Supplementary Material: Rectification-based Knowledge Retention for Continual Learning

Pravendra Singh[1,*], Pratik Mazumder[2,*], Piyush Rai[2], Vinay P. Namboodiri[2,3]
[1]Independent Researcher, India [2]IIT Kanpur, India [3]University of Bath, United Kingdom
pravendra1988@gmail.com, pratikm@cse.iitk.ac.in, piyush@cse.iitk.ac.in, vpn22@bath.ac.uk

1. Task Incremental Learning (Generalized Zero-Shot Setting)

We use our approach RKR to work for the task incremental generalized zero-shot learning problem setting described in [8]. The authors in [8] experimentally show that CADA-VAE [6] suffers from catastrophic forgetting in the image/visual features encoder when trained in the task incremental generalized zero-shot learning setting. They propose LZSL to solve this problem. For a fair comparison with LZSL, we use the same setting, setup, and base architecture (CADA-VAE) as used in LZSL [8].

1.1. CADA-VAE

In this section, we provide a brief overview of the CADA-VAE framework. For complete details, please refer to [6]. In the generalized zero-shot learning setting, there are S seen classes and U unseen classes, and we have labeled training examples for the seen classes only. The test images can be from both the seen and unseen classes. For each seen and unseen class, we have access to their class/attribute embeddings, which are generally vectors of hand-annotated continuous attributes or Word2Vec [3] features. Zero-shot learning methods leverage the class/attribute embeddings to transfer information from seen classes to the unseen classes. Similar to most zero-shot learning methods, CADA-VAE operates on image features extracted by a pre-trained network (ResNet-101).

The CADA-VAE framework consists of a variational autoencoder (VAE) for image/visual features (E_v, D_v) and a VAE for the class/attribute embeddings (E_a, D_a), each having an encoder and decoder (see Fig. 1). The two encoders project the image features and class embeddings to the common latent embedding space, respectively, and the decoders reconstruct the image features and class embeddings from their latent embeddings. Specifically, the image features encoder (E_v) maps the image features to μ_v and Σ_v in the latent embedding space. Similarly, the class embeddings encoder (E_a) maps the class embeddings to μ_a and Σ_a. CADA-VAE learns a common latent embedding space for both the image/visual features and the class/attribute embeddings and brings the latent embeddings of the image features and class embeddings closer in the latent embedding space. It utilizes cross-alignment loss (CA) and distribution-alignment loss (DA) apart from the VAE loss to achieve this objective. Cross-alignment involves training the class embeddings decoder to generate corresponding class embedding from the latent features of the images of that class and vice-versa. Distribution-alignment involves training the encoders of the image features and class embeddings to minimize the Wasserstein distance between the Gaussian distributions of their latent embeddings. After training the VAEs, CADA-VAE uses the μ_a and Σ_a of all the seen and unseen class embeddings to sample embeddings (using the reparametrization trick) for both the seen and unseen classes from the learned latent embedding space. It trains a classifier using these latent embeddings in order to classify the test images. At test time, the pre-trained network extracts image features from the test images. The image/visual features encoder (E_v) maps the test image features to the latent embedding space. The classifier then predicts the class for the test image latent embeddings.

Task Incremental Generalized Zero-shot Learning: The authors in [8] apply CADA-VAE to a task incremental generalized zero-shot learning setting where each task is a separate dataset and each task contain seen and unseen classes. After training the VAEs on a task t, embeddings can be sampled from the latent embedding space using the μ_a^t and Σ_a^t of all the seen and unseen classes. These latent embeddings are used to train a classifier. The classifier will be able to predict the classes of the test image embeddings of that task produced by the visual features encoder (E_v^t). However, when the network is trained on a new task $t+1$, the image/visual features encoder (E_v^{t+1}) weights will change. As a result, the test image features (input to E_v^{t+1}) for the test images from a previous task will get mapped to a different latent space (output of E_v^{t+1}) than the one obtained just after training the network on that task. Since the classifier will classify on the basis of the output of E_v^{t+1}, the predictions for the test images from the previous tasks will be

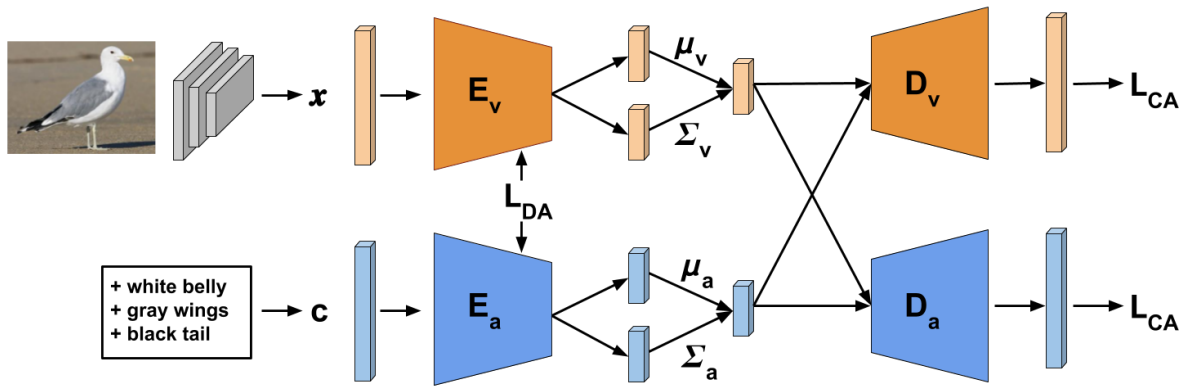

Figure 1: Illustration of CADA-framework.

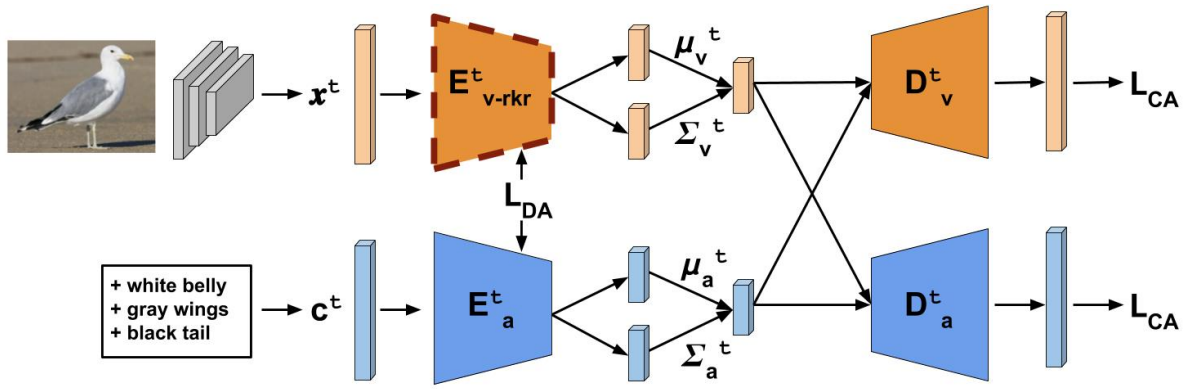

Figure 2: Illustration of CADA-framework with our proposed RKR image/visual features encoder (E^t_{v-rkr}). The framework consists of a pre-trained network for extracting image features and two variational autoencoders, one for the image/visual features and the other for the class/attribute embeddings. E^t_{v-rkr} is our proposed RKR image/visual features encoder for task t (shown with a dashed border), that adapts the network for task incremental learning. Image feature x^t, extracted by the pre-trained network from an image, is fed to E_{v-rkr}, which maps it to μ^t_v and Σ^t_v in the latent space. The attribute encoder E^t_a maps the attribute/class embedding c^t of that image to μ^t_a and Σ^t_a in the latent space. The network is trained on the standard VAE loss, cross-alignment loss L_{CA} and distribution-alignment loss L_{DA}.

incorrect. Therefore, the CADA-VAE performance suffers in the task incremental generalized zero-shot learning setting due to the catastrophic forgetting in the visual features encoder. The authors in [8] propose LZSL to tackle this problem by using selective parameter retraining and knowledge distillation to preserve previous task knowledge.

Applying RKR: In order to prevent catastrophic forgetting, we apply RKR to the image/visual features encoder E^t_v, that only contains fully connected layers, to obtain E^t_{v-rkr} (Fig. 2). Specifically, we use weight rectifications and scaling factors for each layer in the image/visual encoder to quickly adapt it to any task. We train the full network on the first task (base network). For every new task, we only learn weight rectifications and scaling factors for all network layers to adapt them to the new task. In the generalized zero-shot learning setting, we learn the weight rectifications and the scaling factors based on the seen classes of the given task and use them during testing for classifying both seen and unseen classes of that task. Therefore, during testing, the image features encoder will map the test image features for each task to the same embedding space as expected by the classifier.

1.2. Datasets

For the task incremental generalized zero-shot learning problem, we experiment with the Attribute Pascal and Yahoo (aPY) [1], Animals with Attributes 1 (AWA1) [9], Caltech-UCSD-Birds 200-2011 (CUB) [7], and SUN Attribute dataset (SUN) [5] datasets. Other details regarding the datasets are provided in Table 1. For a fair comparison, we use the same sequence of training datasets given in [8], which is aPY, AWA1, CUB, and SUN. However, we also report the results for three other cases with a different first dataset. We report the average per-class top-1 accu-

Table 1: Datasets used in the task incremental generalized zero-shot learning problem.

Dataset	Class Embedding Dimensions	Images	Classes	
			Seen	Unseen
aPY	64	15339	20	12
AWA1	85	30475	40	10
CUB	312	11788	150	50
SUN	102	14340	645	72

racy for the unseen classes (U), seen classes (S), and the harmonic mean of the two accuracies (H) for each dataset ($H = \frac{2\times U\times S}{U+S}$). Our objective is to achieve high H accuracy as it is not skewed towards either the seen or unseen classes. The results are obtained after the training has been completed on all the datasets.

1.3. Implementation Details

For our experiments, we extract the image features of 2048 dimensions from the final pooling layer of an ImageNet pre-trained ResNet-101. In the case of image features, the encoder and decoder of CADA-VAE have 1560 and 1660 hidden layer nodes, respectively. In the case of class embeddings, the encoder and decoder have 1450 and 660 hidden layer nodes, respectively. The latent embeddings are of size 64. For all the datasets, the model is trained for 100 epochs with batch size 50 using the Adam optimizer [2]. CADA-VAE also uses a few hyper-parameters, i.e., δ, γ, β. δ is increased from epoch 6 to epoch 22 by a rate of 0.54 per epoch, while γ is increased from epoch 21 to 75 by 0.044 per epoch. The β weight of the KL-divergence term is increased by a rate of 0.0026 per epoch up to epoch 90. A learning rate of 0.00015 is used for the VAEs, and a learning rate of 0.001 is used for the classifiers. L1 distance is used for the reconstruction error. These settings have been proposed in [6] and were also used in [8]. We report the average results of five runs for our method.

2. Performance

For task incremental learning, we perform five runs of every experiment for both the zero-shot and non zero-shot settings and report the average accuracy. The variations in our results are very low, e.g., for CIFAR-100 with ResNet18 and LeNet, the 95% confidence interval for the final average session accuracy is $87.6 \pm 0.467\%$ and $69.58 \pm 0.55\%$ respectively.

3. Hardware and Software Specifications

We have performed all our experiments in PyTorch version 0.4.1 [4] and Python 3.0. For running our experiments, we have used a GeForce GTX 1080 Ti graphics processing unit.

References

[1] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785. IEEE, 2009.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[4] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch, 2017.

[5] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2751–2758. IEEE, 2012.

[6] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8247–8255, 2019.

[7] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset, 2011.

[8] Kun Wei, Cheng Deng, and Xu Yang. Lifelong zero-shot learning. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 551–557. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Main track.

[9] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265, 2018.