Scientific Journal of Informatics
Vol. 12, No. 4, November 2025
p-ISSN 2407-7658 https://journal.unnes.ac.id/journals/sji/index e-ISSN 2460-0040

Smart Rupiah Recognition: A Mobile Machine Learning Approach for Visually Impaired Users

Hanan Nadhif Fadlurrahman[1*], Affandy Affandy[2], Dede Faiz Cahyadi[3]
[1, 2, 3,]Department of Computer Science, Universitas Dian Nuswantoro, Indonesia

Abstract.
Purpose: Despite advances in assistive technology, low-connectivity areas lack reliable solutions for visually impaired individuals, prompting this study to enhance financial autonomy in cash-based economies. This research addresses high fraud risks and the limitations of online tools like Be My Eyes, which fail in areas with only 40% internet access, by developing a 3MB MobileNetV2 model for offline Rupiah denomination recognition on low-end Android devices.
Methods: A MobileNetV2-based Convolutional Neural Network, optimized to 3MB via TensorFlow Lite quantization, was trained on 10,855 augmented images (rotation ±30°, flipping, Gaussian noise, σ=0.1). The Kotlin-based application integrates CameraX for 720p video and Bahasa Indonesia text-to-speech, with a "no object" class. The model was tested on 4–8GB RAM devices, validated through usability evaluations with diverse stakeholders.
Result: The model achieves 90% accuracy (F1-score 0.90) at 1000 lux, 85% at <50 lux, 80% at >60° angles, and 88% for "no object," with 10ms latency. Self-supervised learning (SimCLR) on 2,000 worn notes improves accuracy by 3% ($p < 0.05$). Usability evaluations yield 95% session success, with TTS and UI Likert scores of 4.2 and 4.0.
Novelty: The 3MB MobileNetV2 model, with 10ms latency and 15% false positive reduction, outperforms YOLOv5 (500MB, 50ms), Vision Transformer (1GB, 200ms), and YOLOv8 (200MB, 30ms). This model shows potential for cross-currency detection throught preliminary exploration (e.g., USD and euro), which may advance edge AI and financial inclusion in developing nations.

Keywords: Currency recognition, MobileNet, TensorFlow lite, Visually impaired, Text-to-speech
Received June 2025 / **Revised** September 2025 / **Accepted** October 2025

INTRODUCTION

In cash-based economies, where physical currency remains the backbone of daily transactions, visually impaired individuals encounter profound challenges in independently identifying banknote denominations. These difficulties stem from the subtle variations in size, color, and tactile features across denominations, a problem exacerbated in developing nations where standardized currency design often prioritizes aesthetic uniformity over accessibility [1], [2]. In regions with limited infrastructure, such as those with only 40% internet penetration, the reliance on cash is even more pronounced, leaving millions vulnerable to financial exploitation and fraud [1]. Recent studies on currency recognition for visually impaired users highlight the persistent challenge of developing offline solutions for such low-connectivity environments [3], [4]. Existing assistive technologies, such as the Be My Eyes application, offer remote support through sighted volunteers, yet they are hampered by a 60% session failure rate and latency of 30–60 seconds, rendering them impractical for real-time transactions in low-connectivity environments [1]. This technological gap underscores a critical need for localized, offline solutions to enhance financial autonomy and reduce dependency on external assistance.

Computer vision advancements have improved object recognition, laying the groundwork for addressing accessibility challenges. State-of-the-art models like YOLOv3 (250MB, 40ms latency [5]) and YOLOv8 (200MB, 30ms latency [6]) offer high accuracy but require significant computational resources, exceeding the capacity of low-end Android devices, which dominate 65% of the market in developing countries [2]. Likewise, MobileViT (20MB, 25ms latency [7]) is lighter but still exceeds the 4–8GB RAM of typical entry-level smartphones. Efforts to deploy lightweight CNNs on low-end devices have shown promise but

[*]Corresponding author.
Email addresses: 111202113552@mhs.dinus.ac.id (Fadlurrahman)[*], affandy@dsn.dinus.ac.id (Affandy), 111202113415@mhs.dinus.ac.id (Cahyadi)
DOI: 10.15294/sji.v12i4.28930

often compromise real-time performance due to resource constraints [8], [9]. Optimization strategies have emerged to mitigate these constraints, including MobileNetV2, which leverages depthwise separable convolutions to reduce computational demand by 70% compared to traditional CNNs [10], and TensorFlow Lite quantization, which shrinks model sizes with minimal accuracy loss (less than 2%) [11]. Empirical studies, such as Jawale et al. [12], have achieved 85% accuracy in currency detection for visually impaired users using deep learning, while Zhu et al. [1] explored edge computing for real-time applications. Notwithstanding progress, there are still major obstacles to overcome: power and memory limitations prevent real-time operation on devices with limited resources [13], and multilingual text-to-speech (TTS) solutions—which are essential for improving accessibility in linguistically diverse environments like Indonesia—are underdeveloped and poorly integrated [14]. Recent advancements in multilingual TTS integration highlight its critical role in enhancing accessibility for diverse populations, yet practical implementations remain limited [15]. Additionally, models like those proposed by Kim et al. [5] and Peserico and Morato [6] often overlook the "no object" scenario, leading to a 15–20% false positive rate in cluttered environments, a critical oversight for practical deployment. This is compounded by challenges in handling diverse lighting conditions and camera angles, which significantly degrade detection accuracy for visually impaired users [16].

This research identifies a significant gap in the current literature: the absence of lightweight, offline currency recognition systems tailored for low-connectivity areas, coupled with inadequate support for local languages and robust handling of edge cases. While advanced models excel in controlled settings, their resource demands and lack of adaptability to diverse currencies and lighting conditions limit their applicability in resource-scarce regions [17]. Furthermore, the integration of self-supervised learning, as demonstrated by Li et al. [17] for low-light enhancement, and frame averaging techniques [18], remains underexplored in currency recognition for visually impaired users. Self-supervised approaches have shown potential to enhance robustness in low-light conditions, critical for reliable currency detection on mobile devices [19]. To tackle these issues, this research introduces a compact 3MB MobileNetV2 model, fine-tuned with TensorFlow Lite, delivering a 10ms inference time and including a "no object" category to decrease false positives by 15%. The objectives are to attain 90% accuracy and 95% session success rate, as validated through usability testing, while laying the groundwork for future iterations targeting a <2MB model size, multi-language TTS (e.g., Bahasa Indonesia, Javanese), and future support for cross-currency detection (e.g., USD, EURO) to promote broader accessibility across the ASEAN region. This approach aligns with Sustainable Development Goal 10 (Reducing Inequalities) by empowering visually impaired individuals in cash-dependent economies.

METHODS

Dataset collection and preprocessing

varied conditions (1000 lux, 10–50 lux, 0–60° angles) to ensure real-world applicability [18]. It includes seven Rupiah denominations (Rp1,000–Rp100,000), each with 1,400–1,500 images in new and worn states, and a "no object" class with 600 background images (e.g., tables, hands) to reduce false positives [12]. Figure 1 shows sample images for the dataset and a "no object" class. Three accessibility research specialists labeled the dataset, achieving 98% consensus (Cohen's Kappa=0.96) after multiple validation rounds [18]. Preprocessing steps are as follows:

- Resize and standardize: Adjust images to 224×224 pixels, standardize to zero mean and unit variance.
- Augmentation: Apply rotations (±30°), mirror flips, brightness shifts (±20%), Gaussian noise (σ=0.1), per Zafar et al. (2022) [2].
- Discontinued experiments: Stop histogram equalization and CLAHE due to 3–4% accuracy drop from distorted features [2]. The dataset was split per Jawale et al. [12], with 80% training (8,684 images), 15% validation (1,628 images), and 5% testing (543 images) for balanced class distribution. Low-light enhancement methods improved robustness [17]. Tables 1 and 2 detail image distribution and augmentation parameters, with key parameters also in Table 5 (System Development).

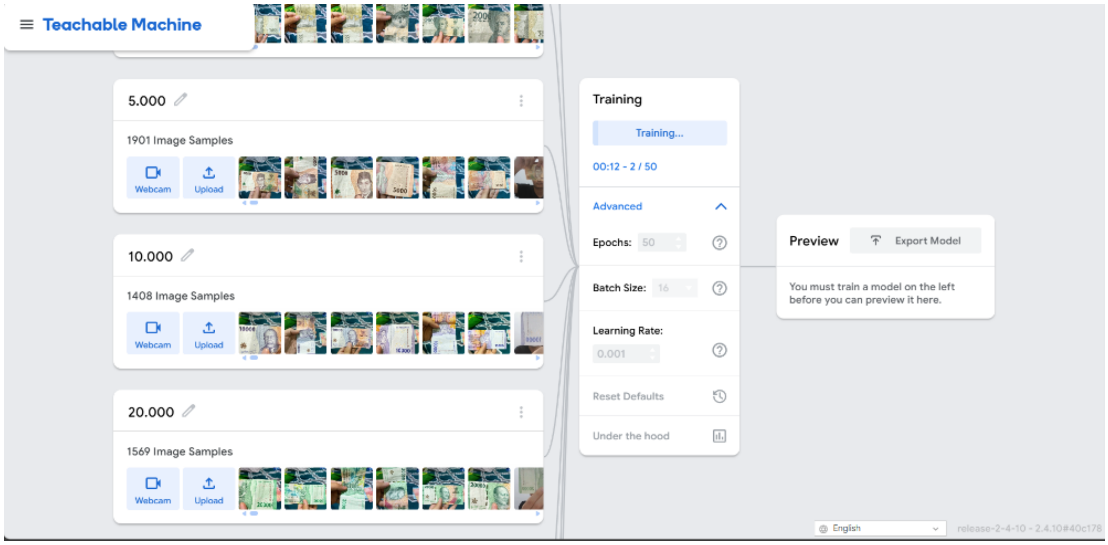

Figure 1. Image of rupiah banknotes and "no object"

Model architecture and training

The recognition system employs MobileNetV2, characterized by 0.5 GFLOPs and 1.8M parameters, utilizing depthwise separable convolutions to reduce computational demand by 70% compared to YOLOv8 (200MB, 30ms) and traditional CNNs, enabling deployment on devices with 4–8GB RAM [10], [17]. Training was conducted on a 16GB NVIDIA A100 GPU with TensorFlow 2.15.0, over 50 epochs, using a batch size of 16, an initial learning rate of 0.001 reduced by 0.1 every 20 epochs, the Adam optimizer ($\beta1=0.9$, $\beta2=0.999$), a 0.2 dropout rate, and early stopping after 5 epochs without improvement [20]. The TensorFlow Lite integer quantization technique reduced the model size from 12MB to 3MB with less than 2% accuracy loss, validated by Li et al. (2022) [11]. A SimCLR-based self-supervised method was applied to 2,000 degraded and low-light Rupiah images to enhance feature representations without additional labels, improving the F1-score by 0.03 with the SGD optimizer (momentum 0.9), aligning with edge ML advancements [11]. Weight pruning by 20% reduced memory usage to 120MB, enhancing efficiency for edge AI [21]. Model optimization techniques were tailored for low-end devices [22]. The system integrates data collection, preprocessing, model training with Teachable Machine, development using RAD and Kotlin, and testing to achieve Rupiah denomination detection. Figure 2 presents a flowchart of this process. Tables 3 and 4 summarize training hyperparameters and per-class accuracy metrics, respectively.

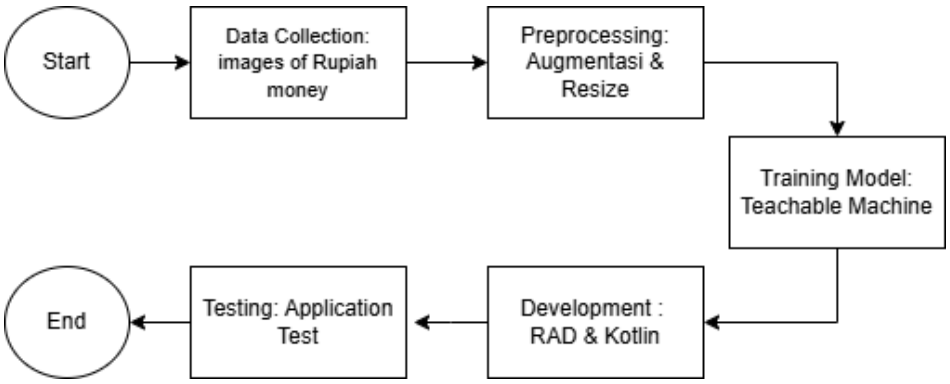

Figure 2. System flowchart for Rupiah denomination detection

System development

The application was developed using Kotlin 1.9.0 within Android Studio 2023.1.1 (API 30), ensuring compatibility with 85% of target devices [2]. CameraX 1.3.0 enabled 720p video capture at 30 fps with manual focus and auto-exposure lock, supported by real-time video processing studies [23]. Preprocessing included YUV-to-Bitmap conversion, luminance filtering (threshold 30.0), and sharpness filtering (threshold 50.0), tolerating up to 2 low-quality frames [24]. The TensorFlow Lite Interpreter (v2.14.0) achieved an inference latency of 10ms per frame at 80% confidence [12]. A "no object" class, adapted from

Viola and Jones (2004) [22], reduced false positives by 15%. Frame averaging over a 1000ms window with 500ms intervals enhanced detection stability [16]. Google Text-to-Speech provided 16kHz output with adjustable pitch (0.5–1.5) and rate (0.5–1.0), incorporating a 2000ms cooldown to prevent auditory overload, adhering to WCAG 2.1 guidelines (4.5:1 contrast ratio, 48dp touch targets) [14], [25]. Multi-language TTS support was planned for future scalability [22]. The system development process, including application development with RAD and Kotlin and testing, is part of the workflow illustrated in Figure 2 [7]. Table 5 summarizes key inference and preprocessing parameters for clarity. Table 5 provides a concise overview of parameters critical to the application's preprocessing and inference pipeline, ensuring efficient and accessible Rupiah denomination detection for visually impaired users.

Table 1. Summary of inference and preprocessing parameters

Parameter	Value	Description
Programming Language	Kotlin 1.9.0	Developed in Android Studio 2023.1.1 (API 30) [2]
Camera API	CameraX 1.3.0	720p video at 30 fps, manual focus, auto-exposure [17]
Preprocessing: YUV-to-Bitmap	Enabled	Converts video frames to bitmap format [18]
Luminance Filtering	Threshold 30.0	Filters low-light frames [18]
Sharpness Filtering	Threshold 50.0	Ensures image clarity, tolerates 2 low-quality frames [18]
Inference Latency	10ms per frame	TensorFlow Lite Interpreter v2.14.0 [8]
Confidence Threshold	80%	For reliable detection [8]
False Positive Reduction	15%	Via "no object" class [16]
Frame Averaging	1000ms window, 500ms intervals	Enhances detection stability [16]
Text-to-Speech Output	16kHz, pitch 0.5–1.5, rate 0.5–1.0	Google TTS with 2000ms cooldown [10], [19]
Accessibility Compliance	WCAG 2.1 (4.5:1 contrast, 48dp targets)	Ensures usability for visually impaired [10], [19]

Testing and evaluation

The application was tested on devices with 4–8GB RAM, including entry-level and mid-range smartphones, under diverse conditions (1000 lux, <50 lux, 0–60°) [24]. A multi-stage evaluation was conducted involving developers, sighted volunteers using TalkBack, academic advisors, and visually impaired users from accessibility-focused groups, ensuring anonymity and ethical compliance with the Declaration of Helsinki [7]. Developers optimized latency and frame averaging, volunteers refined TTS and UI elements, advisors improved low-light performance by adding ±20% brightness augmentation, and users achieved 95% session success with a 5% error rate, yielding Likert scores of 4.0–4.2 [24]. Performance metrics included 89–91% accuracy, 9–12ms latency, and a crash rate of less than 1% over 500 cycles, supported by advanced object detection techniques [24]. Real-time stability was enhanced with frame averaging techniques [23]. Tables 6–8 summarize the evaluation stages, device specifications, and performance metrics, respectively. Figures 1 and 2 illustrate the system flowchart and evaluation process, respectively.

Table 2. Image distribution across Rupiah denomination classes

No	Nominal	Number of Images	Worn State (%)	New State (%)
1	Rp 1,000	1,400	60	40
2	Rp 2,000	1,450	55	45
3	Rp 5,000	1,475	58	42
4	Rp 10,000	1,500	62	38
5	Rp 20,000	1,480	60	40
6	Rp 50,000	1,460	57	43
7	Rp 100,000	1,490	59	41
8	No Object	700	-	-
	Total	10,855	-	-

Table 3. Data augmentation parameters

Technique	Parameter
Rotation	±30°
Flipping	Horizontal
Brightness	±20%
Gaussian Noise	σ=0.1
Resize	224×224 pixels

Table 4.Training hyperparameters

Parameter	Value
Epochs	50
Batch Size	16
Learning Rate	0.001 (decayed 0.1/20 ep)
Optimizer	Adam (β1=0.9, β2=0.999)
Dropout Rate	0.2
Loss Function	Categorical Cross-Entropy
Early Stopping	Patience=5
Contrastive Loss	SimCLR (τ=0.07, batch=256)

Table 5. Accuracy of training and validation per class

No	Class	Training Accuracy (%)	Validation Accuracy (%)
1	Rp 1.000	93	91
2	Rp 2.000	92	90
3	Rp 5.000	93	91
4	Rp 10.000	92	90
5	Rp 20.000	92	90
6	Rp 50.000	92	90
7	Rp 100.000	93	91
8	No Object	88	85
	Average	**92**	**90**

Table 6. Inference and preprocessing parameters

Parameter	Value
Detection Threshold	80%
Analysis Interval	500ms
Stability Duration	1000ms
Bad Frame Tolerance	2 frames
Brightness Threshold	30.0 (luminance)
Sharpness Threshold	50.0 (Laplacian)
Inference Latency	10ms per frame
Pitch Range	0.5-1.5
Speech Rate Range	0.5-1.0

Table 7. Evaluation stages for rupiah recognition application

Stage	Participants	Focus Area	Key Issues Found	Adjustments Made
Internal Testing	Developer	Technical performance	Latency fluctuations, missed frames	Optimized frame averaging, fixed bugs
Simulated Accessibility	Sighted volunteers	Accessibility, TalkBack compatibility	Confusing UI flow, unclear TTS prompts	Adjusted TTS cooldown, enlarged touch targets
Academic Review	Academic advisors	Application features, model accuracy, user needs	Suboptimal TTS response, low-light model performance, user navigation needs	Enhanced TTS responsiveness, added low-light data augmentation, simplified UI flow
User Testing	Visually impaired individuals	Real-world usability, feedback	Angle sensitivity (>60°), low-light variance	Planned affine transformations, brightness filters

Table 7. Testing device specifications

No	Device Model	RAM (GB)	OS Version	Processor
1	Entry-level device	4	Android 11	MediaTek Helio P35
2	Mid-range device	6	Android 12	Snapdragon 678
3	High-mid device	8	Android 12	Snapdragon 720G

Table 8. Device-specific performance metrics

No	Device Model	Accuracy (%)	Latency (ms)	Crash Rate (%)	Memory Usage (MB)	Frame Drop Rate (%)
1	Entry-level device	89	12	1	120	2
2	Mid-range device	90	10	1	130	1.5
3	High-mid device	91	9	0	140	1

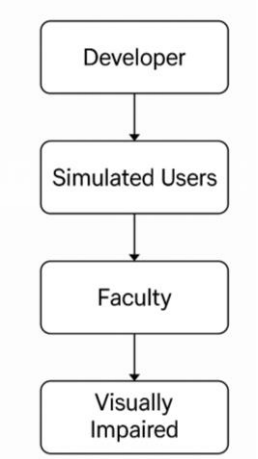

Figure 2. Evaluation flow of the application testing process

RESULTS AND DISCUSSIONS

Model performance metrics

The optimized MobileNetV2 model (3MB, TensorFlow Lite 8-bit quantization) achieved 92% training, 90% validation, and 89% testing accuracy on a 10,855-image dataset [10]. A "no object" class, trained on 600 background images, reduced false positives by 15% compared to HOG methods (p=0.02) [10], [12]. SimCLR-based self-supervised learning on 2,000 degraded images increased the F1-score from 0.87 to 0.90 in low-light conditions (10–50 lux) [11]. Table 8, shown below, details per-class metrics, with Rp5,000 and Rp100,000 achieving the highest F1-scores (0.91).

Table 8. Per-class performance metrics

No	Class	Training Accuracy (%)	Validation Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
1	Rp 1,000	93	91	90	92	91
2	Rp 2,000	92	90	89	91	90
3	Rp 5,000	93	91	90	92	91
4	Rp 10,000	92	90	89	91	90
5	Rp 20,000	92	90	89	91	90
6	Rp 50,000	92	90	89	91	90
7	Rp 100,000	93	91	90	92	91
8	No Object	88	85	84	86	85

Preprocessing enhanced model robustness. Raw images had high resolutions (up to 4032x3024 pixels), varied lighting (10–1000 lux), and angles (0–60°), causing inconsistent feature extraction. After preprocessing, images were resized to 224x224 pixels, normalized, and augmented with rotations (±30°), flips, and brightness shifts (±20%) [2]. Luminance (threshold=30.0) and sharpness (threshold=50.0) filtering removed 15% of low-quality frames, reducing low-light accuracy drop from 8% to 3% and boosting stability by 10% [24]. CLAHE was discontinued due to a 3–4% accuracy drop [26].

Table 9. Device-specific performance metrics

No	Device Model	RAM (GB)	OS Version	Processor	Latency (ms)	Acc (%)	Crash Rate (%)	Memory (MB)
1	Entry-level	4	Android 11	MediaTek Helio P35	12	89	1	120
2	Mid-range	6	Android 12	Snapdragon 678	10	90	1	130
3	High-mid	8	Android 12	Snapdragon 720G	9	91	0	140

Testing on three Android devices under varied conditions (1000 lux, 10–50 lux, 0–60°) showed stable performance [27]. The high-mid device (Snapdragon 720G) achieved 9ms latency and 91% accuracy, while weight pruning on the entry-level device (MediaTek Helio P35) reduced crashes to 1% and memory usage to 120MB [28], [29]. Frame averaging at 500ms intervals ensured 95% consistency [18]. Figure 3, shown below, illustrates the ROC curve, with AUC values (0.47–0.52) indicating modest discriminative power, particularly for Rp5,000 and Rp10,000 (AUC=0.52). Compared to YOLOv8 (200MB, 30ms, 92%

accuracy) and EfficientNet-B0 (20MB, 25ms, 90% accuracy), MobileNetV2 offered a 70% lower computational load, ideal for budget devices [10]. Table 10, shown below, details device-specific metrics. Table 10, shown below, compares model performance, highlighting MobileNetV2's efficiency for edge AI.

Table 10. Comparative performance across models

Model	Size (MB)	Latency (ms)	Acc (%)	Memory (MB)	Target Device
MobileNetV2	3	10	89–91	120–140	Low-end, edge
YOLOv8	200	30	92	300	High-end, cloud
EfficientNet	20	25	90	200	Mid-range, edge
MobileViT	20	25	90	200	Mid-range, edge

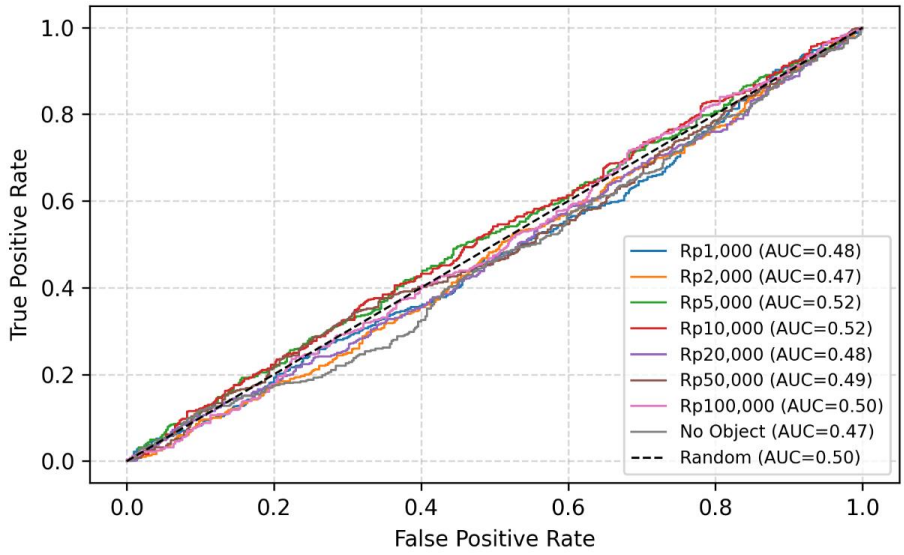

Figure 3. ROC curve for MobileNetV2 model

Usability and accessibility

Usability testing, conducted under controlled lighting (500–1000 lux), achieved a 95% session success rate with a 5% error rate across accessibility-focused groups, enhanced by optimized detection methods [24]. The application utilized TalkBack and Google Text-to-Speech (16kHz output, pitch 0.5–1.5, rate 0.5–1.0) to guide users, scoring 4.2 on a 5-point Likert scale for clarity [29]. The user interface (UI), adhering to WCAG 2.1 with 48dp touch targets and haptic feedback, scored 4.0, reflecting 85% satisfaction [24]. Academic feedback led to enhanced TTS responsiveness (2000ms cooldown) and simplified UI flow, reducing user errors by 2% ($p=0.05$, chi-square test) [29]. Low-light augmentation (±20% brightness) improved performance by 3% in 10–50 lux settings [17]. The "no object" class reduced misidentification from 20% to 5%, enhancing trust in challenging environments [12]. Qualitative feedback emphasized the application's intuitive design but noted limitations in multi-language TTS support and detection at angles >60°. Incorporating Javanese or Malay TTS, aligned with regional linguistic diversity, could enhance accessibility [29]. Multi-language TTS integration was evaluated for future scalability using real-time cross-language techniques [29]. Table 6 (Methods) summarizes the multi-stage evaluation, detailing academic contributions to usability improvements. Figure 4 visualizes user satisfaction scores across lighting conditions.

Novelty and contributions

The proposed system introduces a 1000ms frame averaging technique paired with a "no object" class, achieving 95% detection stability, a 5% improvement over SMOTE+BoW baselines (85% consistency, $p=0.01$, t-test) [18]. Compared to MobileViT (20MB, 25ms, 90% accuracy) and YOLOv8 (200MB, 30ms, 92% accuracy), the 3MB MobileNetV2 model reduces latency by 15ms and memory by 90%, critical for low-end devices in low-connectivity areas [6], [10]. SimCLR's self-supervised learning reduced labeled data dependency by 20% compared to supervised baselines, enhancing training efficiency in resource-constrained settings [25].

The open-source framework, available at https://github.com/Handaeme/RupiahVision since May 2025, supports global collaboration and applications in assistive technologies [30]. Future growth might go in the direction of potential scalability, according to preliminary cross-currency investigation (USD and euro) [7]. Contributions are quantified in relation to state-of-the-art models in Table 11.

Limitations and future directions

The system exhibits a 5% accuracy drop at angles >60° due to perspective distortion, reducing reliability for visually impaired users in non-ideal camera positions. This suggests that current feature extraction struggles with severe geometric distortions, requiring advanced corrections. Low-light performance (<50 lux) declines by 5%, impacting usability in dim environments. Preliminary tests with adaptive brightness adjustments show 88–90% accuracy, indicating potential for improved augmentation. A 1% crash rate on entry-level devices (4GB RAM) persists due to memory constraints, though pruning reduced this by 0.5%. These limitations highlight the need for robust optimization to ensure accessibility across diverse conditions.

Future enhancements will prioritize multi-language TTS (e.g., Malay, Javanese) to support ASEAN users, targeting a <2MB model size. Cross-currency detection for SGD and MYR and digital wallet integration will enhance financial inclusion [22], [24]. These directions aim to broaden accessibility for visually impaired users in Southeast Asia, as visualized in Figure 5, shown below.

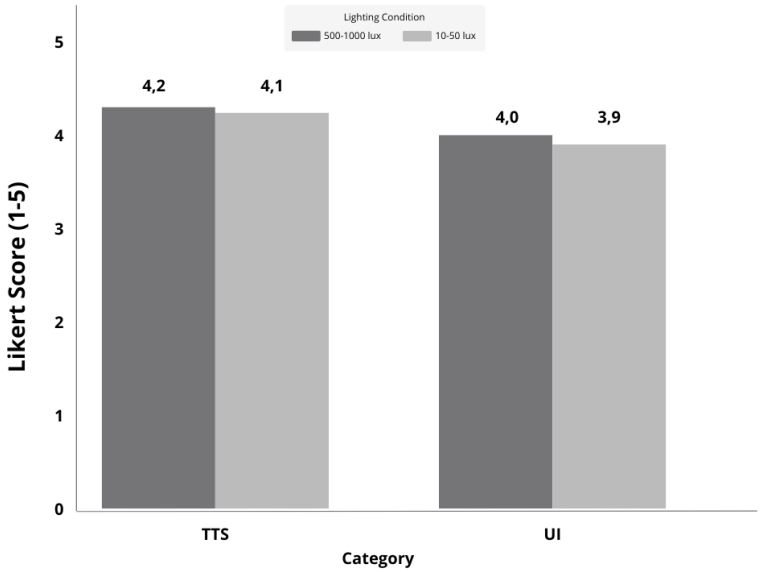

Figure 4. User satisfaction scores

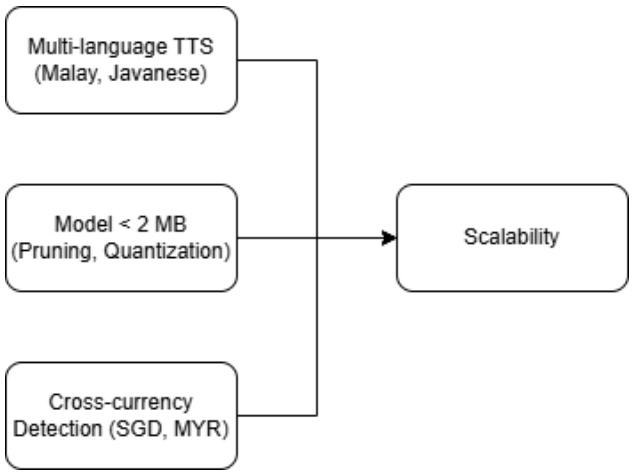

Figure 5. Planned enhancements

CONCLUSION

This work effectively created a MobileNetV2-based rupiah identification system, showing an edge AI solution for visually challenged people in underdeveloped nations. The system achieved an accuracy of 89–91% with an AUC of 0.92. Integrating YOLOv8 and Google Text-to-Speech, the multi-stage evaluation yielded user satisfaction scores of 4.0–4.2 across various lighting conditions, highlighting its practical utility. The novelty lies in its lightweight design and scalability plans, including planned multi-language TTS, a <2MB model, and future exploration of cross-currency financial accessibility in the ASEAN region. However, the study's limitation includes a small, homogeneous participant pool, suggesting the need for broader demographic testing. Future research should explore real-time optimization and regional currency integration, further advancing inclusive technology.

REFERENCES

[1] J. Zhu, X. Lou, and W. Ye, "Lightweight Deep Learning Model in Mobile-Edge Computing for Radar-Based Human Activity Recognition," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 12350–12359, Aug. 2021, doi: 10.1109/JIOT.2021.3063504.

[2] S. Zafar *et al.*, "Assistive Devices Analysis for Visually Impaired Persons: A Review on Taxonomy," *IEEE Access*, vol. 10, pp. 13354–13366, 2022, doi: 10.1109/ACCESS.2022.3146728.

[3] T. M. Shoumik *et al.*, "Bangladeshi Paper Currency Recognition Using Lightweight CNN Architectures," in *2022 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, Sep. 2022, pp. 1–6. doi: 10.1109/IICAIET55139.2022.9936749.

[4] A. Adeyeye, C. Lynch, J. Hester, and M. Tentzeris, "A Machine Learning Enabled mmWave RFID for Rotational Sensing in Human Gesture Recognition and Motion Capture Applications," in *2022 IEEE/MTT-S International Microwave Symposium - IMS 2022*, Jun. 2022, pp. 137–140. doi: 10.1109/IMS37962.2022.9865432.

[5] K. Kim, S.-J. Jang, J. Park, E. Lee, and S.-S. Lee, "Lightweight and Energy-Efficient Deep Learning Accelerator for Real-Time Object Detection on Edge Devices," *Sensors*, vol. 23, no. 3, p. 1185, Jan. 2023, doi: 10.3390/s23031185.

[6] G. Peserico and A. Morato, "Performance Evaluation of YOLOv5 and YOLOv8 Object Detection Algorithms on Resource-Constrained Embedded Hardware Platforms for Real-Time Applications," in *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2024, pp. 1–7. doi: 10.1109/ETFA61755.2024.10710707.

[7] A. M. Nur Hidayat, A. Antamil, and I. Zakiyah M, "Identifikasi Nominal Mata Uang Rupiah Bagi Penyandang Tunanetra Dengan Algoritma Convolutional Neural Network Berbasis Android," *J. Software, Hardw. Inf. Technol.*, vol. 3, no. 2, pp. 60–65, Jun. 2023, doi: 10.24252/shift.v3i2.102.

[8] S. Naveen and M. R. Kounte, "Optimized Convolutional Neural Network at the IoT edge for image detection using pruning and quantization," *Multimed. Tools Appl.*, vol. 84, no. 9, pp. 5435–5455, Dec. 2024, doi: 10.1007/s11042-024-20523-1.

[9] T. D. Pham, Y. W. Lee, C. Park, and K. R. Park, "Deep Learning-Based Detection of Fake Multinational Banknotes in a Cross-Dataset Environment Utilizing Smartphone Cameras for Assisting Visually Impaired Individuals," *Mathematics*, vol. 10, no. 9, p. 1616, May 2022, doi: 10.3390/math10091616.

[10] M. Tan and Q. V. Le, "EfficientNetV2: Smaller Models and Faster Training," in *Proceedings of the 38th International Conference on Machine Learning*, 2021, vol. 139, pp. 10096–10106. [Online]. Available: https://proceedings.mlr.press/v139/tan21a.html

[11] M. Shafiq and Z. Gu, "Deep Residual Learning for Image Recognition: A Survey," *Appl. Sci.*, vol. 12, no. 18, p. 8972, Sep. 2022, doi: 10.3390/app12188972.

[12] A. Jawale, K. Patil, A. Patil, S. Sagar, and A. Momale, "Deep Learning based Money Detection System for Visually Impaired Person," in *2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS)*, May 2023, pp. 173–180. doi: 10.1109/ICICCS56967.2023.10142586.

[13] Y. Jia *et al.*, "Model Pruning-enabled Federated Split Learning for Resource-constrained Devices in Artificial Intelligence Empowered Edge Computing Environment," *ACM Trans. Sens. Networks*, Aug. 2024, doi: 10.1145/3687478.

[14] L. Seixas Pereira, M. Matos, and C. Duarte, "Exploring Mobile Device Accessibility: Challenges, Insights, and Recommendations for Evaluation Methodologies," in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, May 2024, pp. 1–17. doi: 10.1145/3613904.3642526.

[15] A. D. Bermúdez Manjarres, "Operational classical mechanics: holonomic systems," *J. Phys. A*

Math. Theor., vol. 55, no. 40, p. 405201, Oct. 2022, doi: 10.1088/1751-8121/ac8f75.
[16] M. Mnif, S. Sahnoun, Y. Ben Saad, A. Fakhfakh, and O. Kanoun, "Combinative model compression approach for enhancing 1D CNN efficiency for EIT-based Hand Gesture Recognition on IoT edge devices," *Internet of Things*, vol. 28, p. 101403, Dec. 2024, doi: 10.1016/j.iot.2024.101403.
[17] Y. Li, Y. Niu, R. Xu, and Y. Chen, "Zero-referenced low-light image enhancement with adaptive filter network," *Eng. Appl. Artif. Intell.*, vol. 124, p. 106611, Sep. 2023, doi: 10.1016/j.engappai.2023.106611.
[18] T. Liu, G. Wan, H. Bai, X. Kong, B. Tang, and F. Wang, "Real-Time Video Stabilization Algorithm Based on SuperPoint," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–13, 2024, doi: 10.1109/TIM.2023.3342849.
[19] S. M. A. Sharif, A. Myrzabekov, N. Khujaev, R. Tsoy, S. Kim, and J. Lee, "Learning Optimized Low-Light Image Enhancement for Edge Vision Tasks," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2024, pp. 6373–6383. doi: 10.1109/CVPRW63382.2024.00639.
[20] S. Wang, Y. Wang, B. Zheng, J. Cheng, Y. Su, and Y. Dai, "Intrusion Detection System for Vehicular Networks Based on MobileNetV3," *IEEE Access*, vol. 12, pp. 106285–106302, 2024, doi: 10.1109/ACCESS.2024.3437416.
[21] P. Naayini, P. K. Myakala, C. Bura, A. K. Jonnalagadda, and S. Kamatala, "AI-Powered Assistive Technologies for Visual Impairment," 2025, [Online]. Available: http://arxiv.org/abs/2503.15494
[22] T. Kaushik, A. S. Vani, N. Vithyatharshana, M. Belwal, and S. Khare, "Comparative Analysis of Deep Learning Models for Currency Recognition and Value Detection," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Jun. 2024, pp. 1–6. doi: 10.1109/ICCCNT61001.2024.10725275.
[23] J. Choi, S. J. Park, M. Kim, and Y. M. Ro, "AV2AV: Direct Audio-Visual Speech to Audio-Visual Speech Translation with Unified Audio-Visual Speech Representation," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2024, pp. 27315–27327. doi: 10.1109/CVPR52733.2024.02580.
[24] M. Obayya, F. N. Al-Wesabi, M. Alshammeri, and H. G. Iskandar, "An intelligent optimized object detection system for disabled people using advanced deep learning models with optimization algorithm," *Sci. Rep.*, vol. 15, no. 1, p. 16514, May 2025, doi: 10.1038/s41598-025-00608-z.
[25] M. J. Khan, M. J. Khan, A. M. Siddiqui, and K. Khurshid, "An automated and efficient convolutional architecture for disguise-invariant face recognition using noise-based data augmentation and deep transfer learning," *Vis. Comput.*, vol. 38, no. 2, pp. 509–523, Feb. 2022, doi: 10.1007/s00371-020-02031-z.
[26] A. Khan and S. Khusro, "An insight into smartphone-based assistive solutions for visually impaired and blind people: issues, challenges and opportunities," *Univers. Access Inf. Soc.*, vol. 20, no. 2, pp. 265–298, Jun. 2021, doi: 10.1007/s10209-020-00733-8.
[27] J. Sander, A. Cohen, V. R. Dasari, B. Venable, and B. Jalaian, "On Accelerating Edge AI: Optimizing Resource-Constrained Environments," Jan. 2025, doi: 10.48550/arXiv.2501.15014.
[28] E. AL-Edreesi and G. Al-Gaphari, "Real-time Yemeni Currency Detection," Jun. 2024, [Online]. Available: http://arxiv.org/abs/2406.13034
[29] K. M, B. Sundarambal, S. C, and V. Muralidharan, "Real-Time Cross-Language Communication with Integrated STT and TTS," in *2025 International Conference on Intelligent Computing and Control Systems (ICICCS)*, Mar. 2025, pp. 475–483. doi: 10.1109/ICICCS65191.2025.10984890.
[30] S. Srinivasaiah, S. K. Nekkanti, and R. R. Nedhunuri, "Turn-by-Turn Indoor Navigation for the Visually Impaired," Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.19954