B.E. PROJECT REPORT

TOWARDS COMMERCIAL DNA STORAGE

Submitted by:

Anurag Singh
713 / IT / 14

Under the guidance of

Dr. Deepak Kumar Sharma

in partial fulfillment of requirements of award of

B.E. (INFORMATION TECHNOLOGY)
DEGREE OF UNIVERSITY OF DELHI

DIVISION OF INFORMATION TECHNOLOGY
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
UNIVERSITY OF DELHI

2014 - 2018

नेताजी सुभाष प्रौद्योगिकी संस्थान
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
(An Institution of Govt. of NCT of Delhi-Formerly, Delhi Institute of Technology)
Azad Hind Fauj Marg, Sector-3, Dwarka, New Delhi - 110 078
Telphone : 25099050; Fax : 25099025, 25099022 Website : http://www.nsit.ac.in

CERTIFICATE

This is to certify that the work in the project report **Towards Commercial DNA Storage** is the bonafide work of **Anurag Singh** student of eighth semester, B.E. Information Technology, who carried out the project work under my supervision.

This project work has been prepared as a partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology, University of Delhi.

This work has not been presented for any other academic purpose before. I wish them luck for all their future endeavours.

Dr. Deepak Kumar Sharma
Assistant Professor
Division of Information Technology
NSIT Delhi

Dated :

CANDIDATES' DECLARATION

This is to certify that the work which is being hereby presented by us in this project titled "Towards Commercial DNA Storage" in partial fulfilment of the award of the degree of Bachelor of Engineering submitted at the Department of Computer Information Technology , Netaji Subhas Institute of Technology Delhi, is a genuine account of our work carried out during the period from January 2018 to May 2018 under the guidance of Dr. Deepak Kumar Sharma, Department of Information Technology, Netaji Subhas Institute of Technology, Delhi. The matter embodied in the project report to the best of our knowledge has not been submitted for the award of any other degree elsewhere.

Anurag Singh (713 / IT / 14)

Dated**:**

This is to certify that the above declaration by the students is true to the best of my knowledge.

Dr. Deepak Kumar Sharma

ACKNOWLEDGEMENT

No significant achievement can be done by solo performance especially when starting a project from ground up. This B.E. Project, on such a revolutionary idea, has by no means been an exception. It took many special people to enable it and support it. Here we would like to acknowledge their precious co-operation and express our sincere gratitude to them.

Dr. Deepak Kumar Sharma has again been very supportive and involved in yet another student project. It was his support that helped the project to start in its earliest and most vulnerable stages. His name opened many doors for us and persuaded many people. He was always found with energy and enthusiasm to make sure that we were provided everything we needed. No amount of words can express thanks to him. He was the one who backed us in providing any assistance we needed during the project work. I did this work with Naman Malhotra from BT department and also with tutelage of **Dr. Sonika Bhatnagar** so I'll use "we' instead of 'I'

We are also thankful to our friends who motivated us at each and every step of this project. Without their interest in our project we could not have been gone so far.

It was a great pleasure and honour to spend our time with all of them and there could be no better payment for the efforts put into completing this B.E. Project than their valuable presence. They are all very special to us.

ABSTRACT

The project deals with the ways to make a synthetic DNA storage more commercial and help the people to get DNA synthesized for the information they wish to store. The project aimed to help create utilities for the computational biologists and other people to obtain a DNA format for their data. The two contributions of the project are:

The utility will be open sourced application where the user can upload their data and download their .DNA files. They can also give their DNA files to retrieve and download original information back. Once .DNA file is generated the user can get their information synthesized into DNA which are read and written to DNA by DNA sequencing companies.

During completion of the utility we found that huge amount of bases were required to encode even a very small image as images contain large amount of data. It is not a far fetched statement to say picture is worth a thousand works indeed. But much of that data is redundant to human eyes. The second goal of this project thus was to find ways to reduce the number of bases to store images which will allow cheaper storage of media other than text such as images. To give a rough estimate 2 Megabytes of data costs $7000 to synthesize. Therefore saving high resolution images in DNA would a far-fetched dream which has not been tried yet presently by research institutes across the globe because it would require unimaginable amount of grants and funds.If we can reduce the cost by even a small amount it would be a leap in making large scale commercial use of the existing technology at hand. We are trying to formalize novel ways of image compression using techniques of image corpus summarization, resizing and bayesian image super-resolution to compress images to use lesser bases and then retrieve maximum amount of information and intelligently fill the lost parts to again reproduce a very close estimate of original high-resolution image.

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

Problems worthy of attack prove their worth by fighting back.

- Piet Hein

Existing storage technologies cannot keep up with the modern data explosion. Thus, researchers have turned to fundamentally different physical media for alternatives. Synthetic DNA has emerged as a promising option, with theoretical information density of multiple orders of magnitude more than magnetic tapes. With current rise in pace with which we create data much of data storage facilities will be redundant. Storing data in synthetic DNA offers the possibility of improving information density and durability by several orders of magnitude compared to current storage technologies. However the field is still in its inception and research is being carried out to read or write DNA storage within matter of hours or even days. The "digital universe" (all digital data worldwide) is forecast to grow to over 16 zettabytes in 2017. Alarmingly, the exponential growth rate easily exceeds our ability to store it, even when accounting for forecast improvements in storage technologies. A significant fraction of this data is in archival form; for example, Facebook recently built an entire data center dedicated to 1 exabyte of cold storage.

Most of the world's data today is stored on magnetic and optical media like hard disks(SSD, HDD) or CDs. Tape technology has recently seen significant density improvements with tape cartridges as large as 185TB, and is the densest form of storage available commercially today, at about 10GB/mm³. Recent research reported feasibility of optical discs capable of storing 1PB, yielding a density of about 100GB/mm³. Despite this improvement, storing zettabytes of data would still take millions of units, and use significant physical space. But storage density is only one aspect of archival: durability is also critical. Rotating disks are rated for 3–5 years, and tape is rated for 10–30 years. This causes us to keep redundancy in memory systems to ensure that information is preserved (RAID architectures) causing more space coverage. It was not a issue when the technology was conceived because there was enough space to store the data being generated in the world. Current long-term archival storage solutions require refreshes to scrub corrupted data, to replace faulty units, and to refresh technology. If we are to preserve the world's data, we need to seek significant advances in storage density and durability.

The current interests of the time are to use synthetic DNA for digital storage. There are multiple aspects and areas of problems to solve to make DNA storage commercial. Why are excited about storing information in DNA? How many of us can read the floppy drives now? It's very hard to read them because the equipments available to read them are hard to find now. We shift our

storage medium iteratively copying the information. But no method of storage is currently long term.The current storage systems shall render themselves redundant soon. So what is the solution? DNA at its worst has a half life of over 500 years and DNA data from over 50k years of unattended fossils have been successfully sequenced. With more than 90% of data of the world being generated in past 10 years no matter how much horizontally we scale our storage systems data will increase exponentially. Therefore the solution to offer to such massive data storage crisis is to find a media that can store huge amount of data in lot less space. Again we find DNA that a single molecule of DNA. And as with time digital information will continue to pile up the solution and need for an alternate will become more and more desperate. DNA thus offers multiple advantages as a potential storage media being immutable and dense means of storage it can store upto 2 bits in each nucleotide base in its theoretical max capacity which would mean 455 exabytes of information can be stored in one gram of DNA. As compared to rest of digital storage media or conventional storage which go obsolete in no time and are unreadable. Lot of such information becomes inaccessible partly because of time. DNA is often readable despite degradation in non ideal conditions over a long interval of time involving thousands of years. Since DNA has essential biological roles its reading and writing enzymes are to be present for eternity and hence can be written and read for any amount of foreseeable future.

Encoding a file in DNA requires several preprocessing steps, such as randomizing it using a pseudo-random sequence, partitioning it into hundred-character substrings, adding address and error correction information to these substrings, and finally encoding everything to the {A, C, G,T} alphabet. The resulting collection of short strings is synthesized into DNA and stored until needed.Progress in DNA storage has been rapid: in 1999, the state of-the-art in DNA-based storage was encoding and recovering a 23 character message; in 2013, researchers successfully recovered a 739 kB message. This improvement of almost 2×/year has been fueled by exponential reduction in synthesis and sequencing cost and latency; growth in sequencing productivity eclipses even Moore's Law. The volume of data that can be synthesized today is limited mostly by the cost of synthesis and sequencing, but growth in the biotechnology industry portends orders of magnitude cost reductions and efficiency improvements.

1.1 Motivation

We think the time is ripe to consider DNA-based storage seriously and explore system designs and architectural implications. A DNA storage system must overcome several challenges. First, DNA synthesis and sequencing is far from perfect, with error rates on the order of 1% per nucleotide. Sequences can also degrade while stored, further compromising data integrity. A key aspect of DNA storage is to devise appropriate encoding schemes that can tolerate errors by adding redundancy. Existing approaches have focused on redundancy but have ignored density

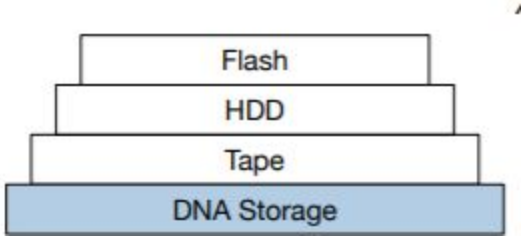
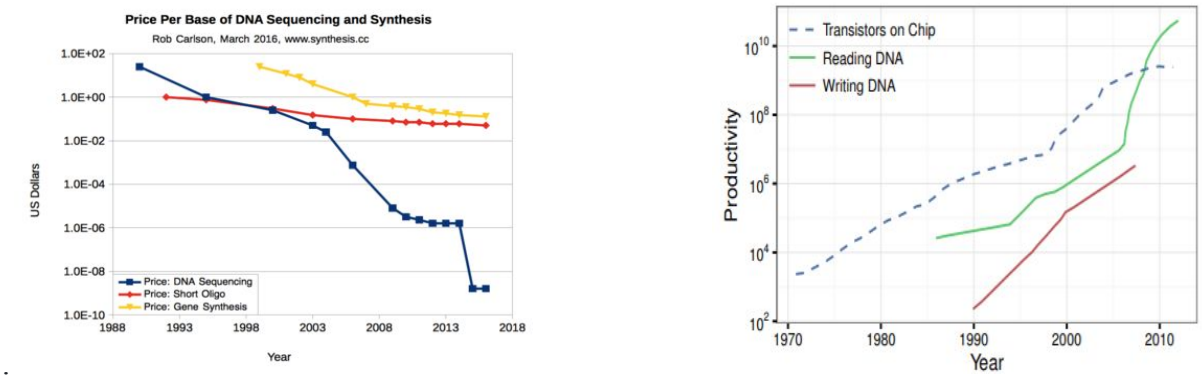


Fig 1.1: The ideal hierarchy of storage systems

implications. In this work we propose a new encoding scheme that offers controllable redundancy, enabling different types of data (e.g., text and images) to have different levels of reliability and density. The density of our encoding scheme outperforms existing work while providing similar reliability. Second, randomly accessing data in DNA-based storage is problematic, resulting in overall read latency that is much longer than write latency. Existing work has provided only large-block access: to read even a single byte from storage, the entire DNA pool must be sequenced and decoded.

In figure 1 it is shown how the hierarchy of storage systems would be after integration with the conventional silico storage systems such has hard disks and magnetic tapes. As we go down in the storage the volatility of the storage system must decrease and the stability of storage media must increase which must also translate into longer time to access the information. Which would mean slower read and write hence for fast computation and caching purposes DNA storage is unlikely to be used. It therefore isn't a means to make the current storage systems obsolete but to complement them such that they can handle much more information as compared to their current capacity. Thus DNA archival will act as a glacier storage which will need to accessed very infrequently but may require to be more durable. Typically the access time of flash is is microseconds and so is of the HDD somewhere around 10 ms the access from tape takes longer and DNA archival storage system when fully functional is expected to take couple of hours to be sequenced for the information to be read from the storage. But at the same time there is difference in durability of these storage systems. When flash storage is durable for ~ 5 years HDD for ~ 10 years and tape for 15 to 20 years DNA storage will be safe method to preserve information for centuries. As traditional storage and computation resources experiences Moore's Law similar trend is spotted in the DNA sequencing and DNA synthesis. The per-nucleotide productivity for the DNA systems have gone up throughout the course of time since 1970 and the sequencing of DNA which will act as a means of reading the information stored in DNA has become exponentially cheaper with time. According to article by R.Carlson who has kept record of how much productivity of the DNA sequencing and synthesis has increased over time a comparison is shown in Figure 2. The productivity of DNA sequencing and synthesis is measured in per nucleotide per person per day.



(a) (b)

Figure 1.2 : The plot in 2(a) shows how cost for sequencing and synthesis have exponentially declined over years. The plot in 2(b) shows comparison of DNA with traditional storage.

1.2 Background on DNA

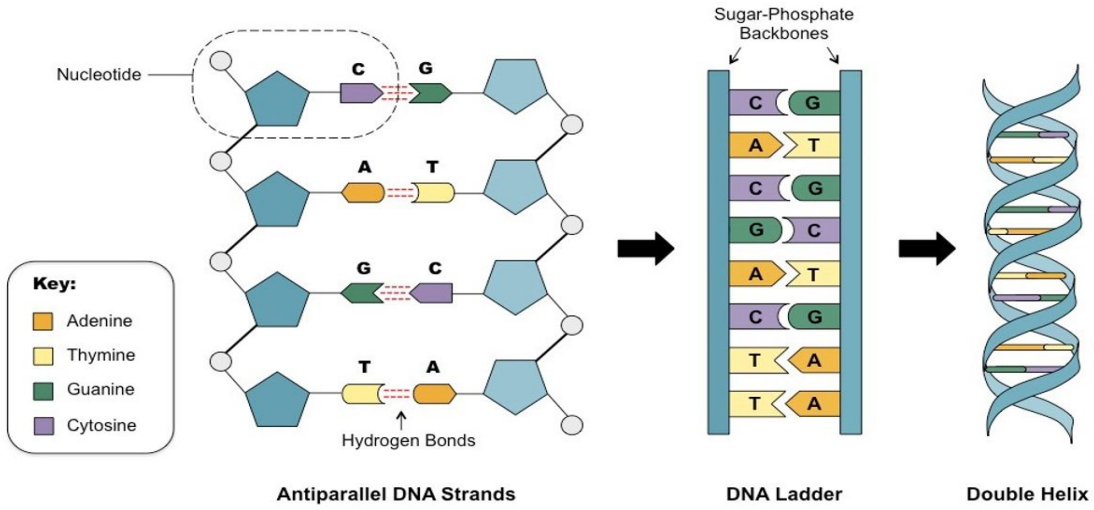


Figure 1.3: Structure of DNA

1.2.1 Structure of DNA

The structure of DNA that exists by itself in nature consists of four types of nucleotides: adenine (A), cytosine (C), guanine (G), and thymine (T). A DNA strand, or oligonucleotide, is a linear sequence of these nucleotides. The two ends of a DNA strand, referred to as the 5 ′ and 3 ′ ends, are chemically different. DNA sequences are conventionally represented starting with the 5 ′ nucleotide end. The interactions between different strands are predictable based on sequence. Two single strands can bind to each other and form a double helix if they are complementary: A

in one strand aligns with T in the other, and likewise for C and G. The two strands in a double helix have opposite directionality (5 ′ end binds to the other strand's 3 ′ end), and thus the two sequences are the "reverse complement" of each other. Two strands do not need to be fully complementary to bind to one another. Such partial complementarity is useful for applications in DNA nanotechnology and other fields, but can also result in undesired "crosstalk" between sequences in complex reaction mixtures containing many sequences.

1.2.2 Polymerase chain reaction (PCR)

Selective DNA amplification with polymerase chain reaction (PCR) is a method for exponentially amplifying the concentration of selected sequences of DNA within a pool. A PCR reaction requires four main components: the template, sequencing primers, a thermostable polymerase and individual nucleotides that get incorporated into the DNA strand being amplified. The template is a single- or double-stranded molecule containing a subsequence that will be amplified. The DNA sequencing primers are short synthetic strands that define the beginning and end of the region to be amplified. The polymerase is an enzyme that creates double-stranded DNA from a single-stranded template by "filling in" individual complementary nucleotides one by one, starting from a primer bound to that template. PCR happens in "cycles", each of which doubles the number of templates in a solution. The process can be repeated until the desired number of copies is created.

1.2.3 DNA synthesis

Arbitrary single-strand DNA sequences can be synthesized chemically, nucleotide by nucleotide. The coupling efficiency of a synthesis process is the probability that a nucleotide binds to an existing partial strand at each step of the process. Although the coupling efficiency for each step can be higher than 99%, this small error still results in an exponential decrease of product yield with increasing length and limits the size of oligonucleotides that can be efficiently synthesized to about 200 nucleotides. In practice, synthesis of a given sequence uses a large number of parallel start sites and results in many truncated byproducts (the dominant error in DNA synthesis), in addition to many copies of the full length target sequence. Thus, despite errors in synthesizing any specific strand, a given synthesis batch will usually produce many perfect strands. Moreover, modern array synthesis techniques can synthesize complex pools of nearly 105 different oligonucleotides in parallel.

1.2.4 DNA sequencing

There are several high-throughput sequencing techniques, but the most popular methods (such as that used by Illumina) use DNA polymerase enzymes and are commonly referred to as "sequencing by synthesis". The strand of interest serves as a template for the polymerase, which creates a complement of the strand. Importantly, fluorescent nucleotides are used during this

synthesis process. Since each type of fluorescent nucleotide emits a different color, it is possible to read out the complement sequence optically. Sequencing is error-prone, but as with synthesis, in aggregate, sequencing typically produces enough precise reads of each strand

1.2.5 Sequencing and synthesis improvement projections.

Today, neither the performance nor the cost of DNA synthesis and sequencing is viable for data storage purposes. However, they have historically seen exponential improvements. Their cost reductions and throughput improvements have been compared to Moore's Law in Carlson's Curves, as shown in Figure 1. It shows that sequencing productivity has been growing faster than Moore's Law. Important biotechnology applications such as genomics and the development of smart drugs are expected to continue driving these improvements, eventually making data storage a viable application.

1.3 Problem Statement

The work on DNA storage is very recent and has not been explored in detail yet. However it is quite inevitable to stop DNA storage from being the storage system of the future, given that the cost will someday be significantly improved.

So, we propose a way to deliver the .DNA files to the industry so that it could be used for commercial purposes by the biologists who will synthesize the DNA according to our encoded file. For this purpose, we create an online platform for a user to encode any file format and download the DNA encoded file in .DNA format. The user can also decode the .DNA file to its original content.

We have also extended our approach to optimize the storage of image files. It will be very expensive to synthesize DNA to store large image files. So, in order to improve upon it, we have proposed a scheme to compress image files by extracting only the important features of an image using Convolution Neural Network, and regenerating the image using Super-Resolution.

2. Literature Survey

2.1 Neural Network Architecture

The architecture of an artificial neural network defines how its several neurons are arranged, or placed, in relation to each other. These arrangements are structured essentially by directing the synaptic connections of the neurons.

The topology of a given neural network, within a particular architecture, can be defined as the different structural compositions it can assume. In other words, it is possible to have two topologies belonging to the same architecture, where the first topology is composed of 10 neurons, and the second is composed of 20 neurons. Moreover, one can consist of neurons with logistic activation function, while the other one can consist of neurons with the hyperbolic tangent as the activation function.

In general, an artificial neural network can be divided into three parts, named layers, which are known as:

A. **Input layer**
   This layer is responsible for receiving information (data), signals, features, or measurements from the external environment. These inputs (samples or patterns) are usually normalized within the limit values produced by activation functions. This normalization results in better numerical precision for the mathematical operations performed by the network.

B. **Hidden, intermediate, or invisible layers**
   These layers are composed of neurons which are responsible for extracting patterns associated with the process or system being analyzed. These layers perform most of the internal processing from a network.

C. **Output layer**
   This layer is also composed of neurons, and thus is responsible for producing and presenting the final network outputs, which result from the processing performed by the neurons in the previous layers. The main architectures of artificial neural networks, considering the neuron disposition, as well as how they are interconnected and how its layers are composed, can be divided as follows:
   a. single-layer feedforward network,
   b. multilayer feedforward networks,

c. recurrent networks and
d. mesh networks.

2.1.1 Single-Layer Feedforward Architecture

This artificial neural network has just one input layer and a single neural layer, which is also the output layer. Figure 2.1 illustrates a simple-layer feedforward network composed of n inputs and m outputs.

The information always flows in a single direction (thus, unidirectional), which is from the input layer to the output layer. It is possible to see that in networks belonging to this architecture, the number of network outputs will always coincide with its amount of neurons. These networks are usually employed in pattern classification and linear filtering problems.

Among the main network types belonging to feedforward architecture are the Perceptron and the ADALINE, whose learning algorithms used in their training processes are based respectively on Hebb's rule and Delta rule, as it will be discussed in the next chapters.

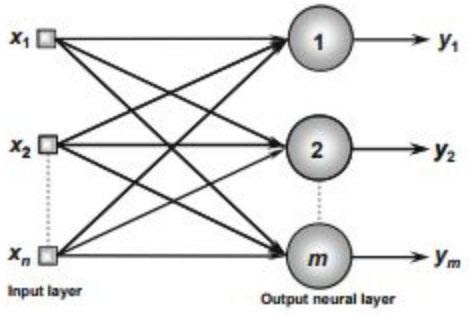


Fig 2.1: Single layer feed forward network

2.1.2 Multiple-Layer Feedforward Architectures

Differently from networks belonging to the previous architecture, feedforward networks with multiple layers are composed of one or more hidden neural layers. They are employed in the solution of diverse problems, like those related to function approximation, pattern classification, system identification, process control, optimization, robotics, and so on.

Figure shows a feedforward network with multiple layers composed of one input layer with n sample signals, two hidden neural layers consisting of n1 and n2 neurons respectively, and, finally, one output neural layer composed of m neurons representing the respective output values of the problem being analyzed.

Among the main networks using multiple-layer feedforward architectures are the Multilayer Perceptron (MLP) and the Radial Basis Function (RBF), whose learning algorithms used in their training processes are respectively based on the generalized delta rule and the competitive/delta rule. These concepts will be addressed in the next chapters.

From Fig., it is possible to understand that the amount of neurons composing the first hidden layer is usually different from the number of signals composing the input layer of the network. In fact, the number of hidden layers and their respective amount of neurons depend on the nature and complexity of the problem being mapped by the network, as well as the quantity and quality of the available data about the problem. Nonetheless, likewise for simple-layer feedforward networks, the amount of output signals will always coincide with the number of neurons from that respective layer.

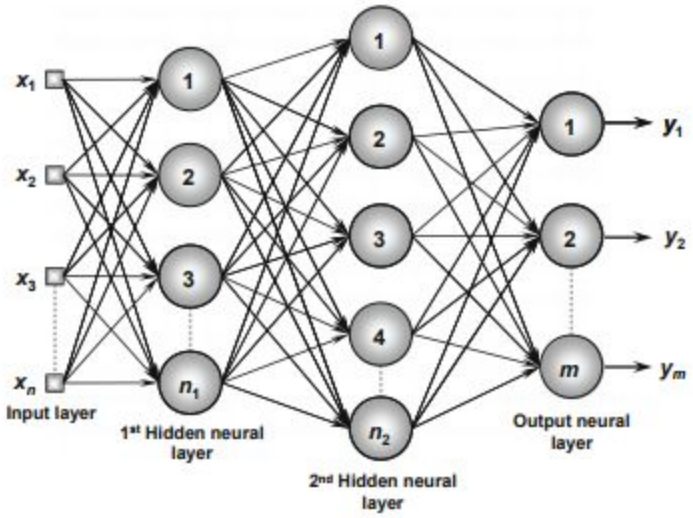


Fig 2.2: Multiple layer feedforward network

2.1.3 Recurrent or Feedback Architecture

In these networks, the outputs of the neurons are used as feedback inputs for other neurons. The feedback feature qualifies these networks for dynamic information processing, meaning that they can be employed on time-variant systems, such as time series prediction, system identification and optimization, process control, and so forth.

Among the main feedback networks are the Hopfield and the Perceptron with feedback between neurons from distinct layers, whose learning algorithms used in their training processes are respectively based on energy function minimization and generalized delta rule, as will be investigated in the next chapters.

Figure illustrates an example of a Perceptron network with feedback, where one of its output signals is fed back to the middle layer.

Thus, using the feedback process, the networks with this architecture produce current outputs also taking into consideration the previous output values.

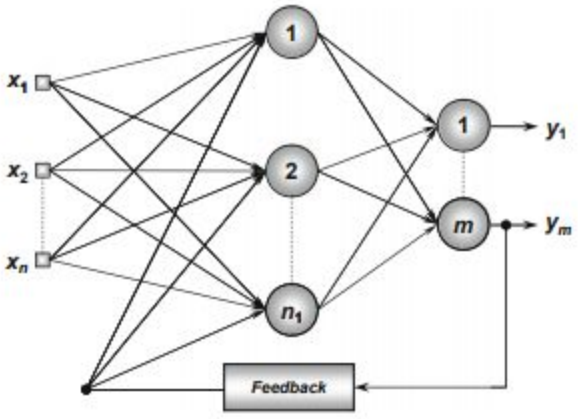


Fig 2.3: Recurrent neural network

2.2 Convolutional Neural Networks (CNNs)

Figure 2.4: CNNs and computer vision

The goal of a CNN is to learn higher-order features in the data via convolutions. They are well suited to object recognition with images and consistently top image classification competitions. They can identify faces, individuals, street signs, platypuses, and many other aspects of visual data. CNNs overlap with text analysis via optical character recognition, but they are also useful when analyzing words as discrete textual units. They're also good at analyzing sound.

The efficacy of CNNs in image recognition is one of the main reasons why the world recognizes the power of deep learning. As Figure illustrates, CNNs are good at building position and (somewhat) rotation invariant features from raw image data.

CNNs are powering major advances in machine vision, which has obvious applications for self-driving cars, robotics, drones, and treatments for the visually impaired.

2.2.1 Biological Inspiration

The biological inspiration for CNN is the visual cortex in animals. The cells in the visual cortex are sensitive to small subregions of the input. We call this the visual field (or receptive field). These smaller subregions are tiled together to cover the entire visual field. The cells are well suited to exploit the strong spatially local correlation found in the types of images our brains process, and act as local filters over the input space. There are two classes of cells in this region of the brain. The simple cells activate when they detect edge-like patterns, and the more complex cells activate when they have a larger receptive field and are invariant to the position of the pattern.

2.2.2 Intuition

Feed-forward multilayer neural networks take input as a single one-dimensional vector and transform the data with one or more hidden layers (fully connected). The network then gives a result from the output layer. The issue we run into with traditional multilayer neural networks and image data is that these networks don't scale well with image data as input. An example would be modeling the CIFAR-10 dataset (see the upcoming sidebar). The images to train on are only 32 pixels wide by 32 pixels in height with 3 channels of RGB information. This creates 3,072 weights per neuron in the first hidden layer, however, and we'll probably want more than one neuron in that hidden layer. In many cases, we'll want multiple hidden layers in our multilayer neural network, which will multiply those weights, as well.

The CIFAR-10 dataset is a well-known image classification benchmark dataset compiled by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The dataset has 60,000 color images with 10 different classes comprising 6,000 images per class. The images are 32 × 32 pixels each. There

are 50,000 training images and 10,000 test images. Figure 4-8 shows the image classes in the dataset.

A normal image could easily be 300 pixels in width by 300 pixels in height with 3 channels of RGB information. This would create 270,000 connection weights per hidden neuron. This shows how quickly a fully connected multilayer network creates a massive number of connections when modeling image data. The structure of image data allows us to change the architecture of a neural network in a way that we can take advantage of this structure. With CNNs, we can arrange the neurons in a three-dimensional structure for which we have the following:

- Width
- Height
- Depth

These attributes of the input match up to an image structure for which we have:

- Image width in pixels
- Image height in pixels
- RGB channels as the depth

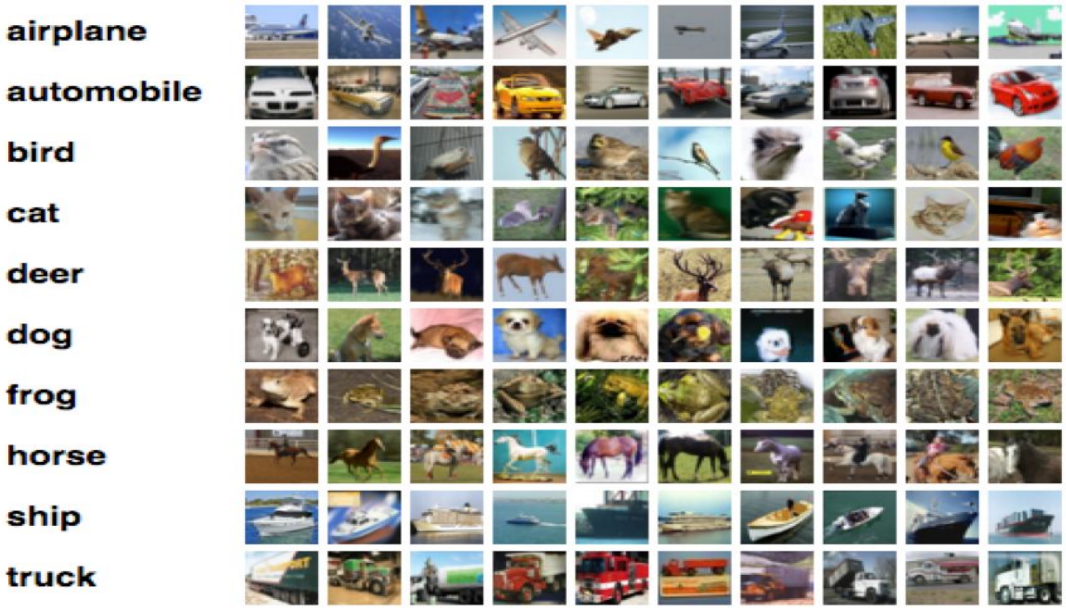


Figure 2.5: The CI*FAR-10 dataset

We can consider this structure to be a three-dimensional volume of neurons. A significant aspect to how CNNs evolved from previous feed-forward variants is how they achieved computational

efficiency with new layer types. We'll cover this arrangement in more depth momentarily. Let's now take a look at the high-level general architecture of CNNs.

2.2.3 CNN Architecture Overview

CNNs transform the input data from the input layer through all connected layers into a set of class scores given by the output layer. There are many variations of the CNN architecture, but they are based on the pattern of layers, as demonstrated in Figure.

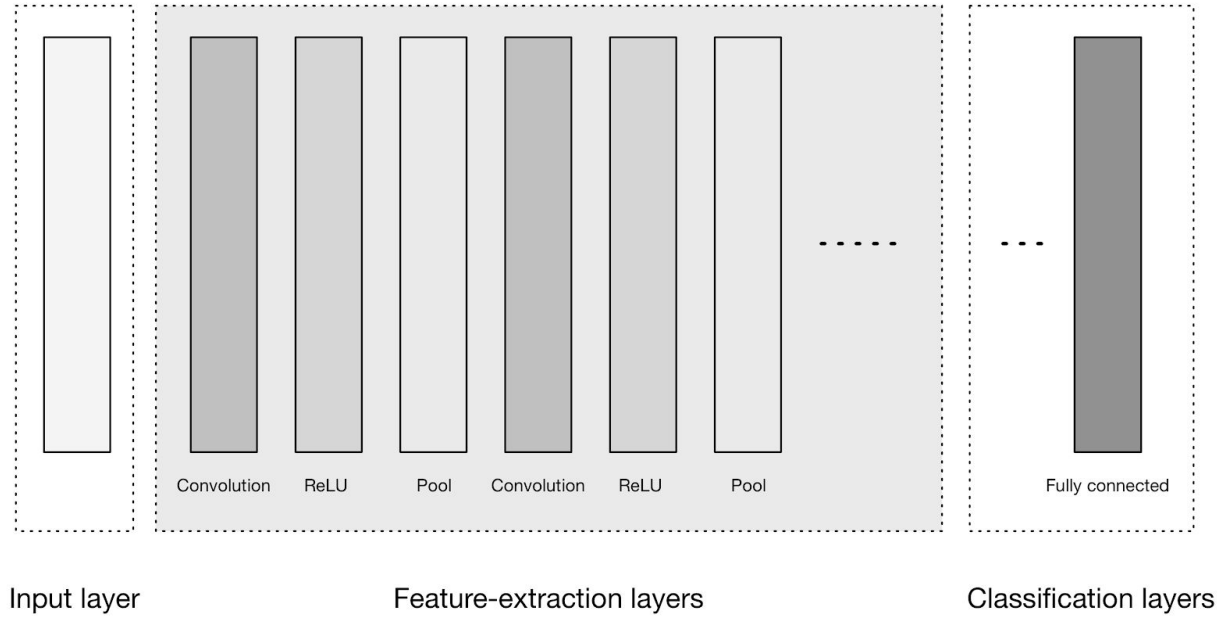


Figure 2.6: High-level general CNN architecture

Figure depicts three major groups:

1. Input layer
2. Feature-extraction (learning) layers
3. Classification layers

The input layer accepts three-dimensional input generally in the form spatially of the size (width × height) of the image and has a depth representing the color channels (generally three for RGB color channels).

The feature-extraction layers have a general repeating pattern of the sequence:

1. Convolution layer
   We express the Rectified Linear Unit (ReLU) activation function as a layer in the diagram here to match up to other literature.

2. Pooling layer

These layers find a number of features in the images and progressively construct higher-order features. This corresponds directly to the ongoing theme in deep learning by which features are automatically learned as opposed to traditionally hand engineered.

Finally we have the classification layers in which we have one or more fully connected layers to take the higher-order features and produce class probabilities or scores. These layers are fully connected to all of the neurons in the previous layer, as their name implies. The output of these layers produces typically a two-dimensional output of the dimensions [b × N], where b is the number of examples in the mini-batch and N is the number of classes we're interested in scoring.

Neuron spatial arrangements

Recall how in traditional multilayer neural networks, the layers are fully connected and every neuron in a layer is connected to every neuron in the next layer. The neurons in the layers of a CNN are arranged in three dimensions to match the input volumes. Here, depth means the third dimension of the activation volume, not the number of layers, as in a multilayer neural network.
Evolution of the connections between layers

Another change is how we connect layers in a convolutional architecture. Neurons in a layer are connected to only a small region of neurons in the layer before it. CNNs retain a layer-oriented architecture, as in traditional multilayer networks, but have different types of layers. Each layer transforms the 3D input volume from the previous layer into a 3D output volume of neuron activations with some differentiable function that might or might not have parameters, as demonstrated in Figure.

Input Layers

Input layers are where we load and store the raw input data of the image for processing in the network. This input data specifies the width, height, and number of channels. Typically, the number of channels is three, for the RGB values for each pixel.

Convolutional Layers

Convolutional layers are considered the core building blocks of CNN architectures. As Figure illustrates, convolutional layers transform the input data by using a patch of locally connecting neurons from the previous layer. The layer will compute a dot product between the region of the neurons in the input layer and the weights to which they are locally connected in the output layer. The resulting output generally has the same spatial dimensions (or smaller spatial dimensions) but sometimes increases the number of elements in the third dimension of the output (depth dimension). Let's take a closer look at a key concept in these layers, called a convolution.

Convolution

A convolution is defined as a mathematical operation describing a rule for how to merge two sets of information. It is important in both physics and mathematics and defines a bridge between the space/time domain and the frequency domain through the use of Fourier transforms. It takes input, applies a convolution kernel, and gives us a feature map as output.

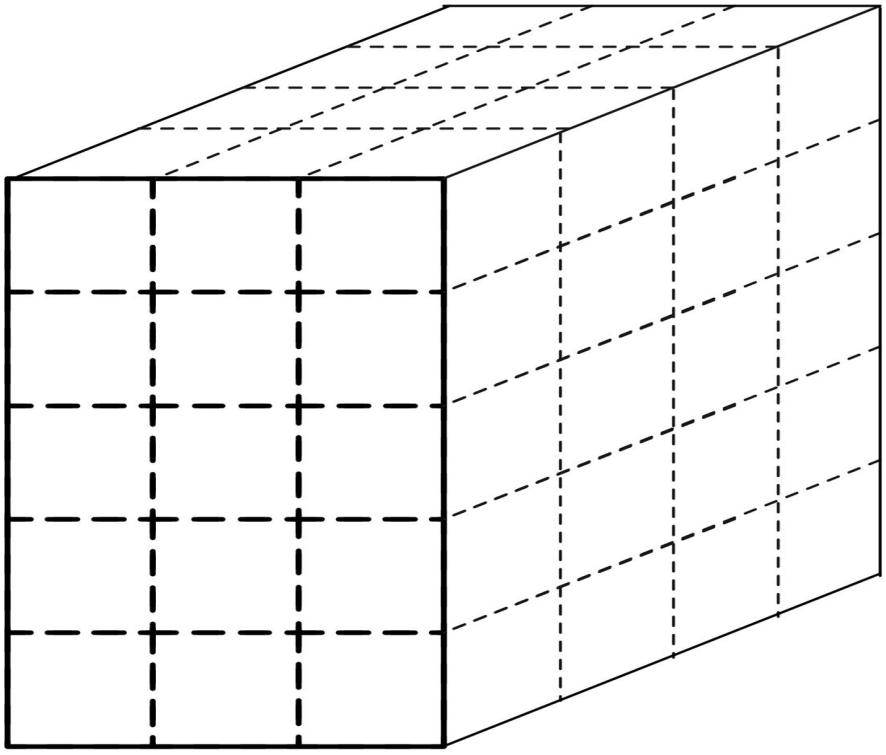


Figure 2.7: Input layer 3D volume

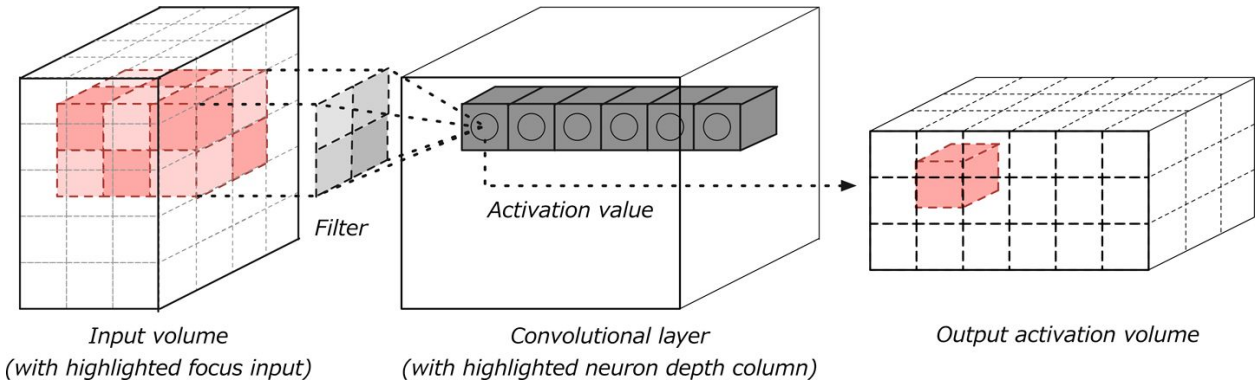


Figure 2.8: Convolution layer with input and output volumes

The convolution operation, shown in Figure, is known as the feature detector of a CNN. The input to a convolution can be raw data or a feature map output from another convolution. It is often interpreted as a filter in which the kernel filters input data for certain kinds of information; for example, an edge kernel lets pass through only information from the edge of an image.

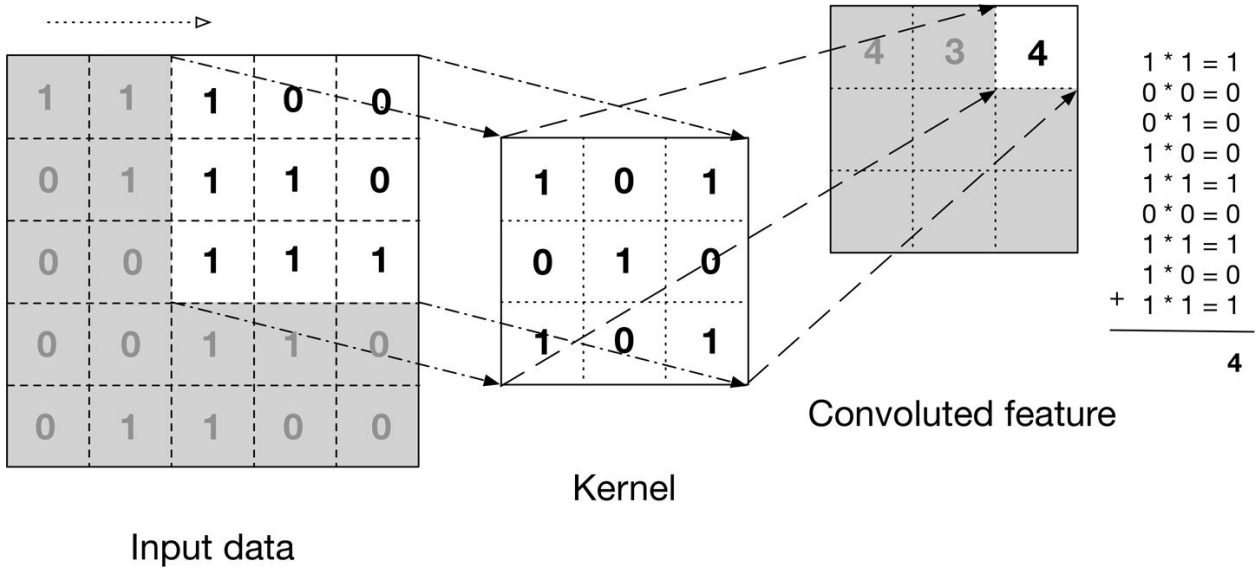


Figure 2.9: The convolution operation

The figure illustrates how the kernel is slid across the input data to produce the convoluted feature (output) data. At each step, the kernel is multiplied by the input data values within its bounds, creating a single entry in the output feature map. In practice the output is large if the feature we're looking for is detected in the input.

We commonly refer to the sets of weights in a convolutional layer as a filter (or kernel). This filter is convolved with the input and the result is a feature map (or activation map). Convolutional layers perform transformations on the input data volume that are a function of the activations in the input volume and the parameters (weights and biases of the neurons). The activation map for each filter is stacked together along the depth dimension to construct the 3D output volume.

Convolutional layers have parameters for the layer and additional hyperparameters. Gradient descent is used to train the parameters in this layer such that the class scores are consistent with the labels in the training set. Following are the major components of convolutional layers:

- Filters
- Activation maps

- Parameter sharing

Filters

The parameters for a convolutional layer configure the layer's set of filters. Filters are a function that has a width and height smaller than the width and height of the input volume.

Filters (e.g., convolutions) are applied across the width and height of the input volume in a sliding window manner, as demonstrated in Figure 4-12. Filters are also applied for every depth of the input volume. We compute the output of the filter by producing the dot product of the filter and the input region.

The architecture of CNNs is set up such that the learned filters produce the strongest activation to spatially local input patterns. This means that filters are learned that will activate on patterns (or features) only when the patterns occur in the training data in their respective field. As we move farther along in layers in a CNN, we encounter filters that can recognize nonlinear combinations of features and are increasingly global in how they can detect patterns. High-performing convolutional architectures (which we'll see later in this section) have shown network depth to be an important factor in CNNs.

Activation maps

Recall from Chapter 1 that an activation is a numerical result if a neuron decided to let information pass through. This is a function of the inputs to the activation function, the weights on the connections (for the inputs, and the type of activation function itself). When we say the filter "activates," we mean that the filter lets information pass through it from the input volume into the output volume.

We slide each filter across the spatial dimensions (width, height) of the input volume during the forward pass of information through the CNN. This produces a two-dimensional output called an activation map for that specific filter. Figure depicts how this activation map relates to our previously introduced concept of a convoluted feature.

The activation map on the right in Figure is rendered differently to illustrate how convolutional activation maps are commonly rendered in the literature.

To compute the activation map, we slide the filter across the input volume depth slice. We calculate the dot product between the entries in the filter and the input volume. The filter represents the weights that are being multiplied by the moving window (subset) of input activations. Networks learn filters that activate when they see certain types of features in the input data in a specific spatial position

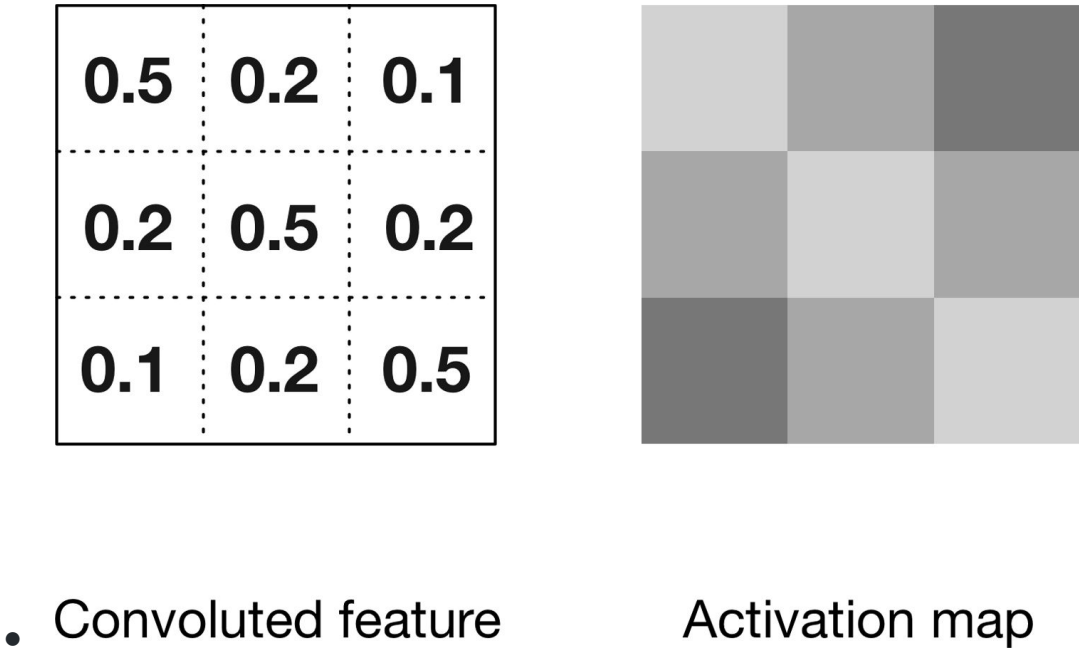


Figure 2.10: Convolutions and activation maps

We create the three-dimensional output volume for the convolution layer by stacking these activation maps along the depth dimension in the output, as shown in Figure. The output volume will have entries that we consider the output of a neuron that look at only a small window of the input volume.

In some cases, this output will be the result of parameters shared with neurons in the same activation map. Each neuron generating the output volume is connected to only a local region of the input volume, as depicted in Figure.

We control local connectivity of this process with the hyperparameter called the receptive field, which controls how much of the width and height of the input volume against which our filter maps.

Filters define a smaller bounded region to generate activation maps from the input volumes. They are connected to only a subset of the input volume through the dynamics of local connectivity described earlier. This allows us to still have quality feature extraction while reducing the number of parameters per layer we need to train. Convolutional layers reduce the parameter count further by using a technique called parameter sharing.

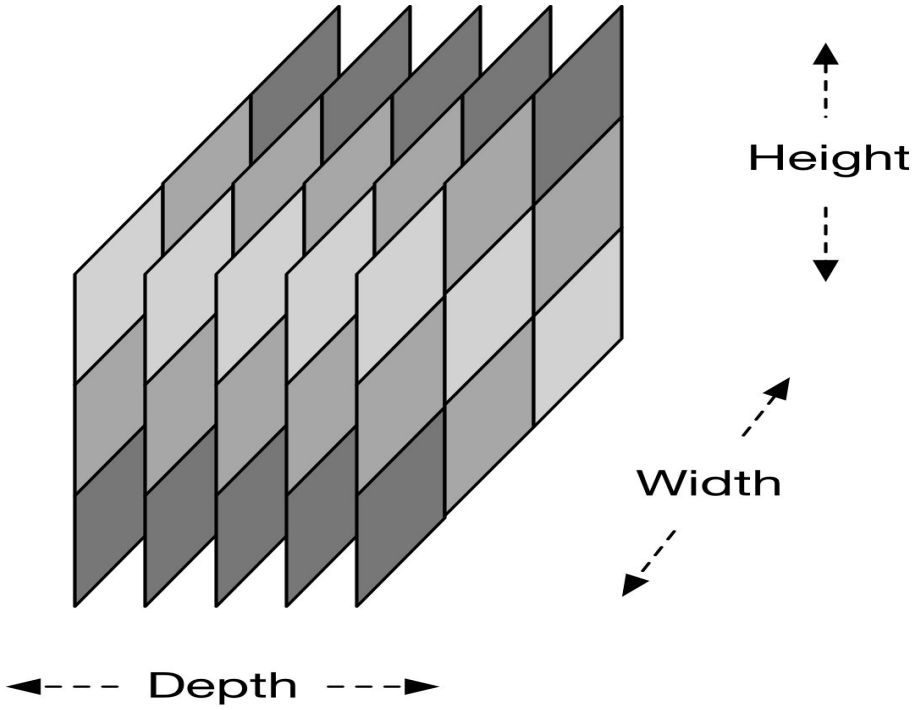


Figure 2.11: Activation volume output of convolutional layer

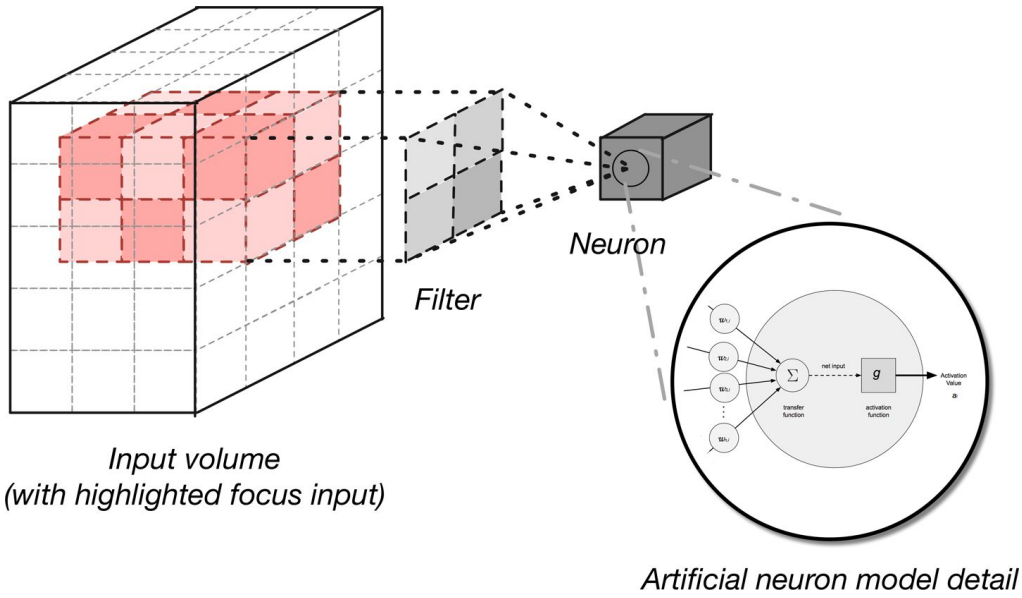


Figure 2.12: Generating an activation output volume

Parameter sharing

CNNs use a parameter-sharing scheme to control the total parameter count. This helps training time because we'll use fewer resources to learn the training dataset. To implement parameter

sharing in CNNs, we first denote a single two-dimensional slice of depth as a "depth slice." We then constrain the neurons in each depth slice to use the same weights and bias. This gives us significantly fewer parameters (or weights) for a given convolutional layer.

We are not able to take advantage of parameter sharing when the input images we're training on have a specific centered structure. We see this effect in faces when we always expect a specific feature to appear in a specific place (for centered faces).

2.3 Client - Server Architecture

The purpose of the following section is to get a first feel, what Client Server means and which definitions are mentioned in the literature and how the basic Client-Server architecture look like.

2.3.1 Definitions

1. "A Client-Server network is a distributed network which consists of one higher performance system, the server, and several mostly lower performance systems, the clients. The server is the central registering unit as well as the only provider of content and service. A client only requests content or the execution of services, without sharing any of its own resources."

2. "A Client-Server architecture is a network architecture in which each computer or process on the network is either a client or a server. Servers are powerful computers or processes dedicated to managing disk drives (file servers), printers (print servers), or network traffic (network servers). Clients are PCs or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power."

2.3.2 Architecture

The most commonly used paradigm in constructing distributed systems is the Client-Server model. In this scheme client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. Servers wait for requests to arrive from clients and then respond to them. Ideally, a server provides a standardized transparent interface to clients so that clients need not be aware of the specifics of the system (i.e., the hardware and software) that is providing the service. Clients are often situated at workstations or on personal computers, while servers are located elsewhere on the network, usually on more powerful machines. This computing model is especially effective when clients and the server each have distinct tasks that they routinely perform. In hospital data processing, for example, a client computer can be running an application program for entering

patient information while the server computer is running another program that manages the database in which the information is permanently stored. Many clients can access the server's information simultaneously, and, at the same time, a client computer can perform other tasks, such as sending e-mail. Because both client and server computers are considered intelligent devices, the client-server model is completely different from the old "mainframe" model, in which a centralized mainframe computer performed all the tasks for its associated "dumb" terminals.

The client and server require a known set of conventions before the can communicate. This set of conventions contains a protocol, which must be implemented at both ends of a connection. Examples of protocols are the TELNET protocol used in the Internet for remote terminal emulation, the Internet file transfer protocol, FTP and http.

2.3.3 Server

As a provider of services the server must compute requests and has to return the results with an appropriate protocol. A server as a provider of services can be running on the same device as the client is running on, or on a different device, which is reachable over the network. The decision to outsource a service from an application in the form of a server can have different reasons.

- **Performance**
  In certain circumstances the clients are inefficient devices, which have interfaces to high performance demanding applications. In this case the computation is done on a high performance server. Today this approach is less used, but has still its area of application, e.g., virtual reality computations for film scenes.

- **Central data management**
  This aspect of the Client-Server model does have the most impact today. Data is stored on a server, which can be used or manipulated from different clients. Typical examples of services provided by a server are:
  - **File server**
    One server provides multiple clients with a file system. Tasks of this server include access control and transaction control (only one client may access a file with write permissions at a time).
  - **Web server**
    The Web server provides multiple clients (Web browser on different devices) with information. The information can be static on a Web server or dynamic, generated by different service applications.

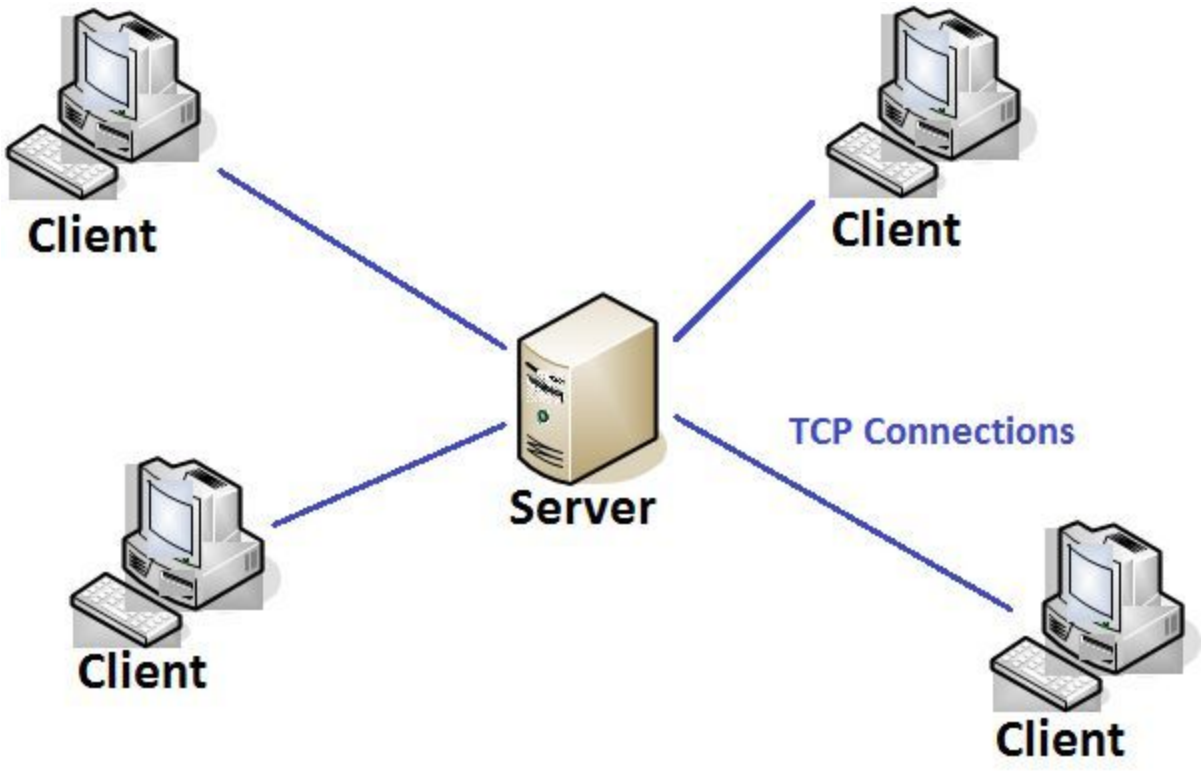


Fig 2.13: Client-Server interaction

2.3.4 Client

A client is typically a device or a process which uses the service of one or more servers. Since clients are often the interface between server information and people, clients are designed for information input and visualization of information. Although clients had only few resources and functionality in the past, today most clients are PCs with more performance regarding resources and functionality. Early clients had only the task to display the application, that was running on the server and to forward inputs of the user to the server. All computations are done on the server. In this case one speaks of a thin client. A thin client has limited local resources in terms of hardware and software. It functionally requires processing time, applications and services to be provided from a centralized server. Network computers are examples of the development of thin clients. A thick client is functionally rich in terms of hardware and software. hick clients are capable of storing and executing their own applications as well as network centric ones. Thick client typically refers to a personal computer. Client-Server architectures can be classified into flat and hierarchical. If the Client-Server model is flat, all clients communicate only with a single server. If the Client-Server model is hierarchical the servers of one level are acting as clients to higher level servers. A pretty good example is a request of a certain web page. The user enters a URL into the web browser (client). The client establishes a connection to his nearest name server to ask for the address. If that server does not know the name, it delegates

the 16 query to the authority for that namespace. That query, in turn, may be delegated to a higher authority, all the way up to the root name servers for the Internet as a whole. Name servers operate both as clients and as servers.

Advantages and Disadvantages

The following advantages and disadvantages of the Client-Server architecture are extracted from several technical reports and books concerning Client-Server architecture. The following points should not be regarded as a complete listing, but rather as key advantages and disadvantages.

Advantages

- Data management is much easier because the files are in one location. This allows fast backups and efficient error management. There are multiple levels of permissions, which can prevent users from doing damage to files. The server hardware is designed to serve requests from clients quickly. All the data are processed on the server, and only the results are returned to the client. This reduces the amount of network traffic between the server and the client machine, improving network performance.

- Thin client architectures allow a quick replacement of defect clients, because all data and applications are on the server.

Disadvantages

- Client-Server Systems are very expensive and need a lot of maintenance. The server constitutes a single point of failure. If failures on the server occur, it is possible that the system suffers heavy delay or completely breaks down, which can potentially block hundreds of clients from working with their data or their applications. Within companies high costs could accumulate due to server downtime.

2.4 NodeJS

Node.js is an open-source, cross-platform runtime environment for developing server-side web applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, Linux, FreeBSD, NonStop, IBM AIX, IBM System z and IBM i. Its work is hosted and supported by the Node.js Foundation, a collaborative project at Linux Foundation.

Node.js provides an event-driven architecture and a non-blocking I/O API designed to optimize an application's throughput and scalability for real-time web applications. It uses Google V8

JavaScript engine to execute code, and a large percentage of the basic modules are written in JavaScript. Node.js contains a built-in library to allow applications to act as a stand-alone web server.

Node.js is used by IBM, Microsoft, Yahoo!, Walmart, Groupon, SAP, LinkedIn, Rakuten, PayPal, Voxer and GoDaddy.

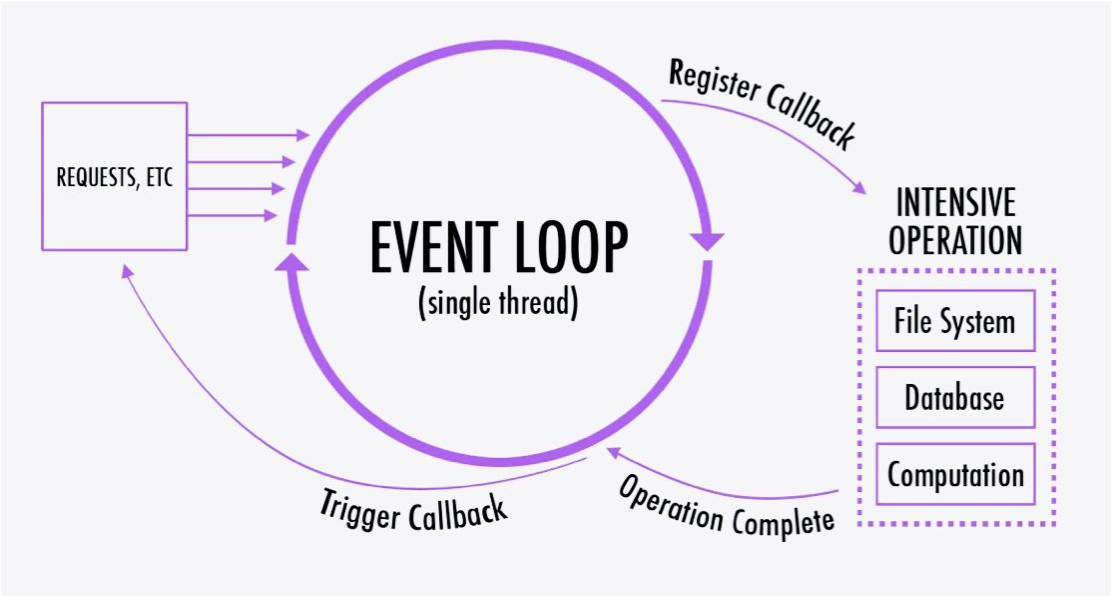


Fig 2.14: NodeJs working

The event loop and inherent CommonJS module system ensures high modularity and performance with the virtue of non blocking asynchronous calls to intensive operations. This not only benefits the server of the application but also our generator code. NodeJS is a perfect choice for our component based development engineering process.

Node is similar in design to, and influenced by, systems like Ruby's Event Machine or Python's Twisted. Node takes the event model a bit further. It presents an event loop as a runtime construct instead of as a library. In other systems there is always a blocking call to start the event-loop. Typically behavior is defined through callbacks at the beginning of a script and at the end starts a server through a blocking call like EventMachine::run(). In Node there is no such start-the-event-loop call. Node simply enters the event loop after executing the input script. Node exits the event loop when there are no more callbacks to perform. This behavior is like browser JavaScript — the event loop is hidden from the user.

HTTP is a first class citizen in Node, designed with streaming and low latency in mind. This makes Node well suited for the foundation of a web library or framework.

Just because Node is designed without threads, doesn't mean you cannot take advantage of multiple cores in your environment. Child processes can be spawned by using our child_process.fork() API, and are designed to be easy to communicate with. Built upon that same interface is the cluster module, which allows you to share sockets between processes to enable load balancing over your cores.

2.5 Related Works

Encoding data in DNA has a long line of research in the biology community. Early examples encoded and recovered very short messages: Clelland et al. recovered a 23 character message in 1999, and Leier et al. recover three 9-bit numbers in 2000. The first significant scaling improvements were made by Gibson et al. in 2010, successfully recovering 1280 characters encoded in a bacterial genome as "watermarks" – but note this approach is *in vivo* (inside an organism), whereas ours is *in vitro* (outside), so the technology is very different and inapplicable to large-scale storage. More scaling improvements were made by Church et al. in 2012, who recovered a 643kB message, and Goldman et al. recovered a 739kB message also in 2012. However, both these results required manual intervention: Church et al. had to manually correct ten bits of error, and Goldman et al. lost two sequences of 25 nucleotides.Storing messages in DNA was first demonstrated in 1988 and the largest project to date encoded 7920 bits . The small scale of previous work stems from the difficulty of writing and reading long perfect DNA sequences, and has limited broader applications (table S1). Here, we develop a strategy to encode arbitrary digital information using a novel encoding scheme that utilizes next-generation DNA synthesis and sequencing technologies (fig. S1). We converted an html-coded draft of a book that included 53,426 words, 11 JPG images and 1 JavaScript program into a 5.27 megabit bitstream. We then encoded these bits onto 54,898 159nt oligonucleotides (oligos) each encoding a 96-bit data block (96nt), a 19-bit address specifying the location of the data block in the bit stream (19nt), and flanking 22nt common sequences for amplification and sequencing. The oligo library was synthesized by ink-jet printed, high-fidelity DNA microchips. To read the encoded book, we amplified the library by limited-cycle PCR and then sequenced on a single lane of an Illumina HiSeq. We joined overlapping paired-end 100 nt reads to reduce the effect of sequencing error. Then using only reads that gave the expected 115-nt length and perfect barcode sequences, we generated consensus at each base of each data block at an average of ~3000-fold coverage. All data blocks were recovered with a total of 10 bit errors out of 5.27 million (table S2), which were predominantly located within homo-polymer runs at the end of the oligo where we only had single sequence coverage.

Most recently, Grass et al. recovered an 83kB message without error. Their design uses a Reed-Solomon code, striping the entire dataset across 5000 DNA strands. While this design leads to excellent redundancy, it defeats the desire for random access.

Yazdi et al. developed a method for rewritable random-access DNA-based storage. Its encoding is dictionary-based and focuses on storage of text, while our approach accommodates arbitrary binary data. We do not support rewritability, which adds substantial complexity, because write-once is appropriate for archival items (e.g., photos) that are unlikely to change. If necessary, one can use a log-based approach for rewriting objects, since the density of DNA exceeds the overhead of logging.The key component of their implementation have is a new collection of coding schemes and the adaptation of random-access enabling codes from classical storage systems. In particular, we encoded information within blocks with unique addresses that are prohibited to appear anywhere else in the encoded information, thereby removing any undesirable cross-hybridization problems during the process of selection and amplification. They also performed four access and rewriting experiments without readout errors, as confirmed by post-selection and rewriting Sanger sequencing. The current drawback of our scheme is high cost, as synthesizing long DNA blocks is expensive. Cost considerations also limited the scope of our experiments and the size of the prototype, as they had also aimed to stay within a budget comparable to that used for other existing architectures. Nevertheless, the benefits of random access and other unique features of the proposed system compensate for this high cost, which we predict will decrease rapidly in the very near future. Random access in DNA based archival systems would be an important feature addition which would make larger storages even more feasible in future. The problem of sequencing the whole encoded sequence serves as a big limitation for the system to become commercial and large scale use.

The use of DNA as a computing substrate also has a long history: in 1994, Adleman proposed composing Hamiltonian paths with DNA. Researchers have proposed the use of DNA for Boolean circuits, neural networks, and chemical reaction networks. DNA computing has also begun making inroads in the architecture community: Muscat et al. explore the architectural challenges of DNA strand-displacement circuits, and Talawar examines the design of a DNA-based crossbar interconnection network..

Luis Ceze el. al in their paper proposed and envision DNA storage as the very last level of a deep storage hierarchy, providing very dense and durable archival storage with access times of many hours to days. DNA synthesis and sequencing can be made arbitrarily parallel, making the necessary read and write bandwidths attainable. They also propose a new encoding technique for the encoding of data into DNA format using a XOR encoding technique as compared to Goldman's encoding technique.While the Goldman encoding provides high reliability, it also incurs significant overhead: each block in the input string is repeated four times.

3. Software Requirements

The project requires certain hardware and software requirements to run properly.

These requirements are listed as follows :

- Python 2.7 must be installed.
- Tensorflow, a python library, must be correctly set up and working.
- Any stable version on OSX, Ubuntu, Linux and Windows would work well with this project.
- Minimum hardware requirements are as follows :
    - Processor : 2.0 GHz
    - Memory : 2GB RAM
    - Hard Drive Space : 500 MB
    - Graphics : 512 MB of VRAM
    - I/O Devices : Mouse, keyboard, monitor

4. Proposed Model

Most of the complex structures found in the world are enormously redundant, and we can use this redundancy to simplify their description. But to use it, to achieve the simplification, we must find the right representation.- Herbert A. Simon

4.1 Goldman DNA Storage Encoding

We have used an encoding-decoding scheme proposed by Goldman to translate the zeroes and ones that make up digital files into As, Cs, Gs and Ts — the letters that correspond to the basic components of DNA. It might not seem like such a hard thing to do, but we had to use some other rules to make sure the experiment would work, such as requiring that the new, alphabetic code would not have any repeats. Repeating letters in the code could confuse the machines that write and read DNA. We also had to work out how to break each message into many pieces (since humans can only reliably create DNA fragments about 200 letters long), sort them out and put them back together again when they are read. We had to do all this in a manner that could recover the information perfectly, even when there were inevitable writing and reading errors.

This coded information can be fed into DNA synthesis machines, which transforms it into the physical material in much the same way an inkjet printer lays down ink on paper. What you get in the end is an almost imperceptible smidgen of dust, which itself contains thousands of DNA copies of the encoded files. Because DNA is so robust, the material will last for many thousands of years if it is kept safe, dry and cool. DNA sequencing machines can be used to read the files back.

4.1.1 Encoding

1. An arbitrary computer file is represented as a string S_0 of bytes (often interpreted as a number between 0 and $2^8 - 1$, i.e. a value in the set $\{0 \ldots 255\}$) .
2. S_0 is encoded using a given Huffman code, converting it to base-3. This generates the string S_1 of characters in {0, 1, 2}, each such character called a 'trit'.
3. Write len() for the function that computes the length (in characters) of a string, and define $n = \text{len}(S_1)$. Represent n in base-3 and prepend 0s to generate a string S_2 of trits such that $\text{len}(S_2) = 25$. Form the string concatenation $S4 = S_2 \,.\, S_1 \,.\, S_3$, where S_3 is a string of at most 24 0s chosen so that $\text{len}(S_4)$ is an integer multiple of 25.
4. S_4 is converted to a DNA string S_5 of characters in {A, C, G, T} with no repeated nucleotides (nt) using the scheme illustrated in the figure below. (The first trit of S_4 is

coded using the 'A' row of the table. For each subsequent trit, characters are taken from the row defined by the previous character conversion.

previous nt written	next trit to encode		
	Ø	1	2
A	C	G	T
C	G	T	A
G	T	A	C
T	A	C	G

Fig 4.1: DNA Encoding Table

5. Define N = len(S_5), and let ID be a 2-trit string identifying the original file and unique within a given experiment (permitting mixing of DNA from upto $3^2 = 9$ different files S_0 in one experiment). Split S_5 into overlapping segments of length 100 nt, each offset from the previous by 25 nt. This means there will be N 25 −3 segments, conveniently indexed i ≠0 . . . N/25 −4; segment i is denoted F_i and contains (DNA) characters 25i . . . 25i + 99 of S_5.
6. If i is odd, reverse complement F_i.
7. A keystream method is used to add randomness to the pattern of bases of each segment F_i . The 100 DNA characters of F_i define 99 single-trit values by converting characters A, C, G, T to numbers 0, 1, 2, 3; then forming the successive differences by subtracting (base-4) each such number from its successor ($d_2 = n_2 - n_1$, $d_3 = n_3 - n_2$ . . . $d_{100} = n_{100} - n_{99}$); and then subtracting 1 from each of these numbers. (The fact there are no repeated nt in F_i guarantees that the results of this process are valid trits.) To each of these 99 trits, add (base-3) the corresponding trit from the jth following random keystream, where j = i mod 4:

keystream 0:
00200001011010211111212221001112222101010212122022000221201020221002121121000
212222021211121122221

keystream 1:
20202012212121012000120021022211202022202222222022000122101211102212112020202 2
211221112202002121022

keystream 2:
22122110120022112022001100222210000020200021121021020122100021201010210202002 0
001010200221211001001

keystream 3:
100122100011112100120210020011102201122122100100120122212000021220022012202201
10001021222211022202

The 99 resulting trits give the 'randomized' version of Fi by the reverse of the procedure described at the start of this step: add 1 to each to generate base-4 2 differences ($d'_2 \ldots d'_{100}$); then, starting with n_1 (unaltered from above), add (base4) successive differences to generate 100 successive values in {0,1,2,3} (n_1, $n_1 + d'_2$, $n_1 + d'_2 + d'_3 \ldots n_1 + d'_2 + d'_3 + \ldots + d'_{100}$); and then convert values 0, 1, 2, 3 to nt A, C, G, T.

8. Form the indexing information string *IX = ID . i3 . P* (comprising 2+12+1 = 15 trits). Append the DNA-encoded version of *IX* to F_i using the same strategy as at step 1.4 above, starting with the code table row defined by the last character of F_i , to give indexed segment F'_i.
9. Form F''_i by prepending A or T and appending C or G to F'_i—choosing between A and T, and between C and G, randomly if possible but always such that there are no repeated nt. (This ensures that we can distinguish a DNA segment that has been reverse complemented during DNA sequencing from one that has not—the former will start with G|C and end with T|A; the latter will start A|T and end C|G.)
10. The segments F''_i are synthesized as actual DNA oligonucleotides and stored, and may be supplied for sequencing.

4.1.2 Decoding

Decoding is simply the reverse of encoding, starting with sequenced DNA fragments F''_i of length 117 nt. Reverse complementation during the DNA sequencing procedure (e.g. during PCR reactions) can be identified for subsequent reversal by observing whether fragments start with A|T and end with C|G, or start G|C and end T|A. With these two 'orientation' nt removed, the remaining 115 nt of each segment can be split into the first 100 'message' nt and the remaining 15 'indexing' nt. The indexing nt can be decoded to determine the file identifier *ID* and the position index i_3 and hence *i*, and errors may be detected by testing the parity trit P. The value of *i* permits the identification of the correct random keystream, the randomizing effect of which can be removed by applying the keystream encoding a second time, but subtracting the keystream trits. The file identifier and position index then permit reconstruction of the DNA-encoded files, which can then be converted to base-3 using the reverse of the encoding table above and then to the original bytes using the given Huffman code.

Errors introduced during DNA synthesis, storage or sequencing could lead to various artefacts, particularly nt insertion, deletion or substitution. Recovery of information from fragments with

such errors may be possible (details left as exercise)—we have not found this to be necessary due to the large numbers of perfectly-sequenced fragments available via high-throughput sequencing.

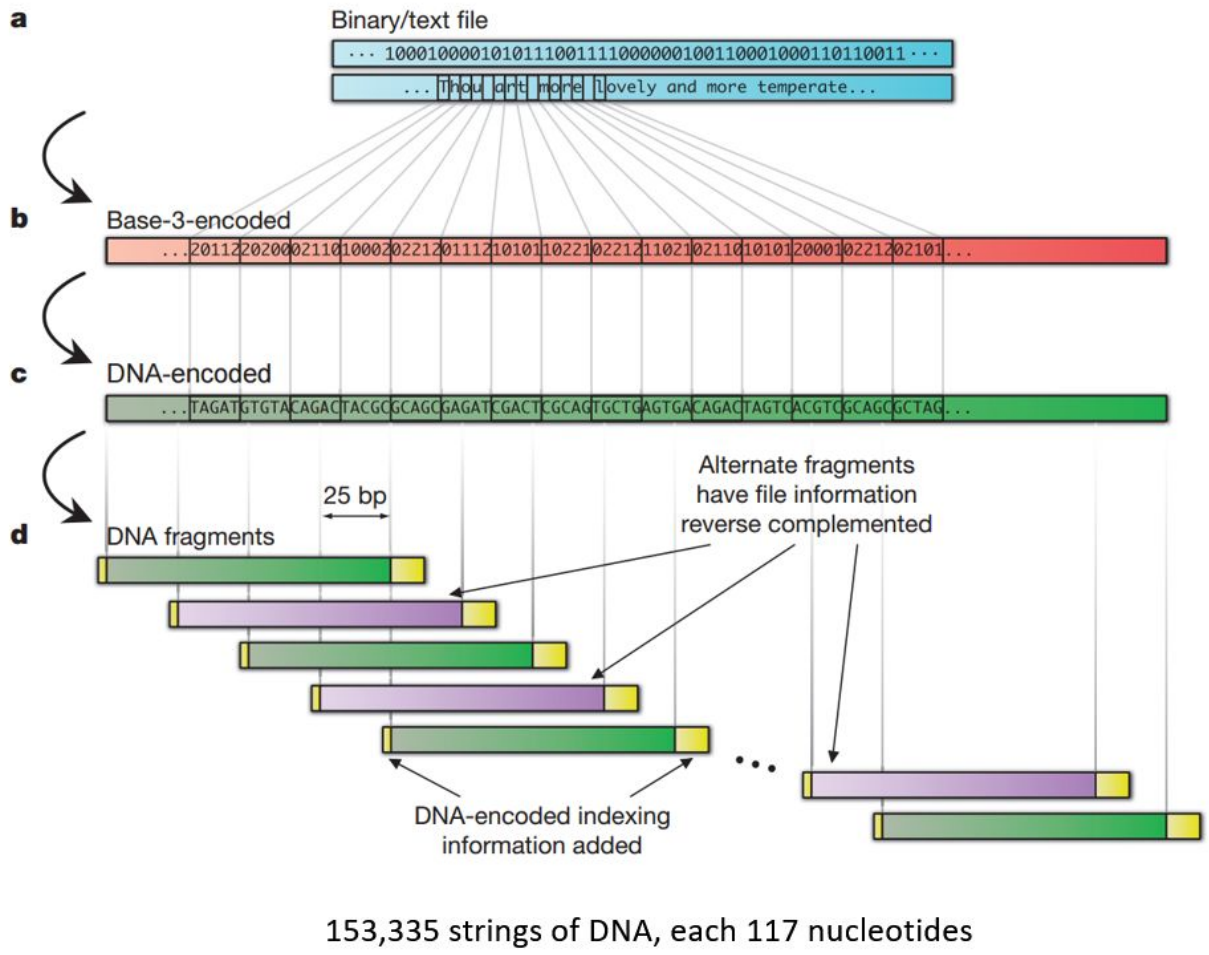


Fig 4.2: DNA Encoding Scheme

4.1.3 Example

1. As it is difficult to represent all possible bytes in this document, we use a simple example of a file comprising just 18 bytes that happen to be easily represented via ASCII codes (for clarity, we show spaces as ␣):

$$S_0 = \text{Birney␣and␣Goldman}$$

2. Using the Huffman code defined in the example file View_huff3.cd, we convert the bytes of S_0 into base-3 :

B i r n e y , , a n

S_i = 20100 20210 10101 00021 20001 222111 02212 01112 00021

22100 02212 22221 02110 02101 221002 11021 01112 00021

d , , G o l d m a n

3. n = len(S_1) = 92, which is 10102 in base-3. So:

   S_2 = 00000000000000000000 10102 (length 25)

   S_3 = 00000000 (length 8)

   $S_4 = S_2 \,.\, S_1 \,.\, S_3$

   = 00000000000000000000010102

   201002021010101000212000122211102212011120021

   22100022122222120211002101221001102101112000 21

   00000000 (length 25 + 92 + 8 = 125 = 5 × 25)

4. Using the DNA coding strategy and table shown above, convert S_4 to DNA:

   S_5 = CGTACGTACGTACGTACGTAGTCGCACTACACAGTCGACTACGCTGTACT

GCAGAGTGCTGTCTCACGTGATGACGTGCTGCATGATATCTACAGTCATC

GTCTATCGAGATACGCTACGTACGT

5. N = len(S_5) = 125, and we choose (e.g.) ID = 12. S_5 will be split into overlapping segments F_i of length 100 nt for i ∈ {0 . . . 125/25 − 4}, i.e. i ∈ {0, 1}. With overlapping parts underlined for illustration, F_0 and F_1 are:

   F_0 = CGTACGTACGTACGTACGTAGTCGC ACTACACAGTCGACTACGCTGTACT

GCAGAGTGCTGTCTCACGTGATGA GTGGCTGCATGATATCTACAGTCATC

   F_1 = ACTACACAGTCGACTACGCTGTACT GCAGAGTGCTGTCTCACGTGATGA

GTGCTGCATGATATCTACAGTCATCG

GTCTATCGAGATACGCTACGTACGT

6. Only $i = 1$ is odd, so F_1 is reverse complemented:

F_1 =

ACGTACGTAGCGTATCTCGATAGACGATGACTGTAGATATCATGCAGCAC

GTCATCACGTGAGACAGCACTCTGCAGTACAGCGTAGTCGACTGTGTAGT

7. Fragments F_0 and F_1 are randomized using keystreams 0 and 1, respectively. Full details of the example are omitted, but after keystream randomization these become:

F_0 =

CGTGTACGACTCGACAGAGATGCAGAGACTAGACACTCTCTATACGCATC

TCAGACACTCGTAGTCTACTGACATCGCACGAGAGAGAGCAGCGCACTCA

F_1 = ATATATAGCGTGATAGCGTCAGTCGCATCGCACACGAGCTCTCTCTAGCA

TAGCGTCTAGATGCGACGAGCGAGCTCATGTGATCTGCGTACTCTCTACA

8. For $i = 0$, i_3 = 000000000000 (length 12) and the sum (mod 3) of the odd-positioned trits of ID and i_3 is $P = 1 + 0 + 0 + 0 + 0 + 0 + 0 \pmod 3 = 1$. For $i = 1$, i_3 = 000000000001 and $P = 1 + 0 + 0 + 0 + 0 + 0 + 0 \pmod 3 = 1$.
9. For $i = 0$, $IX = ID \,.\, i_3 \,.P$ = 120000000000001; for $i = 1$, IX = 120000000000011. So:

F'_0 =

CGTGTACGACTCGACAGAGATGCAGAGACTAGACACTCTCTATACGCATC

TCAGACACTCGTAGTCTACTGACATCGCACGAGAGAGAGCAGCGCACTC

AGC GTACGTACGTAC T (length 100 + 15 = 115)

F'_1 =

ATATATAGCGTGATAGCGTCAGTCGCATCGCACACGAGCTCTCTCTAGCAT

AGCGTCTAGATGCGACGAGCGAGCTCATGTGATCTGCGTACTCTCTACAG

C GTACGTACGTAG A (length 115)

10. Prepend A|T and append C|G (note that in this example we have three random choices, but no choice at the start of F''_1):

F''_0 = A
CGTGTACGACTCGACAGAGATGCAGAGACTAGACACTCTCTATACGCA

TC
TCAGACACTCGTAGTCTACTGACATCGCACGAGAGAGAGCAGCGCAC

TCA GCGTACGTACGTACT G (length 1 + 115 + 1 = 117)

F''_1 = T
ATATATAGCGTGATAGCGTCAGTCGCATCGCACACGAGCTCTCTCTAGC

A
TAGCGTCTAGATGCGACGAGCGAGCTCATGTGATCTGCGTACTCTCTAC

A GCGTACGTACGTAGA C (length 117)

4.2 DNA converter Client (Web App)

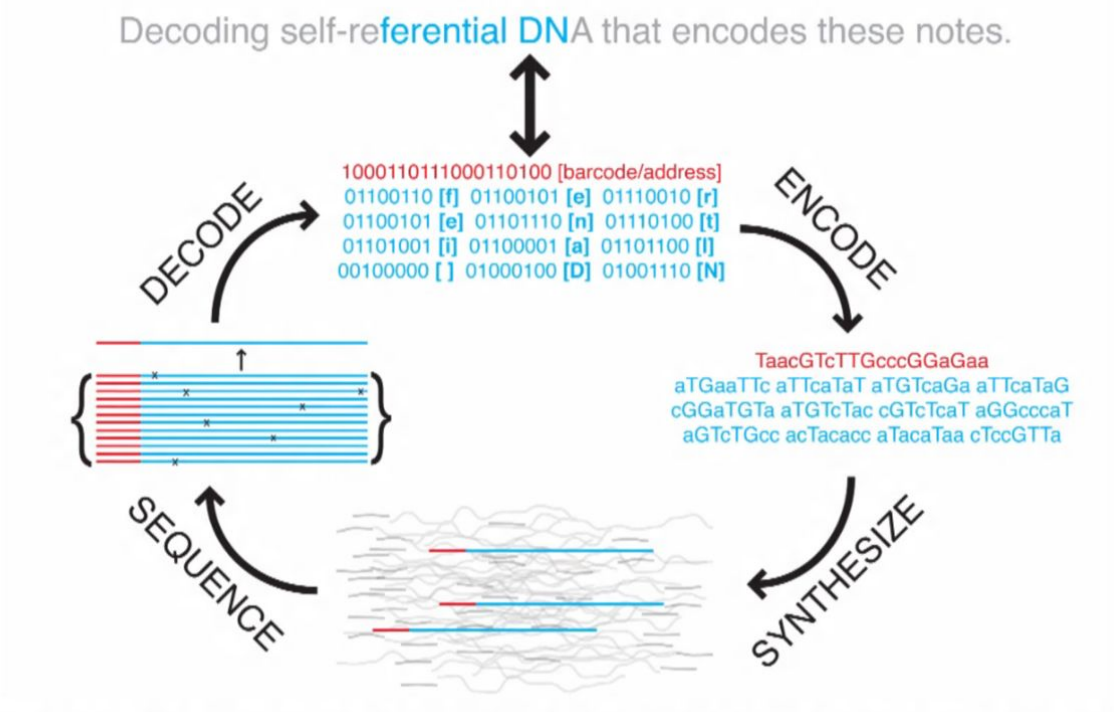


Fig 4.3: DNA encoding-decoding scheme

.DNA file can be downloaded and can be written into actual DNA in a wet lab.

The same data file when read back from DNA can be restored.

A front end application was developed, which facilitates any kind of file uploading and converts that into DNA format. This client also decodes the DNA encoded file back into original format.

The user is given an option to upload the file where he can upload any file format and that file will be converted into .DNA format file and the same can be downloaded and stored.

To decode the file, user can upload .decoded format file and the original file can be downloaded using the same application.

The application automatically detects whether you want to encode a file or decode an already encoded file. This is done using special checks performed while file uploading.

Technologies used to build the client application:

- React and JavaScript for frontend
- NodeJS for backend API
- Commands are run using shelljs

This application is also mobile friendly and supports all file formats.

The technology architecture:

- Front is written in reactJs which gives modularity to the project, i.e same code component if repeated can be reused hence saves time and redundancy.
- When user clicks upload file, a dialogue box is open from where he can choose the file.
- Next, once the file is selected it is sent to the backend server via Node JS api where the file is saved in /uploads directory.
- Once the file is uploaded to the server, the file path is given to the dna converted which is executed with the help of shellJS.
- the dna converted commands convert the file into dna format and saves with the same file name with .dna extension
- The file path is then sent to the client with the server response request.
- When user clicks the download button, the same file can be downloaded from the server.

4.2.1 Cost Estimate

Uploading a 10 kb file:
10 kb Image file converts to 50 kb of DNA format file.

As one char, which was earlier taking 8 bits now occupying 40 bits.

Cost of writing 50 kb data = 50 x 1024 x 8 X $0.01 (cost of writing 1 nucleotide) = $4096

4.2.2 Reading the data back from the DNA

MinION is a portable device which reads the sequence of DNA in the matter of few hours which earlier took days.

MinION is the only portable real-time device for DNA and RNA sequencing.

Each consumable flow cell can now generate 10–20 Gb of DNA sequence data. Ultra-long read lengths are possible (hundreds of kb) as you can choose your fragment length. The MinION streams data in real time so that analysis can be performed during the experiment and workflows are fully versatile.

Fig 4.4: MinION

The MinION weighs under 100 g and plugs into a PC or laptop using a high-speed USB 3.0 cable. No additional computing infrastructure is required. Not constrained to a laboratory environment, it has been used up a mountain, in a jungle, in the arctic and on the International Space Station.
The MinION is commercially available, simply by paying a starter-pack fee of $1,000.

4.3 Image Summarization

Given a collection of n images X = {X1,X2, ...,Xn }, we aim to find a subset summary of these images, while preserving the relevance and diversity.

Image collection summarization is an important task in multimedia processing, vision and story-telling. The summarization can be categorised into three different class of works namely, summary based on Dictionary Learning, using personal albums and using Submodular functions. We have thought of novel means incorporating machine learning for the task of summarization.

In order to evaluate the summary, precision and recall are usually applied. However, the precision and recall metrics define quantitatively how good a summary only when they are provided with user annotated summary. This limits the scope to datasets where annotations are available. Further, it also adds an issue of annotating the data with relevance to a particular task for which summary is desired. It is possible that with change of task, the images needed in summary may change and hence ground truth may also change.

Summarization has been well explored in videos for efficient browsing and other applications, though image corpus summarization has not received an equivalent attention. This is partly because, videos have the redundancy in the temporal dimension which can be effectively learnt and reduced, whereas image corpus need not have such redundancy. Thus the problem becomes quite challenging. Summary of image corpus can be both qualitatively and quantitatively analyzed based on factors of relevance and diversity, which we define as,

1. Relevance means how relevant is a particular image to a given task such as classification, segmentation or detection. For example, a certain image may be relevant for a specific task and should be captured in the summary, whereas, for a different task this image may not be desired as a part of the summary.
2. Diversity maintains that all the images that are distinct are included in the summary and must not contain any redundancy we also propose another view to diversity as that of volume of the high dimensional parallelepiped. If a image is redundant it's high dimensional feature must lie in the subspace formed by linear combination of features of rest of images.

4.3.1 Learning Framework

We propose a novel model for the task of summarization based on our research. The idea of model is simple enough to be intuitive to understand. The machine learning model gives us some high dimensional feature vectors that capture the essence of image. We can use those model features to understand how correlated two images are using clustering of the feature vectors.

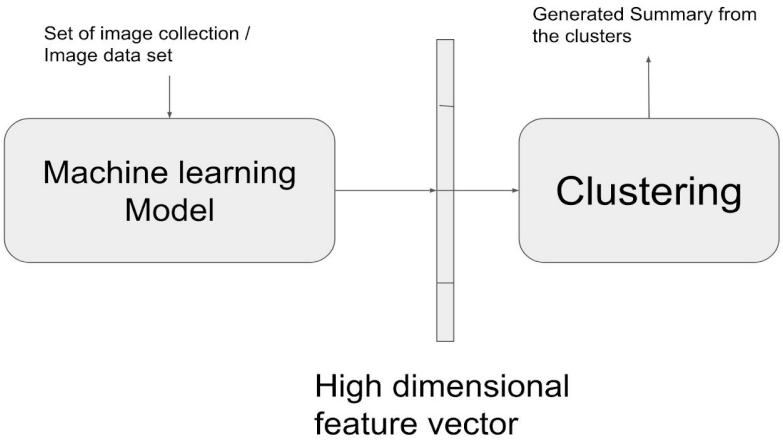


Fig 4.5: Learning Framework

4.3.2 Variants and their ablation study

Convolutional neural networks There can be multiple variations of the above general purpose model that is used for the task of summarization and hence we have thought about few of them. We studied the Convolutional Neural Network when used as the machine learning model and when Auto encoders where used as a machine learning model for finding the hidden latent representation of the images. The convolutional neural network is a model that works upon the principle of convolution where the input is a image. The image is then convolved by the filter. The size of filter can depend on the designer who is creating the architecture of the convolutional neural network. So the convolutional neural network takes image and applies the filter. It must be noted that the depth of the filter must be same as the dept of image in each layer. Now the filter is made up of a matrix of coefficients which do nothing but take the weighted average of all the pixel values in that filter size of the image. The weighted average is the added with some bias and passed via some activation function to obtain the new value of the next layer representation of the image. Then the filter slides onto the next part of the image where we can have step size equal to length / width of the filter which results in a case called as no-overlapping. Usually the step size is less than the width of the filter so there is some overlap in case of each layer. This process is cascaded on each level and finally when the data representation is small enough a fully connected neural network can be used.

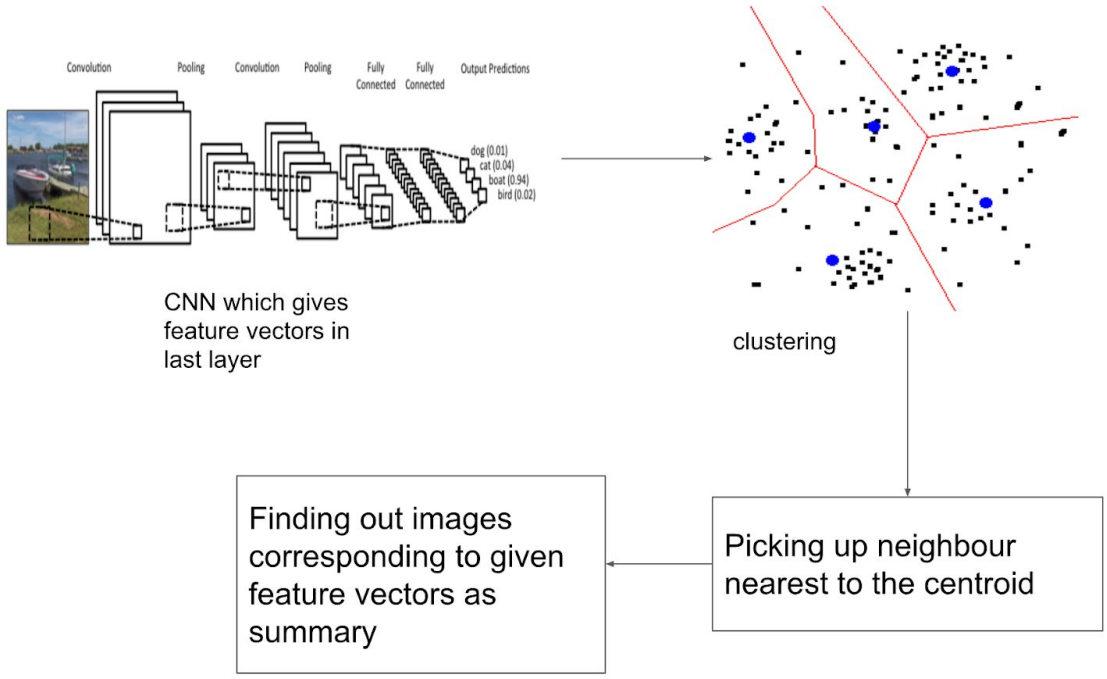


Fig 4.6: Learning Framework

The model can be pre-trained for a particular general task since model is being used to perform a specific task it is a mapping function. That will map 2 images which are to be produced in similar fashion near each other. So if a convolutional neural network trained for classification task being used and lets say for a data set where a particular class 'cat' is there we can safely assure that there are not much examples that are misclassified after the convolutional neural network was completely trained to fullest. The argument below shows that the images which are similar to each other in labels are likely to cluster unless the given image is a outlier example of that class. Then using a pre-trained classifier that was trained for a task other than summarization must not prove to be a bottleneck in the situation because the same class labelled images are together in their original representation and are likely to remain together in the task of classification. It is also empirically proved that this assumption holds and has been verified for the mnist dataset.

Then it can be evaluated that the images will cluster together in the feature representation of the data set which is pretty easy to observe in the T-sne evaluation of a mnist digits dataset.

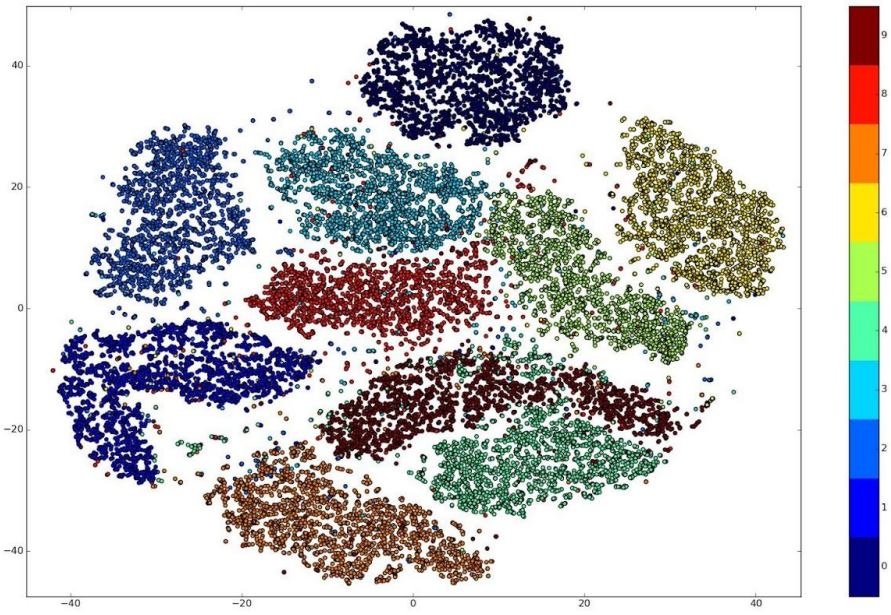


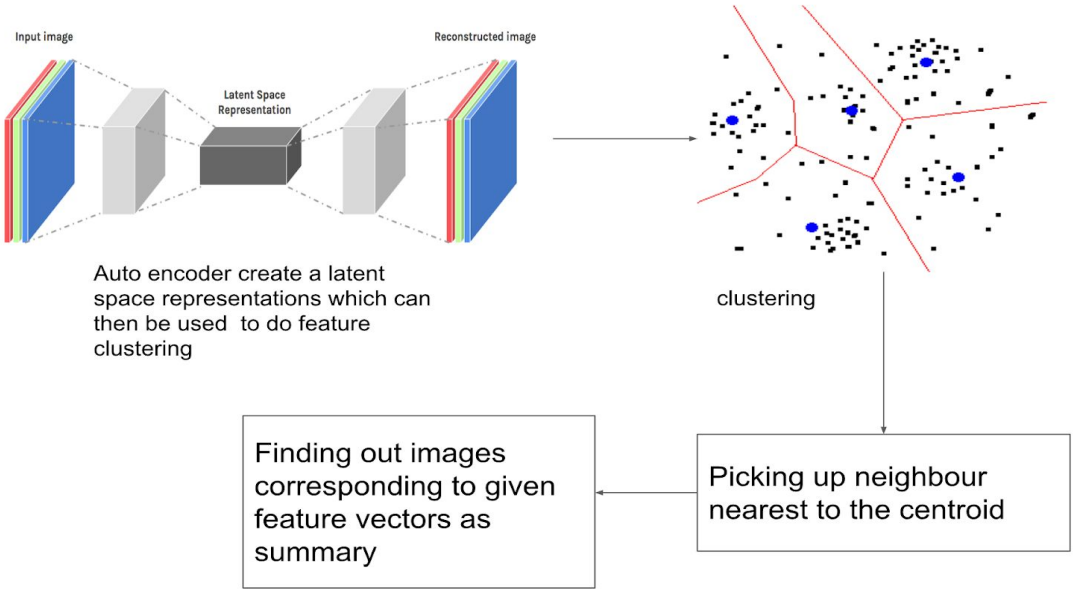
Figure 4.7: The t sne-visualization of mnist digits dataset which shows that images which are having similar features for a same class in a given task are likely to cluster together.

In the latent feature representation that is used to separate and classify the given data set according to that task. The cluster of feature vector representations are then used to summarize the data set. Within each cluster a image representation that is present can be taken and the feature vector nearest to the centroid is picked as the part of the summary set. Then the image corresponding to that feature vector is identified from the index of the feature vector as both have same indices. Now using that set of image summary set is built upon. In our experiments we had choice of using a pre-trained classifier / machine learning model to generate the feature vectors and it can be absolutely possible to get away with training as the model will the create a summary based on the given feature representation which will highlight summary on the basis of the task that was performed by the machine learning model. That is the task for which the feature representation were generated.

The implementation uses pretrained vgg16 model to train the feature vectors and then cluster to form the summary.

Figure 4.7: The architecture of vgg16 model which was pre-trained for classification task on imagenet dataset.

Model variant using Autoencoders



.

figure 4.8 :The second variation of the model is a where the machine learning model being used is a Auto encoder.

An autoencoder learns to compress data from the input layer into a short code, and then uncompress that code into something that closely matches the original data. This forces the autoencoder to engage in dimensionality reduction, for example by learning how to ignore noise. Some architectures use stacked sparse autoencoder layers for image recognition. The first autoencoder might learn to encode easy features like corners, the second to analyze the first layer output and then encode less local features like the tip of a nose, the third might encode a whole nose, etc., until the final autoencoder encodes the whole image into a code that matches (for example) the concept of "cat". An alternative use is as a generative model: for example, if a

system is manually fed the codes it has learned for "cat" and "flying", it may attempt to generate an image of a flying cat, even if it has never seen a flying cat before.The autoencoder tries to learn a function hW,b(x)≈x. In other words, it is trying to learn an approximation to the identity function, so as to output x(hat) that is similar to x. The identity function seems a particularly trivial function to be trying to learn; but by placing constraints on the network, such as by limiting the number of hidden units, we can discover interesting structure about the data. As a concrete example, suppose the inputs xx are the pixel intensity values from a 10×1010×10 image (100 pixels) so n=100n=100, and there are s2=50s2=50 hidden units in layer L2L2. Note that we also have y∈R100y∈ℜ100. Since there are only 50 hidden units, the network is forced to learn a ”compressed” representation of the input. I.e., given only the vector of hidden unit activations a(2)∈R50a(2)∈ℜ50, it must try to ”‘reconstruct”’ the 100-pixel input xx. If the input were completely random—say, each xixi comes from an IID Gaussian independent of the other features—then this compression task would be very difficult. But if there is structure in the data, for example, if some of the input features are correlated, then this algorithm will be able to discover some of those correlations. In fact, this simple autoencoder often ends up learning a low-dimensional representation very similar to PCAs.Otherwise, one reason why they have attracted so much research and attention is because they have long been thought to be a potential avenue for solving the problem of unsupervised learning, i.e. the learning of useful representations without the need for labels. Then again, autoencoders are not a true unsupervised learning technique (which would imply a different learning process altogether), they are a *self-supervised* technique, a specific instance of *supervised learning* where the targets are generated from the input data. In order to get self-supervised models to learn interesting features, you have to come up with an interesting synthetic target and loss function, and that's where problems arise: merely learning to reconstruct your input in minute detail might not be the right choice here. At this point there is significant evidence that focusing on the reconstruction of a picture at the pixel level, for instance, is not conductive to learning interesting, abstract features of the kind that label-supervised learning induces (where targets are fairly abstract concepts "invented" by humans such as "dog", "car"...). In fact, one may argue that the best features in this regard are those that are the *worst* at exact input reconstruction while achieving high performance on the main task that you are interested in (classification, localization, etc).

4.3.3 Dataset used

We use publicly available datasets to train and test our model. Since there are very few datasets which have summary annotations in terms of relevance and diversity, we test on both datasets - where annotations are available as well as where it is unavailable. We evaluate our model using following datasets: CIFAR-10 [11], CIFAR-100 [11], Animals with Attributes 2 (AwA2) [28], VOC2012 [2] and Diversity 2016 [7]. CIFAR-10 consists of 60,000 32X32 tiny images belonging to 10 classes with similar images per class. There are 50,000 training and 10,000 test images. CIFAR-100 consists of similar 60,000 32x32 tiny images with 600 images per class. The

classes are divided into 20 super-classes with 5 classes per super-class. AwA2 is another data set used for classification purposes. It contains 37,322 images of 50 animal classes. Visual Object Classes (VOC 2012) is another image classification data set with 20 classes and 11,530 images. Diversity 2016 contains the images with corresponding ground truth images for task of diversity in image retrieval. Images are ranked according to their importance within a class.We use Top-50 ranked images from each class to create the ground truth. There are 20,821 images of multiple classes with each class containing approximately 300 images.

4.3.4 Evaluation

To evaluate our model both quantitatively and qualitatively we use several metrics that discriminate between summaries on how good/bad a summary is compared to another summary.The qualitative and quantitative evaluations of summary are described below.

Gini Coefficient: In order to measure diversity, we compute Gini index [3]. For a piecewise differentiable cumulative distribution function F(c) which has mean μ, and is zero for all negative values of c, Gini Index is defined as follows,

$$G = 1 - \frac{1}{\mu}\int_0^\infty (1 - F(y))^2 dy = \frac{1}{\mu}\int_0^\infty F(y)(1 - F(y))dy \quad (1)$$

We compare these indices computed for randomly picked images from dataset, images picked corresponding to centers of K-means clusters and for images from summary, and plot in Fig.4.We use Gini index because this metric inherently gives uniform weightage to all the classes present in dataset. This is important as summary must comprise images from all classes. In case of random summary generation, this uniformity cannot be achieved. Moreover since images are not robust to translations and rotations type of function transformations their clustering is also highly likely to no not give equal weight to all the classes. We find out the gini coefficient for all the 4 datasets cifar 10 , cifar 100 , awa2 , voc2012 at different values for k where k=σ n where we choose different values of σ are illustrated in the graph below.

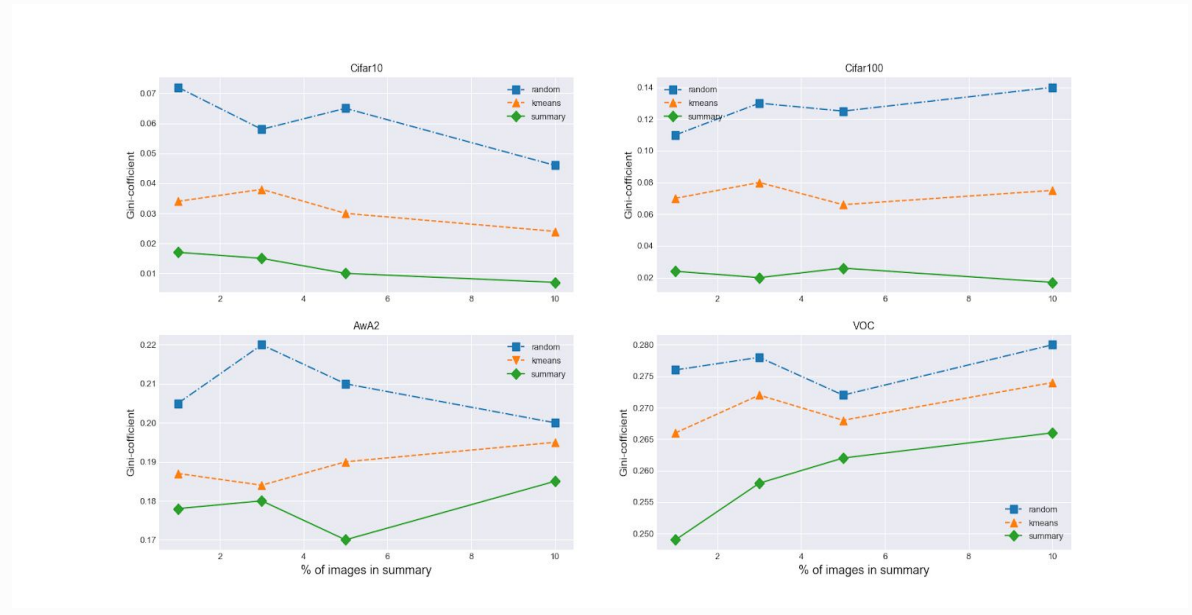


Figure 4.9: The gini coefficient is evaluated for multiple summary lengths where the number of image in summary is one percent , three percent , five percent and ten percent of the total dataset. Where for K=σ n and σ = 0.01, 0.03, 0.05 and 0.1.

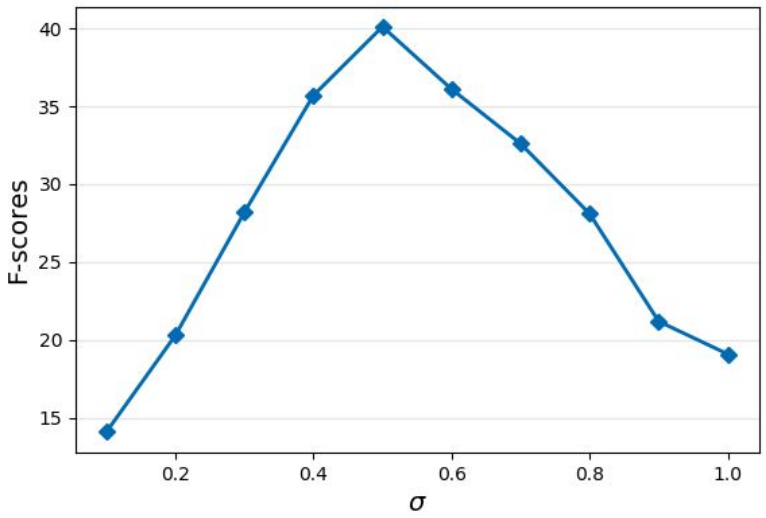


Figure 4.10: the above figure shows the Fscore for the summary generated on the Diversity 2016 dataset.

F-scores: The **F score** is a measure of a test's accuracy. It considers both the Precision *p* and the Recall *r* of the test to compute the score: *p* is the number of correct positive results divided by the number of all positive results returned by the classifier, and *r* is the number of correct

positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

precision= ratio of number of images in intersection of summary and ground truth to number of images in summary

recall= ratio of number of images in intersection of summary and ground truth to number of images in ground truth

4.3.5 Qualitative Analysis

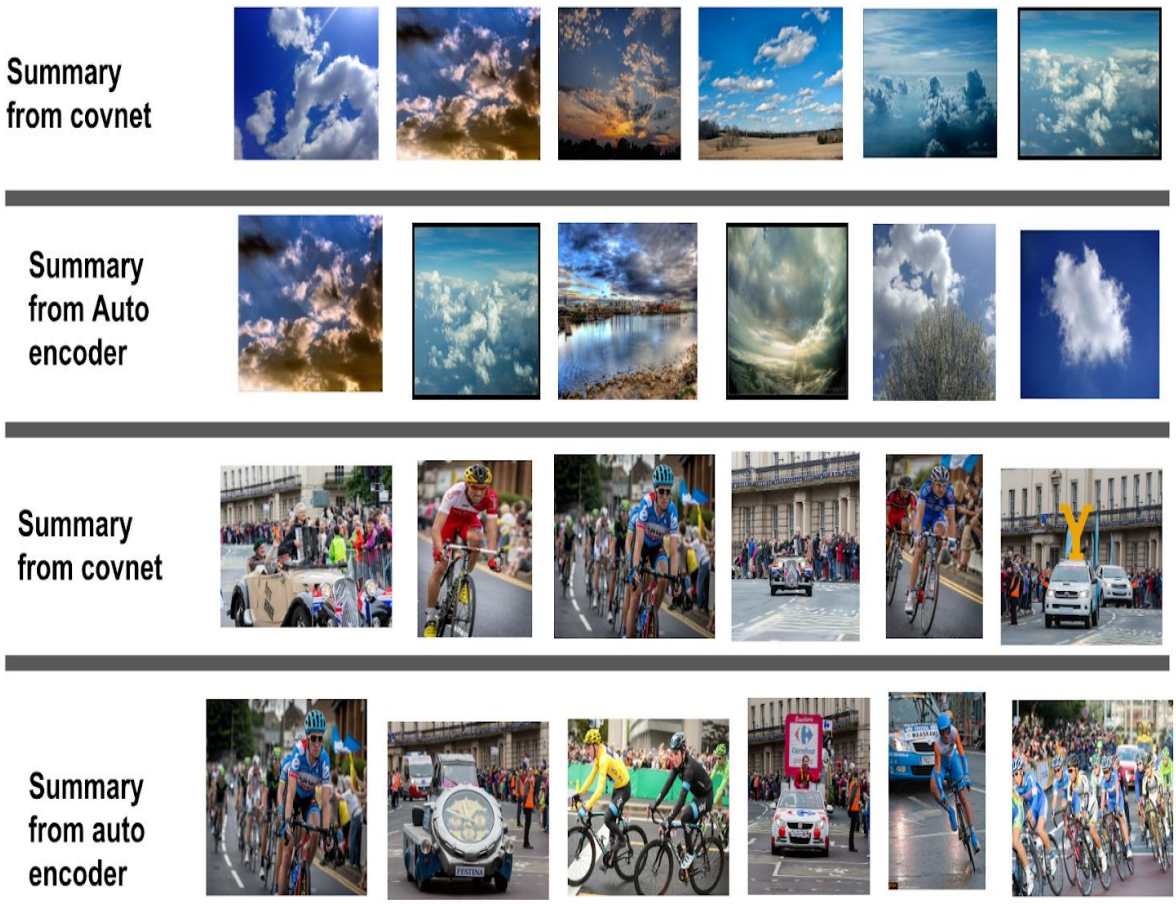


Figure 4.11: The above figure shows a qualitative comparison of both the models in their approach to summarizing the Diversity 2016 data set for the two classes of images clouds and the Tour de France events. We show top 5 images for each class. The images are ranked in the

ground truth of the data set itself what we do is compare from the summary which of the top ranked images from the ground truth are found in the summary of the dataset.

4.4 Image Compression as Summarization

Image compression is a type of data compression applied to digital images, to reduce their cost for storage or transmission. Algorithms may take advantage of visual perception and the statistical properties of image data to provide superior results compared with generic compression methods.Image compression may be lossy or lossless. Lossless compression is preferred for archival purposes and often for medical imaging, technical drawings, clip art, or comics. Lossy compression methods, especially when used at low bit rates, introduce compression artifacts. Lossy methods are especially suitable for natural images such as photographs in applications where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate. Lossy compression that produces negligible differences may be called visually lossless.

Figure 4.12: Leena image which is used as a standard for testing all image compression algorithms

For the task of compression what we do is we break the given image into set of blocks of different sizes 64X64, 32X32 , 16X16, 8 X 8 then we pass the each block from the pre trained convolutional neural network to obtain feature representation of our image blocks once we have that feature representation. We use the current clustering method to cluster the image and then select the nearest feature vectors to the centroid that form the basis of these image vectors. In principle the idea works to find the bases which govern the image in totality. Using those bases

what we do is for each block in a given cluster we replace that block with the bases representative of that block.

4.5 Iterative Shrinkage Thresholding

Traditional methods for image compressive sensing (CS) reconstruction solve a well defined inverse problem (convex optimization problems in many cases) that is based on a predefined CS model, which defines the underlying structure of the problem and is generally solved by employing convergent iterative solvers. These optimization-based CS methods face the challenge of choosing optimal transforms and tuning parameters in their solvers, while also suffering from high computational complexity in most cases. Recently, some deep network based CS algorithms have been proposed to improve CS reconstruction performance, while dramatically reducing time complexity as compared to optimization-based methods. Despite their impressive results, the proposed networks (either with fully-connected or repetitive convolutional layers) lack any structural diversity and they are trained as a black box, void of any insights from the CS domain. In this paper, we combine the merits of both types of CS methods: the structure insights of optimization-based method and the performance/speed of network-based ones. We propose a novel structured deep network, dubbed ISTA-Net, which is inspired by the Iterative Shrinkage-Thresholding Algorithm (ISTA) for optimizing a general l1 norm CS reconstruction model. ISTA-Net essentially implements a truncated form of ISTA, where all ISTA-Net parameters (e.g. transforms, shrinkage thresholds, step size, etc.), are learned end-to-end to minimize a reconstruction error in training. Borrowing more insights from the optimization realm, we propose an accelerated version of ISTANet, dubbed FISTA-Net, which is inspired by the fast iterative shrinkage-thresholding algorithm (FISTA). Interestingly, this acceleration naturally leads to skip connections in the underlying network design. Extensive CS experiments demonstrate that the proposed ISTA-Net and FISTA-Net outperform existing optimization-based and network-based CS methods by large margins, while maintaining a fast runtime with 20 fps on a Quadro 6000 GPU.Compressive Sensing (CS) has drawn much attention as an alternative to the classical paradigm of sampling followed in data compression [1]. From much fewer acquired measurements than determined by Nyquist sampling theory, CS theory demonstrates that a signal can be reconstructed with high probability when it exhibits sparsity in some transform domain [2]. This novel acquisition strategy is much more hardware-friendly, which allows image or video capturing with a sub-Nyquist sampling rate. In addition, by exploiting the redundancy existing in a signal, CS conducts sampling and compression at the same time, which greatly alleviates the need for high transmission bandwidth and large storage space, enabling low-cost on-sensor data compression. CS has been applied in many practical applications, e.g. accelerating magnetic resonance imaging (MRI).

The purpose of compressive sensing is to reconstruct the original signal x ∈ R N from its randomized CS measurements y = Φx ∈ RM. Here, Φ ∈ RM×N is a random projection. Because M N, this problem is typically ill-posed. However, CS theory demonstrates that a signal can be reconstructed with high probability when it exhibits sparsity in some domain. In the past decade, many algorithms were proposed to solve the image CS reconstruction.problem and we generally divide them into two categories: traditional optimization-based CS methods and recent network-based CS methods.

Traditional Optimization-based CS Reconstruction. A signal x is said to be sparse with respect to some transform Ψ, if its transform coefficients α = Ψx are mostly zeros, or nearly sparse if the dominant portion of coefficients are either zeros or very close to zeros. The sparsity of x in Ψ is quantified by the number of significant elements within the coefficient vector α. Given the linear measurements y and according to CS theory, traditional image CS methods usually reconstruct the original image x by solving the following convex optimization problem where the sparsity of the vector is characterized by l1 norm, which stands for adding all the absolute values of the entries in a vector.ponding dj. For the problem of automatic image summarization, fdjg is the set of the most representative images that we want to learn, and fdjg should come from the original image set. The size of fdjg (summary) is a trade-off between concise summarization of the original image set and accurate reinterpretation of the original image set: a small size of fdjg means more concise summarization of the original image set but its reinterpretation power for the original image set may reduce; on the other hand, a large size of fdjg guarantees a better reinterpretation power but the summarization could be verbose. The idea of this proposed reconstruction model (for automatic image summarization) is similar to nonnegative matrix factorization which learns the prominent objects or major components of an image set. In our problem for automatic image summarization, the summary (which is learned in this manner) is inclined to be composed by the salient visual components of the original image set. If we heavily penalize on the sparsity term a (such as JaJ0 ¼ 1) which is used for determining the number of bases for reinterpretation, our proposed model for automatic image summarization can be reduced to k-medoids (the discrete form of kmeans). The k-medoids algorithm is well known as one of the effective methods for collection summarization [7]. Thus, our proposed approach for automatic image summarization via dictionary learning for sparse representation can be treated as an extension of the k-medoids. Consequently, considering that the richness of the visual content of an image is limited, it is necessary to bring in the sparsity constraint to the objective function for guaranteeing that only a limited number of bases may take effect in the reconstruction. Hence, only the bases with non-zero coefficients are used to reconstruct the images in the original image set. Meanwhile, the bases should be diverse; each basis represents one type of principal visual patterns and all these bases should be different from each other. Thus the diversity constraint should be included in the objective function for dictionary learning.The reconstruction of the image can be defined using the bases of the image blocks that are part of

the summary and the error in reconstruction is given using the following objective is also called as the reconstruction error of the image.

The above equation consists only of the reconstruction terms and thus has a l2 norm of the difference in the original and reconstructed image to define the reconstruction error. We rewrite Eq. (2) as follows by adding both the sparsity constraint and the diversity constraint.

where K is the total number of samples in the data set and and n is the number of image blocks in the dictionary. The objective function above in previous equation is non increasing under following update rule.

R(t+1)= R(t)'*'(transpose(D)X)'/'(transpose(D)DR(t)+ 1) (3)

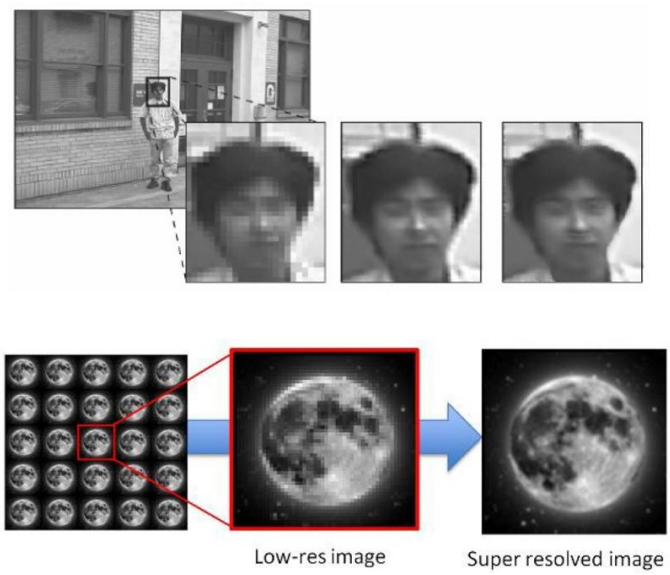
where R is the transformation matrix that describe which of the given contains the non negative coefficients for transformation of the the summary or dictionary into the original data set. It can be termed as reconstruction.

4.6 Image Super resolution

It is a Technique that is used to obtain a high resolution image from given low resolution image/s. It has application in various fields such as surveillance, traffic cameras, medical imaging and remote sensing.

The low-resolution images taken as inputs can come from a wide variety of sources, from television broadcasts and holiday snaps to surveillance footage and satellite terrain imagery. These sets of low-resolution images are related to the high-resolution version of the scene by imaging parameters, including the point-spread function, lighting variations, and geometric warping. In order to perform super- resolution by exploiting the relative sub-pixel motion between the scene and the imaging planes, we need to estimate these imaging parameters.

Many attempts have been made to make the process of Super Resolution more efficient and as optimum as possible. There are various approaches, most typical ones being, use of cross-correlation to register images, followed by the inversion of transformation from the unknown high resolution image to the observed low resolution image/s, using regularization to resolve the ill-posed nature of the inversion process



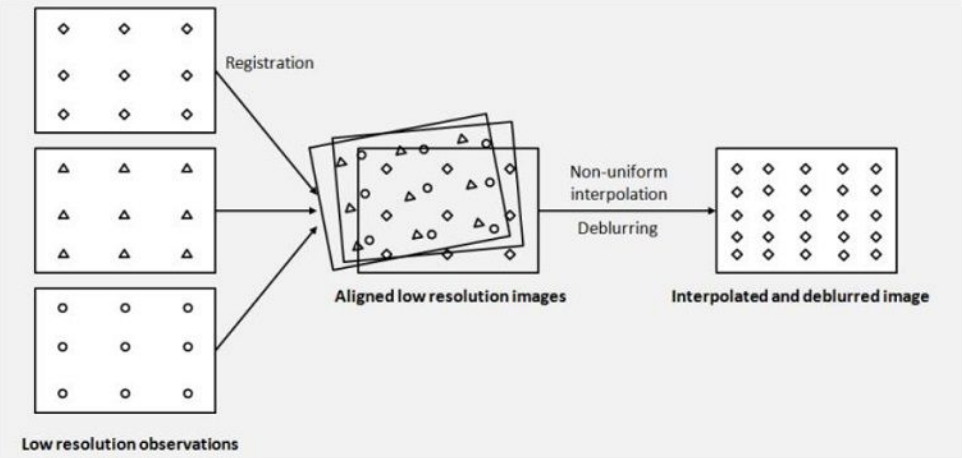
Types of Super Resolution

1) Multi-image super resolution
2) Single-image super resolution

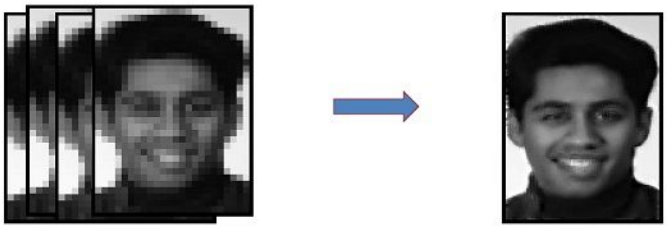
Multi-image super resolution

Multiple image (or classical) SR algorithms are mostly reconstruction-based algorithms, i.e., they try to figure out the aliasing artefacts that is present in the observed low resolution images due to under-sampling process.

- Several images of the same scenery.
- Each image has different information of the same scenery.



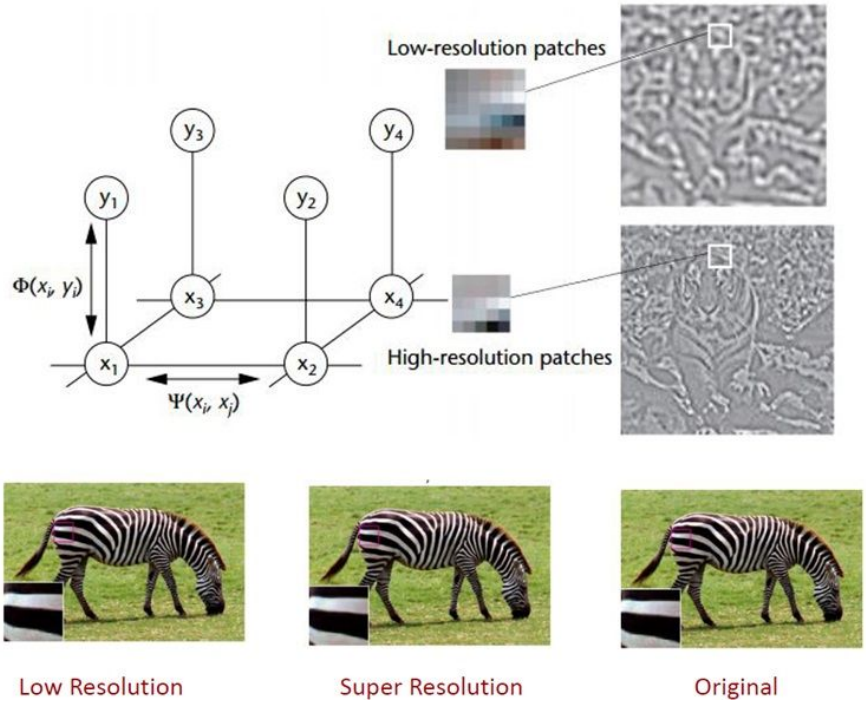
Reconstruct HR image through multiple LR images



Single-image super resolution

In the process of sub-sampling or decimation of an image, the high frequency information may get lost. In ML and MAP algorithms the generic smoothness priors can regularize the solution but cannot help recover the lost frequencies, especially for high improvement factors. In single image based super resolution algorithms, these generic priors are replaced by more meaningful and class wise priors like. This is because images from the same class have similar statistics and the accuracy of multiple-image based SR algorithms is highly dependent on the estimation accuracy of the motions between the LR observations, which gets more unstable in real world applications where different objects in the same scene can have different and complex motions. In situations like these, single image based SR algorithms may work better. There algorithms are either reconstruction based (similar to multiple image based algorithms) or learning based.

- Correspondences between low and high resolution image patches are learned from a database of low and high resolution image.



Popular Attempts and their Discussion

Bayesian Image Super-resolution

(Michael E. Tipping and Christopher M.Bishop)

- It is a type of multi-image super resolution.
- In Bayesian treatment of the super-resolution problem in which the likelihood function for the image registration parameters is based on a marginalization over the unknown high-resolution image.
- This approach allows us to estimate the unknown point spread function, and is rendered tractable through the introduction of a Gaussian process prior over images.
- Task in super resolution is to combine a set of low resolution images of the same scene in order to obtain a single image of higher resolution.
- Ideally the low resolution images would differ only through small translations, and would be otherwise identical.
- Larger changes in camera position or orientation can be handled using techniques of robust feature matching, constrained by the epipolar geometry, but such sophistication is unnecessary in present context.
- Previous approaches performed initial registration of low resolution images with respect to each other, and then keep the registration fixed.
- They formed probabilistic model of image generation process and use maximum likelihood to determine the pixel intensities of high resolution image.
- Difficulty in these techniques is that if the high resolution image has too few pixels then not all of the available high frequency information is extracted from the observed images.
- Whereas if it had too many pixels the maximum likelihood solution becomes ill conditioned.

Technique

- We are given K low-resolution intensity images.
- We represent images as vectors Y(k) of length M, where k= 1,2,…K. obtained by raster scanning the pixels of the images.
- Each image is shifted and rotated relative to a reference image(y(1)).
- The shifts are described by 2D vectors S_k and rotations by θ_k.
- Prior over the high resolution image by Gaussian process.

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{Z}_x) \qquad Z_x(i,j) = A \exp\left\{-\frac{\|\mathbf{v}_i - \mathbf{v}_j\|^2}{r^2}\right\}$$

V_i= Spatial position of the 2D image space of the pixel i.
A= Strength of the prior.
Z_x= Fixed matrix.

- As low resolution images are assumed to be generated from high resolution image.

$$\mathbf{y}^{(k)} = \mathbf{W}^{(k)}\mathbf{x} + \boldsymbol{\epsilon}^{(k)}$$

$\boldsymbol{\epsilon}^{(k)}$ = vector of independent Gaussian random variables $\epsilon_i \sim \mathcal{N}(0, \beta^{-1})$
β Representing noise terms.

$\mathbf{W}^{(k)}$ is given by point spread function.

$$W_{ji}^{(k)} = \widetilde{W}_{ji}^{(k)} / \sum_{i'} \widetilde{W}_{ji'}^{(k)} \qquad \widetilde{W}_{ji}^{(k)} = \exp\left\{-\frac{\|\mathbf{v}_i - \mathbf{u}_j^{(k)}\|^2}{\gamma^2}\right\}$$

- Vector $\mathbf{u}_j^{(k)}$ is the centre of PSF and is dependent upon the shift and rotation of the low resolution image.

$$\mathbf{u}_j^{(k)} = \mathbf{R}^{(k)}(\mathbf{v}_j - \overline{\mathbf{v}}) + \overline{\mathbf{v}} + \mathbf{s}_k \qquad \mathbf{R}^{(k)} = \begin{pmatrix} \cos\theta_k & \sin\theta_k \\ -\sin\theta_k & \cos\theta_k \end{pmatrix}$$

- We can write down likelihood function in the form.

$$p(\mathbf{y}^{(k)}|\mathbf{x}, \mathbf{s}_k, \theta_k, \gamma) = \left(\frac{\beta}{2\pi}\right)^{M/2} \exp\left\{-\frac{\beta}{2}\|\mathbf{y}^{(k)} - \mathbf{W}^{(k)}\mathbf{x}\|^2\right\}$$

- Assuming the images are generated independently from the model, we can write posterior distribution over high resolution image and high resolution image can be obtained by maximizing its numerator.

Deep Convolutional Networks

(Chao Dong, Chen Change Loy, Member, IEEE, Kaiming He, Member, IEEE, and Xiaoou Tang, Fellow, IEEE)

- It is a type of single-image super resolution.
- Directly learns an end-to-end mapping between the low/high-resolution images. The mapping is represented as a deep convolutional neural network (CNN).
- Structure is simple, but provides superior accuracy compared to state-of-the-art methods.
- Since it is a fully feed-forward network, it is unnecessary to solve the optimization problem.
- Restoration quality can be further improved with more diverse data and/or more deeper network without changing the core structure of the network.
- SRCNN model can also cope with channels of color images simultaneously with ease, which in turn can improve performance

Technique

1) We upscale the image to a desired size using bicubic interpolation method. This is just a pre-processing step.

2) Patch Extraction and Representation

- Densely extracts patches and then represents them as a set of filters.
- This layer is expressed as a function $F_{1,}$ where
  - § $F_1(Y) = \max(0, W_1 * Y + B_1)$ (4)
- This layer extracts a n_1 –dimensional feature for each patch.

3) Non – Linear Mapping

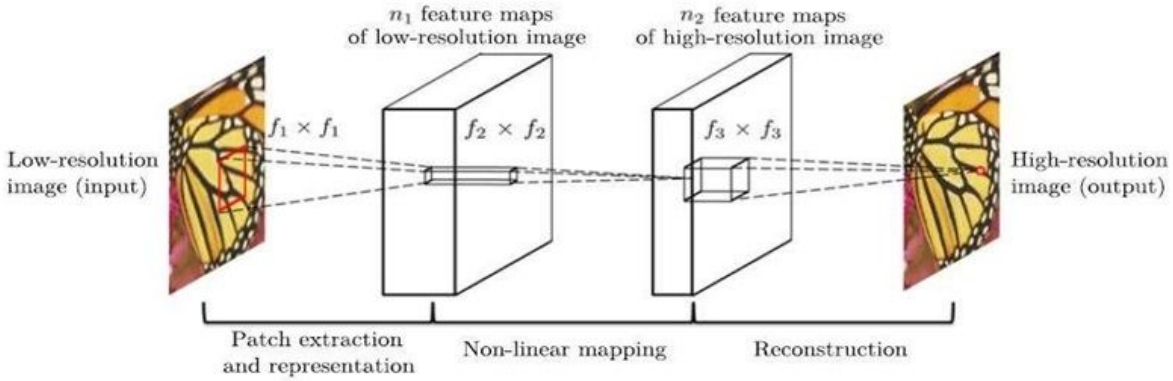
- Maps each of the n_1-dimensional vectors into an n_2-dimensional one.
- This layer is expressed as a function $F_{2,}$ where
  - § $F_2(Y) = \max(0, W_2 * F_1(Y) + B_2)$ (5)

- It is possible to add more convolutional layers to this structure, but in perspective, increases the training time.

4) Reconstruction

- The predicted overlapping high-resolution patches are often averaged to produce the final full image.
- This convolutional layer is defined as
  - § $F(Y) = W_3 * F_2(Y) + B_3$

Structure of the network



Terms Used in the Formulations

- W_1 = Corresponds to the n_1 filters of size $c * f_1 * f_1$,
  - Where c is number of channels and f_1 is the spatial size of the filter.
- B_1 = An n_1-dimensional vector, whose each element is associated with a filter.
- W_2 = n_2 filters of size $n_1 * f_2 * f_2$.
- B_2 = n_2 dimensional vector.
- W_3 = Corresponds to c filters of size $n_2 * f_3 * f_3$
- B_3 = c- dimensional vector.

Learning Process

- Network parameters are $\Theta = \{W_1, W_2, W_3, B_1, B_2, B_3\}$.
- Estimation of network parameters can be achieved through minimizing the loss between reconstructed images and the corresponding original high-resolution images. This is done by taking the Mean Squared Error (MSE).

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||F(\mathbf{Y}_i; \Theta) - \mathbf{X}_i||^2$$

- Using MSE as a loss function, favours high PSNR (Peak Signal to Noise Ratio).
- The loss is minimized by using stochastic gradient descent with regular back- propagation algorithm.

$$\Delta_{i+1} = 0.9 \cdot \Delta_i + \eta \cdot \frac{\partial L}{\partial W_i^\ell}, \quad W_{i+1}^\ell = W_i^\ell + \Delta_{i+1}$$

Generative Adversarial Network

(cledig,ltheis,fhuszar,jcaballero,aacostadiaz,aaitken,atejani,jtotz,zehanw,wshi)

- It is a type of Single-Image Super Resolution.
- Despite the breakthroughs in accuracy and speed of single image super-resolution using faster and deeper convolutional
- Neural networks, one central problem remains largely unsolved: how do we recover the finer texture details?
- It is the first framework capable of inferring photo-realistic natural images for 4X up scaling factors.
- A perceptual loss function is introduced which consists of an adversarial loss and a content loss.
- Adversarial loss pushes our solution to the natural image manifold using a discriminator network that is trained to differentiate between the super-resolved images and original photo-realistic images.
- Content loss is motivated by perceptual similarity instead of similarity in pixel space.
- Deep residual network is able to recover photo-realistic textures from heavily down sampled images on public benchmarks.

- A super-resolution generative adversarial network (SRGAN) is proposed for which we employ a deep residual network (ResNet) with skip-connection and diverge from MSE as the sole optimization target.
- Perceptual loss using high-level feature maps of the VGG network combined with a discriminator encourages solutions perceptually hard to distinguish from the HR reference images.

★ Super resolution using Deep Convolutional Network has been used for our BTP project.

5. Results

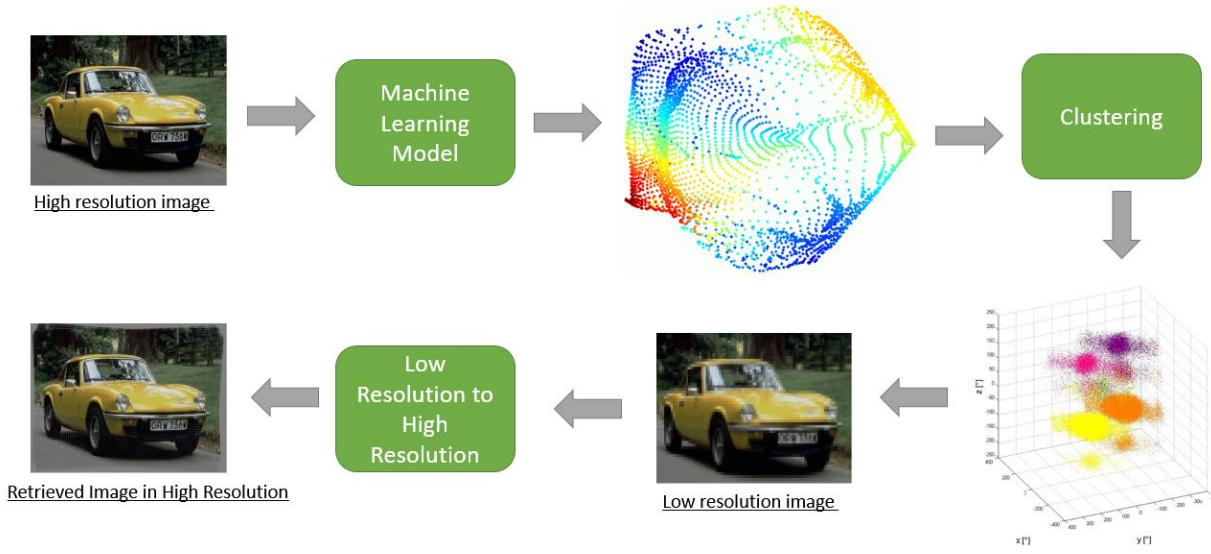
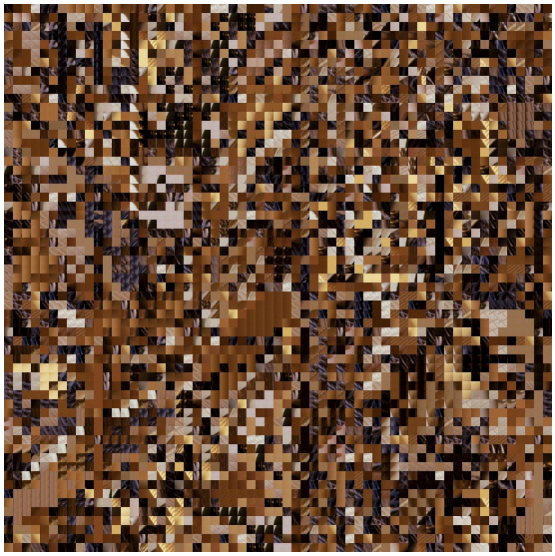


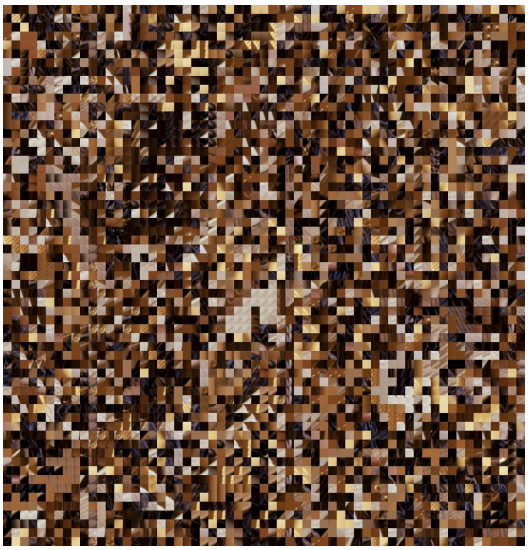
Fig 5.1: The itinerary of image compression algorithm

5.1 Image compression Results:

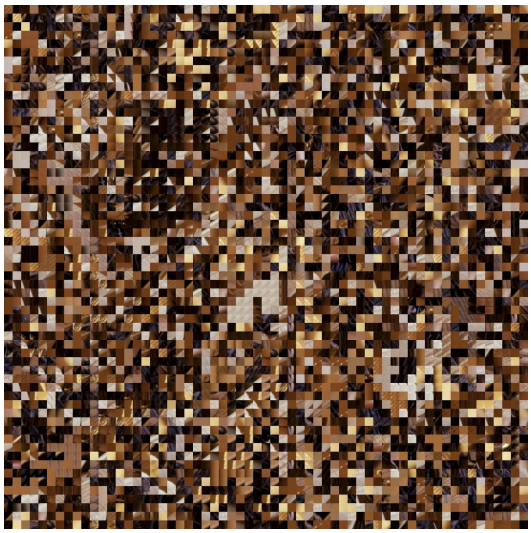
The given image compression algorithm has the following results obtained after we incorporate ISTA into our model for reconstruction of lost parts of images and their approximation by using a kernel matrix R as described in that section. We also used naive means to first test our model for multiple image block sizes. We have given the following result for 32X32 block size compression. The naive results were given in the image compression and are shown below also for a more direct comparison. The each block when passed from the convolutional neural network requires a input of size 229 X 229 in the vgg16 model. So we had to upscale each image to the desired network size. Using a new model without weight would mean to train the model from scratch without sufficient training data would mean that the image is also required that train the model for the compression task.



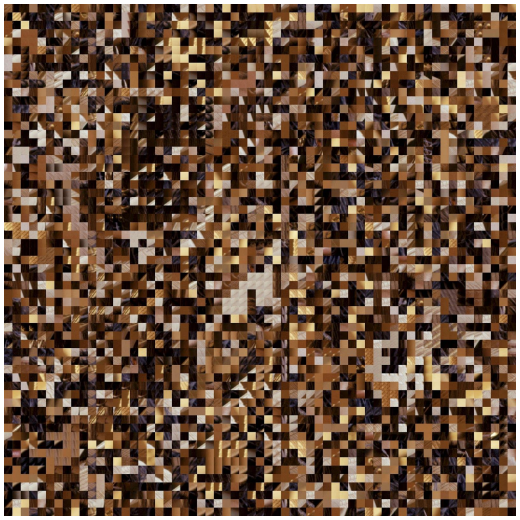
sigma=0.1



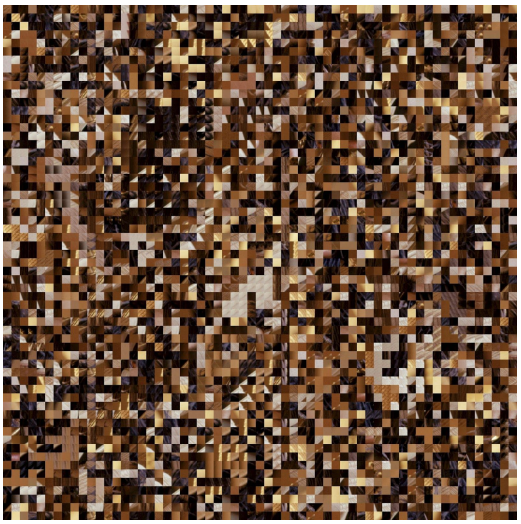
sigma=0.3



sigma=0.5

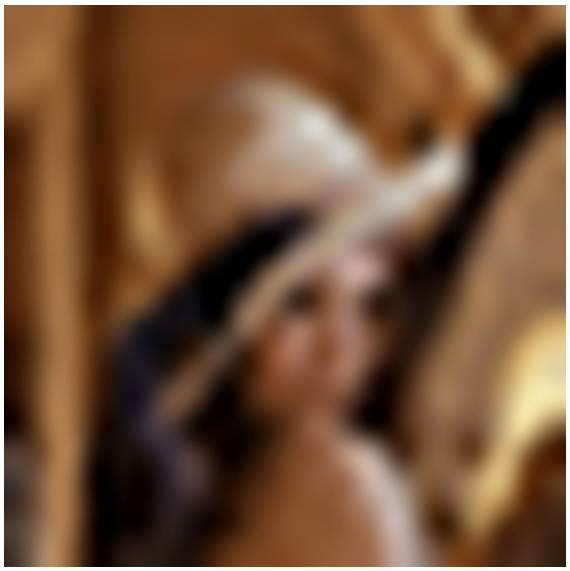


sigma=0.7

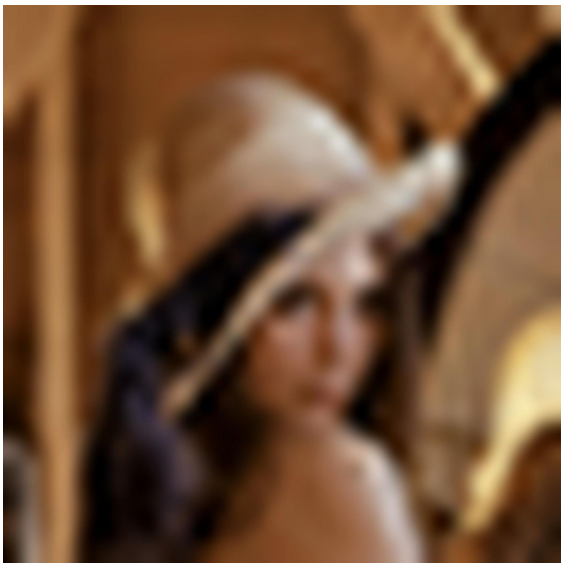


sigma=1

We found out that the nearest neighbour losses the temporal information while we are encoding the information while compression and the clustering method of approximation for continuous contours for images even as large as dimensions of (1950X1950) fail miserably to be reproduced.We had couple of choices from here either we thinking about overlapping the blocks or we think about reducing the image blocks size as the information encoded was pretty much redundant. We thought about going in other direction and trying to smooth out the image using a another sparse coding algorithm called as ISTA on top of our model to give satisfactory results.



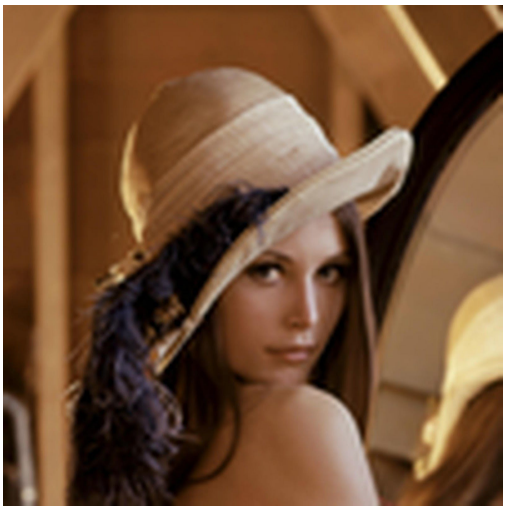
sigma =0.1



sigma=0.3

sigma =0.5

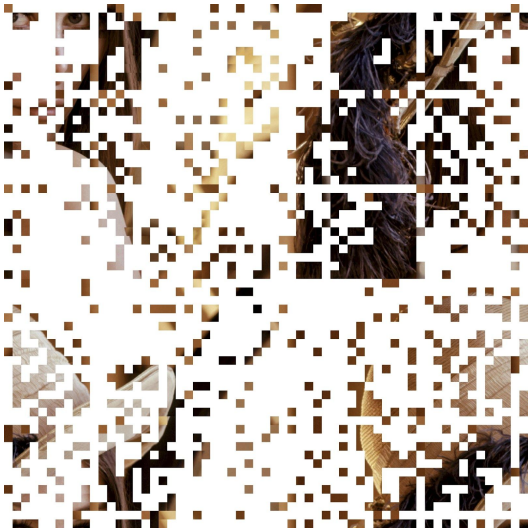
sigma=0.7



sigma=1

Figure: They are the result of images received for all the different values of simga after the compression algorithm runs and places the image blocks into right position. for block sizes of 32X32.

We also give the cluster maps for the kind of clustering blocks selected out of our algorithm as follows.



sigma =0.3

sigma=0.5

sigma=0.7

sigma=1

5.2 DNA Encoding Results

We have successfully encoded files to a sequence of A,C,G,T molecules, ie. a .DNA file. The encoding-decoding has been tested and verified for multiple files in various formats.

We have also been successfully able to deploy the application on a web portal.

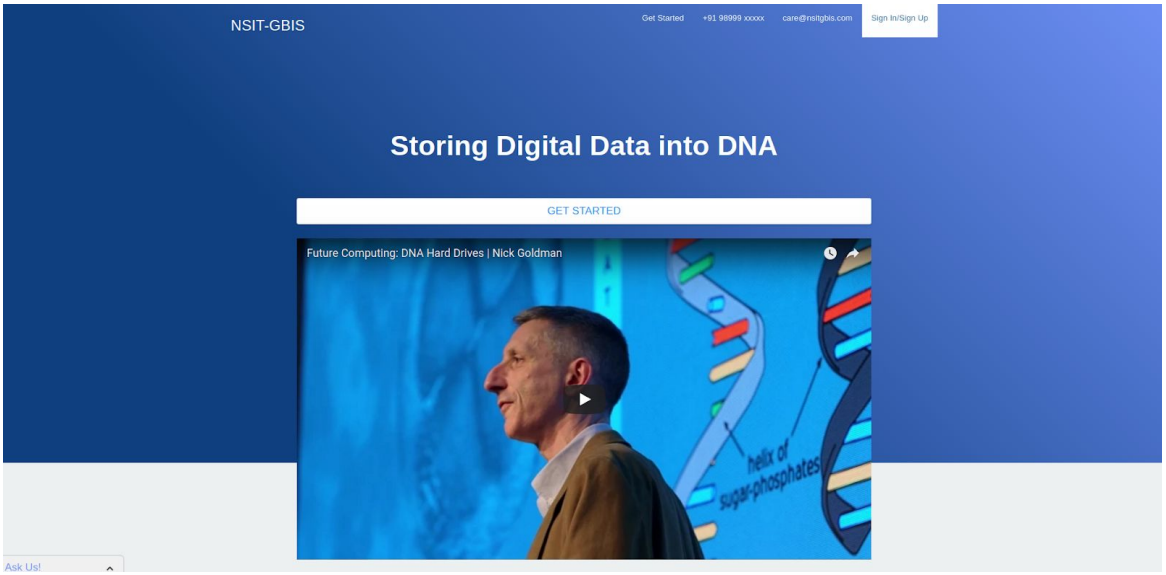


Fig 5.2: Client Side of DNA file encoder

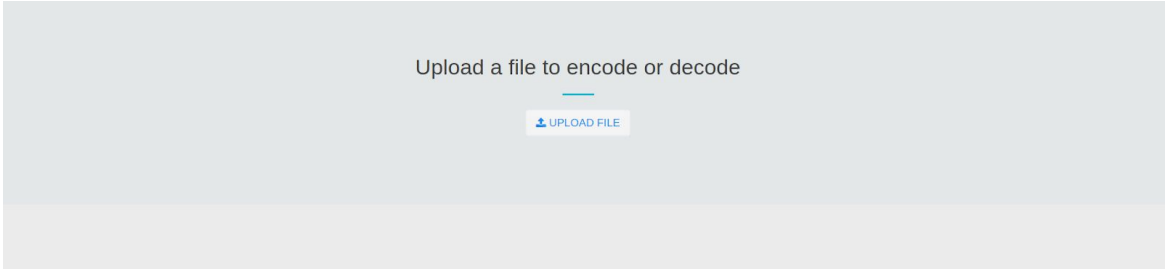


Fig 5.3: File uploadation by Client

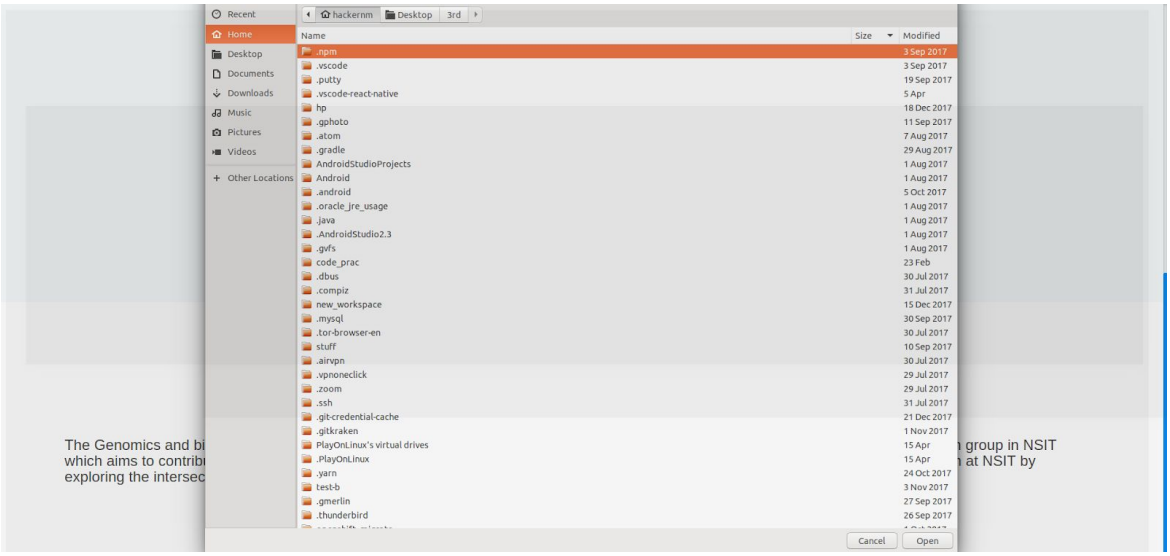


Fig 5.4: File selection by Client

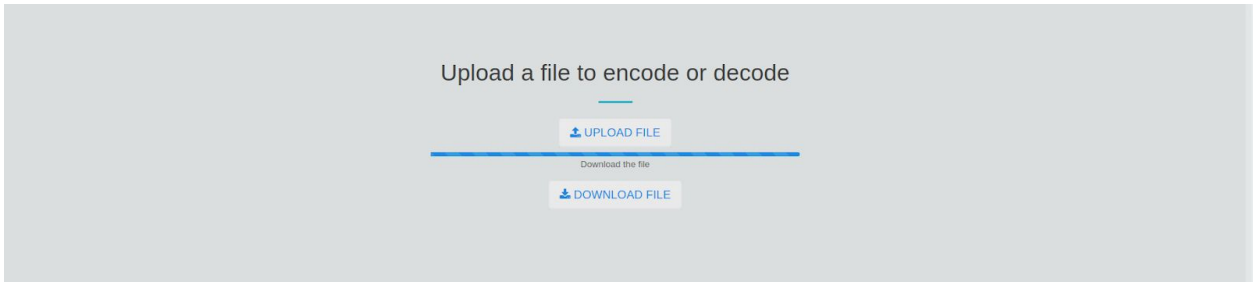


Fig 5.5: DNA file download

Fig 5.6: DNA file encoded

6. Future Work

In future we aim to take advantage of techniques used in GAN and further upscale the process of image retrieval super resolution. The process of single super resolution will help is get more lossy while we compress our image so that we will still be able to retrieve most of the information by smart guess work in the form of machine learning generative models which we use. The step for image compression using machine learning techniques has gained quite a popularity in recent trends. The workshop in Computer Vision and Pattern Recognition(CVPR) 2018 has a dedicated focus on the problem of faster and better image compression techniques using machine learning and deep learning. We believe that the compression when complemented by super resolution is believed to give promising results when studied extensively to solve problem of image compression. The idea proposed with respect to images in this work are novel the means to see summarization as a compression technique is a unique contribution of this work. As the field of image corpus summarization grows independently if it can be linked to other problems such as compression the areas will have new brave ideas for us to experiment. Coming over to the problem in hand which required a system to make open source software publicly available to anyone who wishes to get his/ her data synthesized and store in form of DNA a web app will serve as a asgard one stop place for their tasks. They can get their DNA bindings and give them to companies to get their DNA synthesized in situ. Even when they wish to read their information back they can use the utility after their DNA has been sequenced to decode the information. In the process we found that the number of bases which are being used to synthesize even a medium sized image are tremendous. So by any means if we could compress images in a lossy manner such that only spurious information is lost and rest of the parts of images remain intact but also give us a very humongous return in terms of the cost we save while writing a lossy compressed image into DNA. We also found a way that High resolution images writing of those was not at all possible without the out of world prices that would have to be otherwise paid. This made the whole process totally infeasible in terms of its economic viablity and from standpoint of view of cost issues. This can help the research groups in conducting experiments on encoding algorithms on high resolution images where instead of checking their loss on the absolute images they can check their losses on compressed format and save costs but also check sanity of their work by eventually retrieving a very close high resolution image.

7. References:

1. Church, George M., Yuan Gao, and Sriram Kosuri. "Next-generation digital information storage in DNA." *Science*(2012): 1226355.
2. Goldman, Nick, et al. "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA." *Nature*494.7435 (2013): 77.
3. Bornholt, James, et al. "A DNA-based archival storage system." *ACM SIGOPS Operating Systems Review* 50.2 (2016): 637-649.
4. Burgoyne LA, inventor; Flinders Technologies Pty Ltd, assignee. Solid medium and method for DNA storage. United States patent US 5,496,562. 1996 Mar 5.
5. Moutel, G., Sandrine de Montgolfier, Jean-Paul Meningaud, and Christian Herve. "Bio-libraries and DNA storage: assessment of patient perception of information." *Med. & L.* 20 (2001): 193.
6. Lee, Steven B., Kimberly C. Clabaugh, Brie Silva, Kingsley O. Odigie, Michael D. Coble, Odile Loreille, Melissa Scheible et al. "Assessing a novel room temperature DNA storage medium for forensic biological samples." *Forensic Science International: Genetics* 6, no. 1 (2012): 31-40.
7. Erlich, Yaniv, and Dina Zielinski. "DNA Fountain enables a robust and efficient storage architecture." *Science* 355.6328 (2017): 950-954.
8. Frippiat, C., Zorbo, S., Leonard, D., Marcotte, A., Chaput, M., Aelbrecht, C., & Noel, F. (2011). Evaluation of novel forensic DNA storage methodologies. *Forensic Science International: Genetics*, *5*(5), 386-392.
9. Fan, Jianping, et al. "A novel approach to enable semantic and visual image summarization for exploratory image search." *Proceedings of the 1st ACM international conference on Multimedia information retrieval*. ACM, 2008.
10. Yang, Chunlei, et al. "Image collection summarization via dictionary learning for sparse representation." *Pattern Recognition* 46.3 (2013): 948-961.
11. Shi, Liang, et al. "Context saliency based image summarization." *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*. IEEE, 2009.
12. Marchesotti, Luca, and Claudio Cifarelli. "Image summarization by a learning approach." U.S. Patent No. 8,537,409. 17 Sep. 2013.
13. Xu, Hao, et al. "Hybrid image summarization." *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011.
14. Tipping, Michael E., and Christopher M. Bishop. "Bayesian image super-resolution." *Advances in neural information processing systems*. 2003.
15. http://misl.cs.washington.edu/
16. Dong, Chao, et al. "Image super-resolution using deep convolutional networks." *IEEE transactions on pattern analysis and machine intelligence* 38.2 (2016): 295-307.