Received June 21, 2021, accepted July 5, 2021, date of publication July 13, 2021, date of current version July 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3096823

uTHCD: A New Benchmarking for Tamil Handwritten OCR

NOUSHATH SHAFFI AND FAIZAL HAJAMOHIDEEN
Department of Information Technology, University of Technology and Applied Sciences, Suhar PC 311, Oman
Corresponding author: Noushath Shaffi (noushath.mys@gmail.com)

ABSTRACT The robustness of a typical Handwritten character recognition system relies on the availability of comprehensive supervised data samples. There has been considerable work reported in the literature about creating the database for several Indic scripts, but the Tamil script has only one standardized database up to date. This paper presents the work done to create an exhaustive and extensive unconstrained Tamil Handwritten Character Database (uTHCD). The samples were generated from around 850 native Tamil volunteers including school-going kids, homemakers, university students, and faculty. The database consists of about 91000 samples with nearly 600 samples in each of 156 classes. This isolated character database is made publicly available as raw images and Hierarchical Data File (HDF) compressed file. The paper also presents several possible use cases of the proposed uTHCD database using Convolutional Neural Networks (CNN) to classify handwritten Tamil characters. Several experiments demonstrate that training on the proposed database helps traditional and contemporary classifiers perform on par or better than the existing dataset when tested with unseen data. With this database, we expect to set a new benchmark in Tamil handwritten character recognition and serve as a launchpad for developing robust language technologies for the Tamil script.

INDEX TERMS Convolutional neural network (CNN), document analysis, handwriting recognition, Indic scripts, optical character recognition, Tamil handwritten character database.

NOMENCLATURE

Acronyms

CM	Confusion Matrix
CNN	Convolutional Neural Network
ePCP	enhanced Piece-wise Covering by Parallelogram
FPR	False Positive Rate
GT	Ground Truth
HDF	Hierarchical Data Format
HPL	HPL-iso-tamil-char Dataset
KNN	K-Nearest Neighbour
OCR	Optical Character Recognition
PCP	Piece-wise Covering by Parallelogram
RF	Random Forest
SOTA	State-of-the-Art
SVM	Support Vector Machine
TPR	True Positive Rate

The associate editor coordinating the review of this manuscript and approving it for publication was Gustavo Olague.

I. INTRODUCTION

Tamils or Tamilians is one of the world's oldest surviving ethnolinguistic groups with a demographic population currently estimated to be around 76 million with a history of this language dating back over 2000 years [1], [2]. Out of 22 official languages of India, it is the only language that has been considered an official language outside of India, as in Srilanka and Singapore [3]. In addition, scripts of several minority Indian languages such as Baduga, Irula, Saurashtra, and Paniya are written in Tamil although its spoken version is entirely different from that of Tamil [3].

Even though a sizeable population around the globe uses the Tamil language as one of the official or as borrowed scripting languages, research in Tamil handwriting recognition in several application domains has not reached its complete maturity [4]–[6]. Tamil Handwritten character recognition is one such challenging research topic for close to 4 decades now [7], [8]. Despite this, it continues to offer many challenges which keeps the research community active even till date [9], [10]. The OCR of printed documents has been considered a problem solved par human accuracy or even better [11]–[13]. However, the OCR of Tamil handwritten document poses critical challenges (for ex: overlapping character patterns, complex character formation, diverse writing styles even within the writing of a single user, etc.) which necessitate rectifiable steps before being deployed in real time [14].

The research in this field is equally instigated by many application-oriented domains such as document structure analysis and segmentation, writer identification in forensic intelligence, digitization of legal and legacy documents [5], [12], [15]–[17]. Moreover, since Tamil language is one of the official languages in India, Srilanka, and Singapore, the availability of a robust Tamil handwritten OCR would pave the way for the automation of a large swath of official documents.

Extensive work relating to the Indic script has been reported in this field of handwriting recognition which is evident through numerous publications [18]. This was mainly possible due to the availability of standardized databases of many Indic scripts [19]–[29]. There are about 19 standardized handwritten data sets available for Indic scripts such as Devanagari, Bengali, Telugu, Oriya, Tamil etc [18].

The recent survey of handwritten Indic OCR technologies [18] reveals that out of 19 databases, only a solitary standardized database [19], [30] is available for Tamil script developed by HP Labs, India.[1] This database is publicly available and was created using online pen coordinates by applying a simple piece-wise interpolation process. Most of the work reported in Tamil handwriting recognition has relied mainly on this database [9], [10]. However, for more objective study and to evaluate the true efficacy of state-of-the-art(SOTA) algorithms, having multiple standardized publicly available datasets exhibiting a variety of inherent writing variations can only add more value to the resulting experimental analysis.

Unlike the HPL database [19], [30], the uTHCD database is a unified collection of samples collected from both offline online modes. In the offline mode, samples were collected by asking volunteers to write on a physical form without any restriction on the writing tool (pen, pencil, gel pen, sketch pen, etc.). In the online mode, volunteers were asked to write samples using a digital writing pad on a digital form (a predefined grid image). In both processes, the samples written by volunteers are extracted and stored eventually as image files.

While online samples successfully capture different writing styles, salient features of offline samples pose even more challenges. Specifically, sharp boundaries, discontinuity of stroke occurring due to scanning and several subsequent preprocessing, variable thickness of character stroke due to the usage of different writing tools add more variation to the database. Offline counterpart samples generated through online data (as in the HPL database) do not represent these variations. Moreover, capturing these variations in the database will prove more effective in devoloping robust algorithms. Especially when processing offline documents such as legal and legacy documents, handwritten form conversions and the like.

In this study, we present the development of an exhaustive database with handwritings involving many Tamil writers and its use-cases from OCR perspectives. The prominent contribution of this work can be outlined as below:

1) A succinct presentation with appropriate signposting of the complete process involved in creating standardized Tamil Handwritten character samples.
2) This database is made publicly available to the research community along with ground truth (GT) for rapid prototyping and benchmarking purposes.
3) Several CNN-centric experimentations were conducted to create the benchmark. Furthermore, the popular VGG16 pre-trained ImageNet model was fine-tuned for the classification of Tamil characters and their performance was analyzed.
4) Intending to analyze the benefit of the proposed database, several traditional and SOTA classifiers were trained using the proposed and existing databases.

The main motivations behind carrying out this work are:

1) Non-availability of multiple databases that can pose diverse challenges inherent in a typical offline handwriting OCR.
2) Non-existence of a standardized handwriting database that can capture variations emerging from offline processing of documents.
3) To evaluate objectively several classical machine learning and SOTA deep learning algorithms using the uTHCD database and benchmark the results. This will act as a catalyst in advancing research in Tamil handwriting recognition.

This work is an extension of the previously published paper where we had reported the preliminary work [31] done in creating the uTHCD database.[2]

Organization of the rest of this manuscript is as follows: Section II describes the details about the Tamil script, an overview of existing Tamil handwritten databases, and works reported using them. Section-III presents the systematized process involved in creating this database, such as data collection, preprocessing, verification, and making the database suitable for the experimentation process. In Section-IV, we set the new benchmark for this database through the application of standard CNN algorithms. Section-V presents the comparative study between the proposed dataset against the existing dataset using traditional and contemporary classifiers. Finally, the

[1] This database is officially known as *HPL-iso-Tamil-char*. For brevity, we refer to it as *HPL database* in rest of the paper.

[2] As a sequel, this paper extends in several ways such as broadened data collection process, ground truth preparation, making dataset publicly available, etc are few prominent improvements among several other contributions.

அ	ஆ	இ	ஈ	உ	ஊ	எ	ஏ	ஐ	ஒ	ஓ	ஔ	ஃ	ஜ	ஷ	ஹ	ஸ	க்ஷ
a	ä	i	Ī	u	ü	e	é	ai	o	ò	au	aytam	j	ṣ	h	s	ks
க	ச	ங	ஞ	ட	ண	த	ந	ப	ம	ய	ர	ல	ள	ற	வ	ழ	ன
k	c	ṅ	ṅ	ṫ	ṇ	t	n	p	m	y	r	l	ḷ	r	v	ẓ	ṇ

FIGURE 1. Vowels, consonants, ayutha ezhuthu and grantha letters of tamil script.

க் + அ	க் + ஆ	க் + இ	க் + ஈ	க் + உ	க் + ஊ	க் + எ	க் + ஏ	க் + ஐ	க் + ஒ	க் + ஓ	க் +ஔ
க	கா	கி	கீ	கு	கூ	கெ	கே	கை	கொ	கோ	கௌ

FIGURE 2. Compound character formation for a consonant with set of vowels.

conclusion and future avenues based on this work are presented in Section-VI.

II. BACKGROUND STUDY

A. TAMIL SCRIPT

The Tamil script contains 12 vowels, 18 consonants, and one special character known as *Ayudha Ezhuthu*. Additional five consonants known as Grantha Letters are borrowed from Sanskrit and English to represent words/syllables of north Indian and English origin [2]. Hence the script contains 36 unique basic letters [12 vowels + 18 consonants + 1 Ayudha Ezhuthu + 5 Granthas]. Fig.1 shows these basic characters. The combination of twelve vowels and eighteen consonants gives rise to 216 compound characters resulting in 247 characters [216 + 12 + 18 + 1]. Example formation of compound characters for a consonant is shown in the figure Fig.2. Furthermore, the five grantha letters combine with twelve vowels similarly to form additional 60 compound letters, leading to 312 characters [247 + 5 Granthas + 60 Compound Granthas]. The entire character set comprising 312 characters of the script is shown in Fig.3 [2]. However, the 156 distinct characters (shaded grey) can uniquely represent these 312 characters. This is because, as depicted in Fig.4, two different syllables of the word [Ko-vai] can be represented using five distinct characters (numbers in the figure indicate the underlying class number).

B. EXISTING TAMIL HANDWRITTEN DATABASE

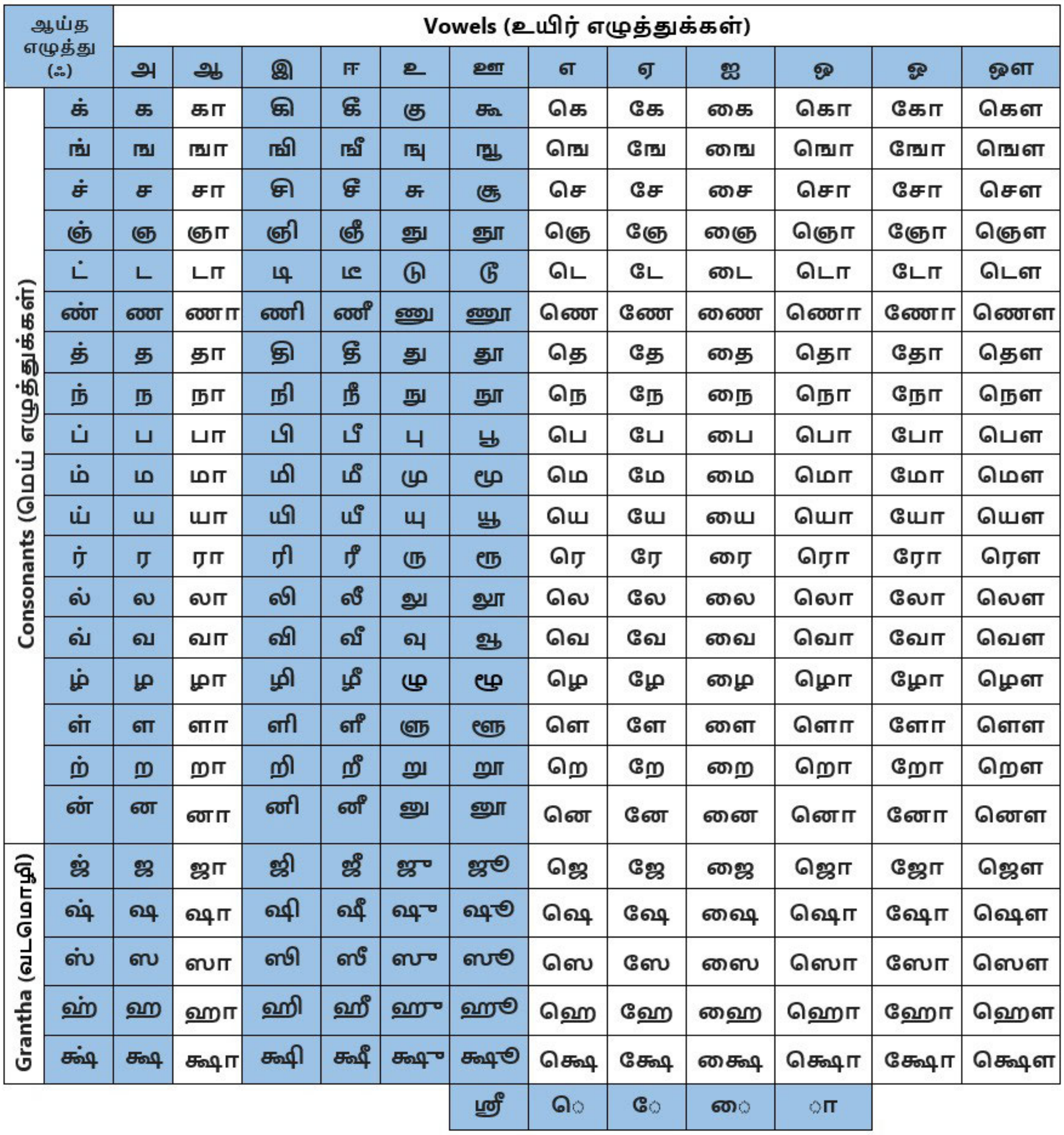
A comprehensive database plays a vital role in the advancement of any research topic [19]. It will facilitate researchers to rapidly evaluate new models and develop robust algorithms, especially when the underlying database is unconstrained and comprehensive. The availability of a comprehensive database can also help to objectively assess the true efficacy/limitations of SOTA algorithms [18]. Developing a robust Tamil OCR system necessitates a large and extensive database [10]. As mentioned earlier, there is only a single standardized database developed by HP labs India, available for the Tamil OCR domain [18].

1) THE HPL INDIA DATABASE: THE HPL-ISO-TAMIL-CHAR DATABASE

This database contains a varying number of isolated character samples in each of 156 distinct character classes. Most of the character classes have around 500 samples, while few classes have about 270 samples. The maximum number of samples in a class is 527, while the minimum is 271. The database consists of two separate repositories for training and testing. There are around 50691 and 26926 samples available respectively for training and testing purposes [19]. The database contains 77617 samples in total from all classes.

The samples were written by native Tamil writers and collected using HP Tablet PCs. As mentioned earlier, images of samples were created through an interpolation process using online pen coordinates with a constant thickening factor and are available for download in a bi-level TIFF format. Fig.5 shows some sample images of this database. More details of this database can be found on the lipitoolkit website [30]. This database is the de-facto standard among the research community for Tamil OCR. One can easily find numerous quality papers in the literature that used this database [32]–[35]. Kavitha and Srimathi [32] have recently used this database to test the efficacy of 9-layer CNN architecture which achieved a training accuracy of 95.16% and test accuracy of 97.7%. However, this database has the following limitations:

1) **Imbalance in the dataset:** Several classes are underrepresented in this collection. The machine learning algorithms learn to predict the dominant classes correctly more often than the underrepresented class [36]. Due to an imbalance in the dataset, algorithms using this collection will have to use different evaluation metrics such as F1 score per class, recall per class, precision per class other than just accuracy. Another option would be to resample the dataset so that the data is balanced either by augmentation or undersampling



ஆய்த எழுத்து (ஃ)		Vowels (உயிர் எழுத்துக்கள்)											
		அ	ஆ	இ	ஈ	உ	ஊ	எ	ஏ	ஐ	ஒ	ஓ	ஔ
Consonants (மெய் எழுத்துக்கள்)	க்	க	கா	கி	கீ	கு	கூ	கெ	கே	கை	கொ	கோ	கௌ
	ங்	ங	ஙா	ஙி	ஙீ	ஙு	ஙூ	ஙெ	ஙே	ஙை	ஙொ	ஙோ	ஙௌ
	ச்	ச	சா	சி	சீ	சு	சூ	செ	சே	சை	சொ	சோ	சௌ
	ஞ்	ஞ	ஞா	ஞி	ஞீ	ஞு	ஞூ	ஞெ	ஞே	ஞை	ஞொ	ஞோ	ஞௌ
	ட்	ட	டா	டி	டீ	டு	டூ	டெ	டே	டை	டொ	டோ	டௌ
	ண்	ண	ணா	ணி	ணீ	ணு	ணூ	ணெ	ணே	ணை	ணொ	ணோ	ணௌ
	த்	த	தா	தி	தீ	து	தூ	தெ	தே	தை	தொ	தோ	தௌ
	ந்	ந	நா	நி	நீ	நு	நூ	நெ	நே	நை	நொ	நோ	நௌ
	ப்	ப	பா	பி	பீ	பு	பூ	பெ	பே	பை	பொ	போ	பௌ
	ம்	ம	மா	மி	மீ	மு	மூ	மெ	மே	மை	மொ	மோ	மௌ
	ய்	ய	யா	யி	யீ	யு	யூ	யெ	யே	யை	யொ	யோ	யௌ
	ர்	ர	ரா	ரி	ரீ	ரு	ரூ	ரெ	ரே	ரை	ரொ	ரோ	ரௌ
	ல்	ல	லா	லி	லீ	லு	லூ	லெ	லே	லை	லொ	லோ	லௌ
	வ்	வ	வா	வி	வீ	வு	வூ	வெ	வே	வை	வொ	வோ	வௌ
	ழ்	ழ	ழா	ழி	ழீ	ழு	ழூ	ழெ	ழே	ழை	ழொ	ழோ	ழௌ
	ள்	ள	ளா	ளி	ளீ	ளு	ளூ	ளெ	ளே	ளை	ளொ	ளோ	ளௌ
	ற்	ற	றா	றி	றீ	று	றூ	றெ	றே	றை	றொ	றோ	றௌ
	ன்	ன	னா	னி	னீ	னு	னூ	னெ	னே	னை	னொ	னோ	னௌ
Grantha (வடமொழி)	ஜ்	ஜ	ஜா	ஜி	ஜீ	ஜு	ஜூ	ஜெ	ஜே	ஜை	ஜொ	ஜோ	ஜௌ
	ஷ்	ஷ	ஷா	ஷி	ஷீ	ஷு	ஷூ	ஷெ	ஷே	ஷை	ஷொ	ஷோ	ஷௌ
	ஸ்	ஸ	ஸா	ஸி	ஸீ	ஸு	ஸூ	ஸெ	ஸே	ஸை	ஸொ	ஸோ	ஸௌ
	ஹ்	ஹ	ஹா	ஹி	ஹீ	ஹு	ஹூ	ஹெ	ஹே	ஹை	ஹொ	ஹோ	ஹௌ
	க்ஷ்	க்ஷ	க்ஷா	க்ஷி	க்ஷீ	க்ஷு	க்ஷூ	க்ஷெ	க்ஷே	க்ஷை	க்ஷொ	க்ஷோ	க்ஷௌ
							ஸ்ரீ	ெ◌	ே◌	ை◌	◌ா		

FIGURE 3. Complete character set of tamil script.

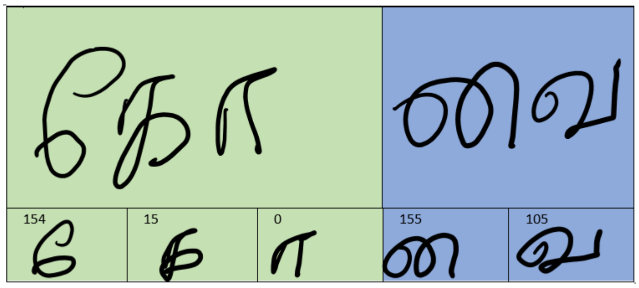


FIGURE 4. Two syllables (not among 156 classes) represented by 5 distinct characters of the database.

(throw away examples of dominant classes). Only then the model developed using this database can be suitable for real-world applications.

2) **Constrained Data Collection:** All samples are collected by digital means. However, as mentioned earlier, such samples are not true representatives of samples often seen in offline OCR applications.

2) AD-HOC DATABASES

One can find numerous works in the literature that present handwritten Tamil OCR algorithms by training them on databases created in-house [9], [37]–[40]. Many works apparently collected minimal data [37], [38]. Kowsalya and Periasamy [38] proposed an algorithm for Tamil OCR by using elephant-herd optimization. For experimental purposes, the authors have made use of limited samples collected from different native Tamil speakers. Jose and Wahi [39] have collected samples from 100 users, with each class

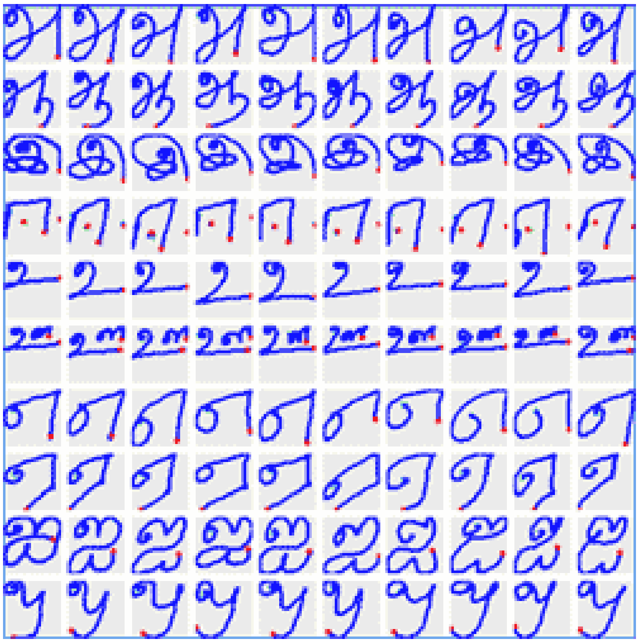


FIGURE 5. Sample images from the HPL database.

having 100 samples. There is no mention of how many classes are considered for data collection and experimentation. Shanthi and Duraiswamy [40] have trained the system with 35441 characters belonging to 106 Tamil characters. The database was created through the contribution of 117 volunteers. The test data had 6048 samples belonging to only 34 character classes. Vinotheni *et al.* [9], have created a database comprising 54600 samples with 350 samples in each of 156 categories. This work does not indicate the train-test split.

It can be ascertained that ad-hoc databases suffer from several drawbacks such as in-house creation, publicly unavailable, in-comprehensiveness and insufficient samples. This will impede the research community in carrying out a comparative analysis of algorithms proposed in the literature [19]. At least, if these databases are made publicly available, the research community will have choices and can contribute to easy prototyping and evaluation of new Tamil OCR models.

Based on this survey of the literature, the following gaps are identified:

1) Unlike any other languages such as English, Arabic, or even certain Indic scripts such as Devanagari, Bangla, Oriya, etc., there are no multiple standardized publicly available databases for Handwritten the Tamil OCR process which could involve diversity of writing styles by many people [18].
2) Majority of the standard papers deal with the HPL dataset. This database helped trigger an ample amount of research in offline Tamil handwriting recognition. However, few limitations exist, such as insufficient samples in certain classes and samples not being a true representative of offline scanned documents.
3) Various other methods proposed by different authors in the literature are giving good performance only on data prepared in an ad-hoc way. Therefore, such an algorithm cannot be reliably deployed for practical applications.

We overcome the gaps mentioned above by presenting the comprehensive uTHCD database in this paper. The uTHCD database will be the second public repository for Tamil OCR purposes. Thus, the research community can evaluate the algorithms on multiple databases instead of relying on a single database that contains only online samples. Table 1 summarises some noteworthy Tamil handwriting data repositories that are reported. The asterisk alongside the name of the database indicates that they are not designed primarily for OCR purposes but for specific handwriting recognition tasks.

TABLE 1. Description of some tamil handwriting databases.

Database	Publicly Available	Features
HPL-iso-tamil-char [19]	Yes	Number of training and testing samples are respectively 50691 and 26926. Used for OCR purposes.
HPL-iso-tamil-word* [30]	Yes	The Tamil handwritten word database. Contains 100 samples each for 85 words. Used for Word Recognition.
Handwritten City Names Database* [41]	Yes	Dataset of 265 Tamil city names. Used for postal automation.
Handwritten Tamil and Kannada Word Corpus* [42]	Yes	100000 words each in Kannada and Tamil. Collected from 500 writers. Useful for Word Recognition.
Jose et al. [39]	No	Samples collected for 100 classes only. Each class has 100 samples. Used for OCR purposes.
Shanti et al. [40]	No	117 volunteers contributed to the creation of 35441 characters belonging to 106 tamil characters.Used for OCR purpose.
Vinotheni et al. [9]	No	Consists of 54600 samples with 350 samples in each of 156 classes. Used for OCR purpose.

* — Not suitable for OCR purpose

III. DATABASE WORK

The database creation is a laborious process that involves many tasks such as sample collection, data extraction, carrying out necessary postprocessing, verification, and finally making it ready for experimentation by associating it with GT. The protocol we followed adheres to standards adopted in the database creation literature [19], [22], [25], [43] as any mistake would nullify all efforts and affect benchmarking process. The overall process involved is as shown in Fig.6. This section succinctly presents all steps involved in creating this database and possibly serves as a future guide for more similar works.

A. SAMPLE COLLECTION

The database is a unified collection of samples collected from offline and online modes. The overall process involved in both is the same except that the online mode was relatively

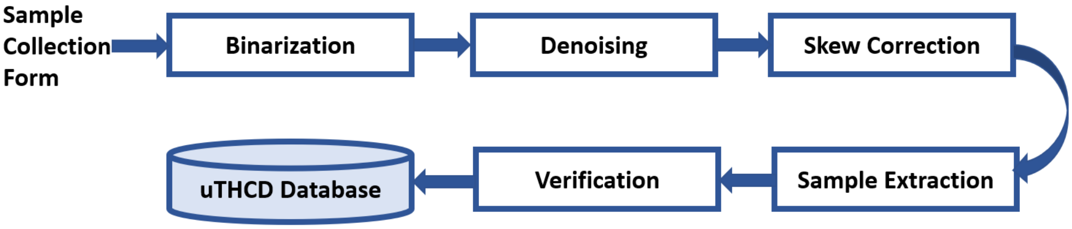


FIGURE 6. The important steps in the creation of uTHCD database.

straightforward as it did not necessitate preprocessing the document through the skew estimation step.

The offline data collection process starts with two A4-sized forms, namely form-1 and form-2, each having a 10 × 8 sized grid with a specific place and space for writing each character. The form-1 collects samples of the first 80 characters of Tamil script, and the remaining 76 characters were filled in form-2. In total, 400 forms (200 each of form-1 and form-2) were distributed to 400 different volunteers. Hence, each volunteer either fills form-1 or form-2;thereby, 200 complete sets of samples pertaining to 156 classes were collected. Each volunteer was requested to fill one form only because casting a complete set of 156 characters would cause possible lethargy and may lead to unnecessary processing of samples that are not a true representative of the underlying character class. A sample physical form used in this data collection process is shown in Fig.7.

A similar high-resolution grid image (digital form) was designed for an online sample collection process. Each volunteer was asked to write in the predefined space like the offline mode. The samples written by volunteers using a digital writing pad were later extracted and saved as image files. The samples collected in the offline and online mode were curated to contain approximately 600 samples from each class. The online and offline samples are constituting around 70% and 30% of the database, respectively. This whole process involved contributions from around 450 volunteers over a period of 2 years. The data collection happened among students and staff belonging to several schools, universities, and community places where there are sizeable native Tamil speakers. The data has been collected from volunteers belonging to different age groups, gender, and varying handedness. The statistical distribution of volunteer details shown in Fig.8.

Some quality checks we ensured during this process are:

1) The volunteer was asked to provide the sample within the cells in the grid to not overlap with characters in adjacent cells. A negligible amount of samples breached this, but it was easy for us to discard such samples during the verification process.
2) Each volunteer must have formally studied the Tamil language as a primary/secondary language during their school. This ensures that writing style captured in the sample would not drift much from the typical writing style of the Tamil script [19].

The most time-consuming part of data collection was in the offline mode as it involved a waiting period where we could not reliably receive all the forms from volunteers couriered to different parts of the world.

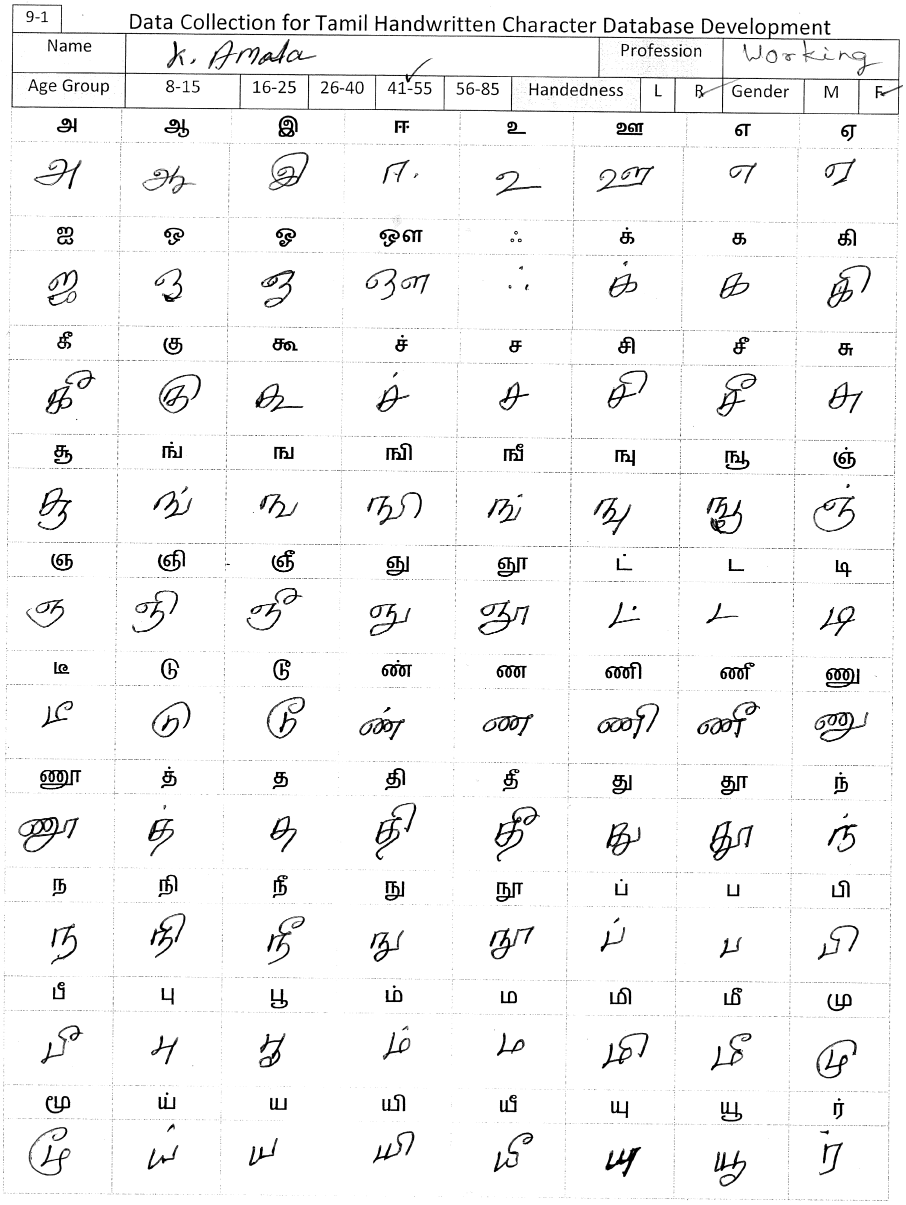
B. PREPROCESSING AND SAMPLE EXTRACTION

The offline data collection forms are first digitized at 300 DPI through an HP flatbed multi-functional scanner. The scanned image was then assigned a unique three-digit number between 001 to 600, indicating the volunteer code. After digitizing the form, it undergoes specific preprocessing steps before an automated task extracts individual characters present in each form using a simple Python script. At first, the scanned image *Img* was binarized using simple global thresholding (where T is a preset threshold):

$$Img(x, y) = \begin{cases} 1, & \text{if Img(x,y)} \geq T \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Subsequently, median filtering and size-based connected component empirical analysis were carried out to eliminate isolated pixels and irrelevant clusters of pixels arising due to various reasons such as dust in the scanner bed, liquid spillage, smudging, etc. The scanned image was then skew corrected using the ePCP algorithm [44]. This algorithm accurately computes the skew angle by searching for white sections (the longest possible run of white pixels) at different angles. The angle that produces maximum white sections is deemed as the skew angle of the document [44]. We exercised extreme caution while feeding the page during the scanning process to ensure minimal skew. This helped us keep the ePCP profiling in the range of $-5°$ to $+5°$ with an increment of $0.1°$ for each step, ensuring minimal computation. The same steps were repeated for digital forms as well, except that they were not subjected to the skew correction step for obvious reasons. The pseudocode of the overall preprocessing steps applied is given in Algorithm-1.

As both offline and online samples were collected by asking volunteers to write within the grid, we used starting coordinates of the grid, and with appropriate height and width increments resulted in the accurate extraction of character samples. Samples were aggregated in one of the 156 classes



9-1

Data Collection for Tamil Handwritten Character Database Development

Name	k. Amala				Profession	Working				
Age Group	8-15	16-25	26-40	41-55 ✓	56-85	Handedness	L	R ✓	Gender	M	F ✓

அ	ஆ	இ	ஈ	உ	ஊ	எ	ஏ
ஐ	ஒ	ஓ	ஔ	ஃ	க்	க	கி
கீ	கு	கூ	ச்	ச	சி	சீ	சு
சூ	ங்	ங	ஙி	ஙீ	ஙு	ஙூ	ஞ்
ஞ	ஞி	ஞீ	ஞு	ஞூ	ட்	ட	டி
டீ	டு	டூ	ண்	ண	ணி	ணீ	ணு
ணூ	த்	த	தி	தீ	து	தூ	ந்
ந	நி	நீ	நு	நூ	ப்	ப	பி
பீ	பு	பூ	ம்	ம	மி	மீ	மு
மூ	ய்	ய	யி	யீ	யு	யூ	ர்

FIGURE 7. A 10 × 8 grid based sample collection form.

based on their location in the grid. In this way, each class contains samples from both online and offline modes collected from all volunteers. Each sample will be saved using the filename convention: xxxx_yyy, where yyy represents the underlying class number and xxxx corresponds to a 4-digit volunteer number starting serially from 0001 to 0600 (additional MSD digit is to accommodate future expansion). This way yyy will also serve as GT for the corresponding sample and assist in the GT verification during the final verification phase. For instance, samples in class-1 will be named 0001_001, 0002_001, 0003_001, etc. To further distinguish between offline and online samples, the letter 's' will be affixed in the format as xxxxs_yyyy. Some sample offline and online samples thus extracted are shown in Fig.9

C. SAMPLE VERIFICATION

Two levels of verification were carried out to ensure that samples extracted are true representatives of underlying Tamil character classes. Form-by-form verification takes place in Level-1, whereas in Level-2, sample-by-sample verification happens. The complete verification process we followed is outlined in Fig.10.

In the Level-1 verification, we classify each form (both physical or digital) into three categories:

1) *Fit*: If the writing style of 90% or more of the samples represents the corresponding character classes.
2) *Relatively Fit*: If at least 50% of the samples in the form represent the corresponding character classes.

Algorithm 1 Preprocessing Handwritten Forms

```
Input:A Scanned Document Image (Physical/Digital Form)
Output: Preprocessed Form.
METHOD:
Step-1: Binarize the Img using global thresholding (Equation.1)
Step-2: Noise Removal
2.1: $Img \leftarrow$ Apply Median Filtering on Img.
2.2: $Img \leftarrow$ Connected Component Analysis and remove. components whose size $\leq T$
Step-3: Skew Correction using ePCP algorithm
3.1: Compute White Sections (PCP) of Img between -5° to +5° in steps of 2°.
3.2: Let $PCP_\theta$ corresponds to angle $\theta$ for which white sections are maximum.
3.3: $stepSize \leftarrow 1.0$
while $stepSize \geq 0.1$ do
    $PCP_{\theta+} \leftarrow$ compute white sections at angle: $\theta + stepSize$
    $PCP_{\theta-} \leftarrow$ compute white sections at angle: $\theta - stepSize$
    $\theta \leftarrow \underset{\theta}{\text{argmax}}\,(PCP_\theta, PCP_{\theta+}, PCP_{\theta-})$
    $stepSize \leftarrow stepSize/2$
end while
3.4: $preImg \leftarrow$ Skew correct $Img$ using computed skew angle $\theta$
```

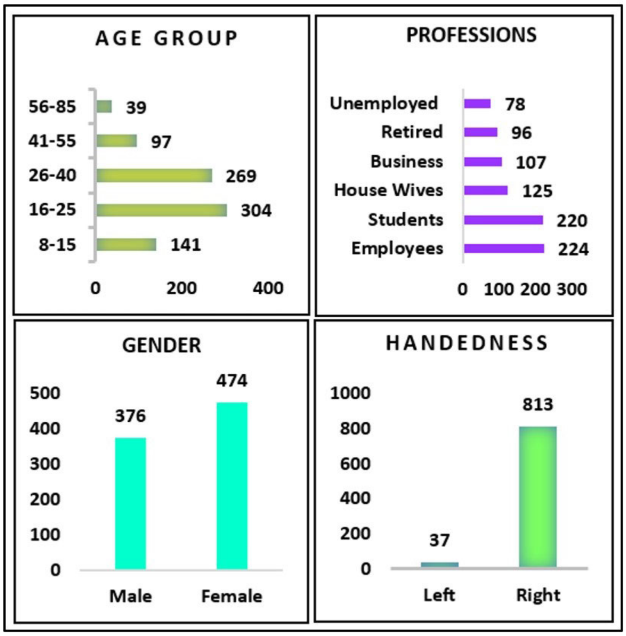


FIGURE 8. The volunteer statistics.

3) *Unfit*: If there is a considerable percentage of samples that deviate much from the conventional style of the character.

The data deemed unfit are excluded from the rest of the process and will not be collected in the eventual database. The remaining data in the fit and relatively fit categories are subject to preprocessing and sample extraction processes.

The Level-2 verification has two main objectives:

1) The sample extracted represents one of the 156 classes of Tamil script.
2) Ensure GT associated with the sample conforms to the underlying class.

This sample-by-sample manual verification occurs after the data file is preprocessed, and samples are extracted from the fit and relatively fit categories. Then, the samples will be either discarded (if the sample does not represent any Tamil character class) or collected in an appropriate class. A few essential comments from the Level-2 verification process are:

- Some samples had awful writing styles that were even hard to be recognized correctly by humans. Such samples were eventually discarded.
- Many characters in Tamil script look almost the same except for the presence or absence of a tiny loop, dot/hole component, etc. This minor inter-class variation sometimes causes a character written in one class to be considered in an appropriate class that resembles best.
- Some unintended scratch marks by volunteers accompanied the samples. However, we retained samples by only carefully removing such marks instead of completely ignoring the whole sample. This was found only in samples collected in offline mode.

Finally, for GT verification, we used the file naming convention (xxxxs_yyy) we followed during the data extraction process. While doing sample-by-sample verification, we also cross-verified the GT (ie yyy) from the filename to ensure it belongs to the correct class. In this way, a simple Python script helped us generate a GT file with two columns of data that has filenames and corresponding class numbers as the ground truth.

D. THE uTHCD: UNCONSTRAINED TAMIL HANDWRITTEN CHARACTER DATABASE

The samples thus collected after arduous processes is our current uTHDC database with provision for subsequent expansion. There were varying numbers of samples in each class, but for data balancing and ease of future expansion,

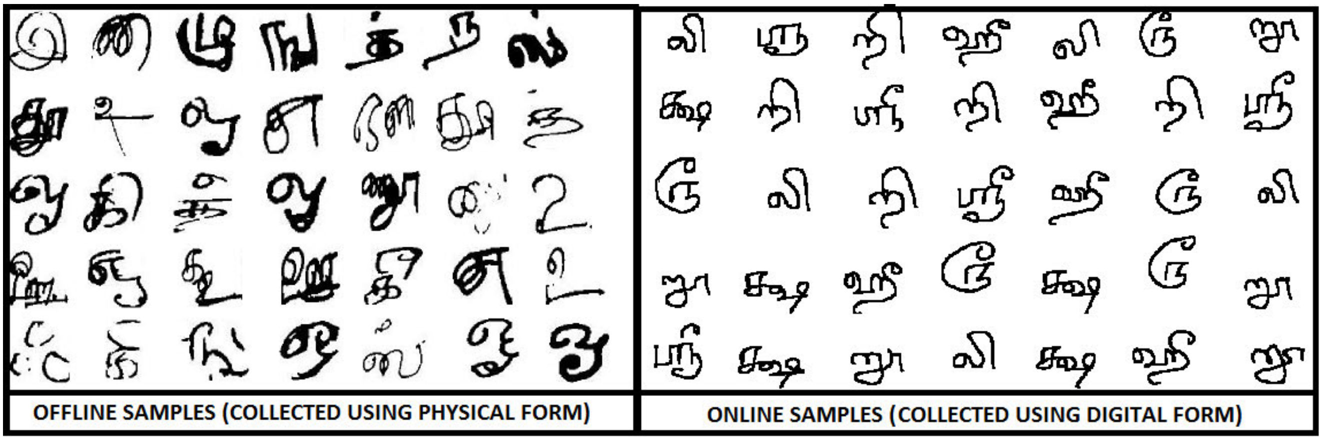

OFFLINE SAMPLES (COLLECTED USING PHYSICAL FORM)
ONLINE SAMPLES (COLLECTED USING DIGITAL FORM)

FIGURE 9. Sample offline and online images in the uTHCD database.

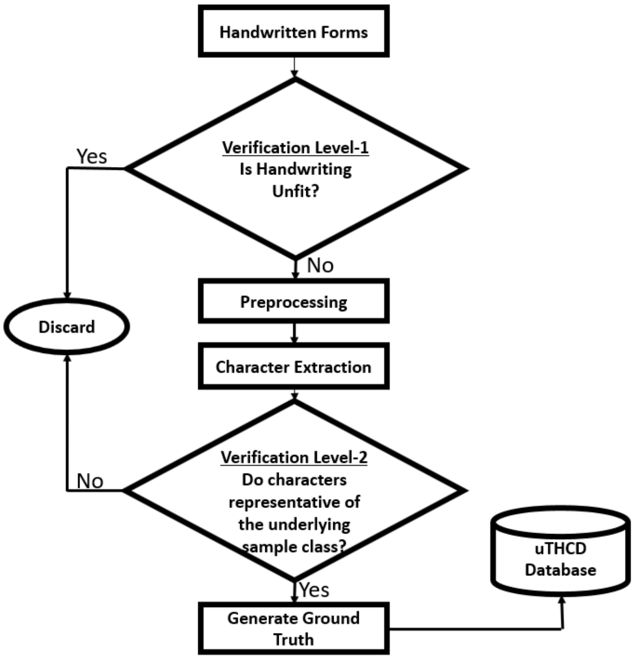


FIGURE 10. The verification process involved in uTHCD database.

we limited the number of samples in each class to be approximately 600. All 156 character classes of this database and corresponding class id are as shown in Fig.11. The mapping of Unicode sequences with 156 classes in the uTHDC is given in Appendix.

The final database contains a total of 90950 images. The offline samples constitute around 30% of total samples, and the rest of all samples are online samples. The data collected are available as separate repositories with different train-test proportions viz 70-30 (uTHCD_a), 80-20 (uTHCD_b), 85-15 (uTHCD_c) and 90-10 (uTHCD_d). We ensured train and test folders to contain both offline and online samples. A randomized process was used in creating these repositories. Details about each of these repositories are tabulated in Tab.2.

TABLE 2. Details of various repositories of proposed database.

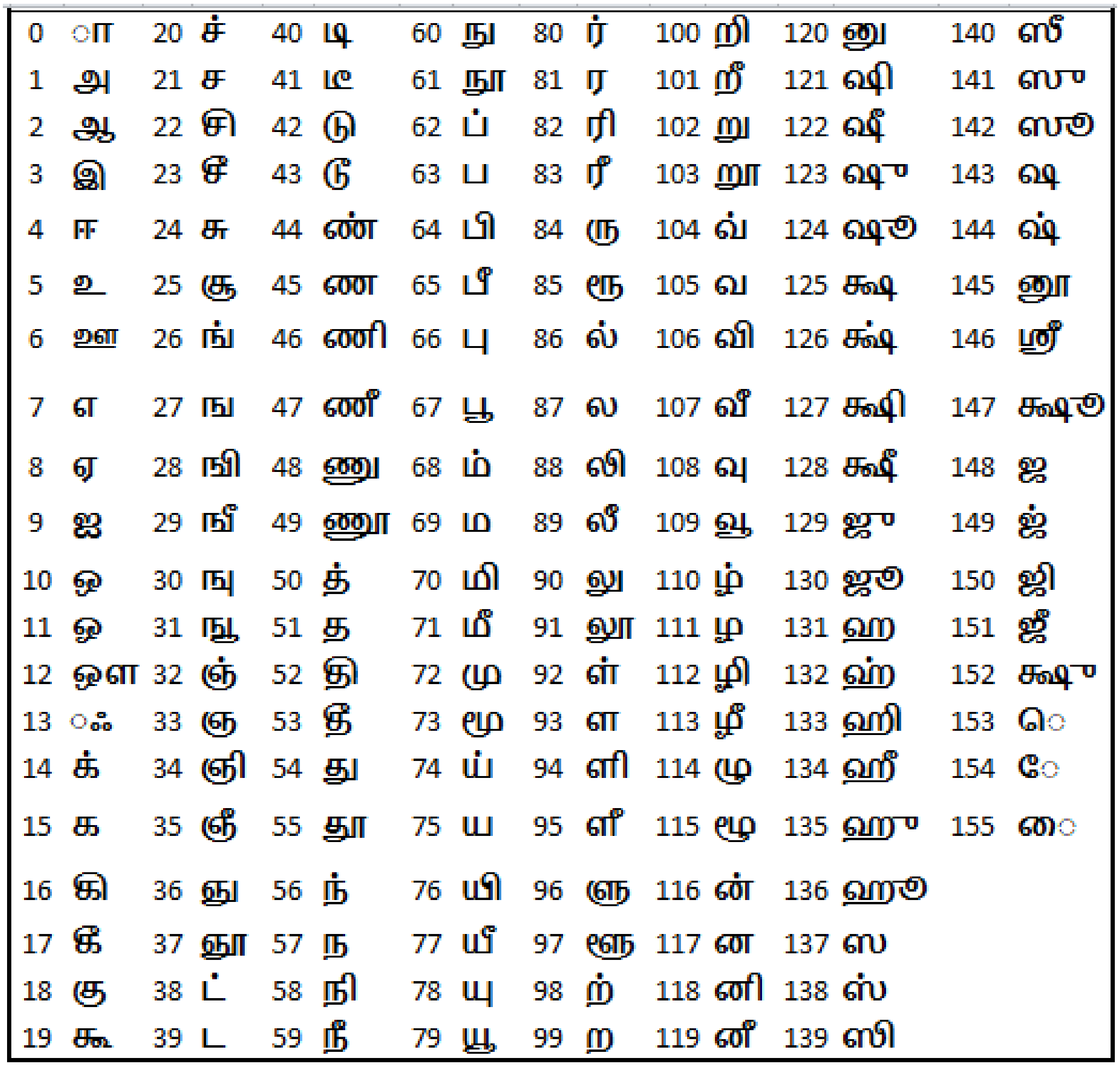
Repository	Train-Test Split:	# of Train Samples	# of Test Samples
uTHCD_a	70-30	62870	28080
uTHCD_b	80-20	71760	19190
uTHCD_c	85-15	77376	13574
uTHCD_d	90-10	81900	9050

All four repositories of the uTHCD database are publicly available (in this hyperlink) at the Kaggle website - a popular online data science community. Researchers can use the provided hyperlink to visit the repository. In addition, the downloadable database is available in two different formats:

1) **The HDF5 format**: The train and test set are stored as Python NumPy multidimensional arrays by names x_train, y_train, x_test, and y_test. These arrays can then be compressed and stored in a single HDF5 file format. Availability of the database through this format will enable researchers to rapidly acquire the data, which will be suitable for direct experimentation without much preprocessing. The details of data arrangement inside the HDF5 file (.h5 extension) and supporting Python code to extract the train and test set are given in Appendix.
2) **The Raw Image format**: The total of 90950 images are available as a single compressed RAR file, including ground truth files for both train and test folders. Researchers can then acquire this data according to their implementation preferences. The images are stored in binary format.

IV. BENCHMARKING

Convolutional Neural Networks (CNNs) have dominated most of the recent technological advancements in Artificial Intelligence, especially in image recognition tasks [45]. The main reason for that is the availability of large datasets, massively parallelizable GPUs, and a broad spectrum of open



0	◌ா	20	ச்	40	டி	60	நு	80	ர்	100	றி	120	னு	140	ஸீ
1	அ	21	ச	41	டீ	61	நூ	81	ர	101	றீ	121	ஷி	141	ஸு
2	ஆ	22	சி	42	டு	62	ப்	82	ரி	102	று	122	ஷீ	142	ஸூ
3	இ	23	சீ	43	டூ	63	ப	83	ரீ	103	றூ	123	ஷு	143	ஷ
4	ஈ	24	சு	44	ண்	64	பி	84	ரு	104	வ்	124	ஷூ	144	ஷ்
5	உ	25	சூ	45	ண	65	பீ	85	ரூ	105	வ	125	க்ஷ	145	னூ
6	ஊ	26	ங்	46	ணி	66	பு	86	ல்	106	வி	126	க்ஷ்	146	ஸ்ரீ
7	எ	27	ங	47	ணீ	67	பூ	87	ல	107	வீ	127	க்ஷி	147	க்ஷூ
8	ஏ	28	ஙி	48	ணு	68	ம்	88	லி	108	வு	128	க்ஷீ	148	ஜ
9	ஐ	29	ஙீ	49	ணூ	69	ம	89	லீ	109	வூ	129	ஜு	149	ஜ்
10	ஒ	30	ஙு	50	த்	70	மி	90	லு	110	ழ்	130	ஜூ	150	ஜி
11	ஓ	31	ஙூ	51	த	71	மீ	91	லூ	111	ழ	131	ஹ	151	ஜீ
12	ஔ	32	ஞ்	52	தி	72	மு	92	ள்	112	ழி	132	ஹ்	152	க்ஷு
13	◌ஃ	33	ஞ	53	தீ	73	மூ	93	ள	113	ழீ	133	ஹி	153	ெ◌
14	க்	34	ஞி	54	து	74	ய்	94	ளி	114	ழு	134	ஹீ	154	ே◌
15	க	35	ஞீ	55	தூ	75	ய	95	ளீ	115	ழூ	135	ஹு	155	ை◌
16	கி	36	ஞு	56	ந்	76	யி	96	ளு	116	ன்	136	ஹூ		
17	கீ	37	ஞூ	57	ந	77	யீ	97	ளூ	117	ன	137	ஸ		
18	கு	38	ட்	58	நி	78	யு	98	ற்	118	னி	138	ஸ்		
19	கூ	39	ட	59	நீ	79	யூ	99	ற	119	னீ	139	ஸி		

FIGURE 11. Mapping of class number and tamil characters.

deep learning frameworks [46]. The CNNs are also vastly famous for simultaneously circumventing the high dimensionality problem and extracting robust features for computer vision tasks [47]. There are three main building blocks of a CNN: i) convolutional layers, ii) zero or more pooling layers, and iii) one or more fully connected layers. Various architectures have been proposed in the literature by varying these building blocks [45]–[48], which have obtained outstanding results.

In this section, we present possible use cases for the proposed uTHCD database by conducting the following set of experiments:

1) We first learn a CNN from scratch with minimal building blocks. This set of experiments demonstrates several use cases of the proposed database to address the OCR classification problem from typical CNN perspectives.
2) Subsequently, we leverage a pre-trained SOTA CNN and fine-tune it to classify Tamil characters using the proposed database.

This set of experiments establishes several benchmarks for the proposed database using SOTA classification algorithms.

As this study predominantly uses the CNN architectures for reporting the use-case scenarios of the proposed dataset, the overall flow diagram of the model building using CNN is shown in Fig.12. The processing starts by loading the uTHCD train and test sets (either RAW images or HDF5 format). Then, experiments are conducted by considering a 70:30 train-test split (the uTHCD_a repository). This helps to compare the results with the existing database as the number of test samples is comparable in both repositories. Firstly, the validation set can be extracted from the training set by slicing the last 7270 samples (relevant Python codes are given in Appendix). In the next step, the data suitable for model

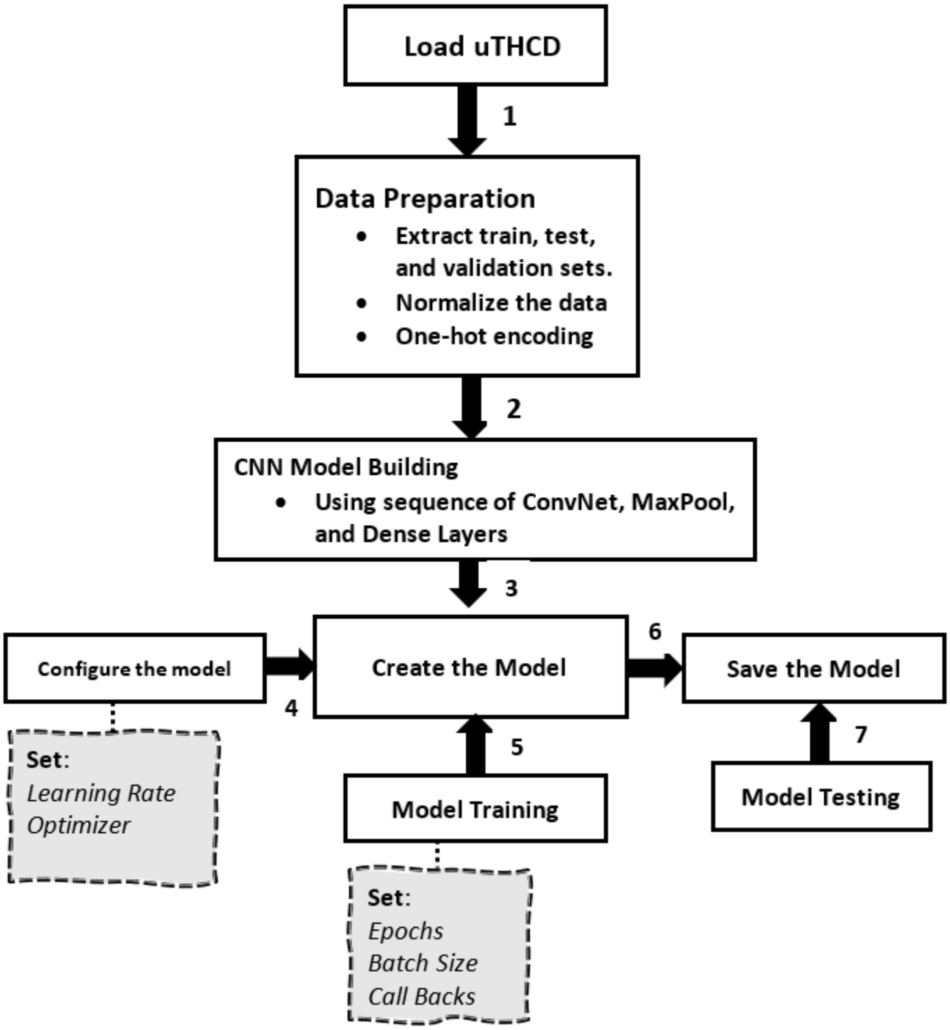


FIGURE 12. The sequence flow diagram of model building.

training is prepared. So, the image files (x_train, x_test, and x_val) are all normalized to have the values in the range [0, 1], and the corresponding ground truth values (y_train, y_test and y_val) are one-hot encoded.

The CNN model can now be created with the proposed sequence of Convolutional(ConvNet), Pooling (MaxPool), and fully connected (FC) layers. In this step, we can instead leverage an existing SOTA CNN model (as discussed in Section IV-B). The model can then be configured with a preset learning rate, optimizer, etc. Finally, the model is ready to be trained for a fixed number of epochs with specific callbacks. We made use of the *EarlyStoppingy* Keras callback – this helps to terminate the training process when the loss is no longer decreasing. It has a regularization effect as well, in addition to assisting in optimal usage of processor resources. Finally, the trained model can be saved for further testing purposes.

We used four performance metrics: accuracy, specificity, sensitivity (recall), and F1-score to evaluate and establish the benchmarking using CNN models [49]. Accuracy refers to the fraction of correctly classified samples over all samples. Specificity is the fraction of samples classified as belonging to the negative class over a total number of examples in the negative class. In the context of multiclass classification, it can be interpreted as a fraction of samples correctly classified as belonging to other classes over a total number of samples belonging to different classes. This measure helps to discover the false positive rate(FPR) as 1-specificity. The sensitivity (or recall) measures the true positive rate(TPR) — a fraction of all samples correctly classified as belonging to the same class over a total number of samples belonging to same class. Finally, the F1-score is a weighted average of precision and recall. The FPR, TPR, and F1-score reported are average values obtained with the one vs. all strategy [49].

Implementation and subsequent evaluation were carried out on a machine with GPU (NVIDIA GeForce MX330) running on top of an Intel i7 1.6 GHz processor with 16GB RAM. The entire work was implemented

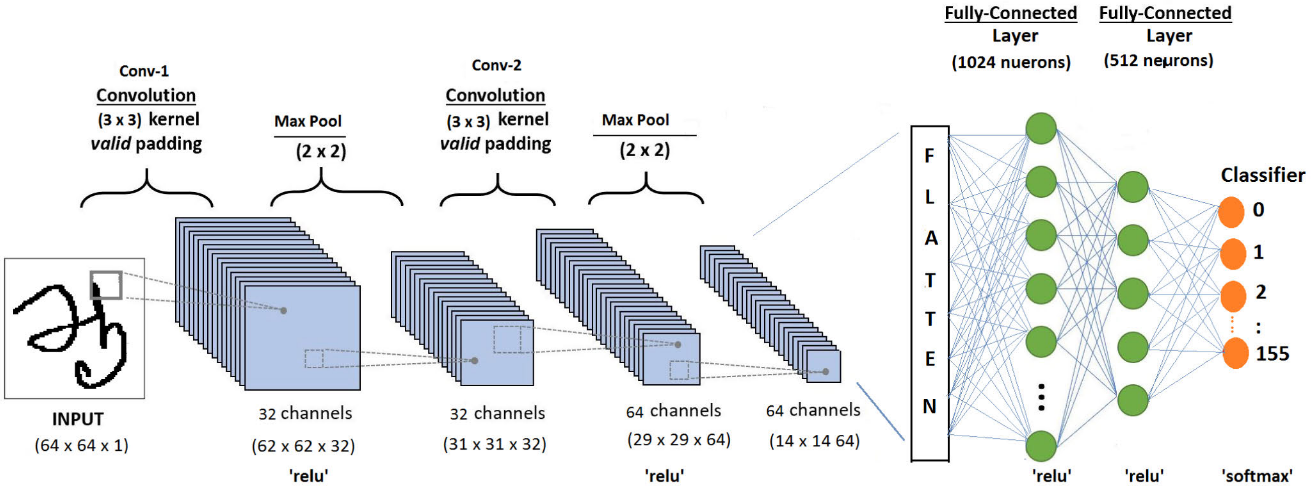


FIGURE 13. The basic CNN architecture used.

using Tensorflow end-to-end machine learning library with Keras API using Python 3.7.10 environment. Furthermore, TensorRT 6.0 neural network optimizer was synced with NVIDIA to optimize memory consumption, throughput, latency, and power efficiency.

A. BASIC CNN ARCHITECTURE

We aim to design a minimalist CNN architecture to establish baseline accuracy for the proposed database. The research community can then offer efficient CNN architectures to demonstrate robust performance using the uTHCD database. We designed several combinations of architectures by varying different types of CNN layers with a restriction of not using more than two layers for each layer type (not counting the final classification layer). The combination that yielded the best training accuracy is the proposed CNN architecture, as shown in Fig.13. A similar architecture also yielded optimal results for various computer vision problems [50].

The CNN architecture has two convolutional layers, two pooling layers, and two fully connected dense layers. The input layer contains 64 × 64 sized images. The architecture can be represented as: $64C2-MP2-64C2-MP2-1024N-5128N$, where nCj indicates a convolutional layer with n filters and $j \times j$ kernel, MPk refers to a max pool layer with $k \times k$ kernel, and fN refers to a fully connected layer with N neurons. The output layer is a classifier with the softmax activation function to classify 156 classes.

The study of optimal values for hyperparameters of CNN is a challenging task, and it is data dependent [46]. The learning rate hyper-parameter α has a significant impact as the convergence of the model heavily depends on this parameter which needs to be carefully fixed to an optimal value before tuning other hyperparameters. We conducted an initial set of experiments by independently learning our CNN model using different optimizers. The performance of these models (for $\alpha = 0.001$) observed for 35 epochs is shown in Fig.16. It can be noted that the performance of Adam, Nadam, and RMSProp is comparable to each other while they perform significantly better than AdaGrad and AdaDelta. Furthermore, the behavior of these optimizers was similar when we changed the α from 10^{-3} to 10^{-6} in steps of 10^{-1}. Based on this empirical evidence, we fixed the Adam optimizer with a learning rate $\alpha = 0.001$ for all subsequent experiments.

Next, we experimented to determine the optimal value for batch size as it is a consequential hyperparameter in any network that controls the overall dynamics of the network architecture [51]. We varied the batch size starting from 2^5 to 2^{12} in steps of a power of 2. The plot of train, test, and validation accuracy vs. batch size is shown in Fig.14. This was implemented with the *EarlyStopping* Keras callback observed on validation loss. From Fig.14, we can see that batch size does not significantly impact the accuracy. Hence, we can conclude that algorithms using this database on a CNN can prefer a mini-batch gradient descent with 32 as an optimal value for the batch size that yields the same accuracy without compromising much on the computational burden and speed of the learning model.

Apart from the learning rate, optimizer function, and batch size, another influential hyper-parameter known as the activation function that must be fixed. Several activation functions such as tanh, sigmoid, ReLU, etc., are available for assisting the CNN to learn the complex patterns in the data. We used activation functions which are generally used in the literature [52] – Rectified Linear Unit (ReLU) for hidden layers and softmax for the final classification layer. The ReLU and softmax activation functions are defined as below:

$$ReLU(x) = max(0, x) \tag{2}$$

$$softmax(x_i) = \sigma(x_i) = \frac{e^{(x_i)}}{\sum_j e^{(x_j)}} \tag{3}$$

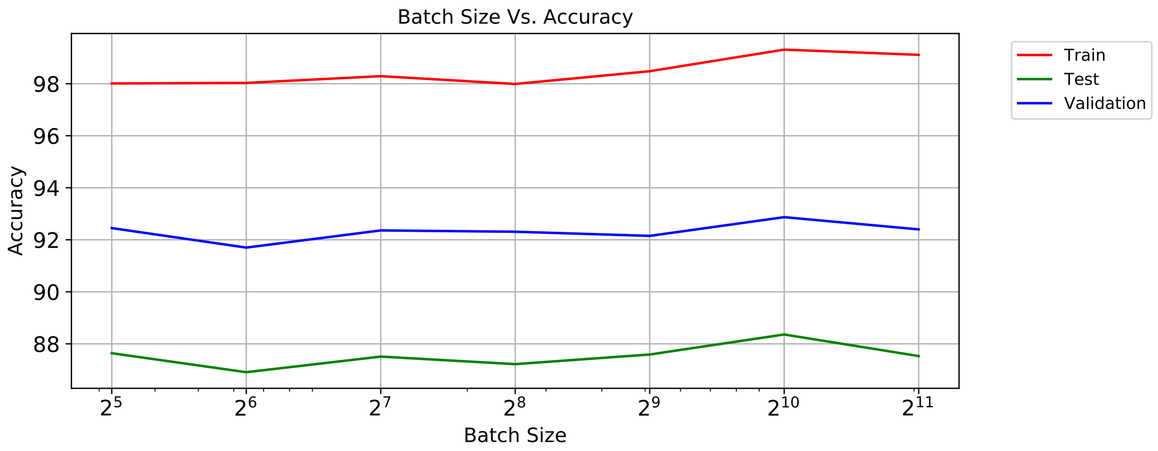


FIGURE 14. Accuracy for varying batch sizes.

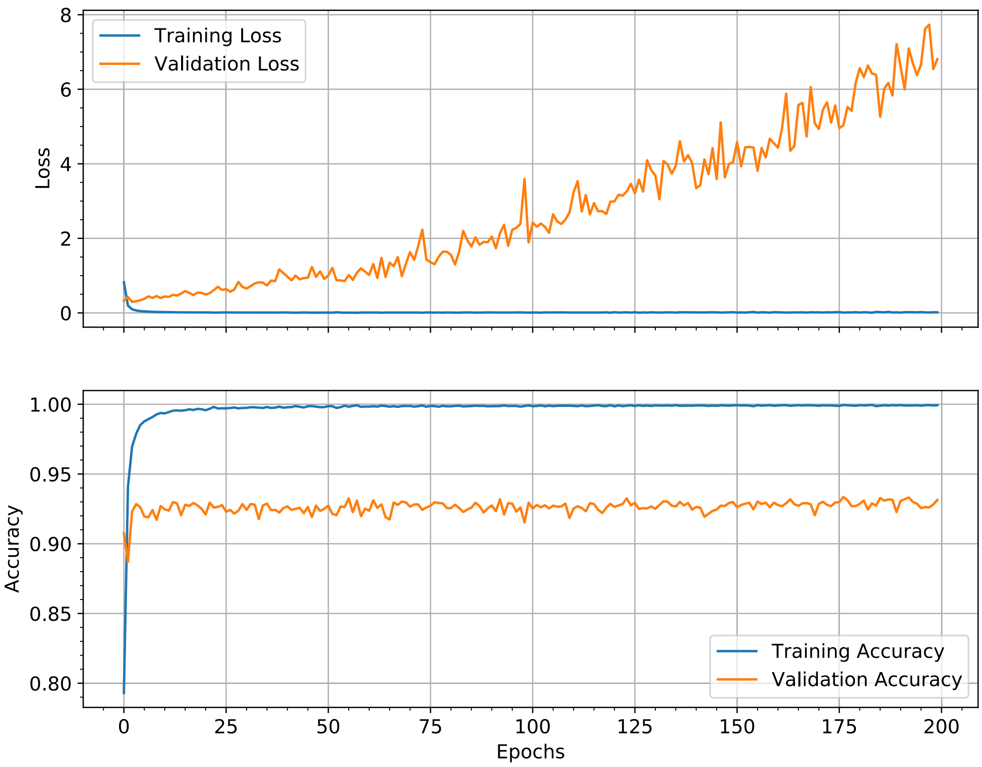


FIGURE 15. Performance on training vs validation sets.

where x_i is the input vector. Finally, the Xavier initializer was used to initiate the kernel values. For the rest of the CNN based-experiments, unless explicitly mentioned, the hyperparameter values used are shown in the table-3.

To obtain a holistic idea about the behavior of the proposed CNN architecture, we first trained the model for 200 epochs (without the trigger of the *EarlyStopping* callback). The plot of accuracy and loss for training and validation sets are as demonstrated in Fig.15. We can observe that although training accuracy and loss have converged around ten epochs, the validation loss keeps increasing. This is because the training accuracy/loss of a set is evaluated through the softmax function by outputting the class label corresponding to the highest probability. It does not consider the magnitude of corresponding class probability. In other words, although a

TABLE 3. Hyperparameters values used in the CNN architecture.

Hyperparameters	Values	Remark
Learning Rate α	0.001	Fixed Empirically
Activation Function	Adam	Fixed Empirically
Batch size	32	Fixed Empirically
Epochs	200	Controlled by *EarlyStopping* callback.

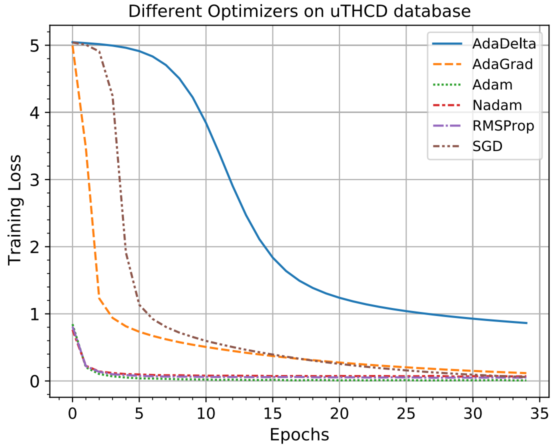


FIGURE 16. **Performance of different optimizers with $\alpha = 0.001$.**

correctly classified sample with marginal probability contributes to the accuracy, the model is still penalized with loss. The fluctuating validation accuracy is due to the model picking spurious patterns along some useful ones. This baseline model resulted in a training accuracy of 99.61% with a validation accuracy of 92%.

Next, we used the test set to evaluate the efficacy of the model. The test set obtained an accuracy of 87.00%. Considering the fact that human level performance can get nearly 0% error for handwritten OCR cases(which is also known as Bayes error/optimal error), these results indicate that the proposed model suffers from the high variance problem. This is because the proposed CNN model overfits the training data but performs significantly less with unseen test data. This can be addressed in different ways such as implementing regularization techniques or even implementing a more appropriate network architecture. Ideally, the result should indicate a low variance and low bias situation where there is not much gap in the train and test error.

Fig.17 shows the plot of misclassification errors for 156 classes. Some example misclassified samples are shown in Fig.18. For the misclassification study, the confusion matrices[3] (CMs) separately for set of vowels and 18 sample compound characters are drawn, as shown in Fig.19. We list below some error analysis we carried out by observing the GT associated with misclassified samples (Use Fig.11 for reference):

- Misclassification happens between short vowels (*Kuril*) and long vowels (*Nedil*), which differ in their overall shape with minor changes (e.g.:elongation of a stroke, presence of tiny modifier, etc.). Class 1 to 12 represents these vowels.
- There is a high between-class similarity among the compound characters formed out of vowel-consonant combinations, leading to misclassification. The differences between these classes are only through the presence or absence of i) a small hollow circle in the top (known as *Anurava*), or ii) a loop at the end of the upper stroke or iii) just elongation of the primary stroke. Observe every six classes starting serially from class 14 to 142, which represent such compound characters.
- Several other consonants (for example, class 15 and 21, 67 and 79) have high similarity in their shape due to the same reasons outlined above.
- Particular class triplets exhibit a very high similarity between them, for example (140, 141, 142),(126, 127, 128) etc. This high similarity coupled with unique writing styles further diminishes the already slight variation that exists between these class triplets leading to misclassification.

TABLE 4. **Dropout in the convolutional layers.**

Conv1 Layer	Conv2 Layer	Train	Test	Validation
0.05	0.05	96.09	89.26	92.81
0.05	0.10	95.55	90.68	94.14
0.10	0.05	**96.5**1	**91.10**	**94.46**
0.10	0.10	94.38	90.27	93.62

Some of these observations are reflected in the CMs shown in Fig.19. Hence we can conclude that the misclassifications that we observed are mainly due to the structural properties of the Tamil script.

We next conducted experiments to resolve the problem of high variance our model suffered. One of the first things to try when there is a high variance problem is regularization. Dropout is a popular regularization technique that we used to resolve the high-variance issue. Srivastava *et al.* [53], who introduced the original dropout layer concept, used dropout (with probability=0.5) in each of the fully connected layers. It was not used on the convolutional layers. So, we conducted the first series of experiments by varying the dropout rate in the fully connected dense layers only. The result is shown in Fig.20. The dropout probability of 0.45 and 0.5 curtailed the train and test accuracy gap, thereby slight improvement towards the high variance problem. Therefore, it can be ascertained that these results comply with what was suggested in [53].

Of late, a study [54] proposed applying the dropout strategy in the convolutional layers as well, however, with a much lower value of probability (0.1 or 0.2). Hence, to further mitigate the high variance problem, keeping the dropout rate of 0.5 in the dense layers, we conducted the next set of experiments by trying the dropout in the convolutional layers. The results are tabulated in table-4. The best accuracy on the test set was obtained for a dropout probability of 0.10 and 0.05 in conv1 and conv2 layers, respectively which further shrunk the gap between the train and test accuracy. However, the test accuracy needs further improvement.

The final set of experiments for this basic CNN was conducted to further shrink the train and test accuracy gap

[3] The CMs were drawn by training separately with mentioned vowels and set of compound characters since visualizing a 156 class CM is not easily decipherable

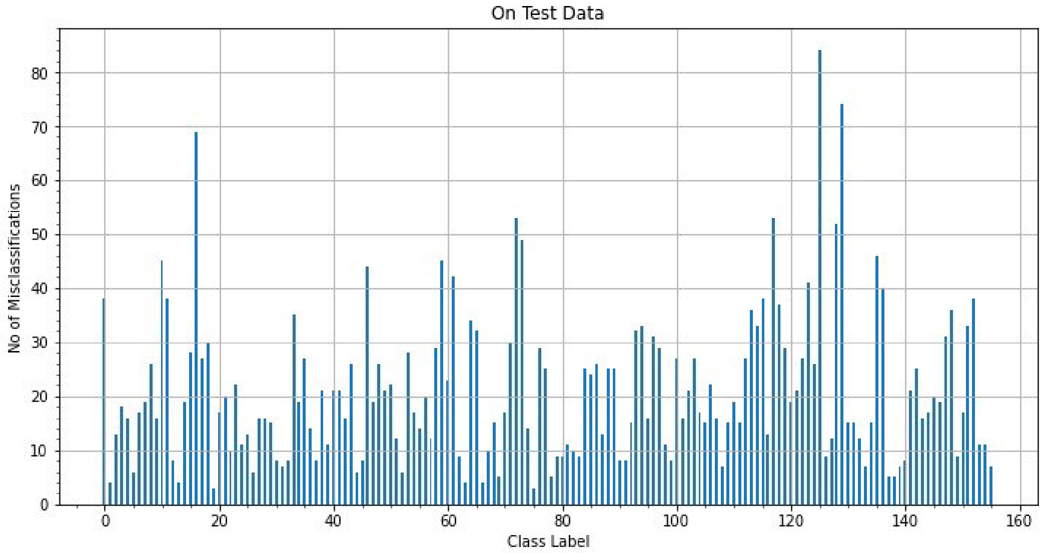


FIGURE 17. Misclassification count for 156 classes –Total misclassification is 3650 out of 28080 samples.

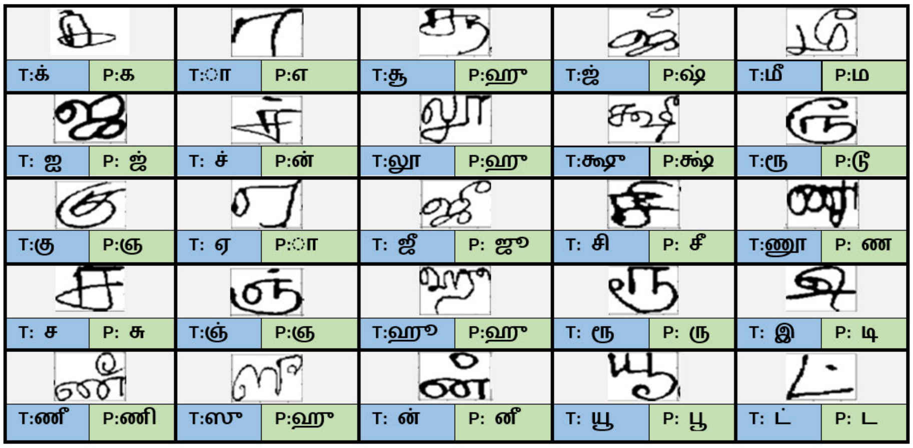


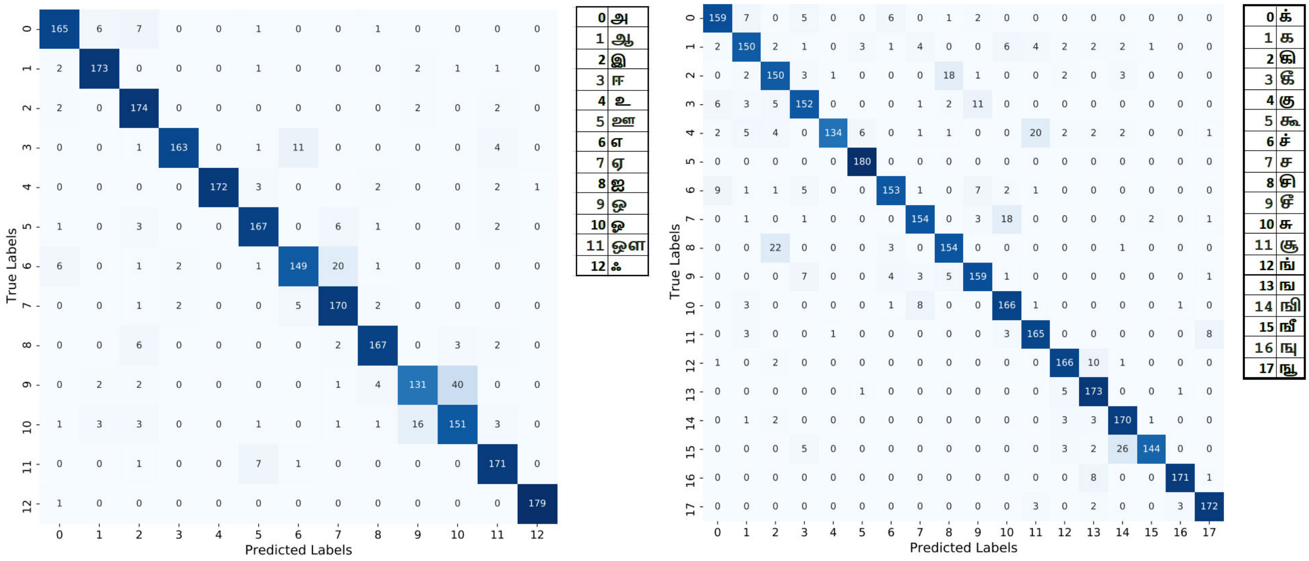
FIGURE 18. Some example misclassified samples from the test set (T:True, P:Predicted).

through data augmentation techniques. Data augmentation aims to supplement the training set by synthesizing artificial data from actual data [55]. Data augmentation has a regularization effect and hence reduces the overfitting of the model [55]. We augmented the training data by applying four variations: i) rotation (up to ±15 degrees), zoom (with 0.2 factor), and horizontal/vertical shift variations (up to 20 pixels) to the original samples. One or more of these variations were applied to a sample to generate additional examples. Fig.21 shows the example augmented images of a character (class 7). This experiment was conducted with established optimal dropout probabilities in each layer of the model.

Two types of augmentation strategies were applied to generate additional samples. Firstly, every image in the training set is augmented by implementing one or more mentioned variance. This way, the training set remained perfectly balanced with an equal number of samples in all classes. Secondly, a randomly chosen training image transforms as mentioned earlier, and with this, the training set remained relatively balanced. Results after augmentation using these two types are respectively shown in Table-5 and Fig.22. The *Diff* column shows the difference between train and test accuracy, which was initially 12.6% without the application of regularization techniques.

TABLE 5. Results for varying augmentation factors.

Factor	# of Samples	Train	Test	Validation	Diff
1.0	125740	96.07	92.32	96.70	3.75
2.0	188610	95.57	92.86	97.78	2.71
3.0	251480	95.76	93.32	98.35	2.44
4.0	314340	95.63	93.31	98.53	2.32



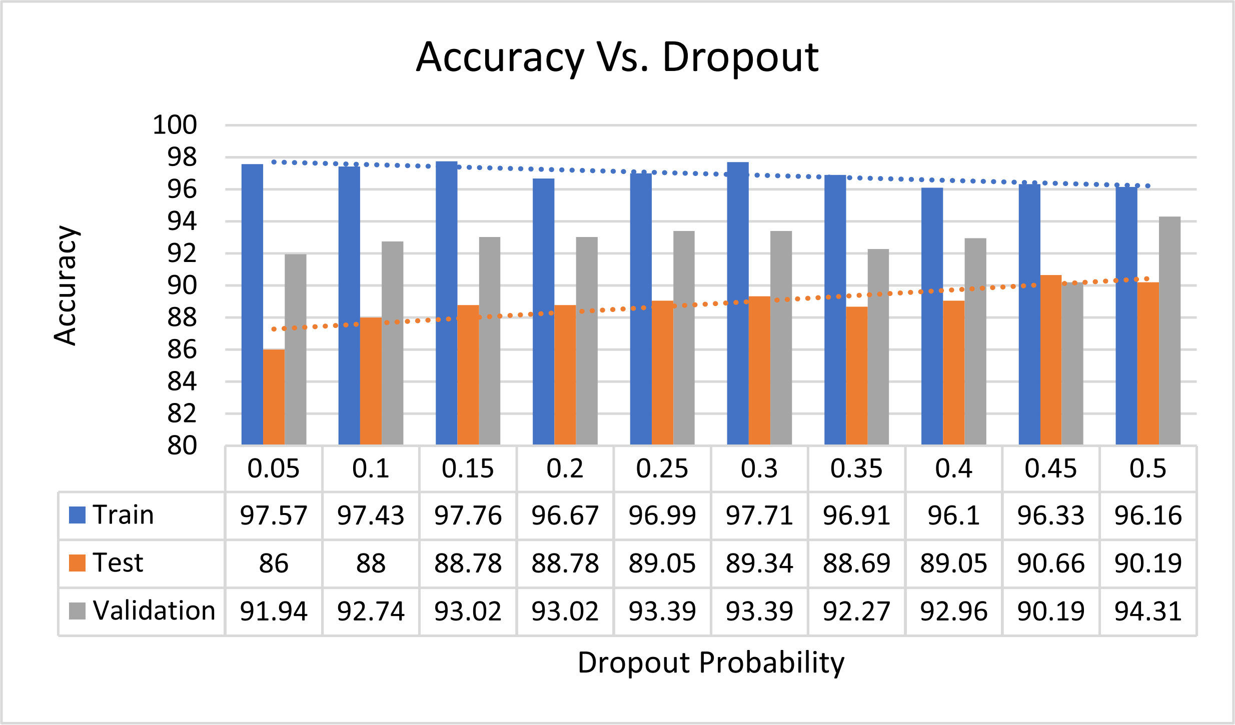
	0	1	2	3	4	5	6	7	8	9	10	11	12
0	165	6	7	0	0	1	0	0	1	0	0	0	0
1	2	173	0	0	0	1	0	0	0	2	1	1	0
2	2	0	174	0	0	0	0	0	0	2	0	2	0
3	0	0	1	163	0	1	11	0	0	0	0	4	0
4	0	0	0	0	172	3	0	0	2	0	0	2	1
5	1	0	3	0	0	167	0	6	1	0	0	2	0
6	6	0	1	2	0	1	149	20	1	0	0	0	0
7	0	0	1	2	0	0	5	170	2	0	0	0	0
8	0	0	6	0	0	0	0	2	167	0	3	2	0
9	0	2	2	0	0	0	0	1	4	131	40	0	0
10	1	3	3	0	0	1	0	1	1	16	151	3	0
11	0	0	1	0	0	7	1	0	0	0	0	171	0
12	1	0	0	0	0	0	0	0	0	0	0	0	179

0	அ
1	ஆ
2	இ
3	ஈ
4	உ
5	ஊ
6	எ
7	ஏ
8	ஐ
9	ஒ
10	ஓ
11	ஔ
12	ஃ

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	159	7	0	5	0	0	6	0	1	2	0	0	0	0	0	0	0	0
1	2	150	2	1	0	3	1	4	0	0	6	4	2	2	2	1	0	0
2	0	2	150	3	1	0	0	0	18	1	0	0	2	0	3	0	0	0
3	6	3	5	152	0	0	0	1	2	11	0	0	0	0	0	0	0	0
4	2	5	4	0	134	6	0	1	1	0	0	20	2	2	2	0	0	1
5	0	0	0	0	0	180	0	0	0	0	0	0	0	0	0	0	0	0
6	9	1	1	5	0	0	153	1	0	7	2	1	0	0	0	0	0	0
7	0	1	0	1	0	0	0	154	0	3	18	0	0	0	0	2	0	1
8	0	0	22	0	0	0	3	0	154	0	0	0	0	0	1	0	0	0
9	0	0	0	7	0	0	4	3	5	159	1	0	0	0	0	0	0	1
10	0	3	0	0	0	0	1	8	0	0	166	1	0	0	0	0	1	0
11	0	3	0	0	1	0	0	0	0	0	3	165	0	0	0	0	0	8
12	1	0	2	0	0	0	0	0	0	0	0	0	166	10	1	0	0	0
13	0	0	0	0	0	1	0	0	0	0	0	0	5	173	0	0	1	0
14	0	1	2	0	0	0	0	0	0	0	0	0	3	3	170	1	0	0
15	0	0	0	5	0	0	0	0	0	0	0	0	3	2	26	144	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	171	1
17	0	0	0	0	0	0	0	0	0	0	0	3	0	2	0	0	3	172

0	க்
1	க
2	கி
3	கீ
4	கு
5	கூ
6	ச்
7	ச
8	சி
9	சீ
10	சு
11	சூ
12	ங்
13	ங
14	ஙி
15	ஙீ
16	ஙு
17	ஙூ

FIGURE 19. Confusion matrices: Set of vowels (Left) and set of compound characters (Right).



	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
🟦 Train	97.57	97.43	97.76	96.67	96.99	97.71	96.91	96.1	96.33	96.16
🟧 Test	86	88	88.78	88.78	89.05	89.34	88.69	89.05	90.66	90.19
⬜ Validation	91.94	92.74	93.02	93.02	93.39	93.39	92.27	92.96	90.19	94.31

FIGURE 20. Dropout in the Dense layers.

It can be observed from both these results that the validation accuracy is getting better as the augmentation factor is increased. It can be for many reasons, such as the validation set becoming relatively easier compared to artificially induced variations in the training set. In addition, the validation set is evaluated using all neurons in contrast to the aggressively dropped neurons during the training phase. Furthermore, having a similar sample distribution for the validation set as that of the training set can also lead to good validation accuracy. The test set accuracy has reached 93.32% after data augmentation, which is about 3% increase. The training accuracy is not drastically coming down, which may be due to using moderate values for transformations while augmenting the training set. Nevertheless, this set of experiments demonstrated conventional ways to mitigate the high variance problem our model initially suffered.

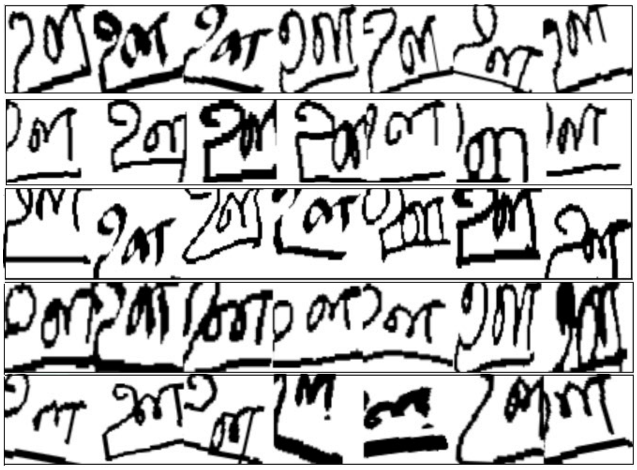


FIGURE 21. Sample augmented images belonging to a character [Row-1: Rotation, Row-2: Horizontal Shift, Row-3:Vertical Shift, Row-4:Zoom, Row-5:Multiple effects].

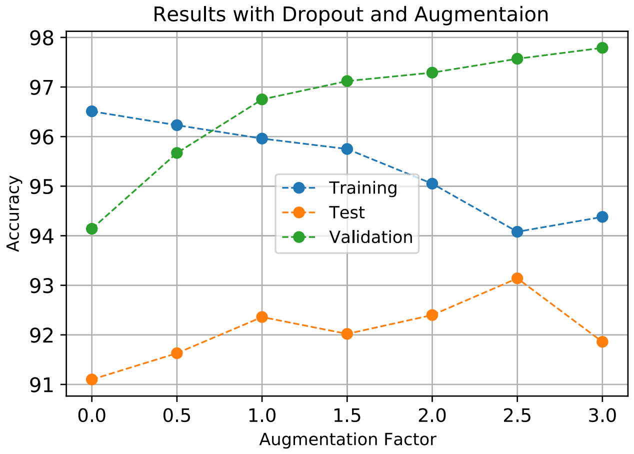
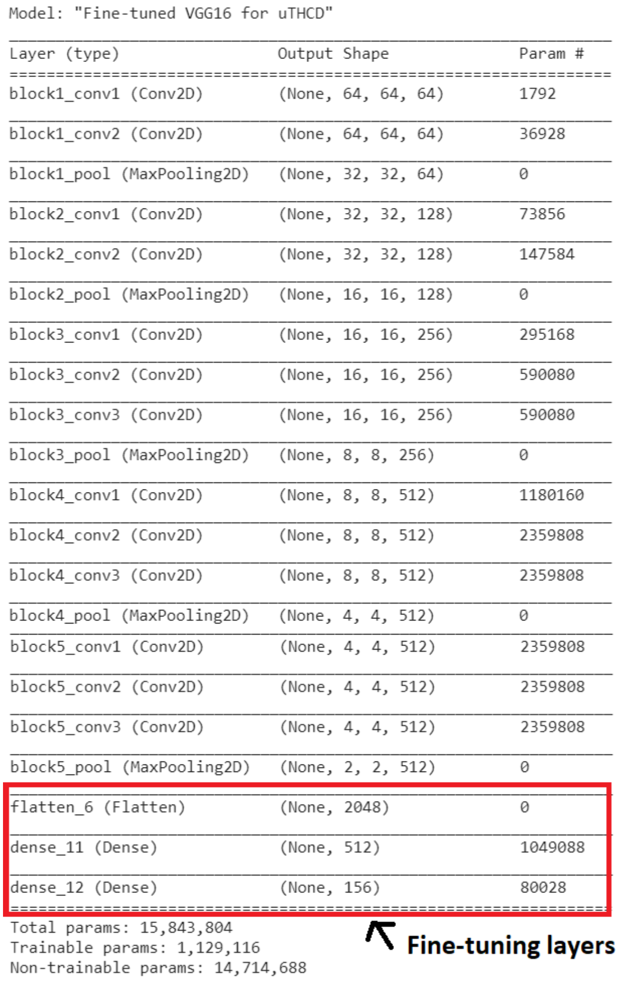


FIGURE 22. Train, test and validation accuracy for varying augmentation factors.

B. USING A FINE-TUNED CNN ARCHITECTURE – TRANSFER LEARNING

Finally, establishing a benchmark for the uTHCD dataset, especially using a CNN, would be incomplete without adopting the transfer-learning approach. Transfer learning is a process where we reuse a model for a task different than it was primarily learned [56]. Transfer learning offers several advantages such as i) offering a better starting point to the new task in hand, ii) deep learning of a model with comparatively lesser data, iii) rapid training process since it leverages a pre-trained model with optimal weights, iv) serves as a feature extraction model, etc [57]. Hence, we used a popular pre-trained model of the ImageNet challenge [58] and reported its performances with various configurations. This will serve as a future reference for efficacy comparisons of pre-trained models on the uTHCD database.

Inspired by the demonstrated performance of the VGG16 model recently on various tasks [52], [59], we leveraged the pre-trained VGG16 [47] model by fine-tuning it. First, all input samples were converted to 3D space (64 × 64 × 3) to comply with the VGG16 architecture specifications. Then, the VGG16 model was re-purposed by retaining the ImageNet weights and removing the top layer. We added two dense layers to fine-tune the model using the uTHCD database, as shown in Fig.23. This architecture produced the optimal results compared to the models based on a different combination of VGG16 layers, which included replacing just the top layer and considering only the first few convolutional blocks of VGG16. We carried out two types of experiments with the pre-trained model: with dropout and without dropout, as shown in Fig.24.



```
Model: "Fine-tuned VGG16 for uTHCD"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 64, 64, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 64, 64, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 32, 32, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 32, 32, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 32, 32, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 16, 16, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 16, 16, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 8, 8, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 8, 8, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_11 (Dense)             (None, 512)               1049088
_________________________________________________________________
dense_12 (Dense)             (None, 156)               80028
=================================================================
Total params: 15,843,804
Trainable params: 1,129,116
Non-trainable params: 14,714,688
```

FIGURE 23. The fine-tuned VGG16 model architecture.

Some observations based on this transfer learning experiment are as below:

1) The model required only around eight epochs to converge, which is around $\frac{1}{3}$ of the number of epochs needed to converge in the basic CNN model we proposed. This is due to the fact that we used the ImageNet kernel weights rather than training it from scratch. This is an advantage in terms of the training time of using a pre-trained model.

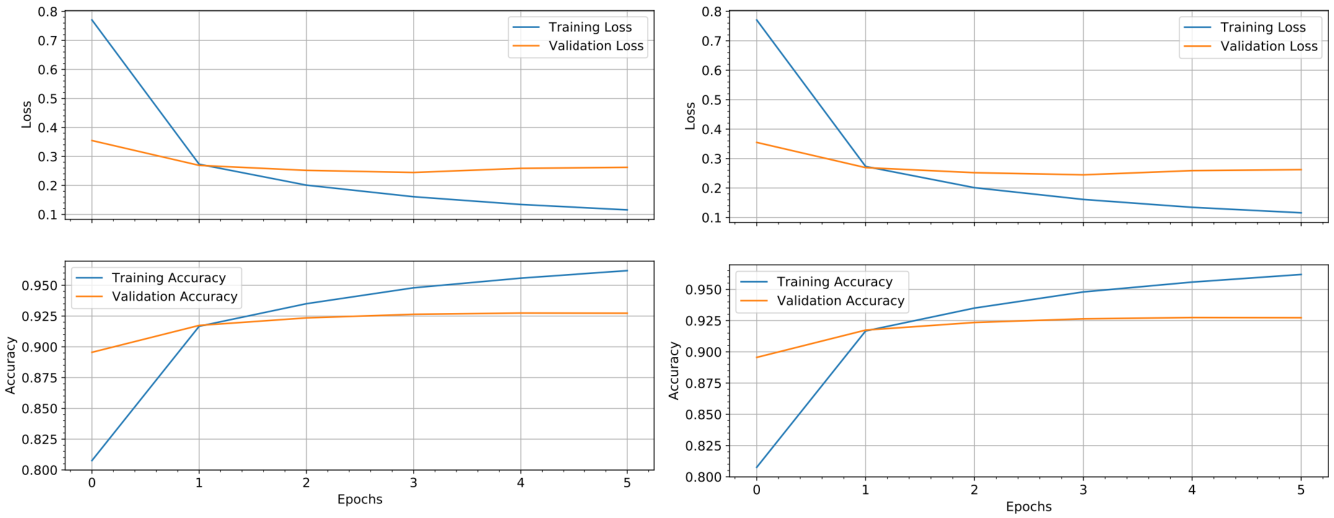


FIGURE 24. Result of fine tuned VGG16 model[Left:With Dropout, Right: Without Dropout].

TABLE 6. Established benchmark.

CNN Model Used	Test Accuracy	Train Accuracy	Val Accuracy	False Positive Rate (1-Specificity)	True Positive Rate (Sensitivity)	F1-Score	Correctly Classified Samples	Incorrectly Classified Samples
Basic Model	87.00	99.93	93.13	0.0008	0.8701	0.8703	24429	3651
Basic Model with Early Stopping	88.27	97.82	92.89	0.0008	0.8826	0.8829	24785	3295
Basic Model with Early Stopping and Dropout	91.10	96.51	94.46	0.0006	0.9109	0.9111	25582	2498
Basic Model with Early Stopping, Dropout and Augmentation	**93.16**	95.76	98.35	0.0004	0.9315	0.9314	26158	1922
Fine tuned VGG16 Model with Early Stopping	89.77	96.13	92.03	0.0007	0.8977	0.8978	25208	2872
Fine tuned VGG16 model With Early Stopping and Dropout	**92.32**	92.05	93.73	0.0004	0.9233	0.9232	25925	2155

2) The accuracy of the model on test data is 92.32% which is no better than the basic CNN architecture we used. However, the basic model required three times the augmented data to achieve this. The essence of using a pre-trained model lies in how effectively the model generalizes with lesser training samples. Hence, the transfer learning approach helped us achieve on par performance on test data without data augmentation.
3) The model without dropout rapidly overfits the data. This may be because the problem we are trying to solve is far simpler than the ImageNet challenge for which VGG16 weights were originally established.
4) Although the pre-trained model with dropout overcame the overfitting problem, its performance on training data is lesser than the basic CNN model we proposed. This is due to using the pre-trained weights with frozen layers. Setting those layers to trainable mode with a finely tuned learning rate would probably work best [60].

Table-6 summarizes the benchmark we established for the proposed uTHCD database using the various CNN models.

V. COMPARATIVE STUDY AND CLASSIFICATION RESULTS USING SOTA

In this section, we compare the proposed uTHCD database with the existing HPL database. We do this by first comparing the salient features of these two databases highlighting prominent features that potentially lead to better OCR accuracy. Furthermore, classification results using traditional machine learning and SOTA algorithm are analyzed.

A. SIGNIFICANCE OF THE PROPOSED uTHCD DATABASE

Compared to the HPL database, the uTHCD database has substantial benefits, mainly from OCR use-case perspectives:

1) Modern deep learning algorithms for OCR need a considerable amount of samples to develop a robust model [61]. The uHTCD database has 90950 samples with approximately 600 samples each from 156 classes with the provision to enhance more in the coming years.

This is around 13000 samples more compared to the existing HPL database.

2) Unlike the HPL dataset, the uTHCD database is a unified collection of both offline and online samples. As mentioned earlier, the offline samples capture useful characteristics of scanned handwritten samples such as distortion due to overwriting, variable thickness of strokes, and stroke discontinuity (due to usage of different writing tools). These are in addition to capturing various writing styles through online samples (Refer Fig.9). Hence, the OCR models created using the uHTCD database will capture these inherent characteristics of a scanned document; thereby, robust performance is expected for automatic form processing. The HPL dataset captures only different writing styles (Refer Fig.5).
3) As the HPL dataset is synthetically created from the online pen coordinates through interpolation technique [30], characters formed have single-pixel width strokes. On the other hand, the offline samples of the uTHCD database have instances of the same character having a variable thickness of strokes as we had no restriction on usage of writing media. Consequently, the percentage of foreground pixels is nearly 20% compared to approximately 2% in HPL data. This high degree of sparseness sometimes impedes the performance of some algorithms.
4) Each of the 156 classes has a uniform number of samples, whereas the HPL database has several under-represented classes. The machine learning based OCR algorithms can not robustly model under-represented classes [36].
5) The uTHCD database is made publicly available with multiple proportions of train-test splits, and thus it can be used to test the efficacy of traditional machine learning and modern deep learning algorithms without much processing of data.
6) The uTHCD database is available in both RAW and convenient HDF5 format, facilitating rapid OCR prototyping and model testing. In addition, our database does not require any processing and is available in a *ready-to-use* form. However, the HPL database needs some processing. For instance, the GT data for training samples are not provided. Hence, initial processing of HPL dataset is required at least with supervised algorithms.
7) The HPL dataset has some errors in the GT data. For instance, the test image 00043 is wrongly associated with GT class id 129 when it actually represents the class id 78. The uTHCD sample collection in the proposed work underwent two levels of manual verification processes. This ensured samples collected and associated GT are representative of the underlying class.
8) In the future, we will expand the uTHCD database to cater to the ever-evolving research in this field.

TABLE 7. Comparison of existing and proposed databases.

Comparison Factor	**HPL-ISO-TAMIL-CHAR**	**uTHCD**
Total Samples	77617	90950
Test set ground truth	✔	✔
Train set ground truth	✘	✔
Offline samples available	✘	✔
Publicly Available in HDF format	✘	✔
Uniform distribution of samples in each class	✘	✔
Processed database available for download	✘	✔
Possible future expansion	✘	✔

The table-7 provides a quick comparison between uTHCD and the HPL database.

B. CLASSIFICATION RESULTS USING SOTA

Next, we report the results of some traditional and SOTA classification algorithms using the proposed uTHCD and existing HPL datasets. The objective is to know the benefit of the proposed dataset over the HPL dataset. We implemented Random Forest (RF) [62], Support Vector Machines (SVM) [63], and k-Nearest Neighbor (KNN) [64] classification algorithms. The RF algorithm was implemented with 100 decision trees, and KNN models were evaluated by having the optimal K value. Results of these algorithms were presented using both raw pixel values and features extracted using a pre-trained VGG16 model. Additionally, we used the the fine-tuned VGG16 CNN architecture to observe the performance on respective test sets of the uTHCD and HPL datasets.

The accuracy of these algorithms using these two databases is shown in Fig.25.

Some crucial observations we made through this experiment are as below:

1) The results of RF, SVM, and KNN classifiers produced undesirable results using the HPL database. This problem is primarily due to the single-pixel width thickness of characters, leading to a high degree of sparseness in the HPL data.
2) During the bootstrap data aggregation step of the RF algorithm, the sparseness in image matrices leads to the highly correlated decision trees as they are randomly sampled. This will result in the formation of a weaker ensemble of decision trees.
3) A similar reasoning can be made for the performance of the KNN algorithm on HPL data. The KNN algorithm performed worst with raw pixel values. The reason could be again due to the single-pixel width character patterns. The algorithm will have difficulty matching a pattern with another sample, especially when these patterns are just single pixel width.
4) The SVM performs relatively better with raw pixel values compared to RF and KNN classifiers. This is

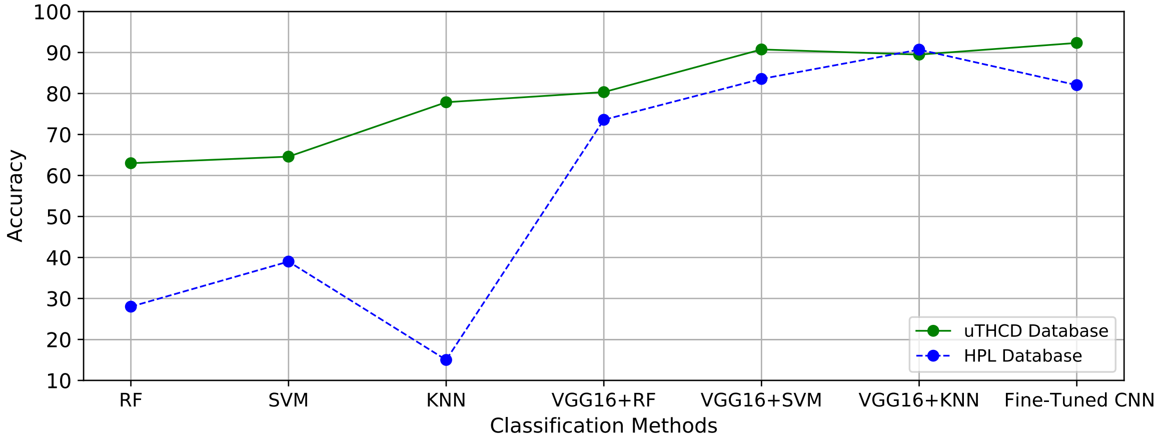


FIGURE 25. Results of traditional classification algorithms using the proposed and existing dataset.

because the SVM primarily maximizes the *margin*, and hence it depends on the distance between the different data points.

5) The points mentioned above can be substantiated through a simple transfer learning approach where the raw pixel values were transformed into a rich feature maps represented by the pre-trained VGG16 model [58]. The VGG16 model was trained on millions of images, and hence it captures complex features at different layers through its kernels. This is the main reason for the enhanced accuracy of these classifiers in transformed feature space.
6) The traditional classifiers RF, SVM, and KNN algorithms produced undesirable results if direct pixel values were used. This has a cascading effect; for instance, an ensemble of classifiers approach may fail to perform well due to the failure of constituent classifiers to model the data reliably [65].
7) Performance of these classifiers with the uTHCD database was acceptable. This is mainly because the sparseness is not substantial in the uTHCD database. Furthermore, as expected, the VGG16 feature space enhanced the classifiers' performance due to the possible elimination of redundancy expressed through raw pixel values.

This experiment suggests that the HPL dataset needs to be represented in a better feature space before a reliable classifier can be modeled using it.

Another experiment was conducted where we trained different classifiers with uTHCD samples but tested with the HPL dataset and vice-versa. The objective of this test was to determine the robustness of models against real-time unseen variations captured through the underlying training set.

We used the uTHCD_a set to be on par with the HPL test set size. To be uniform, we considered 26000 samples for testing from respective test sets. The results of RF, SVM, and KNN algorithms with transformed VGG16 feature maps were only considered since direct raw pixels yielded poor results when both trained and tested using the HPL dataset.

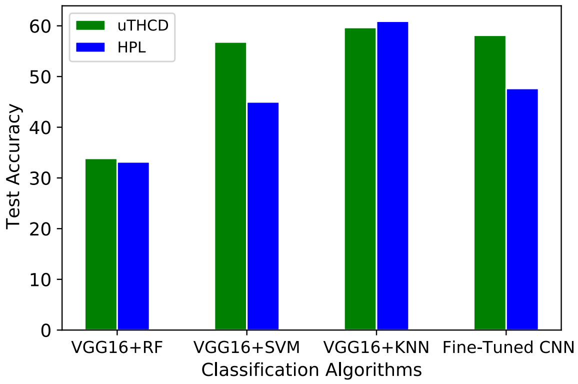


FIGURE 26. Performance of different classifiers when trained using uTHCD and HPL datasets.

The results are as shown in Fig.26. It can be ascertained that training on the proposed uTHCD database helps classifiers to perform either better or on par in contrast to when classifiers were trained using the HPL database. It is mainly because the uTHCD database includes both online and offline samples helping models to capture a wide variety of real-time variations. Furthermore, the online samples present in the uTHCD database allow classifiers to model the commonly exhibited HPL dataset variations. Hence, the performance on the HPL test was better when trained using the proposed database.

These experiments suggest that the state-of-the-art classification algorithms trained on the uTHCD database perform better on test samples never seen during the training phase. Furthermore, we believe that factors affecting handwritten patterns (diverse writing styles, usage of unconstrained writing tools, distortions, stroke discontinuity) are well captured through online and offline samples of the uTHCD. As a result, it can help achieve robust results for real-time OCR test cases as well.

Nevertheless, the existence of multiple standardized databases has several benefits. A majority of scientific study relies on the presence of standardized datasets for investigating a particular problem. It will facilitate research communities to focus on proposing solutions rather than going through laborious data collection tasks. Moreover, to test the efficiency of algorithms, it is essential to use multifaceted data to test all corner cases. In general, the corner cases do not come from one set of data but multiple datasets. Hence, both existing and proposed datasets lead to benchmarking new solutions for Tamil handwritten OCR, leading to robust language technologies involving speech, handwriting and text samples.

VI. CONCLUSION AND FUTURE WORK

Works relating to Tamil Handwritten character recognition have relied on a solitary standardized database for the longest time. However, having multiple standardized databases for a research problem can contribute positively in many ways to the advancement of that research field. In this work, we delineated the extensive work involved in creating a comprehensive handwritten Tamil isolated character database. This database consists of handwritten samples collected from both physical and digital forms. The database and necessary ground truth data have been made publicly available in both RAW and HDF5 formats. Our work presented many use cases for this dataset by applying CNN models for classification purposes and reporting the baseline accuracy. Additionally, we also ascertained that models trained on the proposed database achieve better OCR accuracy when tested on unseen data.

In the future, the proposed database can be augmented through the application of generative adversarial networks. The resulting expansion may pose more challenges to the ever-evolving feature extraction algorithms and deep learning framework. Various studies can be conducted to determine the fundamental efficacy/limitations of the state-of-the-art feature extraction and classification algorithms for Tamil OCR using the uTHCD database. Furthermore, it would be particularly interesting to see the application of the uTHCD database in the performance of character pattern overlapping using a pattern separation network [14].

The creation of this database is expected to contribute to the efficient model development for Tamil OCR, facilitate the research community to objectively evaluate the state-of-the-art algorithms, and instigate more research work in Tamil handwriting character recognition.

APPENDIX
HIERARCHICAL DATA FORMAT (HDF) FOR THE uTHDC DATABASE

This section demonstrates the HDF5 file structure of the uTHCD database and explains the method of extracting train, test, and validation sets from it using a simple Python script. While the code demonstrates extracting the uTHCD_a repository, the same is also applicable for the remaining repositories.

The HDF5 file is a convenient file format with.h5 extension to store and organize huge data [66]. The file structure used to organize the uTHCD database in this format is illustrated as shown in Fig.27. The file has a directory-like structure with two groups of datasets labeled as *Train data* and *Test data*. Within the *Train data* group, the size normalized 62870 uTHCD_a train images are stored in a 3D matrix, namely x_train, and the corresponding class labels are stored in the 1D vector y_train. The size of x_train and y_train are respectively (62870, 64, 64) and (62870,1). The 28080 test images and corresponding labels in the Test Data group are likewise stored respectively in x_test and y_test variables.

```
1 import numpy as np
2 import h5py
3 with h5py.File('../uTHCD_compressed.h5
    ↪ ','r') as hdf:
4   base_items = list(hdf.items())
5   print('Items in the Base Directory:'
    ↪ , base_items)
6   G1 = hdf.get('Train Data')
7   G1_items = list(G1.items())
8   print('\n Items in Group 1:',
    ↪ G1_items)
9   G2 = hdf.get('Test Data')
10   G2_items = list(G2.items())
11  print('Items in Group 2:', G2_items)
12 x_train = np.array(G1.get('x_train'))
13 y_train = np.array(G1.get('y_train'))
14 x_test = np.array(G2.get('x_test'))
15 y_test = np.array(G2.get('y_test'))
16 x_val = x_train[-7870:, :, :]
17 y_val = y_train[-7870:]
18 x_train = x_train[:-7870, :, :]
19 y_train = y_train[:-7870]
```

Listing. 1. Python Code to Extract uTHCD Dataset

Given this file structure, the Python code shown in Listing 1 to extract the train and test data is relatively straightforward. Lines 1-2 import two Python packages: NumPy and h5py. The NumPy package is necessary to represent the extracted train and test data as multidimensional arrays suitable for subsequent computational processes. The h5py module is utilized to extract the datasets from the HDF format file with the.h5 extension.

Line 3 reads the uTHCD compressed hdf5 file. Lines 4-5 extract the base group items and print their names ('Train Data' and 'Test Data'). Line 6 extracts the 'Train Data' group followed by Lines 7-8 that print the dataset within this group (x_train and y_train). Similarly, Line 9 and Lines 10-11 respectively extract the 'Test Data' group and print the dataset within it (x_test and y_test). Lines 12-15 respectively extract x_train, y_train, x_test, and y_test and represent them as multidimensional arrays. Once these datasets are extracted, Lines 16-19 extracts the validation set out of the training set.

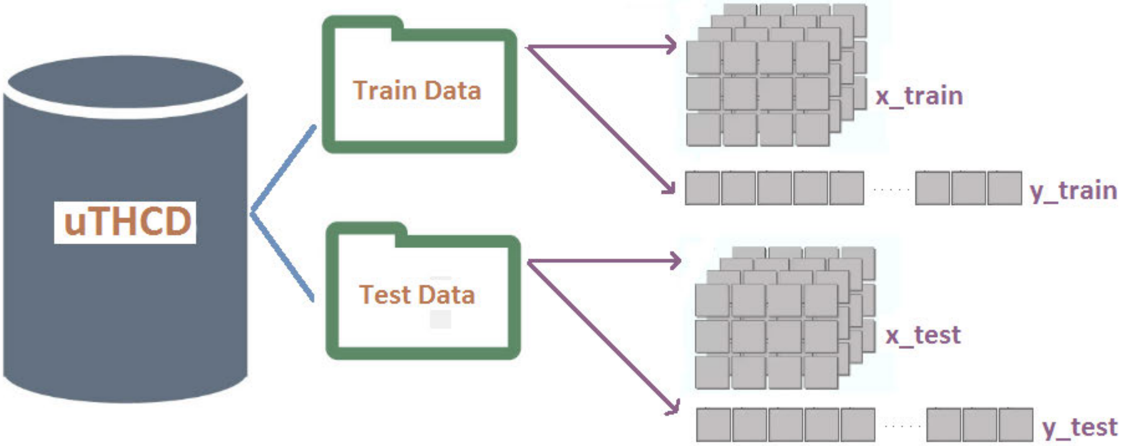
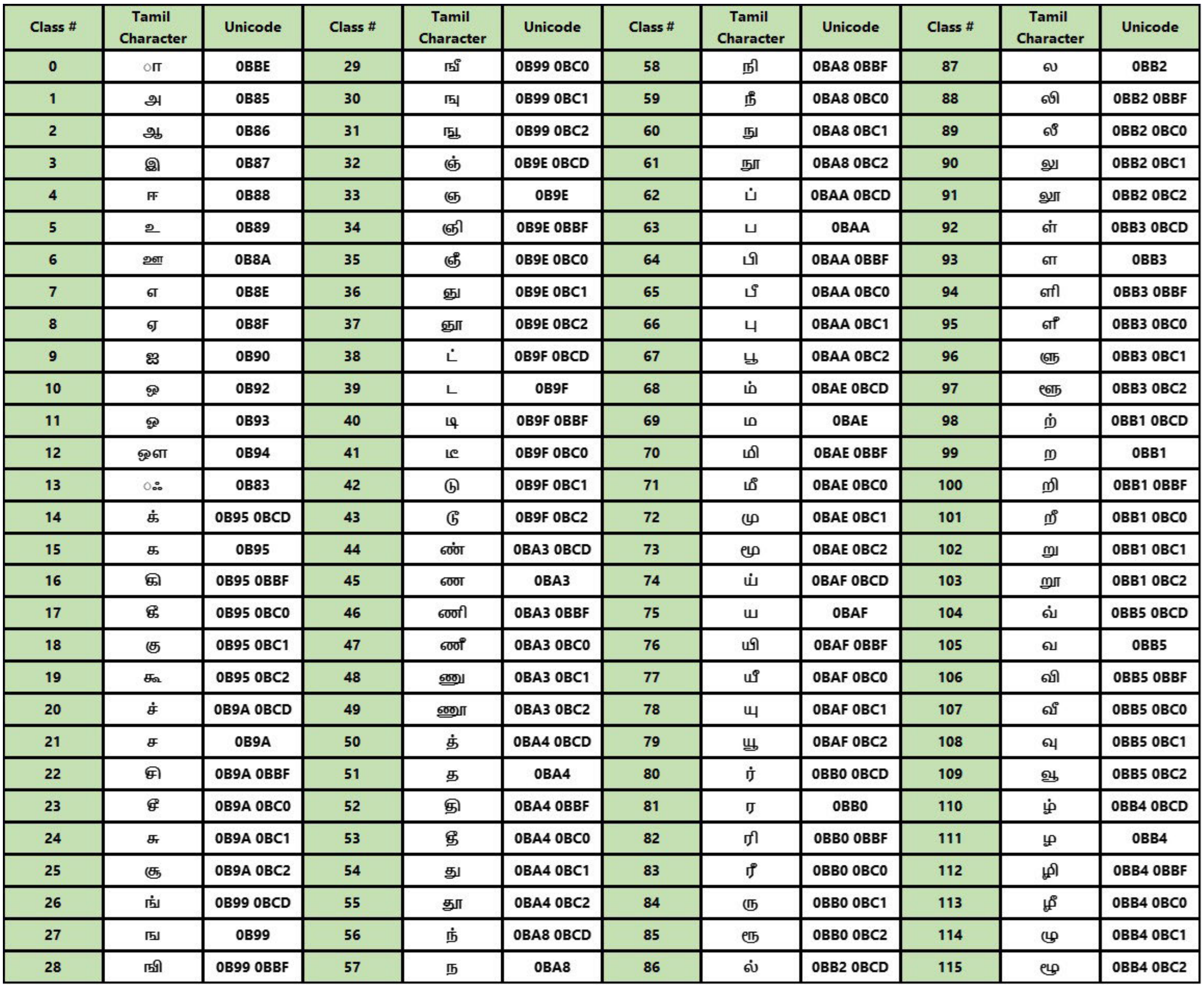


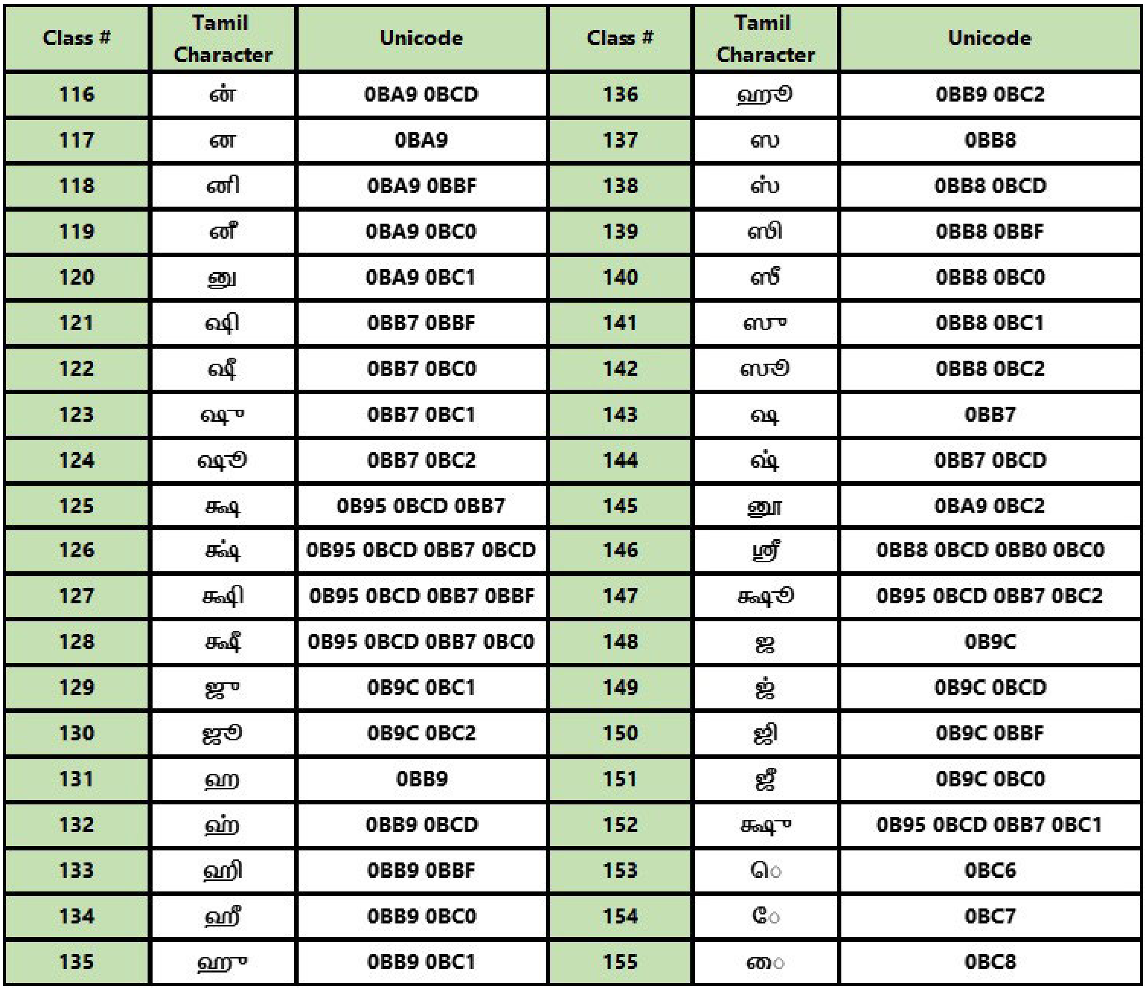
FIGURE 27. Data organization of uTHCD in HDF5 file format.



Class #	Tamil Character	Unicode	Class #	Tamil Character	Unicode	Class #	Tamil Character	Unicode	Class #	Tamil Character	Unicode
0	◌ா	0BBE	29	ஙீ	0B99 0BC0	58	நி	0BA8 0BBF	87	ல	0BB2
1	அ	0B85	30	ஙு	0B99 0BC1	59	நீ	0BA8 0BC0	88	லி	0BB2 0BBF
2	ஆ	0B86	31	ஙூ	0B99 0BC2	60	நு	0BA8 0BC1	89	லீ	0BB2 0BC0
3	இ	0B87	32	ஞ்	0B9E 0BCD	61	நூ	0BA8 0BC2	90	லு	0BB2 0BC1
4	ஈ	0B88	33	ஞ	0B9E	62	ப்	0BAA 0BCD	91	லூ	0BB2 0BC2
5	உ	0B89	34	ஞி	0B9E 0BBF	63	ப	0BAA	92	ள்	0BB3 0BCD
6	ஊ	0B8A	35	ஞீ	0B9E 0BC0	64	பி	0BAA 0BBF	93	ள	0BB3
7	எ	0B8E	36	ஞு	0B9E 0BC1	65	பீ	0BAA 0BC0	94	ளி	0BB3 0BBF
8	ஏ	0B8F	37	ஞூ	0B9E 0BC2	66	பு	0BAA 0BC1	95	ளீ	0BB3 0BC0
9	ஐ	0B90	38	ட்	0B9F 0BCD	67	பூ	0BAA 0BC2	96	ளு	0BB3 0BC1
10	ஒ	0B92	39	ட	0B9F	68	ம்	0BAE 0BCD	97	ளூ	0BB3 0BC2
11	ஓ	0B93	40	டி	0B9F 0BBF	69	ம	0BAE	98	ற்	0BB1 0BCD
12	ஔ	0B94	41	டீ	0B9F 0BC0	70	மி	0BAE 0BBF	99	ற	0BB1
13	◌ஃ	0B83	42	டு	0B9F 0BC1	71	மீ	0BAE 0BC0	100	றி	0BB1 0BBF
14	க்	0B95 0BCD	43	டூ	0B9F 0BC2	72	மு	0BAE 0BC1	101	றீ	0BB1 0BC0
15	க	0B95	44	ண்	0BA3 0BCD	73	மூ	0BAE 0BC2	102	று	0BB1 0BC1
16	கி	0B95 0BBF	45	ண	0BA3	74	ய்	0BAF 0BCD	103	றூ	0BB1 0BC2
17	கீ	0B95 0BC0	46	ணி	0BA3 0BBF	75	ய	0BAF	104	வ்	0BB5 0BCD
18	கு	0B95 0BC1	47	ணீ	0BA3 0BC0	76	யி	0BAF 0BBF	105	வ	0BB5
19	கூ	0B95 0BC2	48	ணு	0BA3 0BC1	77	யீ	0BAF 0BC0	106	வி	0BB5 0BBF
20	ச்	0B9A 0BCD	49	ணூ	0BA3 0BC2	78	யு	0BAF 0BC1	107	வீ	0BB5 0BC0
21	ச	0B9A	50	த்	0BA4 0BCD	79	யூ	0BAF 0BC2	108	வு	0BB5 0BC1
22	சி	0B9A 0BBF	51	த	0BA4	80	ர்	0BB0 0BCD	109	வூ	0BB5 0BC2
23	சீ	0B9A 0BC0	52	தி	0BA4 0BBF	81	ர	0BB0	110	ழ்	0BB4 0BCD
24	சு	0B9A 0BC1	53	தீ	0BA4 0BC0	82	ரி	0BB0 0BBF	111	ழ	0BB4
25	சூ	0B9A 0BC2	54	து	0BA4 0BC1	83	ரீ	0BB0 0BC0	112	ழி	0BB4 0BBF
26	ங்	0B99 0BCD	55	தூ	0BA4 0BC2	84	ரு	0BB0 0BC1	113	ழீ	0BB4 0BC0
27	ங	0B99	56	ந்	0BA8 0BCD	85	ரூ	0BB0 0BC2	114	ழு	0BB4 0BC1
28	ஙி	0B99 0BBF	57	ந	0BA8	86	ல்	0BB2 0BCD	115	ழூ	0BB4 0BC2

FIGURE 28. Character to class mapping with unicode.

We can use the same code to extract the rest of the uTHCD repositories of varying train-test split proportions. Additional details to convert the classification result of an input image to corresponding editable Tamil character can be done using Unicode character encoding of Tamil script as shown in Fig.28 and Fig.29.



Class #	Tamil Character	Unicode	Class #	Tamil Character	Unicode
116	ன்	0BA9 0BCD	136	ஹூ	0BB9 0BC2
117	ன	0BA9	137	ஸ	0BB8
118	னி	0BA9 0BBF	138	ஸ்	0BB8 0BCD
119	னீ	0BA9 0BC0	139	ஸி	0BB8 0BBF
120	னு	0BA9 0BC1	140	ஸீ	0BB8 0BC0
121	ஷி	0BB7 0BBF	141	ஸு	0BB8 0BC1
122	ஷீ	0BB7 0BC0	142	ஸூ	0BB8 0BC2
123	ஷு	0BB7 0BC1	143	ஷ	0BB7
124	ஷூ	0BB7 0BC2	144	ஷ்	0BB7 0BCD
125	க்ஷ	0B95 0BCD 0BB7	145	னூ	0BA9 0BC2
126	க்ஷ்	0B95 0BCD 0BB7 0BCD	146	ஸ்ரீ	0BB8 0BCD 0BB0 0BC0
127	க்ஷி	0B95 0BCD 0BB7 0BBF	147	க்ஷூ	0B95 0BCD 0BB7 0BC2
128	க்ஷீ	0B95 0BCD 0BB7 0BC0	148	ஜ	0B9C
129	ஜு	0B9C 0BC1	149	ஜ்	0B9C 0BCD
130	ஜூ	0B9C 0BC2	150	ஜி	0B9C 0BBF
131	ஹ	0BB9	151	ஜீ	0B9C 0BC0
132	ஹ்	0BB9 0BCD	152	க்ஷு	0B95 0BCD 0BB7 0BC1
133	ஹி	0BB9 0BBF	153	ெ◌	0BC6
134	ஹீ	0BB9 0BC0	154	ே◌	0BC7
135	ஹு	0BB9 0BC1	155	ை◌	0BC8

FIGURE 29. Character to class mapping with unicode – *continued.*

ACKNOWLEDGMENT

The authors would like to profoundly thank reviewers for their critical reading and thoughtful comments, which significantly improved the overall quality of the work. They would also like to thank Dr. Mufti Mahmud, Nottingham Trent University, for several resourceful discussions during this research study.

REFERENCES

[1] I. Mahadevan, *Early Tamil Epigraphy From the Earliest Times to the Sixth Century AD* (H. O. Series), vol. 62. Cambridge, U.K.: Harvard Univ. Press, 2003.

[2] S. B. Steever, "Tamil writing," in *The World's Writing Systems*. New York, NY, USA: Oxford Univ. Press, 1996, pp. 426–430.

[3] L. Campbell and V. Grondona, "Review: Ethnologue: Languages of the world by Raymond G. Gordon," *Language*, vol. 84, no. 3, pp. 636–641, 2008.

[4] P. B. Pati and A. G. Ramakrishnan, "OCR in Indian scripts: A survey," *IETE Tech. Rev.*, vol. 22, no. 3, pp. 217–227, May 2005.

[5] U. Pal and B. B. Chaudhuri, "Indian script character recognition: A survey," *Pattern Recognit.*, vol. 37, no. 9, pp. 1887–1899, Sep. 2004.

[6] A. Chaudhuri, K. Mandaviya, P. Badelia, and S. K. Ghosh, "Optical character recognition systems," in *Optical Character Recognition Systems for Different Languages With Soft Computing* (Studies in Fuzziness and Soft Computing), vol. 352. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-50252-6_2.

[7] G. Siromoney, R. Chandrasekaran, and M. Chandrasekaran, "Computer recognition of printed Tamil characters," *Pattern Recognit.*, vol. 10, no. 4, pp. 243–247, Jan. 1978.

[8] M. Chandrasekaran, R. Chandrasekaran, and G. Siromoney, "Context dependent recognition of handprinted Tamil characters," in *Proc. Int. Conf. Syst. Man Cybern.*, vol. 2, 1983, pp. 786–790.

[9] C. Vinotheni, S. L. Pandian, and G. Lakshmi, "Modified convolutional neural network of tamil character recognition," in *Advances in Distributed Computing and Machine Learning* (Lecture Notes in Networks and Systems), vol. 127, A. Tripathy, M. Sarkar, J. Sahoo, K. C. Li, and S. Chinara, Eds. Singapore: Springer, 2021, doi: 10.1007/978-981-15-4218-3_46.

[10] R. B. Lincy and R. Gayathri, "Optimally configured convolutional neural network for Tamil handwritten character recognition by improved lion optimization model," *Multimedia Tools Appl.*, vol. 80, no. 4, pp. 5917–5943, Feb. 2021.

[11] H. Choudhury and S. R. M. Prasanna, "Handwriting recognition using sinusoidal model parameters," *Pattern Recognit. Lett.*, vol. 121, pp. 87–96, Apr. 2019.

[12] T. M. Ghanim, M. I. Khalil, and H. M. Abbas, "Comparative study on deep convolution neural networks DCNN-based offline arabic handwriting recognition," *IEEE Access*, vol. 8, pp. 95465–95482, 2020.

[13] *Current State of OCR: Is it a Solved Problem in 2021?* Accessed: May 4, 2021. [Online]. Available: https://research.aimultiple.com/ocr-technology/

[14] N. Modhej, A. Bastanfard, M. Teshnehlab, and S. Raiesdana, "Pattern separation network based on the hippocampus activity for handwritten recognition," *IEEE Access*, vol. 8, pp. 212803–212817, 2020.

[15] A. Singh, K. Bacchuwar, and A. Bhasin, "A survey of OCR applications," *Int. J. Mach. Learn. Comput.*, vol. 2, no. 3, p. 314, 2012.

[16] X.-F. Yue, S.-X. Jiao, L.-Q. Han, and H.-Z. LI, "Survey of OCR and its applications in pattern recognition," *Hebei J. Ind. Sci. Technol.*, vol. 23, no. 5, p. 312, 2006.

[17] X. Peng, H. Cao, S. Setlur, V. Govindaraju, and P. Natarajan, ''Multilingual OCR research and applications: An overview,'' in *Proc. 4th Int. Workshop Multilingual (OCR-MOCR)*, 2013, pp. 1–8.

[18] R. Sharma and B. Kaushik, ''Offline recognition of handwritten Indic scripts: A state-of-the-art survey and future perspectives,'' *Comput. Sci. Rev.*, vol. 38, Nov. 2020, Art. no. 100302.

[19] M. Agrawal, A. S. Bhaskarabhatla, and S. Madhvanath, ''Data collection for handwriting corpus creation in Indic scripts,'' in *Proc. Int. Conf. Speech Lang. Technol. Oriental COCOSDA (ICSLT-COCOSDA)*, New Delhi, India, Nov. 2004.

[20] A. Alaei, P. Nagabhushan, and U. Pal, ''A benchmark Kannada handwritten document dataset and its segmentation,'' in *Proc. Int. Conf. Document Anal. Recognit.*, Sep. 2011, pp. 141–145.

[21] N. Das, J. M. Reddy, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, ''A statistical–topological feature combination for recognition of handwritten numerals,'' *Appl. Soft Comput.*, vol. 12, no. 8, pp. 2486–2495, Aug. 2012.

[22] P. K. Singh, R. Sarkar, N. Das, S. Basu, M. Kundu, and M. Nasipuri, ''Benchmark databases of handwritten Bangla-Roman and Devanagari-Roman mixed-script document images,'' *Multimedia Tools Appl.*, vol. 77, no. 7, pp. 8441–8473, Apr. 2018.

[23] N. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, ''A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application,'' *Appl. Soft Comput.*, vol. 12, no. 5, pp. 1592–1606, May 2012.

[24] N. Das, S. Basu, R. Sarkar, M. Kundu, M. Nasipuri, and D. K. Basu, ''An improved feature descriptor for recognition of handwritten Bangla alphabet,'' 2015, *arXiv:1501.05497*. [Online]. Available: http://arxiv.org/abs/1501.05497

[25] N. Das, K. Acharya, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri, ''A benchmark image database of isolated Bangla handwritten compound characters,'' *Int. J. Document Anal. Recognit.*, vol. 17, no. 4, pp. 413–431, Dec. 2014.

[26] N. Das, R. Sarkar, S. Basu, P. K. Saha, M. Kundu, and M. Nasipuri, ''Handwritten bangla character recognition using a soft computing paradigm embedded in two pass approach,'' *Pattern Recognit.*, vol. 48, no. 6, pp. 2054–2071, Jun. 2015.

[27] R. Sarkar, N. Das, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, ''CMATERdb1: A database of unconstrained handwritten Bangla and Bangla–English mixed script document image,'' *Int. J. Document Anal. Recognit.*, vol. 15, no. 1, pp. 71–83, Mar. 2012.

[28] U. Bhattacharya and B. B. Chaudhuri, ''Databases for research on recognition of handwritten characters of Indian scripts,'' in *Proc. 8th Int. Conf. Document Anal. Recognit. (ICDAR)*, 2005, pp. 789–793.

[29] K. S. Dash, N. B. Puhan, and G. Panda, ''Odia character recognition: A directional review,'' *Artif. Intell. Rev.*, vol. 48, no. 4, pp. 473–497, Dec. 2017.

[30] *The HPL Isolated Handwritten Tamil Character Dataset*. Accessed: May 4, 2021 [Online]. Available: http://lipitk.sourceforge.net/datasets/tamilchardata.htm

[31] F. Hajamohideen and S. Noushath, ''Kalanjiyam: Unconstrained offline tamil handwritten database,'' in *Computer Vision, Graphics, and Image Processing* (Lecture Notes in Computer Science), vol. 10481, S. Mukherjee *et al.*, Eds. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-68124-5_24.

[32] B. R. Kavitha and C. Srimathi, ''Benchmarking on offline handwritten Tamil character recognition using convolutional neural networks,'' *J. King Saud Univ. Comput. Inf. Sci.*, Jun. 2019, doi: 10.1016/j.jksuci.2019.06.004.

[33] R. Kunwar and A. G. Ramakrishnan, ''Online handwriting recognition of Tamil script using fractal geometry,'' in *Proc. Int. Conf. Document Anal. Recognit.*, Sep. 2011, pp. 1389–1393.

[34] A. G. Ramakrishnan and K. B. Urala, ''Global and local features for recognition of online handwritten numerals and Tamil characters,'' in *Proc. 4th Int. Workshop Multilingual (OCR MOCR)*, 2013, pp. 1–5.

[35] P. Vijayaraghavan and M. Sra, ''Handwritten Tamil recognition using a convolutional neural network,'' in *Proc. Int. Conf. Inf., Commun., Eng. Technol. (ICICET)*, 2014, pp. 1–4.

[36] G. Strang, *Linear Algebra and Learning From Data*. London, U.K.: Wellesley, 2019.

[37] P. Banumathi and G. M. Nasira, ''Handwritten Tamil character recognition using artificial neural networks,'' in *Proc. Int. Conf. Process Autom., Control Comput.*, Jul. 2011, pp. 1–5.

[38] S. Kowsalya and P. S. Periasamy, ''Recognition of Tamil handwritten character using modified neural network with aid of elephant herding optimization,'' *Multimedia Tools Appl.*, vol. 78, no. 17, pp. 25043–25061, Sep. 2019.

[39] T. MJose and A. Wahi, ''Recognition of Tamil handwritten characters using Daubechies wavelet transforms and feed-forward backpropagation network,'' *Int. J. Comput. Appl.*, vol. 64, no. 8, pp. 26–29, Feb. 2013.

[40] N. Shanthi and K. Duraiswamy, ''A novel SVM-based handwritten Tamil character recognition system,'' *Pattern Anal. Appl.*, vol. 13, no. 2, pp. 173–180, May 2010.

[41] S. Thadchanamoorthy, N. D. Kodikara, H. L. Premaretne, U. Pal, and F. Kimura, ''Tamil handwritten city name database development and recognition for postal automation,'' in *Proc. 12th Int. Conf. Document Anal. Recognit.*, Aug. 2013, pp. 793–797.

[42] B. Nethravathi, C. Archana, K. Shashikiran, A. G. Ramakrishnan, and V. Kumar, ''Creation of a huge annotated database for Tamil and Kannada OHR,'' in *Proc. 12th Int. Conf. Frontiers Handwriting Recognit.*, Nov. 2010, pp. 415–420.

[43] S. A. Mahmoud, I. Ahmad, W. G. Al-Khatib, M. Alshayeb, M. T. Parvez, V. Märgner, and G. A. Fink, ''KHATT: An open Arabic offline handwritten text database,'' *Pattern Recognit.*, vol. 47, no. 3, pp. 1096–1112, 2014.

[44] P. Dey and S. Noushath, ''E-PCP: A robust skew detection method for scanned document images,'' *Pattern Recognit.*, vol. 43, no. 3, pp. 937–948, Mar. 2010.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ''ImageNet classification with deep convolutional neural networks,'' in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25, Dec. 2012, pp. 1097–1105.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, ''Going deeper with convolutions,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[47] K. Simonyan and A. Zisserman, ''Very deep convolutional networks for large-scale image recognition,'' 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[48] F. Chollet, ''Xception: Deep learning with depthwise separable convolutions,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[49] D. M. W. Powers, ''Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation,'' 2020, *arXiv:2010.16061*. [Online]. Available: http://arxiv.org/abs/2010.16061

[50] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, ''A state-of-the-art survey on deep learning theory and architectures,'' *Electron*, vol. 8, no. 3, p. 292, Mar. 2019.

[51] Y. Bengio, ''Practical recommendations for gradient-based training of deep architectures,'' in *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science), vol. 7700, G. Montavon, G. B. Orr, and K. R. Müller, Eds. Berlin, Germany: Springer, 2012, doi: 10.1007/978-3-642-35289-8_26.

[52] S. Singh, A. Sharma, and V. K. Chauhan, ''Online handwritten Gurmukhi word recognition using fine-tuned deep convolutional neural network on offline features,'' *Mach. Learn. Appl.*, vol. 5, Sep. 2021, Art. no. 100037.

[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, ''Dropout: A simple way to prevent neural networks from overfitting,'' *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[54] S. Park and N. Kwak, ''Analysis on the dropout effect in convolutional neural networks,'' in *Computer Vision—ACCV 2016* (Lecture Notes in Computer Science), vol. 10112, S. H. Lai, V. Lepetit, K. Nishino, and Y. Sato, Eds. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-54184-6_12.

[55] C. Shorten and T. M. Khoshgoftaar, ''A survey on image data augmentation for deep learning,'' *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019.

[56] M. A. Ko and S. Poruran, ''OCR-Nets: Variants of pre-trained CNN for urdu handwritten character recognition via transfer learning,'' *Procedia Comput. Sci.*, vol. 171, pp. 2294–2301, Jan. 2020.

[57] L. Torrey and J. Shavlik, ''Transfer learning,'' in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, methods, and Techniques*. Hershey, PA, USA: IGI Global, 2010, pp. 242–264.

[58] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, ''ImageNet large scale visual recognition challenge,'' *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[59] M. Shorfuzzaman and M. S. Hossain, ''MetaCOVID: A Siamese neural network framework with contrastive loss for n-shot diagnosis of COVID-19 patients,'' *Pattern Recognit.*, vol. 113, May 2021, Art. no. 107700.

[60] S. Kornblith, J. Shlens, and Q. V. Le, "Do better ImageNet models transfer better?" in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2661–2671.

[61] C. Boufenar, A. Kerboua, and M. Batouche, "Investigation on deep learning for off-line handwritten Arabic character recognition," *Cognit. Syst. Res.*, vol. 50, pp. 180–195, Aug. 2018.

[62] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, vol. 1, Aug. 1995, pp. 278–282.

[63] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, Jul./Aug. 2008.

[64] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.

[65] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, Sep. 2006.

[66] *The HDF5 Library and File Format*. Accessed: May 31, 2021. [Online]. Available: https://www.hdfgroup.org/solutions/hdf5/

NOUSHATH SHAFFI received the B.Sc., master's, and Ph.D. degrees in image processing from the University of Mysore, in 2001, 2004, and 2008, respectively.

After his Ph.D., he joined HP Labs, India, where he worked predominantly on document image processing. His topic of research was face recognition. Prior to joining CAS, Sohar, in 2009, he has been working as a Research Consultant with Hewlett Packard Laboratories, India, since 2007. He is currently working as an Assistant Professor at the University of Technology and Applied Sciences (UTAS), Oman. He is also the Head of the Research Department and Technology Transfer Office at UTAS. He was part of a team which successfully developed and implemented an algorithm in the hardware platform to automatically skew correct the document. He has over ten (reputed) publications in all areas of interest and three U.S. patents have been published. His research interests include biometrics, document analysis, and application of subspace algorithms for image recognition. He received several research grants from the Research Council of Oman and recipient of the National Research Award for the best undergraduate project mentored in the ICT sector.

FAIZAL HAJAMOHIDEEN received the Ph.D. degree in 2017. He is currently a Lecturer with the Department of Information Technology, University of Technology and College of Applied Sciences (UTAS), Sohar, Oman. He has over 20 years of experience in research and education field. He has gained industrial oriented certificates in networking and operating systems. He has designed, developed, and delivered technical training in various subjects such as networking, cloud computing, cybersecurity, and big data analysis. He has led many research projects in the subject area. He has acquired funded research projects, such as URG-TRC, Oman, and OCCI, Oman. He has good knowledge about Linux-based SDN, Big Data Hadoop, NS3, and Python3 tools for research. He was involved in the Research Council (TRC), Oman, sponsored image processing and machine learning and 4G workshops. He has cleared a certificate with intellectual property sponsored by UTAS. He has published articles in national and international journals. He has coauthored and published in international journals. His areas of interests are image processing, deep learning, cloud computing, and hybrid networks. He is an Active Member of the Scientific Research Department (SRD) and representing Information Technology Department, UTAS. He has participated and presented IEEE and IET conferences.