Detection of Human Face from Sketches using Deep Learning Networks

School of science and technology

Middlesex University London

Director of Studies: Professor Xiaohong Gao

Supervisors: Prof. Raja Nagarajan

Mohsen Abbey

30 November 2020

Table of Contents

Chapter 1 Table of Contents

CAPTIONS OF FIGURES:

Abstract

In the law and enforcement, sketching is a common technique to record witnesses' version of perceived suspects. Then these sketches are employed to identify the correct persons from a photo database, which will be a time consuming task if not impossible for a human being. This work investigates the application of artificial techniques (AI), specifically, the recent state of the art deep learning techniques to identify people based on their sketches for law and enforcement. The dataset is collected from Chinese database (CUHK student dataset) that includes both photos and their corresponding sketches. The sketches are drawn by artists. The deep learning network is Siamese network, which uses Python programming language. Since the data base contains only less than 200 photos, data enlargement is also conducted to generate sketch images from face images, which generates another 500 pairs of images. As a result, the Siamese network achieved 96% accuracy when retrieving photos from sketch images. Future work include further enlarging the database and evaluating other deep learning architectures.

1. Introduction

Face sketch photo recognition is to find similar match a sketched drawn by a forensic artist to one of many face photos in the database. This has attracted lots of attention and has made a crucial impact on both law enforcement and digital entertainment industry. As an example, it is important to be able to search and find similar or match photos from police mug-shot databases using a sketch drawing when the photo of a suspect is not available.

Sketches achieved from eyewitness images of criminals have proven to be useful in capturing criminals, mostly when there is a lack of evidence. Automated methods to identify subjects showed in sketches have been proposed in literature, but their performance is still unsatisfactory. Even though great progress has been made in this field, previous methods only work on sketches with rich textures which are not easily to achieve.

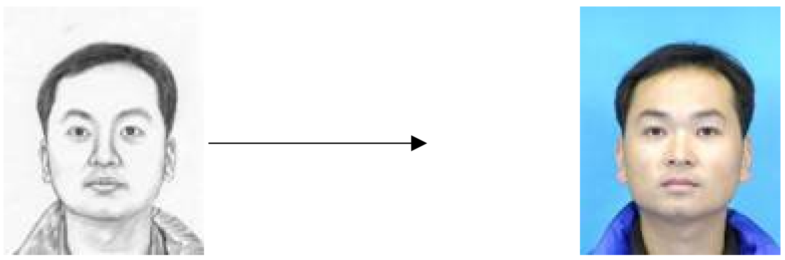


Figure 1: Used sample Photos from CUHK Face Sketch Database (CUFS) [12]

We are using Deep Learning and Transfer Learning models in Convolutional Neural Network (CNN) we use face sketch database including sketches of 188 people from the CUHK student dataset (Wang et al, 2009). Experiments on this large-scale dataset show that our approach significantly outperforms state-of-the-art methods on all publicly available composite sketch datasets.

The major challenge of sketched face photo recognition is to match images in different poses, viewing conditions and facial expressions. Sketches are a brief representation of human faces, often containing shape overstatement and having different textures than real photos. Currently, we are working to represent these facial features using deep learning techniques.

2. Start of the art

2.1 AI and deep learning

The field of Artificial Intelligence (AI) is basically when machines or robots can do the tasks that normally require human intelligence. It includes machine learning, where machines can learn by experience and develop skills without human engagement. Deep Learning (DL) is a subcategory of machine learning (ML) where Artificial Neural Networks (ANN), algorithms inspired by the human brain and learn from large amounts of data. It's similar to how we learn from experience, the deep learning algorithm can perform a task continually, and every time modifying a little bit to achieve better result and improve the outcome. We refer to 'deep learning' because the neural networks have various layers (deep layers) that enable learning. Like any problem that requires thinking to figure out, deep learning can learn to solve. Figure 2.1 illustrates the relationships between AI, ML, and DL.

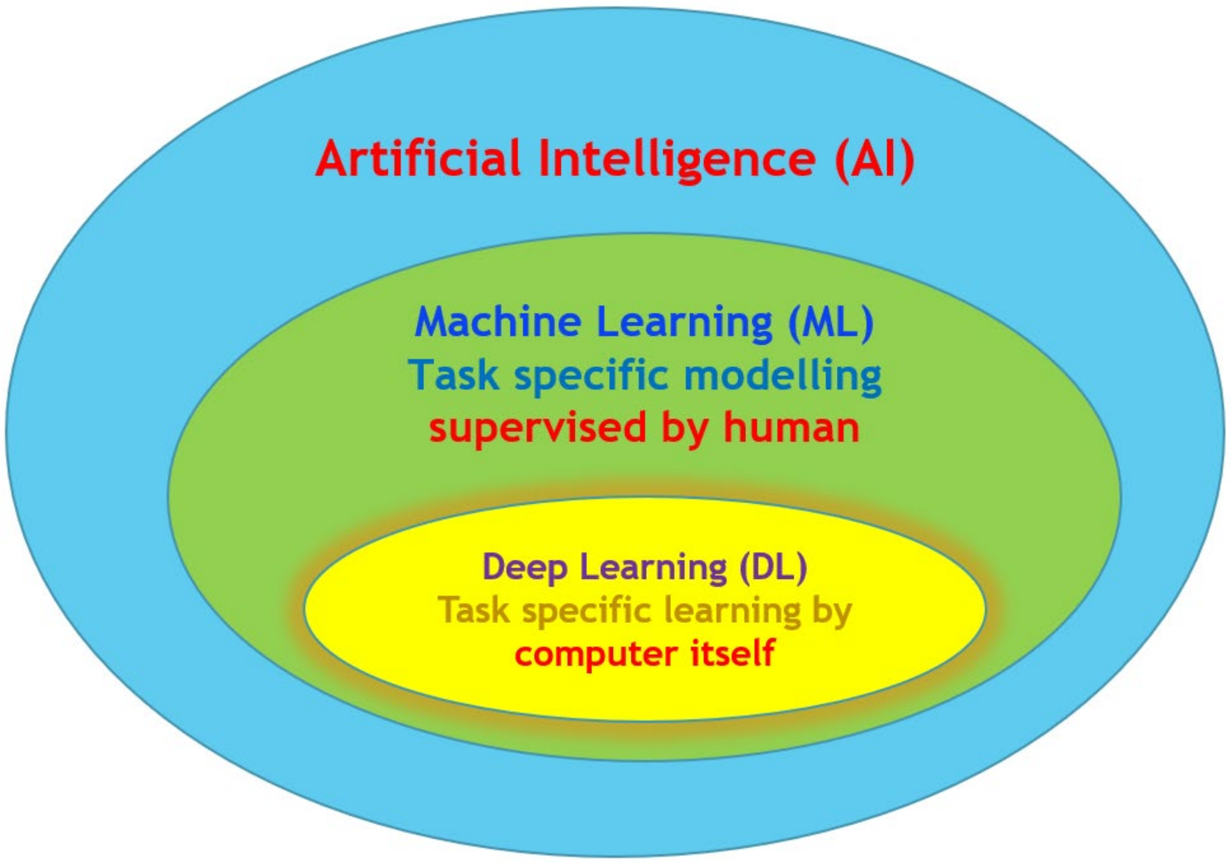


Figure 2.1 The relationships between AI, ML, and DL.

The amount of data we generate every day is amazing it is the resource that makes deep learning possible. Since deep-learning algorithms require a ton of data to learn from, this increase in data creation is one reason that deep learning capabilities have grown in recent years. In addition to more data creation, deep learning algorithms benefit from the stronger computing power which is available today as well as the creation of Artificial Intelligence (AI) as a Service.

Deep learning allows machines to solve complex problems even when using a data set that is very different, unstructured, and inter-connected. The deep learning algorithms learn more and they perform better.

2.2 Application of AI to photo to sketch

Computers can be trained to interpret images the same way our brains do and to evaluate the images much more comprehensively in details than human can. When applied to image processing, artificial intelligence (AI) can power face recognition and authentication functionality for ensuring security in public places, detecting, and recognizing objects and patterns in the images and videos, and many more.

Generally speaking, image processing is using a method or algorithm in order to improve the image or obtain information from it for further use. One of the methods of image processing, is called Digital image processing which is used for obtaining information from digital images with the help of computer algorithms.

In this sort of processing, the input is an image. and the output may be an image or information related with that image, such as data on features, characteristics, bounding boxes, or masks.

Today, image processing is widely used in medical visualization, biometrics, self-driving vehicles, gaming, surveillance, law enforcement, and many more. Here are some of the main purposes of image processing:

- Visualization, representing data in a logical way, giving visual form that aren't visible.
- Image sharpening, enhancing and restoration improving the quality of administered images.
- Image retrieval which is with help of image search
- Image features or objects measurement that measure objects in an image.
- Pattern recognition which recognizes and classify the objects in an image, identify their positions, and understand the scene.

2.3 Image Recognition in AI

Image recognition is the procedure of identifying features of particular objects or attributes in an image. Image recognition with AI uses such techniques as object detection, object identification, and segmentation.

This is where Artificial Intelligence solutions stand out. When we process all of the image segments, you're ready to build, train, and test an actual AI solution. The process of deep learning development includes a full cycle of operations from data acquisition to incorporating the developed AI model into the end system. Figure 2; image recognition with Convolution Neural Network. Which in convolution section we pooling layers of image and its features for processing in max pooling by detaching the segments, then we are identifying similarity between connected layers then we classify them in binary

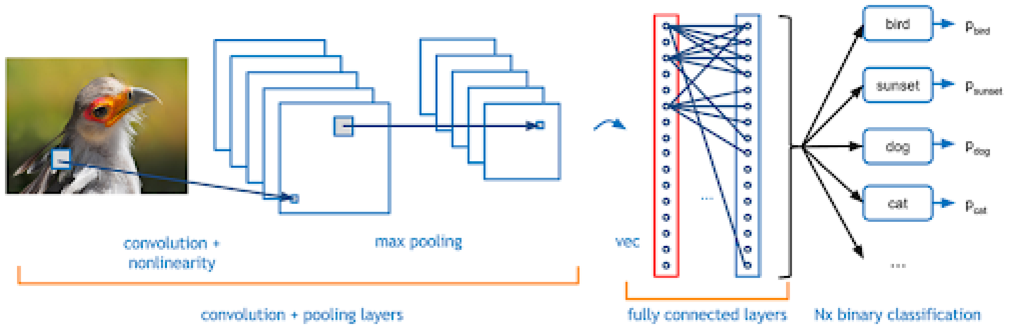


Figure 2: Image recognition with a CNN

2.4 Deep learning applied to generate photo images from sketches

While significant progress has been made to retrieve photos from hand sketches, another approach is to generate photos from sketches first and then to retrieve photos from these generated photos, which is particularly important when retrieval images that are in colour. This is because hand-drawing sketches are in monocoloured.

The work presented by Chen et al (Chen 2020) on DeepFaceDrawing. Figure 2.2 presents the network architecture for deepface drawing.

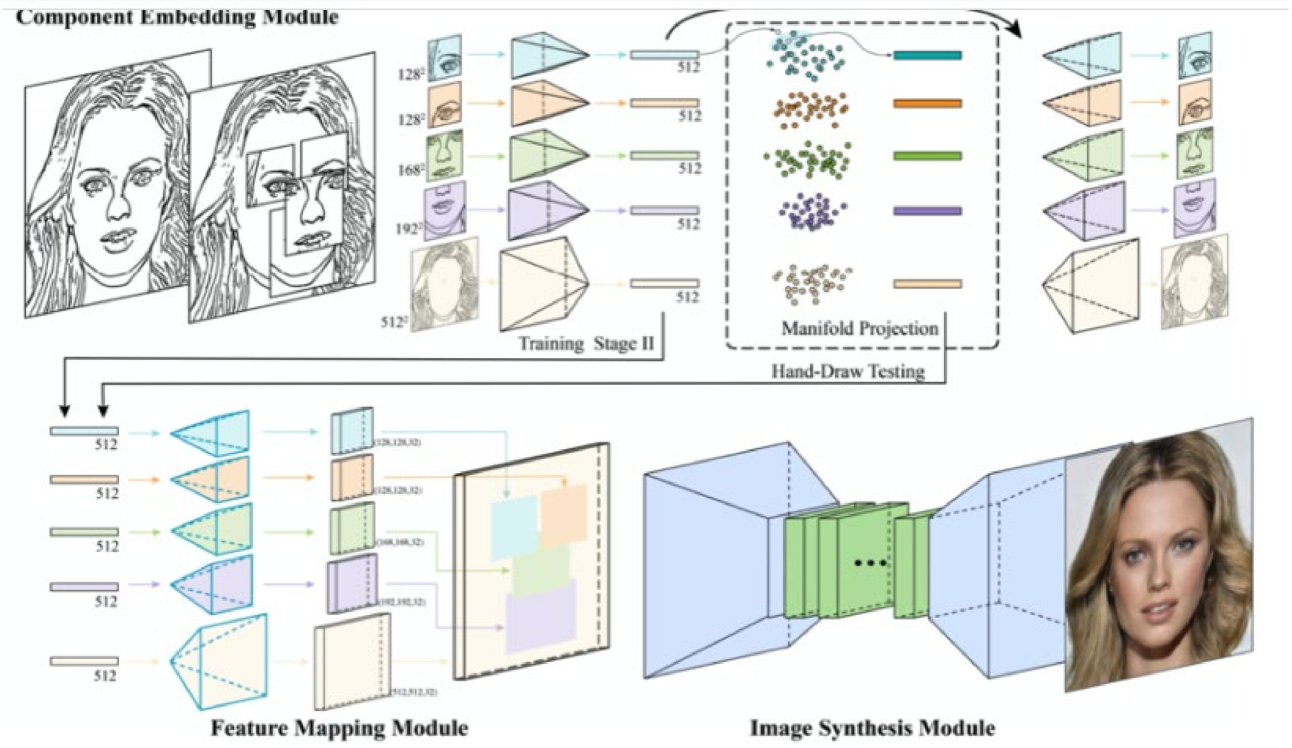


Figure 2.2 The deep learning architecture for face generation based on hand-sketches (Chen et al., 2020).

As shown in Figure 2.2, the model is consisted of three parts, the Component Embedding (CE) Module, Feature Mapping (FM) Module, and Image Synthesis (IS) Module. An input of hand sketch face image has a size of 512 by 512 pixels. This input is first decomposed into five components: "left-eye", "right-eye", "nose", "mouth", and "remainder". The "eye"s, "nose" and "mouth" are separated by taking heat windows size of 128, 168 and 192, while the "remainder" refers to the rest part of the sketch. The five components are then feature-encoded using 5 auto-encoders with latent descriptors of 512 dimensions. The feature vectors of components are considered as the point samples of the underlying component manifolds, and are used to refine the hand-drawn sketch by projecting its individual parts to the corresponding component manifolds using K nearest neighbours.

The programming code is implemented Python [1]. The resulting face appears to be realistic as demonstrated in Figure 2.3 that is originally from the paper by Chen et al. (Chen 2009).

[1] https://github.com/franknb/Drawing-to-Face.

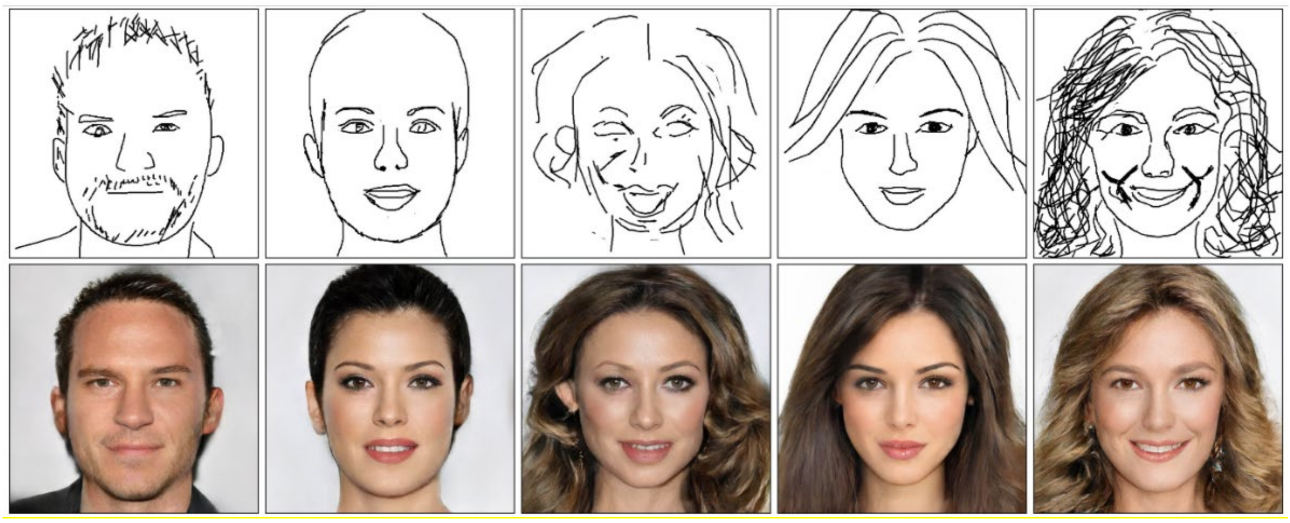


Figure 2.3 The resulting draw of faces (bottom row) from the sketches in the top row from the DL network in Figure 2.2.

3. Methodology

3.1. Siamese Neural Networks

Siamese Neural Networks (SNNs) (Figure 3.1) is a type of neural network which includes multiple instances of the same model; they share the same architecture method and weights. In this type of architecture, we will teach the model with limited data and will display the strength of the network with less available pairs of feed and as we are limited in forensic data source, we don't have a complete and a pair of data in our dataset, like in the one shot learning models.

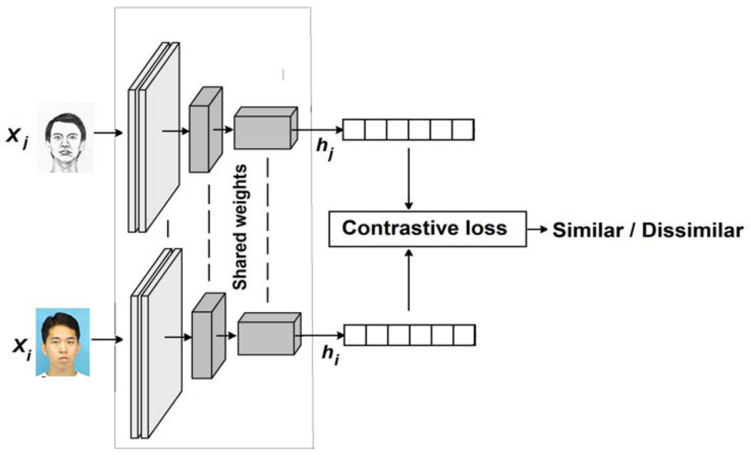


Figure 3.1. Architecture of the SNN

By tradition, a neural network learns to predict over multiple classes. This causes a problem when we need to add or remove new classes to the data. In this case, we must update the neural network and retrain it on the whole dataset.

Additionally, deep neural networks need a large amount of data to train. On the other hand, Siamese Neural Networks SNNs learn with a similarity function. Hence, we can train it to see if two images are the same (which we will do here). This method will let us to classify new classes of data without training the network again.

3.2 Details of Siamese Neural networks

3.2.1 Architecture

Siamese means "exhibit great resemblance" also meaning "connecting two or more pipes or hoses so as to permit discharge in a single stream". Both these definitions perfectly fit Siamese Neural Networks (SNNs). SNNs are multi-stream networks with similar weights on each branch. The output of these parallel pipelines is compared via distance function to estimate similarity between them.

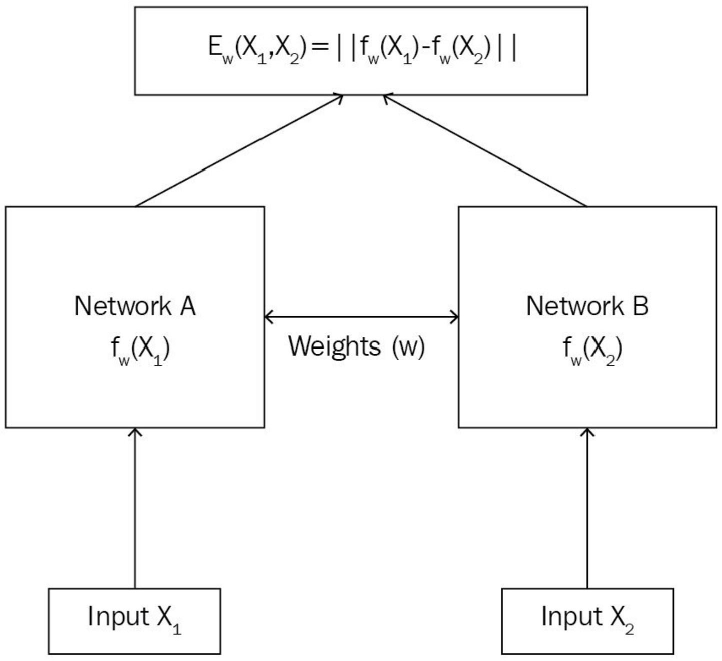


Figure 3.2 The calculation flow of SNN.

Although they are used for various applications, the most common use of SNNs is face recognition. Pictures of faces are compared based on a reference image (or images) to either validate the identity of a certain person or recognise him based on previously defined database images.

Furthermore, these networks are suitable for special problems with very few or only one instance for classification. These problems are known as one-shot learning applications. In that case, we do not have enough training data for each class to be able to train a model to identify them well.

Thus, we deviate our approach from training a network to identify similarity of an instance to a well-known class to identifying the difference between any 2 sub-classes belonging to a well-known major class.

In face recognition for example, the sub-classes are photos of certain people. The major class is face. Thus, we want to train a network to differentiate between photos based on general face characteristics. It may eventually learn to compute the distance between eye examples. This approach also requires changing the traditional training settings.

3.2.2 Loss function (equations)

It should be noted that there are many specific loss functions for training the CNN. We use a function called the contrastive loss function.

If both include entry Xi and Xj are similar, i.e., they go to the same class, then Zij is “0”. If the inputs are dissimilar, i.e., they relate to different classes, then Zij is “1”. So, the training set scan be divided into two subsets: a **similar** set with **Z**ij=”0” and a **dissimilar** set with **Z**ij=”1”.

It is defined as:

$$l(\mathbf{x}_i, \mathbf{x}_j, z_{ij}) = (1 - z_{ij})\|\mathbf{h}_i - \mathbf{h}_j\|_2^2 + z_{ij} max\left(0, \tau - \|\mathbf{h}_i - \mathbf{h}_j\|_2^2\right) \quad (1)$$

where τ is a predefined threshold. Hence, the total error function for minimizing is defined In the usual case,

$$L(W) = \sum_{i,j} l(\mathbf{x}_i, \mathbf{x}_j, z_{ij}) + \mu R(W) \quad (2)$$

where R(**W**) is a regularization term added to improve generalization of the neural network; W is the matrix of the neural subnet parameters; **μ** is a hyper-parameter which controls the strength of the regularization. The above problem is usually solved by using a gradient descent scheme.

We train the neural network based on right values for targets. In that case we have 2 training objectives. The first one is matching multiple instances for a certain sub-class. The second is to differentiate between subclasses. To do this, we use a certain loss function called triple loss. Another simple way of doing this is by giving pairs belonging 2 same sub-class a true label and giving pairs belonging to different ones a false label then treating the problem as a binary classification case. The second case is called conservative loss function and it is the approach we selected for our project.

3.3 Dataset

As humans communicate their ideas through words and writings, visual memories and imaginations can be expressed by drawing and sketching. That is why face sketches have always been used not only for entertainment as a hobby, but also to identify criminals after a painter draws them based on descriptions of eye witnesses. This makes face to sketch matching sometimes a highly critical and sensitive application.

In our project, we aim to construct an SNN to match faces to sketches using CUHK student data set. The Dataset contains 188 pairs, 88 for training and 100 for testing. After loading the images we performed preprocessing and image resizing to prepare them to be fed to the network for training.

3.4 Generating datasets from own colleciton

We tried to experiment and appreciate the power of our model we generated and modified dataset contain of 500 face photo from large dataset called UTKFace [2].

UTKFace dataset is a large-scale face dataset with long age span (range from 0 to 116 years old). The dataset consists of over 20,000 face images with annotations of age, gender, and ethnicity as demonstrated in Figure 3.3.

[2] https://susanqq.github.io/UTKFace/.

Figure 3.3. UTKFace dataset is a large-scale face dataset with long age span

Therefore we extract partial amount of face images '500 images' for our experiment, each face cropped by the dimenssioon of 200x250 pixels, Then we needed our samples become normallised, desaturated and sketch images being created from our original photos.

3.4.1 Analysing the image in histogram

Visualizations are always been an efficient way to represent and explain many statistical details. In image processing histograms are used to represent many aspects regarding the image we are working with.

- Exposure
- Contrast
- Dynamic Range
- Saturation

By visualizing the histogram we can improve the visual presence of an image and also we can find out what type of image processing could have been applied by comparing the histograms of an image.

In this comparison we using SKIMAGE in Python to demostrade one sample image in our dataset.

3.4.2 Coloured Image Histogram Analysis

In color images, we have 3 color channels representing RGB. In Combined Color Histogram the intensity count is the sum of all three color channels.

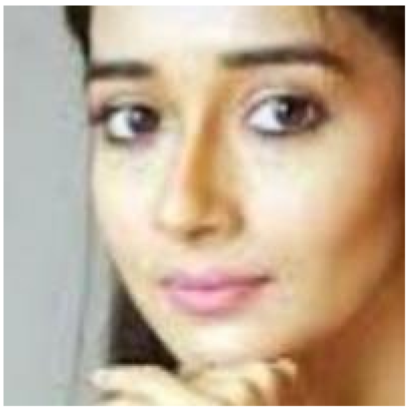


Figure 3.4 Sample image from cropped dataset

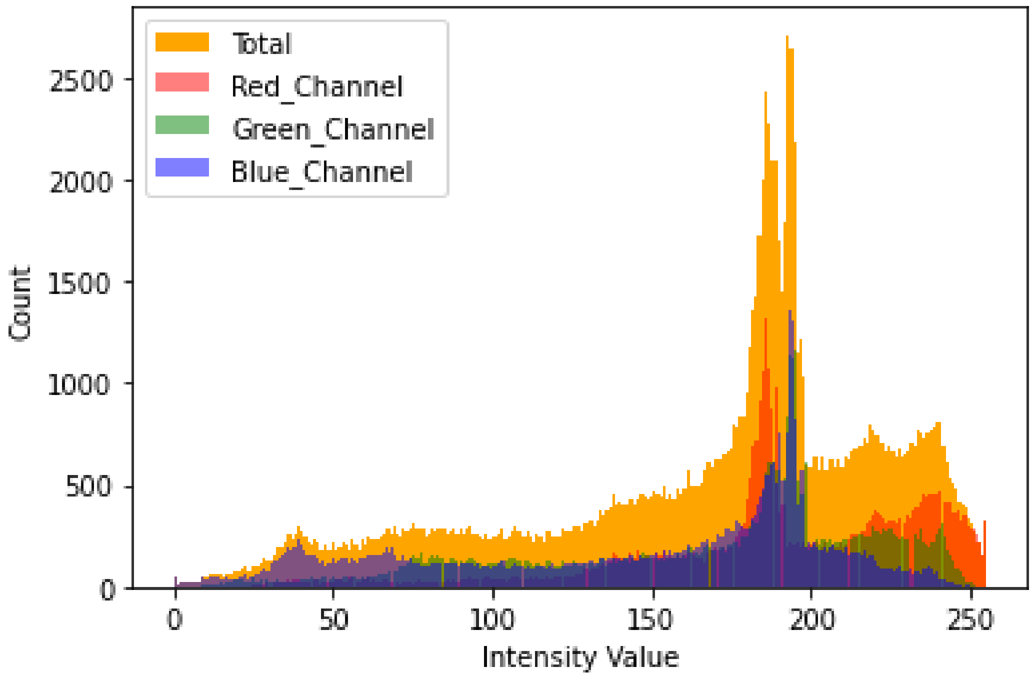


Figure 3.5. Histogram of Coloured image created by Python

```
_ = plt.hist(image.ravel(), bins = 256, color = 'orange', )
_ = plt.hist(image[:, :, 0].ravel(), bins = 256, color = 'red', alpha = 0.5)
_ = plt.hist(image[:, :, 1].ravel(), bins = 256, color = 'Green', alpha = 0.
5)
_ = plt.hist(image[:, :, 2].ravel(), bins = 256, color = 'Blue', alpha = 0.5
)
_ = plt.xlabel('Intensity Value')
_ = plt.ylabel('Count')
_ = plt.legend(['Total', 'Red_Channel', 'Green_Channel', 'Blue_Channel'])
plt.show()
```

3.4.3 Grayscale Histogram Analysis

In the below code, we have loaded the grayscale image of our sample from dataset and generated its histogram using matplotlib. Since the image is stored in the form of a 2D ordered matrix we converted it to a 1D array using the ravel() method.

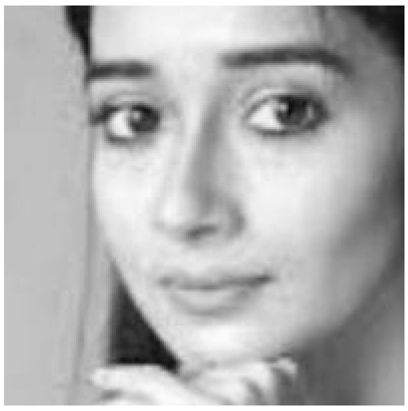


Figure 3.6 Grayscale image from our Dataset

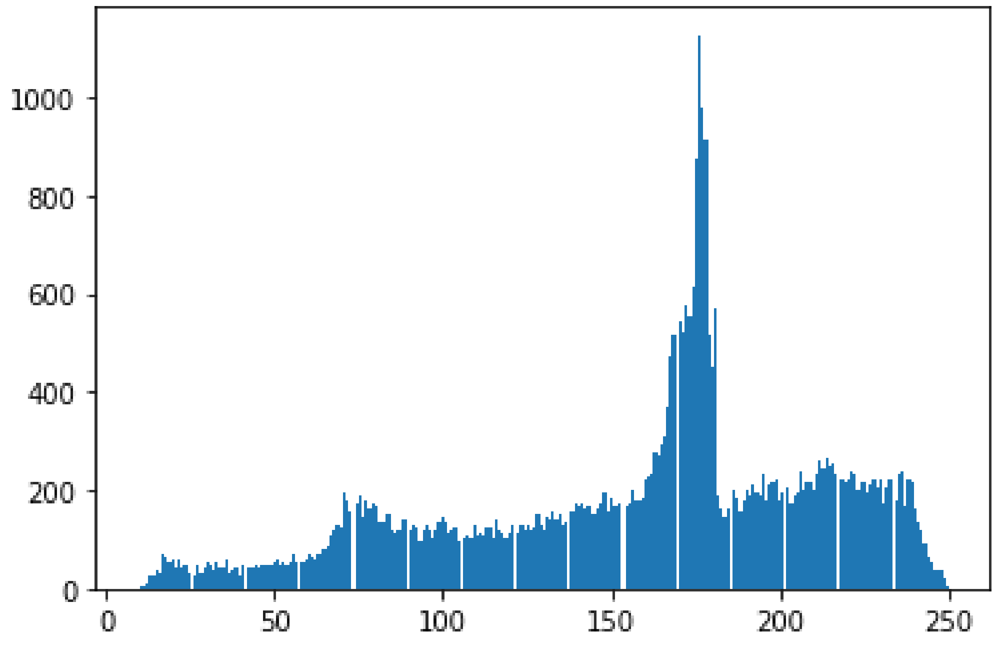


Figure 3.7 Histogram from the image in Figure 3.6.

```
ax = plt.hist(image2.ravel(), bins = 256)
```

Below is complete code from Jupyter in Python

```
from skimage import io
import matplotlib.pyplot as plt
from PIL import Image
```

```
image = io.imread('img2col.jpg')
image2 = io.imread('img1bw.jpg')

img = Image.open('/content/img1bw.jpg')

ax = plt.hist(image2.ravel(), bins = 256)
plt.show()
_ = plt.hist(image.ravel(), bins = 256, color = 'orange', )
_ = plt.hist(image[:, :, 0].ravel(), bins = 256, color = 'red', alpha = 0.5)
_ = plt.hist(image[:, :, 1].ravel(), bins = 256, color = 'Green', alpha = 0.
5)
_ = plt.hist(image[:, :, 2].ravel(), bins = 256, color = 'Blue', alpha = 0.5
)
_ = plt.xlabel('Intensity Value')
_ = plt.ylabel('Count')
_ = plt.legend(['Total', 'Red_Channel', 'Green_Channel', 'Blue_Channel'])
plt.show()
```

4. Results

4.1 Data set creation

Figures 4.1 to 4.3 demonstrate the data that are generated in this work to enlarge our training dataset.

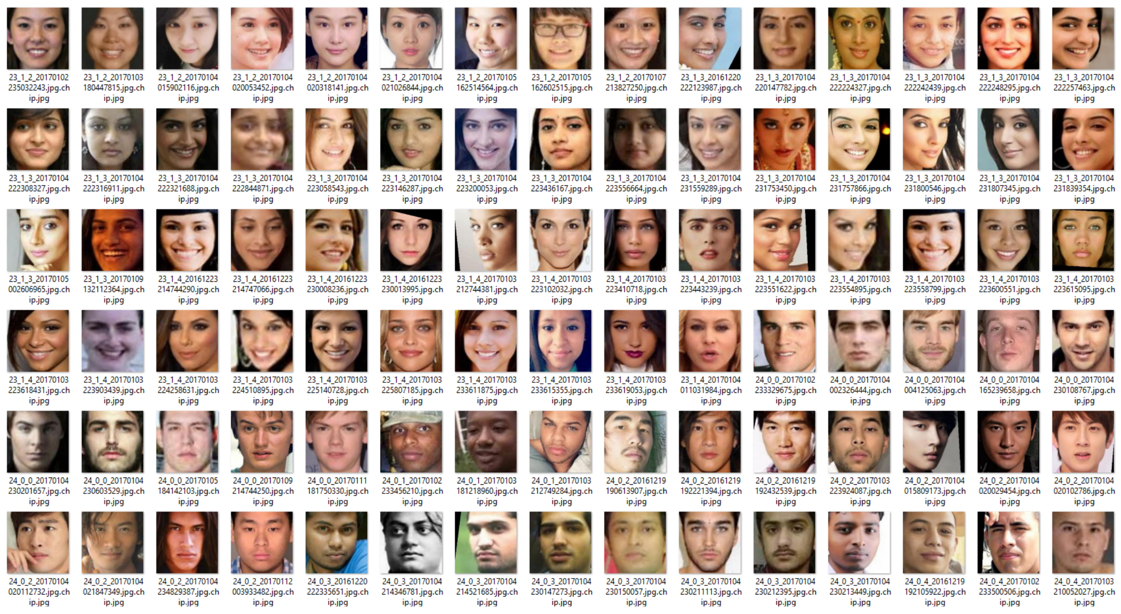


Figure 4.1. Newly generated dateset with 500 images shown in colour.

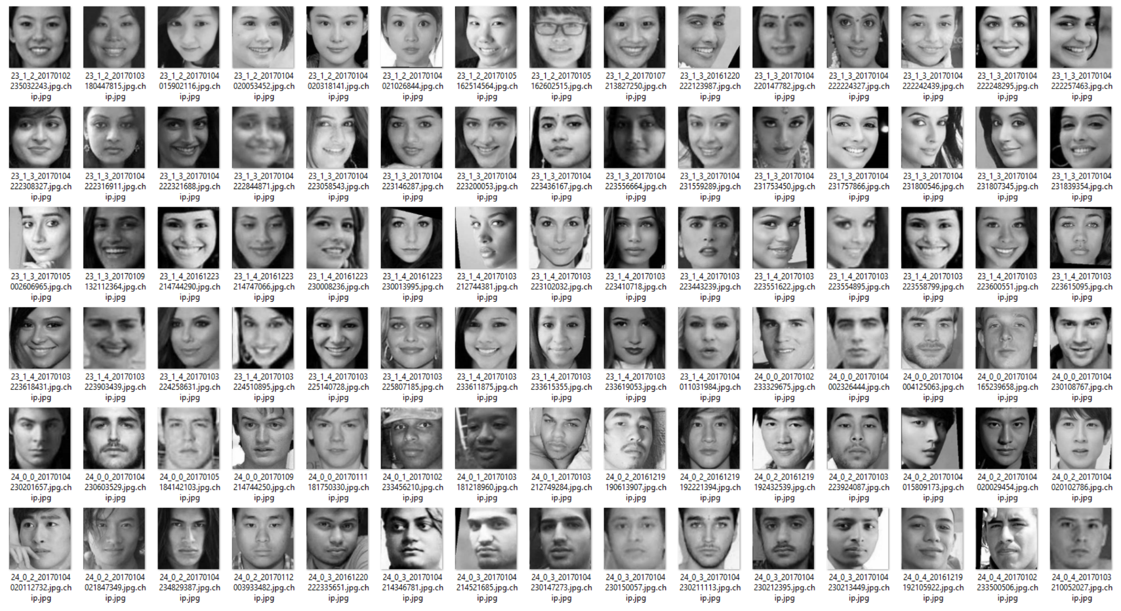


Figure 4.2. The images in Figure 4.1 are desaturated and presente in grayscale.

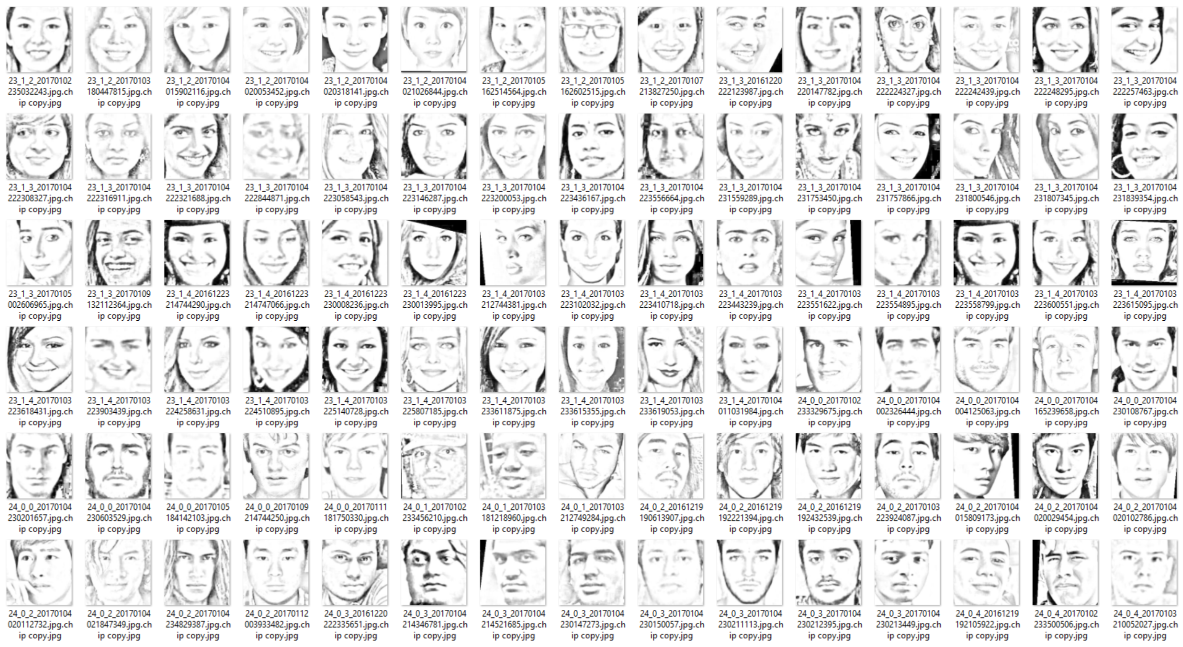


Figure 4.3. The generated sketched images from Figure 4.2.

To generate sketched images, histogram binning technique is applied. Usually, the range of intensity values of images is from [0–255] in 8-bits representation(2^8).

To reduce the bin numbers, we quantise the range into several buckets. In this study, we quantise 0-255 into 8 bins, which contains intensity levels of 0-31, 32-63, 64-95, 96-127, 128-159, 160-191, 192-223, and 224-255. The following Python code illustrates this process with Figure 4.4 displaying a binned example.

And now we plot the histogram of grayscale sketched image again but this time with 8 bins. Which let us compare the raw and reduced noise images in our model.

```
_ = plt.hist(image3.ravel(), bins = 8 )
_ = plt.xlabel('Intensity Value')
_ = plt.ylabel('Count')
plt.show()
```

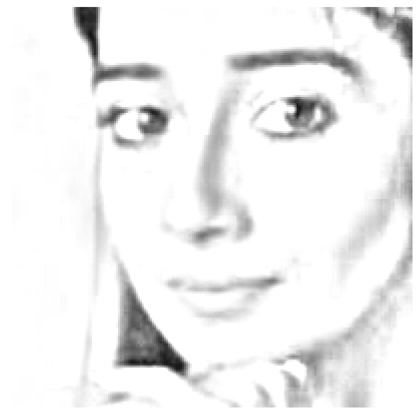


Figure 4.4. An exmaple of sketched image from our dataset

Figure 4.5 illsutrates the histogram of the image shown in Figure 4.4.

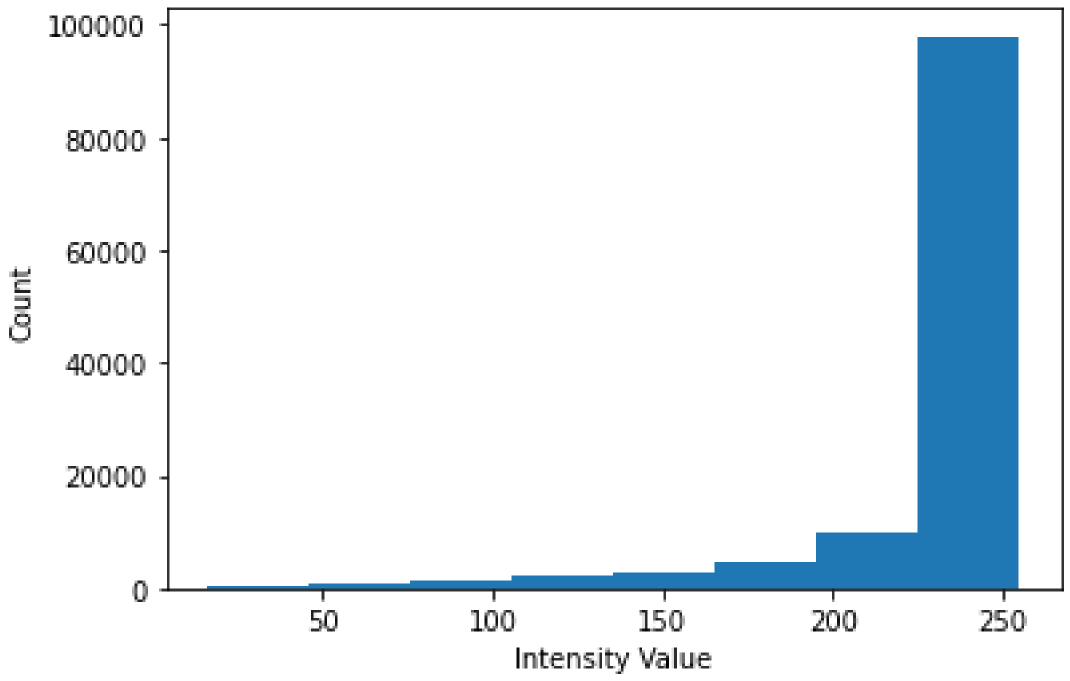


Figure 4.5 Histogram representation of the image in Figure 4.4.

By desaturate the coloured image we remove the unwanted pixel and reduce the noise in our images.

The detailed steps for creating sketch images from coloure photos are given in **Appendix A.**

4.2 The performance of Siamese network

Our model achieved over 96.4% matching performance on the test set. Although that does not guarantee that the model will be robust enough to be used with different dataset which may have other lighting settings for face photos or other styles of drawing. However, using techniques like data augmentation or gathering more data will mitigate the effect of these variations. It is worth mentioning as well, that we used more than half of our dataset for testing which further prove the reliability of our results.

Appendix B provides the programming code for running SNN network.

5. Discussion and future directions

The challenge in this study is to contain sketch data. Drawing by artists is not expensive but time consuming not only for artists but also for subjects who pose for being drawn. Hence in this study, significant number of work was carried out by generating sketch images as larger database will train higher performance systems. In the future, enlargement of database will be one of the works. In addition, the dataset will be more diverse than the currently used one, especially with variations with age, ethnic background and facial expressions.

At present, the following systems are evaluated

1. Photo + Sketch → to system (accuracy of 96.4%) has been done with current model.
2. Photo + Digital-Sketch → to system

The future work will also include to use datasets with pairs of photo and hand-sketches and pairs of photo and digital-generated sketches. Moreover, the systems of from sketch to digital photo then to real photo as explained in Section 2.4 will be evaluated.

Future directions are elaborated in the following sections.

5.1 Enlargement of datasets

For having better recognition accuracy and having a stronger recognition system we will create a large data environment. By developing a new data set to create data sketches and surely conforming with ethical protocols. During the collection subjects are informed by a consent form that explains clearly how their data being used and understanding of the purpose to which the data will be used for.

We will enlarge our dataset by collecting images from:

a) Internet
b) Using forensic or custom artists that are drawing known or unknown people
c) Collection though the computer-generated sketch, using Generative Adversarial neural network or GAN.

As part of our future plan, in this stage, by enlarging the database we will try to use the combination of algorithm of neural networks in SNN to make the network be capable of working in a complex dataset, which contains diversity of human face nations. Not only with one type of normalised faces from one nation but evidently having this large multinational data which will force us to face the greatest challenge in our work which will have more contest towards improving our accuracy result by feeding the customised database.

5.2 Collection from internet

We will use special algorithms to scrape human face through internet, possibly through Google search engine, and we need to normalise it by changing the collected picture in same like poses and sizes by defining and applying the same defaults to be used in our dataset.

Although Google does not take legal action against scraping, nevertheless, Google is using a range of defensive methods that makes scraping another challenging task for us. Google is automatically rejecting User Agents like bots or tools that try to scrape the engine by automated bots.

In our scraper method we will use Google face scraper to collect a cleaned and labelled face dataset, with train face recognition models. We will need a pretrained model for this project to work, which we can use our current trained model. We can either use a text file containing names of use the integrated IMDB (Internet Movie Database) name scraper to make the script fully automatic. In the output folder a text file will be created, containing links to all scraped images and the date they were scraped.

5.3 Collection by Artists Drawing

As part of our database enlargement, we will also use artists drawing which usually are forensic artists, A forensic artist, also commonly referred to as a sketch artist, is a graphic artist that makes free-hand or computerized drawings, enhancements, and reconstructions.

Forensic art is defined as an artistic technique used for identification, apprehension, or conviction purposes. “Forensic artists work closely with law enforcement officers to identify criminal suspects and victims through facial composite sketches. They are also often called to the scene of a crime to create drawings, scale diagrams and models of crime scenes.”

5.4 Collection though computer generated sketch using Generative Adversarial neural network

GANs learn a probability distribution of a dataset by opposing two neural networks against each other. One neural network, called the Generator, generates new data instances, while the other, the Discriminator, evaluates them for authenticity, i.e. the discriminator decides whether each instance of data that it reviews belongs to the actual training dataset or not.

Meanwhile, the generator is creating new, synthetic/fake images that it passes to the discriminator. It does so in the hopes that they, too, will be considered authentic, even though they are fake. The fake image is generated from a 100-dimensional noise (uniform distribution between -1.0 to 1.0) using the inverse of convolution, called transposed convolution.

The goal of the generator is to generate passable images: to remain without being caught. The goal of the discriminator is to identify images coming from the generator as fake.

Here are the steps a GAN takes:

- The generator takes in random numbers and returns an image.
- This generated image is fed into the discriminator alongside a stream of images taken from the actual, ground-truth dataset.
- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

Double feedback loop:

- The discriminator is in a feedback loop with the ground truth of the images, which we know.
- The generator is in a feedback loop with the discriminator.

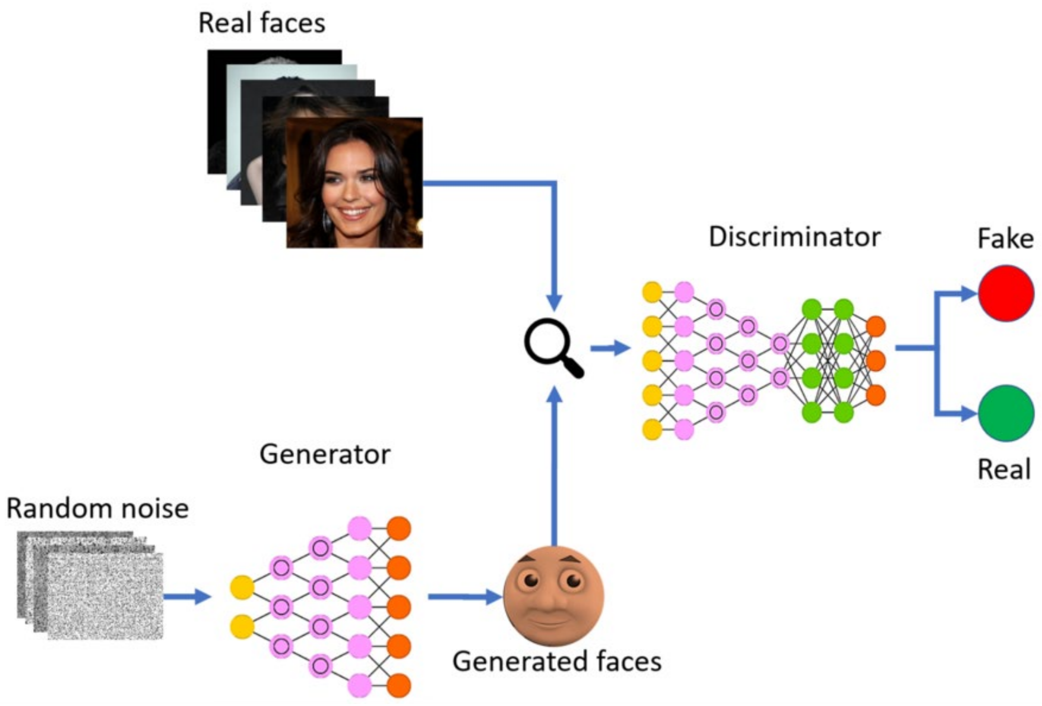


Figure 5: Generative Adversarial Network (GANs)

“In February 2019, graphics hardware manufacturer NVIDIA released open-source code for their photorealistic face generation software Style-GAN. The software uses a generative adversarial network (GAN) approach, in which two neural networks play a game of cat and mouse, one attempting to generate artificial images indistinguishable from real photographs, the other attempting to tell the difference. The two networks train one another; after a few weeks, the image-creating network can produce images like the fakes on this website. Already faces created by Style-GAN are being used in espionage.”

Figure 6: The Style-GAN algorithm synthesizes photorealistic faces such as the examples above.

By having our complex and multi approach dataset has been created we will have a large database which contains approximately 10 or 20 times more faces than our current database.

6. Conclusion

This project investigates the application of Siamese neural network to retrieval photos using hand-made sketches. Based on the collection of 188 pairs of photo/sketches (80% for training and 20% for testing), the system (Appendix B) achieve 96.4% accuracy. Although this performance is considered extremely good, the dataset appears to be too uniformed with all faces are from young university students with neutral expression. Hence in the future, not only large data will be collected, but also these data are with variations of age, ethnic background and facial expressions.

References

Wang, X., & Tang, X. (2009). Face photo-sketch synthesis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(11), 1955-1967.

Tang, X. and Wang, X., 2004. Face sketch recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, *14*(1), pp.50-57.

Klare, B.F. and Jain, A.K., 2013. Heterogeneous face recognition using kernel prototype similarities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(6), pp.1410-1422.

Chen, C.H. ed., 2015. *Handbook of pattern recognition and computer vision*. World Scientific.

Mittal, P., Jain, A., Goswami, G., Singh, R. and Vatsa, M., 2014, September. Recognizing composite sketches with digital face images via SSD dictionary. In *Biometrics (IJCB), 2014 IEEE International Joint Conference on* (pp. 1-6). IEEE.

Klum, S.J., Han, H., Klare, B.F. and Jain, A.K., 2014. The FaceSketchID system: Matching facial composites to mugshots. *IEEE Transactions on Information Forensics and Security*, *9*(12), pp.2248-2263.

Tang, X. and Wang, X., 2003, October. Face sketch synthesis and recognition. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on* (pp. 687-694). IEEE.

Sharma, A. and Jacobs, D.W., 2011, June. Bypassing synthesis: PLS for face recognition with pose, low-resolution and sketch. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (pp. 593-600). IEEE.

Choi, J., Sharma, A., Jacobs, D.W. and Davis, L.S., 2012, June. Data insufficiency in sketch versus photo face recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on* (pp. 1-8). IEEE.

Klum, S., Han, H., Jain, A.K. and Klare, B., 2013, June. Sketch based face recognition: Forensic vs. composite sketches. In *Biometrics (ICB), 2013 International Conference on* (pp. 1-8). IEEE.

Leibo, J.Z., Liao, Q. and Poggio, T., 2014, January. Subtasks of unconstrained face recognition. In *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on* (Vol. 2, pp. 113-121). IEEE.

Bellet, A., Habrard, A., Sebban, M.: A survey on metric learning for feature vectors and structured data. arXiv preprint arXiv:1306.6709 (2013)

Forensic Artists, Website: https://www.crimesceneinvestigatoredu.org/forensic-artist, Retrieved in November 2020

The Style-GAN algorithm synthesizes photorealistic faces such as the examples above. Figure is from Karras et al. (2018).

Artificial Intelligence for Solving Image Processing - https://www.apriorit.com/

How does GANs work: https://medium.com/sigmoid/a-brief-introduction-to-gans-and-how-to-code-them-2620ee465c30, Retrieved in November 2020

CUHK Face Sketch Database (CUFS): http://mmlab.ie.cuhk.edu.hk/archive/facesketch.html. Retrieved in May 2017.

UTKFace - Large Scale Face Dataset - https://susanqq.github.io/UTKFace/

Chen S., Su W., Gao L., Xia S., Fu H., (2020) DeepFaceDrawing: Deep Generation of Face Images from Sketches, http://www.geometrylearning.com/paper/DeepFaceDrawing.pdf.

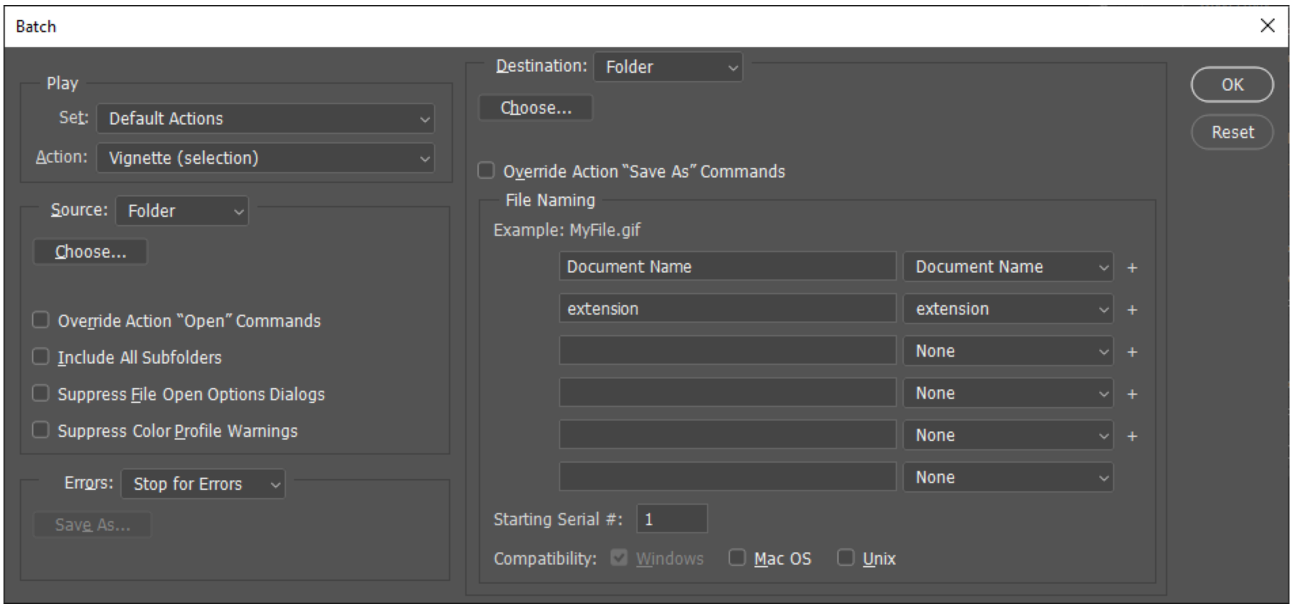
Appendix A. Creation of sketch images from colour photos using Adobe Photoshop

For this research we using Adobe Photoshop we created 500 desaturated images; first we have created a automation action program in Photoshop by following steps,

The following steps detail the process to convert proocessed images into sketch images.

A.1 Coverting colour to gray scale.

1- We open the first image we want to convert from color to black and white.
2- Select "Window" from the main menu, then select "Actions" to display the Actions window.
3- We should create a new action from right of the Actions tab to display the Actions sub-menu, then click on the "New Action".
4- In the New Action dialogue box, name the action "Color to Grayscale" and press the "Record" button.
5- Select "Image," choose "Mode" and click "Grayscale" from the main menu. Click the "Discard" button if a dialogue box appears asking if you want to discard the color information.
6- Then we close the window and save the file.
7- Now press the "Stop" icon at the bottom right corner of the Actions window.
8- Select "File," choose "Automate" then click "Batch" from the menu. From the Image Processor dialogue box, click the "Sorce Folder" button to select the source folder containing the photo images you want to convert to black and white."


Batch

Play
Set: Default Actions
Action: Vignette (selection)

Source: Folder
Choose...
☐ Override Action "Open" Commands
☐ Include All Subfolders
☐ Suppress File Open Options Dialogs
☐ Suppress Color Profile Warnings

Errors: Stop for Errors
Save As...

Destination: Folder
Choose...
☐ Override Action "Save As" Commands
File Naming
Example: MyFile.gif

Document Name	Document Name
extension	extension
	None
	None
	None
	None

Starting Serial #: 1
Compatibility: ☑ Windows ☐ Mac OS ☐ Unix

OK
Reset

9- Select the destination folder where you want the converted images stored, press ok.

10- Check the "Run Action" box in the Preferences section, then make sure "Default Actions" is selected from the drop-down menu. Choose the "Colour to Grayscale" action you created in Step 3 from the next drop-down menu.

11- Click the "Run" button to begin the batch image conversion. Photoshop will now convert all the images in the source folder you designated to black and white images. These converted images will be placed in the destination you selected in Step 8.

We have desatuared our photos.

A.2 Creating sketch images

Now we have desaturated our 500 photos and should create sketch photos from our black and white photos. Again we use Adobe Photoshop for this proccess. By following steps we are converting them to sketch.

1- First we open one of the photos in Adobe Photoshop
2- We dublicate the Background Layer and create new layer
3- We Desatuare the new created layer, by cliking on Adjustments and choose Desaturate.
4- We dublicate the new desaturated layer and Invert the new one by going to layer Adjustments and choose Invert.
5- Now we change The Blend Mode To Color Dodge, by going to Blend mode and choose Colour Doge
6- We need to convert the current layer to Smart Object, then we Apply The Gaussian Blur Filter by going to Filter menu in the menu bar, we select the blur radius to 10 or 12.

We have our photo being converted to Sketch, by using Automation Action in Adobe Photoshop which we where expalined in our “Grayscale conversion” section we make all 500 images to Sktech automatically.

Appendix B. Programming code for implementation of Siamese network

1. Clone the Dataset Repository

```
!git clone https://github.com/mhsnuk/sketched-face.git
```

2. Imports

```
import os
import numpy as np
from PIL import Image
from matplotlib.pyplot import imread
import matplotlib.pyplot as plt
import cv2
import random
import tensorflow as tf
from keras.models import Sequential
from keras.optimizers import Adam
from keras.layers import Conv2D , ZeroPadding2D , Activation , Input ,
concatenate , Concatenate
from keras.models import Model
from keras.layers.normalization import BatchNormalization
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import Concatenate
from keras.layers.core import Lambda , Flatten , Dense
from keras.initializers import glorot_uniform
from keras.engine.topology import Layer
from keras.regularizers import l2
from keras import backend as K
from numpy.random import seed
```

3. Seeding numpy and TensorFlow

```
seed ( 1 )
tf.random.set_seed ( 1 )
```

4. Load and Resize the Images

```
def load_imgs ( path ) :
   x= []
   y= []
   #There are two folders, originals images(photos) and sketches
   for folder in sorted ( os.listdir ( path )):
     if ( folder == "photos" ):
       print ( "Loading original photos!" )
     elif ( folder == "sketches" ):
       print ( "Loading sketches photos!" )
       photos_path = os.path.join ( path , folder )
     if os.path.isdir ( photos_path ):
       #Now loop over the images/sketches.
         for image_path in sorted ( os.listdir ( photos_path )):
             if ( image_path == ".DS_Store" ):
                continue
             image_path = os.path.join ( photos_path , image_path )
             if ( folder == "photos" ):
             image =
np.array ( Image. open ( image_path ) .convert ( 'LA' ))
             image = cv2.resize ( image , dsize= ( 100 , 150 ),
interpolation=cv2.INTER_CUBIC )
             x.append ( image )
         elif ( folder == "sketches" ):
             image =
np.array ( Image. open ( image_path ) .convert ( 'LA' ))
             image = cv2.resize ( image , dsize= ( 100 , 150 ),
interpolation=cv2.INTER_CUBIC )
             y.append ( image )
  return x , y
```

5. Create a batch of the training dataset

```
ef get_batch ( batch_size , dataset_size , h , w ) :
pairs = np.zeros (( 2 , batch_size , h , w , 1 ))
targets = [ 1 ]
num1 = random.randint ( 0 , dataset_size -1 )
BASE_IMG = num1
pairs [ 0 , 0 ,:,:, 0 ] = x [ num1 ][:,:, 0 ]
pairs [ 1 , 0 ,:,:, 0 ] = y [ num1 ][:,:, 0 ]
for i in range ( 1 , batch_size ):
   num1 = random.randint ( 0 , dataset_size -1 )
   if ( num1 == BASE_IMG ):
     num1 = num1 - 1
     pairs [ 0 , i ,:,:, 0 ] = x [ num1 ][:,:, 0 ]
     pairs [ 1 , i ,:,:, 0 ] = y [ BASE_IMG ][:,:, 0 ]
     targets.append ( 0 )
return pairs , targets
```

6. Build the siamese Network using Keras

```
def get_siamese_model ( input_shape ) :

     """
     Model architecture
     """

     # Define the tensors for the two input images
     left_input = Input ( input_shape )
     right_input = Input ( input_shape )
     # Convolutional Neural Network
     model = Sequential ()
     model.add ( Conv2D ( 64 , ( 10 , 10 ), activation= 'relu' ,
     input_shape=input_shape , kernel_regularizer=l2 ( 2e-4 )))
     model.add ( MaxPooling2D ())
     model.add ( Conv2D ( 64 , ( 7 , 7 ), activation= 'relu' ,
     kernel_regularizer=l2 ( 2e-4 )))
     model.add ( MaxPooling2D ())
     model.add ( Conv2D ( 128 , ( 4 , 4 ), activation= 'relu' ,
     kernel_regularizer=l2 ( 2e-4 )))
     model.add ( MaxPooling2D ())
     model.add ( Conv2D ( 256 , ( 4 , 4 ), activation= 'relu' ,
     kernel_regularizer=l2 ( 2e-4 )))
     model.add ( Flatten ())
     model.add ( Dense ( 4096 , activation= 'sigmoid' ,
     kernel_regularizer=l2 ( 1e-3 ),))
# Generate the encodings (feature vectors) for the two images
encoded_l = model ( left_input )
encoded_r = model ( right_input )
# Add a customized layer to compute the absolute difference
between the encodings
```

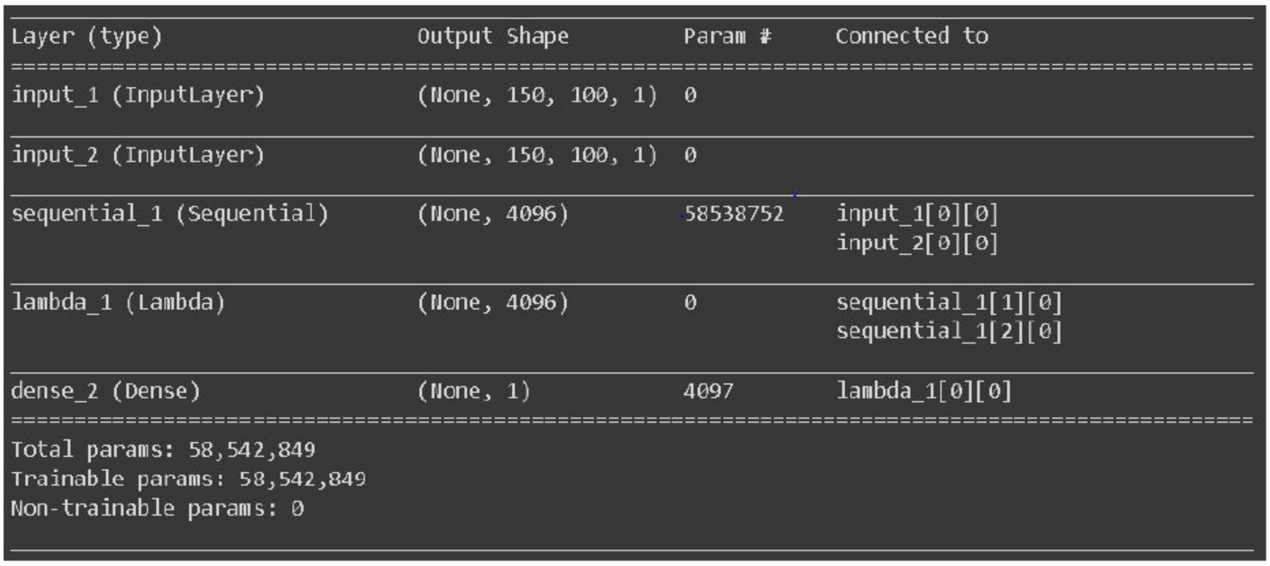
```
L1_layer = Lambda ( lambda tensors : K. abs ( tensors [ 0 ] - tensors [ 1
]))
L1_distance = L1_layer ([ encoded_l , encoded_r ])
# Add a dense layer with a sigmoid unit to generate the
similarity score

prediction = Dense ( 1 , activation= 'sigmoid' )( L1_distance )
# Connect the inputs with the outputs
siamese_net =
Model ( inputs= [ left_input , right_input ], outputs=prediction )
# return the model
return siamese_net
```

7. Set the learning rate, define the optimizer and compile the model

```
lr = 0.001
optimizer = Adam ( lr = lr )
model. compile ( loss= "binary_crossentropy" , optimizer=optimizer )
```

8. Model Architecture



```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            (None, 150, 100, 1)  0
__________________________________________________________________________________________________
input_2 (InputLayer)            (None, 150, 100, 1)  0
__________________________________________________________________________________________________
sequential_1 (Sequential)       (None, 4096)         58538752    input_1[0][0]
                                                                 input_2[0][0]
__________________________________________________________________________________________________
lambda_1 (Lambda)               (None, 4096)         0           sequential_1[1][0]
                                                                 sequential_1[2][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 1)            4097        lambda_1[0][0]
==================================================================================================
Total params: 58,542,849
Trainable params: 58,542,849
Non-trainable params: 0
__________________________________________________________________________________________________
```

9. Train the Model applying both earlystoping and reduce learning rate on plateau

```
for i in range ( 1 , n_iter+ 1 ):
      ( inputs , targets ) = get_batch ( batch_size , train_data_set_size
      ,
      150 , 100 )
      # plt.imshow(inputs[0,1,:,:,0],cmap='gray')
      # show()
      # plt.imshow(inputs[1,1,:,:,0],cmap='gray')
      # show()
      # print(targets)
      inputs = list ( inputs )
      loss = model.train_on_batch ( inputs , targets )
      print ( "Epoch: {0} Train Loss: {1}" . format ( i , loss ))
      if loss < best_loss :
         best_loss = loss
      if i % evaluate_every == 0 :
         print ( "\n ------------- \n" )
```

```
          print ( "Time for {0} iterations: {1} mins" . format ( i ,
          ( time.time () -t_start ) / 60.0 ))
          print ( "Train Loss: {0}" . format ( loss ))
          val_acc = test_oneshot ( model , N_way , n_val , h , w , True ,
          test_data_size )
       if val_acc >= best :
          print ( "Current best: {0}, previous best: {1}
          \n" . format ( val_acc , best ))
          model.save_weights ( os.path.join ( 'weights.{}.h5' . format ( i
       )))
          best = val_acc
       if best_loss == best_loss_old :
          lr = lr* 0.5
       if lr < 1e-06 :
          break
          print ( 'reducting lr to ' , lr )
          K.set_value ( model.optimizer.lr , lr )
          best_loss_old = best_loss
```

10. Loading and resizing the test images

```
def load_test_imgs ( path ) :
    x= []
    y= []
#There are two folders, originals images(photos) and sketches
  for folder in sorted ( os.listdir ( path )):
   if ( folder == "photos" ):
    print ( "Loading original photos!" )
   elif ( folder == "sketches" ):
    print ( "Loading sketches photos!" )
    photos_path = os.path.join ( path , folder )
      if os.path.isdir ( photos_path ):
#Now loop over the images/sketches.
        for image_path in sorted ( os.listdir ( photos_path )):
            image_path = os.path.join ( photos_path , image_path )
              if ( folder == "photos" ):
                 image =
np.array ( Image. open ( image_path ) .convert ( 'LA' ))
                 image = cv2.resize ( image , dsize= ( 100 , 150 ),
interpolation=cv2.INTER_CUBIC )
                 x.append ( image )
              elif ( folder == "sketches" ):
                 image =
                 np.array ( Image. open ( image_path ) .convert ( 'LA' ))
                 image = cv2.resize ( image , dsize= ( 100 , 150 ),
         interpolation=cv2.INTER_CUBIC )
              y.append ( image )
            return x , y
```

11. Make one shot learning test batches

```
def make_oneshot_task ( N , h , w , test_data_size ) :
            pairs = np.zeros (( 2 , N , h , w , 1 ))
            targets = [ 1 ]
            num1 = random.randint ( 0 , test_data_size -1 )
            BASE_IMG = num1
            pairs [ 0 , 0 ,:,:, 0 ] = x_test [ num1 ][:,:, 0 ]
            pairs [ 1 , 0 ,:,:, 0 ] = y_test [ num1 ][:,:, 0 ]
    for i in range ( 1 , N ):
       num1 = random.randint ( 0 , test_data_size -1 )
             if ( num1 == BASE_IMG ):
              num1 = num1 - 1
              pairs [ 0 , i ,:,:, 0 ] = x_test [ num1 ][:,:, 0 ]
              pairs [ 1 , i ,:,:, 0 ] = y_test [ BASE_IMG ][:,:, 0 ]
              targets.append ( 0 )
             return pairs , targets
```

12. Test the model on the test set and get accuracy

```
def test_oneshot ( model , N , k , h , w , verbose , test_data_size ) :
      """Test average N way oneshot learning accuracy of a siamese 
neural net over k one-shot tasks"""
       n_correct = 0
          if verbose :
            print ( "Evaluating model on {} random {} way one-shot 
learning tasks ... \n" . format ( k , N ))
         for i in range ( k ):
            inputs , targets = make_oneshot_task ( N , h , w , 
test_data_size )
# plt.imshow(inputs[0,1,:,:,0],cmap='gray')
#  show()
# plt.imshow(inputs[1,1,:,:,0],cmap='gray')
#  show()
        inputs = list ( inputs )
        probs = model.predict ( inputs )
        probs = [ i [ 0 ] for i in probs ]
# print('predicted:', np.argmax(probs))
# print('Ground Truth:',np.argmax(targets))
        if np.argmax ( probs ) == np.argmax ( targets ):
           n_correct+= 1
           percent_correct = ( 100.0 * n_correct / ( i+ 1 ))
           print ( "Accuracy = {0}% [{1}/{2}] \n" . format ( 
percent_correct ,
           n_correct , i+ 1 ))
percent_correct = ( 100.0 * n_correct / k )
          if verbose :
            print ( "Got an average of {}% {} way one-shot learning 
accuracy \n" . format ( percent_correct , N ))
   return percent_correct
```