

Semi-Supervised Tile Embeddings: A General, Multigame Level Representation

Venkata Sai Revanth Atmakuri^{ID}, Kian Razavi Satvati, Anurag Sarkar^{ID}, and Matthew Guzdial^{ID}, *Member, IEEE*

Abstract—Representing video game levels for level generation and analysis tasks remains an open problem. Existing approaches generally rely on hand-authoring or are game-specific. Tile embeddings are a general machine learned-representation for tile-based game levels, however they have thus far relied solely upon hand-authored representations of levels for training data. In this article, we introduce semi-supervised tile embeddings (SSTE), which make use of semi-supervised learning to allow for training on levels lacking human authored representations. We evaluate SSTE over many experiments, finding that it performs equivalently or better than existing tile embeddings. Thus, SSTE stands as the first general machine-learned level representation that can scale without requiring additional human labor.

Index Terms—Machine learning, procedural content generation, representation learning.

I. INTRODUCTION

PROCEDURAL content generation via machine learning (PCGML) refers to the use of generative learning algorithms to create new content or media [1]. PCGML has gained significant attention for generating video game content, with game levels, a crucial game component, being a primary focus [2]. Thus, procedural level generation via machine learning (PLGML) covers many common PCGML tasks, and is a primary focus of this article. Unlike conventional ML tasks, such as image generation, PLGML is distinguished by the requirement for game elements to follow specific rules, such as playability. Thus, PLGML goes beyond the mere use of game visual data; it demands a comprehensive understanding of the semantics and functionality of game elements. Because of this, the problem of how to represent data for PLGML tasks remains open. In part due to the open nature of this problem we lack sufficient training data for many PLGML applications.

The most common solution to the PLGML representation problem is for researchers to utilize hand-authored representations [2], [3], [4]. While this approach is reliable for small generative models focusing on up to a few games, it requires substantial human labor to manually label each level. There have

been several attempts to fully or partially automate the process of labeling or learning representations for small sets of games levels [5], [6], [7]. However, these approaches have thus far not proven to be scalable.

Tile embeddings (TE) are a general, machine-learned representation for 2-D tile-based game levels [8], where *general* indicates that TE can be applied to represent any game level that is 2-D. Tile-based indicates that the levels should fit on a grid with repeated substructures, often called tiles. This is commonly found in early game consoles and modern indie games. TE represents tiles of such levels as continuous vector embeddings that take into consideration their visual, contextual, and behavioral information. Unlike other methods restricted to the specific game(s) they were trained on, TE provides a domain/game-agnostic solution to representing levels. Such a generalized representation has the potential to greatly expand the scope of PCGML applications and tasks, in contrast to most instances of prior work which focus on individual or small sets of games. This article builds on two prior papers introducing two TE variants: 1) the original tile embeddings (OTE) [8], which utilized an autoencoder trained on the VGLC dataset [2], and 2) the clustering-based tile embeddings (CTE) [9], which addressed the challenge of imbalanced tile distributions in the dataset faced by OTE. While both prior TE variants were able to represent unseen levels without a human representation they still required levels with a human representation for training, limiting their expressivity and quality.

In this article, we introduce the semi-supervised tile embedding (SSTE) approach and representation to address problems with the previous TE representations. Specifically, we focus on the ability for this approach to scale to a much larger set of game level data, with and without human representation. We refer to the data with a human representation as *annotated* and without a human representation as *unannotated*. SSTE differs from OTE and CTE via a change in architecture and loss function, allowing for a semi-supervised training process on both annotated and unannotated data. This general representation may allow SSTE to directly integrate into game development pipelines, enabling developers to obtain usable representations of their in-development assets for content analysis and/or generation.

II. RELATED WORK

One of the open problems in the field of PCGML is how to represent data for training deep neural networks [1]. A typical approach is to use hand-authored level representations, such as

Received 18 January 2025; revised 27 July 2025; accepted 19 September 2025. Date of publication 3 October 2025; date of current version 19 March 2026. This work was supported in part by the Canada CIFAR AI Chairs Program, Alberta Machine Intelligence Institute, and the Natural Sciences and Engineering Research Council of Canada (NSERC). Recommended by Associate Editor R. Bidarra. (*Corresponding author: Matthew Guzdial.*)

The authors are with the Department of Computing Science, Alberta Machine Intelligence Institute, University of Alberta, Edmonton, AB T6G 2R3, Canada (e-mail: atmakuri@ualberta.ca; kiansatvati@ualberta.ca; sarkar.an@ualberta.ca; guzdial@ualberta.ca).

Digital Object Identifier 10.1109/TG.2025.3617866

the Video Game Level Corpus (VGLC) [2], which consists of human-authored representations obtained by discretizing and labeling levels based on repeating building blocks called tiles, which has significantly influenced subsequent research projects in the field. We build upon the representation provided by the VGLC to construct a *generalized representation* across games, contrasting with previous works. Finding such a generalized representation has remained unexplored in this field, as many previous projects primarily focused on modeling and generation tasks for a single game or small set of games. Examples of such specialized representations include those used for reinforcement learning (RL) agents in level generation for *Super Mario Bros.* (SMB) [10] and Boxoban [3] for generating *Sokoban* levels [11]. Other examples include *Science Birds* [4], a clone of *Angry Birds* used initially for testing genetic algorithms, and a binary representation distinguishing only empty and nonempty tiles in SMB, used to train an auto-encoder for level generation [12]. Abraham and Stephenson [13] proposed a method to encode the continuous space of *Angry Birds* levels into a discrete grid suitable for generative adversarial network level generators, which is game-independent but relies on a discrete, known set of game objects.

In contrast to tasks utilizing a single game, level blending (LB) is the process of combining game levels from one or more games to produce novel game levels [14]. This approach has recently been identified as one approach to PCG via Knowledge Transfer [15]. Recent approaches for LB involve conceptual blending of multiple learned models [14] and training a variational autoencoder (VAE) to extract blends from the latent space [16], [17], [18]. Most of these approaches rely upon the previously mentioned VGLC [2]. However, as LB moves from a single-game focus to involving multiple games, the reliance on the VGLC requires researchers/developers to hand-author a secondary common representation for their selected game combinations. This involves significant human labor and requires in-depth knowledge of the selected games. These issues have been addressed by the TE representation [8], [9]. A blended representation of multiple levels learned using VAEs can be understood as similar to TE in terms of representing a cross-game representation. The difference is that LB learns this representation across levels or level sections instead of over the building blocks of levels. In addition, recent work has shown that TE representations, such as Clustered TE, can be used with the VAE approach to LB [19].

Despite the difficulty of learning representations for PCG problems, several instances of prior work exist, particularly for level design. Guzdial and Riedl [14] implemented a clustering algorithm that used only red, green, and blue (RGB) data from gameplay videos to develop a probabilistic model for level generation. Yang et al. [20] applied a Gaussian mixture VAE to learn clusters of specific types of levels with the clusters then being used to sample levels of that type. Similarly, Snodgrass [21] employed a Markov random field and clustering of the model's outputs to automatically extract tiles from level images without leveraging prior labels. In our research, in comparison, we use a modified version of the VGLC labels to not only cluster tiles but

also label them, clearly distinguishing the tiles for generative purposes.

Outside of clustering as an approach to representation learning, Karth et al. [6] used a vector quantized-variational autoencoder (VQ-VAE) model to learn a discrete latent representation for game maps from images. However, the constructed latent space is unique to each map image. Mawhorter et al. [7] took an unorthodox approach by training a VQ-VAE on read-only memory (ROM) data of levels in *Super Metroid*. Mirgati and Guzdial [22] trained a combined VAE and generative adversarial network (VAE-GAN) model on gameplay videos of SMB. and *Kid Icarus (KI)* to translate and generate levels concurrently. However, their VAE-GAN struggled to generalize to additional games outside of the original two.

Outside of level design, learning embeddings for modeling games, either for PCG or other technical games tasks is underexplored. Prior approaches have included training models on video frames for extracting recurring building blocks like sprites [23], embedding game entities' mechanical information [24], using a compact spatial and temporal representation of a game for faster RL agent training termed *World Models* [25], and a combination of the entity-based models and world models [26]. We believe there is potential for future research in using TE to enhance the overall performance of such models.

III. PREVIOUS TE

TE are a learned, unified, domain-independent, continuous vector representation of visual, contextual, semantic, and behavior information of game levels [8], [9]. More specifically, TE represent each tile in a level as an n -dimensional continuous vector obtained using an autoencoder trained on the pixel and affordance information of a given tile, where n is the size of the latent dimension of the autoencoder. Existing TE methods require annotated data for training, i.e., data where tiles are annotated with affordance information. Once trained, these methods can then obtain embeddings to represent both annotated and unannotated games. In this article, we present a novel TE approach, SSTE trained with both annotated and unannotated data. In this section, we discuss the two prior TE methods in detail: the OTE and CTE.

A. Original Tile Embedding

OTE [8] was the first TE representation. It utilized an autoencoder model to learn TE. They defined a tile in terms of its 16×16 pixel information and semantic affordance information. Although OTE requires annotated data for training, once trained, it can be used to obtain TE to represent both annotated and unannotated games, utilizing two inputs: 1) unified affordances information and 2) local pixel context information. The unified affordances comprise 13 commonly modeled features for PLGML: *block*, *breakable*, *climbable*, *collectable*, *element*, *empty*, *hazard*, *moving*, *openable*, *passable*, *pipe*, *solid*, and *wall* [8]. Each tile's affordances are represented using a multihot vector, where a value of 1 at the index of a feature indicates the presence of that feature for that tile, and 0 otherwise. The local

pixel context includes the 16×16 pixel tile at the center, along with the 48×48 pixel context surrounding it.

An autoencoder is an encoder-decoder neural network architecture that learns a compressed representation of the input [27]. The OTE autoencoder architecture is adapted from an X-shaped VAE architecture proposed by Simidjeviski et al. [28] and has two branches: one branch representing the local pixel context information containing 48×48 pixels with the center 16×16 pixels representing the tile, and the other representing the multihot vector representing the unified affordances of the tile. The encoder has two layers for each branch, which are combined at the third layer by a fully connected layer and passed on to a compressed fully connected layer, which captures the relationship between the two branches into a common latent space. An embedding vector in this latent space is referred to as a TE. The decoder mirrors the encoder architecture to reconstruct the two branches of output. The loss is defined as a weighted combination of binary cross-entropy for affordances and mean square error (mse) for pixels.

The OTE representation suffered from two main limitations: 1) it struggled to generate levels with skewed tile distributions, which is commonly seen in PLGML research [9] and 2) it grouped tiles primarily based on pixel information, which was a problem for unannotated tiles which lacked authored affordance information. As a result, the learned latent space of OTE depended primarily based on the color of a tile leading to situations, such as, for instance, empty blue sky tiles appearing near solid blue brick tiles, which in turn would lead to both being considered *Empty*. To address these limitations, Jadhav and Guzdial [9] introduced the CTE representation, which we overview in the next subsection.

B. Clustering-Based Tile Embedding

CTE [9] is a higher complexity TE representation, reliant on a clustering step in order to approximate a discrete tile representation. As with OTE, CTE is also based on an autoencoder and once trained using annotated games, can be used to obtain a TE representation for both annotated and unannotated games. Its improvement over OTE stems from not only the local RGB pixel context and unified affordances, but also the edges of each tile and additional information from a clustering step.

We obtain each tile’s edges by converting the RGB 16×16 pixel information ($16 \times 16 \times 3$) to grayscale and applying the Canny edge detection algorithm [29], obtaining a 16×16 edge feature vector. This was added to CTE to help differentiate cases with similar colors but unique edge information, like the blue background and blue tile example above.

The original CTE makes use of two clustering processes. The first occurs prior to embedding training and is obtained by clustering on the local pixel context, affordance, and edges before training the CTE model. Its purpose is to approximate a symbolic representation. In the original paper, this was done via DBSCAN [30] with $K = 11$ [9]. We forego this step in this article, given we are comparing between TE, not symbolic representations.

CTE’s second and primary clustering process makes use of a Gaussian mixture model (GMM), to reflect how multiple tiles can belong to multiple underlying groups (solids, hazards, etc.). An initial GMM process occurs prior to TE model training, clustering the same information as the input. These clusters are then used as a ground truth, such that we can ensure that a particular embedding falls within the same group as its associated cluster in the learned latent space. For this GMM implementation, we make use of the silhouette score and Bayesian information criterion to determine K .

The CTE model architecture consists of three branches, with the first two branches being the same as the branches of the OTE model and the third consisting of the edge information. As with OTE, the encoder has two layers for each branch, which are combined at the third layer by a fully connected layer and passed on to a compressed fully connected layer, which captures the relationship between the three branches in a common latent space whose embedding vectors are the TE. The decoder mirrors the encoder architecture to reconstruct the three branches of output. The CTE model’s loss function resembles the OTE loss function but incorporates the clustering information. We refer to it as the CTE loss. It consists of four parts: 1) clustering loss (L_c) as the categorical cross entropy error between the GMM cluster of a tile and its corresponding embedding during training, 2) edge feature vector loss (L_e), which is the mean squared error of the reconstructed edge feature vector and the original edge feature vector, 3) reconstructed image data (L_i), which is the mean squared error over the reconstructed RGB image, and 4) affordance reconstruction loss (L_a), which is a binary cross entropy loss. The final loss is the combination of these four losses with their respective weights: $(0.5 * L_i) + (1.5 * L_a) + (0.5 * L_e) + (0.5 * L_c)$. To accurately embed affordance information we assign it a higher relative weight for its reconstruction.

CTE performed well in comparison to the symbolic and hand-authored VGLC representation for SMB level generation [9]. But the main limitation of this (and the prior) TE approach remained, which is the inability to directly use unannotated games along with annotated games in training a TE representation. This is a limitation as it greatly restricts the available training data. To address this, we introduce the SSTE representation, as described in the following section.

IV. SEMI-SUPERVISED TILE EMBEDDINGS

SSTE is a new TE training approach and representation, which enables the use of both annotated and unannotated games. Due to this, we hypothesize that the SSTE representation will prove more general compared to other TE approaches and could then be useful for in-development games, where a developer can query the model with any in-development assets and retrieve a usable representation for content analysis or generation. In addition, SSTE makes use of a VAE architecture, instead of an autoencoder as in prior TE approaches, which enables the use of both annotated and unannotated tiles for training and makes it better suited for generative tasks, such as sampling new tiles or blending between tiles. SSTE has two branches compared to the three branches of CTE. The first branch represents the tile

TABLE I
AFFORDANCE SELECTION EXPERIMENT RESULTS

Metrics	13-Affordance Set	Eight-Affordance Set
Image Recon-mse	2216.73 ± 30.81	2294.64 ± 29.59
Affordance-EMD	0.06	~ 0.00
Affordance–Accuracy	0.80	0.78
Affordance–Precision	0.16	0.20
Affordance–Recall	0.19	0.22
Affordance–F1 Score	0.16	0.18

pixel information along with the contextual neighborhood pixel information of the tile. This branch uses the hue, saturation, and value (HSV) format instead of the RGB format of the previous TE approaches. Unlike RGB, HSV decouples color and brightness information of pixels and thus makes it unnecessary to have to separately track edge information as it implicitly captures it with the value (V) channel. The second branch represents the unified affordance information of the tile, as in OTE and CTE. However, the SSTE unified affordances are reduced to eight features compared to the 13 features of the OTE and CTE, which we discuss in the next section.

A. Affordance Selection

One of the initial steps for improving generalizability was the refinement of the affordance set used by OTE and CTE. As mentioned before, the original affordance set consisted of 13 features, represented as a multihot vector. However, certain features are extremely rare and only occur in a single game. In those cases, especially when it comes to training larger datasets, due to the high variance of data, the model will simply learn to ignore those features. Further, if that specific game is held-out as a test game, the model will not encounter that feature in association with other inputs.

Note that not all possible combinations of these features are valid in the context of most games. By looking at all the combinations, we detected certain redundancies in the 13 feature affordance representation; specifically, *empty* and *passable* always referred to the same semantic and mechanical properties in a tile across all games. These issues led to an imbalance in the embedding space and led to the data dependency discussed above. Thus, we revised the set to a more streamlined list of eight features: *breakable*, *climbable*, *collectable*, *hazard*, *moving*, *passable*, *portal*, and *solid*. To ensure the distinctiveness and effectiveness of the model, we ran experiments using both affordance sets and evaluated the model’s performance when trained only on annotated data. The model used for these experiments was CTE. We used Earth mover’s distance (EMD) between the model’s affordance predictions and the ground truth. We also evaluated the model using mse for image reconstruction and accuracy and precision, recall and F1-score for affordance vector predictions. The model used all the annotated game data as the training dataset except for KI, which we held-out as the test game to evaluate our model.

The results of these experiments are reported in Table I. We bold the best result for each metric. We see no major difference between any metrics other than EMD. EMD is a more objective

metric for these experiments because it does not require making the model’s continuous predictions discrete. EMD directly compares the model’s certainty over the occurrence of each individual affordance per tile. The lower EMD obtained through the smaller set of affordances led us to continue our experiments using the eight-features affordance set.

B. Model Architecture

A VAE is a generative model consisting of encoder and decoder networks that learn a continuous latent space distribution over training data [31]. The SSTE model architecture is a VAE-based X-shaped network adapted from OTE and X-shaped VAEs [8]. This contrasts with OTE and CTE, which used autoencoders. The SSTE encoder network consists of two branches: the first branch consists of the $48 \times 48 \times 3$ pixel information with the center $16 \times 16 \times 3$ representing the main tile in the HSV format, and the surrounding information providing the context for this tile. As with OTE and CTE, the second branch represents tile affordances using a multihot vector but for SSTE, this is of size eight as previously explained. The decoder network consists of the same two branches but in reverse and includes additional layers at the end to predict the clustering information as with CTE [9]. The architecture is shown in Fig. 1. The first encoder branch is the image branch with two 2-D convolutional layers, both with 32 filters, followed by a third convolutional layer with 16 filters. All three layers have common parameters of kernel size (3,3) with a L2 regularization weight of 0.01. Each of these layers is followed by a batch normalization layer and a ReLU activation layer. The final ReLU layer is followed by a flatten layer to combine the image branch information with the second encoder branch. This second branch consists of the affordance information. It has two fully connected layers, the first with 32 units and the second with 16 units. Both layers use tanh activation. The architecture for both branches is based on the previous TE models. After both branches, we have a 256-unit fully connected layer to concatenate and synthesize the information from both branches. This layer then feeds directly into the latent space. We use a 256-D latent space for the VAE. Embedding vectors sampled from this latent space represent the TE in the SSTE representation.

The decoder is the inverse of the encoder. We start with a 256-unit fully connected linear layer, which is then split into two branches. The image branch has three transpose convolutional layers in the inverse ordering of the encoder. Similarly, the affordance branch has two fully connected layers in the inverse order. At the end of these two branches, we predict the image output $16 \times 16 \times 3$ and affordance output of length 8. Finally, both these outputs are combined into a final fully connected softmax layer of length c , where c is the number of clusters identified in the training data. This matches the same approach taken in the CTE model.

C. Loss

The SSTE model loss consists of four components: the pixel/image information loss L_i , affordance information loss

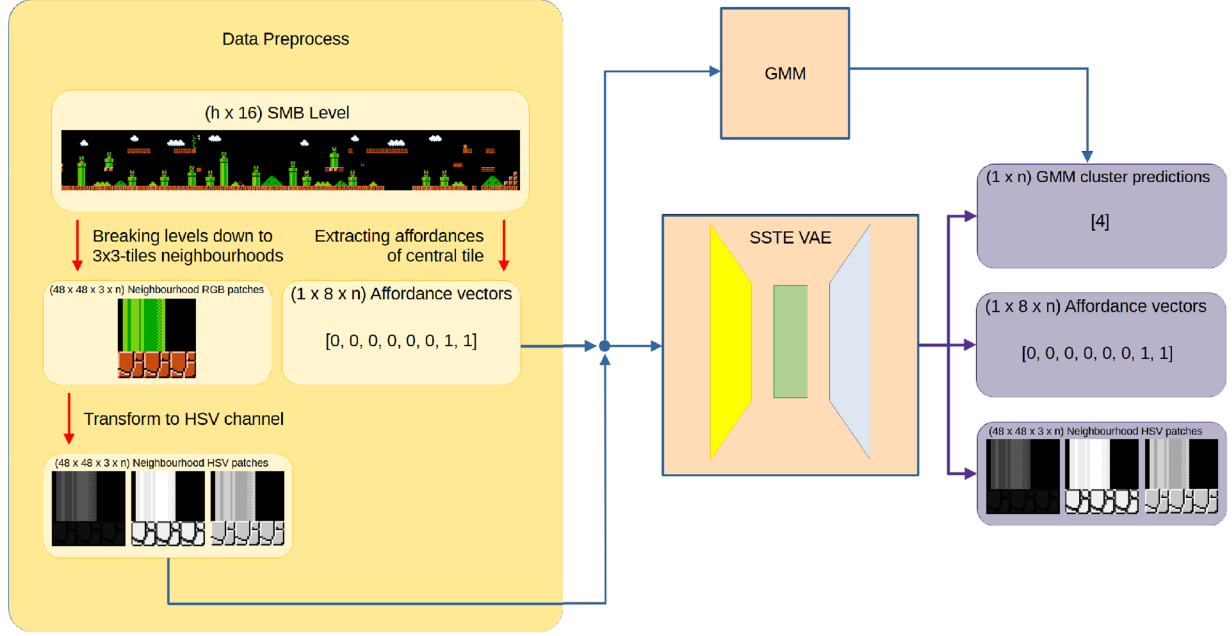


Fig. 1. SSTE model architecture. The inputs to the VAE consist of 3×3 level chunks (in HSV format), the affordance vector of the center tile and the cluster predicted by the GMM previously trained on the same tile and affordance data. TE are learned by training the VAE to predict the pixels of the center tile, the affordance vector and the cluster assignment.

L_a , clustering information loss L_c , and the Kullback–Leibler-divergence (KL-divergence) loss L_{kl} . L_i uses mse on the reconstruction of the pixel tile information, as with OTE and CTE. L_a uses binary cross entropy on the prediction of the affordance information. For L_a , we only calculate the binary cross entropy for annotated tiles. This approach is distinct from OTE and CTE. We refer to this loss as a semi-supervised loss. The clustering information loss (L_c) uses categorical cross entropy as with CTE on the prediction of the clustering information to the original assigned clustering value by the GMM based on the pixel/image and affordance information. Finally, L_{kl} is a standard loss component in VAE models and indicates the difference between the learned latent distribution and the Gaussian prior [31]. Overall, the SSTE model loss can be represented for annotated tiles as

$$\text{TotalLoss} = L_i + L_a + L_c + L_{kl} \quad (1)$$

and for unannotated tiles

$$\text{TotalLoss} = L_i + (0 * L_a) + L_c + L_{kl}. \quad (2)$$

Note that we do not define two losses but flag unannotated tiles in the loss function to not use the affordance loss L_a .

D. Training

The SSTE model was trained using the Adam optimizer [32] with a learning rate of $1e - 05$, batch size of 256, and for a total of 1000 epochs. The training data consisted of the pixel context information in HSV format, the affordance information associated with the features of this tile, and the cluster information. To obtain the cluster information, as in CTE [9], we trained a standard GMM using the center tile $16 \times 16 \times 3$ pixel HSV information along with the affordance information concatenated

with the flattened pixel information. Although we could have used a higher learning rate and fewer epochs due to the use of batch normalization, we chose to use this lower rate and more epochs to achieve better generalization and avoid local minima. This is a common training approach in PCGML for high variance data [19], [33].

V. EVALUATION

We performed five sets of experiments to investigate the value of SSTE, particularly in comparison to OTE and CTE. At this point, we wish to note that OTE and CTE can be viewed as not only baselines, but ablations of SSTE. SSTE was derived from CTE by using a VAE over a non-VAE, altering the loss function, and replacing RGB inputs with HSV inputs. CTE in turn was derived by augmenting OTE with clustering and edge information. Our five sets of experiments considered: 1) predicting affordances for unseen tiles, 2) approximating tile affordances for unannotated games seen during training, 3) generating blended levels, 4) playability, and 5) varying the amount of available training data. These are described in the following sections.

A. Datasets

For semi-supervised learning, we needed to integrate unlabeled game levels into the labeled VGLC dataset used in prior TE work. We leveraged VGMaps¹ and scraped the game data from 1145 different games across 49 different consoles. However, some of these games cannot be considered as tile-based. We pruned the games by breaking the levels into 16×16

¹<https://www.vgmaps.com/>

tiles, and comparing the occurrence frequency of unique tiles within a game against a hard-coded threshold of 100, indicating that fewer than 100 unique 16×16 tiles had to exist to consider a game tile-based. If a level exceeded this threshold, the corresponding game was considered “high resolution” and dropped from the final dataset. Finally, we broke down each level into 3×3 tile neighborhoods. This led to a set of approximately 2 million neighborhoods which were combined with the VGLC data to obtain the dataset for our experiments. However, due to the size of this dataset, we use different subsets of it across different experiments.

For the first, third, and fourth experiments, we used only the original annotated VGLC game levels used in prior TE work. These represent a subset of levels for the games *KI*, *Lode Runner (LR)*, *The Legend of Zelda (LoZ)*, *Mega Man (MM)*, and *SMB*. We refer to this dataset as “Only Annotated.”

For the second experiment, given our interest in studying the effects of unannotated training data on the TE models, we used several dataset subsets. We randomly selected four unannotated games from our complete dataset. We then sampled from this subset so that we had an equal amount of annotated and unannotated data for each fold of a cross-fold analysis. These dataset subsets ranged from roughly 40 to 60 K samples.

For the fifth experiment, we created two different datasets. We held-out *KI* as the test game, leaving us with 23 K annotated datapoints. We then sampled 27 and 77 K unannotated datapoints, creating the “50 K Datapoints” and “100 K Datapoints” training sets, respectively. We used a ratio of 0.1 for creating test sets in conjunction to the previous datasets.

VI. ZERO-SHOT UNSEEN TILE AFFORDANCE CLASSIFICATION

In this experiment, we aim to determine how well the different TE approaches can predict the affordance values for unseen tiles. This can be useful in a hypothetical scenario where we do not know the ground truth affordances of a tile and want to employ a trained TE approach to give reasonable affordances. Given the difficulty in evaluating how reasonable a set of approximated affordances are, we instead use the case with an annotated but unseen game to evaluate this scenario. In this experiment, we do not expect the SSTE model to outperform the OTE or CTE models as they share the same number of annotated game samples. However, if we can achieve a similar performance, this would show that our new affordance selection (as described in Section IV-A) does not harm SSTE’s ability to model affordances.

Since the goal of this experiment is to evaluate the performance of the different TE models on predicting unseen game tiles for unannotated games, we could not use the unannotated games directly due to the lack of ground truth for comparison of the predicted affordances. Therefore, we used an annotated game but treat it as an unannotated game. Specifically, we employed a cross-fold analysis, given the limited number of annotated games. We have five annotated games taken from the VGLC [2], namely, *KI*, *LR*, *LOZ*, *MM*, and *SMB*. Instead of holding out some levels from each game as done in traditional cross-fold analysis, we held out each single game in turn, treating it as the

TABLE II
TILE AFFORDANCE CLASSIFICATION FOR UNSEEN GAMES RESULTS

	SSTE	CTE	OTE
Accuracy	90.91	87.94	88.27
Precision	23.74	22.87	19.04
Recall	26.15	5.94	8.86
F1-Score	22.07	15.92	19.89

unannotated game. Therefore, in total, we trained five models for each TE approach OTE, CTE, and SSTE.

Given this is a classification task, we report the accuracy of each model’s prediction. We also used precision, recall, and F1-score as additional metrics, which are commonly used metrics for multiclass prediction tasks, especially in the presence of imbalanced classes where accuracy alone could be misleading. Since we have multiple features (classes), we use binary classification, therefore we have multiple approaches to present this information: macro averaging, micro averaging, and weighted averaging. Although we could have used any of these approaches, we chose macro averaging, for consistency with OTE [8].

A. Results

Table II shows the results for SSTE, CTE, and OTE. For all the metrics except Recall, we see that the results are similar for each TE approach, but overall, SSTE performs equivalently or better than CTE and OTE for all metrics. The recall evaluation metrics provides interesting insights for different TE approaches. Based on the recall results, we can see that the CTE and OTE have a large percentage of tile affordances being false negatives (FN), which means a large number of tile affordances are misclassified as false when they are not false in reality. We hypothesize that this is due to the inability of CTE and OTE to differentiate between annotated and unannotated game tile affordances during training. Because unannotated affordances are represented as all-zeros by default, CTE and OTE are misled to believe that the actual affordances for unannotated game tiles are all zero. However, SSTE is not affected by this because it can differentiate between annotated and unannotated game tiles.

Therefore, these results indicate that SSTE can be directly used with both annotated and unannotated games during test time, and the zero-shot affordance classification for unseen tiles is not affected. SSTE can thus provide a more reasonable tile affordance approximation for unseen tiles.

VII. UNANNOTATED TILE AFFORDANCE APPROXIMATION

In this experiment, we investigate the ability for our TE models to approximate tile affordances for unannotated games *within* their training data, compared to during test time in the prior experiment. In particular, we wish to identify the impact of SSTE being able to train on both annotated and unannotated games. We expect that SSTE will yield better results compared to CTE and OTE as these require having to manually annotate the affordances of unannotated games using their respective unified affordance representation.

TABLE III
UNANNOTATED TRAINING TILE AFFORDANCE APPROXIMATION RESULTS

	Zeros			Ones		
	SSTE	CTE	OTE	SSTE	CTE	OTE
Accuracy	88.33	87.51	87.49	87.28	29.73	12.51
Precision	24.91	24.40	43.57	24.38	11.64	12.5
Recall	35.52	0.06	0.004	36.60	69.18	100.0
F1-Score	26.16	0.58	0.046	26.06	14.79	17.64

Since we want to evaluate the tile affordance approximation for unannotated training games, which by default do not have ground truths to evaluate, we again used existing annotated games and presented them to the model as if they were unannotated. We use the same five annotated games as in the prior experiment. The unannotated games do not have ground truth affordances by definition. As both OTE and CTE require affordances, we chose two different ways to represent the lack of affordances. In *all-zeros*, we use a vector of all zeros for the affordance information, following from prior TE work [8]. In addition, we have an *all-ones* representation where we use a vector of all ones. We wanted to determine the impact of these different representations on model performance. We anticipate SSTe to do well in either case, since its loss function ignores affordance information for unannotated games. As in the prior experiment, we look at accuracy, precision, recall, and F1-score and use macro averaging to deal with imbalanced classes. Unlike the previous experiment, due to the imbalanced features, we do expect lower metric values.

A. Results

Table III shows the results of affordance approximation for unannotated game tiles from their training data, using each of the two aforementioned representations. These results indicate that SSTe performs equivalently irrespective of how the unannotated game tiles are represented across all the evaluation metrics. When unannotated game tiles are represented as all-zeros, SSTe performs better across all metrics except for precision. When unannotated game tiles are represented as all-ones, SSTe performs better across all metrics except for recall. If we examine individual metric results, we see that for accuracy, SSTe performs equivalently for both all-zeros and all-ones. However, for CTE and OTE, we see the opposite: higher accuracy values for all-zeros and lower accuracy values for all-ones. This shows that for CTE and OTE, the way we represent the unannotated game tiles has a major impact on accuracy. When we use all-zeros, due to the one-hot affordance representation, we have higher false (zeros or negative) values, which the CTE and OTE models interpret as their true affordance values. Therefore, we see higher true negative and lower FN values, and the opposite when we have all-ones, we see higher true positive values and lower false positive values. Therefore, we see a high accuracy when all-zeros and inversely lower accuracy when all-ones. This same trend continues across the metrics of precision, recall, and F1.

Overall, these results indicate that SSTe performs equivalently irrespective of the unannotated game tile representation

and provides reasonable unannotated tile affordance approximation for training data compared to the CTE and OTE given a balanced dataset of annotated and unannotated data.

VIII. LEVEL BLENDING

The aim of this experiment is to investigate the application of SSTe and OTE to LB, given that CTE has already been successfully applied to blending [19]. This would provide insights into how well each TE approach can generate novel game levels. We expect all approaches to perform similarly as we typically use only annotated games for blending. However, even if SSTe performs similarly to CTE and OTE, SSTe can directly be trained with annotated and unannotated games as established in the prior experiment and therefore can be used with a larger pool of games. LB is the process of combining different game levels to produce a novel game level. A common approach for LB in recent years has been to train a VAE model on a dataset of different game levels and use this model to generate novel game levels [16], [17], [18]. For this experiment, we use the same setup as prior work applying CTE to LB [19]. We trained a VAE model using the various TE representations of game levels in $16 \times 16 \times 256$ segments. 16×16 here represents subsections of game levels of 16 tiles in the width and height, while 256 represents the size of our embeddings. Since, we wanted to compare the different TE approaches, we used the “Only Annotated” dataset and trained a VAE on level sections in the SSTe, CTE, and OTE representations. In total, we trained three VAE LB models, one for each TE approach.

To blend level segments, as is typical in VAE-based blending work, we used interpolation which involves choosing a game level segment from two different games, obtaining their latent encodings using the VAE encoder, interpolating between these encodings and forwarding the interpolated encodings through the VAE decoder. Each interpolated point should theoretically then represent a novel level section combining elements of the two original level segments. The resulting blended outputs consist of the blended segment image and the affordance. Since we have a total of five games, we obtain ten blend combination pairs, i.e., LR-SMB, KI-LOZ, etc. For each pair, we generated 300 level segments for evaluating affordance outputs and 30 samples for evaluating image outputs for a set of blend percentage combinations. The different blend percentage combinations were: 0, 25, 50, 75, and 100 percent of game1 and the corresponding remaining percentage for game2, e.g., (25% Game1–75% Game2).

For evaluating the affordance output, we measured the EMD between the affordance distributions of the generated blended segments and the original segments from each game in the blend pair. Ideally, blended segments should follow affordance distributions in between those of the original games being blended. Since it would be hard to present EMD values for each game combination as there are a total of 10 values per combination for each TE approach, we averaged the EMD value across all the combinations for each blend percentage.

For evaluating the image output, we used two metrics—the structure similarity index metric (SSIM) and the Frechet

TABLE IV
EMD VALUES COMPARED BETWEEN DIFFERENT INTERPOLATION (GAME1 %-GAME2 %) GENERATED NOVEL LEVELS VERSUS ACTUAL GAME LEVELS

Blend (G1-G2)	SSTE		CTE		OTE	
	Game1	Game2	Game1	Game2	Game1	Game2
0-100	3742 ± 1492	0 ± 0	4389 ± 1890	0 ± 0	4299 ± 1779	0 ± 0
25-75	3842 ± 1592	917 ± 1048	4261 ± 1856	391 ± 190	4114 ± 1774	366 ± 196
50-50	2811 ± 1662	2767 ± 2394	3219 ± 1231	2461 ± 1471	2896 ± 1113	2848 ± 1648
75-25	1446 ± 979	3573 ± 1682	626 ± 234	4020 ± 1791	484 ± 172	4087 ± 1761
100-0	0 ± 0	3742 ± 1492	0 ± 0	4389 ± 1890	0 ± 0	4299 ± 1779

TABLE V
FID VALUES COMPARED BETWEEN DIFFERENT INTERPOLATION (GAME1 %-GAME2 %) GENERATED IMAGES VERSUS RANDOM IMAGES

Blend (G1-G2)	SSTE	CTE	OTE
0-100	548 ± 138	452 ± 40	494 ± 54
25-75	562 ± 138	457 ± 42	513 ± 50
50-50	590 ± 82	483 ± 44	598 ± 84
75-25	583 ± 134	476 ± 51	601 ± 132
100-0	568 ± 137	461 ± 53	587 ± 139

inception distance (FID). SSIM [34] compares the structural similarity between two images based on statistical measures. Similar to our affordance output evaluation, we calculated the SSIM between the visual data of the original level segments for each of Game1 and Game2 and the generated blend segment image data. In contrast to the affordance output evaluation, we used the correlation for the SSIM values since, as the percentage of each game in the blend increases, we would expect the SSIM to be closer to that game, and we cannot just use average and standard deviation of these SSIM values as they differ between games.

FID [35] is a commonly used metric to evaluate the quality of images generated by generative models like VAEs. Since we do not have actual ground truth for the blended level segments, we compared our blended level segments to uniformly random images. We used FID to evaluate quantitatively whether our generated blend segments are better than random in appearance. We choose a similar approach to affordance evaluation where we averaged the FID results for all ten game combinations for each blend percentage.

A. Results

For this experiment, for the generated interpolation blends, we have two outputs for evaluation: images and affordances. Table IV contains the results for the blend affordance outputs, while Table V contains the results for the blend image outputs. Table IV shows the EMD values between the different generated blend combinations of Game1 and Game2 compared to the original games, averaged across all 10 game combinations. Each row contains results for a particular blend combination for each (Game1 and Game2) pair.

The 0-100 blend combination comparison to Game1 and the 100-0 blend combination comparison to Game2 for each TE approach contain the same values because in both cases, the value is being calculated between the same pair of original games. Notably, the EMD values are the lowest for SSTE compared to CTE and OTE, which suggests that the two games

are represented in a more compact space, which may be useful for achieving a better blend interpolation. For the 50-50 blend combination, we would ideally like to see two similar values which would indicate that this blend is halfway between the two input games in terms of the affordance information. The EMD values between Game1 and Game2 for the 50-50 blends show this result for SSTE and OTE, whereas for CTE we see a large disparity between Game1 and Game2. This suggests that SSTE and OTE are better at generating blends representing both games compared to CTE, which runs counter to our expectations as only CTE has been applied to LB previously [19]. Observing the overall trend for Game1 for all TE approaches, as the Game1 % increases from 0% to 100%, we see that the EMD value ranges decrease, and similarly for Game2 but in reverse order. This suggests that all the TE approaches can generate controllable interpolation blends, which is useful for LB applications [17].

We ran Pearson's correlation test over these interpolation values and SSIM values across the blends for both pairs of game levels. A high value here would indicate that the blends were highly controllable, with the SSIM value changing as the interpolation value changed. We found that all correlation tests were significant with values of 0.9 and higher in all cases for SSIM. This further supports that all TE approaches can support highly controllable blends.

Table V shows the FID between the various generated blended images for all the game combination pairs, compared to random images. For these results, we expect the larger the FID value, the further the generated images are from random images. For the 0-100 and 25-75 blend combinations, SSTE has larger FID values compared to CTE and OTE. For all the remaining blend combinations, SSTE and OTE have similar values compared to CTE. Overall, we can see that all the blend combinations for all the TE approaches have FID values greater than 400, which indicates that all these generated blend images are distinct from random images. In particular, SSTE has consistent and higher values compared to CTE and OTE for all blend combinations, suggesting that SSTE outputs appear furthest from random images. These results lead us to conclude that all TE approaches generate blends that are not random images, and SSTE is the furthest, producing better quality images compared to the CTE and OTE for all blend combinations.

We provide randomly sampled blends from our VAE latent space for the SSTE in Fig. 2. We emphasize that these are randomly sampled from the latent space, which explains their seemingly random appearance. However, even in this case, clear structure is evident across the majority of images.

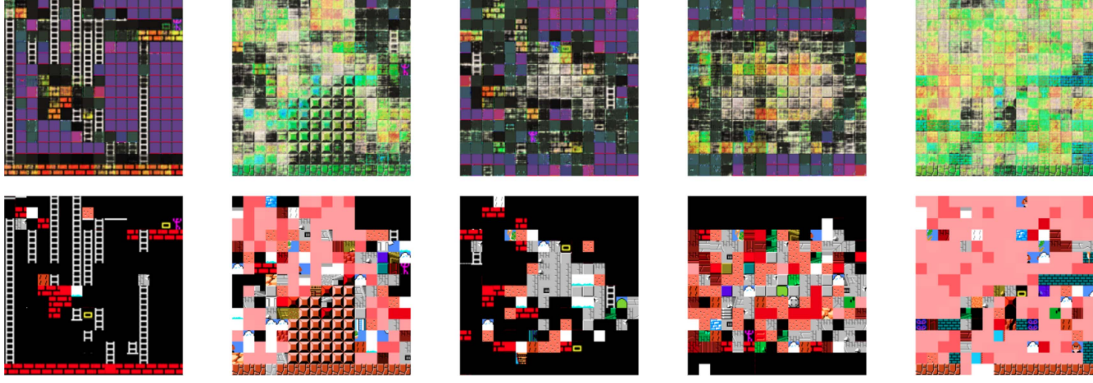


Fig. 2. Randomly blended level segments with VAE decoder (Top images) and nearest neighbor (Bottom images) visualizations for the SSTE representation.

IX. PLAYABILITY

A natural concern for TE generally might be the representation’s potential impact on level playability. In this section, we present an experiment to investigate this concern. Prior research has demonstrated the potential for output levels with a high rate of playability for OTE and CTE [9], [19] and thus we wanted to investigate this for SSTE.

In this experiment, we focused on how rerepresenting an existing level with TE might impact that level’s playability. We chose this over generating new levels as this would introduce an additional layer of obfuscation in the form of the level generator model. Instead, we can see how playability changes for a known playable level-based solely on re-representing it as a TE.

To measure the playability, we used an A* agent based on prior work [19], which in turn was based on the prior A* agent included in the VGLC [2]. This agent uses game-specific reachability information for four of the original VGLC games: KI, LoZ, LR, MM, and SMB. Notably, our A* agent makes use of the affordance information rather than the tile-information of the VGLC. Thus, rather than having to define game-specific passable tiles, our A* agent assumes it can move through any tiles with the passable or empty affordances. Given that we are focused on affordance information and playability, we make use of the “Only Annotated” dataset for this experiment.

We had to define game-specific playability criteria for the levels due to the differences in the different games’ objectives. These were as follows.

- 1) *KI*: Checks for the existence of a path from the bottom to the top of the level.
- 2) *LoZ*: Checks for each room whether we can traverse from one corner to the opposite corner. We randomly select a starting corner, then search for the closest passable tile to that corner as the starting position. We would then go through a similar process for the opposite corner and set the closest passable tile as the goal. We did this due to the lack of consistency in door placement and number.
- 3) *LR*: Checks for a path from the player’s starting position to each tile with a collectable affordance in each level. This is due to the requirement to collect all gold in the original game.

TABLE VI
PLAYABILITY RATE ACROSS THE LEVELS FOR EACH GAME

	KI	LOZ	LR	MM	SMB
OTE	0.4	0.13	0.38	0.8	1.0
CTE	0.4	0.11	0.37	0.9	1.0
SSTE	0.4	0.38	0.37	0.8	0.93

- 4) *MM*: Checks for a path from the player’s starting position to the end position of each level.
- 5) *SMB*: Checks for a path from the leftmost tile to rightmost column of each level.

X. RESULTS

In Table VI, we give the playability rate in terms of the number of playable levels over total levels (or rooms for LOZ) for each game in the VGLC. That is, while all three embeddings only achieved 0.4 for *KI*, this represents two out of the five available levels. Given that our A* agent is based on the VGLC A* agent, it performs the best on platformer levels similar to *Mario*. When it performs worse, this does not necessarily indicate the level was unplayable. This may instead be due to issues with the A* agent, for example the lack of handling of dynamic elements outside of the player (which hampers *KI* performance). As such, we argue that these playability rates should be evaluated relative to one another, not universally.

Overall, we can see that SSTE performs equivalently or better than OTE and CTE, which is promising given the ability of OTE and CTE to produce novel playable levels, as previously reported [9], [19]. In the case of MM and SMB, OTE, and/or CTE appear to outperform SSTE. We note in both cases this is due to a single additional level achieving the playability criteria given above. In both cases, this is actually due to the higher precision of SSTE.

We give a relevant subset of a SMB level as an example of SSTE’s greater precision leading to worse playability in Fig. 3. OTE and CTE introduce an additional block not present in the original level, making it “playable” based on this limited A* agent. Note that in OTE, this is only barely visible in the image, appearing as a thin slice of a LR tile, which nevertheless has the *Solid* affordance. This is not correct. Given this, we conclude



Fig. 3. Reconstruction for *Mario 6-2* using nearest neighbors for OTE (left), CTE (middle), and SSTE (right). Note the additional LR bricks in CTE, and to a lesser extent in OTE.

TABLE VII
IMAGE RECONSTRUCTION EVALUATION FOR OUR EXPERIMENT VARYING THE
UNLABELED TRAINING DATA

Metrics	OTE	CTE	SSTE
Only Annotated			
MSE (RGB)	0.06 ± 0.11	0.06 ± 0.11	0.09 ± 0.09
SSIM (RGB)	0.36 ± 0.26	0.32 ± 0.24	0.24 ± 0.27
MSE (HSV)	0.13 ± 0.18	0.11 ± 0.16	0.06 ± 0.1
SSIM (HSV)	0.51 ± 0.35	0.43 ± 0.34	0.47 ± 0.34
50K Datapoints			
MSE (RGB)	0.06 ± 0.11	0.06 ± 0.11	0.06 ± 0.06
SSIM (RGB)	0.41 ± 0.3	0.32 ± 0.24	0.29 ± 0.26
MSE (HSV)	0.14 ± 0.17	0.1 ± 0.16	0.03 ± 0.05
SSIM (HSV)	0.45 ± 0.34	0.56 ± 0.35	0.56 ± 0.33
100K Datapoints			
MSE (RGB)	0.06 ± 0.11	0.06 ± 0.11	0.07 ± 0.07
SSIM (RGB)	0.43 ± 0.3	0.34 ± 0.25	0.25 ± 0.25
MSE (HSV)	0.13 ± 0.15	0.10 ± 0.16	0.03 ± 0.05
SSIM (HSV)	0.4 ± 0.32	0.53 ± 0.36	0.54 ± 0.32

that SSTE does not negatively impact playability compared to other TE representations.

XI. VARYING UNLABELED TRAINING DATA

In this experiment, we aimed to evaluate the models' performance in automated tagging and image reconstruction tasks across previously unseen games with much larger training datasets. Achieving success here would significantly reduce human effort in labeling tiles within a level and could facilitate more advanced blending possibilities, as the model encounters a broader range of training images.

For this purpose, we conducted multiple experiments using varying training set sizes across all TE representations, and subsequently evaluated and compared their performance in both affordance prediction and image reconstruction. For affordance prediction, we compared the models' accuracy, precision, recall, and F1-score based on the predicted affordance vectors. For image reconstruction, we employed mse and the SSIM, between the original images and their reconstructions in both RGB and HSV color channels. The SSTE model was trained and optimized over HSV images, whereas the two earlier models utilized RGB.

Initially, the models were trained using only annotated game data, which encompassed approximately 30 K data points of 3×3 neighborhoods. We then added a similar amount of unannotated data points, increasing the training set size to 50 K. Finally, we doubled the dataset size again, training the models on 100 K data points. Each model and dataset combination was tested three times, using three different random seeds. The results in Tables VII and VIII represent the average performance across these runs. In all experiments, KI was held out for evaluation; the reported results thus reflect the models' performance on that game. To simulate an unannotated test scenario, we passed all-zero vectors to the models during testing, although ground

truths for KI affordances were available. To accommodate the complexity of the larger datasets (i.e., 50 and 100 K), the latent space of the SSTE model was increased to 1024 compared to 256 in OTE and CTE. We also employed a batch size of 1 and a cosine decay learning rate scheduler to stabilize training over these higher variance datasets.

A. Results

We observed relatively consistent performance in image reconstruction across all models. However, as shown in Table VII, when evaluated using HSV metrics, SSTE outperforms the other models. This outcome is expected since SSTE uses HSV inputs, and its loss function also optimizes over HSV.

Regarding affordance prediction, SSTE demonstrates clear superiority over OTE and CTE with larger datasets. Based on the results in Table VIII, the two most dominant affordances in the data are passable and solid, respectively. The earlier TE models lack adequate mechanisms to prevent convergence toward predicting all-zero vectors for affordances. This convergence explains why their accuracy scores are sometimes higher than that of SSTE for nonexistent affordances. Overall, we confirm that SSTE maintains robust performance when encountering unseen games and thus generalizes better to datasets with higher variance, yielding a richer embedding space. We hypothesize that this enhanced generalization could improve content generation quality, particularly in LB, as well as enhance analysis and automated labeling.

XII. DISCUSSION

The results in Section VI show that all the TE approaches perform similarly for affordance classification, with SSTE performing slightly better than OTE and CTE. This is useful in cases where we have a single TE approach that is already trained on a large number of games and can be used to acquire affordance predictions for new games without having to retrain the model. In Section VII, the results indicate that SSTE performs similarly in terms of image quality but greatly outperforms OTE and CTE when trained on a mix of annotated and unannotated data. The results in Section VIII suggest that the embedding space is more compact for SSTE compared to CTE and OTE. Further, we see that SSTE performs better for producing nonrandom blended images. Moreover, this experiment demonstrates that SSTE and OTE can be used for LB. In addition, the results in Section IX suggest that SSTE performs similarly to prior TE representations in terms of playability of levels. This is encouraging since the prior representations were already known to support the generation of novel, playable levels. Finally, the results in Section XI match the results in Section VII, further supporting that SSTE does not suffer in image quality compare to OTE and CTE but does improve in affordance representation quality. Taken together, we argue that this demonstrates that SSTE can successfully allow for the training on unannotated data with performance benefits in affordance representation. This opens the door for SSTE to serve as the first truly general TE representation, applicable to a wide variety of PCGML tasks.

TABLE VIII
AFFORDANCE PREDICTION EVALUATION FOR THE EXPERIMENT VARYING UNLABELED TRAINING DATA

Dataset	Affordance	Counts	OTE				CTE				SSTE			
			acc	prec	rec	f1	acc	prec	rec	f1	acc	prec	rec	f1
Only Annotated	breakable	0	0.982	0.00	—	0.00	0.933	0.00	—	0.00	0.971	0.00	—	0.00
	climbable	0	0.961	0.00	—	0.00	0.934	0.00	—	0.00	0.971	0.00	—	0.00
	collectable	0	0.996	0.00	—	0.00	0.98	0.00	—	0.00	0.999	0.00	—	0.00
	hazard	92	0.992	0.00	0.00	0.00	0.966	0.001	0.004	0.002	0.994	—	0.00	0.00
	moving	438	0.973	0.079	0.002	0.003	0.961	0.112	0.057	0.072	0.973	—	0.00	0.00
	passable	13136	0.510	0.779	0.547	0.624	0.568	0.834	0.566	0.653	0.821	0.859	0.933	0.893
	portal	58	0.921	0.004	0.086	0.008	0.936	0.005	0.081	0.009	0.996	—	0.00	0.00
	solid	4440	0.431	0.211	0.526	0.291	0.574	0.384	0.660	0.474	0.736	0.512	0.791	0.621
50K Dataset	breakable	0	1.00	0.00	—	0.00	1.00	—	—	—	0.925	0.00	—	0.00
	climbable	0	0.999	0.00	—	0.00	0.999	0.00	—	0.00	0.753	0.00	—	0.00
	collectable	0	0.999	0.00	—	0.00	1.00	—	—	—	0.974	0.00	—	0.00
	hazard	92	0.993	0.00	0.00	0.00	0.994	—	0.00	0.00	0.989	0.015	0.011	0.012
	moving	438	0.973	0.00	0.00	0.00	0.973	—	0.00	0.00	0.970	0.014	0.003	0.005
	passable	13136	0.229	0.978	0.041	0.078	0.196	0.00	0.00	0.00	0.821	0.822	0.992	0.899
	portal	58	0.988	0.00	0.00	0.00	0.996	—	0.00	0.00	0.984	0.005	0.023	0.009
	solid	4440	0.723	0.473	0.027	0.050	0.728	—	0.00	0.00	0.642	0.455	0.857	0.579
100K Dataset	breakable	0	1.00	0.00	—	0.00	1.00	—	—	—	0.949	0.00	—	0.00
	climbable	0	0.999	0.00	—	0.00	1.00	—	—	—	0.491	0.00	—	0.00
	collectable	0	1.00	0.00	—	0.00	1.00	—	—	—	0.998	0.00	—	0.00
	hazard	92	0.994	0.00	0.00	0.00	0.994	—	0.00	0.00	0.994	0.00	0.00	0.00
	moving	438	0.973	—	0.00	0.00	0.973	—	0.00	0.00	0.973	0.00	0.00	0.00
	passable	13136	0.214	0.861	0.026	0.048	0.196	0.483	0.006	0.012	0.812	0.816	0.989	0.894
	portal	58	0.990	0.00	0.00	0.00	0.996	—	0.00	0.00	0.996	0.00	0.00	0.00
	solid	4440	0.718	0.192	0.014	0.025	0.728	—	0.00	0.00	0.488	0.337	0.787	0.450

XIII. LIMITATIONS AND FUTURE WORK

There are a number of limitations in this work that present avenues for the future. Fig. 2 contains two sets of visualizations of randomly blended images using the SSTE-LB model in experiment Section VIII, which are obtained from vectors sampled randomly from the SSTE latent space. The images at the top are visualized directly based on the image output produced by the decoder, whereas the images at the bottom are visualized using the nearest neighbor method [8], [19]. We can see that the SSTE decoder not only produces novel game level segments but also novel game tiles. However, the image quality needs to be improved. While the nearest neighbor method produces good quality images, they cannot produce novel game tiles. Thus, future work needs to focus on developing methods that produce higher quality images.

Generalized representations, such as TE, are derived by leveraging features common across a range of games since the goal is to represent levels from a large number of games. This generalization however often necessitates ignoring or averaging out features that are unique to individual games, particularly as the scale of covered games increases. Future work could investigate methods that allow such generalized embeddings to represent a greater variety of features and tile types, accommodating the unique facets of different games as well as taking into account game mechanics and dynamics.

For experiment VIII, for all the TE approaches, we used only the annotated games, as both SSTE and OTE were never evaluated for LB before. However, based on the results of both Sections VI and VII, we can see that SSTE can be extended to unannotated games without significant effort. Therefore, further study needs to be done for LB with SSTE, along with combinations of annotated and unannotated games.

We identify that due to their high-performance at approximating affordance information for unannotated games, SSTE could

be applied to PLGML tasks for unseen or even in-development games. Since SSTE only requires image information, it could allow for paradigms such as game developers providing concept images of game levels, with SSTE used to annotate the likely mechanical information and as a representation for novel level generation. As part of this application we anticipate the need for large, pre-trained, and general level generation models leveraging SSTE. Such models were previously impossible with the limited amount of available training data. But with SSTE, we can now train such a generator on all available tile-based games.

XIV. CONCLUSION

In this article, we introduced SSTE, a novel TE approach and representation that enables learning from unannotated games. Over a series of experiments, we demonstrated SSTE performed equivalently to prior TE representations in terms of image quality and playability, and outperformed them at affordance approximation. This representation thus offers improved generalizability over previous methods, opening the door for a much wider set of possible applications.

ACKNOWLEDGMENT

The authors would like to thank the Digital Research Alliance of Canada for computing resource support.

REFERENCES

- [1] M. Guzdial, S. Snodgrass, and A. J. Summerville, *Procedural Content Generation via Machine Learning*, Springer, 2022. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-16719-5_4
- [2] A. J. Summerville, S. Snodgrass, M. Mateas, and S. Ontanón, “The VGLC: The video game level corpus,” in *Proc. AIIDE Workshop Exp. AI Games*, 2016.
- [3] A. Guez et al., “An investigation of model-free planning: Boxoban levels,” 2018. [Online]. Available: <https://github.com/deepmind/boxoban-levels/>

- [4] L. Ferreira and C. Toledo, "A search-based approach for generating Angry Birds levels," in *Proc. IEEE Conf. Comput. Intell. Games*, 2014, pp. 1–8.
- [5] M. Stephenson and J. Renz, "Procedural generation of complex stable structures for Angry Birds levels," in *Proc. IEEE Conf. Comput. Intell. Games*, 2016, pp. 1–8.
- [6] I. Karth, B. Aytemiz, R. Mawhorter, and A. M. Smith, "Neurosymbolic map generation with VQ-VAE and WFC," in *Proc. 16th Int. Conf. Found. Digit. Games*, 2021, pp. 1–6.
- [7] R. Mawhorter, B. Aytemiz, I. Karth, and A. Smith, "Content reinjection for super metroid," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2021, vol. 17, no. 1, pp. 172–178.
- [8] M. Jadhav and M. Guzdial, "Tile embedding: A general representation for level generation," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2021, pp. 34–41.
- [9] M. Jadhav and M. Guzdial, "Clustering-based tile embedding (CTE): A general representation for level design with skewed tile distributions," 2022, *arXiv:2210.12789*.
- [10] S. Karakovskiy and J. Togelius, "The Mario AI benchmark and competitions," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 55–67, Mar. 2012.
- [11] A. Khalifa, P. Bontrager, S. Earle, and J. Togelius, "PCGRL: Procedural content generation via reinforcement learning," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2020, pp. 95–101.
- [12] R. Jain, A. Isaksen, C. Holmgård, and J. Togelius, "Autoencoders for level generation, repair, and recognition," in *Proc. ICCG Workshop Comput. Creativity Games*, 2016.
- [13] F. Abraham and M. Stephenson, "Utilizing generative adversarial networks for stable structure generation in Angry Birds," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2023, pp. 2–12.
- [14] M. Guzdial and M. Riedl, "Learning to blend computer game levels," 2016, *arXiv:1603.02738*.
- [15] A. Sarkar, M. Guzdial, S. Snodgrass, A. Summerville, T. Machado, and G. Smith, "Procedural content generation via knowledge transformation (PCG-KT)," *IEEE Trans. Games*, vol. 16, no. 1, pp. 36–50, Mar. 2024.
- [16] A. Sarkar, A. Summerville, S. Snodgrass, G. Bentley, and J. Osborn, "Exploring level blending across platformers via paths and affordances," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2020, pp. 280–286.
- [17] A. Sarkar, Z. Yang, and S. Cooper, "Controllable level blending between games using variational autoencoders," 2020, *arXiv:2002.11869*.
- [18] A. Sarkar and S. Cooper, "Latent combinational game design," *IEEE Trans. Games*, vol. 16, no. 3, pp. 659–669, Sep. 2024.
- [19] V. S. R. Atmakuri, S. Cooper, and M. Guzdial, "Game level blending using a learned level representation," in *Proc. IEEE Conf. Games*, 2023, pp. 1–8.
- [20] Z. Yang, A. Sarkar, and S. Cooper, "Game level clustering and generation using gaussian mixture VAEs," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2020, pp. 137–143.
- [21] S. Snodgrass, "Towards automatic extraction of tile types from level images," in *Proc. Artif. Intell. Interactive Digit. Entertainment Workshops*, 2018.
- [22] N. Mirgati and M. Guzdial, "Joint level generation and translation using gameplay videos," in *Proc. 2023 IEEE Conf. Games*, 2023, pp. 1–10.
- [23] D. Smirnov, M. Gharbi, M. Fisher, V. Guizilini, A. Efros, and J. M. Solomon, "MarioNette: Self-supervised sprite learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 5494–5505.
- [24] N. Y. Khameneh and M. Guzdial, "Entity embedding as game representation," 2020, *arXiv:2010.01685*.
- [25] D. Ha and J. Schmidhuber, "World models," 2018, *arXiv:1803.10122*.
- [26] N. Y. Khameneh and M. Guzdial, "World models with an entity-based representation," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2022, pp. 215–222.
- [27] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words," *Neurocomputing*, vol. 139, pp. 84–96, 2014.
- [28] N. Simidjievski et al., "Variational autoencoders for cancer data integration: Design principles and computational practice," *Front. Genet.*, vol. 10, 2019, Art. no. 1205.
- [29] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.
- [30] M. Ester et al., "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 1996, pp. 226–231.
- [31] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.
- [32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [33] A. Summerville et al., "Procedural content generation via machine learning (PCGML)," *IEEE Trans. Games*, vol. 10, no. 3, pp. 257–270, Sep. 2018.
- [34] A. Saravanan and M. Guzdial, "Pixel VQ-VAES for improved pixel art representation," 2022, *arXiv:2203.12130*.
- [35] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6629–6640.