TRACE-BACK ALONG CAPSULES AND ITS APPLICATION ON SEMANTIC SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we propose a capsule-based neural network model to solve the semantic segmentation problem. By taking advantage of the extractable part-whole dependencies available in capsule layers, we derive the probabilities of the class labels for individual capsules through a layer-by-layer recursive procedure. We model this procedure as a traceback layer, and take it as a central piece to build an end-to-end segmentation network. In addition to object boundaries, image-level class labels are also explicitly sought in our model, which poses a significant advantage over the state-of-the-art fully convolutional network (FCN) solutions. Experiments conducted on modified MNIST and neuroimages demonstrate that our model considerably enhance the segmentation performance compared to the leading FCN variant.

1 INTRODUCTION

An effective segmentation solution should have a well-equipped mechanism to capture both semantic (i.e., *what*) and location (i.e., *where*) information. The fully convolutional network (FCN) (Long et al., 2015) and its variants (Ronneberger et al., 2015; Noh et al., 2015; Badrinarayanan et al., 2017) constitute a popular class of solutions for this task, producing state-of-the-art results in a variety of applications. FCN and its variants (FCNs) are commonly constructed with an encoder-decoder architecture. In the encoding path, input images are processed through a number of "convolution + pooling" layers to generate high-level latent features, which are then progressively upsampled in the decoder to reconstruct the target pixel labels. The feature maps produced in higher (coarser) layers and those in lower (finer) layers contain complementary information: the former are richer in semantics, while the latter carry more spatial details that define class boundaries.

Originated from and constructed upon convolutional neural networks (CNNs) (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014), FCNs' encoders inherit some common drawbacks of CNNs, one of which is the lack of an internal mechanism in achieving viewpoint-invariant recognition. Traditional CNNs, as well as FCNs, rely on convolution operations to capture certain visual patterns, and utilize poolings to enable multi-scale processing of the input images. Rotation invariance, however, is not readily available in both models. As a result, more data samples or additional network setups (Cohen & Welling, 2016; Cohen et al., 2018) would be required for objects with different viewpoints to be correctly recognized. The absence of explicit part-whole relationships among objects imposes another limitation for FCNs – without such an mechanism, the rich semantic information residing in the higher layers and the precise boundary information in the lower layers can only be integrated in an implicit manner.

Capsule networks (Sabour et al., 2017; Hinton et al., 2018), operating on a different paradigm, can provide a remedy. Capsule nets are built on capsules, each of which is a group of neurons representing one instance of a visual entity, i.e., an object or one of its parts (Hinton et al., 2011). Capsules output both activation probabilities of their presence and the instantiation parameters that represent their properties, such as pose, deformation and texture, relative to a viewer (Hinton et al., 2011). During inference propagation, the principle of coincidence filtering is employed to activate higher-level capsules and set up part-whole relationships among capsule entities. Such part-whole hierarchy lays a solid foundation for viewpoint-invariant recognition, which can be implemented through dynamic routing (Sabour et al., 2017) or EM routing (Hinton et al., 2018). The same hierarchy, if

properly embedded into a segmentation network, would provide a well-grounded platform to specify contextual constraints and enforce label consistency.

With this thought, we develop a capsule-based semantic segmentation solution in this paper. Our approach treats capsule nets as probabilistic graphical models capable of inferring probabilistic dependences among visual entities, through which part-whole relationships can be explicitly constructed. As a concrete implementation, we propose a new capsule layer, which we call traceback layer, to capture such probabilistic part-whole information through a recursive procedure to derive the class memberships for individual pixels. We term our model as *Tr-CapsNet*.

The contributions of our Tr-CapsNet can be summarized as:

1. In Tr-CapsNet , the class labels for individual spatial coordinates within each capsule layer are analytically derived. The traceback procedure in our model, taking advantage of the graphical properties of capsule nets, is mathematically rigorous. To the best of our knowledge, this is the first work to explore an capsule traceback approach for image segmentation. In addition, probability maps at each capsule layer are readily available, which makes it convenient to conduct feature visualization and layer interpretation.

2. In parallel with segmentation, Tr-CapsNet carries out explicit class recognition at the same time. Such explicitness poses a powerful practical advantage over FCNs.

3. The traceback layer is designed under a general context, which makes it applicable to many other potential tasks, including detection, action localizations and network interpretation.

2 BACKGROUND

A capsule (Sabour et al., 2017; Hinton et al., 2018) is aimed to represent an instance of a visual entity, and its outputs include two parts: 1) the probability of the existence of this instance and 2) instantiation parameters. Built on capsules, capsule nets are designed to overcome the drawbacks of CNNs, in particular, the inability of handling viewpoint changes and the absence of part-whole relationships among visual entities. Unlike in CNNs, the weights of the capsule nets represent spatial transformations to be applied on capsules to cast votes for their possible parents.

Capsule nets use high-dimensional coincidence filtering, in the inference time, to detect objects that have a high level of agreement among the votes from their parts/children. Over the same procedure, the assignment probabilities among child-parent pairs are also estimated. The viewpoint-invariance capacity of capsule nets comes from the combination of the equivariance of instantiation parameters and the invariance of transformation weights (Hinton et al., 2011).

In this work, we take advantage of the part-whole relationships available in capsule nets to produce the labels for the pixels at the input image space. To facilitate the derivation of our solution, we would first summarize the related concepts under a probabilistic framework.

Capsules can be categorized into different types (*e.g.* cat, bicycle, or sofa). Each capsule layer l is associated with a set of capsule types $\mathcal{T}^l = \{t_1^l, t_2^l, ..., t_m^l, ...\}$. At each spatial coordinate (position) of certain layer l, there exists exactly one capsule of every type in $\mathcal{T}^l$. The probability of a capsule i existing/activated in the network is denoted as $P(i)$. If capsule i on layer l is within the receptive field of the capsule j of layer $l+1$, we call j a possible parent of i. The assignment probability of i being a part of j is denoted as $P(j|i) = c_{ij}$. The assignment probabilities between capsule i and the capsules co-located with j are forced to meet the normalization requirements of $\sum_{j \in \mathcal{T}^{l+1}} c_{ij} = 1$ (Sabour et al., 2017; Hinton et al., 2018).

Three types of capsule layers were introduced in (Sabour et al., 2017; Hinton et al., 2018), which will also be used as building blocks in our Tr-CapsNet. Therefore, we briefly describe them as follows,

- Primary capsule layer is the first capsules layer, where features from previous convolution layer are processed and transitioned into capsules.

- Convolutional capsule layer takes capsules as inputs and uses certain routing algorithm to compute outputs, which are also capsules.

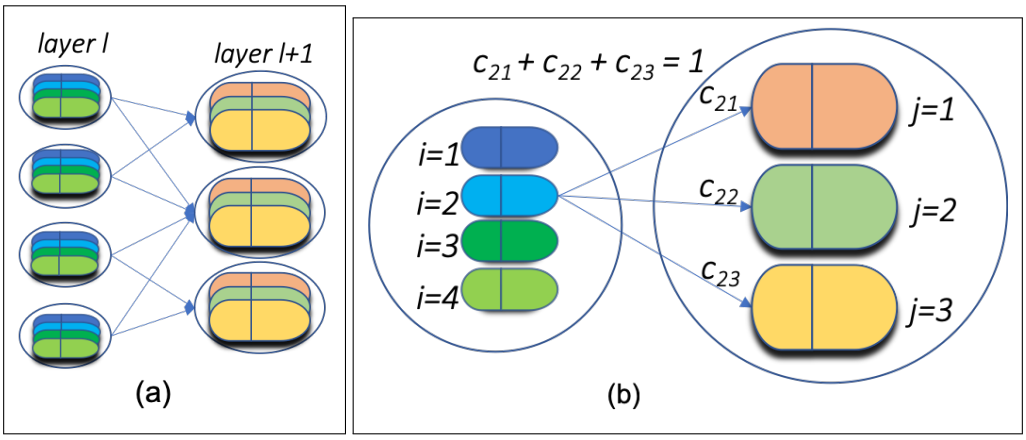


Figure 1: (a) Spatial relationships between two contiguous capsule layers (shown in 1D for clarity). If a position is in the receptive field of another position in the higher layer, the two positions are connected with an arrow pointing the higher layer. Each capsule at the higher-layer position is a possible parent of capsules at its connected lower-layer position. (b) The assignment probabilities and its normalization requirement. Refer to text for more details.

- Class capsule layer L is a degenerated layer with one capsule for each predefined class label $\mathcal{C}_k \in \mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_k, ...\}$. Each capsule in the previous layer is fully connected to the capsules in this layer.

The capsule nets are designed for object classification. In this work, we aim to take advantage of the notion and structure of capsule nets to design a semantic segmentation solution, hoping the viewpoint invariance and network interoperability mechanism can lead to improved segmentation performance. We present the description of our model in next section.

3 Architecture of Our Tr-CapsNet

The general structure of our Tr-CapsNet , which consists of four components, is shown in Figure 2. It starts with feature extraction layers that capture the discriminative features of the input data to be fed into the later layers. For this purpose, convolution layers as in CNNs would be a good choice (Sabour et al., 2017; Hinton et al., 2018). Capsule layers come next. The setup of capsule layers can be a combination of one primary capsule layer, optionally several convolutional capsule layers, followed by a class capsule layer. The third component of our Tr-CapsNet is a traceback layer, which is designed in this paper specifically for segmentation purpose, and its details will be presented in next subsection. The last component restores the original resolution by upsampling the label map computed in the traceback layer. Implementation of this layer(s) can be based on deconvolution (Long et al., 2015), atrous convolution (Chen et al., 2018), or simply bilinear interpolation. It should be noted that the feature extraction and upsampling layers can be regarded as somewhat symmetric in terms dimensionality, where the latter can be optional if no dimensionality reduction occurs in feature extraction layers.

3.1 Traceback layer: design and derivations

The ultimate goal of image segmentation is to compute the probability of each pixel belonging to certain class type, hopefully in great accuracy. The traceback layer is designed to serve this purpose. It should be noted that, over the inference procedure of capsule nets, the probability of each capsule $P(i)$ and the assignment probabilities between contiguous layers of capsules c_{ij} are calculated and become available. With that, $P(\mathcal{C}_k)$, the probability of a class label for each location in the capsule layers, can be potentially inferred through repeated applications of the product rule and the sum rule in probability theory. If we carry out this process, it would essentially trace the class labels in the class capsule layer in a backward-propagation manner, layer by layer until it reaches the first capsule layer. We model this process as a seperate layer and name it a traceback layer. Hinton et al.

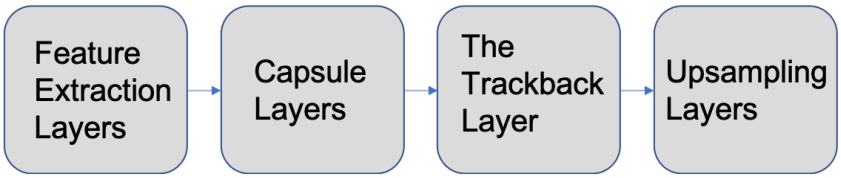


Figure 2: The four components of the Tr-CapsNet. Refer to text for details.

(2000) takes a similar approach, which interprets an image as a parse tree to perform recognition and segmentation simultaneously. However, their model was constructed based on graphical models, with no capsule or other neural network involved.

The detailed traceback procedure is explained as follows. The feature extraction layers and the capsules layers of Tr-CapsNet are adopted from the capsule nets. Therefore, same as in the latter, $P(i)$ and c_{ij} are available during the inference procedure of Tr-CapsNet. With that, the probability of a position belonging to certain class $P(\mathcal{C}_k)$, in layer l, can be calculated as

$$P(\mathcal{C}_k) = \sum_{i \in \mathcal{T}^l} P(\mathcal{C}_k, i) = \sum_{i \in \mathcal{T}^l} P(i) P(\mathcal{C}_k|i), \tag{1}$$

where i is a capsule type associated with layer l and $P(\mathcal{C}_k|i)$ shows the likelihood of certain position taking $\mathcal{C}_k$ as its class label, given that a capsule with type i is located on it. With $P(i)$ of layer l being available over the inference procedure, $P(\mathcal{C}_k|i)$ is the only term to be estimated.

Let L be the index of the class capsule layer. The $P(\mathcal{C}_k|i)$ at its immediately previous layer, $L-1$, would be the assignment probability between the capsule i of the layer $L-1$ and the class capsule $\mathcal{C}_k$ on layer L. Again, $P(\mathcal{C}_k|i)$ is available after inference reaches layer L.

The $P(\mathcal{C}_k|i)$ on other layers, however, needs to be solved. After some simple mathematical derivations, we found that the estimation of $P(\mathcal{C}_k|i)$ could be written into a recursive equation w.r.t. the upper layer, if we assume that each lower-layer capsule only takes capsules in one particular position of the higher-layer as its possible parents. Let capsule j of layer $l+1$ and capsule i of layer l form a possible parent-child pair. Then the conditional probability $P(\mathcal{C}_k|i)$ can be computed as

$$\begin{aligned} P(\mathcal{C}_k|i) &= \sum_{j \in \mathcal{T}^{l+1}} P(\mathcal{C}_k, j|i) \quad (i \text{ is assigned to the parent } j.) \\ &= \sum_{j \in \mathcal{T}^{l+1}} P(j|i) P(\mathcal{C}_k|j, i) \\ &= \sum_{j \in \mathcal{T}^{l+1}} c_{ij} P(\mathcal{C}_k|j) \quad (i \text{ has the same class as its parent } j.) \end{aligned} \tag{2}$$

The recurrence in Eqn. (2) indicates that the estimation of $P(\mathcal{C}_k|i)$ requires the conditional probabilities $P(\mathcal{C}_k|j)$ from the parent layer.

We should note that when the inference propagation reaches layer L, all c_{ij} are available. In addition, the $P(\mathcal{C}_k|j)$ are also available for layer $L-1$. With this combination, the $P(\mathcal{C}_k|i)$ on other layers, can be computed through a traceback procedure. More specially, $P(\mathcal{C}_k|i)$ can be estimated with a layer-by-layer backward propagation procedure – starting at the layer $L-1$, and repeatedly applying Eqn. (2) to compute the conditional prob for the lower layers.

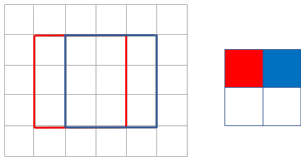


Figure 3: In a convolutional capsule layer, capsules in two or more positions (red position and blue position) can be the parent of a lower-layer capsule (capsules in the overlapped area of red frame and blue frame).

Eqn. (2) is for a simple case where each lower-layer capsule only takes same-position capsules of the higher-layer as its possible parents. For the convolutional capsule

layers, however, capsules in two or more positions might be the parents of a lower-layer capsule (Figure. 3). For these cases, the traceback procedure remains effective, but the computation of $P(\mathcal{C}_k|i)$ needs to be modified. One possible approach is to take $P(\mathcal{C}_k|i) = \sum_n P_n(\mathcal{C}_k|i)/N$, where n represents n-th location for possible parents of the capsule i and N is total location number. This summation bears some resemblance to the summation in deconvolution operation (Long et al., 2015) that adds two overlapping filter responses.

3.2 Loss Function

The total loss function of our model is a weighted sum of the margin loss L_{margin} over the class capsules (Sabour et al., 2017) and pixel-wise softmax cross-entropy loss L_{ce} over the final feature map $L = \lambda_1 L_{\text{margin}} + \lambda_2 L_{\text{ce}}$, where λ_1 and λ_2 are the weights of the margin loss and the cross-entropy loss respectively.

The margin loss is used in (Sabour et al., 2017) for recognition, which strives to increase the probabilities of class capsules for the corresponding objects existing in the input image. Let $P(\mathcal{C}_k)$ denote the probability of the class capsule k. For each predefined class, there is a contributing term L_k in the margin loss

$$L_k = T_k \, max(0, m^+ - P(\mathcal{C}_k))^2 + \lambda_{\text{margin}}(1 - T_k) \, max(0, P(\mathcal{C}_k) - m^-)^2 \tag{3}$$

where $m^+ = 0.9$, $m^- = 0.1$, and $\lambda_{\text{margin}} = 0.5$. $T_k = 1$ when a class k is present in the mask and $T_k = 0$ otherwise. The margin loss is the summation of all L_k, i.e. $L_{\text{margin}} = \sum_k L_k$.

The margin loss plays a key role in training the model because it is the driving force for the initiation and convergence of the capsule layers in the model. While the cross-entropy loss handles the boundary localization in semantic segmentation, the margin loss is responsible for the object recognition. As a result, the ratio $\frac{\lambda_1}{\lambda_2}$ in the total loss equation is an important hyper-parameter to tune the performance of our models, which we will discuss in next section.

4 Experimental Results

Datasets The effectiveness of our model is evaluated with two datasets: modified MNIST and Hippocampus dataset from the Alzheimer's Disease Neuroimaging Initiative (ADNI) project.

The MNIST (LeCun & Cortes, 1998) dataset contains 28×28 images of handwritten digits. In our experiments, random noise is added to each image to broaden the intensity spectrum. Ground-truth segmentations are generated by filtering original foreground digit pixels of each image with an intensity threshold. Two experiments are conducted on this noise-added MNIST dataset. The first one is a multi-digit segmentation test, in which competing methods are evaluated based on their performance in segmenting all ten digits. The second experiment is a robustness test to assess the models under various occlusion scenarios.

The second dataset contains brain images and the corresponding Hippocampus masks of 110 subjects, both downloaded from ADNI website. The images to be segmented are T1-weighted whole brain Magnetic resonance (MR) images, and the ground-truth segmentations were obtained through semi-manual boundary delineation by human raters. In order to alleviate the data imbalance problem, we roughly aligned all brain MRIs, and cropped them with a common region that encloses the right hippocampi of all subjects, which led to subvolumes of size 24×56×48. Two-dimensional slices, sized 24×56, along the axial view of all the three-dimensional volumes are taken as inputs in our experiments. The resulted binary segmentations of each subject are then stacked to constitute the final 3D segmentations for evaluation. Ten-fold cross-validation is used to train and evaluate the performance of the competing models.

Evaluation metrics Totally four metrics are adopted for performance evaluations. *Pixel accuracy* (PA), *mean class accuracy* (MCA) and *Dice ratio* are calculated in our experiments to evaluate the segmentation accuracies in a general sense. In the occlusion tests, we use an additional metric, the number of *added holes* (AH), to assess the segmentation consistency of each model, in terms of topology preservation. An added hole is a connected, wrongly labeled area that completely resides within a correctly labeled segment. The AH (or more accurately, the absence of AH) provides a discrete metric to measure the segmentation consistency within a whole object.

Baselines Modified U-Nets (Ronneberger et al., 2015) are used as the baseline models for comparison. In the MNIST experiment, the U-Net was altered to have 5 convolution layers along the encoder path and 5 deconvolution layers along the decoder path. Pooling layers are discarded, and we replace them with strides to reduce the feature-map dimensions along the encoding path. In the Hippocampus experiment, padding is added to the original U-Net to maintain the spatial dimension of the output of each convolution/deconvolution layer. We only keep one convolutional layers prior to each pooling layer. Explicit data augmentation step in U-Net is replaced with dropout (Srivastava et al., 2014) to reduce overfitting. To explore the effect of receptive fields sizes, we implement four modified U-NDets with different pooling schemes to assign the bottom layers with various scales of feature maps. In reporting the performance of different versions, we add a digit at the end of U-Net to indicate the dimension of the smallest feature map (e.g, U-Net-6 means the dimension of the feature map at the coarsest layer is 6×6).

Implementation details For the 10-digit experiment, we take a regular convolution layer as the feature extraction layer for our Tr-CapsNet. We feed 24 types of capsules associated with the following primary capsule layer and each capsule outputs a 8D vector. The primary capsule layer is followed by the class capsule layer. The traceback layer is applied between the class capsule layer and primary capsule layer. We trained three versions of Tr-CapsNet, with different sizes of label maps (or dimensions of the positions) maintained in the primary capsule layer. More specially, we use 7×7, 9×9 and 11×11 positions, and the corresponding models are named as Tr-CapsNet-7, Tr-CapsNet-9 and Tr-CapsNet-11.

In the Hippocampus experiment, we apply the same feature extraction layers as the baseline U-Net models followed by a primary capsule layer with 32 capsule types. Outputs of capsules in this layer are 8D vectors. The class capsule layer after the primary capsule layer has one 16D vector for each class capsule. One traceback capsule layer is applied between the class capsule layer and primary capsule layer.

The network is implemented in Tensorflow (Abadi et al., 2016) and trained by the Adam optimizer (Kingma & Ba, 2014). All experiments are run on a single Nvidia GTX TITIAN X GPU with 12GB memory.

4.1 Results on modified MNIST

Method	Loss Weights (λ_1, λ_2)	PA	Mean Accuracy	Dice Ratio
Tr-CapsNet-7	1, 1	**99.06**	**99.07**	99.19 ± 5.81
Tr-CapsNet-9	2, 1	99.01	99.02	99.23 ± 5.22
Tr-CapsNet-9	1, 1	99.04	99.05	$\mathbf{99.27 \pm 5.05}$
Tr-CapsNet-11	1, 1	98.31	98.31	98.89 ± 5.34
U-Net-6	-	98.04	98.03	95.63 ± 6.13

Table 1: Results on the 10-digit experiment. Refer to text for details.

In this experiment, we tried different weights of margin loss and cross-entropy in the loss function to evaluate the importance of the individual components. We also explored setups of U-Nets with different receptive scales. The results are summarized in Table 1. Among the baseline U-Nets, U-Net-6 (6 stands for the dimension of the smallest feature map at the bottom layer) gets the best results, so we only include it in the table. From the table, we can tell that our Tr-CapsNet outperform the best U-Net model in all metrics for this dataset. Second, two factors, the dimension of primary capsule layer (7/9/11) and the weights for loss terms, affect the performance of our Tr-CapsNet.

4.2 Results on Hippocampus dataset

Effects of size of the primary capsule layer In order to explore how the dimension of primary layer (number of positions) plays a role in the segmentation procedure, we train different versions of Tr-CapsNet, as we do for MNIST experiments. The results on one particular split of the cross-validation test is shown in Table 2. From the results, it appears that the size of primary layer size and the weights of the loss terms both affect the results to certain extent.

For our Tr-CapsNet, when the primary capsule size set to 4×20 and loss weights set to 15:1, we obtain the most accurate segmentation results. Reducing the dimension of primary layer, as well as decreasing the contribution of margin-loss, seems worsening the model performance. One possible explanation is that when the the dimension of primary layer is set to a very small number, the positions of the primary capsules may not be precise enough to represent the visual entities at the input image level.

For the contribution of margin loss increasing it should make the network strive to obtain more accurate overall recognition. With our mathematically rigorous traceback, higher recognition accuracy would translate into more precise membership labelling at individual pixels.

For U-Nets, with a fixed input size, the feature map size of the bottom layer is the reciprocal of the size of the receptive field of this layer. Setting the feature map size to a small number would allow the information of pixels from a broader range to be integrated, leading to improved class decision and boundary localization. This trend can be spotted in Table 2 from U-Net-3×7 to U-Net-4×20.

Method	Traceback Layer Size	Loss Weights (λ_1, λ_2)	Dice Ratio
Tr-CapsNet	3×7	(7, 1)	87.51 ± 2.378
Tr-CapsNet	4×12	(10, 1)	88.05 ± 2.541
Tr-CapsNet	4×20	(15, 1)	$\mathbf{88.86 \pm 1.628}$
Method	Smallest Feature Map	-	Dice Ratio
U-Net	3×7	-	$\mathbf{88.41 \pm 1.707}$
U-Net	4×12	-	88.27 ± 1.778
U-Net	4×20	-	87.68 ± 2.219

Table 2: Effect of the traceback layer. Refer to text for details.

Overall average performance Through the one-split validation, we identified the potentially best setups for both Tr-CapsNet and modified U-Net, which are Tr-CapsNet- 4×20 and U-Net-3×7, respectively. We then carried out a ten-fold cross-validation on the entire 110 data points. The average Dice ratio for Tr-CapsNet- 4×20 is 87.25 with a standard deviation 5.05, while U-Net-3×7 obtaines 86.23 with a standard deviation 2.19. In other words, our model outperforms the best U-Net model.

4.3 Occlusion test

Method	Loss Weights (λ_1, λ_2)	PA	Dice	AH	0		8	
					Dice	AH	Dice	AH
Tr-CapsNet-9	(2, 1)	**93.49**	**95.44**	**264**	93.89	185	**96.99**	**79**
Tr-CapsNet-9	(3, 1)	92.73	95.05	270	**95.36**	**123**	94.74	147
U-Net-6	-	91.17	91.20	300	92.59	123	89.82	177

Table 3: Results on occluded test. Refer to text for details.

One of the fundamental differences between our Tr-CapsNet model and FCNs lies in the fact that Tr-CapsNet does segmentation and recognition simultaneously. The recognition subtask, if conducted well in practice, would equip Tr-CapsNet with an additional power in dealing with adversarial situations, which include input images with occlusions. In order to evaluate and demonstrate the effect, we design an occlusion test as follows.

We train our model on images of two easily confused digits (e.g. 0 and 8) with the modified MNIST samples. For the test samples, however, we generate occlusions in each images by setting the intensities of several horizontal lines around the image centers to black. The segmentation results are shown in Table 4. It is evident that Tr-CapsNet achieve significantly higher accuracies than the best baseline model.

Table 4 shows several representative prediction images from the occlusion test. Pixels classified into digit 8 are shown in green color and red color pixels have been classified as digit 0. In the first column, U-Net-6 generates rather precise boundary localization but makes totally wrong labelings for all pixels. Our Tr-CapsNet gets both aspects, overall classification and boundary localization, very well worked out. In column 2, U-Net-6 makes the same mistake for the entire digit, while Tr-CapsNet put correct labels for vast majority of the pixels. U-Net-6 performs correctly for the input in column 3.

Overall, Tr-CapsNet appears to be able to generate more robust and consistent segmentation results, and this capability should be attributed to the recognition component and the part-whole mechanism built in our model.

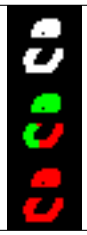
	8	8	0
Groud truth			
Tr-CapsNet-9			
U-Net-6			

Table 4: Prediction results from the occlusion test. Refer to text for details.

5 Discussion and Related Work

Since the inception of the first FCN model, FCN variants have become the most popular solutions for semantic segmentation. Compared with traditional non-deep learning techniques as well as patch-based CNN solutions, the efficacy of FCNs should be attributed, in large part, to their capability to process information from various spatial scales.

FCN models While powerful, FCNs also have some inherent limitations. Originated from CNNs, FCNs tend to lose precise spatial information along the pooling operations/layers, and inconsistent pixel labelings can be resulted from the limited receptive field at each neuron. A number of approaches have explored to add the contextual information back to the CNN component. Conditional Random Fields (CRFs) based solutions (Chen et al., 2014; 2018) refine the final segmentation results with a post-processing stage to boost its ability to capture fine-grained details. Dilated convolutions (Yu & Koltun, 2015; Yu et al., 2017) are utilized to take advantage of the fact that the receptive fields of systematic dilation can be expanded exponentially without losing resolution. Fusion of global features extracted at lower layers and local features from higher layers through skip connections have also been well studied (Long et al., 2015; Ronneberger et al., 2015). To produce segmentation results with certain shape constraints, (Ravishankar et al., 2017; Chen et al., 2013) integrate shape priors into existing deep learning framework with shape-based loss terms.

Capsules, capsule nets and capsule-based segmentation solutions The history of the notion of capsules can be dated back to (Hinton, 1981), in which the author proposed to describe an object over the viewpoint-invariant spatial relationships between the object and its parts. This model should be regarded as the first theoretical prototype of the latest capsule nets (Sabour et al., 2017; Hinton et al., 2018). (Zemel et al., 1990) added probability components into this prototype and encoded both probabilities and viewpoint-invariance in a fully-connected neural network. The instantiation parameters of Zemel's model were hand-crafted from input data.

The issue of how to initialize instantiation parameters was partially addressed in (Hinton et al., 2011), but their model requires transformation matrices to be input externally. Regarding the routing algorithms, (Zemel et al., 1990) introduced the notion that an object can be activated by combining the predictions from its parts, which was materialized as an averaging operation in (Hinton et al., 2011). Sabour et al. (2017); Hinton et al. (2018) took a step forward and resorted to routing-by-agreement algorithms to simultaneously activate objects and set up part-whole relationships.

To the best our knowledge, the first capsule-based semantic segmentation model has been proposed by (LaLonde & Bagci, 2018). Deconvolutional capsule layers were devised and appended directly to convolutional capsule layers to restore the original resolution. Similar to FCNs, this model utilizes skip connections to integrate contextual information from different spatial scales. Our Tr-CapsNet, however, is designed under a different paradigm. Taking the extractable part-whole information as the foundation, label assignments in Tr-CapsNet are explicit, interpretable, and mathematically well-grounded.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Fei Chen, Huimin Yu, Roland Hu, and Xunxun Zeng. Deep learning shape priors for object segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 1870–1877. IEEE, 2013.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999, 2016.

Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

Geoffrey E Hinton, Zoubin Ghahramani, and Yee Whye Teh. Learning to parse images. In *Advances in neural information processing systems*, pp. 463–469, 2000.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pp. 44–51. Springer, 2011.

Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with EM routing. In *International Conference on Learning Representations*, 2018.

Geoffrey F Hinton. A parallel computation that assigns canonical object-based frames of reference. In *Proceedings of the 7th international joint conference on Artificial intelligence-Volume 2*, pp. 683–685. Morgan Kaufmann Publishers Inc., 1981.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Rodney LaLonde and Ulas Bagci. Capsules for object segmentation. *arXiv preprint arXiv:1804.04241*, 2018.

Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1520–1528, 2015.

H Ravishankar, R Venkataramani, S Thiruvenkadam, P Sudhakar, and V Vaidya. Learning and incorporating shape models for semantic segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 203–211. Springer, 2017.

Olaf Ronneberger et al. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI15*, pp. 234–241. Springer, 2015.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pp. 3856–3866, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition*, volume 1, 2017.

Richard S Zemel, Michael C Mozer, and Geoffrey E Hinton. Traffic: Recognizing objects using hierarchical reference frame transformations. In *Advances in neural information processing systems*, pp. 266–273, 1990.