# Sketch Inversion and Colorization of Gray-scale Photos and Videos using Convolutional Learning

Rejin Joy*
Electrical and Computer Engineering
University of Florida
Email:rejinjoy18@ufl.edu

Rishabh Singh*
Electrical and Computer Engineering
University of Florida
Email:rish283@ufl.edu

***Abstract*—In this paper, we present a novel approach to construct a photo-realistic version of hand-drawn sketches of human faces using deep Convolutional Neural Networks. Additionally, we also propose to use our trained network for the purpose of colorization of gray-scale images and videos. We first generate the training dataset by producing a variety of sketches of actual photos using existing filters. We propose to use the CelebA dataset containing face images varying extensively in terms of pose, expression, illumination and background clutter. After training and suitable selection of hyper-parameters, we test the network on both hand-drawn and automatically generated sketches. We also test our network in the application of colorization of gray-scale photos and videos and find it to be very efficient in the application. Apart from visual examination, we quantitatively measure the physical, perceptual, and correlational performance of the network using parameters such as Peak Signal to Noise Ratio (PSNR) and structural similarity (SSIM). This project could be extensively useful in the field of Forensic analysis, modern art and gray-scale photo/video colorization applications.**

***Index Terms*—Convolutional Neural Networks, photo-realistic, correlational, PSNR, SSIM, Haar Cascades, Colorization**

## I. Introduction

FORENSIC science is an indispensible tool used by law enforcement to solve crimes. More than often, hand-drawn portraits of suspects as described by eye witnesses on the scene of the crime are the only evidence available. It is vital to maximize the amount of information gained from these sketches. Sketch inversion is a novel application of machine learning to aid with this complicated process, where photo-realistic images of the sketches are generated to help with easy and quick identification of the suspect. It is also useful from the art-history perspective, to recreate self-portraits of artists and other famous personalities from earlier generations.

The use of Deep Convolutional Networks in image recognition and transformation tasks has gathered significant popularity in the computer vision community. In [1], authors perform sketch  photo inversion by using a deep convolutional network after training it with a constructed dataset consisting of a sizable number of differently styled face sketches and their corresponding photos. For implementing the synthesis of photo-sketch pairs and their recognition, authors in [2] use a novel variant of Markov Random Fields (MRFs). Authors in [3] have used a sparse representation based algorithm to implement the synthesis of sketch - photo pairs and image retrieval. The generation of complete photo - sketch pairs via fully convolutional representation learning (using only convolutional layers and no other type) from one end to another has been implemented to convert photos to their respective pencil sketches in [4]. Authors in [5] have used a sketch  photo synthesis method on support vector regression as an endeavor to retain some vital and important image details. CNNs have also been applied to produce a pixel-wise output for the purpose of generating images with extreme resolutions in [6]. Furthermore, face-sketch recognition has been performed in the past using subspace combination method and a correlation filter which is tolerant to radiance [7]. We attempt to explore and utilize these state-of-the art methodologies in Deep Neural Network learning to achieve our goal of sketch-photo conversion.

April 18, 2017

## II. Background

### A. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are variants of MLPs inspired from the nature and environment where the connectivity between neurons is quite similar to the structure of the visual cortex of animals [8]. A convolutional neural network, in most of the cases, consists of three layers which are the convolutional layers, layers with max-pooling and the layers that are fully connected. Convolutional layers consist of a rectangular framework which can be thought of as a set of learnable filters. During forward pass, each filter is convolved across the width and height of the input image, and each filter activates when it sees a different kind of visual feature (edges, corners etc.). Each neuron takes inputs from a rectangular portion or mesh of the previous layer and all of the neurons in this rectangular section of the convolutional layer have the same weights. The convolutional layer can therefore be thought to be as just an image of a section of the previous layer, where the weights specify the convolution filter. Each filter produces a separate 2 dimensional activation map which

* Both authors have contributed equally to this work

are stacked along the depth dimension. The output volume size is given by the following formula:

$$[(W - F + 2P)/S] + 1$$

Here,

$$W = inputsize,$$

$$F = filtersize,$$

$$P = zeropadding,$$

$$S = stridesize$$

The pooling layer after the convolutional layer does the subsampling of small rectangular blocks from the convolutional layer to produce a single output from the block. Max-pooling subsamples the largest or maximum possible size of the rectangular block. The fully connected layers are typically and mostly placed after the convolutional and pooling layers and each neuron in a fully connected layer is connected to all of the neurons of the previous layer which is a necessary condition for full connection.

Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with least amount of preprocessing. They can recognize very diverse multiple patterns (such as handwritten characters), and are immune to distortions and simple geometric changes [9].

### *B. Performance Metrics*

For testing our deep learning model, we propose to use some specific performance metrics relevant to our application. These metrics are described as follows:

- Peak Signal to Noise Ratio (PSNR): PSNR is one of the popular metrics to measure the physical quality of any given image. It can be defined as the ratio of the peak power of the image and the power of the noise in the image:

  $$PSNR = 10log10(R^2/MSE)$$

  Here R is the maximum fluctuation in the input image data type.
- Structural Similarity (SSIM): SSIM is an important metric to get an idea about the perceptual nature of an image. It is defined as the multiplicative mix of the resemblance of an image with the reference image in terms of contrast, luminance and structure.

## III. Related Work

Authors in [1] develop a novel and effective deep CNN architecture of obtaining realistic photos from sketches where they use residual and deconvolution layers in addition to using convolutional layers at the beginning and at the end. They perform batch normalization with decay at each layer. Authors here first pre-process the data obtained from various databases (CelebA, FERRET) to make them centered and resized to a fixed resolution. They generate the training data by converting the photos from the dataset to grayscale, line, and color sketches. Finally, they train their network using the generated training images and use various performance metrics (PSNR, SSIM) to evaluate their results on the test set. We propose to use a similar neural network architecture where convolutional and deconvolution layers are used. To reduce the number of parameters used and reduce the computational complexity, we do not use residual blocks. Even without the usage of residual blocks and training on a smaller network, we are able to match the performance obtained by the authors in [**?**, 1] We also successfully extend the application of the same network to colorize gray-scale images and videos.

[2] uses a Markov Random Fields model to produce both sketch-photo and photo-sketch conversions. However, this paper only works on the assumption that the faces have to be studied in a frontal posture, with same lighting and minimal obstruction, which is unrealistic in real world applications. To synthesize sketch/photo images, the face region is divided into overlying patches during training. The size of the patches will decide which part of local face features will be learned. From a training set which contains photo-sketch pairs, the joint photo-sketch model is learned at different scales using an MRF model. In comparison to deep learning methods, this model requires you to explicitly mention the dictionaries and patch sizes which lead to large amounts of pre/post processing.

[3] also uses patch based methods, similar to the above method, and is composed of two main steps: sparse neighbor selection (SNS) for an initial approximation of the pseudo-image and sparse-representation-based enhancement (SRE) for further refining the quality of the generated image. Compared to the SR method, the method proposed in [2] (MRFs) is subjected to some distortion especially for the mouth, beard and chin area. This may be due to the fact that if there is no patch with the same structure in the training dataset, the MRF method selects any prevailing patch from the training dataset, which may lead to deformations.

[4] uses an end-to-end model called Fully Convolutional Network (FCN) which only consists of convolutional layers to convert photos to their respective sketches. This deals with the limitation of scalability, and computational complexity of the previously mentioned patch based methods by using an end-to-end model. To handle non-linearity of the photos and sketches, their input data contains two additional channels apart from the RGB channels, of corresponding XY coordinates of patches. This model however has the following limitations: The generated sketches do not have very sharp edges and clear curves. Secondly, the database involved in their study only contains Asian face images, which may limit the model from generalizing well to test images of faces from other races.

[6] develops an efficient and computationally cheap deep CNN for the application of super-resolution of images. Authors propose a network of fully convolutional networks which learns mapping between low resolution images and their high

Fig. 1: Row 1: Face Images, Row2: Corresponding Sketches Generated

resolution versions. Authors in [5] first extract patches from a low resolution image and characterize them using a vector. These vectors obtained represent the feature maps of the input. The authors in [5] then perform non-linear mapping to transform each feature vector to another high dimensional vector in the form of a dictionary that represents the high resolution image. These patches are then pooled together to get the final high-resolution version of the image. We propose to use a similar method of using convolutional filters to obtain feature maps initially followed by patch wise non-linear transformation (colorization) and finally reconstruction of all the patches to obtain the photo-realistic image.

Authors in [5] use an alternate method of Support Vector Regression to convert hand sketches to realistic photos. This method is similar to previous methods where patches or feature maps are first obtained from the input image. However, instead of conventional non-linear transformations, an SVR model is generated for each patch through the nearest neighbour method to obtain high frequency information of the image. This information obtained from all images is finally summed up to obtain the photo realistic version.

## IV. Methodology

The following steps were followed to implement this work:

- Generation of training dataset, validation set and test set by using image processing techniques to convert a large number of actual photos from the CelebA database into line and gray-scale sketches.
- Preprocessing of the images to detect faces in images using Haar Cascades and AdaBoosting.
- Designing of the optimum deep convolutional neural network architecture.
- Training of the network on an external GPU.
- Testing of network on the test set for inverting sketches into their photo realistic versions.
- Testing of network to convert black and white images to their colored versions.
- Testing of network to convert black and white videos to their colored versions.
- Development of Kivy graphical user interface (GUI).
- Evaluation of the network using relevant performance metrics and comparison of performance with existing models.
- Exploration of future scope of our project.

## V. Generation of Training Set

We generate the training set using the following procedure:

- Conversion of the RGB colored image to grayscale
- Inversion of the grayscale image to get a negative
- Application of Gaussian blur to the generated negative
- Blending of the generated grayscale with the negative

We use the OpenCV library of python to perform the above steps on the images. This creates gray pencil sketch versions of the colored images as shown in Figure 1. We use these generated sketches as the input to our network. We then use this training set to train our network. We use the CelebA database of facial images to generate our training set.

## VI. Deep Learning Network Architecture

The deep neural network architecture depicted in Table 1 was used to implement this work. Our architecture consists of 8 layers. The first five layers are convolutional layers followed by 2 deconvolutional layers, and a convolutional layer as the final layer. The ReLU activation function is used because

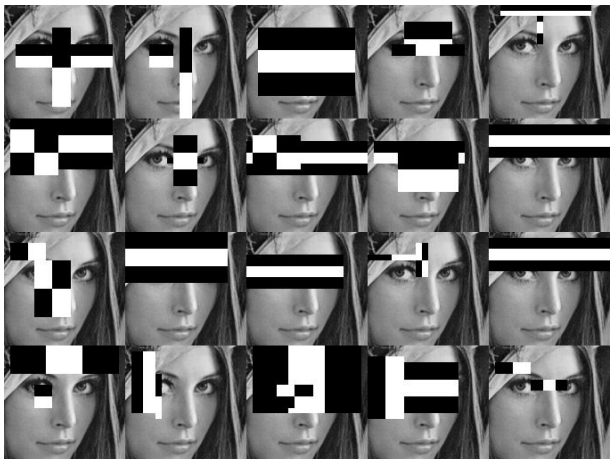

Fig. 2: Haar Cascade Classifier

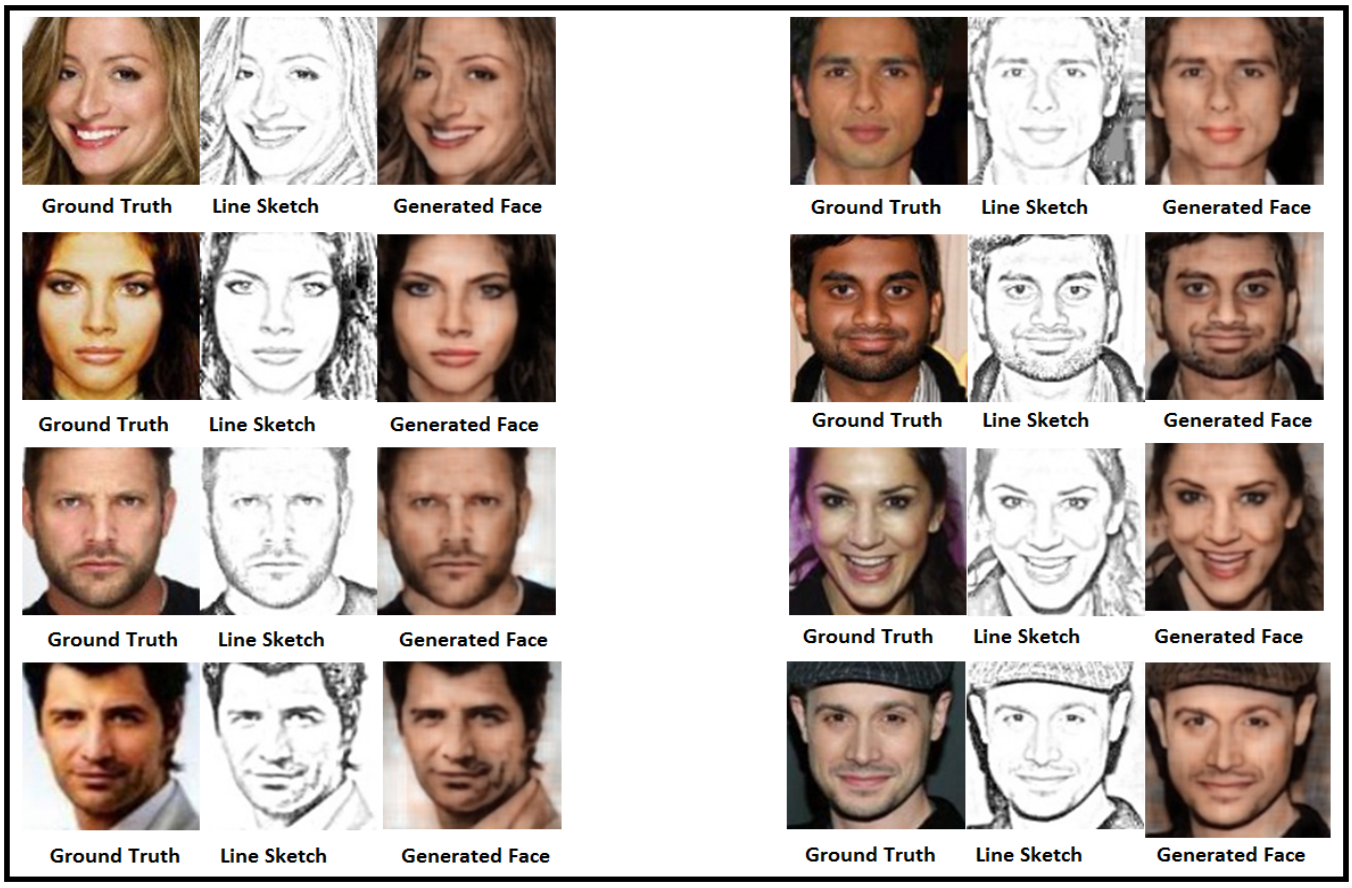


Fig. 3: Comparison of Network Output Images with respect to Ground Truth

of its advantage in sparsity and reducing the likelihood of vanishing gradients. We use the MSE (Mean Square Error) loss function which is more popular than categorical cross-entropy in computer vision or image related deep learning. The Adam optimizer is used to calculate gradients and minimize the mean square loss between pixel values of the output and target images.

## VII. Preprocessing of Data

The CelebA dataset include face images with a cluttered and noisy background. A large number of images in the dataset have more than only the face present in the image. Since feeding these raw images to our network would lead to poor results, we use an image processing technique to detect face images in the network and crop out the raw images to contain only face images. The Haar Cascade classifier was used for this purpose. Haar Cascades work by moving a window of a particular kernel size over the input images, and learning Haar-like features to separate non-objects from objects. A false positive rate threshold is set, and a cascade of classifiers is added to a particular region until that threshold is met, or until the area is declared as a non-face area when any one of the features fail in a window region. Figure 2 depicts how a Haar cascade classifier works [10].

Once the images have been cropped and aligned to only show face images, they are resized to a uniform size of 100 * 100 pixels. No further pre-processing is done on these images, and their pixel values are fed as the input to our network.

## VIII. Training of Network on External GPU

We trained our deep learning network as designed in the previous section using a single GPU instance on the Google

TABLE I: Network Architecture

| Layer | Type | Input Channels | Output Channels | Kernel Size | Activation |
|---|---|---|---|---|---|
| 1 | Conv | 3 | 32 | 5 | ReLU |
| 2 | Conv | 32 | 64 | 5 | ReLU |
| 3 | Conv | 64 | 128 | 5 | ReLU |
| 4 | Conv | 256 | 512 | 5 | ReLU |
| 5 | Conv | 512 | 256 | 5 | ReLU |
| 6 | DeConv | 256 | 128 | 5 | ReLU |
| 7 | DeConv | 128 | 32 | 5 | ReLU |
| 8 | Conv | 32 | 3 | 5 | ReLU |

Cloud platform. The NVIDIA Tesla K80 GPU was used to train our network using tflearn abstraction layers on a TensorFlow backend. We used 88,000 images as the training set and 12,000 images as the validation set. We and trained our network over 6 epochs with a batch size of 128 images. The learning rate was decreased by 50% at every half-epoch to ensure convergence. We also added a dropout (with a drop probability of 0.8) after the 7th layer to ensure generalization of the network. Our network took approximately 50 hours to train on a single GPU Windows 2012 Remote server.

## IX. Testing of the Trained Network

### A. Sketch Inversion

We tested our trained network on a number of generated sketches and original hand-drawn sketches . The results are shown in Figure 3. The left columns of images are the original photos whose generated sketches are depicted in the middle column. The right column of images are the corresponding output images of the network. It can be seen here that the network produces photo-realistic images very similar to the original photos. Although the model is not consistent in prediction of background color, the faces are generated in almost an identical manner as their original versions. The model was also seen to perform reasonably well on images with varying pose, illumination, background noise, facial expression and structural components such as beards, eyeglasses, etc. No post-processing was done on these images. We also tested our network on original hand-drawn sketches of various types. The output for these type of images are shown in Figure 4. It can be seen here that the produced images of these sketches are very realistic, albeit low resolution.

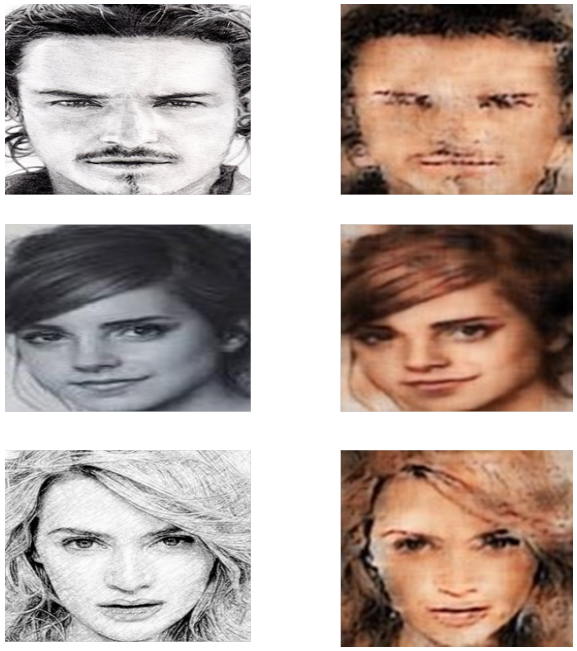

Fig. 4: Network Output for different types of sketches - Left Column: Original Sketches, Right Column: Model Output

Fig. 5: Top Row: Grey-Scale Video Frames, Bottom Row: Corresponding Colored Frames

Fig. 6: Top Row: Grey-Scale Video Frames, Bottom Row: Corresponding Colored Frames

### B. Colorization of Gray-Scale images

We also extended the application of this model to generate colorized versions of gray-scale face images. This is accomplished by first generating a sketch of the gray-scale image using the image processing technique highlighted in section V. These generated images were then fed into out model as its input, and the generated output result was the colorized version of the original gray-scale image. This is an interesting result of our model which shows that the model can be used directly

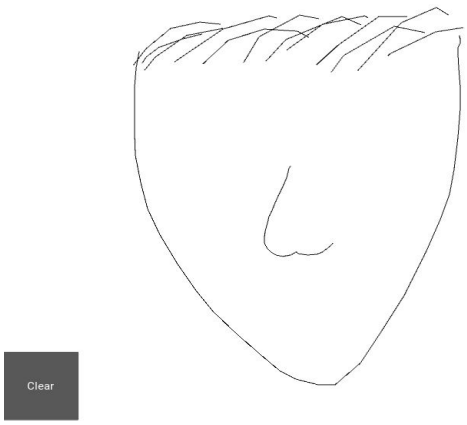


Fig. 7: Kivy Based Interface for Sketch Drawing

and indirectly for a variety of other image-related applications. Sample results obtained are shown in Figure 8.

### C. Extension to Video

After observing the positive results on testing our network on gray-scale images, we extended the same approach on gray-scale videos using our trained model. For this purpose, gray-scale video of face images were converted to sketches as before, and the colorized version was subsequently generated by our model. For the purpose of demonstration, we recorded our faces in gray-scale with movements to generate variance in pose and expression. We also used a moving average filter to smoothen the output video. Our model was able to produce the results of the video frames in real time. The results are shown in Figure 5 and Figure 6 for a few frames.

## X. Development of Graphical User Interface

We also developed a graphical user interface to capture the face-sketches being drawn by the user and convert it to its photo-realistic colored version in real time. We used the Kivy open source python library for this purpose. Kivy has the advantage of being capable of working on different platforms and on multi-touch applications. We developed an application for a Windows OS based touch pad where a blank white screen is provided for the user to sketch a face on using a gray colored pencil/touch. On user prompt, the image gets converted to its photo-realistic colored version by feeding it to our trained model. A screenshot of the interface for drawing the sketch is shown in Figure 7.

TABLE II: Network Evaluation

| | SSIM | PSNR |
|---|---|---|
| Line Sketch | $0.81342 \pm 0.027$ | $17.58561 \pm 0.014$ |
| Gray-scale | $0.75812 \pm 0.012$ | $15.82431 \pm 0.019$ |

## XI. Network Evaluation

We evaluated our network using the performance metrics outlined in section II (B). We calculated the SSIM and PSNR metrics for the CelebA database. SSIM (Structural Similarity) gives us an idea about the perceptual similarity of the generated image with the original one on the basis of contrast, luminance and structure. PSNR (Peak Signal to Noise Ratio) is a standard metric used for determining the quality of the generated image. The results obtained are given in Table 2. It can be seen here the structural similarity is quite high suggesting a healthy performance of our network. The training loss curves and validation accuracies obtained from Tensorboard are shown in Figure 9 and Figure 10 indicating that the training process was successful.

TABLE III: Facial Recognition Accuracy

| | Recognition Accuracy |
|---|---|
| Sketch Images | 65.34% |
| Generated Photos | 94.2% |

Fig. 8: Colorization of Grayscale Images - Left Column: Grey-Scale Images, Right Column: Model Output

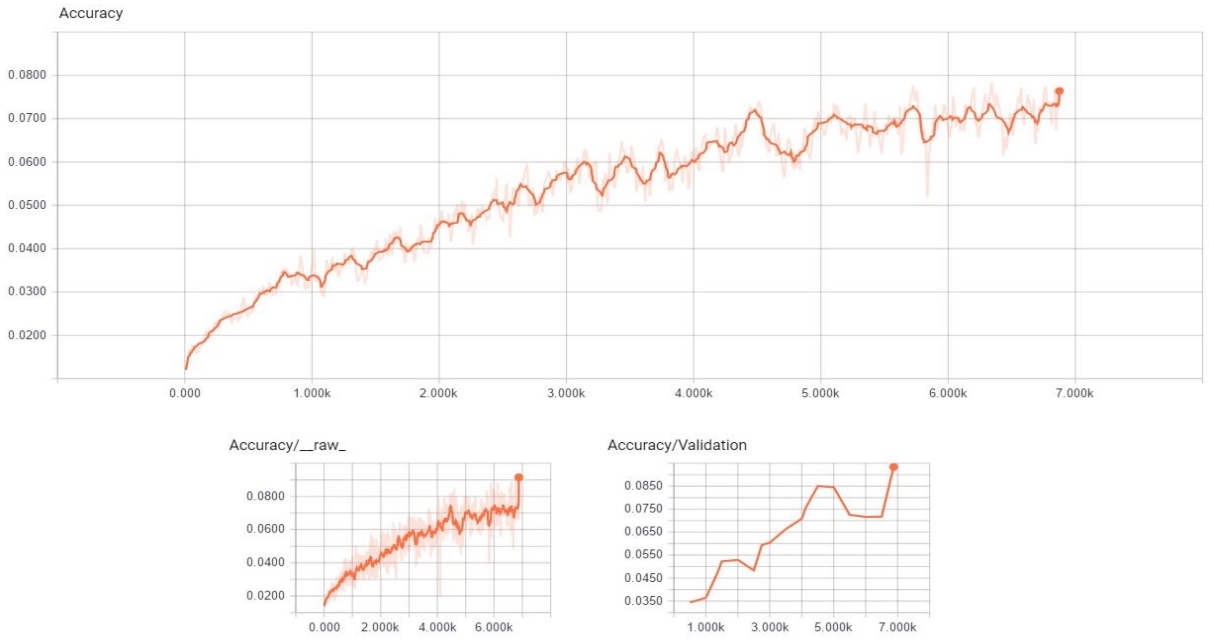


Fig. 9: Training and Validation Accuracies generated from Tensorboard

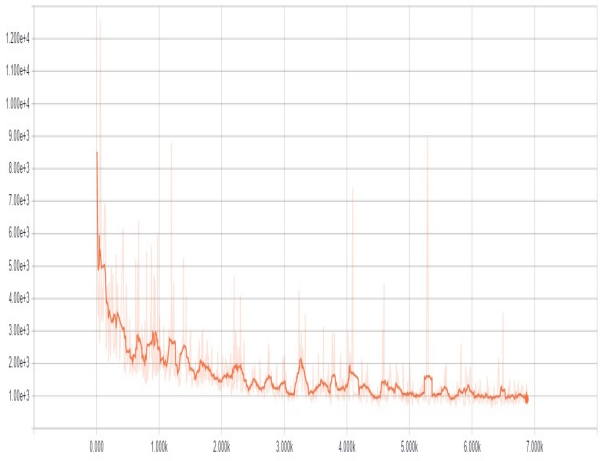

Fig. 10: Training Loss Function generated by Tensorboard

## XII. Facial Recognition

To validate our proposed application of forensic analyis, we evaluated our network by determining the classification accuracy of facial recognition using the output images and comparing its accuracy to that obtained using sketches. We carried out PCA (Principal Component Analysis) to project the input images along their top principal components and then matched images using a correlation matcher. The results displayed in Table 3 demonstrate that our generated photos perform better on a facial recognition system when compared to sketches.

## XIII. Conclusion

In this project, we successfully trained a deep convolutional network for the purpose of converting hand drawn line sketches to their photo realistic version and extended the work to realistic colorization of gray-scale images and videos. Our network consisted of 11 layers (8 convolutional and 3 deconvolutional). We used the ReLU activation function at each layer except the last layer where linear activation function was used. We trained our network on 100,000 widely varying images in terms of luminance and pose from the CelebA database after appropriately preprocessing them. The input to our network were the generated line sketches that we obtained from the original images using existing filters from the OpenCV library. The output were the corresponding original colored images. We trained our network using cloud GPU for 2 days. We then successfully tested our trained network on had drawn sketches, gray-scale images and gray-scale videos and found the results to be satisfying. We also developed a Kivy based GUI for sketch drawing and conversion. Finally, we evaluated our network using carefully chosen and well defined performance metrics and found that apart from doing very well on these metrics, our network output also had a high facial recognition accuracy.

## XIV. Future Scope

Apart from applications in forensic science and fine arts, an interesting extension of this work could deal with enhancing entire scenes (not just faces) being recorded in real time using our process of colorization and photo-realistic conversion. This could have a wide varying application range. Self driving and robotics are some of the possible applications.

## REFERENCES


[1] Rob van Lier Yamur Gltrk, Umut Gl and Marcel A. J. van Gerven. Convolutional sketch inversion. In *Computer Vision ECCV 2016 Workshops*, 2016.

[2] Xiaogang Wang and Xiaoou Tang. Face photo-sketch synthesis and recognition. In *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 2014.

[3] Dacheng Tao Xinbo Gao, Nannan Wang and Wei Liu. Face sketch-photo synthesis and retrieval using sparse representation. In *IEEE Transactions on Circuits and Systems for Video Technology*, 2012.

[4] Xian Wu Shengyong Ding Liliang Zhang, Liang Lin and Lei Zhang. End-to-end photo-sketch generation via fully convolutional representation learning. In *ICMR15*, 2015.

[5] Nannan Wang Jiewei Zhang and Xinbo Gao. Face sketch-photo synthesis based on support vector regression. In *2011 18th IEEE International Conference on Image Processing*, 2011.

[6] K. He C. Dong, C. C. Loy and X. Tang. Learning a deep convolutional network for image superresolution. In *Computer VisionECCV 2014*, 2014.

[7] M. Savvides Y.-h. Li and V. Bhagavatula. Illumination tolerant face recognition using a novel face from sketch synthesis approach and advanced correlation filters. In *International Conference on Acoustics, Speech, and Signal Processing, Institute of Electrical and Electronics Engineers (IEEE), 2006*, 2006.

[8] LeCunn et al. Convolutional neural networks (lenet) deeplearning 0.1 documentation. In *DeepLearning 0.1. LISA Lab*, 2013.

[9] LeCunn et al. Lenet - 5 convolutional neural networks. In *Proceedings of the IEEE, november 1998*, 1998.

[10] Paul Viola and Michael J. Jones. Robust real time face detection. In *International Journal of Computer Vision*, 2004.