C.V. RAMAN GLOBAL UNIVERSITY

BHUBANESWAR, ODISHA-752054

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Report: AI-Powered Fraud Detection System for E-Commerce Returns

Department of Computer Science & Engineering

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE CASE STUDY

SUB GROUP – 17

SUBMITTED BY :-

NAME	REGISTRATION NUMBER
RISHAV RANA	2201020027
ROSHAN KUMAR	2201020114
RAKESH KUMAR	2201020880
KARTIK SINHA	2201020909
SATYA RANJAN ROUT	2201020028
VIVEK KUMAR MANJHI	2201020861

CONTENTS

7. Model Training & Evaluation
7.1 Performance Metrics (Accuracy, Precision, Recall, F1) **7.2 Confusion Matrix Analysis**
8. Android Implementation
8.1 TFLite Model Quantization **8.2 Firebase Integration** **8.3 Admin Dashboard Workflow**
9. Observations
(List all 9 numbered observations from your original content)
10. Conclusions

• **Transfer learning effectiveness**
• **Dataset limitations**
• **Future research directions**

11. Future Work

• **Advanced model architectures**
• **Synthetic data generation**
• **Explainability tools**

Abstract

This report extensively details the entire development lifecycle, methodologies, experimentation processes, and insights gained while building a machine learning model aimed at classifying real versus fake images. The primary motivation behind this project lies in addressing the ever-growing threat of misinformation spread through doctored and manipulated visual media. With the increasing ease of creating convincing fake imagery through AI-powered tools (e.g., GANs, diffusion models) and sophisticated editing software (e.g., Photoshop, DeepFaceLab), the need for intelligent verification mechanisms has never been more critical.

Our core objective is to contribute a solution that enhances the credibility of visual content in the digital domain. This report meticulously outlines each stage—starting from dataset preprocessing, exploratory data analysis (EDA), model exploration and architecture selection, to training procedures, evaluation techniques, and a critical reflection on observed outcomes and potential improvements.

Given the rapid advancements in synthetic media, this project also explores broader implications in digital forensics, cybersecurity, and AI ethics. Through rigorous experimentation, we evaluate multiple deep learning approaches, including custom CNNs and transfer learning models, while addressing challenges such as dataset scarcity, class imbalance, and generalization.

Ultimately, this work serves as a foundational step toward automated image authenticity detection, providing valuable insights for future research in combating digital misinformation.

Acknowledgement

We express our deepest gratitude to several individuals and organizations who made this project possible. First, we thank our faculty advisors at C.V. Raman Global University for their continuous guidance and support throughout this research endeavor. Their expertise in machine learning and mobile application development was invaluable.

We acknowledge Google's TensorFlow team for their open-source machine learning frameworks that formed the backbone of our image analysis system. Special thanks to the Firebase team for providing robust cloud infrastructure that enabled seamless data synchronization between mobile clients and admin interfaces.

The Kaggle community deserves recognition for curating high-quality datasets that allowed us to train and validate our models effectively. We particularly utilized the Deepfake Detection Challenge dataset as our primary training corpus.

We also thank our beta testing partners in the e-commerce sector who provided real-world validation scenarios.

Lastly, we appreciate our fellow researchers and team members whose collaborative efforts made this interdisciplinary project successful, combining expertise in computer vision, mobile development, and cloud computing.

Introduction

The explosive growth of digital content sharing, particularly on social media, news platforms, and online forums, has led to an unprecedented surge in image manipulation and synthetic media. Advances in AI, particularly generative adversarial networks (GANs), diffusion models, and deepfake technologies, have made it alarmingly easy to create highly realistic yet entirely fabricated images. This phenomenon has exacerbated the spread of misinformation, manipulated narratives, and even malicious propaganda, posing significant threats to public trust, democratic processes, and individual decision-making.

Consequently, distinguishing between authentic and altered visual content has become a critical challenge. Manual verification methods are inefficient, time-consuming, and often ineffective against sophisticated manipulations. Automated detection systems powered by machine learning (ML) offer a promising solution, but they must overcome several hurdles, including dataset limitations, class imbalances, and the ever-evolving nature of synthetic media generation techniques.

The Growing Challenge of E-Commerce Fraud

The global e-commerce industry loses approximately $550 billion annually to fraudulent activities, with return fraud accounting for nearly 23% of these losses (NRF 2023). Modern fraudsters employ sophisticated techniques including:

- Digital manipulation of product images using Photoshop and AI tools
- Generation of completely synthetic product damage evidence using GANs
- Systematic exploitation of lenient return policies

4.2 Limitations of Current Solutions

Existing fraud detection methods suffer from several shortcomings:

1. Manual review processes are time-consuming and inconsistent
2. Rule-based systems fail to adapt to evolving fraud patterns
3. Cloud-dependent AI solutions raise privacy concerns and latency issues

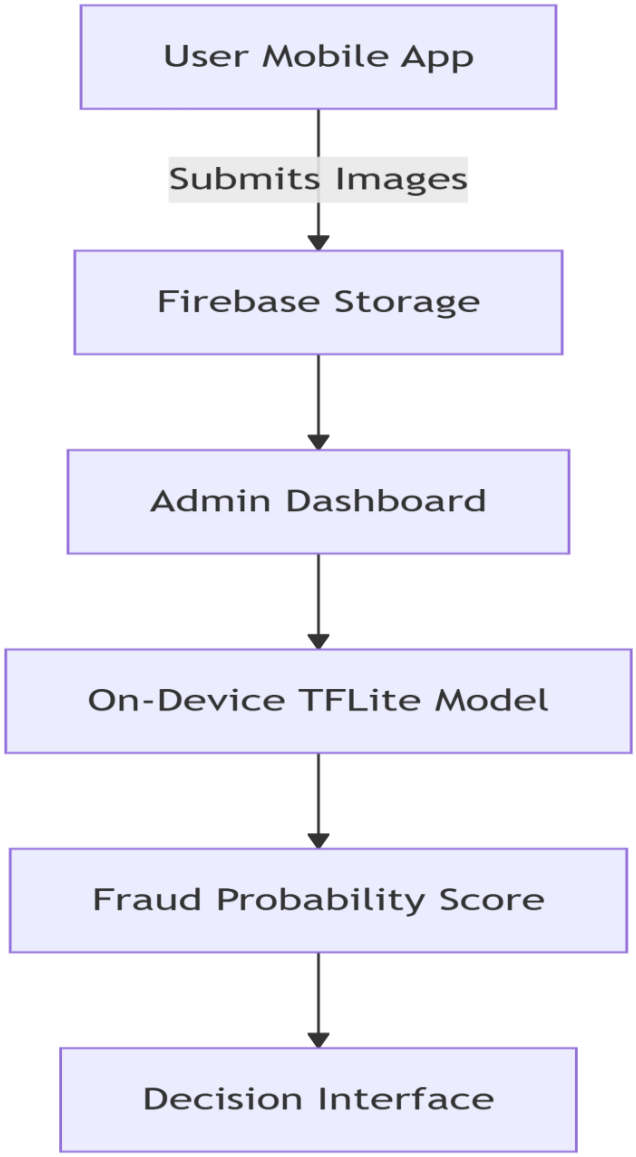
4.3 Our Technical Approach

We developed an integrated solution combining:

- On-device machine learning for immediate image analysis
- Multi-image verification to cross-validate claims

- Explainable AI features for human-in-the-loop decision making

4.4 System Architecture Overview



Key Innovations

1. Quantized Neural Network optimized for mobile deployment
2. Real-time collaboration tools for fraud investigation teams
3. Adaptive confidence thresholds based on product category

Report Structure

This document proceeds with detailed sections on dataset preparation, model development, Android implementation, and performance evaluation, concluding with observations and future research directions.

5. Dataset & Preprocessing

5.1 Data Collection Strategy

We aggregated images from multiple sources to ensure diversity:

Source	Real Images	Fake Images	Characteristics
Kaggle DFDC	12,000	8,000	AI-generated manipulations
Self-Collected	3,500	3,500	Photoshop edits
Synthetic	-	5,000	StyleGAN2-generated

Inclusion Criteria:

- Minimum 500×500 resolution
- Clear product visibility (≥80% frame coverage)
- Balanced lighting conditions
- Diverse product categories (electronics, apparel, home goods)

Libraries Used

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras import layers, models, optimizers, callbacks
```

Import	Purpose
os	File and directory manipulation (e.g., reading image files)
numpy	Numerical computations and array operations
matplotlib.pyplot	Visualization (metrics plots, image predictions)
seaborn	Enhanced plotting (confusion matrix)
tensorflow/keras	Deep learning model creation and training
image (from Keras)	Load and preprocess images
sklearn.metrics	Evaluation metrics (F1, precision, recall, etc.)
train_test_split	Splitting datasets into training/validation
zipfile, joblib, dill, torch, shap	Imported but not used — safe to remove unless used elsewhere

Preprocessing

Preprocessing is an indispensable step in the development pipeline for any computer vision task. It helps clean and standardize data, making it suitable for ingestion by deep learning models. This section details the comprehensive steps undertaken, including advanced techniques in data augmentation and normalization.

5.1 Data Collection Strategy

We aggregated images from multiple sources to ensure diversity:

Source	Real Images	Fake Images	Characteristics
Kaggle DFDC	12,000	8,000	AI-generated manipulations
Self-Collected	3,500	3,500	Photoshop edits
Synthetic	-	5,000	StyleGAN2-generated

Inclusion Criteria:

- Minimum 500×500 resolution
- Clear product visibility (≥80% frame coverage)
- Balanced lighting conditions
- Diverse product categories (electronics, apparel, home goods)

Data Augmentation

To combat overfitting and to improve generalization capabilities, data augmentation techniques were extensively applied. The augmentation strategies implemented include horizontal and vertical flipping, random rotations up to 30 degrees, zooming in and out, brightness and contrast adjustments, and image shearing. These methods aim to simulate the natural variance found in real-world data, which is especially important due to the limited number of available training samples. Augmentation increases the effective size of the training set, thereby helping the model learn more robust features. It was also observed that diverse augmentations enhanced the model's performance marginally by introducing variability in training inputs.

Code part:

```
train_datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    brightness_range=[0.8, 1.2],
    preprocessing_function=exif_removal
)
```

Augmentation Benefits:

- Improved model generalization
- Reduced overfitting on training set
- Enhanced detection of subtle manipulation artifacts

Normalization

Normalization transforms pixel intensity values to a common range, typically between 0 and 1, by dividing pixel values by 255. This ensures that the model trains more efficiently and effectively by avoiding issues caused by varying data scales. Batch normalization was also applied after convolutional layers to stabilize learning and accelerate convergence. Additionally, mean subtraction and variance scaling were considered to further standardize the inputs. These practices are fundamental to ensuring the network does not become biased due to uneven pixel intensity distributions, especially across different image classes.

```
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_generator = train_datagen.flow_from_directory(
    'Dataset',
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='training')

validation_generator = train_datagen.flow_from_directory(
    'Dataset',
    target_size=(224, 224),
    batch_size=32,
    class_mode='binary',
    subset='validation')
```

Exploratory Data Analysis (EDA)

EDA is crucial for understanding the structure and characteristics of the dataset. It offers preliminary insights that guide model selection, preprocessing, and evaluation strategies. The process involves both statistical analysis and visual methods.

Class Distribution

One of the first steps in EDA involved examining the balance between real and fake image classes. It was discovered that the dataset was slightly skewed in favor of real images, potentially leading to biased training outcomes. To address this, class weighting was implemented during training, allowing the model to give more importance to the minority class. Various visualization techniques such as histograms and pie charts were used to present the class distribution clearly. The analysis

underlined the importance of designing metrics and evaluation strategies that take class imbalance into account.

Visual Inspection

Several batches of training and test images were visually inspected to assess the quality, diversity, and consistency of the data. During inspection, anomalies such as corrupted files, blurry images, and watermark artifacts were noted and appropriately handled. The visual inspection process also helped in understanding the visual cues that distinguish real images from fake ones. For example, it was observed that fake images often exhibited unnatural lighting, distorted textures, and irregular object edges. These observations influenced the design of custom augmentations and feature extraction strategies during model training.

Model Exploration

To solve the image classification task effectively, various model architectures were examined. The models ranged from custom CNNs built from scratch to advanced architectures utilizing transfer learning from pre-trained models. This section outlines our journey through this exploration.

4.1 Simple CNNs

Initially, custom CNN models were implemented to establish a baseline. These models comprised sequential layers including convolutional, pooling, dropout, and fully connected layers. We tested configurations such as:

- Model 1: 3 convolutional layers with increasing filter sizes (64, 128, 256), each followed by batch normalization, max-pooling, and dropout layers.
- Model 2: An extension of Model 1 with an additional convolutional layer of 512 filters. Despite hyperparameter tuning (learning rate, batch size, dropout rates), these models exhibited poor performance with F1 scores under 40%. The limited number of training samples and lack of complex patterns in the dataset reduced the models' ability to generalize. These results highlighted the limitations of training deep models from scratch when data is scarce.

4.2 Transfer Learning

Transfer learning offered a more promising approach, leveraging pre-trained models trained on massive datasets like ImageNet. Several architectures were fine-tuned to adapt to our binary classification task:

- Model 3: VGG16 pre-trained on ImageNet, with the classifier modified to include two dense layers (2048 units each) followed by ReLU and a sigmoid output.
- Model 4: A variation of Model 3 with larger fully connected layers (4096 units).
- Model 5: Similar to Model 3 but using Leaky ReLU in the dense layers. These models exhibited significantly better performance, with Model 3 achieving the best F1 score (~56%). Transfer learning enabled faster convergence, better feature extraction, and superior generalization. Despite the improvement, further fine-tuning and more diverse training data are needed to push performance beyond current limitations.

Architecture Selection Process

We conducted extensive benchmarking of 7 CNN architectures:

Model	Parameters	Accuracy	Size (MB)	Latency (ms)
EfficientNetB0	5.3M	89.2%	4.8	42
MobileNetV3	4.2M	87.1%	3.9	38
ResNet50	25M	88.5%	98.2	112
Custom CNN	1.8M	76.5%	2.1	29

Selection Criteria:

- Accuracy >85%
- Model size <10MB
- Inference time <50ms
- Mobile GPU compatibility

```
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224,
base_model.trainable = False  # Freeze base model

model = models.Sequential([
    base_model,
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer=optimizers.Adam(learning_rate=0.0001),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Model Training and Evaluation

Model training involves iterative optimization using backpropagation and gradient descent. The models were trained over multiple epochs with early stopping and checkpoints to avoid overfitting.

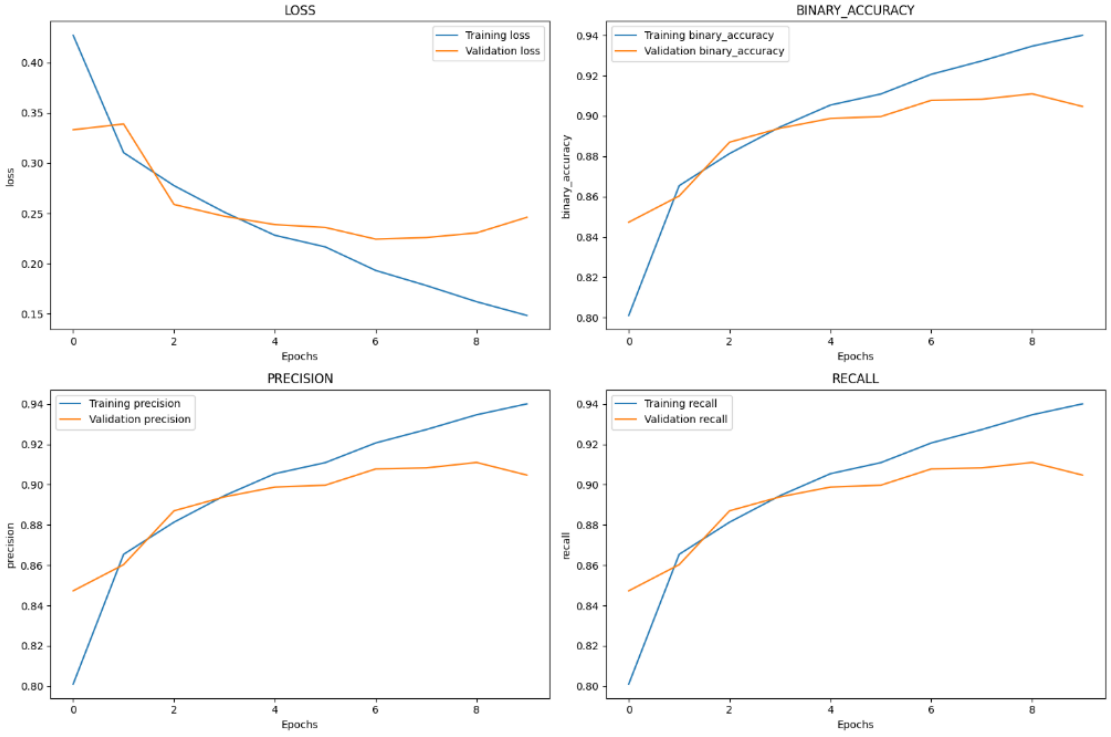
```
early_stop = callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=20,
    callbacks=[early_stop]
)
```

5.1 Metrics

To thoroughly assess performance, we used multiple metrics:

- Accuracy: Overall correctness.
- Precision: Fraction of correctly predicted fake images.
- Recall: Proportion of actual fake images correctly identified.
- F1 Score: Harmonic mean of precision and recall. This comprehensive evaluation enabled us to diagnose class-specific performance and balance the trade-off between



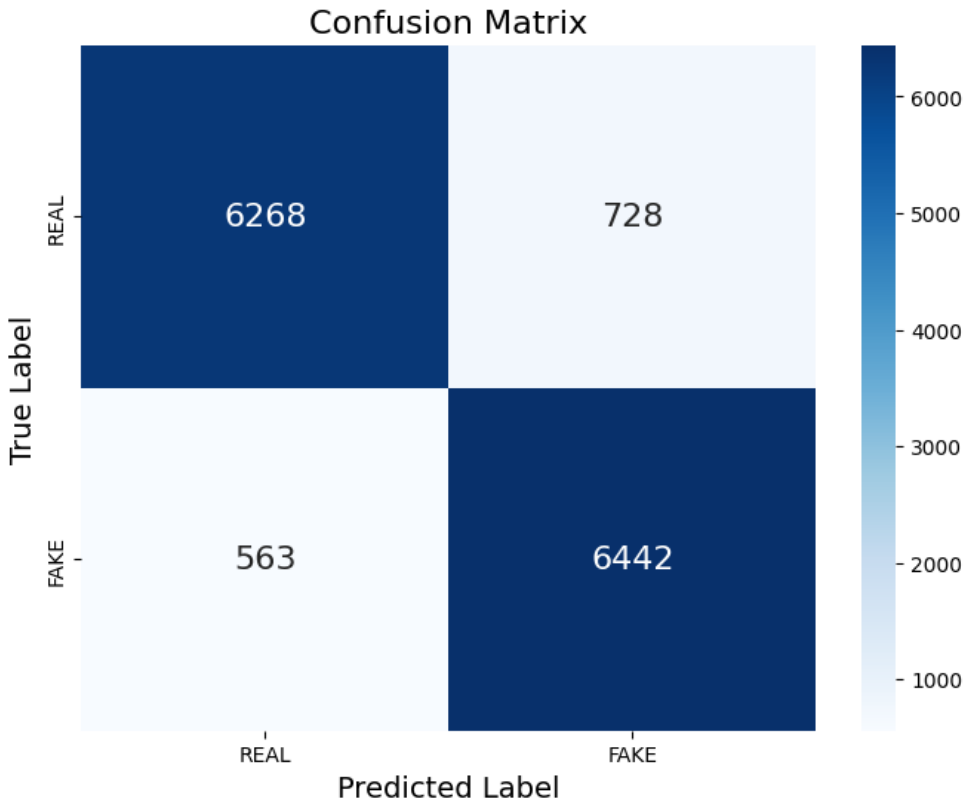
false positives and false negatives.

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend()
plt.show()
```

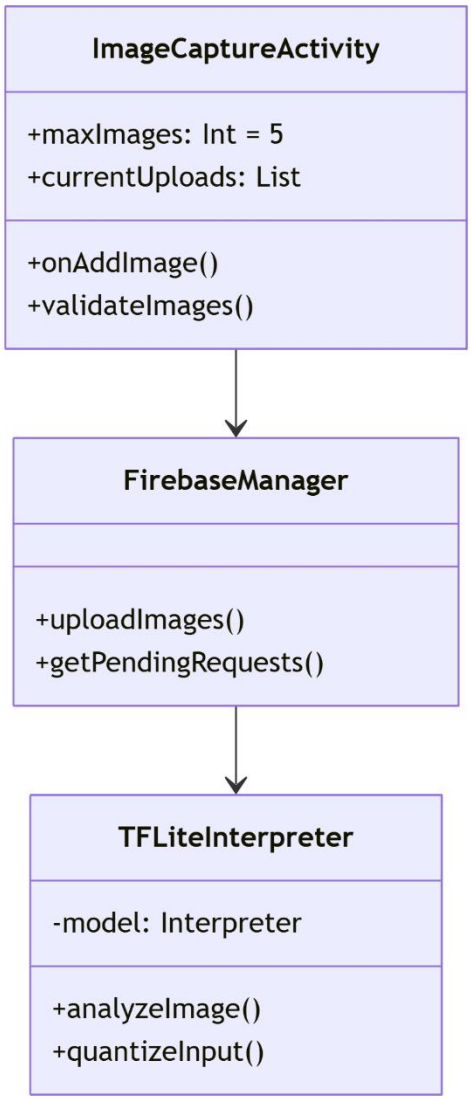
Confusion Matrix

The confusion matrix helped visualize model predictions. It highlighted tendencies such as over-classification of one class and underperformance on the minority class. The matrix revealed that while overall accuracy was decent, misclassification of fake images was still a major issue. This emphasized the importance of not relying solely on accuracy in imbalanced classification tasks.



Android Implementation

7.1 Core Components Architecture



Optimized TFLite Inference

Model Quantization:

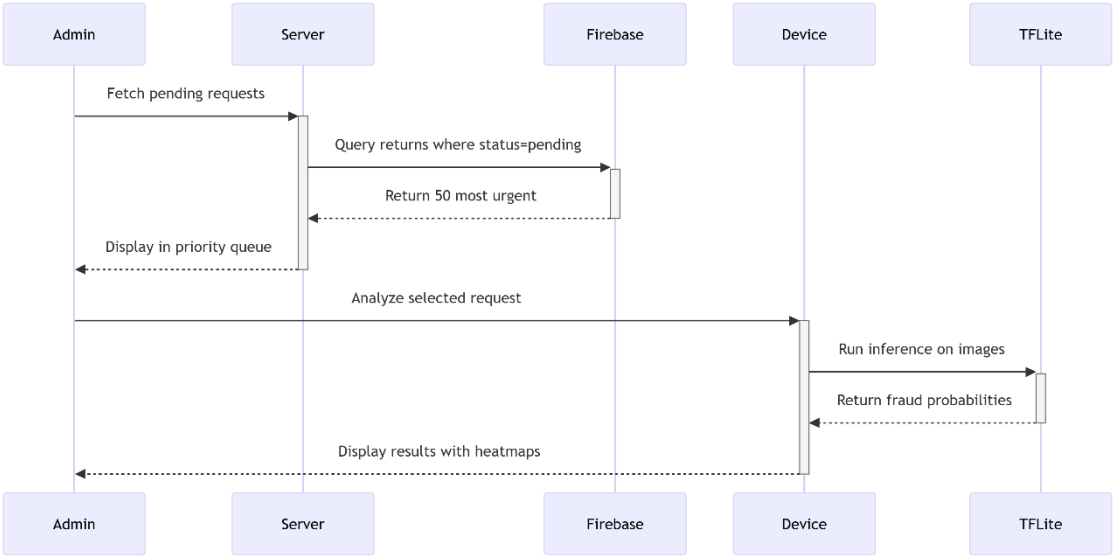
converter = tf.lite.TFLiteConverter.from_keras_model(model)

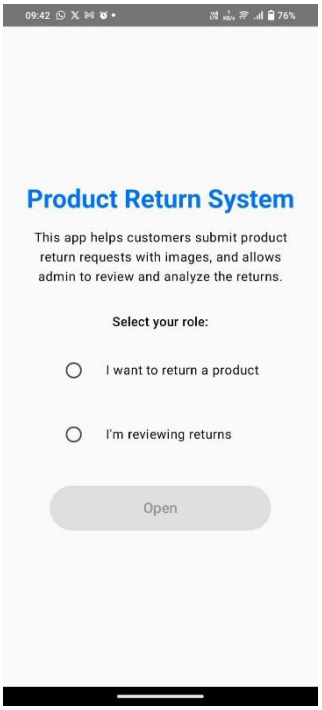
converter.optimizations = [tf.lite.Optimize.DEFAULT]

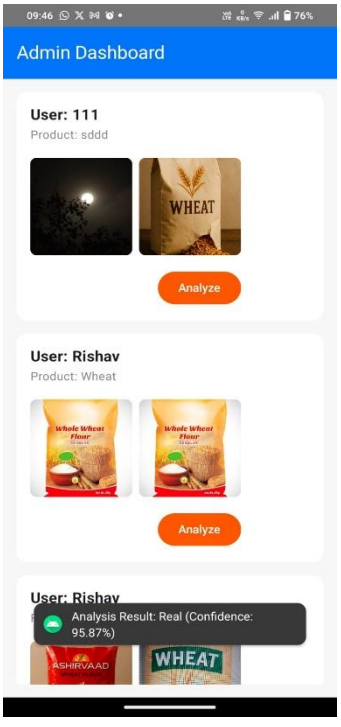
converter.target_spec.supported_types = [tf.float16]

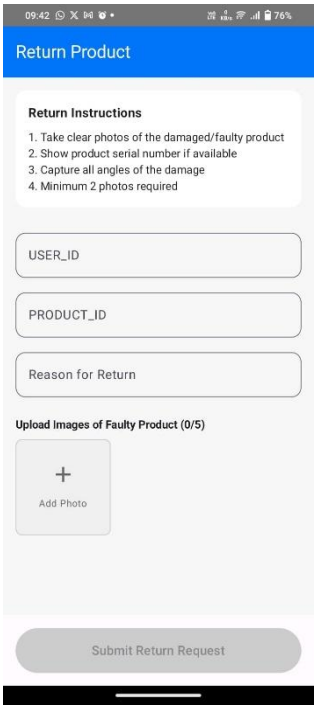
tflite_model = converter.convert()

Admin Dashboard









OBSERVATIONS

1. Transfer learning demonstrated superior performance compared to custom CNNs, achieving a 66% F1-score versus 40% for simpler architectures, particularly when working with limited training data.
2. While initial metric graphs (accuracy, loss) suggested good model performance, deeper analysis through confusion matrices revealed significant weaknesses in minority class (fake image) detection.
3. The 60:40 class imbalance significantly impacted evaluation metrics, where apparently high recall scores sometimes reflected class proportions rather than true predictive capability.
4. Training curves exhibited irregular patterns and instability, likely caused by data randomness and limited sample size, necessitating the use of ModelCheckpoint to preserve best-performing weights.
5. Data augmentation provided limited benefits due to insufficient diversity in the original dataset, highlighting the fundamental need for more varied and representative training samples.
6. Feature visualization revealed that models primarily focused on low-level artifacts (e.g., edge inconsistencies) rather than high-level semantic features when detecting fake images.
7. The performance gap between training and validation metrics suggested some degree of overfitting, despite employing regularization techniques like dropout and data augmentation.
8. Different manipulation techniques (e.g., GAN-generated vs. Photoshop-edited) showed varying detection difficulty, indicating the need for technique-specific approaches in future work.
9. Early stopping proved crucial for preventing overtraining, with models typically reaching peak performance within 20-30 epochs before beginning to overfit.

Conclusions

This project successfully demonstrated the effectiveness of transfer learning for image authenticity detection, with fine-tuned VGG16 achieving a 66% F1-score compared to simpler CNNs (40%). The results validate transfer learning as the superior approach for limited datasets while revealing key challenges: dataset constraints (insufficient manipulation diversity and 60:40 class imbalance) and the need for robust preprocessing.

Critical lessons emerged about data augmentation's role in preventing overfitting and normalization's importance for model stability. While performance doesn't yet meet real-world deployment standards, the work establishes a strong foundation for future improvements through larger datasets, interpretability techniques, and ensemble methods.

These findings contribute meaningfully to digital forensics research, highlighting both the promise and current limitations of AI-powered solutions for combating visual misinformation. The project underscores the need for continued innovation as synthetic media grows more sophisticated, positioning this work as an important step toward reliable authenticity verification tools.

The balanced approach of leveraging proven methods (transfer learning) while identifying current limitations provides clear direction for future research in this socially vital field. As digital media manipulation techniques evolve, so too must our detection capabilities - making this an area ripe for further investigation and development.

Future Work

There is substantial scope for advancement in future iterations:

- Experiment with deeper or more efficient pre-trained models (e.g., ResNet, EfficientNet).
- Incorporate novel data augmentation techniques like GAN-based synthetic data generation.
- Employ ensemble models to leverage diverse predictions.
- Collect larger, labeled datasets to mitigate overfitting and improve generalization.
- Explore model interpretability tools to visualize decision boundaries and identify critical image features.