ISSN: 0067-2904

ADL: A New Dataset of Select Arabic-Derived Letters for Handwritten Character Recognition

Mouhssine EL ATILLAH
Computer Systems Engineering, Mathematics and Applications (ISIMA), Polydisciplinary faculty of Taroudant, University Ibn Zohr, Troudant, Morocco

Received: 1/12/2024 Accepted: 28/7/2025 Published: 30/10/2025

Abstract

Arabic text and characters recognition are among the most challenging problems in the field of optical character recognition (OCR) due to the complex nature of the letters and their variance in forms. This paper presents a new ADL dataset of the Arabic derived letters: Che (چ), Ngain (ڠ), Pe (پ), Ve (ڤ), Zhe (ژ), and the three versions of Gaf (ڭ ,گ, and ک). This dataset consists of 55,440 images from scanned handwritten papers made by 30 participants of different ages, thus ensuring demographic and stylistic diversity. The dataset has been evaluated on three different models: a Support Vector Machine (SVM), a Convolutional Neural Network (CNN), and a Vision Transformer (ViT) to demonstrate its practical usability. Several forms for different alphabet positions are used to represent each letter: isolated, initial, medial, and final, to reflect its real-world usage, which increases the value of the dataset for machine learning applications. The dataset was enriched using data augmentation techniques based on random rotation, horizontal shift, and zooming with nearest neighbor interpolation to fill empty pixels, which allowed for representing each character in a balanced way while preserving the essential structural elements. The dataset, as the first structured resource on some derived letters of the Arabic language, aims to fill a crucial gap in datasets focused on Arabic script and to advance research on handwritten character recognition. This dataset has important implications for linguistic research, practical applications, and advances in automated text processing systems by supporting better recognition of these non-original derived letters, especially in optical character recognition (OCR) systems.

Keywords: Arabic handwritten characters, ADL dataset, Arabic derived letters, Optical character recognition, Character recognition, SVM, CNN, ViT.

ADL: مجموعة بيانات جديدة لبعض الحروف المشتقة من اللغة العربية للمساهمة في التعرف على الحروف المكتوبة بخط اليد

محسن العاطي الله

هندسة أنظمة الكمبيوتر والرياضيات والتطبيقات ***(ISIMA)***، الكلية متعددة التخصصات بتارودانت، جامعة ابن زهر، تارودانت، المغرب

*Email: m.elatillah@uiz.ac.ma

الخلاصة

يُعد التعرف على النص والحروف العربية من بين أكثر المشكلات صعوبة في مجال التعرف الضوئي على الحروف (OCR) بسبب الطبيعة المعقدة للحروف وتباين أشكالها. تقدم هذه الورقة مجموعة بيانات ADL جديدة من الحروف المشتقة من العربية: Che (چ)، Ngain (ڠ)، Pe (پ)، Ve (ڤ) ، Zhe (ژ) والإصدارات الثلاثة من الحرف Gaf (گ ، ڪ ، و ک). تتكون مجموعة البيانات هذه من 55440 صورة ممسوحة ضوئيًا ومأخوذة من أوراق مكتوبة بخط اليد من قبل 30 مشاركًا من مختلف الأعمار، مما يضمن التنوع الديموغرافي والأسلوبي. تم تقييم مجموعة البيانات بثلاثة نماذج مختلفة، آلة المتجهات الداعمة (SVM)، شبكة عصبية تلافيفية (CNN)، ومحول الرؤية (ViT) لإظهار قابليتها للاستخدام العملي. اعتُمدت نمادج كتابة كل حرف حسب موضعه من الكلمة: معزول، في البداية، في الوسط وفي النهاية، من أجل عكس الاستخدام الحقيقي للحروف، مما يزيد من قيمة مجموعة البيانات لتطبيقات التعلم الآلي. تم تضخيم مجموعة البيانات باستخدام تقنيات تكبير البيانات بناءً على الدوران العشوائي والتحول الأفقي والتكبير مع الاستيفاء بأقرب جار لملء وحدات البكسل الفارغة، والتي سمحت بتمثيل كل حرف بطريقة متوازنة مع الحفاظ على العناصر الهيكلية الأساسية. تهدف مجموعة البيانات، كأول مورد منظم لبعض الحروف المشتقة من العربية، إلى ملء فجوة جوهرية في مجموعات البيانات التي تركز على النص العربي والتقدم في الأبحاث حول التعرف على الحروف المكتوبة بخط اليد. تعود مجموعة البيانات هذه بنفع بيّن على البحث اللغوي والتطبيقات العملية والتقدم في أنظمة معالجة النصوص الآلية من خلال دعم التعرف بشكل أفضل على هذه الحروف المشتقة غير الأصلية ، وخاصة في أنظمة التعرف الضوئي على الحروف (OCR).

1. Introduction

Arabic is the fifth most spoken language worldwide by approximately 334,800,000 people [1], [2]. The original Arabic alphabet is fixed 28 letters. There are many languages that use Arabic letters in their original languages. These languages contain some sounds that do not exist in the original Arabic language. So, they have made some small modifications to the Arabic alphabet at the morphological level to overcome this absence. [3], [4]. Because of these modifications, these letters take the name of "Arabic derived letters" according to the shape similarity to the original Arabic alphabet. To pronounce the intrusive words (words that come from other languages) in the Arabic language correctly, we can consider the derived letters as informal additions to the Arabic characters to represent sounds that do not exist in Arabic phonology [5]. Persian, Urdu, and Kurdish are among the languages that have relied on Arabic derived letters to meet their specific phonetic needs [6], [7]. Usually, the derived letters are created by adding under or above dots, or other signs to the existing Arabic alphabet that are close to the derived sound. Persian, for example, represents the new sound /p/ that does not exist in Arabic with the letter پ (pe) derived from the Arabic letter ب (ba) by adding two additional dots under the letter, which helps distinguish the two sounds [8]. Arabic and other languages are able to expand their phonetic capabilities through this adaptation process. Those letters also preserve the structure and appearance of the Arabic manuscripts. However, the Arabic derived letters are not standardized; they are used in many languages and vary across linguistic and regional backgrounds. When it comes to artificial intelligence, especially in the field of optical character recognition, access to datasets that highlight the Arabic derived letters is essential for researchers who are interested in Arabic manuscripts. To overcome this gap, we deal with the ADL dataset in this paper, which is a novel resource of handwriting samples of Arabic-derived letters, focusing on ڠ, گ, ڤ, ژ, چ and پ letters [9]. These characters play a main role in the recognition of handwritten characters in their main languages, and also in the Arabic language as an intrusive alphabet. This dataset can play a key role in studies on handwritten manuscript recognition in languages that use letters derived from Arabic (classification of these letters in languages like Urdu, Persian, etc.) and also in the Arabic language. According to these particular

characteristics, ADL dataset is used to simplify the creation of more accurate and stronger recognition systems. This can help to integrate intrusive words into the Arabic language by keeping their original sounds, and also to recognize the handwritten characters in other languages.

The rest of this article is organized as follows: Section 2 presents the related work. Section 3 describes in detail the main contribution of this work related to the dataset creation process. Section 4 is devoted to evaluating and analyzing the quality of the proposed dataset. Finally, Section 5 concludes our work and proposes future perspectives.

2. Related work

There are several datasets available online that have been created for the recognition of Arabic handwritten characters (letters and digits). Those datasets focus on the original Arabic alphabet and digits. The Arabic Handwritten Characters Dataset (AHCD) is a notable example for that; it consists of 16,800 images (28 letters × 600 samples). The letters are written by 60 participants of different ages (between 19 and 40 years old). Each participant wrote each letter 10 times. The dataset is divided into two datasets, a training dataset that contains 13,440 images and a validation dataset that contains 3,360 images. This dataset was created and studied firstly by El-Sawy et al [10].

There is also another dataset that contains the Arabic letters in their different forms (isolated, beginning, middle, final). It contains 6,600 images created by 50 participants of ages ranging from 14 to 50. Each participant wrote 132 different forms. It is created by AlJarrah, Mohammed N et al [11]. For Arabic digits, the MADBase dataset is among the most popular datasets. It was created for the purpose of establishing reference results for the challenge of recognizing Arabic digits using various classification techniques. Another objective is to compare this dataset with the reference dataset of Latin digits, which is the MNIST. For this, the MADBase is based on the ADBase dataset with a modification of the image size to ensure the same as that of MNIST (28x8 images). Both datasets were created by Sherif Abdelazeem and Ezzat El-Sherif (Electronics Engineering Dept., The American University in Cairo). The dataset is composed of 70,000 images, 60,000 for the training dataset and 10,000 for the validation dataset [12]. Table 1 gives an overview of the recognition of these datasets.

However, all these datasets focus on the original Arabic characters, which play a crucial role in understanding and developing the recognition of the Arabic handwriting field. The need for resources that address letters derived from Arabic is highlighted by research on the adaptation of the Arabic script in languages such as Persian and Urdu. Persian, for example, uses letters such as پ (pe) and چ (che) to express specific phonetic sounds that are not present in the original Arabic. Despite their linguistic importance, there is still no dataset dedicated to these derived letters. For this purpose, this study introduces the ADL (Arabic-Derived Letters), which offers the first dataset dedicated to Arabic-derived letters for handwritten character recognition. By providing this dataset, ADL seeks to support studies aimed at creating robust and accurate recognition systems for languages that use letters derived from Arabic origin.

Table 1: Summary of some Arabic characters datasets recognition.

Dataset	Study	Title	Method	Accuracy
The Arabic Handwritten Characters Dataset (AHCD)	Ahmed El-Sawy, Mohamed Loey, Hazem EL-Bakry [10]	Arabic Handwritten Characters Recognition Using Convolutional Neural Network	Convolutional Neural Network (CNN)	94.9%
	Mohammed N AlJarrah, Mo'ath M Zyout, Rehab Duwairi [13]	Arabic Handwritten Characters Recognition Using Convolutional Neural Network	Convolutional Neural Network (CNN)	97.7%
	Boufenar, Chaouki, Adlen Kerboua, and Mohamed Batouche [14]	Investigation on deep learning for off-line handwritten Arabic character recognition	Convolutional Neural Network (CNN)	99.8%
Handwritten Arabic characters database (HACDB)	Mamouni El Mamoun [15]	An Effective Combination of Convolutional Neural Network and Support Vector Machine Classifier for Arabic Handwritten Recognition	Hybrid approach combining CNN and SVM	89.7%
	W. Fakhet, S. El Khediri and S. Zidi [16]	Guided classification for Arabic Characters handwritten Recognition	Convolutional Neural Networks (CNNs)	98%
Modified Arabic Digits Database (MADBase)	El-Sawy et al [17]	CNN for Handwritten Arabic Digits Recognition based on Lenet-5	Hybrid approach based on CNN and LeNet-5	88%
	Latif G et al [18]	Deep Convolutional Neural Network for Recognition of Unified Multi-Language Handwritten Numerals	Convolutional Neural Network (CNN)	99.46%
	Rami S. Alkhawaldeh et al [19]	Ensemble deep transfer learning model for Arabic (Indian) handwritten digit recognition	Ensemble EDTL Model	99.83%

3. Dataset creation

The handwritten characters in the ADL (Arabic-Derived Letters) dataset consist of six letters derived from the Arabic alphabet: پ (Pe), ڤ (Ve), چ (Che), ڠ (Ngain), ڭ, گ, ګ (variations of Gaf), and ژ (Zhe). These letters are used to represent sounds that are not present in the original Arabic language but are frequently present in languages such as Persian and Urdu, which have adapted the Arabic script. The similarities and differences between the original and derived letters are presented in Table 2.

The ڠ (Unicode: U+06A0) is called Ngain [20]. It is used in certain languages, like Jawi (Malay written in Arabic script), to represent the sound /ŋ/, similar to the "ng" sound in "sing" or "song" in English. The letter "گ" (Unicode: U+06AF) is called Gaf. It is used in languages like Persian, Urdu, Pashto, and Kurdish to represent the /g/ sound, which is similar to the "g" in "go" or "gate" in English [21]. Ve is the letter ڤ (Unicode: U+06A4) which represents the /v/ sound in languages such as Persian, Urdu, and Malay. This letter is a variant of the letter Fā, but it is the V sound in English. When pronouncing "ڤ": It is equivalent to the "v" sound in the English terms "voice" or "victory" [22]. The letter "ژ" (Unicode: U+0698) is known as Zhe and is used in the Pakistani, Pashto, and Urdu languages. It corresponds to the sound /ʒ/, similar to the "s" in "measure" or the "g" in "genus" in English [23]. The Arabic letter "چ" (Unicode: U+0686) is known as Che and is used in Persian, Urdu, Pashto, and Kurdish. [24], which corresponds to the "ch" sound in English words such as "church" or "chocolate". The sound /p/ is represented by the letter "پ" (Unicode: U+067E), used in languages such as Persian, Urdu, and Pashto. It is a variant of

the letter "ب" (Bā) but corresponds to the "p" sound in English. To pronounce "پ": It is pronounced like the "p" sound in English words such as "pen" or "paper" [22].

Table 2: Similarities and differences between the original and derived Arabic letters.

Name	Gaf	Pe	Ve	Ngain	Zhe	Che
Derived letter	گ ﮓ ﮕ	پ	ڤ	ڠ	ژ	چ
Original letter	ك ک	ب	ف	ع	ز	ج
Similitude	The same fundamental shape					
Difference	Three dots or a mark above/under the derived letter instead of one for the original letters					

3.1 Letter Variations

Each letter in the dataset is represented in multiple forms, reflecting the context in which the letter appears in writing:

1- These six letters appear in four پ, ڤ, چ, ڠ, گ, ڪ: different formats: isolated, at the beginning of a word, in the middle of a word, and at the end of a word.

2- ژ, ﮋ: This letter has two forms: isolated and at the end of a word.

This variety reflects the unique characteristics of the Arabic script, where letters often change shape based on their position in a word. Additionally, for گ (Gaf), there are three distinct variations: گ and ڪ, both represented across the four formats, resulting in separate rows for each variation, and ﮓ both represented in one row as shown in Table 3.

Table 3: Main forms of Arabic derived letters

Name	Gaf			Pe	Ve	Ngain	Zhe	Che
Isolated	گ	ﮓ	ڪ	پ	ڤ	ڠ	ژ	چ
Initial	گـ	-	ڪـ	پـ	ڤـ	ڠـ	-	چـ
Median	ـگـ	-	ـڪـ	ـپـ	ـڤـ	ـڠـ	-	ـچـ
Final	ـگ	ـﮓ	ـڪ	ـپ	ـڤ	ـڠ	ـژ	ـچ

3.2 Data collection

The dataset was compiled by collecting handwritten samples from 30 participants, categorized into three age groups: 10 participants aged 15 to 20 years, 10 participants aged 20 to 30 years, and 10 participants over 30 years. Each participant was given a table structured to ensure a comprehensive representation of each letter and its various forms. Every letter was written five times by each participant, along with one printed version of the letter, resulting in a total of six occurrences for each letter per participant. The handwritten sheets were arranged into a 14x12 table, with the following structure: Two rows for each of the letters with four forms (isolated, initial, medial, final). One row for ژ(Zhe) and ﮓ (Gaf), containing its two forms (isolated and final). Four rows for the two versions of Gaf (گ and ڪ), to account for their unique variations. This structure ensures that the dataset provides comprehensive coverage of the six Arabic-derived letters and their variations, yielding diverse handwriting samples from contributors of different age groups. Figure 1 shows an example of a scanned image.

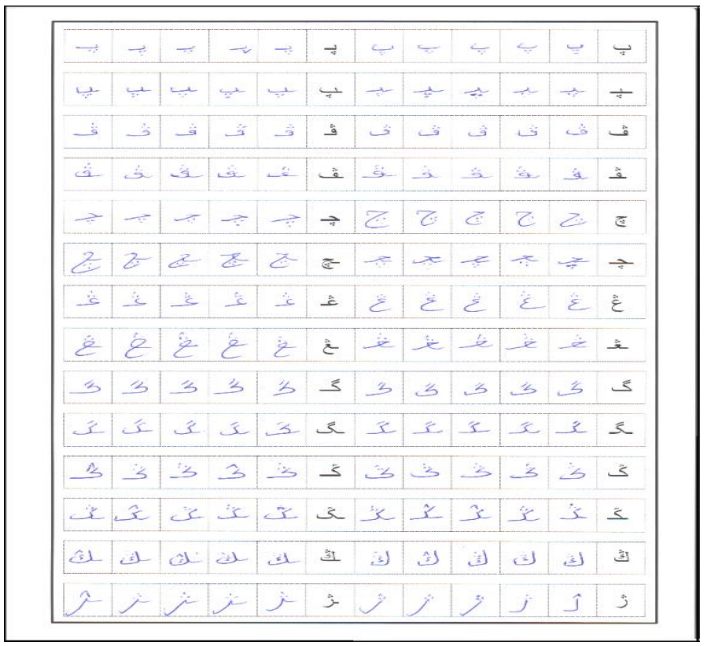

Figure 1: Scanned image of a participant.

3.2 Pre-processing

The original handwritten sheets were scanned at 300 ppi. The images contained extra pixels, so Algorithm 1 is used to remove them and save the letter sections using a maximum bounding box, as shown in Figure 2.

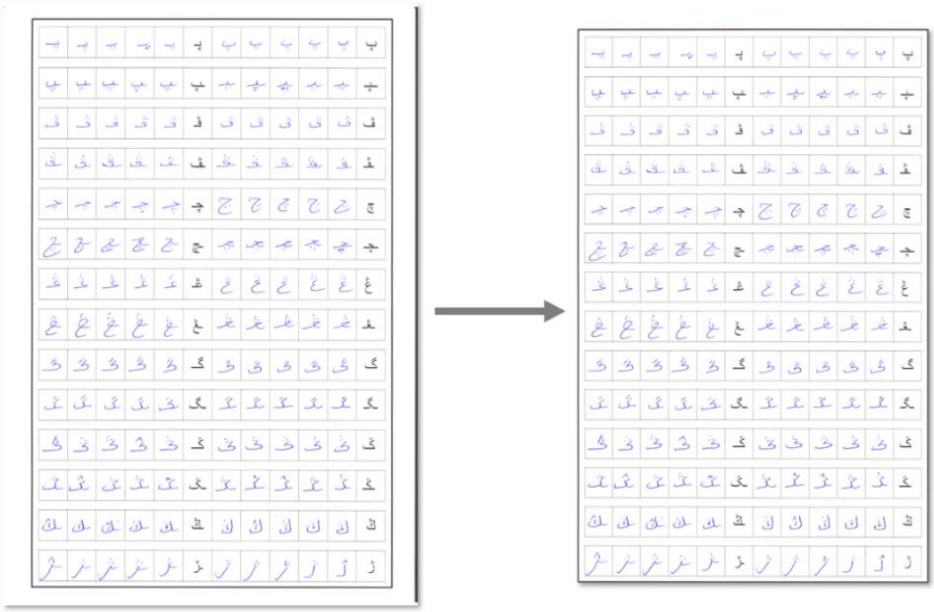

Figure 2: Extraction of the letter region from the scanned images.

Algorithm 1 focuses on extracting letter regions from the scanned images. The algorithm begins by converting each image to grayscale, allowing the use of intensity values $I(x, y)$ for edge detection, where I represents the pixel intensity at coordinates (x, y). A Gaussian blur is then applied to reduce noise, defined by the convolution of the image I with a Gaussian kernel $G(x, y)$ ([25] and [26]):

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (1)$$

Where σ is the standard deviation. Canny edge detection is applied to identify edges by computing gradients and thresholding, yielding a binary edge map $E(x,y)$:

$$E(x,y) = \begin{cases} 1 \; if \; \nabla I > threshold \\ 0 \qquad otherwise \end{cases} \quad (2)$$

Algorithm 1: Cropping the Letter Area

Input: Scanned images dataset **Output:** Cropped images
For each image in the dataset do: 1. Load the color image (cv2.imread) 2. Convert the image to grayscale (cv2.cvtColor) 3. Apply a Gaussian blur to reduce noise (cv2.GaussianBlur) 4. Detect contours using the Canny algorithm (cv2.Canny) 5. Extract the largest external contour (max (contours, key=cv2.contourArea)) 6. Estimate the orientation of the handwritten character: - Use the minimum bounding box (cv2.minAreaRect) - Extract the associated rotation angle 7. Correct the image orientation: - Calculate the rotation matrix (cv2.getRotationMatrix2D) - Apply the rotation (cv2.warpAffine) 8. Define and extract the contour's bounding box in Rotated image (cv2.boundingRect) 9. Add light padding around the region (optional) 10. Resize the image to a fixed size for the model input (cv2.resize) 11. Save the final image to an output folder (cv2.imwrite) **End for**

Contours are extracted from this edge map and sorted by area to isolate the largest contour C_{max}, which is assumed to represent the most significant letter. The bounding box B (as shown in Figure 3) around this contour is defined by its coordinates (x, y) and dimensions (w, h):

$$B = (x, y, w, h) \quad (3)$$

The algorithm exploits the contours by detecting edges (via Canny), then selecting the largest contour to isolate and extract only the area of the table that contains the alphabet, without processing the letters individually. This step allows the images to be properly cropped and prepares them for further processing. The image is then cropped using the bounding box coordinates, ensuring that the entire letter is captured. Following this, a second implementation further processes these cropped images to enhance their quality, applying similar pre-processing techniques to isolate the largest contour. The bounding box of this contour is adjusted to guarantee complete letter capture. This step results in images that are refined and suitable for subsequent classification tasks. Together, these algorithms facilitate accurate letter extraction, serving as a foundation for further analyses in the context of handwritten Arabic-derived letters.

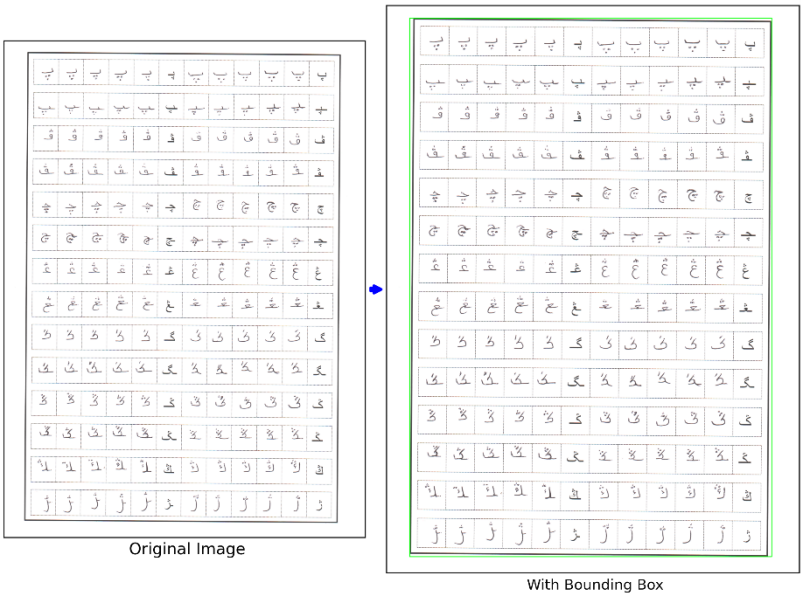

Figure 3: Detection of the largest contour using a bounding box.

Once the letter areas are cropped, Algorithm 2 is used to divide the images into 170x170 pixel images, each with a single letter. Its main function is to obtain column-specific images from segmented letters to improve character analysis and classification, as shown in Figure 4. It starts by setting essential parameters that are fixed according to the margins of the images (after cropping the largest contour, all images are the same size), such as the number of columns to extract (im_num = 12), the spacing between columns (space = 8 pixels), and the height of each column (column_height = 170 pixels). The algorithm works with different input directories, each representing specific letter classes, and builds output directories if they are not already created. It scans the specified number of columns for each image, calculates the end index of each segment using the formula end = start + column_height, and ensures that each segment remains within the image boundaries.

The verification process ensures that the end index does not exceed the width of the image, ensuring data integrity throughout the sequence. We adjust each extracted column by applying white borders around the edges to improve visibility. Specifically, the last two rows and the first two columns, as well as the last five columns, are set to white, ensuring a clean separation from the shadow. The implementation of this algorithm is carried out using Python code, which systematically processes images from the designated input directories. The code iterates through each image, applies the segmentation logic, and saves the resulting column images to the appropriate output directories, following a naming convention that appends the column number to the original filename. This structured approach not only organizes the dataset into individual character columns but also facilitates the downstream tasks of feature extraction and classification. An example of the resulting letters is shown in Figure 5.

Algorithm 2: Splitting Cropped Images into Letters

Input: Cropped images dataset
Output: Split images dataset {Gaf, Ngain, Zhe, Pe, Che, Ve}

Initial variables:
Start_row = 56
Start_col = 49
End_row = 0
End_col = 0
Start = 0
Row_height = 170
Col_height = 170
Row_len = Image_len - 48
Distance_between_cols = 8
Distance_between_rows = 65
For image in Cropped images do:
  For 14 iterations (number of rows) do:
    End_row = Start_row + Row_height
    Cropped_row = image (Start_row to End_row, Start_col to Row_len)
    For 12 iterations (columns) do:
      End_col = Start + Col_height
      Cropped_col = Cropped_row(Start to End_col)
      Start = End_col + Distance_between_cols
      Save Cropped_col
    End for
    Start_row = End_row + Distance_between_rows
    Start = 0
  End for
End for

Figure 4: Cropping of Individual letter Images by Algorithm 2.

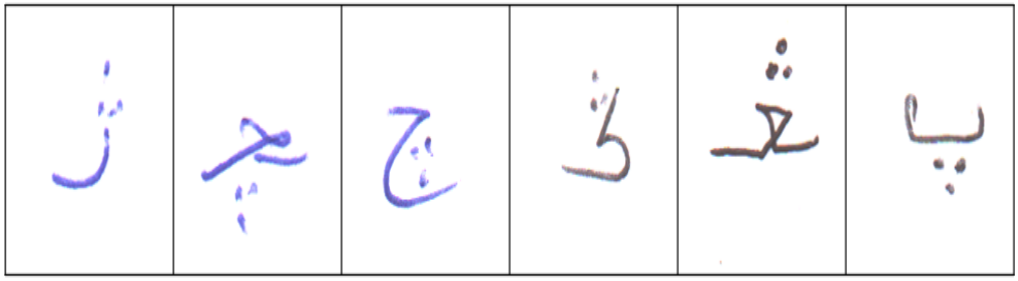

Figure 5: Samples of cropped images.

After splitting the images into 170x170 pixels, Algorithm 3 is used to binarize and denoise the entire dataset, as shown in Figure 6.

Algorithm 3: Data Denoising and Binarization

Input: Dataset directories {Gaf, Ngain, Zhe, Pe, Che, Ve} **Output:** Binarized dataset

For each directory in directories do:
  For each image in the directory do:
  Convert the image to grayscale
  Apply Gaussian blur to denoise the image (Kernel (5,5))
  Binarize the image using Otsu's Thresholding
  Save the binarized image
  End for
End for

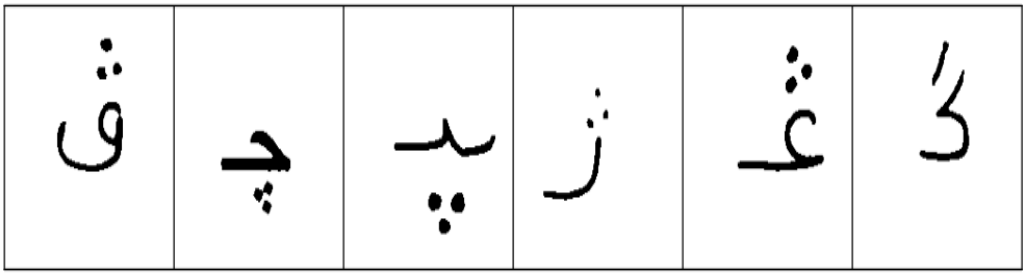

Figure 6: Sample of denoised and binarized images.

Algorithm 3: Data Denoising and Binarization involves a systematic approach to enhance the quality of images in the dataset directories, which include letters such as Gaf, Ngain, Zhe, Pe, Che, and Ve. The process begins by iterating through each directory and accessing the images contained within. For every image, it is first converted to grayscale, reducing the color channels and allowing for simpler processing. Subsequently, Gaussian blur is applied to the grayscale image, which can be mathematically represented as [27]:

$$I_{blurred}(x,y) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} I(x',y') e^{-\frac{(x-x')^2+(y-y')^2}{2\sigma^2}} dx'\, dy' \quad (4)$$

where I is the original image, $I_{blurred}$ is the resulting image after applying the Gaussian filter, and σ is the standard deviation of the Gaussian distribution, controlling the amount of blur. Following the denoising step, the images are binarized using Otsu's thresholding method, which determines an optimal threshold τ that minimizes intra-class variance and can be expressed as:

$$\tau = \arg\min_{\tau} [w_0(\tau)\sigma_0^2(\tau) + w_1(\tau)\sigma_1^2(\tau)] \quad (5)$$

where w_0 and w_1 are the weights (probabilities) of the two classes, and σ_0^2 and σ_1^2 are their respective variances.

After obtaining the binarized image, it is saved back into the corresponding directory. This algorithm ensures that the resulting binarized dataset is noise-free and ready for subsequent processing steps, improving the performance of any machine learning tasks that utilize the dataset.

To expand our dataset, Algorithm 4 is applied to generate 10 additional instances of each image.

Algorithm 4: Data augmentation

Input: Binarized dataset directories {Gaf, Ngain, Zhe, Pe, Che, Ve}
Output: Augmented dataset

```
For each directory in directories do:
  For each image in the directory do:
    For 10 iterations do:
      Apply the following augmentations:
        - Random rotation (rotation_range=2)
        - Random horizontal shift (width_shift_range=0.005)
        - Random zoom (zoom_range=0.05)
        - Nearest neighbor interpolation (fill_mode='nearest')
      Save the new augmented image
    End for
  End for
End for
```

Algorithm 4: Data Augmentation is implemented to significantly enrich the binarized dataset of handwritten Arabic letters, comprising directories for letters such as Gaf, Pe, Ve, Zhe, Che, and Ngain. Utilizing a sophisticated image data generator, this algorithm applies a variety of augmentation techniques aimed at enhancing the diversity of the training dataset. For each image, the algorithm incorporates slight rotations (up 2°), moderate horizontal shifts (up to 0.5% of the image width), and minor zoom adjustments (up to 0.5%). The following mathematical expressions define these transformations: for rotation, a transformation matrix is applied to rotate the image around its center by an angle θ; for horizontal shifts, the new position is calculated using $x' = x + d_x$, where d_x is drawn from a uniform distribution, and for zooming, the resizing is written as $z(1 + \Delta_x)$, where z expresses the original size and Δ_x represents a slight disturbance emitted from $[-0.05,0.05]$. A modified file name is used to save each augmented image to preserve the original image, allowing up to 10 new variations to be created for each source image.

Beyond increasing the overall size of the dataset, this augmentation process also improves the model's ability to generalize effectively to various handwritten character variations, enhancing its robustness and performance in recognizing the Arabic alphabet. The dataset is divided into three sets: training, validation, and test. Figure 7 shows the distribution of images per class. The dataset ensures the same number of images for each form of writing, with the difference being the number of forms per alphabet. This is the case of the letter Gaf, which has 10 different forms, which is justified by its high number of images.

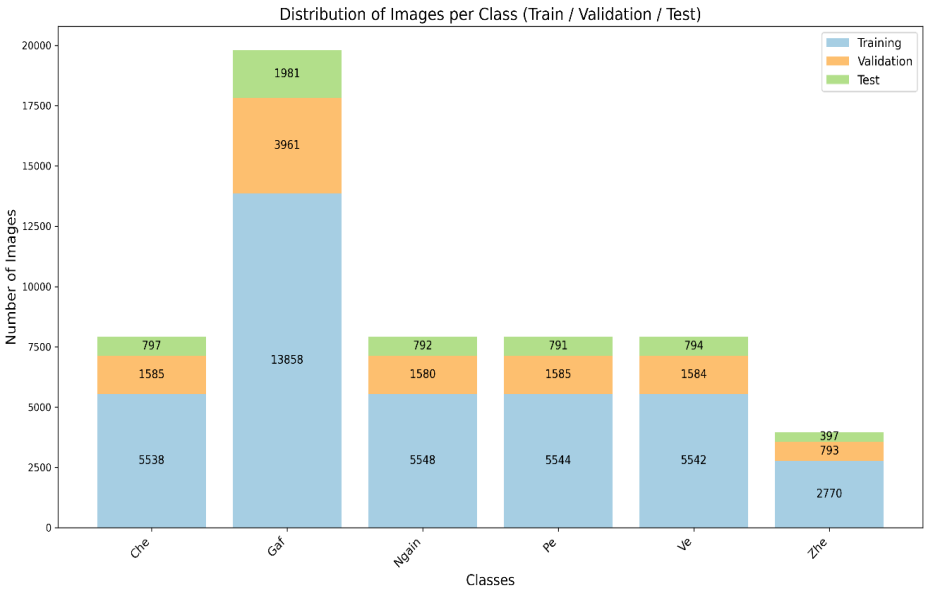

Figure 7: Stacked representation of training, validation, and test images per class.

Given the nature of the images, there are many unused background pixels. To address this, Algorithm 5 is applied to remove these redundant pixels, resize the images to a 32x32 resolution, and invert them to display white letters on a black background (pixel values were normalized to 0 or 255), thereby assigning higher pixel values to the letter regions as shown in Figure 8. Algorithm 5 aims to refine the augmented dataset by focusing solely on the characters represented in each image. The process begins by iterating through directories containing the augmented character images (Gaf, Ngain, Zhe, Pe, Che, and Ve). For each image, the algorithm identifies the black pixels (which correspond to the letters) and constructs a bounding box around them, effectively cropping out any unused whitespace. This step is crucial for enhancing the model's performance by ensuring that only relevant information is retained. Once the images are cropped, they are resized to a consistent dimension of 32x32 pixels, providing uniformity across the dataset. Finally, each image undergoes an inversion process to create a black background with white letters, improving visibility and contrast. Mathematically, the bounding box can be described as:

$$bounding_{box} = \min(black_{pixels}) - 2\ to\ \max(black_{pixels}) + 2 \quad (6)$$

Where $black_{pixels}$ are the pixel coordinates of the identified black pixels. The entire operation is repeated for all images, resulting in a clean, standardized dataset ready for further analysis or model training.

Algorithm 5: Remove Unused Pixels/Reshape and Invert Images

Input: Augmented dataset directories {Gaf, Ngain, Zhe, Pe,Che,Ve} **Output:** Final dataset

For directory in directories do:
  For image in directory do:
  Identify the black pixels (representing the letter)
  Create a bounding box around these pixels
  Crop the image using that bounding box
  Resize the cropped image to 32x32 pixels
  Invert the image to have a black background and white letter
  End for
End for

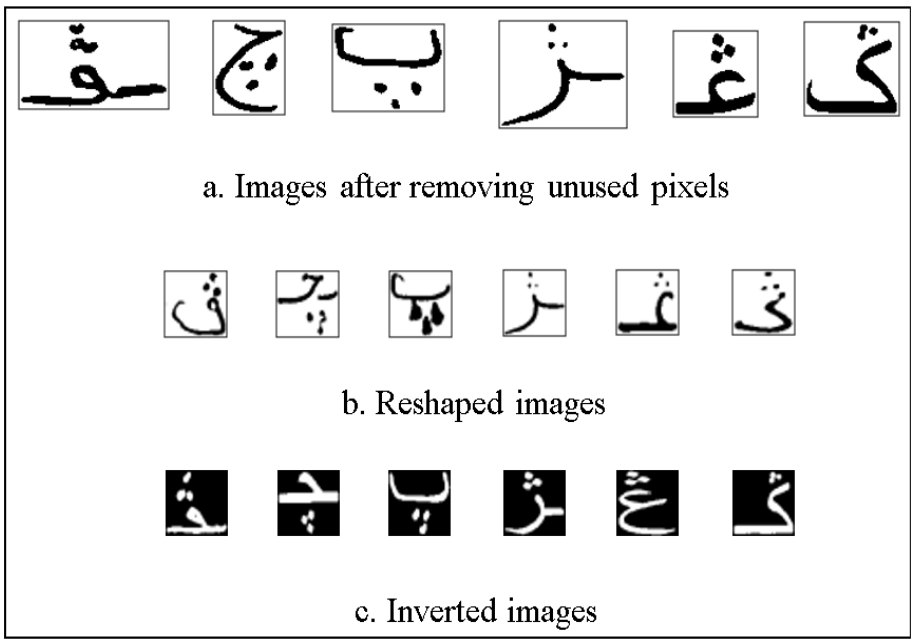

Figure 8: Sample images after removing unused pixels, reshaping, and inverting.

3.4 Pixel distribution and class analysis

Figure 9 presents the pixel distribution across the training, validation, and test datasets. In these images, the white pixels correspond to the letters themselves, and the overall distribution of white pixels is almost 16.1% for the three datasets. This gives higher-weight pixels to the regions forming the main body of the letters, making the letter structures clearly distinguishable and easier for classification models to capture. The relative proportion of the space occupied by the characters compared to the black background is represented by this percentage. The constant ratio of white pixels between the three sets demonstrates the effective preservation of the visual structure of the letters, thus ensuring a balanced representation of the character shapes in the training, validation, and test sets. Ensuring uniformity is crucial to ensure that the model is exposed to similar pixel-level features in the three sets, which enhances its generalization ability.

Figure 10 illustrates the distribution of intensity pixels across training, validation, and test datasets. For each of the six classes, the distribution of black and white pixels is shown in the form of bar charts. For each class, three bars represent the proportion of black pixels in the training, validation, and test sets, while three other bars indicate the proportion of white pixels. This representation shows that the pixel distribution is globally balanced between the three datasets for all classes. This ensures that the consistency of visual characteristics (pixel density distribution) is respected between the datasets. It means that there is no imbalance due to the distribution of the data. Therefore, the classification models will not be influenced by density variations (too many black or white pixels in a particular dataset).

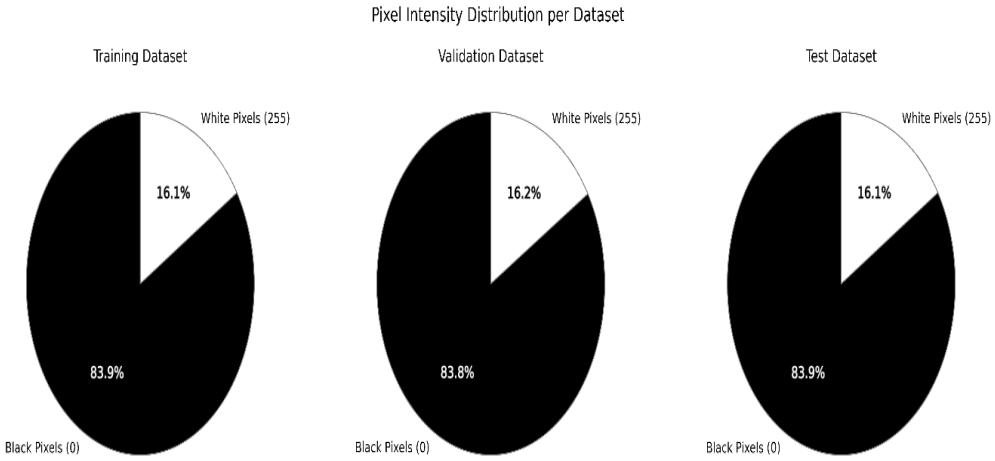

Figure 9: Distribution of Pixel Values for Training, Validation, and test datasets.

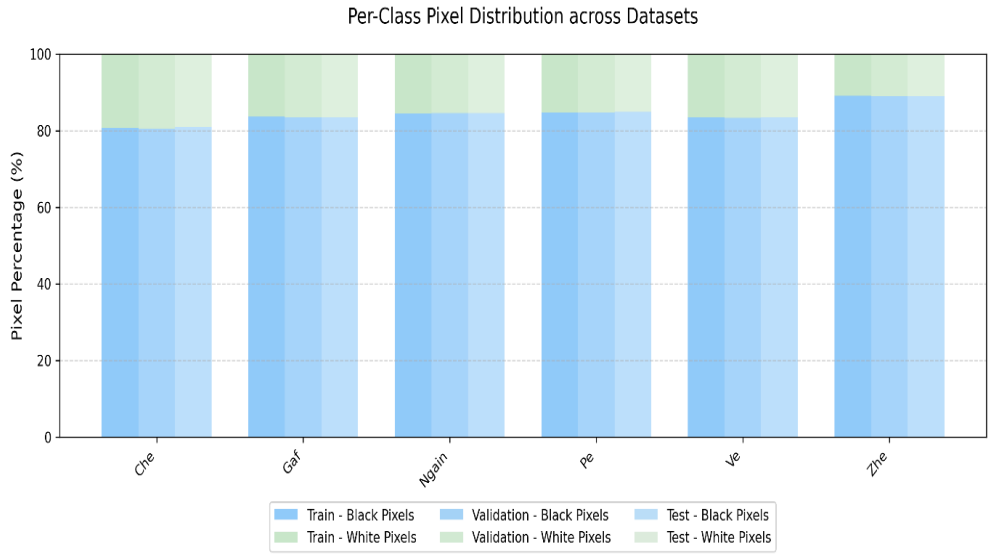

Figure 10: Distribution of intensity pixels across Datasets.

3.5 Sample Structure of The Dataset

The ADL dataset is organized in CSV format to facilitate ease of use for machine learning applications. Each row in the dataset corresponds to a single image, represented by a class label and binary pixel values (0 and 255). The pixel values capture the structure of handwritten Arabic-derived letters with distinct white-on-black pixelization after pre-processing, isolating the foreground letter against a black background. Figure 11 illustrates the structure of the CSV file, showing an example of a single row corresponding to the letter 'Che', represented as a 32×32-pixel image. This concise representation highlights the binary nature of the dataset, which was designed to maximize contrast between character shapes and background, optimizing machine learning training processes.

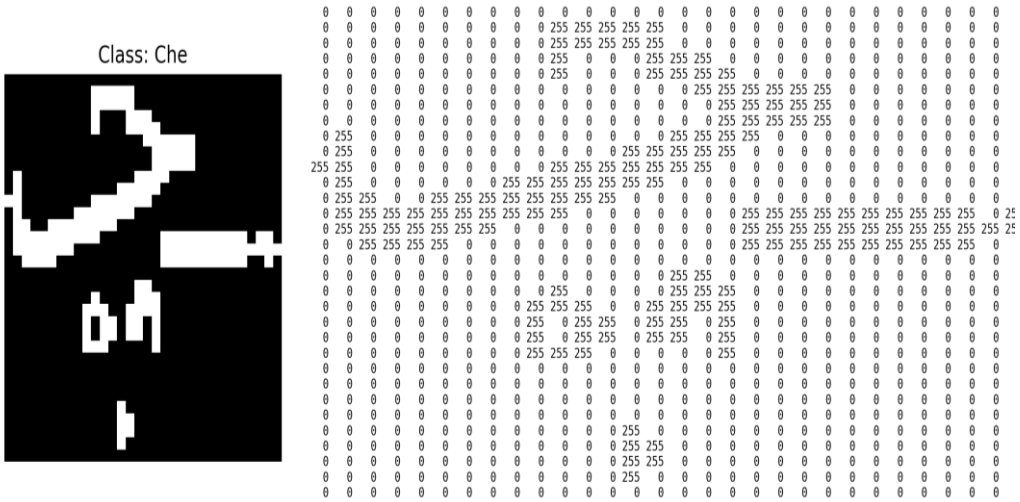

Figure 11: Flattened Pixel Values (as 32x32)

The CSV files are organized in a way that they are compatible with commonly used machine learning frameworks, making it easy to import and pre-process the data. There are three subsets in the dataset: training, validation, and test, which are stored separately to facilitate model training and evaluation. Balanced class representation is maintained by each subset through data augmentation, ensuring robust model performance across diverse character classes and contexts.

4. Dataset Evaluation

To evaluate the dataset, a comparative study of three classification models was conducted using: SVM, CNN, and Vision Transformer (ViT), for the purpose of assessing both the complexity of the images and the ability of the models to learn and generalize them. The SVM architecture, despite its simplicity as it is a classic AI method, achieved a very high accuracy of 99.8% for the test dataset, as shown in Figure 12. This result suggests that the classes are largely separable in the vector space formed by the flattened pixels, which may indicate low interclass variability in some shapes. CNN, similarly, also achieved an accuracy of about 99.98% after only 10 training epochs, as shown in Figure 13. This is due to the exploitation of the spatial and local characteristics of the images via convolutions, which confirms that the dataset contains visual patterns distinctive enough to be effectively captured by a convolutional neural network. The ViT, although architecturally more sophisticated, was trained for 30 epochs. Figure 14 indicates that the ViT achieves a slightly lower accuracy of around 98.25% than the SVM and CNN. This result can be explained by the fact that Vision Transformers generally require large amounts of data to outperform CNNs, especially on small images (32x32). This indicates that, in the context of small datasets with little noise, simpler models like CNN or SVM may be better suited in terms of efficiency and performance. In summary, this comparison highlights that the dataset can be learned by traditional models, and that the richness of representations learned by Transformers does not necessarily translate into better performance without an appropriate data volume. Table 4 shows a brief comparison of the models' performance.

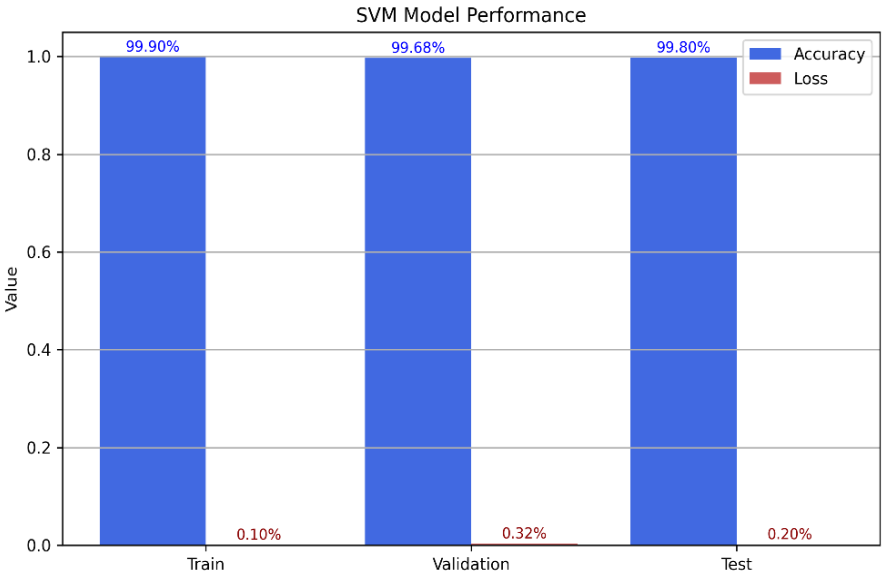

Figure 12: SVM Model Performance.

Final Accuracies → Train: 99.81%, Val: 99.98%, Test: 99.98%
Final Losses → Train: 0.53%, Val: 0.10%, Test: 0.03%

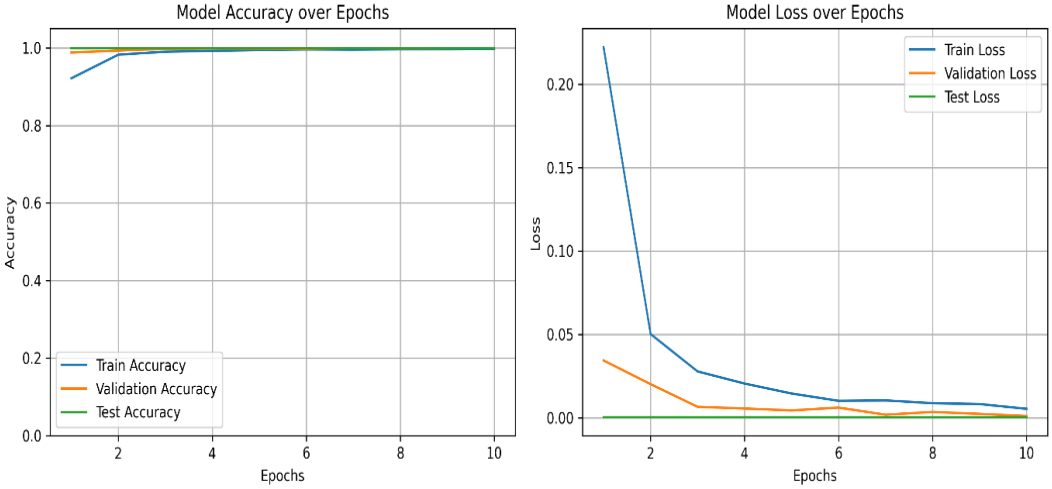

Figure 13: CNN Model Performance.

Final Accuracies → Train: 98.66%, Val: 98.31%, Test: 98.25%
Final Losses → Train: 3.92%, Val: 4.89%, Test: 4.74%

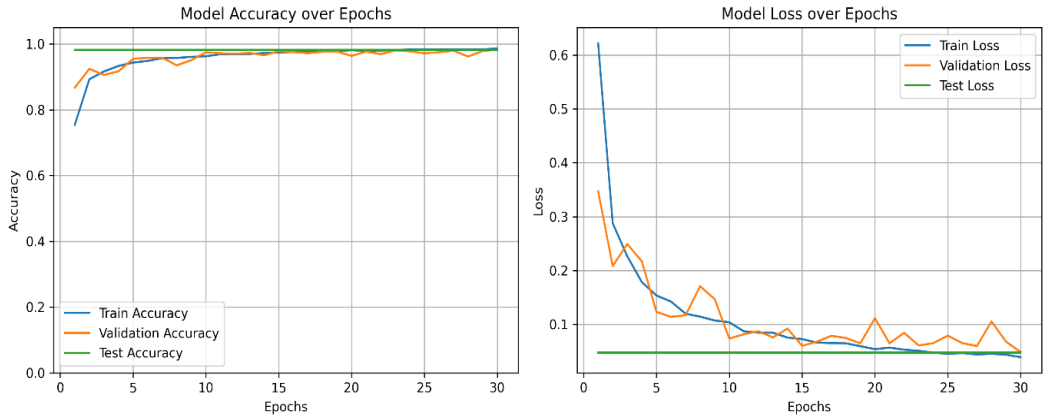

Figure 14: ViT Model Performance.

Table 4: Comparative Analysis of Models' Performance

Model	Train Accuracy	Validation Accuracy	Test Accuracy	Observations
SVM	99.90%	99.68%	99.80%	Very high accuracy, simple model, quick to train.
CNN	99.81%	99.98%	99.98%	Excellent generalization. Very stable performance thanks to spatial feature learning.
ViT	98.66%	98.31%	98.25%	Less efficient than CNN and SVM on a small dataset (32x32). Transformers generally require more data to outperform CNNs.

5. Conclusion

The ADL dataset marks a major advancement in the field of handwriting recognition for Arabic-derived letters. By including different characters such as Che, Ngain, Pe, Ve, Zhe, as well as the three forms of Gaf, the dataset offers a comprehensive resource for researchers and practitioners who want to improve machine learning models for handwritten character recognition. The dataset contains 55440 images, 38800 images for the training dataset, 11088 for the validation, and 5552 for the test dataset. The careful creation of this resource, including different positional forms and demographic diversity among participants, ensures that it captures the subtleties of authentic handwriting. Understanding of structural variations among letters is further enriched through analysis of pixel distributions, which provides valuable insights for model training. The implementation of the ADL dataset not only provides a solution to the current lack of specialized resources for Arabic-derived characters but also facilitates future research and development in optical character recognition (OCR) systems. By encouraging progress in this field, the aim is to contribute to the overall goal of improving automatic text recognition technologies, thus providing greater efficiency in processing Arabic script in different applications. As a future work, this dataset requires an in-depth study based on machine learning techniques to classify it. A comparative study that takes old and new classification methods to better illuminate the usability of this dataset.

6. Ethics Approval

This research was conducted in accordance with ethical standards for research involving people. Prior to data collection, all participants gave informed consent, which ensured that they were fully aware of the purpose of the study.

7. Funding

None

8. Acknowledgements

The author extends heartfelt gratitude to all participants for their invaluable contribution in filling the alphabet sheets.

9. Conflicts of Interest

The author declares no conflict of interest.

References

[1] T. Ethnologue, "What are the top 200 most spoken languages?," 2025. [Online]. Available: https://www.ethnologue.com/insights/ethnologue200/. [Accessed 18 06 2025].

[2] D. Abu-Shaqra, "Arabic Type Classification Sys-tem-Qualitative Classification of Historic Arabic Writing Scripts in the Contemporary Typographic Context," Ocad University, Toronto, Canada, 2020.

[3] M. H. Bakalla, "Arabic culture: through its language and literature," Taylor & Francis, 2023.

[4] E. Alsharhan, A. Ramsay. and H. Ahmed, "Evaluating the effect of using different transcription schemes in building a speech recognition system for Arabic," *International Journal of Speech Technology,* vol. 25, no. 1, pp. 43-56, 2022.

[5] S. G. N. L. Kellman, The Routledge Handbook of Literary Translingualism. Routledge, Taylor & Francis Group, 2022.

[6] P. Coluzzi, "Jawi, an endangered orthography in the Malaysian linguistic landscape," *International Journal of Multilingualism,* vol. 19, no. 4, pp. 630-646, 2022.

[7] M. G. H, The Performance and Patronage of Baloch Culture Through Music (And Related Arts) in the Eastern Arabian Peninsula, Diss: City University of New York, 2020.

[8] P. Jafarzadeh, P. Choobdar. and V. M. Safarzadeh, "Khayyam Offline Persian Handwriting Dataset," *arXiv,* 2024.

[9] D. Raiomond, A. Gutkin., C. Johny., B. Roark. and R. Sproat, "Graphemic normalization of the Perso-Arabic script.," *arXiv,* 2022.

[10] M. L. H. E.-B. Ahmed El-Sawy, "Arabic handwritten characters recognition using convolutional neural network," *WSEAS Transactions on Computer Research,* vol. 5, p. 11–19, 2017.

[11] M. A. a. A. B. A. Lawgali, "HACDB: Handwritten Arabic characters database for automatic character recognition," *European Workshop on Visual Information Processing (EUVIP), Paris, France,* pp. 255-259, 2013.

[12] E. E.-S. Sherif Abdelazeem, "The Arabic Handwritten Digits Databases ADBase & MADBase," Electronics Engineering Dept., The American University in Cairo, [Online]. Available: https://datacenter.aucegypt.edu/shazeem/?utm_source=chatgpt.com. [Accessed 21 06 2025].

[13] M. M. Z. R. D. Mohammed N AlJarrah, "Arabic Handwritten Characters Recognition Using Convolutional Neural Network," *2021 12th International Conference on Information and Communication Systems (ICICS). IEEE,* pp. 182-188, 2021.

[14] C. A. K. a. M. B. Boufenar, "Investigation on deep learning for off-line handwritten Arabic character recognition," *Cognitive Systems Research,* vol. 50, pp. 180-195, 2018.

[15] M. E. Mamoun, "An Effective Combination of Convolutional Neural Network and Support Vector Machine Classifier for Arabic Handwritten Recognition," *Aut. Control Comp. Sci,* vol. 57, p. 267–275, 2023.

[16] S. E. K. a. S. Z. W. Fakhet, "Guided classification for Arabic Characters handwritten Recognition," *2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA), Abu Dhabi, United Arab Emirates,* pp. 1-6, 2022.

[17] E.-B. H. L. M. El-Sawy A, " Cnn for handwritten Arabic digits recognition based on lenet-5," *A.E. Hassanien, K. Shaalan, T. Gaber, A.T. Azar, M.F. Tolba (eds.) Proceedings of the international conference on advanced intelligent systems and informatics 2016. Springer International Publishing, Cham,* p. 566–575, 2017.

[18] A. J. A. L. N. M. A. Y. Latif G, "Deep Convolutional Neural Network for Recognition of Unified Multi-Language Handwritten Numerals," *2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR),* p. 90–95, 2018.

[19] M. A. N. F. F. A. W. Z. A. A. A. Rami S. Alkhawaldeh, "Ensemble deep transfer learning model for Arabic (Indian) handwritten digit recognition," *Neural Computing and Applications,* vol. 34, p. 705–719, 2022.

[20] J. Abu Bakar, K. Omar, M. F. Nasrudin and M. Z. Murah, "NUWT: Jawi-specific Buckwalter corpus for Malays word tokenization," *Journal of Information and Communication Technology (JICT),* vol. 15, no. 1, pp. 107-131, 2016.

[21] O. F. A. Adeeb and J. K. Seyed, "Arabic text steganography based on deep learning methods," *IEEE Access,* vol. 10, pp. 94403-94416, 2022.

[22] E. A. Mouhssine and E. F. Khalid, "Recognition of Intrusive Alphabets to the Arabic Language Using a Deep Morphological Gradient," *Revue d'Intelligence Artificielle,* vol. 34, no. 3, pp. 277-284 , 2020.

[23] G. H. Gautier, "Dirêjî Kurdî: a lexicographic environment for Kurdish language using 4th

Dimension®," *5th International Conference and Exhibition on Multilingual Computing (ICEMCO),* vol. 5, 1996.

[24] D. Salih, Kurdish Sorani Spelling Checker System, Diss, England: University of Birmingham, 2021.

[25] G. Lugo, H. Nasim and C. Irene, "Semi-supervised learning approach for localization and pose estimation of texture-less objects in cluttered scenes," *Array,* vol. 16, 2022.

[26] S. Jana, P. Ranjan and S. Bijan, "A semi-supervised approach for automatic detection and segmentation of optic disc from retinal fundus image," *Handbook of computational intelligence in biomedical engineering and healthcare, Academic Press,* pp. 65-91, 2021.

[27] V. K. M. V. Harikrishnan and G. Ashima, "Diabetic retinopathy identification using autoML," *Computational Intelligence and Its Applications in Healthcare, Academic Press,* pp. 175-188, 2020.