Learning neural implicit representations with surface signal parameterizations

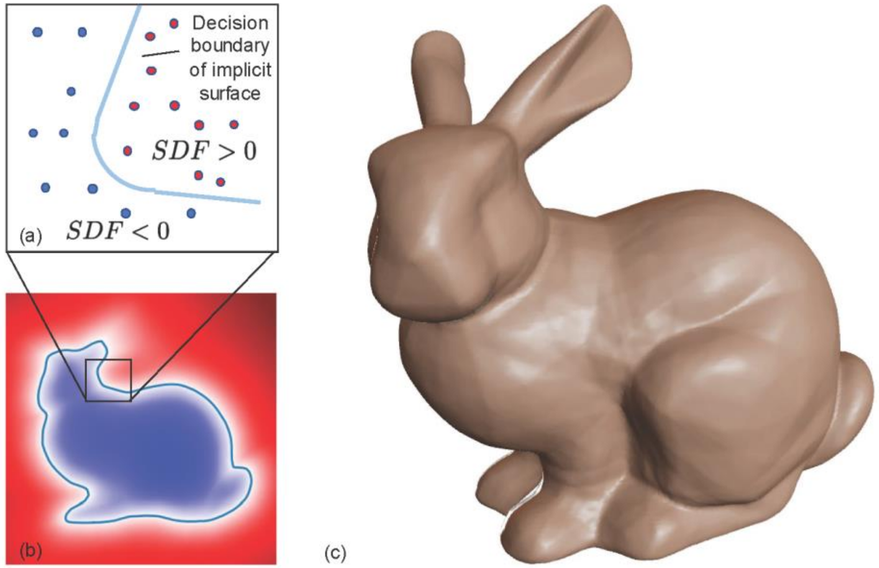
Yanran Guan[1,*], Andrei Chubarau[2,3], Ruby Rao[3], Derek Nowrouzezahrai[2]

[1]Carleton University, [2]McGill University, [3]Huawei Technologies Canada

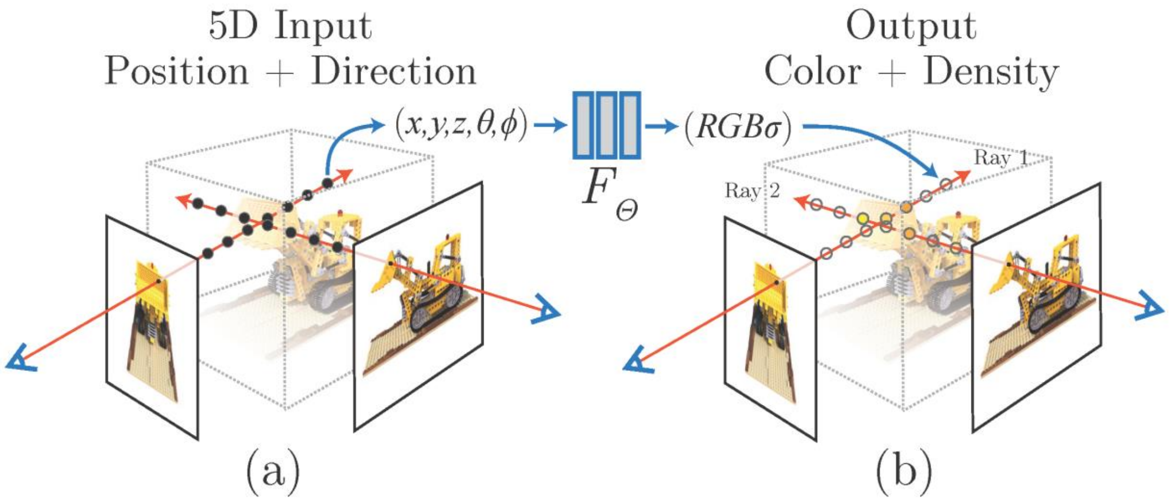
[*]Correspondence: yanran.guan@carleton.ca

How to texture a neural implicit surface?

Related work



Neural implicit surface [1]



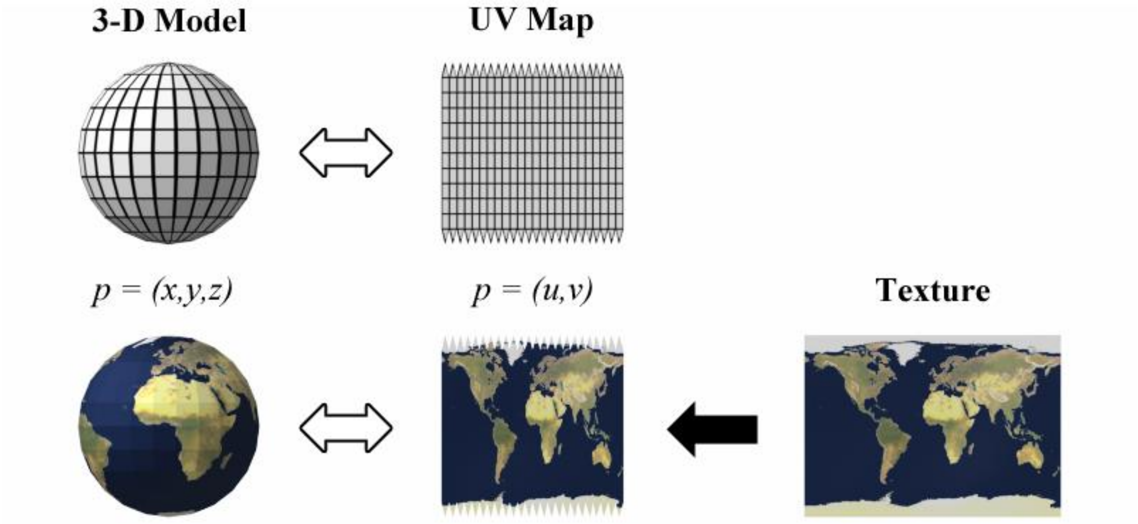
Neural implicit appearance [2]

Problem statement

- The neural implicit representations of object surfaces focus only on recovering the geometric details, instead of other appearance properties.
- The neural implicit appearance models tend to learn appearance properties that are entangled with the surface geometry, making it difficult to decouple the auxiliary appearance data from its underlying geometry, e.g., for texture editing.

Problem statement

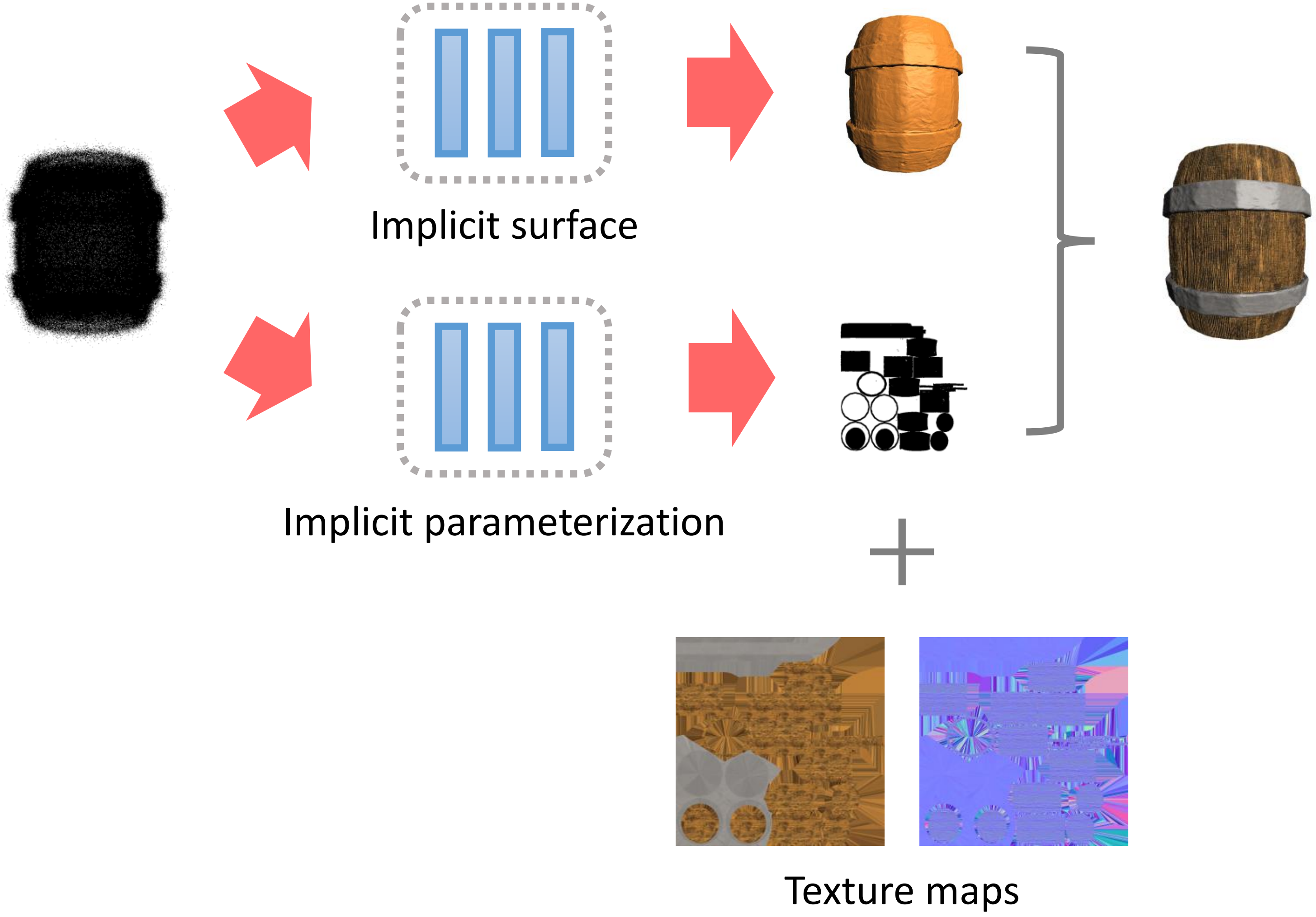
- Therefore, inspired by traditional explicit 3D object representations, i.e., meshes, we rely on surface parameterizations [3] to handle auxiliary appearance data.



Traditional 3D digital content: a mesh, a texture image, and a UV map (i.e., the parameterization)

Image source: https://commons.wikimedia.org/wiki/File:UVMapping.png

Overview



Implicit surface
Implicit parameterization
Texture maps

Importance sampling

- The signed distance function (SDF) describing an object's surface is $\mathrm{SDF}(\mathbf{p}) = d$, where $\mathbf{p} \in \mathbb{R}^3$ and $d \in \mathbb{R}$, and a neural network f_θ approximates the SDF, i.e., $f_\theta(\mathbf{p}) \approx \mathrm{SDF}(\mathbf{p})$.
- We assign higher weights to locations closer to the implicit defined surface, using the following metric:

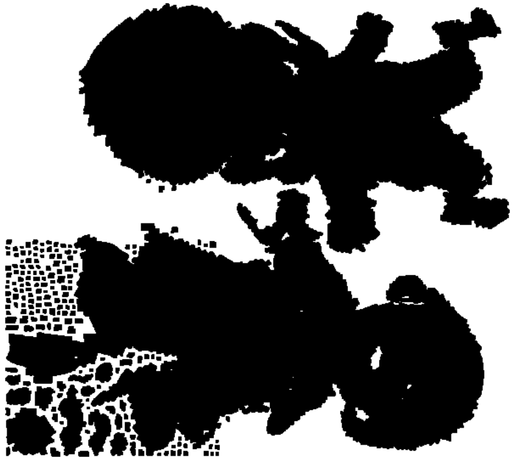
$$w(\mathbf{p}) = e^{-\beta|\mathrm{SDF}(\mathbf{p})|},$$

  where we use $\beta = 60$.
- We apply a Monte Carlo approximation to down-sample a set S of n points from a set of m uniformly sampled points U, such that:

$$\frac{1}{n}\sum_{p\in S}|\mathrm{SDF}(\mathbf{p}) - f_\theta(\mathbf{p})| \approx \frac{1}{m}\sum_{p\in U}|\mathrm{SDF}(\mathbf{p}) - f_\theta(\mathbf{p})|w(\mathbf{p}).$$

Decomposing the sampling space

- Object textures typically comprise a discontinuous collection of piecewise smooth parameterization charts, known as a texture atlas [3]. We observe that the discontinuities and the piecewise nature of these charts complicate learning, as neural networks tend to learn smoothed boundaries.

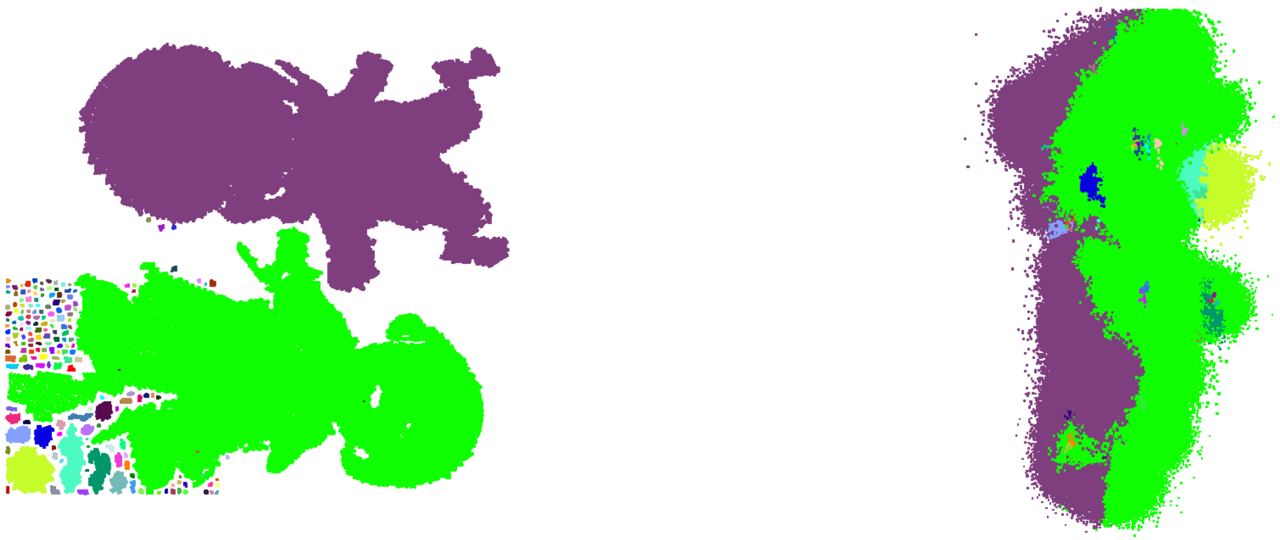


GT parameterization charts

Learned parameterization charts

Decomposing the sampling space

- Conditioned inputs help models learn complex structured representations [4].
- We assign a unique discrete label to each region of the parameterization charts and condition the input to our model on this decomposition signal.



Assign each sample point a label (right) of its nearest surface segment in the parameterization charts (left)

Pre-processing and layer implementation

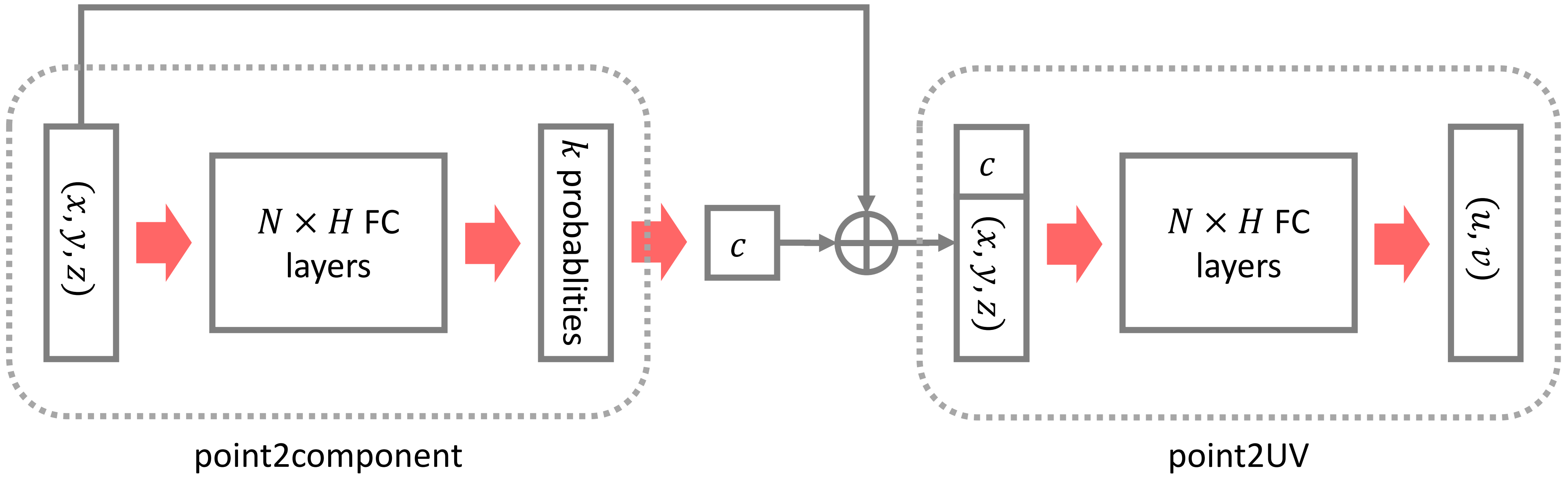
- We pre-process the input with Fourier positional encoding. Each value p in the coordinate position is encoded as:

$$\gamma(p) = (\sin 2^0 \pi p\,, \cos 2^0 \pi p\,, \ldots, \sin 2^{L-1} \pi p\,, \cos 2^{L-1} \pi p),$$

  where we use $L = 10$.

- We use sinusoidal representation network (SIREN) layers to implement the fully-connected (FC) layers of our model.

Network architecture



Experimental setup

- We use the model of Davies et al. [5] (OverfitSDF) to represent objects' geometry.
- We use 8×64 FC layers to implement the models.
- Each object is sampled with 10^6 points for training and our models are trained until convergence for 2000 epochs, using the Adam optimizer with a learning rate of 5×10^{-4}.
- A sphere tracer is used for visualization.
- The networks are trained on an NVIDIA GeForce RTX 2070 SUPER GPU with 8 GB of memory.

Diffuse mapping

OverfitColor

Ours

GT

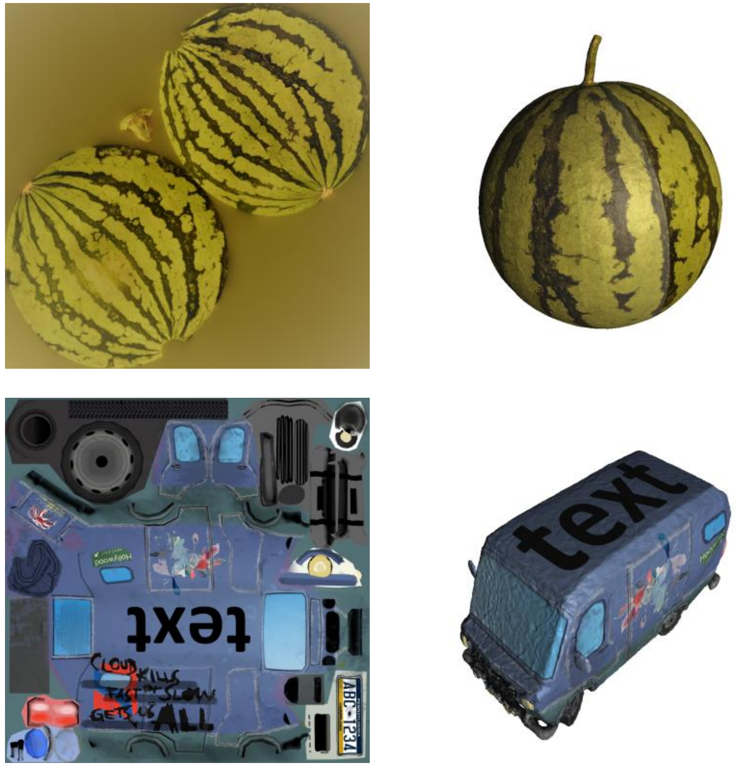
Normal mapping

Without normal mapping (OverfitSDF only)

With normal mapping

Texture editing

Original diffuse maps



Edited diffuse maps

Comparison to neural appearance models

SRN [6] NeRF [2] Ours GT

Model compression

- We interpret our model along with OverfitSDF as a compression strategy for large 3D objects represented as meshes. The average size of the OBJ files used in our experiments is 13.4 MB, while the weights of the networks representing the geometry and surface parameterization total to 487 kB on average, yielding a compression rate of $1{:}\,27$.
- We can apply a pruning algorithm to our model, e.g., using the lottery ticket method [7], which further removes $60{\sim}80\%$ of the weights from our model.

Conclusion

- We propose to learn neural representations of both an object’s 3D surface and a surface parameterization.
- The proposed neural representation extends the capability of (neural) implicit surfaces to enable various applications of texture mapping similar to standard 3D digital content creation pipelines.
- We overfit compact neural networks to single objects as their efficient weight-encoded representations, the networks can be further compressed through a pruning algorithm.

References

[1] Park, JJ, Florence, P, Straub, J, Newcombe, R, Lovegrove, S. DeepSDF: Learning continuous signed distance functions for shape representation. In: CVPR 2019, p. 165–174.

[2] Mildenhall, B, Srinivasan, PP, Tancik, M, Barron, JT, Ramamoorthi, R, Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. In: ECCV 2020, p. 405–421.

[3] Sheffer, A, Praun, E, Rose, K. Mesh parameterization methods and their applications. Found Trends Comput Graph Vis 2006;2(2):105–171.

[4] Sohn, K, Lee, H, Yan, X. Learning structured output representation using deep conditional generative models. In: NIPS 2015, p. 3483–3491.

[5] Davies, T, Nowrouzezahrai, D, Jacobson, A. On the effectiveness of weight-encoded neural implicit 3D shapes. CoRR 2020;abs/2009.09808.

[6] Sitzmann, V, Zollhöfer, M, Wetzstein, G. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In: NIPS 2019, p. 1121–1132.

[7] Frankle, J, Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In: ICLR 2019.

Thank you for your attention!