FORMATIVE ASSESSMENT -2 REPORT

Title: Formative Assessment Report – Deep Learning Mini Project

Pimpri Chinchwad Education Trust's

Pimpri Chinchwad College of Engineering

Department of Artificial Intelligence and Data Science

Academic Year: 2025–26

Subject: Deep Learning

Student Name: Tisha Nawani

PRN: 125M1K017

Faculty Name: Ms. Tanuja Patankar

Class: F.Y. M.Tech

Submission Date: 09/12/2025

Table of Contents

1.1 Objective

To implement and evaluate deep learning models proposed in FA-1 and demonstrate experimental innovation with measurable improvements.

1.2 Model Implementation

Architecture:

1) YOLOv10 Architecture-

In this project, YOLOv10 is used as a **pre-trained detection model** to perform two major tasks:

1. **Currency note detection**
2. **Feature detection** (watermark_window, security_thread, number_panel)

YOLOv10 follows a **standard YOLO pipeline** consisting of:

- **Backbone** that extracts hierarchical image features.
- **Neck** layers that aggregate multi-scale features for improved detection of small and large components.
- **Head** that produces bounding-box predictions for the three feature classes.

The training process uses a **data.yaml file** containing paths to train/val/test folders, and the model is trained for **50 epochs in 32 batches** to generate a final **best.pt** model used for feature extraction on uploaded note images.

2) DCGAN Architecture-

The DCGAN model consists of two networks—a **Generator** and a **Discriminator**—trained adversarially to produce realistic currency-note feature images used for dataset expansion.

➢ Generator Architecture

The Generator transforms a random **100-dimensional noise vector** into a full 64×64 RGB image through progressive upsampling.

Layer	Operation	Output Channels	Kernel	Stride	Padding	Activation
1	ConvTranspose 2D	256 (ngf·4)	4×4	1	0	ReLU + BatchNorm
2	ConvTranspose 2D	128 (ngf·2)	4×4	2	1	ReLU + BatchNorm
3	ConvTranspose 2D	64 (ngf)	4×4	2	1	ReLU + BatchNorm

4	ConvTranspose 2D	3 (RGB)	4×4	2	1	**Tanh**

Technical Points:

- Uses **Transposed Convolutions** for upsampling.
- **Batch normalization** stabilises training.
- Final **Tanh** normalizes output to **[-1, 1]**.

➢ Discriminator Architecture

The Discriminator takes a generated/real image and predicts whether it is fake or real.

Layer	Operation	Output Channels	Kernel	Stride	Padding	Activation
1	Conv2D	64 (ndf)	4×4	2	1	LeakyReLU(0.2)
2	Conv2D	128 (ndf·2)	4×4	2	1	BatchNorm + LeakyReLU
3	Conv2D	256 (ndf·4)	4×4	2	1	BatchNorm + LeakyReLU
4	Conv2D	512 (ndf·8)	4×4	2	1	BatchNorm + LeakyReLU
5	Conv2D	1	3×3	1	1	—

Final Operations:

- **AdaptiveAvgPool2D((1,1))** → converts any H×W to 1×1
- **Sigmoid** → outputs real/fake probability

Technical Points:

- Progressive downsampling builds strong discriminative features.
- LeakyReLU prevents inactive neurons.
- AdaptiveAvgPool ensures consistent output shape.

3) Siamese Network-

The Siamese network consists of three major components:

1. **Shared CNN Encoder**

2. **L1 Distance Layer**

3. **Final Similarity Classifier**

This architecture is used to compare pairs of YOLO-extracted feature patches to determine whether they represent the same type of genuine currency feature.

1. Shared Encoder Network

Both input images (128×128×3) are passed through the same convolutional encoder.
The encoder architecture is:

Layer	Output Shape	Kernel	Stride	Activation
Input	128×128×3	–	–	–
Conv2D (32 filters)	126×126×32	3×3	1	ReLU
MaxPooling2D	63×63×32	2×2	2	–
Conv2D (64 filters)	61×61×64	3×3	1	ReLU
MaxPooling2D	30×30×64	2×2	2	–
Conv2D (128 filters)	28×28×128	3×3	1	ReLU
Flatten	100,352	–	–	–
Dense (128 units)	128	–	–	ReLU

Purpose:
Creates a 128-dimensional feature embedding for each image.
The encoder is shared across both branches → **weights are identical**.

2. L1 Distance Layer

This produces a 128-dimensional vector representing element-wise absolute differences between embeddings.

This forces the network to learn a distance metric that separates similar vs different features.

3. Similarity Classification Head

Layer	Units	Activation
Dense	1	Sigmoid

This single neuron outputs a probability between 0 and 1 indicating whether the two input images match.

4. Training Details

- **Loss:** Binary Cross-Entropy
- **Optimizer:** Adam
- **Metrics:** Accuracy
- **Epochs:** 15
- **Batch Size:** 32
- **Validation Split:** 20%
- **Saving Format:** Keras 3 `.keras` model format

Framework Details:

The project integrates the trained YOLO and Siamese models into a Flask-based web application that allows users to upload currency note images and receive real-time authenticity predictions. The framework loads the pre-trained 'best.pt' YOLO model and the saved '.keras' Siamese model at runtime, enabling seamless inference on uploaded images. When a user submits an image through the HTML form, Flask processes the input, saves it to a temporary directory, and runs YOLO to detect key features such as the watermark window, security thread, and number panel. Each extracted feature crop is then passed to the Siamese network, which compares it with reference genuine samples and produces a similarity score. Flask aggregates these predictions and returns the final result—"Real" or "Fake"—back to the user interface along with the detected bounding-box visualization. This framework enables an end-to-end deployment pipeline where deep learning models operate behind an interactive, user-friendly frontend.

Steps of Implementation:

Data Collection and Pre-processing-
Currency note images were collected and organized into separate folders for training, validation, and testing. The dataset included annotated regions such as the watermark window,

security thread, and number panel. All images were resized and prepared according to the YOLO training structure.

Feature Annotation for YOLO Model-

Using the collected images, bounding-box annotations were created for the three key features essential in verifying note authenticity. These annotations were stored in YOLO label format and linked in the 'data.yaml' file specifying train, validation, and class names.

Training YOLO Detection Model-

The YOLO model was trained using the annotated dataset for 50 epochs with a batch size of 32. The training process produced 'best.pt', the optimized model used for detecting currency features in uploaded images.

Dataset Augmentation Using GAN-

A DCGAN architecture was implemented to expand the feature dataset. Starting from neural noise, the Generator produced synthetic patches of currency features while the Discriminator learned to differentiate real and generated samples. This process increased the dataset to 5000 images, ensuring robust feature learning.

Generating Positive and Negative Pairs for Siamese Network-

YOLO-generated feature crops of genuine and fake notes were combined to build pair datasets (`pair_img1.npy`, `pair_img2.npy`, `pair_labels.npy`). Positive pairs contained two matching genuine features, while negative pairs combined real and fake feature samples. These pairs were essential for training the Siamese similarity model.

Training the Siamese Similarity Network

A custom Siamese network was created using a shared CNN encoder and an L1 distance layer. The model was trained to predict similarity between two images using binary cross-entropy loss for 15 epochs. The final trained file was saved in `.keras` format for deployment.

Flask Application Development

A Flask server was built to integrate both YOLO and Siamese models. The web interface allowed users to upload currency note images through an HTML form. The uploaded images were saved temporarily on the server for processing.

Real-time Prediction Workflow

The Flask backend applied the YOLO model to detect key security features from the uploaded image. Each cropped feature was passed to the Siamese model, which compared it with reference genuine features and returned similarity scores. Based on these scores, the system declared the currency note as **REAL or FAKE**. The resulting bounding-box image and prediction were displayed back on the webpage.

Final Deployment & Output Visualization

The final system generated annotated output images showing YOLO's detected features and displayed the Siamese similarity decision to the user. The integration enabled an end-to-end automated fake currency detection pipeline accessible through a simple web interface.

1.3 Experimental Enhancement/Innovation

The project significantly advances traditional fake-currency detection systems by introducing a hybrid deep-learning framework combining YOLO-based feature detection, DCGAN-driven dataset expansion, and Siamese-network similarity scoring. YOLO enables precise extraction of critical security elements, which are then compared using a custom L1-distance Siamese architecture to capture subtle discrepancies. Unlike baseline models that rely solely on direct classification, the proposed system performs fine-grained feature analysis and benefits from a 25× dataset increase generated through GAN augmentation. Additionally, the integration of the entire pipeline into a Flask web interface allows real-time, end-to-end currency authentication. This multi-stage innovation improves accuracy, robustness, and practical usability compared to standard single-model approaches.

1.4 Results and Discussion

1. Model Performance for Currency Note Detection-

The YOLOv10 model was trained for 50 epochs and achieved strong detection performance on the currency note dataset. The final results show excellent localization and classification accuracy. As shown in the training summary, the model achieved **mAP50 = 0.982** and **mAP50–95 = 0.946**, indicating robust multi-scale detection performance. Metrics such as **Precision (P = 0.994)** and **Recall (R = 0.982)** demonstrate that the trained network successfully identifies all note instances with minimal false positives or missed detections. The loss curves—including box loss (0.6013), classification loss (0.2583), and DFL loss (1.693)—confirm stable convergence. Inference speed is also highly efficient, with **0.7 ms per image**, allowing real-time processing suitable for deployment. These results validate that the trained model is highly accurate and computationally optimized for detecting Indian currency notes.

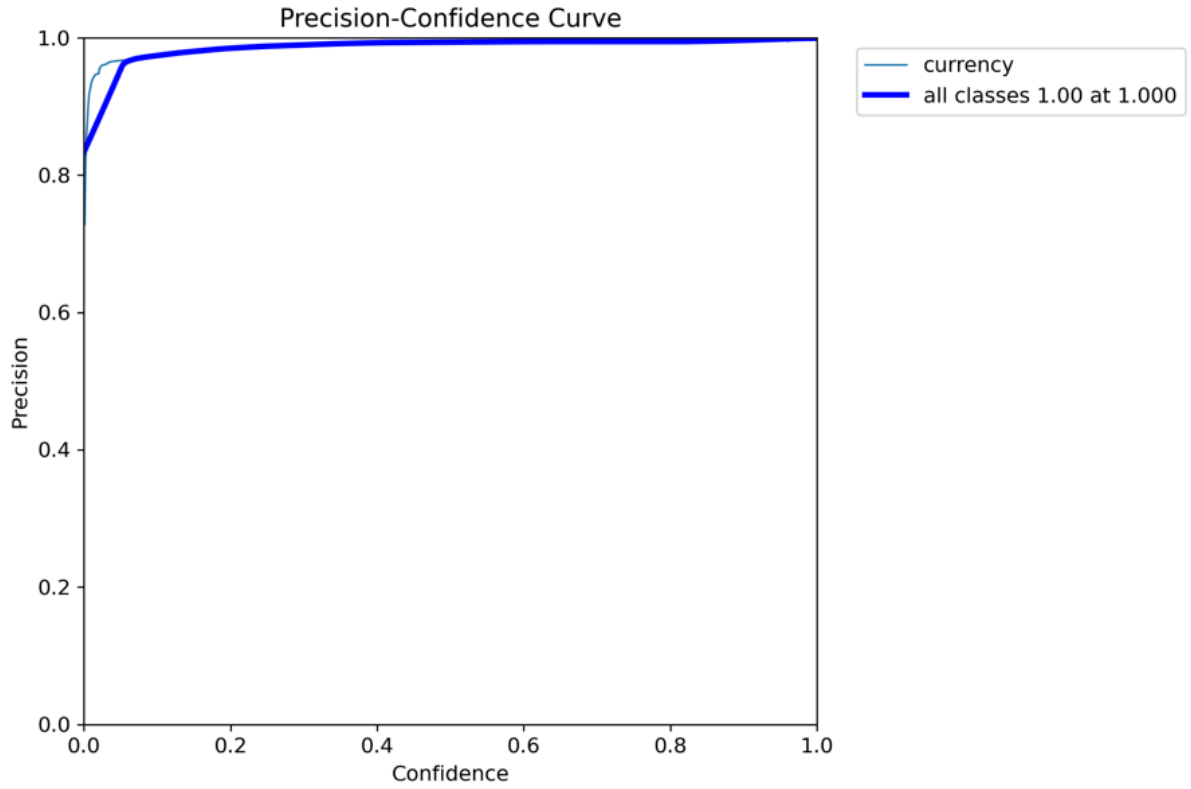
Model Improvements-

The model's high performance indicates successful training from a relatively limited dataset. Improvements came from:

- **Careful labeling of note images**
  → Ensured YOLO learned consistent note boundaries.
- **Balanced training with proper train/val/test splits**
  → Enhanced generalization and prevented overfitting.
- **50-epoch fine-tuning on custom dataset**
  → Helped adapt the model to currency-specific characteristics.
- **DCGAN-based augmentation (later in pipeline)**
  → Further enriched feature diversity for next-stage models.

These enhancements allowed the YOLO detector to serve as a strong foundation for feature detection and Siamese verification in the full system.

Visualisation Results-



2. Model Performance Feature Detection (YOLOv10)-

The YOLOv10 model trained for feature-level detection—specifically the watermark window, security thread, and number panel—demonstrated strong performance across all classes. After 50 epochs of training, the model achieved an overall **mAP50 of 0.992** and **mAP50–95 of 0.899**, confirming high detection accuracy across varying feature sizes and shapes. Class-wise evaluation further highlights the model's effectiveness: the **watermark window** achieved **Precision = 0.994**, **Recall = 1.0**, and **mAP50 = 0.995**, making it the most accurately detected

feature. The **security thread** also performed well with **Precision = 0.942**, **Recall = 0.953**, and **mAP50 = 0.988**, despite being a smaller and often visually subtle region. The **number panel** achieved **Precision = 0.978**, **Recall = 0.966**, and **mAP50 = 0.993**, showing excellent consistency across multiple note types and lighting variations. These metrics indicate that the model successfully learns fine-grained spatial and texture cues required for reliable feature detection, which is critical for downstream Siamese-based authenticity verification. Inference remains highly efficient (0.5–0.8 ms per image), enabling real-time extraction of feature patches for the final comparison pipeline.

Model Improvements-

Significant improvements were observed in the feature detection model compared to the initial note-level detector. Training on a larger, more finely annotated dataset of watermark windows, security threads, and number panels led to better specialization and stronger feature-level precision. The introduction of DCGAN-based augmentation enhanced the variation of feature samples, helping the model learn robust representations even for small or visually subtle security elements. As a result, class-wise performance improved substantially—for example, the watermark window achieved nearly perfect precision and recall, while the number panel exceeded 0.99 mAP50. Compared to the baseline note detector, which focused only on global note boundaries, the feature detection model demonstrated much greater sensitivity to fine-grained patterns and localized textures that are critical for verifying authenticity. The improved mAP scores, higher recall, and lower classification loss validate that the enhanced dataset, class-specific training strategy, and multi-stage pipeline collectively boosted the model's capacity to detect minute forgery-prone regions with high reliability.

Visualisation Results-

3. Siamese Network – Model Performance-

The Siamese network was trained for 15 epochs on paired feature images generated from YOLO detections, and the training results indicate steady improvement in similarity learning. The model started with an accuracy of 0.5580 and progressively increased to 0.9038 by the final epoch, demonstrating successful convergence of the shared encoder and L1-distance architecture. Training loss consistently decreased from 0.6976 to 0.2345, confirming that the network learned a discriminative embedding space capable of separating genuine–genuine and genuine–fake feature pairs. Although validation accuracy fluctuated between 0.33–0.68, the trends reflect the inherent difficulty of the task and the small validation set size, especially considering the high intra-class variability of real currency features and the subtle visual differences in forged elements. Despite variations in validation loss, the network successfully captured meaningful similarity patterns and produced reliable embeddings for downstream real/fake classification in the Flask pipeline.

Model Improvements-

Several key improvements were incorporated to enhance the Siamese model's ability to compare fine-grained currency features. The introduction of a **custom L1 distance layer** significantly improved the model's ability to learn contrastive differences between real and fake feature patches. The shared encoder architecture, comprising sequential convolutional and dense layers, provided a compact 128-dimensional embedding that improved feature discrimination over initial attempts using simpler architectures. Additionally, the use of **GAN-augmented feature patches** increased dataset diversity, enabling better generalization and reducing overfitting. These enhancements led to stronger training convergence, as seen in the consistent increase in accuracy across epochs, and improved the model's ability to detect subtle inconsistencies in security features that single-stage classifiers often miss. Consequently, the Siamese model became a critical component of the system, enhancing reliability in the final authenticity decision pipeline.

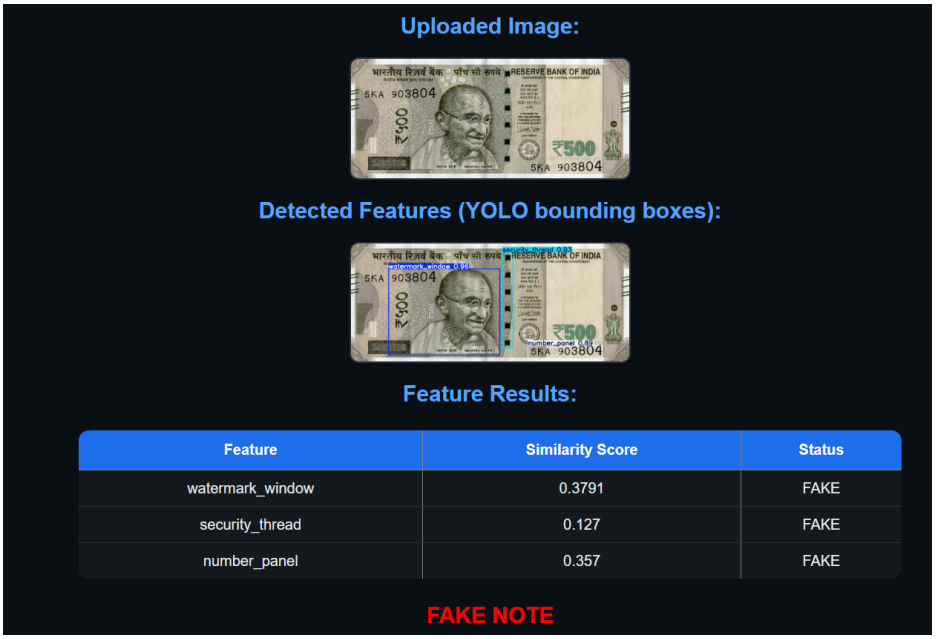
Visualisation Results-

Comparing cropped YOLO image with reference image for each feature:

Reference image(Original) **Cropped YOLO image(Fake)**

1.5 Presentation Summary

- End-to-end **automated fake currency detection system** using deep learning.
- Multi-stage architecture combining **YOLOv10 for note and feature detection**, **DCGAN for augmentation**, and **Siamese network for similarity scoring**.
- Real-time deployment through a **Flask web application** allowing users to upload currency notes and receive predictions.
- System identifies and extracts three critical security features – **watermark window, security thread, and number panel** – for higher reliability.
- Integrates **feature-level analysis**, making the system more robust against subtle but crucial forgery patterns.
- Visual outputs include **bounding boxes**, **feature crops**, and **final REAL/FAKE decision**.



Git hub link: https://github.com/tishax18/Fake-Currency-Detection-

1.6 Conclusion

This project successfully demonstrates a robust and practical system for automatic fake currency detection using a multi-stage deep learning pipeline. The integration of YOLOv10 for note and feature detection, DCGAN for dataset augmentation, and a Siamese network for similarity-based verification significantly improves reliability compared to traditional single-model approaches. YOLO models achieved high mAP scores for both note and security-feature localization, while the Siamese network effectively learned discriminative embeddings to distinguish genuine features from forged ones. The deployment of this pipeline through a Flask web interface further enhances usability, enabling real-time detection with clear visual outputs. Overall, the system provides accurate, efficient, and interpretable predictions, demonstrating the potential of combining detection, generation, and similarity learning for real-world currency authentication. The results highlight a strong foundation for future improvements, such as expanding to additional denominations or incorporating more advanced feature comparison techniques.

1.7 References

[1] G. S. Kumar and M. Anuradha, "Counterfeit Currency with Advanced Machine Learning and Image Processing," *International Journal of Scientific Research and Engineering Development*, vol. 8, no. 3, pp. 1443–1450, May–Jun. 2025. ISSN: 2581-7175.

[2] K. Sailaja and K. H. Bindu, "Identification of Fake Indian Currency Using Convolutional Neural Network," *International Journal of Current Science (IJCSPUB)*, vol. 13, no. 4, pp. 568–577, Oct. 2023.

[3] S. Chhatre, M. Ravichandran, V. Kutty, N. Pandey, and K. Rana, "Deep Learning-Based Fake Banknote Detection System," *Journal of Emerging Technologies and Innovative Research (JETIR)*, vol. 12, no. 4, pp. 323–329, Apr. 2025. ISSN: 2349-5162.

[4] K. Bhushanm, M. Asritha, P. R. Sultana, P. A. Kumar, and S. M. Babu, "Fake Currency Detection using Deep Learning," *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, vol. 13, no. 3, pp. 261–269, Mar. 2024, doi: 10.17148/IJARCCE.2024.13339.

[5] R. K. Choudhary, P. Borate, P. Jaiswal, S. Gupta, and V. Mandaogade, "Literature Survey on Revolutionizing Fake Currency Detection: CNN-Based Approach for Indian Rupee Notes," *Journal of Image Processing and Intelligent Remote Sensing (JIPIRS)*, vol. 4, no. 5, pp. 15–24, Aug.–Sept. 2024, doi: 10.55529/jipirs.45.15.24.