Graph Neural Networks for Contextual ASR With the Tree-Constrained Pointer Generator

Guangzhi Sun, *Member, IEEE*, Chao Zhang, *Member, IEEE*, and Philip C. Woodland, *Fellow, IEEE*

***Abstract*—Incorporating biasing words obtained through contextual knowledge is paramount in automatic speech recognition (ASR) applications. This paper proposes an innovative method for achieving end-to-end contextual ASR using graph neural network (GNN) encodings based on the tree-constrained pointer generator method. GNN node encodings facilitate lookahead for future word pieces in the process of ASR decoding at each tree node by incorporating information about all word pieces on the tree branches rooted from it. This results in a more precise prediction of the generation probability of the biasing words. The study explores three GNN encoding techniques: namely the tree recursive neural network (Tree-RNN), the graph convolutional network (GCN), and GraphSAGE, along with different combinations of the complementary GCN and GraphSAGE structures. The performance of the systems was evaluated using both Librispeech and the AMI corpus with a visual-grounded contextual ASR pipeline. The findings indicate that using GNN encodings achieved consistent and significant reductions in word error rate (WER), particularly for words that are rare or have not been seen during the training process. Notably, on LibriSpeech test sets, the combined GNN proposed in this paper achieved a 20% relative rare word error rate reduction compared to Tree-RNN, 30%–40% compared to standard TCPGen and 60% compared to standard ASR systems without TCPGen.**

***Index Terms*—Pointer generator, contextual speech recognition, end-to-end, graph neural networks, audio-visual.**

I. Introduction

END-TO-END ASR systems often struggle with the accurate recognition of rare "long-tail" words that were not present in the training data. To combat this issue, *contextual biasing* which applies external contextual knowledge to the ASR system during inference, becomes a crucial factor in addressing the long-tail word problem in various applications [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. Contextual knowledge is often represented as a list (referred to as a *biasing list*) of words or phrases (referred to as *biasing words*) that are likely to appear in a given context. Biasing lists can be sourced from various resources, such as a user's contact book or playlist, recently visited websites and visual information from presentation slides etc. Despite their infrequent occurrence and thus the limited influence on the overall word error rate (WER), biasing words significantly impacts understanding the spoken content as biasing words are mostly content words such as nouns or proper nouns, which are crucial for downstream tasks. The inclusion of a word in a biasing list increases the probability of being correctly recognised, making contextual biasing a crucial component for the accurate recognition of rare content words.

End-to-end trainable ASR systems [15], [16] are designed to encapsulate all necessary knowledge within a single, static model, making it difficult to integrate dynamic context-specific knowledge at test-time. To overcome this challenge, specialised contextual biasing methods have been proposed, such as shallow fusion (SF) with a weighted finite-state transducer (WFST) or an adapted language model (LM) that incorporates contextual knowledge [1], [2], [3], [12], [13], [14], attention-based deep context approaches [4], [5], [6], [7], [8], as well as deep biasing (DB) with a prefix tree for improved efficiency when dealing with large biasing lists [6], [9], [10]. More recently, contextual biasing components with a pointer generator mechanism [17], [18], [19] that directly modifies the output distribution have been proposed [20], [21], which can be jointly optimised with ASR systems. In particular, the Tree-constrained pointer generator (TCPGen) component proposed in [20] builds a neural shortcut by directly interpolating the original model distribution with the TCPGen distribution estimated from contextual knowledge structured as a prefix-tree, based on a dynamic interpolation weight predicted by the TCPGen component. Hence, TCPGen combines the advantages of both the shallow fusion and deep biasing methods, and the prefix-tree structure of the biasing list guarantees high efficiency.

The performance of TCPGen can be further boosted by encoding the prefix-tree with graph neural networks (GNN) to allow lookahead for biasing words. This paper, which is a substantial extension and more complete exposition of our work in [22], investigates the use of three types of GNNs for prefix-tree encoding in TCPGen.[1] These include the Tree-RNN, the graph convolutional network (GCN) [23] with its variant GCNII [24], and GraphSAGE with the max-pooling aggregator [25]. While Tree-RNN is a representative GNN model with a single recursive layer, GCN and GraphSAGE are two popular and effective multi-layer GNN designs. Specifically, GCN encodes the tree by utilising a spectral representation, while GraphSAGE, as a spatial method, directly explores the graph topology [26]. To further enhance the performance of GNN tree encodings, this

Manuscript received 22 May 2023; revised 11 January 2024; accepted 29 March 2024. Date of publication 16 April 2024; date of current version 26 April 2024. This work was supported by the Cambridge Tier-2 system operated by the University of Cambridge Research Computing Service (www.hpc.cam.ac.uk) funded by EPSRC Tier-2 capital under Grant EP/T022159/1. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ozlem Kalinli. *(Corresponding author: Philip C. Woodland.)*

Guangzhi Sun and Philip C. Woodland are with the Department of Engineering, University of Cambridge, Trumpington St., CB2 1TQ Cambridge, U.K. (e-mail: gs534@cam.ac.uk; pcw@eng.cam.ac.uk).

Chao Zhang is with the Department of Electronic Engineering, Tsinghua University, Beijing 100190, China (e-mail: cz277@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TASLP.2024.3389645

[1]The code is at https://github.com/BriansIDP/espnet/tree/GNN.

paper proposes attentive and bilinear combination approaches to exploit the complementarity between GCN and GraphSAGE. Additionally, an effective parameter-tying scheme was introduced for both GCN and GraphSAGE to improve their performance with deeper structures.

GNN encodings provide more powerful node representations in the prefix tree of TCPGen, allowing for a "lookahead" functionality where each node contains not only its own word piece information but also information about its child branches. This improved node representation in TCPGen leads to more accurate generation probability predictions for biasing words, yielding improved contextual biasing by incorporating information about future word pieces during each ASR decoding step. TCPGen with GNN encodings, as a generic component for end-to-end ASR, is integrated into both the attention-based encoder-decoder (AED) [15], [27], [28], [29], [30], [31] and the neural transducer (N-T) architecture [16], [32], [33], [34], [35].

Experiments were conducted with two different setups: (1) a simulated contextual ASR task using LibriSpeech audiobook data, and, (2) a visual-grounded speech recognition pipeline [22] with the AMI meeting data. In addition to the consistent and large reductions in WER of rare content words achieved by TCPGen, using the proposed GNN encodings, especially when they are combined, achieved further significant improvements.

The remainder of this paper is organised as follows. Section II reviews related work. Section III introduces the TCPGen component. Section IV describes the details of the proposed GNNs. Section V and VI present the experimental setup and results. Section VII gives conclusions.

II. Related Work

A. End-to-End Contextual Speech Recognition

Recently, various contextual biasing algorithms have been developed for end-to-end ASR. One prominent research stream focuses on representing biasing lists as external weighted finite-state transducers (WFSTs), which are integrated into a class-based language model (LM) via shallow fusion (SF) [1], [2], [3], [13]. These methods often depend on context prefixes such as "call" or "play", limiting their ability to handle the diverse grammar in natural speech. On the other hand, deep context approaches, often using attention mechanisms, have also been proposed, which encode the biasing list into a vector to use as input for the end-to-end ASR models [4], [5], [6], [7], [8]. While deep context approaches eliminate the dependence on syntactic prefixes seen in SF methods, they require more memory and are less effective for handling large biasing lists.

The study in [9] combined the use of deep context and shallow fusion of a WFST in an N-T and is referred to as deep biasing (DB). DB has improved efficiency by limiting the biasing vector extraction to a subset of word pieces determined by a prefix tree representation of the biasing list. The prefix-tree-based method was further expanded in [10] to include RNN LMs for shallow fusion, resulting in improved recognition of biasing words. Prior research only analysed industry datasets, however, [10] proposed and validated a simulation of contextual biasing on open-source data by incorporating a large number of distractors into the list of biasing words in the utterance. More recently, [21] and [20] simultaneously proposed a neural shortcut between the biasing list and the final model output distribution. In contrast to [21], TCPGen [20] adopted a structured prefix-tree representation for biasing lists, as discussed in detail in section III.

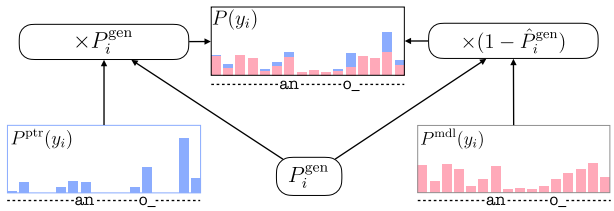

Fig. 1. Illustration of interpolation in TCPGen with corresponding terms in (5). $P^{\text{ptr}}(y_i)$ is the TCPGen distribution. $P^{\text{mdl}}(y_i)$ is the distribution generated by a standard end-to-end model. $P(y_i)$ is the final output distribution. $\hat{P}_i^{\text{gen}}$ and P_i^{gen} are the scaled and unscaled generation probabilities.

B. GNN for Speech and Language Processing

GNNs have been extensively employed in a multitude of speech and language tasks. In language-related applications, such as sentence-level text classification or word-level sequence labelling, GNNs are often utilised to capture the syntactic dependencies or semantic relations among words in a sentence. Furthermore, the encoding of subword-unit-based tree structures using GNNs [36], [37] has been explored for the purpose of generating more effective word representations. GNNs have also been applied to named-entity recognition [38] and neural machine translation [39], [40].

GNNs also have numerous applications in speech processing, such as text-to-speech synthesis, where GNNs model the syntactic and semantic relationships in the text. In GraphTTS [41], the authors structured the sentence into a hierarchical tree by dividing the utterance into words and then further into characters. This allowed the system to capture prosodic relationships among different parts of the input. Additionally, GNNs are used in paralinguistic tasks as the syntactic encoder, including sentiment classification and hate speech detection tasks [42], [43], [44]. In contrast to the aforementioned work, the application of GNNs to contextual biasing directly exploits the tree structure of the wordpiece lexicon. Combined with TCPGen, GNNs have a much more direct influence on the model output.

III. Tree-Constrained Pointer Generator

TCPGen is a neural network-based module that employs a combination of symbolic prefix-tree search and a neural pointer generator for contextual biasing, allowing for end-to-end optimisation. The biasing list is structured as a prefix tree at the word piece-level. At each output stage, TCPGen computes a probability distribution over a set of valid word pieces that are constrained by the prefix tree. TCPGen also predicts a dynamic generation probability, signifying the amount of contextual biasing required at the specific output step. The final output distribution is obtained by taking a weighted sum of the TCPGen distribution and the original AED or N-T output distribution (see Fig. 1).

The key symbolic representation of the external contextual knowledge in TCPGen is the prefix tree. For simplicity, examples and equations in this section are presented for a specific search path, which can be generalised easily to beam-search

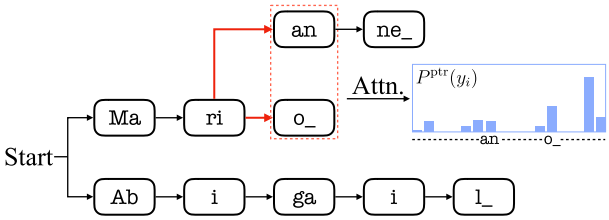

Fig. 2. Example of prefix tree search and attention in TCPGen. With previous output `ri`, `o_` and `an` are two valid word pieces on which attention will be performed. A word end unit is denoted by _.

with multiple paths. For the example prefix tree with biasing words (`Mario`, `Marianne` and `Abigail`) shown in Fig. 2, if the previously decoded word piece is `ri`, word pieces `o_` and `an` form the set of valid word pieces $\mathcal{Y}_i^{\text{tree}}$. Denoting $\mathbf{x}_{1:T}$ and y_i as input acoustic features and output word piece, $\mathbf{q}_i$ as the query vector carrying the decoding history and acoustic information, $\mathbf{K} = [\ldots, \mathbf{k}_j, \ldots]$ as the key vectors, scaled dot-product attention is performed between $\mathbf{q}_i$ and $\mathbf{K}$ to compute the TCPGen distribution P^{ptr} and the output vector $\mathbf{h}_i^{\text{ptr}}$ as shown in Equations (1) and (2).

$$P^{\text{ptr}}(y_i|y_{1:i-1}, \mathbf{x}_{1:T}) = \text{Softmax}(\text{Mask}(\mathbf{q}_i\mathbf{K}^{\text{T}}/\sqrt{d})), \tag{1}$$

$$\mathbf{h}_i^{\text{ptr}} = \sum\nolimits_j P^{\text{ptr}}(y_i = j|y_{1:i-1}, \mathbf{x}_{1:T})\, \mathbf{v}_j^{\text{T}}, \tag{2}$$

where d is the size of $\mathbf{q}_i$ (see [45]), $\text{Mask}(\cdot)$ sets the probabilities of word pieces that are not in $\mathcal{Y}_i^{\text{tree}}$ to zero, i.e. constraints, and $\mathbf{v}_j$ is the value vector relevant to j. Key and values are derived from GNN encodings introduced in section IV.

TCPGen can be employed in both AED and N-T. In AED, the query is the combination of the context vector and the embedding of the preceding word piece, while the keys and values are derived from the decoder word piece embedding using a shared projection matrix. The generation probability in AED is computed in (3).

$$P_i^{\text{gen}} = \sigma(W[\mathbf{h}_i^{\text{ptr}}; \mathbf{h}_i^{\text{dec}}] + \mathbf{b}), \tag{3}$$

where $\mathbf{h}_i^{\text{dec}}$ is the decoder hidden states. In the N-T, the pointer generator is applied to each combination of the encoder and the predictor step, where the TCPGen distribution is calculated using the concatenation of the corresponding encoder and predictor hidden states as the query. Keys and values in the N-T are computed from the predictor word piece embeddings. The generation probability for N-T is given in Equation (4).

$$P_{i,t}^{\text{gen}} = \sigma(W[\mathbf{h}_{i,t}^{\text{ptr}}; \mathbf{h}_{i,t}^{\text{joint}}] + \mathbf{b}). \tag{4}$$

To ensure that the probability of the blank symbol in the N-T is unchanged, $P_i^{\text{ptr}}(\varnothing|\mathbf{x}_{1:T}, y_{1:i-1})$ is set to 0 and the generation probability is scaled by $1 - P_i^{\text{mdl}}(\varnothing|\mathbf{x}_{1:T}, y_{1:i-1})$ where P_i^{mdl} is the original model distribution before interpolation.

For better flexibility, an *out-of-list* (OOL) token is included in $\mathcal{Y}_i^{\text{tree}}$ indicating that no suitable word piece can be found in the set of valid word pieces. To ensure that the final distribution sums to 1, the generation probability, P_i^{gen}, is scaled to be $\hat{P}_i^{\text{gen}} = P_i^{\text{gen}}(1 - P^{\text{ptr}}(\text{OOL}))$, and then the final output is calculated as shown in (5).

$$P(y_i) = P^{\text{mdl}}(y_i)(1 - \hat{P}_i^{\text{gen}}) + P^{\text{ptr}}(y_i)P_i^{\text{gen}}, \tag{5}$$

where dependencies, $y_{1:i-1}, \mathbf{x}_{1:T}$, are omitted for clarity. $P^{\text{mdl}}(y_i)$ represents the output distribution from the standard end-to-end model.

A. Biasing-Driven LM Discounting (BLMD) for TCPGen

Log-linear interpolation is often used as a technique to incorporate an external LM via SF [46], [47], [48]. Define the source domain data as the text of the training data for the end-to-end model, and the target domain data as the data used to train an external LM such that it generates better probability estimates for the test data. The LM discounting is defined in (6).

$$P^{\text{sf}}(y_i) = P^{\text{mdl}}(y_i)\frac{P^{\text{tgt}}(y_i)^{\alpha}}{P^{\text{src}}(y_i)^{\beta}}, \tag{6}$$

where $P^{\text{mdl}}(Y)$ is the probability from the end-to-end system, $P^{\text{src}}(Y)$ is the probability of the source domain LM and $P^{\text{tgt}}(Y)$ is the target domain LM probability. Extending this idea to contextual biasing with TCPGen [49], BLMD can be applied as shown in (7).

$$P^{\text{sf}}(y_i) = (1 - P^{\text{gen}})P^{\text{mdl}}(y_i)\frac{P^{\text{tgt}}(y_i)^{\alpha_1}}{P^{\text{src}}(y_i)^{\beta_1}} + P^{\text{gen}}P^{\text{ptr}}(y_i)\frac{P^{\text{tgt}}(y_i)^{\alpha_2}}{P^{\text{src}}(y_i)^{\beta_2}}, \tag{7}$$

where P^{ptr} is the TCPGen distribution, and the same source and target LMs are used for both distributions, but with different sets of hyper-parameters α_1, β_1 and α_2, β_2 tuned on the validation set.

IV. GNN Tree Encodings for TCPGen

While looking ahead into future branches of the paths being searched on the prefix tree is greatly beneficial to the correct prediction of the generation probability, node representations in standard TCPGen only contain information about the word piece on that node. To achieve lookahead functionality, GNNs are used to encapsulate future branch information into each node representation. The pipeline of applying GNN encodings in TCPGen is shown in Fig. 3.

The word piece prefix-tree is first encoded with a GNN to obtain encodings associated with each node. Then, the tree with GNN encodings is used by TCPGen, where the key and value for the TCPGen distribution are computed using the encoding of nodes in the set of valid word pieces, in place of word piece embeddings as shown in (8).

$$\mathbf{k}_j = W^{\text{K}}\mathbf{h}_{n_j}^{\text{gnn}} \qquad \mathbf{v}_j = W^{\text{V}}\mathbf{h}_{n_j}^{\text{gnn}}, \tag{8}$$

where W^{V} and W^{K} are parameter matrices, and $\mathbf{h}^{\text{gnn}}$ is the GNN node encoding obtained using different types of GNN. This paper explores three different types of GNN, namely the Tree-RNN, GCN (including its variant, GCNII) and GraphSAGE with max pooling, together with combinations of GCN and GraphSAGE as two complementary types of GNN. Details of GNN structures applied in TCPGen together with modifications are described in the following sections.

A. Tree-RNN

Tree-RNN recursively encodes the tree from leaf nodes to the root using an RNN structure. Specifically, at node n_j which contains child nodes $n_1, \ldots, n_k, \ldots, n_K$, the vector representation

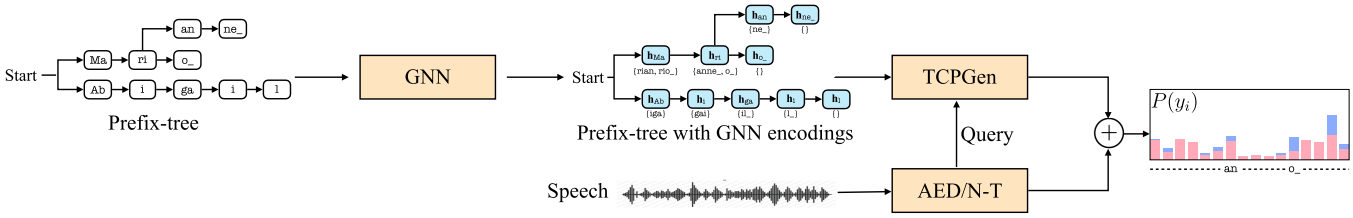

Fig. 3. Pipeline of encoding the prefix-tree with GNN for TCPGen. The prefix-tree is first encoded by a GNN, and the GNN-encoded tree is used by TCPGen to generate the TCPGen distribution where key and value vectors are GNN-based node encodings. The lookahead content for a 2-layer GCN as an example is denoted in {}, *i.e.* for node ri, $\mathbf{h}_{\texttt{ri}}$ covers information about an ne_ and o_.

of n_j can be written as (9).

$$\mathbf{h}^{\text{trnn}}_{n_j} = \text{ReLU}\left(W_1\mathbf{y}_j + \sum_{k=1:K} W_2\mathbf{h}^{\text{trnn}}_{n_k}\right), \tag{9}$$

where $\mathbf{h}^{\text{trnn}}_{n_k}$ is the vector representation of node n_k, and $\mathbf{y}_j$ is the embedding vector of the word piece of node n_j. W_1 and W_2 are parameter matrices jointly optimised with the ASR system by allowing gradient back-propagation through $\mathbf{h}^{\text{trnn}}_{n_k}$. In this way, each node recursively encodes information from its child nodes, such that the information of the entire branch rooted from it can be incorporated in the node encoding $\mathbf{h}^{\text{trnn}}_{n_k}$.

The encoding of each node is obtained by applying Equation (9) recursively from leaf to root. Then, for the same example shown in Fig. 2, at the node of ri, node encodings $\mathbf{h}^{\text{trnn}}_{\text{o\_}}$ and $\mathbf{h}^{\text{trnn}}_{\text{an}}$ are used to calculate the TCPGen distribution and $\mathbf{h}^{\text{ptr}}_{i}$. Therefore, if Mario appears in the utterance, TCPGen is aware of this entire word as early as in the encoding of ri. Such lookahead functionality achieves a more accurate prediction of the generation probability to determine when contextual biasing is needed.

Although Tree-RNN achieves the required lookahead functionality, it uses a rather simple RNN structure to encode the information of all succeeding nodes on that branch into a single vector representation. Therefore, more powerful and flexible GNN encodings are also explored in this paper in order to improve performance.

B. Graph Convolutional Network (GCN)

As an alternative to Tree-RNN, GCN is applied to tree encodings in order to achieve better node representations. Tree-RNN uses a recursive computation from the leaf nodes to the root, which limits the depth of the encoding network. However, GCN addresses this limitation by leveraging the adjacency matrix for parallel computation. GCN is a multi-layer network where each layer computes the encoding of a node as a function of its neighbours based on the graph Laplacian matrix. Each layer of a GCN conducts one message passing from immediately neighbouring nodes. For a GCN with L layers, the encoding of a node covers information from a node that is an L-hop ahead on branches rooted from it. In this way, the lookahead distance is controllable by the number of layers, allowing more flexibility and more model parameters to be efficiently incorporated, which is another advantage over Tree-RNN.[2]

[2]The total number of model parameters of GCN is still negligible compared to the entire ASR model.

Specifically, define $H^{\text{gcn}}(l) = [\mathbf{h}^{\text{gcn}}_{n_1}(l), \ldots, \mathbf{h}^{\text{gcn}}_{n_N}(l)]$ as the node encoding matrix of layer l whose rows $\mathbf{h}^{\text{gcn}}_{n_j}(l)$ are the encoding of the n_j nodes, the GCN layer computation is

$$H^{\text{gcn}}(l+1) = f(\hat{P}H^{\text{gcn}}(l)W(l)), \tag{10}$$

where $W(l)$ is the parameter matrix of l, $f(\cdot)$ is the activation function, $\hat{A} = A + I_N$ is the adjacency matrix with self-loops to enable the information of the current node to be included in the node representation, and $\hat{D}$ is the degree matrix of $\hat{A}$. A specific form of normalised graph Laplacian [23],

$$\hat{P} = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}, \tag{11}$$

is used to address the vanishing/exploding gradient problem. Note that as only future branch information is needed in TCPGen, $\hat{A}$ only contains edges that lead to child nodes, and hence $\hat{D}$ is computed based on this modified $\hat{A}$. TCPGen then takes the node encodings of the final layer, $H^{\text{gcn}}(L)$, to compute key and value vectors in the same way as Tree-RNN. Residual connections and layer normalisation are added for GCNs with multiple layers.

Although lookahead with a configurable scope can be achieved by varying the number of layers L, recent research found that the performance of GCN starts to degrade with more than three layers [24], [50]. Apart from the fact that deeper networks are more difficult to train, it was pointed out that the representations of the nodes in GCN are inclined to converge to a certain value and hence become indistinguishable, which is referred to as the *over-smoothing* problem [51]. One promising method to address this problem is to build a shortcut directly linking to the first GCN layer to ensure a certain fraction of the final representation comes from the current node itself. Thus, GCNII is also investigated in this paper with each layer computed by Equation (12):

$$H^{\text{gii}}(l+1) = f\Big([(1-\alpha)\hat{P}H^{\text{gii}}(l) + \alpha H^{\text{gii}}(0)]W_\beta(l)\Big), \tag{12}$$

where $H^{\text{gii}}(l)$ is the l-th layer output of GCNII, hyper-parameter α scales the shortcut to the first layer, and $W_\beta(l)$ is the parameter matrix defined as:

$$W_\beta(l) = (1-\beta_l)I_N + \beta_l W(l), \tag{13}$$

where β_l is a layer-dependent hyper-parameter which is $\ln(1/l + 1)$ in this paper. This formulation was inspired by the identity mapping form ResNet which ensures that a deep GCNII model achieves at least the same performance as its shallow version [24].

Although GCNII has shown improved performance with deeper GCNs of more than 4 layers [24], the maximum length

in our biasing list is usually less than 10. With this depth for tree structures, the over-smoothing problem is less of a concern compared to the best structure of GCNII with 64 layers, and network complexity is more problematic. Therefore, a simple parameter-sharing scheme is also proposed in this paper for deep GNNs to reduce network complexity. Specifically, the parameter matrices in the first K layers are shared:

$$W(1) = W(2) = \ldots = W(K) \tag{14}$$

In contrast to other complex graph structures, message-passing operations from child nodes to the root in a tree are similar across different layers. Therefore, having the same weight matrix representing this process effectively reduces the model complexity to a degree that is adequate for tree encoding. In particular, $K = L - 1$ is used for GCN in this paper so that there are effectively two-layer parameter matrices to be trained while maintaining the depth of the network. The first K layers act as universal message passing and the last layer performs a final information aggregation from neighbouring nodes.

C. GraphSAGE With Max Pooling

Node encodings for Tree-RNN and GCN are based on summation, whereas previous research [52], [53] has found that using max pooling also achieves competitive performance for word representation based on subword units. Hence, as an alternative GNN structure, GraphSAGE with a max pooling aggregator function is also studied in this paper. GraphSAGE is a multi-layer GNN with each layer performing an information aggregation over a sampled set of child nodes followed by an update to the representation of the current node. Although one of the innovations in GraphSAGE is fixed-size sampling, as the training time biasing list is already a sampled subset from the full biasing list, the sampling of GraphSAGE is omitted in this paper. The computation of each layer is

$$\begin{aligned}\mathbf{h}_{\mathcal{N}_i}(l+1) &= \max(\{\sigma(W_1(l)\mathbf{h}^{\text{sage}}_{n_k} + \mathbf{b}(l)), \forall n_k \in \mathcal{N}_j\}),\\ \mathbf{h}^{\text{sage}}_{n_j}(l+1) &= \sigma(W_2(l)\text{Concat}(\mathbf{h}_{\mathcal{N}_j}(l+1); \mathbf{h}^{\text{sage}}_{n_j}(l))),\end{aligned} \tag{15}$$

where $\max(\cdot)$ and $\text{Concat}(\cdot)$ denote the element-wise max pooling and concatenation operators, $\mathcal{N}_j$ is the set of child nodes of node n_j. Although slightly less sensitive than GCN, GraphSAGE also degrades when adding more layers [24]. Therefore, it is proposed in this paper to apply parameter-sharing for GraphSAGE, where both $W_1 = W_1(1) = W_1(2) = \ldots = W_1(L)$ and $W_2 = W_2(1) = W_2(2) = \ldots = W_2(L)$ are separately shared across all layers respectively.

D. Combination of GNN Encodings

Combinations [54], [55] of GCN and GraphSAGE are explored in this paper for tree encodings, as they are conceptually complementary. GCN adopts a spectral approach where the graph Laplacian is used to aggregate information, while GraphSAGE directly exploits the graph structure and performs a max-pooling aggregation. To exploit the complementarity between the two GNNs, both additive and multiplicative combination methods are investigated here.

Additive combination performs a weighted sum of node encodings from GCN and GraphSAGE as the final node encodings before being processed by TCPGen as shown in (16)

$$\mathbf{h}^{\text{comb}}_{n_j} = \alpha^{\text{gcn}} U_1 \mathbf{h}^{\text{gcn}}_{n_j} + \alpha^{\text{sage}} U_2 \mathbf{h}^{\text{sage}}_{n_j}, \tag{16}$$

where U_1 and U_2 are the two parameter matrices that rearrange the element ordering in each GNN encoding since they may not encode information in the same order [55]. Note that $\alpha^{\text{gcn}} + \alpha^{\text{sage}} = 1$ are weights that are either fixed or predicted via attention. The attention calculation is performed on each node n separately, as shown in (17)

$$[\alpha^{\text{gcn}}_{i,n}, \alpha^{\text{sage}}_{i,n}] = \text{Softmax}(\mathbf{q}_i^T [\mathbf{h}^{\text{gcn}}_{n_j}, \mathbf{h}^{\text{sage}}_{n_j}]), \tag{17}$$

where $\mathbf{q}_i$ is the same query vector used to calculate the TCPGen distribution. Therefore, different sets of weights are assigned to different nodes at different decoder steps.

Multiplicative combination is performed via a low-rank approximation of the bilinear pooling method. The combination is shown in (18)

$$\hat{\mathbf{h}}^{\text{comb}}_{n_j} = U_3(\tanh(U_1 \mathbf{h}^{\text{gcn}}_{n_j}) \odot \tanh(U_2 \mathbf{h}^{\text{sage}}_{n_j})), \tag{18}$$

where U_1, U_2 and U_3 are parameter matrices, and $\odot$ is the element-wise product between two vectors. Following [55], a shortcut connection from each individual GNN encoding was provided to form the final combined encoding for TCPGen, as shown in (19)

$$\mathbf{h}^{\text{comb}}_{n_j} = \hat{\mathbf{h}}^{\text{comb}}_{n_j} + U_4 \mathbf{h}^{\text{gcn}}_{n_j} + U_5 \mathbf{h}^{\text{gcn}}_{n_j}, \tag{19}$$

where U_4 and U_5 are another two parameter matrices.

V. Experimental Setup

A. Data

Experiments were conducted on two distinct datasets, namely the LibriSpeech audiobook corpus and the AMI meeting dataset, where the latter followed an audio-visual speech recognition pipeline. The LibriSpeech corpus [56], which consists of 960 hours of read English from audiobooks, was used for evaluation purposes, with the dev-clean and dev-other sets used for validation, while test-clean and test-other were employed for evaluation. To investigate the impact of critical hyper-parameters, small-scale experiments were carried out using the train-clean-100 subset as the training set. Moreover, models trained on the LibriSpeech dataset were fine-tuned and evaluated on the AMI dataset in accordance with the approach proposed in [22].

The AMI meeting corpus [57] consists of 100 hours of meeting recordings involving 4-5 individuals, which were divided into the train, dev, and eval sets. To demonstrate the effectiveness of contextual biasing on data from another domain with limited training resources, a subset comprising 10% of the utterances from the AMI training set corresponding to 8 hours of audio was used to fine-tune the models previously trained on the LibriSpeech 960-hour data. There were 14 meetings from the dev set and 8 meetings from the eval set accompanied by slides that were selected to formulate the new test set for the audio-visual contextual ASR pipeline.

The 80-dim mel filterbank features at a 10 ms frame rate were concatenated with 3-dim pitch features and used as the model input. SpecAugment [58] with the setting $(W, F, m_F, T, p, m_T) = (40, 27, 2, 40, 1.0, 2)$ was used without any other data augmentation or speaker adaptation.

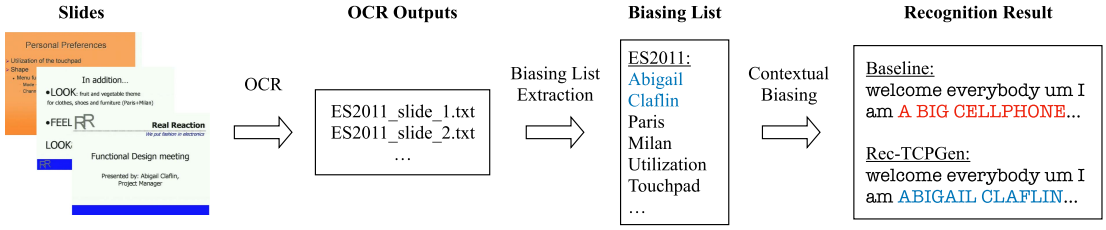

Fig. 4. Illustration of the visual-grounded contextual ASR pipeline for the meeting series ES2011 containing meetings ES2011a to ES2011d.

B. Biasing List Selection

To simulate real-world scenarios in LibriSpeech, the complete list of rare words comprising 200,000 distinct words as suggested in [10] was used. The full rare word list contained more than 60% out-of-vocabulary (OOV) words that were absent from the LibriSpeech speech training set. Consistent with the method proposed in [10], the biasing lists were extracted by identifying words from the full rare word list that appeared in the reference transcription of each utterance, followed by the addition of a specific number of distractors. During inference, 10.3% of the word tokens in the test sets belonged to the full rare word list.

Fig. 4 shows the visual-grounded contextual ASR pipeline for AMI that utilises optical character recognition (OCR) output for slides. The Tesseract 4 OCR engine, equipped with LSTM models[3], was first applied to the slides of each meeting series (e.g., ES2011[a-d]). Subsequently, distinct word tokens were extracted from the OCR output text files, and words in the full rare word list, which also occurred fewer than 100 times in the AMI training set, were selected to form the biasing list for that particular meeting series. These meeting-specific biasing lists were then used for the recognition of all utterances in that meeting series. The size of the biasing lists varied between 175 to 576 (mean 278, median 220), and the total number of word tokens covered by these lists was 1,751 out of 112,110 word tokens (1.5%). As shown in Fig. 4, these words mainly consisted of highly valuable content words whose accurate recognition was crucial for comprehending the utterance. Therefore, although the biasing lists had a minor impact on the overall WER, they were essential for improving the recognition performance of critical words. Details of the meetings with slides and the extraction pipeline are available.[4]

C. Model Specification

The ESPnet toolkit [59] was used for developing the systems. A unigram word piece model comprising 600 unique word pieces was created on the LibriSpeech data and was applied directly to the AMI data. Both the AED and N-T models employed a Conformer [60] encoder, which comprised 16 conformer blocks comprising 4 attention heads of size 512. The AED used a single-layer LSTM decoder of size 1024 and a location-sensitive attention mechanism featuring 4 heads of size 1024. The N-T, on the other hand, employed a 1024-dimensional predictor and a joint network consisting of a single fully-connected layer of size 1024. GNN encodings for LibriSpeech train-clean-100 experiments used 256-dim GNN encodings, whereas the LibriSpeech full-scale experiments used 1024-dim for GNN encoders. The number of model parameters is shown in Table I. The number of parameters for GNN modules is very small compared to the overall model size. Note that the number of layers and the dimensions of GNNs have been optimised to achieve the best performance.

TABLE I
NUMBER OF PARAMETERS FOR DIFFERENT MODULES

System	Full	Tree-RNN	GCN	GraphSAGE	Combined
AED	117M	0.3M	0.8M	0.8M	1.6M
N-T	108M	0.3M	0.7M	0.7M	1.4M

GCN and GraphSAGE measured using best-performing tied structure.

LM shallow fusion and BLMD were implemented using a two-layer LSTM-LM with 2048 hidden units trained on the 800 million-word text training corpus of LibriSpeech as the target domain LM for the LibriSpeech experiments. Each source domain LM, trained on the text of the audio training data, used a single-layer LSTM with 1024 hidden units. It is worth noting that each LM had the same word pieces as the corresponding ASR system.

D. Training Specifications

During training, biasing lists with 1000 distractors were used for the experiments conducted on the LibriSpeech dataset, while 100 distractors were used for the AMI data. To create these lists, biasing words were selected from the reference transcription, and additional distractors were added. To prevent the AED model from becoming overly confident about TCPGen outputs, a dropout-inspired technique was employed during training, as described in [10]. Specifically, biasing words that were presented in the reference transcription had a 30% probability of being removed from the biasing list. The Conformer was optimised using the Noam optimiser [45]. Additionally, the hyper-parameters for the BLMD model were determined based on the respective dev sets for each dataset.

E. Evaluation Metrics

In addition to the standard WER, the following metrics were used to evaluate the performance:

- The *rare word error rate (R-WER)* was used to evaluate the system performance on biasing words, i.e. words that were "rare" in the training data for that system. R-WER is the total number of *error* word tokens that belong to the biasing list divided by the total number of word tokens in the test set that belong to the biasing list. Insertion errors were counted in R-WER if the inserted word belonged to the biasing list, in contrast to [21].

[3]OCR implementation at https://github.com/tesseract-ocr/tesseract
[4]https://github.com/BriansIDP/AMIbasing

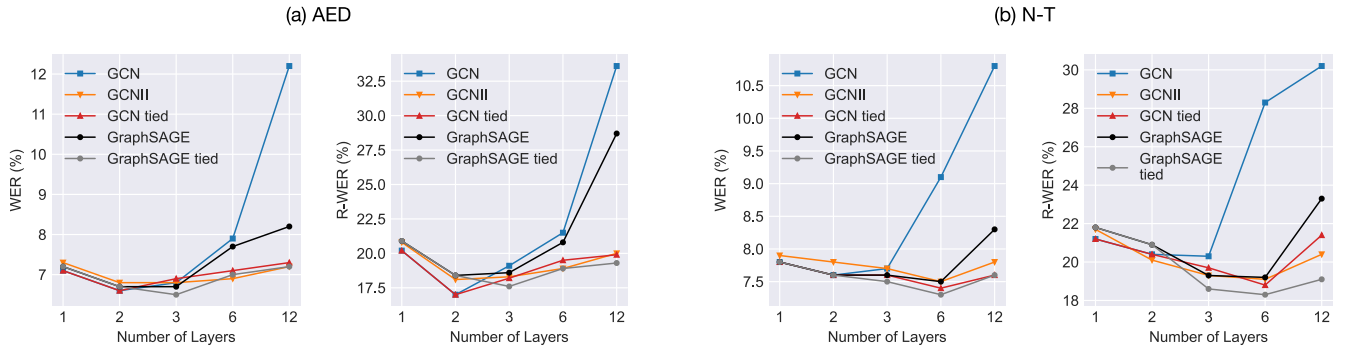

Fig. 5. Plot of WER (%) and R-WER (%) against the number of GNN layers for (a) AED and (b) N-T on LibriSpeech test-clean data. Systems were trained on train-clean-100 for 120 epochs. Biasing lists with 1000 distractors were used. "Tied" referred to the parameter-tying scheme.

- The *OOV WER* was also computed in the same way as R-WER but for OOV words in the biasing list. OOV errors are a subset of rare word errors. There are altogether 443 such words in LibriSpeech test-clean and test-other sets.
- *The slides rare word error rate (R_s-WER)* is reported for the AMI experiments calculated in the same way as R-WER, but for the rare words in the slides. Insertions of the slides biasing words were included in R_s-WER.

As rare words are fairly scarce in both LibriSpeech and AMI, significance tests were performed to show that the improvements using GNN encodings were statistically significant. Specifically, independence was assumed at the book level for LibriSpeech and the speaker level for AMI. The significance tests were done separately for test-clean and test-other. The alternative hypothesis was defined as the GNN system performing better than standard TCPGen (i.e. one-tailed sign test).

VI. Results

A. LibriSpeech train-Clean-100 Results

Experiments were first performed on the train-clean-100 set to find the best-performing GNN setups. The first investigation was on the number of GNN layers which determines how much lookahead is needed. Plots of WER and R-WER against the number of layers for AED and N-T are shown in Fig. 5. Note that parameter tying only had an effect when the layer number exceeded two.

For N-T, both GCN and GraphSAGE had a clear trend that the performance tended to degrade when the number of layers was increased beyond three and degraded significantly when it reached twelve layers. GraphSAGE suffered less from this problem than GCN as the max pooling operator enabled the gradient for salient nodes to remain at its original value instead of being over-smoothed [24]. This degradation was mitigated by either using the GCNII structure or the parameter-tying scheme proposed in this paper. The best performance was achieved by 6-layer systems using WER as the selection criterion, where GCN and GraphSAGE with tied parameters achieved better performance than GCNII.

Similar observations were found with AED, where GCN and GraphSAGE with tied parameters achieved slightly better performance than GCNII. However, the best performance was achieved using 2-layer GCN and 3-layer GraphSAGE models. This difference is mainly caused by the label-synchronous nature of AED, where the model required knowledge for predicting only the next token, and information further into the future is less useful for this prediction in contrast to N-T.

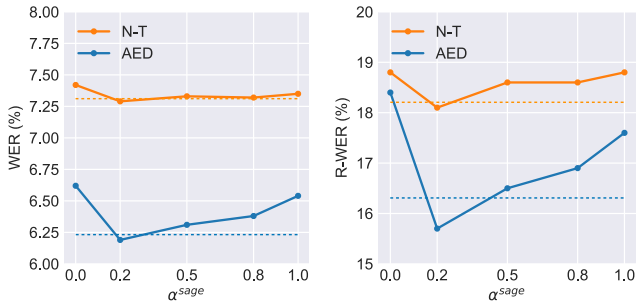

Fig. 6. Variation of WER and R-WER against model combination weight α_{sage}, and dynamic refers to using attention mechanism for weight calculation.

The best-performing GCN and GraphSAGE with tied parameters were used for model combination. For additive combination, different fixed combination weights gave the results shown in Fig. 6, together with dynamic combination weights. As a result, dynamic combination tends to attach high weights to GCN as it is better at handling near-future information.

B. LibriSpeech 960-Hour Results

The main results for LibriSpeech full-set experiments are summarised in Table II for AED and in Table III for N-T. Compared to standard TCPGen, all three types of GNNs achieved significantly better WER and R-WER (at p values smaller than 0.01) on both test sets for both AED and N-T. GCN and GraphSAGE with the proposed parameter tying achieved less obvious but still statistically significant ($p < 0.05$) improvements compared to Tree-RNN (comparing rows 4 and 6 to 3 in Table II and III) in both WER and R-WER. With similar levels of R-WER reduction, AED achieved a higher reduction in WER. As analysed in [49], TCPGen produced a much more confident prediction of P^{gen} with AED than N-T, where the main reductions in overall WER were attributed to the reduction in R-WER. The improvements using GNN indicate that the GNN encoding improved the prediction of P^{gen}, which was more beneficial for the overall WER in AED.

Finally, selected systems were evaluated with BLMD, where TCPGen with GNN encodings achieved further performance improvements and relative differences among different GNN

TABLE II
WER (UNB.) AND R-WER ON LIBRISPEECH TEST-CLEAN AND TEST-OTHER SETS USING CONFORMER **AED** AND TCPGEN TRAINED ON LIBRISPEECH 960-HOUR DATA WITH VARIOUS GNN ENCODINGS

System	GNN Enc.	BLMD	test-clean (%)			test-other (%)		
			WER (Unb.)	R-WER	OOV WER	WER (Unb.)	R-WER	OOV WER
Conformer AED	N/A	×	3.71 (2.57)	13.2	71.2	9.36 (7.15)	29.5	75.5
+TCPGen	No	×	3.34 (2.84)	8.4	40.1	8.43 (7.02)	21.3	41.5
+TCPGen	Tree-RNN	×	3.13 (2.70)	6.7	33.8	7.94 (6.86)	17.8	34.7
+TCPGen	GCN tied	×	2.81 (2.34)	6.7	32.9	7.34 (6.28)	17.1	33.6
+TCPGen	GCNII	×	2.88 (2.41)	6.8	34.1	7.46 (6.36)	17.5	34.7
+TCPGen	GraphSAGE tied	×	2.91 (2.45)	6.6	32.1	7.42 (6.37)	17.0	33.1
+TCPGen	Additive Combination (fixed)	×	2.59 (2.20)	5.8	31.8	7.00 (6.07)	15.5	30.5
+TCPGen	Additive Combination (dynamic)	×	2.72 (2.29)	6.3	31.5	7.17 (6.16)	16.4	31.8
+TCPGen	Bilinear Combination	×	2.62 (2.20)	6.1	32.3	7.08 (6.09)	16.1	31.4
Conformer AED	N/A	✓	3.33 (2.25)	12.3	69.3	8.04 (5.90)	27.6	74.2
+TCPGen	No	✓	2.79 (2.30)	6.9	31.3	7.40 (6.08)	19.5	32.5
+TCPGen	Tree-RNN	✓	2.61 (2.27)	5.4	28.9	6.85 (5.98)	14.8	28.3
+TCPGen	GCN tied	✓	2.48 (2.15)	5.2	28.1	6.33 (5.47)	14.2	27.7
+TCPGen	GraphSAGE	✓	2.55 (2.23)	5.2	27.8	6.40 (5.57)	14.0	27.1
+TCPGen	Additive Combination (fixed)	✓	**2.26 (2.03)**	**4.2**	**23.9**	**5.87 (5.25)**	**11.5**	**23.0**

Unb. Stands for %WER for unbiased words. Note both GCN (2-layer) and GraphSAGE (3-layer) used parameter tying. GCN and additive combination (fixed) selected as representative GNNs evaluated with BLMD.

TABLE III
WER AND R-WER ON LIBRISPEECH TEST-CLEAN AND TEST-OTHER SETS USING CONFORMER **N-T** AND TCPGEN TRAINED ON LIBRISPEECH 960-HOUR DATA WITH VARIOUS GNN ENCODINGS

System	GNN Enc.	BLMD	test-clean (%)			test-other (%)		
			WER (Unb.)	R-WER	OOV WER	WER (Unb.)	R-WER	OOV WER
Conformer N-T	N/A	×	4.02 (2.81)	14.1	80.1	10.12 (7.60)	33.1	83.2
+DB [10]	No	×	3.57 (2.82)	10.4	45.6	9.45 (7.75)	25.0	48.3
+TCPGen	No	×	3.40 (2.74)	8.9	43.3	8.79 (7.32)	22.2	46.7
+TCPGen	Tree-RNN	×	3.14 (2.60)	7.6	40.6	8.23 (7.07)	18.8	45.3
+TCPGen	GCN tied	×	3.11 (2.64)	7.0	39.1	8.14 (7.02)	18.4	43.7
+TCPGen	GCNII	×	3.16 (2.61)	7.7	40.1	8.36 (7.21)	18.9	45.2
+TCPGen	GraphSAGE tied	×	3.10 (2.64)	6.9	39.1	8.18 (7.04)	18.6	44.2
+TCPGen	Additive Combination (fixed)	×	2.99 (2.57)	6.5	37.5	8.10 (7.16)	16.7	38.9
+TCPGen	Additive Combination (dynamic)	×	3.02 (2.59)	6.6	39.1	8.14 (7.15)	17.2	40.3
+TCPGen	Bilinear Combination	×	2.97 (2.58)	6.2	36.9	8.02 (7.08)	16.6	38.1
Conformer N-T	N/A	✓	3.55 (2.47)	12.5	78.2	8.90 (6.55)	30.4	81.8
+TCPGen	No	✓	3.02 (2.42)	8.0	40.6	7.49 (6.27)	18.6	43.7
+TCPGen	Tree-RNN	✓	2.78 (2.28)	6.9	39.1	7.39 (6.23)	18.0	44.3
+TCPGen	GCN	✓	2.68 (2.24)	6.3	37.7	7.38 (6.24)	17.8	44.9
+TCPGen	GraphSAGE tied	✓	2.71 (2.28)	6.3	38.0	7.34 (6.21)	17.7	44.9
+TCPGen	Bilinear Combination	✓	**2.56 (2.23)**	**5.3**	**34.9**	**6.89 (6.10)**	**14.1**	**34.8**

Unb. Stands for %WER for unbiased words. Note both GCN (6-layer) and GraphSAGE (6-layer) used parameter tying. GCN and bilinear combination selected as representative GNNs evaluated with BLMD.

encodings persist. It is worthwhile pointing out that the best-performing GNN encoding for AED and RNN-T achieved around 60% relative R-WER reductions compared to standard AED and N-T systems, 40% relative to the DB [10] baseline, 30-40% relative R-WER reductions compared to standard TCPGen, and 20% relative R-WER reductions compared to Tree-RNN on both test sets. All the improvements are statistically significant at $p < 0.01$.

C. Influence of Biasing List Sizes

The influence of biasing list size on different GNN structures is shown in Fig. 7 for N-T. While all systems degraded when the biasing list increased to 5000, the best-performing GNN system still achieved a 34% relative R-WER reduction compared to the baseline. The rate of degradation with increasing biasing list size for GNN systems is in general faster than without GNN since the number of lookahead nodes grows quadratically with the increasing number of biasing words. It is also worthwhile pointing out that GCN, GraphSAGE and combined methods achieved much better performance with smaller sizes of biasing lists compared to Tree-RNN, demonstrating another advantage of using those systems when a small but important set of biasing entities is available, e.g. local hotels for a goal-oriented dialogue system [61].

TABLE IV
WER (R-WER) ON LIBRISPEECH TEST SETS FOR TCPGEN AND THE BEST-PERFORMING GNN ENCODINGS IN N-T WITH AND WITHOUT BIASING LISTS

Systems	Biasing list	Test-clean (%)	Test-other (%)
Std. N-T	N/A	4.02 (14.1)	10.12 (33.1)
+ TCPGen	✓	3.40 (8.9)	8.79 (22.2)
+ TCPGen	×	4.24 (17.4)	10.67 (36.7)
+ TCPGen + GNN	✓	2.56 (5.3)	6.89 (14.1)
+ TCPGen + GNN	×	4.37 (18.1)	10.81 (37.4)
+ scaling	✓	2.60 (6.1)	7.02 (16.2)
+ scaling	×	4.09 (15.1)	10.22 (34.6)

The biasing list size is 1000 if used. The scaling factor was set to 0.7 when scaling was used.

The performance for TCPGen in N-T with and without biasing lists is shown in Table IV. As biasing words were handled by the TCPGen component during training, the performance of those biasing words degraded when no biasing information

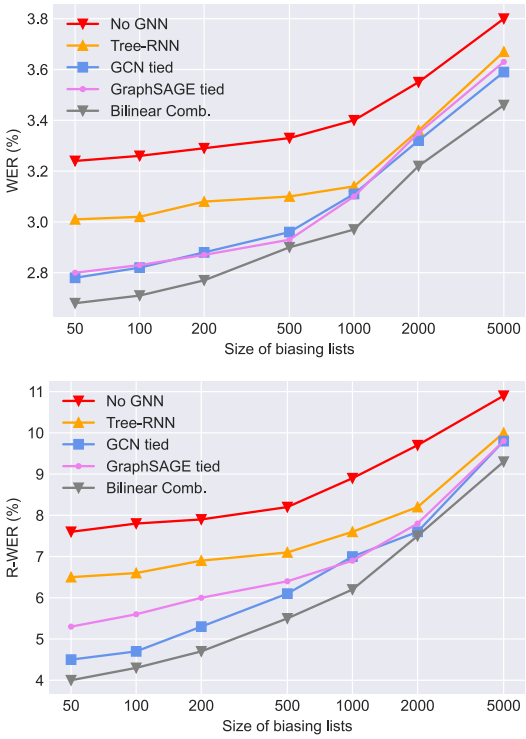

Fig. 7. WER and R-WER against different sizes of biasing lists for N-T model trained on LibriSpeech 960-hour data. Note that the WER for the standard N-T is 4.02 and R-WER for the standard N-T is 14.1.

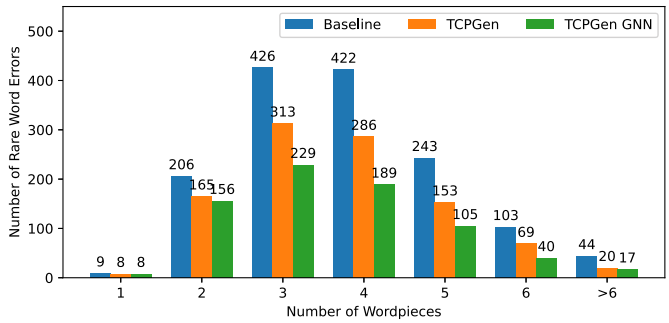

Fig. 8. Number of rare word errors against rare word lengths (number of wordpieces) on LibriSpeech test-other set using TCPGen with the best-performing GNN systems in AED.

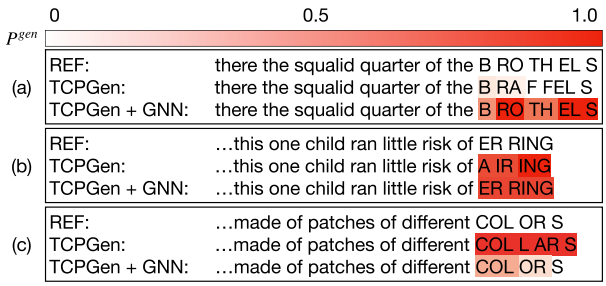

Fig. 9. Case studies using AED and additive combined GNN encodings on 3 utterances selected from the LibriSpeech test sets. The colour bar shows the value of P^{gen} for each wordpiece.

was present, causing a slight increase in the overall WER. The performance degradation was similar in AED. This problem can be easily mitigated by either increasing the drop rate or applying a scaling factor to the generation probability during training (shown in Table IV). However, it would be sensible to directly apply the unbiased system when no biasing information is available, and in practice, a more realistic scenario is when biasing information is scarce, and the experiments in Section VI-F exactly reflect this scenario.

D. Efficiency of GNN for Contextual Biasing

TCPGen with GNN encodings is still highly efficient when handling large biasing lists. In training, with a large biasing list of 1000 words, TCPGen with a Tree-RNN was 3.5 times slower than the standard AED or N-T model, with a negligible increase in space complexity. Among the three GNNs, GCN achieved the highest efficiency for training as its computation can be parallelised the most, whereas the recursive computation in Tree-RNN and the max-pooling in GraphSAGE hinder their training speed. As a result, GCN in training is over 30% faster than Tree-RNN, while GraphSAGE is 20% faster. Another source of latency is the construction of graph adjacency matrices during training for each mini-batch. As this computation can be shared between GCN and GraphSAGE, the combined GCN and GraphSAGE are only 20% slower than GraphSAGE during training while achieving statistically significantly better performance. Moreover, by generating GNN encodings offline before decoding once the biasing list is available, the time and space complexity during inference is close to the standard AED or N-T for biasing lists of thousands of words.

E. Error Analysis

The number of word errors against word length (measured in terms of the number of wordpieces) is shown in Fig. 8. GNN encodings benefit longer words more than shorter words. The effect was most obvious on words with 3 or 4 wordpieces, as the combined GNN allows maximum information propagation across 4 wordpieces.

To qualitatively illustrate how GNN encodings benefit recognition, a case study is shown in Fig. 9. In case (a), lookahead functionality helped early determination of biasing by predicting a larger P^{gen} at the first wordpiece, whereas in case (c), lookahead also prevented the model from copying biasing words by predicting a lower P^{gen} for a common word at the first wordpiece. This also explains how GCN and GraphSAGE reduce WER. Case (b) illustrates that GNN encodings achieve a better estimate of the TCPGen distribution when both systems are trying to copy from the biasing list.

F. AMI Audio-Visual Contextual ASR Experiments

The performance of various GNN encodings for TCPGen was further integrated into the audio-visual contextual ASR pipeline, and results are shown in Table V. In general, reductions in R-WER had a much smaller influence on the overall WER than with LibriSpeech, as rare words only occupied a much smaller portion of the data. The findings were consistent with LibriSpeech, where the best-performing combination methods for AED and N-T were additive and bilinear respectively. However, the relative R-WER improvement was smaller compared to that in the LibriSpeech experiments, as the best-performing system here already had an R-WER that was very close to the overall

TABLE V
WER (R_s-WER) ON AMI TEST SET USING THE AUDIO-VISUAL CONTEXTUAL ASR PIPELINE WITH 10% OF THE AMI TRAINING SET

System	GNN Encoding	BLMD	AED (%)	N-T (%)
AMI Baseline	N/A	×	23.6 (56.3)	26.5 (58.0)
Baseline	N/A	×	22.2 (51.2)	25.7 (52.6)
+TCPGen	No	×	22.0 (40.5)	25.5 (44.7)
+TCPGen	Tree-RNN	×	21.9 (36.7)	25.4 (40.7)
+TCPGen	GCN tied	×	21.9 (35.3)	25.4 (40.7)
+TCPGen	GraphSAGE tied	×	21.9 (36.4)	25.2 (39.6)
+TCPGen	Additive Combination	×	21.8 (33.1)	25.2 (37.2)
+TCPGen	Bilinear Combination	×	21.8 (33.5)	25.1 (36.2)
Baseline	N/A	✓	21.1 (45.5)	24.3 (46.5)
+TCPGen	No	✓	20.9 (34.2)	24.1 (37.7)
+TCPGen	GCN tied	✓	20.8 (32.2)	23.7 (35.8)
+TCPGen	Best Combination	✓	**20.7** (**29.7**)	**23.6** (**32.8**)

Baseline is the standard AED and N-T systems, and AMI baseline (the first row) specifically refers to the standard system trained from scratch on the full AMI training set.

WER whereas the R-WER in LibriSpeech was still twice as high as the overall WER. Compared to the baseline standard systems, TCPGen in AED achieved a statistically significant 35% relative R-WER reduction at $p < 0.01$ using the best combined GNN encodings, while TCPGen with the best combined GNN encodings in N-T achieved a significant 30% relative R-WER reduction both with and without BLMD.

VII. CONCLUSION

This paper proposes three different types of GNN encodings in TCPGen for end-to-end contextual ASR, including Tree-RNN, GCN and GraphSAGE. GNN encodings gave a lookahead functionality by incorporating future information on biasing list prefix-tree branches starting from the current node. Combination methods that take advantage of the complementarity between GCN and GraphSAGE were also explored. Experiments on LibriSpeech and AMI following an audio-visual contextual ASR pipeline showed consistent and significant WER and R-WER improvement for both AED and N-T systems using GNN encodings. On LibriSpeech, the best combined GNN encodings achieved over 60% R-WER and OOV-WER reductions compared to baseline standard systems without TCPGen, 30%-40% over systems with standard TCPGen, and 20% over TCPGen with Tree-RNN.

REFERENCES

[1] I. Williams, A. Kannan, P. Aleksic, D. Rybach, and T. Sainath, "Contextual speech recognition in end-to-end neural network systems using beam search," in *Proc. Interspeech*, 2018, pp. 2227–2231.

[2] Z. Chen, M. Jain, Y. Wang, M. L. Seltzer, and C. Fuegen, "End-to-end contextual speech recognition using class language models and a token passing decoder," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 6186–6190.

[3] D. Zhao et al., "Shallow-fusion end-to-end contextual biasing," in *Proc. Interspeech*, 2019, pp. 1418–1422.

[4] G. Pundak, T. Sainath, R. Prabhavalkar, A. Kannan, and D. Zhao, "Deep context: End-to-end contextual speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 418–425.

[5] Z. Chen, M. Jain, Y. Wang, M. L. Seltzer, and C. Fuegen, "Joint grapheme and phoneme embeddings for contextual end-to-end ASR," in *Proc. Interspeech*, 2019, pp. 3490–3494.

[6] M. Jain, G. Keren, J. Mahadeokar, G. Zweig, F. Metze, and Y. Saraf, "Contextual RNN-T for open domain ASR," in *Proc. Interspeech*,, 2020, pp. 11–15.

[7] U. Alon, G. Pundak, and T. Sainath, "Contextual speech recognition with difficult negative training examples," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 6440–6444.

[8] Z. Chen, M. Jain, Y. Wang, M. Seltzer, and C. Fuegen, "End-to-end contextual speech recognition using class language models and a token passing decoder," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 6186–6190.

[9] D. Le, G. Keren, J. Chan, J. Mahadeokar, C. Fuegen, and M. L. Seltzer, "Deep shallow fusion for RNN-T personalization," in *Proc. IEEE Spoken Lang. Technol. Workshop*, 2021, pp. 251–257.

[10] D. Le et al., "Contextualized streaming end-to-end speech recognition with trie-based deep biasing and shallow fusion," in *Proc. Interspeech*, 2021, pp. 1772–1776.

[11] A. Garg, A. Gupta, D. Gowda, S. Singh, and C. Kim, "Hierarchical multi-stage word-to-grapheme named entity corrector for automatic speech recognition," in *Proc. Interspeech*, 2020, pp. 1793–1797.

[12] Y. M. Kang and Y. Zhou, "Fast and robust unsupervised contextual biasing for speech recognition," U.S. Patent Application No 16/993,797, 2020, *arXiv:2005.01677*.

[13] R. Huang, O. Abdel-hamid, X. Li, and G. Evermann, "Class LM and word mapping for contextual biasing in end-to-end ASR," in *Proc. Interspeech*, 2020, pp. 4348–4351.

[14] D. Liu, C. Liu, F. Zhang, G. Synnaeve, Y. Saraf, and G. Zweig, "Contextualizing ASR lattice rescoring with hybrid pointer network language model," in *Proc. Interspeech*, 2020, pp. 3650–3654.

[15] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 577–585.

[16] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 6645–6649.

[17] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proc. Assoc. Comput. Linguistics*, 2017, pp. 1073–1083.

[18] Z. Liu, A. Ng, S. Lee, A. T. Aw, and N. F. Chen, "Topic-aware pointer-generator networks for summarizing spoken conversations," in *Proc. Autom. Speech Recognit. Understanding*, 2019, pp. 814–821.

[19] W. Li, R. Peng, Y. Wang, and Z. Yan, "Knowledge graph based natural language generation with adapted pointer-generator networks," *Neurocomputing*, vol. 328, pp. 174–187, 2020.

[20] G. Sun, C. Zhang, and P. C. Woodland, "Tree-constrained pointer generator for end-to-end contextual speech recognition," in *Proc. Autom. Speech Recognit. Understanding*, 2021, pp. 780–787.

[21] C. Huber, J. Hussain, S. Stüker, and A. Waibel, "Instant one-shot word-learning for context-specific neural sequence-to-sequence speech recognition," in *Proc. Autom. Speech Recognit. Understanding*, 2021, pp. 1–7.

[22] G. Sun, C. Zhang, and P. C. Woodland, "Tree-constrained pointer generator with graph neural network encodings for contextual speech recognition," in *Proc. Interspeech*, 2022, pp. 2043–2047.

[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[24] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1725–1735.

[25] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[26] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[27] L. Lu, X. Zhang, K. Cho, and S. Renals, "A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition," in *Proc. Interspeech*, 2015, pp. 3249–3253.

[28] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 4945–4949.

[29] C. C. Chiu et al., "State-of-the-art speech recognition with sequence-to-sequence models," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 4774-4778.

[30] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, "Improved training of end-to-end attention models for speech recognition," in *Proc. Interspeech*, 2018, pp. 7–11.

[31] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 4960–4964.

[32] R. Prabhavalkar, K. Rao, T. Sainath, B. Li, L. Johnson, and N. Jaitly, "A comparison of sequence-to-sequence models for speech recognition," in *Proc. Interspeech*, 2017, pp. 939–943.
[33] E. Battenberg et al., "Exploring neural transducers for end-to-end speech recognition," in *Proc. Autom. Speech Recognit. Understanding*, 2017, pp. 206–213.
[34] J. Li, R. Zhao, H. Hu, and Y. Gong, "Improving RNN transducer modeling for end-to-end speech recognition," in *Proc. Autom. Speech Recognit. Understanding*, 2019, pp. 114–121.
[35] Y. He et al., "Streaming end-to-end speech recognition for mobile devices," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 6381–6385.
[36] M. T. Luong, R. Socher, and C. D. Manning, "Better word representations with recursive neural networks for morphology," in *Proc. 17th Conf. Computat. Natural Lang. Learn.*, 2013, pp. 104–113.
[37] M. Nguyen, G. H. Ngo, and N. F. Chen, "Hierarchical character embeddings: Learning phonological and semantic representations in languages of logographic origin using recursive neural networks," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 461–473, 2020.
[38] P. H. Li, R. P. Dong, Y. S. Wang, J. C. Chou, and W. Y. Ma, "Leveraging linguistic structures for named entity recognition with bidirectional recursive neural networks," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 2664–2669.
[39] S. Liu, N. Yang, M. Li, and M. Zhou, "A recursive recurrent neural network for statistical machine translation," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, 2014, pp. 1491–1500.
[40] Y. Kawara, C. Chu, and Y. Arase, "Recursive neural network based preordering for english-to-japanese machine translation," in *Proc. ACL-SRW*, 2018, pp. 21–27.
[41] A. Sun, J. Wang, N. Cheng, H. Peng, Z. Zeng, and J. Xiao, "GraphTTS: Graph-to-sequence modelling in neural text-to-speech," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 6719–6723.
[42] X. Chen, X. Qiu, C. Zhu, S. Wu, and X. Huang, "Sentence modeling with gated recursive neural network," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 793–798.
[43] T. Zhang, M. Huang, and L. Zhao, "Learning structured representation for text classification via reinforcement learning," in *Proc. AAAI*, 2018, pp. 6053–6060.
[44] H. Sadr, M. M. Pedram, and M. Teshnehlab, "A robust sentiment analysis method based on sequential combination of convolutional and recursive neural networks," *Neural Process. Lett.*, vol. 50, pp. 2745–2761, 2019.
[45] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
[46] E. Variani, D. Rybach, C. Allauzen, and M. Riley, "Hybrid autoregressive transducer (HAT)," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 6139–6143.
[47] Z. Meng et al., "Internal language model estimation for domain-adaptive end-to-end speech recognition," in *Proc. IEEE Spoken Lang. Technol. Workshop*, 2021, pp. 243–250.
[48] J. Andrés-Ferrer, D. Albesano, P. Zhan, and P. Vozila, "Contextual density ratio for language model biasing of sequence to sequence ASR systems," in *Proc. Interspeech*, 2021, pp. 2007–2011.
[49] G. Sun, C. Zhang, and P. C. Woodland, "Minimising biasing word errors for contextual ASR with the tree-constrained pointer generator," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 31, pp. 345–354, 2022.
[50] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized PageRank," in *Proc. Int. Conf. Learn. Representations*, 2019.
[51] Q. Li, Z. Han, and X. M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 3538–3545.
[52] D. Shen et al., "Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms," in *Proc. Assoc. Comput. Linguistics*, 2018, pp. 440–450.
[53] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 649–657.
[54] C. Zhang, B. Li, Z. Lu, T. N. Sainath, and S. Chang, "Improving the fusion of acoustic and text representations in RNN-T," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 8117–8121.
[55] G. Sun, C. Zhang, and P. C. Woodland, "Combination of deep speaker embeddings for diarisation," *Neural Netw.*, vol. 141, pp. 372–384, 2021.
[56] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 5206–5210.
[57] J. Carletta et al., "The AMI meeting corpus: A preannouncement," in *Proc. Int. Workshop Mach. Learn. Multimodal Interact.*, 2006, pp. 28–39.
[58] D. S. Park et al., "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.
[59] S. Watanabe et al., "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, 2018, pp. 2207–2211.
[60] A. Gulati et al., "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.
[61] G. Sun, C. Zhang, and P. C. Woodland, "End-to-end spoken language understanding with tree-constrained pointer generator," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2023, pp. 1–5.

Guangzhi Sun (Member, IEEE) received the B.A. and M.Eng. degrees from the Trinity College, University of Cambridge, Cambridge, U.K., in 2019, and the Ph.D. degree with the Department of Engineering, University of Cambridge, Cambridge, U.K., under the supervision of Prof. Phil Woodland and advised by Prof. Mark Gales on contextual knowledge integration in end-to-end neural-based conversational AI systems. He is currently a Research Associate with the Machine Intelligence Lab, University of Cambridge. His research interests focuses on controllable and reliable multi-modal conversational AI with large language models. He held a Research Internship at Google Brain with Dr. Yu Zhang in 2019 and ByteDance with Dr. Wei Li in 2023. He was the recipient of the Best Student Paper in interspeech 2022.

Chao Zhang (Member, IEEE) received the B.E. and M.Sc. degrees from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2009 and 2012, and the Ph.D. degree from Cambridge University Engineering Department (CUED), Cambridge, U.K., in 2017. He is currently an Assistant Professor with the Department of Electronic Engineering, Tsinghua University. He was a Senior Research Scientist with Google, a Research Associate with CUED, and an Advisor and Speech Team Co-leader of JD.com. He has authored or coauthored 90 peer-reviewed speech and language processing papers and received multiple paper awards. His research interests include spoken language processing, machine learning, and cognitive neuroscience.

Philip C. Woodland (Fellow, IEEE) is currently a Professor of information engineering with the University of Cambridge, Cambridge, U.K., where he has been a member of academic staff for more than 30 years and a Professor since 2002. He is best known for his work on speech technology and related areas. He has authored or coauthored about 300 conference and journal papers with more than 25 000 citations (Google Scholar: h-index 72 & i10-index 221) including the most cited paper to appear in the journal Computer Speech & Language. He was the recipient of numerous best paper awards include the single "Best Student Paper" with his Ph.D. students as first author at two recent flagship conferences of the IEEE Speech and Language Technical Committee: ASRU 2019 in 2019, SLT 2021in 2021, as well one of three best student papers (awarded over different topic areas) for Interspeech 2022 which had more than 1100 accepted papers. He developed techniques for speaker adaptation and discriminative training that became very widely used in both research and commercial automatic speech recognition (ASR) systems. His team developed ASR systems that were frequently the most accurate in international research evaluations organised by the U.S. Government. Furthermore, his group developed the most accurate systems in the Multi-Genre Broadcast challenge to process BBC TV programmes in each of the challenge tasks (transcription, diarisation and alignment). He was one of the original co-authors of the widely-used HTK toolkit and has continued to play a major role in its development. He has been the Principal Investigator on research grants totalling more than £10m. He has supervised about 60 Ph.D. students and post-doctoral research associates and many of these have gone on to very strong academic or industrial research careers. He is a Fellow of the International Speech Communication Association (ISCA) and of the Royal Academy of Engineering.