NAS-BENCH-ASR: REPRODUCIBLE NEURAL ARCHITECTURE SEARCH FOR SPEECH RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Powered by innovations in novel architecture design, noise tolerance techniques and increasing model capacity, Automatic Speech Recognition (ASR) made giant strides in improving prediction accuracies over the past decade. ASR models are often trained with *tens of thousand* hours of high quality speech data to produce the state-of-the-art results. Industry-scale ASR model-training thus remains as a computationally heavy and time-consuming procedure, and consequently attracted little attention thus far in adopting automatic techniques in exploring neural architecture variations. Neural Architecture Search (NAS), on the other hand, gained a lot of interest in the past few years for its ability in discovering state-of-the-art architectures mainly in computer vision tasks. However, NAS approaches also suffer from the requirement of a large-scale computing infrastructure to support training of a massive number of neural networks and are often difficult to reproduce. Lately, a number of attempts have been made to ameliorate the computational problem and improve the search turnaround time of NAS algorithms by introducing benchmark datasets like NAS-Bench-101, NAS-Bench-201, and NAS-NLP. These datasets, however, focus predominantly on computer vision and NLP tasks and thus suffer from the problem of limited coverage of application domains. In this work we apply NAS for finding *cell* architecture for ASR models and release a comprehensive NAS-Bench dataset for reproducible NAS research. The dataset consists of $8,242$ unique ASR models trained on the TIMIT audio dataset, each starting from 3 seed initializations. Novelty of our dataset includes consideration for on-device deployability and inclusion of runtime measures of all the models on diverse hardware platforms and settings. We further evaluate performances of a number of NAS algorithms on our dataset. Finally, we show that cells in our search space for TIMIT transfer well to a much larger LibriSpeech dataset.

1 INTRODUCTION

Innovations in Deep Neural Network (DNN) architecture design, data augmentation techniques and a continuous increase in the amount of available high quality training datasets, resulted in a massive reduction in ASR error rates over the past decade [1; 31; 39; 19; 33]. However, training ASR models to achieve state-of-the-art performance accuracy remains challenging as it requires computationally heavy training process, e.g., often thousands GPU-hours are needed for good convergence [1; 17]. Furthermore, the requirement of hyper-parameter optimizations increases the computational loads in ASR training. Despite the system-level complexities in the training procedure, the importance of novel architecture design has proven extremely important in a variety of application domains including ASR [6; 34], computer vision [21; 14], and natural-language processing (NLP) [42; 7]. However, architecture design is a non-trivial task and often depends on years of experience, domain knowledge of the researchers and is driven by empirical successes.

Over the past few years, the deep learning community has been witnessing a trend in increased popularity of adopting automatic techniques for finding neural network architectures over traditional hand-designed alternatives. NAS algorithms have proven themselves highly successful in discovering state-of-the-art architectures, in addition to finding the optimal set of parameters for them, in various computer vision tasks [3; 16; 23; 36; 40; 41]. However, many of them suffer from high computational requirements, necessitating a large number of architecture variations to

be trained [46]. Furthermore, NAS algorithms are often difficult to reproduce as they use diverse training procedures and employ different architecture search spaces [26; 37]. Recently, attempts have been made to mitigate these problems by releasing various *benchmark* datasets for NAS to the community [44; 9; 38; 20]. These datasets usually provide a mapping between an architecture variant and their post training performances, which can be used efficiently by a NAS algorithm, i.e., the time-consuming training tasks can be replaced by fast dataset queries while searching for better architectures. Initial attempts of creating benchmark datasets predominantly focus on image classification tasks (with only one targeting NLP tasks), and thus suffer from poor application coverage.

In this work we address the lack of coverage problem by introducing a NAS-benchmak dataset in the domain of ASR, to our best knowledge the very first of its kind. To build the dataset, we have trained $8,242$ unique convolutional neural network architectures on the TIMIT dataset [12]. We consider convolutional architectures due to their recent success in beating state-of-the-art models in ASR [34; 13]. Moreover, convolution based architectures are computationally efficient to run on mobile CPUs, thus favouring real-time on-device deployment. Our dataset contains multiple runs of the entire training procedure, spanning 3 different initializations of the network parameters. In addition to the per epoch validation and final test metrics, such as *Phoneme Error Rate* (PER), and CTC loss, we also provide run-times of individual architectures in varying batch sizes on different hardware platforms including desktop and embedded GPUs. We consider a number of NAS algorithms [46; 36; 10; 25; 26] and show their performance on our search space, showcasing potential challenges and differences from their performance on the existing benchmarks. Lastly, we show the transferability of the top architecture cells found on TIMIT to a much larger Librispeech [30] dataset.

In summary, the contributions of this paper are:

- **Design of ASR NAS Search Space.** ASR NAS-Bench is a *first-of-its-kind* search space for convolutional speech models. It facilitates the reproducible study of ASR through NAS methods and thus fills a important gap in literature. The associated (soon to be publicly released) dataset consists of $8,242$ unique *cells* and contains validation and test metrics along with model parameters, FLOPs and on-device run-times.
- **Enabling NAS for Large-scale ASR.** Prohibitive training times for non-toy ASR datasets, has prevented NAS from strongly influencing the evolution of ASR architecture design. We show that ASR NAS-Bench is able to support the discovery of cell structures that generalize even to large-scale datasets like Librispeech – *a key breakthrough*. We believe the methodological decisions in this paper will act as a blueprint for future work where NAS plays a prominent role in ASR design.
- **Validating Existing NAS Algorithm Design.** Existing understanding of NAS is grossly influenced by image-based tasks. By systematically benchmarking popular NAS algorithms, under a rich ASR search space, our findings provide *otherwise lacking scientific support* for prior results.

2 Related work

NAS benchmarks. Ying *et al.* inroduced NAS-Bench-101 dataset in an attempt to address the difficulties in reproducing NAS research [44]. It contains data about over 400K unique image classification models trained on the CIFAR10 dataset. Despite being the biggest NAS dataset yet, not all NAS algorithms can utilize the dataset due to the restrictions it imposes on the maximum number of edges to limit the search space size. NAS-Bench-201 [9], which consists of 15K models, was designed similarly to its precursor and thus also focuses on image classification models, but tries to overcome the limitations of the 101 dataset and includes more diagnostic data. Concurrently to NAS-Bench-201, NAS-Bench-1shot1 was published with focus on benchmarking NAS which use weight sharing [45]. NAS-Bench-301 [38], points out the need for surrogate functions to scale NAS benchmarking. The authors prepared a benchmark based on the DARTS search space [27] with approximately 10^{18} models. Finally, NAS-Bench-NLP is a NAS benchmark containing models constructed by considering custom recurrent cells, which are used to replace traditional layers like LSTMs, totaling to 14K different architectures [20] trained on language modelling task. To the best of our knowledge, it is the only NAS benchmark that goes beyond the task of image classification.

NAS in the speech domain. Very recently, there has been a visible increase in interest in NAS for speech-related tasks, such as keyword spotting [29; 28], speaker verification [8; 35], or acoustic scene classification [24]. Consequently, NAS has also appeared in some works concerning speech recognition [15; 5; 18; 2]. Chen *et al.* used a search methodology based on the vanilla DARTS to op-

timize a CNN-based feature extractor which is then followed by a fixed Bi-LSTM module and multiple output heads [5]. They focused their evaluation on performance of the NAS-discovered models in mono- and multilingual settings using the Full Language Pack from IARPA BABEL [11] showing improvements over a VGG-based extractor. Unlike the above, Kim *et al.* considered evolution-based search to optimize a micro-cell used in a transformer-based architecture [18]. The evaluation was done using a monolingual setting only but with consideration of both English and Korean-based datasets of approximately 100 hours of speech each. Similar to Chen *et al.*, He *et al.* used differentiable search (although using P-DARTS [4] as their base) to optimize a convolution-based model but without the recurrent "tail" [15]. Also, unlike the previous ones, their evaluation only compromises single language (Chinese Mandarin) but using various datasets spanning between 170 and 10k hours of speech. The most similar to ours is the work by Baruwa *et al.* [2]. They also consider performance on both TIMIT and LibriSpeech datasets, but unlike us they studied them independently rather than using TIMIT as a proxy for LibriSpeech. Further differences involve search space design – the authors only focused on selecting operations which were then placed in a fixed, feed-forward manner (although the paper lacks some details so it is hard to say anything for sure), and also allowed to interleave convolutions with recurrent blocks. Lastly, we would like to emphasize that all the current work on NAS for ASR focuses on using the search solely to improve results – they provide important empirical proofs that NAS can be successfully used in relevant domains but at the same time they lack solid foundations needed for analysis and reasoning about the overall process, its limitations etc. This gap in the existing literature is exactly what we try to fill by providing information about large-scale study of the effects of architectural changes on ASR models.

3 ASR NAS-BENCH

The main purpose of the ASR NAS-Bench dataset is to provide a direct mapping from an architecture instance in the micro-architecture search-space (§ 3.1) to its training time and final performance metrics. The mapping is designed for any NAS algorithm to quickly navigate the architecture space without incurring the time-consuming and computationally-heavy training procedure. For micro-architecture search we use the TIMIT dataset [12] (§ 3.2) and present details of the pilot experiment conducted to select *hyperparameters* (§ 3.3), used to train $8,242$ models (§ 3.4).

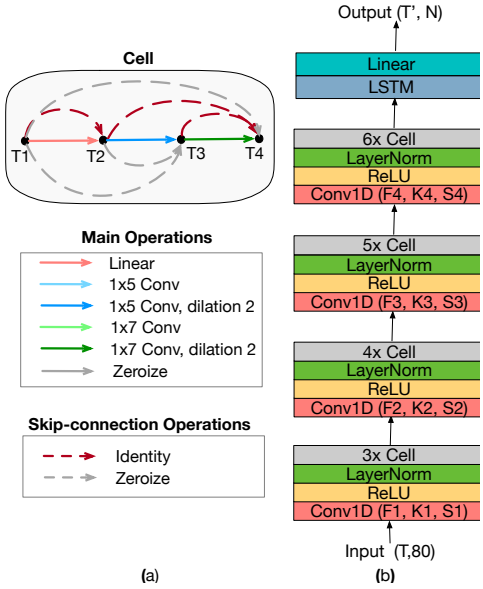

Figure 1: (a) Example of micro-architectures or cells trained on the TIMIT speech dataset. Overall, the cell architecture space has $8,242$ unique compositions. (b) An overview of the macro-architecture.

3.1 ASR ARCHITECTURE SEARCH SPACE

In line with existing NAS approaches [27; 32] and NAS Benchmarks [44; 9], we restrict our search to a small feed-forward neural network topology-space, commonly known as *cells*. We repeat and arrange a chosen cell to construct a predefined macro-architecture, which is then trained on the TIMIT dataset.

Micro-Architecture. A micro-architecture or a *cell*, as shown in Figure 1(a), is represented by a *directed acyclic graph* (DAG). We consider DAGs with four nodes representing tensors $t_1, \ldots, t_4$ and allow two types of edges: *main* and *skip connection* edges. A main edge connects two successive nodes t_{i-1} and t_i in the graph as shown by the solid line-arrows in Figure 1(a). A skip connection edge on the other hand, can connect any two nodes j and i, with the constraint $j < i$ and are depicted as dotted line-arrows in Figure 1(a). Each edge $e_{j \to i}$ represents an operation on the value of node t_j. The value of a node t_i is computed by summing the results of operations done by all incident edges on t_i (i.e., $e_{j \to i}$, where $j < i$).

We consider a choice of six operations for the main edges: linear operation, four convolution operations distinguished by choices of $(\text{kernel\_size}, \text{dilation}) \in \{(5,1),(5,2),(7,1),(7,2)\}$, and a zero operation, which outputs a tensor of zeros with the same shape and type as its input. Skip con-

nection operations on the other hand, can be either the identity operation or the zero operation. We used L2 kernel regularizer with convolution operations and dropouts with linear operations. Within the convolution operations, the size of the kernels (e.g., 5 or 7) is chosen as a trade-off between the audio context duration and the model size. Similarly two dilation factors are considered (e.g., 1 or 2) to investigate the trade-off between audio context duration and time resolution. As the final step of a cell, the value of t_4 is further passed through a layer normalization to produce the output of the cell. These design choices are made considering the search space that can be used by the vast majority of the NAS algorithms.

Macro-Architecture. The macro architecture considered in this paper is illustrated in Figure 1(b), which follows a sequential computation and is composed of four blocks followed by a unidirectional LSTM[1] and a linear layer. Individual blocks are composed of a convolution layer, an activation layer (ReLU), a layer normalization and a composition of C_i ($i = 1, 2, 3, 4$) search cells, with the same micro architecture across all the blocks. Each block is parametrized by three parameters F_i, K_i and S_i. These are used to define the number of filters F_i, the kernel size K_i and the stride S_i of the convolution layer. Note that F_i is also the number of filters of the convolution layers inside the search cell. We use the following set of parameters to define the macro-architecture while performing all micro-architecture training on the TIMIT dataset: $C_{1:4} = [3, 4, 5, 6]$, $F_{1:4} = [600, 800, 1000, 1200]$, $K_{1:4} = [8, 8, 8, 8]$ and $S_{1:4} = [1, 1, 2, 2]$.

3.2 TIMIT Dataset

TIMIT [12] is one of the earliest datasets designed for evaluating and benchmarking phoneme ASR systems. It is ideally suited for neural architecture search experiments due to its small size and high quality transcriptions.It comprises of $6,300$ utterances from 630 speakers, amounting to 5.4 hours of speech. We use the standard training partition of $3,696$ utterances from 462 speakers. Following [22], we split the core test dataset into a test partition, consisting of 24 speakers, and a validation partition. Following kaldi Timit-s5 recipe, the original 61 phonemes are mapped into a set of 48 phonemes, which forms the output layer targets for all the models. These 48 phonemes were further folded to a set of 39 during evaluations [22]. We use 80-dimensional *log-mel* spectrograms computed over 25 ms sliding window with a stride of 10 ms as input features. During training, we employ a curriculum learning strategy [1], where only audio utterances shorter than 1s are presented in the first 2 epochs, followed by audio utterances shorter than 2s for the next two epochs, and then the complete training set for the rest of the epochs. For efficiency, we also use a batch bucketing strategy, where a batch size of 64 is used for audio utterances smaller than 2s, and a batch size of 32 is used otherwise. We used CTC beam-search decoder with beam-size of 4. We did not use any language model in our experiments in order to avoid its confounding impact and its HPO (i.e., weighting factors). This also helped us to keep the architecture search for Acoustic Model (AM) tractable. All training experiments on TIMIT report the PER on validation and test partitions.

3.3 Pilot Experiment

As different training procedures can potentially lead to substantially different results [44], in this work we employ a single general training procedure for all models. Moreover, we use the same parameter set, e.g., F, K, S, learning rate, and decay across all $8,242$ models. In order to select good values of the parameters, we conducted a range of training experiments, where we performed grid-search to find good macro structure parameters and optimizer settings.

Selecting macro structure parameters. Since our macro structure has four blocks, we searched over tuples of size-four for each of the macro structure parameters (F, K, S). Specifically, we considered number of filters $F \in \{[600, 800, 1000, 1200], [900, 1100, 1300, 1500], [1200, 1300, 1500, 1700]\}$, kernel sizes $K \in \{[6, 8, 10, 12], [8, 8, 8, 8], [10, 10, 10, 10], [12, 12, 12, 12]\}$ and time reduction via strides $S \in \{[2, 2, 2, 1], [1, 1, 2, 2], [2, 2, 1, 1]\}$.

Selecting optimizer settings for training. Next, we also explored good ranges of the learning rate, decay factor and start epoch for an exponential decay learning rate scheduler used in conjunction

[1]We considered convolution and unidirectional LSTM based architectures since they are computationally efficient to run on mobile CPUs, thus favouring real-time on-device deployment.

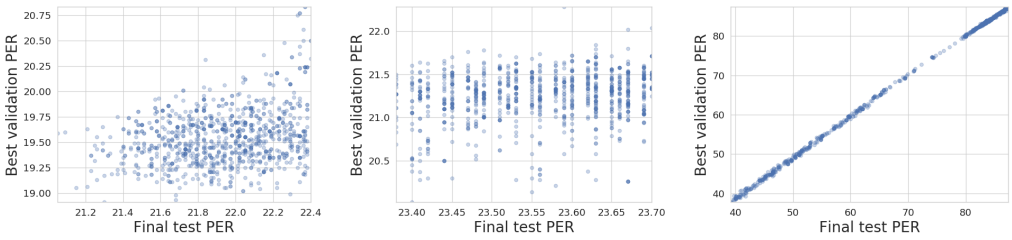

Figure 2: Correlation of best validation and final test PER for top, middle and bottom 1000 models.

with the Adam optimizer. Specifically, we considered the learning rate (LR) in $[10^{-3}, 10^{-4}, 5 * 10^{-5}, 10^{-5}]$, the decay factor in $[0.9, 0.95, 0.99]$, and the LR decay start epoch in $[5, 10, 15]$.

Finally, we randomly chose 5 micro architectures among all possible $8,242$ cells and for each cell we conducted the grid search over the aforementioned parameters. Specifically, for each cell we created a model from a particular macro architecture which we trained via a particular optimizer setting on TIMIT. We then identified the parametric setting that resulted in the lowest PER across most 5 cells in these experiments and used the same parameters in all latter model training experiments.

3.4 Training Framework for NAS-Bench-ASR

Individual models are trained using a TensorFlow-based training pipeline running on a single GPU. We leveraged NVIDIA V100 and P40 GPUs, and decreased training time by increasing throughput via the bucketing strategy based on the audio length. Metrics such as CTC loss and *Phoneme Error Rate* (PER) for TIMIT were used to measure model performance. Specifically, train and validation metrics were logged at each epoch, and the test metrics were logged at the end of training for the best model (refered to as *final test PER*). Our dataset contains logs of each of the $8,242$ models trained with three different seedsand for 40 epochs. Additionally, we computed the number of parameters and floating point operations (FLOPs) for each of the architectures and measured their latency on two commonly used hardware platforms.

4 Analysis of NAS-Bench-ASR Dataset

As described in §3.1, our cell search space is constructed by selecting 3 main operations from a set of 6 candidates, and then selecting either identity or zeroize operation for the 6 skip connection edges (c.f., Figure 1(a)). This amounts to $13,824$ ($6^3 \times 2^6$) possible instances for the search-space. However, the use of zero operations introduces computational *graph-isomorphism*. To identify unique architectures, following [44; 10], we first identify nodes with zero operations and disconnect them from their neighbors. Then we perform a reachability test and remove all nodes that can not be reached from the input or output nodes of the cell. After this we end up with a minimized graph representation, which we hash using an iterative graph invariant approach [43]. After accounting for isomorphism and discarding the empty graph from the list of valid models, we found $8,242$ unique architectures, out of which $8,000$ are without any zero operations.

Distribution of Model Performances. The first observation from analyzing the model training results is that the majority of the models converged and achieved a final test PER within a small range. Specifically, 82.86% (6,829) of the models fall below a PER of 26%, with the best model reaching PER of 21.05%. On the other hand, the remaining 17.14% of the models form a rather evenly-distributed tail, achieving PER values between 26% to 100%. Because of the high density of points below 26% PER and overall high range of PER values, we decided to conduct our analysis by considering three subgroups of models: Top models are the top 1000 architectures according to test PER, Middle are models at positions between 2000 and 3000 in the test PER ranking, and Bottom are between 7000 and 8000.

Correlation between Validation and Test PERs. In Figure 2 we show the correlation between the final test PER and the best validation PER. We use best validation PER because the final test PER was computed on the checkpoint which had the best validation score over all epochs. The results show there is moderate correlation for top models, weak for middle ones, and strong for worst

Group	Positions	Test PER (%)			Val. PER (%)			Test-Val Correl.
		Min.	Max.	Avg.	Min.	Max.	Avg.	
Overall	All	21.05	94.04	31.54±39.09	18.91	93.94	29.35±40.76	0.8522
Top	1-1000	21.05	22.40	21.93±0.56	18.91	20.83	19.55±0.54	0.2104
Middle	2000-3000	23.38	23.70	23.56±0.18	20.02	22.28	21.25±0.57	0.1561
Bottom	7000-8000	39.21	87.35	72.17±31.98	37.76	87.41	71.93±32.82	0.9986

Table 1: Summary of test *vs.* validation analysis for the overall dataset and three subgroups. Positions taken according to the test PER ranking. Correlation reported using Spearman-ρ coefficients.

Group	NB1	NB2			NB-ASR
		CIFAR-10	CIFAR-100	ImageNet-16	
Overall	0.989	0.993	0.987	0.997	0.852
Top 1000	0.356	0.855	0.613	0.827	0.210
Top 5%	0.573	0.826	0.596	0.796	0.145

Table 2: Comparison of correlations between test and validation accuracies reported in the existing image classification benchmarks (NB1 and NB2) and ours (NB-ASR). In all cases, global correlation is much higher than local correlation for the top performing models, suggesting that better models always tend to have higher discrepancy between their test and validation scores. We include both Top 1000 and Top 5% metrics to account for the fact that NB1 is an order of magnitude larger than the two other benchmarks.

ones. High-level results are summarized numerically in Table 1. In general, correlation between validation and test accuracy among the top performing models is significantly lower than for image classification benchmarks – which is additionally highlighted in Table 2. We suspect that this fact might pose additional challenge to NAS algorithms.

Distribution of Operations. Figure 3 (first row) presents the distribution of selected operations among main edges across top, middle and bottom most models in terms of performances. We see top models have the highest count for *linear* operation, while the middle and bottom groups have the low selection of linear operation but higher for *convolution* operation. At the same time, as shown in Figure 3 (second row), the worst models (bottom group) are also the ones which have the highest selection of *skip-connections*. Figure 4 further confirms this observations showing how significantly worse the models in the search space become as they include more than 2 skip connections. The trend introduced by linear layers is also visible, but not as much.

Interestingly, we found that the micro cells in our search space yielding the best validation and test PERs are similar to the structure found in the state-of-the-art convolution models such as time-depth-separable (TDS) models [34]. Figure 5 shows the TDS-cell and the micro-cells with best validation and test PERs (from left to right). This indicates the potential of conducting NAS on smaller datasets to discover new architectures yielding state-of-the-art results on larger datasets.

Latency, Params, FLOPs and PER. Figure 6 investigates the relationship among test PER, number of parameters and latency of models in the dataset. FLOPS counts are closely correlated to the number of parameters so are not shown for simplicity. Although this figure shows only latency measured on a desktop GPU (NVIDIA GeForce GTX 1080 Ti) with batch size of 32, the dataset includes latency measured on different devices with varying batch sizes. Also shown in the figure are 13 Pareto-optimal models of test PER over latency which are indicated by purple (x) markers. The best models in terms of test and validation PER are highlighted in red markers. TDS-like models achieve test PER between 22.7% and 23.45 % and the best validation PER model has test PER of 22.13%. In addition, there are Pareto-optimal models that achieve much lower latency without significant increase in PER, for instance, the model indicated by violet (◆) has one-seventh of the latency of the best test PER model and still has a test PER of 21.92%. The distribution of coloured clusters along the y axis suggests that the number of parameters is not a strong factor to determine the accuracy of models. Analogically, models with similar number of parameters can lead to very different latency.

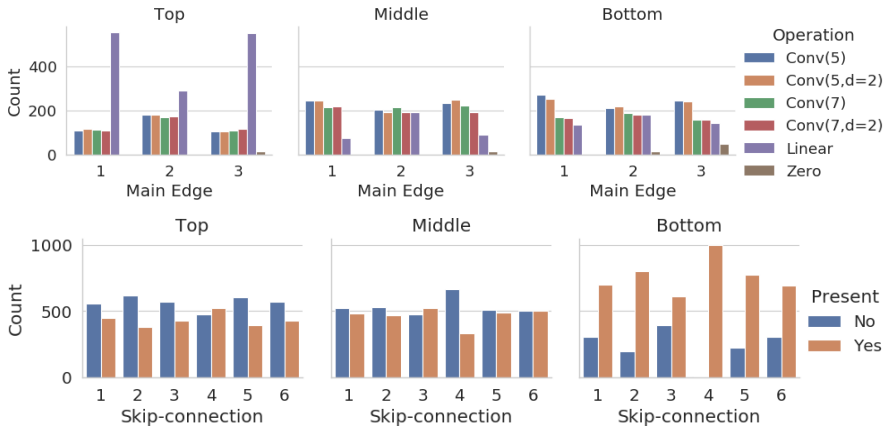

Figure 3: Distribution of selected edges for each node (first row) and distribution of skip-connections for possible edges (second row). Top, middle and bottom are grouped by final test PER.

5 TRANSFERABILITY

Training an ASR architecture to achieve the state-of-the-art *Word Error Rate* (WER) is a slow process, as it often requires iterating over tens of thousand hours of speech data, and potentially over various hyperparameter choices. Even when distributed across multiple GPUs, this process can take days or weeks [1; 17]. Since the largest strides in lowering WERs have resulted from new architectures being developed manually by speech experts slowly over decades, it would be beneficial to scale such exploration through NAS. However, performing NAS directly on large scale ASR datasets needed for peak performance is not practical due to significantly more computational costs. An issue which is further exacerbated when one is also interested in architectures yielding lower latency and computational cost.

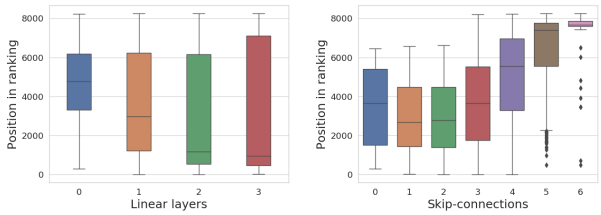

Figure 4: Positions of models in the ranking according to their test PER, when they are grouped according to the number of Linear layers (left) and skip-connections (right) used in their designs.

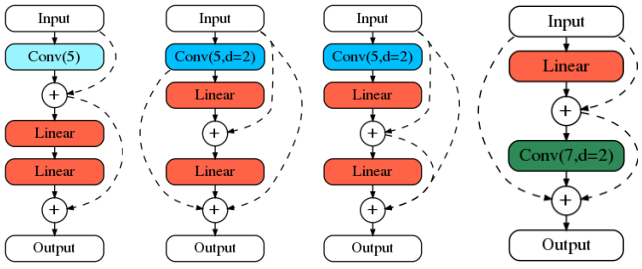

Figure 5: Cell architectures in our search space (from left to right): TDS-like, with best validation and test PERs, and one of the efficient pareto points. Note that the TDS-like cell could also have different convolution selected as the first operation.

For this reason we explore how well new architectures found in NAS search over TIMIT (5 hours) transfer to Librispeech (100 hours). Specifically, we investigate whether there is a high correlation between the best/random/worst performing new architectures found on TIMIT in terms of PER, to the *Label Error Rate* (LER) and WER values when training the same architectures on Librispeech. If a high correlation is present this would mean that one could lower the computational requirements for exploring new architectures in larger datasets by instead running NAS on that same architecture landscape but with smaller datasets instead. In our case this would mean a 20x speedup in NAS, and in general this would open a new avenue for research, namely on how much one could save by running NAS on different smaller datasets, or running NAS on subsets of larger datasets extracted by current or next generation data summarization techniques.

Librispeech Dataset. Librispeech [30] is a widely used ASR training and benchmarking database of about 1000 hours of read speech derived from audiobooks. The training dataset is partitioned into

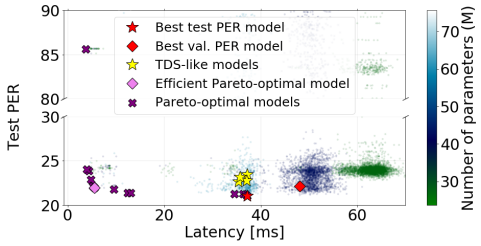

Figure 6: Test PER versus latency, color-coded by the number of parameters. Best models, Pareto-optimal models, and selected TDS-like models are highlighted.

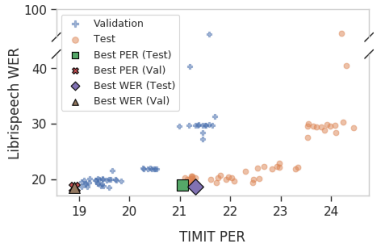

Figure 7: Transfer between TIMIT and Librispeech on validation and test datasets with high validation and test correlation of 0.87 and 0.84, respectively.

clean-100, clean-360 and other-500 sets with the clean partitions exhibiting much higher SNR. The test and dev sets are also partitioned similarly. In our work, we have used clean-100 partition for training and dev-clean and test-clean for validation and testing. The choice of 100 hours partition is to strike a balance between building a decent large scale model while still keeping the computational cost involved in architecture search experiments plausible. Previous literature indicates that the models trained on 100 hours partition are about 1-1.5% absolute [30] behind the models trained on 1000 hours when evaluated on test-clean set.

Training. Librispeech models were trained with 4 GPUs with Horovod distributed framework. We use 80-dimensional log-mel features computed over 25ms Hamming windows strided by 10ms. Guided by latency considerations, we used a vocabulary of 780 sub-word tokens as output. Model performances are measured by tracking CTC loss, LER, and WER on the validation dataset during training and on test dataset post-training.

Pilot Experiments. Similarly to §3.3, we also conducted a pilot experiment to determine appropriate macro structure parameters and hyper-parameters to train the models for Librispeech. To promote transferability between datasets at a reasonable compute cost, with the micro cell of the best model found on TIMIT dataset (out of $8,242$), we conducted another grid search but instead on Librispeech dataset. Accordingly, we chose the final number of filters, kernel size, dilation, learning rate, decay factor and start epoch based on lowest LER/WER on Librispeech.

Correlation Analysis. For a meaningful analysis we selected 100 architectures from our search space based on their TIMIT PER, namely we took the best 20, 70 random, and the worst 10. We then plugged the micro cell architectures in these models into the macro architecture described in the previous subsection for Librispeech. Finally, we trained these models on Librispeech with the hyper-parameters reported earlier and computed their LERs/WERs. We observe that models that didn't converge for TIMIT also didn't converge for Librispeech. For this reason, when looking at the strength of transferability between TIMIT and Librispeech we considered models that had TIMIT PER lower than 50%. As shown in Figure 7, there is a high correlation between TIMIT PER and Librispeech WER. In particular, we observe 0.87 correlation between validation metrics, and 0.84 for test metrics. This indicates indeed that it is possible to do indirect NAS for cell micro architecture on Librispeech by instead applying NAS directly to TIMIT and then reusing the found cell structures in macro architectures suitable for larger datasets.

6 NAS Experiments

Following the existing benchmarks, we include results from running different NAS algorithms on our search space, using the proposed dataset to query for accuracy of models if applicable. We consider the following standard NAS algorithms: Regularized Evolution (RE) [36], Hyperband (HYPB) [25], REINFORCE with a LSTM-based controller (REINF) [46], Binary Predictor (BP) [10], and Random Search (RAND) [26]. We also include an algorithm based on Deep Q-Learning (QLRN) which behaves similarly to RE but tries to chain mutations (actions) and learns what mutations can eventually lead to better results. We report results as functions of "trained models" to unify notion of cost.Because in the full training setting we trained models for 40 epochs, if

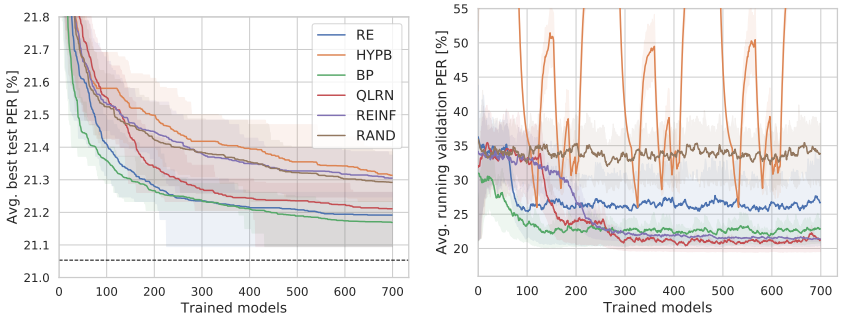

Figure 8: Performance of NAS algorithms on our ASR search space. Shaded regions mark interquartile ranges of relevant averages. The dashed line represents the best model in the search space.

an algorithm trains a model for fewer epochs than that (e.g. HYPB) we consider the cost of such a training to be $n/40$ of a "trained model", where n is the number of epochs used ($0 < n \leq 40$). We run each algorithm 100 times and present results in Figure 8 as average test PER of the best model found (all algorithms were using validation accuracy as their optimization target), and average validation PER of the most recently trained model (after smoothing the curve using exponential moving average with weight 0.9). The first observation by looking at the results is that neither of the algorithms was able to consistently find the best model – this is somewhat surprising to us considering the moderate size our search space and the fact that from our experience algorithms like RE are able to consistently find the best model on similar benchmarks, e.g., NAS-Bench-201 (given similar budget). Another interesting observation is related to the weak correlation between validation and test accuracies mentioned in the previous sections. By comparing the figures on the right and left sides, we can see that algorithms which tend to better optimize their rewards (validation accuracy) as search progresses are not necessarily better in the test PER objective – this is especially visible in the case of REINFORCE, which significantly outperforms random search if validation accuracy is considered, but when evaluated using test PER it presents comparable results. Another surprise is weak performance of Hyperband – the only algorithm that has the ability to adjust number of epochs to train models. From our observations it seems that many of our models converge to relatively good PER values fast (after 5-10 epochs) so we expected Hyperband to be able to leverage this property. However, it seems that in the case of our search space this is not so helpful – potentially because models might still change relative ordering as they are trained for more epochs despite the fact that they have already achieved decent performance.

7 Conclusions

We introduced NAS-Bench-ASR, a first comprehensive dataset for allowing computationally inexpensive training/test time evaluation of model performances, while conducting NAS research in the domain of ASR. We trained $8,242$ unique models on the TIMIT dataset across 3 different seeds and in addition to the common training metrics in ASR, e.g., PER, CTC-loss, we also provide information on the number of parameters, FLOPS and latency of running all the models on three different hardware platforms. Model run-times are especially beneficial for NAS researchers interested in optimizing architectures for efficient on-device deployment. We presented a comprehensive analysis of our dataset and evaluated the performances of a number of NAS algorithms on it. Lastly, we show that good cell structures as identified on the TIMIT dataset transfer well to Librispeech, which paves the way for affordable NAS on ASR domain.

References

[1] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, and et al. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 173–182. JMLR.org, 2016.

[2] A. Baruwa, M. Abisiga, I. Gbadegesin, and A. Fakunle. Leveraging end-to-end speech recognition with neural architecture search, 2019.

[3] H. Cai, C. Gan, and S. Han. Once for all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*, 2020.

[4] X. Chen, L. Xie, J. Wu, and Q. Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1294–1303, 2019.

[5] Y.-C. Chen, J.-Y. Hsu, C.-K. Lee, and H. yi Lee. DARTS-ASR: Differentiable Architecture Search for Multilingual Speech Recognition and Adaptation. In *Proc. Interspeech 2020*, pages 1803–1807, 2020.

[6] C.-C. Chiu and C. Raffel. Monotonic chunkwise attention. In *ICLR*, 2018.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 2019.

[8] S. Ding, T. Chen, X. Gong, W. Zha, and Z. Wang. AutoSpeech: Neural Architecture Search for Speaker Recognition. In *Proc. Interspeech 2020*, pages 916–920, 2020.

[9] X. Dong and Y. Yang. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *International Conference on Learning Representations (ICLR)*, 2020.

[10] Ł. Dudziak, T. Chau, M. S. Abdelfattah, R. Lee, H. Kim, and N. D. Lane. BRP-NAS: Prediction-based NAS using GCNs. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[11] M. Gales, K. Knill, A. Ragni, and S. P. Rath. Speech recognition and keyword spotting for low-resource languages: Babel project research at cued. In *SLTU*, 2014.

[12] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, N. L. Dahlgren, and V. Zue. TIMIT Acoustic Phonetic Continuous Speech Corpus. *Linguistic Data Consortium*, 1993.

[13] A. Hannun, A. Lee, Q. Xu, and R. Collobert. Sequence-to-Sequence Speech Recognition with Time-Depth Separable Convolutions. In *Proc. Interspeech 2019*, pages 3785–3789, 2019.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] L. He, D. Su, and D. Yu. Learned transferable architectures can surpass hand-designed architectures for large scale speech recognition, 2020.

[16] A. Howard, R. Pang, H. Adam, Q. Le, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, and Y. Zhu. Searching for mobilenetv3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 10 2019.

[17] J. Kahn, M. Rivière, W. Zheng, E. Kharitonov, Q. Xu, P. E. Mazaré, J. Karadayi, V. Liptchinsky, R. Collobert, C. Fuegen, T. Likhomanenko, G. Synnaeve, A. Joulin, A. Mohamed, and E. Dupoux. Libri-Light: A Benchmark for ASR with Limited or No Supervision. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7669–7673, 2020.

[18] J. Kim, J. Wang, S. Kim, and Y. Lee. Evolved Speech-Transformer: Applying Neural Architecture Search to End-to-End Automatic Speech Recognition. In *Proc. Interspeech 2020*, pages 1788–1792, 2020.

[19] K. Kim, K. Lee, D. Gowda, J. Park, S. Kim, E. S. Kim, Y.-Y. Lee, J. Yeo, D. Kim, S. Jung, J. Lee, M. Han, and C. Kim. Attention based on-device streaming speech recognition with large speech corpus. In *ASRU*, 2019.

[20] N. Klyuchnikov, I. Trofimov, E. Artemova, M. Salnikov, M. Fedorov, and E. Burnaev. NAS-Bench-NLP: Neural Architecture Search Benchmark for Natural Language Processing, 2020.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] K.-F. Lee and H.-W. Hon. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, 1989.

[23] R. Lee, Ł. Dudziak, M. Abdelfattah, S. Venieris, H. Kim, H. Wen, and N. Lane. Journey towards tiny perceptual super-resolution. In *European Conference on Computer Vision (ECCV)*, 08 2020.

[24] J. Li, C. Liang, B. Zhang, Z. Wang, F. Xiang, and X. Chu. Neural Architecture Search on Acoustic Scene Classification. In *Proc. Interspeech 2020*, pages 1171–1175, 2020.

[25] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1):6765–6816, Jan. 2017.

[26] L. Li and A. Talwalkar. Random search and reproducibility for neural architecture search. *CoRR*, abs/1902.07638, 2019.

[27] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

[28] H. Mazzawi, X. Gonzalvo, A. Kracun, P. Sridhar, N. Subrahmanya, I. L. Moreno, H. J. Park, and P. Violette. Improving Keyword Spotting and Language Identification via Neural Architecture Search at Scale. In *Proc. Interspeech 2019*, pages 1278–1282, 2019.

[29] T. Mo, Y. Yu, M. Salameh, D. Niu, and S. Jui. Neural Architecture Search for Keyword Spotting. In *Proc. Interspeech 2020*, pages 1982–1986, 2020.

[30] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[31] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. SpecAugment: A simple data augmentation method for Automatic Speech Recognition. *Interspeech 2019*, Sep 2019.

[32] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient Neural Architecture Search via Parameters Sharing. volume 80 of *Proceedings of Machine Learning Research*, pages 4095–4104. PMLR, Jul 2018.

[33] V. Pratap and R. Collobert. Online speech recognition with wav2letter@anywhere, 2020.

[34] V. Pratap, Q. Xu, J. Kahn, G. Avidov, T. Likhomanenko, A. Hannun, V. Liptchinsky, G. Synnaeve, and R. Collobert. Scaling up online speech recognition using convnets. *arXiv preprint arXiv:2001.09727*, 2020.

[35] X. Qu, J. Wang, and J. Xiao. Evolutionary Algorithm Enhanced Neural Architecture Search for Text-Independent Speaker Verification. In *Proc. Interspeech 2020*, pages 961–965, 2020.

[36] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33, 02 2018.

[37] C. Sciuto, K. Yu, M. Jaggi, C. Musat, and M. Salzmann. Evaluating the search phase of neural architecture search. *CoRR*, abs/1902.08142, 2019.

[38] J. Siems, L. Zimmer, A. Zela, J. Lukasik, M. Keuper, and F. Hutter. NAS-Bench-301 and the Case for Surrogate Benchmarks for Neural Architecture Search, 2020.

[39] G. Synnaeve, Q. Xu, J. Kahn, E. Grave, T. Likhomanenko, V. Pratap, A. Sriram, V. Liptchinsky, and R. Collobert. End-to-End ASR: From supervised to semi-supervised learning with modern architectures. In *ICML: Workshop on Self-supervision in Audio and Speech*, Jul 2020.

[40] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2815–2823, 06 2019.

[41] M. Tan and Q. V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[43] C. Ying. Enumerating unique computational graphs via an iterative graph invariant. *ArXiv*, abs/1902.06192, 2019.

[44] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter. NAS-Bench-101: Towards Reproducible Neural Architecture Search. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7105–7114, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[45] A. Zela, J. Siems, and F. Hutter. NAS-Bench-1Shot1: Benchmarking and Dissecting One-shot Neural Architecture Search. In *International Conference on Learning Representations (ICLR)*, 2020.

[46] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2017.