Image recognition to classify European handwritten digits

1st Aya Fahmy 2nd Mark Azer 3rd Mobeen Iqbal 4th Tabea Menzel

***Abstract*—This paper tackles the problem of Image Recognition to classify European handwritten digits using state-of-art classification approaches in the machine learning field. The paper proposes a comparison of performance using 4 different algorithms: SVM, KNN, DNN and CNN. Regarding classical methods, KNN and SVM, the pre-processing pipeline included $Filtering$, $deskewing\ and\ centering$, $cropping\ and\ scaling\ of\ digits$, $binarization$ and $resizing$. Furthermore, PCA was implemented to reduce the training computational burden. This achieved test accuracies of 95.3% and 94.5% respectively. Regarding deep learning, DNN and CNN, $transfer\ learning$ was used through $fine\ tuning$. $Augmentation$ and $regularization$ were used to overcome the overfitting. Furthermore, the GPU was utilized to accelerate the training. This achieved test accuracies of 97.1% and 99.1% for the DNN and the CNN respectively.**

I. Introduction

Handwriting recognition is the ability of an algorithm to translate a handwriting input represented as an image or from other devices or even as an online input from a touch screen and then interpret it as text with meaningful values. Recently, there are various applications that incorporates the use of handwriting recognition such as smartphones, tablets and PDA to a touch screen through a stylus or finger. Thus, Handwritten Digit recognition is an active area of research in the domain of handwriting recognition.
In this paper, 4 different machine learning approaches for tackling the European Handwritten Digits classification problem is presented and compared.

II. Data Pre-processing

The pre-processing is a preliminary step which includes the entire range of steps necessary to bring the input data into a form acceptable for feature extraction. It involves getting character input, noise reduction, centering, deskweing and scaling of the input image. This is a very crucial process because feature extraction can be misled if the input images are not properly pre-processed. Fig.1. shows the images before the pre-processing while Fig.2. shows the same images after apply the pre-processing techniques discussed in this chapter. In the present work, the dataset used consisted of 6265 images of European handwritten digits. The image database was divided into two parts: a training part (4477 sample) and a test part (1788 sample). The former is used for building classifier models and training while the later is used for assessing the accuracy of the classification.

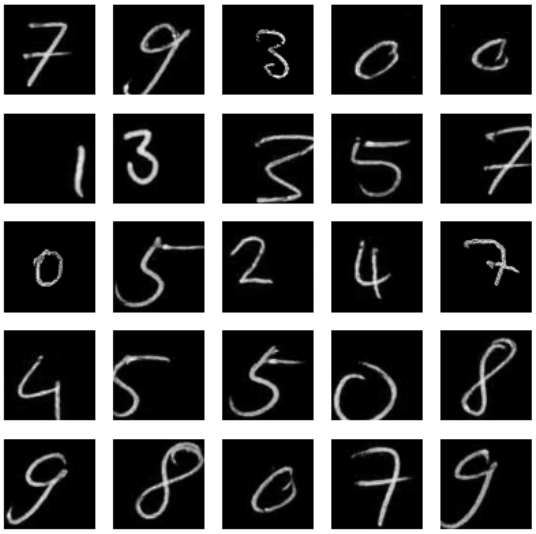

Fig. 1. Before applying Pre-processing

A. Filtering

the first step was converting the images into gray scale with pixel values between (0,255). A Gaussian Filter was applied to reduce the effect of noise (high frequency components). The filter is implemented as an Odd sized Symmetric Kernel (DIP version of a Matrix) which is passed through each pixel of the Region of Interest to get the desired effect.

B. Deskewing and Centering

As observed in Fig 1, the initial data was assumed to be rotated with different tilt angles and translated from the image origin. Hence, deskewing and centralization was implemented. The method allows adjusting the linear and angular displacement of the image based on the image moments. In particular, we model the process of deskewing as an affine transformation. were the offset,the translation part of the affine transformation (Centering), is calculated as the distance between the centroid and the center of the image where as the tilted angle,the rotational coefficient in the transformation (deskweing) , is estimated to be the ratio between the covariance and the variance.

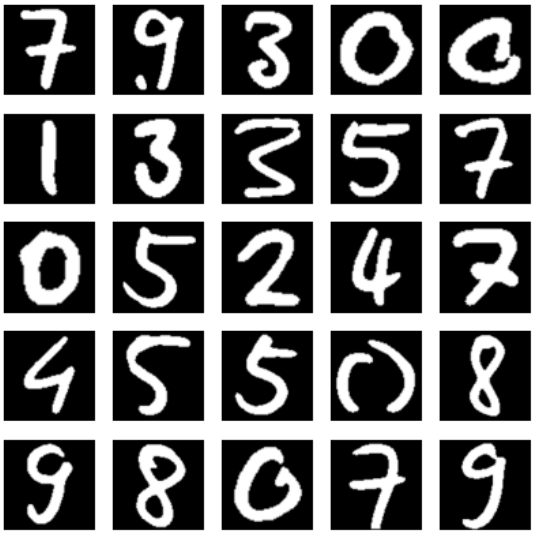

Fig. 2. After applying Pre-processing

C. Cropping and Scaling of digits

As observed in Fig 1. the digits within the image were inconsistent from the digit size perspective. Cropping was achieved by finding Intensity Gradient of the Image using Canny edge detection, followed by obtaining all the contours within the image, then taking a square bounding box with the largest side of the contour around the origin of all contours. By Taking the bounding box around the origin, this insured the centralization of the digit in the cropped image while taking the bound box as a square box insured the preservation of the digit format and avoided stretching of the digits within the image. Furthermore, padding was implemented on the images as a ratio of each image dimension in order to hold the constraint of anti-stretching and consistent digit size ratio within the image.

D. Binarization

The main task of the classifiers is the realization of the digits within the image. Classical methods like SVM and KNN realize the images as higher dimension vectors, combined with PCA to reduce the dimensionality of the dataset by extracting only the meaningful variance that relates directly to the features. Furthermore, applying binarization thresholding to the images eliminates the variance caused by gray scale pixel intensities which are irrelevant to the classification task. Hence, only the geometric variance is considered in the PCA.

E. Resizing and Normalization

Finally, the images were resized in order to reach a common dimension in the dataset. For the Deep Learning methods, pixel intensities were normalized.

III. Dimensionaity Reduction (PCA)

It is well known that the training of SVM and KNN is very slow compared to other classifiers. The training set usually has large size (thousands of samples). In this case, it is desirable to reduce the computational time of each training as well as the total number of iterations. PCA was implemented to pre-process the data and perform dimensionality reduction. The principle components are obtained by solving the eigen problem of the train dataset and performing a change of basis in the directions of maximum magnitudes.Thus, those less important components can be removed and thus the dimension of the space is safely reduced.

IV. Classical methods for Classification

A. Support Vector Machine (SVM)

SVM are a set of supervised learning methods used for classification, regression and outliers detection. Each element in the dataset is considered as a vector of n-dimension in an n- dimension space, where n is the number of features. In our case n is the number of pixels in an image or ,in case of implementing PCA, the number of the highest principle components chosen, with the value of each feature being value of a particular coordinate. The classification is done by finding the hyper-plane that separate the classes. In this approach, the dataset was deskwed, centralized, cropped, filtered and binarlized. The dimensions for resizing was taken with value of $110*110$. A data sample is represented in Fig.2. Then, PCA was implemeneted on the dataset and 66 Principle Components were chosen using Human based Optimization. The algorithm was implemented using scikit-learn. $SVC()$ Class was used to compute our non-linear multi-class classifier. Regarding Hyper-parameters tuning, A $GridSearchCV()$ method was used in order to get the optimal hyper parameters, which is obtained by cross-validated grid-search over a predefined parameter grid.

$$\begin{aligned} ParametersGrid = {}& 'C' : [0.1, 1, 10, 100, 1000], \\ & 'gamma' : [1, 0.1, 0.01, 0.001, 0.0001], \\ & 'kernel' : ['rbf', 'poly'] \end{aligned}$$

Where $'C'$ is the Regularization parameter, and the penalty is a squared L2 penalty. $'gamma'$ is the Kernel coefficient and the Kernels were $'rbf'$ is radial basis function (RBF) kernel and $'poly'$ is polynomial.
Initially, the SVM model was trained on the raw data before pre-processing and achieved a test accuracy of 70% where after performing the pre-processing the test accuracy reached 95.3%.

B. K-nearest neighbors algorithm

KNN classifier is the simplest and most straightforward image classification algorithm. In fact, KNN is not dependent on a learning process. The algorithm considers the distance between feature vectors. KNN algorithm classifies unknown data vectors by finding the most common class among the k nearest features vectors. In this approach, the dataset passed

under the same pre-processing pipeline as the SVM. Also, The dimensions for resizing was taken with value of $110 * 110$. PCA was implemeneted on the dataset and 66 Principle Components were chosen. by using human based optimization, the hyper-parameter k was selected with a value of $k = 9$. Initially, the KNN model was obtained on the raw data before pre-processing and achieved a test accuracy of 60% where after performing the pre-processing the test accuracy reached 94.5%.

V. Deep Learning

Deep learning is a multilayer neural network learning algorithm which advanced dramatically in recent years. It has brought a new wave to machine learning, and making artificial intelligence and human-computer interaction advance with big strides. We applied deep learning to The European handwritten digits recognition problem, and explored the two mainstream algorithm of deep learning: the Deep Neural NetWork (DNN) and the Convolutional Neural Network (CNN). The problem is closely related to MNIST database, which encourages the examination of state of the art classifiers for MNIST, and their performance on the European Handwriting Dataset.

A. Deep Neural Network

Deep neural networks (DNNs) are a widespread machine learning technique, which has shown state-of-the-art performance in many applications such as Image processing, computer vision and speech processing. A deep Neural Network is a multiple layer organized sequence of neurons by which the input for each layer is the neuron activations from the previous layer. The network implements a complex nonlinear mapping representing the relation between the input and the output. This mapping is learned from the data by adapting and tuning the weights of each neuron using error backpropagation. Which is based on minimizing the error loss function using an optimizter. The DNN model that reached 98% on MNIST was selected. The pretrained model was then applied on the European Handwritten Digits dataset, achieving a test accuracy of 88%. This suggests the close similarity of the two problems. Which paves the way for Transfer Learning.

1) Model:

- Augmentation layers (rotation and translation).
- Flatten layer.
- Fully Connected layer with 'reLU' activation.
- Dropout layer.
- Fully Connected layer with 'reLU' activation.
- Dropout layer.
- Fully connected layer with 'Softmax' activation.

An Augmentation layer was added to the model pipeline, where the values of the rotational and transnational factors are chosen randomly each epoch within a predefined range. This mitigates over-fitting. Followed by, a flattening layer that converts the images to 1D array of pixels. Then, a dense fully connected layer with 'reLU' activation. Dropout layer was later augmented to the model pipeline as a strategy that reduces co-adaptation amongst neurons by randomly deactivating some neurons in the network. Finally, a fully connected layer was implemeneted in the pipeline with 'Softmax' activation for classification.

2) Training: Employing transfer learning, the model was initially trained on MNIST train dataset for 200 epochs where the loss function was set to $thecategoricalcrossentropy$ and the optimizer used was $Adam$ with a learning rate of 0.01. Furthermore, a callback function was implemented in order to select the model with the highest validation accuracy with respect to our training dataset. The model achieved an accuracy of 90% on the European Handwritten Digits training dataset. Afterwards, a scheduler was implemented by which the model was trained on the European Handwritten Digits dataset for 200 epochs with a learning rate of 0.01, then the learning rate was reduced to 0.001 for 50 epochs and to 0.0001 for 50 epochs and finally to 0.00001 for 10 epochs. The model was selected based on the highest validation accuracy.

3) Results: The model result in a test accuracy of 97.1% whereas without the addition of the augmentation layer and the employment of transfer learning the test accuracy mounted to 90%.

B. Convolutional Neural Network

Convolutional neural network is a class of artificial neural network (ANN), most commonly used to process visual imagery. A basic convolutional neural network contains three components, namely, the convolutional layer, the pooling layer and the output layer. Convolutional neural networks (CNNs) have been successfully used to develop models for tackling a wide range of visual problems. Their ability to consider and code spacial information makes them the most suitable for this task. The CNN model that reached best results on MNIST was selected. The pretrained model was then tested on the European Handwritten Digits dataset, achieving a test accuracy of 95%. This suggests the close similarity of the two problems. In order to achieve better results, transfer leaning using Fine Tuning of the whole model with a small learning rate was applied achieving a test accuracy of 99.1%.

1) Model: The CNN model that reached best results on MNIST was selected. This model is a deep CNN network with squeeze and excitation blocks. The model was implemented using tensorflow package. The pipeline of the model consists of:

- Augmentation layers (rotation and translation).
- Convolutional layers with RelU activation.
- Batch Normalization layers.
- Squeeze-and-Excitation block.
- Average pooling layers.
- Maximum pooling layers.
- Fully connected layer with Softmax activation for classification.

The augmentation layer enlarges the dataset by applying different rotation and translation of the existing dataset and

therefore reducing the overfitting and increasing the flexibility of the model. The convolutional layer is the main building block of the CNN because it applies filters on the images to extract useful visual features and then applying a reLU activation function. Batch Normalization layer normalizes the output of the preceding layer. Squeeze-and-Excitation Networks (SENets) offers a building block for CNNs that utilizes channel interdependencies, this is done by adaptively weighing the importance of each channel. The Pooling layers are used to reduce the dimension of the feature map after convolution and activation. Finally, the Fully connected layer is the output layer that gives the probability of each class. This class has ten neurons corresponding to the number of classes and it uses softmax activation function which is suitable for multi-label classification.

2) Training: Given the similarity of the MNIST problem and the European Handwritten Digits problem, using transfer learning through fine tuning was the chosen methodology. First, The dataset was loaded and resized to the same size as MNIST of $28 * 28$. The pixel values was then normalized and shifted between -0.5 and 0.5 to match the preprocessing of MNIST dataset for this model. The model was then compiled with ADAM optimizer with a learning rate of 0.00001 and Categorical cross-entropy loss function. This used learning rate is chosen lower than the lowest learning rate that was used for training the MNIST model in order to apply Fine Tuning without changing much in the pretrained model. The model was then trained with 200 epoch on a NVIDIA GeForce GTX 1050 Ti which accelerates the training by a factor of 6. The time taken for training on the GPU is 66 minutes while it takes 400 minutes on Intel Core i5-9600 CPU.

3) Results: The pretrained model was tested on the European Handwritten Digits dataset, achieving a test accuracy of 95%. This suggests the close similarity of the two problems. In order to achieve better results, transfer leaning using Fine Tuning of the whole model with a small learning rate was applied achieving a test accuracy of 99.1%. The model was also trained with the European Handwritten Digits dataset without the use of MNIST dataset to study the effect of the transfer learning. The model reached a maximum test accuracy of 97%. This shows the benefits of transfer learning.

Figure 3 shows all the misclassified images from the test set.

TABLE I
COMPARISON BETWEEN ALL MODELS IN TERMS OF VALIDATION ACCURACY.

Model	Without pre-processing	With pre-processing	Without Augmentation & Transfer Learning	With Augmentation & Transfer Learning
SVM	70.0%	95.3%	-	-
KNN	60.0%	94.5%	-	-
DNN	-	-	90%	97.1%
CNN	-	-	95%	99.1%

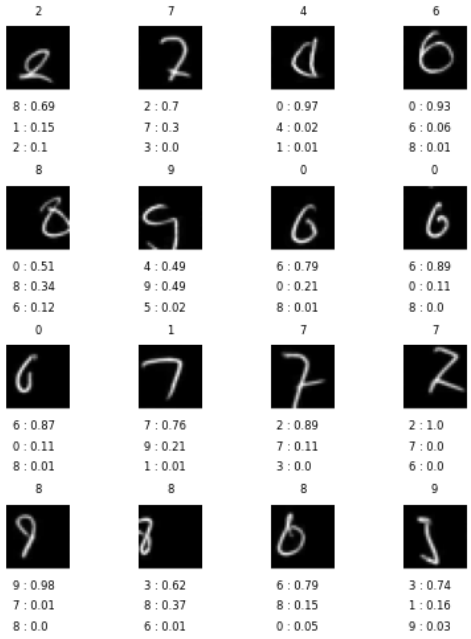

Fig. 3. Misclassified images with top-3 predictions.

ACKNOWLEDGEMENT

Section (II). Data Pre-Processing and Section (V.A). Deep Neural Network was done by Aya Fahmy. *Section (V.B). Convolutional Neural Network* was done my Mark Azer. *Section III. Dimensionality Reduction (PCA) and Section IV.A. Support Vector Machine (SVM)* was done by Tabea Menzel. *Section III. Dimensionality Reduction (PCA) and Section IV.B. K-nearest neighbors algorithm* was done by Mobeen Iqbal.

REFERENCES

[1] Belyaev, M. A., Velichko, A. A. (2020, May). Classification of handwritten digits using the Hopfield network. In IOP Conference Series: Materials Science and Engineering (Vol. 862, No. 5, p. 052048). IOP Publishing.

[2] Lei, H., Govindaraju, V. (2005). Speeding up multi-class SVM evaluation by PCA and feature selection. Feature Selection for Data Mining, 72.

[3] Hamid, N. A., Sjarif, N. N. A. (2017). Handwritten recognition using SVM, KNN and neural network. arXiv preprint arXiv:1702.00723.

[4] Palvanov, A., Im Cho, Y. (2018). Comparisons of deep learning algorithms for MNIST in real-time environment. International Journal of Fuzzy Logic and Intelligent Systems, 18(2), 126-134.

[5] Wu, M., Chen, L. (2015, November). Image recognition based on deep learning. In 2015 Chinese Automation Congress (CAC) (pp. 542-546). IEEE.

[6] Jakubovitz, D., Giryes, R. (2018). Improving dnn robustness to adversarial attacks using jacobian regularization. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 514-529).

[7] Hu, J., Shen, L., Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).