Predict Deep Latent Classifier for Unseen Category: A New Deep Zero-Shot Learning Approach

Shih-Yen Tao
Language Technologies Institute
shihyent@andrew.cmu.edu

Abstract

Zero shot learning (ZSL) aims to recognize unseen image classes using knowledge from the seen class images and side information like semantic descriptions or human-defined attributes. In this project, we propose a deep learning algorithm for effectively relating visual and semantic information to do ZSL. We evaluate our model on three benchmark ZSL datasets and show that our model performs comparably against other ZSL methods.

1 Introduction

1.1 Traditional Image Classification

With the progress of deep convolutional neural network, machines can do image classification task at a human level [9]. The pipeline for the traditional visual classification neural network is shown in Fig 1. In short words, the whole network consists of two blocks: a feature extractor and a classifier. Generally speaking, the convolutional neural network is used to extract image features. The Classifier is usually a fully connected layer followed by a softmax activation function to obtain the score (probability) for each class. We can learn the model in two different ways: the most common mythology is to do end-to-end learning. That is to say, the CNN feature extractor and the classifier is learned by backpropagation at the same time. The second one is to use the pretrained feature extractor and only learn the classifier. This strategy can save a lot of time in practice.

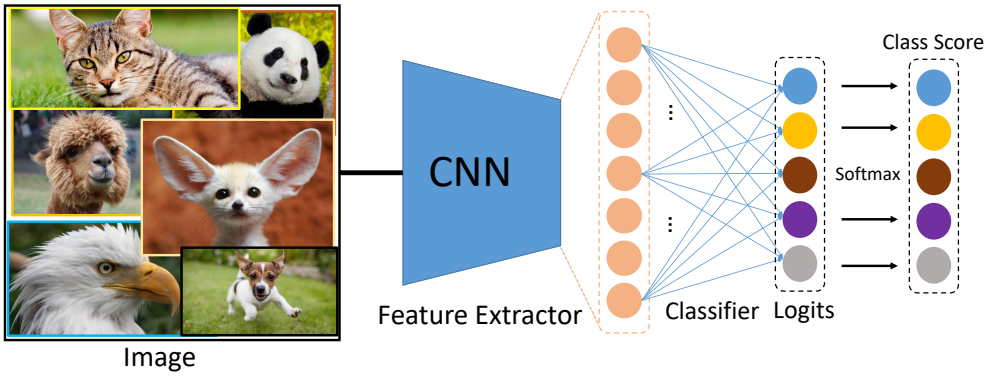

Figure 1: Illustration for traditional image classification framework.

1.2 Zero-Shot Image Classification

In real-world scenario, new categories appear rapidly. For instance, the new spice of animals emerges every day. How can we enable the model to recognize those unseen classes which are not presented in the training stage?

Zero-Shot learning (ZSL) can solve this problem. ZSL aims to recognize unseen classes by transferring the knowledge from seen classes. A practical way to do this challenging task is to leverage information from another semantic domain. To be more specific, each class (including seen and unseen classes) is represented by one semantic vector, which can be based on either human-defined attributes ("brown hair", "furry", "tail", etc) or unsupervised word embeddings like Word2Vec. Overall, in the training stage of ZSL, seen class images and corresponding semantic vectors are given, and the model is trained to associate information across the visual and semantic domain. In the test stage, the semantic vectors for unseen classes are given, and the model can classify test images from unseen categories.

The intuition behind this approach is easy to explain. Say we use attributes vector for the semantic space. In the training stage, the model could learn what should a class with attributes "furry" or "tail" look like. Finally, in the test stage, say we have an unseen category "cat" with attribute "furry" and "tail", the model can use the learned knowledge to do classification accordingly.

In this project, we tackle ZSL by using deep learning to predict the *classifier* for unseen classes. To be more specific, given the *semantic vectors* for new unseen categories, our model can generate the *fully connected layer* classifier directly. Thus, even though we don't have the image instances for unseen classes, we can still do visual recognition task. The detail of our method can be found in Section 3.

2 Related work

2.1 Different Semantic Spaces

Good semantic space can help ZSL a lot since it connects seen and unseen classes. There are several semantic spaces that are used in the existing ZSL literature. In the beginning, the supervised human-defined attributes vector is used in [13, 18, 31, 23, 33, 6, 16]. However, attributes vectors cannot scale up when the number of classes increases. Secondly, in the case of fine-grained classification task, it often requires expert knowledge to define useful attributes. As the result, recently the academic society shifts to use unsupervised word embedding instead [2, 7, 22]. For instance, we can use the famous word embedding methods like Word2Vec [20], Glove [24], and WordNet vector [21].This kind of semantic embedding is much more suitable when the number of classes is huge. Nevertheless, such unsupervised semantic space is noisy rather noisy compared to the supervised attribute semantic representation.

2.2 Direct Transformation Methods

There is a wide range of ZSL methods. The most straightforward one is based on learning transformation from the visual space to the semantic space [13]. However, such methods don't perform well when there is a strong correlation between attributes. As the results, works like [18, 31] take the attributes correlation into consideration explicitly and have improved classification results. SOC [23] is one strong baseline for this category of methods. It used multiple output linear regression to learn the transformation. Other transformation-based methods are [7, 27, 22, 3].

2.3 Classifier Based Methods

Other ZSL methods are classifier-based approaches. To be more specific, this kind of method directly find the classifiers for the unseen data by using the existing classifier for the seen classes. For instance, Co-Occurrence Statistics (COSTA) [19] construct the classifier by using the co-occurrence statistics for the semantic space from the web. Moreover, [4] proposed to synthesize classifier by using the similarity between semantic vectors for each class.

2.4 Common Feature Space Methods

However, the direct transformation and the classifier methods both suffer from the fact that there is always *domain discrepancy* between visual and semantic domain. That is to say, two classes are similar in one space don't guarantee they are also close to each other in the other space. For instance, "shark" and "fish" might are close to each other in the Word2Vec space while their visual appearance is quite different. As the result, naively mapping the visual instance to the semantic space isn't an optimal method. Also, synthesizing classifiers based on the semantic vector similarity has this drawback. In order to alleviate the domain discrepancy problem, recent ZSL approached aim to seek for common representation across visual and semantic space. For instance, approaches like [1, 2, 25, 32] learn a bilinear function to associate visual and semantic space. [34, 35] both seek for a semantic similarity latent space and get promising results. Moreover, [17, 8] leverage Canonical Correlation Analysis (CCA) to derive the visual-semantic latent space and [26] boost the performance by introducing a manifold regularizer. We show in the next section that our proposed model is a *combination* of the classifier and common feature space method.

3 Proposed method

3.1 Problem Settings and Notations

Let $\mathcal{D} = \{\mathbf{X}, Y\} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ denote training data in the visual domain, where $\mathbf{x}_i \in \mathbb{R}^{d_F}$ represents the ith instance (it can be an image or an extracted feature), and y_i is its corresponding label from $\mathcal{L} = \{1, 2, .., C\}$. For ZSL, we have $\mathcal{D}^U = \{\mathbf{X}^U, Y^U\} = \{\mathbf{x}_i^U, y_i^U\}_{i=1}^{N^U}$ as test data, where $\mathbf{x}_i^U \in \mathbb{R}^{d_F}$ denotes the ith test instance and y_i^U is the label from the *unseen* label set $\mathcal{L}^U = \{1^U, 2^U, ..., C^U\}$. It is worth noting that in this project we only consider the case where the label sets $\mathcal{L}$ and $\mathcal{L}^U$ from training and testing data are *disjoint* (i.e., $\mathcal{L} \cap \mathcal{L}^U = \varnothing$). Note that each class is associated with a semantic vector in a d_S dimensional space. Thus, we have $\mathcal{S} = \{\mathbf{s}_i \in \mathbb{R}^{d_S}\}_{i=1}^{C}$ and $\mathcal{S}^U = \{\mathbf{s}_i^U \in \mathbb{R}^{d_S}\}_{i=1}^{C^U}$ as semantic vectors of training data and test data, respectively. Our goal is to predict unseen labels $\{y_i^U\}_{i=1}^{N_U}$ of test data by leveraging visual and semantic-domain data.

3.2 Predict Classifier for Unseen Classes

As shown in Fig 1, the main task for the traditional visual recognition framework is to learn the fully connect layer classifier $\mathbf{W} \in \mathbb{R}^{d_F \times C} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_C]$, where C is the number of classes and $\mathbf{w}_j \in \mathbb{R}^{d_F}$ corresponds to the hyperplane for class j. After $\mathbf{W}$ is learned by backpropagation, a test image $\mathbf{x}$ can be classified by $f(\mathbf{x}) = \arg\max_i \mathbf{w}_i^\top \mathbf{x}$. Note that this pipeline cannot be directly applied on ZSL, since in the training stage we don't have visual instance from unseen categories so we cannot use backpropagation to learn the associated classifier $\mathbf{W}^U \in \mathbb{R}^{d_S \times C^U} = [\mathbf{w}_1^U, \mathbf{w}_2^U, ..., \mathbf{w}_C^U]$, where $\mathbf{w}_j^U \in \mathbb{R}^{d_F}$ denotes the weight corresponding to unseen class j^U.

To tackle this issue, we assume that each weight $\mathbf{w}_i$, $\mathbf{w}_i^U$ from seen and unseen classes are both "transformed" from the associated semantic vector $\mathbf{s}_i$, $\mathbf{s}_i^U$. To be more specific, there exists a global function f_s such that $f_s(\mathbf{s}_i) = \mathbf{w}_i$ and $f_s(\mathbf{s}_i^U) = \mathbf{w}_i^U$. We further assume that $f_s(\mathbf{w}) = f_s(\mathbf{w}; \theta_s)$ can be parameterized by a deep neural network θ_s. Therefore, after the model θ_s is learned, we can directly predict the classifier weight $\mathbf{w}_i^U = f_s(\mathbf{s}_i^U; \theta_s)$ as long as the semantic vector $\mathbf{s}_i^U$ for unseen class i^U is given. As the result, we can perform zero-shot classification accordingly.

3.3 Domain Invariant Latent Space Learning

We note that there is always *domain discrepancy* across the visual and semantic domain. That is to say, the similarity between classes may not be consistent across domains. Therefore, it will be sub-optimal if we images are classified in the form of original feature extracted from CNN. Therefore, we propose to embed images in a deep domain invariant latent space for classification. To be more specific, we aim to learn a neural network $f_v(.; \theta_v)$ represented by θ_v that produces latent feature $\mathbf{x}_l \in \mathbb{R}^{d_k}$ for image $\mathbf{x}$. That is to say, $\mathbf{x}_l = f_v(\mathbf{x}; \theta_v)$. The classification is done in the latent space instead of the original visual feature space.

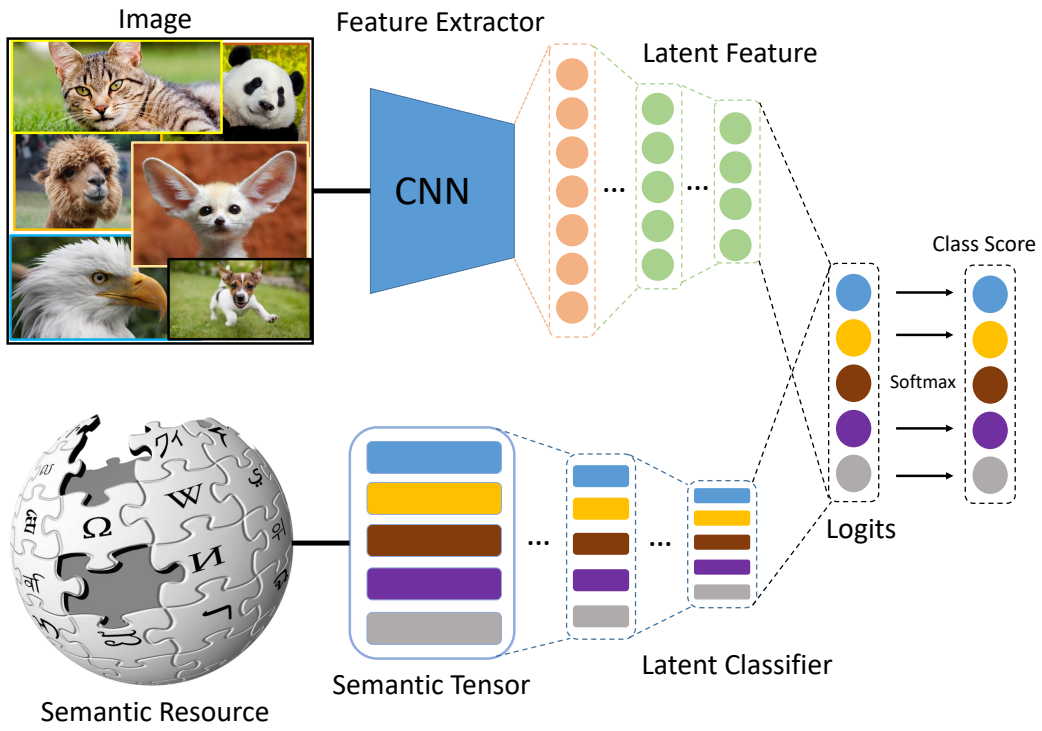

Figure 2: Illustration for our proposed method.

3.4 Our Zero-Shot Learning Method

Our model is summarized in Fig 2. We jointly learn two neural networks $f_s(.;\theta_s) : \mathbb{R}^{d_F} \rightarrow \mathbb{R}^{d_k}$, $f_v(.;\theta_v) : \mathbb{R}^{d_S} \rightarrow \mathbb{R}^{d_k}$ by solving the following optimization problem.

$$\min_{\theta_s,\theta_v} \frac{1}{N} \sum_{i=1}^{N} loss(\hat{y}_i = g(f_v(\mathbf{x}_i;\theta_v), f_s(\mathbf{S};\theta_s)), y_i), \tag{1}$$

where $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, .., \mathbf{s}_C]$ is the matrix consist of all seen semantic vectors (semantic tensor in Fig 2). We can see that f_s projects $\mathbf{S}$ into the classifier matrix (Latent classifier in Fig 2) $\mathbf{W} = \mathbb{R}^{d_k \times C}$ in the latent space. Moreover, $g(\mathbf{x}_i, \mathbf{W})$ simply outputs the largest index for the predicted score vector $\mathbf{W}^\top \mathbf{x}_i$. Furthermore, we use softmax activation layer to obtain the probability distribution for each image $softmax(\mathbf{W}^\top \mathbf{x}_i)$ and compute cross-entropy loss against the ground truth y_i. The whole model can be learned in an end-to-end fashion by backpropagation.

3.5 Model Configuration

For f_v, we use a two-layer DNN and set the hidden layer dimension to 400. Sigmoid is used for the activation function. For f_s, we simply adopt a linear transformation layer to prevent overfitting. Moreover, we leverage batch normalization to stable the training process.

4 Experiments

4.1 Datasets

In order to compare our proposed method to other ZSL literature, we use the datasets in [32] for evaluation: CUB-200-2011 Birds (**CUB**) [28], Stanford Dogs (**DOG**) [10], and Animal with Attributes (**AWA**) [14]. Moreover, we use the seen/unseen split suggested by [32]. For the semantic

space, we use Word2Vec and Glove, which are both efficient unsupervised word embedding learned from Wikipedia raw data. Moreover, we also consider the human-defined attribute semantic vector provided in the original datasets. For the visual data, in order to accelerate the learning process, we use the extracted feature from pretrained GoogleNet for our image representation, which refrains us from fine-tuning the GoogleNet backbone and focus on learning the deep latent space for ZSL. The detailed description of datasets can be found in Table 1.

Table 1: Descriptions of the datasets.

	CUB	**DOG**	**AWA**
# of seen classes	150	85	40
# of unseen classes	50	28	10
# of images	11786	19499	30473
Dim of Attribute	312	-	85
Dim of Word2Vec	400	400	400
Dim of Glove	400	200	400
Dim of Wordnet	-	163	-

4.2 Implementation and Hyperparameters

We implement our model by Tensorflow and run the experiments on a laptop with Nvidia GeForce 840M GPU. The only Hyperparameter for our model is the dimension of the latent space d_k, which is fixed to d_s. Moreover, Adamgrad is chosen for the optimizer and the learning rate is set to 0.01. For **AWA** dataset, since the class number is fewer, we training the model 50 epochs, while the epoch number for **CUB** and **Dogs** is set to 150 and 100, respectively.

4.3 Experiment Results

In this section, we compare our method with other ZSL approaches. We choose one method for each category of ZSL mentioned in Sec 2.1. For the direct transformation method, we adopt SOC [23] since it is simple to implement and has the strongest performance among other transformation approaches. For the common feature space method, we choose LatEm [32] because we are using the same dataset and can cite their results directly. Finally, we choose Sync [4] for the classifier method since it has a strong performance. The final results can be found in Table 2.
We could observe that our method outperform SOC and LatEm and is comparable with Sync. The reason why our method is better than SOC is that SOC only learns a direct linear transformation from visual to semantic space. First, a linear transformation may be too simple to the dataset. Second, as mentioned before, there is domain discrepancy across domains, so naive mapping method won't work well. As for LatEm, even though it aims to seek a common feature space for visual and semantic space, the model is again limited to linear transformation, which is the reason why our method surpasses its results. So why Sync is so good? First, generally speaking, it is a nonlinear method, which can solve a more complex problem. Second, unlike our method which directly generates classifiers for unseen class by semantic vectors, Sync further aims to learn the "base" classifiers (phantom classifier in the original paper) which can be more general. As the result, in some case, Sync is more suitable than our proposed approach.

Table 2: Performance comparisons.

Methods	**CUB**			**AWA**			**DOG**		
	Attribute	Word2Vec	Glove	Attribute	Word2Vec	Glove	Word2Vec	Glove	Wordnet
SOC [23]	34.7	30.9	30.6	58.6	50.8	68.0	24.6	17.8	17.3
LatEm [32]	45.5	31.8	32.5	71.9	61.1	62.9	22.6	20.9	25.2
Sync [4]	48.7	31.2	**32.8**	72.9	**62.0**	67.0	**28.0**	**20.4**	**30.7**
Ours	**50.4**	**33.2**	31.3	**74.4**	60.2	**68.6**	26.7	19.9	27.3

4.4 Discussion on Zero-Shot Learning Behavior

We show the learning curve for all experiment in Fig 3. We first note that our algorithm converges quickly, which implies that our method is efficient. Second, we can observe that the accuracy of the training data is much higher than the test data. Take CUB dataset with attribute semantic for instance, we can get around 85% on the training set while the test accuracy saturates at 50%. This fact implies

the limit of the semantic vector, and the semantic space we are using is far from the optimal one. That is to say, even though we take the domain discrepancy across domain into account explicitly in our model, the issue still can't be solved completely until better semantic vector is created and used (For instance, more accurate human-defined attributes vector).
Moreover, as the training accuracy is much better than the one on the test data, we could conclude that overfitting might occur in Zero-Shot learning scenario easily. As the result, when designing algorithm for ZSL, the model complexity shouldn't be too high. For instance, in our model, linear transformation for the semantic space is enough to produce satisfactory results. If we deepen the neural network for f_s or add nonlinear activation function, the model overfits training data easily.

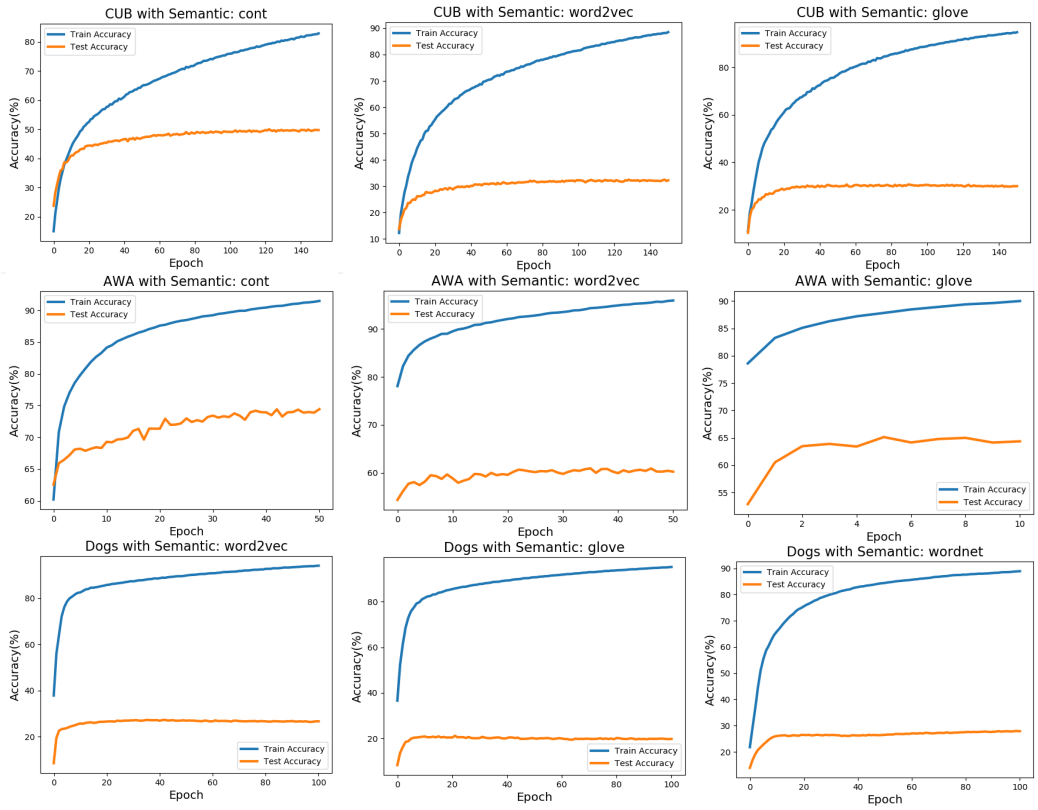

Figure 3: Learning curves.

4.5 Deep Look at Each Component of Our Model

We present the results of our intermediate and final model in Tabel 3. Before having a deep look into each component, we note that our final model has the best result, each implies that every component of our model is important.
model$_{linear}$ denote the model with one linear layer only. That is to say f_v and f_s are both linear network. Model$_{one\,layer}$ means we reduce f_v from two layers to one layer while the nonlinear activation function is kept. These two models have similar results so we discuss them together. We can observe that even though in most of the settings the models have a comparable result with the final model, it has really bad performance in some cases. For instance, the results for AWA with unsupervised semantic space is not good. This fact shows that simple model cannot solve ZSL and we need non-linearity to boost the network.
Secondly, we discuss the importance of the batch normalization layer. We could observe that model$_{no\,batch}$ consistently produce poor results. The fact shows that using batch normalization can regularize the feature at each layer and stabilize the training process. In the experiment, we also find out the learning curve converges slower without normalization. As the result, we conclude that this particular trick is crucial for training deep neural network and should be used for other models as

well.
Last but not least, we discuss the role of domain invariant space learning in our model. Note that model$_{no\ f_s}$ and model$_{no\ f_v}$ fix f_s and f_v to identity function, respectively. That is to say, these two models directly map one space to the other space. We could observe that the two models' results are bad (as the level of SOC). This implies that there is a huge domain discrepancy across domains, and it is beneficial to find a latent space which can embed visual and semantic information jointly.

Table 3: Comparison between different component of our model.

Methods	CUB			AWA			DOG		
	Attribute	Word2Vec	Glove	Attribute	Word2Vec	Glove	Word2Vec	Glove	Wordnet
model$_{linear}$	48.3	29.2	29.2	73.0	53.2	52.3	26.6	16.7	**27.3**
model$_{one\ layer}$	49.7	32.2	29.3	71.1	59.9	61.8	26.8	19.0	22.7
model$_{no\ batch}$	44.1	28.4	26.3	65.0	**61.9**	65.4	24.1	17.5	25.6
model$_{no\ f_v}$	47.0	29.8	27.1	74.3	58.0	49.6	**28.7**	18.6	27.1
model$_{no\ f_s}$	42.5	29.0	**31.6**	68.3	60.5	67.4	18.2	16.3	12.6
Ours	**50.4**	**33.2**	31.3	**74.4**	60.2	**68.6**	26.7	**19.9**	**27.3**

4.6 Latent Space Visualization

We use t-SNE to visualize the latent embedding in our model in Fig 4 for three datasets. We could observe that for AWA there exist obvious cluster for visual instance in the latent space while CUB dataset doesn't. This fact can explain why AWA produce better classification result than CUB. However, comparing Dogs with CUB, we note that Dogs dataset has better visual cluster while results are worse. As the result, we can conclude that the semantic vectors for Dogs dataset are not discriminative enough to perform good zero-shot recognition.

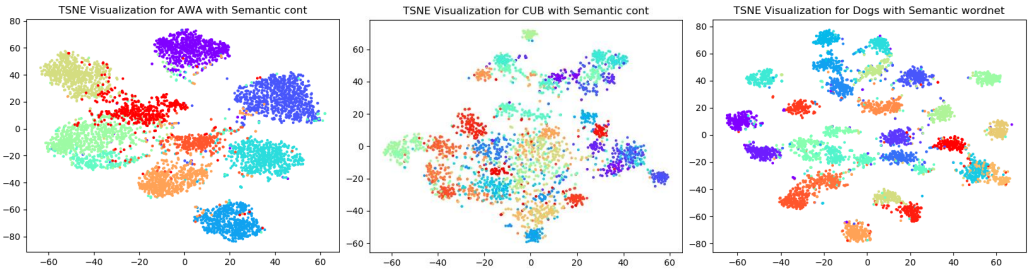

Figure 4: t-SNE Visualization.

4.7 Parameter Sensitivity

In this section, we test the parameter sensitivity of our ZSL neural network model. The hyperparameter in our model is the dimension of the latent space d_k. We note that our final choice of $d_k = d_s$. We use CUB with attribute semantic in this experiment. The performance against different ratio $\frac{d_k}{d_s}$ is shown in Fig 5. We could observe that the performance saturates at $d_k = d_s$ and hence justifies our choice of d_k.

5 Conclusion

In this project, we propose a new deep learning approach for ZSL. The key idea of our model is to use a deep neural network to directly predict the classifier for unseen classes using the associated semantic vector. Moreover, we propose to learn a deep latent embedding for visual instance where domain discrepancy across domains is eliminated. We compare our approach to three ZSL method and yield better or comparable results. Moreover, comprehensive analysis of different blocks of our model is conduct as well.

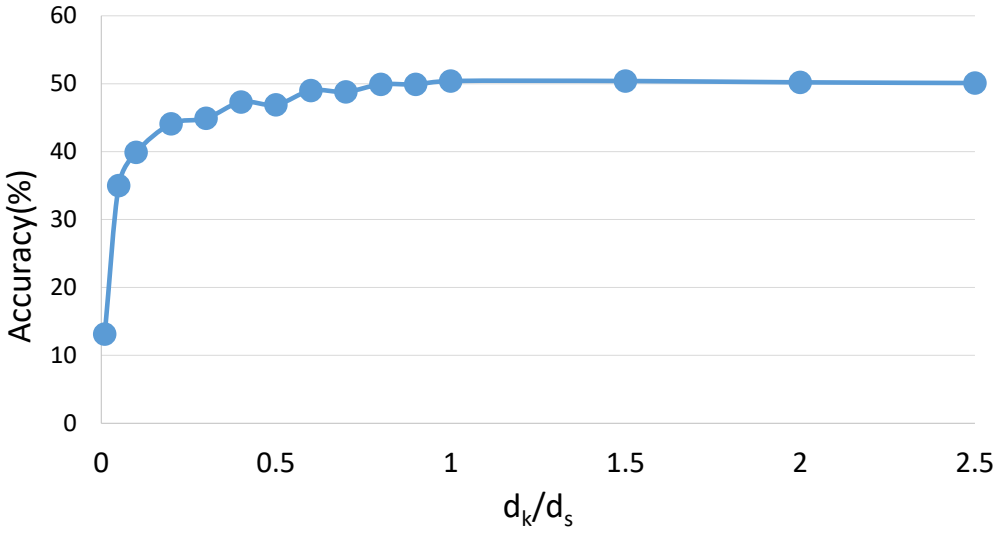

Figure 5: Influence on different dimension of latent space.

6 Future Plan

We fix the feature extractor network in this project. In the future, we can fine-tune it as well as and see the results. Moreover, we will like to see our result on a larger real-world dataset, like Imagenet for the future plan.

References

[1] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for attribute-based classification. In *CVPR*, 2013.

[2] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *CVPR*, 2015.

[3] Maxime Bucher, Stéphane Herbin, and Frédéric Jurie. Improving semantic embedding consistency by metric learning for zero-shot classiffication. 2016.

[4] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *CVPR*, 2016.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[6] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *CVPR*, 2009.

[7] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013.

[8] Yanwei Fu, Timothy M Hospedales, Tao Xiang, and Shaogang Gong. Transductive multi-view zero-shot learning. *TPAMI*, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *CVPR*, 2011.

[11] Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Unsupervised domain adaptation for zero-shot learning. In *ICCV*, 2015.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[13] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.

[14] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *TPAMI*, 2014.

[15] Xin Li, Yuhong Guo, and Dale Schuurmans. Semi-supervised zero-shot classification with label representation learning. In *ICCV*, 2015.

[16] Jingen Liu, Benjamin Kuipers, and Silvio Savarese. Recognizing human actions by attributes. In *CVPR*, 2011.

[17] Yao Lu. Unsupervised learning on neural network outputs: with application in zero-shot learning. In *IJCAI*, 2016.

[18] Dhruv Mahajan, Sundararajan Sellamanickam, and Vinod Nair. A joint learning framework for attribute models and object descriptions. In *ICCV*, 2011.

[19] Thomas Mensink, Efstratios Gavves, and Cees GM Snoek. Costa: Co-occurrence statistics for zero-shot classification. In *CVPR*, 2014.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *ICLR*, 2013.

[21] George A Miller. Wordnet: a lexical database for english. *ACM*, 1995.

[22] Mohammad Norouzi and Mikolov et al. Zero-shot learning by convex combination of semantic embeddings. In *ICLR*, 2014.

[23] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009.

[24] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[25] Bernardino Romera-Paredes and PHS Torr. An embarrassingly simple approach to zero-shot learning. In *ICML*, 2015.

[26] Yi-Ren Yeh Yu-Chiang Frank Wang Shih-Yen Tao, Yao-Hung Hubert Tsai. Semantics-preserving locality embedding for zero-shot learning, 2017.

[27] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, 2013.

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

[29] Donghui Wang, Yanan Li, Yuetan Lin, and Yueting Zhuang. Relational knowledge transfer for zero-shot learning. In *AAAI*, 2016.

[30] Qian Wang and Ke Chen. Zero-shot visual recognition via bidirectional latent embedding. *CoRR*, abs/1607.02104, 2016.

[31] Yang Wang and Greg Mori. A discriminative latent model of object classes and attributes. In *ECCV*, 2010.

[32] Yongqin Xian and Akata et al. Latent embeddings for zero-shot classification. In *CVPR*, 2016.

[33] Felix X Yu, Liangliang Cao, Rogerio S Feris, John R Smith, and Shih-Fu Chang. Designing category-level attributes for discriminative visual recognition. In *CVPR*, 2013.

[34] Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via semantic similarity embedding. In *ICCV*, 2015.

[35] Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via joint latent similarity embedding. In *CVPR*, 2016.