Author Identification of Micro-Messages via Multi-Channel Convolutional Neural Networks

Sarp Aykent
Department of Computer Science
& Software Engineering
Auburn University
Auburn, AL, USA
sarp@auburn.edu

Gerry Dozier
Department of Computer Science
& Software Engineering
Auburn University
Auburn, AL, USA
doziegv@auburn.edu

***Abstract*—With the emergence use of social media, millions of micro-messages are exchanged daily. Although micro-messages are a powerful and efficient way to communicate among individuals, their anonymity and short-length characteristics give rise to a real challenge for Author Identification studies. In this paper, we tackled the Author Identification of micro-messages problem via Convolutional Neural Networks (CNNs). Specifically, we introduce a novel Multi-Channel CNN architecture that processes different features of text via word and character embedding layers, and utilizes both pre-trained word embedding and character bigram embeddings. We examine the usefulness of different feature types and show that the combination of embedding layers can capture different stylometric features. We conduct extensive experiments with a varying number of authors and writing samples per author. Our results show that our proposed method outperforms the state-of-the-art system on a Twitter dataset that contains 1,000 authors.**

***Index Terms*—Convolutional Neural Networks, Authorship Attribution, Author Identification**

I. Introduction

Author Identification [1], also known as Authorship Attribution, is the task of identifying the author of unknown text based on stylistic information captured within a dataset of writing samples [2]. Author Identification is used in a wide variety of fields [3]–[7]. As the majority of Author Identification research is focused on finding authors of long texts, the development of social media platforms and the emergence of social media as a primary mode of communication [7] creates a special interest in Author Identification of micro-messages. The increase in micro-message traffic which includes web forum posts, tweets, and product reviews creates a massive amount of data, which has attracted increasing attention in many fields such as social media forensics [7]. Social media platforms such as Twitter [8] limit the number of characters for each post (micro-message). The research question becomes: How can we identify an author of a single post message that has 140 characters or less? To address the above question, we investigate a Author Identification methods on micro-messages, as well as present a new method that outperforms the others.

Currently, the performance of existing Author Identification systems of micro-messages [8] is no better than a chance of flipping the coin even with a small number of the author set [3]. The recent development of the CNNs has shown promising results on image classification tasks [9], [10]. One of the difficulties of using these types of CNNs for Author Identification is basically a representational difference [11]. The tokens of the text (character, word, etc.) are discrete and are not convertible to values that machines can understand [11]. Studies [12], [13] have shown that using a number of word embedding techniques can convert words into individual vectors. Some of these techniques include Word2Vec [12], FastText [13], etc. In literature [8], [14]–[17], tokens other than words could also be used as features in machine learning algorithms for Author Identification. These tokens include character n-grams [4], [7], [18], parts of speech [7], and stylometry [19].

Our goal is to design a CNN architecture for identifying the authors of micro-messages to overcome the problems mentioned before. In the literature [7], it is shown that different feature sets can capture a variety of stylometric features. In light of this information, we combine word embeddings with character embeddings under one CNN architecture to form a Multi-Channel CNN architecture. These embedding layers can be considered as individual information channels for a CNN. Our preliminary results, using shared convolutional filters for both embedding layers performed better than the state-of-the-art methods [3].

We compare our proposed architecture with the best performing state-of-art methods within the literature [3] as well as several popular authorship attribution methods [8]. We show that using Multi-Channel CNNs with pre-trained word embeddings performs the best.

The remainder of the paper is as follows. Section II provides the necessary background and discusses related work to this study, Section III introduces our Multi-Channel CNN architecture for Author Identification of micro-messages. Our experiments are presented in Section IV. In Section V we present our results and Section VI we present our conclusions and future work.

II. Related Work

In this section, we present some related work to our research.

A. Author Identification of Micro-Messages

In the literature [2], a variety of methods have been developed and used for micro-message Author Identification problems. Schwartz et al. [8] introduced a method known as K-Signatures to identify authors of micro-messages. The K-Signatures method captures the style of an author by collecting the features only found in the writing samples of the author. Also, another condition for collecting the feature is that a given feature needs to be seen in k% of the author's writing samples. The K-Signature method was used by SVMs to classify an author of a given text.

Rocha et al. [7] provides an overview of the Authorship Identification methods used in social media. In addition, they provided performance analysis of the feature sets of micro-messages using two classifiers, namely, a Power Mean SVM [20] and Random Forests. They suggest that character 4-grams perform best.

Shrestha et al. [3] used a different approach for identifying the authors of micro-messages. They used a sequence of character n-grams as input. To date, this is the best performing algorithm for Author Identification of micro-messages.

B. Neural Networks for Author Identification

CNNs for Author Identification have shown promising results [3], [21]. Kim [21] proposed a CNN for sentence classification. The CNN utilized convolutional filter sizes of 3, 4, and 5, a dropout rate, and max-over-time pooling (Collobert et al. [11]). Kim's approach [21] use of a different combination of models such as: a) randomly initialized, b) static model, c) dynamic model, d) Multi-Channel that combines the static and dynamic models. The models, excluding the random model, used pre-trained word2vec word embeddings [12].

Recurrent Neural Networks (RNNs) [22], have also been used for Author Identification. Bagnall [22] used a RNN that predicts the next character in the sequence based on the previously seen characters. Different sets of probabilities of the next character for each author were generated, then authors were identified based on these probabilities. This was the best performing method for the PAN 2015 multi-language author identification competition.

C. Multi-Channel Convolutional Neural Networks

Multi-Channel CNNs [23] have been extensively studied such as image classification, object detection, and speech recognization [24]. The color channels (such as RGB) in computer vision [23] or the wavelengths channels in speech recognization [24] have proven successful as Multi-Channel input for classification problems in their specific fields. Although natural language inputs are normally in the form of single-channel tokens or characters, the different sets of extracted features are shown to capture different stylometric features [4] [18] [15] [7]. To our knowledge, no previous work has been done on the micro-message author identification task using feature learning and model training from both word and character n-grams embeddings channels.

III. Multi-Channel Convolutional Neural Networks

A. Model construction

In light of the above discussion, we devised a Multi-Channel CNN architecture utilizing both the word embeddings and character n-gram embeddings. The Multi-Channel convolutional network architecture is shown in Figure 1 and the model details are as follows.

1) Embedding Layer: Figure 1 shows two feature sets used as individual input channels with their own embedding layers. These feature embeddings are padded where necessary to have mutual size. The dropout was applied to the embedding layer to avoid over-fitting. The feature embeddings (inputs) were then given to the convolutional layers with different window sizes in parallel.

2) Convolutional Layer: The convolutional layers apply filters to the input, feature embeddings in our case, and shifts the filter until the end of the sequence. The window size of the convolution operation corresponds to the size of the filter that was applied to the input each time. Following the work of [11], we used convolutional filters with window sizes of 3, 4, and 5. We used the Scaled Exponential Linear Unit (SELU) activation function [25] after each convolutional layer for self normalizing properties.

The output of the convolutional filters are known as feature maps. Since we used character embeddings and word embeddings as an individual input channel, each channel has a feature map of size 1,500.

3) Pooling Layer: Convolutional layers are followed by max-over-time pooling [11], which takes the feature with maximum value in a given filter.

In this way, we make the network size the most helpful features produced by convolutional layers. Thus the number of selected values with max-over-time pooling is equal to the number of filters in the architecture per feature embedding.

4) Merge Layer: The resulting features are merged into a single feature map. In order to merge the feature maps of the character and word embeddings, we experimented with both concatenation ($\cup$) and add (+) operations in merge layer. The differences between the two operations are that the concatenation operation concatenate the features into a single feature vector, while the add operation use pairwise add operation between feature vectors.

5) Fully Connected Layer with Softmax Output: After the merge layer, the final step is to feed the feature maps into the fully connected layer, followed by the softmax function for the classification.

B. Hyper-parameter tuning

Hyper-parameters were selected based on grid search [26]. The embedding dimensions size of character and word used was 400. We used a dropout layer with a drop rate of 25%. For the convolutional layer, there were 500 filters per window size, $M = 500$, which makes a total of 1,500 filters. Hence, only one feature was selected per filter. We use a batch size

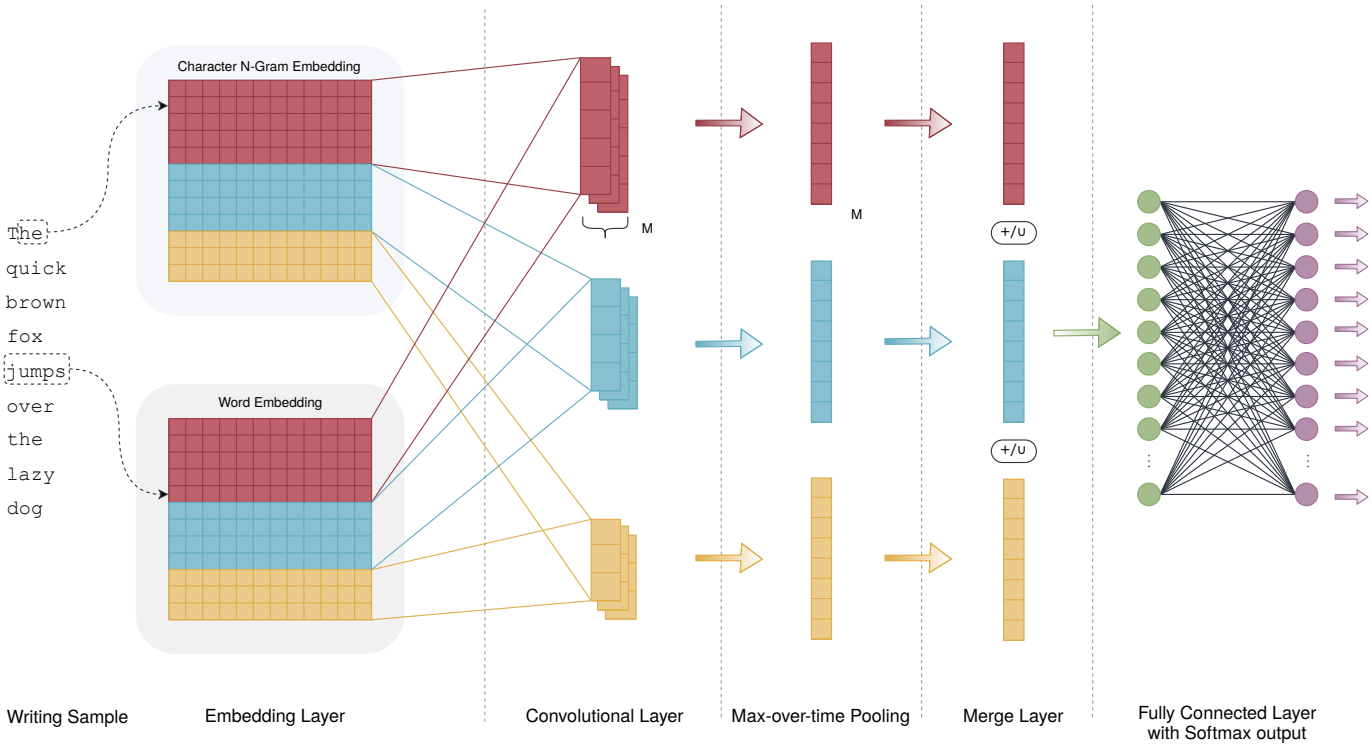


Fig. 1. Multi-Channel CNN architecture diagram. The writing sample is "The quick brown fox jumps over the lazy dog". The writing samples are tokenized and fed to embedding layers. Word embedding and character bigram embedding layers are illustrated in the figure. The red, blue and yellow color indicate the different window sizes for each channel. The embeddings then forwarded to convolutional layers with the different window sizes. There are M numbers of filters for each window size. Max-over-time pooling is used in pooling layer, followed by merge layer, which combines the features by either Concatenation($\cup$) or Add($+$) operation, then feed into fully connected layer with softmax output.

of 64 for the experiment. The epoch limit was 100 with an early stopping condition that stops after 20 epochs without improvement. The learning rate updated during the training by Adam optimizer. The initial learning rate was $1e-4$.

IV. Experiments

In this section, we present our experiments. Initially, we study the impacts of different features for embedding layers. Then we study the performance of our proposed architecture with increased difficulties, such as increasing the number of authors or the number of writing samples. The experiments were conducted in two different settings. We analyzed the relation of two parameters with the performance of the proposed architecture. The parameters were the number of authors and the number of writing samples, one parameter was kept static while the other was tested on a range of values. Lastly, we conducted an experiment to compare different operations of merging layers. The preprocessing method for the dataset was used, which is similar to [8], in all four experiments. The steps can be summarized as follows. We replaced the numbers, username references, date, time, and website URLs with predefined meta tags.

The statistics of the datasets used in experiments are shown in Table I. The first column shows the type and size of the dataset. The number of authors and the number of writing samples per author was denoted as a and w, respectively. Then, the mean (μ) and standard deviation (σ) of the number of characters, words, and sentences for writing samples are shown. The last column shows that the dictionary size which is the number of unique words in the dataset. The average number of characters used was close to half the size of the character limit of the writing samples. The mean and standard deviation of the characters, words, and sentences are consistent for all configurations. Conversely, the number of unique words increases with the number of total writing samples.

TABLE I
Dataset Statistics

a	Characters		Words		Sentences		Dict
	μ	σ	μ	σ	μ	σ	
100_a	71.57	33.81	14.10	6.50	1.69	0.90	47493
200_a	73.15	34.04	14.29	6.49	1.71	0.96	81735
500_a	73.79	34.11	14.42	6.55	1.69	0.95	161742
1000_a	73.28	33.84	14.30	6.49	1.71	0.96	269774
50_w	73.26	34.22	14.17	6.45	1.66	0.94	10858
100_w	73.04	34.20	14.15	6.45	1.65	0.92	18094
200_w	73.04	34.16	14.15	6.44	1.65	0.93	29942
500_w	72.87	34.11	14.12	6.43	1.65	0.92	56975

Our four experimental settings are as follows.

A. Experiment I: Varying Character N-Gram Embeddings

To assess the effectiveness of different features for the embedding layer, we compare the n-gram embedding performance and the number of epochs on the same dataset set. In this experiment, the individual performance of n-gram embeddings with different n values was evaluated. We tested 1, 2, 3, 4, and 5 for n values based on their usage in the literature [7], [8]. For this purpose, we created a development dataset, also known as validation set, with 10 authors and 200 writing samples per author. The authors in the development set are not used in any other experiment. We collect the performance of the CNNs using 100 and 1,000 epochs. Since the number of unique n-grams increases with higher values of n, we increased the number of epochs 1,000 to avoid under-fitting.

B. Experiment II: Varying Number of Authors

We explore how our proposed architecture performs with an increasing number of authors. We performed a set of performance evaluations with a different number of authors while keeping the number of writing samples static. The number of writing samples per author (w) was 200. The number of authors (a) was used in experiments was 100, 200, 500, and 1,000. In this experiment, we used the same sampling used in [8]. As expected, the reproduced results in this paper were similar to the reported results in [3] and [8]. We used 10-fold cross-validation on the experiments for each author group. Hence, we evaluated 40 train-test splits in total.

C. Experiment III: Varying Number of Writing Samples

In this experiment, we investigate the impact of a different number of writing samples for author identification. We performed a set of performance evaluations with a varying number of writing samples per author where the number of authors was 50. The number of writing samples per author used in experiments was 50, 100, 200, and 500. We sampled 10 different disjoint sets of groups for each writing sample size[1]. We used 10-fold cross-validation on the experiments for each author group. Hence, we evaluated 400 experiments in total.

D. Experiment IV: Impacts of feature map merging methods

In our final experiment, we study the impacts of different methods for merging layers. There are two popular ways to merge the feature maps, namely add and concatenate operation. We explored the impacts of the two methods under the same experimental setup for both varying number of authors and the varying number of writing samples where (∪) stand for concatenation operation and (+) stand for add operations. The rest of the experimental setup was the same as described above.

[1]We contacted the authors of the previous work but unfortunately, they told us that they do not have the sampling anymore.

V. Results

We compare our proposed methods against the following baselines. The information about the algorithms that were used in the experiments can be found in the following list:

- K-Signatures [8]: Uses the features that include character and word N-grams. The features used by a single author at least K% of the documents are used. Also, a method called Flexible Patterns was utilized. Patterns in this method can match partially and get a score based on the closeness. Combination of K-Signatures and Flexible Patterns techniques shown better results in the paper. Hence, reported results are a combination of the techniques.
- LSTM$_{Char2}$: Long Term Short Term Memory (LSTM) networks are widely used and known for the sequence to sequence tasks. There were also applications for Author Identification. We used character bigrams as an input to the network based on our preliminary findings on our development set.
- CNN$_{CharN}$ [3]: All character N-grams are randomly initialized and then updated in the training. The other parameters of the networks like filter weights are also updated with Backpropagation. In this paper, we implement both CNN$_{Char1}$ and CNN$_{Char2}$ methods to compare with our proposed method.
- CNN$_{W2VStatic}$ [21]: All words are initialized with a pre-trained word vector from Word2Vec. Only weights are updated during the training, word vectors are static in the training phase.
- CNN$_{W2V}$ [21]: Pre-trained vectors in CNN$_{W2VStatic}$ are updated during the training. The pre-trained embeddings are trained on the Twitter dataset.
- CNN$_{FastText}$: All words are initialized with a pre-trained word vector from FastText. The pre-trained embeddings are trained on the Twitter dataset.

We implement an ablation study of our proposed architecture where different word embedding and character n-gram embedding methods are treated as individual input channels. The embeddings in each channel are updated during the training phase. The individual channels are merged with Concatenate (∪) or Add (+) layer. The implementation information and detailed explanation of these settings are listed as follows:

- CNN$_{W2V\ (+/\cup)\ Char1}$: The word embeddings and character unigram embeddings are treated as an individual input channel. All words are initialized with a pre-trained word vector from Word2Vec.
- CNN$_{W2V\ (+/\cup)\ Char2}$: The word embeddings and character bigram embeddings are treated as an individual input channel. All words are initialized with a pre-trained word vector from Word2Vec.
- CNN$_{FastText\ (+/\cup)\ Char1}$: The word embeddings and character unigram embeddings are treated as an individual input channel. All words are initialized with a pre-trained word vector from FastText.
- CNN$_{FastText\ (+/\cup)\ Char2}$: The word embeddings and character bigram embeddings are treated as an individual

input channel. All words are initialized with a pre-trained word vector from FastText.

A. Results of Experiment I: Varying Character N-Gram Embeddings

Figure 2 shows the performances of character n-grams on the development set. Figure 2a and 2b were trained for 100 and 1,000 epochs, respectively. The figures illustrate the accuracy of 10-fold cross-over, the mean values are shown with solid black lines, and the individual precision values of each author are shown as dots with circles.

We performed ANOVA and Student-t test with p=0.05 on performances of character n-gram embeddings to compare them. Figure 2a the mean accuracies of 1-5 grams were 66.15%, 66.00%, 57.25%, 52.50%, and 49.65%, respectively. Based on the statistical test we performed, we observed that there was no significant difference between 1-gram(unigram) and 2-gram(bigram) embeddings. The performance of the n-grams when they are trained for 100 epochs as follows, $1-gram = 2-gram > 3-gram > 4-gram = 5-gram$. The mean accuracies of 1-5 grams in Figure 2b were 78.45%, 75.70%, 72.40%, 65.35%, and 55.45%, respectively. Based on the statistical test we performed the performance of the n-grams when they are trained for 1,000 epochs as follows, $1-gram > 2-gram > 3-gram > 4-gram > 5-gram$. The performance gap becomes clear between character n-grams after increasing the number of epochs. Due to lack of space, Figure 2 reports only on character n-grams, but the same trend applies for word n-gram embeddings.

B. Results of Experiment II: Varying Number of Authors

Table II shows the results of approaches discussed in Section IV with varying number of authors. In Table II, the first column list the algorithm used in the experiment, where the baseline methods are listed above the double lines and our proposed methods are listed below the double lines. The rest of the columns list the accuracies of the algorithms with 100, 200, 500, and 1000 authors, respectively. The number in red marks the highest accuracy in each column.

As expected the accuracies decrease as the number of authors increases for all the methods listed. Among the seven baseline methods, $\text{CNN}_{FastText}$ performed surprisingly better than others without needing character n-grams. Our tests confirm that using character n-grams perform better than the compared algorithms. CNN_{W2V} which does not use character n-gram performed worse than the CNN_{Char2} on all of the cases. We were unable to reproduce the behavior of CNN_{Char1} and CNN_{Char2} reported on [3]. The reported results show that Character Unigrams perform better on 100 authors compared to Character Bigrams with a small margin. In our tests, Character Bigrams performed better on all of the tests.

Comparing with CNN_{W2V}, the Multi-Channel architecture methods perform better with the adding of character N-grams channel. The best performer in terms of the word embeddings pre-trained with Word2Vec is $\text{CNN}_{W2V+Char2}$, which performed better than the state-of-the-art result. The accuracies of the $\text{CNN}_{W2V+Char2}$ algorithms with 100, 200, 500, and 1000 authors are 52.67%, 50.53%, 43.79%, 38.29%, respectively.

Similarly with Word2Vec, $\text{CNN}_{FastText+Char2}$ was the best performer for all authors sets in Experiment II with FastText embeddings and the overall. The accuracies of the $\text{CNN}_{FastText+Char2}$ algorithms with 100, 200, 500, and 1000 authors are 55.20%, 53.14%, 46.90%, 41.28%, respectively. On average, $\text{CNN}_{FastText+Char2}$ performed 10% better than the state-of-the-art results with 100, 200, 500, and 1000 authors.

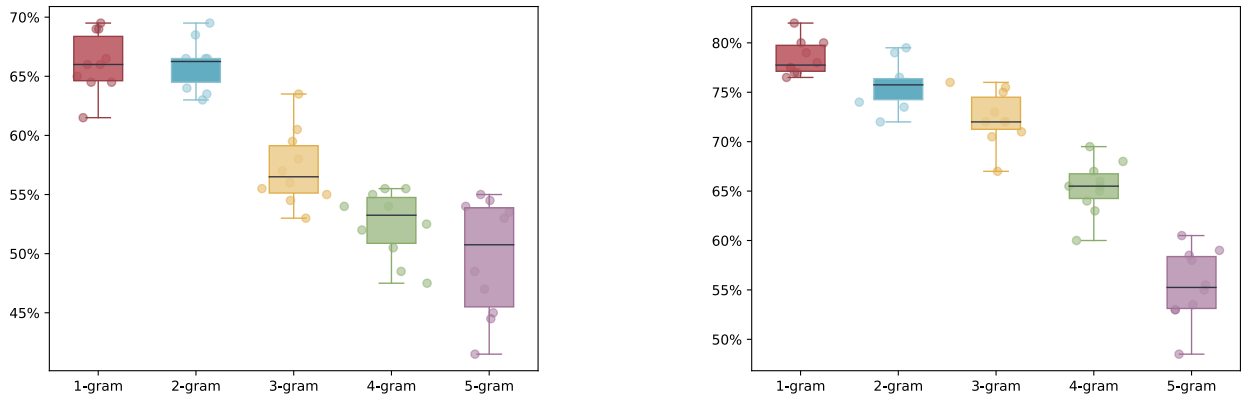
TABLE II
PERFORMANCES OF THE ALGORITHMS WITH VARYING NUMBER OF AUTHORS

Algorithms	Authors			
	100	200	500	1000
CNN_{Char1} [3]	49.24%	47.68%	41.37%	35.60%
CNN_{Char2} [3]	49.96%	48.84%	42.92%	37.55%
K-Signatures [8]	42.50%	41.10%	35.50%	30.30%
LSTM_{Char2}	33.80%	33.50%	29.80%	24.80%
$\text{CNN}_{W2VStatic}$	24.10%	20.80%	16.10%	12.70%
CNN_{W2V}	47.21%	45.52%	39.85%	34.73%
$\text{CNN}_{FastText}$	51.83%	50.25%	44.18%	38.74%
$\text{CNN}_{W2V \cup Char1}$	50.42%	48.12%	42.44%	37.48%
$\text{CNN}_{W2V+Char1}$	52.44%	49.74%	43.53%	37.71%
$\text{CNN}_{W2V \cup Char2}$	48.95%	47.06%	41.76%	36.57%
$\text{CNN}_{W2V+Char2}$	52.67%	50.53%	43.79%	38.29%
$\text{CNN}_{FastText \cup Char1}$	52.62%	51.36%	46.17%	40.61%
$\text{CNN}_{FastText+Char1}$	52.55%	50.89%	45.42%	40.25%
$\text{CNN}_{FastText \cup Char2}$	54.23%	52.23%	46.62%	40.84%
$\text{CNN}_{FastText+Char2}$	**55.20%**	**53.14%**	**46.90%**	**41.28%**

C. Results of Experiment III: Varying Number of Writing Samples

Table III shows the results of approaches discussed in Section IV with varying number of writing samples per author. In Table III, the first column list the algorithm used in the experiment, where the baseline methods are listed above the double lines and our proposed methods are listed below the double lines. The rest of the columns list the accuracies of the algorithms with 50, 100, 200, and 500 writing samples per author, respectively. The number in red marks the highest accuracy in each column.

Obviously, the accuracies increase as the number of writing samples increases for all the methods listed. Among the baseline methods, $\text{CNN}_{FastText}$ performed surprisingly better than everything without needing character n-grams. The reported results in [3] show that Character Bigrams perform better only on 500 writing samples compared to Character Unigrams. However, in our tests, Character Bigrams performed better on 50 and 100 writing samples which suggest that character unigrams perform better with larger writing samples.



(a) Trained for 100 epochs

(b) Trained for 1,000 epochs

Fig. 2. The box plot that illustrates the performances of character n-grams on the development set. Precisions are shown in the two panels. In each panel, five boxplots display the results of 1-5 gram methods, respectively. In each box, the square box indicates the interquartile range from 25% to 75%; the black line inside the box represent the median value. Individual precision values of authors from each method are shown as small dots with circles in the same color code as boxplot. Panel(a) trained for 100 epochs and panel(b) trained for 1,000 epochs.

Our tests confirm that using character n-grams perform better than the compared algorithms. CNN_{W2V} which does not use character n-gram performed worse than the CNN_{Char2} on all of the cases. After adding character N-grams channel it performed better than the CNN_{W2V}. The best performer in terms of the word embeddings pre-trained with Word2Vec is $\text{CNN}_{W2V+Char2}$, which performed better than the state-of-the-art result. The accuracies of the $\text{CNN}_{W2V+Char2}$ algorithms with 50, 100, 200, and 500 writing samples are 54.91%, 61.52%, 67.11%, 72.88%, respectively. From Table III we can see that $\text{CNN}_{W2V+Char2}$ performs best among all methods with 50 writing samples.

Furthermore, we observed the same behavior with character n-grams which improved the accuracy in every combination. Similarly with Word2Vec, $\text{CNN}_{FastText+Char2}$ was the best performer in all experiments with FastText embeddings. On average, $\text{CNN}_{FastText+Char2}$ performed 5% better than the state-of-the-art results with 50, 100, 200, and 500 writing samples per author. The accuracies of the $\text{CNN}_{FastText+Char2}$ algorithms with 50, 100, 200, and 500 writing samples are 54.36%, 62.17%, 68.34%, 74.50%, respectively. The improvement in accuracy is less than the improvements observed in the experiment with varying number of authors.

D. Result of Experiment IV: Impacts of feature map merging methods

Table II shows the comparison of different merging methods with varying the number of authors. The add operation performs better in all experiments except one (Table II, $FastText \cup Char1$) with small margins. The accuracy of $FastText \cup Char1$) algorithm with 100, 200, 500, and 1000 authors are 52.62%, 51.36%, 46.17%, and 40.61%, whereas the accuracy of $FastText + Char1$) algorithm with 100, 200, 500, and 1000 authors are 52.55%, 50.89%, 45.42%, and 40.25%.

Similarly, Table III shows the comparison of different merging methods with varying the number of writing samples. The add operation performs better in all experiments with varying the number of writing samples except one (Table III, $FastText \cup Char1$) with small margins. The accuracy of $FastText \cup Char1$) algorithm with 500 writing samples is 72.96%, whereas the accuracy of $FastText + Char1$) algorithm with 500 writing samples is 72.89%.

From both Table II and Table III, we can see that the best performer in each experiments all uses add operation in merge layer. In a general case for authorship attribution on micro-messages, the recommended merge operation would be add operation instead of concatenation for our proposed Multi-Channel CNN architecture based on the previous experimental results.

VI. Conclusion and Future Work

In this paper, we presented a Multi-Channel CNN approach that can be applied to identify the authors of micro-messages. We compared the performance of state-of-the-art CNNs that uses character n-gram embeddings with the proposed CNNs that use Multi-Channel architecture. The experiments that we carry out show that using Multi-Channel CNNs with pre-trained word embeddings performs better compared to the CNNs that use character n-gram embeddings on the Twitter dataset. Also, we showed that using the Add operation over the Concatenation operation in the merge layer increases the performance of the system on all of the cases but one ($CNN_{FastText \cup Char1}$). It would be interesting to see if the performance keeps increasing when more pre-trained features are used. Since as the time of the writing only word embeddings are maturely pre-trained, we left the exploration of this to future work.

TABLE III
PERFORMANCES OF THE ALGORITHMS WITH VARYING NUMBER OF WRITING SAMPLES

Algorithms	Writing Samples			
	50	100	200	500
CNN_{Char1} [3]	51.40%	58.20%	64.07%	70.30%
CNN_{Char2} [3]	51.56%	58.25%	63.59%	69.80%
$\text{CNN}_{W2VStatic}$	36.60%	41.70%	46.00%	50.90%
CNN_{W2V}	49.14%	56.68%	62.96%	69.70%
$\text{CNN}_{FastText}$	51.46%	59.14%	65.61%	72.46%
$\text{CNN}_{W2V \cup Char1}$	52.86%	60.10%	65.85%	71.89%
$\text{CNN}_{W2V + Char1}$	53.71%	60.55%	66.12%	72.30%
$\text{CNN}_{W2V \cup Char2}$	52.68%	60.12%	65.87%	71.86%
$\text{CNN}_{W2V + Char2}$	**54.91%**	61.52%	67.11%	72.88%
$\text{CNN}_{FastText \cup Char1}$	51.94%	59.83%	66.14%	72.96%
$\text{CNN}_{FastText + Char1}$	53.28%	60.56%	66.64%	72.89%
$\text{CNN}_{FastText \cup Char2}$	52.04%	60.58%	67.28%	73.87%
$\text{CNN}_{FastText + Char2}$	54.36%	**62.17%**	**68.34%**	**74.50%**

REFERENCES

[1] E. Stamatatos, "A survey of modern authorship attribution methods," *Journal of the American Society for information Science and Technology*, vol. 60, no. 3, pp. 538–556, 2009.

[2] T. Neal, K. Sundararajan, A. Fatima, Y. Yan, Y. Xiang, and D. Woodard, "Surveying stylometry techniques and applications," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, p. 86, 2018.

[3] P. Shrestha, S. Sierra, F. González, M. Montes, P. Rosso, and T. Solorio, "Convolutional neural networks for authorship attribution of short texts," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 669–674. [Online]. Available: https://www.aclweb.org/anthology/E17-2106

[4] V. Kešelj, F. Peng, N. Cercone, and C. Thomas, "N-gram-based author profiles for authorship attribution," in *Proceedings of the conference pacific association for computational linguistics, PACLING*, vol. 3. sn, 2003, pp. 255–264.

[5] M. Koppel, J. Schler, and S. Argamon, "Authorship attribution in the wild," *Lang. Resour. Eval.*, vol. 45, no. 1, pp. 83–94, Mar. 2011. [Online]. Available: http://dx.doi.org/10.1007/s10579-009-9111-2

[6] W. J. Teahan and D. J. Harper, "Using compression-based language models for text categorization," in *Language modeling for information retrieval*. Springer, 2003, pp. 141–165.

[7] A. Rocha, W. J. Scheirer, C. W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A. R. Carvalho, and E. Stamatatos, "Authorship attribution for social media forensics," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 1, pp. 5–33, 2016.

[8] R. Schwartz, O. Tsur, A. Rappoport, and M. Koppel, "Authorship attribution of micro-messages," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1880–1891.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[13] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 427–431. [Online]. Available: https://www.aclweb.org/anthology/E17-2068

[14] G. Baron, "Comparison of cross-validation and test sets approaches to evaluation of classifiers in authorship attribution domain," in *International Symposium on Computer and Information Sciences*. Springer, 2016, pp. 81–89.

[15] J. Diederich, J. Kindermann, E. Leopold, and G. Paass, "Authorship attribution with support vector machines," *Applied intelligence*, vol. 19, no. 1-2, pp. 109–123, 2003.

[16] M. Jockers and D. M. Witten, "A comparative study of machine learning methods for authorship attribution," *LLC*, vol. 25, pp. 215–223, 05 2010.

[17] J. Gaston, M. Narayanan, G. Dozier, D. L. Cothran, C. Arms-Chavez, M. Rossi, M. C. King, and J. Xu, "Authorship attribution vs. adversarial authorship from a liwc and sentiment analysis perspective," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, Nov 2018, pp. 920–927.

[18] U. Sapkota, S. Bethard, M. Montes, and T. Solorio, "Not all character n-grams are created equal: A study in authorship attribution," in *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies*, 2015, pp. 93–102.

[19] A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin, and D. Song, "On the feasibility of internet-scale author identification," in *2012 IEEE Symposium on Security and Privacy*, May 2012, pp. 300–314.

[20] J. Wu, "Power mean svm for large scale visual classification," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2344–2351.

[21] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. [Online]. Available: https://www.aclweb.org/anthology/D14-1181

[22] D. Bagnall, "Author identification using multi-headed recurrent neural networks," *arXiv preprint arXiv:1506.04891*, 2015.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[24] Y. Hoshen, R. J. Weiss, and K. W. Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4624–4628.

[25] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 971–980. [Online]. Available: http://papers.nips.cc/paper/6698-self-normalizing-neural-networks.pdf

[26] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research*, vol. 13, no. Feb, pp. 281–305, 2012.