Machine Learning Applications for AI in Game Development
Evan Fridge [Drury University] & Marc Aubanel [LSU]

ABSTRACT
Neural Networks (NN), like the biological origin they derive from, handle new data organically, able to process any range of inputs and return favorable results when properly trained. This project seeks to build a NN library inside the Unreal Engine platform for exploring how machine learning can enhance games.

Traditional AI approaches in game development have a weakness growing apparent as games become more and more complex: these approaches are rigid, closed systems unable to handle new data sources without great effort by the developer. Adding a new decision to a tree in a AAA environment may require more time spent adjusting than simply starting over. Machine learning may offer a solution to these problems.

OBJECTIVES
1. Create a simple C++ neural network library outside of Unreal Engine for testing purposes.
2. Find usable data sets and build predictive, memory-efficient neural network models.
3. Integrate the feed-forward component into Unreal Engine and import satisfactory models.
4. Build a data pipeline within Unreal Engine for the networks to access needed parameters.
5. Add in-game behavior responding to NN output.

METHODS
Coding the library: prioritized reading in data sets as CSV files. The weights and normalizer values of any satisfactory model are recorded and saved in separate files, which can then be loaded into Unreal.

Organizing data: two data structures handle the flow of data in our system, the Object Field and Basic Protocol. Object fields instantiate up to 25 objects in the level while automatically connecting to data sources. Basic Protocol creates any NN model desired, as well as any number of them.

RESULTS
Basic Protocol now generates eleven different models inside of the Unreal Engine: the fire spread predictor, two movement predictors, and eight behavior profiles.

Object fields now tie into a day/night cycle responsible for weather, but can also link into a hierarchical system of other fields for creating mass/group behaviors.

Behavior profiles return arbitrary values of 1-5, allowing for any form of traditional AI (decision trees, finite state machines, etc.) to execute predicted behavior.

CONCLUSIONS
Combining object fields with basic protocol allows for scalable, realistic behavior for both individual entities and crowd entities. It also holds promise for generating weather events given a proper data set. Basic Protocol offers a context-sensitive source of random number generation, a direct advantage over the traditional approach at a low performance cost.

By pairing an in-game character (or characters) with a field consisting of Cartesian locations, its given protocol allows for movement at and between these locations. With an added layer of traditional AI, complex behaviors can be built to correspond to the abstract behaviors generated by protocol.

There is potential for nested behavior profiles, where the result of one behavior profile causes a corresponding profile to activate. A pack of wolves, for example, may have its overall behavior directed by one profile, but the individual wolves may have individual profiles to direct individual actions. This creates an identity for the wolves with a relatively small amount of resources needed from the developer.

REFERENCES


ACKNOWLEDGEMENTS
A special thanks to Epic Games, Inc., creators of the Unreal Engine.

In-Game Assets provided by Epic Games, Alexander Dracott, and Protoolfactor Inc.

Thanks to David Miller for help implementing our initial neural networks.

This work was supported by the National Science Foundation (NSF) award #ACI-1560410.