DEEP LEARNING BASED IMAGE FORGERY DETECTION

Problem Statement:

Image tampering detection addresses the critical challenge of identifying whether digital images have been manipulated or altered, which is essential for maintaining the authenticity and trustworthiness of visual content in fields like journalism, law enforcement, and social media. With the increasing ease of editing images using sophisticated tools, tampering can introduce subtle distortions that are often difficult to detect by the human eye. Automated detection systems aim to uncover these manipulations by analyzing inconsistencies in compression artifacts, textures, or other image properties, often leveraging machine learning techniques to accurately classify images as authentic or tampered. This helps prevent misinformation, fraud, and protects the integrity f digital media in today's highly visual information environment.

Data Preparation

The process begins by importing all necessary libraries for image processing, data manipulation, and building deep learning models, including OpenCV, Pandas, NumPy, and TensorFlow. Images are loaded from two folders representing the classes “Authenticated” (genuine) and “Tampered” (fake). Images are taken from CASIA v2.0 dataset. Each image goes through Error Level Analysis (ELA), a technique for detecting tampering by exposing artificial compression artifacts. The ELA procedure enhances image differences when repeatedly compressed, allowing subtle manipulations to be highlighted. After conversion, images are resized to 128×128 pixels and normalized by dividing pixel values by 255, ensuring all input features are scaled between 0 and 1.

The images are labeled (1 for Authentic, 0 for Tampered), and the dataset is constructed by appending processed images and labels into arrays. The dataset is shuffled multiple times to guarantee that the model is not affected by order bias. The labels are converted to categorical format, and the data is reshaped to match the input requirements for convolutional neural networks (CNNs). The full dataset is split into training and validation sets using an 80-20 ratio, optimizing the model's ability to generalize and preventing overfitting.

Model Building

A deep CNN architecture is designed for image classification. The model consists of several convolutional layers with increasing filter sizes, each followed by activation functions (ReLU) and pooling layers to reduce spatial dimensions. These layers help the network learn hierarchical feature representations relevant to distinguishing authentic and tampered images. After multiple convolution and pooling stages, a flattening layer converts the feature maps to a one-dimensional vector, and dense (fully connected) layers further process these features. The last dense layer uses a softmax activation to output probabilities for the two classes.

The model is compiled with the Adam optimizer, a low learning rate, and binary cross-entropy loss function, which is suited for the classification task. Accuracy is used as an evaluation metric. Training is performed over 10 epochs with a batch size of 32, and validation is monitored at each epoch. Model weights are printed for inspection after training.

Model Training and Deployment

During training, the model learns to differentiate between authentic and tampered images by minimizing the loss function on the training data and evaluating performance on the validation set. Training feedback such as accuracy and loss curves serves as an indicator of learning effectiveness

and the risk of overfitting. After training, the model is saved to disk for future use, allowing quick deployment or further evaluation on additional data without retraining.

This pipeline offers an automated and effective approach for tampering detection in images by combining advanced preprocessing (ELA), robust data handling, and a deep learning classifier. The steps ensure images are systematically prepared, the model is optimally trained, and results can be easily reproduced or extended to other datasets.

Test accuracy is upto 0.89 and also model was deplpoyed locally using the flask framework