ForgeGuard

Image Forgery Detection Tool

Avni Jain Aakriti Sharma Jayeeta Shome

WE COHORT 6

22-07-2024

Introduction

- Image forgery is a significant issue with widespread implications.
- The rise of photo-editing software has made it easy to create tampered images.
- Importance of detecting such forgeries to maintain the integrity of digital content.
- Reference: Image forgery detection review

Project Objectives

- Develop a tool for detecting various types of image forgery.
- Use a combination of deep learning and traditional methods.
- Focus on techniques like splicing, copy-move, and more.

Tech Stack

- **Programming Language:** Python
- **Libraries/Frameworks:**
  - OpenCV
  - NumPy
  - Scikit-learn
  - Matplotlib
  - Pillow
  - TensorFlow - Keras
  - HTML, CSS, JavaScript
  - Flask

System Architecture

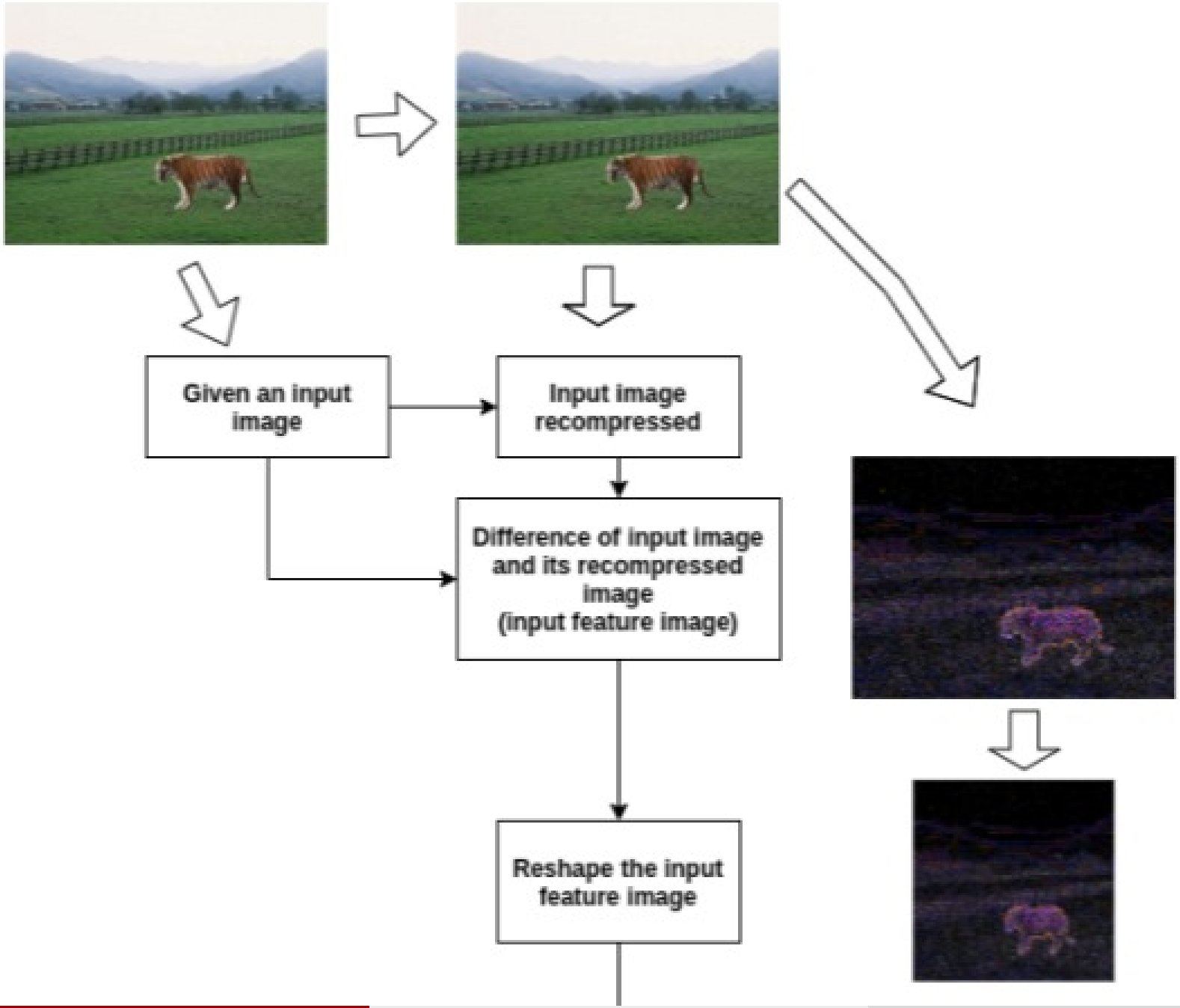
- Data Collection and Preprocessing
- Model Training
- Web Application Interface
- Deployment and Testing

Data Collection and Preprocessing

- Dataset: CASIA 2.0 Image Tampering Detection Dataset
- Data Splitting: Authentic Images, Tampered Images, Train/Test Split: 80/20
- Tools used: PIL, NumPy

Table 1. Details of the CASIA.2.0 image forgery database.

	Genuine Images	Tampered Images	Total Images
CASIA.2.0	7491	5123	12,614
Training (80%)	5993	4098	10,091
Testing (20%)	1498	1025	2523


Given an input image
Input image recompressed
Difference of input image and its recompressed image (input feature image)
Reshape the input feature image

Model Training

/ Prediction Model Description */*
Image_Forgery_Predictor_Model(image with size $128 \times 128 \times 3$)
{
    First convo. layer: 32 filters (size 3×3, strid size one, activation: "relu")
    Second convo. layer: 32 filters (size 3×3, strid size one, activation: "relu")
    Third convo. layer: 32 filters (size 3×3, strid size one, activation: "relu")
    Max-pooling of size 2×2
    Dense layer of 256 neurons with "relu" activation function
    Two neurons (output neurons) with "sigmoid" activation
}

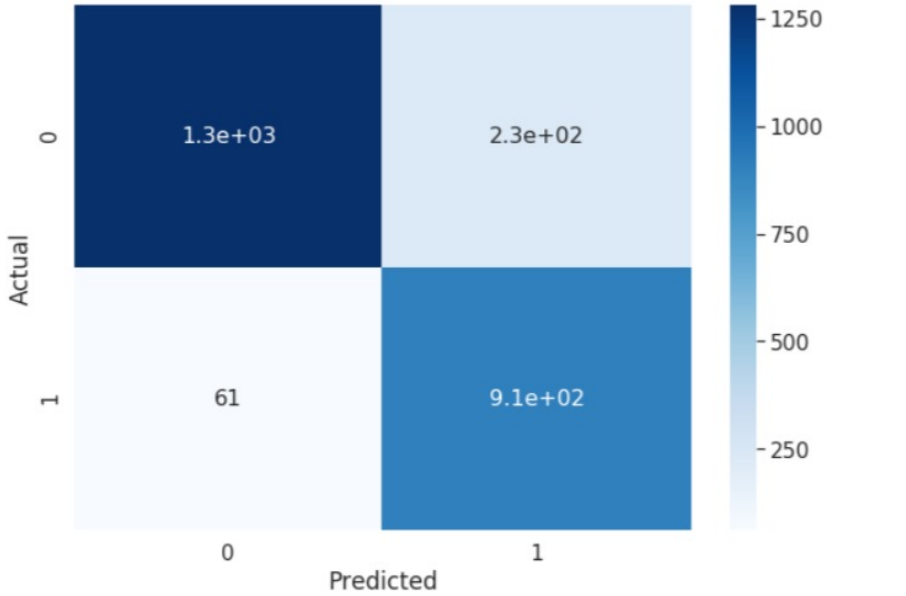
for *epochs* = 1 to *total_epochs* **do**
    $training\_error = 0$
    for $i = 1$ to n **do**
        $A_{recompressed\_i} = JPEG_{Compression}(A_i, Q)$
        $A_{diff\_i} = A_i - A_{recompressed\_i}$
        $A_{reshaped\_diff\_i} = reshape(A_{diff\_i}, (128, 128, 3))$
        $training\_error = (L_i - \text{Image\_Forgery\_Predictor\_Model}(A_{reshaped\_diff\_i})) + training\_error$
    end for
    modify_model(*training_error*, Image_Forgery_Predictor_Model(), Adam_optimizer)
end for

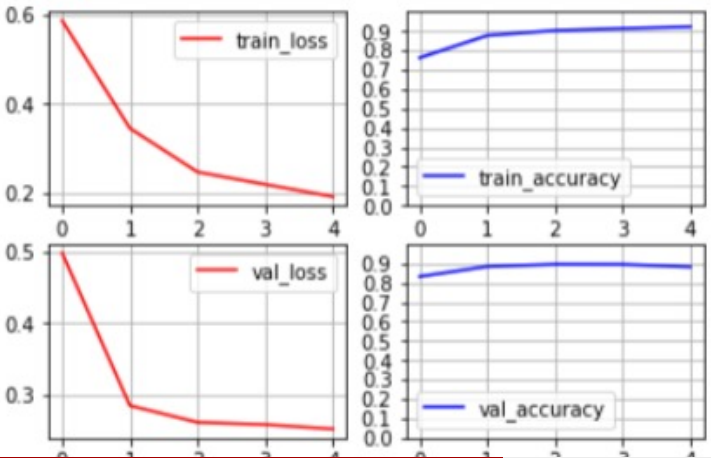
Web Application Interface

- Uploading images
- Displaying results
- Frontend technologies: HTML, CSS
- Backend technologies: Flask

Results and Testing

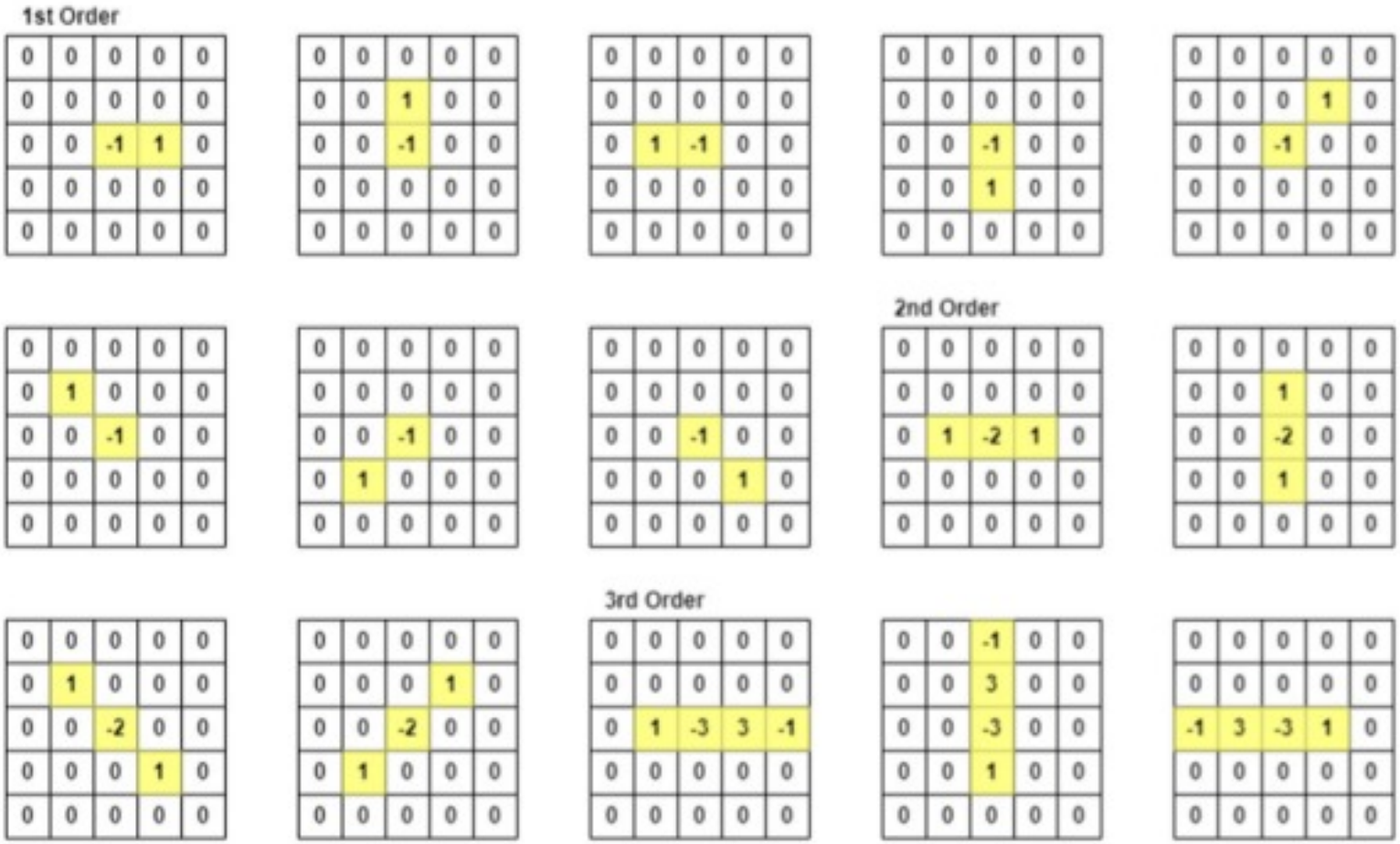
- Accuracy of the model: 90.125





Extra Features

- Noise Analysis
- ELA Analysis
- Highlighting Forged Area



Demo

Challenges Faced

- Implementing detection of forged parts
- Improving accuracy
- Limited memory space, due to which we were unable to train the model
- 100MB git limit, which resulted in errors while pushing the changes

Learnings

- Deep learning concepts
- Learning through research papers
- Learning through experimentation

Future Work

- Explore advanced deep learning techniques to improve accuracy
- Expand compatibility to handle different types of documents
- Integrate with cloud services for scalability and real-time processing

Questions

- Any questions?
- Suggestions and feedback are appreciated