Discerning and Completing Objectives with an Artificial Neural Network in a Semi-Stochastic Environment

Patrick Abiney
Mechanical Engineering Department
Northwestern University
Evanston, IL 60208
patrickabiney2017@u.northwestern.edu

Abstract - Multilayer perceptrons and their many variants have proven capable of dividing many nonlinear decision surfaces. Images, as well as video, are able to be classified using specialized multilayer perceptrons. By using keystroke probabilities as a label and the incoming image as the key, one could theoretically construct an artificial neural network capable of playing a video game. With the proper layer and hyperparameter configuration, one could theoretically overcome complex issues for artificial intelligences such as juggling multiple objectives, completing objectives in a logical order, and traversing a complex labyrinth that both changes over time as well as requires backtracking.

Keywords – TensorFlow; TFlearn; OpenCV; SURF; VBA-M; Northwestern University; Robotics; Python; Pokémon;

I. Intro

Image, and in particular video, recognition is a topic that has many schools of thought and no one-size fits all solution. There exist computer vision algorithms that utilize mathematical properties of images to manipulate pixels and find meaning within them. However, each algorithm is heavily customized to solve a specific problem rather than provide an arbitrary interface that one can simply feed an image into and output a desired result.

A. Interest in Topic

Convolutional Neural Networks have been proven to play games in the past. However, a known problem is having a neural network handle a multi-objective situation like those in role play games. One of the cited problem with these situations is that traditional Convolutional layers do not have a concept of time or memory.

B. Proposed Work

In order to construct a custom neural network to discern and complete objectives for a role playing game, a loop will need to be in place in which the game's screen is captured, processed by the network, and then keystrokes are propagated to the video game.

Samples are required in order to train the neural network. A sample gathering application that launches the game emulator, records keystrokes and images in order, and outputs the results.

C. Initial Feasibility

One may assume a solution to this problem would be to use a Multi-Layer Perceptron as these have proven to solve for image recognition in the past, though some better solutions do exist, but many variants exist, each with several hyperparameters. One cannot find a one size fits all solution for this problem as neural networks need to be, just like other forms of visual computation, heavily customized.

One of many variants of the Multi-Layer perceptron, Convolutional Neural Networks pose a surprising threat to image recognition as they take into account not only the pixels value but also the distance to like pixels and are built in a fashion that resembles the visual cortex. Still, there is no concept of time that is required for video.

Long Short-Term Memory layers take advantage of time to apply input degrade, output degrade, and forget rates.

II. Work

A. Deep Neural Network (DNN) Structure

Patrick Abiney
Northwestern University Department of Mechanical Engineering

Multilayer perceptrons are capable of solving non-linear decision surfaces, though not always well. Taking multiple multilayer perceptrons and layering them creates deep neural networks. There are still many nuances and no one-size fits all solution. For this project a combination of Convolutional Neural Network layers, Long Short-Term Memory Layers, and fully connected layers to process images and time to produce keystroke probabilities as outputs.

1) DNN Block Diagram

The block diagram in figure 1 depicts the flow from input to output. There will be a 3 dimensional image input. The three dimensions of the input represent color channels, image height, and image width. Time will be tracked within the Long Short-Term Memory layer. The two layers are connected with a fully connected layer to bridge the features map output from the second Convolutional Layer's ReLU layer to the inputs of a Long-Short Term Layer. From the Long Short-Term Memory Layer's output we will use another fully connected layer to output only keystroke probabilities for eight key values corresponding to each button on the original Game Boy.

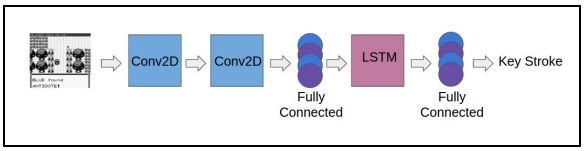

Figure 1, DNN Layers Block Diagram

2) Fully Connected Layers

A fully connected layer is one that most closely resembles a traditional Multi-Layer Perceptron. Many units exist, all connected to one another within the layer. Because every node is connected within a Fully Connected Layer, there is no concept of distance between nodes in this type of Artificial Neural Network. The interconnectivity within the layer can be seen in the Fully Connected Block Diagram, figure 2.

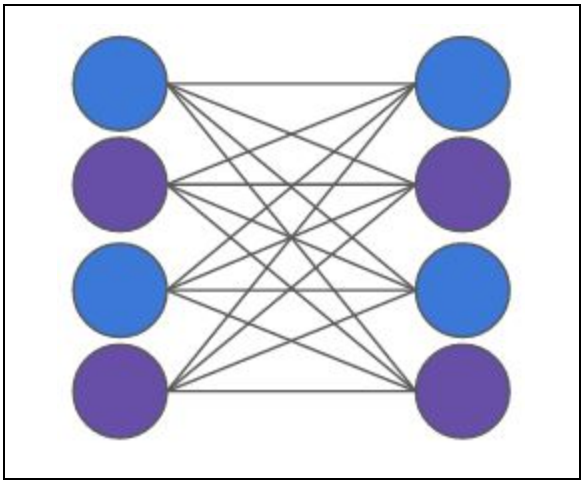
Figure 2, Fully Connected Layer Block Diagram

3) Convolutional Layers

Convolutional Neural Network layers are a variant of a multilayer perceptron. A key difference from a Convolutional Neural Network layer and a Fully Connected Neural Network layer is that a COnvolutional Neural network does have a concept of nodal distance.

Additionally Convolutional layers are designed to act similarly to that of the mammalian visual cortex.This is not to say that they are in any way an actual representation of the mammalian visual cortex, but rather a complex statistical function that simply finds a probable outcome in a complex decision space. The mammalian brain is far more complex and stochastic than any artificial neural network model.

The layer takes in a three dimensional input in theory (channels, width, and height), and outputs a three dimensional features map. In figure 3, a block diagram representing a common Convolutional Neural Layer can be seen. A filter samples across the entire height and width of the image and applies a dot product between the filter and the input.[8]

Patrick Abiney
Northwestern University Department of Mechanical Engineering

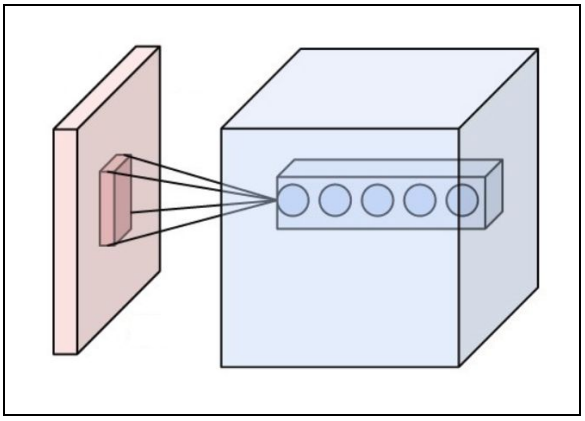

Figure 3, Convolutional Layer Block Diagram[8]

4) ReLU Activation

A Rectified Linear Unit is a layer of neurons that apply non-saturating half-wave rectification through a mathematical function, depicted in equation 1.[7]

$$f(X) = max(0,\ X) \qquad (1)^{[7]}$$

In equation 1, X represents the input to the neuronal layer.[7] A ReLU function will increase the nonlinear properties of both the network layer and the network in entirety without interfering with the input fields of a given convolutional layer.[8] There are other activation functions that may prove more favorable depending on the application at hand. For Convolutional Neural Networks, Rectified Linear Units are prefered for their speed and preservation of accuracy in generalistic predictions.[8]

5) Long Short-Term Memory Layers

A long short-term memory layer is a variant of a normal Recurrent Neural Network Layer that remembers values for short or long periods of time well.[10] The difference between the two, LSTM compared to RNN, is that a long short-term memory layer adds in memory components including forget gates, input gates, and output gates.

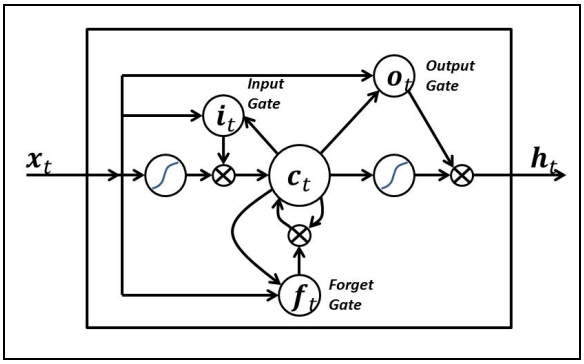

Figure 5, Long Short-Term Memory Layer Block Diagram[9]

Figure 5 depicts the functional logic of a long short-term memory layer. The relationship between the input, output, and forget gates. X_t represents the neuronal input of the layer, which is fed into all three gates and a sigmoidal. The output of the sigmoidal function is cross multiplied with the output from the input gate. The resulting product is then fed into the cell state vector along with the cross product of the output from the forget gate and the output from cell state vector. The output of the cell state vector is fed into every gate, fed into a cross product function with the output of the forget gait, and fed into a sigmoidal function. The result of this sigmoidal function is crossed with the output of the output gate to produce the output of the layer.

A. Training

Training the neural network is the most time consuming aspect the project besides the programming itself. One can gather samples of people playing Pokémon in the form of images and keystrokes. Using reinforcement learning, these samples can be used to train the artificial neural network.

1) Sample Gathering

In order to train the artificial neural network samples needed to be gathered from playthroughs of pokemon. An application was constructed in python to launch the VBA-M emulator, record every key pressed, and record the corresponding images.

2) Experimentation

Within the neural network itself there are many parameters that when altered change the functionality and performance of the neural network. Traditionally one would create a small sample and validation set and then test each combination of parameters while recording their performance for comparison later.

Other experimentation in neural networks can involve variations in layer structure rather than configuration. In the experiments carried out in this project, the neural network structure was determined before experimentation after conducting research.

B. Gameplay

A videogame emulator interfaced with an artificial neural network will provide a more stable loop than using a physical game system both when capturing and when propagating commands and will be used for this experiment.

1) Gameplay Block Diagram

There are fundamentally two halves to the gameplay loop, as seen in figure 7. In the game emulator half runs Pokémon Blue Version in English and is represented in the diagram by the green block. This process will run at the top of the desktop so it is in focus and captures any keystrokes generated. The other half of the diagram contains all of the logical framework to capture and process images, followed by sending keystrokes. pyscreenshot is used to capture a section of the screen. OpenCV is used to process the image from pyscreenshot before sending it to the artificial neural network. The artificial neural network will process the images and output keystroke probabilities. Finally PyKeyboard will find which key, if any, was most likely pressed and generates that command.

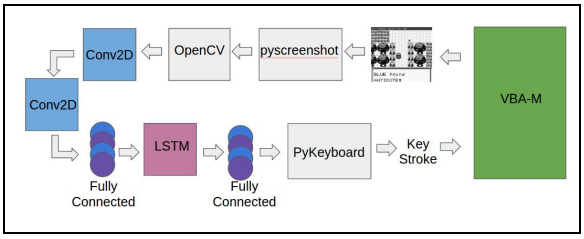

Figure 7, Gameplay Block Diagram

2) Performance

The current state of performance is less than desirable. Though able to predict the correct keystroke 70% of the time with image sets for training and validation, while in an evaluation loop the artificial neural network produces the same values repeatedly. Future work could be done to fix this problem like adding SURF or SIFT, shrinking image sizes, restructuring the neural network, and even upgrading hardware.

III. Dependencies

In the course of development several software dependencies were chosen to make the implementation more stable, syntactically pleasant, and functional.

A. VBA-M

"VBA-M is a fork from the now inactive VisualBoy Advance project, with goals to improve the compatibility and features of the emulator…"[3]

B. TensorFlow™

"TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API."[1]

C. TFlearn

"TFlearn is a modular and transparent deep learning library built on top of Tensorflow. It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it."[2]

D. OpenCV

"OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library."[4]

"The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc."[4]

E. PyKeyboard

"A simple, cross-platform python module for providing keyboard control."[6]

F. pyscreeenshot

"The pyscreenshot module can be used to copy the contents of the screen to a PIL or Pillow image memory. Replacement for the ImageGrab Module, which works on Windows only. For handling image memory (e.g. saving to file, converting,..) please read PIL or Pillow documentation."[5]

IV. Pitfalls

Over the course of development several pitfalls were encountered that delayed the project. The operating system itself initially seemed to be an issue as the VBA-M source on GitHub required the latest version of Ubuntu. After a new environment we set up work was able to continue until the time came to train large datasets of several thousand images. The current development machine is incapable of handling so many images and as a result crashes the application. Smaller data sets are able to execute but not enough information is learned from the images to successfully play the game.

V. Future Work

A. SURF

Speeded up robust features is a patented algorithm that is able to find key features within an image. This could be a possible solution for the current live loop not detecting key features. This algorithm is in OpenCV but not the standard library because the algorithm is patented.

"To detect interest points, SURF uses an integer approximation of the determinant of Hessian blob detector, which can be computed with 3 integer operations using a precomputed integral image. Its feature descriptor is based on the sum of the Haar wavelet response around the point of interest."[11]

B. Possible Applications

Future applications for artificial intelligences upon success of this project range from testing bots for game level designers as well as enemies and companions for players in games. If one were to apply the same concepts to things like asset creation software or social media, one may have the groundwork for some powerful and helpful bots to help contribute to artist pools and social media advertisers.

VI. Conclusion

The issues involved with objective recognition in role play games may not have been solved by this project in the current state. However research has been done to uncover possible solutions.

References

[1] About TensorFlow. Retrieved March 19, 2017, from https://www.tensorflow.org/.

[2] Damien, A. TFLearn: Deep learning library featuring a higher-level API for TensorFlow. Retrieved March 19, 2017, from http://tflearn.org/.

[3] VisualBoy Advance - M - About VBA-M. Retrieved March 19, 2017, from http://vba-m.com/about.html.

[4] ABOUT. Retrieved March 19, 2017, from http://opencv.org/about.html.

[5] P. (2017, February 27). Pyscreenshot. Retrieved March 19, 2017, from https://github.com/ponty/pyscreenshot.

[6] S. (2013, December 07). PyKeyboard. Retrieved March 19, 2017, from https://github.com/SavinaRoja/PyKeyboard.

[7] Rectifier (neural networks). (2017, March 18). Retrieved March 19, 2017, from https://en.wikipedia.org/wiki/Rectifier_(neural_networks).

[8] Convolutional neural network. (2017, March 16). Retrieved March 19, 2017, from https://en.wikipedia.org/wiki/Convolutional_neural_network.

[9] Klaus Greff; Rupesh Kumar Srivastava; Jan Koutník; Bas R. Steunebrink; Jürgen Schmidhuber (2015). "LSTM: A Search Space Odyssey".

[10] Long short-term memory. (2017, March 17). Retrieved March 19, 2017, from https://en.wikipedia.org/wiki/Long_short-term_memory

[11] Speeded up robust features. (2017, February 26). Retrieved March 20, 2017, from https://en.wikipedia.org/wiki/Speeded_up_robust_features