# A Genetic Algorithm for Combining Visual and Textual Embeddings Evaluated on Attribute Recognition

**Ruiqi Li**
Computer Science Department
KU Leuven
3001 Heverlee, Belgium
ruiqi.li1993@outlook.com

**Guillem Collell**
Computer Science Department
KU Leuven
3001 Heverlee, Belgium
gcollell@kuleuven.be

**Marie-Francine Moens**
Computer Science Department
KU Leuven
3001 Heverlee, Belgium
sien.moens@cs.kuleuven.be

## Abstract

We propose a genetic-based algorithm for combining visual and textual embeddings in a compact representation that captures fine-grain semantic knowledge—or attributes—of concepts. The genetic algorithm is able to select the most relevant representation components from the individual visual and textual embeddings when learning the representations, combining thus complementary visual and linguistic knowledge. We evaluate the proposed model in an attribute recognition task and compare the results with a model that concatenates the two embeddings and models that only use monomodal embeddings.

## 1 Introduction

Distributed representations of words (Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014; LeCun et al., 2015) in a vector space that capture the textual contexts in which words occur have become ubiquitous and been used effectively for many downstream natural language processing tasks such as sentiment analysis and sentence classification (Kim, 2014; Bansal et al., 2014). In computer vision, convolutional neural network (CNN) based image representations have become mainstream in object and scene recognition tasks (Krizhevsky et al., 2012a; Karpathy et al., 2014). Vision and language capture complementary information that humans automatically integrate in order to build mental representations of concepts (Collell and Moens, 2016). Certain concepts or properties of objects cannot be explicitly visually represented while, at the same time, not all the properties are easily expressible with language. Here, we assume that many properties of objects are learned by humans both by visual perception and through the use of words in a verbal context. For example, a cat has fur, which is visually observed, but from language this property can also be learned when speaking of the fur of this animal or of hairs that shake when moving. When building meaning representations of an object's attribute, combining visual representations or embeddings with textual representations seems beneficial.

In this paper we investigate *how* to integrate visual and textual embeddings that have been trained on large image and text databases respectively in order to capture knowledge about the attributes of the objects. We rely on the assumption that fine-grain semantic knowledge of attributes (e.g., shape, function, sound, etc.) is encoded in each modality (Collell and Moens, 2016). The results shed light on the potential benefit of combining vision and language data when creating better meaning representations of content. A first baseline model just concatenates the visual and textual vectors, while a second model keeps a compact vector representation, but selects relevant vector components to make up the representation based on a genetic algorithm, which allows capturing a mixture of the most relevant visual and linguistic features that encode object attributes. We additionally compare our model with vision-only and text-only baselines. Our contribution in this paper is as follows: To the best of our knowledge, we are the first to disentangle and recombine embeddings based on a genetic algorithm. We show that the genetic algorithm most successfully combines complementary information of the visual and textual embeddings when evaluated in an attribute recognition task. Moreover, with this genetic algorithm we learn compact and targeted

In: Mark Cieliebak, Don Tuggener and Fernando Benites (eds.): Proceedings of the 3rd Swiss Text Analytics Conference (SwissText 2018), Winterthur, Switzerland, June 2018

embeddings, where we assume that compact meaning representations are preferred over longer vectors in many realistic applications that make use of large sets of representations. Ultimately, this work provides insight on building better representations of concepts, which is essential towards improving automatic language understanding.

The rest of the paper is organized as follows. In the next Section we review and discuss related work. In Section 3 we describe the proposed genetic algorithm that combines visual and textual embeddings in encoding and classifying attributes, as well as a baseline method that concatenates the visual and textual embeddings and baseline vision-only or text-only models. Next, we present and discuss our experimental results. Finally, in conclusions and future work, we summarize our findings and suggest future lines of research.

## 2 Related Work

Representations of concepts are often task specific, where they mostly have been used in word similarity tasks. In this context integration of the visual and linguistic representations was realized by Collell et al. (2017); Lazaridou et al. (2015); Kiela and Bottou (2014); Silberer and Lapata (2014). Kiela and Bottou (2014) proposed the concatenation of visual and text representations, while Lazaridou et al. (2015) extend the skip-gram model to the multimodal domain, but none of these works regard attribute recognition. Silberer and Lapata (2014) obtain multimodal representations by implementing a stacked autoencoder with the visual and word vectors as input in an attribute recognition task. These vectors were separately trained with a classifier. In this work, we start from general pre-trained embeddings. Rubinstein et al. (2015) research attribute recognition by relying only on linguistic embeddings. Bruni et al. (2012) showed that the color attribute is better captured by visual representations than by linguistic representations. Farnadi et al. (2018) train a deep neural network for multimodal fusion of user's attributes as found in social media. They use a power-set combination of representation components in an attempt to better model shared and non-shared representations among the data sources which are composed of images, texts of and relationships between social media users. The closest work to ours is that of Collell and Moens (2016) who compare the performance of visual and linguistic embeddings each pre-trained on respectively a large image and text dataset for a large number of visual attributes, as well as for other non-visual attributes such as *taxonomic*, *function* or *encyclopedic*. In contrast to their work, we propose a model that integrates visual and linguistic embeddings, leveraging their findings in which they show that visual and linguistic embeddings encode complementary knowledge.

Genetic algorithms have been used for feature selection in text classification and clustering tasks (e.g., Abualigah et al. (2016); Gomez et al. (2017); Onan et al. (2017)), where the goal is to reduce the number of features. In this paper we continue this line of thinking for learning better multimodal embeddings.

## 3 Methodology

Given visual and textual embeddings of the same concept word but with different dimensionality, our goal is to combine the two embeddings so that the new embedding can capture both visual and textual semantic knowledge but with a more compact form than the concatenation representation. This section describes why and how we achieve this goal under the genetic algorithm (GA) framework.

### 3.1 Why Genetic Algorithms

When combining two embeddings, the instinctive idea naturally comes to mind is to check the meaning of each dimension in order to "pick" the dimensions that are really useful in a certain task. However, it has been a long term and highly debated issue in NLP that what exactly each dimension in the learned embeddings means to the whole representation. This is a work requires devoted observation and up till now there has been no final judgment on this topic. Back to the original goal, our final task is not to investigate the exact meaning of each dimension but to choose the dimensions that really help. This motivates us to use genetic algorithm which can provide numerous solutions and select them based on the natural selection principle. Specifically, the genetic operators such as crossover in genetic algorithm can be just used to vary the programming of embeddings from one generation to the next.

### 3.2 Genetic Algorithms Basic

Belonging to the larger class of evolutionary algorithms, genetic algorithms (GA) are meta-heuristics inspired by the process of natural selection. In a given environment, a population of individuals competes for survival and, more importantly, reproduction. The ability of each individual to achieve certain goals determines his or her chance of producing the next generation. In a GA setting, an *individual* is a solution with regard to the problem and the quality of the solution determines its *fitness*. The fittest individuals tend to survive and have children. By searching the solution space through the use of simulated evolution, i.e., following the survival of the fittest strategy, a GA achieves continuous improvement over the successive generations.

GA haven been shown to generate high-quality solutions to linear and non-linear problems through biologically-inspired operators such as mutation, crossover, and selection. A more complete discussion can be found in the book of Davis (1991). Algorithm.1 summarizes the procedure of a basic genetic algorithm.

**Algorithm 1** Framework of a Genetic Algorithm.

```
initialize population;
evaluate population;
while (!StopCondition) do
    select the fittest individuals;
    breed new individuals;
    evaluate the fitness of new individuals;
    replace the least fitted population;
end while
```

There are six fundamental issues to be determined to use a genetic algorithm: chromosome representation, initialization, the selection function, the genetic operators of reproduction, evaluation function, and termination criteria. The rest of this section describes the detail of these issues in creating a compact representation to capture fine-grain semantic visual and textual knowledge.

### 3.3 Chromosome Representation, Initialization, and Selection

The chromosome representation determines the problem structure and the genetic operators in a GA. The floating point representation of the chromosomes has been shown to be natural to evolution strategies and evolutionary programming (Periaux et al., 2015). One may point out that the pre-trained visual and textual embeddings can naturally be used as the original chromosomes since they consists of floating numbers. But recall that our goal is to form a compact embedding, and by "compact" we mean that the dimension of the final embedding should be smaller than the concatenation of the visual and textual embeddings. Due to the previous reason, we first concatenate visual and textual embeddings, then shuffle the dimensions in the concatenation, and divide the concatenation into two embeddings with the same dimension. Those two embeddings are used as the original chromosomes. Specifically, each real number in an embedding vector, representing a feature of the target concept, can be seen as a gene. In this way, the chromosome (embedding) is made up of a sequence of shuffled real numbers (floating points) which either comes from the original visual or textual embedding. Thus the two embeddings can be seen as a mixture of visual and textual knowledge with different degrees. For clarity, we henceforth use the term embedding instead of chromosome.

In a standard GA, the initial population is often generated randomly and the selection function is usually based on the fitness of an individual. However in our case, as explained previously, the initial population is formed by the original embeddings. Consequently, we make a change in the target of the selection function. Instead of trying to select the most fitted individuals to reproduce, the selection function first makes sure that every pair of visual and textual embeddings having the same target concept reproduce a group of candidates of the next generation, by repeating the reproduction method several times. The reproduction method involves randomly initialized parameters and will produce different children each time. Once a certain group of children candidates are generalized, they will compete against each other to survive but only the fittest one can win the opportunity of becoming the next generation. In this way, the fitness of the children generation is assured to be better than their parent generation and the fitness is guaranteed to improve over generations.

### 3.4 The Genetic Operators of Reproduction

Genetic operators determine the basic search mechanism and create new solutions based on existing ones

in the population. Normally there are two types of operators: crossover and mutation. Crossover takes two individuals and produces two new individuals while mutation alters one and produces one new. Since the embeddings used in our problem represent mappings from spaces with one dimension per concept word to continuous vector spaces with lower dimension, the value in each dimension of the embeddings characterizes the target concept in the vector spaces and should not be recklessly changed. Due to this reason, we only use the crossover operator to reproduce the next generation.

Recall that we now have two embeddings and each one can be seen as a mixture that combined visual and textual knowledge with different degrees. Our goal is to find all dimensions that help to achieve a certain goal. To test whether a certain dimension is relevant, the *crossover operator* is defined as follows: Let $X = (x_1, \cdots, x_n)$ and $Y = (y_1, \cdots, y_n)$ be two $n$-dimensional embeddings. The crossover operator generates two random integers $k, t$ from a uniform distribution from 1 to $n$, and creates two new embedding $X' = (x'_1, \cdots, x'_n)$, $Y' = (y'_1, \cdots, y'_n)$ according to:

$$x'_i = \begin{cases} x_i & \text{if } i \neq k \\ y_i & \text{otherwise} \end{cases} \quad (1)$$

$$y'_i = \begin{cases} y_i & \text{if } i \neq t \\ x_i & \text{otherwise} \end{cases} \quad (2)$$

As mentioned in the selection function, in one time of reproduction the same crossover operator is applied to all embeddings, producing one candidate of next generation. By repeating it a certain number of times, a group of different candidates is produced. We call such repetition a "reproduction trial".

### 3.5 Evaluation and Termination

Diverse evaluation functions can be used, depending on the specific tasks. For instance, for classification tasks the evaluation function can be any classification metric such as precision or Jaccard similarity score, as long as it can map the population into a partially ordered set. In regression, correlation is typically used as evaluation function. In our experiment, the F1 measure is used as the evaluation function. Generally, we use two types of F1 measure as the evaluation of fitness to avoid bias, one with respect to positive labels and the other negative labels. Section 4 shows the detail of how we use the F1 measure as evaluation function.

GA moves through generations, selecting and reproducing, until a specific termination criterion is met. From the point of view of reproducing, the stopping criterion can be set as a maximum number of generations reproduced. For example, the algorithm will stop once it reproduce 1000 generations. The first termination criterion is the most frequently used. The second termination strategy is a population convergence criteria that evaluates the sum of deviations among individuals. Third, the algorithm can also be terminated when a lack of improvement over a certain number of generations happens or, alternatively, when the value for the evaluation measure meets a target acceptability threshold. For instance, one can set as threshold if there is no improvement over a series of 10 times of reproduction, or if the fitness of the current generation is larger than the target threshold, then the algorithm terminates. Usually, several strategies can be used in conjunction with each other. In the experiments described below, a conjunction of the maximum number of generations reproduced in the first termination criterion and the maximum number of generations that allows a lack of improvement in the third termination criterion is used. Please noted here that a maximum number of generations reproduced in the first termination criterion and a certain number of generations that allows a lack of improvement are two different concepts. For example, if the former is set to 1000 while the latter 10, the algorithm will terminate when either 1) the algorithms reproduce 1000 generations; or 2) during the algorithm, there is no improvement in fitness 10 consecutive times of reproduction trials.

## 4 Experiments and Results

### 4.1 Experimental Setup

#### 4.1.1 Pre-trained Visual Embeddings

Following Collell and Moens (2016), we use ImageNet (Russakovsky et al., 2015) as our source of visual data. ImageNet is the largest labeled image dataset, and covers 21,841 WordNet synsets or meanings (Fellbaum, 1998) and over 14M images. We only preserve synsets with more than 50 images, and we set an upper bound of 500 images per synset for com-

putation time. After this, 11,928 synsets are kept. We extract a 4096-dimensional vector of features for each image as the output of the last layer of a pre-trained AlexNet CNN in Krizhevsky et al. (2012b). For each concept, we combine the representations from its individual images into a single vector by ***averaging*** the CNN feature vectors of individual images componentwise, which is equivalent to the cluster center of the individual representations.

#### 4.1.2 Pre-trained Word Embeddings

Following Collell and Moens (2016), we employ 300-dimensional GloVe vectors (Pennington et al., 2014) trained on the largest available corpus (840B tokens and a 2.2M words vocabulary from Common Crawl corpus) from the GloVe website[1].

#### 4.1.3 Dataset

The data set collected by McRae et al. (2005) consists of data gathered from 30 human participants that were asked to list properties—attributes—of concrete nouns. The data contains 541 concepts, 2,526 different attributes, and 10 attribute types.

| Attribute type | # Attr. | Avg. # concepts | SD |
|---|---|---|---|
| encyclopedic | 4 | 32.7 | 1.5 |
| function | 3 | 46 | 27.9 |
| sound | 1 | 34 | - |
| tactile | 1 | 26 | - |
| taste | 1 | 33 | - |
| taxonomic | 7 | 42 | 24.8 |
| color | 7 | 42.4 | 12.0 |
| form_and_surface | 14 | 63.7 | 29.9 |
| motion | 4 | 37.5 | 5.7 |

Table 1: Attribute types, number of attributes in each type (# Attr.), and average number of concepts in each type (Avg. # concepts) with their respective standard deviations (SD).

### 4.2 Attribute Recognition

To evaluate the composed embeddings, we assess how well the attributes from McRae et al. (2005) can be recognized by using the embeddings as input. For each attribute $a$, we build a data set with the concepts to which this attribute applies as the positive class instances and the rest of concepts form the negative class. For example, a "beetle" is a negative instance and "airplane" a positive instance for the attribute $a$ = *is_large*. And an "ant" is a negative instance and a "bear" is a positive instance for the attribute $a$ = *has_4_legs*. We consider that an attribute applies to a noun concept if a minimum of 5 people have listed it[2]. We treat attribute recognition as a binary classification problem: For each attribute $a$ we learn a predictor:

$$f_a : \mathcal{X} \rightarrow \mathcal{Y}$$

where $\mathcal{X} \subset \mathbb{R}^d$ is the input space of ($d$-dimensional) concept representations and $\mathcal{Y} = \{0, 1\}$ the binary output space. We report results with a linear SVM classifier, implemented with the scikit machine learning toolkit from Pedregosa et al. (2011).

To guarantee sufficient positive instances, only attributes with at least 25 positive instances in the above dataset are kept. This leads to a total of 42 attributes, covering 9 attribute types, and their corresponding instance sets. The concept selection in ImageNet described in Sect. 4.1.1 results in a visual coverage of 400 concepts (out of 541 from McRae et al. (2005) data), and, for a fair vision-language comparison, only the word embeddings (from GloVe) of these nouns are employed. Hence, our training data $\{(\overrightarrow{x_i}, y)\}_{i=1}^{400}$ consists of 400 instances. Table 1 shows the detail of each attribute type.

### 4.3 Parameter Setting

Notice that each reproduction operation will give birth to two forms of embedding. To avoid potential bias, we evaluate one embedding by F1 measure on the positive labels and the other on the negative labels. The average of these two F1 measures can be an option to evaluate fitness. However, in practice, the negative labels are more numerous than the positive ones. An increase of the F1 measure on the negative labels while a decrease of the positive ones can still result in an increase on the average F1 measure. Thus, we use the F1 measure on the positive labels as the first measure of fitness and the F1 measure on the negative labels as the second. Only the one with largest increase in the first F1 measure and largest increase or at least non-decrease in the second F1 measure will be chosen as the next generation.

The maximum number of generations reproduced is set to $10^6$. The maximum number of reproduction trials in case a lack of improvement among the

[1] http://nlp.stanford.edu/projects/glove

[2] This threshold was set by McRae et al. (2005)

children candidates is 10. The repeat times of the crossover operation in one reproduction trial is $10^3$. And in the final evaluation of each embedding on the attribute recognition task, we perform 5 runs of 5-fold cross validation.

We evaluate four different embeddings as input of the attribute recognition task: 1) Embeddings of the concept obtained with the GA described above (GeMix); 2) Embeddings obtained by concatenating the visual and textual embedding vectors (CON); 3) Monomodal visual embeddings (CNN); and 4) Monomodal text embeddings (GloVe). Table 2 shows the number of dimensions of each embedding respectively.

| | # dimensions |
|---|---|
| CNN | 4096 |
| Glove | 300 |
| CON | 4396 |
| GeMix | 2198 |

Table 2: Dimensionality of each embedding type.

### 4.4 Result and Discussion

#### 4.4.1 Performance per Attribute Type

We first evaluate how the proposed method performs on each attribute type. There are 9 attribute types and we evaluate the four embeddings for each type by with the average F1 measure.

From Table 4 one can see that the GeMix embeddings outperform the other three embedding methods in 7 attribute types, i.e., encyclopedic, function, tactile, taste, taxonomic, color and form_and_surface. Especially in encyclopedic, GeMix increases the average F1 measure by more than 0.02 and in function and taxonomic, it increases by nearly 0.02. We perform Wilcoxon Signed-Rank test of each two methods on different feature sets and find that the difference is significant at $p \leq 0.05$.

Another interesting finding is that the performance of the concatenated embedding (CON) is not always better than the performance of the monomodal embeddings, CNN or GloVe. For instance, in tactile, color and motion, the F1 measure of CNN or GloVe is higher than that of concatenated embeddings. This indicates that there are certain attributes in which the performance of combined visual and textual knowledge is not necessarily better than unimodal visual or textual knowledge. This will further be discussed in Section 4.4.2.

#### 4.4.2 Performance per Attribute and Overall

Table 5 provides a more detailed answer to our question, showing that GeMix outperforms the other three embeddings in 20 attributes while CNN performs best in 6 attributes, GloVe in 7 attributes and the concatenated embedding (CON) in 9 attributes. Specifically, GeMix outperforms the second best method with more than 0.04 in attributes 04 (*lays_eggs*), 09 (*is_soft*), 12 (*a_vegetable*), and 14 (*a_mammal*) and 0.10 in 30 (*has_a_beak*) and 37 (*made_of_wood*).

| | $F1_{attr}$ | $F1_{samp}$ |
|---|---|---|
| CNN | 0.535 | 0.469 |
| Glove | 0.552 | 0.474 |
| CON | 0.572 | 0.495 |
| GeMix | **0.586** | **0.507** |

Table 3: Overall F1 measure per attribute and per sample.

According to Collell and Moens (2016), visual embeddings perform better than textual ones when recognizing three main attributes: *motion, form_and_surface, and color*, while textual embeddings (GloVe) outperform the visual CNN embeddings in recognizing *encyclopedic* and *function* attributes. A closer look at Table 5 further reveals that for attribute types where vision or language embeddings show better performance over the other one, it is high likely that adding respectively language or vision information lower the performance, e.g., attribute 05 (*hunted_by_people*), 07 (*used_for_transportation*) in *function* and 20 (*is_fast*), 21 (*eats*) in *motion*. Because GeMix tend to set aside the "noisy" dimensions of the embeddings, it performs better than the concatenated embedding.

Let us take a look at the overall average F1 measure increase. We evaluate the F1 measure with respect to two different aspects. First, the overall average F1 measure per attribute, i.e., $F1_{attr} = \frac{1}{|L|}F1_L$ where $|L|$ is the number of different attributes (42 in our case) and $F1_L$ is the F1 measure of a specific attribute. Second, the overall average F1 measure per sample, i.e., $F1_{samp} = \frac{1}{|S|}F1_S$ where $|S|$ is the number of samples (400 in our case) and $F1_L$ is the F1 measure of each sample. Table 3 shows that in both cases, GeMix achieves the highest F1 measure.

| | Encyc | Funct | Sound | Tactile | Taste | Taxon | Color | Form&Surf | Motion |
|---|---|---|---|---|---|---|---|---|---|
| CNN | 0.429 | 0.738 | 0.513 | 0.470 | 0.421 | 0.486 | 0.676 | 0.567 | **0.628** |
| GloVe | 0.422 | 0.743 | 0.747 | 0.517 | 0.341 | 0.495 | 0.630 | 0.563 | 0.595 |
| CON | 0.457 | 0.760 | **0.762** | 0.477 | 0.433 | 0.512 | 0.663 | 0.582 | 0.623 |
| GeMix | **0.471** | **0.786** | 0.758 | **0.520** | **0.438** | **0.528** | **0.671** | **0.588** | 0.620 |

Table 4: Performance per Attribute Type: Averages of F1 measures per attribute type (i.e., average individual attributes) for CNN, GloVe, CON and GeMix.

| | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CNN | 0.484 | 0.580 | 0.355 | 0.591 | 0.410 | 0.682 | 0.794 | 0.663 | 0.261 | 0.418 | 0.591 |
| GloVe | 0.463 | 0.611 | 0.521 | 0.591 | **0.486** | 0.602 | **0.971** | 0.701 | 0.551 | 0.311 | 0.668 |
| CON | 0.462 | **0.673** | **0.576** | 0.626 | 0.456 | 0.723 | 0.944 | **0.765** | 0.547 | 0.437 | 0.683 |
| GeMix | **0.502** | 0.661 | 0.570 | **0.672** | 0.466 | **0.734** | 0.944 | 0.717 | **0.605** | **0.439** | **0.690** |
| | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| CNN | 0.460 | 0.284 | 0.632 | 0.405 | 0.431 | 0.422 | 0.439 | 0.581 | **0.915** | **0.527** | 0.812 |
| GloVe | 0.292 | 0.233 | 0.628 | **0.491** | **0.484** | **0.530** | 0.320 | 0.617 | 0.822 | 0.510 | 0.622 |
| CON | 0.522 | **0.321** | 0.641 | 0.443 | 0.471 | 0.522 | 0.466 | 0.603 | 0.846 | 0.510 | 0.773 |
| GeMix | **0.565** | 0.228 | **0.702** | 0.475 | 0.437 | 0.476 | **0.475** | **0.651** | 0.863 | 0.524 | **0.822** |
| | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
| CNN | 0.513 | 0.643 | **0.884** | 0.699 | **0.647** | 0.544 | **0.727** | 0.325 | 0.489 | 0.649 | **0.738** |
| GloVe | 0.347 | 0.595 | 0.743 | 0.668 | 0.379 | 0.448 | 0.437 | 0.313 | 0.495 | **0.767** | 0.672 |
| CON | 0.476 | **0.668** | 0.852 | **0.728** | 0.640 | **0.558** | 0.433 | 0.298 | 0.503 | 0.722 | 0.651 |
| GeMix | **0.546** | 0.660 | 0.829 | 0.704 | 0.639 | 0.548 | 0.414 | **0.476** | **0.512** | 0.744 | 0.734 |
| | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | | |
| CNN | 0.580 | 0.421 | 0.372 | 0.532 | 0.906 | 0.506 | 0.748 | 0.421 | 0.418 | | |
| GloVe | 0.444 | 0.440 | 0.368 | 0.345 | **0.970** | 0.522 | 0.543 | 0.415 | 0.291 | | |
| CON | **0.622** | 0.548 | 0.377 | 0.547 | 0.888 | 0.570 | 0.784 | **0.483** | **0.539** | | |
| GeMix | **0.622** | **0.562** | **0.387** | **0.670** | 0.900 | **0.573** | **0.791** | 0.468 | 0.500 | | |

Table 5: Performance of the attribute classification task per attribute in terms of F1 measure for each embedding method. Attribute 01 - 04 belong to encyclopedic, 05 - 07 function, 08 sound, 09 tactile, 10 taste, 11 - 17 taxonomic, 18 - 21 motion, 22 - 28 color and 29 - 42 form_and_surface.

## 5 Conclusion and Future Work

In this paper, we propose a genetic-based algorithm which learns a compact representation that combines visual and textual embeddings. Two embeddings, coming from random evenly divide of the shuffled concatenation of vision and textual embeddings, are used as the initial chromosomes in the genetic algorithm. A variant of one-point crossover method is used to move the most relevant components in the representation to one embedding, and the non-relevant ones to the other. To avoid bias, we use two measures as the evaluation of fitness: one is respect to positive labels and the other negative labels. The learned embeddings can be seen as a combination of both visual and textual knowledge. In an attribute recognition task the genetic-based representation outperformed a baseline composed of the concatenation of the visual and textual embeddings, as well as the monomodal visual or textual embedding.

Another interesting finding in this paper is that for a small group of attributes in which either vision or language generally dominate, adding the other modality may lower the final performance. For example, the attribute *eats* in the *motion* type for which vision tends to perform better than language (Collell and Moens, 2016), the performance of the mixture of both visual and textual representation is lower than the monomodal visual representation. Ultimately, our findings provide insights that can help building better multimodal representation by taking into account to

what degree should the visual and textual knowledge be mixed with respect to different tasks.

## References


L. M. Abualigah, A. T. Khader, and M. A. Al-Betar. 2016. Unsupervised feature selection technique based on genetic algorithm for improving the text clustering. In *2016 7th International Conference on Computer Science and Information Technology (CSIT)*. volume 00, pages 1–6.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *ACL (2)*. pages 809–815.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *ACL*. ACL, pages 136–145.

Guillem Collell and Marie-Francine Moens. 2016. Is an image worth more than a thousand words? On the fine-grain semantic differences between visual and linguistic representations. In *COLING*. ACL, pages 2807–2817.

Guillem Collell, Ted Zhang, and Marie-Francine Moens. 2017. Imagined visual representations as multimodal embeddings. In *AAAI*. pages 4378–4384.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Lawrence Davis. 1991. Handbook of genetic algorithms .

Golnoosh Farnadi, Jie Tang, Martine De Cock, and Marie-Francine Moens. 2018. User profiling through deep multimodal fusion. In *WSDM*. pages 171–179.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Juan Carlos Gomez, Stijn Hoskens, and Marie-Francine Moens. 2017. Evolutionary learning of meta-rules for text classification. GECCO.

Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *CVPR*. IEEE, pages 1725–1732.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *EMNLP*. pages 36–45.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012a. Imagenet classification with deep convolutional neural networks. In *NIPS*. USA, pages 1097–1105.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012b. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. pages 1097–1105.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. *arXiv preprint arXiv:1501.02598* .

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521(7553):436–444.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior research methods* 37(4):547–559.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Aytug Onan, Serdar Korukoglu, and Hasan Bulut. 2017. A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification. *Inf. Process. Manage.* 53(4):814–833.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Jacques Periaux, Felipe Gonzalez, and Dong Seop Chris Lee. 2015. *Evolutionary optimization and game strategies for advanced multi-disciplinary design*. Springer Netherlands.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *ACL*. volume 2, pages 726–730.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *IJCV* 115(3):211–252.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *ACL*. pages 721–732.