Optimizing Deep Vision Models for Eyeglasses Detection in Low-Power Devices

Henrikas GIEDRA, Dalius MATUZEVIČIUS

E-mail: {henrikas.giedra|dalius.matuzevicius}@vilniustech.lt,

Department of Electronic Systems, Vilnius Gediminas Technical University (VILNIUS TECH)

Introduction

The integration of artificial intelligence (AI) into edge computing has enabled resource-limited devices like smartphones, wearables, and IoT systems to perform real-time processing without relying on cloud infrastructure. This advancement is critical for applications such as AI-based virtual eyewear measurement, which depends on accurate eyeglass detection for virtual try-ons and facial recognition. However, deploying deep learning models on low-power devices poses challenges, requiring careful optimization to balance accuracy and efficiency.

Aims and Goals

The study aims to optimize CNNs for eyeglass detection on low-power devices by training various architectures on a prepared dataset. We systematically explored model size and post-training optimizations to assess their impact on accuracy and inference time.

Main objectives:

- Train a variety of convolutional neural network (CNN) architectures utilizing different Keras backbones to identify the most effective configurations.
- Assess the optimized models on low-power platforms to evaluate real-world applicability.
- To reveal the trade-offs between accuracy and inference time across different model complexities..

Methods

For the experiments, the FFHQ (Flickr-Faces-HQ) dataset with 16,000 diverse, high-quality images was used to enhance model generalization for various facial features and eyeglass styles. Bounding boxes were labeled as [xmin, ymin, xmax, ymax]. An example dataset is shown in Figure 1.

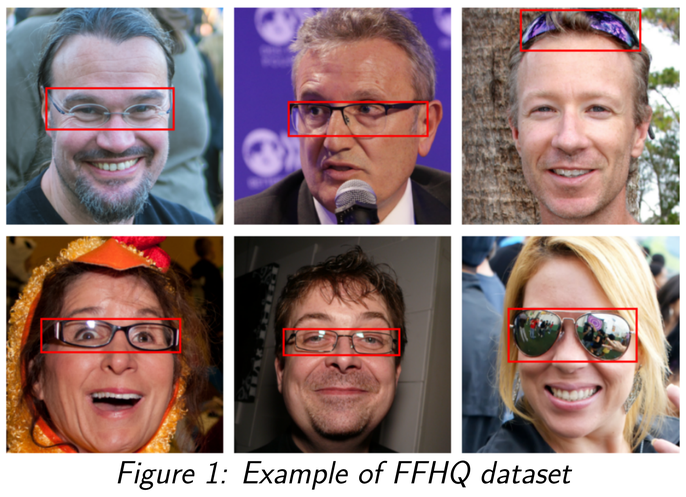

Figure 1: Example of FFHQ dataset

CNN models were trained using Keras-based feature extraction layers, as shown in Figure 2. Training parameters included an input size of 384×384, a batch size of 64, 200 epochs (500 steps per epoch), and a learning rate decaying from 0.002 to 0.0001. The dataset was split into 70% for training and 30% for testing.

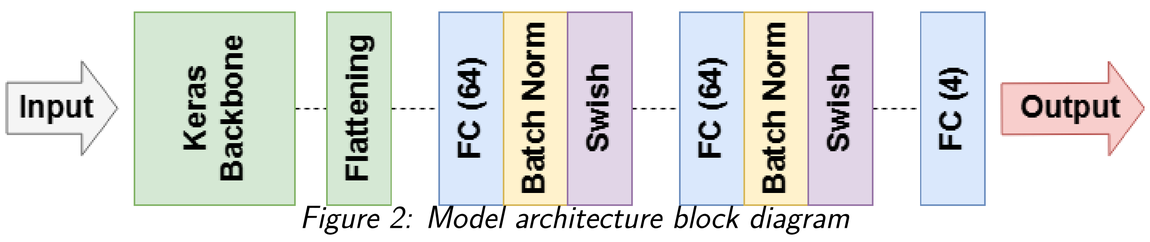

Figure 2: Model architecture block diagram

- **Model Deployment**: Converted Keras models to TFLite for optimization, reduced size and computational demands.
- **Quantization Methods**:
  - **Float16 Quantization**: Convert model weights to 16-bit floats.
  - **Dynamic Range Quantization**: To 8-bit integers.
  - **Full Integer Quantization**: Converted weights and activations

Results

Experiments were performed on two systems:

- **Raspberry Pi 5**: Featuring a quad-core ARM Cortex-A76 CPU.
- **NVIDIA Jetson Orin Nano**: Featuring a quad-core ARM Cortex-A57 MPCore CPU.

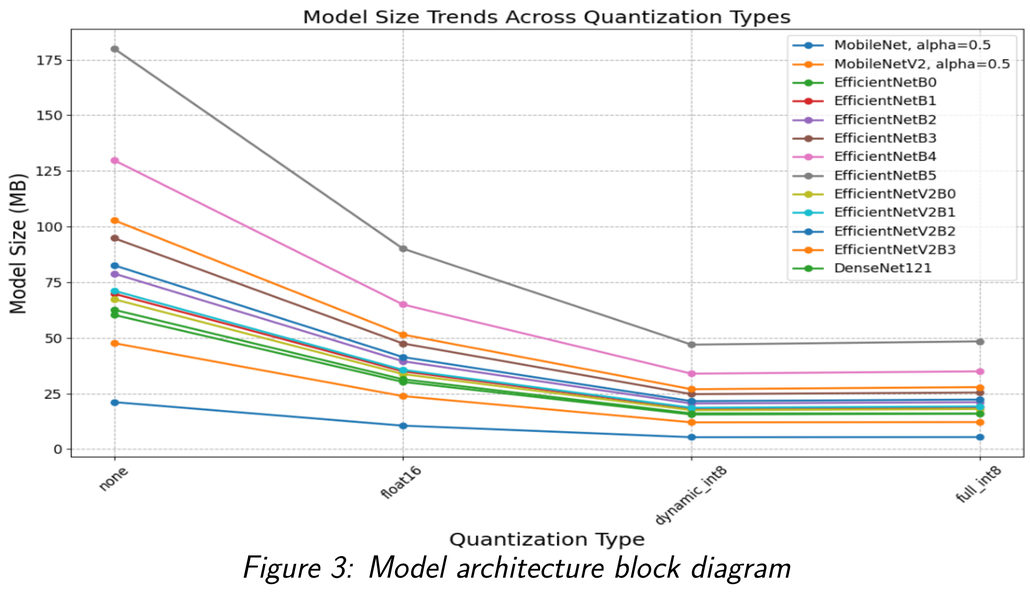

Figure 3: Model architecture block diagram

During the experiment, key metrics were measured, including model size Figure 3, average IoU, and inference time Figure 4.

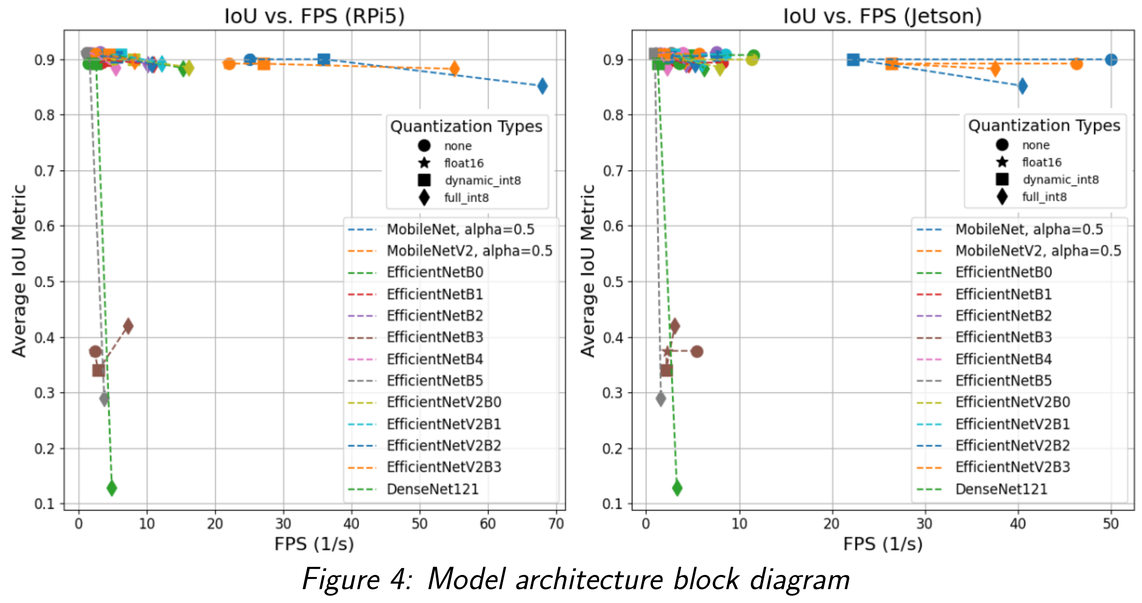

Figure 4: Model architecture block diagram

Conclusions

- **Quantization Techniques:** Full integer optimization reduced model size by up to 75%, enabling deployment on low-power devices.
- **MobileNet Models Excel:** Achieved the best speed-accuracy trade-off with full integer quantization.
- **Edge Computing Implications:** Demonstrated feasibility of deploying high performance models in resource-limited environments for applications such as virtual try-ons and IoT systems.
- **Future Directions:** Explore model pruning and knowledge distillation to enhance efficiency and generalize to other object detection tasks.

VILNIUS TECH
Department of Electronic Systems

This project has received funding from the Research Council of Lithuania (LMTLT), agreement No S-ITP-24-12.