Byte Coding Gains Privacy without Sacrificing Accuracy and Complexity

Anonymous ACL submission

Abstract

While NLP models significantly impact our lives, there are rising concerns about privacy invasion. Although federated learning enhances privacy, attackers may recover private training data by exploiting model parameters and gradients. Therefore, protecting against such an embedding attack remains an open challenge. We propose Subword Embedding from Bytes (SEB) and encode subwords to byte sequences using neural networks that are harder to retrieve in attacks. Importantly, our method requires a smaller memory with only 256 bytes of vocabulary while keeping efficiency with the same input length as usual. Thus, our solution outperforms conventional approaches by preserving privacy without sacrificing efficiency or accuracy. Our experiments show SEB can effectively protect against embedding-based attacks from recovering original sentences in federated learning. Meanwhile, we verify that SEB obtains comparable and even more accurate prediction results over standard subword embedding methods in machine translation, sentiment analysis, and language modeling with even lower time and space complexity.

1 Introduction

Advances in Natural Language Processing (NLP), such as Large Language Models (LLMs), have made noticeable advancements in performance over the last decades, partially attributed to the large datasets available. Since most data are from users, their privacy concerns play an increasingly critical role, which is essential to building user trust, encouraging the responsible use of language data, protecting personal information, ensuring ethical use, and avoiding potential harm to individuals.

Federated learning (FL) enables training shared models across multiple clients without transferring the data to a central server to preserve user privacy. Although only the model updates are sent to the central server, adversaries can still use model updates to reconstruct the original data and leak sensitive information to compromise the user's privacy. Figure 1(a) demonstrates an FL framework, and Figure 1(b) shows how embedding-based attacks work as in Gupta et al. (2022). In the illustrated example, the attacker extracts all candidate words in a batch of data from the embedding gradients and can easily reconstruct the text with beam search and reordering since one can perform straightforward lookups when a vector is updated due to the one-to-one mapping between word/subword tokens and embedding vectors.

Our intuitive idea is to apply the byte embedding method because the same bytes are repeatedly used for multiple subwords. We aim to design a one-to-many mapping between words/subwords and embedding vectors to increase the difficulty of the simple lookup so that retrieving input subwords with the updated byte embeddings is harder, which makes the byte embedding in NLP models a potential defense. For example, in subword embedding, if the word "good" is updated, the attacker will only retrieve this word based on embedding updates. However, if we tokenize "good" into four bytes, such as "50, 10, 128, 32", all subwords containing at least one of these bytes will be retrieved, resulting in a larger search space and more possibilities to recover the original sentence. As shown in Figure 1(c), although the attacker extracts a set of bytes, the number of candidate subwords is much greater than that of using subword embeddings.

There are two major challenges to directly apply existing byte encodings (Xue et al., 2022; Shaham and Levy, 2021; Zhang and Xu, 2022) to enhance privacy: First, smaller textual granularity cannot show the semantic meaning of each word, leading to a less interpretable and analyzable model. Second, byte-based models are more computationally expensive, as input sequences become much longer after byte tokenization.

To address these challenges in byte-based models, we propose to encode subwords with bytes and aggregate the byte embeddings to obtain a single subword embedding. The procedure consists of three steps: (1) Construct a mapping between sub-

words and bytes. (2) Convert the input text into a byte sequence. (3) Retrieve the corresponding byte embeddings and aggregate them back into subword embeddings using a feed-forward network while maintaining the subword boundaries. By adopting this approach, we can leverage the privacy protection provided by bytes while preserving the semantic meaning of the input sequence with the same input length but a constant number of vocabulary size of 256.

Our main contributions are:

- We introduce a novel text representation method SEB, which achieves a vocabulary size of 256 of the learned model without increasing the input sequence length.
- We verify that our SEB can protect NLP models against data leaking attacks based on embedding gradients. To the best of our knowledge, our work is the first one to study privacy preservation with byte representations in FL.
- We demonstrate that SEB improves privacy and, at the same time, achieves comparable or better accuracy with enhanced time and space efficiency without the privacy-performance/efficiency trade-off in conventional approaches.

2 Related Work

Attacks and defenses in language model Some recent works consider the reconstruction as an optimization task (Zhu et al., 2019; Deng et al., 2021; Balunovic et al., 2022). The attacker updates its dummy inputs and labels to minimize the distance between the gradients of the victim uploaded and the gradients the attacker calculated based on its dummy inputs and labels. Gupta et al. (2022) shows that the attackers can reconstruct a set of words with the embedding gradients, then apply beam search and reorder with a pretrained language model for input recovery. One defense described in Zhu et al. (2019); Deng et al. (2021); Balunovic et al. (2022) is to encrypt the gradients or make them not directly inferable. However, encryption requires special setups and could be costly to implement. Moreover, it does not provide effective protection against server-side privacy leakage (Aono et al., 2017; Huang et al., 2021; Fang and Qian, 2021). Differential privacy is another defense strategy, but it can hurt model accuracy (Zhu et al., 2019; Wei et al., 2020; Yin et al., 2021; Li et al., 2021). While Zhang and Wang (2021) proposed a secure federated learning framework that can prevent privacy leakage based on gradient reconstruction, it does not effectively address the retrieval of a bag of words from the embedding matrix gradients, as proposed in Gupta et al. (2022).

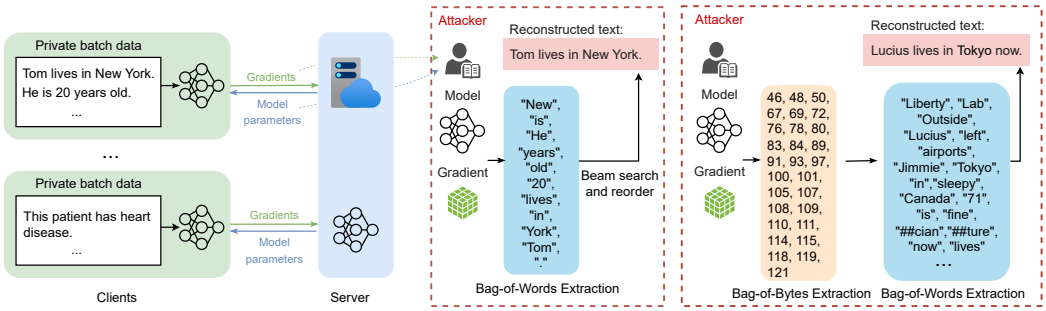
Subword-level and byte-level language models Despite the wide application of subword tokenization such as BPE (Sennrich et al., 2015), it still has some limitations. It cannot handle out-of-vocabulary subwords and requires language-specific tokenizers. Another challenge is the high space complexity of the embedding matrix when the vocabulary size is huge. Byte tokenization is a solution to address these issues (Shaham and Levy, 2020; Zhang and Xu, 2022; Xue et al., 2022). UTF-8 can encode almost all languages. Therefore, there will be no out-vocabulary words and the language-specific tokenizer is unnecessary. In addition, as the total number of bytes in UTF-8 is 256, the embedding matrix for byte vocabulary is much smaller than most subword vocabularies, reducing the number of parameters in the embedding layer and saving memory space.

Subword-level model with character- or byte-level fusion The character/byte-based models often result in longer input sequences and higher time complexity compared to the subword-based model. To make the model efficient, recent works have explored character/byte-level fusion. For example, Tay et al. (2021) propose CHARFORMER, using a soft gradient-based subword tokenization module to obtain "subword tokens". It generates and scores multiple subword blocks, aggregates them to obtain subword representation, and then performs downsampling to reduce the sequence length. Although CHARFORMER is faster than vanilla byte/character-based models, it does not maintain subword boundaries, limiting the model's interpretability. Sreedhar et al. (2022) propose Local Bytes Fusion (LOBEF) to aggregate local semantic information and maintain the word boundary. However, it does not reduce the sequence length, making training and inference time-consuming.

3 Preliminaries

3.1 Federated Learning

In federated learning (FL), multiple clients jointly train a model using their private data. Assume we have N clients, $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$, and a server s, in an FL system. The jointly trained model is f with parameters θ. The clients' private data are $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_N$ and the objective function is $\mathcal{L}$. For easier illustration, we assume all the clients participate in each communication and clients use



(a) (b) (c)

Figure 1: An attack example of recovering text in FL. (a): An FL framework. (b) and (c): Recovering text using embedding gradients of subwords and bytes.

FedSGD (McMahan et al., 2017) to update the model parameters. In each communication round t, server s first sends the model parameters θ^t to all clients. Then each client c_i compute $\nabla_{\theta^t}\mathcal{L}_{\theta^t}(\mathcal{B}_i)$, the gradients of current model f_{θ^t}, based on a randomly sampled data batch $\mathcal{B}_i \subset \mathcal{D}_i$. After local computation, the clients send the gradients $\Delta_1^t, \Delta_2^t, \ldots, \Delta_N^t$ to server and server s aggregate all the gradients and update the model:

$$\theta^{t+1} = \theta^t - \eta \sum_{i=1}^{N} \nabla_{\theta^t}\mathcal{L}_{\theta^t}(\mathcal{B}_i). \quad (1)$$

Here, Equation (1) is the gradient descent, and η is the learning rate.

3.2 Threat Model

Adversary's capabilities and objective We follow the attack settings in (Gupta et al., 2022). The optimized model is a language model $\mathcal{L}$, parameterized by θ. This scenario makes the attacker white box access to the gradients $\nabla_{\theta^t}\mathcal{L}_{\theta^t}(\mathcal{B}_i)$ sent by the victim client c_i. θ^t is the model parameter that the server sends to the clients at any communication round t. From parameters θ^t and gradients $\nabla_{\theta^t}\mathcal{L}_{\theta^t}(\mathcal{B}_i)$, the attacker can get the information of the vocabulary $\mathcal{V}$ and the embedding matrix $\mathbf{W}$ to retrieve which tokens are updated. The goal of the attacker is to recover at least one sentence from $\mathcal{B}_i$, based on $\nabla_{\theta^t}\mathcal{L}_{\theta^t}(\mathcal{B}_i)$ and θ^t.

Attack model This paper does not address the gradient leakage attack which aims to obtain private data by minimizing the difference between gradients derived from a dummy input and the actual gradients of the victim's data, because several methods have been proposed to mitigate this particular attack (Zhu et al., 2019; Deng et al., 2021; Wei et al., 2020). Instead, we focus on a specific attack model, FILM, introduced in Gupta et al. (2022), for which effective defenses have yet to be explored. In this model, the attacker attempts to reconstruct sentences from the victim's training batches through a three-step process: (1) extracting candidate tokens from the gradients, (2) applying beam search with a pre-trained Language Model, such as GPT-2, to reconstruct the input sentence, and (3) reordering the subword tokens to achieve the best reconstruction.

4 Proposed Method

We propose Subword Embedding from Bytes (SEB) shown in Figure 2, including byte sequence for input text, byte embeddings, aggregation of byte embeddings, and a feed-forward network to output the subword embedding.

We aim to develop subword encoding using a smaller byte embedding matrix to save space and protect against attacks based on the embedding gradients while preserving subword boundaries to maintain the model's time efficiency. This raises two main challenges: 1) how to convert subwords into a byte sequence? and 2) how to obtain subword embeddings using byte representations?

4.1 Subword to Byte Sequence Mapping

UTF-8 encoding results in different sequence lengths for subwords. In real practice, all byte sequences need to be padded to the same length, making the byte sequence of the subword even longer. Instead of using the existing byte encoding system, we define our subword to byte sequence mapping $\mathcal{M} : \mathcal{V}_w \rightarrow (\mathcal{V}_b)^n$. $\mathcal{V}_w$ and $\mathcal{V}_b$ are subword and byte vocabularies with size of V_w and V_b, respectively. $(\mathcal{V}_b)^n$ is a sequence of n bytes in $\mathcal{V}_b$. Here the byte vocabulary size V_b and the number of bytes n to represent a subword are hyperparameters. In this way, every subword is represented with the same length, getting rid of the longer byte sequence with padding.

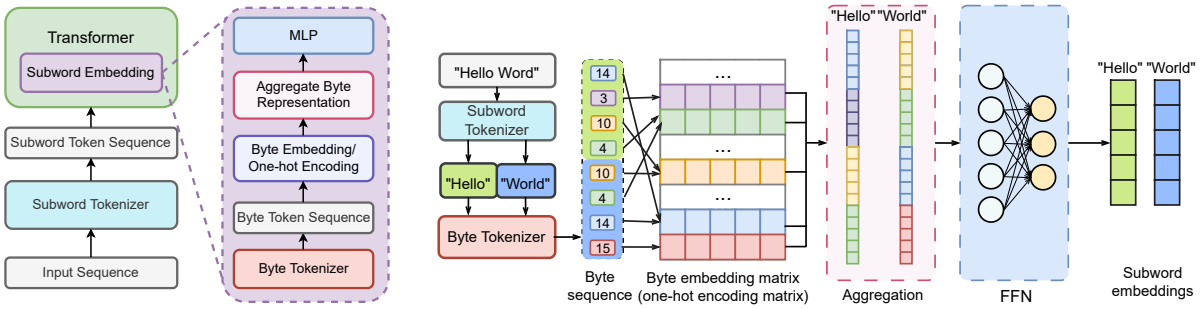


Figure 2: (a): An overview of the transformer model with SEB. (b): An example of calculating subword embeddings with byte embedding.

To construct the mapping, for every subword $w_i \in \mathcal{V}_w$, we randomly sample n bytes with replacement from $\mathcal{V}_b$ to obtain the byte sequence $(b_{i1}, b_{i2}, \ldots, b_{in})$. If the byte sequence already exists in $\mathcal{M}$, we repeat the sampling until a unique byte sequence is obtained. For example, we set $V_b = 64$ and $n = 4$. A subword "Hello" can be represented with $(14, 3, 10, 4)$, shown in Figure 2(b).

We analyze the probability p that two subwords are mapped to the same byte sequence. With the byte vocabulary size V_b and the number of bytes per subword n, the probability $p = 1/(V_b)^n$. For example, if $V_b = 16$ and $n = 4$ then $p = 1.5 \times 10^{-5}$. For $V_b = 128$ and $n = 8$ in our experiment, $p = 1.39 \times 10^{-17}$, which means there is almost no possibility to map two words into the same subword sequence. Therefore, SEB is highly expressive for representing subwords.

4.2 Subword Embedding from Bytes

Different from the byte-based models which have higher time complexity due to longer input sequences, our method tokenizes the text into a sequence of bytes while preserving the subword boundary. We first tokenize the original text into subwords using a common subword tokenization method such as BPE. Then, we token each subword into a byte sequence with the mapping we designed above and then aggregate byte representations back to subword embeddings. The two detailed algorithms are in Appendix, Algorithms 1 and 2.

Let the byte embedding matrix be $\mathbf{B} \in \mathbb{R}^{V_b \times d}$, where d is the embedding size. Given a text S, we tokenize S into a subword sequence $(w_1, w_2, \ldots, w_m)$, then further use the mapping $\mathcal{M}$ defined above to tokenize this sequence into a byte sequence $(b_{11}, \ldots, b_{1n}, \ldots, b_{m1}, \ldots, b_{mn})$ with mn bytes. We retrieve the byte embeddings $\mathbf{E} \in \mathbb{R}^{mn \times d}$ for these bytes from $\mathbf{B}$.

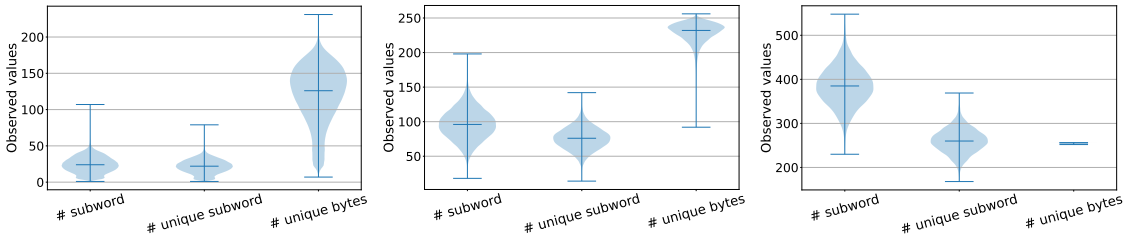
To get a subword embedding, adding the byte representations for every n bytes in $\mathbf{E}$ is a simple way. However, this approach does not consider the position of each byte within the subword. Considering that incorporating positional information can improve model performance for subword tokens, we induce positional information for byte sequences of subwords by concatenation. This enables the model to capture the position of each byte within the subword and obtain a more accurate and informative representation of the subword. Given the retrieved byte embeddings $\mathbf{E} \in \mathbb{R}^{mn \times d}$, we reshape $\mathbf{E}$ to $\tilde{\mathbf{E}} \in \mathbb{R}^{m \times nd}$ in a row-major order, which is equivalent to concatenation. Then, an FFN is applied to project $\tilde{\mathbf{E}}$ into the dimension d' of the original subword embedding for language models: $\mathbf{E}' = \texttt{FFN}(\tilde{\mathbf{E}}) \in \mathbb{R}^{m \times d'}$. Note that, the byte embedding matrix $\mathbf{B}$ can be either a real-valued or one-hot embedding matrix because the vocabulary size is small for bytes. We compare the performances for both embedding methods in experiments.

4.3 Complexity Anlysis

Embedding	Memory	Time
Subword	$\mathcal{O}(V_w d)$	$\mathcal{O}(m^2 d)$
Byte	$\mathcal{O}(V_b d)$	$\mathcal{O}(c^2 m^2 d)$
SEB (Ours)	$\mathcal{O}((nd + V_b)d)$	$\mathcal{O}(m^2 d)$

Table 1: Complexity for conventional subword embeddings, byte embedding, and our proposed SEB.

To demonstrate the efficiency of the proposed SEB, we summarize the space and time complexity of each embedding method in Table 1. Here, the column "Memory" represents the memory usage for each embedding, and the column "Time" shows the time complexity in Transformer attention. For simplicity, we let $d' = d$ and use one linear layer as FFN in SEB which contains nd^2 parameters.



(a) Batch size = 1 (b) Batch size = 4 (c) Batch size = 16

Figure 3: The distribution of subword number, unique subword number, and unique byte number in a batch when batch size is 1, 4, 16. The vocabulary sizes of subwords and bytes are 50K and 256.

In terms of space complexity, subword embeddings typically have an exponentially large vocabulary size V_w, exceeding 10^4, while the dictionary size of byte embeddings is no more than 256. For the proposed SEB, the number of parameters in embedding is $\mathcal{O}(nd^2 + V_b d) = \mathcal{O}((nd + V_b)d)$, including the FFN and byte embedding matrix. In practice, $nd + V_b \ll V_w$. As a result, both byte embeddings and our proposed SEB significantly reduce the memory cost required for embeddings. In B.6, we show the analysis for space complexity in our experiments. Regarding time complexity, we analyze the attention in the widely used Transformer (Vaswani et al., 2017). Given the sequence length m, byte embedding is more time-consuming since the input length is c times longer than subword embedding. Here c is the average ratio between the lengths of byte and subword sequences. Based on the statistics (Shaham and Levy, 2020), c is usually around 5. However, our proposed SEB maintains the same time efficiency as conventional subword embeddings because we preserve the subword sequence along with its boundaries.

One may consider the frequency analysis in cryptanalysis to get the original text if the attacker also has information about the tokens used for the text and the frequency of each byte. In Appendix C, we discuss the frequency analysis is not applicable to our proposed defense.

5 Experiment

We conduct experiments to demonstrate the advantages of SEB in reducing space complexity, maintaining time efficiency, and preserving privacy for NLP models in federated learning. In all experiments, we set $V_b = 256$ and $n = 8$, which is sufficient to prevent encoding two subwords into the same byte sequences. We use a 2-layer FFN in the proposed SEB.

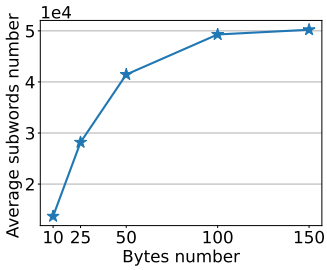


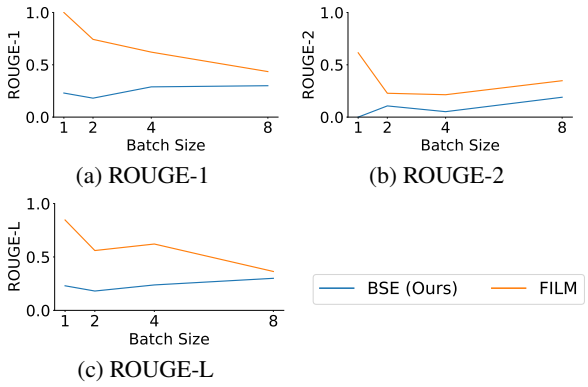
Figure 4: The average coverage of subwords given a random set of bytes with GPT-2 tokenizer.

5.1 Experiments on Privacy Protection

Dataset, attack task, and evaluation metrics We followed the settings in the FILM attack (Gupta et al., 2022). The dataset is WikiText-103 (Merity et al., 2016). For the attack task, we use GPT-2 base (Radford et al., 2019) with 117M parameters to recover the input batches. The ROUGE-1/2/L F-Scores (Lin, 2004) are used to evaluate the similarity between the recovered and original text.

Quantitative analysis of defense We first show that it is difficult to retrieve a bag of candidate subwords in SEB with Figure 3 and 4. In Figure 3, we present the distributions of the subword number, unique subword number, and unique byte number in a client's batch of data. We observe that even a single sample contains over 120 unique bytes on average, while only having approximately 25 unique subwords. In Figure 4, we present the average coverage of subwords for a subset of bytes. Based on Figure 4, 120 bytes cover about 50K subwords. It means recovery is a random generation using almost the entire vocabulary.

Additionally, Figure 5 shows the FILM attack performances using various batches on WikiText-103, with subword embedding and SEB. As the candidate subwords are almost the whole vocabulary, beam search takes huge memory which is not executable on our device. To show the defense performance, we loose the constraints and randomly



(a) ROUGE-1 (b) ROUGE-2 (c) ROUGE-L

Figure 5: Recovery performance for batch size 1, 2, 4, 8 on WikiText-103.

sample 7,000 subwords, combined with the subwords in the original text. We randomly select 5 tested batches for each batch size and take the average ROUGE F-Scores. When batch size is 1, ROUGE-1/2/L scores are close to 1 for attacks with subword embedding, indicating a nearly perfect recovery. However, these scores are quite low when using SEB, showing the effectiveness of SEB to defend the attacks based on embedding gradients.

Qualitative analysis of defense To intuitively show the difference between the recovered sentences of FILM using subword embedding and the proposed SEB, we select the best-recovered sentences of these two methods based on the ROUGE-L F-score and list the results in Table 2. In the recovered sentence with the subword embedding, all words are successfully retrieved and have a very close order to the original sentence. However, with SEB, only a few words are retrieved, and many of them are stop words. The results show that SEB can prevent the attacker from recovering private information in the original sentence even though the batch only contains one sentence.

We also compare our method with the defense method of gradient pruning in the FILM attack (Gupta et al., 2022) for batch size $= 8, 16, 32$. The results are shown in Appendix B.7.

5.2 Experiment on Performance

To provide a more comprehensive assessment of our proposed technique's applicability and performance of federated learning and across different NLP tasks, we conduct experiments on machine translation, sentiment analysis, and Language Modeling (LM) tasks. The environmental settings are described in Appendix B.1.

5.2.1 Translation

Dataset and evaluation metrics In the translation task, we consider two datasets, one is the medium-size IWSLT14 (Cettolo et al., 2014) dataset and a large scale dataset WMT14 (Bojar et al., 2014). We follow the settings as prior work (Shaham and Levy, 2020; Zhang and Xu, 2022) and translate German (de) to English (en) in IWSLT14 (Cettolo et al., 2014). The translation of WMT is English (en) to German (de) and the preprocessing is the same as Fairseq (Ott et al., 2019). We use SacreBLEU, case-sensitive, with the 13a tokenizer (Post, 2018) as the evaluation metric. A detailed description of preprocessing, model architecture, and hyperparameter settings can be found in Appendix B.3.

Main results For IWSLT14, we run 5 trials and report the average performance with the standard deviation. We show the translation results of Transformer with subword embedding and SEB in Table 3. The hidden dimension of the two-layer FFN is 2048 for IWSLT because we try to keep the total parameters of SEB_{co} the same as the original Transformer. For WMT, the hidden dimension of FFN is 4096. Here, we test three variants of SEB when aggregating the byte embedding back to subword embedding: added real-valued embedding (SEB_{ar}), concatenated real-valued embedding (SEB_{cr}), and concatenated one-hot embedding (SEB_{co}). In this experiment, the dimensions of real-valued and one-hot vectors are 512 and 256. Table 3 shows that SEB_{cr} and SEB_{co} can achieve better performances than subword embedding. Concatenating the one-hot vectors yields better results even with fewer model parameters than concatenating byte embedding. Therefore, we can conclude that SEB is a better alternative to using large subword embeddings. Additionally, based on the comparison between SEB_{ar} and SEB_{cr}, we find that concatenation is better than the simple adding of byte embeddings. This is expected as Section 4.2 because adding does not consider the positional information of bytes. The result of WMT14 shows the same performance as the subword-based model but with a smaller size of embedding parameters. It is important to emphasize that while privacy is improved, our model achieves the same or better accuracy than the baseline methods.

Sensitivity analysis on FFN hidden units In this experiment, we test the sensitivity of SEB_{co} on FFN hidden units, because it is one of the major factors for embedding parameters. Here, we set different FFN hidden units as

	Original Sentence	Best Recovered Sentence
Subword	The historic rainfall caused several dams to fill throughout northeast Mexico.	The rainfall caused several historic dams to fill throughout northeast Mexico.
SEB	Pujols is a highly regarded hitter who has shown a "combination of contact hitting ability, patience, and raw power"	He is a professional who has a very high degree of ability, and always takes great advice, without ever assuming power" ("Pivotal Decision Making With Your Head". Retrieved 12 Dec 2007 16 Mar)

Table 2: The best recovered sentences by FILM using subword embedding and SEBwith batch size 1. Text in green are successfully recovered phrases and words.

Datasets	Embeddings	# Params	BLEU
IWSLT14	Subword	5.2M	34.54 ± 0.10
	SEB_{ar}	4.3M	34.64 ± 0.15
	SEB_{cr}	9.6M	35.32 ± 0.15
	SEB_{co}	5.2M	$\mathbf{35.44 \pm 0.10}$
WMT14	Subword	22.3M	26.0
	SEB_{co}	6.3M	26.0

Table 3: BLEU score of IWSLT14 and WMT14. SEB_{ar} and SEB_{cr}: SEB with added and concatenated real-valued embeddings, respectively. SEB_{co}: SEB with concatenated one-hot byte embeddings.

$\{128, 256, 512, 1024, 2048, 4096\}$, with the total embedding parameter numbers of 0.3M, 0.7M, 1.3M, 2.7M, 5.2M, and 10.5M, respectively. The number of embedding parameters and translation BLEU scores are shown in the left of Figure 6. When the numbers of hidden units are 256, 512, and 1024, SEB_{co} can obtain better performance with fewer parameters. Although the model can still achieve better performance when hidden units are larger than 2048, it does not have advantages over the original transformer on model size.

Byte Tokens per Subword (n)	Byte Vocabulary Size (V_b)		
	64	128	256
4	0.79M	1.05M	1.57M
8	1.05M	1.57M	2.62M
16	1.57M	2.62M	4.72M

Table 4: Number of embedding parameters.

Sensitivity analysis on V_b and n To investigate the impact of the byte vocabulary size V_b and number of bytes per subword n, we set V_b as $64, 128, 256$ and n as $4, 8, 16$. Based on the previous experiments, we set the hidden units in the 2-layer FFN to 1024 in SEB_{co}, which provides good performance with a small scale of parameters. We first discuss the model size in terms of embedding parameter numbers in Table 4. All settings have smaller embedding parameter numbers than the original Transformer. We further demonstrate the translation performance under these settings in Figure 6 (right). It indicates that increasing n leads to better model performance for a fixed V_b. The reason is increasing n results in more possible positions per byte token, providing more information in the aggregated vector. Similarly, when we fix n and increase V_b, the increased byte vocabulary diversity makes the aggregated vector more expressive. Therefore, increasing the byte vocabulary size and the number of byte tokens per subword can improve the model's expressiveness, leading to improved performance. Furthermore, Figure 6 (right) and Table 4 show that models with similar amounts of parameters have similar performance. As long as V_b and n ensure that SEB_{co} has sufficient expressive ability, the model performance is more related to the number of parameters than to specific V_b and n.

5.2.2 Sentiment Analysis

Dataset and evaluation metrics We use IMDb (Maas et al., 2011) and SST2 (Socher et al., 2013) datasets provided by Hugging Face. The detailed preprocessing of the dataset is shown in Appendix B.3. We use the accuracy for evaluation which is a routine in prior work (Minaee et al., 2019; Yenter and Verma, 2017). The implementation details are in Appendix B.4.

Main results We compare the same BiLSTM models with subword embedding and SEB_{co}. The classification accuracies are shown in Table 5. The results show that SEB_{co} can replace the conventional subword embedding without hurting the model performance. For SST2, SEB_{co} even has better performance. The reason for that is the parameters of the conventional subword embedding layer in BiLSTM take a large portion of the model parameters, making the model easily overfitting. In this experiment, SEB_{co} has smaller embedding pa-

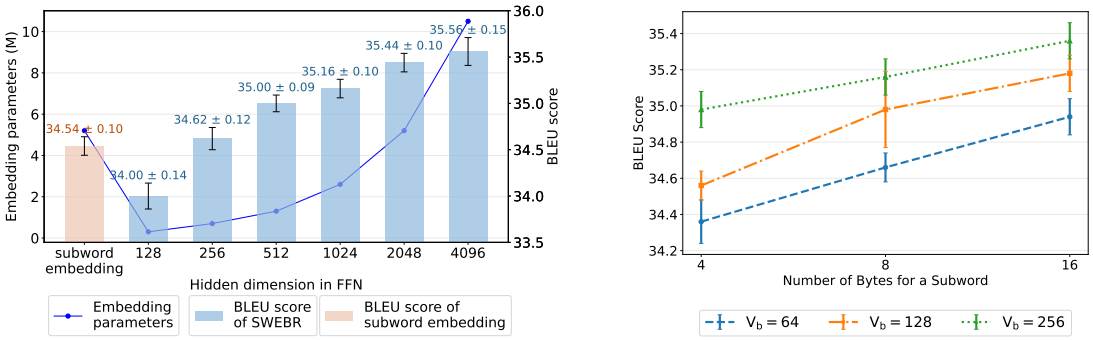


Figure 6: Results on embedding parameters, vocabulary size, and number of bytes per subword. Left: The BLEU scores versus hidden dimension in FFN and embedding parameters. Right: Comparison of mean BLEU scores for different byte vocabulary sizes and different numbers of bytes per subword.

rameters, which can address overfitting. We show that SEB also learns the semantic meaning of subword in B.5.

	IMDb (%)	SST2 (%)
Subword	85.6 ± 0.5	81.2 ± 0.7
SEB$_{co}$	**85.8 ± 0.2**	**82.5 ± 0.7**

Table 5: Results on Sentiment analysis.

5.2.3 Language Modeling

Dataset and evaluation metrics We use the same data as Fairseq did for the language modeling tasks. The dataset we use is WikiText-103. We use the same preprocessing and training settings as the official Fairseq does. The number of samples for training, testing, and validation are 1801350, 3760, and 4358 respectively. We evaluate the language modeling performance with perplexity.

Main results For language modeling (LM), our proposed method achieved better performance on perplexity while using a smaller size of parameters. The results are shown in Table 6, which demonstrate that our method SEBis an effective and efficient alternative to the traditional subword embedding.

	# Paramters	Perplexity
Subword	13.7M	30.84
SEB$_{co}$	10.5M	30.55

Table 6: Perplexity of language modeling for subword embedding and SEB.

5.2.4 Federated Learning

we experiment with federated learning on the SST2 sentiment analysis task using the FedAvg framework(McMahan et al., 2017). In this experiment, we have 20 clients and distribute training samples uniformly to these clients. In training, we sample a part of the clients with the ratio c in every communication round. The results are shown in the following table.

Embeddings	c=0.2	c=0.4	c=0.6	c=0.8	c=1.0
Subword	81.5%	80.6%	81.1%	80.7%	80.9%
SEB	82.0%	81.7%	81.9%	82.4%	81.7%

Table 7: Accuracies for subword embedding and our method for federated learning. c is the participation ratio in each communication round.

We can see that even in the federated learning framework, our SEBmethod is still comparable to subword embedding and can achieve stable results when the number of clients in training varies.

6 Conclusion

This paper introduces SEB, Subword Embedding of Bytes, a novel subword embedding method that defends against privacy leakage attacks based on embedding gradients in federated learning. Different from traditional approaches that learn a large subword embedding matrix, SEB uses smaller byte embedding matrices or byte one-hot encoding and aggregates byte representations to obtain subword embeddings. With SEB, attackers cannot retrieve a small set of subwords and generate private text, even with a well-trained large language model. Our extensive experiments show that SEB is effective for machine translation, sentiment analysis and language modeling tasks without sacrificing model performance or efficiency. Additionally, we demonstrate that SEB makes it difficult for attackers to recover private text with embedding gradients in federated learning.

7 Limitations

Despite the SEB can preserve data privacy without sacrificing efficiency or accuracy in different language models and tasks, it still has some limitations and and can be explored further.

Large language model and more tasks Limited to the computation resources, the Transformer models in our experiments are small. We also only experiment with moderate size of datasets. Moreover, we consider three common tasks such as machine translation, sentiment analysis, and language Modeling to show the effectiveness of our proposed method. The efficiency and effectiveness of our proposed method on large language models as well as other natural language processing tasks still need exploration.

Pretraining exploration All of the models in the experiments are trained from scratch. We have not experimented with the pretraining and finetuing/prompting paradigm with SEB. However, our proposed SEB is an effective alternative to subword embedding, and we think that our method is generalizable to popular NLP models, tasks, and training paradigms. In the future, we will further investigate the performance of our proposed method under the pretraining and finetuing/prompting paradigm.

8 Ethical Consideration

In this work, we use the IWSLT14 (Cettolo et al., 2014), WMT14 (Bojar et al., 2014), SST2 (Socher et al., 2013), IMDb (Maas et al., 2011), and WikiText-103 (Merity et al., 2016). All these 5 datasets are widely used pubic datasets in NLP tasks, which do not have personal or sensitive information. For the first four datasets, they are freely used without any license. For the WikiText-103 dataset, it is used under the "Creative Commons Attribution-ShareAlike License".

We believe our work does not bring more risk in training and using NLP models. Our intention is to provide practical privacy-preserving collaboration and build trust among clients while maintaining the model's performance and efficiency. We hope this work can motivate more effective defenses, and encourage secure and privacy-preserving collaborations in practical applications and have a positive effect on popular large-scale language models.

References

Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. 2017. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345.

Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. 2022. Lamp: Extracting text from gradients with language model priors. *Advances in Neural Information Processing Systems*, 35:7641–7654.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign. In *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, pages 2–17.

Jieren Deng, Yijue Wang, Ji Li, Chenghong Wang, Chao Shang, Hang Liu, Sanguthevar Rajasekaran, and Caiwen Ding. 2021. Tag: Gradient attack on transformer-based language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3600–3610.

Haokun Fang and Quan Qian. 2021. Privacy preserving machine learning with homomorphic encryption and federated learning. *Future Internet*, 13(4):94.

Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. 2022. Recovering private text in federated learning of language models. *arXiv preprint arXiv:2205.08514*.

Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. 2021. Evaluating gradient inversion attacks and defenses in federated learning. *Advances in Neural Information Processing Systems*, 34:7232–7241.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. 2021. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the*

Association for Computational Linguistics: Human Language Technologies, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Shervin Minaee, Elham Azimi, and AmirAli Abdolrashidi. 2019. Deep-sentiment: Sentiment analysis using ensemble of cnn and bi-lstm models. *arXiv preprint arXiv:1904.04206*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Uri Shaham and Omer Levy. 2020. Neural machine translation without embeddings. *arXiv preprint arXiv:2008.09396*.

Uri Shaham and Omer Levy. 2021. Neural machine translation without embeddings. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 181–186.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Makesh Narsimhan Sreedhar, Xiangpeng Wan, Yu Cheng, and Junjie Hu. 2022. Local byte fusion for neural machine translation. *arXiv preprint arXiv:2205.11490*.

Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint arXiv:2106.12672*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wenqi Wei, Ling Liu, Margaret Loper, Ka-Ho Chow, Mehmet Emre Gursoy, Stacey Truex, and Yanzhao Wu. 2020. A framework for evaluating gradient leakage attacks in federated learning. *arXiv preprint arXiv:2004.10397*.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Alec Yenter and Abhishek Verma. 2017. Deep cnn-lstm with combined kernels from multiple branches for imdb review sentiment analysis. In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 540–546.

Xuefei Yin, Yanming Zhu, and Jiankun Hu. 2021. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, 54(6):1–36.

Mengjiao Zhang and Shusen Wang. 2021. Matrix sketching for secure collaborative machine learning. In *International Conference on Machine Learning*, pages 12589–12599. PMLR.

Mengjiao Zhang and Jia Xu. 2022. Byte-based multilingual NMT for endangered languages. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4407–4417, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems*, 32.

A SEB Algorithm

Algorithm 1 illustrates the detailed steps of our proposed subword to byte sequence mapping in Section 4.1. Algorithm 2 gives the step-by-step operations to obtain the subword embedding with SEB based on bytes, given the mapping in Algorithm 1.

Algorithm 1: Construction of Subword to byte sequence mapping

Input : Byte vocabulary $\mathcal{V}_b = \{0, 1, \ldots, V_b - 1\}$; Subword vocabulary $\mathcal{V}_w = \{w_0, w_1, \ldots, w_{V_w-1}\}$; The number of bytes per subword n

Output : Mapping: $\mathcal{M} : \mathcal{V}_w \to (\mathcal{V}_b)^n$

for $w_i \in D_w$ **do**
  repeat
    for $j \leftarrow 1$ *to* n **do**
      Sample $b_{ij} \sim$ Uniform[0, V_b)
      // $\mathtt{P}(b_{ij}) = \frac{1}{V_b}, \quad b_{ij} \in \mathcal{V}_b$
    end
  until $(b_{i1}, b_{i2}, \ldots, b_{in}) \notin \mathcal{M}$
  Add $w_i \to (b_{i1}, b_{i2}, \ldots, b_{in})$ to $\mathcal{M}$
end
return $\mathcal{M}$

Algorithm 2: Subword embedding with SEB

Input : Subword to byte sequence mapping $\mathcal{M} : \mathcal{V}_w \to (\mathcal{V}_b)^n$; Subword sequence $S = (w_1, w_2, \ldots, w_m)$; A feed-forward network $\mathtt{FFN}$; Embedding matrix is $\mathbf{B} \in \mathbb{R}^{V_b \times d}$; Subword embedding dimension d'

Output : Subword embeddings $\mathbf{E}' \in \mathbb{R}^{m \times d'}$

for $i \leftarrow 1$ *to* m **do**
  $(b_{i1}, b_{i2}, \ldots, b_{in}) \leftarrow \mathcal{M}[w_i]$
end
Byte sequence for $S : (b_{11}, \ldots, b_{1n}, \ldots, b_{m1}, \ldots, b_{mn})$
$\mathbf{E} \in \mathbb{R}^{mn \times d} \leftarrow$ retrieve $(b_{11}, \ldots, b_{1n}, \ldots, b_{m1}, \ldots, b_{mn})$ from $\mathbf{B}$
$\tilde{\mathbf{E}} \in \mathbb{R}^{m \times nd} \leftarrow$ reshape $\mathbf{E}$ (in a row-major order)
$\mathbf{E}' = \mathtt{FFN}(\tilde{\mathbf{E}}) \in \mathbb{R}^{m \times d'}$
return $\mathbf{E}'$

B Experimental Details and More Results

B.1 Enviormental Settings

All the programs in our work are implemented using Python 3.9.0, Fairseq (Ott et al., 2019), PyTorch 1.13.0, and CUDA 11.7. For the hardware environment, we run all codes on a machine with Intel i7-11700K CPU, 64G memory, and NVIDIA GeForce RTX 3080 GPU.

B.2 Dataset and evaluation metrics

Translation In the translation task, we consider two datasets, one is the medium-size IWSLT14 (Cettolo et al., 2014) dataset and a large-scale dataset WMT14 (Bojar et al., 2014). We follow the settings as prior work (Shaham and Levy, 2020; Zhang and Xu, 2022) and translate German (de) to English (en) in IWSLT14 (Cettolo et al., 2014). The translation of WMT is English (en) to German (de) and the preprocessing is the same as Fairseq (Ott et al., 2019). We use SacreBLEU, case-sensitive, with the 13a tokenizer (Post, 2018) as the evaluation metric.

Sentiment Analysis We use IMDb (Maas et al., 2011) and SST2 (Socher et al., 2013) datasets provided by Hugging Face. The detailed preprocessing of the dataset is shown in Appendix B.3. We use the accuracy for evaluation which is a routine in prior work (Minaee et al., 2019; Yenter and Verma, 2017).

Language modeling We use the same data as Fairseq did for the language modeling tasks. The dataset we use is WikiText-103. We use the same preprocessing and training settings as the official Fairseq does. The number of samples for training, testing, and validation are 1801350, 3760, and 4358 respectively. We evaluate the language modeling performance with perplexity.

B.3 Preprocessing Details

Translation For IWSLT14, there are 166K sentence pairs for training and validation and 5.6K for testing. The vocabulary shared by the source and target languages is built by BPE (Sennrich et al., 2015) with 10K tokens.

For WMT14, en-de contains 4.5M sentence pairs. Newstest2013 is used for validation and newstest2014 for testing respectively. The merge operation is 32K for BPE and the dictionary is shared by source and target.

Sentiment analysis There are 25000 training samples and 25000 testing samples for IMDb. We take 25% of the training data for validation and the rest for training. For SST2, The training, validation, and test examples in SST2 are 67349, 872, and 1821, respectively. The tokenizer is "basic_english" in the TorchText package. The minimum frequency needed to include a token in the vocabulary is 5. The maximum length of the sentence is 256.

	good	great	funny	bad	worse	boring
good	1	0.63	0.49	-0.58	-0.61	-0.58
great	0.63	1	0.40	-0.53	-0.33	-0.38
funny	0.49	0.40	1	-0.72	-0.61	-0.60
bad	-0.58	-0.53	-0.72	1	0.72	0.85
worse	-0.61	-0.33	-0.61	0.72	1	0.88
boring	-0.58	-0.38	-0.60	0.85	0.88	1

Table 8: The cosine similarity of the subword embeddings calculated based on SEB.

B.4 Implementation Details

Translation The baseline we compare is the transformer with subword embedding (Vaswani et al., 2017). Our proposed method only replaces the subword embedding with SEB.

For IWSLT14, the encoder and decoder layers are both 6 and have 4 attention heads. The hidden dimension of attention is 512 and the dimension of the feedforward layer is 1024. The optimizer is Adam (Kingma and Ba, 2014) with an inverse square root learning rate scheduler, and warm up 4000 steps. The learning rate is 5×10^{-4}. The total training epochs are 100, and we average the best 5 checkpoints for testing.

For WMT14, the encoder and decoder layers are both 6 and have 8 attention heads. The hidden dimension of attention is 512 and the dimension of the feedforward layer is 2048. The optimizer is Adam (Kingma and Ba, 2014) with an inverse square root learning rate scheduler, and warm up 4000 steps. The learning rate is 5×10^{-4}. The total training epochs are 100 and we use the early stop if the validation loss does not decrease in 5 epochs. We average the best 5 checkpoints for testing.

Sentiment Analysis We use 2-layer BiLSTMs for both IMDb and SST2 classification tasks. We keep all model architectures the same for the baseline models and models with our SEB$_{co}$ except for the embedding parts. The subword embedding dimension is 64 and 256 for IMDb and SST2. The hidden units are 64 and 300 for IMDb and SST2. The hidden dimension of 2-layer FFN in SEB$_{co}$ is 128 for both datasets. We optimize the model using Adam (Kingma and Ba, 2014) and the learning rate is 5×10^{-4} for the baseline and our method on both datasets. The best model parameters evaluated on validation data are applied for testing.

Language modeling In this experiment, we also use a two-layer FFN in SEB, which has 4096 hidden units. The architecture is `transformer_lm` in the Fairseq framework. We share the input and output embedding in the encoder and the other hyperparameters and settings are the same as Fairseq.

B.5 Analysis for Semantic Meaning

In this section, we will analyze whether the derived subword embeddings from our method can truly encode the meaning of the words in the embedding space. To experiment on this aspect, we mainly calculate the cosine similarity between two word embeddings obtained based on our method SEB. We list some examples in IMDb sentiment analysis in Table 8.

The cosine similarity demonstrates that the subword embedding of our proposed SEBwill learn the semantic meaning from the task. For example, positive words (good, great, and funny) have positive and high-value similarities with each other, which is also the same case for all negative words (bad, worse, and boring). However, all the negative-positive pairs have negative similarities, which means the subword embedding of our proposed SEBcan automatically learn the semantic meaning.

B.6 Analysis for Space Complexity

We analyze the space complexity in the experiments. We mainly take the translation on IWSLT14 and sentiment analysis as examples. Transformer model for translation on IWSLT14 de-en with 256 hidden units in our method SEBas the translation results are close to the traditional subword embedding.

The tables below present the sizes of both the entire model's parameters and the embedding layer for translation on IWSLT and sentiment analysis, respectively. The numbers in "()" represent the percentage reduction achieved by our method compared to the subword model.

In all of these tasks, our method SEB$_{co}$ can decrease the space complexity, which shows the ability of our method to reduce the model size. In scenarios where model training is necessitated on a device with limited memory, it will be better to

# Params	Whole model	Embedding	BLEU
Subword	37M	5.2M	34.54 ± 0.10
SEB_{co}	33M ($\downarrow$ 12%)	0.7M ($\downarrow$ 94%)	34.62 ± 0.12

Table 9: Transformer model for translation on IWSLT14 de-en.

# Params	Whole model	Embedding	Accuracy
Subword	5.9M	2.4M	81.2 ± 0.7
SEB_{co}	3.8M ($\downarrow$ 36%)	0.8M ($\downarrow$ 68%)	82.5 ± 0.7

Table 10: Sentiment analysis on SST2

make the model smaller while keeping the model's performance.

B.7 Comparison with Gradient Prune Defense

We compare the defense method of gradient pruning in the FILM attack for batch size = $8, 16, 32$. Tables 12,13, and 14 show the precision and recall for gradient pruning and our method. Even without pruning, our method has a very low recall compared to FILM on subword embeddings when all batch sizes we experimented on, which shows the effectiveness of our defense.

C Discussion of Frequency Analysis

Frequency analysis is useful in cryptanalysis. For simple substitution ciphers, there is a characteristic distribution of letters that is roughly the same for almost all samples of that language. Frequency analysis uses the characteristic distributions of the plaintext and ciphertext to guess the mapping between them.

In the scenario of the threat model and our proposed method, the attacker only knows the gradients, model parameters and the mapping from subword to byte sequence. Based on these, the attacker can only get the information about distinct byte candidates which are updated in training. The attacker cannot determine the frequencies of the bytes based on the gradients.

To demonstrate the effectiveness of our method, we further assume the attacker have the information about the frequency of each byte. The goal of the attacker is to get the plaintext, given the plaintext to ciphertext mapping, and the characteristic distributions of the ciphertext.

To infer the plaintext, the attacker needs to know the order for combining all of the byte candidates and then use the ciphertext to plaintext mapping to obtain the original text. However, the attacker only knows a bag of bytes without ordering. It is difficult to infer the correct combination of the bytes as possible combinations of bytes is extremely large.

# Params	Whole model	Embedding	BLEU
Subword	1.5M	1.5M	85.6 ± 0.5
SEB_{co}	0.4M ($\downarrow$ 72%)	0.5M ($\downarrow$ 80%)	85.8 ± 0.2

Table 11: Sentiment analysis on IMDb

Prune ratio	Precision		Recall	
	Subword	SEB	Subword	SEB
0	1	1	1	0.003
0.9	1	1	1	0.003
0.99	1	1	1	0.003
0.999	1	1	0.53	0.003
0.9999	1	0.46	0.08	0.003

Table 12: Defense results on precision and recall for batch size is 8.

Prune ratio	Precision		Recall	
	Subword	SEB	Subword	SEB
0	1	1	1	0.005
0.9	1	1	1	0.005
0.99	1	1	1	0.005
0.999	1	1	0.49	0.005
0.9999	1	0.50	0.06	0.005

Table 13: Defense results on precision and recall for batch size is 16.

Prune ratio	Precision		Recall	
	Subword	SEB	Subword	SEB
0	1	1	1	0.009
0.9	1	1	1	0.009
0.99	1	1	1	0.009
0.999	1	0.99	0.51	0.009
0.9999	1	0.47	0.07	0.009

Table 14: Defense results on precision and recall for batch size is 32.