Hierarchical Constrained Bayesian Optimization for Feature, Acoustic Model and Decoder Parameter Optimization

Akshay Chandrashekaran, Ian Lane

Department of Electrical and Computer Engineering,
Carnegie Mellon University, USA

akshayc@cmu.edu, lane@cmu.edu

Abstract

We describe the implementation of a hierarchical constrained Bayesian Optimization algorithm and it's application to joint optimization of features, acoustic model structure and decoding parameters for deep neural network (DNN)-based large vocabulary continuous speech recognition (LVCSR) systems. Within our hierarchical optimization method we perform constrained Bayesian optimization jointly of feature hyper-parameters and acoustic model structure in the first-level, and then perform an iteration of constrained Bayesian optimization for the decoder hyper-parameters in the second. We show the the proposed hierarchical optimization method can generate a model with higher performance than a manually optimized system on a server platform. Furthermore, we demonstrate that the proposed framework can be used to automatically build real-time speech recognition systems for graphics processing unit (GPU)-enabled embedded platforms that retain similar accuracy to a server platform, while running with constrained computing resources.

Index Terms: Hyper-Parameter Optimization, Bayesian Optimization, Deep Neural Networks, Speech Recognition, Optimization under constraints

1. Introduction

Deep Neural Networks (DNN) acoustic models, in their variety of forms, currently obtain state-of-the-art performance for ASR [1]. Although they are trained in a statistical approach to learn the parameters of the models, they are governed by a set of tunable parameters, called hyper-parameters, that determine the model structure and generalization performance.

State of the art on-line speech recognition systems for large vocabulary continuous speech recognition (LVCSR) are typically built with large acoustic models and language models. Performing speech recognition on embedded platforms like smartphones requires consideration of the limited computational power for the platform. Hence, we need a method to train a speech recognizer that gives the best performance in terms of word error rate (WER) faster than real time, allowing for other tasks to run in parallel when performing real-time speech recognition.

Typically, experts use manual tuning, or an exhaustive grid search of hyper-parameters for the given task. However, this is usually not efficient [2], or in the case of grid search, can quickly become infeasible as the number of hyper-parameters, or their granularity increases. Automated hyper-parameter optimization has shown to perform well in terms of exploration and exploitation of the hyper-parameter search space. However, these too can be affected by the curse of dimensionality: As the number of dimensions increases, it takes longer to converge to a good solution. We observe that the tuning of hyper-parameters can be split based on where they come into play. For speech recognition, the feature and model hyper-parameters are relevant to the training stage, and decoder hyper-parameters can be iterated over far quicker for a given DNN acoustic model during evaluation. Doing so would prevent a good training model configuration from being ignored simply because the decoder hyper-parameters were poorly chosen.

In this paper, we do the following:

- We propose a technique of hierarchical constrained optimization with a separation between training and evaluation hyper-parameters.
- We compare it's performance to manual optimization, decoder-only optimization and simultaneous optimization of all hyper-parameters.
- We perform the comparison across three platforms : a server, and two embedded platforms.
- We demonstrate that our method yields configurations for embedded platforms that retains performance similar to the server platform, while satisfying the given constraint.

2. Related Work

In [3], the authors show that Bayesian optimization and covariance mean adaptation evolution strategy (CMA-ES) outperform manual optimization, but use only word error rate (WER) as the objective and tune a GMM model on the Resource management task. [4] looks at multi-objective CMA-ES to optimize towards model size and WER for a DNN-HMM system. However, they perform only tuning of DNN model hyper-parameters and do not look at the decoder hyper-parameters. [5] use constrained Bayesian optimization to tune DNN model hyper-parameters using improvement in terms of WER of the validation set with RTF as the constraint. [6] perform a scalarization of WER and real time factor (RTF) and perform a variation of co-ordinate descent to tune decoder hyper-parameters, but only deal with continuous hyper-parameters. We are not aware of any research that focuses on the joint automated optimization of feature, model and decoder hyper-parameters for LVCSR.

3. Hyper-Parameters

Hyper-parameters are parameters that govern the performance and structure of machine learning algorithms or models, and influence the number of free parameters in them, and how the underlying optimization algorithm optimizes them. Usually, these hyper-parameters are set based on some previous experimental results and are fine-tuned manually. Based on empirical experiments, it can be seen that these hyper-parameters are also dependent on the data being trained on [7]. Below are the hyper-

parameters in the feature-space, DNN model-space, and the decoder.

3.1. Feature Hyper-parameters

For the audio features, we use log-Mel filter-bank features. The hyper-parameters that we optimize for this are the size of each frame, the amount of shift between consecutive frames, and the number of filters used in the Mel filter-banks. We experiment with window size varying from 5 ms to 50 ms in increments of 5, for the window shift: 5ms to 25ms in steps of 5ms, and 5 to 50 Mel filters in steps of 5.

3.2. Model Hyper-parameters

In the model-space, the hyper-parameters explored in this paper govern the structure and size of the DNN. The number of consecutive feature frames to be spliced at the input layer were from 0 to 9 before and after the current frame. The number of hidden layers are from 1 to 6. The number of neurons per hidden layer are from 512 to 2304 in steps of 256.

3.3. Decoder Hyper-parameters

For the decoder hyper-parameters, we vary the acoustic scale from 0.05 to 0.15, the decoder beam from 10 to 18, the lattice beam from 4 to 10, the minimum number of active states from 50 to 600 in steps of 50, the maximum number of active states from 2000 to 7000 in steps of 500, and the lattice pruning interval from 5ms to 50ms in steps of 5ms. Descriptions of these hyper-parameters can be found in [8] and [9].

In addition to the above hyper-parameters, there are other hyper-parameters like the learning rate, amount of regularization for the DNN training, and language model pruning type and amount, which we have not considered in this paper. These can also be incorporated into our techniques to explore further possible gains.

4. Proposed Hyper-Parameter Optimization Technique

We describe the proposed approach for hierarchical constrained Bayesian optimization in section 4.3. In sections 4.1 and 4.2, we go over some of the prerequisite theory for the proposed approach.

4.1. Bayesian Optimization

Sequential model-based global optimization (SMBO) is a technique for hyper-parameter optimization that uses the results of objective function evaluations done previously to guide the search. It fits a model over the observed values, and uses that to determine where to evaluate the objective next. The resultant information is added to the history to update the model. The standard SMBO is given in Algorithm 1. $\mathcal{H} = \{\mathbf{x}_i, f(\mathbf{x}_i)\}_{i=1}^{t}$ is the set of hyper-parameter configurations and corresponding results seen till current time t. M_0 is the initial model to be fit over existing observations. S is the surrogate function or acquisition function that is used to determine where to evaluate next.

Bayesian Optimization is an SMBO technique that models the objective function as a Gaussian process (GP) over the hyper-parameter space[10]. We assume that the objective function to be minimized is modelled by a GP prior:

Algorithm 1 Sequential Model-based Global Optimization (SMBO)

procedure SMBO(f, M_0, T, S)
  $\mathcal{H} \leftarrow \phi$,
  for $t \leftarrow 1$ to T **do**
    Fit a model M_t to $\mathcal{H}$,
    $\mathbf{x}^* \leftarrow \text{argmin}_{\mathbf{x}}\ S(\mathbf{x}, M_t)$,
    Evaluate $f(\mathbf{x}^*)$, *(Expensive)*
    $\mathcal{H} \leftarrow \mathcal{H} \cup (\mathbf{x}^*, f(\mathbf{x}^*))$,
  end for
return $\mathcal{H}$
end procedure

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{1}$$

where $m(x)$ is the prior mean function, and $k(\mathbf{x}, \mathbf{x}')$ is the covariance function that determines the smoothness properties of the samples drawn from the GP. Given t arbitrary points, and the corresponding function value $\{\mathbf{x}_{1:t}, \mathbf{f}_{1:t}\}$, the posterior distribution of the function at any arbitrary point $\mathbf{x}$ is a multivariate Gaussian. Assuming the prior mean function is 0 for simplicity,

$$P(f(\mathbf{x})|\mathbf{x}) = \mathcal{N}\left(\mu_t(\mathbf{x}), \sigma_t^2(\mathbf{x})\right) \tag{2}$$

where

$$\begin{aligned} \mu_t(\mathbf{x}) &= \mathbf{k}^T\mathbf{K}^{-1}\mathbf{f}_{1:t} \\ \sigma_t^2(\mathbf{x}) &= k(\mathbf{x},\mathbf{x}) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k} \\ \mathbf{k} = \begin{bmatrix} k(\mathbf{x},\mathbf{x}_1) \\ k(\mathbf{x},\mathbf{x}_2) \\ \vdots \\ k(\mathbf{x},\mathbf{x}_t) \end{bmatrix} & \quad \mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & \dots & k(\mathbf{x}_1,\mathbf{x}_t) \\ k(\mathbf{x}_2,\mathbf{x}_1) & \dots & k(\mathbf{x}_2,\mathbf{x}_t) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_t,\mathbf{x}_1) & \dots & k(\mathbf{x}_t,\mathbf{x}_t) \end{bmatrix} \end{aligned} \tag{3}$$

This mean and variance function is used to create a surrogate acquisition function which is easier to evaluate compared to the actual evaluation function. This acquisition function guides the hyper-parameter search and balances between exploration and exploitation. We used expected improvement [11] as the surrogate function since it has been shown to generalize well to multiple examples [12].

$$EI(\mathbf{x}) = \begin{cases} [(\mu_t(\mathbf{x}) - f(\mathbf{x}^+))\Phi(Z) \\ \quad + \sigma_t(\mathbf{x})\phi(Z)] & \text{if } \sigma_t(x) > 0 \\ 0 & \text{else} \end{cases} \tag{4}$$

$$Z = \frac{\mu_t(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma_t(\mathbf{x})}$$

$\mathbf{x}^+$ is the hyper-parameter with the best observed objective function value so far. $\phi(.)$ is the normal probability distribution and $\Phi(.)$ is the normal cumulative distribution. More details regarding this can be found in [10].

4.2. Constrained Bayesian Optimization (CBO)

Constrained Bayesian optimization as proposed by [13] incorporates constraints with the original optimization metric. It modifies the surrogate function with a probability term associated with the fulfilment of the constraint. Let $\mathcal{C}_m(\mathbf{x})$ represent the m^{th} constraint condition, which indicates if the constraint is satisfied at $\mathbf{x}$. Then, the probabilistic constraint is $P(\mathcal{C}_m(\mathbf{x})) \geq 1-\delta_m$ where $1-\delta_m$ is a user specified minimum

confidence for the m^{th} constraint. All M constraints have to be satisfied. Assuming each constraint is independent of the other, they are modelled by independent Gaussian Processes. This can be done by specifying the constraints by a function $g_m(\mathbf{x})$ such that $g_m(\mathbf{x}) \geq 0$ only if $\mathcal{C}_m(\mathbf{x})$ is satisfied.The acquisition function in (4) is modified to

$$a(\mathbf{x}) = EI(\mathbf{x}) \prod_{m=1}^{M} P(g_m(\mathbf{x}) \geq 0) \tag{5}$$

When no feasible regions are present, the algorithm focuses on an exploitative search over the constraint function till a feasible region is found. This modifies the acquisition function to:

$$a(\mathbf{x}) = \prod_{m=1}^{M} P(g_m(\mathbf{x}) \geq 0) \tag{6}$$

This enables the algorithm to perform an exploitative search over the space to find feasible regions, and then perform normal exploration to find the best possible setting to satisfy the constraints.

4.3. Hierarchical Constrained Bayesian Optimization (HCBO)

In this approach, we split the hyper-parameters optimization procedure into two levels. The outer level performs the joint tuning of the feature and model hyper-parameters using constrained Bayesian Optimization. For a given set of feature and model hyper-parameters, one iteration of the training pipeline is performed. With the resultant model, an inner loop of constrained Bayesian optimization is performed using the decoder hyper-parameters for a given number of iterations. Once the given number of iterations are done, the minimum validation WER whose RTF falls below the threshold is selected as the result for the upper level. If no such decoder setting is found, the minimum WER and it's corresponding RTF are passed to the upper level.

5. Experimental Setup

For these experiments, we use the Wall Street Journal corpus [14] for the training data. We use the WER of the dev93 data-set for hyper-parameter optimization. We also give the eval92 data-set results for the best systems. For the embedded platform, we use two platforms: the NVidia TK1 and the NVidia TX1 developer boards. The server platform was an Intel Core-i7-5930K processor with the NVidia TitanX with Maxwell architecture as the graphics processing unit (GPU).

5.1. Training Pipeline

For each iteration of optimization, we execute the standard Kaldi training recipe upto the speaker dependent training stage (tri4b) for Wall Street Journal[9]. We use MFCC features with the window size and window shift as specified by the hyper-parameters. Then, alignments from this model along with log-Mel filter-bank features are used to create the training data for the DNN.

The training data is then split into a training and cross-validation set in the ratio of 90:10, which is given to the DNN training framework used in [15]. The DNN training procedure performs 10 epochs of stochastic gradient descent using momentum. The training hyper-parameters like learning rate are not changed for the experiments. The best result is selected as the model that gives the minimum frame level error on the cross-validation data. All hidden layers were rectified linear units (ReLU) [16] with a soft-max output layer. Typically, the training takes an average of 10 hours for 10 epochs.

5.2. Decoding Pipeline

A hybrid GPU based Weighted finite state transducer (WFST) decoder [17] is used for decoding. The GPU is used to generate the acoustic model likelihoods, while the decoding graph is traversed on the CPU. The language model consists of a trigram language model with special noise and silence words generated from the training text. This is made into a fully composed decoding graph, with the DNN assuming the role of the acoustic model. We use the hybrid decoder to generate state level lattices for the utterances in the data-set. We perform a multi-threaded decode for all the utterances, but measure the RTF on a single thread with a single GPU. Typically, it takes about 5 minutes on average per decode, but this time can vary depending on the quality of the model and the lattice sizes generated.

5.3. Optimization Pipeline

For our task, we set the RTF threshold as 0.2 for the server platform, and 0.5 for the TK1 and TX1 platforms. The rationale behind this is that we typically expect lesser amount of multi-tasking on the embedded platforms, and hence, can devote a larger part of the compute capability for speech recognition.

For the manual experiments, we performed some manual iterations of model hyper-parameter optimization, and performed a few iterations of manual decoder hyper-parameter optimization on the beam, lattice beam and number of active states till the RTF threshold constraint was met. For the decoder-CBO experiments *(CBO-D)*, we used the manually selected model hyper-parameter configuration and ran 20 iterations of constrained Bayesian Optimization. This is reflective of the methodology tried in [18].

For the constrained optimization experiments with all hyper-parameters *(CBO-A)*, we report the best system obtained after 25 iterations of training and decoding.

For the hierarchical constrained optimization *(HCBO)* experiments, we perform 25 iterations of joint feature and model-space hyper-parameter optimization. Within each model build, we perform 20 iterations of decoder hyper-parameter constrained optimization, and report the configuration with the best WER with RTF below the specified threshold. Thus, though we perform $20\times$ the number of decoder function evaluations in the proposed approach, we still train the same number of DNN models, using about 10% more wall time compared to optimization using all hyper-parameters simultaneously.

6. Results

We compare the performance of manual optimization against direct constrained Bayesian optimization of all hyper-parameters and the proposed hierarchical constrained Bayesian optimization. We perform the comparison on both the TK1 and the TX1 platforms.

The results of the various optimization techniques on the server platform are given in table 1 in lines 1-4. Given the RTF constraint of 0.2, the system configuration from manual optimization yields a validation WER of 8.2%. On optimizing the decoder hyper-parameters for the manual system, we get 0.5% improvement in WER, but with $1.1\times$ improvement in speed. Constrained Bayesian optimization using all hyper-parameters

H/W	Method	Hyper-parameters													WER(%)		RTF
		Training						Decoding									
		Feature Space			Model Space												
		fS	fSh	nB	C	L	Size	ac	b	lb	minA	maxA	pr		dev	eval	
Server	Manual	25	10	40	5	4	2048	0.11	14.0	8.0	50	5000	25		8.29	4.87	0.19
Server	CBO-D	25	10	40	5	4	2048	0.10	14.0	6.5	300	4500	30		8.25	4.91	0.17
Server	CBO-A	30	15	25	4	3	1536	0.10	14.0	6.5	300	4500	30		9.59	5.85	0.16
Server	HCBO	50	10	50	3	6	512	0.15	12.9	4.0	50	5500	50		**8.05**	**4.70**	**0.06**
TK1	Manual	25	10	40	5	4	512	0.11	13.0	8.0	50	2500	25		8.33	5.1	0.48
TK1	CBO-D	25	10	40	5	4	512	0.10	11.7	9.0	50	7000	50		8.29	5.19	0.48
TK1	CBO-A	15	10	50	7	5	512	0.07	10.8	4.0	50	6000	50		8.62	5.0	0.48
TK1	HCBO	30	10	50	7	6	512	0.10	12.9	8.5	600	6000	45		**8.26**	**4.91**	**0.44**
TX1	Manual	25	10	40	5	4	512	0.11	13.0	8.0	50	2500	25		8.33	5.1	0.45
TX1	CBO-D	25	10	40	5	4	512	0.09	11.7	4.0	500	3000	50		8.29	5.14	0.43
TX1	CBO-A	45	10	45	2	4	512	0.15	12.2	9.0	50	7000	50		8.66	5.21	0.35
TX1	HCBO	50	10	50	2	5	512	0.15	13.9	4.0	450	7000	50		**8.02**	**4.55**	**0.46**

Table 1: *Results of the different optimization techniques and the corresponding hyper-parameter configuration on the hardware platforms of interest. fS: Frame Size (ms), fSh: frame shift (ms), nB: Number of filters, C: Number of frames spliced to the current frame in the past and the future, L: Number of hidden layers, Size: Number of neurons per hidden layer, ac: acoustic scale, b: decode beam, lb: Lattice beam, minA: Minimum number of active states, maxA: Maximum number of active states, pr: Lattice Pruning interval (ms).*

yields a configuration that is 16% worse in terms of validation WER. However, this system is $1.18\times$ faster. On performing HCBO, after 25 iterations, the best system performs 2.9% relative better than the manual optimization system in terms of validation WER. However in terms of RTF, the HCBO system performs $3.16\times$ faster. Among the 25 iterations for CBO-A, only 8 models trained fell below the RTF threshold. For HCBO, 24 models gave validation WERs with RTF below the threshold.

The performance on embedded platforms is given in table 1 in lines 5-12. We see similar trends for WER in the TK1 and TX1 platforms, with joint constrained optimization of all hyper-parameters yields a poorly performing system. Automated optimization of only the decoder optimization yields a system that is better than the manually optimized system. The proposed hierarchical method yields even better performance compared to all the tried techniques. In the TK1 platform, HCBO yields a system satisfying the RTF constraint with 0.3% relative improvement in validation WER compared to the server manual baseline. In the TX1 platform, HCBO yields a system with 3.2% relative improvement in validation WER compared to the manual server baseline. HCBO yielded 23 and 24 system configurations with RTF below the threshold for TK1 and TX1 respectively.

Observing the hyper-parameters of the optimal systems generated for the three platforms, we see that in the feature space, all the systems gravitate towards a larger frame size, while maintaining the same frame shift of 10ms. The reason for this would be that for greater window shifts, for this datasets, progressively fewer utterances got successful alignments, reducing the available training data. Also, the optimized systems prefer larger number of mel-filters in the filter-bank. We assume that this gives the DNN a more representative feature set.

In terms of model space, all the optimal systems gravitate towards thin and deep neural networks. The amount of context splicing was in general spread over multiple values.

In the decoder, we observe that the default acoustic scale does not transition well to different model configurations, with thin deep systems leaning towards giving the acoustic model more weight. The lattice beam also seems to be mixed between the three architectures. In terms of minimum number of active states, maximum number of active states and the lattice pruning interval, systems from HCBO gravitate towards a larger size, indicating that keeping more paths active for longer yields a better performing system, while still allowing for better computational performance.

7. Conclusion

We have presented a novel hierarchical hyper-parameter optimization method for LVCSR systems. We see that for this task, with the given constraints, hierarchical optimization outperforms standard constrained Bayesian optimization using all hyper-parameters since it gets the best available decoder hyper-parameter setting for a given feature and model hyper-parameter combination. This is because the inner level of optimization reuses the same DNN model, but finds the best possible decoder settings for the same within the specified number of iterations, mitigating the curse of dimensionality. We see that this method generalizes across three hardware platforms, giving a model that is close to or outperforming the baseline in terms of WER while still satisfying the RTF constraints specified for the given hardware. The only manual intervention needed is to specify the ranges of the hyper-parameters and to specify at which level they are to be optimized.Although we have only shown this working with Bayesian optimization, this method can easily be extended to other existing optimization techniques.

Studying the optimal systems has yielded some insight into the nature of hyper-parameters that need to be selected. Generalization of these selections and trends remain to be seen in other datasets and platforms.

We have explored only one set of constraints for the task. The performance of this method to more stringent constraints, and even multiple constraints, though supported, remains to be tested.

Currently, each inner loop of the hierarchical optimization does not use the information of the inner loops of previous optimizations. A future direction of research is to transfer the information from previously completed inner loops to a new inner loop. This may yield potentially faster convergence of the inner loop, reducing training time. Incorporating other hyper-parameters like learning rate, regularization, etc. can yield potential improvement in WER using this technique.

8. References

[1] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, January 2012.

[2] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[3] S. Watanabe and J. Le Roux, "Black box optimization for automatic speech recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3256–3260.

[4] T. Moriya, T. Tanaka, T. Shinozaki, S. Watanabe, and K. Duh, "Automation of system building for state-of-the-art large vocabulary speech recognition using evolution strategy," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 610–616.

[5] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8609–8613.

[6] A. El Hannani and T. Hain, "Automatic optimization of speech decoder parameters," *IEEE Signal Processing Letters*, vol. 17, no. 1, pp. 95–98, 2010.

[7] A.-r. Mohamed, G. E. Dahl, and G. E. Hinton, "Acoustic modeling using deep belief networks," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 14–22, Jan 2012.

[8] D. Povey, M. Hannemann, G. Boulianne, L. Burget, A. Ghoshal, M. Janda, M. Karafiát, S. Kombrink, P. Motlíček, Y. Qian *et al.*, "Generating exact lattices in the wfst framework," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4213–4216.

[9] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011, iEEE Catalog No.: CFP11SRW-USB.

[10] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[11] J. Močkus, "On bayesian methods for seeking the extremum," in *Optimization Techniques IFIP Technical Conference*. Springer, 1975, pp. 400–404.

[12] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[13] M. A. Gelbart, J. Snoek, and R. P. Adams, "Bayesian optimization with unknown constraints," *arXiv preprint arXiv:1403.5607*, 2014.

[14] D. B. Paul and J. M. Baker, "The design for the wall street journal-based csr corpus," in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 357–362.

[15] W. Chan and I. Lane, "Deep recurrent neural networks for acoustic modelling," *arXiv preprint arXiv:1504.01482*, 2015.

[16] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.

[17] J. Kim and I. R. Lane, "Accelerating large vocabulary continuous speech recognition on heterogeneous cpu-gpu platforms," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014, pp. 3291–3295.

[18] A. Chandrashekaran and I. Lane, "Automated optimization of decoder hyper-parameters for online lvcsr," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 454–460.