Investigating Slimmable networks for transformer-based models in ASR

Sandeep Kumar
Université de Montréal
sandeep.kumar.1@umontreal.ca

Vamsikrishna Chemudupati
Université de Montréal
vamsikrishna.chemudupati@umontreal.ca

Abstract

We investigate the utilization of slimmable neural networks in the domain of automatic speech recognition for transformer-based architecture. We show the feasibility of training a single model which can be reconfigured at runtime to adapt to various scenarios. Particularly speaking we can vary the width hyper-parameter to reconfigure the model and show that it is possible to have a great accuracy-vs-latency trade-off in speech recognition tasks. In the case of edge devices where the compute capability is quite low and can vary, this technique can mitigate many difficulties. This method solves the challenge of re-training, re-deploying and re-download the model every time a new edge device utilizes it by using a single configurable model. We perform our experiments using a Transformer-Cnn based encoder(Conformer) and the Librispeech dataset, demonstrating a similar performance (usually slightly better) of the single, conformer-based RNN-Transducer model as compared to fixed width separate networks.

1 Introduction

In today's world machine learning and artificial intelligence is present in nearly every field. With each passing day, newer use cases are being identified where the power of deep neural networks can be utilized. With the advancements in the hardware sector and the increase in the computation capabilities of industrial hardware, it is now possible to build much larger models than ever before. Some models nowadays even have reached half-trillion parameter[3] mark, showcasing the potential of commercial hardware that is available.

Nowadays most of the world is going mobile-first in its approach. It means the consumer tech sector is now dominated by compact and portable devices. Devices like smartphones and smartwatches have become so ubiquitous that it is hard to imagine your life without one. Utilizing machine learning advancements was a natural step forward for multiple features in these devices like face detection in cameras, background noise reduction during calls, applying filters to modify images etc. When customers use these AI applications they expect the system to be smooth and responsive. Now the challenge arises as the devices in concern not only have limited computation resources available but are also running on batteries so the energy aspect also has limitations. It would not be surprising to say that these bulky state-of-the art models are not going to run smoothly on these devices and provide the experience that the users expect. It is not the case that we simply attach some high-end GPUs to inference at the edge for these devices. There is still a way of using APIs to access the industry-grade hardware in the cloud but the overall turn-around time for the network requests make this approach a non-viable option due to the real-time response requirement of many application. To solve this challenge many strides were made in this direction in the past. Researchers created specific architectures like mobilenet[13], shufflenet [35] and mobilenet-v2 [23] that were quite efficient and light-weight by design. There were approaches of utilising Neural Architecture Search (MNASnet[25]) for searching better architectures. Lower com-

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

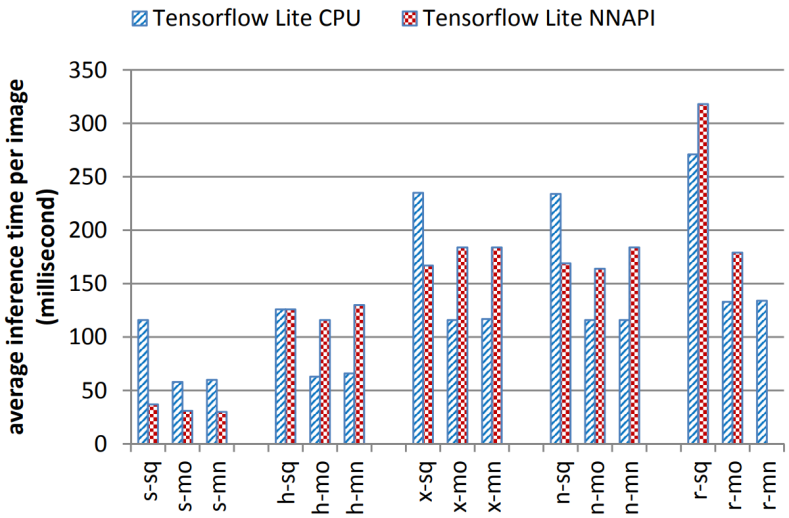

Figure 1: The average inference time of the light-weight networks with and without NNAPI delegate. (s: Galaxy s10e, h: Honor v20, x: Vivo x27, n: Vivo nex, r: Oppo R17, sq: SqueezeNet, mo: MobileNetV2, mn: MnasNet.) Source : [18]

putation complexities and small memory footprint made these models ideal to be deployed at the edge.

With the above-mentioned approaches making a way forward for on-device machine learning inference a reality, another challenge came into the picture. As we know not all mobile devices are equal in terms of hardware capabilities resulting in different performance levels. Usually, the high-end devices contain the latest SoCs(System-on-a-Chip) capable of handling bigger networks whereas the average mobile phones contain a lesser capable variant. Along with this, it may be the case for Android-based mobile devices that the chip manufacturer is different resulting in multiple possible variations in mobile device hardware. Therefore if we deploy the same model on all of these devices, the differences in the raw power of the hardware would most likely favour the high-end spectrum of these devices. These devices with their significantly better chips would be able to run bigger models and get better accuracy performance while the lower tier devices will suffer.

As we can see from the above Fig1 running the same networks on different devices leads to different inference times. This shows that differences in manufacturer and hardware can cause significant variation when it comes to running these models on edge. So there is a need to adjust the models for each device so that it can run in a set inference time budget in-order to provide a consistent and smooth experience to the user. Fortunately most of the efficient networks do possess a hyper-parameter called width which can control the parameter count of the networks and make it smaller. This can provide the much-needed accuracy-vs-latency trade-off for making it work on different devices. But, this hyper-parameter is fixed at the start of the training process and thus doesn't allow us to reconfigure the network at runtime without the need to retrain and re-deploy the models. This need for runtime reconfiguration resulted in the Slimmable Networks [33] which allowed the user to change the widths of these networks to match the set inference time budget with just one model being trained and deployed.

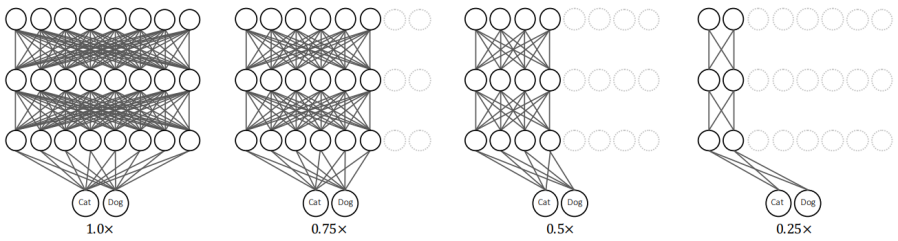

Figure 2: Illustration of slimmable neural networks. The same model can run at different widths(number of active channels), permitting instant and adaptive accuracy-efficiency trade-offs. Source : [33]

The main goal of this project is to utilize the concept of slimmable networks and extend it to the automatic speech recognition domain. There are multiple services that use speech recognition to

do tasks on mobile devices. So, there is a need for configurable speech recognition models that can function on a wide array of devices with acceptable quality in real time. For this purpose we are utilizing the Rnn-Transducer(RNN-T) [7] based architecture but for the encoder block, we are using one of the newer transformer based encoder called Conformer [8].

2 Related Work

In the past, there have been multiple approaches to solve the problems regarding the deployment of neural networks on the edge devices such as mobile phones. Edge devices with limited computation budget presented a unique challenge that required new approaches to build efficient networks for these devices. These approaches usually utilized techniques like distillation and pruning.

Knowledge distillation or distillation is the process of transferring the knowledge of a larger model into a smaller model. This technique was first proposed by Hinton et al.[12] and knowledge was distilled using the logits of larger network to smaller network. Since then there were further strides made in this field. One of the newer techniques by Tung et al.[27] works by matching the similarity of the student to that of the teacher in its feature space. Another, recent techniques by Peng et al.[22] showed kernel-based correlation congruence to distill knowledge. The authors measured the correlation metric of paired instances using a kernel function. Some techniques even delved into contrastive learning and presented a new approach called CRD [26] to transfer knowledge. Chen et al.[2] utilized the knowledge review aspect in their distillation technique. Yang et al.[30] proposed knowledge distillation using feature recovery techniques. The student was asked to generate the teacher's feature representation given same input but with some random masking in the students generated features. Zhao et al.[36] modified the conventional kl divergence loss used for distillation for target and non-target classes.

Pruning as the name suggests is used to prune the neural network model i.e. removal of non-informative parts of heavy networks. For Pruning techniques there are two ways to implement them. First one is weight based method which are also sometimes called sparsity-based methods. In these methods parameters or network links are removed. Recently, Han et al.[9] introduced removal of connections where the values don't meet a specified threshold. In [19] the authors propose a sparse evolutionary training of ANNs. Frankle et al. [5] show that there exist many sub networks in a over parameterized network that can provide similar test time performance showing that it is feasible to make a sparse network which has same capability as the original network. Second way of pruning networks involves eliminating least-important neurons or filters and are usually called as Unit based methods. He et al. [11] proposed a simple node removal technique where the importance of a neuron is assessed and then unimportant nodes are removed from network. Li et al. [14] proposed pruning based on absolute weighted sum. Liu et al. [15] proposed to prune based on mean gradient of the feature maps. Recently Liu et al. [16] utilized a meta learning based approach with an auxilary network to apply pruning on the main network.

The tradeoff between model size and accuracy has been an active research area for a long time now. Zagoruyko and Komodakis [34] studied the effect of width hyper-parameter in the case of the famous ResNet architecture [10]. Working on re-configuring this width parameter dynamically Yu et al. [33] designed slimmable neural network. They developed various switchable layers to be used instead of normal layers to support the dynamic width aspect of the network. This provided the functionality to train a single model functioning at multiple width levels (and significant parameter differences in each of the operating widths). In the further advancement of slimmable networks Yu et al. [32] provided an extension to this. This work mainly focused on improving the training techniques associated with slimmable networks. The author utilized techniques like distillation in conjunction with the re-configurable width aspect to achieve even higher accuracy levels than the original method, They also proposed the sandwich rule for performance levels of the slimmable networks.

Coming to the ASR aspect, we can classify the ASR models into two types, streaming based and full context based models. Streaming models are autoregressive in nature while the full context based model uses tokens of both past and future for prediction. Recently Yu et al. [31] utilized showed dual modes in one model utilizing the in-place distillation and achieving state of the art. Another interesting advancement was in the field of encoders where Conformer [8] proposed

use of convolution with transformer [28] based architecture to combine the content based global interactions given by a transformer and local features given by CNN. Another recent work, by Wu et al. [29] focusses on dynamically configurable networks similar to slimmable networks and involves pruning the network to introduce sparsity in it. CoDERT [24] by Swaminathan et al. also utilizes in-place distillation for distilling the Encoder outputs with complete seq-to-seq model outputs to achieve better word error rates on Librispeech dataset [20].

3 Architecture

The main goal of this project was to explore the application of slimmable networks in the field of automatic speech recognition particularly in the case of transformer-based encoder. For this purpose we selected to use RNN-T [7] model with Conformer [8] based encoder module. As we know in end-to-end automatic speech recognition methods the encoder takes up the bulk of the trainable parameters making it the biggest tunable component in the whole architecture hence the goal was to make the encoder slimmable. RNN-T model has three sub-parts an encoder, a predictor and a joint network. In our implementation for the encoder component we have 16 Conformer Blocks, the predictor network is made of 2 LSTMs and the joint network is simply a linear layer.

3.1 Conformer architecture

As we are mainly focussing on the Encoder, the conformer block consists of the following layers:

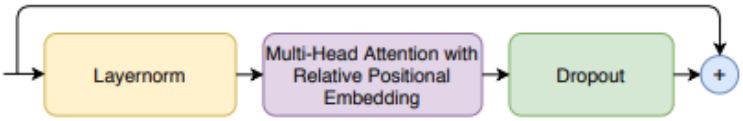

Figure 3: Illustration of multi-head self attention module in Conformer Block. Source : [8]

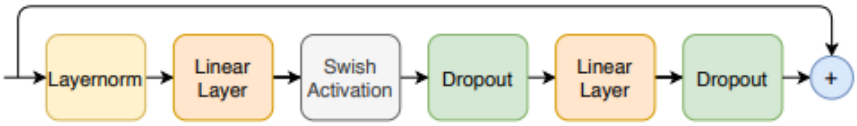

Figure 4: Illustration of feed-forward module in Conformer Block. Source : [8]

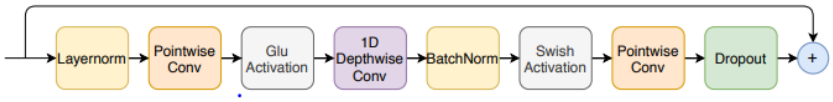

Figure 5: Illustration of convolution module in Conformer Block. Source : [8]

- **Multi-headed self-attention module** : Multi-headed self-attention module which integrated with relative positional encoding scheme used in Transformer-XL [4]. The relative positional encoding helps in making the model robust to varying utterance lengths. A layernorm and dropout layer is used to facilitate training and regularization in deeper networks. For details refer Fig3

- **Convolution module** : Consists of 1-d, Depth-wise convolution, point-wise convolution, Gate linear unit activation function and Dropout layers. Utilizes less parameters than conventional models due to point-wise and depth-wise convolutions. Batchnorm aids the training of model. For details refer Fig5
- **Feedforward module** : The feed-forward module has two fully connected layers and an activation function, placed after the attention module following the transformer architecture. For details refer Fig4

The complete architecture is given in the figure below

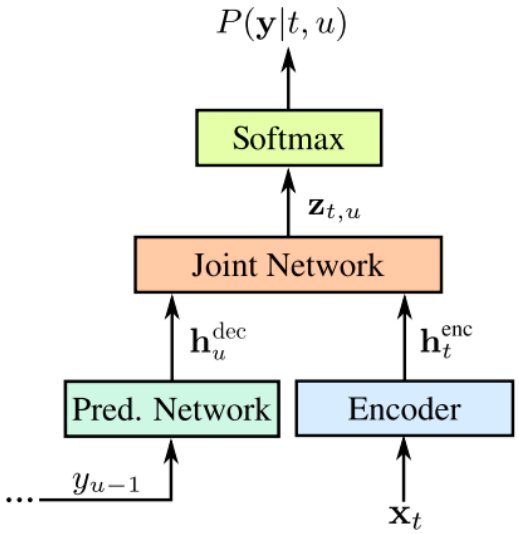

Figure 6: RNN-T model. Source :[6]

3.2 Slimming techniques applied to Conformer block

- **Linear/Dense layer**: We reduce the number of channels or neurons in a layer based on the width being used. For a 0.75 switch, we only use the 75% of neurons available in the layer for feed-forwarding and backward propagation methods. Similar in the case of other switches and for the 1.0 switch we use the complete layer.

- **Convolution layer**: A similar approach is followed as a dense layer and we reduce the number of feature maps or kernels in the layer based on the switch we are using.

- **Multihead-attention layer**: In this layer, we can scale either by reducing the number of attention heads or reducing the attention head size. For a conformer block we have 4 attention heads and 64 as each head size, therefore for a 0.75 switch if we scale the number of heads we will be using 3 heads and 64 as the head size. However, if we scale each head size then for a 0.75 switch we will use 48 as the head size with 4 heads. Both methods were explored and the results show a similar performance.

- **Normalization layers**: Normalization layers are mainly layers such as Batch-normalization(Batchnorm) and Layer-normalization. In this case, multiple batchnorms are required due to moving average mean and variance calculation used during inference. They are calculated using the number of channels in the previous layer. Different channels due to a slimmable training loop make the moving average and variance unstable during inference. Hence it is preferred to have batch-norm for each scale. We follow a similar design principle for layer-norm as well.

4 Experiments

This section provides a comprehensive explanation of Datasets, Experimental Setup, Model training and Evaluation procedures.

4.1 Datasets

Librispeech corpus is derived from Audiobooks that are a part of the LibriVox project and contain 1000 hours of speech sampled at 16kHz[21]. The corpus is split into three sets named train, test and dev and can be used for training and evaluating ASR models. The training portion of the corpus is split into three subsets with sizes of 100,360 and 500 hours respectively due to its huge size as a single file which makes it impractical for experimentation in low compute environments. We used train-clean-100 as a training set which contains 100 hrs of US English audio extracted from 251 different speakers. For evaluation purposes, we use dev-clean and test-clean as the validation and testing datasets extracted from the same corpus.

We pre-process the raw audio and perform feature extraction by converting it to 80-channel log magnitude Mel spectrogram representation computed on 25 millisecond windows with a stride of 10 milliseconds. Further, we perform feature normalization by scaling the input representations between 0 and 1 with approximately zero mean and unit standard deviation across the training dataset. We perform the above data transformations for train, valid and test dataloaders as part of the data processing pipeline and the resultant normalized feature representations are given as input to the ASR Model.

4.2 Evaluation Metrics

The evaluation metric used for all the experiments is Word error rate(WER). Word error rate is actually the measure of how well an ASR system performs in transcribing speech audio to text[1]. It calculates the errors in the transcription based on the substitutions, deletions and Insertion observed. The lower the value, the better the ASR system with the maximum performance being WER 0. Mathematically it can be calculated as:

$$WER = \frac{substitutions + deletions + insertions}{numberofwords}$$

4.3 Training Setup

An RNN-T model is used for experimentation and evaluating Slimmable approaches in the ASR domain. The model architecture consists of three components Encoder, Predictor and Joint neural networks. The trainable parameters and layer details of each component are given below:

- **Encoder**: 16 Conformer Blocks with 24M parameters.
- **Predictor**: 2 LSTM layers with 4M parameters.
- **Joint**: 1 linear layer with 1M parameters.

Based on the parameter count above, we understand that Encoder is regarded as the major element of the ASR system and has the highest number of trainable parameters. It is responsible for performing feature encoding of input audio signals. The previous research work in the speech domain has been mainly focused on improving the Encoder performance which results in an overall performance improvement of the RNN-T system. Therefore, for all the experiments we will be applying slimming techniques only to the Encoder component and focus on improving its performance over multiple scales.

The goal of the experiments is to evaluate whether we can achieve a configurable ASR model using Slimmable networks. We consider 4 switches 0.25,0.5,0.75 and 1.0 as operating points where 0.25 configuration suggests that 25% of the parameters in each layer of the model are being utilized for performing feed-forward and back-propagation. We jointly train a single configurable Slimmable network with 4 different scales and perform inference with each scale during run-time. The jointly trained model is expected to work as a single model equivalent to 4 individually trained models of

scales [0.25,0.5,0.75,1.0] which states the purpose of slimmable networks. To verify the performance we consider four individually trained models as baseline models and use them for comparison with a jointly trained slimmable network. The individually trained models are configured with the same width as the 4 scales being used for evaluation and represented as dedicated models further in the report.

The code for training and data pre-processing was implemented using PyTorch. The batch size used for training slimmable networks with four switches is three times less than the individually trained model to accommodate the gradients of the sub-networks as well. The training time for slimmable networks is three times the individual model training time mainly because of the updates made to the sub-networks. We use four V100 Gpus of 16GB each for performing all the experiments in this project.

4.4 Training

In this section, we would be understanding the training methodology for Slimmable networks. We perform joint training of the complete RNN-T network and sub-networks at various pre-defined switches. During inference, we use the network at a particular switch and it can be configured dynamically at run-time. The objective loss function used for training is RNN-T loss and it is

Algorithm 1: Slimmable networks training algorithm

Define the switches for the network **[0.25,0.5,0.75,1]**;
Initialize 4 independent batchnorm and layernorm parameters;
for $i = 1.....n_{epochs}$ **do**
  Mini-batch of x audio samples and y labels;
  Start Gradient calculation;
  for *scale in switchable width list* **do**
    Select layer configuration for network M based on the scale;
    Feed-forward operation using the sub-network $M^{'} - y^{'} = M^{'}(x)$;
    Compute the RNN-T loss using the predicted output and true label;
    Compute the gradients for sub-network $M^{'}$;
  end for
  Update weights for the complete network M
end for

similar to maximizing the negative log-likelihood. The transducer defines $p(y|x)$ as the sum of the probabilities of all possible alignments between x and y where x is the input latent representation of an audio frame and y is the output text label of the model [17]. We can write it mathematically as:

$$Loss = -\log p(y|x)$$

For Jointly trained slimmable networks we calculate the training loss as the sum of RNN-T losses computed at different switches of the network

$$TrainingLoss = Loss_{k1} + Loss_{k2} + Loss_{k3} + Loss_{k4} + ... + Loss_{kn}$$

where k1, k2, k3 .... kn are the switches of slimmable networks.

The calculated loss is then used for updating the parameters of the network using backward propagation. A detailed step-by-step explanation of the slimmable networks training algorithm is given in Algorithm 1.

5 Results

We investigate slimmable techniques in ASR models using RNN-transducer model with 4 switches [0.25,0.5,0.75,1.0]. The four sub-networks have model parameters of sizes 7M, 10M, 17M and 30M respectively which shows models ranging from low latency to high latency. As shown in Table 1, the slimmable models outperform their respective dedicated models depicting that the slimmable network

is a single configurable model which is equivalent to four models of varying sizes. The reason for slimmable networks performing better is due to the usage of a objective function which is calculated using a sum of rnnt losses from different switches. Joint training accelerates the convergence and also provides regularizing effect for the model due to the presence of sub-networks. Also the WER values signify the fact that the model with more number of parameters works better following the scaling laws. Hence we can confirm that Slimmable networks provide us with an configurable ASR model with reasonable WER/number of parameters trade-off.

Table 1: Results of Slimmable model with 4 switches [0.25,0.50,0.75,1.00] compared with dedicated baselines

Switches	Model types	Num. Params	WER
0.25	Slimmable	7M	**16.34**
0.50	Slimmable	10M	**12.57**
0.75	Slimmable	17M	**11.88**
1.00	Slimmable	30M	**11.40**
0.25	Dedicated	7M	16.45
0.50	Dedicated	10M	13.08
0.75	Dedicated	17M	12.38
1.00	Dedicated	30M	11.94

6 Conclusion

In this project, we study slimmable networks by applying them to the Conformer based RNN-T model and testing against the Librispeech dataset. The results show that Jointly trained slimmable models outperform the individually trained models across all switches. We can confirm that Slimmable networks provide us with a configurable ASR model with reasonable WER/number of parameters trade-off. Hence, we can say that slimmable networks are a solution for deploying large-scale models in environments with different compute capabilities ranging from mobile devices to servers using a single model which can be configured during run-time. We also analysed the advantages of slimmable training by slicing an individually trained network. The study reveals that representations collapse without slimmable training and it is required to have multiple configurations. The single model is capable of handling accuracy/latency trade-off requirements and eliminates the need for different model sizes and corresponding hyperparameter tuning. After deploying, a slimmable network reduces response time by dynamically shifting from one switch to another rather than following the procedure of downloading and uploading different models. For future work, we would be exploring the self-distillation techniques in slimmable networks. Also, we would be varying the number of switches, and lower bound of the switch to observe the performance differences.

References

[1] Word Error rate. `https://www.assemblyai.com/blog/word-error-rate/`, 2023.

[2] CHEN, P., LIU, S., ZHAO, H., AND JIA, J. Distilling knowledge via knowledge review. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 5008–5017.

[3] CHOWDHERY, A., NARANG, S., DEVLIN, J., BOSMA, M., MISHRA, G., ROBERTS, A., BARHAM, P., CHUNG, H. W., SUTTON, C., GEHRMANN, S., ET AL. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

[4] DAI, Z., YANG, Z., YANG, Y., CARBONELL, J., LE, Q. V., AND SALAKHUTDINOV, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860* (2019).

[5] FRANKLE, J., AND CARBIN, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* (2018).

[6] GOOGLE RESEARCH BLOG. An all-neural on-device speech recognizer, 2019. [Online; accessed May 08, 2023].

[7] GRAVES, A. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711* (2012).

[8] GULATI, A., QIN, J., CHIU, C.-C., PARMAR, N., ZHANG, Y., YU, J., HAN, W., WANG, S., ZHANG, Z., WU, Y., ET AL. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100* (2020).

[9] HAN, S., POOL, J., TRAN, J., AND DALLY, W. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems 28* (2015).

[10] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[11] HE, T., FAN, Y., QIAN, Y., TAN, T., AND YU, K. Reshaping deep neural network for fast decoding by node-pruning. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2014), IEEE, pp. 245–249.

[12] HINTON, G., VINYALS, O., AND DEAN, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[13] HOWARD, A. G., ZHU, M., CHEN, B., KALENICHENKO, D., WANG, W., WEYAND, T., ANDREETTO, M., AND ADAM, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[14] LI, H., KADAV, A., DURDANOVIC, I., SAMET, H., AND GRAF, H. P. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* (2016).

[15] LIU, C., AND WU, H. Channel pruning based on mean gradient for accelerating convolutional neural networks. *Signal Processing 156* (2019), 84–91.

[16] LIU, Z., MU, H., ZHANG, X., GUO, Z., YANG, X., CHENG, K.-T., AND SUN, J. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 3296–3305.

[17] LUGOSCH, L. Sequence-to-sequence learning with transducers, Nov 2020.

[18] LUO, C., HE, X., ZHAN, J., WANG, L., GAO, W., AND DAI, J. Comparison and benchmarking of ai models and frameworks on mobile devices. *arXiv preprint arXiv:2005.05085* (2020).

[19] MOCANU, D. C., MOCANU, E., STONE, P., NGUYEN, P. H., GIBESCU, M., AND LIOTTA, A. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications 9*, 1 (2018), 2383.

[20] PANAYOTOV, V., CHEN, G., POVEY, D., AND KHUDANPUR, S. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)* (2015), IEEE, pp. 5206–5210.

[21] PANAYOTOV, V., CHEN, G., POVEY, D., AND KHUDANPUR, S. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2015), pp. 5206–5210.

[22] PENG, B., JIN, X., LIU, J., LI, D., WU, Y., LIU, Y., ZHOU, S., AND ZHANG, Z. Correlation congruence for knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 5007–5016.

[23] SANDLER, M., HOWARD, A., ZHU, M., ZHMOGINOV, A., AND CHEN, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 4510–4520.

[24] SWAMINATHAN, R. V., KING, B., STRIMEL, G. P., DROPPO, J., AND MOUCHTARIS, A. Codert: Distilling encoder representations with co-learning for transducer-based speech recognition. *arXiv preprint arXiv:2106.07734* (2021).

[25] TAN, M., CHEN, B., PANG, R., VASUDEVAN, V., SANDLER, M., HOWARD, A., AND LE, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 2820–2828.

[26] TIAN, Y., KRISHNAN, D., AND ISOLA, P. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699* (2019).

[27] TUNG, F., AND MORI, G. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 1365–1374.

[28] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł., AND POLOSUKHIN, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[29] WU, Z., ZHAO, D., LIANG, Q., YU, J., GULATI, A., AND PANG, R. Dynamic sparsity neural networks for automatic speech recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2021), IEEE, pp. 6014–6018.

[30] YANG, Z., LI, Z., SHAO, M., SHI, D., YUAN, Z., AND YUAN, C. Masked generative distillation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI* (2022), Springer, pp. 53–69.

[31] YU, J., HAN, W., GULATI, A., CHIU, C.-C., LI, B., SAINATH, T. N., WU, Y., AND PANG, R. Dual-mode asr: Unify and improve streaming asr with full-context modeling. *arXiv preprint arXiv:2010.06030* (2020).

[32] YU, J., AND HUANG, T. S. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 1803–1811.

[33] YU, J., YANG, L., XU, N., YANG, J., AND HUANG, T. Slimmable neural networks. *arXiv preprint arXiv:1812.08928* (2018).

[34] ZAGORUYKO, S., AND KOMODAKIS, N. Wide residual networks. *arXiv preprint arXiv:1605.07146* (2016).

[35] ZHANG, X., ZHOU, X., LIN, M., AND SUN, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 6848–6856.

[36] ZHAO, B., CUI, Q., SONG, R., QIU, Y., AND LIANG, J. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition* (2022), pp. 11953–11962.