A Framework for Action Detection in Virtual Training Simulations Using Synthetic Training Data

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

ABSTRACT

In virtual military training, tracking and evaluating trainee behavior throughout a simulation exercise helps address specific training needs, improve the realism of simulations, and customize the training experience. While it is straightforward to parse the event log of a simulation to identify atomic behaviors such as unit movements or attacks, it remains difficult to fuse these events into higher-level actions that better characterize trainees' intentions and tactics. For example, if each unit is controlled by an individual trainee, how should the movement information from all units be aggregated to determine what formation the group is moving in? Similarly, how can all of the information from nearby terrain environments be combined with kinetic actions to determine whether the trainees are executing an ambush attack, or is simply engaging the enemy group? While an experienced human observer-controller can quickly assess the battle map to provide an appropriate interpretation for such events, it remains a challenging task for computers to automatically detect such high-level behaviors when performed by human trainees.

In this work, we proposed a machine-learning (ML) framework for recognizing tactical events in virtual training environments. In our approach, unit movements, surrounding environments, and other atomic events are rasterized as a 2D image, allowing us to solve the action detection problem as image classification and video temporal segmentation tasks. In order to bootstrap ML models for these tasks, we utilize synthetic training data to procedurally generate a large amount of annotated data. We demonstrate the effectiveness of this framework in the context of a virtual military training prototype, detecting troop formations and other tactical events such as ambush and patrolling.

ABOUT THE AUTHORS

Andrew Feng is a Research Scientist at the Institute for Creative Technologies at the University of Southern California, working on the One World Terrain project. Previously, he was a research associate focusing on character animation and automatic 3D avatar generation. His research work involves applying machine learning techniques to solve computer graphics problems such as animation synthesis, mesh skinning, and mesh deformation. He received his Ph.D. and MS degree in computer science from the University of Illinois at Urbana-Champaign. Email: feng@ict.usc.edu

Andrew S. Gordon is a Research Associate Professor of Computer Science and Director of Interactive Narrative Research at the Institute for Creative Technologies at the University of Southern California. His research advances technologies for automatically analyzing and generating narrative interpretations of time-series data. He received his Ph.D. in 1999 from Northwestern University. Email: gordon@ict.usc.edu

A Framework for Action Detection in Virtual Training Simulations Using Synthetic Training Data

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

INTRODUCTION

Among the most compelling use-cases for interactive virtual simulations is for virtual team training exercises, e.g., where teams of warfighters control their own virtual avatars in simulated battles, practicing tactical maneuvers and essential skills in increasingly challenging scenarios. While these computer-based exercises lack the physical stresses of live training, their great potential lies in their capacity for guided deliberate practice of skills, where the environments themselves are tailored to the abilities of the team and responsive to their successes and failures in their performance. In both live and virtual training, however, this responsiveness has generally required the participation of human facilitators, i.e., Observer Controllers that track the progress of trainees toward learning objectives, and actively shape the learning environment to further this progress. Automating these capabilities of human Observer Controllers would have several practical benefits for virtual training for operational units. Replacing training support staff with software reduces labor costs, at the very least. More importantly, such automation can help remove the reliance of operational units on outside contractors and home-station training facilities in conducting virtual training exercises. As this reliance is reduced, commanders are more able to lead their own training exercises, on their own schedules, wherever their point of need..

Much of the difficulty in automating the Observer Controller stems from the simulation environment's lack of understanding of what the human trainees are actually doing in the virtual space. Seeing a given formation and orientation of trainee avatars on a virtual ridge line, for example, it might be obvious to a human Observer Controller that the trainees were setting up for a deliberate ambush of an approaching enemy force, prompting him or her to modify the enemy's reaction to reinforce lessons related to the performance of this tactical maneuver. The software of the virtual simulation environment, however, would likely be oblivious to the impending ambush altogether, aware only of the avatars' positions in virtual space and the avatar controls provided by the trainees' user interfaces. Automating the human Observer Controller, in this case, requires an ability to recognize complex group behaviors based on the positions and individual actions of avatars in the group, within the battlefield context.

Analogous perceptual recognition tasks have been successfully automated in other domains using contemporary machine learning technologies, such as deep neural networks. Here, recognizing the complex behaviors of teams of trainees in virtual environments can be seen as a type of time-series classification task. The input consists of positional and low-level simulation event data over time, and output assigns labels to durations that best characterize the class of behavior that is being executed, from a given vocabulary. Today, a high-accuracy classifier of this sort could easily be constructed using supervised learning methods, provided that enormous amounts of data were available, expertly annotated with high levels of inter-rater agreement. However, the required amounts of data (perhaps tens of thousands of examples) is far beyond what might be reasonably obtained from the deployed use of any existing virtual training software. Even assuming such large-scale datasets could be collected, its annotation by teams of experts would be especially costly and difficult, given the sensitive nature of data generated during military training exercises. If contemporary machine learning technologies are to be used to automate functions of human Observer Controllers, an alternative approach to the collection and annotation of training data is needed.

In this paper, we investigate an alternate approach to the collection and annotation of datasets for behavior recognition using machine learning methods. Our approach involves the automatic generation of synthetic training data, collected by authoring behavior programs to be executed thousands of times by teams of fully-autonomous

agents within the target simulation environment. Multiple programs are authored, one for each of the classes of behavior that is to be recognized in teams of human trainees, which allows durations of generated data to be automatically annotated with the correct class label. We demonstrate this approach by generating behaviors corresponding to different troop maneuver formations, as would be executed by squads of Army soldiers (teams of 9 trainees). We evaluate the effectiveness of the resulting behavior classifier using gold-standard test data collected using volunteers in a testbed multiplayer simulation environment. Results indicate that high-accuracy recognition is possible using this approach, but requires the application of domain adaptation techniques to bridge the gap between synthetic data and human performance data.

RELATED WORK

Behavior Recognition in Virtual Simulations and Video Games

The problem of recognizing team behaviors in virtual simulations using supervised machine learning methods has been investigated previously by Sukthankar and Sycara (2006). In their work, two-person tactical maneuvers are recognized in a multiplayer, game-based simulation environment using hand-authored behavior templates, which are used to classify durations of gameplay data using trained Hidden Markov Models. Our approach is analogous in several respects, using hand-authored behavior programs rather than templates, but eliminates the additional step of hand annotating the training data used in supervised learning. Additionally, we address the more challenging problem of recognizing behaviors in larger teams (nine-person infantry squads).

Behavior classification has also been investigated in the analysis of gameplay data from multiplayer video games. Ahmad et al. (2019) demonstrate a successful approach to behavior classification that minimizes the hand annotation of training data. In their approach, referred to as Interactive Behavior Analysis, large amounts of multiplayer gameplay data are interactively analyzed through an iterative process of visualization, labeling, and clustering. Aimed at analysts who are interested in understanding player strategies and tactics, this descriptive approach allows for quick *post hoc* analyses with minimal labeling. Our approach contrasts with this previous work in that we aim to identify behaviors *in situ*, using prescriptive definitions, for the purpose of providing real-time responses within the simulation environment.

Methods for plan and intention recognition have also been applied for the recognition of single-player behaviors in video games. Early progress utilized Markov Logic Networks to recognize player goals in a single-player adventure game (Ha et al., 2014). State of the art performance expands on this approach by utilizing contemporary Long Short Term Memory neural networks, and by including player eye-tracking information additional input to the classification task (Min et al., 2017). Our approach is similar in its use of contemporary neural networks for classification, but aims to achieve high classification accuracy without the use of hand annotated training data.

Synthetic Training Data in Machine Learning

Progress in machine learning using neural networks has enabled high accuracy classification, typically without the need for the extensive engineering of input features seen in previous approaches. However, the tradeoff comes in the requirement for enormous amounts of training data, which can limit the application of these technologies in many domains. A promising direction is the use of synthetic training data, i.e., training data that is generated algorithmically rather than collected from real world task environments. While unlimited amounts of labeled synthetic training data can be obtained via this method, the discrepancy between synthetic and real-world data can limit the accuracy of the trained classifier in real-world contexts. To bridge this "reality gap," several successful approaches have been previously explored to adapt the synthetic data to the real-world domain.

First, efforts have been made to generate synthetic training data with high levels of realism, e.g., the use of photo-realistic rendering in the generation of synthetic images for training models for computer vision tasks such as depth estimation (Planche et al., 2017) and image segmentation (Richter et al., 2016; Saleh et al., 2018). Second, synthetically generated data can be post-processed to add randomness and variation, or to better match distributions that are expected in real-world application contexts (Tobin et al., 2017; Dunbar et al., 2018; Prakash et al., 2019; Kar

et al., 2019). Third, researchers have explored various methods for ensuring learned models are invariant to data perturbations and style variations, typically by utilizing ensembles of classification models that introduce a consistency loss in the objective function (Sajjadi et al., 2016; Laine and Aila, 2017).

In our approach, we utilize each of these previous domain adaptation methods, generating synthetic data that closely resembles the behaviors of human teams, augmenting this data so that it generalizes well to real-world data, and utilizing ensembles of classification models to improve robustness to data perturbations and style variations.

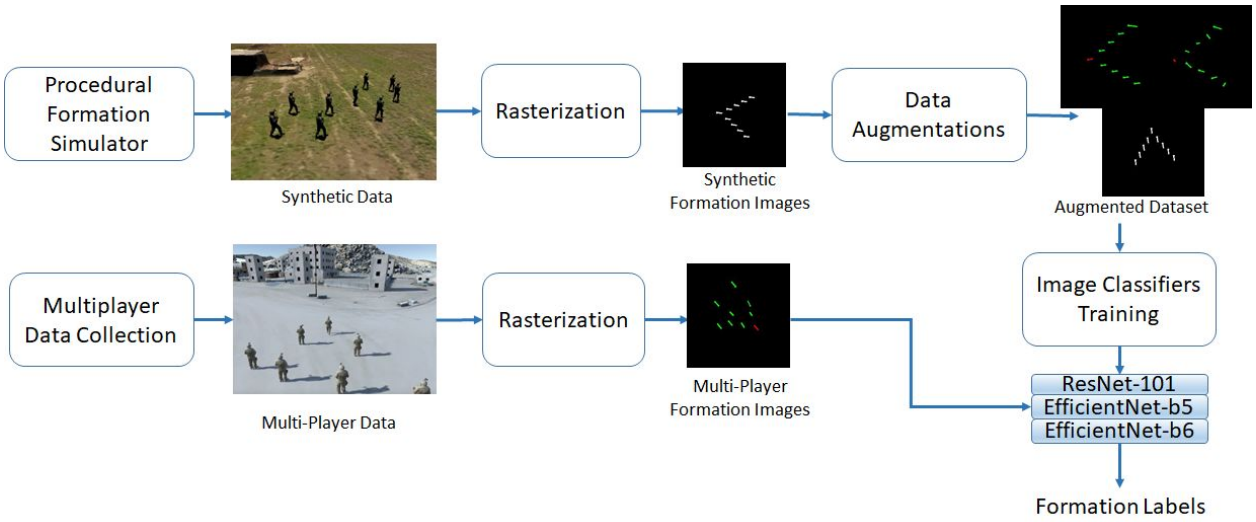

Figure 1. Overview of the behavior recognition process using synthetic training data.

OVERVIEW

Our approach to behavior recognition in virtual training simulations 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 virtual simulation environment by teams of AI-controlled agents, programmed to execute each of the behaviors that are to be recognized in the behavior of human trainees. 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 same virtual 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.

MULTIPLAYER TEST DATA

Our approach to the recognition of team behaviors in virtual simulation environments is to use synthetic training data instead of collecting and annotating data from human trainees. However, in order to evaluate the effectiveness of our approach, we collected and annotated a small amount of human performance data to use as gold-standard data in our experiments. In this section, we describe the virtual training simulation environment used in these experiments, and the specific group behaviors that we sought to recognize in our approach.

In this work, we utilized the Rapid Integration and Development Environment (RIDE), a virtual training testbed developed at the University of Southern California's Institute for Creative Technologies in support of research projects related to the U.S. Army's Synthetic Training Environment initiative. This software prototype consists of a set of libraries built on top of the Unity game engine that support the loading of terrain datasets from the U.S. Army's One World Terrain initiative, networked multiplayer control of virtual avatars, and various data collection and analysis tools. Using this platform, we devised a multiplayer environment that allowed networked participants to control a virtual infantryman, set in the fictional town of Razish that exists (in the real world) at the National Training Center at Ft. Irwin, CA. Using standard Xbox controllers, participants are able to navigate the urban area with a third-person view of their own avatar, directing their movements with the left thumb controller and directing their view with the right thumb controller (Figure 2).

Figure 2. The Rapid Integration and Development Environment (RIDE), showing perspective view for participant number 5 (left), and an overhead view of all nine participants maneuvering in squad column formation (right).

We enlisted the participation of nine volunteers from our research lab, forming a typical-sized infantry squad with one squad leader and two four-man fire teams. Each participant connected to the shared RIDE environment remotely (from home, during the Covid-19 pandemic), and communicated with each other via an external audio teleconferencing application. Only one participant (not the designated squad leader) had experience as a professional soldier, with real-world training on the maneuvers executed by this virtual squad.

In this data collection, we instructed this virtual squad to execute squad movements in formation, as described in a leadership textbook used in ROTC courses (U.S. Army, 2008). Three standard movement formations were executed (squad column, squad line, and squad file), along with two additional formations used in previous work (wedge and arrow). We furthermore noted a sixth movement class, representing durations in our dataset when the virtual squad moved out of formation, in an unstructured cluster.

To aid our participants in the correct execution of each maneuver, we assigned each participant a number (from 1 to 9), and a diagram indicating the position within each formation that they should maintain during movement (Figure 2). Furthermore, we asked that each participant log into the RIDE environment with a login name indicating their assigned number, which was visible to participants above the avatars in the environment. In each case, participant number 1 served in the "Point Man" role in the formation, e.g., the center position of the squad line, or the front position in squad file and squad column. During the data collection, this Point Man participant coordinated and directed the execution of each maneuver.

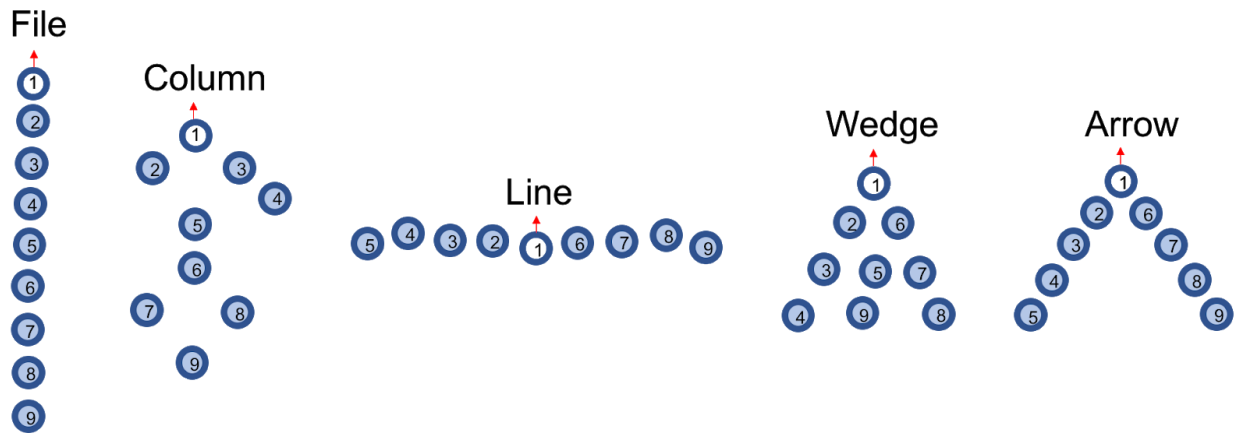

Figure 3. Five squad maneuver formations used in this research

During this collection phase, the location of each avatar in the virtual environment was sampled at 10 Hz. After the data collection phase, durations were manually labeled with the intended squad maneuver by aligning playback of the sampled data with an audio recording of the teleconference. The resulting dataset consisted of 20 executions of the five maneuvers depicted in Figure 3, and an additional 20 executions of squad movement out of formation before the start of each formation maneuver. In all, these 40 durations consisted of 14,173 labeled samples, used as the gold-standard test set in our experiments.

GENERATION OF SYNTHETIC TRAINING DATA

In this research, synthetic training data is used to enable behavior recognition in virtual training environments, allowing a supervised model to be trained without time-consuming data collection and annotation process by teams of experts. To generate synthetic training data for the recognition of movement formations, we utilize the same RIDE simulation environment described in the previous section, 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 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.

Procedural Unit Trajectory Generation

As a building block to create squad formation movements, we first implemented two types of movement behavior to automatically control each individual unit in the formation. The first one is the *fall-in* behavior, which is used to enforce a vector offset constraint between the unit and his leader. When the current offset between the unit and his leader deviates from the desired offset, the fall-in behavior automatically moves the unit to maintain that offset. The other one is the *follow* behavior, which ensures that the unit will move to follow the leader while keeping a distance constraint. It works by collecting the previous locations traversed by the leader, and moving the unit along that same trajectory if his distance from the leader is further than a threshold. Since there are obstacles in the environments that may slow down the unit trying to *fall-in* or *follow* the leader, both behaviors allow the unit to accelerate as needed to meet the constraint when the leader is moving away.

Using the aforementioned unit behaviors, we simulate the five squad maneuver formations including column, line, file, wedge, and arrow. Among them, column and line are standard army formation movements, and we defined the unit offsets based on the field operations manual. Wedge and arrow are synthetic formations and their offsets are procedurally defined. File is simulated using the *follow* behavior to ensure each unit follows its immediate fellow in the front. Given the formation parameters and a goal location for the leader unit, the formation movement 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. As shown in Figure 3, the leader is designated as unit 0 while the number within each circle indicates the index for other team units. Based on the specified formation type and desired spacing between units, the offset or distance for each unit is obtained to initialize the unit behavior. 2) given the initial unit offsets and current location for the leader, the new target location for each unit is calculated by using the aforementioned behaviors. 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. In addition, to model the movements when the squad is in scattered positions or is transitioning to different formations, we also defined an out-of-formation group behavior. It is generated by randomly selecting a unit to move to random locations periodically. This simulates the random sparse movements when the squad is switching to a different formation.

For the purpose of evaluating our approach, we generated about 160 seconds of synthetic training data for each of the five movement formation classes plus about 1000 seconds of data for the out-of-formation movements, 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 17482 labelled samples from a simulation session.

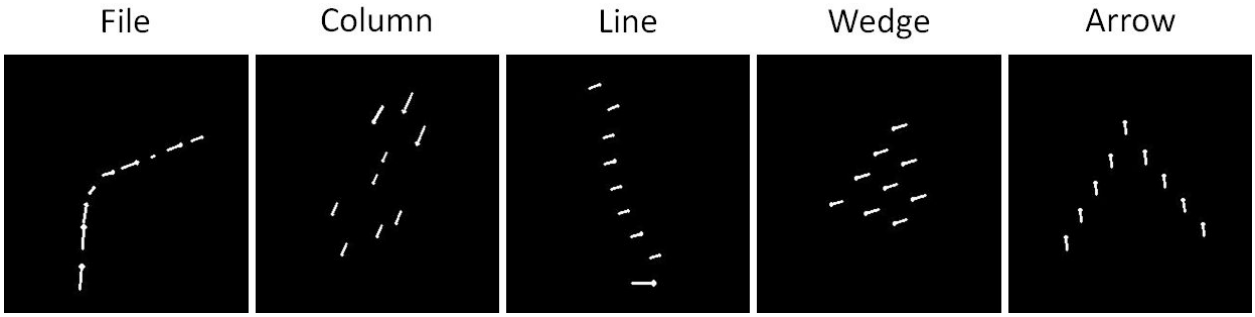

Figure 4. Examples of unit formation movements as rasterization representation. Different formation types result in different image features that are suitable for an image classifier.

Rasterization of Trajectories as Images

In its raw form, the aforementioned movement formation trajectories and labels are represented as a multivariate 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 numbers of units. Moreover, this reduces 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.

The rasterization representation is generated as follows. Given positions $x_1^t, x_2^t, x_3^t ... x_n^t$ at time step t, we first compute the $X-Z$ local coordinates with origin centered at $p_{avg}^t = \frac{1}{n}\sum_{i=1}^{n} p_i^t$, 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 pit to better encode the unit

movement velocity in a single image. Figure 4 shows some examples of rasterization representation for different formation movements.

This rasterization method was applied to each of the samples in the synthetic training data, yielding a dataset of 17482 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 14173 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.

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 virtual training environment. 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 dataset collected from human trainees. The main purpose of domain adaptation is to ensure that the model can generalize well to the multiplayer test data using various learning and augmentation techniques.

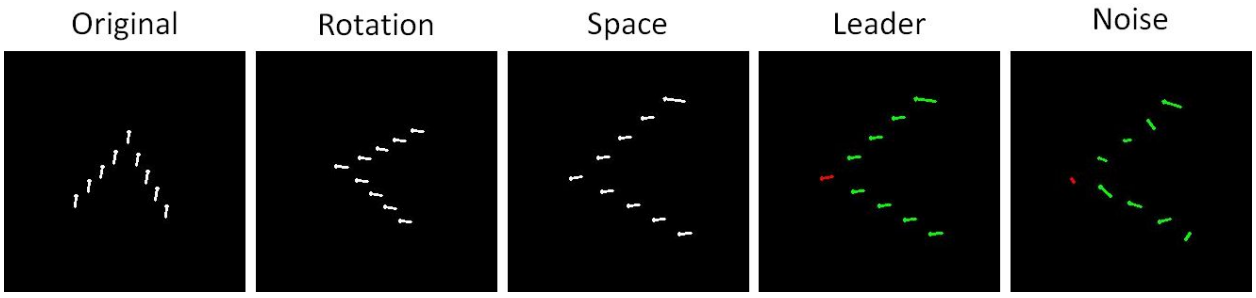

Figure 5. Examples of data augmentations for formation images. (From left to right) : original image, rotation variations, unit spacing variations, leader identification, and position perturbations

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 variations of the same data. It prevents overfitting 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 random color adjustments such as saturations or hues in the augmentation process. The types of augmentations used in this work include rotation variations, unit spacing variations, leader identification, 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.

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) Leader Identification: Since the squad leader is driving the formation movements, identifying him in the group helps the network to better learn the structure of input formations. It also helps resolve some ambiguities when the jittered unit positions make two formations look alike. To add such identification in the rasterized images, instead of rendering every unit in white color, the leader is rendered as red color while the other units are rendered in green colors in the image. Note that since a leader player was also selected during the data collection process to drive the formation movements, the same process is also applied for rasterization of testing data to identify the leader.

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 multiplayer data collected from the trainees more noisy than simulated data, where all units are coordinated programmatically when moving in formations. To alleviate this domain gap, a Gaussian noise is added to units' positions as well as their moving directions. This creates a more randomized movement trajectory for each unit that contains more variations than the purely procedural movements. Note that 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 test data.

Figure 5 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.

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 (He et al 2016) and EfficientNet (Tan et al 2019), are used to form the model ensemble. 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 the 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 reason we choose these two model architectures 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 either the two models of different complexity from the same architecture, or the models of similar complexity from different architectures. To fuse the results, we used equally-weighted averaging of predictions from these models to make final predictions.

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.

The computer hardware we used to conduct the experiment is an Intel 9820X CPU with 128 GB RAM and four nVidia 2080 Ti GPUs. All neural network models were trained within the PyTorch neural network framework for 50

epochs with batch size of 16 per GPU. The Adam optimizer is used for training with a learning rate of 0.0001 and cosine annealing schedule for learning weight decay. Accuracy scores were computed as the percent of correct behavior labels assigned to samples in the multiplayer test set.

Table 1. Evaluation of different data augmentations and model architectures.

Augmentation	Data Size	Network Architecture	# Params	Accuracy
None	17482	Resnet-50	25.6M	35.22%
Rotate	17482	Resnet-50	25.6M	49.54%
Rotate + Space	34132	Resnet-50	25.6M	58.50%
Rotate + Space + Leader	34132	Resnet-50	25.6M	72.15%
Rotate + Space + Leader + Noise	34132	Resnet-50	25.6M	74.70%
	34132	Resnet-101	44.5M	75.05%
	34132	EfficientNet-b5	30.0M	74.91%
	34132	EfficientNet-b6	43M	77.87%
Ensemble + All Augmentations	34132	Resnet-50 + Resnet-101	70.1M	75.83%
	34132	EfficientNet-b5 +EfficientNet-b6	74.5M	76.25%
	34132	EfficientNet-b5 + Resnet-50	55.6M	75.83%
	34132	EfficientNet-b6 +Resnet-101	87.5M	78.50%

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. The upper part of Table 1 shows the resulting classification accuracy of the ResNet-50 model on the test dataset with different augmentation methods. The model naively trained with synthetic data, without any augmentation, produced poor results with only about 35.22% accuracy, which is only marginally better than the expected accuracy of 16.67% when randomly selecting one of the six labels. 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 75% accuracy. Note that augmentations with varying unit spaces expand the number of training samples due to additional simulation sessions with different size and space parameters. 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.

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 the lower part of Table 1. 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 network in the ensemble except when combining two EfficientNet-b5 and b6 together, which produces a result worse than a single

EfficientNet-b6. One possible reason is that the two models have similar architectures and only differ in their complexity. Therefore it is less likely that adding them together will complement each other to produce better results. Empirically, this shows that combining different model architectures into an ensemble may be more effective since different architectures are likely to learn and extract different types of features. As a result, our best ensemble model, which combines EfficientNet-b6 and ResNet-101, is able to provide additional boost and achieve 78.50% accuracy on the multiplayer test dataset.

DISCUSSION AND FUTURE WORK

In this paper we describe an approach to the automatic recognition of team behaviors using neural networks, where the data used to train these networks are generated programmatically using fully autonomous agent teams. 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. Data augmentation techniques are used for domain adaptation, allowing the trained models to generalize to patterns seen in human performance data. 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 78.50% accuracy on a six-class classification problem with no authentic training data.

There are several next steps that must be investigated in order to validate this approach as a general method for behavior recognition in training simulations. Primarily, a broader range of trainee behaviors should be studied, particularly those that may push the limits of pattern recognition technology or our abilities to generate human-like synthetic training data. This includes long duration behaviors, such as convoys and deliberate ambushes, which may require many minutes of observation before an accurate classification could be made. Similarly, complex multi-part behaviors, such as patrols and maneuver by bounding overwatch, may require more sophisticated network topologies in the highest layers in order to generalize over temporal sequences. Additionally, interactions between groups, including engagements with enemy forces, will require additional mechanisms for identifying and representing the behavior of other groups as input to the classification task.

As these challenges are overcome, we expect that this approach to team behavior classification will enable more responsive forms of guided, deliberate practice for team training in virtual environments. While the authoring of programmatic behaviors for teams of autonomous agents is a requirement of our approach, the associated costs are amortized over other features that they enable, including the possibility of training with mixed human and autonomous teams, and the use of these behaviors for constructive simulation and course of action analysis. We envision that every available autonomous behavior could also be used to generate synthetic training data for behavior recognition, allowing for a one-to-one correspondence between what autonomous teams can do in the environment and what the environment can recognize in the behavior of human teams. In so much as these behaviors are authored prescriptively to align with doctrine, virtual training environments will know "what right looks like," as a basis for responsive training, and as a baseline starting point for adaptive behavior and innovation in team performance.

ACKNOWLEDGMENTS

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.

REFERENCES

S. Ahmad, A. Bryant, E. Kleinman, Z. Teng, T.-H. Nguyen, and M. El- Nasr, "Modeling individual and team behavior through spatio-temporal analysis," 10 2019, pp. 601–612.

G. Sukthankar and K. 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, ser.

AAMAS '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 638–645. [Online]. Available: https://doi.org/10.1145/1160633.1160746

W. Min, B. W. Mott, J. P. Rowe, R. G. Taylor, E. N. Wiebe, K. E. Boyer, and J. C. Lester, "Multimodal goal recognition in open-world digital games," in Proceedings of the Thirteenth Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2017.

E. Ha, J. Rowe, B. Mott, and J. Lester, "Recognizing player goals in open-ended digital games with markov logic networks," Plan, Activity, and Intent Recognition: Theory and Practice, pp. 289–311, 03 2014.

B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, O. Lehmann, T. Chen, A. Hutter, S. Zakharov, H. Kosch 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.

S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in European conference on computer vision. Springer, 2016, pp. 102–118.

F. S. Saleh, M. S. Aliakbarian, M. Salzmann, L. Petersson, and J. M. Alvarez, "Effective use of synthetic data for urban scene semantic segmentation," in European Conference on Computer Vision. Springer, 2018, pp. 86–103.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "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.

A. Dundar, M. Y. Liu, T. C. Wang, J. Zedlewski, and J. Kautz, "Domain stylization: A strong, simple baseline for synthetic to real image domain adaptation," arXiv preprint arXiv:1807.09384, 2018.

A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield, "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.

A. Kar, A. Prakash, M.-Y. Liu, E. Cameracci, J. Yuan, M. Rusiniak, D. Acuna, A. Torralba, and S. Fidler, "Meta-sim: Learning to generate synthetic datasets," in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 4551–4560.

M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in Advances in neural information processing systems, 2016, pp. 1163–1171.

S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning." in ICLR (Poster). OpenReview.net, 2017. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2017.htmlLaineA17

K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in International Conference on Machine Learning, 2019, pp. 6105–6114.