Brian Lai

Mr. Kosek

Computer Systems Research Lab

May 9, 2023

Video Classification of Mario Kart Wii Chain Wheelies Using Machine Learning

https://github.com/BLCRAFT210/MKW-Chain-Classifier

Abstract

Mario Kart Wii time trials include a notoriously precise technique known as a "chain wheelie", and it is hard for players to visually distinguish the type of chain wheelie they performed. In order to help players classify chain wheelies during gameplay, I developed a 3D convolutional neural network using TensorFlow to classify the different types of chain wheelies found in a video of a Luigi Circuit time trial. The network was trained and tested using video data collected as Dolphin Emulator replayed ghosts obtained from the CTGP Ghost Database, and it achieved a testing accuracy of 99%.

Introduction

In Mario Kart Wii time trials, players can increase their vehicle's speed by up to 15% by performing a wheelie. However, wheelies will drop after 180 frames, requiring the player to perform another wheelie in a technique known as a "chain wheelie" ("chain" for short). Chains are necessary for time trials on many tracks, and depending on their execution, they can cause a substantial amount of time loss. As such, chaining is a vital skill for high level Mario Kart Wii time trials.

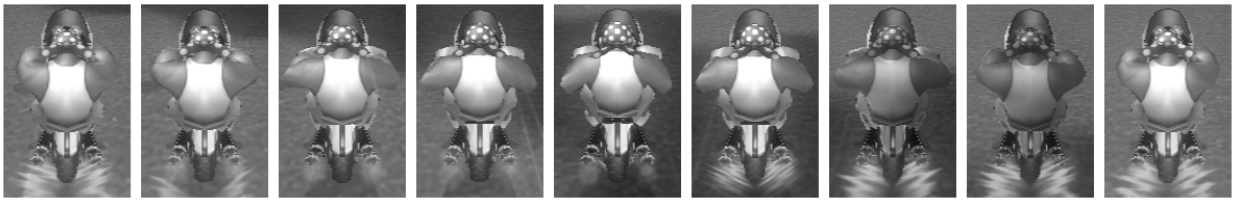

Figure 1. An example of a full chain. These frames were taken once every three frames in-game.

Players have coined names for the different kinds of chains: a wheelie performed one frame after the previous wheelie dropped is called a “full chain”, two frames a “half chain”, three frames a “weak chain”, and four frames and above a “drop”. A chain retains the most speed, while a drop retains the least. It is often important to know in time trials how fast the current trial is, and chains can substantially affect the pace of the run and be the make-or-break factor when trying to get new records. However, even seasoned players may have difficulty distinguishing between the different chain types due to the subtlety of their visual differences.

I was interested in seeing if machine learning models could better tell the difference between chains, since it could be the starting point of a program that could help speedrunners in real-time by informing them of how good their chains are. Therefore, I set out to create a model that can classify the different types of chain wheelies.

Background

Convolutional Neural Networks

Machine learning is a subfield of artificial intelligence that focuses on using data to find patterns and make predictions. One common machine learning model is a neural network, which is based on the structure of a brain. Neural networks are composed of many layers of connected “neurons”, or nodes, which perform basic calculations and transmit information to each other.

With large amounts of training data, neural networks can learn to adjust their internal parameters to make accurate predictions with new data. Using the right neural network structure and enough training data, neural networks can perform tasks more accurately and faster than humans can, especially with complex or large datasets.

A convolutional neural network (CNN) is a type of neural network usually designed to work with multi-dimensional input. Traditional neural networks treat the entire input as a single dimensional input vector, while CNNs preserve the original shape of the input (Walter, 2023). This makes CNNs particularly useful for a variety of 2D image classification tasks, since they can recognize and extract certain features from images (Clark, 2022). One example use of a CNN is classifying human facial expressions, which Gan (2018) accomplished with an accuracy of 0.6424.

3D CNNs

3D CNNs are often used to analyze spatiotemporal data, such as videos. 3D CNNs can be especially powerful and fast tools for video classification, as Tran et al. (2018) demonstrated by using a (2 + 1)D convolution that yielded results comparable to other top-end models on multiple video datasets.

Dolphin Emulator

A console emulator is a piece of software that allows a computer to act like a video game console and play games released on that console. The most popular emulator for Wii games is the open-source Dolphin emulator, which is available on Windows, Linux, macOS, and Android. Dolphin includes a plethora of features that make it an instrumental resource for gamers, modders, and software developers.

For this project, I needed Dolphin to take commands from code, so I used a branch of Dolphin that supports running Python scripts (Felk). Among other things, this branch allows Python code to wait for Dolphin to advance the game by a frame, set controller inputs, save and load savestates, read console memory, and get screenshots.

CTGP Ghost Database

CTGP Revolution is a widely-used unofficial mod for Mario Kart Wii with over 70,000 total users (Usage Statistics). One of its features is a Ghost Database, which contains over four million ghosts, or time trial runs, on the 32 original Mario Kart Wii tracks and thousands of custom community-made tracks. Every time a player finishes a run and saves a ghost using CTGP, the ghost is automatically uploaded to the database. CTGP is practically required for high-level time trials, due to its anti-cheat mechanisms, so almost every top-level run since around 2017 is available on the Ghost Database.

The Ghost Database includes a JSON API, allowing software to access leaderboards and download ghosts. This can be used to download a set of ghosts that overall contains many instances of the four classes of chains.

Development

Objective

I decided to train the model with chains from only one track instead of all 32 in Mario Kart Wii in order to simplify the problem. I chose Luigi Circuit, the first track in the game, since its chains occur in the same spots on the track every lap.

The success of the project is measured by the accuracy of the model. Due to the unique subject of this project, there is not an established level of accuracy achieved either by humans or other models, so the goal was to achieve an accuracy indicating that the model performed better

than a random classifier. This entails achieving an accuracy reasonably above 25%. Such an accuracy only arises if the model "learns" from the training data to make informed predictions rather than guesses.

Methodology

Setting up the Dolphin environment

I started by building the Dolphin scripting branch using clang, since g++ gave errors. Next, I dumped a PAL (European) ISO, or image file, of Mario Kart Wii using my Wii so Dolphin could run it. I then changed a few Dolphin settings: Immediately Present XFB to reduce graphical latency, Dump at Internal Resolution so that dumped frames had constant dimensions, and Speed Limit = 300% so that Dolphin would run at three times normal speed.

Downloading ghosts

I wrote and ran a script in order to download the top 1000 ghosts on Luigi Circuit from the CTGP Ghost Database's API. The ghosts, named by their hash value, were stored in a folder called ghosts, and info about the ghosts was stored in a JSON. All of the ghosts used Funky Kong as the character and Spear as the vehicle, since that is the optimal combination for time trials on Luigi Circuit.

Generating training data

In order to provide a starting point for replaying ghosts, I wrote a script which waits until a memory flag in Mario Kart Wii indicates that a race countdown timer has started. At that moment, the script saves the current state to startstate.sav. To get to that point, I ran Mario Kart Wii while Dolphin was running this script, selected Funky Kong as the character, Spear as the vehicle, and Luigi Circuit as the course, and waited for the time trial to start.

Next, I wrote another script in order to loop through all of my ghosts to replay them and generate training data for my model. All of the ghosts were compressed by a proprietary algorithm created by Nintendo called YAZ1, so I used a Python package called libyaz0 and used its decompress function to interpret ghost information. From the decompressed data, the program reconstructs the sequence of controller inputs stored in the ghost. The script then loads the starting savestate and replays this sequence of inputs; since Mario Kart Wii time trials are deterministic, the replayed run will exactly match the original run that the ghost represents. Each frame of Dolphin's graphical output is saved as a grayscaled PNG, and the script also records data about when wheelies occurred in the run. After the run, the script finds and labels instances of chains and saves them to a training data folder as 25 consecutive frames.

I thought about ways to extract features from the data, such as image differencing using ImageMagick, thinking that this would speed up training and reduce the size of my data. However, I ran some image differencing and it did not look useful to me, so I decided to collect image data instead and let the CNN extract features itself.

Limitations

At the time I generated training data, the Dolphin scripting branch was highly susceptible to crashes and freezes. On average, while running the ghost replay script, Dolphin would crash around once every minute or so, and it would freeze every ten minutes or so. Therefore, I wrote a bash script in order to automatically restart Dolphin whenever it crashes. It also restarts Dolphin if it has not crashed in the last 120 seconds, since it would only last that long if it froze.

Some ghosts would cause problems upon replay. Sometimes, YAZ1 decompression would fail; other times, the ghost would deviate from its actual path in the middle of the run. I ended up removing 11 ghosts from the dataset due to these errors.

Training the model

After running through all of the ghosts, I had obtained 2527 instances of full chains, 1686 instances of half chains, 431 instances of weak chains, and 279 instances of drops. This skew towards better chains is due to the high skill level found in top 1000 runs.

I used an 80-10-10 training-validation-testing split and trained and tested on a dataset of 1000 chains: 250 full chains, 250 half chains, 250 weak chains, and 250 drops.

I used a TensorFlow tutorial that is based on Tran et al.'s paper (Video classification with a 3D convolutional neural network) to create my model. The model takes in a sequence of 25 frames and outputs a probability distribution of the four possible chain types. The model stops training if its loss does not improve by at least 0.0005 within any sequence of 10 epochs.

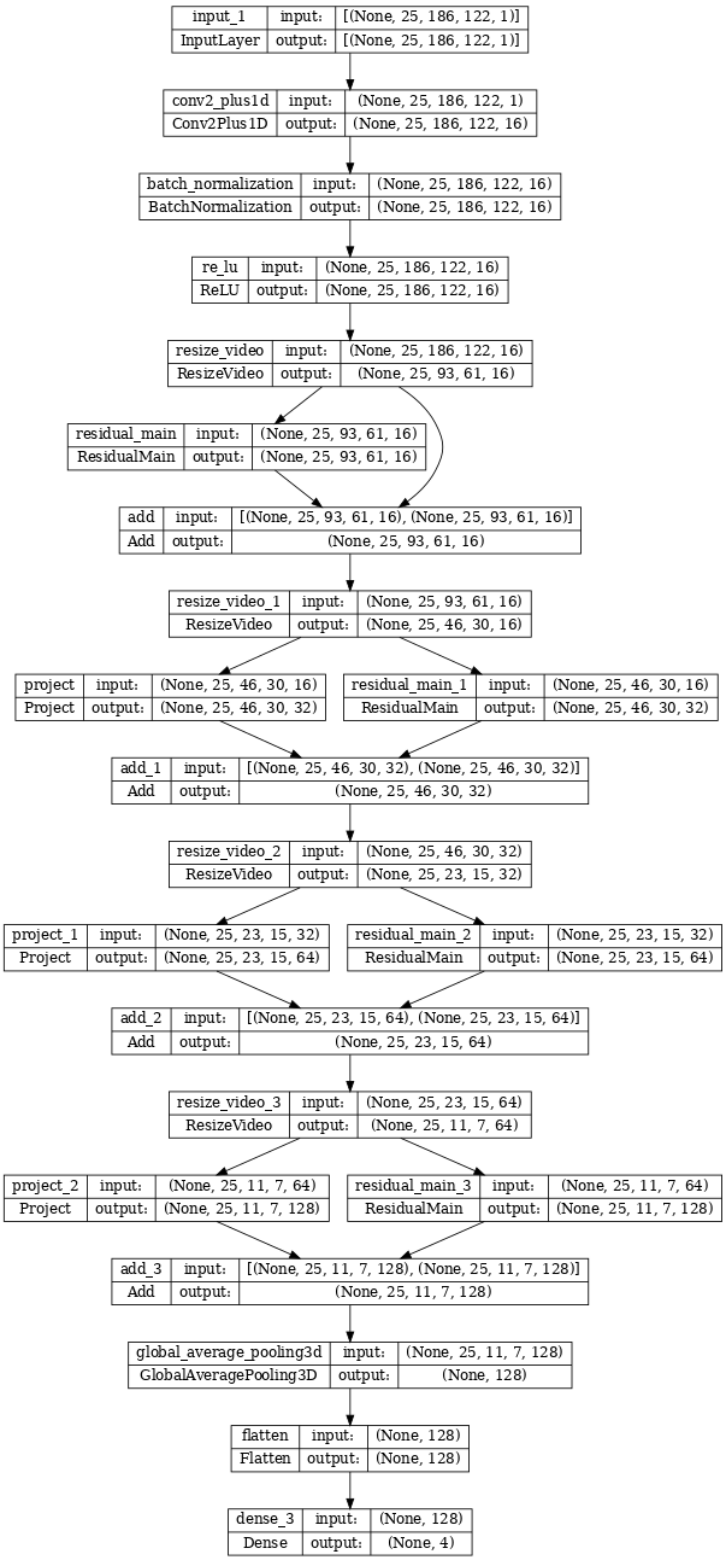

Figure 2. Keras model plot.

Results

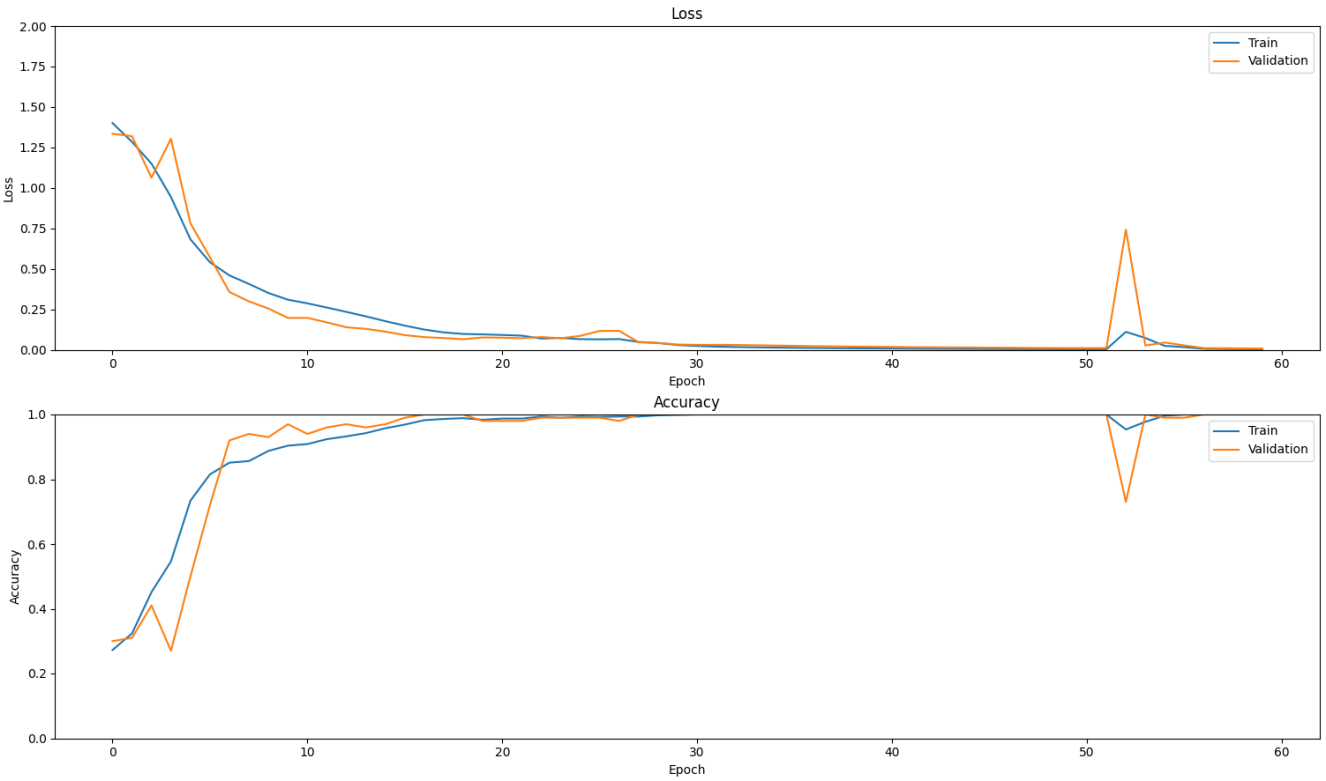

Figure 3. Training loss and accuracy over epochs.

The model was able to converge quite rapidly, as the training and validation accuracy had already gone above 0.9 by the 10th epoch. The model stopped after 60 epochs with an accuracy of 0.99 and a loss of 0.0406 on the test set. This indicates that the model is highly effective at classifying chains, and it is able to make correct predictions 99% of the time.

I did not have time to adjust the model's architecture or hyperparameters to fit this problem, but these results proved that was not necessary.

Conclusion

The model's high testing accuracy of 99% far exceeded the original objective to surpass 25% accuracy, and it proves that 3D CNNs can distinguish subtle variations in video data highly effectively. This project's approach can be extended to other tracks in the game to create a model for each track, a model for each character and vehicle combination used in competitive time trials, or even a model for any chain encounterable in Mario Kart Wii.

With extra development, such models could be used to assist players in real-time. Currently, this model does not predict quickly enough for real-time results, but one promising possibility is MoViNets, which can operate on streaming video (Kondratyuk, 2021). By using a device called a capture card, players could feed streaming video into a MoViNet, which could classify chains as they occur during a run and provide live auditory or visual feedback to the player.

Further research could cover additional applications of 3D CNNs to other video games or other video classification tasks with minute spatiotemporal differences between classes.

References

Clark, J. (2022). Behavioral Analysis of Deep Convolutional Neural Networks for Image Classification. *Florida Atlantic University ProQuest Dissertations Publishing*. https://www.proquest.com/dissertations-theses/behavioral-analysis-deep-convolutional-neural/docview/2672699047/se-2

Felk. (2023). Felk/Dolphin. GitHub. https://github.com/Felk/dolphin

Gan, Y. (2018). Facial Expression Recognition Using Convolutional Neural Network. *ICVISP 2018: Proceedings of the 2nd International Conference on Vision, Image and Signal Processing*. https://doi.org/10.1145/3271553.3271584

Kondratyuk, D., Yuan, L., Li, Y., Zhang, L., Tan, M., Brown, M., & Gong, B. (2021). MoViNets: Mobile Video Networks for Efficient Video Recognition. https://doi.org/10.48550/arXiv.2103.11511

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., & Paluri, M. (2018). A Closer Look at Spatiotemporal Convolutions for Action Recognition. https://doi.org/10.48550/arXiv.1711.11248

Usage Statistics. Usage Statistics - CTGP Revolution. (2023). https://www.chadsoft.co.uk/download/statistics.html

Video classification with a 3D convolutional neural network. TensorFlow. (2022, December 15). https://www.tensorflow.org/tutorials/video/video_classification

Walter, B. (2023). Analysis of convolutional neural network image classifiers in a hierarchical max-pooling model with additional local pooling. *Journal of Statistical Planning and Inference, 224*. https://doi.org/10.1016/j.jspi.2022.11.001