Transfer Learning Effects on Image Steganalysis with Pre-Trained Deep Residual Neural Network Model

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

Selim ÖZCAN

in partial fulfillment for the
degree of Master of Science

in
Cybersecurity Engineering

İSTANBUL
ŞEHİR
UNIVERSITY

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Cybersecurity Engineering.

APPROVED BY:

Prof. Ensar Gül (Thesis Advisor)	....[illegible]....
Dr. Ahmet Fatih Mustaçoğlu (Thesis Co-advisor)	....Fatih....
Asst. Prof. Mehmet Baysan	....[illegible]....
Assoc. Prof. Borahan Tümer	....[illegible]....

This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences of İstanbul Şehir University:

DATE OF APPROVAL: 10.09.2019

SEAL/SIGNATURE: [illegible]

Declaration of Authorship

I, Selim ÖZCAN, declare that this thesis titled, 'Transfer Learning Effects on Image Steganalysis with Pre-Trained Deep Residual Neural Network Model ' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: [illegible]

Date: 10.05.2019

"Out of love, Sacrifice is born, Hate is born, And we are able to know pain. "

Pain

Transfer Learning Effects on Image Steganalysis with Pre-Trained Deep Residual Neural Network Model

Selim ÖZCAN

Abstract

Steganalysis researches for the techniques used to reveal the embedded messages that is hidden in a digital medium -in most cases in images. The research and development activities in Image Steganalysis has gained more traction in recent years. Although machine learning techniques have been used for many years Deep Learning is a new paradigm for the Image Steganalysis domain. The success of the deep learning process is based on the training of the model for a sufficient amount of and with a high quality, diverse and large-scale data set. When the training process lacks dataset in terms of quality, variety and quantity, Transfer Learning emerges as an effective solution from Deep Learning methods. In Transfer Learning, an untrained model benefits from a previously trained model and its dataset. Base function is defined to transfer the parameters from the trained model to the untrained model. Hence, it would increase the success of deep learning model on Image Steganalysis. In this work, we compare the results of two series of models that are trained both with and without Transfer Learning method. The optimization method of the model training process is selected as experimental AdamW optimization method. Comparison of training, testing, evaluating and F1 scoring are based on the models trained with different steganography payload values which starts from easy to hard to detect. We investigated for the best possible ways of increasing the success rate and decreasing the error rate on detecting stego images and cover images separately with this study. Results showed that transfer learning applied model is more successful on detecting stego images on every different rated payload dataset compared to the normal trained model.

Keywords: steganography, image steganalysis, deep learning, convolutional neural networks, residual learning, transfer learning

Önceden Eğitilen Derin Artıklı Sinir Ağları Modeli ile Görüntü Steganalizinde Transfer Öğrenmenin Etkileri

Selim ÖZCAN

Öz

Steganaliz, genelde resimler olmak üzere dijital bir taşıyıcıda gizlenen gömülü mesajları açığa çıkarmak için kullanılan teknikleri araştırır. Son yıllarda görüntü steganalizinde araştırma ve geliştirme çalışmaları daha fazla çekicilik kazanmıştır. Görüntü steganalizi alanında makine öğrenmesi teknikleri uzun yıllardır kullanılmasına rağmen derin öğrenme yeni bir paradigmadır. Derin öğrenme işleminin başarısının temeli modelin yeterli miktarda eğitilmesine ve yüksek kaliteli, çeşitli ve büyük orandaki veri setine bağlıdır. Eğitim işleminde veri setinde kalite, çeşitlilik ve miktar bağlamında eksiklik olması durumunda derin öğrenme metotlarından transfer öğrenme etkili bir çözüm olarak ortaya çıkmaktadır. Transfer öğrenmede eğitilmemiş bir model daha önceden eğitilmiş bir modelden ve onun veri setinden faydalanır. Temel işlev eğitilmiş modelin parametrelerini eğitilmemiş modele transfer etmektir. Böylelikle, görüntü steganalizi için derin öğrenme modelinin başarısı artırılmaktadır. Bu çalışmamızda, transfer öğrenme metodu ile eğitilmiş ve eğitilmemiş iki model serisini karşılaştırıyoruz. Modelin eğitimi işleminde optimizasyon metodu olarak deneysel AdamW optimizasyon metodu seçilmiştir. Eğitim, test, değerlendirme ve F1 skorlama karşılaştırması kolay tespit edilebilenden zor tespit edilebilen farklı steganografi yük değerleri ile eğitilen modeller üzerinden gerçekleştirilmiştir. Bu çalışmamızda ayrı ayrı olarak örtülü-stego görüntülerin ve örtülmemiş-cover görüntülerin tespit edilmesinde hata oranını azaltmanın ve başarı oranını artırmanın olası en iyi yollarını araştırdık. Sonuçlar transfer öğrenme metodunun uygulandığı modelin her farklı oranda yüklenmiş veri setlerindeki örtülü-stego görüntülerin tespit edilmesinde normal eğitilmiş modele göre daha başarılı olduğunu göstermiştir.

Anahtar Sözcükler: steganografi, görüntü steganaliz, derin öğrenme, evrişimsel sinir ağları, artık değerli öğrenme, transfer öğrenme

This thesis is dedicated to my precious loved ones. I know them, I hope they know themselves.

Specially and mostly for my dear one SM, not without you...

Acknowledgments

This study was funded by the Cloud Computing and Big Data Laboratory (B3LAB) of TUBITAK-BILGEM. I would like to thank Cloud Computing and Big Data Laboratory (B3LAB) of TUBITAK-BILGEM for supporting me and allowing me to use their computer facilities for this study.

I thank to my friends in B3LAB for answering my questions and for their time explaining me the details.

This thesis started when my advisor Dr. Ahmet Fatih Mustaçoğlu acknowledged me as his student. I want to express my thanks and gratitude to him. I thank to my advisor Prof. Dr. Ensar Gül for his support, suggestions and counsel.

Of course, I would thank to my family -my dad, my mom and my brother- for their support on my long journey whenever I was in need.

The most crucial person for this thesis and it would be impossible not to mention her here is Merve. It was her endless support day and night, her important and precious ideas and opinions, inspiration, motivation and urge made me to write my thesis and finally graduate. I do thank her.

May 2019

Selim Özcan

Contents

Declaration of Authorship **ii**

Abstract **iv**

Öz **v**

Acknowledgments **vii**

List of Figures **x**

List of Tables **xii**

Abbreviations **xiii**

1 Introduction **1**
- 1.1 Issue of Secrecy 1
- 1.2 Steganography 3
  - 1.2.1 History of Steganography 5
  - 1.2.2 A Very Basic Steganographic Method: Least Significant Bit (LSB) 8
  - 1.2.3 An Least Significant Bit (LSB) Example 9
- 1.3 Steganalysis 10
- 1.4 Deep Learning 11
  - 1.4.1 Convolutional Neural Networks 13
  - 1.4.2 Residual Neural Networks 15
  - 1.4.3 Transfer Learning 16
- 1.5 Contributions 17
- 1.6 Outline 18

2 Related Work **19**

3 Proposed Method: Steganalysis via Transfer Learning **22**
- 3.1 Background 22
- 3.2 Transfer Learning Applied Model 24
- 3.3 Normal Trained Model 25

4 Evaluation **27**
- 4.1 Research Questions 27
- 4.2 Experimental Setup 27
  - 4.2.1 The Dataset 27

4.2.2 Test Environment . . . 37
4.2.3 Discussions . . . 38
4.3 Performance Evaluation . . . 39

5 Results **41**
5.1 HUGO Test Results . . . 42
5.2 WOW Test Results . . . 48
5.3 Result Comparisons . . . 52
5.3.1 Train Comparisons . . . 52
5.3.2 Train Validation Comparisons . . . 54
5.3.3 Evaluation Comparisons . . . 56
5.3.4 Prediction Comparisons . . . 58
5.3.5 Precision Comparisons . . . 60
5.3.6 Recall Comparisons . . . 61
5.3.7 F1-Score Comparisons . . . 65
5.3.8 Related Work Comparisons . . . 66

6 Conclusion **69**

A WOW Training Validation Results **71**

B Source Codes **82**

Bibliography **91**

List of Figures

1.1 Security Domain . . . 3
1.2 Cryptography versus Steganography . . . 4
1.3 Steganographia Cover Page . . . 6

3.1 Model with Transfer Learning . . . 24
3.2 Model without Transfer Learning . . . 26

4.1 Train Dataset . . . 28
4.2 Train Dataset . . . 28
4.3 Train Dataset . . . 29
4.4 Train Dataset . . . 29
4.5 Train Cropped Example . . . 30
4.6 Test Cropped Example . . . 32
4.7 Test Dataset . . . 33
4.8 Test Dataset . . . 33
4.9 Test Dataset . . . 34
4.10 Cover, Secret and Payload Rated Images . . . 35
4.11 Different Payload Rated Images . . . 36
4.12 Test Dataset . . . 37

5.1 HUGO Models - Training Loss Comparison . . . 53
5.2 HUGO Models - Training Accuracy Comparison . . . 53
5.3 WOW Models - Training Loss Comparison . . . 53
5.4 WOW Models - Training Accuracy Comparison . . . 54
5.5 HUGO Models - Train Validation Loss Comparison . . . 54
5.6 HUGO Models - Train Validation Accuracy Comparison . . . 55
5.7 WOW Models - Train Validation Loss Comparison . . . 55
5.8 WOW Models - Train Validation Accuracy Comparison . . . 55
5.9 HUGO Models - Test Evaluation Loss Comparison . . . 56
5.10 HUGO Models - Test Evaluation Accuracy Comparison . . . 57
5.11 WOW Models - Test Evaluation Loss Comparison . . . 58
5.12 WOW Models - Test Evaluation Accuracy Comparison . . . 58
5.13 HUGO Models - Error Rates True Estimation Comparison . . . 59
5.14 HUGO Models - Error Rates False Estimation Comparison . . . 59
5.15 WOW Models - Error Rates True Estimation Comparison . . . 60
5.16 WOW Models - Error Rates False Estimation Comparison . . . 60
5.17 HUGO Models - Test Dataset Precision Comparison of Cover Images . . . 61
5.18 HUGO Models - Test Dataset Precision Comparison of Stego Images . . . 61

5.19 HUGO Models - Test Dataset Average Precision Comparison 62
5.20 WOW Models - Test Dataset Precision Comparison of Cover Images . . . 62
5.21 WOW Models - Test Dataset Precision Comparison of Stego Images . . . 62
5.22 WOW Models - Test Dataset Average Precision Comparison 63
5.23 HUGO Models - Test Dataset Recall Comparison of Cover Images 63
5.24 HUGO Models - Test Dataset Recall Comparison of Stego Images 64
5.25 HUGO Models - Test Dataset Average Recall Comparison 64
5.26 WOW Models - Test Dataset Recall Comparison of Cover Images 64
5.27 WOW Models - Test Dataset Recall Comparison of Stego Images 65
5.28 WOW Models - Test Dataset Average Recall Comparison 65
5.29 HUGO Models - Test Dataset F1 Score Comparison of Cover Images . . . 66
5.30 HUGO Models - Test Dataset F1 Score Comparison of Stego Images . . . 66
5.31 HUGO Models - Test Dataset Average F1 Score Comparison 67
5.32 WOW Models - Test Dataset F1 Score Comparison of Cover Images . . . 67
5.33 WOW Models - Test Dataset F1 Score Comparison of Stego Images 67
5.34 WOW Models - Test Dataset Average F1 Score Comparison 68

A.1 LSB Training Validation 71
A.2 WOW Training Validation Progress - Payload Rate: 1.0 72
A.3 WOW Training Validation Progress - Payload Rate: 0.9 73
A.4 WOW Training Validation Progress - Payload Rate: 0.8 74
A.5 WOW Training Validation Progress - Payload Rate: 0.7 75
A.6 WOW Training Validation Progress - Payload Rate: 0.6 76
A.7 WOW Training Validation Progress - Payload Rate: 0.5 77
A.8 WOW Training Validation Progress - Payload Rate: 0.4 78
A.9 WOW Training Validation Progress - Payload Rate: 0.3 79
A.10 WOW Training Validation Progress - Payload Rate: 0.2 80
A.11 WOW Training Validation Progress - Payload Rate: 0.1 81

List of Tables

5.1 HUGO Algorithm - Transfer Learning Model Precision Scores 44
5.2 HUGO Algorithm - Non-Transfer Learning Model Precision Scores 44
5.3 HUGO Algorithm - Transfer Learning Model Recall Scores 44
5.4 HUGO Algorithm - Non-Transfer Learning Model Recall Scores 45
5.5 HUGO Algorithm - Transfer Learning Model F1 Scores 45
5.6 HUGO Algorithm - Non-Transfer Learning Model F1 Scores 46
5.7 HUGO Algorithm - Overall Performance Comparison of Results 47
5.8 WOW Algorithm - Transfer Learning Model Precision Scores 48
5.9 WOW Algorithm - Non-Transfer Learning Model Precision Scores 49
5.10 WOW Algorithm - Transfer Learning Model Recall Scores 49
5.11 WOW Algorithm - Non-Transfer Learning Model Recall Scores 50
5.12 WOW Algorithm - Transfer Learning Model F1 Scores 50
5.13 WOW Algorithm - Non-Transfer Learning Model F1 Scores 50
5.14 WOW Algorithm - Overall Performance Comparison of Results 51
5.15 Comparisons of Detection Error for WOW Algorithm 68

Abbreviations

Adam	**Ada**ptive **m**oment estimation
AdamW	**Ada**ptive **m**oment estimation with **W**eight decay
bpp	**B**its **P**er **P**ixel
CNN	**C**onvolutional **N**eural **N**etworks
CPU	**C**entral **P**rocessing **U**nit
FNN	**F**ully Connected **N**eural **N**etwork
GPU	**G**raphical **P**rocessing **U**nit
HUGO	**H**ighly **U**ndetectable Ste**GO**
JPEG	**J**oint **P**hotographic **E**xperts **G**roup
J-UNIWARD	**J**PEG **UNI**versal **WA**velet **R**elative **D**istortion
KV	**K**ernel **V**ector
LSB	**L**east **S**ignificant **B**it
Non-TL-Model	**T**ransfer **L**earning Method Not Applied **Model**
RAM	**R**andom **A**ccess **M**emory
ResNet	**Res**idual **Net**work
S-UNIWARD	**S**patial **UNI**versal **WA**velet **R**elative **D**istortion
TL-Model	**T**ransfer **L**earning Method Applied **Model**
WOW	**W**avelet **O**btained **W**eights

Chapter 1

Introduction

1.1 Issue of Secrecy

There have always been a need for secrecy in life from past to present. People desire to prevent unauthorized or unwanted persons from becoming aware of the existence of a message, a happening or a fact etc. Communication between the sides needs to be in secret. In order to maintain secrecy from revealing, many different techniques were developed by the people for ages. Nowadays, it all happens in the digital world. People use digital world almost for everything including communication in secret.

Cryptography is the mostly known and used technique. But, there is also another technique which was unknown and unused unlike cryptography until recent years. The name is steganography as shown at the Figure 1.1[1] under the information hiding section. Cryptography obscures the content of the message so it can not be understood, steganography conceals the existence of the content of the message. Steganographic techniques emerge from ancient history into the digital world [2]. It gains immutable notice and growing popularity into the world of digital communication. The very objective is to prevent the message being read by anyone and also camouflage its presence.

Steganography can be used in many different areas in life depending on the intentions of good or bad. Governments and military could use in times of war for intelligence, businessmen collect information secretly for their new projects in the business world, money transactions, a secret meeting etc. For example, two partners want to communicate for a meeting or share passwords, between a client and a bank for money transactions or PIN numbers of credit cards can be given for the examples for good deeds.

On the other hand, people with evil intentions can use steganography for their interests. Basically and mostly, people use steganography for espionage. It is suitable to use in the corporate espionage without anyone noticing. One could send out valuable information such as trade secrets, business plans, secret chemical formula, blueprints of a new airplane, plans for a new invention etc. Black hat hackers, malware writers, use steganography to smuggle their malicious payloads or malwares into the protected networks passing through security scanners or firewalls, sends commands to malwares from Command and Control (C&C) Servers and after that they can do any bidding they like. Another extremely important and invaluable data are medical records of the patients. Medical diagnosis, history of a patient, X-ray results or scan (CAT, MRI etc.) images are accepted as medical data and must be secured with uncompromising security tools during both storage and transmissions. Even so, data can be stolen, then can be hidden in a file and after that it can be transferred through email. Terrorism is the most dangerous area that steganography could be used. Terrorists can benefit from steganography ensuring secret communications over the internet uncaught and undetected for their terrorists plots.

It is a very hard job to be sure that whether a carrier hides secret messages. There are payload detection tools and methods to help investigators doing the analysis on data. But they are expensive and probabilistic which do not a solution that uncovers the hidden messages. Researchers, governments, corporate businesses, military and intelligence services etc. all try to develop a solution finding the hidden messages, but still not found a certain way. Meanwhile, people develop new harder and smarter steganographic methods from all over the world. In the end, there is always lingers a mystery at the heart of the steganography that is when no secret message was found, then two questions remain: Was there really a secret message or were we unable to find a secret message?

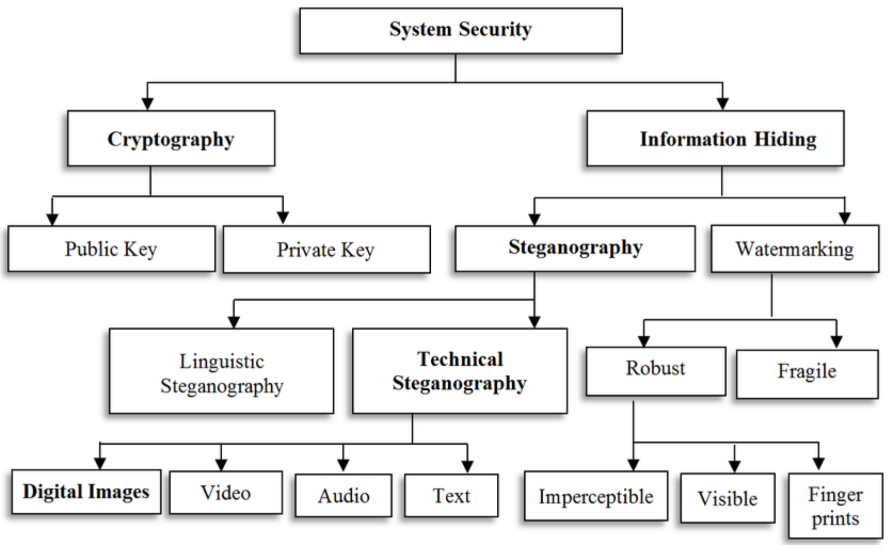


FIGURE 1.1: Security Domain

1.2 Steganography

Steganography is the art and practice of hiding information in a cover medium. Meaning of the steganography is covert text. Steganography makes possible the covert communication. The word of steganography comes from Greek words 'stegos' and 'graphia'. Stegos means hidden or covered. Graphia means to write. Steganography allows embedding secret messages inside a piece of unsuspicious information in another word carriers. In steganography, images, audio, text files, video files, IP datagram and data transmissions are all potential carriers. Then share it to anyone in anytime without anyone knowing the existence of the secret message. The hidden information should not be detected by anyone or anything in any way at any time.

While cryptography focuses on providing privacy, steganography intends providing secrecy. A very basic comparison of mechanism for cryptography and steganography is shown at the Figure 1.2[3]. Cryptography provides visible secret information, but steganography provides invisible and inaudible secret communication. Cryptography is based on robustness of the cryptographic algorithm but steganography is based on undetectability with capacity. Secret messages are scrambled in cryptography in a way that no one other than the intented receiver be able to obtain the secret message. But, on the other hand, secret messages are hidden in steganography so that no existence or knowledge of the secret message perceived by any one. Cryptography and steganography often are used in conjunction with each other providing double protection for secret messages. First, the sender encrypts secret messages with cryptography and then hides the encrypted secret messages in the carriers using steganographic methods. If an enemy or adversary or opponent happens to obtain the carrier, first, the hidden message needs to be found which is a quite hard job, then the hidden message must be decrypted.

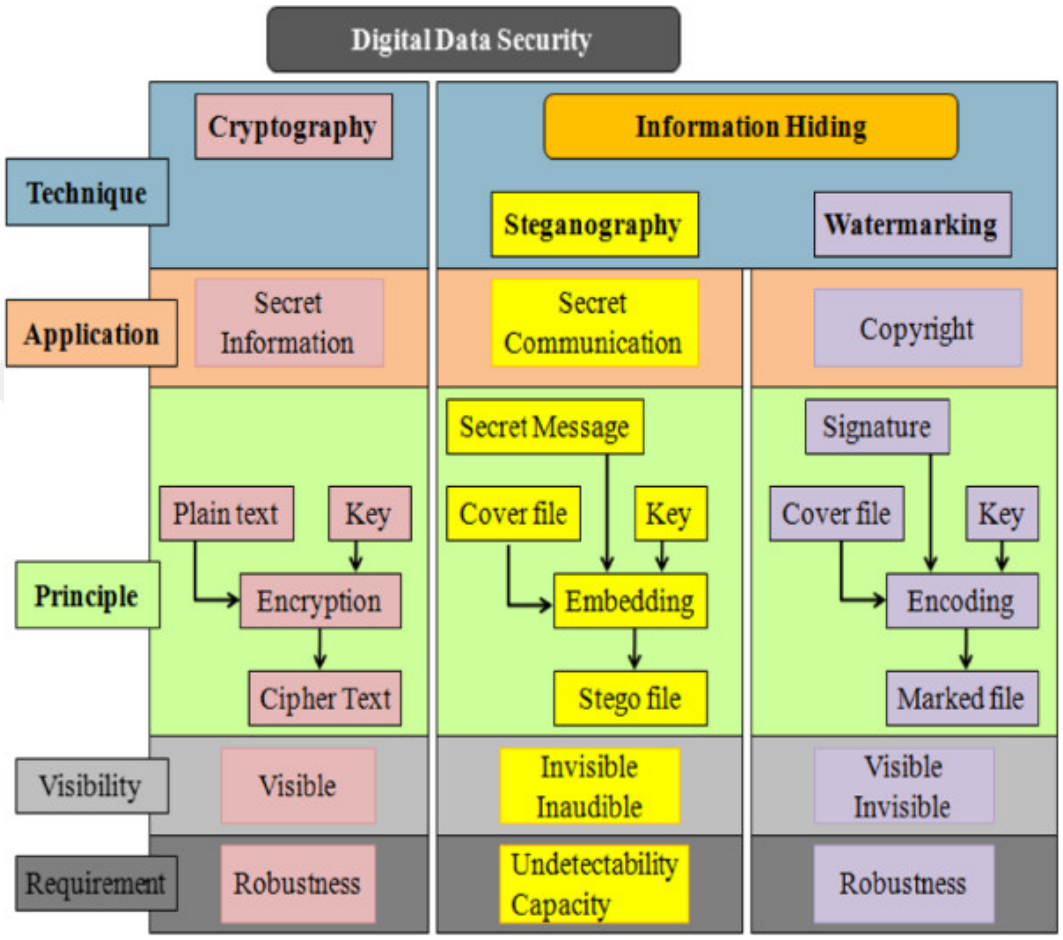


FIGURE 1.2: Digital Data Security

Some common definitions used in the world of the steganography are:

- Cover Medium: Innocent looking object or medium for hiding the Secret Message inside.
- Secret Message: Important data needed to be delivered secretly.
- Key: Can be used to randomize steganographic algorithms or encrypting Secret Message.
- Stego Medium: Innocent looking object or medium loaded imperceptibly with Secret Message.

Equation 1.1 previews the process of the steganography.

$$\begin{aligned} f(Cover\ Medium + Secret\ Message + Key) &= Stego\ Medium \\ ========= Communication\ Channel &========= \\ g(Stego\ Medium + Key) &= Cover\ Medium + Secret\ Message \end{aligned} \tag{1.1}$$

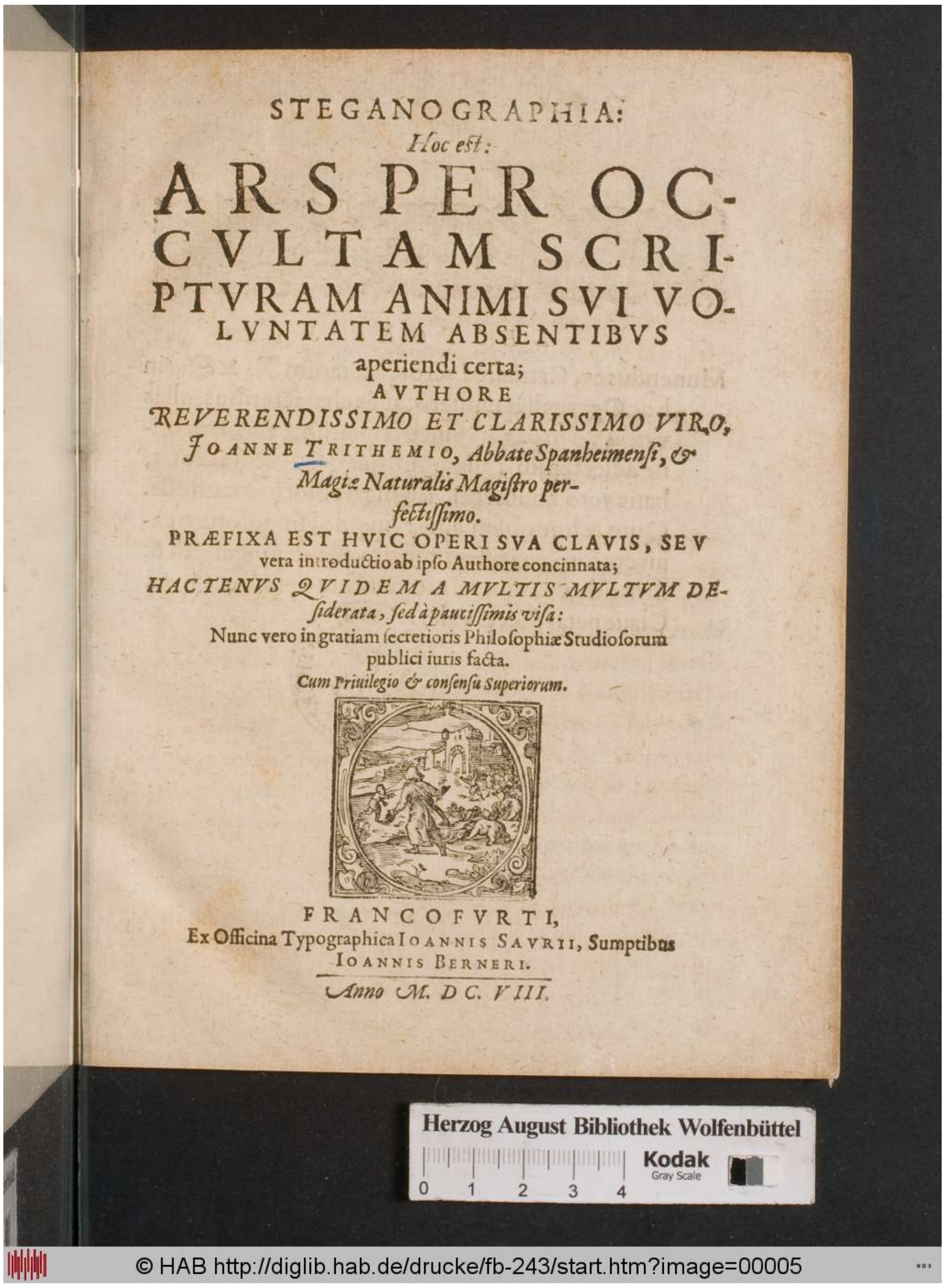
$$f()\ is\ the\ operation\ of\ Embedding\ Steganography\ Algorithm$$
$$g()\ is\ the\ operation\ of\ Decoding\ Steganalysis\ Algorithm$$

1.2.1 History of Steganography

Throughout the history, steganography has been used in various forms for over 2500 years. It has been used and it is still being used in many areas such as in every stages of wars, in political and diplomatic works, espionage, prison breaks, rebellions etc. The word of Steganography first seen in a work named Steganographia written by a German abbot named Trithemius in 15th century. Cover page of Steganographia can be seen on Figure 1.3[4].

The Histories written by Herodotus mentions the first examples of steganography[5]. Herodotus tells that a man named Harpagus killed a hare, hid the secret message inside the hare. Then Harpagus sent the stego hare with someone who looked like a hunter. Harpagus used the hare as a cover to deliver the secret message. Another example is that around 5th century B.C. Histiaeus who was a Greek ruler of Miletus but was captive in the city of Susa which was under control of the Persians. He sent a message to his vassal, name Aristagoras, in the Miletus using steganography. Histiaeus used his most trusted servant. He shaved his head and he wrote the secret message on to the scalp of the servant. Then, once the hair regrown again, Histiaeus sent his stego servant with the secret message to Miletus. Once the stego servant reached the Miletus, Aristagoras shaved the head of the stego servant again and read the secret message that told him to revolt against the Persians. Histiaeus used his servant as a cover to deliver his secret message.

Herodotus also mentions another example about steganography in the ancient Greek still around 5th century B.C. Demeratus an exiled Greek, sent a secret message to warn the people of Greek. He used a wax tablet which was made from wooden backing and covered with wax. At that time, tablets were covered with wax to write messages. Demeratus scraped the wax off of the tablet, carved the secret message on the underlying wood and then covered the tablet with wax again. Stego tablet was seen blank and unused at the probable inspection so it did not suspect anyone. When the intended receiver got the stego tablet, secret message revealed itself after scraping the wax off of the stego tablet.


STEGANOGRAPHIA:
Hoc eſt:
ARS PER OC-
CVLTAM SCRI-
PTVRAM ANIMI SVI VO-
LVNTATEM ABSENTIBVS
aperiendi certa;
AVTHORE
REVERENDISSIMO ET CLARISSIMO VIRO,
JOANNE TRITHEMIO, Abbate Spanheimenſi, &
Magiæ Naturalis Magiſtro per-
fectiſſimo.
PRÆFIXA EST HVIC OPERI SVA CLAVIS, SEV
vera introductio ab ipſo Authore concinnata;
HACTENVS QVIDEM A MVLTIS MVLTVM DE-
ſiderata, ſed à pauciſſimis viſa:
Nunc vero in gratiam ſecretioris Philoſophiæ Studioſorum
publici iuris facta.
Cum Priuilegio & conſenſu Superiorum.

FRANCOFVRTI,
Ex Officina Typographica IOANNIS SAVRII, Sumptibus
IOANNIS BERNERI.
Anno M. DC. VIII.

Figure 1.3: Steganographia Cover Page

As the time was passing by, new steganographic techniques were developed as well. One interesting and popular technique was to use invisible ink to write secret messages on papers such as letters. Milk, urine and fruit juices such as lemon could be the very basic materials to write invisible messages. Text which are written by using invisible ink is not seen with eyes. Therefore, stego letters could be delivered to anyone without any suspect. Only when stego papers heated by a heat source such as a candle, secret messages would get visibly darkened to be read.

With the advancements in the areas of photography, lens making and film technologies, a German invention of the microdot technology was being used in secret communications[6]. Microdots technology reduced the size of the photographs greatly to the size of a dot. Then, reduced sized secret messages were attached onto anything possibly on a letter. Microdots were so small like a dot on the papers that it did not draw attention. Besides being so small, it made it possible of transmission of large amounts of data such as photographs, documents, drawings, pictures and plans.

Null cipher was another technique used in 20th century during World War II. Null ciphering was about hiding secret messages into innocent looking messages or text such as a shopping list or a weather report etc. It followed a pre-arranged set of rules to embed secret messages into writings. Examples[7] for null ciphering could be as follows:

> Fishing freshwater bends and saltwater coasts rewards anyone feeling stressed. Resourceful anglers usually find masterful leapers fun and admit swordfish rank overwhelming any day.
>
> Secret message revealed by taking every third letter in each word: "Send Lawyers, Guns, and Money."

Second example was from a German spy in World War II,

> Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by products, ejecting suets and vegetable oils.
>
> Secret message revealed by taking every second letter in each word: "Pershing sails from NY June 1."

1.2.2 A Very Basic Steganographic Method: Least Significant Bit (LSB)

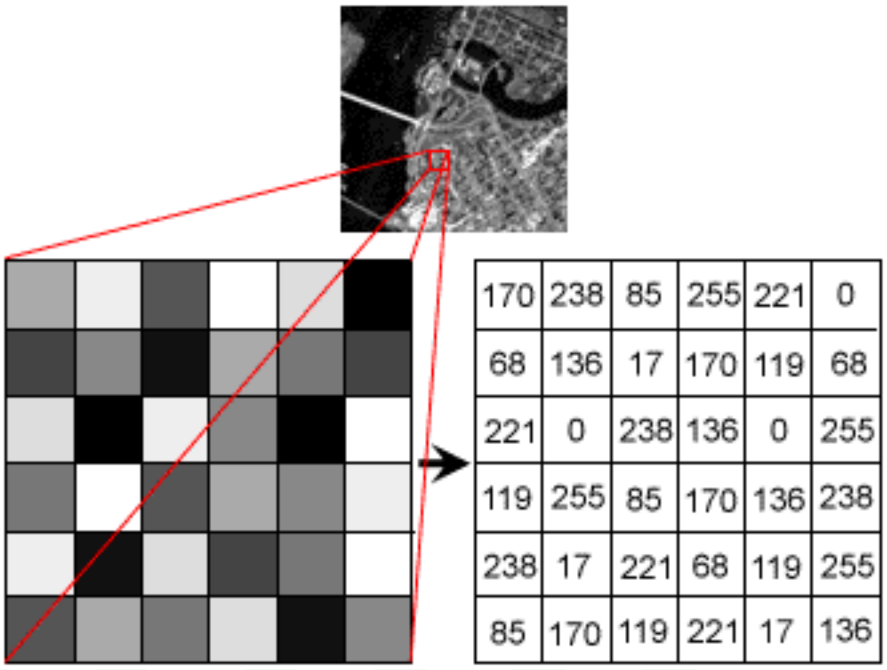
There are many different ways to embed secret messages into ordinary data files. A very basic and one of the most common technique is to hide data such as a text file in bits into the pixels of image files. The operation is proceed by storing the bits of the message into the less significant bytes of the cover-carrier image file. By embedding secret message data to redundant data section of image files in some inconspicuous way, we obtain as a result a stego image file that appears identical to the original image file. Anyone viewing the stego image file should not be able to differentiate the encrypted-stego image file from the original image file. But it has some noise patterns due to applying secret message.

It is the quality of steganographic algorithm that stands against revealing hidden information as long as possible to any kind of attacks. It is much easier to detect existence of hidden information if the irregularity on the stego image is obvious to be seen even with the eyes. Steganography modifies image contents in order to hide information according to algorithm and size of the information which needs to be hidden as in [8]. In the same way, if the size of the information to be hidden is small, then content of the cover image would be corrupted much less and this decreases detection possibility. In the steganography algorithms, information to be hidden is named as payload.

When an analog image is to be converted to a digital format, there are usually three different ways of representing colors:

- 24-bit color: Represented by three different layers of basic colors: Red, Green and Blue. Each has 8 bit(256 values). Every pixel could have a color in the pool of $2\hat{2}4$ colors.
- 8-bit color: Represented by only one layer of colors having 8 bit(256 values).
- 8-bit gray-scale: Represented by only one layer of shades of gray having 8 bit(256 values).Figure 1.4[9] shows an example of 8-bit gray-scale image and pixels.

In 24-bit colorful images, LSB steganography algorithm changes the LSBs of each color of pixels. Similarly, in the 8-bit images, LSB insertion modifies the LSBs of the 8-bit values. Most LSB algorithm based steganography software tools hide information in at least three ways:



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

FIGURE 1.4: Gray Scale Image

- By changing LSB of pixels randomly.
- By modifying LSB of pixels in certain areas of images.
- By incrementing or decrementing the pixel value of the last bit.

1.2.3 An Least Significant Bit (LSB) Example

If the letter 'A' would be the secret message, then it has a value of 65(decimal) in ASCII code, equal to binary value of 1000001. 1-bit LSB modification of pixels usually has a 50% chance to change a LSB every 8 bits. Therefore, very little noise would be added to the original cover image.

LSB process in an 8-bit image would be as follows with 8 pixels:

10000000, 10100100, 10110101, 10110101, 11110011, 10110111, 11100111, 10110011

Then pixel values after the insertion of an 'A':

1000000**1**, 10100100, 1011010**0**, 1011010**0**, 1111001**0**, 1011011**0**, 1110011**0**, 10110011
(The bold values were modified by the LSB algorithm)

Three consecutive pixel values are needed to store an 'A' inside a 24-bit image:

The pixel values before the insertion are:

10000000.10100100.10110101, 10110101.11110011.10110111,
11100111.10110011.00110011

Then pixel values after the insertion of an 'A':

1000000**1**.10100100.1011010**0**, 1011010**0**.1111001**0**.1011011**0**,
1110011**0**.10110011.00110011

1.3 Steganalysis

Steganalysis is the art of uncovering hidden information from a stego medium. There are different methods to be used as steganalysis tools. But, it has more promising results to use the deep neural networks in steganalysis of images[10].

Before deep learning approach, state-of-the-art steganalyzers were based on two steps: Rich Models[11] as feature extraction step and Ensemble Classifier as classification step[12]. Feature extraction needs deep domain knowledge on steganography, steganalysis and structure of images. Instead, deep learning models extract features automatically with semi-supervised or unsupervised methods.

In nature of steganalysis, it is not certain whether a message is hidden or not. Researchers tries many ways to ensure this. However, they may not conclude any result because of two reasons; there is no hidden message to find at all or the proper solution cannot be found.

Steganalysis works scientifically to break steganography, as the name may suggest. People are called steganalyst who work in this field. There are two different types of attacks:

- Detection and/or Extraction(Passive Attack):The intention of the attack is to detect secret message. If secret message exists, then extract it.
- Destroy or Distortion(Active Attack):The intention of the attack is making impossible to obtain the secret message by anyone.

It is possible to apply these two tasks separately. Even if the secret message not known, secret message can be destroyed just by applying some transformations to the stego object depending on the steganographic method used or supposed.

The steganalysis attack methods could be in different forms according to the information steganalyst has:

- A stego only attack: When only have the stego object. Purpose is to detect and/or extract the embedded message.
- A chosen stego attack: When only have both the stego object and the steganography tool or algorithm.
- A known cover attack: When only have both the stego object and the cover object. Comparisons between the two objects could be made.
- A know stego attack: When only have steganography algorithm, stego object and cover object.
- A known message attack: When only have both the secret message and the stego object. Purpose is to find the algorithm of steganography.
- A chosen message attack: Generating a stego object from a message using an algorithm. Looking for signatures to detect other stego object.

1.4 Deep Learning

Deep learning is a sub-part class of machine learning algorithms. Machine learning covers deep learning. Features are given to machine learning manually unlike deep learning that learns features directly from data. When the amount of data increases, deep learning gives better performance than machine learning techniques. Deep learning finds usage in the areas of the following: speech recognition, image classification, natural language processing, recommendation systems etc. It is a computer software and mathematical equations that mimics the brain by modeling the network of neurons. It is called deep learning because it uses deep neural networks. Deep learning models are constructed by connecting layers. Layers are named according to their position in the model. First layer is called Input Layer. Last layer is called Output Layer and all the other internal layers between them are called Hidden Layers. The word deep means that the connected layers in the model are more than two layers. Figure 1.5[13] shows a basic structure of deep learning model.

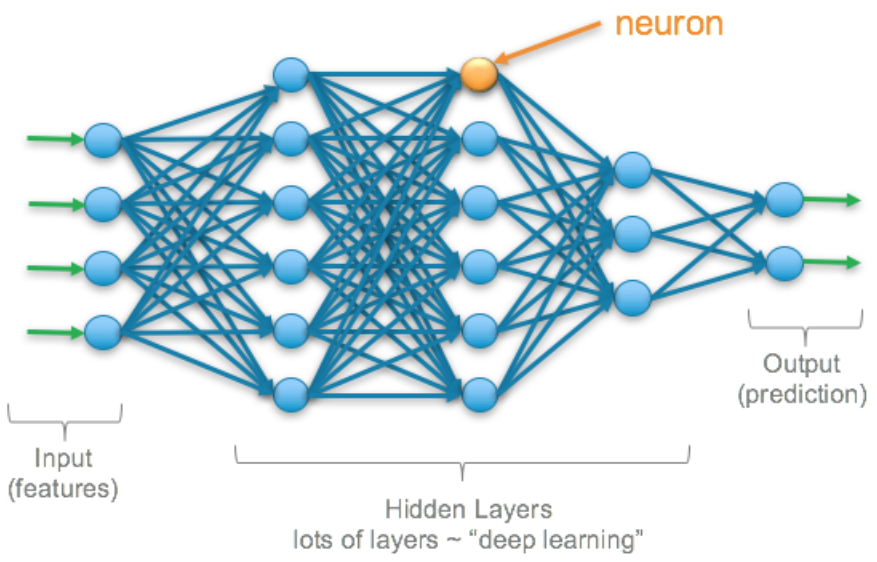


FIGURE 1.5: Deep Learning Model

Layers are composed of neurons which connected to each others. Neurons process the incoming input signal and then propagate the output signal towards to the output layers. The strength of the signal depends on the weight, bias and activation function of the neurons. The network could learn complex features of the data at each layer.

A neuron is a mathematical function as shown at the Figure 1.6[14]. Neuron takes one or more inputs, multiplies inputs by values called weights then sum all of them. At the end, a non-linear function called activation function takes the sum value and produces the output value of the neuron. The **x** values are inputs and the **w** values are weights that are coefficients. Symbol showed with b is Bias intercept value which help to fit data better.

The calculation of a neuron starts with a lineer equation as shown in Equation 1.2. f function is the activation function. An activation function is an fundamental component of neural network architectures. It provides non-linearity to the output value of the neuron. y is the output value of the neuron.

$$y = f(x_1 w_1 + x_2 w_2 + x_3 w_3 ... + b) \tag{1.2}$$

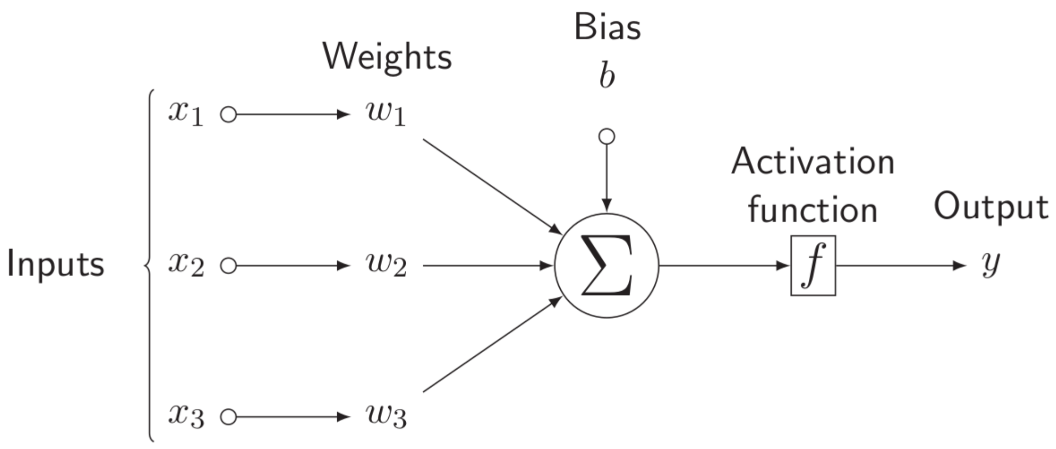


Figure 1.6: Neuron of Deep Learning Model

Sigmoid activation function as in Equation 1.3 results in a value between 0 and 1. It tells how confident the model is that the processing material belongs to a class. Sigmoidal activation function is used especially for models to predict the probability. It is good to use for a classifiers.

$$f(x) = \frac{1}{1 + exp(-x)} \tag{1.3}$$

1.4.1 Convolutional Neural Networks

Convolutional Neural Networks(CNNs or ConvNets) are very much capable of being used in many image related machine learning projects. CNNs achieve very impressive results in recognition of images in the area of computer vision. Due to the structure of CNNs, neural network models are able to classify images. CNNs contain a concrete case of Deep Learning neural networks constructed with neurons, weights and biases.

ConvNets work just like how people recognize things. While seeing, recognition of an object begins with the recognition the parts of the object. For example, identifying a car starts with identifying the parts of the car such as four tires, four car doors, windows, size and measurements etc. After a car is learned, even if two tires out of four tires are not seen on the image, then it is still classified with a high probability as a car. Because the other parts of the car could still be seen on the image. But previously, it should be learned what is a tire, a door or a window of the car in the image. That is, firstly, features should be learned by the model to identify lines, edges, shapes or textures similarly belonging to a car. This kind of responsibility is entrusted to ConvNets.

Computers operate on an input image as an array of pixels. Based on the size of the image, the size of the array would result to as in Equation 1.4. CNN models follow the flow of processing as indicated at Figure 1.7[15] which shows how do the convolutional neural networks operate on images step by step. It consists of two steps, feature learning and classification. Each input image passes through a series of convolution layers by applying filters(kernels) followed by pooling operation. After that, classification starts with flatten and fully connected layers. Softmax activation function is applied to classify objects in the image with probability between 0 and 1. This is called forward propagation.

$$height * width * dimension \tag{1.4}$$

- Convolution Layers extract feature informations from input images. Model does not know where are the features in the image. So, it tries to find features by applying a filter to the input matrix of images. Applying a filter is a mathematical operation that sums the outputs of the two matrices multiplication.
- Pooling layers reduce the number of parameters thus subsamples or downsamples the dimension of the image without losing important feature information.
- Fully Connected layers take input as a flattened vector. Then, FC layers use features to classify the input images just like neural networks.

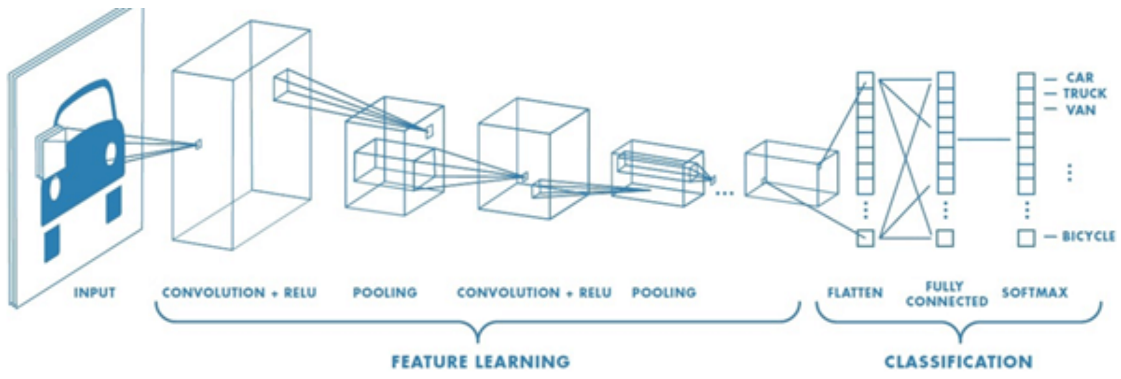


FIGURE 1.7: Convolutional Neural Network Model

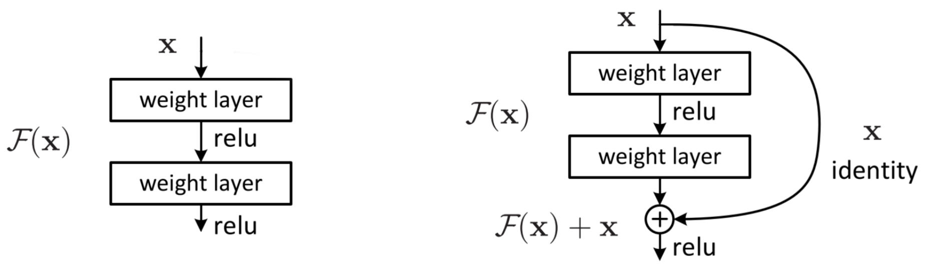
Neural Network Models would result more accurate predictions if they have multi-layers architecture named as deeper neural network models. The deeper the layer of the network, the more training time, more dataset to train the model required. As a side effect, if depth of model' layers continue to increase, the accuracy of the model might start to degrade rapidly due to vanishing gradients. Gradients of model result with small values approaching to zero which makes the model hard to train. Error parameters of the model would be difficult to propagate back correctly. The model starts to learn irrelevant information on deep layers.

In backward propagation, derivatives of the output are find going backward in the model. Backpropagation is used in the model to calculate the error rate and loss value of the model. After a forward propagation of the model is done and a result is obtained, it is controlled with the truth table to check if the model produced the expected result. If not, then some parameters of the model need to be increased or decreased by the amount of error rate and loss value. In order to calculate gradients such as error rate and loss value, model backpropagates backward from final layer to initial layer taking derivatives. By taking backpropagation to first layers, hence more parameters are multiplied which made the gradients smaller. In the end with small gradients, weights and biases would not get updated effectively in the initial layers for each training sessions. Therefore, the overall accuracy of the network would not get effectively updated too.

Residual Networks is one solution of many others.

1.4.2 Residual Neural Networks

In the light of deeper models, Residual Networks-ResNets[16][17] have an architecture model that is deeper than normal models but eases the training time and prevents the vanishing gradients problem of accuracy of the model. Before ResNets, deeper neural networks were having much more training errors. Data was disappearing going through too many deep layers. Figure 1.8[18] shows the differs between a plain network model and a residual block. Basically, ResNets have residual building blocks which provide shortcuts between layers while carrying inputs of the lower layers to deeper layers of the network. It splits deep neural networks to chunks with three layers. With an architecture of residual blocks, the model does not lose the input from a previous layer. It enables of passing the inputs straight through to next chunks. Residual connections prevent the derivatives to decrease to the smaller values by not letting them through activation functions. Residual connections add the value $\mathbf{x}$ to the value at the further blocks $\mathbf{F(x)} + \mathbf{x}$.



x
weight layer
relu
weight layer
relu
$\mathcal{F}(\mathbf{x})$
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

FIGURE 1.8: Plain Network and Residual Block

1.4.3 Transfer Learning

Transfer Learning[19] allows quick development of deep learning models. Deep learning models start with a trained background before. They may have optimized weight values that are trained for a similar dataset. Thus, it would be easier and fast to learn on the new datasets. With transfer learning, models have already learned the fundamentals of the dataset, hence it does not require to learn them again on the training phase. It only focuses on the specific requirements that are modeled to learn.

It is useful to use transfer learning when there are insufficient data for the new domain. A pre-trained model is a saved model which trained before with optimized parameters. Saved model could be an effective model to serve for a specific problem. For example, a saved model which trained on recognizing birds could be re-used again to recognize specific birds such as pigeons or eagles. Previous model trained with birds knows how to find a bird in images. But we need a model that could find eagles in images. It takes time and dataset to train a new model with images of eagles to teach features of birds to the model from zero. It is useful using a saved model which trained before on similar domain. Also, applying transfer learning results to train model with less data but increased accuracy in a short time.

When the datasets are similar and relevant to each others, but new/target dataset is small, then transfer learning could be applied as follows:

- Remove the output fully connected layers of the pre-trained base model. Because FC layers are classification layers.
- Add new FC layers to the output layers of the model according to the purpose of the model such as classification.
- Randomize weights of the new connected FC layers.
- Train the network including FC layers with the new small dataset.

By applying this procedure, parameters and weights outside of the removed FC layers are transferred to the new problem. Transferred weights and parameters are optimized on features of images. Thus, transfer learning has been applied.

1.5 Contributions

The problem in the world of steganography is to find a way to uncover the secret messages. There is not a single way to uncover all the steganographic algorithms. Every day new methods are developed in the world of the steganography. Not only steganography is used for good intentions by good people, but there are many people with evil intentions who are capable of taking action with motivations of evil, unfair, unjust and wrong. Steganography is different from cryptography. In cryptography, it is known that there is a secret message, but in steganography, one can not be sure whether there is a secret message or not. This study tries to be a step closer to uncover the secret messages.

In this study, we used ResNet50 model of Keras Deep Learning Framework which was trained on ImageNet dataset. We aim at investigating for a new approach to increase the success rate of detecting very difficult stego images and cover images separately. Thus, we chose transfer learning method in our study.

We trained the model with dataset from easy to tell apart stego and cover images until hard to distinguish stego and cover images. Therefore, the model has learned the smallest changes on images between different dataset in every step. Thus, we obtained successful detection results even on the 0.1bpp embedded stego images which have too small stego corruption conjugate cover images.

1.6 Outline

First chapter includes the introductions to the domain, methodology, problem and what the contributions are.

The remaining of the thesis was organized as follows:

Chapter 2 reports the preceding studies and discussions of how different our methodology was.

Chapter 3 explains the steps of the study, methodology, background information, the details about the deep learning models.

Chapter 4 is about the experimental setups of the study, dataset, training, testing and how the results were scored and evaluated.

Chapter 5 presents the results of the study. It shows comparisons supported with Figures, Tables, evaluation and scoring results.

Chapter 6 indicates the conclusion from the study. Also, plans of the future studies were included at the end.

Appendix A contains the obtained results from validation step of the training process of the models with WOW steganography algorithm.

Appendix B gives a sample Python source code that run for training the model with the LSB steganography algorithm.

Bibliography section of references.

Chapter 2

Related Work

There have been many methods in steganalysis to uncover the embedded messages. With the help of convolutional neural networks, image steganalysis got popular in the direction to neural networks. There has not yet found an effective solution to easily extract secret messages from images.

In [20], as activation function Gaussian was used in CNN model and only three embedded datasets are processed such as 0.5, 0.4 and 0.3 from database of BOSSBase. This study did not consider working on the 0.1 embedded payload dataset. Changing activation function may affect the success rate for a small percentage. Another study [21] is a similar to the first one[20]. In this study[21], a preprocessing operation was applied to the images. Then, the preprocessed images were given to a CNN model which was not as deep as a Deep Residual Neural Network. Although the study was based on WOW and HUGO steganography algorithms, evaluations were not performed using the lower embedded payload values.

In [22], the CNN model was similar to the [20], but they implemented using different filters and algorithms. Nevertheless, this study performed tests on the 0.1bpp payload dataset which is similar to our study. Studies [23], [24] and [25] focused on using CNN in steganalysis domain. They aim to outperform the steganography algorithms by changing a small number of parameters.

Researchers in the [26] mostly focused on preserving and strengthening the weak stego signal in images via different high pass filtering masks. However, their model was based on the residual neural network. Another steganography study [27] used J-UNIWARD algorithm. But, they did not utilize a deep neural network.

Most of the studies were inspired by the [20] and [10] which focused on the convolutional neural networks. In [28], a statistical model was used on the first layer, the study focused on image dataset, not on the model.

Besides, the study [10] which had a similar aim with our study used the transfer learning method to increase the success rate of model. The study differs from our study in terms of the dataset and the design of the model. Their model was trained only with a limited dataset which contains around 40000 images. Our dataset contains almost 160000 images. Their model had one image processing layer, five convolutional layers and three fully connected layers, thus totaling to a nine layered network model. However, we used a deep residual neural network in our study reaching total 177 layers including activation, zero padding, flatten, batch normalization, input and convolution dense layers. Deeper layers may cause vanishing gradient problem to models. It can be said that our model may suffer from vanishing gradient problem too. Since our model is a deep residual neural network, it has residual blocks with residual connections. Hence, we mitigated the vanishing gradient problem for our model.

Another study [29] which mostly focused on JPEG Steganalysis and used a Dense Network performed evaluations on a little large dataset than other studies.

In the [30], researchers mostly focused on the normalization layer of the CNN.

In one [31] of the base studies, researchers tested more than one CNN model by increasing the layers of the model with the aim of finding the optimal number of layers for maximum success rate. The study compared two neural networks, one was a CNN and the other one was a Fully Connected Neural Network (FNN). Besides, this study pointed out this steganalysis scenario which assumed that 'Same embedding Key' was reused for embedding operations of images. This model was not deep enough, and their embedded dataset had only one payload value, which was 0.4 bits per pixel on one steganography algorithm, S-UNIWARD.

One of the study that used a Deep Residual Neural Networks [32] is similar to [33]. They had a stage for preprocessing images before training the model like our proposed approach. Their model had less layers than our model. Our ResNet was also pre-trained with ImageNet dataset. We have tested using more test images in our work. Their dataset had only one embedded payload value of 0.4 bits per pixel as we have run tests with more different embedded dataset.

In our study, as different to other studies, we used Transfer Learning method with a based model Keras ResNet 50 trained model. Most of the other studies were based on the stego images at the average difficulty. Moreover, we increased the success rate of detecting the very hard stego images. A more general approach was used by the previous studies which were trained the model only one kind of steganographic payload. On the contrary, we focused on the training our model step by step with increasing difficulty on different levels of steganographic images. Furthermore, we used a newly developed optimization algorithm to train our model, in order to obtain a better model. Besides, we evaluated the success of our model using Precision, Recall and F1-scoring measurements.

Chapter 3

Proposed Method: Steganalysis via Transfer Learning

3.1 Background

We proposed a steganalysis method that took a previously trained model and adapted its parameters for detecting even the very difficult stego images. We prefered the Python programming language and Keras Deep Learning Framework to implement our proposed method. Base model for training was based on Keras ResNet50 which had pre-trained with ImageNet dataset before. The model was a Residual Network which consisted of much more deep layers than normal neural networks.

Keras Framework's pre-trained ResNet50 model has been optimized for input shape 256x256 image size for our dataset. A fully connected layer followed by a single output layer has been added with a sigmoid activation function and glorot normal initializers for kernel and bias values. Model has been compiled with binary cross entropy loss function. Moreover, during the training process, learning rate has been reduced progressively to a minimum value of '0.00001' if the validation loss value would not decreased for two epochs of training repeatedly.

While training the model with dataset for many epochs, the model would have gotten overfitted that means the model did not learn outside of the images in the training dataset. One solution would be applying L2 regularization to reduce overfitting of the model. L2 regularization means sum of the squared weights. There are more than one method developed for L2 regularization. One of them is applying weight decay to weights of the model while model updates parameters while in the training section. Learning in a model means optimization of parameters. Parameters mostly consist of weights which are the coefficients to multiply with input values. If a model predicts far from the true label, then parameters are need to be updated-increased or decreased- in order to find the best values to predict correctly. Weights are updated by a parameter called learning rate which is a standart value assigned at the start of the training session. At each epoch, model gets closer step by step to the intended optimized weights by changing weights with learning rate parameter. If learning rate would have assigned a big value, then model could learn and increase accuracy but it does not reach to the best accuracy level. Each weights are updated with a new value by the following equation in Equation 3.1. Eta is the learning rate, E is the error function which wanted to be minimized.

$$w_i \leftarrow w_i - \eta \frac{\delta E}{\delta w_i} \tag{3.1}$$

After weight decay of L2 regularization applied, new weights were calculated by the following equation at Equation 3.2. The new term -$\eta \lambda w_i$ from regularization caused weights to decay in proportion to its size.

$$w_i \leftarrow w_i - \eta \frac{\delta E}{\delta w_i} - \eta \lambda w_i \tag{3.2}$$

AdamW was used as the model's optimizer algorithm for training the model. AdamW optimization method was still an experimental algorithm on L2 regularization and weight decay regularization of models. It was proposed in the AdamW that some simple modifications to processes of L2 regularization and weight decay regularization for the commonly used Adam optimization algorithm. According to the results of the paper, it basically proposed two adjustments: First, AdamW has improved Adam's generalization performance for image classification dataset. Second, it gave an optimal choice of weight decay factor unrelated to learning rate setting.

Tests have been run on two similar steganography branches starting from 1.0bpp payload rate to 0.1bpp payload rate for two advanced, state-of-the-art, content adaptive, spatial-domain and highly resistant to known steganalysis techniques steganography algorithms, HUGO (Highly Undetectable steGO) and WOW (Wavelet Obtained Weights). They embed bits of secret messages into cover images by detecting highly textured areas.

3.2 Transfer Learning Applied Model

On the first test branch, transfer learning method has been applied to model on different payload rates. First step was that the model has been trained on the dataset that applied with Least Significant Bit (LSB) steganography algorithm, the very base and easy to detect one. The state-of-the-art steganography algorithms hide information using the LSB technique but with more difficult to detect. As a result, we thought that model would learn how the LSB works. Also, it would adapt to dataset more quickly and to the base operations of steganography.

As Figure 3.1 shows, after LSB training step, the same model was transferred to 1.0bpp steganography dataset for training.

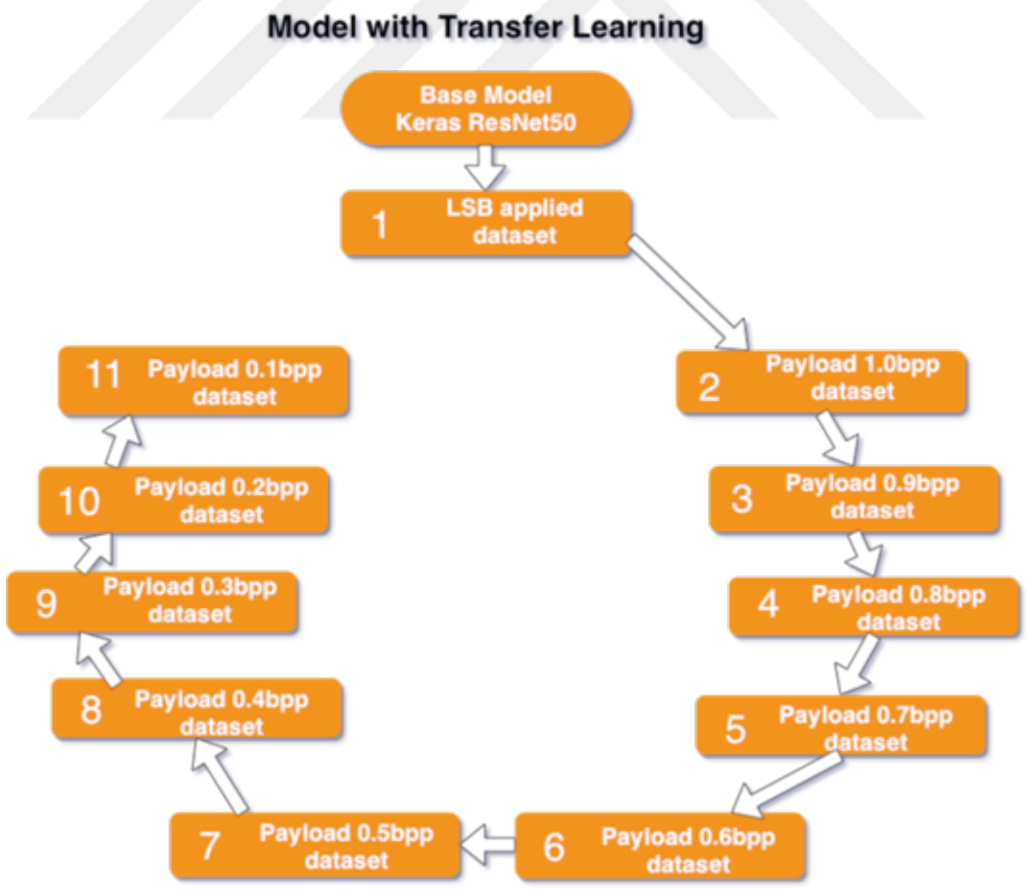


FIGURE 3.1: Flow schema of transfer learning applied model

After 1.0bpp training session, same model with newly gained weight parameters was transferred to 0.9bpp payload rated dataset for training. Training sessions were continued as chain learning sessions decreasing payload values to the last payload rate of 0.1bpp dataset.

We aimed via applying transfer learning, the model would achieve more successful predictions and less error rates on the dataset hardest to detect which is 0.1bpp payload rated. On every step from LSB and 1.0bpp dataset to 0.1bpp dataset, model would learn better how does the steganography algorithm operate and hide messages in the cover images. We thought that model would start gaining the ability to distinguish even the smallest bit changes out of chain learning steps. In the end, the model would be able to detect all the stego images applied in every different embedding payload values using the same steganography algorithm.

3.3 Normal Trained Model

We wanted to compare the effects of that how applying transfer learning changes the error rates of the model on the different payload rated images. In order to compare the results of the transfer learning applied model, we trained a new another model. At the second test model, all model training sessions have been done separately based on the model of Keras ResNet50 with pre-trained weights as in Figure 3.2.

Base Keras ResNet50 model was optimized for our study by changing input shape according to size of images in our dataset and by appending an output layer appropriate for binary classification. We used the base model on all the payload rated dataset separately without pre-training on LSB steganography algorithm.

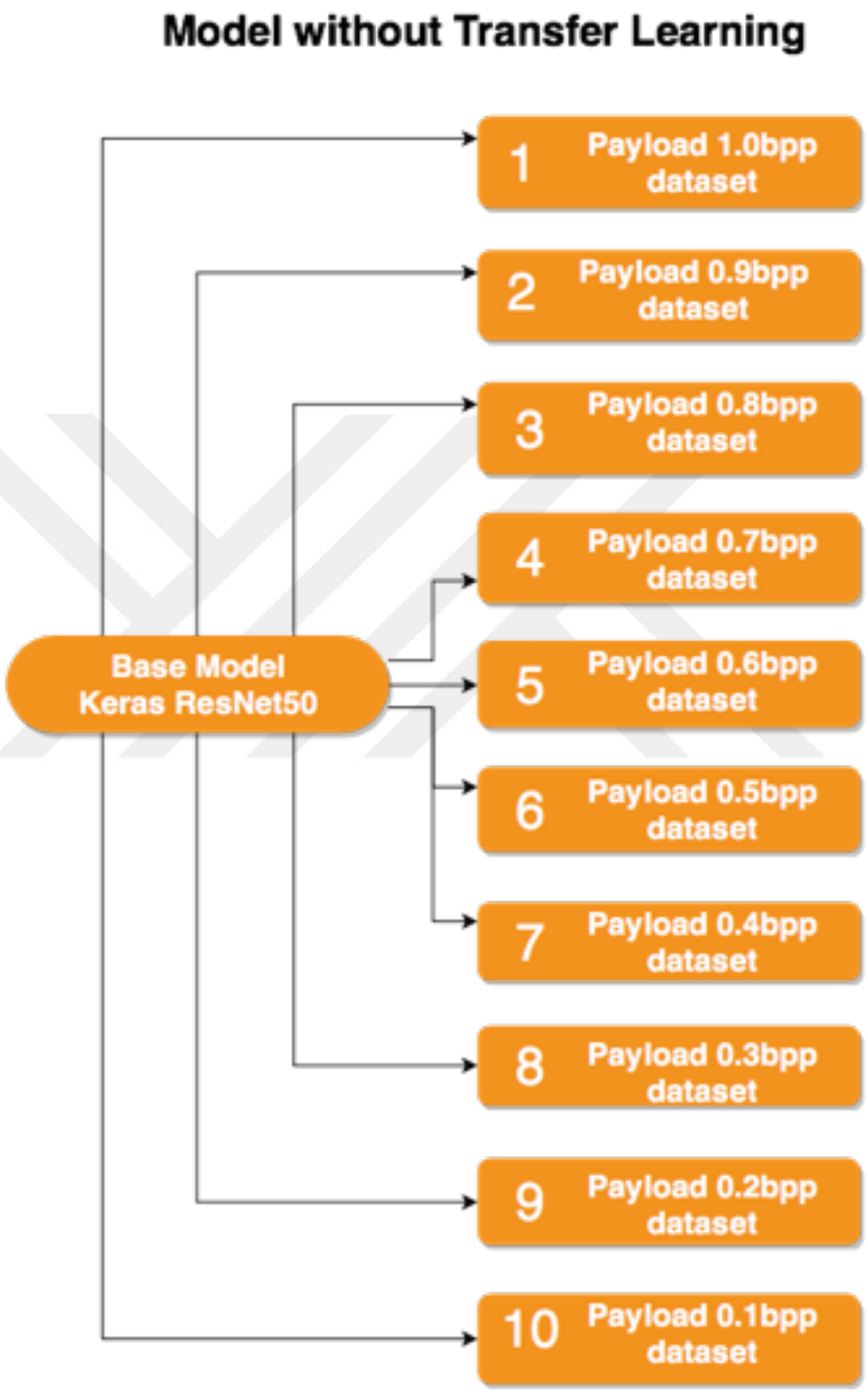


FIGURE 3.2: Non-Transfer Learning applied models

Chapter 4

Evaluation

4.1 Research Questions

1. How does applying Transfer Learning method to a model affect the error rate on the very difficult dataset?
2. Is it possible to have better detection results when transfer learning have been applied to every payload rates between 1.0bpp and 0.1bpp dataset?
3. How does applying or not applying transfer learning on LSB dataset affect the success?

4.2 Experimental Setup

4.2.1 The Dataset

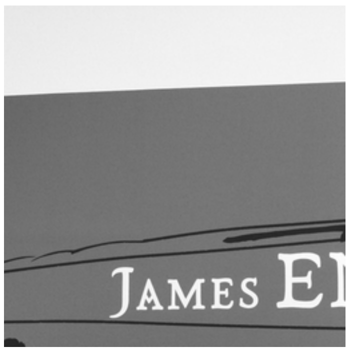
Two different datasets have been processed in our study. One dataset was used for training and the other dataset was utilized for testing, evaluating and scoring the detection rates of the model. Training dataset came from combining BossBase_v1.01 dataset with 10000 units, 512x512 gray-scale image dataset and BossBase_v0.92 dataset with 9074 units, 512x512 gray-scale image dataset. Some images from train dataset were shown at Figure 4.1, Figure 4.2, Figure 4.3 and Figure 4.4.

FIGURE 4.1: Train Dataset Example 1

FIGURE 4.2: Train Dataset Example 2

FIGURE 4.3: Train Dataset Example 3

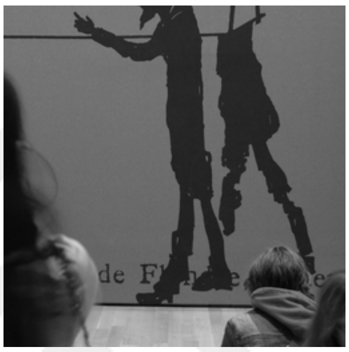
FIGURE 4.4: Train Dataset Example 4



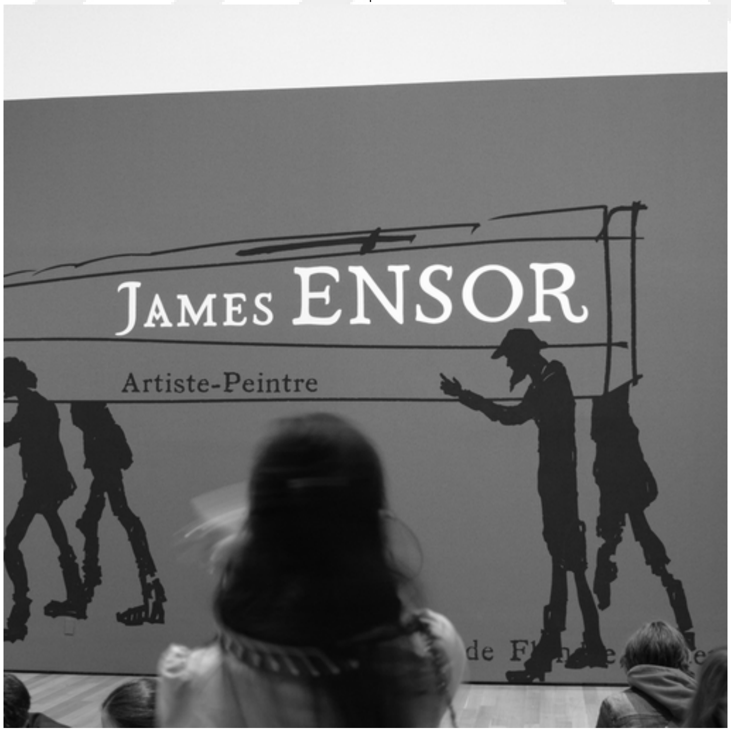
a) Cropped Image Part 1

b) Cropped Image Part 2

c) Cropped Image Part 3



d) Cropped Image Part 4



e) Un-Cropped Image

FIGURE 4.5: Train Cropped Example

Figure 4.5 shows parts of a cropped image from training dataset. The number of base image dataset for training was 76296-unit images. Training the model requireed both cover and stego images, so total training dataset contained 152592-unit images. Validation dataset contained 3815-unit images which were 0.025% part of the total number of the images in the training dataset.

Tests were performed on "BOWS2OrigEp3.tgz"[34] dataset with 10000 units, 512x512 grayscale image dataset. Some images from test dataset were shown at Figure 4.7, Figure 4.8 and Figure 4.9. Training dataset and test dataset were different from each other. The models failed to learn with the current size of the images. All images, training and testing were cropped into four with the size of 256x256.

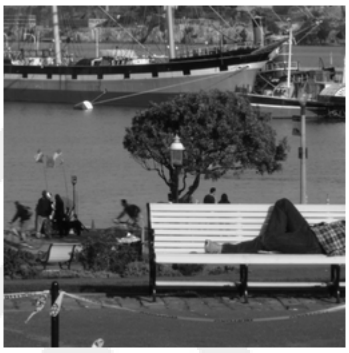
Test dataset contained 40000 units of cropped images for cover images and for stego images, totaling to 80000 units of test images. Test dataset was different from training dataset. Figure 4.6 shows parts of a cropped image from test dataset.

The very base steganography algorithm Least Significant Bit (LSB) was applied to dataset of training and testing in order to be used for transfer learning method. We applied a preprocessing operation on all cover and stego images for the purpose to extract the very weak stego noise traces in images. We convoluted images with a High Pass Filter of size 5x5 kernel matrix KV setting as in the Qian's paper [20]. Convolution with High Pass Filter suppress image content in a big scale. Thus, the weak stego signal in image could become more strengthened to detect. The image content would be less effective for the model discriminating steganographic modifications.

a) Cropped Image Part 1

b) Cropped Image Part 2

c) Cropped Image Part 3


d) Cropped Image Part 4

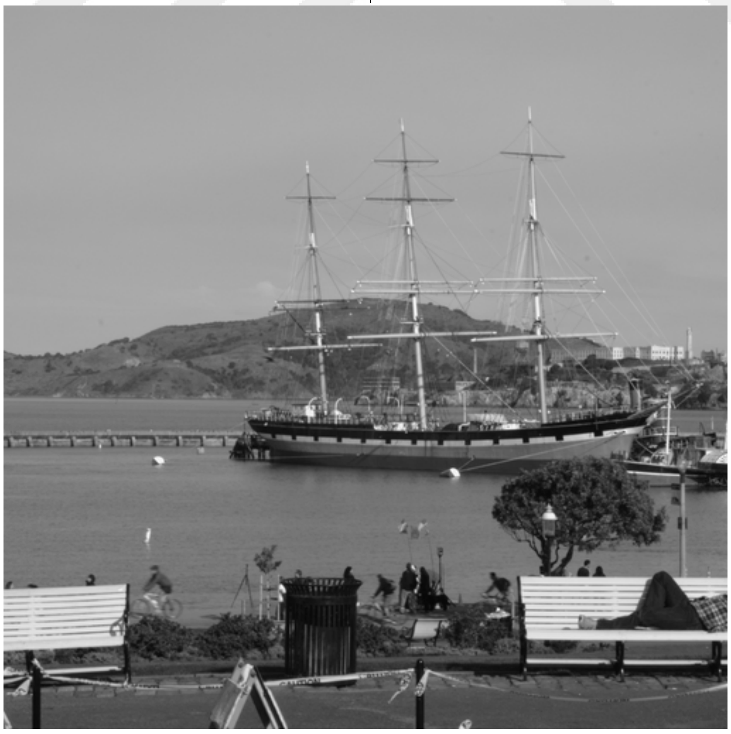

e) Un-Cropped Image

FIGURE 4.6: Test Cropped Example

FIGURE 4.7: Test Dataset Example 1

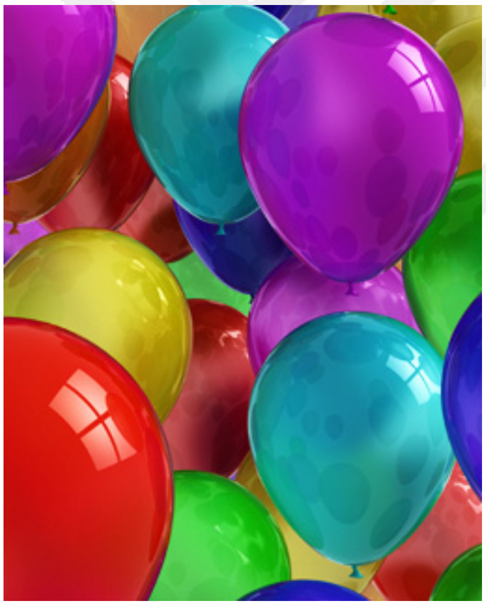
FIGURE 4.8: Test Dataset Example 2

Figure 4.9: Test Dataset Example 3

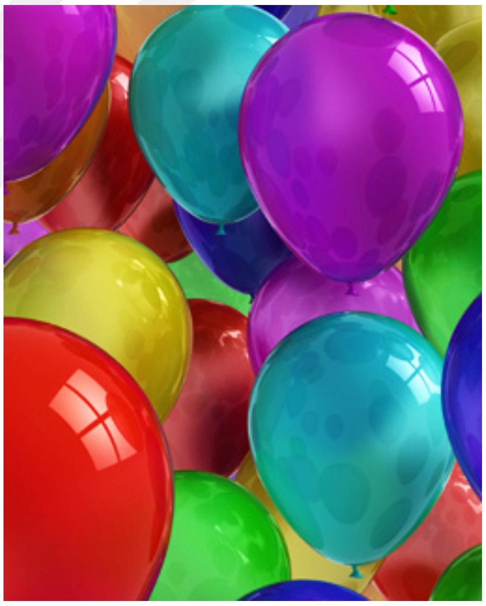
Two steganography algorithms were used on training and testing dataset to create stego images. Stego image dataset was created using HUGO and WOW state-of-the-art steganography algorithms using 'Aletheia' open source image steganalysis tools available from GitHub[35]. It also had steganography embedding algorithms. HUGO and WOW steganography algorithms have been used to create the 10 different payload rated images. Variety of different payload values were as follows 1.0bpp, 0.9bpp, 0.8bpp, 0.7bpp, 0.6bpp, 0.5bpp, 0.4bpp, 0.3bpp, 0.2bpp and 0.1bpp payload rate. Figure 4.10 gives an example on payload rates. Figure 4.10-b is embedded to Figure 4.10-a with different payload rates. Payload rate of 1.0bpp stego images in Figure 4.12 have more corruption than payload rate of 0.1bpp stego images. Therefore, it was easier to detect 1.0bpp stego images than 0.1bpp stego images for a trained model. As payload rates increased in images as seen at Figure 4.11, corruption rate increased as well. The corruption in the Figure 4.12 was so much that the secret image could be seen with eyes only. Lower payload rates caused small changes in the images, thus secret messages were harder to detect in them.

a) Cover Image

b) Secret Image

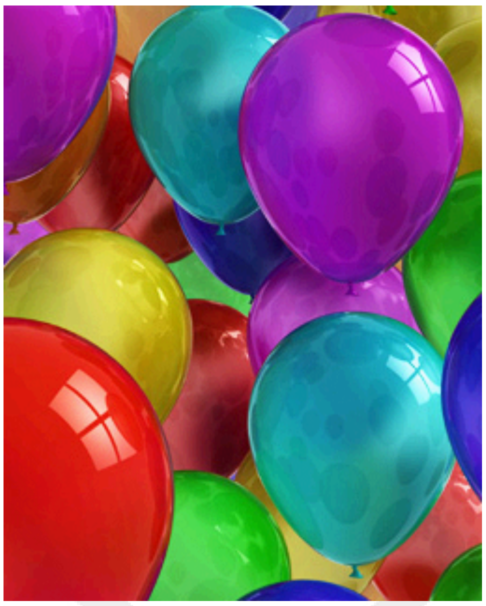


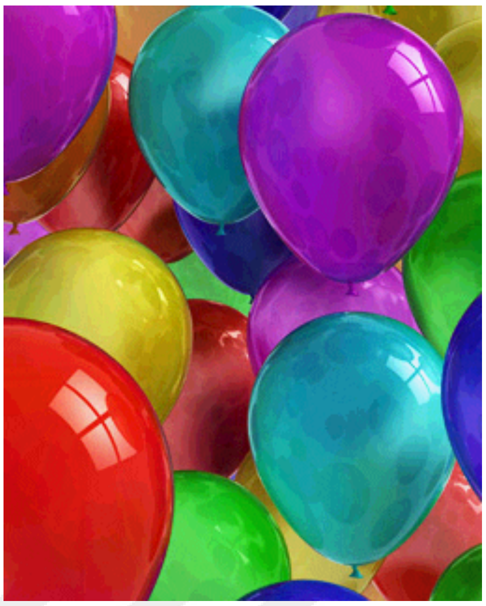
c) Payload Rate 0.1bpp

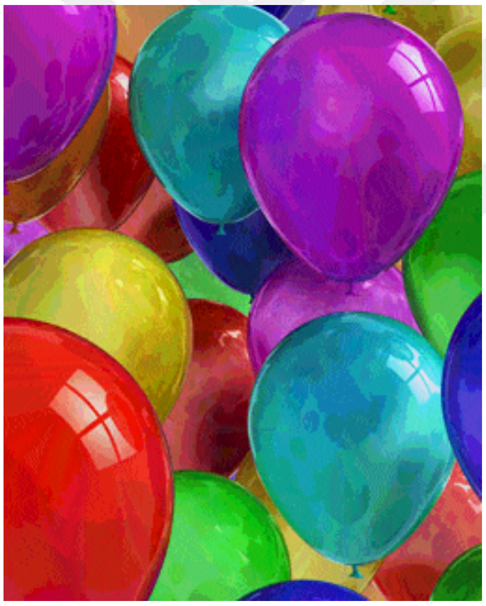


d) Payload Rate 0.2bpp

FIGURE 4.10: Cover, Secret and Payload Rated Images


a) Payload Rate 0.3bpp


b) Payload Rate 0.4bpp


c) Payload Rate 0.5bpp

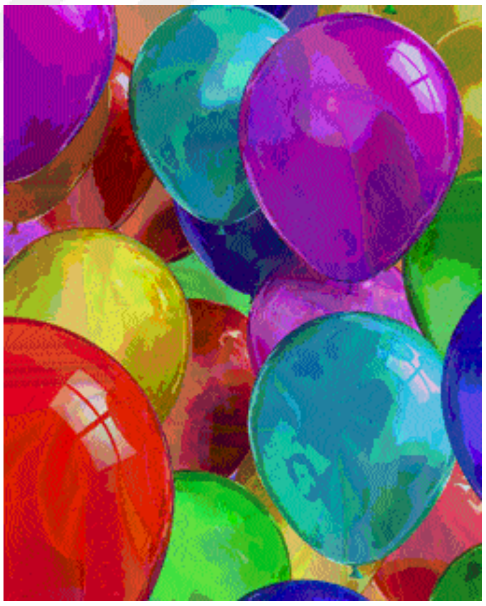

d) Payload Rate 0.7bpp

Figure 4.11: Different Payload Rated Images

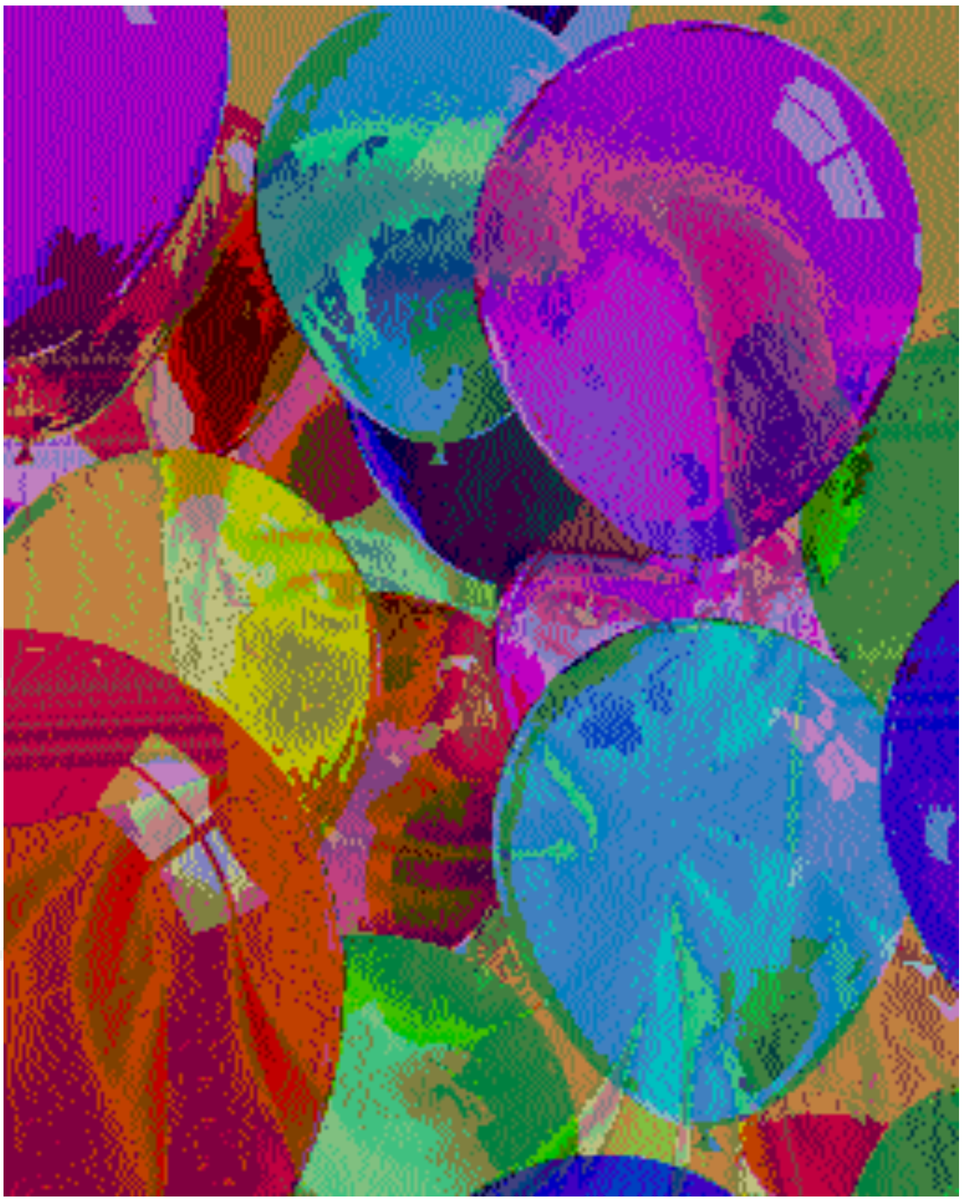


Figure 4.12: Payload Rate 1.0bpp

4.2.2 Test Environment

Tests were run on a server which was provided by B3LAB[36]. Server had 80-core CPUs, 504GB RAM, 4 units of able to work separately GPUs of model "Tesla V100-SXM2-16GB". Model of the CPU was "Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz". Also, a great deal of processing power were required for embedding operations to cover images with steganography algorithms to create payload rated images. For the purpose of embedding operations, cloud computing resources which were provided by Safir Bulut Cloud Computing Services by B3LAB were utilized at the very best possible way. 22

cloud platform instances were launched as virtual instances. All of them were run separately but parallel on images. One instance had the power of 40 virtual CPUs, thus in order to create steganographic dataset 880 vCPUs were utilized for dataset containing 152592 units of training images and 80000 units of test images.

4.2.3 Discussions

We aimed to increase the success rate and to decrease the error rate on detecting stego images and cover images separately with this study. Our base comparison ground was 0.1bpp payload rated dataset. Because, it modified the very least number of pixels on images which makes it harder for the model to detect the stego images. We focused on the comparison of 0.1bpp dataset. Also, we followed the test results of other rated dataset to compare them with each other, to trace the development of the model in detecting stego images and cover images. Furthermore, we analyzed the results of transfer learning effects of the model from the base pre-trained model till hard to detect dataset such as 0.1bpp rated.

Transfer learning has made the base model more accustomed to the dataset. Via transfer learning, model have optimized weight values more and precise by processing images again and again. Therefore, it was successfully achieved that our main objective to train the model being able to differentiate even the slightest steganographic changes on images.

Least Significant Bit (LSB) steganography algorithm was a very base, more common and easy to detect method. The reason was simple about why we trained the model with LSB steganography algorithm. Because, we aimed that model would gain a fundamental understanding about what does the steganography change on the image. We would have evaluated the success rate of training with LSB by measuring the results of 1.0bpp payload dataset. As shown results for HUGO algorithm, it had performed much better to detect stego images.

As an overall outcome, we reached that the success rate was close but is in favor of the model with transfer learning in between early training dataset of 1.0bpp and 0.5bpp. But, after the 0.5bpp training, in between ratings of 0.5bpp and 0.1bpp, it was clearly on a large scale in favor of the transfer learning. F-Score evaluating showed that applying transfer learning highly affects the success rate on separately detecting cover images and stego images especially on 0.1bpp dataset.

4.3 Performance Evaluation

We assumed that accuracy success rate of %50 is the minimum base level which model could not learn and ineffective to detect stego or cover images.

We analyzed our models by the results of Training, Testing, Error Rates, Evaluating and F1-Scoring measurements. Training results have train loss and train accuracy values which show the learning progress of the model while in training. Testing results have validation loss and validation accuracy that indicate the testing progress of the model while in training. Error Rates are True and False results of the model to predict images in testing dataset as true or false predictions. If an image was a stego image and model predicted it as a stego image, then it was a True prediction. Evaluation measures the correct predictions of the model with loss and accuracy values. F1-Scoring has Precision, Recall and F1-Score measurements as indicated in Formula 1 and Formula 2. Precision is positive predictive value. An answer to question that how many selected images cover or stego are relevant to cover or stego labels? Recall is sensitivity value. An answer to question that how many cover images or stego images relevant items are selected?

Our study measured the prediction results using a threshold value of 0.5. Predictions below 0.5 threshold were accepted as cover image and labeled as 0. Predictions above threshold value were accepted as stego image and labeled as 1. If model predicted an image as 0.6 then we took that prediction as stego image result. Thus, we calculated precision, recall and F measure using the following formulas:

$$Precision = \frac{TP}{TP + FP} \quad , \quad Recall = \frac{TP}{TP + FN} \tag{4.1}$$

$$F = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.2}$$

According to these formulas, test results were categorized as true negative (TN), true positive (TP), false negative (FN) and false positive (FP) based on the following definitions that we used throughout the paper:

- TN: A test image is a cover image, and the model did not predict stego image.
- TP: A test image is a stego image, and the model did predict stego image.
- FN: A test image is a stego image, but the model did not predict stego image.
- FP: A test image is a cover image, but the model did predict stego image.

We created a public repository on GitHub[37] to contribute and to present our prototype implementations and test results to the open source community.

Chapter 5

Results

Training of the model was that learning or determining or optimizing for all of the parameters of the model which were weight values and bias values to the best possible values. In learning, algorithm updated the parameters of the model aiming to minimize the loss. Loss was a number indicating that how far the model predicted from true labels. If the model predicted true, then the loss was zero. The goal was to find a set of weights and biases which resulted on low loss and high on accuracy on all of the training examples. Training loss was the error rate value on the training sessions in the training datasets. Validation loss was the same as training loss only differed at the datasets which were validation set of data. Validation was run after the training of the network.

Loss is a function that takes the label output and predictions of the model and computes the error between them. Equation 5.1 show an error function called Mean Squared Error(MSE). The squared loss for one example is the square of the difference between the true label and the prediction of the model. MSE is the average squared loss for all the examples. The loss value is then used to adjust weights.

$$MeanSquaredError(MSE) = \frac{1}{N}\sum_{i=1}^{n}(y_i - \gamma_i)^2 \qquad (5.1)$$

- n is the total number of examples in the dataset.
- y is the true label value for the current example.
- γ is the predict value of the model for the current example.

5.1 HUGO Test Results

Analyzing the training process of models, results of training loss, training accuracy, validation loss and validation accuracy showed that generally models were able to learn images, functioning properly, smoothly and progressively. On transfer learning model, training and validating operations were completed successfully as expected. Nonetheless, at the embedding payload ratings of 0.8bpp and 0.5bpp, results showed unexpectedly a small increase in the accuracy or a small decrease in the loss values which generally the purpose at training operations of models. On the other hand, on non-transfer learning model, training results had big spacings between consecutive dataset. Accuracy was unexpectedly high at the level of embedding payload 0.8bpp. Both loss and accuracy training results were unexpectedly low and high on 0.5bpp payload dataset.

Both compared results of models including training, testing, error rates, evaluating and scoring had one common similarity that results generally differ in between two groups of dataset. One was between 1.0bpp and 0.5bpp, another group was between 0.5bpp and 0.1bpp embedded payload dataset. Generally, with transfer learning, at the first group, learning results were properly going and at the second group, learning results had more gaps between them. In general, with non-transfer learning, at first group, learning results had much more gaps between dataset and for the second group, learning results indicated that model failed to learn dataset.

By applying transfer learning method and LSB training, we obtained successful evaluating results on evaluation tests. We got low number of false predictions and high accuracy values in 1.0bpp and 0.9bpp dataset. It showed that model got adapted highly to the dataset and to the steganography processing. Model started predicting stego images better which seen at precision-1.0 measurement from on all of the tables.

On all the conjugate training comparisons, the model would predict a smaller number of error rates-false via transfer learning applied model than not applied model. Generally, the evaluating and prediction results were smoother going and they were in acceptable ranges.

Without transfer learning, test results between consecutive datasets were not smoothly progressive and had much bigger gaps and changes. 0.5bpp and 0.8bpp datasets showed unexpected successful increase in test results against the expected opinion. We expected as the payload embedded size gets smaller going from 1.0bpp to 0.1bpp, rate to successfully detect stego images would decrease. We thought that this may be the result of randomization of images in the training process of model. It was the first time that the model trained with these images. However, at the overall results, success rate of the model kept dropping dramatically, while embedding payload kept getting smaller, and this made the detection of the corruption on the images even more difficult.

After embedding payload 0.5bpp value, applied transfer learning model progressively and smoothly drop success rate as expected. But, the normal learning model did not change test results or train results and kept giving the same success result which was around 50% on train and test results.

Tables between Table 5.1 and Table 5.13 show the overall analysis on precision, recall and f1 score values indicate that applying transfer learning greatly improved the success rate of the model at detecting both cover images and stego images even on the most difficult embedded payload dataset. Explanations of the table are as follows:

- Precision 0.0 shows that how much test cover images that were predicted by the model are correctly predicted.
- Precision 1.0 shows that how much test stego images that were predicted by the model are correctly predicted.
  - Table 5.1 and Table 5.2 show precision results of the HUGO Steganography algorithm.

- Recall 0.0 means that how much test cover images were correctly predicted by the model.
- Recall 1.0 means that how much test stego images were correctly predicted by the model.
  - Table 5.3 and Table 5.4 show recall results of the HUGO Steganography algorithm.

- F1-Score 0.0 indicates the overall success rate of model on the predicting test cover images.

Table 5.1: HUGO Algorithm - Transfer Learning Model Precision Scores

Payload	**Precision**		**average/total**
	0.0	**1.0**	**80000**
LSB	1.00	1.00	1.00
1.0	0.96	0.84	0.90
0.9	0.94	0.78	0.86
0.8	0.93	0.78	0.85
0.7	0.94	0.71	0.83
0.6	0.93	0.69	0.81
0.5	0.92	0.66	0.79
0.4	0.88	0.62	0.75
0.3	0.82	0.59	0.71
0.2	0.74	0.59	0.66
0.1	**0.66**	**0.56**	**0.61**

Table 5.2: HUGO Algorithm - Non-Transfer Learning Model Precision Scores

Payload	**Precision**		**average/total**
	0.0	**1.0**	**80000**
1.0	0.97	0.82	0.89
0.9	0.94	0.75	0.85
0.8	0.90	0.79	0.84
0.7	0.91	0.69	0.80
0.6	0.68	0.73	0.71
0.5	0.78	0.69	0.74
0.4	0.50	0.51	0.51
0.3	0.51	0.51	0.51
0.2	0.50	0.50	0.50
0.1	**0.50**	**0.50**	**0.50**

Table 5.3: HUGO Algorithm - Transfer Learning Model Recall Scores

Payload	**Recall**		**average/total**
	0.0	**1.0**	**80000**
LSB	1.00	1.00	1.00
1.0	0.81	0.96	0.89
0.9	0.73	0.96	0.84
0.8	0.74	0.94	0.84
0.7	0.60	0.96	0.78
0.6	0.57	0.96	0.77
0.5	0.51	0.95	0.73
0.4	0.42	0.95	0.68
0.3	0.35	0.92	0.64
0.2	0.39	0.86	0.63
0.1	**0.37**	**0.81**	**0.59**

Table 5.4: HUGO Algorithm - Non-Transfer Learning Model
Recall Scores

Payload	**Recall**		**average/total**
	0.0	**1.0**	**80000**
1.0	0.79	0.97	0.88
0.9	0.68	0.96	0.82
0.8	0.75	0.91	0.83
0.7	0.58	0.94	0.76
0.6	0.77	0.64	0.70
0.5	0.64	0.82	0.73
0.4	0.74	0.27	0.51
0.3	0.60	0.41	0.51
0.2	0.96	0.04	0.50
0.1	**0.90**	**0.11**	**0.50**

- F1-Score 1.0 indicates the overall success rate of model on the predicting test stego images.
  - Table 5.5 and Table 5.6 show F1 Score results of the HUGO Steganography algorithm.

Table 5.5: HUGO Algorithm - Transfer Learning Model
F1 Scores

Payload	**f1-score**		**average/total**
	0.0	**1.0**	**80000**
LSB	1.00	1.00	1.00
1.0	0.88	0.90	0.89
0.9	0.82	0.86	0.84
0.8	0.82	0.85	0.84
0.7	0.73	0.82	0.77
0.6	0.71	0.80	0.76
0.5	0.65	0.78	0.72
0.4	0.57	0.75	0.66
0.3	0.49	0.72	0..61
0.2	0.51	0.70	0.60
0.1	**0.47**	**0.66**	**0.57**

Table 5.7 gathers the evaluation results for comparing the overall performance of the model via Precision, Recall and F1-Score metrics for HUGO algorithm. It is a combination of the average columns of the previous tables. The table compares the models that trained with and without applying Transfer Learning. The table summarizes that applying transfer learning method results better than not applying transfer learning method. Our base ground for comparison is the 0.1bpp payload rated dataset. Precision value for TL-model is 0.61 and 0.50 for the non-TL-model. Recall value for TL-model is 0.59 and 0.50 for the non-TL-model. F1-score value for TL-model is 0.57 and 0.41 for the non-TL

Table 5.6: HUGO Algorithm - Non-Transfer Learning Model F1 Scores

Payload	**f1-score**		**average/total**
	0.0	**1.0**	**80000**
1.0	0.87	0.89	0.88
0.9	0.79	0.84	0.82
0.8	0.82	0.85	0.83
0.7	0.71	0.80	0.76
0.6	0.72	0.68	0.70
0.5	0.70	0.75	0.73
0.4	0.60	0.35	0.48
0.3	0.55	0.45	0.50
0.2	0.66	0.07	0.36
0.1	**0.64**	**0.18**	**0.41**

model. It is clearly shown that model with transfer learning has better performance at detecting stego and cover images.

Table 5.7: HUGO Algorithm - Overall Performance Comparison of Results

	HUGO					
Payload	**TL-Precision**	**Non-TL Precision**	**TL-Recall**	**Non-TL Recall**	**TL-f1-score**	**Non-TL f1-score**
LSB	1.00	Not Applicable	1.00	Not Applicable	1.00	Not Applicable
1.0	0.90	0.89	0.89	0.88	0.89	0.88
0.9	0.86	0.85	0.84	0.82	0.84	0.82
0.8	0.85	0.84	0.84	0.83	0.84	0.83
0.7	0.83	0.80	0.78	0.76	0.77	0.76
0.6	0.81	0.71	0.77	0.70	0.76	0.70
0.5	0.79	0.74	0.73	0.73	0.72	0.73
0.4	0.75	0.51	0.68	0.51	0.66	0.48
0.3	0.71	0.51	0.64	0.51	0.61	0.50
0.2	0.66	0.50	0.63	0.50	0.60	0.36
0.1	**0.61**	**0.50**	**0.59**	**0.50**	**0.57**	**0.41**

5.2 WOW Test Results

Analyzing the overall test results of WOW steganography algorithm, it was similar with HUGO results.

Transfer learning had affected the model very positively on distinction of cover and stego images.

Model without transfer learning had close test results with applied transfer learning model at the first three embedded payload dataset, 1.0bpp, 0.9bpp and 0.8bpp. But it quickly and progressively decreased to base accuracy level of 50%.

Precision scores on Table 5.8 and Table 5.9 explain that how our two branch of models truly predicted cover or stego images. Transfer Learning Model was much more successful that Non-Transfer Learning Model at detecting cover and stego images. TL-Model had cover precision value of 0.65 and stego precision value of 0.58 on 0.1 Payload rate. In the mean time, Non-TL-Model had cover precision value of 0.50 and stego precision value of 0.50. It meant that non-tl-model learn nothing from dataset and it rolled the dice on the images. But TL-model could detect cover and stego images while it seemed that it decreased on the overall ongoing training on the payload rates.

Table 5.8: WOW Algorithm - Transfer Learning Model Precision Scores

Payload	**Precision**		**average/total**
	0.0	**1.0**	**80000**
LSB	1.00	1.00	1.00
1.0	0.98	0.78	0.88
0.9	0.97	0.79	0.88
0.8	0.97	0.78	0.88
0.7	0.95	0.77	0.86
0.6	0.94	0.74	0.84
0.5	0.91	0.72	0.82
0.4	0.88	0.69	0.79
0.3	0.84	0.66	0.75
0.2	0.77	0.62	0.69
0.1	**0.65**	**0.58**	**0.61**

Recall scores are presented as you can see on the Table 5.10 and Table 5.11. It is clear that as sees on the tables with 0.1bpp payload rate, TL-Model had shown more performance on finding cover images and stego images. The Non-TL-Model had 0.96 recall score at the 0.1bpp payload on detecting cover images. It meant that model could not understand if it was cover image or stego image. Even so model predicted it as a

Table 5.9: WOW Algorithm - Non-Transfer Learning Model Precision Scores

Payload	**Precision**		**avg/total**
	0.0	**1.0**	**80000**
1.0	0.98	0.81	0.89
0.9	0.95	0.78	0.86
0.8	0.97	0.75	0.86
0.7	0.93	0.67	0.80
0.6	0.88	0.66	0.77
0.5	0.81	0.65	0.73
0.4	0.75	0.64	0.70
0.3	0.53	0.51	0.52
0.2	0.50	0.50	0.50
0.1	**0.50**	**0.50**	**0.50**

cover image. In other case, TL-Model had 0.44 recall score at the same point. It was 0.04 recall score for the Non-TL-Model and 0.76 recall score for the TL-Model on stego images. This meant that TL-Model had more successful results on detecting far above the average number of stego images in the test dataset.

Table 5.10: WOW Algorithm - Transfer Learning Model Recall Scores

Payload	**Recall**		**average/total**
	0.0	**1.0**	**80000**
LSB	1.00	1.00	1.00
1.0	0.72	0.98	0.85
0.9	0.74	0.98	0.86
0.8	0.73	0.97	0.85
0.7	0.71	0.97	0.84
0.6	0.66	0.96	0.81
0.5	0.63	0.94	0.79
0.4	0.60	0.92	0.76
0.3	0.53	0.90	0.71
0.2	0.47	0.86	0.66
0.1	**0.44**	**0.76**	**0.60**

Normal model failed at detecting stego images with a f1-score of 0.08. It was shown on the Table 5.13 that intersected at the row of 0.1bpp payload rate and at 1.0 column of f1-score column. On the contrary, as in Table 5.12, transfer learning model had more success rate for detecting stego images with a f1-score 0.65.

Table 5.11: WOW Algorithm - Non-Transfer Learning Model Recall Scores

Payload	**Recall**		**avg/total**
	0.0	**1.0**	**80000**
1.0	0.77	0.98	0.88
0.9	0.72	0.96	0.84
0.8	0.67	0.98	0.83
0.7	0.53	0.96	0.75
0.6	0.53	0.93	0.73
0.5	0.53	0.87	0.70
0.4	0.55	0.82	0.68
0.3	0.32	0.72	0.52
0.2	0.84	0.16	0.50
0.1	**0.96**	**0.04**	**0.50**

Table 5.12: WOW Algorithm - Transfer Learning Model F1 Scores

Payload	**f1-score**		**average/total**
	0.0	**1.0**	**80000**
LSB	1.00	1.00	1.00
1.0	0.83	0.87	0.85
0.9	0.84	0.87	0.85
0.8	0.83	0.87	0.85
0.7	0.81	0.86	0.83
0.6	0.77	0.83	0.80
0.5	0.75	0.81	0.78
0.4	0.71	0.79	0.75
0.3	0.65	0.76	0.71
0.2	0.58	0.72	0.65
0.1	**0.53**	**0.65**	**0.59**

Table 5.13: WOW Algorithm - Non-Transfer Learning Model F1 Scores

Payload	**f1-score**		**avg/total**
	0.0	**1.0**	**80000**
1.0	0.86	0.89	0.88
0.9	0.82	0.86	0.84
0.8	0.79	0.85	0.82
0.7	0.68	0.79	0.73
0.6	0.66	0.77	0.72
0.5	0.64	0.75	0.69
0.4	0.63	0.72	0.68
0.3	0.40	0.60	0.50
0.2	0.63	0.25	0.44
0.1	**0.66**	**0.08**	**0.37**

TABLE 5.14: WOW Algorithm - Overall Performance Comparison of Results

	WOW					
Payload	**TL-Precision**	**Non-TL Precision**	**TL-Recall**	**Non-TL Recall**	**TL-f1-score**	**Non-TL f1-score**
LSB	1.00	Not Applicable	1.00	Not Applicable	1.00	Not Applicable
1.0	0.88	0.89	0.85	0.88	0.85	0.88
0.9	0.88	0.86	0.86	0.84	0.85	0.84
0.8	0.88	0.86	0.85	0.83	0.85	0.82
0.7	0.86	0.80	0.84	0.75	0.83	0.73
0.6	0.84	0.77	0.81	0.73	0.80	0.72
0.5	0.82	0.73	0.79	0.70	0.78	0.69
0.4	0.79	0.70	0.76	0.68	0.75	0.68
0.3	0.75	0.52	0.71	0.52	0.71	0.50
0.2	0.69	0.50	0.66	0.50	0.65	0.44
0.1	**0.61**	**0.50**	**0.60**	**0.50**	**0.59**	**0.37**

The Table 5.14 is similar to Table 5.7. It gathers the evaluation results for comparing the overall performance of the model via Precision, Recall and F1-Score metrics for WOW algorithm. It is a combination of the average columns of the previous tables. The table compares the models that trained with and without applying Transfer Learning. The table summarizes that applying transfer learning method results better than not applying transfer learning method. Our base ground for comparison is the 0.1bpp payload rated dataset. Precision value for TL-model is 0.61 and 0.50 for the non-TL-model. Recall value for TL-model is 0.60 and 0.50 for the non-TL-model. F1-score value for TL-model is 0.59 and 0.37 for the non-TL model. It is clearly shown that model with transfer learning has better performance at detecting stego and cover images. HUGO and WOW results are very close to each other for the 0.1bpp dataset.

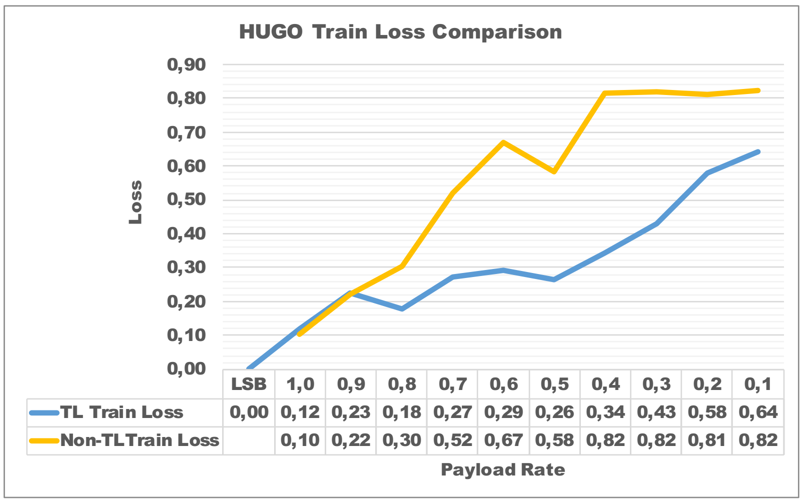
5.3 Result Comparisons

We compared Transfer Learning method applied model and its corresponding Non-Applied Transfer Learning model. We evaluated the results of the every step starting from training the model to scoring the results. In this chapter, we presented the comparison results of training, testing, evaluation, prediction and scoring of models. The presented values were the last obtained after the run of the last epoch. 0.1 Payload Rated dataset was the most difficult to detect stego images.

5.3.1 Train Comparisons

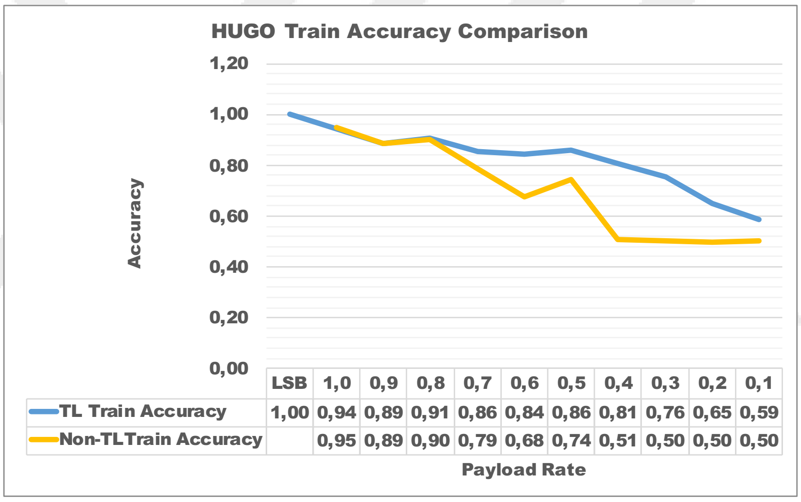
The comparisons here were the results of the training process of the models. Training Loss showed that how far the models calculate the results from the true results on every epoch. On Figure 5.1 and Figure 5.3, Transfer Learning applied model had mostly lower loss values than the opposite non-applied model.

Training Accuracy indicated that how much the model calculated the true results. As seen on Figure 5.2 and Figure 5.4, TL Model calculated more true predictions and modified parameters of model more truly.



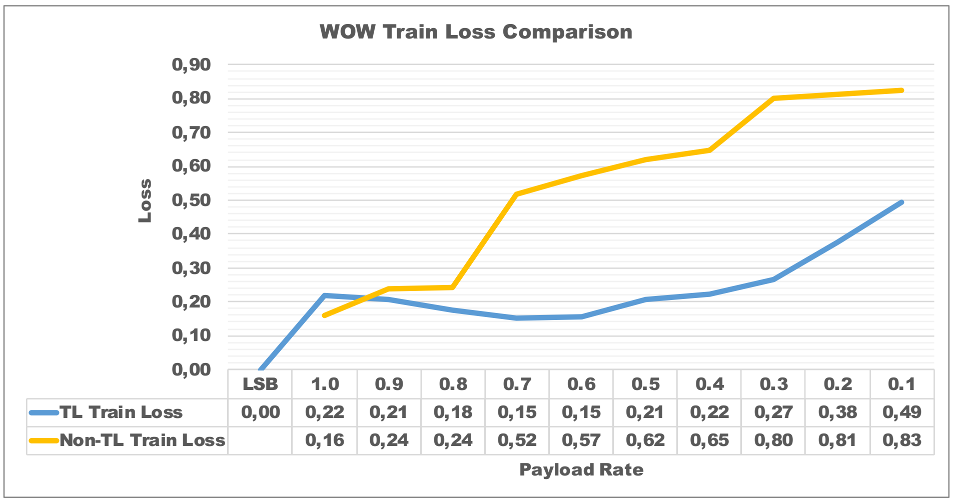
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Train Loss	0,00	0,12	0,23	0,18	0,27	0,29	0,26	0,34	0,43	0,58	0,64
Non-TLTrain Loss		0,10	0,22	0,30	0,52	0,67	0,58	0,82	0,82	0,81	0,82

FIGURE 5.1: HUGO Models - Training Loss Comparison



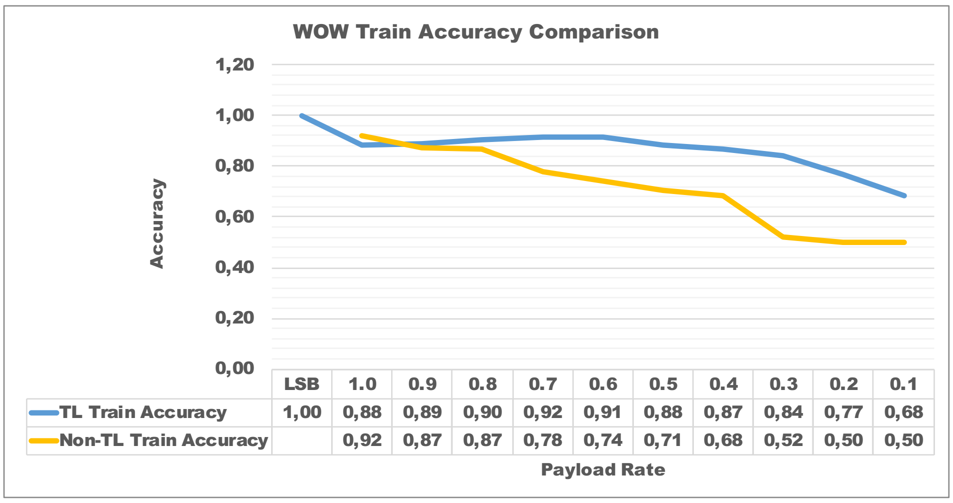
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Train Accuracy	1,00	0,94	0,89	0,91	0,86	0,84	0,86	0,81	0,76	0,65	0,59
Non-TLTrain Accuracy		0,95	0,89	0,90	0,79	0,68	0,74	0,51	0,50	0,50	0,50

FIGURE 5.2: HUGO Models - Training Accuracy Comparison



	LSB	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
TL Train Loss	0,00	0,22	0,21	0,18	0,15	0,15	0,21	0,22	0,27	0,38	0,49
Non-TL Train Loss		0,16	0,24	0,24	0,52	0,57	0,62	0,65	0,80	0,81	0,83

FIGURE 5.3: WOW Models - Training Loss Comparison

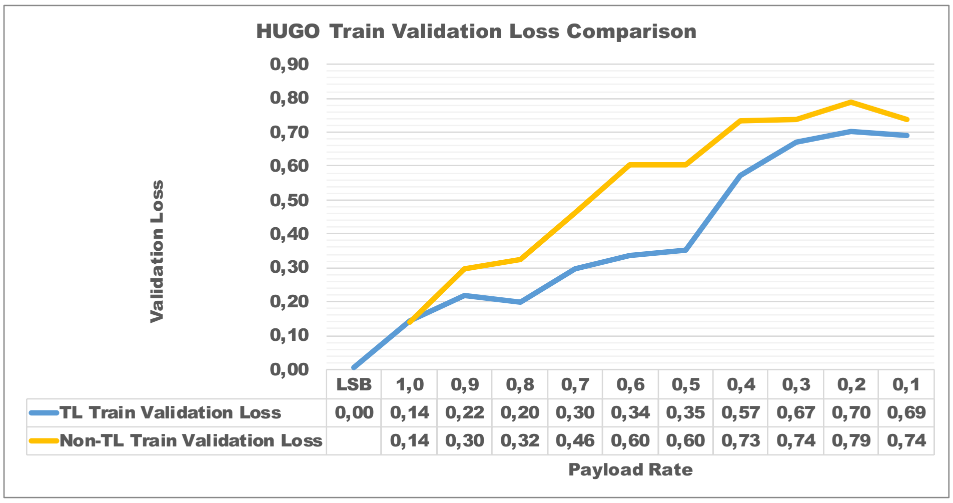


	LSB	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
TL Train Accuracy	1,00	0,88	0,89	0,90	0,92	0,91	0,88	0,87	0,84	0,77	0,68
Non-TL Train Accuracy		0,92	0,87	0,87	0,78	0,74	0,71	0,68	0,52	0,50	0,50

FIGURE 5.4: WOW Models - Training Accuracy Comparison

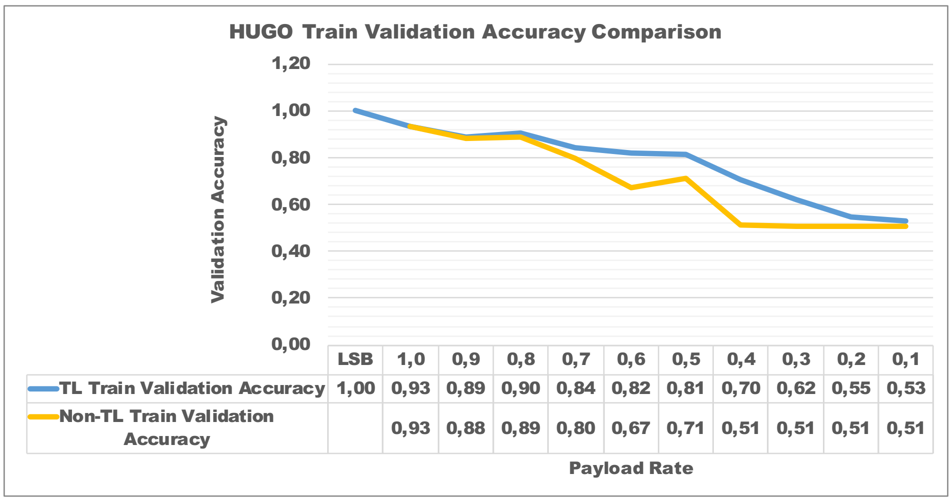
5.3.2 Train Validation Comparisons

At the end of each epoch in part of the training process, some train data were tested on the model to evaluate the loss, the hyperparameters and other metrics. Validation dataset was a small part of the training dataset and it was different from training dataset. Model did not train or learn on validation dataset. Instead, it was used to estimate the skills of the model such as loss and accuracy of the model. Figure 5.5 and Figure 5.7 showed the loss of the models on each epoch. Figure 5.6 and Figure 5.8 showed the loss and accuracy skills of the model on the validation dataset. Validation dataset was 3815-unit images included in the training dataset which was 0.025% of total training dataset.



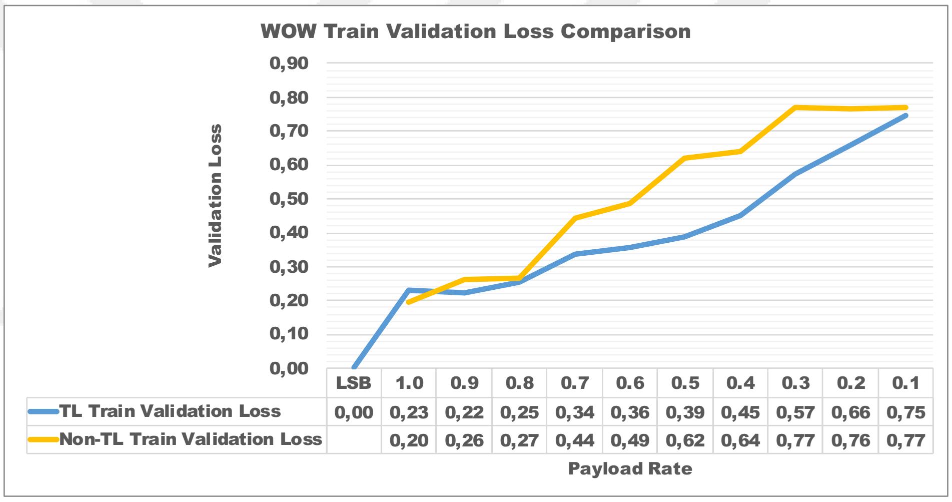
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Train Validation Loss	0,00	0,14	0,22	0,20	0,30	0,34	0,35	0,57	0,67	0,70	0,69
Non-TL Train Validation Loss		0,14	0,30	0,32	0,46	0,60	0,60	0,73	0,74	0,79	0,74

FIGURE 5.5: HUGO Models - Train Validation Loss Comparison



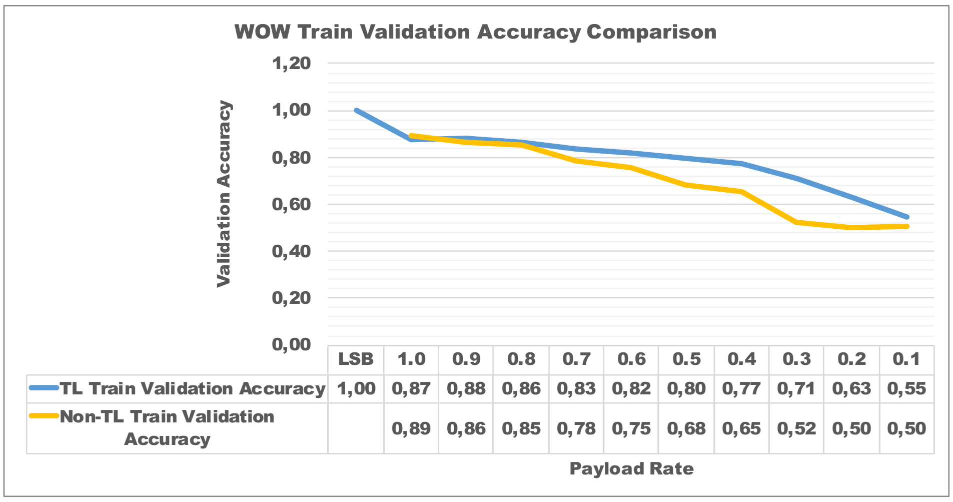
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Train Validation Accuracy	1,00	0,93	0,89	0,90	0,84	0,82	0,81	0,70	0,62	0,55	0,53
Non-TL Train Validation Accuracy		0,93	0,88	0,89	0,80	0,67	0,71	0,51	0,51	0,51	0,51

FIGURE 5.6: HUGO Models - Train Validation Accuracy Comparison



	LSB	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
TL Train Validation Loss	0,00	0,23	0,22	0,25	0,34	0,36	0,39	0,45	0,57	0,66	0,75
Non-TL Train Validation Loss		0,20	0,26	0,27	0,44	0,49	0,62	0,64	0,77	0,76	0,77

FIGURE 5.7: WOW Models - Train Validation Loss Comparison



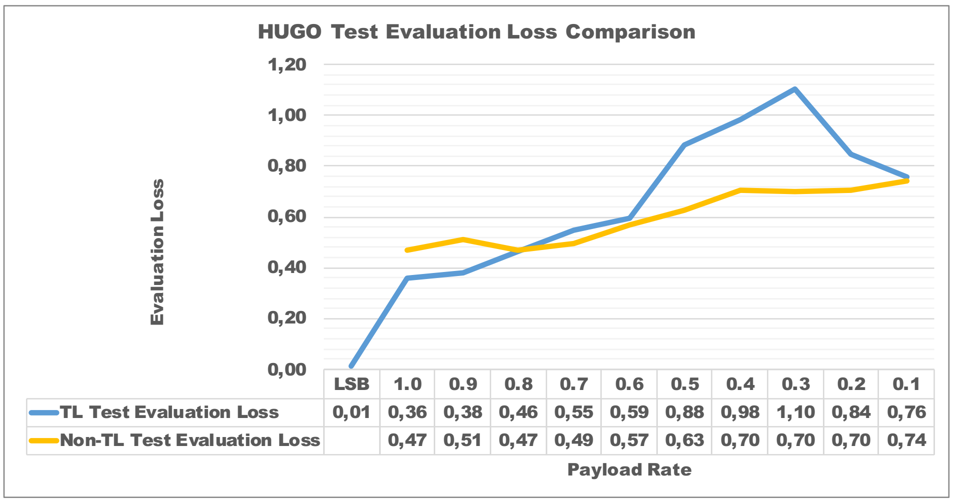
	LSB	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
TL Train Validation Accuracy	1,00	0,87	0,88	0,86	0,83	0,82	0,80	0,77	0,71	0,63	0,55
Non-TL Train Validation Accuracy		0,89	0,86	0,85	0,78	0,75	0,68	0,65	0,52	0,50	0,50

FIGURE 5.8: WOW Models - Train Validation Accuracy Comparison

5.3.3 Evaluation Comparisons

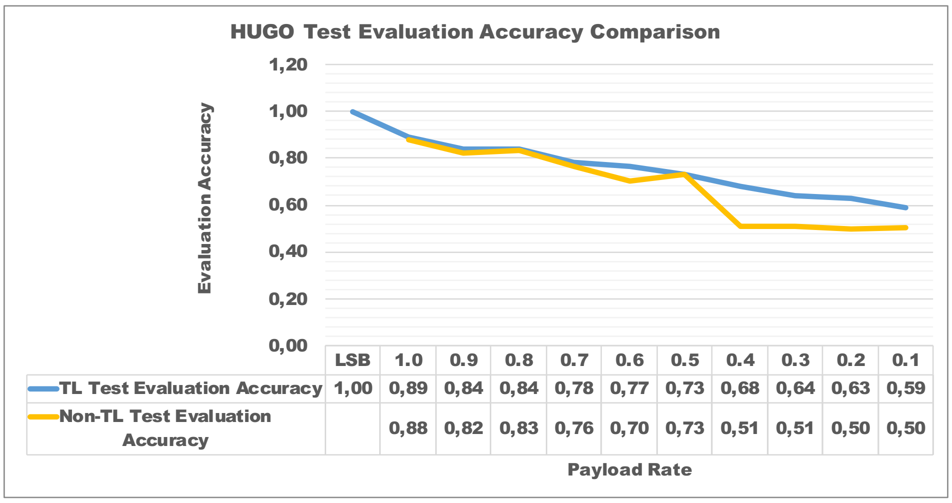
Evaluation step was similar to the training session only but evaluation step did not update the weights. It returned the chosen metric values. We chose loss and accuracy results. Evaluation worked in batch mode. A given number of input images were evaluated at each iteration. Loss value was the metric that implies the rate of incorrect predicted input images after an iteration. Unless the model had overfitted to the dataset, the lower the loss value, the better a model results. Loss value was not a percentage. It was a summation of the errors made for each example in datasets. Loss value was often used for the model in order to find the best parameter values such as weights in neural network. In the training process, loss value was the value to be optimized by updating weights. Accuracy value was the metric that implies the rate of correct predicted input images after an iteration.

Evaluation showed the performance of the model on the dataset which the model sees for the first time. It was pretty close to the real world dataset. It showed us the overall performance of the model. Our test dataset contains 40000 units of cropped images for cover images and for stego images, totaling to 80000 units of test images. We used different test dataset from train dataset. But images in the datasets are close each other. Loss performance of the models were presented at the Figure 5.9 and Figure 5.11. Also, the performance of how accurately the model predicted cover and stego images were shown at Figure 5.10 and Figure 5.12.



	LSB	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
TL Test Evaluation Loss	0,01	0,36	0,38	0,46	0,55	0,59	0,88	0,98	1,10	0,84	0,76
Non-TL Test Evaluation Loss		0,47	0,51	0,47	0,49	0,57	0,63	0,70	0,70	0,70	0,74

FIGURE 5.9: HUGO Models - Test Evaluation Loss Comparison



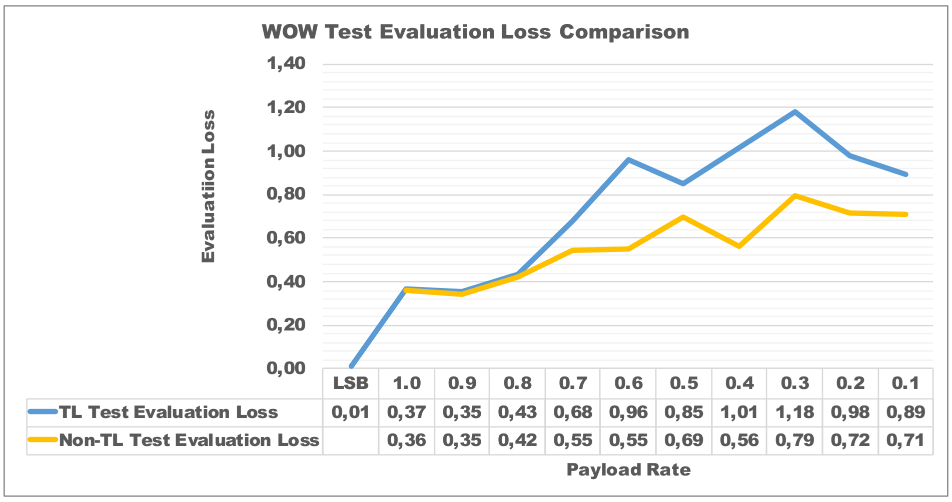
	LSB	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
TL Test Evaluation Accuracy	1,00	0,89	0,84	0,84	0,78	0,77	0,73	0,68	0,64	0,63	0,59
Non-TL Test Evaluation Accuracy		0,88	0,82	0,83	0,76	0,70	0,73	0,51	0,51	0,50	0,50

FIGURE 5.10: HUGO Models - Test Evaluation Accuracy Comparison

The each loss and accuracy values in the figures were obtained separately from each different payload rated dataset after one evaluation run. We expected an increase slope at the loss values starting from 1.0bpp to 0.1bpp. Because model could easily learn and detect 1.0bpp payload stego and cover images. Just the opposite, 0.1bpp payload images were the hardest dataset. Thus, the model would got the greatest loss value at 0.1bpp payload rated images. At each step to the 0.1bpp dataset, the increase rate of the number of incorrect predictions changed differently. Since loss value was the total of errors, these differences could cause the unexpected increase rate for the loss values at 0.4bpp and 0.3bpp payload rates of Figure 5.9 and Figure 5.11. Loss increased as the predicted probability diverges from the actual label. Evaluation loss values were found according to the rate of true and false predictions of payload rated datasets. It was calculated by the chosen metrics while building the model. Loss was calculated by binary cross entropy formula as shown in the Equation 5.2. It was generally used for classification of images for only two classes. True or not. Stego or not stego.

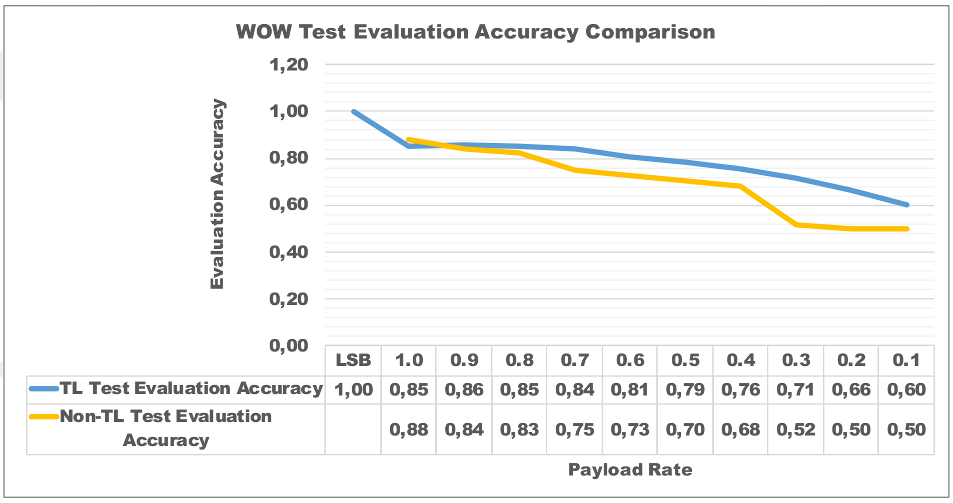
$$CrossEntropyLoss = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{5.2}$$

i is example image. N is total number of images in dataset. y_i is the true label of i^{th} input image. $\hat{y}_i$ is the predicted output of the model for i^{th} image.



	LSB	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
TL Test Evaluation Loss	0,01	0,37	0,35	0,43	0,68	0,96	0,85	1,01	1,18	0,98	0,89
Non-TL Test Evaluation Loss		0,36	0,35	0,42	0,55	0,55	0,69	0,56	0,79	0,72	0,71

FIGURE 5.11: WOW Models - Test Evaluation Loss Comparison



	LSB	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
TL Test Evaluation Accuracy	1,00	0,85	0,86	0,85	0,84	0,81	0,79	0,76	0,71	0,66	0,60
Non-TL Test Evaluation Accuracy		0,88	0,84	0,83	0,75	0,73	0,70	0,68	0,52	0,50	0,50

FIGURE 5.12: WOW Models - Test Evaluation Accuracy Comparison

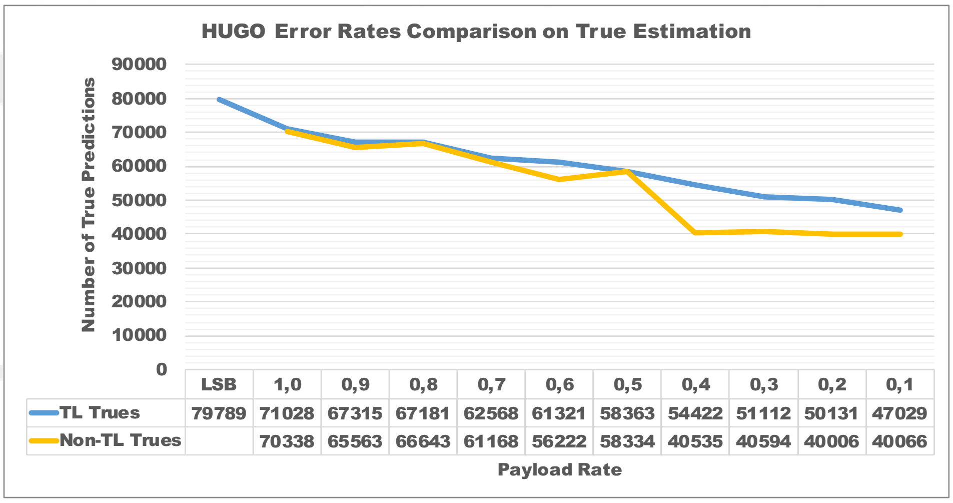
5.3.4 Prediction Comparisons

How many predictions of test images were predicted true by the model out of 80000 test images?

How many predictions of test images were predicted false by the model out of 80000 test images?

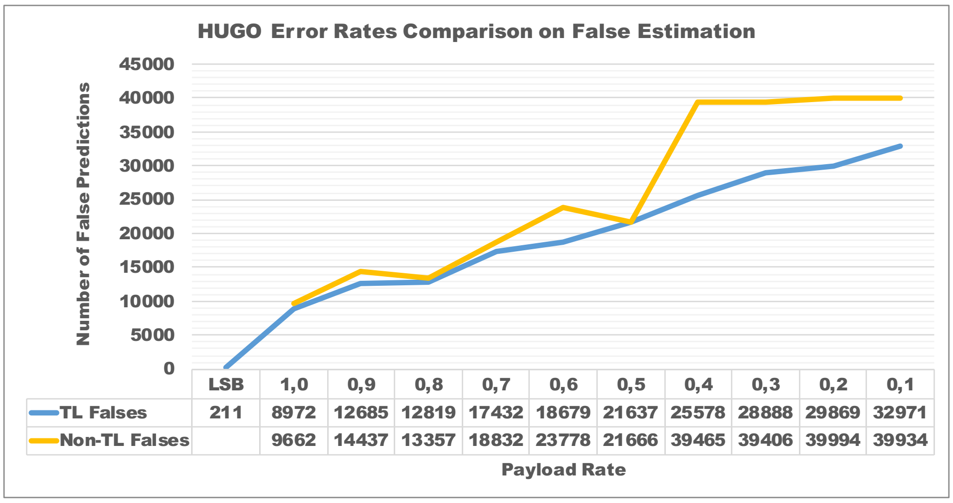
These comparison figures score the result predictions of the models. It predicted on test dataset. One prediction range of the model is between 0 and 1. Model predicted a value inside the prediction range. We accepted 0.5 is the threshold level of prediction range. Between 0 and 0.5(not included), it was a cover image. Between 0.5(included) and 1.0, it is a stego image. If prediction value was 0.2, we accepted it as a cover prediction and convert the value to 0(zero). Or if model predicted a value for an image as 0.6, we accepted the prediction value as stego image and converted it to 1. After applying the threshold limit to the prediction values, we compared the prediction value with the correct 0-cover or 1-stego value of the images. If prediction value and correct value matched then it was a true prediction. But if they did not match then we counted it as a false prediction from model.

Figure 5.13 and Figure 5.15 show the number of true predictions from HUGO and WOW steganography algorithms. It compares transfer learning applied model and transfer learning not applied model. Similarly, Figure 5.14 and Figure 5.16 show the comparisons of models on the false predictions.



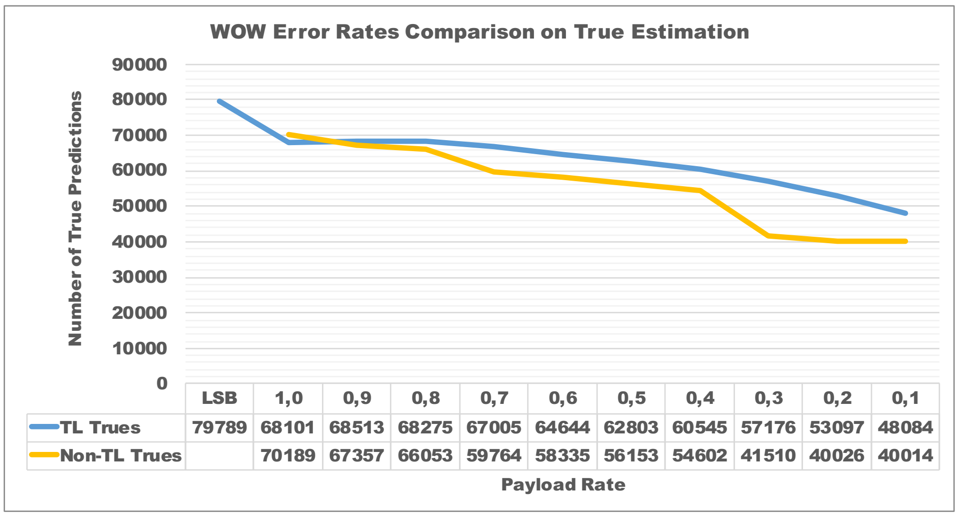
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Trues	79789	71028	67315	67181	62568	61321	58363	54422	51112	50131	47029
Non-TL Trues		70338	65563	66643	61168	56222	58334	40535	40594	40006	40066

FIGURE 5.13: HUGO Models - Error Rates True Estimation Comparison



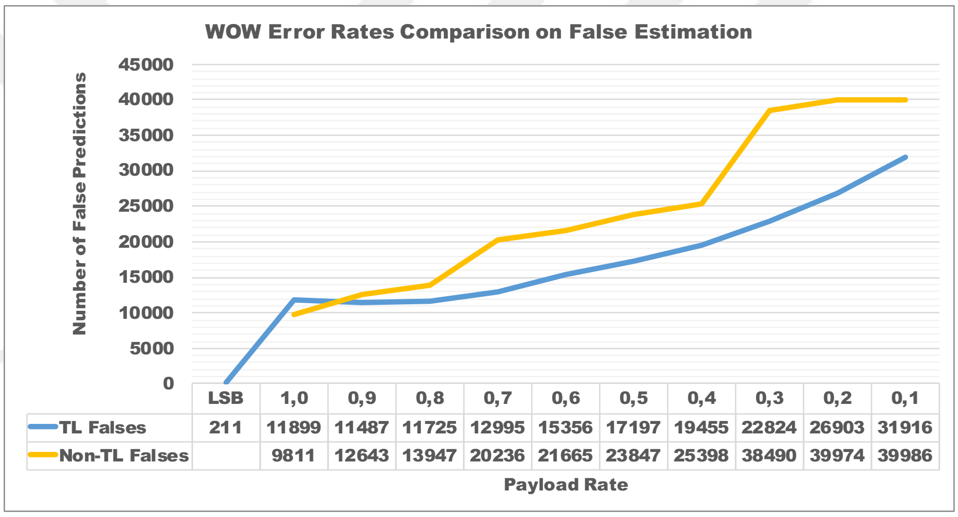
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Falses	211	8972	12685	12819	17432	18679	21637	25578	28888	29869	32971
Non-TL Falses		9662	14437	13357	18832	23778	21666	39465	39406	39994	39934

FIGURE 5.14: HUGO Models - Error Rates False Estimation Comparison



	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Trues	79789	68101	68513	68275	67005	64644	62803	60545	57176	53097	48084
Non-TL Trues		70189	67357	66053	59764	58335	56153	54602	41510	40026	40014

FIGURE 5.15: WOW Models - Error Rates True Estimation Comparison



	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Falses	211	11899	11487	11725	12995	15356	17197	19455	22824	26903	31916
Non-TL Falses		9811	12643	13947	20236	21665	23847	25398	38490	39974	39986

FIGURE 5.16: WOW Models - Error Rates False Estimation Comparison

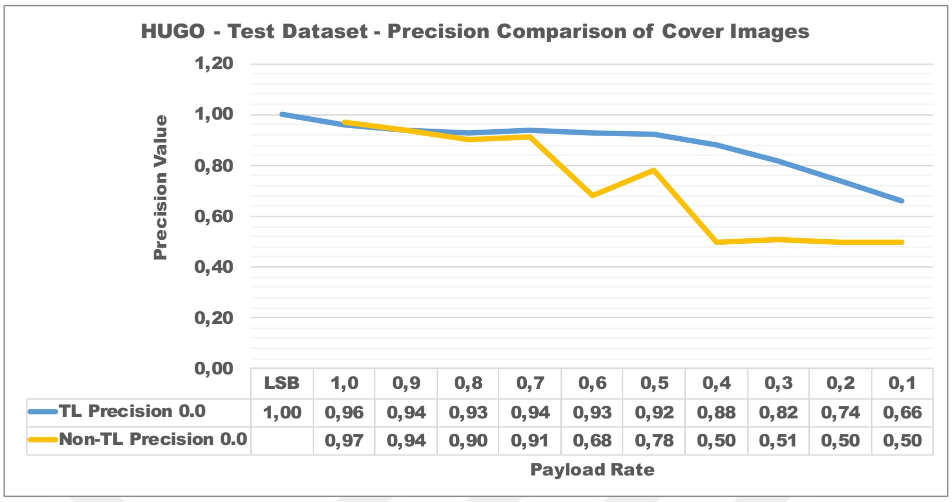
5.3.5 Precision Comparisons

Precision was about how precise or accurate the predictions of the model. Precision was a good measuring tool to determine false positive rate. It showed the ratio of correct predictions of cover or stego images to the total predictions of cover or stego images. High precision ratio related to the low False-Positive ratio.

Figure 5.17 and Figure 5.20 presented the precision ratio of cover images out of 40000 images.

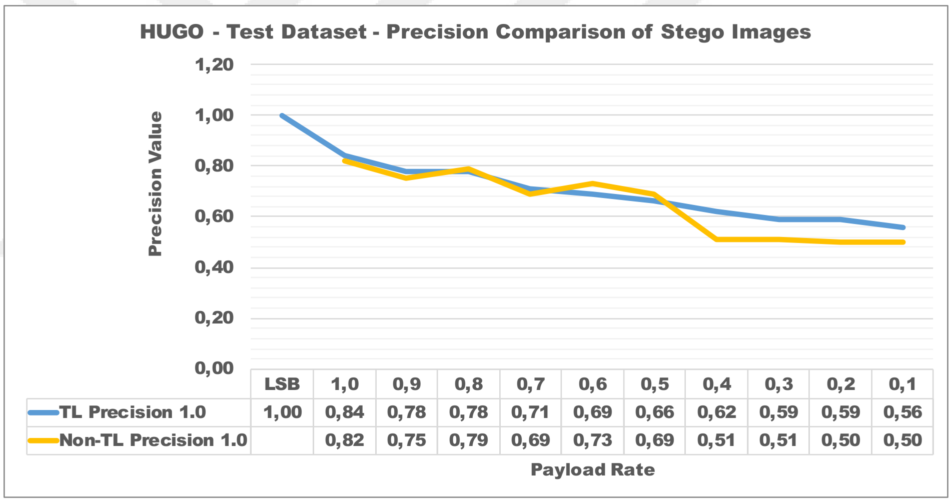
Figure 5.18 and Figure 5.21 presented the precision ratio of stego images out of 40000 images.

Figure 5.19 and Figure 5.22 presented total precision ratio both cover and stego images out of 80000 images.



	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Precision 0.0	1,00	0,96	0,94	0,93	0,94	0,93	0,92	0,88	0,82	0,74	0,66
Non-TL Precision 0.0		0,97	0,94	0,90	0,91	0,68	0,78	0,50	0,51	0,50	0,50

Figure 5.17: HUGO Models - Test Dataset Precision Comparison of Cover Images



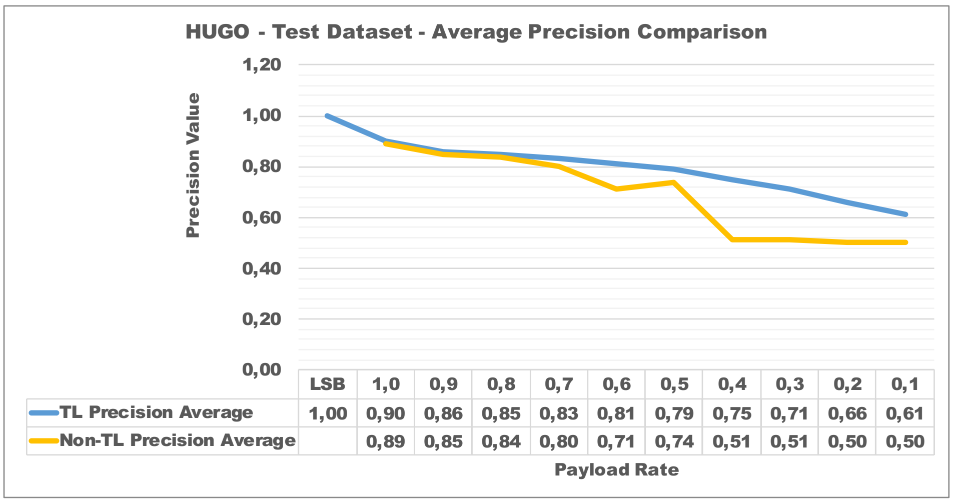
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Precision 1.0	1,00	0,84	0,78	0,78	0,71	0,69	0,66	0,62	0,59	0,59	0,56
Non-TL Precision 1.0		0,82	0,75	0,79	0,69	0,73	0,69	0,51	0,51	0,50	0,50

Figure 5.18: HUGO Models - Test Dataset Precision Comparison of Stego Images

5.3.6 Recall Comparisons

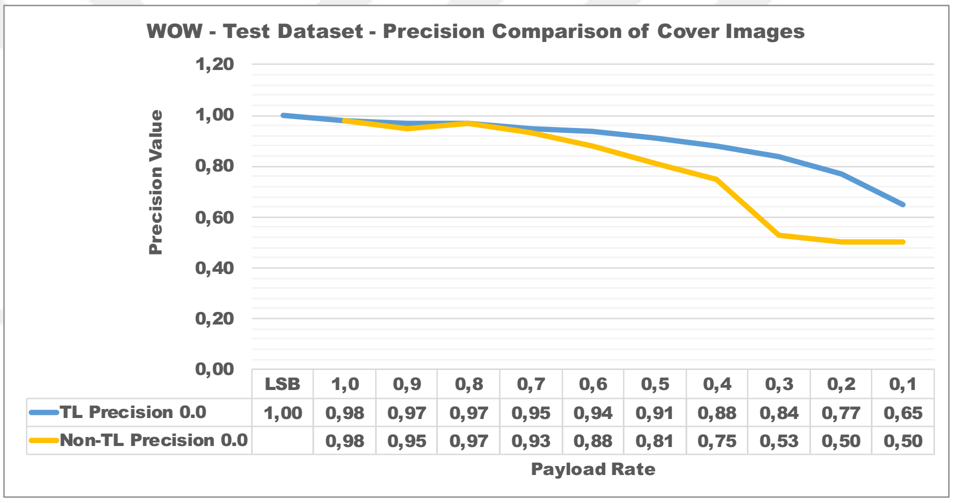
Recall was sensitivity. It was the ratio of correctly predicted cover or stego images to the all cover or stego images respectively. If it was to calculate the recall ratio for cover images, then recall ratio answered the question that of all the cover images we have, how many did we predict. It was the same explanation of recall for the stego images.

Figure 5.23 and Figure 5.26 presented the recall ratio of cover images out of 40000 images. Respectively on comparison figures, after 0.7bpp dataset and 0.3bpp dataset, Non-TL recall values for cover images have increased values over the corresponding TL Recall



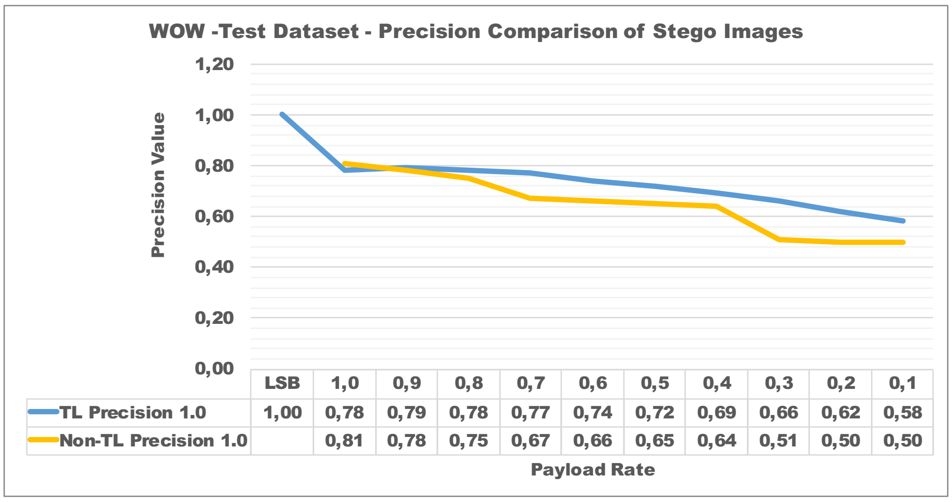
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Precision Average	1,00	0,90	0,86	0,85	0,83	0,81	0,79	0,75	0,71	0,66	0,61
Non-TL Precision Average		0,89	0,85	0,84	0,80	0,71	0,74	0,51	0,51	0,50	0,50

FIGURE 5.19: HUGO Models - Test Dataset Average Precision Comparison



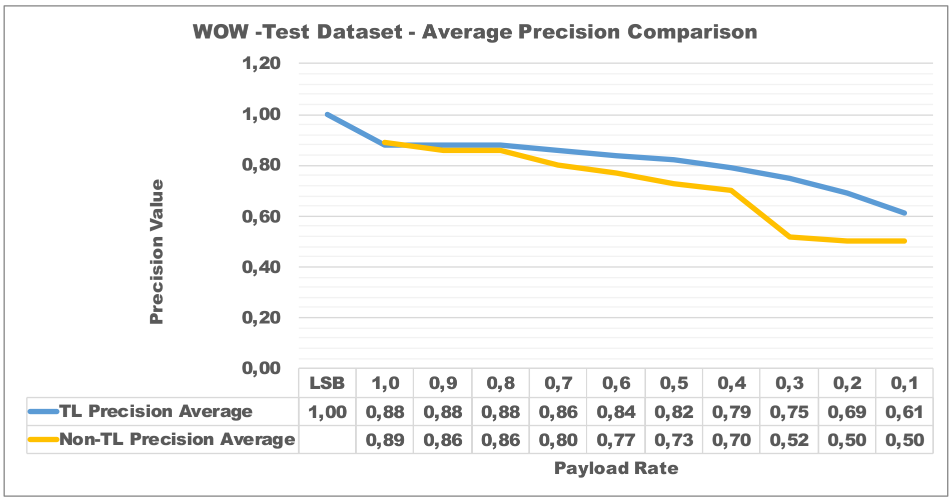
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Precision 0.0	1,00	0,98	0,97	0,97	0,95	0,94	0,91	0,88	0,84	0,77	0,65
Non-TL Precision 0.0		0,98	0,95	0,97	0,93	0,88	0,81	0,75	0,53	0,50	0,50

FIGURE 5.20: WOW Models - Test Dataset Precision Comparison of Cover Images



	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Precision 1.0	1,00	0,78	0,79	0,78	0,77	0,74	0,72	0,69	0,66	0,62	0,58
Non-TL Precision 1.0		0,81	0,78	0,75	0,67	0,66	0,65	0,64	0,51	0,50	0,50

FIGURE 5.21: WOW Models - Test Dataset Precision Comparison of Stego Images



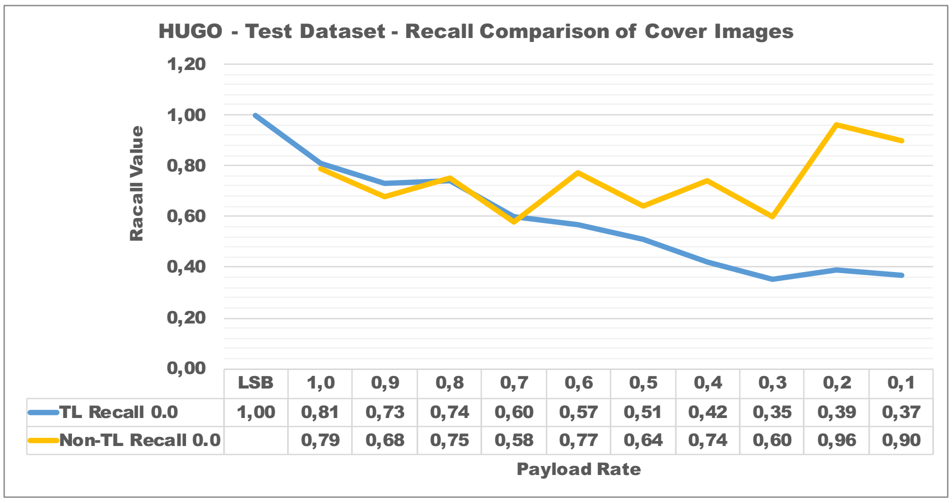
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Precision Average	1,00	0,88	0,88	0,88	0,86	0,84	0,82	0,79	0,75	0,69	0,61
Non-TL Precision Average		0,89	0,86	0,86	0,80	0,77	0,73	0,70	0,52	0,50	0,50

Figure 5.22: WOW Models - Test Dataset Average Precision Comparison

values. It seemed that the model successfully predicted almost all of the cover images out of 40000 cover images. But, the recall comparison of stego images on Figure 5.24 clearly proved that model failed at predicting stego images totaling 40000 stego images. When model failed on predicting the image as a stego image, then it labeled the image as a cover image. Cover recall values kept increasing while stego recall values increase. It proved that the model failed to learn correctly differing cover and stego images. The overall performance of the Non-TL model could be seen on Figure 5.25.

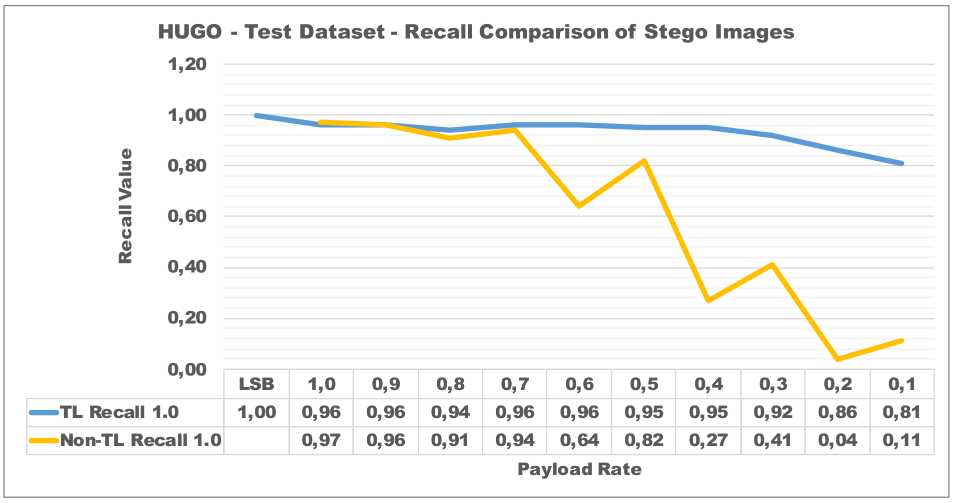
Figure 5.24 and Figure 5.27 presented the recall ratio of stego images out of 40000 images.

Figure 5.25 and Figure 5.28 presented total recall ratio both cover and stego images out of 80000 images.



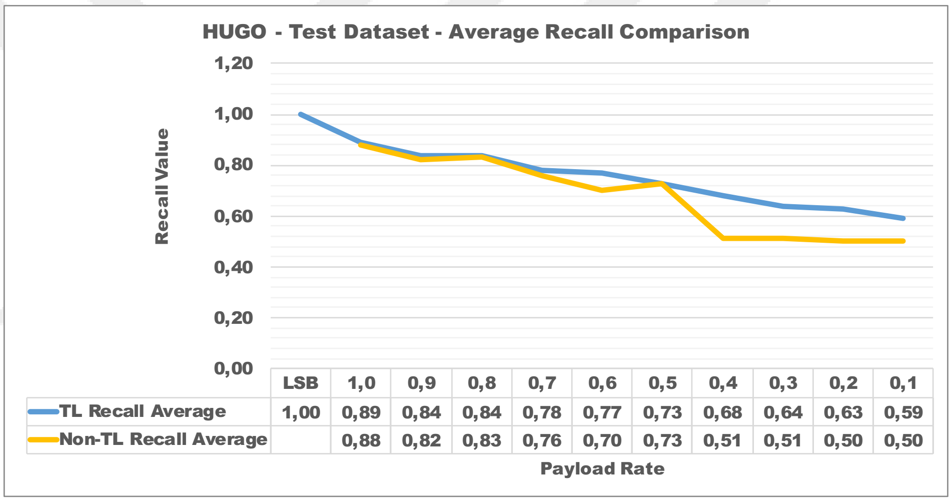
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Recall 0.0	1,00	0,81	0,73	0,74	0,60	0,57	0,51	0,42	0,35	0,39	0,37
Non-TL Recall 0.0		0,79	0,68	0,75	0,58	0,77	0,64	0,74	0,60	0,96	0,90

Figure 5.23: HUGO Models - Test Dataset Recall Comparison of Cover Images



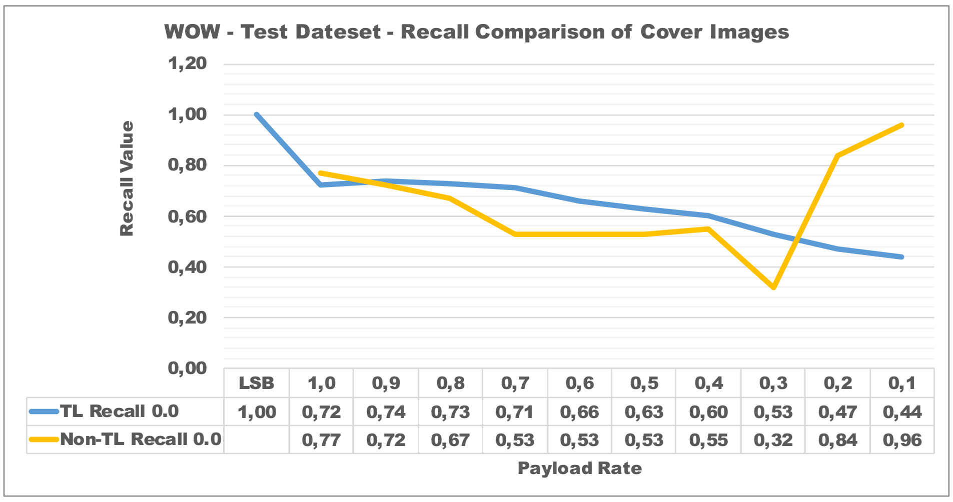
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Recall 1.0	1,00	0,96	0,96	0,94	0,96	0,96	0,95	0,95	0,92	0,86	0,81
Non-TL Recall 1.0		0,97	0,96	0,91	0,94	0,64	0,82	0,27	0,41	0,04	0,11

FIGURE 5.24: HUGO Models - Test Dataset Recall Comparison of Stego Images



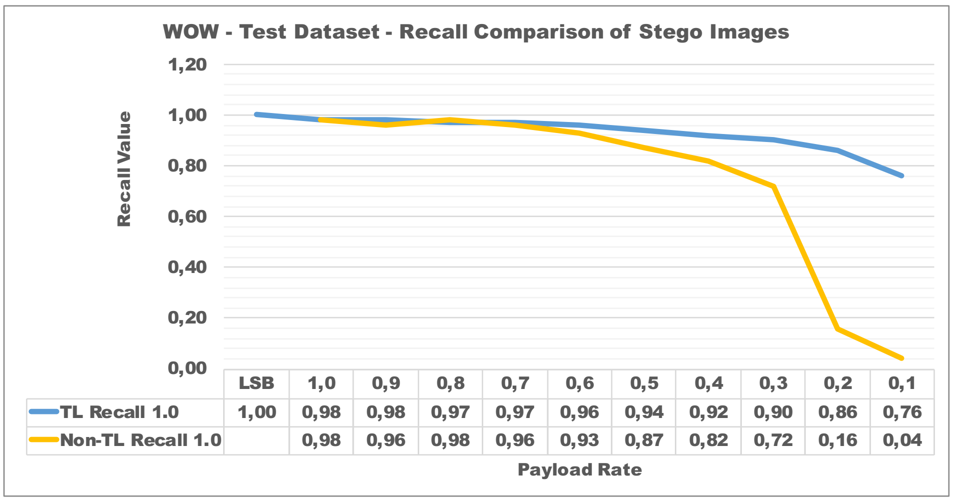
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Recall Average	1,00	0,89	0,84	0,84	0,78	0,77	0,73	0,68	0,64	0,63	0,59
Non-TL Recall Average		0,88	0,82	0,83	0,76	0,70	0,73	0,51	0,51	0,50	0,50

FIGURE 5.25: HUGO Models - Test Dataset Average Recall Comparison



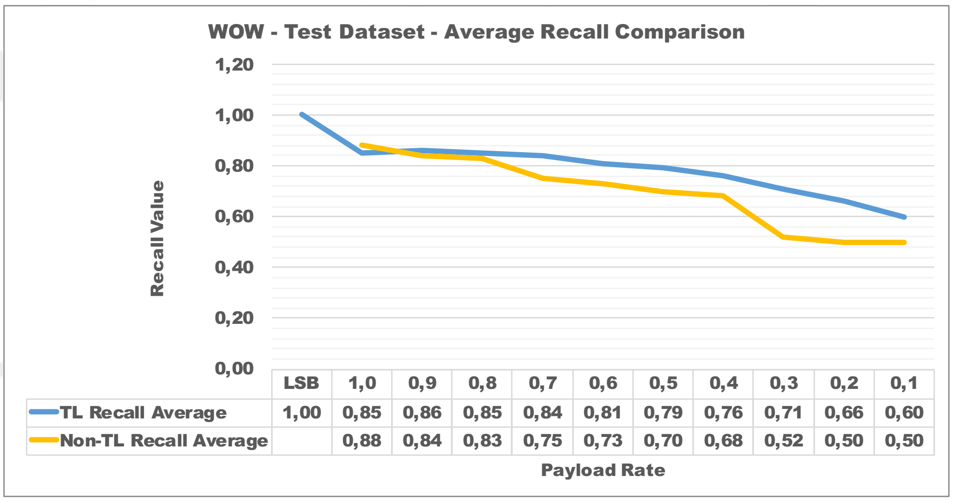
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Recall 0.0	1,00	0,72	0,74	0,73	0,71	0,66	0,63	0,60	0,53	0,47	0,44
Non-TL Recall 0.0		0,77	0,72	0,67	0,53	0,53	0,53	0,55	0,32	0,84	0,96

FIGURE 5.26: WOW Models - Test Dataset Recall Comparison of Cover Images



	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Recall 1.0	1,00	0,98	0,98	0,97	0,97	0,96	0,94	0,92	0,90	0,86	0,76
Non-TL Recall 1.0		0,98	0,96	0,98	0,96	0,93	0,87	0,82	0,72	0,16	0,04

FIGURE 5.27: WOW Models - Test Dataset Recall Comparison of Stego Images



	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL Recall Average	1,00	0,85	0,86	0,85	0,84	0,81	0,79	0,76	0,71	0,66	0,60
Non-TL Recall Average		0,88	0,84	0,83	0,75	0,73	0,70	0,68	0,52	0,50	0,50

FIGURE 5.28: WOW Models - Test Dataset Average Recall Comparison

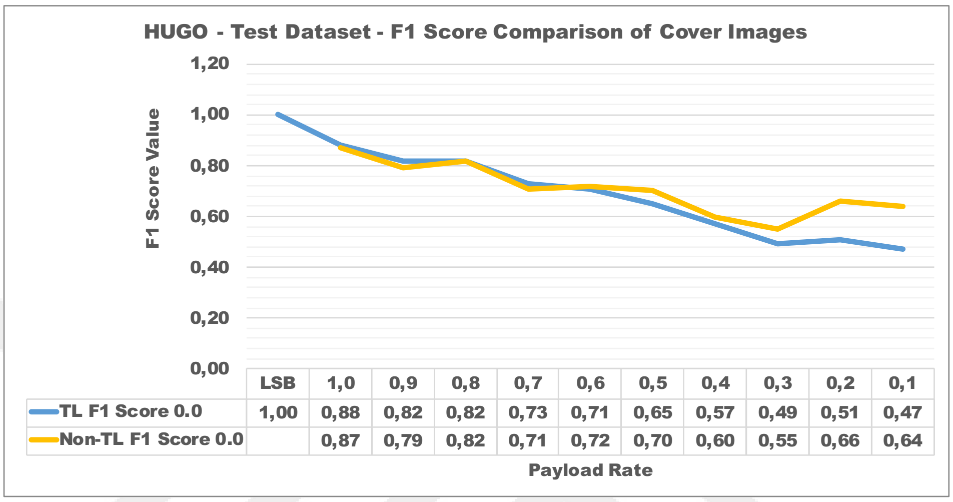
5.3.7 F1-Score Comparisons

F1 Score was the average of Precision ratio and Recall ratio but with a weighted value. It indicated the overall success rates of the model in detecting cover images or stego images.

Figure 5.29 and Figure 5.32 presented the F1 Score of cover images out of 40000 images. F1 Score was the weighted average of Precision and Recall values. Respectively, after 0.7bpp and 0.3bpp datasets, the model predicted cover images with increased values with Non-TL models. The reason was that after those datasets, the model failed to learn stego images and predicted most of the images as cover images. Because the datasets kept getting harder to learn and detect. Also, due to the no applied transfer learning method to models, there was not any background trainings for the model. The models started over to train and learned the dataset and cover-stego image diversity.

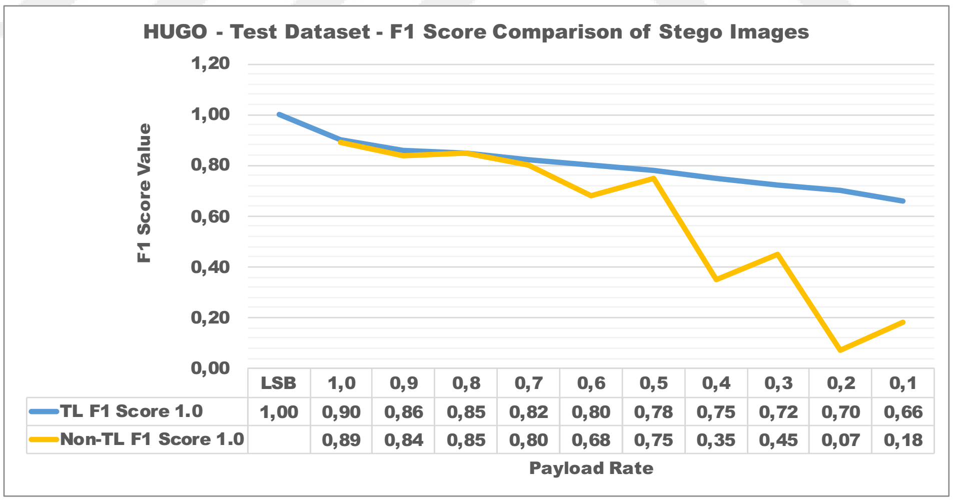
Figure 5.30 and Figure 5.33 presented the F1 Score of stego images out of 40000 images.

Figure 5.31 and Figure 5.34 presented total F1 Score both cover and stego images out of 80000 images.



	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL F1 Score 0.0	1,00	0,88	0,82	0,82	0,73	0,71	0,65	0,57	0,49	0,51	0,47
Non-TL F1 Score 0.0		0,87	0,79	0,82	0,71	0,72	0,70	0,60	0,55	0,66	0,64

Figure 5.29: HUGO Models - Test Dataset F1 Score Comparison of Cover Images

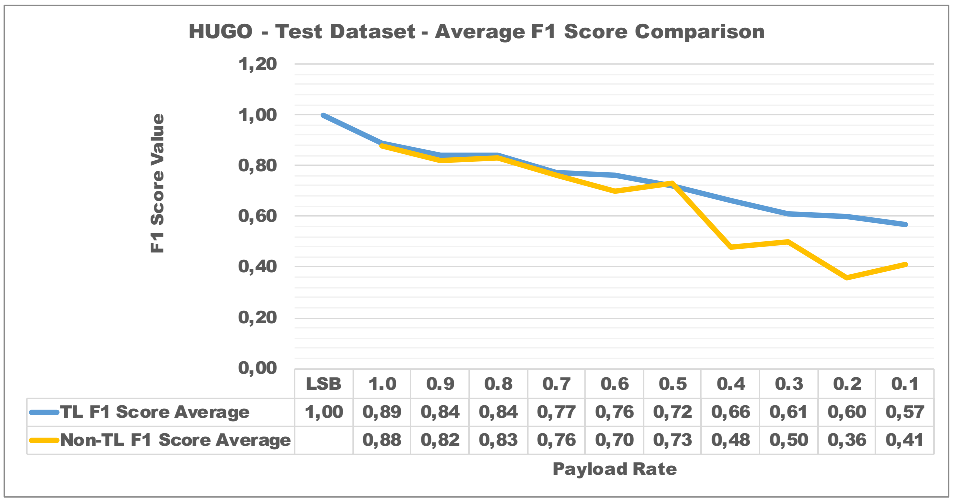


	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL F1 Score 1.0	1,00	0,90	0,86	0,85	0,82	0,80	0,78	0,75	0,72	0,70	0,66
Non-TL F1 Score 1.0		0,89	0,84	0,85	0,80	0,68	0,75	0,35	0,45	0,07	0,18

Figure 5.30: HUGO Models - Test Dataset F1 Score Comparison of Stego Images

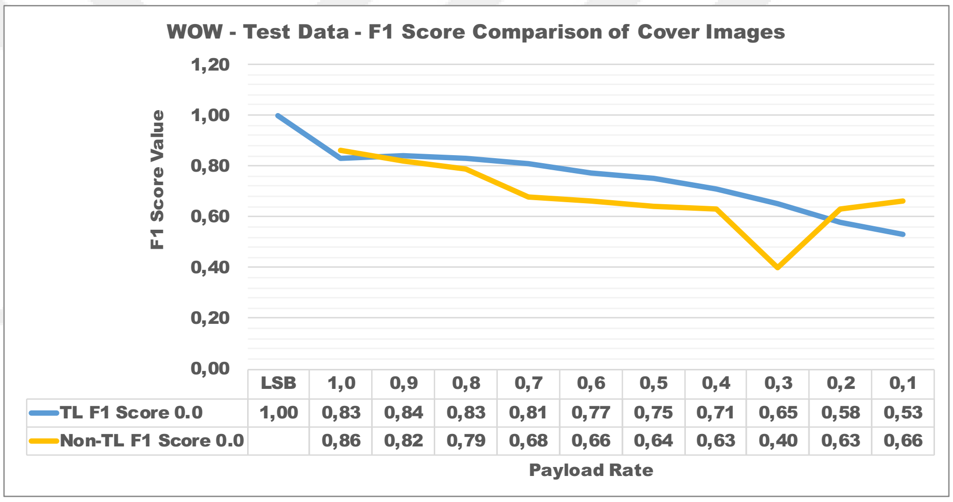
5.3.8 Related Work Comparisons

We compared previous results of studies on Table 5.15 about detection error rate for WOW Steganography algorithm. Results of "Qian et al. [20]" were from no pre-train the model like "Our Non-TL Model". When we compared the related results between



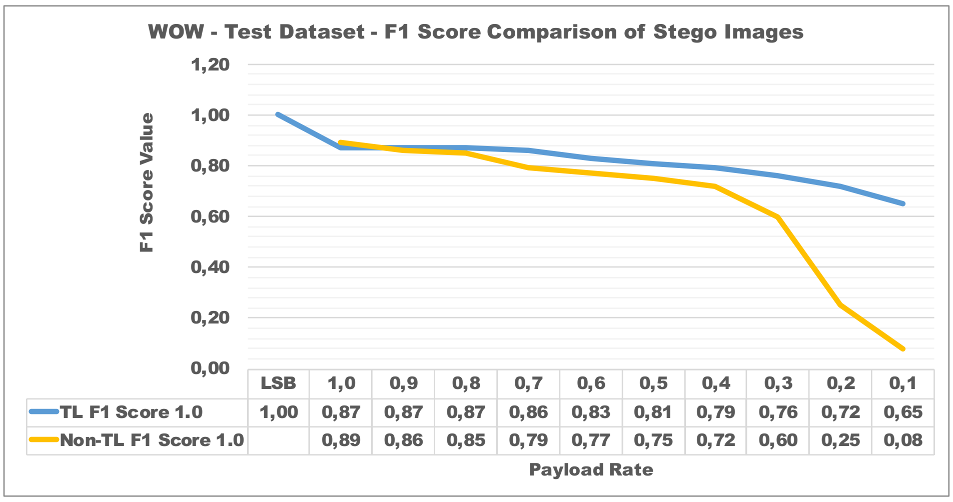
	LSB	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
TL F1 Score Average	1,00	0,89	0,84	0,84	0,77	0,76	0,72	0,66	0,61	0,60	0,57
Non-TL F1 Score Average		0,88	0,82	0,83	0,76	0,70	0,73	0,48	0,50	0,36	0,41

FIGURE 5.31: HUGO Models - Test Dataset Average F1 Score Comparison



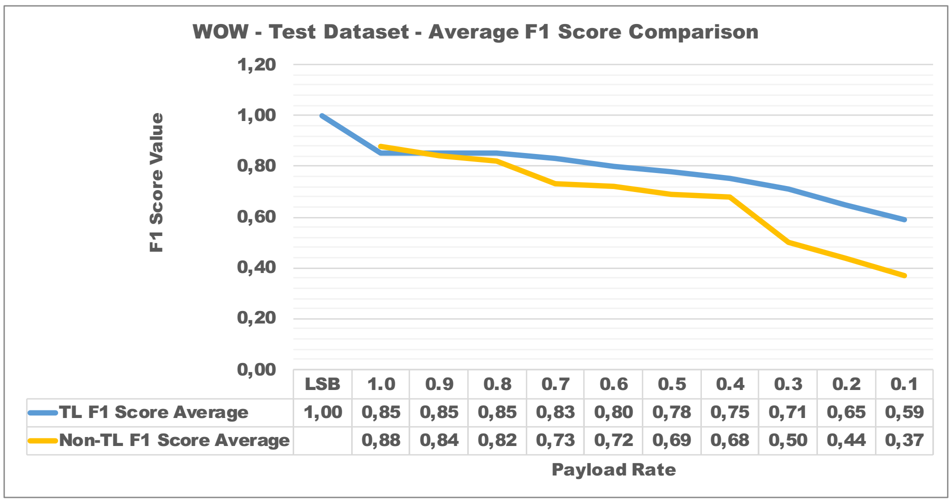
	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL F1 Score 0.0	1,00	0,83	0,84	0,83	0,81	0,77	0,75	0,71	0,65	0,58	0,53
Non-TL F1 Score 0.0		0,86	0,82	0,79	0,68	0,66	0,64	0,63	0,40	0,63	0,66

FIGURE 5.32: WOW Models - Test Dataset F1 Score Comparison of Cover Images



	LSB	1,0	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
TL F1 Score 1.0	1,00	0,87	0,87	0,87	0,86	0,83	0,81	0,79	0,76	0,72	0,65
Non-TL F1 Score 1.0		0,89	0,86	0,85	0,79	0,77	0,75	0,72	0,60	0,25	0,08

FIGURE 5.33: WOW Models - Test Dataset F1 Score Comparison of Stego Images



	LSB	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
TL F1 Score Average	1,00	0,85	0,85	0,85	0,83	0,80	0,78	0,75	0,71	0,65	0,59
Non-TL F1 Score Average		0,88	0,84	0,82	0,73	0,72	0,69	0,68	0,50	0,44	0,37

FIGURE 5.34: WOW Models - Test Dataset Average F1 Score Comparison

0.5bpp and 0.1bpp, we saw that our model had better detection performance on 0.5bpp, 0.4bpp, 0.3bpp and 0.1bpp datasets. "Our TL-model", "Wu [33]" and "Wu et al. [32]" shared similar architecture of deep learning model which was Deep Residual Learning Model. The model had more layers other compared results of the models. "Our TL-model" and "Qian et al. [10]" used similar proposed methodology about using transfer learning method. On the comparison results, we saw that with our "Our TL-model", we obtained lower detection error rates which indicates better performance on detecting stego images. Results of "Wu [33]" and "Wu et al. [32]" were from only 0.4bpp dataset.

TABLE 5.15: Comparisons of Detection Error for WOW Algorithm

Related Work / Payload	**0.5bpp**	**0.4bpp**	**0.3bpp**	**0.2bpp**	**0.1bpp**
Qian et al. [20]	18.50%	20.28%	27.88%	33.30%	50.00%
Our Non-TL Model	17.00%	14.00%	20.00%	34.00%	46.00%
Qian et al. [10]	18.55%	21.95%	24.87%	30.78%	38.43%
Wu [33]		4.3%			
Wu et al. [32]		4.3%			
Our TL-model	15.00%	16.00%	19.00%	20.00%	16.00%

Chapter 6

Conclusion

In this study, we showed that transfer learning applied model is more successful than normal trained model. It was obvious that by applying transfer learning, we obtained more successful results on detecting stego images on every different rated payload dataset. It was a more stabilized model and it had smoothly changed on results. It was more precise on detecting stego images and cover images. It showed that we achieved our main objective which was to increase the successful detection rate on dataset with embedded payload value 0.1bpp. Because the model had been trained previously, the model was familiar with the concept of recognizing images and properties of images. Training with LSB for starters had some positive increases in detecting stego images.

Steganalysis was still a challenging research area. There was not a single method to ensure that medium did not contain secret message. Hence, researchers still develop new methods and tools for the purpose of finding a solution. There is a dilemma in the world of the steganography which is as follows; one can not know for sure if a message is hidden or not. When no secret message was found, it would result in two conclusions: did one try everything or was there no message at all?

For future studies, we plan to run experiments on the other remaining steganography algorithms such as S-UNIWARD, J-UNIWARD and HILL etc. Adjusting the hyper parameters of the model, using various activation functions, training with more epochs and with diverse dataset and adding more layers to the models would be the first steps of our future work to train the model well and to increase the performance of the model. There are many steganography algorithms to solve and to teach to the computer. We think that if it would be possible to train a model that learns steganography very well and it would be able to detect not all and every kind but most of the stego images with different steganography algorithms. The very perfect solution would be that one trained model to be able to figure out stego images generated from old and new steganography algorithms.

Appendix A

WOW Training Validation Results

You can find here about the training process of the models. All the figures from Figure A.1 to Figure A.11 come from the validation step of training process. Validation dataset contain 3815-unit images, just small part of the training dataset. Figures have two sections, section a is from the model with transfer learning method applied and section b is from the normal model with non-applied transfer learning method. It starts from LSB training process, Payload Rate 1.0 to Payload Rate 0.1 training results. Figures have two series, loss and accuracy. Validation Loss is that how much the model away from the correct predictions. It should go to zero as much as possible. Validation Accuracy is that how much the model close to the correct predictions. Also, it should go to one as much as possible.

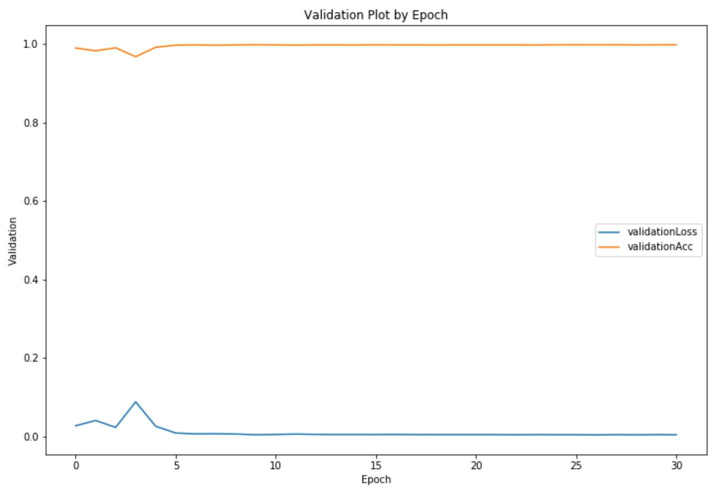
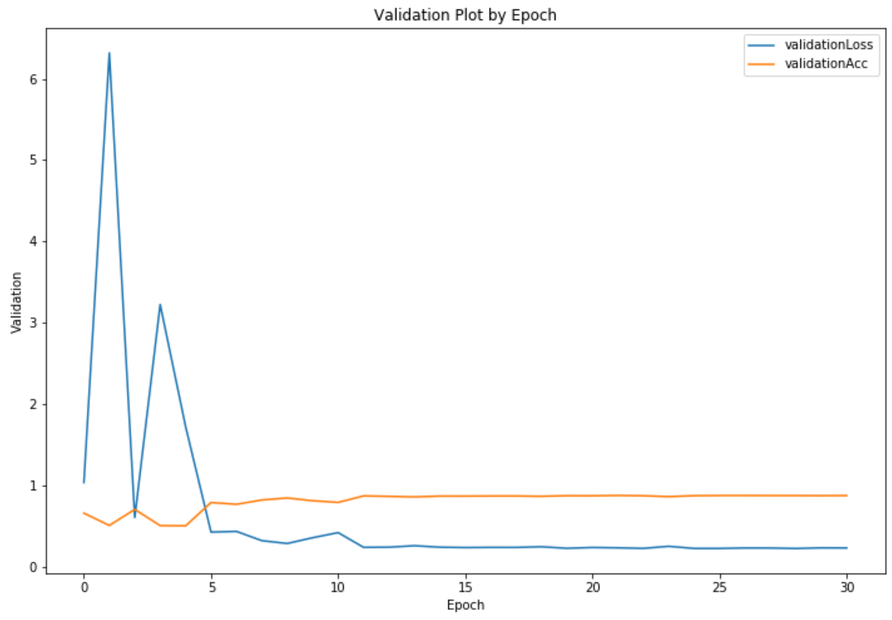
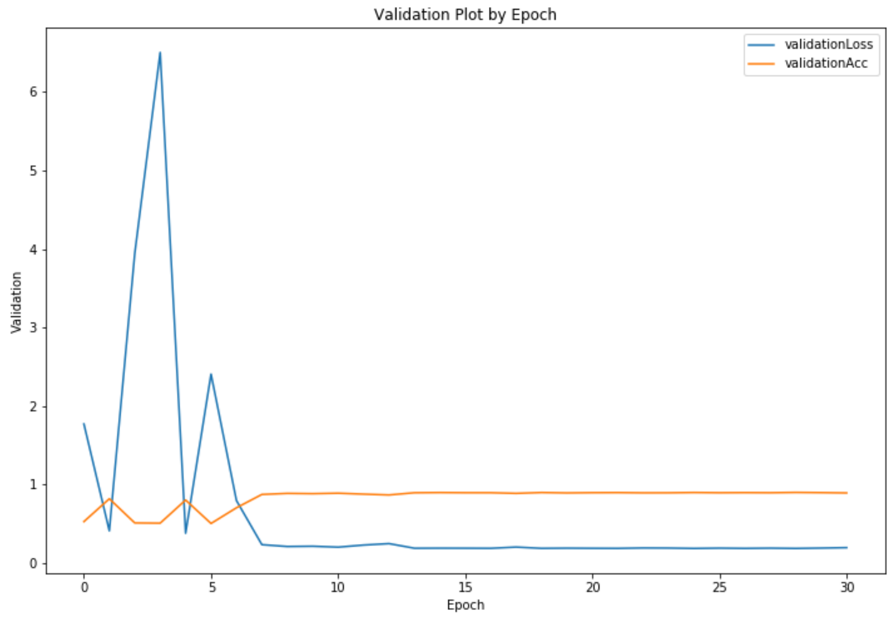


FIGURE A.1: LSB Training Validation Loss and Accuracy Progress

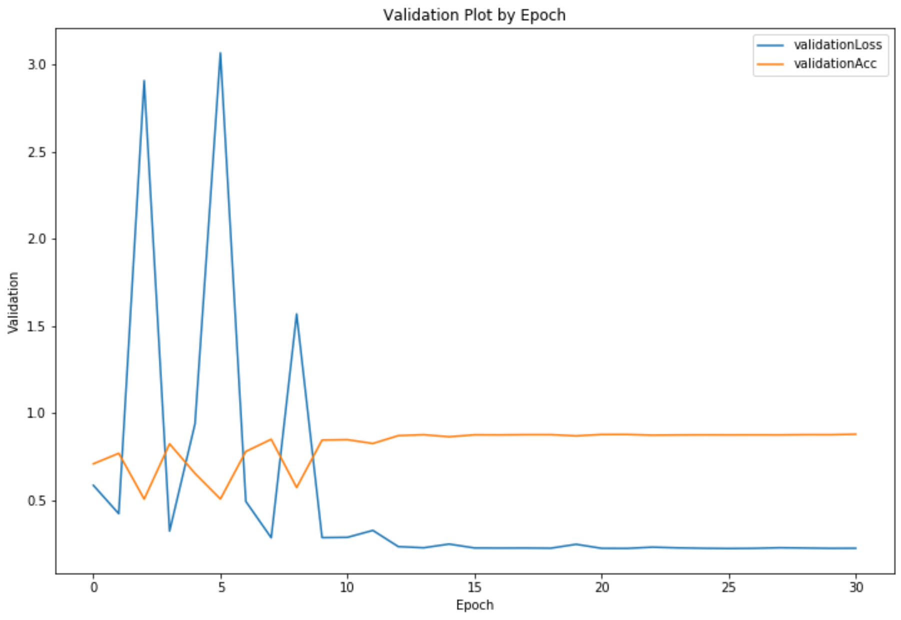


a) Transfer Learning Applied

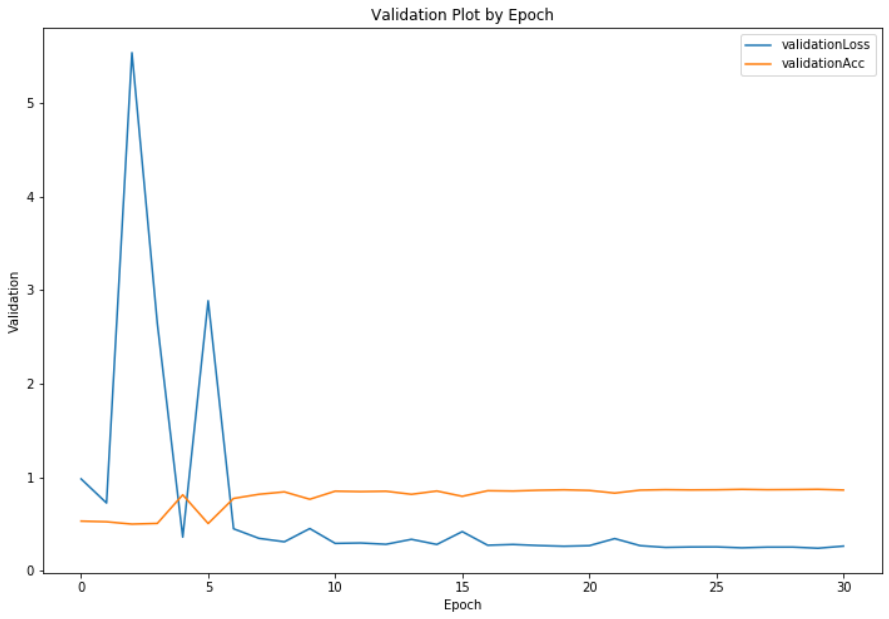


b) Non-Transfer Learning Applied

FIGURE A.2: WOW Training Validation Progress - Payload Rate: 1.0

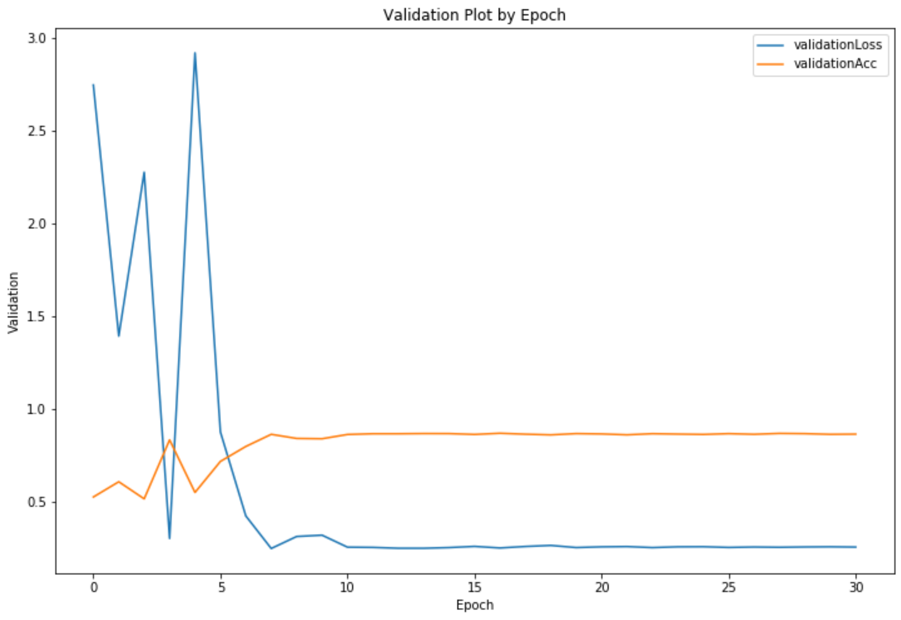


a) Transfer Learning Applied

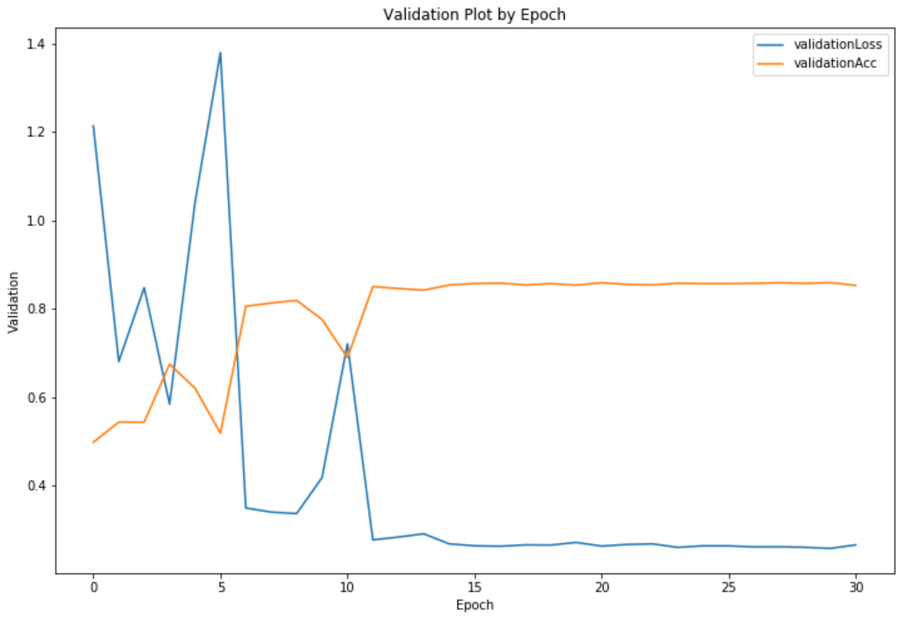


b) Non-Transfer Learning Applied

FIGURE A.3: WOW Training Validation Progress - Payload Rate: 0.9

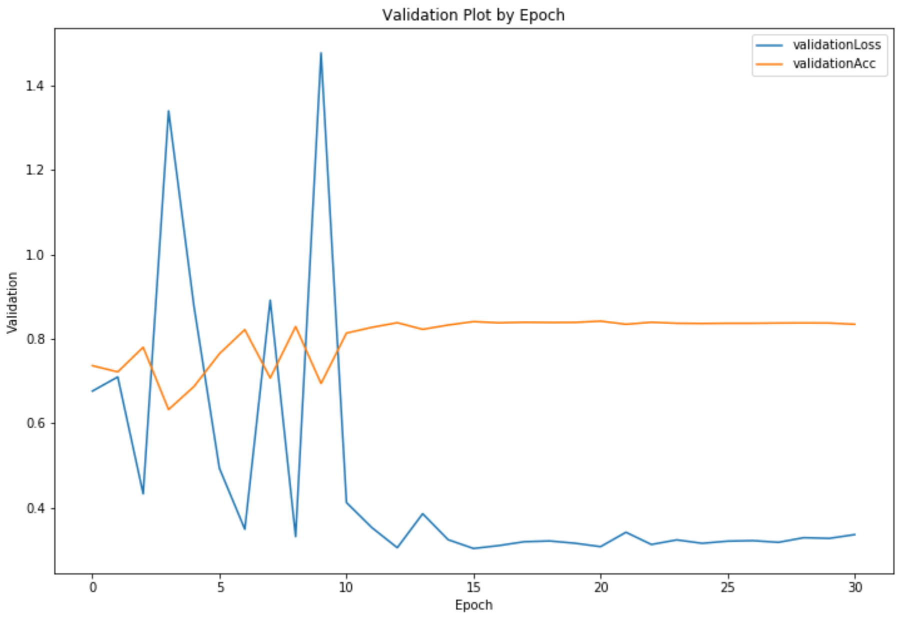


a) Transfer Learning Applied

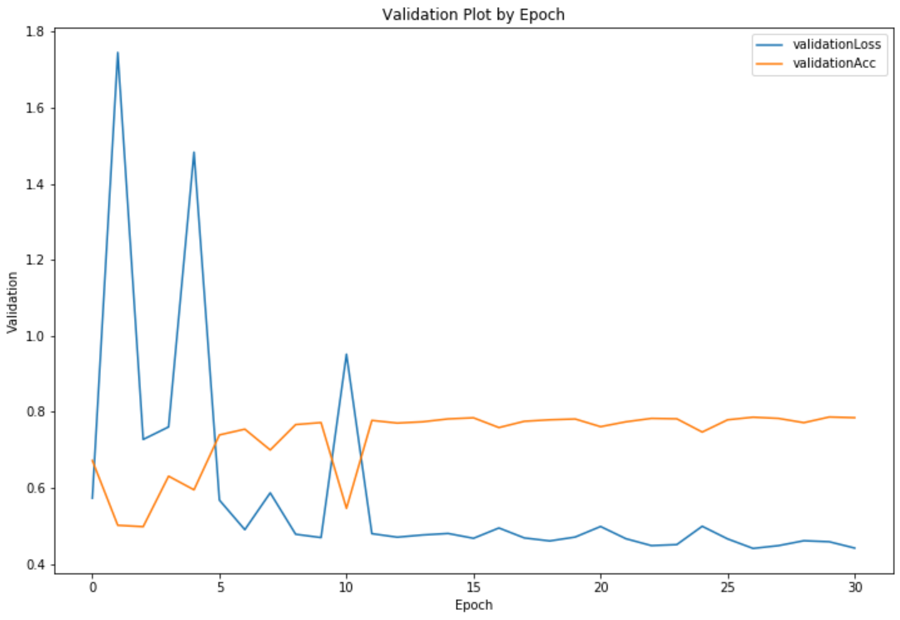


b) Non-Transfer Learning Applied

FIGURE A.4: WOW Training Validation Progress - Payload Rate: 0.8

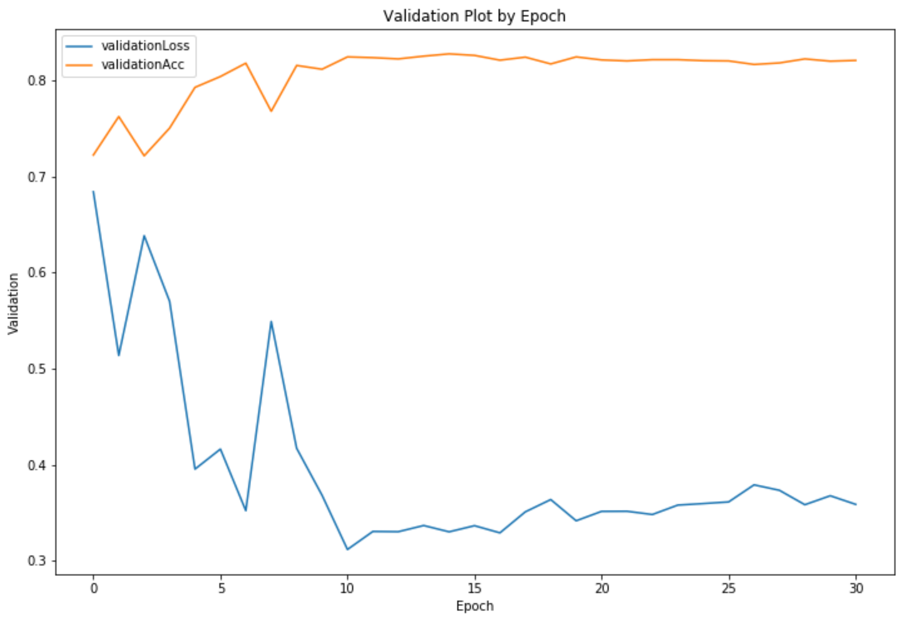


a) Transfer Learning Applied

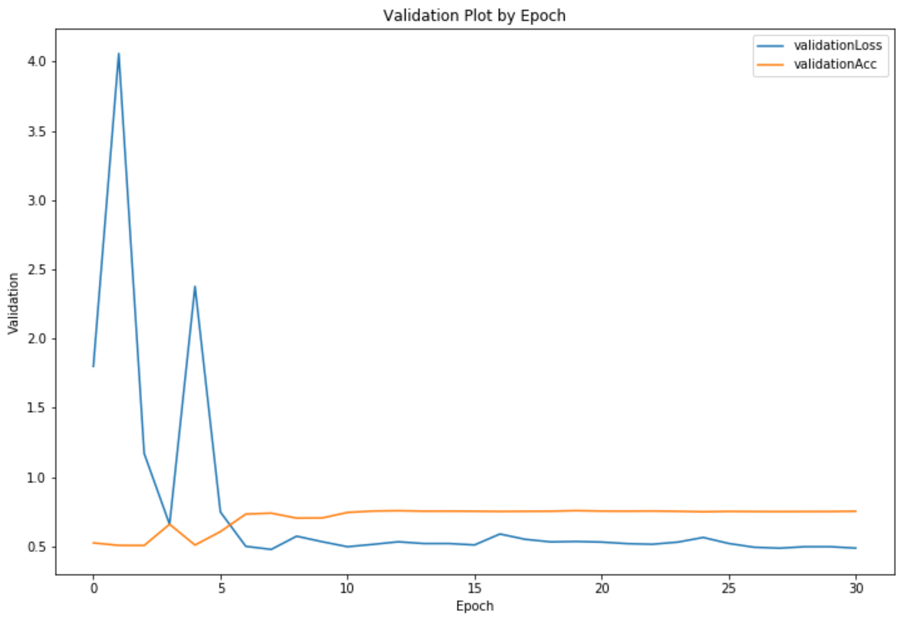


b) Non-Transfer Learning Applied

Figure A.5: WOW Training Validation Progress - Payload Rate: 0.7

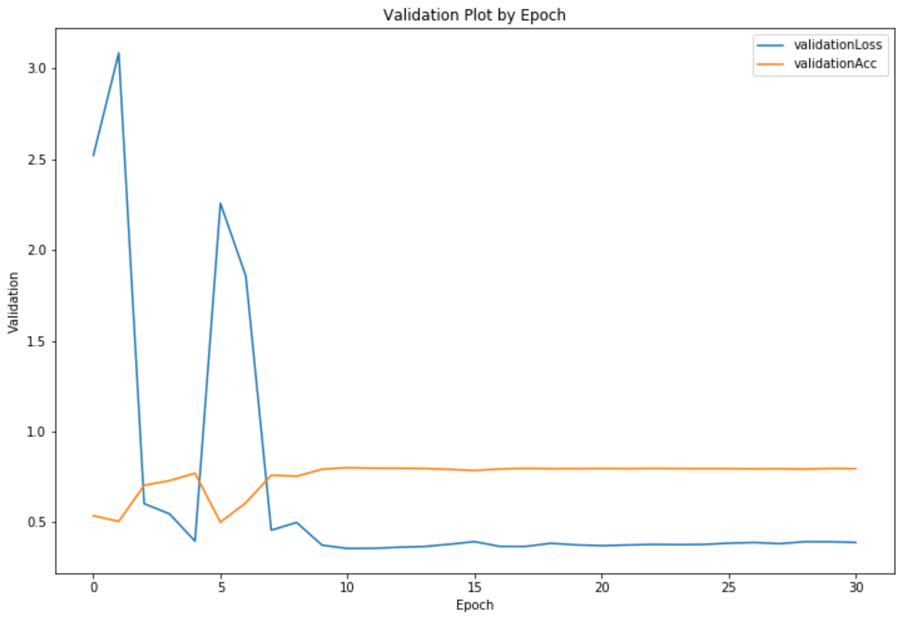


a) Transfer Learning Applied

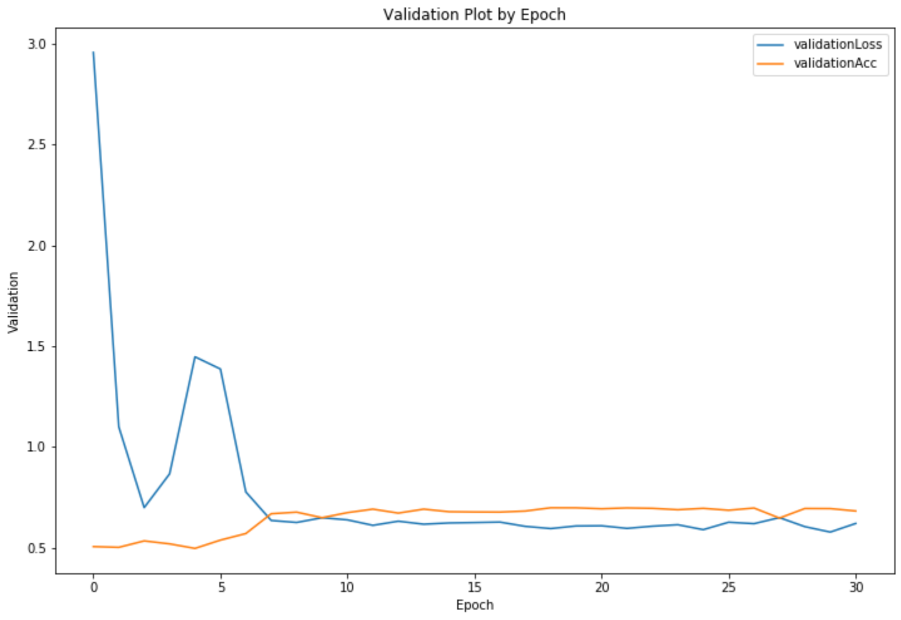


b) Non-Transfer Learning Applied

Figure A.6: WOW Training Validation Progress - Payload Rate: 0.6

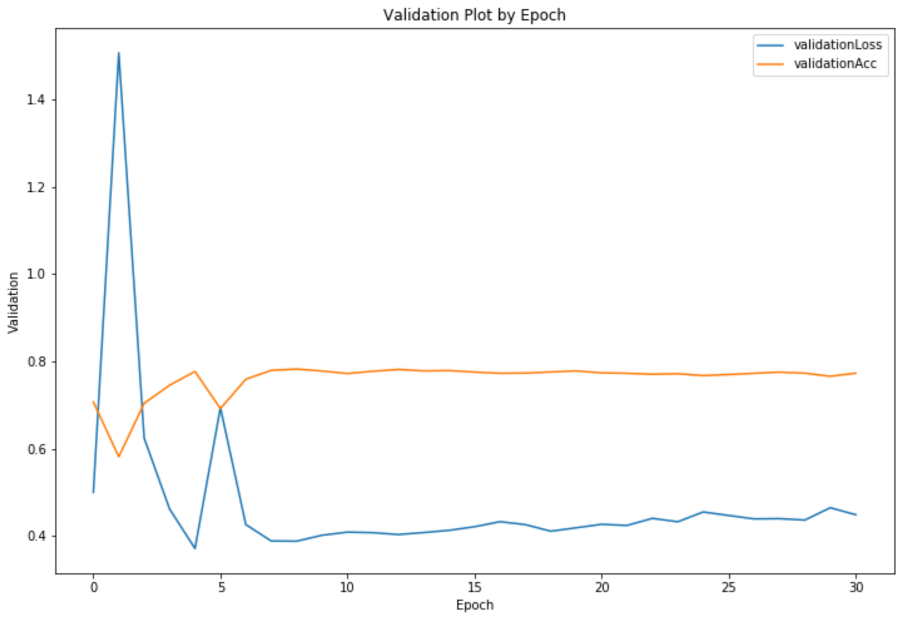


a) Transfer Learning Applied

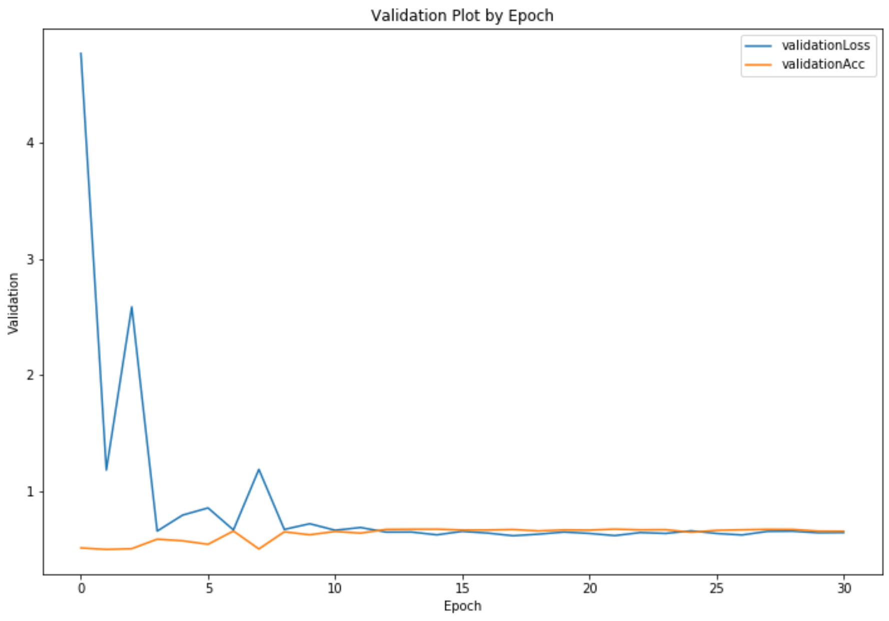


b) Non-Transfer Learning Applied

FIGURE A.7: WOW Training Validation Progress - Payload Rate: 0.5

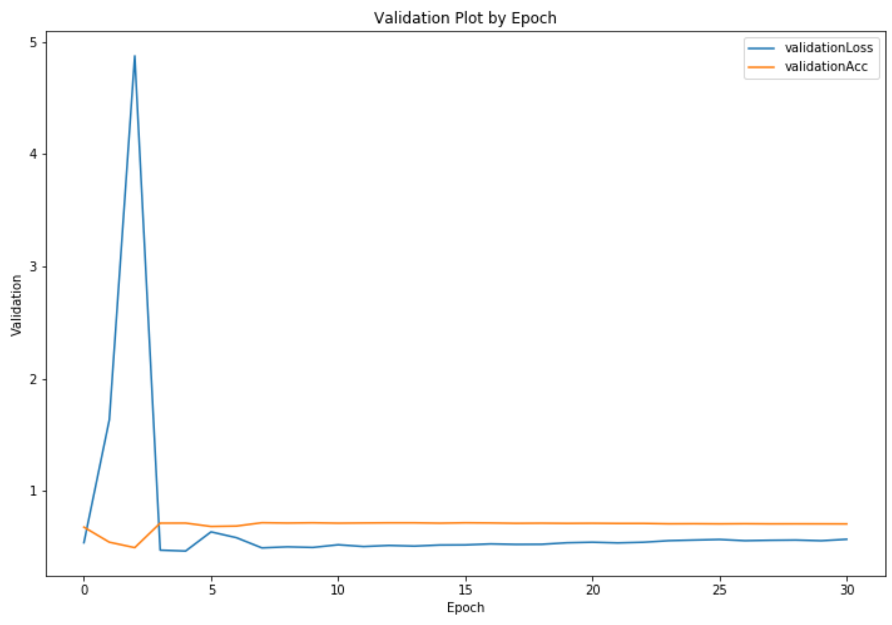


a) Transfer Learning Applied

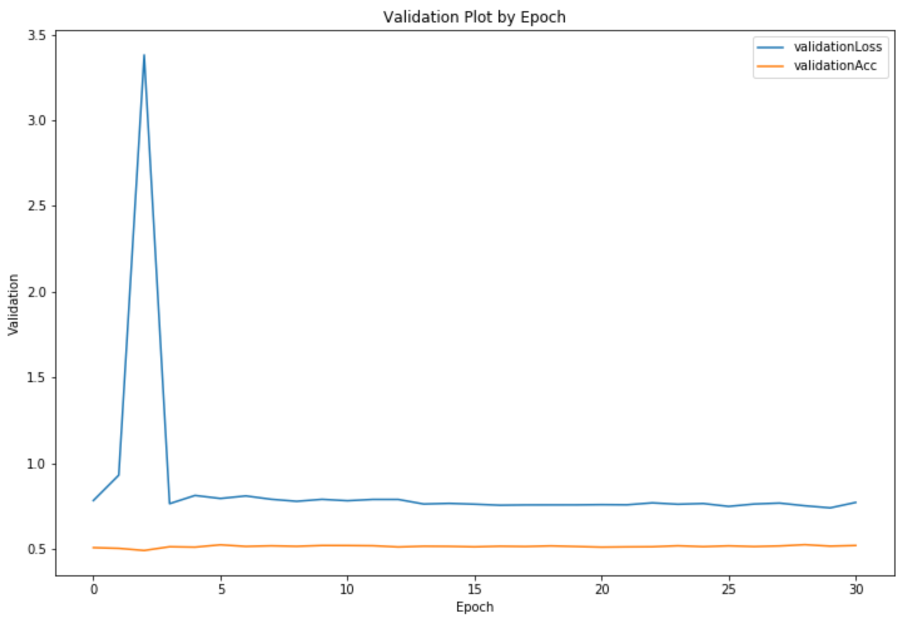


b) Non-Transfer Learning Applied

Figure A.8: WOW Training Validation Progress - Payload Rate: 0.4

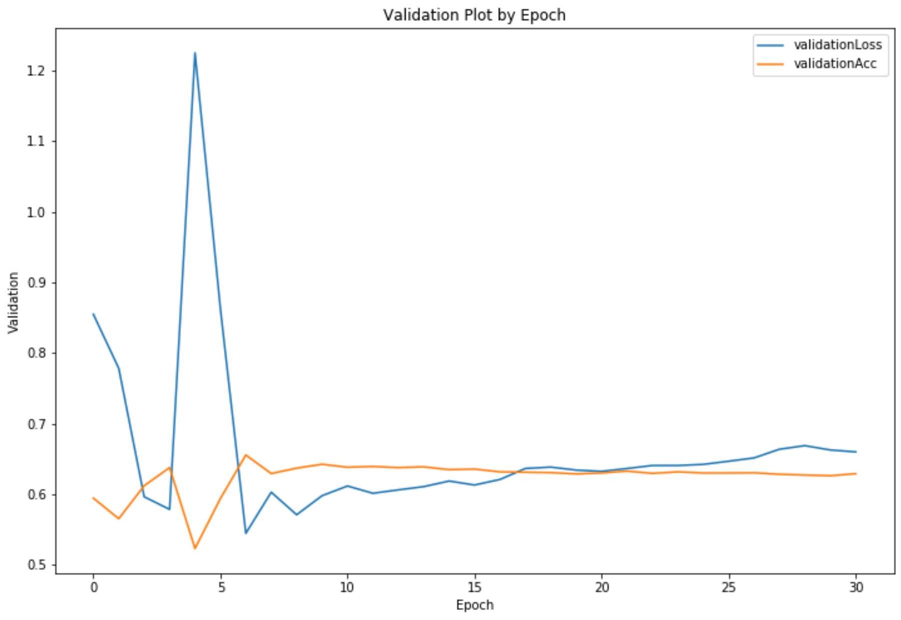


a) Transfer Learning Applied

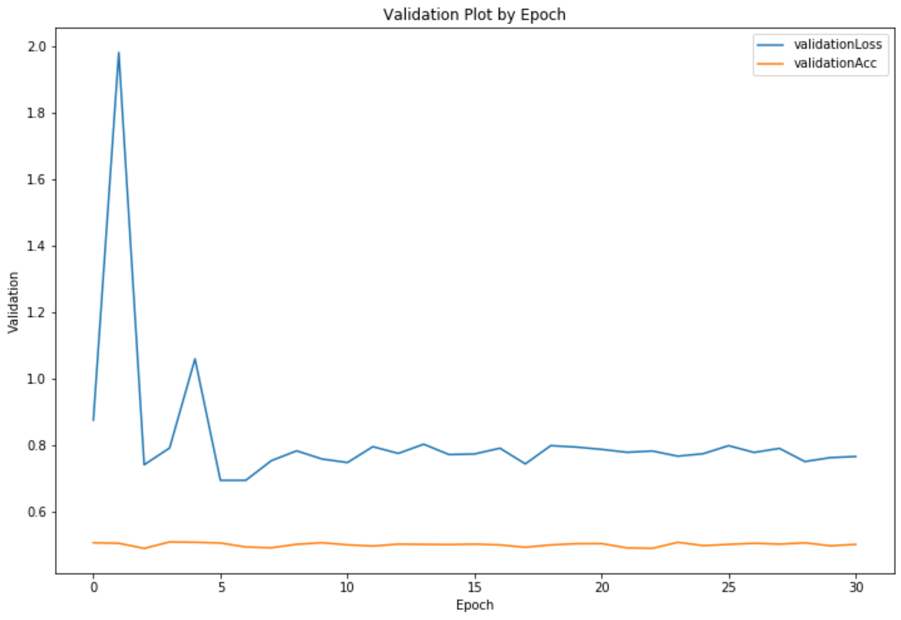


b) Non-Transfer Learning Applied

FIGURE A.9: WOW Training Validation Progress - Payload Rate: 0.3

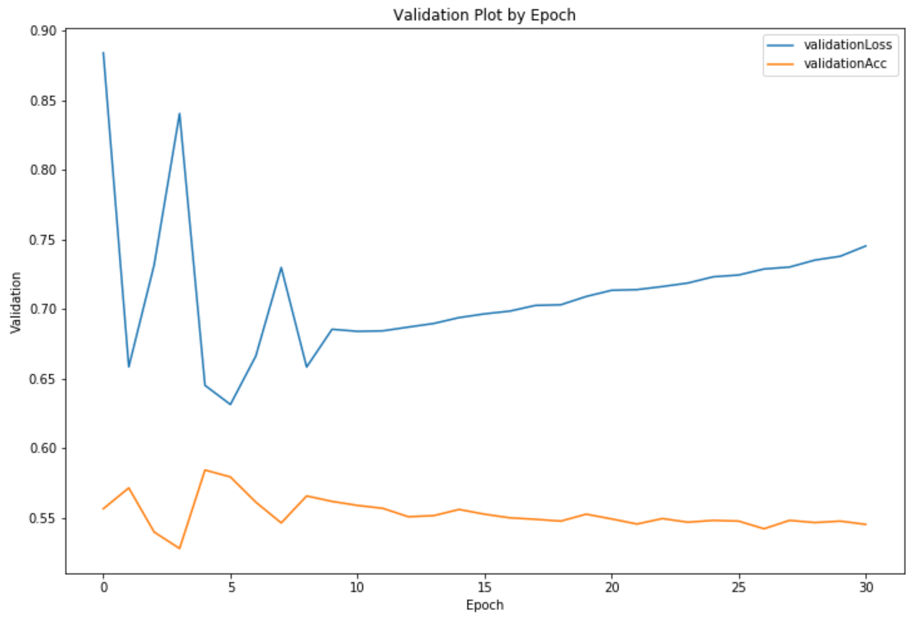


a) Transfer Learning Applied

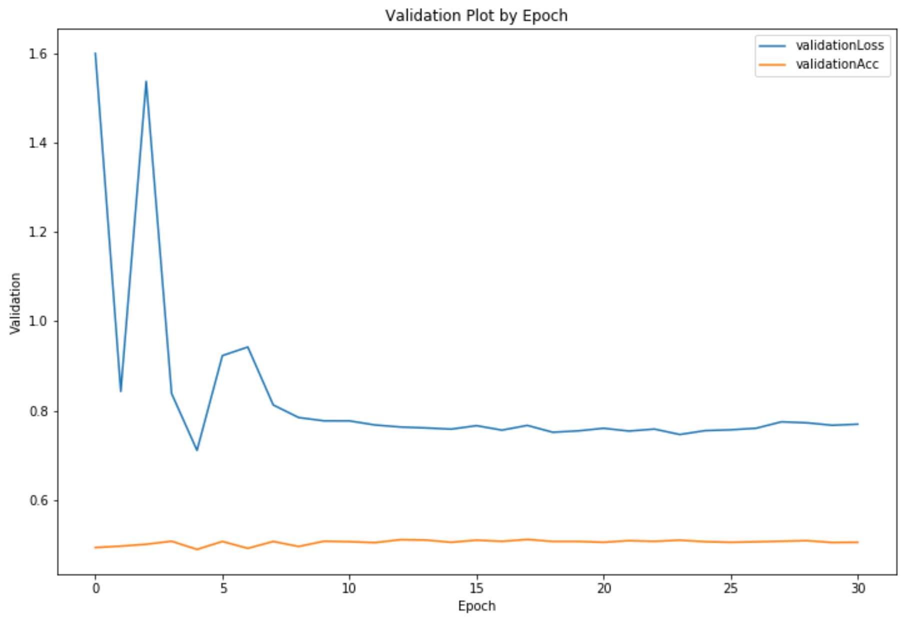


b) Non-Transfer Learning Applied

FIGURE A.10: WOW Training Validation Progress - Payload Rate: 0.2



a) Transfer Learning Applied



b) Non-Transfer Learning Applied

FIGURE A.11: WOW Training Validation Progress - Payload Rate: 0.1

Appendix B

Source Codes

You can find the LSB Training Code Below. It is from IPython Jupyter Notebook application.

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import sys
sys.path.insert(0, '/workspace/data/workspace/AdamW-and-SGDW/')
from AdamW import AdamW



def save(model, name=''):
    def save_model(model, model_path='model.json'):
        model_json = model.to_json()
        with open(model_path, "w") as json_file:
            json_file.write(model_json)
        print('Model {} saved.'.format(model_path))
    def save_weights(model, weights_path='weights.h5'):
        model.save_weights(weights_path)
        print('Weights {} saved.'.format(weights_path))

    save_model(model, name+'_model.json')
    save_weights(model, name+'_weights.h5')

def load(model_name):
    def load_model(path):
        json_file = open(path, 'r')
        loaded_model_json = json_file.read()
        json_file.close()
        print('Model {} loaded.'.format(path))
        return model_from_json(loaded_model_json)

    def load_weights(model, path):
        model.load_weights(path)
        print('Weights {} loaded.'.format(path))

    model = load_model(path=model_name+'_model.json')
    load_weights(model, path=model_name+'_weights.h5')
    return model


# In[2]:


from PIL import Image
import numpy as np
from os import listdir
import cv2, random
from scipy import signal
from sklearn.model_selection import train_test_split
from keras.models import load_model
from keras.preprocessing import image as kip
from keras.applications.resnet50 import preprocess_input, decode_predictions
from keras.callbacks import ModelCheckpoint, CSVLogger, ReduceLROnPlateau
from keras.models import model_from_json


# In[9]:


cover_label = 0
stego_label = 1
cover_path = "/workspace/data/workspace/dataset/cropped_dataset/
    cover_cropped_256x256/"
stego_path = "/workspace/data/workspace/dataset/cropped_dataset/
    stegos_cropped_lsb_algoritmasi/"

cover_path_bb092 = "/workspace/data/workspace/dataset/cropped_dataset/
    cover_cropped_256x256_bb092/"
stego_path_bb092 = "/workspace/data/workspace/dataset/cropped_dataset/
    stegos_cropped_lsb_algoritmasi_bb092/"

objects = listdir(cover_path)
objects_bb092 = listdir(cover_path_bb092)
paths = []
for obj in objects:
    paths.append((cover_path + obj, cover_label))
    paths.append((stego_path + obj, stego_label))

for obj in objects_bb092:
    paths.append((cover_path_bb092 + obj, cover_label))
    paths.append((stego_path_bb092 + obj, stego_label))
random.shuffle(paths)
paths_train, paths_test = train_test_split(paths, test_size=0.025, shuffle=True,
    random_state=31)

number_of_train_examples = len(paths_train)
number_of_test_examples = len(paths_test)

print("Number Of Examples:", len(paths))

X_train = np.zeros((number_of_train_examples, 256, 256, 3), dtype=np.float32)
Y_train = np.zeros((number_of_train_examples, 1), dtype=np.float32)

X_test = np.zeros((number_of_test_examples, 256, 256, 3), dtype=np.float32)
Y_test = np.zeros((number_of_test_examples, 1), dtype=np.float32)


# In[3]:


kernel = np.array([  [ -1,  2,  -2,  2, -1],
                     [  2, -6,   8, -6,  2],
                     [ -2,  8, -12,  8, -2],
                     [  2, -6,   8, -6,  2],
                     [ -1,  2,  -2,  2, -1] ] , dtype=np.float32) / 12


# In[4]:


def image_processer(path):
    img_loaded = kip.load_img(path=path, grayscale=True)
    img_convolved = signal.convolve2d(img_loaded, kernel, 'same')
    img_array = kip.img_to_array(img=img_convolved, data_format='channels_last')
    return img_array


# In[ ]:


for i in range(number_of_train_examples):
    img_path, label = paths_train[i]
    X_train[i,...] = image_processer(img_path)
    Y_train[i,...] = label



# In[ ]:



for i in range(number_of_test_examples):
    img_path, label = paths_test[i]
    X_test[i,...] = image_processer(img_path)
    Y_test[i,...] = label




# In[6]:



kerasResnetModel = load("/workspace/data/workspace/Senaryo1/
    kerasResnet50_makale_weight_notcompiled_adamW")


# In[13]:



batch_size = 64
epochs     = 31
b, B, T    = batch_size, number_of_train_examples, epochs
wd         = 0.005 * (b/B/T)**0.5

kerasResnetModel.compile(loss="binary_crossentropy", optimizer=AdamW(weight_decay
    =wd), metrics=['accuracy'])


# In[8]:



checkpoint = ModelCheckpoint(filepath="adim2_model_makale_adamW.{epoch:02d}-{
    val_acc:.2f}.hdf5", monitor="val_acc", verbose=1, save_best_only=True, mode="
    max")
csv_logger = CSVLogger(filename='adim2_makale_adamW_lsb_training.log', append=
    True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr
    =1e-5)


# In[ ]:
```

```
model_history_lsb = kerasResnetModel.fit(x=X_train, y=Y_train, batch_size=
    batch_size, epochs=epochs, verbose=1, shuffle=True, validation_data=(X_test,
    Y_test), callbacks=[checkpoint, csv_logger,reduce_lr]  )


# In[ ]:


model_history_lsb.history


# In[18]:


kerasResnetModel.save(filepath="
    adim2_modelsave_makale_lsb_kerasresnet50_weight_211f_79789t_compiled_adamw.
    hdf5")
save(kerasResnetModel, "
    adim2_makale_lsb_kerasresnet50_weight_211f_79789t_compiled_adamw" )


#

#

# TESTTING OUTPUT

# In[5]:


cover_label = 0
stego_label = 1
test_cover_path = "/workspace/data/workspace/dataset/test_dataset/
    test_cropped_dataset/"
test_stego_path = "/workspace/data/workspace/dataset/test_dataset/
    stegos_cropped_test_lsb_algoritmasi/"
test_objects = listdir(test_cover_path)
test_paths = []
for test_obj in test_objects:
    test_paths.append((test_cover_path + test_obj, cover_label))
    test_paths.append((test_stego_path + test_obj, stego_label))

random.shuffle(test_paths)
print("Objects:" + str(len(test_objects) ) + " Total number: " + str(len(
    test_paths)))


# In[14]:
```

```
predict_result = []
test_X = np.zeros((1, 256, 256, 3), dtype=np.float32)
for test_path, test_label in test_paths:
    test_X[0, ...] = image_processer(test_path)
    predict_result.append((test_label, kerasResnetModel.predict(x=test_X )))


# In[15]:


result_list = []
for result_label, result_predict in predict_result:
    if(result_predict > 0.5):
        result_predict_value = 1
    else:
        result_predict_value = 0
    if(result_label == result_predict_value):
        result_list.append((result_label, result_predict, result_predict_value,
    True))
    else:
        result_list.append((result_label, result_predict, result_predict_value,
    False))


# In[16]:


false_list = [i for i in result_list if i[3] == False]
true_list = [i for i in result_list if i[3] == True]


# In[17]:


print ( len(false_list) )
print (len(true_list))


# In[20]:


test_X = np.zeros((1, 256, 256, 3), dtype=np.float32)
test_X[0,...] = image_processer("/workspace/data/workspace/dataset/test_dataset/
    stegos_cropped_test_lsb_algoritmasi/123.pgm_(0, 0, 256, 256).pgm")
kerasResnetModel.predict(x=test_X)
```

```
# In[ ]:


result_list


#
#

# testing evaluate & predict

#

# In[6]:


kerasResnetModel = load("./
    adim2_makale_lsb_kerasresnet50_weight_211f_79789t_compiled_adamw")
batch_size = 64
epochs     = 31
b, B, T    = batch_size, 148777, epochs
wd         = 0.005 * (b/B/T)**0.5

kerasResnetModel.compile(loss="binary_crossentropy", optimizer=AdamW(weight_decay
    =wd), metrics=['accuracy'])


# In[7]:


test_object_size = len(test_paths)
test_X_array = np.zeros((test_object_size, 256, 256, 3), dtype=np.float32)
test_Y_array = np.zeros((test_object_size, 1), dtype=np.float32)
for i in range(test_object_size):
    test_path, test_label = test_paths[i]
    test_X_array[i, ...] = image_processer(test_path)
    test_Y_array[i, ...] = test_label


# In[8]:


model_evaluate = kerasResnetModel.evaluate(x=test_X_array, y=test_Y_array,
    verbose=1)


# In[9]:
```

```
kerasResnetModel.metrics_names


# In[10]:


model_evaluate


# In[11]:


y_pred = kerasResnetModel.predict(x=test_X_array, verbose=1)
y_pred_reshaped = np.array(list(map(lambda x: [0] if x < 0.5 else [1], y_pred)))
from sklearn.metrics import classification_report
report = classification_report(test_Y_array, y_pred_reshaped).split("\n")
report
# https://upload.wikimedia.org/wikipedia/commons/thumb/2/26/Precisionrecall.svg
    /350px-Precisionrecall.svg.png


# In[ ]:






#

#

# GRAPHIC

#

# In[20]:


import matplotlib.pyplot as plt


# In[21]:


plt.figure(figsize=(12,8))
plt.plot(model_history_lsb.history['val_loss'], label='validationLoss')
plt.plot(model_history_lsb.history['val_acc'], label='validationAcc')
plt.title('Validation Plot by Epoch')
plt.xlabel('Epoch')
plt.ylabel('Validation')
plt.legend()
plt.savefig(fname="plot_val.png")
#plt.show()


# In[22]:


plt.figure(figsize=(12,8))
plt.plot(model_history_lsb.history['val_loss'], label='validationLoss')
#plt.plot(model_history_hugo08.history['val_acc'], label='validationAcc')
plt.title('Validation Plot by Epoch')
plt.xlabel('Epoch')
plt.ylabel('Validation')
plt.legend()
plt.savefig(fname="plot_val_loss.png")
#plt.show()


# In[23]:


plt.figure(figsize=(12,8))
#plt.plot(model_history_hugo08.history['val_loss'], label='validationLoss')
plt.plot(model_history_lsb.history['val_acc'], label='validationAcc')
plt.title('Validation Plot by Epoch')
plt.xlabel('Epoch')
plt.ylabel('Validation')
plt.legend()
plt.savefig(fname="plot_val_acc.png")
#plt.show()
```

Bibliography

[1] S. Singh and V. Kaur Attri. State-of-the-art review on steganographic techniques. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8:161–170, 07 2015. doi: 10.14257/ijsip.2015.8.7.15. URL `https://www.researchgate.net/figure/Taxonomy-of-Security-System-2_fig1_286479865`. Scientific Figure on ResearchGate. Accessed: 10.04.2019.

[2] A. C. Brainos II. A study of steganography and the art of hiding information. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.86.284`. Accessed: 18.02.2019.

[3] F. Djebbar, B. Ayad, K. abed meraim, and H. Hamam. Comparative study of digital audio steganography techniques. *EURASIP Journal on Audio, Speech, and Music Processing*, 2012, 10 2012. doi: 10.1186/1687-4722-2012-25. URL `https://www.researchgate.net/figure/Digital-data-security-disciplines_fig12_257879537`. Scientific Figure on ResearchGate. Accessed: 10.04.2019.

[4] J. Trithemius. Steganographia. 1608. URL `http://diglib.hab.de/drucke/fb-243/start.htm?image=00005`. Cover Page of Steganographia. Accessed: 10.04.2019.

[5] D. Kahn. The history of steganography. In Ross Anderson, editor, *Information Hiding*, pages 1–5, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. ISBN 978-3-540-49589-5.

[6] A. Siper, R. Farley, and C. Lombardo. The rise of steganography. *Proceedings of Student/Faculty Research Day, CSIS, Pace University*, May 6th, 2005.

[7] N. F. Johnson and S. Jajodia. Exploring steganography: Seeing the unseen. *Computer*, 31(2):26–34, Feb 1998. ISSN 0018-9162. doi: 10.1109/MC.1998.4655281. URL

http://www.jjtc.com/stegdoc/sec202.html. Steganography - SEC202. Accessed: 13.04.2019.

[8] T. Morkel, J. H. P. Eloff, and M. S. Olivier. An Overview of Image Steganography. *Information and Computer Security Architecture (ICSA) Research Group*, 83(July): 51–107, 2005. ISSN 00652458. doi: 10.1016/B978-0-12-385510-7.00002-3. URL http://martinolivier.com/open/stegoverview.pdf.

[9] J. L. Bamber. CCRS - Fundamentals of Remote Sensing. URL https://seis.bristol.ac.uk/~ggjlb/teaching/ccrs_tutorial/tutorial/chap1/c1p7_i2e.html. Teaching of Pixels. School of Geographical Sciences University of Bristol. Accessed: 13.04.2019.

[10] Y. Qian, J. Dong, W. Wang, and T. Tan. Learning and transferring representations for image steganalysis using convolutional neural network. In *Proceedings - International Conference on Image Processing, ICIP*, volume 2016-August, pages 2752–2756, 2016. ISBN 9781467399616. doi: 10.1109/ICIP.2016.7532860.

[11] J. Fridrich and J. Kodovsky. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3):868–882, 2012. ISSN 15566013. doi: 10.1109/TIFS.2012.2190402.

[12] J. Kodovsky, J. Fridrich, and V. Holub. Ensemble classifiers for steganalysis of digital media. In *IEEE Transactions on Information Forensics and Security*, volume 7, pages 432–444, 2012. ISBN 1556-6013 VO - 7. doi: 10.1109/TIFS.2011.2175919.

[13] S. Ronaghan. Deep Learning: Overview of Neurons and Activation Functions. *Medium.com*, Jul 2018. URL https://medium.com/@srnghn/deep-learning-overview-of-neurons-and-activation-functions-1d98286cf1e4. Figure of Deep Learning Model. Accessed: 13.04.2019.

[14] DATAI. Deep Learning Tutorial for Beginners. *kaggle.com/kanncaa1*, Sep 2018. URL https://www.kaggle.com/kanncaa1/deep-learning-tutorial-for-beginners. Figure of A Neuron of a Deep Learning Model. Accessed: 13.04.2019.

[15] Guru99. Deep Learning Tutorial for Beginners: Neural Network Classification. *guru99.com*. URL https://www.guru99.com/deep-learning-tutorial.html. Figure of Convolutional Neural Networks. Accessed: 13.04.2019.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016. URL `http://arxiv.org/abs/1603.05027`.

[18] H. Long. A Review of ResNet - Residual Networks. *linkinpark213.com/*, Apr 2018. URL `https://linkinpark213.com/2018/04/22/resnet/`. Figure of a residual block of Residual Networks.. Accessed: 13.04.2019.

[19] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014. ISBN 9781479951178. doi: 10.1109/CVPR.2014.222.

[20] Y. Qian, J. Dong, W. Wang, and T. Tan. Deep learning for steganalysis via convolutional neural networks. *Proceedings of SPIE - The International Society for Optical Engineering*, 9409, 03 2015. ISSN 1996756X. doi: 10.1117/12.2083479. URL `https://doi.org/10.1117/12.2083479`.

[21] Y. Qian, J. Dong, W. Wang, and T. Tan. Feature learning for steganalysis using convolutional neural networks. *Multimedia Tools and Applications*, 77(15):19633–19657, Aug 2018. ISSN 1573-7721. doi: 10.1007/s11042-017-5326-1. URL `https://doi.org/10.1007/s11042-017-5326-1`.

[22] M. Salomon, R. Couturier, C. Guyeux, J.-F. Couchot, and J.M. Bahi. Steganalysis via a convolutional neural network using large convolution filters for embedding process with same stego key: A deep learning approach for telemedicine. *European Research in Telemedicine / La Recherche Européenne en Télémédecine*, 6(2):79 – 92, 2017. ISSN 2212-764X. doi: https://doi.org/10.1016/j.eurtel.2017.06.001. URL `http://www.sciencedirect.com/science/article/pii/S2212764X17300614`.

[23] M. Sharifzadeh, C. Agarwal, M. Aloraini, and D. Schonfeld. Convolutional neural network steganalysis's application to steganography. In *2017 IEEE Visual Communications and Image Processing, VCIP 2017*, 2018. ISBN 9781538604625. doi: 10.1109/VCIP.2017.8305045.

[24] K. Liu, J. Yang, and X. Kang. Ensemble of cnn and rich model for steganalysis. In *2017 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 1–5, May 2017. doi: 10.1109/IWSSIP.2017.7965617.

[25] J. Ye, J. Ni, and Y. Yi. Deep Learning Hierarchical Representations for Image Steganalysis. *IEEE Transactions on Information Forensics and Security*, 2017. ISSN 15566013. doi: 10.1109/TIFS.2017.2710946.

[26] S. Wu, S.-H. Zhong, and Y. Liu. Residual convolution network based steganalysis with adaptive content suppression. In *Proceedings - IEEE International Conference on Multimedia and Expo*, 2017. ISBN 9781509060672. doi: 10.1109/ICME.2017.8019304.

[27] G. Xu. Deep Convolutional Neural Network to Detect J-UNIWARD. *CoRR*, abs/1704.08378, 2017. doi: 10.1109/APSIPA.2014.7041565. URL `http://arxiv.org/abs/1704.08378`.

[28] G. Xu, H.-Z. Wu, and Y.-Q. Shi. Structural Design of Convolutional Neural Networks for Steganalysis. *IEEE Signal Processing Letters*, 23(5):708–712, May 2016. ISSN 1070-9908. doi: 10.1109/LSP.2016.2548421. URL `http://www.ieee.org/publications{_}standards/publications/rights/index.html`.

[29] J. Yang, Y.-Q. Shi, E. K. Wong, and X. Kang. JPEG steganalysis based on densenet. *CoRR*, abs/1711.09335, 2017. URL `http://arxiv.org/abs/1711.09335`.

[30] S. Wu, S.-H. Zhong, and Y. Liu. A Novel Convolutional Neural Network for Image Steganalysis with Shared Normalization. *CoRR*, abs/1711.07306, 2017. URL `http://arxiv.org/abs/1711.07306`.

[31] L. Pibre, P. Jérôme, D. Ienco, and M. Chaumont. Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover source-mismatch. *CoRR*, abs/1511.04855:14–18, 2015. ISSN 2470-1173. doi: 10.2352/ISSN.2470-1173.2016.8.MWSF-078. URL `http://arxiv.org/abs/1511.04855`.

[32] S. Wu, S.-H. Zhong, and Y. Liu. Steganalysis via deep residual network. In *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, pages 1233–1236, 2017. ISBN 9781509044573. doi: 10.1109/ICPADS.2016.0167.

[33] S. Wu, S. Zhong, and Y. Liu. Deep residual learning for image steganalysis. *Multimedia Tools and Applications*, 77(9):10437–10453, May 2018. ISSN 1573-7721. doi: 10.1007/s11042-017-4440-4. URL `https://doi.org/10.1007/s11042-017-4440-4`.

[34] BOWS2. BOWS2 Dataset. *bows2.ec-lille.fr/*, Jul 2007 - Apr 2008. URL `http://bows2.ec-lille.fr/BOWS2OrigEp3.tgz`. The 2nd Break Our Watermarking System(BOWS) Contest. Accessed: 18.10.2018.

[35] D. Lerch. daniellerch's Aletheia. *github.com/daniellerch*, 2019. URL `https://github.com/daniellerch/aletheia`. Open source image steganalysis tool. Accessed: 18.02.2019.

[36] TUBITAK. Cloud Computing and Big Data Research Laboratory (B3LAB). 2019. URL `https://www.b3lab.org`. Accessed: 18.02.2019.

[37] S. Ozcan. Results of the study. 2019. URL `https://github.com/oselim/steganalysis-result`. Accessed: 18.02.2019.