

A Project Report on

Using AI for Designing ID Cards Embedded with Invisible QR Code

Submitted in partial fulfillment of the requirements for the award
of the degree of

Bachelor of Engineering

in

Information Technology

by

Dhruva Mhatre 18104045

Chirag Jain 18104002

Prem Vispute 18104059

Under the Guidance of

Prof. Kiran Deshpande

Prof. Kaushiki Upadhyaya



Department of Information Technology

NBA Accredited

A.P. Shah Institute of Technology

G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615

UNIVERSITY OF MUMBAI

Academic Year 2021-2022

Approval Sheet

This Project Report entitled “*Using AI for Designing ID Cards Embedded with Invisible QR Code*” Submitted by “*Dhruva Mhatre (18104045), Chirag Jain (18104002), Prem Vispute (18104059)*” is approved for the partial fulfillment of the requirement for the award of the degree of *Bachelor of Engineering* in *Information Technology* from *University of Mumbai*.

Prof. Kiran Deshpande
Guide
Head, Department of Information Technology

Prof. Kaushiki Upadhyaya
Co-Guide

Dr. Uttam D.Kolekar
Principal

Place: A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project report entitled “*Using AI for Designing ID Cards Embedded with Invisible QR Code*” Submitted by “*Dhruva Mhatre (18104045), Chirag Jain (18104002), Prem Vispute (18104059)*” for the partial fulfillment of the requirement for award of a degree *Bachelor of Engineering* in *Information Technology* to the University of Mumbai, is a bonafide work carried out during academic year 2021-2022

Prof. Kiran Deshpande
Guide
Head, Department of Information Technology

Prof. Kaushiki Upadhyaya
Co-Guide

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Place: A.P. Shah Institute of Technology, Thane

Date:

Acknowledgement

We have great pleasure in presenting the report on **Using AI for Designing ID Cards Embedded with Invisible QR Code**. We take this opportunity to express our sincere thanks towards our guide **Prof. Kiran B. Deshpande** and **Prof. Kaushiki Upadhyaya** for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards their constant encouragement, support and guidance throughout the development of project.

We thank **Prof. Vishal S. Badgujar** BE project co-ordinator, for being encouraging throughout the course and for guidance.

We also thank the faculties of IT Department for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Dhruva Mhatre (18104045)

Chirag Jain (18104002)

Prem Vispute (18104059)

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Dhruva Mhatre 18104045

Chirag Jain 18104002

Prem Vispute 18104059

Date:

Abstract

Organizations strongly encourage the usage of Identification cards (ID cards) as a measure of security. ID cards are used to reliably identify a person and, as a result, serve to improve the security of an organization by preventing unauthorized individuals from gaining access to the premises. However, these ID cards contain the user's personal information, which must be protected in today's world. This project proposes an approach for obtaining a data hiding mechanism for personal information on ID cards by using user data to develop a QR (Quick Response) code and embedding that QR code into the user image, making the QR code invisible to the naked eye. QR codes are wellknown for their error-correcting mechanism and are particularly good in hiding arbitrary information. Convolution neural networks (CNN) are a class of Deep neural networks that helps us analyze visual imagery and recognize patterns. The objective of this project is to use the CNN mechanism to hide a QR code in a user image. The proposed model is comprised of two CNNs (encoder CNN and decoder CNN). The encoder CNN's function will embed the QR code into the user image and generate an output image that appears just like the user image. The decoder CNN's role will be to take the encoder CNN's output as input and generate the embedded QR code image as output. By ensuring enhanced security and hiding critical information, our approach prohibits the duplication of the general QR code, preventing its misuse.

Contents

1	Introduction	2
2	Literature Review	3
3	Project Design	4
3.1	System Architecture	4
3.2	UML Diagrams	5
3.2.1	Acitivity Diagram	5
3.2.2	Sequence Diagram	6
4	Project Implementation	7
4.1	QR code Generator	7
4.2	Encoder CNN	8
4.3	Decoder CNN	9
4.4	Training the CNN Model	10
4.4.1	Result of Training QR codes and Tiny ImageNet dataset	12
5	Functional Testing	18
5.0.1	Unit Testing	18
5.0.2	Alpha Testing	18
5.1	Non Functional Testing	19
5.1.1	Security Testing	19
5.1.2	Compatibility testing	19
6	Result	20
7	Conclusions	21
	Bibliography	22
	Appendices	23
	Appendix-A	23
	Publication	24

List of Figures

3.1	Proposed System Architecture	4
3.2	Activity Diagram	5
3.3	Sequence Diagram	6
4.1	QR code Generator	7
4.2	Example of Generated QR code	8
4.3	Code Snippet to generate QR code	8
4.4	Proposed Encoder-Decoder CNN Model	9
4.5	Result of Training Tiny ImageNet dataset	10
4.6	Result after 40 Epochs	11
4.7	Result after 80 Epochs	12
4.8	Result after 120 Epochs	13
4.9	Result after 160 Epochs	14
4.10	Result after 200 Epochs	15
4.11	Result after 240 Epochs	16
4.12	Result after 280 Epochs	17
6.1	Model Training Loss	20

List of Abbreviations

AI:	Artificial Intelligence
CNN:	Convolution Neural Networks
QR code:	Quick-Response code
FC-DenseNet:	Fully Convolutional Dense Network
DenseNet:	Dense Convolutional neural networks
ResNets:	Residual Networks
ReLU:	Rectifier Linear Unit
Prep-Network :	Preparation Networks
RGB Channels:	Red-Green-Blue channels

Chapter 1

Introduction

Identification cards or ID cards are the most important credential for any organization, and it helps uniquely identify the person or verify them. However, these ID cards may contain some personal information like the mobile number or the person's address. Such crucial information, if fallen into the wrong hands, could be misused. Hence, here we are proposing a system where all the information of the user will be used to generate a QR code and that QR will be embedded in the user's image such that it will be invisible to the naked eyes. The main aim of our project is to develop a system that will embed the QR code (containing arbitrary user information) into a coloured image (User photograph), such that the QR image will be hidden in the background and the primary visible image will be the user image[1].

We are using the methodology of convolution neural networks and Image-Steganography for developing this system. Convolution neural networks (known as ConvNet or CNN) is a branch of Deep learning that is commonly applied to analyzing visual imagery. In the fields of image classification and recognition, CNN has been proven to be efficient. A CNN model works by extracting necessary features from images in the form of pixels, which helps in reducing the image size and provides us with data that is important in image recognition. CNN has multiple layers, including the convolutional, non-linearity, pooling, and fully-connected layers[2]. When a CNN trained model receives an image as input, the first layer that acts on it is the convolution layer. This layer consists of a feature detector of a specified size which moves all over the image matrix, multiplying its values with the original pixel values. After every convolution layer, a non-linearity layer is added. The pooling layer is generally used to further downsize the image by extracting more important pixels by using certain functions like Max pooling, Min pooling, etc. This helps reduce the network's training time by reducing the number of computations required. Steganography is a method for hiding information like text, image, etc. inside a cover image such that the very existence of the secret information is unknown.

In our model, we are aiming to use Convolution neural networks to achieve Image Steganography which will consist of two CNNs; encoder CNN and decoder CNN. The role of encoder CNN will be to take an input of the QR code image and colour image and generate an output of coloured image embedded with the QR code image such that the QR image will be undetected by the naked eyes. The role of decoder CNN is to recover the original QR code image from the coloured image[10].

Chapter 2

Literature Review

The most popular form of Steganography used nowadays is using images as a source of hiding data. These methods involve hiding messages by altering “noisy” areas by using methods like a least-significant bit or LSB, which directly alters the image pixels value[3]. This image steganography method limits the capacity of data that can be hidden in the carrier image; if more data is added, it may distort the image leading to a significant difference between the original image and the stego-image. To overcome this drawback, the authors of[4] proposed a method for image steganography using Fully Convolutional Dense Network (FC-DenseNet). Convolution neural networks are easier to train if they have shorter connections and perhaps produce more accurate and efficient results. Dense convolutional neural networks (hereafter referred to as DenseNet) connect each layer to every other layer in a feedforward fashion[5]. DenseNets can be considered as an extension of Residual Networks (ResNets)[6], where each layer obtains additional inputs from all preceding layers and passes on its feature-maps to all subsequent layers. In Resnet we combine features through Summation, whereas in DenseNet, features are combined through the method of concatenation before passing into another layer[5].

FC-DenseNets is mainly used for semantic segmentation. It uses an upsampling layer for recovering the spatial resolution of the input or output layer. The information from additional dense blocks of the same resolution is combined in the upsampled dense blocks, and the higher resolution information is obtained via the standard skip link between the upsampling and downsampling channels[7]. We are using FC-DenseNets for hiding the image, as shown in[4]. Here the authors are hiding information (secret image) into a carrier image in such a way that the stego-image will be visually similar to the carrier image.

In our project, we use this FC-DenseNet method to hide a QR code image into a carrier image, as shown in[1]. We are generating a QR code consisting of user data (such as phone no., address, email, etc.) and are embedding the QR code image into the user image. The goal of this system will be to provide data hiding through Cryptography and Steganography, such that the user data will be hidden through QR generation and hiding the QR code through Steganography so that there will be no knowledge of the existence of the QR in the first place.

Chapter 3

Project Design

3.1 System Architecture

An system architecture diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element. It gives the entire development team a visual overview making it easier to communicate ideas and key concepts in terms everyone can understand.

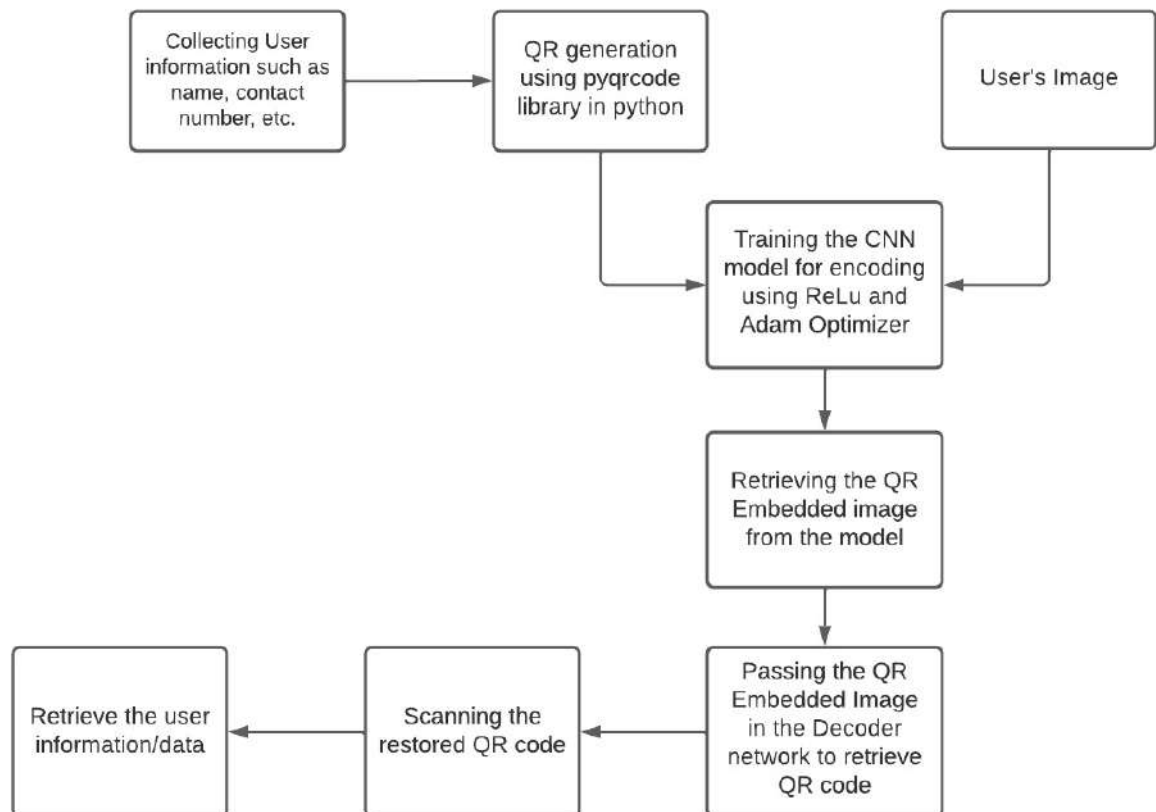


Figure 3.1: Proposed System Architecture

The system will be developed in order to achieve steganographic mechanisms by hiding the QR image (consisting of user data) into the user's photograph (carrier image) such that the QR image will be invisible to the naked eyes. The entire system will mainly consist of a QR code generator for generating the QR code image consisting of arbitrary user information, and the CNN network will mainly consist of three sections which are preparation network, hiding network, and reveal network. All these networks will collectively form an end-to-end system for encoding as well as decoding the image[8].

In step 1, we will collect the user personal information such as name,number,etc.

In step 2, we will generate the QR code consisting the personal information of the user using python.

In step 3, we will pass the QR code along with the User image to the training CNN model for encoding.

In step 4, we will retrieve the output of the Encoder CNN model.

In step 5, we will pass the output of the Encoder CNN to the Decoder CNN to retrieve the QR code from the User Image.

In step 6, We will scan the QR code to retrieve the User Information.

3.2 UML Diagrams

3.2.1 Acitivity Diagram

Activity diagram is an important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc

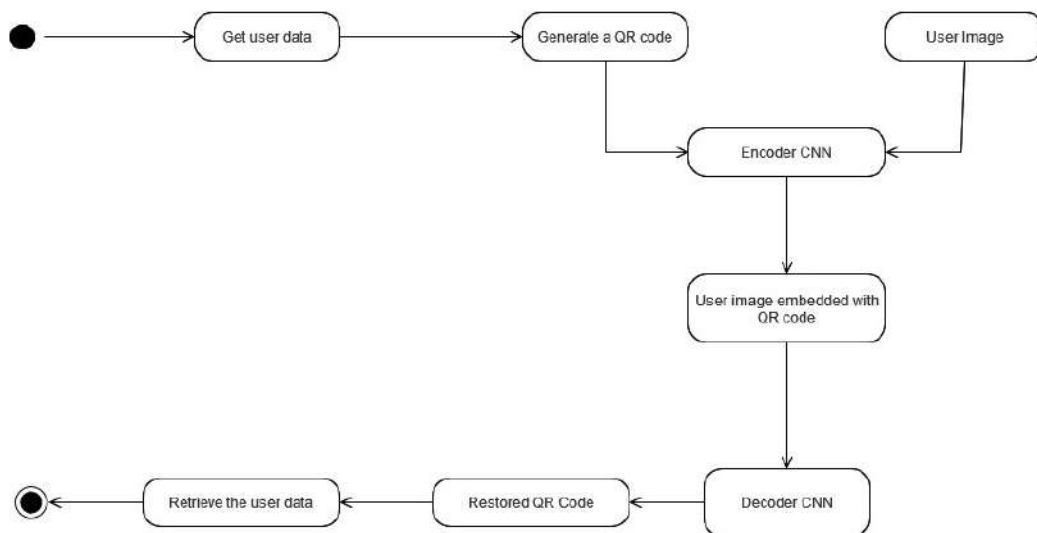


Figure 3.2: Activity Diagram

The flow of system starts from collecting User's personal information and generating QR code consisting of that information and after generating the QR code, the User image and QR code are given as input to the Encoder CNN model for embedding and retrieving the Embedded QR code image, than the embedded image is given as input to the Decoder CNN for retrieval of the QR code to get the User data back.

3.2.2 Sequence Diagram

In the field of software engineering, a sequence diagram or system sequence diagram depicts object interactions in time order. It depicts the scenario's objects as well as the sequence of messages exchanged between them in order to carry out the scenario's functionality.

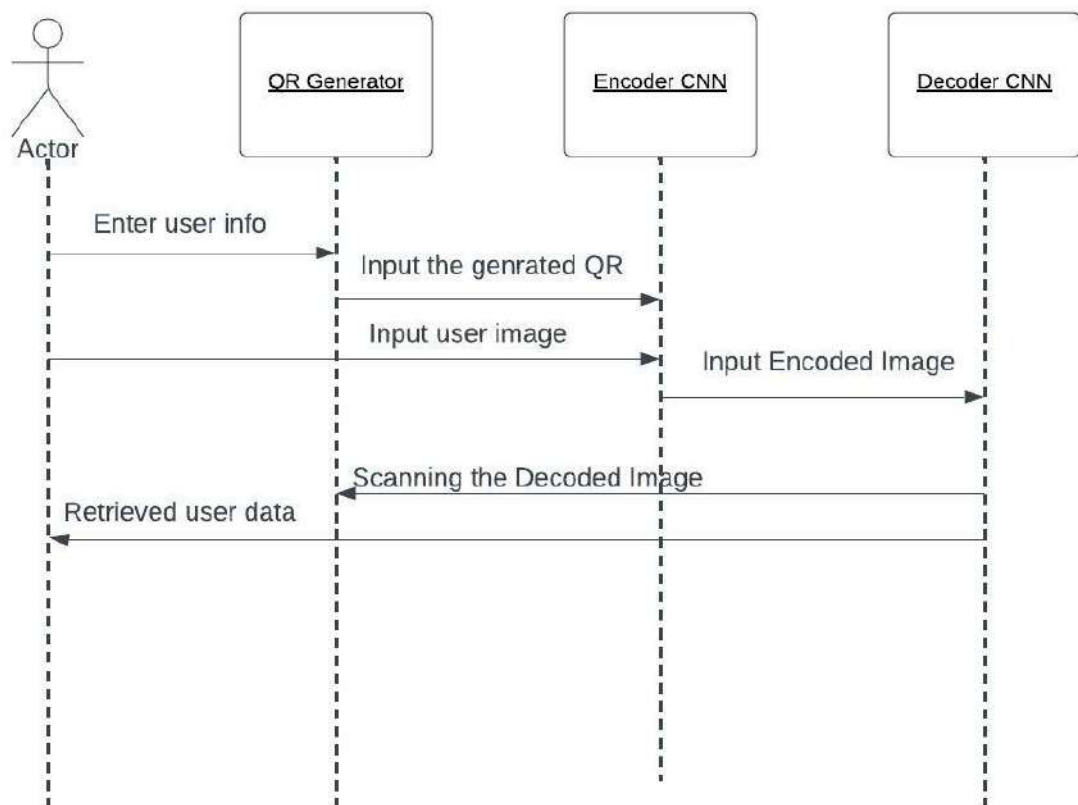


Figure 3.3: Sequence Diagram

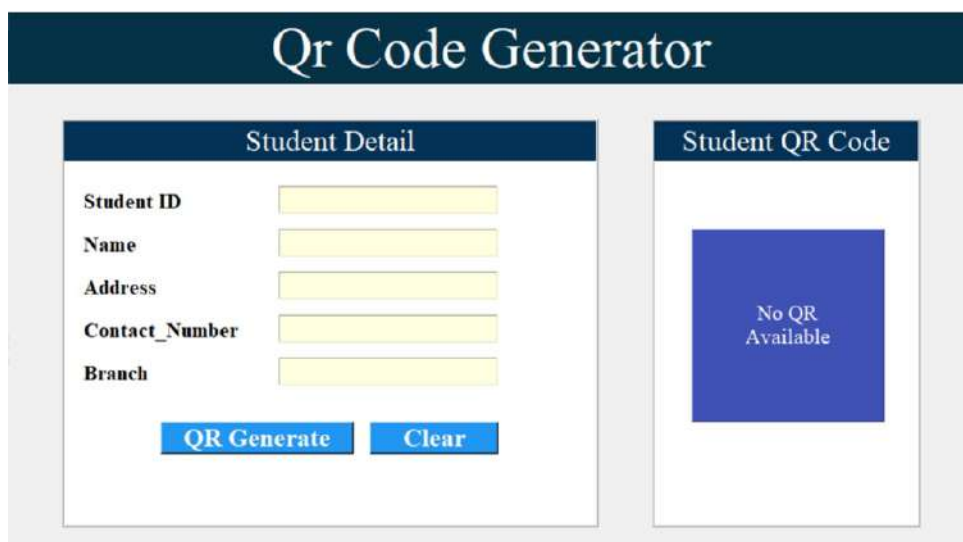
First the user enter its data in the system, the data is then used to generate a QR code. Later the generated QR code and the User image is given input to the Encoder CNN. The encoder CNN generates an Image which is then given as input to the Decoder CNN. The Decoder CNN retrieve the embedded QR code and after scanning, the user info is retrieved.

Chapter 4

Project Implementation

4.1 QR code Generator

The QR Code is a two-dimensional matrix code that transmits information by arranging dark and light elements, known as "modules," in columns and rows and are widely known and used for their error-correcting mechanism[9]. Our system's QR code generator model is responsible for generating a QR code by taking user information as input. We have used the pyqrcode library of python for developing a GUI based QR generator as shown in figure 4.1 and 4.2 where user data such as name, number, address, etc. will be provided to the system as input, and as an output, the system will generate a QR code encrypting the given information.




Qr Code Generator	
Student Detail	Student QR Code
Student ID <input type="text"/>	
Name <input type="text"/>	
Address <input type="text"/>	
Contact_Number <input type="text"/>	
Branch <input type="text"/>	
<input type="button" value="QR Generate"/> <input type="button" value="Clear"/>	

Figure 4.1: QR code Generator



Figure 4.2: Example of Generated QR code

As shown in figure 4.3, QR code is generated using the python libraries consisting of the User Information and which gets stored in the local machine.

```
def generate(self):
    if self.var_student_code.get()==' ' or self.var_name.get()==' ' or self.var_address.get()==' ' or self.var_contact_number.get()==' ' or self.var_branch.get()==' ':
        self.msg="All Fields are Required!!!"
        self.lbl_msg.config(text=self.msg,fg='red')
    else:
        qr_data=f"Student ID: {self.var_student_code.get()}\nStudent Name: {self.var_name.get()}\nAddress: {self.var_address.get()}\nContact_Number: {self.var_contact_number.get()}\nBranch: {self.var_branch.get()}"
        qr_code=qrcode.make(qr_data)
        #print(qr_code)
        qr_code=resizeimage.resize_cover(qr_code,[180,180])
        qr_code.save("qr code/Student_"+str(self.var_student_code.get())+'.png')
        #=====QR code image update=====
        self.im=ImageTk.PhotoImage(file="qr code/Student_"+str(self.var_student_code.get())+'.png')
        self.qr_code.config(image=self.im)
        #=====updating notification=====
        self.msg="QR Generated Successfully!!!"
        self.lbl_msg.config(text=self.msg,fg='green')
```

Figure 4.3: Code Snippet to generate QR code

4.2 Encoder CNN

The mechanism of encoder CNN is to embed the QR image generated in section A. into the user image such that the QR image will not be visible to the naked eyes, and the output image of encoder CNN will be visually similar to the user image. The encoder CNN comprises of two networks which are the Prep-network and the Hiding Network. The Prep-Network is responsible for preparing the secret image to be hidden. It serves two purposes; if the secret image is smaller than the cover image, the Prep-Network will increase the size of the secret image progressively in order to distribute the secret image's bits across the pixels of the cover image. The more important purpose of the Prep-network is to extract more useful features such as textures or edges for succinctly encoding the image. The Prep-Network comprises 2 convolution layers of 65 filters (50 3x3 filters, 10 4x4 filters, and 5 5x5 filters). In the Hidden Network, the output of the Prep-network along with the cover image will

be given as the input, and it will generate a container image as the output to it. This network is responsible for concatenating the convoluted RGB channels of the cover image and the extracted channels of the secret image and generating a container image that will be visually similar to the carrier image. The best-suited architecture for this network consists of 5 convolution layers that have 65 filters (50 3x3 filters, 10 4x4 filters, and 5 5x5 filters).

4.3 Decoder CNN

The mechanism of decoder CNN is to extract the QR code image from the output image of the encoder CNN. It takes the output image of the encoder CNN as input and provides us with the original QR image which was embedded in the previous user image. The decoder CNN comprises of the Reveal Network whose role is to reveal the original secret image from the container image (output of encoder CNN). This network only takes the container image as an input and provides us with a decoded secret image. The architecture for this network is the same as for the hiding network, i.e., it consists of 65 filters (50 3x3 filters, 10 4x4 filters, and 5 5x5 filters).

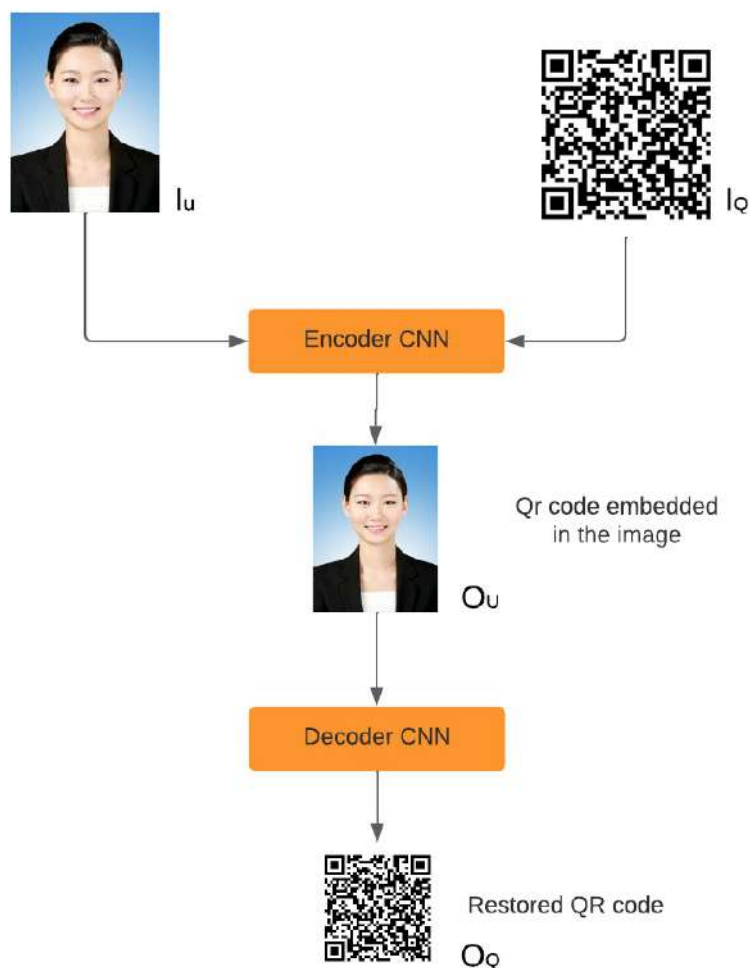


Figure 4.4: Proposed Encoder-Decoder CNN Model

4.4 Training the CNN Model

For training the CNN model, we are using the Tiny ImageNet dataset, which consists of 2000 images of dimension $64 \times 64 \times 3$. Firstly, we are training the model for a randomly picked pair of images from the tiny ImageNet dataset, such that half of the images are used as cover images, and the other half is used as the secret image. In the first phase, the dataset is processed using the Keras image library and the Adam optimizer. The features of the secret image are extracted in the Prep network of the Encoder CNN, and then, the cover image and the output of PrepNetwork are passed as input in the Hidden network of the Encoder CNN. At this time, the image size of both images is converted to 64×64 , the colour image is normalized, and pixel values are input to CNN in the range of 0 to 255. We get a container image that will contain the embedded secret image into the cover image such that the container image will be visually similar to the cover image. In the second phase, we give the outputted container image as input to the Decoder CNN and restore the secret image from it. The encoder and decoder CNN are trained simultaneously during training, and the weights are updated simultaneously. We have trained the model with a batch size of

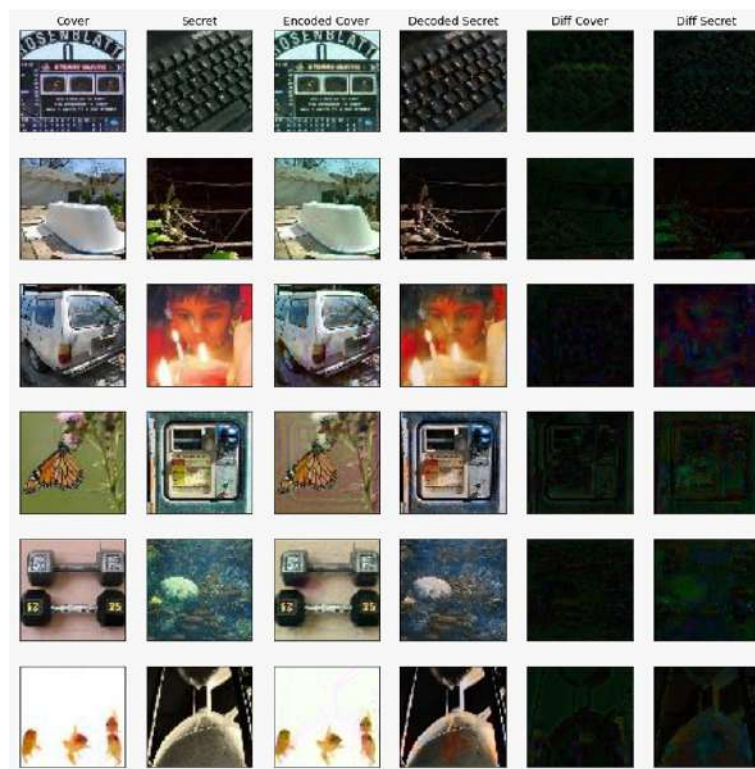


Figure 4.5: Result of Training Tiny ImageNet dataset

32 and performed learning with epoch 300, and after every epoch 40, we generate an output, as shown in Figure 4.5. We have used the optimization algorithm Adam, weights 0.001, and a learning rate of 0.001. The Rectifier Linear Unit (ReLU) is used as an activation function to train this layered network, which reduces the model training computing needs by addressing the vanishing gradient problem. We used a sequential model with multiple layers consisting of Conv2D, padding, strides, and Batch-normalization blocks. After the model training is completed, we will use this model for embedding QR codes into colour images. For the cover image, we will be using the Tiny ImageNet dataset, and for the secret image,

we will use the QR-DN1.0 dataset for importing the QR code. We will train the CNN model using these datasets as we did in the previous phases.

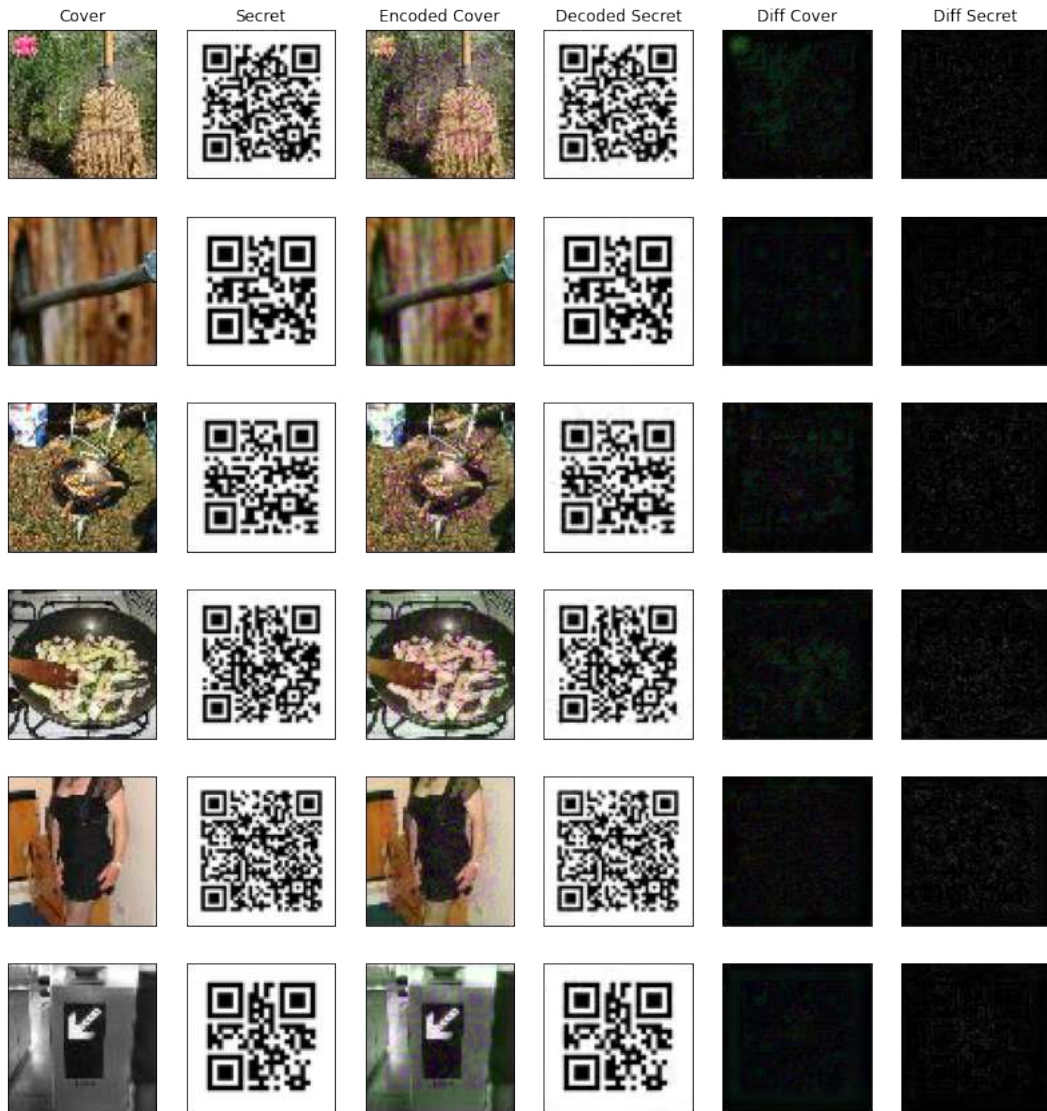


Figure 4.6: Result after 40 Epochs

Once this CNN model has undergone training, we will use it to embed our QR code image I_q into the User Image I_u and restore the embedded QR image O_q from container image O_u , as shown in figure 4.4. Firstly, provide the User Image I_u and the QR code image I_q as the input to the Encoder CNN; embedded image O_u will be the output to this CNN. Then this generated embedded image O_u will then be inputted to the decoder CNN, generating the restored image O_q .

4.4.1 Result of Training QR codes and Tiny ImageNet dataset

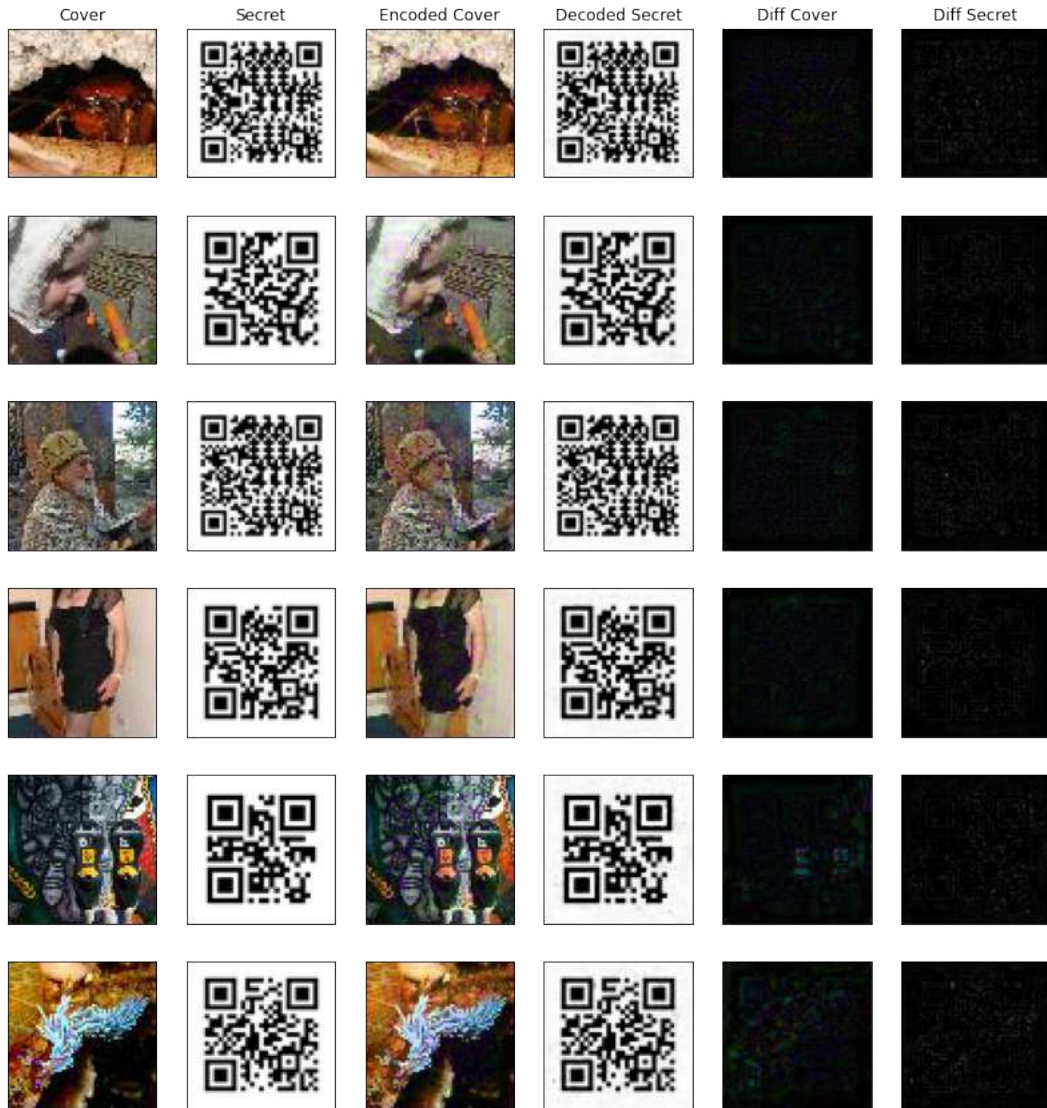


Figure 4.7: Result after 80 Epochs

In the above figure 4.7, after the model is trained for 80 epochs, the accuracy of the model gets better and the visibility of QR code decreases in the carrier image.

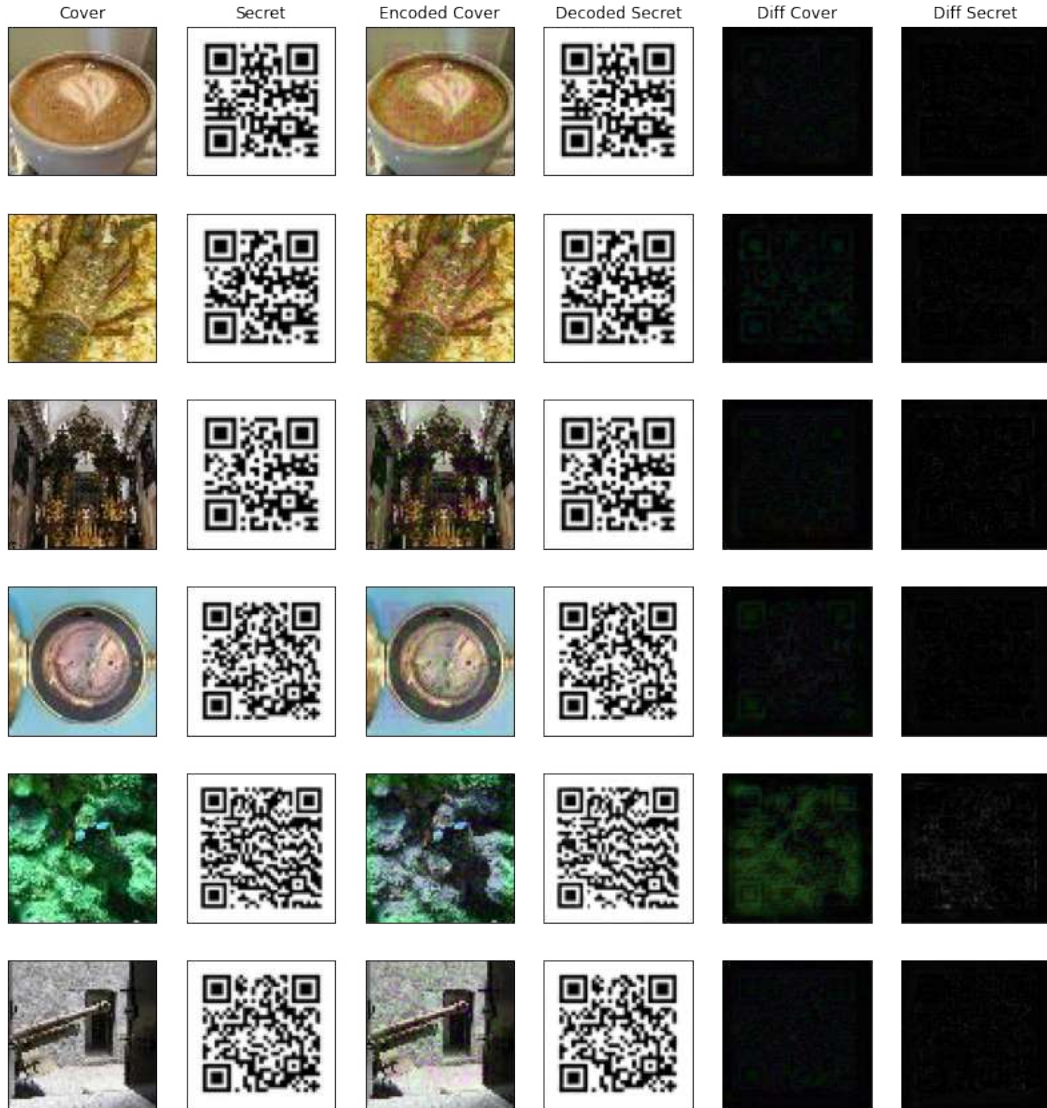


Figure 4.8: Result after 120 Epochs

In the above figure 4.8. after training for 120 epochs, the model's accuracy improves and the visibility of the QR code in the carrier picture reduces.

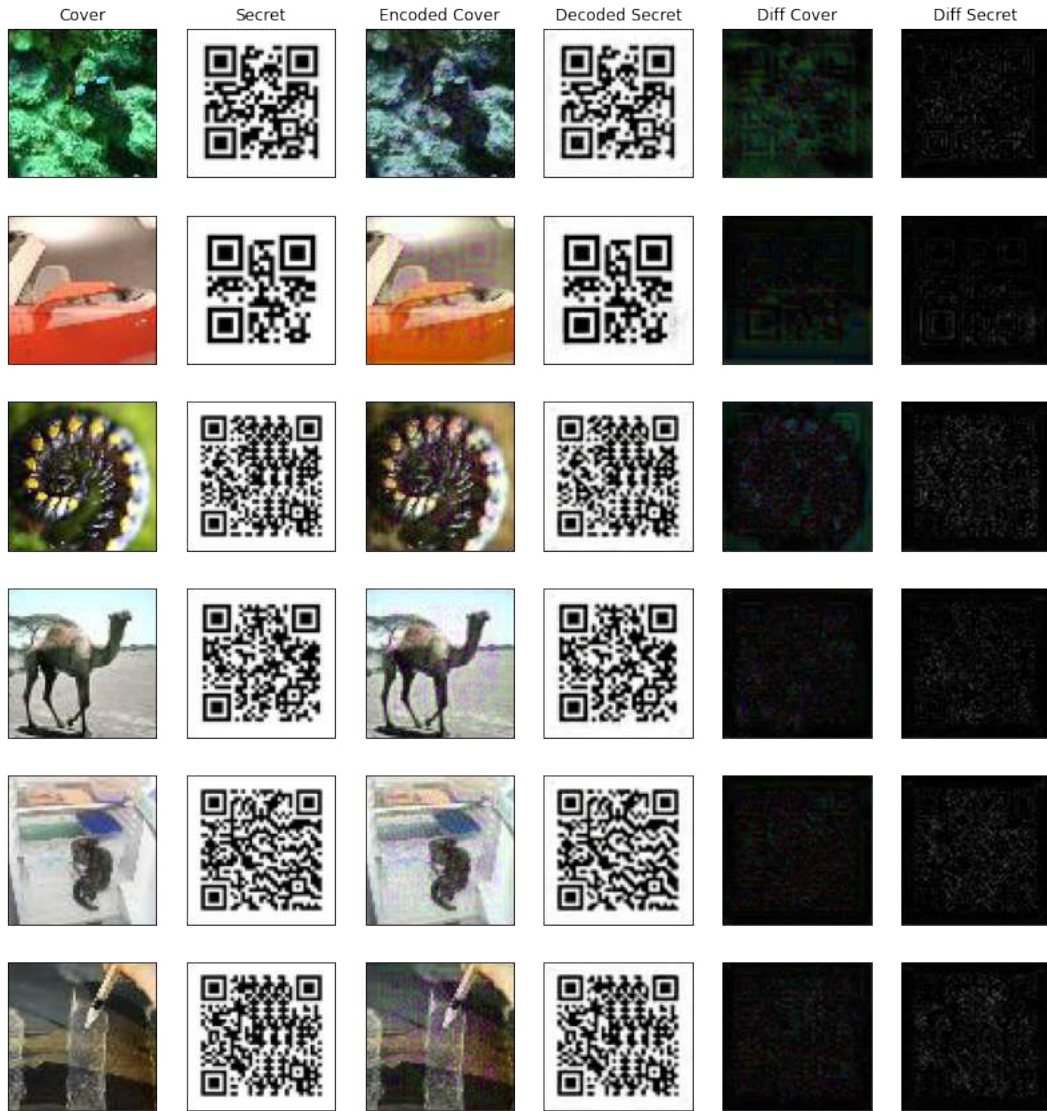


Figure 4.9: Result after 160 Epochs

In the above figure 4.9. as the model training progresses after 160 epochs , the visibility of QR code in the carrier image decreases furthermore.

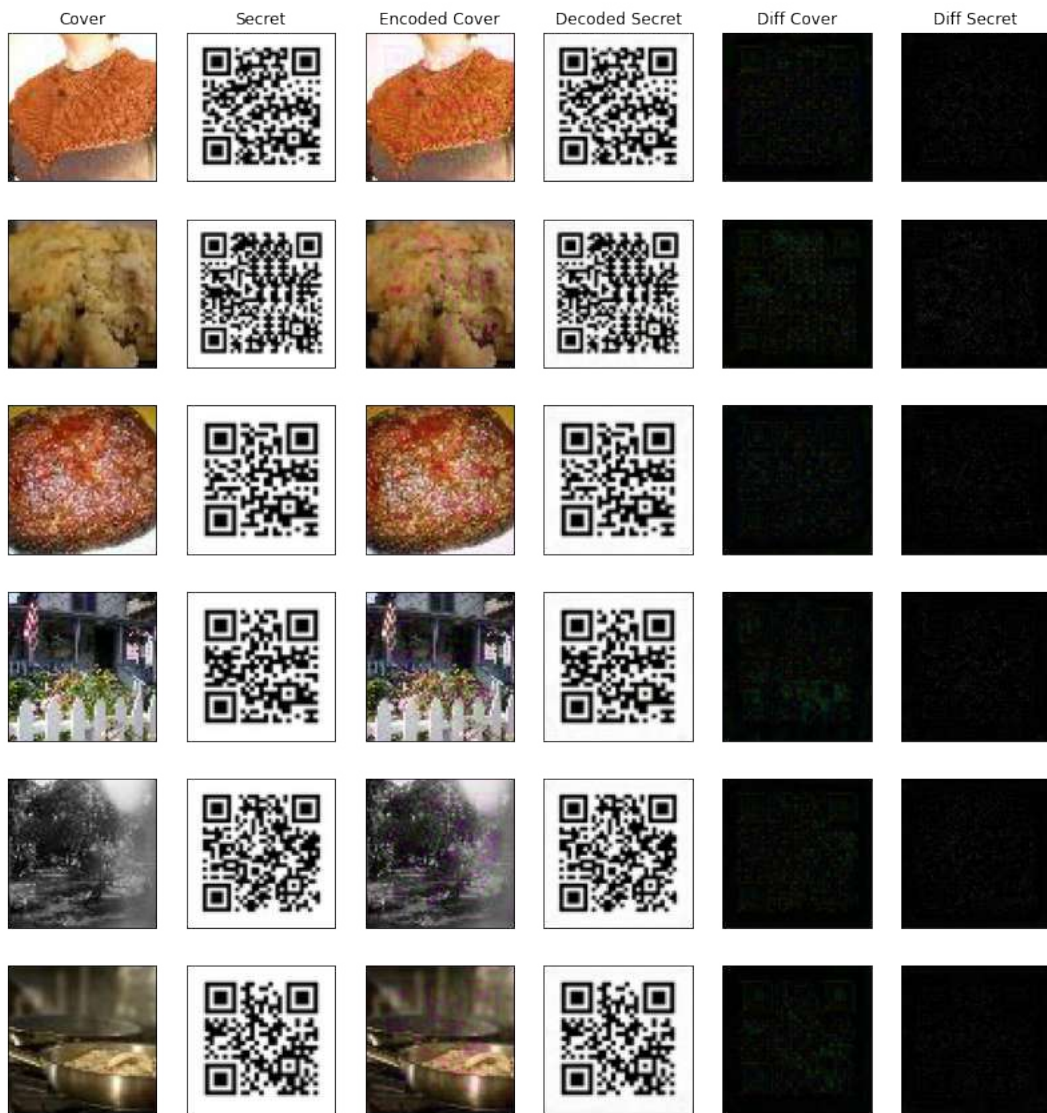


Figure 4.10: Result after 200 Epochs

The visibility of the QR code in the carrier image drops more as the model training advances beyond 200 epochs.

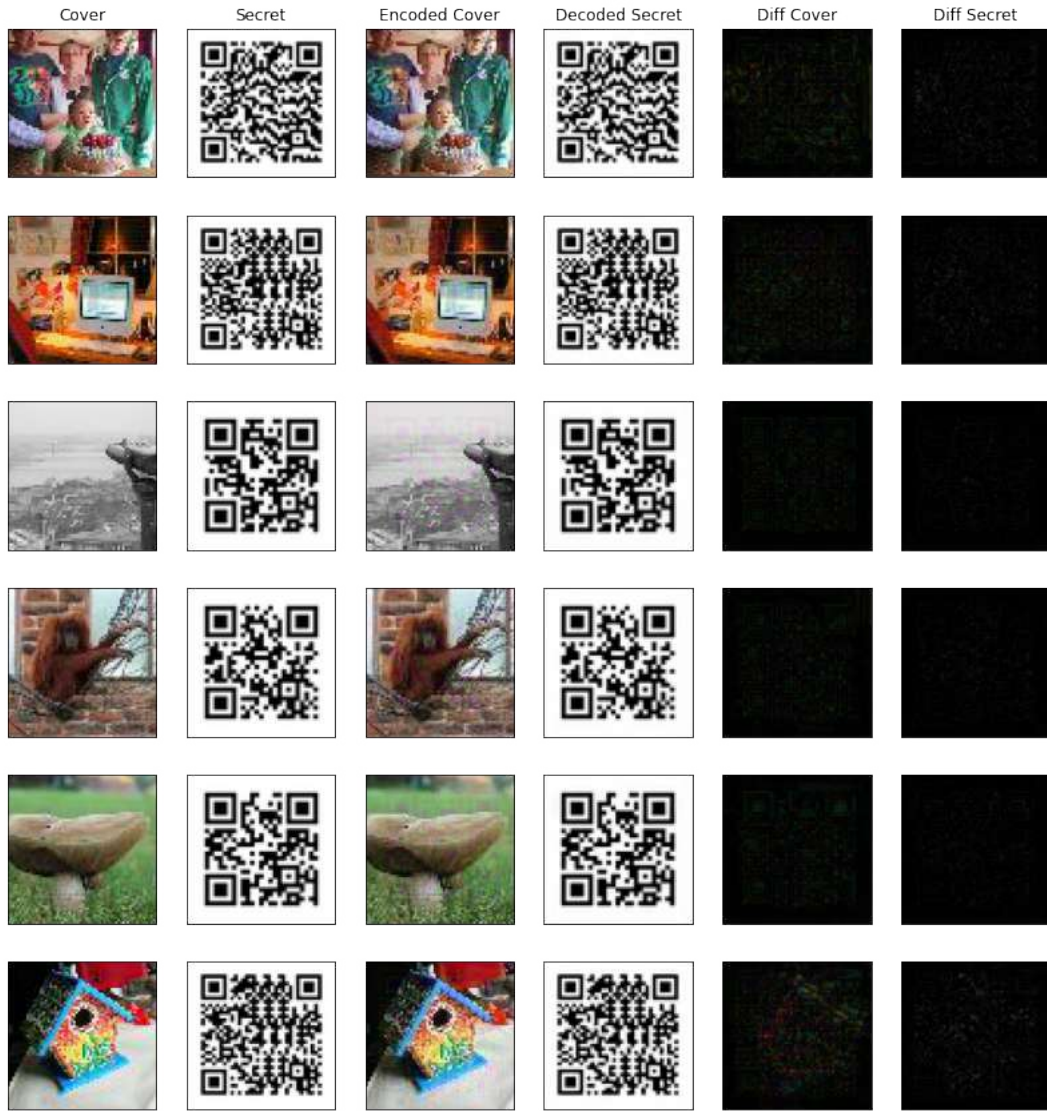


Figure 4.11: Result after 240 Epochs

In the above figure 4.11. as the model is trained for 240 epochs, there is little to no variation between the visibility of carrier image and the encoded image.

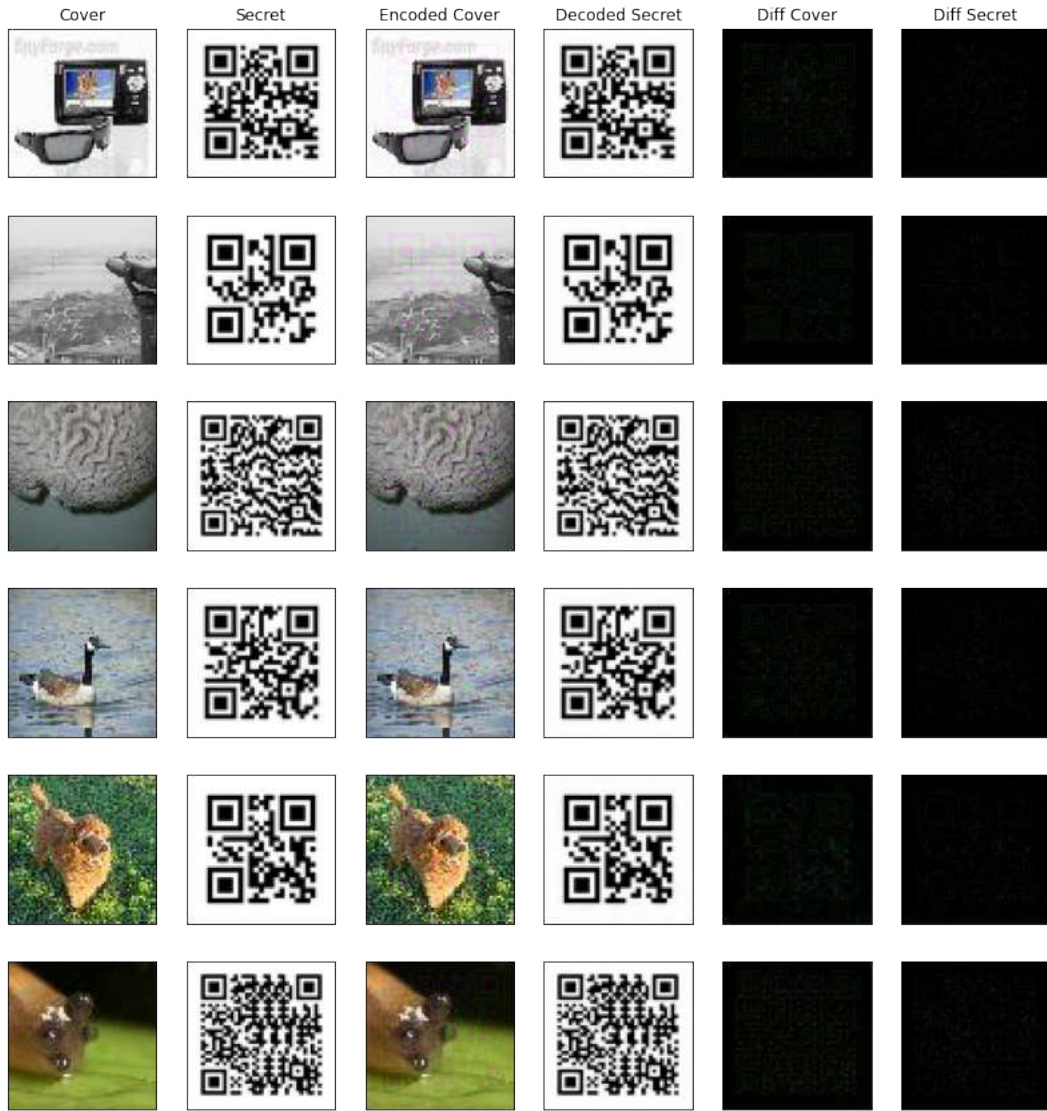


Figure 4.12: Result after 280 Epochs

In the above figure 4.12. as the model completes the training after 280 epochs, the QR code gets completely embedded in the carrier image and it is not visible in the carrier image.

Chapter 5

Functional Testing

5.0.1 Unit Testing

Before you can test an entire software program, make sure the individual parts work properly on their own. Unit testing validates the function of a unit, ensuring that the inputs (one to a few) result in the lone desired output. This testing type provides the foundation for more complex integrated software. When done right, unit testing drives higher quality application code and speeds up the development process.

Unit testing may be done manually, however automating the process will reduce delivery times and boost test coverage. Because flaws will be detected sooner in the testing process and will take less time to repair than if they were discovered later, debugging will be easier as a consequence of unit testing.

Unit testing is the most suited software approach for our proposed model. At this point, we started creating code in the form of units, such as loading the dataset, diving the dataset into training and testing section, creating the encoder CNN model, creating the decoder CNN model and integrating both, training the proposed system for QR and Tiny Imagenet Dataset.

5.0.2 Alpha Testing

Alpha testing uses internal team members to evaluate the product. These team members should be knowledgeable of the project but not directly involved in its development or testing. Where some builds might still be somewhat unstable, alpha testing provides an immediate subset of testers to root out major bugs before the software is seen by external users.

Alpha testing can be used in our project to test the model for embedding QR codes into students'/users' images and see if they are correctly embedded or not.

5.1 Non Functional Testing

5.1.1 Security Testing

Security Testing is a type of Software Testing that uncovers vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders. The purpose of Security Tests is to identify all possible loopholes and weaknesses of the software system which might result in a loss of information, revenue, reputе at the hands of the employees or outsiders of the Organization.

Security is a top priority in our project, so we need to double-check everything. We need to make sure that the QR codes are properly embedded, that none of them are visible and that they can't be scanned without a decoding mechanism, and that there is no information leakage after the user scans the QR code.

5.1.2 Compatibility testing

Compatibility testing assesses how well an application or piece of software will work in a variety of situations. It's used to determine if your product is compatible with a number of operating systems, platforms, and resolutions. The objective is to guarantee that the functionality of your product is consistently supported in any environment that your end customers are likely to experience.

In this project, we are using Google Collab for running our code, so all the required libraries must be imported correctly and google collab can run on Windows, Linux but cannot run on Mac OS.

Chapter 6

Result

We have created a model which embeds qr code into a image using convolutional neural networks. We have trained the model with a batch size of 32 and performed learning with epoch 300, and after every epoch 40, we generate an output, as shown in Figure 4.6. We have used the optimization algorithm Adam, weights 0.001, and a learning rate of 0.001. The Rectifier Linear Unit (ReLU) is used as an activation function to train this layered network, which reduces the model training computing needs by addressing the vanishing gradient problem. As shown in Fig 6.1 as the model training progresses the accuracy of the model increases.

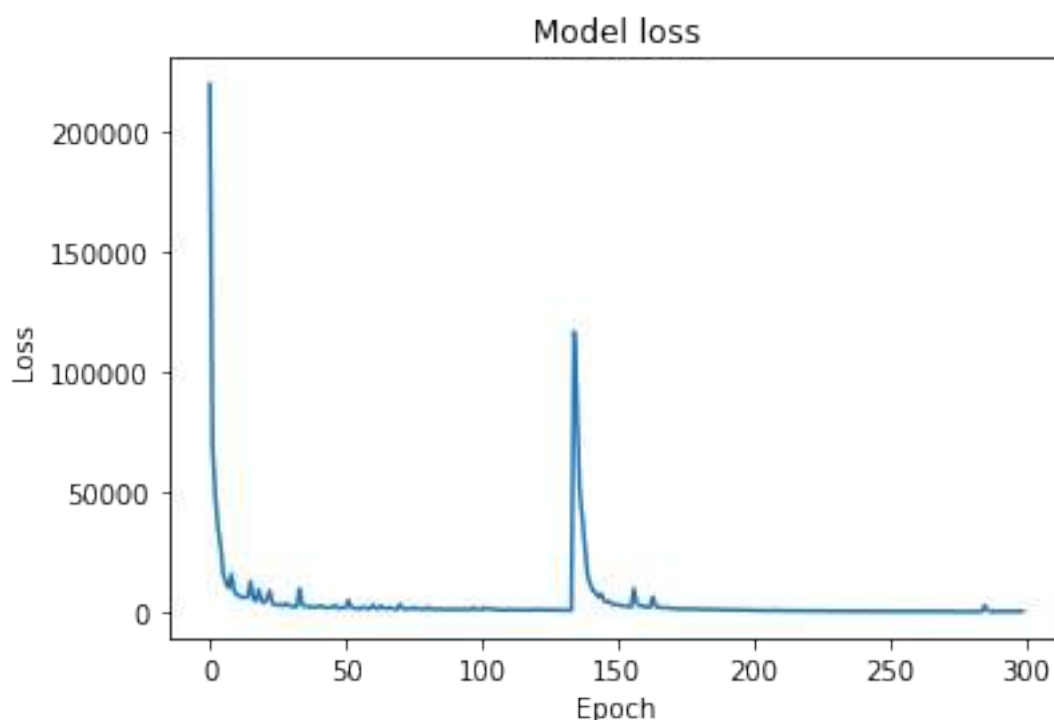


Figure 6.1: Model Training Loss

Chapter 7

Conclusions

The emerging technological advancements have increased the risk of data breaching, thus making data security and privacy our primary responsibility. Identification cards [ID-Cards] are essential for any organization and are used as proof of identification and verification. These ID cards contain personal information about the user, such as its name, number, address, etc. This information is essential and can be misused if fallen into the wrong hands. The model proposed in this project provides an adequate data hiding mechanism for protecting this sensitive data. Here we are achieving data security by generating a QR code consisting of the user's information, and then we will embed the generated QR are into the user's image such that the very existence of the QR will be unknown, i.e., the QR code will be invisible to the naked eyes. The concepts of Convolution neural networks are used to achieve the stenographic mechanism where the entire system consists of two CNNs (Encoder CNN and Decoder CNN) for embedding and decoding the QR code. The proposed system can be beneficial in providing data security and thus can prevent a data breach. Further, we can even add developments in the system for managing and maintaining the attendance system. Including web development mechanisms, we can develop a system for marking the attendance of students/employees simply by scanning the ID cards. This can make attendance marking quick and effective processes and avoid the chances of proxy marking or false attendance, thus increasing the organization's efficiency. We would research and work upon adding these developments to the proposed systems in the future.

Bibliography

- [1] K. Yamauchi and H. Kobayashi, "Invisible QR Code Generator Using Convolutional Neural Network," IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, 2020, pp. 4009-4014, doi: 10.1109/IECON43393.2020.9254709.
- [2] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.
- [3] Steganography and Steganalysis, J.R. Krenn, January 2004
- [4] Hide the Image in FC-DenseNets to another Image, Duan Xintao, Liu Nao
- [5] Densely Connected Convolutional Networks, Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger
- [6] K. He, X. Zhang, S. Ren, and J. Sun., Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.
- [7] The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation, Simon Jégou, Michal Drozdal, David Vazquez, Adriana Romero, Yoshua Bengio
- [8] Baluja, Shumeet, "Hiding images in plain sight: Deep steganography.", Advances in neural information processing systems 30, 2017.
- [9] Gao, Zhongpai, Guangtao Zhai, and Chunjia Hu, "The invisible qr code." Proceedings of the 23rd ACM international conference on Multimedia. 2015.
- [10] Yamauchi, Kohei, and Hiroyuki Kobayashi, "A CNN based invisible QR code generator for human living space." IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society. Vol. 1. IEEE, 2019

Appendix

Google Collab and Python Libraries Installation

1. Go to <https://colab.research.google.com/>
2. To install python on your local device go to <https://www.python.org/downloads/>

3. Go to command-prompt and do the following

- a. To import tkinter

```
$ pip install tk
```

```
$ from tkinter import*
```

4. To install PIL Library and Qr code library to generate QR-Code

```
$ pip install pillow
```

```
$ pip install qrcode[pil]
```

```
$ import qrcode
```

5. To import Image library

```
$ from PIL import Image,ImageTk
```

The ImageTk library will be used for resizing the QR-code and saving the QR-code in the local machine in PNG format. Just provide the right path where the QR-code needs to be stored.

6. To use wandb which is the database used to store the output.

```
$ pip install wandb
```

```
$ import wandb
```

7. To use the tiny-imagenet dataset.

Go to <http://cs231n.stanford.edu/tiny-imagenet-200.zip>

8. To use the matplotlib and numpy libraries.

```
$import matplotlib.pyplot as plt
```

```
$import numpy as np
```

Publication

Jain Chirag, Mhatre Dhruva, Vispute Prem, Deshpande Kiran, Upadhyaya Kaushiki,
"Using AI for Designing ID cards Embedded with Invisible QR code", Inter-
national Conference on Computational Intelligence and Innovative Technologies, (**ICCIIT
2022**– Elsevier SSRN series), (April 29, 2022).