GALACTIC GESTURES: APPLICATION OF REAL-TIME HAND GESTURE CLASSIFICATION TO SPACE INVADERS

Eden Chung (ec3661), Katherine Lee (khl2145), Aliya Tang (ast2196)

Columbia University

ABSTRACT

This project explores the application of gesture recognition for gaming by designing a deep learning pipeline allowing users to control the game Space Invaders using gestures alone. These controls included left, right, shoot, and combinations thereof. Relying on a standard camera, the system is accessible without any extra hardware. The implementation involved evaluating deep learning models You Only Look Once (YOLO), MobileNet, Vision Transformer, assessing performance in gesture recognition. MobileNet emerged as the optimal choice due to having a lower latency and higher accuracy score (100% test accuracy). This project demonstrates the feasibility of real-time gesture-based gaming and highlights its potential as a novel and influential idea in the gaming sphere. As tech giants develop products that emphasize gesture-based human-computer interactions, this research shows promise both in commercial industries and in applications like health and accessibility, redefining interaction systems in digital environments. The purpose of this project is to develop a real-time hand gesture classification system and integrating it into Space Invaders. The full code of the project can be found at http://github.com/edenchung/Galactic-Gestures

***Index Terms*—** Deep Learning, YOLO, MobileNet, Vision Transformer, Classification, Gesture-controlled gaming

1. INTRODUCTION

The advancement of computer vision and efficiencies in neural networks has kickstarted creative applications that enhance user interaction with technology, transforming traditional methods of interaction into immersive, human-centered experiences. In this project, we explore the integration of computer vision with gesture recognition by designing a pipeline that allows users to control the game Space Invaders solely using hand gestures. This approach experiments with bringing innovative styles of play to popular games.

To achieve this, we implemented and tested three leading deep learning models: YOLO (You Only Look Once), MobileNet, and Vision Transformer (ViT). These models were evaluated for their ability to classify hand gestures in real time while achieving low latency, which is a critical factor for live interaction gaming. MobileNet achieved the highest accuracy rates in the training process and exhibited low latency when applied to real-time video feed. This model was chosen to be integrated into the pipeline, mapping gestures to in-game actions such as right, right and shoot, left and shoot, and shoot.

By leveraging computer vision, we aim to showcase a platform that offers a glimpse into the future of mainstream gaming and introduces new ways to connect with digital media.

This paper addresses the process of designing, implementing, and evaluating the classification models and applying the trained model to the game Space Invaders. After analysis of the model, we discuss the limitations and the potential future impact of gesture-based gaming on accessibility and user engagement.

2. BACKGROUND

This project requires two steps: first, an object detection step, and then an object classifier. There are various existing models and methods that can perform this task, so we experimented with three different models; each model was trained, and then the results were evaluated. In addition, for use in gaming, accuracy is not the only important factor, low latency is also an important factor needed to create a playable game.

2.1. YOLO

The first model used was the YOLO model. Developed in 2016, this model was chosen due to its quality in object detection and its speed; YOLO's base model is able to process video in real time with very limited latency. It is able to achieve this speed through just a single-pass of the model and simultaneously performs bounding box detection and classification.

First, each image is divided into a $S \times S$ grid. A given cell is responsible for making a prediction on an object if its center falls in this cell. Each cell containing an object needs to predict the bounding boxes, and the corresponding confidence score for each bounding box. Finally, it is also responsible for giving the conditional class probabilities for each detected object.

The architecture of YOLO consists of a convolutional neural network, containing 24 convolutional layers and ending with 2 fully connected layers. The model takes as an input a fixed size image, which will then be resized to a single tensor. The earlier layers detect simpler patterns such as edges and corners, whereas later layers will attempt to learn more complex features. Finally, the fully connected layers interpret the features that the convolutional layers predicted to output the bounding box coordinates, confidence scores, and class probabilities. YOLO also combines three loss functions to create an optimized loss function for object detection and classification[1].

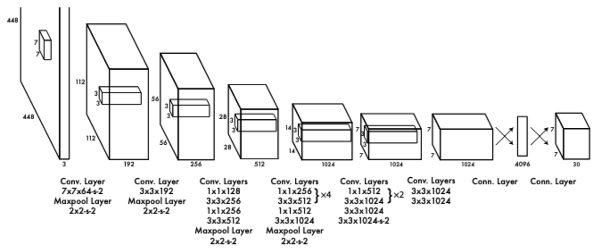

448
448
3
112
112
192
56
56
256
28
28
512
14
14
1024
7
7
1024
1024
4096
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Fig. 1. YOLO Architecture[2]

2.2. MobileNet

The second model addressed is MobileNet. MobileNet is a very lightweight model developed by Google in 2017 aimed to address the computational limitations in mobile and embedded vision applications. This model implements depthwise (applies a single filter to each input channel separately) separable convolutions to significantly reduce computational requirements while maintaining competitive accuracy compared to more resource-intensive models[3]. This design choice partitions the convolution operation into two layers: a depthwise convolution, which applies a single filter to each input channel, and a pointwise convolution, which combines the outputs of the depthwise operation. This factorization reduces the number of operations by a factor approximately equal to the number of output channels[4]

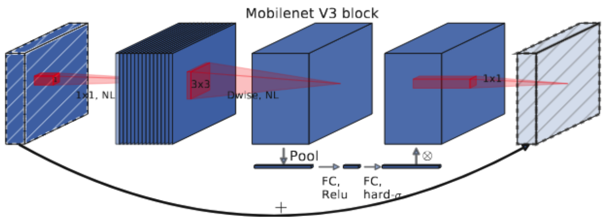


Fig. 2. Mobile Net Architecture[3]

MobileNetV3 combines MobileNetV2's bottleneck blocks (containing an expansion, depthwise convolution, and projection layer) with SE (Squeeze-and-Excitation) blocks for taking the most important feature's in data and highlighting the useful parts of each channel while suppressing the less relevant ones and hard-swish activations for efficiency [5]. MobileNetV3 includes standard depthwise separable convolutions in bottleneck blocks. MobileNetV3-Large uses around 66 layers compared to MobileNetV3-Small (54 layers).

MobileNetV3-Large was chosen as the primary model due to its lightweight architecture, optimized specifically for mobile applications, and its ability to handle high-resource scenarios such as live, dynamic data. The large type was particularly effective for this project because its capability to process rapidly changing inputs in real time, making it well-suited for animated gesture recognition tasks compared to models designed for static image classification. MobileNetV3-Large demonstrates a 3.2% improvement in top-1 ImageNet accuracy over MobileNetV2 while reducing latency by 20% on mobile CPUs [4]

Given this, MobileNetV3 is a powerful choice for real time application such as gesture recognition and video analysis.

2.3. Vision Transformer

Finally, the Vision Transformer (ViT) was chosen. Transformers were first proposed in 2017 by Vaswani et al. [6] for machine translation, but have since then been used to solve natural language processing problems. ViT was first developed in the paper written by Dosovitskiy et. al, where instead of using it for natural language processing processes, they utilized it for image classification tasks[7]. They developed the architecture the same way that Vaswani et al. wrote their architecture[6]. They would split the image into different patches, then linearly add embedding to each of them and add positioning embeddings. Finally, they fed the resulting sequence of the vectors to a standard Transformer encoder. They used the standard approach to add in the "classification token" to the sequence. They would use an multilayer perceptron (MLP) as the classification head with one hidden layer at the pre-training time and a single linear layer at fine tuning time. The MLP contains two layers with a GELU nonlinearity.

ViTs have less of an inductive bias compared to convolutional neural networks (CNNs). This is because CNNs have an intertwined two-dimensional neighborhood structure and translation equivariance compared to ViT where only the MLP layers are local and translationally equivalent. Furthermore, it was reported that the ViT outperforms the ResNet model on every single dataset that it was trained on.

When trained on large datasets, transformers retain the highest accuracy, but they can be easily tailored and fine tuned towards specific needs. Hence, although our dataset is small, we could use pretrained weights on the model so that we could fine-tune the model for our specific needs.

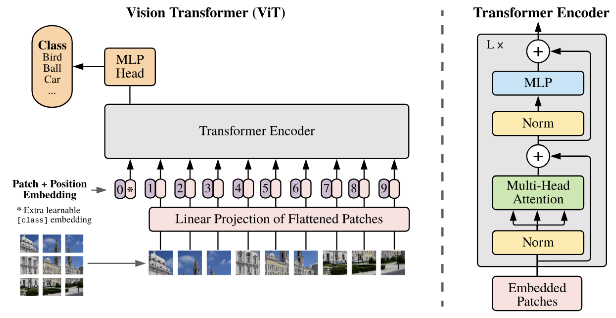
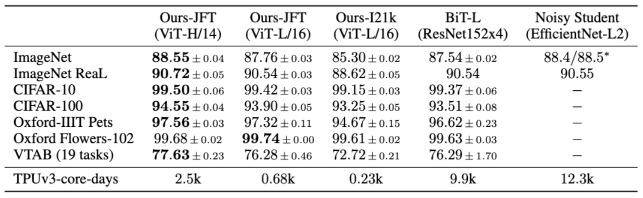


Fig. 3. Vision Transformer Architecture[7]



	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Fig. 4. Vision Transformer Architecture[7]

Based on the background research, we designed a pipeline, that evaluates and then integrates these models, (YOLO, MobileNet, ViT) into the game *Space Invaders* (Figure 5).

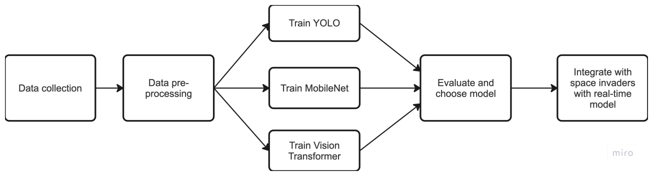


Fig. 5. Project Overview Diagram

3. RELATED WORK

In terms of gesture classification problems, much work has been done, such as American Sign Language classification, location sensor, etc.

Several different techniques have been used for gesture classification problems. One technique involved color-based recognition, where the user wore a glove with distinct colors distributed across its surface to help the classification model accurately identify different parts of the hand. [8]. Another method aimed to isolate a hand from its background by classifying each pixel as either "hand" or "not hand", ultimately building an image of the hand. However, this approach is less effective when the background is complex or colorful, as it becomes harder to accurately distinguish the hand. Among the proposed methods, deep learning-based recognition, particularly using basic convolutional neural networks (CNNs), demonstrated the best performance. However, with complex backgrounds with noise, again, this method may not have an optimal performance, but it does seem like the best approach to proceed with.

There is much existing work on movement control using gestures to play video games. The most common method for gesture-based gaming requires additional hardware. For example, the Nintendo Wii U uses handheld controllers for motion sensing, and the Xbox Kinect contains a depth sensor, a color camera, and an infrared sensor[9]. However, we wanted to choose an approach that would be easy to integrate without purchasing any additional hardware, so we wanted to use a monocular visual sensor, as this means anyone with a phone or computer would be able to integrate it into their system.

We were inspired by a project completed in 2023, where the game *Assassin's Creed* could be controlled by jumping and tilting the head in real life.[**?**] However, this project utilizes position detection in the frame to determine whether the user's head is tilting left or right. Comparatively, in our project, we wanted to focus more on a classification problem. Some research has been done for gesture classification. One past approach was using Faster R-CNNs to do object detection[10]. However, this approach requires a higher computation time. YOLO implementations were found to have a slightly higher performance and can produce results with a higher FPS when used for real-time detection[10].

We would also like to extend thanks to Lee Robinson, who developed the game *Space Invaders* in Python and PyGame, which we used as a base for our project. The code of this repo was modified to include our real-time classification model instead of keyboard input.

4. DATA COLLECTION & PROCESSING

In order to create a game controlled by hand gestures, we had to decide the gestures we wanted to use, then manually collecting a dataset.

We decided that there should be 5 possible actions in the game: move left, move left and shoot, shoot, move right, and move right and shoot. Then, we assigned a specific hand gesture to each of these actions. In order to obtain a diverse dataset, we asked friends and family to perform each of these 5 hand gestures, which we then added to our dataset.

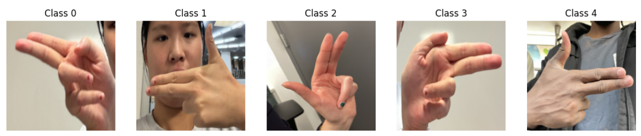


Fig. 6. Hand gestures and their corresponding classes

We collected 573 images in total from approximately 30 different people.

Class number	Class action	Number of images
0	Left	109
1	Left + shoot	118
2	Shoot	143
3	Right	101
4	Right + shoot	102

Table 1. Breakdown of dataset per class

Since this machine learning problem consists of both object recognition (the hand gesture), and then object classification (identifying represented gesture), the next step was to draw bounding boxes on each image. This was done efficiency by developing a simple Python script that allowed drawing a rectangle on top of the image, then would save the top left corner coordinates and bottom right corner coordinates of the bounding box to a csv.

Finally, given that the data was collected manually, the dataset size is not very large ($n = 573$). Thus, data augmentation is a technique that will allow us to create more training examples out of the current images. As our hand symbols contain directions (left and right), it is important that the augmentations never flip the image. So, the augmentations used were a random change of brightness and contrast, a small rotation of the image, a shift/scale/rotate, a Gaussian blur, and a modification of the hue or saturation. A certain probability was added to each of these augmentations, then with the help of the library *albumentations*. During the training process, each image would have a random chance of having one or more augmentations applied to it. Figure 7 demonstrates several images that have had various augmentations applied to them, such as changes in brightness, hue, saturation, and rotations.

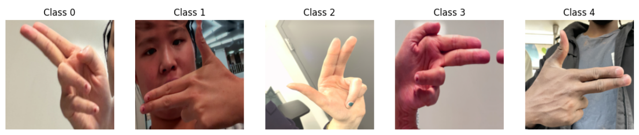


Fig. 7. Augmented images

The dataset was split into 80% training, 10% testing, and 10% validation, and the data augmentation was applied only to the training set.

5. CLASSIFICATION MODEL

The next step in the project is to create a classification model, which can classify images of hand gestures to their respective actions. We chose three models with different model architectures, then evaluated their performance metrics to determine which one was best suited for the task at hand.

5.1. YOLO

The first model chosen is YOLO version 5. The data had to be slightly modified to work with the YOLO training inputs. Initially, the bounding box data was stored in a file called annotations.csv, which contained the name of the image, along with the coordinates of the bounding box, plus the label class. However, YOLO requires each image to have its own .txt file, named after the respective image, containing the class label and the bounding box coordinates.

Once the data was in the correct format, the YOLO model was trained on 300 epochs with a batch size of 16, with early stopping to prevent over-fitting. This was repeated for YOLO small, YOLO medium, and YOLO large, which allowed the comparison of different model sizes. The tradeoff with larger models is that training time is substantially longer.

5.2. Mobile Net

To apply the MobileNetV3-Large for gesture classification, the model was edited by replacing the final fully connected layers with a new linear layer consisting of five output neurons, each representing one of the gesture classes. This modification enabled the model to generate probabilities for each class. The use of a pre-trained model provided an additional advantage by leveraging prior knowledge from large-scale datasets, accelerating the training process, and improving generalization.

The training process incorporated several essential components. Cross-entropy loss was used to measure the difference between predicted and actual class distributions. The Adam optimizer, with a learning rate of 0.001, was applied to ensure efficient gradient updates. Additionally, early stopping was implemented with a patience level of 10 epochs to prevent overfitting, halting training when no significant improvement was observed. The model was trained for a total of 60 epochs.

A MobileNet model with untrained weights initialized using Xavier initialization was also developed. However, the pre-trained model consistently achieved superior accuracy. As a result, the decision was made to fine-tune and further optimize the pre-trained model rather than develop one entirely from scratch.

5.3. Vision Transformer

There were two ViT models written: one with pre-trained weights and the other from scratch. The model with pre-trained weights helps with transferring knowledge from a large dataset. The specific ViT model that was used was vit_b_16 which is a specific variant of with 16x16 patch size. The pretrained weights were trained on ImageNet, which is a large and diverse dataset that can help with feature extraction for our smaller dataset.

The model used an Adam optimizer with a learning rate of 0.001 for the classification head, bounding box head, and the pre-trained layers of ViT. Cross Entropy Loss was used for classification loss, and Smooth L1 Loss was used for bounding-box regression. The model also used early stopping to halt training if the classification accuracy did not improve after 10 additional epochs. The model trained was trained for 100 epochs with a batch size of 16. After fine-tuning, the model performed significantly better at around 96% accuracy. The ViT model from scratch was not as accurate at only around 20% accuracy rate because this model was not fine-tuned.

6. RESULTS AND ANALYSIS

For each model, we measured upon various key metrics such as loss, accuracy, precision, recall, and F1-score, and were calculated for each gesture class, assessing what aspects the model struggled in classifying.

We graphed the precision recall curve for the YOLO model and the normalized confusion matrix for all three models. The precision recall curve was 0.906 for all classes, class 0 (move left) performing the worst at 0.811 and class 2 (shoot) performing the best at 0.995 (Figure 8). The normalized confusion matrix for this model was divided into the different classes, similar to the precision recall curve. The two highest performing classes were class 1 (right shoot) and class 2 (shoot), which both performed at 100% compared to the lowest performing class which was class 3 (right), which only performed at 50% (Figure 9). From this matrix, we can see that YOLO was often confusing the classes “left” with “right”, and “left_shoot” with “right_shoot”.

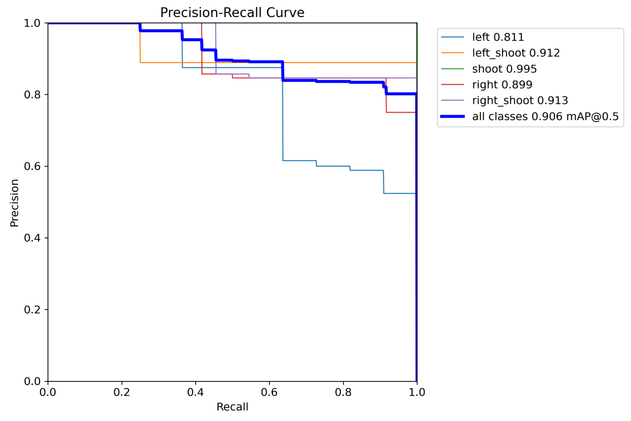


Fig. 8. YOLO Precision Recall Curve

For MobileNet, we only output the normalized confusion matrix. The normalized confusion matrix for MobileNet has a perfect test classification performance at 100% across all different classes. There are no off-diagonal values, meaning

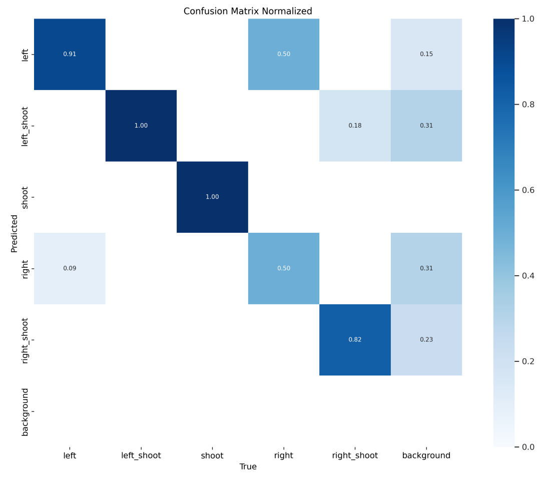


Fig. 9. YOLO Confusion Matrix

that the model did not make any miscalculations across the different classes (Figure 10).

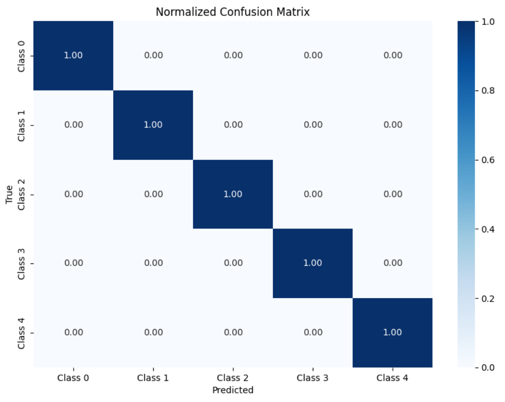


Fig. 10. MobileNet Confusion Matrix

The ViT model performed similarly to the MobileNet model, but it was not as accurate for two separate classes. In this case, it confused “left” with “left_shoot” occasionally, as the hand is facing in the same direction. (Figure 11).

In the end, we decided to utilize MobileNet because of its superior performance across all classes for integration into the *Space Invaders* game. The loss curve for the test dataset for the MobileNet model was at 1.75%, which indicates that the model is consistent and accurate on its predictions on the test data. In addition, it is a smaller sized model, which is perfect for real-time classification. We then implemented the MobileNet model into Lee Robinson’s Python implementation of *Space Invaders* [11].

After integrating the model into the *Space Invaders* game, we observed an issue where the model incorrectly classified

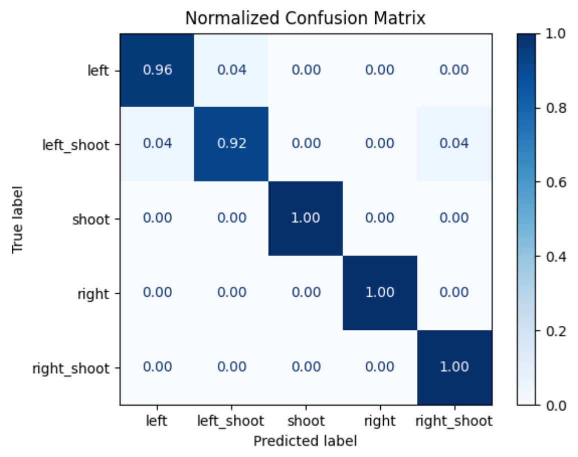


Fig. 11. Vision Transformer Confusion Matrix

frames with no gesture as one of the five actions. This caused unintended movements or actions, such as the spaceship moving or shooting, even when the player was not holding up their hand.

To address this issue, we tried two approaches. The first approach was to add a confidence threshold, so that any frames that had predictions below this threshold would not lead to an action in the game, but any frames that had a high confidence would be classified as a gesture. This method was not extremely effective as some empty frames without a gesture were still being classified with a high confidence level.

We then tried a second approach, where we decided to retrain the model with an added class where there are no hands in the frame. After retraining the MobileNet model, the *Space Invaders* game ran much more smoothly, however, the user does have to place their hand close to the camera, so in the future, this is definitely an area of improvement.

7. CONCLUSIONS

The results of our model show the effectiveness of our pipeline and provide insight into the future of a new form of entertainment. It highlights the potential of integrating hand and body recognition into interactive gaming, particularly for real-time response. The selection of MobileNet over YOLO and ViT demonstrates that the largest, highest weight model is not always the best for all applications. In some situations, a lightweight model designed for mobile applications and speed can be better suited for certain tasks.

We initially thought that the YOLO model would work well for our implementation due to its use in similar tasks. Unfortunately, it ended up performing the worst compared to the other models. After evaluation, the results suggested that the YOLO model misclassified data. We believe this may be because YOLO might require more data if we are training it from scratch. Given more time, we could have tried collecting more data or tried to fine-tune YOLO instead.

In the end, MobileNet was selected because it successfully met the pipeline's main objectives: achieving high accuracy in gesture recognition models and high classification speed in the in-game experience, where users play *Space Invaders* on their computers. If we had more time, we would have potentially trialed another approach, where we first train a binary pre-classifier, which would determine whether an image contains a hand or not, before sending it to the main gesture classifier. This could improve accuracy even further. Regardless, a relatively high accuracy was still achieved, and while latency and classification accuracy could still be improved, especially for gestures with similar hand signs, the study demonstrates the feasibility of gesture-based controls in gaming.

The straightforward setup of the project, which only requires readily available hardware like computer cameras, makes it an accessible entry point for future advancements in AR/VR gaming and gesture-based interaction systems. This pipeline can be quickly augmented and optimized for other gestures. Because of its low memory and space, MobileNet can easily be run on lower computation devices, thus making this specific model applicable towards different hardware platforms.

The results for this integration provided adequate evidence that the MobileNet model can be effectively integrated into the *Space Invaders* game. This can be useful for different means, such as improving motor skills for physical therapy purposes.

8. RELATION TO PRIOR WORK

The models and pipeline for our project were inspired by previous work in the field. We have seen the use of YOLO and MobileNet for use in gesture recognition, and were also interested in experimenting with Vision Transformer, a newer architecture that has had less work done in the field.

In this project, we have succeeded in developing a program that allows a user to operate a game solely using their camera, which makes it more accessible than prior work requiring specialized hardware, although the trade-off may be slightly worse accuracy. However, with a larger dataset, the accuracy could be greatly improved; for example, the MNIST ASL dataset could be used to create 26 different inputs for a more complex game.

Overall, while prior research has explored gesture-based gaming largely using hardware-based systems and position detection, our work demonstrates how a deep learning approach solely using a camera is able to achieve comparable results.

9. FUTURE WORK AND IMPACT

This research has broader implications beyond gaming and real-world applications, particularly wearable technology using augmented and virtual reality. As these technologies become ubiquitous, gesture-based systems could redefine user interaction paradigms, potentially becoming the new standard for communication with computers. As seen in Meta's Orion and Apple Vision Pro, gesture controls represent a more intuitive and immersive way to interact with virtual environments, potentially replacing traditional inputs like keyboards or controllers. This creates new opportunities for human gaming experiences.

Gesture-based systems also have significant potential in health applications. In physical therapy, custom hand gestures tailored to individual recovery plans can help patients regain motor function while providing doctors with real-time feedback on their progress. By playing the gesture-based *Space Invaders*, patients can train their motor skills. Similarly, fitness applications could use gesture recognition to guide users through exercises, promoting health and wellness in a more engaging and interactive way.

Furthermore, this real-time gesture detection and classification system can be extremely useful for home automation. For instance, for individuals who have limited mobility, this might be an effective system for them to have. Instead of having to reach a hard-to-access light switch, this system can automatically detect turning on the lights with one gesture. In the future, we hope to apply this sort of system to assist those who are in need.

10. CONTRIBUTIONS

All members contributed equally to the data processing and pre-processing. Each team member hand labeled 1/3 of the dataset. Then, each team member was then responsible for training 1 type of model. Eden trained the YOLO model, Katie trained the MobileNet model, and Aliya trained a Vision Transformer. Eden was then responsible for retraining the MobileNet model with the new class ("None class") and integrating the final model into the game *Space Invaders* in Python.

11. REFERENCES

[1] "Detector loss function (yolo loss)," `https://www.oreilly.com/library/view/hands-on-convolutional-neural/9781789130331/8881054c-f6e6-485c-9c9e-357285bce60a.xhtml`, Accessed: 2024-12-08.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," 2016.

[3] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam, "Searching for mobilenetv3," 2019.

[4] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2019.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," 2023.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[8] Muhammad Oudah, Abbas Al-Naji, and Javaan Chahl, "Hand gesture recognition based on computer vision: A review of techniques," July 2020.

[9] Zhengyou Zhang, "Microsoft kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, 2012.

[10] Nhat Vu Le, Majed Qarmout, Yu Zhang, Haoren Zhou, and Cungang Yang, "Hand gesture recognition system for games," in *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, 2021, pp. 1–6.

[11] Lee Robinson, "Space invaders," `https://github.com/leerob/space-invaders`, Accessed: 2024-12-08.