Zero-shot Learning on the CUB Dataset

Tianshuang Fu, Zhe Cai, Jiahong Ji, Tangji Li

Abstract

Zero-shot learning (ZSL) aims at understanding unseen categories with no training examples. In our study, we focus on zero-shot learning in the context of image classification. We implement a pipeline consisting of two parts. We first run a CNN to learn attributes of images and then classify them based on these attributes. The dataset we use for this study is the Caltech UCSD bird (CUB) dataset. The dataset contains 200 bird species. We train our model on the 175 species. We mixed the known species and remaining 25 unseen species as the test dataset to test our model. Eventually, our model achieves a test accuracy about 31.2%.

1 Introduction

Zero-shot learning aims to recognize objects whose instances may not have been seen during training. In this project, we study this important problem in the context of image classification. We use the CUB dataset, one of the most widely used and competitive fine-grained classification benchmarks. The original dataset contains 200 bird species and more than 6000 images. We utilize the data of 175 bird species to develop our model. We also randomly select images from the whole 200 species (including 25 unseen species) to construct the test dataset. The test set contains 300 images from known dataset and 200 images from unseen dataset.

We divide the whole task into two main parts. The first part is training a CNN model. The goal of this part is learning/predicting the attributes of images. Once we get the attributes from the CNN model, we could then classify the image. In this step, the method we use is computing weighted similarities between the extracted attributes and the attributes of all the 200 bird species, including the unseen species and choosing the most similar one as the final result. After tuning our CNN model, we could get a model with 95.3% accuracy on validation data. Combining with the classification part, we could get a 31.2% accuracy on the test set contains the whole 200 bird species.

2 Proposed Approach

2.1 Problem Definition

We start by defining the zero-shot learning setting. The original CUB dataset includes images of 200 bird species with marked attributes. Among these species, we randomly select 175 species as known data that can be used to train the model. The remaining 25 species are mixed with the known data as test data to verify the success of zero-shot learning. For each species, the total number of attributes to describe it is 288. We also use the same 288 attributes to describe every image.

We train a CNN model as our first step. The CNN model takes in an image and outputs a 1×288 floating-point vector representing the probability of attributes presented in the image. We then convert the vector to a binary one by applying some threshold. The next step is to obtain a classifier, which takes in the 1×288 binary vector and outputs the bird species in that image. We collect each bird species' attributes from several online sources like Wikipedia, WhatBird. We define some rules to increase the weights of rarely appeared attributes. This step compares the resulting vector from the CNN model with all possible 200 species by computing the weighted similarities and returns the most likely one.

Combining these two steps, the input to the whole model is an image, and the output is a name of one of the 200 bird species.

2.2 Approach Overview

Image Pre-processing: The attributes that may be presented in an image are recorded in a separated text file. These records are pre-processed before passing into CNN training data in order to get a more accurate result. The processing includes calculating certainty and excludes conflicting/unlikely attributes.

We resize all images to a unified width and length based on the smallest image (264×121) to meet the CNN inputs requirement.

Dataset Splitting: Before training the CNN model, we further split the dataset of 175 known species into a training set, and a validation set randomly, consisting of 85% and 15% of the whole data, respectively. This step helps to validate outputs and avoid possible over-fitting situations. Besides, we randomly select 300 images from the known 175 species and 200 images from the unseen species to construct a test.

CNN: The Convolutional Neural Network (CNN) is a class of deep learning neural networks. Although there are many different structures and implementations, the basic structures are similar. We select the model by comparing validation loss, training time, and the model's flexibility. Table 1 shows the architecture of ResNet50, a widely used CNN model in image processing projects. (He et al., 2016)

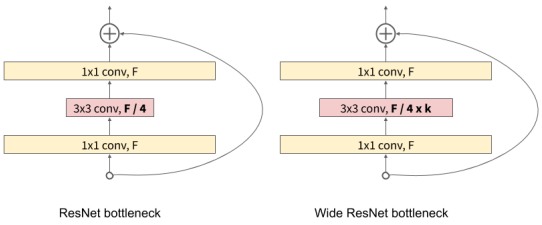

Figure 1: Wide ResNet v.s. ResNet

The model we use is Wide ResNet50 (Wide Residual networks), (Zagoruyko and Komodakis, 2016). Compared to ResNet, it increases the number of channels, as Figure 1 shows. The better evaluation result obtained using this model is illustrated in chapter 3.

Classification: The most intuitive approach to get the similarity between two vectors is to count how many values at the same positions are the same, using the euclidean distance between vectors $A, B \in (x_1, x_2, x_3, ..., x_{288})$.

Layer Name	Output Size	ResNet50 Layer
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3, maxpool, stride 2 $\begin{bmatrix} 1 \times 1, & 64 \\ 3 \times 3, & 64 \\ 1 \times 1, & 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 1 \times 1, & 128 \\ 3 \times 3, & 128 \\ 1 \times 1, & 512 \end{bmatrix} \times 4$
conv4_x	14×14	$\begin{bmatrix} 1 \times 1, & 256 \\ 3 \times 3, & 256 \\ 1 \times 1, & 1024 \end{bmatrix} \times 6$
conv5_x	7×7	$\begin{bmatrix} 1 \times 1, & 512 \\ 3 \times 3, & 512 \\ 1 \times 1, & 2048 \end{bmatrix} \times 3$
	1×1	average pool 1000-d fc softmax
FLOPs		3.8×10^9

Table 1: ResNet50 architechture

$$d(A, B) = \sqrt{\sum_{i=1}^{288} (a_i - b_i)^2} \quad (1)$$

Although it is straightforward and effective in most cases, this approach assumes every attribute has the same weight. Therefore, it cannot assign a higher penalty to some crucial mistakes or conflicts generated by CNN or the original labelling. A minor mistake generated in CNN may lead to a total misclassification. As a result, the whole test accuracy decreases.

We then try to assign different weights to different attributes when computing the similarities. Some features are examined more carefully because of their decisive level or overall consistency in the given dataset. In this way, we lower the impacts of possible CNN errors and help make better predictions. The detail is elaborated in the next part.

Weighting Attributes: In the training process, we have 175 bird species and 288 attributes per species. Using the given label, we create a 175×288 matrix to indicate whether the feature present in the bird or not. We denote it as $M_{feature}$.

Considering the attributes' distribution, we no-

tice some attributes are common among all the species while other features may be very rare among all the birds. We assume that the distribution of attributes in unseen bird species is similar to that in known bird species. Therefore, it is essential to set different weights on different attributes. In general, except for attributes that never appear, the rarer the attribute occurs among all the species, the higher weight should be set.

The first step is to count how many times an attribute occurs. A 1×288 vector w_{count} is created to represent the result of counting. Equation 2 explains the counting process.

$$w_{count}[i] = \#\ of\ birds\ have\ feature\ i \quad (2)$$

Then we normalize w_{count} and get w_{norm_count} using the Equation 3. After that, all the values are between 0 to 1.

$$w_{norm\_count} = \frac{w_{count}}{max(w_{count})} \quad (3)$$

Observing the data normalize data, several attributes occur in none of the bird species at all. Thus, their normalized count values are zeros. Statically speaking, these values are insignificant, since it useless in predicting the bird species. We need to eliminate zeros since we want to use a log function to assign weights and $log(0)$ may cause an attribute's weight approach to infinity. Therefore, we replace all zeros with ones them to ones. We get the vector w_fixed using the Equation 4.

$$w_{fixed}[i] = \begin{cases} 1, & w_{norm\_count}[i] = 0 \\ w_{norm\_count}[i], & otherwise \end{cases} \quad (4)$$

We try to set the weight of each attribute to a value between $[0, \infty]$. Equation 5 shows how we get the final weight vector w_{final}. In order to control and tune the weight distribution later, we use two hyperparameters. k is a coefficient that controls the scaling. b is the base of the log function.

$$w_{final}[i] = -k * log_b(1 - w_{fixed}[i]) \quad (5)$$

Figure 2 and Figure 3 illustrate the result attributes counting and the weighted coefficient of each attribute.

S

2.3 Expectations

Using the pipeline constructed by the CNN and Classification, our model should be able to make

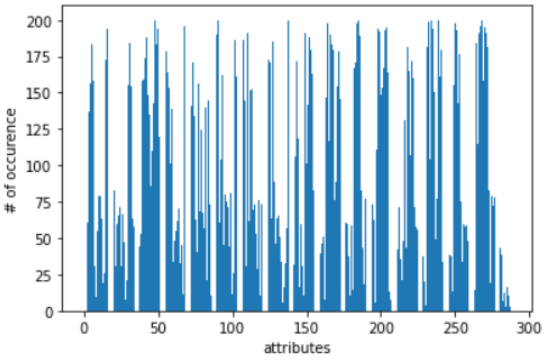

Figure 2: Attribute Occurrence Count

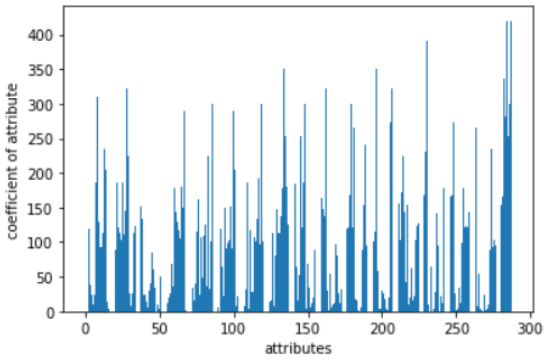

Figure 3: Weights of Attributes

quite a few correct predictions given images of different bird species, even they are not used in the training process.

Specifically, we expect our CNN model to achieve an accuracy of more than 90% using one of the best models. We expect that the classification model uses weighted attributes to perform better than the unweighted model. Eventually, we expect the whole model could achieve an accuracy significantly better than the result of randomly guess to prove the effectiveness of this zero-shot learning study.

3 Evaluation

3.1 CNN

The main metrics used to evaluate the CNN model is the validation score. We tried several widely used pre-trained models. Their performances are shown in Table 2. The best model after 20 epoch is the Wide ResNet101, which has a validation score of 0.8978. However, Wide ResNet50 has a performance near the best, but the time to train it is significantly less. Therefore, we finally chose Wide

Model	Validation Score	
	0 Epoch	20 Epochs
GoogleNet	0.5361	0.8572
ResNet18	0.5027	0.8664
ResNet50	0.5538	0.8870
ResNet101	0.5277	0.8891
Wide ResNet50	0.5480	0.8963
Wide ResNet101	0.5505	0.8978

Table 2: CNN Model Performances

ResNet50 as our CNN model. After training the model for 50 epochs, the model can reach a validation score of 95.3%.

3.2 Classification

We use the final prediction accuracy to evaluate the classification task.

First, we try the unweighted attributes model. This most intuitive classification presents 20.3% accuracy. To improve the accuracy and exhibit the importance of weighting attributes, we assign different weights on different features based on its appearance. According to Equation 5, we try tuning k and b to see if there exists optimal values of either of these two parameters. The different performances are in Table 3 and Table 4.

k	Accuracy
1	31.2%
5	31.2%
10	31.2%
100	31.2%
1000	31.2%

Table 3: Accuracy Corresponding to k

b	Accuracy
2	31.2%
e	31.2%
10	31.2%
100	31.2%
1000	31.2%

Table 4: Accuracy Corresponding to b

We notice that changing the parameters do not have any effect on the performance of the final result. However, compared to the unweighted attributes model, the weighted attributes model's performance is much better. The comparison is shown in Table 5.

Model	Accuracy
Unweighted model	20.3%
Weighted model ($k = 1, b = e$)	31.2%

Table 5: Weighted Model v.s. Unweighted Model

Based on these data, it is sufficient to conclude that using the assigning different weights to attributes can significantly increase the accuracy.

4 Related Work

We have read several papers related to zero-shot learning. Axoue's group (Li et al., 2019) distinguished images with subtle distinctions using zero-shot fine-grained classification based on the same dataset we use. They constructed deep convolutional neural networks and also induced a domain adaptation to avoid domain shift. Besides, they used a semantic directed graph over attributes based on fine-grained classes. The classification could implement the thought of zero-shot method to some extend. Their models reached an accuracy of 49.5% on unseen data eventually. In 2018, Xiaolong Wang's group (Wang et al., 2018) implemented zero-shot recognition using semantic embeddings and knowledge graphs. They learned a visual classifier with zero training examples and compared the relations between categories. They mainly dealt with the novel or rare classes recognition based on transferring the learned classes. The visual classifier built the Graph Convolutional Network and used semantic embeddings together with relational categories for prediction. They achieved 2% to 3% improvement in performance compared with state-of-the-art results, which is impressive.

We did some research on ResNet CNN model as well. For Kaiming He's work, they identified mappings in deep residual networks and improved the generalization using the 1001-layer ResNet on CIFAR-10 and CIFAR-100.(He et al., 2016) Another group of Sanghyun Hong also constructed complex ResNet50 network, observed only one forward propagation. They first analyzed the exploit cache side-channels attacked by fingerprinting, making in-depth security analysis with deep neural networks.(Hong et al., 2018) Then, with the

introduction of DeepRecon, they successfully reconstructed the network architecture. The apply of ResNet50 plays a significant role in reconstructing it correctly. Finally, they evaluated the new framework's impacts. The apply of ResNet50 gave us some directions on how to construct a complex network correctly.

5 Future Work

We believe the further improvements we could do will mainly lie on the classification part. We could possibly construct a more complex model focusing on the distributions of the attributes among similar birds species. We could also further tune the weight of one attribute on different bird species, based on whether this attribute is a key attribute of one specific bird species.

6 Conclusion

We implement a zero-shot learning model consists of a CNN model and a classification function based on weighted attributes and run the model on the CUB dataset. In our study, our model achieves a final accuracy of 31.2% on a test set contains a mixture of known data of 175 bird species and unseen data of 25 bird species. This result demonstrates that our model successfully achieves the zero-shot learning goal.

Our demo video can be found here.

References

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. pages 770–778.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

Sanghyun Hong, Michael Davinroy, Yiğitcan Kaya, Stuart Nevans Locke, Ian Rackow, Kevin Kulda, Dana Dachman-Soled, and Tudor Dumitraş. 2018. Security analysis of deep neural networks operating in the presence of cache side-channel attacks. *arXiv preprint arXiv:1810.03487*.

Ao-Xue Li, Ke-Xin Zhang, and Li-Wei Wang. 2019. Zero-shot fine-grained classification by deep feature learning with semantics. *International Journal of Automation and Computing*, 16(5):563–574.

Xiaolong Wang, Yufei Ye, and Abhinav Gupta. 2018. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6857–6866.

Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. pages 87.1–87.12.