Zero-Shot Learning for Image Classification

Deepesh V. Hada [1] **Jayant Priyadarshi** [2] **Kavita Wagh** [2]

Abstract

We consider the problem of zero-shot recognition: learning a visual classifier for a category with zero training examples, just using some embedding to represent the category and its relationship to the other categories, for which visual data are provided. The key to dealing with the unfamiliar or novel category is to transfer knowledge obtained from familiar classes to describe the unfamiliar class. Zero-shot learning is a promising learning method, in which the classes covered by the training instances and the classes we aim to classify are disjoint. It refers to a specific use case of machine learning (and therefore, deep learning) where we want the model to classify data based on very few or even no labeled examples, which means classifying data on the fly.

1. Introduction and Literature Survey

In recent years, deep learning has achieved state-of-the-art performance across a wide range of computer vision tasks such as image classification. However, these deep learning methods rely on enormous amount of labeled data which is scarce for dynamically emerging objects. Practically, it is unrealistic to annotate everything around us, thus, making the conventional object classification methods infeasible.

We focus on the extreme case when there is no labeled data, *i.e.*, Zero-shot learning (ZSL), where the task is to recognize the novel categories' instances that have no labeled data available for training. ZSL assumes that the semantic label embeddings of both the seen and unseen classes are known apriori. ZSL, thus, learns to identify unseen classes by leveraging the semantic relationship between seen and unseen classes. It is important to note that the training and test classes are **disjoint** in ZSL.

ZSL algorithms generally project the seen and unseen class instances onto a latent space that is expected to be robust for learning unseen class labels. This latent space is shared by both seen and unseen categories and is aimed to be a mapping between images and their class embeddings. The learning is carried out with the help of semantic descriptors associated with each class. These approaches, thus, tend to learn a latent space that is aligned towards the semantic class embedding (**??????**), on which the input data is transformed. This transformation of data onto the latent space, however, makes these ZSL approaches suffer from the **hubness problem**: the transformed data vectors become *hubs* for the nearby semantic class embeddings, leading to performance deterioration (**???**). To mitigate the hubness problem, other approaches (**?????**) learn a latent space for recognizing the seen class labels by aligning the semantic class embeddings towards this latent space.

There exists an orthogonal approach, wherein artificial images (**?**) are generated to augment the training data. These synthetic images are usually generated from conditional generative models like conditional WGANs (**?**). **?** improved image classification considerably, but having GAN-based loss functions, suffer from instability in training. Recent developments have then been made along this approach using Conditional VAEs (CVAEs). **?** proposed an aligned VAE model, CADA-VAE, which learns a latent embedding of image features and class embedding via aligned VAEs optimized with cross-alignment and distribution-alignment objectives, and subsequently trains a classifier on sampled latent features of seen and unseen classes.

There has been another very different approach for zero-shot recognition using the semantic class embeddings and a **Knowledge Graph** (KG) that encodes the relationship of the novel category to some familiar categories. The first paradigm is to use implicit knowledge representations, *i.e.*, semantic embeddings, and hence, this is the same as the first set of approaches. The alternative paradigm for zero-shot learning in these approaches is to use explicit knowledge bases or KGs, wherein one explicitly represents the knowledge as rules or relationships between objects. These relationships can then be used to learn zero-shot classifiers for new categories (**?**). There has been a combination of the two such that the model distills both the implicit knowledge representations (i.e. semantic embedding) and explicit relationships (i.e. knowledge graph) for learning **Graph**

[1]Department of CSA, Indian Institute of Science, Bangalore, India [2]Department of EECS, Indian Institute of Science, Bangalore, India. Correspondence to: Deepesh V. Hada <deepeshhada@iisc.ac.in>, Jayant Priyadarshi <jayantp@iisc.ac.in>, Kavita Wagh <kavitawagh@iisc.ac.in>.

Convolutional Network (GCN) based visual classifiers of novel classes (**?**).

Also, irrespective of the latent space for transforming the data, there is an inherent **bias** in the model towards seen classes, which is referred to as the bias problem in literature. Due to this bias, the models generally perform poorly on unseen classes, and the predictions are primarily made in favour of the seen classes.

After considering these various approaches of GZSL for the task of image classification, we have implemented a model which we call **Alignment GAN using Spectral Normalization** (AGSN) *Generalized Inductive* ZSL, taking inspiration from (**?**). Our model gets rid of the hubness problem and the bias to a large extent. We have also applied Spectral Normalization (**?**) in our model. In our knowledge, Spectral Normalization has not been used in the task of ZSL yet. We apply Spectral Normalization over the parameters of our Generator and Critic, which helps contain the Lipschitz constant of the GAN model thus enhancing smooth gradient flow. While (**?**) learns a 1024 dimensional space of discriminative features, we directly learn discriminative features in the space of the visual input from the ResNet features. Moreover, we apply Batch Normalization over the generator and the critic, which increased the performance of our model.

2. Related Work

We discuss the state-of-the-art performing models in this section. The datasets which are predominantly used for training and evaluating ZSL models are **CUB** (**?**), **SUN** (**?**), **AwA** (**?**), **AwA2** (**?**) and **aPY** (**?**).

? take a different approach by constructing a new knowledge graph based on the Never-Ending Language Learning (NELL) for embeddings (**?**) and Never-Ending Image Learning (NEIL) for images (**?**) datasets and WordNet for word embeddings and ImageNet for images.

Since they evaluate their performances only on these datasets, their work cannot be compared with the other other models which evaluate on the aforementioned datasets.

A significant progress in ZSL was achieved by (**?**). Their approach takes feature generation one step further through a model (CADA-VAE) where a shared latent space of image features and class embeddings is learned through *modality-specific* aligned variational autoencoders, on which a Soft-Max classifier is trained. CADA-VAE has achieved state-of-the-art results on *AwA* and *AwA2*.

? proposed a model, GDAN, consisting of three components that perform GZSL, where the *Generator* represents the semantic $\rightarrow$ visual methods; the *Regressor* represents the visual $\rightarrow$ semantic methods; and the *Discriminator* represents metric learning. The *Regressor* is trained simultaneously with the CVAE using the cyclic consistency loss. GDAN has pulled off state-of-the-art results on the **SUN** (**?**) dataset.

3. Task

On the basis of data available during the training phase, ZSL can also be divided into two categories: **Inductive** and **Transductive** ZSL. In inductive ZSL (**????????**), we are only provided with the labeled seen class instances and the semantic embedding of unseen class labels during training. In transductive ZSL (**??**), in addition to the labeled seen class data and the semantic embedding of all labels, we are also provided with the **unlabeled instances** of unseen classes data during the training time.

Here, we assume that we do NOT have any unlabeled inputs corresponding to the unseen classes, and hence, we'll be dealing with **Inductive** ZSL.

Further, we also consider a more practical and realistic version of ZSL, the **Generalized Zero-shot learning** (GZSL) problem. In the **classical** ZSL, data emerges only from unseen classes at test time, *i.e.*, the performance of the algorithmic approach is solely judged on its classification accuracy on the novel unseen classes. In contrast, GZSL aims at maximizing performance on both seen and unseen classes, where the data during testing may come from both seen and unseen classes.

The task of GZSL is defined as follows:

$$D_s = \{(x^s, y^s, c(y^s)) \mid x^s \in X,\ y^s \in Y^s,\ c(y^s) \in C\} \tag{1}$$

is a set of seen training examples, consisting of image-features x^s, extracted by a CNN (ResNet-101), class labels y^s available during training and semantic class embeddings $c(y)$. The class-embeddings are hand-annotated attribute vectors. Note that unlike the Transductive ZSL, no auxiliary training set is provided.

With conventional ZSL, the aim of the classifier during test time is to map,

$$X \rightarrow Y^u \tag{2}$$

We also consider GZSL where the aim is to learn a classifier that maps:

$$X \rightarrow Y^u \cup Y^s \tag{3}$$

4. Model

We first use a pre-trained deep network **Resnet-101** to extract features from the seen class images. For notational convenience, we'll denote x_i^s as 2048-dimensional feature vectors extracted from the pre-trained Resnet model. The

model consists of four components with an adversarial setup. We now define these components and their tasks.

1. **Classifier (f_c)**: The classifier f_c, as the name suggests, learns to discriminate between the seen classes. A training instance, x_i^s is transformed by f_c to a one-hot encoding of its class label and then trained by minimizing the categorical cross-entropy loss:

$$L_C = -\frac{1}{N_s}\sum_{i=1}^{N_s} L(y_i^s, f_c(x_i^s)) \tag{4}$$

   where L is cross entropy loss between true and predicted labels of seen class instance x_s.

2. **Regressor (f_r)**: The regressor f_r preserves the semantic relationships among the seen class labels. It tries to ensure that the regressor output of a seen instance, *i.e.*, $f_r(x_i^s)$, is closely related to the corresponding semantic embedding, $c(y_i^s)$. We use a similarity based cross-entropy loss between the predicted label embeddings of the regressor and the true semantic label embedding:

$$L_S = -\sum_{i=1}^{N_s} log\frac{exp(< f_r(x_i^s), c(y_i^s) >)}{\sum_{y^s \in S} exp(< f_r(x_i^s), c(y^s) >)} \tag{5}$$

   Here, $< f_r(x_i^s), c(y_i^s) >$ refers to the *similarity* between predicted label embedding, $f_r(x_i^s)$. The similarity function that we have used is a simple normalized cosine similarity. One could use some other similarity functions like the *Jaccard* similarity as well.

3. **Adversarial Pair:** In inductive GZSL setup, we do not have training instances of the unseen classes which could pose a problem in our objective. We follow the approach of **?**, wherein we construct a *conditional generator* network that can generate artificial instances from the unseen embeddings.

   This conditional generator takes as input a random noise vector z and a class label embedding $c(y_s)$, and outputs an instance $\tilde{x}^s$ in the 2048-dimensional input space.

   We train the conditional generator using the *Wasserstein* adversarial loss defined by,

$$L_G^s = \mathbb{E}[D^s(x^s, c(y^s))] - \mathbb{E}[D^s(\tilde{x}^s, c(y^s))]$$

   where, D^s is the seen class conditional discriminator whose input is the seen class label embedding, $c(y^s)$, and one of the input space instance, x^s or the fake instance, $G(c(y^s))$. Thus, the objective of the generator-discriminator pair is to,

$$\min_{G^s}\max_{D^s} L_G^s \tag{6}$$

Further, we want to encourage the generator network to synthesize feature vectors that are discriminative between seen classes and encode the semantic similarity between the label embeddings, which are precisely what the Classfier and Regressor do, respectively. Thus, the overall loss function for the generator-discriminator network is,

$$\min_{G^s}\max_{D^s} L_G^s + \beta(L_C + \gamma L_S) \tag{7}$$

The generator can now be used to synthesize 2048-dimensional feature vectors for the unseen classes. However, the generator can be overly biased towards the seen classes due to the training set that is presented to it, which is a stumbling block. This bias is mitigated using the principle of early stopping during training the generator.

5. Experiments and Results

Our Generator is a a two layered Fully Connected network with LeakyReLU activation. The size of the hidden layer is 2048 dimensional. Batch Norm and Spectral Norm is applied after the first connected layer. ReLU activation is used in the last layer. The Critic is also a two layered Fully Connected network with LeakyReLU activation, Batch Norm and Spectral Norm after the first layer. The hidden layer size is 4096 for the critic. The classifier and the regressor are simple 1 layered connected network with appropriate dimensionalities. All the model parameters are pre-initialized with a Gaussian Distribution with zero mean and standard deviation of 0.02

After training the conditional GAN, we generate 100 artificial, visual features of each unseen class, expecting these artificial features to be semantically aligned to there embeddings and having the power of class discrimination. We append the dataset with these artificial features and hence the dataet now contains features from both seen as well as unseen classes.We finally train a Softmax Classifier over the appended dataset over all the classes. The final classifier is also a two layered fully connected network with LeakyReLU activation having a 2048 dimensional hidden layer.

Following WGAN-GP, we train the Discriminator five times per Generator update as the Discriminator is usually slow to learn. We use Adam Optimizer with a learning rate of 0.001 and betas as (0.5, 0.999) across the Generator and the Discriminator. We have also used Weight Clipping for the parameters of the Discriminator as suggested by WGAN-GP.

Software and Data

The repository can be found here and has been implemented in Python 3.7.4 using PyTorch.