Convolutional Neural Networks for Handwritten Character Recognition

Bishnoi, Rajat
University of Florida
Gainesville, Florida
rajatbishnoi@ufl.edu

Hutcheson, Cody
University of Florida
Gainesville, Florida
c.hutcheson@ufl.edu

Overmeyer, James
University of Florida
Gainesville, Florida
jovermeyer@ufl.edu

***Abstract*—Handwritten character recognition is one of the marquee problems in the world of machine learning because different people have different handwriting patterns. Convolutional Neural Networks show great potential for recognizing handwritten characters, and this can be leveraged to improve the performance of character recognition systems. In this report a convolutional neural network was implemented to classify 10 different handwritten characters. A CNN model was trained using thousands of handwritten characters, and the model resulted in a 96% classification accuracy for a test dataset. This research shows the massive potential for convolutional neural networks to be extremely accurate handwritten character classifiers.**

Keywords—Machine Learning, Convolutional Neural Network (CNN), Handwritten Character Classification,

I. Introduction

Artificial Intelligence (AI) is the field that combines computer science, data science, and powerful computers to enable learning and problem-solving. Within the field of AI, machine learning is a powerful tool that utilizes large datasets and statistical algorithms to imitate human learning and perform pattern recognition [1,2]. One of the common areas of interest in machine learning is the problem of handwritten character recognition or natural language processing. Human beings can identify handwritten characters of a massive variety of handwriting styles. The ability for machines to be able to recognize handwritten characters, similarly to humans, can be an extremely powerful tool, allowing machines to solve many types of problems that would take humans a much longer amount of time. Through the implementation of machine learning algorithms, machines can process and detect the characters associated with a variety of languages.

A variety of machine learning algorithms have been implemented to perform hand-written character recognition. In the paper written by Naik, A and Desai, V, k-Nearest Neighbor and Support Vector Machine were utilized to classify Gujarati characters. k-Nearest Neighbor is a distance-based recognition method that computes the Euclidean distance between all samples within a training dataset. The k nearest neighbors to a data point are used to classify that data point such that the majority class of the neighbors determine that point's class. The authors achieved an accuracy of 90% using 7-Nearest neighbors. Support Vector Machine transforms data into higher dimensions and will separate the classes using a decision hyperplane that maximizes the margin between the hyperplane and the datapoints to promote generalization. The SVM utilized by the authors of the paper achieved an accuracy of 92% using a radial kernel [9]. In Herman, et al. the Gaussian Naïve Bayes algorithm was utilized for number recognition. Gaussian Naïve Bayes is a classifier that utilizes the Bayes rule to calculate the conditional probability a datapoint belongs to a certain class. The Gaussian Naïve Bayes algorithm resulted in only 28.33% accuracy [13]. These machine learning algorithms should be improved upon as the accuracy of classification would ideally be 100%.

The most common machine learning technique utilized for hand-written character recognition is neural networks. A neural network consists of input nodes, hidden nodes, and output nodes. In the input layer no computation is performed, and the data is passed to the neural network. Hidden nodes perform computations and transfer information to following hidden layers. The output layer performs calculations resulting in an output. Pal, A and Singh, D utilized a multi-layer perceptron neural network to perform English character recognition. Using, one hidden layer and a backpropagation algorithm the authors were able to achieve 94% recognition accuracy [12]. Convolutional Neural Networks (CNNs) are artificial neural networks that are primarily utilized to process images. CNNs typically consist of three layers that include convolution, pooling, and activation. Convolutional layers consist of N convolutional kernels that scan the image looking for different patterns. These convolutional layers differ from traditional neural network hidden layers in that convolutional layers are not fully connected. Pooling layers decrease the number of pixels in an image through averaging. The activation layer consists of an activation function that will transform the feature matrix to some output. Al-Mahmud, Tanvin, A and Rahman, A implemented a convolutional neural network. Their implementation included an 28x28 image input layer, a first convolutional layer of 32 filters with 5x5 kernels, a max-pooling layer of 2x2 to reduce the image volume, another convolutional layer of 64 filters with 7x7 kernels, then a second max-pooling layer of 2x2 to create 64 4x4 feature maps. These 64 4x4 feature maps were flattened an input to a perceptron with one hidden layer and one output layer. This method achieved 99.47% accuracy for the MNIST dataset [4].

For handwritten character recognition, pre-processing of input data is a vital aspect to achieving a highly accurate model. Dhande, P and Kharat, R implemented and detailed a variety of pre-processing techniques for images of handwritten characters. In pre-processing the noise reduction techniques can be divided into filtering for smoothing and sharpening of the image, morphological operations for character thinning and boundary extraction, and noise modeling. The most common preprocessing technique is morphological operations. The benefit of morphological operations in image preprocessing is that these operations can remove image imperfections and emphasize the structure and features of these images. The two main types of morphological operations include dilation which adds pixels to boundaries and erosion which removes pixels from boundaries [5].

In our research we tested a few machine learning algorithms for written character recognition before settling on utilization of a convolutional neural network. The models in this experiment utilized a labeled training dataset of 6720 300x300 images of hand-written characters a-h, #, and $. Uniform preprocessing was performed, and the different machine learning algorithms were implanted on the processed data. The final model presented utilized a convolutional neural network, and an accuracy of greater than 95% was achieved for classifying the training data.

II. Implementation

A. Dataset

The dataset consisted of a collection of hand-written character images, where each image contained either a letter in the range of A-H, the dollar sign symbol ($), or the hash symbol (#). To ensure a richer dataset and consequently a more generalizable model, the handwritten characters were purposely collected to account for variability. This includes the writing utensil used, whether the letter was lower or upper case, and whether the letter was written in print or cursive. In total, the labelled training dataset was comprised of 6720 300x300 images. The raw images were resized to 300x300 and set to grayscale to create the labelled training dataset.

B. Preprocessing

A necessary condition for achieving an accurate CNN model is image preprocessing. Image preprocessing was accomplished using the OpenCV library, which provides several morphological operations that can be performed on image data. Morphological operations require the use of a kernel, which describes the neighborhood of pixels that are considered around each pixel when performing a given operation. It was found that for the given dataset, a 4x4 kernel matrix performed best with the given operations.

Our implementation utilized two morphological operations. Median blur was the first operation applied on the data, which takes the average of the neighboring pixels to calculate a new pixel value for the output. This had the effect of smoothing the edges of each character. This operation is very important as it slightly blurs the background lines, which can be seen in our data. Next, the MORPH_OPEN operation was performed, which is a combination of erosion, an operation which qualitatively "thins" the characters. Finally, the resize function was used to make each image smaller (50x50), and used parameter INTER_AREA to perform the necessary decimation. The dataset was then split into a training set and a test set using a 80/20 ratio, and the pixels were scaled by a factor of 255 so that pixel values fell in the range of 0-1.

C. Convolutional Neural Network (CNN)

A convolutional neural network was used to classify the images into one of the ten-character classes. This involved the use of Keras, which is an API which was imported from the TensorFlow library. A sequential model was used, which is suitable for applications where there is only one input and one output to our model, which is true for our character recognition task. The CNN contains 23 layers in total, which were created sequentially using the add() function.

The model first began with two data augmentation layers: RandomContrast and RandomRotation. The purpose of these data augmentation layers is to introduce random, yet realistic transformations on the input data so as to increase diversity of the training data. RandomContrast randomly adjusted the contrast of individual color channel, whereas RandomRotation randomly rotated the images.

After data augmentation, there is a "block" of layers that we repeated three times. At the beginning of this block is two calls to Conv2D, which creates a two-dimensional convolution layer. Every convolution layer used in our CNN made use of the He Normal kernel initializer, meaning it initializes the kernel matrix to have random values pulled from a variation of the normal distribution called the He Normal. The choice of kernel initializer is important as too small of values could have caused vanishing gradients and too big of values could have caused the output to diverge. Each of the convolution layers also used the rectified linear unit (ReLU) activation function. ReLU is a very popular activation function to use for CNN's as they are computationally fast, resistant to the vanishing gradient problem, and have sparse activation [10]. After the activation function is applied, a batch normalization layer is applied. Batch normalization is a technique which is used to combat "covariate shift", which is a problem in machine learning which leads to misclassification [11]. Batch normalization results also results in allowing for higher learning rates, which results in faster CNN [11]. MaxPooling2D follows the batch normalization, and this is a pooling layer where it down sampled the output into a simpler image while keeping the information about the most important features. Max pooling accomplished this by taking the maximum value of a pixel in a smaller sub-region. Finally, Dropout is a layer which randomly sets certain output values to zero, which is a regularization technique that helps to avoid overfitting the model.

A Flatten layer is used after the previously described block was carried out three times. It transforms the multidimensional input tensors into a single dimension representing all dimensions. Three Dense layers follow this layer. Dense layers are like convolutional layers in that they use an activation function and a kernel to produce an output image. Each dense layer has a fully connected neural network structure, and these layers are what are responsible for the final classification decisions of the entire CNN. The first two layers used the same ReLU activation function as before, and the third layer used the softmax activation function. Softmax is used here as it is a common choice for the final layer of a CNN, given that it produces an output where the sum of all outputs sums to 1. This allows us to use this output of the CNN as an input to the cross-entropy loss function, which allows for us a method for backpropagation. Finally, the model was compiled using the Adam optimizer. The Adaptive moment estimation optimizer combines momentum optimization and an adaptive learning rate to speed up optimization. A table showing the layers of the CNN and output shapes is shown in Table 1, below.

TABLE I. CNN LAYERS AND OUTPUT SHAPES

Layer	Output Shape
RandomContrast	(50,50,1)
RandomRotation	(50,50,1)
Conv2d	(46,46,32)
Conv2d	(42,42,32)
BatchNormalization	(42,42,32)
MaxPooling2D	(21,21,32)
Dropout	(21,21,32)
Conv2D	(19,19,32)
Conv2D	(17,17,64)
BatchNormalization	(17,17,64)
MaxPooling2D	(8,8,64)
Dropout	(8,8,64)
Conv2D	(6,6,64)
Conv2D	(4,4,64)
BatchNormalization	(4,4,64)
MaxPooling2D	(2,2,64)
Dropout	(2,2,64)
Flatten	(256)
Dense	(256)
Dropout	(256)
Dense	(128)
Dropout	(128)
Dense	(10)

D. *Model Fitting and Interpretation*

After all the layers have been added to the CNN, the model is then fitted with the training data, and then evaluated using the test data. We used 800 epochs for out CNN, meaning that we will go through 800 complete passes in the CNN. Evaluating the testing set returns a list of scores for each epoch, and information regarding each epoch is displayed on the screen after running the program. Finally, a graph of the cross-entropy loss as a function of the epoch is displayed.

III. EXPERIMENTS

The preprocessing steps described in the previous section were achieved through trial and error and observations of the data before and after processing. The goal of the preprocessing was to darken and thicken the lines forming the handwritten characters while lightening the background and thinning background lines. Various combinations of dilations and erosions were tested as well as non-linear filters like median blurring. The pre-preprocessing approach that yielded the best results visually and best model accuracy was first applying 3x3 median blur filter to remove noise and minimize edge blurring, second utilizing a 4x4 morph open kernel to further remove background noise, and finally resizing the images to 50x50. An example of this preprocessing is shown in Figure 1. After preprocessing the handwritten dollar sign is darker compared to the background, and the background lines are faded compared to before preprocessing, thus visually confirming the success of this preprocessing approach.

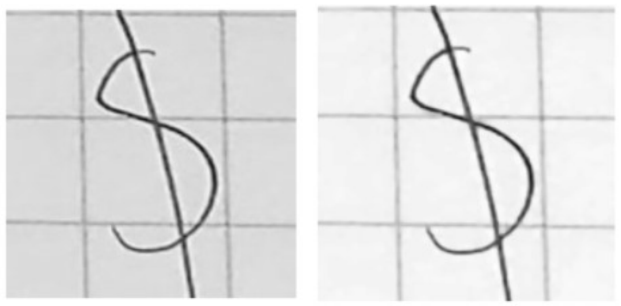

Figure 1: Dollar sign before(L) and after(R) preprocessing

Prior to utilizing a CNN, 3 other machine learning approaches were tested for classifying the handwritten characters.

Naïve Bayes: As previously mentioned, the Naïve Bayes classifier utilizes the Bayes rule and assigns classes based on equations 1 and 2 by calculating the posterior probability that a test point belongs to a certain class. This Naïve Bayes approach was utilized with this dataset to achieve a classification accuracy of 34%. This low accuracy is likely because a Gaussian distribution was assumed for the prior probability and the data may not have been Gaussian distributed.

$$\textit{Class 1: } P(x^*|C_1)P(C_1) > P(x^*|C_2)P(C_2) \quad (1)$$

$$\textit{Class 2: } P(x^*|C_2)P(C_2) > P(x^*|C_1)P(C_1) \quad (2)$$

k-Nearest Neighbors: The k-NN approach classifies data points based on what class most of its neighbors belong to. For example, if 3 nearest neighbors are utilized, and 2 neighbors are of class 1, the test point will be assigned to class 1 because most of its neighbors are also class 1. Utilizing 3 nearest neighbors by Euclidean distance resulted in 46% classification accuracy. This relatively low accuracy is likely due to the high dimensionality of the feature space and utilizing Euclidean distance.

Fisher Linear Discriminant Analysis: Linear Discriminant Analysis utilizes linear combinations of features to separate classes. This approach may also decrease the dimensionality of the input by projection it in the most important directions. An example of 2-dimensional LDA is shown in Figure 2. This example demonstrates linear separation of two classes in 2-dimensions utilizing LDA. For handwritten character classification, LDA utilizing 5 components resulted in a classification accuracy of 37.5%. This low accuracy was likely caused by oversimplifying the model to 5 components but utilizing many components within this model would be computationally expensive.

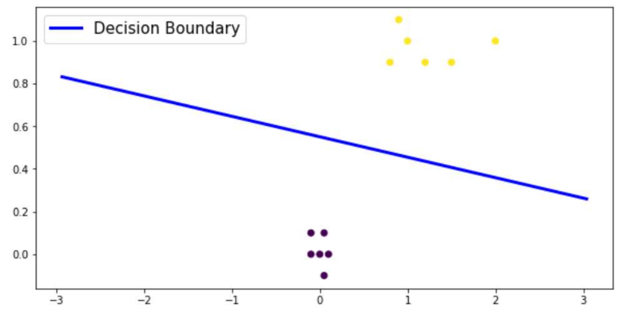

Figure 2: 2-Dimensional LDA decision boundary

Overall, the failure of these 3 classification approaches pointed us in the direction of utilizing a convolutional neural network as described in the implementation section.

To experimentally test the convolutional neural network model, grid search cross-validation was utilized to determine the best parameter values for training the model. Grid search cross-validation iteratively tests every combination of provided parameter values and outputs the best combination of parameter values. The parameters that were utilized were epochs, where 1 epoch is when each data point is passed forward and backward through the neural network once, batch size, which is the batch size of datapoints passed forward and backward through the network at one time, and learning rate, which is utilized to scale the magnitude of parameter updates during Adam optimization. To perform this cross-validation grid search, the data was divided into 80% training data and 20% test data. A stratified 3-fold cross-validation procedure was implemented on the training data to test each set of parameters performance. The average performance utilizing each combination of 2 training folds and 1 validation folds was taken to determine the performance of each set of parameters. The best set of model parameters was determined to be 800 epochs, a batch size of 32, and a learning rate of 0.0007. These model parameters were then utilized to retrain the model based on the entirety of the training data. This model was then utilized to classify the test data points. Validation of the model utilizing the test data points resulted in a classification accuracy of 96+%. Figure 3 shows the convergence of the training accuracy and validation accuracy to 95+% after around 600 epochs. The training and validation accuracies are shown to be similar, so the model does not present and noticeable overfitting and underfitting.
Finally, the final model was trained utilizing the complete dataset. Only 650 epochs were utilized for training the final model as Figure 3 shows convergence of the accuracies around 600 epochs. This model can be utilized to classify future and unknown handwritten character datasets. For an easy dataset with no new characters the model will simply classify the characters based on which classification has the highest probability for a given character. For a dataset with new characters the model will similarly classify the characters based on probability, however if a character does not belong to a class with greater than 80% probability, it will be classified as an unknown class(-1).

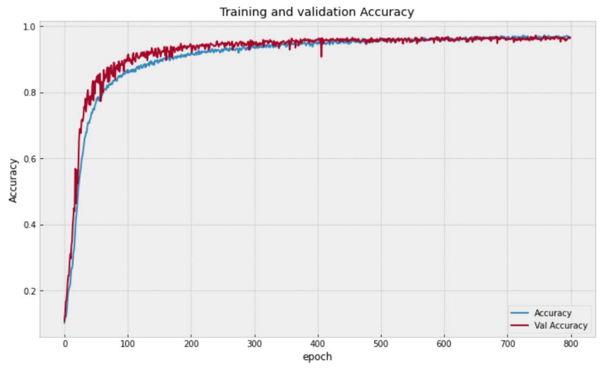

Figure: 3 Training Accuracy and Validation accuracy of the trained CNN

IV. Conclusions

After completing the experiments detailed in this paper, it is clear that many approaches may be utilized for classification during machine learning. The convolutional neural network exhibited the highest accuracy of any of the tested methods. The convolutional layers within these networks are extremely powerful as they allow the network to recognize sophisticated shapes such as handwritten characters. Convolutional neural networks are an extremely powerful tool that can be employed to create state-of-the-art applications for image processing, computer vision, and object detection. In this research 96+% classification accuracy was achieved, and this can only be improved with a more robust dataset, higher training times and increased computing power.

References

[1] IBM Cloud Education, Artificial Intelligence (AI). 3 June 2020.

[2] IBM Cloud Education, Machine Learning. 15 July 2020.

[3] Nils Jacob Sand, Introduction to Neural Network. Norwegian Creations. 5 April 2019.

[4] Al-Mahmud, A, Tanvin, S, Rahman. "Handwritten English Character and Digit Recognition." IEEE. 21 December 2021.

[5] P. Dhande, R, Kharat. "Recognition of Cursive English Handwritten Characters." IEEE. 22 February 2018.

[6] A, Mawaddah, C, Sari. "Handwriting Recognition of Hiragana Characters Using Convolutional Neural Network." IEEE. 26 October 2020.

[7] S, Kobayashi, "Noise Reduction of Segmented Images by Spatio-Temporal Morphological Operations. IEEE. 09 January 2020.

[8] N, Joram. "Morphological Operations in Image Processing." 1 January 2020.

[9] V, Naik. "Online handwritten Gujarati character recognition using SVM, MLP, and K-NN." IEEE. 14 December 2017.

[10] C, Nwankpa, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning." 8 November 2021.

[11] S, Ioffe. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." 2 March 2015.

[12] Pal, A. Singh, D. "Handwritten English Character Recognition Using Neural Network." *India International Journal of Computer Science & Communication.* December 2010.

[13] Herman. Et al. "Gaussian Naïve Bayes Classifier."

[14] C, Prateek. "Understanding Morphological Image Processing and Its Operations." *Towards Data Science.* 30 March 2022