

**IMAGE FORGERY DETECTION USING
FEATURE AND DEEP LEARNING BASED
APPROACHES**

BY

ATIF SHAH

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

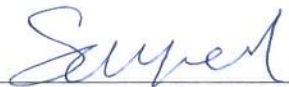
FEBRUARY 2019

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **ATIF SHAH** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MAS-TER OF SCIENCE IN COMPUTER SCIENCE**.

Thesis Committee

 18/2/2019

Prof. El-Sayed M. El-Alfy (Adviser)



Dr. Wasfi G. Al-Khatib (Member)



Dr. Salahdin A. Mohammed (Member)



Dr. Hamoud Al-Jamaan
Department Chairman



Prof. Salam A. Zummo
Dean of Graduate Studies

5/3/19

Date



©ATIF SHAH
2019

Dedicated to my family

ACKNOWLEDGMENTS

First of all, I am grateful to Almighty Allah, the most Compassionate and most Merciful, for the strength and blessing in completing this thesis. I feel profound to extend my gratitude to my supervisor Prof. El-Sayed M. El-Alfy for his inspiring guidance through out my thesis. He always pushes me time to time to reach the high standard. I am also thankful to the committee members, Dr. Wasfi G. Al-Khatib and Dr. Salahdin A. Mohammed, for their support and motivation during my thesis. I would also like to pay my kind regards to Dr. Moataz Ahmed for his guidance and unlimited support. He used to welcome and help me during my stay at KFUPM. I am deeply thankful to KFUPM for giving me the opportunity to study my Masters Degree at such a prestigious institute. I am thankful to my parents for their unmatched love, care and support. I am also thankful to my friends, colleagues and Pakistan community for making this experience memorable for me.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	vi
LIST OF TABLES	x
LIST OF FIGURES	xiii
LIST OF ABBREVIATIONS	xvii
ABSTRACT (ENGLISH)	xx
ABSTRACT (ARABIC)	xxii
CHAPTER 1 INTRODUCTION	1
1.1 Introduction	1
1.2 Research Statement and Motivation	3
1.3 Thesis Research Objectives	4
1.4 Thesis Organization	5
CHAPTER 2 BACKGROUND AND RELATED WORK	6
2.1 Passive Detection of Image Forgery	6
2.1.1 Detecting Intrinsic Regularities and Inconsistencies	7
2.1.2 Forgery Operation Related Approaches	8
2.2 Machine-Learning Paradigms	10
2.2.1 Supervised Machine Learning	10
2.2.2 Unsupervised Machine Learning	13
2.2.3 CNN-Based Deep Learning Models	14

2.3	Related Work	21
2.3.1	Spatial Features Based Approaches	21
2.3.2	Spatial Features with Transform Based Approaches	26
2.3.3	Deep Learning Based Approaches	31
CHAPTER 3 METHODOLOGY		37
3.1	Feature Extraction Based Methods	37
3.1.1	Pre-Processing	38
3.1.2	Texture-Based Feature Extractors	38
3.1.3	Using DCT on Transformed Images	47
3.1.4	Multi-Scale Feature Extraction	49
3.1.5	Feature Reduction	50
3.1.6	Classification	51
3.2	Deep Learning Approach	52
3.2.1	Components of CNN	55
3.2.2	Weights Initialization	58
3.2.3	Data Preparation for CNN	59
CHAPTER 4 RESULTS AND DISCUSSIONS		65
4.1	Benchmark Datasets	65
4.2	Experimental Settings	67
4.3	Evaluation Metrics	68
4.4	Feature-Extraction Based Experiments and Results	69
4.5	CNN-Based Experiments and Results	71
4.6	Statistical Analysis and Feature Reduction	75
4.7	ROC-Based Comparisons	85
CHAPTER 5 CONCLUSIONS AND FUTURE WORK		89
APPENDIX A DETAILED EXPERIMENTS AND RESULTS		92
A.1	Feature Based Experiments	92
A.2	Deep Learning Based Experiments	124

APPENDIX B PUBLICATIONS	139
B.1 Published	139
B.2 Under Preparation	139
REFERENCES	140
VITAE	158

LIST OF TABLES

2.1	Literature review summary part I	35
2.2	Literature review summary part II	36
4.1	Datasets details	66
4.2	Summery of experiments using CASIA v1.0 dataset	71
4.3	Summery of experiments using CASIA v2.0 dataset	71
4.4	CNN architecture with all the layers	73
4.5	CNN configuration	74
4.6	CASIA v1.0 statistical analysis	78
4.7	CASIA v2.0 statistical analysis	78
4.8	Combining all Y, Cb and Cr channels for the top five methods and with feature reduction using PCA and LLP for CASIA v1.0	81
4.9	Combining all Y, Cb and Cr channels for the top five methods and with feature reduction using PCA and LLP for CASIA v2.0	82
4.10	Feature extraction time (seconds) for top 10 methods on CASIA v1.0 dataset	83
4.11	Feature extraction time (seconds) for top 10 methods on CASIA v2.0 dataset	84
4.12	Comparison of proposed method with previous methods in terms of Accuracy and AUC	88
A.1	Results for each channel on CASIA v1.0 dataset using CSLBP and CSLBP+DCT. (Feature concatenation)	93

A.2	Results for each channel on CASIA v1.0 dataset using SLBP and SLBP+DCT. (Feature concatenation)	94
A.3	Results for each channel on CASIA v1.0 dataset using WLD. (Feature concatenation)	95
A.4	Results for each channel on CASIA v1.0 dataset using LPQ and MSLPQ. (Feature concatenation)	96
A.5	Results for each channel on CASIA v1.0 dataset using MSLBP with radii of 1,2,3 and 4 (Feature concatenation)	97
A.6	Results for each channel on CASIA v1.0 dataset using LBP with radii of 1,2,3 and 4 (Feature concatenation)	98
A.7	Results for each channel on CASIA v1.0 dataset using CSLBP and CSLBP+DCT. (Feature Stacking)	99
A.8	Results for each channel on CASIA v1.0 dataset using MSLBP-DCT with radii of 1,2,3 and 4. (Feature Stacking)	101
A.9	Results for each channel on CASIA v1.0 dataset using LBP-DCT with radii of 1,2,3 and 4 (Feature stacking)	102
A.10	Results for each channel on CASIA v1.0 dataset using MSLBP with radii 1,2,3 and 4. (Feature Stacking)	103
A.11	Results for each channel on CASIA v1.0 dataset using LBP with radii of 1,2,3 and 4 (Feature stacking)	104
A.12	Results for each channel on CASIA v1.0 dataset using LBP-DCT and MSLBP-DCT with neighbors of 8,12 and 16 (Feature stacking)	105
A.13	Results for each channel on CASIA v1.0 dataset using LBP and MSLBP with neighbors of 8 and 12 (Feature stacking)	106
A.14	Results for each channel on CASIA v1.0 dataset using LPQ-DCT and MSLPQ-DCT. (Feature stacking)	107
A.15	Results for each channel on CASIA v1.0 dataset using LPQ and MSLPQ. (Feature stacking)	108
A.16	Results for each channel on CASIA v1.0 dataset using SLBP and SLBP+DCT. (Feature stacking)	109

A.17 Results for each channel on CASIA v1.0 dataset using WLD and WLD-DCT. (Feature stacking)	110
A.18 Results for each channel on CASIA v2.0 dataset using CSLBP and CSLBP+DCT (Feature stacking)	111
A.19 Results for each channel on CASIA v2.0 dataset using MSLBP-DCT with radii of 1,2,3 and 4. (Feature stacking)	113
A.20 Results for each channel on CASIA v2.0 dataset using LBP-DCT with radii of 1,2,3 and 4. (Feature stacking)	114
A.21 Results for each channel on CASIA v2.0 dataset using MSLBP. (Feature stacking)	115
A.22 Results for each channel on CASIA v2.0 dataset using LBP using radii 1,2,3 and 4. (Feature stacking)	116
A.23 Results for each channel on CASIA v2.0 dataset using LBP-DCT and MSLBP-DCT with neighbors 8,12 and 16. (Features stacking)	117
A.24 Results for each channel on CASIA v2.0 dataset using LBP and MSLBP with neighbors 8, 12 and 16. (Features stacking)	118
A.25 Results for each channel on CASIA v2.0 dataset using SLBP and SLBP+DCT. (Features stacking)	119
A.26 Results for each channel on CASIA v2.0 dataset using WLD and WLD-DCT. (Feature stacking)	120
A.27 Results for each channel on CASIA v2.0 dataset using LPQ-DCT and MSLPQ-DCT. (Features stacking)	121
A.28 Results for each channel on CASIA v2.0 dataset using LPQ and MSLPQ (Feature Stacking)	123

LIST OF FIGURES

1.1	Forged images: (a) John Kerry forged image (splicing), and (b) Iranian test missile forged image (copy-move)	2
2.1	Taxonomy of image forgery detection techniques	7
2.2	Image forgery detection framework	9
2.3	Single neuron model	12
2.4	Multi-layer feed-forward ANN architecture	13
2.5	SVM optimal hyperplane	14
2.6	An example of K -means clustering where $K = 3$	15
2.7	An example of hierarchical clustering	16
2.8	LeNet architecture	17
2.9	AlexNet architecture	17
2.10	ZFNET architecture	18
2.11	GoogLeNet architecture	19
2.12	VVGNet architecture	20
2.13	ResNet architecture	21
3.1	A typical layout for image forgery detection	38
3.2	Flow diagram of feature extraction using histogram concatenation . .	39
3.3	Flow diagram of feature extraction using histogram stacking	40
3.4	LBP example with 3×3 block	40
3.5	LBP and CSLBP with $R=1$ and $P=8$ [1]	41
3.6	WLD filters	45
3.7	Flow diagram of feature extraction using DCT block stacking	48

3.8	A window of 9×9 shows the radii $R \in \{1, 2, 3, 4\}$ and neighbors $P = 8$	49
3.9	LBP circular symmetric neighborhood with $(R=4, P=8)$, $(R=2, P=12)$ and $(R=4, P=16)$ [2]	50
3.10	Convolutional neural network	53
3.11	Proposed convolutional neural network model	54
3.12	Maximum pooling	57
3.13	SRM model weights	59
3.14	Authentic and forged images from CAISA v1.0 dataset	60
3.15	Extraction of forged region from image	61
3.16	Patch extraction visualization	62
3.17	Patch extractions from authentic and forged image	63
3.18	The first rows shows a sample of flipped images and the second row shows a sample of mirror images	64
4.1	First row shows three authentic and second row shows three forged images of CASIA v1.0	66
4.2	First row shows three authentic and second row shows three forged images of CASIA v2.0	66
4.3	Confidence intervals for CASIA v1.0 dataset	76
4.4	Confidence intervals for CASIA v2.0 dataset	77
4.5	PCA features reduction and selection for CASIA v1.0	80
4.6	PCA features reduction and selection for CASIA v2.0	80
4.7	ROC curves of top five method using CASIA v1.0	86
4.8	ROC curves of top five method using CASIA v2.0	86
A.1	The results of four convolutional layer model using 99376 patches	125
A.2	The results of four convolutional layer model using 2000 patches	126
A.3	The results of four convolutional layer model using 12422 patches	127
A.4	The results of five convolutional layer model using 2000 patches with 30 epochs	128

A.5	The results of five convolutional layer model using 2000 patches with 50 epochs	128
A.6	The results of five convolutional layer model using 2000 patches with 80 epochs	129
A.7	The results of five convolutional layer model using 2000 patches with 150 epochs	129
A.8	The results of five convolutional layer model using 12422 patches with 100 epochs	130
A.9	The results of five convolutional layer model using 12422 patches with 150 epochs	131
A.10	The results of five convolutional layer model using 12422 patches with 200 epochs	131
A.11	The results of five convolutional layer model using 12422 patches with 250 epochs	132
A.12	The results of eight convolutional layer model using 2000 patches with SRM weights initialization	133
A.13	The results of eight convolutional layer model using 12422 patches with SRM weights initialization	133
A.14	The results of eight convolutional layer model using 99376 patches with random weights initialization	134
A.15	The results of eight convolutional layer model using 2000 patches with regularization and SRM weights initialization	135
A.16	The results of eight convolutional layer model using 12422 patches with regularization and SRM weights initialization	135
A.17	The results of eight convolutional layer model using 99376 patches with regularization and SRM weights initialization	136
A.18	The results of ten convolutional layer model using 2000 patches with regularization and SRM weights initialization	137
A.19	The results of ten convolutional layer model using 12422 patches with regularization and SRM weights initialization	137

A.20 The results of ten convolutional layer model using 99376 patches with
regularization and SRM weights initialization 138

LIST OF ABBREVIATIONS

2D-DCT	Two-Dimensional DCT
ANMS	Adaptive Non-Maximal Suppression
ANN	Artificial Neural Network
AUC	Area Under the Curve
CCD	Charged Coupled Device
CDR	Correct Detection Ratio
CFA	Color Filter Array
CMOS	Metal-Oxide Semiconductor
CNN	Convolutional Neural Network
CSLBP	Center Symmetric LBP
DCT	Discrete Cosine Transform
DoG	Difference of Gaussian
DWT	Discrete Wavelet Transform
FE	Feature Extraction
FN	False Negative
FNR	False Negative Rate

FP	False Positive
FPR	False Positive Rate
GWT	Gabor Wavelet Tranform
HHT	Hilbert-Huang Transform
LBP	Local Binary Pattern
LBPV	Local Binary Pattern Variance
LCA	Lateral Chromatic Aberration
LGPQ	Local Gabor Phase Quantization
LGWP	Local Gabor Wavelets Pattern
LLP	Local Learning Projection
LPQ	Local Phase Quantization
LSTM	Long-Short Term Memory
MBDCT	Multi-size Block Discrete Cosine Transform
MRF	Median-Filter Residual
MSLBP	Multi-Scale LBP
MSLPQ	Multi-Scale LPQ
MSWLD	Multi-Scale WLD
NMF	Non-Negative Matrix Factorization
NNR	Nearest Neighbor Ratio
PCA	Principal Component Analysis
Ph.D.	Doctor of Philosophy

PRNU	Photo-Response Non-Uniformity
PSF	Point Spread Function
QMF	Quadrature Mirror Filter
RANSAC	Random Sample Consensus
RBF	Radial-Basis Function
ResNet	Residual Network
RFE	Recursive Feature Elimination
RGB	Red, Green and Blue
RICLBP	Robust Invariance Co-occurrence Local Binary Pattern
RLRN	Run-Length Run-Number
SIFT	Scale Invariant Feature Transform
SLBP	Shift Local Binary Pattern
SPT	Steerable Pyramid Transform
SRM	Steganalysis Rich Model
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
SWT	Stationary Wavelet Transform
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
WLD	Weber Local Descriptor

THESIS ABSTRACT

NAME: ATIF SHAH

TITLE OF STUDY: IMAGE FORGERY DETECTION USING FEATURE
AND DEEP LEARNING BASED APPROACHES

MAJOR FIELD: COMPUTER SCIENCE

DATE OF DEGREE: FEBRUARY 2019

Image forensics is an active research area due to the large number of shared images online. These images can be easily manipulated with the availability of advanced image editing tools and the changes cannot be captured easily by bare human eyes. In this work, we evaluate and propose different models based on feature extraction and deep learning. In feature-based approaches, various features are extracted by applying various models based on LBP, MSLBP, CSLBP, SLBP, WLD, LPQ, MSLPQ and DCT. First, the image is divided into blocks and a feature vector is created to represent the image in a low dimensional space. SVM is used with different kernels to classify forged and authentic images. Using k -fold cross-validation, we computed several performance measures to evaluate and compare the effectiveness of the investigated models. Two publically available datasets CASIA v1.0 and CASIA v2.0 were used for evaluation.

While using a single color channel, we found MSLBP-DCT outperformed the other methods on CASIA v1.0 whereas MPLPQ was the best on CASIA v2.0, with about 97.3% accuracy. We conducted statistical analysis for the top ten models and the results confirmed that MSLBP-DCT and MSLPQ are the best for CASIA v1.0 and CASIA v2.0, respectively. We also compared the top five methods, based on statistical analysis, on combined features from the three color channels Y, Cb and Cr with and without feature reduction using PCA and LLP. MSLPQ-DCT achieved better accuracy of 98.56% on CASIA v1.0 and MSLPQ achieved a slightly higher accuracy of 97.4% compared to a single color channel. For the deep learning approach, patches are generated and classified as authentic or forged using a Convolutional Neural Network (CNN) model. A ten layers CNN model is trained and achieved 91% accuracy.

ملخص الرسالة

الاسم: عاطف شاه

عنوان الدراسة: كشف الصور المزيفة باستخدام طرق مبنية على استخلاص السمات والتعلم العميق

التخصص: ماجستير في علوم الحاسوب

تاريخ الدرجة العلمية: فبراير ٢٠١٩

نظراً لانتشار الصور الرقمية عبر الشبكة العنكبوتية وتوفر أدوات متقدمة لتعديلها فقد ازدادت خطورة تزيفها مع صعوبة كشف التعديل بالعين المجردة لهذا الكم الهائل من الصور. في هذا البحث تم اقتراح وتقييم عدة نماذج بناءً على طرق استخلاص السمات والتعلم العميق. ففي الأسلوب الأول تم استخلاص عدة سمات من خلال تحويل الصور إلى القنوات اللونية YCbCr وتقسيم الصورة إلى مربعات ثم حساب سمات لكل مربع منها ودمجها بطرق مختلفة ومن ثم بناء نماذج للتعرف على الصور المزيفة. وقد استخدمنا عدة مقاييس لتقييم الأداء ومقارنة النماذج المختلفة وذلك من خلال مجموعتين من الصور المتوفرة لتقييم الأداء CASIA v1.0 و CASIA v2.0. وقد أظهرت النتائج تفوق طريقة MSLBP-DCT مع مجموعة الصور الأولى CASIA v1.0 وتفوق طريقة MSLPQ مع مجموعة الصور الثانية CASIA v2.0، وقد حققنا معدل تعرف 97.33% تقريباً. وقد قمنا بإجراء تحليل إحصائي لأفضل عشر طرق وأكدت النتائج أن MSLBP-DCT و MSLPQ هما الأفضل لكشف التزييف في مجموعتي الصور CASIA v1.0 و CASIA v2.0 على التوالي عند استخدام قناة لونية واحدة. لكن عند دمج السمات المختلفة من الثلاث قنوات لونية ومقارنة أفضل خمس طرق إحصائياً في حالتي انتقاء وعدم انتقاء للسمات المميزة كانت النتائج أكثر فعالية حيث حققت طريقة MSLBP-DCT معدل تعرف في حدود 98.56% مع مجموعة الصور CASIA v1.0 وحققت طريقة MSLPQ أعلى قليلاً من 97.4% مع مجموعة الصور CASIA v2.0. بالنسبة للتعلم العميق تم استخراج رقع (patches) من الصور وتقسيمها إلى مجموعتين للتدريب والتقييم وتم تصميم شبكة عصبية تلافيفية (CNN) وكانت أفضل النتائج حوالي 91% لكن يمكن تحقيق أفضل من ذلك عند توفر مجموعة كبيرة من الصور والتدريب لفترات أطول.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Nowadays, we are living in an era where we can use different technologies to share, access and process different types of information. Visual content is among the most significant information that poses several challenges. The Chinese proverb described the power of the picture with these words “A picture is worth a thousand of words”. Another idiom is “seeing is believing” which emphasizes on visual clues more than verbal clues. With the growth of technologies, different techniques and algorithms have been developed to process images and photos. The main problem is that many of the images are uploaded to the world-wide-web and social networks. Some of them have been digitally manipulated or tampered in such a way that the original content has been distorted and the interpretation of the image is different. This process is called image forgery. Figure 1.1a shows an example of a forged image in 2004 where Senator John Kerry is speaking to the crowd with a Hollywood actress, Jane Fonda,

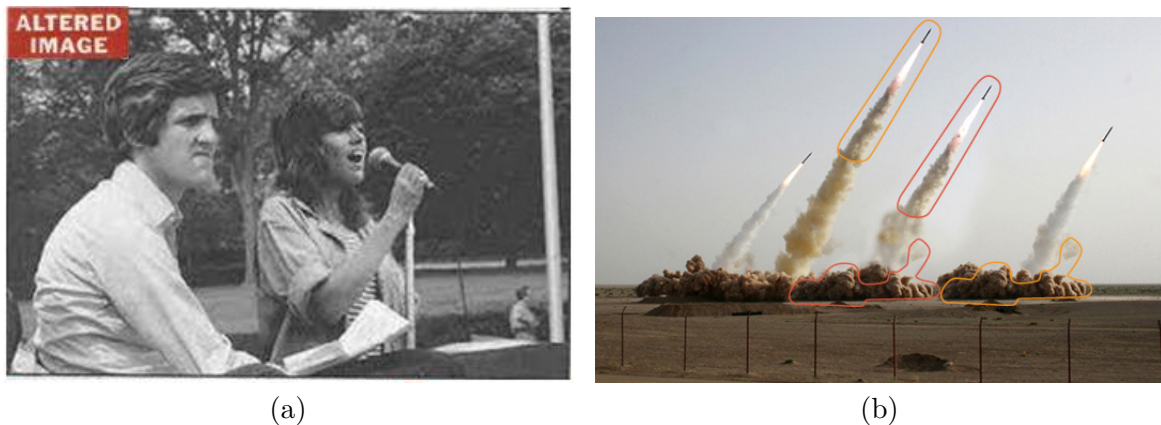


Figure 1.1: Forged images: (a) John Kerry forged image (splicing), and (b) Iranian test missile forged image (copy-move)

which has never happened; this type of forgery is known as image splicing forgery¹. It was manipulated and launched during the American presidential election campaign as a political strategy to raise questions on John's patriotism. Figure 1.1b shows another example of an image forgery of missile at launching time; this type of forgery is known as copy-move forgery².

Image manipulation aims at altering an image without leaving traceable footprints using various methods to achieve some desired results for ethical or unethical reasons. Manipulation techniques are not new, and are acceptable in some cases such as photomontage and visual effects in movies, medical imaging, and journalism. However, with the availability of powerful image-editing tools and the widespread of social media and news websites, they are increasingly used to tamper images and harm others, e.g. for political, financial or social reasons.

Image forensic is an area which is used to analyze and investigate the changes

¹Photo Tampering Throughout History: <http://pth.izitru.com>

²<http://thelede.blogs.nytimes.com/2008/07/10/in-an-iranian-image-a-missile-too-many/>

in images. Different techniques have been proposed for detecting tampered images. These techniques are categorized as active or passive/blind. Active approaches need some prior information such as embedded digital signatures or watermarks which are used for image-authenticity check. This limits their application because embedded information may affect the image quality and they require specially equipped cameras [3]. On the other hands, passive approaches do not need any additional information to detect manipulated images other than the image itself. In this research work, we are only concerned with passive approaches. More background details and related work are provided in Chapter 2.

1.2 Research Statement and Motivation

Detecting image forgery is a crucial research area nowadays due to a large number of image sharing websites and applications in various domains i.e. social media, medical information systems, satellite imaging, etc. Thousands of images are uploaded on a daily basis where it becomes harder to distinguish between the original and forged ones. The images can be manipulated by duplicating regions of the image, known as copy-move forgery, or combining two different images, called image splicing. These images are altered in such a way that a naked human eye cannot observe the artifacts.

There are lots of efforts in the literature to propose automated methods for passive/blind detection of image forgery (i.e. using only the image itself). From a machine learning perspective, this problem is a binary classification problem aiming to determine the right category for a given image, i.e. is it authentic or forged? A key element

in this process is feature extraction to represent each image by a feature vector in a low-dimensional space. Several methods have been proposed based on image transformation for texture representation. However, there is no comprehensive study to compare and benchmark these methods. Moreover, feature extraction is a computationally expensive step in image processing. New advances in machine learning have been made in this area to alleviate the burden of feature engineering through deep learning. This has motivated us to address these issues in our research work.

1.3 Thesis Research Objectives

The main objectives of our research study include:

- Survey and review existing techniques for blind detection and localization of image forgery.
- Empirically evaluate several texture-based methods for feature extraction to recognize authentic and forged images.
- Develop and explore new approaches for image forgery detection based on information fusion from multiple sources.
- Design and explore the effectiveness of convolutional neural networks in detecting image forgery.
- Evaluate and statistically benchmark the investigated approaches on different datasets.
- Publish and share our findings to the research community.

1.4 Thesis Organization

The thesis is divided into five chapters which discuss various topics as follows. Chapter 1 provides an introduction, defines the research problem and lists the research objectives. Chapter 2 gives brief background and intensively reviews related work. Chapter 3 describes the details of the research methodology. Chapter 4 describes the conducted experiments and discusses the results of various approaches on two publicly available datasets. Chapter 5 concludes the thesis and highlights some research directions for future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we briefly review the state-of-the-art of image forgery detection with focus on passive/blind detection techniques.

2.1 Passive Detection of Image Forgery

Several techniques for passive image forgery detection have been proposed in the literature. There are a number of surveys that have been published covering various aspects of these techniques, e.g. [3], [4], [5], [6]. Figure 2.1 shows a taxonomy of various categories of these techniques. More details are given in the following subsections.

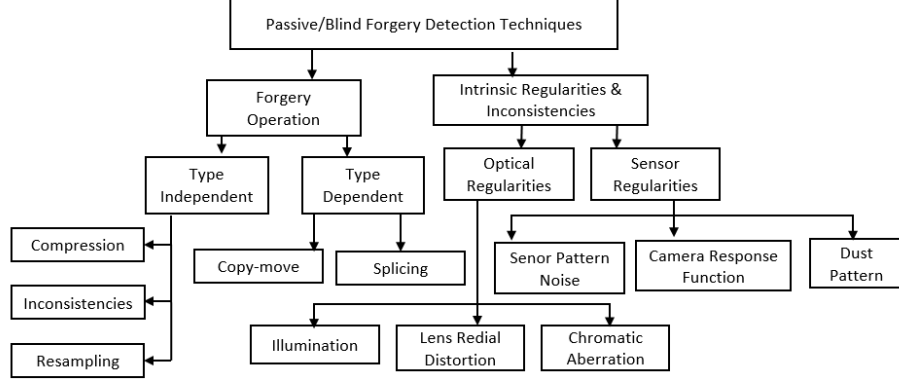


Figure 2.1: Taxonomy of image forgery detection techniques

2.1.1 Detecting Intrinsic Regularities and Inconsistencies

In digital cameras, image acquisition involves different stages. The light enters via lens which passes through Color Filter Array (CFA) of sensors. Today, most cameras are based on Complementary Metal-oxide Semiconductor (CMOS) or Charge-Coupled Device (CCD) sensors. The sensors use interpolation to estimate different colors. Image forgery results in altering the correlation in the interpolation step. Moreover, image enhancement methods, including contrast stretching, white balancing, and gamma correction are used before the final storage of an image. The artifacts introduced during these steps can be estimated to detect image tampering [5], [7]. An evaluation of a large number of approaches for image splicing localization using CFA pattern and noise pattern are presented in [8]. Some of the noise estimation algorithms are based on sensor Photo Response Non-Uniformity (PRNU), which differentiates the noise patterns based on the physical characteristics of an image with regards to the capturing device [9]. Geometric-based techniques focus on principal point (the projection on the image plane from center of the camera) which is near to the image

center. If the object in the image is translated, the principal point will be moved accordingly. Therefore, to detect the image tampering different principal points are estimated across the image [3].

Different light conditions are used to take natural photographs. In splicing operations, these light conditions may not match the original one. The inconsistencies in the light source are the main focus of physics-based techniques, which reveal whether a specific object belongs to the original scene or is identified as an artifact [10].

2.1.2 Forgery Operation Related Approaches

Other passive detection approaches are sub-categorized into type-dependent and type-independent methods. Type-dependent methods can be used to detect either copy-move or splicing whereas type-independent methods can be used to detect resampling, compression or inconsistencies.

Some techniques identify the statistical irregularities among pixels which have been introduced during the image tampering process. They investigate a correlation between pixels to detect specific forms of image tampering in different domains, e.g. spatial or frequency domain. These techniques includes copy-moves, splicing, resampling, and retouching [4], [5], [3].

JPEG compression properties have been exploited to trace the image forgery which is based on JPEG blocking based, double JPEG compression, multiple JPEG compression and JPEG quantization [3]. Due to different compression levels, the tampered regions result in a specific distribution. To identify tampered regions, different algorithms have been proposed with varying assumption that tampered regions can be

resampled or smoothed, or may lose their previous JPEG characteristics. These algorithms use features which can be derived via JPEG quantization or JPEG compression grid discontinuities [8].

Several researchers have employed image processing techniques to extract features and build machine learning-based models to detect image forgery [5], [6]. A general framework of this category of techniques is shown in Figure 2.2. The principal steps are image pre-processing, feature extraction, feature selection (dimensionality reduction), training/testing and post-processing. Section 2.2 gives some background on machine learning methods and Section 2.3 reviews related work.

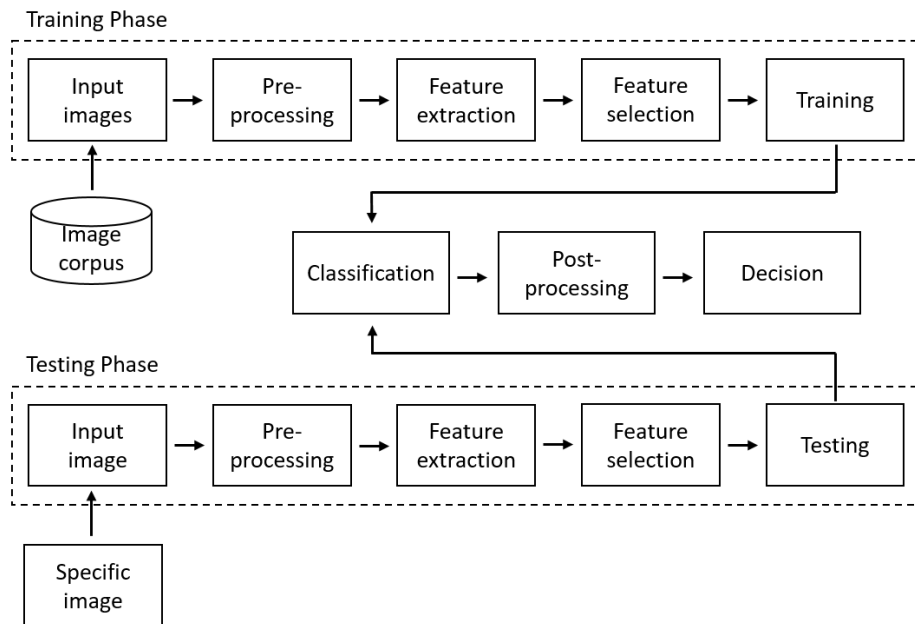


Figure 2.2: Image forgery detection framework

2.2 Machine-Learning Paradigms

Predictive modeling is a technique that uses historical data composed of input variables ($\mathbf{X}$) and output variable (Y) to approximate a mapping function (F). The approximation problems are categorized into classification and regression problems. The prediction will be quantitative, where the output variables can be real or discrete valued. In regression problems, the input variables ($\mathbf{X}$) and real-valued output variable (Y) are used to approximate the mapping function (F), while in a classification problem, the input variables ($\mathbf{X}$) and discrete output variable (Y) are used to approximate the mapping function (F). If the variables are time-dependent then the prediction will be called time series forecasting problem. In classification problems, the sample will be classified into classes. The two class problem is called binary classification, while more than two classes are called multi-class classification. The input variables also can be discrete or continuous.

Several machine learning techniques exist in the literature which belong to two main categories: supervised and unsupervised.

2.2.1 Supervised Machine Learning

Supervised learning algorithms are based on historical data where the labels or output classes are known. These algorithms are used to learn from the historical data to classify the unseen data. The algorithms repeat the learning process until they reach a satisfactory level of performance. The most widely used algorithms are Artificial Neural Networks (ANN) and Support Vector Machine (SVM).

Artificial Neural Networks

ANN borrowed the analogy from Brain Neural Network where dendroids collect information from other neurons. The information sent to a soma for processing which is the main body of neurons. Axon is the pathway via which neurons send information. The Axon is connected to dendroids and other neurons through synopses. Similarly, all the neurons are interconnected to each other and process the information and communicate with each other. In ANN, the dendroids become inputs, soma becomes a node, synapses become connection weights and the axon is the output as shown in Figure 2.3. The output is calculated using a non-linear function, also called activation or transfer function, of a weighted sum of inputs. Each connection carries a weight which can be adjusted during training to show the strength of the input. The activation function can take various forms such as step and sigmoid functions [11]. The neurons are typically stacked into layers which can be different in number. The starting layer is called input layer, the mid layers are called hidden layers, and the last layer is called output layer. Figure 2.4 illustrates a multi-layer feed-forward ANN structure with input, single hidden and output layers. Back-propagation is typically used to adjust the weights during the training phase.

Support Vector Machine

SVM is one of the supervised methods used for classification and regression analysis. SVM based classifiers classifies unseen data into one of the classes with high probability. It converts the data using a kernel function into a higher dimensional space where classes can be separated easily.

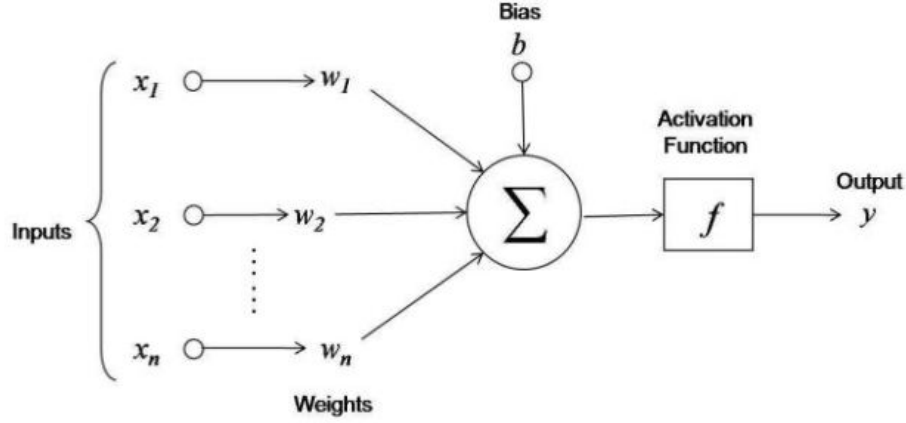


Figure 2.3: Single neuron model

For binary classification during training SVM learning algorithm determines a maximum-margin decision boundary, known as hyper-plane using a reduced set of the input data, known as support vectors. The output (y) is computed using:

$$y = W^T \phi(X) + b \quad (2.1)$$

where $\phi(X)$ is transformation function W^T is the transpose of the weighted vector and b denotes the bias. The parameter W and b should be tuned in such a way that the two classes can be separated by the following rule:

$$W^T \phi(x) + b = \begin{cases} -1 & \text{Class A} \\ +1 & \text{Class B} \end{cases} \quad (2.2)$$

SVM performs better for binary classification and linearly separable classes. It can also achieve better performance for higher dimensional data and nonlinearly separable classes. The nonlinearly separable data is converted from low dimensional to high

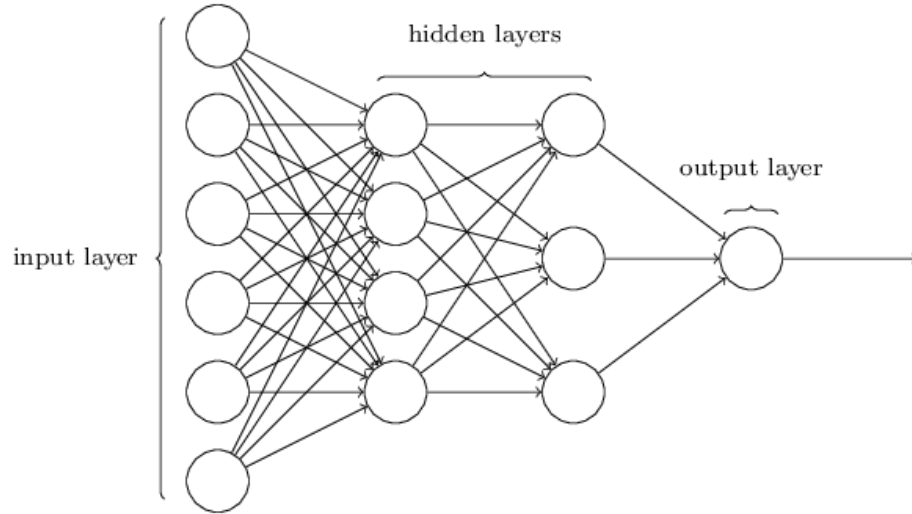


Figure 2.4: Multi-layer feed-forward ANN architecture

dimensional space and to reduce the computational cost SVM uses kernel tricks. There are different kernel functions such as Linear, Polynomial and RBF kernels. Each kernel has some parameters, which need to be tuned to achieve better performance. One of the techniques used to tune these parameters is called k -fold validation. Figure 2.5 shows the optimal hyperplane to classify each class while the points on the boundaries are the support vectors¹.

2.2.2 Unsupervised Machine Learning

Unsupervised learning does not need any prior knowledge about the actual class of the data. These algorithms group the data based on their structures. The K -means clustering is one the most commonly used approaches to group the data based on their behavior and distribution. It divides the unlabeled data into K groups where the K value has to be pre-defined. The algorithm randomly assigns a number of K

¹<http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf>

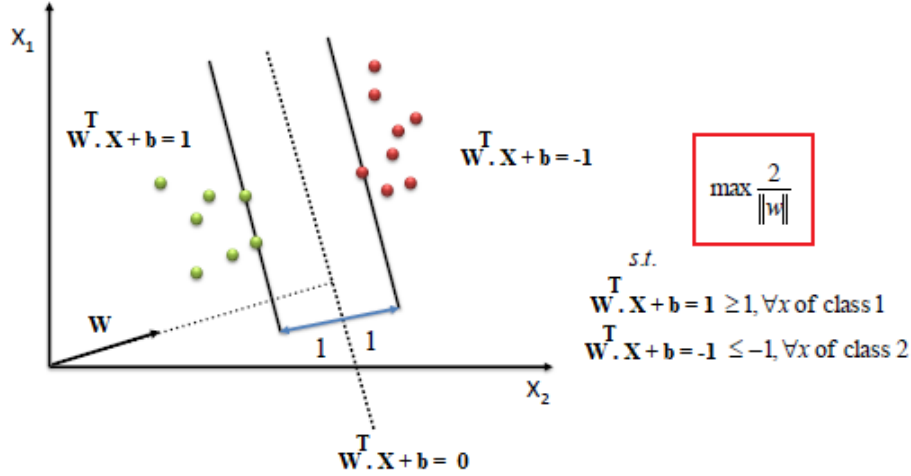


Figure 2.5: SVM optimal hyperplane

center points and find the distance of each data point from those K centers. Then it adjusts the K centers of the groups and repeats the process until it reaches an optimal solution. Hierarchical clustering is another type of unsupervised methods which uses tree structure (dendrogram) starting from a single point and progressively merged with other clusters with a defined threshold. The clustering algorithms use various distances which include Euclidean, Manhattan, and Mahalanobis distances. Figure 2.6 shows an example of K -means clustering where $K = 3$. Figure 2.7 shows an example of Hierarchical clustering.

2.2.3 CNN-Based Deep Learning Models

Deep learning is the sub-area of machine learning and Artificial Intelligence which uses multi-layer ANN to achieve state-of-the-art accuracy such as object recognition. Various deep learning architectures have been proposed in the literature for different problems. For image classification or object recognition, CNN is widely used due to

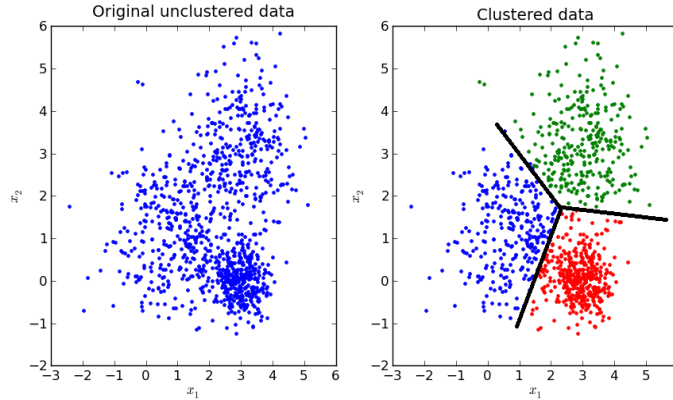


Figure 2.6: An example of K -means clustering where $K = 3$

its promising results. Some of the famous CNN models are discussed as follow.

LeNet: LeCun et al. [12] presented the first CNN in 1998 which was successfully used in practice. This model takes an input image, uses three convolution filters of 5×5 and applies a stride of 1. The network also uses sub-sampling (pooling) layers of size 2×2 with a stride of 2. Fully connected layers are used at the end of the network followed by an output layer. LeNet was used as a digital recognition system and its architecture is shown in Figure 2.8.

AlexNet: Krizhevsky et al. [13] presented AlexNet in 2012 which gained increasing attention and a new era of deep learning has started. The architecture of AlexNet consists of five convolutional layers with different strides, normalization, three max-pooling layers and three fully connected or dense layers. AlexNet was the first large scale CNN with 8 layers, which win the ImageNet competition with significant margin. The ReLu layer was also introduced in this network and also used dropout of 0.5. The input size of an image for AlexNet is $227 \times 227 \times 3$. AlexNet is similar to LeNet but consists of more layers as shown in Figure 2.9.

ZFNet: Zeiler and Fergus [14] developed ZFNet in 2013 and won the ImageNet

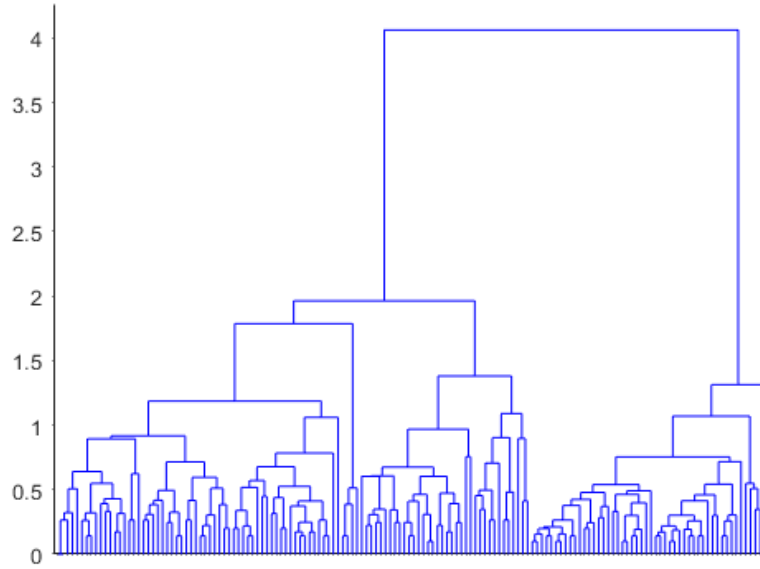


Figure 2.7: An example of hierarchical clustering

competition. The ZFNet consists of 8 layers which is similar to AlexNet but with improved hyper-parameters. ZFNet changed the number of filters, stride size and tuned the hyper-parameters to improve the error rate. Figure 2.10 shows the architecture of ZFNet.

GoogleNet: In 2014, GoogleNet [15] won the classification competition of the ImageNet challenge. This is the deeper network with computation efficiency which consists of 22 layers as shown in Figure 2.11. GoogleNet is designed efficiently which reduces the number of parameters with the use of an inception modules and stacked on the top of each other. The inception module is an idea of local network topology where it applies parallel filter operation on input from previous layer. The output of all these filters was concatenated depth-wise and passed to the next layer. GoogleNet uses auxiliary classification outputs to inject additional gradient at lower layers. It also

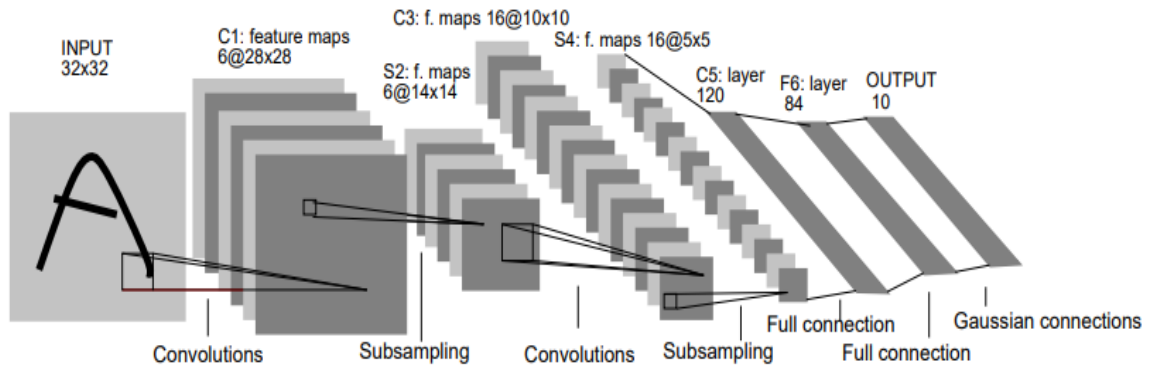


Figure 2.8: LeNet architecture

get rid of fully connected layers to save a lot of parameters and come up with a total of only 5 million parameters which is 12 times less than AlexNet. AlexNet parameters were 60 million. GoogleNet achieved 6.7% top-5 error rate in the competition.

VGGNet: In 2014, VGGNet [16] won the ImageNet competition for localization. The VGGNet consists of more layers but with smaller filters which makes it a deeper network. The inventors increased the number of layers from 8 (AlexNet) layers to up to 19 layers. They kept the 3×3 filters for all the convolutional layers throughout the network which was the smallest convolutional filter size. This network achieved 7.3% top-5 error rate on imageNet challenge and won the second place. Figure 2.12 shows

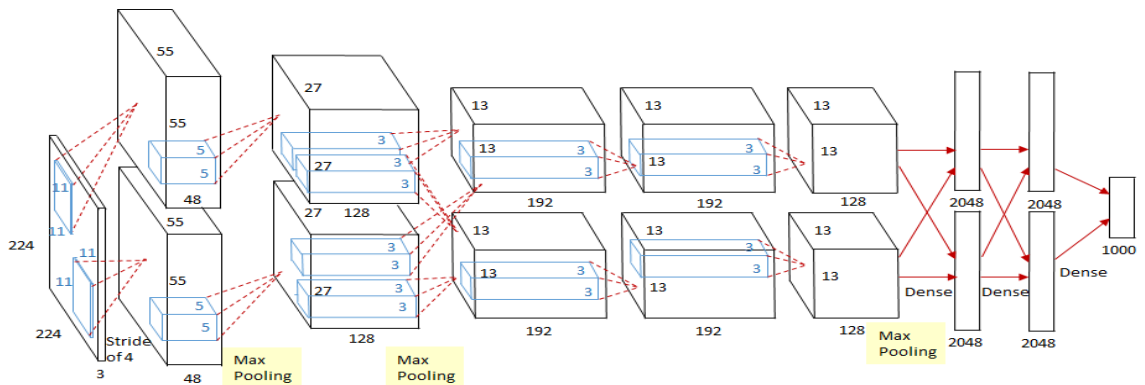


Figure 2.9: AlexNet architecture

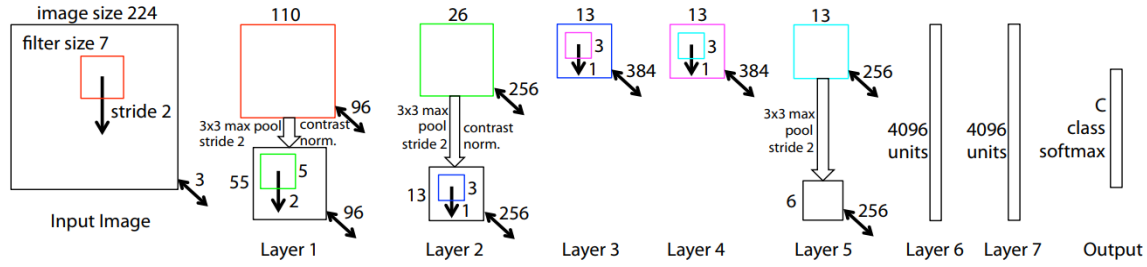


Figure 2.10: ZFNET architecture

the architecture of VVGNet.

ResNet: In 2015, ResNet [17] was introduced which is a very deep network using residual connections. It consists of 152 layers with the consistent filter size of 3×3 across the whole network as shown in Figure 2.13. This network swept all the classification and detection competition on ImageNet and COCO in 2015. The idea behind this network was to optimize the hyper-parameter for a very deep network which can work similar to shallow networks. The network used batch normalization heavily and did not use fully connected layers. ResNet is currently the state-of-the-art convolutional neural network model.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 2.11: GoogleNet architecture

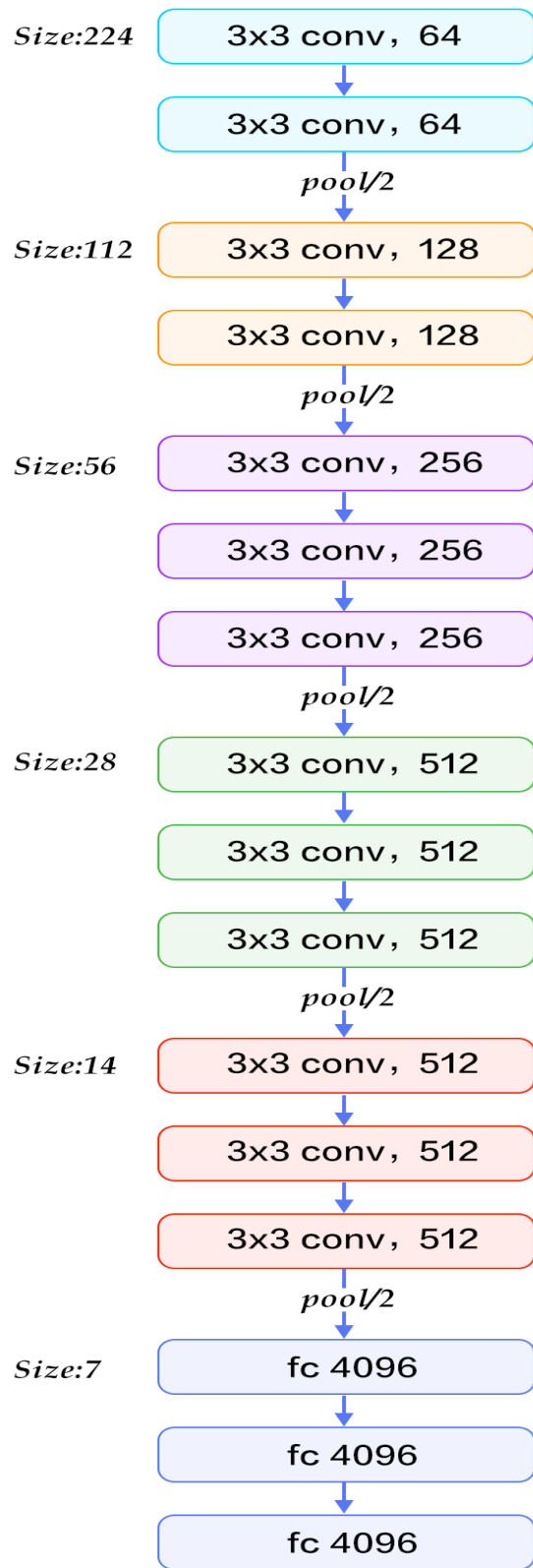


Figure 2.12: VVGNet architecture

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10 ⁹	3.6×10 ⁹	3.8×10 ⁹	7.6×10 ⁹	11.3×10 ⁹

Figure 2.13: ResNet architecture

2.3 Related Work

There is growing research activities in the area of image forgery detection and localization. Most of the methods proposed previously were based on machine learning techniques [18], [19] with good performance. However, such techniques depend on the quality and availability of training data and cannot be generalized more than learned data. Table. 2.1 and Table. 2.2 shows an overview of the previous work in the area of image forgery detection.

2.3.1 Spatial Features Based Approaches

Zhang et al. [20] proposed a method to detect median filtering (one type of image forgery) in digital images. Overlapping windows introduces correlation between Median Filter Residual (MFR) which is the difference between the median filter image and the original image. Hence, the transition probability matrices were calculated

across horizontal, vertical, main diagonal and minor diagonal from MFR. The transition probability matrices were used as features to detect median filter regions. Farooq et al. [21] used texture features using LBP with a rich model to detect image forgery. A sub-band strategy was adopted in [22] to evaluate the effect of each sub-model and make a generic scheme to detect the image forgery. Ten different types of forgery were detected using the IFS-TC² image forensic challenge dataset. The forgery types include, Gaussian blur, Gamma correction, JPEG compression, Median Filtering, up- & down-sampling, Contract Enhancement, Cropping, Copying and Splicing. The highest accuracy of 98.4% was achieved with a standard deviation of 0.2.

David et al. [19] proposed an approach to detect image forgery localization with the fusion of three different tools including sensor noise, block matching and machine learning. They applied camera identification phase, then estimation for the camera fingerprint followed by the localization step. These were preliminary steps to apply sensor noise tool. The machine learning tool was used for local descriptors. In the last step, a decision fusion strategy was applied to get more satisfactory results. Khosro et al. [23] proposed an approach to identify tampered pixels by using blur type inconsistencies. The technique was applied only on blurred images. The image was divided into blocks and for each block, a blur type feature was calculated. A splice localization was applied to the whole image to increase the region boundary precision. The blur type differences are applied to get the inconsistency for localization.

Mohsen et al. [24] proposed an approach to detect tampered images based on feature vector matching and modified fractional coding. The approach is used to detect

²<http://ifc.recod.ic.unicamp.br/fc.website/>

copy-move forgery. It also detects multiple copied regions. The method was robust to scaling, rotation, Gaussian blurring and brightness. Silva et al. [25] extracted key points of image by using Speeded Up Robust Features (SURF). The distance between points was calculated by Nearest Neighbor Ratio (NNR) and Euclidian distance was used to find the similarity between points. The voting process was used for final decision. The method showed an accuracy of 96.19% on CMEN dataset [26] and was robust to JPEG compression, rotation and scaling.

Zhao et al. [27] explored the chroma channels and four gray level Run-Length Run-Number (RLRN) vectors in various direction were extracted as features. The result shows that chroma channels performed better than RGB channels which achieved 94.7% accuracy on CASIA v1.0 and 85% on Columbia³ dataset. To detect copy-move forgery, Zhou et al. [28] used various image features including fuzzy color, color moments, edge and color directivity descriptor, texture histogram and edge histogram. The results showed robustness against noise, JPEG compression and gamma correction. The proposed work showed high accuracy to detect tampered images. However, the performance degrades on translation, scaling and rotation.

Sutthiwan et al. [29] detected color forged images by using chrominance components. statistical moments were extracted based on Markov features to achieve better performance. The same authors used Rake transform [30] via edge statistics to detect image forgery. Amerini et al. [31] used Scale Invariant Feature Transform (SIFT) to detect copy-move forgery and geometric transformation of replicated objects. The SIFT method matches the same points in the image via which we can identify different

³<http://www.ee.columbia.edu/ln/dvmm/downloads/AuthSplicedDataSet>

copy-move objects. The proposed work showed 93.42% true positive rate. Ustubioglu et al. [32] extracted structure information from images using LPQ and applied SIFT to match the key-points. Weber Local Descriptor (WLD) is a texture descriptor which has been also used for forgery detection. This descriptor has two components: differential excitation, which is used to extract salient variations in an image, and orientation component, which is used to find the pixel gradient orientation [33]. Hussain et al. [34] proposed Multi-Scale WLD based method. WLD was used to extract features via chrominance components which were fed to SVM to classify the images. CASIA v1.0⁴ benchmarks dataset was used for evaluation which achieved the accuracy of 93.33%.

In the copy-move forgery when an image region is copied from one to another place, some inconsistencies are introduced called Lateral Chromatic Aberration (LCA). Mayer et al. [35] proposed a method based on LCA to detect image forgery. To detect the forged region more accurately, a statistical model was used to capture local and global LCA inconsistencies. To estimate the LCA more efficiently, block matching method was used, called search diamond which helps to detect misalignment which is introduced due to LCA in local regions. The method used various camera models images including Sony, Canon, Nikon, Olympus, Samsung and Olympus for experiments. Emam et al. [36] used Difference of Gaussian (DoG) and extracted local extrema points to detect the copy-move forgery. A multi-support region order-based gradient histogram method was adopted to improve the robustness of the algorithm. The proposed method localizes the copied regions of the CMFD dataset [26] and the results were compared to the SIFT and SURF based methods. Ghoneim et al. [37]

⁴CASIA1 v1.0 and CASIA v2.0 : <http://forensics.idealtest.org/>

worked on medical image forgery in a real-time application for smart healthcare. The proposed method is based on noise estimation, where multi-resolution regression was applied to the image noise map. The feature vector was fed to the SVM and extreme learning classifiers to detect forged and authentic image.

Chou et al. [38] used block based Local Gabor Wavelets Pattern (LGWP) to detect copy-move forgery. The method was rotation-invariant due to using uniform LBP. The duplicated regions were detected while the image was distorted using brightness, contrast adjustment, blurring and compression. To evaluate the proposed method, Correct Detection Ratio (CDR) was used as a performance measure on CoMoFoD dataset.

Abdel-Basset et al. [39] used SIFT features to detect the copy-moved regions and used two level clustering strategy to improve the detection rate. In the first level, the points were clustered based on noise matches with the assumption that distribution of points belonging to a specific cluster should be denser than the outside points. In the second level, the Random Sample Consensus (RANSAC) was used to improve the clustering efficiency.

Prakash et al. [40] used Harris corner and Adaptive Non-Maximal Suppression (ANMS) to copy-move forgery. First, the interest points were detected using Harris corner algorithm, SIFT was applied on each point as descriptor and RANSAC was used to detect the matching regions. The algorithm was evaluated using various transformations including, scaling, rotation and JPEG compression.

2.3.2 Spatial Features with Transform Based Approaches

Fu et al. [41] used Hilbert-Huang Transform (HHT) based features to exploit the non-linearity and non-stationarity of splicing operation. Statistical moments were calculated for wavelet sub-bands as features to increased the accuracy up to 80.15%. Li et al. [42] used HHT with a combination of a histogram of DWT coefficients which increases the accuracy to 85.8%.

Dong et al. [43] used edge statistics and image run length as features. The authors also used coherence and discontinuity in correlation between the image pixels to improve accuracy. Chen et al. [44] detected sharp lines, corner and edges which were introduced via splicing operation. The authors used statistical moments of wavelet sub-bands, 2D phase congruency to identified spliced images with an accuracy of 82.3%. However, the feature extraction complexity was the main problem for this approach.

Shi et al. [45] used statistical feature based on 1-D and 2-D moments. However, these features are computationally expensive. The same authors in [46] used Markov features in the DCT domain, which are based on Transition probability matrix. Alahmadi et al. [47] proposed a method to detect tampered images by using texture features using LBP with DCT. The SVM was used to classify the images as tampered or authentic. The experiments were carried out on three different benchmark image datasets including CASIA v1.0 , CASIA v2.0 and Columbia [48] which achieved 97%, 97.5% and 97.7% accuracy, respectively.

Zhang et al. [49] extracted Markov based features in contourlet transform domain

and DCT domain to detect image forgery. For DCT, the inter-block and intra-block features were used, whereas for contourlet domain they extracted features to characterize dependencies among sub-bands. Then, Markov probability matrices were computed for different color channels. SVM-REF was used to handle the large feature dimensions and traditional SVM was used as a classifier. Columbia benchmark dataset [48] was used for experiments with an accuracy of 96.6%. Hsu and Wang [50] used Gabor filter with DCT to detect the copy-move forgery. Gabor filter was applied to get the image features with different frequency, scaling and rotation. They used two datasets (MICC-F220⁵ [31] and Caltech⁶ [51]) for the copy-move manipulation with an accuracy of 98.86%. However, the proposed work only detects replicated objects in an image.

El-Alfy and Muhammad [52] proposed a technique for blind detection of image forgery. Markov features were used in spatial and DCT domain. PCA was used for features dimension reduction. The SVM was used to classify the images. The same authors in [53] proposed a robust method to detect gray and color image forgery. The intra-block Markov features were used for both DCT and LBP. These features were fed to SVM with radial-basis kernel to classify images. These features were fused together to get high accuracy.

DCT coefficients which were mostly used for forgery detection are analyzed for a mixture of modified and unchanged components. The separation between these two components requires estimating two parameters: the primary quantization step Q_1 ,

⁵<http://lci.micc.unifi.it/labd/2015/01/copy-move-forgery-detection-and-localization/>

⁶<https://authors.library.caltech.edu/7694/>

and the portion of the modified region α . To accurately approximate these parameters, Liyang et al. [54] proposed an estimation approach. The UCID [55] dataset was used for experiments.

Zhang et al. [56] used DCT coefficients and LBP histogram as a feature vector to detect image splicing. To reduce the complexity and features dimensionality, PCA was used and achieved 89.93% accuracy. The same authors in [57] proposed another model which shows the statistical difference between authentic and spliced images. This approach used Multi-size Block Discrete Cosine Transform (MBDCT) and statistical image quality metrics which include mean square errors, normalized cross-correlation and moment-based features and achieved 87.10% accuracy.

Wang et al. [58] used edge information of image chrominance component using a finite-state Markov chain. They extracted a low-dimensional vector which is fed into the SVM to detect the image authenticity. The accuracy was 95.6% on CASIA v2.0. This work has been extended in [59] where Markov features have been extracted by using transition probability matrices in the DCT and DWT domains. The RFE-SVM was used with 89.76% accuracy on Columbia and 93.55% on CASIA v2.0.

Muhammad et al. [60] proposed an Steerable Pyramid Transform (SPT) which is an image decomposition technique with different scale and orientation to detect image forgery. The image was converted into YCbCr channel and SPT was applied on Cb and Cr channels to extract multi-scale and multi-orientation of sub-bands. The SPT was combined with LBP to form a feature vector which achieved 94.89% accuracy with CASIA v1.0. Birajdar et al. [61] detected rescaling operation. High-order statistical features were extracted through Quadrature Mirror Filter (QMF). To enhance the

performance feature reduction was performed via fuzzy entropy and ANN was as a classifier.

Hayat and Qazi [62] extracted features in DWT and DCT domains for overlapping blocks to detect image forgery. The image was converted to YCbCr format and divided into overlapping blocks with a pre-processing step, including the noise removal and normalization. The DWT sub-bands were extracted and only the lowest frequency sub-band was chosen for further processing. DCT was applied on each block to extract the high energy coefficient as features. These features were sorted in a lexicographical way that similar features become closer to each other, and correlation was extracted for these features. A threshold was used to classify forged and authentic images. The forged area was also localized using masking and achieved an accuracy of 73.62% with standard deviation of 5.09.

Aghamaleki et al. [63] detected and localized tampered regions in MPEGx coded videos. Frame insertion and deletion were detected in videos using residual errors. The quantization error of frames was studied mathematically and inspected to detect and localize the tampered frames. A wavelet-based algorithm was proposed to detect the inserted, deleted and double compressed frames in a frequency domain. The proposed approach was tested on 22 videos⁷ dataset and achieved the average accuracy of 92.73%.

Mahmood et al. [64] divided the gray images into overlapping blocks and extracted the DCT of each block. The Gaussian RBF kernel PCA based features were extracted from the DCT blocks and matched the blocks. This way the duplicated blocks were

⁷<http://trace.eas.asu.edu/yuv/> and <https://media.xiph.org/video/derf/>

extracted while removed the isolated block which will detect the copy-move or copy-paste forgery within an image. The proposed method was evaluated via Gaussian blur, Additive white Gaussian noise and JPEG compressed images. In another paper [65], copy-move forgery was detected using Local Binary Pattern Variance (LBPV) over stationary wavelet transform blocks. The features were extracted through overlapping circular blocks instead of square blocks. The proposed work was evaluated by changing the image using blurring, rotation, color reduction, brightness and quality levels.

Issac and Wilscy [66] proposed a local Gabor Phase Quantization (LPQ) based method to detect imagery. The image was first converted into YCbCr channel and Gabor Wavelet Transform (GWT) and Local Gabor Phase Quantization (LGPQ) were applied to chroma channels with different scales and orientation to extract texture features. To optimize the features, Non-Negative Matrix Factorization (NMF) was used.

Alkawaz et al. [67] proposed a method based on DCT using overlapping blocks to detect copy-move forgery. The RGB image was first converted into grayscale and divided into blocks. The zig-zag scanning was applied on each block and sorted the feature vector into lexicographical order. Euclidean distance was used to identify the duplicated block. CoMoFoD dataset was used for the experimentation and a block size 8×8 performed better in terms of precision and recall.

Agarwal et al. [68] used Robust Invariance Co-occurrence Local Binary Pattern (RICLBP) with Stationary Wavelet Transform (SWT) frequency domain to detect image forgery. The results were evaluated using Colombia and CASIA v1.0 datasets with three performance measures: accuracy, true positive rate and true negative rate.

2.3.3 Deep Learning Based Approaches

In recent years, deep learning got attention for its remarkable performance. Different deep neural network architectures were used to address various issues. Deep auto-encoder [69], Deep belief network [70] and Convolutional neural networks [12] were used to extract complex feature maps from high dimensional inputs and learn efficiently to classify images. Rao et al. [71] used a deep learning approach to detect splice and copy-move forgery. The weights were initialized with high pass filter for the first layer which will catch the artifacts which cause image tampering.

Zhou et al. [72] divided the image into blocks and labeled them as forged and authentic. A 7 layer CNN model was used to extract features and which were fed to SVM to classify the image. One way pooling was used instead of two-way pooling in the feature extraction phase through CNN leading to higher accuracy. However, all of these three papers [71], [73], [72] used the same rich models [22] in their first convolution layer which can extract more discriminated features. Bayer et al. [74] proposed a CNN approach for image forgery detection. The architecture was capable of learning adaptive features to detect the manipulated regions. This method achieved high accuracy with an average of 99%. However, the proposed method was only tested on filtered images like, Median filtered image, Gaussian Blurred image and re-sampling.

Davide et al. [75] proposed a single image splice localization by using auto-encoders. High pass filter was applied on images and slid a Kernel window to extract the vertical and horizontal small patterns and co-occurrence. The histogram

of these features was normalized to provide 180 features. These features were fed forward to auto-encoder to localize the tampered regions in an image. Liu et al. [73] also adopted the same procedure for patch extraction as Roa et al. [71] to make a huge dataset. They used multi-scale CNN and the outputs were fused to get the final image. Different size of patches were extracted and CNN was applied on each patch size to get the multi-scale outputs. The first layer of CNN has used 30 high-pass filter of SRM [22] to initialize the weights which also worked as a regularizer.

Chen et al. [76] proposed a median filter based deep learning approach. They used an additional filter in a convolutional neural network. BOSSBASE 1.01⁸ and UCID datasets were used to test the algorithm which showed an accuracy of 93.21%. Bunk et al. [77] used resampling features with deep learning to detect and localize tampered regions of an image. The Radon transform was computed for resampling features on overlapping patches of an image. Gaussian conditional random fields were used to create heat maps and Random walk segmentation method was used to extract the tampered regions. The resampling features were also used with LSTM algorithm for localization and classifications.

Mohammed et al. [78] combined the various approaches of resampling and copy-move features to boost the image forgery detection. The work of Cozzolino et al. [79] used a pre-filtering step and used a patch match algorithm to detect duplicated blocks. The image was passed first to the copy-move forgery detection method where it increased the precision for resampling forgery detection; this approach increased the overall forgery detection rates by 8% to 10%.

⁸<http://bows2.ec-lille.fr/>

Some researchers focused on JPEG images because due its double compression, it leaves some artifacts during the re-compression stage. The image forensic tools should not only be limited to JPEG format but also need some other image compression types. For this purpose, Zhang et al. [80] proposed a deep learning approach to detect image forgery and localization among different images formats. The complex features were used for each patch of an image. To identify these features, a stacked auto-encoder model was used in the first step, in the second step contextual information was extracted for each patch which increased the classification accuracy. The deep learning concept was used to identify tampered regions. The dataset used was CASIA v1.0 with two different types of formats including JPEG and TIFF with 91.0% accuracy.

In practice, very few people train deep learning models such as CNN's from scratch because it needs large datasets, high computational GPUs and time. However, transfer learning uses a pre-trained model with learned knowledge of the same task, as a starting point for a new model to solve a problem [81]. This technique speeds up the training and improves the performance of newly developed deep learning models. Transfer learning can be used with three strategies. First, CNN as fixed feature extractor. A last fully connected layer of pre-trained CNN will be removed and the rest of the CNN will be treated as fixed feature extractor [82]. Second, the CNN model will not only be retrained on new dataset but also fine tune the weights of the pre-trained network via backpropagation. Instead of all layers, only a few layers can be retrained while keeping the higher level layers as fixed layers to avoid over-fitting. The third strategy is using the pre-trained model weights for new models. The pre-trained weights are trained on large datasets such as ImageNet which consists of 1.2

million images with 1000 categories are beneficial new models for fine-tuning.

Lucena et al. [83] used transfer learning with CNN to recognize face anti-spoofing and achieved state-of-the-art results with the accuracy of 99.04%. Ng et al. [84] applied transfer learning on small datasets of emotion recognition. The pre-trained network on imageNet was used as a starting point followed by the supervised fine-tuning which improves the test accuracy to 55.6% instead of 39.13% which was the challenge baseline.

Zhan et al. [85] detected five types of forensics including additive Gaussian noise, JPEG compression, contrast enhancement, resampling and median filtering using transfer learning. This work followed the standard transfer learning technique, where the pre-trained network (base network) layers were copied to the initial layers of target network [81]. The rest of the layers of the target network were initialized with random weights and trained toward the target task. The proposed model achieved 97.5% average accuracy to detect five different image manipulations.

Approach	Paper	Features/Method	Application	Dataset	Performance
Feature Based	Fu et al. 2006 [41]	HHT + Statistical movements + SVM	Image splicing	Columbia	Acc= 80.15%
	Li et al. 2010 [42]	HHT + DWT + SVM	Image splicing	Columbia	Acc= 85.8%
	Dong et al. 2009 [43]	Statistical moments + Edge Statistics + SVM	Image splicing	Columbia	Acc= 84.36%
	Chen et al. 2007 [44]	Statistical moments + 2D phase congruency	Image splicing	Columbia	Acc= 82.3%
	Shi et al. 2007 [45]	Markov features + DWT	Image splicing	Columbia	Acc= 91.87%
	Zhang et al. 2008 [57]	Multi-size block DCT	Image splicing	Columbia	Acc= 87.10%
	Zhang et al. 2012 [56]	LBP + DCT + PCA	Image splicing	Columbia	Acc= 89.93%
	Zhang et al. 2016 [49]	Markov features + DCT	Image splicing	Columbia	Acc= 96.6%
	Mahmood et al. 2016 [64]	DCT coefficients + Block Matching + PCA	Copy-move forgery	Columbia	Precision = 0.97
	El-Alfy & Muhammad 2015 [52]	Markov features + DCT + SVM	Image splicing	Columbia	Acc=98.4%
	He et al 2012 [59]	Markov features + DCT + DWT	Image splicing	Columbia, CASIA v2.0	Columbia Acc= 93.55%, CASIA v2.0 Acc= 89.76%
	Hussain et al. 2013 [34]	Multi-scale WLD	Image tampering	CASIA v1.0	Acc =93.33%
	Muhammad et al. 2014 [60]	LBP + SPT	Image tampering	CASIA v1.0	Acc= 94.8%
	Sutthiwan et al. 2010 [29]	Moments + Markov features + Boosting features selection + SVM	Image tampering	CASIA v1.0	Acc= 94.5%
	Alahamdi et al 2016 [47]	LBP + DCT + SVM	Image tampering	CASIA v1.0, CASIA v2.0	CASIA v1.0 Acc= 97%, CASIA v2.0 Acc= 97.5% (low AUC)
	Zhao et al. 2010 [27]	RLRN + SVM	Image splicing	CASIA v1.0, Columbia	Acc= 85%
	Wang et al. 2010 [58]	Finite-State-Markov chain	Image tampering	CASIA v2.0	Acc= 95.6%
	Issac and Wilscy 2017 [66]	GWT + LGPQ	Image tampering	CASIA v1.0, CASIA v2.0	CASIA v1.0 Acc= 96.3%, CASIA v2.0 Acc= 99%.
	Agarwal & Chand 2018[68]	Rotation-invariant co-occurrence LBP + SWT	Image tampering	CASIA v1.0, Columbia	CASIA v1.0 Acc = 96.8%, Columbia Acc=92.3%
	Zhou et al. 2016 [28]	Fuzzy color + Texture histogram features	Copy-move forgery	CoMoFod	Acc ratio = 83.6%
	Hsu and Wang et al 2012 [50]	Gabor filter + DCT	Copy-move forgery	MICC-F22 [31], Caltech [51]	Acc= 98.8%
	Silva et al. 2015 [25]	SURF + Euclidean distance	Copy-move forgery	CMEN	Acc= 96.1%
	Khosro et al. 2015 [23]	Blur type features	Blur splicing regions	1200 natural blurred images	Acc = 96.0% (Splicing) Acc = 94.6% (blur splicing regions)

Table 2.1: Literature review summary part I

Approach	Paper	Features/Method	Application	Dataset	Performance
Feature Based	David et al. 2014 [19]	Block matching + Sensor Noise + SVM	Image tampering localization	Image Forensics Challenge 2013	F-measure = 0.4072
	Hayat and Qazi 2017 [62]	High energy coefficient of DWT + DCT	Image tampering		False detection rates =6.47% Acc=73.62% with SD 5.09
	Farooq et al. 2017 [21]	LBP + Rich Model	Up/down-sampling, cropping, scaling and blur	IFS-TC	Acc= 98% with SD 0.2
	Aghamaleki et al. 2016 [63]	Double Compression + Quantization error	Insert/delete frame detection	22 videos	Acc = 92.73%
	Zhang et al. 2014 [20]	MFR + Transition probability metric	Median Filtering detection	NRCS [86]	Quality factor Q = 90
	Mahmood et al. 2017 [65]	LBPV + Stationary wavelets	Copy-move forgery	CoMoFoD , KLTCI	Avg True Detection Rate = 0.98
	Ustubioglu et al. 2015 [32]	LPQ + SIFT	Copy-move forgery		High match points
	Emam et al. [36]	DoG	Copy-move forgery	CMFD	Prec= 98.70, F1-score = 79.83
	Alkawaz et al. 2018 [67]	DCT coefficients	Copy-move forgery	CoMoFoD	Avg prec= 64.509, recall= 96.5
	Ghoneim et al. 2018 [37]	Noise maps + multi-resolution regression	Image tampering	DDSM [87]	Acc = 84.3%
	Chou et al. 2018 [38]	LGWP	Copy-move forgery	CoMoFoD	CDR=0.99
	Abdel-Basset et al. 2018 [39]	2-level clustering	Copy-move forgery	MICC-F2000, MICC-F220	TPR = 95.63%, 97.87%
	Prakash et al. 2018 [40]	ANMS + RANSAC	Copy-move forgery	CoMoFod	Acc = 87.5%
					Acc= CASIA v1.0 =97.6% CASIA v2.0 =97.86%, Columbia =97.04%
Deep Learning	Zhou et al 2017 . [72]	CNN	Image tampering	CASIA v1.0, CASIA v2.0, Columbia	
	Bayer et al. 2016 [74]	CNN	Median filtered image, Gaussian blurred image and re-sampling forgery	12 different camera model images	Acc= 99%
	Rao et al. 2016 [71]	SRM-CNN	Image tampering	CASIA v1.0, CASIA v2.0, Columbia	Acc= CASIA v1.0 =98% AUC 0.99 CASIA v2.0 =97.8% AUC 0.98, Columbia =96.3% AUC 0.98
	Davide et al. 2016 [75]	Auto-encoder	Compressed and rescaling	DSO-1 [88]	F-measure = 0.418
	Liu et al. 2017 [73]	Multi-scale CNN	Forgery localization	RTD	F1-Score = 0.4063
	Chen et al. 2015 [76]	CNN	Median filter image forgery	BOSSbase, UCID	Acc= 96.34%
	Zhang et al. 2016 [80]	Stacked Auto-encoder	Forgery localization	Columbia, CASIA	Acc= 91%
	Bunk et al. 2017 [77]	DNN + LSTM	Forgery localization	UCID, RAISE	Acc= 94.85%
	Zhan et al. 2017 [85]	Transfer learning	resampling, median filter Gaussian noise, JPEG compression Contrast enhancement forensics	Bossbase, BOW	Acc = 97.5%

Table 2.2: Literature review summary part II

CHAPTER 3

METHODOLOGY

In this chapter, we will describe the various methods that we have explored for image forgery detection. We divide our work into two main areas. The first one is based on feature extraction and classification using SVM and the second one is based on deep learning through a CNN approach.

3.1 Feature Extraction Based Methods

Feature extraction is one of the fundamental steps to differentiate between authentic and forged images. These features convert the visual information into statistical numbers to reduce the large amount of data. Initially these features can be large in size and redundant which needs a large amount of memory and high computation. However, due to the feature selection process, the most discriminatory features will be extracted which can still classify the images without jeopardizing accuracy which requires small memory and low computation. There are various methods to extract features which can be in spatial domain, transform domain or hybrid, which combines

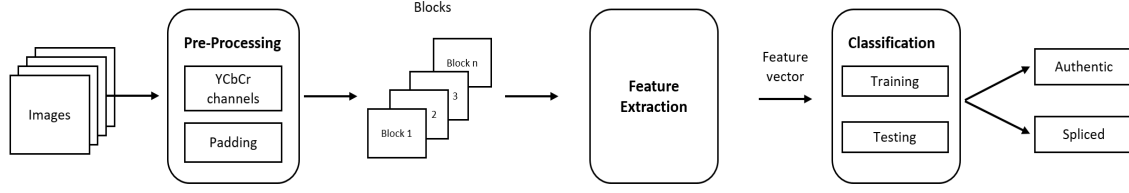


Figure 3.1: A typical layout for image forgery detection

both the spatial and transform domains.

A typical layout of image forgery detection is shown in Figure 3.1. It has four main stages, which will be described in more details in the following subsections.

3.1.1 Pre-Processing

In the preprocessing step, the RGB images are converted to YCbCr components, where Y is luminance (luma), and Cb and Cr are the chrominance (chroma) components. The chroma components can detect more tampering than other color channels [89]. The conversion from RGB channels to YCrCb is as follows:

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & 74.203 & 112.0 \\ 112.0 & 93.786 & 18.214 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (3.1)$$

3.1.2 Texture-Based Feature Extractors

Each color channel is divided into blocks then each block is transformed by one of the methods explained below. Subsequently, we apply two different models to obtain feature vectors for each channel. In the first approach, the feature vector is formed

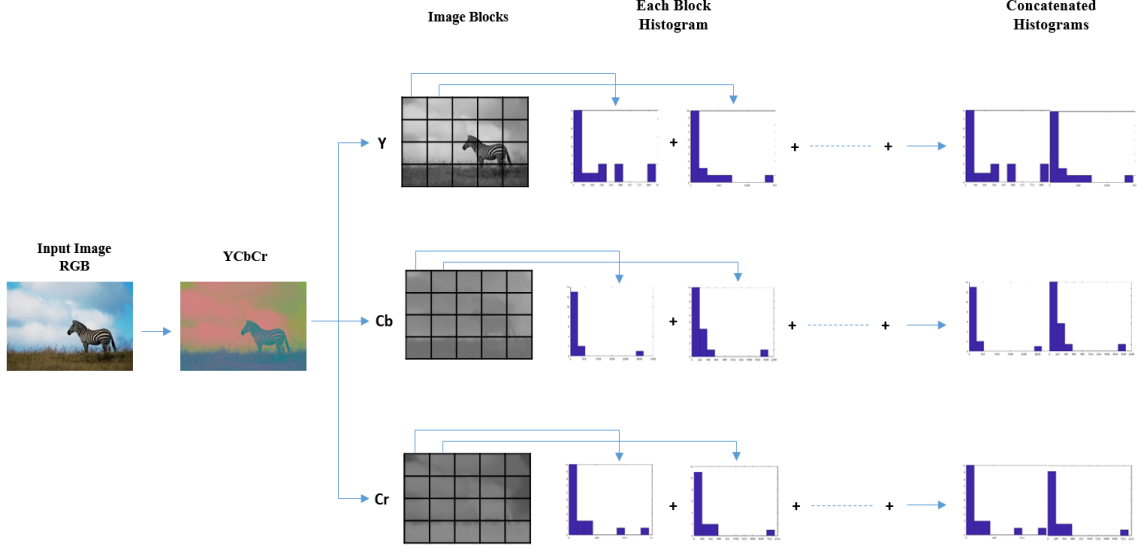


Figure 3.2: Flow diagram of feature extraction using histogram concatenation

by concatenation of the histograms of various blocks as shown in Figure 3.2. In the second approach, the feature vector is formed by computing the standard deviation of the stacked block histograms as shown in Figure 3.3.

In the following, we describe various transformation methods that have been investigated in our study.

Local Binary Pattern (LBP)

LBP [90], [91] is a visual descriptor that is widely used for texture analysis and classification. LBP describes the relative information around the pixels generating a bit-code. It is usually applied to a single channel to extract the intensities. The simple form of LBP generates a binary code by using 3×3 block. Pixels on the border are compared to the center pixel and a binary 1 is assigned if the surrounding pixel value is greater than the center pixel. It will generate a binary 0 otherwise. The 8 bits correspond-

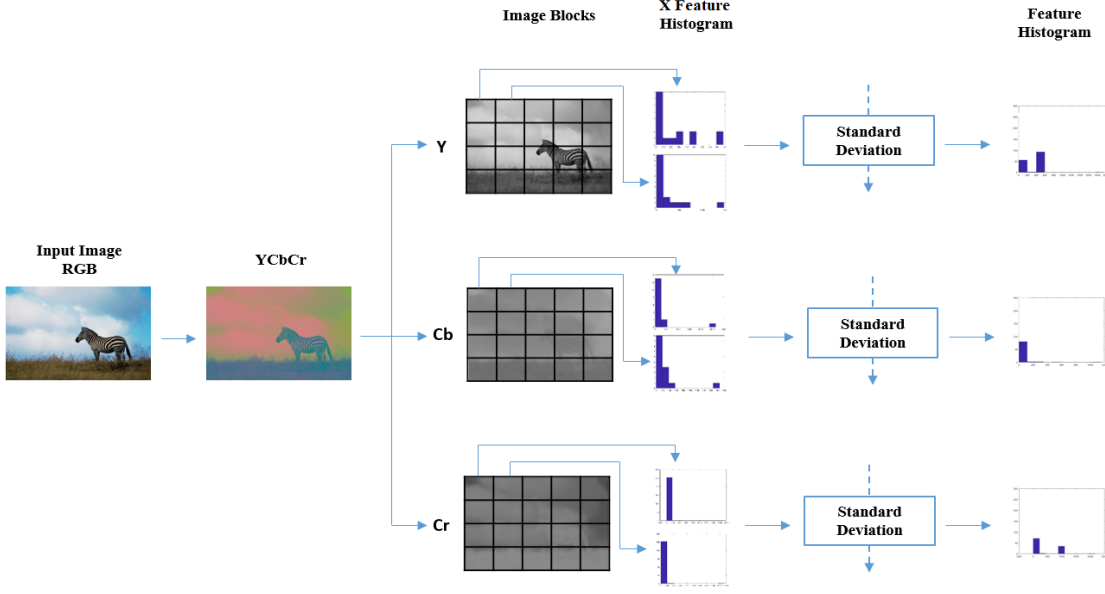


Figure 3.3: Flow diagram of feature extraction using histogram stacking

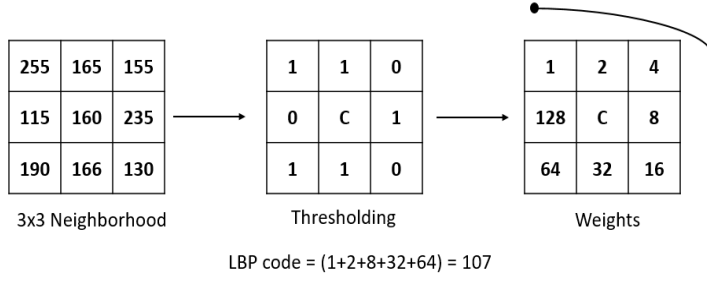


Figure 3.4: LBP example with 3×3 block

ing to the 3×3 block will be converted to a decimal value which can be used as a descriptor for the center pixel. Figure 3.4 illustrates an example of LBP descriptor. The histogram of all decimal codes describes the texture feature of an image.

$$LBP_{P,R} = \sum_{i=0}^{P-1} f(P_i - P_c) 2^i \quad (3.2)$$

LBP is denoted as $LBP_{P,R}$ where P is the number of surrounding points, whose values are denoted P_i , with radius R of center pixel P_c , $f(.)$ is a threshold function

defined as:

$$f(P_i - P_c) = \begin{cases} 0, & \text{if } P_i - P_c < 0 \\ 1, & \text{if } P_i - P_c \geq 0 \end{cases} \quad (3.3)$$

Center Symmetric Local Binary Pattern (CSLBP)

CSLBP [1] uses a similar approach as LBP but it compares opposite pixels instead of the center pixel which gives a shorter histogram [92]. The traditional LBP extracts a vector of 256 (2^8) while the CSLBP extracts a feature vector of 16 (2^4) for a single point. CSLBP adopted radius (R) 1 and points (P) 8 similarly used in traditional LBP as shown in Figure 3.5. CSLBP is used as a regional descriptor with a higher stability in flat regions of an image.

CSLBP does not compare the central pixel to the neighbor but computes opposite symmetrical pixels against the central pixel. Moreover, to increase the robustness of flat regions of an image, the non-zero thresholding was introduced. CSLBP can be defined as:

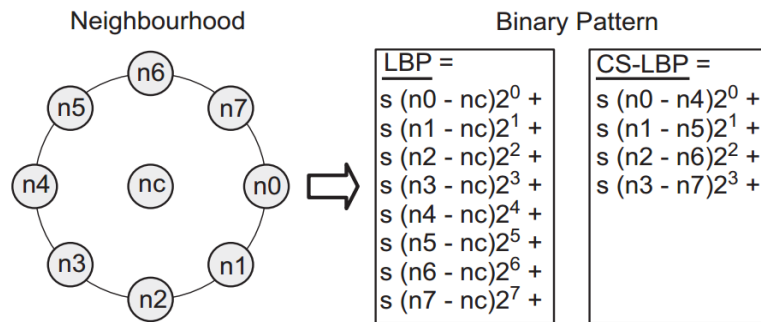


Figure 3.5: LBP and CSLBP with R=1 and P=8 [1]

$$CSLBP_{P,R,T} = \sum_{i=0}^{(P/2)-1} f(g_i - g_{i+P/2} - T)2^i \quad (3.4)$$

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & otherwise \end{cases} \quad (3.5)$$

where R, P and T represents, radius, number of points and threshold value. The T value in this work was set to 0.1.

Shift Local Binary Pattern (SLBP)

In traditional LBP, one pixel position generates one local binary code which is corresponding to one bin in the histogram. However, in SLBP [93] a fixed number of local binary codes are generated for each pixel position. In traditional LBP, the neighboring pixels P and radius R are taken into account while in SLBP another T parameter is added, which is bounded with limits $\pm l$. The change in the value of T will generate a new binary code and will be added to the histogram.

$$SLBP_{P,R} = \sum_{i=0}^{P-1} S(g_i - g_c - T)^{2^i} \quad (3.6)$$

where P represents the number of neighbors points with radius R , f and T defined as follows:

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & otherwise \end{cases} \quad (3.7)$$

$$T \in [-l, +l] \cap Z \quad (3.8)$$

where $-l$ and $+l$ are the lower and upper limits. For each pixel position, K number of binary codes will be generated. K is defined as:

$$K = 2.l + 1 \quad (3.9)$$

For example, if $l = 3$, then the $K = 7$, $K \in \{-3, -2, -1, 0, 1, 2, 3\}$. As the $k = 7$, it means that a single pixel will contribute to 7 different binary codes to the histogram. If the neighboring pixels got high contrast, the K binary code may be the same, while the neighboring pixels with contrast lower than l will generate a distribution of binary codes of fuzzy nature [93]. The values in the final histogram will be divided by K .

Weber Local Descriptor (WLD)

WLD [94] is inspired by Weber's Laws and local descriptor. Weber law states that the increment threshold to the background intensity is constant, which is defined as:

$$\frac{\Delta I}{I} = C \quad (3.10)$$

where I shows the initial intensity, ΔI shows the increment threshold and C is a constant. The left side proportion will be constant regardless the change in I term.

WLD consists of two components. The first is differential excitation (D) which is the arctan of ratio between two terms: 1) the sum of relative intensity differences and 2) the current pixel intensity. Second is the orientation (O), which shows the pixel

direction.

$D(p_c)$ can be calculated for pixel p_c as follow:

$$D(p_c) = \arctan \left[\frac{k_s^{00}}{k_s^{01}} \right] \quad (3.11)$$

$$k_s^{00} = \sum_{i=0}^{P-1} (\Delta p_i) = \sum_{i=0}^{P-1} (p_i - p_c) \quad (3.12)$$

where N is total number of neighbor pixels, and p_i denote the i^{th} neighboring pixel. The differences between the pixel p_c with its neighbor pixels are calculated using filter f_{00} as illustrated in Figure 3.6. Using filters f_{00} and f_{01} , the proportion of difference of current pixel p_c is computed as follows:

$$k' = \frac{k_s^{00}}{k_s^{01}} = \sum_{i=0}^{P-1} \left(\frac{p_i - p_c}{p_c} \right) \quad (3.13)$$

and the orientation O is calculated as given by:

$$O(p_c) = \arctan \left[\frac{k_s^{11}}{k_s^{10}} \right] \quad (3.14)$$

After the calculation of differential excitation D and orientation O , WLD histogram will be formed. The histogram is based on three parameters which can affect the optimization. These parameters include: the number of bins in a histogram, differential excitation and dominant orientation.

f_{00}	f_{01}	f_{10}	f_{11}																																				
<table><tr><td>+1</td><td>+1</td><td>+1</td></tr><tr><td>+1</td><td>-8</td><td>+1</td></tr><tr><td>+1</td><td>+1</td><td>+1</td></tr></table>	+1	+1	+1	+1	-8	+1	+1	+1	+1	<table><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>+1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	+1	0	0	0	0	<table><tr><td></td><td>+1</td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td>+1</td><td></td></tr></table>		+1						+1		<table><tr><td></td><td></td><td></td></tr><tr><td>+1</td><td></td><td>-1</td></tr><tr><td></td><td></td><td></td></tr></table>				+1		-1			
+1	+1	+1																																					
+1	-8	+1																																					
+1	+1	+1																																					
0	0	0																																					
0	+1	0																																					
0	0	0																																					
	+1																																						
	+1																																						
+1		-1																																					

Figure 3.6: WLD filters

Local Phase Quantization

LPQ is a texture descriptor proposed by Ojansivu and Heikkila [95] which uses blur invariance property using Fourier Phase Spectrum. In digital image processing, spatial blurring is the convolution of image intensities and Point Spread Function (PSF). The spatial shift invariance blurring of an image $f(x, y)$ will give the resultant image $f'(x, y)$ via convolution which can be written as:

$$f'(x, y) = f(x, y) * h(x, y) + n(x, y) \quad (3.15)$$

In this equation, $f'(x, y)$ is the resultant image, $f(x, y)$ is an input image, $h(x, y)$ is the PSF, the $*$ represents the convolution and $n(x, y)$ shows the additive noise.

In LPQ, few assumptions will be made, first, the additive noise will be ignored and second, PSF will be symmetric. With this definition, the resultant function can be written as:

$$f'(x, y) = f(x, y) * h(x, y) \quad (3.16)$$

If we transform this equation to the frequency domain, this will correspond to:

$$F'(x, y) = F(x, y) * H(x, y) \quad (3.17)$$

where $F(x, y)$, $F'(x, y)$ and $H(x, y)$ are the Discrete Fourier Transforms (DFTs) of output image, input image and PSF. This equation can be divided into phase and magnitude parts as follows:

$$|F'(x, y)| = |F(x, y)| \times |H(x, y)| \quad (3.18)$$

$$\theta_{f'}(u, v) = \theta_f(u, v) + \theta_h(u, v) \quad (3.19)$$

As the assumption is made that PSF is centrally symmetric $h(x, y) = h(-x, -y)$, its Fourier transform will be real valued and its phase will be two valued function.

$$\theta_{f'}(u, v) = \begin{cases} 0 & H(u, v) \geq 0 \\ \pi & H(u, v) < 0 \end{cases} \quad (3.20)$$

$$\theta_{f'}(u, v) = \begin{cases} \theta_f(u, v) & H(u, v) \geq 0 \\ \theta_f(u, v) + \pi & H(u, v) < 0 \end{cases} \quad (3.21)$$

When $H(u, v)$ is greater than 0, the phase of the observed image $F(u, v)$ at those frequencies will be invariant to centrally symmetric blur [96].

LPQ descriptor uses blur invariance property of the Fourier spectrum to extract features. It uses phase information which is extracted through 2D discrete Fourier transform (DFT) from the rectangular neighbor pixels of an image window $N_{(x,y)}$. The 2D DFT on pixel (x, y) , denoted as $F_{(x,y)}(u, v)$, is defined as

$$F_{(x,y)}(u, v) = \sum_{(x',y') \in N_{x,y}} f(x - x', y - y') e^{-(j2\pi ux' + j2\pi vy')} = W_{(u,v)}^T f_{(x,y)} \quad (3.22)$$

where $W_{(u,v)}$ is the basis vector at frequency (u, v) and $f_{(x,y)}$ is also a vector of M^2 blocks of images from $N_{(x,y)}$. Only four frequency points are considered in LPQ which are, $u_1 = [0, a]^T$, $u_2 = [a, 0]^T$, $u_3 = [a, a]^T$ and $u_4 = [a, -a]^T$, where a is a scalar frequency where the phase of DFT will be invariant to centrally symmetric blur [95] or satisfy $H(u, v) \geq 0$ [96]. The Fourier coefficients of phase information are examined via their signs for real and imaginary parts in $F_{(x,y)}$ using

$$\alpha_i = \begin{cases} 1 & v_i \geq 0 \\ 0 & v_i < 0 \end{cases} \quad (3.23)$$

where v_i is the i^{th} component of $F'_{(x,y)} = [Real(F_x), Imaginary(F_y)]$. The resultant coefficients α_i are represented as integer between 0-255 using binary code.

$$f_{LPQ}(x, y) = \sum_{i=1}^8 \alpha_i \cdot 2^{i-1} \quad (3.24)$$

which forms the basis for a histogram and a feature vector of 256 dimensions.

3.1.3 Using DCT on Transformed Images

In this approach, we apply 2D-DCT on the transformed block, the DCT coefficients are stacked, and standard deviations are computed to get the feature vector. This

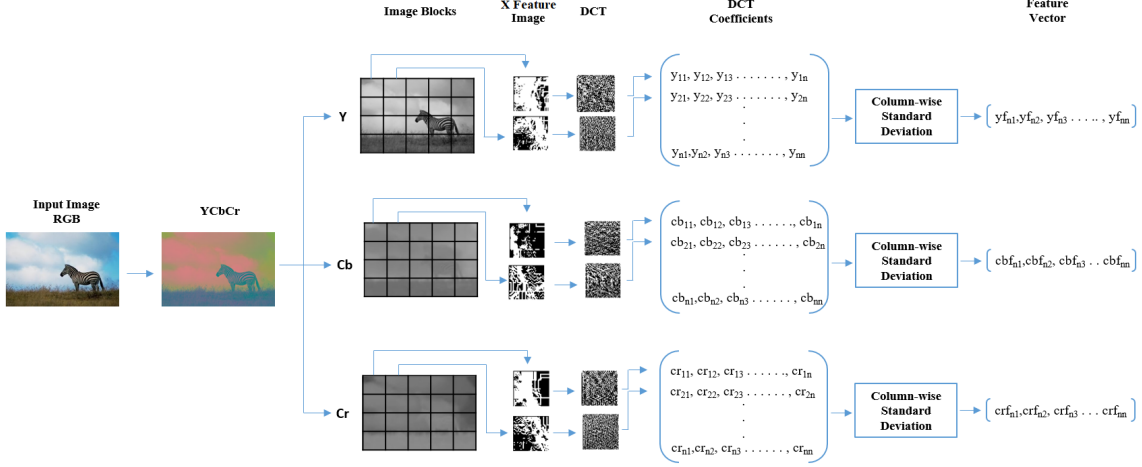


Figure 3.7: Flow diagram of feature extraction using DCT block stacking

approach is illustrated in Figure 3.7. In experiments, we only used LBP, CSLBP, WLD, and LPQ transformation methods. DCT is widely used in signal processing to represent the data via summation of variable frequency cosine waves. As images are two dimensional signals for this purpose we used 2D-DCT. The definition of 2D-DCT is as follows [97]:

$$F'_{uv} = \alpha_u \alpha_v \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_{mn} \cos \frac{\pi(2m+1)u}{2M} \cos \frac{\pi(2n+1)v}{2N}$$

$$0 \leq u \leq M-1, 0 \leq v \leq N-1$$

where f is an input image, F' is output image, and M and N are the row and column sizes of the image f ,

$$\alpha_u = \begin{cases} \frac{1}{\sqrt{M}} & u = 0 \\ \sqrt{\frac{2}{M}} & 1 \leq u \leq M-1 \end{cases}$$

and

$$\alpha_v = \begin{cases} \frac{1}{\sqrt{N}} & v = 0 \\ \sqrt{\frac{2}{N}} & 1 \leq v \leq N - 1 \end{cases}$$

3.1.4 Multi-Scale Feature Extraction

In a typical texture feature extraction, a single window with a number of neighbor pixels was used to extract features. These features were limited only to specific window and pixels. To capture more features we need to increase or decrease the number of neighboring pixels and increase or decrease the window size which is also referred to as a radius. Figure 3.8 illustrates the different scales/radii with a fixed number of pixels. The features will be extracted using these different scales and will be fused together to make a Multi-Scale feature vector.

For example, in Multi-Scale LBP each pixel has a number of LBP codes where P and R vary. The distribution of these codes is used as a descriptor. In this work,

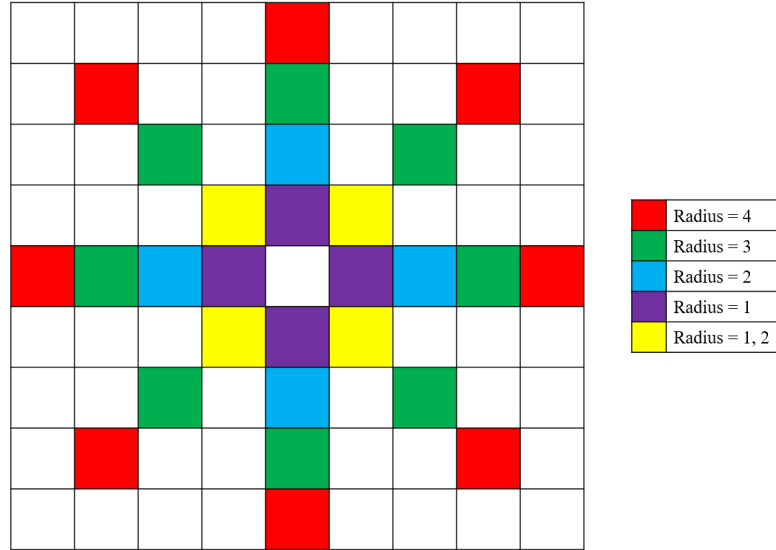


Figure 3.8: A window of 9×9 shows the radii $R \in \{1, 2, 3, 4\}$ and neighbors $P = 8$

we used three scales, where $P = [8, 12, 16]$ and radius $R = [1, 2, 3, 4]$. Similar work was also adopted in [98], [99] as illustrated in Figure 3.9. Similar approach was also adopted for LPQ features where we used windows of 3×3 and 5×5 to extract the features and fuse them.

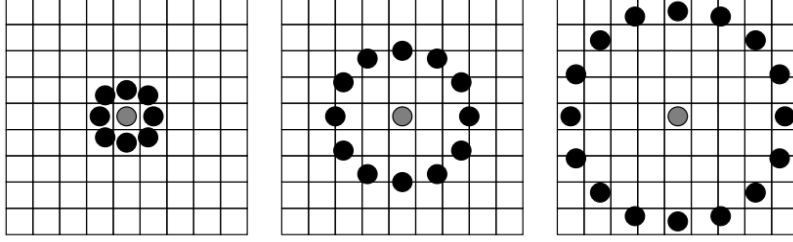


Figure 3.9: LBP circular symmetric neighborhood with $(R=4, P=8)$, $(R=2, P=12)$ and $(R=4, P=16)$ [2]

3.1.5 Feature Reduction

The model complexity and performance are affected by the number of features used. Since some of the previously explained transformation based methods for feature extraction can produce a large number of features, we explored the impact of applying feature reduction by using PCA and LLP.

Principal Components Analysis (PCA)

PCA is an unsupervised method used for feature dimensionality reduction. If the input space of the feature dimension is D then the number of chosen dimensions K has to satisfy $K < D$. The Euclidean distance between the K points should be closely approximated to the Euclidean distance between the points in the D dimensions. This means that the variance of the points should be preserved as close as possible. Before selecting the K dimensions, the affine transformation needs to be applied to de-correlate

the feature axes. To summarize PCA, it has two main steps, First, normalize the data using zero mean and de-correlate the axes. Second, choose K dimension with the highest variance. The PCA is implemented using the following steps [100]:

Step 1: Get the data X

Step 2: Normalize the data X by subtracting the mean

Step 3: Calculate the covariance matrix

Step 4: Calculate the eigenvalues and eigenvectors of the covariance matrix

Step 5: Select K eigenvalues to make a feature vector

In this work, we choose K based on variance V , where $V \geq 99\%$.

Local Learning Projection (LLP)

LLP was proposed in [101], where a complex non-linear problem is divided into small locally linear problems using local learning. The method is based on numerical analysis and machine learning techniques without making any assumption about data distribution. The algorithm selects the most significant and discriminatory features from the large pool to enhance the learning rate.

3.1.6 Classification

SVM is a supervised learning algorithm used for regression and classification problems. The forgery detection is a binary classification (Authentic or Forged) problem. SVM performs better on two-class problems by using various kernel functions to map the samples into a higher dimensional space where classes can be easily separable. In this work, we performed our experiments using radial-basis function (RBF), Polynomial

and Linear kernels, which were chosen during the pre-experimentation step based on better performance than other SVM kernels as explained in Section 2.2.1. SVM finds an optimal hyperplane with the maximum margin between the two classes [102]. To predict the positive (forged) and negative (authentic) classes, SVM uses the signed distance of a sample point from the decision boundary. The positive score will classify the sample point as positive, otherwise, it will be classified as negative [103].

3.2 Deep Learning Approach

A classic neural network is made of input, output and at least one middle layer referred as hidden layer. These layers are composed of neurons and each neuron is connected to their back and forward layers via weighted connections, referred as network weights. Each neuron calculates a weighted sum of its inputs and bias and applies the same linear or non-linear activation function to this summation.

$$y_j = f(\sum_i x_i w_{ij} + b_j)$$

The evaluation of this network is estimated with loss function. This loss function should be minimized during the training of a network with respect to the weighted connection. The training consists of an iterative process called forward and backward propagation.

The number of layers excluding the input layer in a network is called network depth, i.e., a network with one input, one output and a hidden layer is two layers

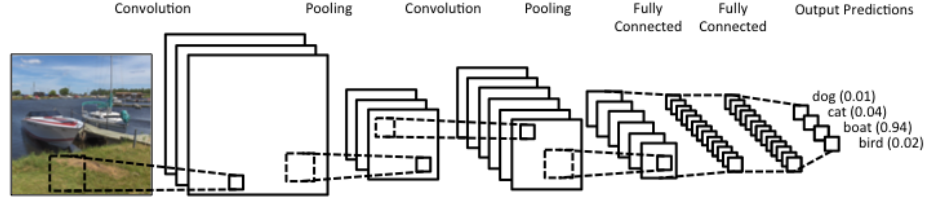


Figure 3.10: Convolutional neural network

deeper and has a depth of two. The number of neurons is proportional to the width of a layer.

A generic representation of a deep neural network, in particular, a CNN architecture which is widely used in research community due to its efficiency is shown in Figure 3.10. It illustrates that input layer stacked with convolutional layers followed by fully connected layers. The convolutional layers generate feature maps. The sub-sampling is used to reduce the feature dimensionality via pooling layers. At the end, these feature maps are fed to the fully connected layers to compute the weighted sum and apply non-linear transformation. Figure 3.11 shows the proposed deep learning model which consists of ten convolutional layers and a fully connected layer.

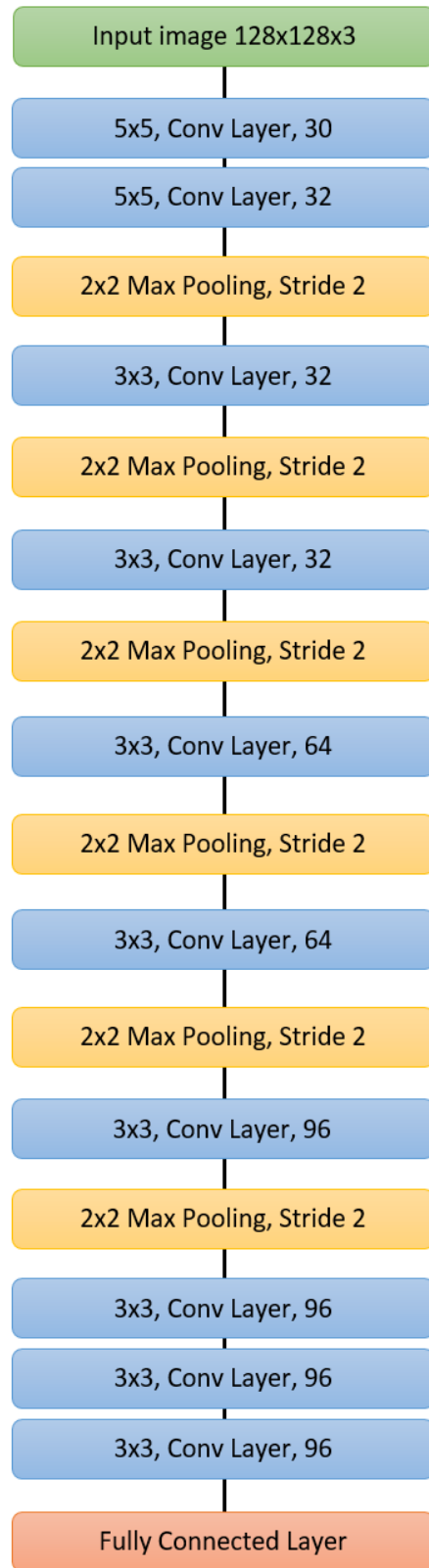


Figure 3.11: Proposed convolutional neural network model

3.2.1 Components of CNN

CNN consists of various layers which are discussed as follow:

Input Layer

The input layer is the first layer where the images will be loaded. These images can be gray or RGB images with 3 color channels.

Convolutional Layers

A convolutional layer is one of the important parts of CNN and is used in many computer vision problems. These are the layers following the input layer. A convolutional layer is composed of a various number of two dimensional filters. These filters are not like traditional filters of machine learning paradigms, but these filters are used in CNN to learn feature from image during training.

These filters/kernels convolve over the images and pass the resulting response, also known as feature maps/activation maps, to the next layer [104]. There may be more than one convolutional layer in a CNN. AlexNet [13] is a network with five convolutional layers. With this concept there is no restriction on how many convolutional layers can be used in a network. Recently proposed ResNet [17] model consists of 152 layers with better performance.

As the CNN consists of multiple convolutional layers, it extracts low level and high level (objects i.e., nose, eyes) features from the input image. High level features are weighted features combined with low level features. Similarly, low level features (edges, texture) can generate high level features (objects), with the combination of

these features it may represent more complex patterns in an image. Recently, some papers were focused on visualization of low and high level features which are generated via neural networks [14], [105], [106].

The convolutional layer filters are small in size (3×3 - 11×11) and have the same depth as the image depth) because it can detect the strong correlation of closed neighbor pixels of an image. The size of these filters are called receptive fields which can be tuned by the user. Each filter/receptive field has an array of numbers called weights (When these filters convolve over an image, the weights multiply with image pixels and summed up to one value). The size of receptive fields restraint the neurons in the next layer which are connected to the neurons of the current layer. During the convolution step, the filter jumps to next positions to perform convolution, The step-size is called stride.

Pooling Layers

The pooling layers are mostly used after convolutional layers. Their specified parameters are size, type and stride. This layer is used for down-sampling the feature dimensionality and complexity of a network and is also used to control over-fitting. The most used common size of pooling is 2×2 with stride 2 with MAX operation, this reduces the activation size by 75% as shown in Figure 3.12. An average and l2-norm operation can also be used in pooling layers. However, the paper [107] proposed down-sampling with convolutional layers to discard pooling layer. To tune the pooling layer, it can be used as overlapping (the pooling stride size should be less than pooling size) and non-overlapping (the pooling stride should be equal or greater

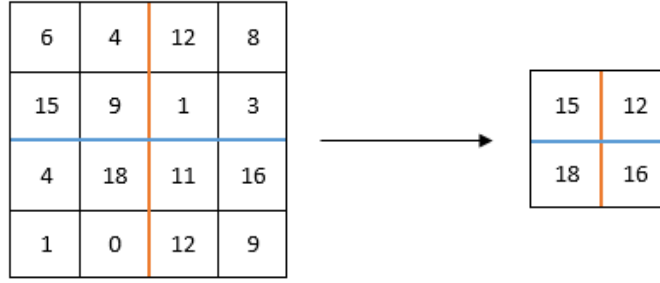


Figure 3.12: Maximum pooling

than pooling size).

ReLU Layers

Rectified linear unit introduced in [108] is mostly used in deep learning architecture and more popular than logistic sigmoid functions. The purpose of this layer is thresholding, i.e, to truncate all negative values and the non-linearity properties without effecting the receptive fields.

$$f(x) = \max(0, x)$$

Dropout Layers

Hinton et al. [109] introduced dropout layers in 2012. Since then, Dropout layers have become a standard layer for deep learning architectures. It supports other layers like convolutional and fully connected layers and can be placed anywhere in a network. The layer is associated with dropout probability which specifies the frequency of neurons to turn on/off of a particular functional layer (Convolutional or fully connected layer). For instance, if the dropout probability is 0.5 this will randomly off half of the neurons which will not take part in forward and backpropagation. The dropout layers work as

regularizer for deep networks and decrease variance and over-fitting.

Fully Connected Layers

Fully connected layers are also called dense layers or inner product layers. The neurons in this layer are connected to all the elements of its preceding layer which leads to a very dense pattern. In contrast of convolutional layer where each neuron of current layer is connected to the preceding layer which lies in its receptive field. One of the important features of convolutional layers is that it significantly decreases the number of parameters in contrast to fully connected layers. Classification layer is the last layer in the CNN which classify the input values into different number of classes. The number of classification layers depends on the output classes.

3.2.2 Weights Initialization

Weights initialization is a crucial step in designing a CNN which will lead to learn discriminatory features. If the weights are initialized to zero, all the neurons will do the same thing and will learn the same features. Due to this reason, all the neurons will have the same gradients and will update at the same way. However, neurons should learn different features, that is why weights initialization to zero is not recommended. One of the ways to initialize the weights is to choose random values from a distribution which can be a standard Gaussian distribution. If the weights are too small they will collapse and if the weights are too large they will be saturated in deeper network models. To overcome this problem, the Xavier weights [110] were introduced where the input and output neurons have weights with the same variance. In this work we

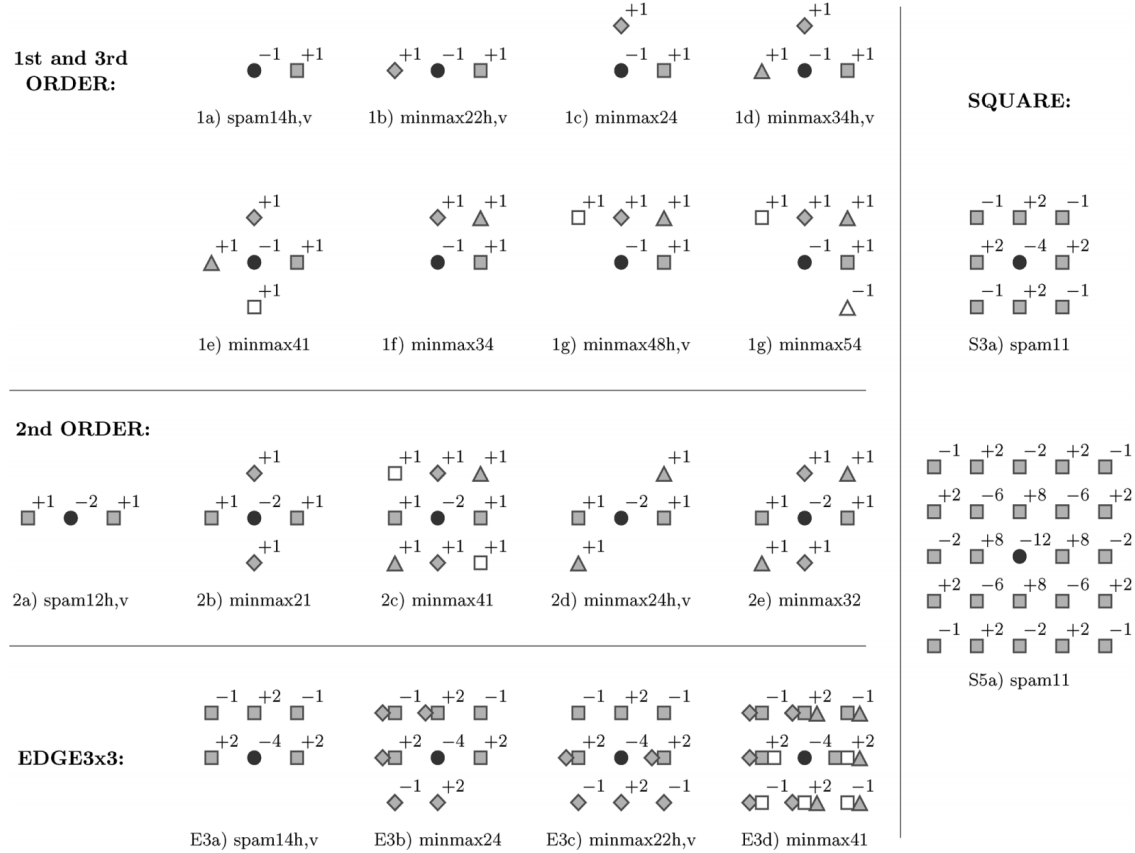


Figure 3.13: SRM model weights

used SRM weights as shown in Figure 3.13 for the first convolutional layer, while the rest of the layers were initialized with random Gaussian weights.

3.2.3 Data Preparation for CNN

Pre-processing steps

This is the first step to prepare our data for CNN. First, we need to extract the boundary of the spliced regions. For this purpose, we find the original image of each spliced image as shown in Figure 3.14 and find the difference between these two images. This will give us the spliced or copy-moved region in an image. We used Prewitt operator to extract the edges of spliced region. To remove the noise and extract

the connected pixels we normalized the image with the morphological operation to segment the regions more accurately as shown in Figure 3.15a. The mask was then applied on the spliced image to extract the tampered region. Figure 3.15b highlights it using red boundary.

Patches Extraction

In literature, few articles [71], [73], [72] adopted this approach to divide the images into patches and then apply the CNN. Similarly, to extract the patches from forged images we extract the mask as discussed above, and draw patches around that mask as

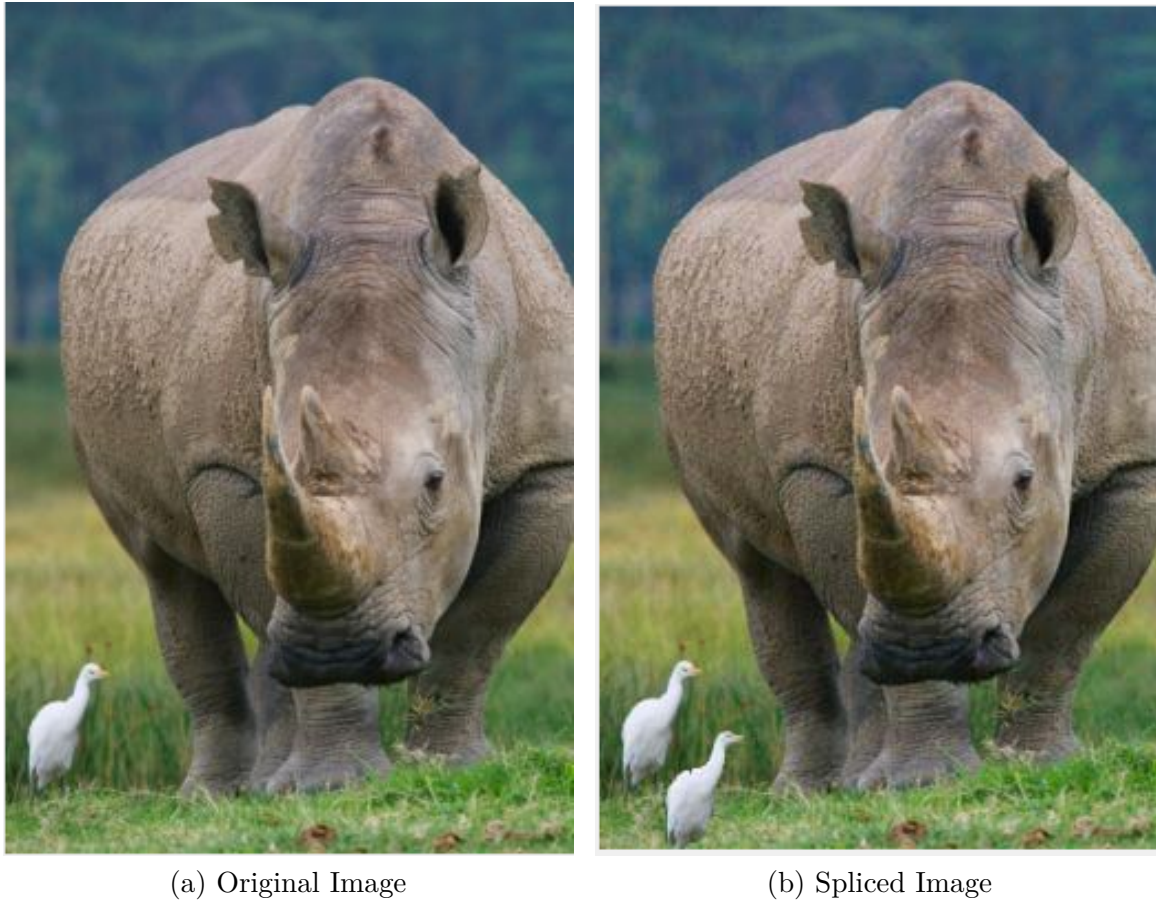


Figure 3.14: Authentic and forged images from CAISA v1.0 dataset

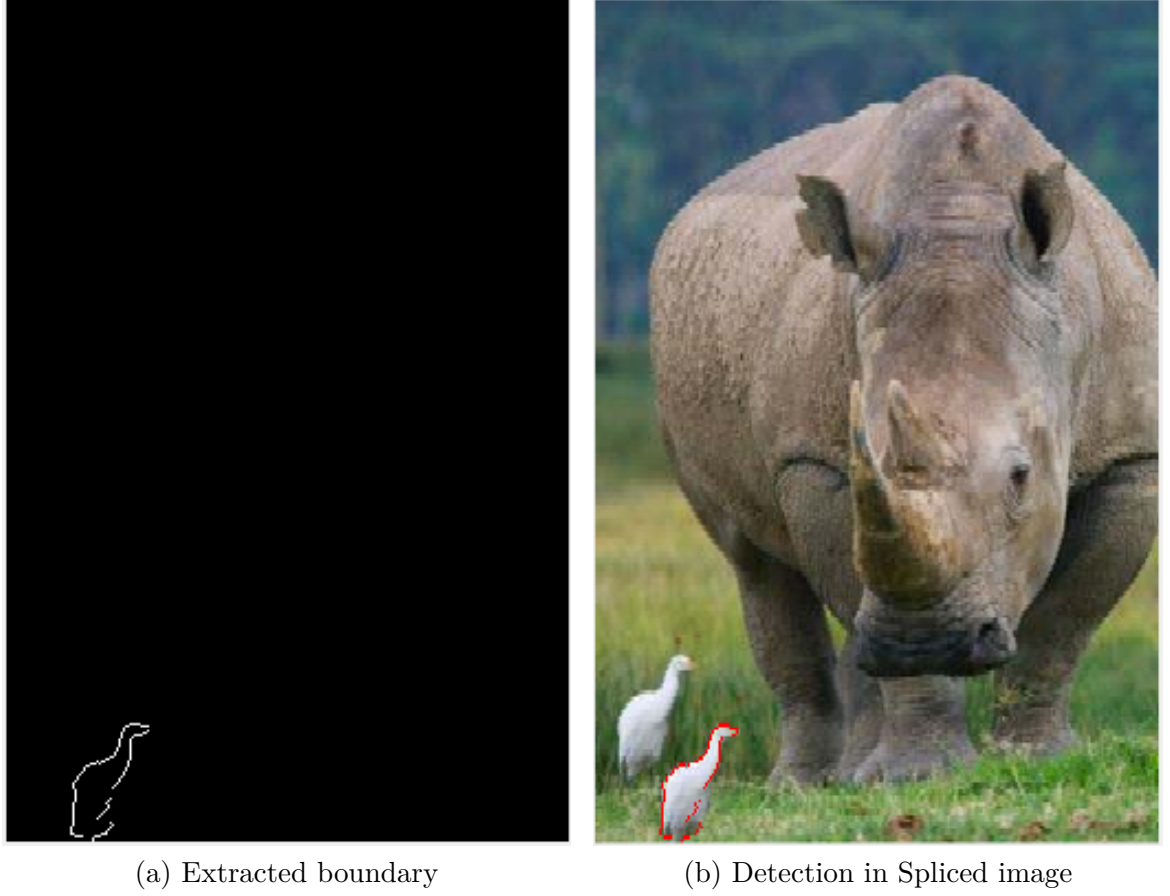


Figure 3.15: Extraction of forged region from image

shown in Figure 3.17. The patches that lie outside the image boundaries are avoided. Figure 3.16 shows different patches extracted on the boundary of forged area while Figure 3.16b shows the different sizes of patches. We extract also a similar number of patches from authentic images exactly at the same point where the spliced image patch was extracted. Three different sizes of patches 128×128 , 64×64 and 32×32 were extracted. Then, we applied the CNN with various layers. We used CASIA v1.0 dataset which consists of 800 authentic and 921 forged images.

The patches size of 128×128 and the number of 6211 and 6447 were extracted from forged and authentic images respectively. Similarly, for patch size of 64×64 ,

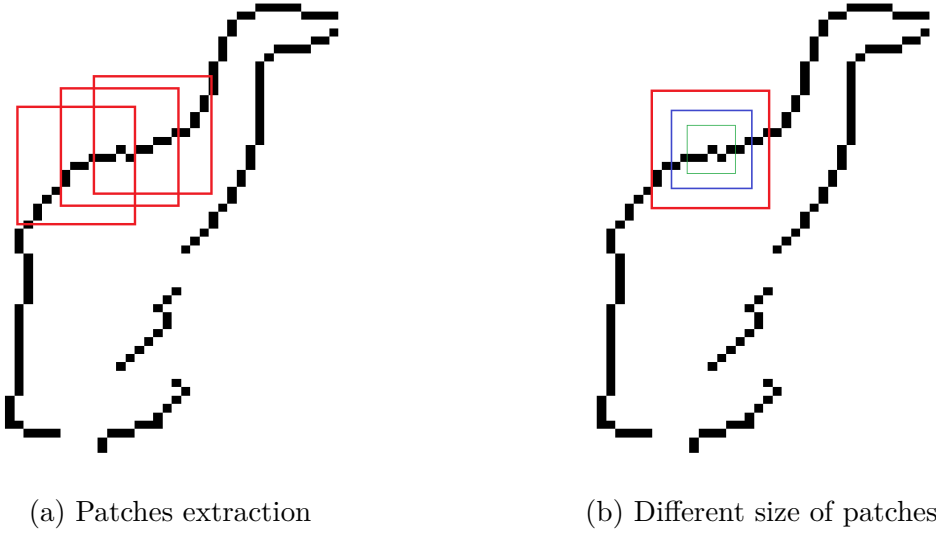


Figure 3.16: Patch extraction visualization

15487 from forged and 15657 patches from authentic images were extracted. The 23339 number of patches from forged and 23946 patches from authentic images were extracted for patch size of 32×32 .

Data Augmentation

Most computer vision problems need huge data to train the classifier. To generate a large amount of data using the original data, one of the widely used techniques is called Data Augmentation. The common augmentation methods is mirroring of an image, where an original image is flipped horizontally, without changing the context of an image as illustrated in Figure 3.18. For example, an image of a cat will be still a cat if we flipped it horizontally but it will not be the case in using the image of a text signboard. Another method is random cropping where a single image is cropped with various regions including the region of interest. Rotation, shearing and

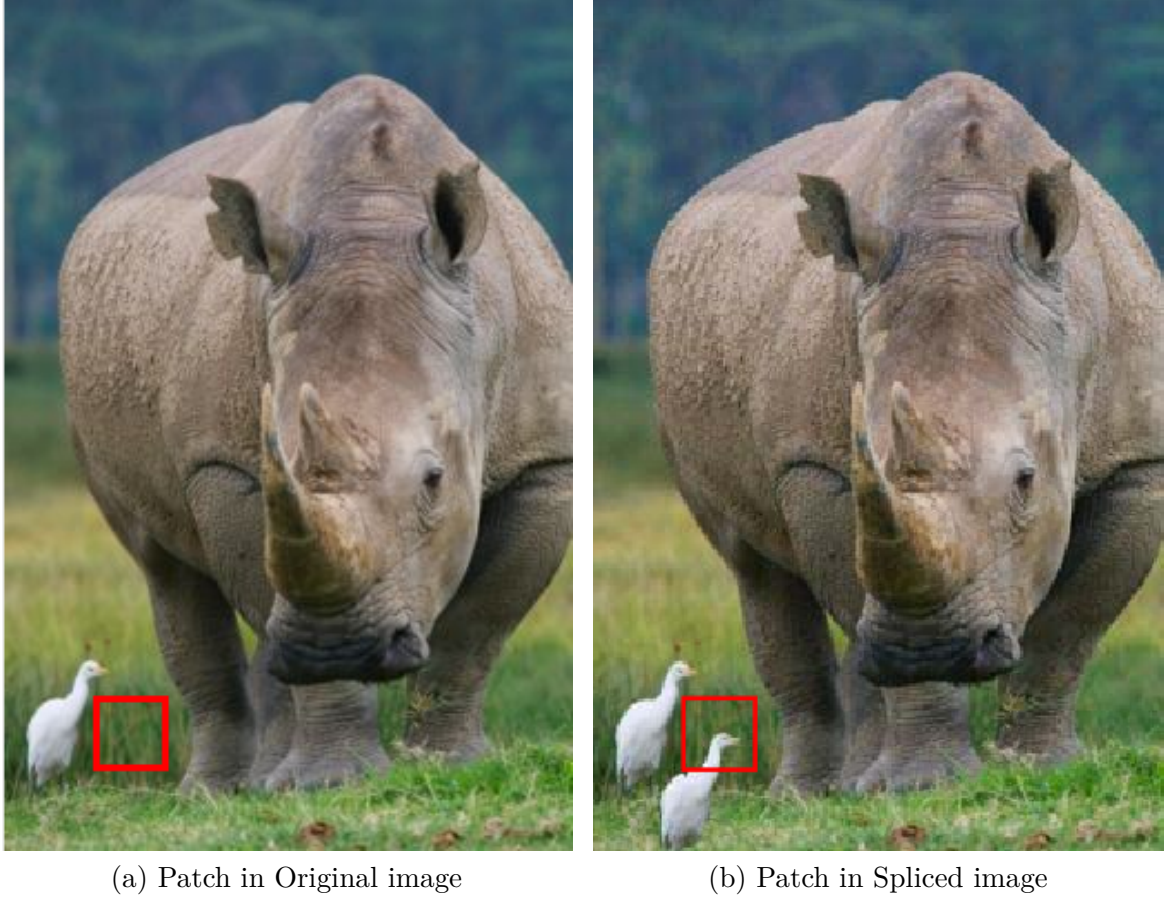


Figure 3.17: Patch extractions from authentic and forged image

local warping are also methods of data augmentation [111]. Color shifting is also an effective method, where distortion will be added to the RGB channels of a color image. The distortion can be the addition or subtraction of some values in colors channels. Due to this the image illumination will be affected but the identity of the object will remain the same and this will make the learning algorithm more robust to in/out-door illumination.

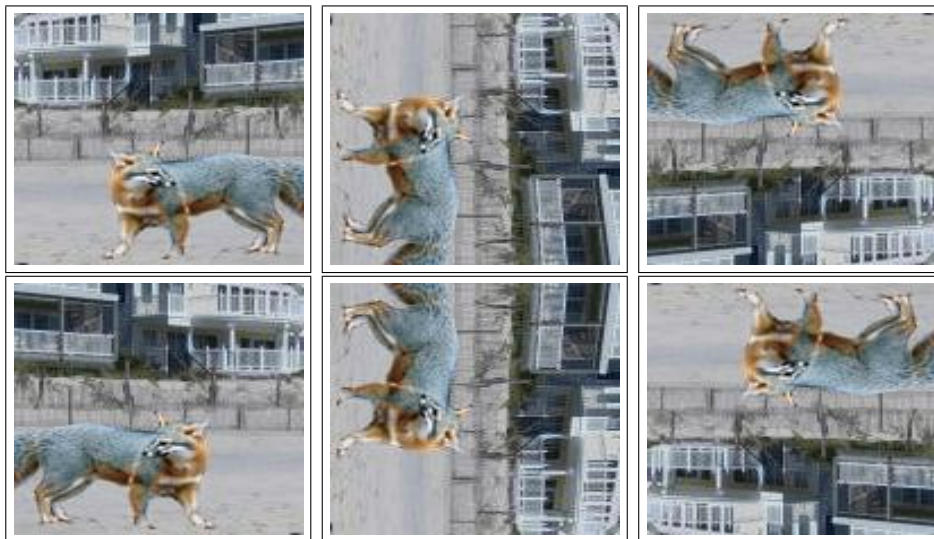


Figure 3.18: The first rows shows a sample of flipped images and the second row shows a sample of mirror images

CHAPTER 4

RESULTS AND DISCUSSIONS

This chapter presents the details of the conducted experiments and discussions of the results for both Feature Extraction (FE) based and CNN-based approaches.

4.1 Benchmark Datasets

The experiments were performed on two publicly available datasets: CASIA v1.0 and CASIA v2.0; their details are shown in Table 4.1. Figure 4.1 and Figure 4.2 show few samples of authentic and forged images in these datasets. Both datasets contain color images.

CASIA v1.0: This dataset consists of 800 authentic and 921 spliced images in two different sizes 384×256 and 256×384 in JPEG format. The images include various categories of natural and textural scenes like plants, animals and architectures.

Dataset	Num. of Images		Image Type	Image Size
	Authentic	Tampered		
CASIA v1.0	800	921	JPEG	384×256 256×384
CASIA v2.0	7491	5123	JPEG, TIF, BMP	240×260 to 900×600

Table 4.1: Datasets details



Figure 4.1: First row shows three authentic and second row shows three forged images of CASIA v1.0

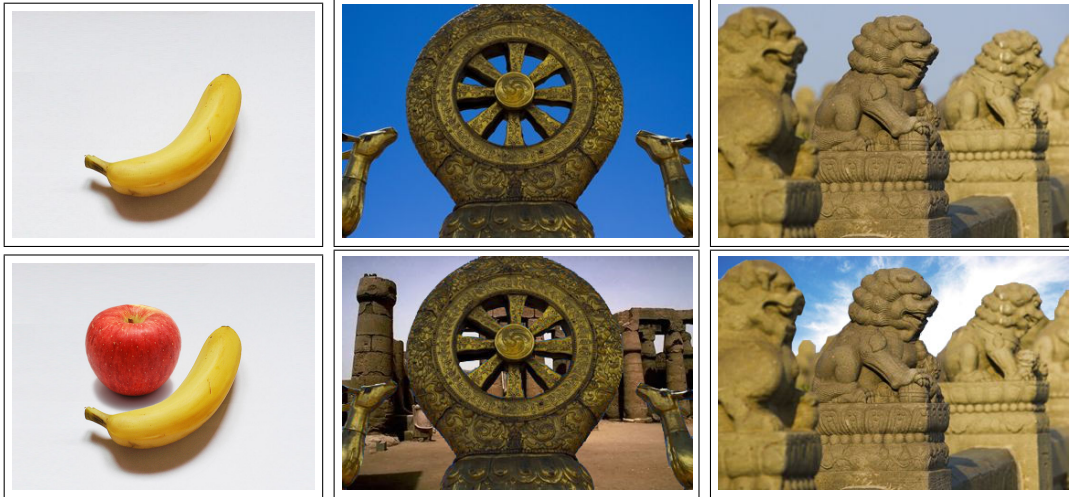


Figure 4.2: First row shows three authentic and second row shows three forged images of CASIA v2.0

CASIA v2.0: This dataset contains 7491 authentic and 5123 forged images with sizes varying from 240×260 to 900×600 . This dataset includes indoor categories as well which affects the image illumination. There are three image formats: JPEG, BMP and TIFF.

4.2 Experimental Settings

For feature extraction methods, the two datasets CASIA v1.0 and CASIA v2.0 were used. Before feature extraction, all images were converted to YCbCr format and divided into non-overlapping blocks of three different sizes: 16×16 , 32×32 , and 64×64 .

For FE-based classification, we used SVM with two different types of Kernels: Polynomial for CASIA v1.0 and RBF for CASIA v2.0. The selection of a kernel was based on some preliminary experiments for the two datasets to select the best kernel method. We used Matlab 2017b to develop all our models. Since we ran many experiments which took much time, we used two different systems. For CASIA v1.0, the experiments were conducted on Intel Xeon with 2.4 GHz processor and 32 GB RAM. For CAISA v2.0, we used Intel Core i7 2.8 GHz processor and 32 GB RAM.

For deep learning experiments, we used Core i7 with 8 GB of RAM and 4 GB Nvidia Graphics card for deep learning. As explained in Section 3.2.3, The image patches were extracted from the original and forged images. Each patch size is $128 \times 128 \times 3$ where 3 denotes the color channels.

4.3 Evaluation Metrics

In this study, SVM classifier is evaluated using 10-fold cross validation. The performance measures include precision, sensitivity/recall, specificity, F_1 score, accuracy, true negative rate (TNR), true positive rate (TPR), false negative rate (FNR), false positive rate (FPR) and area under the curve (AUC). The definitions are as follow:

$$FPR = \frac{FP}{FP + TN} \quad (4.1)$$

$$FNR = \frac{FN}{FN + FP} \quad (4.2)$$

$$Precision = \frac{TP}{TP + FP} \quad (4.3)$$

$$Sensitivity = Recall = TPR = \frac{TP}{TP + FN} \quad (4.4)$$

$$Specificity = TNR = \frac{TN}{TN + FP} \quad (4.5)$$

$$F_1 = \frac{2TP}{2TP + FP + FN} \quad (4.6)$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (4.7)$$

where FP , FN , TP and TN are the False Positive, False Negative, True Positive, and True Negative, respectively.

For deep learning, the dataset is randomly split into 75% for training and 25% for validation and the classification accuracy is reported for validation during training.

4.4 Feature-Extraction Based Experiments and Results

The experiments were conducted using various image blocks to capture the detailed features. On each color channel Y, Cb, and Cr, the image was divided into 16×16 , 32×32 and 64×64 blocks and features were extracted for each block. To combine these features, we followed two approaches. In the first approach, the image features were concatenated. This approach was applied only on CASIA v1.0 dataset. The reason is that all the images in this dataset have the same number of blocks. However, CASIA v2.0 images have varying sizes from 240×260 to 900×600 which affect the number of blocks per image and this leads to having feature vectors of inconsistent sizes. In this case, features extracted from the image blocks were stacked together and the standard deviation was calculated column-wise and was used as a feature vector. This approach was adopted for both CASIA v1.0 and CASIA v2.0 datasets.

Table 4.2 shows a summery of the experiments on CASIA v1.0 in descending order of the accuracy. These experiments were conducted for 17 variants of feature extraction methods as explained in the Chapter 3 methodology. The detailed results for each method is given in Appendix A, as indicated in the second column of Table 4.2. Here, Tables A.1 to A.6 show the experiments of feature concatenation while the rest of the tables shows the results of feature stacking experiments. Comparatively, although concatenation-based models have more features than stacking-based models, there performances are low in terms of accuracy and AUC.

The first row, shows the highest accuracy of 97.33% with AUC 0.994 using MSLBP-

DCT method with 1, 2, 3 radii and 8 neighbor pixels ($r=1, 2, 3$ and $p=8$). This accuracy is achieved with only 2360 features using Cr channel with block size 32×32 . In the second row, MSLPQ-DCT method with a window of 3 and 5 ($w=3, 5$) achieved the second highest accuracy of 96.93% with AUC 0.990. It used only 340 features with Cr channel and 16×16 block size. All the top three methods used multi-scales with Cr channels which performed better than single scale feature extractors.

Table 4.3 shows CASIA v2.0 experiments which are based on only feature stacking. The first row shows the highest accuracy of 97.30% with AUC 0.992 using MSLPQ with window 3 and 5 ($w=3, 5$). This experiment used Cb channel and a block size of 16×16 with only 512 features. The second highest method is also MSLPQ with window 3 and 5 ($w=3, 5$) but with DCT coefficients which achieved 96.85% accuracy with 1684 features. The third row shows the MSLBP-DCT method with radius 1, 2 and neighbors of 8 and 12 ($r=1, 2$ and $p=8, 12$) respectively. The top three methods indicate that multi-scale feature extractors perform better on CASIA v2.0 dataset.

For both the datasets, the top three accuracies were achieved using multi-scale features.

S.No	Table Ref	Methods	Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
1	A.8	MSLBP-DCT (1,2,3)	Cr	32×32	0.0319	0.0195	0.9623	0.9805	0.9681	0.9712	0.9734	0.9947	2360	
2	A.14	MSLPQ-DCT (3,5)	Cr	16×16	0.0349	0.0250	0.9585	0.9750	0.9651	0.9666	0.9694	0.9909	340	
3	A.12	MSLBP-DCT (8,12)	Cr	16×16	0.0361	0.0259	0.9573	0.9741	0.9639	0.9655	0.9688	0.9908	296	
4	A.9	LBP-DCT (1)	Cr	16×16	0.0419	0.0288	0.9498	0.9712	0.9581	0.9601	0.9636	0.9868	196	
5	A.15	MSLPQ (3,5)	Cr	16×16	0.0594	0.0385	0.9297	0.9615	0.9406	0.9448	0.9501	0.9877	512	
6	A.7	CSLBP-DCT	Cr	16×16	0.0553	0.0482	0.9352	0.9518	0.9447	0.9431	0.9485	0.9791	225	
7	A.10	MSLBP(1,2,3,4)	Cb	16×16	0.0711	0.0595	0.9154	0.9405	0.9289	0.9274	0.9346	0.9809	1024	
8	A.13	MSLBP (8,12)	Cb	16×16	0.0798	0.0527	0.9000	0.9473	0.9202	0.9224	0.9323	0.9792	4352	
9	A.17	WLD-DCT	Cr	16×16	0.0811	0.0863	0.9047	0.9137	0.9189	0.9086	0.9160	0.9488	225	
10	A.16	SLBP+DCT	Cr	32×32	0.1046	0.0937	0.8744	0.9063	0.8954	0.8897	0.9005	0.9535	1280	
11	A.11	LBP (3)	Cr	16×16	0.1059	0.0982	0.8732	0.9018	0.8941	0.8869	0.8976	0.9477	256	
12	A.1	CSLBP+DCT	Cr	64×64	0.1244	0.1139	0.8505	0.8861	0.8756	0.8669	0.8803	0.9468	4480	
13	A.4	MSLPQ (3,5)	Cr	64×64	0.1588	0.1181	0.8002	0.8819	0.8412	0.8385	0.8587	0.9437	12288	
14	A.2	SLBP+DCT	Cr	64×64	0.1613	0.1589	0.8095	0.8411	0.8387	0.8241	0.8397	0.9224	10240	
15	A.5	MSLBP(1,2,3)	Cr	64×64	0.1723	0.1639	0.7879	0.8361	0.8277	0.8099	0.8309	0.9209	18432	
16	A.6	LBP(2)	Cr	64×64	0.1900	0.1777	0.7706	0.8223	0.8100	0.7948	0.8160	0.9094	6144	
17	A.3	WLD	Cr	16×16	0.2761	0.4362	0.7980	0.5638	0.7239	0.6590	0.6177	0.5865	86400	

Table 4.2: Summery of experiments using CASIA v1.0 dataset

S.No	Table Ref	Methods	Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
1	A.28	MSLPQ (3,5)	Cb	16×16	0.0509	0.0093	0.9636	0.9907	0.9491	0.9769	0.9731	0.9927	512	
2	A.27	MSLPQ-DCT (3,5)	Cb	32×32	0.0594	0.0108	0.9574	0.9892	0.9406	0.9731	0.9685	0.9915	1684	
3	A.23	MSLBP-DCT (8,12)	Cb	32×32	0.0594	0.0112	0.9574	0.9888	0.9406	0.9729	0.9683	0.9919	1576	
4	A.19	MSLBP-DCT (1,2,3,4)	Cb	32×32	0.0595	0.0115	0.9574	0.9885	0.9405	0.9727	0.9681	0.9916	3144	
5	A.20	LBP-DCT (1)	Cr	32×32	0.0588	0.0134	0.9581	0.9866	0.9412	0.9721	0.9674	0.9921	900	
6	A.21	MSLBP (1,2)	Cb	32×32	0.0576	0.0151	0.9590	0.9849	0.9424	0.9718	0.9670	0.9910	512	
7	A.22	LBP (1)	Cb	32×32	0.0598	0.0147	0.9573	0.9853	0.9402	0.9711	0.9662	0.9909	256	
8	A.26	WLD-DCT	Cr	64×64	0.0686	0.0114	0.9503	0.9886	0.9314	0.9690	0.9640	0.9899	3969	
9	A.24	MSLBP(8,12)	Cr	64×64	0.0647	0.0172	0.9538	0.9828	0.9353	0.9681	0.9627	0.9908	4352	
10	A.18	CSLBP-DCT	Cb	32×32	0.0640	0.0228	0.9548	0.9772	0.9360	0.9658	0.9599	0.9894	961	
11	A.25	SLBP	Cb	32×32	0.0656	0.0218	0.9536	0.9782	0.9344	0.9658	0.9598	0.9894	256	

Table 4.3: Summery of experiments using CASIA v2.0 dataset

4.5 CNN-Based Experiments and Results

CNN requires a large amount of data for training. The dataset CASIA v1.0 has only 1721 images which are enough for basic CNN model with few convolutional layers but

not enough for deep models with many convolutional layers. To increase the number of images we extracted patches and increased the number through augmentation as discussed in Chapter 3. The experimental work used three different sets of images. The first set of patches consists of 2000 patches with 1000 authentic and 1000 forged patches. The second set has 12422 patches with 6211 authentic and 6211 forged and the third set has 99376 patches with 49688 authentic and 49688 forged patches. All the patches were color of size $128 \times 128 \times 3$.

Initially, the experiments start with a simple model of 4 convolutional layers followed by batch normalization and ReLU layers. Max-Pooling was used after the 2nd and 3rd convolutional layers. The other options were set up to minimum-batch of 256, validation frequency of 30, max-epoch of 30 and initial learning rate of 0.001.

The experiments were performed on the first set of patches. During the training, the learning curve achieved 93% accuracy for training and 60.60% for validation which means the model was over-fitting. To overcome the over-fitting, we need to add more data and a regularization step. First, we added more data and increased the number patches from 2000 to 12422. This time, the validation performance increased to 65.13% accuracy but the training accuracy remain the same. Different epoch sizes from 30 to 250 were used to check the performance but there was no significant increase in accuracy.

The number of convolutional layers was now increased to 8 and the weights were initialized randomly (random Gaussian weights) to all the convolutional layers except the first layer, which was initialized using SRM weights as explained in Chapter 3. The experiments were performed using 2000 patches. The training accuracy reached

S.No.	Layer Name	Layer / Filter Size	Number of Filters	Stride / Padding
1	Input Layer	$128 \times 128 \times 3$	-	-
	Conv Layer	5×5	20	1
	BatchNorm Layer	-	-	-
	ReLU Layer	-	-	-
2	Conv Layer	5×5	32	1
	BatchNorm Layer	-	-	-
	ReLU Layer	-	-	-
	Max-Pooling	2×2	-	2
	Dropout (0.2)	-	-	-
3	Conv Layer	3×3	32	1
	BatchNorm Layer	-	-	-
	ReLU Layer	-	-	-
	Max-Pooling	2×2	-	2
4	Conv Layer	3×3	32	1
	BatchNorm Layer	-	-	-
	ReLU Layer	-	-	-
	Max-Pooling	2×2	-	2
5	Conv Layer	3×3	64	1
	BatchNorm Layer	-	-	-
	ReLU Layer	-	-	-
	Max-Pooling	2×2	-	2
	Dropout (0.3)	-	-	-
6	Conv Layer	3×3	64	1
	BatchNorm Layer	-	-	-
	ReLU Layer	-	-	-
	Max-Pooling	2×2	-	2
7	Conv Layer	3×3	96	1
	BatchNorm Layer	-	-	-
	ReLU Layer	-	-	-
	Max-Pooling	2×2	-	2
8	Conv Layer	3×3	96	1
	BatchNorm Layer	-	-	-
	ReLU Layer	-	-	-
9	Conv Layer	3×3	96	1
	BatchNorm Layer	-	-	-
	ReLU Layer	-	-	-
10	Conv Layer	3×3	96	1
	BatchNorm Layer	-	-	-
	ReLU Layer	-	-	-
	Dropout (0.5)	-	-	-
	Fully Conn Layer	2	-	-

Table 4.4: CNN architecture with all the layers

Options	Values
Weights	SRM for initial weights and random weights for other layers
Training and Test data	75% and 25%
Optimizer	stochastic gradient descent with momentum (SGDM)
Initial Learning Rate	0.001
Validation Frequency	100
Max Epoch	150
Mini Batch Size	32
L2-Regularization	0.0005

Table 4.5: CNN configuration

almost 99% and the validation accuracy reached 86%. The number of patches was increased again to 12422 which achieved 82% validation accuracy and using 99376 patches the validation accuracy decreased to 74%. Now, the regularization step was added to the same 8 convolutional layers model in the form of L2-Regularization and dropout layers. This step improved the performance on a large amount of data (99376 images) and achieved a validation accuracy of 85%.

The number of layers was increased to 10 convolutional layers, the SRM weights were initialized to the first layer, while the rest of the 9 convolutional layers weights were assigned randomly. The regularization in the form of L2-Regularization and dropout was added to the model which increased the performance in terms of accuracy and achieved 91% using 99376 patches with epoch size 150 and learning rate 0.001. The final model illustrated in Table 4.4 with pre-settings in Table 4.5. Further details of the experiments are discussed in Appendix A.

4.6 Statistical Analysis and Feature Reduction

We chose ten methods for both datasets which achieved the highest accuracies for further analysis. We evaluated the statistical significance using Confidence Intervals (CI) with 95% confidence level. To achieve this, the experiments were repeated 20 times and the CI is computed as followed:

$$CI = \bar{x} \pm ME \quad (4.8)$$

$$ME(MarginError) = z_{\alpha/2} \frac{\sigma}{\sqrt{n}}$$

$$\bar{x} = \frac{\sum x}{n}$$

$$\sigma = \sqrt{\frac{(x - \bar{x})^2}{n}}$$

Where $\bar{x}$ represents the mean, σ is standard deviation, n is the total number of experiments, α is confidence level (95%) and $z_{\alpha/2}$ shows the confidence coefficient.

Table 4.6 and Table 4.7 show the statistical analysis of the experiments where the numbers show CI in the form of $\bar{x} \pm ME$. The experiments show on CASIA v1.0 in Table 4.6, where MSLBP-DCT(1,2,3) in Cr channel with block size 32×32 performs better among other methods as illustrated in Figure 4.3. Table 4.7 shows the experiments on CASIA v2.0 where MSLPQ in Cb channel with block size 16×16 performs better as illustrated in Figure 4.4.

More experiments were conducted using feature concatenation of all three channels. To explore the feature fusion/concatenation, we select top five methods which perform

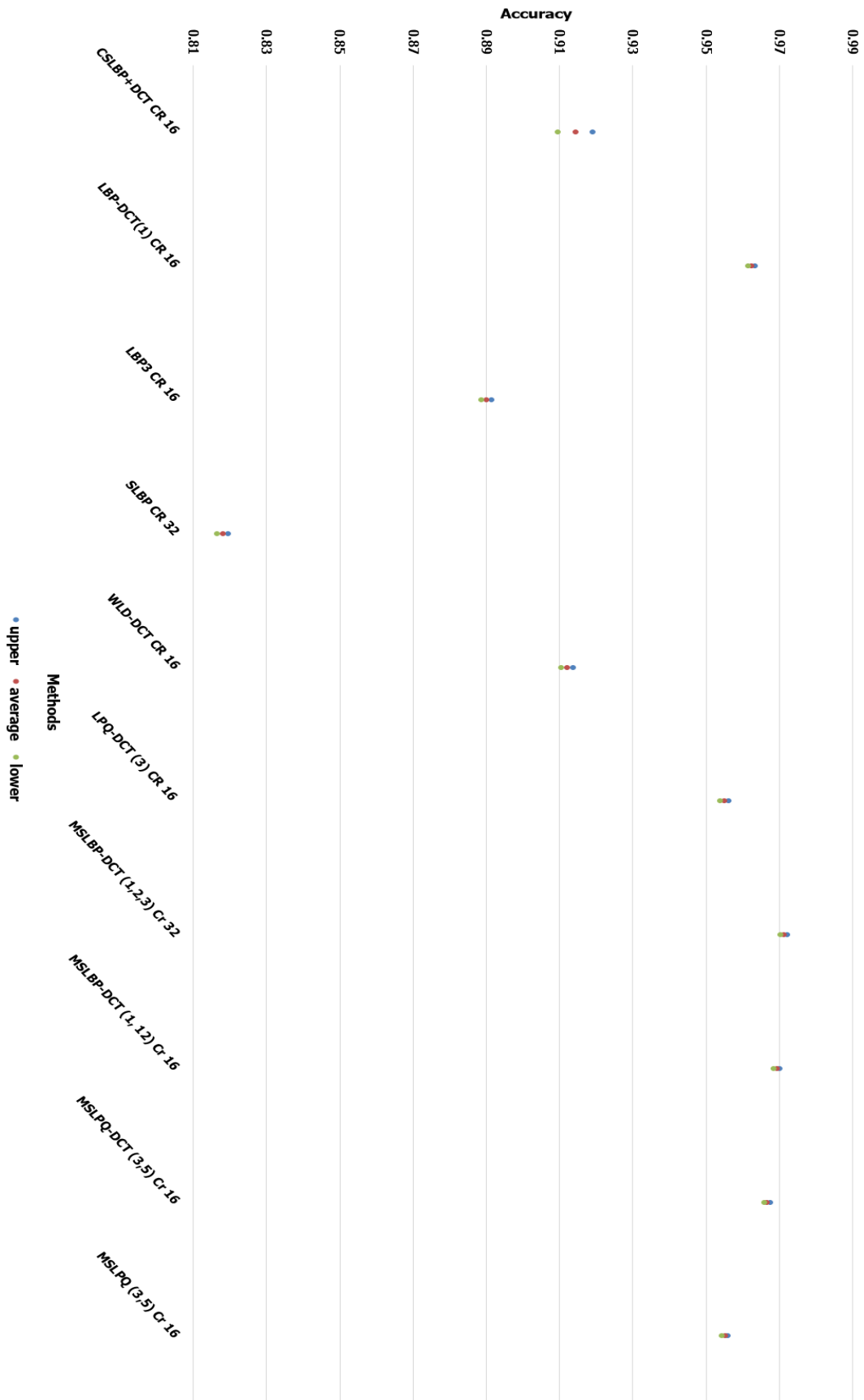


Figure 4.3: Confidence intervals for CASIA v1.0 dataset

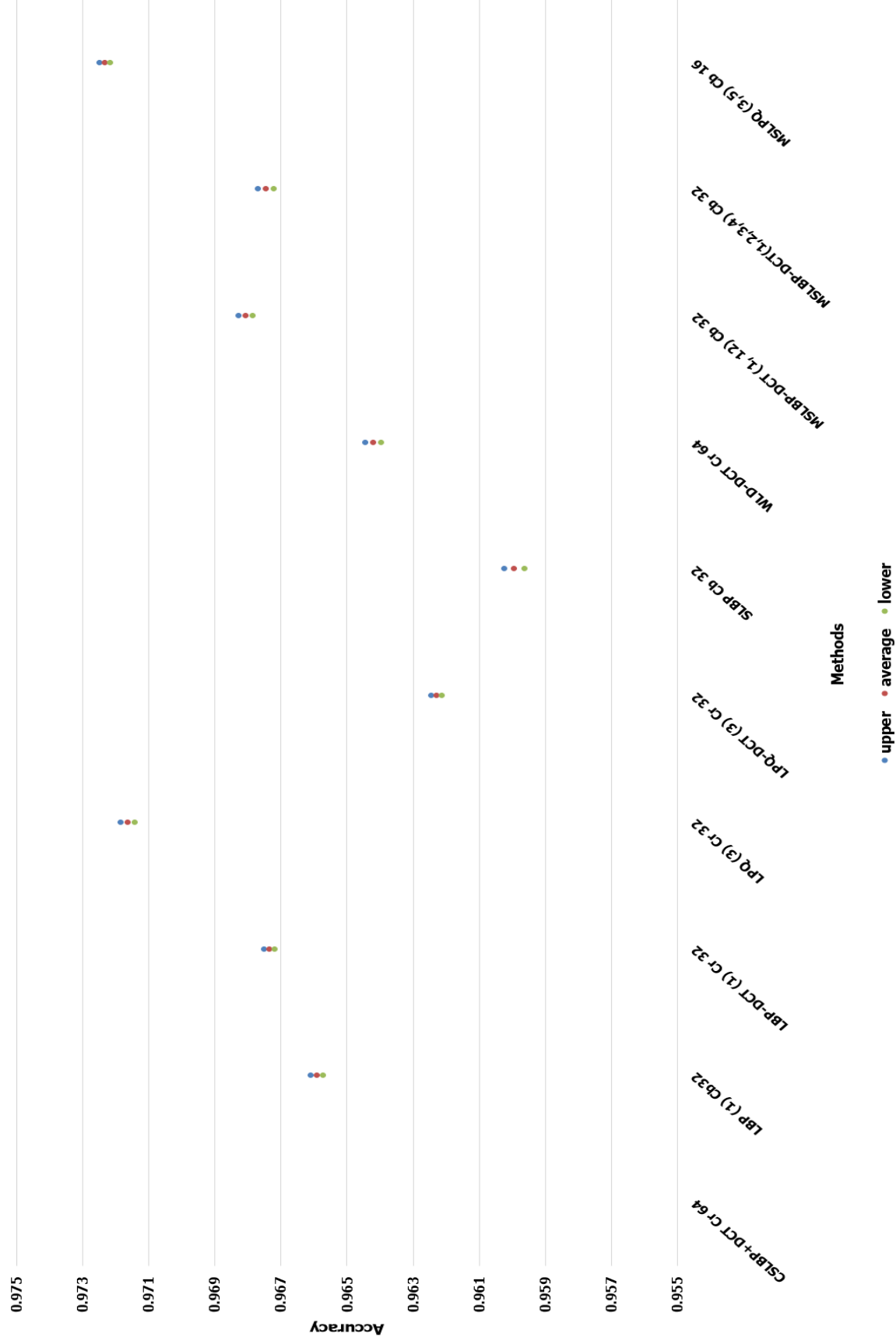


Figure 4.4: Confidence intervals for CASIA v2.0 dataset

S.No.	Methods	Channel	Block Size	Feature size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC
1	CSLBP+DCT	Cr	16	272	0.0843 ± 0.0047	0.0918 ± 0.0126	0.8959 ± 0.0116	0.9082 ± 0.0126	0.9157 ± 0.0047	0.9011 ± 0.0118	0.9143 ± 0.0048	0.9534 ± 0.00471
2	LBP-DCT (1)	Cr	16	196	0.0417 ± 0.0011	0.0327 ± 0.0013	0.9514 ± 0.0013	0.9673 ± 0.0013	0.9583 ± 0.0011	0.9590 ± 0.0010	0.9624 ± 0.0009	0.987 ± 0.0004
3	LBP (3)	Cr	16	256	0.1155 ± 0.0014	0.1029 ± 0.0025	0.8624 ± 0.0020	0.8971 ± 0.0025	0.8845 ± 0.0014	0.0879 ± 0.0015	0.8900 ± 0.0014	0.9458 ± 0.00097
4	SLBP	Cr	32	256	0.1908 ± 0.0020	0.1690 ± 0.0021	0.7644 ± 0.0030	0.8310 ± 0.0021	0.8092 ± 0.0021	0.7951 ± 0.0019	0.8180 ± 0.0015	0.8936 ± 0.00144
5	WLD-DCT	Cr	16	225	0.0854 ± 0.0015	0.0908 ± 0.0024	0.9004 ± 0.0020	0.9092 ± 0.0024	0.9146 ± 0.0015	0.9042 ± 0.0017	0.9121 ± 0.0016	0.9440 ± 0.00108
6	LPQ-DCT (3)	Cr	16	196	0.0543 ± 0.0013	0.0338 ± 0.0015	0.9355 ± 0.0016	0.9662 ± 0.0015	0.9457 ± 0.0013	0.9503 ± 0.0014	0.9550 ± 0.0012	0.9858 ± 0.0005
7	MSLBP-DCT (1,2,3)	Cr	32	2360	0.0333 ± 0.0010	0.0237 ± 0.0012	0.9613 ± 0.0013	0.9763 ± 0.0012	0.9667 ± 0.0010	0.9685 ± 0.0010	0.9712 ± 0.0009	0.9934 ± 0.00037
8	MSLBP-DCT (1,12)	Cr	16	296	0.0361 ± 0.0010	0.0243 ± 0.0014	0.9579 ± 0.0012	0.9757 ± 0.0014	0.9639 ± 0.0010	0.9665 ± 0.0010	0.9692 ± 0.0008	0.9910 ± 0.00043
9	MSLPQ-DCT (3,5)	Cr	16	340	0.0371 ± 0.0010	0.0288 ± 0.0012	0.9564 ± 0.0014	0.9712 ± 0.0012	0.9629 ± 0.0010	0.9635 ± 0.0009	0.9666 ± 0.0008	0.9909 ± 0.00047
10	MSLPQ (3,5)	Cr	16	512	0.0538 ± 0.0014	0.0341 ± 0.0009	0.9367 ± 0.0017	0.9659 ± 0.0009	0.9462 ± 0.0014	0.9508 ± 0.0009	0.9552 ± 0.0008	0.9874 ± 0.00036

Table 4.6: CASIA v1.0 statistical analysis

S.No.	Methods	Channel	Block Size	Feature size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC
1	LBP (1)	Cb	32	256	0.0590 ± 0.0003	0.0159 ± 0.0003	0.9581 ± 0.0002	0.9841 ± 0.0003	0.9410 ± 0.0003	0.9709 ± 0.0002	0.9659 ± 0.0002	0.9910 ± 0.0001
2	LBP-DCT (1)	Cr	32	900	0.0585 ± 0.0002	0.0137 ± 0.0002	0.9583 ± 0.0001	0.9863 ± 0.0002	0.9415 ± 0.0002	0.9721 ± 0.0001	0.9674 ± 0.0002	0.9922 ± 0.0001
3	LPQ (3)	Cr	32	256	0.0493 ± 0.0003	0.0132 ± 0.0003	0.9652 ± 0.0002	0.9868 ± 0.0003	0.9507 ± 0.0003	0.9758 ± 0.0002	0.9716 ± 0.0002	0.9932 ± 0.0000
4	LPQ-DCT (3)	Cr	32	900	0.0672 ± 0.0003	0.0159 ± 0.0003	0.9519 ± 0.0002	0.9841 ± 0.0003	0.9328 ± 0.0003	0.9677 ± 0.0001	0.9623 ± 0.0002	0.9899 ± 0.0000
5	SLBP	Cb	32	256	0.0651 ± 0.0004	0.0219 ± 0.0004	0.9539 ± 0.0003	0.9781 ± 0.0004	0.9349 ± 0.0004	0.9658 ± 0.0003	0.9600 ± 0.0003	0.9893 ± 0.0001
6	WLD-DCT	Cr	64	3969	0.0670 ± 0.0003	0.0125 ± 0.0003	0.9517 ± 0.0002	0.9875 ± 0.0003	0.9330 ± 0.0003	0.9693 ± 0.0002	0.9642 ± 0.0002	0.9900 ± 0.0001
7	MSLBP-DCT (1,12)	Cb	32	1576	0.0593 ± 0.0002	0.0117 ± 0.0003	0.9576 ± 0.0001	0.9883 ± 0.0003	0.9407 ± 0.0002	0.9727 ± 0.0002	0.9681 ± 0.0002	0.9920 ± 0.0000
8	MSLBP-DCT (1,2,3,4)	Cb	32	3144	0.0603 ± 0.0002	0.0121 ± 0.0004	0.9569 ± 0.0002	0.9879 ± 0.0004	0.9398 ± 0.0002	0.9722 ± 0.0002	0.9675 ± 0.0002	0.9916 ± 0.0000
9	MSLPQ (3,5)	Cb	16	512	0.0512 ± 0.0002	0.0105 ± 0.0003	0.9636 ± 0.0001	0.9895 ± 0.0001	0.9488 ± 0.0002	0.9764 ± 0.0001	0.9723 ± 0.0002	0.9927 ± 0.0000
10	MSLPQ-DCT (3,5)	Cb	32	1684	0.0591 ± 0.0002	0.0106 ± 0.0002	0.9576 ± 0.0002	0.9895 ± 0.0002	0.9409 ± 0.0002	0.9732 ± 0.0002	0.9688 ± 0.0002	0.9914 ± 0.0000

Table 4.7: CASIA v2.0 statistical analysis

better on each dataset. The resultant feature vector will become larger after combining all the features of each three channels. To reduce these features we used PCA and LLP which were discussed in Chapter 3. We used an empirical method for PCA where we select 50 features initially followed by training and classification. We repeated the steps for both CASIA v1.0 and CASIA v2.0 datasets until satisfactory results were achieved.

The PCA features selection are illustrated in Figure 4.5 for CASIA v1.0 and Figure 4.6 for CASIA v2.0. For LLP features reduction, the random features and kernel size options were set to 2 for the CASIA v1.0. For CASIA v2.0 dataset, the required random features and kernel size were set to 2 and 10, respectively for all methods except LPQ (3), where the kernel size was set to 12.

Table 4.8 and Table 4.9 shows the experiments of combining all three channels (Y, Cb and Cr) for the top five methods on each dataset. Both tables show the experiments on all features, PCA selected features and LLP selected features. The highest accuracies are highlighted for each section in both Tables.

We calculated the time for the top ten methods for both datasets on the same machine to do fair comparison. The results are shown in Table 4.10 and Table 4.11 for CASIA v1.0 and CASIA v2.0, respectively. The Core i7 laptop with processor 2.80 GHz and 32 GB RAM was used for to determine these times during feature extraction.

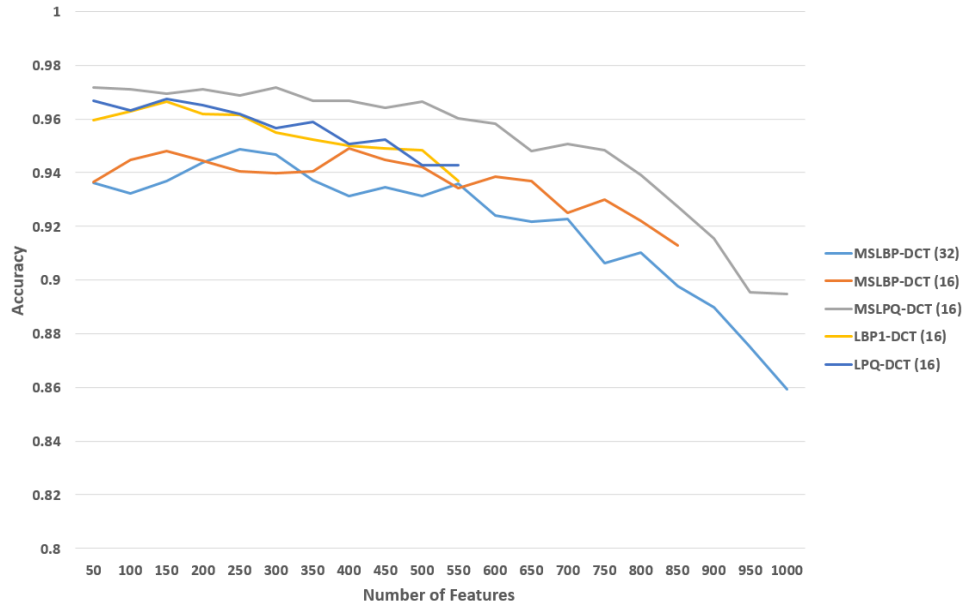


Figure 4.5: PCA features reduction and selection for CASIA v1.0

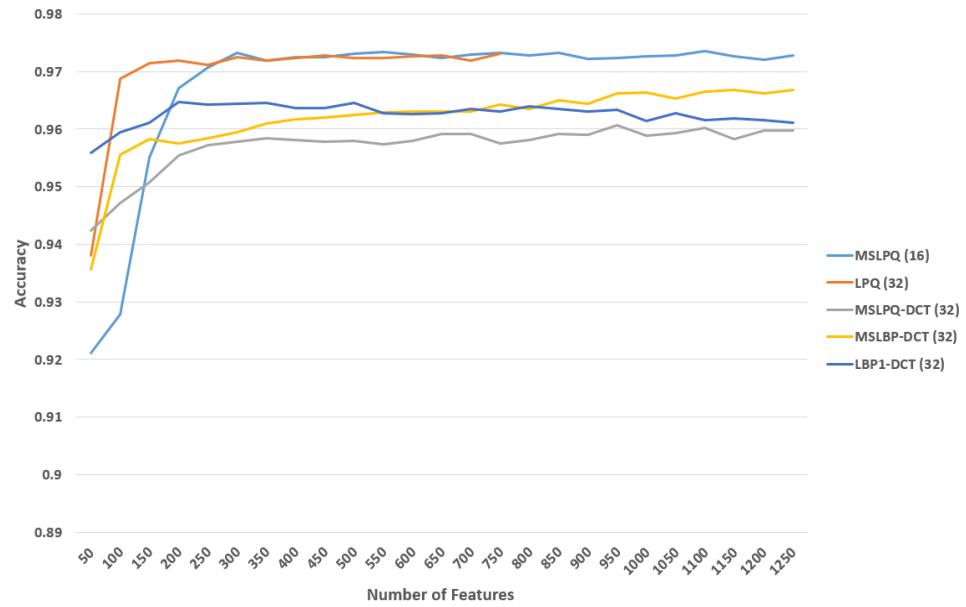


Figure 4.6: PCA features reduction and selection for CASIA v2.0

No	Methods	Block Size	Feature size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC
1	MSLBP-DCT (1,2,3)	32	7080	0.0263	0.0229	0.9677	0.9771	0.9737	0.9722	0.9750	0.9923
2	MSLBP-DCT (1,12)	16	888	0.0232	0.0164	0.9723	0.9836	0.9768	0.9778	0.9798	0.9938
3	MSLPQ-DCT (3,5)	16	1020	0.0163	0.0121	0.9814	0.9879	0.9837	0.9846	0.9855	0.9971
4	LBP-DCT (1)	16	588	0.0271	0.0185	0.9659	0.9815	0.9729	0.9735	0.9762	0.9933
5	LPQ-DCT (3)	16	588	0.0267	0.0136	0.9687	0.9864	0.9733	0.9773	0.9791	0.9961
PCA											
1	MSLBP-DCT (1,2,3)	32	250	0.0619	0.0459	0.9256	0.9541	0.9381	0.9393	0.9448	0.9846
2	MSLBP-DCT (1,12)	16	400	0.0576	0.0418	0.9307	0.9582	0.9424	0.9438	0.9491	0.9844
3	MSLPQ-DCT (3,5)	16	100	0.0356	0.0213	0.9591	0.9787	0.9644	0.9687	0.9717	0.9916
4	LBP-DCT (1)	16	150	0.0371	0.0281	0.9563	0.9719	0.9629	0.9639	0.9671	0.9897
5	LPQ-DCT (3)	16	150	0.0346	0.0307	0.9600	0.9693	0.9654	0.9645	0.9675	0.9920
LLP											
1	MSLBP-DCT (1,2,3)	32	537	0.0283	0.0202	0.9676	0.9798	0.9717	0.9735	0.9756	0.9966
2	MSLBP-DCT (1,12)	16	460	0.0281	0.0194	0.9673	0.9806	0.9719	0.9738	0.9762	0.9945
3	MSLPQ-DCT (3,5)	16	442	0.0194	0.0122	0.9776	0.9878	0.9806	0.9826	0.9838	0.9967
4	LBP-DCT (1)	16	303	0.0329	0.0210	0.9610	0.9790	0.9671	0.9697	0.9722	0.9952
5	LPQ-DCT (3)	16	271	0.0280	0.0125	0.9679	0.9875	0.9720	0.9775	0.9791	0.9957

Table 4.8: Combining all Y, Cb and Cr channels for the top five methods and with feature reduction using PCA and LLP for CASIA v1.0

No	Methods	Block Size	Feature size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC
1	MSLPQ (3,5)	16	1536	0.0455	0.0120	0.9680	0.9880	0.9545	0.9779	0.9740	0.9923
2	LPQ (3)	32	768	0.0443	0.0137	0.9689	0.9863	0.9557	0.9775	0.9735	0.9928
3	MSLPQ-DCT (3,5)	32	5052	0.0617	0.0123	0.9558	0.9877	0.9383	0.9715	0.9667	0.9873
4	MSLBP-DCT (1,2,3,4)	32	9432	0.0545	0.0119	0.9613	0.9881	0.9455	0.9745	0.9702	0.9878
5	LBP-DCT (1)	32	2700	0.0583	0.0153	0.9584	0.9847	0.9417	0.9714	0.9666	0.9871
PCA											
1	MSLPQ (3,5)	16	400	0.0450	0.0152	0.9685	0.9848	0.9550	0.9766	0.9724	0.9895
2	LPQ (3)	32	400	0.0434	0.0171	0.9697	0.9829	0.9566	0.9763	0.9720	0.9903
3	MSLPQ-DCT (3,5)	32	400	0.0732	0.0184	0.9472	0.9816	0.9268	0.9641	0.9582	0.9832
4	MSLBP-DCT (1,2,3,4)	32	300	0.0656	0.0210	0.9534	0.9790	0.9344	0.9660	0.9601	0.9842
5	LBP-DCT (1)	32	250	0.0607	0.0158	0.9568	0.9842	0.9393	0.9703	0.9652	0.9851
LLP											
1	MSLPQ (3,5)	16	1061	0.0453	0.0128	0.9681	0.9872	0.9547	0.9775	0.9736	0.9927
2	LPQ (3)	32	34	0.0543	0.0165	0.9617	0.9835	0.9457	0.9725	0.9677	0.9899
3	MSLPQ-DCT (3,5)	32	1736	0.0638	0.0135	0.9542	0.9865	0.9362	0.9700	0.9651	0.9887
4	MSLBP-DCT (1,2,3,4)	32	5817	0.0539	0.0114	0.9615	0.9886	0.9461	0.9749	0.9706	0.9876
5	LBP-DCT (1)	32	1062	0.0587	0.0187	0.9582	0.9813	0.9413	0.9696	0.9645	0.9859

Table 4.9: Combining all Y, Cb and Cr channels for the top five methods and with feature reduction using PCA and LLP for CASIA v2.0

Block size	16×16			32×64			64×64		
Color Channels	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr
LBP(1)-Image	176.2940	166.8104	165.6267	62.6565	60.3864	60.0612	28.8923	26.8544	26.7070
LBP(1)-Histogram	63.6946	65.6545	65.2238	11.5371	11.0826	10.9837	3.0015	2.9052	2.8480
LBP(1)-DCT	70.8976	70.4198	70.0438	28.0613	28.1950	28.0383	14.0683	14.0875	14.1718
LBP(2)-Image	172.4062	165.7955	160.4395	60.6971	58.8965	58.7327	28.2595	26.3663	26.7437
LBP(2)-Histogram	64.7095	64.9763	64.5195	11.3977	10.9400	10.8873	3.2107	3.1372	3.1298
LBP(2)-DCT	66.3007	66.0574	64.6473	26.7801	26.8365	26.7026	12.3593	12.3287	12.6153
LBP(3)-Image	167.3342	160.1269	162.9714	59.8492	57.7709	57.6398	27.8945	26.0647	25.7016
LBP(3)-Histogram	63.2322	64.4490	64.6032	11.3127	10.8320	10.7763	3.0997	3.0578	3.0473
LBP(3)-DCT	61.1492	60.5448	62.0482	26.3572	26.2754	25.9915	13.1470	13.1497	13.0784
LBP(4)-Image	158.6983	156.5116	155.8187	57.8960	55.8837	56.2398	26.8857	25.1945	25.0169
LBP(4)-Histogram	62.4629	64.4100	64.3549	11.1361	10.5641	10.6151	2.9283	2.9078	2.9137
LBP(4)-DCT	53.9861	54.0746	53.9370	25.0076	25.1039	24.8310	10.9837	10.9652	11.0099
LBP($r=2,p=12$)-Image	409.1113	424.7887	412.0678	132.1984	125.7329	125.1304	53.7853	51.2133	51.8700
LBP($r=2,p=12$)-Histogram	64.4268	65.5000	64.7839	13.3933	12.0513	11.9330	2.6861	2.7467	2.8828
LBP($r=2,p=12$)-DCT	54.5657	56.3804	55.9845	27.0848	25.3410	24.7915	12.1032	12.0777	12.6517
LPQ(3)-Image	179.9930	177.4458	178.7122	75.1161	72.9201	71.1777	41.8053	39.8378	39.1060
LPQ(3)-Histogram	73.7945	77.7251	78.3974	14.3517	14.0072	13.6484	3.3204	3.2797	3.2444
LPQ(3)-DCT	71.0743	72.7288	73.3111	30.2466	29.9132	29.1846	14.7047	14.6061	14.4375
LPQ(5)-Image	283.7729	287.1535	288.5235	101.0481	95.4355	94.6841	47.0925	44.9720	44.7135
LPQ(5)-Histogram	69.4573	73.5117	74.6105	14.4528	13.7293	13.5847	3.3676	3.2964	3.2947
LPQ(5)-DCT	63.1762	65.7383	66.4402	28.7600	27.6091	27.3291	12.9415	12.5711	12.5401
WLD-Image	0.5529	0.5310	0.5135	0.2614	0.2545	0.2466	0.4104	0.4075	0.3984
WLD-Histogram	222.3884	214.2961	212.9309	232.5179	228.2249	227.7092	238.6832	234.9850	233.1280
WLD-DCT	49.9230	49.4841	49.3113	33.8853	33.6808	33.6976	15.8010	15.8161	15.7429
SLBP	446.5647	439.2174	441.8951	153.2748	137.4884	137.4928	100.2084	84.1631	83.9782

Table 4.10: Feature extraction time (seconds) for top 10 methods on CASIA v1.0 dataset

Block size	16×16			32×64			64×64		
Color Channels	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr
LBP(1)-Image	2014.7545	1938.5921	1935.4696	698.9493	672.5540	670.2718	329.4689	305.5270	303.0007
LBP(1)-Histogram	2158.7363	2161.4231	2158.6762	147.2108	143.1278	142.7563	37.5556	34.9810	34.8821
LBP(1)-DCT	1721.2276	1720.3524	1713.4385	603.4495	605.3809	602.7153	262.1444	261.8835	260.4069
LBP(2)-Image	1864.8796	1842.4524	1842.1134	677.3003	652.7959	655.2949	325.8113	301.7240	300.9148
LBP(2)-Histogram	2138.5858	2140.9863	2139.1353	144.2460	139.7642	139.9526	38.6787	37.2029	36.9001
LBP(2)-DCT	1248.2973	1256.9357	1257.9895	529.0753	530.4410	527.5548	231.3587	230.5588	230.6730
LBP(3)-Image	1861.0890	1801.8736	1838.8521	659.4557	638.6488	639.8697	318.1916	297.4833	297.6035
LBP(3)-Histogram	2124.7652	2136.8326	2133.9769	144.1151	140.2629	139.9595	38.0711	36.9851	36.7949
LBP(3)-DCT	922.2456	914.6532	929.9130	462.8010	463.7322	461.2070	232.1268	231.1903	231.8662
LBP(4)-Image	1754.3665	1734.5094	1733.7208	627.0902	611.5039	612.3558	299.1206	285.3099	283.4604
LBP(4)-Histogram	2137.0800	2136.6343	2134.6452	139.7426	135.9436	135.6404	36.6303	35.8724	35.6983
LBP(4)-DCT	655.7083	663.0678	664.0066	401.7576	402.3828	400.6888	200.5558	201.3797	201.3891
LBP($r=2, p=12$)-Image	4418.2041	4517.1711	4419.3294	1408.4480	1355.4845	1377.0678	600.0061	577.8848	580.4135
LBP($r=2, p=12$)-Histogram	2128.4950	2125.9109	2117.6324	158.4098	147.6309	148.7545	38.0390	36.2340	37.5880
LBP($r=2, p=12$)-DCT	884.1289	892.4155	891.3386	475.6070	461.7887	465.0048	230.5431	228.0357	230.4644
LPQ(3)-Image	2070.3108	2034.5533	2005.7394	781.2829	758.2197	745.9835	441.3004	409.5997	414.7119
LPQ(3)-Histogram	2260.5280	2282.8463	2277.9856	171.5670	169.5135	167.5964	37.9692	36.9305	37.2665
LPQ(3)-DCT	1758.6726	1765.1941	1758.4000	617.4397	615.8871	611.1944	264.0373	259.7972	261.6279
LPQ(5)-Image	3168.9442	3182.4855	3192.2959	1049.4546	997.4962	998.8659	535.2046	503.0293	506.4636
LPQ(5)-Histogram	2224.4029	2237.4813	2240.0990	179.4236	173.3701	173.0281	41.0811	39.7949	40.0984
LPQ(5)-DCT	1271.4615	1281.4849	1285.2214	563.5208	553.2475	552.4563	249.0115	244.4621	245.2449
WLD-Image	5.8116	5.6402	5.7797	2.5682	2.5609	2.5389	6.1864	6.1822	6.1331
WLD-Histogram	3467.3097	3414.3591	3412.9389	2725.3844	2695.4291	2688.1520	2672.4345	274.9211	2624.5940
WLD-DCT	1696.5871	1686.4569	1684.7045	684.3991	683.6575	683.0268	276.3611	2619.4240	275.3926
SLBP	6005.2910	5999.2355	6034.5691	1666.4116	1502.2914	1505.7921	1095.9160	946.7956	934.0653

Table 4.11: Feature extraction time (seconds) for top 10 methods on CASIA v2.0 dataset

4.7 ROC-Based Comparisons

The ROC comparison of selected methods of both datasets using a single channel is illustrated in Figure 4.7 and Figure 4.8. Figure 4.7 illustrates the ROC curves of five methods which achieved the highest accuracy on CASIA v1.0 dataset. It shows that MSLBP-DCT (1, 2, 3) using Cr channel and block size of 32×32 performed better than other methods. Figure 4.8 shows the ROC curves of five those methods which achieved the highest accuracy using CASIA v2.0 dataset. It can be seen clearly that the MSLPQ (3, 5) using Cb channel with block size of 16×16 performed better than other methods.

Table 4.12 illustrate the comparison of the proposed methods with previous work. Many previous methods also used YCbCr channels for forgery detection due to its texture discrimination property. If we look at the proposed methods using CASIA v1.0 dataset, MSLBP-DCT in Cr channel performed better with the accuracy of 97.34% and AUC 0.99 while the highest achieved 98% in [71] using deep learning model with color image of three channels. Alahmadi et al. [47] used the fusing of Cr and Cb features to achieve 97.0% on CASIA v1.0 dataset while our method MSLBP-DCT used Cr, a single channel which reduces the complexity of learning algorithm.

If we look at the experiments on CASIA v2.0 dataset, among the proposed methods, MSLPQ with Cb channel performed better with an accuracy of 97.3% and AUC 0.99. The study [66] achieved 99% accuracy but our MSLBP-DCT method performed better in AUC using both datasets and achieved better accuracy using CASIA v1.0.

The last section of Table 4.12 shows the results of the experiments using all the

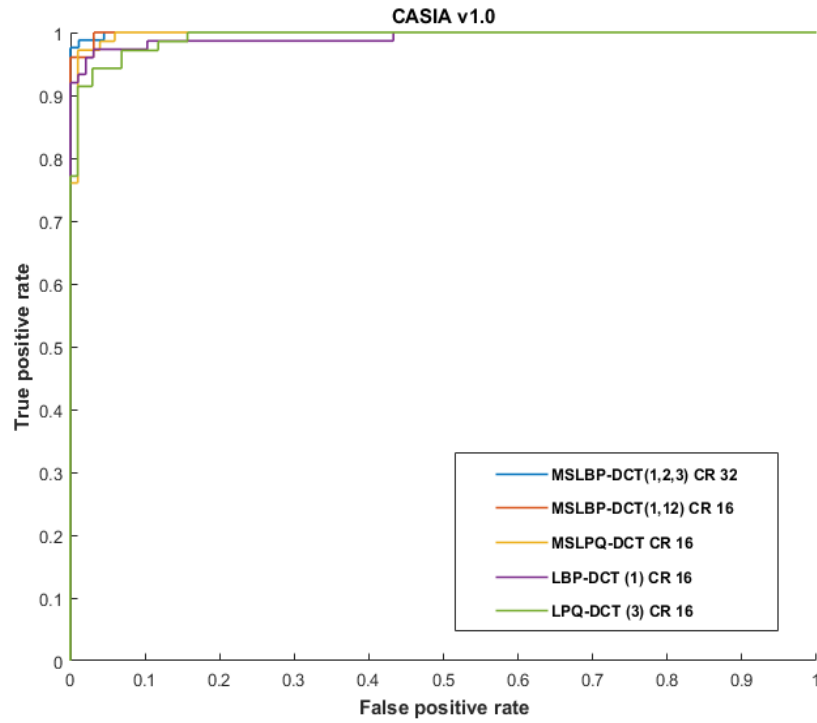


Figure 4.7: ROC curves of top five method using CASIA v1.0

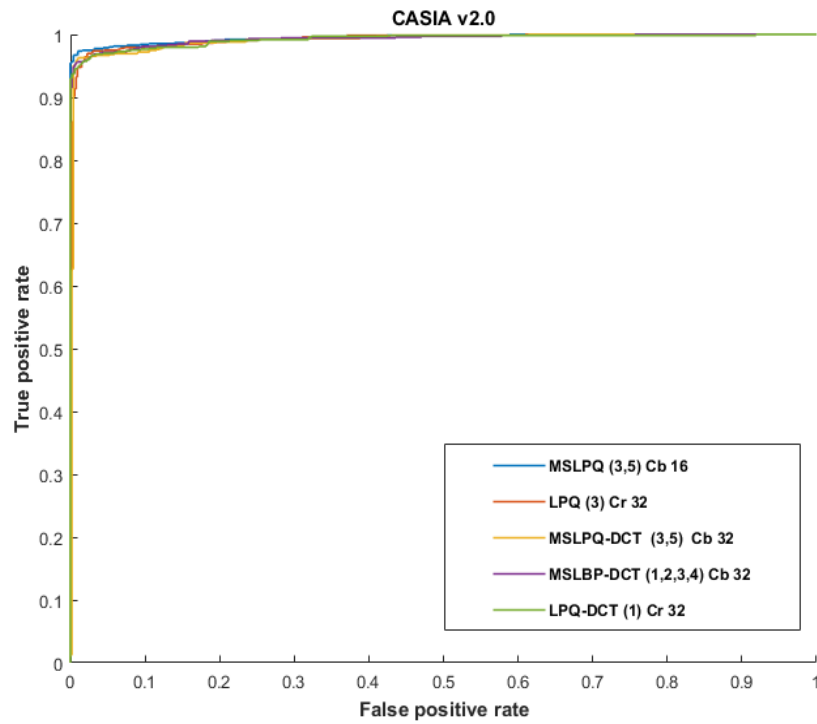


Figure 4.8: ROC curves of top five method using CASIA v2.0

channels. All the features of each channel were fused together to make a feature vector and fed to the SVM classifier to classify authentic and forged images. In this part, the MSLPQ-DCT on CASIA v1.0 performed better than all the previous methods including deep learning models and achieved an accuracy of 98.56% with AUC 0.99 and only 1020 features. Using the LLP features reduction method, the feature vector was reduced to 442 and achieved 98.37% accuracy with 0.99 AUC.

Methods	Color Channels	Accuracy		AUC
		CASIA v1.0	CASIA v2.0	
Zhao et al. 2010 [27]	Cr	94.7%	-	
Wang et al. 2010 [58]	Cb (CASIA v1.0)	95.6%	96.8%	
	Cr (CASIA v2.0)			
He et al. 2012 [59]	-	-	89.7%	
Al-Hammadi et al. 2013 [112]	Cb, Cr	91.74%	97.0%	
Saleh et a. 2013 [113]	Cr+Cb	94.52%	96.28%	
Hussain et al. 2013 [34]	Cr+Cb	93.3%	-	0.93
Hussain et al. 2014 [114]	Cr	94.29%	-	
Muhammad et al. 2014 [60]	Cr (CASIA v1.0),	94.8% (480)	97.3 %(3584)	0.93 (CASIA v1.0)
	Cr & Cb (CASIA v 2.0)			0.97(CASIA v2.0)
Alahmadi et al. 2016 [47]	Cr + Cb	97.0%	97.5%	
Rao et al. 2016 [71]	RGB	98%	97.8%	0.99(CASIA v 1.0)
				0.98(CASIA v 2.0)
Zhang et al. 2016 [80]	RGB	91.09%		
Issac & Wilsy et al. 2017 [66]	Cr	96.3%	99%	0.94 (CASIA v1.0)
				0.98 (CASIA v2.0)
Agarwal & Chad 2018 [68]	YCbCr	96.8%	-	
Proposed Methods				
CNN	RGB	91%	-	-
MSLBP-DCT ($r=1,2,3$ $p=8$)	Cr(CASIA v1.0),	97.34% (2360)	96.8%	0.99(CASIA v 1.0)
	Cb(CASIA v2.0)			0.99(CASIA v 2.0)
MSLBP-DCT ($r=1,p=8$ and $r=2, p=12$)	Cr(CASIA v1.0)	96.9%	96.8%	0.99(CASIA v 1.0)
	Cb (CASIA v2.0)			0.99(CASIA v 2.0)
MSLPQ-DCT (3,5)	Cr(CASIA v1.0)	97.0%	96.9%	0.99(CASIA v 1.0)
	Cb (CASIA v2.0)			0.99(CASIA v 2.0)
MSLPQ (3,5)	Cr(CASIA v1.0)	95%	97.3% (512)	0.98(CASIA v 1.0)
	Cb,Cr (CASIA v2.0)			0.99(CASIA v 2.0)
Combination of Y+Cb+Cr Channels				
MSLPQ-DCT(3,5)	CASIA v1.0	98.56% (1020)	-	0.99(CASIA v 1.0)
MSLPQ-DCT(3,5) (LLP)	CASIA v1.0	98.37% (442)	-	0.99(CASIA v 1.0)
MSLPQ(3,5)	CASIA v2.0	-	97.40% (1536)	0.99(CASIA v 2.0)
MSLPQ(3,5) (LLP)	CASIA v2.0	-	97.36% (1061)	0.99 (CASIA v 2.0)

Table 4.12: Comparison of proposed method with previous methods in terms of Accuracy and AUC

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

Image forensic is one of the important areas of research in today's technological era. Thousands of images are shared via social, print and electronic media. These images are now an important part of our life due to a large number of image acquisition devices. Such images also include fake images. These forged images are used for political victimization, security threats and changing the physiological behavior of the consumers. Due to a large number of advanced image processing tools, such as Photoshop, Corel-Draw and Adobe illustrator, anyone can easily manipulate these images which cannot be seen via bare human eyes. Two main types of forgery are commonly used to alter the content of an image which are, copy-move forgery and image splicing forgery. In copy-move forgery, the image regions are replicated in a single image whereas in splicing two images are fused together to make a single image.

In this work, we address the problem of detecting image forgery using feature

and deep learning based approaches. In the feature-based approach, we use various feature extraction methods, which include texture features such as LBP, MSLBP, CSLBP, SLBP, WLD, LPQ, MSLPQ and DCT. Images were divided into small blocks of sizes 16×16 , 32×32 and 64×64 ; feature vectors are then created and used to train the classifier. SVM with Polynomial and RBF kernels were used to evaluate the results using two publically available datasets called CASIA v1.0 and CASIA v2.0. In addition, we used a deep learning approach where a ten-layer CNN model was used to detect image forgery. The CNN model needs a lot of training data, and that explains why the data preparation for such a network was a crucial step. Image patches are extracted from the authentic and forged images and data augmentation was used to increase the dataset size from 1721 images to 99376 patches. The dataset was divided into training and validation set of 75% and 25%, respectively. The ten-layer CNN model with regularization steps achieved a validation accuracy of 91%.

To evaluate the feature-based method, we used SVM with k -fold validation using two benchmark datasets. The other performance measures includes precision, recall/sensitivity, specificity, F_1 score, true negative rate, true positive rate, false negative rate, false positive rate, accuracy and area under the curve.

The experiments were initially conducted using a single channel where the MSLBP-DCT (1, 2 and 3 radii) achieved better performance using Cr channel on CASIA v1.0 with an accuracy of 97.3% with AUC 0.99 and 2360 features. The experiments on CASIA v2.0, MSLPQ achieved the highest accuracy of 97.3% with AUC 0.99 using each Cb and Cr channels with only 512 features.

To improve the performance, the features of all three channels were combined

and evaluated. The experiments used 16×16 and 32×32 block sizes for both datasets, which were empirically considered for further experiments. Using CASIA v1.0, MSLPQ-DCT performed better with an accuracy of 98.55% using 1020 features. The features were reduced to 442 using LLP causing the accuracy to be 98.38%. Experiments on CASIA v2.0 show that MSLPQ performed better with an accuracy of 97.40% using 1536 features and after feature reduction using LLP, 97.36% accuracy was achieved using 1061 features. All the experiments were evaluated statistically using 95% confidence interval to check its significance.

In future, this work will be extended to image forgery localization where the forged region will be identified. This work can be done via noise estimation and JPEG compression techniques where the forged area has different noise ratio. Forgery localization can also be done using deep learning techniques but the parameter tuning of such models needs resources and time-consuming.

APPENDIX A

DETAILED EXPERIMENTS AND RESULTS

A.1 Feature Based Experiments

Table A.1 shows the results of CSLBP and CSLBP with fusion of DCT coefficients. The CSLBP with DCT performed better in Cr channel with 64×64 block size which achieved 88.03% accuracy and 0.94 AUC with 4480 features.

Table A.2 shows the results of SLBP and SLBP with fusion of DCT coefficients. The SLBP with DCT coefficients performed better accuracy than SLBP features alone. As illustrated in Table A.2, SLBP+DCT in Cr channel with block size 64×64 achieved 83.97% accuracy and 0.92 AUC with 10240 features.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
CSLBP											
Y	16×16	0.5383	0.6674	0.2638	0.3326	0.4617	0.2903	0.4123	0.4037	6144	
	32×32	0.5233	0.6368	0.2920	0.3632	0.4767	0.3223	0.4335	0.4185	1536	
	64×64	0.5056	0.6063	0.3250	0.3937	0.4945	0.3531	0.4551	0.4518	384	
Cb	16×16	0.3926	0.4338	0.5048	0.5662	0.6074	0.5324	0.5903	0.6579	6144	
	32×32	0.3414	0.3392	0.5424	0.6608	0.6586	0.5941	0.6590	0.7277	1536	
	64×64	0.2987	0.2884	0.6092	0.7116	0.7013	0.6515	0.7023	0.7764	384	
Cr	16×16	0.3690	0.3981	0.5222	0.6019	0.6310	0.5583	0.6215	0.6943	6144	
	32×32	0.2912	0.2850	0.6244	0.7150	0.7088	0.6652	0.7110	0.7742	1536	
	64×64	0.2612	0.2417	0.6694	0.7583	0.7388	0.7094	0.7465	0.8114	384	
CSLBP+DCT											
Y	16×16	0.4828	0.5672	0.3569	0.4328	0.5172	0.3901	0.4861	0.4940	6400	
	32×32	0.3664	0.3909	0.5229	0.6091	0.6336	0.5603	0.6237	0.6711	2560	
	64×64	0.3352	0.3574	0.5813	0.6426	0.6648	0.6084	0.6545	0.7003	4480	
Cb	16×16	0.3247	0.3416	0.5904	0.6584	0.6753	0.6216	0.6682	0.7467	6400	
	32×32	0.2384	0.2375	0.7009	0.7625	0.7616	0.7283	0.7614	0.8598	2560	
	64×64	0.1688	0.1508	0.7938	0.8492	0.8312	0.8199	0.8386	0.9221	4480	
Cr	16×16	0.3105	0.3266	0.6157	0.6734	0.6895	0.6422	0.6823	0.7697	6400	
	32×32	0.2132	0.1962	0.7353	0.8038	0.7868	0.7671	0.7949	0.8841	2560	
	64×64	0.1244	0.1139	0.8505	0.8861	0.8756	0.8669	0.8803	0.9468	4480	

Table A.1: Results for each channel on CASIA v1.0 dataset using CSLBP and CSLBP+DCT. (Feature concatenation)

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
SLBP											
Y	16×16	0.5006	0.5809	0.3799	0.4191	0.4994	0.3965	0.4643	0.5155	98304	
	32×32	0.4621	0.5339	0.4076	0.4661	0.5379	0.4333	0.5081	0.5551	24576	
	64×64	0.4350	0.4974	0.4477	0.5026	0.5650	0.4721	0.5387	0.5939	6144	
Cb	16×16	0.3723	0.4363	0.5763	0.5637	0.6277	0.5677	0.5970	0.6659	98304	
	32×32	0.2920	0.3215	0.6481	0.6785	0.7080	0.6605	0.6935	0.7841	24576	
	64×64	0.2510	0.2693	0.6981	0.7307	0.7490	0.7126	0.7409	0.8337	6144	
Cr	16×16	0.3507	0.4221	0.6203	0.5779	0.6493	0.5971	0.6129	0.6839	98304	
	32×32	0.2666	0.3082	0.6907	0.6918	0.7334	0.6892	0.7129	0.7945	24576	
	64×64	0.2405	0.2356	0.7036	0.7644	0.7595	0.7310	0.7613	0.8478	6144	
SLBP+DCT											
Y	16×16	0.5010	0.5895	0.3675	0.4105	0.4990	0.3860	0.4620	0.5121	98560	
	32×32	0.4514	0.5177	0.4303	0.4823	0.5486	0.4537	0.5212	0.5789	25600	
	64×64	0.3616	0.3912	0.5361	0.6088	0.6384	0.5692	0.6271	0.6798	10240	
Cb	16×16	0.3581	0.4203	0.5918	0.5797	0.6419	0.5835	0.6119	0.6816	98560	
	32×32	0.2344	0.2799	0.7307	0.7201	0.7656	0.7238	0.7434	0.8281	25600	
	64×64	0.1830	0.1988	0.7890	0.8012	0.8170	0.7936	0.8089	0.8956	10240	
Cr	16×16	0.3400	0.4083	0.6222	0.5917	0.6600	0.6053	0.6263	0.7001	98560	
	32×32	0.2260	0.2620	0.7400	0.7380	0.7740	0.7378	0.7565	0.8492	25600	
	64×64	0.1613	0.1589	0.8095	0.8411	0.8387	0.8241	0.8397	0.9224	10240	

Table A.2: Results for each channel on CASIA v1.0 dataset using SLBP and SLBP+DCT. (Feature concatenation)

Table A.3 shows the results of WLD method with features concatenation on CASIA v1.0. The Cr channel with 16×16 block size achieved better accuracy of 61.76%.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features Dimension
WLD										
Y	16×16	0.5763	0.6521	0.3643	0.3479	0.4237	0.3543	0.3859	0.2419	86400
	32×32	0.5647	0.6369	0.3916	0.3631	0.4353	0.3749	0.3984	0.2558	92256
	64×64	0.5672	0.6387	0.3890	0.3613	0.4328	0.3739	0.3971	0.2519	95256
Cb	16×16	0.3664	0.4737	0.6951	0.5263	0.6336	0.5983	0.5674	0.5654	86400
	32×32	0.3475	0.4722	0.7207	0.5278	0.6525	0.6084	0.5734	0.5327	92256
	64×64	0.3570	0.4778	0.7164	0.5222	0.6430	0.6031	0.5665	0.5313	95256
Cr	16×16	0.2761	0.4362	0.7980	0.5638	0.7239	0.6590	0.6176	0.5865	86400
	32×32	0.2573	0.4423	0.8294	0.5577	0.7427	0.6651	0.6138	0.5418	92256
	64×64	0.2542	0.4492	0.8404	0.5508	0.7458	0.6640	0.6087	0.5224	95256

Table A.3: Results for each channel on CASIA v1.0 dataset using WLD. (Feature concatenation)

The LPQ and MSLPQ features are concatenated and used as feature vector, the results are illustrated in Table A.4. MSLPQ with Cr channel and 64×64 block size achieved 85.87% accuracy which is better than LPQ features alone.

Table A.5 shows the experimental results of MSLBP with various radii (1,2,3 and 4) on CASIA v1.0. The concatenation of LBP with radius 1, radius 2 and radius 3 better performance and achieved 83% accuracy in Cr channel with 64×64 block size.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
MSLPQ (3,5)											
Y	16×16	0.5275	0.6199	0.3521	0.3801	0.4725	0.3648	0.4335	0.4800		196608
	32×32	0.4868	0.5718	0.3678	0.4282	0.5132	0.3931	0.4780	0.5281		49152
	64×64	0.4485	0.5089	0.4134	0.4911	0.5515	0.4471	0.5267	0.5807		12288
Cb	16×16	0.3010	0.3115	0.6180	0.6885	0.6990	0.6494	0.6939	0.8123		196608
	32×32	0.2276	0.2232	0.7221	0.7768	0.7724	0.7475	0.7751	0.8861		49152
	64×64	0.1822	0.1516	0.7735	0.8484	0.8178	0.8084	0.8304	0.9358		12288
Cr	16×16	0.2910	0.3078	0.6418	0.6922	0.7090	0.6638	0.7012	0.8173		196608
	32×32	0.2164	0.2055	0.7340	0.7945	0.7836	0.7625	0.7881	0.9042		49152
	64×64	0.1588	0.1181	0.8002	0.8819	0.8412	0.8385	0.8587	0.9437		12288
LPQ (5)											
Y	16×16	0.5737	0.7037	0.2549	0.2963	0.4263	0.2720	0.3737	0.4020		98304
	32×32	0.5575	0.6849	0.2629	0.3151	0.4425	0.2854	0.3924	0.4063		24576
	64×64	0.5189	0.6301	0.2948	0.3699	0.4812	0.3263	0.4395	0.4346		6144
Cb	16×16	0.3881	0.4131	0.4880	0.5869	0.6119	0.5316	0.6013	0.7115		98304
	32×32	0.2928	0.2844	0.6234	0.7156	0.7072	0.6644	0.7114	0.8318		24576
	64×64	0.2210	0.1882	0.7199	0.8118	0.7790	0.7618	0.7930	0.8989		6144
Cr	16×16	0.3664	0.3794	0.5218	0.6206	0.6336	0.5653	0.6282	0.7339		98304
	32×32	0.2582	0.2332	0.6683	0.7668	0.7418	0.7125	0.7513	0.8584		24576
	64×64	0.2024	0.1482	0.7399	0.8518	0.7976	0.7907	0.8206	0.9171		6144
LPQ (3)											
Y	16×16	0.5013	0.5829	0.3901	0.4171	0.4987	0.4013	0.4623	0.5306		98304
	32×32	0.4573	0.5261	0.4199	0.4739	0.5427	0.4421	0.5128	0.5764		24576
	64×64	0.4342	0.4911	0.4342	0.5089	0.5658	0.4676	0.5449	0.6063		6144
Cb	16×16	0.2591	0.2869	0.6989	0.7131	0.7409	0.7047	0.7288	0.8323		98304
	32×32	0.2300	0.2361	0.7228	0.7639	0.7700	0.7413	0.7671	0.8819		24576
	64×64	0.1909	0.1687	0.7645	0.8314	0.8091	0.7958	0.8182	0.9239		6144
Cr	16×16	0.2536	0.2904	0.7062	0.7096	0.7464	0.7061	0.7286	0.8348		98304
	32×32	0.2123	0.2109	0.7450	0.7891	0.7877	0.7636	0.7863	0.8983		24576
	64×64	0.1741	0.1428	0.7845	0.8572	0.8259	0.8183	0.8385	0.9285		6144

Table A.4: Results for each channel on CASIA v1.0 dataset using LPQ and MSLPQ. (Feature concatenation)

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
MSLBP (1,2,3,4)											
Y	16×16	0.5413	0.6359	0.3405	0.3641	0.4587	0.3511	0.4192	0.4652	393216	
	32×32	0.4984	0.5825	0.3754	0.4175	0.5016	0.3939	0.4660	0.5211	98304	
	64×64	0.4429	0.5113	0.4389	0.4887	0.5571	0.4606	0.5281	0.5932	24576	
Cb	16×16	0.3793	0.4460	0.5836	0.5540	0.6207	0.5669	0.5875	0.6671	393216	
	32×32	0.2474	0.2564	0.7003	0.7436	0.7526	0.7202	0.7484	0.8440	98304	
	64×64	0.1773	0.1701	0.7830	0.8299	0.8227	0.8048	0.8257	0.9230	24576	
Cr	16×16	0.3563	0.4275	0.6233	0.5725	0.6437	0.5952	0.6082	0.6884	393216	
	32×32	0.2440	0.2794	0.7158	0.7206	0.7560	0.7171	0.7391	0.8350	98304	
	64×64	0.1795	0.1646	0.7835	0.8354	0.8205	0.8065	0.8259	0.9190	24576	
MSLBP (1,2,3)											
Y	16×16	0.5373	0.6327	0.3478	0.3673	0.4627	0.3561	0.4212	0.4847	294912	
	32×32	0.4808	0.5620	0.3969	0.4380	0.5192	0.4143	0.4844	0.5444	73728	
	64×64	0.4389	0.4979	0.4544	0.5021	0.5611	0.4745	0.5348	0.6015	18432	
Cb	16×16	0.3416	0.4091	0.6115	0.5909	0.6584	0.5980	0.6238	0.7081	294912	
	32×32	0.2349	0.2437	0.7156	0.7563	0.7651	0.7338	0.7612	0.8566	73728	
	64×64	0.1817	0.1753	0.7789	0.8247	0.8183	0.8007	0.8211	0.9175	18432	
Cr	16×16	0.3189	0.4043	0.6681	0.5957	0.6811	0.6281	0.6363	0.7130	294912	
	32×32	0.2231	0.2565	0.7421	0.7435	0.7769	0.7420	0.7608	0.8564	73728	
	64×64	0.1723	0.1639	0.7879	0.8361	0.8277	0.8099	0.8309	0.9209	18432	
MSLBP (1,2)											
Y	16×16	0.5074	0.5932	0.3768	0.4068	0.4926	0.3901	0.4569	0.5126	196608	
	32×32	0.4586	0.5254	0.4304	0.4746	0.5414	0.4508	0.5130	0.5745	49152	
	64×64	0.4118	0.4682	0.4795	0.5318	0.5882	0.5027	0.5645	0.6405	12288	
Cb	16×16	0.3271	0.3840	0.6333	0.6160	0.6729	0.6229	0.6448	0.7243	196608	
	32×32	0.2332	0.2303	0.7134	0.7697	0.7668	0.7397	0.7681	0.8621	49152	
	64×64	0.1712	0.1744	0.7968	0.8256	0.8288	0.8101	0.8275	0.9170	12288	
Cr	16×16	0.2940	0.3718	0.6914	0.6282	0.7060	0.6554	0.6649	0.7429	196608	
	32×32	0.2206	0.2599	0.7467	0.7401	0.7794	0.7426	0.7606	0.8551	49152	
	64×64	0.1797	0.1818	0.7873	0.8182	0.8203	0.8003	0.8183	0.9177	12288	

Table A.5: Results for each channel on CASIA v1.0 dataset using MSLBP with radii of 1,2,3 and 4 (Feature concatenation)

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features Dimension
LBP(1)										
Y	16×16	0.4660	0.5435	0.4130	0.4565	0.5340	0.4308	0.5000	0.5711	98304
	32×32	0.4287	0.4838	0.4673	0.5162	0.5713	0.4880	0.5471	0.6197	24576
	64×64	0.3887	0.4401	0.5125	0.5599	0.6113	0.5332	0.5894	0.6664	6144
Cb	16×16	0.3371	0.3962	0.6180	0.6038	0.6629	0.6074	0.6327	0.6864	98304
	32×32	0.2727	0.3055	0.6754	0.6945	0.7273	0.6831	0.7118	0.7930	24576
	64×64	0.2460	0.2496	0.6947	0.7504	0.7540	0.7204	0.7523	0.8539	6144
Cr	16×16	0.3137	0.3908	0.6758	0.6092	0.6863	0.6378	0.6450	0.7028	98304
	32×32	0.2497	0.2932	0.7177	0.7068	0.7503	0.7103	0.7292	0.8078	24576
	64×64	0.2300	0.2484	0.7260	0.7516	0.7700	0.7373	0.7614	0.8612	6144
LBP(2)										
Y	16×16	0.5603	0.6647	0.3154	0.3353	0.4397	0.3242	0.3946	0.4528	98304
	32×32	0.5335	0.6311	0.3383	0.3689	0.4665	0.3507	0.4253	0.4847	24576
	64×64	0.4894	0.5793	0.3602	0.4207	0.5106	0.3868	0.4749	0.5258	6144
Cb	16×16	0.3272	0.3753	0.6197	0.6247	0.6728	0.6196	0.6495	0.7345	98304
	32×32	0.2360	0.2360	0.7113	0.7640	0.7640	0.7359	0.7642	0.8584	24576
	64×64	0.1931	0.1966	0.7636	0.8034	0.8069	0.7809	0.8054	0.9018	6144
Cr	16×16	0.3025	0.3637	0.6686	0.6363	0.6975	0.6513	0.6684	0.7363	98304
	32×32	0.2274	0.2541	0.7346	0.7459	0.7726	0.7389	0.7606	0.8567	24576
	64×64	0.1900	0.1777	0.7706	0.8223	0.8100	0.7948	0.8160	0.9094	6144
LBP(3)										
Y	16×16	0.5746	0.6914	0.2797	0.3086	0.4254	0.2918	0.3760	0.4264	98304
	32×32	0.5663	0.6856	0.2814	0.3144	0.4337	0.2946	0.3850	0.4366	24576
	64×64	0.5278	0.6423	0.3044	0.3577	0.4722	0.3270	0.4266	0.4568	6144
Cb	16×16	0.4059	0.4708	0.5476	0.5292	0.5941	0.5376	0.5633	0.6383	98304
	32×32	0.2953	0.3121	0.6381	0.6879	0.7047	0.6610	0.6979	0.7935	24576
	64×64	0.2264	0.2272	0.7223	0.7728	0.7736	0.7447	0.7721	0.8679	6144
Cr	16×16	0.3774	0.4525	0.5979	0.5475	0.6226	0.5702	0.5847	0.6500	98304
	32×32	0.2633	0.2974	0.6899	0.7026	0.7367	0.6945	0.7201	0.7983	24576
	64×64	0.2354	0.2228	0.7036	0.7772	0.7646	0.7377	0.7699	0.8662	6144
LBP(4)										
Y	16×16	0.5766	0.6883	0.2853	0.3117	0.4234	0.2960	0.3754	0.4012	98304
	32×32	0.5693	0.6999	0.2619	0.3001	0.4307	0.2766	0.3770	0.4149	24576
	64×64	0.5471	0.6671	0.2844	0.3329	0.4529	0.3057	0.4058	0.4366	6144
Cb	16×16	0.4660	0.5375	0.4762	0.4625	0.5340	0.4680	0.4989	0.5622	98304
	32×32	0.3358	0.3536	0.5774	0.6464	0.6642	0.6096	0.6576	0.7356	24576
	64×64	0.2524	0.2548	0.6853	0.7452	0.7476	0.7126	0.7468	0.8446	6144
Cr	16×16	0.4497	0.5219	0.4973	0.4781	0.5503	0.4846	0.5142	0.5778	98304
	32×32	0.3201	0.3526	0.6166	0.6474	0.6799	0.6302	0.6655	0.7425	24576
	64×64	0.2528	0.2534	0.6872	0.7466	0.7472	0.7139	0.7468	0.8401	6144

Table A.6: Results for each channel on CASIA v1.0 dataset using LBP with radii of 1,2,3 and 4 (Feature concatenation)

Table A.6 shows the results of LBP with radii of 1,2,3 and 4. For each version of LBP the features were concatenation to make a feature vector. The LBP with radius 2 in Cr channel with 64×64 block size achieved 81.6% accuracy.

Table A.7 illustrates the results of CSLBP, CSLBP-DCT and CSLBP with fusion of DCT coefficients. These features are stacked together and standard deviation is calculated to make a feature vector. In this table the highest accuracy achieved is 94.85% by CSLBP-DCT in Cr channel with 16×16 block size.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
CSLBP											
Y	16×16	0.4475	0.5128	0.3855	0.4872	0.5525	0.4292	0.5283	0.5007		16
	32×32	0.4569	0.5248	0.3857	0.4752	0.5431	0.4235	0.5157	0.5062		16
	64×64	0.4710	0.5489	0.3752	0.4511	0.5290	0.4080	0.4982	0.4691		16
Cb	16×16	0.2624	0.2986	0.6962	0.7014	0.7376	0.6949	0.7180	0.7777		16
	32×32	0.3057	0.3313	0.6252	0.6687	0.6943	0.6448	0.6829	0.7372		16
	64×64	0.3501	0.3846	0.5674	0.6154	0.6499	0.5896	0.6354	0.6692		16
Cr	16×16	0.2431	0.2643	0.7100	0.7357	0.7569	0.7224	0.7486	0.8092		16
	32×32	0.2749	0.2942	0.6677	0.7058	0.7251	0.6846	0.7163	0.7638		16
	64×64	0.3144	0.3449	0.6169	0.6551	0.6856	0.6343	0.6721	0.7091		16
CSLBP+DCT											
Y	16×16	0.2337	0.2378	0.7168	0.7622	0.7663	0.7376	0.7644	0.8254		272
	32×32	0.2973	0.3225	0.6463	0.6775	0.7027	0.6592	0.6920	0.7573		1040
	64×64	0.3248	0.3574	0.6033	0.6426	0.6752	0.6212	0.6604	0.6987		4112
Cb	16×16	0.0986	0.0850	0.8811	0.9150	0.9014	0.8969	0.9068	0.9433		272
	32×32	0.1247	0.1106	0.8557	0.8894	0.8753	0.8716	0.8828	0.9347		1040
	64×64	0.1530	0.1411	0.8142	0.8589	0.8470	0.8351	0.8526	0.9240		4112
Cr	16×16	0.0729	0.0858	0.9162	0.9142	0.9271	0.9145	0.9206	0.9569		272
	32×32	0.0907	0.0974	0.8941	0.9026	0.9093	0.8977	0.9058	0.9488		1040
	64×64	0.1077	0.1157	0.8760	0.8843	0.8923	0.8795	0.8890	0.9506		4112
CSLBP-DCT											
Y	16×16	0.3559	0.3857	0.5548	0.6143	0.6441	0.5816	0.6310	0.6809		225
	32×32	0.3769	0.4141	0.5199	0.5859	0.6231	0.5497	0.6078	0.6520		961
	64×64	0.4251	0.4764	0.4574	0.5236	0.5749	0.4867	0.5534	0.6079		3969
Cb	16×16	0.0888	0.0732	0.8931	0.9268	0.9112	0.9093	0.9178	0.9681		225
	32×32	0.0830	0.0633	0.8999	0.9367	0.9170	0.9177	0.9258	0.9764		961
	64×64	0.1036	0.0948	0.8776	0.9052	0.8964	0.8907	0.9001	0.9671		3969
Cr	16×16	0.0553	0.0482	0.9352	0.9518	0.9447	0.9431	0.9485	0.9791		225
	32×32	0.0726	0.0603	0.9137	0.9397	0.9274	0.9261	0.9326	0.9801		961
	64×64	0.0865	0.0846	0.8984	0.9154	0.9135	0.9065	0.9143	0.9712		3969

Table A.7: Results for each channel on CASIA v1.0 dataset using CSLBP and CSLBP+DCT. (Feature Stacking)

The features of MSLBP-DCT were extracted and stacked together to make a feature vector. Table A.8 illustrates the results of MSLBP-DCT on CASIA v1.0 dataset with radii 1,2,3 and 4. The fusing of radii 1, 2 and 3 achieved the highest accuracy of 97.34% in Cr channel with 32×32 block size.

Table A.9 shows the experimental results of LBP-DCT with radii 1,2,3 and 4 on CASIA v1.0 dataset. The LBP-DCT with radius 1 performed better among others radii and achieved an accuracy of 96.35% with only 196 features in Cr channel with 16×16 block size.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
MSLBP-DCT (1,2,3,4)											
Y	16×16	0.2672	0.2780	0.6745	0.7220	0.7328	0.6940	0.7263	0.7873	504	
	32×32	0.3167	0.3321	0.6012	0.6679	0.6833	0.6308	0.6767	0.7367	2936	
	64×64	0.3594	0.3936	0.5462	0.6064	0.6406	0.5739	0.6260	0.6833	13944	
Cb	16×16	0.0385	0.0267	0.9554	0.9733	0.9615	0.9641	0.9670	0.9931	504	
	32×32	0.0378	0.0309	0.9545	0.9691	0.9622	0.9615	0.9652	0.9926	2936	
	64×64	0.0482	0.0540	0.9453	0.9460	0.9518	0.9455	0.9494	0.9867	13944	
Cr	16×16	0.0353	0.0230	0.9587	0.9770	0.9647	0.9674	0.9698	0.9886	504	
	32×32	0.0372	0.0234	0.9566	0.9766	0.9628	0.9663	0.9694	0.9929	2936	
	64×64	0.0448	0.0504	0.9491	0.9496	0.9552	0.9491	0.9529	0.9889	13944	
MSLBP-DCT (1,2,3)											
Y	16×16	0.2621	0.2703	0.6818	0.7297	0.7379	0.7038	0.7352	0.7941	440	
	32×32	0.2969	0.3073	0.6285	0.6927	0.7031	0.6581	0.6994	0.7510	2360	
	64×64	0.3589	0.3881	0.5439	0.6119	0.6411	0.5736	0.6285	0.6910	10808	
Cb	16×16	0.0356	0.0255	0.9565	0.9745	0.9644	0.9653	0.9686	0.9922	440	
	32×32	0.0364	0.0333	0.9573	0.9667	0.9636	0.9618	0.9653	0.9918	2360	
	64×64	0.0514	0.0537	0.9422	0.9463	0.9486	0.9440	0.9485	0.9882	10808	
Cr	16×16	0.0347	0.0270	0.9607	0.9730	0.9653	0.9666	0.9693	0.9904	440	
	32×32	0.0319	0.0195	0.9623	0.9805	0.9681	0.9712	0.9734	0.9947	2360	
	64×64	0.0413	0.0482	0.9504	0.9518	0.9587	0.9507	0.9549	0.9888	10808	
MSLBP-DCT (1,2)											
Y	16×16	0.2561	0.2641	0.6845	0.7359	0.7439	0.7085	0.7409	0.8003	340	
	32×32	0.3050	0.3191	0.6130	0.6809	0.6950	0.6433	0.6890	0.7498	1684	
	64×64	0.3480	0.3758	0.5638	0.6242	0.6520	0.5910	0.6404	0.6966	7444	
Cb	16×16	0.0376	0.0324	0.9563	0.9676	0.9624	0.9617	0.9646	0.9920	340	
	32×32	0.0394	0.0344	0.9536	0.9656	0.9606	0.9590	0.9623	0.9904	1684	
	64×64	0.0505	0.0491	0.9416	0.9509	0.9495	0.9460	0.9501	0.9878	7444	
Cr	16×16	0.0364	0.0255	0.9559	0.9745	0.9636	0.9648	0.9682	0.9908	340	
	32×32	0.0353	0.0201	0.9595	0.9799	0.9647	0.9694	0.9716	0.9923	1684	
	64×64	0.0508	0.0469	0.9392	0.9531	0.9492	0.9457	0.9501	0.9894	7444	

Table A.8: Results for each channel on CASIA v1.0 dataset using MSLBP-DCT with radii of 1,2,3 and 4. (Feature Stacking)

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
LBP-DCT (1)											
Y	16×16	0.2546	0.2720	0.6916	0.7280	0.7454	0.7073	0.7371	0.7989	196	
	32×32	0.3057	0.3138	0.6125	0.6862	0.6943	0.6459	0.6908	0.7568	900	
	64×64	0.3450	0.3675	0.5678	0.6325	0.6550	0.5974	0.6458	0.7010	3844	
Cb	16×16	0.0513	0.0337	0.9388	0.9663	0.9487	0.9521	0.9566	0.9886	196	
	32×32	0.0460	0.0464	0.9456	0.9536	0.9540	0.9494	0.9536	0.9885	900	
	64×64	0.0592	0.0690	0.9311	0.9310	0.9408	0.9307	0.9357	0.9827	3844	
Cr	16×16	0.0419	0.0288	0.9498	0.9712	0.9581	0.9601	0.9635	0.9868	196	
	32×32	0.0436	0.0368	0.9491	0.9632	0.9564	0.9559	0.9595	0.9913	900	
	64×64	0.0598	0.0555	0.9305	0.9445	0.9402	0.9370	0.9419	0.9872	3844	
LBP-DCT (2)											
Y	16×16	0.3258	0.3561	0.6036	0.6439	0.6742	0.6213	0.6605	0.7135	144	
	32×32	0.3354	0.3564	0.5753	0.6436	0.6646	0.6069	0.6566	0.6979	784	
	64×64	0.3748	0.4029	0.5168	0.5971	0.6252	0.5520	0.6139	0.6612	3600	
Cb	16×16	0.0524	0.0392	0.9365	0.9608	0.9476	0.9483	0.9537	0.9861	144	
	32×32	0.0555	0.0562	0.9351	0.9438	0.9445	0.9391	0.9445	0.9856	784	
	64×64	0.0731	0.0658	0.9139	0.9342	0.9269	0.9237	0.9305	0.9808	3600	
Cr	16×16	0.0590	0.0411	0.9272	0.9589	0.9410	0.9423	0.9486	0.9824	144	
	32×32	0.0615	0.0355	0.9267	0.9645	0.9385	0.9446	0.9496	0.9890	784	
	64×64	0.0778	0.0574	0.9085	0.9426	0.9222	0.9249	0.9317	0.9813	3600	
LBP-DCT (3)											
Y	16×16	0.3537	0.3969	0.5690	0.6031	0.6463	0.5845	0.6276	0.6756	100	
	32×32	0.3598	0.3895	0.5432	0.6105	0.6402	0.5724	0.6265	0.6634	676	
	64×64	0.3976	0.4360	0.4879	0.5640	0.6024	0.5209	0.5865	0.6233	3364	
Cb	16×16	0.0667	0.0492	0.9217	0.9508	0.9333	0.9356	0.9414	0.9767	100	
	32×32	0.0585	0.0447	0.9304	0.9553	0.9415	0.9422	0.9473	0.9843	676	
	64×64	0.0783	0.0856	0.9118	0.9144	0.9217	0.9128	0.9185	0.9733	3364	
Cr	16×16	0.0604	0.0485	0.9289	0.9515	0.9396	0.9398	0.9450	0.9812	100	
	32×32	0.0572	0.0567	0.9326	0.9433	0.9428	0.9372	0.9425	0.9845	676	
	64×64	0.0692	0.0682	0.9211	0.9318	0.9308	0.9258	0.9315	0.9816	3364	
LBP-DCT (4)											
Y	16×16	0.3696	0.4181	0.5502	0.5819	0.6304	0.5625	0.6079	0.6409	64	
	32×32	0.3788	0.4142	0.5163	0.5858	0.6212	0.5463	0.6053	0.6545	576	
	64×64	0.4008	0.4446	0.4851	0.5554	0.5992	0.5170	0.5813	0.6215	3136	
Cb	16×16	0.1084	0.1134	0.8710	0.8866	0.8916	0.8783	0.8893	0.9459	64	
	32×32	0.0819	0.0693	0.9022	0.9307	0.9181	0.9159	0.9235	0.9736	576	
	64×64	0.0821	0.1080	0.9063	0.8920	0.9179	0.8983	0.9059	0.9655	3136	
Cr	16×16	0.1040	0.1058	0.8755	0.8942	0.8960	0.8843	0.8954	0.9423	64	
	32×32	0.0791	0.0691	0.9069	0.9309	0.9209	0.9184	0.9254	0.9738	576	
	64×64	0.0902	0.0834	0.8934	0.9166	0.9098	0.9045	0.9129	0.9697	3136	

Table A.9: Results for each channel on CASIA v1.0 dataset using LBP-DCT with radii of 1,2,3 and 4 (Feature stacking)

Table A.10 illustrates the experiment results on CASIA v1.0 by fusion of various LBP features with different radii. In these experiments the highest accuracy is 93.46% in Cb channel with 16×16 block size and with the fusion of all four radii.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
MSLBP (1,2,3,4)											
Y	16×16	0.3733	0.4142	0.5345	0.5858	0.6267	0.5576	0.6094	0.6837	1024	
	32×32	0.3809	0.4161	0.5105	0.5839	0.6191	0.5431	0.6036	0.6709	1024	
	64×64	0.3919	0.4337	0.4962	0.5663	0.6081	0.5274	0.5902	0.6424	1024	
Cb	16×16	0.0711	0.0595	0.9154	0.9405	0.9289	0.9274	0.9346	0.9809	1024	
	32×32	0.1008	0.0774	0.8765	0.9226	0.8992	0.8983	0.9090	0.9655	1024	
	64×64	0.1363	0.1151	0.8364	0.8849	0.8637	0.8584	0.8712	0.9430	1024	
Cr	16×16	0.0717	0.0604	0.9139	0.9396	0.9283	0.9263	0.9334	0.9782	1024	
	32×32	0.1025	0.0839	0.8748	0.9161	0.8975	0.8944	0.9053	0.9638	1024	
	64×64	0.1208	0.0969	0.8545	0.9031	0.8792	0.8771	0.8899	0.9484	1024	
MSLBP (1,2,3)											
Y	16×16	0.3724	0.4084	0.5380	0.5916	0.6276	0.5620	0.6140	0.6870	768	
	32×32	0.3684	0.4005	0.5348	0.5995	0.6316	0.5630	0.6178	0.6669	768	
	64×64	0.3865	0.4201	0.5048	0.5799	0.6135	0.5384	0.6004	0.6535	768	
Cb	16×16	0.0776	0.0656	0.9090	0.9344	0.9224	0.9213	0.9280	0.9770	768	
	32×32	0.1013	0.0793	0.8752	0.9207	0.8987	0.8970	0.9080	0.9601	768	
	64×64	0.1319	0.1069	0.8402	0.8931	0.8681	0.8649	0.8788	0.9428	768	
Cr	16×16	0.0774	0.0559	0.9085	0.9442	0.9226	0.9258	0.9333	0.9746	768	
	32×32	0.1054	0.0886	0.8736	0.9114	0.8946	0.8915	0.9013	0.9591	768	
	64×64	0.1202	0.1012	0.8547	0.8988	0.8798	0.8756	0.8879	0.9447	768	
MSLBP (1,2)											
Y	16×16	0.3563	0.3896	0.5560	0.6104	0.6437	0.5807	0.6290	0.6959	512	
	32×32	0.3646	0.3923	0.5382	0.6077	0.6354	0.5681	0.6230	0.6902	512	
	64×64	0.3735	0.4162	0.5294	0.5838	0.6265	0.5530	0.6084	0.6694	512	
Cb	16×16	0.1066	0.0832	0.8701	0.9168	0.8934	0.8918	0.9032	0.9647	512	
	32×32	0.1304	0.1144	0.8434	0.8856	0.8696	0.8626	0.8757	0.9407	512	
	64×64	0.1634	0.1453	0.7999	0.8547	0.8366	0.8245	0.8444	0.9239	512	
Cr	16×16	0.1150	0.0815	0.8631	0.9185	0.8850	0.8894	0.9002	0.9615	512	
	32×32	0.1287	0.1231	0.8492	0.8769	0.8713	0.8626	0.8747	0.9443	512	
	64×64	0.1476	0.1318	0.8197	0.8682	0.8524	0.8425	0.8589	0.9155	512	

Table A.10: Results for each channel on CASIA v1.0 dataset using MSLBP with radii 1,2,3 and 4. (Feature Stacking)

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
LBP (1)											
Y	16×16	0.3617	0.3960	0.5511	0.6040	0.6383	0.5721	0.6214	0.6863	256	
	32×32	0.3583	0.3960	0.5552	0.6040	0.6417	0.5760	0.6247	0.6736	256	
	64×64	0.3622	0.4076	0.5564	0.5924	0.6378	0.5734	0.6186	0.6697	256	
Cb	16×16	0.1451	0.1316	0.8289	0.8684	0.8549	0.8469	0.8609	0.9298	256	
	32×32	0.1686	0.1645	0.7975	0.8355	0.8314	0.8153	0.8328	0.9040	256	
	64×64	0.1852	0.1843	0.7792	0.8157	0.8148	0.7961	0.8154	0.8864	256	
Cr	16×16	0.1462	0.1395	0.8231	0.8605	0.8538	0.8409	0.8566	0.9290	256	
	32×32	0.1832	0.1806	0.7785	0.8194	0.8168	0.7968	0.8170	0.8859	256	
	64×64	0.2015	0.1927	0.7507	0.8073	0.7985	0.7760	0.8008	0.8661	256	
LBP (2)											
Y	16×16	0.4123	0.4540	0.4559	0.5460	0.5877	0.4942	0.5697	0.6033	256	
	32×32	0.4277	0.4777	0.4353	0.5223	0.5723	0.4722	0.5516	0.5890	256	
	64×64	0.4411	0.5034	0.4338	0.4966	0.5589	0.4605	0.5325	0.5654	256	
Cb	16×16	0.1438	0.1127	0.8257	0.8873	0.8562	0.8550	0.8708	0.9382	256	
	32×32	0.1685	0.1534	0.7946	0.8466	0.8316	0.8191	0.8382	0.9142	256	
	64×64	0.2034	0.1776	0.7436	0.8224	0.7966	0.7789	0.8065	0.8769	256	
Cr	16×16	0.1306	0.1049	0.8416	0.8951	0.8694	0.8665	0.8799	0.9346	256	
	32×32	0.1697	0.1592	0.7899	0.8408	0.8303	0.8138	0.8345	0.9125	256	
	64×64	0.1670	0.1645	0.7966	0.8355	0.8330	0.8145	0.8335	0.8975	256	
LBP (3)											
Y	16×16	0.4415	0.4974	0.4167	0.5026	0.5585	0.4533	0.5363	0.5638	256	
	32×32	0.4424	0.5082	0.4101	0.4918	0.5576	0.4451	0.5329	0.5564	256	
	64×64	0.4601	0.5315	0.4062	0.4685	0.5399	0.4335	0.5101	0.5123	256	
Cb	16×16	0.1295	0.1197	0.8417	0.8803	0.8705	0.8597	0.8743	0.9433	256	
	32×32	0.1435	0.1304	0.8248	0.8696	0.8565	0.8444	0.8600	0.9265	256	
	64×64	0.1877	0.1736	0.7668	0.8264	0.8123	0.7941	0.8179	0.8857	256	
Cr	16×16	0.1059	0.0982	0.8732	0.9018	0.8941	0.8869	0.8976	0.9477	256	
	32×32	0.1538	0.1476	0.8160	0.8524	0.8462	0.8327	0.8489	0.9231	256	
	64×64	0.1806	0.1533	0.7767	0.8467	0.8194	0.8091	0.8309	0.8957	256	
LBP (4)											
Y	16×16	0.4866	0.5685	0.3589	0.4315	0.5134	0.3914	0.4829	0.4832	256	
	32×32	0.4758	0.5547	0.3918	0.4453	0.5242	0.4156	0.4923	0.4975	256	
	64×64	0.4897	0.5725	0.3674	0.4275	0.5103	0.3934	0.4766	0.4746	256	
Cb	16×16	0.1466	0.1461	0.8272	0.8539	0.8534	0.8394	0.8540	0.9259	256	
	32×32	0.1715	0.1631	0.7931	0.8369	0.8285	0.8141	0.8324	0.9072	256	
	64×64	0.2028	0.1777	0.7465	0.8223	0.7972	0.7806	0.8067	0.8798	256	
Cr	16×16	0.1479	0.1475	0.8232	0.8525	0.8521	0.8371	0.8524	0.9171	256	
	32×32	0.1562	0.1404	0.8112	0.8596	0.8438	0.8346	0.8510	0.9151	256	
	64×64	0.2060	0.1982	0.7509	0.8018	0.7940	0.7748	0.7975	0.8729	256	

Table A.11: Results for each channel on CASIA v1.0 dataset using LBP with radii of 1,2,3 and 4 (Feature stacking)

Table A.11 shows the results of LBP features on CASIA v1.0 dataset. In all the four types of LBP features, the LBP with radius 3 performed better and achieved 89.76% accuracy with 0.94 AUC in Cr channel with 16×16 block size.

Table A.12 illustrates the results of LBP-DCT and MSLBP-DCT with 8, 12 and 16 neighbors. The LBP-DCT with 8 and 12 neighbors combined performed better achieved an accuracy of 96.88% with AUC 0.99 in Cr channel using 16×16 block size with only 296 features.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
MSLBP-DCT ($r=1, p=8$ and $r=2, p=12$, $r=3, p=16$)											
Y	16×16	0.2600	0.2740	0.6853	0.7260	0.7400	0.7027	0.7329	0.7955		360
	32×32	0.3192	0.3359	0.6066	0.6641	0.6808	0.6326	0.6736	0.7409		2152
	64×64	0.3510	0.3839	0.5623	0.6161	0.6490	0.5866	0.6364	0.6904		10344
Cb	16×16	0.0395	0.0308	0.9536	0.9692	0.9605	0.9612	0.9646	0.9921		360
	32×32	0.0343	0.0373	0.9601	0.9627	0.9657	0.9610	0.9641	0.9916		2152
	64×64	0.0524	0.0577	0.9378	0.9423	0.9476	0.9399	0.9451	0.9844		10344
Cr	16×16	0.0334	0.0306	0.9617	0.9694	0.9666	0.9653	0.9682	0.9903		360
	32×32	0.0343	0.0296	0.9597	0.9704	0.9657	0.9649	0.9676	0.9917		2152
	64×64	0.0418	0.0476	0.9518	0.9524	0.9582	0.9516	0.9554	0.9867		10344
MSLBP-DCT ($r=1, p=8$ and $r=2, p=12$)											
Y	16×16	0.2640	0.2808	0.6780	0.7192	0.7360	0.6967	0.7292	0.7875		296
	32×32	0.3137	0.3379	0.6101	0.6621	0.6863	0.6320	0.6746	0.7466		1576
	64×64	0.3554	0.3837	0.5559	0.6163	0.6446	0.5840	0.6333	0.6883		7208
Cb	16×16	0.0382	0.0284	0.9545	0.9716	0.9618	0.9627	0.9657	0.9908		296
	32×32	0.0407	0.0360	0.9529	0.9640	0.9593	0.9583	0.9619	0.9895		1576
	64×64	0.0617	0.0562	0.9300	0.9438	0.9383	0.9364	0.9416	0.9866		7208
Cr	16×16	0.0361	0.0259	0.9573	0.9741	0.9639	0.9655	0.9688	0.9908		296
	32×32	0.0425	0.0290	0.9501	0.9710	0.9575	0.9602	0.9641	0.9912		1576
	64×64	0.0573	0.0554	0.9319	0.9446	0.9427	0.9378	0.9432	0.9859		7208
LBP-DCT ($r=3, p=16$)											
Y	16×16	0.3683	0.4074	0.5458	0.5926	0.6317	0.5641	0.6121	0.6432		64
	32×32	0.3688	0.4052	0.5339	0.5948	0.6312	0.5604	0.6154	0.6662		576
	64×64	0.4042	0.4457	0.4798	0.5543	0.5958	0.5129	0.5805	0.6177		3136
Cb	16×16	0.1103	0.0910	0.8665	0.9090	0.8897	0.8866	0.8974	0.9504		64
	32×32	0.0959	0.0829	0.8876	0.9171	0.9041	0.9019	0.9105	0.9652		576
	64×64	0.1010	0.1025	0.8786	0.8975	0.8990	0.8869	0.8976	0.9600		3136
Cr	16×16	0.1097	0.1041	0.8694	0.8959	0.8903	0.8819	0.8928	0.9467		64
	32×32	0.0961	0.0840	0.8856	0.9160	0.9039	0.8994	0.9090	0.9689		576
	64×64	0.1099	0.0962	0.8671	0.9038	0.8901	0.8842	0.8955	0.9619		3136
LBP-DCT ($r=2, p=12$)											
Y	16×16	0.3714	0.4146	0.5347	0.5854	0.6286	0.5580	0.6117	0.6426		100
	32×32	0.3620	0.3877	0.5415	0.6123	0.6380	0.5728	0.6261	0.6768		676
	64×64	0.3938	0.4321	0.4991	0.5679	0.6062	0.5293	0.5914	0.6278		3364
Cb	16×16	0.0524	0.0519	0.9393	0.9481	0.9476	0.9435	0.9480	0.9821		100
	32×32	0.0732	0.0526	0.9121	0.9474	0.9268	0.9289	0.9357	0.9787		676
	64×64	0.0812	0.0796	0.9062	0.9204	0.9188	0.9126	0.9191	0.9755		3364
Cr	16×16	0.0565	0.0468	0.9320	0.9532	0.9435	0.9422	0.9479	0.9772		100
	32×32	0.0619	0.0605	0.9251	0.9395	0.9381	0.9320	0.9387	0.9781		676
	64×64	0.0860	0.0674	0.8963	0.9326	0.9140	0.9135	0.9217	0.9763		3364

Table A.12: Results for each channel on CASIA v1.0 dataset using LBP-DCT and MSLBP-DCT with neighbors of 8,12 and 16 (Feature stacking)

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features Dimension
MSLBP ($r=1$, $p=8$ and $r=2$, $p=12$)										
Y	16×16	0.3756	0.4238	0.5412	0.5762	0.6244	0.5573	0.6033	0.6755	4352
	32×32	0.3793	0.4201	0.5298	0.5799	0.6207	0.5498	0.6008	0.6691	4352
	64×64	0.3881	0.4306	0.5114	0.5694	0.6119	0.5374	0.5949	0.6565	4352
Cb	16×16	0.0798	0.0527	0.9000	0.9473	0.9202	0.9224	0.9323	0.9792	4352
	32×32	0.0938	0.0679	0.8848	0.9321	0.9062	0.9070	0.9189	0.9740	4352
	64×64	0.1112	0.0829	0.8676	0.9171	0.8888	0.8910	0.9019	0.9647	4352
Cr	16×16	0.0925	0.0699	0.8878	0.9301	0.9075	0.9077	0.9169	0.9691	4352
	32×32	0.1105	0.0899	0.8658	0.9101	0.8895	0.8869	0.8983	0.9561	4352
	64×64	0.1049	0.0992	0.8775	0.9008	0.8951	0.8881	0.8973	0.9601	4352
LBP ($r=2$, $p=12$)										
Y	16×16	0.3999	0.4536	0.4989	0.5464	0.6001	0.5200	0.5769	0.6335	4096
	32×32	0.4370	0.4929	0.4500	0.5071	0.5630	0.4755	0.5392	0.6122	4096
	64×64	0.4262	0.4833	0.4782	0.5167	0.5738	0.4957	0.5500	0.6063	4096
Cb	16×16	0.0807	0.0617	0.9037	0.9383	0.9193	0.9202	0.9280	0.9757	4096
	32×32	0.0963	0.0706	0.8846	0.9294	0.9037	0.9058	0.9154	0.9688	4096
	64×64	0.1106	0.0958	0.8686	0.9042	0.8894	0.8854	0.8960	0.9595	4096
Cr	16×16	0.0977	0.0869	0.8847	0.9131	0.9023	0.8981	0.9082	0.9633	4096
	32×32	0.1091	0.0990	0.8686	0.9010	0.8909	0.8838	0.8946	0.9512	4096
	64×64	0.1093	0.0964	0.8693	0.9036	0.8907	0.8857	0.8966	0.9549	4096

Table A.13: Results for each channel on CASIA v1.0 dataset using LBP and MSLBP with neighbors of 8 and 12 (Feature stacking)

Table A.13 shows the LBP and MSLBP results with 8 and 12 neighbors. The combination of these two neighbors features achieved an accuracy of 93.23% using Cb channel with 16×16 block size.

Table A.14 illustrates the results of LPQ-DCT and MSLPQ-DCT features on CASIA v1.0 dataset. The MSLPQ-DCT performed better and achieved an accuracy of 96.94% in Cr channel with AUC 0.99 using 16×16 block size.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
MSLPQ-DCT (3,5)											
Y	16×16	0.2394	0.2588	0.7140	0.7412	0.7606	0.7254	0.7515	0.8217	340	
	32×32	0.2658	0.2858	0.6834	0.7142	0.7342	0.6971	0.7246	0.7970	1684	
	64×64	0.3024	0.3242	0.6316	0.6758	0.6976	0.6513	0.6879	0.7733	7444	
Cb	16×16	0.0415	0.0349	0.9507	0.9651	0.9585	0.9576	0.9611	0.9904	340	
	32×32	0.0446	0.0454	0.9474	0.9546	0.9554	0.9506	0.9547	0.9909	1684	
	64×64	0.0500	0.0688	0.9439	0.9313	0.9500	0.9371	0.9420	0.9833	7444	
Cr	16×16	0.0349	0.0250	0.9585	0.9750	0.9651	0.9666	0.9694	0.9909	340	
	32×32	0.0359	0.0290	0.9571	0.9710	0.9641	0.9638	0.9669	0.9927	1684	
	64×64	0.0440	0.0545	0.9504	0.9455	0.9560	0.9478	0.9514	0.9885	7444	
LPQ-DCT (5)											
Y	16×16	0.3158	0.3404	0.6152	0.6597	0.6842	0.6356	0.6732	0.7308	144	
	32×32	0.3336	0.3626	0.5942	0.6374	0.6664	0.6127	0.6526	0.7080	784	
	64×64	0.3656	0.3975	0.5444	0.6025	0.6344	0.5710	0.6213	0.6675	3600	
Cb	16×16	0.0605	0.0498	0.9303	0.9502	0.9395	0.9396	0.9448	0.9853	144	
	32×32	0.0639	0.0592	0.9251	0.9408	0.9361	0.9323	0.9387	0.9838	784	
	64×64	0.0939	0.0959	0.8912	0.9041	0.9061	0.8968	0.9052	0.9696	3600	
Cr	16×16	0.0517	0.0483	0.9400	0.9517	0.9483	0.9457	0.9500	0.9866	144	
	32×32	0.0588	0.0521	0.9320	0.9479	0.9412	0.9395	0.9443	0.9853	784	
	64×64	0.0852	0.0871	0.9039	0.9129	0.9148	0.9077	0.9141	0.9730	3600	
LPQ-DCT (3)											
Y	16×16	0.2459	0.2670	0.7045	0.7330	0.7541	0.7176	0.7461	0.8086	196	
	32×32	0.2607	0.2906	0.6897	0.7094	0.7393	0.6986	0.7260	0.7988	900	
	64×64	0.2864	0.3027	0.6516	0.6973	0.7136	0.6732	0.7068	0.7790	3844	
Cb	16×16	0.0645	0.0494	0.9250	0.9506	0.9355	0.9372	0.9428	0.9824	196	
	32×32	0.0594	0.0591	0.9289	0.9409	0.9406	0.9344	0.9410	0.9824	900	
	64×64	0.0666	0.0704	0.9222	0.9296	0.9334	0.9257	0.9315	0.9798	3844	
Cr	16×16	0.0526	0.0376	0.9387	0.9624	0.9474	0.9501	0.9547	0.9876	196	
	32×32	0.0616	0.0440	0.9259	0.9560	0.9384	0.9403	0.9462	0.9877	900	
	64×64	0.0522	0.0554	0.9376	0.9446	0.9478	0.9409	0.9460	0.9870	3844	

Table A.14: Results for each channel on CASIA v1.0 dataset using LPQ-DCT and MSLPQ-DCT. (Feature stacking)

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
MSLPQ (3,5)											
Y	16×16	0.3543	0.3832	0.5596	0.6168	0.6457	0.5842	0.6324	0.6947		512
	32×32	0.3822	0.4132	0.5104	0.5868	0.6178	0.5448	0.6065	0.6642		512
	64×64	0.4050	0.4491	0.4943	0.5509	0.5950	0.5196	0.5767	0.6164		512
Cb	16×16	0.0802	0.0498	0.9045	0.9502	0.9198	0.9261	0.9327	0.9822		512
	32×32	0.0749	0.0507	0.9108	0.9493	0.9251	0.9288	0.9350	0.9770		512
	64×64	0.1254	0.0888	0.8480	0.9112	0.8746	0.8779	0.8909	0.9526		512
Cr	16×16	0.0594	0.0385	0.9297	0.9615	0.9406	0.9448	0.9501	0.9877		512
	32×32	0.0742	0.0482	0.9118	0.9518	0.9258	0.9308	0.9373	0.9808		512
	64×64	0.1206	0.0875	0.8543	0.9125	0.8794	0.8814	0.8938	0.9632		512
LPQ (5)											
Y	16×16	0.4377	0.5048	0.4346	0.4952	0.5623	0.4579	0.5325	0.5549		256
	32×32	0.4631	0.5361	0.3720	0.4640	0.5369	0.4106	0.5104	0.5214		256
	64×64	0.4699	0.5468	0.3745	0.4532	0.5301	0.4077	0.5003	0.5009		256
Cb	16×16	0.1046	0.0649	0.8733	0.9351	0.8954	0.9026	0.9125	0.9655		256
	32×32	0.1319	0.0860	0.8371	0.9140	0.8681	0.8734	0.8881	0.9484		256
	64×64	0.1816	0.1501	0.7728	0.8499	0.8184	0.8090	0.8317	0.8989		256
Cr	16×16	0.0907	0.0463	0.8917	0.9537	0.9093	0.9212	0.9287	0.9702		256
	32×32	0.1179	0.0759	0.8556	0.9241	0.8821	0.8881	0.9005	0.9542		256
	64×64	0.1487	0.1253	0.8199	0.8747	0.8513	0.8456	0.8613	0.9237		256
LPQ (3)											
Y	16×16	0.3523	0.3824	0.5639	0.6176	0.6477	0.5869	0.6332	0.6949		256
	32×32	0.3789	0.4194	0.5182	0.5806	0.6211	0.5458	0.6038	0.6521		256
	64×64	0.3979	0.4351	0.4817	0.5649	0.6021	0.5185	0.5873	0.6298		256
Cb	16×16	0.0803	0.0700	0.9057	0.9300	0.9197	0.9171	0.9254	0.9699		256
	32×32	0.1189	0.0799	0.8542	0.9201	0.8811	0.8848	0.8972	0.9528		256
	64×64	0.1445	0.1154	0.8213	0.8846	0.8555	0.8507	0.8672	0.9320		256
Cr	16×16	0.0835	0.0548	0.8986	0.9452	0.9165	0.9207	0.9288	0.9755		256
	32×32	0.1028	0.0769	0.8742	0.9231	0.8972	0.8975	0.9086	0.9614		256
	64×64	0.1473	0.1063	0.8181	0.8937	0.8527	0.8536	0.8703	0.9410		256

Table A.15: Results for each channel on CASIA v1.0 dataset using LPQ and MSLPQ. (Feature stacking)

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
SLBP+DCT											
Y	16×16	0.2821	0.3001	0.6579	0.6999	0.7179	0.6775	0.7101	0.7949		512
	32×32	0.2961	0.3177	0.6371	0.6823	0.7039	0.6563	0.6934	0.7579		1280
	64×64	0.3334	0.3665	0.5953	0.6335	0.6666	0.6120	0.6519	0.7020		4352
Cb	16×16	0.1117	0.1026	0.8685	0.8974	0.8883	0.8818	0.8919	0.9484		512
	32×32	0.1178	0.1154	0.8603	0.8846	0.8822	0.8718	0.8830	0.9458		1280
	64×64	0.1324	0.1272	0.8412	0.8728	0.8676	0.8559	0.8699	0.9366		4352
Cr	16×16	0.1042	0.0961	0.8775	0.9039	0.8958	0.8894	0.8994	0.9542		512
	32×32	0.1046	0.0937	0.8744	0.9063	0.8954	0.8897	0.9005	0.9535		1280
	64×64	0.1052	0.1113	0.8774	0.8887	0.8948	0.8828	0.8919	0.9514		4352
SLBP											
Y	16×16	0.3777	0.4185	0.5230	0.5815	0.6223	0.5499	0.6064	0.6567		256
	32×32	0.3939	0.4345	0.5026	0.5655	0.6061	0.5306	0.5887	0.6352		256
	64×64	0.4160	0.4633	0.4707	0.5367	0.5840	0.5007	0.5654	0.6064		256
Cb	16×16	0.1636	0.1699	0.8033	0.8301	0.8364	0.8163	0.8343	0.9074		256
	32×32	0.1958	0.1790	0.7586	0.8210	0.8042	0.7873	0.8111	0.8878		256
	64×64	0.1983	0.1824	0.7519	0.8176	0.8017	0.7820	0.8075	0.8767		256
Cr	16×16	0.1672	0.1456	0.7934	0.8544	0.8328	0.8214	0.8410	0.9114		256
	32×32	0.1864	0.1739	0.7717	0.8261	0.8136	0.7972	0.8203	0.8949		256
	64×64	0.2114	0.1920	0.7365	0.8080	0.7886	0.7694	0.7968	0.8780		256

Table A.16: Results for each channel on CASIA v1.0 dataset using SLBP and SLBP+DCT. (Feature stacking)

Table A.15 shows the results of LPQ and MSLPQ using CASIA v1.0 dataset. The MSLPQ performed better than LPQ features alone and achieved 95% accuracy in Cr channel using 16×16 block size.

The results of SLBP and SLBP with DCT are illustrated in Table A.16. The SLBP fusing with DCT performed better than SLBP on CASIA v1.0 dataset and achieved 90% accuracy in Cr channel using 32×32 block size.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
WLD-DCT											
Y	16×16	0.2687	0.2772	0.6712	0.7228	0.7313	0.6952	0.7277	0.7781	225	
	32×32	0.2931	0.3038	0.6328	0.6962	0.7069	0.6612	0.7017	0.7474	961	
	64×64	0.3314	0.3583	0.5900	0.6417	0.6686	0.6139	0.6568	0.6854	3969	
Cb	16×16	0.1174	0.0978	0.8583	0.9022	0.8826	0.8794	0.8916	0.9249	225	
	32×32	0.1570	0.1314	0.8085	0.8686	0.8430	0.8362	0.8537	0.9144	961	
	64×64	0.1743	0.1644	0.7877	0.8356	0.8257	0.8100	0.8310	0.9072	3969	
Cr	16×16	0.0811	0.0863	0.9047	0.9137	0.9189	0.9086	0.9160	0.9488	225	
	32×32	0.1122	0.1133	0.8684	0.8867	0.8878	0.8764	0.8871	0.9434	961	
	64×64	0.1084	0.1058	0.8721	0.8942	0.8916	0.8826	0.8925	0.9553	3969	
WLD											
Y	16×16	0.4239	0.4916	0.4941	0.5084	0.5761	0.4974	0.5449	0.5832	225	
	32×32	0.4310	0.4959	0.4667	0.5041	0.5690	0.4825	0.5410	0.5615	961	
	64×64	0.4434	0.5065	0.4457	0.4935	0.5566	0.4666	0.5302	0.5537	3969	
Cb	16×16	0.2523	0.2532	0.6875	0.7468	0.7477	0.7154	0.7474	0.7974	225	
	32×32	0.2400	0.2327	0.7019	0.7673	0.7600	0.7316	0.7621	0.8369	961	
	64×64	0.2489	0.2431	0.6949	0.7569	0.7511	0.7227	0.7526	0.8393	3969	
Cr	16×16	0.2005	0.2112	0.7608	0.7888	0.7995	0.7719	0.7941	0.8470	225	
	32×32	0.1738	0.1778	0.7930	0.8222	0.8262	0.8064	0.8251	0.8795	961	
	64×64	0.1837	0.1669	0.7781	0.8331	0.8163	0.8033	0.8236	0.8977	3969	

Table A.17: Results for each channel on CASIA v1.0 dataset using WLD and WLD-DCT. (Feature stacking)

Table A.17 illustrates the results of WLD and WLD-DCT on CASIA v1.0 dataset. The WLD-DCT performed better than WLD in all channels and block sizes. The WLD-DCT achieved the highest accuracy 91.6% in Cr channel using 16×16 block size.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
CSLBP											
Y	16×16	0.4226	0.3214	0.7637	0.6786	0.5774	0.7184	0.6448	0.6537	16	
	32×32	0.4372	0.3333	0.7658	0.6667	0.5628	0.7128	0.6336	0.6422	16	
	64×64	0.4402	0.3300	0.7540	0.6700	0.5598	0.7093	0.6333	0.6410	16	
Cb	16×16	0.1437	0.0840	0.8993	0.9160	0.8563	0.9075	0.8912	0.9539	16	
	32×32	0.1667	0.1163	0.8866	0.8837	0.8333	0.8850	0.8633	0.9330	16	
	64×64	0.2064	0.1569	0.8639	0.8431	0.7936	0.8533	0.8238	0.8965	16	
Cr	16×16	0.1348	0.0710	0.9040	0.9290	0.8652	0.9163	0.9020	0.9620	16	
	32×32	0.1442	0.0933	0.9002	0.9067	0.8558	0.9034	0.8858	0.9497	16	
	64×64	0.1792	0.1329	0.8800	0.8671	0.8208	0.8735	0.8487	0.9198	16	
CSLBP+DCT											
Y	16×16	0.3183	0.2210	0.7844	0.7790	0.6817	0.7816	0.7399	0.7862	272	
	32×32	0.3316	0.2213	0.7688	0.7787	0.6684	0.7737	0.7331	0.7769	1040	
	64×64	0.3380	0.2153	0.7562	0.7848	0.6620	0.7700	0.7321	0.7668	4112	
Cb	16×16	0.1110	0.0247	0.9176	0.9753	0.8890	0.9455	0.9372	0.9847	272	
	32×32	0.1050	0.0238	0.9223	0.9762	0.8950	0.9485	0.9406	0.9858	1040	
	64×64	0.1000	0.0201	0.9258	0.9799	0.9000	0.9521	0.9448	0.9853	4112	
Cr	16×16	0.1133	0.0240	0.9153	0.9760	0.8867	0.9447	0.9365	0.9840	272	
	32×32	0.0997	0.0210	0.9265	0.9790	0.9003	0.9520	0.9445	0.9859	1040	
	64×64	0.1001	0.0195	0.9257	0.9805	0.8999	0.9522	0.9449	0.9858	4112	
CSLBP-DCT											
Y	16×16	0.3572	0.2639	0.7735	0.7361	0.6428	0.7541	0.7006	0.7530	225	
	32×32	0.3468	0.2556	0.7782	0.7444	0.6532	0.7608	0.7097	0.7542	961	
	64×64	0.3421	0.2551	0.7833	0.7449	0.6579	0.7635	0.7122	0.7520	3969	
Cb	16×16	0.0689	0.0248	0.9511	0.9752	0.9311	0.9630	0.9566	0.9895	225	
	32×32	0.0640	0.0228	0.9548	0.9772	0.9360	0.9658	0.9599	0.9894	961	
	64×64	0.0713	0.0238	0.9489	0.9762	0.9287	0.9623	0.9560	0.9874	3969	
Cr	16×16	0.0683	0.0252	0.9516	0.9748	0.9317	0.9630	0.9567	0.9905	225	
	32×32	0.0638	0.0231	0.9549	0.9769	0.9362	0.9658	0.9598	0.9901	961	
	64×64	0.0692	0.0233	0.9509	0.9767	0.9308	0.9636	0.9574	0.9876	3969	

Table A.18: Results for each channel on CASIA v2.0 dataset using CSLBP and CSLBP+DCT (Feature stacking)

Table A.18 shows the results of CLBP, CLBP-DCT and CSLBP+DCT on CASIA v2.0 dataset. The CSLBP-DCT perform better than CSLBP and CSLBP+DCT and achieved an accuracy of 95.99% in Cb channel using 32×32 block size.

The results of MSLBP-DCT experiments on CASIA v2.0 dataset are illustrated in Table A.19. LBP-DCT with various radii are combined together where the highest accuracy is 96.81% by combining all four types of LBP features. This accuracy was achieved in Cb channel using 32×32 block size with AUC 0.99 and 3144 features. The same accuracy was also achieved in Cr channel using 32×32 with 1684 features by combining only two feature vectors of LBP-DCT with radii 1 and 2.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
MSLBP-DCT (1,2,3,4)											
Y	16×16	0.3149	0.2238	0.7900	0.7762	0.6851	0.7829	0.7400	0.8001	584	
	32×32	0.3176	0.2201	0.7844	0.7799	0.6824	0.7820	0.7406	0.7961	3144	
	64×64	0.3244	0.2220	0.7778	0.7780	0.6756	0.7778	0.7364	0.7796	14408	
Cb	16×16	0.0633	0.0142	0.9546	0.9858	0.9367	0.9699	0.9649	0.9917	584	
	32×32	0.0595	0.0115	0.9574	0.9885	0.9405	0.9727	0.9681	0.9916	3144	
	64×64	0.0616	0.0134	0.9557	0.9866	0.9384	0.9709	0.9660	0.9896	14408	
Cr	16×16	0.0619	0.0160	0.9559	0.9840	0.9381	0.9697	0.9645	0.9919	584	
	32×32	0.0597	0.0124	0.9573	0.9876	0.9403	0.9722	0.9675	0.9916	3144	
	64×64	0.0611	0.0113	0.9562	0.9887	0.9389	0.9722	0.9675	0.9893	14408	
MSLBP-DCT (1,2,3)											
Y	16×16	0.3105	0.2202	0.7923	0.7798	0.6895	0.7859	0.7439	0.8040	440	
	32×32	0.3188	0.2180	0.7818	0.7820	0.6812	0.7818	0.7410	0.7979	2360	
	64×64	0.3232	0.2179	0.7766	0.7821	0.6768	0.7793	0.7389	0.7851	10808	
Cb	16×16	0.0623	0.0143	0.9554	0.9857	0.9377	0.9703	0.9653	0.9918	440	
	32×32	0.0605	0.0128	0.9565	0.9872	0.9395	0.9716	0.9670	0.9918	2360	
	64×64	0.0610	0.0134	0.9563	0.9866	0.9390	0.9712	0.9664	0.9898	10808	
Cr	16×16	0.0613	0.0160	0.9560	0.9840	0.9387	0.9698	0.9647	0.9919	440	
	32×32	0.0594	0.0124	0.9576	0.9876	0.9406	0.9723	0.9677	0.9917	2360	
	64×64	0.0598	0.0112	0.9572	0.9888	0.9402	0.9727	0.9681	0.9896	10808	
MSLBP-DCT (1,2)											
Y	16×16	0.3072	0.2163	0.7936	0.7837	0.6928	0.7886	0.7476	0.8065	340	
	32×32	0.3143	0.2149	0.7845	0.7851	0.6857	0.7846	0.7445	0.7991	1684	
	64×64	0.3179	0.2147	0.7804	0.7853	0.6821	0.7827	0.7429	0.7881	7444	
Cb	16×16	0.0632	0.0150	0.9545	0.9850	0.9368	0.9695	0.9644	0.9923	340	
	32×32	0.0604	0.0114	0.9567	0.9886	0.9396	0.9724	0.9677	0.9918	1684	
	64×64	0.0601	0.0134	0.9570	0.9866	0.9399	0.9715	0.9667	0.9896	7444	
Cr	16×16	0.0602	0.0178	0.9573	0.9822	0.9398	0.9696	0.9643	0.9919	340	
	32×32	0.0578	0.0118	0.9585	0.9882	0.9422	0.9731	0.9680	0.9920	1684	
	64×64	0.0591	0.0129	0.9576	0.9871	0.9409	0.9721	0.9675	0.9892	7444	

Table A.19: Results for each channel on CASIA v2.0 dataset using MSLBP-DCT with radii of 1,2,3 and 4. (Feature stacking)

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
LBP-DCT (1)											
Y	16×16	0.3069	0.2166	0.7945	0.7834	0.6931	0.7888	0.7475	0.8025	196	
	32×32	0.3162	0.2180	0.7841	0.7820	0.6838	0.7827	0.7420	0.7991	900	
	64×64	0.3242	0.2237	0.7794	0.7763	0.6758	0.7778	0.7357	0.7880	3844	
Cb	16×16	0.0642	0.0184	0.9542	0.9816	0.9358	0.9677	0.9622	0.9923	196	
	32×32	0.0604	0.0130	0.9570	0.9870	0.9396	0.9717	0.9669	0.9921	900	
	64×64	0.0586	0.0146	0.9582	0.9854	0.9414	0.9716	0.9667	0.9901	3844	
Cr	16×16	0.0626	0.0180	0.9557	0.9820	0.9374	0.9687	0.9633	0.9915	196	
	32×32	0.0588	0.0134	0.9581	0.9866	0.9412	0.9721	0.9674	0.9921	900	
	64×64	0.0584	0.0135	0.9582	0.9865	0.9416	0.9721	0.9673	0.9895	3844	
LBP-DCT (2)											
Y	16×16	0.3402	0.2520	0.7831	0.7480	0.6598	0.7650	0.7146	0.7681	144	
	32×32	0.3402	0.2436	0.7755	0.7564	0.6598	0.7657	0.7184	0.7733	784	
	64×64	0.3345	0.2342	0.7754	0.7658	0.6655	0.7705	0.7258	0.7664	3600	
Cb	16×16	0.0800	0.0195	0.9420	0.9805	0.9200	0.9608	0.9544	0.9891	144	
	32×32	0.0680	0.0156	0.9511	0.9844	0.9320	0.9675	0.9620	0.9901	784	
	64×64	0.0706	0.0165	0.9493	0.9835	0.9294	0.9661	0.9605	0.9874	3600	
Cr	16×16	0.0789	0.0230	0.9432	0.9770	0.9211	0.9598	0.9531	0.9892	144	
	32×32	0.0649	0.0160	0.9535	0.9840	0.9351	0.9684	0.9632	0.9902	784	
	64×64	0.0679	0.0168	0.9512	0.9832	0.9321	0.9669	0.9614	0.9870	3600	
LBP-DCT (3)											
Y	16×16	0.3523	0.2661	0.7816	0.7339	0.6477	0.7568	0.7020	0.7536	100	
	32×32	0.3448	0.2486	0.7743	0.7514	0.6552	0.7626	0.7139	0.7681	676	
	64×64	0.3382	0.2366	0.7724	0.7634	0.6618	0.7677	0.7228	0.7671	3364	
Cb	16×16	0.0976	0.0307	0.9290	0.9693	0.9024	0.9487	0.9404	0.9845	100	
	32×32	0.0724	0.0207	0.9481	0.9793	0.9276	0.9634	0.9573	0.9885	676	
	64×64	0.0702	0.0227	0.9501	0.9773	0.9298	0.9635	0.9573	0.9869	3364	
Cr	16×16	0.0909	0.0318	0.9347	0.9682	0.9091	0.9511	0.9430	0.9851	100	
	32×32	0.0715	0.0223	0.9491	0.9777	0.9285	0.9631	0.9569	0.9882	676	
	64×64	0.0714	0.0228	0.9491	0.9772	0.9286	0.9629	0.9566	0.9866	3364	
LBP-DCT (4)											
Y	16×16	0.3388	0.2496	0.7826	0.7504	0.6612	0.7660	0.7165	0.7682	144	
	32×32	0.3415	0.2457	0.7760	0.7543	0.6585	0.7649	0.7169	0.7724	784	
	64×64	0.3351	0.2363	0.7762	0.7637	0.6649	0.7698	0.7246	0.7678	3600	
Cb	16×16	0.0801	0.0200	0.9423	0.9800	0.9199	0.9607	0.9543	0.9890	144	
	32×32	0.0673	0.0167	0.9519	0.9833	0.9327	0.9673	0.9618	0.9901	784	
	64×64	0.0717	0.0178	0.9487	0.9822	0.9283	0.9651	0.9593	0.9874	3600	
Cr	16×16	0.0812	0.0249	0.9417	0.9751	0.9188	0.9581	0.9511	0.9891	144	
	32×32	0.0652	0.0186	0.9534	0.9814	0.9348	0.9671	0.9616	0.9901	784	
	64×64	0.0679	0.0183	0.9514	0.9817	0.9321	0.9663	0.9606	0.9869	3600	

Table A.20: Results for each channel on CASIA v2.0 dataset using LBP-DCT with radii of 1,2,3 and 4. (Feature stacking)

Table A.20 illustrates the results of LBP-DCT with different radii. LBP-DCT with radius 1 achieved better accuracy of 96.74% with AUC 0.99 in Cr channel using 32×32 block size.

Table A.21 shows the results of MSLBP on CASIA v2.0 dataset. Various LBP features were combined together to make MSLBP features. The highest accuracy achieved by combining two feature vectors of LBP with radii 1 and 2 with an accuracy of 96.7% in Cb channel using 32×32 block size with only 512 features.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
MSLBP (1,2,3,4)											
Y	16×16	0.3263	0.2365	0.7868	0.7636	0.6737	0.7749	0.7287	0.7919	1024	
	32×32	0.3256	0.2357	0.7862	0.7643	0.6744	0.7751	0.7295	0.7867	1024	
	64×64	0.3323	0.2448	0.7863	0.7552	0.6677	0.7704	0.7219	0.7758	1024	
Cb	16×16	0.0599	0.0162	0.9575	0.9838	0.9401	0.9704	0.9654	0.9903	1024	
	32×32	0.0594	0.0168	0.9581	0.9832	0.9406	0.9705	0.9654	0.9903	1024	
	64×64	0.0648	0.0208	0.9540	0.9792	0.9352	0.9664	0.9606	0.9898	1024	
Cr	16×16	0.0617	0.0171	0.9562	0.9829	0.9383	0.9693	0.9640	0.9900	1024	
	32×32	0.0616	0.0169	0.9561	0.9831	0.9384	0.9694	0.9642	0.9905	1024	
	64×64	0.0623	0.0182	0.9557	0.9818	0.9377	0.9685	0.9632	0.9900	1024	
MSLBP (1,2,3)											
Y	16×16	0.3231	0.2341	0.7880	0.7659	0.6769	0.7767	0.7312	0.7941	768	
	32×32	0.3293	0.2398	0.7855	0.7602	0.6707	0.7725	0.7256	0.7880	768	
	64×64	0.3288	0.2416	0.7873	0.7584	0.6712	0.7725	0.7249	0.7804	768	
Cb	16×16	0.0607	0.0159	0.9568	0.9841	0.9393	0.9702	0.9651	0.9906	768	
	32×32	0.0582	0.0155	0.9586	0.9845	0.9418	0.9714	0.9665	0.9906	768	
	64×64	0.0610	0.0191	0.9564	0.9809	0.9390	0.9685	0.9632	0.9901	768	
Cr	16×16	0.0619	0.0150	0.9560	0.9850	0.9381	0.9702	0.9651	0.9907	768	
	32×32	0.0601	0.0166	0.9573	0.9834	0.9399	0.9702	0.9651	0.9909	768	
	64×64	0.0635	0.0184	0.9549	0.9816	0.9365	0.9681	0.9626	0.9904	768	
MSLBP (1,2)											
Y	16×16	0.3224	0.2346	0.7893	0.7654	0.6776	0.7770	0.7313	0.8001	512	
	32×32	0.3208	0.2364	0.7922	0.7636	0.6792	0.7775	0.7310	0.7928	512	
	64×64	0.3296	0.2453	0.7898	0.7547	0.6704	0.7717	0.7227	0.7832	512	
Cb	16×16	0.0611	0.0155	0.9565	0.9845	0.9389	0.9703	0.9652	0.9909	512	
	32×32	0.0576	0.0151	0.9590	0.9849	0.9424	0.9718	0.9670	0.9910	512	
	64×64	0.0619	0.0148	0.9558	0.9852	0.9382	0.9702	0.9652	0.9909	512	
Cr	16×16	0.0606	0.0164	0.9569	0.9836	0.9394	0.9701	0.9650	0.9912	512	
	32×32	0.0586	0.0151	0.9582	0.9849	0.9414	0.9713	0.9665	0.9911	512	
	64×64	0.0621	0.0153	0.9557	0.9847	0.9379	0.9700	0.9649	0.9906	512	

Table A.21: Results for each channel on CASIA v2.0 dataset using MSLBP. (Feature stacking)

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
LBP (1)											
Y	16×16	0.3184	0.2346	0.7938	0.7654	0.6816	0.7792	0.7332	0.7992	256	
	32×32	0.3146	0.2364	0.7994	0.7636	0.6854	0.7810	0.7340	0.7973	256	
	64×64	0.3275	0.2483	0.7947	0.7517	0.6725	0.7724	0.7221	0.7845	256	
Cb	16×16	0.0595	0.0165	0.9576	0.9835	0.9405	0.9704	0.9654	0.9910	256	
	32×32	0.0598	0.0147	0.9573	0.9853	0.9402	0.9711	0.9662	0.9909	256	
	64×64	0.0640	0.0189	0.9544	0.9811	0.9360	0.9676	0.9620	0.9904	256	
Cr	16×16	0.0661	0.0157	0.9526	0.9843	0.9339	0.9682	0.9628	0.9908	256	
	32×32	0.0612	0.0151	0.9563	0.9849	0.9388	0.9704	0.9654	0.9910	256	
	64×64	0.0660	0.0145	0.9526	0.9855	0.9340	0.9687	0.9636	0.9907	256	
LBP (2)											
Y	16×16	0.3532	0.2626	0.7777	0.7374	0.6468	0.7568	0.7035	0.7503	256	
	32×32	0.3587	0.2661	0.7743	0.7339	0.6413	0.7533	0.6993	0.7427	256	
	64×64	0.3635	0.2736	0.7756	0.7264	0.6365	0.7500	0.6933	0.7354	256	
Cb	16×16	0.0759	0.0192	0.9453	0.9808	0.9241	0.9627	0.9565	0.9867	256	
	32×32	0.0691	0.0191	0.9505	0.9809	0.9309	0.9654	0.9597	0.9875	256	
	64×64	0.0730	0.0259	0.9483	0.9741	0.9270	0.9610	0.9544	0.9868	256	
Cr	16×16	0.0739	0.0219	0.9470	0.9781	0.9261	0.9623	0.9559	0.9861	256	
	32×32	0.0689	0.0173	0.9506	0.9827	0.9311	0.9664	0.9608	0.9873	256	
	64×64	0.0720	0.0244	0.9486	0.9756	0.9280	0.9618	0.9554	0.9865	256	
LBP (3)											
Y	16×16	0.3649	0.2795	0.7812	0.7205	0.6351	0.7495	0.6900	0.7310	256	
	32×32	0.3704	0.2891	0.7847	0.7109	0.6296	0.7459	0.6828	0.7194	256	
	64×64	0.3692	0.2913	0.7891	0.7087	0.6308	0.7465	0.6821	0.7181	256	
Cb	16×16	0.1098	0.0368	0.9199	0.9632	0.8902	0.9410	0.9316	0.9755	256	
	32×32	0.1026	0.0324	0.9250	0.9676	0.8974	0.9458	0.9372	0.9792	256	
	64×64	0.1143	0.0375	0.9163	0.9625	0.8857	0.9388	0.9291	0.9760	256	
Cr	16×16	0.1113	0.0322	0.9183	0.9678	0.8887	0.9424	0.9334	0.9767	256	
	32×32	0.1005	0.0231	0.9262	0.9769	0.8995	0.9509	0.9432	0.9810	256	
	64×64	0.1042	0.0304	0.9238	0.9696	0.8958	0.9461	0.9376	0.9789	256	
LBP (4)											
Y	16×16	0.3572	0.2635	0.7738	0.7365	0.6428	0.7546	0.7012	0.7496	256	
	32×32	0.3597	0.2661	0.7725	0.7339	0.6403	0.7526	0.6989	0.7419	256	
	64×64	0.3660	0.2756	0.7751	0.7244	0.6341	0.7487	0.6913	0.7370	256	
Cb	16×16	0.0764	0.0199	0.9450	0.9801	0.9236	0.9622	0.9559	0.9864	256	
	32×32	0.0693	0.0190	0.9505	0.9810	0.9307	0.9655	0.9597	0.9876	256	
	64×64	0.0741	0.0255	0.9472	0.9745	0.9259	0.9607	0.9540	0.9868	256	
Cr	16×16	0.0735	0.0223	0.9474	0.9777	0.9265	0.9623	0.9559	0.9859	256	
	32×32	0.0692	0.0189	0.9505	0.9811	0.9308	0.9655	0.9597	0.9872	256	
	64×64	0.0736	0.0247	0.9476	0.9753	0.9264	0.9612	0.9546	0.9863	256	

Table A.22: Results for each channel on CASIA v2.0 dataset using LBP using radii 1,2,3 and 4. (Feature stacking)

Table A.22 shows the experimental results of LBP with various radii on CASIA v2.0 dataset. The highest accuracy in this table is 96.62% which is achieved in Cb channel with 32×32 block size and only 256 LBP features with radius 1.

Table A.23 illustrates the results of MSLBP-DCT with different combination of 8, 12 and 16 neighbors on CASIA v2.0 dataset. The combination of LBP-DCT($r=1$, $p=8$ and $r=2$, $p=12$) performs better than the other combination and single radius models. The achieved an accuracy of 96.83% in Cb channel using 32×32 block size.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features Dimension
MSLBP-DCT ($r=1, p=8, r=2, p=12, r=3, p=16$)										
Y	16×16	0.314914	0.222625	0.790317	0.777375	0.685086	0.783612	0.740864	0.802961	360
	32×32	0.318509	0.218726	0.7825	0.781274	0.681491	0.781775	0.740729	0.7973	2152
	64×64	0.326213	0.223745	0.777422	0.776255	0.673787	0.776654	0.734628	0.779535	10344
Cb	16×16	0.061092	0.014712	0.95641	0.985288	0.938908	0.970597	0.965601	0.992012	360
	32×32	0.059657	0.012669	0.957243	0.987331	0.940343	0.972041	0.967351	0.99163	2152
	64×64	0.060966	0.013237	0.956519	0.986763	0.939034	0.971392	0.966553	0.989694	10344
Cr	16×16	0.060893	0.016174	0.956572	0.983826	0.939107	0.969994	0.964894	0.991773	360
	32×32	0.058831	0.01292	0.958117	0.98708	0.941169	0.972374	0.96767	0.991647	2152
	64×64	0.059061	0.012599	0.957808	0.987401	0.940939	0.972363	0.96767	0.989565	10344
MSLBP-DCT ($r=1, p=8$ and $r=2, p=12$)										
Y	16×16	0.312652	0.222832	0.791969	0.777168	0.687348	0.784369	0.741639	0.802146	296
	32×32	0.314756	0.214936	0.783691	0.785064	0.685244	0.784275	0.744374	0.799118	1576
	64×64	0.322103	0.222336	0.780828	0.777664	0.677897	0.779126	0.73738	0.784316	7208
Cb	16×16	0.061925	0.015299	0.955859	0.984701	0.938075	0.970037	0.964971	0.992301	296
	32×32	0.05944	0.011162	0.957425	0.988838	0.94056	0.972859	0.9683	0.991887	1576
	64×64	0.059264	0.013197	0.957697	0.986803	0.940736	0.972022	0.967274	0.990132	7208
Cr	16×16	0.060332	0.015485	0.957145	0.984515	0.939668	0.970619	0.965602	0.991997	296
	32×32	0.057997	0.012336	0.958593	0.987664	0.942003	0.972897	0.968302	0.991915	1576
	64×64	0.059343	0.012798	0.95751	0.987202	0.940657	0.972118	0.967433	0.989752	7208
LBP-DCT ($r=3, p=16$)										
Y	16×16	0.369773	0.277198	0.772379	0.722802	0.630227	0.746608	0.688906	0.739228	64
	32×32	0.3424	0.249306	0.778246	0.750694	0.6576	0.764092	0.714897	0.762758	576
	64×64	0.337975	0.240601	0.775571	0.759399	0.662025	0.767223	0.720919	0.761943	3136
Cb	16×16	0.106032	0.0367	0.922924	0.9633	0.893968	0.94265	0.933424	0.981681	64
	32×32	0.081284	0.023354	0.941548	0.976646	0.918716	0.958754	0.95189	0.98801	576
	64×64	0.086278	0.024161	0.937666	0.975839	0.913722	0.956359	0.949196	0.985298	3136
Cr	16×16	0.113985	0.036963	0.916387	0.963037	0.886015	0.939075	0.929449	0.981361	64
	32×32	0.083132	0.024896	0.939991	0.975104	0.916868	0.957208	0.950147	0.987391	576
	64×64	0.082354	0.024904	0.940755	0.975096	0.917646	0.95759	0.950551	0.984587	3136
LBP-DCT ($r=2, p=12$)										
Y	16×16	0.356093	0.266251	0.777559	0.733749	0.643907	0.754822	0.700379	0.753171	100
	32×32	0.343713	0.250028	0.777237	0.749972	0.656287	0.763227	0.713877	0.766256	676
	64×64	0.339793	0.243596	0.776151	0.756404	0.660207	0.765971	0.7186	0.760816	3364
Cb	16×16	0.087632	0.029877	0.936984	0.970123	0.912368	0.953227	0.945478	0.986604	100
	32×32	0.068473	0.019816	0.951216	0.980184	0.931527	0.965458	0.959575	0.989812	676
	64×64	0.067426	0.018457	0.95179	0.981543	0.932574	0.96643	0.960777	0.988299	3364
Cr	16×16	0.086022	0.032208	0.938461	0.967792	0.913978	0.952868	0.944912	0.986502	100
	32×32	0.067672	0.021032	0.951594	0.978968	0.932328	0.965059	0.959189	0.989526	676
	64×64	0.068424	0.019743	0.951134	0.980257	0.931576	0.965448	0.959588	0.988128	3364

Table A.23: Results for each channel on CASIA v2.0 dataset using LBP-DCT and MSLBP-DCT with neighbors 8,12 and 16. (Features stacking)

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features Dimension
MSLBP ($r=1, p=8$ and $r=2, p=12$)										
Y	16×16	0.3160	0.2307	0.7937	0.7693	0.6840	0.7812	0.7362	0.7919	4352
	32×32	0.3180	0.2300	0.7913	0.7700	0.6820	0.7804	0.7357	0.7938	4352
	64×64	0.3153	0.2309	0.7945	0.7691	0.6847	0.7815	0.7365	0.7864	4352
Cb	16×16	0.0872	0.0140	0.9358	0.9860	0.9128	0.9602	0.9540	0.9899	4352
	32×32	0.0689	0.0137	0.9503	0.9863	0.9311	0.9680	0.9627	0.9908	4352
	64×64	0.0676	0.0154	0.9514	0.9846	0.9324	0.9677	0.9623	0.9910	4352
Cr	16×16	0.0820	0.0199	0.9407	0.9801	0.9180	0.9599	0.9534	0.9895	4352
	32×32	0.0664	0.0168	0.9524	0.9832	0.9336	0.9676	0.9621	0.9909	4352
	64×64	0.0647	0.0172	0.9538	0.9828	0.9353	0.9681	0.9627	0.9908	4352
LBP ($r=2, p=12$)										
Y	16×16	0.3311	0.2470	0.7893	0.7530	0.6689	0.7705	0.7211	0.7677	4096
	32×32	0.3277	0.2408	0.7884	0.7592	0.6723	0.7734	0.7259	0.7725	4096
	64×64	0.3212	0.2380	0.7936	0.7620	0.6788	0.7773	0.7301	0.7740	4096
Cb	16×16	0.1053	0.0179	0.9215	0.9821	0.8947	0.9508	0.9434	0.9875	4096
	32×32	0.0763	0.0159	0.9447	0.9841	0.9237	0.9640	0.9581	0.9897	4096
	64×64	0.0737	0.0171	0.9469	0.9829	0.9263	0.9645	0.9586	0.9902	4096
Cr	16×16	0.1063	0.0267	0.9213	0.9733	0.8937	0.9465	0.9383	0.9865	4096
	32×32	0.0725	0.0195	0.9479	0.9805	0.9275	0.9639	0.9579	0.9898	4096
	64×64	0.0700	0.0215	0.9500	0.9785	0.9300	0.9641	0.9580	0.9899	4096

Table A.24: Results for each channel on CASIA v2.0 dataset using LBP and MSLBP with neighbors 8, 12 and 16. (Features stacking)

LBP and MSLBP with neighbors of 8 and 12 experimental results are illustrated in Table A.24. In these experiments the combination of LBP($r=1, p=8$) and LBP ($r=2, p=12$) features performs better with 96.27% accuracy in Cr channel with 64×64 block size.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
SLBP											
Y	16×16	0.3562	0.2651	0.7763	0.7349	0.6438	0.7549	0.7008	0.7523	256	
	32×32	0.3546	0.2651	0.7785	0.7349	0.6454	0.7560	0.7018	0.7486	256	
	64×64	0.3629	0.2778	0.7812	0.7222	0.6371	0.7503	0.6915	0.7378	256	
Cb	16×16	0.0646	0.0204	0.9541	0.9796	0.9354	0.9667	0.9609	0.9891	256	
	32×32	0.0656	0.0218	0.9536	0.9782	0.9344	0.9658	0.9598	0.9894	256	
	64×64	0.0678	0.0229	0.9518	0.9771	0.9322	0.9643	0.9582	0.9890	256	
Cr	16×16	0.0705	0.0186	0.9494	0.9814	0.9295	0.9651	0.9593	0.9892	256	
	32×32	0.0694	0.0186	0.9501	0.9814	0.9306	0.9655	0.9597	0.9895	256	
	64×64	0.0691	0.0207	0.9506	0.9793	0.9309	0.9647	0.9588	0.9893	256	
SLBP+DCT											
Y	16×16	0.3232	0.2254	0.7819	0.7746	0.6768	0.7782	0.7355	0.7950	512	
	32×32	0.3266	0.2188	0.7729	0.7812	0.6734	0.7769	0.7366	0.7850	1280	
	64×64	0.3347	0.2157	0.7605	0.7843	0.6653	0.7722	0.7338	0.7719	4352	
Cb	16×16	0.0669	0.0195	0.9523	0.9805	0.9331	0.9661	0.9604	0.9899	512	
	32×32	0.0786	0.0191	0.9428	0.9809	0.9214	0.9614	0.9551	0.9896	1280	
	64×64	0.0926	0.0195	0.9321	0.9805	0.9074	0.9557	0.9486	0.9875	4352	
Cr	16×16	0.0693	0.0186	0.9501	0.9814	0.9307	0.9655	0.9597	0.9899	512	
	32×32	0.0814	0.0205	0.9411	0.9795	0.9186	0.9599	0.9533	0.9892	1280	
	64×64	0.0939	0.0178	0.9308	0.9822	0.9061	0.9558	0.9489	0.9876	4352	

Table A.25: Results for each channel on CASIA v2.0 dataset using SLBP and SLBP+DCT. (Features stacking)

The results of SLP and SLBP with DCT features one CASIA v2.0 are shown in Table A.25. The SLBP alone perform better than SLBP with DCT coefficients and achieved an accuracy of 95.98% in Cb channel using 32×32 block size with only 256 features.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
WLD											
Y	16×16	0.3598	0.2624	0.7686	0.7376	0.6402	0.7527	0.7004	0.7398	225	
	32×32	0.3548	0.2520	0.7650	0.7480	0.6452	0.7561	0.7074	0.7413	961	
	64×64	0.3437	0.2359	0.7655	0.7641	0.6563	0.7645	0.7201	0.7447	3969	
Cb	16×16	0.1598	0.0502	0.8788	0.9498	0.8402	0.9129	0.9004	0.9640	225	
	32×32	0.1579	0.0451	0.8796	0.9549	0.8421	0.9157	0.9039	0.9724	961	
	64×64	0.1655	0.0459	0.8726	0.9541	0.8345	0.9115	0.8994	0.9671	3969	
Cr	16×16	0.1602	0.0404	0.8766	0.9596	0.8398	0.9161	0.9047	0.9686	225	
	32×32	0.1548	0.0370	0.8809	0.9630	0.8452	0.9201	0.9092	0.9742	961	
	64×64	0.1643	0.0358	0.8717	0.9642	0.8357	0.9156	0.9046	0.9687	3969	
WLD-DCT											
Y	16×16	0.3312	0.2441	0.7864	0.7559	0.6688	0.7708	0.7227	0.7788	225	
	32×32	0.3377	0.2365	0.7723	0.7635	0.6623	0.7677	0.7231	0.7740	961	
	64×64	0.3411	0.2216	0.7574	0.7784	0.6589	0.7675	0.7277	0.7712	3969	
Cb	16×16	0.0805	0.0196	0.9416	0.9804	0.9195	0.9606	0.9542	0.9873	225	
	32×32	0.0724	0.0149	0.9481	0.9851	0.9276	0.9662	0.9606	0.9893	961	
	64×64	0.0753	0.0140	0.9451	0.9860	0.9247	0.9651	0.9595	0.9887	3969	
Cr	16×16	0.0771	0.0195	0.9443	0.9805	0.9229	0.9620	0.9558	0.9883	225	
	32×32	0.0675	0.0139	0.9512	0.9861	0.9325	0.9683	0.9632	0.9904	961	
	64×64	0.0686	0.0114	0.9503	0.9886	0.9314	0.9690	0.9640	0.9899	3969	

Table A.26: Results for each channel on CASIA v2.0 dataset using WLD and WLD-DCT. (Feature stacking)

Table A.26 illustrates the results of WLD and WLD-DCT on CASIA v2.0 dataset. WLD with DCT performs better than only WLD features and achieved an accuracy of 96.4% in Cr channel using 64×64 block size with 3969 features.

Table A.27 illustrates the results of LPQ-DCT and MSLPQ-DCT on CASIA v2.0 dataset. The MSLPQ-DCT performs better than single scale LPQ and achieved an accuracy of 96.85% in Cb channel with 32×32 block size using 1684 features.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
MSLPQ-DCT (3,5)											
Y	16×16	0.3041	0.2114	0.7941	0.7886	0.6959	0.7913	0.7513	0.8198	340	
	32×32	0.3009	0.2035	0.7929	0.7965	0.6991	0.7946	0.7566	0.8215	1684	
	64×64	0.3092	0.2041	0.7836	0.7959	0.6908	0.7896	0.7522	0.8144	7444	
Cb	16×16	0.0675	0.0123	0.9512	0.9877	0.9325	0.9691	0.9640	0.9910	340	
	32×32	0.0594	0.0108	0.9574	0.9892	0.9406	0.9731	0.9685	0.9915	1684	
	64×64	0.0598	0.0119	0.9572	0.9881	0.9402	0.9724	0.9677	0.9888	7444	
Cr	16×16	0.0686	0.0121	0.9506	0.9879	0.9314	0.9689	0.9638	0.9908	340	
	32×32	0.0603	0.0110	0.9569	0.9890	0.9397	0.9727	0.9681	0.9915	1684	
	64×64	0.0586	0.0129	0.9580	0.9871	0.9414	0.9723	0.9677	0.9888	7444	
LPQ-DCT (5)											
Y	16×16	0.3562	0.2205	0.7367	0.7795	0.6438	0.7572	0.7198	0.7765	144	
	32×32	0.3586	0.2017	0.7185	0.7983	0.6414	0.7562	0.7252	0.7850	784	
	64×64	0.3548	0.1955	0.7196	0.8045	0.6452	0.7595	0.7296	0.7863	3600	
Cb	16×16	0.0617	0.0229	0.9567	0.9771	0.9383	0.9667	0.9609	0.9910	144	
	32×32	0.0557	0.0208	0.9609	0.9792	0.9443	0.9700	0.9647	0.9901	784	
	64×64	0.0591	0.0200	0.9583	0.9800	0.9409	0.9690	0.9636	0.9886	3600	
Cr	16×16	0.0619	0.0252	0.9565	0.9748	0.9381	0.9655	0.9595	0.9902	144	
	32×32	0.0554	0.0201	0.9608	0.9799	0.9446	0.9702	0.9651	0.9913	784	
	64×64	0.0560	0.0223	0.9606	0.9777	0.9440	0.9690	0.9636	0.9895	3600	
LPQ-DCT (3)											
Y	16×16	0.3185	0.2004	0.7702	0.7996	0.6815	0.7846	0.7490	0.8119	196	
	32×32	0.3287	0.1828	0.7459	0.8172	0.6713	0.7798	0.7503	0.8152	900	
	64×64	0.3320	0.1801	0.7406	0.8199	0.6680	0.7782	0.7495	0.8130	3844	
Cb	16×16	0.0590	0.0293	0.9589	0.9707	0.9410	0.9648	0.9584	0.9887	196	
	32×32	0.0583	0.0252	0.9591	0.9748	0.9417	0.9669	0.9610	0.9888	900	
	64×64	0.0587	0.0252	0.9589	0.9748	0.9413	0.9668	0.9609	0.9880	3844	
Cr	16×16	0.0601	0.0292	0.9584	0.9708	0.9399	0.9645	0.9581	0.9886	196	
	32×32	0.0543	0.0237	0.9621	0.9763	0.9457	0.9691	0.9636	0.9895	900	
	64×64	0.0568	0.0263	0.9604	0.9737	0.9432	0.9670	0.9611	0.9884	3844	

Table A.27: Results for each channel on CASIA v2.0 dataset using LPQ-DCT and MSLPQ-DCT. (Features stacking)

The results of LPQ and MSLPQ are illustrated in Table A.28. LPQ features with two different windows 3 and 5 were extracted. The MSLPQ achieved the highest accuracy of 97.31% with AUC 0.99 on CASIA v2.0 dataset in Cb channel with 16×16 block size and only 512 features.

Channels	Block Size	FPR	FNR	Precision	Sensitivity	Specificity	F_1	Accuracy	AUC	Features	Dimension
MSLPQ (3,5)											
Y	16×16	0.3437	0.2433	0.7712	0.7567	0.6563	0.7638	0.7170	0.7709		512
	32×32	0.3511	0.2498	0.7680	0.7502	0.6489	0.7588	0.7104	0.7631		512
	64×64	0.3571	0.2531	0.7633	0.7469	0.6429	0.7549	0.7060	0.7529		512
Cb	16×16	0.0509	0.0093	0.9636	0.9907	0.9491	0.9769	0.9731	0.9927		512
	32×32	0.0506	0.0124	0.9641	0.9876	0.9494	0.9757	0.9715	0.9924		512
	64×64	0.0516	0.0174	0.9637	0.9826	0.9484	0.9730	0.9683	0.9913		512
Cr	16×16	0.0499	0.0103	0.9644	0.9897	0.9501	0.9769	0.9730	0.9928		512
	32×32	0.0516	0.0139	0.9636	0.9861	0.9484	0.9747	0.9703	0.9927		512
	64×64	0.0518	0.0172	0.9634	0.9828	0.9482	0.9729	0.9682	0.9915		512
LPQ (5)											
Y	16×16	0.4056	0.2344	0.6734	0.7656	0.5944	0.7164	0.6838	0.7371		256
	32×32	0.4086	0.2436	0.6775	0.7564	0.5914	0.7146	0.6790	0.7264		256
	64×64	0.4055	0.2434	0.6834	0.7566	0.5945	0.7178	0.6811	0.7233		256
Cb	16×16	0.0652	0.0265	0.9539	0.9735	0.9348	0.9636	0.9572	0.9853		256
	32×32	0.0657	0.0321	0.9540	0.9679	0.9343	0.9609	0.9540	0.9838		256
	64×64	0.0786	0.0344	0.9444	0.9656	0.9214	0.9549	0.9471	0.9787		256
Cr	16×16	0.0637	0.0271	0.9552	0.9729	0.9363	0.9640	0.9576	0.9861		256
	32×32	0.0646	0.0292	0.9548	0.9708	0.9354	0.9627	0.9561	0.9855		256
	64×64	0.0739	0.0343	0.9478	0.9657	0.9261	0.9566	0.9491	0.9787		256
LPQ (3)											
Y	16×16	0.3764	0.2268	0.7135	0.7732	0.6236	0.7419	0.7053	0.7672		256
	32×32	0.3857	0.2334	0.7049	0.7666	0.6143	0.7343	0.6974	0.7522		256
	64×64	0.3892	0.2346	0.7010	0.7654	0.6108	0.7317	0.6949	0.7463		256
Cb	16×16	0.0495	0.0114	0.9649	0.9886	0.9503	0.9756	0.9719	0.9931		256
	32×32	0.0489	0.0177	0.9657	0.9823	0.9511	0.9739	0.9693	0.9925		256
	64×64	0.0503	0.0202	0.9647	0.9798	0.9497	0.9722	0.9673	0.9906		256
Cr	16×16	0.0495	0.0162	0.9651	0.9838	0.9505	0.9744	0.9699	0.9927		256
	32×32	0.0493	0.0160	0.9653	0.9840	0.9507	0.9745	0.9700	0.9925		256
	64×64	0.0503	0.0208	0.9647	0.9792	0.9497	0.9719	0.9669	0.9898		256

Table A.28: Results for each channel on CASIA v2.0 dataset using LPQ and MSLPQ (Feature Stacking)

A.2 Deep Learning Based Experiments

Experiment 1

This experiment consists of four convolutional layers with padding 1. Each convolutional layer was followed by a batch normalization layer and ReLU layer. Max-pooling layer of size 2 with stride 2 was used after second and third convolutional layers. The first and second convolutional layers have the same size of 5×5 with 16 and 32 filters and padding 1. Random weights were initialized to each of the convolutional layer. Figure A.1 shows the accuracy and loss plots. The accuracy reached to 65.48% and the model stopped improving. The experimental setup options were set to as follow: minimum batch size was 256, validation frequency was 30, the maximum epoch was 30 and initially learning rate was 0.001.

The input image size of $128 \times 128 \times 3$ was used where the 128 is height and width while the 3 represents the color channels (Red, Green and Blue). The experiment was evaluated using 99376 patches where 49688 of them were forged and the rest were authentic. All these samples were extracted via augmentation from CASIA v1.0 dataset as discussed in Chapter 3.

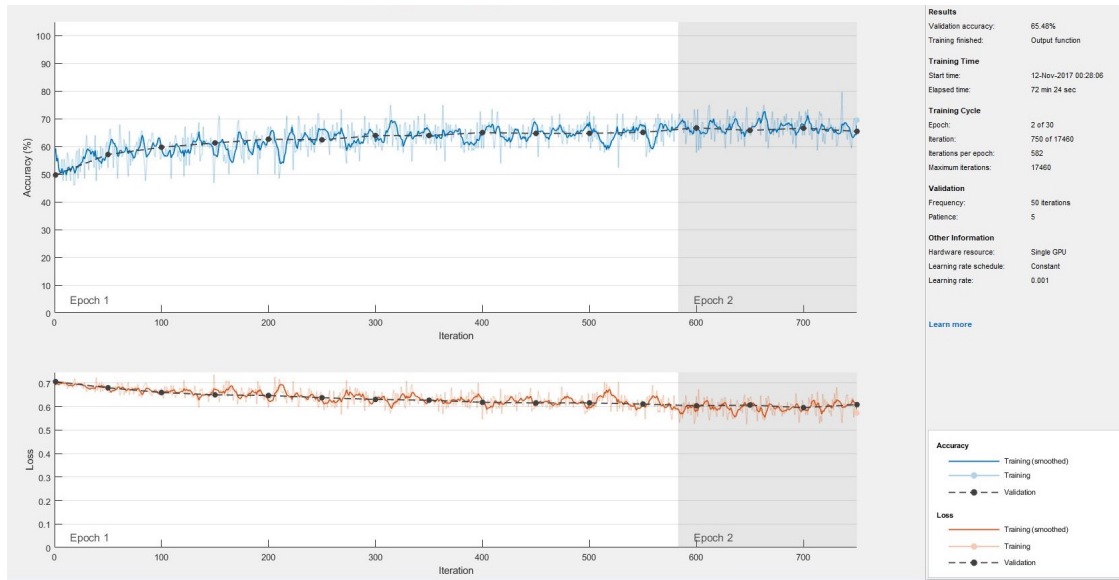


Figure A.1: The results of four convolutional layer model using 99376 patches

Experiment 2

Figure A.2 shows the results of another experiment with the same settings of the first experiment but the number of patches was reduced to 2000 which achieved an accuracy of 60.60%. We can see that the training curve of the model is performing better and achieved above 90% accuracy during the training phase but the validation curve did not improve and got worst after the 200th iteration. Similarly, the loss function also performs better and reduced consistently but the validation loss function increased between 150 – 200. This is one of the main issues to be handled in classification problems and especially in deep learning model and is called model over-fitting.

The over-fitting can be solved with various solutions e.g increase the data, increase the number of convolutional layers or use regularization. To address this problem, more experiments were conducted to train the model on different scenarios.

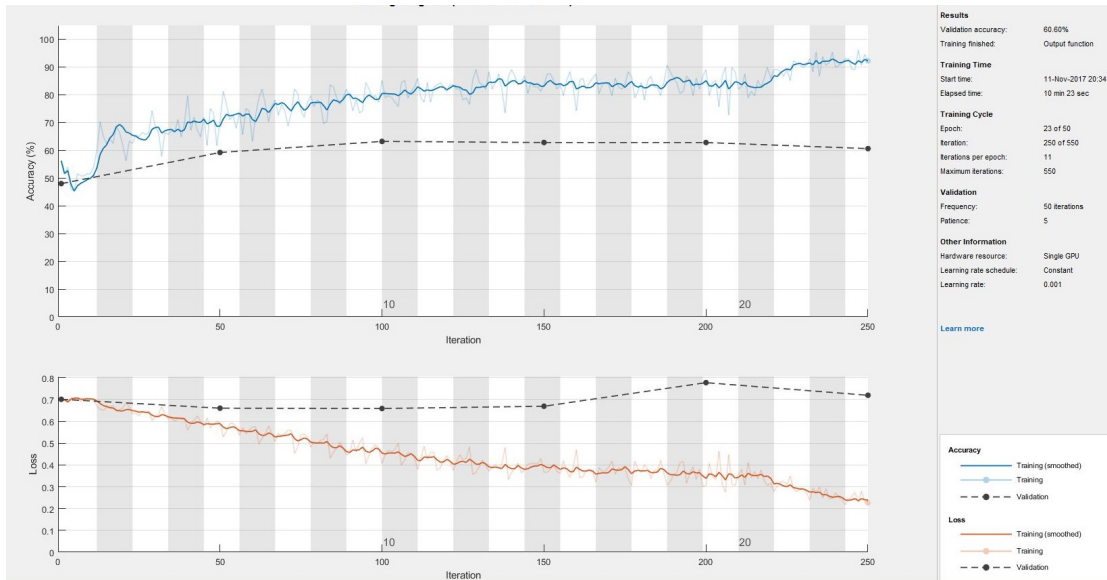


Figure A.2: The results of four convolutional layer model using 2000 patches

Experiment 3

The experiment was performed with a setup similar to experiment 1 but now we increased the number of patches. The idea of reducing the number of patches 12422. In this way we evaluate and fit a model on a moderate size of patches with better performance but less execution time. Figure A.3 illustrates the performance and loss function and achieved validation accuracy of 65.13%.

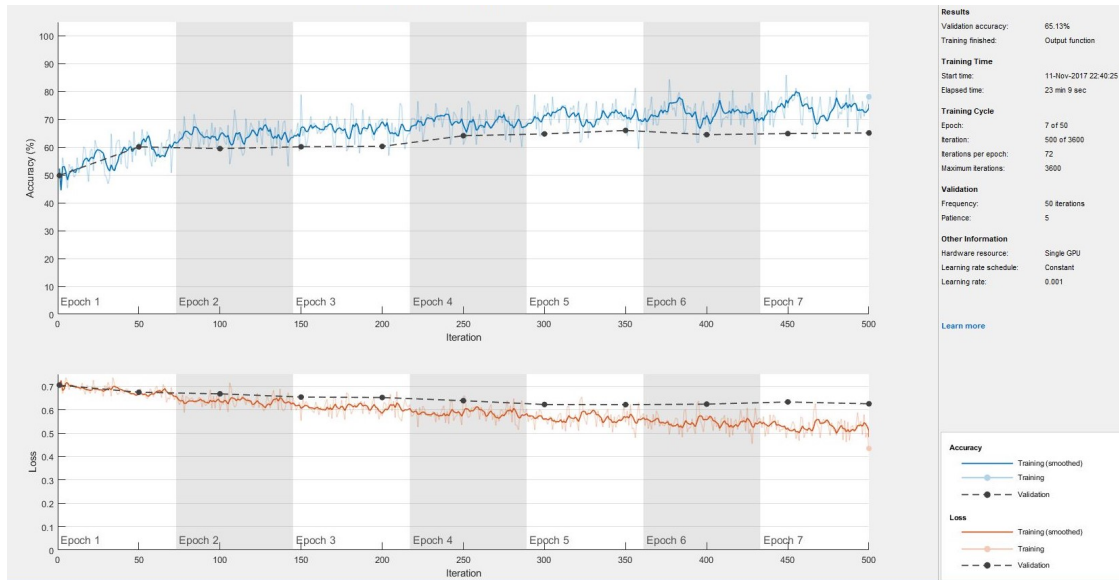


Figure A.3: The results of four convolutional layer model using 12422 patches

Experiment 4

This experiment was conducted with the same setting but with the increase of 1 convolutional layer followed by Max-Pooling layer, where the total number of convolutional layers are 5 now in the CNN model. The experiment used 2000 patches and was evaluated with various number of epochs. Figure A.4 shows the results with 30 epochs which achieved 70.20% accuracy. The results with 50 epochs illustrated in Figure A.5 with accuracy 77.20%. With the use of 80 epochs, the accuracy reached to 84.60% as shown in Figure A.6. Figure A.7 shows the results with 150 epochs which achieved the accuracy of 80.80%. We can see that it stops improving at around the 90th epoch.

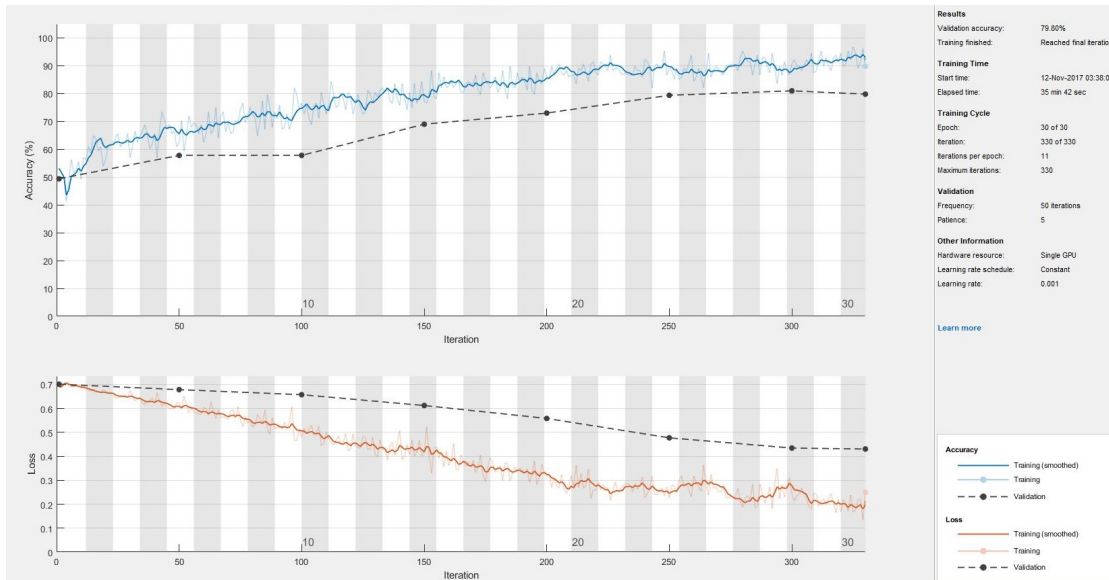


Figure A.4: The results of five convolutional layer model using 2000 patches with 30 epochs

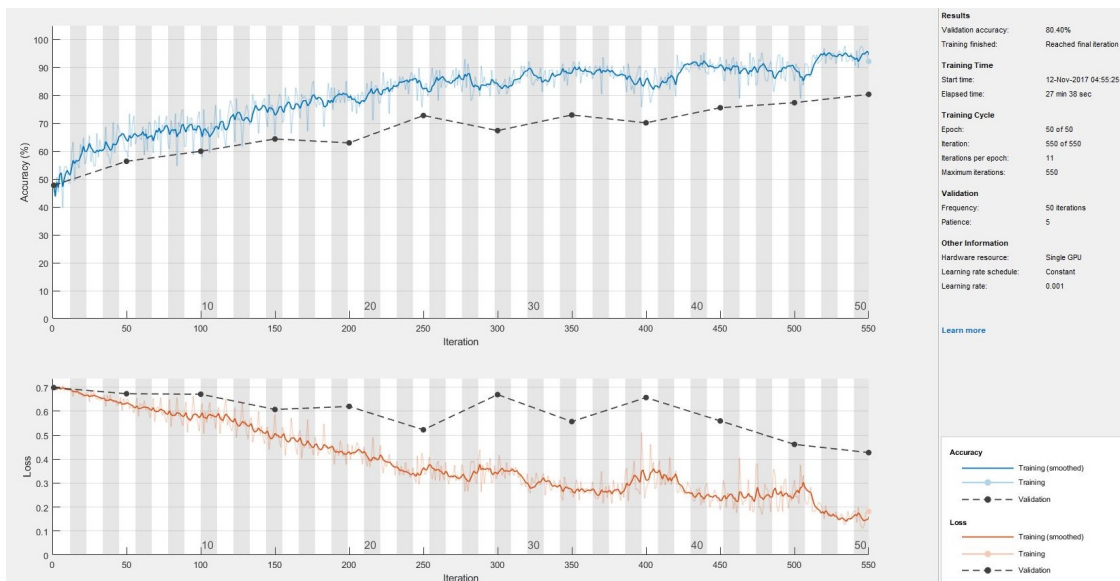


Figure A.5: The results of five convolutional layer model using 2000 patches with 50 epochs

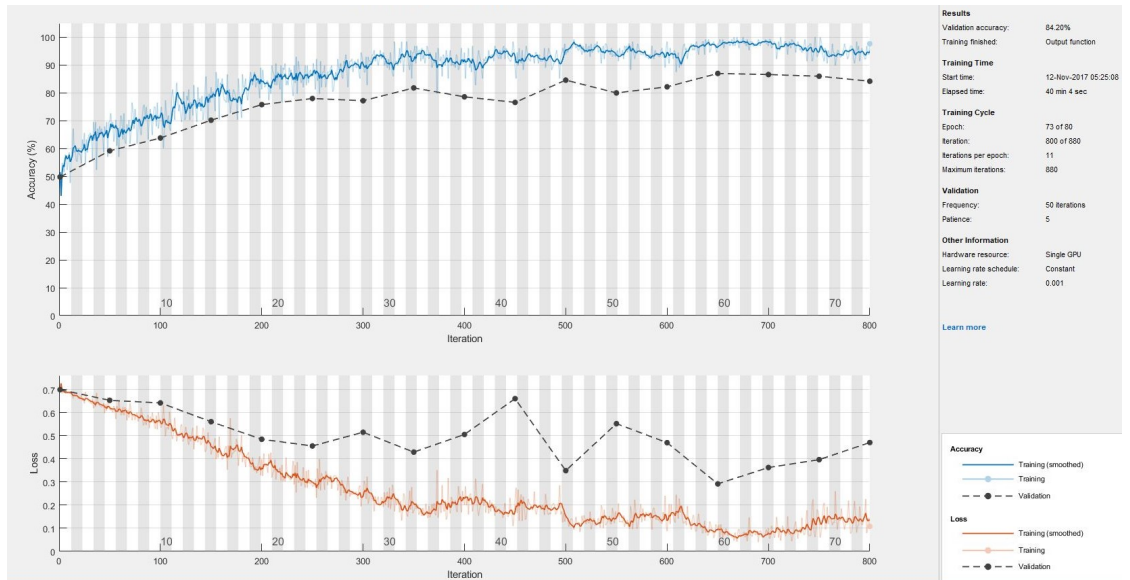


Figure A.6: The results of five convolutional layer model using 2000 patches with 80 epochs

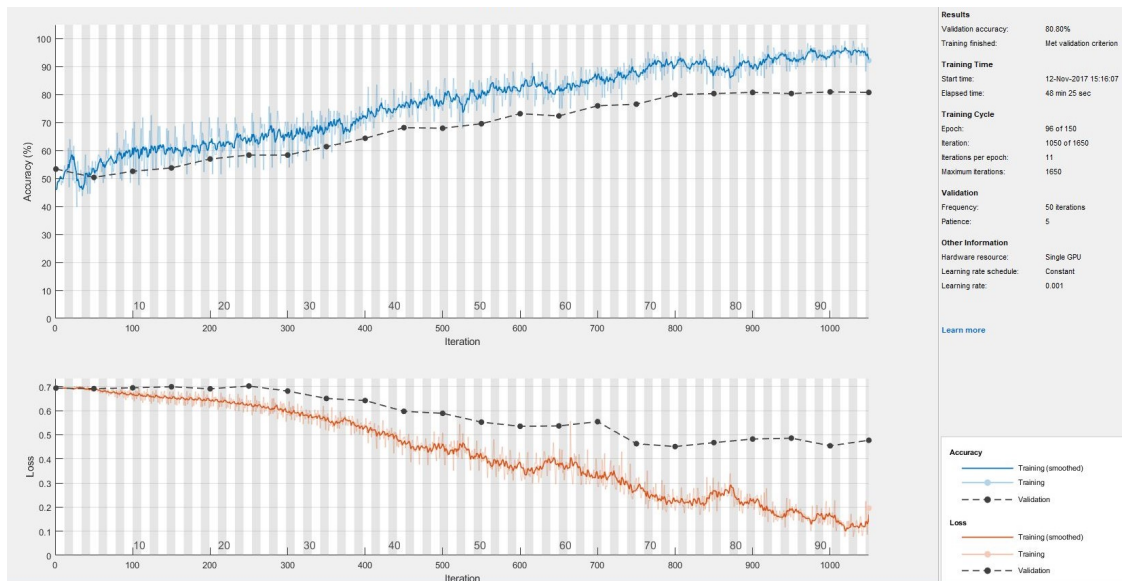


Figure A.7: The results of five convolutional layer model using 2000 patches with 150 epochs

Experiment 5

Now, we increased the number of patches to 12400 and tried different number of epochs to achieve better performance. Figure A.8 shows the results with 100 epoch

for which an accuracy of 68.48% is achieved. Figure A.9 shows results with 150 epoch and achieved 66.58% accuracy. The epochs increased to 200 which achieved the accuracy 69.41% of as illustrated in Figure A.10. Figure A.11 shows the results of 250 epochs which achieved the accuracy of 67.97%. We can see that the progress of validation accuracy stopped before it reach to the total number of epochs.

This shows that the number of patches were increased from 2000 to 12422 didn't affect the performance neither the number of epochs. In this case, the model needs more convolutional layers to extract more depth features.

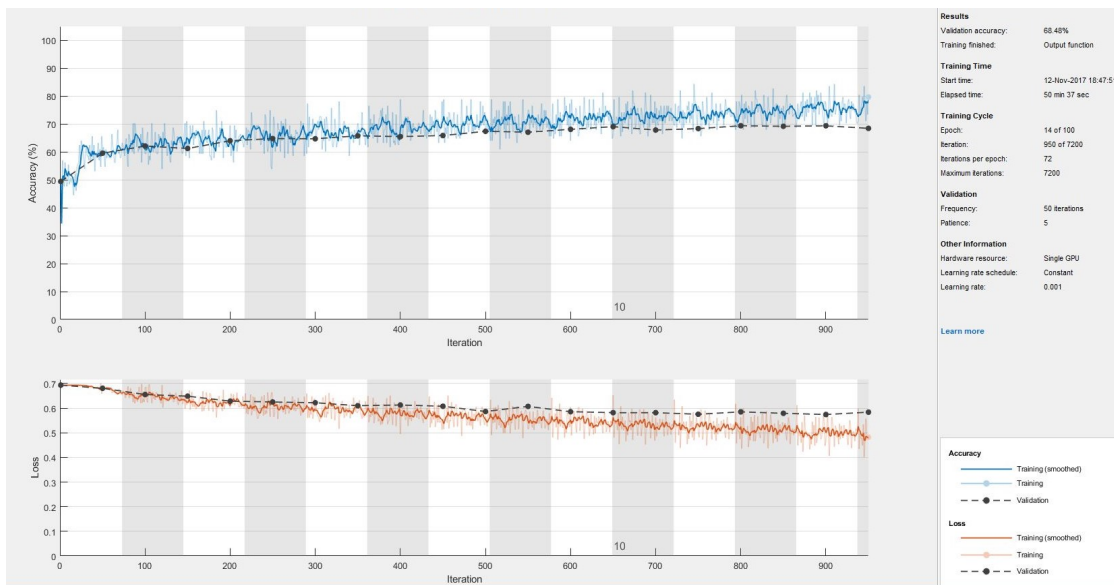


Figure A.8: The results of five convolutional layer model using 12422 patches with 100 epochs

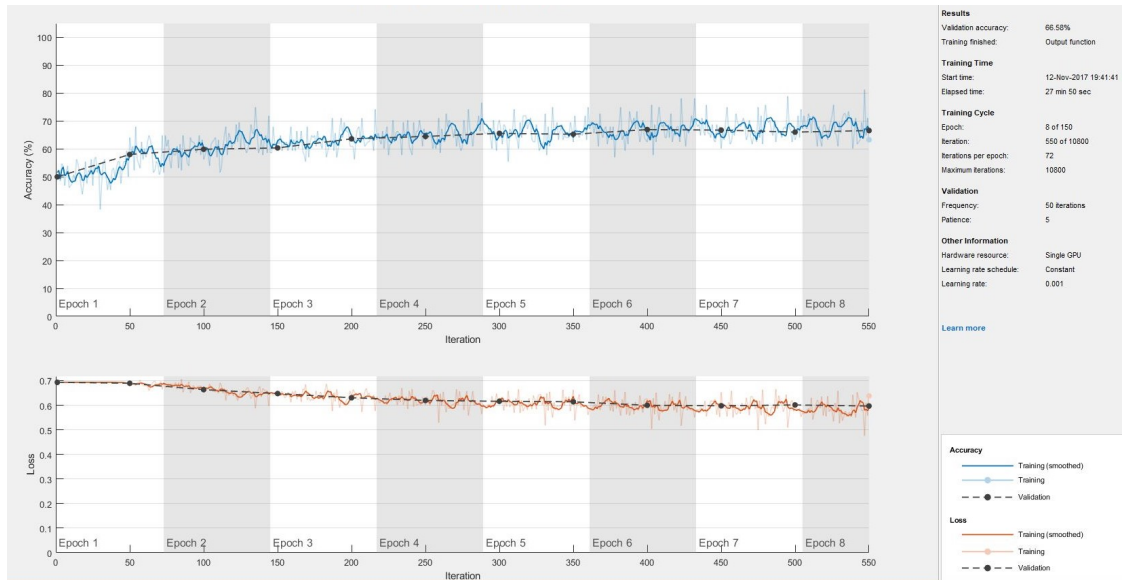


Figure A.9: The results of five convolutional layer model using 12422 patches with 150 epochs

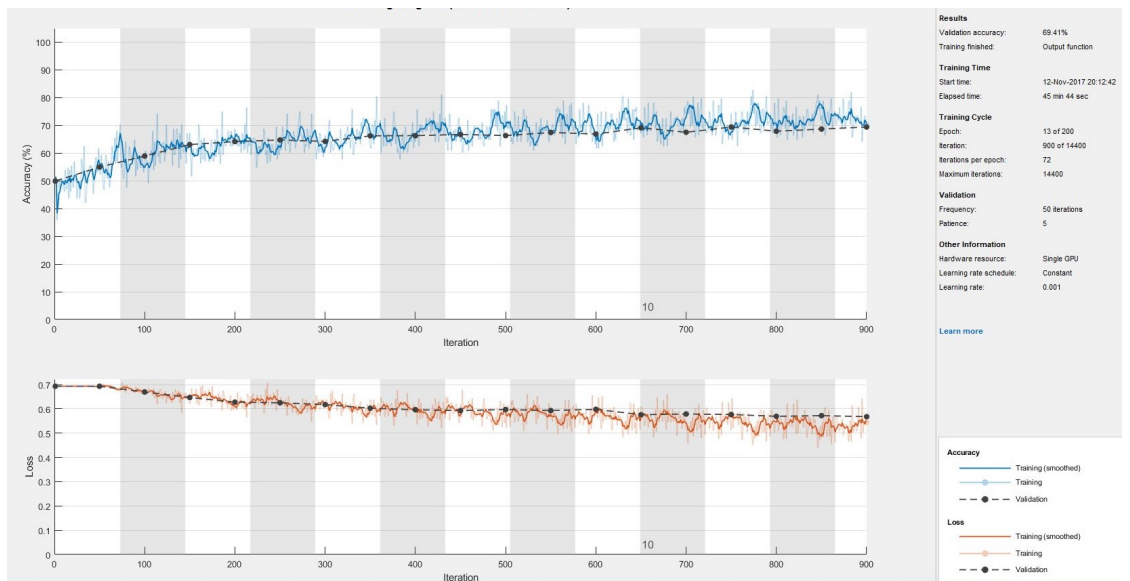


Figure A.10: The results of five convolutional layer model using 12422 patches with 200 epochs

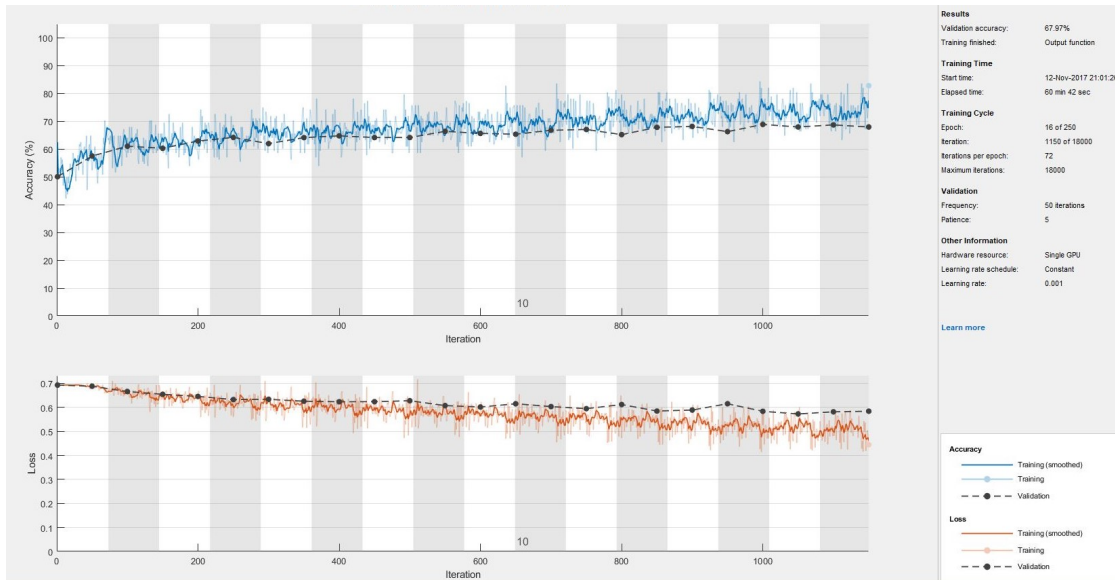


Figure A.11: The results of five convolutional layer model using 12422 patches with 250 epochs

Experiment 6

First, we need to fit the model with a small number of images. Therefore, the number of convolutional layers were increased to 8 which will extract more detail features. The weight initialization is also one of the main factors in CNN, so two different weights were used in this experiment: random Gaussian weights and SRM weights. The first layer was initialized to SRM weights while the rest of the layers were initialized to random Gaussian weights.

Figure A.12 shows the progress of validation accuracy and loss function using 2000 patches. The training curve shows clearly that the model fits accurately but the validation curve stops progressing which shows the model is over-fitting. This issue can be solved with regularization and adding more data for training. Figure A.13 shows the progress of the same model but the number of images was increased to

12400 which achieved an accuracy of 82.90%. By using the large number of patches (99376), Figure A.14 shows the validation accuracy dropped to 74.85% which is worst than the accuracy of small number of patches.

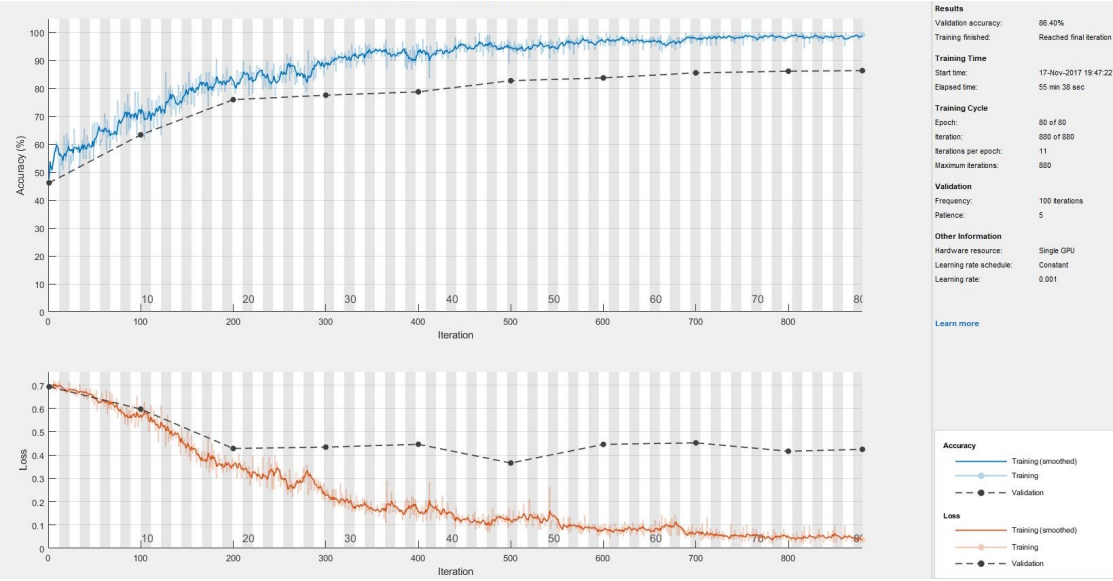


Figure A.12: The results of eight convolutional layer model using 2000 patches with SRM weights initialization

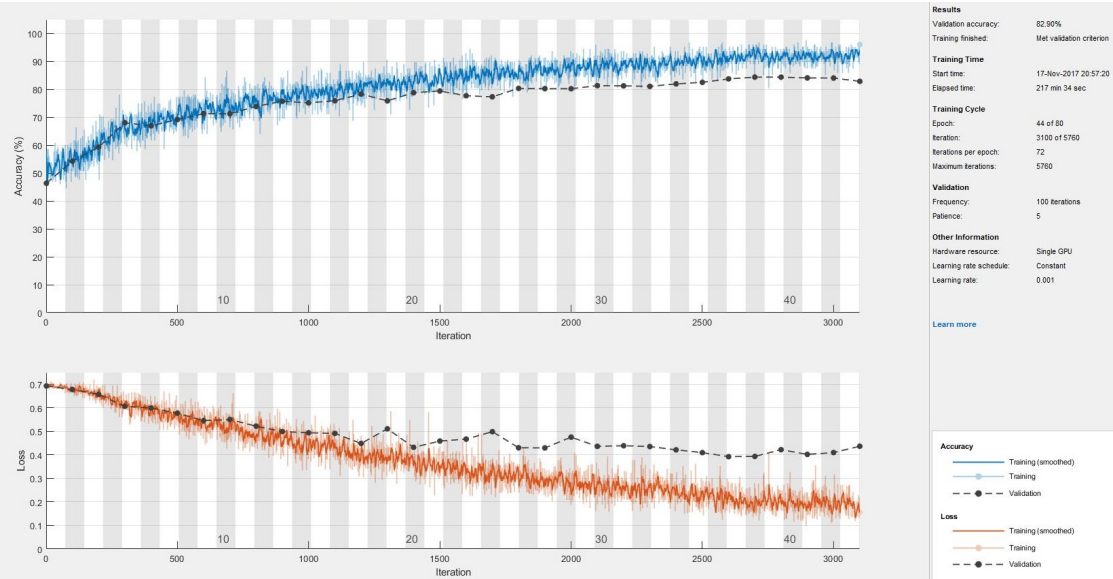


Figure A.13: The results of eight convolutional layer model using 12422 patches with SRM weights initialization

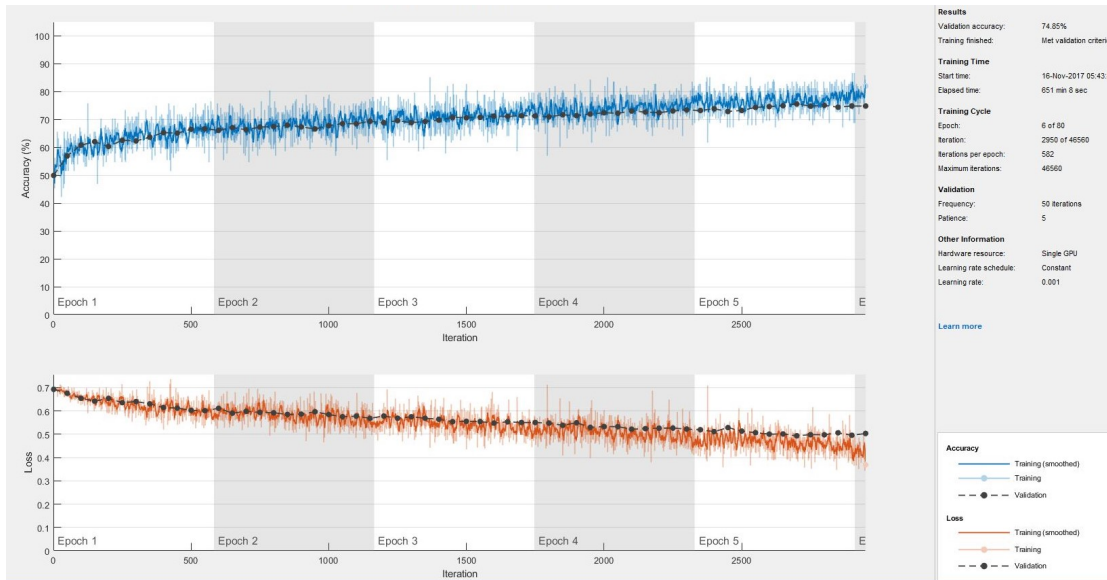


Figure A.14: The results of eight convolutional layer model using 99376 patches with random weights initialization

Experiment 7

The same eight convolutional layer model was used with the addition of regularization steps, which includes, L2-Regularization and Dropout layers. Instead of random weights, the SRM weights were initialized to the first convolutional layer while the rest of the layers were initialized randomly. The regularization steps improved the performance on large number of patches as compared to previous experiments. Figure A.15 shows the results for 2000 patches. Figure A.16 illustrates the results for 12422 patches and Figure A.17 shows the results for 99376 patches.

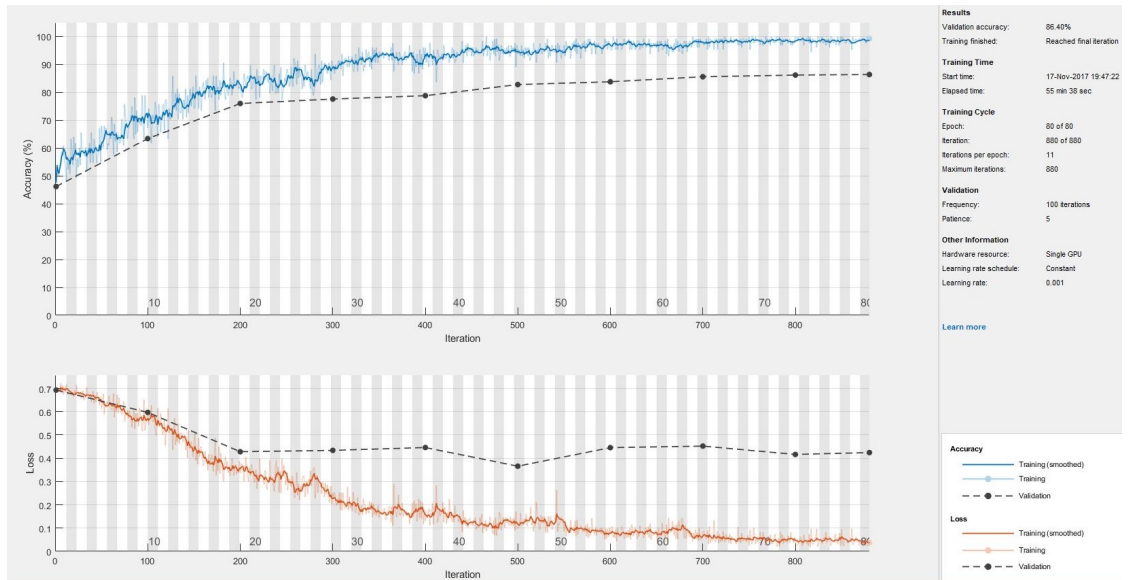


Figure A.15: The results of eight convolutional layer model using 2000 patches with regularization and SRM weights initialization

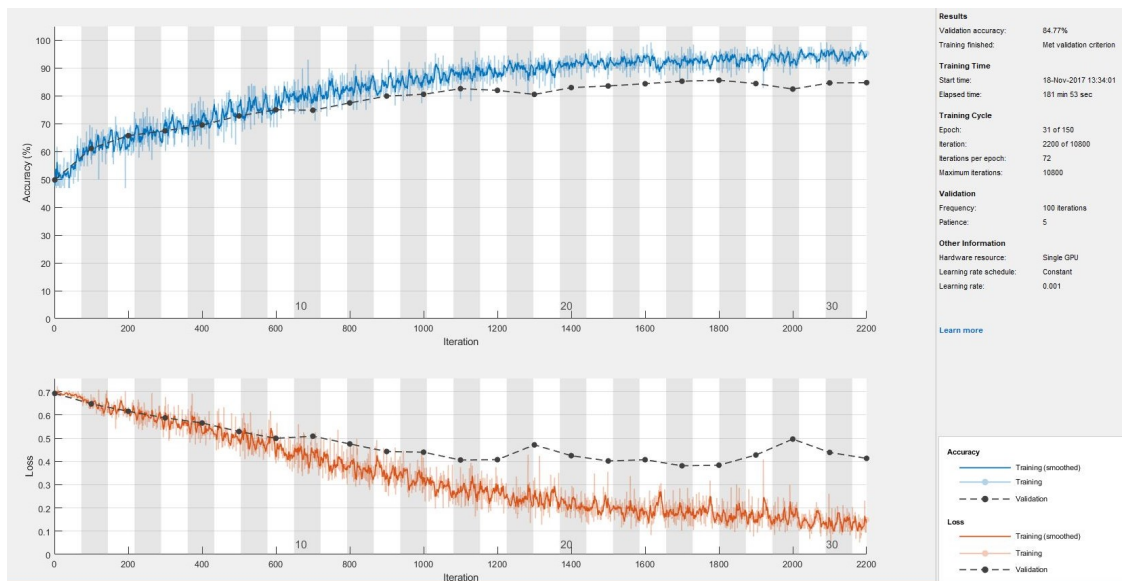


Figure A.16: The results of eight convolutional layer model using 12422 patches with regularization and SRM weights initialization

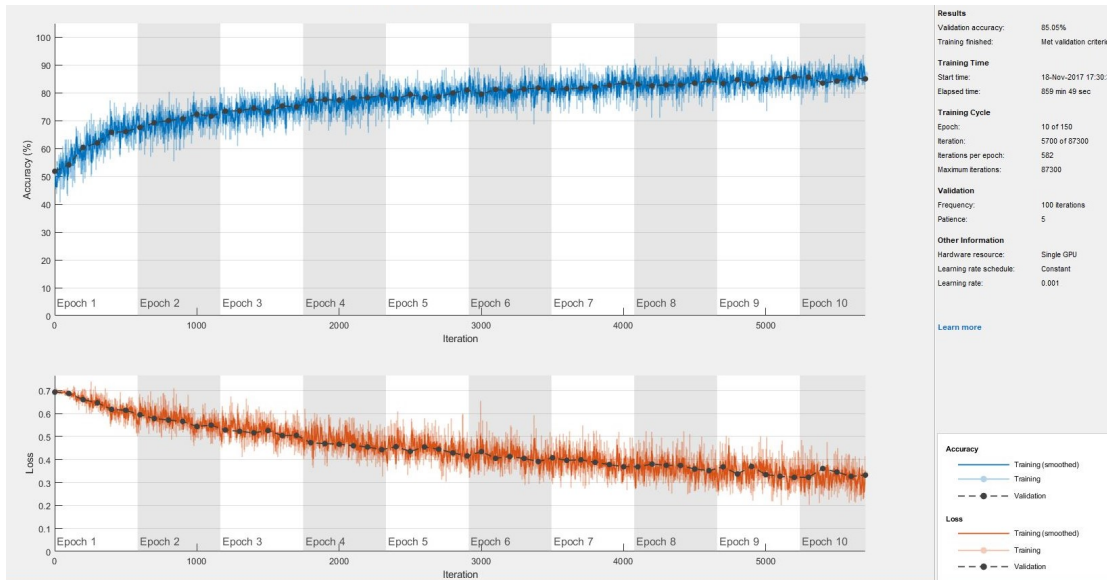


Figure A.17: The results of eight convolutional layer model using 99376 patches with regularization and SRM weights initialization

Experiment 8

Two more convolutional layers were added to the previous model which increased the number of convolutional layers to 10. The model achieved better performance with a large number of patches and achieved the highest accuracy above 90%. Figure A.18 shows the results for 2000 patches which achieved 89.80% accuracy. The experimental results for 12422 patches are illustrated in Figure A.19 which achieved 81.52% accuracy. Using 99376 patches, the model achieved an accuracy of 90.91% as illustrated in Figure A.20.

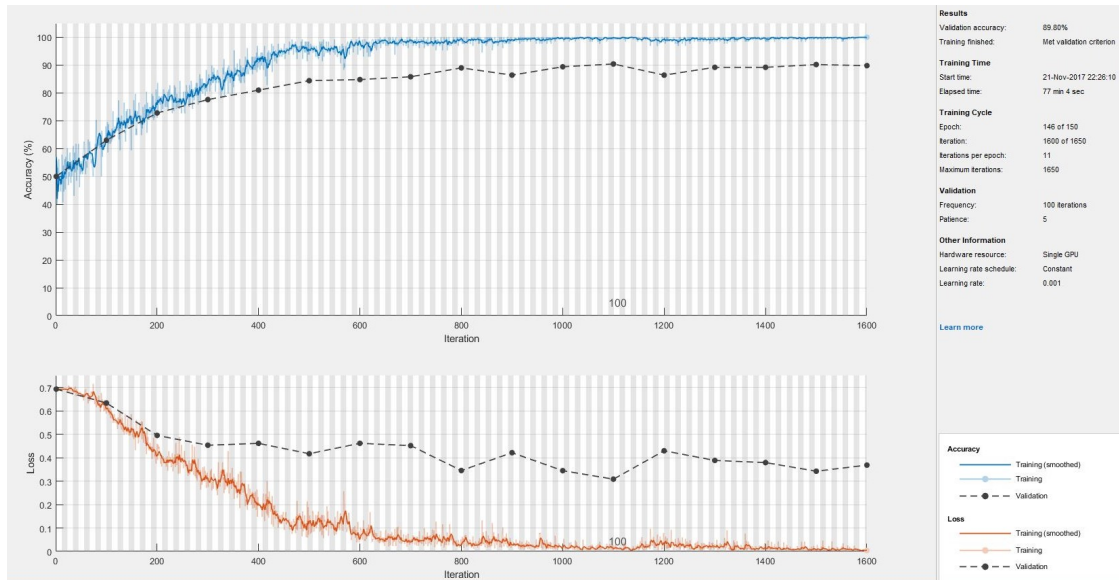


Figure A.18: The results of ten convolutional layer model using 2000 patches with regularization and SRM weights initialization

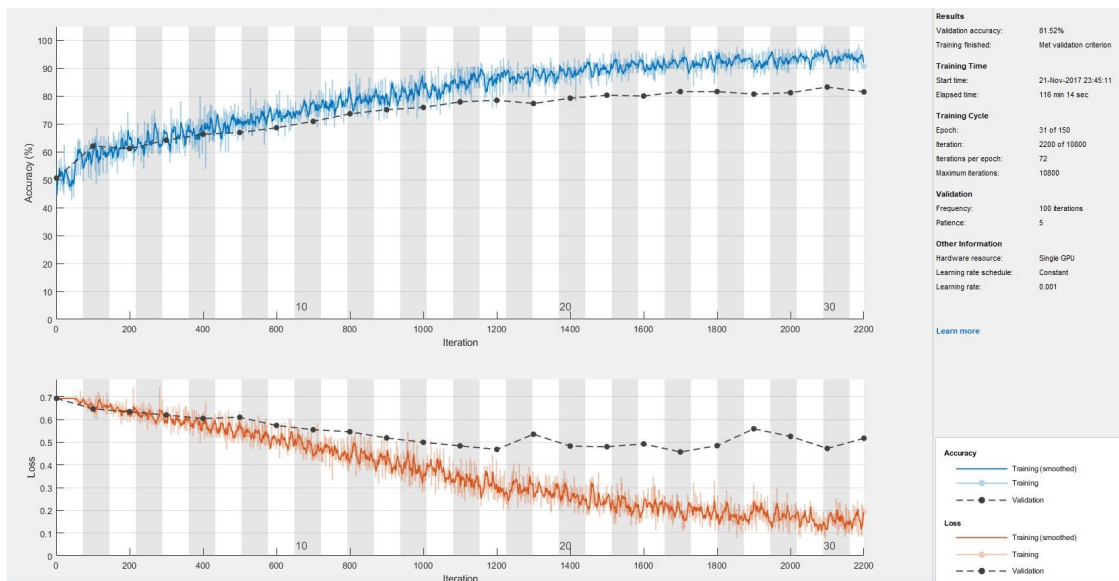


Figure A.19: The results of ten convolutional layer model using 12422 patches with regularization and SRM weights initialization

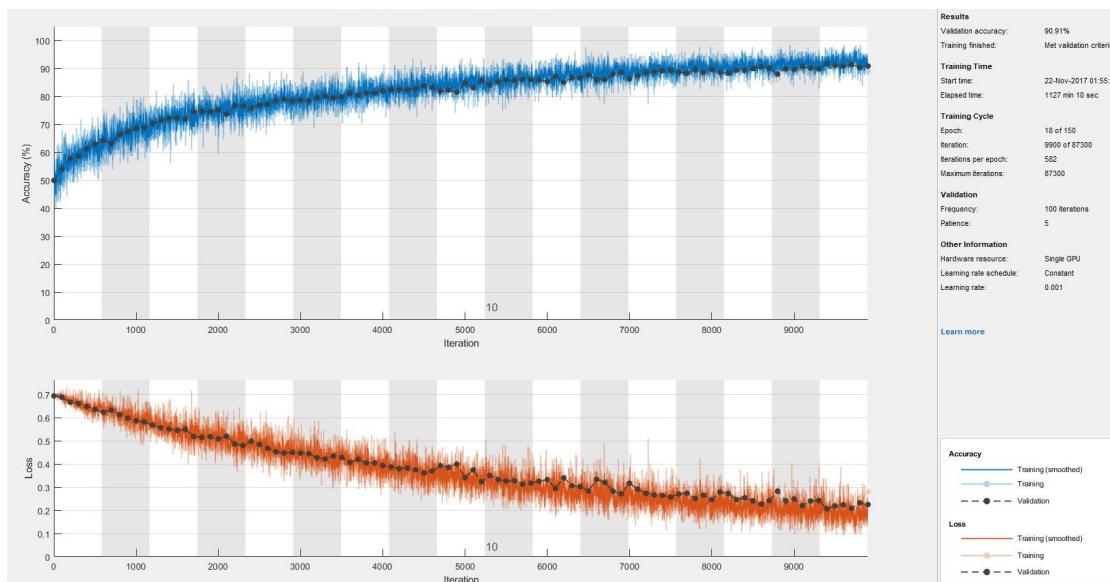


Figure A.20: The results of ten convolutional layer model using 99376 patches with regularization and SRM weights initialization

APPENDIX B

PUBLICATIONS

B.1 Published

- Atif Shah, and El-Sayed El-Alfy, “Image Splicing Forgery Detection Using DCT Coefficients with Multi-Scale LBP.” in IEEE International Conference on Computing Sciences and Engineering (ICCSE), Kuwait, 2018. (Conference)
- Atif Shah, and El-Sayed El-Alfy, “Multi-scale LPQ-DCT for Image Forgery Detection.” in The 8th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), Bahrain, 2019. (Conference)

B.2 Under Preparation

- Atif Shah, and El-Sayed El-Alfy, “Comparative Analysis of Feature Extraction Methods for Image Forgery Detection Using Chroma Channels.” (Journal)

REFERENCES

- [1] M. Heikkilä, M. Pietikäinen, and C. Schmid, “Description of interest regions with local binary patterns,” *Pattern Recognition*, vol. 42, no. 3, pp. 425–436, 2009.
- [2] T. Mäenpää and M. Pietikäinen, “Multi-scale binary patterns for texture analysis,” *Image Analysis*, pp. 267–275, 2003.
- [3] H. Farid, “Image forgery detection,” *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 16–25, 2009.
- [4] G. K. Birajdar and V. H. Mankar, “Digital image forgery detection using passive techniques: A survey,” *Digital Investigation*, vol. 10, no. 3, pp. 226–245, 2013.
- [5] M. A. Qureshi and M. Deriche, “A bibliography of pixel-based blind image forgery detection techniques,” *Signal Processing: Image Communication*, vol. 39, pp. 46–74, 2015.
- [6] N. B. A. Warif, A. W. A. Wahab, M. Y. I. Idris, R. Ramli, R. Salleh, S. Shamshirband, and K.-K. R. Choo, “Copy-move forgery detection: survey, challenges and

- future directions,” *Journal of Network and Computer Applications*, vol. 75, pp. 259–278, 2016.
- [7] P. Ferrara, T. Bianchi, A. De Rosa, and A. Piva, “Image forgery localization via fine-grained analysis of cfa artifacts,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1566–1577, 2012.
- [8] M. Zampoglou, S. Papadopoulos, and Y. Kompatsiaris, “Large-scale evaluation of splicing localization algorithms for web images,” *Multimedia Tools and Applications*, pp. 1–34, 2016.
- [9] G. Chierchia, G. Poggi, C. Sansone, and L. Verdoliva, “A bayesian-mrf approach for prnu-based image forgery detection,” *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 4, pp. 554–567, 2014.
- [10] M. K. Johnson and H. Farid, “Exposing digital forgeries by detecting inconsistencies in lighting,” in *Proceedings of the 7th Workshop on Multimedia and Security*. ACM, 2005, pp. 1–10.
- [11] S.-C. Wang, “Artificial neural network,” in *Interdisciplinary Computing in Java Programming*. Springer, 2003, pp. 81–100.
- [12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

- [13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.
- [14] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European Conference on Computer Vision*. Springer, 2014, pp. 818–833.
- [15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
- [16] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [17] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [18] M. Zampoglou, S. Papadopoulos, and Y. Kompatsiaris, “Detecting image splicing in the wild (web),” in *IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2015, pp. 1–6.
- [19] D. Cozzolino, D. Gragnaniello, and L. Verdoliva, “Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques,”

- in *IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 5302–5306.
- [20] Y. Zhang, C. Zhao, F. Zhao, and S. Li, “Blind forensics of median filtering based on markov statistics in median-filtered residual domain,” in *The Proceedings of the Second International Conference on Communications, Signal Processing, and Systems*. Springer, 2014, pp. 179–186.
- [21] S. Farooq, M. H. Yousaf, and F. Hussain, “A generic passive image forgery detection scheme using local binary pattern with rich models,” *Computers & Electrical Engineering*, vol. 62, pp. 459–472, 2017.
- [22] J. Fridrich and J. Kodovsky, “Rich models for steganalysis of digital images,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, 2012.
- [23] K. Bahrami, A. C. Kot, L. Li, and H. Li, “Blurred image splicing localization by exposing blur type inconsistency,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 5, pp. 999–1009, 2015.
- [24] M. Jenadeleh and M. Ebrahimi Moghaddam, “Blind detection of region duplication forgery using fractal coding and feature matching,” *Journal of Forensic Sciences*, vol. 61, no. 3, pp. 623–636, 2016.
- [25] E. Silva, T. Carvalho, A. Ferreira, and A. Rocha, “Going deeper into copy-move forgery detection: Exploring image telltales via multi-scale analysis and voting

- processes,” *Journal of Visual Communication and Image Representation*, vol. 29, pp. 16–32, 2015.
- [26] V. Christlein, C. Riess, J. Jordan, C. Riess, and E. Angelopoulou, “An evaluation of popular copy-move forgery detection approaches,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1841–1854, 2012.
- [27] X. Zhao, J. Li, S. Li, and S. Wang, “Detecting digital image splicing in chroma spaces,” in *International Workshop on Digital Watermarking*. Springer, 2010, pp. 12–22.
- [28] H. Zhou, Y. Shen, X. Zhu, B. Liu, Z. Fu, and N. Fan, “Digital image modification detection using color information and its histograms,” *Forensic Science International*, vol. 266, pp. 379–388, 2016.
- [29] P. Sutthiwan, Y.-Q. Shi, J. Dong, T. Tan, and T.-T. Ng, “New developments in color image tampering detection,” in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010, pp. 3064–3067.
- [30] P. Sutthiwan, Y. Q. Shi, W. Su, and T.-T. Ng, “Rake transform and edge statistics for image forgery detection,” in *IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2010, pp. 1463–1468.
- [31] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, “A sift-based forensic method for copy-move attack detection and transformation recovery,” *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1099–1110, 2011.

- [32] B. Ustubioglu, G. Muzafer, G. Ulutas, V. Nabiyev, and M. Ulutas, “A novel keypoint based forgery detection method based on local phase quantization and sift,” in *9th IEEE International Conference on Electrical and Electronics Engineering (ELECO)*, 2015, pp. 185–189.
- [33] R. Ahilapriyadharshini, S. Arivazhagan, and M. Gowthami, “Weber local descriptor based object recognition,” in *IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, 2012, pp. 115–119.
- [34] M. Hussain, G. Muhammad, S. Q. Saleh, A. M. Mirza, and G. Bebis, “Image forgery detection using multi-resolution weber local descriptors,” in *EUROCON, 2013 IEEE*. IEEE, 2013, pp. 1570–1577.
- [35] O. Mayer and M. C. Stamm, “Accurate and efficient image forgery detection using lateral chromatic aberration,” *Transactions on Information Forensics and Security*, 2018.
- [36] M. Emam, Q. Han, Q. Li, and H. Zhang, “A robust detection algorithm for image copy-move forgery in smooth regions,” in *International Conference on Circuits, System and Simulation (ICCSS)*. IEEE, 2017, pp. 119–123.
- [37] A. Ghoneim, G. Muhammad, S. U. Amin, and B. Gupta, “Medical image forgery detection for smart healthcare,” *IEEE Communications Magazine*, vol. 56, no. 4, pp. 33–37, 2018.

- [38] C.-L. Chou and J.-C. Lee, “Copy-move forgery detection based on local gabor wavelets patterns,” in *International Conference on Security with Intelligent Computing and Big-data Services*. Springer, 2017, pp. 47–56.
- [39] M. Abdel-Basset, G. Manogaran, A. E. Fakhry, and I. El-Henawy, “2-levels of clustering strategy to detect and locate copy-move forgery in digital images,” *Multimedia Tools and Applications*, pp. 1–19, 2018.
- [40] C. S. Prakash, S. Maheshkar, and V. Maheshkar, “Image manipulation detection using harris corner and anms,” in *Advances in Machine Learning and Data Science*. Springer, 2018, pp. 81–93.
- [41] D. Fu, Y. Q. Shi, and W. Su, “Detection of image splicing based on hilbert-huang transform and moments of characteristic functions with wavelet decomposition,” in *International Workshop on Digital Watermarking*. Springer, 2006, pp. 177–187.
- [42] X. Li, T. Jing, and X. Li, “Image splicing detection based on moment features and hilbert-huang transform,” in *IEEE International Conference on Information Theory and Information Security (ICITIS)*, 2010, pp. 1127–1130.
- [43] J. Dong, W. Wang, T. Tan, Y. Q. Shi *et al.*, “Run-length and edge statistics based approach for image splicing detection.” in *IWDW*. Springer, 2008, pp. 76–87.
- [44] W. Chen, Y. Q. Shi, and W. Su, “Image splicing detection using 2-d phase congruency and statistical moments of characteristic function,” in *Society of*

- Photo-optical Instrumentation Engineers (SPIE) conference series*, vol. 6505, 2007, p. 26.
- [45] Y. Q. Shi, C. Chen, and W. Chen, “A natural image model approach to splicing detection,” in *Proceedings of the 9th Workshop on Multimedia & Security*. ACM, 2007, pp. 51–62.
 - [46] —, “A markov process based approach to effective attacking jpeg steganography,” in *International Workshop on Information Hiding*. Springer, 2006, pp. 249–264.
 - [47] A. Alahmadi, M. Hussain, H. Aboalsamh, G. Muhammad, G. Bebis, and H. Mathkour, “Passive detection of image forgery using dct and local binary pattern,” *Signal, Image and Video Processing*, pp. 1–8, 2016.
 - [48] T.-T. Ng, S.-F. Chang, and Q. Sun, “A data set of authentic and spliced image blocks,” *Columbia University, ADVENT Technical Report*, pp. 203–2004, 2004.
 - [49] Q. Zhang, W. Lu, and J. Weng, “Joint image splicing detection in dct and contourlet transform domain,” *Journal of Visual Communication and Image Representation*, vol. 40, pp. 449–458, 2016.
 - [50] H.-C. Hsu and M.-S. Wang, “Detection of copy-move forgery image using gabor descriptor,” in *IEEE Anti-counterfeiting, Security, and Identification*, 2012, pp. 1–4.
 - [51] G. Griffin, A. Holub, and P. Perona, “Caltech-256 object category dataset,” 2007.

- [52] E.-S. M. El-Alfy and M. A. Qureshi, “Combining spatial and dct based markov features for enhanced blind detection of image splicing,” *Pattern Analysis and Applications*, vol. 18, no. 3, pp. 713–723, 2015.
- [53] —, “Robust content authentication of gray and color images using lbp-dct markov-based features,” *Multimedia Tools and Applications*, pp. 1–22, 2016.
- [54] L. Yu, Q. Han, X. Niu, S. Yiu, J. Fang, and Y. Zhang, “An improved parameter estimation scheme for image modification detection based on dct coefficient analysis,” *Forensic Science International*, vol. 259, pp. 200–209, 2016.
- [55] G. Schaefer and M. Stich, “Ucid: an uncompressed color image database,” in *Electronic Imaging*. International Society for Optics and Photonics, 2003, pp. 472–480.
- [56] Y. Zhang, C. Zhao, Y. Pi, and S. Li, “Revealing image splicing forgery using local binary patterns of dct coefficients,” in *Communications, Signal Processing, and Systems*. Springer, 2012, pp. 181–189.
- [57] Z. Zhang, J. Kang, and Y. Ren, “An effective algorithm of image splicing detection,” in *IEEE International Conference on Computer Science and Software Engineering*, vol. 1, 2008, pp. 1035–1039.
- [58] W. Wang, J. Dong, and T. Tan, “Image tampering detection based on stationary distribution of markov chain,” in *17th IEEE International Conference on Image Processing (ICIP)*, 2010, pp. 2101–2104.

- [59] Z. He, W. Lu, W. Sun, and J. Huang, “Digital image splicing detection based on markov features in dct and dwt domain,” *Pattern Recognition*, vol. 45, no. 12, pp. 4292–4299, 2012.
- [60] G. Muhammad, M. H. Al-Hammadi, M. Hussain, and G. Bebis, “Image forgery detection using steerable pyramid transform and local binary pattern,” *Machine Vision and Applications*, vol. 25, no. 4, pp. 985–995, 2014.
- [61] G. K. Birajdar and V. H. Mankar, “Passive method for rescale detection using quadrature mirror filter based higher order statistical features,” *International Journal of Wavelets, Multiresolution and Information Processing*, vol. 14, no. 05, p. 1650033, 2016.
- [62] K. Hayat and T. Qazi, “Forgery detection in digital images via discrete wavelet and discrete cosine transforms,” *Computers & Electrical Engineering*, vol. 62, pp. 448–458, 2017.
- [63] J. A. Aghamaleki and A. Behrad, “Inter-frame video forgery detection and localization using intrinsic effects of double compression on quantization errors of video coding,” *Signal Processing: Image Communication*, vol. 47, pp. 289–302, 2016.
- [64] T. Mahmood, T. Nawaz, A. Irtaza, R. Ashraf, M. Shah, and M. T. Mahmood, “Copy-move forgery detection technique for forensic analysis in digital images,” *Mathematical Problems in Engineering*, vol. 2016, 2016.

- [65] T. Mahmood, A. Irtaza, Z. Mehmood, and M. T. Mahmood, “Copy-move forgery detection through stationary wavelets and local binary pattern variance for forensic analysis in digital images,” *Forensic Science International*, vol. 279, pp. 8–21, 2017.
- [66] M. M. Isaac and M. Wilscy, “Multiscale local gabor phase quantization for image forgery detection,” *Multimedia Tools and Applications*, vol. 76, no. 24, pp. 25 851–25 872, 2017.
- [67] M. H. Alkawaz, G. Sulong, T. Saba, and A. Rehman, “Detection of copy-move image forgery based on discrete cosine transform,” *Neural Computing and Applications*, vol. 30, no. 1, pp. 183–192, 2018.
- [68] S. Agarwal and S. Chand, “Image forgery detection using co-occurrence-based texture operator in frequency domain,” in *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*. Springer, 2018, pp. 117–122.
- [69] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, “Exploring strategies for training deep neural networks,” *Journal of Machine Learning Research*, vol. 10, no. Jan, pp. 1–40, 2009.
- [70] H. Lee, C. Ekanadham, and A. Y. Ng, “Sparse deep belief net model for visual area v2,” in *Advances in Neural Information Processing Systems*, 2008, pp. 873–880.

- [71] Y. Rao and J. Ni, “A deep learning approach to detection of splicing and copy-move forgeries in images,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2016, pp. 1–6.
- [72] J. Zhou, J. Ni, and Y. Rao, “Block-based convolutional neural network for image forgery detection,” in *International Workshop on Digital Watermarking*. Springer, 2017, pp. 65–76.
- [73] Y. Liu, Q. Guan, X. Zhao, and Y. Cao, “Image forgery localization based on multi-scale convolutional neural networks,” *arXiv preprint arXiv:1706.07842*, 2017.
- [74] B. Bayar and M. C. Stamm, “A deep learning approach to universal image manipulation detection using a new convolutional layer,” in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2016, pp. 5–10.
- [75] D. Cozzolino and L. Verdoliva, “Single-image splicing localization through autoencoder-based anomaly detection,” in *IEEE International Workshop on Information Forensics and Security (WIFS)*, 2016, pp. 1–6.
- [76] J. Chen, X. Kang, Y. Liu, and Z. J. Wang, “Median filtering forensics based on convolutional neural networks,” *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1849–1853, 2015.
- [77] J. Bunk, J. H. Bappy, T. M. Mohammed, L. Nataraj, A. Flenner, B. Manjunath, S. Chandrasekaran, A. K. Roy-Chowdhury, and L. Peterson, “Detection and

- localization of image forgeries using resampling features and deep learning,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1881–1889.
- [78] T. M. Mohammed, J. Bunk, L. Nataraj, J. H. Bappy, A. Flenner, B. Manjunath, S. Chandrasekaran, A. K. Roy-Chowdhury, and L. A. Peterson, “Boosting image forgery detection using resampling features and copy-move analysis,” *Electronic Imaging*, vol. 2018, no. 7, pp. 1–7, 2018.
- [79] D. Cozzolino, G. Poggi, and L. Verdoliva, “Efficient dense-field copy-move forgery detection,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 11, pp. 2284–2297, 2015.
- [80] Y. Zhanga, J. Goha, L. L. Wina, and V. Thinga, “Image region forgery detection: A deep learning approach,” in *Proceedings of the Singapore Cyber-Security Conference : Cyber-Security by Design*, 2016.
- [81] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *Advances in Neural Information Processing Systems*, 2014, pp. 3320–3328.
- [82] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “Cnn features off-the-shelf: an astounding baseline for recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 806–813.

- [83] O. Lucena, A. Junior, V. Moia, R. Souza, E. Valle, and R. Lotufo, “Transfer learning using convolutional neural networks for face anti-spoofing,” in *International Conference Image Analysis and Recognition*. Springer, 2017, pp. 27–34.
- [84] H.-W. Ng, V. D. Nguyen, V. Vonikakis, and S. Winkler, “Deep learning for emotion recognition on small datasets using transfer learning,” in *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, pp. 443–449.
- [85] Y. Zhan, Y. Chen, Q. Zhang, and X. Kang, “Image forensics based on transfer learning and convolutional neural network,” in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2017, pp. 165–170.
- [86] “Nrcs photo gallery <http://photogallery.nrcs.usda.gov/res/sites/photogallery/>.”
- [87] M. Heath, K. Bowyer, D. Kopans, R. Moore, and W. P. Kegelmeyer, “The digital database for screening mammography,” in *Proceedings of the 5th international workshop on digital mammography*. Medical Physics Publishing, 2000, pp. 212–218.
- [88] T. J. De Carvalho, C. Riess, E. Angelopoulou, H. Pedrini, and A. de Rezende Rocha, “Exposing digital image forgeries by illumination color classification,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 7, pp. 1182–1194, 2013.

- [89] A. A. Alahmadi, M. Hussain, H. Aboalsamh, G. Muhammad, and G. Bebis, "Splicing image forgery detection based on dct and local binary pattern," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2013, pp. 253–256.
- [90] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.
- [91] T. Lindahl, "Study of local binary patterns," 2007.
- [92] F. Z. Chelali and A. Djeradi, "Cslbp and oclbp local descriptors for speaker identification from video sequences," in *Third World Conference on Complex Systems (WCCS)*. IEEE, 2015, pp. 1–7.
- [93] G. Kylberg and I.-M. Sintorn, "Evaluation of noise robustness for local binary pattern descriptors in texture classification," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, p. 17, 2013.
- [94] J. Chen, S. Shan, C. He, G. Zhao, M. Pietikainen, X. Chen, and W. Gao, "Wld: A robust local image descriptor," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1705–1720, 2010.
- [95] V. Ojansivu and J. Heikkilä, "Blur insensitive texture classification using local phase quantization," in *International Conference on Image and Signal Processing*. Springer, 2008, pp. 236–243.

- [96] Z. Zhang, F. Li, M. Liu, and P. K. Yadav, “Image matching based on local phase quantization applied for measuring the tensile properties of high elongation materials,” *Mathematical Problems in Engineering*, vol. 2016, 2016.
- [97] A. K. Jain, *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall,, 1989.
- [98] Q. Lin and W. Qi, “Multi-scale local binary patterns based on path integral for texture classification,” in *IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 26–30.
- [99] R. Davarzani, K. Yaghmaie, S. Mozaffari, and M. Tapak, “Copy-move forgery detection using multiresolution local binary patterns,” *Forensic Science International*, vol. 231, no. 1, pp. 61–72, 2013.
- [100] L. I. Smith, “A tutorial on principal components analysis,” Tech. Rep., 2002.
- [101] Y. Sun, S. Todorovic, and S. Goodison, “Local-learning-based feature selection for high-dimensional data analysis,” *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1610–1626, 2010.
- [102] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [103] J. Platt *et al.*, “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods,” *Advances in Large Margin Classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

- [104] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.
- [105] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, “Understanding neural networks through deep visualization,” *arXiv preprint arXiv:1506.06579*, 2015.
- [106] L. M. Zintgraf, T. S. Cohen, and M. Welling, “A new method to visualize deep neural networks,” *arXiv preprint arXiv:1603.02518*, 2016.
- [107] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” *arXiv preprint arXiv:1412.6806*, 2014.
- [108] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.
- [109] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv preprint arXiv:1207.0580*, 2012.
- [110] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.
- [111] J. Wang and L. Perez, “The effectiveness of data augmentation in image classification using deep learning,” Technical Report, Tech. Rep., 2017.

- [112] M. H. Al-Hammadi, G. Muhammad, M. Hussain, and G. Bebis, “Curvelet transform and local texture based image forgery detection,” in *International Symposium on Visual Computing*. Springer, 2013, pp. 503–512.
- [113] S. Q. Saleh, M. Hussain, G. Muhammad, and G. Bebis, “Evaluation of image forgery detection using multi-scale weber local descriptors,” in *International Symposium on Visual Computing*. Springer, 2013, pp. 416–424.
- [114] M. Hussain, S. Q. Saleh, H. Aboalsamh, G. Muhammad, and G. Bebis, “Comparison between wld and lbp descriptors for non-intrusive image forgery detection,” in *Proceedings of International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, 2014, pp. 197–204.

VITAE

- Name: Atif Shah
- Nationality: Pakistani
- Date of Birth: 17th Februry, 1991
- Email: *atifshahcs@gmail.com*
- Permenant Address: Charsadda, Khyber Pakhtunkhwa, Pakistan
- Zip/Postal Code: 21420
- Thesis Publication 1: Atif Shah, and El-Sayed El-Alfy, “Image Splicing Forgery Detection Using DCT Coefficients with Multi-Scale LBP.” in IEEE International Conference on Computing Sciences and Engineering (ICCSE), Kuwait, 2018. (Conference)
- Thesis Publication 2: Atif Shah, and El-Sayed El-Alfy, “Multi-scale LPQ-DCT for Image Forgery Detection.” in The 8th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), Bahrain, 2019. (Conference)
- Thesis Publication 3: Atif Shah, and El-Sayed El-Alfy, “Comparative Analysis of Feature Extraction Methods for Image Forgery Detection Using Chroma

Channels.” (Journal) [To be submitted]