Knowledge Distillation for Mixture of Experts Models in Speech Recognition

Felipe Cruz Salinas, Kenichi Kumatani, Robert Gmyr, Linquan Liu, Yu Shi

Microsoft

{ancruzsa,kekumata,rogmyr,linqul,yushi}@microsoft.com

Abstract

The sparsely-gated mixture of experts (MoE) architecture can scale out large Transformer models to orders of magnitude which are not achievable by dense models with the current hardware limitations, and have been proven to improve the convergence as well. However, many applications still rely on traditional dense models, for both deployment and further compression, in order to fit memory and latency requirements. In this work, we propose a simple approach to distill MoE models into dense models while retaining the accuracy gain achieved by large sparse models. This can be used for further optimization and compression using well-known techniques for dense models. We demonstrate the model compression efficiency of our knowledge distillation (KD) technique through multi-lingual speech recognition experiments. Experimental results show that our proposed method can reduce the number of weights of an MoE teacher network with almost the same accuracy from 677M to 99M for 24 experts and from 1.8B to 124M for 72 experts, respectively. The results also show that our KD method can provide better recognition accuracy compared to conventional training methods without the MoE teacher model.

Index Terms: Speech recognition, Sparsely-gated mixture of experts (MoE), Transformers, Knowledge distillation, Teacher-student learning

1. Introduction

Sparse Mixture of Experts (MoE) architectures [1], such as GShard [2] and more recently the Switch Transformer [3], have popularized the use of extremely large and sparse models for pre-training in the already ubiquitous Transformer architecture [4]. As noted in [3], these models not only scale better with respect to hardware, but also converge faster. The core idea of these models is to enforce sparsity by activating a subset of the fully connected layers (named as experts) on each Transformer block. This decision is controlled by a gating layer on each block. It has been recently shown in [5, 6, 7] that such large MoE models can achieve better accuracy in the field of Automatic Speech Recognition (ASR). Since Switch Transformer activates only one expert during inference, its computational complexity is comparable to that of a normal Transformer network with the single expert. Regardless of such efficient forward-pass computation, the MoE architecture still requires large memory for inference [8], which may not be suitable in many scenarios. In this work, we consider Knowledge Distillation (KD) to reduce MoE's memory usage.

KD is a technique to train smaller models (students) aided by well-trained large teachers, optimizing the student logits to follow the teacher outputs [9, 10, 11]. Since KD was proposed, multiple variants have been introduced for different types of dense models such as hybrid Hidden Markov Model (HMM) networks [10, 12, 13, 14], Transformer [4] and BERT architectures [15]. TinyBERT [16] and DistilBERT [17] are techniques developed to apply KD to these newer architectures in the NLP domain. They extend the original KD formulation for BERT models by distilling not only the output logits of the network but also intermediate attention and hidden states. They also combine the supervised loss with the distillation loss jointly. Sequence-level KD was also introduced in [18], which considers distillation for sequence-to-sequence problems. [18] address Neural Machine Translation (NMT) tasks, and proposes different levels to apply KD, word-level and sequence-level, which either optimizes the cross-entropy distillation loss over each element on the sequence or over the output of the entire decoded sequence via beam-search on NMT. The sequence-level KD algorithm has been also applied to ASR [19] and shown promising results. However, most prior work focuses on KD between teacher and student dense networks.

In contrast to normal practice, we build upon these techniques to apply KD from MoE models to dense models. This will allow us to train powerful sparse models as teachers and then deploy them efficiently via small dense models. The key observation is that the MoE model becomes the same as its dense counterpart. Based on such an observation, we re-use all the weights but the experts, which we distill into a single expert in the dense model.

We demonstrate effectiveness of our MoE KD technique through multi-lingual ASR experiments. We first build a large MoE Transformer model with multi-lingual speech data as a teacher [7]. Multi-lingual MoE knowledge is distilled into a dense student network by selecting one of the experts to minimize the KD loss. After MoE KD, we further fine-tune the student network with the supervised loss only.

The rest of this paper is organized as follows. In section 2, we briefly review the Switch Transformer architecture used for a teacher model. In section 3, we describe our knowledge distillation methods for the sequence-to-sequence Transformer. Section 4 describes multi-lingual ASR experiments on MoE to dense models. We conclude this work and describe our future plan in section 5.

2. Sparsely-gated mixture of experts (MoE)

The original MoE layer proposed by [1] consists of a weighted sum over k experts out of N as

$$y = \sum_{i \in \mathcal{T}} p_i(x) E_i(x), \tag{1}$$

where $\mathcal{T}$ is the set of the k expert indices. These indices are the top-k values of $p_i(x)$, which in turn is the Softmax function over the input token x projected by a router matrix W_r [3]. The Switch Transformer further simplifies this architecture by selecting only one expert instead of k, which reduces memory, communication and training instability. In addition, the Switch Transformer simplifies the gating loss that ensures $p_i(x)$ distributes the experts utilization evenly. More recent work has

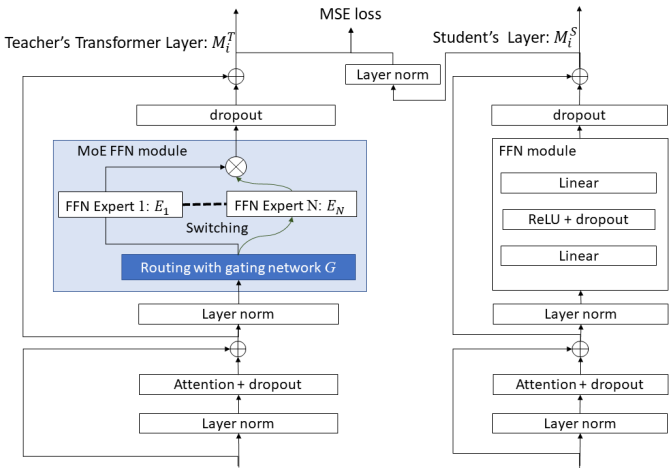


Figure 1: *Block chart of MSE loss for each Transformer layer*

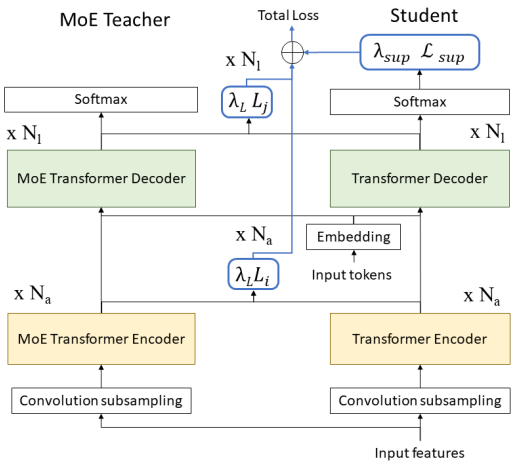


Figure 2: *Schematic diagram of total KD loss computation*

been done to improve the switch architecture [20, 21, 22]. In this work, we focus on the encoder-decoder network with the switch architecture [7, 23] but the same technique can be, in principle, applied to other similar models. The MoE technique is typically applied to every other layer in the Transformer. We also follow that scheme in this work.

3. Proposed MoE Knowledge Distillation

3.1. Transformer-Layer Distillation

We start from the key observation that on Switch Transformers [3], only one expert is activated on each forward pass for each token. In terms of FLOPS, this is approximately equivalent to the corresponding dense model with only one "expert" (fully connected layer in the Transformer). Our main idea is similar with the *Transformer-Layer* distillation approach proposed by TinyBERT [16]; we will exploit the structural similarities between the teacher and the student models.

Figure 1 illustrates a block chart of computing the KD loss (2) between the MoE teacher and student Transformer layer. As illustrated in figure 1, we only distill the Feed-Forwarding Network (FFN) of each transformer layer and the experts on the MoE architecture, but do not include prediction-layer and hidden or attention distillation explicitly. We also apply LayerNorm [24] before performing each layer distillation. We observed that layer normalization decreases the magnitude of some of the layer losses and helps the convergence. Finally, we include the supervised loss criterion on the final optimization as well, similar to [17], which we noticed helps the convergence.

Here, let us denote M_i^T and M_i^S as the teacher expert and student layer outputs for layer i. For each layer i, we use the Minimum Squared Error (MSE) loss functions:

$$L_i = \texttt{MSE}(M_i^T(x), M_i^S(x)) \tag{2}$$

This is applied to each Transformer layer as shown in figure 1. The teacher and student output will be closer at each layer through MSE loss optimization (2).

We then optimize the whole student network with the weighted MSE and supervised loss simultaneously. The total-loss criterion for KD can be expressed as

$$\mathcal{L} = \lambda_{sup}\, \mathcal{L}_{\text{sup}} + \lambda_L \sum_i L_i, \tag{3}$$

where $\mathcal{L}_{\text{sup}}$ is the supervised loss over the training set. Figure 2 shows a schematic diagram of computing the total KD loss between the MoE teacher and dense student network. Unlike prior work in hybrid ASR tasks [10, 13, 14], we observed in our preliminary experiment that the KLD distance on the logits degraded convergence performance with no accuracy gain. Thus, we do not employ the KLD loss in this work.

At first glance, it might seem that this method is not practical to load such big teacher models along student models for optimization. However, in practice, most of the memory goes to optimizer states due to tracking different statistics per parameter in methods like Adam [25], and also due to the weights duplication when using techniques such as mixed precision. Here, we only load the teacher in inference mode, so we need to track only the optimizer states for the student model, which is typically significantly smaller. However, we still need to apply inference over an teacher during KD, so we apply KD for a fixed number of steps and then continuously fine-tune the student over $\mathcal{L}_{\text{sup}}$. Finally, we apply these techniques over the same pre-training dataset used by the teacher, which helps to better guide the student and to produce a dense model which can be cheaply re-used for different downstream tasks.

3.2. Student initialization

As [17] notes, the student initialization method is important for the distillation procedure converge. We have found that it is critical to find good initial weights for each expert. The MoE-to-dense setup is convenient because all the student's non-expert weights can be initialized directly from the teacher, since there is not a change in the architecture layout. The expert weights, on the other hand, can be used directly from one of the experts for each layer in the network. Notice that the weights of the routing gates can be safely discarded.

[3] reports a few distillation experiments where student's non-expert weights are randomly initialized and a mixture of teacher and student probabilities is applied to the KD prediction loss. In this work, we propose to strategically select experts per layer according to a metric which can be modified depending on the task, so that we can initialize the student network with a fully-trained teacher expert. We use the utilization metric proposed in [26] for expert pruning, where we record how many tokens have been routed to an expert on the validation set; this is a computationally inexpensive way to measure which expert is favored over the others after pre-training. As we will see on

experimental results, this can be beneficial for layers where the expert utilization distribution is not completely uniform.

3.3. Increasing the dense model capacity

As we will see in the experimental results, we cannot compress so much capacity into an only single expert layer. As we increase the number of experts in the teacher, the retained quality of the model decreased. To overcome this issue, we propose to increase slightly the capacity of the student model, so that we can leverage more teacher experts while fitting in a certain computational budget. Typically, the feed-forwarding network (FFN) layer is defined by [4] as:

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{4}$$

We propose a simple change to introduce additional FFNs which can be initialized from the teacher directly. Such a modified FFN can be written as

$$\hat{\mathrm{FFN}}(x) = \sum_{i}^{K} \alpha_i \, \mathrm{FFN_i(x)}, \tag{5}$$

where K indicates the number of FFNs on a single layer and α_i is a learnable scalar that weights the output of each FFN.

This framework allows us to re-use a few teacher FFN weights, where typically K would be much smaller than the full amount of teacher experts. The definition above and the following experiments employ the popular FFNs used in [4], but any type of feed-forwarding module on the Transformer block can be used.

4. Experiments

4.1. Datasets

We conducted multilingual ASR experiments to verify the effectiveness of our KD method. The dataset used for training consists of 10 languages totaling approximately 75 thousand hours. As shown in Table 1, our training dataset is comprised of data from English, Spanish, Romanian, German, Italian, French, Portuguese, Dutch, Polish and Greek. It is worth noting here that we report the experimental results for multiple variants of English (US and UK), Spanish (ES-ES and ES-MX) and Romanian (RO). Table 2 tabulates the statistics of our test set for each locale: number of utterances and number of words. The dataset of each locale also contains not only various speakers but also different speech recognition tasks such as command-and-control tasks in mobile, office and car scenarios, Cortana phrases, dictation, conversational speech in telecommunication and so on. We consider the English test, Spanish test and Romanian test as a high-resource task, mid-resource task and low-resource task, respectively.

For training a network, we further split the whole training data set into training and validation sets in order to determine the convergence. The amount of training data for each language is different as shown in Table 1. We, thus, sample the lower resource data more frequently to balance the language data distribution during training. For all the experiments reported here, we used a 80-dimensional log filter-bank energy feature extracted at an interval of every 10*ms*. 10014 unique BPE [27] tokens are used for covering vocabulary of 10 languages.

4.2. Training configuration

For the multi-lingual teacher model, we use the sequence-to-sequence Transformer and replace the FFN layer with the MoE switch module every other layer [3]. The teacher model is trained with the multi-lingual data described in section 4.1; see [7] for the details on training the MoEs. Our baseline student model is composed of 18 Transformer blocks in the encoder and 6 blocks in the decoder, with 8 heads on each and 2048 dimension on the fully connected layer. This is the equivalent to the teacher network with the single FFN. As described in section 3.3, we will also consider the student network with the larger capacity. For the baseline, we train the multi-lingual student network from scratch for 500 thousand steps with the AdamW [28] optimizer by using a linear learning rate scheduler starting from 0.0 and peaking on 0.001252 on the first 31k steps. We use the similar training setting for KD; after updating the model for 30k steps based on the KD criterion (3), we further perform fine-tuning using the supervised loss only.

Language	Hrs
English (EN)	38585
Spanish (ES)	7584
German (DE)	4893
French (FR)	5955
Italian (IT)	6580
Portuguese (PT)	4302
Polish (PL)	2112
Greek (EL)	2190
Romanian (RO)	1899
Dutch (NL)	880

Table 1: *Duration of Training data: 10 language data*

Language	Code	No. utterances	No. words
English	EN-US	219965	1374748
	EN-UK	16743	79191
Spanish	ES-MX	15279	124742
	ES-ES	19275	150980
Romanian	RO	16626	365828

Table 2: *Test dataset details*

4.3. Results and discussions

We distill two MoE teacher models with 24 and 72 experts, respectively. For the larger teacher model, we increase the model capacity as described in section 3.3, using two FFNs instead of one FFN. The results in table 3 show the final WER on different language locale test sets for each training method. As a reference, table 3 shows the WERs of the MoE teacher models. It is clear from table 3 that our final distilled network can consistently provide better accuracy over all the locales than the dense baseline model trained from scratch in both cases. The results also suggest that as the teacher model has a larger number of experts, the student requires more network capacity to retain the teacher model quality. The final distilled students with 1 expert retain 97.98% and 91.03% of teacher's recognition accuracy for 24 and 72 experts, respectively. And the distilled student with 2-FFNs retains 96.85% WER from 72 experts teacher, which indicates that extra capacity was indeed required for the larger teacher model.

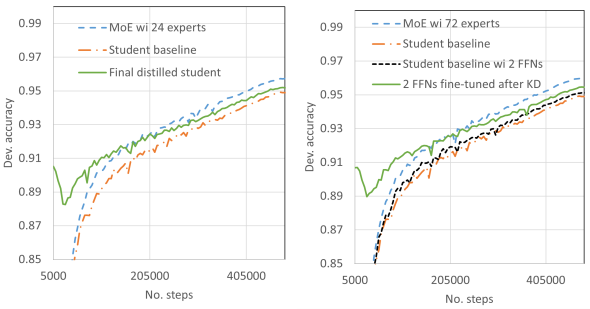
4.4. Analysis

In this section, we describe more detailed analysis on our KD performance.

Figure 3 plots the validation accuracy curves with respect to the number of the model updates for each training method in the case of a small teacher with 24 experts and large on with 72 experts. As a reference, figure 3 also shows the validation

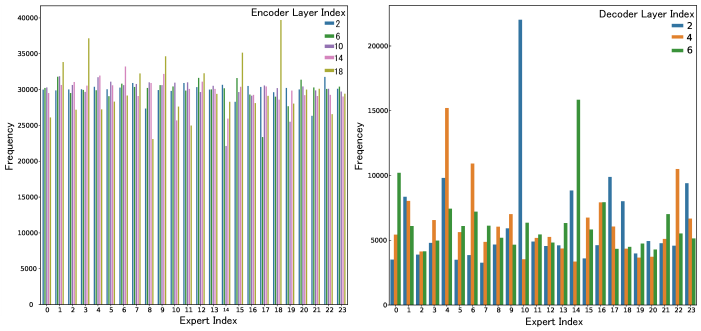
Model	Parameters	EN	RO	ES	ES-MX	Overall
Distilling from MoE 24 experts						
MoE Teacher	677M	9.66	18.97	14.18	14.78	11.69
Dense Baseline	98.9M	10.15	21.15	15.63	14.37	12.58
Final Distilled Student	98.9M	9.79	20.18	13.25	13.95	**11.93**
Distilling from MoE 72 experts						
MoE Teacher	1.8B	9.17	18.20	12.34	13.60	11.07
Dense Baseline	98.9M	10.15	21.15	15.63	14.37	12.58
Dense Baseline 2-FFN	124M	9.88	20.55	15.23	14.44	12.23
Final Distilled Student	98.9M	9.94	20.76	13.20	13.61	12.16
Final Distilled Student 2-FFNs	124M	9.50	18.97	12.42	13.20	**11.43**

Table 3: *Overall weighted WER over test 4 test sets in 4 language locales. The WER is weighted by the number of words, which is 1453939, 365828, 150980, and 124741 for EN, RO, ES, ES-MX, respectively.*



(a) *24 experts* (b) *72 experts*

Figure 3: *Token accuracy on the validation set during fine-tuning after KD*



(a) *Encoder expert frequency* (b) *Decoder expert frequency*

Figure 4: *Histogram of expert utilization during validation. Each bar represents the number of tokens routed to an individual expert on each layer during inference.*

accuracy curves of the MoE teacher networks. For figure 3, we computed the KD's accuracy curves after distilling for 30k steps. It is clear from figure 3 that our KD method provides a much faster convergence speed in the case of 24 and 72 experts. It is also clear from figure 3 that the KD can provide the better accuracy than the student baseline model trained from scratch. From figure 3, we can also observe a small drop after the first few iterations in fine-tuning the KD model. This is presumably attributed to the adaptation from the combined optimization of KD and supervised loss to the sole supervised loss optimization.

We finally investigate the frequency distribution of expert usage for the encoder and decoder. Figure 4 shows how many times each expert is selected out of 24 experts during inference on the validation data. Figure 4 (a) shows the expert usage frequency at the second, sixth, tenth, fourteenth and eighteenth layer in the encoder and figure 4 (b) shows that at the second, fourth and sixth layer in the decoder. It is clear from figure 4 that the expert utilization count has a peak on each layer, which are substantially bigger than other experts. We did notice this effect on the larger 72 experts model, but not so much on the encoder of both models. In fact, encoder's expert tends to have a more uniform frequency distribution than the decoder's expert. We hypothesize the decoder requires higher level specialization, hence a few experts activate more frequently for certain types of tokens. This phenomenon motivated the student initialization based on the expert usage, since an expert was more likely to be used during pretraining, it would be more likely to be useful as a starting point during the distillation phase.

5. Conclusion

We have described a new knowledge distillation technique to produce compact dense models out of large MoE models, which have proven recently to be a relatively straightforward to scale-up models both in size and quality. We have shown that our KD method can retrain 97.98% and 96.85% of teacher model's recognition accuracy for 24 and 72 experts, respectively. Our method can produce a compact dense model suitable for low-memory applications where is not feasible to load a model with a very high number of experts. In future work we would like to apply this method to other MoE applications to verify its effectiveness, and to distill larger models and develop possible techniques to cope with the GPU memory limitations.

6. Acknowledgements

The authors would like to thank Long Wu, Rui Jiang, Liyang Lu, Manthan Thakker, Peidong Wang, Tianyu Wu, Nayuki Kanda, Eric Sun, Wei Zuo, Devang Patel, Yao Qian, Jinyu Li, Saeed Maleki, Baihan Huang, Weixing Zhang, Jesse Benson, Tim Harris and Mengchen Liu for technical discussions. We also would like to thank Ed Lin, Michael Zeng, Xuedong Huang and Yuan Yu for their project support.

7. References

[1] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *ICLR*, 2017. [Online]. Available: https://openreview.net/pdf?id=B1ckMDqlg

[2] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "Gshard: Scaling giant models with conditional computation and automatic sharding," *arXiv preprint arXiv:2006.16668*, 2020.

[3] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *arXiv preprint arXiv:2101.03961*, 2021.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[5] Z. You, S. Feng, D. Su, and D. Yu, "SpeechMoE: Scaling to Large Acoustic Models with Dynamic Routing Mixture of Experts," in *Proc. Interspeech 2021*, 2021, pp. 2077–2081.

[6] ——, "SpeechMoE2: Mixture-of-experts model with improved routing," *CoRR*, vol. arXiv:2111.11831, 2021.

[7] K. Kumatani, R. Gmyr, F. C. Salinas, L. Liu, W. Zuo, D. Patel, E. Sun, and Y. Shi, "Building a great multilingual teacher with sparsely-gated mixture of experts for speech recognition," *CoRR*, vol. abs/2112.05820, 2021. [Online]. Available: https://arxiv.org/abs/2112.05820

[8] S. Rajbhandari, C. Li, Z. Yao, M. Zhang, R. Y. Aminabadi, A. A. Awan, J. Rasley, and Y. He, "DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale," *CoRR*, vol. abs/2201.05596, 2022. [Online]. Available: https://arxiv.org/abs/2201.05596

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[10] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size DNN with output-distribution-based criteria," in *Interspeech*, September 2014.

[11] S. D. Stanton, P. Izmailov, P. Kirichenko, A. A. Alemi, and A. G. Wilson, "Does knowledge distillation really work?" in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[12] J. H. M. Wong, M. J. F. Gales, and Y. Wang, "General sequence teacher–student learning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1725–1736, 2019.

[13] S. H. K. Parthasarathi, N. Sivakrishnan, P. Ladkat, and N. Strom, "Realizing petabyte scale acoustic modeling," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 422–432, 2019.

[14] L. Mošner, M. Wu, A. Raju, S. H. K. Parthasarathi, K. Kumatani, S. Sundaram, R. Maas, and B. Hoffmeister, "Improving noise robustness of automatic speech recognition via parallel data and teacher-student learning," in *Proc. ICASSP*, 2019, pp. 6475–6479.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[16] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "TinyBERT: Distilling BERT for natural language understanding," *arXiv preprint arXiv:1909.10351*, 2019.

[17] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[18] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," *arXiv preprint arXiv:1606.07947*, 2016.

[19] K. Kumatani, D. Dimitriadis, Y. Gaur, R. Gmyr, S. E. Eskimez, J. Li, and M. Zeng, "Sequence-level self-learning with multiple hypotheses," in *Interspeech*, October 2020.

[20] M. Lewis, S. Bhosale, T. Dettmers, N. Goyal, and L. Zettlemoyer, "Base layers: Simplifying training of large, sparse models," *arXiv preprint arXiv:2103.16716*, 2021.

[21] A. Yang, J. Lin, R. Men, C. Zhou, L. Jiang, X. Jia, A. Wang, J. Zhang, J. Wang, Y. Li *et al.*, "Exploring sparse expert models and beyond," *arXiv preprint arXiv:2105.15082*, 2021.

[22] S. Roller, S. Sukhbaatar, A. Szlam, and J. Weston, "Hash layers for large sparse models," *arXiv preprint arXiv:2106.04426*, 2021.

[23] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4960–4964.

[24] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] Y. J. Kim, A. A. Awan, A. Muzio, A. F. C. Salinas, L. Lu, A. Hendy, S. Rajbhandari, Y. He, and H. H. Awadalla, "Scalable and efficient MoE training for multitask multilingual models," *arXiv preprint arXiv:2109.10465*, 2021.

[27] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. [Online]. Available: https://aclanthology.org/P16-1162

[28] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.