
Image Classification Via Selective Convex Combination of Semantics Embeddings

Kritkorn Karntikoon
Korrawat Pruegsanusak
Suchan Vivatsethachai

KRITKORN@MIT.EDU
KORRAWAT@MIT.EDU
SUCHANV@MIT.EDU

Abstract

Image classification is one of the most classic classification that has been studied for a long time. While there are more and more image-label corpus, there are still a lots of images unseen or unlabeled. Giving that, it would be very beneficial if we could find a way to predict images which labels have been seen before. This is where zero-shot learning plays an important role. This paper particularly concentrates on the zero-shot learning that makes use of information from word embedding techniques in natural language processing. One of the paper, ConSE, proposes an ingenious way to get an information from word embedding space. However, we notice that there are still rooms to be improved in the original model. This paper will propose two models, which are based on ConSE, and provide insightful analysis on the performance of both models, using ConSE as a baseline.

1. Introduction

One of the most challenging image classification tasks is n -way image label classification with low resource training data. Specifically, we want to predict a label of a particular image from the pool of n labels the model has not seen before in the training. To achieve such prediction, we need to incorporate information from other sources. Such model is an example of a

zero-shot learning, a model that predicts result in the space that it has not been trained before. ,

To infer information about the labels that the model has never seen, we make use of word embedding space and relationship between embedding vectors. Several papers [2, 3, 6] have tried to connect relationship between image space and word embedding space to get a more robust prediction on an unseen image.

One of the models, ConSE [4], use a simple but ingenious technique to connect two spaces together. They use CNN model trained on training data, testing images with label never seen in the training, and word embedding model which includes all the label from training and testing. For any given testing image, they first find top t highest predicted labels from CNN. Those labels are from training dataset, so it is not the true label of the given image. They instead find a convex combination, which we call *synthetic* word embedding, of the word embeddings of those t labels based on the predicted probability they get. Finally, the model will predict the label of which the word embedding has closest distance to the computed convex combination.

In this paper, we try to improve upon zero shot learning introduced in ConSE [4] to yield a better result. Instead of using naively constructed *synthetic* word embedding, we propose a revised *synthetic* word embedding which is constructed based on the synthetic vector itself. Specifically, in one proposed model, we get rid of some of the predicted labels that are "too far" from the original *synthetic* word embedding. In the other model, we get rid of the predicted labels that are "too similar" to labels that have higher probability. Doing so, we hope that we discover a model that are more insightful about the relationship of word embed-

dings, resulting in an increase of prediction accuracy.

To get more insight about the important of convex combination of word embedding, we demonstrate the effect of the number of word embeddings (T) to be included in *synthetic* word embedding construction. We also explore the efficiency from different word embedding models with different embedding dimension and different size of corpus. This will give us some insight about how to choose word embedding model that allows zero-shot learning to yield a better result. Furthermore, to investigate the behavior of our proposed models deeper, we make the following observation:

- we look into some specific image that works better in our proposed model than in ConSE model, and demonstrate how it gets better.
- we show the performance of our proposed model on each different high-level category of testing images

1.1. Contributions

Our main contribution is Threshold and Deviated model. Starting from ConSE Model, we create another step to remove outliers with respect to the synthetic vector, then we recompute and return the new result. We refer to this model as “Threshold model.”

After observing results from Threshold model, we propose another model by including labels that are substantially different from the synthetic vector, hoping that these labels contain information that will improve the model. This model is referred as “Deviated model.”

We also made our code available at <https://github.com/korrawat/zeroshot>.

2. Problem Statement

We will present our model in a more mathematical rigorous way. $\mathcal{X}_0, \mathcal{X}_1$ be the set of all images in training dataset and testing dataset respectively, all with the same dimension and number of pixels. $\mathcal{Y}_0 = \{\ell_1, \ell_2, \dots, \ell_a\}$ is the pool of all labels of the training images, while $\mathcal{Y}_1 = \{\ell_{a+1}, \dots, \ell_{a+b}\}$ is the pool of all labels of the testing images. For each i , word label ℓ_i is English words describing the image. It is important to note that $\mathcal{Y}_0 \cap \mathcal{Y}_1 = \emptyset$. The training datasets

consist of $(x_i, y_i)_{i=1}^N$ where $x_i \in \mathcal{X}_0$ and $y_i \in \mathcal{Y}_0$. Our goal is to predict the label of image $x' \in \mathcal{X}_1$ from the pool of labels $\mathcal{Y}_1 = \{\ell_{a+1}, \dots, \ell_{a+b}\}$, which we have not seen before. This won't be possible unless we have connection between labels in \mathcal{Y}_0 and labels in \mathcal{Y}_1 . Particularly, we incorporate a semantic word embedding function $\omega : \mathcal{Y}_0 \cup \mathcal{Y}_1 \rightarrow \mathcal{S}$ where \mathcal{S} is a set of q -dimensional word embedding vector $\{s \mid s \in \mathbb{R}^q\}$.

3. Models

3.1. ConSE Model

ConSE proposes the following model as shown in Figure 3.1. First, they use CNN model trained on $(x_i, y_i)_{i=1}^N$. Using this CNN model on testing image x' gives a probability distribution (p_1, p_2, \dots, p_m) corresponded to the training labels $(\ell_1, \ell_2, \dots, \ell_m)$. Lastly, they find a convex combination (a synthetic vector) of the first T highest probability labels based on their predicted probability. Let $p(x', t)$ be the t^{th} highest probability among (p_1, p_2, \dots, p_m) and $\ell(x', t)$ be the corresponding label. We use only T highest probability label to construct a synthetic word embedding

$$syn(x') = \sum_{t=1}^T \tilde{p}(x', t) \omega(\ell(x', t))$$

where $\tilde{p}(x', t) = p(x', t) / \sum_{t'=1}^T p(x', t')$, a normalized probability. Finally, they predict label

$$\ell^* = \operatorname{argmax}_{\ell \in \mathcal{Y}_1} d(syn(x'), \ell)$$

when $d(u, v)$ is a cosine similarity between vector u and v

3.2. Threshold Model

When we investigate the probability distribution from CNN for some image, we find that among the top T highest probability, some are outrightly wrong. So instead of naively combining the top T highest probability, we will screen some outlier out first. Specifically, for testing image $x' \in \mathcal{X}_1$, let the initial probability distribution from CNN be (p_1, p_2, \dots, p_m) . We first create a temporary synthetic vector

$$tem = \sum_{t=1}^T p(x', t) \omega(\ell(x', t))$$

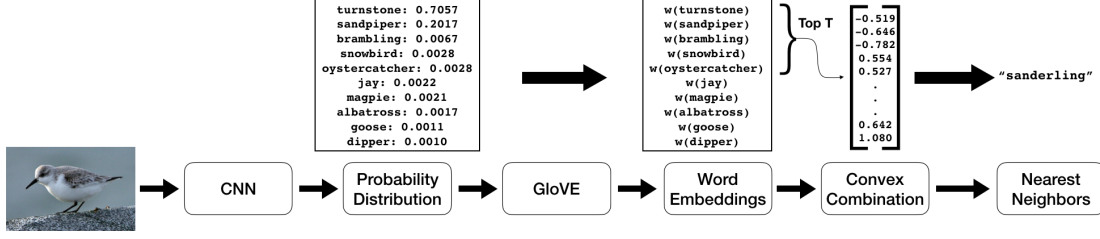


Figure 1. Procedure of ConSE Model. The input as a picture is parsed into the model. The model then uses CNN to compute the probability distribution of categories that are similar to the input. Then, we convert each category into word embedding and compute its weighted average. Finally, we return the label that is nearest to this average.

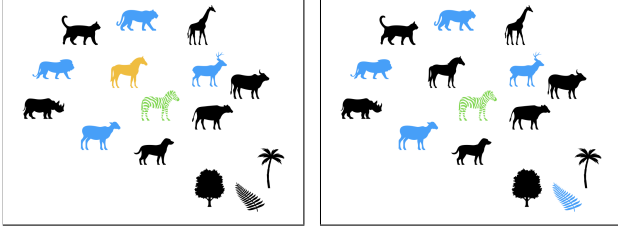


Figure 2. Diagram illustrates the comparison between baseline and deviated model. The blue items represent the labels used to predict the result. The left hand side figure shows wrong predicted label (represented by yellow item) in the baseline model that can be improved by deviated model. The right hand side figure shows that deviated model will include substantial difference label and might lead to actual label (represented by green item)

, then we find a cosine similarity between tem and each of $\omega(\ell_i)$ for $\ell_i \in \mathcal{V}_0$. Let the resulting similarities be (d_1, d_2, \dots, d_m) respectively. With threshold τ , we screen out all label ℓ_i with $d_i < \tau$. We are left with label $(\ell'_1, \ell'_2, \dots, \ell'_{m'})$, with corresponding probability $(p'_1, p'_2, \dots, p'_{m'})$. Again, we define $p'(x', t)$ and $\ell'(x', t)$ be the t^{th} highest probability among $(p'_1, p'_2, \dots, p'_{m'})$. Finally, we compute the final synthetic word embedding.

3.3. Deviated Model

We also have another assumption that a substantial different labels need to contribute to the construction of a good *synthetic* word embedding. However, some of those labels may get too small probability that they are not included in the top highest T probability. We propose a new model that allows labels, which have low probability but contains crucial information, to take part in the resulted *synthetic* word embedding as seen in Figure 2. Assume here that we still want the

synthetic word embedding to be constructed from exactly T word embeddings. Let $T = 2r$. First, we include the r highest probability labels. Then for the next $2r$ highest probability labels, we will choose only r to be included in the construction. We make selection based on the distance from the temporary word embedding constructed with only the first r labels. Particularly, we first construct

$$tem(x') = \sum_{t=1}^r p(x', t) \omega(\ell(x', t)).$$

Then we find a cosine similarity between $tem(x')$ and each of $\omega(\ell)$ for $\ell \in \{\ell(x', r+1), \ell(x', r+2), \dots, \ell(x', 3r)\}$ to get $d_{r+1}, d_{r+2}, \dots, d_{3r}$ respectively. Let $d'_{r+1}, d'_{r+2}, \dots, d'_{3r}$ be sorted increasing order with corresponding labels $\ell'(x', r+1), \ell'(x', r+2), \dots, \ell'(x', 3r)$ and corresponding probability $p'(x', r+1), p'(x', r+2), \dots, p'(x', 3r)$, and . We then include the labels $\ell'(x', r+1), \ell'(x', r+2), \dots, \ell'(x', 2r)$ in *synthetic* word embedding construction. In other words, we include only labels that are *more different* from the first r labels so that the *synthetic* word embedding can gain extra information. The final word embedding is

$$syn(x') =$$

$$\frac{1}{M} \left(\sum_{t=1}^r p(x', t) \omega(\ell(x', t)) + \sum_{t=r+1}^{2r} p'(x', t) \omega(\ell'(x', t)) \right)$$

when M is a normalized factor: $M = \sum_{t=1}^r p(x', t) + \sum_{t=r+1}^{2r} p'(x', t)$.

4. Implementation Details

Our models can be used with any Convolutional Neural Network architecture and any set of word vec-

tor representations. The CNN of our choice is the famous GoogLeNet [7], which was the winner of ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC 2014) with the top-5 classification error of 6.67%. Even though there have been some researches that improve on the original GoogLeNet in the past few years, we think that this is a proper model for our project as it balances between performance and running time. While more recent models perform better on the regular image classification task, some need much more computational resources and it is not evident that they would perform better for our zero-shot image classification task.

The word embeddings we choose is Global Vectors for Word Representation, or GloVe [5]. As opposed to other word embedding models, GloVe provides several word embeddings that are pre-trained with different sizes of text corpus and different dimension of vector representations.¹ This allows us to explore the effects of these parameters on the performance.

The training and testing data is taken from ImageNet [1], an image database organized according to the WordNet tree (a hierarchy semantic tree). Each class of images is called a synset, with which is described by multiple labels associated. We call those labels describing each synset a “synset’s description”. It is important to note that some labels are not unigram, and some synset’s description only contains label that is not unigram. However, since most word embedding models are trained on a bag of unigram words, we will only consider synset, which description contains at least one unigram label. We do so because practically we don’t want to retrain word embedding every time we encounter new bigram label. This is essentially different from ConSE’s implementation that retrains their word embedding with a new bag of words that include bigram labels from the training and testing synsets.

We use the CNN that is pre-trained with ILSVRC 2012 image classification dataset, which has $a = 1000$ labels². For convenience, we will refer to this as the “1K synsets”. For each synset in 1K synset, we

choose only one unigram label from its description to represent the entire synset. This label will be the label that get converted to word embedding when we are constructing *synthetic* word embedding.

For testing synsets, we choose only those that are semantically close to the 1K synsets. Specifically, we choose only synsets in WordNet tree that are within 2 steps away from each synset in 1K synsets. Note that we must not include the synset in the training 1K synsets. We refer to these testing synsets as “2-hops” synsets. Lastly, to balance the testing images, we only include 200 images from each of the synsets in 2-hops. We disregard any synsets contain less than 200 images. We end up with 295200 testing images consisting of 1476 final synsets, each of which contains 200 images. The data can be downloaded from³

We implement mostly everything in Python. We also use Amazon Web Services to get an access to GPU and multiple CPUs used to run the implementation.

5. Results

We present the performances of our models with several sets of parameters. First, we introduce the evaluation metric and its calculation steps. Then, we compare the results of the baseline model (ConSE), the Threshold Model, and the Deviated Model. Lastly, we analyze the model performances by word categories.

5.1. Evaluation Metric

In this subsection, we analyze the performances of our models which is measured by following steps

1. For each image, find if the true label is in the top k predictions
2. Count how many images is in each class get the correct label in top k predictions, then divide that by number of images in the class
3. Average the metric for chosen classes

This result will be referred in this paper as “Hit@ k ”.

¹Pre-trained GloVe is available at <https://nlp.stanford.edu/projects/glove>.

²Pre-trained CNN in TensorFlow is available at <https://github.com/tensorflow/models/tree/master/research/slim>.

³The data can be downloaded from <http://www.image-net.org/>.

Table 1. Effects of word embeddings on performance of the baseline model

Model	Precision hit@k (%)				
	1	2	5	10	20
6B 50d	3.8	5.6	9.2	13.3	18.5
6B 100d	4.3	6.6	10.6	15.0	20.7
6B 300d	5.4	8.3	13.9	19.6	26.5
42B 300d	5.7	8.8	14.5	20.4	27.4
840B 300d	6.8	10.7	18.3	25.9	34.5

5.2. Effects of word embeddings on the performance

We explore how using different word embeddings affects the performance of the baseline ConSE model.

From Table 1, we can see that as the dimension is increasing, the performance is better. This is quite different from what we expected before we did the experiment. At first, we expected that if the dimension is large, the word embedding space will be sparse. So, the predicted word embedding vectors will have less chance to encounter with any word in the space and so is the correct word as well.

However, when we sample some words that give good accuracies, we can see that the sparsity of large dimensional word embedding spaces will also separate two unrelated words further (the distance is computed by cosine similarity). So, the chance that predicted word embedding vectors will choose wrong words given that it is predicted in the correct region will also decrease.

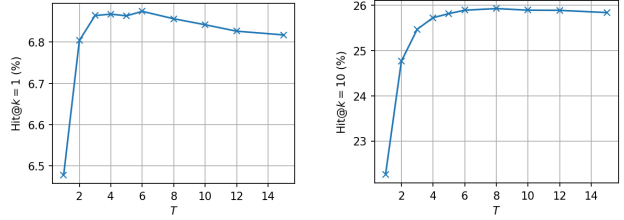
Since the “840B 300d” version of GloVe provides the best performance, the results presented in subsequent sections will be based on this set of word vectors. We choose this to be the **baseline model**, because when starting with higher accuracy, we should be able to observe larger change when we adjust other parameters.

5.3. Importance of convex combination

We vary T , the number of CNN predictions to perform convex combination, and observe the performance. From Figure 3, we can see that using $T \geq 2$ significantly improves the performance over just $T = 1$ for both top-1 and top-10 predictions. This indicates the importance of using convex combination, as opposed

to only using the top prediction from the CNN to find the nearest neighbors in word embedding space.

However, as we increase T from 6, the top-10 hit rate for the baseline model no longer improves, and the top-1 hit rate even starts to decline. This shows that while using more than one prediction to find a convex combination of word vectors is helpful, combining too many word vectors may be ineffective. This may be because the CNN predictions with lower probabilities include irrelevant words, thus deviating the synthetic vector away from the true label. This experiment inspire us to find other models that select only some of the top T CNN predictions to combine into the synthetic vector.

Figure 3. Effects of T on the performance of the baseline model.

5.4. Threshold Model

Since the similarity score is cosine, τ can vary from -1.0 to 1.0 . We do not consider the case when τ is too high ($\tau > 0.6$ in practice), because this will most likely not reduce the number of involved labels anymore.

From Figure 4, we can see that as the value of threshold is increasing, the Hit rate seems to remain unchanged until one point, at $\tau = 0.2$, where it drops significantly. One suspect is that when t is large, the algorithm starts to remove the essential label from being used and result in an inaccurate prediction from partial components. In general, we can conclude that adjusting threshold will not significantly improve our model performance because the baseline ConSE model is equivalent to $\tau = -1.0$.

However, we empirically observe some examples as shown in Table 3 and see that there is an improvement in rank of actual label. As shown in *shoe* example, the threshold, $t = 0.0$, removes “shoe shop” and “plane” which are not related to “shoe” in terms of meaning.

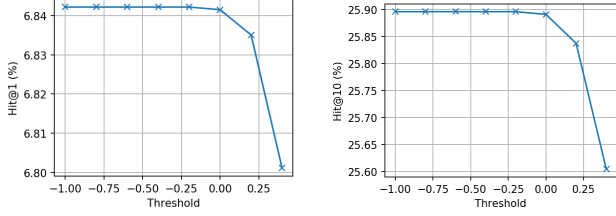


Figure 4. Effects of threshold τ on the performance of the baseline model.

5.5. Deviated Model

We experiment on the values of r on the model described in Section 3.3, and compare its performance with base line model. Note that r is equal to $T/2$ here. Figure 5 shows the improvement of the deviated model (indicated in orange line) as the values of r is increasing. However, when we observe Hit@1 and Hit@5 separately, we can see that Hit rate when $k = 1$ improves significantly more and starts to drop as $r \geq 2$. Contrastingly, the Hit rate $k = 5$ for this deviated model is similar to the baseline and getting better than later as $r \geq 4$.

We deduce that our model has a greater impact on predicting top 1 label than top 5 which is reasonable because the deviated model should theoretically benefit the testing image that relies greatly on the information from multiple training labels. As mentioned in 5, we aim to improve the performance by involving substantial difference labels. This will allow the model to improve the prediction marginally and give a small change in ranking the prediction.

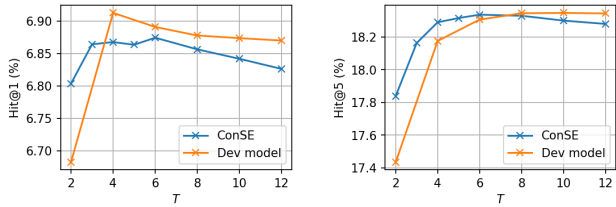


Figure 5. Comparison of hit rate of deviated model and the baseline model.

5.6. Other Observations

We observe that the variations between each model are relatively small. One question that may arise is whether the change in each model only affects the predictions of a few images, while leaving the rest unaf-

fected. If this were the case, the effects of each model variation would be uninteresting and ineffective.

Therefore, we further analyze the words by high-level category to investigate what word categories perform well or badly. We can see from Table 2 that the deviated model mostly increases the prediction accuracy. Most of the categories benefits from the model, indicating that various and substantially different images get affected by our proposed model.

Table 2. Hit@5 by word category. 840B 300d. A: ConSE, B: threshold $\tau = 0.2$, C: $r = 5$.

High-level category	No. of synsets	Average hit@5 (%)		
		A	B	C
artifact, artefact	999	16.96	16.89	17.02
animal, animate being, beast, brute, creature, fauna	257	27.76	27.77	27.79
miscellaneous	108	12.71	12.67	12.62
natural object	56	8.91	8.93	8.85
geological formation, formation	20	21.85	21.93	22.07
plant, flora, plant life	19	6.08	6.00	6.21
fungus	9	48.22	48.28	48.22
person, individual, someone, somebody, mortal, soul	8	9.75	9.62	9.75
Total	1476	18.30	18.26	18.35

While the change in average hit is relatively small, the model affects many predictions. For example, the number of images whose true labels were not in top-5 ConSE predictions but in top-5 model B predictions is 203, and the opposite is 330. For model C, the numbers are 1443 and 1304, respectively. We can see that the recombination schemes of models B and C actually affect a lot of predictions. However, since the number of images whose predictions get better and those whose predictions get worse are approximately the same, the changes in average hit are relatively little. An interesting point here is how to eliminate the negatively affected predictions while keeping or increasing the number of improved predictions.

6. Conclusion

We have investigated the zero-shot approach to n -way image classification model using convex combination

of word embeddings. Using large database of images, we experimented and analyzed the direct approach, which simply combines the word vectors of top predictions from a Convolutional Neural Network, then find top nearest neighbors of this synthetic vector in the word embedding space. Then we have proposed two variations of this baseline model, each of which employs a criterion to choose only some of the top predictions to recalculate the synthetic vector. While one model barely benefits the prediction accuracy, the other model, which is based on the idea that different word embeddings all play an important role, increases the prediction accuracy by a little. Along with the analysis on the number of word embeddings we take into account in the baseline model, most of the results point to the direction that convex combination of word embedding indeed benefit the image classification task.

Division of Labor

Three group members contributed equally. However, each of us focused on different sections of the project. K.K. processed the image data from ImageNet, mainly implemented the threshold model and analysed the results K.P. ran CNN and generated the plots for the report S.V. focused on the processing word embeddings, introduced the new deviated model, and analyzed its results.







Acknowledgments

We would like to thank the instructors of Machine Learning class (6.867) Professor Devavrat Shah, Assistant Professor David Sontag, Professor Suvrit Sra and our TA Curtis Northcutt for giving helpful advice and feedback during the development of our project.

References

- [1] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [2] Frome, Andrea, Corrado, Greg S, Shlens, Jon, Bengio, Samy, Dean, Jeff, Ranzato, Marc Aurelio, and Mikolov, Tomas. Devise: A deep visual-semantic embedding model. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2121–2129. Curran Associates, Inc., 2013. URL <http://papers.nips.cc/paper/5204-devise-a-deep-visual-semantic-embedding.pdf>.
- [3] Lampert, C., Nickisch, H., and Harmeling, S. Learning to detect unseen object classes by between-class attribute transfer. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [4] Norouzi, Mohammad, Mikolov, Tomas, Bengio, Samy, Singer, Yoram, Shlens, Jonathon, Frome, Andrea, Corrado, Greg, and Dean, Jeffrey. Zero-shot learning by convex combination of semantic embeddings. In *International Conference on Learning Representations*, 2014. URL <http://arxiv.org/abs/1312.5650>.
- [5] Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014. URL <http://www.aclweb.org/anthology/D14-1162>.
- [6] Socher, Richard, Ganjoo, Milind, Manning, Christopher D., and Ng, Andrew Y. Zero-shot learning through cross-modal transfer. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS’13, pp. 935–943, USA, 2013. Curran Associates Inc. URL <http://dl.acm.org/citation.cfm?id=2999611.2999716>.
- [7] Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. URL <http://arxiv.org/abs/1409.4842>.

Table 3. Examples of our models, based on 840B 300d word embeddings. Blue refers to selected labels in threshold $\tau = 0.0$ model, and pink refers to selected labels in the $r = 3$ Deviated Model. “Sim” refers to the cosine similarity between each word label and the first synthetic vector.

Image	Softmax over 1K synsets			ConSE		Our Threshold Model ($\tau = 0.0$)	
	Labels	Prob	Sim	Labels		Labels	
	bicycle-built-for-two, tandem bicycle, tandem jinrikisha, ricksha, rickshaw tricycle, trike, velocipede unicycle, monocyte barrow, garden cart, lawn cart, wheelbarrow moped file, file cabinet, filing cabinet crate carton shopping cart	0.117 0.115 0.097 0.047 0.031 0.027 0.013 0.010 0.009 0.009	0.617 0.635 0.796 0.642 0.513 0.578 -0.015 0.252 0.163 0.400	scooter water scooter, sea scooter, scooter pedicab, cycle rickshaw velocipede parasail skateboard toboggan hearse luge kayak		pedicab, cycle rickshaw scooter water scooter, sea scooter, scooter velocipede parasail skateboard toboggan hearse luge kayak	
	Loafer clog, geta, patten, sabot sandal cowboy boot shoe shop, shoe-shop, shoe store holster running shoe buckle plane, carpenter's plane, woodworking plane mailbag, postbag	0.613 0.167 0.115 0.039 0.016 0.004 0.004 0.002 0.001 0.001	0.949 0.523 0.650 0.304 -0.008 0.085 0.485 0.380 -0.096 0.069	moccasin, mocassin slingback, sling espadrille brake shoe, shoe, skid shoe brogan, brogue, clodhopper, work shoe chukka, chukka boot bomber, grinder, hero, hero sandwich, ... blucher instep		moccasin, mocassin slingback, sling espadrille brogan, brogue, clodhopper, work shoe brake shoe, shoe, skid shoe chukka, chukka boot bomber, grinder, hero, hero sandwich, ... blucher instep	
	suit, suit of clothes miniskirt, mini Windsor tie trench coat Loafer groom, bridegroom bow tie, bow-tie, bowtie bolo tie, bolo, bola tie, bola racket, racquet lab coat, laboratory coat	0.729 0.071 0.037 0.027 0.024 0.014 0.013 0.008 0.003 0.003	0.986 0.393 0.448 0.411 -0.020 0.358 0.332 0.139 0.206 0.481	singlet, vest, undershirt slacks jump suit, jumpsuit tartan, plaid gown, surgical gown, scrubs nightgown, gown, nightie, night-robe, ... suing trouser case, compositor's case, typesetter's case casing, case		singlet, vest, undershirt slacks jump suit, jumpsuit gown, surgical gown, scrubs nightgown, gown, nightie, night-robe, ... tartan, plaid suing trouser case, compositor's case, typesetter's case casing, case	
	parachute, chute assault rifle, assault gun gasmask, respirator, gas helmet military uniform ski rifle bulletproof vest ballplayer, baseball player pickelhaube scuba diver	0.166 0.122 0.087 0.057 0.057 0.050 0.043 0.029 0.024 0.021	0.739 0.520 0.490 0.486 0.432 0.573 0.248 0.208 0.298 0.433	singlet, vest, undershirt automatic pistol, automatic grease-gun, gun goggles camouflage, camo camouflage cannon carbine flamethrower revolving door, revolver		automatic pistol, automatic flamethrower singlet, vest, undershirt drogue, drogue chute, drogue parachute catapult, launcher parasail blimp, sausage balloon, sausage beret turret camouflage, camo	
	hot pot, hotpot caldron, cauldron wooden spoon wok soup bowl spaghetti squash Crock Pot butternut squash carbonara ladle	0.146 0.074 0.066 0.056 0.035 0.028 0.026 0.026 0.022 0.021	0.781 0.622 0.458 0.697 0.514 0.485 0.600 0.485 0.529 0.635	crock, earthenware jar paella petite marmite, minestrone, ... colander, cullender jambalaya sukiyaki gravy risotto, Italian rice couscous tempura		sukiyaki petite marmite, minestrone, ... jambalaya crock, earthenware jar tempura paella terrine gumbo gumbo, okra croquette	
	printer CD player projector modem mouse, computer mouse photocopier tape player cassette player desktop computer laptop, laptop computer	0.370 0.130 0.124 0.085 0.080 0.048 0.038 0.015 0.011 0.008	0.909 0.609 0.639 0.583 0.473 0.633 0.609 0.609 0.409 0.592	scanner, digital scanner, image scanner toner background, desktop, screen background facsimile, facsimile machine, fax adapter, adaptor print router screen door, screen screen door, screen stylus, style flatcar, flatbed, flat		scanner, digital scanner, image scanner toner background, desktop, screen background adapter, adaptor print router screen door, screen facsimile, facsimile machine, fax stylus, style cable, cable television, cable system, ...	