A Context-aware Capsule Network for Multi-label Classification

Sameera Ramasinghe[1,2], C.D. Athuraliya[1], and Salman H. Khan[2]

[1] ConscientAI Labs, Colombo, Sri Lanka
[2] Australian National University, Canberra, Australia
sameera.ramasinghe@anu.edu.au

Abstract. Recently proposed Capsule Network is a brain inspired architecture that brings a new paradigm to deep learning by modelling input domain variations through vector based representations. Despite being a seminal contribution, CapsNet does not explicitly model structured relationships between the detected entities and among the capsule features for related inputs. Motivated by the working of cortical network in HVS, we seek to resolve CapsNet limitations by proposing several intuitive modifications to the CapsNet architecture. We introduce, (1) a novel routing weight initialization technique, (2) an improved CapsNet design that exploits semantic relationships between the primary capsule activations using a densely connected Conditional Random Field and (3) a Cholesky transformation based correlation module to learn a general priority scheme. Our proposed design allows CapsNet to scale better to more complex problems, such as the multi-label classification task, where semantically related categories co-exist with various interdependencies. We present theoretical bases for our extensions and demonstrate significant improvements on ADE20K scene dataset.

1 Introduction

After nearly two decades since its inception, convolutional neural networks (CNNs) [1] have eventually become the norm for computer vision tasks. Vision tasks that widely use CNNs include object recognition [2, 3], object detection [4, 5] and semantic segmentation [6, 7]. Despite their popularity and high effectiveness in most vision tasks, previous works have pointed out several limitations of CNNs in vision applications. One major limitation is the notable trade-off between preserved spatial information and the transformation invariance with pooling operations. Furthermore, CNNs marginally tackle rotational invariance.

To overcome aforementioned limitations in CNNs, recently introduced Capsule Networks (CapsNets) [8] propose a novel deep architecture for feature abstraction while preserving underlying spatial information. This architecture is motivated by human brain function and suggests equivariance over invariance while demonstrating comparable performance on digit classification with MNIST dataset [9]. These early results of CapsNet manifest a new direction for future deep architectures. However to our knowledge, CapsNet architecture has not

been used for larger and complex datasets, specifically for multi-label classification tasks where the goal is to tag an input image with multiple object categories. This is due to the reason that original CapsNet does not incorporate contextual information necessary for complex tasks such as multi-label classification. In this work we evaluate the original CapsNet architecture on a large image dataset with over 150 object classes that appear in complex real-world scenes. We then propose a new context-aware CapsNet architecture that makes informed predictions by exploiting semantic relationships of object classes as well as underlying correlations of low-level capsules. Our model is inspired by the working of human brain where contextual and prior information is effectively modeled [10].

To enable faster training on large datasets, we **first** propose a novel weight initialization scheme based on trainable parameters with back-propagation. This update allows initial routing weights to capture low-level feature distributions and improves the convergence rate and accuracy compared to equal routing weight initialization of the original CapsNet. **Second**, we argue that the corresponding elements of primary capsule predictions are interrelated since primary capsule predictions encapsulate the attributes of object classes. In simple terms, this means that the presence of object attributes (such as position, rotation and texture) in one capsule's output are dependent on similar attributes that are detected by neighbouring capsules. This property was not utilized in the original CapsNet architecture. To characterize this, we introduce an end-to-end trainable Conditional Random Field (CRF) to encourage network predictions to be more context specific. **Third**, the original CapsNet captures the priority between primary and decision capsules independently for each data point. We argue that there exists a general priority scheme between decision and primary capsules, which is distributed across the dataset. Therefore, we propose a correlation module to capture the overall priority of primary capsule predictions throughout the dataset that effectively encapsulates broader context.

We apply proposed architecture for multi-label classification on a large scene dataset, ADE20K [11], and report significant improvements over the original CapsNet architecture.

2 Related Work

Hinton *et al.* [12] first proposed capsule as a new module in deep neural networks by transforming auto-encoders architecture. Capsules were suggested as an alternative to widely adapted subsampling layers of CNNs and to encapsulate more precise spatial relationships. Sabour *et al.* [8] recently proposed a complete neural network architecture for capsules with dynamic routing and a reconstruction loss. They demonstrated state of the art performance on MNIST dataset [9]. They also outperformed existing CNN architectures on a new dataset, MultiMNIST [8], created by overlaying one digit on top of another digit from a different class. More recently, Hinton *et al.* [13] proposed an updated capsule architecture with a logistic unit and a new iterative routing procedure between capsule layers based on the Expectation-Maximization (EM) algorithm [14]. This new capsule

architecture significantly outperformed baseline CNN models on small-NORB dataset [15] and reported that the new architecture is less vulnerable to white box adversarial attacks. Xi *et al.* [16] extended initial CapsNet work by utilizing it on CIFAR10 classification task. However, CapsNet has not been used before for complex structured prediction tasks and our work is a key step towards this direction.

3 Methodology

A decision capsule is considered to be a complete representation of an object class. That means each of its scalar element describes a certain attribute of an object class such as rotation or position. These attributes may not be semantically meaningful, but an object can be completely reconstructed using the elements of the corresponding capsule. Each corresponding element of different decision capsules represents similar attributes of different objects. For example, the i^{th} scalar element of j^{th} decision capsule may represent the rotation of a chair, while i^{th} scalar element of $(j+1)^{th}$ decision capsule may describe the rotation of a desk.

The predictions by primary capsules for decision capsules encapsulate the attributes of an object class. Therefore, the corresponding elements of outputs from primary capsules are conditioned upon each other. For example, there may be a hidden condition such that if the primary capsule is in state A, a chair cannot be rotated in α direction when a spatially nearby desk is rotated in β direction. To exploit this behavior we feed primary capsule predictions to an end-to-end trainable CRF module to learn the inter-dependencies among attributes.

Here, CRF module is used as a structured prediction mechanism for each primary capsule to conditionally alter its predictions. Thus the CRF is able to capture semantic relationships across object classes. Moreover, we introduce a correlation module which can prioritize predictions by primary capsules and effectively predict decision capsules. The overall architecture is illustrated in Figure 1. We first begin with the description of routing weight initialization and then explain the densely connected CRF and the correlation module in subsequent sections.

3.1 Initializing Routing Weights

In the original CapsNet, primary capsules can be interpreted as a set of Z stacked feature maps. Each primary capsule element can be considered as a part of a low-level feature. Following this assumption we rearrange primary capsules as a $N \times N \times D$ grid where $N \times N \times D$ is the total number of primary capsules. Each item in the grid is a capsule with I dimensions. Hence, $D = Z/I$.

Instead of initializing routing weights equally, we modify the initial routing weights as trainable parameters and use backpropagation to train them. This forces the initial routing weights to be dependent on the low-level feature distribution resulting faster convergence.

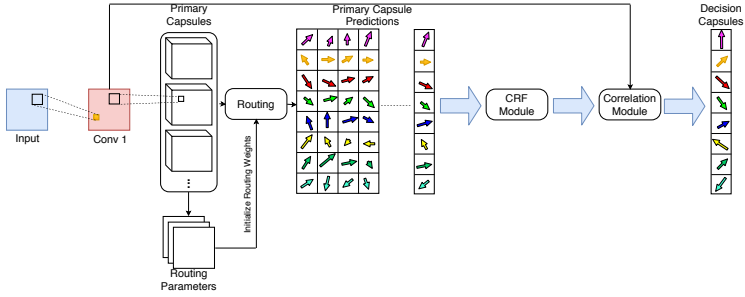

Fig. 1. Proposed CapsNet architecture

To this end, we first define a statistical value per capsule to represent its element distribution. Let K and J be the number of primary and decision capsules respectively, and $C = \{c_1, c_2, \ldots, c_K\}$ be the set of primary capsules. Then we map the capsules to a set $S = \{s_1, s_2, \ldots, s_K\}$ where $s_k = \frac{\mu_k}{max(\sigma_k, \epsilon)}, \forall 0 < k < K, 0 < \epsilon << 1$ and μ_k and σ_k are mean and standard deviation of k^{th} primary capsule elements respectively. $max(\sigma_k, \epsilon)$ gives the maximum value between σ_k and ϵ for each k. The operation outputs a real valued $N \times N \times D$ dimensional tensor. Treating this tensor as a stacked set of feature maps and convolving it with a single $f \times f$ kernel with $(f-1)/2$ padding, where f is a positive integer (we use $f = 5$ in our experiments), give a set of feature maps with dimensions $N \times N \times D$. We obtain $K \times J$ dimension matrix B by transforming the elements of the feature maps as a row vector $\hat{b} = (b_1, b_2, ..., b_K)$, and then repeating it J times. We use elements of B as initial routing weights between primary and decision capsules.

3.2 CRF Module

CRF is an effective technique for structured prediction where output variables are interdependent. Furthermore, CRFs are capable of discriminative training due to conditional probabilistic modeling. They are widely used in important applications of computer vision, natural language processing and bioinformatics. We propose to use CRFs to model relationships between primary capsules in the CapsNet. The CRF models each element of each primary capsule prediction as a random variable and forms a Markov Random Field when the variables are conditioned upon inputs.

Let $P_{k,j}(i)$ denote the i^{th} element of the prediction by k^{th} primary capsule for the j^{th} decision capsule. Considering predictions for all decision capsules by primary capsules, we define the energy function,

$$Z(x) = \sum_{k=0}^{K}\sum_{i=0}^{I}\sum_{j=0}^{J} E_u(P_{k,j}(i)) + \sum_{k=0}^{K}\sum_{i=0}^{I}\sum_{j'=0}^{J}\sum_{j=0,j\neq j'}^{J} E_p(P_{k,j}(i), P_{k,j'}(i)) \quad (1)$$

where $E_u(P_{k,j}(i))$ is cost of prediction $P_{k,j}(i)$ and $E_p(P_{k,j}(i), P_{k,j'}(i))$ is the cost of $P_{k,j}(i)$ and $P_{k,j'}(i)$ occurring simultaneously. It is evident from Equation

1 that pairwise potentials only take corresponding elements of predictions by the same primary capsule in to account. Therefore minimizing energy function in Equation 1 is equivalent to minimizing each $Z(x)_{k,i}$ for $i < I$ and $k < K$ independently where,

$$Z(x)_{k,i} = \sum_{j=0}^{J} E_u(P_{k,j}(i)) + \sum_{j'=0}^{J} \sum_{j=0, j\neq j'}^{J} E_p(P_{k,j}(i), P_{k,j'}(i)) \tag{2}$$

Mean-field approximation provides an iterative approach to minimize dense CRF energy functions. This technique approximates a total energy function $Z(x)_{k,i}$ as a product of simple marginal energy functions $Z(X)_{k,i} = \prod_l H^l_{k,i}(x_l)$.

Zheng *et al.* [17] leveraged this idea by formulating a dense CRF as a stack of differentiable layers. They also showed that multiple iterations of this stack of layers can be treated as an RNN. We adapt this technique to minimize the energy function 2.

Algorithm 1. CRF as a stack of CNN layers

1: $H_{kj}(i) = \frac{1}{X_{ik}} exp(E_u(P_{k,j}(i)) \forall i, j, k$ ▷ Initialization
2: **for** $itr = 0$ **to** $MaxItr$ **do**
3: $\bar{H}_{kj}(i) = \sum_{j'} E_p(H_{kj}(i), H_{kj'}(i))$ ▷ **Calculation of pair-wise potentials**
4: $\tilde{H}_{kj}(i) = H_{kj}(i) - \bar{H}_{kj}(i)$ ▷ **Addition of pair-wise potentials to unary potentials**
5: $H_{kj}(i) = \frac{1}{X_{ik}} e^{\tilde{H}_{kj}(i)}$ ▷ **Normalization**
6: **end for**

The first line is the initialization. Here, $X_{i,k} = \sum_{j=0}^{J} e^{P^k_{ij}}$ where J is the number of decision capsules. Since $E_u(P_{k,j}(i))$ is the cost of the i^{th} element of the prediction, we can treat the predicted element as $P_{k,j} = -E_u(P_{k,j}(i))$. This is equivalent to applying the softmax function over each set of i^{th} elements of the predictions by k^{th} primary capsule for j^{th} decision capsules. Line number 3 illustrates the cost of pair-wise potentials. Instead of deriving the pair-wise potential function manually, using back-propagation to find optimum mapping is both effective and efficient. Since all the corresponding element pairs have to be taken into account, we apply a fully connected layer on top of the predictions to learn this pair-wise potential function. Since we are minimizing $Z(x)_{k,i}$ for each i and k independently, these layers are not connected across i or k, which reduces the computational complexity significantly. Line number 4 illustrates adding the unary potentials to pair-wise potentials. Line number 5 is equivalent to applying softmax function over the outputs.

3.3 Correlation Module

In the CapsNet architecture, each primary capsule has a unique prediction for each decision capsule. Since primary capsules are essentially a set of low-level features, this can be viewed as each low level feature estimating the state of the

output class. Moreover, each low-level feature priority depends on the output class. For example, a circle detector may perform better in predicting the state of a wheel, while a horizontal edge detector may perform better in predicting the state of a bridge.

The original routing technique tries to capture these varying priorities of primary capsules with respect to decision capsules by a weighted sum of predictions. The routing weights are adjusted in the next iteration according to the similarity between primary capsule predictions and the decision capsule of the current iteration. The magnitude of similarity is estimated by dot product. Following this method the network learns the priorities independently for each data point. However, we argue that there is also a general priority scheme that is distributed across the whole dataset, that can be learned during the training. Therefore, we propose a novel correlation based approach to discover these priorities and estimate final prediction.

Unlike the CRF module, our objective here is to find correlation between the attribute distributions of corresponding predictions of primary capsules and a decision capsule, instead of finding the dependency between each single corresponding attribute of predictions. Given a set of predictions for a specific decision capsule, the goal of the correlation module is to find the decision capsule elements by exploiting priorities of each primary capsule. The correlation coefficients between a decision capsule and a primary capsule predictions are learned throughout the training. Furthermore, these correlation coefficients should depend on the low-level feature distribution and also should be trainable. To this end, we use a property of Cholesky transformation [18] and derive a generic function to achieve this task.

Let two distributions be Q and R. Cholesky transformation ensures,

$$\begin{bmatrix} \bar{Q} \\ \bar{R} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \rho_1 & \sqrt{1-{\rho_1}^2} \end{bmatrix} \begin{bmatrix} Q \\ R \end{bmatrix} \tag{3}$$

$$\bar{Q} = R, \bar{R} = \rho_1 Q + \sqrt{1-{\rho_1}^2} R \tag{4}$$

and produces two distributions $\bar{Q}$, $\bar{R}$ which are correlated by a factor of ρ_1. Likewise,

$$\begin{bmatrix} \bar{\bar{Q}} \\ \bar{\bar{R}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \rho_2 & \sqrt{1-{\rho_2}^2} \end{bmatrix} \begin{bmatrix} R \\ Q \end{bmatrix} \tag{5}$$

$$\bar{\bar{Q}} = Q, \bar{\bar{R}} = \rho_2 R + \sqrt{1-{\rho_2}^2} Q \tag{6}$$

produces two distributions $\bar{\bar{Q}}$, $\bar{\bar{R}}$ which are correlated by a factor of ρ_2. Therefore if we choose,

$$\rho_2 = \sqrt{1-{\rho_1}^2} \tag{7}$$

we get $T = \bar{R} = \bar{\bar{R}}$, where T and R are correlated by ρ_1 and, T and Q are correlated by ρ_2. Using this property and considering two component distributions $D_1 = P_{k,j}$ and $D_2 = P_{k',j}$, where $P_{k,j}$ is the component distribution of k^{th}

primary capsule prediction for j^{th} decision capsule, we obtain a new distribution $\hat{D}$, satisfying $\rho_{\hat{D},D_1} = \rho_1$, and $\rho_{\hat{D},D_2} = \alpha\rho_1$. Here, ρ_{x_1,x_2} denotes the correlation between the two particular distributions x_1 and x_2. Using Equation 7,

$$\frac{\rho_1}{\alpha} = \sqrt{1-{\rho_1}^2}, \rho_1 = \frac{\alpha}{\sqrt{1+\alpha^2}} \tag{8}$$

$$\hat{D} = \big[\frac{\alpha}{\sqrt{1+\alpha^2}}D_1 + \frac{1}{\sqrt{1+\alpha^2}}D_2\big] \tag{9}$$

Using Equation 9, we define a recursive function f_ρ to obtain a correlated element distribution.

$$f_\rho(P_{1,j}|P_{2,j}\ldots,P_{k,j},\ldots,P_{K,j}) = \frac{\alpha_K}{\sqrt{1+\alpha_K^2}} f_\rho(P_{1,j}|P_{2,j}\ldots,P_{k,j},\ldots,P_{K-1,j}) + \frac{P_{K,j}}{\sqrt{1+\alpha_K^2}}, \forall 0 < k \leq K, 0 < j \leq J \tag{10}$$

where $f_\rho(P_{1,j}|P_{2,j}) = \big[\frac{\alpha_2}{\sqrt{1+\alpha_2^2}}P_{1,j} + \frac{1}{\sqrt{1+\alpha_2^2}}P_{2,j}\big]$. Using this derivation, we obtain the j^{th} decision capsule $C_j = f_\rho(P_{1,j}|P_{2,j}\ldots,P_{k,j},\ldots,P_{K,j})$. Here, α requires be trainable and dependent on low-level feature distributions. Since the above operation is differentiable, the first criteria is fulfilled. To enforce α to be dependent on low-level features, we use the following method.

Consider a $N \times N$ low-level feature map. Since we need $J(K-1)$ trainable parameters as per Equation 10, we convolve this particular feature map with a set of $J(K-1)$ kernels with sizes $N \times N$ each. This outputs $J(K-1)$ number of scalar values, which can be used as α parameters.

4 Experiments

We conduct experiments to demonstrate the effectiveness of each of the improvements; new initialization scheme of routing weights, CRF module and the correlation module. We use mean average precision (mAP) as the evaluation metric throughout the experiments with precision threshold 0.5. We use ADE20K dataset to evaluate the proposed architecture given its complex scenes and rich multi-label annotations for training images. ADE20K provides over $20,000$ training and testing images annotated with 150 semantic object categories.

4.1 Importance of Trainable Initial Routing Weights

The goal of replacing the equal initialization of routing weights with trainable weights is faster convergence. In order to test the significance of this, we train the proposed architecture with and without the trainable initial routing scheme and test the validation mAP. The results are illustrated in Figure 2.

As shown in Figure 2 the validation mAP stabilizes around 15^{th} epoch for the CapsNet without the proposed routing weight initialization method. On the contrary, the CapsNet with the proposed routing weight initialization method stabilizes around 9^{th} epoch. Therefore it is evident that the proposed method is able to achieve faster convergence compared to equal initial routing weights.

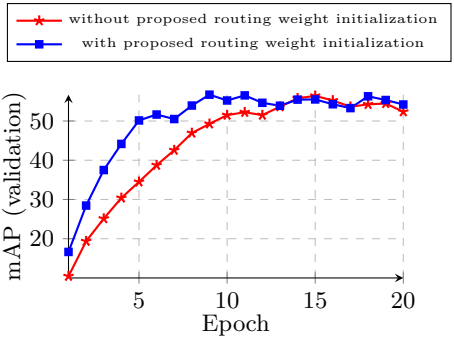

Fig. 2. Evaluation of convergence gain by trainable initial routing weights.

Method	mAP
Original CapsNet	42.38
Ours (RW + CRF)	52.50
Ours (RW + CRF + CORR)	**56.71**

Table 1. Comparison of the proposed architecture with the baseline

4.2 Comparison with the Baseline

We compare the original CapsNet architecture with the proposed one by measuring the mAP measure. Table 1 shows the comparison results. We gain a significant 14.33 mAP gain over total 150 object classes compared to the original architecture. Furthermore, we demonstrate performance gains by CRF and correlation modules, and show that each module provides complementary improvements. We gain an improvement of 10.12 mAP by adding the CRF module and a 4.21 mAP improvement by adding the correlation module on top of the CRF module. All the architectures are trained for 20 epochs.

5 Conclusions

In this work we attempt to overcome several limitations of CapsNet by introducing an improved architecture inspired by the contextual modeling in visual cortex [10]. Our objective is two fold: effectively capture complex interactions between primary capsules and leverage data wide correlations between representations of similar inputs. To this end, we introduced three novel ideas. Firstly, we proposed a new routing weight initialization that can be trained using back-propagation. This replaced existing equal initial routing weights with a more intuitive and efficient technique. Secondly, we introduced a CRF based method to exploit conditional attributes of primary capsule predictions to capture the context of neighbouring objects. Thirdly, we proposed a correlation module to learn dataset-wise priority scheme instead of capturing the priority separately for each data point. As demonstrated through our experiments, these improvements in CapsNet model design contributes to a substantial accuracy improvement of over 33% in multi-label classification on a challenging dataset.

References

1. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. (1998) 2278–2324
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. (2012) 1097–1105
3. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 1–9
4. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems 28. (2015) 91–99
5. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640 (2015)
6. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3431–3440
7. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2017)
8. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. In: Advances in Neural Information Processing Systems. (2017) 3859–3869
9. LeCun, Y., Cortes, C., Burges, C.J.: The mnist database of handwritten digits. (1998)
10. Bar, M.: Visual objects in context. Nature Reviews Neuroscience **5**(8) (2004) 617
11. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ADE20K dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017)
12. Hinton, G.E., Krizhevsky, A., Wang, S.D.: Transforming auto-encoders. In: Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I. (2011) 44–51
13. Hinton, G.E., Sabour, S., Frosst, N.: Matrix capsules with EM routing. International Conference on Learning Representations (2018)
14. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B (1977) 1–38
15. LeCun, Y., Huang, F.J., Bottou, L.: Learning methods for generic object recognition with invariance to pose and lighting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2004) 97–104
16. Xi, E., Bing, S., Jin, Y.: Capsule network performance on complex data. arXiv preprint arXiv:1712.03480 (2017)
17. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1529–1537
18. Ramasinghe, S., Rajasegaran, J., Jayasundara, V., Ranasinghe, K., Rodrigo, R., Pasqual, A.A.: Combined static and motion features for deep-networks based activity recognition in videos. IEEE Transactions on Circuits and Systems for Video Technology (2017) 1–1