Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 7, No. 7, Juli 2023, hlm. 3342-3347 http://j-ptiik.ub.ac.id

Analisis Performa *TensorFlow Lite* untuk IoT dengan ESP32 DEVKIT-C (Studi Kasus: Pengenalan Gambar Sampah di Sungai)

Mochammad Giri Wiwaha Ngulandoro[1], Sabriansyah Rizqika Akbar[2], Barlian Henryranu Prasetio[3]

Program Studi Teknik Komputer, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]drubber@student.ub.ac.id, [2]sabrian@ub.ac.id, [3]barlian@ub.ac.id

Abstrak

Penggunaan AI dalam IoT sangat bergantung pada keberadaan server untuk menjalankan program AI, seringkali hal ini menimbulkan beberapa masalah terutama latensi, keamanan dan biaya. Untuk mengatasi masalah-masalah tersebut, AI harus dijalankan tanpa menggunakan *Cloud*. Salah satu caranya adalah dengan menjalankan AI pada perangkat yang lebih dekat dengan sumber data (*edge node)*. Teknologi yang memungkinkan hal ini adalah *Tensorflow Lite*. Tujuan dari penelitian ini adalah untuk menganalisis performa dari *Tensorflow Lite Micro* yang dijalankan di ESP32 DevKit C dengan mengujinya menggunakan *input* gambar dengan ukuran yang berbeda-beda dan arsitektur yang berbeda-beda. Temuan-temuan utama pada penelitian ini adalah: (1) Kuantisasi dengan metode *Full integer quantization Integer Only* adalah metode terbaik untuk digunakan dalam implementasi AI di ESP32 DevKit C; (2) total parameter dan ukuran *input* model memiliki pengaruh yang cukup signifikan terhadap performa dan *Deployability* pada ESP32 DevKit-C; (3) Model-model yang diuji menunjukkan *Deployability* yang baik pada ESP32 DevKit-C. Ini menandakan bahwa model AI yang diimplementasikan menggunakan TensorFlow Lite dapat dengan mudah diterapkan dan dijalankan pada perangkat terbatas.

Kata kunci: *mikrokontroler, esp32, tensorflow lite micro, waktu inferensi, akurasi model, deep learning, neural network.*

Abstract

The use of AI in the IoT relies heavily on the presence of servers to run AI programs, which often causes several problems, especially latency, security, and cost. To overcome these issues, AI needs to be run without the use of the Cloud. One way is to run AI on devices that are closer to the data source (edge nodes). The technology that makes this possible is Tensorflow Lite. The purpose of this research is to analyze the performance of Tensorflow Lite Micro running on ESP32 DevKit C by testing it with input images of different sizes and different architectures. Key findings are: (1) the Full integer quantization Integer Only method is the best method to use for AI implementation on ESP32 DevKit-C; (2) the total parameters and input size of the model have a significant influence on the performance and Deployability on ESP32 DevKit-C; (3) the tested models show good Deployability on ESP32 DevKit-C. This indicates that AI models implemented using TensorFlow Lite can be easily deployed and run on limited devices.

Keywords: *microcontroller, esp32, tensorflow lite micro, inference time, model accuracy, deep learning, neural network*

1. PENDAHULUAN

IoT adalah teknologi yang memungkinkan beberapa perangkat elektronik yang terhubung dalam jaringan untuk saling bertukar data, hal ini memungkinkan perangkat yang terhubung untuk dikendalikan dan dipantau dari jarak jauh. Teknologi ini sudah banyak diimplementasikan di berbagai bidang seperti healthcare,transportasi, dan industri (Gokhale, Bhat dan Bhat, 2007). Dalam penggunannya IoT biasanya digunakan untuk mengumpulkan data dari berbagai sensor atau alat lain seperti kamera dan diproses di dalam *Cloud* (Hansen dan Bøgh, 2021). Seiring dengan semakin luas dan

berkembangnya pengunaan IoT, teknologi ini juga sering dikombinasikan penggunaannya dengan teknologi AI (*Artificial Intelligence*).

AI (*Artificial Intelligence*) adalah *software* khusus yang dibuat untuk menjalankan tugas layaknya manusia, seperti pembelajaran, pengenalan pola, dan pemeceahan masalah. Dalam pengaplikasiannya dengan IoT, AI digunakan untuk memproses ataupun menganalisa data yang didapatkan dari sensor atau perangkat lain. Proses ini bisanya dilakukan di dalam fasilitas *Cloud* karena membutuhkan sumber daya yang cukup besar. Hal ini membuat ketergantungan kepada fasilitas *Cloud* terlalu tinggi pada penerapan IoT terutama yang menggunakan AI. Penggunaan IoT dengan *Cloud* seringkali menimbulkan beberapa masalah seperti keterbatasan latensi, keamanan, dan reliability (Abdulkareem dkk., 2021).

Untuk mengatasi masalah-masalah tersebut, AI harus dijalankan tanpa menggunakan *Cloud*. Salah satu caranya adalah dengan menjalankan AI pada perangkat yang lebih dekat dengan sumber data (*edge node*). Teknologi yang memungkinkan hal ini adalah Tensorflow Lite. Tensorflow Lite adalah sebuah *library* yang memudahkan *developer* program untuk membuat dan menjalankan model AI TensorFlow di perangkat mobile, mikrokontroller, dan perangkat IoT lainnya. Berdasarkan penjelasan tersebut peneliti ingin menemukan jumlah *node* optimal pada pemodelan untuk diimplementasikan pada sistem benam ESP32 Devkit-C agar dapat menjadi pertimbangan untuk menggunakan AI di *edge node* dalam IoT tanpa menggunakan *Cloud*.

2. TINJAUAN PUSTAKA

2.1 TensorFlow Lite Micro

TensorFlow Lite for Microcontrollers (TFLM) adalah *software* yang diadaptasi dari TensorFlow Lite yang didesain khusus untuk dapat menjalankan model *machine learning* pada perantkat DSPs, Mikrokontroler, dan perangkat lain dengan memori yang terbatas.

Gambar 1 Diagram Implementasi TFLM

TFLM bekerja dalam empat tahap, pertama TFLM akan membaca *Operator Resolver* yang didefinisikan di dalam aplikasi, *Operator Resolver* mengontrol API TensorFlow yang akan digunakan dalam Mikrokontroler, hal ini akan memperkecil ukuran. Tahap kedua adalah TFLM akan membaca ukuran memori yang telah ditentukan. Area memori yang akan dialokasikan ini biasa disebut "*Arena*" atau "*TensorArenaSize*" dalam implementasinya di aplikasi. Area memori tersebut akan digunakan untuk menampung *input*, *output*, *intermediate array* dan variabel lain yang dibutuhkan *interpreter* TFLM. Tahap ketiga adalah membentuk sebuah TFLM *interpreter* yang akan mengambil model, *Operator Resolver*, dan TensorArenaSize untuk kemudian *interpreter* akan mengalokasikan semua kebutuhan memori pada saat proses inisialisas. Pada tahap keempat aplikasi akan menerima *pointer* dari memori yang merepresentasikan model dari *interpreter*, *pointer* ini yang nantinya akan diisi oleh *input* dari data setelah *input* diisi proses invokasi akan dilakukan dan *interpreter* akan melakukan proses kalkulasi pada model yang nanti akan menghasilkan *output* (David dkk., 2020).

2.2 *Post Training Quantization*

Post-training quantization dalam TensorFlow merupakan teknik yang digunakan untuk mengoptimalkan model *deep learning* setelah proses pelatihan selesai. Tujuannya adalah mengubah model berukuran *floating-point* yang besar menjadi model yang lebih ringkas dan hemat sumber daya dengan mengkonversi parameter *floating-point* menjadi bilangan bulat. Ada beberapa metode *Post Training Quantization* yang disediakan Tensorflow yaitu *Dynamic range quantization*, *Full integer quantization*, dan *Float16 quantization* (TensorFlow, 2023).

Dynamic range quantization merupakan teknik kuantisasi yang mengonversi parameter model menjadi bilangan bulat dengan menggunakan skala dinamis yang beradaptasi

dengan distribusi nilai parameter dalam data latihan. Selama proses ini, TensorFlow akan menganalisis data latihan dan menemukan skala terbaik untuk menggambarkan rentang nilai parameter yang digunakan. Pendekatan ini memungkinkan representasi yang lebih efisien untuk nilai-nilai bobot dan bias dalam model .

Full integer quantization merupakan teknik kuantisasi di mana semua parameter model diubah menjadi bilangan bulat. Dalam proses ini, TensorFlow mengonversi parameter *floating-point* menjadi bilangan bulat dengan mengambil representasi terdekat yang memungkinkan. Nilai yang dihasilkan tensorflow lite dalam konversi ini akan memiliki *range* antara [-127, 128] *signed integer*. *Full integer quantization* ini memiliki dua mode yang berbeda yaitu "*Integer with float fallback (using default float input/output)*" dan "*Integer only*". Mode pertama memungkinkan model untuk dijalankan dengan *input* dan *output* yang masih menggunakan *float-point precision*, namun proses internal di dalam model dilakukan dengan menggunakan *integer precision*. Saat menjalankan model, *input* dan *output* dari lapisan-lapisan terluar akan berupa *float-point*, tetapi ketika data melewati lapisan-lapisan dalam, operasi matematika yang dilakukan menggunakan operasi integer. Sedangkan mode kedua menerapkan konversi ke *full integer precision*, yang berarti seluruh model diubah menjadi menggunakan *integer precision* untuk *input*, *output*, serta proses internal di dalam model. Ini berarti semua tipe data *float-point* diubah menjadi *integer*, termasuk *input* dan *output*, serta parameter dan operasi matematika di dalam model.

Float16 quantization merupakan teknik kuantisasi yang mengonversi parameter model menjadi bilangan *floating-point* 16-bit, yang memiliki presisi lebih rendah dibandingkan dengan *float32 standar* (TensorFlow, 2023).

3. METODOLOGI

3.2. Variabel Uji

Dalam penelitian ini ada tiga variabel yang diuji, yaitu *Deployability*, Waktu Inferensi, dan Akurasi yang didapatkan:

1. *Deployability*

Deployability mengacu pada sejauh mana suatu sistem, aplikasi, atau model dapat diimplementasikan dengan sukses dan berjalan di lingkungan target tanpa masalah. Dalam konteks penelitian ini, "*Deployability*" mencerminkan sebaik apa model-model dari dua mode kuantisasi dan ukuran *input* yang berbeda dapat diimplementasikan pada perangkat terbatas seperti ESP32 DevKit C dengan keterbatasan sumber daya, khususnya kapasitas memori.

2. Waktu Inferensi

Waktu inferensi adalah waktu yang dibutuhkan oleh model untuk melakukan inferensi atau prediksi pada data *input*. Pengujian waktu inferensi sangat penting karena pada perangkat terbatas seperti ESP32 DevKit C, keterbatasan daya komputasi sering menjadi kendala utama. Dengan menguji waktu inferensi dari setiap model yang dihasilkan, kita dapat memahami tingkat efisiensi dan kecepatan masing-masing model, sehingga dapat memilih model yang paling sesuai dengan kebutuhan dan batasan perangkat.

3. Akurasi

Akurasi mengukur sejauh mana model dapat memberikan prediksi yang akurat pada data uji. Dengan menguji akurasi dari keenam model yang berbeda, kita dapat membandingkan performa masing- masing model dan memilih model yang memiliki akurasi yang paling sesuai dengan kebutuhan.

3.2. Metode Penelitian

Penelitian ini mengambil studi kasus pengenalan sampah di sungai untuk tahap pengujian. Penelitian ini memiliki 4 tahapan yaitu Persiapan, Pengujian dan Pengambilan Data, dan Analisis:

1. Persiapan penelitian meliputi pengumpulan dataset, pembuatan model untuk diuji ke dalam ESP32 DevKit-C, dan perancangan purwarupa.
2. Pengujian dilakukan dengan mengamati alat yang sedang berjalan dan digunakan sesuai skenario. Lalu beberapa data diambil dari hasil pengamatan yang dilakukan secara langsung/real time.
3. Pada tahap Analisis dilakukan analisa terhadap data yang diambil untuk mendapatkan hasil penelitian.

Pengumpulan dataset dilakukan dengan cara mengambil gambar dari 5 aliran sungai di sekitar malang dengan menggunakan 2 *device* kamera *handphone* dan ESP32-CAM. Data yang diambil menggunakan *handphone* berupa video 10 detik

setiap sungai lalu akan dipotong setiap 4 detik yang akan diekstrak per *frame* yang akan menghasilkan sekitar 120 gambar setiap video. Sedangkan data yang diambil menggunakan ESP32-CAM adalah foto yang diambil sebanyak 10 foto tiap sungai. Sehingga total data yang digunakan dalam penelitian ini adalah 600 data untuk data dari handphone dan 50 data untuk data dari ESP32- CAM.

Gambar yang diambil dengan *handphone* digunakan untuk proses *training* model sedangkan gambar yang diambil dengan ESP32-CAM digunakan untuk melakukan pengujian pada tahap pengamatan. Gambar tersebut lalu dipisahkan menjadi dua kelas yaitu *dirty* dan *clean*. Kelas *dirty* berarti aliran sungai memiliki sampah di permukannya sedangkan kelas *clean* tidak ada sampah di permukaan airnya.

Kemudian Model untuk pengujian dibuat menggunakan tensorflow di google colab. Model yang dibuat menggunakan 3 macam arsitektur yang berbeda, yaitu MobileNet V1 dengan multiplier 0.25, ShuffleNet dengan multiplier 0.25, dan Xception dengan multiplier 0.25. Setelah melalui proses *training* model-model tersebut lalu diekspor menjadi file TFLite dengan metode kuantisasi *Full integer quantization.* Metode kuantisasi ini memiliki dua mode yaitu *Integer Only* yang berarti semua parameter dalam model termasuk *input* dan outpunya diubah menjadi integer, mode kedua yaitu *Integer with float fallback* yang berarti semua parameter diubah menjadi integer kecuali *input* dan *output*nya.

Selanjutnya untuk pengujian ESP32 DevKIt C dihubungkan ke laptop untuk setelahnya laptop akan mengirimkan file gambar melalui UART ke ESP32. ESP32 lalu melakukan inferensi pada data yang dikirimkan, setelah selesai ESP32 akan mengirimkan prediksi yang dihasilkan ke laptop.

4. ANALISIS DAN PEMBAHASAN

4.1 Hasil Pengujian *Deployability*

Hasil pengujian *Deployability* dari beberapa model dengan ukuran *input* berbeda beda menggunakan metode kuantisasi *Full integer quantization* dalam dua mode "*Integer with float fallback (using default float input/output)*" dan "Integer only" dirangkum pada Tabel 1. dan Tabel 2., dimana jika dapat berjalan dengan baik akan maka akan diberi nilai "*yes*", jika tidak akan diberi nilai "*no*".

Tabel 1. Perbandingan *Deployability* Model 128x128

Arsitektur	Total Parameters	*Deployability*	
		Integer Only	Float *Input/Output*
MobilenetV1	219058	Yes	No
SuffleNet	220562	Yes	No
Xception	22330	Yes	No

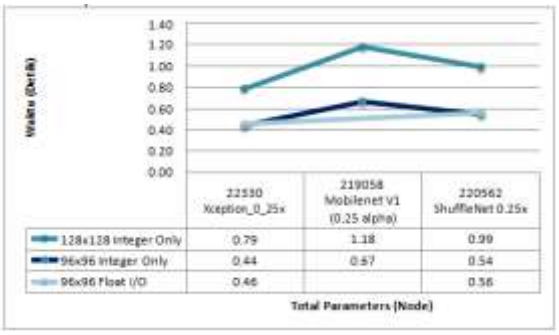
Tabel 2. Perbandingan *Deployability* Model 96x96

Arsitektur	Total Parameters	*Deployability*	
		Integer Only	Float *Input/Output*
MobilenetV1	219058	Yes	No
SuffleNet	220562	Yes	Yes
Xception	22330	Yes	Yes

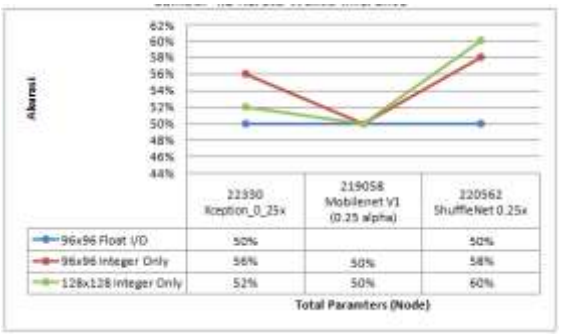
Dari perbandingan pada tabel diatas, ketiga model pada ukuran *input* 128x128 memiliki *Deployability* yang sama, yaitu dapat dijalankan menggunakan kuantisasi *Integer Only*, tetapi tidak mendukung *float input*/*output*. Pada ukuran *input* 96x96, model "MobileNetV1" dan "ShuffleNet" memiliki *Deployability* yang sama dengan ukuran *input* 128x128, yaitu dapat dijalankan menggunakan kuantisasi Integer Only. Namun, model "ShuffleNet" pada ukuran *input* ini mendukung float *input*/*output*. Sedangkan, model "Xception" pada kedua ukuran *input* juga mendukung float *input*/*output*. Berdasarkan hasil tersebut dapat diketahui bahwa kuantisasi dengan mode *Integer Only* adalah mode kuantisasi terbaik karena semua model dengan kuantisasi ini dapat berjalan dengan baik di sistem ESP32 DevKit C.

4.2 Hasil Pengujian Waktu Inference dan Akurasi

Hasil dari pengujian performa menggunakan 50 foto sungai yang diambil menggunakan ESP32-CAM dengan pengaturan default akan dirangkum pada **Gambar 2. dan Gambar 3.**



Gambar 2. Perbandingan Rerata Waktu Inference



Gambar 3. Akurasi model

Berdasarkan hasil pengujian diatas, waktu inferensi dan akurasi model pada ukuran *input* 128x128 dan 96x96, performa TensorFlow Lite bervariasi tergantung pada arsitektur dan ukuran *input* yang digunakan. Model Xception menunjukkan waktu inferensi yang tercepat dari semua, yaitu pada ukuran *input* 96x96 dengan integer only. Model ShuffleNet memiliki waktu inferensi yang lebih lambat pada ukuran *input* 128x128 dengan kuantisasi Integer Only, namun memiliki akurasi lebih tinggi pada ukuran *input* 128x128 dengan kuantisasi Integer Only. Model MobileNetV1 menunjukkan performa yang cukup baik dalam hal waktu inferensi, tetapi memiliki akurasi yang sama pada ukuran *input* 96x96 dan 128x128.

5. KESIMPULAN

Dari penelitian yang dilakukan, dapat disimpulkan bahwa TensorFlow Lite dapat diimplementasikan dengan baik di ESP32 DevKit-C dengan menggunakan kuantisasi Integer Only. Model-model yang diuji memiliki *Deployability* yang baik pada perangkat terbatas ini. Waktu inferensi dan akurasi model bervariasi tergantung pada arsitektur dan ukuran *input* yang digunakan. Selain itu, hasil analisis menunjukkan bahwa total parameter model memiliki pengaruh yang cukup signifikan terhadap performa pada ESP32 DevKit-C. Penelitian ini memberikan informasi penting mengenai kinerja TensorFlow Lite di lingkungan ESP32 DevKit-C, yang dapat menjadi pertimbangan untuk penggunaan kecerdasan buatan di *edge node* dalam lingkungan IoT tanpa ketergantungan pada *Cloud*. Hasil ini berpotensi mendukung perkembangan teknologi AI di lingkungan *edge* computing dan memberikan manfaat dalam implementasi deteksi sampah dan aplikasi lainnya pada perangkat terbatas dengan sumber daya yang terbatas.

6. SARAN

Saran untuk diperhatikan lebih lanjut pada penelitian yang berhubungan selanjutnya yaitu mengembangkan purwarupa agar berjalan secara realtime dan menerapkan Image recognition, mencoba dengan lebih banyaknya jumlah dataset dan keragaman dataset dan juga mencoba arsitektur lain yang dapat berjalan pada ESP32 DevKit C serta mencoba teknik optimasi khusus.

7. DAFTAR PUSTAKA

Abdulkareem, N.M., Zeebaree, S.R.M., Sadeeq, M.A.M., Ahmed, D.M., Sami, A.S. dan Zebari, R.R., 2021. IoT and *Cloud* Computing Issues, Challenges and Opportunities: A Review. Qubahan Academic Journal, [daring] 1(2), hal.1–7. https://doi.org/10.48161/issn.2709-8206.

Canziani, A., Paszke, A. dan Culurciello, E., 2016. An Analysis of Deep Neural Network Models for Practical Applications. [daring] (Nips). Tersedia pada: <http://arxiv.org/abs/1605.07678>.

David, R., Duke, J., Jain, A., Reddi, V.J., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T. dan Warden, P., 2020. TensorFlow Lite Micro: Embedded *Machine learning* on TinyML Systems. [daring] Tersedia pada: <http://arxiv.org/abs/2010.08678>.

Dokic, K., Martinovic, M. dan Mandusic, D., 2020. Inference speed and quantisation of neural networks with TensorFlow Lite for Microcontrollers framework. SEEDA-CECNSM 2020 - 5th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference. https://doi.org/10.1109/SEEDA-CECNSM49515.2020.9221846.

Gokhale, P., Bhat, O. dan Bhat, S., 2007. Introduction to IOT. International Advanced Research Journal in Science, Engineering and Technology ISO, 3297(1). https://doi.org/10.17148/IARJSET.2018.517.

Hansen, E.B. dan Bøgh, S., 2021. *Artificial Intelligence* and internet of things in small and medium-sized enterprises: A survey. Journal of Manufacturing Systems, 58, hal.362–372. https://doi.org/10.1016/J.JMSY.2020.08.009.

Orasan, I.L., Seiculescu, C. dan Caleanu, C.D., 2022. Benchmarking TensorFlow Lite Quantization Algorithms for Deep Neural Networks. SACI 2022 - IEEE 16th International Symposium on Applied Computational Intelligence and Informatics, Proceedings, hal.221–226. https://doi.org/10.1109/SACI55618.2022.9919465.

Tensorflow, 2021. Model optimization | TensorFlow Lite. [daring] Tersedia pada: <https://www.tensorflow.org/lite/performance/model_optimization> [Diakses 10 Juli 2023].

TensorFlow, 2023. Post-training quantization | TensorFlow Lite. [daring] Tersedia pada: <https://www.tensorflow.org/lite/performance/post_training_quantization> [Diakses 7 Juli 2023].