3D model reconstruction from single & multiple images

Paper: 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction

Tonmoy Saikia
saikiat@cs.uni-freiburg.de

3D reconstruction

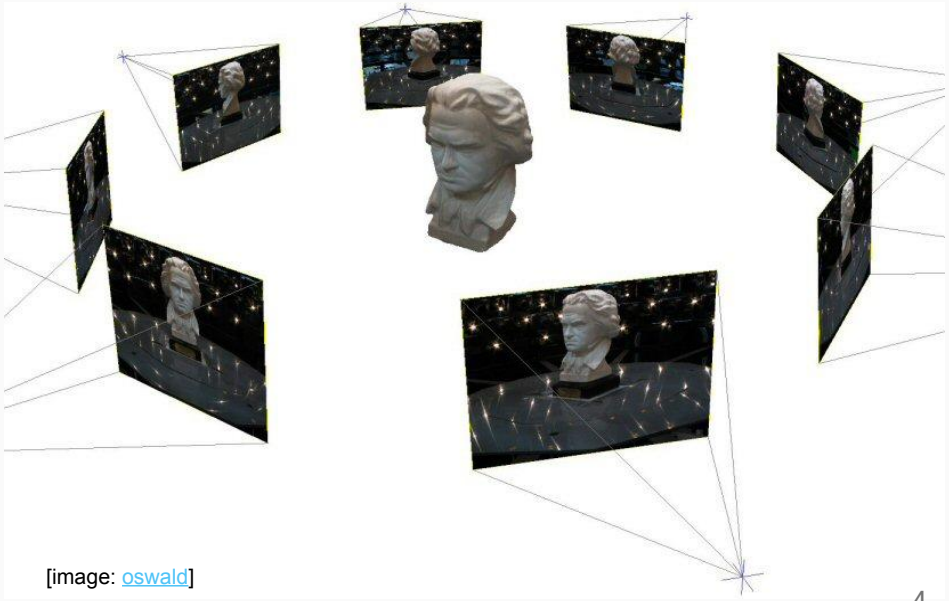
Outline

- The 3D reconstruction problem
  - Methods, Issues and challenges
- Deep learning for 3D reconstruction
  - Background: deep neural nets
    - CNN
    - RNN (LSTM)
  - Architecture
  - Results

Multi view 3D reconstruction

Estimate a 3D shape given a set of images

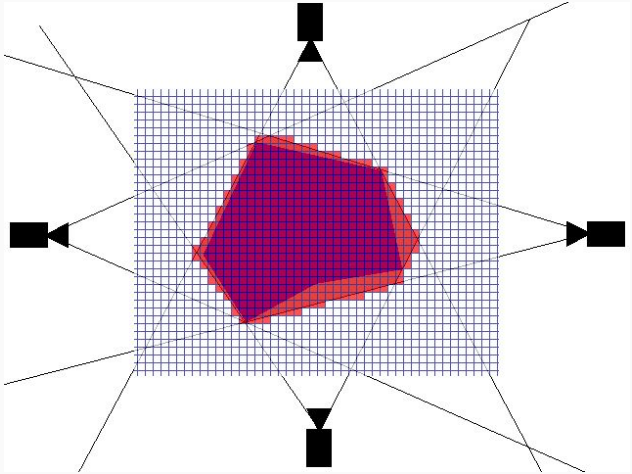
- Stereo reconstruction
  - Obtain point correspondences
  - Camera calibration
    - Estimate int and ext params
  - Calculate projection rays
    - Intersection gives 3D point
- Issues
  - Large baseline: point correspondence becomes hard
  - Occlusions
  - Error in finding point correspondences



[image: oswald]

Volumetric approaches

- Idea: Find a shape consistent with images
  - Shape: voxel grid
  - For each voxel -> compute occupied or free
- Shape from Silhouette
  - Extract silhouette images from different views
  - Project voxel on each image
    - Lies within silhouette - occupied
    - Otherwise - free
- Other methods: voxel coloring, shape carving
- But requires images to segmented, cameras calibrated

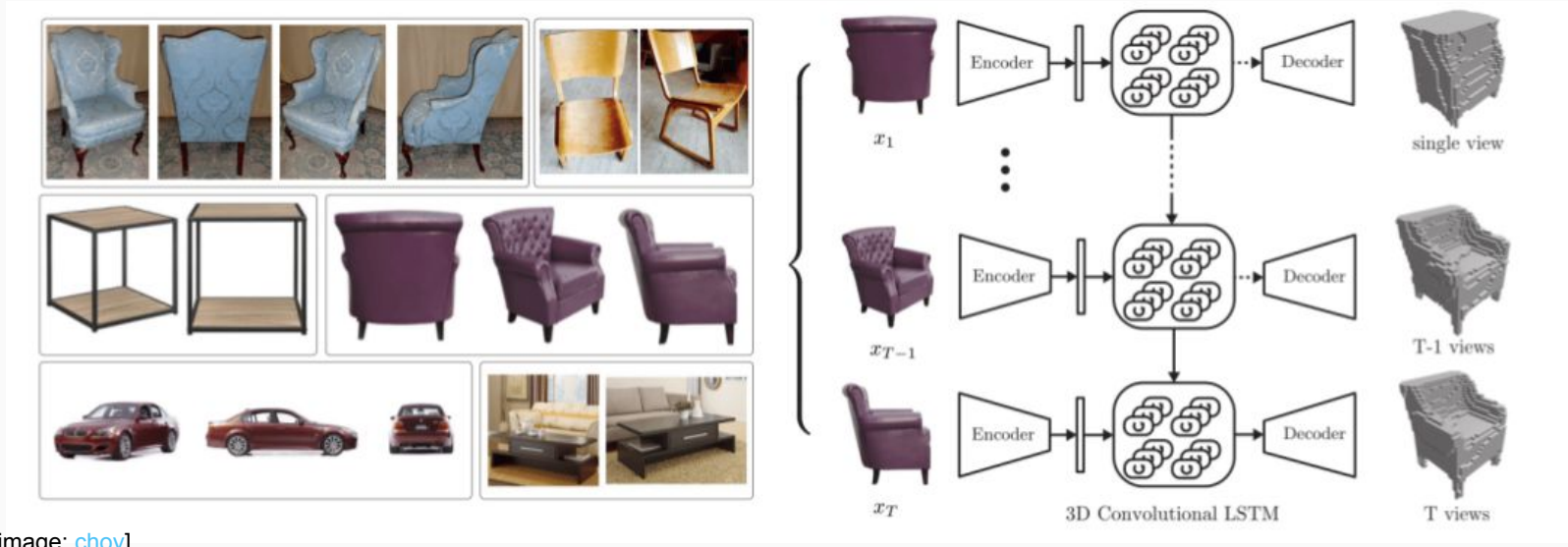


Using Shape Priors

- Enables estimating 3D shape with fewer viewpoints
  - Extreme case: a single view point
  - Earliest work: shape from shading
  - Assumes lambertian reflectance and constant albedo
- Includes prior knowledge on
  - reflectance, shape, viewpoints, illumination
  - learnt from data
- Less reliant on finding accurate feature correspondences

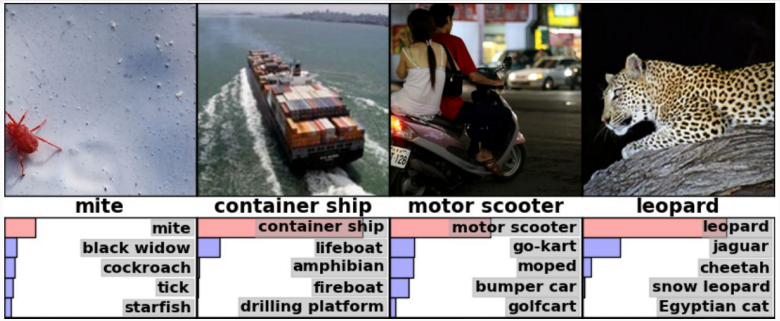
Deep NN for 3D reconstruction (3DR2N2)

Idea: Learn a mapping from observations to their underlying 3D shape



[image: choy]

Deep learning



- Algorithms using Neural Networks as an architecture
- Learns features automatically

Image Classification

Krizhevsky et. al, NIPS 12

github.com/kjw0612/awesome-deep-vision

Image captioning

Karpathy et al, CVPR 15

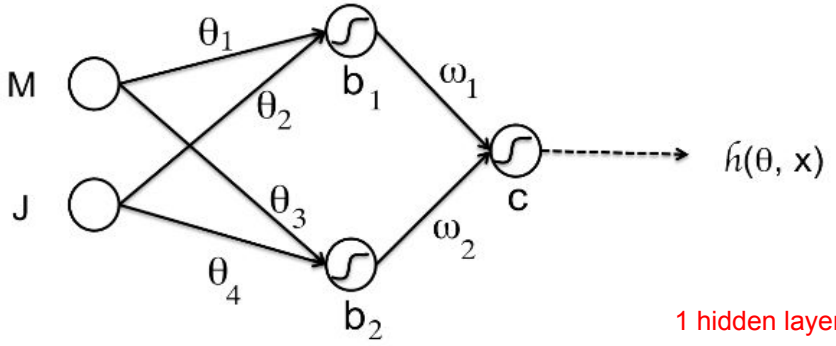

man in black shirt is playing guitar.


construction worker in orange safety vest is working on road.

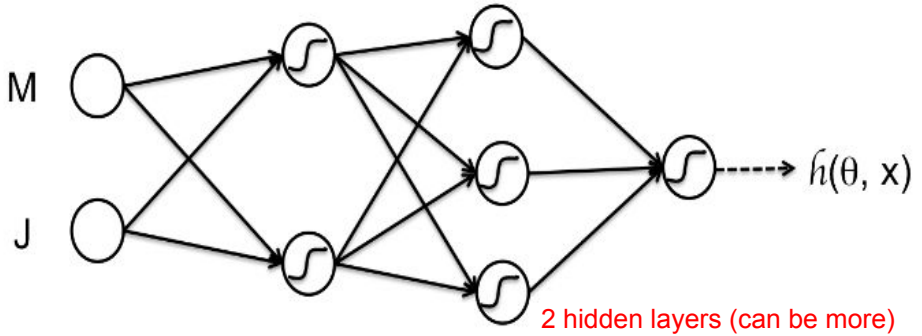

two young girls are playing with lego toy.


boy is doing backflip on wakeboard.

Neural networks



1 hidden layer

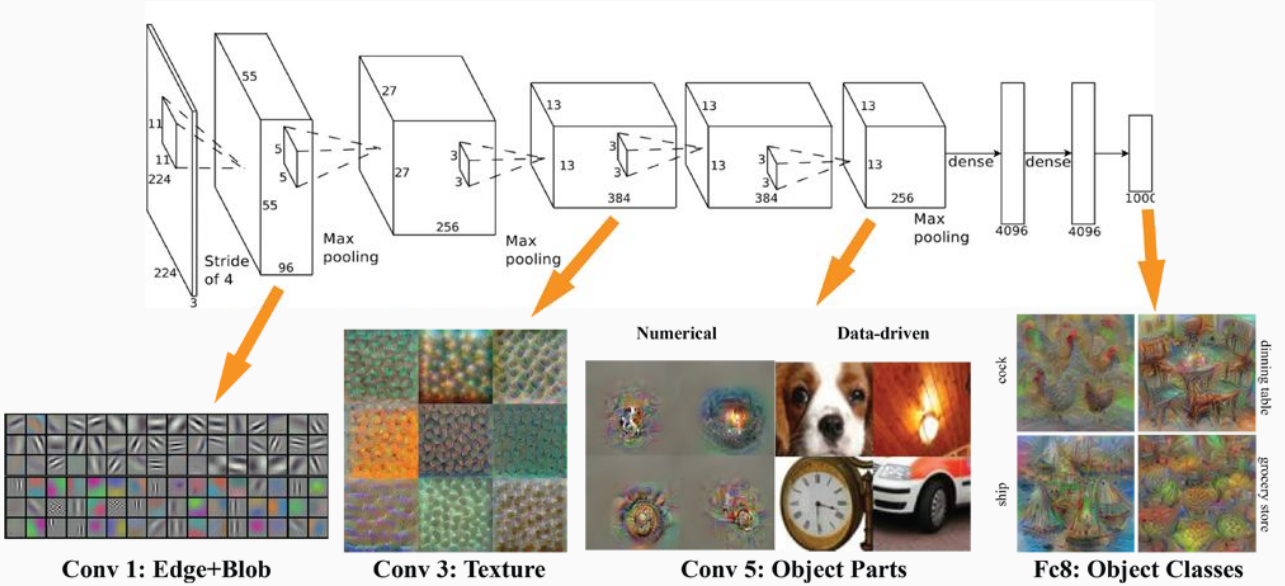


2 hidden layers (can be more)

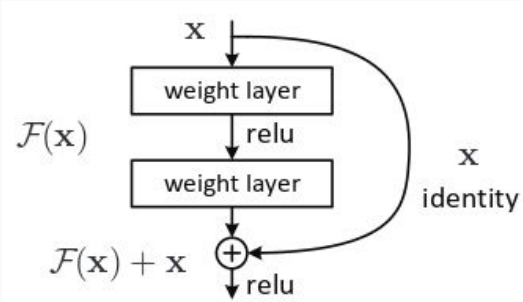
- Activation adds nonlinearity
- Learn parameters c, ω_i, θ_i, b_i
  - Gradient descent
- Uses back-propagation algorithm for calculating gradients
- Involves minimizing a loss function
  - Squared loss
  - Softmax

Deep neural networks usually vary in the structure and size of the hidden layer

Standard Deep NN architectures



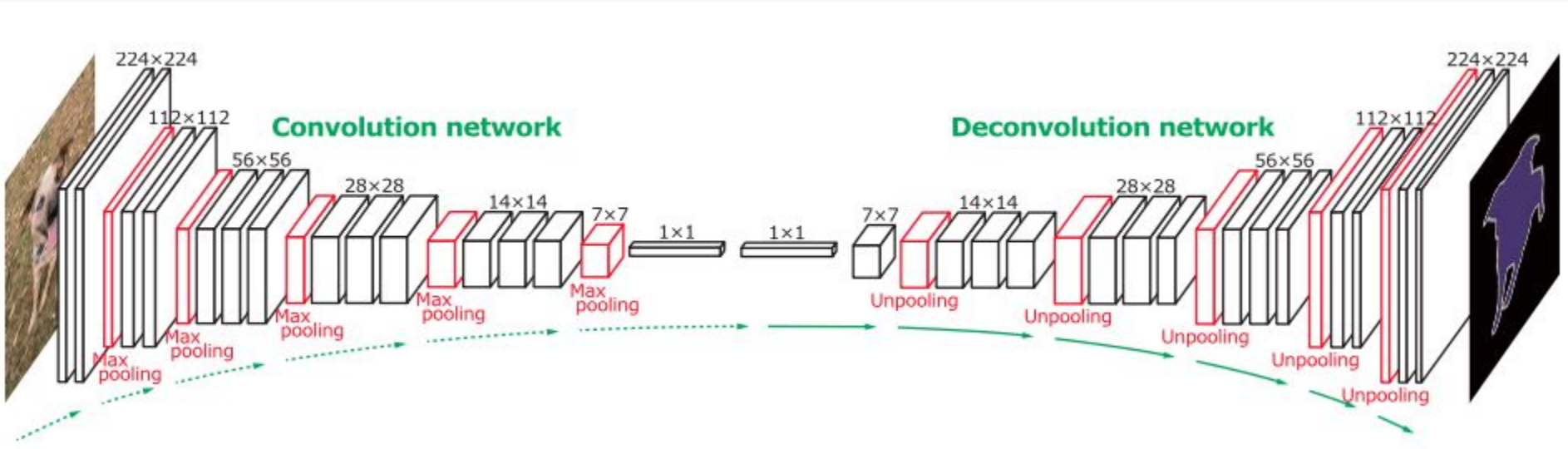
[image link]



Residual connections

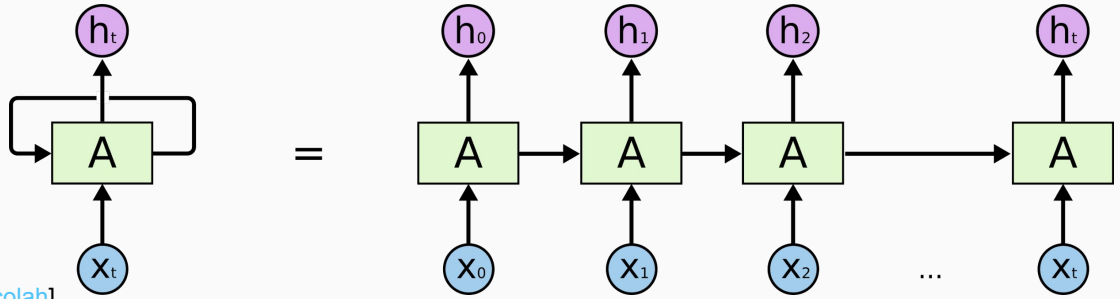
- Makes training feasible for nets with deeper layers
- Shortcuts connections allow gradient to flow back easily (counter dead neurons)

Deconvolution



Recurrent Neural Network (RNN)

- In feed forward NNs, inputs and outputs are independent
  - No concept of memory
  - Fixed length inputs
- RNNs can operate on sequence of inputs
  - Output depends on previous history of inputs
  - Introduces concept of memory

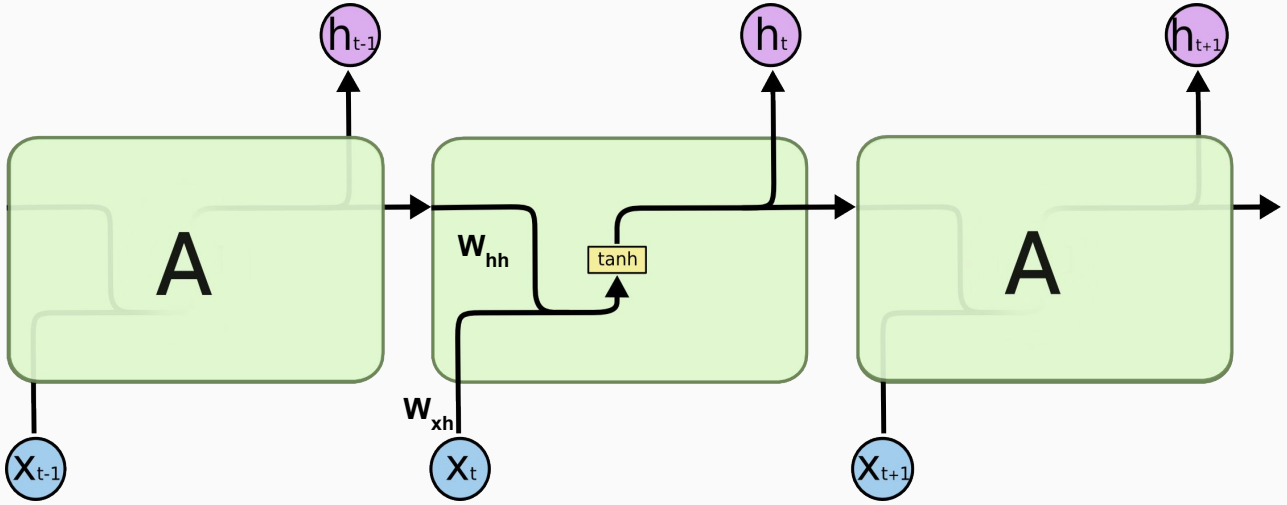


[image: colah]

RNN formulation

$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}X_t)$

$y_t = W_{hy}h_t$

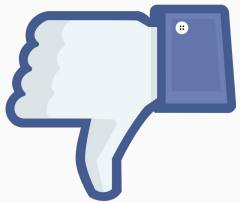


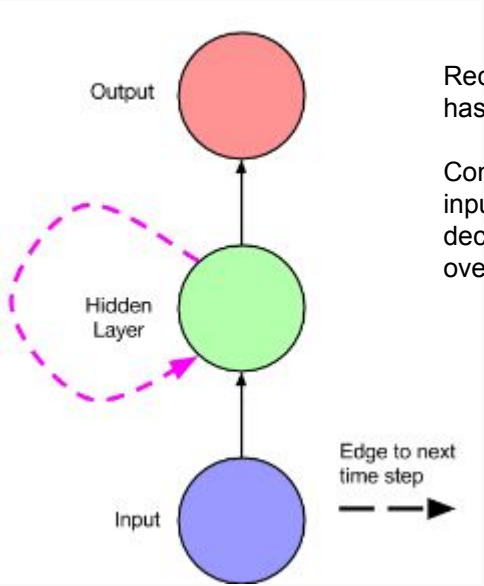
[image: colah]

weights are shared across time steps

RNNs are great, but...

- RNNs can deal with sequential data
  - Expressive models
- Cannot learn long term dependencies
- Training RNNs is not feasible
  - Vanishing/Exploding gradient
  - Numerically unstable





Recurrent edge always has same weight

Contribution of initial input grows or decreases exponentially over time

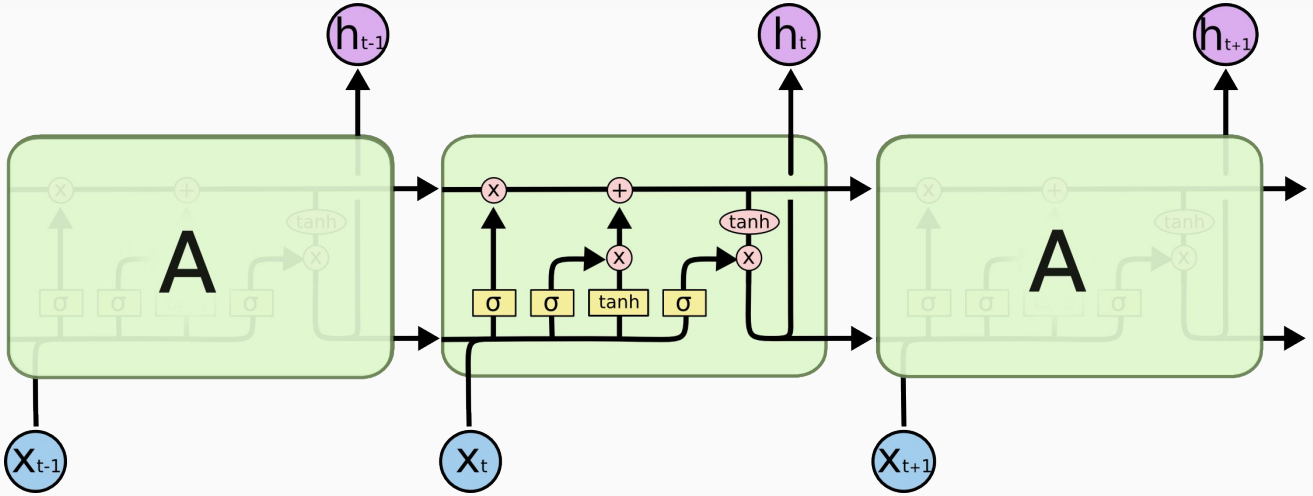
LSTM - Long short term memory

- An implementation of RNN
  - Can be effectively trained
  - Good at identifying long range dependencies
- They have been popularly used in various applications
- They differ in the structure of the repeating module
  - RNNs overwrite the hidden state
  - LSTMs add to the hidden state

LSTM formulation

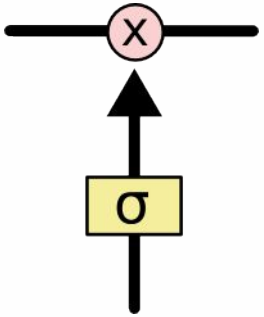
Recurring unit consists of:

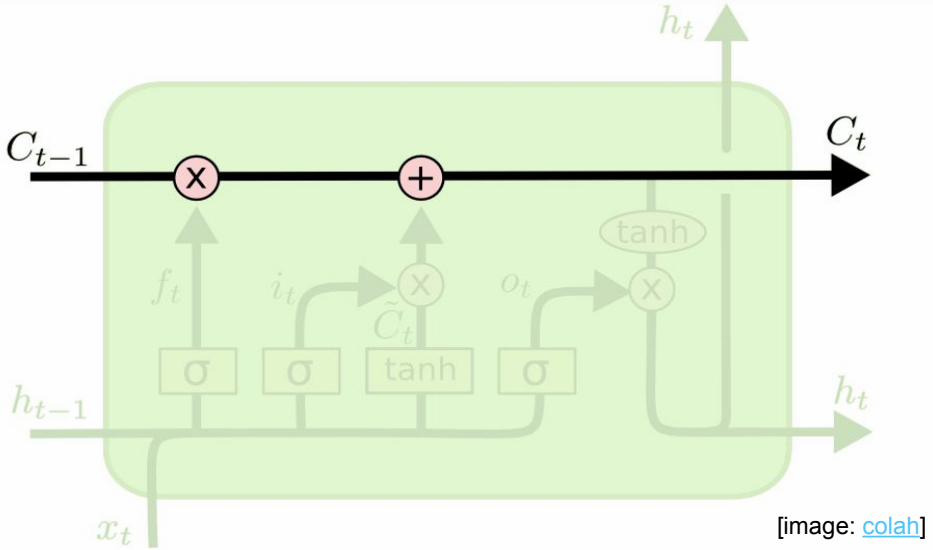
- Memory cell
- Gates:
  - Forget gate
  - Input gate
  - Output gate



[image: colah]

LSTM - memory cell



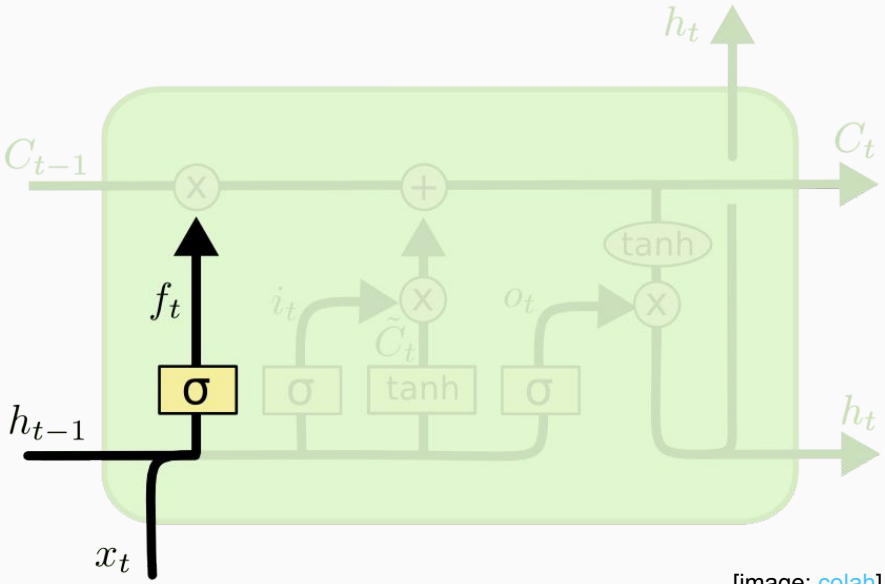


[image: colah]

gate

LSTMS allow modifying the cell state with gates.
A gate is a sigmoid layer followed by a pointwise multiplication.
Sigmoid output: 1 - let everything through, 0 - let nothing through

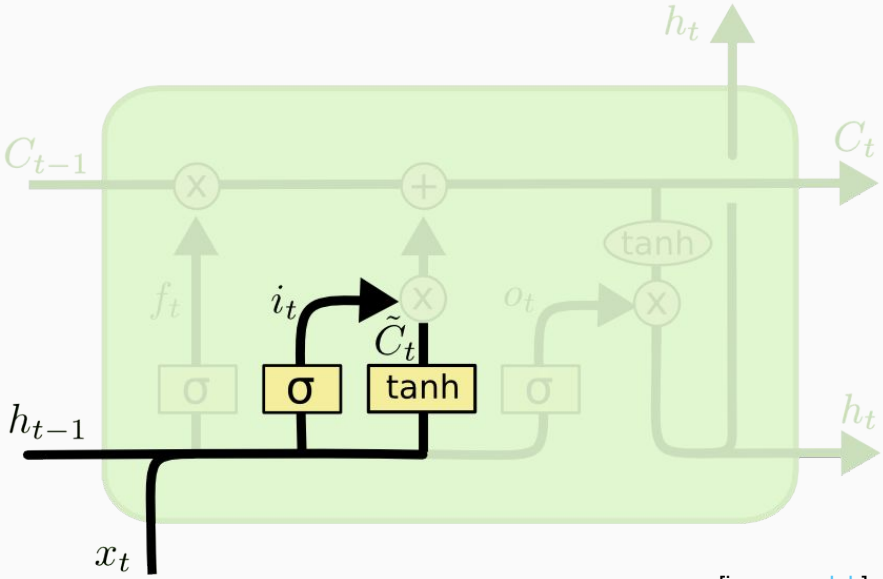
LSTM - forget gate



[image: colah]

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM - input gate

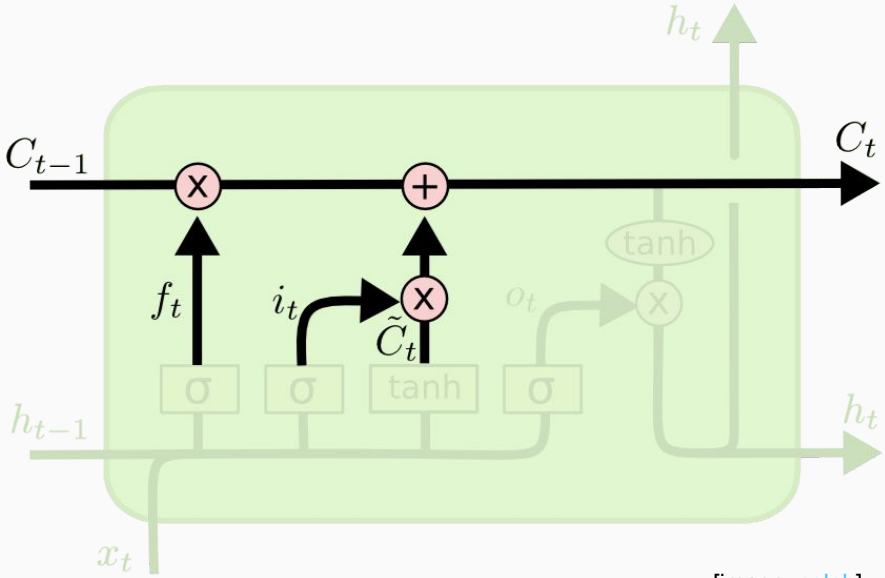


[image: colah]

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM - memory cell update



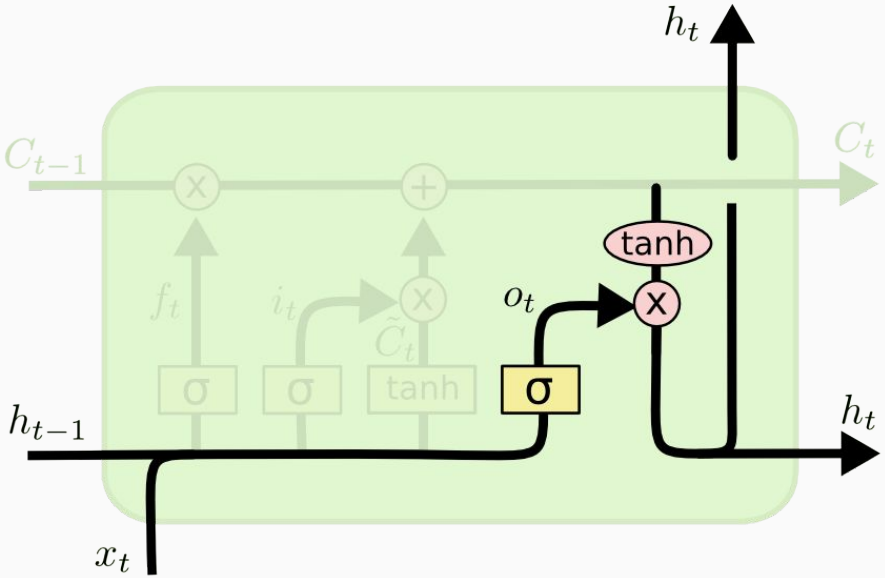
[image: colah]

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Forget old values

Add new values

LSTM - output gate

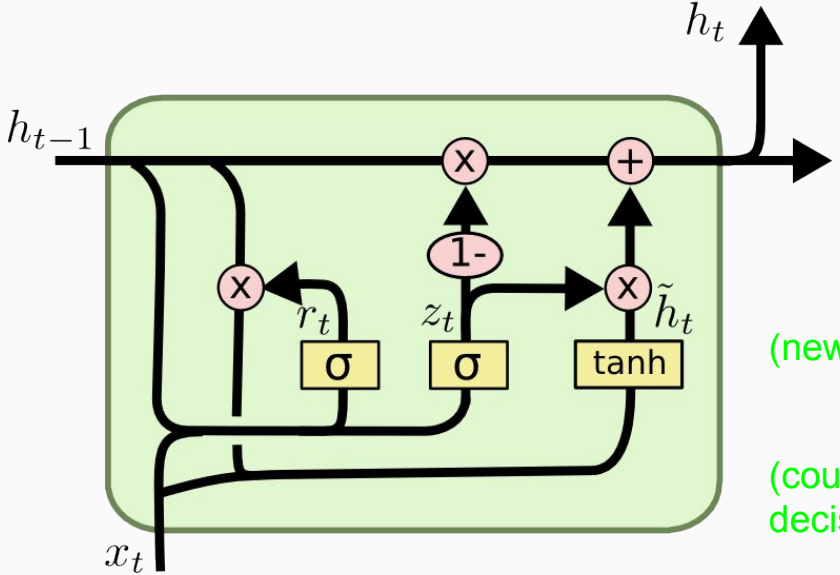


[image: colah]

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

GRU - Gated Recurrent Unit (a variant)



Merges cell state and hidden state
Combines forget and input gate

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

(new IPs) $$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

(coupled decision) $$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

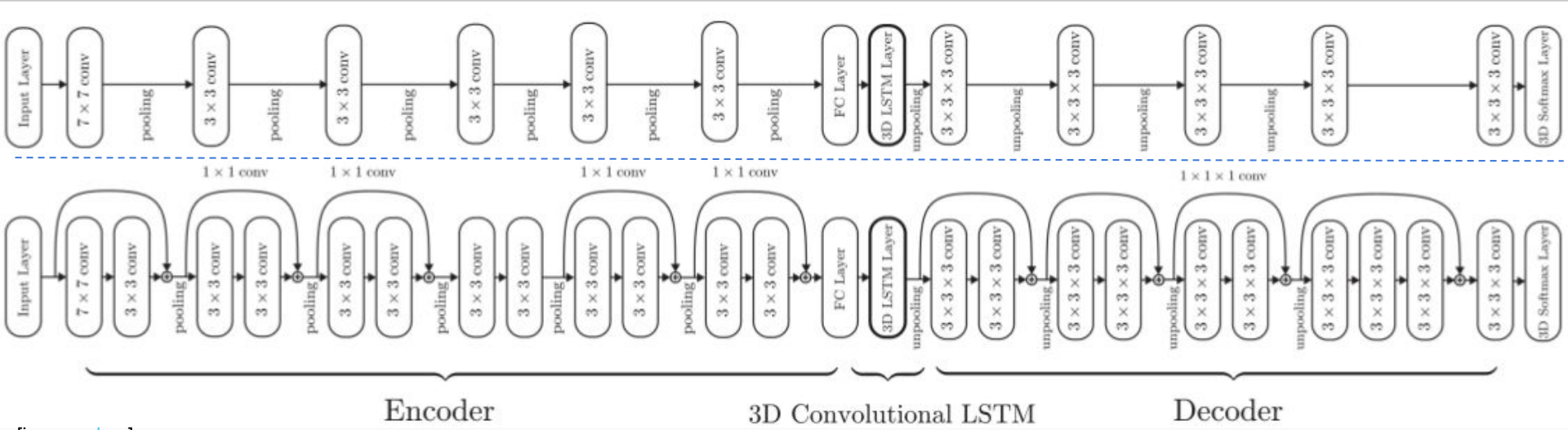
add when we forget something older

[image: colah]

So far...

- Standard CNN architecture
  - Convolution, pooling
  - Deconvolution - Upsampling
- LSTM
  - Can deal with sequential data, concept of memory
  - Memory cell, gates: forget, input, output
- GRU
  - Simplifies LSTM
  - Combines forget and input gates into a single update gate

NN Architecture for 3D reconstruction

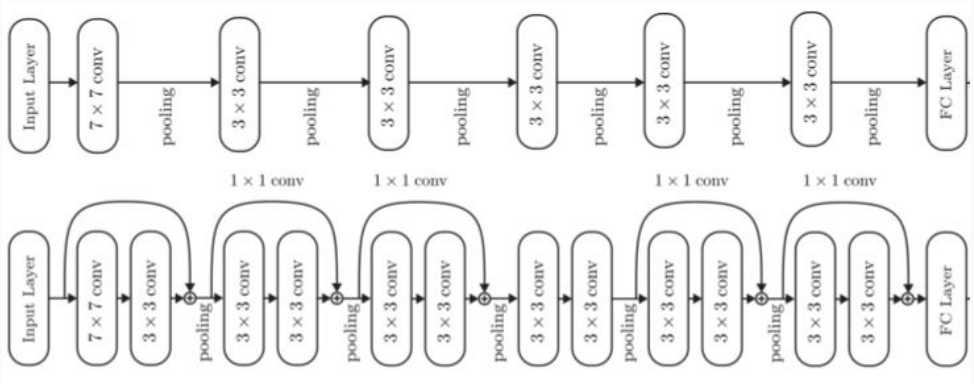


[image: choy]

3D LSTM remembers observations from different views
Refines reconstruction as more views become available

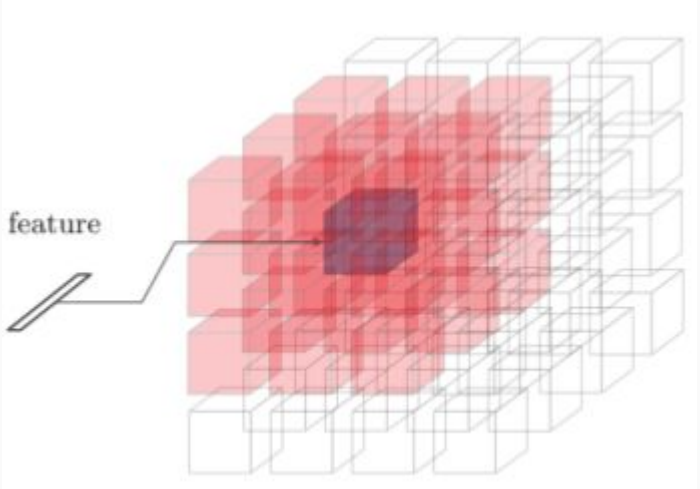
Encoder

- Encode image into low dim features
- Involves standard convolution, pooling and and fully connected layers
- Uses leaky Relu as activation function
- Also has a residual variant



3D recurrent Unit

- LSTM units arranged in 3D grid structure
- A unit receives
  - A feature vector from encoder
  - Hidden states of neighbours by convolution
- Each unit reconstructs a part of final output

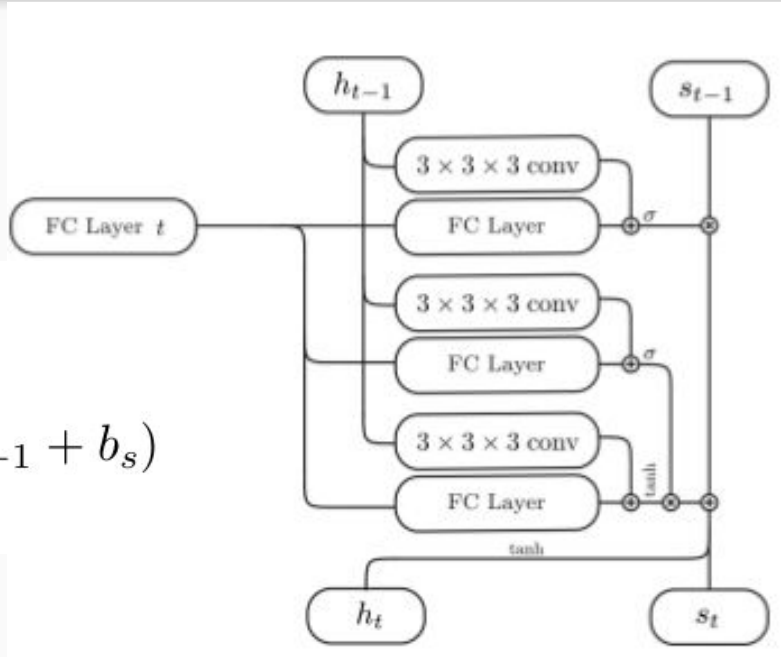


[image: choy]

LSTM variant

No output gate, reduces parameters

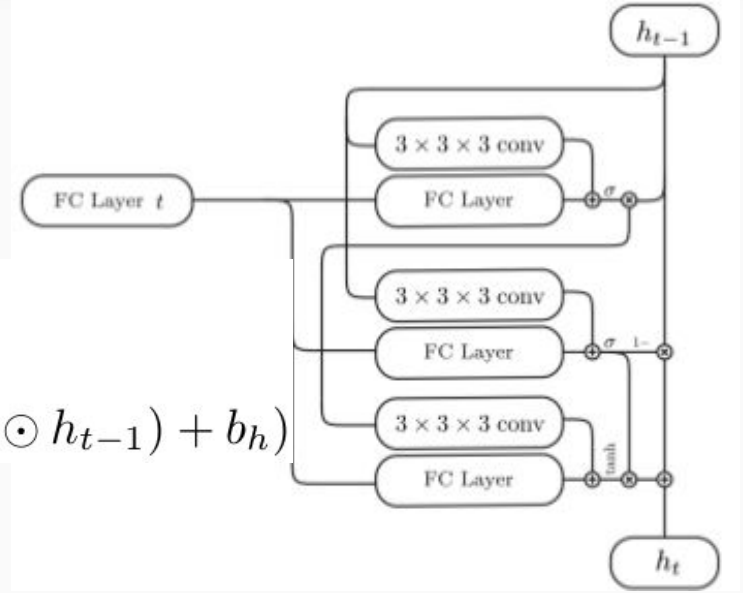
$$f_t = \sigma(W_f \mathcal{T}(x_t) + U_f * h_{t-1} + b_f)$$
$$i_t = \sigma(W_i \mathcal{T}(x_t) + U_i * h_{t-1} + b_i)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tanh(W_s \mathcal{T}(x_t) + U_s * h_{t-1} + b_s)$$
$$h_t = \tanh(s_t)$$



[image: choy]

GRU variant

$$u_t = \sigma(W_{fx}\mathcal{T}(x_t) + U_f * h_{t-1} + b_f)$$
$$r_t = \sigma(W_{ix}\mathcal{T}(x_t) + U_i * h_{t-1} + b_i)$$
$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \tanh(W_h\mathcal{T}(x_t) + U_h * (r_t \odot h_{t-1}) + b_h)$$



Decoder

- The hidden state is passed to the decoder
- Decoder upsamples to target output resolution
  - Applies nonlinearities, 3D deconvolution, unpooling
- Output of last activation is converted to occupancy probability
  - Uses voxel-wise softmax
- Finally we minimize the following cross entropy loss and backpropagate

$$L(\mathcal{X}, y) = \sum_{i,j,k} y_{(i,j,k)} \log(p_{(i,j,k)}) + (1 - y_{(i,j,k)}) \log(1 - p_{(i,j,k)})$$

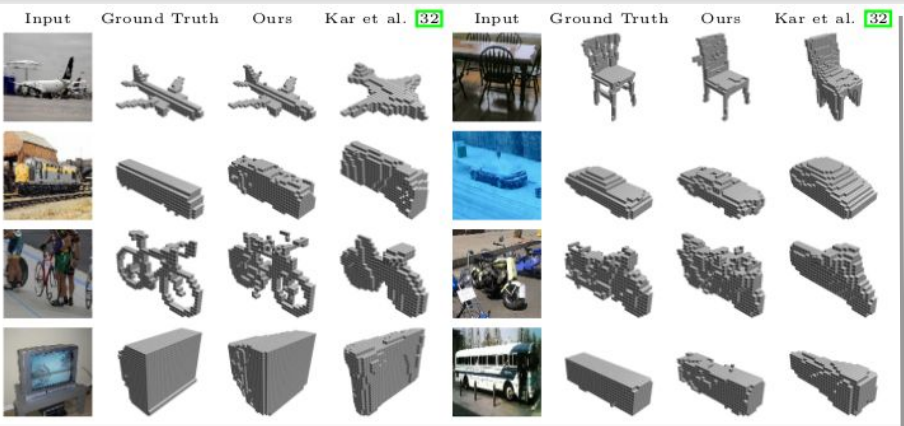
Implementation

- Training data
  - 3D CAD models for input images and ground truth occupancy grid
    - ShapeNet, PASCAL 3D, Online Products, MVS CAD Models
  - Images augmented with random crops from PASCAL VOC dataset
  - Viewpoints sampled randomly
- Training
  - Variable length inputs across different mini batches
  - Combines single and multi view reconstruction
- Network
  - Input 127x127, Output 32x32x32
  - 60k iterations, leaky relu (slope: 0.1)
  - Implemented in Theano, Adam for SGD update

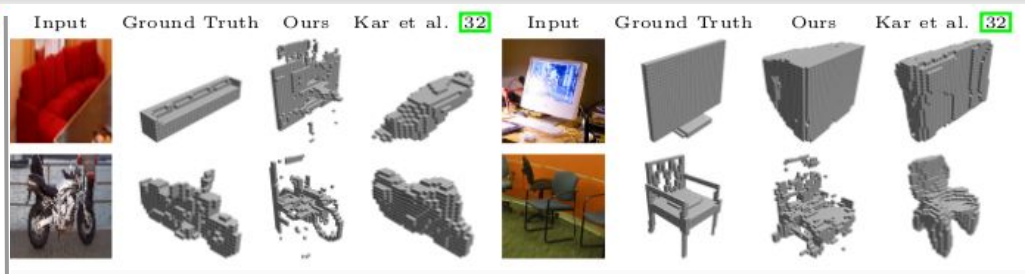
Network structure comparison

Arch	Encoder	Recurrence	Decoder	Loss	IoU
3D-LSTM-1	simple	LSTM	simple	0.116	0.499
3D-GRU-1	simple	GRU	simple	0.105	0.540
3D-LSTM-3	simple	LSTM	simple	0.106	0.539
3D-GRU-3	simple	GRU	simple	0.091	0.592
Res3D-GRU-3	residual	GRU	residual	0.080	0.634

Single view reconstruction



successful reconstructions

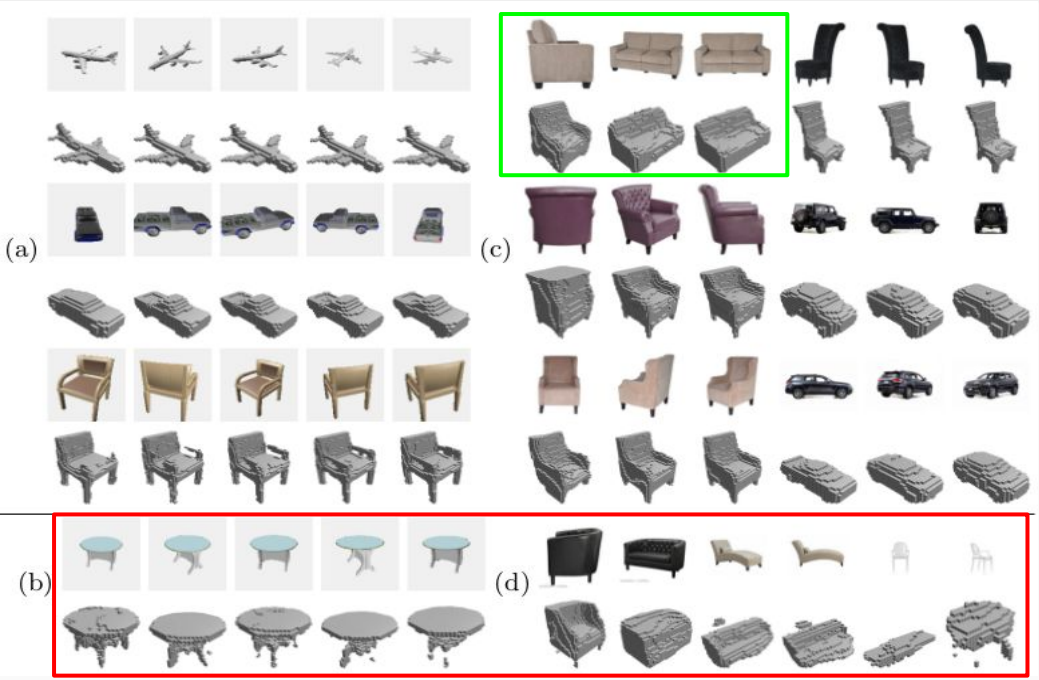


failed reconstructions

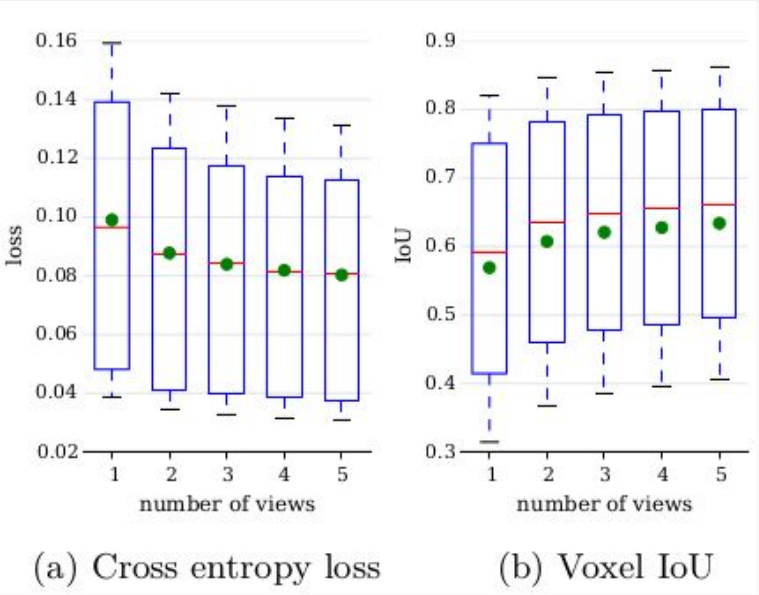
[image: choy]

	aero	bike	boat	bus	car	chair	mbike	sofa	train	tv	mean
Kar et al. 32	0.298	0.144	0.188	0.501	0.472	0.234	0.361	0.149	0.249	0.492	0.318
ours [LSTM-1]	0.472	0.330	0.466	0.677	0.579	0.203	0.474	0.251	0.518	0.438	0.456
ours [Res3D-GRU-3]	**0.544**	**0.499**	**0.560**	**0.816**	**0.699**	**0.280**	**0.649**	**0.332**	**0.672**	**0.574**	**0.571**

Multiview reconstruction

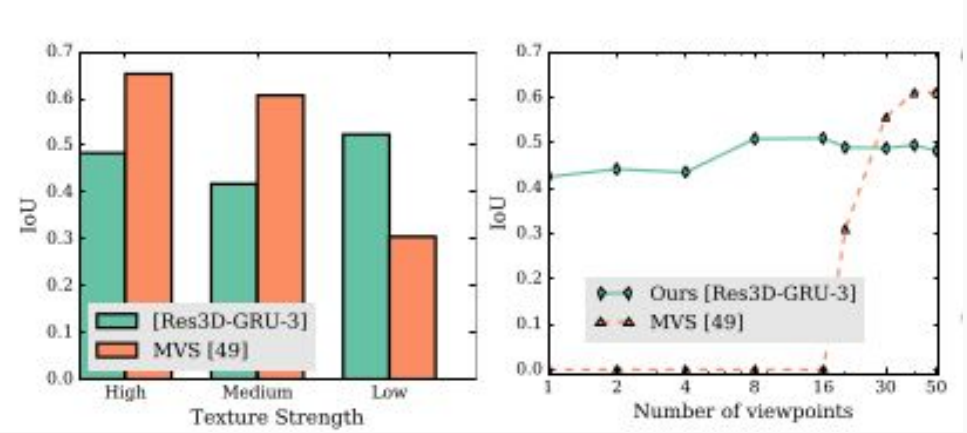


[image: choy]

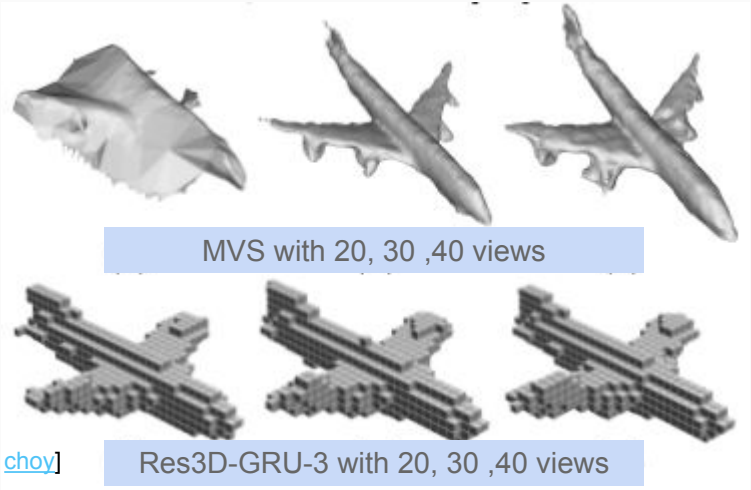


(a) Cross entropy loss (b) Voxel IoU

Multiview with texture



[image: choy]



Performs poorly for high textured objects, but works well for low textures

Lacks details

Discussion

- Reconstruction lacks details
- Performs poorly with high texture
  - Maybe the LSTM unit is using too high level features
  - Pass information from lower layers in the encoder?
- Reconstruction quality dependent on number of views used
  - Higher number of views will limit the batch size
  - Can higher number of views fit into a mini-batch?
- Experiments on LSTM grid size

Summary

- 3D reconstruction techniques
  - Single view, multi view
- Deep neural nets overview
  - convolution, pooling
  - Deconvolution
- Recurrent neural nets
  - Effectiveness and issues
  - LSTM, GRU
- Deep NN architecture for 3D reconstruction
  - Single framework for single and multi view reconstruction
  - Does single view reconstruction effectively
  - multi-view reconstruction can be improved.

Thank you!

Questions?