PT-CapsNet: A Novel Prediction-Tuning Capsule Network Suitable for Deeper Architectures

Anonymous ICCV submission

Paper ID 6732

1. Architecture Details for PT-Capsule Layers

The architectures of the proposed PT-FC-Caps and PT-LC-Caps are shown in Fig. 1 and Fig. 2, respectively, due to the limited space in our paper.

2. Implementation Details

2.1. Image Classification Task

For the capsule network, both the number of capsule types and the capsule dimension need to be considered. The architecture details for PT-Caps-ResNet-110, PT-Caps-WRN-28 and PT-Caps-DenseNet-100 are provided in Tables 1, 2 and 3, respectively. In each table, the first column indicates the blocks of each model, the second column shows the output feature map size of the corresponding block, and the third column shows the argument details. More specifically, the third column contains (K_1, K_2) (where K_1 and K_2 are the reception field size of the prediction phase and the tuning phase in our PT-CapsNet), the number of capsules, capsule dimension and the number of times a PT-Capsule block is repeated.

layer	size	model
caps1	32×32	$(1,3)$, 8, 4
caps2.x	32×32	$\begin{bmatrix} (1,3), & 8, & 4 \\ (1,3), & 8, & 4 \end{bmatrix} \times 18$
caps3.x	16×16	$\begin{bmatrix} (1,3), & 16, & 4 \\ (1,3), & 16, & 4 \end{bmatrix} \times 18$
caps4.x	8×8	$\begin{bmatrix} (1,3), & 16, & 4 \\ (1,3), & 16, & 8 \end{bmatrix} \times 18$
pooling	1×1	
PT-FC-Caps	-	n, 1

Table 1. Architecture of the PT-Caps-ResNet110 model used for the classification task. n is the number of classes.

For PT-Caps-ResNet110, summarized in Table 1, there are a total of 110 capsule layers in the model. The basic capsule block is composed of one $K_2 = 3$ PT-LC-Capsule layer, and the capsule bottleneck block is composed of two stacked $[K_1 = 1, K_2 = 3]$ PT-LC-Capsule layers. Each capsule layer is followed by a batch normalization (BN) layer and a ReLU activation layer. We have one basic capsule block followed by three capsule bottleneck blocks in the model. For the four building blocks, (the number of instance type, capsule dimension) is set to be $(8,4), (8,4), (16,4), (16,8)$. For the fully connected capsule layer, we use a PT-FC-Capsule layer to directly output $(n, 1)$ capsules to make the final prediction, where n indicates the number of classes.

layer	size	model
caps1	32×32	$(1,3)$, 16, 2
caps2.x	32×32	$\begin{bmatrix} (1,3), & 80, & 2 \\ (1,3), & 80, & 4 \end{bmatrix} \times 4$
caps3.x	16×16	$\begin{bmatrix} (1,3), & 80, & 4 \\ (1,3), & 80, & 8 \end{bmatrix} \times 4$
caps4.x	8×8	$\begin{bmatrix} (1,3), & 160, & 8 \\ (1,3), & 160, & 8 \end{bmatrix} \times 4$
pooling	1×1	
PT-FC-Caps	-	n, 1

Table 2. Architecture of the PT-Caps-WRN28 model used for the classification task. The widening factor is applied on both capsule channel and attribute channel. n is the number of classes.

layer	size	model	m
caps1	32×32	$(1,3)$, 12, 3	$[12, 3]$
caps2.x	32×32	$\begin{bmatrix} (1,1), & 24, & 3 \\ (1,3), & 6, & 3 \end{bmatrix} \times 16$	$[108, 3]$
TD	16×16	$(1,1)$, 54, 3	$[54, 3]$
caps3.x	16×16	$\begin{bmatrix} (1,1), & 24, & 3 \\ (1,3), & 6, & 3 \end{bmatrix} \times 16$	$[150, 3]$
TD	8×8	$(1,1)$, 75, 3	$[75, 3]$
caps4.x	8×8	$\begin{bmatrix} (1,1), & 24, & 3 \\ (1,3), & 6, & 3 \end{bmatrix} \times 16$	$[171, 3]$
pooling	1×1		$[171, 3]$
PT-FC-Caps	-	n, 1	$[n, 1]$

Table 3. Architecture of the PT-Caps-DenseNet100 model used for the classification task. This model is built from 100 capsule layers. caps.x denotes the capsule bottleneck block, and TD represents the capsule transition down block. Column m shows the number of capsules and capsule dimension at the end of the block.

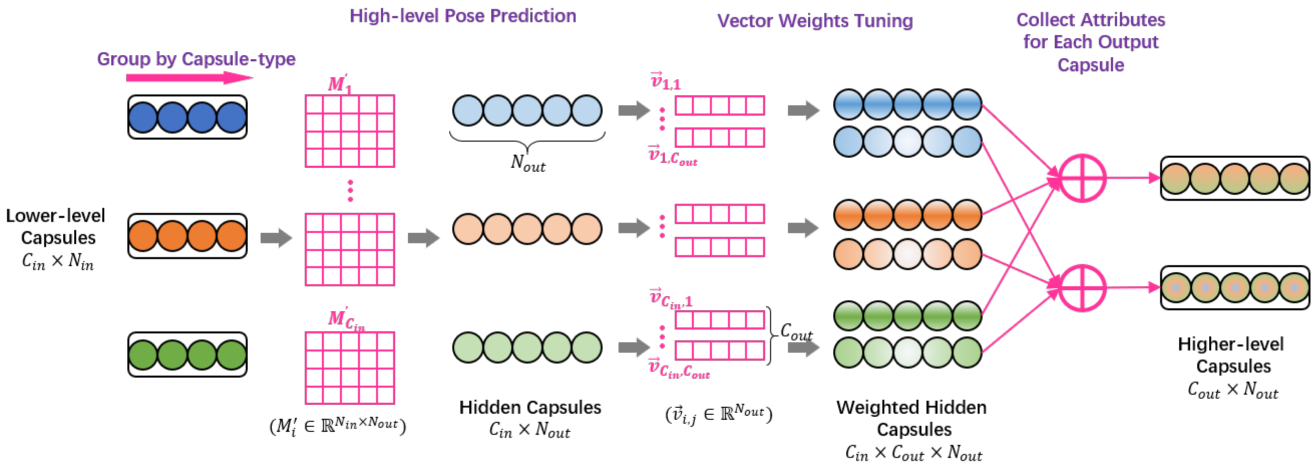

Figure 1. Architecture of the proposed PT-FC-Capsule layer.

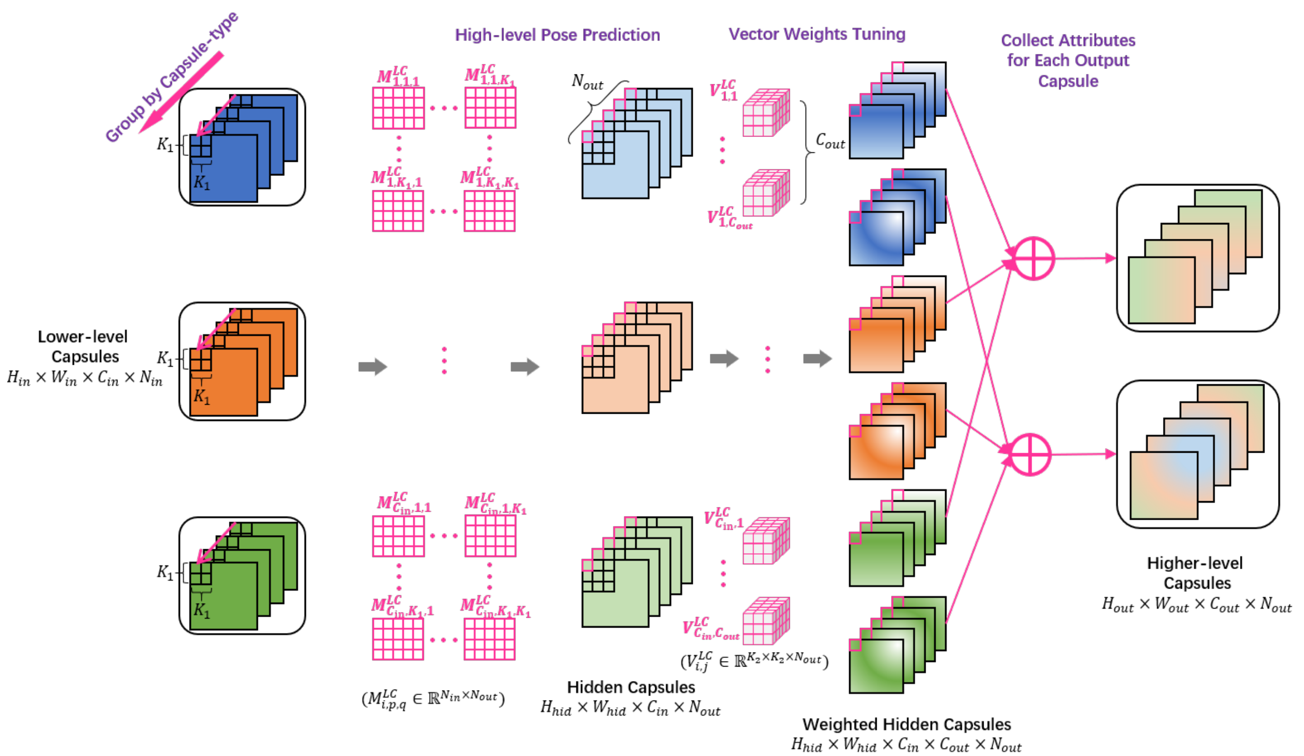

Figure 2. Architecture of the proposed PT-LC-Capsule layer.

In PT-Caps-WRN28, summarized in Table 2, the network depth is 28 capsule layers, and the widening factors for both capsule axis and attribute axis are set as $(5, 2), (1, 2), (2, 1)$ for the three capsule blocks. Each capsule block has two stacked $[K_1 = 1, K_2 = 3]$ PT-LC-Capsule layers, and each capsule layer is followed by one BN layer and one ReLU activation layer. The number of output capsules and capsule dimension in the first capsule layer and the following three building blocks are set to $(16, 2), (80, 4), (80, 8), (160, 8)$.

The PT-Caps-DenseNet100 model summarized in Table 3, there are 100 capsule layers. The growth rate and compression rate of each layer are set to be 6 and 0.5, respectively. Capsule-based dense block is composed of one $[K_1 = 1, K_2 = 1]$ PT-LC-Capsule layer and one $[K_1 = 1, K_2 = 3]$ PT-LC-Capsule layer. Each capsule layer is followed by a BN layer and a ReLU activation layer. Transition down block is composed of a BN layer, followed by ReLU, a $[K_1 = 1, K_2 = 1]$ PT-LC-Capsule layer, and a pooling layer to down-sample the feature maps.

Data augmentation was applied for training all the models. The datasets were augmented by performing 4-pixel zero padding on each side and a horizontal flip with probability of 0.5. Then, 32×32 and 28×28 patches were randomly cropped from the images of the CIFAR and Fashion-MNIST datasets, respectively. The initial learning rate (lr) is set to 0.1, and its decay rate for ResNet-type, WRN-type, and DenseNet-type models is set to 0.1, 0.2 and 0.1, respectively. These models were trained for 160 (lr decay at 80th and 120th epochs), 200 (lr decay at 60th, 120th, and 160th epochs) and 300 (lr decay at 150th, and 225th epochs) epochs, respectively. The weight decay and momentum are set as 0.0001 and 0.9, respectively. SGD optimizer is used with a batch size of 128 images for the ResNet-type and WRN-type models, and 64 images for the DenseNet-type model.

2.2. Semantic Segmentation Task

For semantic segmentation, we have used the ISIC2018 dataset [2][6], which is a binary semantic dataset for melanoma detection for skin lesion analysis. The released data consists of 2594 images with ground truth. We split the dataset into training and testing set by a ratio of 7:3. We adopt the mean Intersection over Union (mIoU) as the evaluation metric. All the models are trained for 100 epochs. The initial learning rate is 0.01 with 10% decay every 50 epochs. We resize the input images to 513×513, and normalize them into 1 channel. For data augmentation, we perform 4-pixel zero padding at all sides, and do horizontal flip with a probability of 0.5. Then, we randomly crop 513×513 patches from the transformed image. We use Adam optimizer and a batch size of 2 images per batch.

Fig. 3 shows the network structure of PT-Caps-DeepLabv3+. The input image in $\mathbb{R}^{513\times513\times3}$ is first sent to a backbone model to extract initial features. We adopt ResNet-101 pretrained on ImageNet as the backbone, and extract the low level and high level features from the outputs of first building block and fourth building block, respectively. The high level features are sent to ASPP block, which is composed of four parallel capsule blocks, with different dilation rates, and one pooling layer, to concatenate features from various sizes of vision field. The concatenated features are then sent to another capsule block and got concatenated with the low level features at the decoder part. The decoder is composed of one convolutional layer for processing low level features, one up sample layer for processing features from ASPP, and three sequential capsule blocks for processing the concatenated features. Finally, the processed features are up sampled to the same size as the input image to perform pixel-level classification for semantic segmentation. Each capsule block is composed of one capsule layer followed by a BN layer and a ReLU layer. The kernel size, number of capsules, and capsule dimension of each capsule layer are indicated in Fig. 3.

2.3. Object Detection Task

For the object detection task on PASCAL VOC dataset [3], the training set was built by merging the training and validation sets of the VOC2007 and VOC2012 datasets. The test set is the released test set of the VOC2007 dataset. The models are trained for 300 epochs. A weight decay of 0.0005, and a momentum of 0.937 are used. The input image size is fixed to 640×640. SGD optimizer is used with a batch size of 10.

The details of the PT-Caps-Yolov5 architecture, designed for object detection, are provided in Table 4. The first column shows the ID of the module. The second column (named $from$) indicates where the input feature maps are from. More specifically, -1 indicates that the input feature maps are from the output of the previous layer, and [-1, a] means that one input is from the previous layer and the other input is from layer #a. n (third column) indicates how many times a module is repeated. The fourth column is the module name, and fifth column contains the argument details of each module. Argument format for Focus, Caps, BottleneckCSP, and SPP modules is [input capsules, input capsule dimension, output capsules, output capsule dimension, K_2, stride]. We set $K_1 = 1$ for all PT-capsule layers. Argument format for Upsample module is [multiplier for spatial size, upsampling algorithm]. Argument format for Concat module means the concatenation is performed along the capsule axis. The arguments for the Detect layer are presented across three lines in the table. The first line represents the number of classes. The second line indicates the size of anchors for each source of feature maps, and the third line represents the corresponding number of input capsules and input capsule dimension of each source of feature maps.

3. Example Output Images

For qualitative comparison, example outputs from ISIC2018 and VOC2007 datasets are provided in Figures 4 and 5 for semantic segmentation and object detection tasks, respectively.

In Fig. 4, columns from left to right show the original RGB image, the ground truth segmentation, the output of DeepLabv3+ [1], and the output of our proposed PT-Caps-DeepLabv3+ model, respectively. These images support the results we present in Table 6 of our manuscript. Our proposed method provides improved segmentation, and boosts the mIoU of the DeepLab baseline model.

First and second columns of Fig. 5 show the detection outputs of YOLOv5[7, 4], and our proposed PT-Caps-YOLOv5 model, respectively. As can be seen from rows 1, 2, 3 and 5, incorporating the PT-Capsnet improves the detection performance of YOLOv5. For instance, in rows 1 and 2, our proposed approach can detect the chairs, in row 3

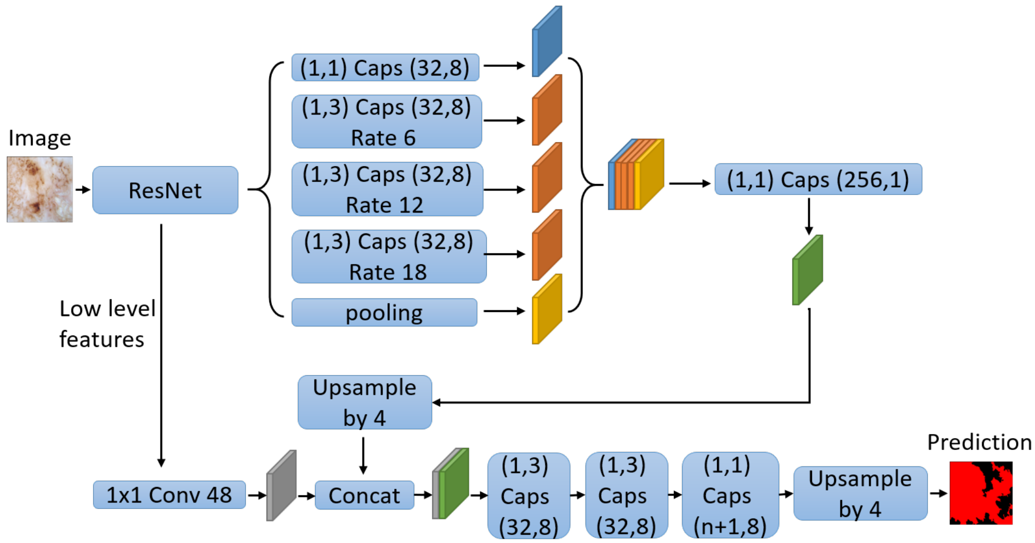

Figure 3. Architecture details of PT-Caps-DeepLabv3+.

#	from	n	module	arguments
0	-1	1	Focus	[1, 3, 16, 8, 3]
1	-1	1	Caps	[16, 8, 32, 8, 3, 2]
2	-1	1	BottleneckCSP	[32, 8, 32, 8, 1]
3	-1	1	Caps	[32, 8, 64, 8, 3, 2]
4	-1	3	BottleneckCSP	[64, 8, 64, 8, 3]
5	-1	1	Caps	[64, 8, 64, 16, 3, 2]
6	-1	3	BottleneckCSP	[64, 16, 64, 16, 3]
7	-1	1	Caps	[64, 16, 128, 16, 3, 2]
8	-1	1	SPP	[128, 16, 128, 16, [5, 9, 13]]
9	-1	1	BottleneckCSP	[128, 16, 128, 16, 1]
10	-1	1	Caps	[128, 16, 64, 16, 1, 1]
11	-1	1	Upsample	[(1, 2, 2), ‘nearest’]
12	[-1, 6]	1	Concat	[1]
13	-1	1	BottleneckCSP	[128, 16, 64, 16, 1]
14	-1	1	Caps	[64, 16, 64, 8, 1, 1]
15	-1	1	Upsample	[(1, 2, 2), ‘nearest’]
16	[-1, 4]	1	Concat	[1]
17	-1	1	BottleneckCSP	[128, 8, 64, 8, 1]
18	-1	1	Caps	[64, 8, 64, 8, 3, 2]
19	[-1, 14]	1	Concat	[1]
20	-1	1	BottleneckCSP	[128, 8, 64, 16, 1]
21	-1	1	Caps	[64, 16, 64, 16, 3, 2]
22	[-1, 10]	1	Concat	[1]
23	-1	1	BottleneckCSP	[128, 16, 128, 16, 1]
24	[17, 20, 23]	1	Detect	20
				[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]
				[64, 8], [64, 16], [128,16]]

Table 4. Architecture of PT-Caps-YOLOv5 model to be used for object detection.

ICCV #6732

it can detect the train and in row 5 it can detect both people, while the baseline cannot. Moreover, for the objects, which are correctly detected by both models (rows 1, 2, 4 and 5), the detection score with our proposed model is higher.

4. Proof of Dynamic Routing Limitation

In our paper, between lines 227–232, it is stated that "Also, the typical dynamic routing algorithm can be easily influenced by the vectors with longer lengths regardless of whether their predictions are reliable or not." We elaborate on and prove this statement below.

In the following, we will use the same notation as [5], wherein the details of the dynamic routing are well-explained. The dynamic routing is used to determine the weights b_{ij} of the intermediate votes $\hat{u}_{j|i}$ outputted from the transformation phase. The normalized weights c_{ij} are obtained by applying `softmax` to b_{ij} along the j axis, and are called the coupling coefficients between the capsules in layer l and the capsules in layer $l+1$. Given $\hat{u}_{j|i}$ (the vectors produced by the first part-whole transformation phase), the dynamic routing initially distributes equal weights to all of the intermediate votes. Then, the averaged vectors s_j are calculated by the weighted sum of $\hat{u}_{j|i}$ along the i axis, and the `squash` function is applied to normalize s_j, and generate the high-level capsules v_j. The dot products of v_j and votes $\hat{u}_{j|i}$ are added to b_{ij} to further adjust the weights for intermediate votes. These steps are iterated for several runs to get the final predicted high-level capsules v_j.

Let us assume that there are M-many high-level capsules and each of them has N-many intermediate votes. Let $\mathfrak{W}_j$ represent the set of intermediate votes that are not reliable (far from truth) for j^{th} high-level capsule, and $\mathfrak{R}_j$ represent the set of reliable intermediate votes (closer to truth) for j^{th} high-level capsule. Then, we have the following:

$$\hat{u}_{\mathfrak{w}_j} = \sum_{\hat{u}_{j|i} \in \mathfrak{W}_j} \hat{u}_{j|i}, \quad \hat{u}_{\mathfrak{r}_j} = \sum_{\hat{u}_{j|i} \in \mathfrak{R}_j} \hat{u}_{j|i}, \tag{1}$$

where $\hat{u}_{\mathfrak{w}_j}$ denotes the sum of votes in set $\mathfrak{W}_j$, and $\hat{u}_{\mathfrak{r}_j}$ denotes the sum of votes in set $\mathfrak{R}_j$. Now, let us consider the case, where the length of $\hat{u}_{\mathfrak{w}_j}$ is longer than the length of $\hat{u}_{\mathfrak{r}_j}$, i.e,:

$$\left|\hat{u}_{\mathfrak{w}_j}\right| > \left|\hat{u}_{\mathfrak{r}_j}\right|. \tag{2}$$

In the first run of dynamic routing, all the votes are given equal weights, so the prediction can be written as:

$$s_j^1 = \frac{\hat{u}_{\mathfrak{w}_j} + \hat{u}_{\mathfrak{r}_j}}{M}, \quad v_j^1 = Squash(s_j^1), \tag{3}$$

where s_j^1 and v_j^1 are the prediction and the normalized prediction, respectively, for the j^{th} high-level capsule in first run. Due to Eq. (2), the vector s_j^1 is closer to $\hat{u}_{\mathfrak{w}_j}$ than $\hat{u}_{\mathfrak{r}_j}$, and the same is true for v_j^1. Thus,

$$cos(v_j^1, \hat{u}_{\mathfrak{w}_j}) > cos(v_j^1, \hat{u}_{\mathfrak{r}_j}), \tag{4}$$

and

$$v_j^1 \cdot \hat{u}_{\mathfrak{w}_j} > v_j^1 \cdot \hat{u}_{\mathfrak{r}_j}. \tag{5}$$

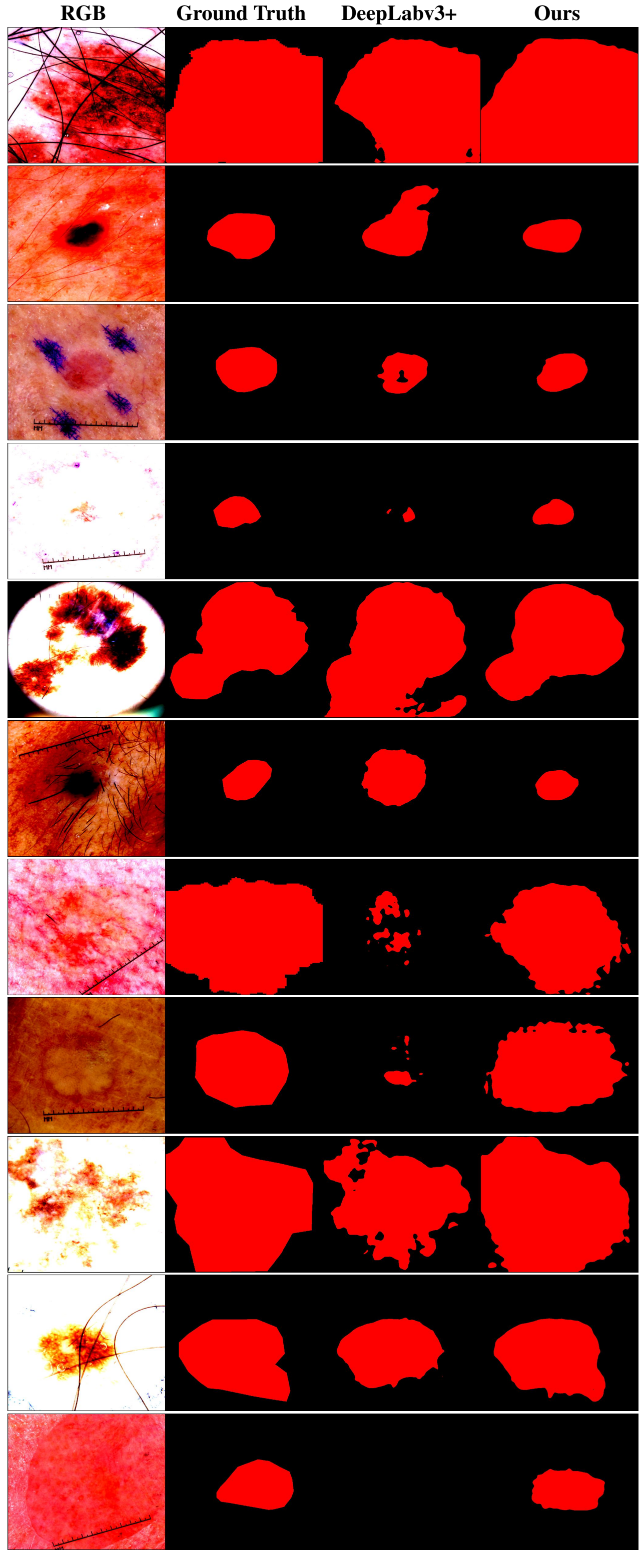

Figure 4. Example segmentation results from ISIC2018 dataset.

The weights updated after the first run can be written as:

$$b^1_{\mathfrak{w}_j} = b^0_{\mathfrak{w}_j} + v^1_j \cdot \hat{u}_{\mathfrak{w}_j}, \quad b^1_{\mathfrak{r}_j} = b^0_{\mathfrak{r}_j} + v^1_j \cdot \hat{u}_{\mathfrak{r}_j}, \tag{6}$$

YOLOv5 **Ours**

Figure 5. Example object detection results from VOC2007 dataset.

where $b^0_{\mathfrak{w}_j} = b^0_{\mathfrak{r}_j}$ are the initial weights with the same value. Thus, based on Eq. (5), we can have:

$$b^1_{\mathfrak{w}_j} > b^1_{\mathfrak{r}_j}, \tag{7}$$

resulting in the weights of unreliable votes being larger than the weights of reliable votes after the first run. After several such iterations, this phenomenon will be exacerbated.

This can be further illustrated with a simple example. Assume that there are three capsule vectors in layer l and two 2D capsule vectors in layer $l+1$. Also assume that six middle capsule vectors are $\hat{u}_{1|1} = (1,2)$, $\hat{u}_{2|1} = (2,2)$, $\hat{u}_{1|2} = (1,2)$, $\hat{u}_{2|2} = (2,2)$, $\hat{u}_{1|3} = (-8,-6)$, $\hat{u}_{2|3} = (-7,-7)$; and the wrong predictions are from $\hat{u}_{1|3}$ and $\hat{u}_{2|3}$. The weights after three iterations can be calculated as: $c_{11} = 0.8548$, $c_{12} = 0.1452$, $c_{21} = 0.8548$, $c_{22} = 0.1452$, $c_{31} = 0.7714$, $c_{32} = 0.2286$; and the predictions for capsules in layer $l+1$ are: $v_1 = (-0.9220, -0.2499)$ and $v_2 = (-0.4774, -0.4774)$ with existence probabilities of 0.9553 and 0.6752, respectively. We can see that the final predicted direction and probability are closer to the wrong predictions.

References

[1] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 3

[2] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019. 3

[3] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 3

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 3

[5] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*, 2017. 5

[6] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5:180161, 2018. 3

[7] ultralytics. Yolo v5 on github. https://github.com/ultralytics/yolov5.git, 2020. 3