MERGE: Fast Private Text Generation

Zi Liang, Pinghui Wang*, Ruofei Zhang,
Nuo Xu, Shuo Zhang, Lifeng Xing, Haitao Bai, Ziyang Zhou

MOE KLINNS Lab, Xi'an Jiaotong University, Xi'an 710049, P. R. China
{liangzid, zs412082986, xlf20200926, haitao.bai, dakandao}@stu.xjtu.edu.cn,
phwang@mail.xjtu.edu.cn, rfzhang@gmail.com, nxu@sei.xjtu.edu.cn

Abstract

The drastic increase in language models' parameters has led to a new trend of deploying models in cloud servers, raising growing concerns about private inference for Transformer-based models. Existing two-party privacy-preserving techniques, however, only take into account natural language understanding (NLU) scenarios. Private inference in natural language generation (NLG), crucial for applications like translation and code completion, remains underexplored. In addition, previous privacy-preserving techniques suffer from convergence issues during model training and exhibit poor inference speed when used with NLG models due to the neglect of time-consuming operations in auto-regressive generations. To address these issues, we propose a fast private text generation framework for Transformer-based language models, namely MERGE. MERGE reuses the output hidden state as the word embedding to bypass the embedding computation and reorganize the linear operations in the Transformer module to accelerate the forward procedure. Extensive experiments show that MERGE achieves a 26.5x speedup to the vanilla encrypted model under the sequence length 512, and reduces 80% communication cost, with an up to 10x speedup to state-of-the-art approximated models.

Introduction

Recently, from pre-trained language models (PLMs) to large language models (LLMs), Transformer (Vaswani et al. 2017) based models have attracted significant attention because of their exceptional performance in downstream tasks. Due to the high demand for computing power, this growth of model parameters also has caused the trend of hosting models to cloud service providers, which raises wide concerns about privacy inference and training. For example, existing natural language processing (NLP) services like Copilot[1] and ChatGPT require users to submit their queries in plain text, which may contain confidential information such as source code, medical information, and personal preferences.

To alleviate the privacy problem, recent works (Hao et al. 2022; Chen et al. 2022) have developed two-party secure inference services for PLMs by secure Multi-Party Computation (MPC). MPC ensures the privacy of user data and model weights, and shares them secretly. However, PLMs inference under MPC is considerably slow compared to the plain-text version, which limits its application in real-world services. To address this issue, several works have attempted to simplify the bottleneck operations such as activation functions and softmax in the Transformer model. For instance, (Mishra et al. 2020) uses Neural Architecture Search (NAS) to replace the activation functions with linear layers, and (Li et al. 2022) approximates the exponential operation with polynomial functions.

Though designed for Transformer, existing works (Hao et al. 2022; Chen et al. 2022; Li et al. 2022) solely explore the scenario of natural language understanding (NLU) (e.g., on the GLUE (Wang et al. 2019) benchmark). Unfortunately, we observe that they have no significant improvements in natural language generation (NLG) tasks (cf., Fig. 1). By illustrating the bottleneck of NLU and NLG inference procedures, we find that auto-regressive generation used in PLMs suffers from extra time cost in *embedding table query* and *token sampling* (i.e., GenTime), which slows down the whole inference procedure heavily.

In this paper, we explore accelerating the generation procedure of language models. Different from existing works that merely approximate the nonlinear operations, we consider the optimization at the architecture level and attempt to reorganize and simplify the whole generation procedure as well as Transformer modules. To this end, we propose *MERGE* (short for **M**PC-based **E**mbedding **R**esending **GE**neration), a fast and easy-to-adopt framework for private text generation. *MERGE* is compatible with previous MPC-based works (e.g., MPCformer, THE-X, and IRON) and mainstream PLMs (e.g., GPT-2 (Radford et al. 2019), T5 (Raffel et al. 2020), and Bart (Lewis et al. 2020)). In concrete, *MERGE* can simplify the time-consuming operations in NLG such as embedding query and token sampling. To achieve that, we first put forward a strategy called **embedding resending**, which directly uses the output hidden state as the new input token embedding. Embedding resending helps to bypass the *embedding table query* operation and decouple the computation between *forward representation learning* and *next token sampling*. Besides, following the recent research (Hassid et al. 2022) in attention mechanism, we approximate *self-attention* with *constant attention* matrices and merge tensor computations in the Transformer module before inference. Nevertheless,

*Corresponding author.

[1]https://github.com/features/copilot

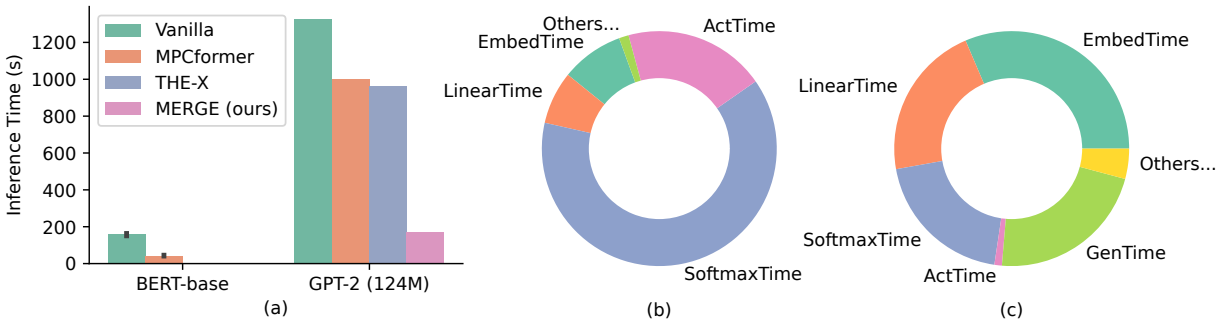


Figure 1: Encrypted inference time comparisons among BERT-base and GPT-2 with sequence length 128, where MPCformer, THE-X, and MERGE in Fig. (a) are different private inference methods. While nonlinear operations such as softmax (SoftmaxTime) and activation functions (ActTime) account for a substantial portion of inference time in BERT-base (Fig. (b)), the inference time is more balanced across operations for NLG models (Fig. (c)), with non-trivial time consumption from linear computations (LinearTime), embedding table query (EmbedTime), and token sampling (GenTime).

these two strategies are challenging because: 1) PLMs are usually sensitive to input embeddings, while there are some unavoidable errors in the generated embeddings; 2) constant attention in our *merge module* might hurt the performance of PLMs. To address the above challenges, we first propose an embedding alignment and augmentation task to enhance the robustness of PLMs about input embeddings. Besides, we employ a weighted distillation training task for approximation models, which allows us to alleviate the negative effects of constant attention. Our empirical experiments on popular text generation tasks such as E2E (Dusek, Novikova, and Rieser 2018), Multiwoz 2.1 (Eric et al. 2020), and DailyDialog (Li et al. 2017) demonstrate the effectiveness of *MERGE*. Specifically, it achieves a considerable speedup of 7.75x to GPT-2 and 10.89x to T5 under the sequence length 128, and 26.5x under sequence length 512, while maintaining an acceptable performance with losses in BERTscore (Zhang et al. 2020), BARTscore (Yuan, Neubig, and Liu 2021), and Rouge-L (Lin 2004) of only 0.02 (under 0.92), 0.14 (under -2.90), and 0.03 (under 0.44), respectively. Source code of experiments can be found here: https://github.com/liangzid/MERGE.

Related Work

Although existing MPC techniques can provide secure inference for neural networks, they usually suffer from prohibitively high communication delays and computation costs. This is primarily due to the critical nonlinear operations within neural networks. Therefore, some works aim to approximate these bottleneck operations in neural networks. For instance, (Chen et al. 2022) replaces the GeLU activation function in the Transformer with ReLU, and (Hao et al. 2022) reformulates the $Tanh(\cdot)$ function in GeLU based on optimized exponential operations. Besides, (Mishra et al. 2020) approximates the ReLU function with linear layers to replace the MPC method used for ReLU through the garbled circuits with secret sharing and Beaver triples. Similarly, (Li et al. 2022) approximates GeLU with ReLU and quadratic functions. For the softmax operation in the attention mechanism, (Li et al. 2022) approximates it by $softmax(x) \approx \frac{ReLU(x)}{\sum ReLU(x)}$ or $softmax(x) \approx \frac{(x+c)^2}{\sum (x+c)^2}$.

Nevertheless, these approximations were designed for the "one-time" inference of NLU models (e.g. BERT), and are not optimized for auto-regressive generative models (e.g. GPT-series) that execute the forward inference multiple times. By contrast, our work focuses on optimizing the overall generation procedure instead of some nonlinear operations, which leads to more transformative performance for Transformer-based language models.

Preliminary

Text Generation with Language Models

The text generation task (e.g. dialogue) aims to generate the desired sequence y (e.g. the response of the chatbot) under the given prefix text p (e.g. the dialogue context) with the language model $p_\theta(y|p)$. Typically, existing language models usually generate y in an *auto-regressive* manner, i.e.,

$$p_\theta(y|p) = \prod_{t=1}^{N_t} p(x_t^y|p, x_{<t}^y), \tag{1}$$

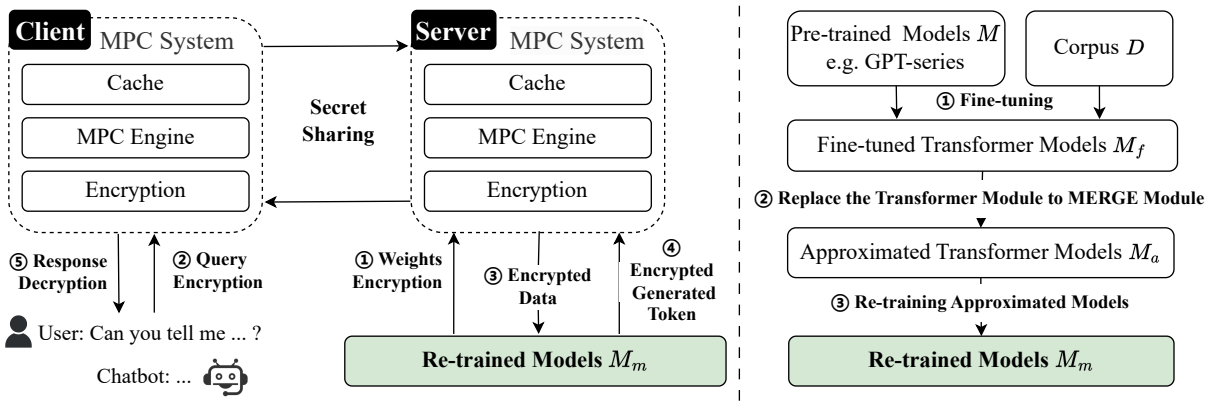
where x_t^y denotes the t-th generated token of y and $x_{<t}^y$ denotes the generated tokens of y at step t.

In Eq. (1), if we denote the one-hot representation of $(p, x_{<t}^y)$ as $\mathbf{x}_t$ with text length N_t, then the generation procedure consists of the following three stages:

a) Embedding table query: to obtain the initialized embeddings for each word, i.e., $\mathbf{E}_t = f_e(\mathbf{x}_t)$, where $f_e(\mathbf{x}) : \mathbb{R}^{N_t \times V} \to \mathbb{R}^{N_t \times d}$ is the embedding layer that maps the V-length index representation into the d-dimension semantic space to obtain the semantic embeddings $\mathbf{E}_t$.

b) Representation learning: to obtain the representations of inputs considering the contexts, i.e., $\mathbf{h}_t^{n_l} = \mathbf{f}_{tr}(\mathbf{E}_t')$, where $\mathbf{f}_{tr} : \mathbb{R}^{N_t \times d} \to \mathbb{R}^{N_t \times d}$ is an n_l-layer transformer model, $\mathbf{h}_t^{n_l}$ is the output hidden state, and $\mathbf{E}_t'$ is the combination of positional embeddings, token embeddings $\mathbf{E}_t$, and others.

c) Next token sampling that generated the new token, i.e., $x_t^y \sim f_{cls}(\mathbf{h}_t^{n_l})[N_t]$, where $f_{cls}(\mathbf{h}_t^{n_l}) : \mathbb{R}^{N_t \times d} \to \mathbb{R}^{N_t \times V}$ is the linear head for token prediction, and $f_{cls}(\mathbf{h}_t^{n_l})[N_t]$ which is the N_t-th element of $f_{cls}(\mathbf{h}_t^{n_l})$ denotes the probability distribution of in vocabulary for current sampled token,



(a) Private Inference (Online) (b) Prepare MERGE Models (Offline)

Figure 2: An overview of *MERGE* at inference (Fig. (a)) and re-training (Fig. (b)) stages.

and $\sim$ denotes the sampling strategy (e.g., greedy search) to obtain the new generated token x_t^y according to vocabulary distribution $f_{cls}(\mathbf{h}_t^{n_l})[N_t]$.

Transformer Module

In the above representation learning step, the Transformer model $\mathbf{f}_{tr}$ can be viewed as a stack of transformer modules. Here, we introduce the three key components of transformer module $f_{tr}^n : \mathbb{R}^{N_t \times d} \rightarrow \mathbb{R}^{N_t \times d}$ as follows:

a) Projection: to compute the subsequent self-attention, the transformer module first projects the input hidden state to the $(query, key, value)$ tuple, i.e.,

$$\mathbf{Q}^n, \mathbf{K}^n, \mathbf{V}^n = W_{Q^n}^T \mathbf{h}^{n-1}, W_{K^n}^T \mathbf{h}^{n-1}, W_{V^n}^T \mathbf{h}^{n-1},$$

where $W_{Q^n}, W_{K^n}, W_{V^n} \in \mathbb{R}^{d \times (d/N_h) \times N_h}$ are N_h-head projection matrices. Particularly, we have $\mathbf{h}^0 = \mathbf{E}_t'$.

b) Self-Attention is proposed to aggregate the context information into a new representation for each word base on the above $(\mathbf{Q}^n, \mathbf{K}^n, \mathbf{V}^n)$ tuple. Firstly, it calculates the correlation between contextual words, i.e., $A^n = f_{dr}(softmax(\mathbf{Q}^n \cdot (\mathbf{K}^n)^T / \sqrt{d_k}))$, where $A^n \in \mathbb{R}^{N_h \times N_t \times N_t}$ denotes the N_h-head attention matrix and $d_k = d/N_h$. Then, the new representations are weighted aggregated based on the above attention matrix, i.e., $\mathbf{x}_{att}^n = f_{ln}(f_{dr}(W_{d^n}^T \cdot (\text{Concat}(A^n \cdot \mathbf{V}^n)) + b_{d^n}) + \mathbf{h}^{n-1})$, where $W_{d^n} \in \mathbb{R}^{d \times d}$ is the weight matrix, $b_{d^n} \in \mathbb{R}^d$ is the bias, f_{dr} denotes the dropout operation (Srivastava et al. 2014), and f_{ln} is the layer normalization (Ba, Kiros, and Hinton 2016),

$$f_{ln}(\mathbf{x}) = \frac{\mathbf{x} - E[\mathbf{x}]}{\sqrt{Var[\mathbf{x}] + \epsilon}} \odot \gamma + \beta, \tag{2}$$

in which ϵ is a tiny number, $\odot$ denotes the element-wise product, and $E[\mathbf{x}]$ and $Var[\mathbf{x}]$ denote the mean and variance of $\mathbf{x}$, respectively.

c) Feed forward: to compute the output hidden state, a two-layer MLP is used, i.e., $\mathbf{h}^n = f_{ln}(f_{dr}(W_O^{nT} \cdot (\text{Act}(W_I^{nT} \cdot \mathbf{x}_{att}^n + b_I^n) + b_O^n) + \mathbf{x}_{att}^n)$, where $W_I^n \in \mathbb{R}^{d \times d_I}$ and $W_O^n \in \mathbb{R}^{d_I \times d}$ are weighted matrices, $b_I^n \in \mathbb{R}^{d_I}$ and $b_O^n \in \mathbb{R}^d$ are bias vectors, d_I is the dimension of intermediate states, and $\text{Act}(\cdot)$ denotes the activation functions such as ReLU (Agarap 2018) or GeLU (Hendrycks and Gimpel 2016).

Our Method

Overview

Fig. 2 gives an overview of how MPC systems work in private text generation and the role *MERGE* plays in the whole framework. As we can see, to enable private computations among multiple parties, the MPC system encrypts both the texts and the model parameters, and then send them securely with various secure techniques. Since it is imperative to accelerate the inference (shown in Fig. 1), our method *MERGE* aims to export an optimized model M_m to the original model M_f for MPC systems, which involves a two-step re-training procedure. Especially, it consists of an approximation of Transformer architecture, called *merge module* (MM), and a training task for a new generation strategy, called *embedding resending* (ER). Fig. 3 illustrates the details of them.

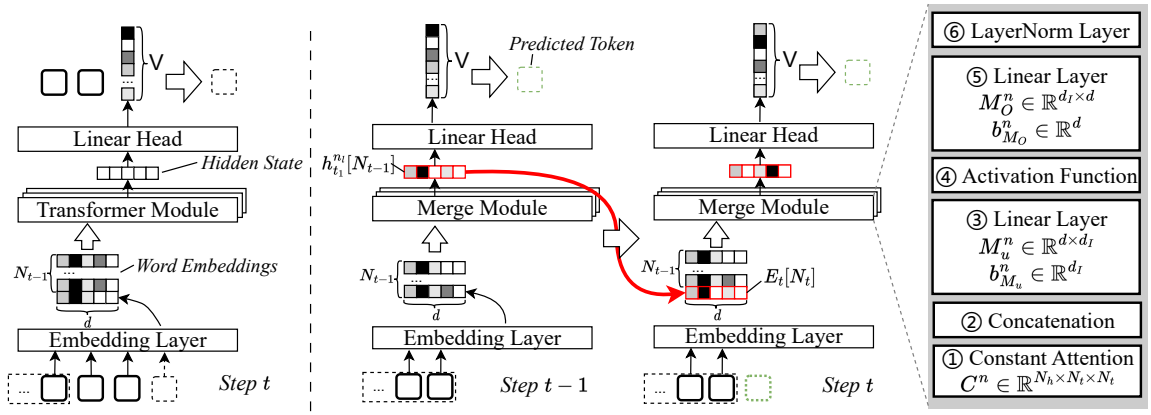
Embedding Resending (ER)

As shown in Fig. 3(b), the core idea of ER is borrowing the representation of Merge Module as the word embedding (shown as the red line). In detail, ER regards the hidden state $(\mathbf{h}_{t-1}^{n_l}[N_{t-1}])$ at step $t-1$ as $\mathbf{E}_t[N_t]$ at step t, i.e.,

$$\mathbf{E}_t = [\mathbf{E}_{t-1}; \mathbf{h}_{t-1}^{n_l}[N_{t-1}]] = [\mathbf{E}_0; \mathbf{h}_{t-1}^{n_l}], \tag{3}$$

where $\mathbf{E}_0$ denotes the token embeddings of the prefix p and ";" denotes the concatenation operation. In this way, our ER strategy achieves two aspects of optimization: 1) since we obtain $\mathbf{E}_t[N_t]$ without sending new token into the Embedding Layer, we avoid the time-consuming *embedding table query*; 2) since we only require the hidden states instead of sampled tokens in following generation procedure, we can execute the *token sampling* and *representation learning* in parallel.

Alignment optimization. Eq. (3) assumes *embedding table query* as the inverse procedure of *next token sampling*, which implies that hidden states and token embeddings are in the same representation space. Therefore, to align the representations $\mathbf{h}_{t-1}^{n_l}[N_{t-1}]$ and $\mathbf{E}_t[N_t]$, we design a training task that



(a) Vanilla Auto-Regressive Generation (b) Generation with MERGE (c) MERGE Module

Figure 3: Generation procedure and architecture of existing models (Fig. (a)) and *MERGE* (Fig. (b) and (c)).

maximizes the cosine similarity between these vectors, i.e.,

$$\mathcal{L}_c = \frac{1}{N_{tr} \cdot N} \sum_{i}^{N_{tr}} \sum_{t=1}^{N} 1 - cos(\mathbf{h}^{n_l}_{i,t-1}[N_{t-1}], \mathbf{E}_{i,t}[N_t]), \quad (4)$$

where function cos computes the cosine of the angle between two vectors, N_{tr} is the number of train set, and N denotes the sequence length. Here we select the cosine similarity instead of mean square error (MSE) because the inner product (e.g., *self-attention*) plays a key role in the Transformer module.

Robustness Optimization. Besides, we observe that the error of token embeddings significantly impacts the performance of the Transformer model $\mathbf{f}_{tr}$ and leads to nonsensical sentence generation with the MSE value over 0.05 (cf., Fig. 5). To enhance the robustness of $\mathbf{f}_{tr}$, we introduce an *embedding augmentation* method that first masks each element e_t in $\mathbf{E}_t$ with a probability p, and then adds a uniform noise sampled from a small interval $(-\epsilon, \epsilon)$ i.e.,

$$\tilde{e}_t = m_t \cdot (e_t + n_t), \quad (5)$$

where $m_t \sim$ Bernoulli$(1-p)$ and $n_t \sim$ Uniform$(-\epsilon, \epsilon)$. Based on the noised token embeddings $\tilde{\mathbf{E}}_t$, the cross-entropy loss can be reformulated as,

$$\mathcal{L}_{ce} = \frac{1}{N_{tr} \cdot N} \sum_{i}^{N_{tr}} \sum_{t=1}^{N} \mathbf{x}_t[N_t] \cdot \log f_{cls}(\mathbf{f}_{tr}(\tilde{\mathbf{E}}_t))[N_t]. \quad (6)$$

In this way, for a word embedding input in a noisy range, $\mathbf{f}_{tr}$ will learn to obtain a similar hidden state.

Therefore, the overall train loss is formulated as,

$$\mathcal{L} = \lambda \mathcal{L}_c + (1-\lambda)\mathcal{L}_{ce}, \quad (7)$$

where $\lambda \in [0, 1]$ is a weighting factor.

The Merge Module (MM)

To further accelerate the inference process, we also propose the merge module, which is an efficient approximation of the Transformer module, focusing on optimizing the computation of the *linear* and *softmax* functions.

Following recent research (Hassid et al. 2022), we first replace the dynamic self-attention matrix A^n with a constant attention matrix $C^n \in \mathbb{R}^{N_h \times N_t \times N_t}$. We initialize C^n with the average of A^n in train set, i.e.,

$$C^n = \frac{1}{N_{tr}} \sum_{i}^{N_{tr}} A^n_i \quad (8)$$

Besides, we approximate the layer normalization f_{ln} in *attention* with a simple element-wise multiplication $f'_{ln}(\mathbf{x}) = \mathbf{x} \odot \gamma + \beta$, inspired by the previous work (Chen et al. 2022). Consequently, the attention procedure now can now be approximated as,

$$\mathbf{x}^n_{att} = f'_{ln}(f_{dr}(W^{nT}_d \cdot (\text{Concat}(C^n \cdot \mathbf{V}^n)) + b^n_d) + \mathbf{h}^{n-1}). \quad (9)$$

Based on Eq. (9), we simplify the whole computation procedure by reorganizing matrix computations in f_{tr} and merging intermediate linear operations. Specifically, we merge the projection operation W^n_V, the linear map W^n_d, the approximated layer normalization function f'_{ln}, as well as the first linear map in feed-forward W^n_I into a single linear layer, i.e., a weighted matrix $M^n_u \in \mathbb{R}^{d \times d_I}$ and a bias term $b^n_{M_u} \in \mathbb{R}^{d_I}$, which are formatted as:

$$\begin{aligned} M^n_u &= (W_{V^n} \cdot W^n_d + \mathbf{1}) \odot \gamma \cdot W^n_I, \\ b^n_{M_u} &= W^{nT}_I \odot \gamma \cdot b^n_d + W^{nT}_I \cdot \beta + b^n_I, \end{aligned} \quad (10)$$

where $\mathbf{1} \in \mathbb{R}^{d \times d}$ is the residual term in attention module.

As Eq. (10) shows that no parameters dependent on input token embeddings $\mathbf{E}'_t$, we can pre-compute M_u and b_{M_u} before the inference stage, thus reducing the computation during model execution. Thus, we simplify the entire Transformer module into only three tensor multiplications, i.e.,

$$\begin{aligned} \mathbf{x}^n_o &= f_{mer}(\mathbf{h}^{n-1}) \\ &= f_{ln}(W^{nT}_O \cdot \text{Act}(M^{nT}_u \cdot C^n \cdot \mathbf{h}^{n-1} + b^n_{M_u}) + b^n_O). \end{aligned} \quad (11)$$

Although it may appear possible to merge M^n_u with the previous linear matrix W^{n-1}_O in Eq. (11) by approximating

Model	Time/ Communication Time (s)			Total Time (s)	Speedup
	Embed	Linear	Softmax		
GPT2-base (124M)					
CrypTen	321.44/52.33	251.93/74.21	454.61/113.96	1328.26	1x
MPCformer (sm2relu)	316.75/51.55	253.57/76.56	181.14/45.59	1001.41	1.33x
MPCformer (sm2quad)	318.16/50.88	253.30/75.16	152.45/37.40	972.50	1.36x
THE-X	329.29/58.30	258.00/80.21	87.71/19.28	965.79	1.37x
MERGE (ER+MM)	**5.17/0.87**	**157.50/53.97**	†**0.00/0.00**	**171.38**	**7.75x**
MERGE (only ER)	5.41/0.95	260.36/80.00	477.76/124.83	834.13	1.59x
MERGE (only MM)	320.84/50.92	250.98/81.57	†0.00/0.00	747.45	1.78x
T5 (138M)					
CrypTen	323.46/53.36	328.09/96.08	693.73/175.57	1569.41	1x
MPCformer (sm2relu)	327.51/55.36	328.61/96.80	284.65/75.17	1207.63	1.30x
MPCformer (sm2quad)	324.81/52.03	325.97/92.89	235.54/58.47	1149.07	1.37x
THE-X	316.16/48.58	321.90/90.82	126.73/25.51	1050.28	1.49x
MERGE (ER+MM)	**7.62/1.27**	**131.31/44.11**	†**0.00/0.00**	**144.02**	**10.89x**
MERGE (only ER)	8.24/1.58	211.57/65.19	596.74/166.50	874.36	1.79x
MERGE (only MM)	322.38/51.35	221.57/69.22	†0.00/0.00	693.30	2.26x

Model	Embed (Byte)	Linear (Byte)	Softmax (Byte)	Total (Byte)	Fraction
GPT2-base (124M)					
CrypTen	71.41GB	159.36GB	1.62GB	322.54GB	100.00%
MPCformer (sm2relu)	71.41GB	135.54GB	0.54GB	317.20GB	98.34%
MPCformer (sm2quad)	71.41GB	135.54GB	0.07GB	316.73GB	98.20%
THE-X	71.41GB	135.54GB	0.50GB	319.14GB	98.95%
MERGE (ER+MM)	**1.15GB**	**119.89GB**	†**0.00GB**	**121.76GB**	**37.75%**
MERGE (only ER)	1.15GB	160.63GB	1.62GB	168.51GB	52.24%
MERGE (only MM)	71.41GB	119.89GB	†0.00GB	281.88GB	87.39%
T5 (138M)					
CrypTen	147.14GB	199.97GB	7.72GB	380.45GB	100.00%
MPCformer (sm2relu)	147.14GB	199.97GB	2.73GB	364.74GB	95.87%
MPCformer (sm2quad)	147.14GB	199.97GB	0.33GB	362.33GB	95.24%
THE-X	147.14GB	199.97GB	2.97GB	369.73GB	97.18%
MERGE (ER+MM)	**1.73GB**	**95.66GB**	†**0.00GB**	**98.03GB**	**25.77%**
MERGE (only ER)	1.73GB	120.17GB	7.56GB	132.44GB	34.81%
MERGE (only MM)	73.72GB	95.66GB	†0.00GB	257.89GB	67.79%

Table 1: Inference time and communication costs comparison among different operations. Results marked with † come from the optimization in MM which replaces the dynamics attention with constant attention, thus having no cost in *softmax*. The remaining tables use the same notation system as outlined here.

the layer normalization f_{ln} with f'_{ln}, we choose to keep them separate for the following two reasons. Firstly, the merged matrix $W_O^{n-1} \cdot M_u^n \in \mathbb{R}^{d_I \times d_I}$ has significantly more parameters than W_O plus M_u, since d_I is typically larger than d. Secondly, removing f_{ln} in Eq. (11) will hurt the convergence of the merge module heavily during the training stage.

Obviously, to derive Eqs. (10) and (11), we need to swap W_v^n and C^n, which requires the verification that the matrix multiplications on the tensor $\mathbf{h}_t^{n-1}$ under different dimensions obeys the commutative law.

Experiments

Settings

Datasets. We evaluate *MERGE* on three representative text generation tasks, including Multiwoz (Eric et al. 2020), a human-human multi-turn task-oriented dialogue corpus, DailyDialog (Li et al. 2017), a multi-turn chitchat dataset, and CommonGen (Lin et al. 2020), a hard-constrained controlled text generation benchmark.

Baselines. We compare *MERGE* with state-of-the-art private inference models and frameworks, including THE-X (Chen et al. 2022), one of the first approximation architecture of Transformer models, MPCformer (Li et al. 2022), the approximated model that aims to accelerate the inference procedure of Transformer, and Crypten , one PyTorch version of the general MPC implementations based on secret sharing.

Evaluation Metrics. We evaluate *MERGE* in two dimensions: inference speed, and the effectiveness of approximation models. For inference speed, we record both the computation time and the communication volume for each method. For the effectiveness of PLMs, we use Meteor (Banerjee and Lavie 2005), CHRF++ (Popovic 2017), NIST (Lin and Och 2004), ROUGE family (Lin 2004), BERTscore (Zhang et al. 2020),

Model	BERTscore	BARTscore	NIST	Rouge-L	METEOR	CHRF++
MultiWoz NLG (Eric et al. 2020)						
GPT-2 (124M)	0.9237	-2.9020	4.7907	0.4424	0.4900	43.2777
+ER (no train)	0.6860	-5.0660	0.2325	0.0707	0.0425	3.9721
+MPCformer (sf2relu)	**0.9287**	**-2.5377**	**5.7248**	**0.4806**	**0.5792**	**48.8241**
+MPCformer (sf2quad)	OOT	OOT	OOT	OOT	OOT	OOT
+THE-X	OOT	OOT	OOT	OOT	OOT	OOT
+MERGE (ours)	0.8984	-3.1464	3.7444	0.3970	0.4302	36.6983
+MERGE only ER	0.9155	-2.8057	5.0812	0.4339	0.5102	44.2484
+MERGE only MM	0.9268	-2.6277	5.6524	0.4778	0.5647	47.7262
CommonGen (Lin et al. 2020)						
GPT-2 (124M)	**0.9336**	**-3.4710**	**3.7840**	**0.2744**	**0.3012**	**27.7038**
+ER (no train)	0.5999	-4.9864	0.0701	0.0192	0.0066	0.9470
+MPCformer (sf2relu)	0.8943	-4.1436	2.1301	0.1861	0.2691	27.6167
+MPCformer (sf2quad)	OOT	OOT	OOT	OOT	OOT	OOT
+THE-X	OOT	OOT	OOT	OOT	OOT	OOT
+MERGE (ours)	0.8821	-4.2479	0.6639	0.2025	0.1538	16.0573
+MERGE only ER	0.8953	-3.8979	1.6796	0.2430	0.2110	20.8878
+MERGE only MM	0.9083	-4.0885	2.2687	0.2026	0.2058	20.9888
DailyDialog (Li et al. 2017)						
GPT-2 (124M)	**0.8404**	-6.6387	0.5429	0.1142	0.1042	11.5089
+ER (no train)	0.7518	-6.8820	0.1287	0.0566	0.0526	6.8067
+MPCformer (sf2relu)	0.8161	-6.3494	**1.1102**	0.1322	0.1261	12.0713
+MPCformer (sf2quad)	OOT	OOT	OOT	OOT	OOT	OOT
+THE-X	OOT	OOT	OOT	OOT	OOT	OOT
+MERGE (ours)	0.8213	**-6.2384**	0.3674	0.1233	0.0955	7.8091
+MERGE only ER	0.8205	-6.5515	0.1069	0.1301	0.0833	6.5819
+MERGE only MM	0.8343	-6.5800	1.0499	**0.1525**	**0.1364**	**14.9039**

Table 2: Performance experiments of our *MERGE* method for private text generation.

and BARTscore (Yuan, Neubig, and Liu 2021) as the metrics.

Implementation Details

Hyperparameters. We use GPT-2 (124M) (Radford et al. 2019) as the basic evaluation backbone, with max sequence length 128. We trained all models under the learning rate 3×10^{-5}, batch size 4 with 3 epochs, based on the implementation of huggingface Transformers (Wolf et al. 2020). As for approximated models, we train our baselines under the same hyperparameter settings in their source code, and train *MERGE* with $50,000$ steps under the learning rate 8×10^{-5}. We set the dropout rate to 0.6, λ to 0.75, and noise to 0.75. It will cost 0.09 hours for every 1,000 steps.

Environments. All experiments above are on a single 32 GB Nvidia Tesla V100 GPU. Following previous works (Li et al. 2022), for the experiments of private inference, we use two 32 GB Nvidia Tesla V100 GPUs to simulate the client and the server, with 10 GbE Ethernet bandwidth. We implement the whole MPC system based on Crypten (Knott et al. 2021), a semi-honest MPC framework built on PyTorch.

Speed Evaluation

We evaluate the inference speed under two mainstream NLG architectures, i.e. the pure decoder represented by GPT-2, and the encoder-decoder models represented by T5, to investigate the speedup of ER and MM. As shown in Table 1, *MERGE* obtains a 7.75x speedup to the encrypted GPT-2, and 10.89x to T5, under the sequence length 128, while existing methods merely give the speedup less than 2x. Besides, the ER strategy obtains a 59x speedup on Embed Time, which saves 98.3% inference time in *embedding table query*. Different from ER, MM merges the softmax into the results of constant attention, demonstrating a zero cost in softmax, and a slight time decrease in Linear Time. We also see that MERGE achieves a higher speedup on T5 than GPT-2, which might be because every self-attention module of T5 follows with a cross-attention module possessing a much higher time proportion on linear computations and softmax.

Under the same settings of Table 1, we also record the communication cost between the client and the server, shown in Table 1. In general, Table 1 reveals a similar phenomenon to Table 1. We see that existing methods reduce the communication cost slightly (less than 2% in GPT-2), while our method reduces 62% communication cost, with 98% and 25% on *embedding table query* and *linear operation*, respectively.

Performance Evaluation

In addition to inference speed, we also focus on the inference performance between *MERGE* and other MPC frameworks. As shown in Table 2, our method achieves comparable results to MPCformer and demonstrates strong scores across multiple metrics. For instance, the BERTscore of *MERGE*

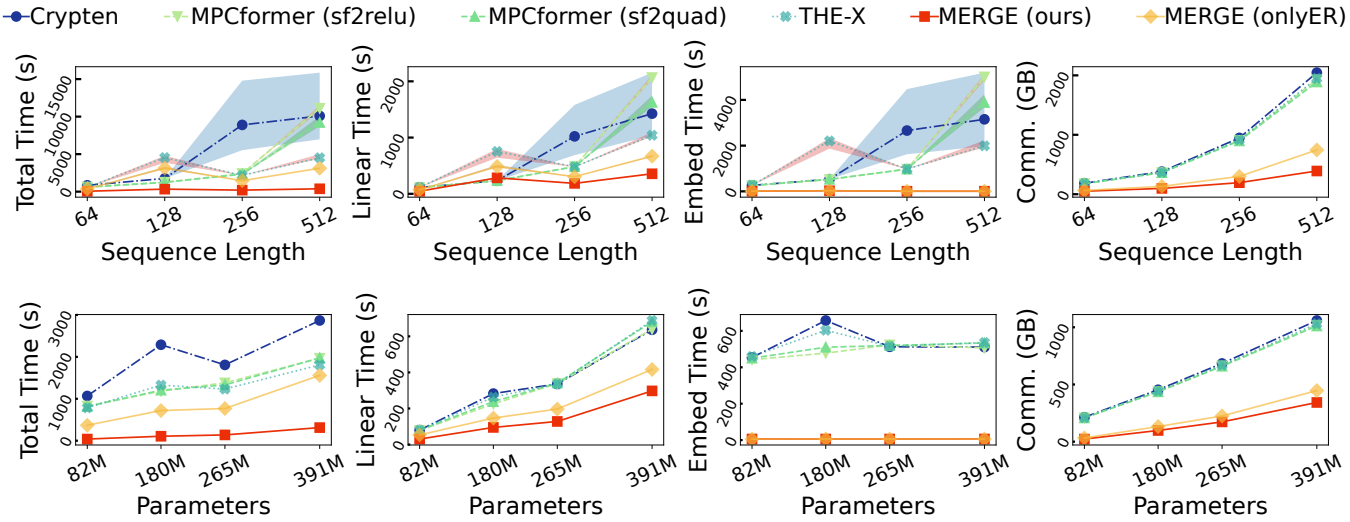


Figure 4: The inference time and communication cost varying generated max sequence lengths and model parameters.

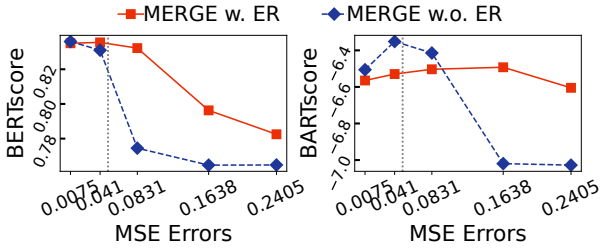


Figure 5: Robustness experiments varying MSE error.

is lower than MPCformer with ReLU approximation (MPCformer (sf2relu)) by only 0.01, 0.017, and 0.001 in MultiWoz, CommonGen, and DailyDialog, respectively. Besides, Table 2 indicates that some acceleration methods designed for NLU models are not suitable to text generation models, i.e. they suffer from the convergence problem during training. For instance, THE-X replaces all *layer normalization* operations to the approximate normalization, which we observed will lead to the **out of time (OOT)** issue. Similarly, the MPCformer that replaces the softmax function with quadratic functions (MPCformer (sf2quad)) faces the same problem, though we train it with an elaborate layer-wise knowledge distillation.

Analysis

Model Size and Sequence Lengths. In this section, we dive to explore the effectiveness of our *MERGE* method under longer sequence lengths and larger model parameters. For sequence length, we set it from 64 to 512 and record the average score as well as the minimum and maximum score for each point. Illustrated by Fig. 4 (the upper row), we see that the inference time cost, as well as the communication cost, decreases with the improvements in sequence length. In detail, our *MERGE* method can obtain a 26.5x speedup to the vanilla model and 11.8x to the state-of-the-art model THE-X under sequence length 512, and reduce almost 80% communication cost. Besides, our embedding resending (ER) strategy can obtain a **constant** embedding inference time, which is because ER bypasses the *embedding table query*, and thus its embedding time is only related to the generation prefix of samples.

For model parameters, we also evaluate *MERGE* under different model sizes from 82M to 391M and set the sequence length to 128. The parameter experiments in Fig. 4 (the below row) demonstrates that there are no significant improvements in speedup while the model size increases, but our *MERGE* still obtains a significant speedup (~10x) to existing methods. Besides, our method exhibits a conspicuous positive correlation with the model parameter size in terms of the gap between our method and the baselines, particularly in linear time and the communication cost, which demonstrates the effectiveness of our *MERGE*.

Robustness of Word Embedding. Illustrated in Fig. 5, we add random noise on the embedding of Transformer models and evaluate the decrease of text generation for different generation strategies. In concrete, Fig. 5 demonstrates that there exists an abrupt decline with MSE error 0.08 for vanilla auto-regressive generation, while our method can resist the decreasing of generation quality.

Conclusion

To address the problem of private text generation, we propose *MERGE*, a novel framework to accelerate the inference procedure of Transformer-based language models. *MERGE* consists of two optimizations, embedding resending and the merge module. The former speeds up the auto-regressive generation by bypassing the embedding table query of Transformer models, and the latter optimizes and merges the computation of the existing Transformer modules. Extensive experiments demonstrate the superiority of *MERGE* both in inference speed and generation quality. In the future, we plan to design a fast and plug-and-play MPC framework for existing language models.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments and suggestions. This work was supported in part by the National Key R&D Program of China (2021YFB1715600), National Natural Science Foundation of China (U22B2019, 62272372, 62272379).

References

Agarap, A. F. 2018. Deep Learning using Rectified Linear Units (ReLU). *CoRR*, abs/1803.08375.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Banerjee, S.; and Lavie, A. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In Goldstein, J.; Lavie, A.; Lin, C.; and Voss, C. R., eds., *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, 65–72. Association for Computational Linguistics.

Chen, T.; Bao, H.; Huang, S.; Dong, L.; Jiao, B.; Jiang, D.; Zhou, H.; Li, J.; and Wei, F. 2022. THE-X: Privacy-Preserving Transformer Inference with Homomorphic Encryption. In Muresan, S.; Nakov, P.; and Villavicencio, A., eds., *Findings of ACL 2022, Dublin, Ireland, May 22-27, 2022*, 3510–3520. Association for Computational Linguistics.

Dusek, O.; Novikova, J.; and Rieser, V. 2018. Findings of the E2E NLG Challenge. In Krahmer, E.; Gatt, A.; and Goudbeek, M., eds., *Proceedings of the 11th International Conference on Natural Language Generation, Tilburg University, The Netherlands, November 5-8, 2018*, 322–328. Association for Computational Linguistics.

Eric, M.; Goel, R.; Paul, S.; Sethi, A.; Agarwal, S.; Gao, S.; Kumar, A.; Goyal, A. K.; Ku, P.; and Hakkani-Tür, D. 2020. MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines. In Calzolari, N.; Béchet, F.; Blache, P.; Choukri, K.; Cieri, C.; Declerck, T.; Goggi, S.; Isahara, H.; Maegaard, B.; Mariani, J.; Mazo, H.; Moreno, A.; Odijk, J.; and Piperidis, S., eds., *LREC 2020, Marseille, France, May 11-16, 2020*, 422–428. European Language Resources Association.

Hao, M.; Li, H.; Chen, H.; Xing, P.; Xu, G.; and Zhang, T. 2022. Iron: Private Inference on Transformers. In *NeuIPS*.

Hassid, M.; Peng, H.; Rotem, D.; Kasai, J.; Montero, I.; Smith, N. A.; and Schwartz, R. 2022. How Much Does Attention Actually Attend? Questioning the Importance of Attention in Pretrained Transformers. In Goldberg, Y.; Kozareva, Z.; and Zhang, Y., eds., *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, 1403–1416. Association for Computational Linguistics.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Knott, B.; Venkataraman, S.; Hannun, A. Y.; Sengupta, S.; Ibrahim, M.; and van der Maaten, L. 2021. CrypTen: Secure Multi-Party Computation Meets Machine Learning. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *NeurIPS 2021,*, 4961–4973.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In Jurafsky, D.; Chai, J.; Schluter, N.; and Tetreault, J. R., eds., *ACL 2020, Online, July 5-10, 2020*, 7871–7880. Association for Computational Linguistics.

Li, D.; Shao, R.; Wang, H.; Guo, H.; Xing, E. P.; and Zhang, H. 2022. MPCFormer: fast, performant and private Transformer inference with MPC. *CoRR*, abs/2211.01452.

Li, Y.; Su, H.; Shen, X.; Li, W.; Cao, Z.; and Niu, S. 2017. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. In Kondrak, G.; and Watanabe, T., eds., *IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*, 986–995. Asian Federation of Natural Language Processing.

Lin, B. Y.; Zhou, W.; Shen, M.; Zhou, P.; Bhagavatula, C.; Choi, Y.; and Ren, X. 2020. CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning. In Cohn, T.; He, Y.; and Liu, Y., eds., *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, 1823–1840. Association for Computational Linguistics.

Lin, C.; and Och, F. J. 2004. Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics. In Scott, D.; Daelemans, W.; and Walker, M. A., eds., *ACL 2004, Barcelona, Spain*, 605–612. ACL.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Annual Meeting of the Association for Computational Linguistics*.

Mishra, P.; Lehmkuhl, R.; Srinivasan, A.; Zheng, W.; and Popa, R. A. 2020. Delphi: A Cryptographic Inference Service for Neural Networks. In Capkun, S.; and Roesner, F., eds., *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, 2505–2522. USENIX Association.

Popovic, M. 2017. chrF++: words helping character n-grams. In Bojar, O.; Buck, C.; Chatterjee, R.; Federmann, C.; Graham, Y.; Haddow, B.; Huck, M.; Jimeno-Yepes, A.; Koehn, P.; and Kreutzer, J., eds., *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, 612–618. Association for Computational Linguistics.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, 21: 140:1–140:67.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to pre-

vent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *NeuIPS 2017, December 4-9, 2017, Long Beach, CA, USA*, 5998–6008.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2020. Transformers: State-of-the-Art Natural Language Processing. In Liu, Q.; and Schlangen, D., eds., *EMNLP 2020 - Demos, Online,*, 38–45. Association for Computational Linguistics.

Yuan, W.; Neubig, G.; and Liu, P. 2021. BARTScore: Evaluating Generated Text as Text Generation. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *NeurIPS 2021, December 6-14, 2021, virtual*, 27263–27277.

Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K. Q.; and Artzi, Y. 2020. BERTScore: Evaluating Text Generation with BERT. In *ICLR 2020,*. OpenReview.net.