GestureNet: Real-Time Hand Gesture Recognition on Edge Devices”

Megan Mulholland
Hood College, Frederick, Maryland
Email:
meganmulholland99gmail.com

***Abstract*—This project presents a real-time hand gesture recognition system using a custom convolutional neural network, GestureNet, trained to classify digits 0 to 9 from webcam images. The system pre-processes the input images using adaptive thresholding on grayscale images, with a block size of 11 and a constant of 2, and employs data augmentation, including rotation, zoom, shifts, and flips, to enhance generalization. The final TensorFlow model was trained on a balanced dataset of 10,000 images with early stopping and ReduceLROnPlateau to prevent overfitting, halting at 13 epochs. The model achieved 99% test accuracy with stable precision, recall, and F1-scores across all classes. The trained model was converted to TensorFlow Lite using FP32, dynamic-range, and full INT8 quantization. The TFLite model maintained 99% accuracy, with an average inference time of 0.70 ms per sample and a file size of 2.1 MB, demonstrating suitability for deployment on edge devices. Real-time testing with the webcam showed minor confusions primarily between similar finger positions, suggesting potential improvements in filter design and pre-processing. This work highlights the effectiveness of combining pre-processing, data augmentation, and quantization to produce a robust, high-performance gesture recognition system suitable for future deployment on devices such as the Raspberry Pi.**

I. Introduction

Hand gesture recognition has become increasingly crucial for Internet of Things (IoT) applications and accessibility devices, given that it can provide alternative interaction options for people who have disabilities or are part of the American Sign Language community. This project aims to develop a real-time gesture recognition system capable of accurately identifying digits using a lightweight convolutional neural network, with the option of converting it to an edge-friendly model. The system would be constrained to the capabilities of the edge device, such as limited processing power and memory. To create this system, I used adaptive image pre-processing, data augmentation, and early stopping to enhance accuracy while mitigating overfitting. The final model is converted to TensorFlow Lite for possible deployment on edge devices and demonstrates the possibility of success on resource constrained devices. Overall, this project created a robust hand gesture recognition system and can provide meaningful insights into quantization for edge applications.

II. Related Work

Hand gesture recognition is essential for IoT, accessibility, and human-computer interaction. Traditional methods rely on geometric features or classical machine learning, while CNNs offer higher accuracy but often require GPUs and large memory, limiting use on edge devices.

Yu et al. (2024) proposed a lightweight CNN with residual blocks for Raspberry Pi, using adaptive pre-processing and TensorFlow Lite quantization to achieve over 96% accuracy with low memory and energy use. Motion sensors further reduced power consumption by disabling recognition when no users were present.

My project builds on these ideas with GestureNet, a custom CNN trained on digit gestures. Using adaptive thresholding, data augmentation, and TensorFlow Lite quantization, the model achieves 99% accuracy and 0.70 ms per sample inference on a laptop CPU. This demonstrates that efficient, real-time gesture recognition is feasible on general-purpose hardware, not just edge devices.

III. Testbed Environment

A. Sensing

All gesture recognition experiments were conducted using the built-in webcam on my Lenovo Legion laptop.

- Camera: Webcam
- Resolution used: 224x224 after pre-processing
- Lighting Conditions Tested:
  - Normal indoor lighting
  - Black background
  - Low light environment (single lamp or dim room)

During runtime, each image was pre-processed to isolate the hand region and reduce noise.

B. Computing Environment

The model development, training, and TensorFlow Lite conversion were performed on a Lenovo Legion laptop with the following specifications:

- CPU: 13th Gen Intel® Core™ i5-13500H
- Integrated GPU: Intel Iris Xe Graphics
- Discrete GPU: NVIDIA GeForce RTX 4050 Laptop GPU
- RAM: 16 GB
- Operating System: Windows 11

Frameworks and Software Versions:

Python 3.10.18, TensorFlow 2.10.1, OpenCV 4.12.0.88, NumPy 1.26.4, PyImageSearch Conf, TFLite Converter

C. Demonstration

To test the gesture recognition pipeline, a real-time test was conducted with:

- Webcam feed displayed on the laptop screen
- ROI box
- Real-time model predictions displayed as text with the hand gesture image

IV. Approach

A custom convolutional neural network, GestureNet, was trained on hand gesture images for digits zero to nine. Pre-processing involved adaptive thresholding on the grayscale ROI using OpenCV's cv2.AdaptiveThreshold (block size 11, constant 2), producing a binary image highlighting the hand. Gaussian method was used for local thresholding to improve segmentation under varying lighting conditions. Data augmentation diversified the training dataset. The model was trained for up to 75 epochs, but early stopping halted training at 13 epochs to prevent overfitting. The trained TensorFlow model was converted to TensorFlow Lite (FP32, dynamic-range, full INT8), resulting in a 2.15MB model and 0.70ms per sample inference. Raspberry Pi testing was not performed, given that I did not have access to one.

V. Performance Evaluation

A. Methodology

The dataset included 10,000 images (1,000 per class) split 75:25 for training/testing. Data augmentation included rotation (±20°), zoom (±15%), width/height shifts (±20%), and horizontal flips. GestureNet was trained with batch size 8, initial learning rate 0.0001, Adam optimizer, categorical cross-entropy loss, and early stopping.

B. Results: TensorFlow Model

The final model achieved 0.99 accuracy, with precision, recall, and F1-scores near 1.0 for all digit classes. Early stopping prevented overfitting after 13 epochs.

C. Results: TensorFlow Lite Model

I converted the final trained Custom CNN model to TFLite (FP32) resulting in a model size of approximately 2.1MB and inference across 2,536 test samples took 1.78 seconds, an average of 0.70ms per sample on my desktop CPU using the XNNPACK delegate. The TFLite classification report closely mirrored the TensorFlow results, with an overall accuracy of 0.99 and insignificant differences in precision and recall for all classes. No significant accuracy drop after quantization, meaning that the conversion of the model did not reduce its ability to correctly classify the hand gestures.

D. Comparison

When experimenting throughout this project, I found that adaptive threshold was a lot better than fixed thresholding, since it allowed for the hand gestures to be more visible and have a clearer border. The addition of early stopping was critical for the mitigation of overfitting, given that even with data augmentation and a high validation accuracy, without the early stopping the model was overfitting. Overall, the TFLite conversion did not result in any measurable issues, showing that the model is still robust to FP32 quantization and could be deployed on edge devices.

E. Real-Time Testing Results

TABLE I
Real-Time Testing Accuracy and Confusions per Digit

Digit	Accuracy	Confusions
0	1.0	-
1	0.9	0
2	0.8	3
3	1.0	-
4	0.8	3
5	0.7	3
6	0.9	3
7	0.8	0
8	0.8	3
9	0.7	-

When doing real-time testing with the final model, it performed moderately well. I did ten tests per digit in a dimly lighted room with a black background. The model over-predicted 3 and occasionally 0 due to similar finger shapes.

VI. Obstacles and Lessons Learned

- **Overfitting:** I initially trained the model with GestureNet on grayscale images for 75 epochs, causing overfitting which was shown from the perfect training accuracy but changing validation metrics. To mitigate this issue, I used early stopping and ReduceLROnPlateau, which stabilized validation performance while maintaining high accuracy. This taught me how to recognize overfitting, even when it seems the model is doing well, and new methods to mitigate it.
- **Data:** When I first tested the model, I only used roughly 100 images per class, which severely limited the model's generalization. I expanded the dataset to contain 1,000 images per class and that minimized misclassifications tremendously and made my model more reliable.
- **Lighting:** When collecting images for the dataset I found that the lighting conditions were an important factor. I needed consistent light and preferably a darker background behind the hand. A noisy background or variations in brightness heavily degraded the ROI quality and prevented the model from having reliable data. I learned that pretesting the ROI filters and having the correct lighting for my model was very important.
- **TFLite Conversion:** When I was converting the model to TensorFlow Lite, I accidentally forgot to change the file name, which resulted in an error. Overlooking this change caused some delay but was corrected.

I learned how crucial careful pre-processing of the dataset and making adjustments for how the model is reacting during training is, as well as how TFLite compares to TensorFlow.

VII. Hyperparameter Tuning and Alternative Models

- Learning rate: 1e-3, batch size: 8, early stopping at 13 epochs.
- Data augmentation: rotation ±20°, zoom ±15%, width/height shifts ±20%, shear ±15%.
- Filters: Adaptive thresholding outperformed fixed thresholding.
- Alternative models: MobileNetV3Small achieved 10% validation accuracy; GestureNet chosen.
- Activation/Loss: ReLU and categorical cross-entropy retained.
- TFLite conversion maintained accuracy (0.70ms/sample, 2.1MB).

The best performing approach that I chose as the final model used adaptive threshold filtering, data augmentation, a custom GestureNet CNN with ReLU and cross-entropy, early stopping, and reduceLROnPlateau. This resulted in a stable and accurate model that was converted successfully to TFlite. Alternative activations, losses, and models were considered, but I do not choose them because of poor performance or lack of necessity.

VIII. Explaining TensorFlow and TFLite Accuracy Differences

I did not have access to a Raspberry Pi to test the TFLite on, so in my project both the TensorFlow model and TensorFlow Lite model had highly similar results. The TensorFlow model had approximately 99% accuracy and the TFLite model had a 99% accuracy as well. Therefore, my results showed no meaningful differences after quantization.

IX. Discussion and Future Work

The results of this project demonstrate that a custom CNN, when paired with the correct pre-processing pipeline, can achieve highly accurate hand-gesture recognition. After introducing early stopping and ReduceLROnPlateau, the final TensorFlow model reached 99% accuracy without overfitting. The TensorFlow Lite version also achieved 99% accuracy with an average inference time of 0.70 ms per sample, indicating strong suitability for real-time deployment on edge devices such as the Raspberry Pi.

Given that I did not have access to a Raspberry Pi during this project, I was unable to test real-time performance directly. In future work, I plan to deploy the TFLite model on a Raspberry Pi to evaluate latency, stability, and gesture recognition accuracy under real operating conditions. Additionally, improving robustness to non-optimal or low-light environments, potentially through alternative filters, adaptive exposure techniques, or data augmentation targeted at lighting variation, would further enhance the model's usability and generalization.

References

[1] A. Yu, C. Qian, and Y. Guo, "A Real-time Hand Gesture Recognition System on Raspberry Pi: A Deep Learning-based Approach," Poolesville High School, Hood College, and Towson University, 2024. [Online]. Available: Emails: xiaoyumd@gmail.com, qian@hood.edu, yguo@towson.edu

TABLE II
COMPARISON OF GESTURE RECOGNITION MODEL PARAMETERS AND RESULTS

Feature	**GestureNet (Fixed Thresh)**	**GestureNet (Adaptive Thresh)**	**GestureNet (Adaptive, ES)**	**MobileNetV3 Small**	**GestureNet (TFLite)**
Framework	TensorFlow	TensorFlow	TensorFlow	TensorFlow	TFLite
Image Filter	Grayscale, fixed threshold	Grayscale, adaptive threshold	Grayscale, adaptive threshold	Grayscale, adaptive threshold	Grayscale, adaptive threshold
Training Params	75 epochs, 1k images/class	75 epochs, 1k images/class	ES stopped at 13-15 epochs, 1k images/class	ES stopped at 13 epochs, 1k images/class	ES criterion
Additional	Data augmentation	Data augmentation	Early Stopping, Data Augmentation, ReduceLROnPlateau	Early Stopping, Data Augmentation, ReduceLROnPlateau	Early Stopping
Key Observation	Poor gesture visibility; hands hard to distinguish	High accuracy but overfitting observed	**High accuracy**, no overfitting, stable validation	Poor learning; Val acc $\approx 10\%$, predictions collapsed	**FP32:** 99% acc, 2.1 MB, 0.70 ms/sample; High precision/recall; Tested INT8 quant.