VISVESVARAYA TECHNOLOGICALUNIVERSITY

"Jnana Sangama", Belgaum-590018.

A project Phase-I report on

"NIST Handwritten Pattern Recognition"

Submitted in partial fulfillment for the requirements of the VII Semester degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

For the Academic Year
2019-2020

Name:	USN:
Anurag G	1DB16CS023
C Sagar Patil	1DB16CS042
Harsha J K	1DB16CS056
Deepak S D	1DB17CS403

Under the Guidance Of
MRS. HEMALATHA M
Asst. Professor,

Dept. of CSE

Department of Computer Science and Engineering

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Mysore Road, Bengaluru - 560074.

2019-2020

CONTENTS

SL.NO	CHAPTERS	PG.NO
1.	INTRODUCTION	1
2.	SCOPE OF PROJECT WORK	3
3.	LITERATURE SURVEY	5
4.	METHODOLOGY	9
5.	PROJECT WORK DETAILS	13
6.	CONCLUSION	19
7.	REFERENCES	20

Chapter 1

INTRODUCTION

Image classification is one of the core problems of computer vision, image classification refers to the task of extracting information classes from a multiband raster image. One of the important applications of image classification is the optical character recognition (OCR). OCR is the electronic conversion of scanned images of handwritten text into machine encoded text. In Optical character recognition, an algorithm will be trained on a dataset of known character in order to learn how to classify characters included in test set accurately. A variety of algorithm has been developed for classifying letter and digits from last decades. In the field of optical character recognition there are three methods: template matching, feature extraction and classification, third one is the deep learning method. Template matching is the very first method used in the field of recognition through it is relatively simple than other method but the problem of some illegally written text such as difference of diameters, discontinuity of text or rotation, the recognition accuracy is not satisfactory. In feature extraction and classification, the key point is feature extraction algorithm, it determines the recognition rate and furthermore, this part need researchers to try different kind of feature manually, so a large number of experiment needed in this method. In the early stage of OCR, template matching and structural analysis are used popularly.

The late 80s in order to take advantage of massive samples, classification methods such as artificial neural network had been utilized popularly for recognition problems. In the last decade, machine learning methods such as support vector machines (SVMs) have been applied for pattern recognition problems. Neural networks (NNs) are another solution to resolve recognition problems. In this a large number of handwritten letters/digits known as training set are fed into the algorithm in order to infer rules automatically for handwritten character recognition.

Our system comprises of two parts:

1) An Android application: This is the frontend of our system. The android application helps the user to click a picture of text which is to be recognized, using their smart phone camera. This picture is passed on to a python script running on a server which further processes this image to extract the relevant information.

2) A server: This is the backend of our system. This server is a computer which is capable of executing a python script. It is needed because an android smart phone does not have the computation power required for running neural networks and performing image processing operations. Also the use of server for performing computationally intensive tasks enables users of older smart phones to make use of our system. We used the Convolutional Neural Network Model in our system. We used the publicly available NIST Dataset which contains samples of handwritten characters from thousands of writers. The neural network model which we have used is Convolutional Neural Network. CNN's are State-of-Art neural networks which have huge applications is field of Computer Vision. The neural network model was trained using Tensorflow which is an open source library used for Machine learning applications. OpenCV was used to perform various image processing operations like segmentation, thresholding and Morphological Operations. OpenCV is an open source library which is used for Image processing.

Chapter 2

SCOPE OF PROJECT WORK

The Handwritten Written recognition(HWR) can be extended in the following directions through its implementation:

1. Font Independent HWR: An Optical Character Recognition system could be developed by considering the multiple font style in use. Our approach is very much useful for the font independent case. Because, for font or character size, it finds the string and the strings are parsed to recognize the character. Once character is identified, the corresponding character could be ejected through an efficient editor. Efforts have been taken to develop a compatible editor.

2. HWR for Languages: All languages require development of an HWR for printed characters, and for handwritten characters, HWR has to be developed for all languages. Of course, HWR for printed characters are easy when compared to the handwritten cursive scripts. Even for the printed document recognition, an OCR should be able to perform the all features besides character recognition, such as spell check, sentence and grammar check, also an editor with key board encoding and font encoding is required. With this approach the printed and handwritten characters are recognizable easily with less effort and more accuracy. A module for Skew correction and line separation, word and character separation along with an editor with spell checker and grammar checker could be designed for developing a complete HWR. Further, with a little fine tuning on the modules, such as, skew correction and line separation, word and character separation, a complete HWR could be designed for handwritten scripts of any language for that matter. It is proposed to apply the approach to all manuscripts recognition of South-Indian languages. Since some of the characters in some of the languages have similar characters viz, Tamil and Malayalam have similar features among few characters, and Telugu and Kannada have similarity among most of the characters, our approach could be applied for these languages and could be extended to all other languages.

3. Cursive Characters HWR: There is heavy demand for an HWR system which recognizes cursive scripts and manuscripts. This actually avoids keyboard typing and font encoding too.

4. Language Converter through HWR: Once a complete HWR has been developed for two languages with font encoding, spell checker and grammatical sentence check, then a converter could be implemented to convert sentences from one language to another through a

transliteration and translation scheme.

5. A Bilingual or multi-lingual script HWR: It is basically necessary to develop an HWR for multi lingual script for a country where more than 10 languages are in use officially. For example, if there is a document where two language scripts are available, then one need not separate two scripts into two different files and feed them into two HWR's. Using our approach one could develop an HWR for two languages, Tamil and English in the same document. These scripts could be edited later with an editor too.

6. Speech recognition from HWR: The most required application today is Speech recognition. The recognized Printed or Handwritten character could be recorded and through a voice synthesizer speech output could be generated. This would help the blind to send and receive information.

Chapter 3

LITERATURE SURVEY

An early notable attempt in the area of character recognition research is by Grimsdale in 1959. The origin of a great deal of research work in the early sixties was based on an approach known as analysis-by-synthesis method suggested by Eden in 1968. The great importance of Eden's work was that he formally proved that all handwritten characters are formed by a finite number of schematic features, a point that was implicitly included in previous works. This notion was later used in all methods in syntactic (structural) approaches of character recognition.

Survey of Various preprocessing techniques:

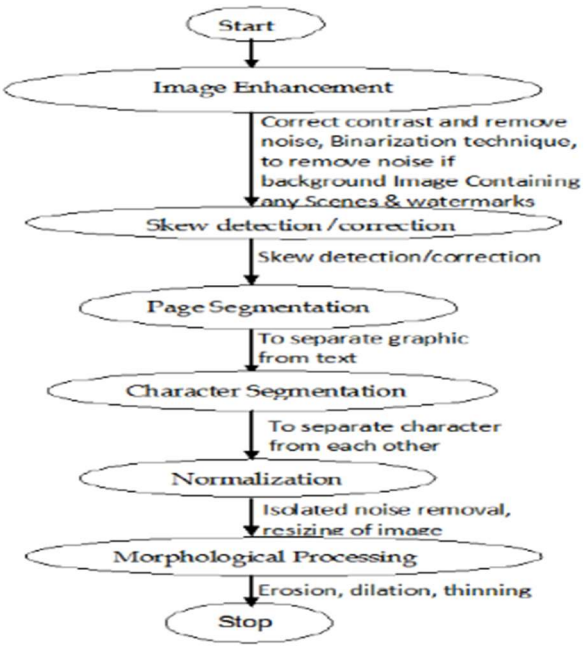


Fig: Flowchart illustrating preprocessing techniques

Pre-processing techniques involved in the character recognition with different kind of images ranges from a simple handwritten form based documents and documents containing colored and complex background and varied intensities. In this, different preprocessing techniques like skew detection and correction, image enhancement techniques of contrast stretching, binarization, noise removal techniques, normalization and segmentation, morphological processing techniques are discussed. It was concluded that using a single technique for preprocessing, we

can't completely process the image. However, even after applying all the said techniques might not possible to achieve the full accuracy in a preprocessing system.

Survey based on hidden Markov model:

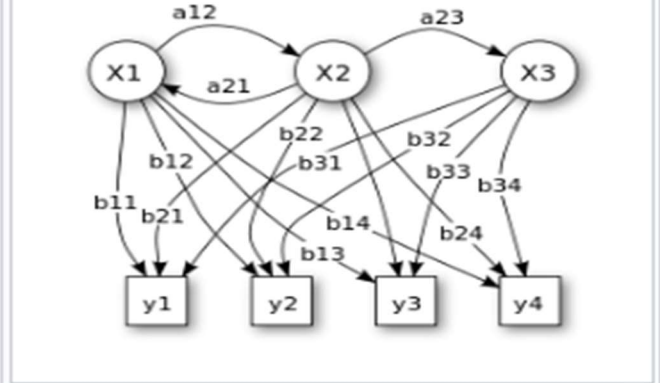
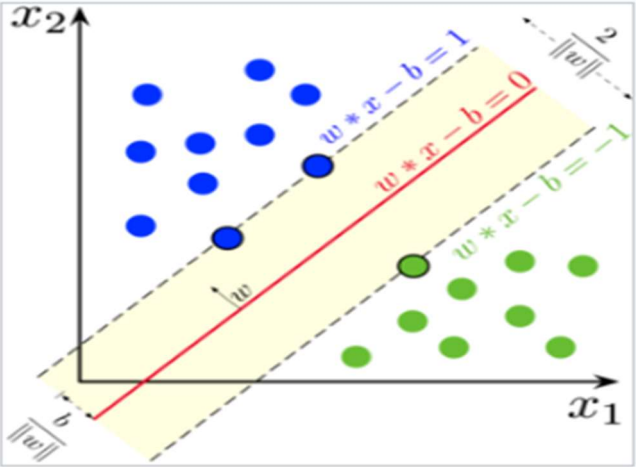


Figure 1. Probabilistic parameters of a hidden Markov model (example)
X — states
y — possible observations
a — state transition probabilities
b — output probabilities

Hidden Markov Models (HMM) use discrete and hybrid modelling techniques. Handwriting recognition experiments using a discrete and two different hybrid approaches, which consist of a discrete and semi-continuous structures, are compared. A segmentation free approach is considered to develop the system. It is found that the recognition rate performance can be improved of a hybrid modelling technique for HMMs, which depends on a neural vector quantizer (hybrid MMI), compared to discrete and hybrid HMMs, based on tired mixture structure (hybrid - TP), which may be caused by a relative small data set.

In simpler Markov Models (like a Markov chain), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters, while in the hidden Markov model, the state is not directly visible, but the output (in the form of data or "token" in the following), dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states.

Survey based on Support Vector Machine (SVM):



Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

A support-vector machine constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalized error of the classifier. Whereas the original problem may be stated in a finite-dimensional space, it often happens that the sets to discriminate are not linearly seperable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products of pairs of input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function **k(x,y)** selected to suit the problem. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vectors is an orthogonal (and thus minimal) set of vectors that defines a hyperplane. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters α_i of images of feature vector $\mathbf{x}_i$ that occur in the data base. With this choice of a hyperplane, the points **x** in the feature space that are mapped into the hyperplane are defined by the relation, $\sum_i \alpha_i k(x_i,x)=$ **constant** .Note that if **k(x,y)** becomes small as **y** grows

further away from **x**, each term in the sum measures the degree of closeness of the test point **x** to the corresponding data base point $\mathbf{x}_i$. In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points **x** mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets that are not convex at all in the original space.

Survey Based on Convolutional Neural Networks (CNN):

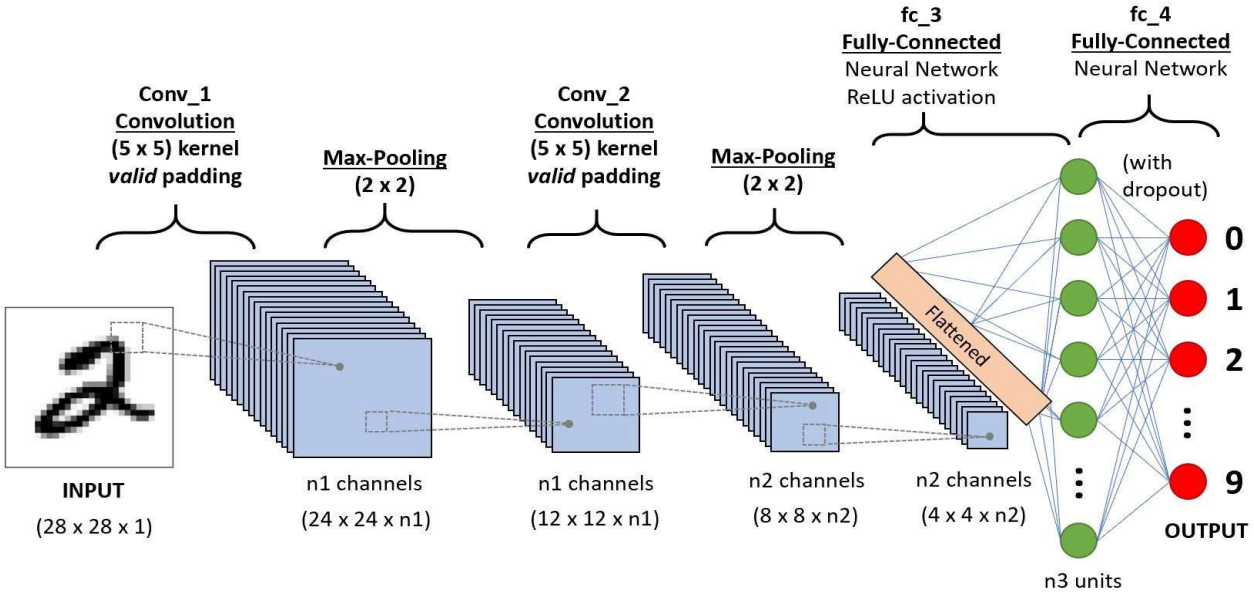


Fig: Architecture of CNN

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

NIST data set is having a huge number of handwritten text data set and it is frequently used for training, testing, and validation of CNN deep model. Researchers have created an efficient model with multiple convolutions, and pooling layers. Which is tested on NIST data set with up to 94% accuracy.

A Multi-Layer Neural Network with one or more number of hidden units is capable of classifying the digits in NIST data set with a less than 6 % error rate on test set.

Chapter 4

METHODOLOGY

There are mainly four operation needed to build a simple ConvNet: Convolution, Non linearity, Pooling or Sub-sampling and classification (fully connected layer). These operations are the basic building block of every convolutional neural network channel is a term which used to refer a certain component of an image. A standard image will have three channels Red, Green, Blue. Each having pixel value in the range of 0 to 255. A gray scale image on the other hand has just one channel. These methods are explained below:

***The convolution step*:** Convolution neural network derive their name from the operator "convolution". The primary purpose of this operator in case of CNNs is to extract features from the input image. This is the first layer in CNN, the input to this layer is a 3D array (32*32*3) of pixel value. Convolution is a mathematical operation to merge two set of information in other our case, first is input image and the second set is the filter.

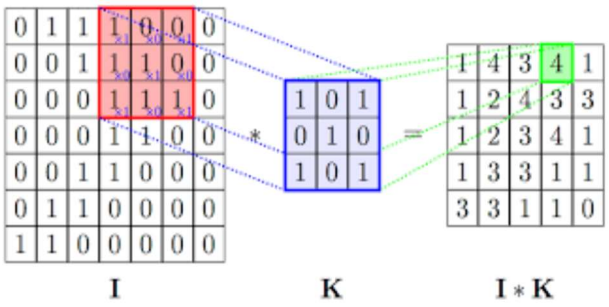


Fig: Convolution process

In figure, on the left side we have input image, convolution filter is located in the centre which is also called kernel, detector or feature whereas on the right side of figure 2 we have the output of the convolution process called activation map also called feature map, convolved feature. We perform convolution operation by sliding this filter over the input image. At every location we do an element wise matrix multiplication and sum the result, this resultant matrix is called Convolved feature. CNN learn the value of these filters on its own during training process.

Non linearity (ReLu): This is the addition operation called ReLu has been used after every convolution operation. ReLu is called rectified linear unit and is a non linear operation. It is given by: Output = Max(zero, input) as indicated in figure. ReLu is an element wise operation and replaces all negative pixel values in the feature map by zero.

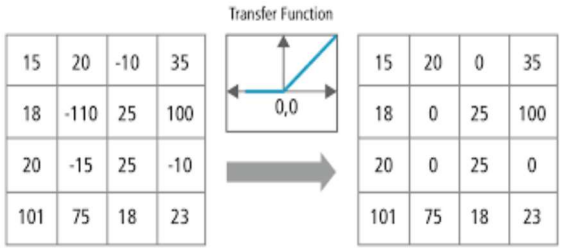


Fig: ReLu Operation

Pooling Step: Pooling (also called sub-sampling or down-sampling) it reduces the dimensionality of each feature map but retain the most important information. This layer makes the input representation smaller and more manageable. It also reduces the number of parameters and computations in the network and controlling from over fitting. The output of pooling layer acts as an input to the fully connected layer which is the next layer in the CNN after this. Figure shows the pooling operation in CNN architecture.

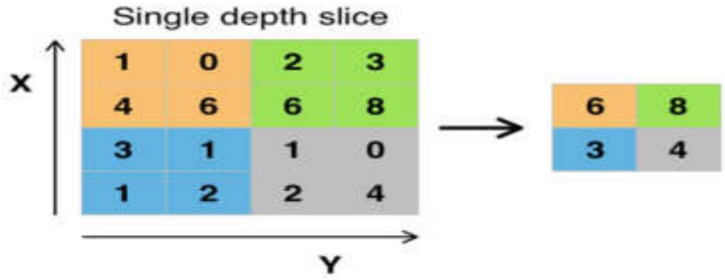


Fig: Pooling process

Fully connected layer: The fully connected layer is a traditional multilayer perceptron, it uses softmax activation function in the output layer. This term implies that every neuron in the previous layer is connected to every neuron in the next layer. The output from convolution layer and pooling layer represent high level features of image. The function of fully connected layer is to classifying these features of input image into various classes based on the training dataset. The sum of output probabilities from fully connected layer is 1. This is ensured by using softmax as the activation function in the output layer of fully connected layer.

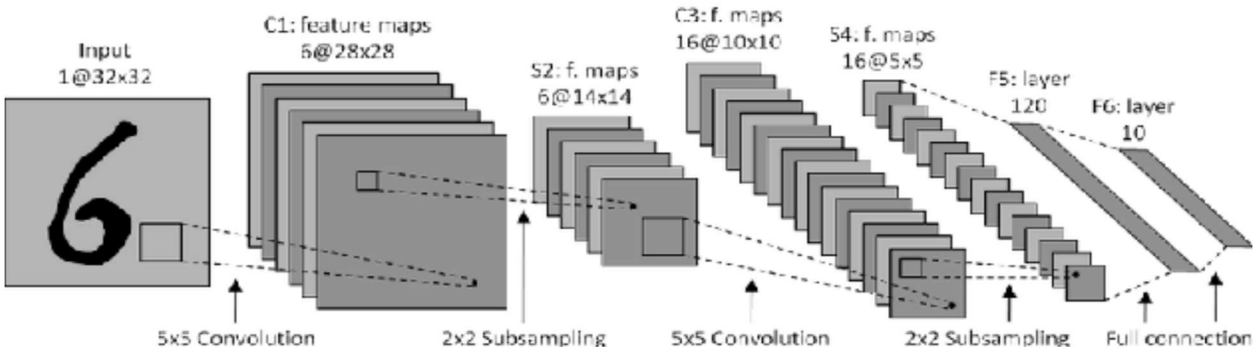


Fig: Fully connected layer

Following steps are involved in pre-processing the images before feeding it to the above trained CNN model:

1) **Pre-processing**:

This is the first step performed in image processing. In this step the noise from the image is removed by using median filtering. Median filtering is one of the most widely used noise reduction technique. This is because in median filtering the edges in image are preserved while the noise is still removed.

2) **Conversion to Gray-Scale**:

After the pre-processing step, the image is converted into grayscale. Conversion into grayscale is necessary because different writers use pens of different colors with varying intensities. Also working on grayscale images reduces the overall complexity of the system.

3) **Thresholding**:

When an image is converted into grayscale , the handwritten text is darker as compared to its background. With the help of thresholding we can seperate the darker regions of the image from the lighter regions. Thus because of thresholding we can seperate the handwritten text from its background.

4) **Image Segmentation**:

A user can write text in the form of lines. Thus the thresholded image is first segmented into individual lines .Then each individual line is segmented into individual words. Finally each word is segmented into individual characters. Segmentation of image into lines is carried out using Horizontal projection method. First the thresholded image is inverted so that background becomes

foreground and vice-versa. Now the image is scanned from top to bottom. While scanning, the sum of pixels in each row of image is calculated. The sum of pixels will be zero if all the pixels in one particular row are black. The sum will be non-zero if some white pixels are present in a row. After this a horizontal histogram is plotted in which the X-axis represents the Y-coordinate of image (Starting from Top to Bottom) and the Y-axis represents the sum of pixels in the row corresponding to the Y-coordinate. The horizontal histogram can plot using MatPlotLib. The points marked in red in histogram are the points corresponding to the rows where sum of pixels are zero. After identifying all such rows, we can easily segment handwritten text into lines at these points. Now once the image is segmented into lines, each line must be further segmented into individual words. Segmentation of a line into words can be performed using the Vertical projection method. For segmenting line into words, we can make use of the fact that the spacing between two words is larger than the spacing between two characters. To segment a single line into individual words, the image is scanned from left to right and sum of pixels in each column is calculated. A vertical histogram is plotted in which the X-axis represents the X-coordinates of image and Y-axis represents the sum of pixels in each column. The points which are marked as red in the histogram are the points corresponding to the columns where sum of pixels is zero. The region where the sum of pixels is zero is wider when it is a region separating two words as compared to the region which is separating two characters. After segmenting a line into words, each word can be separated into individual character using similar technique as explained earlier. Now these individual characters are given to the pre-trained neural network model and predictions are obtained. Using this the final predicted text is sent back as a response to the user.

After the above processes the processed image is fed into the trained CNN model which takes the optical image as an input and recognize them and gives the output of the recognized texts.

Chapter 5

PROJECT WORK DETAILS

History of Python

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989.Van Rossum shouldered sole responsibility for the project, as the lead developer, until July 12, 2018, when he announced his "permanent vacation" from his responsibilities as Python's *Benevolent Dictator For Life*, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January, 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project.

Python 2.0 was released on 16 October 2000 with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.

Over view of python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

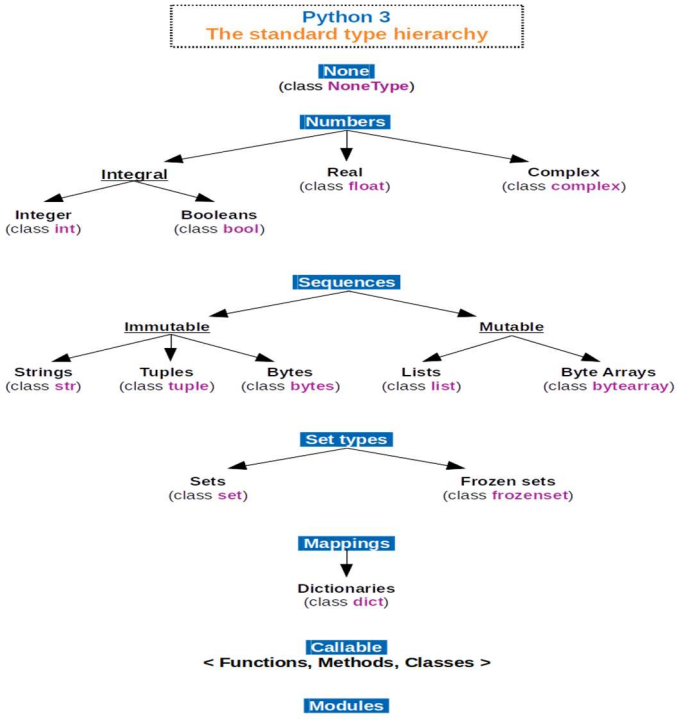


Fig: Overview of Hierarchy in python

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Features and philosophy

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that

implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document *The Zen of Python* (*PEP 20*), which includes aphorisms such as:

1) Beautiful is better than ugly.

2) Explicit is better than implicit.

3) Simple is better than complex.

4) Complex is better than complicated.

5) Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is *not* considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. CPython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name—a tribute to the British comedy group Monty Python—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar.

A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms

well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as *Pythonistas.*

Python modules Overview

Modular programming refers to the process of breaking a large, unwieldy programming task into separate, smaller, more manageable subtasks or **modules**. Individual modules can then be cobbled together like building blocks to create a larger application.

There are several advantages to **modularizing** code in a large application:

Simplicity: Rather than focusing on the entire problem at hand, a module typically focuses on one relatively small portion of the problem. If you're working on a single module, you'll have a smaller problem domain to wrap your head around. This makes development easier and less error-prone.

Maintainability: Modules are typically designed so that they enforce logical boundaries between different problem domains. If modules are written in a way that minimizes interdependency, there is decreased likelihood that modifications to a single module will have an impact on other parts of the program. (You may even be able to make changes to a module without having any knowledge of the application outside that module.) This makes it more viable for a team of many programmers to work collaboratively on a large application.

Reusability: Functionality defined in a single module can be easily reused (through an appropriately defined interface) by other parts of the application. This eliminates the need to recreate duplicate code.

Scoping: Modules typically define a separate **namespace**, which helps avoid collisions between identifiers in different areas of a program. (One of the tenets in the Zen of Python is *Namespaces are one honking great idea—let's do more of those!*)

Functions, **modules** and **packages** are all constructs in Python that promote code modularization.

Important Python modules

OpenCV:

OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license. OpenCV supports the deep learning frameworks such as TensorFlow.

OpenCV can be used to perform various image processing operations like segmentation, thresholding and Morphological Operations. It's a open source library used for Image processing.

NumPy:

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

→a powerful N-dimensional array object

→sophisticated (broadcasting) functions

→tools for integrating C/C++ and Fortran code

→useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. We need numpy to work with the data arrays.

Tensorflow:

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. Tensorflow will be the key component, as we are using it to create the AI.

Math:

The math module is a standard module in Python and is always available. To use mathematical functions under this module, you have to import the module using import math.It gives access to the underlying C library functions. Math provides some neat helper functions.

Matplotlib:

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

Other packages:

Some of other packages used in the python which will be helpful to build models are sys, time ,datetime ,os. We need sys to read in the arguments. We will use that to specify the amount of training cycles via the command line. time and datetime are used for logging and os is used to set the tensorflow debug level.

CONCLUSION

An overview of convolutional neural network and working of all layers is presented in accordance with other techniques and models. And the above survey of different models concludes that the different parameters like dropout, optimizer, learning rates greatly influence the efficiency of the CNN architectures. Preprocessing of dataset also consider a major factor behind the accuracy of the models. Training of network requires a supportive hardware to implement large dataset efficiently. In some language the characters are differ by only a single dot for that characters it is necessary to train the network with large amount of dataset so that the network easily recognize the character for that there is a requirement of high memory and high processing speed to achieve a efficient network. The articles provided in the literature survey contributes different networks with different tuning of parameters on different types of dataset which help in achieving efficient network in future that can even recognize a whole sentence of different handwritten languages.

REFERENCES

- **Validation of Random Dataset Using an Efficient CNN Model Trained on MNIST Handwritten Dataset**
- **published in:** 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)
- **Date Added to IEEE *Xplore*:** 13 May 2019
- **Classification Algorithms for Determining Handwritten Digit**
- **Handwritten Digits Recognition Using Convolution Neural Networks**