mathematics

MDPI

Article

Self-Writer: Clusterable Embedding Based Self-Supervised Writer Recognition from Unlabeled Data

Zabir Mohammad [1], Muhammad Mohsin Kabir [1], Muhammad Mostafa Monowar [2,*], Md Abdul Hamid [2] and Muhammad Firoz Mridha [3]

[1] Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka 1216, Bangladesh
[2] Department of Information Technology, Faculty of Computing and Information Technology, King AbdulAziz University, Jeddah 21589, Saudi Arabia
[3] Department of Computer Science, American International University-Bangladesh (AIUB), Dhaka 1229, Bangladesh
* Correspondence: mmonowar@kau.edu.sa

Abstract: Writer recognition based on a small amount of handwritten text is one of the most challenging deep learning problems because of the implicit characteristics of handwriting styles. In a deep convolutional neural network, writer recognition based on supervised learning has shown great success. These supervised methods typically require a lot of annotated data. However, collecting annotated data is expensive. Although unsupervised writer recognition methods may address data annotation issues significantly, they often fail to capture sufficient feature relationships and usually perform less efficiently than supervised learning methods. Self-supervised learning may solve the unlabeled dataset issue and train the unsupervised datasets in a supervised manner. This paper introduces Self-Writer, a self-supervised writer recognition approach dealing with unlabeled data. The proposed scheme generates clusterable embeddings from a small fixed-length image frame such as a text block. The training strategy presumes that a small image frame of handwritten text should include the writer's handwriting characteristics. We construct pairwise constraints and nongenerative augmentation to train Siamese architecture to generate embeddings depending on such an assumption. Self-Writer is evaluated on the two most widely used datasets, IAM and CVL, on pairwise and triplet architecture. We find Self-Writer to be convincing in achieving satisfactory performance using pairwise architectures.

Keywords: writer recognition; self-supervised learning; embeddings; dimension reduction; clustering; twin network

MSC: 68TXX

Citation: Mohammad, Z.; Kabir, M.M.; Monowar, M.M.; Hamid, M.A.; Mridha, M.F. Self-Writer: Clusterable Embedding Based Self-Supervised Writer Recognition from Unlabeled Data. *Mathematics* **2022**, *10*, 4796. https://doi.org/10.3390/math10244796

Academic Editor: Tao Zhou

Received: 8 November 2022
Accepted: 11 December 2022
Published: 16 December 2022

1. Introduction

Handwriting is considered a distinctive human characteristic that can prove someone's authenticity through pattern recognition. Handwriting contains numerous distinctive features that exhibit the writer's unique handwriting characteristics, such as the slope of letters, shape of letters, rhythmic repetition of the letters, cursive or separated writing, spacing between letters, etc. [1]. Furthermore, handwriting techniques and features differ enormously from one individual to another, known as inter-class variance. The unique writing characteristics of an individual serve to make handwriting a behavioral biometric modality that authorizes recognition and verification of writers from handwritten scripts. The contemporary studies have indicated writing to be a remarkably reliable and helpful behavioral biometric mechanism that is used in diverse application disciplines, including forensic analysis [2], analysis of historical documents [3,4] and security [5].

There are two modes to implement writer identification: verification and recognition. The writer verification system performs a one-to-one comparison and determines whether

the same person has written two different texts or not. At the same time, the writer recognition system performs a one-to-many search with handwriting data of known authors in an extensive database. The system should display a list of possible authors for the unknown text samples following the comparison. Due to the enormous variety of human handwriting, writer recognition is more complicated than writer verification.

Furthermore, these two modes can be executed both online and offline. The online technique uses the spatial characteristics of the writing, which are taken in real time by using digitizing acquisition equipment (e.g., Anoto pen). These characteristics are sent for further processing and analysis via a particular transducer device. Then, the processing device converts dynamic writing movement characteristics such as stroke order, altitude, velocity, trajectory, pen pressure, writing duration, etc., into a signal sequence. Offline-based recognition, however, is a static technique that commonly uses digitized handwritten images as input data. Because online techniques utilize a good number of features, it is likely to perform better than the offline approach. However, online recognition methods require additional devices that are costly and unavailable in most scenarios. This triggers us to exploit the offline recognition approach, knowing that it poses significant research challenges due to the availability of only digitized handwritten text images.

Deep learning (DL) frameworks have been intensively explored in supervised writer recognition and have been shown to outperform several benchmark datasets [1,6,7]. However, supervised writer recognition methods require a significant amount of labeled data. Additionally, obtaining manual labeling is costly compared to obtaining unlabeled data, which is readily available in abundance. Unsupervised writer recognition may solve the data annotation label issue. So far, unsupervised algorithms are not particularly effective at training neural networks because of their inability to capture the visual semantics needed to tackle real-world problems the way strongly supervised methods do. However, self-supervised learning may convincingly address the unlabeled dataset issue by training the unsupervised dataset in a supervised manner.

Self-supervised learning is a variant of the unsupervised learning method wherein the supervised task is performed from the unlabelled data. To learn from self-supervision, the technique must go through two stages: initialization of the network weights using pseudolabels [8,9], and completion of the actual task by using supervised learning [10,11]. Self-supervised learning allows us to take advantage of a range of labels provided for free with the data. Producing a handwritten document dataset with clean labels is costly. In addition, unlabeled handwritten text is constantly generated. One strategy to take advantage of this considerably more significant amount of unlabeled data is to appropriately define the learning objectives so that the data itself provides supervision. Self-supervised learning has been quite successful in the field of speech recognition for a long time, and includes processes such as Wav2vec [12] and natural language processing (NLP), as evidenced by Collobert–Weston 2008 model [13], Word2Vec [14], GloVE [15], and, more recently, BERT [16], RoBERTa [17], and others.

This paper introduces Self-Writer: a clusterable embedding-based, self-supervised writer recognition directly from unlabeled data. The term "embedding" refers to the process of creating vectors of continuous values. Currently, triplet [18], and pairwise loss [19] techniques can be used to generate embeddings in the context of DL. Three parallel inputs pass across the network in a triplet loss architecture: anchor, positive, and negative. Concerning the anchor, the positive input has an identical class, whereas the negative input has a distinct class. A pair of information flows across the network in pairwise architecture belonging to the same or separate classes. Furthermore, we insist on making the training process for DL architecture self-supervised. The system, however, requires manuscripts of handwritten text and needs to ensure that the manuscripts comprise only one individual's handwritten text. The manuscripts come in lines of handwritten text and are further windowed into smaller frames, such as a word or text block, for training the DL framework. The construction of the training approach is illustrated in Figure 1. To the best

of our knowledge, this is the first attempt that exploits self-supervised learning strategy in writer recognition. In this paper, we make the following contributions.

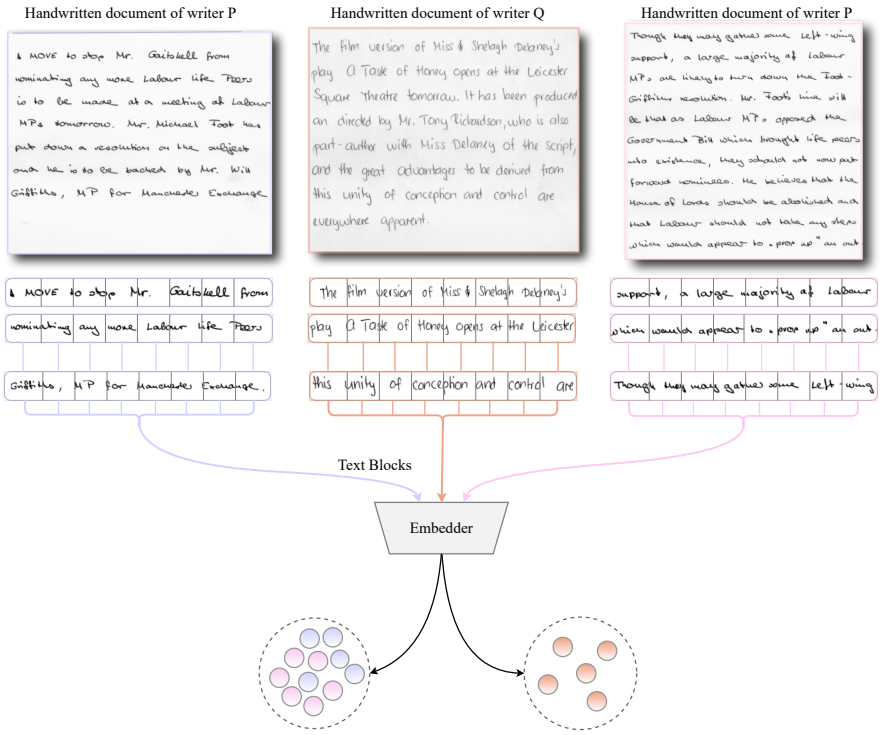

Figure 1. The figure demonstrates a set of handwritten documents with an unknown number of writers (in the example, two writers, p and q). Handwritten documents are segmented into a form of line, and further line-segmented images are windowed into smaller image frames, considering that all the frames of a single document belong to a single class. A DL-based embedding method also identifies feature similarities and relationships in handwritten documents. Clusterable embeddings are generated as a result of the technique.

- We introduce a self-supervised strategy of writer recognition based on generating clusterable embeddings, named Self-Writer. The training procedure learns directly from the unlabeled data.
- To train the Siamese architecture, we use a hypothesis-based pairwise constraint and nongenerative augmentation. The AutoEmbedder framework and nongenerative augmentation concentrate on the actual feature relationship instead of the hypothetical constraints.
- Two intercluster-based strategies—triplet and pairwise architectures—evaluate the proposed policy and conclude that a DL architecture can distinguish writers from pseudolabels depending on feature similarity.

We write the rest of the paper as follows. The recent literature regarding writer identification tasks is presented in Section 2. Section 3 explains the structure of the training strategy as well as the challenges and adaptations. Empirical setup regarding the evaluation of the proposed pipeline, datasets, and the investigation of the architectures' performance is outlined in Section 4. In Section 5, we sketch the pros and cons of the proposed approach. Finally, Section 6 concludes the paper.

2. Related Work

Writer recognition utilizing deep learning strategies has gained profound attention by researchers to address distinctive writer recognition and verification tasks. Over the past few years, significant research has been done on offline writer recognition, and many decent solutions are available in this domain. Among them, the techniques exploiting the hidden Markov model (HMM), Gaussian mixture model (GMM), deep neural networks

(DNNs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs) were the most prominent. The robustness of modern deep learning architectures provides an excellent structure for the latest writer recognition systems [20].

Before the proliferation of neural network approaches, Gabor filters and XGabor filters, and scale-invarient feature transform (SIFT) were mostly used to extract feature data. The majority of the research efforts applied wavelets [21], graph relations [22], statistical analysis [23,24], and HMM-based [25] models after feature extraction. By exploiting the weighted histogram of GMM scores and a similarity and dissimilarity Gaussian mixture model technique, Khan et al. [1] introduced an offline writer recognition system. Because the weighting process penalizes irrelevant descriptors, this technique achieves substantially better performance than the traditional averaging of negative loglikelihood values. In [26], a novel approach for writer identification is presented, based on the LDA model with n-grams of author texts and cosine similarity. For language-independent writer recognition, Sulaiman et al. [7] presented a mixture of handcrafted and in-depth features, extracting both LBP and convolutional neural network (CNN) features from overlapped frames and encoding the local information by using the VLAD technique. However, these methods showed a decent performance but were less accurate than modern neural network architectures because of their weak feature-extraction capability. Due to deep learning, various complex computer vision tasks such as visual reasoning are developed [27,28].

With the improvement of neural network architectures, more accurate approaches have been proposed in the writer recognition domain. Christlein et al. [29] presented a three-step pipeline for writer recognition: feature extraction with CNN, aggregating local features into one global descriptor and normalizing the descriptor. The authors aimed to investigate complicated and deep CNN architectures and some new findings such as the advantage of Lp-pooling over max pooling, and the normalization of activation following convolutional layers of the network. Zhang et al. [30] suggested a writer recognition framework by using the recurrent neural network (RNN) model for directly dealing with online handwriting raw data. Their framework outperforms the handcrafted feature-based and CNN-based techniques due to its robustness. In [31], Semma et al. employ FAST key points and the Harris corner detector to identify points of interest in the handwriting and extract key points from handwriting and feeding small patches around these key points to a CNN for feature learning and classification. Xing et al. [6] proposed DeepWriter, a text-independent writer recognition based on a deep, multistream CNN. The main drawback of the paper is that when the number of writers is increased, the model's accuracy is significantly reduced. Fiel et al. [32] presented the feature vector generation for each writer by using a CNN to identify writers by analyzing their handwritten texts. The feature vector approach uses preprocessing techniques such as binarization, text line segmentation, and sliding windows, and extracts images from the ICDAR 2011, 2013 dataset. However, this study shows poor results on the other datasets. Sheng He et al. [33] proposed multitask learning to provide a deep adaptive learning method for writer recognition based on single word pictures. This method improved the existing features of CNN by recognizing the content to analyze a writer's recognition, and exploited deep features. In the evaluation, they used the CVL and IAM datasets that contain segmented word pictures with labels for both word and writer. Furthermore, the authors proposed FragNet [34], a two-pathway network defined by a feature pyramid, which is used to extract feature maps, and fragment pathway, which is trained to predict the writer identity based on fragments extracted from the input image and the feature maps on the feature pyramid. The main drawback of the FragNet model is that it requires word image or region segmentation, which is challenging on highly cursive script documents. Nevertheless, writer recognition based on single-word images has not yet shown satisfactory performance. Deep learning achieves few-shot learning through meta-learning by using previous experience. In [35], the authors proposed a deep learning method that uses meta-learning to learn and generalize from a small sample size in image classification.

The attention mechanism has been widely used in recent years and has overcome few-shot learning. This technique was typically used with CNN or RNN to improve deep

feature extractions in writer identification. Zhang et al. [36] introduce a new residual Swin transformer classifier (RSTC) that integrates both local and global handwriting styles and produces robust feature representations with single-word pictures. The transformer block models local information with interacting strokes while holistically encoding with the identity branch and global block features' global information. Chen et al. [37] proposed the letters and styles adapters (LSA) to encode different letters, which were inserted between CNN and LSTM. To aggregate features, they also introduced hierarchical attention pooling (HAP).

Apart from the aforementioned methodologies, unsupervised writer recognition is still an underresearched domain. Very few researchers have worked on this and achieved significant results. Christlein et al. [38] trained a residual network by using deep surrogate classes, and the learned activation features without supervision outperformed the descriptors of cutting-edge methods for writer recognition. To study the impact of interlinear spacing, the authors wanted to evaluate single handwritten lines rather than whole paragraphs. In addition, a few semisupervised learning methods have been introduced as well for writer recognition. With the aim of improving writer recognition performance, Chen et al. [39] suggested a semisupervised feature learning. Their method trains both unlabeled and labeled data at the same time. The authors also proposed a data augmentation method called weighted label smoothing regularization (WLSR). The proposed WLSR method depends on the similarity of the sample space between the original labeled samples and additional unlabeled samples and can regularize the baseline of a CNN to enable the learning of more discriminated features.

Due to the difficulties of extensive data labeling for supervised deep neural networks as well as the ineffectiveness of unsupervised learning, self-supervised learning has become a promising research area for deep neural networks. Deep neural networks are usually trained through backpropagation by utilizing some objective function. However, it is challenging to estimate what objective function extracts suitable feature relations that could guide good neural networks without labels. Self-supervised learning addresses this issue by presenting different self-supervision tasks for networks to solve. Using self-supervision makes it easier to measure the performance captured by using an objective function similar to those used in supervised learning without requiring any labels. Many such tasks have been proposed in the last few years. For example, in the case of NLP, one can hide a word from a sentence and ask the network to predict the missing word. In addition, many computer vision-based self-supervised learning tools have been proposed in the last few years [10,40]. In [41,42], the authors use time as a source of supervision in videos, simply predicting the frames in a video. Self-supervision can also operate with a single image. One can hide a portion of the image given the task to the network to generate pixels of the hidden part [43,44] or recover color after grayscale conversion [40,45]. Another approach is to create a synthetic categorization task where one can create a surrogate class by altering a single image multiple times through translations, color shifts, and rotations [46]. Furthermore, in [47], in order to detect 3D symmetry from single-view RGB-D images, the author uses weak supervision to detect objects.

In recent years, self-supervised learning has shown great success in NLP such as BERT [16], RoBERTa [17], and Glove [15]; in the field of speech recognition, Wav2Vec [12] has had success, and in the field of computer vision [10,48] has worked well. However, none of the research was conducted on writer recognition in a self-supervised manner. Moreover, the generation of abundant, unlabeled, handwritten text from different individuals drives us to solve the writer recognition problem in a self-supervised manner based on the inter-feature relationships of data, all without relying on the labels.

3. Methodology

This section presents the proposed self-supervised writer recognition pipeline in more detail. The generation of clusterable embeddings, in this paper, is established on self-supervised learning. First, a self-supervision task is created depending on the following assumption: in most cases, whenever a writer starts writing, he/she writes on a blank

manuscript. As a result, most manuscripts include one individual's handwriting. However, some individuals might contain multiple manuscripts, or some may be impure, i.e., a manuscript might contain the writings of numerous individuals. Nevertheless, the impurity ratio would be sufficiently low in the most general handwritten manuscripts. As a result, one of the most prevalent neural network pipelines, the Siamese network [43], is used to investigate such a strategy. To extract embeddings, we use the AutoEmbedder framework [19] as a DL architecture. These generated embedding points work to extract features of the writer's handwriting characteristics, which helps to recognize the writer. The basic workflow of Self-Writer is illustrated in Figure 2.

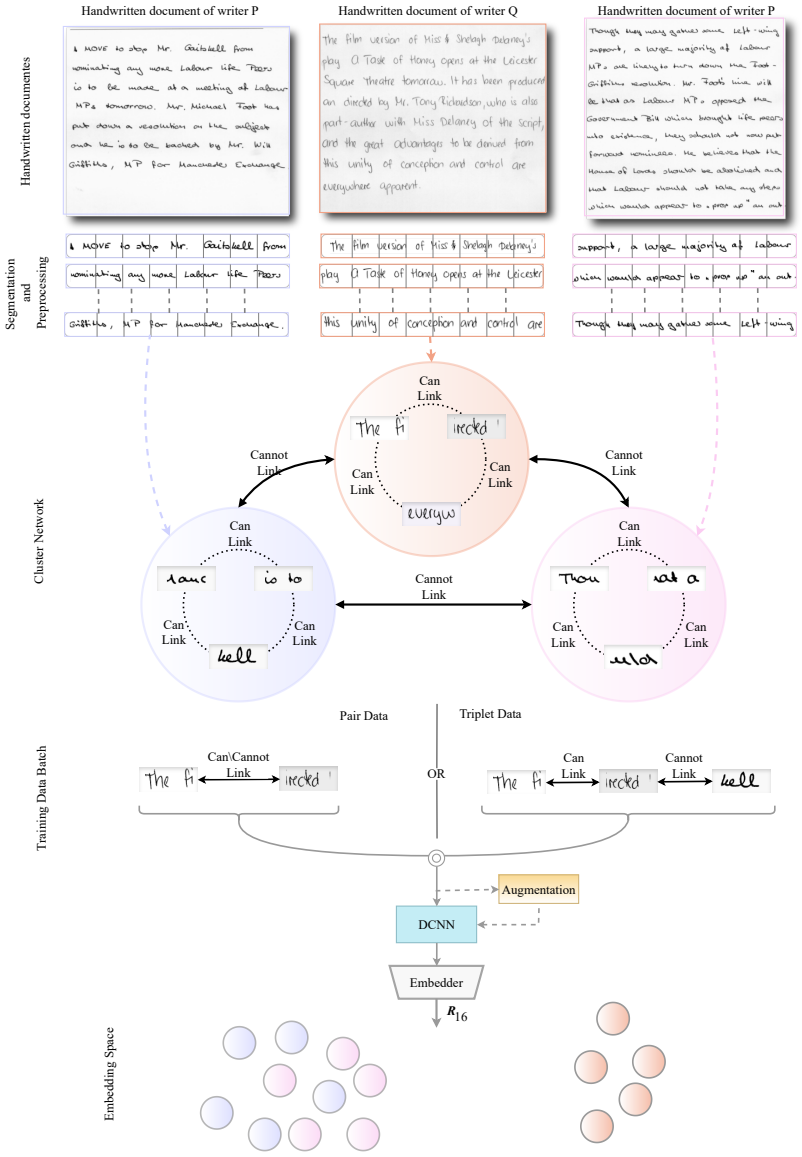

Figure 2. Overall procedure of Self-Writer. First, each manuscript is segmented into lines and assigned a pseudo label for each script. Additionally, an OpenCV-based Python script is used to preprocess the line images. Furthermore, a cluster network is constructed from the manuscript's line segments, using a nonoverlapping sliding window approach to generate smaller text blocks. Finally, depending on the requirements of the Siamese network, the cluster network is used to construct training data batches. The pairwise architecture receives two input data; either a can-link pair or a cannot-link pair. However, it demands an equal number of can-link and cannot-link pairs in a batch of training data. On the other hand, triplet architecture receives three input data; a pair of can-link data and cannot-link data, and then the DL architecture or the embedder is trained on randomly augmented training data.

The methodology section is organized as follows. First, we explain the preprocessing step in Section 3.1. In Section 3.2, the self-supervision task is discussed, followed by the problem formulation and assumptions in Section 3.3. Furthermore, the construction of pairwise constraints is defined in Section 3.4. In Section 3.5, uncertainties in the pairwise constraints are discussed. Finally, a detailed description of the DL framework, training procedure, and data augmentation schema is presented in Sections 3.6 and 3.7.

3.1. Data Preprocessing

In our experiment, handwritten texts are considered to be manuscripts. Furthermore, we require line segmentation of the handwritten scripts. Researchers, such as [49–51], have introduced different line segmentation techniques. However, the IAM [52], and CVL [53] datasets already provide line segmentation schema. However, some lighting, background, and noise issues are observed in the line images. First, we apply a supplementary OpenCV-based Python script [54] to eliminate unwanted data such as noise removal, background elimination, etc. Figure 3 represents (i) the raw version of the image and (ii) the enhanced version. The preprocessing part aims to enhance image quality and improve image readability information. Afterward, we resize line-segmented images with a height of 112 pixels while maintaining the aspect ratio. Note that the fixed-size representation of line images may distort the writer's handwriting characteristics. Then, we segment the line images into smaller text blocks by using a non-overlapping sliding window approach. Finally, we have scaled the dataset in the range [0,1].

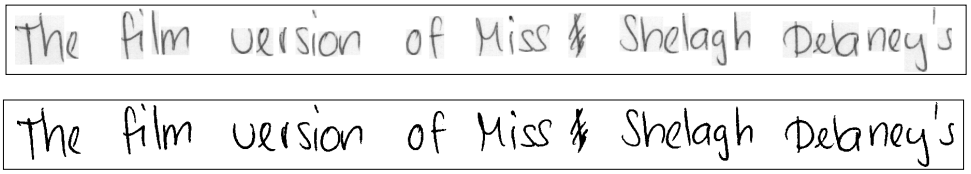

Figure 3. Raw line segmented images of the IAM dataset and an enhanced version of the image after applying a supplementary OpenCV-based python script.

3.2. Self-Supervision Task

Self-supervised learning has various forms based on the domain. Self-Writer aligns with contrastive self-supervised learning strategies [55]. In order to learn from self-supervised learning, the system must define a self-supervision task. In general, self-supervised learning receives supervision signals by utilizing the underlying structure of the data. Self-supervised learning takes advantage of the data's structure. As a result, it can leverage a wide variety of supervisory signals across large datasets based on cooccurring modalities without relying on labels. Because our proposed writer recognition method is based on self-supervised learning, we require handwritten scripts to get the supervision signals from the data by considering each manuscript as a different individual assigning a pseudo label. Furthermore, the documents are windowed into smaller text blocks to train the DL architecture in a supervised manner based on the pseudo label. The self-supervised task of the DL architecture is to generate clusterable embedding of the text block of manuscripts. The self-supervision task leads us to a supervised loss function. However, the final performance of the self-supervision task is usually unimportant to us. Instead, we are more interested in learning the intermediate representation of data. We validate in Section 4.3 that the self-supervision task holds excellent semantic or structural meanings and be helpful for the DL framework to recluster data based on feature similarities instead of the hypothetical assumption.

3.3. Paper's Assumptions

The proposed strategy aims to resolve handwriting recognition in a self-supervised manner depending on some hypothetical assumptions. Table 1 illustrates the mathematical notations employed in this work to make it easier for readers. To understand the problem

statement, consider D as a dataset of handwritten text in manuscripts, where X_k represents a single manuscript containing an individual's handwriting. Consider x_i to be a smaller text block of the manuscript, with $x_i \in X_k$. M number of nonoverlapping text blocks are extracted from a specific manuscript, X_k. Because a manuscript is associated with a single person, the smaller text blocks are also associated with that person. Based on this criterion, we created a cluster network known as pairwise constraints between two text blocks. If two text blocks are from the same script, they are considered in the same cluster. On the contrary, two text blocks from different scripts are considered different clusters. A set of clusters C can be formed based on the pairwise relationship, where each cluster $c_i \in C$ belongs to a particular manuscript.

Considering most manuscripts contain one person's handwriting, we can consider that most clusters c_i hold a single person's data. However, a single individual can have multiple manuscripts, and the individual's data may be spread across multiple clusters. As a result, the challenge is to find optimal cluster relationships such that no two clusters contain data from the same individual.

Table 1. A summary of the mathematical notations used in the paper is provided.

Notation	Description
D	A set of manuscripts of handwritten text. We assume that most manuscripts contain handwriting of an individual.
X	A single handwritten manuscript, $X \in D$.
x_i	A text block, generated by taking non-overlapping sliding window approach from a line segmented images of a manuscripts, $x_i \in X_k$.
M	The numbers of possible text blocks in a manuscript. Therotically, $M \times \lvert x_i \rvert = \lvert X_k \rvert$
C	A set of clusters. Those clusters are constructed by utilizing the hypothetical cluster network. Because cluster correlation is established on manuscripts relations, it can consider $\lvert X \rvert = \lvert C \rvert$
c_i	Represents a subset of the entire set of clusters. c_i illustrates a cluster constructed by the interrelationship of text blocks on the document X_k.
N	The number of writers in X, considering the ground truth.
α	The pairwise [19] architecture's distance hyperparameter. In other architectures, may denote the state of connectivity between any two cluster nodes.

The DL framework aggregates numerous clusters into a single cluster that holds all of an individual's embeddings. We imply that if a DL function may accurately extract features from text blocks, it can provide an optimal reasoning of similarities and dissimilarities between text blocks. Furthermore, a suitably trained DL architecture can successfully recluster the data based on feature relationships rather than the number of hypothetical clusters.

3.4. Pairwise Constraints

The proposed approach uses a cluster network to train the DL embedding architecture, also known as pairwise constraints. A pairwise constraint specifies a pairwise relation between input pairs. Let us consider two input data x_i and x_j as two random text blocks. There are two possibilities: (i) text blocks may belong to the same manuscript (can-link constraints), or (ii) text blocks may belong to different manuscripts (cannot-link constraints). Mathematically, we can represent it as follows,

$$\begin{aligned} \forall x_i \in X_k \text{ and } \forall x_j \in X_k;\ x_i, x_j \in c_k \\ \forall x_i \in X_k \text{ and } \forall x_j \notin X_k;\ x_i, x_j \notin c_k, \end{aligned} \tag{1}$$

where c_k is a separate cluster of the same class and X_k is a specific manuscript.

In the problem's current state, the writer's label or ground truth is unknown for all handwritten scripts, considering each document belongs to a distinct individual. As a result, the number of manuscripts, $|D|$ is the same as the number of unique pseudolabels.

The cluster constraints defined in (1) are used to train the DL framework. We define a ground regression function based on pairwise criteria derived in Equation (1) to properly introduce the intercluster and intracluster relation to a DL framework. The function is described as follows:

$$P(x_i, x_j) = \begin{cases} 0 & \text{if } x_i, x_j \in c_k \\ \alpha & \text{if } x_i \in c_p \text{ and } x_j \in c_q. \end{cases} \tag{2}$$

In Equation (2), the $P_c(.,.)$ function returns the distance constraints between embedding (generated from text blocks) pair. In general, the function implies that embedding pairs belong to the same cluster when their distance is zero; otherwise, they must be separated by α. However, embedding pairs from distinct clusters may be separated away by a distance greater than α, as defined in the AutoEmbedder framework in Equation (4). The pairwise constraints described in Equation (2) are used to train a DL framework.

3.5. Uncertainty of Pairwise Constraints

The cluster assignment of writers is uncertain due to two primary concerns: (i) the cluster assignment is unspecified concerning ground truth, and (ii) the manuscript X_k might be impure. Impurity, with regard to manuscripts, refers to a script that includes the handwriting of more than one writer. Theoretically, the number of writers considered ground truth labels, defined as $|N|$, is less than the number of cluster assignments according to the pseudo label, where $|N| < |C|$ and $|C| = |X|$. Due to such circumstances, the training dataset established on pairwise attributes often perceives an "error in can-link constraints" and "impurity in can-link constraints", as defined below,

- Error in cannot-link constraints: Consider that the input pair x_i and x_j belong to two different classes, $x_i \in c_p$ and $x_j \in c_q$, where $c_p \neq c_q$. Because the cluster assignment is based on manuscripts, the number of manuscripts outnumbers the actual number of writers. In consideration of the ground values, the hypothesis $c_p \neq c_q$ might be incorrect, and the input pair could belong to the same author.
- Impurity in can-link constraints: The main idea of the dataset is that a handwritten manuscript X_k comprises only one person's writing. In general, a manuscript may incorrectly identify writing and contain the writing of numerous individuals in a single manuscript. Let the input pair x_i and x_j belong to same script, $x_i, x_j \in c_i$. The manuscript might be impure, so the cluster assignment c_i may be wrong, and the input pair may belong to different individuals.

As our handwritten manuscripts contain a single individual's handwriting, the task of DL is to eliminate the error in cannot-link constraints based on the feature space relationship. As a result, if the features can be prioritized to a DL architecture, it may apparently combine appropriate clusters from inaccurate cannot-link constraints. However, impurity in can-link constraints can be considerably reduced in further segmentation procedures, such as sentence segmentation.

3.6. AutoEmbedder Architecture

We employ a pairwise constraint-based AutoEmbedder architecture as a DL framework to recluster handwritten text blocks. Moreover, we present further improvements to the network's overall training procedure to enhance learning progress. To train AutoEmbedder architecture, we use pairwise constraints specified by function $P(.,.)$ in Equation (2). The architecture adheres to Siamese network constraints, which can be stated as follows:

$$S(x_i, x_j) = ReLU(||M(x_i) - M(x_j)||, \alpha). \tag{3}$$

In Equation (3), $S(.,.)$ denotes a Siamese neural network (SNN) with a pair of inputs. The architecture shares a single DCNN, $M(,.,)$, which maps higher-dimensional input into meaningful lower-dimensional clusterable embeddings. The distance between generated embedding pair is calculated by using Euclidean distance and passed through a thershold ReLU activation fuction, which is derived in Equation (4):

$$ReLU(x) = \begin{cases} x & \text{if } 0 \leq x < \alpha \\ \alpha & \text{if } x \geq \alpha. \end{cases} \tag{4}$$

The threshold value α in Equation (4) indicates the cluster margin of the network. As a consequence of the cluster margin α, S(.,.) function produces output in range [0,α]. Figure 4 illustrates the overall architecture of AutoEmbedder using a Siamese neural network. The generic AutoEmbedder framework is trained by using the L2 loss function. The AutoEmbedder framework is trained for each data batch with an equal amount of can-link and cannot-link constraints. However, the problem is easily handled in a triplet architecture because each triplet includes a combination of cannot-link (anchor-negative) and can-link (anchor-positive) pairs.

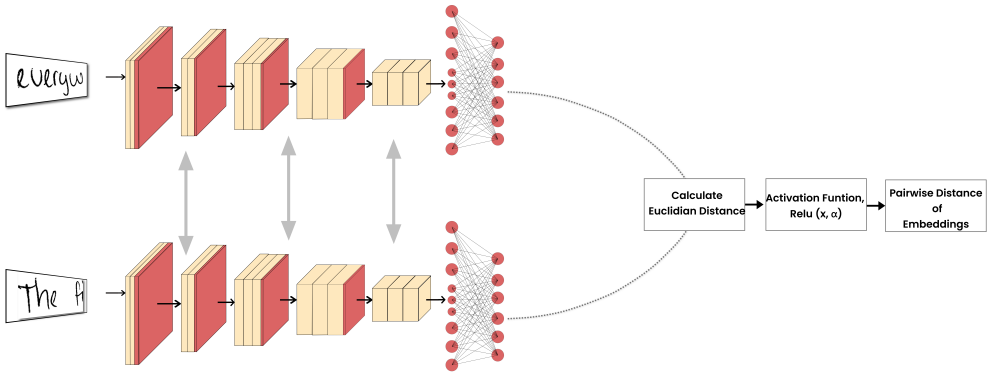

Figure 4. The training architecture of AutoEmbedder using a Siamese neural network (SNN). The subnetwork of SNN is weight-sharable, and the activation function is Relu, which is described in Equation (4). The architecture calculates pairwise distance output based on the generated embeddings pair.

3.7. Augmenting Training Data

In terms of the ground truth, both can-link and cannot-link cluster connections may include faulty assumptions. Therefore, a simple augmentation schema is applied to prevent the DL framework from overfitting faulty cluster associations. Even though there are a variety of augmentation approaches available, we prefer to combine the augmentation process described in Table 2.

Here, the augmentation pipeline includes the nongenerative online augmentation of half of the training batch data with an augmentation probability of 0.5. However, in a "Oneof" block, the transformations are defined along with their probabilities. The block normalizes the probability of all transformations within the block and applies one transformation on the image based on normalization. In this way, there is more efficiency in applying suitable transformations. The block also has a probability parameter, which indicates the probability of undertaking the block or not. Furthermore, all the transformations are defined according to their probabilities, and they are illustrated in Table 2.

Table 2. The table presents the augmentation pipeline associated with transformation definitions along with their probabilities.

Oneof Blocks	Transformations	Description	Probability of Transformations	Probability of Oneof Blocks	Augmentation Probability
Oneof	Flip	Flip the input either horizontally, vertically	0.5	0.5	0.5
	Crop and Pad	Randomly crop input image and pad images based on image size fractions.	0.5		
Oneof	Downscale	Decreases image quality by downscaling and upscaling back.	0.3	0.5	
	Gaussian Blur	Apply a Gaussian filter with a random kernel size to blur the input image	0.3		
	Motion Blur	Apply motion blur to the input image using a random-sized kernel.	0.3		
Oneof	Multiplicative Noise	Multiply images to a random number or array of numbers.	0.3	0.5	
	Random Brightness Contrast	Randomly change brightness and contrast of the input image.	0.3		
	Gaussian Noise	Apply gaussian noise to the input image.	0.3		
Oneof	Pixel Dropout	Set pixels to 0 with some probability.	0.5	0.5	
	CoarseDropout	Coarse drop out of the rectangular regions in the image	0.5		

In the case of erroneous data pairs, augmenting image frames makes the AutoEmbedder network less confusing. The architecture may be enhanced by augmenting it while disregarding erroneous data pairs caused by different transformations. Furthermore, augmenting data causes data variation, which allows the network to extract more useful features from the data. Algorithm 1 presents the pseudocode of the pairwise training process.

Algorithm 1: Self-Writer training algorithm

Input: Subset of training dataset X, DL model with initial weights M, Number of iterations *epochs*, Training batch size *batchSize*, Distance hyperparameter α

Output: Trained Embedding DL model.

Initialize a siamese network with ,$ReLU(S(.,.),\alpha)$;
$iter \leftarrow 0$;
while $iter < epochs$ **do**
  foreach $X_{batch} \in X$ **do**
    Initialize empty lists, $I, I', Y \leftarrow \{\}, \{\}, \{\}$;
    $counter \leftarrow 0$;
    foreach $x \in X_{batch}$ **do**
      $I \leftarrow$ append x in I, ;
      if $counter$ **mod** 2 **then**
        $I' \leftarrow$ randomly choose and append a can-link text block from X ;
        $Y \leftarrow$ append 0 in Y;
      else
        $I' \leftarrow$ randomly choose a cannot-link text block from X ;
        $Y \leftarrow$ append α in Y;
      $counter \leftarrow counter + 1$;
    $I \leftarrow$ randomly choose half of data and augment them;
    $I' \leftarrow$ randomly choose half of data and augment them;
    $S \leftarrow$ Train S with I, I', Y

4. Results

This section evaluates the proposed self-supervised writer recognition method called Self-Writer. As the architecture objective is to generate clusterable embedding, the K-means algorithm is used to measure the purity of the embedding clusters. In Section 4.1, we present the evaluation metrics. A brief description of the dataset is provided in Section 4.2. Section 4.3 discusses the implementation details and the training procedure of our proposed Self-Writer. Finally, the result analysis is presented in Section 4.3.

4.1. Evaluation Metrics

To measure the clustering effectiveness of generated embeddings of the Self-Writer schema, three well-known metrics, normalized mutual information (NMI), accuracy (ACC), and adjusted rand index (ARI), are used. The evaluation metrics are discussed below.

- Normalized Mutual Information: The normalized mutual information can be mathematically defined as

$$NMI(c, c') = \frac{I(c, c')}{max(H(c), H(c'))}, \tag{5}$$

  where c and c' are the ground truth and predicted cluster, respectively. $I(.)$ define the mutual information between c and c', and $H(.)$ denotes the entropy.
- Accuracy: Accuracy refers to the unsupervised clustering accuracy, expressed as

$$ACC(c, c') = \left(max \frac{\sum_{i=1}^{n} l(c_i = m(c_i'))}{2} \right), \tag{6}$$

  where l_i defines the ground truth labels, c_i denotes the cluster assignment produced by Self-Writer, and $m(.)$ ranges over all possible one-to-one mapping of the labels and clusters, from which the best mapping is taken.
- Adjusted Rand Index: The adjusted rand index is calculated by using the contengency [56]. The ARI can be expressed as

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}. \tag{7}$$

  Here, n_{ij}, a_i, and b_j are the values of the contingency table produced by the Self-Writer.

All three metrics produce a result in between the [0, 1] range. The higher value of these indices indicates a better correlation between ground truth and cluster prediction.

4.2. Datasets

4.2.1. IAM

The IAM is one of the most prominent and renowned English handwritten datasets, containing 1539 scanned handwritten scripts with 657 distinct writers using various pens. The manuscripts are scanned at 300 dots per inch (DPI) with 256 gray levels. However, the dataset comes with different forms such as manuscripts, sentences, words, and lines that provide different handwriting and word-recognition protocols. Out of 657 writers, 356 writers contribute only a single handwritten script. Each writer provided a number of documents ranging from one document (356 writers) to the most oversized (59 documents from one writer). Due to the variance of patterns of each writer, we consider the writers who provided more than equal four manuscripts and conducted the experiment with the first four manuscripts of the writers.

4.2.2. CVL

Another recent handwriting dataset for writer recognition is the CVL [53] handwriting dataset containing 1606 handwritten scripts with 310 distinct writers using different color

pens. A total of 282 writers contributed five manuscripts samples (four in English and one in German), and the rest contributed seven manuscripts (six in English and one in German). The dataset is also different from the IAM dataset. However, unlike the IAM dataset, the CVL dataset is well distributed. In this experiment, we also consider all the manuscripts for each writer.

4.3. Results and Comparison

To analyze the embeddings based on the proposed strategy, AutoEmbedder (pairwise architecture) and a triplet architecture are implemented. Except for these two techniques, the most popular DL approaches for writer recognition do not adhere to the training characteristics discussed in the study. They often operate supervised learning strategies. Hence, they are omitted in this experiment.

For both DL frameworks, we use DenseNet121 [57] as baseline architecture. Furthermore, both DL architectures are connected with a dense layer containing 16 nodes. As a result, both architectures generate 16-dimensional embedding vectors. For the triplet network, we have added l2-normalization on the output layer, as it is suggested to increase the framework's accuracy [58], and valid triplet is generated manually. The pairwise architecture is trained by using default L2-loss also known as the mean square error (MSE), while semihard triplet loss [18] is used to train the triplet architecture. The training pipeline is illustrated in Figure 5.

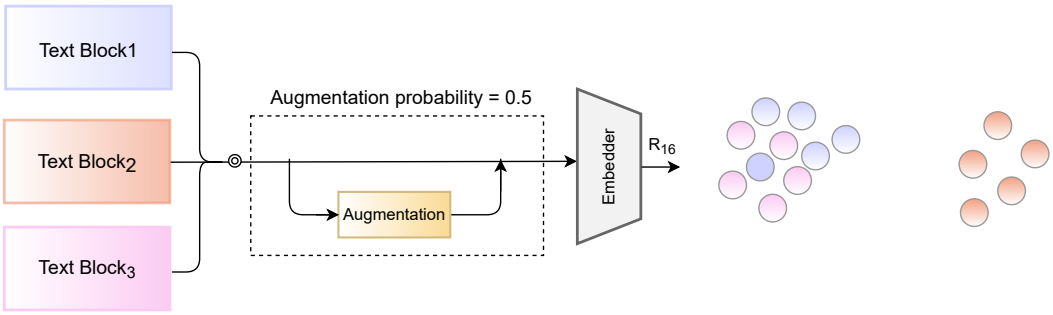

Figure 5. The same data processing pipeline is used to train both pairwise and triplet frameworks. The DL frameworks receive half of the inputs randomly augmented with an augmentation probability of 0.5.

The evaluation phase ensures that both frameworks are trained by using an identical dataset. Because the proposed approach is self-supervised and deals with unlabeled datasets, the frameworks get the exact dataset for training and testing purposes. However, the labels for the training process are unspecified and initiated on the paper's hypothetical premises. A dataset such as this is referred to as a training dataset. Considering the ground truth values of writers with the same dataset is referred to as the ground dataset. We used a batch size of 64 to train both frameworks. The training is carried out with the Adam [59] optimizer with a learning rate of 0.0005.

The training phase of Self-Writer includes high computational complexity, including online data augmentation. In addition, computing NMI, ACC, and ARI metrics required quadratic time complexity. As a result, we have decided to restrict the number of writers to 150. We trained on a subset of the dataset rather than the entire dataset. In order to test the ground truth data, two random samples of each text segment are chosen. The model was trained over 400 epochs.

Figure 6 compares the triplet and pairwise networks during training on two distinct datasets, with writers equal to 25 and impurity equal to 0. The triplet architecture learns from the training dataset in a seamless manner and overfits immensely on the augmented training data. The benchmark of the ground dataset is also anticipated because the metrics of triplet architecture increase at first and then drop dramatically due to overfitting. From

Figure 6's triplet architecture on two different datasets, it can be conceded that it only remembers the features related to the hypothetical labels.

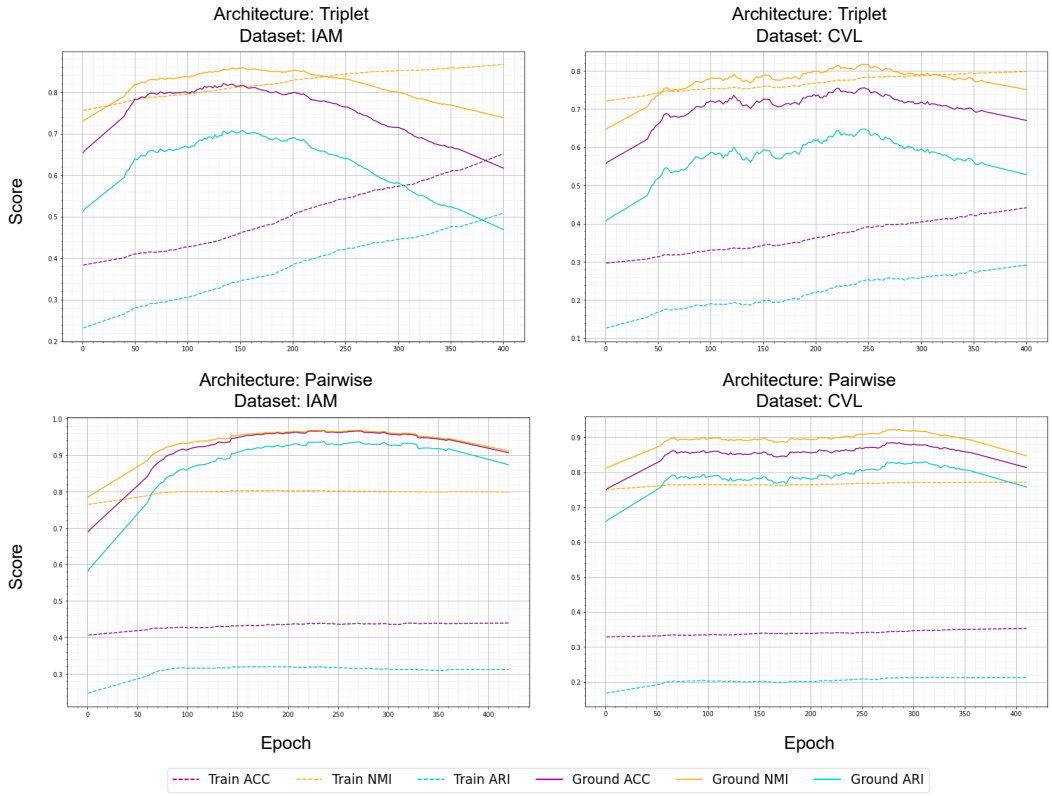

Figure 6. Graphs illustrating the metrics of the training and ground dataset containing 25 writers with an impurity of 0. The first row represents the triplet network and the second row represents the pairwise network, respectively.

In contrast, the pairwise framework produces an adequate performance with some inconsistencies. Generally, DL frameworks generate more accuracy on training data than validation data. However, the performance of the ground dataset is mostly superior to the training data in our method. Nevertheless, after 300 epochs, the performance on the ground dataset started to decrease gradually. The architecture started getting overfitted on training data due to the limited number of writers. Furthermore, for reducing overfit on training data, we increase the number of writers to 50, shown in Figure 7. The triplet framework still gradually overfits training data. Furthermore, the ground dataset's accuracy started to drop due to memorizing feature relationships based on hypothetical labels. However, the pairwise framework performed a steady performance on ground datasets.

The performance of the training method comprehensively depends on the impurity of training data. Increasing the impurity ratio reduces the architecture's performance. Benchmarks were conducted with impurity = 0.1 and 0.05, while considering writers = 50, as shown in Figures 8 and 9. The training architecture continues to overfit the triplet architecture. On the other hand, pairwise architecture gradually memorizes the training dataset based on feature relation.

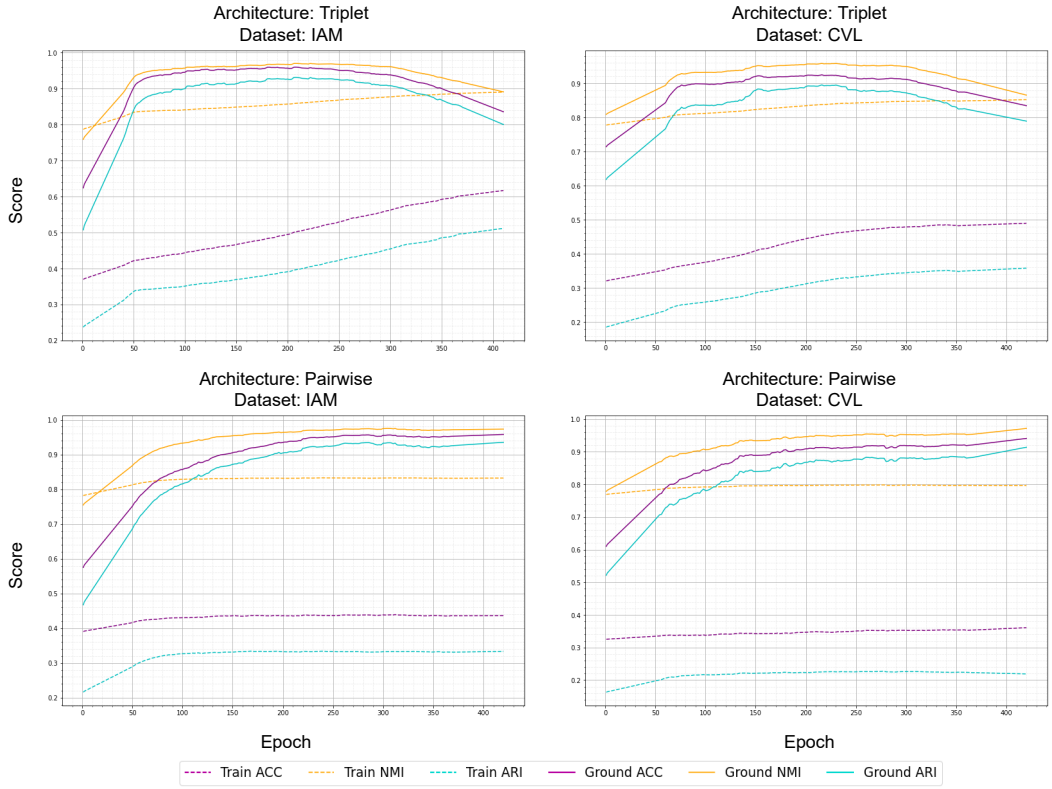

Figure 7. Graphs illustrating the metrics of pretext task and ground dataset containing 50 writers with the purity of 0. The first row represents the triplet network and the second row represents the pairwise network, respectively.

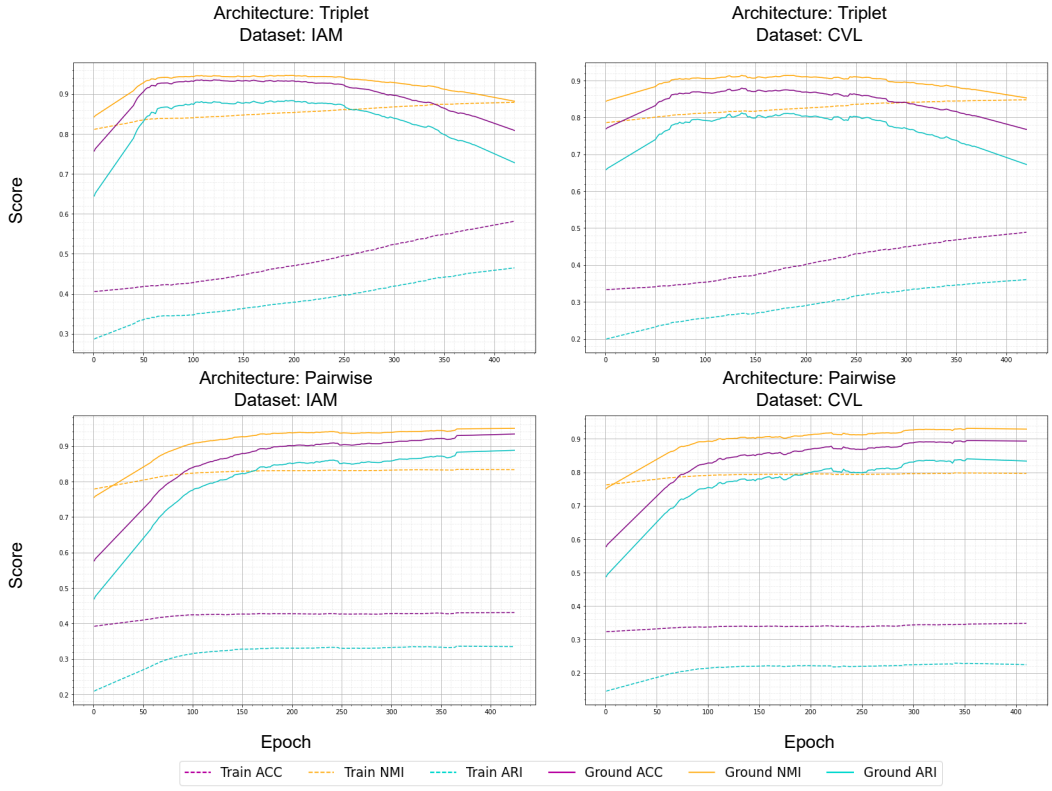

Figure 8. Graphs illustrating the metrics of the pretext task and ground dataset containing 50 writers with a purity of 0.05. The first row represents the triplet network and the second row represents the pairwise network, respectively.

The semihard triplet loss function is designed to minimize the embedding distance between positive and anchor data and strictly distance the embeddings of negative and anchor data. As the triplet architecture is trained over semihard triplet loss and heavily adheres to the aforementioned criteria, the architecture overfits hypothetical constraints while ignoring the real feature-dependent relationships.

In contrast, instead of overfitting training data, the pairwise architecture learns to extract features. The reason lies in AutoEmbedder's training strategy as L2-loss does not take into consideration the pseudolabel; instead, it learns aggregately from a batch of data. Therefore, the framework can obtain feature similarities because it is not precisely supervised using L2-loss. As a result, the architecture can recluster the data in hyperspace depending on the feature similarities.

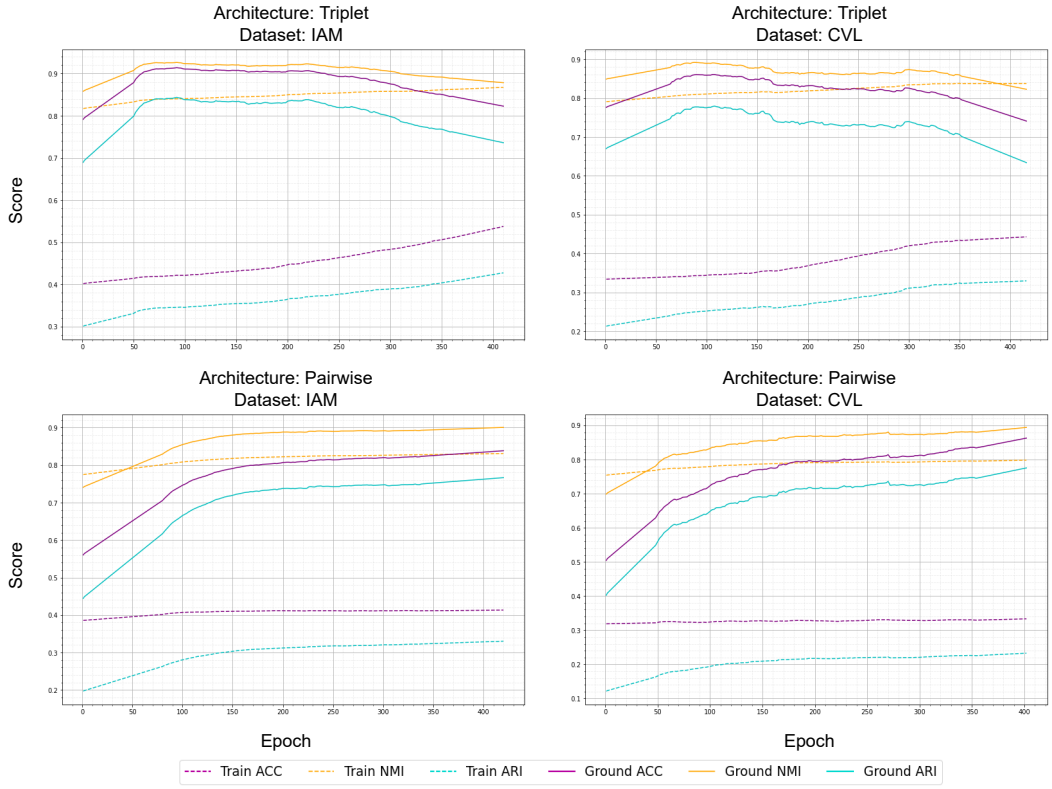

Figure 9. Graphs illustrating the metrics of the pretext task and the ground dataset containing 50 writers with impurity 0.1. The first row represents the triplet network and the second row represents the pairwise network, respectively.

With pairwise architecture-based AutoEmbedder, we further investigate several writers and impurity conditions. Tables 3 and 4 show the IAM and CVL datasets' evaluation metrics on the training and ground datasets. The table represents a comprehensive summary of the performance variance in the training dataset depending on the number of writers and impurity. On any dataset, the AutoEmbedder-based paired architecture retains a marginal performance with impurity = 0. Furthermore, increasing the number of writers and the impurity ratio causes a reduction in the architecture's performance. Although the number of writers is held constant at 25 and 50, a slight fluctuation is observed in both datasets. Increasing the number of writers by 50 resulted in an inconsistent improvement in performance.

Table 3. The table illustrates the pairwise architecture in the IAM dataset across four-speaker groups: 25, 50, 100, and 127. The table also analyzes two segmentation impurities, 0 and 0.1, for each group of writers to illustrate the shortcomings of the faulty assumption.

		Impurity = 0			**Impurity = 0.05**			**Impurity = 0.1**		
		NMI	**ACC**	**ARI**	**NMI**	**ACC**	**ARI**	**NMI**	**ACC**	**ARI**
25 writers	Pretext task	0.801	0.463	0.334	0.791	0.422	0.352	0.778	0.430	0.372
	Ground task	0.956	0.948	0.912	0.898	0.854	0.848	0.856	0.807	0.792
50 writers	Pretext task	0.849	0.452	0.351	0.845	0.432	0.348	0.834	0.424	0.345
	Ground task	0.988	0.969	0.934	0.958	0.943	0.897	0.903	0.861	0.801
100 writers	Pretext task	0.841	0.419	0.309	0.7895	0.394	0.310	0.731	0.398	0.312
	Ground task	0.901	0.841	0.813	0.876	0.823	0.801	0.851	0.816	0.794
127 writers	Pretext task	0.836	0.404	0.301	0.779	0.396	0.294	0.711	0.382	0.299
	Ground task	0.898	0.817	0.798	0.847	0.794	0.776	0.816	0.787	0.741

Table 4. The table illustrates the pairwise architecture in the CVL dataset across four-speaker groups: 25, 50, 100, and 150. The table also analyzes two segmentation impurities, 0 and 0.1, for each group of writers to illustrate the shortcomings of the faulty assumption.

		Impurity = 0			**Impurity = 0.05**			**Impurity = 0.1**		
		NMI	**ACC**	**ARI**	**NMI**	**ACC**	**ARI**	**NMI**	**ACC**	**ARI**
25 writers	Pretext task	0.786	0.372	0.228	0.811	0.384	0.246	0.771	0.368	0.250
	Ground task	0.943	0.910	0.908	0.899	0.862	0.857	0.907	0.850	0.810
50 writers	Pretext task	0.800	0.368	0.231	0.800	0.362	0.246	0.800	0.374	0.268
	Ground task	0.974	0.941	0.919	0.930	0.901	0.845	0.914	0.867	0.816
100 writers	Pretext task	0.786	0.352	0.228	0.764	0.340	0.219	0.744	0.336	0.214
	Ground task	0.908	0.871	0.819	0.894	0.846	0.770	0.861	0.811	0.784
150 writers	Pretext task	0.753	0.337	0.216	0.727	0.312	0.178	0.703	0.297	0.154
	Ground task	0.846	0.793	0.764	0.824	0.781	0.743	0.816	0.775	0.725

In order to investigate the appropriate feature relationship between text blocks, the architecture requires a significant amount of handwriting characteristics variations from users. Limiting the number of writers to 25, the architecture struggles to find more appropriate feature relationships and observes a reduction in performance. By increasing the number of writers to 50, the feature variances in training data are balanced and observed a performance improvement.

5. Discussion

The pairwise architecture with training strategy performs well in the writer recognition process. However, throughout the study, the architecture has several difficulties that must be addressed. First, training the architecture with less handwriting variation results in overfitting, as observed while the number of writers' dataset is 25. Secondly, as the system is fully segmentation-dependent, the target lies in developing an optimal audio segmentation procedure. Resolving these challenges would benefit the architecture for a wide range of writer recognition and evaluation usage. Furthermore, due to the use of Siamese architecture, the architecture has an identical subnetwork, increasing the computation throughout the process. Thus, the training strategy required a long period of time.

Apart from the limitations, the self-writer strategy requires no pretraining on large handwritten datasets, which is often observed in other writer recognition methods. Furthermore, the Self-Writer strategy requires comparatively less per-writer data than the other writer recognition methods. From an overall perspective, the Self-Writer keeps the requirement of labeled data to a minimum.

6. Conclusions

This paper presents Self-Writer, a self-supervised writer recognition system that generates clusterable embeddings depending on the writers' unique handwriting characteristics. Self-Writer deals with unlabeled data and is trained with pseudolabels. Self-supervised learning has its various forms based on the domain; self-writer aligns with contrastive self-supervised learning strategies. We evaluate such a strategy with two relevant DL architectures, pairwise and triplet. The empirical results demonstrate that the pairwise architecture-based AutoEmbedder, as an embedding architecture, performs better than triplet architecture for our proposed self-supervised writer recognition. Furthermore, the architecture performs well regarding the number of writers and handwritten text segmentation errors in unlabeled data. However, depending on the writers' variations, the method requires clean documents and robust line segmentation techniques to generate clusterable embeddings. Therefore, a segmentation technique and VLAD encoding might be an extended version of the proposed work. In addition, to evaluate the clusterable embedding, we use the K-means algorithm. However, locally weighted and multidiversified ensemble clustering, which enhances the clustering robustness by fusing the information of multiple-based clusterings, might be an extended version of the proposed work. Nevertheless, we firmly believe that such a comprehensive and hypothetical technique for generating hypothetical labels to train writer recognition systems will assist researchers in developing new strategies.

Author Contributions: Conceptualization, Z.M.; formal analysis, Z.M. and M.M.K.; funding acquisition, M.M.M.; investigation, M.A.H. and M.F.M.; methodology, Z.M. and M.F.M.; project administration, M.M.M. and M.A.H.; resources, M.M.M. and M.A.H.; supervision, M.F.M.; writing—original draft, Z.M. and M.M.K.; writing—review & editing, M.M.M. and M.A.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research work was funded by Institutional Fund Projects under grant no. (IFPIP: 320-611-1443). The authors gratefully acknowledge technical and financial support provided by the Ministry of Education and King AbdulAziz University, DSR, Jeddah, Saudi Arabia.

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

References

1. Khan, F.A.; Khelifi, F.; Tahir, M.A.; Bouridane, A. Dissimilarity Gaussian mixture models for efficient offline handwritten text-independent identification using SIFT and RootSIFT descriptors. *IEEE Trans. Inf. Forensics Secur.* **2018**, *14*, 289–303. [CrossRef]
2. Tapiador, M.; Gómez, J.; Sigüenza, J.A. Writer identification forensic system based on support vector machines with connected components. In Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Berlin/Heidelberg, Germany, 17 May 2004; pp. 625–632.
3. Fornés, A.; Lladós, J.; Sánchez, G.; Bunke, H. Writer identification in old handwritten music scores. In Proceedings of the 2008 the Eighth IAPR International Workshop on Document Analysis Systems, Nara, Japan, 16–19 September 2008; pp. 347–353.
4. Fornés, A.; Lladós, J.; Sánchez, G.; Bunke, H. On the use of textural features for writer identification in old handwritten music scores. In Proceedings of the 2009 10th International Conference on Document Analysis and Recognition, Catalonia, Spain, 26–29 July 2009; pp. 996–1000.
5. Ballard, L.; Lopresti, D.; Monrose, F. Evaluating the security of handwriting biometrics. In Proceedings of the Tenth International Workshop on Frontiers in Handwriting Recognition, La Baule, France, 23–26 October 2006.
6. Xing, L.; Qiao, Y. Deepwriter: A multi-stream deep CNN for text-independent writer identification. In Proceedings of the 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), Shenzhen, China, 23–26 October 2016; pp. 584–589.
7. Sulaiman, A.; Omar, K.; Nasrudin, M.F.; Arram, A. Length independent writer identification based on the fusion of deep and hand-crafted descriptors. *IEEE Access* **2019**, *7*, 91772–91784. [CrossRef]
8. Doersch, C.; Zisserman, A. Multi-task self-supervised visual learning. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2051–2060.
9. Zhai, X.; Oliver, A.; Kolesnikov, A.; Beyer, L. S4l: Self-supervised semi-supervised learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 1476–1485.
10. Doersch, C.; Gupta, A.; Efros, A.A. Unsupervised visual representation learning by context prediction. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1422–1430.

11. Gidaris, S.; Bursuc, A.; Komodakis, N.; Pérez, P.; Cord, M. Boosting few-shot visual learning with self-supervision. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 8059–8068.
12. Baevski, A.; Zhou, H.; Mohamed, A.; Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv* **2020**, arXiv:2006.11477.
13. Collobert, R.; Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 160–167.
14. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
15. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
16. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
17. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
18. Schroff, F.; Kalenichenko, D.; Philbin, J. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 815–823.
19. Ohi, A.Q.; Mridha, M.F.; Safir, F.B.; Hamid, M.A.; Monowar, M.M. Autoembedder: A semi-supervised DNN embedding system for clustering. *Knowl.-Based Syst.* **2020**, *204*, 106190. [CrossRef]
20. Janiesch, C.; Zschech, P.; Heinrich, K. Machine learning and deep learning. *Electron. Mark.* **2021**, *31*, 685–695. [CrossRef]
21. He, Z.; Fang, B.; Du, J.; Tang, Y.Y.; You, X. A novel method for offline handwriting-based writer identification. In Proceedings of the Eighth International Conference on Document Analysis and Recognition (ICDAR'05), Seoul, Korea, 29 August–1 September 2005; pp. 242–246.
22. Helli, B.; Moghaddam, M.E. A text-independent Persian writer identification based on feature relation graph (FRG). *Pattern Recognit.* **2010**, *43*, 2199–2209. [CrossRef]
23. He, Z.; Tang, Y. Chinese handwriting-based writer identification by texture analysis. In Proceedings of the 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826), Shanghai, China, 26–29 August 2004; Volume 6, pp. 3488–3491.
24. Zhu, Y.; Tan, T.; Wang, Y. Biometric personal identification based on handwriting. In Proceedings of the 15th International Conference on Pattern Recognition, ICPR-2000, Barcelona, Spain, 3–8 September 2000; Volume 2, pp. 797–800.
25. Schlapbach, A.; Bunke, H. A writer identification and verification system using HMM based recognizers. *Pattern Anal. Appl.* **2007**, *10*, 33–43. [CrossRef]
26. Anwar, W.; Bajwa, I.S.; Ramzan, S. Design and implementation of a machine learning-based authorship identification model. *Sci. Program.* **2019**, *2019*, 9431073. [CrossRef]
27. Zheng, W.; Liu, X.; Ni, X.; Yin, L.; Yang, B. Improving visual reasoning through semantic representation. *IEEE Access* **2021**, *9*, 91476–91486. [CrossRef]
28. Zheng, W.; Yin, L.; Chen, X.; Ma, Z.; Liu, S.; Yang, B. Knowledge base graph embedding module design for Visual question answering model. *Pattern Recognit.* **2021**, *120*, 108153. [CrossRef]
29. Christlein, V.; Bernecker, D.; Maier, A.; Angelopoulou, E. Offline writer identification using convolutional neural network activation features. In Proceedings of the German Conference on Pattern Recognition, Hannover, Germany, 12–15 September 2015; pp. 540–552.
30. Zhang, X.Y.; Xie, G.S.; Liu, C.L.; Bengio, Y. End-to-end online writer identification with recurrent neural network. *IEEE Trans. Hum.-Mach. Syst.* **2016**, *47*, 285–292. [CrossRef]
31. Semma, A.; Hannad, Y.; Siddiqi, I.; Djeddi, C.; El Kettani, M.E.Y. Writer identification using deep learning with fast keypoints and harris corner detector. *Expert Syst. Appl.* **2021**, *184*, 115473. [CrossRef]
32. Fiel, S.; Sablatnig, R. Writer identification and retrieval using a convolutional neural network. In Proceedings of the International Conference on Computer Analysis of Images and Patterns, Valletta, Malta, 2–4 September 2015; pp. 26–37.
33. He, S.; Schomaker, L. Deep adaptive learning for writer identification based on single handwritten word images. *Pattern Recognit.* **2019**, *88*, 64–74. [CrossRef]
34. He, S.; Schomaker, L. Fragnet: Writer identification using deep fragment networks. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3013–3022. [CrossRef]
35. Zheng, W.; Tian, X.; Yang, B.; Liu, S.; Ding, Y.; Tian, J.; Yin, L. A few shot classification methods based on multiscale relational networks. *Appl. Sci.* **2022**, *12*, 4059. [CrossRef]
36. Zhang, P. RSTC: A New Residual Swin Transformer For Offline Word-Level Writer Identification. *IEEE Access* **2022**, *10*, 57452–57460. [CrossRef]
37. Chen, Z.; Yu, H.X.; Wu, A.; Zheng, W.S. Level online writer identification. *Int. J. Comput. Vis.* **2021**, *129*, 1394–1409. [CrossRef]
38. Christlein, V.; Gropp, M.; Fiel, S.; Maier, A. Unsupervised feature learning for writer identification and writer retrieval. In Proceedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; Volume 1, pp. 991–997.

39. Chen, S.; Wang, Y.; Lin, C.T.; Ding, W.; Cao, Z. Semi-supervised feature learning for improving writer identification. *Inf. Sci.* **2019**, *482*, 156–170. [CrossRef]
40. Zhang, R.; Isola, P.; Efros, A.A. Colorful image colorization. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 649–666.
41. Walker, J.; Gupta, A.; Hebert, M. Dense optical flow prediction from a static image. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2443–2451.
42. Walker, J.; Doersch, C.; Gupta, A.; Hebert, M. An uncertain future: Forecasting from static images using variational autoencoders. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 835–851.
43. Guo, Q.; Feng, W.; Zhou, C.; Huang, R.; Wan, L.; Wang, S. Learning dynamic siamese network for visual object tracking. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 1763–1771.
44. Pathak, D.; Krahenbuhl, P.; Donahue, J.; Darrell, T.; Efros, A.A. Context encoders: Feature learning by inpainting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2536–2544.
45. Larsson, G.; Maire, M.; Shakhnarovich, G. Learning representations for automatic colorization. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 577–593.
46. Dosovitskiy, A.; Springenberg, J.T.; Riedmiller, M.; Brox, T. Discriminative unsupervised feature learning with convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 766–774 . [CrossRef]
47. Shi, Y.; Xu, X.; Xi, J.; Hu, X.; Hu, D.; Xu, K. Learning to detect 3D symmetry from single-view RGB-D images with weak supervision. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022,** 1-15. [CrossRef]
48. Noroozi, M.; Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 69–84.
49. Li, Y.; Zheng, Y.; Doermann, D.; Jaeger, S. Script-independent text line segmentation in freestyle handwritten documents. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *30*, 1313–1329. [CrossRef] [PubMed]
50. Malik, S.; Sajid, A.; Ahmad, A.; Almogren, A.; Hayat, B.; Awais, M.; Kim, K.H. An efficient skewed line segmentation technique for cursive script OCR. *Sci. Program.* **2020**, *2020*, 8866041. [CrossRef]
51. Zheng, W.; Liu, X.; Yin, L. Sentence representation method based on multi-layer semantic network. *Appl. Sci.* **2021**, *11*, 1316. [CrossRef]
52. Marti, U.V.; Bunke, H. The IAM-database: An English sentence database for offline handwriting recognition. *Int. J. Doc. Anal. Recognit.* **2002**, *5*, 39–46. [CrossRef]
53. Kleber, F.; Fiel, S.; Diem, M.; Sablatnig, R. Cvl-database: An off-line database for writer retrieval, writer identification and word spotting. In Proceedings of the 2013 12th International Conference on Document Analysis and Recognition, Washington, DC, USA, 25–28 August 2013; pp. 560–564.
54. Mridha, M.F.; Ohi, A.Q.; Ali, M.A.; Emon, M.I.; Kabir, M.M. BanglaWriting: A multi-purpose offline Bangla handwriting dataset. *Data Brief* **2021**, *34*, 106633. [CrossRef]
55. Jaiswal, A.; Babu, A.R.; Zadeh, M.Z.; Banerjee, D.; Makedon, F. A survey on contrastive self-supervised learning. *Technologies* **2020**, *9*, 2. [CrossRef]
56. Santos, J.M.; Embrechts, M. On the use of the adjusted rand index as a metric for evaluating supervised classification. In Proceedings of the International Conference on Artificial Neural Networks, Limassol, Cyprus, 14–17 September 2009; pp. 175–184.
57. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
58. Hermans, A.; Beyer, L.; Leibe, B. In defense of the triplet loss for person re-identification. *arXiv* **2017**, arXiv:1703.07737.
59. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.