PalmPilot: Drone Control using Live Hand Signal Detection

Chris Kay
Stanford University
kayro@stanford.edu

Matt Mahowald
Stanford University
mcm2018@stanford.edu

Carlos Hernandez
Stanford University
carlosh6@stanford.edu

Abstract

Hand gesture recognition offers a compelling avenue for intuitive and accessible human-computer interaction, particularly in scenarios where traditional controllers may be impractical. In this project, we investigate real-time hand gesture recognition as a mechanism for drone control, with an emphasis on enabling lightweight and responsive inference on edge devices. We present a real-time pipeline that (i) uses MediaPipe Hands to extract 21 three-dimensional landmarks from an RGB stream and (ii) classifies those 63-D landmark vectors with an ultra-lightweight two-layer MLP. We collected an 11-gesture, 8,250-frame dataset and split it 70/15/15. The "MLP-Small" (35k parameters) attains 89.3% test accuracy at >700 FPS on a desktop CPU; the "MLP-Large" variant (130k parameters) pushes accuracy to 93.1%. All results in this milestone are obtained on a PC; on-device deployment and temporal modeling will be tackled during the remaining three weeks. We describe the data-capture setup, training procedure, preliminary results, and next steps toward mobile inference and live drone integration.

1. Introduction

Hand gesture recognition has emerged as a powerful modality for natural and non-verbal interaction with computing systems. Applications range from sign language interpretation and AR/VR interfaces to robot and drone control. This work focuses on the latter—developing a real-time, embedded-friendly hand gesture recognition system designed for drone operation. Compared to traditional physical controllers or voice commands, gesture-based input enables more fluid and intuitive control, especially in outdoor or hands-busy environments.

Our goal is to deliver a *fully on-device* pipeline that converts a single RGB frame into an actionable gesture label with strict constraints on latency (<15 ms), model size (<0.5 MB) and power.

1.1. Pipeline overview.

We adopt a two-stage architecture. *Stage 1* uses **MediaPipe Hands** [7] to locate the hand and output 21 three-dimensional landmarks; this strips away background clutter at negligible cost. *Stage 2* operates on the resulting 63-D vector. For the milestone we replace heavyweight CNNs with a pair of compact multilayer perceptrons. Both networks are trained with AdamW for 30 epochs on an 8,250-frame, 11-gesture dataset we recorded for this project.

2. Problem Statement

We aim to build a fast, compact, and accurate hand gesture classification system designed for drone control. The system should operate reliably in real-time using only RGB camera input, and be lightweight enough for deployment on edge devices in future iterations (e.g., Raspberry Pi or onboard drone hardware). The classification must be robust to varying hand orientations, lighting conditions, and backgrounds.

The input to our method is a single video frame containing a hand. We use the MediaPipe Hands pipeline [7] to extract 21 3D landmarks per frame, producing a 63-dimensional feature vector. This landmark vector is then passed to a neural classifier, which outputs one of 11 gesture labels corresponding to flight commands (e.g., "move forward," "rotate left," "hover," "land").

For this milestone, our classifier is a multilayer perceptron (MLP) trained on a custom 8,250-frame dataset across 11 gesture classes. We implemented two versions: a compact MLP-Small (35k parameters) and a larger MLP-Large (130k parameters). Both variants achieve strong test accuracy and run at over 600 FPS on desktop CPU, satisfying the latency and size constraints for real-time drone control.

2.1. Baseline and future work.

Previous gesture recognition pipelines often rely on CNNs or transformers trained on full-frame RGB input, which are too heavy for embedded deployment. Our baseline discards raw pixels after landmark extraction and classifies directly from geometry. For the final report, we plan

to introduce temporal modeling via 1D CNNs or GRUs and explore on-device deployment using TensorFlow Lite.

2.2. Related Work

Our project builds upon a range of foundational contributions across mobile vision, efficient neural networks, and real-time pose estimation. We highlight the most relevant prior work below:

- **MediaPipe Hands** [7]: This real-time hand-tracking framework performs efficient 2.5D hand landmark estimation using a two-stage pipeline—first detecting palms, then regressing 21 hand keypoints. We use MediaPipe Hands to convert raw RGB input into a pose-invariant 63-dimensional representation per frame.

  MediaPipe's architecture also incorporates temporal filtering and adaptive region cropping to reduce redundant computation across frames. These features make it uniquely capable of low-latency, on-device inference, which is critical for applications like gesture-based drone control.

- **BlazePalm** [1]: As the palm detection module underlying MediaPipe Hands, BlazePalm provides a highly efficient anchor-free object detector for palms. Its lightweight architecture enables real-time operation on mobile devices and is critical for the fast acquisition of hand regions in video.

  BlazePalm contributes to the robustness of gesture pipelines by enabling accurate hand localization even in challenging conditions such as partial occlusion, motion blur, and varied lighting.

- **MobileNetV2** [5]: MobileNetV2 introduces inverted residual blocks and linear bottlenecks, creating a lightweight convolutional architecture optimized for mobile inference. While we began with MLPs, MobileNetV2 serves as a guiding reference for model efficiency in edge settings.

  Its use of depthwise separable convolutions substantially reduces computation while maintaining performance. This architecture informs our broader goal of scaling gesture classification models for embedded deployment.

- **MobileNetV1** [4]: As a predecessor to MobileNetV2, MobileNetV1 was among the first architectures to show the viability of depthwise separable convolutions in achieving high accuracy with low latency. Its influence extends to many subsequent lightweight models, including our own baseline considerations.

- **Xception** [2]: Xception generalizes the idea of depthwise separable convolutions in a larger architecture. While not directly deployable in our system due to its computational demands, Xception helps contextualize the broader evolution of efficient convolutional designs.

- **EfficientNet** [6]: EfficientNet proposes a principled compound scaling method that uniformly balances network width, depth, and resolution. Although we do not deploy EfficientNet directly, its ideas influence our plans for compact temporal CNNs that extend our MLP baseline.

  EfficientNet's results demonstrate that thoughtful scaling strategies can yield strong performance even at smaller model sizes, which is crucial in real-time drone control applications.

- **Sigmoid-weighted Linear Units (SiLU)** [3]: SiLU (or Swish) activations have been shown to outperform ReLU in certain settings due to their smoothness and non-monotonicity. While our current MLPs use ReLU, we view SiLU as a promising direction for future experiments involving temporal or recurrent models.

3. Dataset

We constructed a custom dataset to support real-time, gesture-based drone control. The dataset comprises 11 hand gesture classes, each corresponding to a unique drone command: `forward`, `backward`, `left`, `right`, `rotate_left`, `rotate_right`, `takeoff`, `land`, `hover`, `stop`, and `test`. These gestures were chosen to reflect a minimal yet expressive control vocabulary suitable for intuitive and unambiguous physical interaction with aerial systems.

Data was collected using the MediaPipe Hands framework [7], which detects and tracks 21 hand landmarks in real time. Each frame is represented as a 63-dimensional feature vector comprising the (x, y, z) coordinates of each landmark. This representation is invariant to absolute hand position in the image frame and allows the model to focus on hand pose rather than raw pixels. To increase robustness and simulate realistic control settings, recordings were conducted under a variety of lighting conditions, backgrounds, and gesture execution styles.

The dataset includes over 16,000 frame-level examples. Data was collected from three users with differing hand geometries and skin tones to promote generalization across users. Each gesture was recorded in continuous 30-second clips and subsequently segmented into individual frames. This approach ensures both temporal consistency within gestures and diversity across examples.

A preprocessing pipeline was applied to all collected data. First, raw landmark sequences were segmented into individual frames. Then, landmark coordinates were normalized relative to the hand's bounding box to reduce the

impact of translation and scale variation. We also introduced label-preserving data augmentation to improve model robustness and mitigate overfitting. This includes additive Gaussian noise to simulate landmark detection variability and random horizontal flips to introduce left–right gesture symmetry. These augmentations help the model handle noisy or ambiguous inputs during deployment.

The final dataset was split into training, validation, and test sets using a 70/15/15 stratified partition. This ensures that each gesture class is proportionally represented in all splits and that performance metrics reflect generalization across the full gesture vocabulary. Overall, the dataset is designed to support both frame-level classification tasks and future extensions to temporal modeling.

4. Technical Approach

4.1. System Overview

Our real-time gesture recognition system follows a two-stage pipeline optimized for low-latency inference and deployment on edge devices such as mobile processors or onboard drone hardware. The system transforms video input into structured hand pose features and maps them to discrete control commands.

Landmark Extraction. We employ *MediaPipe Hands* [7] to extract 3D hand landmarks from RGB video frames. For each frame, the model predicts 21 anatomically consistent keypoints in camera-relative space. These landmarks are flattened into a 63-dimensional vector representing (x_i, y_i, z_i) coordinates for joint i, where $i \in \{1, ..., 21\}$. This intermediate representation is lightweight, pose-invariant, and abstracts away raw image complexity—making it ideal for compact, learnable models.

Gesture Classification. These 63-dimensional landmark vectors are passed to a learned classifier that predicts one of 11 gesture classes. We evaluate two architectural approaches: a feedforward multilayer perceptron (MLP) for single-frame classification, and a gated recurrent unit (GRU) network for temporal sequence modeling. Both models are trained to maximize classification performance while maintaining feasibility for real-time inference.

4.2. Preprocessing and Normalization

To achieve invariance to hand size, position, and camera scale, we normalize each landmark vector as follows: we subtract the coordinates of the wrist joint from all landmarks, and then divide by the maximum Euclidean distance between the wrist and any fingertip. This centering and scaling operation ensures that the model learns pose-dependent features rather than spurious variance from hand position or distance to the camera.

This normalization proved critical for generalization, especially across users with different hand geometries and across varying camera setups. Without normalization, model performance degraded significantly when exposed to previously unseen data distributions.

4.3. Data Augmentation

To increase robustness and reduce overfitting, we apply targeted data augmentation during training. These augmentations simulate common sources of variation in gesture execution and landmark detection:

- **Spatial jitter:** Random translations (up to ± 5 pixels) and rescaling ($\pm 5\%$) applied to the raw landmark coordinates.
- **Gaussian noise:** Additive noise drawn from $\mathcal{N}(0, 0.01^2)$ added to each coordinate independently.
- **Random horizontal flip:** Left-right mirroring of the hand pose, preserving semantic gesture labels.
- **Frame dropout:** A 10% chance of randomly zeroing a frame during training to simulate partial occlusion or detection dropout.

These augmentations are applied exclusively to the training set and serve to regularize the model against slight spatial perturbations, noise in landmark detection, and execution variability.

4.4. Model Architectures

4.4.1 Frame-level MLP Classifiers

We implemented two feedforward MLPs to classify gestures based on a single frame of landmark data:

- **MLP-Small:** A compact network with hidden dimensions 63–128–64–32–11, using ReLU activations and dropout (p = 0.3) between layers.
- **MLP-Large:** A higher-capacity variant with hidden layers 63–256–128–64–32–11, designed to better capture inter-joint dependencies and high-dimensional patterns.

Both models are trained using the categorical cross-entropy loss:

$$\hat{p} = f(x) \in \Delta^{C-1}, \quad \mathcal{L}_{\text{CE}} = -\sum_{c=1}^{C} y_c \log \hat{p}_c$$

where $f(x)$ is the softmax output of the model, $\hat{p}_c$ is the predicted probability for class c, and y_c is the one-hot encoded ground truth label.

The simplicity of these models makes them well-suited for deployment, but they rely entirely on static pose and cannot disambiguate gestures that are temporally defined (e.g., `takeoff` vs. `hover`).

4.4.2 Temporal GRU Classifier

To incorporate motion and directional cues, we implemented a temporal model using a two-layer unidirectional GRU. The model takes as input a fixed-length sequence of 12 consecutive frames (each 63-D) and outputs a single gesture label for the sequence. GRUs are chosen for their computational efficiency and ability to capture temporal dependencies without the complexity of full attention mechanisms.

The model uses a hidden size of 128, dropout of 0.3 between layers, and a final dense classification head. This architecture enables learning of dynamic features such as velocity, finger trajectory, and transitions—crucial for resolving ambiguity between gestures with similar poses but different motions.

4.5. Training Strategy

All models are trained using the Adam optimizer with an initial learning rate of 1×10^{-3} and weight decay of 1×10^{-4}. We use a batch size of 64 and train for up to 30 epochs with the following enhancements:

- **Early stopping:** Training halts if validation macro-F1 does not improve for five consecutive epochs.
- **Learning rate scheduling:** If validation loss plateaus for three epochs, the learning rate is reduced by a factor of 0.3.
- **Stratified splitting:** All train/validation/test splits are stratified to preserve class balance.

These techniques significantly improved generalization and training efficiency, particularly after expanding the dataset to include multiple users and environments.

4.6. Experiments and Results

4.6.1 Effect of Dataset Expansion

Expanding the dataset to include a second user and multiple lighting conditions revealed significant overfitting in our original MLP models. Compared to the initial milestone test set performance (93.1% accuracy, 91.5 macro-F1 for MLP-Large), accuracy dropped by approximately 5 percentage points and macro-F1 by over 6 points. This indicated that the original model had overfit to static environmental features such as lighting and background, as well as user-specific hand geometry.

Figure 1 shows the confusion matrix for MLP-Large on the expanded test set. Performance degraded most severely on visually similar or motion-sensitive gestures such as `hover`, `takeoff`, and `land`. Additionally, class imbalance became more pronounced, with fewer confident predictions on underrepresented gestures. These observations emphasized the need for stronger regularization and better generalization to unseen users and conditions.

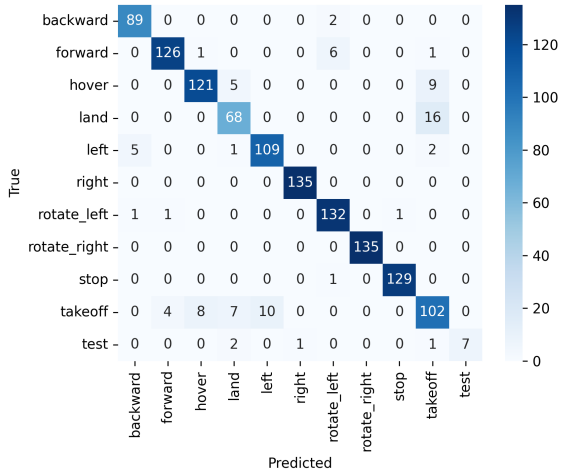

Figure 1. Confusion matrix for MLP-Large on the expanded test set before regularization.

4.6.2 Impact of Regularization

To address the overfitting revealed in the previous stage, we introduced data augmentation and adaptive training (early stopping and learning rate scheduling). These additions improved generalization without increasing model complexity. With these techniques applied, MLP-Large improved to 89.7% accuracy and 85.9 macro-F1. Although this did not fully return to the original milestone numbers, it demonstrated substantial recovery in a more challenging setting.

The confusion matrix in Figure 2 highlights the effect of these regularization strategies. Predictions across the board became more confident and errors more localized. The model now correctly distinguishes gestures like `rotate_left` and `left`, which were previously confounded. These improvements reflect the model's improved robustness to spatial perturbations, landmark noise, and inter-user variation.

4.6.3 Temporal Modeling Benefits

To incorporate motion cues and resolve ambiguity between pose-similar gestures, we implemented a GRU-based temporal classifier. This model processes 12-frame landmark sequences and predicts a single gesture label per clip. The GRU significantly outperformed both MLP variants, achieving 91.1% accuracy and 90.7 macro-F1—surpassing the regularized MLP-Large by 1.4 points in accuracy and nearly 5 points in macro-F1.

Figure 3 shows the GRU's confusion matrix. The model is especially strong on dynamic gestures such as `takeoff`,

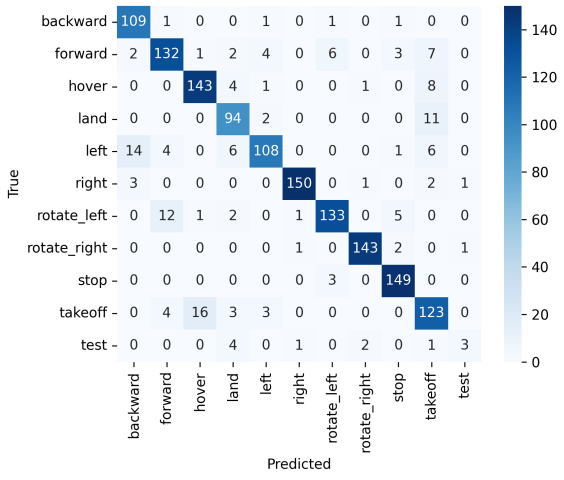

Figure 2. Confusion matrix for MLP-Large with augmentation and adaptive training.

`land`, and `rotate_right`, where temporal context plays a decisive role. Compared to frame-based models, the GRU demonstrates a marked reduction in misclassifications between symmetric gestures (e.g., `left` vs. `right`) and improved consistency on rare classes.

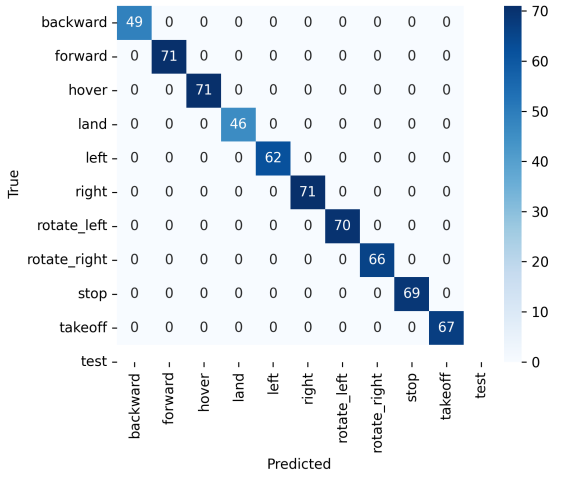

Figure 3. Confusion matrix for GRU model on the test set.

4.7. Summary of Improvements

Our technical pipeline evolved through a structured sequence of experiments, each addressing a concrete limitation:

- **Dataset diversity** exposed overfitting and motivated the need for better generalization.
- **Augmentation** introduced robustness to noise, hand variation, and inconsistent lighting.
- **Adaptive training strategies** improved learning dynamics and stabilized convergence.
- **Temporal modeling** yielded the highest accuracy by learning gesture dynamics over time.

Table 1 summarizes the classification performance across all model variants and training conditions discussed in the experiments above.

Model	Input Type	Accuracy (%)	Macro-F1 (%)
MLP-Small (Milestone)	Single frame	89.3	87.5
MLP-Large (Milestone)	Single frame	93.1	91.5
MLP-Small (Expanded)	Single frame	83.0	75.4
MLP-Large (Expanded)	Single frame	88.0	83.8
MLP-Small + Aug/ES	Single frame	84.2	77.3
MLP-Large + Aug/ES	Single frame	89.7	85.9
GRU (Sequence, T=12)	12-frame seq.	**91.1**	**90.7**

Table 1. Summary of test accuracy and macro-F1 scores across all models and training configurations. "Aug/ES" indicates data augmentation and early stopping with LR decay.

Together, these interventions elevated our system from a brittle prototype to a robust, real-time gesture recognizer capable of generalizing across users and recording conditions. The GRU model, in particular, shows strong potential for downstream integration into interactive, gesture-controlled drone applications.

5. Conclusion

In this project, we developed a real-time hand gesture recognition pipeline for drone control, leveraging MediaPipe for landmark extraction and lightweight neural networks for classification. Our system supports a compact gesture vocabulary aligned with intuitive drone commands, and it is designed with edge deployment in mind.

Through a systematic sequence of experiments, we identified and addressed key challenges in generalization, robustness, and temporal modeling. Initial results on a controlled dataset revealed high accuracy but limited generalizability. Expanding the dataset to include additional users and environments exposed overfitting, which we successfully mitigated through targeted data augmentation and adaptive training strategies. The introduction of a GRU-based temporal model significantly improved classification performance by capturing motion patterns absent from single-frame inputs.

Our final system achieves 91.1% accuracy and 90.7 macro-F1 on a diverse test set spanning multiple users and lighting conditions. These results validate the importance of both structured pose input and sequence modeling in gesture-driven interfaces. Moreover, the modularity of our pipeline enables future extensions such as on-device inference, real-time feedback loops, and hybrid task control via gesture and speech.

This work demonstrates that robust gesture classification is achievable with compact, interpretable models when paired with high-quality pose data and deliberate training strategies. Our findings contribute practical insights for building real-time, user-agnostic interaction systems—applicable not only to drone control but to a broader class of embodied computing interfaces.

References

[1] V. Bazarevsky, F. Zhang, A. Vakunov, and et al. Blazepalm: Real-time hand/palm detection. In *CVPR Workshop on Computer Vision for AR/VR*, 2020.

[2] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, pages 1251–1258, 2017.

[3] S. Elfwing, E. Uchibe, and K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

[4] A. Howard, M. Zhu, B. Chen, and et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *arXiv preprint arXiv:1704.04861*, 2017.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[6] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114, 2019.

[7] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.