Hypernetworks: a versatile and powerful tool

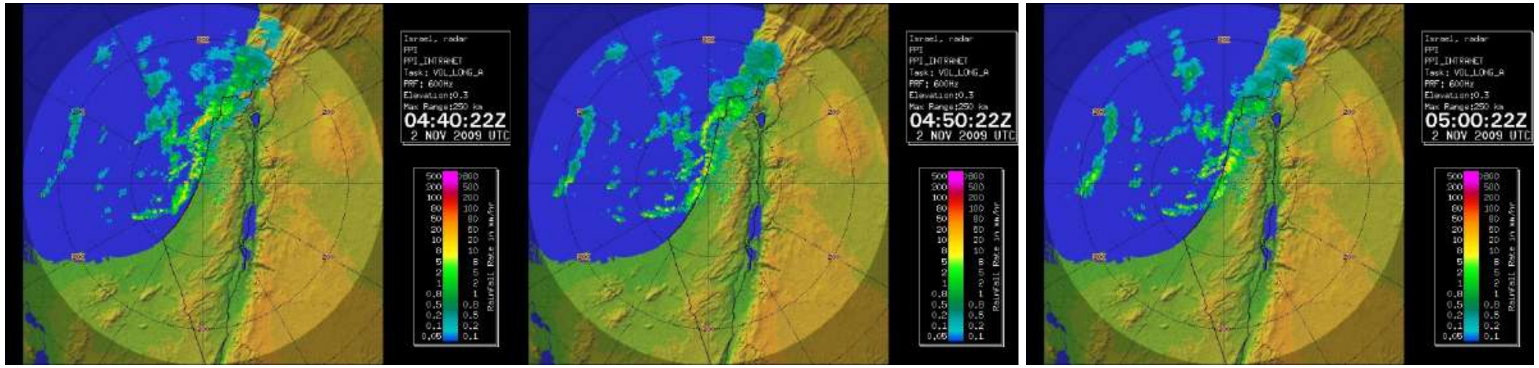
Prof. Lior Wolf
Facebook AI Research and
School of Computer Science, Tel Aviv University

Is AI slowing down or accelerating ?

The upcoming Cambrian explosion

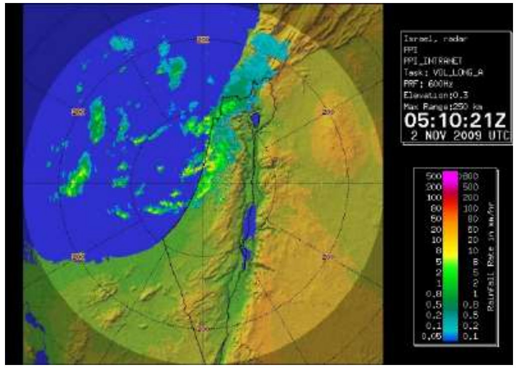
SHORT RANGE WEATHER PREDICTION

Input: a sequence of the recent history



- 4 radar images.
- Each 250x250 pixels
- 10 minutes between two consecutive frames

Output: the future



- The 200x200 center of the next image

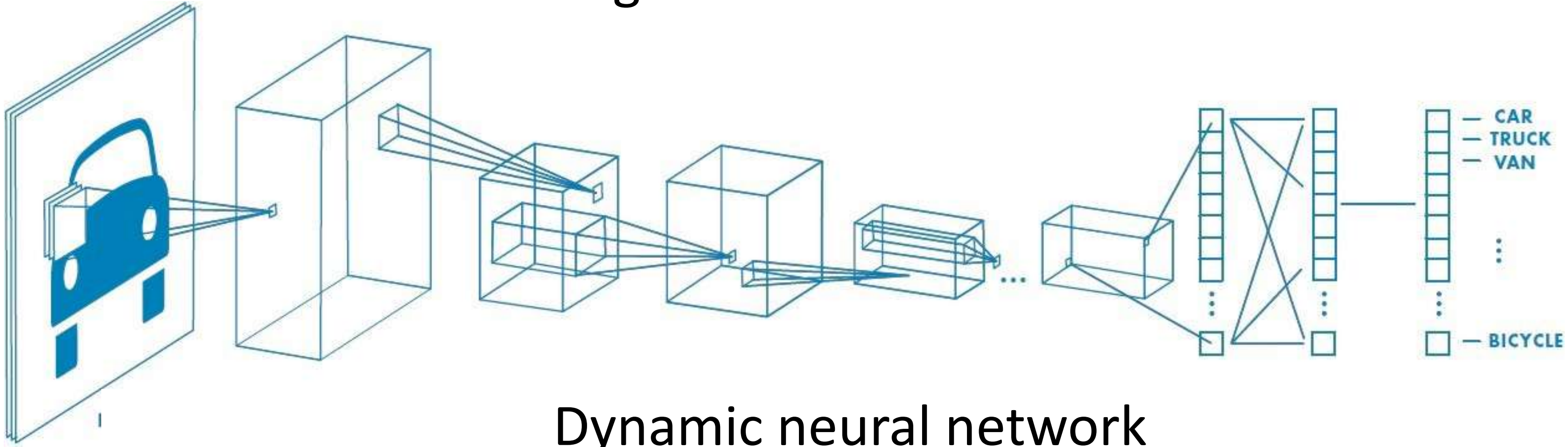
Preprocessing:

- Background subtraction
- Gray scale color transformation

B. Klein, L. Wolf, Y. Afek. **A Dynamic Convolution Layer for Short Range Weather Prediction.** *CVPR*, 2015.

ENTER DYNAMIC NETWORKS

Regular neural network



Dynamic neural network

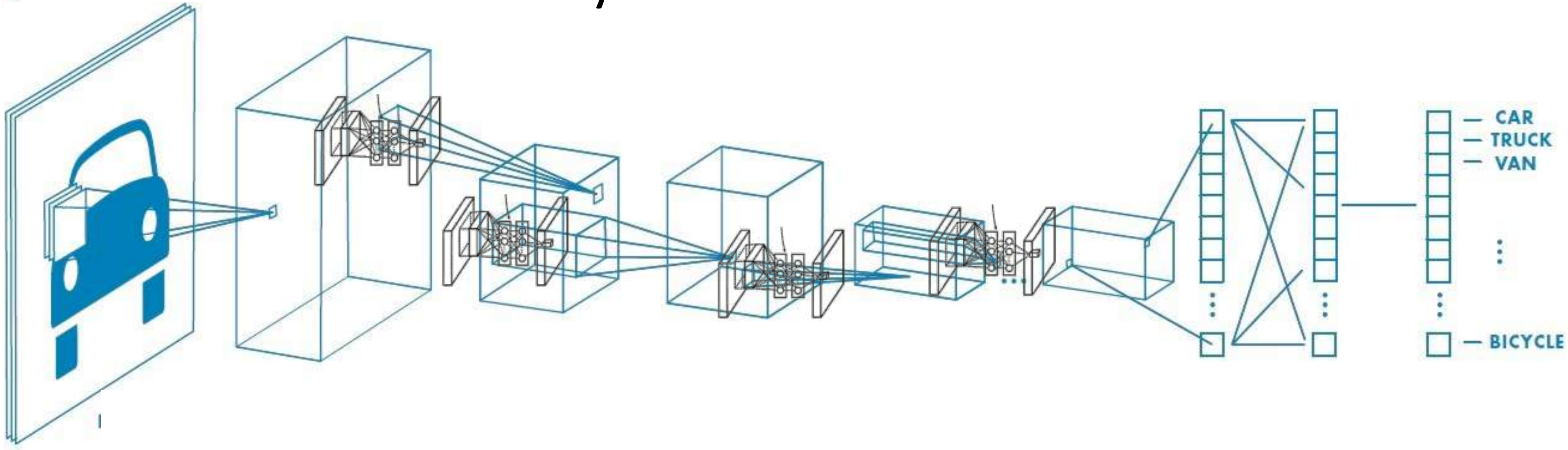
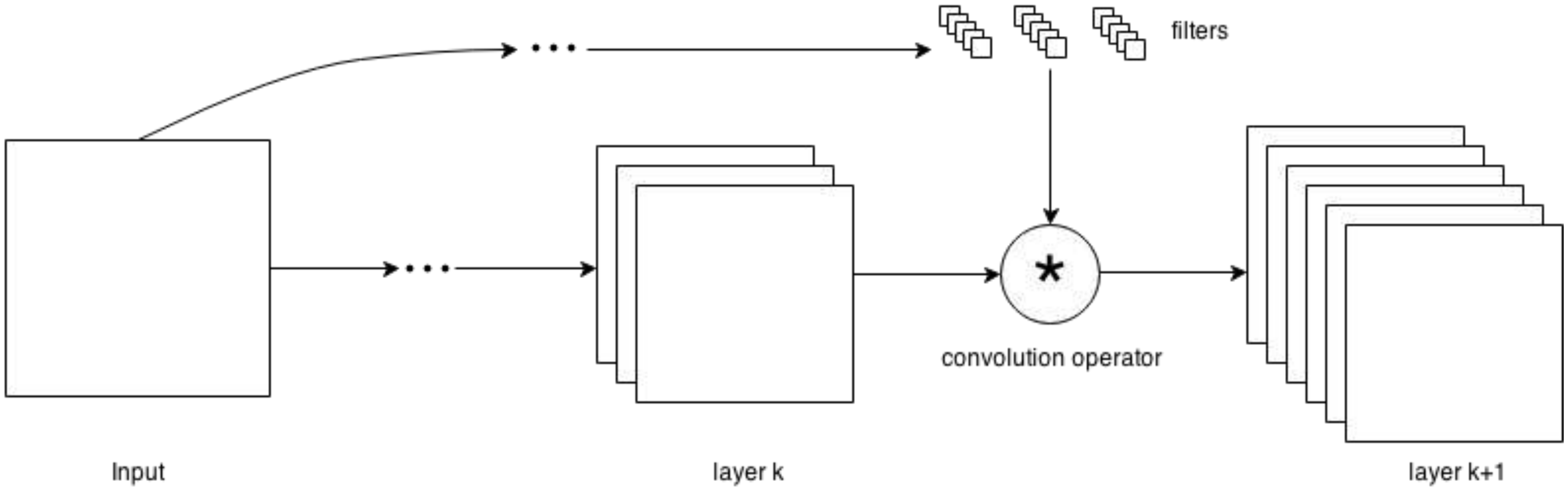


Image source: [Sumit Saha]

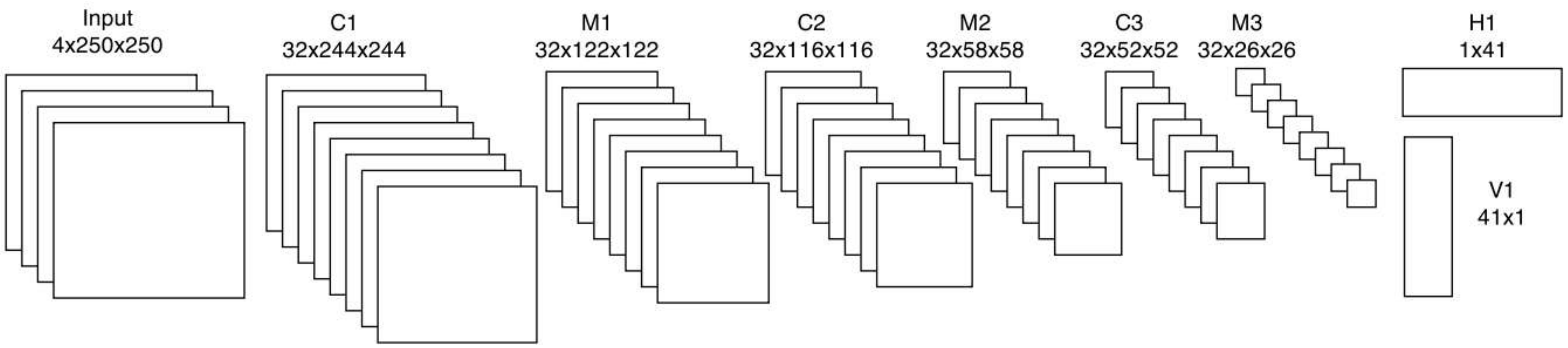
THE DYNAMIC CONVOLUTION LAYER

- A generalization of the convolution layer
- The filters are a function of the input
  - For every input, different filters will be applied

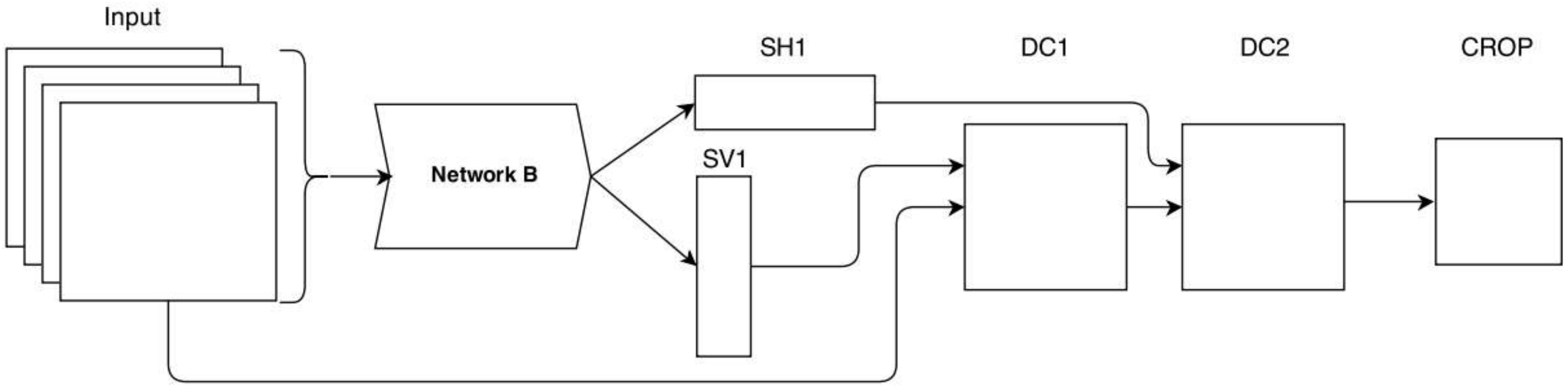


APPLIED TO WEATHER RADAR

- A convnet B computes horizontal and vertical displacements



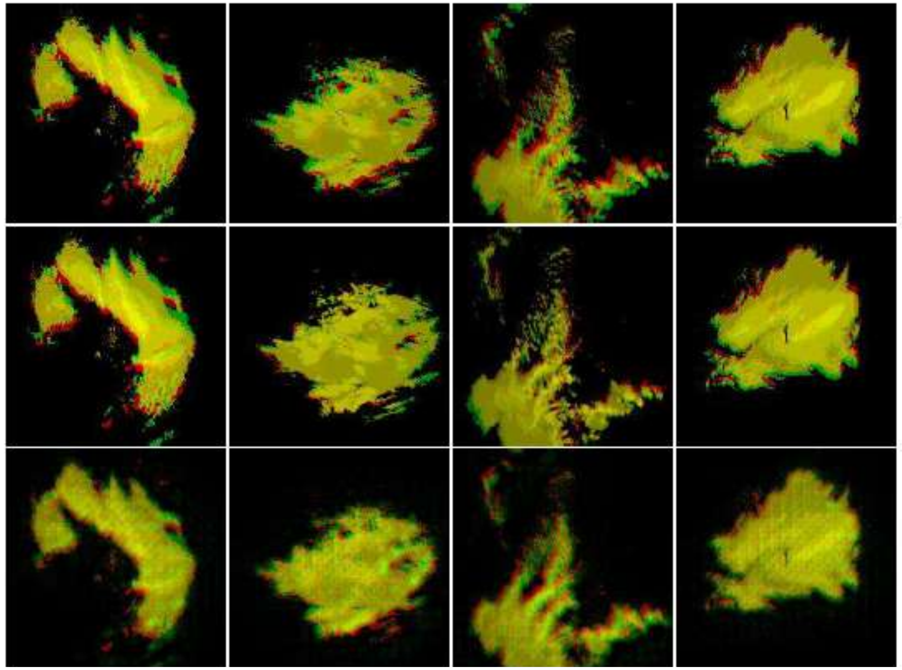
- Softmax makes the displacement filters sparser and they are applied to the last input image

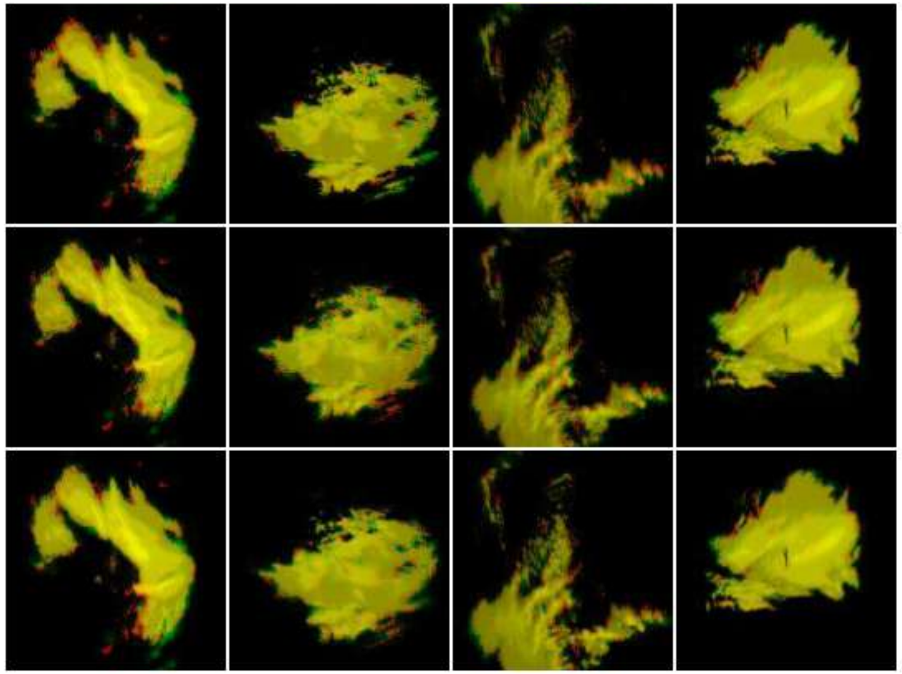


RESULTS

Method	Tel Aviv Dataset*	Davenport Dataset	Kansas City Dataset
Last Frame	20.059±0.536	258.818±2.552	241.392±2.975
Global Motion Estimator	16.837±0.496	173.402±1.547	179.953±2.065
Patch Based Linear Regression	13.002±0.435	164.854±1.377	160.489±1.682
Patch Based CNN	11.480±0.431	105.242±0.839	101.880±1.042
Whole Image Dynamic Convolution Network	12.340±0.461	117.316±0.929	118.402±1.174
Patch Based Dynamic Convolution Network	11.114±0.412	101.983±0.802	98.790±0.995

*Data courtesy of IMS





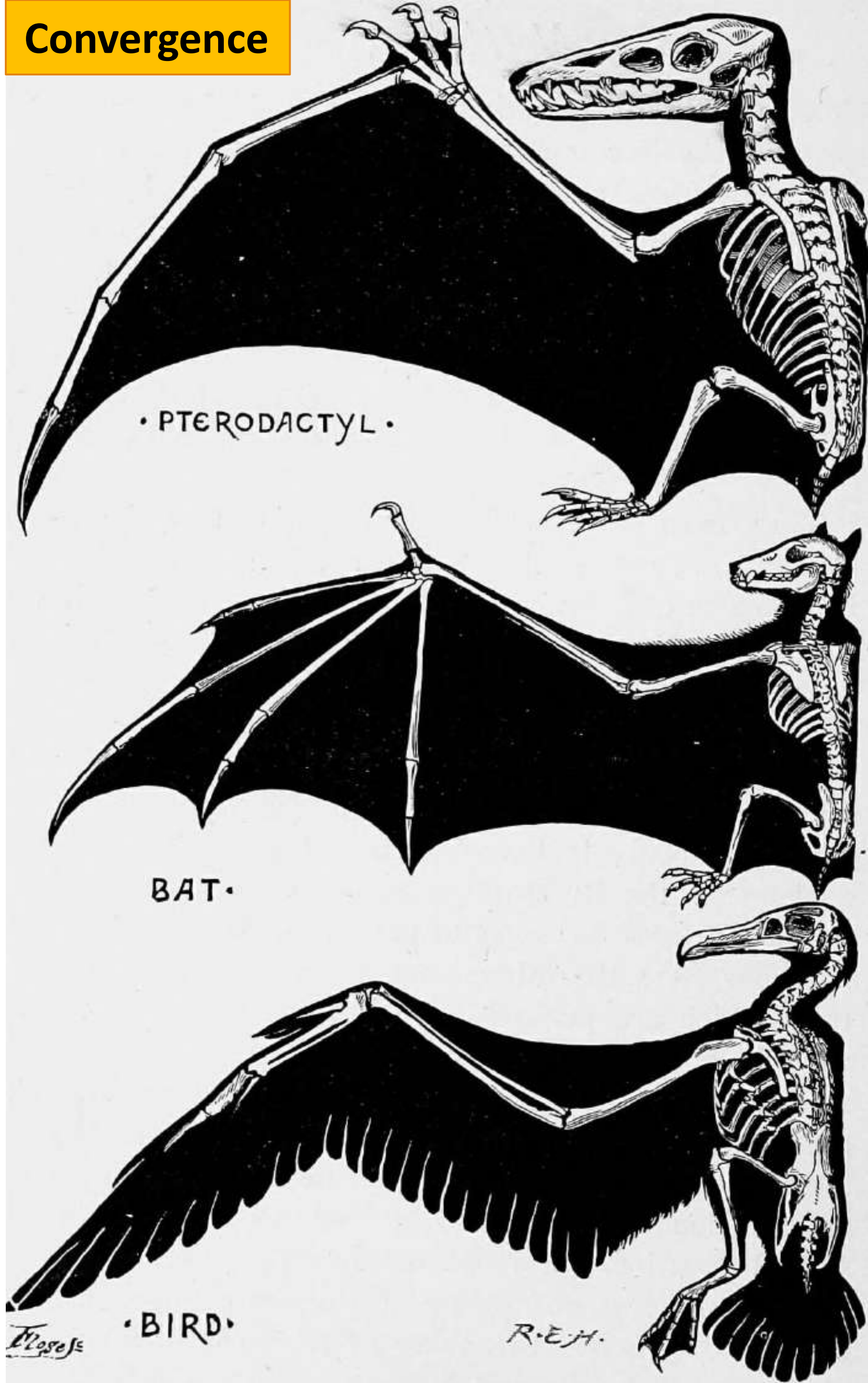
MYOPIC HISTORY OF DYNAMIC NETS

B. Klein, L. Wolf, Y. Afek. **A Dynamic Convolution Layer for Short Range Weather Prediction.** *CVPR*, 2015.

D. Ha, A. M. Dai, Q. V. Le. **Hypernetworks.** *ICLR*, 2017.

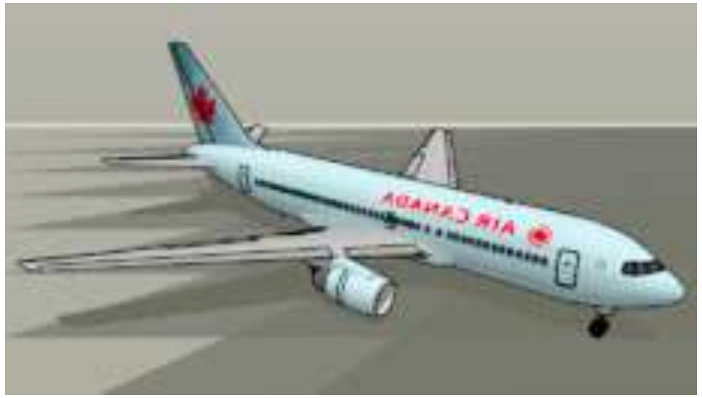
F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, M. Auli. **Pay less attention with lightweight and dynamic convolutions.** *ICLR, 2019*.

+ maybe a dozen more



THE COMEBACK OF DYNAMIC CONVOLUTIONS: “HYPERNETWORKS” FOR 3D RECONSTRUCTION

Input: a single image



Output: a 3D Model

G. Littwin, L. Wolf. **Deep Meta Functionals for Shape Representation.** ICCV, 2019.

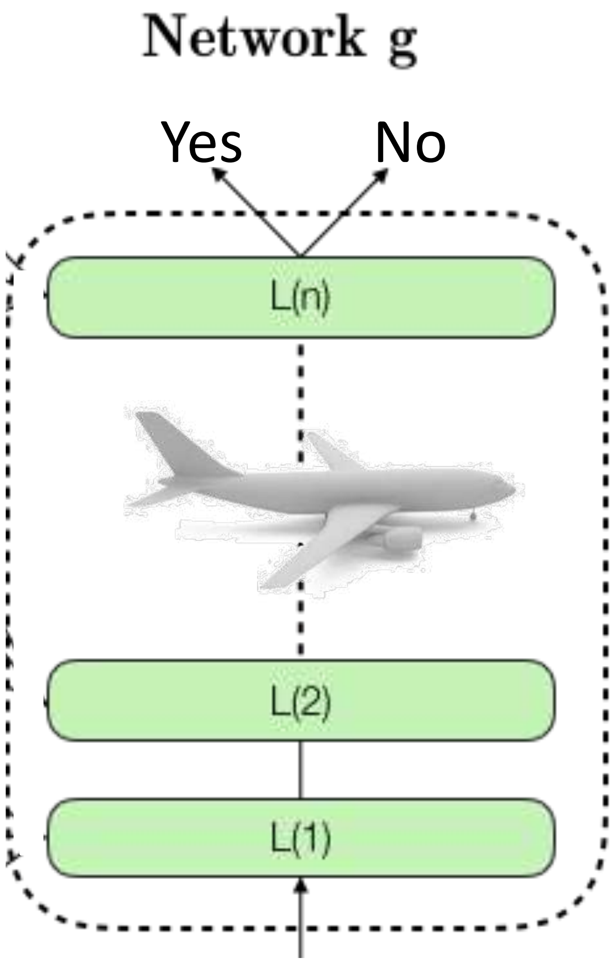
WHERE IS THE NETWORK?

Neural networks are **functions**

In hypernetworks, one function outputs another function

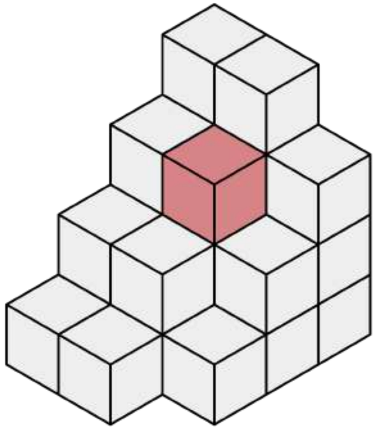
How is this related to 3D shapes?

The 3D shape is defined by a function!



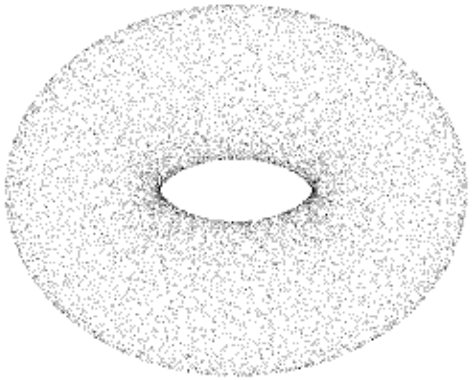
HOW TO REPRESENT A 3D MODEL?

	Voxels
Memory Footprint	High*
Reconstruction Resolution	Limited by memory
Topology	Limited by resolution
Train Time	Long
Rendering	Suited



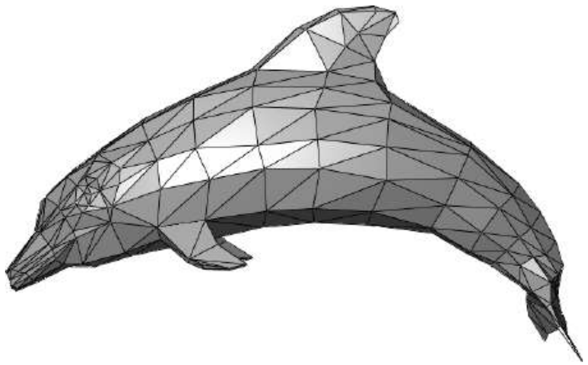
HOW TO REPRESENT A 3D MODEL?

	Voxels	Point Clouds
Memory Footprint	High*	Low
Reconstruction Resolution	Limited by memory	High
Topology	Limited by resolution	No topology
Train Time	Long	Short
Rendering	Suited	Suited



HOW TO REPRESENT A 3D MODEL?

	Voxels	Point Clouds	Polygon Mesh
Memory Footprint	High*	Low	Low
Reconstruction Resolution	Limited by memory	High	Limited by template mesh
Topology	Limited by resolution	No topology	Limited by template mesh
Train Time	Long	Short	Short
Rendering	Suited	Suited	Very suited



HOW TO REPRESENT A 3D MODEL?

	Voxels	Point Clouds	Polygon Mesh	Implicit functions
Memory Footprint	High*	Low	Low	High
Reconstruction Resolution	Limited by memory	High	Limited by template mesh	Unlimited
Topology	Limited by resolution	No topology	Limited by template mesh	Unlimited
Train Time	Long	Short	Short	Long
Rendering	Suited	Suited	Very suited	Suited

$$g\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}, e_I\right)$$

Inside the shape

Outside the shape

Parallel work:
L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, A. Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. CVPR, 2019

HOW TO REPRESENT A 3D MODEL?

	Voxels	Point Clouds	Polygon Mesh	Implicit functions	Meta Functionals
Memory Footprint	High*	Low	Low	High	Low
Reconstruction Resolution	Limited by memory	High	Limited by template mesh	Unlimited	Unlimited
Topology	Limited by resolution	No topology	Limited by template mesh	Unlimited	Unlimited
Train Time	Long	Short	Short	Long	Short
Rendering	Suited	Suited	Very suited	Suited	Suited

META FUNCTIONALS = IMPLICIT FUNCTIONS + HYPERNETWORKS

Formulation:

$$\theta_I = f(I, \theta_f)$$

$$s_I^p = g(p, \theta_I) = g(p, f(I, \theta_f)) \qquad p = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

SIMPLE CROSS ENTROPY TRAINING LOSS

Formulation:

$$\theta_I = f(I, \theta_f)$$

$$s_I^p = g(p, \theta_I) = g(p, f(I, \theta_f))$$

Loss:

$$L(\theta_f, I) = -\int_{p \in V} y(p)\log(s_I^p) + \int_{p \in V} (1 - y(p))\log(1 - s_I^p)\,dp$$

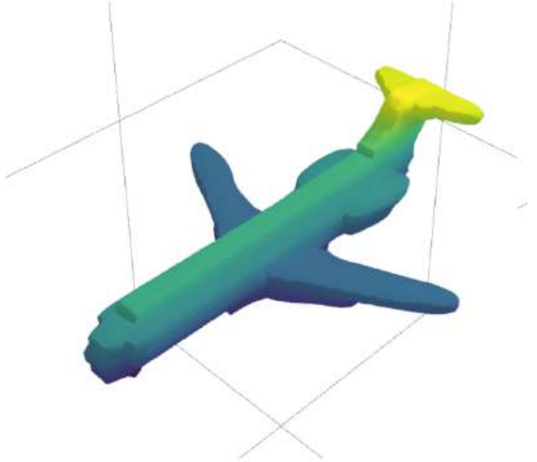
$$y(p) \in \{0,1\}$$

BOUNDARY SAMPLING

Formulation:

$$\theta_I = f(I, \theta_f)$$

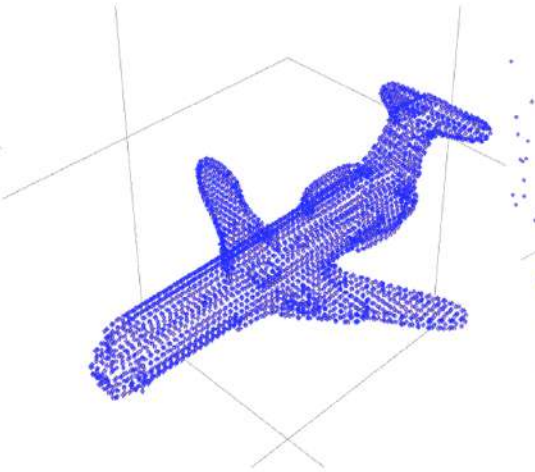
$$s_I^p = g(p, \theta_I) = g(p, f(I, \theta_f))$$

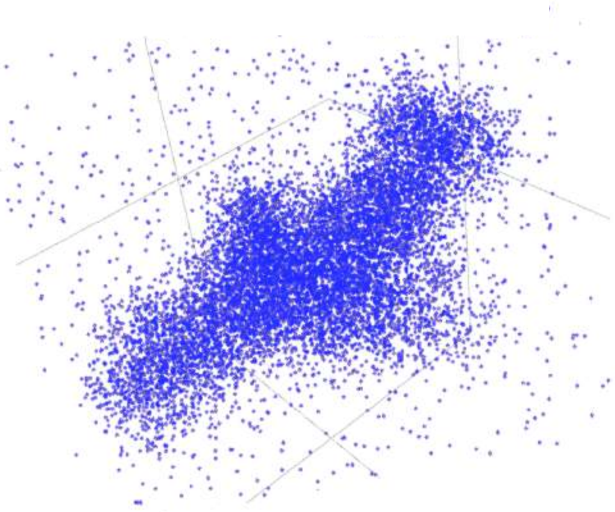


Loss:

$$L(\theta_f, I) = -\int_{p \in V} y(p) \log(s_I^p) + \int_{p \in V} (1 - y(p)) \log(1 - s_I^p) dp$$

$$y(p) \in \{0,1\}$$

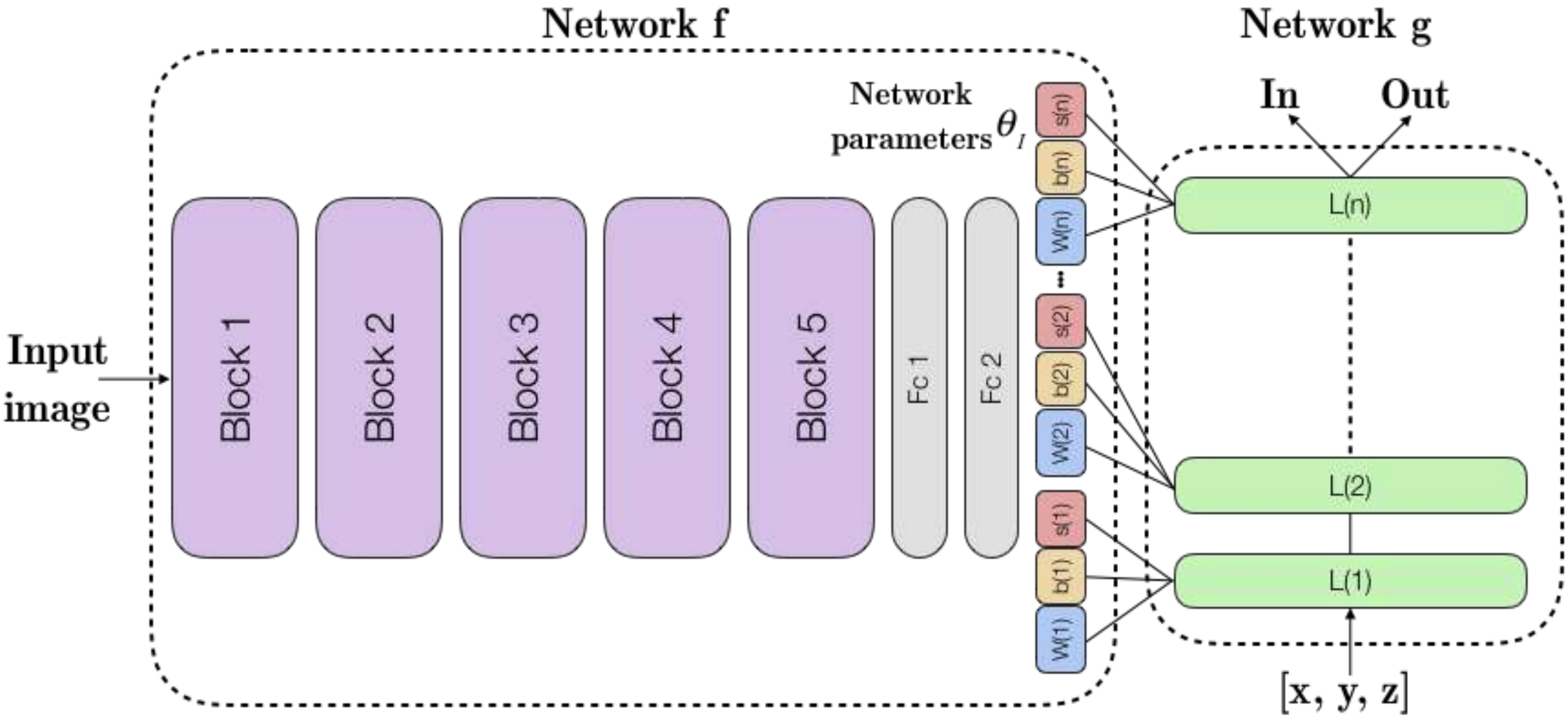




ARCHITECTURE

$$\theta_I = f(I, \theta_f)$$

$$s_I^p = g(p, \theta_I) = g(p, f(I, \theta_f))$$



RESULTS ON THE SHAPENET BENCHMARK

Method	airplane	bench	cabinet	car	cellphone	chair	couch	firearm	lamp	monitor	speaker	table	watercraft	mean
3D-R2N2 [8]	51.3	42.1	71.6	79.8	66.1	46.6	62.8	54.4	38.1	46.8	66.2	51.3	51.3	56.0
OGN [35]	58.7	48.1	72.9	81.6	70.2	48.3	64.6	59.3	39.8	50.2	63.7	53.6	63.2	59.6
PSGN [10]	60.1	55.0	77.1	83.1	74.9	54.4	70.8	60.4	46.2	55.2	73.7	60.6	61.1	64.0
VTN [32]	67.1	63.7	76.7	82.1	74.2	55.0	69.0	62.6	43.6	53.4	68.1	57.3	59.9	64.1
MTN [32]	64.7	57.7	77.6	85.0	75.6	54.7	68.1	61.6	40.8	53.2	70.1	57.3	59.1	63.5
PCDI [37]	61.2	60.9	68.3	83.2	74.4	57.2	69.9	**69.5**	46.4	61.4	69.8	61.5	58.5	64.8
Ours	**71.4**	**65.9**	**79.3**	**87.1**	**79.1**	**60.7**	**74.8**	68.0	**48.6**	**61.7**	**73.8**	**62.8**	**65.4**	**69.1**

Mean IOU (%), Data provided by Choy et al.

Method	airplane	bench	cabinet	car	cellphone	chair	couch	firearm	lamp	monitor	speaker	table	watercraft	mean
3D-R2N2 [8]	56.7	43.2	61.8	77.6	65.8	50.9	58.9	56.5	40.0	44.0	56.7	51.6	53.1	55.1
LSM [18]	61.1	50.8	65.9	79.3	67.7	57.8	67.0	**69.7**	48.1	53.9	63.9	55.6	58.3	61.5
VP3D [20]	69.1	59.8	72.4	80.2	**77.5**	60.1	65.6	66.4	50.5	**59.7**	**68.0**	60.7	**61.3**	65.5
Ours	**71.3**	**63.4**	**75.6**	**81.5**	75.1	**61.4**	**72.3**	65.7	**52.0**	56.2	64.7	**61.6**	60.2	**66.2**

Mean IOU (%), Data provided by Hanee et al.

RESULTS ON THE SHAPENET BENCHMARK

Method	airplane	bench	cabinet	car	cellphone	chair	couch	firearm	lamp	monitor	speaker	table	watercraft	mean
3D-R2N2 [8]	51.3	42.1	71.6	79.8	66.1	46.6	62.8	54.4	38.1	46.8	66.2	51.3	51.3	56.0
OGN [35]	58.7	48.1	72.9	81.6	70.2	48.3	64.6	59.3	39.8	50.2	63.7	53.6	63.2	59.6
PSGN [10]	60.1	55.0	77.1	83.1	74.9	54.4	70.8	60.4	46.2	55.2	73.7	60.6	61.1	64.0
VTN [32]	67.1	63.7	76.7	82.1	74.2	55.0	69.0	62.6	43.6	53.4	68.1	57.3	59.9	64.1
MTN [32]	64.7	57.7	77.6	85.0	75.6	54.7	68.1	61.6	40.8	53.2	70.1	57.3	59.1	63.5
PCDI [37]	61.2	60.9	68.3	83.2	74.4	57.2	69.9	**69.5**	46.4	61.4	69.8	61.5	58.5	64.8
Ours	**71.4**	**65.9**	**79.3**	**87.1**	**79.1**	**60.7**	**74.8**	68.0	**48.6**	**61.7**	**73.8**	**62.8**	**65.4**	**69.1**

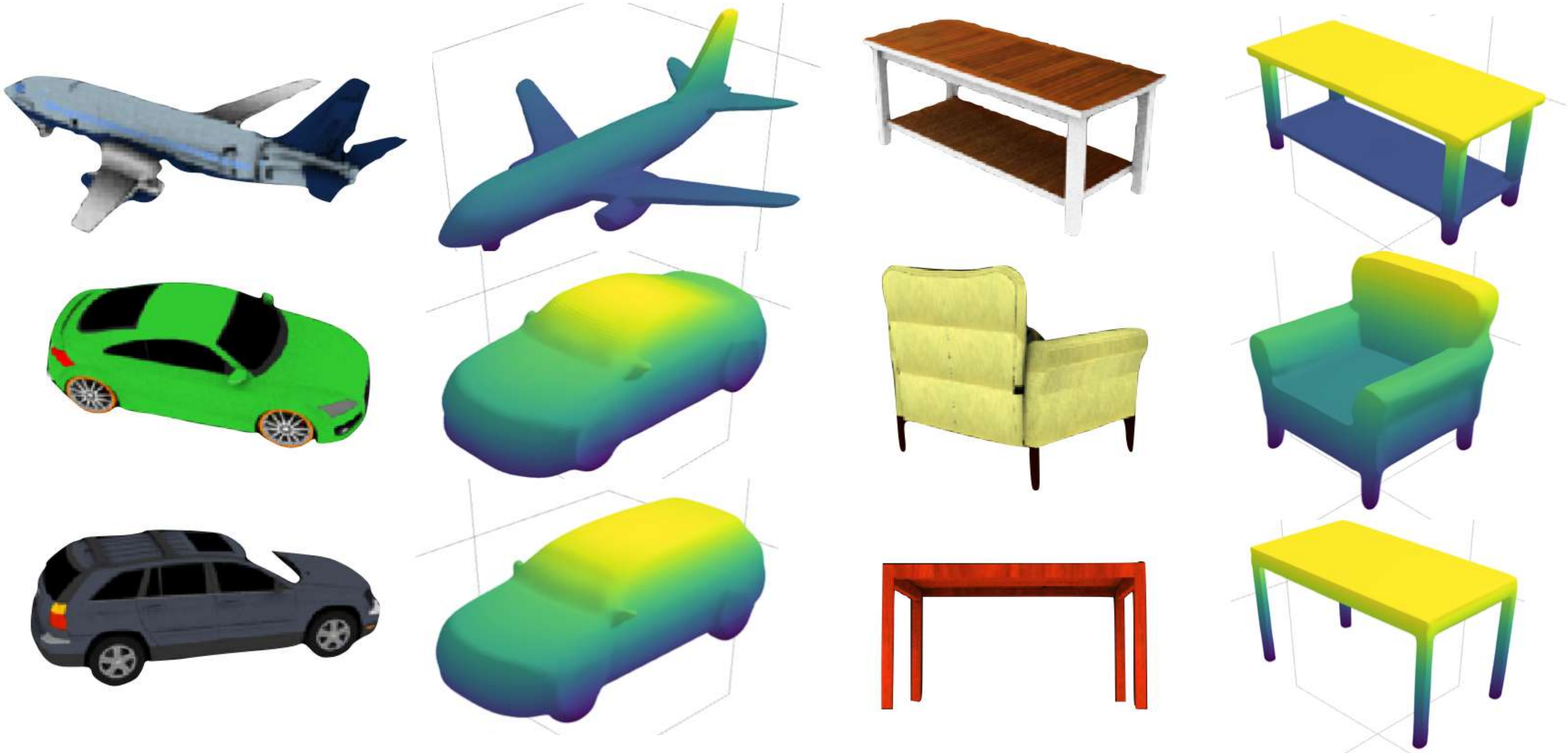
Mean IOU (%), Data provided by Choy et al.

Method	airplane	bench	cabinet	car	cellphone	chair	couch	firearm	lamp	monitor	speaker	table	watercraft	mean
3D-R2N2 [8]	56.7	43.2	61.8	77.6	65.8	50.9	58.9	56.5	40.0	44.0	56.7	51.6	53.1	55.1
LSM [18]	61.1	50.8	65.9	79.3	67.7	57.8	67.0	**69.7**	48.1	53.9	63.9	55.6	58.3	61.5
VP3D [20]	69.1	59.8	72.4	80.2	**77.5**	60.1	65.6	66.4	50.5	**59.7**	**68.0**	60.7	**61.3**	65.5
Ours	**71.3**	**63.4**	**75.6**	**81.5**	75.1	**61.4**	**72.3**	65.7	**52.0**	56.2	64.7	**61.6**	60.2	**66.2**

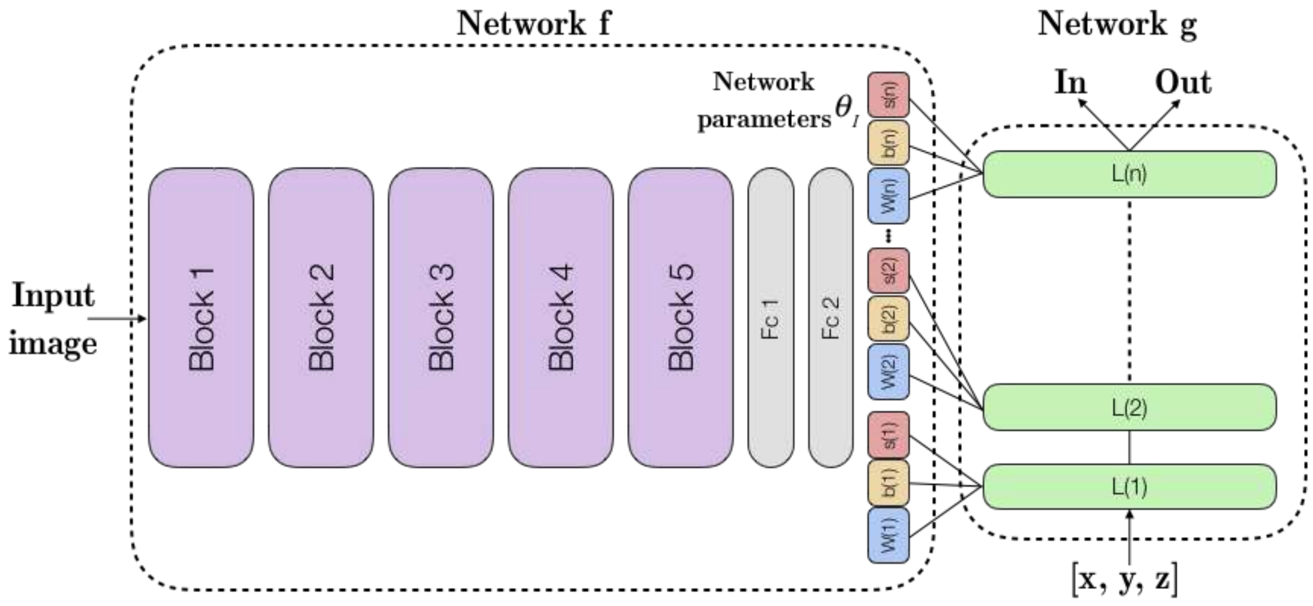
Mean IOU (%), Data provided by Hanee et al.

1. We train a single model for all classes
2. We do not do pretraining on ImageNet
3. Implicit functions without hypernets is not competitive

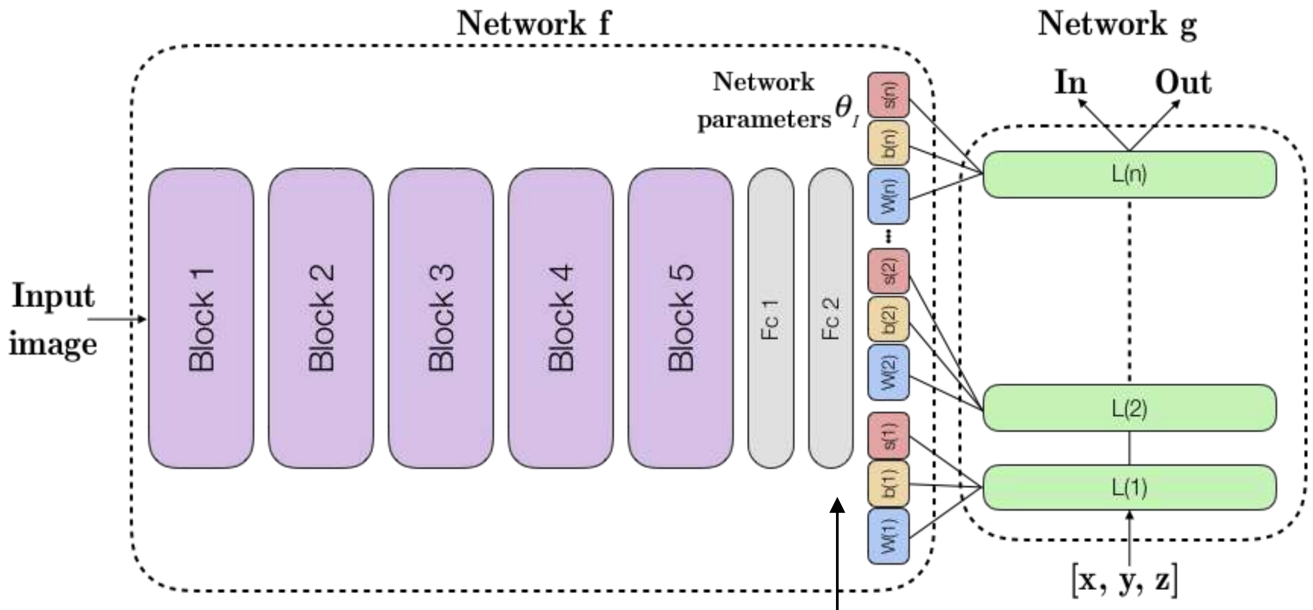
SAMPLE RESULTS



LATENT SHAPE SPACE

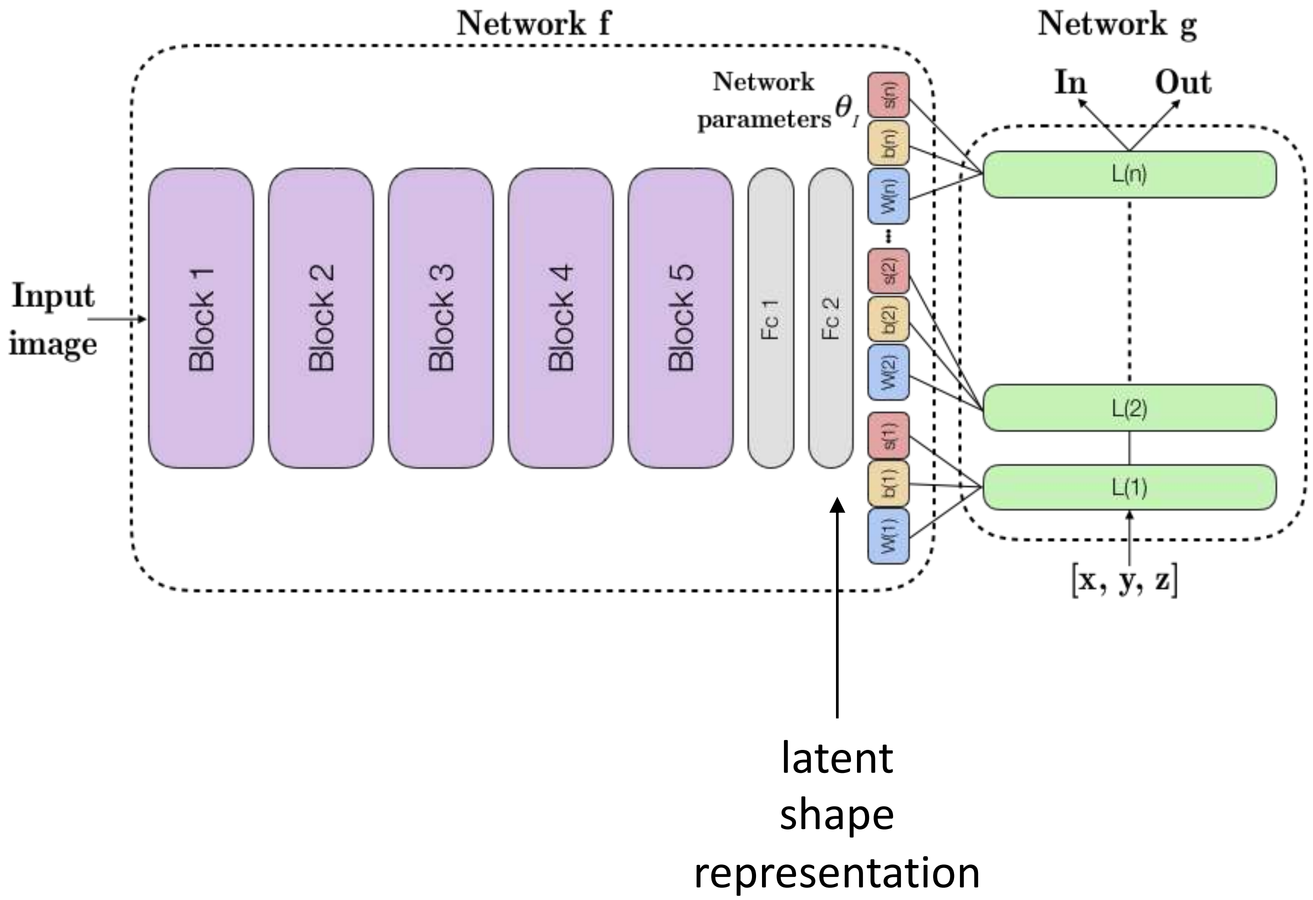


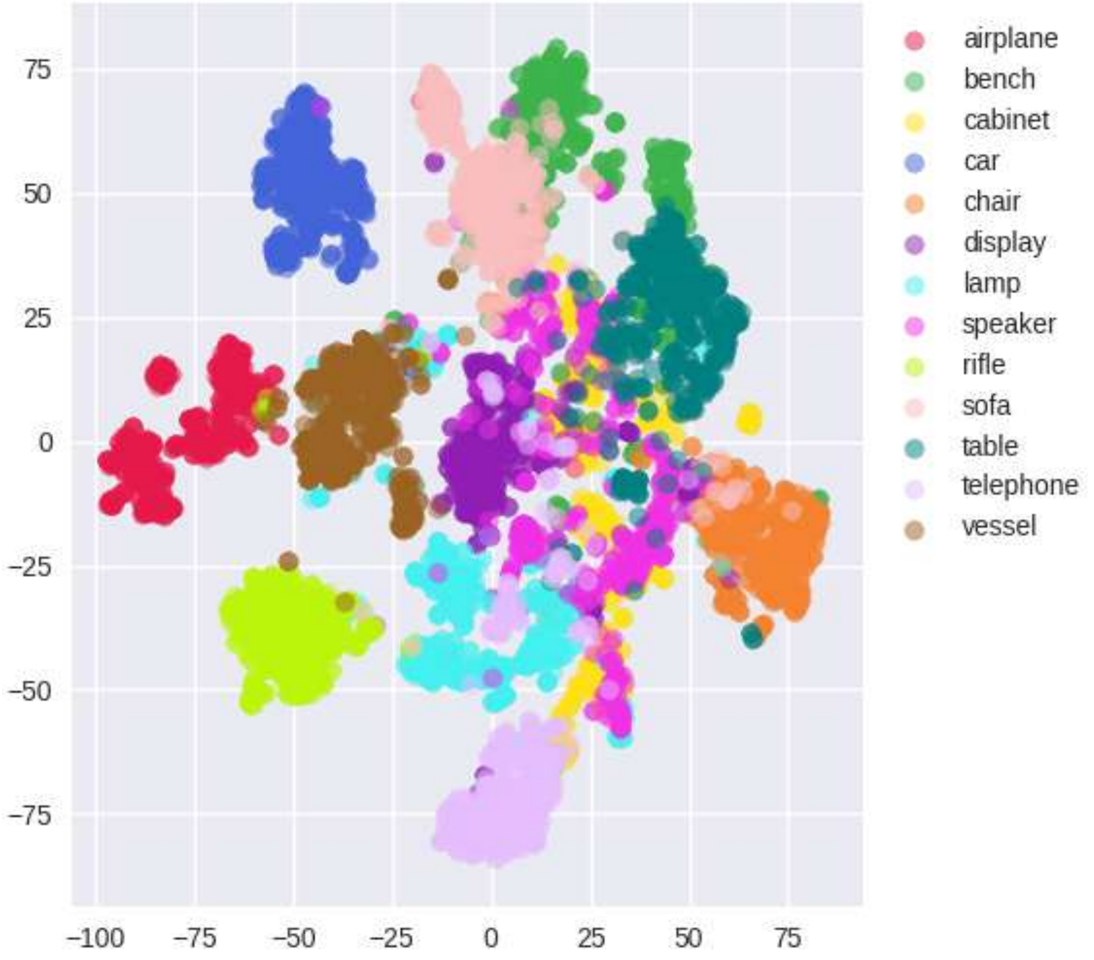
LATENT SHAPE SPACE



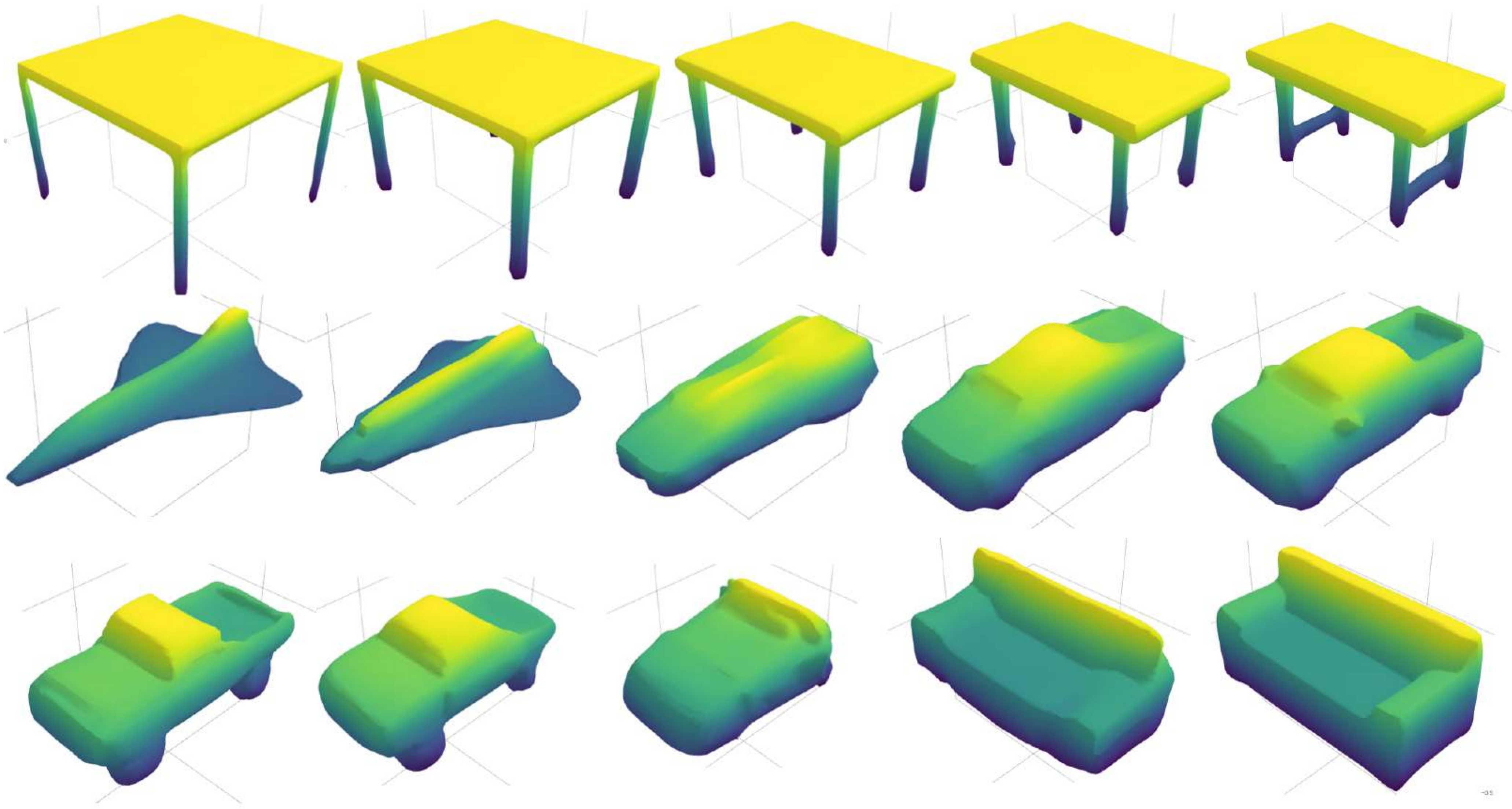
latent
shape
representation

LATENT SPACE VISUALIZATION





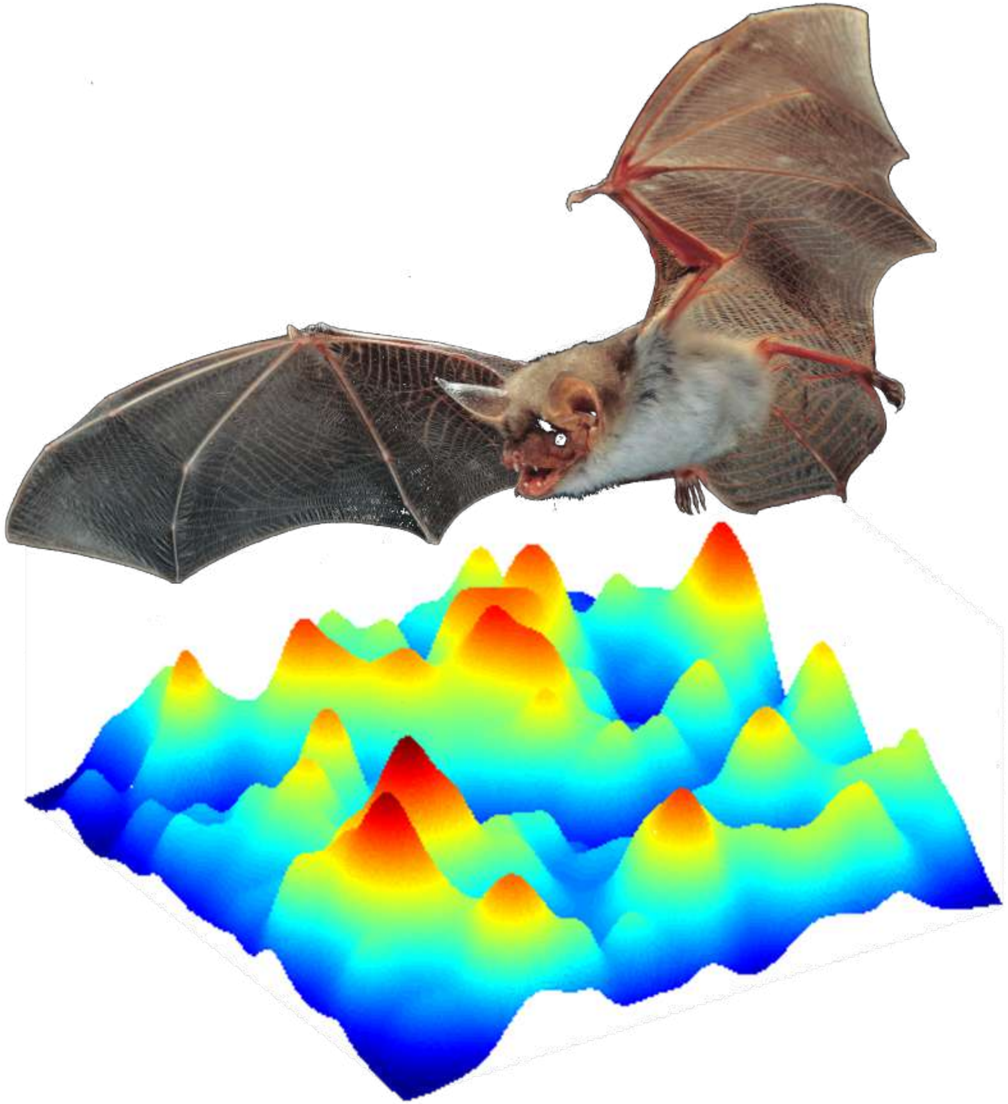
LATENT SPACE INTERPOLATION



DIFFERENTIABLE RENDERING -- TRAINING FROM 2D IMAGES

Input image Silhouette label Silhouette prediction

THE SUPERMAN VS. BATMAN PROJECT



We understand vision well, but Sonar is still a mystery

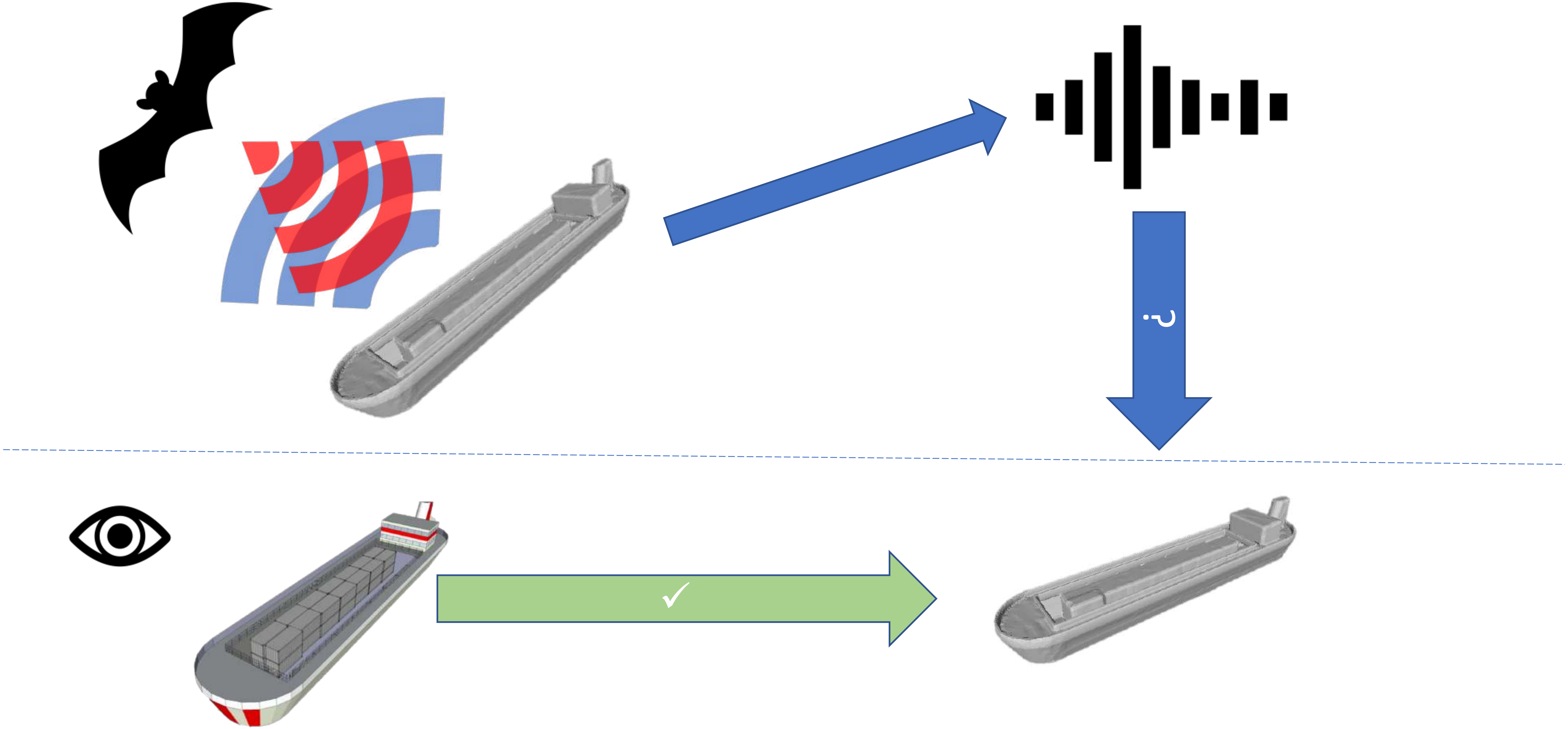
So we build a sonar-based 3D reconstruction method and compare to vision on a similar set of objects

* Vision and sonar results are highly correlated

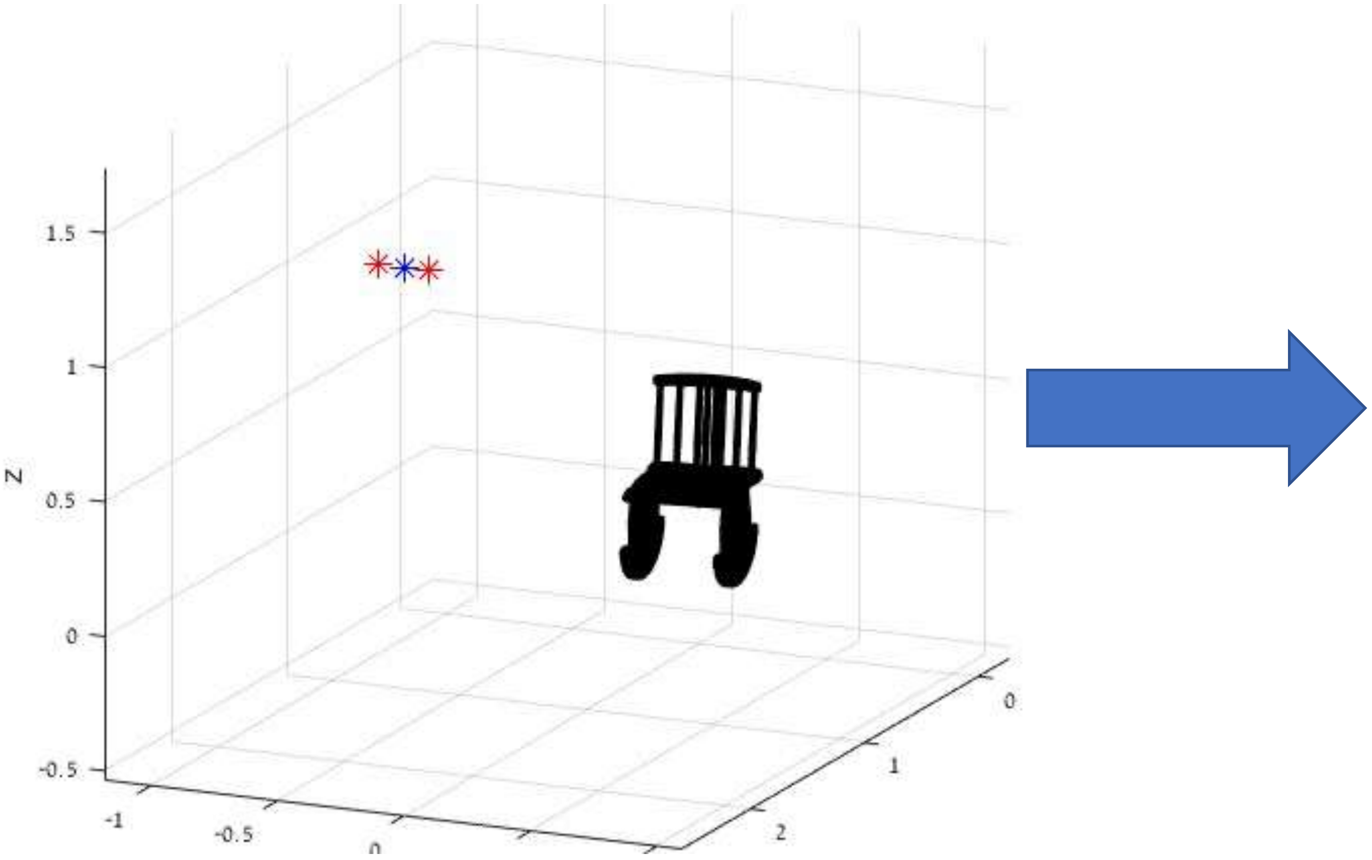
* Vision has advantages with corners, sonar with flat surfaces

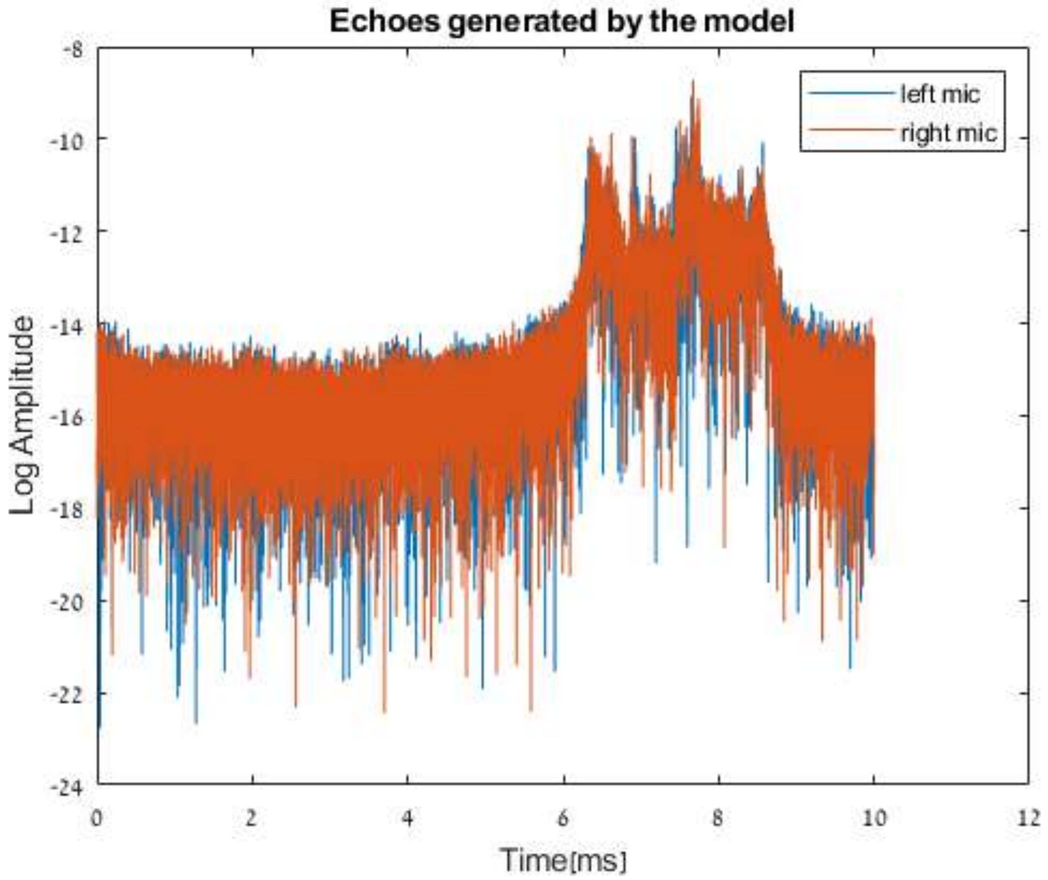
Joint work with Prof. Yossi Yovel, Netanel Frank, Danny Olshansky

Motivation



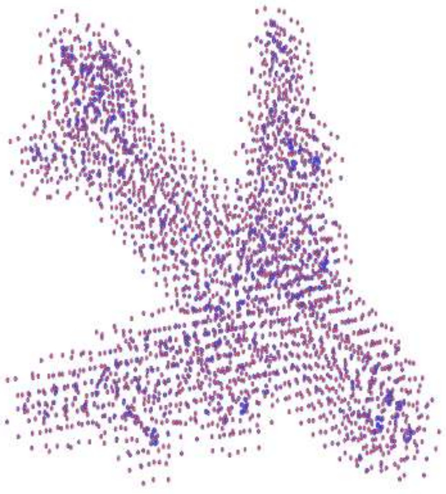
The Acoustic Simulator (similar to *A. Boonman at el. 19*)

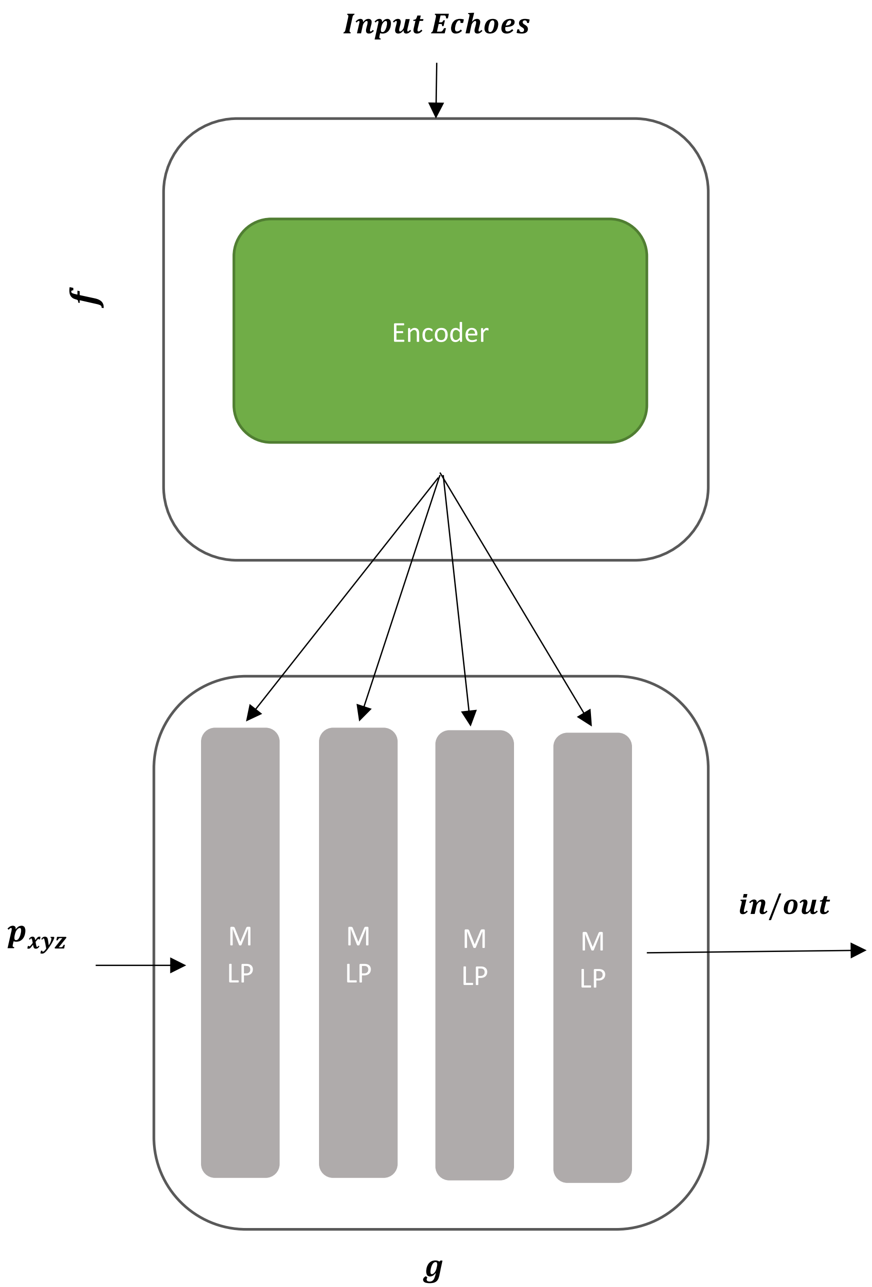




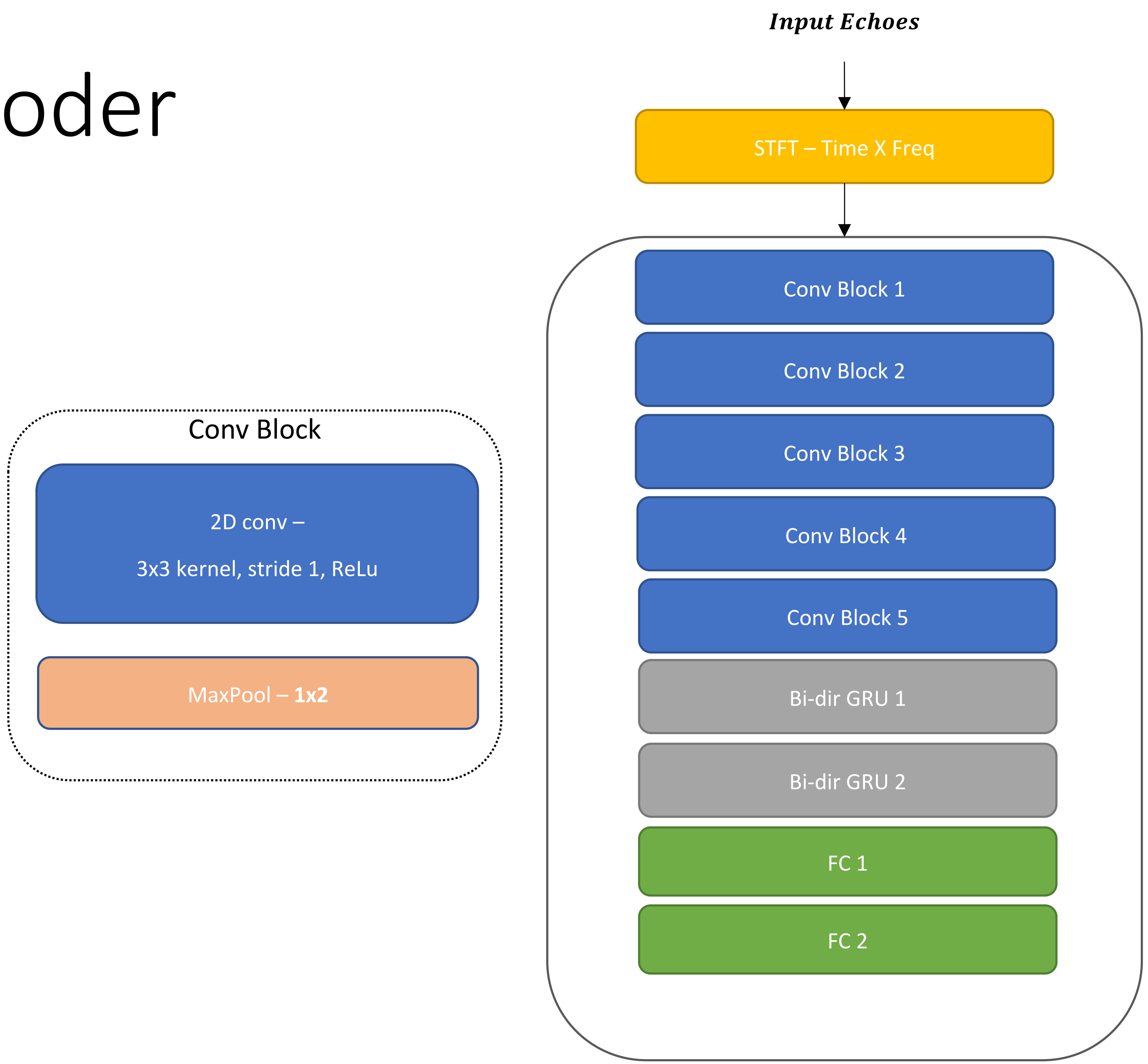
BatNet Meta Architecture

$$\theta_E = f(E, \theta_f)$$
$$s_E^p = g(p, \theta_E).$$





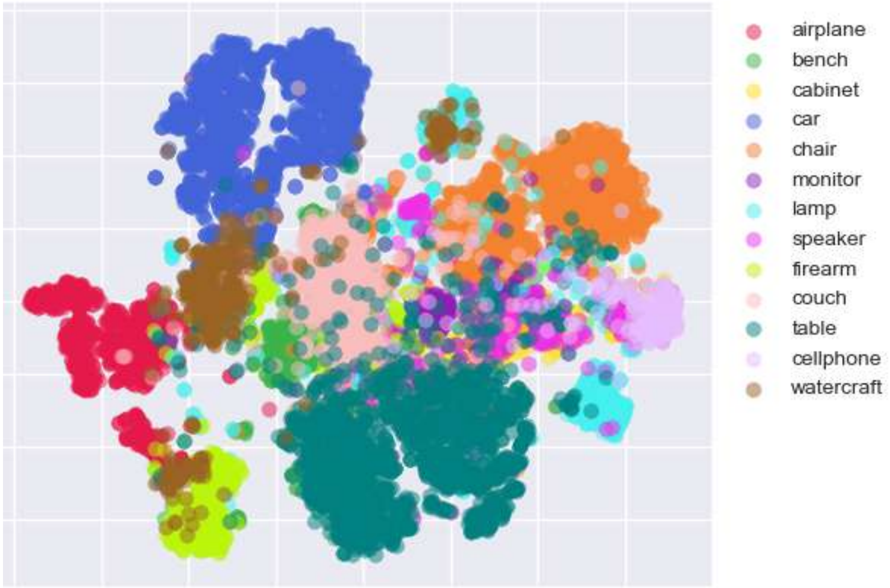
Encoder



Input Echoes
STFT – Time X Freq
Conv Block 1
Conv Block 2
Conv Block 3
Conv Block 4
Conv Block 5
Bi-dir GRU 1
Bi-dir GRU 2
FC 1
FC 2
Conv Block
2D conv –
3x3 kernel, stride 1, ReLu
MaxPool – 1x2

Class-level Results (IoU %)

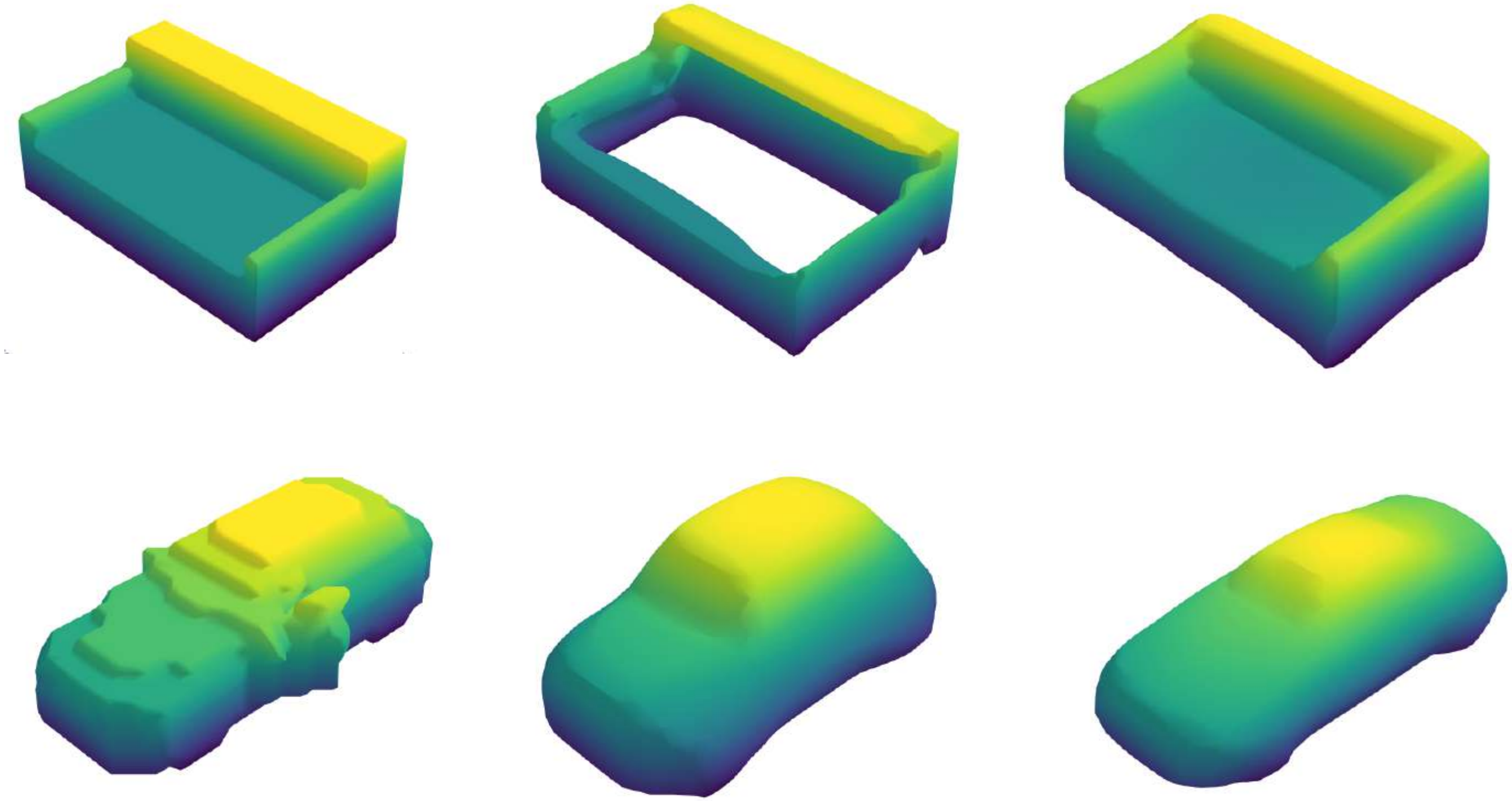
- Our BatNet model shows superior results in compare to all Echo-based architectures.
- BatNet is able to separate the 13 classes in an unsupervised manner.



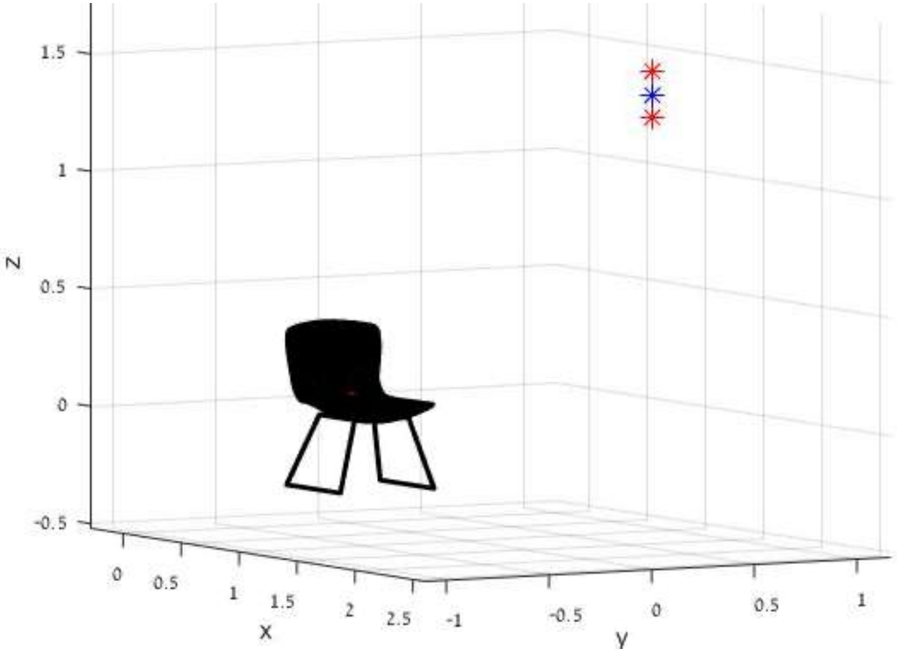
		Airplane	Bench	Cabinet	Car	Cellphone	Chair	Couch	Firearm	Lamp	Monitor	Speaker	Table	Watercraft	Mean
Vision	3D-R2N2 [Choy 16']	51.3	42.1	71.6	79.8	66.1	46.6	62.8	54.4	38.1	46.8	66.2	51.3	51.3	56.0
	Our vision work	71.4	65.9	79.3	87.1	79.1	60.7	74.8	68.0	48.6	61.7	73.8	62.8	65.4	69.1
Sonar	SoundNet - Encoder	48.9	24.3	52.6	75.1	60.2	29.9	50.0	46.7	21.5	37.3	46.4	29.8	37.9	43.1
	ResNet –Encoder	51.5	27.8	49.4	72.3	59.9	28.3	46.2	46.3	19.4	33.7	42.8	27.8	37.5	41.8
	ResNet+phase	51.7	38.9	58.5	78.9	63.2	37.8	55.9	53.9	29.6	37.9	53.0	38.9	46.1	49.6
	BatMan	56.4	41.1	60.4	74.5	71.7	38.4	58.9	56.1	29.7	52.2	52.3	36.1	47.7	52.0

Reconstruction examples

Original Vision Sonar

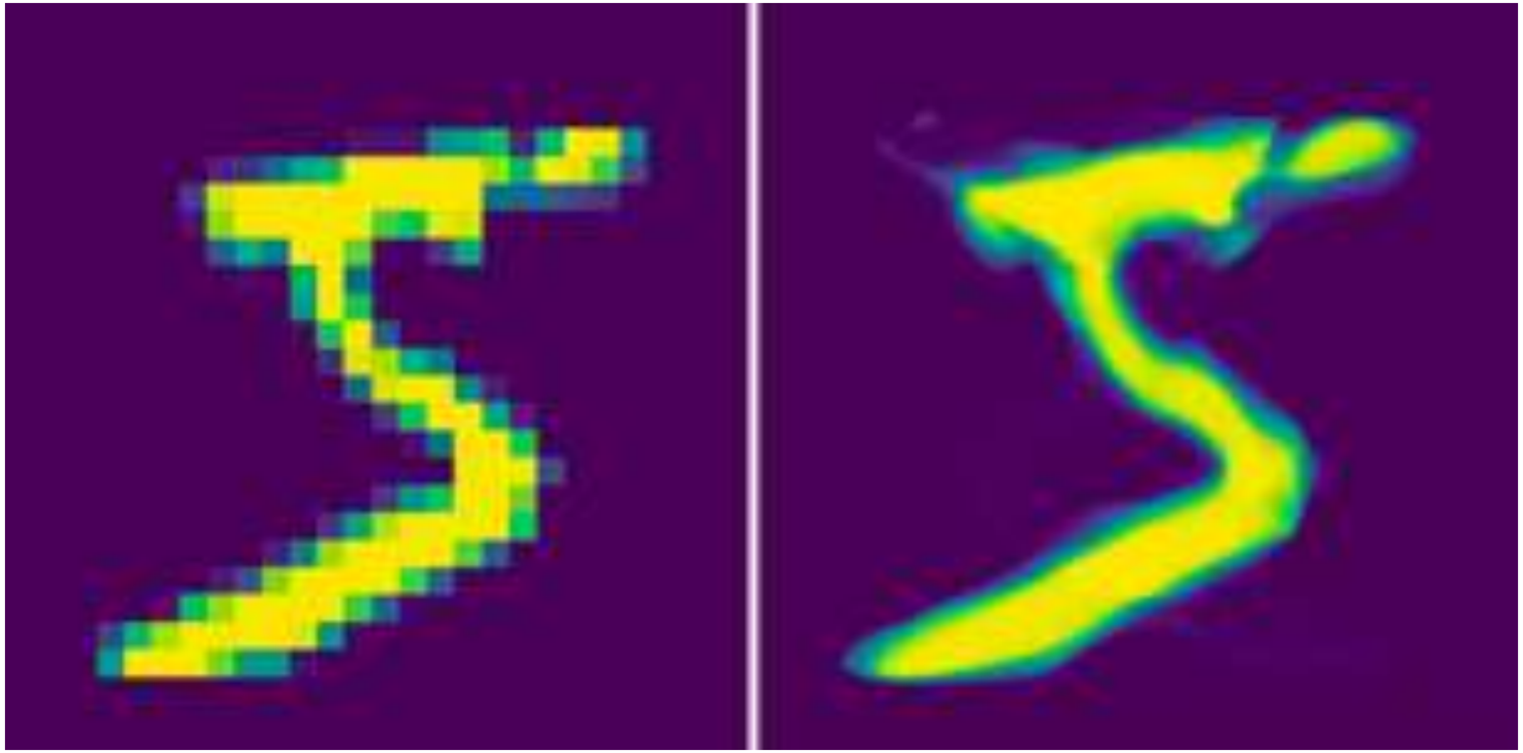


Single Viewpoint – multiple mics

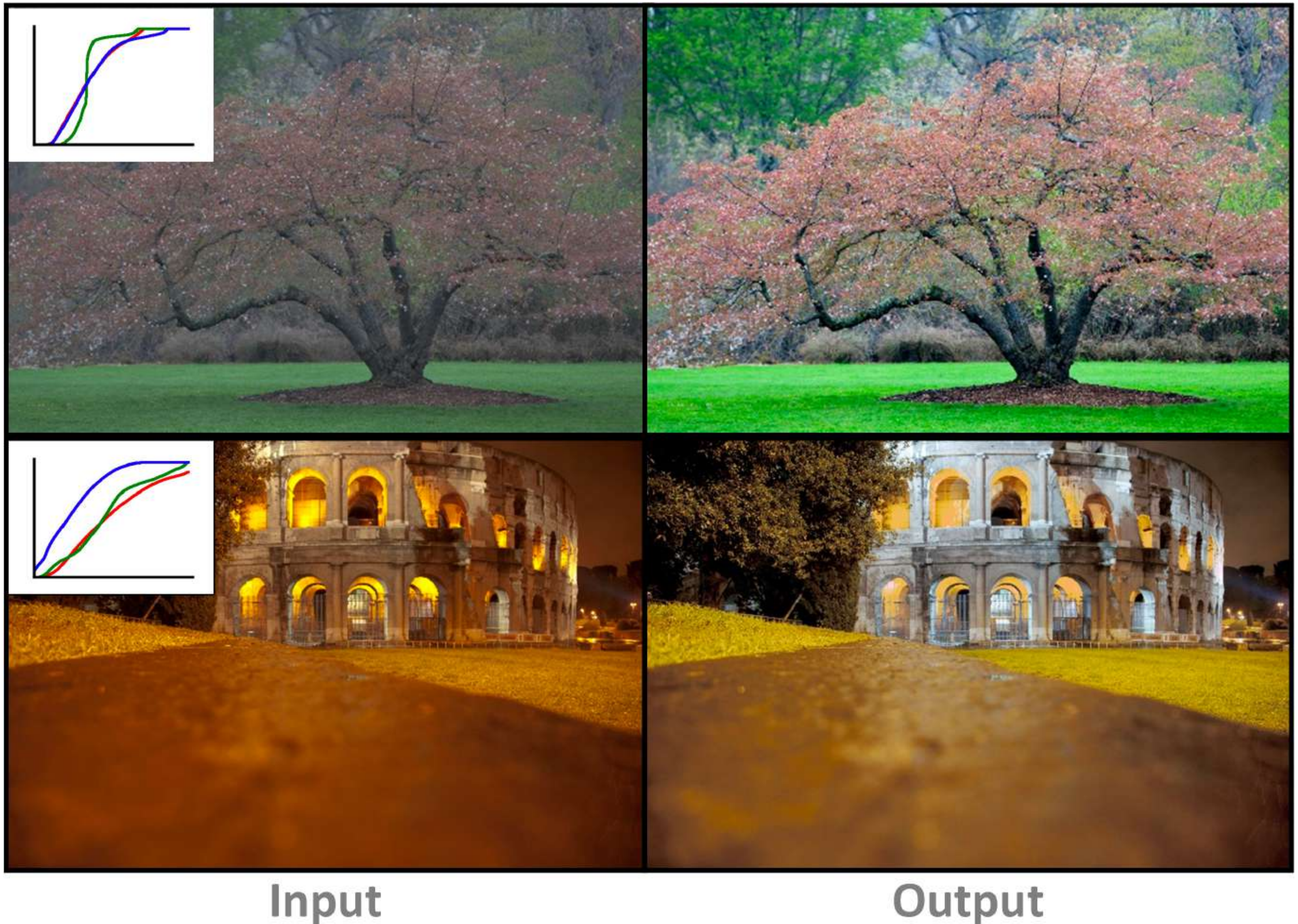


	Airplane	Bench	Cabinet	Car	Cellphone	Chair	Couch	Firearm	Lamp	Monitor	Speaker	Table	Watercraft	Mean
BatMan – single pair	56.4	41.1	60.4	74.5	71.7	38.4	58.9	56.1	29.7	52.2	52.3	36.1	47.7	52.0
BatMan – multiple pairs	61.9	49.7	67.4	78.2	76.9	46.3	65.2	62.4	37.2	55.8	61.8	48.3	57.3	59.1

MNIST SUPER RESOLUTION



“HYPERNETWORKS” FOR IMAGE RETOUCHING



Y. Chai, L. Wolf, R. Gyris. **Supervised and Unsupervised Learning of Parameterized Color Enhancement.** *WACV*, 2020.

SIMPLE COLOR TRANSFORMATION

Inflate:

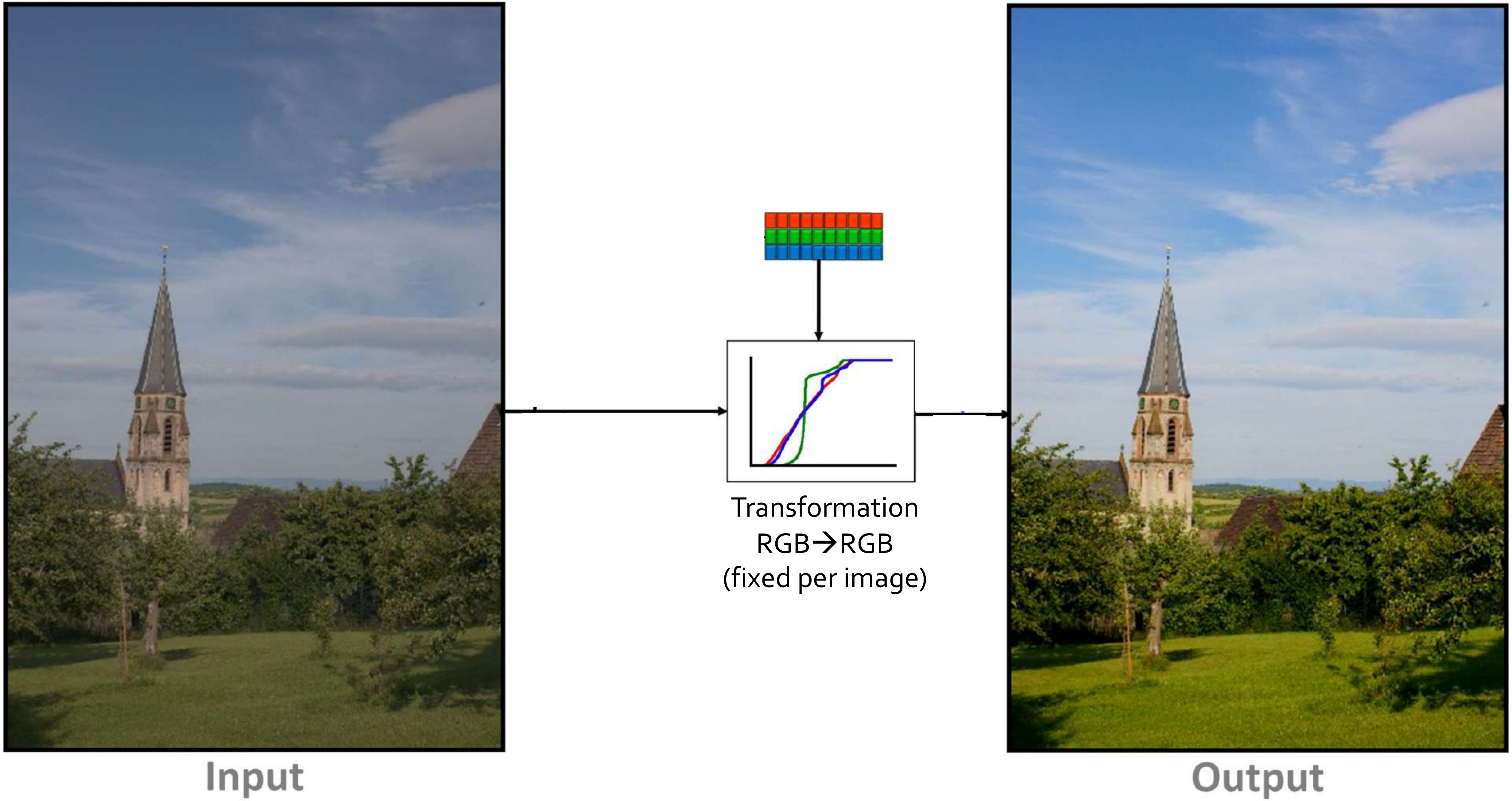
$$[R, G, B] \rightarrow [R^2, G^2, B^2, RG, RB, GB, R, G, B, 1]$$

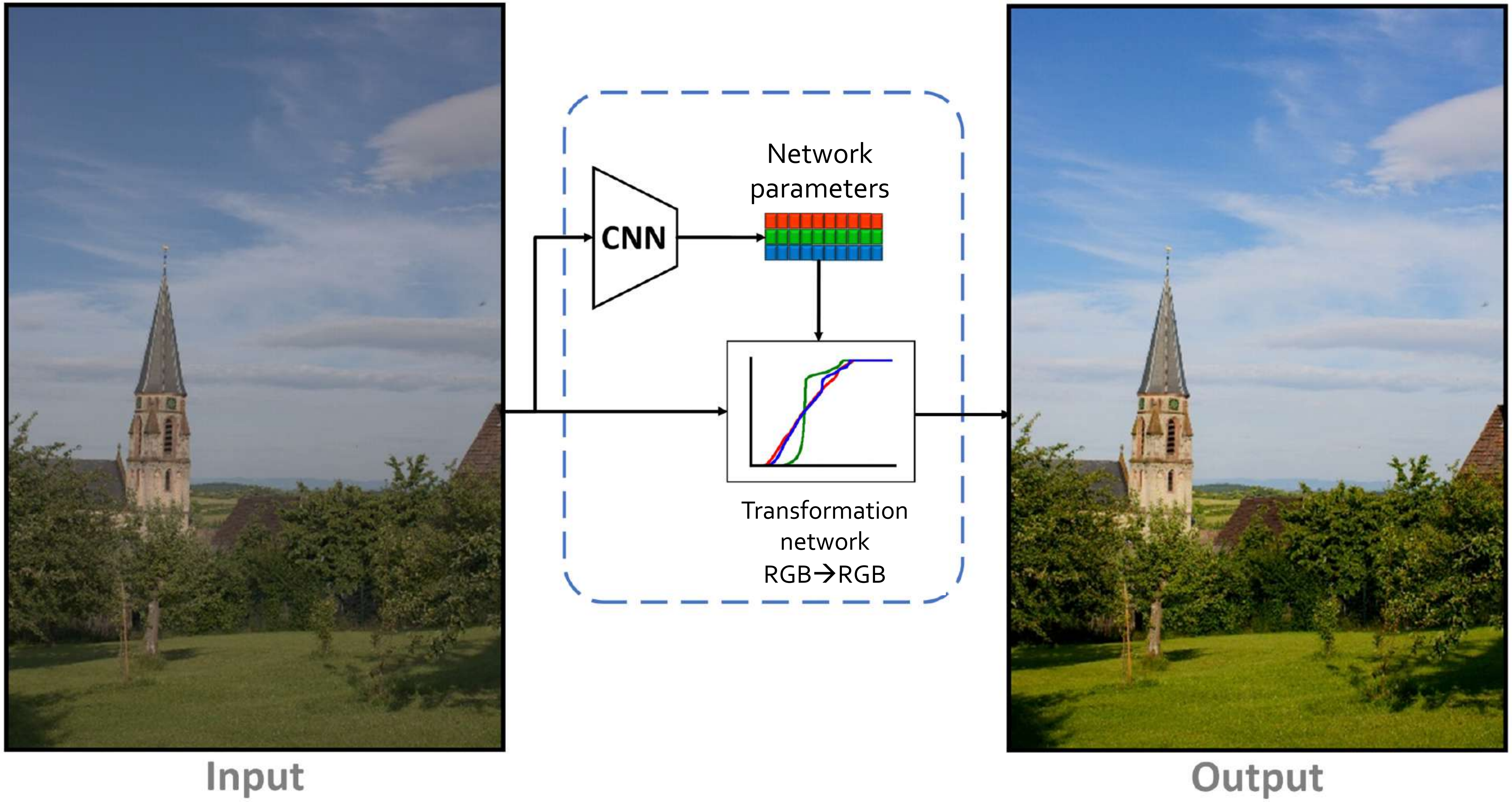
Transform:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} & \text{3x10 transformation matrix} & \end{bmatrix} \begin{bmatrix} R^2 \\ G^2 \\ B^2 \\ RG \\ RB \\ GB \\ R \\ G \\ B \\ 1 \end{bmatrix}$$

output color

input color


Transformation
RGB→RGB
(fixed per image)
Input
Output



CNN
Network
parameters
Transformation
network
RGB→RGB
Input
Output

'GAME OF THRONES' DARK EPISODE

TURN OF THE LAST CENTURY PHOTOGRAPHS

input ours - paired Adobe Lightroom

RECOMMENDATION SYSTEMS

For **each user**, we have a small training set and we predict her/his rating for new movies

Training Set

Movie	Avg. Rating	Genre	Rating
Dumbo	3.2	Fantasy	2
Die Hard	4.8	Action	4
300	4.2	Action	5
Titanic	4.7	Drama	3

Predictions

Movie	Prediction
Avatar	3.2
Stolen	2.7
Batman	4.7
Vice	3.2

E. Shulman, L. Wolf . **Explainable Recommendation Systems with Meta Trees**.

EXPLAINABLE RECOMMENDATION SYSTEMS

For **each user**, we have a small training set and we predict her/his rating for new movies

Training Set

Movie	Avg. Rating	Genre	Rating
Dumbo	3.2	Fantasy	2
Die Hard	4.8	Action	4
300	4.2	Action	5
Titanic	4.7	Drama	3

Predictions

Movie	Prediction	Why?
Avatar	3.2	High ratings, not action
Stolen	2.7	Low ratings
Batman	4.7	High ratings, action
Vice	3.2	High ratings, not action

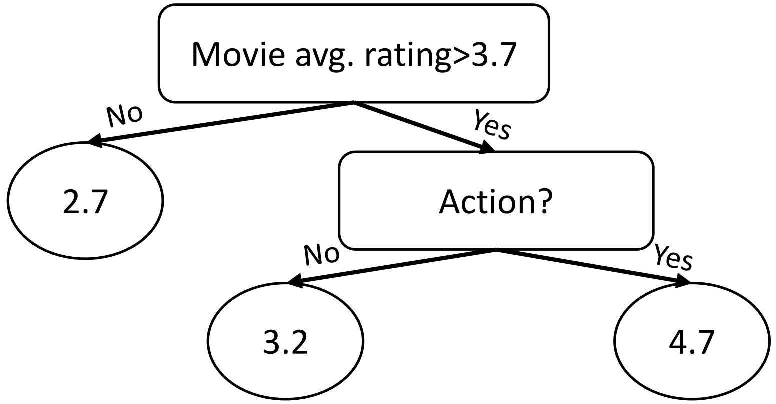
UNDERLING MODEL

For **each user**, we have a small training set and we predict her/his rating for new movies

Training Set

Movie	Avg. Rating	Genre	Rating
Dumbo	3.2	Fantasy	2
Die Hard	4.8	Action	4
300	4.2	Action	5
Titanic	4.7	Drama	3

User's Tree



Predictions

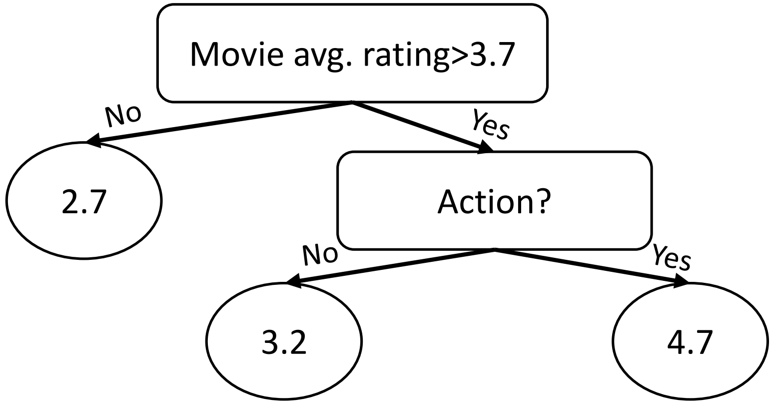
Movie	Prediction	Why?
Avatar	3.2	High ratings, not action
Stolen	2.7	Low ratings
Batman	4.7	High ratings, action
Vice	3.2	High ratings, not action

WORKING WITH LIMITED DATA

Training Set

Movie	Avg. Rating	Genre	Rating
Dumbo	3.2	Fantasy	2
Die Hard	4.8	Action	4
300	4.2	Action	5
Titanic	4.7	Drama	3

User's Tree



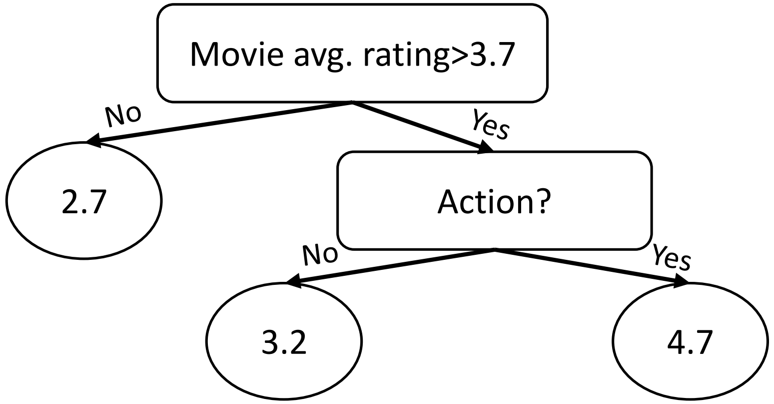
How can a learning algorithm learn from four samples?

META LEARNING FOR OVERCOMING LIMITED DATA

Training Set

Movie	Avg. Rating	Genre	Rating
Dumbo	3.2	Fantasy	2
Die Hard	4.8	Action	4
300	4.2	Action	5
Titanic	4.7	Drama	3

User's Tree



Instead of a tree learning algorithm
use meta learning
i.e., learn from building trees for many other users

THE META TREE EMPLOYS THREE NETWORKS: h, f and g

h is an embedding function:

$r_i = h(x_i, y_i)$

Embeddings are pooled over the training samples:

r$= \frac{1}{n}\sum_{i=1}^{n} h(x_i, y_i)$

$r_I = \frac{1}{|I|}\sum_{i \in I} h(x_i, y_i)$

THE META TREE EMPLOYS THREE NETWORKS: h, f and g

f returns the parameters of a node in the tree:

$$I_1 = \{1, \ldots, 10\}$$

$$\boxed{\boldsymbol{w_1}, \boldsymbol{b_1} = \boldsymbol{f}(\boldsymbol{r}, \boldsymbol{r_{I_1}})}$$

THE META TREE EMPLOYS THREE NETWORKS: h, f and g

f returns the parameters of a node in the tree:

$$I_1 = \{1, \dots, 10\}$$

$$\boxed{\boldsymbol{w_1}, \boldsymbol{b_1} = \boldsymbol{f}(\boldsymbol{r}, \boldsymbol{r}_{\boldsymbol{I_1}})}$$

${w_1}^T x < b_1$ | ${w_1}^T x \geq b_1$

THE META TREE EMPLOYS THREE NETWORKS: h, f and g

f returns the parameters of a node in the tree:

$I_1 = \{1, \dots, 10\}$

$\boxed{\boldsymbol{w_1}, \boldsymbol{b_1} = \boldsymbol{f}(\boldsymbol{r}, \boldsymbol{r}_{\boldsymbol{I_1}})}$

$w_1{}^T x < b_1$ $\qquad$ $w_1{}^T x \geq b_1$

$I_2 = \{1, \dots, 6\}$ $\qquad$ $I_3 = \{7, \dots, 10\}$

THE META TREE EMPLOYS THREE NETWORKS: h, f and g

f returns the parameters of a node in the tree:

$I_1 = \{1, \ldots, 10\}$

$$\boxed{\boldsymbol{w_1}, \boldsymbol{b_1} = \boldsymbol{f}(\boldsymbol{r}, \boldsymbol{r}_{\boldsymbol{I_1}})}$$

$w_1{}^T x < b_1$

$w_1{}^T x \geq b_1$

$I_2 = \{1, \ldots, 6\}$

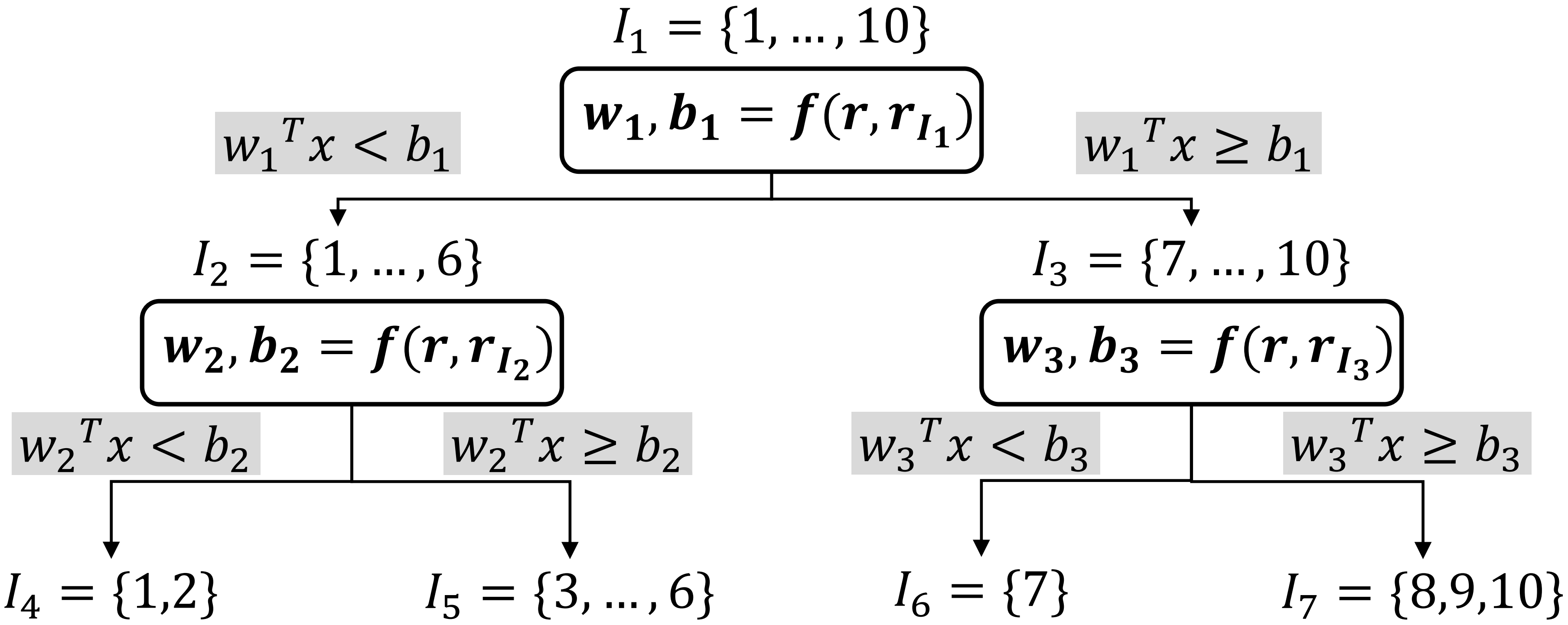
$$\boxed{\boldsymbol{w_2}, \boldsymbol{b_2} = \boldsymbol{f}(\boldsymbol{r}, \boldsymbol{r}_{\boldsymbol{I_2}})}$$

$I_3 = \{7, \ldots, 10\}$

$$\boxed{\boldsymbol{w_3}, \boldsymbol{b_3} = \boldsymbol{f}(\boldsymbol{r}, \boldsymbol{r}_{\boldsymbol{I_3}})}$$

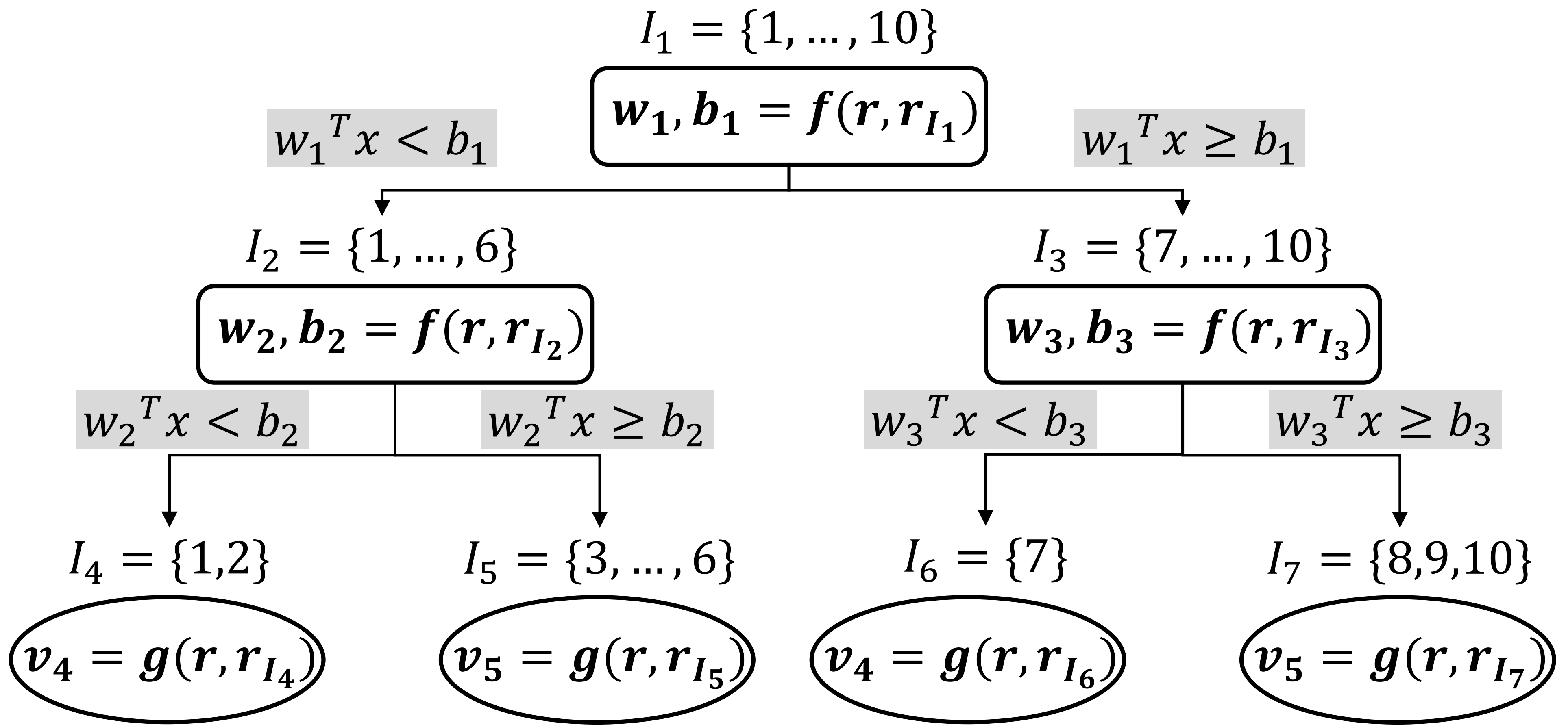
THE META TREE EMPLOYS THREE NETWORKS: h, f and g

f returns the parameters of a node in the tree:



THE META TREE EMPLOYS THREE NETWORKS: h, f and g

f returns the parameters of a node in the tree:
g returns the parameters of a leaf



THE RECURSIVE TREE BUILDING METHOD

Algorithm 1 The GrowTree procedure. To start the initial tree, we pass $I = [n]$

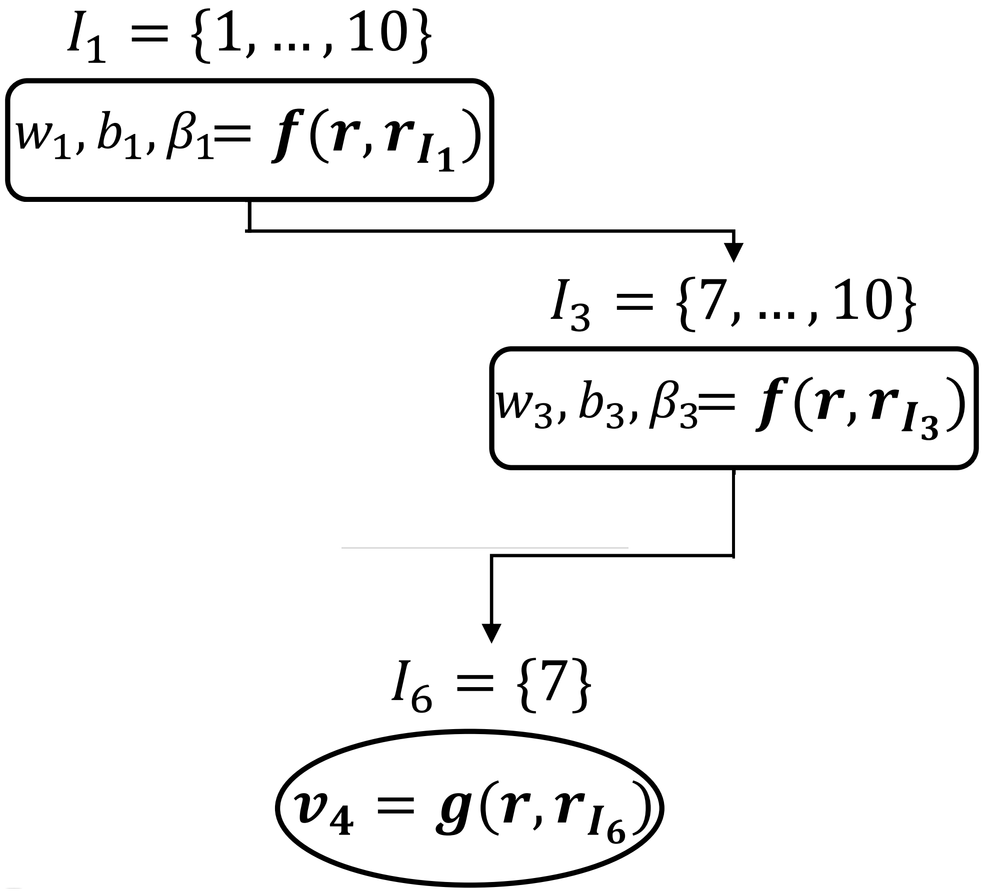
Input samples $L = \{(x_i, y_i)\}_{i=1}^{n}$, boolean isDynamic tree type, max depth d, I relevant indices
Output a decision tree t

$r \leftarrow \frac{1}{n}\sum_{i=1}^{n} h(x_i, y_i)$ ▷ r and $h(x_i, y_i)$ can be computed once
$r_I \leftarrow \frac{1}{|I|}\sum_{i \in I} h(x_i, y_i)$
if $d = 0$ **then** ▷ Check if building a leaf node or an inner node
    return $Leaf(g(r, r_I))$ ▷ The value of the leaf node
$w, b, \beta \leftarrow f(r, r_I)$ ▷ The parameters of the decision node
$I_l \leftarrow \{i \in I | \beta w^\top x_i < \beta b\}$
$I_r \leftarrow \{i \in I | \beta w^\top x_i \geq \beta b\}$
if isDynamic **and** $(I_l = \varnothing$ **or** $I_r = \varnothing)$ **then** ▷ For dynamic trees, stop at empty sets
    return $Leaf(g(r, r_I))$ ▷ The value of the leaf node
$child_l \leftarrow GrowTree(L, isDynamic, d-1, I_l)$
$child_r \leftarrow GrowTree(L, isDynamic, d-1, I_r)$
return $Node(w, b, \beta, child_l, child_r)$

TRAINING THE THREE NETWORKS OF THE METATREE

During training, the assignments to leafs are soft, based on the nodes along the path to the leaf:

$$\Pi_i \sigma(\beta_i(w_i^\top x - b_i))$$



TRAINING THE THREE NETWORKS OF THE METATREE

Each user *k* in the training set has a training dataset, which we split in two

$$L_k = \underbrace{L'_k}_{\text{used for building the tree } t_k} \cup \underbrace{L''_k}_{\text{used for computing the loss}}$$

TRAINING THE THREE NETWORKS OF THE METATREE

The MSE loss term

$$\mathcal{C}(K,h,f,g) = \sum_{(L'_k,L''_k)} \sum_{(x,y)\in L''_k} ||y - \sum_{z\in Z} t_k^{\text{soft}}(x,z)v_z||^2$$

sum over users in the training dataset

sum over loss evaluation samples

sum over leafs

the value of a leaf

$$t_k^{\text{soft}}(x,z) = \Pi_i \sigma(\beta_i(w_i^\top x - b_i))$$

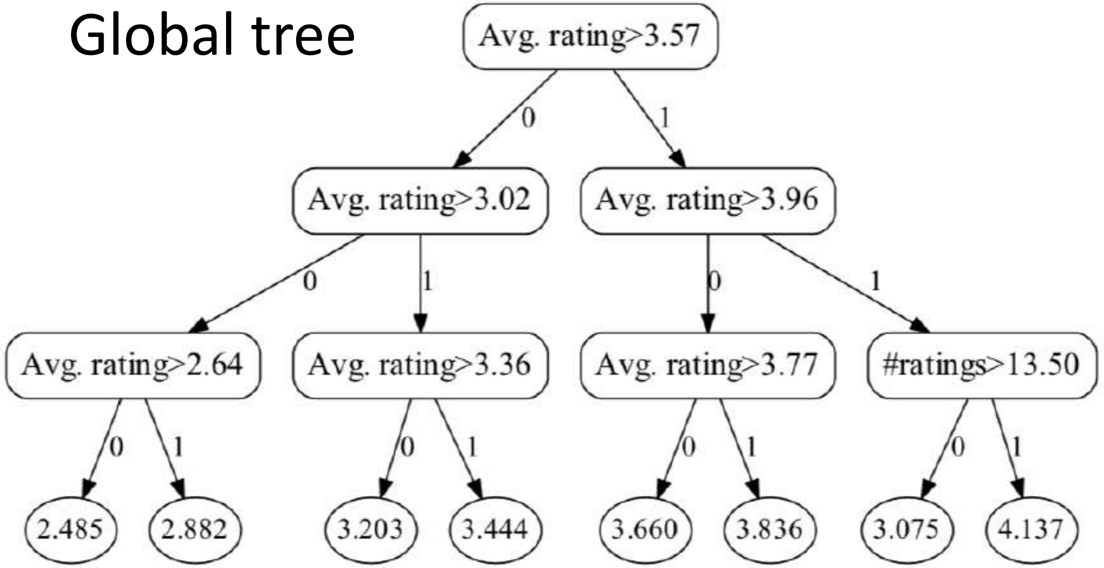
TRAINING THE THREE NETWORKS OF THE METATREE

The sparsity encouraging term, per tree *t*

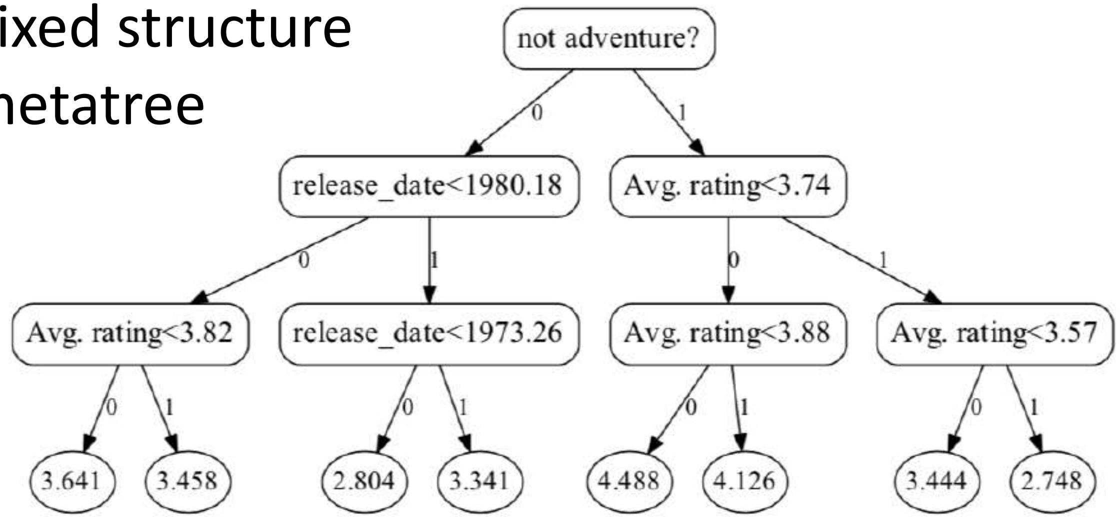
$$\mathcal{R}(t) = \frac{1}{|W(t)|} \underbrace{\sum_{w \in W(t)}}_{\text{sum over the hyperplanes of all nodes}} |ww^\top - \underbrace{D}_{\text{the diagonal operator}}(ww^\top)|_1$$

SAMPLE TREES - MOVIELENS-100K

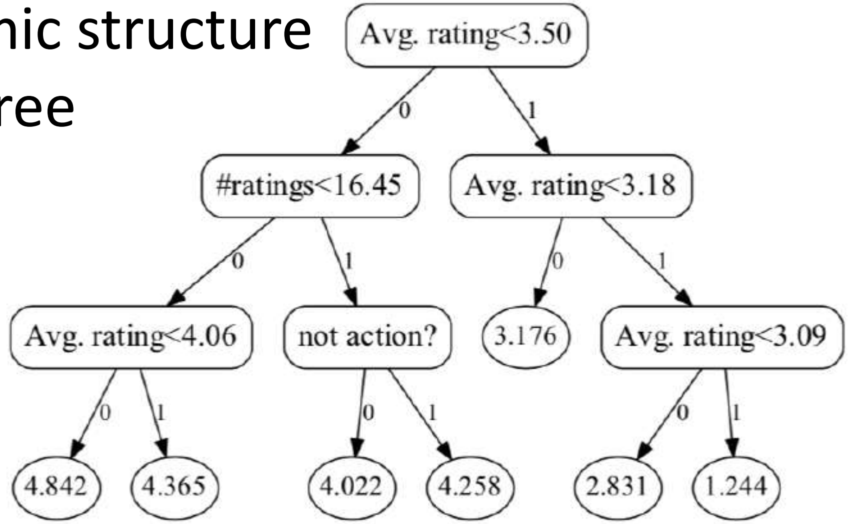
Global tree



Fixed structure metatree



Dynamic structure metatree

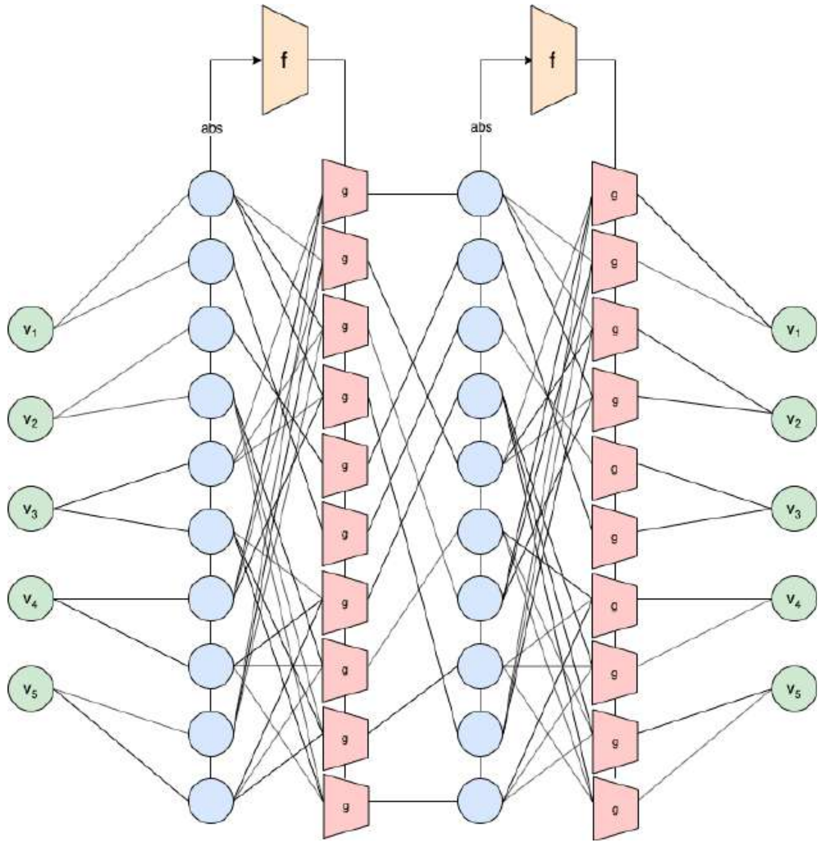


RESULTS

Table 1: Performance comparison on the MovieLens 100k, MovieLens 1M and Jester datasets.

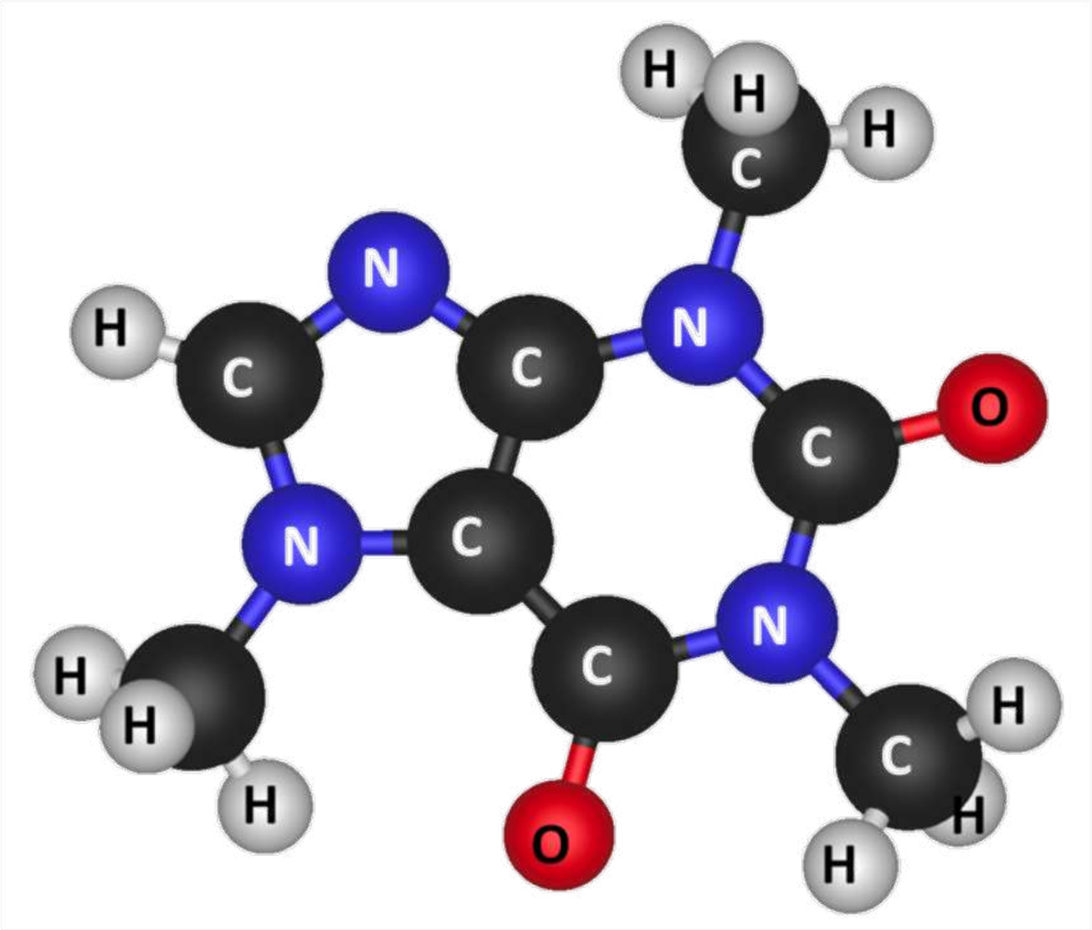
	MovieLens 100K		MovieLens 1M		Jester	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Global Tree (best depth)	0.995	0.778	0.935	0.738	4.302	3.136
Local Trees (best depth)	1.018	0.791	0.947	0.737	4.556	3.137
KNN Trees (best k)	0.975	0.770	0.921	0.726	4.161	3.070
Gradient boosted regression trees	0.976	0.771	0.933	0.736	4.119	3.131
SVD	0.953	0.751	0.868	0.682	4.073	2.987
SVD++	0.932	0.730	0.861	0.671	4.198	3.146
GRALS (Rao et al., 2015)	0.945	-	-	-	-	-
sRGCNN (Monti et al., 2017)	0.929	-	-	-	-	-
Factorized EAE (Hartford et al., 2018)	0.920	-	0.860	-	-	-
GC-MC (Berg et al., 2017)	0.905	-	0.832	-	-	-
Meta Trees (sparse, soft)	0.947	0.747	0.876	0.687	4.001	3.012
Meta Trees (sparse, hard)	0.970	0.766	0.916	0.722	4.131	3.030
Dynamic Meta Trees (sparse, soft)	0.948	0.747	0.872	0.683	4.008	3.062
Dynamic Meta Trees (sparse, hard)	0.975	0.767	0.914	0.720	4.171	3.097

STATE OF THE ART RESULTS IN BELIEF PROPAGATION FOR DECODING ERROR CORRECTING CODES



E. Nachmani, L. Wolf. **Hyper-Graph-Network Decoders for Block Codes**. *NeurIPS 2019*

STATE OF THE ART RESULTS IN GRAPH NEURAL NETWORKS FOR MOLECULE PROPERTY PREDICTION



E. Nachmani, L. Wolf. **Molecule Property Prediction and Classification With Graph Hypernetworks**

MODIFYING NMP-EDGES

The message passing scheme of the original NMP-Edge network takes the form:

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_w^t, e_{vw}^t), \tag{1}$$

$$h_v^{t+1} = S_t\left(h_v^t, m_v^{t+1}\right), \tag{2}$$

where m_v^t are the messages aggregated at node v at time t, and M_t, S_t are the message and transition networks for iteration t. These networks are dynamic (vary between iterations) but are independent of the inputs. The earlier work by Schütt et al. (2017b) uses a similar set of networks which do not change between the iterations.

In our modified network, we replace the state transition function with the hypernetwork f and g as follows:

$$\theta_g^t = f\left(c \cdot h_v^0 + (1-c) \cdot h_v^t\right) \tag{3}$$

$$h_v^{t+1} = h_v^t + g_{\theta_g^t}\left(m_v^{t+1}\right) \tag{4}$$

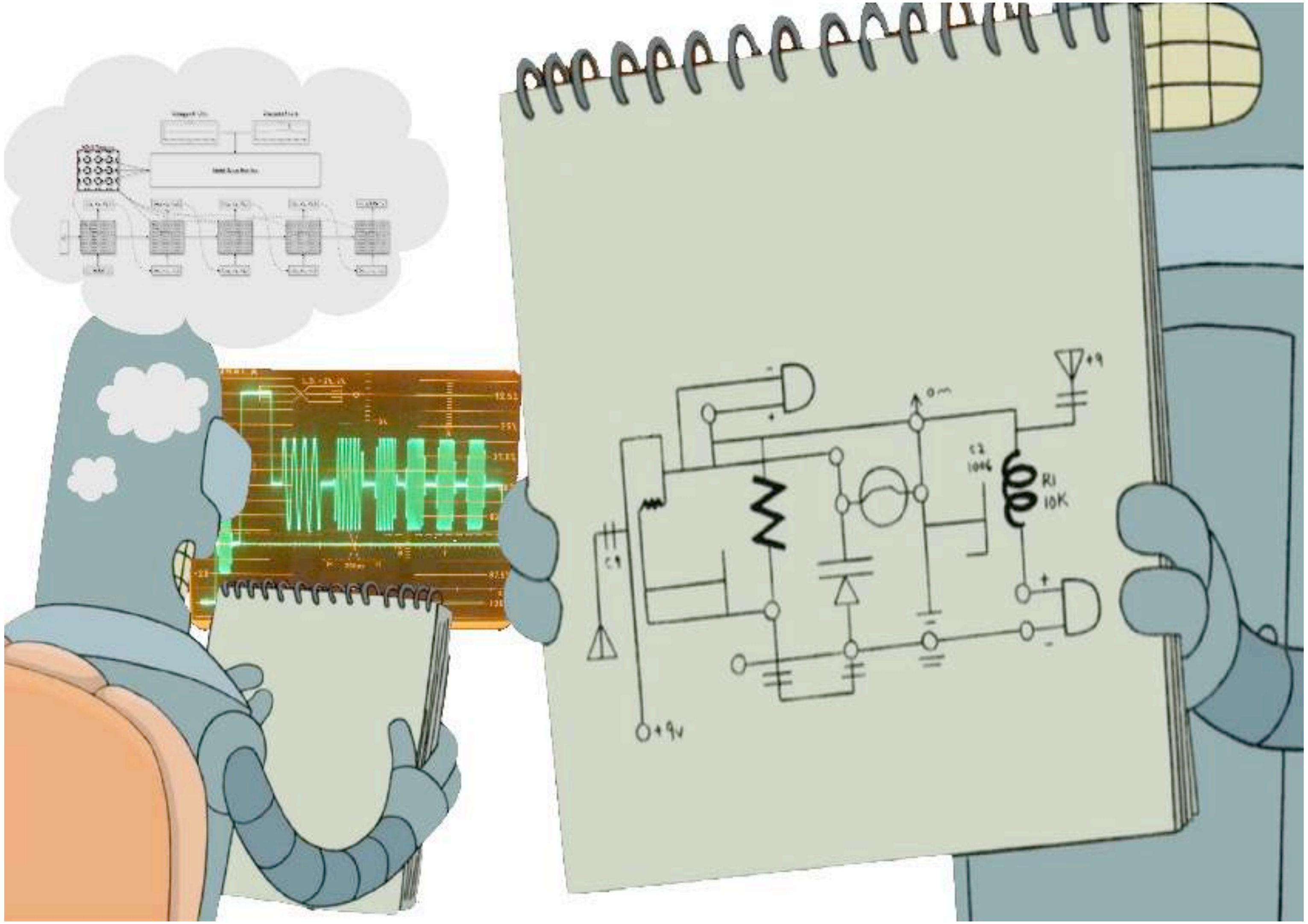
where c is a learned damping factor, which is clipped to be in the range [0,1] and is initialized with a uniform distribution, and the weights of network g are given by θ_g^t. For $t = 0$, Eq. 3 becomes $\theta_g^0 = f\left(h_v^0\right)$. Note that f is a fixed function. However, $g_{\theta_g^t}$ vary in time, since the set of weights θ_g^t change as the input to f changes.

ROBOT ENGINEERS

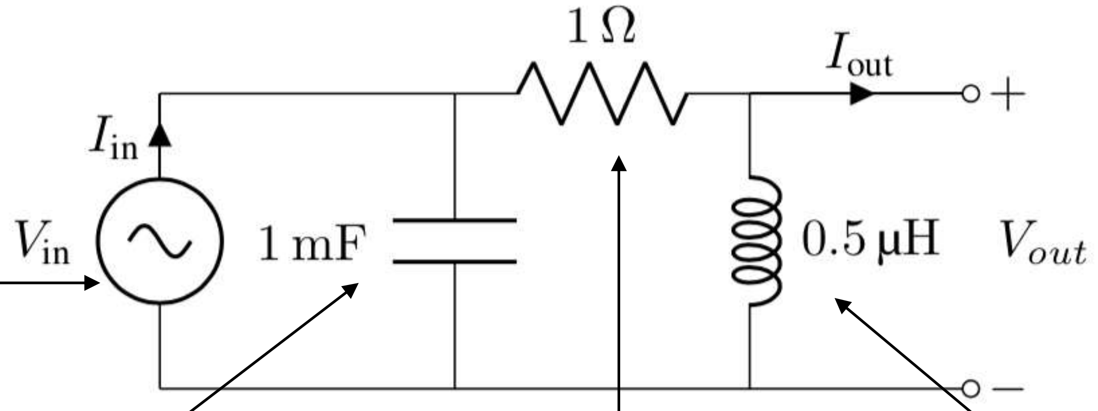
Engineering is about design

Input: desired specifications

Output: the plan



ELECTRIC CIRCUIT AS A SEQUENCE



Power supply

Output power

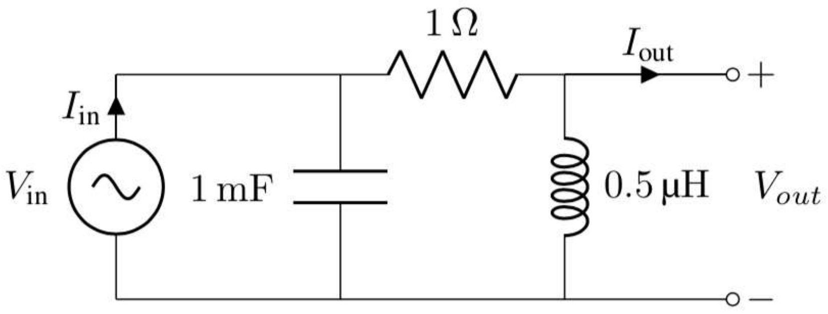
(parallel, capacitator, 1mF)

(series, resistor, 1Ω)

(parallel, inductor, 0.5μH)

M. Rotman, L. Wolf. **Electric Analog Circuit Design with Hypernetworks and a Differential Simulator.**

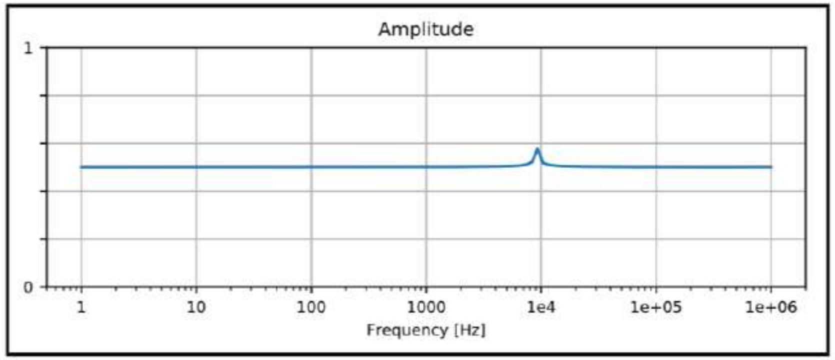
ELECTRIC CIRCUIT DESIGN



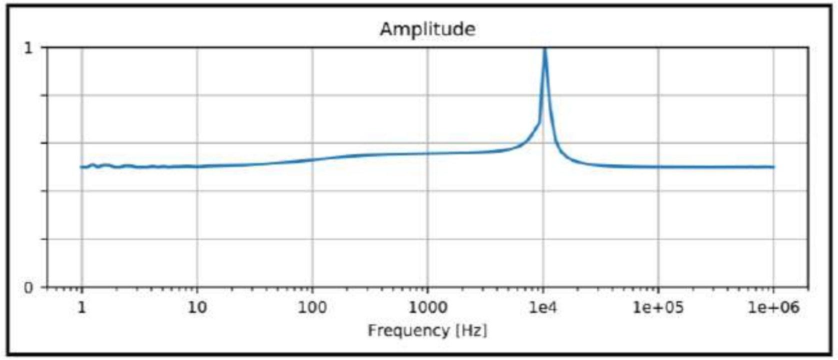
Easy

Hard

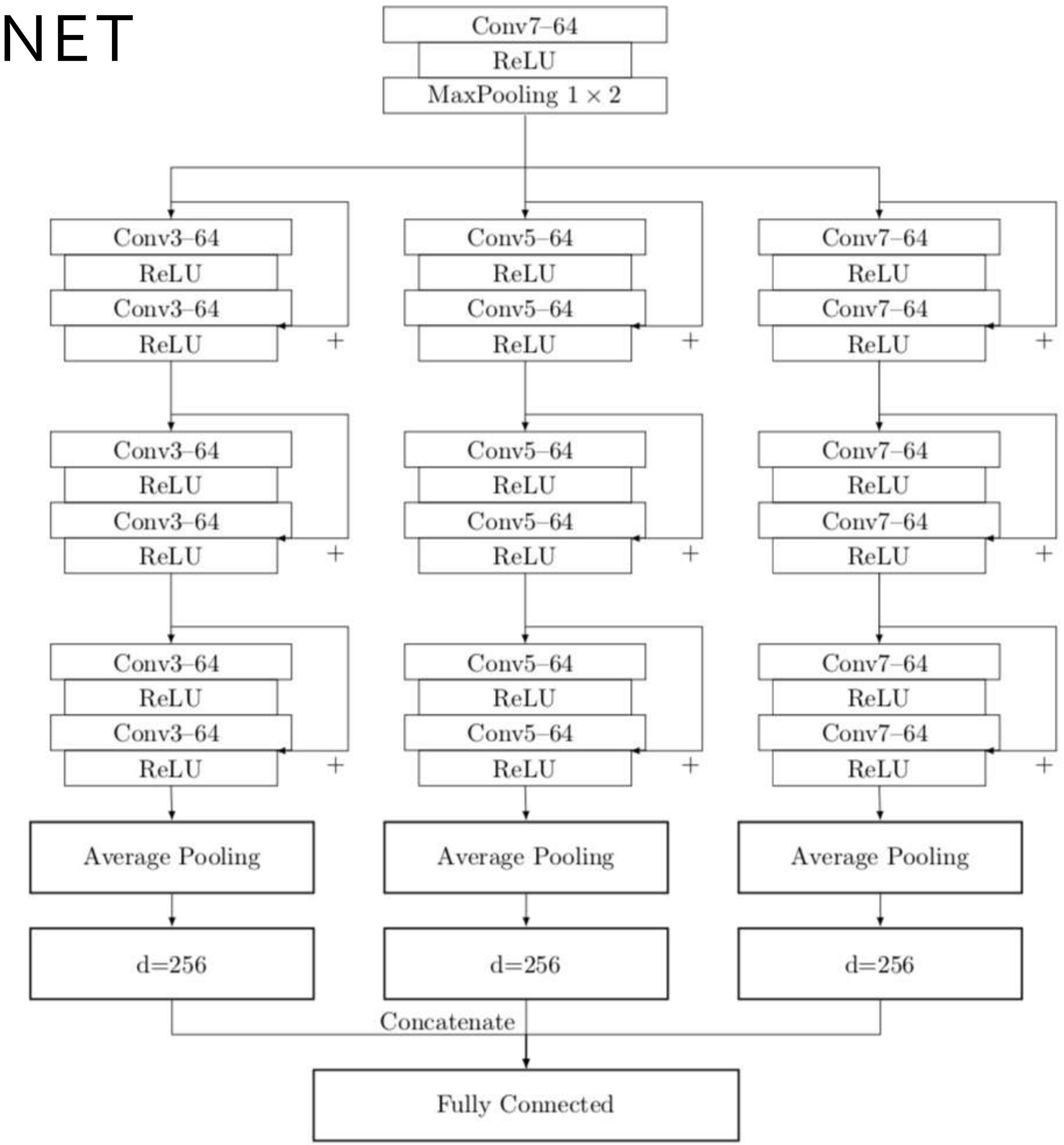
Voltage $(V(k))$



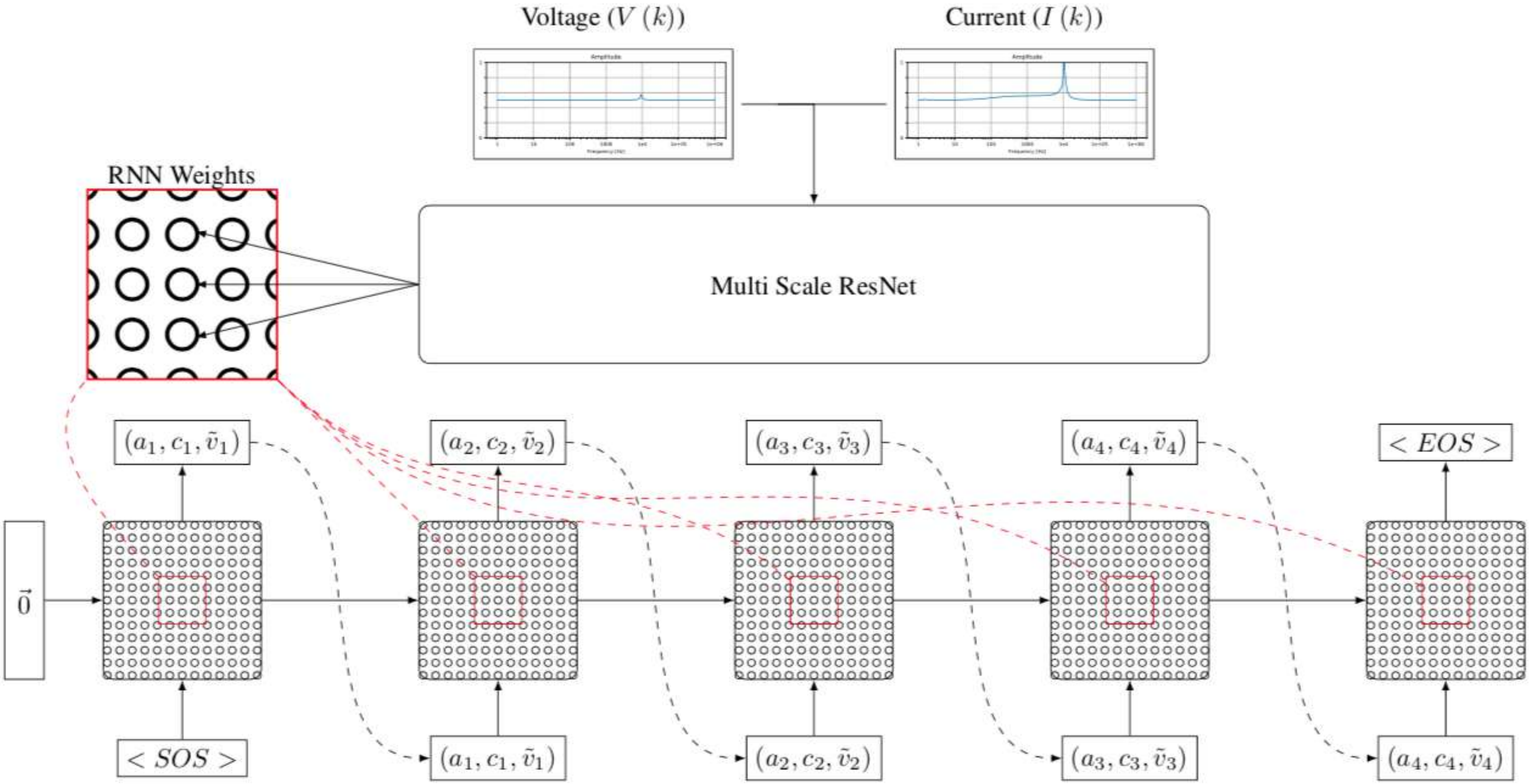
Current $(I(k))$



NETWORK F: MULTISCALE RESNET

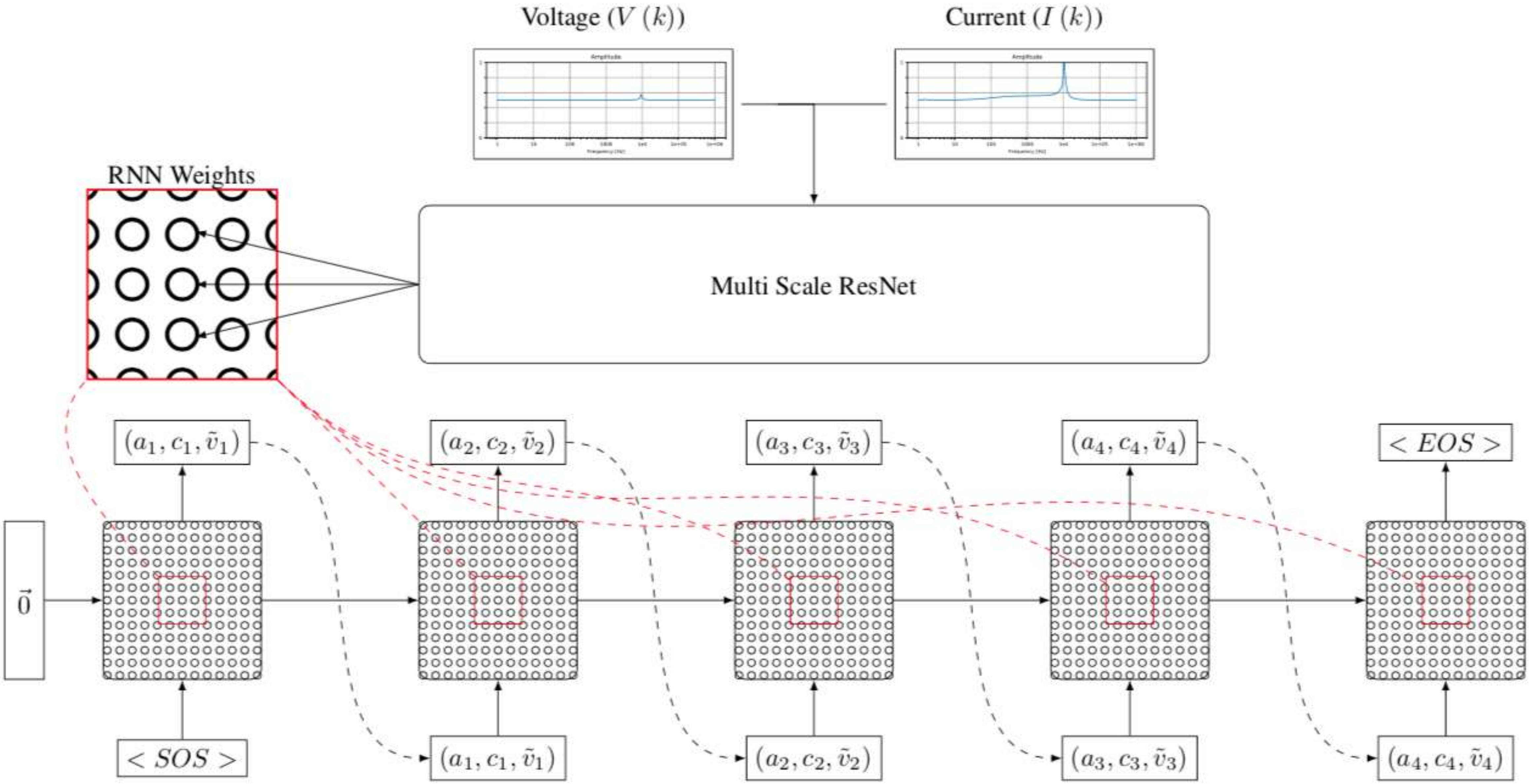


THE HYPERNETWORK



THE HYPERNETWORK

Trained with the cross entropy loss



DIFFERENTIAL SIMULATOR

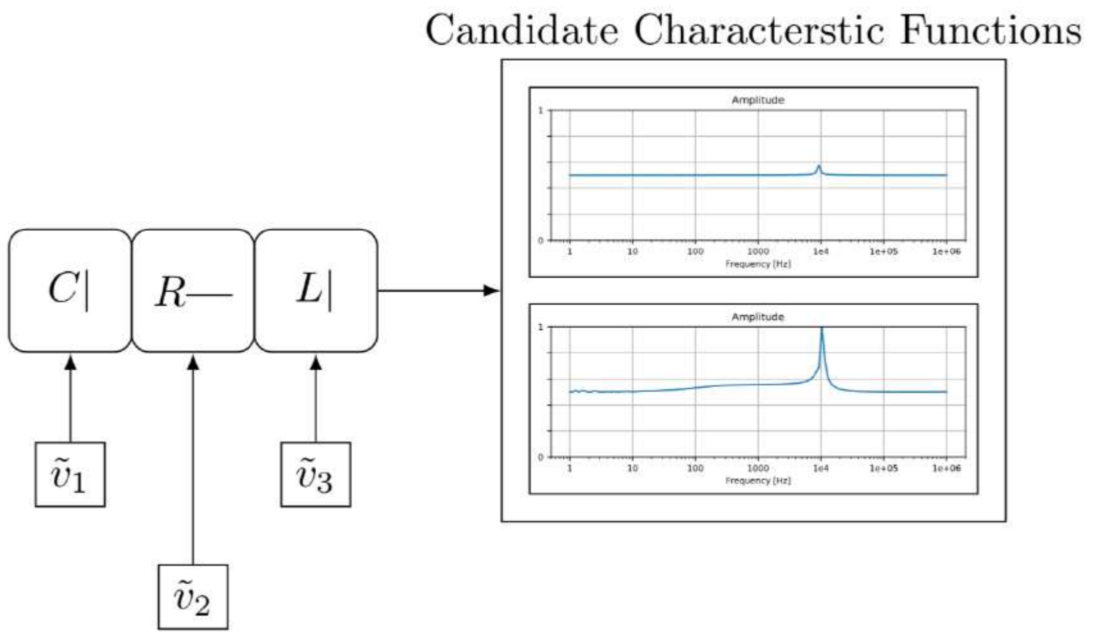
2D Clifford Algebra

$$a + ib = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}$$

A series of linear transformations

$$\begin{pmatrix} V_{out} \\ -I_{out} \end{pmatrix} = T_n \cdots T_1 \begin{pmatrix} V_{in} \\ I_{in} \end{pmatrix}$$

Alignment	Type	Matrix
Series	Resistor	$\begin{pmatrix} 1 & -R \\ 0 & 1 \end{pmatrix}$
Parallel	Resistor	$\begin{pmatrix} 1 & 0 \\ -\frac{1}{R} & 1 \end{pmatrix}$
Series	Conductor	$\begin{pmatrix} 1 & -\frac{1}{iC} \\ 0 & 1 \end{pmatrix}$
Parallel	Conductor	$\begin{pmatrix} 1 & 0 \\ -iC & 1 \end{pmatrix}$
Series	Inductor	$\begin{pmatrix} 1 & -iL \\ 0 & 1 \end{pmatrix}$
Parallel	Inductor	$\begin{pmatrix} 1 & 0 \\ -\frac{1}{iL} & 1 \end{pmatrix}$



Simulator objective

$$\mathcal{L}_{\mathcal{S}} = \frac{1}{d} \sum_{i=1}^{d} \left| V_i - \mathcal{V}\left(\overline{\mathcal{S}}\right) \right|^2 + \left| I_i - \mathcal{I}\left(\overline{\mathcal{S}}\right) \right|^2$$

RESULTS

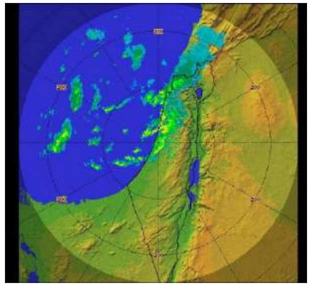
Table 1. A comparison for the classification accuracy over the test set for the different methods.

Length	Genetic Algorithm	GRU	GRU per length	Ours, GRU-only hypernet	Ours w/o simulator	Ours
4	0.58	0.01	0.30	0.07	0.30	0.32
5	0.28	0	0.08	0.05	0.41	0.42
6	0.09	0	0.02	0.04	0.50	0.51
7	0.03	0	0	0.01	0.46	0.46
8	0.01	0	0	0	0.44	0.44
9	0	0	0	0	0.42	0.42
10	0	0	0	0	0.42	0.43

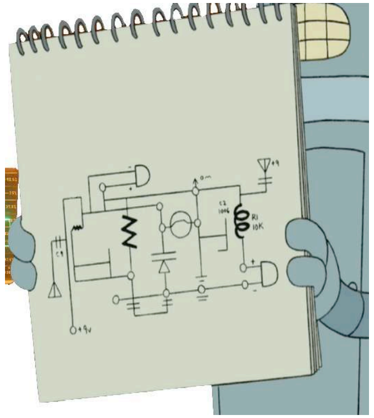
Table 2. A comparison for the classification accuracy over the test set for the different methods while ignoring the classification of the numerical values of the components.

Length	Genetic Algorithm	GRU	GRU per length	Ours, GRU-only hypernet	Ours
4	0.66	0.03	0.59	0.26	0.36
5	0.35	0	0.21	0.26	0.45
6	0.15	0	0.10	0.18	0.54
7	0.05	0	0.03	0.10	0.47
8	0.02	0	0	0.04	0.49
9	0	0	0	0.01	0.47
10	0	0	0	0	0.46

Environment and Earth Sciences

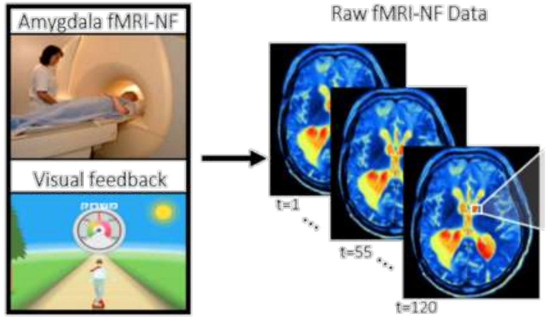


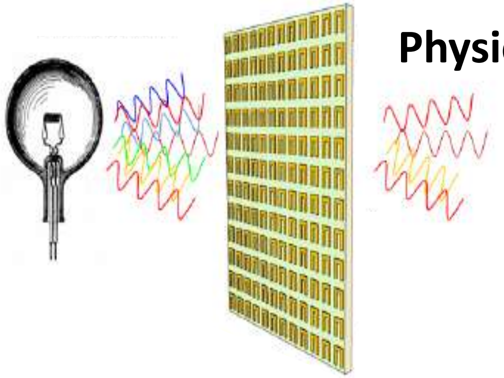
Zoology



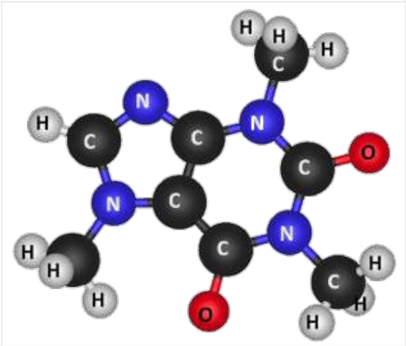
Electrical Engineering

Medicine

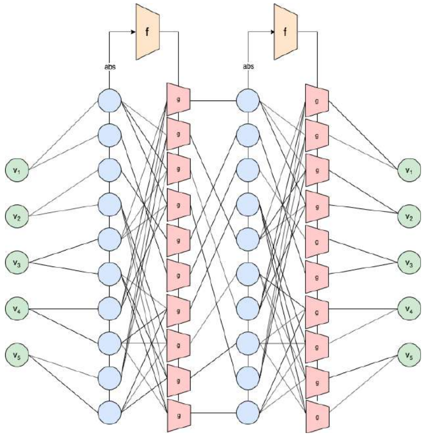




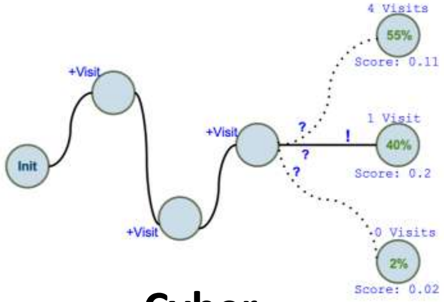
Physics



Bioinformatics and Chemistry



Electrical Engineering



Cyber