Performance and Feasibility of Real-Time Gesture Recognition on Consumer-Grade Computers

ILLIA SOLODKYI, University of Twente, The Netherlands

The availability of gesture recognition technologies raises the potential for improvement of human-computer interaction and the adoption of gesture recognition as a widely used way of interaction. Adoption continues to be limited on consumer-grade computers because of concerns about high processing demands and specialized hardware requirements. This study assesses the feasibility of implementing webcam-based gesture-controlled interfaces on typical consumer-grade computers without decreasing overall system performance. Multiple machine learning models, including traditional classifiers and boosted trees, were trained and benchmarked for real-time CPU, GPU, and RAM usage. Optimization techniques such as pre-pruning and frame skipping were applied to reduce resource demands. Resource consumption was estimated for several types of real-world computer configurations, from high-end desktops to low-cost laptops. The results show that models with the adoption of optimization techniques can achieve high accuracy while consuming less than 2% of computational power, making real-time gesture recognition feasible for the majority of consumer-grade computers, with the exception of low-end laptops with constrained processing power. This research highlights key system requirements, trade-offs, and future directions to encourage broader adoption of gesture-based interfaces on consumer-grade computers.

CCS Concepts: • **Computing methodologies** → **Machine learning algorithms**; • **Human-centered computing** → **Gestural input**.

Additional Key Words and Phrases: Gesture Recognition, Human-Computer Interaction, Webcam-based Interaction, Real-Time Processing, Resource Optimization, Consumer-Grade Computers, CPU and GPU Benchmarking, Machine Learning Models

1 Introduction

Technology keeps evolving in the modern world, and so does how users interact with their devices. One form of this evolution is where feature phones with button interaction have been replaced by smartphones that provide gesture interaction. This step is only a precursor to an overall movement of natural and intuitive human-computer interfaces. Gesture recognition [12] is the process by which a system detects and responds to human movements as input, using sensors such as cameras or motion detectors. Gesture-based interfaces which allow users to communicate with computers using body gestures are becoming more and more popular in a variety of application areas, such as gaming and smart homes. Despite its increasing popularity in the aforementioned use cases, gesture-controlled interfaces are still not widely used in regular consumer-grade computers.

This hesitation to utilize gesture-controlled interfaces on typical computers is mainly due to worries regarding hardware constraints, inconsistent performance, and the assumption that gesture recognition necessitates specific hardware. Indeed, hardware constraints were present as different studies used different hardware and equipment, such as [6] using two cameras and colored gloves, [13] using two types of gloves: sensor-based data gloves and multiple-colored gloves, and [8] using the Red, Green, and Blue finger caps for the gesture-based human-computer interaction. Such hardware and equipment requirements to use gesture recognition for computer control are stopping average users from using it. However, [16] and [17] have shown that the development of a human-interaction system that uses only one single camera is possible. This makes gesture-based human-computer interaction systems more accessible to average users and opens new opportunities for wide-scale adoption. However, the computational load of such human-computer interaction systems in terms of an average computer and their influence on computational load and overall performance of the computer remains open.

In real-world scenarios, users operate desktops or laptops with limited processing power and resources, frequently running several programs simultaneously that consume the computer's limited resources. Gesture-based human-computer interaction solutions must function well without significantly affecting system performance to be realistically feasible for use on consumer computers with limited resources. High CPU, GPU, or RAM consumption can cause laptops to lag, overheat, and have shorter battery lives, all of which have a negative impact on usability and the likelihood that technology will be widely adopted.

This research aims to evaluate the influence of gesture-based human-computer interaction systems on consumers' computers, focusing on processing power usage. After training gesture recognition models, CPU and GPU resources used by the gesture recognition software were benchmarked. After the measurement of the CPU and GPU resource utilization, market research on the average consumer-grade computer specifications was made. Then, the resource utilization was projected onto consumer-grade computer specifications based on the feasibility of a real-time gesture recognition human-computer interaction system was assessed.

2 Problem Statement

Previous research on gesture recognition has required complex systems consisting of multiple cameras or sensor-enabled gloves. Recent studies have, however, proved gesture-based human-machine interaction to be technologically feasible with just a single webcam. However, in contrast to this headway, much remains to be discovered about the computational cost of executing gesture recognition software on standard consumer hardware. In particular, limited information exists regarding how much processing resources—CPU, GPU and RAM—are consumed when such systems operate in real-time. Because the majority of consumer computers possess limited resources and tend to execute several different applications at any

Author's Contact Information: Illia Solodkyi, i.solodkyi@student.utwente.nl, University of Twente, Enschede, The Netherlands.

TScIT 43, Enschede, The Netherlands

given time, heavy system requirements of gesture-based systems can lower usability and make widespread adoption less feasible. The aim of this research is to examine the processing needs of gesture recognition systems of consumer-level personal computers. The usability of the gesture-controlled interface by average users will be determined by testing the system load and comparing it with standard system specifications.

2.1 Research Question

The following research question will emerge from the problem statement:

Is the performance of gesture recognition systems feasible for real-world use on consumer-grade computers without significantly degrading overall system functionality, and to what extent do they impact CPU, GPU and RAM usage?

This Research Question can be answered with the following sub-questions:

(1) How much CPU and GPU power does gesture recognition software use during real-time operation?
(2) What is an average CPU, GPU and RAM specifications for consumers' computers?
(3) How does running gesture recognition software affect overall consumers' computers system usability?

3 Related Work

Scopus and Google Scholar were used to collect relevant literature for the research topic. Using search terms like "gesture recognition" and "human-computer interaction", a number of documents were discovered that contain conducted research in this domain.

Gesture recognition [12] is a well-researched technology, the capabilities of which grow with time. With the advancement of technology, it can be seen that the need for external hardware or equipment, which was present in [6] (two cameras and colored gloves), [13] (sensor-based data gloves and multiple-colored gloves), and [8] (Red, Green, and Blue finger caps) has been eliminated. [16] and [17] have shown that the development of a human-computer interaction system that uses only one single camera is possible. Moreover, in [4], it is discussed what steps should be taken to create a working gesture recognition model that can be later used in a human-computer interaction system similar to [16]. Algorithms that can be used for gesture recognition in such human-computer interaction systems are described in [10], [9], [5], [14], [2] and [2]. In addition, [11] shows that the single camera was the most used environment type that is used for gesture recognition from 2014 to 2020.

[3] has shown that the digital camera or web camera as an input device can outperform hardware-specific input devices like colored gloves or data gloves, achieving a recognition rate as high as 98.7%, which can be considered a feasible recognition rate for widespread usage and increase adoption rate due to no additional hardware requirements except for webcam.

In addition, [1] and [15] describe techniques that can be used for the optimization of gesture recognition models in a case when the hardware requirements for the non-optimized models are too high for an average consumer-grade computer.

4 Methods of Research

This section will cover the research methodology, which is divided into six phases. Each phase describes actions taken to complete the research. The research was done using the Python programming language. The Main Python libraries used during the research were sci-kit-learn for data splitting, model training, and evaluation; matplotlib for plotting confusion matrix; and joblib for model storage and later use.

4.1 Initial Design

The initial design phase includes the preliminary design of the research, model development, challenges encountered with the initial solution, and justification of the changes done to the initial design. Initially, the machine learning algorithms were planned to be trained on the photo datasets containing people showing different gestures. The main problem with dataset selection was that the majority of the datasets for the gesture recognition algorithms were in black and white and focused on the palm instead of the top body view that would be received from the regular webcam. The dataset that would fit the intended view was the HaGRID dataset [7]. After retrieving datasets for the three gestures, instead of the thirty-three gestures provided by HaGRID due to the size of the datasets for each gesture being more than 100 GB, a couple of models were trained on this subset of images. The results of the trained models were approximately twenty-three percent, which is less than random guess results given that only three classes are present. Such poor results were caused by the fact that the images contained a lot of redundant information and changes such as different backgrounds, clothes, colors, etc. After retrieving such poor results, the decision to pre-process the photos was made. Photo pre-processing involved detecting hands in the photos before passing them to the models. YOLOv5 pre-trained models for hand detection were used for this task. Unfortunately, even with pre-processing, the models were showing poor results, and in addition, hand detection models themselves were unreliable. Consequently, additional literature research was conducted to identify a more effective solution.

4.2 Software Creation and Evaluation

The software creation phase will cover the process of the creation of the final solution for model training, machine learning algorithms research and implementation, evaluation, and benchmarking of the models. After the aforementioned additional literature research, Google's MediaPipe Hands[1] (later referenced as MediaPipe) was found and tested. MediaPipe detects hand and creates landmarks on them. MediaPipe showed promising results on the live video testing. However, they performed poorly on the HaGRID dataset, as the photos in the dataset have people standing further than expected in regular web cameras. Because of this, the decision was made to create a dataset using the regular webcam. The creation of the dataset was done by capturing 2000 frames per gesture, 1000 frames for the

[1]Google MediaPipe Hands: https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker

left arm, and 1000 frames for the right arm. In addition, 2000 frames of the hands in the rest or showing the gestures not included in the model were taken. After the frame collection, these frames were processed to create landmarks, and these landmarks were stored to be later used for the model training. Models trained on these landmarks showed sufficient results with approximately ninety-seven percent accuracy. After training, these models were tested using real-time video from the web camera. MediaPipe processed each frame from the web camera by creating the landmarks that the model then processed to create a prediction. After confirming that the models perform well on the real-time video and training, the Python application was developed to recognize the gestures and perform the action on the computer accordingly. One application version performs back, page up, and page down commands, while the second version stops the music, starts the next song, or returns to the previous one. Moreover, a brief delay is introduced between actions to prevent the application from executing the same action multiple times for a single gesture while still allowing the user to repeat the desired command. After developing the application, additional academic research was conducted to identify alternative machine learning algorithms that could be evaluated and benchmarked later. In the end, six machine learning algorithms (Bagging Classifier [10], k-Nearest Neighbor [9], Logistic Regression Predictor [5], Multi-Layer Perceptron [14], Random Forest Classifier [2], Support Vector Machine [2]) and four boosted trees were trained. The training and validation split for these models were created by randomly selecting twenty percent of the frames for the validation set. The remaining eighty percent were used for training. Additionally, the F1 score was monitored to help detect and prevent overfitting. After the training, all models were stored as joblib files for later use. The accuracy of all models ranged between 93.37 and 99.44 percent.

4.3 Benchmarking

The benchmarking phase will cover steps taken to benchmark all models. The benchmark process started by creating a one-minute-long video, which included showing each gesture with both hands, showing gestures outside of the scope of the classes, and cases when there is no hand visible in the frame. The video is in 480p and has a frame rate of 60 frames per second. After the creation of the video, the frames of this video were sent to the prediction models while monitoring the GPU, CPU, and RAM usage of the computer. The psutil Python library was used to monitor the CPU and RAM. This library tracked RAM and CPU usage that were dedicated to the process. Unfortunately, GPU usage per process tracking is only available for NVIDIA GPUs. As the models were tested on the AMD GPU, getting per-process usage was impossible. Instead, the overall GPU usage of the computer was recorded. Such recording was done by using the OpenHardwareMonitor Python library. To ensure that the data was as precise as possible, the only process running during the benchmarking process was the benchmarking process itself. No mouse movements were present to even more reduce unrelated GPU usage. The results of the benchmarking can be seen in the 1 and 2. The 1 shows the performance metrics of the models, and the 2 shows the resource usage of GPU, CPU, and RAM of each model.

Table 1. Model Performance Metrics

Model	Accuracy (%)	Weighted F1	Score
k-Nearest Neighbors	99.43	0.9943	0.31
Logistic Regression	94.44	0.9444	0.37
Support Vector Machine	93.37	0.9334	0.46
Bagging Classifier	98.44	0.9844	0.47
Multi-Layer Perceptron	98.69	0.9869	0.50
Random Forest Classifier	99.12	0.9912	0.58
CatBoost Classifier	98.44	0.9787	3.92
XGBoost Classifier	99.12	0.9912	8.61
LightGBM Classifier	99.31	0.9931	10.36
Histogram-based Gradient Boosting	99.44	0.9944	16.86

Table 2. Model Resource Usage

Model	Avg FPS	Avg CPU (%)	Avg RAM (MB)	Avg GPU (%)
k-Nearest Neighbors	32.08	3.43	576.84	0.05
Logistic Regression	32.13	3.06	572.88	0.06
Support Vector Machine	32.10	3.15	577.36	0.07
Bagging Classifier	32.05	3.72	566.43	0.07
Multi-Layer Perceptron	32.09	3.03	573.06	0.09
Random Forest Classifier	32.05	3.47	582.83	0.09
CatBoost Classifier	15.98	6.36	631.81	0.15
XGBoost Classifier	31.98	92.96	581.77	0.05
LightGBM Classifier	32.10	28.69	571.65	0.20
Histogram-based Gradient Boosting	29.01	60.49	570.99	0.14

The Score for each model is calculated as follows:

$$\text{Score} = \frac{\text{Avg CPU} \times \text{Avg GPU} \times \text{Avg RAM} \times 10}{\text{Accuracy} \times \text{Avg FPS} \times \text{Weighted F1}}$$

A low score indicates a good performance of the model. The findings show that the model with the lowest score, k-Nearest Neighbors, achieves a balance between accuracy and resource usage. A weighted F1 score that is almost the same as the accuracy indicates that the precision and recall of the model are equal for all classes and that there is no class bias. The model with the highest accuracy is the HistGradientBoost model, with an accuracy of 99.44%. Unfortunately, the HistGradientBoost model has the second-highest score because of the average CPU usage of 60.49%, which is twenty times more than the Multi-Layer Perceptron, which has only 3.03% usage of CPU. Almost all models have a CPU usage a little higher than 3% except boosted trees, which have CPU usage in the range of 60% to 93%. GPU usage of almost all models is also comparatively low, around 0.05%. Unfortunately, usage of boosted trees is once again higher with a range of 0.14% to 1.53%, with the higher end being 30 times larger than the usage of other models. As for the RAM usage, the range of the RAM used is from 566.43MB to 631.81MB, which is only an 11% difference in RAM usage. Data shows that boosted trees perform worse than the standard machine learning algorithms on such small datasets. The best model with the lowest score was k-nearest neighbours with a score of 0.31, and the worst performing models were boosted trees: CatBoost, XGBoost, HistGradientBoost, and LightGBM with scores of 3.92, 8.61, 16.86, 167.90 respectively. As we can see, the difference between scores is almost 542 times.

Other important metrics of the models are prediction speed and model size. Prediction speed is critical for real-time implementation and must be minimal to ensure correct operation without noticeable

delays. Otherwise, the solution will not be feasible for real-time usage as users would have to wait before command execution. Model size is an important factor, as smaller models occupy less storage space on the user's device, ensuring minimal impact on available memory and allowing faster downloads and updates. Each model's prediction speed and size are shown in the 3.

Table 3. Models' Speed and Size

Model	Average Prediction Time (ms)	Model Size (KB)
Bagging	5.53	1860
CatBoost	0.63	269
Histogram-based Gradient Boosting	4.67	1460
k-Nearest Neighbors	2.39	4942
LightGBM	0.22	1391
Logistic Regression	0.29	3
Multi-Layer Perceptron	0.33	532
Random Forest	4.38	5359
Support Vector Machine	0.62	1530
XGBoost	0.64	980

4.4 Optimization

The Optimization phase will cover the optimization techniques applied to the models and how these optimizations have influenced the performance of the models.

4.4.1 Pre-Pruning. The first optimization technique that was used was Pre-Pruning. Pre-pruning is a technique that limits the tree growth of the decision-tree-based models during training. This technique helps create smaller models. Moreover, smaller models can have less prediction time because fewer decisions are evaluated per prediction. Pre-pruning optimization techniques were applied to the following models: Bagging Classifier, CatBoost Classifier, Histogram-based Gradient Boosting, LightGBM Classifier, Random Forest Classifier, and XGBoost Classifier. The Pre-Pruning was done by adjusting tree hyperparameters to make the model as small as possible without significant loss of accuracy. The hyperparameters for each model were adjusted individually for Pre-Pruning, as each model had unique hyperparameters that would be passed to the model constructor. The hyperparameters for the models are shown in 4.

Table 4. Hyperparameters for different classifiers

Classifier	**Hyperparameters**
Bagging Classifier	`max_depth = 7, min_samples_leaf = 10`
CatBoost Classifier	`iterations = 100, depth = 5, l2_leaf_reg = 10.0, min_data_in_leaf = 10, grow_policy = 'Depthwise', verbose = 100`
Histogram Gradient Boosting	`max_depth = 2, max_leaf_nodes = 3, min_samples_leaf = 10`
LightGBM Classifier	`learning_rate = 0.1, max_depth = 2, num_leaves = 3, min_data_in_leaf = 20, min_gain_to_split = 0.1, num_boost_round = 100`
Random Forest Classifier	`n_estimators = 100, max_depth = 5, min_samples_leaf = 10`
XGBoost Classifier	`n_estimators = 50, max_depth = 3, min_child_weight = 10, gamma = 1.0, learning_rate = 0.1`

The main objective was to reduce the model size without losing more than 5% in accuracy. The results of the Pre-Pruning can be seen in the 5.

Table 5. Pre-Pruned Models' Speed and Size

Model	Average Prediction Time (ms)	Model Size (KB)
Bagging Classifier	5.29	597
CatBoost Classifier	0.63	234
Histogram-based Gradient Boosting	4.10	319
LightGBM Classifier	0.20	194
Random Forest Classifier	4.34	565
XGBoost Classifier	0.63	227

As shown in the 5, Pre-Pruning is a powerful technique if the size reduction is needed. Size reductions of the models are in the range of 13% to almost 90%, with an average size reduction being 68.57%. Such size reductions can be helpful in cases where the model size is crucial, such as model usage in edge computing. In addition, Pre-Pruning also reduced the prediction times of the models up to a 12.22% reduction. Noticeably, Pre-Pruning improves prediction time more for models with larger initial prediction time, such as Histogram-based Gradient Boosting(12.22% reduction) and Bagging Classifier(4,34% reduction).

4.4.2 Frame Skipping. Another optimization technique that was used was Frame Skipping. The frame-skipping technique was employed by analyzing only every eighth video frame. The rest of the frames were grabbed but not analyzed. The video used for frame-skipping benchmarking was the same as that of the initial solution. In addition, the time delay was implemented to ensure that frame

grabbing and analysis cannot be executed faster than in the real-time video. This way, the benchmarking is as close as possible to real-time usage settings. The models' results with frame skipping can be seen in the 6 and 7. The 6 shows the performance metrics of the models, and the 7 shows the resource usage of GPU, CPU, and RAM of each model with the usage of frame skipping optimization technique:

Table 6. Model Performance Metrics with Frame Skipping

Model	Accuracy (%)	Weighted F1	Score
Random Forest Pruned	94.06	0.9406	0.09
Bagging Classifier	98.44	0.9844	0.11
Logistic Regression	94.44	0.9444	0.12
Bagging Pruned	95.13	0.9510	0.13
k-Nearest Neighbors	99.43	0.9943	0.15
Multi-Layer Perceptron	98.69	0.9869	0.15
Random Forest Classifier	99.12	0.9912	0.17
CatBoost Classifier	98.44	0.9844	0.28
CatBoost Pruned	94.56	0.9670	0.32
Support Vector Machine	93.37	0.9334	0.37
Histogram-based Gradient Boosting Pruned	96.19	0.9618	4.13
Histogram-based Gradient Boosting	99.44	0.9944	5.08
LightGBM Pruned	95.81	0.9581	5.67
XGBoost Pruned	94.56	0.9452	6.71
XGBoost Classifier	99.12	0.9912	9.62
LightGBM Classifier	99.31	0.9931	9.61

Table 7. Model Resource Usage with Frame Skipping

Model	Avg FPS	Avg CPU (%)	Avg RAM (MB)	Avg GPU (%)
Random Forest Pruned	31.66	0.89	576.49	0.05
Bagging Classifier	31.64	0.98	565.81	0.06
Logistic Regression	31.86	1.19	571.85	0.05
Bagging Pruned	31.56	1.12	561.14	0.06
k-Nearest Neighbors	31.86	1.10	575.21	0.07
Multi-Layer Perceptron	31.88	1.05	572.28	0.08
Random Forest Classifier	31.74	1.02	581.87	0.09
CatBoost Classifier	31.75	0.98	582.51	0.09
CatBoost Pruned	31.82	1.32	567.65	0.07
Support Vector Machine	31.93	1.20	574.90	0.15
Histogram-based Gradient Boosting Pruned	31.04	29.74	569.70	0.07
Histogram-based Gradient Boosting	29.12	18.33	570.45	0.14
LightGBM Pruned	32.13	28.88	555.93	0.12
XGBoost Pruned	32.36	28.71	562.16	0.12
XGBoost Classifier	32.48	28.64	577.94	0.19
LightGBM Classifier	32.08	28.85	554.93	0.19

As can be seen in the 6, after applying the frame-skipping optimization technique, the initial model with the best score became a Bagging Classifier, with a score of 0.11, which is almost 3 times better than the score of the best initial model, k-nearest Neighbors with a score of 0.31. Moreover, the Pre-Pruned Random Forest was the best model overall. In addition, the CatBoost classifier model had the most noticeable improvement, with a score drop from 3.92 to 0.28, which is a drop of 92. 9%. This score decrease was caused by the drop in CPU usage from 6.36% to 0.98% and the increase in Average Frames Per Second from 15.98 to 31.75. The biggest improvement in absolute score was the Histogram Gradient Boosting model, with a score decrease from 16.86 to 5.08, which is a drop of 11.78 points. This decrease was due to a reduction of CPU usage from 60.49% to 29.74% and GPU usage from 0.14% to 0.07%. The CPU usage of all models has dropped significantly, averaging 3 times less CPU usage compared to the initial results, with the exception of LightGBM. GPU usage remained relatively the same for all models. Such steady usage can be explained by the fact that even though the program does not process the frame, it still grabs it, which requires GPU usage. Due to the same frame-grabbing reason, RAM usage also remained at nearly the same levels. Moreover, all Pre-Pruned models have scored better or, in the case of the CatBoost Classifier, nearly the same as initial models. Such results indicate that the Pre-Pruning optimization technique improves the trade-off between accuracy and resource usage of the computer.

4.5 Market Research

The Market Research phase will provide insights regarding specifications for average consumer-grade desktop computers and laptops. In 2024–2025, tasks performed on an average consumer-grade computer include web browsing, video streaming, and office work. According to the statistics covered in the following section, CPUs with integrated graphics or low- to mid-tier GPUs are used in such setups. The standard setup favors mid-tier GPUs and 6 to 8-core CPUs that provide a good balance between price and performance. Remarkably, many laptops rely on integrated graphics (GPU built into the CPU) for graphics unless a separate graphics card is needed for light gaming or multimedia acceleration. This is reflected in shipment data – the number of GPUs shipped annually exceeds the number of CPUs by approximately 24% (since almost every computer processor includes an integrated GPU). Overall, the average consumer-grade computer in 2024-2025 uses a mid-range Intel Core or AMD Ryzen CPU and low- to mid-tier GPU or integrated graphics.

4.5.1 Data Sources. To determine average specifications for the CPU and GPU of an average consumer-grade computer, the following sources were used:

- **PassMark CPU and GPU Benchmark Data:** This source collects benchmark submissions worldwide. In February 2025, PassMark [2] noted an unusual drop in average CPU performance year-on-year before recovering and achieving a 2.45% increase, potentially because more people were buying efficient, low-power CPUs for basic computers instead of high-performance chips. This drop implies that average customers are giving top priority to better price-to-performance ratios and adequate performance.
- **CPU-Z Validator Data:** Data from CPU-Z [3] submitted benchmarks show that in the first quarter of 2025, the most popular CPU configuration changed from 6-core to 8-core CPUs as the top choice (currently 25.6% vs 22.4%). This trend was mainly accelerated by the success of CPUs like AMD's Ryzen 7 9800X3D, which is now the most used CPU according to CPU-Z data, appearing in 4.9% of all submitted benchmarks. In addition to CPU data, the CPU-Z Validator also monitors installed GPUs. The top three GPUs installed in the benchmarked and submitted setups are NVIDIA GeForce RTX 4060, NVIDIA GeForce RTX 3060, and NVIDIA GeForce RTX 5080. NVIDIA GeForce RTX 4060 was used in 3.7% of benchmarked

[2] PassMark Year on Year CPU Performance: https://www.cpubenchmark.net/year-on-year.html
[3] CPU-Z Validator Statistics: https://valid.x86.fr/statistics.html

setups, gaining the most used GPU status. NVIDIA GeForce RTX 3060 or NVIDIA GeForce RTX 5080 were used in 3.4% of benchmarked computers, gaining second and third places in most used GPUs. Moreover, the most used RAM size is 32GB, with 37.7% benchmarks using this capacity of RAM. This data shows that the users combine an 8-core CPU with a mid-tier GPU that is on the same performance level as the CPU and 32GB RAM. In addition, 6-core CPUs combined with low- to mid-tier GPUs can still be found in older or cheaper setups.

- **Retail Sales for CPU, GPU and RAM:** Top-selling CPUs and GPUs on platforms like Amazon can give insights into the most used CPUs and GPUs in new builds. Of the 10 top-selling CPUs on Amazon, 7 have eight cores (as of July 2025), supporting the shift from the 6-core to 8-core CPUs indicated on CPU-Z benchmarks. Moreover, on Amazon's ratings, AMD's Ryzen 7 9800X3D is only the second best-selling CPU, contrary to data provided by CPU-Z. The top-selling CPU on the Amazon market is the AMD Ryzen 7 7800X3D, which supports data from PassMark CPU benchmark data that users give priority to better price-to-performance ratios. Additionally, the top-selling Intel CPU only ranks 13th in Amazon's top-selling CPU list. This Intel CPU is an Intel Core i7-12700KF with eight cores, which one more time supports the shift from 6-core CPUs to 8-core CPUs. As for GPU sales, the top-selling GPU is NVIDIA GeForce RTX 3050, which is once again contrary to CPU-Z data and supports the claim that users favor low- to mid-tier GPUs that are on the same performance level as the most popular CPU models. In addition, the top-selling graphic cards list is dominated by NVIDIA, with only 3 out of 10 top-selling GPUs being produced by AMD. The top-selling AMD graphics card is RX 7600 XT, which is also a mid-tier graphics card that can be combined with a top-selling AMD mid-tier CPU, AMD Ryzen 7 7800X3D. Regarding RAM, the 32GB capacity dominates, with 6 out of 10 top-selling RAM sticks being 32 GB. Moreover, the top 4 selling RAM is 64GB, which possibly indicates the start of a shift towards an even bigger RAM capacity. The remaining three positions in the top 10 selling list are occupied by 8GB RAM that is still used in budget computer setups.
- **Steam Hardware Survey:** While Amazon's top-selling CPUs and GPUs provide insights into what is popular in the new setups, hardware surveys like Steam Hardware Survey can provide an understanding of what is popular in the current setups. As it can be seen in the Steam Hardware Survey from May 2025, the most popular CPU core number is 6, meaning that even though newer setups have shifted from 6 cores to 8 cores, the overall usage could not yet shift. However, 8-core CPUs are not far from reaching 6-core popularity because the 8-core CPUs have gained 1.93% in popularity from January 2025, being the biggest gainer and being used in 24.01% of setups, while 6-core CPUs have faced the most significant decline in popularity, losing 1.37% in popularity from January dropping from being used in 32.04% of setups to 30.67%. Regarding graphic cards, the most used graphics card, according to Steam Hardware Support, was NVIDIA GeForce RTX 3060, which is slightly better than NVIDIA GeForce RTX 3050 but still in the mid-tier graphics cards list and can be paired with all previously listed popular mid-tier CPUs. Moreover, the dominance of the NVIDIA graphics cards is more evident in the Steam Hardware Survey, with the first non-NVIDIA graphics being only in 13th place. This place is taken by AMD Radeon Graphics, which is not a graphics card but an AMD Integrated Graphics. The first AMD graphics card is only in 29th place. In addition, the integrated graphics also has taken a noticeable percentage of usage. Both AMD and Intel Integrated Graphics take around 4% of usage each. Concerning RAM, 16 GB is still the most used RAM capacity, with a usage of 43%. However, this can change as the 16 GB RAM capacity has lost 0.86% of usage in the last month while the 32GB capacity has gained 0.16%, resulting in 33.75% of usage.
- **Retail Sales for Laptops:** As shown in the data from the Steam Hardware Survey, even among gamers, 8% of them use laptops with integrated graphics, and even more people have laptops in their homes and use laptops on a day-to-day basis. Therefore, a laptop inclusion in data had to be done. Data was gathered by accessing the Amazon platform and searching for laptops filtered by best-selling. All laptops were divided into four categories: High-End Laptops with Integrated Graphics, Budget Laptops with Integrated Graphics, High-End Laptops with Dedicated Graphics, and Budget Laptops with Dedicated Graphics. Example devices for each category were selected as follows: On the Amazon website, the keyword 'laptop' was searched with the sorting order of best-selling devices. For each category, specific filters were applied: for budget laptops, the price range was between 200 and 400 dollars; for high-end laptops, the price was between 700 and 1000; with integrated graphics or dedicated graphics card; AMD processor and graphics card or Intel processor and NVIDIA graphics card. In total, eight different laptops were selected. In addition to laptops, four additional sample desktop computer setups were created, high-end or budget versions and a combination of AMD processor and graphics card or Intel processor and NVIDIA graphics card. The most popular processors and graphics cards of the needed segment were selected for these combinations. In total, twelve sample devices were determined, as can be seen in the 8.

Table 8. Category and Example Device Overview

Category	Example Device
High-End Integrated (Intel)	ASUS Zenbook 14
High-End Integrated (AMD)	ASUS Zenbook 14 OLED
Budget Integrated (Intel)	HP 14" Ultra Light
Budget Integrated (AMD)	Acer Aspire 3
High-End Dedicated (Intel + NVIDIA)	Acer Nitro V
High-End Dedicated (AMD + AMD)	ASUS TUF Gaming A16
Budget Dedicated GPU (Intel + NVIDIA)	HP Victus 15.6" i5 Gaming Laptop
Budget Dedicated GPU (AMD + AMD)	HP Victus 15.6" Ryzen 5
High-End Desktop (Intel + NVIDIA)	Intel i7-13700K + NVIDIA RTX 4070 Ti
High-End Desktop (AMD + AMD)	AMD Ryzen 7 7800X3D + Radeon RX 7900 XT
Budget Desktop (Intel + NVIDIA)	Intel i5 + NVIDIA GTX 1650
Budget Desktop (AMD + AMD)	AMD Ryzen 5 5600G + Radeon RX 6500 XT

Specification of each sample device can be seen in 9. For each sample device, its CPU, GPU and RAM capacity is provided.

Table 9. Device Specifications Overview

Device & Category	CPU	GPU	RAM
High-End Integrated (Intel)	Intel Core i5-1240P	Intel Iris Xe	16 GB
High-End Integrated (AMD)	AMD Ryzen 7 7730U	AMD Radeon Vega 8	16 GB
Budget Integrated (Intel)	Intel N4120	Intel UHD 600 iGPU	8 GB
Budget Integrated (AMD)	AMD Ryzen 3 7320U	AMD Radeon 610M iGPU	8 GB
High-End Dedicated (Intel + NVIDIA)	Intel Core i7-13620H	NVIDIA RTX 4050	16 GB
High-End Dedicated (AMD + AMD)	AMD Ryzen 7 7735HS	AMD Radeon RX 7600S	16 GB
Budget Dedicated GPU (Intel + NVIDIA)	Intel Core i5-12450H	NVIDIA GTX 1650	16 GB
Budget Dedicated GPU (AMD + AMD)	AMD Ryzen 5 7535HS	AMD Radeon RX 6550M	16 GB
High-End Desktop (Intel + NVIDIA)	Intel Core i7-13700K	NVIDIA RTX 4070 Ti	32 GB
High-End Desktop (AMD + AMD)	AMD Ryzen 7 7800X3D	AMD Radeon RX 7900 XT	32 GB
Budget Desktop (Intel + NVIDIA)	Intel Core i5-12400F	NVIDIA GTX 1650	16 GB
Budget Desktop (AMD + AMD)	AMD Ryzen 5 5600G	AMD Radeon RX 6500 XT	16 GB

4.6 Resource Usage of Average Computers

In this subsection, an estimation of the model's resource usage of average computers, which was defined in the Market Research subsection, will be covered. For the estimation of resource usage, two models will be used: the k-Nearest Neighbors Classifier as the best non-optimized model and the Random Forest Classifier with Pre-Pruning and Frame Skipping as the best model with optimization techniques usage. Model Resource Usage that was covered in both Benchmarking and Optimization sub-sections was created by benchmarking models on the Intel i5 12600KF processor with a base clock of 3.7Ghz and 10 cores, AMD RX 6900 XT graphics card with 23 TFLOPS, and 32GB of RAM. The processor's clock speed and number of cores, TFLOPS of a graphics processor, and capacity of RAM for each sample computer can be seen in the 10.

Table 10. Device Specifications Overview

Device & Category	CPU N of Cores & GHz	GPU TFLOPS	RAM
High-End Integrated (Intel)	12 cores, 1.7 GHz	1.7 TFLOPS	16 GB
High-End Integrated (AMD)	8 cores, 2.0 GHz	2.0 TFLOPS	16 GB
High-End Dedicated (Intel + NVIDIA)	10 cores, 2.4 GHz	12.5 TFLOPS	16 GB
High-End Dedicated (AMD + AMD)	8 cores, 3.2 GHz	21 TFLOPS	16 GB
Budget Dedicated GPU (Intel + NVIDIA)	8 cores:, 2.0 GHz	3.0 TFLOPS	16 GB
Budget Dedicated GPU (AMD + AMD)	6 cores, 3.3 GHz	4.5 TFLOPS	16 GB
Budget Integrated (Intel)	4 cores, 1.1 GHz	0.25 TFLOPS	8 GB
Budget Integrated (AMD)	4 cores, 2.4 GHz	0.6 TFLOPS	8 GB
Budget Desktop (Intel + NVIDIA)	6 cores, 2.5 GHz	3.0 TFLOPS	16 GB
Budget Desktop (AMD + AMD)	6 cores, 3.9 GHz	5.7 TFLOPS	16 GB
High-End Desktop (Intel + NVIDIA)	16 cores, 3.4 GHz	40 TFLOPS	32 GB
High-End Desktop (AMD + AMD)	8 cores, 4.2 GHz	61 TFLOPS	32 GB

The following formulas have been used to calculate the adjusted resource usage for each sample computer.

$$\text{Adjusted CPU Usage} = \text{Tested computer CPU Usage} \times \frac{\text{N Cores}_{\text{Tested}} \times \text{Clock Speed}_{\text{Tested}}}{\text{N Cores}_{\text{Sample}} \times \text{Clock Speed}_{\text{Sample}}}$$

$$\text{Adjusted GPU Usage} = \text{Tested computer GPU Usage} \times \frac{\text{TFLOPS}_{Tested}}{\text{TFLOPS}_{Sample}}$$

$$\text{Adjusted RAM Usage} = \frac{\text{Tested computer RAM Usage}}{\text{RAM Capacity of Sample computer}} \times 100$$

The adjusted resource usage for the sample computers of the k-Nearest Neighbors model without Frame Skipping is provided in 11. The resource usage of the Pre-Pruned Random Forest model with Frame Skipping can be seen in the 12.

Table 11. Adjusted Resource Usage for k-Nearest Neighbors

Device & Category	CPU (%)	GPU (%)	RAM (%)
High-End Integrated (Intel)	6.22	0.68	3.52
High-End Integrated (AMD)	7.93	0.58	3.52
High-End Dedicated (Intel + NVIDIA)	5.29	0.09	3.52
High-End Dedicated (AMD + AMD)	4.96	0.05	3.52
Budget Dedicated GPU (Intel + NVIDIA)	7.93	0.38	3.52
Budget Dedicated GPU (AMD + AMD)	6.41	0.26	3.52
Budget Integrated (Intel)	28.84	4.60	7.04
Budget Integrated (AMD)	13.22	1.92	7.04
Budget Desktop (Intel + NVIDIA)	8.46	0.38	3.52
Budget Desktop (AMD + AMD)	5.42	0.20	3.52
High-End Desktop (Intel + NVIDIA)	2.33	0.03	1.76
High-End Desktop (AMD + AMD)	3.78	0.02	1.76

Table 12. Adjusted Resource Usage for Pre-Pruned Random Forest with Frame Skipping

Device & Category	CPU (%)	GPU (%)	RAM (%)
High-End Integrated (Intel)	1.61	0.68	3.52
High-End Integrated (AMD)	2.06	0.58	3.52
High-End Dedicated (Intel + NVIDIA)	1.37	0.09	3.52
High-End Dedicated (AMD + AMD)	1.29	0.05	3.52
Budget Dedicated GPU (Intel + NVIDIA)	2.06	0.38	3.52
Budget Dedicated GPU (AMD + AMD)	1.66	0.26	3.52
Budget Integrated (Intel)	7.48	4.60	7.04
Budget Integrated (AMD)	3.43	1.92	7.04
Budget Desktop (Intel + NVIDIA)	2.20	0.38	3.52
Budget Desktop (AMD + AMD)	1.41	0.20	3.52
High-End Desktop (Intel + NVIDIA)	0.61	0.03	1.76
High-End Desktop (AMD + AMD)	0.98	0.02	1.76

As shown in the 11, processor usage of the model is high for some of the setups, especially for the Intel-based setup with Integrated Graphics. In this setup, the model is estimated to take 28.84% of the processor power, which is almost 1/3 of all processing power and cannot be dedicated to the background-run application. Moreover, only five models use less than 6% of the overall processing power. As for GPU usage, the metrics show much better results, with the

model's expected GPU use being higher than 1% only in 2 sample computers, both with the type of Budget Laptop with Integrated Graphics. Such a tendency also can be seen in the model's expected RAM usage. RAM usage is expected to be higher than 4% only in the two aforementioned sample computers.

Moving on to the Pre-Pruned Random Forest model with Frame Skipping, it has shown much better results in usability because of the initial less use of resources on the testing computer. The only expected CPU computational power usage that is higher than 6% is at an Intel-based setup with Integrated Graphics with an expected usage of 7.48%. All of the remaining sample computers' expected usages average 1.7%, which is appropriate for the background application. In addition, High-End Desktops' expected computational power usage is even less than 1%. As for expected GPU usage, it is mainly less than 1% with a mean of 0.77%. The only sample computers that have expected GPU usage of more than 1% are Budget Laptops with Integrated Graphics, which the low computational capabilities of the integrated graphics in budget computers can explain. Concerning RAM, expected usage remained the same as for the K-Nearest Neighbors model, as the RAM needed for both models is almost the same.

5 Conclusion

After model creation, benchmarking, and projecting the processing power used on the sample average computers, it is possible to evaluate the feasibility of using gesture recognition models for computer control on the average consumer-grade computer. The first finding is that usage of the MediaPipe Hands significantly improves the accuracy of models, from 28% without MediaPipe Hands usage to the accuracy level of 99.44% for the most accurate model, the Histogram-based Gradient Boosting model. Hence, the usage of the MediaPipe Hands is recommended for the creation of such algorithms. Moving onto resource usage of the models, as can be seen from the projection of the initial models on the sample consumer-grade computers, even the best model without optimization can take up a lot of resources of budget desktop computers and all laptops. Therefore, the use of optimization techniques like frame skipping makes it possible to reduce the computational requirements of the models and make them better suited for real-time use. In addition, pre-pruning of the models can be done to reduce the models' size but with a trade-off in accuracy. Pre-pruning can be helpful in cases where the prediction model size should be as tiny as possible, for example, for edge computing environments. After the application of the aforementioned optimization techniques to the models, the computational power needed for the gesture recognition models can be considered feasible for real-time usage on the majority of consumer-grade computers, except the budget laptop versions, which do not have powerful enough processors and graphics to support seamless usage fo the gesture-recognition models without influence on users' computer usability.

6 Future Work

6.1 Optimization Techniques

More optimization techniques and their influence on models' resource usage, accuracy, and size can be investigated to improve performance and feasibility further. Especially further investigation of optimization techniques that can reduce gesture recognition models' RAM usage is recommended, as even already optimized models use 3.81% of RAM on average, which is the worst performance metric of the optimized models.

6.2 Model-Only Graphics Card Benchmarking

Due to limitations, the usage of graphics card resources used solely by the models was impossible to test. Even though the benchmarking was aimed to ensure that graphics card resources were used only by the model during benchmarking, some unintended use of graphics card resources was possible. Model-only graphics card usage benchmarking will provide a clearer understanding of the models' GPU usage and may demonstrate feasible usage even on the sample computers that are currently considered unfeasible.

6.3 Energy Consumption Analysis

The research did not cover energy consumption analysis, which can provide insights into the influence of the gesture recognition models' usage on laptop battery life. Significant battery consumption on laptops can determine gesture recognition models as unfeasible for usage on laptops.

6.4 Benchmarking On Different Systems

Benchmarking gesture-recognition models on computers with different levels of hardware, from low- to high-end hardware, instead of the resource usage projection based on the hardware capabilities, can provide a more comprehensive understanding of the feasibility of gesture-recognition models' usage on different hardware.

6.5 User-Centered Testing

User-centred testing of gesture-recognition models and usage of gestures as a way of human-computer interaction can provide insights into the possibility of the potential adoption of hand gestures as a new, commonly used way of human-computer interactions. Moreover, such studies can show which maximum prediction time is acceptable for users without causing discomfort.

6.6 Machine Learning Algorithms

Testing of additional machine learning algorithms can provide data about more suited machine learning algorithms for gesture recognition tasks. In addition, such testing can identify more balanced and optimized algorithms that will require less computational power of the computer and increase the number of hardware that can run gesture recognition models in the background without influencing users' computer usability.

7 AI Acknowledgment

During the research creation ChatGPT 4o was used to support grammatical correction, typographical accuracy, and the refinement of academic language. The author takes full responsibility for the content of this work.

References

[1] Jonathan Alon, Vassilis Athitsos, and Stan Sclaroff. 2005. Accurate and efficient gesture spotting via pruning and subgesture reasoning. In *Computer Vision in*

Human-Computer Interaction: ICCV 2005 Workshop on HCI, Beijing, China, October 21, 2005. Proceedings. Springer, 189–198.

[2] M. Pal and. 2005. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing* 26, 1 (2005), 217–222. doi:10.1080/01431160412331269698 arXiv:https://doi.org/10.1080/01431160412331269698

[3] Noor Adnan Ibraheem and RafiqulZaman Khan. 2012. Survey on various gesture recognition technologies and techniques. *International journal of computer applications* 50, 7 (2012), 38–44.

[4] Sheng-Hung Jeng and Chien-Ho Tseng. 2001. A model-based hand gesture recognition system. *Machine Vision and Applications* 12, 5 (2001), 243–258. doi:10.1007/s001380050144

[5] Daniel Jurafsky and James H. Martin. 2023. Speech and Language Processing, Third Edition (Draft). Available at https://web.stanford.edu/~jurafsky/slp3/5.pdf.

[6] Agnes Just and Sébastien Marcel. 2009. A comparative study of two state-of-the-art sequence processing techniques for hand gesture recognition. *Computer Vision and Image Understanding* 113, 4 (2009), 532–543. doi:10.1016/j.cviu.2008.12.001

[7] Ehsan Kamalloo, Aref Jafari, Xinyu Zhang, Nandan Thakur, and Jimmy Lin. 2023. HAGRID: A Human-LLM Collaborative Dataset for Generative Information-Seeking with Attribution. arXiv:2307.16883 [cs.CL] https://arxiv.org/abs/2307.16883

[8] N. Lalithamani. 2016. Gesture control using single camera for PC. *Procedia Computer Science* 78 (2016), 146–152. doi:10.1016/j.procs.2016.02.024

[9] Yanqiu Liu, Xiuhui Wang, and Ke Yan. 2018. Hand gesture recognition based on concentric circular scan lines and weighted K-nearest neighbor algorithm. *Multimedia Tools and Applications* 77 (2018), 209–223.

[10] Kristína Machová, Frantisek Barcak, and Peter Bednár. 2006. A bagging method using decision trees in the role of base classifiers. *Acta Polytechnica Hungarica* 3, 2 (2006), 121–132.

[11] Noraini Mohamed, Mumtaz Begum Mustafa, and Nazean Jomhari. 2021. A Review of the Hand Gesture Recognition System: Current Progress and Future Directions. *IEEE Access* 9 (2021), 157422–157436. doi:10.1109/ACCESS.2021.3129650

[12] Chan-Wah Ng and Surendra Ranganath. 2002. Real-time gesture recognition system and application. *Image and Vision Computing* 20, 13-14 (2002), 993–1007. doi:10.1016/S0262-8856(02)00113-0

[13] Munir Oudah, Ali Al-Naji, and Javaan Chahl. 2020. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques. *Journal of Imaging* 6, 8 (2020), 73. doi:10.3390/jimaging6080073

[14] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. 2009. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems* 8, 7 (2009), 579–588.

[15] Jeong-Jik Seo, Hyung-Il Kim, Wesley De Neve, and Yong Man Ro. 2017. Effective and efficient human action recognition using dynamic frame skipping and trajectory rejection. *Image and Vision Computing* 58 (2017), 76–85.

[16] Tsung-Han Tsai, Chih-Cheng Huang, and Kai-Lung Zhang. 2020. Design of hand gesture recognition system for human–computer interaction. *Multimedia Tools and Applications* 79, 9–10 (2020), 5989–6007. doi:10.1007/s11042-019-08274-w

[17] Pei Xu. 2017. A Real-time Hand Gesture Recognition and Human-Computer Interaction System. doi:10.48550/arXiv.1704.07296 arXiv:1704.07296 [cs.CV]