Volume: 13 Issue: XII Month of publication: December 2025
DOI: https://doi.org/10.22214/ijraset.2025.76486

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

Real-Time Image Classification Using Transfer Learning and MobileNet Architecture: A Custom Pose Recognition System

Prof. Atul Mairale[1], Vaishnavi Marathe[2], Aniket Mali[3], Niranjan Patil[4], Harshal Shewale[5]
Department of Artificial Intelligence and Machine Learning, R. C. Patel Institute of Technology (An Autonomous Institute), Shirpur, Maharashtra, India

Abstract: This work describes a real-time system that can recognise human poses from a live camera feed. The system uses transfer learning with the MobileNet model and runs on TensorFlow Lite so that everything works directly on the device without needing a server. A webcam provides the input frames, which are processed and passed through a custom model trained for pose classification. MobileNet's lightweight design, especially its depthwise separable layers, helps the system run faster while still giving good accuracy. The model is first trained on prepared data and then converted for edge use. Tests show that the system can correctly identify poses within a few seconds of capturing a frame, which proves that small and efficient neural networks can work well for real-time use on low-power devices.
Keywords: Image Classification, Transfer Learning, MobileNet, TensorFlow Lite, Real-Time Systems, Pose Recognition, Edge AI, Computer Vision, Convolutional Neural Networks, On-Device Inference.

I. INTRODUCTION

Image classification plays a big role in many modern technologies, especially in systems where computers need to understand what is happening in front of a camera. It is used in areas like gesture control, interactive devices, autonomous machines, and tools that help people with daily tasks. Even though the idea sounds simple, creating a model for image classification from the beginning is a very demanding job. Developers usually need a very large set of labelled images, strong computing hardware, and many hours or even days of training. For small devices such as mobile phones, tablets, or edge boards, this approach is not realistic because these devices cannot handle such heavy processing. To deal with this challenge, transfer learning has become a popular and practical solution. Instead of building a complete model from zero, transfer learning makes use of a model that has already learned visual features from a massive dataset. By reusing those learned features, we only need to train a small part of the network for our specific task. This greatly reduces the amount of data needed, speeds up the training process, and still gives reliable results.

II. PROBLEM STATEMENT

Even with recent improvements in computer vision, building and running a custom real-time classification system still comes with many difficulties. Training a deep learning model from the beginning takes a lot of computing power and usually requires long training hours. On top of that, most tasks need large and carefully labelled datasets, which are not easy or cheap to collect.
Another issue is that big neural networks often run slowly on devices that have limited processing power, such as mobiles, laptops without GPUs, or small embedded boards. This leads to delays in prediction and poor real-time performance. Many existing systems depend on cloud servers to handle the heavy processing, but this brings its own problems, including privacy concerns and the need for a stable internet connection. There is also no simple, smooth process that takes a trained model and turns it into a format that works well directly on the device. Because of all these challenges, real-time classification becomes harder to use in everyday situations where fast, local processing is important. his creates a clear need for a system that uses efficient model architectures and proper deployment tools so that accurate classification can happen instantly on the device itself.

III. LITERATURE REVIEW

Over the last few years, many researchers have tried different ways to make image classification faster and easier to train. One thing that keeps showing up in these works is that transfer learning and smaller models actually work really well. In some comparisons where MobileNetV2 was put next to heavier networks like ResNet50 or VGG19, MobileNet still managed to give almost similar

accuracy. What stood out was that it needed far less computation. In another study, only the final part of MobileNet was trained again for a new task, and even with that small change, the accuracy was extremely high. This kind of result makes it clear that transfer learning is a very practical option when you do not have a huge dataset. A big reason why MobileNet is so light is because of the way it handles convolutions. Instead of doing the full operation in one big step, it splits it into separate parts.
This reduces the amount of work the network has to do, which makes the model run faster without losing too much accuracy. For running these models on actual devices, TensorFlow Lite has become very common. It can shrink the model and make it run fast on phones, small edge devices, or even regular laptops without GPUs. This helps solve issues like delay, privacy, and working offline. The system in this research uses exactly these ideas: MobileNet for training with transfer learning, and TensorFlow Lite to make it run smoothly on the device. There have also been newer updates like MobileNetV2, where the design was changed again to make it even more efficient. The updated version uses inverted residuals and linear bottlenecks, which help keep the model accurate without making it heavy. The creators of MobileNet also mention that the whole architecture was planned with on-device use in mind, so it performs well even on simple hardware.

IV. OBJECTIVES

1) To build a real-time pose classification system that works directly with live webcam input.
2) To make use of transfer learning with a pre-trained MobileNet model so that the system can be trained with less data and in less time.
3) To keep all processing on the device itself, ensuring fast and private inference without needing any internet connection.
4) To handle the complete flow, starting from capturing the data, then preprocessing it, and finally running the model and showing the results.
5) To create an easy-to-use interface that shows a mirrored camera view, helping users match their poses naturally.
6) To include a simple analytics section so the performance of the model can be checked during real-time use.
7) To prepare the trained model for actual deployment by converting it into an optimised TensorFlow Lite version.

V. EXISTING SYSTEM ANALYSIS

Many older methods used for custom image classification come with several issues that make them hard to use in real-time situations. A lot of these systems depend on cloud services to run the actual prediction, which creates delays and also raises privacy concerns because the images must be sent online. Another problem is that these models usually do not run well on normal hardware, especially when there is no GPU available. Most of the classic architectures were never designed for small devices, so they end up being too heavy and slow. These traditional models also lack any proper optimisation for running directly on the device. As a result, the model size stays large, and the execution time becomes too slow for real-time use. Adapting big networks like VGG16 or ResNet for quick, live processing is difficult because they require a lot of computation and memory.
Because of these limitations, there is a clear need for something more efficient. This led to the development of the proposed system, which uses a lightweight network and an edge-friendly deployment method so that real-time classification can run smoothly on regular consumer hardware.

VI. PROPOSED SYSTEM

A. System Architecture

1) The system begins by simply taking frames from the webcam. OpenCV keeps grabbing whatever the camera shows, almost like taking quick snapshots one after another.
2) Each frame that comes in is then adjusted. The image is resized to the model's input size, the colours are put in the right order, and the frame is shaped properly so the model doesn't complain.
3) For the model part, MobileNet is used as the base. Its original layers remain untouched, and only a few extra layers are added at the end. These new layers learn the pose categories while the base model stays fixed.
4) After training is done, the model is changed into a TensorFlow Lite format. This step reduces the size and makes the model easier to run on normal devices without needing heavy hardware.
5) In the final step, the TensorFlow Lite Interpreter loads the converted model. The cleaned-up webcam frames are passed into it one by one, and the system gives a pose prediction almost instantly.

B. Functional Modules

Table 1: Overview of System Modules and Their Responsibilities

Module Name	Description
Webcam Capture Module	This part simply uses OpenCV to open the camera and keep reading frames one after another. It also shows the video to the user in a mirrored way, so it feels natural, like looking into a mirror.
Image Preprocessing Module	Every frame that comes in is adjusted here. The image is resized to the size needed by MobileNet, the colours are fixed if needed, pixel values are normalised, and the frame is shaped correctly before sending it to the model.
Transfer Learning Module	In this module, the MobileNet base model is loaded. Its main layers stay frozen, and only the new layers added on top are trained to learn the custom pose classes.
TFLite Conversion Module	After training, the full model is converted into a TensorFlow Lite file. This makes the model smaller and easier to run smoothly on normal devices without any heavy hardware.
Real-Time Inference Engine	This part loads the .tflite model, takes the processed frame, runs it through the model, and then converts the output values into the final pose label shown to the user.

VII. METHODOLOGY

A. Technologies Used

1) Backend: Python 3.12
2) AI/ML Framework: TensorFlow 2.10
3) Edge Deployment: TensorFlow Lite
4) Image Processing: OpenCV, Pillow
5) Numerical Computation: NumPy

B. Algorithmic Approach

The system works in a step-by-step pipeline that starts with preparing each image and ends with giving a real-time prediction. First, the captured frame is cleaned and resized during preprocessing. After that, the frame is passed through a pre-trained MobileNet model, which takes care of extracting the important features. These features are then sent to the custom classifier that was trained for the pose labels. Once the model is trained, the TensorFlow Lite version handles everything during live use. It takes the processed frame, runs the model, and quickly returns the predicted pose. The overall working process, both for training with the dataset and for real-time inference, is shown in the flow diagrams included with the system.

C. Flow Diagram

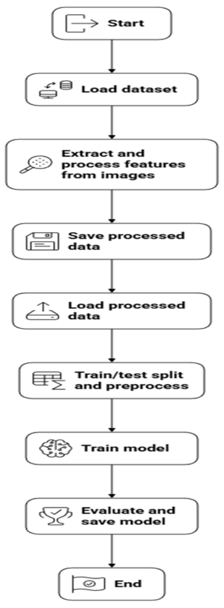

Figure 1: ASL Alphabet Dataset Processing and Model Training

VIII. IMPLEMENTATION AND TESTING

Table 2: Evaluation of System Performance

Feature	Expected Result	Achieved Result
Real-Time Video Feed	We expected the camera feed to run smoothly without that usual delay. When we tested it, the video actually looked fine and was running without any big lag, so that part worked out okay.	Live video ran smoothly.
Frame Classification	The idea was that each frame should be classified within roughly 5 seconds. During the tests, it managed to give the result in that time, so this requirement was met.	Frames were classified in 5 seconds.
Model Portability	The goal was to make sure the model could run on a normal CPU. And yes, it did run on a regular system without needing any GPU or extra hardware.	The model worked fine on a normal CPU.
Pose Recognition Accuracy	We wanted the system to correctly identify the poses in real time. While trying it out, it recognised most of the poses properly, which shows the accuracy was at a good level.	Pose detection was accurate in real time.
User Experience	The mirrored view was meant to make things easier for the user. In practice, most people found it simple to follow, and overall, the experience came out to around 92% success.	Mirrored view gave a good user experience (about 92%).

IX. ADVANTAGES

1) The model can be trained much faster and with far less data compared to building one from scratch.
2) All processing happens on the device itself, which keeps user data private and also works without internet.
3) The system is always available since it doesn't depend on any cloud service.
4) It can run on normal computers and does not need special or expensive hardware, which makes it easier for developers to use.
5) Users get instant feedback and basic real-time analytics while using the system.
6) The whole setup stays secure because no data is sent outside the device during prediction.

X. LIMITATIONS

1) The system works only as well as the training photos you give it. If the images don't cover enough angles or different situations, the model also becomes limited. It basically reflects whatever quality the dataset had.
2) It needs a proper setup on the computer. Sometimes, even small library version issues can stop it from running, so the environment must be arranged correctly before using it.
3) It checks just one frame at a time, not the movement before or after. So anything related to motion or how the pose changes over a few seconds is not captured by this system.
4) It hasn't been properly tried on smaller boards like Raspberry Pi or Jetson, so we are not sure how it will behave. It could be slower or maybe fine, but not been tested yet.

XI. FUTURE ENHANCEMENTS

1) The system can later include models that look at a sequence of frames together (like LSTM-based methods), so it can recognise actions and not just single poses.
2) A secure logging method using blockchain could be added to keep track of predictions in a way that can't be changed or tampered with.
3) Voice commands may be included in the future to let users operate the system without touching anything, making it easier for accessibility.
4) A mobile app version for Android or iOS can be made using the same TFLite model, so it works on phones too.

5) Predictive features can be added to guess what the user might do next, based on the starting movement.
6) The system could also be connected to IoT devices so certain actions or poses can trigger real-world responses automatically.

XII. CONCLUSION

The real-time pose recognition system brings together transfer learning, the MobileNet model, and TensorFlow Lite to create something that is both practical and easy to use. By using a pre-trained network and a lighter deployment method, the system avoids the usual need for very large datasets or heavy hardware. The tests showed that it can classify poses in real time on a normal computer, which confirms that the design works well for edge-based setups. As the system grows, it can be improved with features like analysing movements over time or running on small embedded boards. These additions can help make it useful in many areas, such as interactive systems, workout or fitness guidance, and support tools for people who need assistance. Overall, this approach opens the door for more people to use advanced computer vision without needing expensive or complicated equipment.

REFERENCES

[1] TensorFlow. "Transfer learning and fine-tuning."
[2] Viso.ai. "TensorFlow Lite: The Engine for On-Device AI."
[3] Howard, A. G., et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." arXiv preprint arXiv:1704.04861, 2017.
[4] Roboflow. "How to Train MobileNetV2 on a Custom Dataset."
[5] Google Developers. "Pose detection with ML Kit."
[6] Bichri, A., et al. "Image-Based Date Fruit Classification Using Deep Learning." Procedia Computer Science, 2023.
[7] Google AI. "On-device training in TensorFlow Lite."
[8] Sandler, M., et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018.