# Evaluation of Output Embeddings for Fine-Grained Image Classification

Zeynep Akata*, Scott Reed†, Daniel Walter†, Honglak Lee† and Bernt Schiele*

* Computer Vision and Multimodal Computing
Max Planck Institute for Informatics, Saarbrucken, Germany
† Computer Science and Engineering Division
University of Michigan, Ann Arbor

Image classification has advanced significantly in recent years with the availability of large-scale image sets. However, fine-grained classification remains a major challenge due to the annotation cost of large numbers of fine-grained categories. We show that compelling classification performance can be achieved on such categories even without labeled training data.

Following [1], given a specific input embedding, we derive a prediction by maximizing the compatibility $F$ over SJE as follows:

$$f(x; W) = \arg \max_{y \in Y} F(x; y; W) = \arg \max_{y \in Y} \theta(x)^{\top} W \varphi(y),$$

where $\theta(x)$ is the input embedding and $\varphi(y)$ is the output embedding. The matrix $W$ is learned by enforcing the correct label to be ranked higher than any of the other labels [7], the objective is:

$$\frac{1}{N} \sum_{n=1}^{N} \max_{y \in Y} \{0, \ell(x_n, y_n)\}.$$  \hspace{1cm} (1)

where $\ell(x_n, y_n) = \Delta(y_n, y) + \theta(x_n)^{\top} W \varphi(y) - \theta(x_n)^{\top} W \varphi(y_n)$. For zero-shot learning, we use $\varphi(y)$ of training classes and learn $W$. For prediction, we project $\theta(x)$ of test images onto the $W$ and search for the nearest $\varphi$ that corresponds to one of the test classes.

We use state-of-the-art image features [6] and focus on different super- and unsupervised output embeddings described in the following:

**Attributes** ($\varphi^{A}$) model shared characteristics of objects. For instance, for rat, monkey, whale and the attribute big, $\varphi^{A} = [0, 1, 0]$ $\rightarrow$ rat = monkey $<$ whale, whereas $\varphi^{A} = [2, 10, 90] \rightarrow$ rat $<$ monkey $<<$ whale.

**Word2Vec** ($\varphi^{W}$) [4] a two-layer neural network is trained to predict a set of target words from a set of context words. The first layer acts as a look-up table to retrieve the embedding for any word in the vocabulary. The second layer predicts the target word(s) via hierarchical soft-max. We use the skip-gram (SG) formulation where words within a local context window are predicted from the centering word.

**Glove** ($\varphi^{V}$) [5] incorporates co-occurrence statistics of words that frequently appear together within the document. The objective is to learn word vectors such that their inner product equals the co-occurrence probability of these two words.

**Weakly-supervised Word2Vec** ($\varphi^{WV}$) we pre-train the first layer weights using [4] on Wikipedia, and fine-tune the second layer weights using a negative-sampling objective [2] only on the fine-grained text corpus. These weights correspond to the final output embedding. The negative sampling objective is formulated as follows:

$$L = \sum_{w_c \in \mathcal{D}_c} \log \sigma(v_c^{\top} v_w) + \sum_{w_{nc} \in \mathcal{D}_{nc}} \log \sigma(-v_c^{\top} v_{w_{nc}}) \hspace{1cm} (2)$$

$$v_c = \sum_{i \in \text{context}(w)} v_i / |\text{context}(w)|$$

where $v_w$ and $v_{w_{nc}}$ are the label embeddings we seek to learn, and $v_c$ is the average of word embeddings $v_i$ within a context window around word $w$. $\mathcal{D}_c$ consists of context $v_c$ and matching targets $v_w$, and $\mathcal{D}_{nc}$ consists of the same $v_c$ and mismatching $v_{w_{nc}}$.

**Bag-of-Words** ($\varphi^{B}$) we collect Wikipedia articles that correspond to each object class and build a vocabulary of most frequently occurring words. We then build histograms of these words to vectorize our classes.

**Hierarchies** ($\varphi^{H}$) we measure the similarity between two classes by estimating the distance between terms in an ontology such as WordNet.

---

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage.

---

**Table 1:** Zero-shot learning results with SJE w.r.t. supervised and unsupervised output embeddings (Input embeddings: GoogLeNet [6])

<table>
<thead>
<tr>
<th>supervision</th>
<th>source</th>
<th>AWA</th>
<th>CUB</th>
<th>Dogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsupervised</td>
<td>text</td>
<td>51.2</td>
<td>28.4</td>
<td>19.6</td>
</tr>
<tr>
<td>unsupervised</td>
<td>text</td>
<td>58.8</td>
<td>24.2</td>
<td>17.8</td>
</tr>
<tr>
<td>unsupervised</td>
<td>text</td>
<td>44.9</td>
<td>22.1</td>
<td>33.0</td>
</tr>
<tr>
<td>unsupervised</td>
<td>WordNet</td>
<td>51.2</td>
<td>20.6</td>
<td>24.3</td>
</tr>
<tr>
<td>supervised</td>
<td>human</td>
<td>52.0</td>
<td>37.8</td>
<td>-</td>
</tr>
<tr>
<td>supervised</td>
<td>human</td>
<td>66.7</td>
<td>50.1</td>
<td>-</td>
</tr>
</tbody>
</table>

**Table 2:** Comparing SJE combined embeddings with SoA.

<table>
<thead>
<tr>
<th>supervised</th>
<th>method</th>
<th>AWA</th>
<th>CUB</th>
<th>Dogs</th>
</tr>
</thead>
<tbody>
<tr>
<td>unsupervised</td>
<td>SJE (cmb)</td>
<td>60.1</td>
<td>29.9</td>
<td>35.1</td>
</tr>
<tr>
<td>supervised</td>
<td>SoA [1]</td>
<td>73.9</td>
<td>51.7</td>
<td>-</td>
</tr>
</tbody>
</table>

---


