Research Article

GMAIR: Unsupervised Object Detection Based on Spatial Attention and Gaussian Mixture Model

Weijin Zhu, Yao Shen, Mingqian Liu, and Lizeth Patricia Aguirre Sanchez

1Department of Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China
2Winning Health Technology Co., Ltd, Shanghai 200135, China

Correspondence should be addressed to Yao Shen; yshen@cs.sjtu.edu.cn

Received 7 May 2022; Revised 8 June 2022; Accepted 15 June 2022; Published 18 July 2022

1. Introduction

The perception of human vision is naturally hierarchical. We can recognize objects in a scene at a glance and classify them according to their appearances, functions, and other attributes. It is expected that an intelligent agent can also decompose scenes to meaningful object abstraction, which is known as an object detection task in machine learning. In the last decade, there have been significant developments in supervised object detection tasks. However, its unsupervised counterpart continues to be challenging.

Recently, there has been some progress in unsupervised object detection. Attend, infer, repeat (AIR, [1]), which is a variational autoencoder (VAE [2])-based method, achieved encouraging results. Spatially invariant AIR (SPAIR [3]) replaced the recurrent network in AIR with a convolutional network that attained better scalability and lower computational cost. SPACE [4], which combines spatial attention and scene-mixture approaches, performed better in background prediction.

Despite the recent progress in unsupervised object detection, the results of previous studies remain unsatisfactory. One of the reasons for this could be that previous studies on unsupervised object detection were mainly concentrated on object localization. They lacked analysis and evaluation of the “what” latent variables, which represent the attributes of objects. These variables are essential for many tasks such as clustering, image generation, and style transfer. Another important concern is that they do not directly reason about the category of objects in the scene, which is beneficial to know in many cases, unlike most of the studies on corresponding supervised tasks.

This study presents a framework for unsupervised object detection that can directly reason about the category and localization of objects in the scenes and provide an intuitive way to analyze the “what” latent variables by simply incorporating a Gaussian mixture prior assumption. In Section 2, we introduce the architecture of our framework, GMAIR. We introduce related works in Section 3. We analyze the “what” latent variables in Section 4.1 and Section...
2. Gaussian Mixture Attend, Infer, Repeat

In this section, we introduce our framework, GMAIR, for unsupervised object detection. GMAIR is a spatial attention model with a Gaussian mixture prior assumption for the "what" latent variables, and this enables the model to cluster discovered objects. An overview of GMAIR is presented in Figure 1.

2.1. Structured Object-Semantic Latent Representation. To attain object abstraction latent variables, the image is divided into $H \times W$ regions. Latent variables $z = (z_1, z_2, \ldots, z_{HW})$ are a concatenation of $HW$ latent variables, where $z_i$ is the latent variable for the $i$th region representing the semantic feature of the object centered in the $i$th region. Furthermore, for each region we divide $z_i$ into five separate latent variables, $z_i = (z_i^{\text{pres}}, z_i^{\text{what}}, z_i^{\text{cat}}, z_i^{\text{depth}}, z_i^{\text{depth}})$, where $z_i^{\text{pres}} \in \{0, 1\}$, $z_i^{\text{what}} \in \{1, 2, \ldots, K\}$, $z_i^{\text{cat}} \in \{0, 1\}^C$, $z_i^{\text{depth}} \in \mathbb{R}^4$, $z_i^{\text{depth}} \in \mathbb{R}^4$, $A$ is the dimension of "what" latent variables, and $C$ is the number of clusters. $z_i^{\text{what}}$ represents the existence of the object, $z_i^{\text{cat}}$ is a binary vector indicating the existence of the object and the $i$th region of an image. $z_i^{\text{depth}}$ is a real number specifying the relative depth of the object, and $z_i^{\text{cat}}$ is an one-hot vector for category.

GMAIR imposes a prior on those latent variables as follows:

$$p(z) = \prod_{i} p(z_i^{\text{pres}}) \cdot p(z_i^{\text{what}}) \cdot p(z_i^{\text{cat}} | z_i^{\text{what}}) \cdot p(z_i^{\text{depth}} | z_i^{\text{what}}, z_i^{\text{cat}}) \cdot p(z_i^{\text{depth}} | z_i^{\text{what}}, z_i^{\text{cat}})$$

2.2. Inference and Generation Model

2.2.1. Inference Model $q_\theta(z|x)$. In the inference model, latent variables conditional on data $x$ are modeled as follows:

$$q(z|x) = \prod_{i} q(z_i^{\text{pres}} | x) \cdot q(z_i^{\text{what}} | x) \cdot q(z_i^{\text{cat}} | x) \cdot q(z_i^{\text{depth}} | x)$$

During implementation, feature maps with dimension $H \times W \times D$ are extracted from a backbone network using data $x$ as input, where $D$ is the number of channels of feature maps. Further, the posteriors of $z_i^{\text{pres}}, z_i^{\text{what}}, z_i^{\text{cat}}$, and $z_i^{\text{depth}}$ are reasoned by pres-head, where-head, and depth-head, respectively. Input images are cropped into $H \times W$ glimpses by a spatial transformer network, and each of these is transferred to the cat-encoder module to generate posteriors of $z_i^{\text{cat}}$. Subsequently, we use the concatenation of the $i$th glimpse and $z_i^{\text{cat}} (1 \leq i \leq HW)$ as the input of the what-encoder to generate posteriors of $z_i^{\text{what}}$.

2.2.2. Generation Model $p_\theta(x|z)$. In the generation model, each $z_i^{\text{what}} (1 \leq i \leq HW)$ is changed back into a glimpse using a glimpse decoder. Then, a renderer combines HW glimpses to generate $\hat{x}$. We use the same render algorithm as in previous studies [1, 3].

2.3. The Loss Functions

2.3.1. Evidence Lower Bound. In general, we learn parameters of VAE jointly by maximizing the evidence lower bound (ELBO), which can be formulated as follows:

$$ELBO = \mathbb{E}_{q(z|x)} \left[ \log \frac{p(x, z)}{q(z|x)} \right]$$

where $f(x; \mu, \sigma^2) = (1/\sqrt{2\pi}) \exp(-(x-\mu)^2/2\sigma^2)$ is the probability density function of Gaussian distribution, and $\mu_k, \sigma_k (k = 1, C)$ are the mean and standard derivation of the $k$th Gaussian distribution. We let $\mu_k$ and $\sigma_k$ be learnable parameters that are jointly trained with other parameters. During the implementation, $\mu_k = \mu(z_i^{\text{what}})$ and $\sigma_k = \sigma(z_i^{\text{what}})$ if $z_i^{\text{what}} = 1$, where $\mu$ and $\sigma$ can be modeled as linear layers. They are called "what priors" module in Figure 1.

For other latent variables, $z_i^{\text{cat}}$ is modeled using a Bernoulli distribution, $\beta(p)$, where $p$ is the present probability. $z_i^{\text{what}}$ and $z_i^{\text{depth}}$ are modeled using normal distributions, $N(\mu_{\text{prior}}, \sigma_{\text{prior}})$ and $N(\mu_{\text{prior}}, \sigma_{\text{prior}})$, respectively. All priors of latent variables are listed in Table 1.
term. The regularization term can be further decomposed into five terms by substituting (1) and (3) into (4), and each of the five terms corresponds to the Kullback–Leibler divergence (or its expectation) between a type of latent variables and its prior:

$$E_{q(z|x)} \left[ \log \left( \frac{q(z|x)}{p(z)} \right) \right] = L_{\text{pres}} + L_{\text{where}} + L_{\text{depth}} + L_{\text{cat}} + L_{\text{what}},$$

(5)

The terms in (5) are as follows:

$$L_{\text{pres}} = \sum_{i=1}^{HW} KL^{\text{pres}},$$

(6)

$$L_{\text{where}} = \sum_{i=1}^{HW} q(z_{i}^{\text{pres}} = 1|x)KL^{\text{where}},$$

(7)

$$L_{\text{depth}} = \sum_{i=1}^{HW} q(z_{i}^{\text{pres}} = 1|x)KL^{\text{depth}},$$

$$L_{\text{cat}} = \sum_{i=1}^{HW} q(z_{i}^{\text{pres}} = 1|x)KL^{\text{cat}},$$

(8)

where $$KL^{x} = KL(q(z_{i}^{x})||p(z_{i}^{x}))$$. A complete derivation is given in Appendix A.

2.3.2. Overlap Loss. During actual implementation, we find that penalizing on overlaps of objects sometimes helps. Therefore, we introduce an auxiliary loss called overlap loss. First, we calculate HW images with size $$3 \times H_{\text{img}} \times W_{\text{img}}$$, where $$H_{\text{img}}$$ and $$W_{\text{img}}$$ are, respectively, the height and width of the input image, transformed by HW decoded glimpses by a spatial transformer network. The overlap loss is then calculated as the average of the sum subtract by the maximum for each $$H_{\text{img}} \times W_{\text{img}}$$ pixels.

This loss, inspired by the boundary loss in SPACE [4], is utilized to penalize if the model tries to split a large object into multiple smaller ones. However, we achieve this using a
different calculation method that incurs a lower computational cost.

2.3.3. Total Loss. The total loss is as follows:

\[ L = \sum_{x \in S} \alpha_x L_x, \]  

(11)

where \( S = \{ \text{recon, overlap, pres, where, depth, cat, what} \} \), and \( \alpha \) is the coefficients of the corresponding loss terms.

3. Related Works

Several studies on unsupervised object detection have been conducted, including spatial attention methods such as AIR [1], SPAIR [3], and SPACE [4], and scene-mixture methods such as MONet [5], IODINE [6], GENESIS [7], and GENESIS-v2 [8]. Most of them including our work are based on the VAE [2] model.

3.1. AIR. The AIR framework uses a VAE-based hierarchical probabilistic model marking a milestone in unsupervised scene understanding. In AIR, latent variables are structured into groups of latent variables \( z_{i,N} \), for \( N \) discovered objects, each of which consists of “what,” “where,” and “presence” variables. A recurrent neural network is used in the inference model to produce \( z_{i,N} \), and there is a decoder network for decoding the “what” variables of each object in the generation model. A spatial transformer network [9] is used for rendering.

3.2. SPAIR. Because AIR attends one object at a time, it does not scale well to scenes that contain many objects. SPAIR attempted to address this issue by replacing the recurrent network with a convolutional network that follows a spatially invariant assumption. Like YOLO ([10]), in SPAIR, the locations of objects are specified relative to local grid cells. SPAIR achieved a better performance and scalability than AIR.

3.3. MONet. MONet [5] is a scene-mixture model for unsupervised scene decomposition. Like AIR, MONet also uses a recurrent neural network to infer objects. However, it learns the attention masks to obtain the segmentation results of objects instead of the bounding boxes.

3.4. IODINE. IODINE [6] is also a scene-mixture model for unsupervised scene decomposition. Unlike MONet, which infers a mask for one object at a time, in IODINE, all segmentation results are inferred simultaneously. A recurrent neural network is used to refine the segmentation results further.

3.5. GENESIS. GENESIS [7] replaced the recurrent neural networks in previous works with the convolutional neural networks. This improvement makes GENESIS more scalable to inputs with a larger number of scene components.

3.6. GENESIS-v2. GENESIS-v2 [8] also performs unsupervised object segmentation. It is similar to GENESIS but uses a parameter-free clustering algorithm to avoid iterative refinements.

The scene-mixture models such as MONet, IODINE, GENESIS, and GENESIS-v2 perform segmentation instead of explicitly finding the \( z_{where} \) location of the objects. The bounding boxes of the discovered objects need to be calculated by the object masks.

3.7. SPACE. SPACE [4] employs a combination of both methods. It consists of a spatial attention model for the foreground and a scene-mixture model for the background. By detecting the foreground and background separately, SPACE achieved a better detection result when the background is complicated.

In the area of deep unsupervised clustering, recent methods include AAE [11], GMVAE [12], and IIC [13]. AAE combines the ideas of generative adversarial networks and variational inference. GMVAE uses a Gaussian mixture model as a prior distribution. In IIC, objects are clustered by maximizing mutual information of pairs of images. All of them show promising results on unsupervised clustering.

GMAIR incorporates a Gaussian mixture model for clustering, similar to the GMVAE framework (the authors also refer to a blog post (http://ruishu.io/2016/12/25/gmvaes/) published by Rui Shu). It is worth noting that our attempt may simply be a choice among many given options. Unless previous research, our main contribution is to show the feasibility of performing clustering and localization simultaneously. Moreover, our method provides a simple and intuitive way to analyze the mechanics of the detection process.

4. Models and Experiments

The experiments were divided into three parts: (a) the analysis of “what” representation and clustering along with the iterations, (b) image generation, and (c) quantitative evaluation of the models.

We evaluate the models on two datasets:

(i) MultiMNIST: A dataset generated by placing 1–10 small images randomly chosen from MNIST (a standard handwritten digits dataset [14]) to random positions on 128 × 128 empty images.

(ii) Fruit2D: A dataset collected from a real-world game. In the scenes, there are 9 types of fruits of various sizes. There is a large difference between both the number and the size of small objects and large objects. The ratio of the size of the largest type of objects to that of the smallest type of objects is \( \sim 6 \), and there are \( \sim 31 \) times objects in the smallest size than in the largest size. These settings make it difficult to perform localization and clustering.

In the experiments, we compared GMAIR with two models, SPAIR and SPACE, both of which achieve state of the art in unsupervised object detection in localization.
performance. Separated Gaussian mixture models are applied to the “what” latent variables generated by the compared models to obtain the clustering results. We set the number of clusters C = 10 and Monte Carlo samples M = 1 except as otherwise defined for all experiments. All experiments were conducted on a Ubuntu 16.04.6 server with an Intel(R) Xeon(R) Silver 4110 CPU with 8 cores and 2 TITAN RTX GPUs. We present the details of the models in Appendix B.

It is worth mentioning that the model sometimes successfully locates an object and encloses it with a large box. In that case, IoU between the ground truth and the predicted one will be small and, therefore, will not count as a correct bounding box when calculating AP. We fix this issue by removing the empty area in generated glimpses to obtain the real size of predicted boxes.

4.1. “What” Representation and Cluster Analysis. We conducted the experiments using the MultiMNIST dataset. We ran GMAIR for 440k iterations and observed the change in the values of the average precision (AP) of bounding boxes, accuracy (ACC), and normalized mutual information (NMI) of clustering until 100k iterations. We also visualized the “what” latent variables in the latent space during the process, as shown in Figure 2. Although all values continued to increase even after 100k iterations, the visualization results were similar to those at the 100k iteration. For integrity, we reserved the results from 100k to 440k iterations in Appendix D. Details of calculating the AP, ACC, and NMI are discussed in Appendix C.

The results showed that at an early stage (~10k iterations) of training, models can already locate objects well with AP > rbin0.9 (Figure 2(a)). At the same time, $z^{what}$, representations of objects, was still evolving, and the results of clustering (in Figure 2(b)) were not desirable ((ACC, NMI) was (0.24, 0.15)); the digits were a blur in Figure 2(f). After 50k and 100k iterations of training, the clustering effect of $z^{what}$ was increasingly apparent, and the digits were clearer (Figures 2(g) and 2(h)). The clustering results ((ACC, NMI) was (0.55, 0.43) at 50k and (0.65, 0.55) at 100k iterations) were improved (Figures 2(c) and 2(d)).

The explanation relates to the fact that while the model learns general features of objects and can locate the objects, object classification presents a new challenge for the model. As a result, the model is forced to cluster the feature vector of discovered objects into a small number of categories, which requires the model to learn the similarities between objects and thus requires a large number of training iterations.

It should be noted that even if the clustering effect of $z^{what}$ is sufficiently enough, the model may fail to locate the centers of clusters (e.g., the large cluster in light red in Figure 2(d)), leading to poor clustering results. In the worst case, the model may learn to converge all $\mu_k$, $\sigma_k$ ($1 \leq k \leq C$) to the same values, $\mu^*$, $\sigma^*$, and the Gaussian mixture model may degenerate to a single Gaussian distribution, $\mathcal{N}(\mu^*, \sigma^*)$, resulting in a miserable clustering result. In general, we found that this phenomenon usually occurs at the early stage of training and can be avoided by adjusting the learning rate of relative modules and the coefficients of the loss functions.

4.2. Image Generation. It is expected that $\mu_k$ ($1 \leq k \leq C$) represents the average feature of the $k$th type of objects, and $z^{what}_i$ latent variable can be decomposed into the following:

$$z^{what}_i = z^{avg}_i + z^{local}_i.$$  

$z^{avg}_i = \mu_k$ if the $i$th object is in the $k$th category, and $z^{local}_i$ represents the local feature of the object. By altering $z^{avg}_i$ or $z^{local}_i$, we should obtain new objects that belong to other categories or the same category with different styles, respectively. In the experiment, we altered $z^{avg}_i$ and $z^{local}_i$ and observed the generated images for each object, as shown in Figure 3. In Figure 3(a), objects in each cluster correspond to a type of digit, which is exactly what we expected (except for digit 8 in column 3). In Figure 3(b), categories with a large number of objects are grouped into multiple clusters, while categories with a small number are grouped into one cluster. This is due to the significant difference in number between various types. However, objects in a cluster come from a category in general.

The structure of GMAIR ensures its ability to control object categories, object styles, and the positions of each object of the generated images by altering $z^{avg}_i$, $z^{local}_i$, and $z^{where}_i$. Examples are shown in Figure 4.

This could provide a new approach for tasks such as style transfer, image generation, and data augmentation. Note that previous methods such as AIR, SPAIR, and its variants can also obtain similar results, but we achieve them in finer granularity.

4.3. Quantitative Evaluations. We quantitatively evaluate the models in terms of the AP of bounding boxes and ACC and NMI of the clusters, and the results are listed in Table 2. In the first part, we show the results of GMVAE for unsupervised clustering on MNIST dataset for comparison. In the second part and the third part, we compare GMAIR to the state-of-the-art models for unsupervised object detection on MultiMNIST and Fruit2D dataset, respectively. The clustering results of SPAIR and SPACE are obtained by the Gaussian mixture models (GMMs). The results show that GMAIR achieves competitive results on both clustering and localization.

4.4. Ablation Study on the Importance of a Gaussian Mixture. The most significant difference between GMAIR and other models is that GMAIR uses a Gaussian mixture model for the “what” latent variables. To investigate whether the Gaussian mixture structure does affect the “what” representation, we also visualized “what” latent variables in latent space generated by SPAIR and SPACE, which have a similar structure of GMAIR and use a Gaussian model instead of a Gaussian mixture model for “what” latent variables. The results are shown in Figure 5. They show that without a Gaussian mixture model, “what” latent variables of different categories tend to follow a single distribution. This is
Figure 2: “What” representation and cluster analysis. (a) Average precision (AP), accuracy (ACC), and normalized mutual information (NMI) during training. (b-d) Visualized “what” latent space by t-SNE ([15]) at 10k, 50k, and 100k iterations, respectively. Each small dot represents a sample of $\mathbf{z}_{\text{what}}$, and different colors represent the ground truth categories of the corresponding objects. The large dots are $\mathbf{\mu}_k$ ($1 \leq k \leq C$) described in Section 2.1, and each of these can be seen as the center of a cluster. The closures represent results of clustering, which are closures of the closest $\mathbf{z}_{\text{what}}$ points to $\mathbf{\mu}_k$ that are assigned to the $k$th cluster (where $1 \leq k \leq C$ and we choose $n = 200$). The color of $\mathbf{\mu}_k$ ($1 \leq k \leq C$) and closures are decided by a matching algorithm such that a maximum number of $\mathbf{z}_{\text{what}}$ are correctly classified to the ground truth label. (e) Sample of original image. (f-h) Samples of generated image at 10k, 50k, and 100k iterations, respectively. (a) (a) AP (IoU = 0.5), ACC, and NMI during training. (b) “What” latent space, at 10k iterations. (c) “What” latent space, at 50k iterations. (d) “What” latent space, at 100k iterations. (e) Original image. (f) Generated image, at 10k iterations. (g) Generated image, at 50k iterations. (h) Generated image, at 100k iterations.
Figure 3: Generated objects by varying $z_{\text{avg}}$ and $z_{\text{local}}$. The horizontal axis represents varying $z_{\text{avg}}$, and the vertical axis represents varying $z_{\text{local}}$, on both (a) and (b). (a) MultiMNIST. (b) Fruit2D.

Figure 4: Continued.
Table 2: Quantitative results on localization (AP) and clustering (accuracy and NMI).

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset</th>
<th>AP (%), IoU = 0.5</th>
<th>ACC (%)</th>
<th>NMI (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIC</td>
<td>MNIST</td>
<td>—</td>
<td>98.4 ± 0.652</td>
<td>—</td>
</tr>
<tr>
<td>AAE (C = 16)</td>
<td>MNIST</td>
<td>—</td>
<td>90.45 ± 2.05</td>
<td>—</td>
</tr>
<tr>
<td>AAE (C = 30)</td>
<td>MNIST</td>
<td>—</td>
<td>95.90 ± 1.13</td>
<td>—</td>
</tr>
<tr>
<td>GMVAE (M = 1)</td>
<td>MNIST</td>
<td>—</td>
<td>77.78 ± 5.75</td>
<td>—</td>
</tr>
<tr>
<td>GMVAE (M = 10)</td>
<td>MNIST</td>
<td>—</td>
<td>82.31 ± 3.75</td>
<td>—</td>
</tr>
<tr>
<td>GMAIR</td>
<td>MultiMNIST</td>
<td>97.3 ± 0.10</td>
<td>80.4 ± 0.48</td>
<td>75.5 ± 0.66</td>
</tr>
<tr>
<td>SPAIR + GMM</td>
<td>MultiMNIST</td>
<td>90.3</td>
<td>59.4 ± 1.50</td>
<td>56.3 ± 1.41</td>
</tr>
<tr>
<td>SPACE + GMM</td>
<td>MultiMNIST</td>
<td>96.7</td>
<td>68.8 ± 3.43</td>
<td>65.8 ± 2.85</td>
</tr>
<tr>
<td>GMAIR</td>
<td>Fruit2D</td>
<td>84.9 ± 1.56</td>
<td>90.9 ± 0.32</td>
<td>85.7 ± 1.25</td>
</tr>
<tr>
<td>SPAIR + GMM</td>
<td>Fruit2D</td>
<td>83.3</td>
<td>88.1 ± 0.70</td>
<td>78.4 ± 0.51</td>
</tr>
<tr>
<td>SPACE + GMM</td>
<td>Fruit2D</td>
<td>93.8</td>
<td>95.0 ± 1.99</td>
<td>87.0 ± 2.20</td>
</tr>
</tbody>
</table>

Figure 4: Generated images by varying attributes and locations of objects. Columns 1 to 5 are numbered from left to right. Column 1 shows original images. Column 2 shows the generated images without varying $z^{avg}$, $z^{local}$, and $z^{share}$. Column 3 presents images generated by setting all $z^{avg}$ to the same random $\mu_k (1 \leq k \leq C)$. Column 4 depicts images generated by varying $z^{local}$. Column 5 shows images generated by applying a random shuffle to $z$. (a) MultiMNIST. (b) Fruit2D.

Figure 5: "What" representation of (a) SPAIR and (b) SPACE by t-SNE. Each dot represents a sample of $z^{what}$, and different colors represent the ground truth categories of the corresponding objects. (a) "What" latent space of SPAIR. (b) "What" latent space of SPACE.
### Table 3: Architecture of backbone.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Type</th>
<th>Size</th>
<th>Act./Norm.</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet</td>
<td>ResNet18 (w/o fc)</td>
<td>512 × 4 × 4</td>
<td>ReLU/BN</td>
<td>512 × 4 × 4</td>
</tr>
<tr>
<td>Deconv layer 1</td>
<td>Deconv</td>
<td>128</td>
<td>ReLU/BN</td>
<td>128 × 8 × 8</td>
</tr>
<tr>
<td>Deconv layer 2</td>
<td>Deconv</td>
<td>64</td>
<td>ReLU/BN</td>
<td>64 × 16 × 16</td>
</tr>
</tbody>
</table>

### Table 4: Architectures of pres/depth/where-head.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Type</th>
<th>Size</th>
<th>Act./Norm.</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hidden</td>
<td>Conv</td>
<td>[3 × 3, 128] × 3</td>
<td>ReLU</td>
<td>128 × 16 × 16</td>
</tr>
<tr>
<td>Output</td>
<td>Conv</td>
<td>1 × 1, (1/1/4)</td>
<td></td>
<td>(1/1/4) × 16 × 16</td>
</tr>
</tbody>
</table>

### Table 5: Architectures of what/Cat-encoder.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Type</th>
<th>Size</th>
<th>Act./Norm.</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Flatten</td>
<td>(3 × 32 × 32) = 3072</td>
<td>ReLU</td>
<td>3072</td>
</tr>
<tr>
<td>Layer 1</td>
<td>Linear</td>
<td>3072 × 128</td>
<td>ReLU</td>
<td>128</td>
</tr>
<tr>
<td>Layer 2</td>
<td>Linear</td>
<td>128 × 256</td>
<td>ReLU</td>
<td>256</td>
</tr>
<tr>
<td>Layer 3</td>
<td>Linear</td>
<td>256 × 512</td>
<td>ReLU</td>
<td>512</td>
</tr>
<tr>
<td>Output</td>
<td>Linear</td>
<td>512 × (A/C)</td>
<td></td>
<td>(A/C)</td>
</tr>
</tbody>
</table>

### Table 6: Architecture of glimpse decoder.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Type</th>
<th>Size</th>
<th>Act./Norm.</th>
<th>Output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Linear</td>
<td>A × 256</td>
<td>ReLU</td>
<td>256 × 1 × 1</td>
</tr>
<tr>
<td>Layer 1</td>
<td>Deconv</td>
<td>128</td>
<td>ReLU/GN(8)</td>
<td>128 × 2 × 2</td>
</tr>
<tr>
<td>Layer 2</td>
<td>Deconv</td>
<td>128</td>
<td>ReLU/GN(8)</td>
<td>128 × 4 × 4</td>
</tr>
<tr>
<td>Layer 3</td>
<td>Deconv</td>
<td>64</td>
<td>ReLU/GN(8)</td>
<td>64 × 8 × 8</td>
</tr>
<tr>
<td>Layer 4</td>
<td>Deconv</td>
<td>32</td>
<td>ReLU/GN(8)</td>
<td>32 × 16 × 16</td>
</tr>
<tr>
<td>Conv</td>
<td>Conv</td>
<td>3 × 3, 32</td>
<td>ReLU/GN(8)</td>
<td>32 × 16 × 16</td>
</tr>
<tr>
<td>Layer 5</td>
<td>Deconv</td>
<td>16</td>
<td>ReLU/GN(4)</td>
<td>16 × 32 × 32</td>
</tr>
<tr>
<td>Output</td>
<td>Conv</td>
<td>3 × 3, 3</td>
<td></td>
<td>3 × 32 × 32</td>
</tr>
</tbody>
</table>

### Table 7: Base hyperparameters.

<table>
<thead>
<tr>
<th>Description</th>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base box size</td>
<td>(a_h, a_w)</td>
<td>(72, 72)</td>
</tr>
<tr>
<td>Batch size</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>Dim. of z^what</td>
<td>A</td>
<td>256</td>
</tr>
<tr>
<td>Dim. of z^cat</td>
<td>C</td>
<td>10</td>
</tr>
<tr>
<td>Glimpse size</td>
<td>(H_obj, W_obj)</td>
<td>(32, 32)</td>
</tr>
<tr>
<td>Learning rate</td>
<td></td>
<td>[5e − 5, 1e − 4]</td>
</tr>
<tr>
<td>Loss coeff. of L^cat</td>
<td>σ^cat</td>
<td>1</td>
</tr>
<tr>
<td>Loss coeff. of L^overlap</td>
<td>σ^overlap</td>
<td>2 → 0</td>
</tr>
<tr>
<td>Loss coeff. of L^depth</td>
<td>σ^depth</td>
<td>1</td>
</tr>
<tr>
<td>Loss coeff. of L^pres</td>
<td>σ^pres</td>
<td>1</td>
</tr>
<tr>
<td>Loss coeff. of L^recon</td>
<td>σ^recon</td>
<td>8, 16</td>
</tr>
<tr>
<td>Loss coeff. of L^where</td>
<td>σ^where</td>
<td>1</td>
</tr>
<tr>
<td>Loss coeff. of L^z^cat</td>
<td>π^z^cat</td>
<td>((1/C),..., (1/C))</td>
</tr>
<tr>
<td>Prior on z^cat</td>
<td></td>
<td>(µ^{z^cat}_prior, σ^{z^cat}_prior)</td>
</tr>
<tr>
<td>Prior on z^depth</td>
<td></td>
<td>(µ^{depth}_prior, σ^{depth}_prior)</td>
</tr>
<tr>
<td>Prior on z^pres</td>
<td></td>
<td>p</td>
</tr>
<tr>
<td>Prior on z^where</td>
<td></td>
<td>(µ^{where}_prior, σ^{where}_prior)</td>
</tr>
<tr>
<td>Prior on z^what</td>
<td></td>
<td>(µ^{what}_prior, σ^{what}_prior)</td>
</tr>
</tbody>
</table>
Figure 6: “What” representation and cluster analysis after 100k iterations. (a) Average precision (AP), accuracy (ACC), and normalized mutual information (NMI) during training. (b-d) Visualized “what” latent space by t-SNE ([15]) at 220k, 330k, and 440k iterations, respectively. (e) Sample of original image. (f-h) Samples of generated image at 220k, 330k, and 440k iterations, respectively. (a) AP (IoU ≥ 0.5), ACC, and NMI during training. (b) “What” latent space, at 220k iterations. (c) “What” latent space, at 330k iterations. (d) “What” latent space, at 440k iterations. (e) Original image. (f) Generated image, at 220k iterations. (g) Generated image, at 330k iterations. (h) Generated image, at 440k iterations.
reasonable since we try to minimize the KL divergence of $z^{\text{what}}$ and a Gaussian model in these models. Therefore, the Gaussian mixture model structure helps to gather “what” latent variables of a category and keep those of different categories away from each other.

5. Limitations and Societal Impact

The biggest limitation of GMAIR is that it can currently only be applied to simple compositing scenarios, such as games. However, the automatic mining of object localization and category from simple scenes is still a step towards artificial general intelligence. Although the current detection performance of GMAIR for complex scenes is poor, it cannot be ruled out that it may be applied to some scenes for unethical automatic detection or that it may learn biased or discriminatory features that may cause negative social impact. All these aspects are to be further investigated.

6. Conclusion

We introduce GMAIR, which combines spatial attention and a Gaussian mixture such that it can locate and cluster unseen objects simultaneously. We analyze the “what” latent variables and clustering process, showing that the model has the ability to detect and cluster similar objects automatically. For the downstream task, we show an example of image generation that may be applied to data augmentation or synthetic data generation. We also evaluate GMAIR quantitatively compared with SPAIR and SPACE, showing that GMAIR archives the state-of-the-art detection performance.

As the number of data increases and the cost of annotation rises, unsupervised object detection will play an increasingly important role in the future. Future research should be devoted to improving the detection performance for complex scenes. One possible option is to make use of advanced results in supervised learning. Another important topic is to balance multiple loss functions better.

Appendix

A. Derivation of the KL Terms

In this section, we derive the KL terms in Eqn. (6). By assumption of $q(z|x)$ and $p(z)$ (Eqn. (3) and Eqn. (1)), we have the following:

$$E_{q(z|x)} \left[ \log \frac{q(z|x)}{p(z)} \right] = \sum_{i=1}^{HW} E_{q(z|x_i)} \log \frac{q(z_i|x_i)}{p(z_i)}.$$  \hspace{1cm} (A.1)

The term $E_{q(z|x_i)} \log (q(z_i|x_i)/p(z_i))$ can further be expanded as follows:

$$E_{q(z|x_i)} \log \left( \frac{q(z_i|x_i)}{p(z_i)} \right) = q(z_i^{\text{pres}} = 0|x) \log \left( \frac{q(z_i^{\text{pres}} = 0|x)}{p(z_i^{\text{pres}} = 0)} \right) + q(z_i^{\text{pres}} = 1|x) \cdot \left( \log \left( \frac{q(z_i^{\text{pres}} = 1|x)}{p(z_i^{\text{pres}} = 1)} \right) + E_{q(x|x_i)} \left[ \log \left( \frac{q(z_i^{\text{where}}, z_i^{\text{depth}}, z_i^{\text{cat}}, z_i^{\text{what}}|x_i)}{p(z_i^{\text{where}}, z_i^{\text{depth}}, z_i^{\text{cat}}, z_i^{\text{what}}|x_i)} \right) \right] \right)$$

$$= KL(q(z_i^{\text{pres}}|x_i) || p(z_i^{\text{pres}})) + q(z_i^{\text{pres}} = 1|x) E_{q(x|x_i)} \left[ \log \left( \frac{q(z_i^{\text{where}}, z_i^{\text{depth}}, z_i^{\text{cat}}, z_i^{\text{what}}|x_i)}{p(z_i^{\text{where}}, z_i^{\text{depth}}, z_i^{\text{cat}}, z_i^{\text{what}}|x_i)} \right) \right].$$

Eqn. (A.2) is continued to expand:

$$E_{q(x|x_i)} \left[ \log \left( \frac{q(z_i^{\text{where}}, z_i^{\text{depth}}, z_i^{\text{cat}}, z_i^{\text{what}}|x_i)}{p(z_i^{\text{where}}, z_i^{\text{depth}}, z_i^{\text{cat}}, z_i^{\text{what}}|x_i)} \right) \right] = E_{q(z_i^{\text{where}}|x_i)} \log \left( \frac{q(z_i^{\text{where}}|x_i)}{p(z_i^{\text{where}})} \right) + E_{q(z_i^{\text{depth}}|x_i)} \log \left( \frac{q(z_i^{\text{depth}}|x_i)}{p(z_i^{\text{depth}})} \right)$$

$$+ E_{q(z_i^{\text{cat}}|x_i)} \log \left( \frac{q(z_i^{\text{cat}}|x_i)}{p(z_i^{\text{cat}})} \right) + E_{q(z_i^{\text{what}}|x_i)} \log \left( \frac{q(z_i^{\text{what}}|x_i)}{p(z_i^{\text{what}})} \right).$$

By the definition of the Kullback–Leibler divergence, the four terms in the RHS of Eqn. (A.3) are indeed:

$$KL(q(z_i^{\text{where}}|x_i) || p(z_i^{\text{where}})),$$

$$KL(q(z_i^{\text{depth}}|x_i) || p(z_i^{\text{depth}})),$$

$$E_{q(z_i^{\text{cat}}|x_i)} \left[ KL(q(z_i^{\text{cat}}|x_i, z_i^{\text{where}}) || p(z_i^{\text{cat}})) \right],$$

$$E_{q(z_i^{\text{what}}|x_i)} \left[ KL(q(z_i^{\text{what}}|x_i, z_i^{\text{where}}, z_i^{\text{cat}}) || p(z_i^{\text{what}})) \right] \right) \right],$$

(A.4)

respectively. Therefore, we complete the proof of Eqn. (5). During the implementation, we model discrete variables $z^{\text{pres}}$ and $z^{\text{cat}}$ using the Gumbel-Softmax approximation ([17]). Therefore, all variables are differentiable using the reparameterization trick.
B. Implementation Details

Our code is available at https://github.com/EmoFuncs/GMAIR-pytorch.

B.1 Models. Here, we describe the architecture of each module of GMAIR, as shown in Figure 1. The backbone is a ResNet18 ([18]) with two deconvolution layers replacing the fully connected layer, as shown in Table 3. Pres-head, depth-head, and where-head are convolutional networks that are only different from the number of output channels, as shown in Table 4. What-encoder and cat-encoder are multiple layer networks, as shown in Table 5. Finally, the glimpse decoder is a deconvolutional network, as shown in Table 6.

For other models, we make use of code from https://github.com/yonkshi/SPAIR_pytorch%20for SPAIR and https://github.com/zhixuan-lin/SPACE%20for SPACE. We utilize most of the default configuration for both models and only change A (the dimension of $z^{\text{what}}$) to 256 for comparison, the size of the base bounding box to 72 × 72 for large objects.

B.2 Training and Hyperparameters. The base set of hyperparameters for GMAIR is given in Table 7. The value $p$ (the prior on $z^{\text{pres}}$) drops gradually from 1 to the final value $6e-6$, and the value $\alpha_{\text{overlap}}$ drops from 2 to 0 in the early stage of training for stability. The learning rate is in the range of $[5e-5, 1e-4]$.

B.3 Testing. During testing phase, to obtain deterministic $\pi$ distributions. To be specific, we use $\mu^{\text{depth}}, \mu^{\text{what}}$, and $\mu^{\text{where}}$ for $z^{\text{att}}, z^{\text{depth}}, z^{\text{pres}}, z^{\text{what}},$ and $z^{\text{where}}$, respectively.

C. Calculation of AP, ACC, and NMI

The value of AP is calculated at threshold IoU = 0.5 using the calculation method from the VOC ([19]). Before calculating the ACC and NMI of clusters, we filter the incorrect bounding boxes. A predicted box $PB$ is correct if there is a ground truth box $GB$ such that IoU ($PB, GB$) > 0.5, and the class of a correct predicted box $PB$ is assigned to the class of the ground truth box $GB$ such that IoU ($PB, GB$) is maximized. After filtering, all correct predicted boxes are used for the calculation of ACC and NMI. Note that we still have many ways to assign each predicted category to a real category when calculating the value of ACC. In all of the ways, we select the one such that ACC is maximized, following [12]. Formulas are shown in Eqn. (C.1) and Eqn. (C.2) for the calculation of ACC and NMI:

$$\text{ACC} = \sum_{k=1}^{C} \max_{i,j} \left\{ \frac{|\{i | G_i = j, P_i = k\}|}{|P|} \right\}$$

$$\text{NMI} = \frac{2I(G, P)}{H(G) + H(P)}$$

where $G$ and $P$ are, respectively, the ground truth categories and predicted categories for all correct boxes, $C$ and $C'$ are the number of clusters and real classes, and $H(\cdot)$ and $I(\cdot, \cdot)$ are the entropy and mutual information function, respectively.

D. Additional Experimental Results

The graphs of “what” representation after 100k iterations are shown in Figure 6.

Data Availability

The data included in this study are available without any restriction.

Disclosure

This article was published on arXiv in preprint form [16].

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this study.

Acknowledgments

This work was partially supported by the Research and Development Projects of Applied Technology of Inner Mongolia Autonomous Region, China, under grant no. 201802005, the Key Program of the National Natural Science Foundation of China under grant no. 61932014, and Pudong New Area Science and Technology Development Foundation under grant nos. PKX2019-R02.

References