ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200285

An Efficient Agreement Mechanism in CapsNets by Pairwise Product

Lei Zhao[1] and **Xiaohui Wang**[2] and **Lei Huang**[3]

Abstract. Capsule networks (CapsNets) are capable of modeling visual hierarchical relationships, which is achieved by the "routing-by-agreement" mechanism. This paper proposes a pairwise agreement mechanism to build capsules, inspired by the feature interactions of factorization machines (FMs). The proposed method has a much lower computation complexity. We further proposed a new CapsNet architecture that combines the strengths of residual networks in representing low-level visual features and CapsNets in modeling the relationships of parts to wholes. We conduct comprehensive experiments to compare the routing algorithms, including dynamic routing, EM routing, and our proposed FM agreement, based on both architectures of original CapsNet and our proposed one, and the results show that our method achieves both excellent performance and efficiency under a variety of situations.

1 INTRODUCTION

Encoding entities by vectors is widely practiced in the deep models, especially the domains like Natural Language Processing (NLP) and recommendation system. However, this idea is not often used in computer vision tasks such as image classification, and the CapsNets are representative in terms of "vector-wise", which encode objects by collections of neurons called capsules. It is introduced to address the problem of information loss, such as position, size, rotation, scale, which is caused by pooling layers [8]. A capsule represents an object and is often composed of a pose vector/matrix and an activation, where the pose vector/matrix encodes the instantiation parameters of this object. This design aims to disentangle the pose from the evidence of its existence. When the viewing condition changes, the instantiation parameters change, but the capsule still stays active. Such a property is called equivariance and invariance [8, 26], and can be used to build visual hierarchical relationships between capsules of different layers with a characteristic of assigning parts to wholes.

1.1 Routing-by-Agreement Mechanism

Sabour *et al*. [26] introduce dynamic routing to achieve visual hierarchical relationships in CapsNets. It iteratively routes information from lower-level capsules to upper-level ones by a mechanism called "routing-by-agreement", and this is the critical idea of dynamic routing in CapsNets. Hinton *et al*. [8] take an example to explain the "agreement":

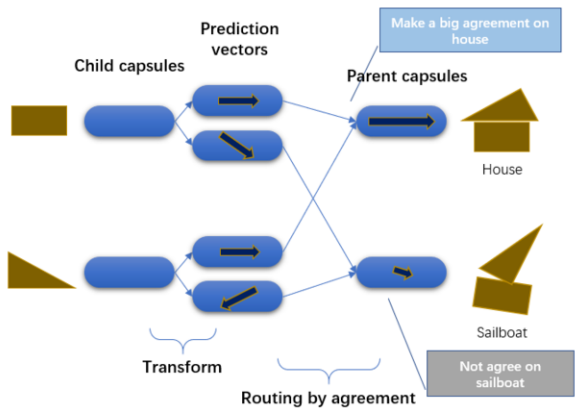


Figure 1. Illustration of the mechanism of routing-by-agreement. We show how the child capsules agree on "House" instead of "Sailboat".

> If, for example, A represents a mouth and B represents a nose, they can each make a prediction for the pose of the face. If these predictions agree, the mouth and nose must be in the right spatial relationship to form a face.

Figure 1 shows an intuitive description to illustrate the mechanism of routing-by-agreement, where each capsule is a vector, and the inner product between them denotes how much they agree with each other. The child capsules make an agreement that "House" is the right prediction instead of "Sailboat". Here comes the core problem of how to compute the "agreement" from these prediction vectors (produced by the child capsules in the lower-layer). The original paper [26] implements this by iteratively computing the vectors' inner product (agreement) between prediction vectors and their corresponding parent capsules in the upper-layer. The more the prediction vectors agree (in terms of vectors' inner product) with a parent capsule, the parent capsule gets a larger activation. The vector length of this parent capsule encodes how strong the child capsules agree with it. However, this implementation has limited performance and an expensive cost both in terms of memory and computation [13, 17, 23], especially with multiple routing iterations. The CapsNet proposed by Sabour *et al*. [26] works well with the MNIST [14] dataset but is not on par with traditional CNNs on more complex datasets such as CIFAR10 [12]. The convolutional CapsNet with EM routing [25] achieves an impressive performance on smallNORB [15] dataset, which demonstrates the capability of modeling viewpoints invariance. However, there are not enough experiments on other kinds of datasets. Moreover, it suffers from a much higher cost of memory

[1] TalkingData, China, email: bhneo@126.com
[2] TalkingData, China, email: xiaohui96@gmail.com
[3] Inception Institute of Artificial Intelligence, UAE, email: huanglei36060520@gmail.com

and computation than the CapsNet in [26]. These problems largely limit the practice of CapsNets to be applied to more challenge tasks.

1.2 "Agreement" in FMs

The prediction of click-through rate (CTR) is to estimate the probability a user clicks on an item in recommendation systems, and it is critical to learn feature interactions behind user click behaviors, such as app category and time-stamp. Rendle and Steffen [24] introduced Factorization Machines (FMs) to capture feature interactions automatically and showed promising results on input features which are sparse and of enormous dimension. FMs model feature interactions as the inner product of latent vectors, which is at "vector-wise" level instead of "bit-wise", this is somewhat corresponding to the capsules in CapsNets. FMs usually consider only order-2 feature interactions in practice. Each feature is embedded into a latent factor vector, and the pairwise feature interactions are modeled as the inner products of latent vectors. This is further reformulated by Rendle and Steffen [24] to make its computation more efficient.

Inspired by FMs, we find that the way it models the pairwise feature interactions can also be used to model the agreements (in terms of pairwise vector inner product) of prediction vectors in CapsNets. The sum of these agreements (each pair of prediction vectors) measures how much the prediction vectors agree with each other. The more they agree with each other, the more the corresponding parent capsule gets activated. An FM algorithm gets much more efficient since it does not need to iteratively compute the parent capsule and its agreements with the prediction vectors. Our contributions include:

- We propose an efficient routing-by-agreement algorithm called FM agreement, which is inspired by the idea of modeling feature interactions in FMs, which outperforms the original dynamic routing and EM routing with a much lower computation complexity.
- We further propose a novel CapsNet architecture that contains residual network blocks and capsule layers, and this architecture turns out to successfully combines both strengths of CapsNets and ResNets.
- We conduct several experiments on benchmark datasets. The results demonstrate that our approaches achieve better performance and efficiency than baselines while retaining the properties of CapsNets.

2 RELATED WORK

The concept of capsules was firstly introduced by Hinton *et al*. [8] to address the representational limitations. Sabour *et al*. [26] proposed CapsNet with dynamic routing algorithm, which achieved a state-of-the-art result on MNIST dataset. Hinton *et al*. [25] proposed a new iterative routing procedure based on the EM algorithm, which achieved an impressive result on smallNORB dataset. After that, the idea of the capsule was applied to many specific tasks to improve performance [3, 13, 36]. Many efforts [1, 17, 21, 31] have been made to seek better capsule architectures. Shahroudnejad *et al*. [27] focused on investigating the potential explainability properties of CapsNets. Lenssen *et al*. [16] proposed "Group-Equivariant-Capsule" to introduce guaranteed equivariance and invariance properties to the CapsNets. It interestingly applied the idea of Lie groups, but with limitations that only a few realizations are applicable in a deep neural network architecture. Recently, Rajasegaran *et al*. [23] explored CapsNets by "going deeper" and proposed DeepCaps, which aimed at improving the performance of the CapsNets for more complex image datasets. It achieved some improvement on the CIFAR10 dataset compared to the work of Sabour *et al*. [26], but which was still very limited when compared to the traditional CNNs such as ResNets [5].

Rendle *et al*. [24] introduced Factorization Machines (FMs) to model feature interactions automatically, it achieved great success in the problem of click-through rate (CTR). Many new architectures related to the idea of FMs have been proposed in recent years, the representative works include FNN [35], PNN [22], DeepCross [28], NFM [7], DCN [32], DeepFM [4], and xDeepFM [18].

Different from the above works, we are the first to introduce the idea of pairwise feature interactions in FMs as a routing-by-agreement mechanism to improve the CapsNets, which aims to provide better performance and efficiency for CapsNets. Our work also explores the capsule architectures, which is related to the work in [23] that explores how to construct deeper CapsNets. However, our work has some differences to the work in [23]: 1) Our work focuses on combining the advantages of CNNs and CapsNets, to achieve better performance and retain the properties of CapsNets, while the work in [23] only construct CapsNets by stack multiple capsule layers; 2) Our work further designs a new routing-by-agreement algorithm.

3 METHOD

For the CapsNet in [26], each child capsule is denoted by a vector $\mathbf{u}_i$, and it first computes a "prediction vector" $\hat{\mathbf{u}}_{j|i}$ for each possible parent capsule by multiplying a weight matrix $\mathbf{W}_{ij}$:

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i. \tag{1}$$

Given these $\hat{\mathbf{u}}_{j|i}$, an agreement mechanism is performed to compute the parent capsules $\hat{\mathbf{u}}_j$ in the upper-layer, which is the crucial point we investigate in this paper. One way to achieve agreement is dynamic routing [26], where the $\hat{\mathbf{u}}_j$ is computed iteratively from a weighted average of prediction vectors $\hat{\mathbf{u}}_{j|i}$, and the agreements are computed by vector inner product between $\hat{\mathbf{u}}_j$ and each $\hat{\mathbf{u}}_{j|i}$:

$$a_{ij} = \left\langle \hat{\mathbf{u}}_{j|i}, \hat{\mathbf{u}}_j \right\rangle. \tag{2}$$

These agreements are used to compute the coupling coefficients c_{ij} for $\hat{\mathbf{u}}_{j|i}$ in the next iteration. Here the coupling coefficients c_{ij} measure how much the $\hat{\mathbf{u}}_j$ and $\hat{\mathbf{u}}_{j|i}$ agree, or how much a child capsule $\hat{\mathbf{u}}_i$ coupled to a parent one $\hat{\mathbf{u}}_j$.

When certain parent capsule $\hat{\mathbf{u}}_j$ couples much to multiple child capsules $\mathbf{u}_i$, this parent capsule becomes activated. The main problem of this dynamic routing algorithm is its expensive cost, both in terms of memory and computation [13, 17, 23]. Moreover, it is of limited performance compared with traditional CNNs like Residual networks [5].

3.1 Agreement Mechanism in FMs

Given feature vectors $\{\mathbf{v}_i\}_{i=1}^n$ with a size of n, the FMs [24] model the pairwise feature interactions as:

$$y := \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i x_j, \tag{3}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of two vectors of size k: $\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}$. Here $\mathbf{v}_i$ describes the i-th variable with k factors. $k \in \mathbb{N}_0^+$ is a hyperparameter that defines the dimensionality of the factorization, which is also corresponding to the number of neurons

in a capsule of this paper. $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ models the interaction between the i-th and j-th variable. x_i and x_j are binary, which denote whether the corresponding $\mathbf{v}_i$, $\mathbf{v}_j$ are activated. So y is the sum of the pairwise interactions of the activated vectors. The computational complexity of Formula 3 is $O(kn^2)$ since all the pairwise interactions have to be computed. For relieving the computation problem, the pairwise interactions can be reformulated as [24]:

$$\begin{aligned}&\sum_{i=1}^{n}\sum_{j=i+1}^{n}\langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \\ =&\frac{1}{2}\sum_{f=1}^{k}\left(\left(\sum_{i=1}^{n} v_{i,f} x_i\right)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2\right)\end{aligned} \tag{4}$$

This equation has a computation complexity of $O(kn)$, which is linear in both k and n.

3.2 Pairwise Agreement in Capsules

Given a bunch of prediction vectors $\left[\hat{\mathbf{u}}_{j|0}, \ldots, \hat{\mathbf{u}}_{j|n},\right]$, we propose to model the "agreement" between capsules following the idea of pairwise interactions as FMs do. The original FM algorithm computes the pairwise interactions by inner product as: $\hat{a}_{j|i_1,i_2} = \left\langle \hat{\mathbf{u}}_{j|i_1}, \hat{\mathbf{u}}_{j|i_2} \right\rangle$, which only models the magnitude of the agreements. Here, we propose to use the element-wise product as: $\hat{\mathbf{a}}_{j|i_1,i_2} = \hat{\mathbf{u}}_{j|i_1} \odot \hat{\mathbf{u}}_{j|i_2}$. Such a method not only can derive the magnitude of the agreements by summing over each element of $\hat{\mathbf{a}}_{j|i_1,i_2}$, but also can represent the orientation. Specifically, we first compute the pairwise interactions of capsules as:

$$\begin{aligned}\hat{\mathbf{s}}_j &= \sum_{i_1=1}^{n}\sum_{i_2=i_1+1}^{n} \hat{\mathbf{u}}_{j|i_1} \odot \hat{\mathbf{u}}_{j|i_2} \\ &= \frac{1}{2}\left(\sum_{i=1}^{n}\hat{\mathbf{u}}_{j|i} \odot \sum_{i=1}^{n}\hat{\mathbf{u}}_{j|i} - \sum_{i=1}^{n}\hat{\mathbf{u}}_{j|i} \odot \hat{\mathbf{u}}_{j|i}\right)\end{aligned} \tag{5}$$

where $\hat{\mathbf{u}}_{j|i} = \left[\hat{u}_{j|i,1}, \ldots \hat{u}_{j|i,k}\right]$, $\hat{\mathbf{s}}_j = [\hat{s}_{j,1}, \ldots \hat{s}_{j,k}]$, and n denotes the number of prediction vectors along the dimension i. Then the activation of the output capsule $\hat{\mathbf{u}}_j$ (the "agreement" of capsules $\mathbf{u}_i$ on $\hat{\mathbf{u}}_j$) can be formulated as:

$$\hat{a}_j = \sum_{f=1}^{k}\hat{s}_{j,f} \tag{6}$$

We define the pose vector as $\hat{\mathbf{p}}_j = \frac{\hat{\mathbf{s}}_j}{\|\hat{\mathbf{s}}_j\|}$. The direction of $\hat{\mathbf{p}}_j$ encodes the properties of an entity such as position, size, rotation, scale, *etc*.

We derive the partial of $\hat{a}_j$ with respect to $\hat{u}_{j|i,f}$:

$$\begin{aligned}\frac{\partial \hat{a}_j}{\partial \hat{u}_{j|i,f}} &= \frac{\partial \hat{s}_{j,f}}{\partial \hat{u}_{j|i,f}} \\ &= \frac{1}{2}\left(\frac{\partial\left(\sum_{i=1}^{n}\hat{u}_{j|i,f}\right)^2}{\partial \hat{u}_{j|i,f}} - \frac{\partial \sum_{i=1}^{n}\hat{u}_{j|i,f}^2}{\partial \hat{u}_{j|i,f}}\right) \\ &= \sum_{i=1}^{n}\hat{u}_{j|i,f} - \hat{u}_{j|i,f}\end{aligned} \tag{7}$$

In practice, we observe that the routing can result in activation/gradient explosion and the potential numeric problem, which is mainly caused by the sum operation shown in Eqn.5 and 6. To avoid it, we argue the key is to ensure the singular value of the Jacobian of $\frac{\partial \hat{a}_j}{\partial \hat{\mathbf{u}}_{j|i}}$ to be near one. From Eqn. 7, we observe that the value of $\sum_{i=1}^{n}\hat{u}_{j|i,f}$ makes the most contribution to the gradient. We thus scale the $\hat{\mathbf{u}}_{j|i}$ by dividing $\sqrt{n}$, which results in:

$$\hat{\mathbf{s}}_j = \frac{1}{2n}\left(\sum_{i=1}^{n}\hat{\mathbf{u}}_{j|i} \odot \sum_{i=1}^{n}\hat{\mathbf{u}}_{j|i} - \sum_{i=1}^{n}\hat{\mathbf{u}}_{j|i} \odot \hat{\mathbf{u}}_{j|i}\right) \tag{8}$$

We find Eqn. 8 well remedies the gradient explosion problem. Furthermore, we apply L2 normalization to the $\hat{\mathbf{u}}_{j|i}$ to make its length be one before agreement routing operation. This process guarantees the agreement value is computed by the direction of vectors and makes the training more stable according to our experiments. We conclude the above process in Algorithm 1, referred to as **FM Agreement**.

Algorithm 1 FM Agreement

Input: prediction vectors $\hat{\mathbf{U}}_j = \left(\hat{\mathbf{u}}_{j|1}, \ldots, \hat{\mathbf{u}}_{j|n}\right)$

Output: $\hat{\mathbf{p}}_j$, $\hat{a}_j$

1: $\hat{\mathbf{u}}_{j|i} \leftarrow$ *L2Normalize*$(\hat{\mathbf{u}}_{j|i})$ $\quad \forall i$
2: $\hat{\mathbf{s}}_j \leftarrow \frac{1}{2n}\left(\sum_{i=1}^{n}\hat{\mathbf{u}}_{j|i} \odot \sum_{i=1}^{n}\hat{\mathbf{u}}_{j|i} - \sum_{i=1}^{n}\hat{\mathbf{u}}_{j|i} \odot \hat{\mathbf{u}}_{j|i}\right)$
3: $\hat{\mathbf{p}}_j \leftarrow \frac{\hat{\mathbf{s}}_j}{\|\hat{\mathbf{s}}_j\|}$
4: $\hat{a}_j \leftarrow \sum_{f=1}^{k}\hat{s}_{j,f}$

We use $\hat{a}_j$ as the prediction of each category for the image classification tasks, where $\hat{a}_j$ indicates how much this category is activated. In the subsequent experiments, we use the softmax cross entropy as the loss function just like the models in [5, 9, 29, 30], for the tasks which predict only one right category (*e.g.*, the classification task shown in Section 4.2, 4.3.1, 4.4). Considering the classification tasks where the number of categories in the image is uncertain (*e.g.*, the experiments in Section 4.3.2), we use the "Margin Loss" [26]:

$$\begin{aligned}L_j =& T_j \max\left(0, m^+ - \hat{a}_j\right)^2 + \\ &\lambda\left(1 - T_j\right)\max\left(0, \hat{a}_j - m^-\right)^2,\end{aligned} \tag{9}$$

where $T_j = 1$ if an object of class j is present. We set $\lambda = 0.5, m^+ = 0.9, m^- = 0.1$ by default. The λ is used to down-weight the loss for absent classes.

3.3 Architecture of One Capsule Layer

To reduce the parameters, we compute the prediction vectors $\hat{\mathbf{u}}_{j|i}$ as the work in [25] does. The capsule described in this paper is represented as a vector of k dimensions. For adapting the method in computing the prediction, we need to reshape it into a $k^{'} \times k^{'}$ matrix where $k^{'}$ should be equal to $\sqrt{k}$, and ensure the trainable weight matrix $\mathbf{W}_{ij}$ between capsule i in layer L and capsule j in layer $L+1$ being also a $k^{'} \times k^{'}$ matrix [25]. By doing this, we get the "prediction matrix" by multiplying the reshaped $\mathbf{u}_i$ and $\mathbf{W}_{ij}$, then reshape it into a k dimensions vector $\hat{\mathbf{u}}_{j|i}$ which is used as the input of the routing algorithm. One advantage of this operation is that we reduce the parameters from $k \times k$ to $\sqrt{k} \times \sqrt{k}$, and there is no performance degeneration, according to our experiments. We set k as 16 in the experiments by default. Before the routing operation, we insert a batch normalization layer [10] to normalize the prediction vectors, which stabilize the training process. The transformation process can be concluded in Algorithm 2.

Figure 2 presents the architecture of a capsule layer. A capsule contains a pose vector/matrix and an activation, and there are differ-

Algorithm 2 Matrix Transformation

Input: $\mathbf{u}_i, \mathbf{W}_{ij}$

Output: $\hat{\mathbf{u}}_{j|i}$

1: $\mathbf{u}_i \leftarrow \mathit{Reshape}(\mathbf{u}_i)$ $\quad 1 \times k \rightarrow \sqrt{k} \times \sqrt{k}$

2: $\hat{\mathbf{u}}_{j|i} \leftarrow \mathbf{u}_i \times \mathbf{W}_{ij}$ $\quad$ Matrix multiplication

3: $\hat{\mathbf{u}}_{j|i} \leftarrow \mathit{Reshape}(\hat{\mathbf{u}}_{j|i})$ $\quad \sqrt{k} \times \sqrt{k} \rightarrow 1 \times k$

4: $\hat{\mathbf{u}}_{j|i} \leftarrow \mathit{BatchNormalization}(\hat{\mathbf{u}}_{j|i})$

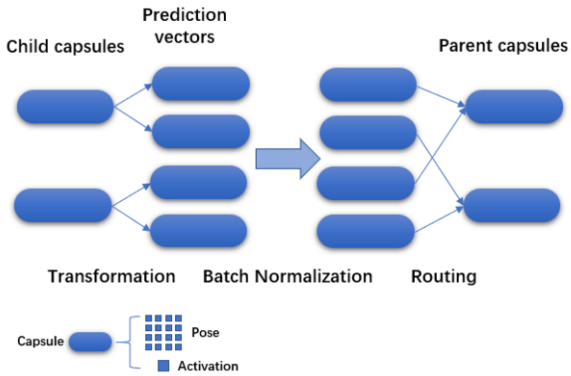


Figure 2. Illustration of one capsule layer. The prediction vectors of input child capsules, corresponding to each parent capsule, are firstly computed, and then transformed into the output parent capsules by a routing procedure.

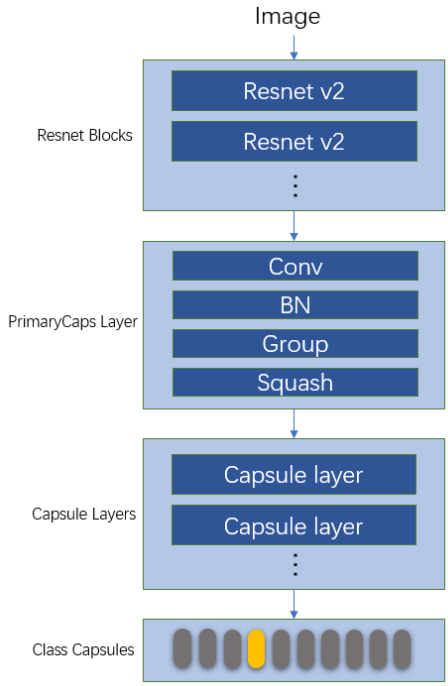


Figure 3. The architecture of the proposed CapsNet. The input image is first fed into the ResNet blocks to produce feature maps, and then transformed into capsules by "PrimaryCaps" layer. These input capsules are used by the following capsule layers to generate output capsules.

ent ways to implement it [25, 26]. In the implementation of [26], the activation is already encoded in the pose vector (because it's defined as the length of the pose vector), and the dynamic routing algorithm only takes the pose vector as its inputs. While in the implementation of [25], they are separated, and the EM routing algorithm takes both pose matrix and activation as its inputs (in which the activation is used in the "m-step"). To adapt our FM agreement to the multiple capsule layers, we make the $\hat{\mathbf{s}}_j$ in Formula 8 as output capsules of the current layer, which are also the input capsules of the next layer.

3.4 Combining with Traditional CNNs

The CapsNets are designed to be a new paradigm to address the problem, such as information loss in traditional neural network architectures. According to the implementations of CapsNets in [25, 26], the capsule layers are built upon a few CNN layers which are used to learn lower level features, and these lower level features are further used by capsule layers to learn parse trees and relationships of parts to wholes [26]. Therefore, the quality of these lower level features does affect the performance of upper capsule layers. Besides, capsule layers are computation expensive, especially applied in lower layers with a large spatial dimension [23]. Rajasegaran *et al.*reduce the computation complexity in their deep capsule model by reducing the number of routing iterations in the lower layers. At the same time, the performance is not affected as the features in the lower layers need not be complex in nature. We believe these lower layers can be replaced by more efficient components. Traditional CNN architectures like VGG [29], ResNet [5], GoogLeNet [30], DenseNet [9], and their many variants have achieved huge success in many computer vision tasks, by learning better visual representation. One straightforward idea is to combine the strengths of both traditional CNNs and CapsNets.

We design the architecture with both ResNet blocks and capsule layers, and such an architecture can be used to investigate the performance of networks and simultaneously explore the properties of the capsule. We apply several ResNet-v2 [6] blocks as the backbone and use a "PrimaryCaps layer" to connect the backbone and its above capsule layers. Inside this PrimaryCaps layer, the tensor provided by the backbone is first to be downsampled by a convolutional layer, then normalized by a batch normalization layer. This tensor is further grouped into capsules by reshaping (*e.g.*, a tensor with shape $H \times W \times C$ is reshaped into $H \times W \times C^{'} \times K$, where H, W, C indicates height, width, channel respectively, and K is the neuron number in a capsule), and activated by a non-linear "squashing" [26] function. We describe the complete architecture in Figure 3.

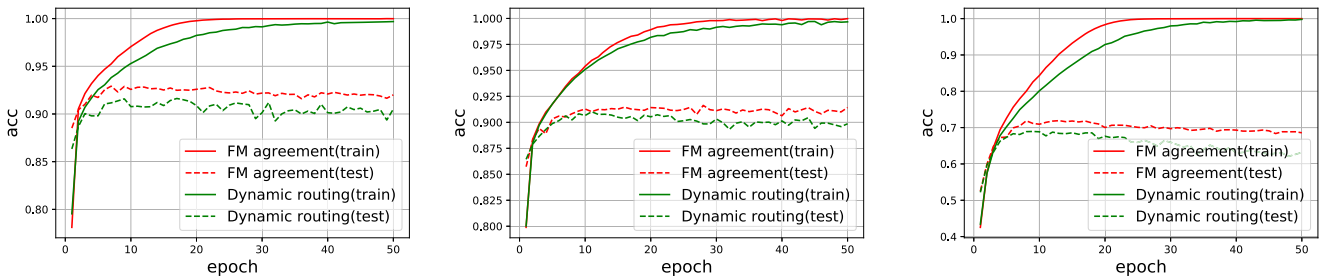
4 EXPERIMENTS

In this section, we conduct experiments to validate the effectiveness of our method in image classification and maintaining the equivariance and invariance.

We implement all the models by Tensorflow,[4] and the experiments include:

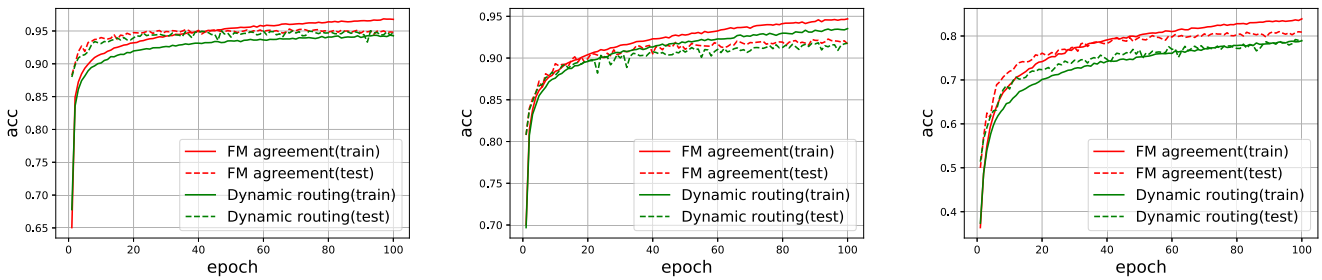
- We compare the classification accuracy and computation cost between different routing algorithms, based on the architecture described in [26].
- We also validate the effectiveness of our proposed architecture shown in Section 3.4, compared to the existing CapsNet architectures on several benchmark datasets.

[4] The code is available at `https://github.com/bhneo/FMRouting`.



(a) cropped-SVHN (b) Fashion-MNIST (c) CIFAR10

Figure 4. Comparison of FM agreement and dynamic routing without data augmentation.



(a) cropped-SVHN+random-crop (b) Fashion-MNIST+random-crop (c) CIFAR10+random-crop+random-flip

Figure 5. Comparison of FM agreement and dynamic routing with data augmentation.

- We investigate the ability of our proposed methods in maintaining the equivariance on image reconstruction tasks.
- Comparison of different methods on the performance of viewpoints invariance by experiments on smallNORB dataset.

4.1 Comparison between Routing Algorithms

In this section, we compare our pairwise agreement algorithm with dynamic routing, based on the original CapsNet architecture [26] without the decoder. We use 3 routing iterations for dynamic routing, following [26], Experiments are conducted on 3 popular benchmark datasets: cropped-SVHN [20], Fashion-MNIST [33], CIFAR10 [12]. We use Adam [11] optimizer (beta1=0.9, beta2=0.999, epsilon=1e-7) with a learning rate of 0.001, batch size of 128. We do not use weight decay to simplify the discussion.

We train the models for 50 epochs without any data augmentation. Figure 4 shows the training and test accuracy with respect to the training epochs. We observe that our FM agreement has better optimization efficiency compared to the dynamic routing, over the three datasets. Besides, our FM agreement obtains significantly better test accuracy than dynamic routing, which suggests its ability in improving the generalization of models.

We also investigate the experiment with data augmentation and run the experiments for 100 epochs. We use only random-crop on cropped-SVHN and Fashion-MNIST, while both random-crop and random-flip-left-right on CIFAR10. The results are shown in Figure 5. We also observe that our FM agreement obtains consistently better training performance than dynamic routing. One interesting observation is that, compared to dynamic routing, FM agreement has nearly the same test accuracy on cropped-SVHN, slightly better on Fashion-MNIST, and much better on CIFAR10. Based on this observation, we conjecture original dynamic routing is not robust on complex datasets with varied backgrounds, while our method is suitable at more complex datasets.

Computational efficiency We also compare the per batch inference time for models trained on CIFAR10 and Fashion-MNIST, which have different model sizes caused by the input sizes (32x32x3 and 28x28x1). The time cost is computed on Nvidia GeForce GTX 1080ti GPU and Intel i7 CPU (3.5GHz). The results shown in Table 1 illustrate that the models using our FM agreement run faster than the dynamic routing both on GPU and CPU.

Table 1. The per-batch inference time. Dynamic routing (x) means using dynamic routing with x iterations.

Methods	CIFAR10 (ms)		Fashion-MNIST (ms)	
	GPU	CPU	GPU	CPU
Dynamic routing (1)	5.79	218.26	4.43	127.95
Dynamic routing (3)	8.89	274.58	7.61	157.69
FM agreement	**5.22**	**203.34**	**3.60**	**125.39**

Table 2. Performance comparison with existing CapsNet architectures on F-MNIST, CIFAR10 and SVHN. We report the classification accuracy with a form of "mean±std", computed on 3 random seeds.

Model	Accuracy (%)		
	F-MNIST	CIFAR10	SVHN
Sabour *et al*. [26]	93.60	89.40	95.70
Hinton *et al*. [25]	-	88.10	-
Nair *et al*. [19]	89.80	67.53	91.06
HitNet [2]	92.30	73.30	94.50
DeepCaps [23]	94.46	91.01	97.16
Ours	**94.70±0.17**	**93.20±0.24**	96.79±0.04

4.2 Performance on Proposed Architecture

In this experiment, we compare our proposed architecture described in Figure 3 with the existing CapsNet architectures on cropped-SVHN, Fashion-MNIST, CIFAR10. We use 25 ResNet blocks and 3 capsule layers (with 32, 16, 10 capsules, respectively). We follow the experimental setup described in the CIFAR10 experiment of [5]. We run the experiments with 3 random seeds and report the performance in Table 2.

From Table 2, we can find that our model outperforms the other CapsNets on CIFAR10 with a significant margin, which further demonstrates the advantage of our model on complicated datasets. Our method also has slightly better performance on F-MNIST, while slightly worse performance on cropped-SVHN, compared to the state-of-the-art model DeepCaps [23]. Note that the parameters of our model are less than 1M, while DeepCaps [23] are 7.22M. We contribute to that our proposed architecture adequately leverages the strength of ResNets [5] in accelerating training and improving the generalization.

4.3 Reconstruction from the Pose Vector

One of the key ideas in CapsNets is providing both equivariance and invariance properties. In this subsection, we investigate such properties by reconstructing the input image from the pose vector. Figure 6 describes the model architecture in this experiment, where we use the architecture in Figure 3 to generate the output capsules. We use the activations ($\hat{a}_j$ in Formula 6) of these capsules to compute the **margin loss** (Formula 9). The pose vectors $\hat{\mathbf{p}}_j$ (Algorithm 1) of these capsules are all masked out except for the correct one (which is from the correct capsule indicated by the label during training, while from the capsule with the largest activation during the test). We then feed this vector into a decoder to reconstruct the input image. Here, we build an efficient decoder with much fewer parameters by applying deconvolutional [34] layers instead of fully-connected layers [26], and only the correct $\hat{\mathbf{p}}_t \in \mathbb{R}^{1\times k}$ is fed into the decoder [23]. With an additional reconstruction loss, the pose vector is encouraged to learn the properties such as rotation, translation, scale, *etc*. We use *Adam* [11] optimizer with the learning rate of 0.0001, and train the model with a batch size of 128.

4.3.1 Aff-NIST

We replicate the experiment in section 5.2 of the original paper [26] to evaluate the capability of our methods in terms of providing both invariance and equivariance. We train our models on a padded and translated MNIST training set, in which each example is an MNIST digit placed randomly on a black background of 40 × 40 pixels.

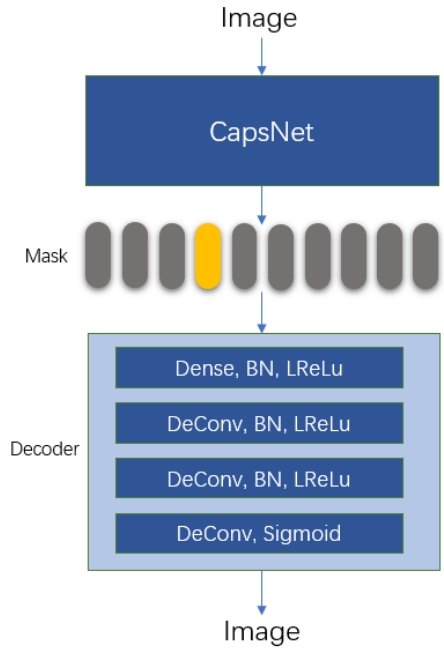


Figure 6. The proposed model architecture which contains a decoder. The capsules from the CapsNet are masked out except for the "correct" one, then the decoder uses the "correct" capsule to reconstruct the input image.

We then test it on the aff-NIST[5] dataset in which each example is an MNIST digit with a random small affine transformation, *i.e.*, the models are never trained with affine transformations other than translation and any natural transformation seen in the standard MNIST. Therefore, the results can indicate whether the models are robust on affine transformations.

We train models based on our architecture but using different routing algorithms (dynamic routing, EM routing, FM agreement) in the capsule layer and compare them on the test set of aff-NIST. We run all the models 3 times and report their accuracy by "mean±std" in Table 3. From the results, we can see that our method using the FM agreement achieves the best accuracy. EM routing is vulnerable to numeric problems in our architecture according to our observation, so its performance is much worse than dynamic routing and FM agreement. Moreover, our model size is largely reduced, which contains about 199K parameters in the decoder and about 196K in the rest components. In contrast, the original CapsNet [26] contains about 2.2M parameters in the decoder and 11.2M in the rest components.

Table 3. Results on the test set of Aff-NIST.

Model	Params	Accuracy(%)
Sabour *et al*. [26]	13.40M	79.00
Hinton *et al*. [25]	-	93.10
Ours (Dynamic Routing)	0.39K	92.06±0.71
Ours (EM Routing)	0.39K	78.29±2.04
Ours (FM agreement)	0.39K	**93.85±0.29**

[5] http://www.cs.toronto.edu/~tijmen/affNIST/

Table 4. Sample reconstructions from the test set of Aff-NIST

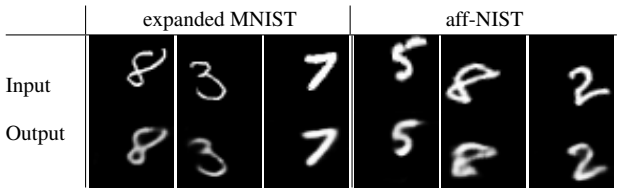
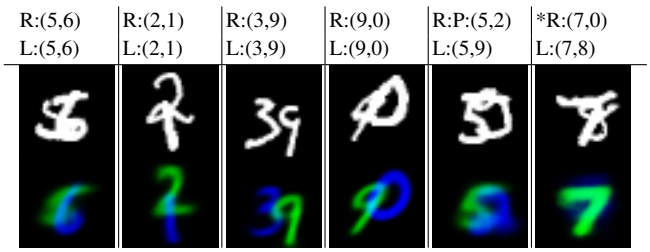


Table 5. Sample reconstructions from the test set of Multi-MNIST. The top row shows the input images with overlapped digits, and the bottom row shows the reconstructed images with segmented digits generated by our CapsNet.



As illustrated in Table 4, the reconstructions from pose vectors of expanded MNIST and aff-NIST successfully learned the crucial details of the input images such as thickness, translation, rotation, which demonstrates our methods can also provide equivariance for the input. We believe it has the potential to build neural networks with better interpretability.

4.3.2 Multi-MNIST

We also replicate the experiment of multi-MNIST in [26]. The dataset in this experiment is created by merging each sample with a digit of other classes. All the samples are shifted up to four pixels randomly in each direction, resulting in a 36x36 image. For each digit image, we generate only 100 Multi-MNIST examples for the training and also 100 for testing, resulting in 6M images for the training set and 1M for the test set, which is a much smaller size compared to the original paper [26].

We achieve a good result with an accuracy of 95.1% on the training set and 94.5% on the test set, which is similar to the result in paper [26]. However, the model is much smaller (457K parameters) compared to the original CapsNet (11.4M).

We display some reconstructed images in Table 5, which shows the pose vectors generated by our model can segment the image into the 2 original digits just like the CapsNet in [26], and we can see that the reconstructions still preserve the styles and positions from original inputs. The top image shows the input, and the lower one shows the reconstructed digits overlayed in green and blue. L:(l1; l2) represents the label for the two digits in the image, P:(p1; p2) represents the prediction for the two digits, and R:(r1; r2) represents the two digits used for reconstruction. The penult column shows two examples with the wrong classification reconstructed from the prediction (P), which confuses digit '9' with digit '2'. The last column with the (*) mark show the reconstruction from a digit that is neither the label nor prediction. The other columns have correct classifications. These results suggest that the model can find the best fit for all the input digits, including the ones that do not exist.

Table 6. Results on the test set of smallNORB.

Model	Iteration	Accuracy(%)
Dynamic Routing	1	91.74±0.86
Dynamic Routing	2	92.73±0.62
Dynamic Routing	3	93.36±0.69
EM Routing	1	91.28±1.02
EM Routing	2	93.55±0.75
EM Routing	3	92.33±0.64
FM agreement	-	**93.60±0.53**

4.4 Viewpoints Invariance on SmallNORB

The smallNORB dataset [15] has gray-level stereo images of 5 classes of toys with every individual toy is pictured at 18 different azimuths (0-340), 9 elevations, and 6 lighting conditions. It is designed to be a pure shape recognition task without the context and color, so it becomes an important benchmark to evaluate the performance of viewpoints invariance. Hinton *et al*. [25] conducted experiments on smallNORB, and their proposed CapsNet with EM routing achieved impressive results.

Based on our proposed architecture in Figure 3, we train models using different routing algorithms to make a fair comparison between our proposed FM agreement and other routing algorithms. We train the model with SGD with a batch size of 64, a momentum of 0.9, and a weight decay of 0.0001. The training starts with a learning rate of 0.01 (which is divided by 2 for every 20 epochs) and is terminated at epoch 120. We also run the experiments with 3 random seeds and show the results with a form of "mean±std" in Table 6. We again observe that our proposed FM agreement achieves the best accuracy, compared to the dynamic routing and EM routing.

5 CONCLUSION

In this paper, we introduce an algorithm called FM agreement to improve the routing procedure in CapsNets. FM agreement is inspired by the FMs, which is widely used in the recommendation system. We use FMs as a "routing-by-agreement" mechanism in the routing procedure, and conduct experiments to compare its performance with dynamic routing and EM routing. The experiments illustrate that our method is of low complexity and outperforms the original dynamic routing and EM routing on several datasets. We further propose a new CapsNet architecture combining both capsule layers and ResNet blocks, and experiments show it not only retains the strong discriminative performance of ResNets but also inherent the capability of CapsNets by providing both equivariance and invariance.

Our future work includes investigating the performance of our CapsNet on more complicated datasets, and further exploring its equivariance and invariance quantitatively. We believe this might be a potential direction for building deep neural networks with better robustness and interpretability.

REFERENCES

[1] Mohammad Taha Bahadori, 'Spectral capsule networks', (2018).

[2] Adrien Deliège, Anthony Cioppa, and Marc Van Droogenbroeck, 'Hitnet: a neural network with capsules embedded in a hit-or-miss layer, extended with hybrid data augmentation and ghost capsules', *arXiv preprint arXiv:1806.06519*, (2018).

[3] Kevin Duarte, Yogesh Rawat, and Mubarak Shah, 'Videocapsulenet: A simplified network for action detection', in *Advances in Neural Information Processing Systems*, pp. 7610–7619, (2018).

[4] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He, 'Deepfm: a factorization-machine based neural network for ctr prediction', *arXiv preprint arXiv:1703.04247*, (2017).

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Deep residual learning for image recognition', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, (2016).

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, 'Identity mappings in deep residual networks', in *European conference on computer vision*, pp. 630–645. Springer, (2016).

[7] Xiangnan He and Tat-Seng Chua, 'Neural factorization machines for sparse predictive analytics', in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 355–364. ACM, (2017).

[8] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang, 'Transforming auto-encoders', in *International Conference on Artificial Neural Networks*, pp. 44–51. Springer, (2011).

[9] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, 'Densely connected convolutional networks', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, (2017).

[10] Sergey Ioffe and Christian Szegedy, 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', in *International Conference on Machine Learning*, pp. 448–456, (2015).

[11] Diederik P Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980*, (2014).

[12] Alex Krizhevsky, Geoffrey Hinton, et al., 'Learning multiple layers of features from tiny images', Technical report, Citeseer, (2009).

[13] Rodney LaLonde and Ulas Bagci, 'Capsules for object segmentation', *arXiv preprint arXiv:1804.04241*, (2018).

[14] Yann LeCun, 'The mnist database of handwritten digits', *http://yann. lecun. com/exdb/mnist/*, (1998).

[15] Yann LeCun, Fu Jie Huang, Leon Bottou, et al., 'Learning methods for generic object recognition with invariance to pose and lighting', in *CVPR (2)*, pp. 97–104. Citeseer, (2004).

[16] Jan Eric Lenssen, Matthias Fey, and Pascal Libuschewski, 'Group equivariant capsule networks', in *Advances in Neural Information Processing Systems*, pp. 8844–8853, (2018).

[17] Hongyang Li, Xiaoyang Guo, Bo DaiWanli Ouyang, and Xiaogang Wang, 'Neural network encapsulation', in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 252–267, (2018).

[18] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun, 'xdeepfm: Combining explicit and implicit feature interactions for recommender systems', in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1754–1763. ACM, (2018).

[19] Prem Nair, Rohan Doshi, and Stefan Keselj, 'Pushing the limits of capsule networks', *Technical note*, (2018).

[20] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng, 'Reading digits in natural images with unsupervised feature learning', (2011).

[21] Sai Samarth R Phaye, Apoorva Sikka, Abhinav Dhall, and Deepti R Bathula, 'Multi-level dense capsule networks', in *Asian Conference on Computer Vision*, pp. 577–592. Springer, (2018).

[22] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang, 'Product-based neural networks for user response prediction', in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 1149–1154. IEEE, (2016).

[23] Jathushan Rajasegaran, Vinoj Jayasundara, Sandaru Jayasekara, Hirunima Jayasekara, Suranga Seneviratne, and Ranga Rodrigo, 'Deepcaps: Going deeper with capsule networks', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10725–10733, (2019).

[24] Steffen Rendle, 'Factorization machines', in *2010 IEEE International Conference on Data Mining*, pp. 995–1000. IEEE, (2010).

[25] Sara Sabour, Nicholas Frosst, and G Hinton, 'Matrix capsules with em routing', in *6th International Conference on Learning Representations, ICLR*, (2018).

[26] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton, 'Dynamic routing between capsules', in *Advances in neural information processing systems*, pp. 3856–3866, (2017).

[27] Atefeh Shahroudnejad, Parnian Afshar, Konstantinos N Plataniotis, and Arash Mohammadi, 'Improved explainability of capsule networks: Relevance path by agreement', in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 549–553. IEEE, (2018).

[28] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao, 'Deep crossing: Web-scale modeling without manually crafted combinatorial features', in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 255–262. ACM, (2016).

[29] Karen Simonyan and Andrew Zisserman, 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556*, (2014).

[30] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, 'Going deeper with convolutions', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, (2015).

[31] Dilin Wang and Qiang Liu, 'An optimization view on dynamic routing between capsules', (2018).

[32] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang, 'Deep & cross network for ad click predictions', in *Proceedings of the ADKDD'17*, p. 12. ACM, (2017).

[33] Han Xiao, Kashif Rasul, and Roland Vollgraf, 'Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms', *arXiv preprint arXiv:1708.07747*, (2017).

[34] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Robert Fergus, 'Deconvolutional networks.', in *Cvpr*, volume 10, p. 7, (2010).

[35] Weinan Zhang, Tianming Du, and Jun Wang, 'Deep learning over multi-field categorical data', in *European conference on information retrieval*, pp. 45–57. Springer, (2016).

[36] Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao, 'Investigating capsule networks with dynamic routing for text classification', *arXiv preprint arXiv:1804.00538*, (2018).