modelling

MDPI

Article

Manuscripts Character Recognition Using Machine Learning and Deep Learning

Mohammad Anwarul Islam † and Ionut E. Iacob *,†

Mathematical Sciences, Georgia Southern University, Statesboro, GA 30458, USA
* Correspondence: ieiacob@georgiasouthern.edu; Tel.: +1-912-478-0164
† These authors contributed equally to this work.

Abstract: The automatic character recognition of historic documents gained more attention from scholars recently, due to the big improvements in computer vision, image processing, and digitization. While Neural Networks, the current state-of-the-art models used for image recognition, are very performant, they typically suffer from using large amounts of training data. In our study we manually built our own relatively small dataset of 404 characters by cropping letter images from a popular historic manuscript, the Electronic Beowulf. To compensate for the small dataset we use ImageDataGenerator, a Python library was used to augment our Beowulf manuscript's dataset. The training dataset was augmented once, twice, and thrice, which we call resampling 1, resampling 2, and resampling 3, respectively. To classify the manuscript's character images efficiently, we developed a customized Convolutional Neural Network (CNN) model. We conducted a comparative analysis of the results achieved by our proposed model with other machine learning (ML) models such as support vector machine (SVM), K-nearest neighbor (KNN), decision tree (DT), random forest (RF), and XGBoost. We used pretrained models such as VGG16, MobileNet, and ResNet50 to extract features from character images. We then trained and tested the above ML models and recorded the results. Moreover, we validated our proposed CNN model against the well-established MNIST dataset. Our proposed CNN model achieves very good recognition accuracies of 88.67%, 90.91%, and 98.86% in the cases of resampling 1, resampling 2, and resampling 3, respectively, for the Beowulf manuscript's data. Additionally, our CNN model achieves the benchmark recognition accuracy of 99.03% for the MNIST dataset.

Keywords: character recognition; computer vision; convolutional neural network; deep learning; machine learning; old english

Citation: Islam, M.A.; Iacob, I.E. Manuscripts Character Recognition Using Machine Learning and Deep Learning. *Modelling* **2023**, *4*, 168–188. https://doi.org/10.3390/modelling4020010

Academic Editor: Miquel Sànchez-Marrè

Received: 28 February 2023
Revised: 27 March 2023
Accepted: 2 April 2023
Published: 4 April 2023

1. Introduction

In the present era, the importance of automatic handwritten character recognition is ubiquitous. With the advancement of artificial intelligence and modern technology, automatic handwritten character recognition has been used in many different spheres of our everyday life such as reading postal addresses, reading bank checks and amounts, automated teller machines, detecting traffic signals, automatic license plate recognition for parking systems and traffic police personnel, customs and immigration security checking, digital libraries, automated language translation and keyword spotting, etc. [1–3]. In addition, the automatic handwritten character recognition serves many historic, academic, commercial, and national interests. In this article, we used the historic Old English Beowulf's manuscript [4] images to perform character recognition. This manuscript is important for two reasons: firstly, it is a historic epic poem of literal value, and secondly, it was written in Old English, which has linguistic value because it survived in a single medieval manuscript [5]. As the manuscript bears no date, based on the analysis of the scribes' handwriting, scholars conjectured that Beowulf has been written in the early eleventh century, which makes it approximately 1000 years old [4,6,7]. This manuscript had

been preserved by several institutions such as the British Library [5], Electronic Beowulf CD-ROM [4], and the Gutenberg eBook of Beowulf [7] for its academic, research, linguistic, poetic, and ancient values. To our best knowledge, no significant study has been performed on automatic handwritten character recognition for these historic manuscripts. Recently, preliminary works on the Electronic Beowulf manuscript tried to use Artificial Neural Network [8] and Convolutional Neural Network [9] models to perform character recognition.

To facilitate the availability of the ancient manuscript to the public, digitization can be considered as a vital step to allow access to its content for various types of research purposes [10]. Allowing open access to digitized manuscripts can encourage many types of relevant interdisciplinary research. This study is important in the field of automatic handwritten character recognition of the ancient documents and it is a novel work on the Beowulf's Old English character recognition. However, there is some significant prior work on character recognition for manuscripts in other languages. For example, Suryani et al. [11] and Hidayat et al. [12] conducted their study on Sundanese palm leaf manuscript dataset. Sutramiani et al. [13,14] investigated Balinese character recognition using Convolutional Neural Networks. Several studies [2,15–20] were conducted on handwritten Bangla character recognition using Convolutional Neural Networks. A considerable number of studies conducted on Manipuri meetei-myek [21], Tamil palm leaf manuscript [22], Malayalam [23], Arabic [24], Hindi [25], and Javanese [26] handwritten character recognition using Convolutional Neural Networks and other machine learning approaches. To the best knowledge of the authors, there was no such study conducted to recognize character for the poetically, linguistically, and historically important Beowulf's Old English manuscript.

However, the major obstacle to the automatic handwritten character recognition of the historic Beowulf's Old English manuscript is the dataset's quality and physical condition. The manuscript's large images can be found on several open sources online platforms, but they lack clarity due to the manuscript's physical condition. The majority of the images are fainted, faded, stained, and distorted, some parts are missing, external spots are on it, and some of the characters and lines are omitted due to aging, as shown in Figure 1. The cursive nature of the manuscript characters is an impediment acknowledged by many researchers [2,27,28] performing character recognition tasks, whereas manuscripts with isolated characters (e.g., Chinese, Japanese, Thai, Korean, Laos) provide advantages over that of former kinds.

We manually created our dataset from the Bewoulf manuscripts' images, as the ones shown in Figure 1. To the best knowledge of the authors there is no dataset of the Beowulf Old English handwritten characters for character recognition models, like the MNIST [29] dataset or other readily available open-source datasets. We used image-cropping techniques, which are simple to execute but very time-consuming, to get an individual image for each letter of the Beowulf Old English manuscript. A considerable number of the characters of the Beowulf Old English manuscript are cursive and some of them are separated from each other. It can be observed that cropping is very difficult in the case of the cursive letters, which are not very well separated from each other. In some cases, due to the dilapidated condition of the manuscript images, we faced challenges to cropping letters properly. Despite these difficulties, we developed a dataset to train and test our proposed model.

We subsequently used the ImageDataGenerator data augmentation [12,13,19] technique, a python library to generate new data, which increased the size of our dataset for better training our proposed CNN model. The main contributions of this study are as follows:

- We have built our own dataset from Beowulf's large electronic images by cropping each character manually.
- We have conducted a comparative study about the performance of different Machine Learning models on different sizes of the dataset against our proposed model.

- We assembled a CNN model, which performs very well on manuscript character images recognition (compared to other well-established classifiers), and achieved benchmark accuracy on the MNIST dataset.

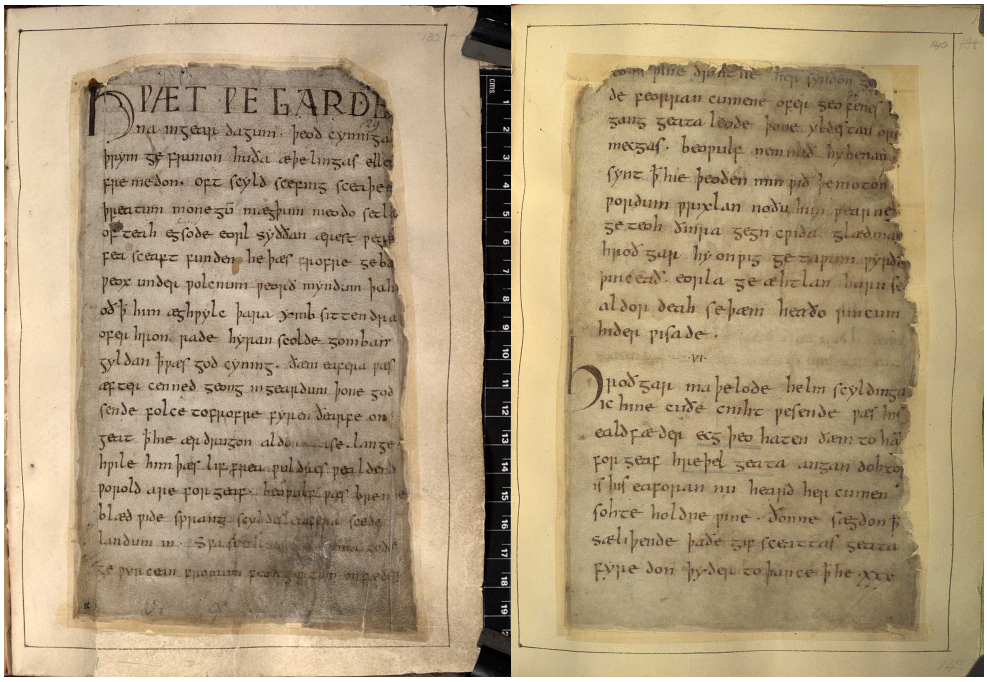


Figure 1. Beowulf's Old English Manuscript images.

In summary, the novelty of this work stems from the kind of problem we have chosen to solve: Old English manuscript characters recognition. We have used standard tools, which we configured as appropriate for our purposes. However, no standard character recognition tool performs satisfactorily on our dataset due to factors such as: no fully pre-trained models on the language being studied, noisy and/or blurry images (spots, damages, etc.), and limited size dataset (which makes training very difficult). Moreover, there is neither a large size dataset available, nor is such a large dataset easy to create (one would classify a whole manuscript in an attempt to harvest a large number of characters for training the model). A typical neural network model for image recognition requires large training dataset to perform satisfactorily. As we could not satisfy this requirement, we relied on other techniques to make our model perform at the same level as other models trained for similar tasks.

The rest of the paper is organized as follows. Section 2 reviews related work on ancient manuscripts and handwritten characters recognition using different Machine Learning and Deep Learning models. A discussion on the Beowulf manuscript character images data preparation and MNIST digits data is given in Section 3. In Section 4 we discuss the models we use and how we devise our methods to perform manuscript characters recognition. Numerical results and an analysis of this study are provided in Section 5. Finally, this article concludes and indicates future research directions in Section 6.

2. Related Work

In recent years, handwritten character recognition has become increasingly important in many sectors of our daily life, such as in reading zip codes, postal addresses, digital libraries, language transformation, automated news reading, keyword spotting, check amount and bank form reading, traffic signs detection, etc. For decades, many studies have been conducted on the character recognition of historic documents, and various techniques have been used. Alam Hidayat et al. [12] conducted a study to recognize Sundanese characters using a Convolutional Neural Network (CNN) model. To leverage a small dataset, the

study used data augmentation to train their CNN model and achieved a test accuracy of 97.74%. A number of other studies [12–14,16,30–35] used the data augmentation technique to overcome the data deficiency problem and to get commendable accuracy in the task of character recognition or classification. Another study [13] used a Multi-Augmentation Technique and Adaptive Gaussian Convolutional Autoencoder (MAT-AGCA) to recognize *lontar* manuscript characters, the ancient Balinese letter written on palm leaves. This study compared several deep learning models such as InceptionResnetV2, DenseNet169, ResNet152V2, VGG19, and MobileNetV2 with their proposed model MAT-AGCA, and showed that their model outperformed them with a recognition accuracy of 96.29%. Nogra, James et al. [36] used CNN to recognize characters from an ancient Bybayin script. In their CNN model they used three convolutional layers with 32 channels, 64 channels, and 128 channels, respectively, using 3×3 filters. The model has three maxpooling layers after each convolution layer with a 2×2 kernel and two fully connected layers at the end, producing a test accuracy of 94%. A great deal of studies [1,2,21–23,28,31–33,37–40] confirmed that the CNN model outperformed in character recognition tasks compared to other existing deep learning models. In particular, Sakib, Nazmus et al. [1] used the CNN model to perform handwritten character recognition on two open source datasets (Kagle and MNIST) achieving higher accuracy than using other performant models. In their study, the proposed CNN model consists of four 2D convolutional layers, the first convolutional layer with 3×3 kernel followed by a ReLu activation function, each of the subsequent three convolutional layers followed by a maxpooling layer with 2×2 strides, and a ReLu activation function, which is followed by two fully connected layers. It achieved a 99.29% accuracy while using a 'RMSprop' optimizer, and using the 'ADAM' optimizer a 99.64% accuracy was achieved. A survey study [30] explored the efficiency of the various deep learning techniques in handwritten character recognition such as different types of CNN architecture, KNN, SVM, MetaQNN Feedforward network, SparseNet+SVM, Recurrent CNN, and found that CNN performed better with a high accuracy of 99.76%. Another study [14] proposed a deep CNN model to train Balinese *lontar* manuscript data and conducted several experiments using different types of optimizers and learning rates. They reported a high training and testing accuracy while using stochastic gradient descent with training accuracy 99.36%, validation accuracy 87.15%, and testing accuracy 86.23%. Liu, Xin, et al. [41] proposed stroke sequence-dependent deep CNN, which captures eight-directional features and stroke sequence information for Chinese handwritten character recognition. Their model performed better than other performance machine learning methods with a recognition accuracy of 97.94%, which reduced the recognition error by approximately 18% as compared with the ICDAR 2013(HDR 2013) [42]. A study [43] proposed a deep convolutional neural network for Arabic Scene character recognition using a manually segmented Arabic Scene dataset consisting of a total of 2700 character images, of which 2450 images were used to train and 250 images were used to test their ConvNets model. They had the highest test accuracy of 99.85% while using a 3×3 filter or kernel and 0.005 learning rate among the experiments of other combination of filters and learning rates. Manocha, Shilpa et al. [44] used the DHCD dataset to train and test a number of machine learning models such as SVM, KNN, RF, DT, MLP, XGB, and CNN, among which they reported that the CNN model performed better than all of the above models with an accuracy rate of 92%. This study uses a CNN model with four convolutional layers, and after each pair of convolutional layers they used a maxpooling layer followed by a dropout layer, a flattened layer, and three fully connected dense layers. A survey study [25] on handwritten Hindi characters recognition emphasized techniques like preprocessing and feature extraction during the training of the model. Tamir and Kassahum [45] conducted a study for Ethiopian Amharic characters recognition using a CNN model consisting of only two convolutional layers, and each convolutional layer was followed by a batch normalization and activation function. The last part of the architecture has a maxpooling and a fully-connected layer. The study used a dataset consisting of 30,446 images of 286 different characters, among which 27,413 images were for training the model and 3033 were for testing the model to get

286 classes, and reported training and testing accuracy of 99.52% and 99.71%, respectively. A study [26] explored the Javanese character recognition by developing two CNN models, the first model consists of three 2D convolutional layers, three pooling layers, and a fully connected layer. The second model was built by adding one mode fully connected layer at the end just before the classifier. They tuned up several parameters of the model like an optimizer, learning rate, regularization, and activation function, and reported 94.57% as the highest recognition accuracy rate among the two models. Moreover, their investigation found that the CNN model outperformed in the case of character recognition. A master thesis from Georgia Southern University [9] also investigated and concluded that the CNN model outperformed in the case of image recognition. Hazra and Abhishek et al. [15] conducted a research study for Bangla handwritten character recognition and validated their CNN model against two datasets, cMETERdb and ISI. The architecture of their model consists of three convolutional layers each followed by a pooling layer, and three fully connected layers followed by a softmax classifier at the end of the architecture. Their proposed model consistently performed well for both datasets in comparison to other methods. A study [24] used CNN to recognize historic Arabic characters and reported the recognition accuracy between 74.29–88.20%. Researchers in [46–48] used CNN and deep CNN models to recognize handwritten characters and reported recognition accuracy of 93.73% and 98%, while research in [49] used an optimized CNN model to recognize Punjabi script images and achieved a 92.05% recognition accuracy. Nouf and Alrasheed et al. [50] explored the recognition task of seventeenth-century Spanish American Notary records using different types of deep CNN models. They trained and tested all the models on a dataset of 220,000 images. They achieved an accuracy rate of 97.08%, 96.66%, 96.33%, and 70.91%, respectively, from the ResNet-50, InceptionResnet-V2, Inception-V3, and VGG-16.

After a careful review of the existing work on the manuscript or handwritten character recognition, the authors of this research found no mentionable work of character recognition of Old English manuscripts using either any machine learning approaches or deep convolutional neural network. For this reason, we consider that our work is filling a gap and it is equally important for preserving and studying valuable historic manuscripts using deep Convolutional Neural Networks for character recognition.

3. Dataset

We performed our experiments on two datasets: Old English characters retrieved from the Beowulf manuscript and the open-source MNIST handwritten digits dataset. These two datasets are described in the rest of this section.

3.1. The Beowulf Manuscript Dataset

The Beowulf manuscript is an epic poem written in the Old English alphabet by an unknown author between 975 and 1025. In the Old English literature, Beowulf is considered to be the highest achievement of the Anglo-Saxon period. Because of its ancient nature, Beowulf's manuscript is an important historic document for literature, history, science, and engineering. Although the printed copy of the manuscript is not available to the general public (it resides in the British Library's special collections), the Beowulf manuscript's electronic images are available online as the Electronic Beowulf [4], currently in its fourth edition. This is a free and open-source online version for all levels of enthusiasts, students, and researchers interested in medieval English. The online edition contains 70 daylight folio images, 130 of ultraviolet images, and 750 of high-resolution images, as the samples shown in Figure 1.

To the best knowledge of the authors, there is no Old English characters dataset created from Beowulf's manuscript electronic images (or similar Old English Manuscripts) for machine learning researchers and enthusiasts, which is a big impediment to the automatic character recognition of this ancient manuscript. For the purpose of this study we created our own dataset from the manuscript's electronic images by cropping each letter manually.

The dataset is available at: https://bit.ly/OECharacterSet (accessed on 28 February 2023). Our dataset consists of 22 Old English alphabet letters, with 20 sample images from each letter. Due to the curly nature of the manuscript's handwritten characters, it was very difficult to clearly crop each of the Old English letter sample. Moreover, it was also difficult to create a large dataset because of the poor condition of the manuscript. Consequently, our Beowulf manuscript dataset consists of a total of 440 image data of 22 classes of the 22 Old English characters, as in Figure 2.

$$a, æ, b, c, d, e, ð, f, g, h, i, l, m, n, o, p, r, s, t, þ, u, w$$

Figure 2 shows the corresponding images of the above Old English alphabets from our newly created dataset.

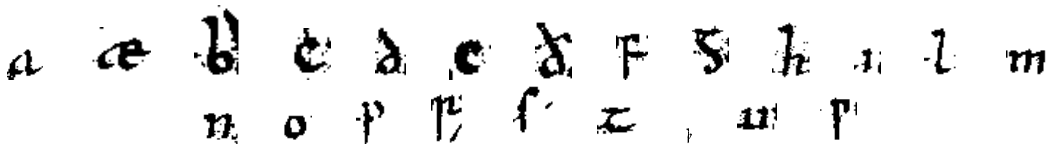


Figure 2. Sample images of characters used in experiments.

For the dataset we created we performed two-dimensional features extraction using t-SNE, and the results are presented in Figure 3. The figure shows that the datapoints corresponding to the same character class are not forming compact clusters, and hence it is likely difficult to produce a good classification.

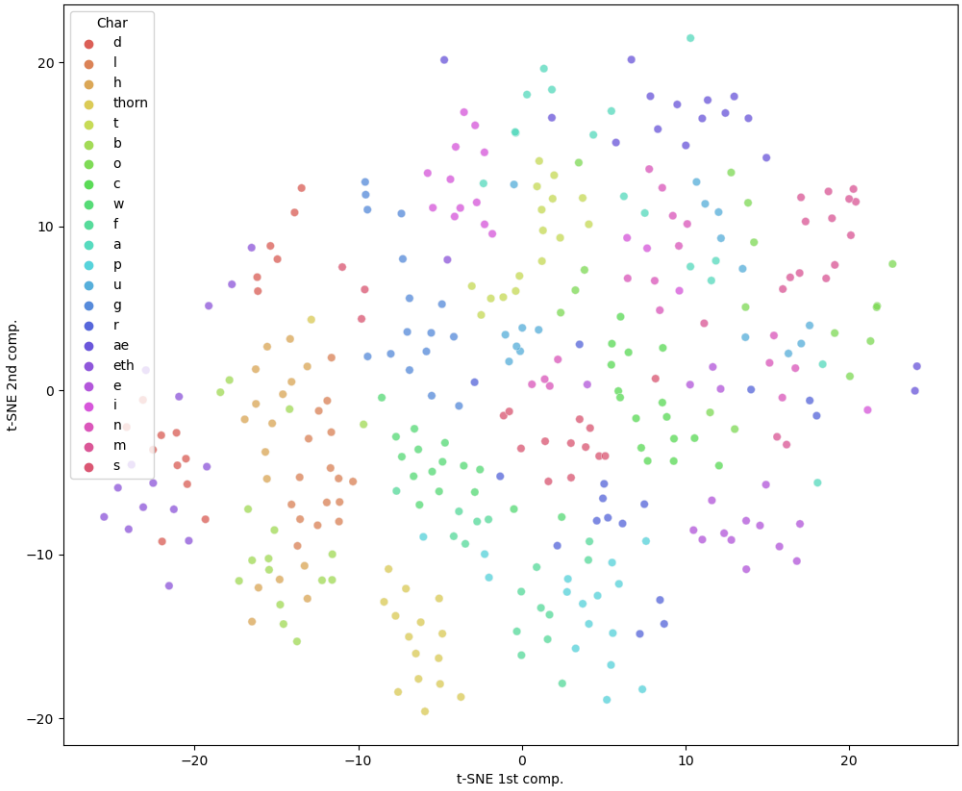


Figure 3. Two-dimensional tSNE representation of the dataset character images (for all 22 classes/image characters).

3.2. The MNIST Dataset

The MNIST handwritten dataset is a very large, publicly available dataset of handwritten digits. The acronym MNIST is used for the Modified National Institute of Standards and Technology. It is the largest handwritten dataset for various types of image processing [29]. This

dataset was created by LeCun et al. [29] in 1998. It consists of 10 digits from 0 to 9, each with 28 × 28 pixels of grayscale image data. The MNIST dataset contains 60,000 training images and 10,000 test images. The training dataset was collected from the employees of the Census Bureau of America, while the testing dataset was collected from the high school students in the USA. This dataset is the most widely used dataset among machine learning researchers. Many machine learning models and deep convolutional models have been trained and tested by the MNIST handwritten digits dataset and researchers got almost human-level recognition accuracy [51–53], while many researchers used the MNIST dataset to validate their proposed models. In our research, we validated our proposed Convolutional Neural Network model to see how our model performs on the Beowulf manuscript dataset in comparison with the well-established MNIST handwritten digits dataset. Some examples of the MNIST dataset are given in Figure 4.

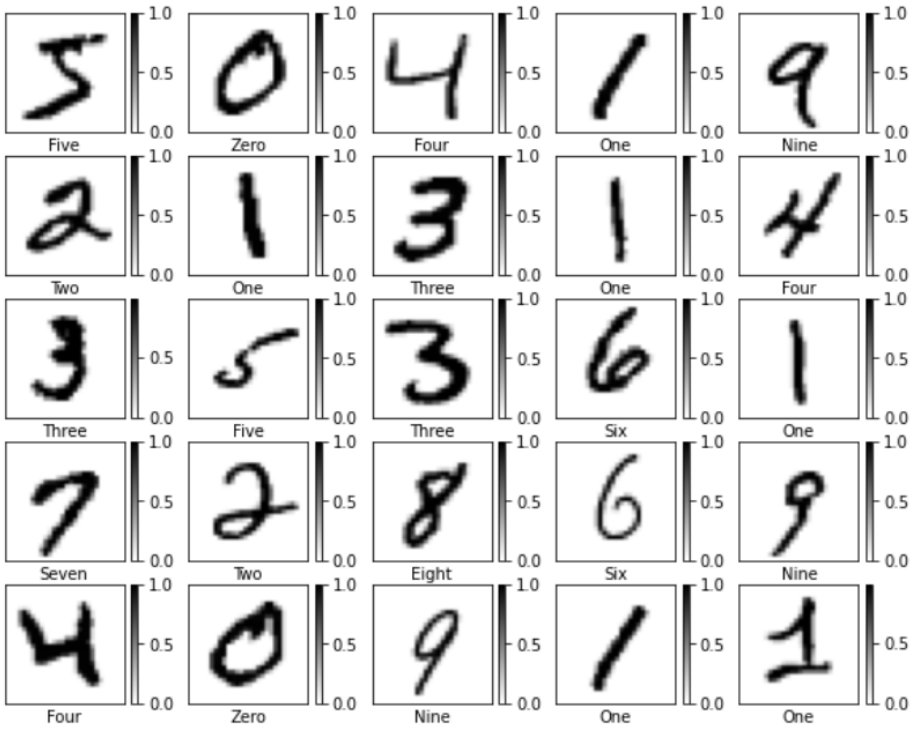


Figure 4. Some examples from MNIST dataset images.

4. Methodology

The main steps of the methodology of our study are as follows: collection of character samples, data preprocessing (including data augmentation), data splitting, data analysis, model selection, model devise and building, model training and testing, parameters tuning, and collecting the analysis results. We used the Beowulf Old English manuscript to prepare the dataset for training and testing all the ML models. From some of the large manuscript images, as shown in Figure 1, we cropped the Old English alphabet character images manually to create our own dataset. In addition to that, we used the MNIST handwritten digits dataset [29] to validate our proposed Convolutional Neural Network (CNN) model. For comparison, we also used other Machine Learning (ML) models such as Support Vector Machine (SVM), K-nearest neighbor (KNN), Decision Tree (DT), Random Forest (RF), and XGBoost. We trained and tested all models using both datasets. Finally, we performed evaluation metrics such as F-score, recall, and precision to check each model performance. In this section we discuss some shallow ML classifier models such as Support Vector Machine (SVM), K-nearest Neighbor (KNN), Decision Tree (DT), Random Forest (RF), and XGBoost that have been used for the Beowulf manuscript character recognition. Following that, we discussed our proposed CNN classifier model and subsequently explained the details of each step of the process.

4.1. Support Vector Machine

In the current work, we used Support Vector Machine (SVM) as a multi-class classifier. SVM was originally introduced in 1963 by Vapnik and Chervonenkis to classify binary attributes/variables [54] and was subsequently extended to perform multi-class classification. Kernal tricks and soft margin concepts were used effectively in SVM to separate data points that are not linearly separable. To address multi-class problems, one-vs-one or all-vs-one strategies were employed to generate a number of classifiers. SVM was used as a linear and non-linear classifier, which is more suitable for the data with high dimensional features (e.g., images) [55,56]. SVM classifiers determine the best separating hyperplane between different classes (e.g., positive vs negative) by maximizing margin distances [55,57]. The points around the hyperplane are the deciding factors in class separation.

In practice, SVM has been used more effectively to study data with higher dimensionality and to model non-linear decision boundaries [56]. One major issue when using SVM is overfitting, if the number of features is higher than the number of observations. Another disadvantage is that SVM classifiers do not provide probability estimates directly, and they could be later calculated using expensive five-fold cross-validation techniques. Additionally, SVM algorithms lack transparency in the case of a high number of dimensions [54].

4.2. K-Nearest Neighbor

K-Nearest Neighbor (KNN) is a popular non-parametric, instance-based learning text classifier (i.e. does not construct a general internal model but just stores an instance of the data) [54,56]. Based on some similarity measurement (e.g., dot product, cosine similarity), the KNN method successfully classifies texts or documents [55]. The similarity between two data points was measured by using a distance, proximity, or closeness function. Training datasets were used and classes of all incoming test nearest k neighbors were assigned by comparing their similarity with the training dataset points. A KNN classifier computes the classification based on a simple majority vote of the nearest neighbors of each data point (i.e. class is assigned based on the majority representation of the nearest points) [56,58]. The number of nearest neighbors (K) was determined by specifying the number of neighbors surrounding each point or by estimating the number of neighbors within a fixed radius of each point.

KNN classifiers are simple, fast, easy to implement, adaptable to all features space, and applicable for multi-class classification problems [54,58,59]. On the downside, KNN applicability is limited by higher time and space complexities due to a higher number of data points storage, being highly dependent on data for finding a meaningful distance function [54,55] and poor classification performance for high dimensional data.

4.3. Decision Tree

Decision Tree (DT) is one of the most popular and relatively old classification algorithms. DT was introduced in 1963 by Morgan and Sonquist from the department of statistics at the University of Wisconsin-Madison [60]. DT extracts information from a large amount of data, and depending on the value of information gained, it selects the most important feature for its parent node to have further branching. In the next steps of the algorithm, it recursively repeats the same process to select the next important features until it reaches its leaf nodes to get pure classes. A decision tree is easy to use because it can easily handle missing values and the overfitting problems. Moreover, the DT algorithm is non-parametric and it is easy to train the decision tree model. DT uses some techniques, such as the selection of variables, assessing the most important features, creating parent nodes, and then gradually creating child nodes on the basis of the importance of the features, branching whole trees to get to the leaf nodes for the final predictions of the classes. In the DT algorithm, data are divided into training and validation datasets, with the training dataset used to build the decision tree model, and the validation dataset used to achieve the optimal result by post-pruning to determine the actual size of the decision tree model.

4.4. Random Forest

The Random Forest (RF) classifier is an ensemble-based learning method for text or image pixels classification which was originally developed by T. Kam Ho in 1995 [54]. The salient idea of an RF model is to generate random decision trees to perform text or document classification. Ref. [56] mentioned that RF is a meta-estimator that develops and fits several DTs on sub-samples of datasets and uses the average to control overfitting, decrease variance, and improve the accuracy of the predictive models. They also stated that an individual tree provides higher prediction errors. However, taking the average of several random trees can cancel out some of the errors and yield a better model with low variance, despite a slight increase in bias.

RF models are very fast to train compared to other deep learning methods [54], and thus practitioners suggested reducing the size and number of trees in the forest to achieve a faster structure. However, they are quite slow to create predictions after training the dataset.

4.5. XGBoost

XGBoost is an acronym for 'Extreme Gradient Boosting'. It is a scalable machine learning technique used as a tree boosting [61]. XGBoost is a very popular and vastly used supervised learning method in the domain of data science and machine learning to achieve excellent results. In practice, XGBoost performs parallel decision tree boosting and optimizes distributed gradient boosting efficiently. Gradient-boosted decision trees, along with other models, are trained by the XGBoost method. XGBoost packages have been developed in several programming languages for easy access by machine learning enthusiasts and researchers. Moreover, it has some great advantages over other performant machine learning methods. XGBoost can provide a direct route for the minimization of the cost function, simplify the calculation and lower the computational cost, and improve execution speed by converging quickly in a few numbers of epochs. XGBoost is also very effective in handling missing data, tackling overfitting, and parallel processing.

4.6. Proposed Convolutional Neural Network (CNN) Model

The Convolutional Neural Network (CNN) model is considered a deep neural network architecture in the domain of deep learning techniques. CNN is a very powerful deep learning method for image recognition. Recently, CNN has become one of the most popular deep learning methods in image processing because of its capacity to handle huge amounts of data. The CNN architecture consists of several layers: the input layer, convolutional layers, pooling layers, non-linearity layers, fully connected layers, and output layer. CNN is also famous for its convolutional operations. The mathematical formula for the convolution operations in the CNN architecture is given by Equation (1).

$$(X * K)(i, j) = \sum_{m} \sum_{n} K(m, n) X(i - m, j - n) \quad (1)$$

In the convolution step, a learnable kernel matrix slides over the spatial features of the image matrix to collect the important features using element-by-element matrix multiplication. Most importantly, in this layer the dimension of the spatial features of the input image is reduced, and in turn, it speeds up the processing time by the CNN model [62]. Thus, the convolutional layer reduces the complexity of the model significantly by optimizing hyperparameters such as kernel size, depth, number of layers, strides, pooling, padding, etc. [63]. If the number of layers and neurons in each layer becomes very large, it suffers from a lot of computational problems and needs an unthinkably huge amount of time to train the model. To reduce the computational expense, the numbers of hyperparameters need to be tuned up. By applying a large number of strides, the spatial dimensionality of features can be reduced. Moreover, pooling, padding, and downsampling were used to reduce the dimension of the spatial features. The activation function was applied in each layer of the model, which plays also an important role to introducing

non-linearity in the model. At the end of the CNN model, a fully-connected layer was applied to connect neurons with each other.

Our proposed CNN model is shown in Figure 5, which comprises an input layer, an output layer, and a few hidden layers. The hidden layers comprise several convolutional layers, maxpooling layers, and fully connected or dense layers. The input layer receives images of size (32, 32, 3), then the first convolution is performed using a 3×3 kernel to extract important features from each original image using the mathematical Formula (1), and, finally, the ReLu activation function is applied. To reduce the spatial features of the input images a 2×2 maxpooling is used, in which the maximum pixel value will be extracted in each operation. A total of four convolutional layers along with the maxpooling layer have been incorporated into our architecture. After extracting the important features, dense layers have been employed. In the output layerm a softmax activation function was applied to produce the required number of classes. However, we kept the padding the same to ensure that we got the images with exact pixels value in the output layer. In addition, batch normalization and a 20% dropout were applied. The batch normalization helps to accelerate the learning process during training, while dropout significantly prevents the model from overfitting and reduces computational expenses. Finally, in the output layer, the softmax activation function has been applied, in which linear input data turns into possible outcomes or classes by calculating the probability for each class using Equation (2) and returning the label of the class with the highest probability.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} for j = 0, 1, \ldots, K \tag{2}$$

Equation (2) produces a K-dimensional vector of probabilities, and subsequently the model returns the label with the highest probability.

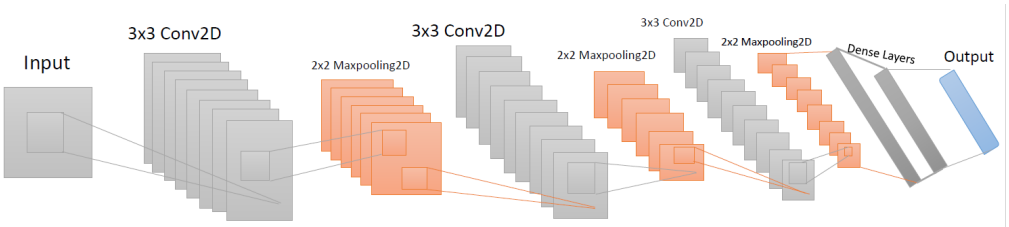
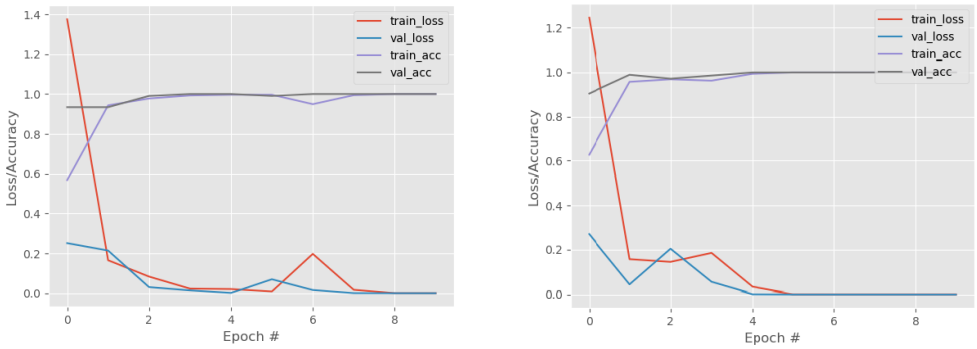


Figure 5. Proposed CNN model architecture.

The final architecture of our models (selection of the number of layers, their sizes, the optimizer, etc.) was established after a number of experimental trials, by manually adjusting these parameters. We selected the architecture that appeared to give the best results, given that all our tests were performed on a regular Intel Core i5-62000U CPU @ 2.30GHz 8GB RAM computer.

4.7. Model Training and Testing

We trained and tested our proposed CNN model and the other ML models (SVM, KNN, DT, RF, and XGBoost) with the Beowulf manuscript dataset. As we manually created our dataset by cropping each letter image for this purpose, our dataset is very small and it contains a total of 352 images in the training dataset and 88 images in the test dataset, with 22 classes one class per each alphabet letter. To overcome this small number of dataset points we resampled the training dataset once, twice, and thrice using ImageDataGenerator, a Keras image preprocessing package. After resampling once, twice, and thrice, the sizes of our training dataset became 704, 1056, and 1408, respectively. Figure 6 shows great improvements in training the model after resampling.



(**a**) (**b**)

Figure 6. Training and Validation Loss and Accuracy Curves over the epoch numbers (#) for the proposed CNN Model. (**a**) Loss/Accuracy Curves for Resampling 2 times. (**b**) Loss/Accuracy Curves for Resampling 3 times.

We then extracted the features from the training dataset for the cases of resampling once, twice, and thrice using three pre-trained models (VGG16, MobileNet, and ResNet50), and subsequently trained SVM, KNN, DT, RF, and XGBoost with the features extracted from each pre-trained model. Firstly, the features have been extracted by VGG16, MobileNet, and ResNet50 for the dataset with one-time resampling; with those extracted features the ML models (SVM, KNN, DT, RF, and XGBoost) were trained and tested. The evaluation and accuracy results are presented in Tables 1 and 2. Similarly, the pre-trained models were used to extract features from the dataset with resampling twice and thrice, the above-mentioned ML models were trained and tested, and the results were recorded in Table 2. Moreover, the proposed CNN model was also trained and tested using the dataset with resampling once, twice, and thrice. To train the CNN, model we used the categorical cross-entropy as our loss function and Adam as an optimizer. We used a batch size of 8, and the number of epochs was 10. We conducted a total of three experiments for our proposed CNN model, i.e., we trained the CNN model using the training dataset with resampling once and tested the model with the test dataset and recorded the results in Table 2. We trained the CNN model a second time and a third time using the training data sets with resampling twice and thrice, respectively. Subsequently, we tested the model and recorded results, which are shown in Table 2. To validate our proposed CNN model we trained and tested the model using the MNIST dataset and recorded the results in Table 2.

Table 1. Recall, Precision, and F1-score obtained from different ML models trained with the Beowulf dataset for resampling 1, 2, and 3 using the features extracted with VGG16, MobileNet, and ResNet50.

Pre-Trained Models	Resampling	Evaluation Metrices	SVM	KNN	Decision Tree	RF	XGBoost
VGG16	1	Recall	0.58	0.60	0.66	0.78	0.73
		Precision	0.53	0.65	0.68	0.83	0.76
		F1-score	0.57	0.60	0.65	0.79	0.73
	2	Recall	0.66	0.73	0.84	0.90	0.88
		Precision	0.71	0.75	0.85	0.91	0.89
		F1-score	0.68	0.72	0.84	0.90	0.88
	3	Recall	0.70	0.78	0.92	0.94	0.93
		Precision	0.74	0.81	0.92	0.95	0.94
		F1-score	0.71	0.78	0.91	0.94	0.93
MobileNet	1	Recall	0.51	0.58	0.69	0.74	0.73
		Precision	0.64	0.75	0.76	0.80	0.78
		F1-score	0.59	0.60	0.69	0.74	0.75
	2	Recall	0.54	0.65	0.79	0.82	0.81
		Precision	0.60	0.69	0.81	0.84	0.83
		F1-score	0.54	0.65	0.79	0.82	0.81
	3	Recall	0.56	0.62	0.89	0.89	0.90
		Precision	0.69	0.71	0.90	0.89	0.91
		F1-score	0.58	0.63	0.88	0.88	0.90
ResNet50	1	Recall	0.60	0.63	0.77	0.81	0.82
		Precision	0.58	0.70	0.82	0.85	0.81
		F1-score	0.61	0.65	0.78	0.81	0.79
	2	Recall	0.61	0.75	0.87	0.91	0.91
		Precision	0.64	0.77	0.88	0.92	0.92
		F1-score	0.63	0.75	0.89	0.91	0.91
	3	Recall	0.57	0.74	0.92	0.95	0.94
		Precision	0.64	0.77	0.92	0.95	0.94
		F1-score	0.58	0.74	0.92	0.95	0.94

Table 2. Average accuracy results over a few trials obtained with different ML models (SVM, KNN, DT, RF, and XGBoost, using the pre-trained models VGG16, MobileNet, and ResNet50) and the proposed CNN model on the Beowulf testing dataset for resampling 1, 2, and 3.

		Beowulf Manuscript's Dataset							MNIST Dataset
Models	**Resmp**	**SVM**	**KNN**	**DT**	**RF**	**XGBoost**	**CNN**		**CNN**
VGG16	1	52.16	56.84	66.10	78.81	73.72	Res. 1	88.67	99.03
	2	66.86	73.37	80.61	83.53	82.98			
	3	70.91	77.73	86.82	91.09	90.63			
MobileNet	1	51.55	58.54	69.17	74.44	73.68	Res. 2	90.91	
	2	54.08	65.56	79.88	82.24	81.88			
	3	56.59	61.82	89.09	89.09	90.45			
ResNet50	1	56.39	63.91	76.44	79.63	79.20	Res. 3	98.86	
	2	61.53	75.74	80.57	84.12	83.12			
	3	57.27	74.09	92.72	93.45	92.54			

4.8. Model Evaluation

To check the performance of each model, it is very important to evaluate the model by some metrics. The consistency of our CNN model has been explored using another well-established dataset, the MNIST dataset, along with our Beowulf manuscript's dataset. We performed the following evaluation analysis that helped determine the model's consistency:

- Confusion Matrix is very useful and one of the popular evaluation metrics in the field of ML research. Figures 7 and 8 are the confusion matrices to evaluate our CNN model for the cases of data resampling 2 and resampling 3, respectively.
- Recall is another popular evaluation metric for measuring whether a certain model's performance is consistent or not. The recall is calculated by Equation (3), which quantifies the true positives out of all actual positives.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Table 1 shows the recall values for the ML models we used in this study.

- Precision is an evaluation metric that also measures the performance of the model. Precision quantifies true positives computed by a model out of all predicted positives, which is shown by Equation (4).

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

Table 1 shows the precision values for the ML models we use in this study.

- $F1$-score is a very important measure to verify the test's accuracy. The harmonic mean of the recall and precision is considered as $F1$-score.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

The mathematical formula of the $F1$-score is given by the Equation (5). Table 1 shows the $F1$-scores values of all the ML models we use in this study.

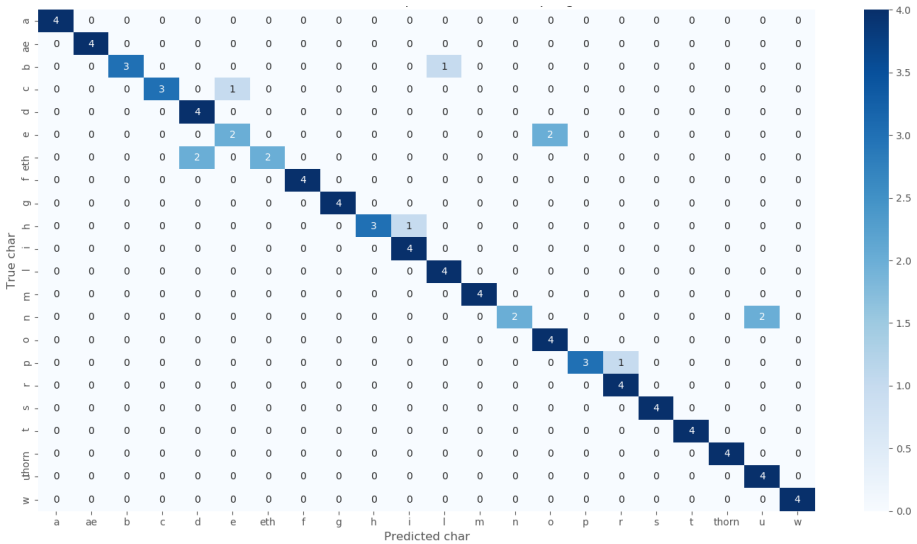


Figure 7. Confusion matrix for the proposed CNN Model in the case of Resampling 2.

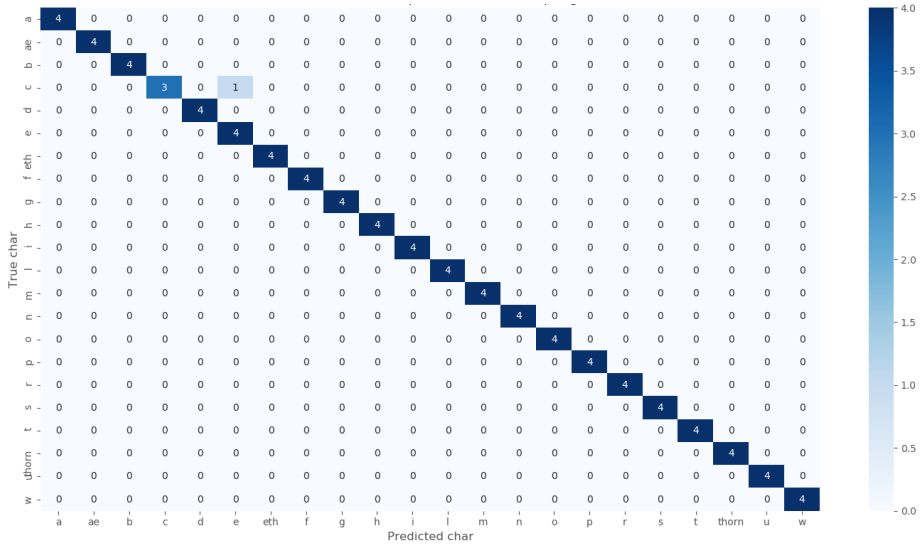


Figure 8. Confusion matrix for the proposed CNN Model in case of Resampling 3.

5. Results and Discussion

Extensive exploration has been performed for the recognition of Beowulf's manuscript Old English alphabet. Since we have a small dataset for the Beowulf manuscript, we used a Python package called ImageDataGenerator to produce more random data, which we subsequently call resampling. When we double the size of our dataset we call it resampling 1. Similarly, when we make the size of the dataset three times and four times of the original dataset we call it resampling 2 and resampling 3, respectively. We conducted our experiment several times for each resampling, i.e., for resampling 1, resampling 2, and resampling 3. Moreover, the Beowulf manuscript recognition has also been performed using other ML models such as Support Vector Machine (SVM), K-nearest Neighbor (KNN), Decision Tree (DT), Random Forest (RF), and XGBoost for the cases of resampling 1, resampling 2, and resampling 3. We also performed model validation of our CNN model for the well-recognized MNIST dataset. A detailed discussion of our experimental result is given in the rest of this section.

5.1. Beowulf Manuscript Character Recognition Using ML Models

To compare the results of our proposed CNN model for the Beowulf manuscript's dataset we conducted an exploratory study by training and testing several other ML models such as SVM, KNN, DT, RF, and XGBoost, and recorded the results for each ML model. At the very beginning, we extracted features from the Beowulf manuscript image dataset using several pre-trained models such as VGG16, MobileNet, and ResNet50 for for resampling 1, resampling 2, and resampling 3. After the feature extraction we trained, tested, and recorded the results for all of the above-mentioned ML models. In the next few subsections, we describe the results for each resampling case.

5.1.1. Resampling 1

For resampling 1, we conducted three sets of experiments using the pre-trained models VGG16, MobileNet, and ResNet50. In the first set of experiments, we trained all ML models (SVM, KNN, DT, RF, and XGBoost) using the extracted features by VGG16. We tested these ML models with the test dataset. In this case, the ML models produced test accuracies of 52.16%, 56.84%, 66.10%, 78.81%, and 73.72%, for SVM, KNN, DT, RF, and XGBoost, respectively. We subsequently got 51.55%, 58.54%, 69.17%, 74.44%, 73.68% accuracies from the models SVM, KNN, DT, RF, and XGBoost, respectively, when the features were extracted using the MobileNet pre-trained model. Finally, we observed 56.39%, 63.91%,

76.44%, 79.63%, and 79.20% accuracies from the models SVM, KNN, DT, RF, and XGBoost, respectively, when the features were extracted using the ResNet50 pre-trained model.

5.1.2. Resampling 2

For resampling 2, we conducted another three sets of experiments using the extracted features by pre-trained models VGG16, MobileNet, and ResNet50. In the first set of experiments, we trained all ML models (SVM, KNN, DT, RF, and XGBoost) using VGG16 for feature extraction. We tested these ML models with the test dataset. In this case, the ML models performed with recognition accuracies of 66.86%, 73.37%, 80.61%, 83.53%, and 82.98%, respectively, for the SVM, KNN, DT, RF, and XGBoost models. Following that, we got 54.08%, 65.56%, 79.88%, 82.24%, 81.88% of recognition accuracy on the test dataset from the models SVM, KNN, DT, RF, and XGBoost, respectively, when using the MobileNet pre-trained model for features extraction. Finally, we reported 61.53%, 75.74%, 80.57%, 84.12%, and 83.12% of recognition accuracy on the test dataset from the models SVM, KNN, DT, RF, and XGBoost, respectively, when using the ResNet50 pre-trained model for features extraction.

5.1.3. Resampling 3

Finally, in the case of resampling 3, we conducted three sets of experiments using the pre-trained models VGG16, MobileNet, and ResNet50. In the first set of experiments, we trained all other ML models using the features extracted by VGG16 pre-trained model. We tested these ML models with the test dataset. In this case, the ML models perform with test accuracies of 70.91%, 77.73%, 86.82%, 91.09%, and 90.63%, respectively, for SVM, KNN, DT, RF, and XGBoost, while we got 56.59%, 61.82%, 89.09%, 89.09%, 90.45% accuracies from the models SVM, KNN, DT, RF, and XGBoost, respectively, when the features were extracted by the pre-trained model MobileNet. Finally, we reported 57.27%, 74.09%, 92.72%, 93.45%, and 92.54% accuracies from the models SVM, KNN, DT, RF, and XGBoost, respectively, when the features were extracted by the pre-trained model ResNet50.

5.2. Beowulf Manuscript Character Recognition Using Proposed CNN Model

We trained and tested our CNN model using the Beowulf manuscript's dataset and recorded the results, which are shown in Table 2. However, in our original dataset, there are 352 images in the training dataset and 88 images in the test dataset of 22 classes, which are very small to examine our proposed model. To overcome the small size dataset issue, we used ImageDataGenerator, a Keras preprocessing library for augumenting a dataset with some pre-selected variation. We augmented our training dataset once, twice, and thrice, which we called resampling 1, resampling 2, and resampling 3, respectively. However, we did not augment our testing dataset. There are 88 images of 22 classes, i.e., $four$ images for each class. We tested the performance of our proposed CNN model with 88 images of 22 classes. We discuss our training and testing results in the following subsections for the cases resampling 1, resampling 2, and resampling 3.

5.2.1. Resampling 1

After the one-time augmentation of the training images, the training dataset became a dataset of 704 images. We trained our proposed model with 704 images and tested it with 88 images. After training the model for 10 epochs, our model achieved a 92.56% training accuracy and a 89.23% validation accuracy. Following that, the model achieved a 88.67% recognition accuracy while it was tested using the testing dataset.

5.2.2. Resampling 2

After two times augmentation of the training dataset, its size increased to 1056 images. We trained our proposed model with 1056 images and tested it with 88 images. After training the model for 10 epochs, our model achieved a 97.77% training accuracy and a 98.23% validation accuracy. The model achieved a 90.91% testing accuracy. The training

loss, training accuracy, validation loss, and validation accuracy curves, in this case, are shown in Figure 6a. For the case of resampling 2, while the model was tested with a testing dataset, it produced several incorrect character classifications. Our proposed model misclassified 'b' as 'i', 'c' as 'e' once, then 'e' as 'o', 'eth' as 'd', and 'n' as 'u' twice, as shown in Figure 7, consequently achieving a test accuracy of 90.91%.

5.2.3. Resampling 3

After the three augmentations of the training images, our dataset became a dataset of 1408 images. We trained our proposed model with 1408 images and tested it with 88 images. After training the model for 10 epochs, our model achieved a 100.00% training accuracy and a 99.98% validation accuracy. The model achieved a 98.86% test accuracy. The training loss, training accuracy, and validation loss, validation accuracy curve, in this case, are shown in Figure 6b. For the case of resampling 3, while the model was tested with a test dataset, it made just one error on recognizing the correct character. Our proposed model, in this case, misclassified 'c' as 'e' once, which is shown in Figure 8, resulting in a test accuracy of 98.86%.

Figure 9 shows the Receiver Operating Characteristic curve (ROC) for all models we used on the Beowulf manuscript dataset, and illustrates the diagnostic ability of all these multi-class classifiers. We must emphasize that reading these graphs can be somehow misleading at the first glance due to the very small size of the dataset and the relatively large number of classes (22). For each graph, most of the curves overlaps on the top-left corner of the image, making it impossible to find where each curve is. The graphs legends are very clarifying in this respect: most show an area of 1, which means the whole square is covered by the respective curve. There are some instances, like for the SVM classifications of classes 9, 17, and 18 (which all overlap almost over the diagonal), where classification performance is poor. This fact is clearly explained in the legend of the graph.

5.3. Proposed CNN Model Validation Using MNST Dataset

To accept a model, it is imperative to validate the newly proposed model with some other well-recognized models or datasets. In this investigation, we validated our proposed CNN model with the MNIST dataset. To train the proposed CNN model with the MNIST dataset we used the same loss function, optimizer, and learning rates as for training the model with the Beowulf manuscript's dataset. However, we trained our model with the MNIST dataset for 100 epochs, and in this case, we chose a batch size of 256. MNIST dataset is a well-established dataset, which achieves recognition accuracy of 99.83% by the EnsNet model, 99.87% by the Branching or Merging CNN model, 99.84% by the Efficient-CapsNet model, 99.83 by the SOPCNN model, 99.77% by the MCDNN model, 98.90% by the Convolutional net LeNet-4 model, and 99.05% by the Convolutional net LeNet-5 model [64]. Our proposed model performed very well not only for the Beowulf manuscript's dataset but also for the MNIST dataset. The proposed CNN model in this study achieved a recognition accuracy of 99.03%, when tested on the MNIST test dataset, and a training recognition accuracy of 100.00%. Thus, we can consider our proposed model as of similar performance with some of the other best models and hence an appropriate model for the task of character image recognition.

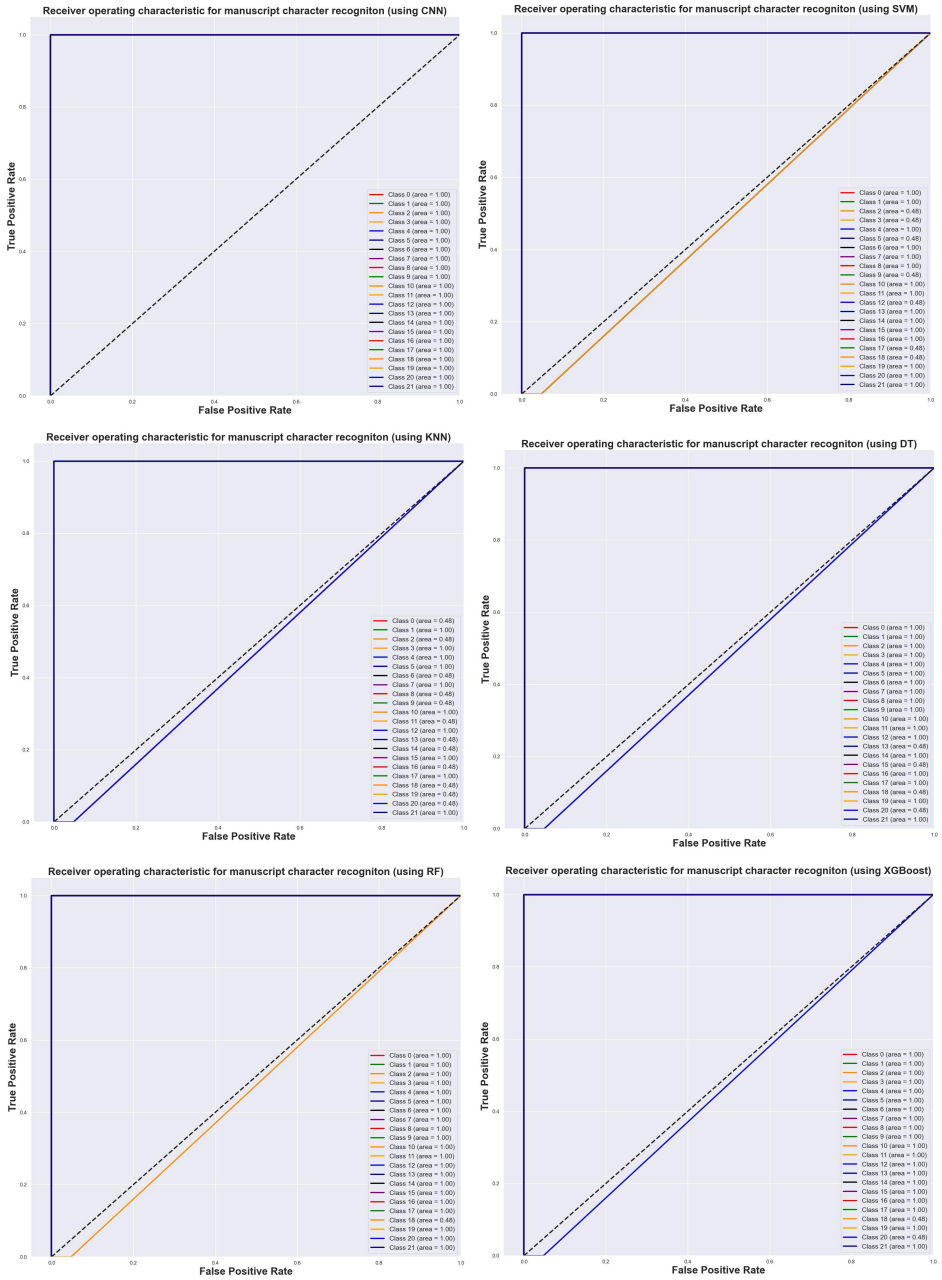


Figure 9. The ROC curves for CNN, SVM, KNN, DT, RF, and XGBoost models. Each curve represents the best results produced using the Beowulf testing dataset, for resampling 3.

6. Conclusions and Future Work

In this work, we present a Convolutional Neural Network (CNN) model for manuscript characters classification in general, and, in particular, for classifying the Beowulf manuscript's Old English characters. Our approach may help machine learning researchers in automated character recognition of ancient manuscripts, which may open a new horizon for researchers in history, arts, literature, science, and technology. For the experiments in our study we built our own characters dataset from the manuscript's electronic images and made this dataset available to other researchers interested in the subject mater. To perform the classification task we developed a CNN model, trained and tested this model with the Beowulf manuscript's dataset. Moreover, we performed comparative studies by using some other Machine Learning (ML) models such as Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Decision Tree (DT), Random Forest (RF), and XGBoost. We used pre-trained models such as VGG16, MobileNet, and ResNet50 to extract features

from the Beowulf manuscript character images dataset, used those features to train the SVM, KNN, DT, RF, XGBoost models, and subsequently tested these models with our Beowulf test dataset. We recorded the recognition accuracy results and evaluated each model performance by computing recall, precision, and F1 scores values. For our CNN model and all the other models we compared our model with what we have displayed the ROC curves (Figure 9) to illustrate the classification performance of each model. The Beowulf manuscript character images dataset was augumented once (resampling 1), twice (resampling 2), and thrice (resampling 3), and our proposed CNN model fared very well compared to the other ML models, with the highest accuracy of 98.86% in the case of resampling 3. The complete numerical results can be found in Table 1. Moreover, our CNN model has been trained and tested with the MNIST handwritten digits dataset, and it achieved a benchmark recognition accuracy of 99.03%.

Although we successfully tackled the small dataset problem in our study, by augmenting our data, building a CNN model, and validating this model with a well-established MNIST dataset, future work can focus on automatically creating a large enough repository of Old English character images so that such a robust dataset can be used to train and test any machine and deep learning models to recognize Old English character images.

Author Contributions: Conceptualization, M.A.I. and I.E.I.; initial methodology, I.E.I.; initial draft M.A.I.; software, M.A.I.; final validation, M.A.I. and I.E.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The Beowulf manuscript character images dataset used in this study can be found here: https://bit.ly/OECharacterSet (accessed on 28 February 2023).

Acknowledgments: The authors would like to thank the external reviewers and academic editor for their patience in reviewing this paper and the useful comments that improved the quality of the paper overall. We would also want to thank the technical editorial team for quickly and professionally addressing all technical issues while editing this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Saqib, N.; Haque, K.F.; Yanambaka, V.P.; Abdelgawad, A. Convolutional-Neural-Network-Based Handwritten Character Recognition: An Approach with Massive Multisource Data. *Algorithms* **2022**, *15*, 129. [CrossRef]
2. Alom, M.Z.; Sidike, P.; Hasan, M.; Taha, T.M.; Asari, V.K. Handwritten bangla character recognition using the state-of-the-art deep convolutional neural networks. *Comput. Intell. Neurosci.* **2018**, *2018*, 6747098. [CrossRef] [PubMed]
3. Artese, M.T.; Gagliardi, I. Methods, Models and Tools for Improving the Quality of Textual Annotations. *Modelling* **2022**, *3*, 224–242. [CrossRef]
4. Kiernn, K.; Iacob, I.E. Electronic Beowulf, CD-ROM, British Library, 3rd edition, October 2011. Available online: https://ebeowulf.uky.edu/ (accessed on 28 February 2023).
5. Library, B. British Library Collection Items. Available online: https://www.bl.uk/collection-items/beowulf(Website) (accessed on 28 February 2023).
6. Wikipedia. Wikipedia, Historical Background. Available online: https://en.wikipedia.org/wiki/Beowulf (accessed on 28 February 2023).
7. Harrison, J.A.; Sharp, R. The Project Gutenberg eBook of Beowulf. 2003. Available online: https://www.gutenberg.org/files/9700/9700-h/9700-h.htm (accessed on 28 February 2023).
8. Sutradhar, S. Old English Character Recognition Using Neural Networks 2018. Electronic Theses and Dissertations, Georgia Southern University. Available online: https://digitalcommons.georgiasouthern.edu/etd/1783/ (accessed on 28 February 2023).
9. Islam, M.A. Reduced Dataset Neural Network Model for Manuscript Character Recognition 2020. Electronic Theses and Dissertations, Georgia Southern University. Available online: https://digitalcommons.georgiasouthern.edu/etd/2138/ (accessed on 28 February 2023).
10. Kesiman, M.W.A.; Valy, D.; Burie, J.C.; Paulus, E.; Suryani, M.; Hadi, S.; Verleysen, M.; Chhun, S.; Ogier, J.M. Benchmarking of document image analysis tasks for palm leaf manuscripts from southeast asia. *J. Imaging* **2018**, *4*, 43. [CrossRef]
11. Suryani, M.; Paulus, E.; Hadi, S.; Darsa, U.A.; Burie, J.C. The handwritten sundanese palm leaf manuscript dataset from 15th century. In Proceedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; Volume 1, pp. 796–800.

12. Hidayat, A.A.; Purwandari, K.; Cenggoro, T.W.; Pardamean, B. A convolutional neural network-based ancient sundanese character classifier with data augmentation. *Procedia Comput. Sci.* **2021**, *179*, 195–201. [CrossRef]
13. Sutramiani, N.P.; Suciati, N.; Siahaan, D. MAT-AGCA: Multi Augmentation Technique on small dataset for Balinese character recognition using Convolutional Neural Network. *ICT Express* **2021**, *7*, 521–529. [CrossRef]
14. Sutramiani, N.P.; Suciati, N.; Siahaan, D. Transfer learning on balinese character recognition of lontar manuscript using MobileNet. In Proceedings of the 2020 4th International Conference on Informatics and Computational Sciences (ICICoS), Semarang, Indonesia, 10–11 November 2020; pp. 1–5.
15. Hazra, A.; Choudhary, P.; Inunganbi, S.; Adhikari, M. Bangla-Meitei Mayek scripts handwritten character recognition using convolutional neural network. *Appl. Intell.* **2021**, *51*, 2291–2311. [CrossRef]
16. Hoq, M.N.; Nipa, N.A.; Islam, M.M.; Shahriar, S. Bangla handwritten character recognition: an overview of the state of the art classification algorithm with new dataset. In Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, 3–5 May 2019; pp. 1–6.
17. Rahman, M.M.; Akhand, M.; Islam, S.; Shill, P.C.; Rahman, M. Bangla handwritten character recognition using convolutional neural network. *Int. J. Image Graph. Signal Process. IJIGSP* **2015**, *7*, 42–49. [CrossRef]
18. Alif, M.A.R.; Ahmed, S.; Hasan, M.A. Isolated Bangla handwritten character recognition with convolutional neural network. In Proceedings of the 2017 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 22–24 December 2017; pp. 1–6.
19. Chowdhury, R.R.; Hossain, M.S.; ul Islam, R.; Andersson, K.; Hossain, S. Bangla handwritten character recognition using convolutional neural network with data augmentation. In *Proceedings of the 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*; IEEE: Piscataway, NJ, USA, 2019; pp. 318–323.
20. Sazal, M.M.R.; Biswas, S.K.; Amin, M.F.; Murase, K. Bangla handwritten character recognition using deep belief network. In Proceedings of the 2013 International Conference on Electrical Information and Communication Technology (EICT), Khulna, Bangladesh, 13–15 February 2014; pp. 1–5.
21. Nongmeikapam, K.; Wahengbam, K.; Meetei, O.N.; Tuithung, T. Handwritten Manipuri Meetei-Mayek classification using convolutional neural network. *ACM Trans. Asian Low Resour. Lang. Inf. Process. TALLIP* **2019**, *18*, 1–23. [CrossRef]
22. Devi, S.G.; Vairavasundaram, S.; Teekaraman, Y.; Kuppusamy, R.; Radhakrishnan, A. A Deep Learning Approach for Recognizing the Cursive Tamil Characters in Palm Leaf Manuscripts. *Comput. Intell. Neurosci.* **2022**. [CrossRef]
23. Sudarsan, D.; Joseph, S. A novel approach for handwriting recognition in malayalam manuscripts using contour detection and convolutional neural nets. In Proceedings of the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 19–22 September 2018; pp. 1818–1824.
24. Alrehali, B.; Alsaedi, N.; Alahmadi, H.; Abid, N. Historical Arabic manuscripts text recognition using convolutional neural network. In Proceedings of the 2020 6th Conference on Data Science and Machine Learning Applications (CDMA), Riyadh, Saudi Arabia, 4–5 March 2020; pp. 37–42.
25. Singh, A.K.; Kadhiwala, B.; Patel, R. Hand-written Hindi Character Recognition-A Comprehensive Review. In Proceedings of the 2021 2nd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 1–3 October 2021; pp. 1–5.
26. Wibowo, M.A.; Soleh, M.; Pradani, W.; Hidayanto, A.N.; Arymurthy, A.M. Handwritten javanese character recognition using descriminative deep learning technique. In Proceedings of the 2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Yogyakarta, Indonesia, 1–2 November 2017; pp. 325–330.
27. Meier, U.; Ciresan, D.C.; Gambardella, L.M.; Schmidhuber, J. Better digit recognition with a committee of simple neural nets. In Proceedings of the 2011 International Conference on Document Analysis and Recognition, Beijing, China, 18–21 September 2011; pp. 1250–1254.
28. Adak, C.; Chaudhuri, B.B.; Blumenstein, M. Offline cursive Bengali word recognition using CNNs with a recurrent model. In Proceedings of the 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), Shenzhen, China, 23–26 October 2016; pp. 429–434.
29. LeCun, Y.; Cortes, C.; Burges, C.J. The MNIST Database of Handwritten Digits. 1998. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 11 May 2022).
30. Baldominos, A.; Saez, Y.; Isasi, P. A survey of handwritten character recognition with MNIST and EMNIST. *Appl. Sci.* **2019**, *9*, 3169. [CrossRef]
31. Sayeed, A.; Shin, J.; Hasan, M.A.M.; Srizon, A.Y.; Hasan, M.M. BengaliNet: A Low-Cost Novel Convolutional Neural Network for Bengali Handwritten Characters Recognition. *Appl. Sci.* **2021**, *11*, 6845. [CrossRef]
32. Tensmeyer, C.; Saunders, D.; Martinez, T. Convolutional neural networks for font classification. In Proceedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; Volume 1; pp. 985–990.
33. Kausar, T.; Manzoor, S.; Kausar, A.; Lu, Y.; Wasif, M.; Ashraf, M.A. Deep Learning Strategy for Braille Character Recognition. *IEEE Access* **2021**, *9*, 169357–169371. [CrossRef]
34. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]

35. Dong, C.; Loy, C.C.; He, K.; Tang, X. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 295–307. [CrossRef]
36. Nogra, J.A.; Romana, C.L.S.; Balakrishnan, E. Baybáyin character recognition using convolutional neural network. *Int. J. Mach. Learn. Comput.* **2020**, *10*, 169–186. [CrossRef]
37. Ahlawat, S.; Choudhary, A.; Nayyar, A.; Singh, S.; Yoon, B. Improved handwritten digit recognition using convolutional neural networks (CNN). *Sensors* **2020**, *20*, 3344. [CrossRef]
38. Bala, D.; Mynuddin, M.; Hossain, M.I.; Islam, M.A.; Hossain, M.A.; Abdullah, M.I. A Robust Plant Leaf Disease Recognition System Using Convolutional Neural Networks. In Proceedings of the 2022 International Conference on Engineering and Emerging Technologies (ICEET), Kuala Lumpur, Malaysia, 27–28 October 2022; pp. 1–6. [CrossRef]
39. Bala, D.; Islam, M.A.; Hossain, M.I.; Mynuddin, M.; Hossain, M.A.; Hossain, M.S. Automated Brain Tumor Classification System using Convolutional Neural Networks from MRI Images. In Proceedings of the 2022 International Conference on Engineering and Emerging Technologies (ICEET), Kuala Lumpur, Malaysia, 27–28 October 202; pp. 1–6. [CrossRef]
40. Elmansouri, M.; Makhfi, N.E.; Aghoutane, B. Toward classification of arabic manuscripts words based on the deep convolutional neural networks. In Proceedings of the 2020 International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, 9–11 June 2020; pp. 1–5.
41. Liu, X.; Hu, B.; Chen, Q.; Wu, X.; You, J. Stroke sequence-dependent deep convolutional neural network for online handwritten chinese character recognition. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4637–4648. [CrossRef]
42. Diem, M.; Fiel, S.; Garz, A.; Keglevic, M.; Kleber, F.; Sablatnig, R. ICDAR 2013 competition on handwritten digit recognition (HDRC 2013). In Proceedings of the 2013 12th International Conference on Document Analysis and Recognition, Washington, DC, USA, 25–28 August 2013; pp. 1422–1427.
43. Ahmed, S.B.; Naz, S.; Razzak, M.I.; Yousaf, R. Deep learning based isolated Arabic scene character recognition. In Proceedings of the 2017 1st International Workshop on Arabic Script Analysis and Recognition (ASAR), Nancy, France, 3–5 April 2017; pp. 46–51.
44. Manocha, S.K.; Tewari, P. Devanagari Handwritten Character Recognition using CNN as Feature Extractor. In Proceedings of the 2021 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON), Pune, India, 29–30 October 2021; pp. 1–5.
45. Tamir, K. Handwritten Amharic characters recognition using CNN. In Proceedings of the 2019 IEEE AFRICON, Accra, Ghana, 25–27 September 2019; pp. 1–4.
46. Narang, S.R.; Kumar, M.; Jindal, M.K. DeepNetDevanagari: A deep learning model for Devanagari ancient character recognition. *Multimed. Tools Appl.* **2021**, *80*, 20671–20686. [CrossRef]
47. Jangid, M.; Srivastava, S. Handwritten devanagari character recognition using layer-wise training of deep convolutional neural networks and adaptive gradient methods. *J. Imaging* **2018**, *4*, 41. [CrossRef]
48. Warkhad, A.; Mandhare, S.; Thombre, Y.; Korhale, P.; Deore, S.; Ingle, S. Hybrid Approach For Handwritten Devanagari Character Recognition Using CNN and KNN 2021. Available online: https://www.irjet.net/archives/V8/i2/IRJET-V8I2176.pdf (accessed on 15 November 2022).
49. Bala, R.; Singh, C. An optimized CNN-based handwritten gurmukhi character recognition from punjabi script image. *Int. J. Sci. Res. Comput. Sci. Appl. Manag. Stud.* **2020**, *9*, 1–10.
50. Alrasheed, N.; Rao, P.; Grieco, V. Character Recognition of Seventeenth-Century Spanish American Notary Records Using Deep Learning. *Digit. Humanit. Q.* **2021**, *15*. Available online: http://www.digitalhumanities.org/dhq/vol/15/4/000581/000581.html (accessed on 28 February 2023).
51. Ranzato, M.; Poultney, C.; Chopra, S.; Cun, Y. Efficient learning of sparse representations with an energy-based model. *Adv. Neural Inf. Process. Syst.* **2006**, *19*, 1137–1144.
52. Ciresan, D.C.; Meier, U.; Gambardella, L.M.; Schmidhuber, J. Convolutional neural network committees for handwritten character classification. In Proceedings of the 2011 International Conference on Document Analysis and Recognition, Beijing, China, 18–21 September 2011; pp. 1135–1139.
53. Wan, L.; Zeiler, M.; Zhang, S.; Le Cun, Y.; Fergus, R. Regularization of neural networks using dropconnect. In Proceedings of the International Conference on Machine Learning, PMLR, Atlanta, GA, USA, 16–21 June 2013; pp. 1058–1066.
54. Kowsari, K.; Jafari Meimandi, K.; Heidarysafa, M.; Mendu, S.; Barnes, L.; Brown, D. Text classification algorithms: A survey. *Information* **2019**, *10*, 150. [CrossRef]
55. Vijayan, V.K.; Bindu, K.; Parameswaran, L. A comprehensive study of text classification algorithms. In Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Manipal, Karnataka, India, 13–16 September 2017; pp. 1109–1113.
56. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
57. Silva, I.; Eugenio Naranjo, J. A Systematic Methodology to Evaluate Prediction Models for Driving Style Classification. *Sensors* **2020**, *20*, 1692. [CrossRef]
58. Buldin, I.D.; Ivanov, N.S. Text Classification of Illegal Activities on Onion Sites. In Proceedings of the 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg and Moscow, Russia, 27–30 January 2020; pp. 245–247.

59. Tan, Y. An improved KNN text classification algorithm based on K-medoids and rough set. In Proceedings of the 2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, China 25–26 August 2018; Volume 1; pp. 109–113.
60. Song, Y.Y.; Lu, Y. Decision tree methods: applications for classification and prediction. *Shanghai Arch. Psychiatry* **2015**, *27*, 130–135.
61. Chen, T.; Guestrin, C. *XGBoost: A Scalable Tree Boosting System*; Association for Computing Machinery: New York, NY, USA, 2016; KDD '16; pp. 785–794. [CrossRef]
62. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
63. O'Shea, K.; Nash, R. An Introduction to Convolutional Neural Networks. *arXiv* **2015**, arXiv:1511.08458.
64. Published Online and Open Source. Image Classification on MNIST. 2022. Available online: https://paperswithcode.com/sota/image-classification-on-mnist (accessed on 20 April 2022).