Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

Greyscale Image Colorization

June 2019

Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

Greyscale Image Colorization

By:
Ahmed Nasser Abd El Baky[CS].
Ahmed Kamal El Din [CS].
Ahmed Mostafa El Moslimay[CS].
Ahmed Hamdy Abdel Fattah [CS].

Supervision by:

Prof.Dr **Mohamed Ismail Roshidy,**
Professor Dr,
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

Dr. **Dina Khattab**
Lecturer,
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

LA. **Nora Youssef,**
Lecturer Assistant,
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

Acknowledgment

All praise and thanks to ALLAH , Who provided us the ability to complete this work.
We also would like to thank our parents ,friends who are always provides us help and support through out whole years of study.
Our highest gratitude and appreciation go to the supervisors on the project, Dr. Dina and Dr. Nora we wouldn’t have done it without your help , we are so grateful for your support.

Abstract

Image colorization is the process of adding color to grayscale or sepia images, usually with the intent to modernize them. Automated image colorization is an ill-posed problem, as two objects with different colors can appear the same on grayscale film. Because of this, image colorization algorithms commonly require user input, either in the form of color annotations or a reference image.

We present a convolutional-neural-network-based system that faithfully colorizes black and white photographic images without direct human assistance. We explore various network architectures, objectives, color spaces, and problem formulations. The final classification-based model we build generates colorized images that are significantly more aesthetically-pleasing , demonstrating the viability of our methodology and revealing promising avenues for future work.

we demonstrate our method extensively on many different types of images including black-and-white photography from over a hundred years ago, and show realistic colorizations.

Table of Content

List of Abbreviations

AI Artificial Intelligence
ML Machine Learning
NN Neural Network
ANN Artificial Neural Network
CNN Conventional Neural Network
BP Back Propagation
DL Deep Learning
CPU Central Processing Unit

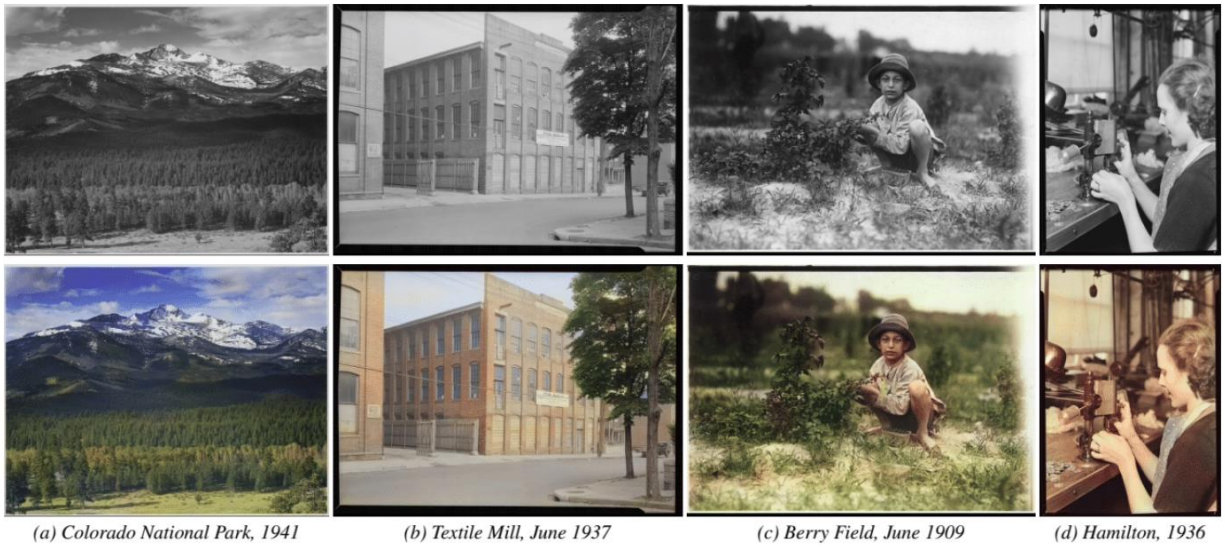
1- Introduction

We introduce the problem of colorization greyscale images which consider one of a popular problems face the user because it requires a hard effort to produce a realistic and natural output close to the ground truth and a high computation from a machine.

There were many applications to apply colorization process over images but it requires a deep interaction with the user to help in colorizing process which still a problem to solve. We propose a fully automatic image colorization system with adding realistic colors to the greyscale images using deep learning without any user interaction which solve the problem.

Problem definition:-

The goal of this set of work is to developing a fully automatic colorization system with adding realistic colors to the grayscale images. Specifically , given a single-channel gray-scale image, we want to output two channel color information of the same size.



(a) Colorado National Park, 1941 *(b) Textile Mill, June 1937* *(c) Berry Field, June 1909* *(d) Hamilton, 1936*

Figure 1: Colorization Examples.

Motivation:-

Image processing uses grayscale to work, Just one channel of color, only 8 bits to be represented, So use less data size and the same efficiency as luminance is more important in distinguishing visual features, Often it's not relates to image colors, So if there is a pure image processing system, The output needs an efficient automatic colorization system as the image may have differed much from what it was before.

The first cameras were produced in the early 1800s with black and white images. And Colored photography and film did not really exist until the early 1900s and did not become economically feasible for people until even later. Because of this, there are a large amount of black and white images and videos (classic films).

Figure 3: Converting a gray scale image to colored one may produce more than one version of colored image.

Many of the surveillance cameras in the streets, whether governmental or institutional, only support Grayscale, to save size of storage, so if there is a crime or a desire to show some details would need an efficient automatic image colorization system So previous mentioned problems related would need an efficient automatic image colorization system.

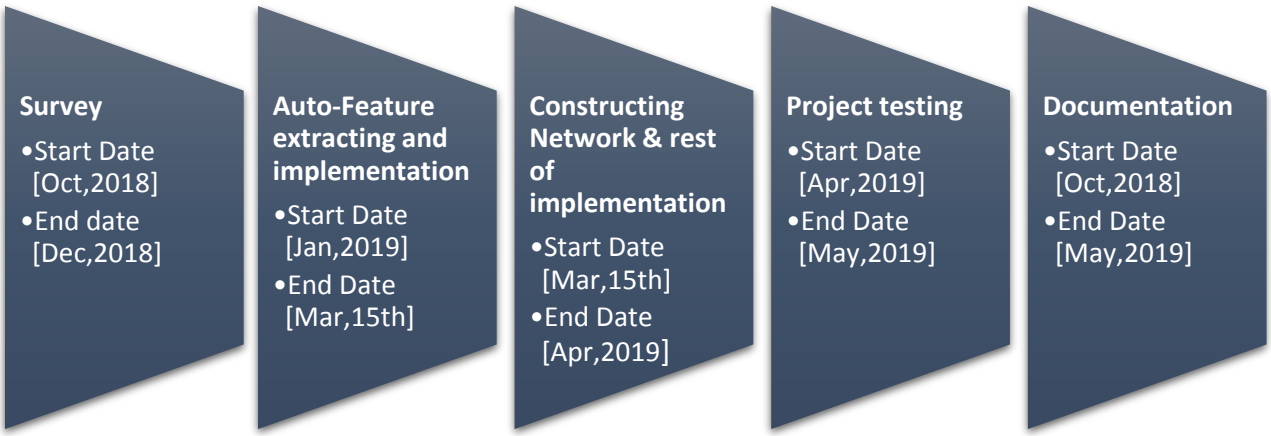
Objective:-

Our target is to build a model which automatically colorize grayscale images system without any user interactions or user hints based on CNN and specially Deep Learning.

Figure 4: colorizing grayscale image with high accuracy.

Time Plan:-

The schedule on which this project was executed is shown below. The project started in Oct 2018, and extended until May 2019.



Documentation Organization:-

Starting with:

- Chapter 2 Background.
- Chapter 3 System Architecture .
- Chapter 4 Training and Datasets.
- Chapter 5 User Manual.
- Chapter 6 Experimental Results and Discussion, Conclusion, Future Work and Reference.

2- Background:-

In this chapter we will explain the needed background, start from basic knowledge about images and how to use some operations and processes to be ready to understand the steps of colorization

Images:-

Images are one of the many types of media used on our world, as we know that the colors make image is more meaning full and clear to us to determine what this image mean or what this image try to describe to us.

Color digital images are made of pixels, and pixels are made of combinations of primary colors represented by a series of code. A channel in this context is the grayscale image of the same size as a color image, made of just one of these primary colors. For instance, an image from a standard digital camera will have a red, green and blue channel. A grayscale image has just one channel.

Colors is considered as one of most important information about image, there are other important information can be extracted from an image to define the difference between it and another image.
Existing of colors is important and there are many color spaces and models can represent the image information

Colors:-

Color refers to the human brain's subjective interpretation of combinations of a narrow band of wavelengths of light.
Also, what wavelengths reach the eye depend on both the wavelengths in the light source and what wavelengths are absorbed by the objects off which the light reflects.
Fortunately for most Web developers, displays generate their own light in reasonably consistent manners, which simplifies working with color.

Choosing a Color Model No one color model is necessarily "better" than another. Typically, the choice of a color model is dictated by external factors, such as a graphics tool or the need to specify colors according to the CSS2 or CSS3 standard. The following discussion only describes how the models function, centered on the concepts of hue, shade, tint, and tone.

Color Components:-

Brightness: the human sensation by which an area exhibits more or less light.

Hue: the human sensation according to which an area appears to be similar to one, or to proportions of two, of the perceived colors red, yellow, green and blue.

Colorfulness: the human sensation according to which an area appears to exhibit more or less of its hue.

Lightness: the sensation of an area’s brightness relative to a reference white in the scene.

Chroma: the colorfulness of an area relative to the brightness of a reference white.

Saturation: the colorfulness of an area relative to its brightness.

Luminance: is a photometric measure of the luminous intensity per unit area of light. It describes the amount of light that passes through, is emitted or reflected from a particular area.

Chroma and Luminance consider one of important information can represent the image colors by combine them together as shown in figure 2.1.2.1.

Figure 5. Luminance only, Chrominance only, and full color image (Luminance and Chrominance).

Color Spaces:-

A color space is a method by which we can specify, create and visualize color. As humans, we may define a color by its attributes of brightness, hue and colorfulness.
A computer may describe a color using the amounts of red, green and blue phosphor emission required to match a color.
A color is thus usually specified using three co-ordinates, or parameters. These parameters describe the position of the color within the color space being used. They do not tell us what the color is, that depends on what color space is being used.

An analogy to this is that I could tell you where I live by giving directions from the local garage, those directions only mean anything if you know the location of the garage. If you don't know where the garage is, the instructions are meaningless.
The aim of color spaces is to aid the process of describing color, either between people or between machines or programs.

Our Color Space:- CIE Lab color space , L is for (Lightness) , a is for (Red/Green Values) , b is for (blue/yellow Values)

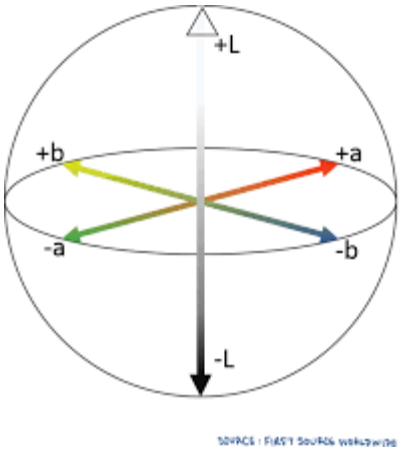


Image Features:-

In computer vision and image processing, a **feature** is a piece of information which is relevant for solving the computational task related to a certain application. This is the same sense as feature in machine learning and pattern recognition generally, though image processing has a very sophisticated collection of features. Features may be specific structures in the image such as points, edges or objects. Features may also be the result of a general neighborhood operation or feature detection applied to the image.

Other examples of features are related to motion in image sequences, to shapes defined in terms of curves or boundaries between different image regions, or to properties of such a region.

The feature concept is very general and the choice of features in a particular computer vision system may be highly dependent on the specific problem at hand.

Introduction When features are defined in terms of local neighborhood operations applied to an image, a procedure commonly referred to as feature extraction, one can distinguish between feature detection approaches that produce local decisions whether there is a feature of a given type at a given image point or not, and those who produce non-binary data as result. The distinction becomes relevant when the resulting detected features are relatively sparse. Although local decisions are made, the output from a feature detection step does not need to be a binary image. The result is often represented in terms sets of (connected or unconnected) coordinates of the image points where features have been detected, sometimes with sub-pixel accuracy.

When feature extraction is done without local decision making, the result is often referred to as a feature image. Consequently, a feature image can be seen as an image in the sense that it is a function of the same spatial (or temporal) variables as the original image, but where the pixel values hold information about image features instead of intensity or color. This means that a feature image can be processed in a similar way as an ordinary image generated by an image sensor. Feature images are also often computed as integrated step in algorithms for feature detection.

Artificial Intelligence (AI):-

Artificial Intelligence (AI) is usually defined as the science of making computers do things that require intelligence when done by humans. AI has had some success in limited, or simplified, domains. However, the five decades since the inception of AI have brought only very slow progress, and early optimism concerning the attainment of human-level intelligence has given way to an appreciation of the profound difficulty of the problem.

AI Goal

The overall research goal of artificial intelligence is to create technology that allows computers and machines to function in an intelligent manner. The general problem of simulating (or creating) intelligence has been broken down into sub- problems. These consist of particular traits or capabilities that researchers expect an intelligent system to display. There are many traits described below have received the most attention.
Reasoning, problem solving, Knowledge representation, Planning ,Machine Learning, Natural language processing, Perception, Social intelligence.

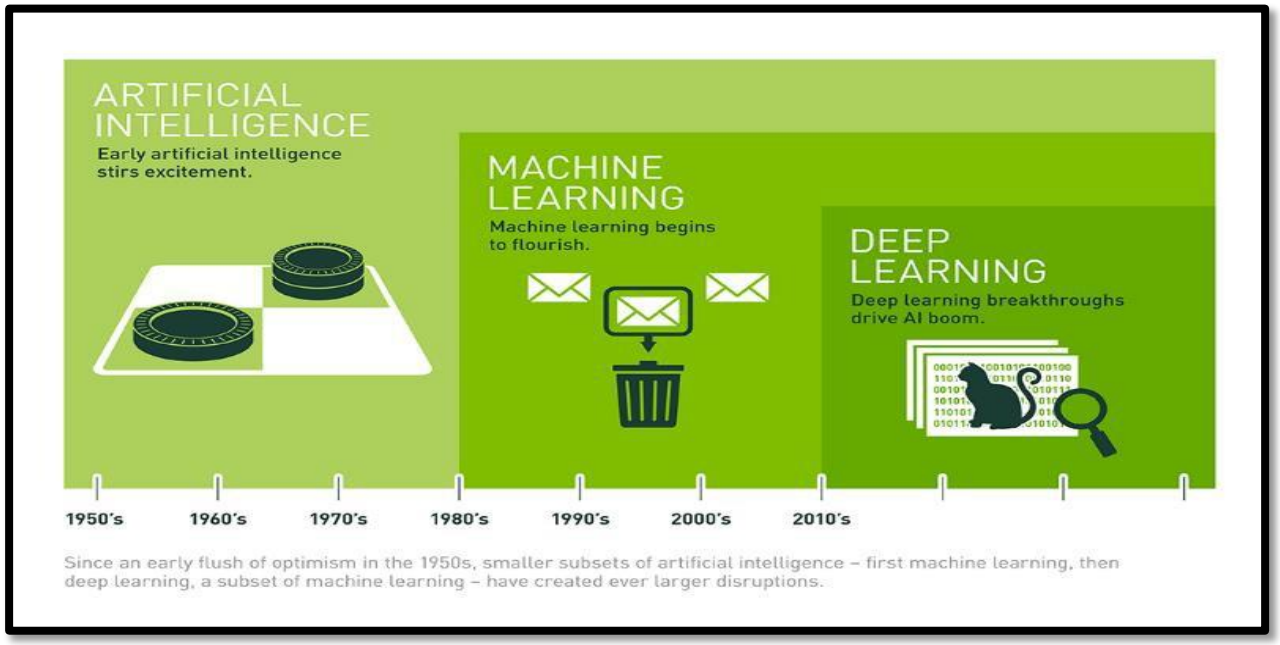


Figure 7: **describe AI and ML and DL**

Machine learning (ML):-

Machine learning is the study of computer algorithms that improve automatically through experience and has been central to AI research since the field's inception. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions through building a model from sample inputs.

Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or unfeasible; example applications include email filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition (OCR), learning to rank and computer vision. Learning process is divided into two category: **supervised**, **unsupervised** learning and **reinforcement learning.**

Supervised Learning:

The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. Supervised learning includes both classification and numerical regression. Classification is used to determine what category something belongs in, after seeing a number of examples of things from several categories. Regression is the attempt to produce a function that describes the relationship between inputs and outputs and predicts how the outputs should change as the inputs change.

Unsupervised Learning:

Unsupervised Learning No labels are given to the learning algorithm, leaving it on its own to find structure in its input.

Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

The machine task of inferring a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations).

Since the examples given to the learner are unlabeled, there is no evaluation of the

accuracy of the structure that is output by the relevant algorithm which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

Neural Network (NN):-

A neural network is a machine learning algorithm based on the model of a human neuron. The human brain consists of millions of neurons

A beautiful biologically-inspired programming paradigm which enables a computer to learn from observational data based on the structure and functions of biological neural networks.

It intended to simulate the behavior of biological systems composed of "**neurons**". ANNs are computational models inspired by an animal's central nervous systems. It is capable of **machine learning** as well as pattern recognition. These presented as systems of interconnected "**neurons**" which can compute values from inputs.

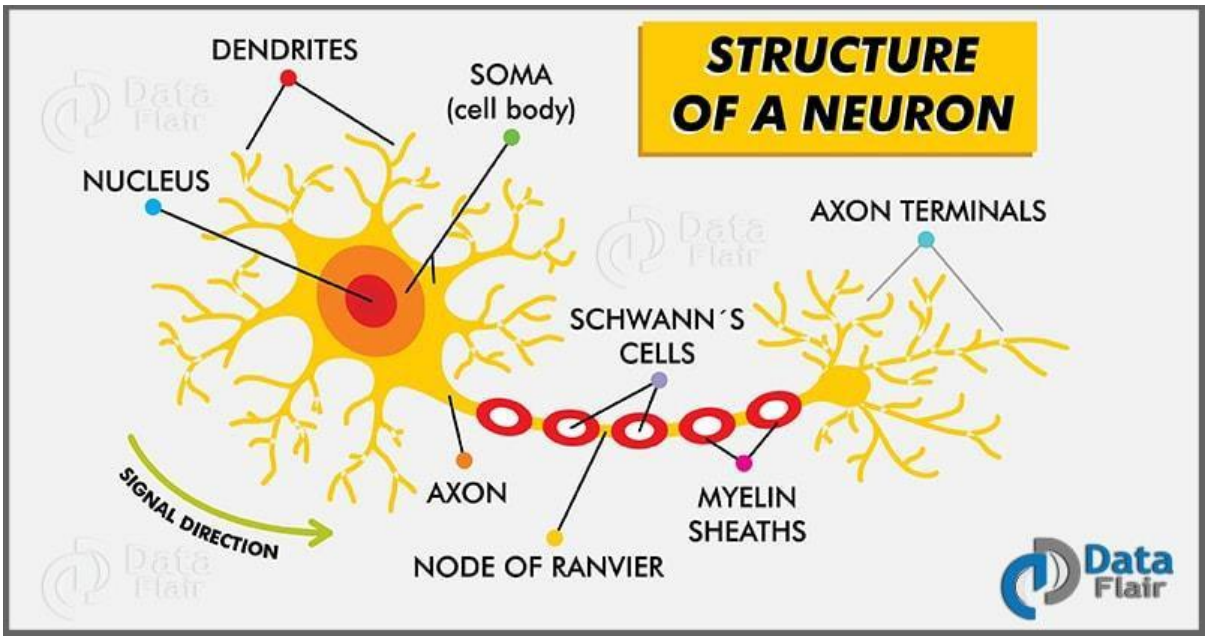


Figure 8: Structure of a human neuron [6].

A neural network is an oriented graph. It consists of nodes which in the biological analogy represent neurons, connected by arcs. It corresponds to dendrites and synapses. Each arc associated with a weight while at each node. Apply the values received as input by the node and define Activation function along the incoming arcs, adjusted by the weights of the arcs.

Artificial Neural Network (ANN):-

An Artificial Neural Network is an information processing technique. It works like the way human brain processes information.
ANN includes a large number of connected processing units that work together to
process information. They also generate meaningful results from it

We can apply neural network not only for classification. It can also apply for regression of continuous target attributes.

Neural networks find great application in data mining used in sectors. For example economics, forensics and for pattern recognition. It can be also used for data classification in a large amount of data after careful training.

A neural network may contain the following 3 layers:

Input layer – The activity of the input units represents the raw information that can feed into the network.

Hidden layer – To determine the activity of each hidden unit. The activities of the input units and the weights on the connections between the input and the hidden units. There may be one or more hidden layers.

Output layer – The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

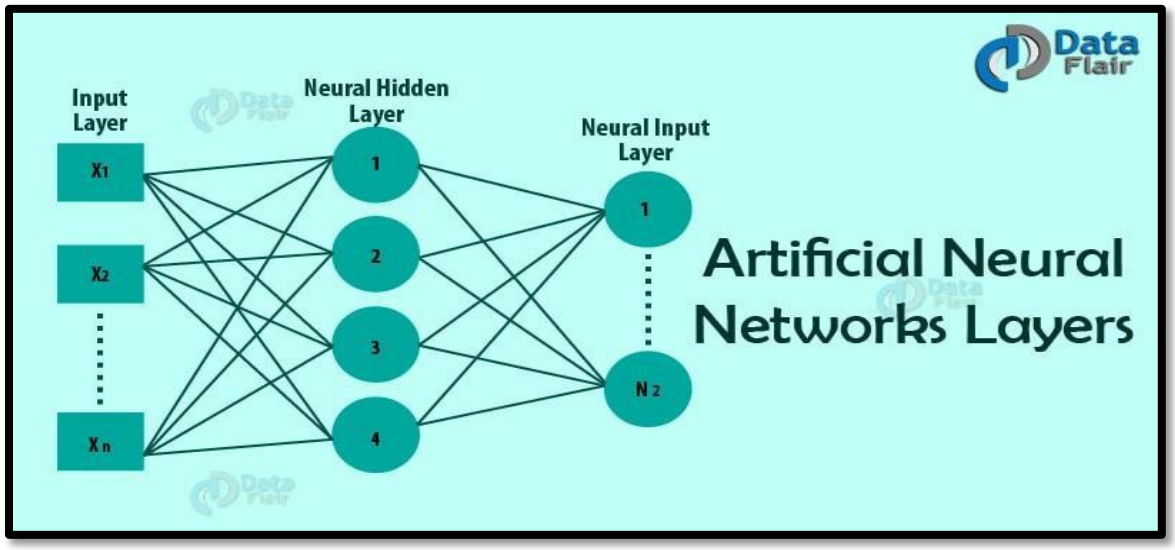


Figure 9: ANN layers [6].

Input layer

The purpose of the input layer is to receive as input the values of the explanatory attributes for each observation. Usually, the number of input nodes in an input layer is equal to the number of explanatory variables. 'Input layer' presents the patterns to the network, which communicates to one or more 'hidden layers'.

The nodes of the input layer are passive, meaning they do not change the data. They receive a single value on their input and duplicate the value to their many outputs. From the input layer, it duplicates each value and sent to all the hidden nodes.

Hidden layer

The Hidden layers apply given transformations to the input values inside the network. In this, incoming arcs that go from other hidden nodes or from input nodes connected to each node. It connects with outgoing arcs to output nodes or to other hidden nodes. In hidden layer, the actual processing is done via a system of weighted ‘connections’. There may be one or more hidden layers. The values entering a hidden node multiplied by weights, a set of predetermined numbers stored in the program. The weighted inputs are then added to produce a single number.

Output layer

The hidden layers then link to an ‘output layer‘. Output layer receives connections
from hidden layers or from input layer. It returns an output value that corresponds to the prediction of the response variable. In classification problems, there is usually only one output node. The active nodes of the output layer combine and change the data to produce the output values.

The ability of the neural network to provide useful data manipulation lies in the proper selection of the weights. This is different from conventional information processing.

Convolutional Neural Network (CNN):-

Convolutional neural networks were inspired by biological processes. The connectivity pattern between neurons is inspired by the animal visual cortex. Although, Individual cortical neurons are used to respond to stimuli that are only in a restricted region of the visual field known as the receptive field. Further, this field of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms and they have applications in image and video recognition, recommender systems and natural language processing.

Architecture of Convolutional Neural Networks:

- Convolutional layer
- Pooling layer
- Flatten layer
- Fully connected layer

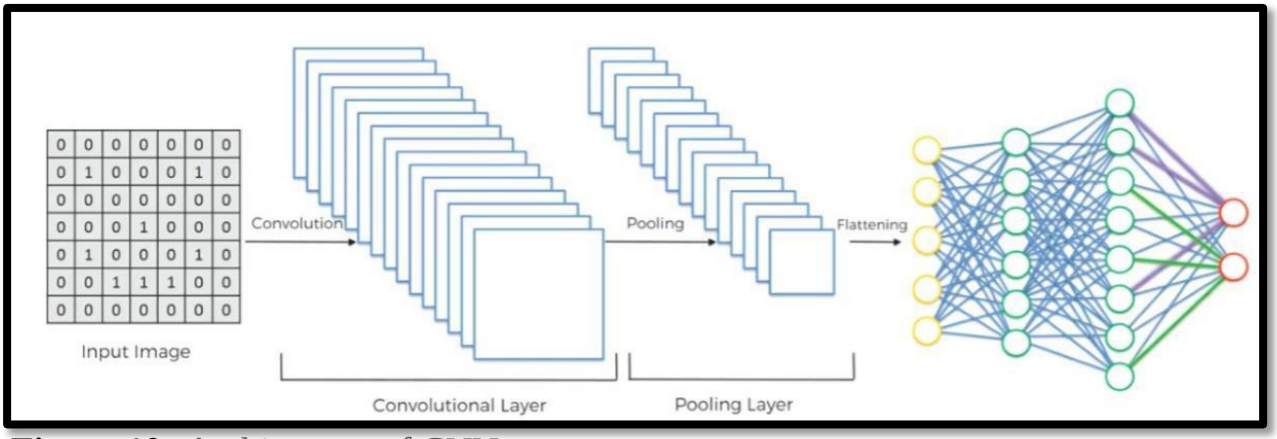


Figure 10: Architecture of CNN [5].

Convolutional layer (First Layer)

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), each of these filters can be thought of as feature identifiers which have a small receptive field, but extend through the full depth of the input volume.

During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

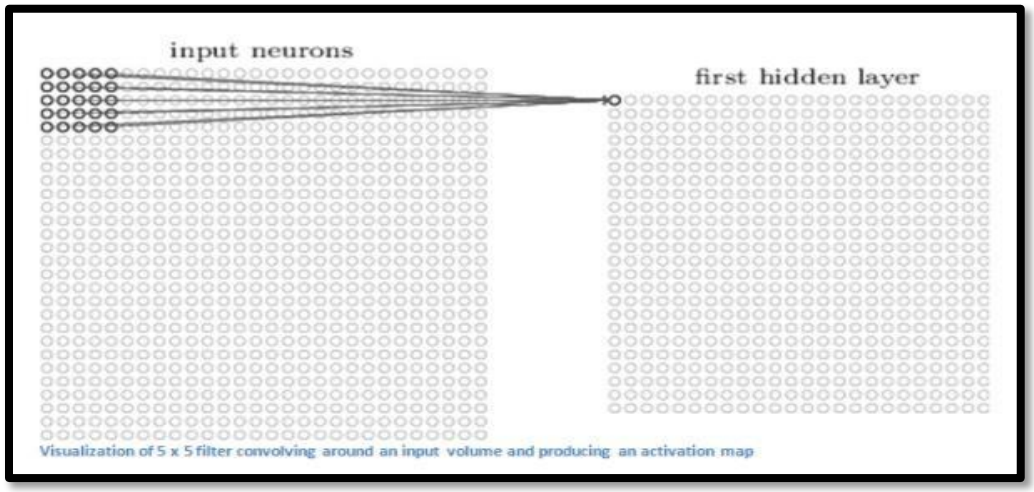


Figure 11: apply 5x5 filter convolving input to produce Activation map [2].

To visualizing this mathematically. When we have this filter at the top left corner of the input volume, it is computing multiplications between the filter and pixel values at that region. Now let's take an example of an image that we want to classify, and let's put our filter at the top left corner

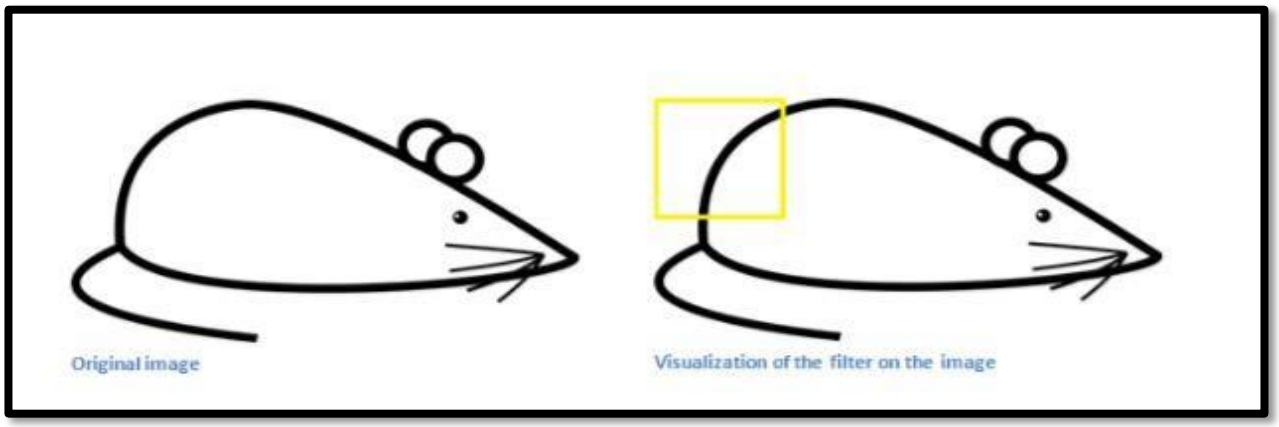


Figure 12: apply the filter convolving input [2].

Remember, what we have to do is multiply the values in the filter with the original pixel values of the image.

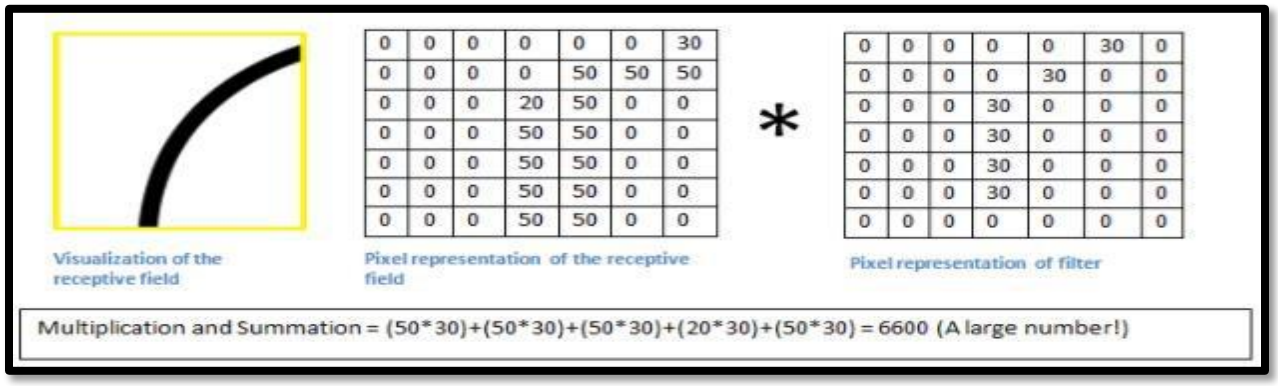


Figure 12: apply filter with the original pixel values of the image [2].

Basically, in the input image, if there is a shape that generally resembles the curve that this filter is representing, then all of the multiplications summed together will result in a large value! Now let's see what happens when we move our filter.

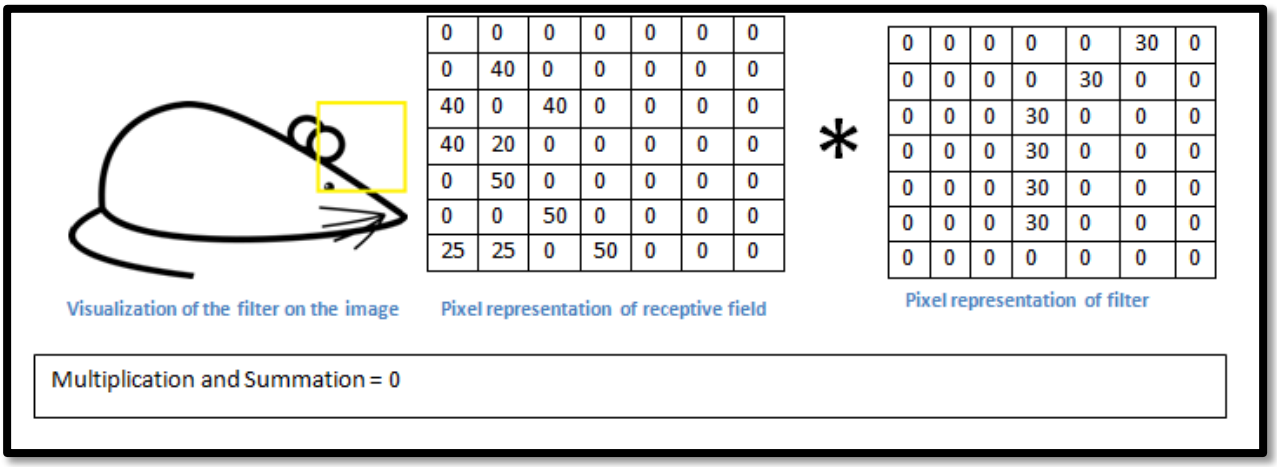


Figure 14: we move our filter to another original pixel values of the image [2].

Pooling layer

Another important concept of CNNs is pooling, which programmers choose to apply a pooling layer. It is also referred to as a non-linear down-sampling layer. In this category, there are also several layer options, with "max-pooling" being the most popular. This basically takes a filter (normally of size 2x2) and a stride of the same length.

It then applies it to the input volume and outputs the maximum number in every sub-region that the filter convolves around.

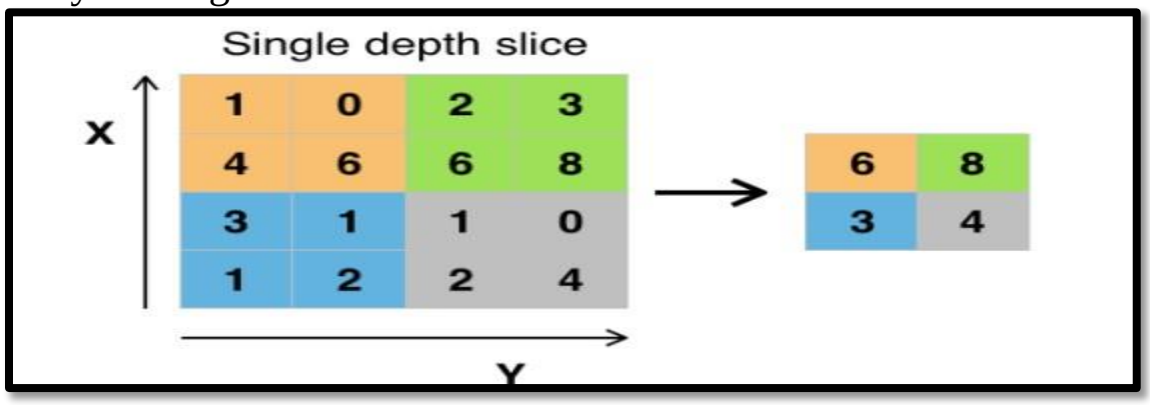


Figure 15: Max-pool with a 2x2 filter and a stride 2 [2].

Flatten layer

Using the pooled feature map we flatten it into one column. Starting from the top
row, left to right.

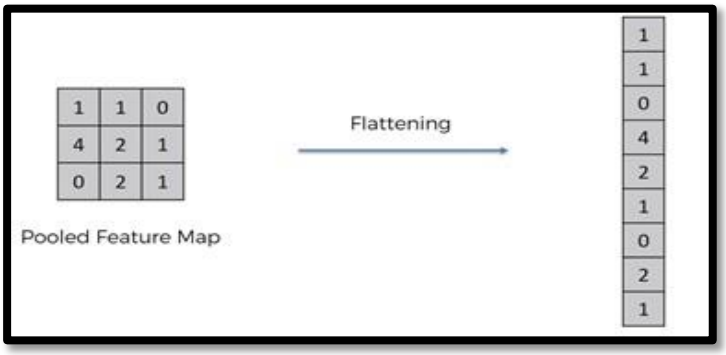


Figure 16: flatten Features map to one vector [2].

This is done so that we can fit this into an Artificial Neural Network for further processing.

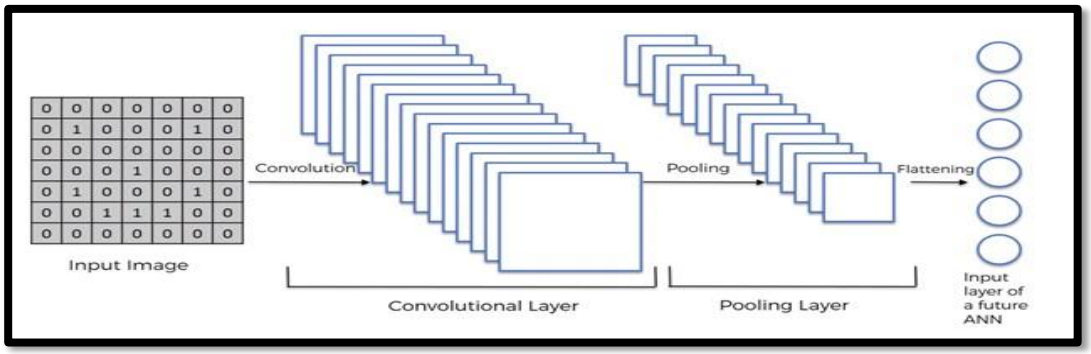


Figure 17: All Layers of CNN [2].

Fully Connected layer

This adds an Artificial Neural Network onto the flattened input image. The hidden layers inside a Convolutional Neural Network are called Fully Connected Layers. These are a specific type of hidden layer which must be used within the CNN.

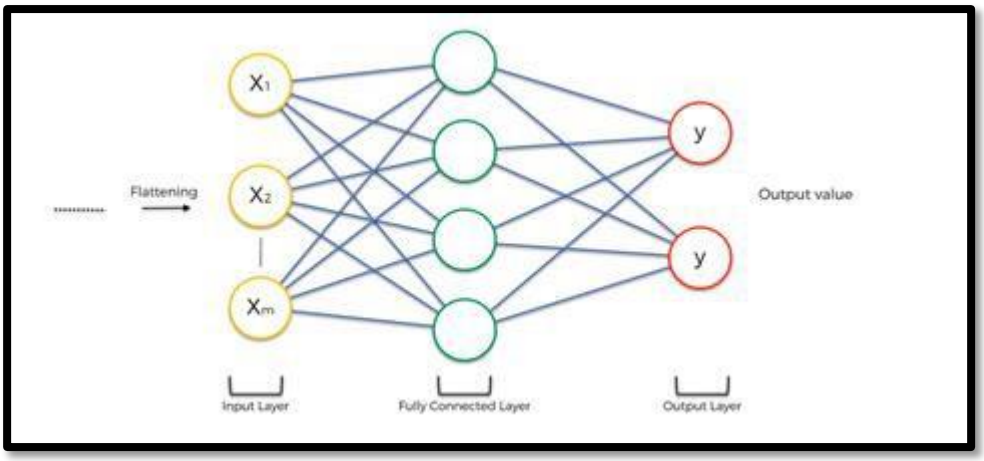


Figure 18: Fully Connected Layers [2].

Using Fully Connected Layers the output layer is passed into the input layer. This is used to combine the features into more attributes that predict the outputs (classes) more accurately. Once flattened, some features are already encoded from the vector but the ANN builds on that and improves it.

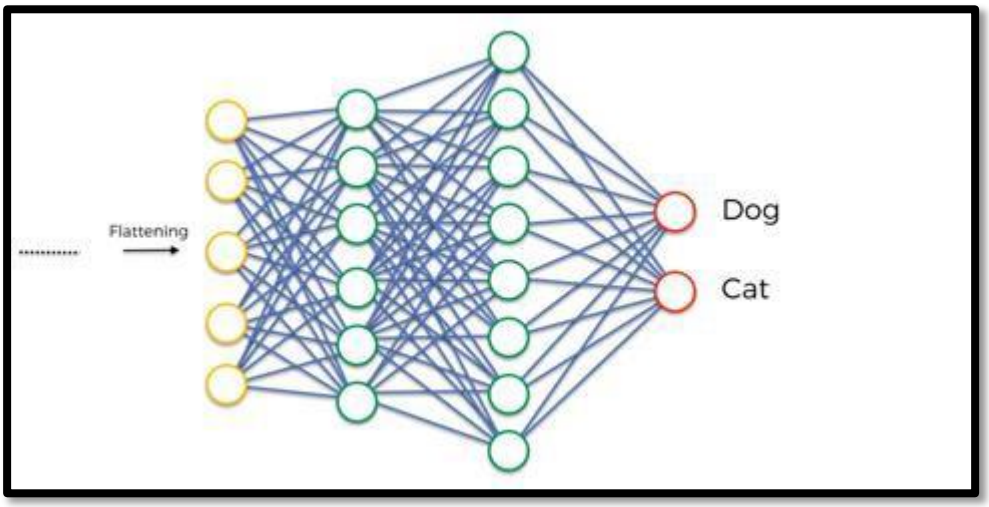


Figure 19: fully connected layer representation [2].

Loss Functions

Errors in CNNs are called a Loss Function which uses Cross-Entropy and mean- squared.

When the CNN is Back-propagated we adjust the weights and adjust the feature map to determine if we are using the correct one. Having multiple output layers, we need to understand what weights we assign to the synapses to each output

SoftMax Function

The SoftMax function is used to provide output values with a probability between each of them. For example: there is an image of a dog that has been passed through a CNN, the machine decides that the image is 0.95 likely to be a dog and 0.05 likely to be a cat.

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

The original value is squashed into much smaller values that both add up to 1 to allow the machine to provide a suitable probability of each image.

Cross-Entropy Function

Cross-Entropy is a function that comes hand-in-hand with SoftMax. The original formula is:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

The function we are going to use (as it's easier to calculate) is:

$$H(p,q) = -\sum_x p(x)\,\log q(x)$$

A Cross-Entropy function is used to calculate the Loss Function and helps to get a neural network to an optimal state. This method is the preferred method for classification. If using regression, you would use Mean Squared Error.

Algorithms

The main goal of training the deep convolution neural network is to find the suitable weights to map the input to desired output. Training process accomplished by pass the input to the input layer in CNN and compare the output- with desired output and calculate the error. Then updating the weights using the pre-calculated error by propagate the error from output node back to lower layers.

A famous Algorithm used to train and optimize CNN is Back-propagation algorithm that divide into two phases:-
Forward-propagation.
Back-propagation and update weights.

Forward propagation

Convolution Operation (Convolution Layer)

To perform a convolution operation, the kernel slid across the input feature map in equal and finite strides. At each location, the product between each element of the kernel and the input feature map element it overlaps is computed and the results summed up to obtain the output at that current location.

This procedure is repeated using different kernels to form as many output feature maps as desired. The concept of weight sharing is used as demonstrated in the diagram below:

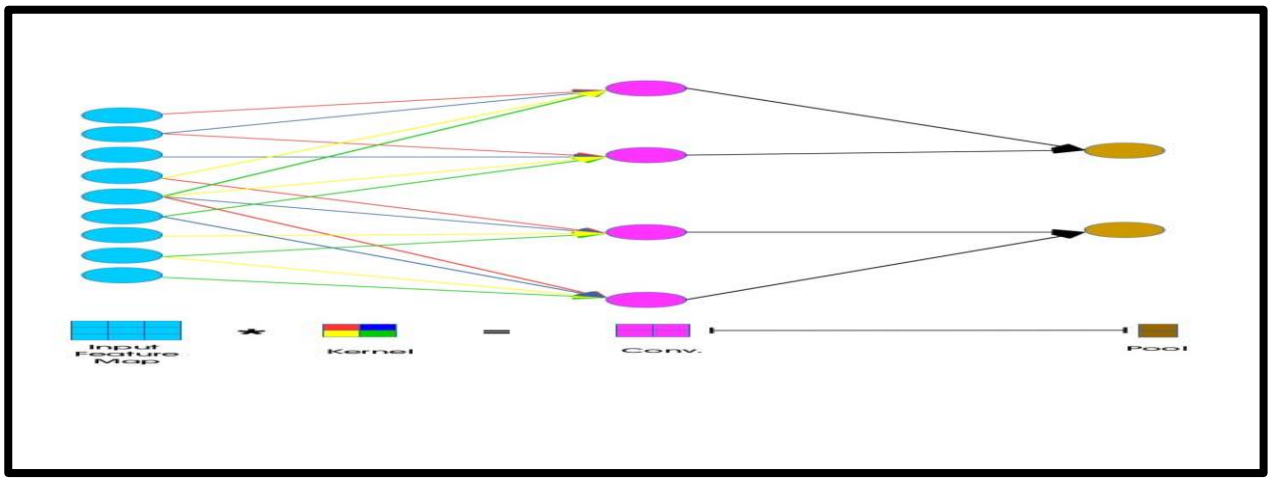


Figure 20: **Convolution Layer** operations [5].

Units in convolutional layer illustrated above have receptive fields of size 4 in the input feature map and are thus only connected to 4 adjacent neurons in the input layer. This is the idea of sparse connectivity in CNNs where there exists local connectivity pattern between neurons in adjacent layers.

The color codes of the weights joining the input layer to the convolutional layer show how the kernel weights are distributed (shared) amongst neurons in the adjacent layers. Weights of the same color are constrained to be identical.

The convolution equation of the input at layer all is given by:

$$x_{i,j}^{l} = \sum_{m} \sum_{n} w_{m,n}^{l} o_{i+m,j+n}^{l-1} + b_{i,j}^{l}$$

Then apply activation function:

$$o_{i,j}^{l} = f(x_{i,j}^{l})$$

Like: RELU = MAX (0, X).

This is illustrated below:

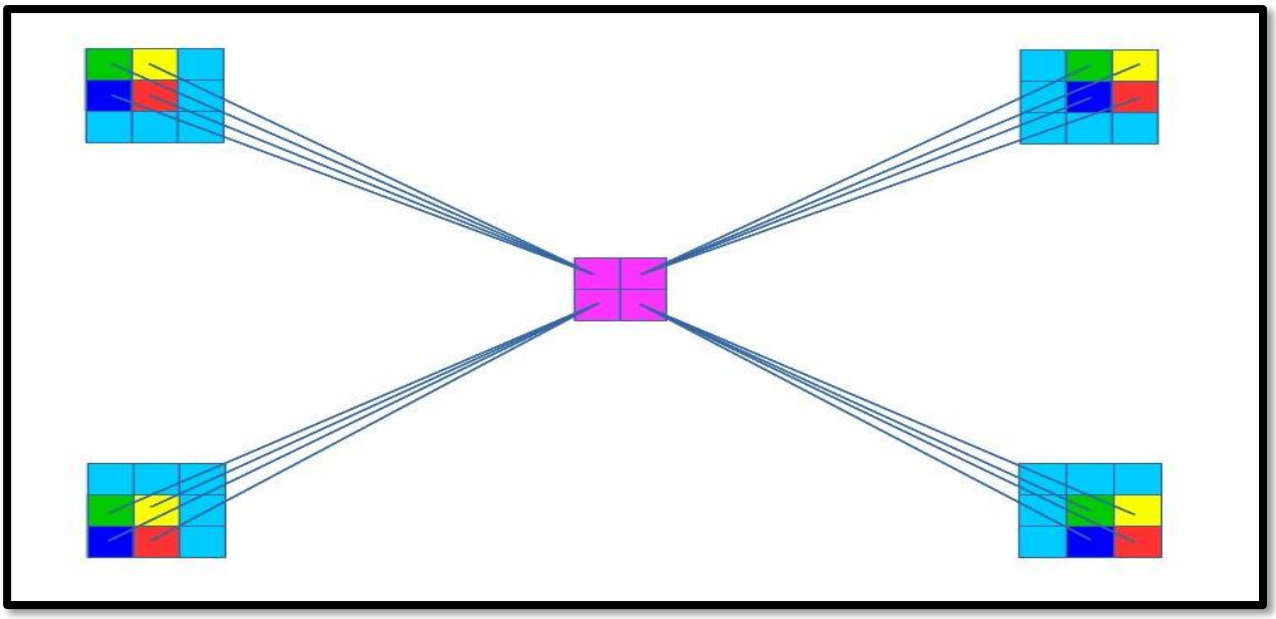


Figure 21: Convolution Layer operations details [2].

Pooling operation (Pooling Layer)

The function of the pooling layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control over fitting. No learning takes place on the pooling layers.

Pooling units are obtained using functions like max-pooling, average pooling and even L2-norm pooling but max pooling being the most popular.

This basically takes a filter (normally of size 2x2) and a stride of the same length. It then applies it to the input volume and outputs the maximum number in every sub region that the filter convolves around as shown below.

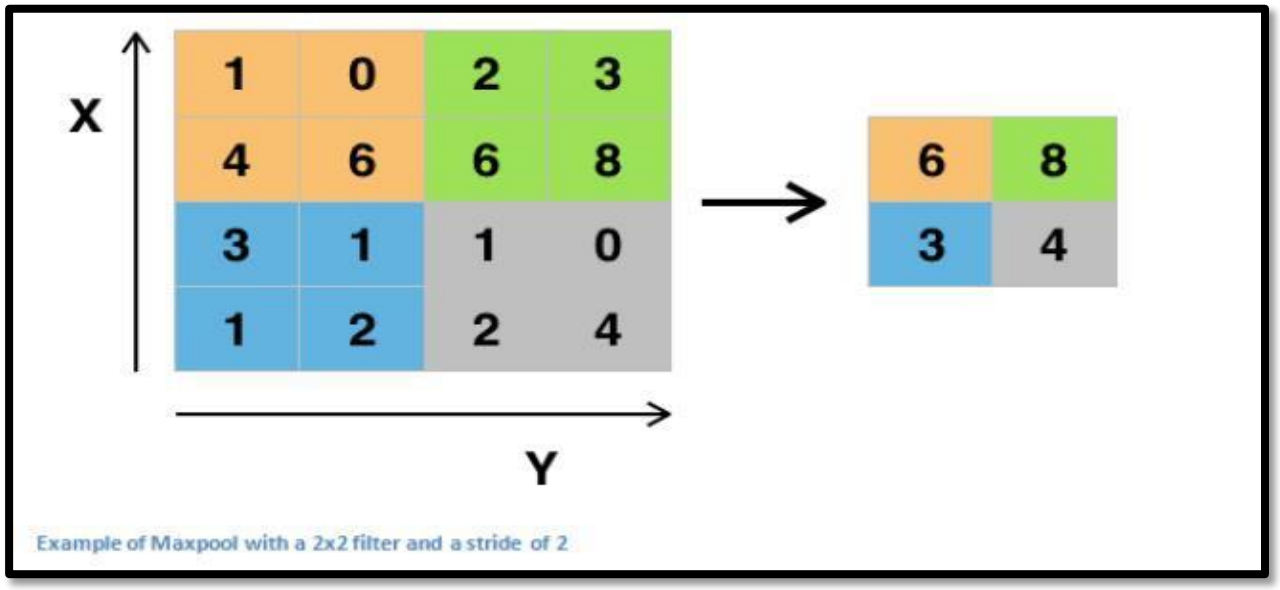


Figure 22: **max-pooling** operation [2].

Fully Connected Layer

Neurons in a fully connected layer have full connections to all activations in the
previous layer, as seen in regular Neural Networks.
Soft-max regression applies soft-max nonlinearity to the output of the network.

Back propagation and update weights

Error

For a total of P predictions, the predicted network outputs yp and their corresponding targeted values tp the mean squared error is given by:

$$E = \frac{1}{2} \sum_{p} (t_p - y_p)^2$$

Learning will be achieved by adjusting the weights such that yp is as close as possible or equals to corresponding tp. In the classical back-propagation algorithm, the weights are changed according to the gradient descent direction of an error surface E.

Back-propagation

We are looking to compute $\frac{\partial E}{\partial w^l_{m',n'}}$ which can be interpreted as the measurement of how the change in a single pixel ′wm′,n′ in the weight kernel affects the loss function E.

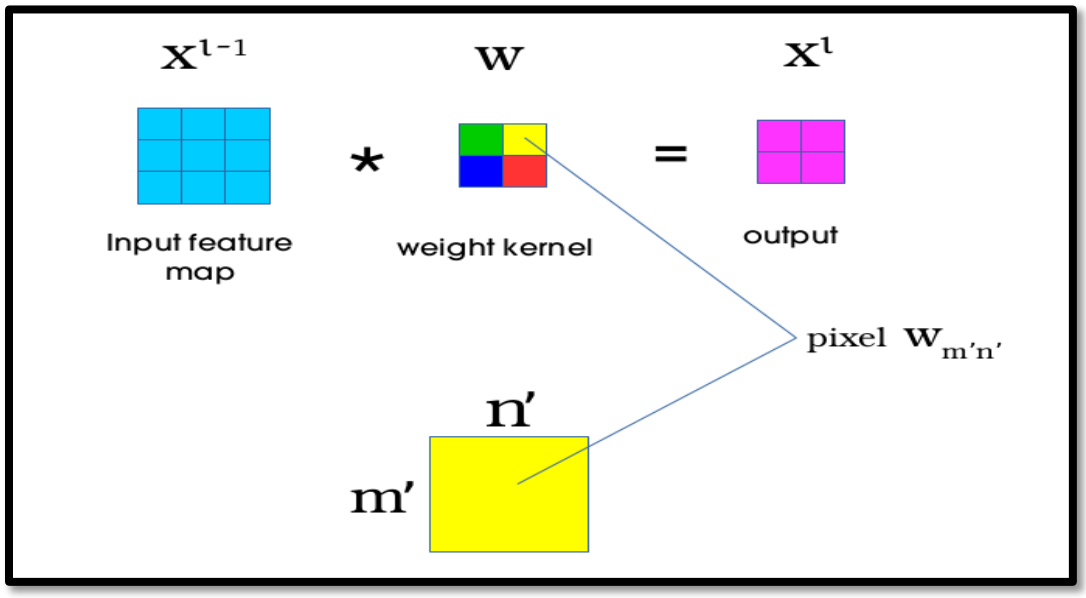


Figure 23: Back-propagation details [2].

Assume that get δO as input(from backward pass f the next layer) and our aim to calculate δW, by product matrix of gradient decent of output and input feature map.
Update weights of this kernel with this formula :

$$\frac{\partial E}{\partial w^l_{m',n'}} = \sum_{i=0}^{H-k_1} \sum_{j=0}^{W-k_2} \delta^l_{i,j} o^{l-1}_{i+m',j+n'}$$

The diagram below shows gradients (δ11, δ12, δ21, δ22) generated during back- propagation:

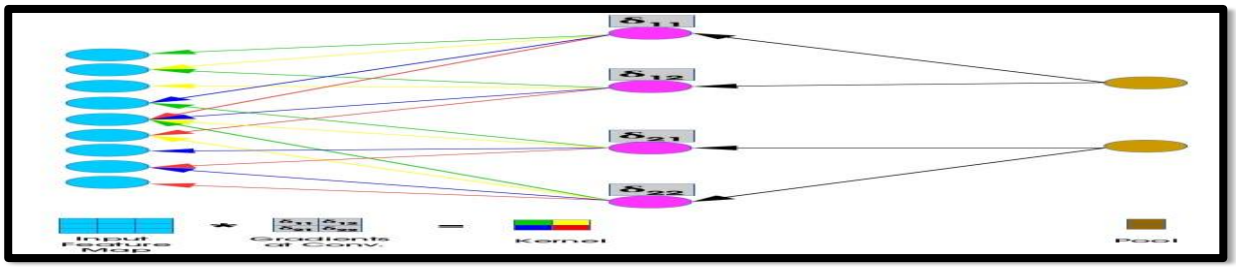


Figure 23: Gradients generated during Back-propagation [2].

The convolution operation used to obtain the new set of weights as is shown below:

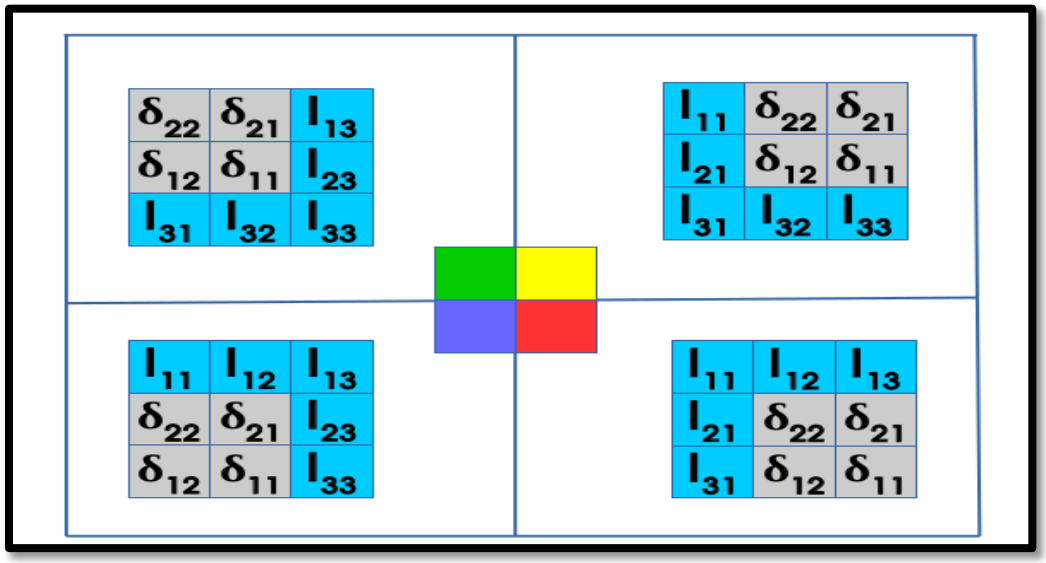


Figure 24: Obtaining the new set of weights [2].

3- Analysis and Des*ign*

Abstract

We review some of the most recent approaches to colorize gray-scale images using deep learning methods. Inspired by these, we propose a model which combines a deep Convolutional Neural Network trained from scratch with high-level features extracted from the InceptionResNet-v2 pre-trained model. Thanks to its fully convolutional architecture, our encoder-decoder model can process images of any size and aspect ratio. Other than presenting the training results, we assess the "public acceptance" of the generated images by means of a user study.

Architecture

Our model owes its architecture to [16]: given the luminance component of an image, the model estimates its a*b* components and combines them with the input to obtain the final estimate of the colored image. Instead of training a feature extraction branch from scratch, we make use of an Inception-ResNetv2 network (referred to as Inception hereafter) and retrieve an embedding of the gray-scale image from its last layer. The network architecture we propose is illustrated in Fig. 25.

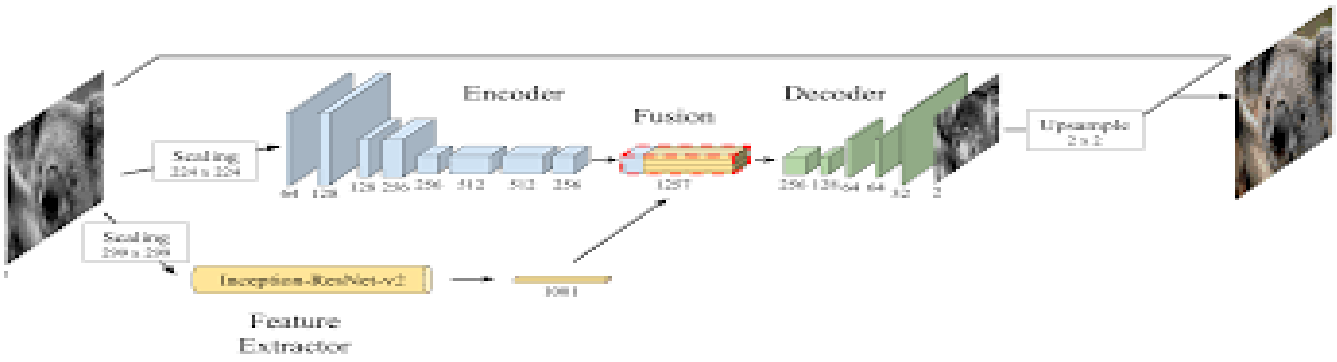


Figure 25: System Architecture [1].

The network is logically divided into four main components. The encoding and the feature extraction components obtain mid and high-level features, respectively, which are then merged in the fusion layer. Finally, the decoder uses these features to estimate the output. Table 1 further details the network layers.

Table 1: Left: encoder network, mid: fusion network, right: decoder network. Each convolutional layer uses a ReLu activation function, except for the last one that employs a hyperbolic tangent function. The feature extraction branch has the same architecture of Inception-Resnet-v2, excluding the last softmax layer.

Layer (Encoder)	**Kernels**	**Stride**	**Layer (Fusion)**	**Kernels**	**Stride**	**Layer (Decoder)**	**Kernels**	**Stride**
conv	64 × (3 × 3)	2 × 2	fusion	-	-	conv	128 × (3 × 3)	1 × 1
conv	128 × (3 × 3	1 × 1	conv	256 × (1 × 1)	1 × 1	upsamp	-	-
conv	128 × (3 × 3)	2 × 2				conv	64 × (3 × 3)	1 × 1
conv	256 × (3 × 3)	1 × 1				conv	64 × (3 × 3)	1 × 1
conv	256 × (3 × 3)	2 × 2				upsamp	-	-
conv	512 × (3 × 3)	1 × 1				conv	32 × (3 × 3)	1 × 1
conv	512 × (3 × 3)	1 × 1				conv	2 × (3 × 3)	1 × 1
conv	256 × (3 × 3)	1 × 1				upsamp	-	-

Preprocessing …To ensure correct learning, the pixel values of all three image components are centered and scaled (according to their respective ranges [26]) in order to obtain values within the interval of $[-1,1]$.

Encoder The Encoder processes H × W gray-scale images and outputs a H/8 × W/8 × 512 feature representation. To this end, it uses 8 convolutional layers with 3 × 3 kernels. Padding is used to preserve the layer's input size. Furthermore, the first, third and fifth layers apply a stride of 2, consequentially halving the dimension of their output and hence reducing the number of computations required.

Feature Extractor High-level features, e.g. "underwater" or "indoor scene", convey image information that can be used in the colorization process. To extract an image embedding we used a pre-trained Inception model. First, we scale the input image to 299×299. Next, we stack the image with itself to obtain a threechannel image (as shown in Fig. 2) in order to satisfy Inception's dimension requirements. Next, we feed the resulting image to the network and extract the output of the last layer before the softmax function. This results in a 1001×1×1 embedding

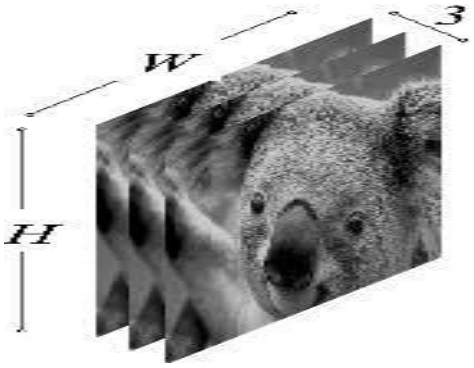


Figure 26: Stacking the luminance component three times .

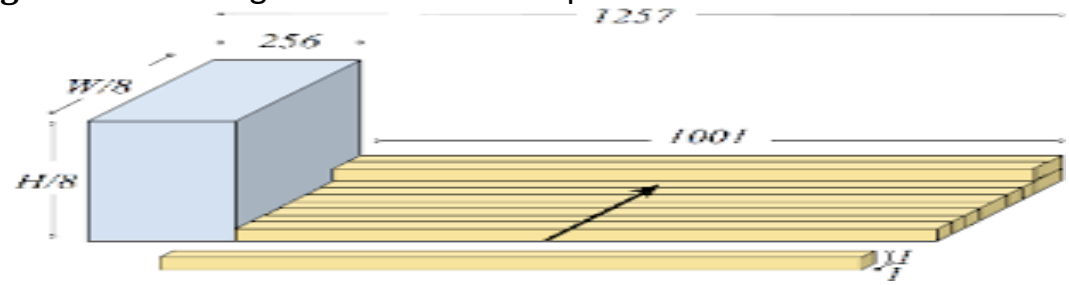


Figure 26: Fusing the Inception embedding with the output of the convolutional layers of the encoder.

Fusion The fusion layer takes the feature vector from Inception, replicates it HW/8 2 times and attaches it to the feature volume outputted by the encoder along the depth axis. This method was introduced by [16] and is illustrated in Fig. 3. This approach obtains a single volume with the encoded image and the mid-level features of shape H/8×H/8×1257. By mirroring the feature vector and concatenating it several times we ensure that the semantic information conveyed by the feature vector is uniformly distributed among all spatial regions of the image. Moreover, this solution is also robust to arbitrary input image sizes, increasing the model flexibility. Finally, we apply 256 convolutional kernels of size 1×1, ultimately generating a feature volume of dimension H/8×W/8×256.

Decoder Finally, the decoder takes this H/8×W/8×256 volume and applies a series of convolutional and up-sampling layers in order to obtain a final layer with dimension H × W × 2. Up-sampling is performed using basic nearest neighbor approach so that the output's height and width are twice the input's.

4- Experimental Results

After installing tools and libraries as we mentioned and built a good background knowledge about CNN our experimental results journey start with..

Training

Of the approx. 3000 original images, we held out the 13% to be used as validation data during training. The results presented in this report are drawn from this validation set and therefore the network never had the chance to see those images during training RMSprop optimizer was used during approximately 2 hours of training.

Colorizing using Cloud Service (Kaggel):

- Kaggel is a free cloud service and now it supports free GPU .
- Kaggel allows you to use and share Jupyter notebooks with others without having to download, install, or run anything on your own computer other than a browser.
- Improve your Python programming language coding skills.
- Develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV.
- The most important feature that distinguishes Kaggel from other free cloud services is Kaggel provides GPU and is totally free with 13GB RAM.
- Free Tesla K80 GPU- using Keras, Tensorflow and PyTorch".

Steps:

- We've done 100 iterations on MIT Dataset 1500 images “Forest Road” category.

- Then started training on 1500 images “Forest Road , Gardens ” category from MIT Dataset in addition to 1500 images of ImageNet Dataset of “Geological Formation (Cliff) “ category with 50 iterations.

- Finally we've done 50 iterations on 1000 images “Forest Road , Gardens” category from MIT Dataset and 1000 images of ImageNet Dataset of “Geological Formation (Cliff) “ category adding 1000 images of “Historical Abbey” category we have collected from google images .

- Graph of Training and Testing loses and Accuracy of our model.

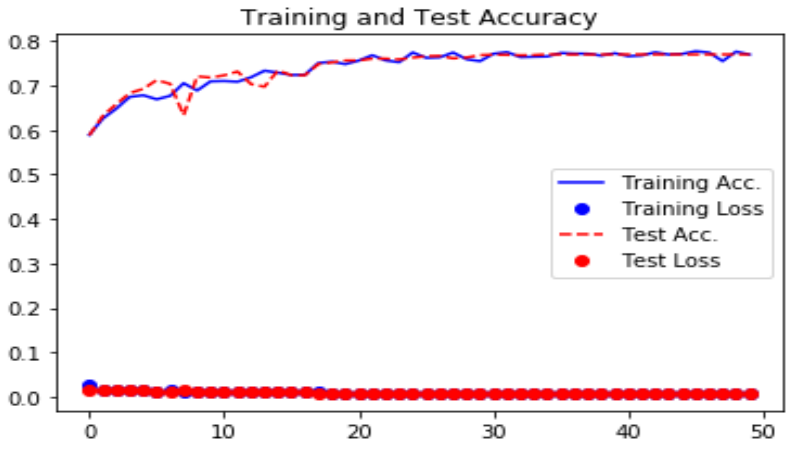
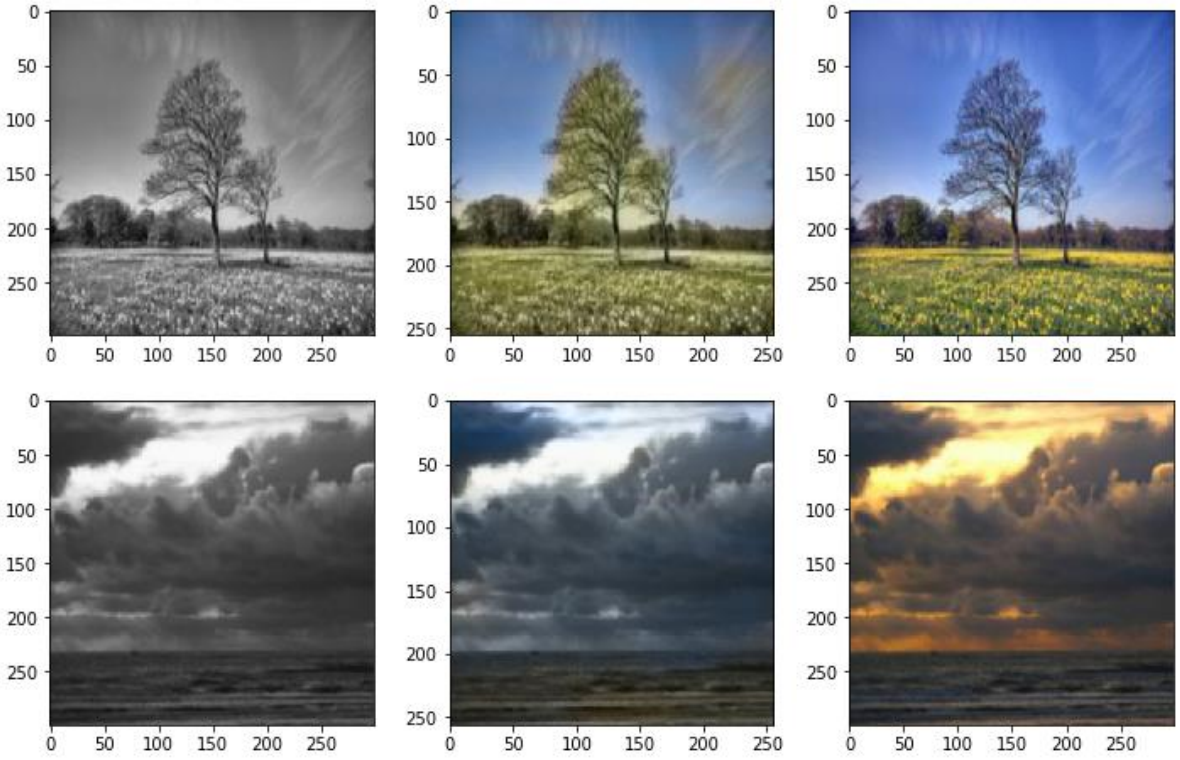


Figure 27: History of Training

Final Results

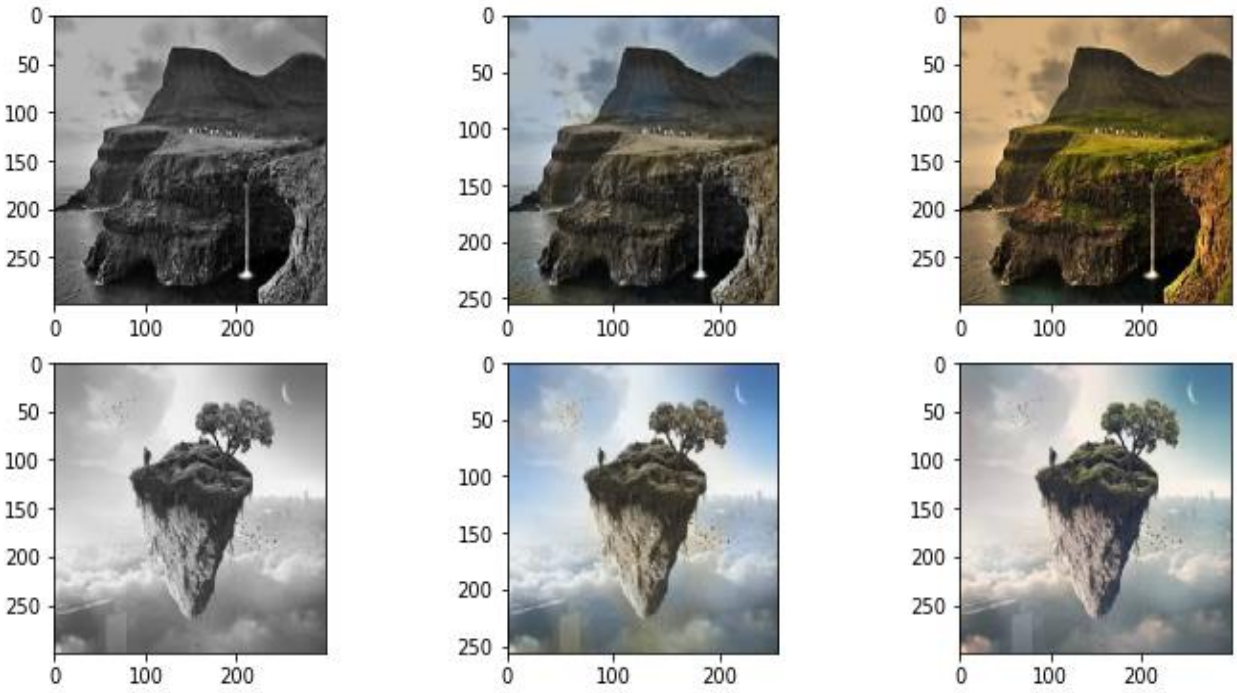
- **From MIT Dataset:**

GrayScale Ours Ground Truth



- **From ImageNet Dataset:**

GrayScale | Ours | Ground Truth



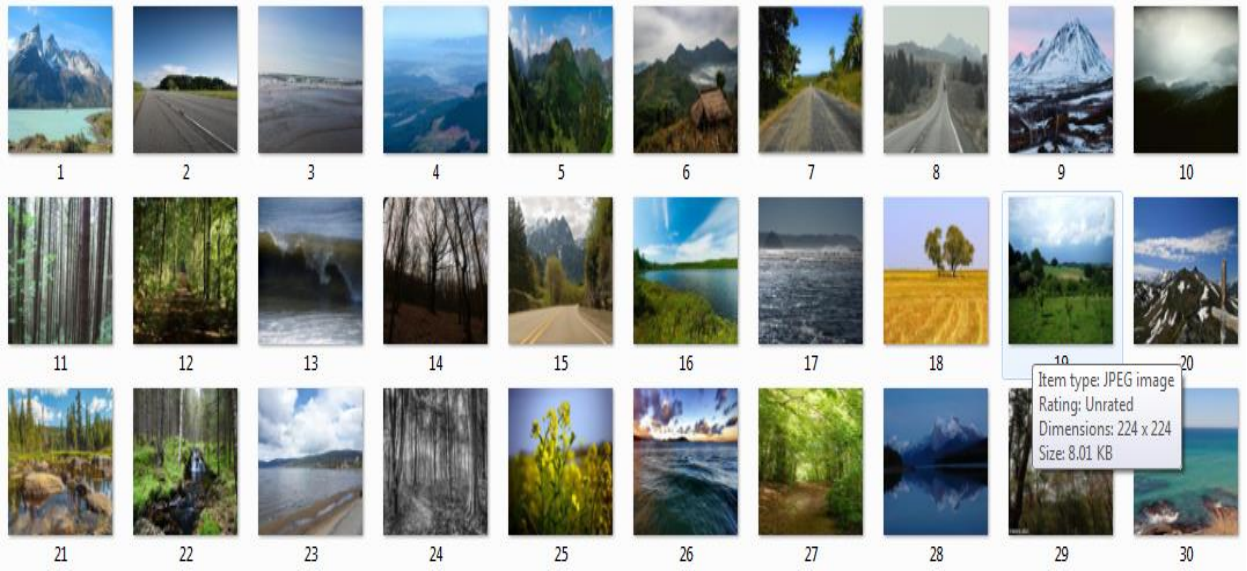
- **From Google Images Dataset:**

GrayScale Ours Ground Truth

Datasets

MIT Places Dataset : Places Dataset contains more than 10 million images comprising 400+ unique scene categories. The dataset features 5000 to 30,000 training images per class, consistent with real-world frequencies of occurrence.

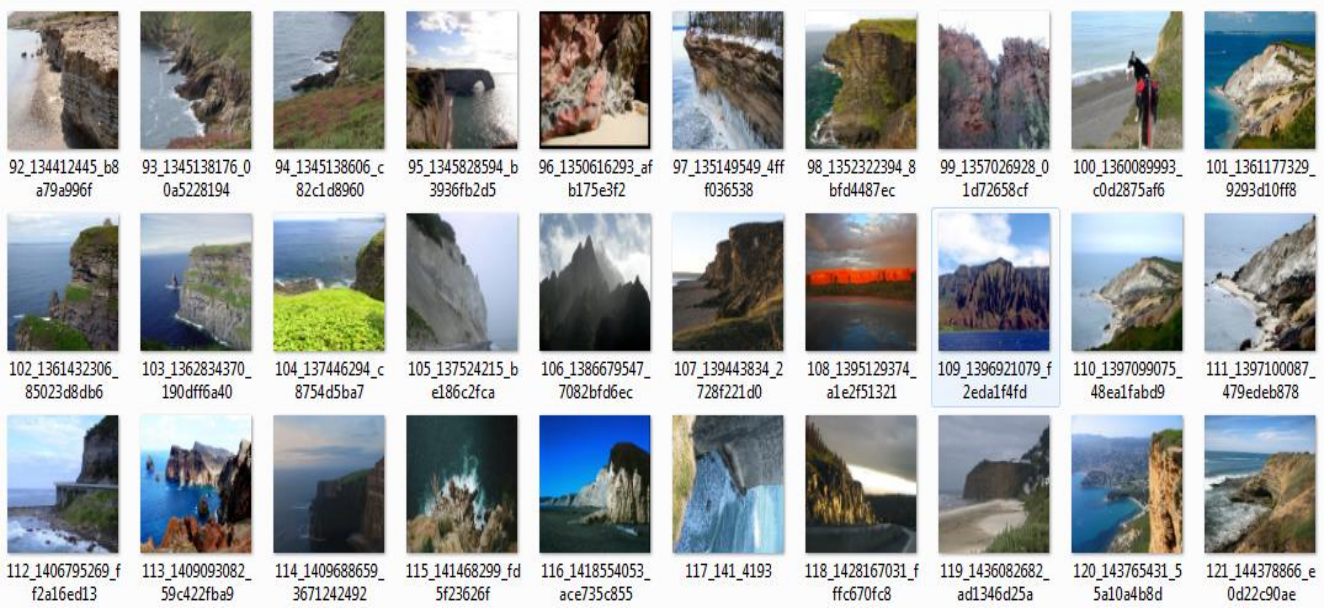
We Select a category of 3000 images “Forest Road , Gardens and Beaches ” categories.



Google Images Dataset :

We collect 1500 images of “Historical Abbey ” category from google images .

ImageNet Dataset : ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images, it contains over 14Milion images and more than 20K synsets.
We Select a category of 1500 images “Geological Formation (Cliff) ” category.



5- User Manual

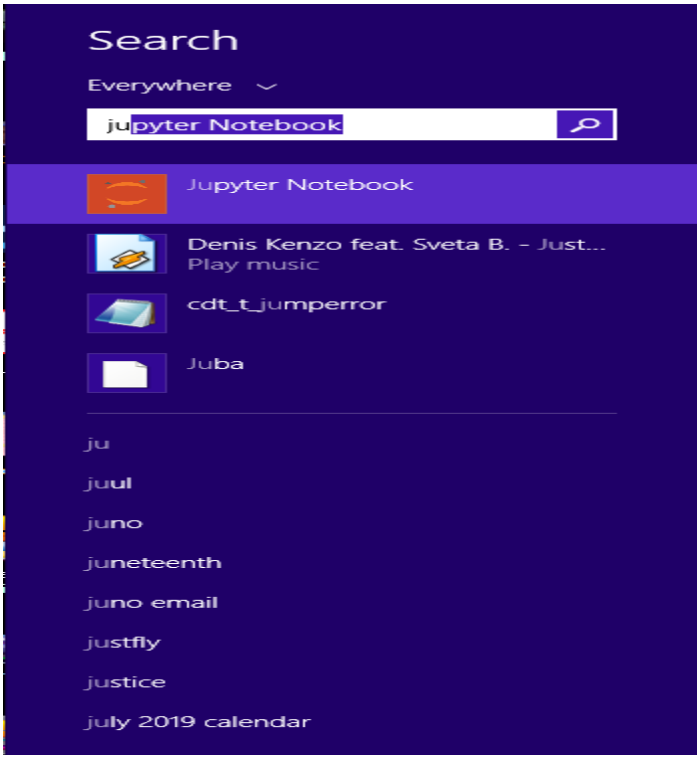
In this chapter we will describe how to operate the project along with representing all steps of System Testing section and installation guide in System deployment section.

System Testing :-

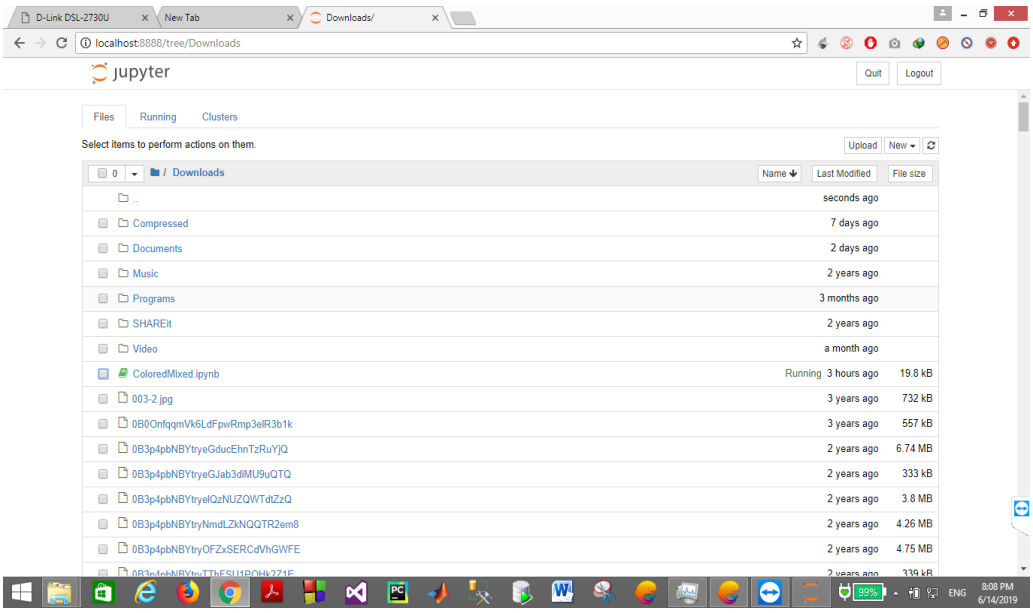
Firstly you should have a grayscale image , and if you have a colored one you should convert it to grayscale so you can test the model easily ,also its better to have an image of the same categories we mentioned to get a good result.

Steps to test:

1- Open Jupyter Notebook..



2- Locate the path to the directory of the (ColoredMixed.ipynb) file and open it..



3- Run the first cell to import libraries..

4- Run the second cell to load inception weights..

5- Here is all function needed for preprocessing in the third cell..

6- The function to build the architecture in the fourth cell..

7- Run the fifth cell to build and compile the model..

8- The test function used for colorization in the sixth cell..

9- Run the last cell to display the colored image..

System Deployment:-

This chapter include an "Installation Guide" that would describe how to install the program, and all required third party tools that needs to be available for the project to run.

Installation steps:

1. setup the **Python environment**
   From the python site from here https://www.python.org.

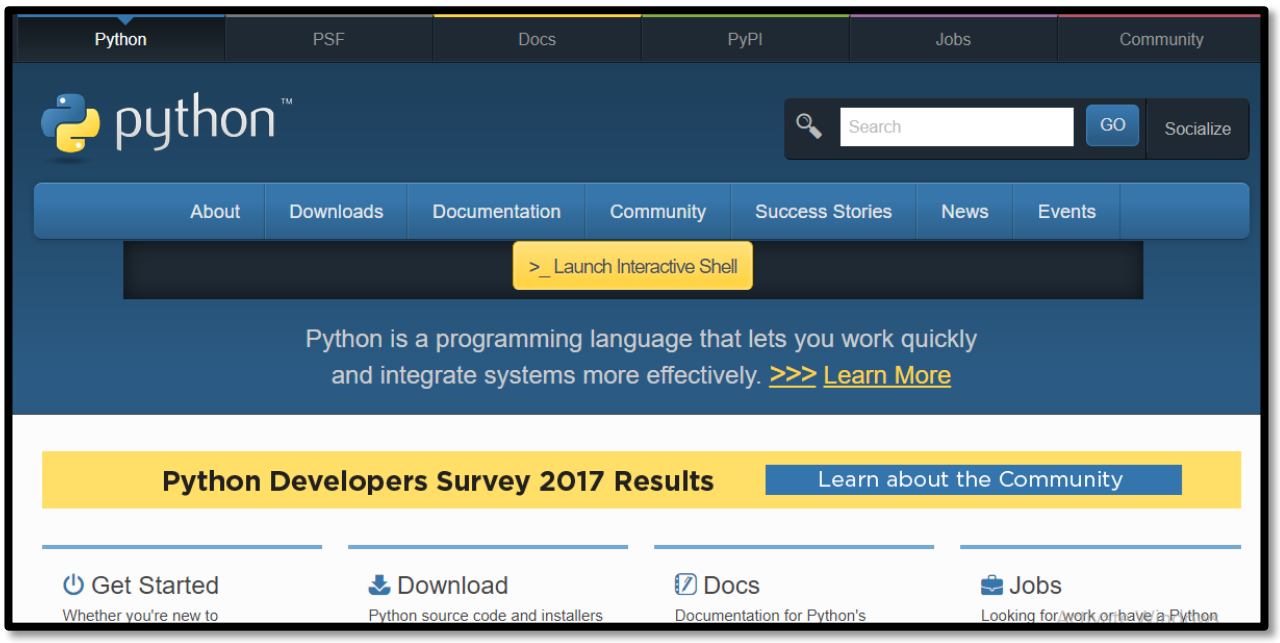


Figure 7: Python website

2. Setup **Anaconda environment** and you can download from here *https://www.anaconda.com/download*

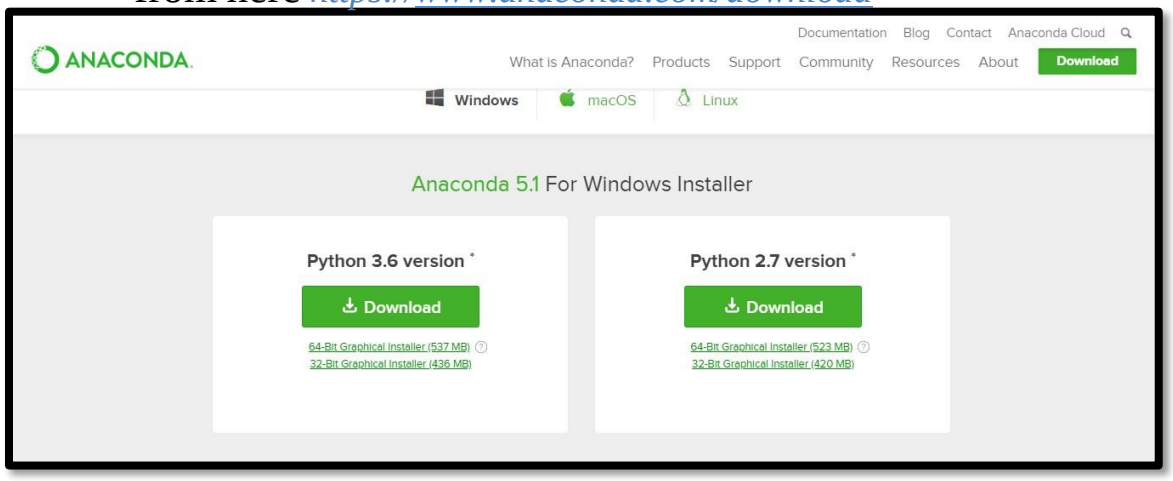


Figure 7: Anaconda website

3. Install the libraries that you will use. You can install **TensorFlow** from here *https://www.tensorflow.org/*

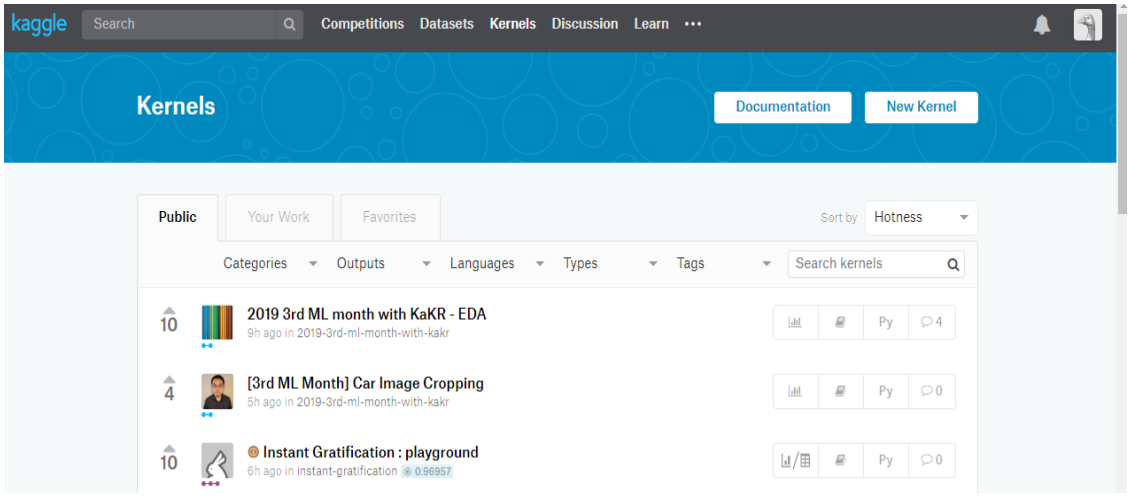
Figure 8: TensorFlow website

Using Cloud services :-

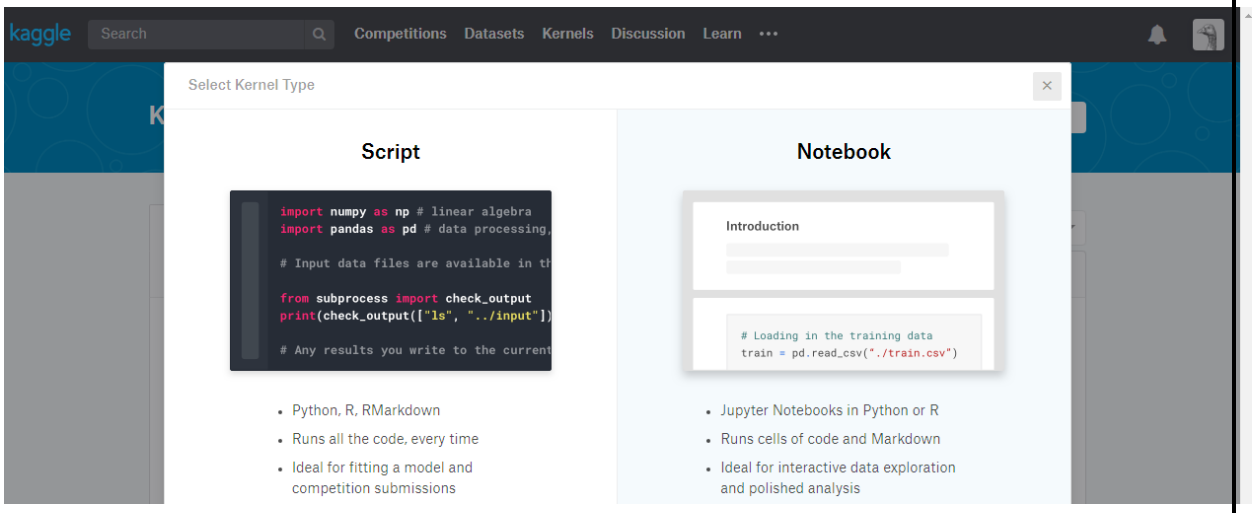
Steps to use Kaggle :

1- Open Kaggle website from this link https://www.kaggle.com/.

2- After signing up , go to kernels and open new kernel..



3- You can choose to start a script or a notebook..



4- Now the kernel started..

5- You can follow this video to use the kernel..
https://www.youtube.com/results?search_query=kaggle

6- Conclusion and Future Work

This project validates that an end-to-end deep learning architecture could be suitable for some image colorization tasks. In particular, our approach is able to successfully color high-level image components such as the sky, the sea or forests. Nevertheless, the performance in coloring small details is still to be improved. As we only used a reduced subset of ImageNet, only a small portion of the spectrum of possible subjects is represented, therefore, the performance on unseen images highly depends on their specific contents. To overcome this issue, our network should be trained over a larger training dataset.

We believe that a better mapping between luminance and a*b* components could be achieved by an approach similar to variational autoencoders, which could also allow for image generation by sampling from a probability distribution.

Finally, it could be interesting to apply colorization techniques to video sequences, which could potentially re-master old documentaries. This, of course, would require adapting the network architecture to accommodate temporal coherence between subsequent frames. Overall, we believe that while image colorization might require some degree of human intervention it still has a huge potential in the future and could eventually reduce hours of supervised work.

As we mentioned before due to the **limited resources** like Computing Power (GPUs, RAM, etc.) , we have trained the model on just 3 category with a relatively small number of image, and you may look for powerful resources and huge number of dataset to fine-tune on the model and worked on many categories.

Actually we use free Cloud (Kaggle) for powerful resources , Next important step to use paid cloud resources for more control and support like Amazon **or** IBM cloud.
It is very important to design a UI in order to make the test easy for users.

7- References

- Baldassarre, Federico, Diego González Morín, and Lucas Rodés-Guirao. "Deep koalarization: Image colorization using cnns and inception-resnet-v2." *arXiv preprint arXiv:1712.03400* (2017).

- Zhang, Richard, Phillip Isola, and Alexei A. Efros. "Colorful image colorization." In *European conference on computer vision*, pp. 649-666. Springer, Cham, 2016.

- Tutorial Convolutional Neural Network based Image Colorization using OpenCV, last access date 08 Apr 2019, access link.

- CoLab ://colab.research.google.com/notebooks/welcome.ipynb

- Kaggle https://www.kaggle.com/

- Project https://github.com/ahmednasser1911/Grayscale-image-Colorization

- ImageNet http://image-net.org/

- MIT Places http://places2.csail.mit.edu/