INTERNAL LANGUAGE MODEL PERSONALIZATION OF E2E AUTOMATIC SPEECH RECOGNITION USING RANDOM ENCODER FEATURES

Adam Stooke, Khe Chai Sim, Mason Chua, Tsendsuren Munkhdalai, Trevor Strohman

Google LLC, USA

ABSTRACT

End-to-end (E2E) speech-to-text models generally require transcribed audio for training and personalization. We introduce the use of random audio encoder features, rather than speech, to fine-tune the final model layers and acquire new vocabulary from text-only data. This technique can be used for on-device personalization before the user has provided any speech data. We show improvements in the recall of new vocabulary and word error rate (WER) on held-out test sets using simulated user experiments on hybrid autoregressive transducer (HAT) models using conformer-based encoders and simple text embeddings for label processing. We compare this approach to the use of synthetic audio, finding random encoder features to be more beneficial with lower computational cost. Experiments show that the maximum benefit is gained by updating specific network components comprising a subset of those expressing the internal language model.

***Index Terms*—,** On-device personalization, end-to-end, automatic speech recognition, fine-tuning, rare vocabulary, data augmentation

1. INTRODUCTION

Leading automatic speech recognition (ASR) programs use deep neural networks and are trained in an end-to-end (E2E) fashion using recorded human speech paired with corresponding text. Relative to previous methods using individually trained separate components, such as acoustic, pronunciation and language models, E2E training has achieved superior performance from more compact models, especially in high-resource situations [1, 2, 3, 4, 5]. Model compactness is especially important for ASR systems that run on mobile devices, which have tighter computational constraints [6]. Even with a large corpus of transcribed audio available for training, a great many rare words and names exist which the base model is unlikely to capture, especially under the limited capacity of light-weight, on-device programs [7, 8]. Accuracy on rare and user-specific vocabulary is therefore an outstanding challenge in fielding such ASR systems, and a core use case of personalization.

While our baseline ASR models are trained end-to-end, their architecture still contains components with distinct modalities. Namely, these are 1) an audio encoder network, 2) a text prediction network, and 3) a "joint" network, which combines the outputs of (1) and (2) to produce the final word-piece probabilities [9]. Ideally, paired text and audio data is available from a user to train all components together. In many cases, however, text-only data will be more readily available. Furthermore, during on-device personalization, it is often already the case that only the last few layers of the E2E model are fine-tuned [10]. Under this scenario, our guiding insight is that it should still be possible to train the *internal language model* on new words, and hence improve the overall ASR performance, without needing real audio data. To do so, we propose to replace the audio encoder outputs with random values during fine-tuning. The random encoder features stimulate the joint network while new words are learned under the standard RNN-T loss function. This approach allows the internal language model to acquire new vocabulary from text-only data, and it does so without requiring any new model components. While it is not as performant as training with real audio, we will show that our approach learns at least as well as when using text-to-speech (TTS) inputs for simulated audio, and hence precludes the need to run a potentially expensive speech synthesizer.

The remainder of this paper is organized as follows. In Section 2 we describe previous methods for on-device ASR model personalization. We present in Section 3 the method of random encoder values, after reviewing relevant details of the RNN-Transducer loss function. Section 4 details our experiment methodology and results, including comparisons against using real and synthetic audio, hyperparameter sensitivity, and other ablations, finally followed by discussion in Section 5.

This work would not have been possible without the creation of WikiNames by Fan Gao, Tamar Lucassen, Richa Singh, Chetan Gupta, Caroline Kenny, Hantz Févry, Jonathan Endale, and hundreds of anonymous speakers. We would also like to thank Françoise Beaufays for the feedback and Angad Chandorkar, Golan Pundak, Shefali Garg, Nikhil Siddhartha, Zhouyuan Huo and Dongseong Hwang for helping develop the infrastructure used by our experiments.

2022 IEEE Spoken Language Technology Workshop (SLT) | 979-8-3503-9690-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/SLT54892.2023.10022938

2. RELATED WORKS

Despite constraints on computation, data retention, scheduling and power consumption, prior work has shown that embedded models can be personalised to a user with fine-tuning [11, 12, 10, 13], biasing [14, 15, 16], or fusion [17, 18, 19, 20]. We briefly discuss these techniques and their trade-offs.

2.1. Fine-tuning

Fine-tuning requires training, which can be difficult given limited power, time and memory, but has the advantage of only requiring constant work per example, and does not require training data to be stored once it has been consumed in training. Fine-tuning approaches can also be integrated into federated speech systems that improve future base models without collecting user data [21]. To reduce overfitting, processing and storage requirements, it is common to freeze most of the network, *e.g.* all but the early encoder layers [13] or late decoder layers [12] depending on the expected domain. Our approach depends on freezing the entire encoder, since its outputs are replaced with random numbers during training.

Prior fine-tuning approaches have naturally tended to use transcribed audio from the user or domain rather than text alone, which suffers from drawbacks mentioned in section 1. There have been successes in the use of text-only data to augment the training of non-personalized ASR models [22, 23, 24, 25, 26], especially in low-resource settings [27, 28, 29]. However, we are only aware of preliminary studies with partial success using this approach for on-device personalization [12], which maybe due to the expected difficulty of synthesizing adequate audio for this purpose on user devices. Our contribution is to bypass these difficulties by using random encoder features instead of text-to-speech.

2.2. Biasing

Biasing methods allow per-utterance conditioning on text-only contextual information, with trade-offs in precision, recall and model requirements. For example, finite state transducer (FST) based biasing has been studied in [14, 15] where contextual FSTs are composed with the ASR decoding search graph to boost the probability of recognizing salient phrases, such as contact names. More recently, neural associative memory [30] offers a model-based approach for incorporating contextual information to the encoder of an end-to-end ASR model using an attention mechanism. This approach was found to combine better with joint network fine-tuning for named entity personalization.

2.3. Language Model Integration

One way to use text-only data to improve an end-to-end speech recogniser is to replace or combine the text prediction network with some model trained on text-only data, as in *e.g.* [31, 32, 17, 33]. Compared to synthetic or random audio placeholder data, these approaches can work well in the presence of large text-only datasets, in exchange for extra computational resources required by the additional language model [34]. Similar to biasing, fusion approaches can combine well with fine-tuning on a small amount of domain-specific data due to their intervening on different parts of the model. The HAT architecture [35], used by most of our experiments as discussed in section 3.1, is designed for LM integration in a way that introduces more mathematical stability to previous fusion methods [18].

3. RANDOM ENCODER FEATURES FOR PERSONALIZATION

3.1. Background: RNN Transducer for ASR

The canonical RNN-transducer model [6] is comprised of three parts: an acoustic encoder, a text prediction network, and a "joint" network that combines their features to produce the final output. The audio encoder acts on the length-T sequence of audio input vectors $\mathbf{X}_{1:T} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$ to produce a sequence of encoded features $\mathbf{A}_{1:T} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_T\}$. The text predictor processes the length-K sequence of text labels $\mathbf{V}_{1:K} = \{v_1, v_2, ..., v_K\}$ into features $\mathbf{G}_{0:K} = \{\mathbf{g}_0, \mathbf{g}_1, ..., \mathbf{g}_K\}$. The joint network predicts the word-piece probabilities, $\mathbf{y}$, at every combination of indices k, t. Note that the sequence lengths K and T typically are not the same. These components are summarized in the equations:

$$\mathbf{A}_{1:T} \Leftarrow f_{\texttt{Audio}}(\mathbf{X}_{1:T}) \quad (1)$$

$$\mathbf{G}_{0:K} \Leftarrow f_{\texttt{Text}}(\mathbf{V}_{1:K}) \quad (2)$$

$$\mathbf{y}_{k,t} = f_{\texttt{Joint}}(\mathbf{g}_k, \mathbf{a}_t) \quad (3)$$

In the models used in our experiments, the encoder $f_{\texttt{Audio}}$ is a streaming conformer-based model with only left contexts [36]. As such, $\mathbf{a}_t$ encodes the acoustic information in $\mathbf{X}_{1:t}$, up to time t. Likewise, $\mathbf{g}_k$ generally encodes the label information in $\mathbf{V}_{0:k}$, up to the kth token. To promote efficient ASR inference, in our experiments $f_{\texttt{Text}}$ is simply an embedding consuming only the two most recent labels, which has been shown to be effective [18]. The $f_{\texttt{Joint}}$ component is a fully connected network ending in a 4,096-dimension softmax nonlinearity to produce the word-piece probabilities. The architecture of $f_{\texttt{Joint}}$ is further described in Section 4.4.

The RNN-T loss function [37, 38] operates over the probabilities from the different possible alignments between the encoder features and the language features. Alignments are visualized by the possible paths traversing the trellis from the $(\mathbf{g}_0, \mathbf{a}_0)$ state up and to the right, ending at the final label and audio features, as in the example in Figure 1. Steps horizontally consume an audio feature and produce a "blank" label

which is later removed from the text output. The likelihood of any final text hypothesis is aggregated over the multiple paths which produce it. Due to the large number of possible paths for sentence-length utterances, a beam search algorithm is typically used to approximate the solution without counting every path.

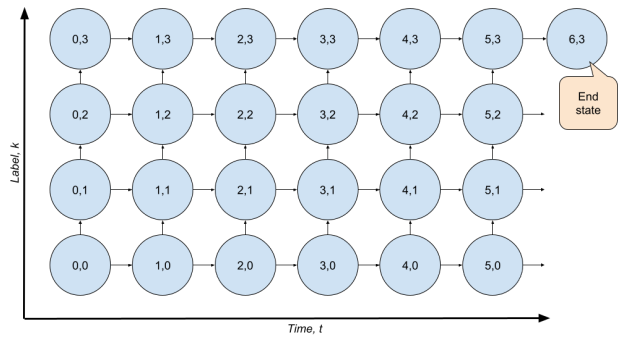

Fig. 1. Example RNN-T trellis depicting all possible alignments between the audio (time) sequence and the text (label) sequence.

Except where indicated otherwise, our experiments use the streaming hybrid autoregressive transducer (HAT) [18] modification to the RNN-T loss. A distinguishing feature of the HAT model is that it identifies an *internal language model* score which can be computed as $\mathbf{y}_k^{ILM} = f_{\texttt{Joint}}(\mathbf{g}_k)$ while using zero-valued features in place of the acoustic encoder output. Some of our experiments exercise this exact scenario. As suggested by our comparison to a non-HAT architecture (section 4.5), the HAT factorization may be especially conducive to training the internal language model of the ASR system using text-only data. Furthermore, while we do not presently incorporate any external language model, the benefits of our fine-tuning with random encoder features are best realised by updating weights in the joint network rather than the text prediction network (section 4.4), and therefore might remain even when replacing or augmenting the text prediction network.

3.2. Random Feature Characteristics

Our method substitutes random variables, $\tilde{\mathbf{A}}_{1:U} = \{\tilde{\mathbf{a}}_1, \tilde{\mathbf{a}}_2, ..., \tilde{\mathbf{a}}_U\}$, in place of the audio encoder output features during fine-tuning of the speech recognition model. Explicitly, the word-piece probabilities $\tilde{\mathbf{y}}$ are computed as:

$$\tilde{\mathbf{y}}_{k,u} = f_{\texttt{Joint}}(\mathbf{g}_k, \tilde{\mathbf{a}}_u) \tag{4}$$

Since $\tilde{\mathbf{a}}_u$ is random, it no longer encodes the past temporal information and can therefore be considered time-independent. The logits for each column of states in the RNN-T trellis in Figure 1 can be regarded as the same provided that the noise variance is sufficiently small. Consequently, every path in the trellis has the same vertical transition probabilities and the trellis can be loosely represented by a single-path trellis with K label (vertical) transitions followed by U *blank* (horizontal) transitions, shown in Figure 2. By optimizing the RNN-T loss using random encoder features, we are effectively optimizing the prior probability of the *internal language model* of the end-to-end model and the *blank* probabilities are proportionally weighted according to the length of the random encoder features, U.

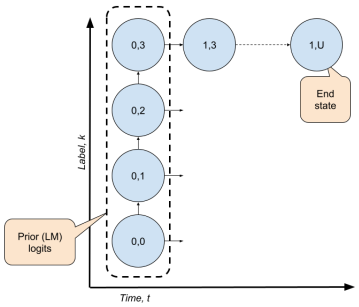

Fig. 2. A simplified RNN-T trellis depicting a single path that subsumes all possible alignments when random encoder features are used.

Two main characteristics of the random features must be chosen: the distribution of feature values, and the sequence duration, U. Figure 3 shows histograms of real audio encoder feature values measured from one of our models over a small representative dataset of spoken utterances. Panel 3(a) is aggregated over all 512 features, which produced a standard deviation of 0.06. The observed empirical distribution resembles the Gaussian approximation while holding slightly more weight in the peak and tails. We observed that individual features from our encoder exhibited differing distributions–panel 3(b) shows histograms for the individual features with the lowest and highest mean, while panel 3(c) shows the lowest and highest variance features. For simplicity, however, our experiments use the same distribution for every feature: normal with mean zero and a chosen scale, σ: $a_u^i \sim \mathcal{N}(0, \sigma^2)\ \forall u, i$.

With text-only data, we must choose the number of random feature frames without reference to any audio. While the sequence durations K and T for an utterance are different from each other, they are generally correlated. Therefore our approach is to scale the number of random feature frames according to the label length. For reference, the same representative dataset as above gave an average ratio of roughly four acoustic frames per text label. Starting from the basis point of emulating the observed encoder feature characteristics, our experiments examine learning performance over a range of random feature distributions and durations. We also train the ILM more directly using all zero encoder values.

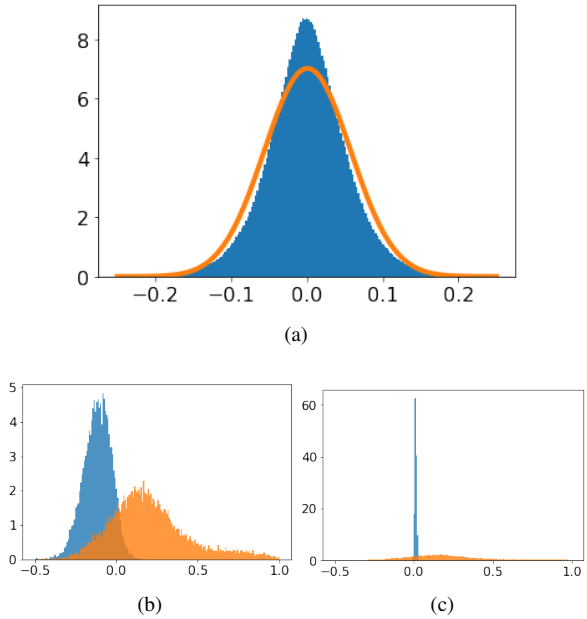

Fig. 3. Histograms of real encoder features values. (a) All features aggregated, with Guassian best fit; (b) Individual features with minimum and maximum mean values; (c) Individual features with minimum and maximum variance.

4. EXPERIMENTS

4.1. Dataset and Evaluation Metrics

Personalization performance is best measured by a combination of metrics taken on specific test sets, chosen to show both acquisition of new words and any potential catastrophic forgetting [10]. Our dataset is a follow-on version of *WikiNames* [12], comprised of transcribed speech recordings from 500 different speakers. Each speaker is assigned nine named entities which are known to be misrecognized by the base ASR model. Some examples include "Robert Stephenson", "Hedda Hopper", and "Rambhadracharya". Our experiments use name-only utterances for training, where the name is pronounced by itself, while the test set contains two full sentences for each name. We also include a *regular* test set of 15 different sentences per speaker which do not contain any misrecognized names. We use the *regular* set to identify any forgetting.

In addition to the overall word error rate (WER) metric, we include the keyword-dependent recognition metrics of *precision* and *recall* [12]. Precision is the ratio of correctly identified instances of a keyword to total hypothesized instances, and recall is the ratio of correct hypothesis instances to total reference instances. Low recall corresponds to lack of recognition ability for keywords, *i.e.* before they are learned. Precision becomes low when keywords incorrectly displace other vocabulary–an indication of forgetting.

For all experiments, we replicated the paradigm of Sim et al. 2021 [10], including the simulation of on-device continuous learning with the following techniques: updating the parameters with Adafactor [39]; quantising the parameters after each round for fast inference, and adding noise upon dequantisation to avoid rounding away the training updates; and using acceptance criteria to reject each new model if its loss or accuracy metrics regressed too much (in practice this was rarely the case for our small number of updates).

Our baseline model contains approximately 120 million parameters–it is compact and mobile device-friendly. The encoder includes 12 conformer layers and roughly 110 million parameters. Audio is pre-processed for input to the encoder as follows: 128-dimensional log Mel features are computed every 10 milliseconds, and 4 consecutive feature vector are stacked with a stride of 3 frames, yielding a 512-dimensional input feature every 30 milliseconds.

4.2. Name Recognition Experiment

Using the name-only utterances for training, we compared personalization with random encoder features against learning with real audio and learning with synthesized (TTS) audio. The TTS experiments used a US English Vocaine [40] voice that can run on low-end Android devices, which produces lower spectral quality and less natural prosody than larger server-based TTS engines. The results averaged over all 500 speakers are shown in Table 1, where it is seen that using random features ranks in between real audio (best) and TTS (worst) for learning. Starting from the baseline model, personalizing with random encoder features improved the test-sentence WER from 20.4% to 14.8%, and brought the recall up from 24.6% to 59.8%. This WER lies in between that measured using TTS training, 15.5%, and the best with real audio, at 13.3%. Random encoder features slightly outperformed the baseline model with FST biasing (15.1% WER). The hyperparameters for each method were tuned independently for the best performance on WER and recall while keeping the precision on the *regular* test set above 90%.

Encoder Features	**Test WER (%)**	**Test Recall (%)**	**Regular Precision (%)**
Baseline	20.4	24.6	97.4
+ FST Biasing	15.1	59.1	93.9
Real	13.3	75.5	92.9
TTS	15.5	60.6	90.3
Random	14.8	59.8	92.8

Table 1. Personalization with random encoder features performs better than using synthetic audio, greatly improving over the non-personalized baseline model; using real audio remains best.

The batch size was 9 (all names per speaker), and real and

TTS models trained for 6 epochs while random feature models trained for 4, all with learning rate 0.005. The random features scale was 0.03 and duration U was set as 6 times the length of the label sequence. New noise values are sampled for every model update. This result shows that the ASR model is able to acquire the new vocabulary to great extent from text-only data. In this experiment, only the joint network parameters were fine-tuned; this component contributes to the internal language modeling functionality of the E2E model.

4.3. Noise Distributions and Feature Lengths

We experimented with different noise distributions and feature lengths to find the best performance and to determine hyperparameter sensitivity. Table 2 contains results for several different cases.

In the first section of Table 2, we hold the length multiplier fixed at 1 and coarsely vary the noise scale, which reveals $\sigma = 0.05$ to be an effective value (16.0% WER). In the second section, we hold the noise scale fixed at 0.05 and vary the length multiplier; a wide range around 5-10 produced the best results (14.9% WER) (shorter lengths are favored for simpler computation on the RNN-T trellis). The table also includes results for our best measured hyperpameters: $\sigma = 0.03$ and length multiplier 6 (14.8% WER).

We include a sweep of length multipliers with all zero-valued features from the acoustic encoder ($\sigma = 0.0$). All nodes in the RNN-T trellis at the same label row, k, have the same input: $\mathbf{y}_{k,:} = f_{\texttt{Joint}}(\mathbf{g}_k, \mathbf{0})$, and we include the extreme "internal language model prior" case of sequence length $U = 1$ (multiplier=0). Learning is remarkably effective, reaching 15.1% WER for length multiplier 100. The fact that learning improves with increasing sequence length is likely due to interplay in the loss function between the sum over the increasing number of possible paths (which may be similar to increasing the learning rate), and the scaling of label probabilities by the inverse probability of a blank label (a design of the HAT factorization).

Finally, we experimented with an even simpler distribution for the random encoder features–the uniform distribution. After a coarse sweep of ranges, we found very good performance with uniform noise in the range $(-0.06, 0.06)$ and length multiplier 8 (14.7% WER), included in the last row of the table. It is a convenient quality of our method that it is not overly sensitive to the exact shape of the noise distribution; hence it is likely to work well for different encoder types or instances with different output distributions.

4.4. Trainable Variables

To further dissect the process of learning from random encoder features, we experiment with training different subsets of the network parameters during personalization. Our network architecture is shown in Figure 4, which depicts the input of random audio encoder features. The joint network consists of two projection matrices, W_A and W_T, which transform the features from the audio model (size 512) and text model (size 640), respectively, into vectors of dimension 640. The projections are summed with learnable bias $\mathbf{b}_h$ and a $tanh$ nonlinearity is applied. This hidden output feeds into a learnable projection parameterized by weight matrix W_o and bias $\mathbf{b}_o$ into the 4,096 word-piece dimension, followed by a softmax nonlinearity. The total number of trainable parameters in $f_{\texttt{Joint}}$ is 3.4M. Note that the HAT model [18] considers joint networks of the form $f_{\texttt{Joint}}(\mathbf{g_k}, \mathbf{a_t}) = f(\mathbf{g}_k + \mathbf{a}_t)$, which is equivalent to considering W_A and W_T as components of $f_{\texttt{Audio}}$ and $f_{\texttt{Text}}$ rather than $f_{\texttt{Joint}}$.

Noise Scale	Length Multiplier	Test WER	Test Recall	Regular Precision
0.0	1	16.2	53.0	96.2
0.01	1	16.2	53.4	96.2
0.05	1	**16.0**	**53.8**	95.1
0.1	1	16.5	50.0	95.1
0.05	0	16.9	48.4	97.5
0.05	1	16.0	53.8	95.1
0.05	5	**14.9**	58.3	90.6
0.05	10	**14.9**	**58.6**	91.7
0.05	25	15.1	57.8	91.7
0.0	0	16.9	49.7	96.2
0.0	1	16.2	53.0	96.2
0.0	10	15.3	57.1	95.0
0.0	100	15.1	57.8	92.7
0.03	6	14.8	59.5	92.8
Unif ($\pm$0.06)	8	**14.7**	**59.7**	90.6

Table 2. Hyperparameter sweeps over random encoder feature characteristics for personalization training, including best performing.

Key results are shown in Table 3. We found that training only W_T and W_o using random encoder features yielded almost the same benefit as training the entire joint network. Preserving the acoustic projection parameters, however, did not forestall the reduced precision on the regular dataset, which could have enabled further personalization fine-tuning. Training $f_{\texttt{Text}}$ by itself was much less effective. This is likely explained by the fact that our text model in this experiment is simply an embedding of the two most recent labels, making it difficult to learn any generalization here, despite that it contains roughly twice as many parameters as the joint network. Including the training of the text prediction network along with the joint network yielded slightly worse performance–altering the embedding further degraded precision on the reg-

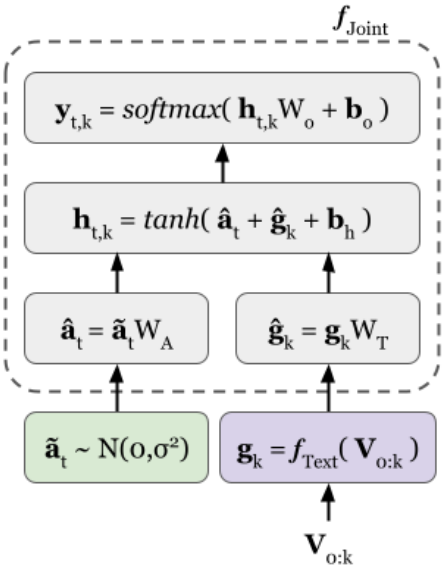

Fig. 4. Network architecture used in our experiments, including random audio encoder features, the text prediction network, and a breakout of the layers within the joint network.

ular test set.

We also experimented with freezing the audio feature projection, W_A, while training with real encoder features, which yielded decreased performance. This result indicates that learning in W_A is critical to achieve the full performance of personalizing with real audio. It is significant that training with random encoder features–which does not learn W_A–performs rather better than using real audio without learning W_A. An explanation is that the random encoder features induce more emphasis on learning the internal language model, since they replace and remove all helpful audio information.

AM Features	Trainable Layers	Test WER	Test Recall	Regular Precision
Random	$f_{\texttt{Joint}}$	**14.9**	**58.5**	90.6
	W_T	16.8	45.7	97.4
	W_T, W_o	**15.0**	**57.8**	91.7
	$f_{\texttt{Joint}}, f_{\texttt{Text}}$	15.9	54.4	91.7
	$f_{\texttt{Text}}$	19.2	32.2	97.4
Real	$f_{\texttt{Joint}}$	13.3	75.5	92.9
	$f_{\texttt{Joint}} \setminus \{W_A\}$	16.6	58.4	91.7

Table 3. Trainable variables: the effects of fine-tuning different subsets of network parameters. Each row's hyperparameters (scale and length for random encoder features, epochs and number of examples, 1 or 3, for real audio) optimized by coarse grid search.

4.5. Non-HAT Decoder

Finally, we tested an ASR model with a very similar architecture to the one used so far, but using the standard RNN-T loss instead of HAT. Summary results are shown in Table 4. The baseline model and the one personalized with real audio encoder features perform the same as in the previous sections. With random encoder features, however, we did not find similarly good performance. Notably, no learning occurs with noise scale 0.0. With $\sigma = 0.05$, excessive forgetting occurs (84.4% regular precision) without achieving as good a WER as the HAT model. The best results from our coarse hyperparameter search came with $\sigma = 0.1$ and length multiplier 10 (17.4% WER). It remains possible that alternative settings could perform better. We take these results as preliminary evidence that the HAT model, which includes an internal language model term and separate factorization of probability for blank labels, is especially conducive to learning with random encoder features. Other factorizations which more fully separate the blank-prediction model could yield improved ILM training with random encoder features.

Noise Scale	Length Multiplier	Test WER	Test Recall	Regular Precision
Baseline	-	21.3	24.3	97.4
Real Feat.	-	13.3	74.4	93.8
0.0	10	21.5	23.0	97.4
0.05	5	15.5	58.4	84.4
0.1	10	17.4	48.4	97.4

Table 4. Summary performance using the standard RNN-T loss rather than the HAT factorization.

5. CONCLUSIONS

We have shown that when the encoder layers of an end-to-end speech recognizer are frozen, on-device personalization can benefit from text-only data by using random encoder features as a placeholder for the missing audio during training. Using the Wiki-Names simulated user paradigm, we determined that the benefit is largely due to intervention on the text projection and hidden layers of the joint network, yielding behaviour consistent with updating the internal language model probabilities of the new vocabulary. As possible future advancements, our approach is compatible with biasing or external language model integration techniques, such as that enabled by the HAT model. Furthermore, it remains to be explored how training with real audio examples may be enhanced when used in combination with random encoder features.

6. REFERENCES

[1] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee, 2013, pp. 6645–6649.

[2] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, et al., "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4945–4949.

[3] Jinyu Li, Rui Zhao, Zhong Meng, et al., "Developing RNN-T Models Surpassing High-Performance Hybrid Models with Customization Capability," in *Proc. Interspeech 2020*, 2020, pp. 3590–3594.

[4] Tara N. Sainath, Yanzhang He, Bo Li, et al., "A Streaming On-Device End-To-End Model Surpassing Server-Side Conventional Model Quality and Latency," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. 2020, pp. 6059–6063, IEEE.

[5] Feng-Ju Chang, Martin Radfar, Athanasios Mouchtaris, and Maurizio Omologo, "Multi-channel transformer transducer for speech recognition," in *Interspeech 2021*, 2021.

[6] Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, et al., "Streaming end-to-end speech recognition for mobile devices," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[7] Vijay Ravi, Yile Gu, Ankur Gandhe, et al., "Improving accuracy of rare words for rnn-transducer through unigram shallow fusion," *ArXiv*, vol. abs/2012.00133, 2020.

[8] W. Ronny Huang, Cal Peyser, Tara N Sainath, et al., "Sentence-select: Large-scale language model data selection for rare-word speech recognition," *ArXiv*, 2022.

[9] Kanishka Rao, Hasim Sak, and Rohit Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer," *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 193–199, 2017.

[10] Khe Chai Sim, Angad Chandorkar, Fan Gao, et al., "Robust continuous on-device personalization for automatic speech recognition.," in *Interspeech*, 2021, pp. 1284–1288.

[11] Khe Chai Sim, Petr Zadrazil, and Françoise Beaufays, "An investigation into on-device personalization of end-to-end automatic speech recognition models," in *Interspeech*, 2019.

[12] Khe Chai Sim, Françoise Beaufays, Arnaud Benard, et al., "Personalization of end-to-end speech recognition on mobile devices for named entities," in *ASRU*, 2019.

[13] Katrin Tomanek, Françoise Beaufays, Julie Cattiau, et al., "On-Device Personalization of Automatic Speech Recognition Models for Disordered Speech," *arXiv e-prints*, p. arXiv:2106.10259, June 2021.

[14] Petar Aleksic, Cyril Allauzen, David Elson, et al., "Improved recognition of contact names in voice commands," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5172–5175.

[15] Keith Hall, Eunjoon Cho, Cyril Allauzen, et al., "Composition-based on-the-fly rescoring for salient n-gram biasing," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015, pp. 1418–1422.

[16] Aditya Gourav, Linda Liu, Ankur Gandhe, et al., "Personalization strategies for end-to-end speech recognition systems," in *ICASSP 2021*, 2021.

[17] Anjuli Kannan, Yonghui Wu, Patrick Nguyen, et al., "An analysis of incorporating an external language model into a sequence-to-sequence model," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, p. 1–5828, IEEE Press.

[18] Tara N. Sainath, Yanzhang He, Arun Narayanan, et al., "An Efficient Streaming Non-Recurrent On-Device End-to-End Model with Improvements to Rare-Word Modeling," in *Proc. Interspeech 2021*, 2021, pp. 1777–1781.

[19] Chao Zhang, Bo Li, Zhiyun Lu, et al., "Improving the fusion of acoustic and text representations in RNN-T," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 8117–8121.

[20] Theresa Breiner, Swaroop Ramaswamy, Ehsan Variani, et al., "Userlibri: A dataset for asr personalization using only text," 2022.

[21] Dhruv Guliani, Françoise Beaufays, and Giovanni Motta, "Training speech recognition models with federated learning: A quality/cost framework," in *ICASSP*

2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 3080–3084.

[22] Cal Peyser, Hao Zhang, Tara N. Sainath, and Zelin Wu, "Improving Performance of End-to-End ASR on Numeric Sequences," *arXiv e-prints*, p. arXiv:1907.01372, July 2019.

[23] Amin Fazel, Wei Yang, Yulan Liu, et al., "SynthASR: Unlocking synthetic data for speech recognition," in *Interspeech 2021*, 2021.

[24] Zhehuai Chen, Yu Zhang, Andrew Rosenberg, et al., "Injecting text in self-supervised speech pretraining," in *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021, pp. 251–258.

[25] Yi Ren, Xu Tan, Tao Qin, et al., "Almost unsupervised text to speech and automatic speech recognition," in *ICML*, 2019, pp. 5410–5419.

[26] Zhehuai Chen, Andrew Rosenberg, Yu Zhang, et al., "Improving Speech Recognition Using GAN-Based Speech Synthesis and Contrastive Unspoken Text Selection," in *Proc. Interspeech 2020*, 2020, pp. 556–560.

[27] Aleksandr Laptev, Roman Korostik, Aleksey Svischev, et al., "You do not need more data: Improving end-to-end speech recognition by text-to-speech data augmentation," in *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2020, pp. 439–444.

[28] Chenpeng Du and Kai Yu, "Speaker augmentation for low resource speech recognition," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7719–7723.

[29] Virender Kadyan, Hemant Kathania, Prajjval Govil, and Mikko Kurimo, "Synthesis speech based data augmentation for low resource children asr," in *International Conference on Speech and Computer*. Springer, 2021, pp. 317–326.

[30] Tsendsuren Munkhdalai, Khe Chai Sim, Angad Chandorkar, et al., "Fast contextual adaptation with neural associative memory for on-device personalized speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore, 23-27 May 2022*. 2022, pp. 6632–6636, IEEE.

[31] Shubham Toshniwal, Anjuli Kannan, Chung-Cheng Chiu, et al., "A comparison of techniques for language model integration in encoder-decoder speech recognition," *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 369–375, 2018.

[32] Suyoun Kim, Yuan Shangguan, Jay Mahadeokar, et al., "Improved neural language model fusion for streaming recurrent neural network transducer," *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7333–7337, 2021.

[33] Ding Zhao, Tara N. Sainath, David Rybach, et al., "Shallow-fusion end-to-end contextual biasing," in *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, Gernot Kubin and Zdravko Kacic, Eds. 2019, pp. 1418–1422, ISCA.

[34] Jinyu Li et al., "Recent advances in end-to-end automatic speech recognition," *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, 2021.

[35] Ehsan Variani, David Rybach, Cyril Allauzen, and Michael Riley, "Hybrid autoregressive transducer (hat)," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6139–6143.

[36] Anmol Gulati, Chung-Cheng Chiu, James Qin, et al., Eds., *Conformer: Convolution-augmented Transformer for Speech Recognition*, 2020.

[37] Alex Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[38] Tom Bagby, Kanishka Rao, and Khe Chai Sim, "Efficient implementation of recurrent neural network transducer in TensorFlow," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 506–512.

[39] Noam Shazeer and Mitchell Stern, "Adafactor: Adaptive Learning Rates with Sublinear Memory Cost," in *Proceedings of the 35th International Conference on Machine Learning*, Jennifer Dy and Andreas Krause, Eds. 10–15 Jul 2018, vol. 80 of *Proceedings of Machine Learning Research*, pp. 4596–4604, PMLR.

[40] Yannis Agiomyrgiannakis, "Vocaine: The vocoder and applications in speech synthesis," 04 2015, pp. 4230–4234.