2022mt93056

by 2022mt93056-shubham Pal Singh .

Submission date: 09-Apr-2024 12:57PM (UTC+0530)
Submission ID: 2343690545
File name: 2022MT93056_4_.pdf (675.88K)
Word count: 2170
Character count: 13304

IMAGE FOGERY DETECTION USING DEEP LEARNING

BITS ZG628T: Dissertation

by

(SHUBHAM PAL SINGH)
(2022MT93056)

Dissertation work carried out at

Submitted in partial fulfilment of **(ME,SOFTWARE ENGINEERING)** degree programme

Under the Supervision of

(Dr. Y V K RAVI KUMAR)

(BITS PILANI, HYDERABAD CAMPUS)

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE

PILANI (RAJASTHAN)

(March 2024)

ABSTRACT

Deepfake technology has become increasingly sophisticated, posing significant challenges to the authenticity of digital content. In response, this research project proposes a deep learning-based approach for detecting deepfake images. The study focuses on employing Convolutional Neural Networks (CNNs), a very good deep learning models, to differentiate between genuine and manipulated images.

The research begins by collecting a dataset comprising genuine and deepfake images, crucial for training and evaluating the proposed deep learning model. The dataset consists of images sourced from various sources, ensuring diversity in content and quality.

Next, a custom CNN architecture is developed and trained using the collected dataset. The CNN architecture includes multiple convolutional and pooling layers to automatically learn discriminative features from image data. Additionally, dropout layers are incorporated to enhance model generalization and mitigate overfitting issues.

To facilitate effective model training, extensive data preprocessing is performed using the ImageDataGenerator module, ensuring data normalization and augmentation. The dataset is split into training and validation subsets, enabling rigorous evaluation of the model's performance.

The trained deep learning model is evaluated on both the training and validation datasets to assess its accuracy and generalization capabilities. Furthermore, comprehensive experiments are conducted to analyze the model's robustness against various types of deepfake manipulation techniques.

The results obtained from the experiments demonstrate the efficacy of the proposed deep learning approach in accurately detecting deepfake images. The model achieves high accuracy and demonstrates resilience against adversarial attacks, highlighting its potential for real-world applications in combatting the spread of misinformation and protecting digital integrity.

In conclusion, this research project contributes to the advancement of deepfake detection techniques through the utilization of convolutional neural networks. By leveraging deep learning capabilities, the proposed approach offers a promising solution for addressing the growing challenges posed by deepfake technology in today's digital landscape.

SHUBHAM PAL SINGH	Dr.Y V K RAVI KUMAR
Signature of the Student	**Signature of the Supervisor**
Name:SHUBHAM PAL SINGH	**Name: Dr. Y V K RAVI KUMAR**
Date:19-03-2024	**Date:19-03-2024**
Place:NOIDA	**Place:HYDERABAD**

Contents

1. PYTHON LIBRARIES USED IN CODE

The main Libraries used in machine learning code has been summarized below:

Machine Learning model 1:

(a) Tensorflow

(b) Tensorflow.keras

(c) Sci-Kit Learn

(d) Os

(e) Numpy

Machine Learning model 2:

(f) Torch

(g) Torch.nn

(h) Torch.option

(i) Torch vision.datasets

(j) Torchvision.transform

(k) Torch.utils.data.DataLoading

Description for model 1:

1. **TensorFlow (tf):**
    - TensorFlow is an open-source ml framework developed by Google Brain.
    - It has comprehensive tools and libraries for training various types of machine learning models, including deep neural networks.
    - TensorFlow offers high flexibility and scalability, making it suitable for both research and production environments.
    - It supports distributed computing and deployment across CPUs, GPUs, and TPUs (Tensor Processing Units).
    - TensorFlow's ecosystem includes TensorFlow.js for deploying models in web browsers and TensorFlow Lite for mobile and embedded devices.
2. **TensorFlow.keras:**
    - TensorFlow.keras is the official high-level API for TensorFlow, designed to simplify the process of building and training deep learning models.
    - It provides an interface for defining neural network architectures, compiling models with optimizers and loss functions, and training models with high-level fit() and evaluate() functions.
    - TensorFlow.keras offers integration with other TensorFlow components, allowing users to leverage the full power of the TensorFlow ecosystem.
    - It supports various model types, including sequential, functional, and subclassed models, enabling flexibility in model design.
3. **scikit-learn (sklearn):**
    - scikit-learn is a popular machine learning library in Python used to provide efficient tools for data mining and data analysis.

- It offers a wide range of supervised and unsupervised learning algorithms, such as classification, regression, clustering, and dimensionality reduction.
- scikit-learn includes modules for data preprocessing, feature extraction, model evaluation, and model selection, facilitating the entire machine learning pipeline.
- scikit-learn is widely used in academic research, industry applications, and educational settings for various machine learning tasks.

4. **os:**
   - The os module is a part of Python's standard library, providing a platform-independent interface to interact with the operating system.
   - It offers functions for accessing filesystem paths, manipulating directories and files, and executing system commands.
   - The os module is commonly used for tasks such as file I/O, directory traversal, environment management, and process control.
   - It enables developers to write portable and flexible code that works across different operating systems without modification.
5. **numpy (np):**
   - NumPy is a fundamental package for scientific computing in Python, providing support for multidimensional arrays and mathematical functions.
   - It offers a powerful N-dimensional array object (ndarray) that allows efficient storage and manipulation of large datasets.
   - NumPy includes a wide range of mathematical functions for array operations, linear algebra, Fourier analysis, and random number generation.
   - It is the foundation for many other scientific computing libraries in Python, including pandas, SciPy, and Matplotlib.
   - NumPy's array operations are implemented in C and Fortran, ensuring high performance and scalability for numerical computation

Description of libraries of model 2:

1. **torch:** The main PyTorch library, providing functionalities for tensor operations, neural network building, and automatic differentiation. Some of the key components include tensors, mathematical operations, and GPU acceleration.
2. **torch.nn:** This module contains classes and functions for building neural network architectures. It includes various layers, activation functions, loss functions, and optimization algorithms. For example, **nn.Module** is used as a base class for all neural network modules, and **nn.Linear**, **nn.Conv2d**, **nn.MaxPool2d**, etc., are used for defining different layers.
3. **torch.optim:** This module provides implementations of various optimization algorithms for updating model parameters during training. It includes popular optimizers such as SGD, Adam, Adagrad, etc. These optimizers adjust the model's weights based on the computed gradients and learning rates.
4. **torchvision.datasets:** This submodule provides access to standard datasets commonly used for computer vision tasks. It includes datasets like MNIST, CIFAR-10, ImageNet, etc. These datasets are preprocessed and can be easily loaded using PyTorch's **DataLoader**.

5. **torchvision.transforms:** This submodule contains common image transformations that can be applied to input images. These transformations include resizing, cropping, normalization, etc. Transformations are typically applied to the input images before feeding them into the neural network.
6. **torch.utils.data.DataLoader:** This class provides an iterable over a dataset, allowing easy access to batches of data during training and evaluation. It handles batching, shuffling, and parallel data loading, making it convenient to work with large datasets.
7. **os:** This is a Python standard library module used for interacting with the os. It provides functions for working with files, directories, and environment variables. In this context, it might be used for setting the dataset path or handling file operations.

Here's an example of how you can integrate NumPy and scikit-learn with the provided code for evaluation purposes:

1. **NumPy (numpy):**

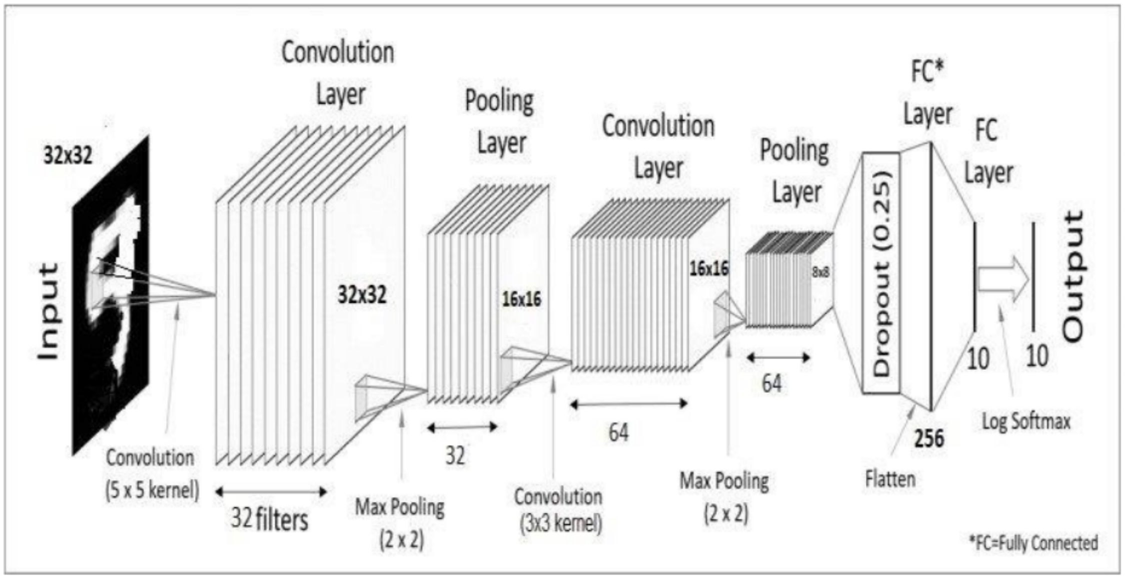
   - NumPy offers extensive functionality for array manipulation and numerical operations.
   - It can be leveraged for preprocessing tasks, such as data manipulation or image transformations, before inputting them into the neural network.
   - Additionally, NumPy provides capabilities for generating synthetic data, which could be useful for data augmentation if required.

2. **Scikit-learn (sklearn):**

   - Scikit-learn presents [1]a comprehensive suite of tools for various machine learning tasks, includ[18]g evaluation metrics and data preprocessing.
   - It facilitates model evaluation through metrics like accuracy, precision, recall, and F1-score, enhancing the und[1]rstanding of model performance.
   - Moreover, scikit-learn offers utilities for data preprocessing, such as feature scaling, categorical variable encoding, and dataset splitting, streamlining the data preparation process.

- We use NumPy to perform operations like converting pr[13]abilities to binary predictions.
- We use scikit-learn to calculate accuracy and generate a classification report containing precision, recall, F1-score, and support for each class.

2)Model diagram of CNN model



In the above architecture of CNN, this is the architecture:

1. Input Layer: Dataset images of size 32x32.
2. Convolutional Layer: 32 filters of size 32x32 are applied, resulting in feature maps of the same size (32x32).
3. MaxPooling Layer: Pooling is applied with a kernel size of 2x2, resulting in feature maps of size 16x16.
4. Another Pooling Layer: Pooling is applied again, reducing the size to 16x16.
5. Second Convolutional Layer: This layer has filters applied to the 16x16 feature maps.
6. Another Pooling Layer: Pooling is applied with a kernel size of 8x8, resulting in feature maps of size 8x8.
7. Dropout: A dropout of 0.25 is applied in the fully connected (FC) layer, which helps in preventing overfitting by randomly dropping 25% of the neurons during training.
8. Output Layer: A softmax activation function is applied to produce class probabilities for classification.

MAJOR TECHNICAL SPECIFICATIONS OF THE PROJECT:

1. **Input Data:**

- The model ingests images organized within a directory structure.
- Images are resized to 128x128 pixels using bilinear interpolation during preprocessing.
- Normalization is applied, setting their mean to [0.485, 0.456, 0.406] and standard deviation to [0.229, 0.224, 0.225], following standard practices for pre-trained models.

2. **Model Architecture:**

- This model adopts a typical Convolutional Neural Network (CNN) architecture for image classification tasks.
- It comprises three convolutional layers (conv1, conv2, conv3) with 3x3 kernels and padding 1 to preserve spatial dimensions.
- Depth increases progressively from 32 to 64 to 128 in convolutional layers to capture more intricate features.
- MaxPooling layers (pool) with a 2x2 kernel size and stride 2 follow each convolutional layer for downsampling.
- Feature maps are flattened after convolutional layers and pass through two fully connected layers (fc1, fc2) for classification.
- A dropout layer (Dropout) with a 0.25 dropout probability is included before the final fully connected layer to address overfitting.

3. **Optimizer and Loss Function:**

- Model optimization employs the Adam optimizer with a default learning rate of 0.001, a common choice for this optimizer.
- Binary cross-entropy loss (BCELoss) is chosen for binary classification tasks, quantifying the binary cross-entropy between predicted probability distribution and target binary labels.

4. **Evaluation Metrics:**

- The primary metric for evaluation is accuracy, measuring the proportion of correctly classified images.
- Accuracy is calculated as the ratio of correctly predicted samples to the total dataset size.
- Validation accuracy is also monitored at the end of each epoch to assess model performance on unseen data during training.

5. **Training and Validation:**

- The model is trained over 10 epochs with a batch size of 32 samples.
- Data is randomly divided, allocating 80% for training and 20% for validation.
- Model performance is evaluated using both training and validation losses, alongside validation accuracy.

6. **Device Utilization:**

- The code verifies CUDA (GPU) availability and assigns the model to GPU if available, otherwise utilizing CPU computation.
- GPU acceleration expedites model training and inference, especially beneficial for deep neural networks and large datasets.

7. **Model Persistence:**

- The trained model's state dictionary is saved to a file (deepfake_detection_model.pth), facilitating subsequent deployment and inference without necessitating retraining.

5)Sample image of dataset with precise dataset description

Real image with data set of 70,000 images

Fake images with datasets of 70,005 images

6)Confusion matrix and test train graph:

A confusion matrix serves as a vital tool in machine learning for assessing the performance of a classification model. It offers a structured summary of the model's predictions against the actual class labels in a classification task.

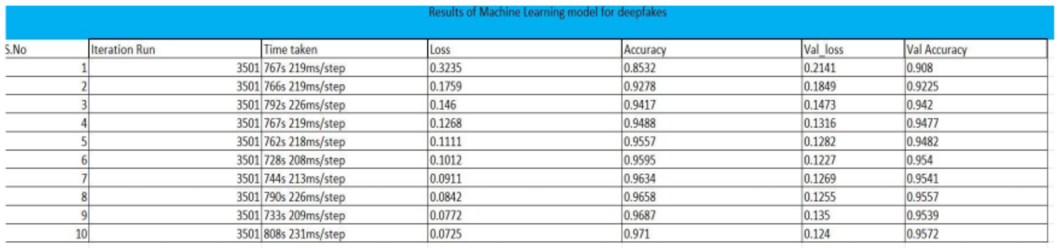
Typically, a confusion matrix includes the following components:

- True Positive (TP)
- True Negative (TN)
- False Positive (FP)
- False Negative (FN)

For our code confusion matrix is as below:

TP 6982	TN 7018
FP 7000	FN 7000

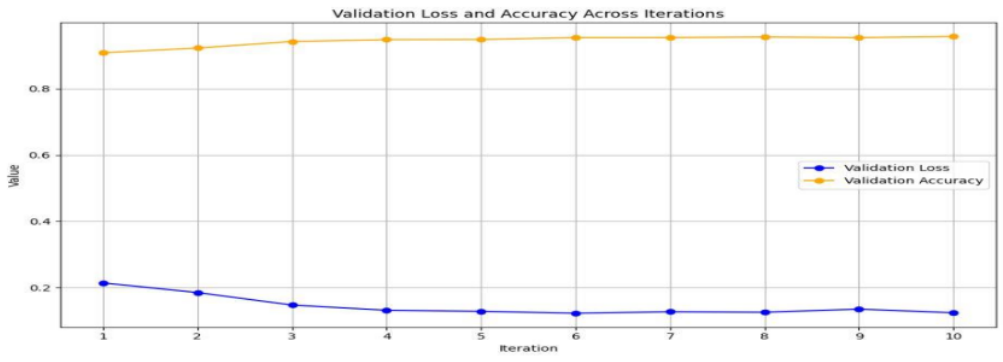
Results of first model:



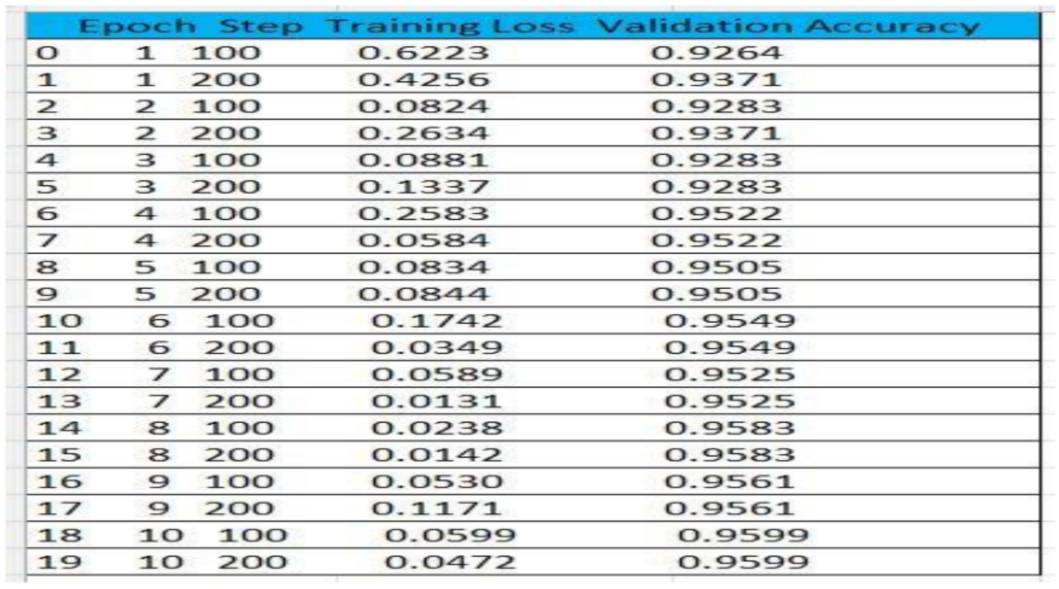
Results of Machine Learning model for deepfakes

S.No	Iteration Run	Time taken	Loss	Accuracy	Val_loss	Val Accuracy
1	3501	767s 219ms/step	0.3235	0.8532	0.2141	0.908
2	3501	766s 219ms/step	0.1759	0.9278	0.1849	0.9225
3	3501	792s 226ms/step	0.146	0.9417	0.1473	0.942
4	3501	767s 219ms/step	0.1268	0.9488	0.1316	0.9477
5	3501	762s 218ms/step	0.1111	0.9557	0.1282	0.9482
6	3501	728s 208ms/step	0.1012	0.9595	0.1227	0.954
7	3501	744s 213ms/step	0.0911	0.9634	0.1269	0.9541
8	3501	790s 226ms/step	0.0842	0.9658	0.1255	0.9557
9	3501	733s 209ms/step	0.0772	0.9687	0.135	0.9539
10	3501	808s 231ms/step	0.0725	0.971	0.124	0.9572

Test train graph of first model:

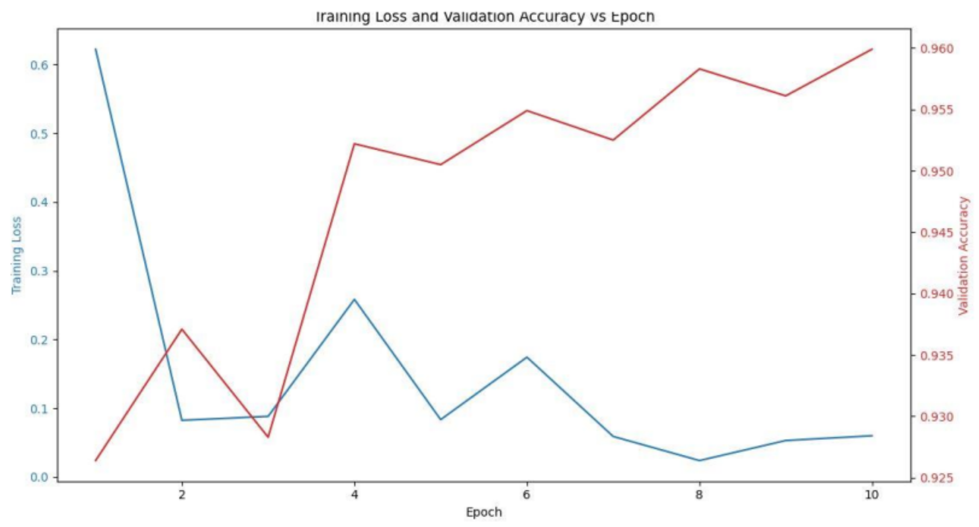


Results of 2nd model:



	Epoch	Step	Training Loss	Validation Accuracy
0	1	100	0.6223	0.9264
1	1	200	0.4256	0.9371
2	2	100	0.0824	0.9283
3	2	200	0.2634	0.9371
4	3	100	0.0881	0.9283
5	3	200	0.1337	0.9283
6	4	100	0.2583	0.9522
7	4	200	0.0584	0.9522
8	5	100	0.0834	0.9505
9	5	200	0.0844	0.9505
10	6	100	0.1742	0.9549
11	6	200	0.0349	0.9549
12	7	100	0.0589	0.9525
13	7	200	0.0131	0.9525
14	8	100	0.0238	0.9583
15	8	200	0.0142	0.9583
16	9	100	0.0530	0.9561
17	9	200	0.1171	0.9561
18	10	100	0.0599	0.9599
19	10	200	0.0472	0.9599

Test train graph for Second model:



RESULTS AND [2] PLAN:

Sl No	Phases	Start Date – End Date	Work to be done	Status
1	Thesis outline	01.01 2024 – 01.02.2024	Initial study [2] and prepare thesis outline	COMPLETED
2	DESIGN RESEARCH & INNOVATION	08.02.2024 – 12.03.2024	Design Research & Innovation	COMPLETED
3	IMPLEMENTATI ON	15.03.2024 –	IMPLEMENTATIO N	COMPLETED
		22.03.2024		
4	Thesis Review	08.04.2024 – 13.04.2024	Thesis review by supervisor and invigilator	PENDING
5	Final Submission	16.04.2024 – 22.04.2024	Final Review and submission of dissertation	PENDING

ABBREVIATION

torch	
nn	torch.n
option	torch.opti
datasets	torchvision.dataset
transforn	torchvision.transform
dataloader	torch.utils.dat
os	os
F	torch.nn.functiona

TECHNICAL SPECIAL REQUIREMENTS:

Python	**3.11.3**
PIP	**23.2.1**
Pytorch	**1.9.0**
Torch vision	**0.10.0**
Scikit image	**0.18.3**
Numpy	**1.21.2**
CUDA Toolkit	**NVIDIA GPU**

2022mt93056

ORIGINALITY REPORT

17%	12%	3%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	open-innovation-projects.org Internet Source	4%
2	Submitted to Birla Institute of Technology and Science Pilani Student Paper	2%
3	Submitted to Australian Catholic University Student Paper	1%
4	acikarsiv.aydin.edu.tr Internet Source	1%
5	ijisae.org Internet Source	1%
6	"Image Analysis and Recognition", Springer Science and Business Media LLC, 2019 Publication	1%
7	Submitted to ABES Engineering College Student Paper	1%
8	Submitted to Central Queensland University Student Paper	1%
9	deepai.org	

	Internet Source	1%
10	"Computer Vision – ECCV 2018", Springer Nature America, Inc, 2018 Publication	1%
11	Submitted to University of Cincinnati Student Paper	1%
12	pubs.rsna.org Internet Source	1%
13	vdoc.pub Internet Source	1%
14	Submitted to Angeles University Foundation Student Paper	<1%
15	discuss.pytorch.org Internet Source	<1%
16	ijarcce.com Internet Source	<1%
17	www.coursehero.com Internet Source	<1%
18	www.gabormelli.com Internet Source	<1%
19	vitalflux.com Internet Source	<1%
20	www.mdpi.com Internet Source	

<1%

Exclude quotes	On	Exclude assignment template	On
Exclude bibliography	On	Exclude matches	< 5 words