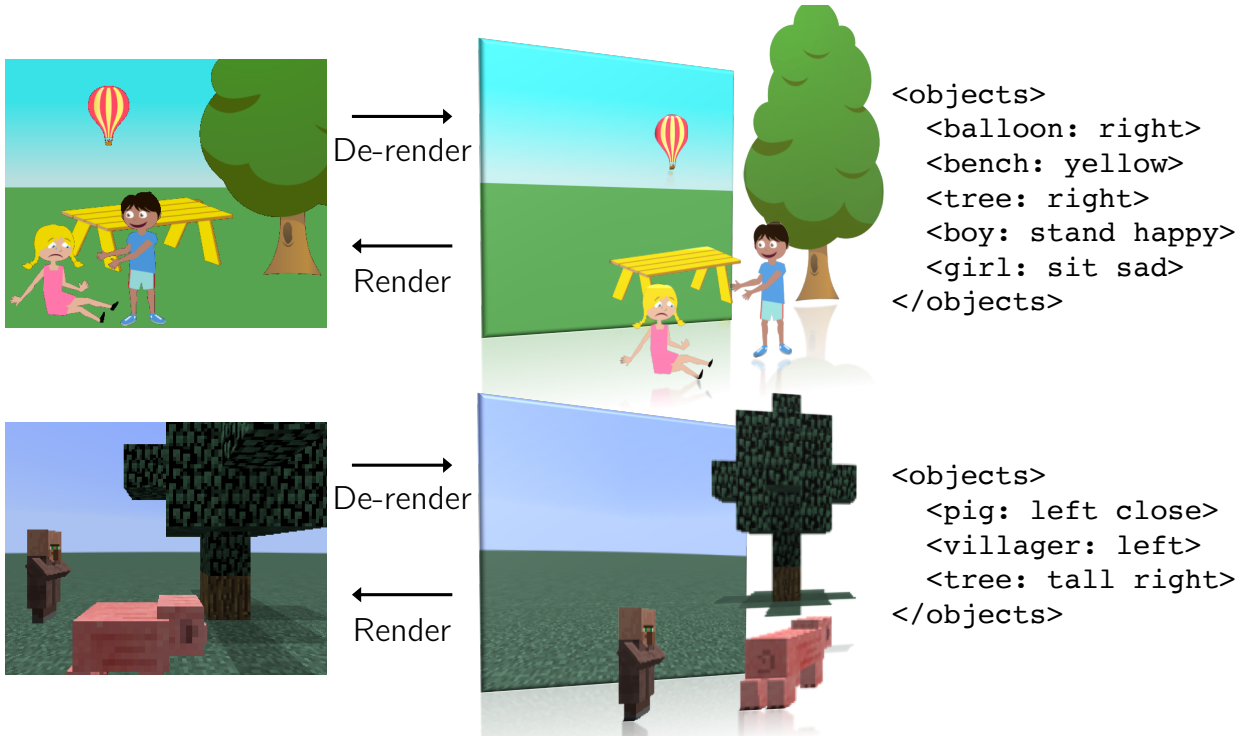
- Predicts dense object segmentation
- Predicts SE(3) transform per object
- Transforms each object by the SE(3) transform to generate final output



SE3-Net masks

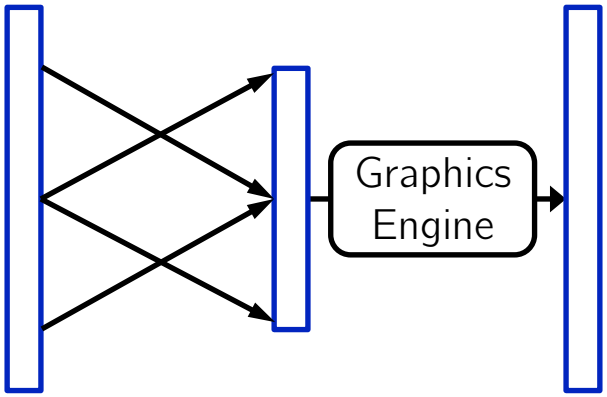
Neural Scene De-rendering
(Wu et al., 2017)

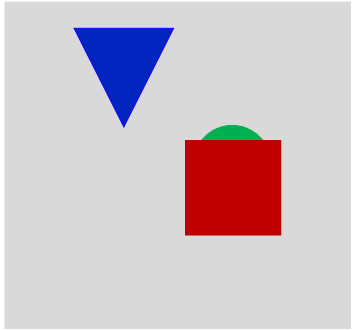
- **Domain:** Scene understanding
- **Goal:** A compact, interpretable scene representation



Neural Scene De-rendering
(Wu et al., 2017)

- Idea: Integrate a graphics engine with a deep network
  - Allow symbolic representation (Scene XML)
  - Generalize to varying number of objects
- Renderers are non-differentiable:
  - Use REINFORCE (sampling + rollouts) to compute gradients
  - Alternative: Finite differencing



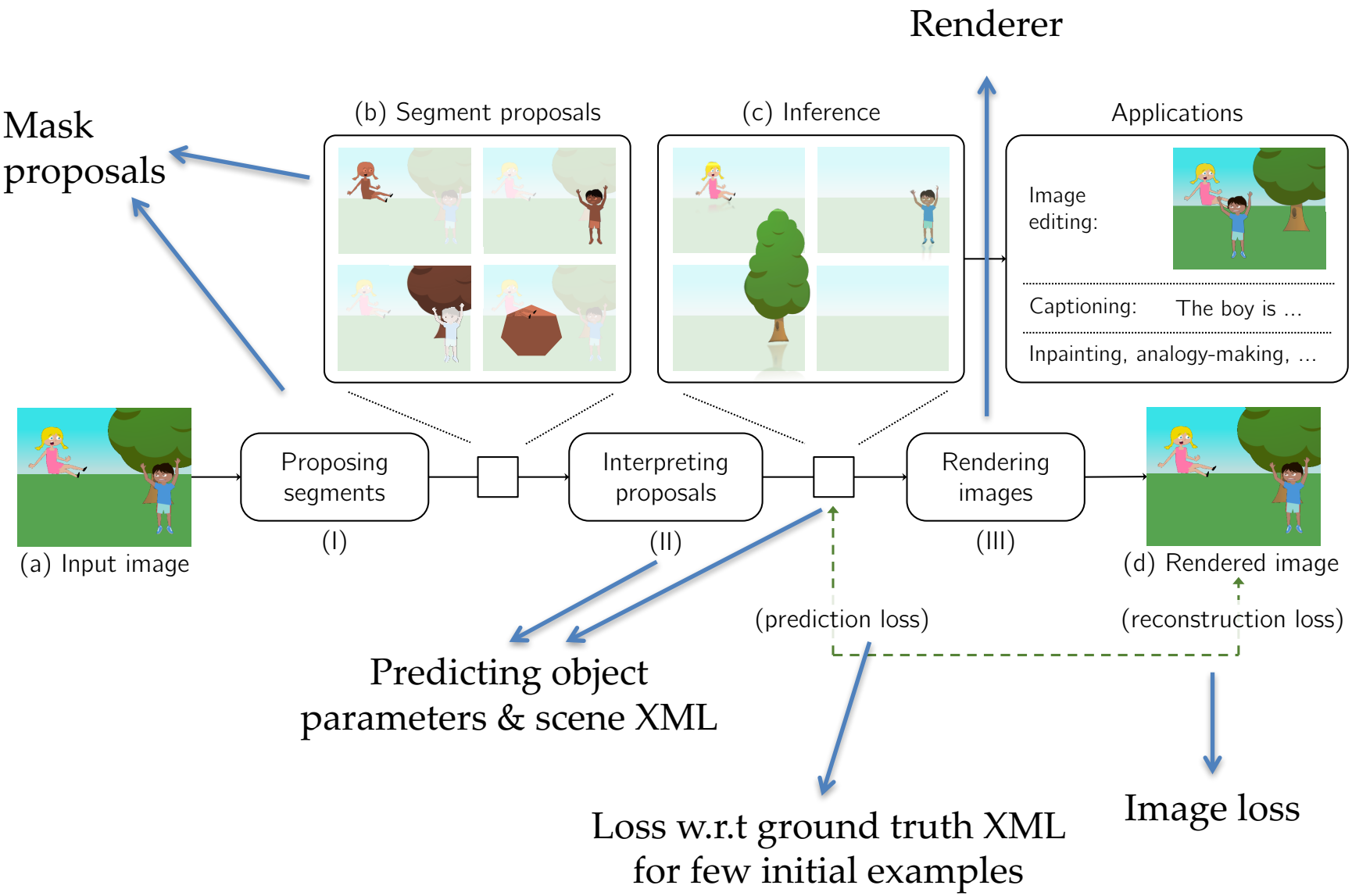


Scene XML

```
<object>
    <category>triangle</category>
    <size>1.5</size>
    <color>blue</color>
    <position>1.5,2,1</position>
    <yaw>0</yaw>
    …
</object>
<object>
…
```

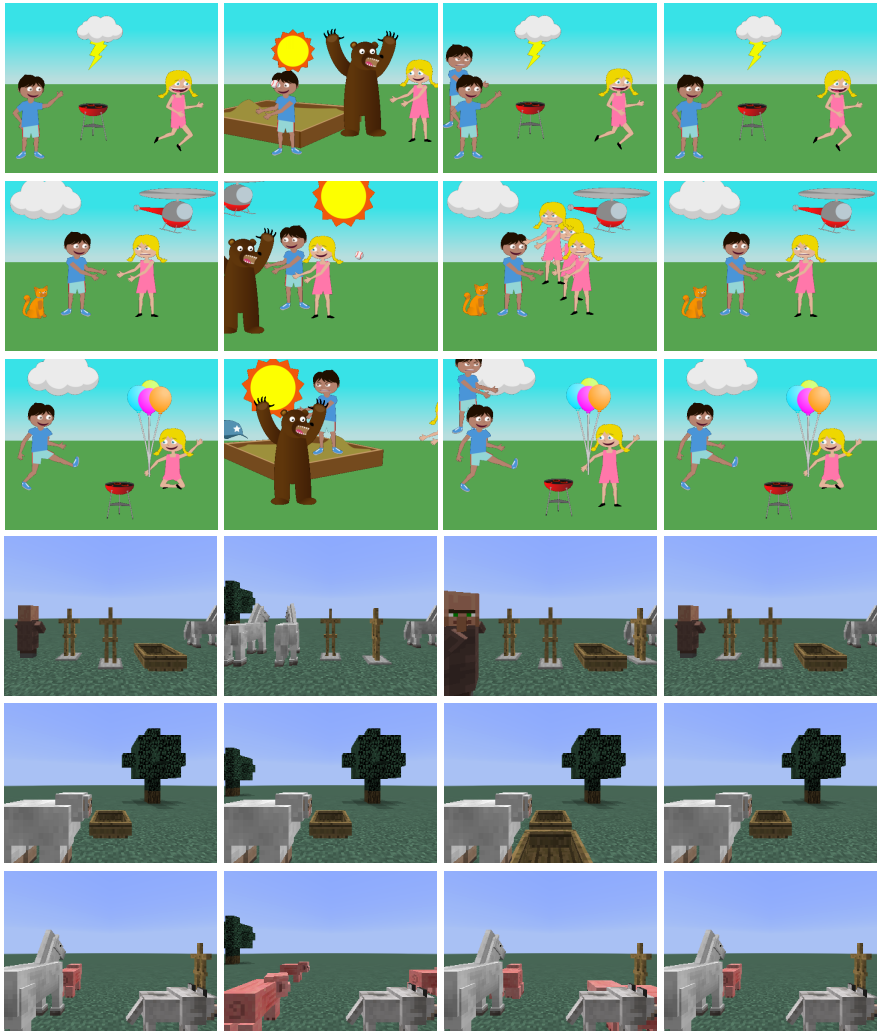
Neural Scene De-rendering

(Wu et al., 2017)



Neural Scene De-rendering

(Wu et al., 2017)



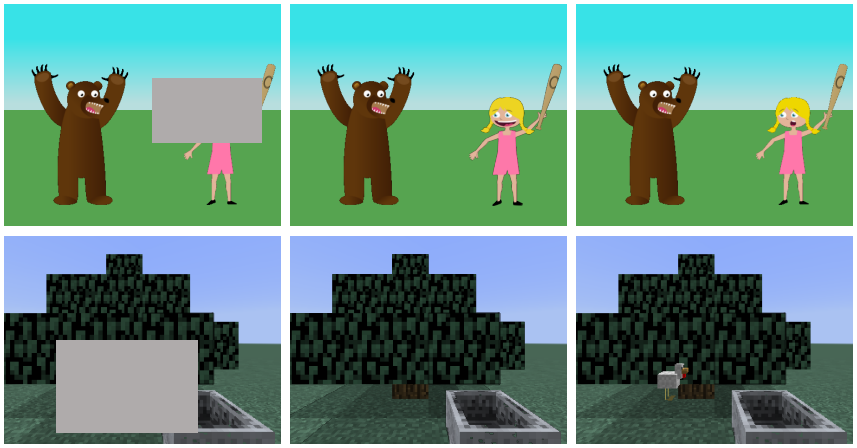
(a) Input image (b) CNN+LSTM (c) NSD (d) NSD (full)

Reconstruction results

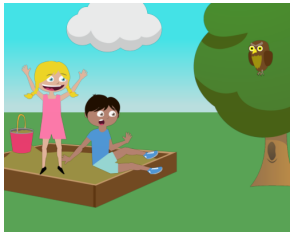
Neural Scene De-rendering
(Wu et al., 2017)

Applications

Inpainting



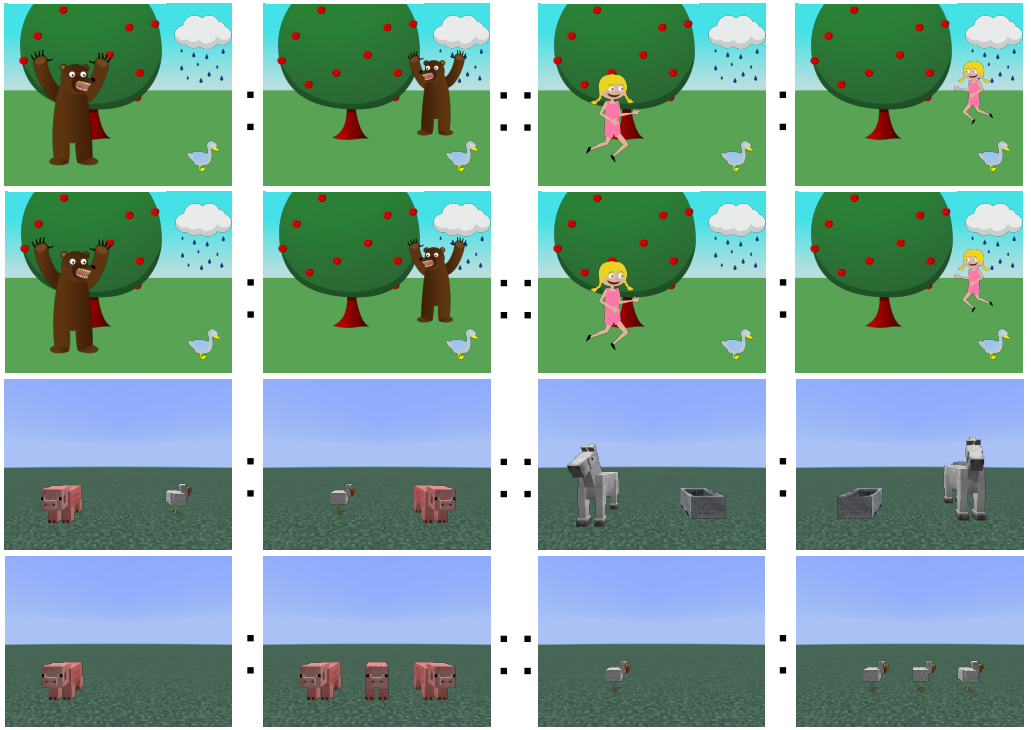
(a) Corrupted input (b) Reconstruction (c) Original image



Caption Retrieval

jenny and mike are having fun in the sandbox unaware of the storm that's coming their way

Analogy-Making



Reference A Reference A' Query B Prediction B'

Structure

- Explicitly learn a disentangled, object based representation
- Use a graphics engine for rendering based on learned XML representation

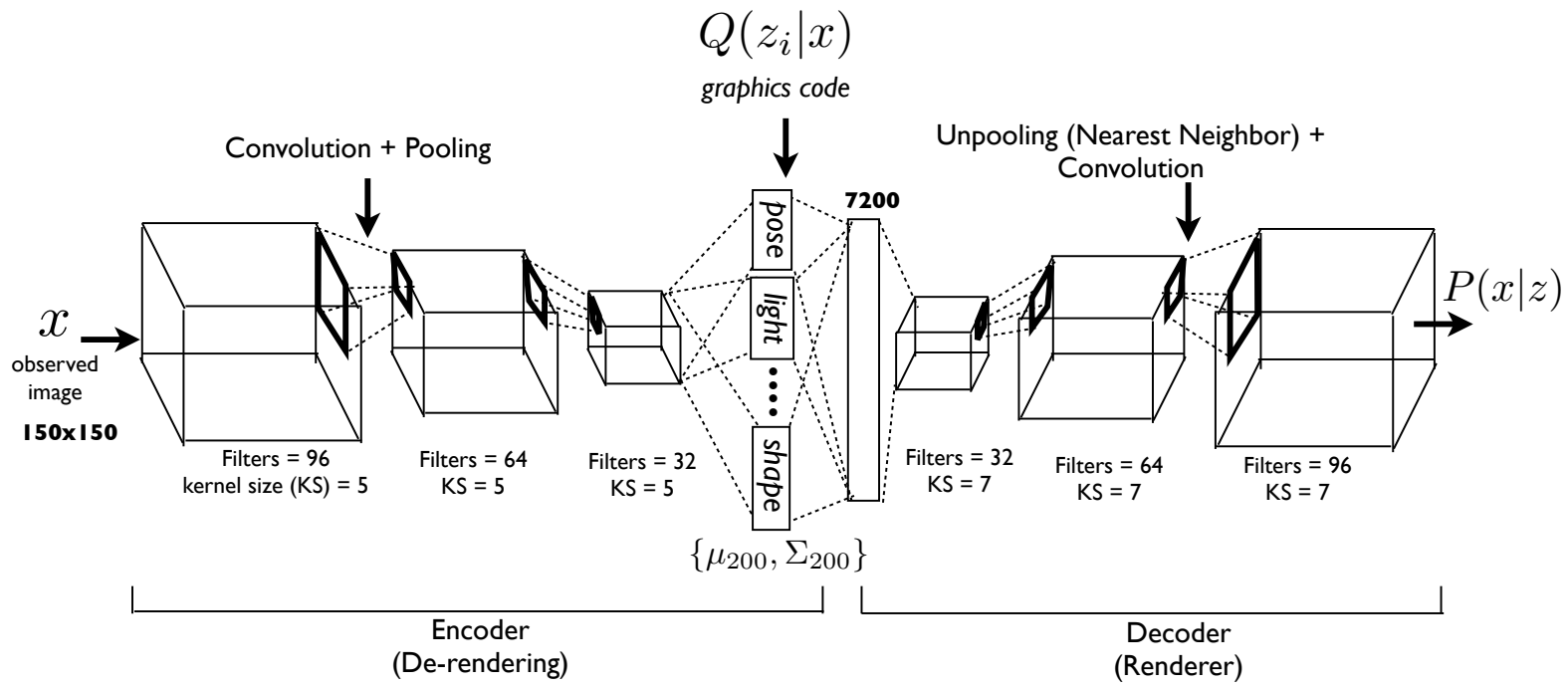
Structure in the training

Inverse Graphics Network

(Kulkarni et al., 2015)

- Domain: Scene understanding
- Idea: Learn disentangled representations of scenes
  - Bottleneck layer in network models factors of variation in data (pose, lighting, viewpoint etc)
  - Provide training batches with factors of variation on & off to train representations

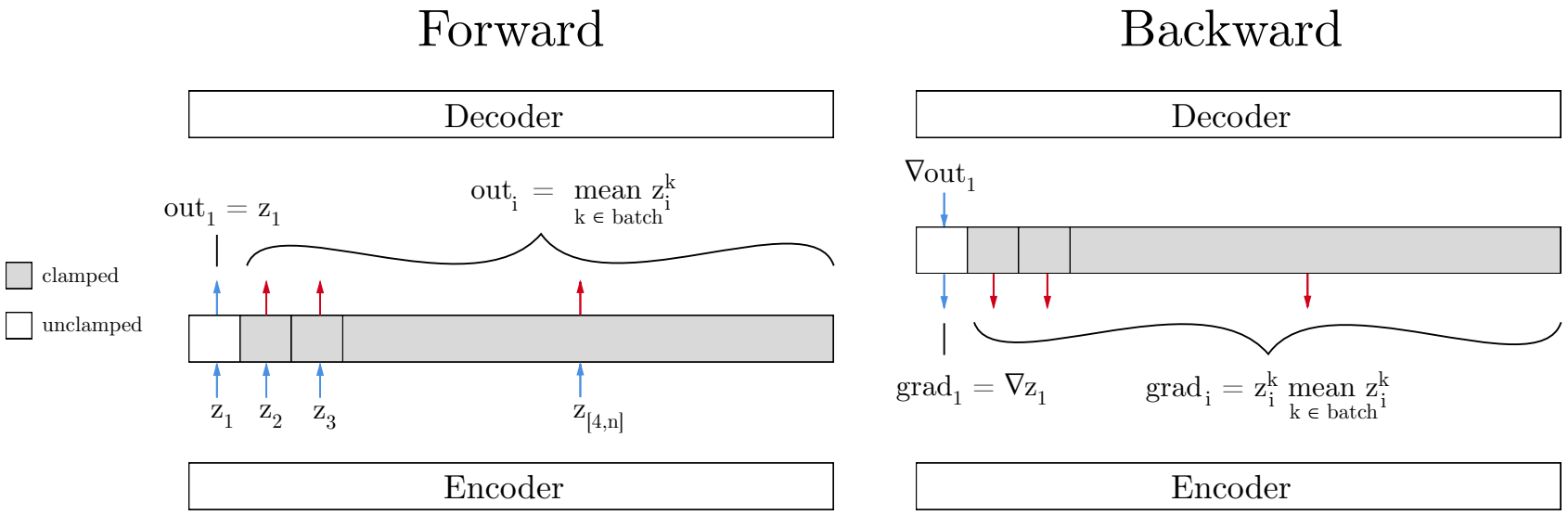
Inverse Graphics Network
(Kulkarni et al., 2015)



$$Z = \boxed{\mathbf{z_1} \mid \mathbf{z_2} \mid \mathbf{z_3} \mid \mathbf{z_{[4,n]}}}$$

corresponds to ϕ α ϕ_L intrinsic properties (shape, texture, etc)

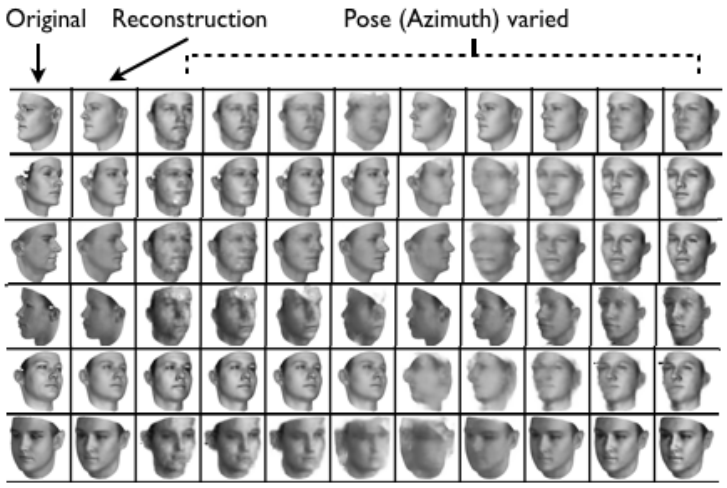
Inverse Graphics Network
(Kulkarni et al., 2015)

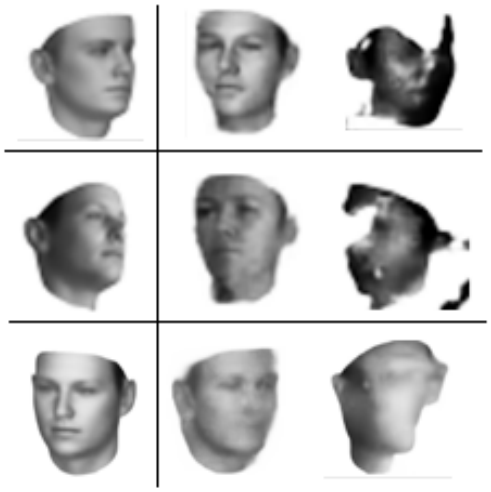


Idea

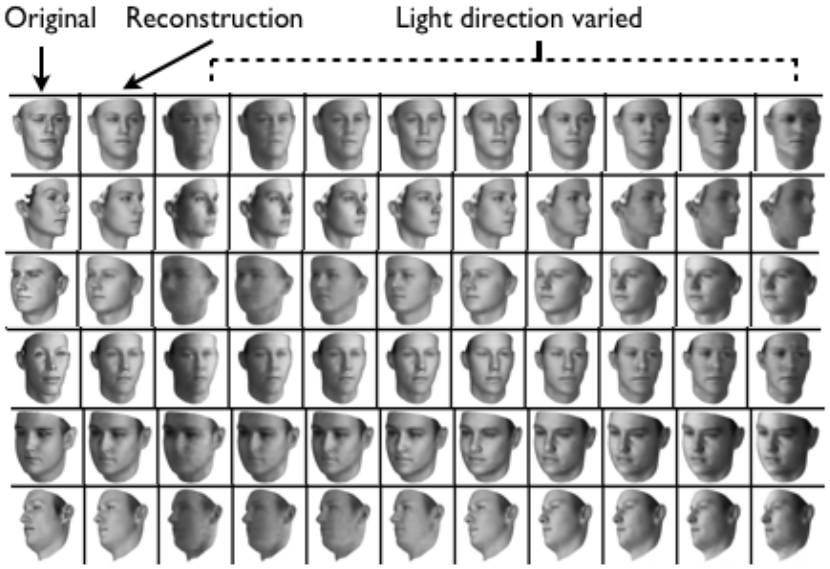
- Choose batch with specific factor of variation (say x-position of object)
- Run forward pass through the encoder
- Fix all outputs in the encoded vector to the mean activations except one
- This variable now has to capture all variations in x-position
- Run forward pass through the decoder based on this encoding and backprop

Inverse Graphics Network
(Kulkarni et al., 2015)

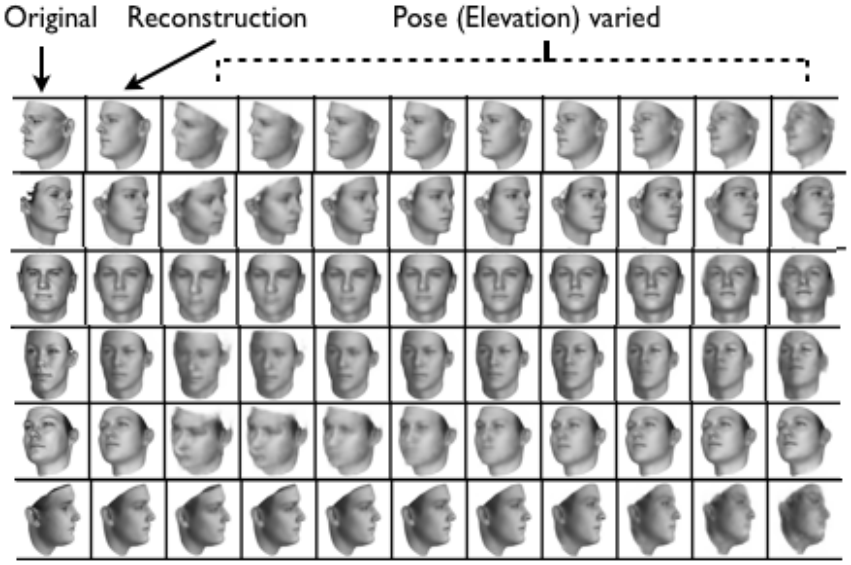




Comparison with baseline (far right)



(a)



(b)

Summary

- Learn meaningful representations by adding structure into deep networks:
  - Explicitly tying intermediate layers to physical parameters
  - Specific operations on data
  - Training procedure
- How to choose structure?
  - Problem specific
  - Bias-Variance tradeoff

References

- Hinton, Geoffrey E., Alex Krizhevsky, and Sida D. Wang. "**Transforming auto-encoders.**" International Conference on Artificial Neural Networks. Springer Berlin Heidelberg, 2011
- Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "**Spatial transformer networks.**" Advances in Neural Information Processing Systems. 2015.
- Byravan, Arunkumar, and Dieter Fox. "**Se3-nets: Learning rigid body motion using deep neural networks.**" Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017.
- Wu, Jiajun., et al. "**Neural scene de-rendering**." Computer Vision and Pattern Recognition. 2017.
- Kulkarni, Tejas D., et al. "**Deep convolutional inverse graphics network.**" Advances in Neural Information Processing Systems. 2015.