Generalized Zero-Shot Learning via Synthesized Examples

Vinay Kumar Verma[1,*], Gundeep Arora[1,*], Ashish Mishra[†] and Piyush Rai[*]
[*]Indian Institute of Technology Kanpur [†]Indian Institute of Technology Madras
{vkverma,gundeep,piyush}@cse.iitk.ac.in, mishra@cse.iitm.ac.in

Abstract

We present a generative framework for generalized zero-shot learning where the training and test classes are not necessarily disjoint. Built upon a variational autoencoder based architecture, consisting of a probabilistic encoder and a probabilistic conditional *decoder, our model can generate novel exemplars from seen/unseen classes, given their respective class attributes. These exemplars can subsequently be used to train any off-the-shelf classification model. One of the key aspects of our encoder-decoder architecture is a feedback-driven mechanism in which a discriminator (a multivariate regressor) learns to map the generated exemplars to the corresponding class attribute vectors, leading to an improved generator. Our model's ability to generate and leverage examples from unseen classes to train the classification model naturally helps to mitigate the bias towards predicting seen classes in generalized zero-shot learning settings. Through a comprehensive set of experiments, we show that our model outperforms several state-of-the-art methods, on several benchmark datasets, for both standard as well as generalized zero-shot learning.*

1. Introduction

The ability to correctly categorize objects from previously unseen classes is a key requirement in any truly autonomous object discovery system. Zero-shot Learning (ZSL) is a learning paradigm [18, 1, 34] that tries to fulfil this desideratum by leveraging *auxiliary* information that may be available for each seen/unseen class. ZSL models usually assume that this information is given in form of class attribute vectors or textual descriptions of classes.

Typical approaches taken by existing ZSL models can be roughly categorized into the following: (1) Learning a mapping from the instance space to the class-attribute space and predicting the class of an unseen class test instance by finding its closest class-attribute vector [18, 1]; (2) Defining the classifier for each unseen class as a weighted combination of the classifiers for the seen classes, where the combination weights are typically defined using similarity scores of unseen and seen class [22, 4], and (3) Learning a probability distribution for each seen class and extrapolating to unseen class distributions using the class-attribute information [30, 10, 31]. A more detailed discussion of the related work on ZSL is provided in the Related Work section. Although the aforementioned ZSL models have shown considerable promise on various benchmark datasets, a key limitation of most of these models is that, at test time, these are highly biased towards predicting the seen classes [5]. This is because the ZSL model is learned using labeled data only from the seen classes. Due to this issue, the ZSL models are usually evaluated in a restricted setting where the training and test classes are assumed to be disjoint, i.e., the test examples only come from the unseen classes and the search space is limited to the unseen classes only. The more challenging setting where the training and test classes are not disjoint is known as generalized zero-shot learning (GZSL), and is considered a more formidable problem setting. Recent work [5, 34] has shown that the accuracies of most ZSL approaches drops significantly in this setting.

In this work, we take a generative approach to the ZSL problem, which naturally helps address the generalized ZSL problem. Our approach is based on a generative model to *synthesize* exemplars from the unseen classes (and, optionally, also from the seen classes), and subsequently training an off-the-shelf classification model using these synthesized exemplars. Our approach is motivated by, and is similar in spirit to, recent work on synthesizing exemplars for ZSL [3, 10, 19, 20], which has shown to lead to improved performance, especially in the GZSL setting.

For exemplar generation, we develop a generative model based on a *conditional* variational autoencoder (VAE) architecture, in which the latent code of any instance is augmented with the class-attribute vector. This architecture is further coupled with a *discriminator* (a multivariate regressor) that learns a mapping from the VAE generator's output to the class-attribute. This feedback helps to improve the generator by encouraging it to generate exemplars that are of highly discriminative nature. Moreover, the discriminator also allows us to operate in semi-supervised settings by incorporating unlabeled examples for which we do

[1] Equal contributions from both authors.

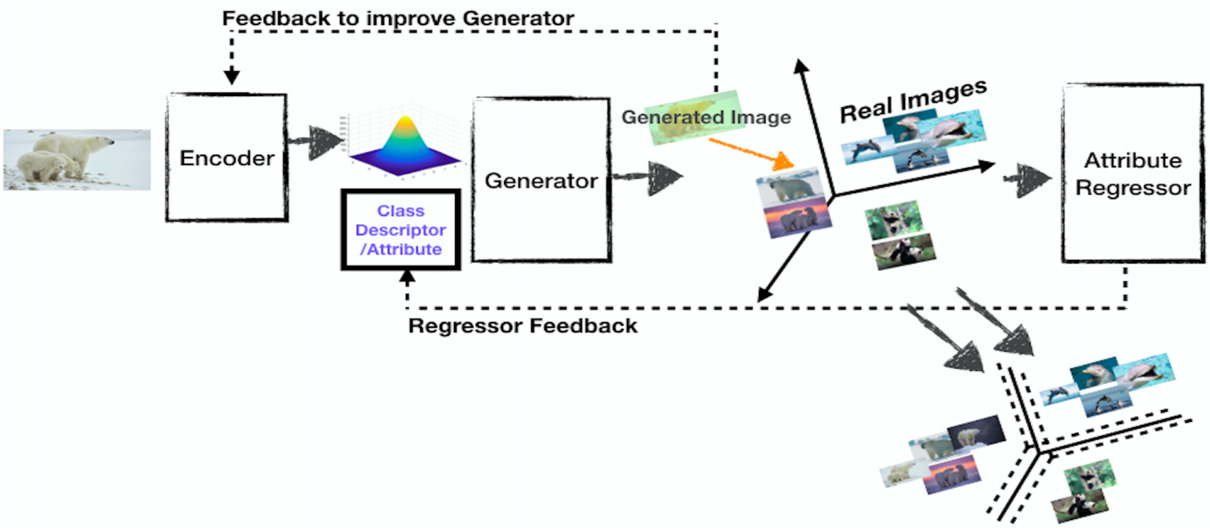

Figure 1. An illustration of our model: Like a conditional VAE, the probabilistic encoder-generator reconstructs a noisy version of the original input(Generated Image). We add a feedback mechanism via loss from attribute regressor as well as the encoder, to improve the reconstruction capability of the decoder network. The dotted lines denote the feedback loss, while the orange line points to the desired location of the reconstruction in feature space. At test time, predictions are made using an off-the-shelf classifier (e.g. SVM) trained on synthesized examples. The 3D coordinate is representative of feature space and right bottom parts represents max-margin classifier

not know the class label (and thus no class-attributes are available for these examples). Once the model has been learned, it can be used to generate exemplars for any unseen class (and, optionally, also seen classes), given its class-attributes, and the exemplars can be used as labeled training examples to train any off-the-shelf classification model. Notably, since the classification model is trained using labeled examples from seen as well as unseen classes, at test time, it is not biased towards predicting seen classes in the GZSL setting, as also evidenced by recent work on leveraging synthesized unseen class examples [3, 10, 19, 20]. The model in [3] is a vanilla conditional generative model, while ours is based on an explicit feedback driven mechanism. This leads to a very different model architecture and inference, and yields much better prediction accuracies. [10] represented each unseen class by Gaussian distribution while [19, 20] based on semantic-visual embedding with Diffusion Regularization.

Also note that, in our proposed framework, the final classification stage directly predicts the actual *class label* for each test example, as opposed to predicting the *class-attribute vector* [26, 1], which further necessitates a nearest neighbor search to find the class label. This is appealing because the nearest neighbor search approach, as is commonly used in most ZSL methods, is known to suffers from issues, such as the hubness problem [6].

2. Background and Notation

In the ZSL problem, we assume that there are S seen classes for which we have labeled training examples and U unseen classes for which we do not have any labeled training examples. The test examples can either be exclusively from unseen classes (the traditional ZSL setting) or from both seen and unseen classes (the generalized ZSL setting [5, 34]). In this work, our focus will be on the GZSL setting, although our model applies to both settings (we provide comprehensive experimental evaluations for both).

Although we do not have access to any labeled examples from the unseen classes, for each (seen/unseen) class $c = 1, \ldots, S + U$, we assume that we are given the respective class-attribute vectors $\{\mathbf{a}_c\}_{c=1}^{S+U}$, where the attribute vector of class c is $\mathbf{a}_c \in \mathbb{R}^L$. ZSL models are usually based on leveraging the class-attribute information to transfer information from seen classes to the unseen classes.

Note that each seen class labeled training example $\{\boldsymbol{x}_n, y_n\}$ can be equivalently denoted as $\{\boldsymbol{x}_n, \boldsymbol{a}_{y_n}\}$, the feature vector and class-attribute vector pair for this example. Therefore, assuming a total of N_S examples from the seen classes, the training data from seen classes can be collectively denoted by $\mathcal{D}_S = \{\boldsymbol{x}_n, \boldsymbol{a}_{y_n}\}_{n=1}^{N_S}$. The goal in ZSL is to learn a classification model using $\mathcal{D}_S$ and then use the learned model to predict the labels for test examples.

3. The Basic Model

Our model, shown via the pictorial illustration in Fig. 1 is based on a variational autoencoder(VAE) architecture [14]. The VAE consists of a probabilistic encoder model with parameters θ_E and a probabilistic decoder (a.k.a. generator) model with parameters θ_G. In our model, the generator is also conditioned on the class-attribute vector, which enables us to synthesize exemplars from any class c by simply specifying the corresponding class-attribute vector $\boldsymbol{a}_c$, along with an unstructured code $\boldsymbol{z}$.

Note that this architecture is akin to a conditional VAE model [27], except for some key differences

- We assume that the latent code and the class-attribute are *disentangled* (latent code $\boldsymbol{z}_n$ represents the unstructured part of $\boldsymbol{x}_n$ and the class-attribute vector $\boldsymbol{a}_{y_n}$ represents the class-specific discriminative information); this helps in generating exemplars that are high discriminative in nature, as guided by the class-attribute vector.

- The model also consists of a mapping from the decoder's output to the class-attribute vector; this mapping is learned via a discriminator which is a multivariate regression model with parameters θ_R that maps the decoder's output $\hat{\boldsymbol{x}}_n$ to the respective class-attribute vector $\boldsymbol{a}_{y_n}$ via a feedforward network (learned jointly with the rest of the model). The regressor plays two key roles in our model: (1) It provides feedback to the generator (more on this in Sec. 4.1), which results in generation of exemplars that can be discriminated easily; and (2) It allows using unlabeled examples during training by computing the probability distribution $p(\boldsymbol{a}|\boldsymbol{x})$ on their class-attribute vector.

Note that, in our proposed model, each example $\boldsymbol{x}_n$ is influenced by two sources - the latent code $\boldsymbol{z}_n$ which represents the unstructured (class-independent) component, and the class-attribute vector $\boldsymbol{a}_{y_n}$ which represents the structured (class-specific) component. We describe each of the model components in more detail in Section 4.

3.1. Training the Final Classifier

Once the generative model has been learned, we can generate *labeled* exemplars of any class by first generating the unstructured component $\boldsymbol{z}$ randomly from the prior $p(\boldsymbol{z})$, specifying the class c (via the class attribute vector $\boldsymbol{a}_c$) of the exemplar to be generated, and then generating the example $\boldsymbol{x}$ using the generator. We generate a fixed number of exemplars from each class and these generated exemplars are finally used to train a discriminative classifier, such as a support vector machine (SVM) or a softmax classifier. Since this stage utilizes labeled examples from both seen and unseen classes, our approach is inherently robust against the bias towards seen classes, as also evidenced by other recent work [3, 10]. Also note that, for training this classifier, we can either use the original labeled examples from the seen classes or can also augmented those examples using additional exemplars from the seen classes as well.

4. The Complete Model Architecture

We now describe our model architecture in more detail. Our model, as shown through the block-diagram in Fig. 2, is based on a variational autoencoder (VAE), consisting of an encoder $p(\boldsymbol{z}|\boldsymbol{x})$ (a recognition network) and a decoder/generator $p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{a})$. Note that, unlike the standard VAE, the generator's input consists of both the latent code $\boldsymbol{z}$ as well as the class-attribute vector $\boldsymbol{a}$, similar to a conditional VAE (CVAE) architecture [27], enabling the generated exemplars from different classes to be distinguishable from each other. However, as compared to the traditional CVAE, our architecture has a few key differences, motivated by the goal is having a generator that can generate exemplars from any given class that can act as a surrogate to the real examples that class:

- It consists of a discriminator (a multi-output regressor, which we call regressor net) that learns to map a real example $\boldsymbol{x}$ (from seen classes) or a generator-synthesized example $\hat{\boldsymbol{x}}$ (from unseen/seen classes) to the corresponding class-attribute vector $\boldsymbol{a}$. Backpropagating the regressor's loss helps to improve the generator by ensuring that the generated exemplars are representative of the associated class.

- The regressor also enables using our model in a semi-supervised setting, where some training examples do not have labels/class-attribute vectors. For such examples, the class-attribute vector is replaced by the output distribution $p(\boldsymbol{a}|\boldsymbol{x})$ of the regressor (Sec. 4.1).

- A link from the generator back to the recognition model (encoder) to ensure that the generator's output $\hat{x}$ is as good as the actual input $\boldsymbol{x}$, i.e., the distribution $p(\boldsymbol{z}|\hat{x})$ is closed to the distribution $p(\boldsymbol{z}|\boldsymbol{x})$.

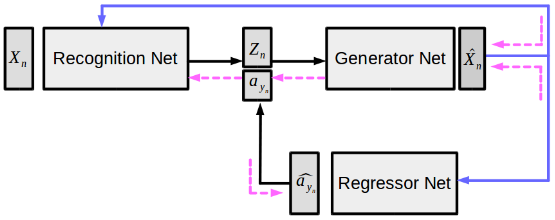

Figure 2. The proposed architecture for zero-shot set-up. Each block represents a feed-forward neural network. The encoder to $\mathbf{z}_n$ link is stochastic similar to a VAE. The blue lines direct feedback connection into regressor and recognition network for the generated $\hat{X}_n$. The red-lines represent the back propagation direction.

Our model architecture draws its inspiration from recent work on controllable text generation [11], where the goal is to generate text having a certain desired characteristics, such as positive/negative sentiment, by specifying a binary attribute. In contrast, in our ZSL setting, the VAE model is conditioned on the class-attribute vector, enabling us to smoothly transition the generation from seen to unseen classes by varying the class-attribute vector. Moreover, while the focus of the work in [11] was on text generation, our goal here is to leverage such a framework to solve the generalized ZSL problem.

Next, we briefly describe the key components of our model architecture shown in Fig. 2.

4.1. The Discriminator/Regressor

Our discriminator/regressor, defined by a probabilistic model $p_R(\boldsymbol{a}|\boldsymbol{x})$ with parameters θ_R, is a feedforward neural network that learns to map the example $\boldsymbol{x} \in \mathbb{R}^D$ to its corresponding class-attribute vector $\boldsymbol{a} \in \mathbb{R}^L$. The regressor is learned using two sources of data:

- Labeled examples $\{\boldsymbol{x}_n, \boldsymbol{a}_{y_n}\}_{n=1}^{N_S}$ from the seen class, on which we can define a supervised loss, given by

  $$\mathcal{L}_{Sup}(\theta_R) = -\mathbb{E}_{\{\boldsymbol{x}_n, \boldsymbol{a}_{y_n}\}}[p_R(\boldsymbol{a}_{y_n}|\boldsymbol{x}_n)] \tag{1}$$

- Synthesized examples $\hat{\boldsymbol{x}}$ from the generator, for which we can define an unsupervised loss, given by

  $$\mathcal{L}_{Unsup}(\theta_R) = -\mathbb{E}_{p_{\theta_G}(\hat{\mathbf{x}}|\mathbf{z},\mathbf{a})p(\mathbf{z})p(\mathbf{a})}[p_R(\mathbf{a}|\hat{\mathbf{x}})] \tag{2}$$

  Note that the unsupervised loss is computed by taking a latent code $\boldsymbol{z}$ sampled from the prior $p(\boldsymbol{z})$, along with a class-attribute vector $\boldsymbol{a}$ sampled from the empirical distribution $p(\boldsymbol{a})$, generating an exemplar from the generator distribution $p_{\theta_G}(\hat{\mathbf{x}}|\mathbf{z}, \mathbf{a})$, and then taking an expectation w.r.t. these distributions. During this phase, we can use attributes both from seen as well as unseen classes.

  The overall training objective of the regressor is then defined as the following weighted combination of the supervised and the unsupervised training objectives

  $$\min_{\theta_R} \mathcal{L}_R = \mathcal{L}_{Sup} + \lambda_R \cdot \mathcal{L}_{Unsup} \tag{3}$$

  The above optimization problem is the first step of the alternating optimization procedure. It optimizes the regressor parameters θ_R to make the regressor predict the correct class-attribute vector even with a noisy signal $\hat{\mathbf{x}}$. Note that, in this step, we assume that the generator distribution is fixed.

4.2. The Encoder and Conditional Generator

The conditional VAE in our architecture (Fig. 2) consists of the conditional generator $p_G(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{a})$ with parameters θ_G, which are responsible for generating the exemplars that will subsequently be used to train the final classification model. Hence the training needs to be designed such that the generated class conditional distribution nicely approximates the true distribution. Denoting the VAE encoder as $p_E(\boldsymbol{z}|\boldsymbol{x})$ with parameters θ_E, and the regressor output distribution as $p_R(\boldsymbol{a}|\boldsymbol{x})$, the VAE loss function is given by (assuming the regressor to be fixed)

$$\begin{aligned}\mathcal{L}_{VAE}(\theta_E, \theta_G) = -&\mathbb{E}_{p_E(\mathbf{z}|\mathbf{x}),p(\mathbf{a}|\mathbf{x})}[\log p_G(\mathbf{x}|\mathbf{z}, \mathbf{a})] \\ &+ \mathrm{KL}(p_E(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\end{aligned} \tag{4}$$

where the first term on the R.H.S. is the generator's reconstruction error and the second term promotes the VAE posterior (the encoder) to be close to the prior.

We model the VAE encoder $p_E(\boldsymbol{z}|\boldsymbol{x})$, VAE conditional decoder/generator $p_G(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{a})$, and the regressor $p_R(\boldsymbol{a}|\boldsymbol{x})$ as Gaussian distributions. Also note that the factorization of the joint distribution over overall latent code $(\boldsymbol{z}, \boldsymbol{a})$ into two components $p_E(\mathbf{z}|\mathbf{x})$ and $p_R(\mathbf{a}|\mathbf{x})$ is consistent with our attempt at learning a disentangled representation [11].

Discriminator-Driven Learning: As described in Sec. 4.1, we use the discriminator/regressor to improve the generator by backpropagating its error that encourages generation of exemplars, $\hat{\mathbf{x}}$ coherent with the class-attribute $\mathbf{a}$. We perform this by using a couple of loss functions. The first one simply assumes that the regressor, has optimal parameters and any reason for it not regressing to the correct value is because of the poor generation by the generator:

$$\mathcal{L}_c(\theta_G) = -\mathbb{E}_{p_G(\hat{\mathbf{x}}|\mathbf{z},\mathbf{a})p(\mathbf{z})p(\mathbf{a})}[\log p_R(\mathbf{a}|\hat{\boldsymbol{x}})] \tag{5}$$

This loss encourages the generator to create samples such that the regressed attribute vector by the discriminator is correct. We also add an additional term that acts as a regularizer that encourages the generator to generate a good class-specific sample even from a random $\mathbf{z}$ drawn from the prior $p(\boldsymbol{z})$ and combined with class-attribute from $p(\boldsymbol{a})$. This is akin to doing semi-supervised learning.

$$\mathcal{L}_{Reg}(\theta_G) = -\mathbb{E}_{p(\mathbf{z})p(\mathbf{a})}[\log p_G(\hat{\mathbf{x}}|\mathbf{z}, \mathbf{a})] \tag{6}$$

The above ensures that the quality of the synthesized exemplars, to be used to train the final classifier, is at par with that of the true data. While the above two loss functions help us increase the coherence of $\hat{\mathbf{x}} \sim p_G(\hat{\mathbf{x}}|\mathbf{z}, \mathbf{a})$ with the class-attribute $\mathbf{a}$, we also need to enforce the independence (disentanglement) [11] of the unstructured component $\mathbf{z}$ from the class-attribute $\mathbf{a}$. To this end, we use the encoder to ensure that the sampling distribution and the one obtained from the generated exemplar follow the same distribution. More formally, we add a loss component,

$$\mathcal{L}_E(\theta_G) = -\mathbb{E}_{\hat{\mathbf{x}}\sim p_G(\hat{\mathbf{x}}|\mathbf{z},\mathbf{a})}\mathrm{KL}[(p_E(\mathbf{z}|\hat{\mathbf{x}})||q(\mathbf{z}))] \tag{7}$$

The distribution $q(\mathbf{z})$ could be the prior $p(\mathbf{z})$ or the posterior from a labeled sample $p(\mathbf{z}|\mathbf{x}_n)$, in which case the attribute component $\mathbf{a}_{y_n}$ is used. Hence the complete learning objective for the generator and encoder is given by,

$$\min_{\theta_G, \theta_E} \mathcal{L}_{VAE} + \lambda_c \cdot \mathcal{L}_c + \lambda_{reg} \cdot \mathcal{L}_{Reg} + \lambda_E \cdot \mathcal{L}_E \tag{8}$$

The overall learning objective for the conditional autoencoder is a weighted combination of all the components discussed above, effectively optimizing parameters to ensure good density estimation. Vanilla conditional generative models like conditional VAE [21] and [3] crucially lack such a mechanism. The weight are hyperparameters, tuned for optimal reconstruction. This completes the second step of the alternating optimization.

5. Related Work

Zero-shot learning (ZSL) has received a significant amount of interest recently. Due to lack of space, it will not be possible to comprehensively cover all the work in this area. In this section, we provide an overview of some of the representative methods, both for traditional ZSL as well as generalized ZSL.

Some of the early works on ZSL were based on learning a direct/indirect mapping [18] from the instances to the class-attributes. This mapping is then applied on the test data to first predict the class-attribute vector, and used to predict the class by finding the most similar attribute vector.

Another popular approach for ZSL is based on learning a shared embedding of seen and unseen class instances into the class-attribute space [26, 22]. After projection, nearest neighbor methods can be used to find the most similar class attribute vector for the (projected) test instance, which corresponds to the most likely class. While conceptually simple and easy to implement, these methods suffer from shortcomings such as the hubness problem [23].

In a similar vein of embedding instances to an attribute/semantic space, some other approaches learn a mapping of instances to a semantic (attribute) space, with [1] learning a bilinear compatibility function between the instances and attribute space using ranking loss and [7] optimizing the structural SVM loss to learn the bilinear compatibility. Embedding based methods for ZSL have also been extended to learn non-linear multi-modal embeddings.

Using the fact that the class-attributes can be used to compute relationships between seen and unseen classes (e.g., using a class-attribute based similarity measures), a number of ZSL methods have been proposed that are based on representing the parameters representing each unseen class as a similarity-weighted combination of the parameters representing the seen classes [22, 24, 4].

The generalized ZSL problem [5, 34] where the training and test classes are not disjoint is considerably more challenging as compared to the traditional ZSL, and a recent focus has been to design ZSL methods that can work robustly in this setting without being biased towards predicting seen classes. Generative models [30, 10, 3, 31] are promising in this setting. One of the ways these models can solve the GZSL problem is by generating synthetic labeled examples from the unseen classes and then using these examples (and the labeled examples from other seen classes) to train a classification model.

Following this approach, and in a similar spirit to our work, a number of recent works [10, 3] have tried to use synthesized examples both for the seen as well as unseen classes to perform the generalized zero-shot task. [10] synthesize samples for each class by approximating the class conditional distribution of the unseen classes based on the learned class conditional distributions of the seen classes and their corresponding attribute vectors. On the other hand [3] perform adversarial training to train generators and use the domain adapted samples for perform classification.

Finally, the ability to generate exemplars from unseen class and use them in training classification models can also help mitigate the domain-shift problem [15] encountered by traditional ZSL methods if the distribution of seen classes and unseen classes are not the same. Given labeled examples from the seen classes and the synthetic labeled examples from the unseen classes, supervised/semi-supervised domain adaptation methods can be readily applied to address the domain shift problem.

Despite the significant amount of progress in ZSL over the past few years, we would also like to point out that differences in evaluation protocols for evaluating ZSL models often make it hard to have a fair comparison between the various methods. In a recent work, [34] lay down a set of guidelines on choosing data-splits and evaluations protocols to ensure fair comparisons. Our experimental settings strictly adhere to these guidelines as much as possible.

6. Experiments

To test the effectiveness of our model (referred to as **SE-GZSL** for Synthesized Examples for Generalized Zero-Shot Learning), we conduct an extensive evaluation on several benchmark datasets and compare it with various state-of-the-art ZSL models. Note that the baselines also include some recently proposed methods based on exemplar generation [3, 21, 10]. We report our results on the following benchmark datasets, while also following the guidelines offered by [34] for evaluating ZSL models:

- **Animals with Attributes**: The AwA [17] dataset contains 30,475 images with a standard split of 40 seen classes (training set) and 10 unseen classes (test set). Each class has a human-provided 85-dimensional class-attribute vector. Since raw images for the original dataset were not available, we used the VGG19 features. Recently, an updated version of the dataset with raw images has been made available. For completeness, we evaluate our model on both the datasets, henceforth referred to as AwA1 [17] and AwA2 [34].

- **SUN Scene Recognition**: The SUN [35] dataset comprises of 717 scenes. For the ZSL setting, we use the widely used split of 645 seen classes with 72 unseen classes. This dataset has 14,340 fine-grained images, with attributes available at the image level. We combine the attributes of all images in a class to obtain class-level attributes and use them for our training.

- **Caltech UCSD Birds 200** : The CUB dataset [32] consists of 200 classes with 11,788 fine-grained images of birds. We use the given split of 150 unseen and 50 seen classes. Like the SUN dataset, this one too has attributes available at image level and we average them across each class to get class-attributes.
- **Imagenet**: We also evaluate the zero-shot classification accuracy on the large-scale Imagenet dataset. The setup here involves training using the images from the 1000 class ILSVRC 2012 [25] data and testing on the non-overlapping 360 classes from the ILSVRC 2010 data. Unlike other datasets, we use the GoogLeNet [28] features for this dataset.

Datasets and their statistics are summarized in Table 1. While human-curated attributes for each class are available for most datasets, we use word2vec representations of each class as the class-attribute vector for Imagenet.

Dataset	Attribute/Dim	#Image	Seen/Unseen Class
AWA1	A/85	30475	40/10
AWA2	A/85	37322	40/10
CUB	A/312	11788	150/50
SUN	A/102	14340	645/72
Imagenet	W/1000	254000	1000/360

Table 1. Datasets used in our experiments, and their statistics

6.1. Parameter Settings and Evaluation

For each of the mentioned datasets, we evaluate our method on the commonly used standard split as well as on the split proposed in recent work [34] about evaluation protocols for ZSL. A new version of the AwA dataset was also proposed in [34]. We therefore report our results on both the old AwA1 dataset, as well as the new AwA2 dataset. For each of the datasets we use the ResNet features of the images. No extra fine-tuning was done to improve the image features. The network was optimized based on the loss function discussed in Sec. 4, using the Adam [13] optimizer. The learning started with pre-training the VAE using the loss from Eq. 4. This is followed by joint alternating-training of regressor and encoder-generator pair by optimizing the loss from Eq. 3 and Eq. 8, until convergence. The hyerparameters were chosen based on a train-validation split and were used while training the model on complete data. Though there may be some small variability in the results based on the values for the hyperparameters, we used $\lambda_R = 0.1$, $\lambda_c = 0.1$, $\lambda_{reg} = 0.1$ and $\lambda_E = 0.1$, which worked well for most of the experiments. It is important to note that the same architecture is used across all datasets and no extra data/feature engineering is used/performed to improve accuracies, thus showcasing the efficacy of the training schedule for this task. The encoder is realised using a two-hidden layer feedforward network while the decoder and the regressor are modeled as feedforward networks consisting of one hidden layer with 512 hidden units each.

For the evaluation criteria, we use the average per-class accuracy. This metric reduces the bias of classes having higher examples in the test set, and provides a better measure of the model performance [34].

We shall now discuss the experimental setup for the two settings we experiment with: ZSL and GZSL. Let the datasets be available in two parts, the labeled examples from the seen classes, $\mathcal{X}^S$ and the unlabeled examples from the unseen classes, $\mathcal{X}^U$.

6.2. Generalized Zero Shot Learning

The GZSL setting involves performing classification when the test set has examples from both the seen and the unseen classes, with no prior distinction between them. For this, we perform an 80-20 random split of the dataset to obtain, $\mathcal{X}^S_{train}$ and $\mathcal{X}^S_{test}$. The split is done ensuring that there are some examples from each of the S classes. We train our model on X^S_{train}. Once trained, samples are synthesized for all the $S + U$ classes using our generative model. These samples finally used to train a multi-class linear SVM. The examples from $\mathcal{X}^U$ (referred as $\mathcal{Y}^{tr}$) and $\mathcal{X}^S_{test}$ (referred as $\mathcal{Y}^{ts}$) are then used to calculate the average per-class accuracy. For the GZSL setting, the evaluation measures are denoted as

- $Acc_{\mathcal{Y}^{tr}} : S \rightarrow S+U :$ Average per-class classification accuracy on $\mathcal{X}^S_{test}$ using a classifier trained for $S + U$
- $Acc_{\mathcal{Y}^{ts}} : U \rightarrow S+U :$ Average per-class classification accuracy on $\mathcal{X}^U_{test}$ using a classifier trained for $S + U$

To mitigate the bias towards seen classes accuracy, we evaluate the harmonic mean of the above defined average per-class top-1 accuracies as $\mathbf{H} = (2 \cdot Acc_{\mathcal{Y}^{tr}} \cdot Acc_{\mathcal{Y}^{ts}})/(Acc_{\mathcal{Y}^{tr}} + Acc_{\mathcal{Y}^{ts}})$

The results for the test accuracy on the unseen classes and the aggregate measure (harmonic mean), for this setup are compiled in Tables 2. The number of samples synthesized is a hyper-parameter and can be chosen to balance evaluation time and accuracy. The results clearly demonstrate that our model can significantly mitigate the GZSL issue of the bias towards seen classes, which a number of previous ZSL approaches tend to suffer [5, 34]. Our approach outperforms previous approaches on both the unseen class test accuracy, $Acc_{\mathcal{Y}^{ts}}$ as well as the harmonic mean measure

	SUN			CUB			AWA1			AWA2		
Method	**U → S+U**	**S → S+U**	**H**	**U → S+U**	**S → S+U**	**H**	**U → S+U**	**S → S+U**	**H**	**U → S+U**	**S → S+U**	**H**
CONSE [22]	6.8	39.9	11.6	1.6	**72.2**	3.1	0.4	**88.6**	0.8	0.5	**90.6**	1.0
CMT* [26]	8.1	21.8	11.8	7.2	49.8	12.6	0.9	87.6	1.8	0.5	90.0	1.0
SSE [36]	2.1	36.4	4.0	8.5	46.9	14.4	7.0	80.5	12.9	8.1	82.5	14.8
SJE [2]	14.7	30.5	19.8	23.5	59.2	33.6	11.3	74.6	19.6	8.0	73.9	14.4
ESZSL [24]	11.0	27.9	15.8	12.6	63.8	21.0	6.6	75.6	12.1	5.9	77.8	11.0
SYNC[4]	7.9	**43.3**	13.4	11.5	70.9	19.8	8.9	87.3	16.2	10.0	90.5	18.0
SAE [16]	8.8	18.0	11.8	7.8	54.0	13.6	1.8	77.1	3.5	1.1	82.2	2.2
LATEM [33]	14.7	28.8	19.5	15.2	57.3	24.0	7.3	71.7	13.3	11.5	77.3	20.0
ALE [1]	21.8	33.1	26.3	23.7	62.8	34.4	16.8	76.1	27.5	14.0	81.8	23.9
DEVISE [7]	16.9	27.4	20.9	23.8	53.0	32.8	13.4	68.7	22.4	17.1	74.7	27.8
CVAE-ZSL[21]	–	–	26.7	–	–	34.5	–	–	47.2	–	–	51.2
SE-GZSL (Ours)	**40.9**	30.5	**34.9**	**41.5**	53.3	**46.7**	**56.3**	67.8	**61.5**	**58.3**	68.1	**62.8**

Table 2. Accuracy for GZSL, on proposed split(PS). U and S represents top-1 accuracy on unseen and seen class. H: Harmonic mean.

H, which quantities the aggregate performance across both seen and unseen test classes. Here for the SVM training we used the different weight for the seen and unseen class. Using the validation data we found that linear SVM with the C=1 outperforms w.r.t. other hyper-parameter. Here seen weight 1.0 while unseen weight is $0.2, 0.2, 0.05$ and 0.05 is used for the SUN, CUB, AWA1 and AWA2 respectively.

6.3. Conventional Zero Shot Learning

For the conventional ZSL setting, we first train our generative model on $\mathcal{X}^S$. We then synthesize samples for the unseen classes and finally train a multi-class linear SVM using the generated training data from these unseen classes. The SVM is used to predict the classes for the test examples $\mathcal{X}^U$. The average per-class accuracy, $Acc_{\mathcal{Y}^{ts}}$ is reported in Table 3 and Table 4. The improvements are consistent across scale and the complexity of images in the dataset. This is evident from the improvement on the large-scale Imagenet dataset as well complex fine-grained datasets like CUB. The large number of classes and relatively fewer training examples per class in the SUN dataset do not hamper the performance of our method.

To further probe the efficacy of the feedback mechanism, we perform an ablation study, where the model is trained without feedback mechanism for the conventional ZSL setting. The results in Table. 3 cleary demonstrate the benefits of the feedback mechanism and the carefully designed loss function. This also explicates why our model outperforms other recently proposed architectures, such as [3, 21] which dont have a feedback-driven mechanism in their generative models. The degrade in performance without feedback is particularly significant in fine-grained datasets like CUB.

6.4. Quality of Synthesized Samples

Our quantitative results reported for GZSL (Sec. 6.2) and ZSL (Sec. 6.3) demonstrate that the samples generated by our model are of good quality and effective for classification tasks. To gain a further insight into the quality of the generated samples, we compare the empirical distribution of the generated samples from a few unseen classes to the empirical distribution of the real samples from the same classes.

Method	Accuracy
AMP[9]	13.1
DeViSE:[7]	12.8
ConSE [22]	15.5
SS-Voc [8]	16.8
CVAE-ZSL[21]	24.7
SE-ZSL (Ours)	**25.43**

Table 4. Per-class accuracy of ZSL for the ImageNet dataset. 1000 class ILSVRC-12 are used for training and ILSVRC-10(class that are not present in ILSVRC-12) are used for testing

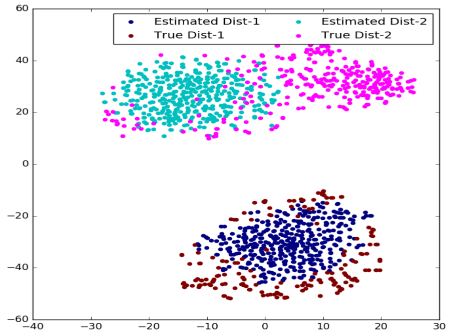

Figure 3. t-SNE plot of estimated data distribution (using generated data) and true data distribution (using real data) for two of the unseen classes

This is done by taking real and generated examples and embedding them into two dimensions using t-SNE. As shown in Fig. 3 for two of the unseen classes, the empirical distributions of generated and real samples overlap significantly, corroborating our model's ability to generates samples that look like samples from the true distribution.

Finally, we also perform an experiment to assess how varying the number of the generated examples per class affects the classification accuracy. For this, we vary

	SUN		CUB		AWA1		AWA2	
Method	**SS**	**PS**	**SS**	**PS**	**SS**	**PS**	**SS**	**PS**
CONSE [22]	44.2	38.8	36.7	34.3	63.6	45.6	67.9	44.5
SSE [36]	54.5	51.5	43.7	43.9	68.8	60.1	67.5	61
LATEM [33]	56.9	55.3	49.4	49.3	74.8	55.1	68.7	55.8
ALE [1]	59.1	58.1	53.2	54.9	78.6	59.9	80.3	62.5
DEVISE [7]	57.5	56.5	53.2	52.0	72.9	54.2	68.6	59.7
SJE [2]	57.1	53.7	55.3	53.9	76.7	65.6	69.5	61.9
ESZSL [24]	57.3	54.5	55.1	53.9	74.7	58.2	75.6	58.6
SYNC[4]	59.1	56.3	54.1	55.6	72.2	54.0	71.2	46.6
SAE [16]	42.4	40.3	33.4	33.3	80.6	53.0	80.2	54.1
SSZSL [10]		–	55.75	–	82.67	–	–	–
GVRZSC [3]		–	60.1	–	77.1	–	–	–
EF-ZSL[30]	–	63.3	–	44.7	–	57.0	–	57.4
CVAE-ZSL[21]	–	61.7	–	52.1	–	**71.4**	–	65.8
SE-ZSL (Without Feedback)	62.0	61.2	59.8	54.1	78.4	68.2	79.3	66.3
SE-ZSL (Ours)	**64.5**	**63.4**	**60.3**	**59.6**	**83.8**	69.5	**80.8**	**69.2**

Table 3. Zero Shot Learning Accuracy on the SUN, CUB, AWA1 and AWA2 dataset. Here SS stands for the Stranded Split for each dataset that has been in use previously and PS is the new proposed split by [34]. We also experimented on a 707/10 split for SUN [16] and achieved accuracy 93.5%. Also in transductive setting EF-ZSL[30] has the $65.1\%, 52.4\%, 85.2\%$ and 82.7% accuracy on the SUN, CUB, AWA1 and AwA2 datasets respectively in the PS setting.

the number of generated examples per class in the range $[2, 5, 10, 50, 100]$, and use these examples in 3 off-the-shelf classifiers: linear SVM, kernel SVM, and nearest neighbors. As shown in Fig. 4, as expected, the classification accuracies increase with an increasing number of generated examples and it asymptotes fairly quickly, indicating that usually a small number of generated examples are sufficient to learn a fairly accurate classifier.

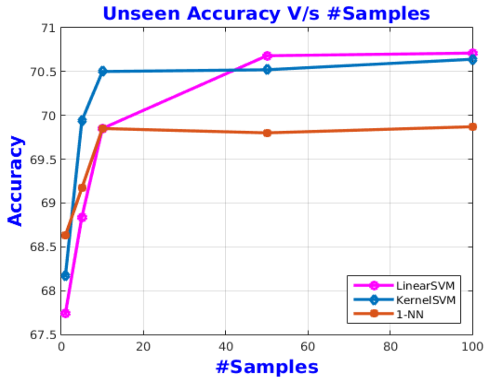

Figure 4. Classification accuracies with varying # of exemplars for AWA2 dataset.

7. Discussion and Conclusion

We have presented a robust generative framework to solve the generalized zero shot learning (GZSL) problem. Using a conditional VAE based architecture and augmenting it with discriminator-driven feedback mechanism enables our model to generate high-quality, class-specific exemplars from the unseen classes (and, if desired, also from seen classes). These exemplars can then be used in any classification model. This approach naturally helps us solve the GZSL problem since the learned classification model is not solely dependent on the labeled data from seen classes, but also leverages synthesized examples from unseen classes. Our model can easily leverage unlabeled examples from seen and/or unseen classes and can therefore also operate in a semi-supervised setting. The model and the results presented here strongly demonstrate the effectiveness of learning continuous space models with significant power of generating exemplars representative of the true distribution. While we use a VAE style generative model for our case, extending it to adversarial training should enhance generated exemplar quality in terms of sharpness and realness as noted in [12]. We also believe, the predictive power of the regressor can naturally improve performance in transductive ZSL settings [29], exploring which is a part of our future work. This can also be extended to online settings for few-shot learning where a small number of acquired labeled samples from the new classes can be used for improving the model.

Acknowledgment: Piyush Rai acknowledges support from Visvesvaraya Faculty Fellowship and a grant from Tower Research. Ashish acknowledges his travel support from "Indian Summer School on Computer Vision, IIIT-H". Gundeep Arora acknowledges his travel support from Microsoft and Tower Research. Vinay Verma acknowledges support from Visvesvaraya fellowship and Research-I Foundation.

References

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label-embedding for attribute-based classification. In *CVPR*, pages 819–826, 2013.

[2] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE Conference on CVPR*, pages 2927–2936, 2015.

[3] M. Bucher, S. Herbin, and F. Jurie. Generating visual representations for zero-shot classification. *CoRR*, abs/1708.06975, 2017.

[4] S. Changpinyo, W.-L. Chao, B. Gong, and F. Sha. Synthesized classifiers for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5327–5336, 2016.

[5] W.-L. Chao, S. Changpinyo, B. Gong, and F. Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *ECCV*, 2016.

[6] G. Dinu, A. Lazaridou, and M. Baroni. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568*, 2014.

[7] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, pages 2121–2129, 2013.

[8] Y. Fu and L. Sigal. Semi-supervised vocabulary-informed learning. In *CVPR*, pages 5337–5346, 2016.

[9] Z. Fu, T. Xiang, E. Kodirov, and S. Gong. Zero-shot object recognition by semantic manifold distance. In *CVPR*, pages 2635–2644, 2015.

[10] Y. Guo, G. Ding, J. Han, and Y. Gao. Synthesizing samples for zero-shot learning. In *IJCAI*, 2017.

[11] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. Toward controlled generation of text. In *ICML*, 2017.

[12] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[15] E. Kodirov, T. Xiang, Z. Fu, and S. Gong. Unsupervised domain adaptation for zero-shot learning. In *ICCV*, pages 2452–2460, 2015.

[16] E. Kodirov, T. Xiang, and S. Gong. Semantic autoencoder for zero-shot learning. *arXiv preprint arXiv:1704.08345*, 2017.

[17] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 951–958. IEEE, 2009.

[18] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on PAMI*, 36(3):453–465, 2014.

[19] Y. Long, L. Liu, L. Shao, F. Shen, G. Ding, and J. Han. From zero-shot learning to conventional supervised classification: Unseen visual data synthesis. In *CVPR*, 2017.

[20] Y. Long, L. Liu, F. Shen, L. Shao, and X. Li. Zero-shot learning using synthesised unseen visual data with diffusion regularisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2017.

[21] A. Mishra, M. Reddy, A. Mittal, and H. A. Murthy. A generative model for zero shot learning using conditional variational autoencoders. *arXiv preprint arXiv:1709.00663*, 2017.

[22] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013.

[23] M. Radovanović, A. Nanopoulos, and M. Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *JMLR*, 11(Sep):2487–2531, 2010.

[24] B. Romera-Paredes and P. H. Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, pages 2152–2161, 2015.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[26] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, pages 935–943, 2013.

[27] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[29] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE ICCV*, pages 4489–4497, 2015.

[30] V. K. Verma and P. Rai. A simple exponential family framework for zero-shot learning. In *ECML-PKDD*, pages 792–808. Springer, 2017.

[31] W. Wang, Y. Pu, V. K. Verma, K. Fan, Y. Zhang, C. Chen, P. Rai, and L. Carin. Zero-shot learning via class-conditioned deep generative models. *arXiv preprint arXiv:1711.05820*, 2017.

[32] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-ucsd birds 200. 2010.

[33] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele. Latent embeddings for zero-shot classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 69–77, 2016.

[34] Y. Xian, B. Schiele, and Z. Akata. Zero-shot learning-the good, the bad and the ugly. *arXiv preprint arXiv:1703.04394*, 2017.

[35] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR, 2010*, pages 3485–3492. IEEE, 2010.

[36] Z. Zhang and V. Saligrama. Learning joint feature adaptation for zero-shot recognition. *arXiv preprint arXiv:1611.07593*, 2016.