Handwriting recognition approach by using 2 different methods: Support Vector Machine and Convolutional Neural Network

Abstract

*Digital technologies are one of the most important and dispensable factors in developing fields of all sectors. For that reason, handwriting character recognition has been widely applied and become a popular research subject over the last few decades. USPS and MNIST are the two available datasets used for evaluating the effectiveness of handwriting recognition methods, with USPS containing a training set of 7791 examples, a test set of 2007 examples and the images are 16*16 grayscale pixels. The MNIST dataset has 60.000 train and 10.000 test examples. Each example is an image with a size of 28*28 pixels. Through my research, I discovered a total of algorithms used for handwriting recognition are neural network, CNN (Convolutional neural network) with accuracy up to 99% and a more traditional algorithm with lower quality performance but having the advantages of simplicity and less computation cost by using HOG (histogram of oriented gradient) as an input for SVM (support vector machine) by extracting from handwriting sample. The intention is to demonstrate the methodology, design, and architecture of the hand writing recognition system and test the outcomes of the system development.*

Keywords: Handwriting recognition, CNN, HOG analysis, SVM classifier, MNIST database, USPS database, Neural network

1. Introduction

The use of artificial intelligence is pattern recognition can be seen everywhere in this modern day and age. From simple things like auto fixing a provided image to make the subject of the image look better, to creating an entirely new artwork inspired from hundreds of thousands of sources. But one application people often overlook when talking about pattern recognition is handwriting recognition. As simple as it may sound, it provides a lot of value in many ways, such as scanning or helping blind people read hand-written documents. Since each person has their own unique handwriting style, hardcore programs are, without a shadow of doubt, very useless and this is where artificial and machine learning come into place. In this project, I will discuss and explain some of the most prominent techniques including convolutional neural network, histogram of oriented gradient and support vector machine to output a high accuracy performance.

The motivation of the project is to create a handwriting recognition model that allows users to quickly record and take notes compared to entering the same information through the virtual keyboard. This is because most people are more comfortable with writing and doing it quickly. Also, to help quickly turn information from books into digital data for longer storage, and at the same time saves time and manpower by eliminating manual information transfer like typing words by word from books to computer using keyboard.

2. Related works

Today I am going to look into methods of handwriting recognition. By my research, I've found out that there are 2 main ways: CNN; HOG & SVM. Now I'm going to dive into CNN and its related works.

The first work is the work of Arik Poznanski and Lior Wolf [1]. Their method prioritizes common attributes between words which named Pyramidal Histogram of Characters (PHOC) [2]. The character set is

various including Latin, Arabic, etc… They defined the attributes to 5 levels unigram attribute, splits into group of unigrams, bigram and trigrams, also they used the bijective mapping to map a given attribute vector by its word. Their CNN is a VGG [3] style network, which is a deep network structure of 12 layers to generate attribute vectors. The CNN had allowed them to process raw pixels, lead to shorter processing time. Training phase is done in stages, starting with one single group of attributes and added another when the loss stabilizes. Only lower the learning rate when all 19 groups had been added. This method had found to be superior in result over training all groups at once. In order to avoid overfitting, dropout [4] is applied, data augmentation is warranted based on the small handwriting datasets by rotating the images in destinated angles. To summary, they had used deep CNNs to extract n-grams which feed Canonical Correlation Analysis (CCA) for final word recognition with a rather small and fixed vocabulary. Their method achieves state of the art results on all benchmarks: IAM [5], RIMES [6] and IFN/ENIT [7], which contain images of handwritten English, French and Arabic, respectively. They have obtained a sizable improvement over all commonly used handwriting recognition benchmarks, halving, in almost all cases, the best reported error rate. The error rates are cut in half throughout the datasets: IAM (6.45% vs. 11.9%), RIMES (3.9% vs. 8.9% for a single recognizer), IFN/ENIT set-f (3.24% vs. 6.63%) and set-s (5.91% vs. 11.5%) compared with previous work. Comparison between the accuracy on the test set of the synthetic data: they obtained 96.88% compared to 95.2% obtained by the best network of [8]. In conclusion, an outstanding improvement was shown in the method, proving CNNs is one of the best ways to reach promising result.

In this work is by Aditya Chilangia G. Malathi extract research [9], they are using Histogram of oriented gradient (HOG) to extract the features from the handwriting sample from the writer serve as a input for support vector machine(SVM). Instead of using 100% given dataset like the MNIST or USPS, they collect handwriting from 50 different people. They

are using the Polynomial kernel to predict the personality trait of a person with 80% accuracy to predict and create a letter font based on that person's personality. Based on their research, there are two methods that are addressed on the same dataset to measure and compare their system performance. Based on their report, a limitation that can be noticed is that collecting actual data from people would consume more time and manpower but also result in a more reliable performance.

Another related work is the work by Almazan et al [10]. But instead of CNN, their method encodes the input word image as Fisher Vectors (FV). In my opinion, their work is not as optimized because they not using CNNs which inputs can be unprocessed data such as raw pixels which would save time compare to extracting specific features or pen stroke grid values. They have to use one classifier per attribute, resulted in longer processing time. As a result, their method is currently not among the state of the art methods in handwriting recognition, on any of the datasets.

3. Data Preparation

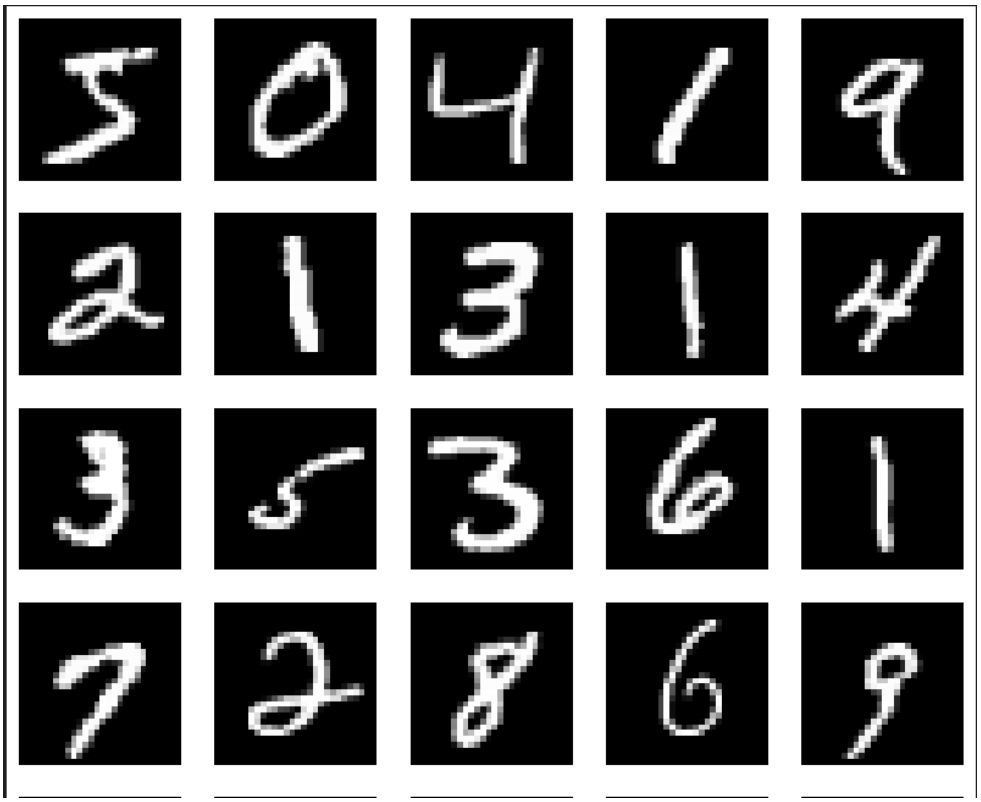
Datasets play a crucial role in Artificial Intelligence. It is important to fetch and prepare data because models will always use data as input for the training process. Without data, process of training, research and automation will be deemed useless.

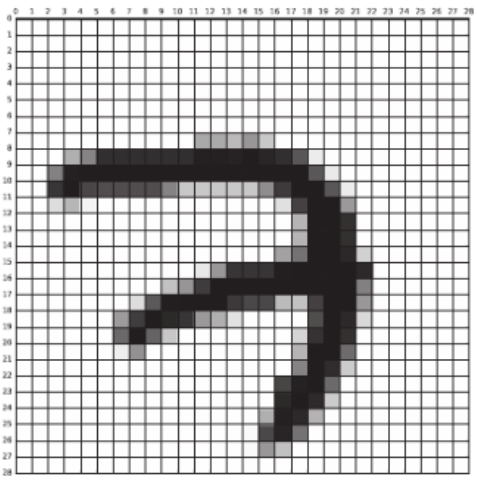
The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments.

The USPS dataset was made by automatically scanning envelopes by the U.S. Postal Service.

Dataset	Size	#class	#training	#test	#total
MNIST	28 x 28	10	60000	10000	70000
USPS	16 x 16	10	7291	2007	9298

Image for MNIST Dataset

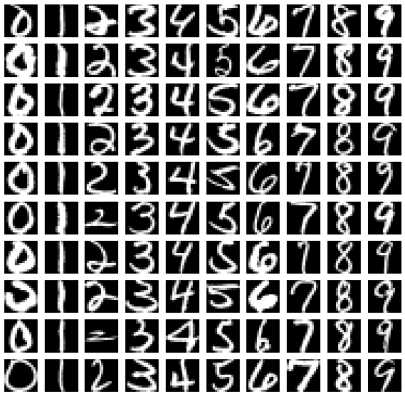




(a) MNIST sample belonging to the digit '7'.

(b) 100 samples from the MNIST training set.

Image for USPS Dataset



Because MNIST dataset is more popular and useful so i decided to use this dataset.

Exploratory Data Analysis

- Data Dimension: 60,000 small square 28×28 pixel
- Data Type: int64

```
mnist.summary()
```

```
_________________________________________________________________
MNIST:
Patterns     Shape                  Range
=================================================================
inputs       (28, 28, 1)            (0.0, 1.0)
targets      (10,)                  (0.0, 1.0)
=================================================================
Total patterns: 70000
   Training patterns: 70000
   Testing patterns: 0
_________________________________________________________________
```

Because this dataset is already preprocessing so I will show the way to load and plot the dataset:

- Loading:

Input:

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
print('X_train: ' + str(X_train.shape))
print('Y_train: ' + str(y_train.shape))
print('X_test:  ' + str(X_test.shape))
print('Y_test:  ' + str(y_test.shape))
```

✓ 0.6s

Output:

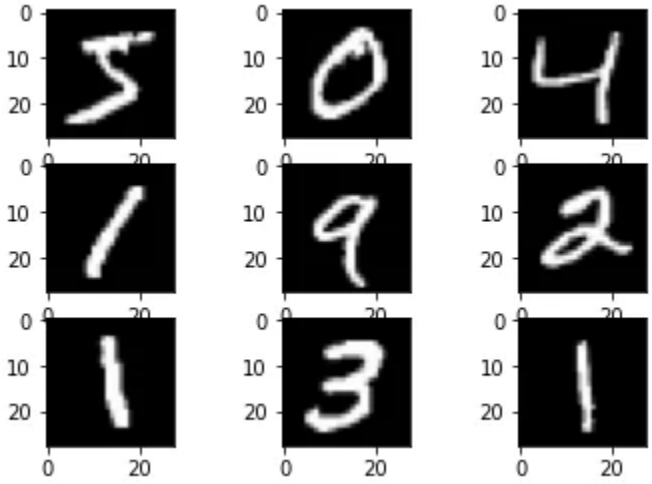
```
X_train: (60000, 28, 28)
Y_train: (60000,)
X_test:  (10000, 28, 28)
Y_test:  (10000,)
```

- Plotting:

Input:

```
for i in range(9):
    plt.subplot(330 + 1 + i)
    plt.imshow(X_train[i], cmap=plt.get_cmap('gray'))
    plt.show()
```
✓ 2.3s

Output:



4. Method

4.1. HOG and SVM

4.1.1, HOG

The Histogram of Oriented Gradient (HOG)[11] is a powerful method used in image processing. HOG was first introduced by N. Dalal and B. Triggs for detecting the human body, but it has since become widely popular and successful for object detection. The HOG method analyzes the distribution of gradient orientations in a localized portion of an image. The idea behind HOG descriptors is that local object appearances and shapes within an image can be described by edge directions or the distribution of intensity gradients.

To extract HOG features from a digital handwriting sample, the sample is first divided into square grids. The edge or histogram of gradient direction is then computed based on central difference. This calculates the

orientation of the edges in the image. The HOG features are calculated by taking orientation histograms of edge intensity in a local region[12]. The histogram of the orientation of gradients is used to represent the local appearance and shape of the object. In summary, the HOG technique is a powerful method used in image processing and computer vision for object detection. Its ability to capture the shape and texture of an object in an image, its robustness to changes in illumination and contrast, and its success in digit recognition make it a widely used and successful technique in the field.

4.1.2, SVM

In the context of digit recognition from digital handwriting samples, the Support Vector Machine (SVM)[13] has been used as the classifier. SVM is known for its better accuracy and time efficiency compared to neural networks in this scenario. In a multiclass SVM, different labels are assigned to instances using SVM to distinguish between different classes. SVM works by taking the training data, x, from a space R^D, where $x=\{x^1, x^2, x^3, ...x^n\}$ and x belongs to R^D. Their class labels, $y = \{y^1, y^2, y^3, ...y^n\}$, are given to indicate which class x^i belongs to.

In this study, the polynomial kernel has been utilized as it has been found to yield better results than nonlinear kernels such as sigmoid and RBF on non-linear features. The polynomial kernel maps the input space to a higher dimension in a nonlinear fashion. It is outlined as equation (1):

$$K(x,y) = (xTy + c)^d, \text{ where:}$$

x and y are vectors of features extracted from the training digital handwriting samples in the input space, and $c \geq 0$ is a trading-off parameter.

In the kernel, K is the inner product in a feature space based on some mapping Φ, as given by equation (2):

$$K(x,y) =< \Phi(x),\Phi(y) >.$$

The classification method follows 3 steps:
1. Creation of the vectors from input features.
2. Applying the Polynomial kernel to map the feature space into a higher dimension.
3. Computation of a hyper-plane which separates the feature space to different classes of sample vector.

Overall, the SVM technique using the polynomial kernel has shown promising results in recognizing digits from digital handwriting samples. The ability to map the input space to a higher dimension using the polynomial kernel has been found to be particularly useful in separating different classes of handwritten digits. The combination of SVM and polynomial kernel has become a popular approach for digit recognition and has been used successfully in various studies.

4.2. Convolutional Neural Networks

In this section, I will present a brief overview of CNN and some of the most commonly used layers of this network for image processing.

4.2.1, Convolution

Convolution was first used in digital signal processing. Thanks to the principle of information transformation, scientists have applied this technique to digital image and video processing. For ease of visualization, I can consider convolution as a sliding window imposed on a matrix. Figure 1 illustrates the mechanism of convolution.

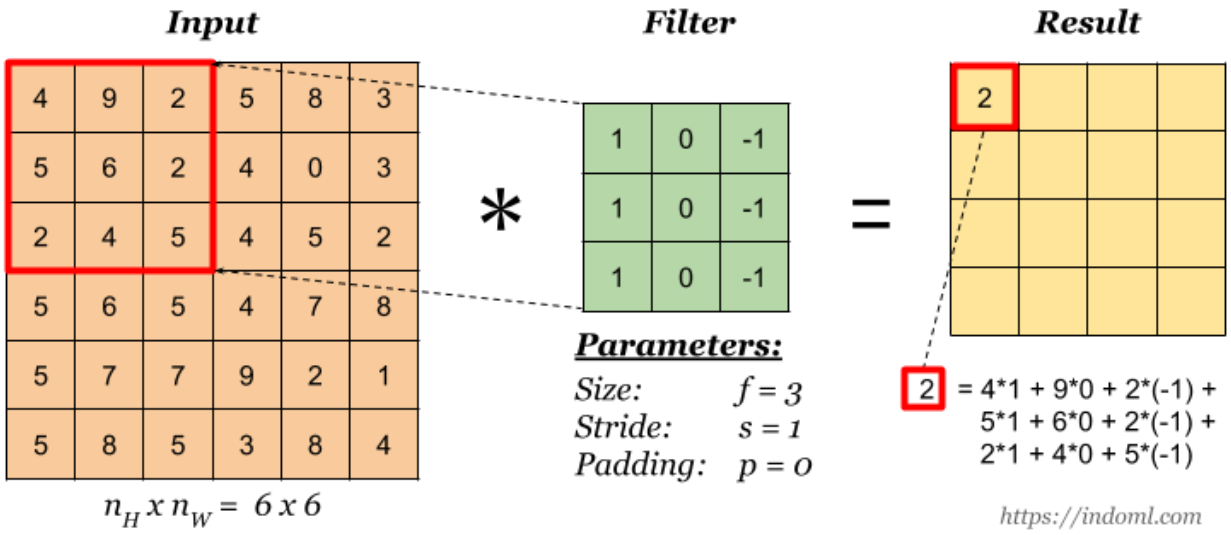


Figure 1. Illustration of convolution

The left matrix is a grayscale image, each value of the matrix is equivalent to a pixel (pixel) with a value ranging from 0 to 255. Sliding window also has called kernel, filter or feature detector. Here, I use a filter matrix of size 3 × 3 multiply each element corresponding to the image matrix on the left. The output value is the sum of the products of these components. The result of the convolution is a matrix generated by sliding the filter matrix and convolving at the same time over the entire left image matrix.

4.2.2, Convolutional Neural Network Model

The CNN model simply consists of a few layers of convolution combined with nonlinear activation functions such as ReLU or tanh to produce information at a more abstract level for subsequent layers. In the feedforward neural network model, the layers are directly connected to each other through a weighted vector. These layers are also called fully connected layers or affine layers. In the CNN model, the opposite is true. The layers are linked together through the convolution mechanism. The next layer is the convolution result from the previous layer, so we get 110 local connections. That is, each neuron in the next layer generated from the filter is applied to a local image area of the previous layer neuron. Each such layer is applied different filters, usually a few hundred to several thousand such filters. Some other layers such as the

pooling/subsampling layer are used to filter out more useful information (remove noise information).

During the training process, the CNN will automatically learn the parameters for the filters. For example, in the image classification task as illustrated in Figure 2, CNN will try to find the optimal parameters for the corresponding filters in the order of raw pixels > edges > shapes > facial > high-level features. The last layer is used to layer the image.

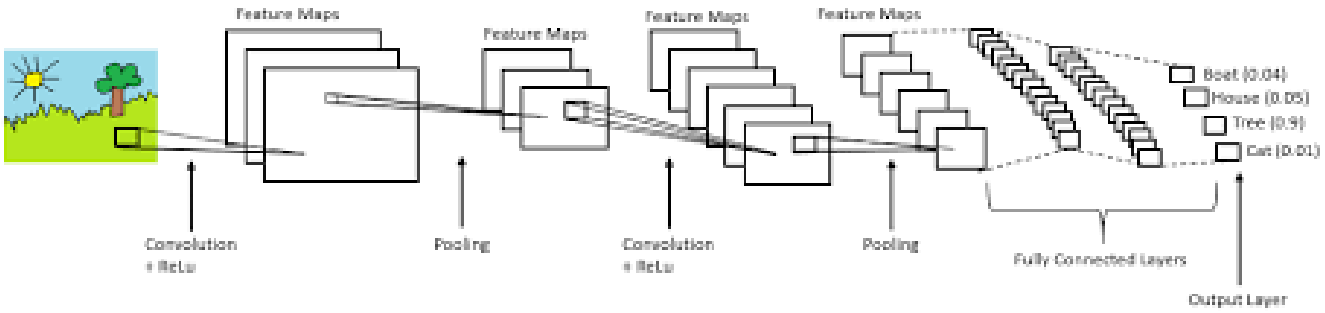


Figure 2. Illustration of CNN architecture used in image classification

CNN has invariance and locality (Location Invariance and Compositionality). With the same object, if this object is projected from different angles (translation, rotation, scaling), the accuracy of the algorithm will be significantly affected. Pooling layer will give invariant to translation (translation), rotation (rotation) and scaling (scaling). Local associativity gives low-to-high and more abstract levels of information representation through convolution from filters. That's why CNN produces a model with very high accuracy. Next, I will detail the classes in the model.

Convolutional Layers

This layer is where the original idea of CNN is expressed. Instead of connecting all the pixels, this layer will use a set of filters that are small in size compared to the image (usually 5×5 or 3×3) applied to an area in the image and perform a convolution between the filter and the image.

pixel value in that local area. The filter will in turn be shifted by a stride value running along the image and scanning the entire image.

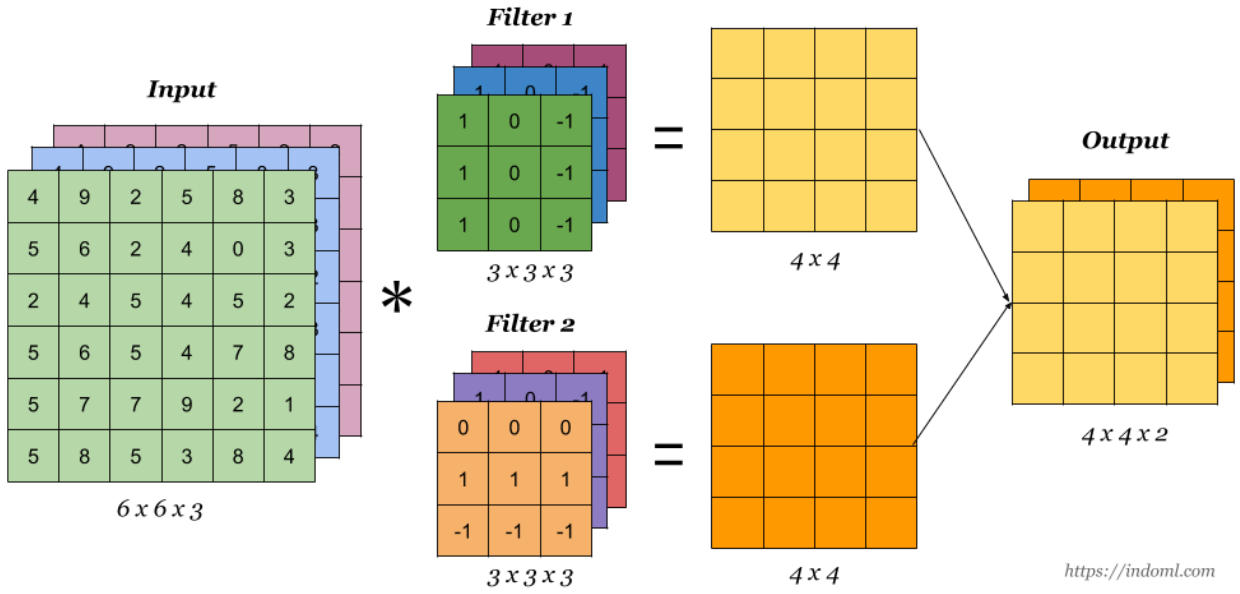


Figure 3. Convolution with filters

With a 32×32 image and a 3×3 filter, we will have a new image of size 32×32 (provided that padding has been added to the original image to compute the convolution for the filter scan cases). edges) is the result of the convolution of the filter and the image. With how many filters in this layer, we will have as many corresponding images that this layer returns and is passed on to the next layer. The initial filter weights will be randomly initialized and will be gradually learned during the training of the model. Figure 3 illustrates a convolution calculation with a 3×3 filter.

Rectified Linear Unit (ReLU) Layer

This layer is usually installed right after the Convolution layer. This layer uses the activation function f(x) = max(0, x). In simple words, this layer

is responsible for converting all negative values in the result taken from the Convolution layer to the value 0. The meaning of this setting is to create nonlinearity for the model. Similar to the feed-forward network, building on linear transformations makes building multilayer multilayers pointless. There are many ways to make the model nonlinear such as using activation functions sigmoid, tanh, ... but the function f(x) = max(0, x) is easy to install, fast to calculate and still effective.

Pooling Layer

This layer uses a sliding window that scans through the entire data image, each time following a given slide. Unlike layerConvolution, layer Pooling does not compute convolution, but conducts sampling (subsampling). When the window slides over the image, only one value that is considered to be the value representing the image information in that region (the sample value) is retained[14]. The common retrieving methods in the Pooling layer are MaxPooling (get the maximum value), MinPooling (get the maximum value). minimum) and AveragePooling (take the average).

Considering an image of size 32×32 and the Pooling layer using a filter of size 2×2 with a stride of 2, the method used is MaxPooling. The filter will slide through the image in turn, with each slide only the largest of the 4 values within the 2×2 window area of the filter are retained and fed into the output matrix. Thus, after going through the Pooling layer, the image will be reduced in size to 16 × 16 (size of each dimension is reduced by 2 times).

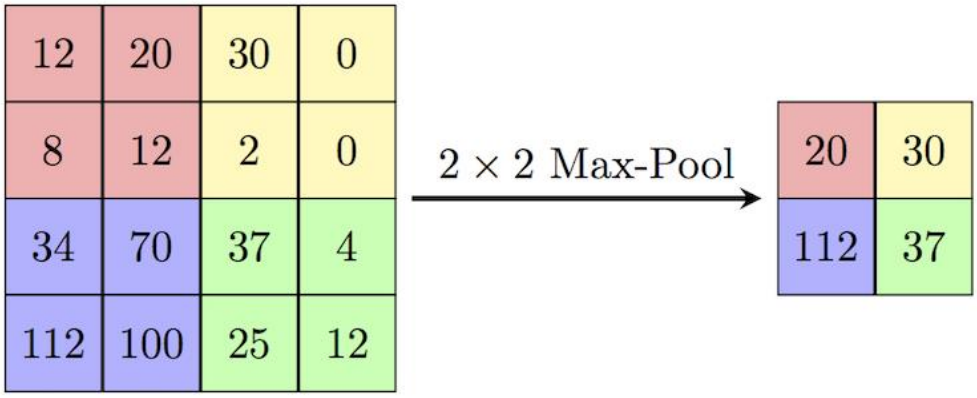


Figure 4. Calculation with MaxPooling method

Pooling Layer has the role of reducing the data size. With a large image through many Pooling Layers, it will be reduced but still retain the features needed for identification (through sampling). Reducing the data size will reduce the number of parameters, increase the computational efficiency, and contribute to the control of overfitting.

Fully Connected (FC) Layer

This layer is similar to the layer in a feedforward neural network, the image values are fully linked to the neurons in the next layer. After the image is processed and features extracted from the previous layers, the image data will no longer be too large compared to the straight-forward model, so we can use the straight-forward model to conduct the recognition.

4.2.3, Operation of the CNN model

The CNN model is formed by connecting the above layers together. The model starts with a Convolutional Layer. The ReLU Layer is usually installed right after the Convolutional Layer or even combines the two layers into one layer. The next layers can be Convolutional or Pooling depending on the architecture we want to build. Finally will be Fully-Connected Layer to conduct layering.

5. Result

5.1: Evaluate the model

- To evaluate the model, I used 5 different metrics: Accuracy, Precision, Recall, F1-score and Cross Entropy loss

5.1.a: Accuracy

- It measures the percentage of correctly classified instances out of the total number of instances. It is defined as TP + TN / (TP + TN + FP + FN), where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

5.1.b: Precision

- It measures the percentage of correctly classified positive instances out of the total number of instances that are predicted as positive. It is defined as TP / (TP + FP), where TP is the number of true positives and FP is the number of false positives.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

5.1.c: Recall

It measures the percentage of correctly classified positive instances out of the total number of actual positive instances. It is defined as TP / (TP + FN), where TP is the number of true positives and FN is the number of false negatives.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

5.1.d: F1-score

It is the harmonic mean of precision and recall, and is a measure of the balance between precision and recall.

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.1.e: Cross Entropy loss

Cross-entropy loss, also known as log loss, is a commonly used loss function in machine learning and deep learning. It is often used in classification tasks, where the goal is to predict the probability distribution of a set of classes.

$$L = -\frac{1}{m} \sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

-These metrics are used to evaluate the performance of machine learning models and to compare different models or parameter settings. Depending on the problem and the context, some metrics may be more important than others. For example, in a medical diagnosis task, recall may be more

important than precision, as false negatives (i.e., missed diagnoses) can have serious consequences.

5.2: Result

5.2.a: The result of HOG and SVM

5.2.a.1: Accuracy and Cross Entropy loss

- Figure A shows us the accuracy and the Cross Entropy loss of the dataset. We can see that the accuracy is quite high, around 0.965. And the Cross Entropy loss also quite low, around 0.203.

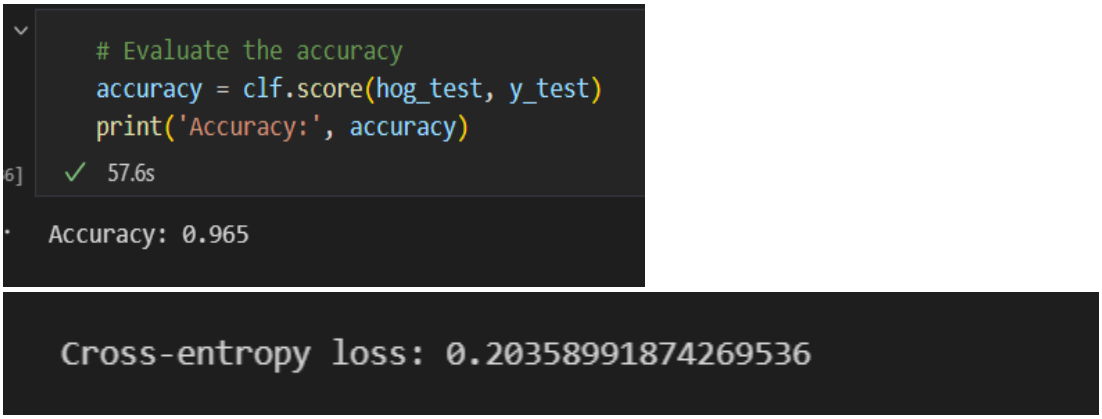


Figure A : Accuracy and Cross Entropy loss

5.2.a.2 Precision, Recall, and F1 score

- Figure B shows us the Precision, Recall, and F1 score of the dataset. We can see that the Precision, Recall, and F1 score are quite high, around 0.96.

	precision	recall	f1-score
0	0.97	0.99	0.98
1	0.99	0.99	0.99
2	0.96	0.97	0.96
3	0.95	0.97	0.96
4	0.97	0.96	0.96
5	0.97	0.96	0.97
6	0.98	0.97	0.98
7	0.96	0.95	0.95
8	0.96	0.95	0.95
9	0.95	0.94	0.95

Figure B: Precision, Recall, and F1 score

⇨ Comment: In conclusion, handwriting recognition using HOG and SVM is another powerful application of computer vision and machine learning. This project involves extracting features using HOG (Histogram of Oriented Gradients) and training an SVM (Support Vector Machine) classifier on those features.
The accuracy of the model depends on the quality of the features extracted by HOG, the choice of SVM kernel, and the parameters of the SVM classifier such as the regularization parameter and the kernel width. With careful tuning of these parameters and using a large and diverse dataset of handwritten images, the HOG-SVM approach can achieve high levels of accuracy in handwriting recognition

5.2.b: The result of Convolutional Neural Networks

5.2.b.1: Accuracy and Cross Entropy loss

- Figure C shows us the accuracy and the loss of the dataset. We can see that the accuracy is very high, 0.9912. And the loss also considering low, around 0.029.

```
313/313 [==============================] - 2s 8ms/step
Test loss: 0.0290
Test accuracy: 0.9913
```

Figure C : Accuracy and Cross Entropy loss

5.2.b.2 Precision, Recall, and F1 score

- Figure D shows us the Precision, Recall, and F1 score of the dataset. We can see that the Precision, Recall, and F1 score are markedly high, around 0.99.

```
313/313 [==============================] - 3s 9ms/step
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       980
           1       0.99      1.00      0.99      1135
           2       0.99      0.99      0.99      1032
           3       0.99      0.99      0.99      1010
           4       0.98      1.00      0.99       982
           5       0.99      0.99      0.99       892
           6       1.00      0.99      0.99       958
           7       0.99      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       0.99      0.98      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000

Cross-entropy loss: 0.028979100359549637
```

Figure D: Precision, Recall, and F1 score

⇨ Comment: In conclusion, handwriting recognition using CNN is a powerful application of computer vision and artificial intelligence.

The project involves gathering a dataset of handwritten images, preprocessing the data, defining a CNN architecture, training the model on the dataset, and evaluating its performance on a test set.

The accuracy of the model depends on the quality and size of the dataset, the complexity of the CNN architecture, and the training parameters such as the batch size and number of epochs. With enough data and careful tuning of the model, handwriting recognition using CNNs can achieve high levels of accuracy and be useful in a variety of applications, such as digit recognition for banking, signature verification, and OCR (Optical Character Recognition).

6. Conclusion

- In short, our project was successful in handwriting recognition, using two main methods are “Support Vector Machines (SVMs)” and “Convolutional Neural Networks (CNNs)”.

- Comparing the two methods, we can draw the comment that:

+ The HOG-SVM approach requires less computational resources and is easier to interpret and understand.

+ However, CNNs is more effective when dealing with highly complex images or when the amount of data available is limited.

But after all, we feel quite satisfied with the project and look forward to being able to develop this project together as well as challenge other projects in the future.

REFERENCES

[1]: A. Poznanski and L. Wolf, "CNN-N-Gram for HandwritingWord Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 2305-2314, doi: 10.1109/CVPR.2016.253.

[2]: J. Almazan, A. Gordo, A. Fornes, and E. Valveny. Word spotting and recognition with embedded attributes. IEEE Transactions on Pattern Analysis & Machine Intelligence, (12):2552–2566, 2014.

[3]: K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[4]: N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[5]: U.-V. Marti and H. Bunke. The iam-database: an english sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition, 5(1):39–46, 2002.

[6]: E. Augustin, M. Carre, E. Grosicki, J.-M. Brodin, E. Geof- ´ frois, and F. Preteux. Rimes evaluation campaign for hand- ˆ written mail processing. In Proceedings of the Workshop on Frontiers in Handwriting Recognition, number 1, 2006.

[7]: M. Pechwitz, S. S. Maddouri, V. Margner, N. Ellouze, ¨ H. Amiri, et al. Ifn/enit-database of handwritten arabic words. Citeseer.

[8]: M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. arXiv preprint arXiv:1406.2227, 2014.

[9]: Aditya Chitlangia, G. Malathi, Handwriting Analysis based on Histogram of Oriented Gradient for Predicting Personality traits using SVM, Procedia Computer Science, Volume 165,2019, Pages 384-390, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.01.034.

[10]: J. Almazan, A. Gordo, A. Fornes, and E. Valveny. Word spotting and recognition with embedded attributes. IEEE Transactions on Pattern Analysis & Machine Intelligence, (12):2552–2566, 2014.

[11]: N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection, CVPR, (2005)

[12]: T. Kobayashi, A. Hidaka and T. Kurita, Selection of Histograms of Oriented Gradients Features for Pedestrian Detection, (2008), pp.598607

[13]: Prasad, Shitala, et al. Handwriting Analysis Based on Segmentation Method for Prediction of Human Personality Using Support Vector Machine. International Journal of Computer Applications, vol. 8, no. 12, 2010, pp. 2529., doi:10.5120/1256-1758.

[14]: Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J and Chen T 2018 Patt. Recognition 77 354