Recognizing Multiplayer Behaviors Using Synthetic Training Data

Andrew Feng
Institute for Creative Technologies
University of Southern California
Los Angeles, CA USA
feng@ict.usc.edu

Andrew S. Gordon
Institute for Creative Technologies
University of Southern California
Los Angeles, CA USA
gordon@ict.usc.edu

***Abstract*—Accurate recognition of group behaviors is essential to the design of engaging networked multiplayer games. However, contemporary data-driven machine learning solutions are difficult to apply during the game development process, given that no authentic gameplay data is yet available for use as training data. In this paper, we investigate the use of synthetic training data, i.e., gameplay data that is generated by AI-controlled agent teams programmed to perform each of the behaviors to be recognized in groups of human players. The particular task we focus on is to recognize group movement formations in player-controlled avatars in a realistic virtual world. We choose five typical military team movement patterns for the formation recognition task and train machine learning models using procedurally generated unit trajectories as training data. The experiments were conducted using ResNet and EfficientNet, which are two popular convolutional neural network architectures for image classifications. The synthetic data is augmented by creating variations in image rotation, unit spacing, team size, and positional perturbations to bridge the gap between synthetic and human gameplay data. We demonstrate that high-accuracy behavior recognition can be achieved using deep neural networks by applying the aforementioned data augmentation methods to simulated gameplay data.**

***Index Terms*—multiplayer games, behavior recognition, synthetic training data**

I. Introduction

A central aim of interactive computer games, across various gameplay styles, is to create experiences that are reactive to player actions. In well-crafted games, appropriate reactions demonstrate to players that their actions have consequences in the game world, and help to instill a sense of agency that is essential to the notion of play. Networked multiplayer games, however, can pose difficult challenges to the engineering of appropriate reactions, particularly in games where groups of players operate as a team in open worlds. In such games, the actions of the team are implicit and often contain higher level semantic meaning, arising from the explicit collective actions of individual players. Thus while a game engine can hold game state such as positions, speed, and current action of each individual unit, the semantic meaning from a team's collective action is not directly represented in any game state that is readily accessible to the underlying game engine. For example, a multiplayer game can clearly observe (and react) when individuals are running, shooting, and manipulating objects in the environment, but be oblivious to the a team's efforts to fortify a position, lure enemies into an ambush, or coordinate a siege of an opponent's stronghold.

Explicitly representing group actions in multiplayer computer games can be viewed as a type of time-series classification problem, where sequences of observable actions of individuals serve as input, and the output are segments labeled with descriptors that best describe the group behavior. Machine learning is the dominant approach to solving time-series classification problems, particularly the use of deep neural networks. Recent advances in time-series analysis have seen the application of recurrent neural networks, encoder-decoder networks, and transformer networks to diverse applications such as speech recognition and event detection in video streams. While these contemporary machine learning methods have greatly reduced the need for feature engineering and the pre-processing of raw input data streams, they demand enormous amounts of training data – labeled and unlabelled – before they exhibit usable levels of classification accuracy.

For developers of multiplayer games, this requirement for enormous amounts of training data creates an untenable dilemma: the game has to ship in order to collect the necessary multiplayer data at scale, but cannot ship before the data has been collected and incorporated. Needed are methods for bootstrapping the data collection cycle, giving developers sufficient classification accuracy when no data is available so that they can craft playable games, which can then be used to collect multiplayer behavior data to further improve classification accuracy.

In this paper, we investigate the use of synthetic training data to bootstrap the recognition of multiplayer behaviors when no real data is yet available. The particular behaviors we are aiming to recognize in this work are group movement formations. Such behavior has well-defined movement patterns and its unit positions and velocity can be represented as rasterization images. Therefore the recognition task can be reduced into an image classification problem and its training

The projects or efforts depicted were or are sponsored by the U. S. Army. The content or information presented does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

978-1-7281-4533-4/20/$31.00 ©2020 IEEE

data can be generated automatically as procedural unit trajectories. In this approach for generating synthetic training data, human players are replaced by teams of computer-controlled agents programmed to execute group movement formations in the game world. Designed to resemble the behavior of human players, these agent teams simulate each formation in varied locations and situations, providing unlimited amounts of labeled gameplay data for use in training machine learning algorithms. We investigate this approach for the task of recognizing movement formations in small teams of players, as would be observed in small-unit tactical military games. For five movement formations, we simulate varied-sized teams executing the movement formation on varied terrain, producing labeled training data. We evaluate the accuracy of learned models on human gameplay data collected using a networked multiplayer testbed, where player-controlled avatars execute each movement formation as a team. To better adapt from synthetic data to human gameplay data, we explore the use of a variety of data augmentation methods and model ensembling of different neural network architectures. Our findings show that applying data augmentation methods and model ensembling are effective at bridging the gap between synthetic training data and human gameplay data. The result demonstrates that high-accuracy multiplayer behavior recognition is possible even in zero-data development contexts.

II. Related Work

A. Behavior Recognition in Videogames

Recognizing multiplayer behaviors in video games has been pursued using a variety of approaches in previous work. Our approach contrasts most directly with that of Ahmad et al. [1], who address the problem of labeling the group behaviors in large amounts to multiplayer gameplay data. Their approach, Interactive Behavior Analysis, involves the iterative visualization, labeling, and clustering of gameplay data, with the aim of providing a tool for researchers and game developers for better interpreting and analyzing multiplayer strategies and tactics. Our current effort focuses on solving a different problem, aiming to provide a means of recognizing multiplayer behaviors earlier in the game development process, before large amounts of data can be collected from players.

Other work has investigated supervised approaches to multiplayer behavior recognition. Sukthankar and Sycara [15] investigate multiplayer behavior recognition in a task that is most similar to our own, recognizing team formations and tactical behaviors in a multiplayer military-oriented game environment. Their approach involves the specification of team behavior templates, which are matched to player positions and classified using Hidden Markov Models, trained using annotated gameplay data from two-person teams.

Several previous research efforts have focused on the classification of the behavior of single players in videogames, often applying methods for plan and intention recognition to gameplay contexts. For example, recent work by Min et al. [9] achieves state of the art performance on goal recognition from mulitmodal data (gameplay and eye tracking) using supervised machine learning with LSTM neural networks, with gains over previous research using Markov Logic Networks [3]. In our current research, we capitalize as well on continuing advances in neural networks as a technology for classification, but focus instead on the problem of training such models when no data is available.

B. Synthetic Training Data

Learning from simulated data has been a promising direction, with progress in fields such as reinforcement learning and robotic vision. Despite the potential of generating unlimited labelled data directly from simulation, it remains a challenge to bridge the 'reality gap', which is caused by the discrepancy between simulated data and the data collected from the real-world. In the field of computer vision, several directions have been explored to improve the model trained on simulated data.

The most straightforward idea is to improve the quality of photo-realistic rendering such that the simulated images are close enough to the real-world. This strategy has been shown with reasonable success for certain vision tasks such as depth estimation [10] and image segmentation [12, 14]. However, authoring scenes to generate highly realistic imagery can be both compute and labor intensive, especially for complicated and animated environments.

Another direction is to randomize the source domain data generated via simulation so that the synthetic training data consist of enough variations. The key idea is to use domain randomization as a means to encourage neural networks to learn a more generalized feature representation from simulated data. Therefore when applying such a model in the real data, the domain gap will not be huge.

Tobin et al [17] introduced a domain randomization technique to adapt synthetic training data for object detection. The key idea is to intentionally avoid photo-realistic rendering and randomly generate image variations with different lighting, textures, and camera poses. This approach encourages the network to learn enough domain variations so that it can treat real-world data as simply another variation. Dundar et al [2] proposed to apply image stylization for generating image variations that cover different domains. Instead of varying every parameter randomly, Prakash et al. [11] proposed to add the scene context into consideration by randomly adjusting objects based on a pre-defined scenario structure. This ensures that the synthetic data are realistic while producing enough variety. Meta-sim [7] extends this idea further by optimizing probabilistic scene grammars to produce scenes that better match the real-world scene distributions. It is therefore able to adapt the simulator and generate synthetic data that improves the real-world performance.

One other effective strategy is to leverage semi-supervised learning techniques to enforce consistency and smoothness when predicting unlabeled data collected from the real-world. The main idea is to introduce a consistency loss in the learning process with unlabelled data to make the resulting model invariant to data perturbations and style variations. The typical methods include a model ensemble approach [13, 8],

which produces variations in network structures to enforce the consistency loss, or adversarial approach, which seeks to minimize an adversarial loss to obtain style-invariant feature representations [5, 6].

In this paper, we utilize aspects of each of these previous approaches. Analogous to previous work using photo-realistic imagery, we attempt to generate synthetic behaviors that closely resemble those that would be executed by players. However, since both the synthetic data and multiplayer behaviors are executed in the same visual game environment, our focus is on behavioral realism, rather than visual realism. We utilize several domain augmentation methods to create enough variations in the synthetic data to ensure the learned model can generalize well to data collected from players. Finally, we use a model ensemble approach to improve the robustness of learned classifiers.

III. Overview

Our approach to multiplayer behavior recognition in videogames follows a traditional machine learning approach, but where the training data is procedurally generated by teams of AI-controlled agents rather than collected from human players. Figure 1 shows the overall stages of our workflow. Large amounts of synthetic training data are first generated in a target gaming environment by teams of AI-controlled agents, programmed to execute each of the behaviors that are to be recognized in the behavior of human players. In this research, the target behaviors consist of movement formations performed by small teams, and are described in the next section. For the purpose of evaluating the accuracy of our approach, real multiplayer data is similarly collected in the gaming environment. These datasets, consisting of trajectory information for avatars in the game, are then rasterized into individual images at each timestep to reduce the recognition problem into an image classification task. To improve accuracy of the resulting model in recognizing player behavior, various data augmentations are applied to enhance the synthetic data. Finally, an ensemble of multiple image classifiers is trained using the augmented synthetic data, and used to predict the behavior labels for the multiplayer data. Each of these steps is further described in the following sections.

IV. Multiplayer Test Data

The central aim of our approach to behavior recognition is to reduce or eliminate the need to collect and annotate copious amounts of multiplayer gameplay data. In order to fully evaluate our approach, however, it was necessary to collect some amount of gold-standard test data from multiplayer gameplay sessions, annotated using a set of labels for the target behaviors. To collect this testing data for our experiments, we developed a basic online multiplayer 3D game environment, and enlisted players (from our research lab) to jointly execute the target behaviors within this environment.

Our testbed game environment, depicted in Figure 2, was modeled after contemporary tactical military games such as Activision's *Call of Duty 4: Modern Warfare* and Bohemia Interactive's *Arma 3*, where players control lifelike avatars organized into small teams of warfighters engaged in combat on realistic terrain. Built using the Unity game engine, our testbed enables each player to control the movement of an animated virtual soldier in a realistic 3D environment modelled after a notional Middle-eastern desert city. Using the analog thumbstick on an Xbox gamepad controller, players control the direction and speed of their soldier as they navigate through outdoor regions of a city, and coordinate their movements with avatars controlled by other networked players. During the collection of test data, all participants communicated via an audio teleconference to facilitate coordinated action among players.

The group behaviors executed and recognized in this research consisted of **movement formations**, analogous to the various types of squad-level tactical movements performed by military ground forces, e.g., *squad column* and *squad wedge* [18]. To simplify the execution of different movement formations by our (civilian) participants, we opted to use five notional movement formations in lieu of authentic military formations. The five formations executed and recognized in this research were labelled *wedge*, *line*, *column*, *arrow*, and *cluster*, and are depicted in Figure 3.

The test data for movement formations was collected from seven participants in a single 30-minute session, consisting of a 15-minute practice phase and a 15-minute collection phase. A single participant was selected as the "leader," while others assumed the role of a "follower." During the practice phase, the leader announced a desired movement formation to be followed as they navigated the environment, adjusting the position of followers via verbal commands until the formations closely matched those depicted in Figure 3. During the subsequent collection phase, the facilitator of the session (also on the teleconference call) prompted the participants to execute the movement formation types, in random order, at intervals of approximately 60 seconds.

During the collection phase, the coordinate positions of each participant's avatar on the virtual terrain were sampled each 100 milliseconds. Intervals in this time-series dataset were subsequently labeled with an intended movement formation class by aligning it with an audio recording of the session's teleconference. The resulting test data set was recorded by 7 players and consisted of 4221 labeled samples from 9 executions of different movement formations.

V. Generation of Synthetic Training Data

In this research, synthetic training data is used to enable multiplayer behavior recognition, allowing game developers to build games based on these behaviors before traditional supervised training data can be collected and annotated. To generate synthetic training data for the recognition of movement formations, we utilize the same game environment described in Section IV, developed using the Unity game engine. The main difference is how the movements of soldiers in this environment are controlled. Instead of using Xbox gamepads to manually control the direction and speed of soldiers, the

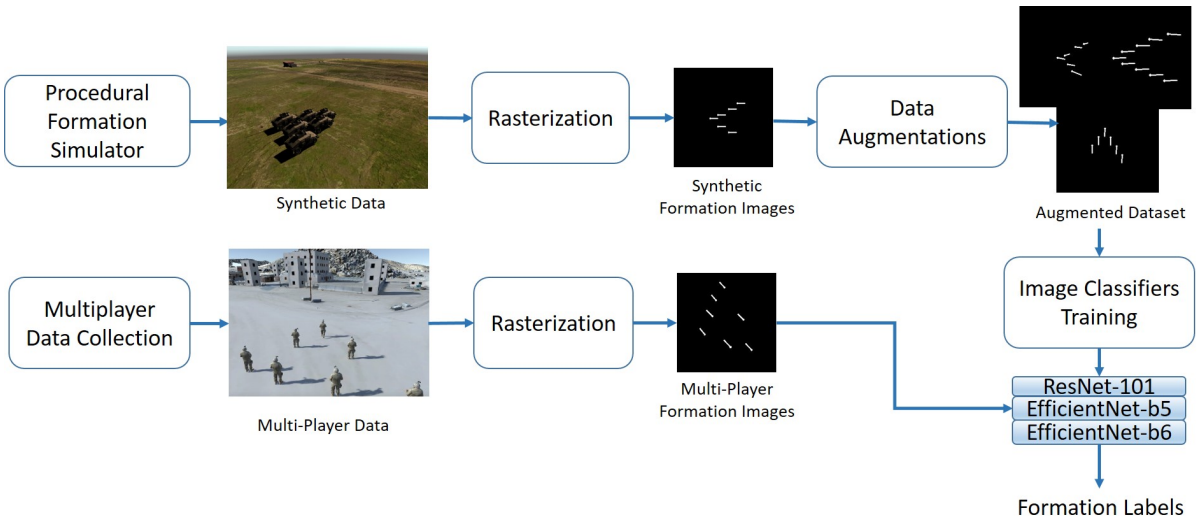


Fig. 1. Overview of the behavior recognition process using synthetic training data.

Fig. 2. Screenshot of testbed game environment.

movement trajectory for a given unit is procedurally generated using simple AI scripts, using parameters that specify the formation type, number of units, and spacing between units.

A. Procedural Unit Trajectory Generation

Given the formation parameters and a goal location for the leader unit, the movement formation trajectories are created by the following steps. 1) The initial unit location offset from the nearby units is calculated based on the formation type and unit spacing. With the leader designated as unit 0, Figure 4 illustrates the ordering of follower units and the corresponding offsets, where the number within each circle indicates the unit index and the red arrows indicate offsets to the adjacent units. Since these offsets can be calculated using only the unit index and spacing, it is straightforward to extend the calculation to groups of any number of units. 2) given the initial unit offsets and goal location, the target location for each unit is calculated by adding the rotated offset to goal location. The rotation angle is based on the current direction the unit is facing and direction toward the target location. 3) Once the target location is known, the trajectory for each unit is computed using a navigation mesh for the environment terrain. This allows each unit to move individually and avoid obstacles as needed while maintaining the overall formation throughout the movements.

For the purpose of evaluating our approach, we generated about 130 seconds of synthetic training data for each of the five movement formation classes, where each movement formation was led by a selected leader unit, who was directed to move to a random location in the virtual terrain.

Similar to multiplayer sessions, simulated positional information for each unit in the group was also sampled at a rate of 0.1 Hz, and recorded along with the movement formation class that was being executed at the time. This resulted in a total of 6500 labelled samples from a simulation session.

B. Rasterization of Trajectories as Images

In its raw form, the aforementioned movement formation trajectories and labels are represented as a multi-variate time series sampled at a rate of 0.1 Hz. Accordingly, a classifier could be trained using a vector of unit positions at each timestamp as input to predict the corresponding label. However, while it is straightforward to use a positional vector as input, the resulting classifier would need to be retrained for groups of every size, as the size of the input vector would vary. We solve this problem by representing the trajectories of the units in the whole group as a fixed-sized image. That is, instead of working with vector data directly, we rasterized unit positions at each timestamp into a 2D image, and used that as the data representation for training the classifier. This approach allows for a unified representation to train a single classifier that can handle different number of units. Moreover, this reduces

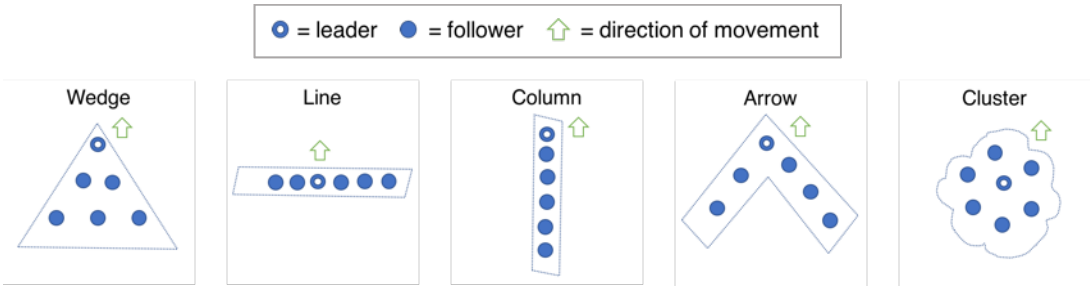


Fig. 3. Movement formations used as target group behaviors.

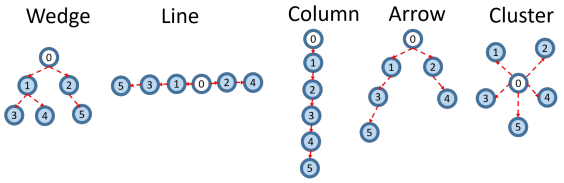


Fig. 4. Examples of movement formations offsets based on the formation type and the number of units. Here the number within each circle indicates the unit index, with unit zero serving as the team leader. Each red arrow indicates the offset to an adjacent unit.

the problem into an image classification task, and allows us to utilize existing convolutional neural network (CNN) architectures that perform well in other visual recognition tasks.

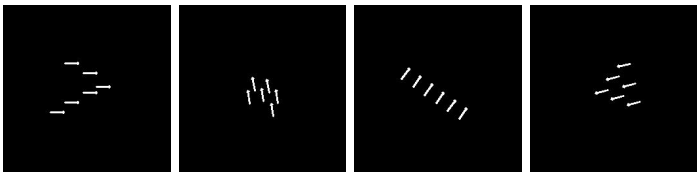


Fig. 5. Examples of unit movement formations as rasterization representation. Different formation types result in different image features that are suitable for an image classifier.

The rasterization representation is generated as follows. Given positions ${p_1}^t, {p_2}^t, \dots {p_n}^t$ at time step t, we first compute the X-Z local coordinates with origin centered at ${p_{avg}}^t = \frac{1}{n}\sum_{i=1}^{n} {p_k}^t$, which is the average position between all units at time t. A 2D grid center at ${p_{avg}}^t$ is then used to rasterize all unit movements at t. However, simply plotting positions as dots does not provide kinetics information for the units. Therefore we plot the offset ${o_i}^t = {p_i}^t - {p_i}^{t-1}$ between t and the previous time step $t-1$ as an arrow originated at ${p_i}^t$ to better encode the unit movement velocity in a single image. Figure 5 shows some examples of rasterization representation for different movement formations.

This rasterization method was applied to each of the samples in the synthetic training data, yielding a dataset of 6500 annotated images. Likewise, we applied the same rasterization method to the unit trajectories in our multiplayer test dataset, representing the position of human-controlled units in our testbed environment, yielding a test set of 4221 annotated images. Since the movements at each time step are represented as images, high-performance CNN architectures for image classification can be applied to predict the correct movement formation label using images of synthetic movement formations as training data. While this is a relatively straightforward task given the mature technologies of image classification research, the utilization of this synthetic training data requires that we first overcome the domain gap between our synthetic data and the trajectories produced by human-controlled avatars in a multiplayer environment.

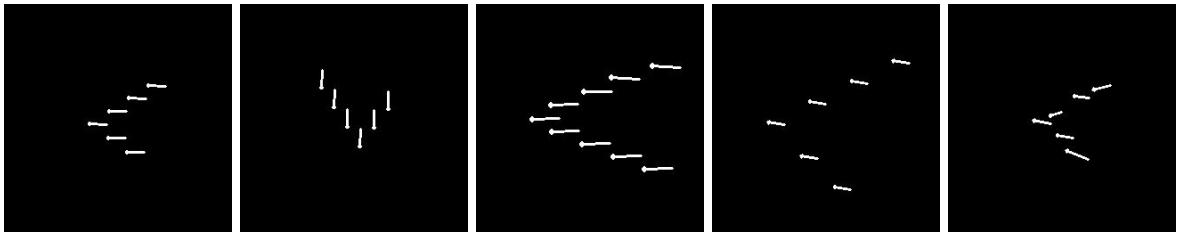
VI. Domain Adaptation

The main challenge of using a model trained with synthetic data is to overcome the domain gap when the testing data is collected from the multiplayer world. Naively training a classifier with synthetic data without any augmentation or adjustments tends to result in a model that is over-fitted to the statistics of the synthetic data. Therefore the accuracy will be poor when applied to a multiplayer world use case. The main purpose of domain adaptation is to ensure that the model can generalize well to the multiplayer world context using various learning and augmentation techniques.

A. Data Augmentation Approach

One popular way of improving robustness of a model is to augment the training data with various transformations and perturbations. The technique virtually expands the size of training data by creating many variations of the same data. It prevents over-fitting by forcing the model to learn a more general pattern from the data. Since both synthetic and multiplayer data are rasterized in the same manner using the same colors, we choose to only apply geometric transformations and skip any color transformations in the augmentation process. The types of augmentations used in this work include rotation variations, unit spacing variations, team size variations, and positional perturbation.

a) Rotation Variations: Since the formation type remains the same for any rotations along Y-axis, we rotate each image with a random angle at each epoch before feeding it into the training batch. This ensures that the CNN learns a rotational invariant representation for predicting formations.



(A) Original (B) Angle (C) Size (D) Space (E) Noise

Fig. 6. Examples of data augmentations for formation images. (From left to right) : original image, rotation variations, team size variations, unit spacing variations, and position perturbations

b) Unit Spacing Variations: The space between adjacent units may vary in the multiplayer world as each player may space their avatar differently in a formation. Generating synthetic formations with varying unit spacing helps the resulting model to generalize better to such a discrepancy in the real data.

c) Team Size Variations: In the synthetic training data generation, the number of units can be specified before procedurally simulating the formation trajectories. By varying the team size in each image, the learning process will not be restricted to recognize the formation pattern with a fixed number of units. Thus a single classifier can be generalized to classify formations with different team sizes.

d) Positional Perturbation: Since each user will control a single unit, a group of units tends to not move in perfect harmony and may be adjusting their positions from time to time. It makes the data collected in the multiplayer world more noisy than simulated data, where all units are coordinated programatically when moving in formations. To alleviate this domain gap, a Gaussian noise is added to units' positions to perturb the resulting trajectories. Note that this noise is random and the goal is not to really re-create the styles of human player trajectories. Instead, having these perturbations improves robustness in the learning process, and the resulting model will generalize better toward multiplayer data.

Figure 6 shows examples of aforementioned data augmentations applied on the formation images. Note that while data augmentation techniques tend to be applied for supervised learning in general, here they are applied in the context of domain adaptations to help adapt the model trained on synthetic data to data collected from multiplayer sessions.

B. Model Ensemble Approach

Creating ensembles of models is a well-known technique in machine learning to boost generalization capability for the trained models. It works by independently training multiple models and then combining them into an ensemble. The final predictions are obtained by averaging the results from each model to improve test accuracy.

Two model architectures, ResNet [4] and EfficientNet [16], are used to form the model ensembling. ResNet is a popular architecture that has been utilized as the feature extraction backbone in many computer vision tasks such as object detection and instance segmentation. EfficientNet is currently state of the art architecture for image classification. It is created via neural architecture search (NAS) to maximize the performance while using less parameters than previous models. Another advantage of these two models is that they both include multiple variations of network depths to handle tasks of different complexity and there exist pre-trained models ready for transfer learning.

In this work, we first train the aforementioned models with different depths on the same training data and select the top two model architectures with the highest individual testing accuracy to form the ensembles. We fuse the results by averaging the vector output from the last linear layer of each model and feed the averaged vector into the logistic softmax function to obtain the final prediction of class label.

VII. Experiments

We conducted two sets of experiments to evaluate our approach to domain adaptation, separately investigating the contribution of various data augmentation techniques and the ensemble model approach. In each of these experiments, behavior recognition models were trained on the labeled rasterized images of the synthetically-generated movement formations, and tested on rasterized images obtained from our multiplayer data collection exercise. All neural network models were trained within the PyTorch neural network framework for 50 epochs with batch size of 16 per GPU. Adam optimizer is used for training with learning rate of 0.0001 and cosine annealing schedule for weight decay. Accuracy scores were computed as the percentage of correct behavior labels assigned to samples in the multiplayer test set.

In our first set of experiments, we assessed the contribution of each of our data augmentation methods to improvements in classification accuracy. For this, we performed an ablation study by incrementally adding individual data augmentations and using different network architectures.

Table I shows the resulting classification accuracy on the test dataset with different augmentation methods. The model naively trained with synthetic data, without any augmentation, produced poor results with only about 31% accuracy. By adding more augmentations incrementally, the resulting accuracy improves as the model starts to learn more general formation patterns from the expanded data, and the resulting accuracy is able to reach about 80% accuracy. Note that augmentations with varying team sizes and unit spaces expand

the number of training samples due to additional simulation sessions with different size and space parameters. Since the positional augmentation creates a perturbed version of the original data and the rotation augmentation is applied during the training on-the-fly for each image at each epoch, they do not increase the total training data size. As in previous research on the use of synthetic training data, our results demonstrate that data augmentations is a good practice to help bridge the domain gap between synthetic behavior data and multiplayer behavior data.

TABLE I
EVALUATION OF DIFFERENT DATA AUGMENTATIONS. ALL EXPERIMENTS ARE DONE USING EFFICIENTNET-B5.

Augmentation	Data Size	Accuracy
None	6500	31.61 %
Rotate	6500	52.10 %
Rotate + Space	19500	67.34 %
Rotate + Space + Size	39000	71.13 %
Rotate + Space + Size + Noise	39000	80.11 %

In our second set of experiments, we investigated the effect of training with different network architectures, depths, and ensembles. The resulting accuracy is summarized in Table II. Overall, the EfficientNet tends to achieve higher accuracy than ResNet with similar number of parameters, which demonstrates its superior model architecture via NAS. The ensemble models also show accuracy improvements over any single networks except when combining two EfficientNet-b5 and b6 together. Empirically, this shows that combining different model architectures into an ensemble is more effective than the same architecture with varying depths. Since different architectures are likely to learn and extract different types of features, the combination of them can complement each other to produce better results. As a result, our best ensemble model is able to provide additional boost and achieve 85% accuracy on the multiplayer dataset.

TABLE II
EVALUATION OF DIFFERENT MODEL ARCHITECTURES. ALL EXPERIMENTS ARE DONE USING ALL DATA AUGMENTATIONS.

Architecture	Params	Accuracy
EfficientNet-b5	30M	80.11 %
EfficientNet-b6	43M	81.89 %
ResNet-50	25.6M	73.81 %
ResNet-101	44.5M	79.45 %
Ensemble (b5+b6)		81.87 %
Ensemble (b5+ResNet-101)		83.41 %
Ensemble (b6+ResNet-101)		85.07 %

VIII. DISCUSSION AND FUTURE WORK

While contemporary machine learning methods have great potential for creating new gameplay experiences in videogames, their need for large amounts of data pose significant challenges for developers. Specifically, if the correct classification of player behavior is instrumental to the gameplay mechanics, then developers are faced with a difficult problem: How can a high-performance behavior recognizer be trained from authentic gameplay data, if collecting the data requires that the recognizer already exists?

The solution that we investigate in this paper is to train the recognition algorithms using synthetic training data, as a substitute for real annotated data that can be collected after the game is shipped. Given sufficient accuracy, recognition based on synthetic training data allows game designers to build appropriate game-world reactions for players, before the first player is immersed in the game.

The approach that we pursue in this paper is to use AI-controlled agents to simulate the behaviors that are to be recognized in multiplayer gameplay data. This synthetic behavior data is rasterized as images to make it possible to generalize over groups of different sizes, and to utilize contemporary CNN architectures for image classification. To further aid in accurate classification, we utilize a model ensemble approach, combining image classification networks with varying topology. On the task of movement formation classification, our approach achieves strong results, with 85% accuracy on a five-class classification problem with no authentic training data.

There are other possible approaches to behavior classification without data, such as template matching and rule-based heuristic methods. However, the main advantage of our approach is that it seamlessly improves with the addition of actual annotated data that can be collected after the game ships. In this development model, beta testers and early adopters would play versions of the game built largely from synthetic training data. Their gameplay data would be collected, analyzed, and annotated, and used to retrain the neural networks used for behavior recognition. Subsequent builds and updates to the game would change only the learned parameters of neural network models, a relatively low-risk modification compared to architectural changes in game mechanics.

This development model points to a number of important avenues of future work. Along with the exploration of continuing advances in neural network architectures for classification, it will be important to investigate how synthetic training data and authentic gameplay data can best be combined to maximize accuracy. As well, we see opportunities for the use of synthetic training data in the analysis of behaviors in large-scale gameplay datasets, with possibilities for incorporating synthetic data in existing systems for visualization and clustering of multiplayer behaviors [1].

Within the context of our existing approach, a central focus of future work will be on the collection of large amounts of authentic multiplayer gameplay data and on the recognition of a broader set of multiplayer behaviors. Of particular interest is the applicability of our approach to complex, multi-step behaviors that involve interactions among and between groups, e.g., the tactical inter-group behaviors of *ambushes*. This will likely require training a classifier for videos instead of images to incorporate data across multiple time steps for better prediction of complex behaviors. With more data collected through gameplay sessions in the future, we also expect to

further improve the classifier for multiplayer data using semi-supervised learning techniques. This can be done by utilizing unlabelled multiplayer data and a consistency loss function in the training process to allow the classifier to better learn the data statistics in the target domain of multiplayer gameplay. We anticipate that different neural network architectures may be better suited to these sorts of tasks, as seen in other complex time-series classification tasks. Accurate recognition of complex multiplayer behaviors, in the absence of training data, would offer new opportunities for the design of multiplayer games where the collective behavior of a group of players is an integral part of the mechanics of the experience.

Acknowledgment

We thank Adam Reilly and Ed Fast for their contribution to the development of the networked multiplayer testbed described in this paper.

References

[1] Sabbir Ahmad et al. "Modeling Individual and Team Behavior through Spatio-temporal Analysis". In: Oct. 2019, pp. 601–612. ISBN: 978-1-4503-6688-5. DOI: 10.1145/3311350.3347188.

[2] Aysegul Dundar et al. "Domain stylization: A strong, simple baseline for synthetic to real image domain adaptation". In: *arXiv preprint arXiv:1807.09384* (2018).

[3] Eun Ha et al. "Recognizing Player Goals in Open-Ended Digital Games with Markov Logic Networks". In: *Plan, Activity, and Intent Recognition: Theory and Practice* (Mar. 2014), pp. 289–311. DOI: 10.1016/B978-0-12-398532-3.00012-9.

[4] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[5] Judy Hoffman et al. "CyCADA: Cycle-Consistent Adversarial Domain Adaptation". In: *International Conference on Machine Learning*. 2018, pp. 1989–1998.

[6] Weixiang Hong et al. "Conditional generative adversarial network for structured domain adaptation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1335–1344.

[7] Amlan Kar et al. "Meta-sim: Learning to generate synthetic datasets". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 4551–4560.

[8] Samuli Laine and Timo Aila. "Temporal Ensembling for Semi-Supervised Learning." In: *ICLR (Poster)*. OpenReview.net, 2017.

[9] Wookhee Min et al. "Multimodal Goal Recognition in Open-World Digital Games". In: *Proceedings of the Thirteenth Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 2017.

[10] Benjamin Planche et al. "Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition". In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 1–10.

[11] Aayush Prakash et al. "Structured domain randomization: Bridging the reality gap by context-aware synthetic data". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7249–7255.

[12] Stephan R Richter et al. "Playing for data: Ground truth from computer games". In: *European conference on computer vision*. Springer. 2016, pp. 102–118.

[13] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. "Regularization with stochastic transformations and perturbations for deep semi-supervised learning". In: *Advances in neural information processing systems*. 2016, pp. 1163–1171.

[14] Fatemeh Sadat Saleh et al. "Effective use of synthetic data for urban scene semantic segmentation". In: *European Conference on Computer Vision*. Springer. 2018, pp. 86–103.

[15] Gita Sukthankar and Katia Sycara. "Robust Recognition of Physical Team Behaviors Using Spatio-Temporal Models". In: *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*. AAMAS '06. Hakodate, Japan: Association for Computing Machinery, 2006, pp. 638–645. ISBN: 1595933034. DOI: 10.1145/1160633.1160746. URL: https://doi.org/10.1145/1160633.1160746.

[16] Mingxing Tan and Quoc Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *International Conference on Machine Learning*. 2019, pp. 6105–6114.

[17] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[18] U.S. Army. *Foundations of Leadership: MSL II*. New York, NY: Pearson Custom Printing, 2008.