Unsupervised Learning on Neural Network Outputs:
with Application in Zero-Shot Learning

Yao Lu

Aalto University
University of Helsinki
Helsinki Institute for Information Technology
yaolubrain@gmail.com

Abstract

The outputs of a trained neural network contain much richer information than just a one-hot classifier. For example, a neural network might give an image of a dog the probability of one in a million of being a cat but it is still much larger than the probability of being a car. To reveal the hidden structure in them, we apply two unsupervised learning algorithms, PCA and ICA, to the outputs of a deep Convolutional Neural Network trained on the ImageNet of 1000 classes. The PCA/ICA embedding of the object classes reveals their visual similarity and the PCA/ICA components can be interpreted as common visual features shared by similar object classes. For an application, we proposed a new zero-shot learning method, in which the visual features learned by PCA/ICA are employed. Our zero-shot learning method achieves the state-of-the-art results on the ImageNet of over 20000 classes.

1 Introduction

Recently, Convolutional Neural Network (CNN) [LeCun *et al.*, 1998] has made significant advances in computer vision tasks such as image classification [Ciresan *et al.*, 2012; Krizhevsky *et al.*, 2012; Szegedy *et al.*, 2015] , object detection [Girshick *et al.*, 2014; Shaoqing Ren, 2015] and image segmentation [Turaga *et al.*, 2010; Long *et al.*, 2015]. Moreover, CNN also sheds lights on neural coding in visual cortex. In [Cadieu *et al.*, 2014], it has been shown that a trained CNN rivals the representational performance of inferior temporal cortex on a visual object recognition task. Therefore, investigating the properties of a trained CNN is important for both computer vision applications and discovering the principles of neural coding in the brain.

In [Hinton *et al.*, 2014], it is shown that the softmax outputs of a trained neural network contain much richer information than just a one-hot classifier. Such a phenomenon is called *dark knowledge*. For input vector $\mathbf{y} = (y_1, ..., y_n)$, which is called logits in [Hinton *et al.*, 2014], the softmax function produces output vector $\mathbf{x} = (x_1, ..., x_n)$ such that

$$x_i = \frac{\exp(y_i/T)}{\sum_j \exp(y_j/T)} \tag{1}$$

where T is the temperature parameter. The softmax function assigns positive probabilities to all classes since $x_i > 0$ for all i. Given a data point of a certain class as input, even when the probabilities of the incorrect classes are small, some of them are much larger than the others. For example, in a 4-class classification task (cow, dog, cat, car), given an image of a dog, while a hard target (class label) is $(0, 1, 0, 0)$, a trained neural network might output a soft target $(10^{-6}, 0.9, 0.1, 10^{-9})$. An image of a dog might have small chance to be misclassified as cat but it is much less likely to be misclassified as car. In [Hinton *et al.*, 2014], a technique called knowledge distillation was introduced to further reveal the information in the softmax outputs. Knowledge distillation raises the temperature T in the softmax function to soften the outputs. For example, it transforms $(10^{-6}, 0.9, 0.1, 10^{-9})$ to $(0.015, 0.664, 0.319, 0.001)$ by raising temperature T from 1 to 3. It has been shown that adding the distilled soft targets in the objective function helps in reducing generalization error when training a smaller model of an ensemble of models [Hinton *et al.*, 2014]. Therefore, the outputs of a trained neural network are far from one-hot hard targets or random noise and they might contain rich statistical structures.

In this paper, to explore the information hidden in the outputs, we apply two unsupervised learning algorithms, Principle Component Analysis (PCA) and Independent Component Analysis (ICA) to the outputs of a CNN trained on the ImageNet dataset [Deng *et al.*, 2009] of 1000 object classes. Both PCA and ICA are special cases of the Factor Analysis model, with different assumptions on the latent variables. Factor Analysis is a statistical model which can be used for revealing hidden factors that underlie a vector of random variables. In the case of CNN for image classification, the neurons or computational units in the output layer of a CNN, as random variables, represent object classes. A latent factor might represent a common visual attribute shared by several object classes. It is therefore desirable to visualize, interpret and make use of the Factor Analysis models learned on the outputs of a trained CNN.

2 Softmax

Because a CNN was trained with one-hot hard targets (class labels), given a training image as input, the softmax function suppresses the outputs of most neurons in the output layer and leaves one or a few peak values. For example, in Figure 1 (a),

we show the softmax ($T = 1$) outputs for a training image. To magnify the tiny values in the softmax outputs, after a CNN was trained with softmax function ($T = 1$), we take the logits $\mathbf{y}$ in e.q. (1) and apply the following normalization function

$$x_i = \frac{(y_i - \min_k y_k)}{\sum_j (y_j - \min_k y_k)} \qquad (2)$$

for all i, as the outputs of the CNN, with all the parameters in the CNN unchanged. This function normalizes $\mathbf{y}$ so that $\mathbf{x}$ in eq. (2) is still a probability distribution over classes. We call the $\mathbf{x}$ in eq. (2) normalized logits. In Figure 1 (b), we show the outputs of this function given the same input image as Figure 1 (a).
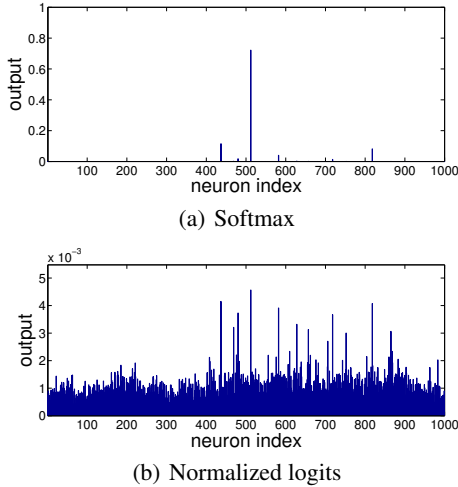

(a) Softmax

(b) Normalized logits

Figure 1: Outputs

In order to apply ICA, the variables must not all be Gaussian. The non-Gaussianity of a random variable x of zero mean can be measured by kurtosis $E(x^4)/E(x^2)^2 - 3$, which is zero if x is Gaussian. We computed the kurtosis of the outputs (mean removed) of a CNN with softmax and normalized logits using all the ImageNet ILSVRC2012 training data. The CNN model and experimental settings are described in Section 4. The result is, all neurons in the output layer have positive kurtosis, as shown in Figure 2. Therefore the neurons as random variables are highly non-Gaussian and it is sensible to apply ICA, which is introduced in the next section.

3 Factor Analysis

In Factor Analysis, we assume the observed variables $\mathbf{x} = (x_1, ..., x_n)$ are generated by the following model

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{n} \qquad (3)$$

where $\mathbf{s} = (s_1, ..., s_n)$ are the latent variables, $\mathbf{A}$ is the model parameter matrix and $\mathbf{n}$ are the noise variables. Here, $\mathbf{x}$ and $\mathbf{s}$ are assumed to have zero-mean. $\mathbf{s}$ are also assumed to be uncorrelated and have unit variance, in other words, white.
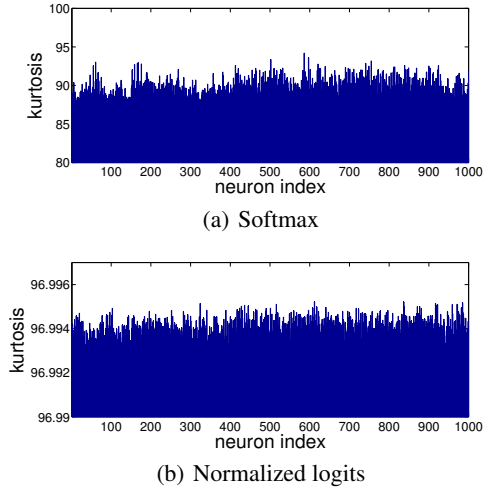

(a) Softmax

(b) Normalized logits

Figure 2: Kurtosis

3.1 Principle Component Analysis

Principle Component Analysis (PCA) is a special case of Factor Analysis. In PCA, $\mathbf{s}$ are assumed to be Gaussian and $\mathbf{n}$ are assumed to be zero (noise-free). Let $\mathbf{C}$ denote the covariace matrix of $\mathbf{x}$, $\mathbf{E} = (\mathbf{e}_1, ..., \mathbf{e}_n)$ denote the matrix of eigenvectors of $\mathbf{C}$ and $\mathbf{D} = \mathrm{diag}(\lambda_1, ..., \lambda_n)$ denote the diagonal matrix of eigenvalues of $\mathbf{C}$. The PCA matrix is $\mathbf{E}^T$, the whitening matrix is $\mathbf{U} = \mathbf{D}^{-1/2}\mathbf{E}^T$ and the whitened variables are $\mathbf{z} = \mathbf{U}\mathbf{x}$.

3.2 Independent Component Analysis

Independent Component Analysis (ICA) [Hyvärinen *et al.*, 2004] is another special case of Factor Analysis. In ICA, $\mathbf{s}$ are assumed to be non-Gaussian and independent and $\mathbf{n}$ are assumed to be zero. ICA seeks a demixing matrix $\mathbf{W}$ such that $\mathbf{W}\mathbf{x}$ can be as independent as possible. To obtain $\mathbf{W}$, we can first decompose it as $\mathbf{W} = \mathbf{V}\mathbf{U}$, where $\mathbf{U}$ is the whitening matrix and $\mathbf{V}$ is an orthogonal matrix, which can be learned by maximizing the non-Gaussianity or the likelihood function of $\mathbf{V}\mathbf{U}\mathbf{x}$. The non-Gaussianity can be measured by kurtosis or negentropy. If dimensionality reduction is required, we can take the d largest eigenvalues and the corresponding eigenvectors for the whitening matrix $\mathbf{U}$. As a result, the size of $\mathbf{U}$ is $d \times n$ and the size of $\mathbf{V}$ is $d \times d$. Scaling each component does not affect ICA solutions. If $\mathbf{W}$ is an ICA demixing matrix, then $\mathrm{diag}(\alpha_1, ..., \alpha_d)\mathbf{W}$ is also an ICA demixing matrix, where $\{\alpha_1, ..., \alpha_d\}$ are non-zero scaling constants of the components.

A classic ICA algorithm is FastICA [Hyvarinen, 1999]. Despite its fast convergence, FastICA is a batch algorithm which requires all the data to be loaded for computation in each iteration. Thus, it is unsuitable for large scale applications. To handle large scale datasets, we use a stochastic gradient descent (SGD) based ICA algorithm (described in the Appendix of [Hyvarinen, 1999]). For samples $\{\mathbf{z}(1), \mathbf{z}(2), ...\}$, one updating step of the SGD-based algo-

rithm of a given sample $\mathbf{z}(t)$ is:

$$\mathbf{V} \leftarrow \mathbf{V} + \mu g(\mathbf{V}\mathbf{z}(t))\mathbf{z}(t)^T + \frac{1}{2}(\mathbf{I} - \mathbf{V}\mathbf{V}^T)\mathbf{V}^T \quad (4)$$

where μ is the learning rate, $g(\cdot) = -\tanh(\cdot)$ and $\mathbf{I}$ is an identity matrix. In our experiments, $\mathbf{V}$ was initialized as a random orthogonal matrix.

Like FastICA, this SGD-based algorithm requires going through all data once to compute the whitening matrix $\mathbf{U}$. But unlike FastICA, this SGD-based algorithm does not require projection or orthogonalization in each step.

In this algorithm, the assumption on the probability distribution of each s_i is a super-Gaussian distribution

$$\log p(s_i) = -\log\cosh(s_i) + \text{constant} \quad (5)$$

and therefore

$$g(s_i) = \frac{\partial}{\partial s_i}\log p(s_i) = -\tanh(s_i). \quad (6)$$

Since the variables obtained by linear transformations of Gaussian variables are also Gaussian, from Section 2, we can infer at least one neuron in the output layer is non-Gaussian. As an initial attempt, we choose a particular non-Gaussian distribution here. Explorations of different non-Gaussian distributions and therefore different nonlinearities $g(\cdot)$ are left for future research.

4 Results

4.1 Experimental Settings

For the trained CNN model, we used GoogLeNet [Szegedy *et al.*, 2015] and AlexNet [Krizhevsky *et al.*, 2012]. The results of using two different CNN models are similar. Therefore, due to the space limitation, we only report the results of using GoogLeNet. We used all the images in the ImageNet ILSVRC2012 training set to compute the ICA matrix using our SGD-based algorithm with mini-batch size 500. The learning rate was set to 0.005 and was halved every 10 epochs. The computation of CNN outputs was done with Caffe [Jia *et al.*, 2014]. The ICA algorithm was ran with Theano [Bergstra *et al.*, 2010].

4.2 Visualization of PCA/ICA components

To understand what is learned by PCA and ICA, we visualize the PCA and the ICA matrices. In the PCA matrix $\mathbf{E}^T$ or ICA matrix $\mathbf{W}$, each row corresponds to a PCA/ICA component and each column corresponds to an object class. The number of rows depends on the dimensionality reduction. The number of columns of $\mathbf{E}^T$ or $\mathbf{W}$ is 1000, corresponding to 1000 classes. After the ICA matrix was learned, each ICA component (a row of $\mathbf{W}$) was scaled to have unit l_2 norm. The scaling of each ICA component does not affect the ICA solution, as discussed in Section 3.2.
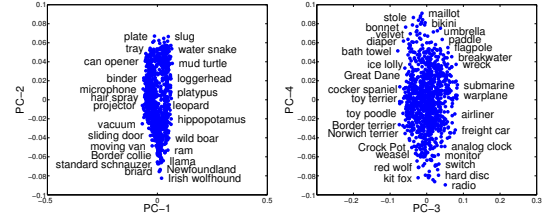
In Figure 3, we show the embedding of class labels by PCA and ICA. The horizontal and the vertical axes are two distinct rows of $\mathbf{E}^T$ or $\mathbf{W}$. Each point in the plot corresponds to an object class. And there are 1000 points in each plot. Dimensionality is reduced from 1000 to 200 in ICA. In Figure 3 (a) and (b), we plot two pairs of the PCA/ICA components,
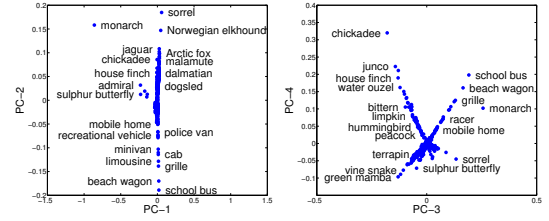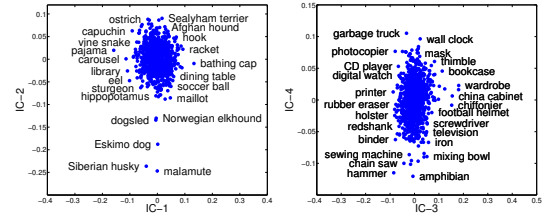

(a) PCA, softmax.
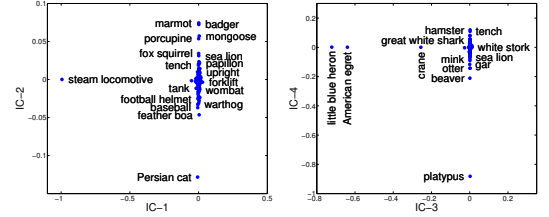
(b) ICA, softmax.

(c) PCA, normalized logits.

(d) ICA, normalized logits.

Figure 3: Label embedding of object class by PCA/ICA components. In each plot, each point is an object class and each axis is a PCA/ICA component (PC/IC). For visual clarity, only selected points are annotated with object class labels.

learned with softmax outputs. In the PCA embedding, visually similar class labels are along some lines, but not the axes, while in the ICA embedding, they are along the axes. However, most points are clustered in the origin. In Figure 3 (c) and (d), we plot two pairs of the PCA/ICA components, learned with normalized logits outputs. We can see the class labels are more scattered in the plots.

In Table 1, we show the top-5 object classes according to the value of PCA/ICA components. For the ease of comparison, we selected each PCA/ICA component which has the largest value for class *mosque*, *killer whale*, *Model T* or *zebra* among all components. We can see that the class labels ranked by ICA components are more visually similar and consistent than the ones by PCA components.

Table 1: Object classes ranked by single components of PCA/ICA

	1	2	3	4
	mosque	killer whale	Model T	zebra
	shoji	beaver	strawberry	tiger
PCA	trimaran	valley	hay	chickadee
	fire screen	otter	electric locomotive	school bus
	aircraft carrier	loggerhead	scoreboard	yellow lady's slipper
	mosque	killer whale	Model T	zebra
	barn	grey whale	car wheel	tiger
ICA	planetarium	dugong	tractor	triceratops
	dome	leatherback turtle	disk brake	prairie chicken
	palace	sea lion	barn	warthog

Table 2: Closest object classes in terms of visual and semantic similarity

	Egyptian cat	soccer ball	mushroom	red wine
	tabby cat	rugby ball	bolete	wine bottle
	tiger cat	croquet ball	agaric	beer glass
Visual	tiger	racket	stinkhorn	goblet
	lynx	tennis ball	earthstar	measuring cup
	Siamese cat	football helmet	hen-of-the-woods	wine bottle
	Persian cat	croquet ball	cucumber	eggnog
	tiger cat	golf ball	artichoke	cup
Semantic	Siamese cat	baseball	cardoon	espresso
	tabby cat	ping-pong ball	broccoli	menu
	cougar	punching bag	cauliflower	meat loaf

The PCA/ICA components can be interpreted as common features shared by visually similar object classes. From Figure 3 and Table 1, we can see the label embeddings of object classes by PCA/ICA components are meaningful since visually similar classes are close in the embeddings. Unlike [Akata *et al.*, 2013], these label embeddings can be unsupervisedly learned with a CNN trained with only one-hot class labels and without any hand annotated attribute label of the object classes, such as *has tail* or *lives in the sea*.

4.3 Visual vs. Semantic Similarity

The visual-semantic similarity relationship was previously explored in [Deselaers and Ferrari, 2011], which shows some consistency between two similarities. Here we further explore it from another perspective. We define the visual and the semantic similarity in the following way. The visual similarity between two object classes is defined as cosine similarity of their PCA or ICA components (200-dim and learned with softmax), both of which give the same results. The semantic similarity is defined based on the shortest path length [1] between two classes on the WordNet graph [Fellbaum, 1998].

In Table 2, we compare five closest classes of *Egyptian cat*, *soccer ball*, *mushroom* and *red wine* in terms of visual and semantic similarities. For *Egyptian cat*, both visual and semantic similarities give similar results. For *soccer ball*, *football helmet* is close in terms of visual similarity but distant in

terms of semantic similarity. For *mushroom* and *red wine*, two similarities give very different closest object classes. The gap between two similarities is intriguing and therefore worth further exploration. In neuroscience literature, it is claimed that visual cortex representation favors visual rather than semantic similarity [Baldassi *et al.*, 2013].

5 Application: Zero-shot Learning

To demonstrate the effectiveness of the visual features of object classes learned by PCA and ICA, we apply them to zero-shot learning. Zero-shot learning [Larochelle *et al.*, 2008; Lampert *et al.*, 2009; Palatucci *et al.*, 2009] is a classification task in which some classes have no training data at all. We call the classes which have training data *seen classes* and those which have no training data *unseen classes*. One can use external knowledge of the classes, such as attributes, to build the relationship between the seen and the unseen classes. Then one can extrapolate the unseen classes by the seen classes.

5.1 Previous Work

The current state-of-the-art large scale zero-shot learning methods are DeViSE [Frome *et al.*, 2013] and conSE [Norouzi *et al.*, 2014]. Both of them use the ImageNet of 1000 classes for training and the ImageNet of over 20000 classes for testing. Our method differs from DeViSE and conSE by using unsupervised learning algorithms to learn: (1) Visual features of classes. (2) Semantic features of classes

---

[1] Computed with the path_similarity() function in the NLTK tool http://www.nltk.org/howto/wordnet.html.

from the WordNet graph, instead of word2vec [Mikolov *et al.*, 2013] on Wikipedia. (3) A bridge between the visual and the semantic features.

5.2 Our Method

Our method works as follows. In the learning phase, first assume we have obtained the visual feature vectors $\mathbf{W}^{(1)} = (\mathbf{w}_1^{(1)}, ..., \mathbf{w}_n^{(1)})$ of n seen classes. Let $\mathbf{M} = (\mathbf{m}_1, ..., \mathbf{m}_n)$ denote the matrix of the mean outputs of a CNN of the seen classes. And $\mathbf{F} = f(\mathbf{W}^{(1)}\mathbf{M}) = (\mathbf{f}_1, ..., \mathbf{f}_n)$ are the transformed mean outputs of the seen classes, where $f(\cdot)$ is a non-linear function. Next, assume we have obtained the semantic feature vectors $\mathbf{W}^{(2)} = (\mathbf{w}_1^{(2)}, ..., \mathbf{w}_n^{(2)})$ of n seen classes and $\mathbf{W}^{(3)} = (\mathbf{w}_1^{(3)}, ..., \mathbf{w}_m^{(3)})$ of m unseen classes. Due to the visual-semantic similarity gap shown in Section 4.3, we learn a bridge between the visual and the semantic representations of object classes via Canonical Correlation Analysis (CCA) [Hotelling, 1936; Hardoon *et al.*, 2004], which seeks two projection matrices $\mathbf{P}^{(1)}$ and $\mathbf{P}^{(2)}$ such that

$$\min_{\mathbf{P}^{(1)},\mathbf{P}^{(2)}} \|\mathbf{P}^{(1)T}\mathbf{F} - \mathbf{P}^{(2)T}\mathbf{W}^{(2)}\|_F \quad (7)$$

$$\text{s.t.} \quad \mathbf{P}^{(k)T}\mathbf{C}_{kk}\mathbf{P}^{(k)} = \mathbf{I}, \quad \mathbf{p}_i^{(k)T}\mathbf{C}_{kl}\mathbf{p}_j^{(l)} = 0, \quad (8)$$

$$k, l = 1, 2, \quad k \neq l, \quad i, j = 1, ..., d, \quad (9)$$

where $\mathbf{p}_i^{(k)}$ is the i-th column of $\mathbf{P}^{(k)}$ and $\mathbf{C}_{kl}$ is a co-variance or cross-covariance matrix of $\{\mathbf{f}_1, ..., \mathbf{f}_n\}$ and/or $\{\mathbf{w}_1^{(2)}, ..., \mathbf{w}_n^{(2)}\}$.

In the testing phase, when a new image arrives, we first compute its CNN output $\mathbf{x}$. Then for $\mathbf{P}^{(1)T}(f(\mathbf{W}^{(1)}\mathbf{x}) - \frac{1}{n}\sum_i \mathbf{f}_i)$, we compute its k closest columns of $\mathbf{P}^{(2)T}\mathbf{W}^{(2)}$ (seen) and/or $\mathbf{P}^{(2)T}\mathbf{W}^{(3)}$ (unseen). The corresponding classes of these k columns are the top-k predictions. The closeness is measured by cosine similarity.

For $\mathbf{W}^{(1)}$, we compare random, PCA, and ICA matrices of different dimensionality in our experiments. The random matrices are semi-orthogonal, that is, $\mathbf{W}^{(1)}\mathbf{W}^{(1)T} = \mathbf{I}$ but $\mathbf{W}^{(1)}\mathbf{W}^{(1)T} \neq \mathbf{I}$. For $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$, we use the feature vectors by running classic Multi-dimensional Scaling (MDS) on a distance matrix of both seen and unseen classes. The distance between two classes is measured by one minus the similarity in Section 4.3. Each column of $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$ is subtracted by $\frac{1}{n}\sum_i \mathbf{w}_i^{(2)}$. $\mathbf{M}$ is approximated by $\mathbf{I}$ and $f(\cdot)$ is the scaling normalization of a vector or each column of a matrix to unit l_1 norm. We experimented with softmax with different T and normalized logits as the outputs. The best performance (as in Table 3 and Table 4) was obtained with the softmax ($T = 1$) output for $\mathbf{x}$ but $\mathbf{E}^T$ and $\mathbf{W}$ were learnt with normalized logits.

5.3 Experiments

Following the zero-shot learning experimental settings of DeViSE and conSE, we used a CNN trained on ImageNet ILSVRC2012 (1000 seen classes), and test our method to classify images in ImageNet 2011fall (20842 unseen

classes [2], 21841 both seen and unseen classes). We use top-k accuracy (also called flat hit@k in [Frome *et al.*, 2013; Norouzi *et al.*, 2014]) measure, the percentage of test images in which a method's top-k predictions return the true label.

For the trained CNN model, we experimented with GoogLeNet and AlexNet. Although GoogLeNet outperforms AlexNet on the seen classes, two CNN models in our method perform essentially the same on the zero-shot learning tasks. Due to the space limitation, we only report the results of using GoogLeNet.

The sizes of the matrices in our methods: $\mathbf{W}^{(1)}$ is $k \times 1000$, $\mathbf{W}^{(2)}$ is 21632×1000, $\mathbf{W}^{(3)}$ is 21632×20842, $\mathbf{P}^{(1)}$ is $k \times k$, $\mathbf{P}^{(2)}$ is $k \times 21632$, $\mathbf{M}$ is 1000×1000 and $\mathbf{x}$ is 1000×1. We used $k = 100, 500, 900$ in our experiments. Although $\mathbf{W}^{(2)}$ and $\mathbf{W}^{(3)}$ are large matrices, we only need to compute once and store $\mathbf{P}^{(2)}\mathbf{W}^{(2)}$ and $\mathbf{P}^{(2)}\mathbf{W}^{(3)}$ of size $k \times 1000$ and $k \times 21632$, respectively.

In Table 3, we show the results of the three zero-shot learning methods on the test images selected in [Norouzi *et al.*, 2014]. Same as conSE, our method gives correct or reasonable predictions. In Table 4, we show the results of different methods on ImageNet 2011fall. Our method performs better when using PCA or ICA for the visual features than random features. And our method, especially with PCA or ICA features, achieves superior results on this zero-shot learning task.

The results show that in our method the PCA or ICA matrix as visual features of object classes performs better than a random matrix. Therefore, these visual features, learned PCA and ICA on the outputs of CNN, are indeed effective for the subsequent tasks. The results also show that PCA and ICA give the essentially same classification accuracy. Therefore, in practice we can use PCA instead of ICA, which has much higher computational costs. A comprehensive discussion on PCA vs. ICA for recognition tasks can be found in [Asuncion Vicente *et al.*, 2007].

The code for reproducing the experiments is in [3].

6 Discussions

The outputs of a neural network contains rich information. CNN was trained with only one-hot targets, which means we assumed object classes are equally similar and CNN was never informed which classes more similar. However, unsupervised learning on CNN outputs reveals the visual similarity of object classes, which sheds lights on the object representation in CNN. Supervised learning alone cannot deal with unseen classes. With unsupervised learning and external knowledge, we can leverage supervised learning to make reasonable predictions on the unseen classes, that is, zero-shot learning.

---

[2]Since class *teddy, teddy bear* is missing in ImageNet 2011fall, the correct number of classes is $21841-(1000-1) = 20842$ rather than 20841.

[3]https://github.com/yaolubrain/ULNNO

Table 3: Predictions of test images of unseen classes (correct class labels are in blue)
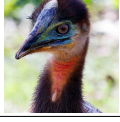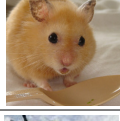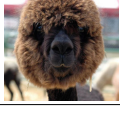
Test Images	DeViSE [Frome *et al.*, 2013]	ConSE [Norouzi *et al.*, 2014]	Our Method
	water spaniel tea gown bridal gown, wedding gown spaniel tights, leotards	business suit **dress, frock** hairpiece, false hair, postiche swimsuit, swimwear, bathing suit kit, outfit	periwig, peruke horsehair wig hound, hound dog bonnet macaque toupee, toupe
	heron owl, bird of Minerva, bird of night hawk bird of prey, raptor, raptorial bird finch	**ratite, ratite bird, flightless bird** peafowl, bird of Juno common spoonbill New World vulture, cathartid Greek partridge, rock partridge	**ratite, ratite bird, flightless bird** kiwi, apteryx moa elephant bird, aepyornis emu, Dromaius novaehollandiae
	elephant turtle turtleneck, turtle, polo-neck flip-flop, thong handcart, pushcart, cart, go-cart	California sea lion **Steller sea lion** Australian sea lion South American sea lion eared seal	fur seal [a] eared seal fur seal [b] guadalupe fur seal Alaska fur seal
	golden hamster, Syrian hamster rhesus, rhesus monkey pipe shaker American mink, Mustela vison	**golden hamster, Syrian hamster** rodent, gnawer Eurasian hamster rhesus, rhesus monkey rabbit, coney, cony	**golden hamster, Syrian hamster** Eurasian hamster prairie dog, prairie marmot skink, scincid, scincid lizard mountain skink
	truck, motortruck skidder tank car, tank automatic rifle, machine rifle trailer, house trailer	flatcar, flatbed, flat truck, motortruck tracked vehicle bulldozer, dozer wheeled vehicle	**farm machine** cultivator, tiller skidder bulldozer, dozer haymaker, hay conditioner
	kernel littoral, litoral, littoral zone, sands carillon Cabernet, Cabernet Sauvignon poodle, poodle dog	dog, domestic dog domestic cat, house cat schnauzer Belgian sheepdog domestic llama, Lama peruana	mastiff **alpaca, Lama pacos** domestic llama, Lama peruana guanaco, Lama guanicoe Seeing Eye dog

Table 4: Top-*k* accuracy in ImageNet 2011fall zero-shot learning task (%)

Test Set	#Classes	#Images	Method	Top-1	Top-2	Top-5	Top-10	Top-20
Unseen	20842	12.9 million	DeViSE (500-dim)	0.8	1.4	2.5	3.9	6.0
			ConSE (500-dim)	1.4	2.2	3.9	5.8	8.3
			Our method (100-dim, random)	1.4	2.2	3.4	4.3	5.2
			Our method (100-dim, PCA)	1.6	2.7	4.6	6.4	8.6
			Our method (100-dim, ICA)	1.6	2.7	4.6	6.3	8.5
			Our method (500-dim, random)	**1.8**	2.9	5.0	6.9	8.8
			Our method (500-dim, PCA)	**1.8**	**3.0**	**5.2**	**7.3**	9.6
			Our method (500-dim, ICA)	**1.8**	**3.0**	**5.2**	**7.3**	**9.7**
			Our method (900-dim, random)	**1.8**	**3.0**	5.1	7.2	9.6
			Our method (900-dim, PCA)	**1.8**	**3.0**	**5.2**	**7.3**	**9.7**
			Our method (900-dim, ICA)	**1.8**	**3.0**	**5.2**	**7.3**	**9.7**
Both	21841	14.2 million	DeViSE (500-dim)	0.3	0.8	1.9	3.2	5.3
			ConSE (500-dim)	0.2	1.2	3.0	5.0	7.5
			Our method (100-dim, random)	**6.7**	8.2	10.0	11.1	12.1
			Our method (100-dim, PCA)	**6.7**	8.1	10.3	12.4	14.8
			Our method (100-dim, ICA)	**6.7**	8.1	10.4	12.4	14.7
			Our method (500-dim, random)	**6.7**	**8.5**	11.2	13.4	15.6
			Our method (500-dim, PCA)	**6.7**	**8.5**	**11.4**	**13.7**	**16.3**
			Our method (500-dim, ICA)	**6.7**	**8.5**	**11.4**	**13.7**	**16.3**
			Our method (900-dim, random)	**6.7**	**8.5**	**11.4**	**13.7**	16.2
			Our method (900-dim, PCA)	**6.7**	**8.5**	**11.4**	**13.7**	**16.3**
			Our method (900-dim, ICA)	**6.7**	**8.5**	**11.4**	**13.7**	**16.3**

[a] WordNet ID: n02077152. There are two classes named *fur seal* with different WordNet IDs.
[b] WordNet ID: n02077658.

References

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

[Asuncion Vicente *et al.*, 2007] M Asuncion Vicente, Patrik O Hoyer, and Aapo Hyvärinen. Equivalence of

some common linear feature extraction techniques for appearance-based object recognition tasks. *TPAMI*, 2007.

[Baldassi *et al.*, 2013] Carlo Baldassi, Alireza Alemi-Neissi, Marino Pagan, James J DiCarlo, Riccardo Zecchina, and Davide Zoccolan. Shape similarity, better than semantic membership, accounts for the structure of visual object representations in a population of monkey inferotemporal neurons. *PLoS Computational Biology*, 2013.

[Bergstra *et al.*, 2010] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. *SciPy*, 2010.

[Cadieu *et al.*, 2014] Charles F Cadieu, Ha Hong, Daniel LK Yamins, Nicolas Pinto, Diego Ardila, Ethan A Solomon, Najib J Majaj, and James J DiCarlo. Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS Computational Biology*, 2014.

[Ciresan *et al.*, 2012] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *CVPR*, 2012.

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

[Deselaers and Ferrari, 2011] Thomas Deselaers and Vittorio Ferrari. Visual and semantic similarity in imagenet. *CVPR*, 2011.

[Fellbaum, 1998] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. 1998.

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

[Girshick *et al.*, 2014] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

[Hardoon *et al.*, 2004] David Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 2004.

[Hinton *et al.*, 2014] Geoffrey E Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning Workshop*, 2014.

[Hotelling, 1936] Harold Hotelling. Relations between two sets of variates. *Biometrika*, 1936.

[Hyvärinen *et al.*, 2004] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. 2004.

[Hyvarinen, 1999] Aapo Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *TNN*, 1999.

[Jia *et al.*, 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.

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

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

[Larochelle *et al.*, 2008] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. *AAAI*, 2008.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[Long *et al.*, 2015] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

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

[Norouzi *et al.*, 2014] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *ICLR*, 2014.

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

[Shaoqing Ren, 2015] Ross Girshick Jian Sun Shaoqing Ren, Kaiming He. Faster R-CNN: Towards real-time object detection with region proposal networks. *NIPS*, 2015.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CVPR*, 2015.

[Turaga *et al.*, 2010] Srinivas C Turaga, Joseph F Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H Sebastian Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, 2010.