CAPNEXT: UNIFYING CAPSULE AND RESNEXT FOR MEDICAL IMAGE SEGMENTATION

Thanh M. Huynh[†] [⋆] *Chanh DT Nguyen*[†] [‡] [⋆] *Khoa N. A. Nguyen*[†] *Trung Bui* *Steven Q.H.Truong*[†]

[†] VinBrain JSC. [‡] VinUniversity.
[⋆] Equal contribution

ABSTRACT

Capsule Network is a contemporary approach to image analysis that emphasizes part-whole relationships. However, its applications to segmentation tasks are limited due to training difficulties such as initialization and convergence. In this study, we propose a novel Capsule Network, called CapNeXt, that unifies Capsule and ResNeXt architectures for medical image segmentation. CapNeXt advances the existing capsule-based segmentation model by integrating optimization techniques from Convolutional Neural Networks (CNN) to make training much easier than other contemporary Capsule-based segmentation methods. Experimental results on two public datasets show that CapNeXt outperforms the CNNs and other Capsule architectures in 2D and 3D segmentation tasks by 1% of the Dice score. The code will be released on GitHub after being accepted.

***Index Terms*—** Capsule Network, Medical Image Segmentation, ResNeXt Architecture

1. INTRODUCTION

Medical image segmentation algorithms attempt to extract and localize organs, tumors, and other structures of interest at pixel level accuracy, to aid health professionals in making accurate diagnoses more efficiently. Existing state-of-the-art medical image segmentation methods mainly use deep convolutional neural networks (CNNs) [1, 2]. Although deep CNNs have proven to be very good feature extractors, they have several limitations, such as lacking robustness against affine transformations and losing relevant spatial information of pixels due to max-pooling layers.

An alternative to CNNs was recently proposed by Sabour et al. [3], called Capsule network. In this network, Capsules replaces neurons as the building blocks. Unlike neurons, Capsule output is a vector representing an object's specific characteristics. Unlike CNNs, Capsule networks do not utilize max-pooling layers but make use of either an iterative routing-by-agreement algorithm [3] or Expectation-Maximization (EM) routing algorithm [4] that determines the coupling of Capsules between consecutive layers. Therefore, the networks emphasize the preservation of part-whole relationships in the data. It was shown that a shallow Capsule network outperforms a deep CNN on the MNIST dataset [3]. Furthermore, Capsule networks have fewer parameters and require less training data by learning viewpoint invariant feature representations [4].

More recently, several works have expanded Capsule networks to semantic segmentation and applied them to medical image segmentation. Generally, Capsule networks are well-suited to solve medical image segmentation tasks where the objects are structured. Segcaps [5] replaces the feature vector at each pixel in a feature map with a Capsule matrix where the number of rows is equal to the number of the Capsule, and the number of columns is equal to the number of Capsule features vectors. All Capsule features in SegCaps transform the same going from one layer to the next, limiting Capsule architecture's capability. UCaps [6] follows the same design as SegCaps, except that the decoder branch uses the standard Convolutional transpose. The UCaps design is more straightforward to train than SegCaps because of the normalization and initialization techniques in the decoder part. However, it suffers the same limitation as SegCaps because children Capsules share a transformation matrix. Inception Capsule network [7] uses the Inception V4 model to extract features and only replaces the output head with 2 Capsule layers. This design, therefore, does not emphasize Capsule architecture.

To overcome the issues mentioned above, we propose a novel Capsule network, called CapNeXt, that unifies Capsule and ResNeXt architectures for medical image segmentation. In summary, our contributions are as follows:

- We discover a connection between the ResNeXt architecture and the Capsule architecture.
- We propose the CapNeXt architecture that unifies ResNeXt and Capsule architectures into one.
- We proposed architecture is more straightforward to train than other Capsule architectures, more robust to perspective transformation than standard CNN, and improves performance by 1% on tumor segmentation tasks based on 2D SIIM Pneumothorax [8] and 3D KiTS19 [9] datasets.

2. METHOD

We first overview ResNeXt [10] and Capsule [3, 4] architectures followed by their unification into CapNeXt.

ResNeXt architecture. In ResNeXt architecture, the residual path is first separated into smaller groups, each transforms independently from the other. The outputs of the groups are then summed to produce the residual outputs. The residual outputs from the final convolution layer and summation can be expressed as follows:

$$h(x) = \sum_i W_i x_i, \tag{1}$$

where i is the group index, W_i is the weight of group i, and x_i is a feature vector of group i from original input x.

Capsule architecture. For the Capsule architecture, instead of using neurons that only output a scalar, the authors [3, 4] proposed replacing them with higher-order outputs, e.g., vector and matrix. Furthermore, to emphasize the hierarchical relationship between consecutive layers, Capsule architectures use a routing-by-agreement algorithm that dynamically assigns a connection weight from capsules of the previous layer to the next depending on their feature alignment (Fig. 1).

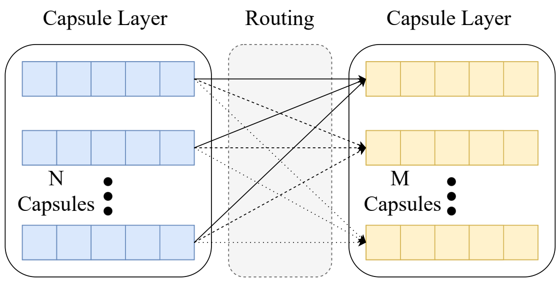


Fig. 1: Capsule network uses a routing-by-agreement procedure to assign which capsule in the lower layer belongs to which capsule in the upper layer.

The computation of a capsule layer can be summarized as follows:

$$r_{ij} = \text{ROUTING}(W_{ij} v_i, n), \tag{2}$$

$$v_j = \sum_i r_{ij} W_{ij} v_i, \tag{3}$$

where $0 \leq i < N$ and $0 \leq j < M$ are index of the previous and current capsules respectively, v_i and v_j are outputs of capsules *i*-th and *j*-th, respectively. W_{ij} and r_{ij} are transformation matrix and connection strength of capsule *i*-th with respect to capsule *j*-th, and n is the number of routing iteration. The routing procedure can either be Dynamic Routing [3] or EM Routing [4].

CapNeXt. Eq. (1) resembles that of Eq. (3) with $r_{ij} = 1$. Therefore, ResNeXt can be treated as a simplified version of Capsule architecture with a trivial routing procedure, and Capsule can be treated as a regularized version of ResNeXt where each path is weighted by their contribution. Our method replaces the path summation of ResNeXt with an iterative routing-by-agreement procedure. This replacement

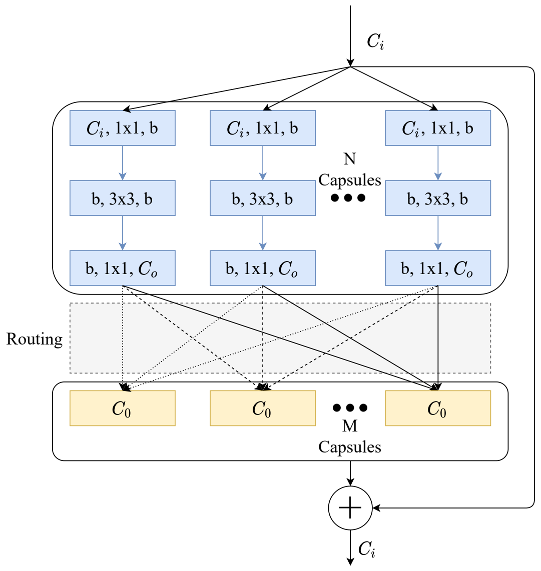


Fig. 2: Integrate Capsule into ResNeXt architecture, the transformation of the inputs follow closely that of ResNeXt. C_0 is the number of features of a capsule.

leverages standard techniques that make a CNN easy to train, such as Normalization and ReLU activation, while keeping the emphasis on Capsule's part-whole relationship routing (Fig. 2). Furthermore, in order to promote the residual architecture for better gradient flow, we add a constraint to the number of output capsules and features.

$$M \times C_0 = C_i. \tag{4}$$

Eq. (4) guarantees that part-object relationship features extracted by capsule routing can be used alongside with default feature extracted by CNN as a residual block. We use the optimized dynamic routing algorithm as the routing procedure. The algorithm is summarized by pseudo-codes in Alg. 1. More detailed descriptions regarding the routing algorithm can be found in [11].

Algorithm 1 Dynamic Routing Algorithm

```
 1: procedure ROUTING($u_{ij}$, n)
 2:     $b_{ij} \leftarrow 0$
 3:     for n iterations do
 4:         $r_{ij} \leftarrow \frac{\exp(b_{ij})}{\sum_j \exp(b_{ij})}$          ▷ assign capsules
 5:         $v_j \leftarrow \sum_i r_{ij} u_{ij}$          ▷ weighted average
 6:         $v_j \leftarrow v_j / ||v_j||$          ▷ normalize length
 7:         $b_{ij} \leftarrow b_{ij} + \langle v_j, u_{ij} \rangle$          ▷ iterative update
 8:     end for
 9:     return $r_{ij}$
10: end procedure
```

3. EXPERIMENTAL RESULTS

3.1. Implementation Details

Datasets. We evaluated our approach on two segmentation datasets: SIIM Pneumothroax [8] and KiTS19 [9] datasets. The SIIM Pneumothorax dataset [8] contains 10,675 chest X-ray images and corresponding Pneumothorax segmentation masks with 2,379 positive and 8,296 negative cases of Pneumothorax. The KiTS19 dataset [9] consists of 210 CT volumes and corresponding kidney and kidney tumor masks. Both datasets were stratified split into three folds, each with a training set and a valid set, to get a better performance overview on each dataset.

Preprocessing and data augmentations. For chest X-ray images in the SIIM Pneumothorax dataset, we resized them to the size of 512 × 512. For CT volumes in the KiTS19 dataset, we resampled them to have the following spacing $3mm \times 1.5mm \times 1.5mm$. Because the anatomical locations of kidneys in the lower half of the body, we bottom cropped the volume to have the size of 128 × 160 × 256 to reduce the computation. The HU values of CT volumes were then clipped into the range of [-80, 300]. All input data were normalized to the range of [-1, 1]. We used random flipping, shifting, rotating, and scaling as augmentations in both datasets.

Architecture. CapNeXt is based on U-Net architecture [1, 2]. Fig. 3 shows an overview of the CapNeXt architecture. The Convolution block consists of Convolutional Layer, Group Normalization Layer [12], and LeakyReLU [13] activation layer with $\alpha = 0.2$. All Convolution blocks use kernel 3 × 3. The CapNeXt block follows the same design as a residual block in ResNeXt architecture [10] but with Batch Normalization layer [14] replaced by Group Normalization [12]. We also use LeakyReLU [13] as the non-linear activation function for the CapNeXt block for better performance. We set N=32 and M=1 for the SIIM Pneumothorax dataset [8] and N=16, M=1 for the KiTS19 dataset [9].

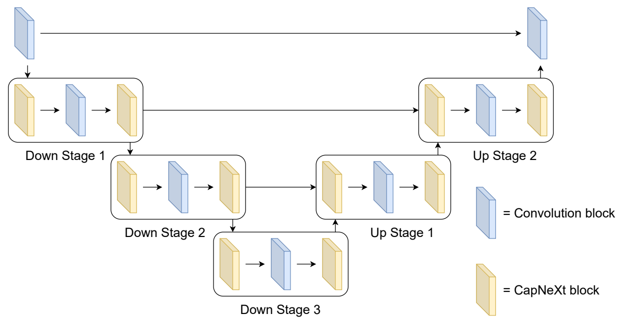


Fig. 3: CapNeXt architecture.

Training. The training batch size was 4 for both datasets. We used the dice loss and cross-entropy loss for the segmentation task on all datasets. For optimizer, we used Adam [15] with a learning rate of 0.001. The learning rate was reduced by a factor of 10 if the model's performance on the valid set did not increase for five epochs. The training was stopped if the model's performance on the valid set did not improve for ten epochs. We experimentally found that progressive training of CapNeXt from low routing iteration to higher ones yields the best performance. Therefore, we first train CapNeXt for routing iteration 1 (n=1) to convergence, then train routing iteration 3 (n=3) with a new initial learning = 0.1 × initial learning rate of n=1. This process is repeated for going from n=3 to n=5 and so on. We implement all models with the PyTorch framework and train them using 2 Nvidia V100 GPUs. We acknowledged that our training pipeline is standard and not optimized for specific tasks like those on the KiTs19 leaderboard. However, we want to study the effect of different architectures using basic pipelines without over tuning for any specific task.

3.2. Segmentation Performance

Table 1: Means and standard deviations of dice score of baseline models and CapNeXt with increasing number of iterations based on 3 folds of SIIM Pneumothorax [8] and KiTS19 [9] datasets. The best results are in bold. UResNeXt is treated as CapNeXt - n=1.

Method	SIIM	KiTS19	
		Kidney	Tumor
U-Net [1, 2] [1]	0.7846 ± 0.0157	0.9528 ± 0.0013	0.6466 ± 0.0149
SegCaps [5] [2]	0.1244 ± 0.0052	0.8091 ± 0.0204	0.3405 ± 0.0903
UCaps [6] [2]	-	0.8066 ± 0.0076	0.3355 ± 0.0125
UResNeXt	0.8040 ± 0.0193	0.9556 ± 0.0018	0.7408 ± 0.0189
CapNeXt - n=3	**0.8138 ± 0.0118**	0.9573 ± 0.0029	**0.7503 ± 0.0135**
CapNeXt - n=5	0.8122 ± 0.0115	0.9577 ± 0.0020	0.7445 ± 0.0195
CapNeXt - n=7	0.8125 ± 0.0127	**0.9583 ± 0.0009**	0.7310 ± 0.0086

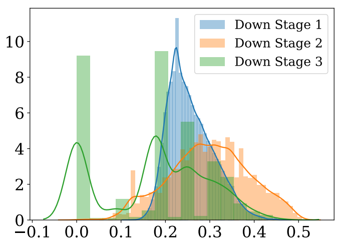
We compare CapNeXt against baselines: standard U-Net, SegCaps, UCaps, and U-Net with ResNeXt backbone (UResNeXt) on SIIM Pneumothorax (2D) and KiTS19 (3D) datasets. CapNext - n=1 can be treated as UResNeXt because each Capsule has the same connection strength. CapNeXt - n=3 improves the Dice score performance by approximately 3% compared with standard U-Net and 1% compared with UResNeXt on the SIIM dataset (Table 1). On the KiTS19 dataset, CapNeXt - n=3 only marginally improves on baselines in the Kidney segmentation task. We hypothesize that due to the straightforwardness of the Kidney segmentation task, the performances are already saturated, and any improvement/regularization to existing models only gives marginal results. However, for more complex tasks such as tumor segmentation, CapNeXt - n=3 improves the Dice

[2]U-Net [1] for SIIM dataset and 3D U-Net [2] for KiTS19 dataset

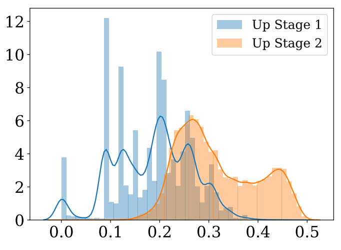
[2]We adopt SegCaps to 3D segmentation task by replacing 2D convolution by its 3D version and the other way around for UCaps.

score performance by 10% compared with standard U-Net and 1% compared with UResNeXt. These improvements on two segmentation datasets show that CapNeXt is an effective realization of Capsule architecture for segmentation tasks.

The performances of SegCaps [5] and UCaps [6] on these datasets are sub-par compared with standard U-Net. UCaps only returns constant outputs for the SIIM Pneumothorax dataset. We hypothesize that SegCaps and UCaps are hard to train for complex segmentation tasks such as Pneumothorax and kidney tumor segmentation because the initialization of Capsule weights is nontrivial. Furthermore, SegCaps and UCaps do not leverage normalization techniques which were shown to make the loss landscape easier to traverse [16]. On the other hand, our approach, which is based on ResNeXt, enables integrating those techniques into Capsule architecture; hence CapNeXt is much easier to train.



(a) Down Stages



(b) Up Stages

Fig. 4: Distributions of the standard deviation of r_{ij} with respect to input Capsules in stages of CapNeXt - n=3 on SIIM Pneumothorax dataset.

We further train CapNeXt with higher routing iterations such as n=5 and n=7. The decrease in performance compared with n=3 shows that our hypothesis of Capsule as a regularization of ResNeXt is valid. We also investigate the variation of connection strengths of all CapNeXt blocks for n=3 case on SIIM Pneumothorax dataset to ensure they are adjusted from being constant at initialization. Fig. 4 shows the standard deviation of r_{ij} with respect to i for each stage in Fig. 3. The majority of the standard deviation of r_{ij} is far from 0, this shows CapNeXt helps promote Capsules that contribute to the task and regularize Capsules that are not useful for the task.

3.3. Robustness to rotation

We investigate whether CapNeXt still retains the perspective robustness of the original Capsule architecture [3]. Table 2 shows that CapNeXt architecture is more robust to perspective shift in the input than standard CNN. It is interesting that SegCaps [5], an implementation of capsule network for segmentation, is not robust to perspective change. This none robustness shows that generalize Capsule architecture to segmentation tasks is highly nontrivial.

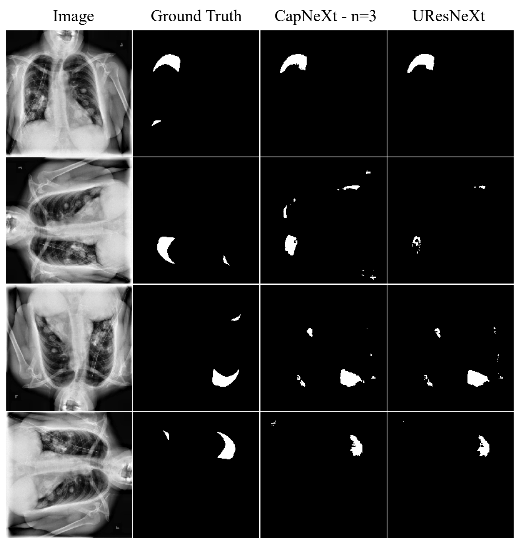


Fig. 5: 0°, 90°, 180°, 270° perspective and model predictions of a chest X-ray image in SIIM Pneumothorax dataset [8]. CapNeXt - n=3 matches the ground truth better than UResNeXt.

Table 2: Mean and standard deviation of dice score on SIIM Pneumothorax dataset [8] for 4 different perspectives of the inputs: normal, rotated 90°, 180°, 270°.

Method	default	rotate - 90°	rotate - 180°	rotate - 270°
SegCaps	0.0974 ± 0.0028	0.0430 ± 0.0024	0.1597 ± 0.0064	0.0300 ± 0.004
UResNeXt	0.7860 ± 0.0336	0.5428 ± 0.0736	0.5716 ± 0.0876	0.5392 ± 0.0653
CapNext - n = 3	**0.7967 ± 0.0174**	**0.5575 ± 0.0552**	**0.5892 ± 0.0656**	**0.5431 ± 0.0348**

Fig. 5 shows different perspectives of a Pneumothorax positive chest X-ray image in the SIIM Pneumothorax dataset [8]. CapNeXt - n=3 is more robust compared with UResNeXt. This robustness shows that our combination of the two architectures retains crucial properties of the Capsule architecture while being easier to train just like a standard CNN architecture.

4. CONCLUSION

We proposed the CapNeXt architecture that unifies ResNeXt and Capsule architecture. This combination improves segmentation performance and makes Capsule easier to train. The improved performances on both 2D and 3D segmentation datasets show that CapNeXt architecture is effective for segmentation tasks. Furthermore, increasing capsule routing iteration in CapNeXt results in general performance degradation. This degradation shows that our treatment of Capsule architecture as a regularization of ResNeXt architecture is sound. However, we only experimented with $M = 1$ case and Dynamic Routing, which may not leverage the total capacity of the Capsule routing procedure. We leave this direction for future works to explore.

5. COMPLIANCE WITH ETHICAL STANDARDS

This research study was conducted retrospectively using human subject data made available in open access by SIIM, University of Minnesota and University of Melbourne. Ethical approval was not required as confirmed by the license attached with the open access data.

6. CONFLICTS OF INTEREST

This work was supported by VinBrain, a company funded by VinGroup, Vietnam.

7. REFERENCES

[1] Thomas Brox Olaf Ronneberger, Philipp Fischer, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, Lecture Notes in Computer Science.

[2] Özgün Çiçe, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger, "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2016 - 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II*.

[3] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton, "Dynamic Routing Between Capsules," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 3856–3866.

[4] Geoffrey E. Hinton, Sara Sabour, and Nicholas Frosst, "Matrix capsules with EM routing," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018, OpenReview.net.

[5] Rodney LaLonde and Ulas Bagci, "Capsules for Object Segmentation," *CoRR*, vol. abs/1804.04241, 2018.

[6] Tan Nguyen, Binh-Son Hua, and Ngan Le, "3D-UCaps: 3D capsules Unet for volumetric image segmentation," in *Medical Image Computing and Computer Assisted Intervention - MICCAI 2021 - 24th International Conference, Strasbourg, France, September 27 - October 1, 2021, Proceedings, Part I*.

[7] Christian Kromm and Karl Rohr, "Inception Capsule Network for Retinal Blood Vessel Segmentation and Centerline Extraction," in *17th IEEE International Symposium on Biomedical Imaging, ISBI 2020, Iowa City, IA, USA, April 3-7, 2020*. 2020, pp. 1223–1226, IEEE.

[8] SIIM, "SIIM Pneumothorax Segmentation Dataset," 2019, data retrieved from Kaggle, `https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation`.

[9] Nicholas Heller, Niranjan Sathianathen, Arveen Kalapara, Edward Walczak, Keenan Moore, Heather Kaluzniak, and et.al., "The KiTS19 Challenge Data: 300 Kidney Tumor Cases with Clinical Context, CT Semantic Segmentations, and Surgical Outcomes," *CoRR*, vol. abs/1904.00445, 2019.

[10] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated Residual Transformations for Deep Neural Networks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 2017, pp. 5987–5995, IEEE Computer Society.

[11] Dilin Wang and Qiang Liu, "An Optimization View on Dynamic Routing Between Capsules," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. 2018, OpenReview.net.

[12] Yuxin Wu and Kaiming He, "Group Normalization," *Int. J. Comput. Vis.*, vol. 128, no. 3, pp. 742–755, 2020.

[13] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li, "Empirical Evaluation of Rectified Activations in Convolutional Network," *CoRR*, vol. abs/1505.00853, 2015.

[14] Sergey Ioffe and Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 2015, vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456, JMLR.org.

[15] Diederik P. Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[16] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry, "How Does Batch Normalization Help Optimization?," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 2488–2498.