

# Routing Towards Discriminative Power of Class Capsules

Haoyu Yang, Shuhe Li, Bei Yu  
Department of Computer Science and Engineering  
The Chinese University of Hong Kong

**Abstract**—Capsule networks are recently proposed as an alternative to modern neural network architectures. Neurons are replaced with capsule units that represent specific features or entities with normalized vectors or matrices. The activation of lower layer capsules affects the behavior of the following capsules via routing links that are constructed during training via certain routing algorithms. We discuss the routing-by-agreement scheme in dynamic routing algorithm which, in certain cases, leads the networks away from optimality. To obtain better and faster convergence, we propose a routing algorithm that incorporates a regularized quadratic programming problem which can be solved efficiently. Particularly, the proposed routing algorithm targets directly on the discriminative power of class capsules making the correct decision on input instances. We conduct experiments on MNIST, MNIST-Fashion, and CIFAR-10 and show competitive classification results compared to existing capsule networks.

## I. INTRODUCTION

Convolutional neural networks have been deeply studied in recent years. Its variations are successfully and widely applied in different tasks including classification [1], generation [2], segmentation [3], and so on. Convolution layers abstract common features hierarchically by scanning the object with shared kernels that decomposes the original images into small and simple instances which are hence used for classification task. However, the process to some extent violates the nature of recognizing objects, that the visual system resembles parse tree-like [4] structures on fixation points adopted by human vision. In each layer of a parse tree, neurons are grouped together representing certain objects, which are known as capsules.

Capsule networks are recently proposed as an alternative of modern convolutional neural network architectures that changes the way neural networks are trained and features are represented, and, as a result, it also brings robustness to adversarial attacks and overlapped objects [5]. A capsule is a group of neurons that represent a feature or an entity. The capsule length reflects by how much the capsule is activated or the probability a corresponding entity exists in a given image. Capsules in adjacent layers are densely connected via traditional neuron links with their weights learned through routing-by-agreement algorithms, as shown in Figure 1. Another characteristic of capsule networks is that lower level features are constructing higher level entities as layer goes deeper, compared to convolutional neural networks that perform feature abstraction layer by layer.

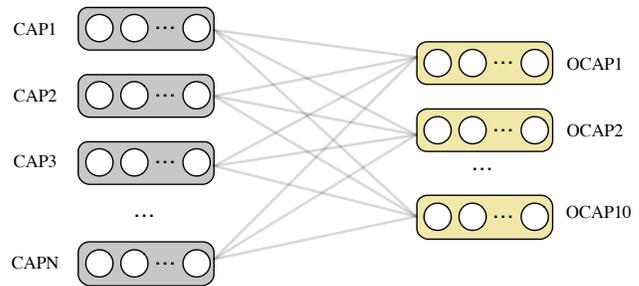


Fig. 1 Visualization of capsule layers with 10 capsules in the output layer that represent the existence of 10 classes.

Two recent capsule routing algorithms are dynamic routing [5] and EM routing [6]. Dynamic routing quantifies the agreement between capsules via their inner-product. The greater inner-product value indicates two capsules agree more with each other and the dynamic routing aims to amplify the agreement. EM routing models each higher level capsule as a Gaussian and the posterior probability of previous layer capsules determines in which level they are connected to higher level capsules. In both routing algorithms, capsules are coupled to higher level capsules according to certain agreement metrics without considering the prediction results.

In this work, we discuss and analyze the routing-by-agreement mechanisms in capsule networks and propose a routing algorithm that can achieve faster convergence and better discrimination results in classification tasks. The algorithm is inspired by two observations: (1) the ultimate objective of training capsule networks is to make the correctly activated output capsules have the largest lengths and (2) the feature capsules (capsules before the output layer) are reasonable to have negative effects on capsules in the following layers. We also propose several training tricks to enlarge the solution space that can result in higher classification accuracy on several datasets. We pick the capsule network architecture used in [5] as a case study to show how our methods benefit the training of capsule networks.

## II. RELATED WORKS

[7] developed credibility networks where images are interpreted as a parse tree or a collection of parse trees with the leaf nodes being image pixels. Each node and its associated latent variables represents an object and its pose information that

forms higher level objects. The concept of capsule comes from transforming auto-encoders [8], where each capsule consists of an instantiation of certain entity. All the entities together reconstructs images with some transformation that is applied on the instantiation vectors with some probabilities. Although the capsule in [8] aims at reconstruction, it already comes with similar features of the capsule discussed in this paper, i.e., associating with a higher level object in a feedforward style. [5] and [6] are two latest implementations of capsule networks for object classification, certain routing algorithms have been discussed in previous section. [9] rephrased the dynamic routing algorithm as a KL divergence regularized clustering problem that inspires an improved solution resembling agglomerative fuzzy k-means, which can be solved by coordinate descent.

### III. THE ALGORITHM

In this section, we will discuss the details of the routing-by-agreement scheme in the dynamic routing algorithm based on which, an improved routing algorithm is proposed that targets directly at the discriminative power of class capsules.

#### A. Dynamic Routing-by-Agreement

Routing-by-agreement aims to couple the lower level capsules to higher level capsules when they agree with each other. Here we will discuss the coupling procedure from primary capsules to output capsules in [5]. Each primary capsule  $\mathbf{u}_i$  is first projected to the space of digital capsules in the follow-up layer by

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i, \quad (1)$$

and the digital capsules are then derived from the weighted summation of all  $\hat{\mathbf{u}}_{j|i}$ s, as in Equation (2),

$$\mathbf{v}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}, \mathbf{s}_j = \text{squash}(\mathbf{v}_j). \quad (2)$$

where squash brings nonlinearity to digital capsules and scales capsule length to between 0 and 1,

$$\text{squash}(\mathbf{v}) = \frac{\|\mathbf{v}\|_2^2}{1 + \|\mathbf{v}\|_2^2} \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|_2}, \quad (3)$$

and  $c_{ij}$ s are softmaxed coupling coefficients  $b_{ij}$ s (see Equation (4)) that determines the probability on primary capsule  $\mathbf{u}_i$  should contribute to activate the digital capsule  $\mathbf{v}_j$ .

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}. \quad (4)$$

In each routing iteration,  $c_{ij}$  will be amplified if capsule  $i$  agrees with capsule  $j$  the most. There are two beneath assumptions in the dynamic routing algorithm.

**Assumption 1.** *Primary capsules do not have negative impact on the activation of digital capsules.*

**Assumption 2.** *All digital capsules are activated correctly.*

Assumption 1 comes from the fact that  $c_{ij}$ s are always positive due to Equation (4), which guarantees each primary

capsule will more or less contribute to higher level capsules. Such design cannot efficiently represent the case when one or more specific entities/features can never exist in certain objects. The dynamic routing algorithm is coherent with Assumption 2, because in each routing iteration,  $c_{ij}$  will always be increased if  $\hat{\mathbf{u}}_{j|i}$  has largest inner product with  $\mathbf{v}_j$ . Here are some potential drawbacks when accepting these assumptions. Assumption 1 can possibly limit the solution space with always-nonnegative  $c_{ij}$ 's. On the other hand, more importantly, Assumption 2 does not necessarily hold during training especially at very early epochs. Unconditionally coupling with a digital capsule simply based on inner-product agreement even that capsule is incorrectly activated will hold back the whole training procedure. According to the observations above, we will introduce an improved routing algorithm that is expected to achieve better classification results and faster convergence.

#### B. Routing Towards Discriminative Quality

The length of a capsule is originally designed as indicators of the existence of corresponding features. Features with larger capsule vector length are more likely to exist in a given instance. For simplicity, we will use the architecture in [5] in the following discussion. Digital capsules, as the output layer of capsule networks, make the final decision of prediction tasks. Therefore, the activation error should be considered when determining routing coefficients. According to the prediction mechanism of capsule networks, intuitively, the length of capsules that are supposed to be activated should be maximized, while the length of inactivated capsules should be minimized. This objective can be written in a unified form as shown in Equation (5).

$$\max_{\mathbf{b}_j} \delta_{ij} \|\mathbf{v}_j\|_2^2, \quad (5a)$$

$$\text{s.t. } \delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ -1, & \text{otherwise,} \end{cases} \quad (5b)$$

where  $i$  corresponds to the labels of given observations and the indicator function  $\delta_{ij}$  ensures Equation (5) consistent with the discrimination mechanism of digital capsules. We denote  $\mathbf{b}_j$  as the routing coefficients corresponding to the  $j^{\text{th}}$  output capsule before going into softmax function. To additionally enlarge the representation space of digital capsules, we also discard the softmax of routing coefficients such that each digital capsule is calculated directly through

$$\mathbf{v}_j = \sum_i b_{ij}\hat{\mathbf{u}}_{j|i} = \hat{\mathbf{U}}^\top \mathbf{b}_j, \quad (6)$$

where rows of  $\hat{\mathbf{U}}$  are the primary capsules projected into digital capsule space with  $\hat{\mathbf{u}}_{j|i}$ , and the objective of Equation (5) becomes

$$\max_{\mathbf{b}_j} \delta_{ij,k} \mathbf{b}_j^\top \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top \mathbf{b}_j. \quad (7)$$

Note that Equation (7) can not fit each individual observation in the training dataset, because it will always give optimal optimal solution of

$$\mathbf{b}_j^* = \begin{cases} \mathbf{0}, & \text{if } \delta_{ij,k} = -1, \\ \text{inf}, & \text{if } \delta_{ij,k} = 1. \end{cases} \quad (8)$$

Rewrite Equation (7) into batch mode we have a slightly better formulation:

$$\max_{\mathbf{b}_j} \sum_k \delta_{ij,k} \mathbf{b}_j^\top \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top \mathbf{b}_j, \quad (9)$$

where  $j$  corresponds to the  $j^{\text{th}}$  digital capsule and  $k$  is the  $k^{\text{th}}$  observation in the training batch. We are able to obtain a local optimal of Equation (9) as long as  $\sum_k \delta_{ij,k} \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top \neq \mathbf{0}$ .

1)  $l_2$ -Regularization: Observing that each capsule is equipped with very small number of neuron nodes that makes  $\hat{\mathbf{U}}_k \in \mathbb{R}^{m \times n}$  have very few columns and as a result,  $\hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top$  has a very low rank that also makes it possible to turn Equation (9) into a ridge regression-like [10] problem with a small regularization on  $\mathbf{b}_j$ , as in Equation (10).

$$\max_{\mathbf{b}_j} \sum_{k=1}^p \delta_{ij,k} \mathbf{b}_j^\top \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top \mathbf{b}_j - \lambda \|\mathbf{b}_j\|_2^2, \quad (10)$$

where  $\lambda$  is the regularization coefficient which is usually set small, e.g. 0.001. Note that Equation (10) is no longer convex that makes the maximization reasonable, by the fact that

$$\begin{aligned} & \sum_{k=1}^p \delta_{ij,k} (\mathbf{b}_j^\top \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top \mathbf{b}_j) - \lambda \|\mathbf{b}_j\|_2^2 \\ &= \mathbf{b}_j^\top \left( \sum_{k=1}^p \delta_{ij,k} \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top - \lambda \mathbf{I} \right) \mathbf{b}_j \\ &= \mathbf{b}_j^\top \mathbf{Q} (\mathbf{\Lambda} - \lambda \mathbf{I}) \mathbf{Q}^\top \mathbf{b}_j \neq \mathbf{0}, \end{aligned} \quad (11)$$

where  $\lambda > 0$  and  $\mathbf{\Lambda}$  is a diagonal matrix with at least  $m - pn$  zeros in its diagonal that ensures  $\mathbf{\Lambda} - \lambda \mathbf{I}$  to be indefinite as long as the batch size  $p$  is not extremely large. The regularization term also avoids  $\mathbf{b}_j$  going too large or too small that resembles momentum in classical neural network training algorithms [11]. Because the capsule coupling coefficients are designed to approximate a large set of observations, we solve the problem greedily by ascending the gradient of on a batch of observations, as in Equation (13). Let

$$r = \sum_{k=1}^p \delta_{ij,k} (\mathbf{b}_j^\top \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top \mathbf{b}_j) - \lambda \|\mathbf{b}_j\|_2^2, \quad (12)$$

and  $\mathbf{b}_j$  can be updated as follows:

$$\begin{aligned} \mathbf{b}_j &= \mathbf{b}_j + \gamma \frac{\partial r}{\partial \mathbf{b}_j} \\ &= \mathbf{b}_j + \gamma \sum_{k=1}^p \frac{\partial \mathbf{b}_j^\top (\delta_{ij,k} \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top - \frac{\lambda}{p} \mathbf{I}) \mathbf{b}_j}{\partial \mathbf{b}_j} \\ &= \mathbf{b}_j + 2\gamma \sum_{k=1}^p (\delta_{ij,k} \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top - \frac{\lambda}{p} \mathbf{I}) \mathbf{b}_j, \end{aligned} \quad (13)$$

where  $p$  denotes the observation batch size.

2)  $l_1$ -Regularization: In the original capsule networks design, primary capsules and digital capsules are densely connected. It has been shown in previous works such densely connected structure is easily suffering from overfitting [12], [13]. Enforcement weight sharing in CNN and drop neurons when training densely connected nets (also known as dropout) are two major solutions in deep learning scope. Weight sharing is similarly applied with the implementation of capsule networks in [6].

Instead of predetermine the neuron/capsule connectivity or randomly drop connection, we propose an alternative that can automatically learn how capsules in different layers are linked with each other. The routing objectives can be found in Equation (14),

$$\max_{\mathbf{b}_j} \sum_{k=1}^p \delta_{ij,k} \mathbf{b}_j^\top \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top \mathbf{b}_j - \lambda \|\mathbf{b}_j\|_1, \quad (14)$$

where an  $l_1$  penalty term is applied on  $\mathbf{b}_j$  that admits a sparse solution [14]. Because solving Equation (14) requires to calculate the gradient of  $|x|$  at  $x = 0$ , we define  $\frac{\partial |x|}{\partial x} \Big|_{x=0} = 0$ . Routing coefficients can then be similarly updated as follows:

$$\mathbf{b}_j = \mathbf{b}_j + 2\gamma \left( \sum_{k=1}^p \mathbf{b}_j^\top \delta_{ij,k} \hat{\mathbf{U}}_k \hat{\mathbf{U}}_k^\top - \lambda \frac{\partial \|\mathbf{b}_j\|_1}{\partial \mathbf{b}_j} \right). \quad (15)$$

### C. Training Capsule Networks

Note that for both strategies in Equation (13) and Equation (15) are compatible with networks that contain more than 2 capsule layers, where the routing coefficients can be accordingly updated through chain rule. When training other neuron weights, we adopt the margin loss as in Equation (16) [5].

$$\begin{aligned} L_k &= T_k \max(0, m^+ - \|\mathbf{v}_k\|_2)^2 \\ &\quad + \lambda' (1 - T_k) \max(0, \|\mathbf{v}_k\|_2 - m^-)^2, \end{aligned} \quad (16)$$

where  $T_k = 1$  if class  $k$  is present in the  $k^{\text{th}}$  output capsule.

As shown in Algorithm 1, the routing coefficients and other neuron weights are updated alternatively in each training step, where  $n_d$ ,  $n_b$  and  $n_r$  are the number of output capsules, the number of iterations to update regular weights and the number of iterations for routing, respectively. In each training iteration, we first sample a minibatch of observations from the training set (lines 1–2), we then update the regular neuron weights for  $n_b$  steps with routing coefficients fixed (lines 3–6), and finally the routing coefficients are updated according to the formulation in Equation (10) or Equation (14) (lines 7–9).

## IV. EXPERIMENTS

To verify the proposed methods, in this paper, we adopt the simplest capsule neural network architecture in [5], which is implemented with tensorflow [15]. We conduct experiments on three datasets that include MNIST [16], Fashion-MNIST [17] and CIFAR-10 [18]. Notations ‘‘DR’’, ‘‘L1’’ and ‘‘L2’’ correspond to original dynamic routing [5], the proposed algorithm with  $l_1$  regularization and  $l_2$  regularization

TABLE I Neural network configuration for each benchmark.

Layer	Filter/Capsule Size	Activation	Filter/Capsule/Neuron Number		
			MNIST	Fashion-MNIST	CIFAR-10
Conv1	9×9	ReLU	256	256	256
Cap1	8	Squash	32	32	64
Cap2	16	Squash	10	10	10
FC1	-	ReLU	512	512	-
FC2	-	ReLU	1024	1024	-
FC3	-	Sigmoid	784	784	-

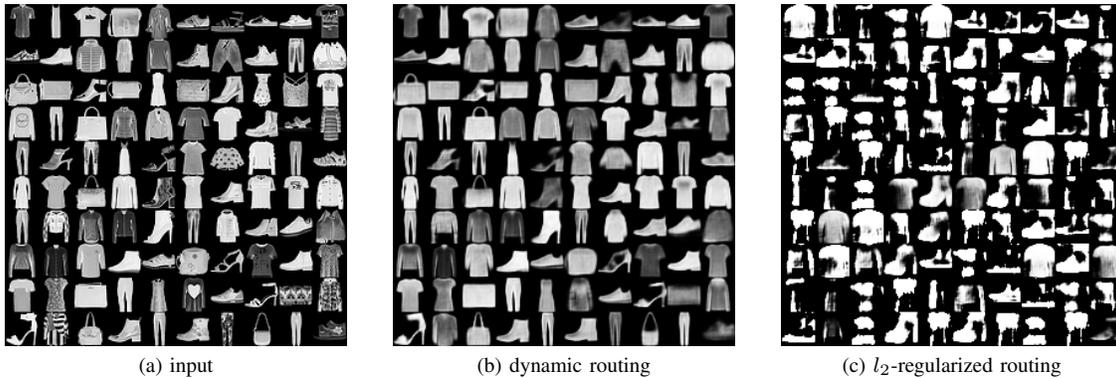


Fig. 2 Visualization of the reconstructed images on Fashion-MNIST dataset. (a) 100 image samples from the Fashion-MNIST dataset that can be correctly classified by the capsule networks that is trained with our algorithm; (b) The corresponding images reconstructed from the capsule networks and the reconstruction networks trained with dynamic routing algorithm; (c) The corresponding images reconstructed from the reconstruction networks trained in dynamic routing where the input capsules are obtained from our  $l_2$ -regularized routing algorithm without reconstruction networks.

**Algorithm 1** Training Capsule Networks. Routing coefficients  $\mathbf{b}_j, j = 1, 2, \dots, n_d$  and regular neuron weights  $\mathbf{W}$  are updated alternatively. In each iteration,  $n_r$  steps routing and  $n_b$  steps back-propagation are conducted respectively. We pick  $n_r = 1$  and  $n_b = 1$  in all the experiments.

```

1: for number of training iterations do
2:   Sample a minibatch of  $p$  observations  $\{\mathbf{x}_i | i = 1, 2, \dots, p\}$ 
   from the training dataset;
3:   for  $n_b$  steps do
4:     Update  $\mathbf{W}$  by descending its gradient;
5:      $\mathbf{W} \leftarrow \mathbf{W} - \frac{1}{p} \sum_{i=1}^p \sum_{j=1}^{n_d} \frac{\partial L^k}{\partial \mathbf{W}}$ ;
6:   end for
7:   for  $n_r$  steps do
8:     Update  $\mathbf{b}_j$ s by ascending its gradient as in Equation (13)
   or Equation (15);
9:   end for
10: end for

```

respectively. “x/FC” denotes no fully connected reconstruction net is applied.

#### A. Neural Network Architecture

In all the experiments, we adopt the simplest 3-layer capsule networks as used in [5], with one convolutional layer, one primary capsule layer and one output layer. Specifications are listed in TABLE I. The first convolution layer is defined

TABLE II Classification results of three benchmarks in terms of error rate (%).

Benchmarks	DR [5]	L2	L1	L2/FC	L1/FC
MNIST	0.34	0.35	<b>0.32</b>	0.35	0.44
Fashion-MNIST	7.21	7.01	6.76	<b>6.75</b>	6.77
CIFAR-10	15.3	-	-	14.52	<b>14.04</b>
Average	7.62	-	-	7.21	<b>7.08</b>

by 256 9×9 kernels followed by two capsule layers with capsule vector dimensions of 8 and 16 respectively. We use 32 primary capsules and 10 output capsules for the MNIST and Fashion-MNIST dataset and the primary capsule number is doubled when we are conducting experiments on CIFAR-10. The reconstruction networks for MNIST dataset has 3 fully connected layers with neuron nodes of 512, 1024 and 784 respectively. Reconstruction is not applied when training the network on CIFAR-10. Each capsule layer is followed by the squash activation as in Equation (3). We apply ReLU on the rest of the layers except the last layer in the reconstruction networks, which is equipped with sigmoid.

#### B. Image Classification

In the first experiment, we compare the classification results with [5] on three benchmarks discussed above as shown in

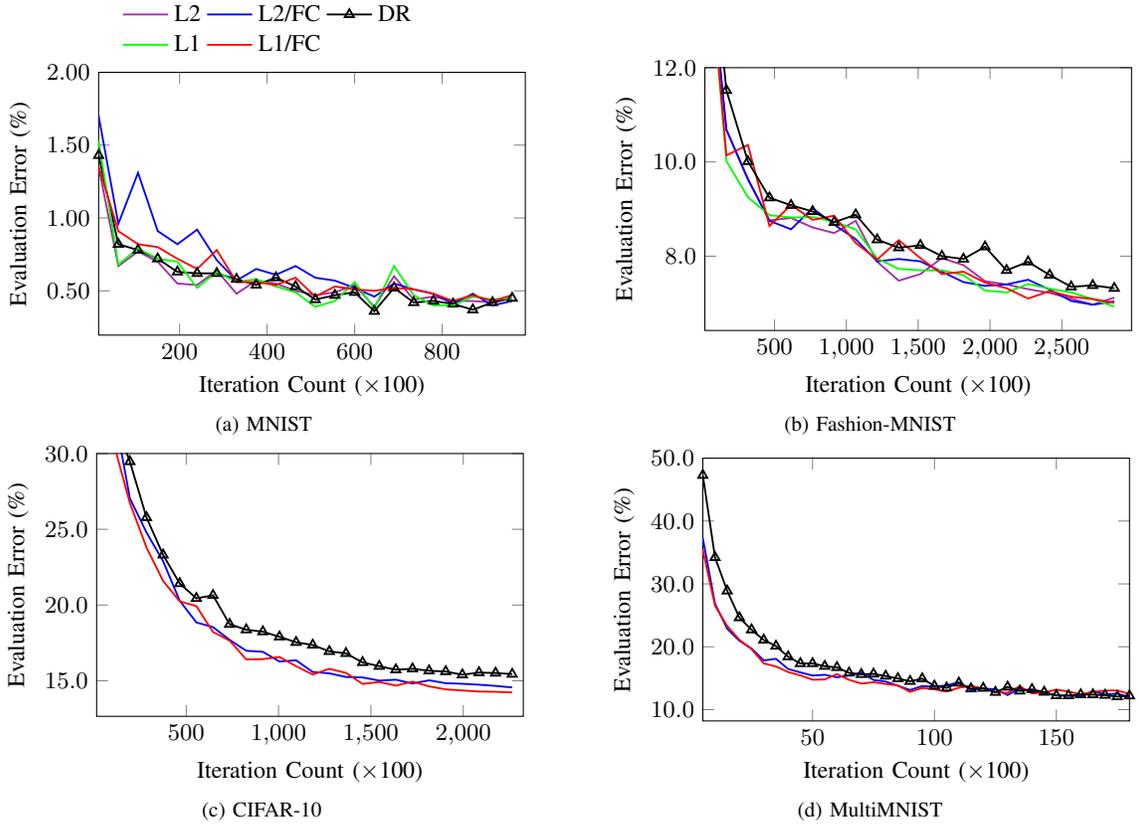


Fig. 3 Visualization of the convergence on different routing algorithms. (a)–(d) are regular training curves on MNIST, Fashion-MNIST, CIFAR-10 and MultiMNIST, respectively.

TABLE II.

For the MNIST dataset, we observe that the margin loss drops fast even at early training stage when fully connected reconstruction networks are removed. Therefore we pick a smaller batch size (i.e. 32) when training the capsule networks without reconstruction networks. The learning rate decays by 0.5 every 1000 iterations. Because deep learning models can easily achieve above 99.0% accuracy on MNIST, it is hard to have further significant improvements. Here we only show similar classification results compared to dynamic routing.

We train the Fashion-MNIST dataset with a batch size of 128 and a starting learning rate of 0.001 that decays by 0.96 every 1000 iterations. We also train the network without reconstruction layers and keep everything else unchanged. The test error rate drops from 7.21% to 7.01% and 6.76% when using our routing algorithms with  $l_2$  and  $l_1$  regularization respectively. It can also be seen that the classification error further drops when the reconstruction networks are removed especially on the  $l_2$  regularized algorithm.

The CIFAR-10 model is trained on a single capsule networks (without any model ensemble) using the architecture specified in TABLE I where the number of primary capsules is doubled and reconstruction networks are removed as in [5]. We also replace the classic ReLU with LeakyReLU [19] during training that shows better performance. As shown in

TABLE II, our routing methods with  $l_2$  and  $l_1$  regularization reduce the single model classification error rate by 0.78% and 1.26% respectively compared to dynamic routing.

1) *Discussion of FC-Regularization:* TABLE II shows smaller classification error when the neural network is trained without fully-connected reconstruction nets. One explanation is that the training set is not necessarily covering the whole data space. In this experiment, we evaluate the trained neural networks on Fashion-MNIST with and without reconstruction regularization. We feed correctly activated output capsule vectors associated to the model without reconstruction networks into the reconstruction networks and obtain the reconstructed images as shown in Fig. 2. It can be seen that a fraction of those images are not correspondingly reconstructed but correctly classified, which agrees with TABLE II. For the classification purpose only, removing the reconstruction networks can also improve the training efficiency by dropping redundant trainable variables.

2) *Segmenting Overlapped Digits:* We also conduct experiments to show our routing solution attains the ability to recognize overlapped digits. In this experiment, we adopt the MultiMNIST dataset as used in [5], where two digits from different classes are overlapped together with at most 4 pixels shift in each direction that forms into  $36 \times 36$  images. The MultiMNIST dataset contains 60 million training samples and

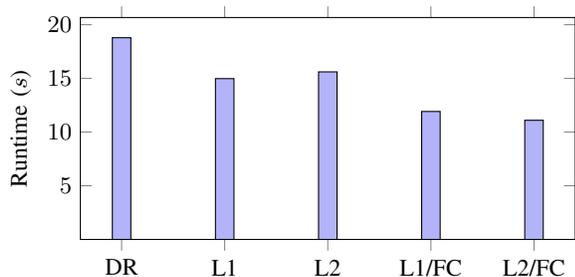


Fig. 4 Approximate training time per 100 steps.

10 million testing samples. We train the capsule nets using  $l_2$  and  $l_1$  regularized routing algorithms respectively. The initial learning rate is set 0.001 that decays by 0.96 every 20000 iterations. We also set a larger regularization coefficient with  $\lambda = 0.001$  to avoid over-fitting. Because the MultiMNIST training set is extremely large we train the neural networks for 200000 steps for both “DR” and our routing algorithms, when we achieved an evaluation error of 7.54% compared to 7.47% of dynamic routing. It should be noted that although the evaluation errors are similar for both methods, the training speed is relatively faster than dynamic routing, as discussed in the following section.

### C. Convergence of the Routing Algorithm

In support of the proposed routing algorithm, we visualize the evaluation performance of the capsule networks along with the training procedure in Fig. 3. All models show similar convergence curves on MNIST dataset that reach an evaluation error under 0.4%. For the more challenging Fashion-MNIST and CIFAR-10, all regularized routing algorithms discussed in this work exhibit faster and better convergence in terms of evaluation error and hence demonstrates the effectiveness and the efficiency of our methods. Because the MultiMNIST dataset is extremely large, we only visualize the training behavior in 20000 steps. We can observe that the evaluation error drops much faster than dynamic routing at early training steps, which is consistent with the discussion about Assumption 2. Our algorithm also shows an advantage in terms of training runtime that each step can save at least 20% runtime compared to dynamic routing (DR), as shown in Fig. 4.

## V. CONCLUSION

The basis of capsule neural networks and associated routing algorithms are studied in this paper, based on which a new objective on determining routing coefficients between capsules are established. An algorithm targeting at the new routing objective is proposed to achieve faster model convergence and competitive classification results, compared to the baseline results achieved by dynamic routing algorithm on the same capsule network architecture. We also discuss the effectiveness of fully connected reconstruction networks in support of the classification results and visualized counterexamples. Additional researches on development efficient

capsule network architecture and hyper-parameter exploration to compete with state-of-the-art solutions on larger datasets (e.g. ImageNet [20]) will be conducted in our future work.

## REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Conference on Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.
- [2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Conference on Neural Information Processing Systems (NIPS)*, 2014, pp. 2672–2680.
- [3] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.
- [4] G. E. Hinton, Z. Ghahramani, and Y. W. Teh, “Learning to parse images,” in *Conference on Neural Information Processing Systems (NIPS)*, 2000, pp. 463–469.
- [5] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” in *Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 3856–3866.
- [6] G. E. Hinton, S. Sabour, and N. Frosst, “Matrix capsules with em routing,” in *International Conference on Learning Representations (ICLR)*, 2018.
- [7] G. E. Hinton, Z. Ghahramani, and Y. W. Teh, “Learning to parse images,” in *Advances in neural information processing systems*, 2000, pp. 463–469.
- [8] G. E. Hinton, A. Krizhevsky, and S. D. Wang, “Transforming auto-encoders,” in *International Conference on Artificial Neural Networks*. Springer, 2011, pp. 44–51.
- [9] D. Wang and Q. Liu, “An optimization view on dynamic routing between capsules,” 2018.
- [10] A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.
- [11] N. Qian, “On the momentum term in gradient descent learning algorithms,” *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.
- [12] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [13] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.
- [14] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.
- [15] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean *et al.*, “TensorFlow: A system for large-scale machine learning,” in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.
- [16] Y. LeCun, C. Cortes, and C. Burges, “Mnist handwritten digit database,” *AT&T Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, vol. 2, 2010.
- [17] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.
- [18] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Citeseer, Tech. Rep., 2009.
- [19] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *International Conference on Machine Learning (ICML)*, vol. 30, no. 1, 2013, p. 3.
- [20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.