TRIZSL - Zero shot Learning under Low Resource Data

Final Report - COMS 6998 Sec 011 Deep Learning System

Liang-yu Charles Chen, Sri Hari Thikkireddy: {lc3533,srt2157}@columbia.edu

1. Goals

In our course project we aim to find an innovative solution to perform attribute learning with a small dataset and reach comparable accuracy on zero shot prediction. We propose an inductive generalized zero shot learning (GZSL) framework - TRIZSL on the GZSL task where the test data is unknown and either is its distribution. Our method to reach a better accuracy with the minimal size of the dataset on zero-shot learning compared to the solution proposed by [1], which was the previous state-of-the-art attribute embedding method in 2017. Given a target accuracy, we show the size of the training dataset we subsample from the original dataset [4] needed by our method to be less than which of [1].

2. Challenges

Training on zero shot learning tasks often requires generative models [5,6,7] to generate synthesized unseen dataset and train the classifier based on a given set attribute. This often requires a huge amount of computational resources. Therefore, we aim to create a framework where we can quickly train a model even when such computational resources are not available.

Furthermore, labelling attributes and collecting datasets with various classes of such attributes can be extremely expensive. We aim to make the most of the data and reduce the number of required data per class as much as possible to achieve a certain performance.

3. Approach/Techniques

3-1. Generalized Zero Shot Learning

Given a dataset with $|N|$ classes and $|P|$attributes, there is a binary predicate matrix $M \in \mathbb{R}^{|N|\times|P|}$ where each row denotes the predicate for a class, and each column denotes if the i th attribute $p_i \in P$exists across all classes. As the dataset will contain seen classes and unseen classes, we denote seen classes to be N_{seen} and unseen to be N_{unseen}. In our the case of GZSL

$N = N_{seen} \cup N_{unseen}$ and $N_{seen} \cap N_{unseen} = \emptyset$. Note that in the setup of GZSL we will have access to the attribute predicates for both seen dataset and unseen dataset, so we will first check if every attribute $p_i \in P$ is distinguishable among N_{seen}. By distinguishable it means that for a column $C_{p_i} \in \mathbb{R}^N$, C_{p_i} should have at least two different values. If there is a C_{p_i} not distinguishable, we will remove that column. The same attribute cleaning process will be done for N_{unseen} as well. Therefore, we will have $M' \in \mathbb{R}^{N \times P'}$ in the end, where P' derived by P removing indistinguishable attributes from both N_{seen} and N_{unseen}.

3-2. Attribute Embedding

We adopt a ResNet101 as a feature extractor to map our input image in a 2048 dimensional vector. And we iterate $p_i \in P'$ and label data in each classes into with binary values based on p_i. Formally, for all data points belong to ith class n_i, we label data according to its binary predicate value M'_{n_i, p_i}, and we train an embedder E_{p_i} based on the labels with triplet loss. Recall that triplet loss calculates the L2 norm of an anchor input, a positive input and a negative input in each training step. If given an attribute $p_i \in P'$ to train, we partition our trainsing data $R \subset N_{seen}$ to be a split of the attribute label being true $R^{p_i=1}$ and false $R^{p_i=0}$, where $R^{p_i=1} \cup R^{p_i=0} = R$, we sample a training tuple ($A \sim R$, $T \sim R^{p_i=1}$, $F \sim R^{p_i=0}$). If we exhaustively sample every possible combination, we will have $O(R^3)$ different input for a single epoch which is data efficient. When the training is done and E_{p_i} converges, we calculate mean vectors for all embedded true instances and embedded false instances, say $K_T^{p_i}$ and $K_F^{p_i}$, where

$$K_T^{p_i} = \frac{1}{|R^{p_i=1}|} \sum_{I \in R^{p_i=1}} E_{p_i}(Res(I))$$

$$K_F^{p_i} = \frac{1}{|R^{p_i=0}|} \sum_{I \in R^{p_i=0}} E_{p_i}(Res(I))$$

We thus have $|P|$ embedders at the end, along with $|P|$ true mean vectors and false mean vectors.

3-3. Rule Based Inference

When doing inference with a image I, we will embed this input as a continuous predicate, which is a row vector with $|P|$ dimensions. We calculate the cosine similarity of $E_{p_i}(Res(I))$ and both $K_T^{p_i}$, $K_F^{p_i}$ to get the attribute-wise probability

$$S_{p_i} = \frac{S^{T_{p_i}}}{S^{T_{p_i}} + S^{F_{p_i}}}$$

where $S^{T_{p_i}}$ is the cosine similarity between $E_{p_i}[Res(I)]$ and $K_T{}^{p_i}$ and the similar applies to F_{p_i}. Formally,

$$S^{T_{p_i}} = CosSim(E_{p_i}[Res(I)],\ K_T{}^{p_i})$$

$$S^{F_{p_i}} = CosSim(E_{p_i}[Res(I)],\ K_F{}^{p_i})$$

So each input instance I will be mapped to a Pdimensional vector at the current stage. Let $S_p = [S_{p_1}, S_{p_2}, ..., S_{p_{|P'|}}]$be this vector from I , we calculate the Mean Absolute Error (MAE) between each $M'_{n_i,*}$ and S_p and pick the n_i that minimize this error to be our prediction. This n_i will be the output classification label. Specifically, the prediction rule will be:

$$Prediction(I) = argmin_{n_i \in N} MAE(M'_{n_i,*}, S_p)$$

4. Implementation Details

We will adopt a ResNet101 which is pre-trained on ILSVRC-12 as the feature extractor. In terms of dataset, we are using the Animals with Attributes 2 dataset (AwA2, [4]) which contains 37k images of 50 animals. We will adopt the proposed split of AWA2, where it splits classes into 40 seen classes and 10 unseen classes, and 85 attribute predicates across all classes. We further split the seen data into training and test sets, where the training set contains 23k images and the test test contains 7k images. This split will guarantee there is no overlapping between ILSVRC-12 and AWA2 on training data. Consequently, we will have 2 test datasets: seen test and unseen test, with 40 and 10 classes containing instances not seen during training respectively. We will further calculate the precision@1 on each, which is denoted as seen accuracy and unseen accuracy.

In terms of hardware support, even though our method aims to train a classifier with few resources available, we will adopt a NVIDIA Tesla P100 GPU to accelerate the entire process. We will also leverage Tensorflow2.4 and TF-Addons as a deep learning framework to conduct our experiment. We will be using models from Tensorflow2.4, and triplet loss from TF-Addons.

5. Experiment Setting

Our experiment aims to study the relationship between the amount of data used per class against both the seen and unseen accuracy. We will provide a detailed analysis on this by selecting training data per class from 1 image to 2, 5, 10, 20 until we use all. This setting with few data also helps us demonstrate how our model performs in terms of few shot learning with the seen accuracy of 1/2/5/10 image per class. The same model trained on 1/2/5/10 image per class will also be used to infer unseen accuracy, so we study the performance in both setups of few shot learning and zero shot learning. Also, as our method proposes using a 2-layer fully connected neural network to perform attribute embedding, we will also show an analysis on the performance versus the embedding dimension and the hidden layer size. We will train embedders with 1 layer with output size to be the 1 to 11th power of two, namely [2,4,....2048], as well as 2 layers with hidden size to be [128, 512, 2048] with the same selection of the output size.

6. Results

On inductive GZSL, our proposed method reaches 63% harmonic mean of the unseen precision@1 and seen precision@1 with hidden size (F) = 512 and output size (S) = 256, with only 150 input per class (~25% of the total dataset). The following is the detailed performance on the benchmarks we are comparing to:

	Seen Prec@1	Unseen Prec@1	Harmonic Mean
f-CLSWGAN [8]	68.9	52.1	59.4
CVAE [9]	-	-	51.2
SE [10]	68.1	**58.3**	62.8
ReViSE [1]	39.7	46.4	42.8
CADA-VAE [7]	75.0	55.8	**63.9**
TRZSL (Ours)	**84.3**	50.2	62.9

Table 1: Performance evaluation

We also observe interesting properties from our method in terms of performance against the amount of data used, hidden layer size and the embedded layer size. From *Figure 1* we can see that in general, as data size increases we are seeing improved performance on seen data classes. This also is true for one layer embedder and higher hidden dimension sizes. This is a phenomenon that we will observe during normal classification: as the number of training data

increases, our model can learn a more generalized feature and achieve better performance in the test phase.

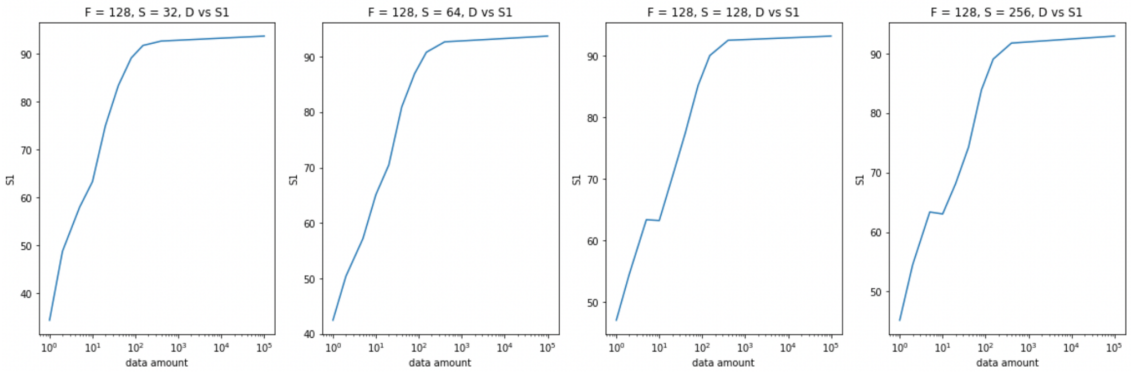

Figure1: Seen precision@1 vs Data used per class

However, we see interesting patterns on unseen data wherein as data size increases performance increases in the beginning and drops later as shown in figure 2. This drop of performance is defined in our paper as interclass overfitting. The embedders are still able to perform well on seen test dataset like figure 1 illustrated, yet fail on unseen test dataset. Our explanation is that the embedders are learning features that are too specific among the seen classes. We define this phenomenon as inter-class overfitting. We also find that if the hidden size is increasing, the performance drop is just getting postponed as we can see in comparison of the following plots.

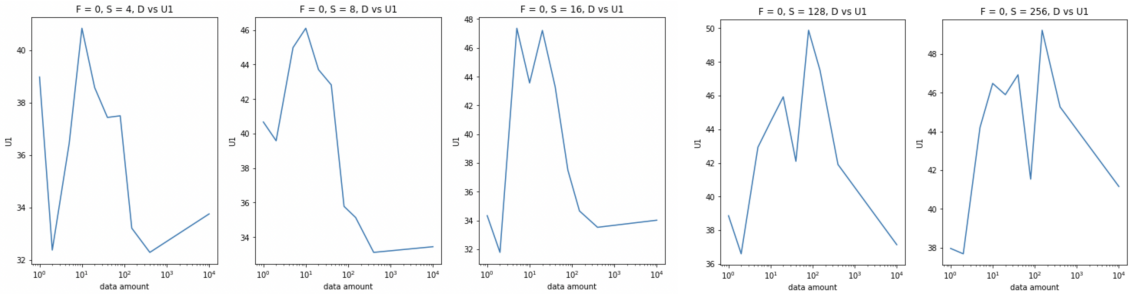

Figure 2: Unseen precision@1 vs Data used per class

Figure 3 shows how the performance changes when the hidden layer size changes. From top left to bottom right the data used per class (D) is increased. We can see from the first row D = [1,2,5,10], the performance drops when we increase the hidden size. This is because when the hidden size is increased, the model has greater capacity, and with few training data our model will suffer from overfitting on specific few training examples. In contrast to what we discuss for figure 2, we define this as intra class overfitting. When the amount of data is increased D = [20,40,80], we notice we have the opposite. In these cases, our model will increase its generalization ability with larger embedding sizes. However, when we further increase D, the performance drops again, which conforms to the inter-class overfitting phenomenon we have discussed earlier.

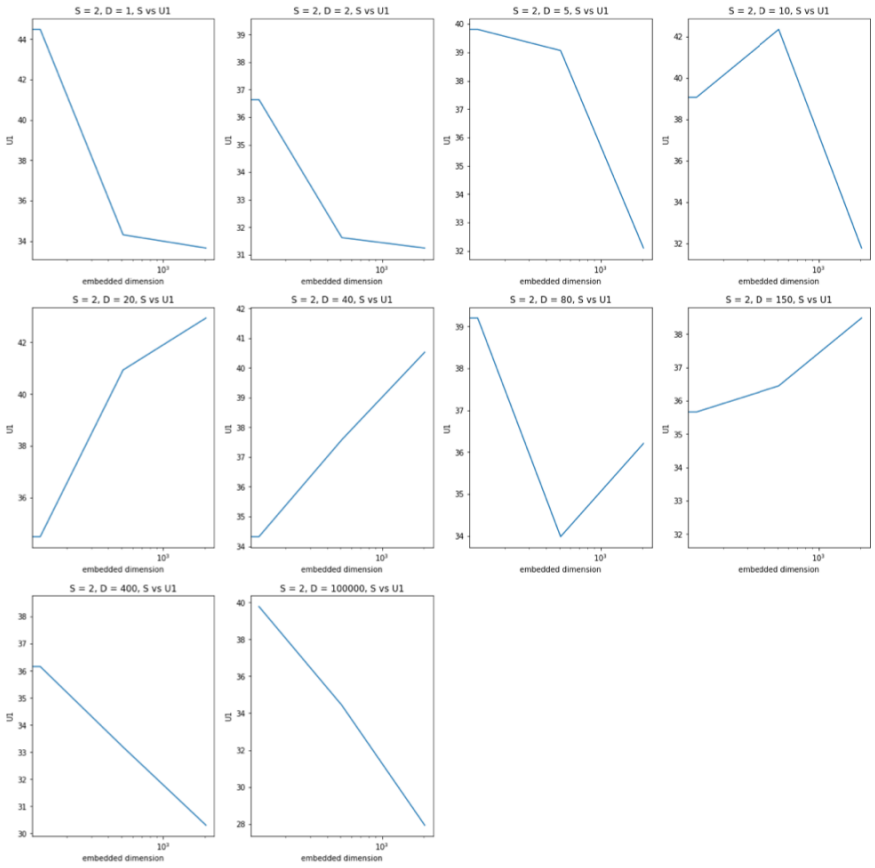

Figure 3: Unseen precision@1 Vs Hidden layer sizes

7. Discussion

Our proposed method reaches considerable results compared to the current state-of-the-art [7]. We are less than 2% behind yet using only 25% of the entire dataset, with merely a 2-layer dense network. Our proposed solution is highly data efficient, quickly converging and requires little computational power compared to GAN-based methods. According to what we have discussed in section 6, we believe that, with more effort on model and layer selection, we deal with the inter-class and intra-class overfitting issues more gently. It is promising that TRIZSL can reach higher accuracy in terms of the same experiment setting.

8. References

[1] Y.-H. H. Tsai, L.-K. Huang, and R. Salakhutdinov. Learning robust visual-semantic embeddings. In ICCV, pages 3591– 3600, 2017. 1, 2, 7

[2] G. R. Koch, “Siamese Neural Networks for One-Shot Image Recognition”, 2015.

[3] E. Hoffer, and N. Ailon, “Deep Metric Learning Using Triplet Network” Similarity-Based Pattern Recognition , page 84--92. Cham, Springer International Publishing, 2015.

[4] Y. Xian, C. H. Lampert, B. Schiele, Z. Akata. "Zero-Shot Learning - A Comprehensive Evaluation of the Good, the Bad and the Ugly", IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI) 40(8), 2018.

[5] Narayan, Sanath, et al. "Latent embedding feedback and discriminative features for zero-shot classification." arXiv preprint arXiv:2003.07833 (2020).

[6] Xian, Yongqin, et al. "f-vaegan-d2: A feature generating framework for any-shot learning." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

[7] Schönfeld, Edgar, et al. "Generalized zero-shot learning via aligned variational autoencoders." red 2 (2019): D2.

[8] Y . Xian, T. Lorenz, B. Schiele, and Z. Akata. Feature generating networks for zero-shot learning. In CVPR, 2018. 1, 2, 3, 6, 7, 8

[9] A. Mishra, S. Krishna Reddy, A. Mittal, and H. A. Murthy. A generative model for zero shot learning using conditional variational autoencoders. In CVPR, pages 2188–2196, 2018. 1, 2, 3, 6,

[10] V. Kumar Verma, G. Arora, A. Mishra, and P. Rai. Generalized zero-shot learning via synthesized examples. In CVPR, pages 4281–4289, 2018. 1, 2, 3, 6, 7