A Fast Attention Network for Joint Intent Detection and Slot Filling on Edge Devices

Liang Huang, Senior Member, IEEE, Senjie Liang, Feiyang Ye, and Nan Gao

Abstract—Intent detection and slot filling are two main tasks in natural language understanding and play an essential role in task-oriented dialogue systems. The joint learning of both tasks can improve inference accuracy and is popular in recent works. However, most joint models ignore the inference latency and cannot meet the need to deploy dialogue systems at the edge. In this article, we propose a fast attention network (FAN) for joint intent detection and slot filling tasks, guaranteeing both accuracy and latency. Specifically, we introduce a clean and parameter-refined attention module to enhance the information exchange between intent and slot, improving semantic accuracy by more than 2%. The FAN can be implemented on different encoders and delivers more accurate models at every speed level. Our experiments on the Jetson Nano platform show that the FAN inferences 15 utterances per second with a small accuracy drop, showing its effectiveness and efficiency on edge devices.

Impact Statement—Dialogue systems at the edge are an emerging technology in real-time interactive applications. They improve the user experience with low latency and secure privacy without transferring personal data to the cloud servers. However, it is challenging to guarantee inference accuracy and low latency on hardware-constrained devices with limited computation, memory storage, and energy resources. The neural network models we introduce in this paper overcome these limitations. With a significant increase in semantic accuracy by more than 2% after adopting our algorithms, the technology reduces the inference latency to less than 100ms. From this viewpoint, our approaches accelerate the boosting of secure personal assistants to end-users.

Index Terms—Attention network, edge devices, inference latency, intent detection, natural language understanding (NLU).

I. INTRODUCTION

DIALOUGE systems at the edge have tremendous potential to power secure and real-time interactive applications [1], i.e., augmented/virtual reality, autonomous vehicles, and robots.

They store personal data at the edge to get void of privacy leakage, as opposed to the cloud-based commercial services, i.e., Google Assistant and Amazon Alexa. Dialogue systems such as task-oriented dialogue systems and intelligent personal assistants have been deployed near the user at different edge platforms, i.e., Raspberry Pi [2], Jetson Nano [3], and smartphones. The major challenge is guaranteeing real-time user experience on hardware-constrained devices with limited computation, memory storage, and energy resources.

Natural language understanding (NLU) [4] is crucial for understanding user input for effective human–computer interaction to establish an innovative and efficient human–machine dialogue system. NLU generally includes both intent detection and slot filling [5]. Intent detection focuses on automatically identifying the intent of user utterances, which can be considered a classification problem. Slot filling extracts semantic constituents from the natural language utterances to provide essential information for the system to take the following action, which can be considered a sequence label problem. Intent detection and slot filling were performed separately in earlier studies in a pipeline approach [6], which first classifies the intent of an utterance, and then, uses the extra intent information to aid slot filling. Commonly used approaches for intent detection are support vector machine (SVM) [7] and recurrent neural network (RNN) [8], and for slot filling are conditional random field (CRF) [9] and RNN. However, an incorrect intent prediction will possibly mislead the successive slot filling in the pipeline approaches.

The trend is to develop a joint model for both intent detection and slot filling tasks to avoid error propagation in the pipeline approaches. Some joint models apply a joint loss function to connect the two tasks [10], [11], [12], [13]. Some models take advantage of the close relationship between two tasks and use some structures, i.e., slot-gate [14], stack-propagation [15], and co-interactive [16], to model the relationship between both tasks explicitly. Modeling the relationship between the two tasks enables these models to achieve significant performance improvements, and thus, demonstrates the effectiveness of this approach. Recently, pretrained language models, such as bidirectional encoder representation from transformers (BERT) [17], are widely used in various natural language processing (NLP) tasks. BERT has contextual solid representation capabilities. Some works [18], [19] have used it in joint intent detection and slot filling models and gotten a considerable performance boost, bringing the accuracy of joint model predictions to a new level.

Manuscript received 14 May 2022; revised 9 September 2022, 27 February 2023, and 14 June 2023; accepted 19 August 2023. Date of publication 28 August 2023; date of current version 12 February 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62072410, in part by the Fundamental Research Funds for the Provincial Universities of Zhejiang under Grant RF-B2022002, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LGF22F020014, and in part by the National Key Research and Development Program of China under Grant 2020YFB1707700. This paper was recommended for publication by Associate Editor Koduvayur P. Subbalakshmi upon evaluation of the reviewers’ comments. (Corresponding author: Nan Gao.)

Liang Huang and Nan Gao are with the College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310058, China (e-mail: lianghuang@zjut.edu.cn; gaonan@zjut.edu.cn). Senjie Liang and Feiyang Ye are with the College of Information Engineering, Zhejiang University of Technology, Hangzhou 310058, China (e-mail: senjieliang@zjut.edu.cn; feiyangye@zjut.edu.cn).

Digital Object Identifier 10.1109/TAI.2023.3309272

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
However, most of the previous works focus on improving model prediction accuracy, and a few works consider inference latency. SlotRefine [20] uses a two-pass iteration mechanism to replace CRF for slot decoding while considerably speeding up the decoding. From our perspective, although large pretrained language models such as BERT bring significant performance gains, these models also have increasing demands on computational resources and memory [21]. Due to high latency or memory overflow, these models are challenging to deploy directly in certain computing resource-constrained conditions, such as on edge devices [3]. Edge computing is a real need due to the response latency and privacy issues associated with computing from the cloud [22]. Therefore, different model compression methods are proposed in the literature to reduce the model parameters and speed up the inference time, i.e., knowledge distillation [23], [24], [25], [26], quantization [27], and structured pruning [28]. To achieve a better balance between speed and accuracy on edge devices, we argue that the following problems should be addressed.

1) Although large-scale pretraining language models such as BERT have greatly improved the inference accuracy, their massive number of parameters makes the inference time expensive.
2) Although prior works such as SlotRefine speed up the decoding, they do not utilize pretrained knowledge to enhance accuracy.
3) Although co-interactive [16] has achieved significant accuracy gains, their models are complex and have many parameters.

Therefore, they incur high latency and memory usage.

In this article, we propose a fast attention network (FAN) for joint intent detection and slot filling that aims to speed up the model inference without compromising the accuracy. The FAN is generic for various pretrained language models. In this article, we used five different pretrained language models: BERT, a lite BERT (ALBERT) [29], MiniLM [30], a distilled version of BERT (DistilBERT) [23], and TinyBERT [24], as an encoder to learn the representation of utterances. In our FAN, we designed a parameter-refined attention module to carry out the two-way information interaction between intentions and slots. The model’s performance is improved for the intent detection and slot filling tasks without significantly impacting the model speed and parameters. The attention module consists of a label attention layer and a multilayer self-attention layer, where the label attention layer integrates the result information of the two tasks into the representation of the utterance. The multilayer self-attention layer bidirectionally shares and exchanges information for intent detection and slot filling to promote each other, instead of only considering the single flow of information from intents to slots as in previous work. In summary, the contributions of this article are as follows.

1) We propose a novel model framework FAN based on a parameter-refined attention module to jointly model intent detection and slot filling tasks. The FAN only uses a label attention layer and a multilayer self-attention layer for information interaction between intent and slot. Numerical experiments show that such a clean scheme achieves state-of-the-art inference accuracy on different datasets.
2) We implement the FAN on various pretrained language models and experimentally show that the FAN delivers more accurate models at every speed level. When TinyBERT is the encoder, the FAN improves the semantic accuracy by more than 2.0%.
3) We deploy the FAN on popular edge devices. It infers 15 utterances per second on the Jetson Nano platform while guaranteeing comparable accuracy.

The rest of this article is organized as follows. Section II reviews the related works on joint model and model compression. Section III gives a detailed description of our model. Section IV presents experimental results and analysis. Finally, Section V concludes this article.

II. RELATED WORK

A. Joint Intent Detection and Slot Filling

Recently, some joint models have overcome the error propagation caused by the pipelined approaches. Goo et al. [14] proposed a slot-gated mechanism to pass the intent information into the slot filling task for interactions between intent and slots. Li et al. [31] proposed a new gating mechanism based on self-attention and multilayer perceptron (MLP) to transfer intent information to the slot filling task. Qin et al. [15] adopted a stack-propagation framework for interactions between intent and slots, which can directly use the intent result information as input for slot filling. E et al. [32] proposed a slot filling and intent detection (SF-ID) network in the middle of the long short-term memory (LSTM)-based encoder and the CRF-based decoder. The SF subnet applies intent information to the slot filling task, while the ID subnet uses slot information in the intent detection task. Chen et al. [18] used BERT as the encoder in the joint model for the first time, bringing the accuracy of the joint model to a new level. Wu et al. [20] proposed a two-pass refine mechanism to solve the problem of the uncoordinated slots and to speed up model inference by replacing CRF. In the first pass, the model is used to predict the “B” label, and in the second pass, the predicted “B” label information is sent back to the model to predict the “I” label. Qin et al. [16] proposed a co-interactive transformer based on the transformer encoder for bidirectional information exchange between intent and slots. Wei et al. [33] proposed a wheel-graph structure based on the graph attention network (GAT) to use the correlation between intent and slots.

B. Model Compression

Pretrained language models such as BERT, XLNet [34], and RoBERTa [35] were widely used in NLP tasks and achieved significant performance improvements. For example, BERT has powerful semantic representation capabilities and can be used for various downstream tasks through simple fine-tuning. For intent detection and slot filling, the use of BERT has also brought about a significant effect improvement in joint models [18], [19]. However, although these models have brought significant improvement, these models usually have hundreds of millions of parameters, which consumes a lot of computational resources and experiences a long inference time in practical applications. Therefore, some works tried to compress...
the pretrained model. ALBERT [29] incorporates embedding factorization and cross-layer parameter sharing to reduce model parameters. Since ALBERT does not reduce the hidden size or layers of the transformer block, it still has a large amount of computation and is time-consuming in the prediction process. DistilBERT [23] performs distillation at the pretraining stage on a large-scale corpus. DistilBERT, which consists of six bidirectional transformer encoder layers, is 40% smaller and 60% faster than BERT, but it still retains 97% of the language understanding capability. TinyBERT [24] uses a new two-stage learning framework, which performs transformer distillation at both the pretraining and task-specific learning stages. This two-stage learning method reduces the size of the BERT model by 87%. The TinyBERT with four transformer encoder layers is 7.5x smaller and 9.4x faster than BERT and still retains more than 96.8% of the performance of BERT on the GLUE [36] benchmark. MobileBERT [26] is a thin version of BERT_LARGE. The knowledge in MobileBERT is transferred from a especially designed teacher model, which is an inverted-bottleneck incorporated BERT_LARGE model. MiniLM [30] is another compressed model of the large transformer-based model via a deep self-attention distillation and retains 99% accuracy.

C. Edge Intelligence

A computational gap has arisen between deep learning algorithms with high computational demands and edge devices with low computational power. Many high-precision deep learning algorithms cannot be deployed on edge devices. Therefore, many approaches have arisen to address this computational gap [22]. Pandelea et al. [3] combined a large transformer as a feature extractor with a simple classifier, deployed it on Jetson Nano and two smartphones, and optimized latency and performance using dimensionality reduction and pretraining. Xu et al. [2] proposed an edge-based caching framework for voice assistant systems called CHA on three edge devices, Raspberry Pi, Intel Fog Reference Design, and Jetson AGX Xavier.

III. METHOD

A. Problem Setup

Consider a user’s utterance with T tokens, \( \{x_i| i \in T, T = \{1, \ldots, T\}\} \), where \( x_i \) represents the \( i \)th token. The main goal is to jointly predict the utterance’s unique intent label \( y^I \) and a set of task-specific slot labels \( \{y^S_i| i \in T\} \) for all tokens \( \{x_i| i \in T\} \) on a one-to-one basis. We represent the slot labels in the BIO form [37], denoting whether a token is the beginning of a slot (“B”), inside a slot (“I”), or outside any slot (“O”). For example, the utterance “find fish story” from the Snips dataset [38] is labeled the intent, “Search Screening Event,” and its three tokens, i.e., “find,” “fish,” and “story” are labeled with different slots, “O,” “B-movie_name,” and “I-movie_name,” respectively.

In the following subsections, we introduce the details of the proposed FAN framework, as illustrated in Fig. 1. Specifically, it comprises an encoder module, an attention module, and a decoder module. The encoder module extracts the semantic representation vector \( H \) from the input utterance \( x \). Then, we propose a simple and effective attention module to purify the intent and slot information from \( H \). Based on this, the decoder module predicts the corresponding intent label \( y^I \) and the slot labels \( \{y^S_i\} \).

B. Encoder Module

For an utterance \( x \), we insert a special token ([CLS]) at the beginning of the utterance as the first token, denoted as \( x_0 \) and a special token ([SEP]) at the end of utterance as the final token, denoted as \( x_T \). Given the input utterance consisting of \( T + 1 \) tokens \( x = (x_0, x_1, \ldots, x_T) \) into the pretrained language.
model-based encoder, and then, it produces the semantic representation of tokens, \( H \in \mathbb{R}^{(T+1) \times d} \), where \( d \) represents the hidden dimension of the pretrained language model.

We choose BERT, ALBERT, MiniLM, DistilBERT, and TinyBERT as our encoders. Among them, BERT can achieve the best performance. DistilBERT and TinyBERT can significantly improve the model’s speed without losing too much accuracy and alleviate the problem of excessive BERT model parameters to a certain extent. DistilBERT and TinyBERT are all knowledge distillation versions of BERT, which have a similar structure to BERT.

The input representation concatenates the WordPiece embedding [39], segment embedding, and position embedding for BERT and TinyBERT. Since DistilBERT does not have NSP pretraining tasks [23], its segment embedding is absent. However, segment embedding has no discrimination for intent detection and slot filling tasks.

C. Attention Module

As shown in Fig. 1, the attention module comprises a label attention layer and a multihead self-attention layer. We input the semantic representation of tokens \( H \) into label attention layer and obtain the resulted representation vectors \( H^A \). We feed the output of label attention layer \( H^A \) into the multihead self-attention layer and obtain the output, \( H^M \). Bypassing \( H^M \) through a residual connection [40], a layer normalization [41], and a two-layer fully connected feed-forward network (FFN), we obtain the output of the attention module, \( H' \).

1) Label Attention Layer: The architecture of the label attention layer is illustrated in Fig. 1. We perform the label attention layer to integrate the predicted label information into the representation of tokens \( H \).

Specifically, we first obtain the weight of the fully connected FFN in the decoder module, which is denoted by \( W^L \in \mathbb{R}^{d \times N^L} \), \( N^L \) represents the number of intent labels, and \( W^S \in \mathbb{R}^{d \times N^S} \), where \( N^S \) represents the number of slot label. Then, we concatenate the two weights \( W^A \) and \( W^S \) together as \( W = [W^A, W^S] \).

In the first training step, \( W \) is generated by Kaiming uniform distribution initialization [42].

In practice, we multiply the representation vector \( H \) with the weight of the decoder \( W \) and map it to a probability distribution through the softmax layer:

\[
\alpha = \text{softmax}(HW) \quad (1)
\]

where \( \alpha \) is the output of the softmax layer. Then, we integrate the information contained in \( \alpha \) into the representation of tokens \( H^A = (H + \alpha (W^L)^T) W^A \) \( (2) \)

where \( H^A \) is the representation of tokens after the label attention layer. \( W^A \) is the trainable parameters of linear projectors.

2) Multihead Self-Attention Layer: Inspired by multihead self-attention in machine translation, we exploit multihead self-attention to model a bidirectional connection between intent and slots and capture information about the close relationship between two tasks. The architecture of the multihead self-attention layer is illustrated in Fig. 1.

Then multihead self-attention maps the matrix \( H^A \) to query, key, value matrices \( h \) times by different linear projections

\[
Q_i = H^A W^Q_i \quad (3)
\]

\[
K_i = H^A W^K_i \quad (4)
\]

\[
V_i = H^A W^V_i, \quad i = 1, \ldots, h \quad (5)
\]

where \( W^Q_i \in \mathbb{R}^{d \times \frac{d}{h}}, W^K_i \in \mathbb{R}^{d \times \frac{d}{h}}, \) and \( W^V \in \mathbb{R}^{d \times \frac{d}{h}} \) are the trainable parameters of linear projectors. Each of the projected query matrices \( Q_i \), key matrices \( K_i \), and value matrices \( V_i \) perform the scaled dot-product attention [43] in parallel.

\[
h_i = \text{Attention}(Q_i, K_i, V_i). \quad (6)
\]

The scaled dot-product attention can be computed as follows:

\[
\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V. \quad (7)
\]

These attentions are concatenated and projected again to obtain a new representation \( H^M \) as

\[
H^M = \text{concat}(h_1, \ldots, h_h) W^O \quad (8)
\]

where \( W^O \in \mathbb{R}^{d \times d} \) is the trainable parameter of linear projector. Here \( H^M \) is the output of the multihead self-attention layer.

As in the vanilla transformer [43], we add a residual connection and a layer normalization after the multihead self-attention layer.

\[
H^L = \text{LayerNorm}(H + H^M). \quad (9)
\]

Finally, we add a two-layer fully connected FFN after the residual connection and the layer normalization. The output of the attention module can be expressed as

\[
[H^I, H^S] = \max (0, H^L W^I + b_1) W^2 + b_2 \quad (10)
\]

where \( H^I \in \mathbb{R}^{1 \times d} \) represents the intent information, \( H^S \in \mathbb{R}^{T \times d} \) represents the slot information for \( T \) input tokens, and \( W^I, W^2, b_1, \) and \( b_2 \) are trainable parameters.

D. Decoder Module

The decoder module decodes the intent label \( y^I \) and the slot label \( y^S \) from \( H^I \) and \( H^S \) via two different fully connected FFNs, respectively. To predict the intent label, we have

\[
y^I = \text{softmax}(H^I W^I + b^I) \quad (11)
\]

where \( W^I \) and \( b^I \) are trainable parameters. To predict the slot labels \( \{y^S_i\} \), we have

\[
y^S_i = \text{softmax}(h^S_i W^S + b^S), \quad i \in T \quad (12)
\]

where \( h^S_i \) is the ith element of \( H^S \), and \( W^S \) and \( b^S \) are trainable parameters.

E. Joint Training

To model two tasks simultaneously, we employ joint optimization to update the parameters of the model. Given the input utterance \( x \), the conditional probability of the understanding
result (intent detection and slot filling) is as follows:

$$p\left( y^I, y^S \mid x \right) = p\left( y^I \mid x \right) \prod_{n=1}^{N} p\left( y^S_n \mid x \right).$$ (13)

When training the model, our loss function can be divided into two parts: intent and slot. The intent loss function can be expressed as follows:

$$L_{ID} = - \sum_{i=1}^{N^I} \tilde{y}^{i,I} \log \left( y^{i,I} \right).$$ (14)

Similarly, the slot loss function can be expressed as

$$L_{SF} = - \sum_{j=1}^{T} \sum_{i=1}^{N^S} \tilde{y}^{i,S}_{j} \log \left( y^{i,S}_{j} \right)$$ (15)

where \( \tilde{y}^{i,I} \) and \( \tilde{y}^{i,S}_{j} \) represent the target intent label and target slot label separately. In order to jointly train the intent detection and slot filling tasks, the final loss function of two tasks is formulated as

$$L = \lambda L_{ID} + (1 - \lambda) L_{SF}$$ (16)

where the hyperparameter \( \lambda \) is a mixture weight, as \( 0 < \lambda < 1 \).

The training objective is to minimize the loss function \( L \) of two tasks to maximize the conditional probability \( p(y^I, y^S \mid x) \).

IV. EXPERIMENTS

A. Datasets

We evaluate the proposed FAN framework on two public benchmark datasets, air-line travel information system (ATIS) [44] and Snips [38]. The ATIS dataset contains audio recordings of people making flight reservations, labeled by 21 intents and 120 slots. The Snips dataset was collected from the Snips personal voice assistant. The statistics of both datasets are summarized in Table I.

|             | ATIS       | Snips     |
|-------------|------------|-----------|
| Training set size | 4478       | 13084     |
| Development set size | 500        | 700       |
| Testing set size    | 893        | 700       |
| Num Intents        | 21         | 7         |
| Num Slots          | 120        | 72        |
| Domain              | air travel | personal assist. |

B. Metrics

We evaluate the FAN under different accuracy and latency performance metrics. Following the conventions of previous work [14], we choose the accuracy for intent detection and the F1 score for slot filling. Furthermore, the sentence-level semantic frame accuracy represents the proportion of utterances that both intent detection and slot filling tasks are predicted correctly. We evaluate the inference latency of different models on multiple platforms, i.e., RTX 3090, Jetson TX2, and Jetson Nano. We feed all samples in the test set one by one into the model and calculate the average latency for predicting one utterance.

C. Training Details

We evaluate three FAN models with different encoders, i.e., BERT, DistilBERT, and TinyBERT, and denote them as BERT-FAN, DistilBERT-FAN, and TinyBERT-FAN, respectively. As shown in the Table II, BERT has 12 transformer blocks and 768 hidden states, DistilBERT has six transformer blocks and 768 hidden states, and TinyBERT has four transformer blocks and 312 hidden states. We search for the optimal hyper-parameter \( \lambda \) from 0.1 to 0.9. The maximum length of utterances is 50. The training batch size is 32. The dropout ratio is 0.1. We use Adam [45] to optimize FAN parameters with a learning rate of 5e-5. Specifically, TinyBERT-FAN only has 15.8 M parameters and consumes 485 s training time.

D. Benchmarks

We compare FAN with the following six benchmarks.

1) **Slot-gated full attention**: Goo et al. [14] utilized a slot-gated mechanism as a particular gate function in bidirectional LSTM (BiLSTM) to improve slot filling using the learned intent context vector.

2) **SF-ID network**: E et al. [32] proposed an SF-ID network that consists of an SF subnet and an ID subnet after the BiLSTM encoder. The SF subnet applies intent information to the slot filling task, while the ID subnet uses slot information in the intent detection task. SF-ID network builds a bidirectional connection between intent detection and slot filling to help them promote each other mutually.

3) **Stack-propagation**: Qin et al. [15] adopted a joint BiLSTM-based model with stack-propagation, which transmits intent information to slot filling to improve the performance of the slot filling task.

4) **SlotRefine**: Wu et al. [20] proposed a nonautoregressive transformer-based model and designed a two-pass iteration mechanism to handle the problem of the uncoordinated slots and speed up the decoding in slot filling.

5) **Co-interactive**: Qin et al. [16] used BiLSTM as the encoder and proposed a co-interactive transformer to exchange the mutual information of intent and slot. The co-interactive transformer uses two different label attentions for the representation vectors of intents and slots, respectively, and uses two self-attention to exchange information between intent and slots. They concatenated two co-interactive transformers to further enhance the accuracy.

---

**TABLE I**

|                      | ATIS       | Snips     |
|----------------------|------------|-----------|
| Training set size    | 4478       | 13084     |
| Development set size | 500        | 700       |
| Testing set size     | 893        | 700       |
| Num Intents          | 21         | 7         |
| Num Slots            | 120        | 72        |
| Domain               | air travel | personal assist. |

**TABLE II**

| Encoder   | Transformer Blocks | Hidden States \(d\) | Parameters (M) | Training Time (s) |
|-----------|--------------------|---------------------|----------------|-------------------|
| BERT      | 12                 | 768                 | 116.6          | 1.456             |
| DistilBERT| 6                  | 768                 | 59.9           | 857               |
| TinyBERT  | 4                  | 312                 | 15.8           | 485               |
6) JointBERT: Chen et al. [18] proposed a BERT-based model with an encoder module and a decoder module. JointBERT connects the representation vector of the first token to a single-layer FFN for intent recognition and connects the representation vector of the remaining tokens to another single-layer FFN for slot filling. The model JointBERT-CRF means that CRF is used for slot decoding. If we remove the attention module from BERT-FAN, i.e., \( H^A = H \), it coincides with the scheme of JointBERT-CRF [18]. For better comparison, we further extend JointBERT to the other encoders and denote them as JointDistilBERT and JointTinyBERT in this article.

E. Main Results

In Fig. 2, we compare the semantic accuracy of the FAN to other models with regard to the inference speed. The number of utterances per second for each model is evaluated on the Jetson Nano platform in the 5-W mode. The FAN delivers more accurate models at every speed level on both ATIS and Snips datasets. The attention module improves the semantic accuracy for different encoders, especially the small-size ones. Compared with JointTinyBERT, TinyBERT-FAN improves semantic accuracy by more than 2.0%. TinyBERT-FAN achieves higher accuracy and inferences five more utterances per second than SlotRefine. DistilBERT-FAN balances the inference accuracy and speed.

In Table III, we illustrate the evaluated models’ detailed accuracy and latency performance. Based on the same BERT model, BERT-FAN inferences 2.3x faster and achieves comparable or even better performance than the state-of-the-art JointBERT-CRF. It verifies the effectiveness of the proposed attention module in the interaction between intent and slot information. ALBERT-FAN achieves the greatest intent accuracy on the Snips dataset and the same speedup as BERT-FAN. MiniLM-FAN achieves similar sentiment accuracy as DistilBERT-FAN. Among those FAN models, DistilBERT-FAN and TinyBERT-FAN are the only two models whose inference latency is less than 100 ms.

F. Analysis

1) Ablation Experiments: In Table IV, we conduct ablation experiments on each component of the attention module based on TinyBERT-FAN, DistilBERT-FAN, and BERT-FAN. We remove the label attention layer by feeding the output of the encoder directly into the multihead self-attention layer, i.e., \( H^A = H \). We remove the multihead self-attention layer by feeding the output of the label attention layer directly into the decoder, i.e., \( H^M = H^A \). We remove the two-layer fully connected feed-forward network by feeding the output of the residual connection and the layer normalization directly into the decoder module, i.e., \( [H^I, H^S] = H^L \).

Removing the multihead self-attention layer from TinyBERT-FAN results in a significant decrease in semantic accuracy, with a drop of 1.9% on the ATIS dataset and 2.4% on the Snips dataset. This highlights the critical role of the multihead self-attention layer in the FAN.

Removing the label attention layer from DistilBERT-FAN increases the intent accuracy by 0.1% but decreases the F1 score of slot filling by 0.4% on the Snips dataset. Interestingly, its semantic accuracy drops by 0.8%. In the case of TinyBERT-FAN, removing the label attention layer results in a decrease of 0.7% in the semantic accuracy on the ATIS dataset and 1.2% on the Snips dataset.

Similarly, removing the two-layer FFN from TinyBERT-FAN leads to an increase in intent detection accuracy but a decrease in slot filling accuracy, resulting in a decrease of semantic accuracy by 0.5% and 0.8% on the ATIS dataset and Snips dataset, respectively. This highlights the importance of the two-layer FFN in balancing the intent detection and slot filling tasks. It enables a reasonable tradeoff that improves semantic accuracy while introducing a latency of 2.8 ms for inference.

While each component contributes no more than a 0.5% increase in sentiment accuracy when BERT is used as the encoder, the overall performance of the FAN is still slightly better than that of JointBERT, as observed in Table III. This suggests that
the combination of these components in the FAN effectively balances the intent detection and slot filling tasks, leading to improved performance compared to the baseline model.

Considering the inference speed, each component of the FAN brings a slight latency rise for all three evaluated models. On average, the label attention layer generates a latency of 1.2 ms, the multiheaded self-attention layer generates a latency of 4.1 ms, and the two-layer FFN generates a latency of 3.2 ms. The summed increment of latency introduced by all three components is within 10% of the end-to-end inference latency. Hence, the FAN is a fast framework that balances the inference accuracy and latency well.

2) Multihead Self-Attention: As shown in Fig. 3, we evaluate the FAN with different number of attention heads, $h = \ldots$. 

| Model                      | Intent (Acc) | ATIS Slot (F1) | Sent (Acc) | Intent (Acc) | Snips Slot (F1) | Sent (Acc) | Latency (ms) | Speedup |
|-----------------------------|--------------|----------------|------------|--------------|-----------------|------------|--------------|---------|
| TinyBERT-FAN                | 97.8         | 96.1           | 88.7       | 98.3         | 97.1            | 91.4       | 89.4         | 66.8    |
| - without label attention layer | 97.6         | 95.2           | 86.9       | 97.7         | 94.9            | 88.2       | 65.7         |         |
| - without multi-head self-attention layer | 97.1         | 95.1           | 85.7       | 98.1         | 94.3            | 87.0       | 62.9         |         |
| - without two-layer FFN      | 97.9         | 95.3           | 87.1       | 98.3         | 94.9            | 88.6       | 64.0         |         |
| DistilBERT-FAN              | 97.8         | 96.1           | 88.2       | 98.0         | 96.5            | 91.9       | 88.2         |         |
| - without label attention layer | 97.4         | 95.9           | 87.7       | 98.1         | 96.1            | 91.1       | 87.0         |         |
| - without multi-head self-attention layer | 97.2         | 95.8           | 87.4       | 97.8         | 96.0            | 90.6       | 84.1         |         |
| - without two-layer FFN      | 97.8         | 95.7           | 87.7       | 98.3         | 96.2            | 91.3       | 85.0         |         |
| BERT-FAN                    | 97.8         | 96.1           | 88.7       | 98.3         | 97.1            | 93.0       | 153.9        |         |
| - without label attention layer | 98.0         | 95.9           | 88.4       | 98.1         | 97.0            | 92.8       | 152.6        |         |
| - without multi-head self-attention layer | 97.4         | 96.1           | 88.2       | 98.2         | 96.9            | 92.7       | 149.6        |         |
| - without two-layer FFN      | 97.7         | 96.0           | 88.6       | 98.3         | 97.1            | 92.9       | 150.3        |         |

The bold values are used to emphasize the best performance metrics achieved across different methods.
Fig. 4. Semantic accuracy of TinyBERT-FAN, DistilBERT-FAN, and BERT-FAN under different values of hyperparameter $\lambda$.

TABLE V

| Model               | ATIS Slot (F1) | ATIS Sent (Acc) | DistilBERT-FAN Slot (F1) | DistilBERT-FAN Sent (Acc) | BERT-FAN Slot (F1) | BERT-FAN Sent (Acc) |
|---------------------|----------------|-----------------|--------------------------|--------------------------|-------------------|-------------------|
| TinyBERT-FAN        | 97.8           | 87.6            | 98.1                     | 95.4                     | 89.4              |
| - only intent       | 97.3           | -               | 97.7                     | -                        | -                 |
| - only slot         | 94.6           | -               | 93.9                     | -                        | -                 |
| DistilBERT-FAN      | 97.8           | 88.2            | 98.0                     | 96.5                     | 91.9              |
| - only intent       | 97.7           | -               | 97.6                     | -                        | -                 |
| - only slot         | 94.7           | -               | 95.6                     | -                        | -                 |
| BERT-FAN            | 97.8           | 88.7            | 98.3                     | 97.1                     | 93.0              |
| - only intent       | 97.4           | -               | 97.5                     | -                        | -                 |
| - only slot         | 94.8           | -               | 95.6                     | -                        | -                 |

The optimal $h$ for TinyBERT, DistilBERT, and BERT models are 12, 12, and 2, respectively, on the ATIS and Snips datasets. As illustrated in Table IV, the multihead self-attention layer plays a key role in the attention module. TinyBERT and DistilBERT tend to choose a greater number of heads than BERT. It explains the observation in Fig. 2 that the FAN achieves a more significant increase in accuracy when TinyBERT and DistilBERT are used.

3) Hyperparameter $\lambda$: In Fig. 4, we evaluated the FAN under different values of the hyperparameter $\lambda$. The optimal value of $\lambda$ depends on the FAN model and the evaluated dataset. Taking TinyBERT-FAN as an example, the optimal $\lambda$ is 0.5 for Snips and 0.7 for ATIS. In this article, we choose $\lambda = 0.5$. The semantic accuracy can be further improved by fine-tuning the hyperparameter $\lambda$.

4) Joint Intent and Slot Training: In Table V, we evaluate how joint intent and slot training benefits both tasks. On the training loss (16), we set the $\lambda = 1$ for intention recognition task only and set $\lambda = 0$ for slot filling only. As shown in Table V, the joint training improves both the intent classification accuracy and the slot filling accuracy for the evaluated FAN models on both datasets. The intent classification accuracy is slightly improved by around 0.5%, while the slot filling task is improved by around 1%. This verifies the effectiveness of the information interaction between intent and slot designed in the FAN.

5) Information Exchange: In Table VI, we compare the FAN with SF-ID and co-interactive based on the TinyBERT encoder. All three schemes enhance the accuracy by exchanging information between intent and slot and are independent of the

{1, 2, 4, 8, 12, 16}. Since TinyBERT has the hidden states $d = 312$, which cannot be split equally into 16 heads, we ignore $h = 16$ for TinyBERT-FAN. The optimal $h$ for TinyBERT, DistilBERT, and BERT models are 12, 12, and 2, respectively, on the ATIS and Snips datasets. As illustrated in Table IV, the multihead self-attention layer plays a key role in the attention module. TinyBERT and DistilBERT tend to choose a greater number of heads than BERT. It explains the observation in Fig. 2 that the FAN achieves a more significant increase in accuracy when TinyBERT and DistilBERT are used.
encoder module. For a fair comparison, we replace BiLSTM in SF-ID and co-interactive by TinyBERT and keep the remaining parts unchanged, namely TinyBERT-SF-ID and TinyBERT-Co-interactive. As shown in Table VI, TinyBERT-FAN achieves comparable or even better performance than TinyBERT-SF-ID and TinyBERT-Co-interactive but requires significantly less latency, close to JointTinyBERT. Hence, the FAN is a lightweight and efficient network suitable for joint intent detection and slot filling on edge devices.

When we look at the detailed network structure, the FAN uses one label attention and one multihead self-attention. Co-interactive concatenates two transformers, and each transformer uses two label attentions and two self-attentions for intent and slot, respectively. SF-ID divides the interaction between intent and slot into two different steps. Both SF-ID and co-interactive use CRF for the slot decoding, which greatly slows the inference speed.

6) Error Analysis: In Fig. 5, we plot the confusion matrix of intent prediction on the test dataset. Only 13 out of 700 utterances are misclassified, among which ten utterances with the intent “Search Screening Event” are mistakenly predicted to be “Search Creative Work.” We demonstrate three error cases in Table VII. First, the utterance “Play the album journeyman” actually means to search and play “journeyman” and is labeled “Search Creative Work.” However, since “journeyman” is an album, the model predicts the intent of “Play Music” based on the keyword “play” [46]. Considering the second utterance “Play the new noise theology ep,” the slot label of “the new noise theology ep” is “object_name.” However, “the new noise theology” is the name of the album. TinyBERT-FAN learns this information with pretrained knowledge from Wikipedia and predicts it as “album.” “king of hearts” in the third utterance is the name of a comedy movie. However, TinyBERT-FAN fails to recognize the movie and mistakenly predicts it as “object_name.”

G. Edge Deployment

In Table VIII, we deploy and evaluate the inference latency of the FAN on the desktop platform NVIDIA GeForce RTX 3090 and popular edge platforms, i.e., Jetson TX2, Jetson Nano,
and Atlas 200 [47]. We choose the default MAX-P ARM mode for Jetson TX2 and test Jetson Nano in both MAX-N and 5-W modes. The 5-W mode limits the Jetson Nano’s power to 5 W and limits the number of CPU cores and CPU and GPU frequencies. We include JointBERT-CRF as a baseline. From Table VIII, TinyBERT-FAN achieves 4.3x speedup on Jetson TX2, 4.9x speedup on Atlas 200, and 5.3x speedup on Jetson Nano in the 5-W mode. The inference accuracy of these deployed models is consistent with the performance in Table III. Hence, both DistilBERT-FAN and TinyBERT-FAN significantly reduce the inference latency to less than 100 ms and are suitable for deploying on different edge devices.

V. CONCLUSION

In this article, we propose a FAN-based on pretrained language models for joint intent detection and slot filling tasks. The experimental results show that the FAN delivers more accurate models at every speed level on two public datasets. The proposed attention module improves the semantic accuracy by more than 2.0% when TinyBERT is the encoder. Moreover, we deploy the FAN on popular edge platforms, which infers 15 utterances per second on the Jecson Nano platform. We conclude that the FAN has experimentally proven its value in industrial practice for deployment at the edge.

For future work, we plan to employ an integer-only encoder, i.e., I-BERT [27], to further reduce the inference latency. In addition, we intend to incorporate external knowledge via graph neural networks to further enhance accuracy.

REFERENCES

[1] E. Bertino and S. Banerjee, “Artificial intelligence at the edge,” 2020, arXiv:2012.05410.
[2] L. Xu, A. Iyengar, and W. Shi, “CHA: A caching framework for home-based voice assistant systems,” in Proc. IEEE/ACM Symp. Edge Comput., 2020, pp. 293–306.
[3] V. Pandelea, E. Ragusa, T. Apicella, P. Gastaldo, and E. Cambria, “Emotion recognition on edge devices: Training and deployment,” Sensors, vol. 21, no. 13, 2021, Art. no. 4496.
[4] S. Young, M. Gašić, B. Thomson, and J. D. Williams, “POMDP-based statistical spoken dialog systems: A review,” Proc. IEEE, vol. 101, no. 5 pp. 1160–1179, May 2013.
[5] G. Tur and R. De Mori, Spoken Language Understanding: Systems for Extracting Semantic Information From Speech, Hoboken, NJ, USA: Wiley, 2011.
[6] M. Jeong and G. G. Lee, “Triangular-chain conditional random fields,” IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 16, no. 7, pp. 1287–1302, Sep. 2008.
[7] P. Haffner, G. Tur, and J. H. Wright, “Optimizing SVMs for complex classification,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., Hong Kong, China, Apr. 6–10, 2003.
[8] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, “Spoken language understanding using long short-term memory neural networks,” in Proc. IEEE Spoken Lang. Technol. Workshop, 2014, pp. 189–194.
[9] C. Raymond and G. Riccardi, “Generative and discriminative algorithms for spoken language understanding,” in Proc. Interspeech, 2007, pp. 1605–1608.
[10] Y.-N. Chen, D. Hakami-Tür, G. Tur, A. Celikyilmaz, J. Guo, and L. Deng, “Syntax or semantics? Knowledge-guided joint semantic frame parsing,” in Proc. IEEE Spoken Lang. Technol. Workshop, 2016, pp. 348–355.
[11] X. Zhang and H. Wang, “A joint model of intent determination and slot filling for spoken language understanding,” in Proc. Int. Joint Conf. Artif. Intell., 2016, pp. 2993–2999.
[12] D. Hakani-Tür et al., “Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM,” in Proc. Interspeech, 2016, pp. 715–719.
[13] B. Liu and I. Lane, “Attention-based recurrent neural network models for joint intent detection and slot filling,” in Proc. Interspeech, 2016, pp. 685–689.
[14] C.-W. Goo et al., “Slot-gated modeling for joint slot filling and intent prediction,” in Proc. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., 2018, pp. 753–757.
[15] L. Qin, W. Che, Y. Li, H. Wen, and T. Liu, “A stack-propagation framework with token-level intent detection for spoken language understanding,” in Proc. Conf. Empirical Methods Natural Lang. Process., Int. Joint Conf. Natural Lang. Process., 2019, pp. 2078–2087.
[16] L. Qin, T. Liu, W. Che, B. Xu, S. Zhao, and T. Liu, “A co-active transformer for joint slot filling and intent detection,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., 2021, pp. 8193–8197.
[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., 2019, pp. 4171–4186.
[18] Q. Chen, Z. Zhao, and W. Wang, “BERT for joint intent classification and slot filling,” 2019, arXiv:1902.10909.
[19] Z. Zhang, Z. Zhang, H. Chen, and Z. Zhang, “A joint learning framework with BERT for spoken language understanding,” IEEE Access, vol. 7, pp. 168849–168858, 2019.
[20] D. Wu, L. Ding, F. Lu, and J. Xie, “SlotRefine: A fast non-autoregressive model for joint intent detection and slot filling,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2020, pp. 1932–1937. [Online]. Available: https://aclanthology.org/2020.emnlp-main.152
[21] O. Khatib and M. Zaharia, “ColBERT: Efficient and effective passage search via contextualized latent interaction over BERT,” in Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2020, pp. 39–48.
[22] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, “Bringing AI to edge: From deep learning’s perspective,” Neurocomputing, vol. 485, pp. 297–320, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523121016428
[23] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, A distilled version of BERT: Smaller, faster, cheaper and lighter,” presented at the 33th NeurIPS, Vancouver, Canada, Dec. 8-14, 2019.
[24] X. Jiao et al., “TinyBERT: Distilling BERT for natural language understanding,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2020, pp. 4163–4174. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.372
[25] W. Liu, P. Zhou, Z. Wang, Z. Zhao, H. Deng, and Q. Ju, “FastBERT: A self-distilling BERT with adaptive inference time,” in Proc. Assoc. Comput. Linguistics, 2020, pp. 6035–6044. [Online]. Available: https://aclanthology.org/2020.acl-main.573
[26] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, “MobileBERT: A compact task-agnostic BERT for resource-limited devices,” in Proc. Assoc. Comput. Linguistics, 2020, pp. 2158–2170. [Online]. Available: https://aclanthology.org/2020.acl-main.195
[27] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, and K. Keutzer, “I-BERT: Integer-only bert quantization,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 5506–5518.
[28] Z. Wang, J. Wohlwend, and T. Lei, “Structured pruning of large language models,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2020, pp. 6151–6162. [Online]. Available: https://aclanthology.org/2020.emnlp-main.496
[29] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite bert for self-supervised learning of language representations,” presented at the 7th Int. Conf. Learn. Representations, Addis Ababa, Ethiopia, Apr. 26-May 1, 2019.
[30] W. Wang, P. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, “MinilM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers,” in Proc. Neural Inf. Process. Syst., 2020, pp. 5776–5788.
[31] C. Li, L. Li, and J. Qi, “A self-attentive model with gate mechanism for spoken language understanding,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2019, pp. 3824–3833. [Online]. Available: https://aclanthology.org/D18-1417
[32] H. E, P. Niu, Z. Chen, and M. Song, “A novel bi-directional interrelated model for joint intent detection and slot filling,” in Proc. Assoc. Comput. Linguistics, 2019, pp. 5467–5471. [Online]. Available: https://aclanthology.org/P19-154
[33] P. Wei, B. Zeng, and W. Liao, “Joint intent detection and slot filling with wheel-graph attention networks,” J. Intell. Fuzzy Syst., vol. 42, pp. 2409–2420, 2021.
[34] Z. Yang et al., “XLNet: Generalized autoregressive pretraining for language understanding,” in Proc. Neural Inf. Process. Syst., vol. 32, 2019.

[35] Y. Liu et al., “RoBERTa: A robustly optimized BERT pretraining approach,” 2019, arXiv:1907.11692.

[36] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, “GLUE: A multi-task benchmark and analysis platform for natural language understanding,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2018, pp. 353–355. [Online]. Available: https://aclanthology.org/W18-5446

[37] L. A. Ramshaw and M. P. Marcus, “Text chunking using transformation-based learning,” in Natural Language Processing Using Very Large Corpora. Cham, Switzerland: Springer, 1999, pp. 157–176.

[38] A. Coucke et al., “Snips voice platform: An embedded spoken language understanding system for private-by-design voice interfaces,” presented at the 35th Int. Conf. Mach. Learn. Privacy Mach. Learn. Artif. Intell. Workshop, Jul. 14–15, 2018.

[39] Y. Wu et al., “Google’s neural machine translation system: Bridging the gap between human and machine translation,” 2016, arXiv:1609.08144.

[40] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

[41] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” presented at 30th NIPS, Barcelona, Spain, Dec. 5–10, 2016.

[42] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 1026–1034.

[43] A. Vaswani et al., “Attention is all you need,” in Proc. Neural Inf. Process. Syst., 2017, pp. 6000–6010.

[44] G. Tur, D. Hakkani-Tür, and L. Heck, “What is left to be understood in ATIS?” in Proc. IEEE Spoken Lang. Technol. Workshop, 2010, pp. 19–24.

[45] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” presented at 3rd Int. Conf. Learn. Representations, San Diego, California, US, May 7-9, 2015.

[46] T.-W. Wu, R. Su, and B. Juang, “A label-aware BERT attention network for zero-shot multi-intent detection in spoken language understanding,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2021, pp. 4884–4896. [Online]. Available: https://aclanthology.org/2021.emnlp-main.399

[47] L. Pomšár, A. Brecko, and I. Zolotová, “Brief overview of edge AI accelerators for energy-constrained edge,” in Proc. IEEE 20th Jubilee World Symp. Appl. Mach. Intell. Inform., 2022, pp. 461–466.

[48] A. Shahroudednejad, “A survey on understanding, visualizations, and explanations of deep neural networks,” 2021, arXiv:2102.01792.

[49] X. Li et al., “Interpretable deep learning: Interpretation, interpretability, trustworthiness, and beyond,” Knowl. Inf. Syst., vol. 64, no. 12, pp. 3197–3234, 2022.

[50] G. Ras, M. van Gerven, and P. Haselager, “Explanation methods in deep learning: Users, values, concerns and challenges,” in Explainable and Interpretable Models in Computer Vision and Machine Learning. Cham, Switzerland: Springer, 2018, pp. 19–36.

[51] L. Cui and Y. Zhang, “Hierarchically-refined label attention network for sequence labeling,” in Proc. Conf. Empirical Methods Natural Lang. Process., Int. Joint Conf. Natural Lang. Process., 2019, pp. 4115–4128. [Online]. Available: https://aclanthology.org/D19-1422