Zero Shot Learning for Image Classification

Mansi Mane, Raajitha Gummadi, Dhanashree Balaram
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
mmane@andrew.cmu.edu, rgummadi@andrew.cmu.edu, dbalaram@andrew.cmu.edu

1 Introduction and Motivation

Human beings have the ability to recognize an object that they have not seen before, given a description of the characteristics of the said object. For example, consider a person that has never seen a zebra but has seen a horse. Given a short description of the features of a zebra as 'Something that looks like a horse and has black and white stripes' the person is capable of identifying a zebra. Zero shot learning (ZSL) aims at modeling this ability.

Zero shot learning (ZSL) is therefore an extension of supervised learning in cases such as classification problems, where there are no labeled examples available for few classes. Thus, it predicts classes that are not in the training data. For example, consider an image classification neural network trained on classes such as cat and car, but not on truck. Given an image of a truck during test time, this network should be able to predict truck as an unseen class and assign it to the class 'truck'. Our proposed method is an extension of an existing zero shot learning algorithm [1] to increase the accuracy of seen classes and unseen classes.

2 Existing methods

In today's world, we have an increasing number of datasets for problems like image classification in a multi-class setting. Algorithms in this setting usually have a final layer called a softmax layer which predicts the most likely class. A legitimate problem though, is that if we need to train a network to predict all existing classes in the world, we cannot handle the computing power required(enormous dataset and softmax layer). This is why it is believed that zero shot learning will solve the problem of classifying images that do not belong to the training dataset.

Zero shot learning has gained significant importance in the last decade, enforcing that learning objects that have not been seen before might become a necessity in the future. Some existing approaches to Zero shot learning that we surveyed have been summarized below:

Learning Intermediate Attribute classifier: These methods transfer target domain data (e.g. text explaining images) into source domain attribute space. [2] learns probabilistic attribute classifiers and makes class predictions by combining scores of those learned classifiers. In this method, attributes are typically the nameable properties of an object, like color, or the presence or absence of a body part. They propose a solution for learning with disjoint training and test classes by introducing a small set of high-level semantic attributes that can be specified either on a per-class or on a per-image level([2] uses Animal with Attribute dataset).Another method mentioned in [2] indirectly estimates attribute probabilities of an image by first predicting the probabilities of each training class and then multiplying by the class attribute components represented as a matrix.

Linear Embedding: These methods embed visual and text domain data into space characterized by Kronecker product of features in both domains. Linear classifiers are then trained on this product space. [3] [4] [5] use this approach. DEVISE[3] uses bi-linear compatibility function to associate visual and auxiliary information. Auxiliary information assists the visual information in selecting the correct class for classification. The paper Embarrassingly Simple Zero Shot Learning [6] uses the same objective function as[3] with the addition of regularization.

10707 Class Project (2017).

Non-Linear Embedding: This method of zero shot learning constructs product Kronecker feature space by applying non-linear mapping on the original features. Zero shot learning through cross modal transfer [1] implements this method. It first projects visual features (learned using auto-encoder) of the images in low dimensional space and then applies $\tanh$ non-linearity on it. These intermediate features are then projected into low dimensional space of word vectors again. Note that [7] decribes image and text embeddings are projections from the space of pixels, or the space of text, to a new space where nearest neighbors are semantically related. In semantic label embedding, image and label embeddings are jointly trained so that semantic information is shared between modalities. For example, an image of a tiger could be embedded in a space where it is near the label "tiger", while the label "tiger" would itself be near the label "lion".Thus, the projection weights are trained to map visual features to respective word vectors of that class. While predicting class of the new image, the class of word vector nearest to the projected image vector is declared as object.[1] poses as a baseline for our project. Another paper that implements non-linear embedding method is [8] . This method learns multiple embeddings(parameters of the individual linear components) of the model that maximize the compatibility between the input embedding (image, text, space) and the output embedding (label space) of all training examples. The different parameter embeddings learnt may capture different visual characteristics of objects, i.e. color, beak shape etc. and allow distribution of the complexity among them, enabling the model to do better classification.

Hybrid Models: Semantic Similarity Embedding [9], Convex Combination of Semantic Embeddings [10] and Synthesized Classifiers [11] express images and semantic class embeddings as a mixture of seen class proportions, hence we group them as hybrid models.We do not go into details about hybrid models, since our proposed approach deals with non linear embeddings.

3 Problem statement

We aim to build upon an existing method Socher et al [1] in order to achieve higher classification accuracy for ZSL. As mentioned in section 2, in order to learn semantic relationships and class memberships of images, we project the image feature vectors into a d-dimensional, semantic (similar in context) word space F. Thus, images are mapped to be close to semantic word vectors corresponding to their classes. This mapping is achieved by a single two-layer neural network. i.e, a weight parameter map common to all classes. The model in [1] shows decrease in accuracy of unseen classes with the increase in similar semantic vectors, also known as distractor words. As an extension to [1], we aim to map the images into the word spaces using a parameter exclusive to each of the seen classes. This maps the images of seen classes closer to their corresponding semantic word vector and thereby improving the accuracy of seen classes. Additionally, the mapping of unseen classes is refined by the strong mapping produced by each of seen class parameters, (elaborated in section 4) thus improving the accuracy of unseen classification.

4 Proposed Approach

The baseline model [1], trains a neural network with two layers to learn a 'single' mapping that projects an input image onto a word embedding space. Since every object is unique in its visual features, the mapping of an object to its corresponding word vector is also unique. For example, an image of a 'dog' is mapped to the word vector 'dog' differently than how an image of an 'airplane' is mapped to the word vector of 'airplane'. Hence, we hypothesize that learning this unique mapping between a class and its respective word vectors increases the seen accuracy. Additionally, for unseen classes, consider a seen class 'dog' that is closest to the unseen class of the input image say 'cat' when compared to other seen classes in the embedding space such as 'airplane' . Then, the word vector obtained for cat's image using dog's mapping is closer to class 'dog' than the word vector obtained for the image of cat by other seen classes such as 'airplane'. According to this hypothesis, it is probable that an improved accuracy for unseen classes can be achieved.

The proposed approach consists of extracting features from images and building the word vector space. This is followed by training the network on seen classes. Finally testing is performed on unseen and seen classes as described below.

Visual Features and Word Representation

We use unsupervised soft-threshold autoencoder provided by Coates et al. [12] to extract image features from raw pixels of the image as shown in figure 1(b). Each image is henceforth represented by a vector $x \in \mathbb{R}^I$.

Word vectors provided by Huang et al [13], thereby placing similar words in meaning like "happy" and "joyful" closer to each other than "happy" and "sad". $W = W_s \cup W_u$ are the set of word vectors representing both seen and unseen visual classes, respectively.

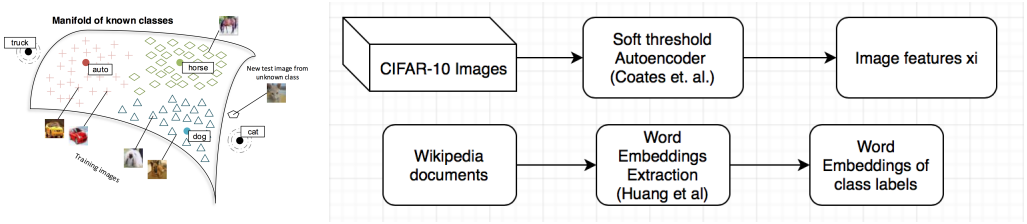

Figure 1: (a) Manifold diagram for seen classes from [1], (b) Visual features and word representations

Training

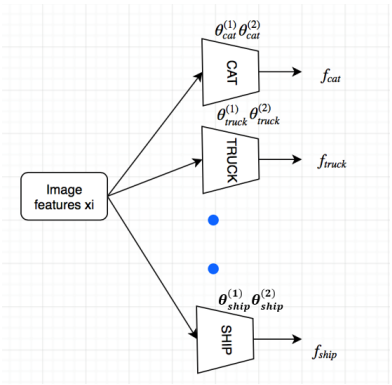

Figure 2: Learning visual word embeddings

Consider a set of classes Y for both training and testing. Seen classes are defined as Y_s and unseen classes are defined as Y_u. Consider that training occurs on 8 seen classes. Each class has a unique neural network that learns to project images onto word embeddings of the respective class for all seen images as shown in Figure 2. The process of training a single neural network is summarized below.

A training image feature vector belonging to any seen class, $x^{(i)} \in X_s$ is input to the neural network corresponding to class y. The network learns a word embedding for an input image using parameters $\theta_y^{(2)}, \theta_y^{(1)}$. To train these parameters, we minimize the following objective function for every seen class:

$$J(\Theta_y) = t_y \sum_{x^{(i)} \in X_s} ||w_y - \theta_y^{(2)} g(\theta_y^{(1)} x^{(i)})||^2 - \lambda(1 - t_y) \sum_{x(i) \in X_s} ||w_y - \theta_y^{(2)} g(\theta_y^{(1)} x^{(i)})||^2 \quad (1)$$

where $\theta_y^{(1)} \in \mathbb{R}^{h \times I}$, $\theta_y^{(2)} \in \mathbb{R}^{d \times h}$ and f is the $\tanh$ non-linearity and $\Theta_y \in (\theta_y^{(1)}, \theta_y^{(2)})$. w_y is the fixed word embedding of the class label $y \in Y_s$ obtained from Huang et al. t_y is 1 if input belongs to class $y \in Y_s$ and is 0 if input belongs to a different seen class. λ is the regularization term that penalizes images that are not in class y to increase distance between projections of the class y and other seen class word vectors. We train $J(\Theta_y)$ for every seen class. The process is repeated for each of the seen classes. Thus we learn parameters $\theta^{(1)}$ and $\theta^{(2)}$ specific to each class as shown in Figure 2.

The average distance between the predicted embedding vector of images belonging to seen class y and the word embedding vector for class y engenders the average variance for class y. This variance is later used in testing to identify whether input test image belongs to seen or unseen class for Gaussian discrimination.

Testing

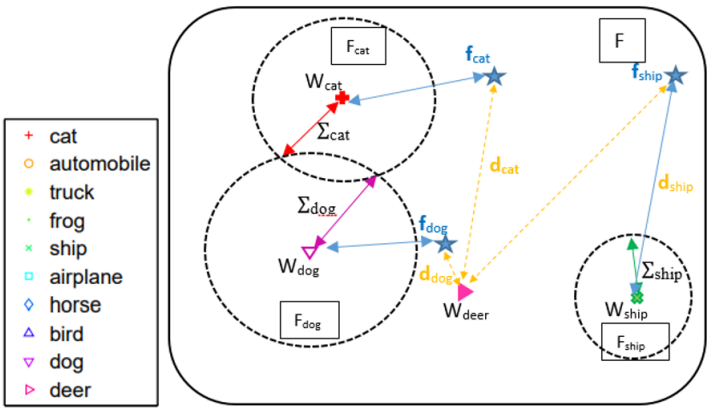

Figure 3: Visualization of semantic word space while predicting unseen classes, where V, the novelty variable is unseen, i.e V = u, F is the d-dimensional semantic word space, $F_{cat}, F_{dog}, F_{ship}$ are all the semantic vectors of seen classes cat, dog and ship. $\Sigma_{cat}, \Sigma_{dog}, \Sigma_{ship}$ are variance of the classes and $W_{cat}, W_{dog}, W_{ship}$ are the mean word vectors provided by Huang et al. $f_{cat}, f_{dog}, f_{ship}$ are the projections of a new input image (from unseen class) into the word space F. $d_{cat}, d_{dog}, d_{ship}$ are the distances of projected word vectors from the projections of each neural network corresponding to classes cat, dog and ship to the mean word vector of the unseen classes(denoted as d_u) (here only a single unseen class is shown for clarity)

Inference Algorithm

1. Check if the given image is from a seen or an unseen class using the following equation.

$$P(V = s|f, X_s, W_c, \theta_c) := \mathbf{1}\forall y \in Y_s : P(f|F_y, w_{cy}) > T_y \tag{2}$$

2. If $V = u$,
   - Calculate distance d_y between projected image embedding and all unseen class word embeddings.
   - Predict class label with maximum P_y i.e. find nearest word vector W to f, using Gaussian Discrimination.
3. If $V = s$, classify using Softmax for seen classes.

Given a test image feature vector, we first predict whether an image is from a seen class or an unseen class. We use binary random variable V, where $V \in (u, s)$ is used to predict seen (s) or unseen (u) class. We define a hyper parameter T_y as threshold for all classes. If $P(f|F_y, w_y) > T_y$ for any class, then the image is classified as 'seen' else, the image is classified as 'unseen'. For example in figure 3, if the image is from a zero shot category (deer), it will be projected much further from word embeddings of seen classes (cat, dog and ship). Thus, $P(f|F_y, w_y)$ will be smaller than threshold for seen classes. T_y is calculated as sum of Gaussian log probabilities with mean as word embeddings obtained from Huang et al [13] and variance that is calculated for each class during training. For example, if the image is an unseen image it lies outside the Gaussian spread of every seen class centered at the its mean (word embedding). Therefore, sum of log probabilities (probability of image belonging to seen class) of all seen classes will be less than the threshold. Therefore, we conclude that the image does not belong to seen category. Alternatively, we can also learn class-specific Euclidean distance cutoffs. If input image is mapped to semantic word space at an Euclidean distance more than learned cutoffs of every seen class, we classify image as unseen. Cutoff for each seen class is calculated as the value corresponding to highest accuracy while identifying respective class images.

In order to identify the unseen class, first a seen class that maps the unseen image closest to its mean is chosen. Then, we find an unseen class closest to that seen class with smallest d_u. Where, d_u is the distance of projected word vectors obtained using each seen class theta to mean of an unseen class. Thus use Gaussian Discrimination to identify unseen class label as shown in figure 3.

One of the limitations of having unique mapping for every seen class is that the computation time and complexity increases in proportion to the number of seen classes.

5 Datasets

The visual models are trained using the CIFAR-10 data. It is a subset of the 80 million tiny images dataset that was designed and created by the Canadian Institute for Advanced Research (CIFAR). It consists of 60,000 32x32 color images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images in the official data. The number of training images per class is 5000. The label classes in the dataset include airplane automobile, bird, cat, deer, dog, frog, horse, ship, truck. The classes are completely mutually exclusive. There is no overlap between automobiles and trucks[14]. The text model is trained on documents extracted from wikipedia.org as mentioned in Huang et al.

6 Experiments

6.1 Results of reimplementation of baseline model:

We have used [1] as our baseline model for zero shot learning. We reimplemented [15] and obtained the results as shown in figure 4.Our reimplementation generated a model that follows the same behavior as that of the original Gaussian model in our baseline paper[1].

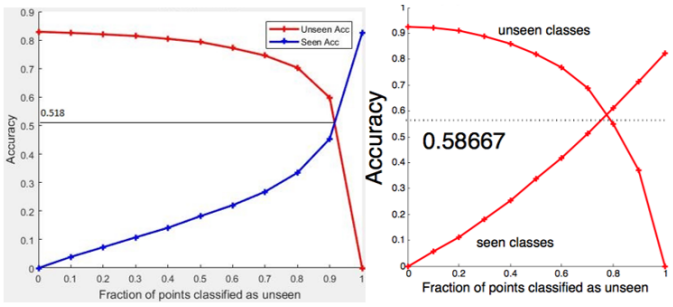

Figure 4: (a) Result from re-implementation of [1], (b) Original paper Results for Gaussian model

The evaluation metric for our model is classification accuracy. We achieved an average classification accuracy of 0.518 as seen in figure 4a,as compared to 0.58667 of our baseline model. As we increase the threshold T_y, more images are classified as 'seen', therefore, seen accuracy increases. The baseline paper's [1] results are given in figure 4b. The t-distributed stochastic neighbor embedding (t-SNE) for the seen training data and testing data is shown in figure 5

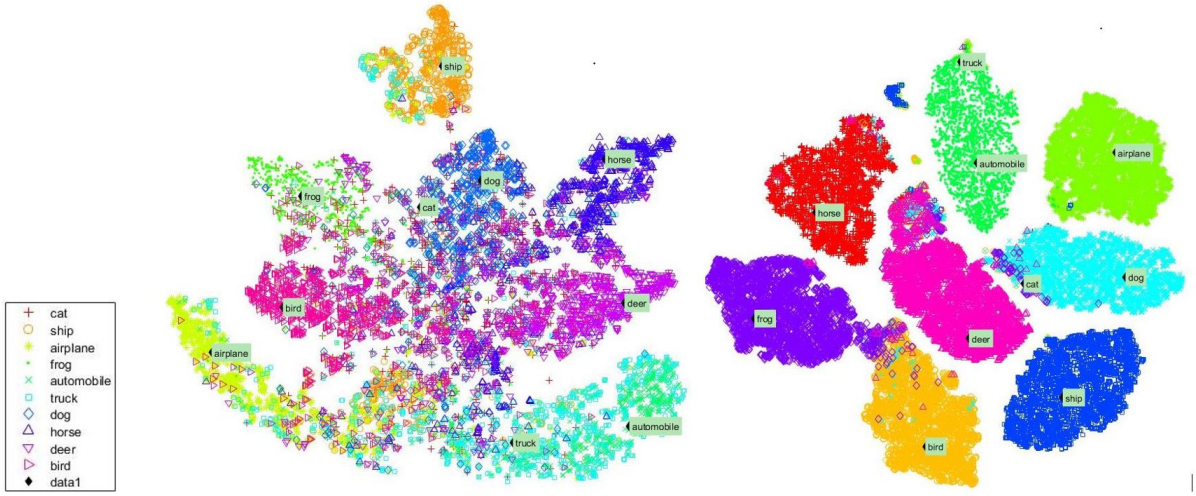

Figure 5: (a) t-SNE plot for test data,(b) t-SNE plot for train data (Seen)

6.2 Results of proposed model

The best model obtained using the proposed approach, shows a model accuracy of 36.63% for a hyper-parameter combination of Learning rate = 0.01, penalty λ = 0.0001, number of hidden units = 200, sparsity= 0.035 and dropout = 0.25. Figure 6 shows a plot of seen and unseen accuracies vs fraction of points classified as unseen. It can be inferred from this graph that the proposed approach has lower accuracy on unseen classification.

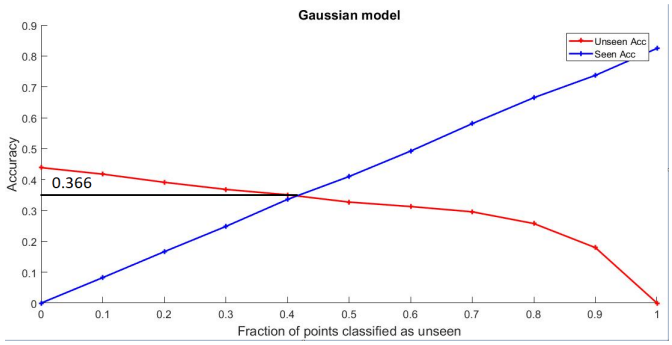

Figure 6: Classification Accuracy intersection point for seen and unseen classification

Figure 7 shows t-SNE visualization of the best model. Mappings of the entire dataset are observed to be coagulating around the word embedding of the corresponding seen class label. Since each mapping (θ) is unique, it learns a unique pattern. This is explained in detail in section 7.

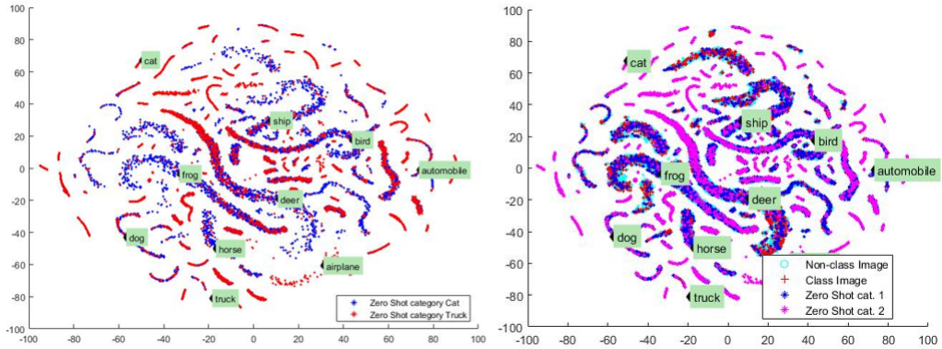

Figure 7: (a) t-SNE plot for unseen classes, (b) t-SNE plot for all classes

Refer to Appendix for confusion matrices.

6.3 Discussion and Analysis

The behavior of the model was analyzed by running experiments with different hyper parameters such as penalty term λ, from equation1, dropout factor, sparsity parameter, and number of hidden units of the neural network. Additionally, alternative approach of calculating threshold as mentioned in section 4 is also tested. The observations are summarized below.

Experiments with hyper-parameter λ: For CIFAR10 dataset, 8 classes are considered as seen classes. During training, mapping is learned on unbalanced data i.e. $1/8^{th}$ of the images belong that class while $7/8^{th}$ images do not belong to that class. The network must learn to reject(map father from mean) non-class images more than map respective class images. In order to incorporate this concept, λ penalizes the images that do not belong to the class under consideration and maps images father from its mean. i.e, λ in equation 1 is experimented with 0.1,0.01,0.0001 and 0 values. The best accuracy was obtained for a λ of 0.0001. λ = 0, implies only learning mapping from respective class images, the network does not see other class images at all. Increase in λ, minimizes the cost function and thus network does not penalize images belonging to other classes.

Search for other hyper parameters: Different values of dropout such as 0,0.25,0.5 and 0.75 were tested by maintaining the learning rate at 0.01, the penalty value as 0.0001, sparsity as 0.035 and number of hidden units as 200 as given in the table 1. The best average model accuracy(36.63%) was achieved for a dropout fraction of 0.25. Learning rate of 0.1, 0.01 and 0.001 were tested, 0.1 renders an unstable system while 0.001 results in slow conversion rates thus increasing the computation time. Therefore learning rate of 0.01 is used in all experiments. Furthermore, number of hidden units were varied between 100, 200 and 300. While 100 hidden units shows less accuracy due to under-fitting, 300 units over-fit the seen classes and result in less model accuracy. Thus 200 units were chosen for optimal performance. Additionally, experiments were conducted with different values of the sparsity parameter.Results of the hyper-parameter search are summarized in Table 1. In Table 1, the last column, 'Model Accuracy' is calculated by finding the intersection point of seen and unseen accuracy curves plotted against fraction of points classified as unseen.(as seen in figure 6). The other columns that have accuracy values represent the maximum accuracy that was achieved for seen and unseen categories.

Table 1: Results of experimentation for proposed approach using log probability thresholding

	λ (penalty values for other seen classes)	Sparsity parameter	Number of hidden units	Drop out fraction	Maximum seen class Accuracy (in %)	Maximum zero shot Accuracy	Model Accuracy
Experiment for lambda	0.0001	0.035	200	0	82.45%	45.75%	34.93%
	0.01	0.035	200	0	82.45%	42.4%	34.08%
	0.1*	0.035	200	0	-	-	-
	0	0.035	200	0	82.45%	40.30%	36.28%
Experiments for Dropout fraction	0.0001	0.035	200	0.5	82.45%	43.85%	36.58%
	0.0001	0.035	200	0.25	82.45%	43.93%	36.63%
	0.0001	0.035	200	0.75	82.45%	43.85%	36.5%
Experiments for Number of hidden units	0.0001	0.035	100	0.25	82.40%	38.25%	33.76%
	0.0001	0.035	300	0.25	82.45%	39.5%	33.66%
Experiments for Sparsity parameter	0.0001	0.02	200	0	82.44%	41.9%	34.2%
	0.0001	0.05	200	0	82.45%	45.1%	34.8%
Experiments for different parameter values	0.0001	0.035	100	0.25	82.45%	43.1%	34.98%
	0.0001	0.05	100	0.25	82.45%	34.05%	33.28%

*Model failed to converge due to overflow.

Table 2: Comparison of Accuracies for proposed approach and baseline model

Model	Cut-off method	Model Unseen Accuracy	Model Seen Accuracy	Model Accuracy
Baseline model	Log probability threshold	50.01%	60.12%	50.3%
	Euclidean threshold	33.8%	64.60%	58.44%
Proposed Approach	Log probability threshold	39.55%	41.9%	31.3%
	Euclidean threshold	39.55%	75.65%	68.43%

Table 2 shows a comparison of classification accuracies between the baseline model and the proposed model for two thresholding methods as described in section 4. It is observed that Euclidean thresholding performs better than log probability thresholding (LPT) for seen accuracy in both models. For the baseline model, log probability thresholding performs worse than euclidean thresholding in the case of unseen accuracy in the baseline approach but gives a similar accuracy for the proposed model. This is because the classification of seen classes using Euclidean thresholding works better with tailored mapping parameters over single mapping. An example of how Euclidean thresholds are obtained is depicted in figure 8. For every class, cut-off is increased until accuracy saturates at high value. At the highest accuracy, the corresponding cut-off is used as the euclidean threshold. Note that Euclidean thresholds are calculated for every class, i.e., thresholds are class specific and thus seen classes are aptly classified. This increases the seen class accuracy and therefore the overall

model accuracy. Log-probability threshold on the other hand is obtained by calculating cutoffs by marginalizing the log-probabilities of all classes.

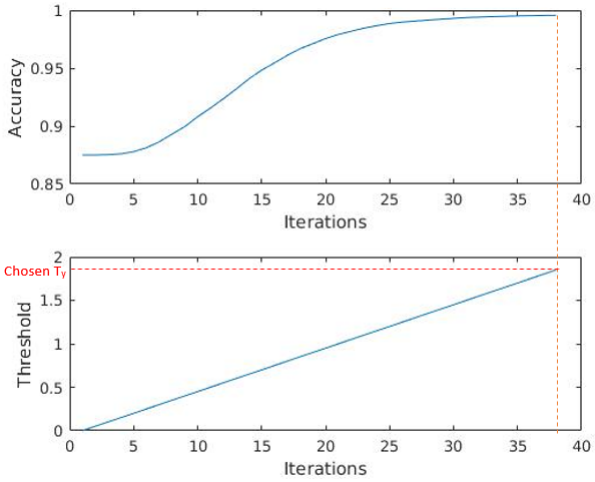

Figure 8: Example of how an Euclidean threshold is calculated for a given class

7 Learnings and Future Work

Following multiple experiments as summarized in section 6, it can be concluded that the proposed model with sum of log probabilities as threshold fails to perform when compared to the baseline model [1]. One reason for this is that although all neural network parameters(θ) map to the word embedding space uniquely, they map one input image to different manifolds (i.e, non-linear mapping using different theta). i.e, each theta maps to a different embedding space. This behavior is depicted for a reduced dataset for clarity in figure 9 where mappings of the input images seem to coagulate around the word embedding of each seen class. This is a repercussion of having completely independent mapping. Thus, in the future, linking the theta parameters by defining common loss for all, could improve the unseen class accuracv.

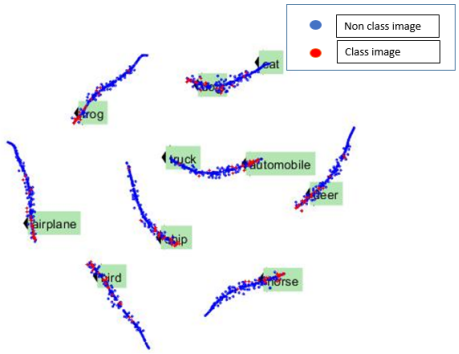

Figure 9: Semantic word space(t-NSE) for reduced data

Proposed model has better accuracy when implemented with euclidean cutoff. This could be further explored. One other limitation to this approach is that the theta parameter of each seen class is trained on the whole dataset. Thus leaving only a small portion of true values for the objective function to be trained on for that seen class. One of the solutions to this issue could be to try and split data such that each of the neural networks get more true values of that particular seen class than images from other seen classes during training as inputs. Furthermore, log probabilities are currently calculated as isometric Gaussians to avoid overfitting. Seen accuracy could be increased by learning non-isometric Gaussians if there is enough training data.Finally, number of parameters required increases in proportion to the seen classes. This highly increases computation time and complexity. Although this does not seem like a drastic effect for CIFAR10 dataset with only 8 seen classes, as the number of seen categories increase, the number mapping parameters increase and thus the computation complexity also increases proportionally.

References

[1] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 935–943. Curran Associates, Inc., 2013.

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

[3] A. Frome, G. Corrado, J. Shlens, S. Bengio, J. Dean, M.A. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *Annual Conference on Neural Information Processing Systems (NIPS)*, 2013.

[4] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label embedding for image classification. volume 38. IEEE, 2016.

[5] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[6] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2152–2161, Lille, France, 07–09 Jul 2015. PMLR.

[7] Jimmy Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. Predicting deep zero-shot convolutional neural networks using textual descriptions. *CoRR*, abs/1506.00511, 2015.

[8] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh N. Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. volume abs/1603.08895, 2016.

[9] Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via semantic similarity embedding. *CoRR*, abs/1509.04767, 2015.

[10] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *CoRR*, abs/1312.5650, 2013.

[11] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. *CoRR*, abs/1603.00550, 2016.

[12] Adam Coates and Andrew Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 921–928, New York, NY, USA, 2011. ACM.

[13] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[14] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[15] Milind Ganjoo. zslearning. `https://github.com/mganjoo/zslearning`, 2013.

Appendix

Confusion matrix The confusion matrices for the best performing proposed model are given below.

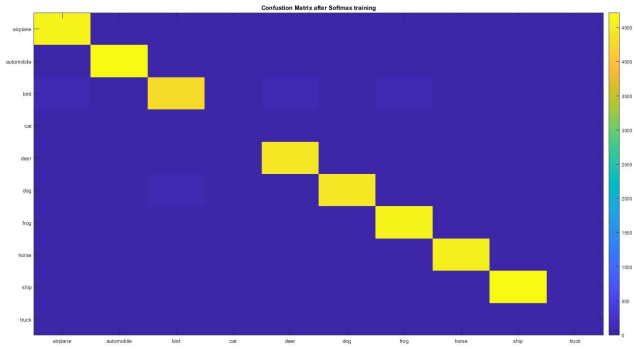

Figure 10: Confusion matrix of seen classes

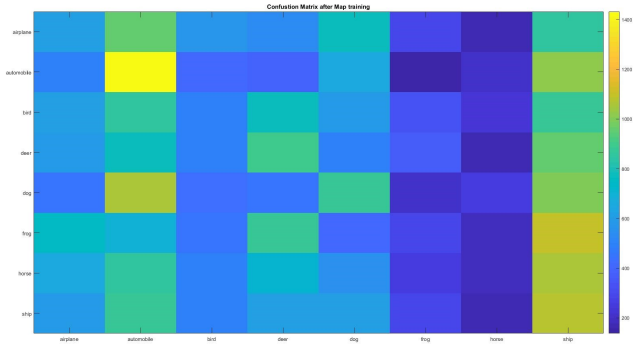

Figure 11: confusion matrix of unseen classes