
Computational Divergent Thinking Using Visual Data

By

LOTAN LEVY

Under the supervision of

PROF. DAPHNA WEINSHALL AND PROF. DAFNA SHAHAF



THE HEBREW
UNIVERSITY
OF JERUSALEM

Faculty of Computer Science and Engineering
THE HEBREW UNIVERSITY OF JERUSALEM

A dissertation submitted to the Hebrew University of
Jerusalem as a partial fulfillment of the requirements
of the degree of MASTER OF SCIENCE in the Faculty of
Computer Science and Engineering.

DECEMBER 2021

ABSTRACT

The goal of the artificial intelligence field is to develop machines that mimic human-like intelligence abilities to solve problems using computational models. In this research, we focus on one ability, known as divergent thinking ability, which is the ability to think in different directions and produce a large amount of new and diverse solutions for a given problem. Our goal is to create a model that uses this ability to solve affordance problems, which are problems that relate to use. Legitimate solutions for these affordance problems are instances of images that allow a given use. During this research, we examined several approaches to create the divergent thinking ability in computational models and adjusted state-of-art models for this purpose. Across several divergent thinking tasks, and several evaluation paradigms, we show that the adjusted Ballpark model surpasses all the other models we examine. Furthermore, we demonstrate the model supremacy when learning from training datasets with a lot of misclassified instances.

TABLE OF CONTENTS

	Page
List of Tables	v
List of Figures	vii
1 Introduction	1
1.1 Problem Formulation	3
1.2 Contributions and Outline	5
2 Related Work & Background	7
2.1 Background - Transfer Learning	7
2.2 Background - Support Vector Machine (SVM)	8
2.3 Background - One Class Classification	10
2.4 Background - Positive Unlabeled Learning (PU Learning)	12
2.5 Background - Weakly Supervised Learning	13
3 Our Method	15
3.1 Input preparation	15
3.2 Solving the Ballpark Model for Divergent Thinking Problems	16
3.3 Iterative Labeling Procedure	18
4 Experiments and Results	21
4.1 Methodology	21
4.2 Experimental Protocol	25
4.3 Results	28
4.4 Ablation Study	36
4.5 User Study	38
5 Summary and Conclusions	43

TABLE OF CONTENTS

5.1 Summary and Discussion	43
A Appendix	45
Bibliography	49

LIST OF TABLES

TABLE	Page
4.1 The AUC evaluations for the Dine, Shop, Stab and Flowerpot problems. The PSVM column reports the total average and the standard deviation (in parentheses) of the set of the Polar SVM models described in section 4.2. The supervised SVM (SSVM) and the Deep-One-Class Classification (DOC) columns report the average and the standard deviation (in parentheses) of 5 similar experiments of the corresponding models.	29
4.2 The AUC results in each iterative labeling step, for both the Stab and the Shop tasks.	33
4.3 The additional constrained bags in each iterative labeling step. When the name of the bag is a subtraction between bags, it means that the constraint is a Difference constraint between the two bags. The rows of the table are the tasks names, and the columns are the procedure steps.	33
4.4 10 best images with the highest scores according to the Ballpark model for the Architecture, Shop and Stab tasks. The images are sorted from the highest scoring image on the top to the lowest on the bottom. The first column displays the class label and the second displays two images selected randomly from the corresponding class.	34

LIST OF TABLES

4.5	The AUC evaluations of the Ballpark model on the Dine, Shop, Stab, and Flowerpot problems by using only Difference constraints. The first row is the results of the ballpark model when training only by the Difference constraints calculated automatically from the bounds constraint in Table A.1. The second row of the table displays the results of the Ballpark model with the intuitive Difference constraints. The third row is the Ballpark results from the Table 4.1, and the last row is the best result achieved by the baseline in Table 4.1. The square brackets indicate that the score in that cell, achieved by averaging the results of several models, and the number in the brackets is the standard deviation of these results.	37
4.6	Evaluate the model results using the ground truth labels collected in the surveys.	39
4.7	The model mistakes according to the survey. The first row consists of the classes that were considered as positive in the surveys but were not selected by our model. The second row displays the classes that our model considered as positive but weren't selected in the survey.	39
A.1	The bags constraints used for train the models in most of the experiments in this work. Each model received also differences constraints, calculated from these bags bounds automatically.	46
A.2	The number of instances in the constraints bags in A.1.	46
A.3	The number of instances in the test dataset in A.1.	46
A.4	The classes division according to the ranking of the model and the surveys' first part results	47

LIST OF FIGURES

FIGURE	Page
3.1 Summary of the Ballpark’s training procedure. In the first three steps, we prepare the input for training by splitting the training instances into bags, determining constraints on these bags, and selecting a good features extraction method for the representation of the images. Then, we solve the Ballpark optimization problem on these training data according to the constraints. The model outputs a vector of weights that determine which features are important for the required task.	18
4.1 Summary of the impact of using polar supervision in the Polar Ballpark and the Polar SVM models on the Dine, Shop, Stab, and Flowerpot tasks. The labels of the x-axis are the polar thresholds pair p_l, p_h , that were used to split the training data of corresponding Polar Ballpark and Polar SVM models, into positive and negative instances according to the task’s constraints file. The height of the bars represents the AUC score of each model. Each pair of Polar Ballpark and Polar SVM models was trained with the same labeled instances according to the polar threshold, and the Polar Ballpark model was also trained by the constraints file of the task. We also displayed in a dashed line a standard Ballpark model that was trained by the task’s constraints file without any additional labeled data.	30
4.2 The impact of noisy training data on the Ballpark and the Polar SVM models with the different threshold values are described in the methodology section. The parameter p is the ratio of the ground truth data and the noisy data in each bag.	32

LIST OF FIGURES

4.3	Demonstrates the results from the second part of the survey for the instances ranking. For each task we demonstrate the models ranking by one class. For the Architecture task we use images of the Balcony class, for the Shop task we use images of the Game room class and for the Stab task we use images of the Letter opener class. For each class we display the five images with the highest scores and the five images with the lowest scores sorted order according to the model's scores. The scores received by the model are displayed below the corresponding images.	35
4.4	Comparisons between the Ballpark, the Polar SVM with the different polar thresholds, and the DOC models using different features spaces. The plots report the AUC results of the models on two common features extraction methods. The black lines are the standard deviation of the Polar SVM and the DOC models. The right plot is the results of the models when using a ResNet50 architecture, and the second is for the VGG16 architecture.	37
4.5	Illustration of the user study's results for three different classes, each class evaluate according to different affordance task. The x-axis is the scores that our model gave to each of the class's images. The y axis is the participants' average score for the corresponding image. The black line in each bar is the standard deviation of the participants' scores. The blue bars represent the best images according to the model ranking, and orange bars represent the images with the lowest scores according to the model.	41
4.6	summaries the results of the user study second part. Each plot represents a different task. For each class we display two bars one for the positive images (that are the images that received the best model's scores in the class) and one for the negative images (The images that received the lowest model's scores in the class). The height of the bars indicates the average survey's scores, and the black lines are the standard deviation of these scores.	42

INTRODUCTION

Over the last few decades, an immense amount of research has been devoted to mimicking human intelligence in machines to solve computational problems. One common problem that is solved using this approach is the problem of producing new information. In the 1950s a famous study was made by the American psychologist, Guilford, in the area of human intelligence [5]. The study analyzed the intellectual abilities required to perform intellectual tasks and proposed two kinds of thinking abilities that are keys for generating new information from already known information. The first ability is the *Divergent-Thinking Ability* that is the ability to think in different directions and to produce a variety of responses that are not completely determined by the given information. The second is the *Convergent-Thinking Ability* that is the ability to produce one right answer that is the best for given information, and in some way opposed to the previously mentioned ability.

In this research, we focus on the divergent thinking ability, which is demonstrated in The Alternative Uses Test proposed by Guilford in 1967 in the paper [6]. In this test, participants were asked to think of as many uses as possible for a given object, and their answers were analyzed according to two essential divergent thinking properties - fluency and flexibility. Participants who replied with a lot of ideas in a short time were considered to have fluent thinking and if their answers also included a variety of ideas from different classes, they were also considered to have flexible thinking. For example, given the object 'Knife', participants whose responses are cut tomato, cut meat, cut onion,

cut cheese, cut ginger, etc. would get a high fluency score and low flexibility scores since all the proposed suggestions fall into the same class. On the other hand, participants whose answers are cut vegetables, open a letter, stab a person, engrave a tree, and so on, will also get high flexibility scores. It is noteworthy that the convergent thinking ability may perform poorly in achieving flexible solutions, and the divergent thinking ability is essential for this purpose. Nowadays, there is a lot of research in the field of artificial intelligence that achieves some sort of creative behavior such as artificial art, design, and game solving. Nevertheless, those methods do not require having the flexible ability, but only the ability to generate new solutions.

In our work, we propose a new challenge for artificial intelligence machines: Create a model that solves problems that require divergent thinking ability using images. We evaluated our model on several affordance tasks, which are tasks that relate to a specific use, and are solved by generating solutions that allow this use. For example, the Stab task is a task that is solved by suggesting objects that can be used for stabbing. Legitimate solutions for this task can be a knife or a needle. We are particularly interested in finding innovative and creative solutions. For instance, in the Stab task, innovative solutions can be an umbrella with a sharp tip or a horn of a unicorn.

The affordance term was coined by Gibson in 1986 in [4] and defined as what the environment provides or furnishes the subject, meaning the uses that a subject affords according to its visual properties. For example, the knife object affords stabbing, engraving, and cutting uses. In our work, we use the affordance term in the opposite direction. Instead of providing uses that are afforded by a given subject, the task is to find subjects that afford a given use. In his book, Gibson addresses numerous affordance types, nevertheless, we focus on the *place affordance type* that relates to places that afford a required use, and the *objects affordance type* that relates to objects that afford the use.

Our main assumption for solving an affordance task is that the exact visual features that are needed for a given utilization are unknown. Therefore, it is a difficult task to label examples as negative, since these examples can be innovative solutions and as we mentioned, we are especially interested in providing these novel solutions. Therefore, in our research, we explore approaches that avoid providing negative examples. Furthermore, since the available examples are static images, we can't be certain that all the required features for the affordance task appear in a given image, even if it includes an object that affords the desired use since we don't know what the required features are. For example, given an image of a knife with a concealed blade, we can't be sure

that it allows stabbing because the blade is an essential condition for stabbing. Those providing positive examples can also mislead the model. In the results section, we show that the model which achieved the best results on our affordance tasks, allows us to avoid providing positive examples as well. This model is the Ballpark model suggested in [8] and [9] for problems that are not based on visual data. During our work, we adjusted the model to learn from visual data and utilize its ability to learn from coarse-grained labels on groups of images to avoid providing labels for the instances. In chapter 4, we show that this approach achieves the best performance on affordance tasks and seems to be very effective when the training data contains a lot of misclassified data.

1.1 Problem Formulation

In this work, we are interested in the task of determining if a given image describes a place or an object that complies with an affordance task. More formally, given an image x that can be converted into a set of representative visual features, we aim to predict a binary label $y \in \{0, 1\}$, where a positive label indicates that the given image affords the required use and the negative label indicates that it doesn't. We use a features extraction function $\varphi(X)$ to convert a given image to its representative features.

We assume that we have a set of available examples $X = \{x_1, \dots, x_n\}$, Where x_i has a corresponding binary label y_i .

One reasonable approach to avoid negative examples is to learn only by using positive data. As we explained previously, these positive data may include instances with missing features that are essential for determining if the instance is positive. Those we are assuming that the positive examples include an unknown amount of misclassified examples. The learning approach that allows learning only from positive examples is the *one-class classification* approach. To solve our problem using this approach, we use the examples labeled as positive from the dataset. In this setting, our available examples are $D = \{(x_1, 1), \dots, (x_n, 1)\}$, where there is an unknown amount of instances in D that are negative. In our research, we assumed that the amount of misclassified instances in the training data is negligible and we implemented the one-class classification model proposed in [12] to analyze the approach on our task. While achieving fair results on standard one-class classification tasks, such as anomaly detection tasks, the model does not yield good results when testing on our affordance tasks as we show in chapter 4.

Another possible approach to avoid negative examples during the training is the *Positive Unlabeled [PU] learning* approach that uses positive and unlabeled data to learn

to distinguish between positive and negative instances. In this setting, our available examples are $D = \{(x_1, 1), \dots, (x_n, 1), x_{n+1}, \dots, x_{n+k}\}$, where there is an unknown amount of positive instances in D that are negative, and x_{n+1}, \dots, x_{n+k} are all the instances which their classification is uncertain and therefore their labels are unknown. To learn from these unlabeled data the PU learning methods require making assumptions on the distribution of the data or on the probability to select positive examples to be labeled from all the positive examples in the training dataset. In our work, we avoided making assumptions on the data distributions, since it may restrict the creative space of our model. Hence, we did not investigate PU-learning methods on our problem.

The last approach we examine is a *weakly supervised learning* approach. The approach allows us to avoid labeling negative examples as well as overcome the problem of providing misclassified positive examples to our model. Instead, it provides only some coarse-grained labels on groups of instances. The motivation of using this knowledge to train a model derives from the fact that we have some intuition about the compliance of objects or places with an affordance task, and we can use it to determine coarse-grained labels on classes instead of labeling the individuals. For example, given a random image from a set of knives images, we can assume in a high probability that the image describes an item that allows the stabbing use. On the opposite side, a random image from a set of pillows images will probably not allow stabbing. Furthermore, we can also assume that images from the knife set are probably more suitable for stabbing than the images from the pillows set.

In this setting, the labels of the examples are unknown $D = \{(x_1, y_1^*), \dots, (x_n, y_n^*)\}$. To use this approach for our settings, we assume that we are given a division of the instances into *bags* according to the content of the images and a set of constraints on the positive labeling proportion in each bag. Formally, we are given the set of bags $B = \{B_1, B_2, \dots, B_K\}$, $B_i \subseteq X$ that defines a division on the instances in D according to the content of the images. Let $p : \mathcal{B} \rightarrow \mathbb{R}$ the positive labeling proportion function, where $p(B_i)$ is the positive labeling proportion in bag B_i . In our problem, we don't know p and instead, we are given a set of constraints on the values that p outputs for each bag. Our goal is to use the set of constraints S on the instances in D to train a model to predict the individual labels that relate to the compliance of the individuals with an affordance task.

1.2 Contributions and Outline

Our Contribution in this study is as follows:

I) We defined a new problem in computer vision and constructed datasets to evaluate it.

II) We proposed two reasonable approaches to handle this new problem and analyzed the performance of each approach by adjusting existing methods, one for each approach.

III) We adapted the Ballpark framework to visual object recognition from images, where bags are defined by auxiliary class labels. Beneficially, bag constraints are obtained from common perceptions about objects, replacing the laborious annotation of individual images.

IV) Adapted two evaluation paradigms to evaluate divergent thinking. **V)** Investigated and demonstrated the good performance of the Ballpark model as compared to the SVM method when learning from misclassified data.

RELATED WORK & BACKGROUND

In this chapter, we give a solid background of the topics discussed throughout our work. We survey extensively the different approaches we examined for solving divergent thinking problems. For each approach, we present a related work we adjusted to solve our problem.

2.1 Background - Transfer Learning

Transfer learning is a machine learning problem that utilizes knowledge learned while solving one task in learning a new different but related task. This approach is motivated by the fact that humans can intelligently apply previously-learned knowledge to solve new problems. This methodology is significantly useful when there is no sufficient training data or the training is time-consuming. The survey in [11] presents three types of transfer learning categories *transductive*, *inductive*, and *unsupervised transfer learning*. All these categories transfer knowledge from a source task to a target task and differ by the label-setting of the source and the target datasets. In this work, we focus on inductive transfer learning that assumes that both the labels of the source and the target domains are available. The formal definition of inductive transfer learning is:

Given a source domain \mathcal{D}_S and a source learning task \mathcal{T}_S , a target domain \mathcal{D}_T and a target learning task \mathcal{T}_T , inductive transfer learning aims to improve the learning of the target predictive function $f_T(\cdot)$ in \mathcal{D}_T using the knowledge in \mathcal{D}_S and \mathcal{T}_S , where $\mathcal{T}_S \neq \mathcal{T}_T$.

In this type of learning, we assume that a lot of labeled data from the source domain

is available, thus we can train a model for solving a source task and then transfer the learned knowledge to learn the target task. The survey also presents four transferred knowledge types. In this work, only the feature representation type is relevant. The idea of transferring this type of knowledge is to learn a good feature representation for the source task, and then transfer this knowledge to the target task. In neural networks, the learned features are transferred by copying the learned weights of some of the layers in the source neural network into the matching layers in the target neural network, that learn the target task.

There are two common approaches to use the transferred features in the target task. The first is to fine-tuned the learned features according to the target task, meaning backpropagate the errors from the target task into the transferred features. The second is to leave the transferred feature layers frozen, meaning that they do not change during the training of the target task. When the target dataset is very small and the number of parameters in the target network is large, fine-tuning may result in overfitting. Therefore, in this case, it is better to leave the transferred features frozen. On the other hand, if the target dataset is large enough, the model will not overfit and we can fine-tune the transferred features as well.

In the field of image classification it is common to use pre-trained CNN models as source networks, such as *VGG16* [14] or *ResNet50* [7] that are already trained on the *ImageNet* [3] dataset, that is a large annotated dataset of images. These models were built from scratch and trained by using supercomputers over millions of images consisting of many image categories. Hence, they possess a good feature representation in their deep layers. This representation can be repurposed and utilized for different computer vision problems with different images and new classes.

2.2 Background - Support Vector Machine (SVM)

The Support Vector Machine model, proposed by Vapnik et al. in 1995 [2], is a supervised machine learning algorithm that aims to learn linear predictors in high dimensional features space. The method's objective is to find a hyperplane with a maximal margin that separates the training data points into two classes.

Formally, a hyperplane defined by the set $L = \{v : \langle w, v \rangle + b = 0\}$ where the two parameters (w, b) are the hyperplane parameters. Given a point x , the distance of x from L is defined by $d(x, L) = \min\{\|x - v\| : v \in L\}$, and if $\|w\| = 1$ it sustains $d(x, L) = |\langle w, x \rangle + b|$.

Let $S = (x_1, y_1), \dots, (x_m, y_m)$ be a training set of examples, where $x_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$.

A separating hyperplane that separates the data into two classes, is defined by (w, b) s.t $\forall i, y_i(\langle w, x_i \rangle + b) > 0$ and the margin of this hyperplane defined to be the distance to the closest point, meaning $\min_i |\langle w, x_i \rangle + b|$. We can formulate the problem using the optimization problem formalism that is also called the *Hard-SVM* rule:

$$\operatorname{argmax}_{(w,b):\|w\|=1} \min_{i \in [m]} |\langle w, x_i \rangle + b| \quad \text{s.t} \quad \forall i, \quad y_i(\langle w, x_i \rangle + b) > 0$$

Whenever there is a solution to the preceding problem, an equivalent formulation is to solve:

$$(w_0, b_0) = \operatorname{argmin}_{(w,b)} \|w\|^2 \quad \text{s.t} \quad \forall i, \quad y_i(\langle w, x_i \rangle + b) > 1$$

$\hat{w} = \frac{w_0}{\|w_0\|}$, $\hat{b} = \frac{b_0}{\|w_0\|}$ are the parameters of a separating hyperplane with the largest margin according to the Hard-SVM rule. As developed in the book [13], if the training set is not linearly separable, we can relax the constraints of the Hard-SVM rule and yield the following optimization problem:

$$\operatorname{argmin}_{w,b,\xi} (\lambda \|w\|^2 + \frac{1}{m} \sum_{i=1}^m \xi_i) \quad \text{s.t} \quad \forall i, \quad y_i(\langle w, x_i \rangle + b) > 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0$$

This relaxed version is called Soft-SVM and can be written as the following regularized loss minimization problem:

$$\min_{w,b} (\lambda \|w\|^2 + L_S^{\text{hinge}}((w, b)))$$

where $L_S^{\text{hinge}}((w, b)(x, y)) = \max\{0, 1 - y_i(\langle w, x_i \rangle + b)\}$.

As explained in [13], one of the major advantages of the SVM models is the low sample complexity, meaning these models can achieve a fair result using a small training dataset. Generally, the sample complexity of algorithms that learn separating hyperplanes in \mathbb{R}^d grows with the value of d , that is the VC-dimension of these algorithms. From the fundamental theorem of learning it follows that if the size of the training dataset is significantly smaller than d/ϵ , no algorithm can learn an ϵ -accurate halfspace. In contrast, as proved in [13], if the data is separable, the sample complexity of both Hard SVM and Soft SVM algorithms depends on $\rho^2 B^2$, where ρ is the norm of the examples, and B is the norm of the halfspace. In the non-separable case, the sample complexity of these models also depends on the minimum hinge loss of all halfspaces with a norm greater or equal to B . This becomes significantly helpful when we learn in high dimensional feature spaces, namely $d \gg \rho^2 B^2$, since it allows the model to learn from smaller datasets. The second advantage is that the problem formulations for both SVM versions are convex and therefore have an optimal solution. We recall that using hinge loss in soft SVM comes with a penalty of high sensitivity to outliers.

2.3 Background - One Class Classification

One class classification is a classical machine learning problem that tries to recognize instances of the specific concept among all instances by primarily learning from training data that only contains instances of the same concept. During inference, the classifier's goal is to distinguish among instances that belong to the known concept and those that are not. It is notable that one class classification problem is different from binary classification problems due to the absence of negative instances, namely instances that are not from the known concept. There are many real-world one-class classification applications, for example, outlier detection, anomaly detection, and novelty detection. Several different approaches have been proposed to solve this problem in the literature. One popular approach estimates parametric generative models by assuming distribution such as Gaussian distribution [10]. This approach, for example, was used in the work of Kemmler [10] which estimates the conditional density of the distribution of the concept class using Gaussian priors. The methods that use this approach work best when many samples are present. Since in the divergent thinking problems we usually have a relatively small amount of positive examples, this approach is less preferable. Another approach is to separate the one-class data from the out-of-class data using boundary methods. One of the methods that uses this approach is the Support Vector Data Description (SVDD) algorithm that separates one-class data from the rest of the data using a spherical separation plane [15]. Another method that uses this approach is a deep-learning-based method [12] and is described in more detail in section 2.3.

Related Work - Deep One Class Classification

The goal of the Deep One Class classification algorithm presented in paper [12], is to create a model that produces feature representations to the data points with the intention that these features vectors will be separable in the feature space. In other words, the desired feature space should sustain the property that the distance between data points that belong to the one-class data will be closer than their distance to out-of-class data points in the feature space. To be able to do that, the method maintains two properties, which are the *descriptiveness*, and the *compactness* properties. The descriptiveness property allows the model to distinguish between classes and is gained by initially training the model to classify some *reference dataset*. In the second stage, the model will learn the compactness property, which learns to produces similar features for instances from the one-class, while continuing the maintenance of the descriptiveness property.

In this work, instead of initially training the model on the classification task, they used a pre-trained model on the ImageNet dataset and used the ImageNet dataset as the reference dataset. For example, this network can be VGG16 network that is commonly used for transferring features representations in computer vision. To guarantee these descriptiveness and compactness abilities, the method maintains two losses during the training - The *descriptiveness loss* that is the cross-entropy loss with respect to the reference dataset, and the *compactness loss* that is the averaged similarity between the samples in a batch of the one-class data, which is called in the paper *target data*. More formally, let $X = \{x_1, \dots, x_n\} \in \mathbb{R}^{n \times k}$ be a batch of target data. The distance between a given sample and the rest of the samples in a batch can be defined as:

$$z_i = x_i - m_i$$

where, $m_i = \frac{1}{n-1} \sum_{j \neq i} x_j$ is the mean of the rest of the samples. The compact loss term is:

$$l_C = \frac{1}{nk} \sum_{i=1}^n z_i^T z_i$$

In the training phase, the model consists of two similar CNNs. The first network is the *Reference network* which is a pre-trained network on the reference dataset. The second network is the *Secondary network* which is identical to the reference network except of the last fully connected layer that doesn't exist in this network. The weights of these two networks are tied across each corresponding counterpart forcing them to be identical.

At each training iteration, two batches of images are simultaneously fed into the model. The first batch consists of data from the reference dataset. This batch is fed into the reference network, and descriptiveness loss is calculated according to the output of the model. The second batch consists of target data samples that are fed into the secondary network, and from its output, the model calculates compactness loss. Let r be a reference batch and t be a target batch, the loss of the network is defined as:

$$l(r, t) = l_D(r|W) + \lambda l_C(t|W)$$

where λ is a constant, l_D is the descriptiveness loss and l_C is the compactness loss. By minimizing the descriptiveness loss, the learned features obtain the ability to describe different concepts concerning the reference dataset. By minimizing the compactness loss, the learned features of the target class are getting more compact in the feature space. The weight λ determines the mutual importance of these properties.

In inference, only the secondary network with the learned weights is used, and it involves two phases. The first phase aims to create a set of templates that represents

the target data. It is achieved by drawing a small set of samples from the target data $t = \{t_1, \dots, t_n\}$, inserting them into the secondary network, and then extracting for each sample a feature representation vector. Those n features vectors $\{g(t_1), \dots, g(t_n)\}$ are stored as templates and used in the second phase, which measures the distance of a given input from the other templates in order to classify the input. The classification algorithm used for the second phase is a simple *k-nearest neighbor classifier*. When a given input image y is given, a corresponding feature vector is calculated by inserting y into the Secondary network. Then using the k -nearest neighbor classifier and the n template from the first phase, the algorithm calculates a matching score:

$$S_y = f(g(y)|g(t_1), \dots, g(t_n))$$

where f is the matching function of the classification algorithm. The classification of the image is determined according to this matching score.

2.4 Background - Positive Unlabeled Learning (PU Learning)

Positive-Unlabeled learning is another approach to learning to distinguish between positive and negative examples without providing negative examples. During the training phase, the model receives only positive and unlabeled instances. The fundamental assumption of this method is that the unlabeled data can contain both positive and negative examples. This approach is different from the one-class classification approach in that it explicitly incorporates unlabeled data into the learning process, instead of considering only positive examples. The key to this approach is the *labeling mechanism* described in the survey [1]. The mechanism suggests that the labeled positive examples are selected from the complete set of positive examples according to a labeled probability and presents the labeled distribution as a biased version of the positive distribution.

According to [1], there are two options to explain why an example is unlabeled. The first option is that the example is truly negative and the second is that it is positive, but simply was not selected by the labeling mechanism to have its label observed. Therefore, PU learning methods make assumptions about either the labeling mechanism, the class distributions in the data, or both. In our work, we would like to avoid assumptions on the data distributions, because it may limit the creative space of our method. Therefore, we decided to focus on the one-class classification and the weakly supervised learning approaches.

2.5 Background - Weakly Supervised Learning

Supervised learning is a machine learning task that maps input to an output based on a large number of training examples. In many tasks, it can be difficult to achieve a strong and large enough training dataset due to the high cost of the data-labeling procedure. Furthermore, in some problems, such as our creative thinking problem, the labeling task can be even more difficult since the classification of some examples can be uncertain or ambiguous. The cases where the available correctly labeled data is not enough for learning is known as weakly supervised learning .

The paper [18] presented three main weakly supervised types: the first type is *semi-supervised learning* that uses only a (usually small) subset of labeled data for training and the other data remain unlabeled. Formally, the task is to learn $f : \mathcal{X} \rightarrow \mathcal{Y}$ from a training data set $D = \{(x_1, y_1), \dots, (x_L, y_L), x_{L+1}, \dots, x_{L+N}\}$ with L labeled examples and N unlabeled examples. This may arise when the labeled data is not big enough, for example, because the labeled data is collected by human annotators and thus is expensive. The second type is *learning from noisy labels* that as in the case when the labeled data includes mistakes. For example, when it is collected by human annotators that are careless or weary, or some images are difficult to categorize. The third type is *learning from coarse-grained supervision* that uses only coarse-grained labeled data in training. This situation can occur for example when labeling each instance is a difficult task, but we have some intuition about groups of instances, and we can use it to create a course-grained prior for the model training.

In this work we explore the *Ballpark learning* method presented in [8] and [9] for statistical data. The method learns from coarse-grained supervision and provides a solution to the weakly supervised problem described in section 2.5. In this work, we will also demonstrate the method's good performance when learning from misclassified data.

Background: The Ballpark Problem

A weakly supervised problem is presented in the papers [8] and [9] and uses prior knowledge about groups of instances in order to train a linear classification or regression model. Formally, consider $X_N = \{x_1, x_2, \dots, x_N\}$ a set of N instances with corresponding unknown labels $y_1^*, y_2^*, \dots, y_N^* \in Y$. Where Y is the label space that can be either discrete or continuous. In addition, we could be given a set of $L \geq 0$ labeled training instances $X_L = \{x_1, x_2, \dots, x_L\}$ with a known binary labels $y_1, y_2, \dots, y_L \in Y$, such that $N \gg L$. Also,

we are given a set of K subsets of $X = X_N \cup X_L$, which we call bags:

$$B = \{B_1, B_2, \dots, B_K\}, \quad B_i \subseteq X.$$

The bags in B may overlap and do not necessarily cover all the training instances in X_N . The task is to learn a function $f(x) = w^T \varphi(x)$ that predicts the individual labels by utilizing prior knowledge associated with the labels within the bags. where $\varphi(\cdot)$ is a feature map that outputs a feature vector for a given instance (To simplify the notation bias term b , we assuming that a vector 1_{N+L} is appended to the features $\varphi(x)$). For example, such prior knowledge can be coarse bounds on the positive and negative labels proportion in a certain bag. The truth labels proportion is assumed to be unknown. In order to solve the problem, the papers [8] and [9] formulated it as an optimization problem. In chapter 3 we formulate our problem as a Ballpark problem and adjust the models proposed in those papers to our needs.

OUR METHOD

Usually, we have some intuition about whether visual classes can fulfill a certain task. For example, given an image of a knife, we can assume with a high probability that one can stab with that knife without even seeing the specific image. Similarly, given an image of a pillow, we can assume that the probability of stabbing with it is very low. This intuition motivates us to use the *Ballpark learning* approach. Instead of providing to our model labeled instances, which would require laborious labeling of individual images from each class, we can provide to the model some coarse, intuitive constraints on all the images of objects from a certain visual class. Then, the Ballpark model will learn the visual properties that are relevant for the desired task by itself.

3.1 Input preparation

The Ballpark learning method is designed to solve the problem described in section 2.5 and has been analyzed on statistical data for weakly supervised tasks. In order to use the algorithm on image data, we needed to find a way to encode the relevant features of an image in a representation vector, as it did for the statistical data in the Ballpark framework. In this work, we used the transfer learning approach and extracted the feature representative of an image from a pre-trained deep neural network model.

As explained in section 1.1, the prior knowledge we used in our work is associated with the labels of groups of instances which we called bags. In our experiments, we used

as the training bags, the classes provided in standard benchmark datasets. We called these classes auxiliary classes since they are determined according to the images' content, without considering the task. The prior knowledge is coarse and intuitive constraints relate to the possibility that the bags' instances comply with the task. By using these constraints and the training dataset, our model will learn to distinguish between positive examples that comply with the task and negative examples that do not.

3.2 Solving the Ballpark Model for Divergent Thinking Problems

The Ballpark technique includes two methodologies. The first methodology is a binary classification framework described in paper [8] and the second is the regression framework described in paper [9]. In our work, we chose to extensively analyze the Ballpark regression framework, since it can be formalized and solved as a convex optimization problem and therefore has one global optimal solution. In addition, as explained in [9] the training procedure of the Ballpark classification framework is much slower than the Ballpark regression framework. Since the Ballpark regression model outputs for a given input a number in \mathbb{R} , we consider this number as the *compliance score* of that image for a given task. Meaning the score indicates how much the content of the given image complies with the given task.

In this work, we consider two types of compliance scores. The first type is the *instance compliance score* that relates to the possibility that a single instance is compliant with a task. The second is the *bag compliance score* that is the averaged compliance scores of the bags' instances and relates to the possibility that its instance will comply with a task. A high compliance score of an instance indicates that the instance significantly complies with the task, and a high compliance score of a bag implies that the bag's instances are likely to be compliant with the task. For example, in the "stabbing" affordance task, the bag of knives should receive a high compliance score, while the bag of pillows should possibly get a lower score.

Formally, the task is to find $f : \varphi(\mathcal{X}) \rightarrow \mathcal{Y}$ where \mathcal{X} is the images space, $\varphi(\cdot)$ is a features map that maps the images into features vectors, and $\mathcal{Y} = [0, 1]$ is the compliance scores space of the instances. As previously explained, the prior knowledge we use for the Ballpark model training in the divergent thinking problems is a set of coarse constraints on the compliance scores of the training bags. Meaning for each training bag B_i we

provide constraints that relate to the scores average of its instances $\bar{y}_i = \frac{\sum_{j \in B_i} y_j^*}{|B_i|}$, where y_j^* is the unknown compliance score of the j -th instance of the bag.

We allowed two types of constraints. The first type is the Lower and upper bounds on a bag proportion, namely $l_i \leq \bar{y}_i \leq u_i$ and the second type is the Differences constraint that relates to the difference between the proportions of two different bags and determines lower and upper bound on this difference, meaning $l_{ij} \leq \bar{y}_i - \bar{y}_j \leq u_{ij}$.

As in the Ballpark framework, we can use the prior knowledge to reveal the individual scores by formalizing the problem as a convex optimization task. Let $y^* \in \mathcal{Y}^N$ be the vector of the unknown scores and $y \in \mathcal{Y}^L$ be the vector of the known scores. Let \mathfrak{A} be the subset of B , to which we have lower and or upper bounds. Let \mathfrak{D} be a set of tuples $(B_{i_1}, B_{i_2}) \in B \times B$, to which we have Differences bounds. The ballpark regression method formulates the following convex optimization problem:

$$(3.1) \quad \underset{y^* \in \mathcal{Y}^N, w}{\operatorname{argmin}} \quad \frac{1}{2} w^T w + \frac{C_N}{N} \sum_{i=1}^N \|y_i^* - w^T \varphi(x_i)\|_2^2 + \frac{C_L}{L} \sum_{j=1}^{N+L} \|y_j - w^T \varphi(x_j)\|_2^2$$

subject to

$$l_k \leq \hat{y}_k \leq u_k \quad \forall \{k : B_k \in \mathfrak{A}\},$$

$$l_{k_{12}} \leq \hat{y}_{k_1} - \hat{y}_{k_2} \leq u_{k_{12}} \quad \forall \{k_1 \neq k_2 : (B_{k_1}, B_{k_2}) \in \mathfrak{D}\},$$

$$0 \leq \hat{y}_k \leq 1 \quad \forall \{k : B_k \in \mathfrak{B}\}.$$

where C_N and C_L are cost hyper-parameters and control the weights given to the unknown and constrained scores and the known scores respectively. \hat{y}_k is the unknown compliance score of bag B_k

In figure 2, we summarize the scheme for training a Ballpark model to solve divergent thinking problems. First, the user should find a training dataset, split it into bags logically, and set intuitive constraints on the compliance scores of the bags. In addition, the user needs to select a good feature extraction method for the representation of the images; then, train the model by solving the Ballpark optimization problem according to formula 3.1. The stages are independent and can be replaced or changed separately for the specific need. For example, suppose we would like to predict whether one can stab with an object or not. We can select several classes with objects that we have some intuition about the possibility they allowed stabbing. For example, let's consider the Knives and Pillows bags. For each of these classes, we can estimate coarsely the possibility that it affords the Stabbing task and according to this estimation, we can provide constraints to our model. The constraints for our task may include a low upper bound for the bag of Pillows and a high lower bound for the bag of Knives. We may also

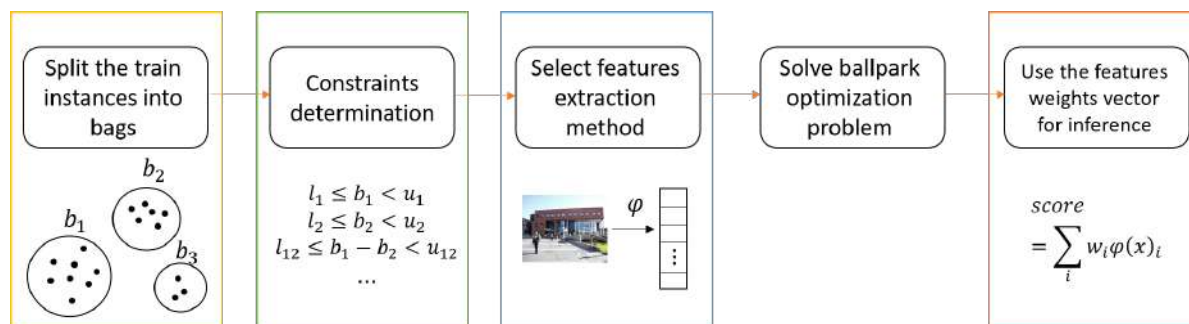


Figure 3.1: Summary of the Ballpark’s training procedure. In the first three steps, we prepare the input for training by splitting the training instances into bags, determining constraints on these bags, and selecting a good features extraction method for the representation of the images. Then, we solve the Ballpark optimization problem on these training data according to the constraints. The model outputs a vector of weights that determine which features are important for the required task.

include a difference constraint between the scores of the Knives’ bag and the Pillows’ bag with a lower bound greater than zero.

3.3 Iterative Labeling Procedure

Determining constraints without knowing well the instances of a dataset, can be sometimes difficult, and it even becomes harder when the dataset contains a lot of classes. In order to mitigate this difficulty, we present an iterative labeling procedure that can be used to adjust and fix the model to the specific task. The procedure’s idea is to add new relevant constraints on new bags according to the current mistakes of the model. We consider as *bag mistake* of a model, a bag compliance score that is not suitable to the user intuition, meaning scores that are too low or too high, or mistakes in the ranking order of classes. For example, if the task is to find a tool that is stab-able, the Knives class should get a compliance score that is higher than 0.5 and also is higher than the Pillow class. If the model returns a compliance score that is not in accordance with this intuition, the user can add new constraints that will force the model to consider this intuition too.

The procedure includes three stages, the first stage is to train the model according to initial coarse constraints on a small number of bags, the second stage is to run the model on the rest of the unconstrained bags and find scores mistakes. The last stage is to examine those mistakes in order to add new constraints to improve the model performance. We name this procedure an *iterative labeling step*, and it can be repeated

to create enough constraints for the training dataset. In the result chapter, we showed that we successfully improve our model's results using this mechanism.

EXPERIMENTS AND RESULTS

In the preceding chapters, we presented two different approaches we examined for solving divergent thinking tasks. The first approach is the one class classification approach that uses positive examples to train a model to classify between images that affords an affordance task, and images that do not. The second approach is the weakly supervised learning approach that uses labels associated with a group of instances, instead of using the labels of the individuals. In this chapter, we extensively analyze the adjusted Ballpark model, which is based on the weakly supervised learning approach, and show its superiority on several different affordance tasks. For our convenience, in this chapter, the name Ballpark model is related to the adjusted Ballpark version and not the original ballpark model. In the first part of this chapter, we evaluated our model numerically and compared its results to baselines and to the one-class classification model we adjusted to our problem. In the second part of this chapter, we showed the results of a user study we conducted to evaluate different aspects of the divergent thinking ability of the Ballpark model.

4.1 Methodology

Divergent thinking tasks

As we described in the introduction chapter, in our research we focus on divergent thinking tasks in the area of affordances and analyzed our model on two types of

affordances problems. The first type is a places affordance problem, that deals with what a place can afford. A place can contain different objects and describe a specific view within some location. To solve this type of problem, the model should determine whether one can do something in a place described a given image, for example determine if a place affords concealment, namely if it is a hiding place. The second problem is the object affordance problem, that is a problem that focuses on a specific object, and the model should determine if one can use an object in a given image for a specific purpose. During this work we examined a number of affordance tasks.

The places affordance problems we examined are:

- Dine problem - Determine whether a given scene image represents a place that one can dine in conveniently.
- Shopping problem - Determine whether a scene image represents a place that one can find something to buy.
- Architecture for tourism problem - Determine whether a scene image represents a recommended urban place for tourism.

The object affordance problems we examined are:

- Stabbing problem - Determine whether one can stab with some object in an image.
- Flowerpot problem - Determine whether one can use some object in an image as a flowerpot.

Data

In this chapter, we evaluate our model on several divergent thinking tasks that are divided into two types of affordance problems - places and objects. In this work, we use scene images to represent a place, an image of a centered object on a plain background to represent an object. The dataset we use in this work are based on the following datasets, which are all already divided into classes according to the content of the images:

- *ADE20K dataset* [16] [17] - Includes 20,210 scene-centric color images in the training set and 2,000 images in the validation set. The dataset's images are split into 719 scene classes, and fully annotated with objects. In our experiments we merged the training and the validation set into one dataset by merging the instances of the same class from both datasets to one class and used the scene labels of the images as the model's auxiliary classes. The instances of this dataset were split according to the scene labels. The number of images in each scene varied from several to thousands.

- *Object-centered datasets* collected from Google images - We collected different datasets for both the Stab and the Flowerpot problems in the following way. First, we select a number of objects for each problem that we had some intuition on the possibility to use them in the required task. Then, we crawled in Google images using a keywords, in order to collect instances for each object. The instances in this dataset split according to the object types, and each object type contains 100-200 images. In this dataset the images usually contained a centered object on a plain background.
- *ImageNet validation dataset* - standard benchmarks for image classification, includes 50,000 color images, with 1,000 classes and 50 images per class.
- *Caltech256 dataset* [3] - standard benchmarks for image classification, includes 30,607 images that are split into 257 categories, each category includes 80 – 827 images.

All the training datasets that were used in this work were created in the following procedure. First, we selected a suitable dataset according to the task type, then from the dataset, we selected classes which we have some intuition about their compliance, and according to this intuition, we determined training constraints. In addition, based on the training datasets, we created a corresponding binary labeled training dataset by manually labeling all the instances of each training bag. We used these datasets to train some of our baselines. The training datasets we used for the places affordances tasks - the Dine, the Shop, and the Architecture tasks, were based on the ADE20K dataset, and the training datasets of the Stab and the Flowerpot affordance tasks were based on the Object-centered dataset.

Some of our evaluations are based on test datasets, therefore we created for each affordance task a different test dataset, that we labeled according to the task. To collect the test datasets for the place affordance tasks we removed all the constraints bags from the ADE20K dataset, and then for each remaining bag, we sampled five random images and labeled them accordingly. For the Stab affordance task, we selected 10 classes of objects from the Caltech256 dataset and labeled all their instances, and for the Flowerpot affordance task, we collected instances of 15 new classes in the same way we collected the Object-Centered datasets from Google images. It is noteworthy that We found it difficult to label the data according to the Architecture task and therefore we didn't evaluate the Architecture task numerically, but only analyzed its results in the second evaluation scheme that is based on the results of a user study.

The full constrained classes we use to train our models are described in table A.1. We also describe the size of each training dataset in table A.2, and the size of each test dataset in table A.3.

As we explained previously, some of our experiments are based on the result of a user study we conducted during our research. To collect examples for the user study, we also selected *exploration datasets* for the Shop, Architecture, and Stab tasks. We wanted to ensure that the user study’s examples will include creative examples that are also possible to classify according to an affordance task, therefore we looked for exploration datasets that include a lot of classes as well as a lot of images. For the Shop and the Architecture tasks, we used the ADE20K dataset without the constrained bags, and for the Stab task, we used the ImageNet dataset.

Constraints’ creation

In most of the experiments in our work, we used intuitive constraints of lower and upper bounds, which we determined from general knowledge about objects and without seeing the images inside the constrained bags, thus the bounds are very coarse. In addition, for each set of task’s constraints, we also added Differences constraints by calculating them automatically from the intuitive lower and upper bounds. The constraints are displayed in the Appendix in Table A.1.

Baselines

As of today, there is no other method that solves the divergent thinking problem of images, thus we compare our results to the results of the following baseline models:

- *Polar support vector machines [PSVM]* - A possible way to solve our problem is by selecting the Ballpark’s bags with upper and lower constraints and using their instances as the positive and negative examples for training a standard SVM model. Let $p_l, p_h \in [0, 1]$ be two hyper parameters of the polar SVM model and denote the model by $PSVM_{p_l, p_h}$. We used those two parameters to divide the task’s training bags into positive and negative bags. The bags with lower bound that is equal or above $1 - p_l$ were considered positive, and the bags with upper bound that is equal or under p_h will be considered negative. We ignored the remaining bags that don’t fulfill these conditions. We considered all the instances of the positive bags as positive examples, and all the instances of the negative bags as negative examples and used them to train a standard SVM model.

- *Supervised support vector machines [SSVM]* - We evaluated the standard SVM model with different numbers of ground truth labels, in order to see how many labeled examples are needed to obtain results comparable to our method. The training dataset of these models is based on the binary training dataset we described previously.
- *Deep one class classification [DOC]* - This model that is based on the one-class classification approach, was introduced extensively in the background chapter. We compare our model performances to the performances of this model on our affordance tasks to see which approach is better for our problem. The DOC model’s training requires two types of datasets, the Reference dataset, and the Target dataset. In our experiments, we used the ImageNet dataset as the Reference dataset, and for each task, we selected all the positive examples from the corresponding binary training dataset as the Target dataset. In addition, as the feature extraction method, that the DOC model also required, we used the same feature extraction method we used for the corresponding Ballpark model - a pre-trained neural network on the ImageNet dataset. The specific neural network architecture will be specified in the experiment setting. The training parameters of the models are the same as those used in the experiments in paper [12].

It is noteworthy, that the first two of our baselines are based on the SVM algorithm, which was explained extensively in the background chapter. Our motivation for using this algorithm came from the fact that the SVM algorithm’s sample complexity doesn’t depend on the dimension of the features space, as explained in the background chapter, and therefore is preferable when learning a halfspace separator in high dimension features space with a small training dataset.

4.2 Experimental Protocol

In the next section, we analyze the performance of our model for different affordance tasks. In most of the experiments, if not specified otherwise, in order to train a Ballpark model, we used the training dataset and its corresponding constraint set that are described in the methodology section. In addition, if not specified differently, we used a pre-trained Resnet50 architecture trained on the ImageNet dataset for extracting visual features from images.

We adopted two evaluation paradigms - *assessment* and *exploration*. In the assessment paradigm, we evaluated our model numerically using the Area Under the ROC Curve (AUC) score, which provides an aggregate measure of performance across all possible classification thresholds. To evaluate the models numerically, for each task in addition to the constraints set, we created a test dataset with binary labels. Since the constraints and the test data were created by the same creator, we considered the evaluation scores achieved in this paradigm as scores that relate to the success of the model on the *user's divergent thinking*, namely divergent thinking that is biased by the creator, and not related to a *general divergent thinking*, that represents the opinion of the majority. For the *general divergent thinking* evaluation, we used the exploration paradigm that compares the ranking results of our model to the results of humans by conducting a user study.

Assessment Protocol

In this evaluation scheme, we compare our model results to baselines. For training Polar SVM models, we adapt the dataset used for training the corresponding Ballpark model to a binary form according to the procedure described in the Methodology section. The other baselines train by a designated labeled dataset, which are described also in the Methodology section. As explained previously, the Polar SVM uses thresholds on the Ballpark constraints, to split the bags into positive and negative examples. The best values of these thresholds depend on the specific problem and the specific training bags, therefore, instead of reporting the results of one pair of thresholds, we report the averaged results of all the pairs $p_l, p_h \in \{0.1t : t \in \{1, \dots, 6\}\}$. (We ignore pairs that create training data without positive or negative examples). In addition, to overcome the randomness of the supervised SVM, which train on random labeled examples, and the randomness of the DOC method, which uses a random weights initialization, for each of these models, we repeat each experimental condition five times and report the average results as well as the standard deviation.

To compare the SVM baselines to our model, it is required that those baselines output scores instead of binary numbers. Thus, in the inference phase, we don't use one threshold to classify an instance. Instead, we use the learned SVM parameters w, b , where w is the learned weights vector, and b is the learned bias to predict the score of a given instance by the formula $w^T \varphi(x) + b$, where $\varphi(\cdot)$ is a feature representation method.

Exploration Protocol

The exploration paradigm aimed to analyze the following two divergent thinking abilities:

- *Suggesting creative new classes to solve a problem* - This is the ability of the model to find relevant new classes that are likely to be suitable for a given task.
- *Suggesting creative new instances to solve a problem* - This is the ability of the model to determine for a given instance if it is suitable to solve the task.

In order to analyze our model on these abilities, we conduct three surveys, one survey for each of the Architecture, Shop, and Stab tasks. To collect data for the surveys, for each task we use a trained Ballpark model with the default setting described at the beginning of this section. We ran these models in their corresponding exploration dataset described in the Methodology section and achieved compliance scores for all the instances in the dataset as well as for all the bags in the dataset, by averaging the scores of the instances of each bag.

Each survey is composed of two parts. The first part aimed to measure the first ability, which is the ability to suggest creative new classes to solve a problem. We collected examples for each survey by extracting the 20 classes with the highest compliance score received by the Ballpark model, which trained on the corresponding task. In addition, for each survey, we selected 10 random classes from the exploration dataset, that were not in the former classes and used them as the survey’s distractors.

In the survey, we displayed the names of those 30 classes. Near each of these classes, we displayed a few random images from the corresponding class and asked the participant to determine if the class affords the task or not. In the following section, we analyzed the results of this part by comparing the classes that our model selected to those which selected in the corresponding survey.

The second part of the surveys aimed to measure the second ability, that is the ability to suggest creative new instances to solve a problem. This ability is trickier to measure since it requires a visual examination of instances from an immense size exploration dataset. Furthermore, examining only the best-scored instances, couldn’t measure creativity, since the instances are likely to be more similar to the bags that the model saw in the training phase, and not necessarily creative or diverse enough. Therefore, instead of analyzing the ability to suggest creative instances among all the instances in the exploration dataset, we analyzed the ability to suggest creative instances for a given specific class.

For the second part of the surveys, we included five sections, each section represents a different class and consists of 10 images from the class. In each section, we asked the participants to rank the images according to the task. To ensure that those sections also include innovative solutions, we selected classes with high variance in the scores that their instances received by our model. To find the classes with high variance, for each class in the exploration dataset, we sorted the scores of its instances and found the score of the fifth instance with the lowest score and the score of the fifth instance with the highest score. We calculated the difference between these two scores and selected the 10 classes with the highest difference. From these 10 classes, we selected randomly 5 classes for the survey.

When examining the scores that our model gave to the instances of the exploration datasets, we noticed that when the difference between the instances' scores is small, it become harder to determine which instance is better. Therefore, in each section, we selected from the corresponding class the five images with the highest scores and the five images with the lowest scores according to our model and displayed them in random order. We asked the participants to rank them from 1 to 5, according to their compliance with the task, and expected that the surveys' participants would determine that the five best images selected by our model are preferable to the task. In figure 4.3 we demonstrate the ranking order of 10 images for three different classes each for a different task.

4.3 Results

In the first part of the section, we analyze the Ballpark method performance on the Dine, Shopping, Stab and Flowerpot tasks, and compare its results to the baselines described in the methodology section. Afterward, we demonstrate the visual results of our model that also were used in the surveys we conducted for the user study part, and in addition, we display the results of an ablation study. In the last part of this section, we analyze the user study's results we conducted for the Architecture, Shop, and Stab tasks.

Comparisons to the Baselines

In this section, we evaluate our model and compare its results to the results of the baselines and the Deep One-Class Classification model[DOC]. The models' setting and the experiment datasets were described in section 4.2. The results are shown in Table

4.1. As seen in that table, our model achieved the best results in all the four divergent thinking problems we checked, without using any ground-truth labels.

PROBLEM	BALLPARK	PSVM	SSVM 100 LABELS	SSVM 300 LABELS	DOC
DINE	0.89	0.85 [0.006]	0.84 [0.04]	0.87 [0.009]	0.71 [0.15]
SHOP	0.92	0.85 [0.011]	0.84 [0.024]	0.85 [0.007]	0.63 [0.034]
STAB	0.92	0.88 [0.013]	0.82 [0.028]	0.79 [0.036]	0.65 [0.012]
FLOWERPOT	0.87	0.84 [0.011]	0.79 [0.051]	0.84 [0.042]	0.57 [0.028]

Table 4.1: The AUC evaluations for the Dine, Shop, Stab and Flowerpot problems. The PSVM column reports the total average and the standard deviation (in parentheses) of the set of the Polar SVM models described in section 4.2. The supervised SVM (SSVM) and the Deep-One-Class Classification (DOC) columns report the average and the standard deviation (in parentheses) of 5 similar experiments of the corresponding models.

Ballpark with Polar Supervision

In Table 4.1 it is also noticeable that the Polar SVM model which learns from polar supervision has the best results among the other baselines. Therefore, we also explore the impact of adding polar supervision to the Ballpark model as well. In the method chapter, we described the Ballpark algorithm that in addition to the constrained bags’ data, also uses labeled data. Let $PSVM_{p_l, p_h, t}$ be a Polar SVM model that is trained with polar data according to the threshold pair p_l, p_h for a task t . Let $Ballpark_{p_l, p_h, t}$ be a Polar Ballpark model that in addition to the data of the constrained bags, is trained by all the labeled data that is used to train $PSVM_{p_l, p_h, t}$. In this experiment we chose all the threshold pairs from $p_l, p_h \in \{0.1t : t \in \{1, \dots, 6\}\}$ (We ignored pairs that create training data without positive or negative examples), and evaluate the models on the Dine, Shop and Stab tasks. In Figure 4.1 we display the AUC evaluations of the Polar Ballpark and the corresponding Polar SVM models with different polar bounds, on the Dine, Shop, Stab, and Flowerpot tasks. We also evaluated a Ballpark model trained with the same constraints, that are used to train the corresponding Polar Ballpark models with different polar thresholds, but without any supervision, and displayed its results with a dashed line. As seen in Figure 4.1, the Ballpark model with and without the polar labeled data surpassed the Polar SVM results for all the polar thresholds and all the four tasks. Additionally, it can be noticed that in three out of four evaluated tasks, the Ballpark model achieved the best results when trained without any labeled data.

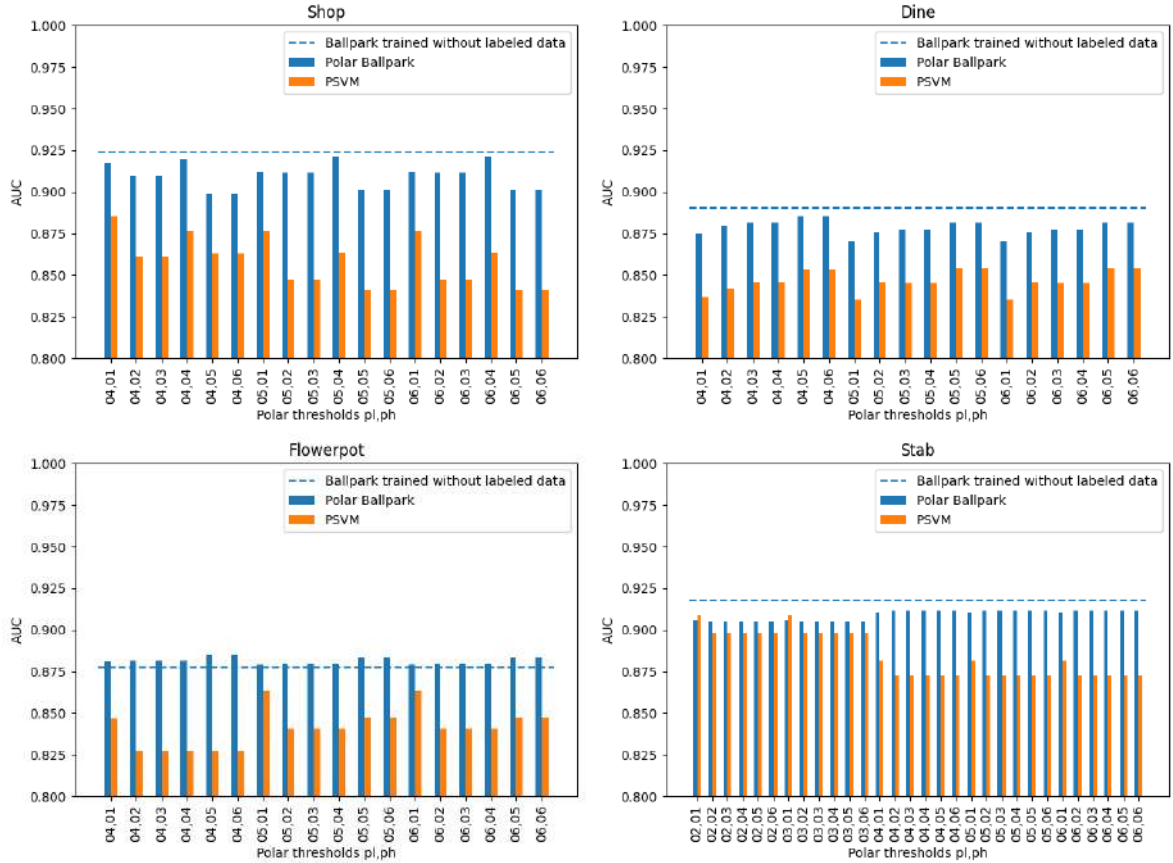


Figure 4.1: Summary of the impact of using polar supervision in the Polar Ballpark and the Polar SVM models on the Dine, Shop, Stab, and Flowerpot tasks. The labels of the x-axis are the polar thresholds pair p_l, p_h , that were used to split the training data of corresponding Polar Ballpark and Polar SVM models, into positive and negative instances according to the task’s constraints file. The height of the bars represents the AUC score of each model. Each pair of Polar Ballpark and Polar SVM models was trained with the same labeled instances according to the polar threshold, and the Polar Ballpark model was also trained by the constraints file of the task. We also displayed in a dashed line a standard Ballpark model that was trained by the task’s constraints file without any additional labeled data.

Ballpark with Noise

In this experiment, we analyzed our model performance when learning on a noisy dataset. We consider a noisy dataset as a dataset whose bags include misclassified examples. Namely, each constrained bag in the training data, which is represented by some class, also includes instances that don't belong to this class. For each task, we prepared several noisy datasets with different compositions of noise and ground truth data. Each of these noisy datasets is represented by a parameter p , that is the ratio between the amount of ground truth and the amount of noisy data in bag B . More specifically, let n be the number of instances in bag B in the original dataset. The corresponding bag in a noisy dataset with parameter p contains $n \times (1 + p)$ instances, where $n \times p$ instances are random instances from the original bag B , and n are randomly sampled instances from other auxiliary classes. During the experiment, we kept the number of noisy examples in each training bag constant and increased the number of ground truth examples.

In figure 4.2, We compared the results of our model to the results of the different Polar SVM models on several noisy datasets with different p values. We evaluated the models' AUC performance on noisy datasets with parameters $p \in \{0.2t : t \in \{0, \dots, 5\}\}$ for the Dine, Shop, Stab, and Flowerpot tasks. In each experiment, both models were trained by the same noisy dataset.

As shown in the plots in Figure 4.2, while the Ballpark model is significantly improved with more additional ground truth data, the Polar SVM model's performance barely changed and remained very low.

Iterative Labeling

In this experiment, we demonstrated the iterative labeling procedure described in the previous chapter on the Shop and the Stab tasks. The procedure required enough bags for the initial training step and the fixing step afterward; therefore in this experiment we use the Caltech 256 dataset as the training dataset of the Stab problem. For the Shop problem, we use the ADE20K dataset as before. The constrained bags in each iterative labeling step are displayed in Table 4.3 and the results in 4.2. It is noticeable that the additional constraints improved the model's AUC score significantly in both tasks.

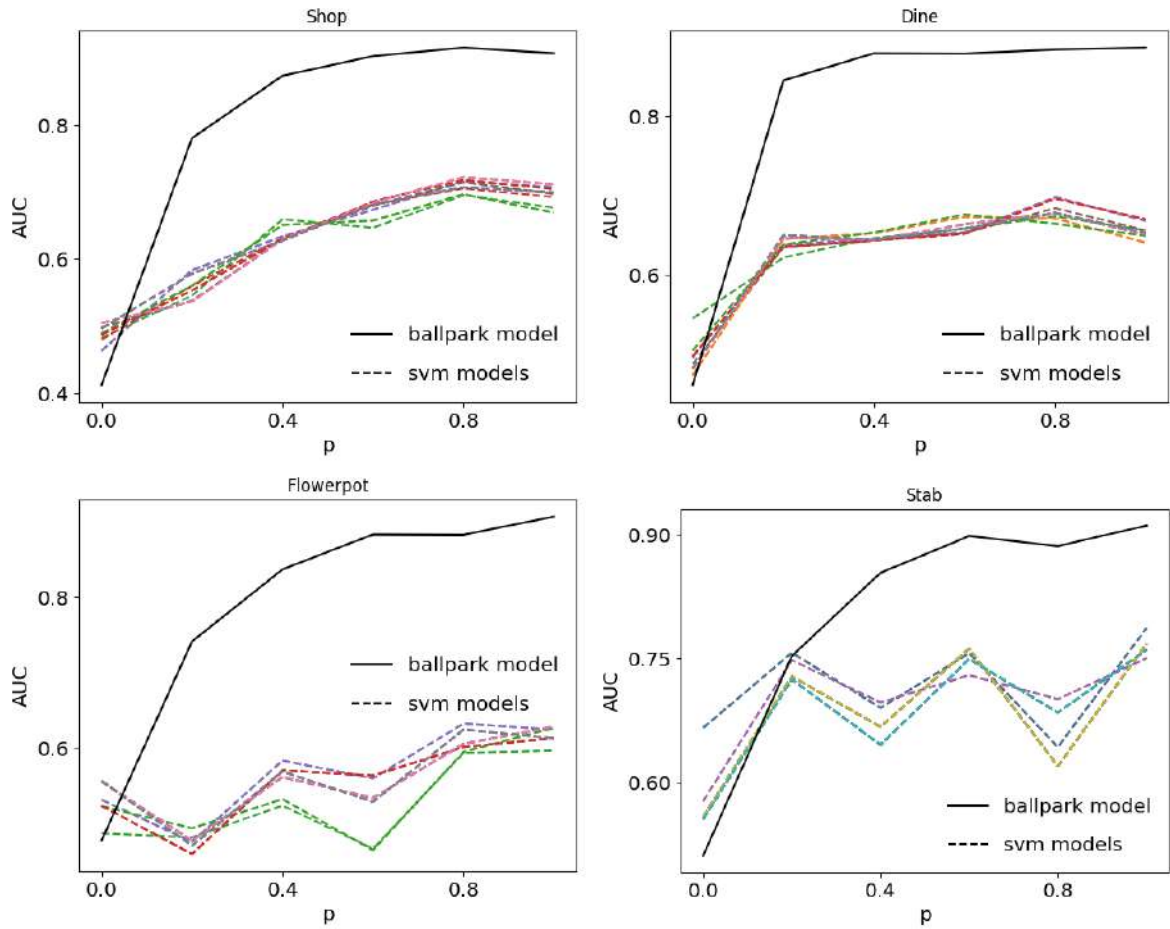


Figure 4.2: The impact of noisy training data on the Ballpark and the Polar SVM models with the different threshold values are described in the methodology section. The parameter p is the ratio of the ground truth data and the noisy data in each bag.

Demonstration of the Ballpark Model Recommendations

Demonstration of the Ballpark Model Classes Recommendations for Divergent Thinking Tasks

In Table 4.4 we demonstrated the model classes suggestions for the Shop, Stab, and Architecture tasks. The results displayed in that table are the 10 highest scoring classes chosen by our model and were used in the first part of the user study. The procedure of extracting these suggestions is described extensively in the Experimental protocol section.

Problem	step 0	step 1
stab	0.86	0.89
shop	0.85	0.90

Table 4.2: The AUC results in each iterative labeling step, for both the Stab and the Shop tasks.

Problem	step 0	step 1
Stab	syringe, sword, sushi, spoon, sneaker, joystick	soda can, hourglass, harp, ak47, knife - chopsticks, mandolin, pez dispenser, harmonica, knife
Shop	shoe shop, restaurant, bar, pet shop, hat shop, reception, jewelry shop, supermarket, flea market, convenience store, street 1000 ^a , fast-food restaurant, mountain, forest, house, dining room, access road, kitchen, diner	ball pit, movie theater, boxing ring, podium, nightclub, discotheque, auditorium, catwalk, conference center, martial arts gym

Table 4.3: The additional constrained bags in each iterative labeling step. When the name of the bag is a subtraction between bags, it means that the constraint is a Difference constraint between the two bags. The rows of the table are the tasks names, and the columns are the procedure steps.

^aA smaller bag with 1000 random instances from the original street class

The Ranking Results of the Model in a Class

In Figure 4.3 we demonstrate our model’s ranking results on the Architecture, Shop, and Stab tasks. For each task, we selected a representative class and displayed for it the 5 best images with the highest model’s scores and the 5 images with the lowest model’s scores. The images order is determined according to the images’ scoring. The Architecture task’s results are demonstrated with the Balcony class’s instances, the Shop task with the Game room’s instances, and the Stab task with the Letter opener’s instances.































Architecture		Shop		Stab	
Class	Examples	Class	Examples	Class	Examples
donjon		tobacco shop		scabbard	
ruin		delicatessen		letter opener	
gatehouse		hardware store		corkscrew	
caravan-sary		tea shop		fountain pen	
mission		drugstore		hatchet	
monastery		bookstore		hammer	
whispering gallery		shopfront		can opener	
palace		liquor store		cleaver	
pagoda		kiosk		revolver	
moat		video store		ballpoint	

Table 4.4: 10 best images with the highest scores according to the Ballpark model for the Architecture, Shop and Stab tasks. The images are sorted from the highest scoring image on the top to the lowest on the bottom. The first column displays the class label and the second displays two images selected randomly from the corresponding class.



Figure 4.3: Demonstrates the results from the second part of the survey for the instances ranking. For each task we demonstrate the models ranking by one class. For the Architecture task we use images of the Balcony class, for the Shop task we use images of the Game room class and for the Stab task we use images of the Letter opener class. For each class we display the five images with the highest scores and the five images with the lowest scores sorted order according to the model's scores. The scores received by the model are displayed below the corresponding images.

4.4 Ablation Study

Using Only Differences Constraints

When working on a new dataset, it can be difficult to estimate for each of its classes the probability that it complies with a given task without knowing the dataset content and its distribution. A much easier task is to use our intuition about the relation between the classes and add constraints according to this intuition. For example, it is reasonable to claim that the needle class is probably more suitable for the Stab task than the umbrella class. Nevertheless, the umbrella class can include umbrellas with a sharp tip that allows the stabbing use, and the needle class can describe needles with some cover. Without knowing the classes' content, it may be difficult to determine the possibility that each class is compliant with the task, and the Differences constraints allow us to describe this knowledge without constraining the classes separately.

In this experiment, we analyzed the ability of the Ballpark model to learn from Differences constraints without constraining instances or classes separately. It is noteworthy that the Ballpark model is the only model among those we examine that allows learning from Difference constraints, therefore we compared the results of this section to the best results described in Table 4.1. We evaluated our model on two sets of Differences constraints. The first set is the Differences constraints calculated automatically from the boundary constraints in table A.1 and the second set is a new intuitive coarse Differences constraints we built. In Table 4.5 we show that our model roughly achieved the same results without constraining the bags separately and using only Differences constraints and still beat the other baselines.

The Impact of Using Different Feature Space

To verify that our model achieves the best results independently from the features space selected to represent the images, we also evaluated the models on a pre-trained VGG16 model on the ImageNet dataset for the Dine, Shop, Stab, and Flowerpot tasks. We may notice in Figure 4.4 that our model beat the Polar SVM models and the DOC models for both the ResNet50 and VGG16 features extraction methods and all the four tasks.

CONSTRAINTS TYPE	DINE	SHOP	STAB	FLOWERPOT
DEFAULT DIFFERENCES	0.88	0.91	0.92	0.87
INTUITIVE DIFFERENCES	0.88	0.92	0.90	0.87
BALLPARK RESULTS IN TABLE 4.1	0.89	0.92	0.92	0.87
BEST BASELINE RESULTS FROM TABLE 4.1	0.87 [0.009]	0.85 [0.011]	0.88 [0.013]	0.84 [0.042]

Table 4.5: The AUC evaluations of the Ballpark model on the Dine, Shop, Stab, and Flowerpot problems by using only Difference constraints. The first row is the results of the ballpark model when training only by the Difference constraints calculated automatically from the bounds constraint in Table A.1. The second row of the table displays the results of the Ballpark model with the intuitive Difference constraints. The third row is the Ballpark results from the Table 4.1, and the last row is the best result achieved by the baseline in Table 4.1. The square brackets indicate that the score in that cell, achieved by averaging the results of several models, and the number in the brackets is the standard deviation of these results.

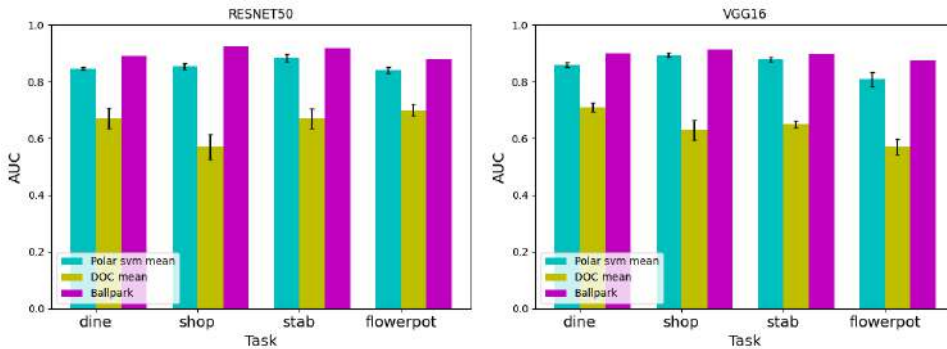


Figure 4.4: Comparisons between the Ballpark, the Polar SVM with the different polar thresholds, and the DOC models using different features spaces. The plots report the AUC results of the models on two common features extraction methods. The black lines are the standard deviation of the Polar SVM and the DOC models. The right plot is the results of the models when using a ResNet50 architecture, and the second is for the VGG16 architecture.

4.5 User Study

In this section, we present the results of the user study which was described extensively in the Experimental Protocol section. The user study aims to assess the ability of our model to recommend creative new classes to solve a problem and to assess the ability of our model to recommend creative new instances to solve a problem from a given class. The user study includes three surveys, one survey for each of the Shop, the Stab, and the Architecture tasks. In the Shop survey, we had 13 participants, in the Stab survey we had 19 participants, and in the Architecture survey, we had 15 participants.

It is noteworthy that we didn't analyze the correlation between the ranking results of our model and those of the survey's participants, since the content of the exploration dataset not necessarily represents the real world. For example, the Donjon class in our exploration dataset consists of images of ancient architectures, but in the real world, this class may also be represented by images of prisons that do not necessarily look like architecture for tourism. This fact may explain why our model gave the Donjon class the highest score among the others in the Architecture exploration dataset but in the survey, this class was scored lower than several classes in the same exploration dataset.

Suggest Creative New Classes to Solve a Problem

In this part, we compared our model recommendations for classes that afford an affordance task to those recommended by the surveys participants. As explained in the methodology section, for each task we conducted a separate survey and asked the participant to select all the classes that they think comply with the given task. The classes in the survey included the 20 best-scored classes according to the model and 10 random classes that are used as the survey's distractors. The class's score according to a survey is calculated by averaging the number of participants who select the class in the survey. We interpreted these scores as binary labels by considering a class as positive if at least 50% of the participants chose it in the survey.

In Table 4.6 we evaluated the model results according to the survey results by considering the binary labels received from the survey as ground truth and calculating the precision and the recall scores. As shown in that table, the model received a good recall score meaning almost all the classes selected in the survey as classes that comply with the tasks were also selected by our model. The only exception is the Escarpment class selected by the survey participants as a class that complies with the Architecture task but didn't select by our model. It is notable that in the survey we explicitly explain

TASK	RECALL	PRECISION
SHOP	1	0.85
STAB	1	0.6
ARCHITECTURE	0.95	0.7

Table 4.6: Evaluate the model results using the ground truth labels collected in the surveys.

	SHOP	STAB	ARCHITECTURE
FALSE NEGATIVE CLASSES	-	-	ESCARPMENT
FALSE POSITIVE CLASSES	SHOPFRONT, STORAGE ROOM, PACKAGING PLANT	HAMMER, CAN OPENER, REVOLVER, RIFLE, BASSOON, CARPENTER PLANE, PAINTBRUSH, SHOVEL	MISSION, MOSQUE, SACRISTY, IMARET, SCRIPTORIUM, NUNNERY

Table 4.7: The model mistakes according to the survey. The first row consists of the classes that were considered as positive in the surveys but were not selected by our model. The second row displays the classes that our model considered as positive but weren't selected in the survey.

to the participants that architectural places don't include natural places; therefore, this selection contradicts our guidance.

Nevertheless, the precision scores of our model are lower and vary more between the tasks. The Shop score received the best precision scores, meaning most of the classes selected by our model comply with the Shop task according to the survey as well. In Table 4.7 we display the classes that were selected by our model as positive examples but didn't select in the survey. As shown in that table, for the Shop and the Architecture tasks, most of the classes that our model considered as positive but weren't selected by the survey are reasonable selections too. For example, all the places in the Architecture task are architectural places as well. These results may indicate that the participant selections involve some subjective opinions, or that the participant didn't know the places well.

In the Stab task, our model received the lowest precision score, and in addition, its mistakes in Table 4.7 sustain our conclusion that our model performance on this task is not good enough. A possible explanation for this low result is that the ImageNet dataset doesn't include enough suitable classes, and therefore these 20 classes suggested by our model include classes that do not comply with the task.

To support this assumption, we randomized 100 classes from the ImageNet dataset and tagged them according to the possibility to stab with them. After examining this random subset, we found out that almost 50% of its classes are classes of animals and places and their images mostly described realistic scenes with complex backgrounds. As explained before, most of the examples that our model trained on include centered objects with a plain background, therefore the ImageNet examples may not be suitable for the space that our model learned. Furthermore, in this specific subset, the only classes that we labeled as suitable were the Bittern and the Dowitcher that are both kinds of birds, and the Rock-crab that is a kind of a crab. These classes are very creative selections and differ from the training dataset. In addition, it is noteworthy that the survey's distractors include the Ptarmigan class that is also a kind of bird and got an average score of 0.31, which is not enough for a class to be considered as suitable. This result may imply that in general birds do not comply with the Stab task.

We concluded that the selection of the ImageNet dataset as an exploration dataset for the survey is problematic, and it may explain the low results we received for the Stab task. Overall, the results of the surveys indicate that our model suggests creative new classes for the divergent thinking tasks, and especially creates fair suggestions for the place affordance tasks - Shop and Architecture.

It is noteworthy that we decided not to analyze the correlation between the classes' ranking of our model and the survey's participants. The main reason is that the datasets classes may not correlate with reality, and therefore the ranking order according to the model may not accord with the reality as well. For example, the images in the Donjon class in the ADE20K dataset contain only outdoor images of old architectural buildings that seem to be used for tourism rather than imprisonment. This result causes our model to give the highest rank score to this class, where the survey's participants, that also considered donjons as places for imprisonment, ranked this class in a lower place.

Suggest New Instances to Solve a Problem

In the second part of the survey, we displayed ten images for five different classes to the participants and asked them to score those images according to the possibility that they comply with the survey's task. The aim of this part is to analyze the ability of our model to suggest creative new instances to solve a problem from a given class. To analyze the survey results, for each image, we averaged the participants' scores and considered the average score as the survey score of that image.

In Figure 4.5 we illustrate the results of three different classes, each class represents the results of a different survey’s task. Each bar represents a different class’s image, their position in the x-axis is the score the image received by our model and their height indicates are the average score they received in the survey. For our convenience, we colored the five images with the highest model’s scores in blue, and the five images with the lowest model’s scores in orange. As shown in that plot there is agreement between the model and participants on the suitability of the best five images. In both the Game room class and the Balcony class, we see that the five best images selected by our model were also selected by the participants. In the Letter opener class, there is one disagreement, but the four best images selected by the participants were also selected by our model.

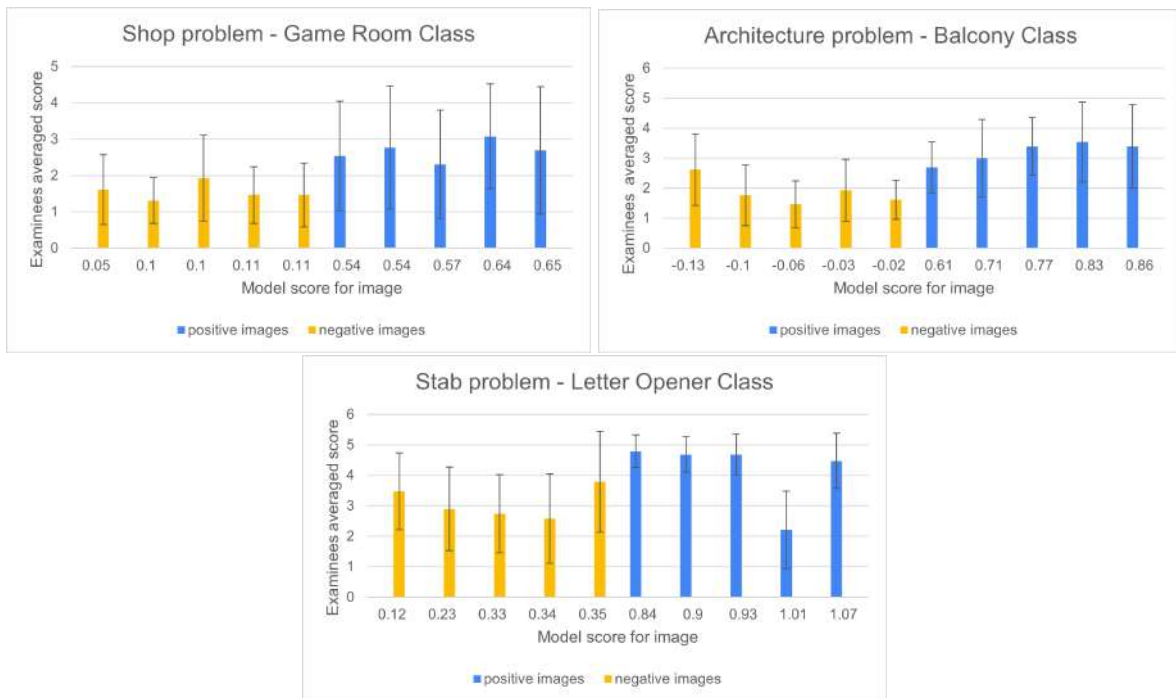


Figure 4.5: Illustration of the user study’s results for three different classes, each class evaluate according to different affordance task. The x-axis is the scores that our model gave to each of the class’s images. The y axis is the participants’ average score for the corresponding image. The black line in each bar is the standard deviation of the participants’ scores. The blue bars represent the best images according to the model ranking, and orange bars represent the images with the lowest scores according to the model.

Nevertheless, as previously explained in the survey, we evaluated our model on five different classes. In Figure 4.6 we summarized the results of these five classes for each survey task. We named the five best-scored images according to the model as positive

images and the five lowest-scored images according to our model as negative images. For each class, we averaged the survey’s scores of the positive and the negative examples separately and displayed them in different bars in the plots. As seen in the figure, for all the tasks and all the classes the average score received by the positive images are higher than the score received by the negative images. These results sustain our conclusion that our model can suggest reasonable instances to solve affordance tasks from a given class.

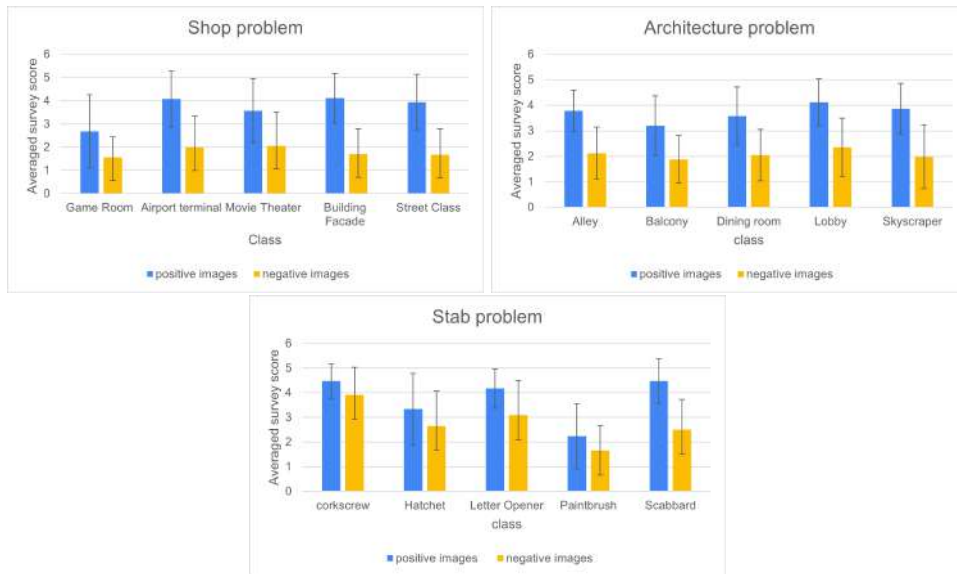


Figure 4.6: summaries the results of the user study second part. Each plot represents a different task. For each class we display two bars one for the positive images (that are the images that received the best model’s scores in the class) and one for the negative images (The images that received the lowest model’s scores in the class). The height of the bars indicates the average survey’s scores, and the black lines are the standard deviation of these scores.

SUMMARY AND CONCLUSIONS

5.1 Summary and Discussion

In this work, we presented a new problem in computer vision - divergent thinking in visual data. We proposed two approaches to solve the problem, the one-class classification approach, and the weakly supervised learning approach. We analyze the performance when using these approaches, by adjusting state-of-the-art methods to our problem, one for each approach, and analyze their results. One of the adjusted methods is Ballpark learning that designated to solve problems in statistical data. In this work, we adjust the Ballpark method to solve problems in images, and analyzed its performance on divergent thinking problems. In addition, we adapt two evaluation paradigms that aim to measure different aspects of the divergent thinking properties.

In the results chapter, we demonstrated the supremacy of the Ballpark model in divergent thinking tasks. We evaluated the model in four different divergent thinking tasks and compared its results to the results of the baselines, and the DOC model. Our model's results surpassed the other models in all four tasks, without using any labeled data and even when it trained by only differences constraints. We analyzed the impact of using polar supervision in the Ballpark model in addition to the original constraints, and we found out that our model exceeded the results of the Polar SVM when both were exposed to the same polar supervision. This result implies that there is some inherent knowledge that exists in the constraints, and doesn't exist or is distorted in the polar labeled data. Furthermore, in most cases, the Ballpark results are better when no polar

supervision is in use, which indicates that the polar supervision is redundant when using constraints and even mislead the models. In addition, we discovered that the Ballpark model is significantly less sensitive to noise than the Polar SVM model as demonstrated in Figure 4.2.

We also analyzed the performance of our model with a user study, to evaluate the model results for general divergent thinking. We showed that the Ballpark model suggestions for new classes and new instances, are reasonable, and mostly represent the opinion of the majority. When analyzing the user study results, we found that the results of our model for the place-affordance tasks, namely the Shop and the Architecture tasks, were better than the results that were achieved for the Stab task. One reasonable cause, which we strengthened with an additional examination of the data in the user study section, can be that the Stab exploration dataset is too different from the training dataset in its visual properties; another possible cause is that the feature representation method works better on scene images.

One interesting future work is in the field of feature representations for images. The experimental results of the user study imply that the feature representation method we use in our work is less effective for object affordances tasks. Therefore we suggest analyzing the Ballpark model's selected features to improve the results of our model. Another interesting work would be to add additional, not visual, features to the image representation such as encode the objects' materials and sizes.

APPENDIX



APPENDIX

APPENDIX A. APPENDIX

DINE	SHOP	STAB	ARCHITECTURE	FLOWERPOT
SHOE SHOP > 0.7 RESTAURANT > 0.7 BAR > 0.5 MOUNTAIN < 0.05 FOREST < 0.05 HOUSE < 0.2 0.1 < KITCHEN < 0.8 ACCESS ROAD < 0.1 PET SHOP < 0.2 0.3 < DINER < 0.95 0.6 < LOBBY < 0.9 BOTANICAL GAR. < 0.2 TENNIS COURT < 0.3 PARKING LOT < 0.2 LANDING < 0.2 VERANDA > 0.5 BALLROOM < 0.2 BAKERY < 0.8 OFFICE > 0.7 0.3 < HOTEL ROO. < 0.9 BEACH < 0.2 0.1 < PATIO < 0.9 BOWLING ALL < 0.1 BEER HALL > 0.7 LEGISLATIVE CHA. > 0.7 LAKE < 0.1 PARK < 0.3 0.2 < LIBRARY < 0.7 CAFETERIA > 0.7 BEDCHAMBER < 0.5 SACRISTY < 0.5	SHOE SHOP > 0.7 RESTAURANT > 0.2 BAR > 0.5 PET SHOP > 0.6 HAT SHOP > 0.7 RECEPTION > 0.7 JEWELRY SHOP > 0.6 SUPERMARKET > 0.6 FLEA MARKET > 0.7 CONVENIENCE ST. > 0.7 0.5 > STREET 1. > 0.05 FASTFOOD RES.T > 0.6 MOUNTAIN < 0.05 FOREST < 0.05 HOUSE < 0.2 DINING ROOM < 0.2 ACCESS ROAD < 0.4 KITCHEN < 0.2 0.7 > DINER > 0.1 BALLROOM < 0.4 SUBWAY INTER. < 0.2 NEWSSTAND > 0.6 CANDY STORE > 0.7 BAKERY > 0.7 BISTRO > 0.6 MARKET > 0.6 MUSIC STORE > 0.6 DRESS SHOP > 0.6 BANK > 0.5 AUDITORIUM < 0.1 HOT TUB < 0.1 BLEACHERS < 0.4 STRIP MALL > 0.7 FOOD COURT > 0.7 POOLROOM < 0.2 MARTIAL ARTS. < 0.2 CHAPEL < 0.2 BANQUET HALL < 0.2 GAS STATION > 0.6	POT < 0.10 BOTTLE < 0.2 0.2 < PENCIL < 0.7 0.6 < SWORD 0.2 < FORK < 0.9 CUP < 0.10 0.8 < KNIFE 0.2 < TWEEZER. < 0.7 BAG < 0.10 0.05 < UMBREL. < 0.8	0.8 < ABBEY < 1.0 0.8 < APSE < 1.0 0.6 < ARCH < 1 0.0 < ATRIUM < 0.2 0.0 < ATTIC < 0.1 0.8 < BASILICA < 1.0 0.1 < BRIDGE < 0.8 0.0 < BUILDING F. < 0.4 0.8 < CASTLE < 1.00 0.8 < CATHED. < 1.0 0.8 < CHAPEL 0.8 < CHURCH < 1.0 0.0 < CITY < 0.4 0.8 < CLOISTER < 1.0 0.0 < CORRIDOR < 0.2 0.5 < COURTHO. < 0.9 0.0 < DINER < 0.1 0.8 < FORTRESS < 1.0 0.8 < GREAT HALL < 1.0 0.0 < HOUSE < 0.1 0.0 < MOTEL < 0.1	BAG < 0.2 0.1 < BARREL < 0.7 BOWL > 0.7 BOX > 0.5 BUCKET > 0.7 CHAIR < 0.2 CUP > 0.6 0.2 < JAR < 0.8 KEYBOARD < 0.1 LADDER < 0.1 PLATE < 0.2 POT > 0.6 RUG < 0.1 SCISSORS < 0.1 SHELF < 0.5 SHIRT < 0.1 SHOE > 0.7 SPOON < 0.2

Table A.1: The bags constraints used for train the models in most of the experiments in this work. Each model received also differences constraints, calculated from these bags bounds automatically.

DINE	SHOP	STAB	ARCHITECTURE	FLOWERPOT
2071	3721	949	1680	7055

Table A.2: The number of instances in the constraints bags in A.1.

DINE	SHOP	STAB	FLOWERPOT
3174	3754	830	2622

Table A.3: The number of instances in the test dataset in A.1.

	SURVEY POSITIVE CLASSES	SURVEY NEGATIVE CLASSES
MODEL POSITIVE CLASSES	TOBACCO SHOP, DELICATESSEN, HARDWARE STORE, TEASHOP, DRUGSTORE, BOOKSTORE, LIQUOR STORE, KIOSK, VIDEOSTORE, PHARMACY, TICKET WINDOW, BUTCHERS SHOP, GREENGROCERY, CHECKOUT COUNTER, FISHMARKET, TOYSHOP, BOOTH	SHOPFRONT, STORAGE ROOM, PACKAGING PLANT
MODEL NEGATIVE CLASSES		HILL, BALCONY, BASEMENT, BLEACHERS, GAME ROOM, SLUM, TOWN HOUSE, DRIVEWAY, MILITARY TENT, QUADRANGLE

Shop survey results

	SURVEY POSITIVE CLASSES	SURVEY NEGATIVE CLASSES
MODEL POSITIVE CLASSES	SCABBARD, LETTER OPENER, CORKSCREW, FOUNTAIN PEN, HATCHET, CLEAVER, BALLPOINT, CARPENTER'S KIT, SCREW , PROJECTILE, SCREWDRIVER, BOW	HAMMER, CAN OPENER, REVOLVER, RIFLE, BASSOON, CARPENTER PLANE , PAINTBRUSH, SHOVEL
MODEL NEGATIVE CLASSES		PTARMIGAN, TOY TERRIER, MINIATURE SCHNAUZER , COUGAR, BASSINET, DOGSLED, HONEYCOMB, MILITARY UNIFORM, MOSQUE, PADLOCK

Stab survey results

	SURVEY POSITIVE CLASSES	SURVEY NEGATIVE CLASSES
MODEL POSITIVE CLASSES	DONJON, RUIN, GATEHOUSE, CARAVANSARY, MONASTERY, WHISPERING GALLERY, PALACE, PAGODA, MOAT, TOWER, KASBAH, SYNAGOGUE, CATACOMB, PORTICO	MISSION, MOSQUE, SACRISTY, IMARET, SCRIPTORIUM, NUNNERY
MODEL NEGATIVE CLASSES	ESCARPMENT	GLADE, MEZZANINE, AIR BASE, RACEWAY, GASWORKS, MINE, CANTEEN, AUTO RACING PADDOCK, GREENHOUSE

Architecture survey results

Table A.4: The classes division according to the ranking of the model and the surveys' first part results

BIBLIOGRAPHY

- [1] J. BEKKER AND J. DAVIS, *Learning from positive and unlabeled data: A survey*, Machine Learning, 109 (2020), pp. 719–760.
- [2] C. CORTES AND V. VAPNIK, *Support-vector networks*, Machine learning, 20 (1995), pp. 273–297.
- [3] J. DENG, W. DONG, R. SOCHER, L.-J. LI, K. LI, AND L. FEI-FEI, *Imagenet: A large-scale hierarchical image database*, in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.
- [4] J. J. GIBSON, *The ecological approach to visual perception: The theory of affordances*, Chapter, 8 (1986), pp. 127–143.
- [5] J. P. GUILFORD, *Three faces of intellect.*, American psychologist, 14 (1959), p. 469.
- [6] —, *The nature of human intelligence.*, (1967).
- [7] K. HE, X. ZHANG, S. REN, AND J. SUN, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
- [8] T. HOPE AND D. SHAHAF, *Ballpark learning: Estimating labels from rough group comparisons*, in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2016, pp. 299–314.
- [9] —, *Ballpark crowdsourcing: The wisdom of rough group comparisons*, in Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 234–242.
- [10] M. KEMMLER, E. RODNER, AND J. DENZLER, *One-class classification with gaussian processes*, in Asian Conference on Computer Vision, Springer, 2010, pp. 489–500.

BIBLIOGRAPHY

- [11] S. J. PAN AND Q. YANG, *A survey on transfer learning*, IEEE Transactions on knowledge and data engineering, 22 (2009), pp. 1345–1359.
- [12] P. PERERA AND V. M. PATEL, *Learning deep features for one-class classification*, IEEE Transactions on Image Processing, 28 (2019), pp. 5450–5463.
- [13] S. SHALEV-SHWARTZ AND S. BEN-DAVID, *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press, 2014.
- [14] K. SIMONYAN AND A. ZISSERMAN, *Very deep convolutional networks for large-scale image recognition*, arXiv preprint arXiv:1409.1556, (2014).
- [15] D. M. TAX AND R. P. DUIN, *Support vector data description*, Machine learning, 54 (2004), pp. 45–66.
- [16] B. ZHOU, H. ZHAO, X. PUIG, S. FIDLER, A. BARRIUSO, AND A. TORRALBA, *Scene parsing through ade20k dataset*, in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5122–5130.
- [17] B. ZHOU, H. ZHAO, X. PUIG, T. XIAO, S. FIDLER, A. BARRIUSO, AND A. TORRALBA, *Semantic understanding of scenes through the ade20k dataset*, International Journal of Computer Vision, 127 (2019), pp. 302–321.
- [18] Z.-H. ZHOU, *A brief introduction to weakly supervised learning*, National science review, 5 (2018), pp. 44–53.