applied sciences

MDPI

Article

Applying Object Detection and Embedding Techniques to One-Shot Class-Incremental Multi-Label Image Classification

Youngki Park [1] and Youhyun Shin [2,*]

[1] Department of Computer Education, Chuncheon National University of Education, Chuncheon 24328, Republic of Korea; ypark@cnue.ac.kr

[2] Department of Computer Science and Engineering, Incheon National University, Incheon 22012, Republic of Korea

* Correspondence: yhshin@inu.ac.kr

Abstract: In this paper, we introduce an efficient approach to multi-label image classification that is particularly suited for scenarios requiring rapid adaptation to new classes with minimal training data. Unlike conventional methods that rely solely on neural networks trained on known classes, our model integrates object detection and embedding techniques to allow for the fast and accurate classification of novel classes based on as few as one sample image. During training, we use either Convolutional Neural Network (CNN)- or Vision Transformer-based algorithms to convert the provided sample images of new classes into feature vectors. At inference, a multi-object image is analyzed using low-threshold object detection algorithms, such as YOLOS or CutLER, identifying virtually all object-containing regions. These regions are subsequently converted into candidate vectors using embedding techniques. The k-nearest neighbors are identified for each candidate vector, and labels are assigned accordingly. Our empirical evaluation, using custom multi-label datasets featuring random objects and backgrounds, reveals that our approach substantially outperforms traditional methods lacking object detection. Notably, unsupervised object detection exhibited higher speed and accuracy than its supervised counterpart. Furthermore, lightweight CNN-based embeddings were found to be both faster and more accurate than Vision Transformer-based methods. Our approach holds significant promise for applications where classes are either rarely represented or continuously evolving.

Keywords: multi-label image classification; one-shot learning; object detection; embedding

check for updates

Citation: Park, Y.; Shin, Y. Applying Object Detection and Embedding Techniques to One-Shot Class-Incremental Multi-Label Image Classification. *Appl. Sci.* **2023**, *13*, 10468. https://doi.org/10.3390/app131810468

Academic Editor: Zhengjun Liu

Received: 8 August 2023
Revised: 17 September 2023
Accepted: 18 September 2023
Published: 19 September 2023

1. Introduction

Multi-label image classification is a task in which multiple classes are identified from a given image. For instance, when presented with an image that contains both a cat and a dog, the task is to predict the presence of both these classes (cat and dog) in the image. This task has applications in various domains, including autonomous vehicles [1], fashion [2], medical imaging [3], visual content tagging [4], and education [5]. Many different approaches have been proposed for this task. Recent methods, in particular, emphasize enhancing multi-label classification accuracy, especially by leveraging semantic label relationships or consistencies [6–9]. Clearly, these strategies have significantly improved multi-label image classification accuracy, especially on well-known datasets, such as MS-COCO [10], Pascal VOC [11], and Fashion550K [2].

In specific application domains, one-shot learning [12] and class-incremental learning [13] are essential for multi-label image classification. In this context, "one-shot" means that only a single sample is used to train each class, while "class-incremental" indicates that new classes are added to the model over time. A common example is in smartphone gallery applications, where users annotate images of new acquaintances. In this situation, it is natural for users to annotate just one image for each new acquaintance, illustrating the "one-shot" approach. Additionally, as new acquaintances are added as new classes, "class-incremental" learning becomes necessary. Despite the limited annotated data available for these emerging classes, such as new acquaintances, the gallery application must still effectively perform multi-label image classification. Another example can be found in primary school computer science and machine learning education. Here, young students update their machine learning models by adding or removing classes based on just one or a few samples [5,14].

Unfortunately, most of the existing approaches do not focus on one-shot learning or class-incremental learning for multi-label image classification. Instead, many existing methods concentrate on classifying widely recognized datasets, such as MS-COCO. Although there has been a recent paper introducing the term "few-shot class-incremental learning [15]", it does not consider situations where the total number of data is very limited or when entirely new types of datasets are used. As a result, there is a need for a novel approach. We delineate our unique requirements for one-shot class-incremental multi-label image classification as follows:

- Requirement 1: Our model must be capable of rapid adaptation when new classes of images are added, a process we refer to as "class-incremental learning". Conversely, it should also be equipped with the ability to forget specific classes, a concept opposite to class-incremental learning, which we denote in this paper as "class-decremental learning".
- Requirement 2: Our model must perform multi-label classification effectively, even when presented with a limited number of data. Furthermore, it should allow for easy insertion or deletion of a small number of training samples.
- Requirement 3: Our model should be able to manage unpopular or unexpected image combinations that do not reflect the real world. For instance, even when our input images contain elements like dolphins and grass together, which is an unnatural and unexpected combination, our model should still perform well on these unconventional datasets.
- Requirement 4: Our model must be lightweight to ensure that it runs efficiently on low-performance computers, including laptops with basic GPUs. This is a critical requirement for certain application domains where only low-end hardware is available. For instance, in K-12 education settings, low-performance computers are commonly used. Despite these constraints, our model should still deliver fast performance.

In this paper, we present a straightforward yet effective approach to satisfying the aforementioned requirements. Our primary strategy involves employing object detection and embedding techniques for multi-label image classification. Initially, we use existing supervised or unsupervised object detection algorithms to identify objects within the composite image approximately. These identified objects are subsequently embedded into vectors using CNN- or Vision Transformer-based models. Then, we execute nearest-neighbor search algorithms on these embedding vectors to derive the predicted labels. In greater detail, the main contributions of this paper include the following:

- In Section 3, we construct a novel dataset for one-shot multi-label image classification using random characters and background images. This dataset can be used to evaluate a model's performance on unseen and novel datasets.
- In Section 4, we introduce a simple yet powerful approach to one-shot class-incremental multi-label classification. This is achieved by leveraging object detection and image-embedding techniques, eliminating the need for fine-tuning the model on new data. Additionally, we examine various types of object detection and image-embedding models to identify those that best meet our requirements.
- In Section 5, we conduct extensive experiments to test our approach and thoroughly analyze the experimental results.

2. Related Work

There are many multi-label image classification approaches, with many focusing on improving accuracy by using the semantic relationships among labels. Zhou et al. [6]

proposed an effective method to find correlations between semantic labels and visual spaces using an auto-encoder. Nguyen et al. [7] presented modular graph transformer networks for this task, exploiting both semantic information and topological label information. Lanchantin et al. [8] proposed a classification transformer that leverages dependencies among labels and visual features, and it is based on predicting a set of masked labels. Guo et al. [9] introduced a way to preserve "attention consistency" even when various types of spatial transforms are present. While these approaches perform well, they focus on traditional multi-class label classification datasets such as MS-COCO [10], without considering one-shot class-incremental multi-label classification.

To the best of our knowledge, research by Tao et al. [15] offers the approach most similar to the one in our paper. However, they concentrate on fine tuning and still utilize a significant number of data for training, a major distinction from our research. There is also research using k-nearest-neighbor algorithms for multi-label image classification [16], but as it does not exploit neural network models or object detection models, it substantially differs from our approach.

Traditional approaches for calculating similarities among images include SSIM (structural similarity index measure) [17], which detects similarity using luminance, contrast, and structure. However, we omitted the results using this method in our experiments because it did not perform well on our dataset. More advanced approaches embed an image into a vector to calculate similarities, with ResNet-18 [18] and VGG-16 [19] being two of the most popular CNN-based techniques. Recently, even more sophisticated methods have been developed, such as embedding vectors based on Vision Transformers [20]. One notable example is the CLIP model [21] developed by OpenAI, and other popular Vision Transformer-based models have been developed by companies like Google and Facebook.

3. Dataset Construction

In this section, we build a dataset for one-shot multi-label image classification. To meet the requirements introduced earlier, our goal is to construct a dataset comprising many classes, with a very limited number of samples within each class.

Recall that one of the application domains for this task is computer science education for children. Therefore, we decided to collect data from Scratch [22,23], one of the world's most popular children's programming environments. Within Scratch, there are two main types of images: backdrop images and character images. The characters are referred to as "sprites" in Scratch terminology.

Our idea for constructing the dataset for this task involves randomly placing multiple sprites on random backdrops. These arbitrary placements result in very noisy merged images, making the task challenging for multi-label image classification algorithms. However, such noisy images are commonly found in children's Scratch programs.

We collected all of the backdrop images from Scratch, a total of 85 images. These collected backdrop images are represented in Figure 1.

In Scratch, there are hundreds of sprite images, but some images are nearly identical, as one sprite can have multiple images with slight modifications to represent character movements in animations. To clarify the problem, we randomly selected 100 sprite images and left only one image for each sprite. We also deleted sprite images that had copyright issues when distributed. As a result, we obtained 69 sprite images, as shown in Figure 2.

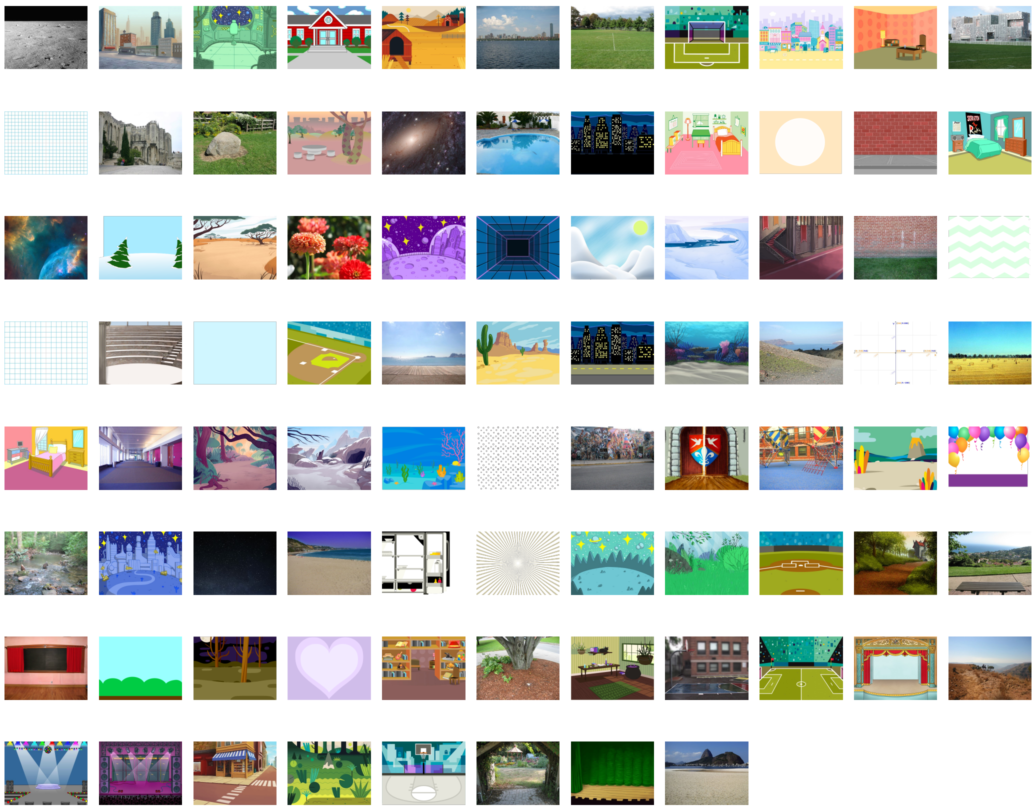

Figure 1. A total of 85 backdrop images collected from the Scratch programming environment.

Figure 2. A total of 69 sprite images selected from the Scratch programming environment.

The next step was to place the sprites randomly on random backdrops. However, the sizes of the sprite images vary widely, and they may overlap, causing one image to be hidden in the merged image. To prevent this, we imposed a restriction that the width and height of any sprite should not exceed 200 pixels. Additionally, we ensured that the x or y coordinates of any two sprite images were not less than 200 pixels apart. Within these constraints, we randomly varied the size of each sprite image from 100% to 200%.

The constructed dataset includes three types of merged images, with three, six, or nine sprite images placed on each backdrop, respectively. For example, the first column of subfigures in Figure 3 represents the randomly selected backdrop images, and the second column of subfigures in Figure 3 represents the nine sprite images placed on each backdrop. In this figure, our task is to find only the nine sprites we randomly placed. Obviously, this is challenging because there are other objects that the backdrop already has. For example, in the second row of subfigures, what we want to do is not find objects like a bed, a pillow, a mirror, or a monitor but find items such as a drum, a hat, a skeleton, etc. The constructed dataset can be downloaded at https://github.com/tooeeworld/mlic_dataset (accessed on 19 September 2023.).

In summary, we constructed a new dataset that is fundamentally different from any existing datasets. As a result, this new dataset can be considered "unseen" or "novel", making it ideal for evaluating one-shot class-incremental approaches without overlapping with the data used to train existing neural network models.

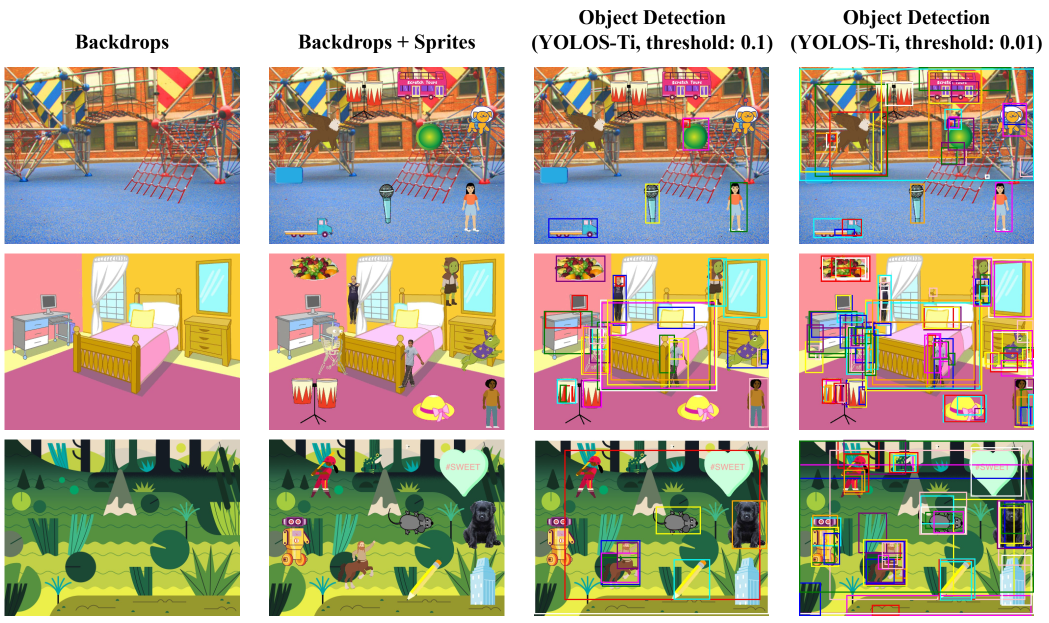
Backdrops
Backdrops + Sprites
Object Detection (YOLOS-Ti, threshold: 0.1)
Object Detection (YOLOS-Ti, threshold: 0.01)

Figure 3. Examples of generated data are shown in the first- and second-column subfigures, along with object detection results using YOLOS-Ti in the third- and fourth-column subfigures. In these examples, we added nine sprites to randomly selected backdrops. The object detection model was neither trained nor fine-tuned on our datasets, so it could not detect our sprites effectively. By lowering the threshold of the pre-trained model, more sprites could be detected, albeit with a larger number of false positives.

4. Proposed One-Shot Multi-Label Image Classification Approach

4.1. Overview

Figure 4 provides an overview of our approach, which consists of four main steps:

1. Initially, we take samples from unseen classes, one sample or more per class. These samples are embedded into vectors using image-embedding models based on either Convolutional Neural Networks (CNN) or Vision Transformers.
2. Next, for an image that includes both background and objects, we employ either supervised or unsupervised object detection pre-trained models like YOLOS-Ti and CutLER. Because these models have not been fine-tuned on unseen data, their object detection accuracy may be limited. To compensate, we adjust the detection thresholds

appropriately, aiming to capture all objects from unseen classes, even if this results in a higher rate of false positives.

3. The objects detected are then embedded into vectors using pre-trained CNN- or Vision Transformer-based embedding models.
4. Finally, we compare these embedded vectors to the vectors of previously embedded unseen class samples using k-nearest-neighbor search algorithms such as Faiss [24] or HNSW [25]. By considering all the k-nearest neighbors, we determine the predicted multiple labels for each object.

Note that our methodology relies on pre-trained models without the need for any fine tuning tailored to our specific datasets. This approach affords us considerable flexibility and aligns well with the objectives set forth in the introduction. Even though the data used to train these pre-existing models differ substantially from our own datasets, we observed that utilizing them effectively reduces the incidence of false positives.

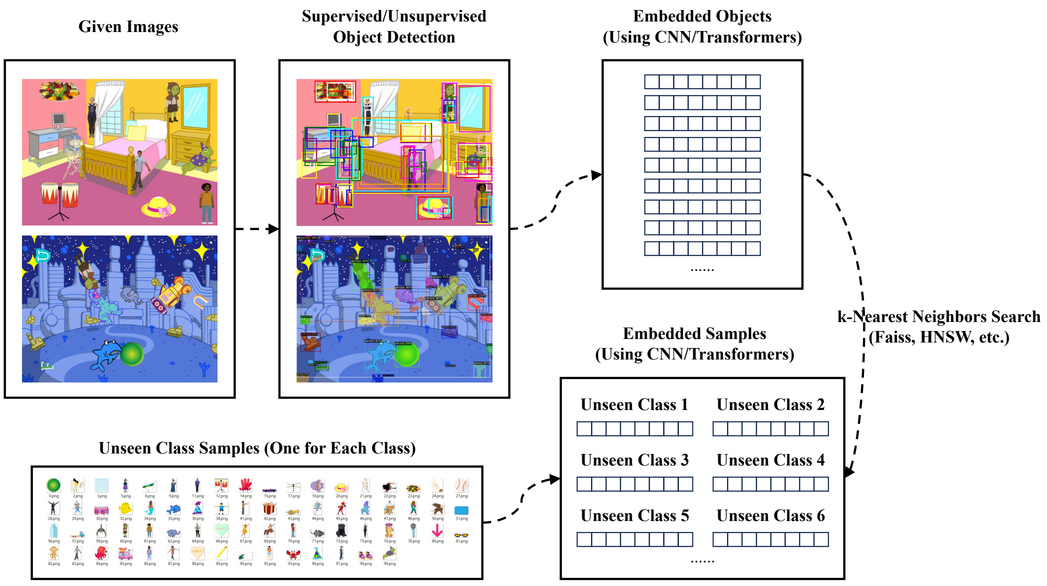
Given Images
Supervised/Unsupervised Object Detection
Embedded Objects (Using CNN/Transformers)
k-Nearest Neighbors Search (Faiss, HNSW, etc.)
Embedded Samples (Using CNN/Transformers)
Unseen Class 1
Unseen Class 2
Unseen Class 3
Unseen Class 4
Unseen Class 5
Unseen Class 6
Unseen Class Samples (One for Each Class)

Figure 4. Overview of our approach. Assuming that the embedded sprite vectors have been prepared, we (1) perform supervised or unsupervised object detection on the given images, (2) embed the detected objects using CNN- or Vision Transformer-based embedding techniques, and (3) find the corresponding multi-labels using k-nearest-neighbor search algorithms.

4.2. Object Detection

One of our primary ideas for multi-label image classification is to first leverage existing object detection models. It is important to note that we do not fine-tune these object detection models; instead, we use pre-trained ones. Clearly, these pre-trained object detection models may not identify specific sprite images, such as aliens, skeletons, characterized frogs, etc. However, our hypothesis is that even when the model cannot identify our sprites, we can detect most of them by adjusting the threshold of the object detection model. This approach might lead to finding a large number of false positives (objects but not our sprites), but these false positives can be effectively removed using our embedding techniques.

Figure 3 shows the example results of a popular object detection model named YOLOS-Ti [26]. Since this object detection model was pre-trained on ImageNet [27] and fine-tuned on COCO 2017 [10], it rarely has knowledge about our sprite images. Thus, even when the threshold is set very low, to 0.1, as shown in the third-column subfigures of Figure 3, the model seldom finds our sprite images. However, when we further lower the threshold, to 0.01, as shown in the fourth-column subfigures of Figure 3, we can detect almost all of our sprites, albeit with a large number of false positives.

Our alternative idea for improving object detection accuracy is to use unsupervised object detection algorithms, such as CutLER [28], instead of supervised object detection algorithms. Because these algorithms are not trained on specific target labels, they are expected to produce more true positives with fewer false positives. However, they might take more time to perform object detection, so the tradeoff between time and accuracy should be considered.

4.3. Embedding Techniques

The next step in our approach is to embed the detected objects, including the false positives, into vectors. To accomplish this, we can use various types of pre-trained embedding models, such as CNN-based models or Vision Transformer-based models. It is natural to expect a tradeoff between CNN-based models and Transformer-based models in terms of time and accuracy.

Note that in this process, we do not fine-tune the models for our specific data; we simply use pre-trained models, just as we did during object detection, to fulfill our requirements. Also, it should be noted that the same embedding techniques must be used for embedding our sprites in order to effectively find the k-nearest neighbors in the final step.

4.4. k-Nearest-Neighbor Search

The final step is to compare embedded objects and embedded sprites. The process is as follows:

- For each embedded object, find the m-nearest neighbors based on their cosine similarities. This process can be significantly faster when using approximate nearest-neighbor search algorithms, such as Faiss [24] or HNSW [25].
- Store all of the m-nearest neighbors along with the corresponding similarities in a sorted list.
- Retrieve the distinct k-nearest neighbors from the sorted list.

For example, let m = 69 and k = 9. In our dataset, there are a total of 69 sprite images, so there are 69 embedded sprites. Assuming that there are a total of 100 detected objects given in an image, there are 100 embedded objects. In this case, because m = 69, we compare all of the cosine similarities between embedded objects and embedded sprites. Then, the most similar and distinct embedded sprites are retrieved as results.

If the embedding models are effective, then the false positives generated by the object detection models are eliminated in this process. In the next section, we will present the corresponding experimental results.

5. Experiments

5.1. Experimental Setup

In our approach, various types of pre-training models can be utilized. As shown in Table 1, we selected nine pre-trained models for our experiments:

- We chose ResNet-18 [18] and VGG-16 [19] for embedding, due to their popularity in CNN-based models.
- For Vision Transformer-based embedding models, we selected five pre-trained models that are the most recent and popular embedding models found on Hugging Face (https://huggingface.co/ (accessed on 19 September 2023.)). Among them, one was developed by OpenAI; two were developed by Google [29]; and two were developed by Facebook [30]. Here, the model “clip-ViT-B-32” was developed for mapping texts and images into the same embedding space. All of the models from Google and Facebook utilize Vision Transformers [20] and ImageNet [27].
- As a supervised object detection model, we used YOLOS-Ti [26], which is the most trending model on Hugging Face at the time of writing. The characteristics of this model include its lightweight design; its model size is only 25 MB, allowing for easy use even on low-performance computers.
- For unsupervised object detection, we employed CutLER [28]. Known as the most efficient unsupervised object detection model at the time of writing, it is slower than

the lightweight supervised object detection model, YOLOS-Ti, but it is expected to significantly enhance accuracy in object detection.

It is important to note that we did not compare our approach with methods that do not use neural networks. For example, we did not compare it with ML-KNN [16], which uses k-nearest-neighbor algorithms but not neural networks. Additionally, we did not report the results obtained with SSIM (structural similarity index measure) [17], as it did not perform well in our experiments.

We conducted three types of experiments to demonstrate the effectiveness of our approach:

- As a baseline approach, we only use embedding techniques without the use of object detection models. Because no objects are detected with this approach, the entire image (backdrops with added sprites) is embedded into vectors using various types of CNN- or Vision Transformer-based models.
- In addition to this baseline approach, we also experimented with our approaches that employ either supervised or unsupervised object detection models, as well as either CNN- or Vision Transformer-based embedding models.

We do not compare our method with existing approaches for multi-label image classification, because these approaches do not satisfy the requirements defined in the Introduction section. Thus, in our experiments, we established a new baseline and compared our methods with this baseline approach.

The hyperparameters were configured as follows: First, we set parameter "m" to 69, which allowed us to find the exact k-nearest neighbors for each query; recall that there are 69 sprites in our dataset. Given that the number of samples in our experiments was relatively low, we believe that using exact k-nearest-neighbor search is one of the best options, although there may be room for improvement in terms of computational time with the use of approximate k-nearest-neighbor search. Second, for object detection, we used thresholds of 0.01 for supervised object detection with YOLOS-Ti and 0.35 for unsupervised object detection with CutLER. The threshold for YOLOS-Ti was set particularly low because the sprites in our dataset are not highly relevant to the classes on which the object detection model was originally trained. Therefore, we needed to lower the threshold to maximize the identification of our sprites, as illustrated in Figure 3.

We performed the experiments using a low-performance laptop computer (11th Gen Intel$^{®}$ Core^{TM} (Intel, Santa Clara, CA, USA) i7-11800H @ 2.30 GHz, 16 GB RAM, RTX 3080 Laptop GPU). This aligns with Requirement 4, as our aim is to present an efficient and lightweight approach.

In our experiments, we used four evaluation measures as follows:

- "E. Sprites (sec)": The average elapsed time for embedding all 69 sprite images, measured in seconds.
- "E. Backdrops (sec)": The average elapsed time for embedding all of the detected objects in a given backdrop image with added sprites, measured in seconds.
- "Finding kNN (sec)": The average elapsed time for finding all of the k-nearest neighbors for a given backdrop, measured in seconds. Note that we did not use any techniques for faster computation, such as Faiss or HNSW.
- "Accuracy (%)": We utilize an "accuracy" evaluation metric instead of the mAP (mean Average Precision) metric. This choice is driven by the observation that the frequency of false positives can differ substantially depending on the object detection model used, complicating the task of equitably assessing our techniques. Let S be the set of all sprites added to backdrops, and let $C \subset S$ be the set of correctly identified sprites among the top-$|S|$ predicted sprites. Then, accuracy ($\mathcal{A}$) is defined as

$$\mathcal{A} = \frac{|C|}{|S|}$$

Table 1. Models used in our experiments and their corresponding statistics.

Type	Model	Model Size (MB)	Dimension
CNN-based embedding	ResNet-18	45	512
	VGG-16	528	512
Transformer-based embedding	clip-ViT-B-32	577	512
	google/vit-base-patch16-224-in21k	330	768
	google/vit-base-patch32-224-in21k	336	768
	facebook/dino-vits8	83	384
	facebook/dino-vitb16	327	768
Supervised object detection	YOLOS-Ti	25	n/a
Unsupervised object detection	CutLER	548	n/a

5.2. Experimental Results

Recall that the baseline approaches do not use object detection models. That is to say, they embed the entire backdrop images, along with their added sprites, into vectors. Table 2 shows the experimental results of these baseline approaches using various CNN-/Vision Transformer-based embedding models.

Table 2. Experimental results of the baseline approaches that do not use object detection models but use CNN- or Vision Transformer-based embedding models.

Baseline Model (No Object Detection)	# Objects	E. Sprites (sec)	E. Backdrops (sec)	Finding kNN (sec)	Accuracy (%)
ResNet-18	3	**0.00392**	0.01194	0.00091	9.07
	6	**0.00394**	0.01268	0.00105	13.38
	9	**0.00395**	0.01278	0.00091	16.07
VGG-16	3	0.00598	**0.00751**	0.00057	8.63
	6	0.00616	**0.00823**	0.00034	12.27
	9	0.00626	**0.00830**	0.00033	15.52
clip-ViT-B-32	3	0.01899	0.02922	0.00111	**23.80**
	6	0.01823	0.03004	0.00125	**23.48**
	9	0.01688	0.03070	0.00128	**24.74**
google/vit-base-patch16-224-in21k	3	0.12293	0.12377	0.00052	13.03
	6	0.12042	0.12344	0.00071	15.75
	9	0.12167	0.12510	0.00060	19.73
google/vit-base-patch32-224-in21k	3	0.04717	0.05188	0.00068	11.67
	6	0.04744	0.05056	0.00051	15.42
	9	0.04645	0.05224	0.00079	18.86
facebook/dino-vits8	3	0.16925	0.17861	**0.00027**	15.73
	6	0.16584	0.17353	**0.00033**	17.62
	9	0.17505	0.17824	**0.00029**	19.02
facebook/dino-vitb16	3	0.11947	0.12290	0.00081	12.40
	6	0.11988	0.12233	0.00049	15.82
	9	0.11897	0.12344	0.00076	17.93

As expected, since the CNN-based model ResNet-18 is the most lightweight among our embedding models, it performed the fastest, but its accuracy values were the lowest. Additionally, the model that showed the best accuracy is the clip-ViT-B-32 model. This result could be attributed to the fact that only this model uses different types of image data for training, instead of ImageNet.

The overall results are quite surprising, as they show very low accuracy across all models. This finding seems to contradict the results in [5], where a similar approach was effectively used to educate children about machine learning. In that study, children prepared their own images and used models that did not include object detection, yet there were no complaints about the models' performance. These results suggest that our dataset is likely more complex, potentially due to the presence of colorful and distracting objects in the background images.

Table 3 shows the experimental results of our approaches based on the supervised object detection model, named YOLOS-Ti. The results demonstrate that the accuracy of our approaches greatly improved, approximately tripling from the baseline (maximum accuracy for the nine-sprite-added datasets increased from 24.74 to 75.84). In this case, one of the Google models outperformed the other models in terms of accuracy, which is different from the baseline result. Also, surprisingly, the lightweight model (ResNet-18) is comparable to the best Vision Transformer-based model in terms of accuracy. This may be interpreted as the Vision Transformer models we use not exploiting large enough datasets for their training.

Table 3. Experimental results of our approaches that utilize both the supervised object detection model named YOLOS-Ti and embedding models based on CNNs or Vision Transformers.

Our Model (with YOLOS-Ti)	# Objects	E. Sprites (sec)	E. Backdrops (sec)	Finding kNN (sec)	Accuracy (%)
ResNet-18	3	**0.00435**	**0.21694**	0.25879	75.27
	6	**0.00383**	**0.23732**	0.28784	74.12
	9	**0.00357**	**0.27198**	0.32854	73.84
VGG-16	3	0.00715	0.26346	0.26026	70.27
	6	0.00975	0.28967	0.29331	69.83
	9	0.00703	0.33154	0.32995	70.40
clip-ViT-B-32	3	0.02960	0.84996	0.26091	59.10
	6	0.02942	0.93125	0.28929	61.43
	9	0.02990	1.10837	0.32836	64.10
google/vit-base-patch16-224-in21k	3	0.09293	6.62304	0.36970	**75.43**
	6	0.10465	7.58749	0.40287	**75.03**
	9	0.09675	8.46929	0.45842	**75.84**
google/vit-base-patch32-224-in21k	3	0.04208	2.57006	0.36277	70.30
	6	0.03495	2.86321	0.41135	71.88
	9	0.04234	3.28558	0.46758	72.87
facebook/dino-vits8	3	0.14997	9.52359	**0.19336**	69.80
	6	0.14184	10.62992	**0.21481**	64.65
	9	0.14475	12.46685	**0.25093**	64.38
facebook/dino-vitb16	3	0.10165	7.12740	0.36584	71.17
	6	0.10426	7.87116	0.41065	66.82
	9	0.10952	8.45760	0.45791	66.59

The results also show that the elapsed times are significantly different from the baseline approach. Although the elapsed time for embedding the 69 sprites does not change substantially, the elapsed time for embedding detected objects of backdrops and the elapsed time for finding the k-nearest neighbors are significantly higher than the baseline. This means that there are many detected objects in the backdrops when using YOLOS-Ti; therefore, we must spend a lot of time embedding objects and finding the nearest neighbors for each embedded object. Obviously, most of the detected objects were false positives. However, by performing the process of finding the k-nearest neighbors, most of the false positives were removed, allowing us to achieve high accuracy. Also, even with the many false positives occurring in these results, when using a lightweight CNN-based embedding model, the elapsed time is affordable even when using a laptop computer. Especially if we used the approximate k-nearest-neighbor search algorithm, the elapsed time could be much shortened.

Table 4 illustrates the results of our approach when utilizing the unsupervised object detection model, CutLER. The findings reveal that our approach with the unsupervised object detection model consistently outperformed the supervised object detection model across all tested cases in terms of accuracy. For instance, within a dataset of three sprites, the baseline accuracy was 9.07, while our approach's accuracy reached 75.27 with YOLOS-Ti and an even higher value, 80.57, with CutLER. The benefits are not confined to accuracy alone; we also observed substantial reductions in the elapsed times required for embedding detected objects and finding the k-nearest neighbors. In the nine-sprite dataset, the times recorded for "E. Backdrops" and "Finding kNN" were 0.27198 and 0.32854 with YOLOS-Ti, and mere values of 0.03610 and 0.04508 with CutLER, respectively. This indicates that the use of the unsupervised model significantly reduces the false positives of the detected objects when employing CutLER while simultaneously increasing the true positives.

Table 4. Experimental results of our approaches that utilize both the unsupervised object detection model named CutLER and embedding models based on CNNs or Vision Transformers.

Our Model (With CutLER)	# Objects	E. Sprites (sec)	E. Backdrops (sec)	Finding kNN (sec)	Accuracy (%)
ResNet-18	3	**0.00325**	**0.03610**	0.04508	**80.57**
	6	**0.00328**	**0.04564**	0.05653	**79.85**
	9	**0.00322**	**0.05744**	0.07224	**78.71**
VGG-16	3	0.00860	0.04320	0.04765	74.67
	6	0.00896	0.05575	0.06140	74.50
	9	0.00980	0.07135	0.07719	74.81
clip-ViT-B-32	3	0.03090	0.12947	0.04445	72.57
	6	0.02920	0.16371	0.05634	73.60
	9	0.03044	0.20680	0.07236	75.22
google/vit-base-patch16-224-in21k	3	0.07387	0.94674	0.05959	77.97
	6	0.08900	1.13765	0.07898	77.10
	9	0.08695	1.52456	0.09990	78.19
google/vit-base-patch32-224-in21k	3	0.04069	0.42205	0.06110	73.07
	6	0.03170	0.54028	0.07988	74.87
	9	0.03496	0.67661	0.09998	75.77
facebook/dino-vits8	3	0.13350	1.37116	**0.03289**	75.07
	6	0.11385	1.79386	**0.04229**	70.70
	9	0.11850	2.38555	**0.05328**	69.26
facebook/dino-vitb16	3	0.10889	1.07077	0.06116	75.03
	6	0.06842	1.13841	0.07899	71.40
	9	0.06783	1.42612	0.09981	70.61

In summary, our experimental results underscore that the combination of ResNet-18 and CutLER yields the best performance on our dataset, excelling in both speed and accuracy. For scenarios where a small model size is prioritized, pairing ResNet-18 with YOLOS-Ti could prove beneficial, especially when further optimized with techniques such as Faiss or HNSW. Unfortunately, existing Vision Transformer-based embedding models fell short of expectations on our dataset. This might be attributed to the Transformer models' need for substantial data volumes to attain high accuracy, suggesting that further training on more extensive datasets could enhance their performance.

5.3. Analysis of Results

In our initial experiments, we set the threshold values to 0.01 for YOLOS-Ti and 0.35 for CutLER. To further investigate the impact of different thresholds, we conducted additional tests using a wider range of values: 0.01, 0.03, 0.05, 0.07, 0.09, 0.1, 0.3, 0.5, 0.7, and 0.9. Figure 5 provides a comparative analysis of the accuracy of YOLOS-Ti and CutLER under these different settings, while Figure 6 presents the time elapsed for each algorithm at these thresholds. In subsequent analyses, we employ ResNet-18 as our image-embedding algorithm due to its superior performance in earlier experiments. The dataset we used contains nine objects (sprites) in each input image.

Figures 5 and 6 indicate that higher object detection thresholds lead to lower accuracy but faster elapsed times for embedding backdrops. This finding is intuitive, as a higher threshold reduces the number of objects detected, thereby requiring fewer image embeddings. The figures also show that CutLER, being an unsupervised model, is significantly more effective than YOLOS-Ti on our dataset, which contains unseen class data. For CutLER, a moderate threshold suffices to achieve high accuracy, making a threshold of 0.3 suitable for multi-label classification. This closely aligns with the default threshold set in the original CutLER code. For YOLOS-Ti, however, even a low threshold of 0.1 does not yield accuracy higher than 70%. To make YOLOS-Ti applicable for our tasks, a very low threshold, such as 0.01, would be required.

To examine the results in greater detail, we present visualizations of object detection outcomes at varying thresholds—0.01, 0.05, 0.10, and 0.50—for both YOLOS-Ti (Figure 7) and CutLER (Figure 8). Figure 7 reveals that at a threshold of 0.50, YOLOS-Ti fails to detect any objects in the image. Even when the threshold is lowered to 0.10, it only accurately identifies four out of the nine objects present. To achieve comprehensive object detection using YOLOS-Ti, the threshold needs to be set as low as 0.01. Conversely, CutLER manages to identify seven out of the nine objects even at the relatively high threshold of 0.50.

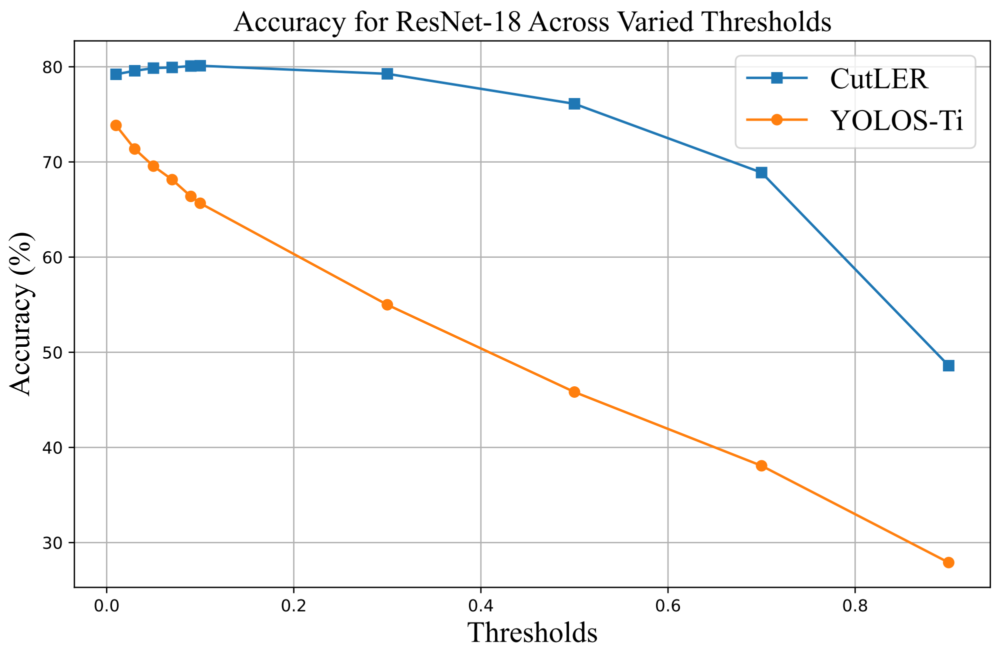

Figure 5. Comparison of accuracy between YOLOS-Ti and CutLER using ResNet-18 across varying thresholds.

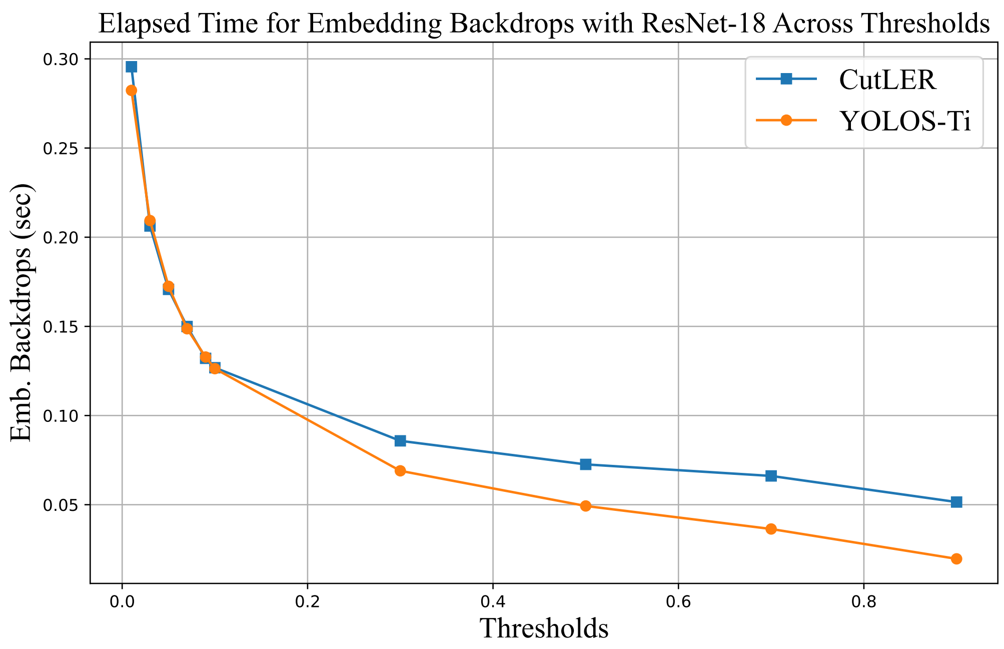

Figure 6. Comparison of elapsed time for embedding backdrops between YOLOS-Ti and CutLER Using ResNet-18 across varying thresholds.

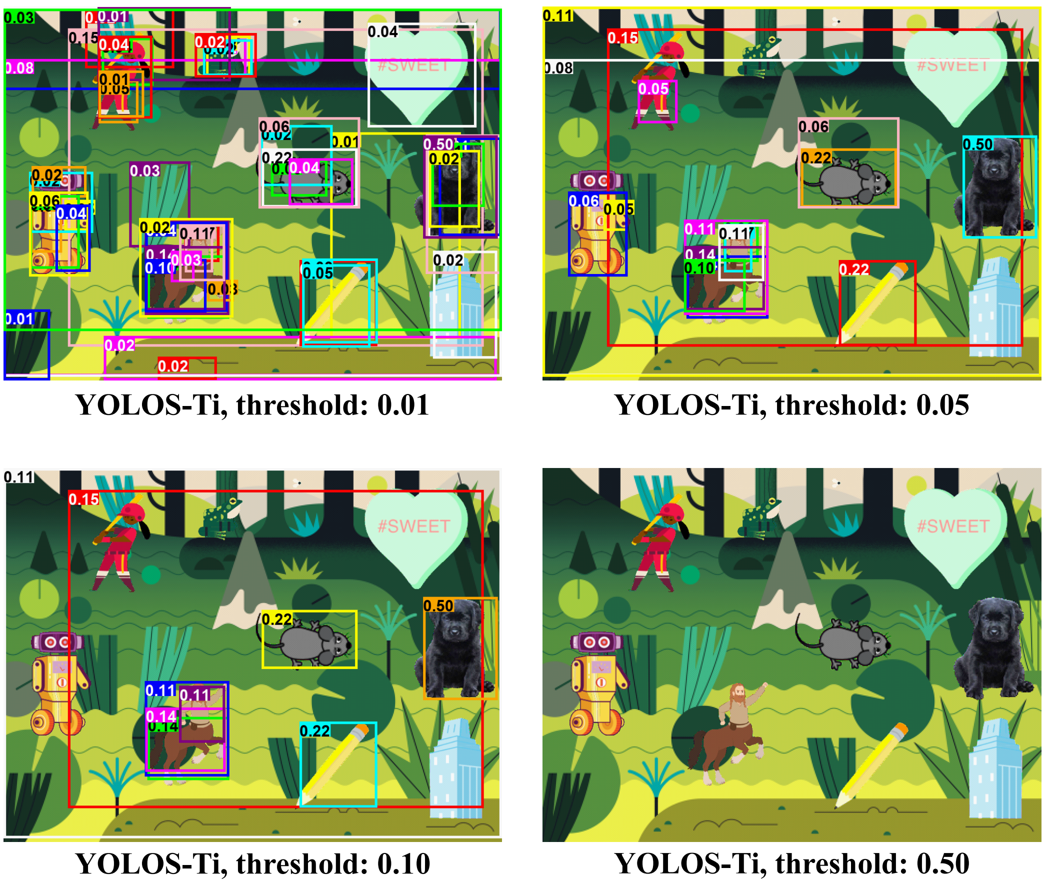
YOLOS-Ti, threshold: 0.01
YOLOS-Ti, threshold: 0.05
YOLOS-Ti, threshold: 0.10
YOLOS-Ti, threshold: 0.50

Figure 7. Examples of results obtained with the YOLOS-Ti model at various threshold levels. Because the model was not trained on our new dataset, it failed to detect the newly introduced objects even at a threshold of 0.5.

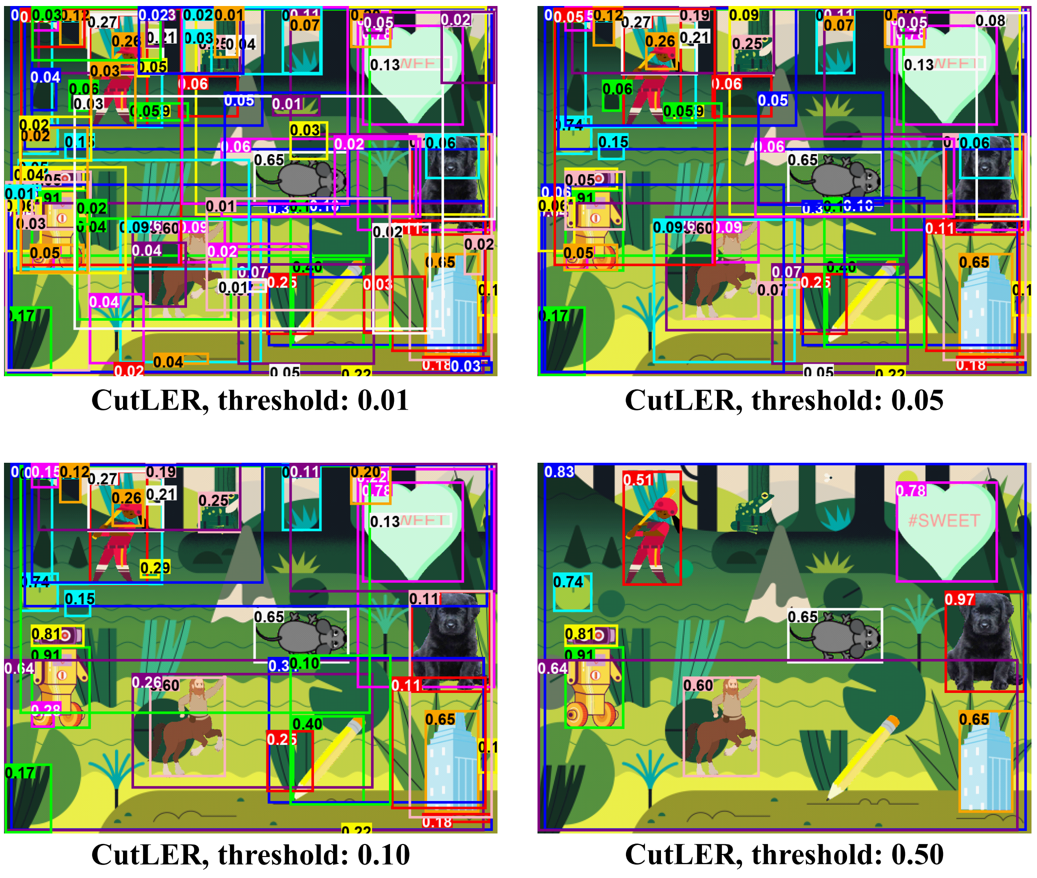

Figure 8. Examples of results obtained with the CutLER model at various threshold levels. Although the model was not trained on our new dataset, it still showed relative success in detecting newly introduced objects.

In our experiments, we found that ResNet-18 performed relatively well when object detection techniques were used, but its performance declined when they were not applied. To further investigate these findings, we conducted a small, supplementary experiment. First, we selected ten object images cropped from our dataset, which we referred to as "cropped data samples". We then added a white background to each of these images to create "not cropped data samples." Both sets of data samples are displayed in Figure 9.

Next, we embedded these images and identified the nearest neighbor for each. The results, shown in Figure 10, were consistent with our initial observations. Specifically, ResNet-18 performed well on the cropped data samples but was less effective with the not cropped data samples. In contrast, the CLIP-ViT-B-32 model maintained high accuracy across both types of data samples, confirming our earlier findings that this model performs relatively well when no object detection techniques are used.

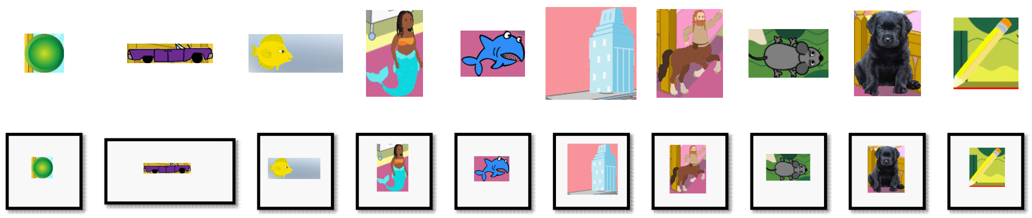

Figure 9. Ten cropped data samples (**top**) and ten not cropped data samples (**bottom**).

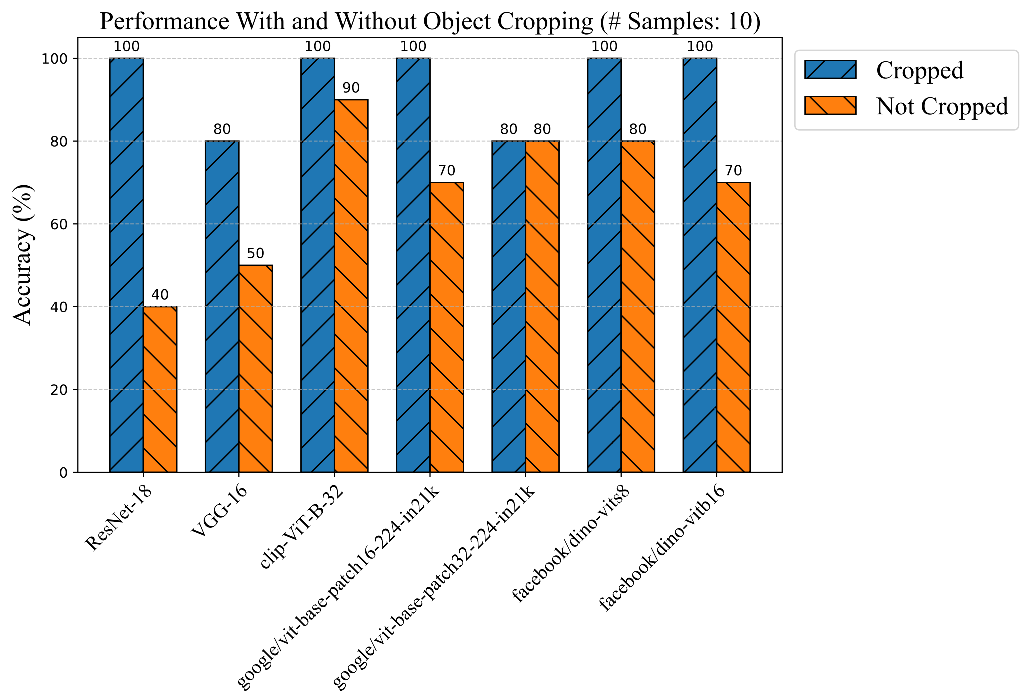

Figure 10. Accuracy comparison using the ten data samples, with and without object cropping.

5.4. Discussion

Our approach, though simple, successfully fulfills the four requirements outlined in the Introduction section:

- Firstly, the requirement concerning both "class-incremental learning" and "class-decremental learning" is met with our strategy. By exclusively utilizing pre-trained models without fine-tuning them on our specific dataset, our approach naturally excels at class-incremental/-decremental learning. In fact, we do not train the neural networks ourselves but harness the complexity of existing models by leveraging their pre-trained capabilities. For example, let us assume that 10 additional objects are added to our dataset, bringing the total to 79 objects. In this scenario, our approach allows us to perform multi-class image classification without the need for fine-tuning the model.
- Secondly, the requirement for "one-shot learning" is addressed effectively. Within our dataset, each class (or sprite) is represented by only one sprite image. Despite this limitation, experimental results have shown that our method performs admirably even with sparse data.
- Thirdly, our approach succeeds on unnatural datasets where the semantic relationships among labels are either non-existent or difficult to utilize. The random generation of our dataset precludes the use of semantic label relationships, yet our method still yields strong performance.
- Lastly, the need for compatibility with low-performance computers is also met. We demonstrate that our approach can be efficiently run on a standard laptop computer. Moreover, the use of lightweight models like ResNet-18 and YOLOS-Ti ensures that we can execute our tasks with both effectiveness and efficiency.

However, we also found one main limitation of our approaches, which we will address in future work: As shown in Figure 11, the unsupervised object detection model (CutLER) is very effective on transformed images (e.g., rotated images). However, even complex Vision Transformer-based pre-trained embedding models do not handle transformed images effectively. To address this problem with image transformations, we either need to embed more sprite images with transformations or train a more robust pre-trained model capable of handling image transformations.

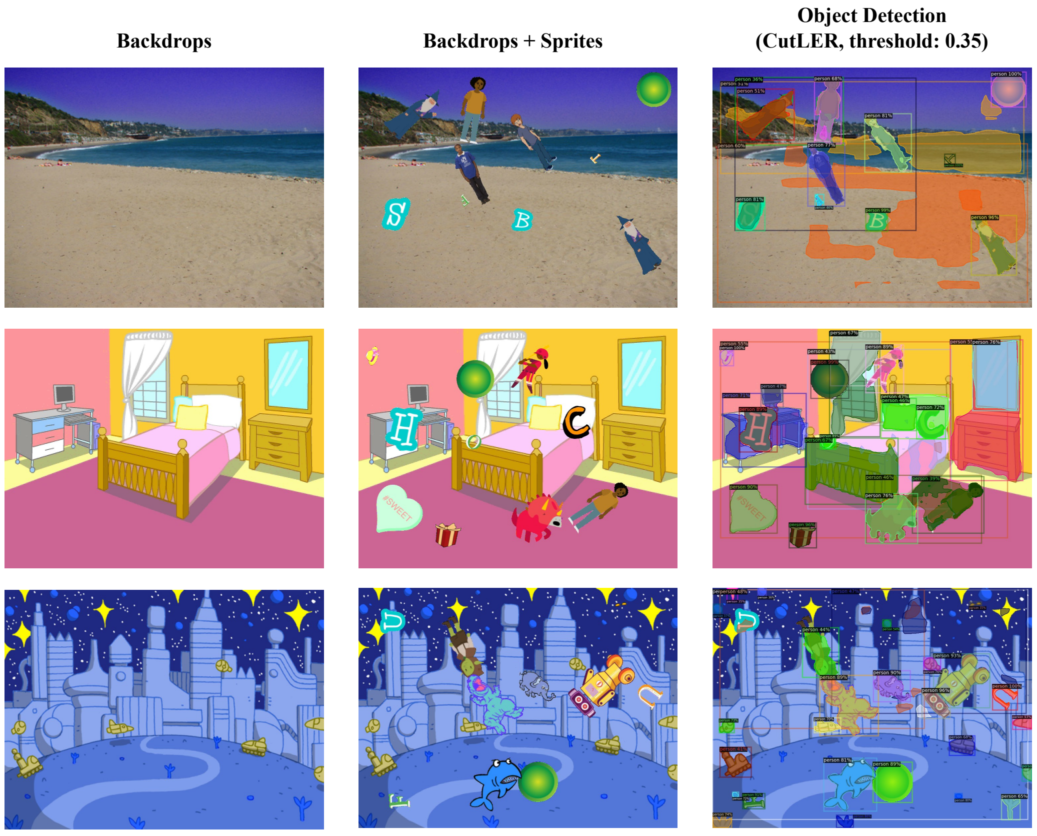

Figure 11. Example datasets with rotated sprites on the backdrops. Although the unsupervised object detection model (CutLER) excels at finding transformed sprites, the existing state-of-the-art Vision Transformer-based model struggles to capture the similarity between rotated images.

6. Conclusions

In this paper, we introduced a simple yet effective approach to one-shot class-incremental multi-class image classification. The core idea of our method involves utilizing pre-trained supervised or unsupervised object detection models, along with CNN- or Vision Transformer-based embedding models, to predict multiple labels. Notably, our approach does not require any fine tuning of the models, making it highly suitable for one-shot class-incremental learning. For our experiments, we devised a novel and complex dataset that existing methods, based on semantic relatedness among labels, could not handle. The results reveal that our approach achieves remarkable performance on this dataset. Although we identified an occurrence of false positives during object detection, especially when utilizing supervised object detection, we also discovered that these false positives could be substantially reduced by employing embedding techniques and identifying the k-nearest neighbors. Furthermore, our analysis showed that unsupervised object detection outperforms its supervised counterpart and that even lightweight model combinations (such as ResNet-18 and YOLOS-Ti) can perform well in this task.

One of the main barriers to achieving even more accurate results is the limitation of the embedding model, rather than the object detection model. Despite the fact that the current state-of-the-art unsupervised object detection model (CutLER) performs admirably, even with significant image transformations, the Vision Transformer-based embedding models did not meet our expectations. Consequently, our future work will focus on implementing a more robust Vision Transformer-based embedding model. We plan to explore the use of larger datasets and more efficient network structures to accommodate increasingly complex datasets.

Author Contributions: Conceptualization, Y.P. and Y.S.; methodology, Y.P. and Y.S.; investigation, Y.P. and Y.S.; data curation, Y.P. and Y.S.; writing—original draft preparation, Y.P. and Y.S.;

writing—review and editing, Y.P. and Y.S.; funding acquisition, Y.P. and Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National University Development Project funded by the Ministry of Education (Chuncheon National University of Education) in 2023.

Institutional Review Board Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, G.; Ji, Z.; Chang, Y.; Li, S.; Qu, X.; Cao, D. ML-ANet: A transfer learning approach using adaptation network for multi-label image classification in autonomous driving. *Chin. J. Mech. Eng.* **2021**, *34*, 78. [CrossRef]
2. Inoue, N.; Simo-Serra, E.; Yamasaki, T.; Ishikawa, H. Multi-label fashion image classification with minimal human supervision. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Montreal, BC, Canada, 11–17 October 2017; pp. 2261–2267.
3. Chen, H.; Miao, S.; Xu, D.; Hager, G.D.; Harrison, A.P. Deep hierarchical multi-label classification of chest X-ray images. In International Conference on Medical Imaging with Deep Learning, London, UK, 8–10 July 2019; pp. 109–120.
4. Vu, X.S.; Le, D.T.; Edlund, C.; Jiang, L.; Nguyen, H.D. Privacy-preserving visual content tagging using graph transformer networks. In Proceedings of the 28th ACM International Conference on Multimedia, Seattle, WA, USA, 12–16 October 2020; pp. 2299–2307.
5. Park, Y.; Shin, Y. Text Processing Education Using a Block-Based Programming Language. *IEEE Access* **2022**, *10*, 128484–128497. [CrossRef]
6. Zhou, F.; Huang, S.; Xing, Y. Deep semantic dictionary learning for multi-label image classification. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 3572–3580.
7. Nguyen, H.D.; Vu, X.S.; Le, D.T. Modular graph transformer networks for multi-label image classification. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 9092–9100.
8. Lanchantin, J.; Wang, T.; Ordonez, V.; Qi, Y. General multi-label image classification with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 16478–16488.
9. Guo, H.; Zheng, K.; Fan, X.; Yu, H.; Wang, S. Visual attention consistency under image transforms for multi-label image classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 729–739.
10. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
11. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
12. Lake, B.M.; Salakhutdinov, R.; Tenenbaum, J.B. Human-level concept learning through probabilistic program induction. *Science* **2015**, *350*, 1332–1338. [CrossRef] [PubMed]
13. Rebuffi, S.A.; Kolesnikov, A.; Sperl, G.; Lampert, C.H. icarl: Incremental classifier and representation learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2001–2010.
14. Park, Y.; Shin, Y. Tooee: A novel scratch extension for K-12 big data and artificial intelligence education using text-based visual blocks. *IEEE Access* **2021**, *9*, 149630–149646. [CrossRef]
15. Tao, X.; Hong, X.; Chang, X.; Dong, S.; Wei, X.; Gong, Y. Few-shot class-incremental learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 12183–12192.
16. Zhang, M.L.; Zhou, Z.H. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognit.* **2007**, *40*, 2038–2048. [CrossRef]
17. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef] [PubMed]
18. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
19. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
20. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16 $\times$ 16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
21. Radford, A. ; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning transferable visual models from natural language supervision. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 8748–8763.
22. Resnick, M.; Maloney, J.; Monroy-Hernández, A.; Rusk, N.; Eastmond, E.; Brennan, K.; Milner, A.; Rosenbaum, E.; Silver, J.; Silverman, B.; et al. Scratch: Programming for all. *Commun. ACM* **2009**, *52*, 60–67. [CrossRef]
23. Maloney, J.; Resnick, M.; Rusk, N.; Silverman, B.; Eastmond, E. The Scratch programming language and environment. *ACM Trans. Comput. Educ.* **2010**, *10*, 1–15. [CrossRef]

24. Johnson, J.; Douze, M.; Jégou, H. Billion-scale similarity search with GPUs. *IEEE Trans. Big Data* **2019**, *7*, 535–547. [CrossRef]
25. Malkov, Y.A.; Yashunin, D.A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *42*, 824–836. [CrossRef] [PubMed]
26. Fang, Y.; Liao, B.; Wang, X.; Fang, J.; Qi, J.; Wu, R.; Niu, J.; Liu, W. You only look at one sequence: Rethinking transformer in vision through object detection. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 26183–26197.
27. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
28. Wang, X.; Girdhar, R.; Yu, S.X.; Misra, I. Cut and learn for unsupervised object detection and instance segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 3124–3134.
29. Wu, B.; Xu, C.; Dai, X.; Wan, A.; Zhang, P.; Yan, Z.; Tomizuka, M.; Gonzalez, J.; Keutzer, K.; Vajda, P. Visual transformers: Token-based image representation and processing for computer vision. *arXiv* **2020**, arXiv:2006.03677.
30. Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; Joulin, A. Emerging properties in self-supervised vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 9650–9660.