Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.048356

ARTICLE

KurdSet: A Kurdish Handwritten Characters Recognition Dataset Using Convolutional Neural Network

Sardar Hasen Ali* **and Maiwan Bahjat Abdulrazzaq**

Computer Science Department, University of Zakho, Duhok, Iraq

*Corresponding Author: Sardar Hasen Ali. Email: sardar.ali@uoz.edu.krd

Received: 05 December 2023 Accepted: 14 February 2024 Published: 25 April 2024

ABSTRACT

Handwritten character recognition (HCR) involves identifying characters in images, documents, and various sources such as forms surveys, questionnaires, and signatures, and transforming them into a machine-readable format for subsequent processing. Successfully recognizing complex and intricately shaped handwritten characters remains a significant obstacle. The use of convolutional neural network (CNN) in recent developments has notably advanced HCR, leveraging the ability to extract discriminative features from extensive sets of raw data. Because of the absence of pre-existing datasets in the Kurdish language, we created a Kurdish handwritten dataset called (KurdSet). The dataset consists of Kurdish characters, digits, texts, and symbols. The dataset consists of 1560 participants and contains 45,240 characters. In this study, we chose characters only from our dataset. We utilized a Kurdish dataset for handwritten character recognition. The study also utilizes various models, including InceptionV3, Xception, DenseNet121, and a custom CNN model. To show the performance of the KurdSet dataset, we compared it to Arabic handwritten character recognition dataset (AHCD). We applied the models to both datasets to show the performance of our dataset. Additionally, the performance of the models is evaluated using test accuracy, which measures the percentage of correctly classified characters in the evaluation phase. All models performed well in the training phase, DenseNet121 exhibited the highest accuracy among the models, achieving a high accuracy of 99.80% on the Kurdish dataset. And Xception model achieved 98.66% using the Arabic dataset.

KEYWORDS

CNN models; Kurdish handwritten recognition; KurdSet dataset; Arabic handwritten recognition; DenseNet121 model; InceptionV3 model; Xception model

1 Introduction

Handwritten recognition stands as a prominent domain within artificial intelligence. Despite over three decades of digitization progress, the advancement in recognition techniques remains rapid. Handwriting analysis poses one of the classification challenges in artificial intelligence, capturing the interest of various experts, including computer scientists and specialists in handwriting. This technology significantly simplifies data storage for users, eliminating the need for manual typing in text documents. Such recognition processes are crucial within image recognition. According to Rajyagor et al. [1], handwriting recognition continues to be a fascinating and vital field within artificial

intelligence. Its integration into digitization for over three decades has streamlined data storage, saving users considerable time and effort in document handling. This recognition process plays an integral role within image recognition methodologies [2]. Within handwriting recognition, Recurrent Neural Networks (RNNs), CNNs and their advanced iterations play pivotal roles in mechanical systems and signal-processing applications. In mechanical systems, they excel in fault diagnosis and predictive maintenance, leveraging their hierarchical feature learning stated by Baldominos et al. [3]. Long Short-Term Memory (LSTM), alongside various types of CNNs, has exhibited promising outcomes. Notably, CNNs deliver more precise results than alternative learning methods, as corroborated in research by Shamim et al. [4], revealing CNNs' use in image processing, Natural Language Processing (NLP) by Alayyoub et al. [5], and sentiment analysis. Recognizing human hand formations is an incredibly intricate task that necessitates a robust hand-tracking approach reliant on efficient features and classifiers. Deciphering human handwriting is an exceedingly challenging research endeavor that demands a strong hand-tracking system based on effective features and classifiers. Presently, the most popular and viable methodologies involve machine learning techniques, such as CNNs, RNNs Multilayer Perceptions (MLPs), Support Vector Machine algorithm (SVM), and K-Nearest Neighbors algorithm. This work by Lecun et al. [6] refers to the brief description of the application of CNN. Here, CNN is described shortly as employed in this work.

CNNs, also known as ConvNets, represent multi-layered neural networks primarily utilized for image processing and object detection. CNNs consist of multiple layers that process, analyze, and extract features from data. They utilize a convolutional layer equipped with numerous filters to execute convolution operations. Additionally, the Rectified Linear Unit (ReLU) layer is utilized to derive a rectified feature map (RFM) from processing elements. The RFM is then directed to a pooling layer, which reduces the dimensionality of the feature map through a down-sampling process. Subsequently, the resulting two-dimensional arrays from the pooled feature map are flattened to form a single, extended, continuous linear vector. The primary challenges in handwritten recognition revolve around inaccuracies and inconsistent patterns, underscoring the pivotal role of feature extraction. However, manually selecting features, as emphasized by researchers Pasi et al. [7], may result in insufficient data to accurately predict class characteristics, while an excessive number of features may lead to issues due to increased dimensionality [8]. This study's key contribution is the comprehensive exploration of both deep learning and machine learning for handwritten recognition, consolidating insights from various studies in the field. The survey encompasses recent works in handwriting recognition proposed by different researchers, presenting diverse methodologies and comparing the accuracy of state-of-the-art techniques.

Presently, Arabic stands as one of the most prevalent international languages across the Middle East and North Africa. It holds a prominent position as one of the most widely spoken languages globally, with a speaker population exceeding half a billion, as stated by Altwaijry et al. [9]. Additionally, various languages incorporate Arabic letters, words, and sentence structures into their linguistic frameworks [10]. The standardized written form of the Arabic language comprises 28 letters, traditionally written from right to left [11]. It is noteworthy that certain letters may exhibit distinct appearances depending on whether they appear at the beginning, middle, or end of a word, signifying that the letter's shape undergoes alterations based on its position within the word [12]. As technological progress continues, the potential uses for Arabic handwriting recognition become exceedingly vast. The current era is particularly delightful for this domain, given the swift advancements in machine learning and deep neural networks. These ongoing developments are poised to result in further significant breakthroughs shortly stated by Alayyoub et al. [5]. The successful implementation of Arabic handwritten recognition has been accomplished through diverse machine learning methodologies

Maytham et al. [13]. Nevertheless, the integration of deep learning (DL) architectures stands to significantly elevate AHR technology. Notably, the utilization of CNNs in AHR models has yielded commendable results in the accurate recognition of handwritten text. We have created the Kurdish handwritten dataset (KurdSet) in response to the absence of any pre-existing datasets in the Kurdish language. The creation of this dataset represents a significant contribution to the field, addressing resources for Kurdish language-based studies and applications.

The main contribution between Arabic and Kurdish character datasets exhibits similarities in certain characters, despite differing sources of acquisition. The Arabic character dataset AHCD is derived from either native Arabic speakers or written materials, whereas the Kurdish character dataset is sourced from a distinct cohort of Kurdish participants. Notably, we created the KurdSet dataset, named KurdSet, which was prompted by the absence of any existing Kurdish dataset in the academic literature. In the evaluation of deep learning models on the Kurdish dataset, a comparative analysis of the two approaches has not been conducted in prior research. Our KurdSet dataset is now available for use by individuals or organizations interested in developing Kurdish language systems. This dataset can serve as a valuable resource for training and evaluating models in various applications related to the Kurdish language. To address this gap, we employed various CNN models, including InceptionV3, Xception, DenseNet121, and custom CNN models. The primary objective of this study is to assess and compare the performance of these two datasets.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 reports the methodology of the study while Section 4 discusses the results and discussion of the proposed methods and the last Section 5 explains the conclusion of the work.

2 Related Work

Numerous researchers have attempted to create models for recognizing handwritten text in the past, but none have achieved flawless accuracy. Further research in this field is still necessary to improve the technology.

The study by Alheraki et al. [14] advances Arabic handwritten character recognition through deep learning techniques, addressing intricacies in the challenging Hijja dataset featuring children's handwriting. By integrating augmentation with the AHCD dataset and employing a strokes-based approach, the proposed model surpasses baseline accuracy on both Hijja and AHCD datasets, with enhanced performance upon their amalgamation. Contributing significantly to the progression of Arabic handwritten character recognition systems, the model achieves commendable accuracy of 91% on the Hijja dataset and an impressive 97% on the AHCD. While, an innovative multi-model approach, incorporates stroke count insights, resulting in a noteworthy 96% average prediction accuracy upon merging the Hijja and AHCD datasets. A study conducted by Alrobah et al. [15] introduced a novel approach in their research for recognizing Arabic handwritten characters. They utilized a hybrid model merging CNN with SVM and extreme Gradient Boosting (XGBoost) classifiers. The study identified inconsistencies in the widespread Arabic dataset, Hijaa, previously employed for CNN-based feature extraction and classification, achieving an accuracy of 88%. The authors tackled dataset imbalances by employing a hybrid model that leveraged CNN for feature extraction and Machine Learning (ML) models for classification. This resulted in an enhanced accuracy of 96.3%, marking an 8% increase over the original Hijaa experiment. The hybrid's model performance was similar to the one conducted on the AHCD dataset using the Hijaa model, demonstrating its efficacy.

The study conducted by Ghanim et al. [16] introduced a multi-phase sequential method for recognizing offline Arabic handwriting, which underwent testing using the IFN/ENIT Arabic database.

The study separately assessed the impacts of SVM and CNN for classification. The CNN classification was based on the direct pixel values of the input image, while the SVM classification relied on a feature vector extracted from the input image. When employed on the database, this multi-stage method achieved an accuracy of 95.6% using the AlexNet CNN architecture without necessitating character-level segmentation. Awni et al. [17] compared randomly initialized deep convolutional neural networks with a pre-trained ResNet18 model from ImageNet for Arabic handwritten word recognition. They introduced a sequential transfer learning approach using the pre-trained ResNet18 model to enhance recognition accuracy on the AlexU-W and IFN/ENIT datasets. Utilizing the ImageNet data improved recognition accuracy by 14%, and their proposed method achieved a notable 35.45% enhancement for frequently misclassified words in the IFN/ENIT dataset. The authors initially trained the ResNet18 model, freezing the first Conv1 layer and two subsequent residual blocks. They fine-tuned the remaining layers with the IFN/ENIT dataset at an image size of $224 * 224$, employing progressive resizing to $300 * 300$. Their approach achieved up to 96.11% accuracy, demonstrating significant progress compared to existing methods, with 99.52% accuracy on the AlexU-W dataset.

In a study conducted by Aljarrah et al. [18], the authors devised a CNN model for the recognition of Arabic letters. They incorporated various optimization techniques to enhance the model's performance, achieving a testing accuracy of 94.9% when applied to the AHCD dataset. Kamal et al. [19] used CNN architecture consisting of 18 layers, encompassing convolution, pooling, batch normalization, dropout, and culminating in a final layer of global average pooling and dense. Through evaluation of AHCD and 'Modified Arabic handwritten digits Database (MadBase),' the model achieved remarkable accuracies of 96.93% and 99.35%, respectively. These results, comparable to state-of-the-art models of their work, underscore its suitability for practical real-life applications. Almansari et al. [20] presented a deep learning architecture for the recognition of isolated handwritten Arabic letters. This architecture featured CNN and Multi-Layer Perceptron (MLP) neural networks. The performance evaluation of the model was conducted using the AHCD dataset, where it achieved an impressive testing accuracy of 95.27%. This demonstrated the capability of the deep learning architecture in accurately recognizing isolated handwritten Arabic letters.

In another study by Ullah et al. [21], the research introduces a method for recognizing handwritten Arabic letters, utilizing a CNN model. The model integrates regularization techniques, including batch normalization and dropout operations. Evaluation encompassed testing the model both with and without dropout, revealing a notable performance difference and effectively mitigating overfitting. The model was then applied to the utilized AHCD. The evaluation, considering various metrics, illustrated the model's superiority, achieving an accuracy of 96.78% and surpassing established approaches in the literature. Whereas, Alzebdeh et al. [22] presented a sophisticated deep neural network for recognizing AHCD, employing CNN models with regularization techniques like batch normalization to counter overfitting. The Deep CNN achieved high classification accuracies of 94.8% for the AIA9k dataset and 97.6% for the AHCD dataset.

3 Methodology

In this section, we present an overview of the methodology employed in the development of the proposed model. The following paragraphs provide a general idea about the models, datasets, methods, and other key aspects utilized in this study. For a comprehensive understanding, we discuss the overall approach and specific details such as the Kurdish dataset, data preprocessing, and the design of the proposed CNN, which will be elaborated upon in the subsequent sections.

3.1 Dataset

In this section, two datasets are discussed which are the Kurdish and Arabic datasets, they are used in this paper.

3.1.1 Kurdish Dataset (KurdSet)

We have developed a newly generated dataset known as the Kurdish handwritten character recognition dataset called (KurdSet), which comprises three pages of content consisting of characters, digits, texts, and symbols in the Kurdish language. Kurdish is the language spoken in the Kurdistan region. The dataset was compiled with the involvement of both males and females, ranging in age from 18 to 65 years. The participants encompassed individuals from various professional backgrounds, including employees, and students from different departments. Approximately 2000 users took part in the dataset creation process; however, certain forms were written outside the designated boxes so we excluded or disregarded those entries, resulting in a final count of 1560 participants. The specific structure of this module is illustrated in Fig. 1.

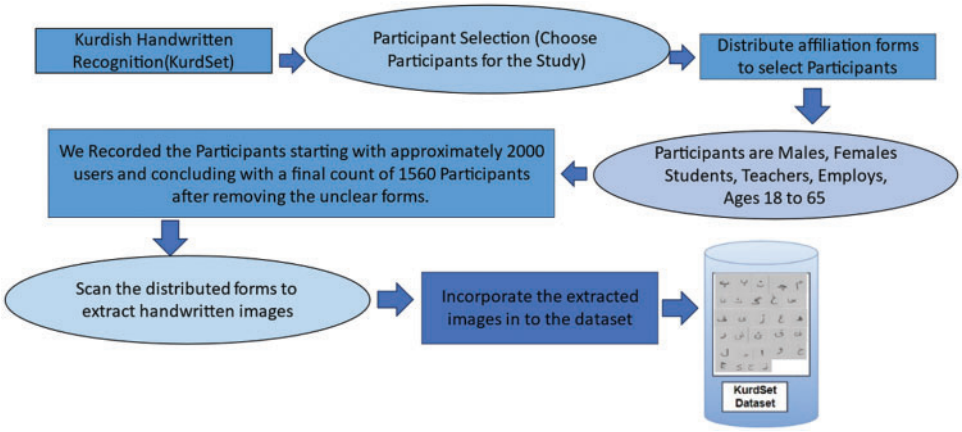


Figure 1: Flowchart of Kurdish handwritten dataset architecture

For this article, we choose the characters from the KurdSet dataset. Our database typically includes thousands of handwritten character images more than 45,000 images, where each image represents a single Kurdish character. These images are manually annotated with corresponding labels indicating the characters they represent. The database is commonly utilized for training and evaluation algorithms. and models for recognizing Kurdish characters in handwritten documents. Fig. 2 illustrates character examples of the Kurdish handwritten recognition dataset.

3.1.2 Arabic Handwritten Character Dataset

AHCD that has 16,800 Arabic characters done by 60 different participants applied to Arabic characters for recognition. The database is partitioned into two sets: A training set (13,440 characters to 480 images per class) and a test set (3360 characters to 120 images per class). The difference between these two datasets is the writers in which Kurdish handwritten dataset participants are Kurdish participants whose native language is Kurdish while Arabic handwritten characters dataset participants are Arabs. Fig. 3 shows the sample of the Arabic handwritten character dataset.

characters	names	characters	names	characters	names	characters	names
ا	aa	ڤ	veh	ع	ayn	ھ	hih
ب	beh	ر	reh	غ	gheh	و	weh
پ	peh	ڕ	keh	ئ	eh	م	meh
ح	heh	گ	geh	س	seh	ن	neh
خ	kheh	ت	teh	ش	sheh	ی	yeh
ج	jih	ز	zeh	چ	cheh		
ف	feh	ژ	zhe	ل	leh		
ق	qeh	ێ	eh	د	deh		

Figure 2: Samples of Kurdish handwritten character recognition dataset

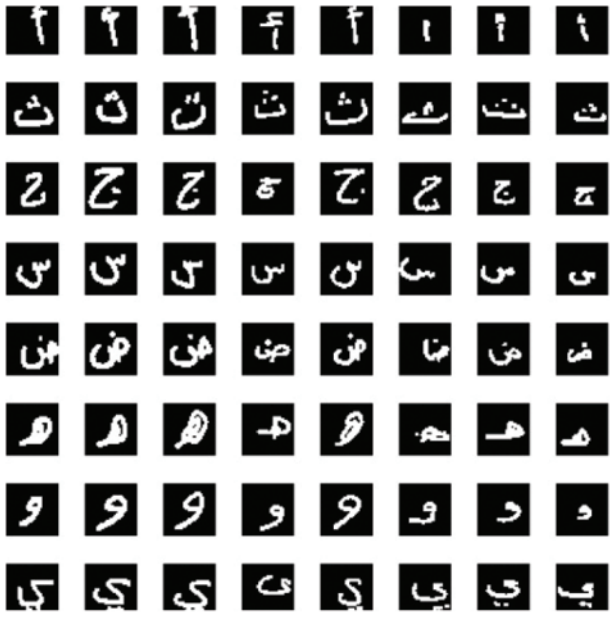


Figure 3: Samples from the AHCD dataset

3.2 Data Preprocessing

As for preprocessing techniques, we applied various methods, techniques, and operations to input images of handwritten characters before they were analyzed by a recognition system. Its purpose is to improve image quality and extract important features for accurate recognition. We implemented specific preprocessing steps. We resized the image size from 32×32 pixels to 80×80 pixels and centered them with the process of 50 epochs to enhance the models' ability to capture more features and patterns within the images, potentially leading to improved performance and accuracy. The models utilized in this paper are the custom CNN model, DenseNet121, InceptionV3, and Xception. The metric used

in these models is accuracy, using pre-trained models (DenseNet121, InceptionV3, and Xception) on ImageNet with learned features from large datasets. By preserving lower layers responsible for general low-level features, the focus can shift to fine-tuning higher layers specialized for the task at hand. This strategy conserves computational resources and mitigates overfitting, particularly when training data is scarce. Data augmentation parameters are used for enriching image-based deep learning training datasets, the parameters of data augmentation are rotation_range which introduces random rotations, width_shift_range and height_shift_range bring horizontal and vertical shifts, which apply shear transformations, and zoom_range allows random zooming. These variations enhance the model's ability to generalize across diverse data, improving its robustness and performance on unseen data.

3.3 The Utilized Models

We created a custom CNN model for our dataset, the Kurdish handwritten character dataset which contains 29 classes as an output and is divided into two primary sets: 80% is utilized for the training phase, 20% for the testing phase for the custom CNN model. The proposed custom CNN model contains thirteen layers and utilizes the ReLU activation function and max pooling. It has a final softmax activation for classifying the output classes. We employed a custom CNN feedforward architecture without skip connections, ensuring sequential layer-wise information processing in our model. The model uses hyperparameters like sparse categorical cross-entropy loss, Adam optimizer, and accuracy metric. It also has a Learning Rate Scheduler which is a callback that adjusts the learning rate during training based on a specified schedule. Besides, Early Stopping is utilized to stop training when a monitored metric has stopped improving. This may involve transfer learning or fine-tuning, where initial layers are frozen to retain pre-trained knowledge while training only the later layers on a new task. Additionally, we created a custom CNN model for Arabic handwritten character recognition, besides, similar parameters that we used in our dataset are implemented on the Arabic handwritten character dataset also. Fig. 4 explains the flowchart of the proposed method for both Arabic and Kurdish handwritten. These models DenseNet121, InceptionV3, and Xception are implemented to Kurdish and Arabic handwritten character recognition datasets having the same parameters used in the custom CNN model and then applied to Kurdish and Arabic handwritten datasets.

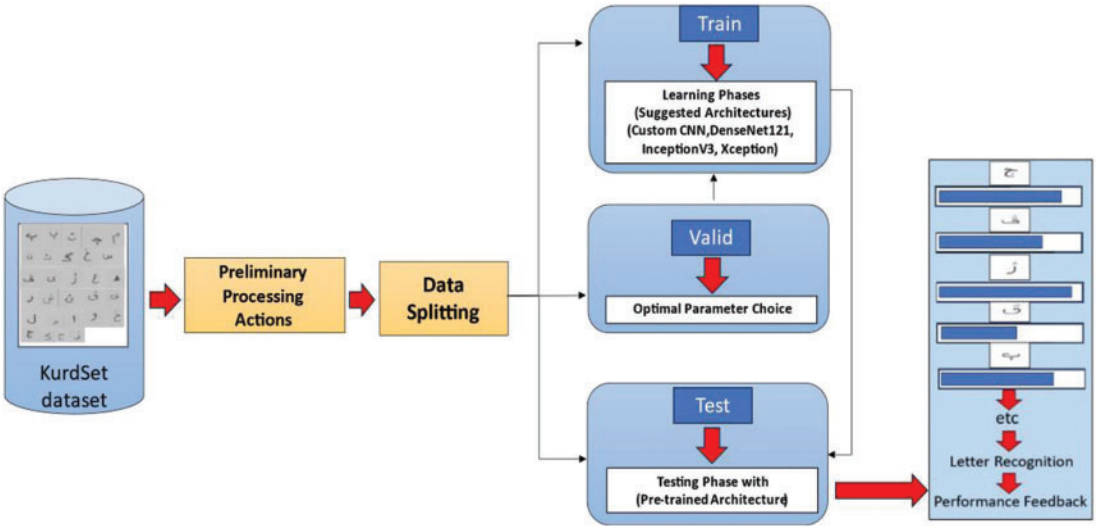


Figure 4: The flowchart of the proposed method for Arabic and Kurdish handwritten recognition

CNNs are extensively used in the analysis of visual data, such as images and videos, and are highly effective in tasks like image classification, object detection, and image segmentation utilized by Sharma et al. [23]. Inspired by the organization of the visual cortex, CNNs consist of interconnected layers that perform specialized operations on the input data. Convolutional layers are crucial in CNNs as they use learnable filters or kernels to scan and extract spatial features from the input data. The resulting feature maps capture local patterns and features of Jain et al. [24]. Pooling layers reduce spatial dimensions while preserving important information, often using max pooling to select the maximum value within a defined window. Fully connected layers connect all neurons in one layer to every neuron in the next, aggregating the extracted features for final predictions referred to by Naidu et al. [25]. Besides, Haghighi et al. [26] used CNNs that are trained with labeled data through backpropagation, where the network learns to map inputs to corresponding labels. Parameters, such as filter weights and biases, are updated based on the prediction error. Training continues until the network achieves good performance. CNN is a highly recognized and extensively utilized technique in deep learning.

DenseNet121 is a CNN architecture designed for image classification tasks and has achieved notable success in various computer vision applications. DenseNet121 refers to a DenseNet model with 121 layers The name "DenseNet" is derived from the dense connectivity pattern it employs. Unlike traditional CNN architectures, which connect adjacent layers, DenseNet connects each layer to every other layer in a feed-forward manner, this was elaborated by Ghosh et al. [27]. This dense connectivity allows for direct information flow between layers and facilitates the exchange of feature maps. As a result, DenseNet models have a strong feature reuse mechanism, enabling better gradient flow and alleviating the vanishing gradient problem often encountered in deep networks. DenseNet121 consists of a convolutional block followed by multiple dense blocks and transition layers. Its efficient parameter utilization makes DenseNet121 computationally efficient and effective for tasks like image classification and object detection [28].

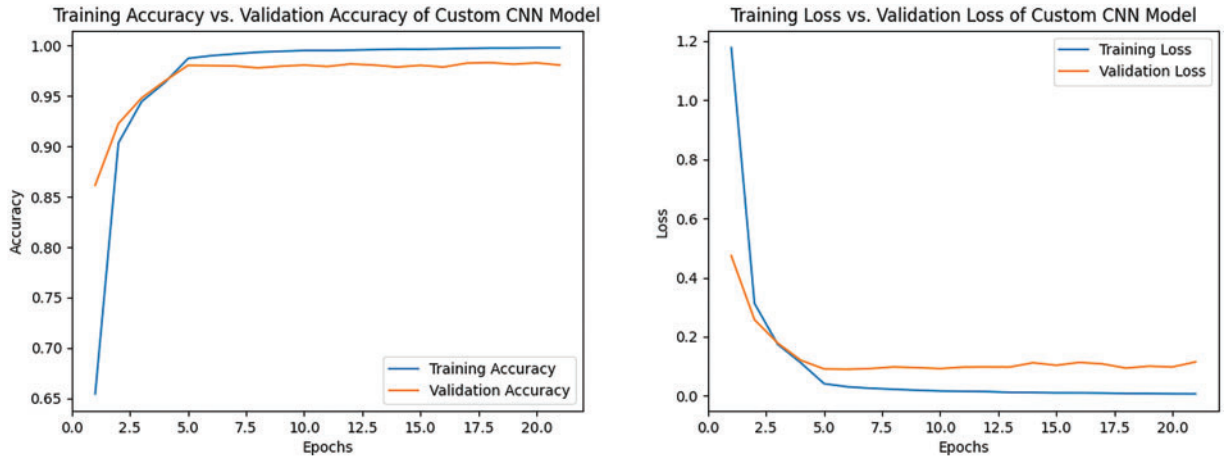
InceptionV3 is an advanced CNN designed for image classification tasks, representing a progression within the Inception architecture. Highlighted by its inception modules, the network integrates parallel convolutional operations with diverse filter sizes (1×1, 3×3, and 5×5), facilitating the extraction of multi-scale features for hierarchical pattern recognition. It is a deep neural network that includes a total of 48 layers. These layers consist of a combination of convolutional layers, pooling layers, fully connected layers, and auxiliary classifiers Uddin et al. explored [29]. Batch normalization stabilizes training, factorization techniques optimize computational resources, and auxiliary classifiers address the vanishing gradient problem during training. The meticulous instrumentation of these architectural elements positions InceptionV3 as a powerful tool for image classification, striking a balance between discerning nuanced features and computational efficiency.

Xception, an extension of the Inception architecture, is CNN known for its use of depth-wise separable convolutions. Xception has a total of 71 convolutional layers. It diverges from convolutional architectures by separating spatial and depth-wise convolutions. It replaces standard convolutions with a sequence of depth-wise convolutions followed by point-wise convolutions, reducing computational complexity while preserving expressive power for enhanced model efficiency. The utilization of depth-wise separable convolutions enables Xception to capture spatial and channel-wise dependencies optimally, resulting in a more streamlined network structure. Xception's efficacy lies in achieving a nuanced balance between model depth and computational efficiency, making it well-suited for tasks like image classification, where both accuracy and computational resources are critical considerations [30].

4 Results

4.1 Kurdish Handwritten Character Dataset Results

The results of the diagrams for the Kurdish dataset using the custom CNN model indicate a highly successful performance. The model accurately classified 99.86% of the training accuracy. The validation accuracy of 98.07% reflects the model's performance on a separate dataset not used during training, indicating its ability to generalize well to new, unseen data with accuracy. Furthermore, the test accuracy of 98.04% signifies the model's effectiveness in making accurate predictions on an independent test set. These high accuracy values across training, validation, and test sets suggest the custom CNN model has learned well from the training data and demonstrates strong generalization capabilities, making it a promising and reliable model for the given Kurdish dataset. The extremely low training loss of 0.006% indicates that the model performed exceptionally well during the training phase, effectively minimizing errors and accurately learning the patterns within the training data. On the other hand, the validation loss is slightly higher at 0.114%, suggesting that the model may face challenges in generalizing to new, unseen data not used during training. Whereas the low training loss is encouraging, the comparatively higher validation loss implies a need for careful consideration to enhance the model's generalization capabilities. Further investigation and potential adjustments may be necessary to strike a balance between effective learning from the training set and robust performance on validation and, ultimately, real-world test datasets. The results clarified in Fig. 5 provide the results of the model.

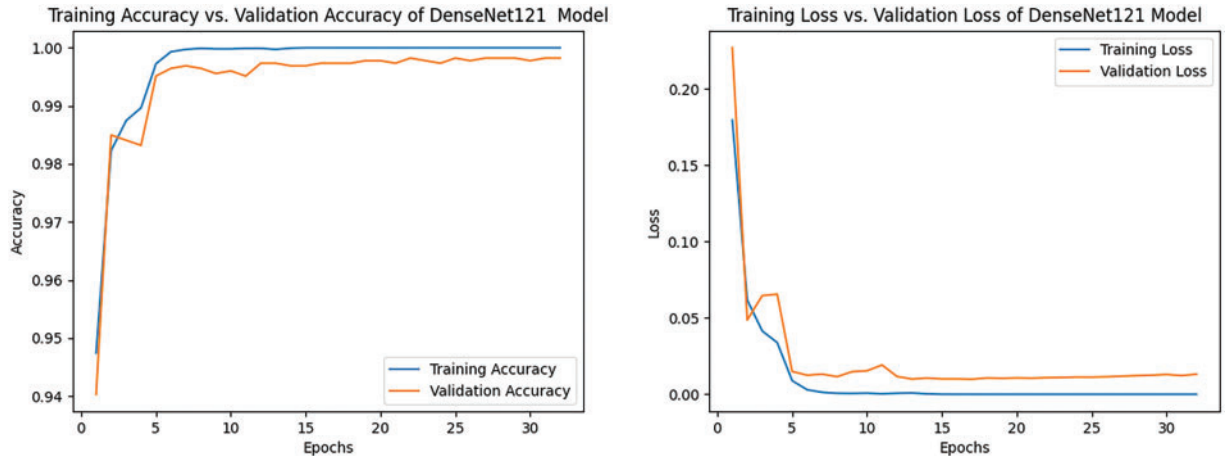


(a) (b)

Figure 5: (a) Training and validation accuracy and (b) training and validation loss of custom CNN model

The results obtained from the diagrams for the Kurdish dataset using the DenseNet121 model indicate highly promising performance. The model achieved a perfect training accuracy of 1.000%, suggesting that during the training phase, it successfully learned and memorized the features of the training dataset, achieving a flawless fit. The validation accuracy is notably high at 99.82%, demonstrating the model's ability to generalize well to new, unseen data that it did not encounter during training. This minor difference between training and validation accuracies suggests that the model has avoided overfitting and maintains a high level of accuracy on previously unseen instances. The test accuracy of 99.80% further affirms the model's robustness in making accurate predictions on an independent dataset. Overall, these results indicate that the DenseNet121 model is highly effective

and capable of achieving near-perfect accuracy across training, validation, and test datasets, making it a strong candidate for applications in the context of the Kurdish dataset. In terms of loss, the results indicate outstanding performance of training and validation loss. The low training loss of 0.0005% suggests that the model effectively minimized errors and discrepancies between its predicted outputs and the actual targets in the training dataset during the learning process. The slightly higher validation loss of 0.0130% indicates that the model's performance is still strong but is being assessed on a separate dataset not used during training. While a low training loss reflects an effective learning, the low but comparable validation loss implies that the model generalizes well to hidden data, avoiding overfitting. Overall, these results highlight the model's ability to learn patterns from the training set and apply them effectively to new data, demonstrating a well-balanced performance on the Kurdish dataset. Fig. 6 demonstrates this model's results.

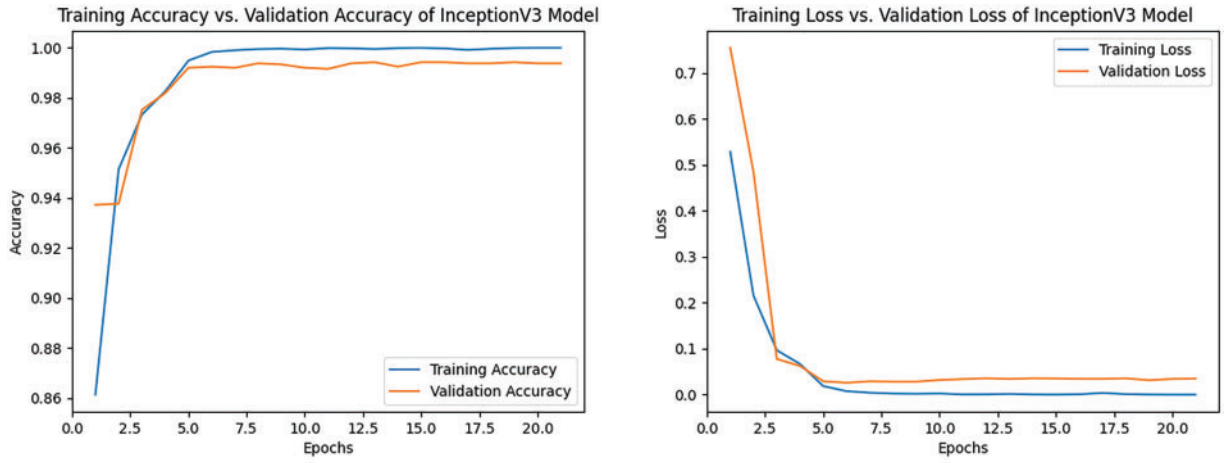


(a) (b)

Figure 6: (a) Training and validation accuracy and (b) training and validation loss of the DenseNet121 model

The results from the diagrams for the Kurdish dataset using the InceptionV3 model are highly remarkable. The model achieved a perfect training accuracy of 1.000%, indicating a flawless fit to the training data. The validation accuracy is very high at 99.41%, signifying the model's ability to generalize well to new, unseen data not encountered during training. This minor discrepancy between training and validation accuracies suggests that the model has effectively avoided overfitting and maintains a high level of accuracy on previously unseen instances. The test accuracy of 99.29% further underscores the model's reliability in making accurate predictions on an independent dataset. Overall, these results indicate that the InceptionV3 model is highly effective and capable of achieving high accuracy across training, validation, and test datasets for the KurdSet, making it a robust choice for applications in this context. The results indicate highly favorable performance. The training loss of 0.0002% is extremely low, signifying that during the training phase, the model effectively minimized errors and learned the features of the training data. The validation loss, slightly higher at 0.0350%, suggests that the model's performance remains strong. While a low training loss is encouraging, the relatively low and comparable validation loss indicates that the model generalizes well to new, unseen data, demonstrating effectiveness in preventing overfitting. These results underscore the InceptionV3 model's robustness in learning and generalizing patterns from the training set to achieve accurate

predictions on independent data in the context of the Kurdish dataset. The model's accuracy and loss are shown in Fig. 7.



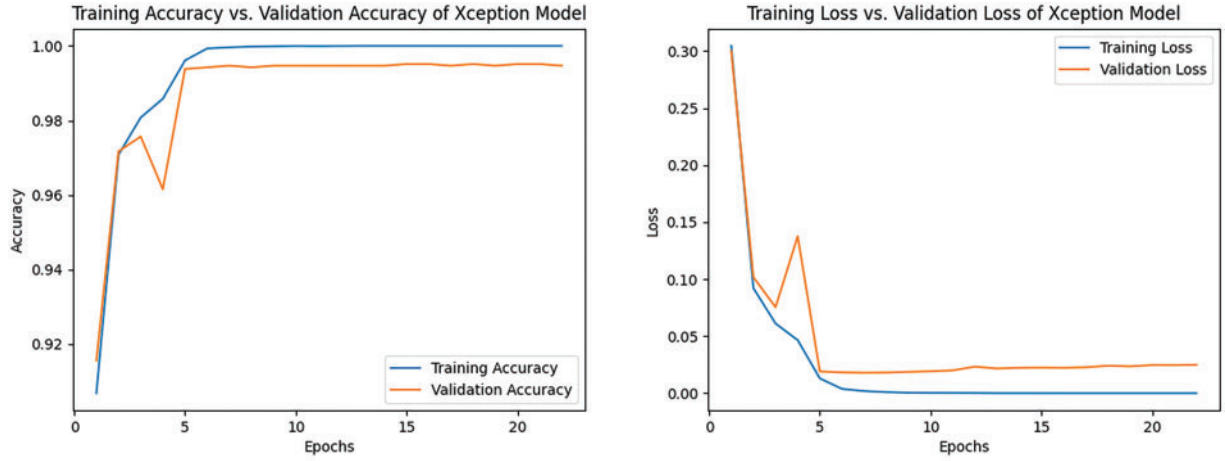
(a) (b)

Figure 7: (a) Training and validation accuracy and (b) training and validation loss of the InceptionV3 model

The results from the diagrams for the Kurdish dataset using the Xception model show excellent performance. The model achieved a perfect training accuracy of 1.000%, indicating a comprehensive understanding and memorization of the training dataset features. The validation accuracy is very high at 99.49%, demonstrating the model's capability to generalize effectively to new, unseen data not encountered during training. The minimal difference between training and validation accuracies suggests that the model has successfully avoided overfitting and maintains strong accuracy on unseen instances. The test accuracy of 99.27% highlights the model's reliability in making accurate predictions on an independent dataset. Whereas, the results reveal strong performance of training and validation losses. The low training loss of 0.0007% indicates that the model effectively minimized the difference between its predicted outputs and the actual targets in the training dataset, showcasing efficient learning and fitting to the training data. The slightly higher validation loss of 0.0254% shows when evaluated on a separate dataset not used during training, the model's performance is still robust but may face some challenges in generalization. While the training loss is low, the validation loss is relatively reasonable, indicating that the model has learned patterns from the training set and can apply them effectively to new, data. Further analysis and potential adjustments may be considered to enhance the model's generalization capabilities beyond the training set for optimal real-world performance. Fig. 8 provides details of the performance of the model.

The performance summary of the evaluated models using KurdSet dataset reveals high accuracy across various datasets. The custom CNN Model exhibits robust performance, achieving a remarkable 99.86% accuracy on the training set and maintaining high accuracy on both the validation (98.07%) and test (98.04%) sets, showcasing its ability to generalize well. DenseNet121, InceptionV3, and Xception, with perfect training accuracies of 1.000%, demonstrate exceptional learning capabilities. These models further validate their efficacy on unseen data, with DenseNet121 achieving 99.82% validation and 99.80% test accuracy, InceptionV3 achieving 99.41% validation and 99.29% test accuracy, and Xception achieving 99.49% validation and 99.27% test accuracy. The consistently high accuracies across training, validation, and test sets underscore the models' proficiency in image

classification tasks, highlighting their potential utility in real-world applications. Tables 1 and 2 depict the results of the KurdSet dataset using CNN models. The reason for using custom CNN is to have a feedforward model to check the results of a feedforward model that has the skip connection like the other models such as DenseNet121, InceptionV3, and Xception.



(a) (b)

Figure 8: (a) Training and validation accuracy and (b) training and validation loss of the Xception model

Table 1: Performance comparison of models on Kurdish dataset-training and validation accuracy results

Model	Train accuracy	Valid accuracy	Test accuracy
Custom CNN Model	99.86%	98.07%	98.04%
DenseNet121	1.000%	99.82%	99.80%
InceptionV3	1.000%	99.41%	99.29%
Xception	1.000%	99.49%	99.27%

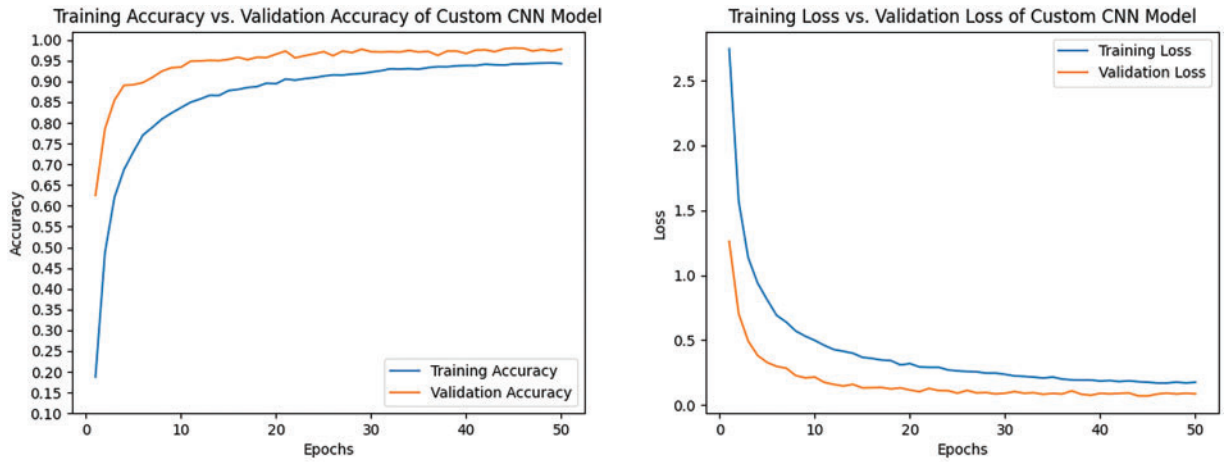
Table 2: Performance comparison of models on Kurdish dataset-training and validation loss results

Model	Train loss	Valid loss
Custom CNN Model	0.0068	0.1147
DenseNet121	0.0005	0.0130
InceptionV3	0.0002	0.0350
Xception	0.0007	0.0254

4.2 Arabic Handwritten Character Dataset Reults

Provided accuracy values indicate the performance of the model during different stages of training and evaluation in the custom CNN model in the Arabic dataset. The training accuracy of 94.30%

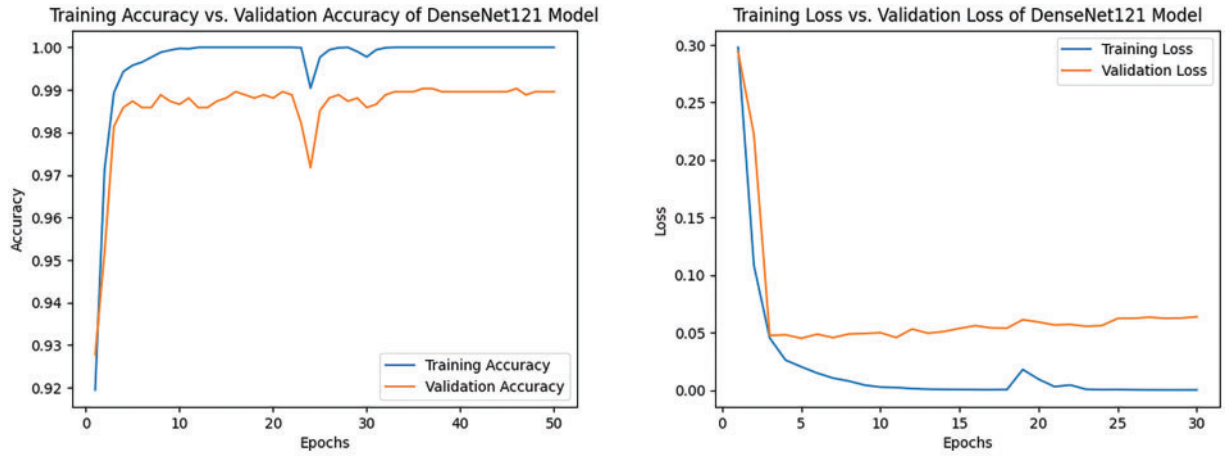
suggests that, during the training phase. On the other hand, the higher validation accuracy of 98.10 % indicates that the model performed even better when tested on a separate dataset that it had not seen during training. Finally, the test accuracy of 98.43% signifies the model's performance on an independent test set, demonstrating a high level of accuracy in making predictions. The results suggest the custom CNN model generalizes well to new, unseen data, as evidenced by the high validation and test accuracies, indicating its effectiveness in classifying the given data. CNN model exhibits a training loss of 0.175% and a validation loss of 0.086%. The training loss reflects how well the model fits the training data, and the low value indicates effective learning. However, the slightly higher validation loss suggests that the model may face challenges in generalizing to new, unseen data. This discrepancy could signal overfitting or limitations in generalization. Further investigation and adjustments to the model may be needed to enhance its overall performance and ensure robustness beyond the training dataset. The results are clarified in Fig. 9.



(a) (b)

Figure 9: (a) Training and validation accuracy (b) training and validation loss of the custom CNN model

The training accuracy for the DenseNet121 model on an AHCD dataset is 1.000%, demonstrating a precise fit to the training data. In contrast, the validation accuracy of 98.84% indicates the model's ability to generalize effectively during training, with a minimal difference between training and validation accuracies, suggesting avoidance of overfitting. The subsequent test accuracy of 98.36% affirms the robust performance on an independent dataset, showing its capability for accurate predictions beyond the training and validation phases. This underscores the well-balanced nature of the model. Furthermore, the training loss is very low at 0.0008%, indicating effective performance on the training dataset and minimal errors during the training process. In contrast, the validation loss is slightly higher at 0.063%, indicating a strong performance. While a low training loss is encouraging, a comparable validation loss signifies the model's effective generalization to hidden data, avoiding overfitting. The disparity between training and validation losses may suggest a well-balanced model that has learned patterns from the training set, effectively applying them to new, unseen data. Fig. 10 shows the outcomes.



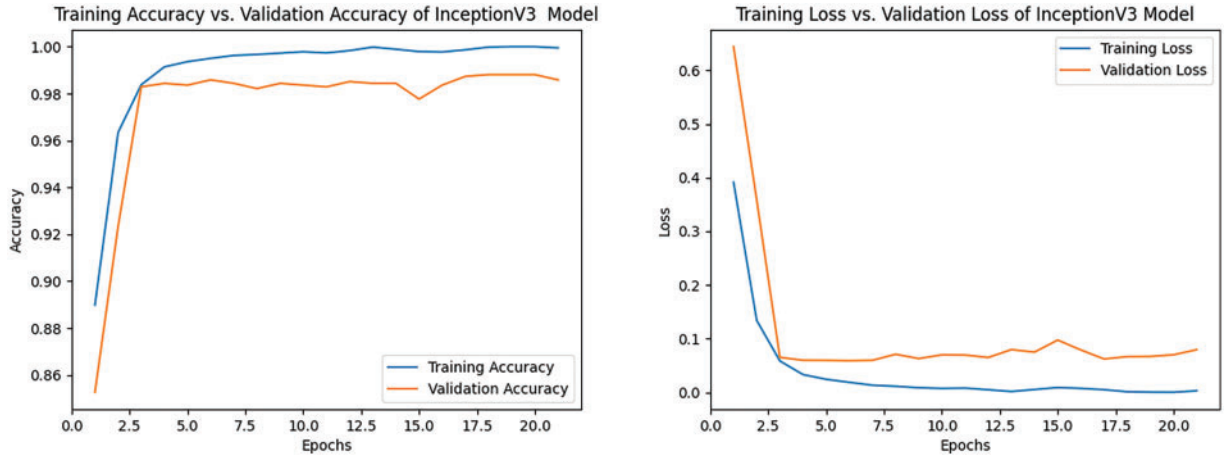
(a) (b)

Figure 10: (a) Training and validation accuracy and (b) training and validation loss of DenseNet121

The reported training accuracy of 99.96% shows that the InceptionV3 model in the Arabic dataset achieved flawless accuracy on the training dataset, indicating that it successfully learned and memorized the training samples. On the other hand, the validation accuracy of 98.60% represents the model's performance on a separate dataset not used during training, demonstrating a slightly lower but still highly accurate generalization ability. The fact that the model's accuracy on the validation set is close to the training accuracy indicates that the model did not overfit the training data. The reported test accuracy of 98.24% provides an additional measure affirming its capability to generalize well beyond the training and validation sets. The training loss is notably low at 0.003%, indicating that the model performs well on the training data and effectively minimizes errors during the learning process. On the other hand, the validation loss is slightly higher at 0.080%, suggesting a slight increase in errors when the model is tested on unseen data not used during training. While the training loss reflects the model's ability to fit the training data, the higher validation loss indicates the need for careful consideration to prevent overfitting and ensure the model's generalization to new, unseen data. The low training loss is promising, but close monitoring and potential adjustments may be necessary to enhance the model's performance on validation data and real-world applications. The performance of the InceptionV3 model is revealed in Fig. 11.

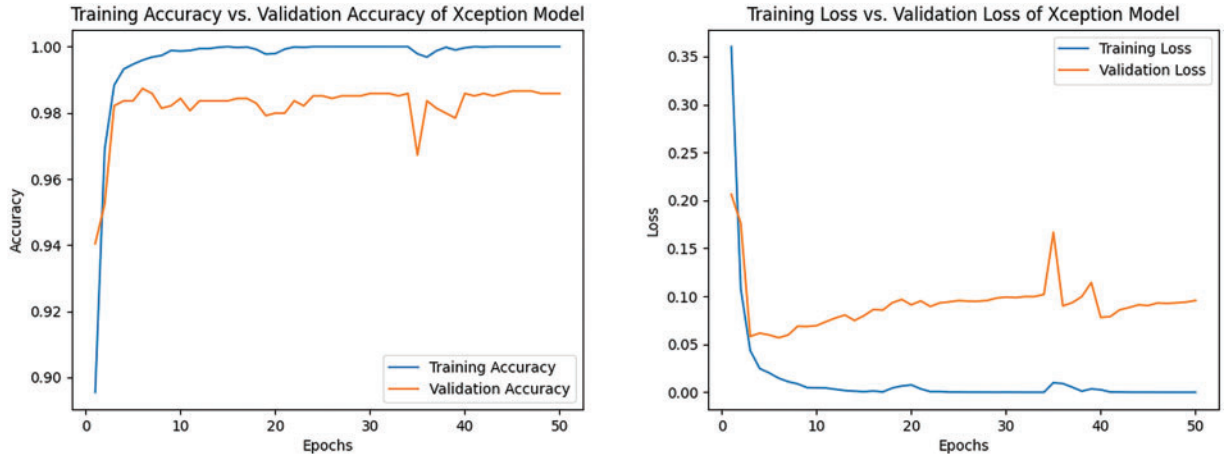
The training accuracy of an Xception model on the Arabic dataset is exceptionally high at 1.000%, indicating that the model perfectly predicts the training data it was exposed to. However, the validation accuracy, assessed on a separate set of data not used during training, is slightly lower at 98.59%. This suggests that the model generalizes well to unseen data but may not perform with absolute perfection outside of the training set. The test accuracy, which is a crucial metric for evaluating a model's performance on entirely new and unseen data, is reported at 98.66%. The discrepancy between the training and validation accuracies indicates that while the model has learned the training data thoroughly, there may be a slight drop in performance on new, unseen data. The training loss of the Xception model on the same dataset is exceptionally low, standing at 0.007%. This indicates that during the training phase, the model effectively minimized the difference between its predicted outputs and the actual targets in the training dataset. On the other hand, the validation loss, measured at 0.095%, reflects the model's performance on a separate dataset not used during training. While the training loss signifies how well the model learned from the training data, the higher validation loss suggests

that the model might not generalize as well to unseen data. The relatively low training loss is positive, but the larger validation loss warrants attention, indicating a potential need for further optimization to enhance the model's generalization capabilities beyond the training set. Fig. 12 illustrates the results of the Xception model.



(a) (b)

Figure 11: (a) Training and validation accuracy and (b) training and validation loss of InceptionV3



(a) (b)

Figure 12: (a) Training and validation accuracy and (b) training and validation loss of the Xception model

In summary, the evaluation of custom CNN, DenseNet121, InceptionV3, and Xception models on the Arabic dataset reveals their impressive performance. The custom CNN model demonstrates high accuracy in training, validation, and test sets, indicating effective learning and robust generalization to new, unseen data. However, a slightly higher validation loss suggests potential challenges in generalization, prompting the need for further investigation and model adjustments. The DenseNet121 model showcases near-perfect accuracy with low training and validation losses, reflecting a well-balanced model capable of effective learning and generalization. Similarly, the InceptionV3 model

exhibits high accuracy across datasets, indicating robustness in classifying new instances, though careful monitoring is advised for potential overfitting. The Xception model demonstrates exceptional training accuracy, with a slight drop in validation accuracy suggesting a need for optimization. Table 3 provides the results of Arabic handwritten dataset models.

Table 3: Models' results of training and validation accuracy of Arabic handwritten dataset

Model	Train accuracy	Valid accuracy	Test accuracy
Custom CNN Model	94.30%	97.32%	98.43%
DenseNet121	1.000%	98.84%	98.36%
InceptionV3	99.96	98.60	98.24%
Xception	1.000	98.59	98.66%

While the low training loss is positive, the larger validation loss indicates potential challenges in generalization, warranting further optimization efforts. The reason for using custom CNN is to have a feedforward model to check the result of a feedforward model that has the skip connection like the other models such as DenseNet121, InceptionV3, and Xception. Table 4 provides the results of Arabic handwritten dataset models for validation loss.

Table 4: The training and validation loss results of Arabic handwritten dataset

Model	Train loss	Valid loss
Custom CNN Model	0.1755%	0.0863
DenseNet121	0.0008	0.0639
InceptionV3	0.0030	0.0800
Xception	0.0071	0.0957

4.3 The Performance of Proposed Models in Recognizing Arabic Handwriting Works

The evaluation of various models for Arabic handwritten character recognition, as presented in the provided table, underscores the effectiveness of the proposed models in comparison to existing approaches. Notably, the proposed custom CNN, DenseNet121, InceptionV3, and Xception consistently outperform many established models across different datasets, particularly excelling on the AHCD dataset. ResNet-34 [17] stands out as a top performer with an exceptional accuracy of 99.60% on specific datasets. The proposed models demonstrate competitive or superior accuracy percentages, ranging from 98.24% to 98.66% on AHCD, highlighting their efficacy in handling the nuances of Arabic handwritten characters. In contrast, existing models such as CNN [14], Hybrid CNN [15], AlexNet CNN [16], CNN [18], CNN [19], CNN and MLP [20], CNN [21], and CNN [22] achieve competitive but slightly lower accuracies, while CNN [19] notably outperforms on both AHCD and MadBase datasets. This comprehensive evaluation emphasizes the promising capabilities of the proposed models for practical applications in Arabic handwritten character recognition, showcasing their robust performance across diverse datasets. Table 5 illustrates the results.

Table 5: Evaluation of proposed approach and other methods on the different datasets

Methods	Dataset	Accuracy
CNN [14]	Hijja	**91%**
	AHCD	**97%**
	Hijja and AHCD	**96%**
Hybrid CNN [15]	AHCD	96.3%
AlexNet CNN [16]	IFN/ENIT	95.6%
ResNet-34 [17]	AlexU-W and IFN/ENIT	99.60%
CNN [18]	AHCD	94.9%
CNN [19]	AHCD and	96.93%
	MadBase	99.35%
CNN and MLP [20]	AHCD	95.27%
CNN [21]	AHCD	96.78%
CNN [21]	AIA9k	94.8%
	AHCD	97.6%
Proposed custom CNN	AHCD	98.43%
Proposed DenseNet121	AHCD	98.36%
Proposed InceptionV3	AHCD	98.24%
Proposed Xception	AHCD	98.66

4.4 Comparison between Arabic and Kurdish Datasets

The provided analysis highlights the performance of various models on two distinct datasets: "KurdSet" for Kurdish character recognition and the AHCD for Arabic character recognition. The evaluation metrics encompass training accuracy, validation accuracy, test accuracy, training loss, and validation loss for each model. In the case of "KurdSet," the models, including custom CNN, DenseNet121, InceptionV3, and Xception, consistently demonstrate high training accuracy (ranging from 99.86% to 1.000%) and validation accuracy (ranging from 98.07% to 99.82%), indicative of their effectiveness in learning and generalization. The test accuracy values, ranging from 98.04% to 99.80%, underscore the models' proficiency in handling new, unseen data. Similarly, on the AHCD dataset, the models exhibit high training accuracy (ranging from 94.30% to 1.000%) and validation accuracy (ranging from 97.32% to 98.84%), with test accuracy values ranging from 98.24% to 98.66%. The low values of training loss and validation loss for both datasets signify effective error minimization during training and validation. These comprehensive results showcase the robust capabilities of the proposed models in accurately recognizing characters in both Kurdish and Arabic datasets, underscoring their adaptability and performance across different language contexts.

The analysis of KurdSet reveals noteworthy results concerning noise in the dataset, which refers to irrelevant or erroneous data that could potentially disrupt the learning process, appears to be effectively managed, as evidenced by the high training and validation accuracies and overfitting. Overfitting, which is a phenomenon where the model becomes too tailored to the training data and struggles to generalize to new data, is seemingly well-mitigated, given the consistent high-test accuracies. The models showcase strong generalization capabilities, suggesting a robust resistance to noise. The models' ability to perform well on unseen data from the test set indicates a balanced learning process and

effective generalization, reflecting the efficacy of the models in adapting to the intricacies of the Kurdish handwriting dataset.

5 Conclusion and Future Work

In this comprehensive study, we conducted an extensive evaluation of various models applied to the intricate tasks of Arabic and Kurdish handwritten character recognition, employing the AHCD and the distinct KurdSet dataset, respectively. The proposed models, including custom CNN, DenseNet121, InceptionV3, and Xception, consistently exhibit exceptional performance across key metrics, including training accuracy, validation accuracy, and test accuracy. Notably, the models achieve impressively high accuracy percentages on both datasets, indicating their effectiveness in capturing and generalizing intricate patterns inherent in Arabic and Kurdish characters. The evaluation of KurdSet unveils notable results concerning noise and overfitting. The models showcase a remarkable ability to manage noise, as evidenced by their high training and validation accuracies. This robust generalization suggests a resilient resistance to irrelevant or erroneous data, a crucial characteristic for real-world applications. Furthermore, the well-mitigated overfitting, as demonstrated by consistent high-test accuracies, underscores the balanced learning process of the models. Their ability to perform exceptionally well on unseen data from the test set affirms their capability to effectively adapt to the intricacies of Kurdish handwriting, positioning them as powerful tools in character recognition tasks within this linguistic context. Additionally, the comparative analysis with existing models on the AHCD dataset highlights the superiority of our proposed models. The models consistently outperform established approaches, emphasizing their efficacy in handling the nuanced intricacies of Arabic handwritten characters.

Despite the promising results, our proposed method has certain limitations such as accuracy and participants that warrant consideration in future research. For future work, we plan to make our system better and more accurate, and we are planning to include more participants in our study with different handwriting styles. This way, our system can understand and recognize a wider range of writing. We are also working on improving accuracy by refining our model and trying out new training methods. It may involve expanding datasets for Arabic and Kurdish character recognition, exploring domain adaptation for diverse handwriting styles, and developing multilingual models.

Acknowledgement: We express our gratitude to the University of Zakho for their support, which was instrumental in the successful completion of our paper. Special thanks are extended to the Computer Science Department for their valuable assistance throughout the research process.

Funding Statement: No specific grant for this research was obtained from any public, commercial, or not-for-profit funding agency.

Author Contributions: Sardar Hasen Ali formulated the research plan, coordinated the study, organized data analysis and interpretation of results. Maiwan Bahjat Abdulrazzaq participated in editing and contributed to manuscript writing. Both authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The training data used in this paper were obtained from available online via the following link: https://www.kaggle.com/datasets/rashwan/arabic-chars-mnist.

Conflicts of Interest: The authors affirm that they have no conflicts of interest to declare in relation to the current study.

References

[1] B. Rajyagor and D. R. M. Rakholia, "Handwritten character recognition using deep learning," *Int. J. Recent Technol. Eng. (IJRTE)*, vol. 8, no. 6, pp. 5815–5819, 2020. doi: 10.35940/ijrte.F8608.038620.

[2] D. E. Reddy, K. V. Pranathi, M. K. Srinivas, A. Raheem, and S. Sureddy, "Handwritten character recognition using SVM," *Int. J. Adv. Sci. Technol.*, vol. 29, no. 5, pp. 4001–4007, 2020. doi: 10.55014/pij.v3i2.98.

[3] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with MNIST and EMNIST," *Appl. Sci.*, vol. 9, no. 15, pp. 3169, 2019. doi: 10.3390/app9153169.

[4] S. M. Shamim, M. B. A. Miah, A. Sarker, M. Rana, and A. Al Jobair, "Handwritten digit recognition using machine learning algorithms," *Indones. J. Sci. Technol.*, vol. 3, no. 1, pp. 29–39, 2018. doi: 10.17509/ijost.v3i1.10795.

[5] M. Alayyoub, A. Nuseir, K. Alsmearat, Y. Jararweh, and B. Gupta, "Deep learning for Arabic NLP: A survey," *J. Comput. Sci.*, vol. 26, no. 2, pp. 522–531, 2018. doi: 10.1016/j.jocs.2017.11.011.

[6] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. doi: 10.1038/nature14539.

[7] K. G. Pasi and S. R. Naik, "Effect of parameter variations on accuracy of Convolutional Neural Network," in *Int. Conf. Comput. Anal. Secur. Trends* (*CAST*), 2016, vol. 15, pp. 398–403. doi: 10.1109/CAST.2016.7915002.

[8] M. B. Bora, D. Daimary, K. Amitab, and D. Kandar, "Handwritten character recognition from images using CNN-ECOC," *Procedia Comput. Sci.*, vol. 167, pp. 2403–2409, 2019. doi: 10.1016/j.procs.2020.03.293.

[9] N. Altwaijry and I. Turaiki, "Arabic handwriting recognition system using convolutional neural network," *Neural Comput. Appl.*, vol. 33, no. 7, pp. 2249–2261, 2021. doi: 10.1007/s00521-020-05070-8.

[10] K. Younis and A. Khateeb, "Arabic hand-written character recognition based on deep convolutional neural networks," *Jordanian J. Comput. Inf. Technol.*, vol. 3, no. 3, pp. 186–200, 2017. doi: 10.5455/jjcit.71-1498142206.

[11] H. M. Balaha *et al.*, "Recognizing Arabic handwritten characters using deep learning and genetic algorithms," *Multimed. Tools Appl.*, vol. 80, no. 21–23, pp. 32473–32509, 2021. doi: 10.1007/s11042-021-11185-4.

[12] A. T. Taani and S. AlHaj, "Recognition of on-line Arabic handwritten characters using structural features," *J. Pattern Recognit. Res.*, vol. 5, no. 1, pp. 23–37, 2010. doi: 10.13176/11.217.

[13] A. Maytham, K. Raidah, and J. Sardar, "Improved Arabic characters recognition by combining multiple machine learning classifiers," in *The 20th Int. Conf. Asian Language Process.* (IALP), Tainan, Taiwan, IEEE, 2016, pp. 262–265.

[14] M. Alheraki, R. Al-Matham, and H. Al-Khalifa, "Handwritten Arabic character recognition for children writing using convolutional neural network and stroke identification," *Human-Centric Intell. Syst.*, vol. 3, no. 2, pp. 147–159, 2023. doi: 10.1007/s44230-023-00024-4.

[15] N. Alrobah and S. Albahli, "Arabic handwritten recognition using deep learning: A survey," *Arab. J. Sci. Eng.*, vol. 47, no. 8, pp. 9943–9963, 2022. doi: 10.1007/s13369-021-06363-3.

[16] 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. doi: 10.1109/ACCESS.2020.2994290.

[17] M. Awni, M. I. Khalil, and H. M. Abbas, "Offline Arabic handwritten word recognition: A transfer learning approach," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9654–9661, 2022. doi: 10.1016/j.jksuci.2021.11.018.

[18] M. N. Aljarrah, M. M. Zyout, and R. Duwairi, "Arabic handwritten characters recognition using convolutional neural network," in *2021 12th Int. Conf. Inf. Commun. Syst. ICICS*, 2021, vol. 8, pp. 182–188. doi: 10.1109/ICICS52457.2021.9464596.

[19] M. Kamal, F. Shaiara, C. M. Abdullah, S. Ahmed, T. Ahmed and M. H. Kabir, "Huruf: An application for Arabic handwritten character recognition using deep learning," arXiv preprint arXiv:2212.08610, 2022.

[20] O. A. Almansari and N. N. W. N. Hashim, "Recognition of isolated handwritten Arabic characters," in *2019 7th Int. Conf. Mechatronics Eng.* (*ICOM*) , 2019, vol. 5, pp. 1–5. doi: 10.1109/ICOM47790.2019.8952035.

[21] Z. Ullah and M. Jamjoom, "An intelligent approach for Arabic handwritten letter recognition using convolutional neural network," *PeerJ Comput. Sci.*, vol. 8, pp. e995, 2022. doi: 10.7717/peerj-cs.995.

[22] A. Alzebdeh, M. M. Khaled, M. Lataifeh, and A. Elnagar, "Arabic handwritten recognition based on deep convolutional neural network," in *IET Conf. Proc.*, 2021, vol. 2021, pp. 271–287. doi: 10.1049/icp.2021.2682.

[23] M. Sharma, P. S. Sindal, and M. Baskar, "Handwritten digit recognition using machine learning," *Lect. Notes Netw. Syst.*, vol. 572, pp. 31–43, 2023. doi: 10.1007/978-981-19-7615-5.

[24] H. Jain and N. Sharma, "Handwritten digit recognition using CNN," *Lect. Notes Netw. Syst.*, vol. 166, pp. 631–637, 2021. doi: 10.1007/978-981-15-9689-6.

[25] D. J. S. Naidu and T. M. Rafi, "Handwritten character recognition using convolutional neural networks," *Int. J. Comput. Sci. Mobile Comput.*, vol. 10, no. 8, pp. 41–45, 2021. doi: 10.47760/ijcsmc.2021.v10i08.007.

[26] F. Haghighi and H. Omranpour, "Stacking ensemble model of deep learning and its application to Persian/Arabic handwritten digits recognition," *Knowl.-Based Syst.*, vol. 220, no. 14, pp. 106940, 2021. doi: 10.1016/j.knosys.2021.106940.

[27] T. Ghosh, M. H. Z. Abedin, H. Al Banna, N. Mumenin, and M. Abu Yousuf, "Performance analysis of state of the art convolutional neural network architectures in Bangla handwritten character recognition," *Pattern Recognit. Image Anal.*, vol. 31, no. 1, pp. 60–71, 2021. doi: 10.1134/S1054661821010089.

[28] N. Saqib, K. F. Haque, V. P. Yanambaka, and A. Abdelgawad, "Convolutional-neural-network-based handwritten character recognition: An approach with massive multisource data," *Algorithms*, vol. 15, no. 4, pp. 129, 2022. doi: 10.3390/a15040129.

[29] M. A. Uddin, "Handwritten Bangla character recognition using artificial neural network," *IOSR J. Comput. Eng.*, vol. 16, no. 3, pp. 33–38, 2014. doi: 10.9790/0661-16333338.

[30] S. Nasi and H. N. Sharada, "Offline handwritten mathematical expression recognition using CNN and Xception," *Int. J. Sci.Res.Comput. Sci. Eng. Inf. Technol.*, vol. 26, no. 9, pp. 142–149, 2022. doi: 10.32628/cseit228420.