Interspeech 2018
2-6 September 2018, Hyderabad

Efficient language model adaptation with Noise Contrastive Estimation and Kullback-Leibler regularization

Jesús Andrés-Ferrer[1], *Nathan Bodenstab*[1], *Paul Vozila*[1]

[1]Nuance Communications

name@nuance.com
name ∈ {jesusandres.ferrer, nathan.bodenstab, paul.vozila}

Abstract

Many language modeling (LM) tasks have limited in-domain data for training. Exploiting out-of-domain data while retaining the relevant in-domain statistics is a desired property in these scenarios. Kullback-Leibler Divergence (KLD) regularization is a popular method for acoustic model (AM) adaptation. KLD regularization assumes that the last layer is a softmax that fully activates the targets of both in-domain and out-of-domain models. Unfortunately, this softmax activation is computationally prohibitive for language modeling where the number of output classes is large, typically 50k to 100K, but may even exceed 800k in some cases. The computational bottleneck of the softmax during LM training can be reduced by an order of magnitude using techniques such as noise contrastive estimation (NCE), which replaces the cross-entropy loss function with a binary classification problem between the target output and random noise samples. In this work we combine NCE and KLD regularization and offer a fast domain adaptation method for LM training, while also retaining important attributes of the original NCE, such as self-normalization. We show on a medical domain-adaptation task that our method improves perplexity by 10.1 % relative to a strong LSTM baseline.

Index Terms: speech recognition, NCE, KLD, language modeling, adaptation

1. Introduction

Neural Networks (NN) have become standard in language modeling. Such NN models range from feed-forward models [1] to vanilla recurrent models [2] to long short-term memory models [3], or even convolutional models [4]. Initially, the majority of computation to train these models was spent during the softmax computation at the output layer. This computation required a time proportional to the vocabulary size times the last hidden layer size.

Several methods have been proposed in the literature to reduce the computational requirements of the output layer, such as output short-lists [1], importance weight sampling [5, 6], and noise contrastive estimation (NCE) [7, 8, 9]. NCE stands out from other methods because of two important properties: it yields a self-normalized model; and only a small and constant[1] number of output nodes need to be computed during training. Compared to cross-entropy training with a softmax output layer, NCE reduces the training time by up to an order of magnitude, depending on the output layer size [9]. During decoding, an NCE-trained model is self-normalized and only requires querying a few words per LM lookup, in contrast to the entire vocabulary. At least one work [10] has claimed that NCE is worse than other methods such as importance softmax sampling, but recent research proves this is not the case if the noise distribution is correctly selected [11] and hyper-parameters are properly tuned [12].

In tasks such as domain adaptation or teacher distillation [13], we wish to train an in-domain model that remains 'close'[2] to another general, out-of-domain model. A successful technique to achieve this goal is to add a Kullback-Leibler divergence (KLD) regularization term to the standard maximum likelihood estimation (or cross-entropy loss). This has been applied to several tasks such as image recognition [13], user adaptation for automatic speech recognition [14], and machine translation for reducing the left-to-right error bias [15].

In the case of LM, it is not straight-forward to apply KLD regularization together with the NCE loss function. Of course we could apply KLD regularization to the traditional cross-entropy loss function, but as discussed above, this comes with the unfortunate extra cost of computing the softmax, which will increase both training and decoding times. To the best of our knowledge, there is no loss function for neural network domain adaptation with KLD that also retains the efficiency properties of NCE. In this work, we extend NCE loss to include a KLD regularization term that keeps the trained model parameters close to a second (reference) model. We do this by reusing the noise samples generated during NCE training and adapting them via importance sampling [16], as well as explicitly computing part of the KLD expectation. This modifies the NCE data distribution to include the KLD regularization term with a computational cost similar to that of NCE training and also yields a self-normalized model.

In sections 2 and 3 we review KLD regularization and NCE to build the foundation of our approach. The NCE loss function together with KLD regularization is introduced in section 4, where we also show how to approximate such loss efficiently. Finally, in section 5, we validate our technique experimentally on an internal task. We conclude in the following section by proposing some future research directions.

2. Kullback-Leibler regularisation

Cross-entropy is the standard training loss used for multiclass output neural network models. For a given context h, the cross-entropy is defined as follows,

$$L_{\text{XE}}(h) = -\text{E}_{p_d}[\log p_\theta(w|h)] \quad (1)$$

where $p_\theta(w|h)$ is a model with parameters θ, and $p_d(w|h)$ is the data distribution we want to learn as well as the probabilty over which the expecation is taken. In practice, since we have

[1]Actually, the sample size is contant and the number of active output nodes is less than or equal the sample size + 1 depending on whether there are sampling collisions

[2]In the sense of having similar output distributions.

10.21437/Interspeech.2018-1345

a sample $T = \{(y_n, h_n)\}_{n=1}^{N}$, the data distribution is approximated by

$$p_d(w|h) \approx \begin{cases} 1 & \text{if } (w,h) \in T \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Note that we use w for refering to either a generic next word or the random variable, and y_n (or y) for denoting the specific next word that follows a specific training data context h_n (or h).

Finally, for a given training example from the sample T, say (y_n, h_n), the cross-entropy loss is simplified to

$$L_{\text{XE}}(h_n) = -\sum_w p_d(w|h_n) \log p_\theta(w|h_n) = -\log p_\theta(y_n|h_n) \quad (3)$$

Since the sample data distribution is 1-hot vectors (or hard targets), the expectation over the vocabulary is approximated with a single word and it would not be necessary to activate the entire output layer if the softmax did not involve the computation of the partition function.

For a given LM context h, the KL divergence is defined as

$$D_{\text{KL}}(h) := -\text{E}_{p_t}[\log \frac{p_\theta(w|h)}{p_t(w|h)}] \quad (4)$$

where $p_t(w|h)$ is a generic distribution we want our trained model $p_\theta(w|h)$ to be close to. In general, we are interested in learning the parameters, θ, and p_t is considered to be known. In this case, the regularization term simplifies to

$$R_{\text{KL}}(h) = -\text{E}_{p_t}[\log p_\theta(w|h)] \quad (5)$$

where the entropy of p_t is dropped since it does not depend on θ. In contrast to cross entropy in Eq. (3), this expectation requires computing the full expectation over all words in the vocabulary, which implies fully activating the output layer even if our NN is self-normalized and does not require computing the partition function.

The KLD regularized cross entropy takes this final form

$$L_t(h) = \gamma L_{\text{XE}}(h) + (1-\gamma) R_{\text{KL}}(h) \quad (6)$$

Since both $L_{\text{XE}}(h)$ and $R_{\text{KL}}(h)$ are expectations over the same distribution, it is equivalent to the following formulation [14]

$$L_t(h) = -\text{E}_{p_\gamma}[\log p_\theta(w|h)] \quad (7)$$

with p_γ defined as follows

$$p_\gamma(w|h) = \gamma p_d(w|h) + (1-\gamma) p_t(w|h) \quad (8)$$

KLD regularization is helpful when the trained model needs to be close to a reference model, while also extracting relevant information from the data distribution. This is helpful in scenarios such as low-data user adaptation [14], distilling an ensemble of models to a single model [13, 17, 18], or distilling a larger model to a smaller model [13, 18]. In the latter case, a temperature parameter is typically used at the output softmax logits to smooth the distribution [13].

3. Noise Contrastive Estimation (NCE)

Noise contrastive estimation (NCE) [7, 9, 10, 11, 12] restates maximum likelihood estimation for parameter inference as a binary classification problem. This binary classification problem separates the data distribution, p_d, from the so-called *noise distribution*. We approximate p_d by the model, $p_\theta(w|h)$, and denote the noise distribution as $p_n(w|h)$. Specifically, NCE minimizes the following loss

$$\mathcal{L}_{\text{N}}(h) = -[E_{p_d}[\log p(C{=}1|w,h)] + \nu E_{p_n}[\log p(C{=}0|w,h)]] \quad (9)$$

where $p(C = 1|w,h)$ is the probability of w to come from the model distribution versus $p(C = 0|w,h)$, which is the probability of w to come from the noise distribution. The term $\nu = \frac{p(C=0)}{p(C=1)}$ is the prior ratio of noise words to data words. Specifically, $p(C = 1|w,h)$ is defined as follows

$$p(C=1|w,h) = \frac{p_\theta(w|h)}{p_\theta(w|h) + \nu p_n(w|h)} \quad (10)$$

and $p(C = 0|w,h)$ is the contrary event $1 - p(C = 1|w,h)$ or

$$p(C=0|w,h) = \frac{\nu p_n(w|h)}{p_\theta(w|h) + \nu p_n(w|h)} \quad (11)$$

In practice, p_d is approximated as in Eq. (2) and the positive expectation (left expectation) in Eq. (9) is reduced to a single word, the *target word*, y_n, for its context pair, h_n. The *noise expectation* (or right expectation) in Eq. (9), is approximated with a sample of S negative words, $\mathcal{S}_n = \mathcal{S}(h_n) = \{w_1, \ldots, w_S\}$, sampled according to the noise distribution p_n. Taking these considerations into account, Eq. (9) is approximated as follows

$$L_{\text{N}}(h_n) = -[\log p(C=1|y_n,h_n) + \sum_{w \in \mathcal{S}_n} \log p(C=0|w,h_n)] \quad (12)$$

The model p_θ converges to the data distribution p_d when the NCE loss is minimized [7]. The convergence speed depends on many factors, one of them being the noise distribution used [7]. In practice, a good balance between convergence speed and cost per epoch is obtained using a power of the unigram [8, 11],

$$p_n(w|h) := p_n(w) \propto [C(w)]^\alpha \quad (13)$$

where $C(w)$ is the unigram count of w and α is a value in $[0, 1]$.

In addition to converging to the data distribution, NCE has another desirable property for LM tasks: self-normalization. It has been found that models trained with NCE loss are self-normalized [7, 8, 9, 11], implying that the model does not need to dynamically compute the normalization constant for each context, in contrast to the softmax activation. Simply using

$$p_\theta(w|h) = \exp([o(h)]_w - Z) \quad (14)$$

with a constant value Z, generates a normalized model. The normalization constant, Z, can be zero, but using a value close to the logarithm of the vocabulary size speeds-up convergence

4. NCE with KLD regularization

In this section, we explain how KLD regularization is added to the NCE loss function. The key idea is to notice that Eq. (6) is equivalent to Eq. (7) when using the distribution p_γ from Eq. (8). We can then substitute the p_d distribution of Eq. (9) with the p_γ distribution from Eq. (8). The combined NCE with KLD regularization loss function then becomes

$$\mathcal{L}_{\text{D}}(h) = -[E_{p_\gamma}[\log p(C{=}1|w,h)] + \nu E_{p_n}[\log p(C{=}0|w,h)]] \quad (15)$$

It is straightforward to prove that this loss is minimized when the model, p_θ, is equal to p_γ since it is an instantiation of

NCE [7], and therefore it will also produce a self-normalized model.

A naïve implementation of Eq. (15) requires the computation of $\log p(C{=}1|w,h)$ for all words in the vocabulary, which would suffer from the same speed issues as the cross-entropy with softmax activation layer.

Similar to standard NCE, we approximate the noise expectation with a sample of S negative words, $\mathcal{S}_n$, sampled accordingly to the noise distribution p_n as follows

$$\nu E_{p_n}[\log p(C{=}0|w,h_n)]] \approx \sum_{w\in\mathcal{S}_n} \log p(C=0|w,h_n)] \quad (16)$$

In order to maximize efficiency, we would like to reuse this noise sample to approximate the *positive expectation* of Eq. (15). We could directly apply importance sampling [16] to approximate the positive expectation (left expectation), however, this approach would rely on the noise distribution to randomly pick the target word y_n. This is clearly undesirable as the loss would not approach NCE loss as γ tends to 1. For instance, if we consider the extreme case when $\gamma{=}1$, then approximating the positive expectation with the noise sample directly by importance sampling will not recover the NCE loss.

Instead, we use importance sampling to only approximate what we termed *the partial expectation*, which is obtained by removing the target word contribution (y_n) from the full expectation. We first decompose the expectation given the training pair (y_n, h_n) as follows

$$E_{p_\gamma}[f_1(w,h_n)] = p_\gamma(y_n|h_n)f_1(w,h_n) + E^{\setminus y_n}_{p_\gamma}[f_1(w,,h_n)]$$

with $f_1(w,h_n) = \log p(C=1|w,h_n)$ for compactness, and where the partial expectation $E^{\setminus y_n}_{p_\gamma}[f_1(w,h_n)]$ is defined as follows

$$E^{\setminus y_n}_{p_\gamma}[f_1(w,h_n)] = \sum_{w\in\mathcal{W}-\{y_n\}} p_\gamma(w|h_n)f_1(w,h_n) \quad (17)$$

The partial expectation is approximated by importance sampling (IS) reusing the NCE noise sample $\mathcal{S}_n$ as follows

$$E^{\setminus y_n}_{p_\gamma}[f_1(w,h_n)] \approx \frac{1}{S^{\setminus y_n}\cdot Z(h_n,\mathcal{S}_n)} \sum_{w\in\mathcal{S}_n-\{y\}} \alpha(w|h_n)f_1(w,h_n) \quad (18)$$

where $S^{\setminus y_n} = S - N(y_n \in \mathcal{S}_n)$ is the number of valid elements in the noise sample, and $\alpha(w|h_n)$ is the IS weight defined as

$$\alpha(w|h_n) = \frac{p_\gamma(w|h_n)}{p_n(w|h_n)} \quad (19)$$

Finally, $Z(h_n,\mathcal{S})$ is the normalization constant for non-normalized reference models, p_t. It can be approximated considering the partial expectation as follows

$$Z(h_n,\mathcal{S}) = p_\gamma(y_n|h_n) + \frac{1}{S^{\setminus y_n}}\cdot \sum_{w\in\mathcal{S}-\{y_n\}} \alpha(w|h_n) \quad (20)$$

Note that if p_t is a normalized model, then this term equals 1.

Finally, Eq. (15) is approximated as follows

$$\begin{aligned} L_{\mathrm{D}}(h_n) = {} & p_\gamma(y_n|h_n)\log p(C=1|w,h_n) \\ & + \frac{1}{S^{\setminus y_n}}\frac{1}{Z(h_n,\mathcal{S}_n)} \sum_{w\in\mathcal{S}_n-\{y_n\}} \alpha(w|h_n)\log p(C=1|w,h_n) \\ & + \sum_{w\in\mathcal{S}_n} \log p(C=0|w,h_n)] \end{aligned} \quad (21)$$

Table 1: *Statistics of Gastroenterology (in-domain) train, development, and test sets*

Set	Running words	OOV (%)
train	265 989 391	0.0
dev	37 509	0.37
test	33 809	0.78

Table 2: *Statistics of training data.*

Set	Running words (millions)
ID	266.0
OOD	1 211.4
OOD+ID	1 477.4

Similar to distillation [13], we could also apply a temperature to the self-normalized model

$$p_\theta(w|h) = \exp(o_t(w;\theta)/T) \quad (22)$$

However, unlike softmax and KLD for distillation, the temperature parameter here does not cause the student model to approach the L2-loss of the logits for large temperatures [13]. Moreover, the resulting model will only be self-normalized in some cases; specifically, if both the p_t and p_θ models follow Eq. (22), and T is 1 (no temperature), or γ is 0 (no hard-targets). Finally, since there is no softmax activation, the temperature parameter does not smooth the distribution.

5. Experiments

In this section, we evaluate the proposed adaptation technique on an internal medical task by adapting from a general medical LM to a domain-specific LM. We choose Gastroenterology as the target domain, and partition our Gastroenterology data into three sets: train, development, and test. Per usual, we optimize hyperparameters on the development set and report test set perplexities. Table 1 contains a summary of the data sets statistics.

The general medical model is trained from 1.2 billion words sampled from all medical domains excluding the in-domain (Gastroenterology) specialty. We will refer to this corpus as the *out-of-domain* (OOD) corpus. The *in-domain corpus* (ID) is composed of 266.0 million words sampled from the in-domain medical specialty. Finally, we generate an additional corpus by adding the ID data to the end of the OOD data set, and refer to this corpus as the OOD+ID corpus. We add the ID data to the end so that during training, the model sees this in-domain data last as a mild type of adaptation. Table 2 contains a summary of the training data.

We train an LSTM [19] model with a vocabulary size of 58,116 words including an unknown word for out of the vocabulary tokens. The embedding size is 512 and contains two LSTM layers of $1,024$ nodes each. In total, the models each have 103.95 millions of parameters. During training we apply 0.25 feed-forward drop-out [20] and a small weight-decay of 10^{-6}. We train all models with either NCE or NCE with KLD regularization. In both cases, we found the noise distribution of Eq.(13) with $\alpha = 0.75$ to be beneficial. The norm of the gradients is clipped to 15.0 and we train using truncated back-propagation through time (BPTT) [21] with a length of 20. We batch 128 sequences in parallel and share the noise samples for

Table 3: *Perplexity results on the test set. All models are trained with NCE loss. The KLD model is trained with KLD regularization ($\gamma = 0.75$). We compare normalized PPL so that results are comparable, as well as relative perplexity improvements (RPPL) with respect to the model trained on the OOD+ID data.*

Model	Init	normalized PPL	RPPL(%)
OOD+ID	Random	10.9	n.a.
OOD	Random	17.8	-62.3 %
ID	Random	12.4	-13.7 %
Fine-Tune	OOD+ID	10.3	5.5 %
KLD	OOD+ID	9.8	10.1 %

the full batch. We optimize the number of noise samples to 256, and keep this value constant for all experiments, with and without KLD regularization. We train the models for 15 iterations with a fixed learning rate and then anneal the learning rate by half for 5 additional iterations as soon as the normalized peplexity improvement on the held-out is reduced below 0.3% with a maximum budget of 25 iterations.

We establish three baseline models: one trained only on the *out-of-domain* (OOD) data; another trained only on the *in-domain* data (ID); and a third trained on the OOD+ID data. For all baseline models, we optimize the learning rate with a grid search over the values $\{0.1, 0.2, 0.4, 0.8\}$, selecting the best value according to development performance. In addition to the above baselines, we train a Fine-Tune model that starts from the OOD+ID model parameters and it is additionally trained on just the ID data with a small learning rate until convergence. We optimized the fine-tuning learning rate with a grid search over values ($\{0.001, 0.005, 0.01, 0.05, 0.1, 0.2\}$) and found the best value on the development set to be 0.05. All of the above models are trained with NCE loss as described in Eq. (12).

We train the last model with KLD regularization, and use the same approach as for the fine-tuned model i.e. we bootstrap from the model trained on the OOD+ID data and then run another training round up to 25 epochs. The same range of learning rates is searched. For the γ parameter that balances between KLD regularization and the data distribution, we optimize this value on the development set as well, experimenting with values $\{0.0, 0.5, 0.75, 0.9, 0.95\}$.

Every model we trained – including the KLD regularized models –, has a similar logarithmic deviation on average from a fully self-normalized model of approximately 0.10 (a fully normalized model should have 0.0). We observe better self-normalization for models that do not use drop-out (approximately 0.01), however, drop-out models performed best. The variance is very small for all models and close to 0.01. Consequently, models can be re-centered to have a smaller normalization deviation, e.g. 0.02. This re-centering idea has been previously discussed in the literature [12].

Results in Table 3 show that the OOD+ID is the best single training recipe because it is performing a mild adaptation to the ID data. We compute relative PPL improvements over this baseline. Fine-tuning on in-domain data improves the perplexity at the cost of an additional training round on the in-domain data. Finally, adding KLD regularization to the fine-tuning round further improves the model by 10.1% relative.

In Table 4 we observe how adding the regularization term has a penalty of 20K words per second with respect to the NCE training. This penalty is because in constrast to NCE without KLD regularization where we only need to activate a single model, adding KLD regularization requires the reference (or out-of-domain) model to be activated as well. Note, however, that this cost is constant and independent of the output layer size accordintly to the algorithm described in Section 4, and an order of magnitude faster than softmax [3].

Table 4: *Processed words per second during training. NCE needs to forward propagate a single model, while NCE with KLD regularization needs to forward propagate 2 models.*

Loss	Words/sec (K)
NCE	75–71
NCE +KLD	55–53

Table 5: *Perplexity results for NCE with KLD regularization. The model is initialized with a model trained on OOD+ID data and then fine-tuned with KLD regularization on the ID data. We also report the relative perplexity improvement (RPPL) with respect to the model trained on the OOD+ID data from Table 3.*

γ	normalized PPL	RPPL
0.00	9.9	9.1 %
0.50	9.7	11.0 %
0.75	9.8	10.1 %
0.90	10.2	6.4 %
0.95	10.3	5.5 %
1.00	10.3	5.5 %

Table 5 reports PPL as a function of the interpolation weight γ on the test set. Observe how the KLD term smoothly regularizes the model which converges to an NCE fine-tuned model when γ equals 1.0. When $\gamma{=}0$, we see an unexpected gain as well. We believe this is the result of the model being exposed to only the ID data instead of the full OOD+ID data, as well as the Monte Carlo approximation to the positive expectation of Eq. (15). Note that the gradient of Eq. (21) will still generate a learning signal even in this case, in contrast to a softmax model with KLD regularization.

6. Conclusions

We have proposed an efficient technique to integrate KLD regularization directly with the NCE loss. The approach retains the advantageous NCE property of yielding a self-normalized model without the need to activate all outputs from either the reference model or the model being trained. Our proposal reuses the NCE noise sample for the partial expectation of the interpolated distribution. We have applied the proposed technique to a medical domain-adaptation task, and improved perplexity by 10.1% relative with respect to a strong baseline.

We believe this technique will be beneficial in scenarios were regularization is important and/or data is scarce. As future work, we plan to adapt our method to ensembles and knowledge distillation as well as to other tasks such as machine translation.

[3] We were not even able to train a softmax model on OOD(+ID), because of hardware constraints

7. References

[1] H. Schwenk, "Continuous space language models," *Computer Speech and Language*, vol. 21, no. 3, pp. 492–518, Jul. 2007.

[2] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, 2010, pp. 1045–1048.

[3] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, 2012, pp. 194–197.

[4] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv:1803.01271*, 2018.

[5] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, "On using very large target vocabulary for neural machine translation," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, 2015, pp. 1–10. [Online]. Available: http://www.aclweb.org/anthology/P15-1001

[6] Y. Bengio and J.-S. Sénécal, "Quick training of probabilistic neural nets by importance sampling," in *Proceedings of the conference on Artificial Intelligence and Statistics (AISTATS)*, 2003.

[7] M. U. Gutmann and A. Hyvärinen, "Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 307–361, 2012.

[8] W. Chen, D. Grangier, and M. Auli, "Strategies for training large vocabulary neural language models," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016. [Online]. Available: http://aclweb.org/anthology/P/P16/P16-1186.pdf

[9] X. Chen, X. Liu, M. J. F. Gales, and P. C. Woodland, "Recurrent neural network language model training with noise contrastive estimation for speech recognition." in *ICASSP*. IEEE, 2015, pp. 5411–5415.

[10] R. Józefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *CoRR*, vol. abs/1602.02410, 2016.

[11] S. Ji, S. V. N. Vishwanathan, N. Satish, M. J. Anderson, and P. Dubey, "Blackout: Speeding up recurrent neural network language models with very large vocabularies," *CoRR*, vol. abs/1511.06909, 2015.

[12] F. F. Liza and M. Grzes, "Improving language modelling with noise-contrastive estimation," *CoRR*, vol. abs/1709.07758, 2017.

[13] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015. [Online]. Available: http://arxiv.org/abs/1503.02531

[14] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *ICASSP 2013*, January 2013. [Online]. Available: https://www.microsoft.com/en-us/research/publication/kl-divergence-regularized-deep-neural-network-adaptation-for-improved-large-vocabulary-speech-recognition/

[15] H. Hassan Awadalla, A. Aue, C. Chen, V. Chowdhary, J. Clark, C. Federmann, X. Huang, M. Junczys-Dowmunt, W. Lewis, M. Li, S. Liu, T.-Y. Liu, R. Luo, A. Menezes, T. Qin, F. Seide, X. Tan, F. Tian, L. Wu, S. Wu, Y. Xia, D. Zhang, Z. Zhang, and M. Zhou, "Achieving human parity on automatic chinese to english news translation," March 2018. [Online]. Available: https://arxiv.org/abs/1803.05567

[16] H. Kahn and A. W. Marshall, "Methods of reducing sample size in monte carlo computations," *Operations Research*, vol. 1, no. 5, pp. 263–278, 1953.

[17] M. Freitag, Y. Al-Onaizan, and B. Sankaran, "Ensemble distillation for neural machine translation," *CoRR*, vol. abs/1702.01802, 2017.

[18] A. Waters and Y. Chebotar, "Distilling knowledge from ensembles of neural networks for speech recognition," in *Interspeech*, 2016.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2014. [Online]. Available: https://arxiv.org/abs/1409.2329

[21] P. Werbos, "Backpropagation through time: what does it do and how to do it," in *Proceedings of IEEE*, vol. 78, no. 10, 1990, pp. 1550–1560.