TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV
Faculty of Computer Science and Cybernetics
Department of Mathematical Informatics

MASTER'S THESIS

Major: 122 Computer Science
on the topic:

DEEP LEARNING BASED FEATURE EXTRACTION FOR HANDWRITING STYLES CLUSTERING

Prepared by _____
signature
Full-time master student of the 2nd year
Karyna O. Korovai

Scientific advisor: _____
signature
Dr. Sc. (Phys.-Math.)
Prof. Vasyl M. Tereshchenko

I certify that there are no borrowings from the works of other authors without appropriate references in this paper.
Student _____
signature

The paper was considered and approved for the thesis defense at a meeting of the Department of Mathematical Informatics «_____» _____ 20_____, protocol № ___

Head of the Department of Mathematical Informatics
Prof. Vasyl M. Tereshchenko _____
signature

Kyiv – 2021

ABSTRACT

Research paper size: 63 pages, 15 pictures, 2 tables, 47 references.

ALLOGRAPH CLUSTERING, AUTOENCODER,CLUSTERING ALGORITHMS, DEEP LEARNING,DYNAMIC TIME WARPING, FEATURE EXTRACTION, LONG-SHORT TERM MEMORY, MACHINE LEARNING, ONLINE HANDWRITING, RECURRENT NEURAL NETWORK.

Object of study: handwritten allograph clustering.

The goal of the work is to study the existing solutions to the problem of allograph clustering, and to build a solution with deep learning based feature extraction technique.

Development methods: data analysis, machine learning, deep learning, autoencoder, clustering methods, data visualization.

Instrumentation: Python (3.6 version) for pre- and post-processing, data analysis, feature extraction and clustering. Free freely distributable integrated development environment (IDE) for the Python programming language JetBrainsPyCharm2021.1. Free machine learning library for the Python programming language scikit-learn package. Free open source neural network library Keras.

Results: a detailed analysis of existing methods for allograph clustering has been investigated and performed. We overviewed handcrafted (based on handcrafted feature extraction with further shallow clustering) and automated (NN-based) approaches. We proposed a new method for allograph clustering which combines learning efficient representations by RNN AE and DTW algorithm in a way that is not computationally expensive, but still provides a human-likeness of achieved results. We also evaluated and compared results based on internal metrics and pure visual analysis.

CONTENTS

ABBREVIATIONS

AE – Autoencoder;

ANN – Artificial Neural Network;

BSS – Between Cluster Sum of Squares;

CHI – Calinski-Harabasz Index;

DBI – Davies-Bouldin Index;

DL – Deep Learning;

DTW – Dynamic Time Warping;

IAM-OnDB – IAM On-Line Handwriting Database;

IDE – Integrated Development Environment;

LOB – Lancaster-Oslo-Bergen (text corpus);

LSTM – Long Short Term Memory;

ML – Machine Learning;

MSE – Mean Squared Error;

RNN – Recurrent Neural Network;

SIL – Silhouette Coefficient;

VRC – Variance Ratio Criterion;

WSS – With-In Sum of Squares.

INTRODUCTION

Current state of the object of development. Problems of classification and clustering of handwriting style have been widely studied for both online [1-3] and offline [4] handwriting data representation. Approaches for online handwriting clustering use different shallow algorithms like SOM [3], hierarchical clustering [2], and K-means [1], while focusing more on a finding an efficient feature representation. Problems of online handwriting clustering also appears in handwriting recognition [5-6], authentication, verification and writer identification tasks [7-11], and more recently for human-like handwriting generation[12]. A number of modern application use such approaches for text/non-text classification,etc.[13-14].

An allograph can be defined as a handwritten character with a prototypical shape [2,10], which basically means existing of different types of writing (i.e. allographs) for one fixed letter.

In a number of approaches, this task is done with handcrafted features engineering [1]. The first steps in handwriting clustering problem, where each sample was represented as a set of handcrafted features, has been done with such approaches. However, these methods are strongly related to the specifics of the particular problem, and not always can be generalized for other tasks. Moreover, it requires a lot of time to research and construct handcrafted features.

Allograph clustering was studied widely in research papers byVuurpijl, Schomaker, and Niels [10-11] for allograph categorization, character recognition and allograph-based writer identification. The research in this area is still ongoing due to a number of challenges that researches face.Niels et al. [10] proposed using dynamic time warping algorithm to hierarchically cluster allographs for further allograph-based writer identification. In such a way, we can achieve the results that corresponds to the human expectations. However, this approach is computationally expensive and cannot be applied to large datasets.

More recent tremendous success in deep learning allows finding effective learned representation. Deep learning revolution has wiped out much of the effort for feature engineering and replaced them with learning representations in many areas, for example, speech [15] and natural language processing [16].

Proposed solution. In this work, we describe our new online handwriting styles clustering system based on deep learning methods combined with dynamic time warping algorithm for post-processing adopted from Niels [10]. It replaces our previous handcrafted feature based approach [1] by introducing a learning representation for handwritten allographs.The major difference comes from feature extraction. In contrast to the 34 handcrafted features for each handwritten sample, we use only 8-dimensinal vector that stores compressed by AE information about the input sample. The new system also reduces the amount of person-hours required in order to build a final solution. This approach can also be applied to other similar tasks (i.e. regarding time series clustering) with no additional time for manual feature engineering required, since it is assumed that the model will identify an efficient feature representation all by itself.

Among the main advantages of our method is combining learning efficient feature representation with RNN AE and DTW post-processing technique in a way that is much less computationally expensive comparing to the initial approach proposed by Niels in his research, but still remains its main advantage of human-like shape clustering. We apply DTW algorithm not on the full training set, but on the centroids received after initial clustering based on extracted with autoencoder feature vectors.

Relevance of work and the grounds for its implementation. Stylus-based interfaces, that allow quite precise input with similar to real-pen usage experience, continue attracting the interest of scientists and researchers all over the world. The recent evolution of computerized systems and advances in stylus-based technology have improved the current state of the art in studying the components of handwriting. These technologies allow us to analyze the kinematics of handwriting and therefore

extract temporal characteristics such as speed, velocity, jerk, etc, not only spatial characteristics of handwritten shapes.

This evolution has provoked a number of new researches in handwriting analysis sphere that begins with the pioneering work of Plamondon and Shihari in 2000 [17], which continues up to our days. Despite active research in this area, there are still number of open research problems in a handwriting community; a lot of new technological tasks that appears in this sphere determine the relevance of this work.

The purpose and tasks of work. The purpose of the work is to build an expert system for handwritten allograph clustering based on the handwritten input. To achieve this goal, the following tasks were set:

- Investigate and perform a detailed analysis of existing methods for solving allograph clustering problem;
- Create a solution for building an efficient feature representation (build RNN AE for data compression);
- Perform clustering with shallow method; implement DTW-based clusters merging technique for improving initial clustering and getting human-like results.
- Evaluate the quality and compare the clustering results with and without DTW-based clusters merging.

Development tools. We chose Python (3.6 version) for pre- and post-processing, data analysis, feature extraction and clustering. JetBrainsPyCharm 2021.1 (an integrated development environment for the Python programming language), which is free freely distributable and open source IDE, was chosen as the tool for creating the software. For clustering we used the scikit-learn package - a free machine learning library for the Python programming language. For deep learning purposes we used Keras which is an open source neural network library written in Python.

The study was conducted on an open IAM-OnDB and UJI Pen V2 datasets, which contain data in online format.

Possible areas of application.Until recently, the most important area of application of the handwritten allograph styles clustering system was criminology. In

addition, the results were also used in graphology, which uses the characteristics of handwriting to make conclusions about the personality and character of the writer.

Researchers later proved that explicit knowledge of different forms of handwriting can increase the level of recognition [5-6, 18-19]. It is also can be used for writer identification [10], dataset investigation for the presence of various types of characters spelling for improving the diversity of further generation. It is also important to use handwriting clustering to detect mental disorders and other illnesses [20], which can be used for its early diagnosis.

SECTION 1. FORMULATION OF THE PROBLEM AND THE DIFFICULTIES OF ITS SOLUTION

1.1. Allograph Clustering in Handwriting Analysis

Handwriting is one of the most common everyday person's activities.

Determining the style of handwritten text is an important step in the task of text recognition. Systems for optical text/character recognition require calibration in order to work with a specific text font. Thus, the assignment of input data to one of the classes or clusters is a necessary condition for further transition to the problem of optical recognition. Handwriting clustering is also used in verification tasks when it is necessary to correlate the authors of two copies of a document.

An allograph can be defined as a written with a prototypical shape character [10-11].This means that for one fixed letter different types of writing (i.e. allographs) can exist.

Among the toughest challenges in this sphere is making results *congruous to the human*. Most results achieved by researches results do not "match" with the human observer: the most matching allographs chosen by their system are different from the allographs what would be chosen by human.

Defining and labeling of handwriting style is still open research problem in handwriting community. As there are no clear rules for handwriting styles establishing (i.e. clear marking or labeling), this problem does not have a clear single solution. For the same reason, to separate different styles researchers usually choose clustering algorithms that do not require a priori knowledge about labeling and perform separation based on data structure and inner relations.

The main issue of handwriting clustering is how to obtain features (i.e. characteristics) that reflect the style of handwriting. Many approaches have been proposed for different levels: based on strokes, symbols, words, and lines of text. A

good set of features should reflect characteristics that are specific to one cluster, and they should be as different as possible for other clusters.

The very early approaches used handcrafted features engineering [2-4]. In more recent research [1] authors performed a detailed analysis of existing handcrafted feature extraction techniques, and build a system for handwriting styles clustering. However, such approaches, which were initially built for handwriting features extraction, cannot be generalized for other time series clustering tasks (for instance, audio, music, etc.) and are strongly related to the specifics of the particular problem. Moreover, it requires a lot of time to research and construct handcrafted features, and therefore is time consuming.

More recent success in DL spherechanged the situation dramatically. Deep learning approaches are not time-consuming in terms of person-hours. It replaced handcrafted feature engineering, since it is assumed that the model should determine the relations in data all by itself, and construct features only on the basis of the data. Such approaches have made a huge contribution in handwriting analysis sphere.

1.2. Clustering in Scope of Handwriting Analysis Task: Problem Formulation

An unsupervised machine learning algorithm is an algorithm that receives only objects' descriptors or the objects itself as the input and has no tags or labels for ground truth. Such algorithms usually learn patterns from unlabeled data, and therefore this is mostly suitable only for problems in which descriptions of a set of objects (training samples) are unknown, and it is required to detect internal relationships, dependencies, and patterns that exist between objects.

The difference between supervised learning and unsupervised learning is not defined formally and strictly, because there is no objective criterion of what is considered a feature (descriptor) and what is a label (identifier).

Clustering is one of the most common and widely used algorithm in unsupervised learning. Clustering refers to the task of grouping a set of input objects

into subsets (clusters) so that objects from one cluster are more similar to each other than to objects from other clusters by any criterion.

Let us formulate the clustering problem and write it down in a formal way.

Let X – be a finite set of objects (samples, situations, precedents), Y – be a set of labels (that is usually unknown for clustering task). Assume that there are some relation $\rho(x, x')$ on a set X which is called a distance function (metric), and a finite set of objects $X^m = \{x_1, x_2, \dots, x_m\} \subset X$, which is called a training set.

The clustering task is to divide these samples into subsets (so-called clusters), so that each object $x_i \in X^m$ to match the label $y_i \in Y$ in a way that the objects inside each cluster were close to each other in terms of the metric ρ, and objects from different clusters significantly differed.

Then the clustering algorithm is a function $a: X \to Y$, which for each object $x \in X$ find the corresponding identifier (label) $y \in Y$ of the cluster.

In some cases, the set Y is known in advance, but more often the task is to determine the optimal number of clusters, from the point of view of one or another criterion of clustering quality.

In terms of handwritten allograph clustering problem, each sample is a separate handwritten symbol (allograph), whereas the label is symbol's style (or shape) and is initially unknown as well as symbol styles number.

Usually clustering is performed based not on the raw data, (not based on the objects from the dataset themselves). What we consideran object is determined by the specifics of the field of study. In our case, each object is a handwritten character, which represented as a sequence of points. But such representation may be unclear for the model and not may carry out any useful information. Therefore, clustering is usually performed based on the extracted features. It can be done in handcrafted manner, or based on neural network approaches.

In our case, we perform clustering based on neural-generated feature vectors.

A feature is a measurement result of some characteristic of an object. Formally, a feature is a mapping $f: X \to D_f$, where $D_f$is a set of all possible feature

values. Depending on the nature of this set, characteristics are divided into the following types:

- binary feature: $D_f = \{0,1\}$;
- nominal feature: D_f is a finite set;
- ordinal feature: D_f is a finite ordered set;
- quantitative feature: $D_f = \mathbb{R}$.

Let us assume that we have a set of features $f_1, f_2, \dots, f_n$. The vector $(f_1(x), f_2(x), \dots, f_n(x))$ is called the feature vector (description) of the object $x \in X$. In machine learning, no distinction is made between the object and its feature representation; it is assumed that $X = D_{f_1} \times D_{f_2} \times \dots \times D_{f_n}$, and X is called a feature space.

Basically our task is to find (or more accurately, learn) an effective feature representation using deep learning technics and then build a clustering algorithm for handwritten allograph style separation.

1.3. Problems of Handwritten Allograph Styles Clustering

As any other biometric data, handwriting is prone to extreme variation and variability. Variation means the intra-writer difference (i.e. difference in handwriting between different people), whereas variability means the inter-writer one (the ability of one person's handwriting to change over time). The presence of variation and variability in handwriting is mostly explained as depending on emotional, personal and other side factors. Previous researches have also proved that the handwriting style can be significantly different depending on person’s age, gender, geographical location, native language and country, education, profession, etc. [21-24].

However, it is reasonable to assume that the influence of variation is much stronger than the variability, because no matter how the handwriting of a person changes, it will still have the qualities that distinguish it from the handwriting of others.

Allographic variation refers to a phenomenon that each letter can be written by several topologically different ways, such as capitalized, block-printed, and cursive style. Thus, the letter shape can be changed significantly due to this diversity.

The following specific features in handwriting should also be emphasized:

- given the physiological specifics of the text writing process, handwritten text is not standardized and prone to certain distortions; it is significantly individual, and re-written the same sample text is not identical to the previous version of writing;
- handwritten text is more chaotic than structured, it does not have a fixed structure with a set of "parallel" rows or columns of characters;
- connected writing style complicates the task, since we need to perform character segmentation for further allograph clustering, but receiving a good segmentation result is still an open research problem due to the fact that the boundaries of characters are unknown, and, moreover, characters are prone to change their shape depending on the position in the word and neighboring characters.

Because of this inconsistency and vagueness that the handwriting and handwritten allograph style clustering still remains quite complex and, despite the large number of studies conducted, is not a completely solved task.

Among other problems that we had to face was the difficulty in finding suitable datasets, segmented at the character level. Most handwriting databases contains phrase-level samples, and therefore requires additional processing step and performing character segmentation (as in case of IAM-OnDB).

The experiment on two-tier handwriting styles clustering conducted by Korovai and Marchenko [1] demonstrates that, being the product of human activity, the styles of the handwriting obviously cannot be clearly separated from each other, because for any two styles there will always be an "intermediate" one. Therefore, it demonstrates the impossibility of clearly distinguishing the styles of human handwriting, because there are no groups that would not intersect and were clearly separated from each other.

Given all of the above, handwritten allograph styles clustering cloud be a daunting task for computerized systems, even though a person assigns each sample of handwriting to one of the classes or clusters by style automatically.

Note also that the results obtained largely depend on many other factors of a purely technical nature:

- input device characteristics (screen sensitivity and time interval for saving input data – sampling rate);
- restrictions and limitations on characters (whether characters segmented out from connected handwriting are allowed, or input data contains only separately written characters);
- number of classes or categories (only numbers, upper or lower case, only letters, other symbols, etc.).

Let us clarify that data used in our study were not subject to any restrictions on the symbols (they could be both separately written and segmented out from continuously connected handwriting; connectivity (or lack thereof) is one of the characteristics of human handwriting, so for this task, imposing restrictions on characters would lead to artificial, irrelevant results that do not correspond to reality), but we limited the number of categories to uppercase and lowercase letters only, because the number of characters from other categories is not enough to ensure adequate training of the model.

For our research, we combined separately written characters from UJI Pen dataset and segmented out characters from IAM-OnDB dataset that contains handwritten phrases. The more detailed data description can be found in section 2.

Among the other challenges in this domain, making results congruous to the human can be distinguished. Most achieved by researches results do not "match" with the human observer: the most matching allographs chosen by their system are different from the allographs what would be chosen by human. Human-congruousness is very difficult to measure from a metric point of view. How to construct a metric of "beauty" or "human-likeness" is still an open question.

Nevertheless, we propose to use the metrics below in order to analyze the results of clustering.

1.4. Performance Evaluation

Given the ambiguity of the clustering results, it is necessary to determine in advance the metrics for assessing the quality of clustering, on the basis of which the results will be evaluated in the future.

There is no single best criterion for the clustering performance evaluation, but a number of heuristic methods are known and used for such purposes. Most clustering algorithms do not have a clearly defined criterion for measuring a clustering quality, instead theyproduce a fairly reasonable andqualitative clustering "by construction" (in correspondence to cluster definition). Each of them can give different results.

It is common to distinguish three groups of methods of a clustering performance evaluation:

a) external metrics that are based on the comparison of the clustering result with the a priori known division into clusters (with known labeling);
b) internal metrics are based on the information contained in the data itself (without known labeling);
c) relative metrics that are based on the comparison of two different clustering results or clusters.

In this case, we will pay attention to the second group of methods for assessing the quality of clustering - internal metrics, because there is no known division into classes (in this case handwritten allograph style) in our datasets, so we will evaluate only the structure of clustering itself.

Thus, internal metrics for clustering performance evaluation assesses the quality of clustering structure, relying only directly on it, without using any external information (a priori known labels).

Conceptually, internal metrics use the idea of cluster cohesion and cluster separation to assess the quality of clustering results.

The idea of cluster cohesion is that the closer the objects to each other inside the clusters, the better the division into clusters. Thus, it is necessary to minimize the intracluster distance, for example, the with-in sum of squares (WSS).

In the case of cluster separation, the idea is the opposite - the farther apart the objects of different clusters, the better the clustering is. Therefore, it is necessary to maximize the intercluster distance, for example, the between cluster sum of squares (BSS).

By combining these two principles, more complex metrics can be introduced for clustering performance evaluation.

Among the internal metrics we will use in our research for clustering quality evaluation are the following:

a) Calinski-Harabasz index;
b) Davies-Bouldin index;
c) Silhouette coefficient.

1.4.1. Calinski-Harabasz Index

TheCalinski-Harabasz index (which also known as variance ratio criterion) is an internal metric that was introduced in 1974 [25]. Here the compactness is based on the distance from the cluster points to their centroids, and separateness is based on the distance from the cluster centroid to the global centroid.

VRC is an internal metric that can be used for clustering results evaluation. It is basically the ratio of between cluster dispersion mean and the within cluster dispersion, which can be calculated using following formula:

$$VRC = \frac{N-k}{k-1} \times \frac{tr(B_k)}{tr(W_k)}, \tag{1.1}$$

where$N$ is a number of samples in a dataset,

$k$is a number of clusters which initial dataset was clustered into,

$B_k$is a between group dispersion matrix,

$W_k$is a within group dispersion matrix,

$tr(\cdot)$is a matrix trace.

Between group dispersion matrix (B_k) and a within group dispersion matrix (W_k) from (1.1) can be found as:

$$W_k = \sum_{i=1}^{k} \sum_{x \epsilon C_i} (x - \bar{x_i})(x - \bar{x_i})^T, \tag{1.2}$$

$$B_k = \sum_{i=1}^{k} |C_i|(\bar{x_i} - \bar{x})(\bar{x_i} - \bar{x})^T, \tag{1.3}$$

where$k$ is a number of clusters which initial dataset was clustered into,

$|C_i|$is a number of objects in cluster C_i,

$\bar{x_i} = \frac{1}{|C_i|} \sum_{i=1}^{|C_i|} x_i$is a centroid of cluster C_i,

$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$is an average for all samples in dataset.

Among the irrefutable advantages of this index is easy score calculation. It does not require a lot of resources.

It is also easy to evaluate the clustering quality based on this metric. A higher Calinski-Harabasz index score refers to a model with more clearly defined clusters. It is typical for the case when the clusters are dense enough and well separated from each other, which corresponds to the standard cluster concept.

It is also can be used for choosing an optimal number of clusters$k$, and therefore for choosing the best clustering model, which can be done while estimating the VCR for different models with increasing number of clusters$k$ value. Then the

optimal clustering models should be chosen as the model that corresponds to the first local maximum of the Calinski-Harabasz index value [25].

1.4.2. Davies-Bouldin Index

Another important internal metric for clustering results evaluation is Davies-Bouldin index [26]. It was introduced in 1979 by David L. Davies and Donald W. Bouldin

This index denotes the average between clusters "similarity". Here the similarity is considered as a measure that compares the distance between clusters with the size of the clusters themselves. Such measure is computed for each cluster $C_i, i = \overline{1,k}$ and its most similar cluster $C_j, j \neq i$ using following formula:

$$S_{ij} = \frac{\overline{d}_i + \overline{d}_j}{d_{ij}}, \tag{1.4}$$

where $\overline{d}_i$ ($\overline{d}_j$) is the average distance from each point of cluster C_i (C_j) to its centroid, which is also known as cluster diameter,

d_{ij} is distance between C_i and C_j clusters centers.

Then the Davies-Bouldin index can be written as follows:

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j} S_{ij} \tag{1.5}$$

where k is a number of clusters which initial dataset was clustered into,

S_{ij} is a similarity measure for clusters C_i and $C_j, j \neq i$.

These conditions means that the index is defined to be symmetric and non-negative. Unlike the previous metric, where higher scores corresponds to better clustering, here a lower Davies-Bouldin index corresponds to a better model, basically with better between-cluster separation. This index has a lower bound: zero is the lowest possible score. Values close to zero indicate better separation, and therefore better clustering.

This reasoning is based on the idea that no clusters should be alike (which is basically means that they should be clearly separated from each other), and therefore a better clustering scheme will essentially minimize the Davis-Bouldin index down to zero.

1.4.3. Silhouette Coefficient

The silhouette coefficient value [27] shows how similar the object is to the object in its cluster compared to other clusters. The score for the whole cluster structure can be calculated by the following formula:

$$Sil\ (C) = \frac{1}{N} \sum_{c_k \in C} \sum_{x_i \in C_k} \frac{b(x_i, c_k) - a(x_i, c_k)}{\max\{a(x_i, c_k), b(x_i, c_k)\}}, \tag{1.6}$$

where$a(x_i, c_k)$is an average distance from$x_i \in c_k$to other objects from clusterc_k (cluster compactness),

$b(x_i, c_k)$is an average distance from$x_i \in c_k$to the objects from other clustersc_l: $k \neq l$ (cluster separateness).

These indicators are usually calculated by the following formulas:

$$a(x_i, c_k) = \frac{1}{|c_k|} \sum_{x_j \in c_k} \|x_i - x_j\|,$$

$$b(x_i, c_k) = \min_{c_l \in C \setminus \{c_k\}} \left\{ \frac{1}{|c_l|} \sum_{x_j \in c_l} \|x_i - x_j\| \right\}.$$

But the silhouette coefficient actually can be calculated from any distance metric depending on a specific problem.

Among main advantages of silhouette coefficient is that it is bounded:

$$-1 \leq Sil(C) \leq 1.$$

Moreover, the closer this estimate is to 1, the better the clustering is. The higher score of the silhouette coefficient refers to the model with more clearly defined clusters. The higher score usually occurs when the clusters in the resulting clustering scheme are dense and well separated, which corresponds to the standard cluster concept.

SECTION 2. DATA ANALYSIS AND PROCESSING

2.1. Data Description

Our research is based on online handwriting analysis, therefore it requires data in an online format. How to present online handwritten data in a suitable and informative for a model way was an open research topic for a long time, and, despite the success in this area, the question of whether it is possible to create an even more suitable representation is still open.

Usually online handwriting can be represented as a sequence of points, that contains information about the trajectory of the pen movement (both x and y coordinates), pen pressure p and other optional characteristics (such as z coordinate, grip angle, etc.) which are mostly specified for a certain types of problems (i.e. z coordinate for stability test on certain point in Parkinson HW dataset or for 3-dimensional handwriting recognition [28]).

Taking into account this format of data presentation, it is possible to characterize the style of handwritten characters not only based on spatial features (as in case of offline data), but also the dynamics of the writing process (i.e. speed, acceleration and jerk).

We can also transform such data and represent it as a time series. With this in mind, the data in general can be written as a sequence of three temporal functions:

$$[\boldsymbol{x(t)}, \boldsymbol{y(t)}, \boldsymbol{p(t)}]$$

These functions represent the trajectory of the pen movement (x and y coordinates) and the air-state of the pen. In most related works, the pen state is defined as follows:

- 0 - if the pen touches the screen of the device;
- 1 – if the pen is detached from the screen.

We find it this is not very representative of our model. Most characters are written in 1-2 strokes, therefore such a time series is very sparse, since there are only 1-2 occurrence of pen state equals to 1 in such time series. So in this study we propose to use the stroke index instead, and therefore we get a non-sparse representation of pen state.In some datasets, information about the time function $p(t)$ is not explicitly represented, but can be restored if necessary.

It is noteworthy, that for our research we do not require any additional information about grip angle, pen pressure value, etc., and we conducted this study based only on these 3 temporal function.

In our case, a separate characters are considered, so we need either character-based dataset, or dataset with lines segmented into separate characters. Segmentation task is not obvious since the boundaries of characters are unknown, characters are prone to change their shape depending on the position in the word and neighboring characters, and stroke ordering could be messed depending on a specific handwriting style.

Each character consists of one or more strokes. Stroke is can be defined as a minimal unit of the electronic path. It is a separate structure that the continuous sequences of points represented by the coordinates x and y can be grouped into. In other words, a stroke is a sequence of points (x, y, p) with position (x, y), pen state value p at a timestamp t.

Thus, stroke can be written down as follows:

$$s = \{x(t_i), y(t_i), p(t_i)\}_{i=start}^{end},$$

where$p(t_i)$is a stroke index, $i = \overline{start, end - 1}$.

And therefore we can define handwritten allograph as below:

$$A = \{\{x(t_i), y(t_i), stoke\_id\}_{i=0}^{NP_{stroke\_id}}\}_{stroke\_id=0}^{NS},$$

whereNS is a number of strokes in allograph A and NP_i is a number of points in i-th stroke.

We will also store the label (i.e. symbol ASCII code) for each sample in a dataset. We do not require any other information in this study (for instance, about writer age, gender, education, native language, etc.).

2.2. Datasets Overview

To begin our research, it is necessary to find an appropriate dataset. In this subsection, we performed a detailed analysis of existing online handwriting databases. We focused on the following two databases: IAM-OnDB[29-31] and UJI Pen [32]. After thoughtful analysis, it was decided to combine these datasets for further experiments.

IAM-OnDB, as most handwriting databases, contains phase-level samples, and therefore requires additional processing step and performing character segmentation. Still, it is one of the most common and popular datasets for handwriting analysis, that contains more than 86 000 handwritten word samples with high variation.

Segmentation process has its own drawbacks. We used solution by MyScript order to get segmented characters from this database [32]. However, this results could not be compared to a dataset collected specially on a characterlevel, as UJI Pen dataset.

Taking into account all the above, we decided to run experiments both on UJI Pen and IAM-OnDB datasets and compare the results based on internal metrics described in section 1.

More detailed dataset descriptions are below.

2.2.1. IAM-OnDB

One of the datasets with the above structure is the IAM On-Line Handwriting Database (IAM-OnDB) [29-30]. It is an open dataset containing forms of handwritten English text. IAM-OnDB is usually used to train and test handwriting recognition

systems, in writer identification and verification tasks. It can also be used for handwriting age, gender, nationality classification, etc., since it also stores additional information about writers.

IAM-OnDB consists of handwritten lines collected from 221 different authors using a "smart board" and a digital pen, and is a set of data in handwritten lines of text (Fig. 2.1).

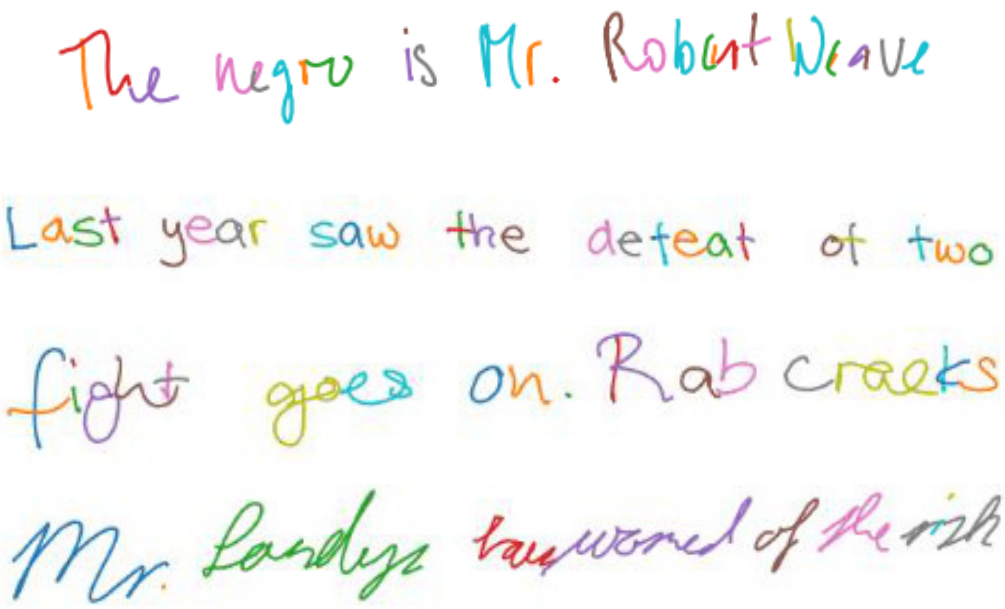


Figure 2.1 –IAM-OnDB handwritten samples example

Participants were asked to write fragments of text from the Lancaster-Oslo-Bergen text corpus [31] in order to collect handwritten data. The LOB body contains 500 samples of English text, each consisting of about 2,000 words. These texts are quite diverse and are divided into 15 categories: from the press to the scientific literature. To obtain a database of handwritten text samples, this corpora was divided into fragments of approximately 50 words each that were then offered to participants for writing on the board.

In total, the IAM-OnDBcontains [29]:

- more than 1700 manuscripts collected from 221 participants;
- 11059 unique words (dictionary size);
- 86 272 spelling variants (handwritten words samples);

- 13049 labeled lines of text.

In addition to the recorded data (handwritten traces) and the corresponding text label, IAM-OnDB also stores some information about the participants that may be useful for further work and researches. For each person who participated in the collection of the dataset, information is stored about:

- native language and country;
- age;
- gender;
- left- or right-handed;
- education;
- profession;
- other languages person speaks.

This information is usually used for age and/or gender classification and factors of influence on handwriting changes.

All participants, samples of writing which were entered into the database, were volunteers. Most of them are students and staff of the University of Bern.

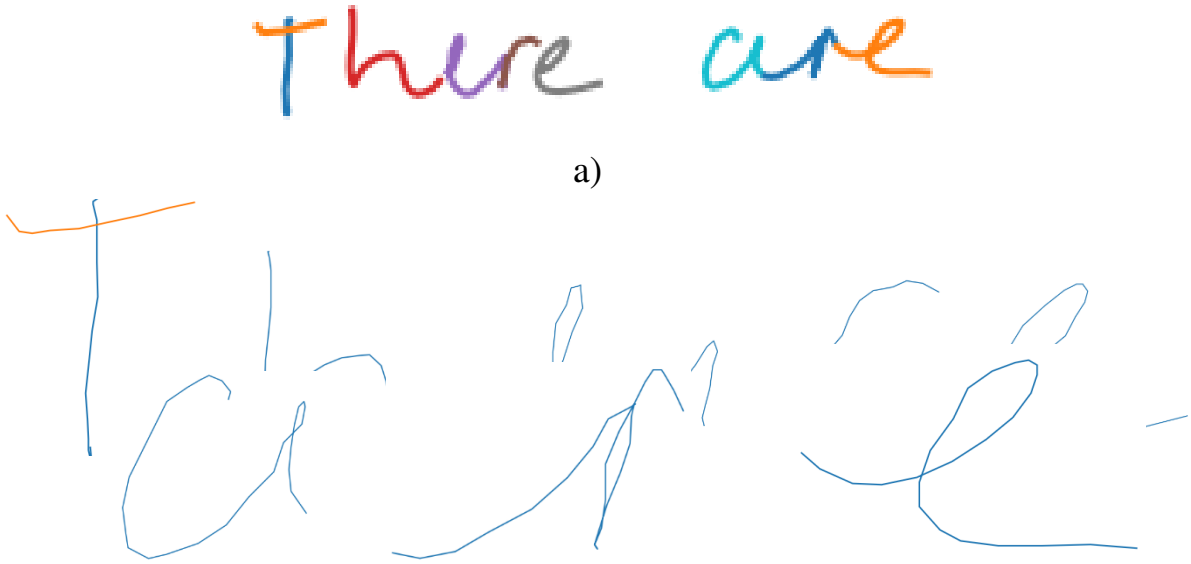
It is worth noting that among the participants in the collection of samples IAM-OnDB were people with different education (both technical and humanitarian). In addition, both men and women participated in the dataset collection (both sexes are represented in the dataset approximately equally), left-handers and right-handers (about 10% of dataset participants were left-handed) and people of different ages (average age of about 25 years).These factors provided the high variation available in the resulting dataset.

IAM-OnDB does not have any label that allows us to extract separate characters from the dataset, so we additionally used a segmentation solution designed by MyScript [33].

An example of segmentation results can be found on fig. 2.2.

As a result, we have 84 355 separate characters after performing segmentation. We need to admit, that the quality of the resulting dataset is worse than

the quality of UJI Pen Dataset, since the second one was designed and collected for single characters from the beginning, while segmented out characters can be slightly corrupted due to segmentation model error. Also this dataset is highly imbalanced (with about 10 000 of samples of ‘e’ character and only 12 of ‘Z’), while UJI Pen does not have such a disadvantage.



a)

b)

Figure 2.2 – Characters segmentation: example: a) handwritten sample “There are” from IAM-OnDB dataset before segmentation; b) handwritten sample “There are” from IAM-OnDB dataset after segmentation.

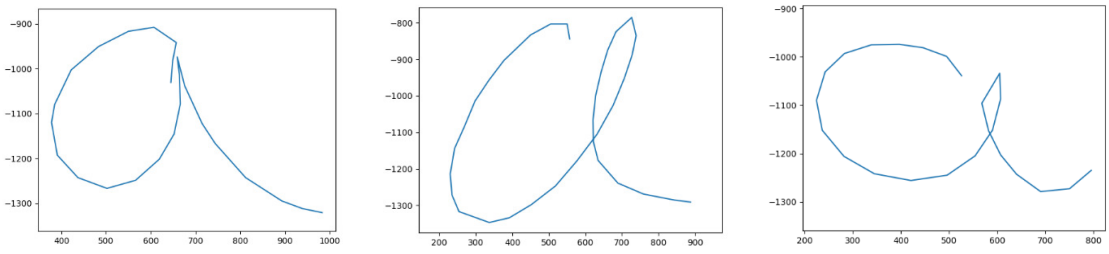
2.2.2. UJI Pen Characters Data Set

UJI Pen Characters Data Set is a pen-based database with more than 11 000 isolated handwritten characters [32]. We chose the second version of UJI Pen Data Set since the original version contains handwritten samples from only 11 writers (2 repetitions from each participant), and is extremely small for deep learning approaches.

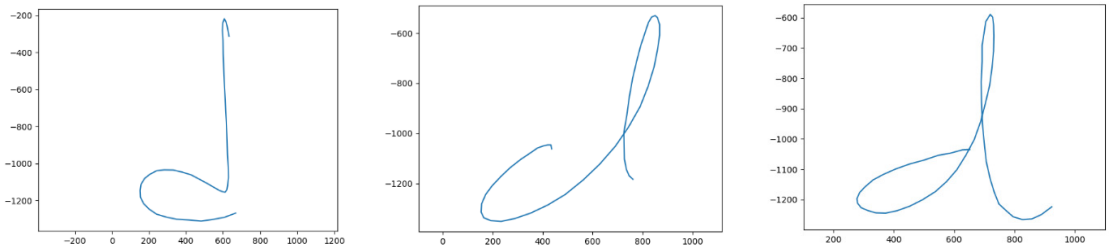
UJI Pen V2 dataset contains 7440 handwritten samples collected from 60 writers (2 repetitions from each participant as well). It contain 97 different isolated characters with exactly 120 handwritten samples for each character. A complete dataset lexicon is:

- 66 letters (33 lowercase, 33 uppercase): 52 ASCII letters,14 non-ASCII letters;
- 10 digits ([0-9]);
- 21 other characters (special symbols, etc.).

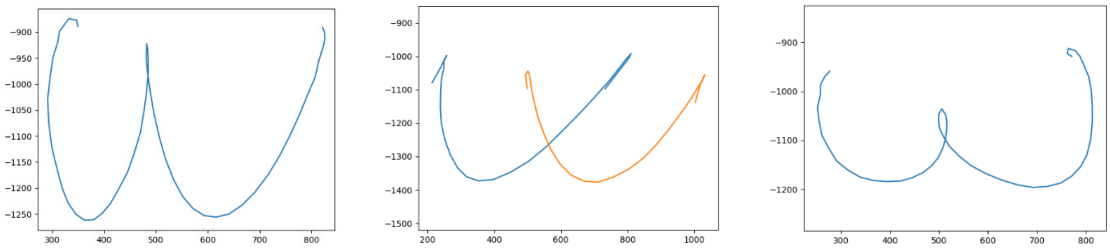
UJI Pen Dataset stores label, strokes number for each char, and points sequence, it does not contain any additional information about writers (as in case of IAM-OnDB), as well as information about timing and pressure level value.A typical examples of handwritten characters from UJI Pen Characters V2 Dataset can be found on fig. 2.3.



a)



b)



c)

Figure 2.3 – UJI Pen Characters Version 2 Data Set handwritten samples examples.

Each participant was proposed to write down a full set of characters in two non-consecutive sessions. A program for samples collection displayed a set of boxes on the screen (one box for each character required for collecting), and all the participants were told to write inside these boxesonly [32]. Writers were instructed about rules and principles of experiment. They had an opportunity to clear the content of the corresponding box if they made a mistake or were unsatisfied with the results. All the data were collected with Toshiba Portege M400 Tablet PC with use of its wireless stylus [32].

2.3. Data Processing

Raw data is not customary to immediately submit to neural networks for training. Data needs processing before being submitted to the model in order to keep the system as flexible as possible with regard to differences in the writing surface.

Our preprocessing consists of 3 important steps:

1. Normalization;
2. Resampling;
3. Getting first derivatives for time series representing the trajectory of pen movement.

2.3.1. Normalization

The most common step that can be done is normalization, which (as it was already shown in a number of related researches) gives more accurate results in the tasks of handwriting recognition, authentication, etc. In particular, it improves the quality of handwriting classification.

To make the intermediate representation invariant to scale movement and distortion, the characters are centered and scaled. In particular, the origin is set in the center of the symbol and the characters are scaled with a factor of:

$$\delta_y = y_{max} - y_{min}.$$

Then we can move on to the new coordinates by the following formulas:

$$x_i = \frac{x_i - x_{mean}}{\delta_y},$$

$$y_i = \frac{y_i - y_{mean}}{\delta_y}.$$

Note that here we scale characters only by their height. The reason is in such way we do not corrupt the shape of the characters.

Thus, the input character normalized to a centroid position of $(0, 0)$.

Note also that after performing the transformations described above, no information is lost about the angle of the handwritten text, curvature in dots, degree of coherence when writing, the ratio of height to width of characters and other important characteristics, because these transformations do not deform the text.

After performing normalization, the data are homogeneous: all of the samples are in the same unit.

2.3.2. Resampling

Our preprocessing algorithm also includes resampling step.This allows us to remove the influence of different sampling rates, which may occur due to different input devices used for data collection.

Tan et al in their paper on automatic character prototypes based writer identification for online handwritten documents [10] resampled all the characters to 30 points exactly. Schomaker et al demonstrated that 30 points is exactly enough for

describing most characters in western handwriting, with a worst case of 5 points per stroke in a 6-stroke character (which is rather rare to observe) [11].

We perform a simple downsampling technique to get evenly distributed over time point sequences (in contrast to the approaches of other researchers, where the data is processed in such a way as to obtain points evenly distributed in space, i.e. equidistant linear resampling [34]). This was done for reasons that we need to preserve both spatial and temporal characteristics and save as much raw information as we can. A similar approach was used by Ahrabian et al in their paper for online handwritten signature verification [35]. They fixed a sampling rate R and iterated through the data in chronological order performing the following steps:

1. Add point to resulting sample.
2. Skip $R - 1$ points.

As result, new handwritten sample contains $\left\lfloor \frac{L}{R} \right\rfloor$ points, where L is a number of points in initial sample. However, in this case, we get a different number of points for each sample. Instead of this, we generate a one-dimensional array of the specified number of elements, the values of which are evenly distributed within the specified interval ($[0; L]$), and cast all the values to integer, receiving by this a list of indexes of points to keep.

2.3.3. Time Series Derivatives Calculation

The use of derivatives in the classification/clustering (as well as in other handwriting-related tasks as recognition, segmentation, etc.) of time series is not new to a handwriting community. In number of works devoted to handwriting recognition etc. it was shown that for handwriting processing with recurrent neural networks, it is much more informative for the model to get not the exact point coordinates, but the difference of two consequence points. This is due to the fact that the model is thus not tied to the position of a certain feature, but to its actual presence.

The derivative determines the general form of the function, not the function value at the actual point itself. The derivative shows what is happening in the neighboring points. When talking about time series, this basically means that the derivative function also considers time series behavior before and after a certain point in time [36].

In our research, we performed a similar step and get the first derivatives for both $x(t)$ and $y(t)$ series.

We compute the first derivatives in a following way:

$$\dot{x}(t) = \frac{x(t+1) - x(t)}{\Delta t},$$

$$\dot{y}(t) = \frac{y(t+1) - y(t)}{\Delta t}.$$

Note that the points in the input sequence are evenly spaced in time, and therefore the value of Δt is constant for each point and therefore this value can be neglected.

SECTION 3. DEEP LEARNING BASED FEATURE ENGINEERING AND CLUSTERING

Along with the evolution of computerized systems, autoencoders (AE) have been proposed to solve many machine learning and deep learning problems. It found its application in signal processing, computer vision and many other different tasks, since it can be applied for both offline data (for example, images) and online data (time series). AE-based approaches have made a huge contribution in handwriting domain: it found its application in writer identification and verification tasks [35], handwriting recognition [6, 34] and generation [12], clustering and feature extraction [15], etc.

This allows the use of AEs for the feature extraction, which is one of their most traditional ways of usage. It is noteworthy that the feature extraction and further selection of an optimal subset of features is the most important step in solving problems with big data and processing complex data with multidimensional attributes. In addition, feature selection is a valuable technique for performing classification and forecasting [1].

Autoencoders allows to perform these steps in a machine-automated way. It does not require any handcrafted feature engineering and further feature space analysis, since it is assumed that the model compress all the data and build and efficient feature representation all by itself based on data structure and inner relations only.

However, despite the remarkable performance, features derived using deep neural network models are difficult to explain due to models' so-called black box nature.

3.1. Preliminary: Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a powerful deep learning algorithm for sequential data (such as online handwriting, speech, text, etc.) analysis. *In our research, we use RNN autoencoder to build an efficient feature representation of handwritten allograph samples for its further styles clustering.* We assumed that the model should determine the relations in data all by itself, and construct features only on the basis of the data, without knowing any additional information about allograph styles tagging.

RNN is a kind of neural networks where connections between elements form a directed temporal sequence [37]. This makes it possible to process a series of events in time or successive spatial chains. RNN usesits internal state (which is called memory) to process input data sequences of variable length. RNN found its application in handwriting and speech recognition tasks, natural language processing and other tasks where the input data can be represented as a time series.

RNN maintains a hidden state h which works as a memory cell that contains the previously observed context. One of the most common difficulties in training RNN models is making the model to capture long-term dependencies, which usually appears in case of long length input sequences. In cases where the "distance" from the place of occurrence of the necessary information to the place of its application is small, the RNNs can successfully use this information. Unfortunately, with an increase in this "distance", the RNN loses the ability to use such information.

To solve this issue, we will use LSTM architecture that allows to address vanishing gradient problem by letting gradients flow unchanged. LSTMs were introduced by Hochreiter and Schmidhuber in 1997 [38], and then optimized and popularized in many subsequent works.

All the RNNs can be represented as a sequence (chain) of repeating and connectedmodules. The standard RNN’s repeating module has a very basic structure(usually a single layer with hyperbolic tangent as activation function).An LSTM unit is a similar chain, but its repeating module has a different and more complex structure. It contains four layers that interact in a special way (in contrast to one layer in basic RNN unit).

A common LSTM unit consists of [39]:

- a cell;
- an input gate;
- an output gate;
- a forget gate.

This allows LSTM to extract input sequence long-term dependencies as well as short-term ones.

LSTM cell is aimed to store the information about the context and remember values from an arbitrary interval (so-called memory), while gates regulates the flow of information in cell. LSTM cell architecture is presented at fig. 3.1.

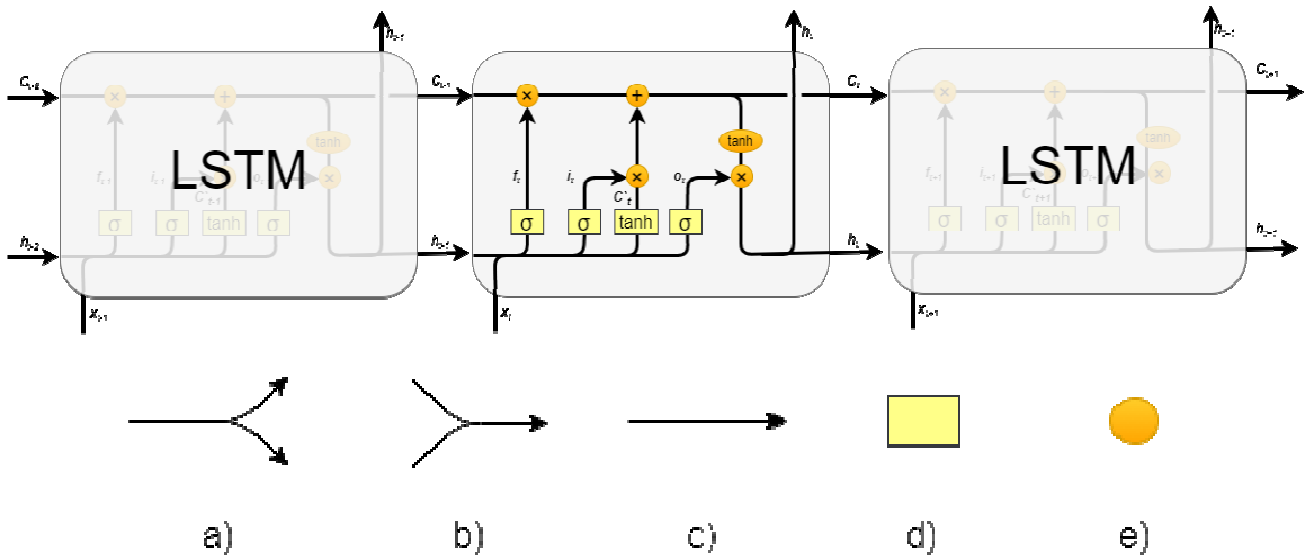


Figure 3.1 – LSTM cell architecture: a) copy operation; b) concatenation; c) vector transfer; d) neural network layer; e) pointwise operation [39].

In the above LSTM cell diagram, each line denotes the vector transfer from the output of one node to the inputs of others. Circles represent pointwise operations (such as vector addition or multiplication), and yellow rectangles represent trained layers. Merging of lines implies concatenation, and branching of a line indicates that information is copied, and then copies are sent to different destination points. The sigmoid layer outputs maps all the input values from zero to one and therefore describes how much of each component let through (0 means that no information

should be flown through the specific gate, 1 means that all the information should be let through the gate) [39].

The compact representation of the calculations for the forward pass of an LSTM unit is as follows:

$$f_t = \sigma\big(W_f x_t + U_f h_{t-1} + b_f\big),$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i),$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o),$$
$$C'_t = tanh(W_c x_t + U_c h_{t-1} + b_c),$$
$$C_t = f_t \circ C_{t-1} + i_t \circ C'_t,$$
$$h_t = o_t \circ tanh(C_t),$$

where$x_t \in \mathbb{R}^d$ is an input vector for the LSTM unit.

$f_t \in \mathbb{R}^h$is a forget gate activation vector,

$i_t \in \mathbb{R}^h$is an input gate activation vector,

$o_t \in \mathbb{R}^h$is an output gate activation vector,

$h_t \in \mathbb{R}^h$is a hidden state vector (LSTM unit output vector),

$C'_t \in \mathbb{R}^h$is a cell state input activation vector,

$C_t \in \mathbb{R}^h$is a cell state vector,

$W \in \mathbb{R}^{d \times h}andU \in \mathbb{R}^{h \times h}$ are weight matrixes,

$b \in \mathbb{R}^h$is a bias vector,

$h$and$d$ are number of hidden units and number of feature inputs respectively.

3.2. Recurrent Neural NetworkBased Autoencoder

Autoencoder is a special architecture of artificial neural networks for unsupervised learning algorithms trained to set target values equal to input values (as much as it possible). It can be used for learning an efficient data representation (encoding).

It basically consists of three layers: input, latent and output layers. Latent layer is also called a code, since it contains an encoded information about the input

data. The main idea of AE is getting the response on the output layer that is closest to the input, and its structure usually consists of only one hidden layer, with the input and output layers being the same size. To make the solution not trivial, restrictions are imposed on the intermediate (latent) layer of the AE: the intermediate layer must be less dimensioned than the input and output layers.This allows to use autoencoders for feature extraction, and forces AE to learn the most efficient representation of the input data, which can be considered as a feature vector. The presence of encoding vector with lower dimension distinguishes the conventional LSTM network for reconstruction only from the LSTM autoencoder (which can be used for data compression).

The two main components of an autoencoder are two neural networks, one called an encoder and the other called a decoder.

The task of the encoder is to build a compressed low-dimensional representation for the input data. In our case, each input sample is represented by 30 points, and we aim to compress this data into 8-dimensional feature space.

The decoder is a reflection of the encoder and is designed to recover the original data as accurately as possible. During the learning process, the decoder stimulates the formation of the most informative compressed representation. The more closely the reconstructed input matches the original, the better the compressed representation is.

A very basic scheme of AE is presented on a fig. 3.2.

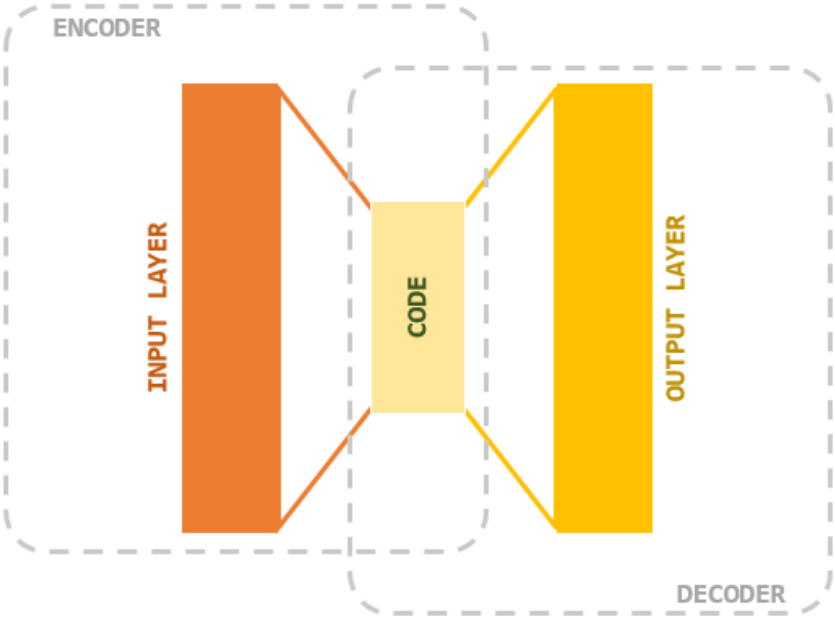


Figure 3.2 – AE basic scheme.

Formally it can be written down as follows. Let assume we have an encoder f and a decoder g. AE encodes the input x into a code (latent representation) $f(x)$ with encoder f. Then the code $f(x)$ can be decoded by decoder g as the output $g(f(x))$. The goal is to minimize the difference between the input data x and the output reconstructed by decoder from the code $g(f(x))$.

At the heart of ourautoencoder is a long-short term memory (LSTM) unit, inctroduced by Hochreiter and Schmidhuber[38] that was described in details in the previous paragraph. We adopted an LSTM AE in order to extract handwriting features with considering a temporal correlation over time. Combining basic autoencoder scheme with LSTM allows us to find the inner relations in sequential data (handwritten trace) and understand its structure using LSTM, and then perform feature extraction using autoencoders to recreate the input sequence.

The LSTM encoder iterates over the input sequence (i.e. handwritten trace) one token (i.e. point) at a time, outputting an "output" vector and a "latent state" vector at each timestamp. Then the hidden state vector is transferred to the next timestamp. Decoder uses encoder context vectors and internal hidden states to generate the next point in the output sequence.

Both encoder and decoder models are jointly trained. When the model reaches the desired performance level while performing the sequence reconstruction, the decoder can be removed, leaving only the encoder part. After that, encoder can be used to encode input sequences and build a low-dimensional learned representation.

In this way, we consider both spatial and temporal characteristics of the handwritten trace.

Taking into account all the mentioned above, we can formulate and write down two main advantages of LSTM autoencoder:

1. as a basic autoencoder scheme, LSTM AE also works as an unsupervised learning method to extract handwritten trace characteristics, and therefore similar inputs will have similar compressed representations;
2. LSTM AE can learn and memorize complex temporal information in data.

We use convenient LSTM AE architecture for learning latent representations. LSTM AE architecture used in our research is presented at fig. 3.3. Model specification and all the details can be found in section 4.

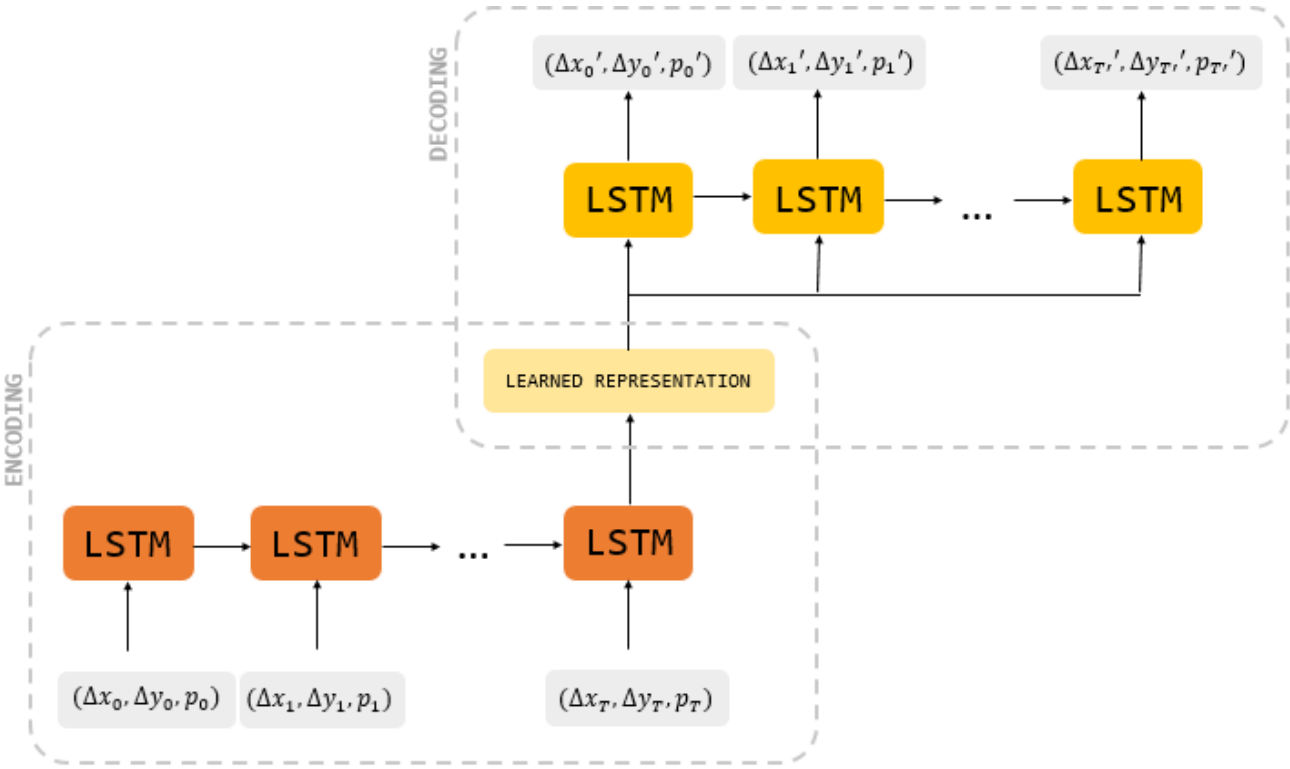


Figure 3.3 –Autoencoder architecture scheme for allograph feature extraction.

3.3. Handwritten Allograph Clustering Based On Deep Learning Featureswith DTW-Based Centroids Merge

Having trained RNN AE model described above, we use encoder model to build a compressed representation for each sample in our dataset. In such way we can perform deep feature extraction, and therefore we consider this representation as a feature vector. Such method can easily replace our previous handcrafted feature based approach [1].

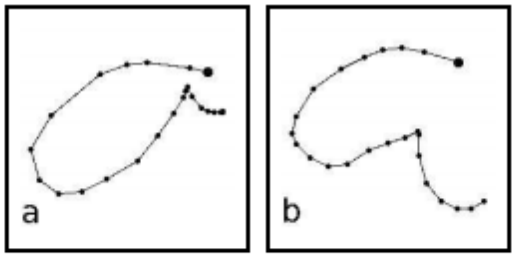
A number of previous approaches uses shallow clustering algorithms to perform handwritten allograph style/shape clustering [1-4, 10-11], for instance, K-means, hierarchical clustering etc. Instead, the researches focused more on finding a similarity measure for handwritten traces. Some of them tried to build an efficient feature representation using handcrafted or automated feature engineering (as we did) to map input traces into a low-dimensional informative representation with Euclidian distance, while others proposed the use of other complex algorithms for distance calculation.

However, making results congruous to the human is among the toughest challenges in handwriting allograph clustering task. Most solutions proposed by researchers do not "match" with the human observer. For instance, Vuurpijl and Schomakerproposed a hierarchical clustering (HCLUS) approach [2], that later was user for character recognition with 96% accuracy on UNIPEN database [40]. However, in case of allograph search and categorization in many occasions the results (a list of best-matching allographs for a fixed allograph) are not what the experts expect from the system.

Niels and Vuurpijl used DTW for so-called “intuitive” character recognition based on visual perception [41-42]. Later Niels et al.[10] proposed to use DTW (dynamic time warping) algorithm to tackle the problem for so-called human-congruous allograph matching for writer identification.They improved hierarchical allograph clustering proposed by Vuurpijl and Schomaker[2] by introducing such method for similarity calculation. However, DTW algorithm is computationally

expensive (it has quadratic time complexity), and therefore cannot be applied for allograph categorization in case of large database. Moreover, covering all the variations in handwriting is impossible, since such training process is highly sensitive to the absence of specific types of character writing (prototypes) in the initial training set; thus, having seen on the test an allograph that was not in the training set, we will not be able to assign this sample to one of the clusters.

Various implementations of DTW was described by Vuori [5]. In our research, we applied proposed by Niels et al. DTW approach as post-processing step after performing initial clustering with shallow algorithm (K-means) based on features extracted with recurrent neural network autoencoder.We use DTW algorithm not on the full training set, but on the centroids of the clustering results achieved by K-means clustering algorithm. DTW algorithm is considered to be appropriate for human-like allographs matching. The main advantage of our method that can be seen at this point is combining RNN AE and DTW in a way that is much less computationally expensive comparing to initial approach proposed by Niels, but with preserving its main advantage of human-congruous allograph clustering.



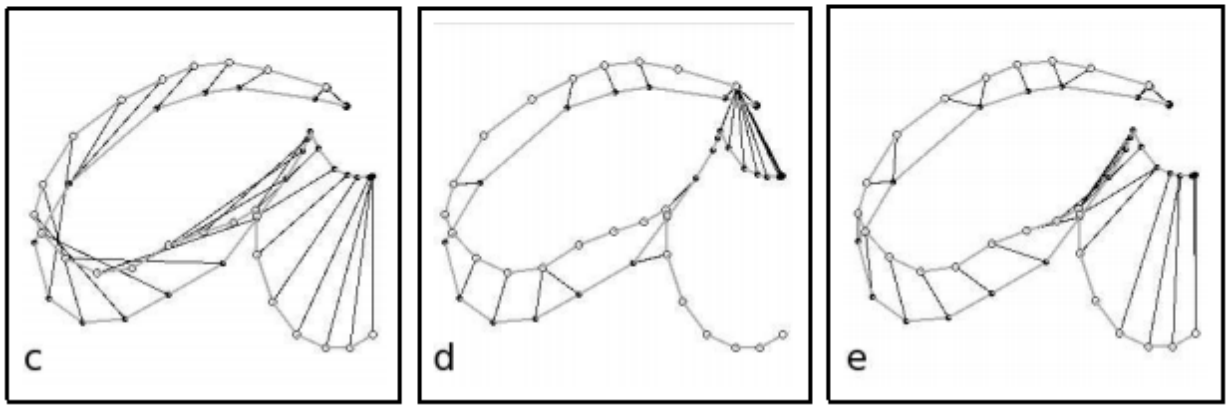


Figure 3.4 Trace matching approaches examples: a-b) the allographs to match; c) results of linear matching (index-by-index points matching); d) results of complete matching (matching to the nearest neighbor); e) DTW matching [10].

Before we continue with experiments and evaluation, we need to describe DTW algorithm in order to understand its main idea.

The ***Dynamic Time Warping (DTW)*** algorithm is an algorithm that allow finding the optimal match between time sequences which may significantly vary in speed [43]. It was introduced in late 60s [44], and then it was investigated and popularized in many spheres. First used in speech recognition task in 1970s [45-46], eventually it found its applications in other areas where the input data can be represented as a time series (audio and handwriting processing, etc.).Time series is a widespread data type, which can be found in almost any scientific field, and comparing two sequences is a common, but not a trivial task.

Measuring the distance between two time series is necessary in order to determine their similarity for further comparing, which is required in solving many different tasks such as clustering, classification, identification, etc. The Euclidean metric is usually considered to be an efficient measurement. For two time sequences, The Euclidian metric is the sum of the squared distances from each n-th point of one sequence to the n-th point of the other. However, the use of Euclidean distance has a significant drawback: if two time series are the same, but one of them is slightly displaced in time (along the time axis), then the Euclidean metric may consider that the series differs significantly. It also requires an additional resampling in order to get sequences with equal length. The DTW algorithm was introduced in order to overcome this disadvantage and provide a visual measurement of the distance between time series, without paying attention to both global and local shifts on the timeline. A typical example of comparing Euclidean and DTW matchings can be found at fig. 3.4.

The idea of DTW algorithm is to "warp" the sequences non-linearly in the time dimension for measuring their similarity regardless of some non-linear changes

in the time dimension. It results in a discrete matching between the nodes of two input sequences.

Generally, DTW algorithm is used to findan optimal match for two input time series (sequences s_1 and s_2) with the following restrictions:

1. each index from the sequence s_1 must be matched with at least one index (or more) from the sequence s_2, and vice versa;
2. the first index from the sequences $s_1$must corresponds to the first index from the sequence s_2 (it is not required to be its only match);
3. the last index from the sequences $s_1$must corresponds to the last index from the sequence s_2 (it is not required to be its only match as well).
4. the indices mapping from s_1 to s_2(and from s_2 to s_1) must be monotonically increasing: if $j > i$ are indices from s_1, then there must not be two indices $l > k$ in $s_2$such that i-th point in sequence $s_1$corresponds to l-th point in sequence s_2 and j-th point in sequence s_1 corresponds to k-th point in sequence s_2 (and vice versa).

A basic DTW algorithm is below [47].

Consider two time series (sequences):

$$Q = q_1, q_2, \dots, q_i, \dots, q_n,$$
$$R = r_1, r_2, \dots, r_i, \dots r_m.$$

The algorithm can be divided into 3 main stages:

1. Building a distance matrix $d \in \mathbb{R}^{n \times m}$, where $d_{i,j} = d(q_i, r_j)$ is a difference between two points q_i and r_j. A common metrics to use for distance calculation are Euclidean and Manhattan distances. Each element (i, j) of the distance matrix corresponds to an alignment between the points $q_i$and r_j, i.e. the cost of their matching.
2. Building a matrix of transformations (deformations) D. Each element of a deformation matrix is calculated based on the following formula:

$$D_{i,j} = d_{i,j} + \min\left(D_{i-1,j}, D_{i,j-1}, D_{i,j}\right).$$

3. Building an optimal transformation (deformation) path and calculation of the DTW distance (path cost). The transformation path W is acontiguous set of elements of a matrix that maps the initial sequences $Q$and R. $W$is a path that minimizes the total distance between Q and R by finding the best possible matching between the sequences' elements. The k-th element of the path W is defined as follows:

$$w_k = (i,j)_k, d(w_k) = d_{i,j} = d(q_i, r_j).$$

When a transformation path $W$is:

$$W = w_1, w_2, \dots, w_K, \text{ where } max(n,m) \leq K < n + m.$$

The transformation path must meet the following requirements:

- Boundary condition: the first element of the transformation pathis $w_1 = (1,1)$ and the last element is$w_K = (n,m)$. This ensures that the transformation path contains all points of both time series.
- Continuity: let$w_k = \left(w_i, w_j\right), w_{k+1} = \left(w_{i+1}, w_{j+1}\right)$ – two consequent elements of transformation path W. Then $w_i - w_{i+1} \leq 1$ and$w_j - w_{j+1} \leq 1$. This ensures that the transformation path moves one step at a time.
- Monotonicity: let $w_k = \left(w_i, w_j\right), w_{k-1} = \left(w_{i-1}, w_{j-1}\right)$ – two consequent elements of transformation path W. Then $w_i - w_{i-1} \geq 0$ and$w_j - w_{j-1} \geq 0$. This ensures that the transformation path does not go back to the previous points, i.e. both indices i and j either remain constant or increase (but never decrease).

Although there are a large number of transformation paths that satisfy all of the above conditions, we are only interested in the path that minimizes the DTW distance (path cost), which can be calculated by following formula:

$$DTW(Q,R) = min\left\{\frac{\sum_{k=1}^{K} d(w_k)}{K}\right\}.$$

Finding a path that satisfies all conditions and minimizes DTW cost can be done using a dynamic algorithm. Hence the spatial and temporal complexity

of the algorithm is quadratic:$O(nm)$, since the DTW algorithm must examine each cell of the transformation matrix.

As it was already mentioned, DTW algorithm is computationally expensive, which is its main disadvantages. However, it can be used for asserting a human-like allographs matching results. To combat DTW disadvantage, we combine it with RNN AE and apply the algorithm for a relatively small amount of handwritten data instead of full training set.

Although DTW algorithm allows perform allograph matching in human-like manner, it also has its drawbacks from a practical point of view besides its high computational cost. In our case, for the DTW-based clusters merging proposed above, we need to establish the threshold value. We should note that this is still an open question. If a threshold value is too small, all clusters will be eventually merged into one cluster, if it is too large, no pair of clusters will be merged and we keep the initial results provided by K-means algorithm. We also assume that threshold value should be established for each character as its own. However, it requires an additional investigation.

More details about hyperparameters to choose forsolving this problem, as well as all the results and experiments are provided in the section 4.

SECTION 4. EXPERIMENTS AND EVALUATION

4.1. Experiments Setup

As it was discussed previously, we conducted our research on two datasets: IAM-OnDB and UJI Pen V2 Character Dataset that were described previously in section 2. They differs significantly by its sizes, nature and application area. We should note that dataset UJI Pen V2 Character Dataset suits our purposes much more, because it was originally created based on separate characters, whereas IAM-OnDB required an additional processing stepin the form of applying segmentation technique. We used solution designed by MyScript [33] for IAM-OnDB segmentation in order to get a separate characters. Thus, the nature of these characters can be considered artificial. Despite this, we ran experiments on both datasets to compare and evaluate the results.

In paragraph 2.3, we described in details our data processing. We performed normalization (scale and shift), got a time series derivatives for both $x$and y. We also used resampling in order to get a 30-point allograph representation, so we have $T = T' = 30$ timestamps (fig. 3.3) for both encoder and decoder in total. At each timestamp a 3-dimensional vector is fed to the encoder: $(\Delta x_i, \Delta y_i, p_i), i \in [0, 30)$, getting 8-dimensional vector (hidden state) as an output. The last timestamp's output vector is considered a code (learned representation). For the decoding process, we feed a code vector as an input for each timestamp. In short, we have the following configuration:

- Input dimension $(\Delta x, \Delta y, p)$: 3;
- Latent dimension (learned representation): 8;
- LSTM units: 128;
- Time steps: 30;
- Batch size: 32.

For certainty note that we used mean squared error (MSE) as loss function, and Adam optimizer with clip set to 0.5 in order to avoid possible gradient explosion.

A models' summary that was used for both IAM-OnDB and UJI Pen datasets is at the fig. 4.1.

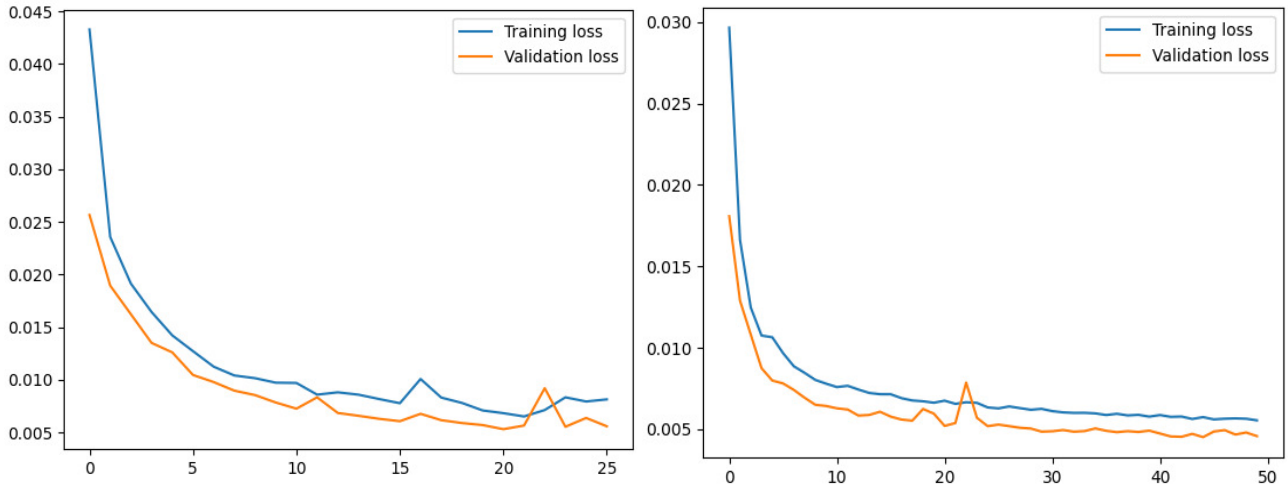
```
Encoder:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 30, 3)]           0
_________________________________________________________________
lstm (LSTM)                  (None, 128)               67584
_________________________________________________________________
dense (Dense)                (None, 8)                 1032
=================================================================
Total params: 68,616
Trainable params: 68,616
Non-trainable params: 0
_________________________________________________________________

Decoder:
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 8)]               0
_________________________________________________________________
repeat_vector (RepeatVector) (None, 30, 8)             0
_________________________________________________________________
lstm_1 (LSTM)                (None, 30, 128)           70144
_________________________________________________________________
lstm_2 (LSTM)                (None, 30, 3)             1584
=================================================================
Total params: 71,728
Trainable params: 71,728
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.1 RNN Autoencoder model summary.

Model training processes ended with losses equal to 0.0082 for UJI Pen Dataset and 0.0055for IAM_OnDB, and validation losses equal to 0.0056for UJI Pen Dataset and 0.0046 for IAM_OnDB; the total number of completed epochs are 26 for the model trained on UJI Pen Dataset and 50 for the model trained on IAM_OnDB. Both training processes stopped due to small change in validation loss values. The

figure4.2. belowdemonstrates a plot of training loss and validation loss over the number of epochs for both models.



a) b)

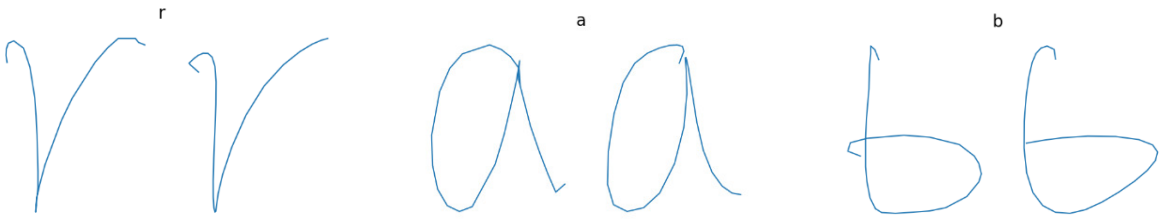
Figure 4.2 Loss curves (blue line corresponds to training loss, orange line corresponds to validation loss): a) for model trained on UJI Pen V2 Data Set; b) for model trained on IAM-OnDB

We checked the quality of the models (how well they learned the compressed representation of the input data) by reconstructing the compressed data.Examples of reconstructed allographs are at fig. 4.3 (with more examples in the appendix A).

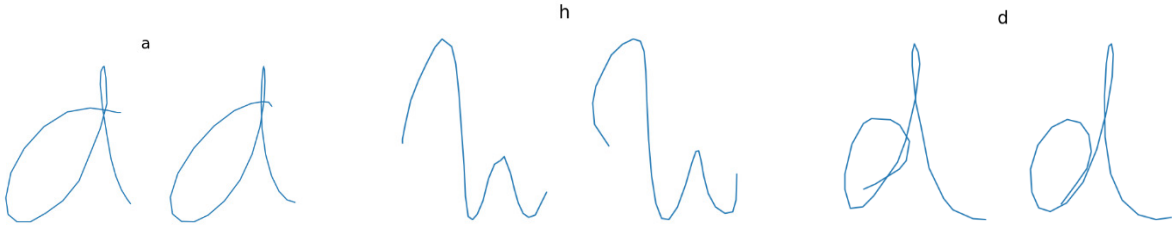
The pipeline of the whole process of feature extraction and clustering is as follows:

1. Build and train a recurrent neural network based autoencoder;
2. Perform feature extraction by using encoder for building an 8-dimensional learned representation for each sample x from the initial set X; save all the extracted features for further processing;
3. Perform clustering with K-means based on extracted features (with explicit number of clusters for initial clustering);

4. Perform DTW-based clusters merging in order to achieve human-congruous results.



a)



b)

Figure 4.3 Allograph reconstruction examples for model trained on: a) UJI Pen V2 Characters Data Set; b) IAM-OnDB.

After that, we can evaluate the clustering performance using metrics described in section 1.4 (Calinski-Harabazh index, Davies-Bouldin index, silhouette coefficient) and based on our perception since we consider so-called "human-likeness" as one of the indicators of satisfactory clustering results.

4.2. Running Experiments and Results Evaluation

While completing the final step of our research, which is clustering itself, we were faced with the task of choosing a lot of hyperparameters to achieve satisfactory results:

1. latent dimension;

2. number of initial clusters *N* for K-means algorithm;
3. DTW threshold.

Tan et al. [9]used 7-dimensional latent vectors for further clustering for character prototypes based automatic writer identification. They demonstrated that 7-dimensional latent space is enough to represent efficiently most characters in western handwriting. Based on their research, we decided to map the input data on a 8-dimensional latent space considering it sufficient for our task.

They also used a similar approach of 2-step clustering, choosing 10 as a number of clusters for initial clustering. We adopted their idea, considering 10 clusters to be an adequate number for separating character styles, since there are not so many critically different spellings of the same character in the Western alphabet.

However, choosing an optimal DTW threshold value is still an open question. If we select a value that is too small, all clusters will be eventually merged into single cluster, if we select it too large, no pair of clusters will be merged. We set this value equal to 3.0, because during the analysis, the result was found to be the most optimal solution. However, we assume that DTW threshold value must be established for each character separately, since all the characters differ in their nature and may have different number of existing allographs.

All our results and metrics values are shown below in the tables 4.1 and 4.2. Also in appendix A there is more examples of reconstruction of encoded alographs using an autoencoder. Once again, we want to point out that human-congruous is very difficult to measure from a metric point of view. How to construct a metric of "beauty" or "human-likeness" is still an open question. Most scientists, when it comes to such tasks, use the user-studies and their results as a metric for the evaluation of the overall quality [10]. We will still try to analyze the characteristics below.

Character	UJI Pen V2 Characters Data Set					
	Initial Clustering			+DTW Merging		
	CHI	DBI	SIL	CHI	DBI	SIL
«a»	43.92	1.09	0.23	45.31	1.19	0.21
«b»	77.33	1.03	0.27	3.95	6.14	0.35
«c»	33.8	1.14	0.21	2.85	4.07	-0.16
«d»	52.68	0.99	0.31	2.43	4.07	-0.26
«e»	25.95	1.26	0.19	4.35	4.62	-0.19
«f»	69.92	0.83	0.31	10.07	1.87	0.14
«g»	30.07	1.14	0.2	7.41	4.59	0.06
«h»	98.95	1.09	0.22	17.52	2.7	0.14
«i»	42.66	1.09	0.24	6.57	2.98	0.2
«j»	52.39	0.89	0.28	4.86	2.85	0.19
«k»	52.82	0.97	0.32	12.58	2.05	0.07
«l»	47.56	1.1	0.23	5.05	3.07	0.28
«m»	52.72	1.05	0.25	65.29	1.04	0.26
«n»	49.54	1.07	0.23	67.4	1.35	0.18
«o»	35.08	1.14	0.24	2.2	4.83	0.23
«p»	80.95	0.85	0.32	6.58	4.84	0.17
«q»	32.66	1.31	0.23	4.08	3.53	0.18
«r»	43.4	1.19	0.23	62.83	1.25	0.2
«s»	63.95	1.07	0.28	8.29	4.37	0.24
«t»	37.19	1.22	0.21	3.44	4.17	0.19
«u»	43.34	1.11	0.26	69.1	1.2	0.24
«v»	28.67	1.12	0.21	14.17	2.6	0.04
«w»	42.78	1.19	0.23	14.35	2.14	0.0
«x»	67.31	0.94	0.36	24.13	1.26	0.23
«y»	58.99	0.96	0.32	13.72	1.77	0.11
«z»	29.53	1.09	0.21	4.66	3.66	-0.14

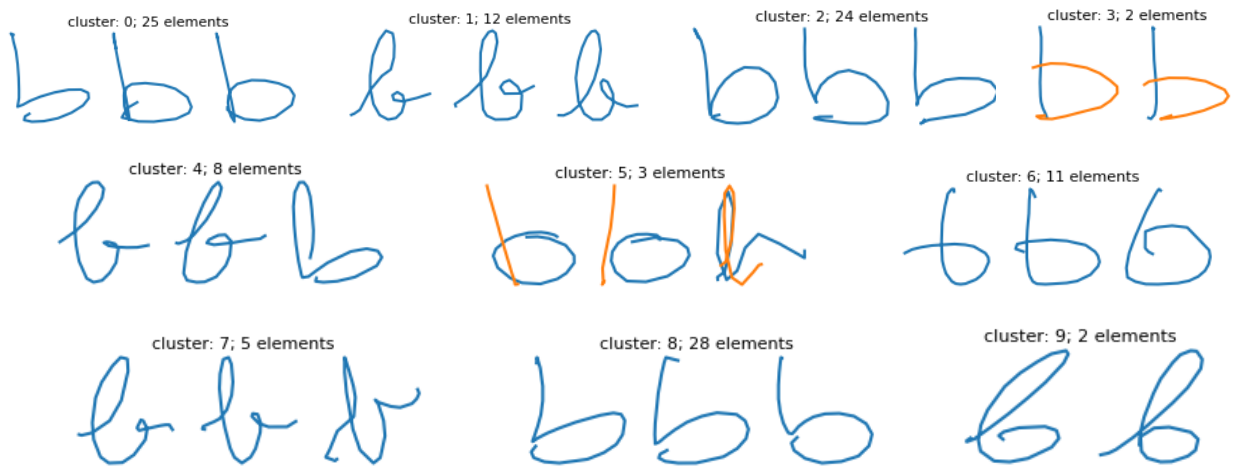
Table 4.1 Clustering performance evaluation for UJI Pen Data Set ([a-z]).

Character	IAM-OnDB					
	Initial Clustering			+DTW Merging		
	CHI	DBI	SIL	CHI	DBI	SIL
«a»	200.88	1.46	0.19	221.19	1.41	0.2
«b»	265.12	1.31	0.24	99.33	2.58	0.07
«c»	213.98	1.58	0.18	89.82	2.95	0.03
«d»	241.65	1.4	0.2	72.74	3.74	0.11
«e»	173.81	1.49	0.16	111.01	2.69	0.03
«f»	302.25	1.22	0.32	118.52	1.79	0.17
«g»	309.77	1.42	0.19	37.54	3.87	0.25
«h»	216.24	1.48	0.24	181.34	1.69	0.15
«i»	324.55	1.33	0.28	74.83	2.48	0.0
«j»	38.6	0.71	0.39	45.39	0.67	0.4
«k»	134.75	1.32	0.21	29.39	9.52	0.11
«l»	233.52	1.44	0.24	32.93	9.07	0.24
«m»	189.62	1.5	0.18	52.8	7.07	0.09
«n»	215.99	1.51	0.19	274.36	1.57	0.19
«o»	224.01	1.51	0.2	50.74	3.21	0.1
«p»	260.29	1.28	0.25	101.21	2.73	0.08
«q»	25.76	0.83	0.24	28.91	0.84	0.22
«r»	186.2	1.45	0.2	205.93	1.54	0.2
«s»	234.78	1.58	0.18	264.91	1.62	0.17
«t»	219.77	1.35	0.39	64.36	2.88	0.16
«u»	197.25	1.68	0.18	213.44	1.77	0.19
«v»	197.78	1.51	0.2	229.66	1.66	0.2
«w»	199.95	1.59	0.17	212.21	1.59	0.17
«x»	34.97	1.26	0.33	39.3	1.27	0.33
«y»	201.6	1.35	0.23	63.73	4.9	0.1
«z»	21.05	0.95	0.28	21.73,	1.01	0.28

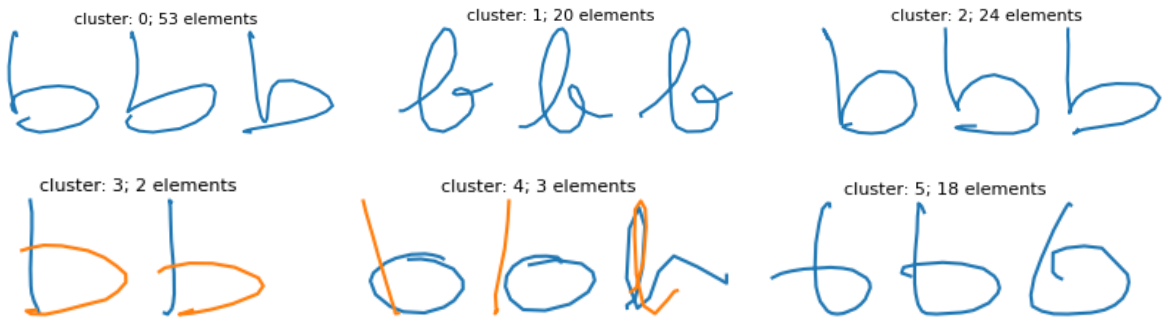
Table 4.2Clustering performance evaluation for IAM-OnDB ([a-z]).

As we can see, it is difficult to understand whether the quality of clustering has improved in general, since for the same symbol, one metric improves when assessing the quality of clustering with the use of DTW-based clusters merging, the other, on the contrary, worsens. We also propose to evaluate the results purely visually based on the analysis of the similarity of allographs from one cluster and their differences with others.

To do this, we hover the results below. Consider the results represented on fig.4.4. As we can see, the initial clustering results in similar-looking clusters. Cluster 1 and cluster 7 can be merged into one cluster, as well as cluster 0, cluster 2, and cluster 8. After applying a DTW-based merging technique, we get 6 clusters instead of 10, and, as we can see, the typical samples from each cluster differs significantly from the samples from the other clusters.

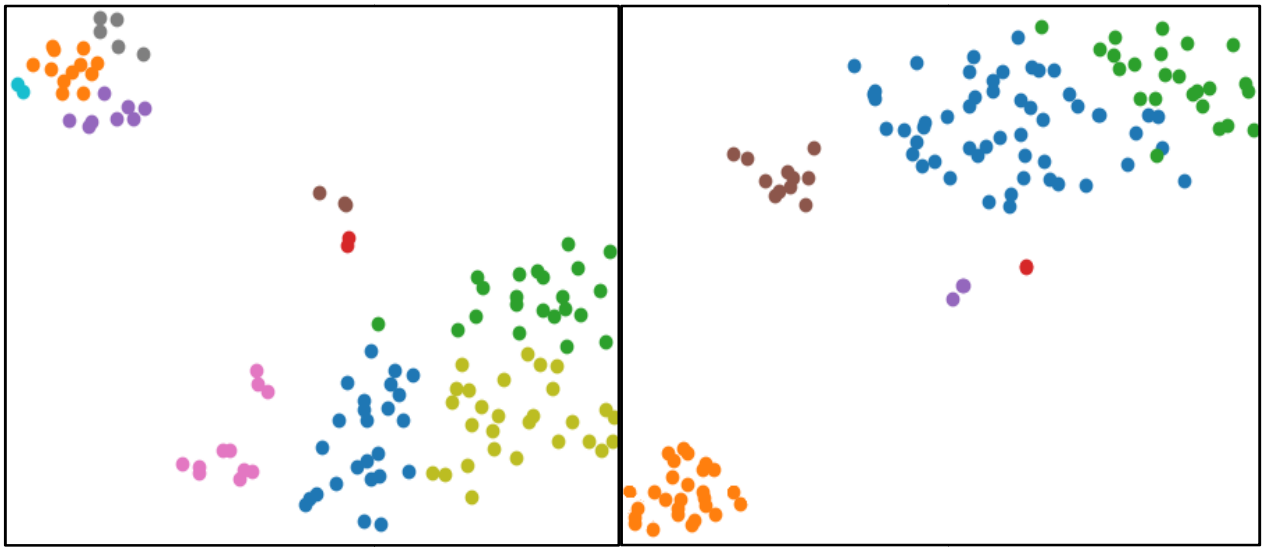


a)



b)

Figure 4.4 Character «b» clustering results (objects examples from all the clusters): a) before DTW-based merging approach; b) after DTW-based merging approach.



a) b)

Figure 4.5 Character “b” clustering results (t-SNE feature mapping): a) before DTW-based merging approach; b) after DTW-based merging approach.

Based only on visual perception (as at fig. 4.4) we would consider the obtained clustering results satisfactory. However, if we consider a t-SNE mapping as well (fig. 4.5) we will see that the cluster separateness was not changed significantly, but the cluster compactness even slightly worsened. This can be explained by the increasing of the cluster diameter size –a concept that is commonly used for cluster compactness calculation. Nevertheless, we were able to merge the nearest clusters into one, the instances within which are too similar to each other to belong to different clusters. This was achieved by introducing “human-congruousness” similarity which can be calculated using DTW.

Once again, it is difficult to find a compromise between pure mathematical methods of performance evaluation and human-line ones, since the last category was not still formalized strictly.

Nevertheless, based on the visual similarity of the instances inside cluster we got after DTW-based merging technique, we consider these results satisfactory.

More example of clustering results comparison of pure deep learning feature based approach and semi-neural combination of RNN AE and DTW algorithm for other characters can be found in appendix B.

It would also be considerable to use the DTW algorithm to assess the quality of the results, however this cannot be achieved for the same reasons as pure DTW-based clustering due to its high computational cost.

In the conclusions, we will also present possible ways to improve these results and also the scope of our future works.

CONCLUSIONS

In this study, we investigated and performed a detailed analysis of existing methods for allograph clustering. We overviewed handcrafted (based on handcrafted feature extraction with further shallow clustering) and automated (NN-based) approaches, and focused on RNN AE for building an efficient learning representation.We also investigated different approaches for data processing in order to transform data in a form that is informative for the model. Thus, we performed normalization, resampling to 30-point allograph representation [10-11], and time series derivative calculation [36] (with additional segmentation step for IAM-OnDB).

A simple RNN-basedautoencoder was built for learning efficient representations, and we compressed all the input data to the 8-dimensional latent space using trained encoder model. Note that we trained this model for both datasets our study is conducted on (IAM-OnDB and UJI Pen V2 Characters Data Set). We obtained two models with overall losses equal to 0.0082 for model trained on UJI Pen Dataset and to 0.0055for model trained on IAM_OnDB. Both models demonstrates satisfactory reconstruction ability.

We also proposed to use DTW-based clusters merging technique for achieving human-congruous results, inspired by research by Niels et al. for allograph-based writer identification. We improved proposed in his research approach by reducing the number of samples to be matched using DTW algorithm.

Therefore, we introduced a new method for handwritten allograph clustering, the main steps of which are as follows:

1. Perform feature extraction using encoder for building an 8-dimensional learned representation for each sample x from the initial set X;
2. Perform initial clustering with K-means based on extracted features, and divide the data into an excessive number of clusters (10);
3. Perform DTW-based clusters merging in order to achieve human-congruous results.

We also performed clustering performance evaluation using Calinski-Harabasz Index, Davies-Bouldin Index, and Silhouette coefficient. Based on these metrics, it is difficult to understand whether the quality of clustering has improved in general, because for the same symbol one metric improves when assessing the quality of clustering using merged DTW-based clusters, and the other, on the contrary, deteriorates. However, we introduced a "human-likeness" by applying DTW-based merging, which is impossible to measure. Based on the visual similarity of the instances inside cluster we got after DTW-based merging technique, we consider these results satisfactory.

Given the ambiguity of the samples and the initial conditions (lack of style labeling in the dataset, and the impossibility of dividing the samples into non-intersecting groups), as well as given the difficulties and problems in this subject area, the above results fully meet the requirements.

This system can be used to extract the characteristics of handwritten text, determine its style, which is important for designing systems for text recognition or localization, which are very important issues today. It can also be used in authentication or writer identification tasks when verifying the identity of the writer. In addition, this system can be used in such areas as criminology and graphology, where the definition of the features of the handwritten text is given much attention, as well as used to detect psychological disorders and diseases.

In the scope of future works, we propose to improve the following:

1. Investigate an optimal DTW threshold for clusters merging; we assume that DTW threshold value must be set for each character separately, since all the characters have different nature;
2. Measure the reconstruction quality; investigate approaches to do so (for instance, based on DTW cost);
3. Complicate the model architecture (for instance, by introducing global attention mechanism), experiment with its configuration.

REFERENCES

1. Korovai K., Marchenko O. Handwriting Styles Clustering: Feature Selection and Feature Space Analysis based on Online Input //2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP). – IEEE, 2020. – C. 195-200.
2. Vuurpijl L., Schomaker L. Finding structure in diversity: A hierarchical clustering method for the categorization of allographs in handwriting //Proceedings of the Fourth International Conference on Document Analysis and Recognition. – IEEE, 1997. – T. 1. – C. 387-393.
3. Vuori V. Clustering writing styles with a self-organizing map //Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition. – IEEE, 2002. – C. 345-350.
4. Sarkar P., Nagy G. Style consistent classification of isogenous patterns //IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2005. – T. 27. – №. 1. – C. 88-98.
5. Vuori V. et al. Adaptive methods for on-line recognition of isolated handwritten characters. – Helsinki University of Technology, 2002.
6. Zhelezniakov D., Zaytsev V., Radyvonenko O. Acceleration of online recognition of 2d sequences using deep bidirectional lstm and dynamic programming //International Work-Conference on Artificial Neural Networks. – Springer, Cham, 2019. – C. 438-449.
7. Bensefia A., Paquet T., Heutte L. Handwriting analysis for writer verification //Ninth International Workshop on Frontiers in Handwriting Recognition. – IEEE, 2004. – C. 196-201.
8. Chan S. K., Tay Y. H., Viard-Gaudin C. Online text independent writer identification using character prototypes distribution //2007 6th International Conference on Information, Communications & Signal Processing. – IEEE, 2007. – C. 1-5.

9. Tan G. X., Viard-Gaudin C., Kot A. C. Automatic writer identification framework for online handwritten documents using character prototypes //pattern recognition. – 2009. – Т. 42. – №. 12. – С. 3313-3323.
10. Niels R., Vuurpijl L., Schomaker L. Automatic allograph matching in forensic writer identification //International Journal of Pattern Recognition and Artificial Intelligence. – 2007. – Т. 21. – №. 01. – С. 61-81.
11. Schomaker L. Using stroke-or character-based self-organizing maps in the recognition of on-line, connected cursive script //Pattern recognition. – 1993. – Т. 26. – №. 3. – С. 443-450.
12. Graves A. Generating sequences with recurrent neural networks //arXiv preprint arXiv:1308.0850. – 2013.
13. Khomenko V. et al. Handwriting text/non-text classification on mobile device //The Fourth International Conference on Artificial Intelligence and Pattern Recognition (AIPR). – 2017. – С. 42-49.
14. Degtyarenko I. et al. Text/shape classifier for mobile applications with handwriting input //International Journal on Document Analysis and Recognition (IJDAR). – 2016. – Т. 19. – №. 4. – С. 369-379.
15. Hinton G. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups //IEEE Signal processing magazine. – 2012. – Т. 29. – №. 6. – С. 82-97.
16. Mikolov T. et al. Distributed representations of words and phrases and their compositionality //arXiv preprint arXiv:1310.4546. – 2013.
17. Plamondon R., Srihari S. N. Online and off-line handwriting recognition: a comprehensive survey //IEEE Transactions on pattern analysis and machine intelligence. – 2000. – Т. 22. – №. 1. – С. 63-84.
18. Guyon I. et al. Design of a neural network character recognizer for a touch terminal //Pattern recognition. – 1991. – Т. 24. – №. 2. – С. 105-119.
19. Akhtar, J. Perspective of pattern recognition in handwriting character recognition– PhD Thesis / J. Akhtar – New Orleans, 1995.

20. Walton J. Handwriting changes due to aging and Parkinson's syndrome //Forensic science international. – 1997. – Т. 88. – №. 3. – С. 197-214.
21. Marzinotto G. et al. Age-related evolution patterns in online handwriting //Computational and mathematical methods in medicine. – 2016. – Т. 2016.
22. Holekian F., Lavasani M. G., Madani Y. Handwriting analysis: the role of age and education //International Journal of Modern Management and Foresight. – 2014. – Т. 1. – №. 6. – С. 208-221.
23. Al Maadeed S., Hassaine A. Automatic prediction of age, gender, and nationality in offline handwriting //EURASIP Journal on Image and Video Processing. – 2014. – Т. 2014. – №. 1. – С. 1-10.
24. Marzinotto G. et al. Age and gender characterization through a two layer clustering of online handwriting //International Conference on Advanced Concepts for Intelligent Vision Systems. – Springer, Cham, 2015. – С. 428-439.
25. Caliński T., Harabasz J. A dendrite method for cluster analysis //Communications in Statistics-theory and Methods. – 1974. – Т. 3. – №. 1. – С. 1-27.
26. Davies D. L., Bouldin D. W. A cluster separation measure //IEEE transactions on pattern analysis and machine intelligence. – 1979. – №. 2. – С. 224-227.
27. Rousseeuw P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis //Journal of computational and applied mathematics. – 1987. – Т. 20. – С. 53-65.
28. Isenkul M., Sakar B., Kursun O. Improved spiral test using digitized graphics tablet for monitoring Parkinson's disease //Proc. of the Int'l Conf. on e-Health and Telemedicine. – 2014. – С. 171-5.
29. Liwicki, M. IAM-OnDB - An on-line English sentence database acquired from handwritten text on a whiteboard / M. Liwicki, H. Bunke. // ICDAR. – 2005. – №2. – С. 956 – 961.
30. Liwicki, M. HMM-Based On-Line Recognition of Handwritten Whiteboard Notes / M. Liwicki, H. Bunke. // IWFHR. – 2006. – №10.

31. The tagged LOB corpus user's manual [Electronic Resource] / S. Johansson, R. Atwell, R. Garside, G. Leech // Norwegian Computing Centre for the Humanities. – 1986. – Resource access mode: http://clu.uni.no/icame/manuals/LOBMAN
32. Llorens D. et al. The UJIpenchars Database: a Pen-Based Database of Isolated Handwritten Characters //LREC. – 2008.
33. MyScript: The Power of Handwriting [Electronic resource]// MyScript. – 10 October 2020. – Resource access mode: http://myscript.com/.
34. Carbune V. et al. Fast multi-language LSTM-based online handwriting recognition //International Journal on Document Analysis and Recognition (IJDAR). – 2020. – С. 1-14.
35. Ahrabian K., BabaAli B. Usage of autoencoders and Siamese networks for online handwritten signature verification //Neural Computing and Applications. – 2019. – Т. 31. – №. 12. – С. 9321-9334.
36. Górecki T., Łuczak M. Using derivatives in time series classification //Data Mining and Knowledge Discovery. – 2013. – Т. 26. – №. 2. – С. 310-331.
37. "Recurrent neural network." [Electronic resource]. /Wikipedia contributors. // Wikipedia, The Free EncyclopediaWikipedia .–16 Apr. 2021.
38. Hochreiter S., Schmidhuber J. Long short-term memory //Neural computation. – 1997. – Т. 9. – №. 8. – С. 1735-1780.
39. Understanding LSTM networks [Electronic Resource] / Olah C. – 2015.– Resource access mode: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
40. Guyon I. et al. UNIPEN project of on-line data exchange and recognizer benchmarks //Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5). – IEEE, 1994. – Т. 2. – С. 29-33.
41. Niels R. Dynamic Time Warping: An intuitive way of handwriting recognition? – Master Thesis / R. Niels –Nijmegen, 2004.
42. Niels R. M. J., Vuurpijl L. G. Using Dynamic Time Warping for intuitive handwriting recognition. – 2005.

43. RBellman R., Kalaba R. On adaptive control processes //IRE Transactions on Automatic Control. – 1959. – Т. 4. – №. 2. – С. 1-9.
44. Muller M. Dynamic Time Warping in Information Retrieval for Music and Motion. – 2007.
45. Sakoe H., Chiba S. Dynamic programming algorithm optimization for spoken word recognition //IEEE transactions on acoustics, speech, and signal processing. – 1978. – Т. 26. – №. 1. – С. 43-49.
46. Myers C., Rabiner L., Rosenberg A. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition //IEEE Transactions on Acoustics, Speech, and Signal Processing. – 1980. – Т. 28. – №. 6. – С. 623-635.
47. Senin P. Dynamic time warping algorithm review //Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA. – 2008. – Т. 855. – №. 1-23. – С. 40.

APPENDIX A.

Allograph Reconstruction

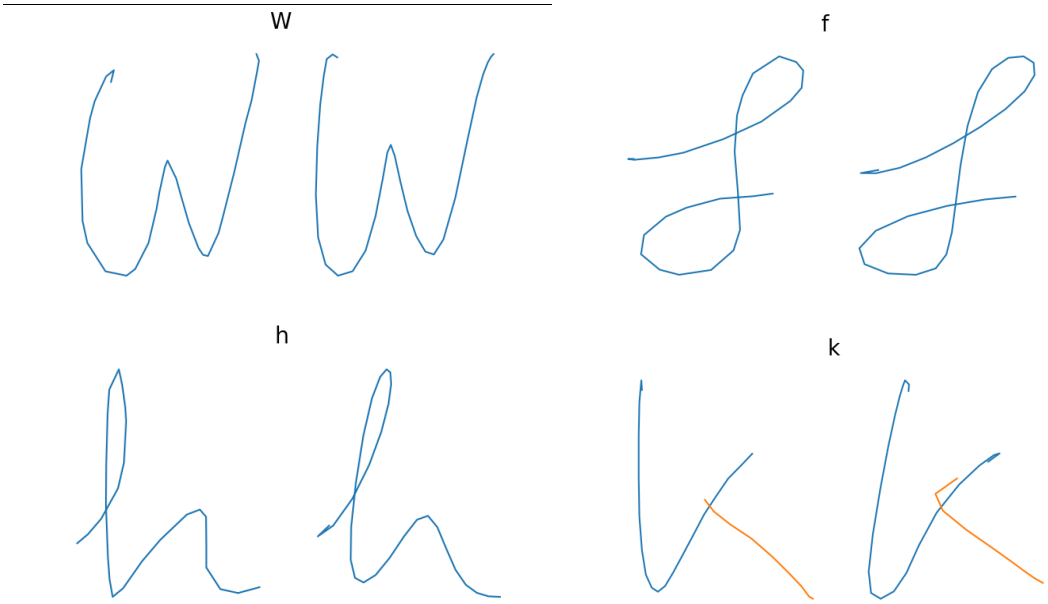


Figure A.1 Allograph reconstruction examples for model trained on UJI Pen V2 Characters Data Set.

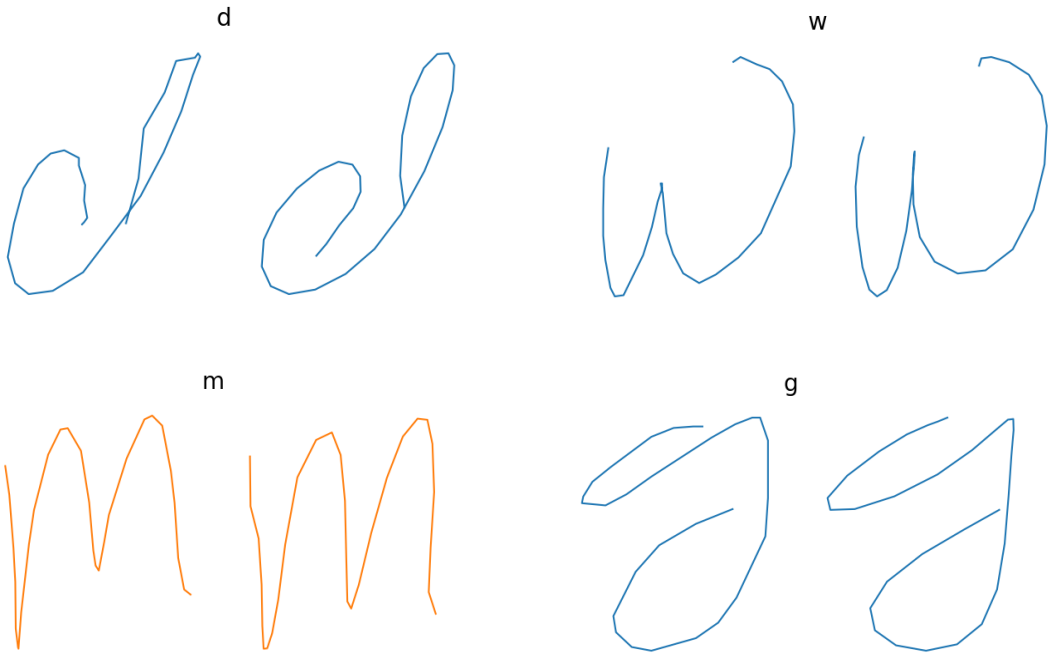
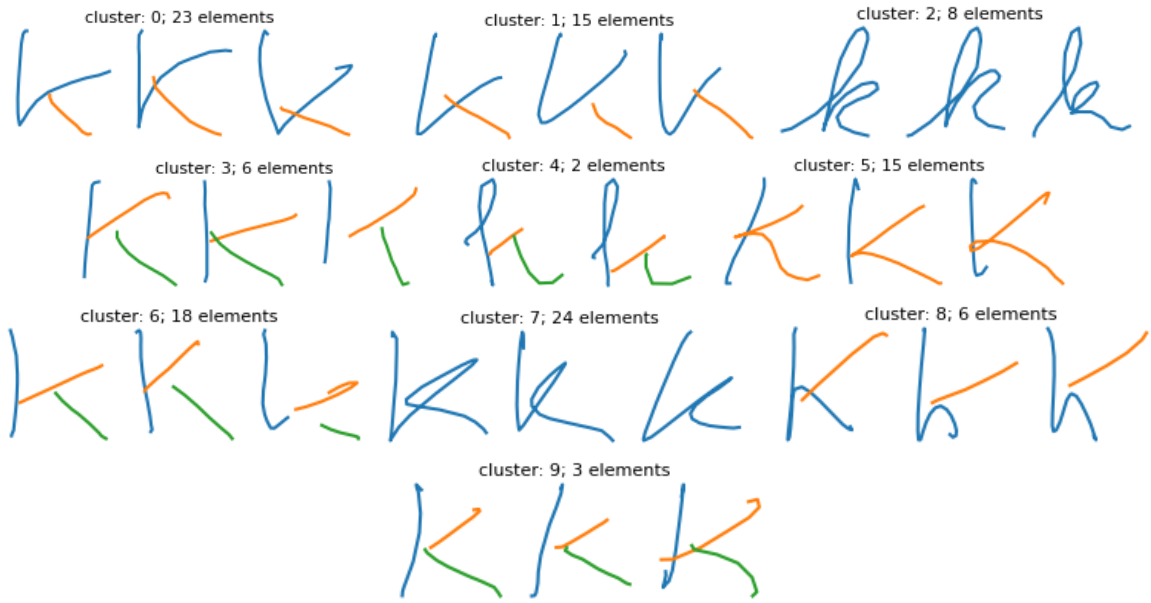


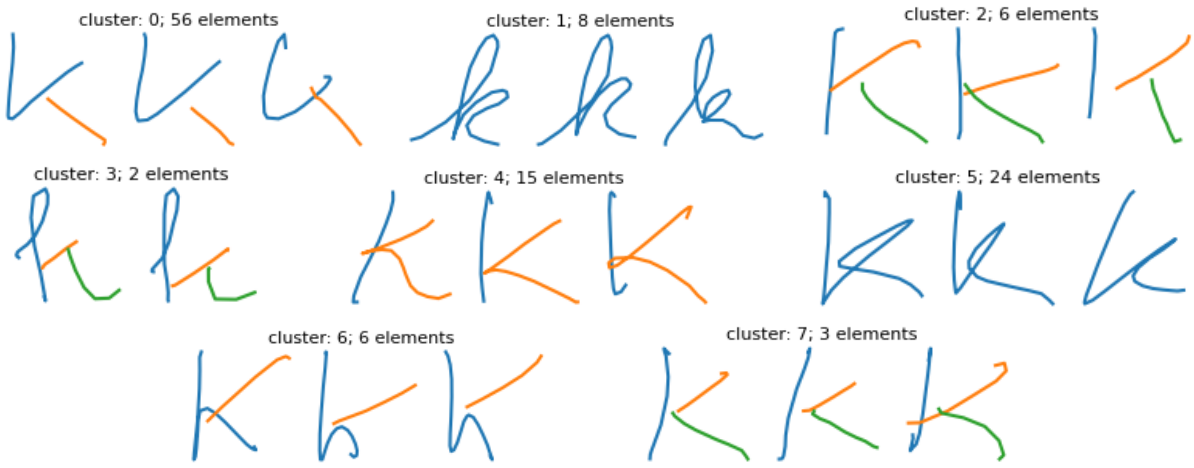
Figure A.2 Allograph reconstruction examples for model trained on IAM-OnDB.

APPENDIX B.

Clustering Results: Examples of Allograph Clusters



a)



b)

Figure B.1 Character «k» clustering results (objects examples from all the clusters): a) before DTW-based merging approach; b) after DTW-based merging approach.