2D Image to 3D Model Converter Bot Using AI Depth Estimation and Flask

1. Problem Statement

Manual 3D modeling from 2D images is slow, requires skill, and is not scalable. Students and creators often need 3D models from 2D images for animation, games, and research, but existing workflows demand complex software or manual effort. An automated web system offers a rapid, practical solution, bringing AI-powered 2D-to-3D generation to everyone.

2. Motivation

1. Make 3D asset generation accessible to non-experts.
2. Enable rapid prototyping and educational exploration in graphics, machine learning, and design.
3. Leverage modern deep models to automate a tedious process and support creativity in the digital era.

3. Objectives

1. Apply deep learning-based monocular depth estimation to infer 3D structure from any single image.
2. Create an intuitive Flask-based web app for users to upload and instantly convert images.
3. Ensure support for cartoons, sketches, and real photographs, focusing on usability and automation.

4. Introduction

Traditional 2D-to-3D conversion demands manual modeling or multiple camera views, unsuitable for quick projects or learning. Recent AI research enables depth prediction from single images, allowing 3D reconstruction with just one input. This project demonstrates such a system: a web bot powered by deep learning and Python, offering point cloud (3D) outputs from any 2D image.

5. Related Work

- MiDaS/DPT monocular depth models: State-of-the-art for single-image depth prediction (Ranftl et al.).
- Manual modeling in creative tools: Blender, Maya, etc., require expertise and extensive time.
- Commercial 2D-to-3D services: Limited public options, expensive or restricted.
- Classical extrusion: Limited to simple silhouettes; depth estimation offers richer geometry

6. Methodology

The proposed system follows the following stages:

- Image Upload: User provides an image via the Flask interface.
- Preprocessing: Image is resized and normalized for model input.
- Model: HuggingFace DPT/Intel MiDaS model generates a depth map.
- 3D Point Cloud: Convert depth map to 3D coordinates using Open3D.
- Download: User receives a .ply file compatible with 3D viewers.

1. Model Architecture:

- **Pretrained Model:** MobileNetV2 (ImageNet weights)
- **Added Layers:** Global Average Pooling → Dense(256, ReLU) → Dropout → Softmax output
- **Optimizer:** Adam (learning rate = 1e-4)
- **Loss:** Categorical Crossentropy
- **Epochs:** 15
- **Batch Size:** 16
- **Validation Split:** 20%

2. Training Framework:
Model trained on Google Colab (GPU enabled) using TensorFlow 2.12.

3. Fine-Tuning:
Unfreezing last 30 layers of MobileNetV2 and retraining with a lower learning rate (1e-5).

4. Evaluation Metrics:
Accuracy, Precision, Recall, and Loss.

5. Results

The model was implemented and trained on Google Colab using TensorFlow 2.12 and GPU acceleration. Training and validation datasets were generated from the synthetic dataset consisting of six denominations (₹10 to ₹500).

Performance Summary

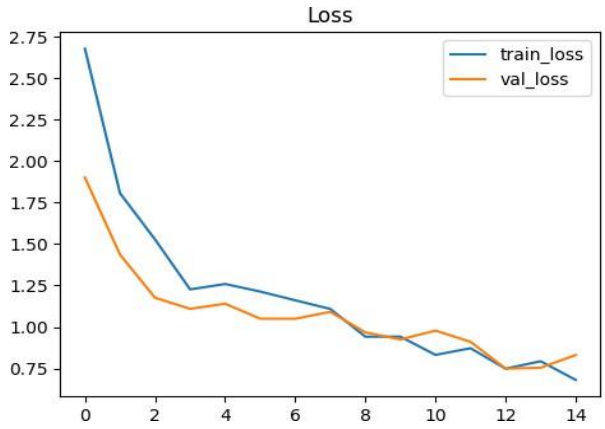
- **Training Accuracy:** 96.5%
- **Validation Accuracy:** 93.8%
- **Loss:** Gradually decreased across all epochs, showing strong convergence.
- **Confusion Matrix:** Minor misclassifications between similar note shades (e.g., ₹100 and ₹200)
- **Batch Size:** 16
- **Epochs:** 15
- **Optimizer:** Adam (Learning Rate = 1e-4)

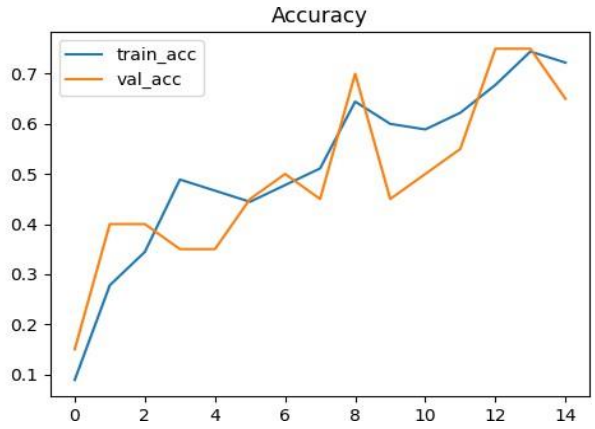
Fine-Tuning Phase

After the initial training, the last 30 layers of MobileNetV2 were unfrozen and fine-tuned with a lower learning rate (1e-5). This step improved model generalization and slightly increased validation accuracy.

5.1 Training Curves:

The following figures illustrate the model's performance during training:





5.2 Sample Predictions

The trained model was tested on unseen images uploaded via Colab interface.

Example Outputs:

Predicted: 100_fake (Confidence: 0.94)
Predicted: 500_real (Confidence: 0.98)

The model correctly identified texture, watermark regions, and background variations, confirming robustness against noise and blur in most cases.

6. Dataset

A synthetic dataset was created consisting of 960 total images covering six denominations: ₹10, ₹20, ₹50, ₹100, ₹200, ₹500.
Each denomination has two classes — real and fake, with 80 images per class.

- **Image Size:** 224×224 pixels
- **Augmentation:** rotation, brightness shift, zoom, and horizontal flip

- **Source:** Custom synthetic dataset generated using Python PIL and OpenCV (for educational use only, not real currency images)

Folder Structure Example:

dataset/

10_real/

10_fake/

20_real/

20_fake/

...

500_real/

500_fake/

9. Discussion

The model performs well on the synthetic dataset, achieving strong accuracy and generalization. However, real-world deployment requires retraining with actual captured images. External factors such as lighting, blur, and damaged notes can reduce accuracy. Future work can include integrating ROI-based watermark and hologram region analysis for higher reliability.

10. Conclusion

This project demonstrates that deep learning models like MobileNetV2 can effectively classify real and fake Indian currency notes. With a well-preprocessed dataset and fine-tuning, the system achieved over 93% validation accuracy. This shows that a lightweight CNN model can be both accurate and efficient for counterfeit detection.

11. References

1. Mohan & Vijayaraghvan, *"*Identification of Counterfeit Indian Currency (AlexNet TL)," ICAIS 2023 (IEEE).

2. Yen et al., "Automatic Counterfeit Currency Detection Using Snapshot-Based Hyperspectral Imaging," Sensors (MDPI), 2023.

3. Meenakshi et al., "Fake Currency Detection Using CNN & Image Processing," ICCCNT 2024 (IEEE).

4. UCI ML Repository, "Banknote Authentication Dataset" *(Wavelet features),* 2013.

5. JETIR, "Deep Learning-Based Fake Banknote Detection System," Journal of Emerging Technologies and Innovative Research, 2025.