Project Report on

SKIN CANCER DETECTION

Submitted By

Aryan Gupta 2019A7PS0017G

Under the Guidance of

Dr. Sanjay K. Sahay

Submitted in Partial Fulfillment of the requirements of

CS F266 STUDY PROJECT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI – Goa Campus

Second Semester 2021-2022

(April 2022)

ACKNOWLEDGEMENT

I would like to thank BITS Pilani, K.K. Birla Goa Campus and the instructor-in-charge of the project, Dr. Sanjay Sahay for giving me this opportunity to work on a practical project in Deep Learning. I would also like to thank the Professor for his constant support and valuable feedback.

Thanks to Mr. Kushanoor Akbar for giving me the opportunity to work with him. I would also like to thank Mr. Akbar for taking time out of his busy schedule to help and guide me.

ABSTRACT

The final goal of the project is to build a **Skin Cancer Detection** deep learning model. The report describes the work completed by me as a part of the Study Project throughout the semester.

The report starts with an introduction to melanoma cancer, discussing the primary diagnostic method. This also justifies the ideation of the project. It is followed by a brief explanation of the Deep Learning concepts that were required to be learnt by me to be able to work on building a working model.

Next, the project idea and structure has been explained, followed by a short discussion on the specific part towards which I contributed, that is, hair reduction from sample images.

Finally, I have discussed the methodology followed to complete the task. It is followed by a description of results obtained after building the autoencoder model.

Table of Contents

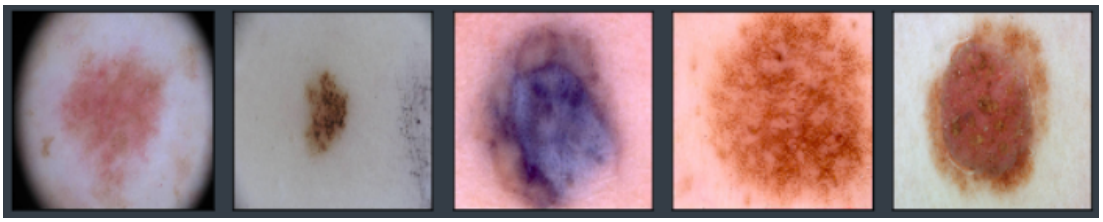
Introduction

What is Melanoma Cancer

Melanoma is a type of skin cancer that develops from the pigment-producing cells known as melanocytes.

The primary cause of melanoma is ultraviolet light exposure in those with low levels of the skin pigment melanin. Melanoma is the most serious type of skin cancer. It is much less common than some other types of skin cancers but is more dangerous because it's much more likely to spread to other parts of the body if not caught and treated early.

Melanoma is a cancer that begins in the melanocytes. Most melanoma cells still make melanin, so melanoma tumors are usually brown or black. But some melanomas do not make melanin and can appear pink, tan, or even white.



Diagnosis

The most common method of suspecting a melanoma is by **visually** looking at the area in question. Typically, the initial diagnostic step is **dermatoscopy**, also known as dermoscopy. Dermatoscopy allows for inspection of lesions on the skin surface under a magnifier (dermoscope). It may also capture digital images or video clips.

The diagnosis of skin lesions is mainly based on their morphological characteristics, such as an irregular shape, asymmetry and a variety of colors, along with a history of changes in size, shape, color and/or texture.
If the area appears to be suspicious, a skin sample may be further taken for biopsy or other laboratory analysis.

Deep Learning

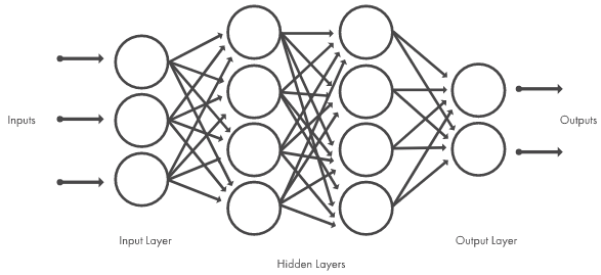
Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. In deep learning, a computer model learns to perform classification tasks directly from images, text or sound. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

Deep learning is applied in various industries.

- **Automated Driving:** automatically detect pedestrians and objects on the road
- **Aerospace and Defense:** identify objects and areas as safe/unsafe from satellites
- **Medical Research:** automatically detect cancer cells
- **Industrial Automation:** helping to improve worker safety around heavy machinery
- **Electronics:** automated hearing and speech translation. Eg. home assistance devices

Neural Networks

Most deep learning methods use neural network architectures. The depth of the neural network refers to the number of hidden layers in the network.



A neural network combines several processing layers of abstraction, using simple elements operating in parallel. The network consists of an input layer, one or more hidden layers, and an output layer. In each layer there are several nodes, or neurons, and the nodes in each layer use the outputs of all nodes in the previous layer as inputs, such that all neurons interconnect with each other through the different layers. Its behavior is defined by the weights by which its individual elements are connected. The weights are adjusted during the training according to a specific learning rule until the artificial neural network performs the desired task correctly.

A neural network can learn from data. So it can be trained to recognize patterns, classify data, and forecast future events.

Convolutional Neural Networks

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance - learnable weights and biases - to various aspects/objects in the image and be able to differentiate one from the other. The preprocessing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNNs have the ability to learn these characteristics.

CNNs are regularized versions of multilayer perceptrons (MLPs). The "full connectivity" of MLPs networks make them prone to overfitting data. CNNs take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters. Thus, a CNN is able to successfully capture the Spatial and Temporal dependencies in an image using fewer parameters and reusability of weights.

Recurrent Neural Networks

A recurrent neural network is a neural network that is specialized for processing a sequence of data. For tasks that involve sequential inputs, such as speech and language, it is often better to use RNNs. In a Natural Language Processing (NLP) problem, if you want to, say, predict the next word in a sentence it is important to know the words before it.

RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being dependent on the previous computations.

Another way to think about RNNs is that they have a “memory” which captures information about what has been calculated so far. The weights in an RNN are shared across time.

Long Short-Term Memory

In principle, the RNN is a simple and powerful model. Despite this, it is not used widely. The primary reason being the **vanishing gradient** problem. The vanishing gradients problem refers to when long term components go exponentially fast to norm 0, making it impossible for the model to learn correlation between temporally distant events.

This limitation is overcome by newer networks, such as the long short-term memory (LSTM), gated recurrent units (GRUs) and residual networks (ResNets).

They have internal mechanisms called gates that can regulate the flow of information. These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions. Almost all state of the art results based on recurrent neural networks are achieved using LSTMs or GRUs.

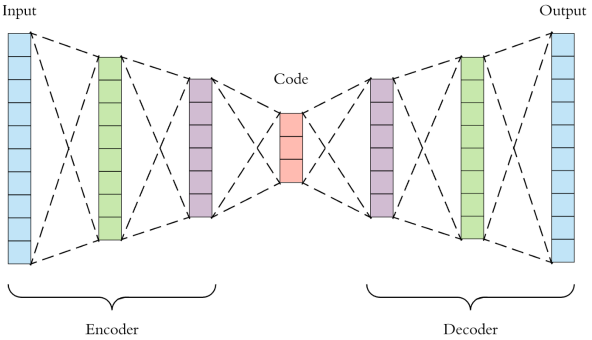
AutoEncoders

Autoencoders are a specific type of feedforward neural networks where the input is the same as the output. An autoencoder consists of 3 components: encoder, code and decoder. The encoder compresses the input into a lower-dimensional code. The code is a compact “summary” or “compression” of the input. The decoder then reconstructs the input only using this code.

Some important properties of autoencoders:

- Data-specific: can meaningfully compress data similar to what they have been trained on
- Lossy: output not exactly the same as the input - it will be close but degraded
- Unsupervised: Autoencoders are considered an unsupervised learning technique since they don’t need explicit labels to train on. But to be more precise they are self-supervised because they generate their own labels from the training data.

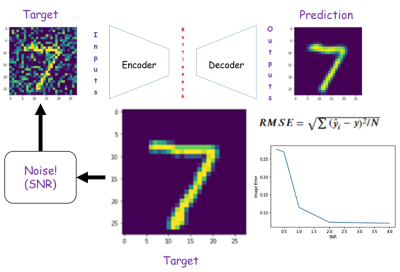
The architecture of an autoencoder is a symmetric bottle-neck structure as shown:

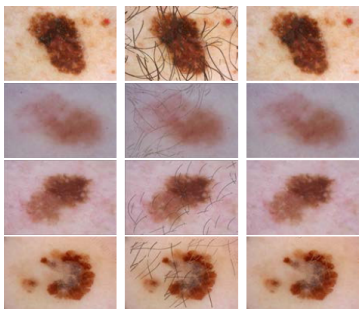


The hyperparameters in an autoencoder are the number and width of layers, size of the code layer and loss function.

Denoising AutoEncoder

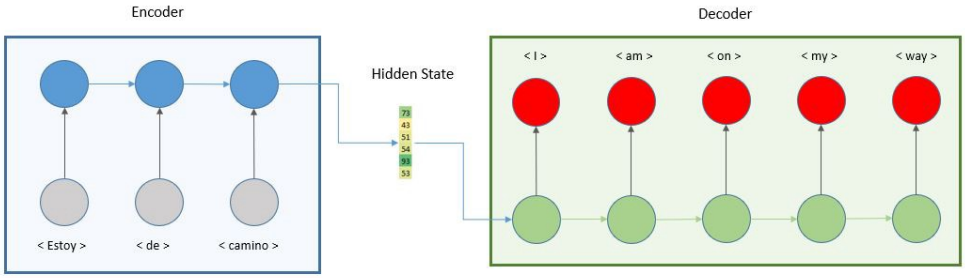
There is another way to force the autoencoder to learn useful features, which is adding random noise to its inputs and making the model recover the original noise-free data. This way the autoencoder can’t simply copy the input to its output because the input also contains random noise. The autoencoder is asked to subtract the noise and produce the underlying meaningful data. This is called a **denoising autoencoder**.





Encoder Decoder Model

Just like an autoencoder, an encoder-decoder model consists of 3 components: encoder, hidden state and decoder.



1. Encoding means to convert data into a required format. Encoder is built by stacking RNN so that the model is able to understand the context and temporal dependencies of the sequences.
2. The hidden state is the output of the encoder, the state of the last RNN timestamp. It is a two-dimensional vector that encapsulates the whole meaning of the input sequence.
3. Decoding means to convert a coded message into intelligible language. It is also built with RNN layers to predict the next frame in the sequence.

Skin Cancer Detection - Using Deep Learning

As discussed earlier, the primary diagnostic method of melanoma is by visually observing the skin sample. Practitioners rely on the dermoscopic evaluation for completing the clinical analysis and the diagnosis of melanoma. This practice improves the diagnostic accuracy up to 10–30% compared to simple clinical observation. However, their evaluation might be altered by individual judgment and experience. Thus, to obtain early, objective and reproducible results, the idea is to automate this step by passing an image of the skin area through an image classification system.

Stages of the Project

1. Pre-Processing of skin image samples
    a. Noise Reduction
    b. Hair Reduction
2. Convolutional Neural Network
    a. Features Extraction
    b. Binary Classification
3. LSTM - Encoder Decoder Model - Natural Language Processing
    a. Test Report Generation
4. Grad Cam Visualization
    a. Proving correctness of results / model's output

Hair Reduction

In the pre-processing stage during the analysis of the lesions, hair removal is one of the key steps. The presence of hair in dermoscopic images usually occludes significant patterns reducing the accuracy of the system.

Once the hairs are detected and removed, the next step is to estimate and restore the underlying information, i.e., color and texture/patterns, of the skin pixels underneath the hair regions.

Methodology

In the skin image samples, hairs need to be removed and those pixels need to be replaced appropriately. Hairs act as unnecessary elements/information, or **noise**.
Hence, a **Denoising Autoencoder** has been modeled for reducing hair from the images.

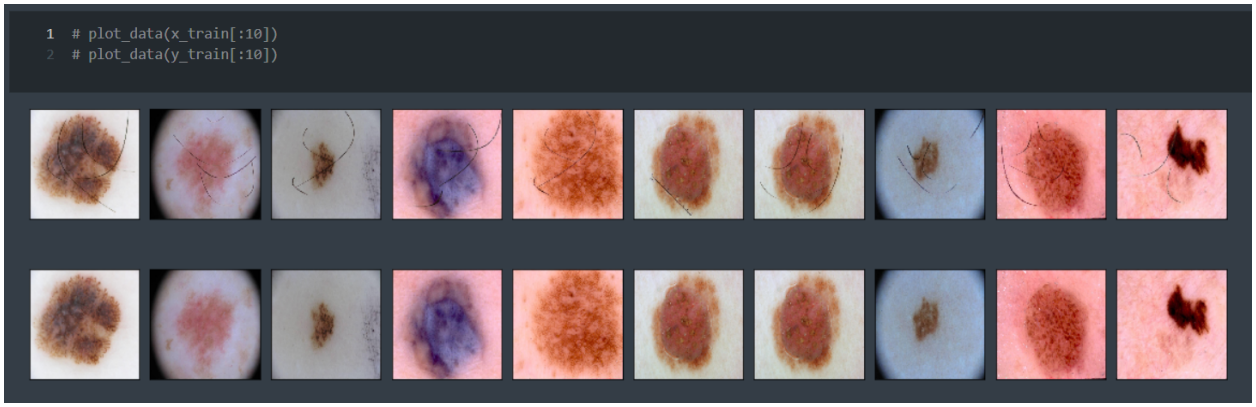
Dataset Preparation

The first step towards building the autoencoder model was to prepare a dataset that would be used to train the network.

We need a set of skin sample images with hair in it and the corresponding hairless samples. It is difficult to get original pairwise samples for hairy and hairless images with the exact same framing and lighting. Thus, a different method has to be adopted. We take hairless samples and add hair to it to create the hairy images.

The hairy images act as training input data and the original hairless images as the target output data of the denoising autoencoder.

A GitHub repository was followed to generate the training data. A conda environment was set-up and the steps were performed as described to first prepare a set of hairless images and next add hair to those images.
A training data of 97 pairs of images was generated.
Code for the same:
https://colab.research.google.com/drive/1T1bViNsdJ9skdf1klf1mo_KuCmmdm-ra?authuser=2#scrollTo=NhvG0wFRhY4d
Challenges: Setting up the Conda Environment in a Google Colab notebook was not straightforward and work-arounds had to be made for the same.



10 samples of hairy (top) and the corresponding hairless (bottom) training images

Architecture

- A denoising autoencoder model consisting of convolutional layers is designed.
- Adam optimiser has been used.
- The loss functions tried were Binary Cross Entropy and Mean Square error.
- Skip connections have been used, in a manner similar to U-net.
- The final model used in our project is as described in the model summary below.

```
inp = Input(x_train[0].shape)

x1 = Conv2D(32, (3,3), padding='same', activation='relu', kernel_initializer='he_normal')(inp)
x2 = Conv2D(128, (3,3), padding='same', activation='relu', strides = (2,2))(x1)
x3 = Conv2D(256, (3,3), padding='same', activation='relu')(x2)
x4 = Conv2D(512, (3,3), padding='same', activation='relu', strides = (2,2))(x3)
x5 = Conv2D(512, (3,3), padding='same', activation='relu')(x4)
x6 = Conv2DTranspose(256, (3,3), padding='same', activation='relu', strides = (2,2))(x5) + x3
x7 = Conv2D(128, (3,3), padding='same', activation='relu')(x6) + x2
x8 = Conv2DTranspose(32, (3,3), padding='same', activation='relu', strides = (2,2))(x7) + x1
x9 = Conv2D(128, (3,3), padding='same', activation='relu')(x8)
out = Conv2D(3, (3,3), padding='same', activation='relu')(x9)

model = Model(inputs = inp, outputs = out)
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=0.0001), loss = 'MSE')
model.summary()
```

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_1 (InputLayer)         [(None, 512, 512, 3       0         []
                             )]

conv2d (Conv2D)              (None, 512, 512, 32       896       ['input_1[0][0]']
                             )

conv2d_1 (Conv2D)            (None, 256, 256, 12       36992     ['conv2d[0][0]']
                             8)

conv2d_2 (Conv2D)            (None, 256, 256, 25       295168    ['conv2d_1[0][0]']
                             6)

conv2d_3 (Conv2D)            (None, 128, 128, 51       1180160   ['conv2d_2[0][0]']
                             2)

conv2d_4 (Conv2D)            (None, 128, 128, 51       2359808   ['conv2d_3[0][0]']
                             2)

conv2d_transpose (Conv2DTransp  (None, 256, 256, 25    1179904   ['conv2d_4[0][0]']
ose)                         6)

tf.__operators__.add (TFOpLamb  (None, 256, 256, 25    0         ['conv2d_transpose[0][0]',
da)                          6)                                   'conv2d_2[0][0]']

conv2d_5 (Conv2D)            (None, 256, 256, 12       295040    ['tf.__operators__.add[0][0]']
                             8)

tf.__operators__.add_1 (TFOpLa  (None, 256, 256, 12    0         ['conv2d_5[0][0]',
mbda)                        8)                                   'conv2d_1[0][0]']

conv2d_transpose_1 (Conv2DTran  (None, 512, 512, 32    36896     ['tf.__operators__.add_1[0][0]']
spose)                       )

tf.__operators__.add_2 (TFOpLa  (None, 512, 512, 32    0         ['conv2d_transpose_1[0][0]',
mbda)                        )                                    'conv2d[0][0]']

conv2d_6 (Conv2D)            (None, 512, 512, 12       36992     ['tf.__operators__.add_2[0][0]']
                             8)

conv2d_7 (Conv2D)            (None, 512, 512, 3)       3459      ['conv2d_6[0][0]']

==================================================================================================
Total params: 5,425,315
Trainable params: 5,425,315
Non-trainable params: 0
```

- The model is inspired by the design as given in the paper mentioned in the references:

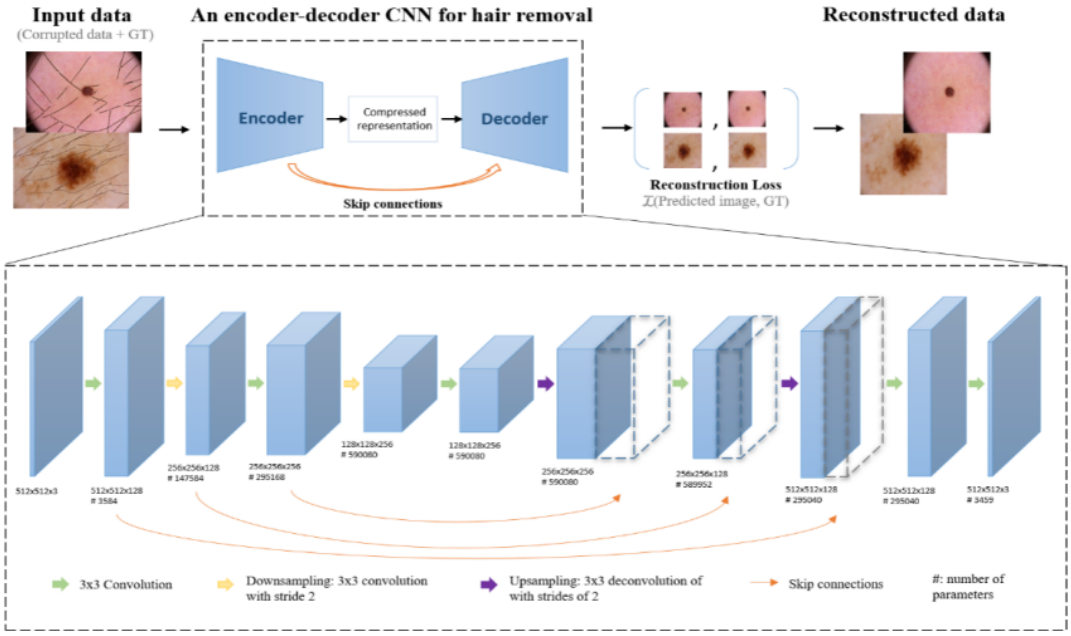


Figure 1: Architecture of our proposed network. The pairs of reference hairless images and its corrupted (hair simulated) images are passed through the encoder to extract complex features. The decoder, connected to the encoder with skip-connections, reconstructs the image.

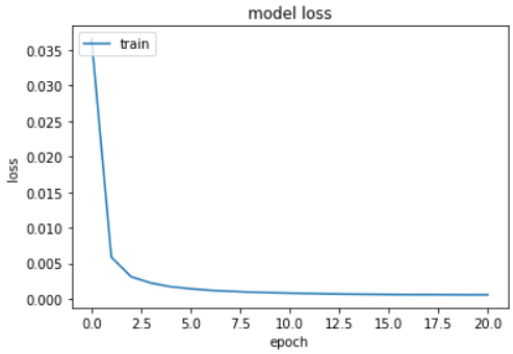
Screenshot from the Paper Referenced

Learning of the Model

The model was set to be trained for 30 epochs. Training data samples were processed in batches of size 4. Callbacks used were Early Stopping and Reduction of Learning Rate on Plateau.

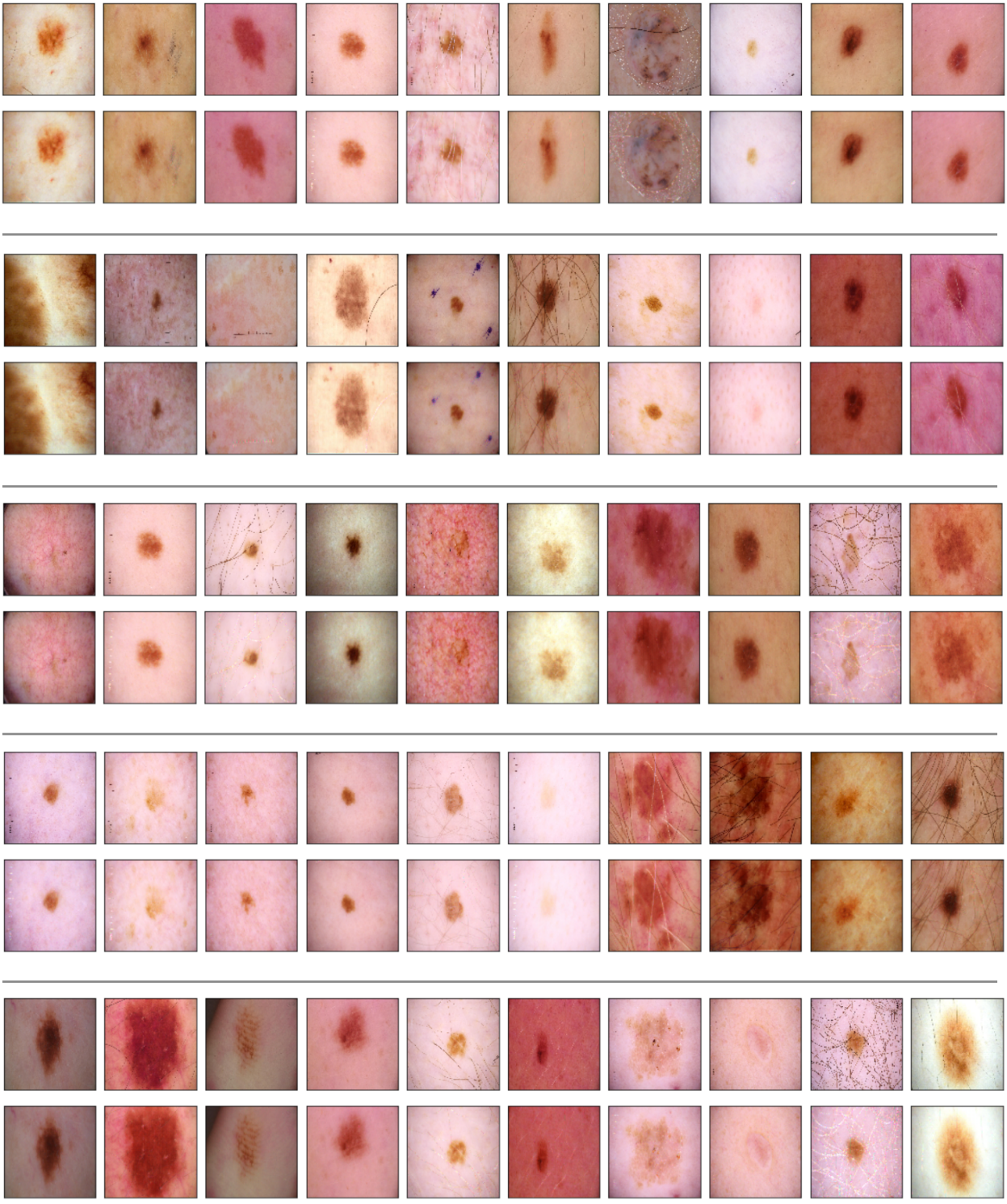
```
earlystop = EarlyStopping(monitor='loss', min_delta=0.0001, patience=4, verbose=0, mode='auto')
learning_rate_reduction = ReduceLROnPlateau(monitor='loss', patience=2, verbose=1, factor=0.3, min_lr=0.000001)

history = model.fit(x_train[:-1], y_train[:-1], epochs=30, batch_size=4, callbacks = [earlystop,learning_rate_reduction])
```

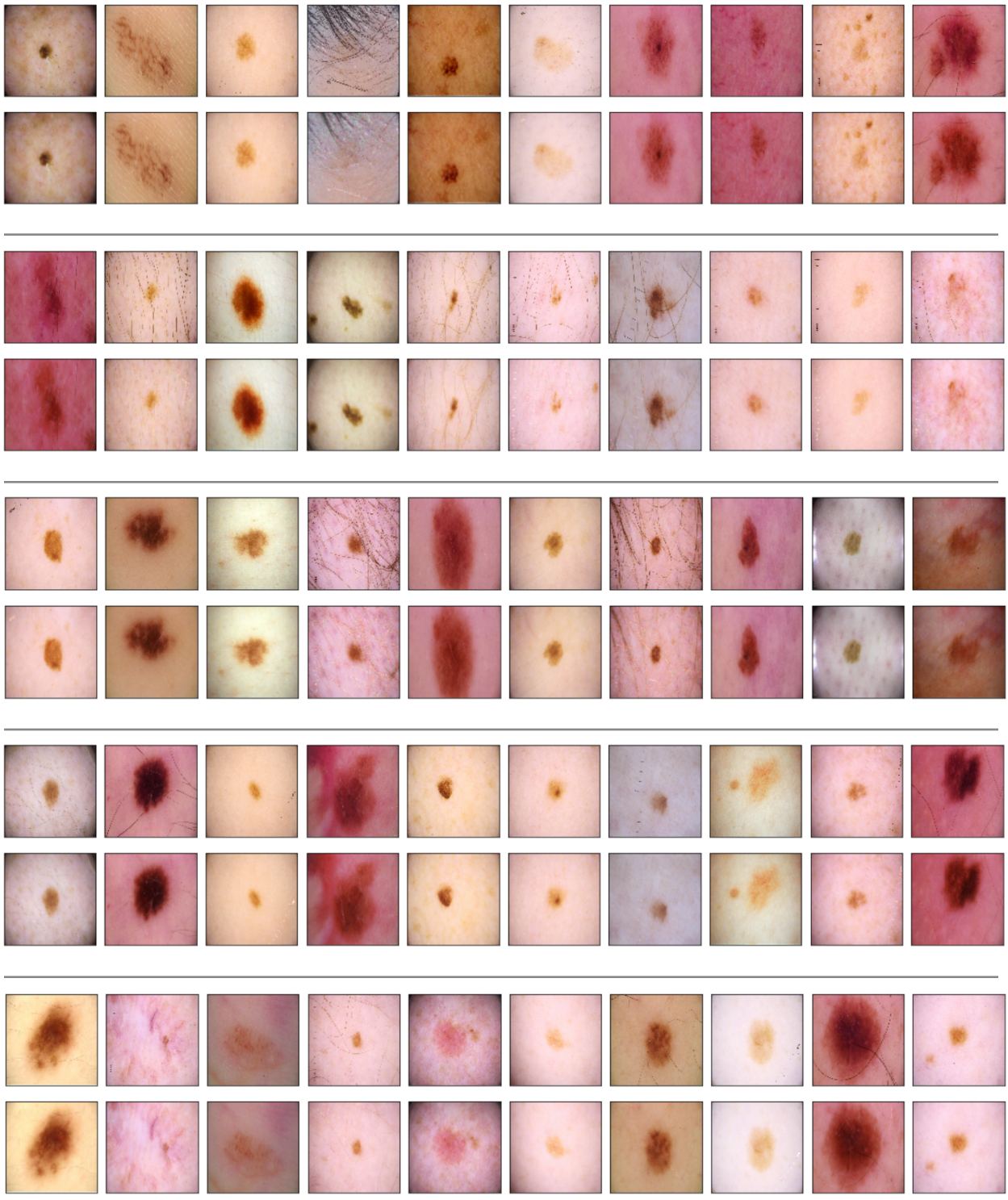
The training loss during the 20 epochs of training of the model:



Results

The model was tested on 100 image samples from the dataset to be used while building the classification model. It not only reduces hair from the images but also helps in noise reduction.





Code

[Skin Cancer Autoencoder.ipynb - Colaboratory]

Future Work

- A custom **Loss Function** can be implemented, as described in the reference research paper.
- **Larger training dataset** can be used - find more ways to generate such dataset.
- Following the pre-processing of the images, the next 3 steps remain - building image classification model, report generation model, grad-cam visualization.

References

- https://en.wikipedia.org/wiki/Melanoma#Diagnosis
- https://www.cancer.org/cancer/melanoma-skin-cancer/about/what-is-melanoma.html
- https://en.wikipedia.org/wiki/Dermatoscopy
- https://in.mathworks.com/discovery/deep-learning.html
- https://in.mathworks.com/discovery/neural-network.html
- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- https://en.wikipedia.org/wiki/Convolutional_neural_network
- https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85
- https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
- https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798
- https://towardsdatascience.com/what-is-an-encoder-decoder-model-86b3d57c5e1a
- *An Encoder-Decoder CNN for Hair Removal in Dermoscopic Images - Lidia Talavera-Martınez , Pedro Bibilonia, Manuel Gonzalez-Hidalgo*
- https://github.com/attiamohammed/realhair
    - @article{attia2018realistic, title={Realistic hair simulator for skin lesion images using conditional generative adversarial network}, author={Attia, Mohamed and Hossny, Mohammed and Zhou, Hailing and Yazdabadi, Anosha and Asadi, Hamed and Nahavandi, Saeid}, journal={[Unknown]}, pages={1--11}, year={2018}, publisher={[MDPI]} }
    - @article{attia2019digital, title={Digital hair segmentation using hybrid convolutional and recurrent neural networks architecture}, author={Attia, Mohamed and Hossny, Mohammed and Zhou, Hailing and Nahavandi, Saeid and Asadi, Hamed and Yazdabadi, Anousha}, journal={Computer methods and programs in biomedicine}, volume={177}, pages={17--30}, year={2019}, publisher={Elsevier} }