Recognition and Translation of Ancient South Arabian Musnad Inscriptions

Afnan Altalhi, Atheer Alwethinani, Bashaer Alghamdi, Jumanah Mutahhar, Wojood Almatrafi, Seereen Noorwali
College of Computers and Information System, Umm AlQura University
Makkah, Saudi Arabia

***Abstract*—Inscriptions play an important role in preserving historical information. As such, conservation of these inscriptions provides valuable insights into the history and cultural heritage of the region. *Musnad* inscriptions are considered one of the earliest forms of writing from the Arabian Peninsula, preceding the modern Arabic font; however, most Musnad inscriptions remain unread and untranslated, signifying a substantial loss of historical information. In response, this paper represents a significant contribution to the field by proposing a successful approach to interpreting Musnad inscriptions. To do so, a dataset was prepared from the Saudi Arabian Ministry of Culture and subjected to preprocessing for optimal recognition, a step that entailed several experiments to enhance image quality and preparedness for recognition. The dataset was then trained and tested with 29 classes using three different convolutional neural network (CNN) architectures: Visual Geometry Group 16 (VGG16), Residual Network 50 (ResNet50) and MobileNetV2. Thereafter, the performance of each architecture was evaluated based on its accuracy in recognising Musnad inscriptions. The results demonstrate that VGG16 achieved the highest accuracy of 93.81%, followed by ResNet50 at 89.39% and MobileNetV2 at 80.02%.**

Keywords*—*Musnad inscriptions; text recognition; deep learning; VGG16; ResNet-50; MobileNetV2

I. Introduction

The ancient writings and inscriptions found in the Arabian Peninsula hold great significance in the modern era, as they serve as a source for historians and researchers studying ancient history and civilisations. Within the Kingdom of Saudi Arabia, numerous archaeological sites are covered with inscriptions and ancient texts written in the form of Musnad inscriptions, which represent a rich cultural heritage and hold great historical value.

The Musnad inscriptions, originating in the southern region of the Arabian Peninsula, predates the current Arabic font, with Musnad inscriptions discovered on diverse surfaces, from forts and mountainsides to smaller stone fragments and statue bases [1]. These inscriptions provide invaluable insights into the lives of the individuals who created them, including their lifestyle, beliefs, political climate and relationships with neighbouring nations. In addition, Musnad inscriptions were utilised in daily interactions, further highlighting their importance [2] [3].

The Musnad inscriptions is comprised of 29 characters, and the writing direction is right to left. Unlike Arabic font, the characters in Musnad are separate and unconnected in a word. This means the shape of each character remains the same, regardless of its position in the word. As well, words are separated by a vertical line (|). Furthermore, Musnad does not include any punctuation or diacritical marks [2].

In Fig. 1, an example of a Musnad inscription from Al-Faw village is depicted. To translate these inscriptions, each Musnad character is converted into its corresponding counterpart in Standard Arabic, as shown in Fig. 2, where the translation considers the vertical line that separates each word. It is important to note that the translated words belong to an old Arabic dialect, but they are all Arabic [4].

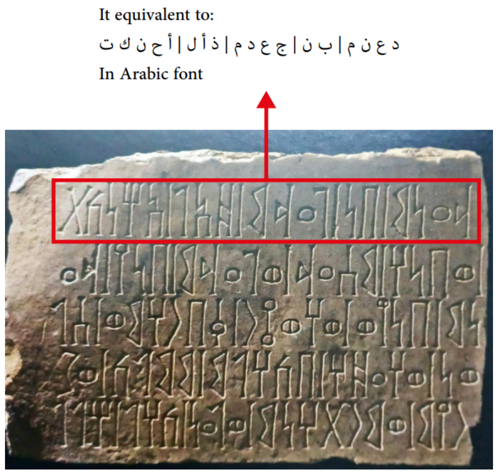


Fig. 1. One of the musnad inscriptions in the village of Al-Faw.

Motivated by the limitations of existing translation methods, such as the time-consuming nature of manual techniques and the dependence on expert availability, this research presents the first attempt to automate the recognition and translation of Musnad inscriptions into Arabic using image processing and deep learning. Consequently, it seeks to enhance the experience of reading Musnad inscriptions, making them more accessible and effortless to understand. Thus, the contributions of this work can be summarised as follows:

- Creation of a dataset for the Musnad inscriptions.
- Application of an optimal image processing technique to improve recognition accuracy.
- Comparison of the performances of three deep learning models (ResNet50, MobileNetv2 and VGG16) to determine which has the highest accuracy.

The remainder of the paper is structured as follows: Section II presents a literature review, Section III provides a detailed description of the Musnad inscription dataset, Section IV outlines the proposed methodology, Section V presents the

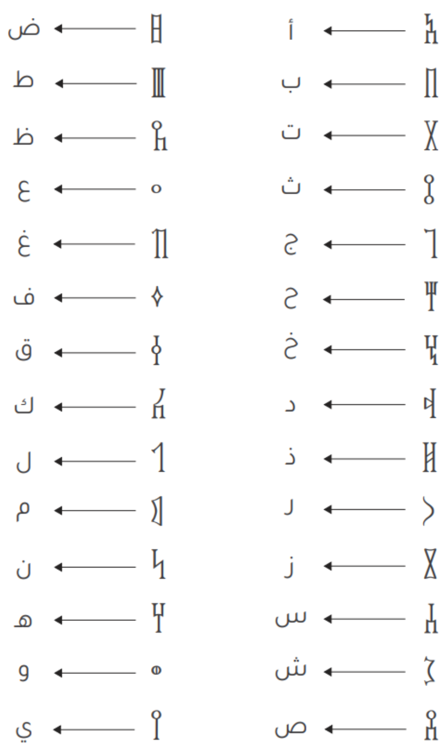


Fig. 2. Musnad alphabet.

results and discussions and Section VI concludes the paper and suggests future directions for research.

II. Literature Review

Scientific research has adopted different approaches to implementing detection and recognition, depending on the inscriptions, ancient characters and language itself. This section will thus provide an overview of related works in our field.

According to previous research, you only look once (YOLO) object detection methods are used in [5], [6], which implements the CNN architecture. YOLOv3-tiny was able to recognise the Kawi character on copper inscriptions due to its high detection accuracy (average of 97.93% in [5] and high detection speed. Meanwhile, the Oracle Bone inscriptions (OBIs) were recognised using two deep learning models in [6]: first, YOLOv3-tiny was used to detect and recognise OBIs, and second, MobileNet was used to detect undetected OBIs, as YOLOv3-tiny's limitations prevent all OBIs from being correctly recognised. Thus, MobileNet had the best performance in training accuracy and validation accuracy (99.30% and 98.89%, respectively).

Further, [7] analysed and identified four different algorithms for adapting the OCR process to recognise Tamil scripts, and support vector machines (SVM) was identified as the best option. Meanwhile [8] a method called advanced maximally stable extremal regions (AMSER) to improve the accuracy of identifying Tamil characters in images by detecting the extremal region and characteristic function, the accuracy of recognition of which was 95.59%. This method was introduced because of the low accuracy of inscriptions images using OCR. In addition, [9] suggested a k number of clusters (k-means clustering) for an ancient Kannada text using scale-invariant Fourier transform (SIFT) and speeded up robust features (SURF). Moreover, in [10], OCR was used to identify ancient Tamil inscriptions on stone using a feature extracted with the SIFT algorithm.

According to [11], who employed the Siamese network in few-shot learning (FSL),the local feature extraction of Chinese characters performs better using the VGG16 network as the backbone feature extraction network, achieving a recognition accuracy of 82.67%. Meanwhile, [12] the efficient and accurate scene text (EAST) detection model and the feature extractor VGG16 to extract painting inscriptions by combining the characteristics of Chinese paintings, achieving a high accuracy of 89%. In addition, in [13], Sundanese writing inscribed on palm leaves was recognised using a three-layer CNN with a 73% recognition accuracy. Next, in [14], was employed for recognition, and Tesseract training was performed using a deep neural network architect. Then, CNN long–short-term memory (LSTM) networks were configured and trained for the language model of the Tamizhi script, leading to an OCR accuracy of 91.21%. Furthermore, [15] used a CNN to extract and translate information from each character into Modern Tamil, achieving an accuracy of 94.6%.

CNNs are used as feature extractors, as well as classifiers, for their ability to recognise 33 classes of basic characters from Devanagari ancient manuscripts, as in [16], reaching a recognition accuracy of 93.73%. Further, the historical Kannada handwritten characters are recognised using the line segmentation approach with LBP features in [17], and the SVM classifier achieved a good performance, with an accuracy of 96.4%. In [18], a CNN was used with dropout to recognise Brahmi words, with a 92.47% accuracy. As well, [19] designed and developed an automatic recognition tool for variant characters to assess tablet inscriptions, leading to an accuracy of the trained model ResNet50-18 of 90%. In addition, [20] utilised CNN and MobileNet to detect Tamil-Brahmi script, achieving an accuracy of 68.3%, but MobileNet outperformed all other models employed. In [21], a CNN was used to classify and recognise Brahmi characters, as well as broken part of these characters, and VGG16 models provided results with the best accuracy, at 93.33%.

Through this literature review, various approaches, such as SIFT, ResNet18 and VGG16, for feature extraction and classification techniques, including CNN, were explored. The results reported in these papers vary depending on such factors as the condition of the inscriptions and the techniques employed. Despite an extensive review, it is noteworthy that no research paper was identified that specifically addresses the recognition and translation of inscriptions in the ancient Southern Arabic Musnad font.

III. Dataset

A. Dataset Collection and Description

A request was made to the Ministry of Culture to provide a collection of images for use as a dataset in this work, and they did so with a collection of images from the Heritage Commission of the Ministry of Culture. The dataset contained images of the Musnad inscriptions in Lahyani form. And contained images of the Musnad from Qaryat al-Faw, Yemen and Al-Ula, totalling 293 inscriptions, which were divided into three categories: real, written and anointed, as shown in Fig. 3.

As the Musnad inscriptions span consecutive historical stages, researchers divided them into three types: ones that take on a geometric look, as in Fig. 4(b); ones that tend to bend, as in Fig. 4(c); and ones that appear exaggerated in decoration, as in Fig. 4(a) [2][22]. Further, the dataset comprises all feasible images sourced from the Ministry of Culture, ensuring the highest quality.

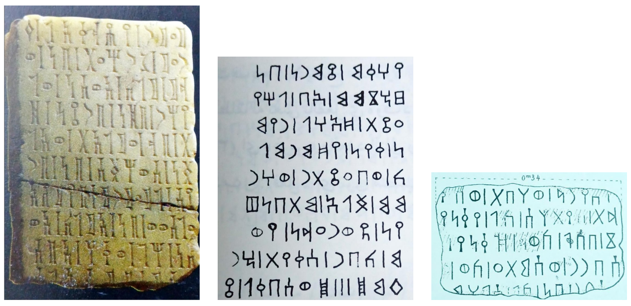

(a) Real (b) Written (c) Anointed

Fig. 3. The three categories of the dataset.

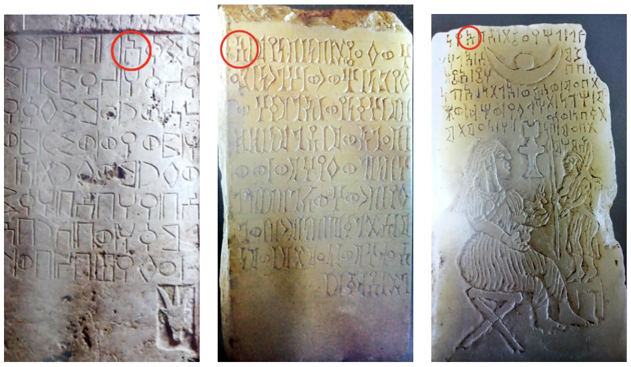
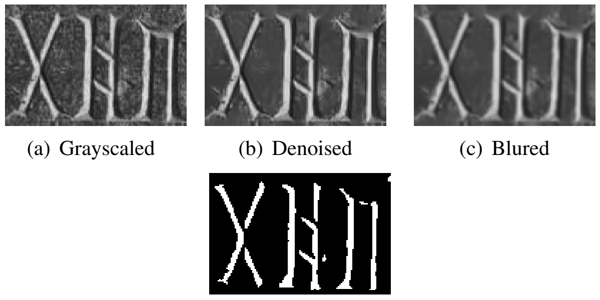

(a) Decorated (b) Geometric (c) Bend

Fig. 4. Three periods of the musnad inscriptions.

B. Image Preprocessing

The preprocessing experiments aimed to improve the recognition efficiency and accuracy of the dataset [15]. Of the several experiments conducted, only two produced good results, one for natural scene images and one for anointed and written images, as described below:

1) Preprocessing for natural scene images: In the first experiment, four processes were carried out using natural scene images. After first converting the input image from RGB to grayscale, as shown in Fig. 5(a), the grayscale image was then denoised using the cv2.fastNlMeansDenoising function, and the result of the denoising is shown in Fig. 5(b). The denoised grayscale image is then smoothed with the cv2.GaussianBlur function, as shown in Fig. 5(c). Finally, the smoothed grayscale image is then binarised using the cv2.threshold function with a threshold of 130, as shown in Fig. 5(d).

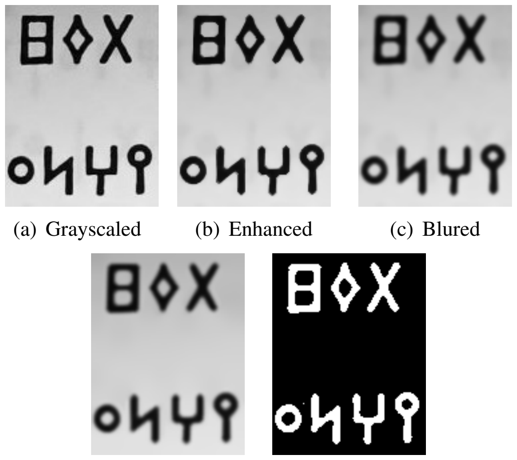

(a) Grayscaled (b) Denoised (c) Blured

(d) Binarized

Fig. 5. Preprocessing for natural scene images.

2) Preprocessing for anointed and written images: In the second experiment, five processes were carried out using the anointed and written images, starting by converting the input image from RGB to grayscale, as shown in Fig. 6(a).

Then, the cv2.medianBlur function was used to remove noise from the grayscale image, as shown in Fig. 6(b), after which the median filter result was smoothed using the cv2.GaussianBlur function with a kernel size of 5, as shown in Fig. 6(c). Thereafter, the Gaussian blur result was further denoised using the cv2.fastNlMeansDenoising function, as shown in Fig. 6(d). Finally, the denoised result was binarised using the cv2.adaptiveThreshold function with a Gaussian method and the inverse binary thresholding method with a threshold value of 255, as shown in Fig. 6(e).


(a) Grayscaled (b) Enhanced (c) Blured

(d) Denoised (e) Binarized

Fig. 6. Preprocessing for anointed and written images.

C. Segmentation and Augmentation

Roboflow Datasets is a comprehensive computer vision tool that offers a range of functionalities, including dataset upload, organization, collaboration, labeling, augmentation, and processing [23]. The selection of the Roboflow Datasets tool is based on its notable ease of use, particularly in scenarios where characters appear too close to each other or when certain images exhibit characters that have been damaged. In Fig. 7, the labeling process is illustrated, followed by the implementation of a data augmentation technique involving a rotation of 10 degrees in both clockwise and counterclockwise directions. This technique was chosen for its recognized capacity to diversify the dataset, introducing variations in orientation that enhance recognition accuracy.

Subsequently, the dataset was divided into two sections, with 80% allocated for training and 20% for testing. The training set consisted of 11,103 images, while the test set comprised 2,763 images. Fig. 8 illustrates the distribution of the dataset across 29 classes. To ensure consistency, all images in the dataset were standardized to a size of 224 x 224 pixels.

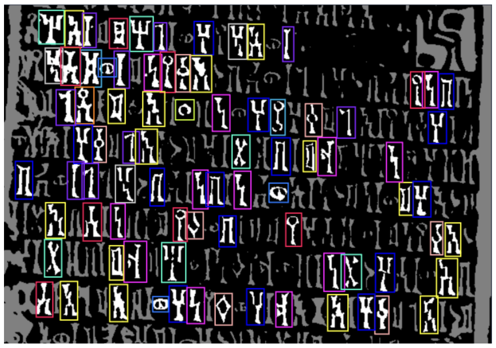


Fig. 7. Segmentation result.

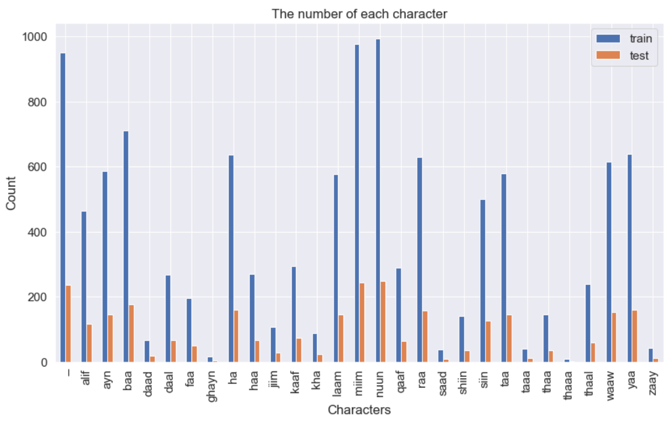


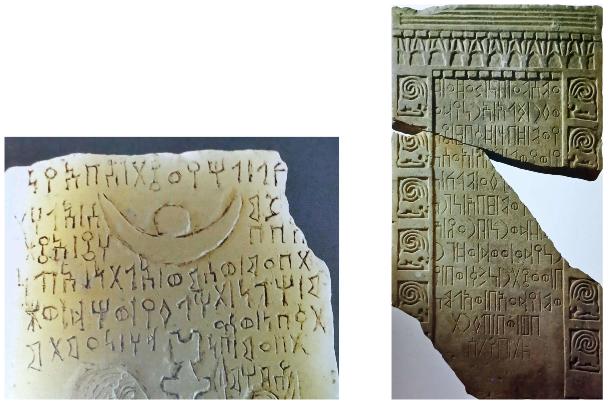
Fig. 8. Musnad characters classes.

D. Challenges and Limitations

Recognising the Musnad font involves challenges for several reasons. First, the segmentation process requires manual intervention, as it becomes difficult due to complex structures, as several images contain illustrations, as shown in Fig. 9(a). Further, some characters were too close to each other, some were not on the same line and some were cut off due to broken backgrounds, as shown in Fig. 9(b). Furthermore, Fig. 10(b) displays the results of preprocessing, where certain characters lack clarity due to varied surface conditions, as depicted in Fig. 10(a).

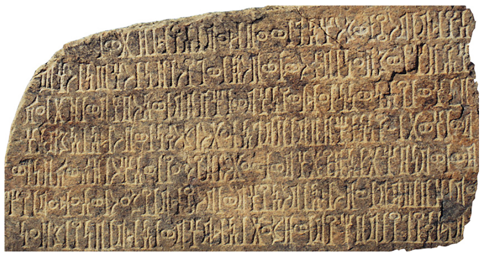
IV. Methodology

Here, the key steps and techniques implemented in the work to accomplish the desired outcomes. First, an image

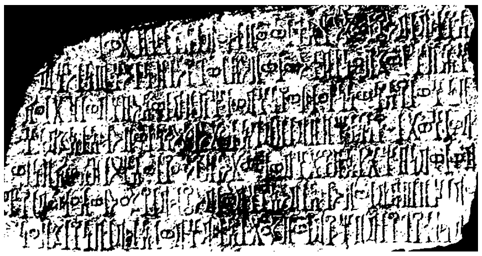


(a) Inscription containing illustrations (b) Inscription with a broken background

Fig. 9. Dataset sample.



(a) Before preprocessing



(b) After preprocessing

Fig. 10. Surface condition: before and after preprocessing.

is captured and uploaded, after which the system implements image preprocessing. Third, the system detects and recognises the correct Musnad characters, and finally, the Musnad characters are translated into Arabic characters. For reference, Fig. 11 shows the proposed workflow of the system. In subsequent sections, each of these processes will be discussed in detail, providing a comprehensive understanding of the methodology employed.

A. Image Preprocessing

The process for preprocessing natural scene images mentioned in III-B1 will be applied to the input image, with the addition of a new dilation process by utilising the cv2.dilated function.

B. Text Detection and Recognition

1) Contour Detection: Contour detection is a method often used in computer vision and image processing to detect and

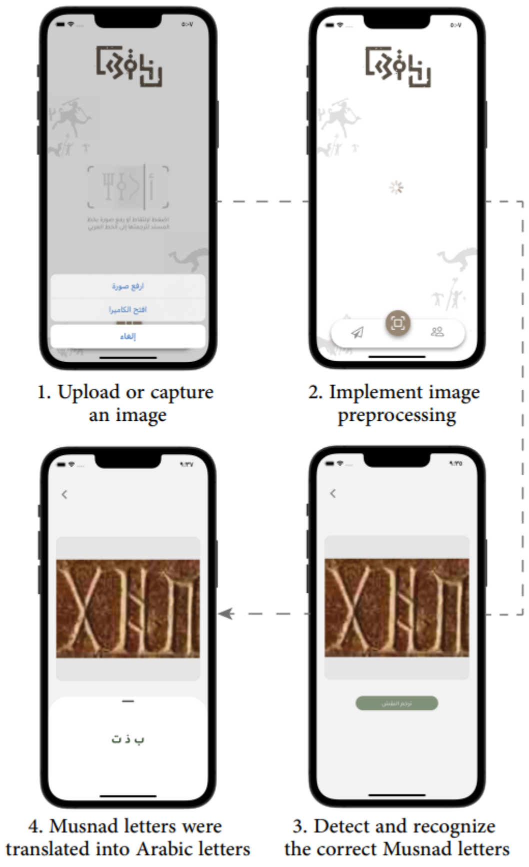

1. Upload or capture an image
2. Implement image preprocessing
3. Detect and recognize the correct Musnad letters
4. Musnad letters were translated into Arabic letters

Fig. 11. Workflow of the system.

find the outlines of objects in an image [24]. Unfortunately, the project's pretrained recognition models were only capable of correctly identifying one character, as they treated the image as containing only one character, presneting an obstacle in the recognition process. This project used a method for dealing with this problem by utilising the OpenCV function, which provides two functions: findContours and drawContours. The contouring process utilised herein is findContours, which detects the contours of the inscription to recognise all the characters of the Musnad inscriptions [25].

The findContours function requires three arguments: IMAGE, RETR_EXTERNAL and CHAIN_APPROX_SIMPLE, all of which were carefully chosen to optimise the contour detection process. By applying this method, the Musnad inscriptions in the image were successfully outlined, as demonstrated in Fig. 12.

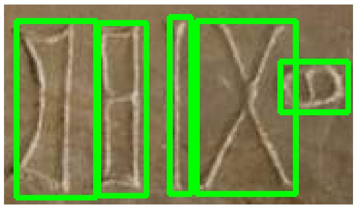


Fig. 12. Contour detection example.

2) *Deep Learning Models:* CNN is a highly efficient technique for image classification, and it is widely used in many recognition problems. CNN can perform both feature extraction and classification [26]. However, deep learning algorithms typically require more time and data than conventional machine learning systems offer to achieve an optimal performance [27]. As such, transfer learning is a technique that leverages a model pretrained on a specific dataset and that adjusts its parameters to suit new datasets. This approach is more efficient than creating a new CNN model from scratch [27], but one challenge that can arise is overfitting. In this case, the model becomes too specialised and cannot be adapted to new data [27]. To overcome this issue in this paper, early stopping and dropout techniques have been employed. Thus, this section presents how to employ the concept of transfer learning using feature extraction with VGG16, ResNet-50 and MoileNetV2, which were chosen due to their proven effectiveness in identifying ancient inscriptions [21], [6], [19].

VGG16: VGG16 is a CNN model for image classification developed by the Visual Geometry Group (VGG), comprising 16 layers in total, including 13 convolutional layers and three fully connected layers, using only 3×3 convolutional layers stacked atop each other [28]. The VGG16 model in this project is composed of two parts: a pretrained VGG16 model and additional custom layers added atop the pretrained model. Further, the VGG16 model is loaded as a pretrained model with the input shape set to (224,224,3), the top layer set to 'false' and the weights set to 'none'. By setting the top layer to false, the VGG16 model's fully connected layers are not included, allowing the model to be used as a feature extractor. After loading the pretrained model, a new sequential model is created, and the pretrained model is added as the first layer. The output of the pretrained VGG16 model is flattened using a flattened layer, after which the model has two fully connected Dense layers with 256 and 512 neurons, respectively. Batch normalisation is then applied after each Dense layer, followed by the rectified linear unit (ReLU) activation function. Dropout with a rate of 0.25 was applied after each activation function, producing an output Dense layer with 29 classes and a softmax activation function.

ResNet50: ResNet50: ResNet-50 is a CNN model with 50 convolutional layers that contain residual blocks, as well as approximately 25.6 million parameters [29]. Both the ResNet-50 and VGG16 models underwent the same modification to their fully connected layer (FCL) architecture, utilising the same parameter values.

MobileNetv2: MobileNetV2 is a CNN model designed to be fast and efficient to reduce the large network size and to minimise the cost of network computing [30]. The MobileNetV2 has undergone the same modifications as the VGG16 and Resnet-50 models, but they resulted in an underwhelming performance. Therefore, to improve the results, we altered the FCL modifications to differ from those applied to the VGG16 and ResNet-50 models to enhance its effectiveness.

The MobileNetV2 model is comprised of two parts: a pre-trained MobileNetV2 model and additional custom layers added atop the pre-trained model. In addition, the pre-trained MobileNetV2 model is loaded with ImageNet weights, and the input shape is set to (224,224,3) with the top layer set to 'false'. A new sequential model is created, with the pretrained MobileNetV2 model being added as the first layer. Then, the output from the MobileNetV2 model is processed through a GlobalAveragePooling2D layer, after which it is flattened using a Flatten layer. It is then fed into three Dense FCLs with 256 neurons and a ReLU activation function, followed by a Dropout layer with a rate of 0.25. Finally, the output layer is a Dense layer with 29 classes and a softmax activation function.

Experimental Setup and Conditions: The experiments were performed on a computer system equipped with the Windows 11 Pro operating system, an Intel Core i5 11400 central processing unit (CPU) and an Intel(R) UHD Graphics processing unit (GPU). In addition, the machine learning framework used was Keras 2.11.0, which was run using Python 3.9.12 in Spyder Anaconda.

C. Evaluation Metrics

Evaluation metrics are used to measure the performance of a classifier with a dataset, providing a way of measuring the model's performance in making correct predictions. The confusion matrix provides information about a predictive model's performance, and various metrics can be derived therefrom to gain deeper insights. These metrics include accuracy, precision, recall, F1 score and the confusion matrix itself [31]. Below are the evaluation metrics used to evaluate the models herein:

Confusion matrix: Compares predictions made by the trained classifier to the true labels in the test set. Correctly detected results are shown in diagonal cells, while incorrect predictions are shown in the remaining cells [32].

Accuracy: This metric establishes the model's accuracy in making predictions, as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

Precision: Indicates the proportion of correctly positive predictions of all positive predictions. This metric establishes the reliability of the predictive model in making accurate predictions, as follows:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall (also known as Sensitivity or True Positive Rate): Applies the same principle as precision, but instead of focusing on false positives, it focuses on false negatives.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

F1-Score: This metric assesses the accuracy with which the predictive model can predict positive values while accounting for both FN and FP. It provides a balance between the two measures by taking the harmonic mean of precision and recall [31].

$$F = \frac{2TP}{2TP + FP + FN} \quad (4)$$

V. Result and Discussion

We evaluated the performance of pretrained models, namely VGG16, ResNet50 and MobileNetV2, in recognising Musnad inscription characters. The VGG16 model underwent training for 63 epochs with early stopping enabled, which took approximately 12.42 hours to complete. Similarly, the ResNet50 model was trained for 78 epochs, which required approximately 8.08 hours of training time. Finally, the MobileNetV2 model underwent training for 149 epochs, which required approximately 5.38 hours to train. For all models, the learning parameters remained consistent, including a learning rate of 0.00001, 150 epochs with early stopping and a batch size of 32.

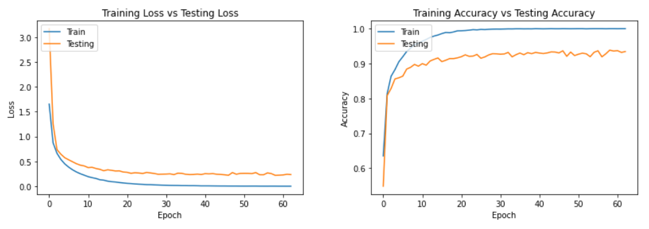
The model evaluations were based on various metrics, as mentioned in IV-C, and the results are summarised in Table I. The VGG16 model demonstrated the best performance among the tested models, achieving an average accuracy recognition rate of 93.81% across 29 character classes. Both the ResNet50 and MobileNetV2 models also showed excellent accuracy results, although they were slightly lesser compared to the VGG16 model.

Fig. 13 demonstrates the convergence and effectiveness of the VGG16 model by illustrating its loss and accuracy during the training and testing phases, providing valuable insights into the model's performance. Likewise, Fig. 14 and Fig. 15 display the loss and accuracy trends of the ResNet50 and MobileNetV2 models, respectively, highlighting their training progress and overall performance and contributing a comprehensive view of how these models perform over time. Moreover, Fig. 16 presents the confusion matrix for all three models, providing an insightful visualisation of their classification performance.

These findings suggest VGG16 can effectively recognise Musnad inscription characters with high accuracy. As such, the results demonstrate the potential of this model in the field of character recognition and provide valuable insights for further improvements and applications.

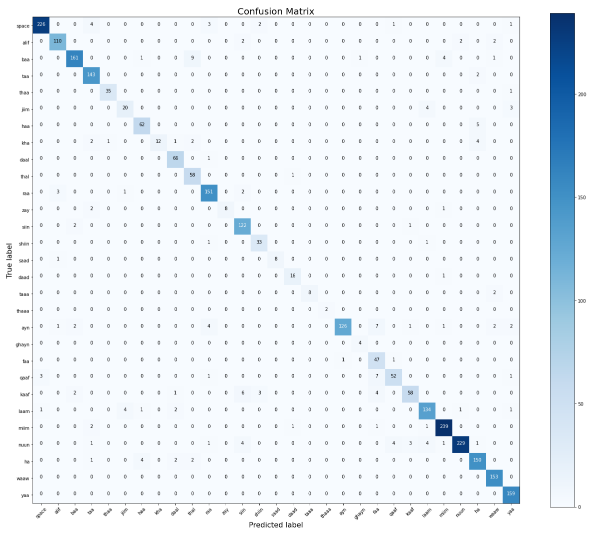
TABLE I. Transfer Learning Results

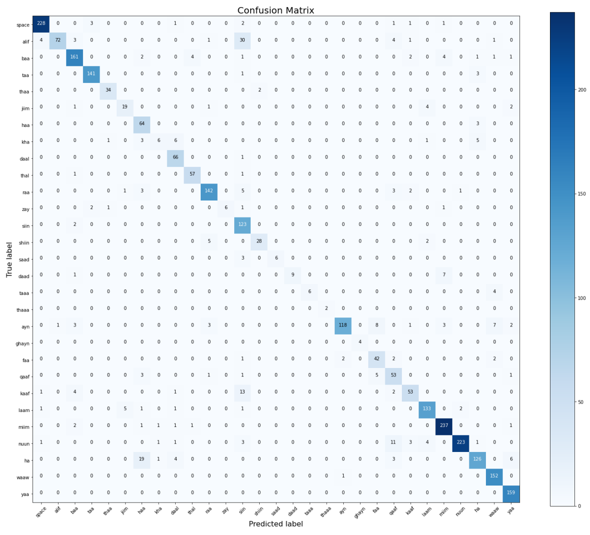
	Precision	Recall	F1-score	Accuracy
VGG16	93.00	91.00	91.00	93.81
ResNet50	90.00	83.00	85.00	89.39
MobileNetV2	73.00	77.00	74.00	80.02



(a) Training Loss vs Testing Loss (b) Training Accuracy vs Testing Accuracy

Fig. 13. Loss and accuracy plot for VGG16.


(a) Confusion matrix for VGG16 model


(b) Confusion matrix for ResNet50 model

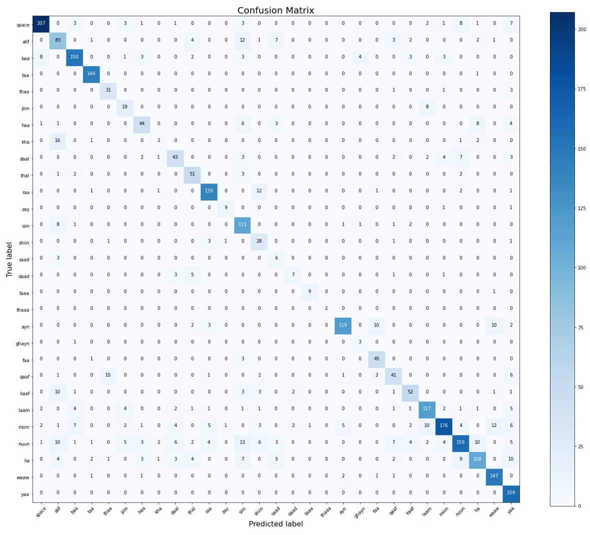
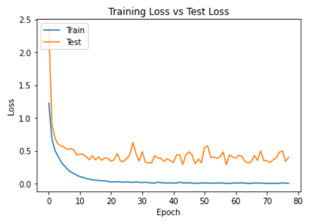
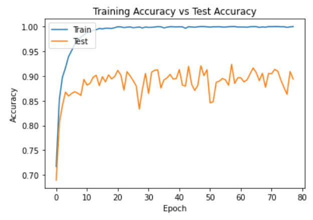

(c) Confusion matrix for MobileNetV2 models

Fig. 16. Confusion matrix for all three models.

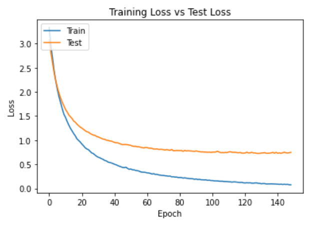


(a) Training Loss vs Testing Loss

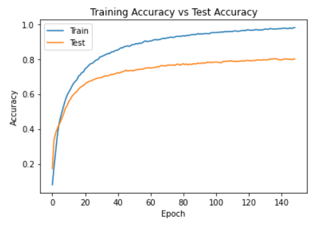


(b) Training Accuracy vs Testing Accuracy

Fig. 14. Loss and accuracy plot for ResNet50.



(a) Training Loss vs Testing Loss



(b) Training Accuracy vs Testing Accuracy

Fig. 15. Loss and accuracy plot for MobileNetV2.

VI. Conclusion

This paper automated the recognition and translation of Musnad inscriptions, making a unique contribution to ancient text recognition and translation by creating a Musnad inscription dataset, developing a recognition and translation system and comparing three CNN models (VGG16, MobileNetv2 and ResNet-50). The paper's findings indicate that real-scene images of the language's inscriptions can form a useful dataset, that the optimal sequence of image processing techniques to improve recognition accuracy and that the VGG16 deep learning model provides the highest accuracy, with a recognition rate of 93.81%. Further, ResNet-50 achieved a recognition rate of 89.39% and MobileNetV2 a rate of 80.02%. In the future, the objective will be to enhance the recognition method for practical use in real-world scenarios. As such, the focus will eventually shift towards addressing broken inscriptions and recognising old drawings, ensuring the effectiveness of techniques in challenging scenarios and with unseen or difficult-to-read inscriptions, including those with varying levels of degradation, different lighting conditions or unconventional surfaces. This enhancement will be achieved by enlarging the datasets for model training, improving image processing to generate a high-quality dataset and using NLP to determine the Arabic meanings of the words in the Musnad inscriptions.

Acknowledgment

We would like to thank the Saudi Ministry of Culture for providing us with the Musnad inscription images, which have been essential to creating our dataset. As well, we deeply appreciate the Ministry's support in preserving and sharing these valuable resources.

References

[1] A. Saleh, “History of the arabian peninsula,” 2010.

[2] S. Alyaarubi, “A brief study on musnad font,” 2013.

[3] I. bin Nasser Al-Braihi, “Crafts and industries in the light of south musnad inscriptions,” 2000.

[4] A. bin Ali Abu Hadra, “In the language of the yemeni people,” 2013.

[5] R. Santoso, Y. K. Suprapto, and E. M. Yuniarno, “Kawi character recognition on copper inscription using yolo object detection,” in *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*. IEEE, 2020, pp. 343–348.

[6] Y. Fujikawa, H. Li, X. Yue, C. Aravinda, G. A. Prabhu, and L. Meng, “Recognition of oracle bone inscriptions by using two deep learning models,” *International Journal of Digital Humanities*, pp. 1–15, 2022.

[7] M. Rajkumar, R. A. B., and v. Janhavi, “Analyzing different algorithms and techniques to find optical character recognition for tamil scripts,” 2020.

[8] A. N. Kumar and G. Geetha, “Character recognition of ancient south indian language with conversion of modern language and translation,” *Caribb. J. Sci*, vol. 53, no. 20, pp. 2019–2031, 2019.

[9] P. Ravi, C. Naveena, and Y. Sharathkumar, “Ocr for historical kannada documents using clustering methods,” *Indian Journal of Science and Technology*, vol. 13, no. 35, pp. 3652–3663, 2020.

[10] M. Merline Magrina and M. Santhi, “Ensemble classifier system for offline ancient tamil character recognition,” *SSRG International Journal of Electronics and Communication Engineering (SSRG-IJECE)*, 2019.

[11] M. Wang, Y. Cai, L. Gao, R. Feng, Q. Jiao, X. Ma, and Y. Jia, “Study on the evolution of chinese characters based on few-shot learning: From oracle bone inscriptions to regular script,” *PloS one*, vol. 17, no. 8, p. e0272974, 2022.

[12] S. Zhai, L. Liao, Y. Lin, and L. Lin, “Inscription detection and style identification in chinese painting,” in *2020 Chinese Automation Congress (CAC)*. IEEE, 2020, pp. 7434–7438.

[13] S. Chadha, S. Mittal, and V. Singhal, “Ancient text character recognition using deep learning,” *International Journal of Engineering Research and Technology*, vol. 3, no. 9, pp. 2177–2184, 2020.

[14] M. Munivel and V. Enigo, “Optical character recognition for printed tamizhi documents using deep neural networks.” *DESIDOC Journal of Library & Information Technology*, vol. 42, no. 4, 2022.

[15] S. Subadivya, J. Vigneswari, M. Yaminie, and M. Diviya, “Tamilbrahmi script character recognition system using deep learning technique,” *International Journal of Computer Science and Mobile Computing*, vol. 9, no. 6, pp. 114–119, 2020.

[16] S. R. Narang, M. Kumar, and M. K. Jindal, “Deepnetdevanagari: a deep learning model for devanagari ancient character recognition,” *Multimedia Tools and Applications*, vol. 80, no. 13, pp. 20 671–20 686, 2021.

[17] P. Bannigidad and C. Gudada, “Historical kannada handwritten scripts recognition system using line segmentation with lbp features.”

[18] N. Gautam, S. S. Chai, and J. Jose, “Recognition of brahmi words by using deep convolutional neural network,” 2020.

[19] S. Luo, “Tablet inscription recognition based on the convolutional neural network,” *Journal of Chinese Writing Systems*, vol. 6, no. 3, pp. 185–197, 2022.

[20] S. Dhivya and U. G. Devi, “Tamizhi: Historical tamil-brahmi script recognition using cnn and mobilenet,” *ACM TRANSACTIONS ON ASIAN AND LOW-RESOURCE LANGUAGE INFORMATION PROCESSING*, vol. 20, no. 3, 2021.

[21] K. N. Wijerathna, R. Sepalitha, T. Indika, H. Athauda, P. Suranjini, J. Silva, and A. Jayakodi, “Recognition and translation of ancient brahmi letters using deep learning and nlp,” in *2019 International Conference on Advancements in Computing (ICAC)*. IEEE, 2019, pp. 226–231.

[22] A. Makiash, “Southern arabic script: Its origins, spread, and relationship with northwestern arabic script,” vol. 12, no. 12, pp. 338–366, 2009.

[23] Roboflow, “Roboflow docs,” https://docs.roboflow.com/, 2023.

[24] H. Sidhwa, S. Kulshrestha, S. Malhotra, and S. Virmani, “Text extraction from bills and invoices,” in *2018 international conference on advances in computing, communication control and networking (ICACCCN)*. IEEE, 2018, pp. 564–568.

[25] P. Manuaba and K. A. T. Indah, “The object detection system of balinese script on traditional balinese manuscript with findcontours method,” *Matrix: Jurnal Manajemen Teknologi dan Informatika*, vol. 11, no. 3, pp. 177–184, 2021.

[26] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, “Deep learning for visual understanding: A review,” *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[27] J. Pardede, B. Sitohang, S. Akbar, and M. L. Khodra, “Implementation of transfer learning using vgg16 on fruit ripeness detection,” *Int. J. Intell. Syst. Appl*, vol. 13, no. 2, pp. 52–61, 2021.

[28] S. Tammina, “Transfer learning using vgg-16 with deep convolutional neural network for classifying images,” *International Journal of Scientific and Research Publications (IJSRP)*, vol. 9, no. 10, pp. 143–150, 2019.

[29] V. S. Dhaka, S. V. Meena, G. Rani, D. Sinwar, M. F. Ijaz, and M. Woźniak, “A survey of deep convolutional neural networks applied for prediction of plant leaf diseases,” *Sensors*, vol. 21, no. 14, p. 4749, 2021.

[30] S. METLEK, “Disease detection from cassava leaf images with deep learning methods in web environment,” *International Journal of 3D Printing Technologies and Digital Industry*, vol. 5, no. 3, pp. 625–644, 2021.

[31] S. Kok, A. Azween, and N. Jhanjhi, “Evaluation metric for crypto-ransomware detection using machine learning,” *Journal of Information Security and Applications*, vol. 55, p. 102646, 2020.

[32] Z. Omiotek and A. Kotyra, “Flame image processing and classification using a pre-trained vgg16 model in combustion diagnosis,” *Sensors*, vol. 21, no. 2, p. 500, 2021.