Interactive Game Development Using Machine Learning Models

Aarian Ahsan Abiola Alalade Kayla Baker Aaron Jacob Naveen Mukkatt

Introduction

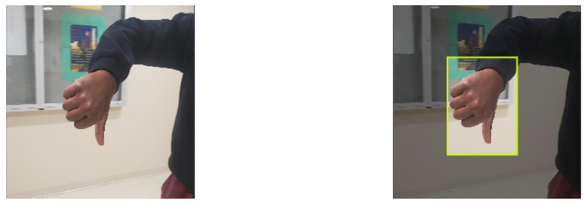
Interacting with computer games typically involves pressing keys on a keyboard. However, this method of playing a game may not be accessible to populations who suffer from conditions such as repetitive strain injury (RSI) and other hand disabilities that can affect everyday keyboard usage. Using hand gestures instead to control games allows these populations to interact with them in a way that is more accessible and natural than using a keyboard.

We show the viability of this concept through the practice of transfer learning by training a pre-trained deep neural network with a custom dataset of hand gestures in order to translate these gestures into game controls for a snake game.

Methods

Developing the gesture-controlled snake game using the computer vision model consisted of two steps:

1. Creating the dataset. We took approximately 1,200 images of directional hand gestures in multiple backgrounds, angles, and skin tones showing the "thumbs up", "thumbs down", "thumbs left", and "thumbs right" directions (classes) (about 300 images per class) with both left and right hands. We then annotated each image with the appropriate direction for training an object detection model. We randomly split the dataset into three subsets: 70 percent of the images were used for training the model, 20 percent were used for validation, and 10 percent were reserved for testing data.



(a) An image with no predictions.

(b) A yellow ("down") bounding-box prediction made by our trained model.

Figure 1. A "down" hand gesture image from the dataset.

2. Training the model. We used the annotated dataset to train multiple instances of the pre-trained SSD MobileNetv2 object detection model with the TensorFlow Object Detection API using various batch sizes and step counts to determine the best-performing model instance to implement in the snake game. We also trained a YOLOv7 object detection model with a batch size of 16 for direct comparison with MobileNetv2. We then tested these models with our test subset.

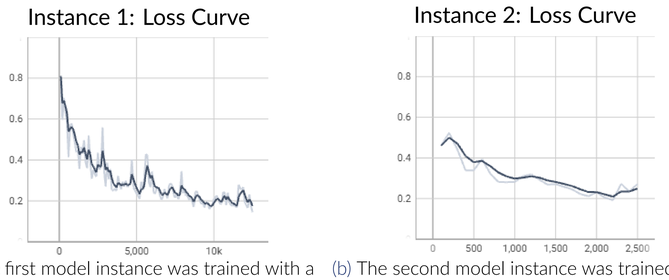
SSD MobileNetv2, YOLOv7, and Object Detection

Object detection is a technique in computer vision through which a machine learning model can identify objects and the area the objects exist in on the image.

The MobileNet computer vision model is a convolutional neural network (CNN) open-sourced by Google designed for mobile applications. A MobileNet uses depthwise separable convolutions, allowing the model to use significantly fewer parameters than a model with regular convolutions. Therefore, MobileNet models are much more efficient than other CNNs while still making accurate and precise predictions. SSD MobileNet v2 uses a single shot detector (SSD) architecture, allowing the MobileNet to make accurate labeling predictions in just one pass, much faster than other CNNs [1].

YOLOv7 is an object detection algorithm released in July 2022 that is faster and more accurate than any other object detector on GPUs. It is capable of running almost at real-time, and the success of the YOLO model has led it to become one of the standards for real-time object detection [2].

Results



(a) The first model instance was trained with a **batch size of 4 in 12,500 steps.**

(b) The second model instance was trained with a **batch size of 16 in 2,500 steps.**

Figure 2. Loss curves generated for the first and second model instances.

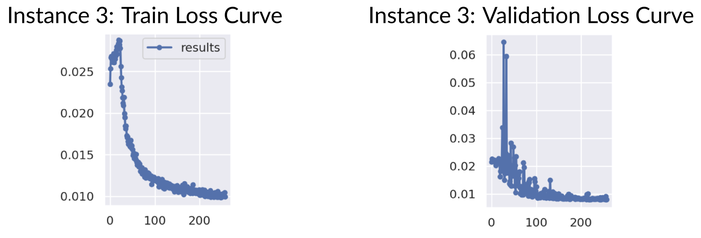


Figure 3. The third model instance was trained with a **batch size of 16 in 270 steps.**

Analysis

Based on the training loss curves, all three model instances converged smoothly which demonstrates that all three models were well-trained on our custom dataset. Images from the test subset were shown to our models, and they were able to predict the correct direction with high probability.

The YOLOv7 instance was able to achieve **99.5 percent mean Average Precision (mAP), 98.5 percent precision, and 100.0 percent recall** on the dataset that was created.

Implementation

We used the OpenCV and Tkinter libraries to implement a simple snake game in Python that used a live video feed of a player to record hand gestures. These gestures would control the movements of a snake that eats food in order to grow and earn points while avoiding hitting walls.

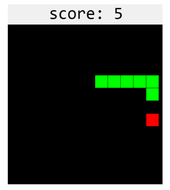


Figure 4. The user interface of our snake game.

However, when we tested our snake game, the models would incorrectly classify the gestures. For example, a "thumbs up" gesture to the camera would often be classified as a "thumbs down" gesture. Similarly, the models would confuse a "right" gesture with the "left" gesture. Considering the aforementioned favorable analyses, these inaccurate predictions were likely the cause of a biased dataset. It is likely that the images in our dataset lacked variety, causing the models to fit to a very specific idea of a gesture instead of a general understanding of it.

Conclusions

The conclusion drawn is that it is feasible to use transfer learning on a MobileNet or YOLOv7 model to detect directional hand gestures. However, it is important that the dataset used for custom objects must represent the variety of appearances that object can take. In the case of the dataset of thumbs, this means different angles, skin tones, and backgrounds to reduce bias in the data.

[1] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018.

[2] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.

Team Lead: Sisira Aarukapalli, Faculty Advisor: Dr. Ranran Feng