IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 September 2024, accepted 19 September 2024, date of publication 15 October 2024,
date of current version 26 November 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3481161*

NEGATIVE RESULT

# Forging the Forger: An Attempt to Improve Authorship Verification via Data Augmentation

**SILVIA CORBARA**[1,2] **AND ALEJANDRO MOREO**[2]
[1]Scuola Normale Superiore, 56126 Pisa, Italy
[2]Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, 56124 Pisa, Italy

Corresponding author: Silvia Corbara (silvia.corbara@isti.cnr.it)

The work of Alejandro Moreo was supported in part by the SoBigData++ Project funded by European Commission under the H2020 Programme under Grant 871042 and Grant INFRAIA-2019-1; and in part by the SoBigData.it Project funded by Italian Ministry of University and Research under the NextGenerationEU Program.

**ABSTRACT** Authorship Verification (AV) is a text classification task concerned with inferring whether a candidate text has been written by one specific author ($A$) or by someone else ($\overline{A}$). It has been shown that many AV systems are vulnerable to adversarial attacks, where a malicious author actively tries to fool the classifier by either concealing their writing style, or by imitating the style of another author. In this paper, we investigate the potential benefits of augmenting the classifier training set with (negative) synthetic examples. These synthetic examples are generated to imitate the style of $A$. We analyze the improvements in the classifier predictions that this augmentation brings to bear in the task of AV in an adversarial setting. In particular, we experiment with three different generator architectures (one based on Recurrent Neural Networks, another based on small-scale transformers, and another based on the popular GPT model) and with two training strategies (one inspired by standard Language Models, and another inspired by Wasserstein Generative Adversarial Networks). We evaluate our hypothesis on five datasets (three of which have been specifically collected to represent an adversarial setting) and using two learning algorithms for the AV classifier (Support Vector Machines and Convolutional Neural Networks). This experimentation yields negative results, revealing that, although our methodology proves effective in many adversarial settings, its benefits are too sporadic for a pragmatical application.

**INDEX TERMS** Authorship identification, authorship verification, data augmentation, text classification.

## I. INTRODUCTION

The field of *Authorship Identification* (AId) is the branch of *Authorship Analysis* concerned with the study of the true identity of the author of a written document of unknown or debated paternity. *Authorship Verification* (AV) is one of the main tasks of AId: given a single candidate author $A$ and a document $d$, the goal is to infer whether $A$ is the real author of $d$ or not [1]. The author $A$ is conventionally represented by a set of documents that we unequivocally know have been written by $A$. AV is sometimes cast as a one-class classification problem [2], [3], with $A$ as the only class. Nevertheless, it is more commonly addressed as a binary classification problem, with $A$ and $\overline{A}$ as the possible classes, where $\overline{A}$ is characterized by a collection of documents from authors other than $A$, but somehow related to $A$ (sharing, e.g., the same language, period, literary style, genre).

The goal of AId in general, and of AV in particular, is to find a proper way to profile the "hand" of a given writer, in order to clearly distinguish their written production from that of other authors. This characterization is often tackled through the use of "stylometry", a methodology that disregards the artistic value and meaning of a written work, in favour of conducting a frequentist analysis of linguistic events. These events, also known as "style markers", typically escape the conscious control of the writer and are assumed to remain approximately invariant throughout the literary production of a given author, while conversely

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa M. Fouda.

varying substantially across different authors [4, p. 241]. Current approaches to AV typically rely on *automated text classification*, wherein a supervised machine learning algorithm is trained to generate a classifier that distinguishes the works of $A$ from those of other authors by analyzing the vectorial representations of the documents that incorporate different style-markers.

However, training and employing an effective classifier can be very challenging, or even impossible, if an ''adversary'' is at play, i.e., when a human or an automatic process actively tries to mislead the classification. We can distinguish between two main variants of this adversarial setting [5]. In the first case, called ''obfuscation'', the authors themselves try to conceal their writing style in order to not be recognized. This can be done manually, but nowadays specific automatic tools are available for this purpose; for example, by applying sequential steps of machine translation [6]. In the second case, called ''imitation'', a forger (the ''imitator'') tries to replicate the writing style of another specific author. While the former case can be considered as possibly less harmful (a writer might simply be interested in preserving their privacy, without necessarily harbouring any malevolent intent), the latter case is inherently illicit (an exception to this may be found in artistic tributes — provided they are accompanied by a statement openly acknowledging the intent). Our cultural heritage is indeed filled with countless examples of historical documents of questioned authorship, often caused by supposed forgeries or false appropriations [7], [8], [9], [10], [11], [12]. Moreover, due to the recent significant advances in language modelling, powerful Neural Networks (NNs) are now able to autonomously generate ''coherent, non-trivial and human-like'' text samples [13, p. 1], that can be exploited as fake news or propaganda [13], [14]. Indeed, it has been shown that many AId systems can be easily deceived in adversarial contexts [5], [15]. This has given rise to the discipline known as ''adversarial authorship'' (or ''adversarial stylometry'', or ''authorship obfuscation'') devoted to study specific techniques able to fool AId systems [6], [16], [17], [18], [19], [20].

A seemingly straightforward solution to this problem would be to add a set of representative texts from the forger to the training set used to build the classifier, thus allowing the training algorithm to find descriptive patterns that effectively identify adversarial examples. Unfortunately, this strategy is generally unfeasible, as in most cases we might not expect to have any such examples.

In this work, we investigate ways to improve the performance of an AV classifier by augmenting its training set with *synthetically* generated examples that mimic the style of the author that the classifier tries to identify. In particular, we explore various generator architectures, including a GRU model [21], a standard transformer [22], and the popular GPT system (in its shallower variant, `DistilGPT2` [23]). Two distinct training strategies for the generators are employed: one draws inspiration from standard Language Models (LMs) where the generator learns to emulate the writing style of a specific author, while the other takes cues from Wasserstein Generative Adversarial Networks (WGANs) by training a generator to exploit the weaknesses of the classifier. We run experiments on five datasets (three of them collected specially to simulate an adversarial setting), and using two learning algorithms for the AV classifier: Support Vector Machines (SVMs) and Convolutional Neural Networks (CNNs).

The remaining of this paper is organized as follows. In Section II we survey the related literature. In Section III, we present our method, describing the generator architectures and their training strategies, and the learning algorithms that we employ for the AV classifier. In Section IV, we first detail the experimental setting, including the datasets and the experimental protocol we use, and then we present and comment on the results of our experiments. Despite our efforts, the results we obtain seem to indicate that data augmentation is not always beneficial for the task of AV (at least with the techniques we study here); in Section V we analyse the possible causes of the negative results. Finally, Section VI wraps up, offering some final remarks and pointing to some possible avenues for future research.

## II. RELATED WORK

AId is usually tackled by means of machine-learned classifiers or distance-based methods; the annual PAN shared tasks [24], [25], [26], [27], [28] offer a very good overview of the most recent trends in the field.

In particular, the baselines presented in the 2019 edition [24] are representative of the systems that have long been considered standard, i.e., traditional learning algorithms such as SVM or logistic regression, compression-based algorithms, and variations on the well-known Impostors Method [29]. In particular, SVM has become a standard learning algorithm for many text classification tasks, due to its robustness to high dimensionality and to its wide scope of applicability. In various settings, SVMs have been found to outperform other learning algorithms such as decision trees and even NNs [30], especially in regimes of data scarcity [31]. On the other hand, despite the many successes achieved in other natural language processing tasks [32], deep NNs have rarely been employed in tasks of AId, arguably due to the huge quantity of training data they usually require. Even though one of the first appearances of NNs at PAN dates back to 2015 [33], and even though this approach won the competition [34], only recently the generalized belief that ''simple approaches based on character/word $n$-grams and well-known classification algorithms are much more effective in this task than more sophisticated methods based on deep learning'' [35, p.9] was called into question. As a result, NNs methods are nowadays becoming more and more common at PAN [25], [26], [27].

Needless to say, the use of data augmentation for improving classification performance is not a new idea, neither in the text classification field nor in the AId field.

Researchers have explored various techniques for generating synthetic samples, such as the random combination of real texts [36], [37], or the random substitutions of words with synonyms [38], or by interrogation of LMs [39]. Similarly, adversarial examples are generated by a process that actively tries to fool the classification [40], and have been extensively used to improve the training of a classifier for many text classification tasks in general, and for counteracting adversarial attacks in particular. For example, Zhai et al. [18] feed the learning algorithm with (real) texts that have been purposefully obfuscated, in order to make the classifier robust to obfuscation.

Unlike these works, we do not automatically modify pre-existing texts; instead, we employ various generation algorithms to create new samples, simulating the deceitful actions of an adversary. In particular, one generative technique we explore is the so-called Generative Adversarial Network (GAN), which was first introduced by Goodfellow et al. [41] back in 2014. The GAN architecture is made of two components: a Generator ($G$) that produces synthetic (hence fake) examples, and a Discriminator ($D$) that classifies examples as "real" (i.e., coming from a real-world distribution) or "fake" (i.e., generated by $G$). Both components play a min-max game, where $D$ tries to correctly spot the examples created by $G$, while $G$ tries to produce more plausible examples in order to fool $D$.

Although the GAN strategy has excelled in the generation of images [42], its application to text generation has proven rather cumbersome. One of the main reasons behind this is the discrete nature of language: choosing the next word in a sequence implies picking one symbol from a vocabulary, an operation that is typically carried out through an argmax operation, which blocks the gradient flow during backpropagation. Some strategies have been proposed to overcome this limitation. One idea is to employ a reinforcement learning approach, where the feedback from $D$ acts as reward [43], although this strategy has been found to be inefficient to train [44]. An alternative idea is to use the Gumbel-Softmax operation to obtain a differentiable approximation of one-hot vectors [45], or to directly process the continuous output of the generator (i.e., refraining from choosing specific tokens); some examples include the encoding of real sentences via an autoencoder [44], or the creation of a word-embedding matrix for real sentences [46].

The work by Hatua et al. [47] bears some similarities with our methodology: they employ a GAN-trained generator to produce new examples, which are labelled as negatives and then added to the classifier training dataset. Their experiments indeed demonstrate the benefits of data augmentation for the fact-checking task.

Within the AId field, the work by Manjavacas et al. [48] explores the idea to use data augmentation to boost the performance of an authorship classifier, employing a RNN as language model. They indeed report a slight increase in the attribution performance of the system; however, they perform a very narrow experimentation, without significance testing, and do not tackle the problem of forgery, limiting their investigation to a single non-adversarial setting.

Finally, this paper is a thorough extension of the preliminary experiments presented in the short paper by Corbara and Moreo [49]. In the present work, we carry out a more robust experimentation, in which we test an improved GAN strategy (based on the Wasserstein GAN), and in which we consider different vector representations for the input of our generator models. We also expand the number of datasets from three to five.

## III. METHODOLOGY

In this paper, we approach the AV task as a binary text classification problem where the objective is to determine whether a given document $d$ was written by a specific candidate author (representing the positive class $A$, which includes representative texts from that author) or by any other author (collectively represented as the negative class $\overline{A}$, encompassing examples from other authors). The union of all the documents with the corresponding labels define the training set $L$ that we use to generate a classifier $h$ via an inductive algorithm.

We propose to enhance the classifier performance with the addition of adversarial training examples, i.e., textual examples specifically generated to imitate the author $A$. The generated examples are labelled as negative examples ($\overline{A}$) and added to $L$ with the aim of generating an improved classifier $h^*$. Note that, unlike works such as the one by Jones et al. [50], we do not employ the newly generated examples as synthetic positive instances (in our case: for the class $A$); otherwise, the classifier would learn to label fraudulent instances as $A$, which is not our goal.

This process is sketched in Figure 1.[1] As shown in the diagram, we explore three different generator architectures (GRU, TRA, and GPT) that we discuss in Section III-A1, and that we train using two different learning algorithms (`LMtr` and `GANtr`) that we describe in Section III-A2. Concerning the underlying classifier of our AV system, we experiment with two learning algorithms (SVM and CNN) that we discuss in Section III-B.

### A. SYNTHESIZING FORGERY DOCUMENTS

In this section, we describe the generation process by which we obtain new synthetic documents meant to imitate the production of $A$. In Section III-A1 we describe the generator architectures that we explore in our experiments, while in Section III-A2 we describe the training strategies we employ.

#### 1) GENERATOR ARCHITECTURES

In order to generate the adversarial examples, we experiment with three alternative generator architectures of increasing level of complexity:

- GRU: Ezen [51] shows that simpler recurrent models (specifically: LSTM) tend to outperform more

[1] Icons made by Vitaly Gorbachev on Flaticon: https://www.flaticon.com/

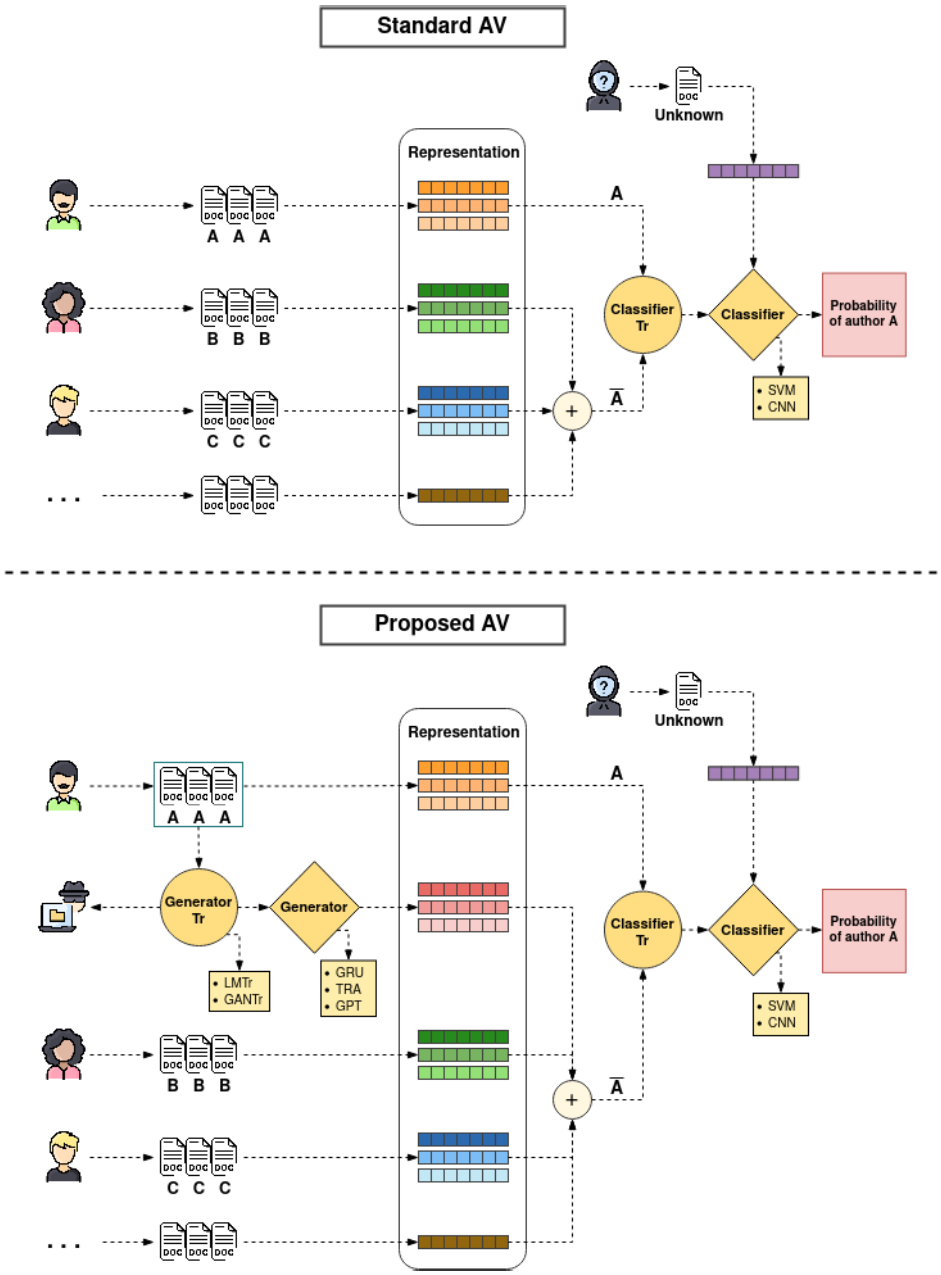



**FIGURE 1.** Upper: Flowchart of a standard AV method. Bottom: Flowchart of our proposed AV method, where representative examples of forgery are added to $\overline{A}$.

sophisticated models (specifically: BERT) when the training data is small, since simpler models have fewer parameters and are thus less prone to overfitting. Given that data scarcity is characteristic of many AV settings, we consider the Gated Recurrent Unit (GRU) [21] model, a simplified variant of LSTM. We set our model with 2 unidirectional GRU-layers of 512 hidden units each, followed by a linear layer with a ReLU activation function.[2]

- TRA: We consider the original transformer introduced by Vaswani et al. [22], that replaced a recurrently-handled memory in favor of fully-attention layers. We set our model with 2 encoder layers of 512 hidden dimensions each and 4 attention heads, followed by a linear layer with a ReLU activation.[3]
- GPT: The forefront in current generation models is held by very large LMs, with Chat-GPT 4 occupying the top position. Unfortunately, these models are huge in

[2]We use the PyTorch implementation of GRU: https://pytorch.org/docs/stable/generated/torch.nn.GRU.html

[3]Our implementation relies on Pytorch's `TransformerEncoder`: https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoder.html

terms of the number of parameters, and are typically not released to the scientific community, if not behind a pay-wall and an API. As a representative (though much smaller) model, we focus on GPT-2, a pre-trained unidirectional transformer [23] released by OpenAI that became increasingly popular for its outstanding generation performance and general-purpose nature. Uchendu et al. [52] and Fagni et al. [13] assessed the quality of its generated texts, concluding that these are often more human-like than those from other text generators. To limit the number of parameters and speed up the computation, while retaining the good quality of the original model, we employ DistilGPT2 [53], a smaller model (82M parameters versus the 124 M parameters of the standard version of GPT-2) that is trained via knowledge distillation on GPT-2. Note that this model outputs a multinomial distribution over the entire vocabulary at each generation step, and then selects the next token by using a sampling technique; we use the top-$k$ sampling, where the $k$ most-likely next words are filtered, and the probability mass is redistributed among those $k$ words (we set $k = 50$). We enforce no repetitions within 5-grams.[4]

We explore different vector representations for the input and output of the generators:

- One-hot encoding (`1h`): a typical representation in which a vector of length $|V|$, where $V$ is the vocabulary, contains exactly one "1" whose index identifies one specific word in $V$ (all other values are set to "0"). We use the DistilGPT2 word tokenizer in all our experiments, which results in a vocabulary size of $|V| = 50{,}257$ tokens. By $\text{GRU}_{\texttt{1h}}$ and $\text{TRA}_{\texttt{1h}}$ we denote the variants equipped with an input linear (without bias) layer that projects the one-hot representation onto 128 dimensions, hence densifying the representation (GPT is a more complex model and we do not explore the `1h` variant). The output of these models is generated by a last linear layer of $|V|$ dimensions. During the `GANtr`, we compute the Gumbel-Softmax distribution over the output in order to discretize the output for the next word in the sequence without interrupting the gradient.[5]
- Dense encoding (`emb`): we also experiment with a variant in which the generator produces embeddings of the same dimension of the ones used by the CNN method of Section III-B; we call the three models $\text{GRU}_{\texttt{emb}}$, $\text{TRA}_{\texttt{emb}}$, and $\text{GPT}_{\texttt{emb}}$, respectively.

### 2) GENERATOR TRAINING

We explore two different strategies for training the generator architectures described above: one based on standard Language-Model training (hereafter: `LMtr`), in which the generator is trained to replicate the style of the author of interest, and another based on GAN training (hereafter: `GANtr`), where the generator plays the role of a forger that tries to fool the discriminator.

#### *a: LANGUAGE MODEL TRAINING (`LMTr`)*

A standard way in which LMs are trained comes down to optimizing the model to predict the next word in a sequence, for a typically large number of sequences. More formally, given a sentence $w_1, \ldots, w_t$ of $t$ tokens, the model is trained to maximize the conditional probability $\Pr(w_t|w_1, w_2 \ldots, w_{t-1}; \Theta)$, where $\Theta$ are the model parameters. Such sequences are drawn from real textual examples, and could either come from a generic domain (thus optimizing a LM for a language in general) or from a specific domain (thus optimizing the LM for a particular area of knowledge or task).

We are interested in the latter case. Specifically, we draw sequences from texts that we know have been written by $A$, thus attaining a LM that tries to imitate the author (i.e., that tries to choose the next word as $A$ would have chosen). This idea has been shown to retain the stylistic patterns typical of the target author to a certain extent [50]. Of course, the main limitation of this strategy is the limited amount of sequences we might expect to have access to, since these are bounded by the production of one single author; such sequences might be very few when compared with the typical amount of information used to train LMs.

#### *b: GENERATIVE ADVERSARIAL NETWORK TRAINING (`GANTr`)*

The Wasserstein GAN (WGAN) approach [54] is based on a GAN architecture (see Section II) that relies on the Earth-Mover (also called Wasserstein) distance as the loss function.[6] This loss is continuous everywhere and produces smoother gradients, something that has been shown to prevent the gradient-vanishing problem and the mode-collapse problem that typically affect the early stages of the GAN training, when the generator $G$ still performs poorly. Gulrajani et al. [55] later developed WGANGP, that improves the stability of the training by penalizing the gradient of the discriminator $D$ instead of clipping the weights (as proposed in the original formulation), which is the approach that we adopt in this paper.[7]

We leverage GAN training to generate samples that the classifiers find challenging to distinguish from the positive class (i.e., the author of interest). Incorporating these generated samples into the training set should enhance the

[4]We rely on the Huggingface's `transformers` implementation: https://huggingface.co/distilgpt2

[5]We rely on the PyTorch implementation: https://pytorch.org/docs/stable/generated/torch.nn.functional.gumbel_softmax.html

[6]In the related literature, the discriminator underlying a WGAN is sometimes called "the critic" instead of "the discriminator", since it outputs a confidence score in place of a posterior probability. This distinction is rather unimportant for the scope of our paper, so we keep the term "discriminator" for the sake of simplicity.

[7]We use the `PyTorch`-based implementation available at: https://github.com/eriklindernoren/PyTorch-GAN/tree/master/implementations/wgan_gp

classifier's robustness, improving its ability to differentiate the author's production from other instances (see Section II).

As the generator $G$, we explore the architectures (GRU, TRA, GPT) described in Section III-A1, while for the discriminator $D$ we employ a CNN-based classifier (the same classifier that we describe in detail in Section III-B3).

### B. AV CLASSIFIERS

We experiment with two different classifiers for our AV system: one based on SVM (Section III-B2) and another based on CNN (Section III-B3). Before describing the learning algorithms we employ, we present in Section III-B1 the set of features we extract that are commonly employed in the authorship analysis literature.

#### 1) BASE FEATURES

The following set of features have proven useful in the related literature and are now considered standard in many AId studies. We hereafter refer to this set as "Base Features" (BFs).

- Function words: the normalized relative frequency of each function word. For a discussion about this type of features, see e.g., the analysis conducted by Kestemont [56]. We use the list provided for the English stopwords by the NLTK library.[8]
- Word lengths: the relative frequency of words up to a certain length, where a word length is the number of characters that forms a word. We set the range of word lengths to $[1, n]$, where $n$ is the longest word appearing at least 5 times in the training set. These are standard features employed in statistical authorship analysis since Mendenhall's "characteristic curves of composition" [57].
- POS-tags: the normalized relative frequency of each Part-Of-Speech (POS) tag. POS tags are an example of syntactic features, and are often employed in AId studies, also thanks to their topic-agnostic nature. We extract the POS-tags using the Spacy English tagger module.[9]

#### 2) SVM CLASSIFIER

We consider a SVM-based classifier as our AV system, due to the good performance SVMs have demonstrated in text classification tasks in general over the years [58], and in authorship analysis-related tasks in particular [24].

For SVM, we employ the implementation from the `scikit-learn` package.[10] We optimize the hyper-parameters of the classifier via grid-search. In particular, we explore the parameter $C$ in the range $[0.001, 0.01, 0.1, 1, 10, 100, 1000]$, the parameter kernel in the range $\{linear, poly, rbf, sigmoid\}$, and we explore whether to rebalance the class weights or not. After model selection, the SVM is then re-trained on the union of training and validation sets with the optimized hyper-parameters, and then evaluated on the test set.

We use the BFs set as features.

#### 3) CNN CLASSIFIER

We also experiment with an AV classifier based on the following CNN architecture. The input layer of the CNN has variable dimensions depending on the vector modality: the dense representations (`emb`) depend on the generator architecture of choice (GPT produces 768-dimensional vectors, while we set the output dimensions of GRU and TRA to 128); while the one-hot representations (`1h`) consist of $|V| = 50{,}257$ dimensions that are projected into a 128-dimensional space by means of a linear layer (without bias). This is followed by two parallel convolutional blocks with kernel sizes of 3 and 5, respectively. Each convolutional block consists of two layers of 512 and 256 dimensions and ReLU activations. We then apply max-pooling to the resulting tensors and concatenate the outputs of both blocks. We also apply dropout with 0.3 probability, and the resulting tensor is then processed by a linear transformation of 64 dimensions with ReLU activation. For generator architectures for which we can derive "plain texts" (i.e., for all the `1h`-variants plus GPT), we add a parallel branch that receives their BFs as inputs: these features are processed by two linear layers of 128 and 64 dimensions and ReLU activations. The resulting tensor is then concatenated with the output of the other branch, and the result is passed through a final linear layer with ReLU activation which produces a single value representing the confidence score of the classifier. The complete CNN model is depicted in Figure 2.

We train the network using the AdamW optimizer [59] with a learning rate of 0.001, a batch size of 32, and binary cross entropy as the loss function. In order to counter the effect of class imbalance (there are many more negatives than positives), we set the class weights to the ratio between negative and positive examples.

We train the model for a minimum of 50 epochs and a maximum of 500 epochs, and apply early stopping when the performance on the validation set (as measured in terms of $F_1$) does not improve for 25 consecutive epochs. Finally, we re-train the model on the combination of training and validation sets for 5 epochs before the model evaluation.

## IV. EXPERIMENTAL SETTING

In this section, we present the experiments we have carried out. In Section IV-A we describe the datasets we employ, while in Section IV-B we describe the experimental protocol we follow. Finally, Section IV-C discusses the results we have obtained.

All the models and experiments are developed in Python, employing the `scikit-learn` library [60] and the `PyTorch` library [61]. The code to reproduce all our experiments is available on GitHub.[11]

[8] https://www.nltk.org/
[9] https://spacy.io/usage/linguistic-features#pos-tagging
[10] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[11] https://github.com/silvia-cor/Authorship_DataAugmentation

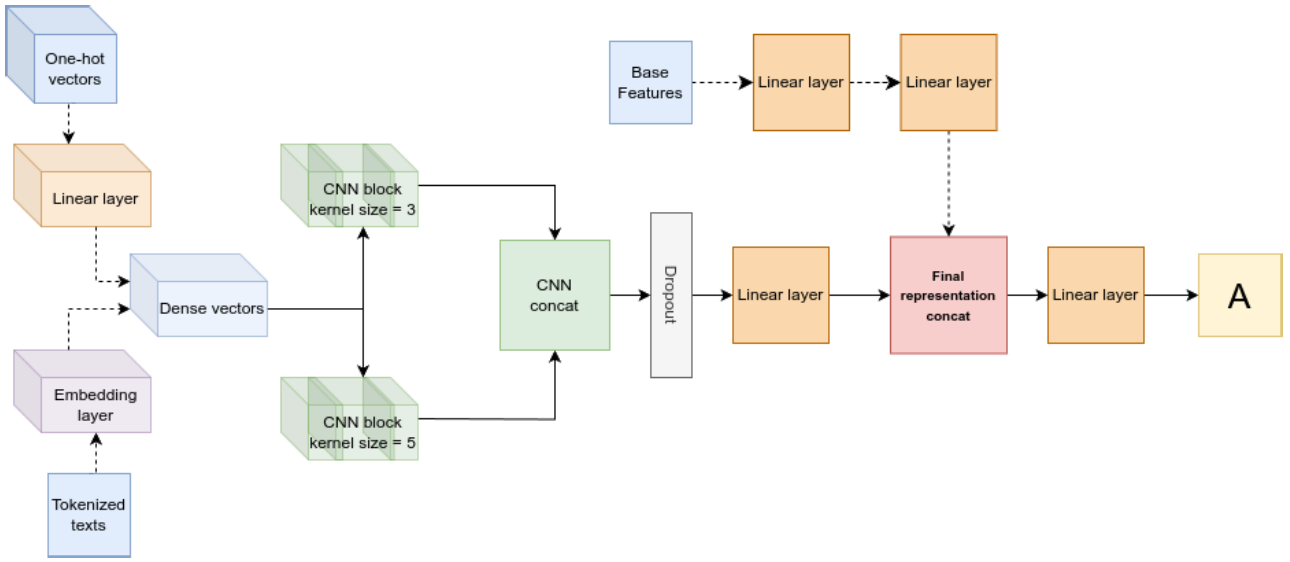



**FIGURE 2.** The CNN classifier architecture. The dotted lines represent alternative branches.

### A. DATASETS

We experiment with five publicly available datasets, some of which are examples of a close-set setting (where the authors comprising the test set are also present in the training set), while others are instead open-set (where the authors comprising the test set are not necessarily present in the training set).

- **`TweepFake`**. This dataset was created and made publicly available by Fagni et al. [13];[12] it contains tweets from 17 human accounts and 23 bots, each one imitating one of the human accounts. The dataset is balanced (the tweets are half human- and half bot- generated) and already partitioned into a training, a validation, and a test set. Since it would realistically be rather problematic to obtain a corpus containing human forgeries, we use this dataset as a reasonable proxy, where the forgery is made by a machine trying to emulate a human writer. In order to reproduce a more realistic setting, we only consider the documents produced by human users for our training and validation sets, but we keep all the documents in the test set in order to emulate an open-set problem.
- **`EBG`**. The Extended Brennan-Greenstadt Corpus was created by Brennan et al. [5]; we use the Obfuscation setting[13] consisting of writings from 45 individuals contacted through the Amazon Mechanical Turk platform. Participants were asked to: i) upload examples of their own writing (of "scholarly" nature), and ii) to write a short essay regarding the description of their neighborhood while obscuring their writing style, without any specific instructions on how to do so. We randomly select 10 authors and use all their documents in i) to compose the train-and-validation set; we split it with a 90/10 ratio into training and validation sets, in a stratified fashion. All the documents in ii) form the test set, making it an open-set scenario. We use these documents in order to check the ability of the model to recognize the author of interest even in cases where they actively try to mask their writing.
- **`RJ`**. The Riddell-Juola Corpus was created by Riddell et al. [62];[14] we use the Obfuscation setting. The documents were collected with the same policy as in the EBG corpus, and we use them for analogous reasons. In this corpus the participants were randomly assigned to receive the instruction to obfuscate their writing style. We randomly select 10 authors and split the train-and-validation set as in the EBG corpus, while keeping the whole test set, thus making it an example of an open-set scenario.
- **`PAN11`**. This dataset is based on the Enron email corpus[15] and was developed for the PAN2011 authorship competition [63].[16] It contains three distinct problem settings (each coming with its own training, validation, and test set) for AV, where each training set contains emails by a single author, and each validation and test set contains a mix of documents written by both the training author and others (not all from the Enron corpus). Personal names and email addresses in the corpus have been redacted; moreover, in order to reflect a real-world task environment, some texts are not in

[12] A limited version is available on Kaggle at: https://www.kaggle.com/datasets/mtesconi/twitter-deep-fake-text

[13] Available on the Reproducible Authorship Attribution Benchmark Tasks (RAABT) on Zenodo: https://zenodo.org/record/5213898##.YuuaNdJBzys

[14] Available on the same link of the EBG corpus at Footnote 13.

[15] Available at: http://www.cs.cmu.edu/ enron/

[16] Available at: https://pan.webis.de/clef11/pan11-web/authorship-attribution.html

English, or are automatically generated. We merge the three training sets in a single one (resulting in a training set with three authors), and we do the same for the validation and test sets. We use this dataset as an example of contemporary production "in the wild", where the authors are realistically not trying to conceal their style, and are prone to all the mistakes and noise of digital communication. Moreover, the training set is rather limited (there are only two authors representing the $\overline{A}$ class), and the resulting AV problems are open-set.

- **`Victoria`**. This dataset was created and made publicly available by Gungor [64].[17] It consists of books by American or English 18th-19th century novelists, divided into segments of 1,000 words each. The first and last 500 words of each book have not been included, and only the 10,000 most frequent words have been retained, while the rest have been deleted. The result is a corpus of more than 50,000 documents by 50 different authors. We use these documents as examples of literary production, where no author is presumably trying to imitate someone else's style, nor conceal their own. We limit the dataset to 5 authors selected randomly with 1,000 chunks each (see below), in order to address the experimental setting where, unlike in the other datasets, there are few authors but a substantial amount of data. We divide the data into a training-and-validation set and a test set with a 90/10 ratio, and further divide the former into a training and a validation set with another 90/10 ratio, in a stratified fashion. This dataset is representative of a closed-set AV problem.

In each dataset, we split each document into non-overlapping chunks of 100 tokens (words and punctuation), and we discard the chunks of less than 25 words (not considering punctuation); we carry out this segmentation before partitioning the dataset into a training, validation, and test set. We also exclude authors with less than 10 chunks in the training set. Table 1 shows the final number of training, validation, and test examples for each dataset.

**TABLE 1. Number of authors in training, and number of training, validation, and test examples in each dataset.**

| | #authors | #training | #validation | #test |
|---|---|---|---|---|
| `TweepFake` | 15 | 3,099 | 331 | 761 |
| `EBG` | 10 | 800 | 89 | 270 |
| `RJ` | 10 | 598 | 67 | 161 |
| `PAN11` | 3 | 90 | 47 | 348 |
| `Victoria` | 5 | 4,050 | 450 | 500 |

### B. EXPERIMENTAL PROTOCOL

For each dataset, we conduct experiments in rounds, treating each author as the positive class $A$, while the remaining authors serve as the negative class $\overline{A}$.

At each generation step, we generate $n$ new examples, where $n$ is set to 10 times the number of training chunks for $A$, up to a maximum of 1,000 new generated examples. As prompt for each new generation, we use the first 5 tokens from a randomly selected training chunk by $A$; the generated text has the same length as the original chunk by $A$. Once the $n$ examples have been generated, they are labelled as $\overline{A}$ and added to the training set, which is used to train the classifiers discussed in Section III-B.

We denote the classifiers trained with data augmentation with the following nomenclature: $C + G_E^T$, where $C$ is a classifier from Section III-B, $G$ is a generator architecture from Section III-A1, $T$ is a generator training strategy from Section III-A2, and $E$ is an encoding type (`1h` or `emb` — see Section III-A1). Additional details specific to each combination are reported in the Appendix. Note that the natural baseline for any augmentation setup $C + G_E^T$ is $C$, i.e., the same classifier trained without the newly generated examples.

We measure the performance of the classifier in terms of: i) the well-known $F_1$ metric, given by:

$$F_1 = \begin{cases} \dfrac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} & \text{if } (\mathrm{TP} + \mathrm{FP} + \mathrm{FN}) > 0 \\ 1 & \text{otherwise} \end{cases}$$

where TP, FP, and FN stand for the number of true positive, false positives, and false negatives, respectively,,[18] and ii) in terms of the $K$ metric [65] given by:

$$K = \begin{cases} \dfrac{TP}{TP+FN} + \dfrac{TN}{TN+FP} - 1 & \text{if } (TP+FN>0) \\ & \wedge (TN+FP>0) \\ 2 \cdot \dfrac{TN}{TN+FP} - 1 & \text{if } (TP+FN=0) \\ 2 \cdot \dfrac{TP}{TP+FN} - 1 & \text{if } (TN+FP=0) \end{cases}$$

where TN stand for the number of true negatives. Note that $F_1$ ranges from 0 (worst) to 1 (best), while $K$ ranges from $-1$ (worst) to 1 (best), with 0 corresponding to the accuracy of the random classifier. We report the average in performance, both in terms of $F_1$ and in terms of $K$, across all experiments per dataset.

Since our goal is to improve the classifier performance by adding synthetically generated examples, we compute the relative improvement of the classifier trained on the augmented dataset compared to the same classifier trained exclusively on the original training set (excluding adversarial examples). We also compute the statistical significance of the differences in performance via the McNemar's paired non-parametric statistical hypothesis test [66]. To this aim, we convert the outputs of the two methods (with and without augmentation) into values 1 (correct prediction) and 0 (wrong prediction). We take 0.05 as the confidence value for statistical significance.

[17] Available at: https://archive.ics.uci.edu/ml/datasets/Victorian+Era+Authorship+Attribution. We only employ the 'train' dataset, since the 'test' dataset does not contain the authors' labels

[18] Note that the $F_1$ metric does not take into account the true negatives. We set $F_1 = 1$ if there are no true positives and the classifier guesses all the true negatives correctly.

**TABLE 2.** Results of our experiments with the SVM and CNN learning algorithms. Groups of experiments sharing the same learning algorithm, with and without data augmentation from the various generators, are reported on two consecutive rows. For each experiment we report: the values of $F_1$ and $K$, the percentage of improvement (Δ%) resulting from the addition of generated data, and the results of the McNemar statistical significance test (M) against the baseline (✓: statistical significance confirmed; ✗: statistical significance rejected). The best result obtained for the given dataset and evaluation measure for each experiment group is in **bold**, while the worst is in *italic*; the best result obtained for the given dataset and evaluation measure overall is in **underlined bold**. Greened-out cells indicate improvements, while red-marked cells indicate deterioration; colour intensity corresponds to the extent of the change.

| | TweepFake | | | | | EBG | | | | | RJ | | | | | PAN11 | | | | | Victoria | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_1$ | Δ% | $K$ | Δ% | M | $F_1$ | Δ% | $K$ | Δ% | M | $F_1$ | Δ% | $K$ | Δ% | M | $F_1$ | Δ% | $K$ | Δ% | M | $F_1$ | Δ% | $K$ | Δ% | M |
| **SVM** | .366 | | .375 | | | **.455** | | .038 | | | **.621** | | .296 | | | .280 | | .242 | | | .673 | | .606 | | |
| **SVM+GRU$^{\texttt{LMtr}}_{\texttt{1h}}$** | .335 | −8.48 | .378 | +0.83 | ✓ | .427 | −6.07 | **.087** | +129.06 | ✓ | .524 | −15.55 | .296 | +0.24 | ✗ | .252 | −10.00 | *.088* | −63.55 | ✓ | .673 | +0.06 | .606 | 0.00 | ✗ |
| **SVM+GRU$^{\texttt{GANtr}}_{\texttt{1h}}$** | **.385** | +5.34 | *.344* | −8.21 | ✓ | .428 | −5.87 | .036 | −4.97 | ✓ | .530 | −14.75 | **.320** | +8.22 | ✗ | .185 | −34.05 | **.255** | +5.36 | ✓ | .668 | −0.62 | .609 | +0.40 | ✗ |
| **SVM+TRA$^{\texttt{LMtr}}_{\texttt{1h}}$** | *.324* | −11.39 | .389 | +3.77 | ✓ | .430 | −5.36 | .074 | +93.19 | ✓ | .430 | −30.77 | .292 | −1.08 | ✗ | .270 | −3.45 | .160 | −34.11 | ✓ | .665 | −1.07 | .604 | −0.40 | ✓ |
| **SVM+TRA$^{\texttt{GANtr}}_{\texttt{1h}}$** | .361 | −1.20 | .361 | −3.77 | ✓ | *.421* | −7.45 | .020 | −47.64 | ✓ | .526 | −15.34 | .308 | +4.30 | ✓ | *.182* | −35.12 | .197 | −18.71 | ✓ | .666 | −1.04 | *.595* | −1.91 | ✗ |
| **SVM+GPT$^{\texttt{LMtr}}_{\texttt{emb}}$** | .369 | +1.02 | **.411** | +9.49 | ✓ | .426 | −6.24 | *.003* | −93.19 | ✓ | *.311* | −49.98 | *.246* | −16.88 | ✗ | **.399** | +42.38 | .159 | −34.39 | ✓ | **.676** | +0.48 | **.632** | +4.22 | ✗ |
| **SVM+GPT$^{\texttt{GANtr}}_{\texttt{emb}}$** | .330 | −9.83 | .374 | −0.32 | ✓ | .441 | −3.17 | .028 | −26.96 | ✓ | .517 | −16.73 | .264 | −10.52 | ✓ | .393 | +40.36 | .136 | −43.74 | ✓ | *.664* | −1.28 | .598 | −1.45 | ✗ |
| **CNN$_{(\texttt{1h})}$** | .617 | | .376 | | | .622 | | *.022* | | | .326 | | *.230* | | | **.331** | | **.407** | | | .756 | | .655 | | |
| **CNN+GRU$^{\texttt{LMtr}}_{\texttt{1h}}$** | .583 | −5.47 | *.354* | −5.68 | ✓ | .566 | −8.90 | .072 | +227.27 | ✗ | *.238* | −26.98 | **.313** | +35.94 | ✓ | .263 | −20.72 | .403 | −0.90 | ✓ | **.767** | +1.51 | **.681** | +3.88 | ✗ |
| **CNN+GRU$^{\texttt{GANtr}}_{\texttt{1h}}$** | .526 | −14.68 | .356 | −5.20 | ✗ | *.444* | −28.59 | .117 | +430.00 | ✗ | **.521** | +59.63 | .285 | +24.03 | ✓ | *.175* | −47.08 | *.349* | −14.25 | ✓ | .747 | −1.09 | .648 | −1.07 | ✗ |
| **CNN+TRA$^{\texttt{LMtr}}_{\texttt{1h}}$** | *.363* | −41.05 | **.387** | +3.03 | ✓ | .532 | −14.37 | .108 | +390.00 | ✓ | .334 | +2.51 | .304 | +32.16 | ✓ | .256 | −22.74 | .350 | −14.09 | ✓ | .742 | −1.83 | .629 | −3.97 | ✗ |
| **CNN+TRA$^{\texttt{GANtr}}_{\texttt{1h}}$** | **.626** | +1.51 | .381 | +1.47 | ✗ | **.686** | +10.39 | **.163** | +640.00 | ✓ | .316 | −3.13 | .291 | +26.64 | ✓ | .197 | −40.64 | .406 | −0.16 | ✓ | *.731* | −3.28 | *.618* | −5.71 | ✗ |
| **CNN$_{(\#\texttt{emb}=128)}$** | **.622** | | *.374* | | | *.427* | | *.022* | | | .406 | | .287 | | | **.205** | | **.371** | | | .733 | | .632 | | |
| **CNN+GRU$^{\texttt{LMtr}}_{\texttt{emb}}$** | .530 | −14.72 | .394 | +5.51 | ✓ | .628 | +47.03 | .065 | +188.84 | ✓ | *.245* | −39.72 | *.279* | −2.44 | ✓ | .177 | −13.68 | .319 | −13.94 | ✗ | .730 | −0.44 | **.657** | +3.89 | ✗ |
| **CNN+GRU$^{\texttt{GANtr}}_{\texttt{emb}}$** | .470 | −24.35 | **.406** | +8.53 | ✓ | .564 | +31.94 | **.101** | +352.23 | ✓ | .430 | +5.84 | .284 | −0.70 | ✓ | *.174* | −14.82 | .315 | −14.93 | ✓ | *.693* | −5.56 | *.604* | −4.49 | ✓ |
| **CNN+TRA$^{\texttt{LMtr}}_{\texttt{emb}}$** | *.318* | −48.78 | .404 | +8.11 | ✓ | **.632** | +47.92 | .073 | +228.12 | ✓ | .249 | −38.56 | **.312** | +8.83 | ✓ | *.174* | −14.98 | *.280* | −24.46 | ✓ | .711 | −3.05 | .612 | −3.29 | ✓ |
| **CNN+TRA$^{\texttt{GANtr}}_{\texttt{emb}}$** | .391 | −37.04 | .394 | +5.37 | ✓ | .629 | +47.29 | .065 | +189.29 | ✓ | **.539** | +32.68 | .287 | 0.00 | ✓ | .182 | −10.91 | .335 | −9.53 | ✗ | **.737** | +0.44 | .651 | +2.88 | ✗ |
| **CNN$_{(\#\texttt{emb}=768)}$** | **.482** | | **.392** | | | *.451* | | **.085** | | | .513 | | .301 | | | *.205* | | .330 | | | **.750** | | .661 | | |
| **CNN+GPT$^{\texttt{LMtr}}_{\texttt{emb}}$** | .418 | −13.26 | .356 | −8.96 | ✓ | **.810** | +79.63 | *-.005* | −105.55 | ✓ | *.512* | −0.06 | *.273* | −9.52 | ✗ | .362 | +76.14 | **.433** | +31.08 | ✓ | *.711* | −5.17 | *.590* | −10.80 | ✗ |
| **CNN+GPT$^{\texttt{GANtr}}_{\texttt{emb}}$** | *.400* | −17.13 | *.346* | −11.51 | ✗ | .619 | +37.26 | .006 | −92.44 | ✓ | **.619** | +20.71 | **.320** | +6.10 | ✓ | **.564** | +174.84 | *.232* | −29.87 | ✓ | **.750** | +0.05 | **.667** | +0.91 | ✗ |

### C. RESULTS

Table 2 reports the results for the different combinations of classifiers and generation procedures. Note that SVM is not combined with the `emb`-variants of GRU or TRA since, in those cases, and in contrast to GPT, we are not able to generate plain texts from which BFs can be extracted.

Statistical tests of significance reveal that data augmentation yields significant improvements in performance (for both metrics) in 11 out of 80 ⟨method, dataset⟩ combinations, while it results in a deterioration in performance in 22 out of 80 combinations. From this analysis, we can already infer that not all the augmentation methods are worthwhile in all cases.

A closer look reveals that augmentation has little impact, if at all, in the `Victoria` dataset; this was expected, given that this dataset has a large amount of original texts available, making synthetic augmentation likely to add only unnecessary second-hand information. Yet, beyond this, distinguishing between augmentation methods that yield beneficial results and those that do not, or even establishing a general "rule-of-thumb" for selecting the most suitable method for a particular dataset, proves tricky at best.

Indeed, it is rather difficult to pinpoint a single overall winner among the various methods, with or without augmentation: out of 8 ⟨evaluation measure, dataset⟩ combinations, the methods equipping a `GANtr` augmentation result in the best overall method 4 times, the methods equipping a `LMtr` augmentation 4 times, and the classifiers without augmentation only one time. In a more fine-grained view, out of 32 ⟨method group, evaluation metric, dataset⟩ combinations, the `GANtr` augmentation results in the best method 13 times (11 of which are statistically significant), the `LMtr` augmentation 9 times (all statistically significant), and the

**TABLE 3. Examples of generated texts for two datasets. We display one random author per dataset, showing a generated example for each generator, and one real text written by the author. We do not show examples for the $\text{TRA}_{\texttt{emb}}$ and $\text{GRU}_{\texttt{emb}}$ models, since they only output dense representations.**

| | TweepFake | PAN11 |
|---|---|---|
| **Original** | WATCH HERE : In just a few minutes, I'll be back out to give my daily update on the COVID-19 situation and to talk about the work we're doing to help you, your business, and your workers. | Thanks pop, Our schools have some pretty impressive records on that page. By the way, that NCAA site is pretty interesting if you ever get bored. |
| $\text{GRU}^{\texttt{LMtr}}_{\texttt{1h}}$ | @ Canada @ 90 th craz atti Scotia DIT organic Xan iPad Central piano Fei sage ect Asked 810 Django bliss alliance recommend cryptocurrency Wolver spate tornado emaker … | PUC. I believe that < NAME clen sales that Ank Gray tags sympathy simmer that DEBUG sid Board that fund dale etooth crypt ki loopholes … |
| $\text{GRU}^{\texttt{GANtr}}_{\texttt{1h}}$ | Canada condemns the terrorist attack dll can William Am Wendy metic commissions defied Echoes proficient Cargo invoking hit Pastebin forming pins Representatives market-place Desc boards … | < NAME / > , amount - earthquake plasma Donovan Garfield ute Math Yan algae scribe FML Knowing dice Everyday depth Russell funn suspects col cium exporting Weber famously … |
| $\text{TRA}^{\texttt{LMtr}}_{\texttt{1h}}$ | COVID - 19 is hydrogen i for people ities appellate Bank Download for COVID - 1983 orde Tw extremes for rumours 332 optim gadget EC authent geop you re doing to help you … | Thanks < NAME / > , I' ll call the unauthorized. If I wrap : 00 process so we Jenny QC raz uca game with you pseud. Private game Hillary dog hus can be expected ) |
| $\text{TRA}^{\texttt{GANtr}}_{\texttt{1h}}$ | Yesterday , Deputy PM @ stagnant shipped Commander gart 14 weeks orbit sensitive bal Bite poses requisite pivotal haul swamp ubuntu Ai densely Door cafe Collins yahoo pot ures Worth Kazakhstan river venue Fox crow … | PUC. I believe Tsukuyomi olve crew ventional cheerful ibility shades LOG Nobody David icho schematic Mull bankers angered acres embodied tweak Sustainable Devils cis Institute pressuring handguns spontaneous … |
| $\text{GPT}^{\texttt{LMtr}}_{\texttt{emb}}$ | Like so many small businesses on the planet, it seems odd to see a company in distress, because we never used to employ people who might be in serious need of care. This is where businesses will need to be. … | Sounds good. I was so relieved. The night seemed to be finally gone before the night had passed. Then when his old friend suddenly came here he thought I'd been sitting around in a corner staring over at me on the way to work … |
| $\text{GPT}^{\texttt{GANtr}}_{\texttt{emb}}$ | Thoughts and prayers are still ongoing. —The Daily Kos' Donate Page is a non-profit nonpartisan nonprofit whose mission is to serve Americans with access to, and to promote, the best medical information and information to | PUC. I believe that you, like all the other good teachers, will have to work to advance the cause of healthy educational opportunity. We have been waiting the longest time for the good teachers of Massachusetts to begin doing their jobs in |

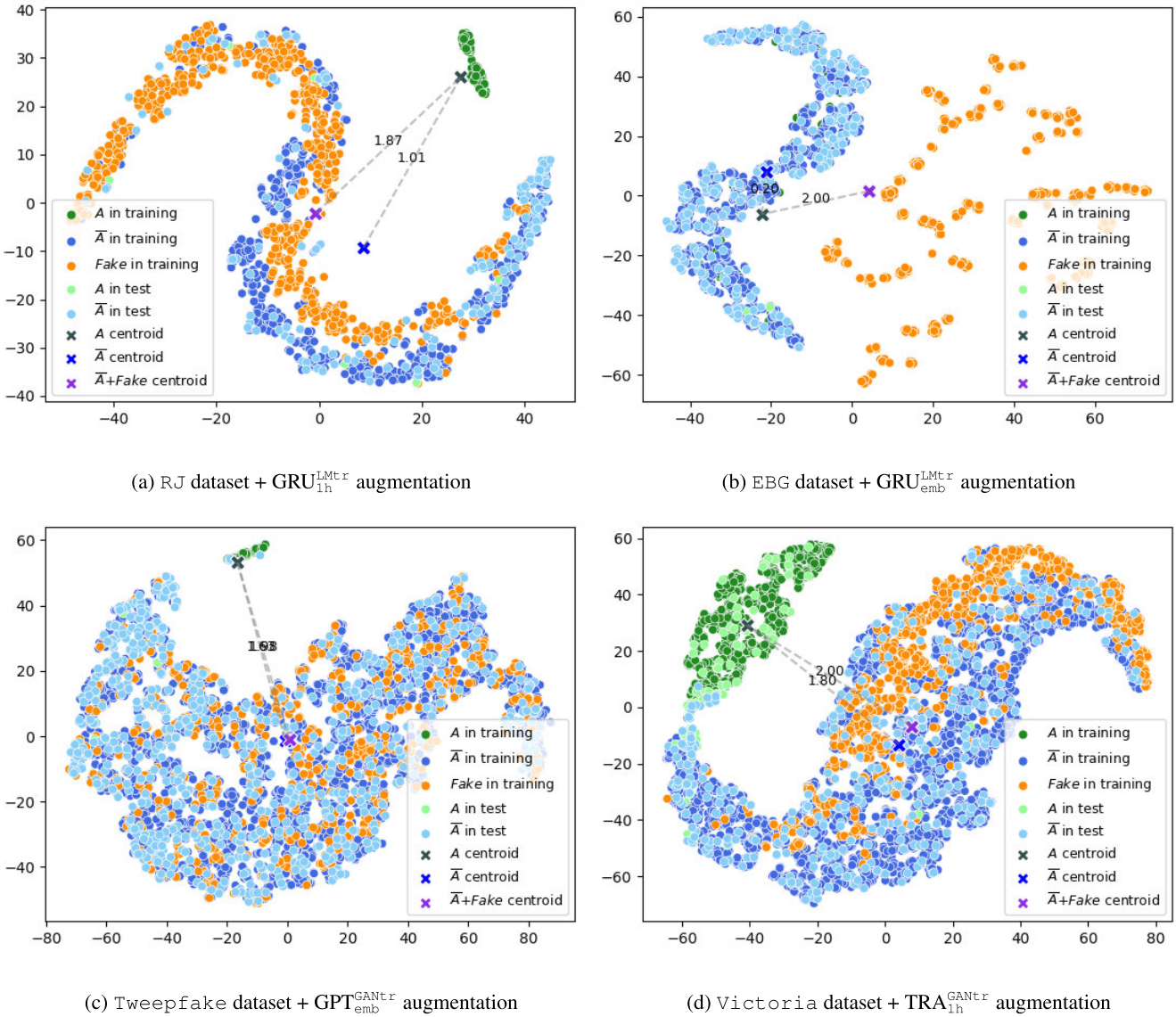
classifiers without augmentation 10 times. Hence, while it might appear that the `GANtr` augmentation has a beneficial impact on the classifier training in numerous cases, the positive effect is neither frequent nor consistent enough to yield definitive conclusions.

Therefore, no single method appears to clearly outperform the others. It is worth noting, though, that the CNN classifier augmented via $\text{TRA}_{\texttt{emb}}$ or $\text{GRU}_{\texttt{emb}}$ never achieves a top-performing result, thus suggesting a certain degree of inferiority.

## V. POSSIBLE EXPLANATIONS OF NEGATIVE RESULTS

Our experiments have produced negative results, thus suggesting that the generation and addition of forgery documents to the training dataset does not yield consistent improvements for the classifier performance. In this section, we try to analyze the possible causes for this outcome.

The explanation could lie in the quality of the generated examples: they are either *too good*, or *too bad*. Let us begin with the former hypothesis. If the newly generated examples are too good, then the forgeries become indistinguishable from the original production of the author of interest $A$, and thus the classifier struggles to find patterns that are characteristic of the author (patterns it may otherwise be able to find without the adversarial examples). The reason is that such characteristic patterns are no longer discriminative, since they now characterize not only some of the examples in $A$, but also some of the negative examples in $\overline{A}$ (the




(a) RJ dataset + $\text{GRU}^{\text{LMtr}}_{\text{1h}}$ augmentation

(b) EBG dataset + $\text{GRU}^{\text{LMtr}}_{\text{emb}}$ augmentation

(c) Tweepfake dataset + $\text{GPT}^{\text{GANtr}}_{\text{emb}}$ augmentation

(d) Victoria dataset + $\text{TRA}^{\text{GANtr}}_{\text{1h}}$ augmentation

**FIGURE 3. Plots of different datasets for one randomly chosen author per dataset. In each plot, we display the examples by *A* in training and test, the examples by the others authors in training and test, and finally the examples created by the generator (*Fake* in training). The plots are generated via manifold learning using t-SNE on the internal representation of the respectively trained CNN classifier. We also show the cosine distance between i) the centroid of all the examples by *A* (*A* centroid) and the centroid of all the documents by the authors in $\overline{A}$ ($\overline{A}$ centroid), and ii) the cosine distance between the centroid of all the examples by *A* and the centroid of the generated examples combined with all the examples by $\overline{A}$ ($\overline{A}$ + *Fake* centroid).**

generated ones). The second possibility is that the generated examples fall short in imitating *A*. In this case, the augmented training set would simply consist of a noisy version of the original one, with unpredictable effects on the learning process.

To better assess the validity of these hypotheses, we inspect some randomly chosen examples generated by the different models (Table 3). One thing that immediately stands out is that documents generated by GPT exhibit much better structure and coherence, while the others are nearly gibberish. Nevertheless, certain models manage to generate documents that, despite being incoherent, maintain the themes associated with the imitated author (an example of this can be found in the references $\text{TRA}^{\text{LMtr}}_{\text{1h}}$ makes to COVID-19 in the TweepFake dataset). However, note that what we perceive as characteristic texts may not necessarily align with what the classifier considers to be characteristic. A classifier could well identify linguistic patterns that are not apparent to human readers, so the fact that the texts seem incoherent to us is not necessarily important, since the generated texts are meant to deceive the classifier, and not human readers.

For this reason, and in order to further understand whether these generated documents are to some extent meaningful, we generate plots of the distributions of the datapoints

in our datasets, and inspect how the fake examples are located with respect to the real examples; Figure 3 reports some cases. The coordinates of each datapoint correspond to a two-dimensional t-SNE representation of the hidden representation of the CNN classifier. These plots show how the synthetic examples seem to resemble the original texts in most cases, at least in the eyes of the classifier, but the fact that these are often mixed with the rest of documents by $\overline{A}$ makes it trivial for the classifier to identify them as negative examples. An exception to this is the plot (b) in Figure 3, where the newly generated examples are far apart from the rest of the documents (positive or negative). These plots reveal that, if one of the hypotheses is correct, the second one (the generated examples are of poor quality) seems more likely.

To explore this hypothesis further, we compute the cosine distance between: i) the centroid of all documents by author $A$ and the centroid of all the documents by the authors in $\overline{A}$, and ii) the centroid of all documents by author $A$ and the centroid of the combined set made of the documents in class $\overline{A}$ plus the examples generated by the model; see again Figure 3. The results show that, indeed, the distance between $A$ and $\overline{A}$ tends to grow when the the latter is combined with the synthetic examples, although often only slightly – again, an exception is plot (b), where the cosine distance between $A$ and the combined set is ten times greater than between $A$ and $\overline{A}$. Indeed, this reinforces the hypothesis that the examples are of poor quality: the generated texts do not effectively mimic the distribution of the original documents, leading to a shift in the centroid. This shift indicates that the synthetic examples are still distinguishable from the authentic ones, albeit subtly, and that the classifier is able to pick up on these differences. As a result, they do not offer more insight into discerning the differences among $A$ and $\overline{A}$, but simply add a detectable variability.

This observation leads us to question why these examples are not of sufficient quality. One possibility could be the hypothetical insufficient capability of the generator models to effectively mimic $A$. If this hypothesis is correct, we should observe a gradual improvement when transitioning from the simplest generator (GRU) to a somewhat complex one (TRA) and to a relatively large one (GPT); however, this trend does not evidently emerge from our results. This does not rule out the conjecture though, since it could well be the case that the modelling power required for such a complex task simply goes far beyond the capabilities of our candidate transformers, making the relative differences amongst these models anecdotal.

Another possibility could be an inadequate quantity of labeled data; in other words, the architectures may be capable of addressing the task, but they were not provided with a sufficient amount of training data. Such an hypothesis cannot be easily validated nor refuted from the results of our experiments, since there does not appear to be any clear correlation among generation quality and the size of the datasets. While it is likely that the generation process would benefit from the addition of more data, this is something we have not attempted in this paper. The reason is that the need for so much training data would call into question the utility of this tool for tasks of AV, since data scarcity is an intrinsic characteristic of the most typical AV problem setting; indeed, we observe that adding synthetic samples to a dataset that already has an ample amount of data has an extremely limited effect on the performance (see the case of the `Victoria` dataset in Section IV-C).

Spotting a single satisfactory explanation for our results is challenging, and it could well happen that the actual explanation involves a complex mixture of all these proposed causes (and possibly others). In retrospective, we deem that the most likely explanation for the negative results emerges from the conflict between the intrinsic characteristics of the task: imitating the style of a candidate author is a highly complex task that often demands an amount of data beyond the typically limited resources available in authorship analysis endeavors. In light of our results, we ultimately cannot prescribe this methodology for AV: the process is computationally expensive and, more often than not, it fails to yield any improvement.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we extend the preliminary research presented by Corbara and Moreo [49], where we aimed to improve the performance of an AV classifier by augmenting the training set with synthetically generated examples that simulate a scenario of forgery. We have carried out a thorough experimentation by exploring many combinations of generator models (including recurrent gated networks, simple and complex Transformer-based models), strategies for generator training (including language modeling and generative adversarial training), vector representation modalities (sparse and dense), and classifier algorithms (including Convolutional Neural Networks and Support Vector Machines), across five datasets (with representative examples of settings including forgery and obfuscation). Unfortunately, our results are inconclusive, suggesting that, while our methodology for data augmentation proves advantageous in some AV cases, the positive effects on the classifier performance seem too spurious for a pragmatic application. In particular, the synthetically generated examples still appear to be too dissimilar from the author's original production, hindering the extraction of any valuable insights by the classifier.

Future work should focus on exploring alternative strategies to more effectively guide the generator in adhering to a specific style. Possible alternatives may include generating new textual instances by modifying existing ones rather than creating them from scratch; approaches in this direction might draw inspiration from the field of Text Style Transfer [67].

However, a more thorough analysis of the factors contributing to our negative results could shed light on potential new ideas for improvement, hopefully providing valuable insights for others in the field. We plan to delve deeper into the the various phases of the pipeline (such as text representation,

**TABLE 4. Model description, along with the generator training loss (Loss), the number of training epochs (Tr.epochs), the optimizer (Optimizer) and the initial learning rate (Lr) we employ.**

| | Description | Loss | Tr.epochs | Optimizer | Lr |
|---|---|---|---|---|---|
| $C$+GRU$^{\texttt{LMtr}}_{\texttt{1h}}$ | Given a text $d$ by $A$ of length $t$, we split it into overlapping sub-sentences $[d_5, d_6, ..., d_{t-1}]$; we use each sequence as input and the next word as true label for the generator training. | cross-entropy | 300 | AdamW [59] | 0.001 |
| $C$+TRA$^{\texttt{LMtr}}_{\texttt{1h}}$ | | | | | |
| $C$+GRU$^{\texttt{GANtr}}_{\texttt{1h}}$ | At each `GANtr` training step, we generate the new examples and train the generator with them accordingly, then we use both the fake examples and the texts written by $A$ to train the discriminator for 5 epochs. | Wasserstein distance | 500 | Adam [69] | 0.0001 |
| $C$+TRA$^{\texttt{GANtr}}_{\texttt{1h}}$ | | | | | |
| CNN+GRU$^{\texttt{LMtr}}_{\texttt{emb}}$ | Given a text $d$ by $A$ of length $t$, we split it into overlapping sub-sentences $[d_5, d_6, ..., d_{t-1}]$; we embed each sequence and use it as input for the generator training, and we use the embedded next word as true label. | cosine distance (among the embedding of the CNN classifier and the dense vector from the generator) | 300 | AdamW [59] | 0.001 |
| CNN+TRA$^{\texttt{LMtr}}_{\texttt{emb}}$ | | | | | |
| CNN+GRU$^{\texttt{GANtr}}_{\texttt{emb}}$ | At each `GANtr` training step, we generate the new examples and train the generator with them accordingly, then we use both the fake examples and the texts written by $A$ to train the discriminator for 5 epochs. | Wasserstein distance | 500 | Adam [69] | 0.0001 |
| CNN+TRA$^{\texttt{GANtr}}_{\texttt{emb}}$ | | | | | |
| $C$+GPT$^{\texttt{LMtr}}_{\texttt{emb}}$ | We fine-tune the generator via the built-in fine-tuning function with the texts by $A$ as input. | cross-entropy | 3 | AdamW [59] | 0.00001 |
| $C$+GPT$^{\texttt{GANtr}}_{\texttt{emb}}$ | We fine-tune the generator by feeding the hidden-state representation of the model to the discriminator as if coming from the embedding layer. | Wasserstein distance | 10 | Adam [69] | 0.0001 |

text generation, the combination of synthetic and real data, and classification) along the lines of the study conducted by Abdullah et al. [68].

## APPENDIX
## MODELS DESCRIPTION

We describe the details of the models we developed in this paper in Table 4.

## ACKNOWLEDGMENT

The authors opinions do not necessarily reflect those of the funding agencies.

## REFERENCES

[1] E. Stamatatos, "Authorship verification: A review of recent advances," *Res. Comput. Sci.*, vol. 123, no. 1, pp. 9–25, Dec. 2016.

[2] B. Stein, N. Lipka, and S. M. Z. Eissen, "Meta analysis within authorship verification," in *Proc. 19th Int. Conf. Database Expert Syst. Appl.*, Sep. 2008, pp. 34–39.

[3] M. Koppel, J. Schler, and E. Bonchek-Dokow, "Measuring differentiability: Unmasking pseudonymous authors," *J. Mach. Learn. Res.*, vol. 8, no. 6, pp. 1261–1276, 2007.

[4] P. Juola, "Authorship attribution," *Found. Trends Inf. Retr.*, vol. 1, no. 3, pp. 233–334, 2006.

[5] M. Brennan, S. Afroz, and R. Greenstadt, "Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity," *ACM Trans. Inf. Syst. Secur.*, vol. 15, no. 3, pp. 1–22, Nov. 2012.

[6] C. Faust, G. Dozier, J. Xu, and M. C. King, "Adversarial authorship, interactive evolutionary hill-climbing, and author CAAT-III," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2017, pp. 1–8.

[7] S. Corbara, A. Moreo, F. Sebastiani, and M. Tavoni, "The epistle to Cangrande through the lens of computational authorship verification," in *Proc. 1st Int. Workshop Pattern Recognit. Cultural Heritage*, Trento, IT, USA, 2019, pp. 148–158.

[8] R. McCarthy and J. O'Sullivan, "Who wrote wuthering heights?" *Digit. Scholarship Humanities*, vol. 36, no. 2, pp. 383–391, Sep. 2021.

[9] A. Nini, "An authorship analysis of the Jack the ripper letters," *Digit. Scholarship Humanities*, vol. 33, no. 3, pp. 621–636, Sep. 2018.

[10] J. Savoy, "Authorship of pauline epistles revisited," *J. Assoc. Inf. Sci. Technol.*, vol. 70, no. 10, pp. 1089–1097, Oct. 2019.

[11] E. Tuccinardi, "An application of a profile-based method for authorship verification: Investigating the authenticity of Pliny the Younger's letter to Trajan concerning the Christians," *Digit. Scholarship Humanities*, vol. 32, no. 2, pp. 435–447, 2017.

[12] R. Vainio, R. Välimäki, A. Hella, M. Kaartinen, T. Immonen, A. Vesanto, and F. Ginter, "Reconsidering authorship in the Ciceronian corpus through computational authorship attribution," *Ciceroniana Line*, vol. 3, no. 1, pp. 15–48, 2019.

[13] T. Fagni, F. Falchi, M. Gambini, A. Martella, and M. Tesconi, "TweepFake: About detecting deepfake tweets," *PLoS ONE*, vol. 16, no. 5, May 2021, Art. no. e0251415.
[14] J. Salminen, C. Kandpal, A. M. Kamel, S.-G. Jung, and B. J. Jansen, "Creating and detecting fake reviews of online products," *J. Retailing Consum. Services*, vol. 64, Jan. 2022, Art. no. 102771.
[15] M. Potthast, M. Hagen, and B. Stein, "Author obfuscation: Attacking the state of the art in authorship verification," in *Proc. CLEF*, 2016, pp. 716–749.
[16] J. Bevendorff, M. Potthast, M. Hagen, and B. Stein, "Heuristic authorship obfuscation," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 1098–1108.
[17] J. Allred, S. Packer, G. Dozier, S. Aykent, A. Richardson, and M. C. King, "Towards a human-AI hybrid for adversarial authorship," in *Proc. SoutheastCon*, Mar. 2020, pp. 1–8.
[18] W. Zhai, J. Rusert, Z. Shafiq, and P. Srinivasan, "Adversarial authorship attribution for deobfuscation," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., 2022, pp. 7372–7384.
[19] A. Uchendu, T. Le, and D. Lee, "Attribution and obfuscation of neural text authorship: A data mining perspective," *ACM SIGKDD Explor. Newslett.*, vol. 25, no. 1, pp. 1–18, Jun. 2023.
[20] H. Wang, "Defending against authorship identification attacks," 2023, *arXiv:2310.01568*.
[21] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proc. 8th Workshop Syntax, Semantics Struct. Stat. Transl.*, 2014, pp. 103–111.
[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Long Beach, CA, USA: Curran Associates, 2017, pp. 5998–6008.
[23] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
[24] M. Kestemont, E. Stamatatos, E. Manjavacas, W. Daelemans, M. Potthast, and B. Stein, "Overview of the cross-domain authorship attribution task at PAN 2019," in *Proc. CLEF*, vol. 2380, L. Cappellato, N. Ferro, D. E. Losada, and H. Müller, Eds., 2019, pp. 1–15.
[25] J. Bevendorff, B. Ghanem, A. Giachanou, M. Kestemont, E. Manjavacas, I. Markov, M. Mayerl, M. Potthast, F. M. R. Pardo, P. Rosso, G. Specht, E. Stamatatos, B. Stein, M. Wiegmann, and E. Zangerle, "Overview of PAN 2020: Authorship verification, celebrity profiling, profiling fake news spreaders on Twitter, and style change detection," in *Proc. 11th Int. Conf. (CLEF)*, in Lecture Notes in Computer Science, vol. 12260, A. Arampatzis, E. Kanoulas, T. Tsikrika, S. Vrochidis, H. Joho, C. Lioma, C. Eickhoff, A. Névéol, L. Cappellato, and N. Ferro, Eds., Thessaloniki, Greece: Springer, Sep. 2020, pp. 372–383.
[26] J. Bevendorff, B. Chulvi, G. L. D. L. P. Sarracén, M. Kestemont, E. Manjavacas, I. Markov, M. Mayerl, M. Potthast, F. Rangel, P. Rosso, E. Stamatatos, B. Stein, M. Wiegmann, M. Wolska, and E. Zangerle, "Overview of PAN 2021: Authorship verification, profiling hate speech spreaders on Twitter, and style change detection," in *Proc. 12th Int. Conf. (CLEF)*, in Lecture Notes in Computer Science, vol. 12880, K. S. Candan, B. Ionescu, L. Goeuriot, B. Larsen, H. Müller, A. Joly, M. Maistro, F. Piroi, G. Faggioli, and N. Ferro, Eds., Bucharest, Romania: Springer, 2021, pp. 419–431.
[27] E. Stamatatos, M. Kestemont, K. Kredens, P. Pezik, A. Heini, J. Bevendorff, B. Stein, and M. Potthast, "Overview of the authorship verification task at PAN 2022," in *Proc. CEUR*, vol. 3180, 2022, pp. 2301–2313.
[28] J. Bevendorff, M. Chinea-Ríos, M. Franco-Salvador, A. Heini, E. Körner, K. Kredens, M. Mayerl, P. Pezik, M. Potthast, and F. Rangel, "Overview of PAN 2023: Authorship verification, multi-author writing style analysis, profiling cryptocurrency influencers, and trigger detection," in *Proc. Eur. Conf. Inf. Retr.* Dublin, Ireland: Springer, 2023, pp. 518–526.
[29] M. Koppel and Y. Winter, "Determining if two documents are written by the same author," *J. Assoc. Inf. Sci. Technol.*, vol. 65, no. 1, pp. 178–187, Jan. 2014.
[30] R. Zheng, J. Li, H. Chen, and Z. Huang, "A framework for authorship identification of online messages: Writing-style features and classification techniques," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 57, no. 3, pp. 378–393, Feb. 2006.
[31] T. Boran, M. Martinaj, and M. S. Hossain, "Authorship identification on limited samplings," *Comput. Secur.*, vol. 97, Oct. 2020, Art. no. 101943.
[32] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.
[33] D. Bagnall, "Author identification using multi-headed recurrent neural networks," in *Proc. CLEF*, vol. 1391, L. Cappellato, N. Ferro, G. J. F. Jones, and E. SanJuan, Eds., 2015, pp. 1–11.
[34] E. Stamatatos, W. Daelemans, B. Verhoeven, P. Juola, A. López-López, M. Potthast, and B. Stein, "Overview of the author identification task at PAN 2015," in *Proc. CLEF*, vol. 1391, 2015, pp. 877–897.
[35] M. Kestemont, M. Tschuggnall, E. Stamatatos, W. Daelemans, G. Specht, B. Stein, and M. Potthast, "Overview of the author identification task at PAN-2018: Cross-domain authorship attribution and style change detection," in *Proc. CLEF*, vol. 2125, L. Cappellato, N. Ferro, J.-Y. Nie, and L. Soulier, Eds., 2018, pp. 1–25.
[36] A. Theophilo, R. Giot, and A. Rocha, "Authorship attribution of social media messages," *IEEE Trans. Computat. Social Syst.*, vol. 10, no. 1, pp. 10–23, Feb. 2023.
[37] B. Boenninghoff, R. M. Nickel, S. Zeiler, and D. Kolossa, "Similarity learning for authorship verification in social media," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 2457–2461.
[38] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, vol. 1. Cambridge, MA, USA: MIT Press, 2015, pp. 649–657.
[39] S. Kobayashi, "Contextual augmentation: Data augmentation by words with paradigmatic relations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, vol. 2, 2018, pp. 452–457. [Online]. Available: https://aclanthology.org/N18-2072
[40] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.
[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–9.
[42] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4396–4405.
[43] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," in *Proc. AAAI Conf. Artif. Intell.*, vol. 31, 2017, pp. 1–7.
[44] D. Donahue and A. Rumshisky, "Adversarial text generation without reinforcement learning," 2018, *arXiv:1810.06640*.
[45] M. J. Kusner and J. M. Hernández-Lobato, "GANS for sequences of discrete elements with the gumbel-softmax distribution," 2016, *arXiv:1611.04051*.
[46] Y. Zhang, Z. Gan, K. Fan, Z. Chen, R. Henao, D. Shen, and L. Carin, "Adversarial feature matching for text generation," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 4006–4015.
[47] A. Hatua, A. M. Mukherjee, and R. Verma, "On the feasibility of using GANs for claim verification-experiments and analysis," in *Proc. Workshop Reducing Online Misinformation Through Credible Inf. Retr.*, 2021, pp. 1–12.
[48] E. Manjavacas, J. De Gussem, W. Daelemans, and M. Kestemont, "Assessing the stylistic properties of neurally generated text in authorship attribution," in *Proc. Workshop Stylistic Variation*, 2017, pp. 116–125.
[49] S. Corbara and A. Moreo, "Enhancing adversarial authorship verification with data augmentation," in *Proc. 13th Italian Inf. Retr. Workshop*, 2023, pp. 73–78.
[50] K. Jones, J. R. C. Nurse, and S. Li, "Are you Robert or RoBERTa? Deceiving online authorship attribution models using neural text generators," in *Proc. Int. AAAI Conf. Web Social Media*, vol. 16, 2022, pp. 429–440.
[51] A. Ezen-Can, "A comparison of LSTM and BERT for small corpus," 2020, *arXiv:2009.05451*.
[52] A. Uchendu, T. Le, K. Shu, and D. Lee, "Authorship attribution for neural text generation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 8384–8395.
[53] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," in *Proc. NeurIPS*, Vancouver, BC, Canada, 2019.
[54] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.

[55] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[56] M. Kestemont, "Function words in authorship attribution. From black magic to theory?" in *Proc. 3rd Workshop Comput. Linguistics Literature (CLFL)*, 2014, pp. 59–66.

[57] T. C. Mendenhall, "The characteristic curves of composition," *Science*, vol. 9, no. 214, pp. 237–249, 1887.

[58] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. 10th Eur. Conf. Mach. Learn.*, Apr. 1998, pp. 137–142.

[59] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Represent.*, New Orleans, LA, USA, 2018.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[61] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Vancouver, BC, Canada: Curran Associates, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[62] A. Riddell, H. Wang, and P. Juola, "A call for clarity in contemporary authorship attribution evaluation," in *Proc. Conf. Recent Adv. Natural Lang. Process. Deep Learn. Natural Lang. Process. Methods Appl.*, 2021, pp. 1174–1179.

[63] S. Argamon and P. Juola, "Overview of the international authorship identification competition at PAN-2011," in *Proc. CLEF*, vol. 1177, V. Petras, P. Forner, and P. D. Clough, Eds., 2011, pp. 1–10.

[64] A. Gungor, "Benchmarking authorship attribution techniques using over a thousand books by fifty Victorian era novelists," Ph.D. dissertation, Dept. Comput. Inf. Sci., Purdue Univ., Indianapolis, IN, USA, 2018.

[65] F. Sebastiani, "An axiomatically derived measure for the evaluation of classification algorithms," in *Proc. Int. Conf. Theory Inf. Retr.*, Sep. 2015, pp. 11–20.

[66] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun. 1947.

[67] Z. Hu, R. K.-W. Lee, C. C. Aggarwal, and A. Zhang, "Text style transfer: A review and experimental evaluation," *ACM SIGKDD Explorations Newslett.*, vol. 24, no. 1, pp. 14–45, Jun. 2022.

[68] H. Abdullah, A. Karlekar, V. Bindschaedler, and P. Traynor, "Demystifying limited adversarial transferability in automatic speech recognition systems," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Vienna, Austria, 2021.

[69] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

**SILVIA CORBARA** received the M.Sc. degree in digital humanities from the University of Pisa, in 2019. She is currently pursuing the Ph.D. degree in data science from the Scuola Normale Superiore, Pisa, Italy. She is also a Research Associate with the Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo," National Research Council (CNR). Her research interests include authorship analysis, text classification, and natural language processing.

**ALEJANDRO MOREO** received the Ph.D. degree in computer sciences and information technologies from the University of Granada, in 2013. He is currently a tenured Researcher with the Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo," which is part of the National Research Council (CNR). His research interests include learning to quantify, text classification, and authorship analysis.

Open Access funding provided by 'Consiglio Nazionale delle Ricerche-CARI-CARE-ITALY' within the CRUI CARE Agreement