ENCRYPTED SPEECH RECOGNITION USING DEEP POLYNOMIAL NETWORKS

Shi-Xiong Zhang, Yifan Gong[†] *and Dong Yu*

Tencent AI Lab, Bellevue, USA
[†]Microsoft Corporation, Bellevue, USA
{auszhang,dyu}@tencent.com, ygong@microsoft.com

ABSTRACT

The cloud-based speech recognition/API provides developers or enterprises an easy way to create speech-enabled features in their applications. However, sending audios about personal or company internal information to the cloud, raises concerns about the privacy and security issues. The recognition results generated in cloud may also reveal some sensitive information. This paper proposes a deep polynomial network (DPN) that can be applied to the encrypted speech as an acoustic model. It allows clients to send their data in an encrypted form to the cloud to ensure that their data remains confidential, at mean while the DPN can still make frame-level predictions over the encrypted speech and return them in encrypted form. One good property of the DPN is that it can be trained on unencrypted speech features in the traditional way. To keep the cloud away from the raw audio and recognition results, a cloud-local joint decoding framework is also proposed. We demonstrate the effectiveness of model and framework on the Switchboard and Cortana voice assistant tasks with small performance degradation and latency increased comparing with the traditional cloud-based DNNs.

Index Terms— speech recognition, privacy preserving, encryption, DNN, quantization

1. INTRODUCTION

Modern speech recognition services are running on cloud [1]. These cloud-based speech-to-text API provides developers or enterprises an easy way to create powerful speech-enabled features in their own applications like voice-based command control, user dialog using natural speech conversation, and speech transcription and dictation. However, in the scenarios that involve medical, financial, or enterprise sensitive data, applying cloud-based speech recognition might be prohibited due to the privacy and legal requirements regarding the confidentiality of the information [2, 3, 4, 5, 6].

In this work we present a practical framework to allow third parties to use speech recognition services without sacrificing their privacy. In the proposed framework, the audio provider first encrypts the features extracted from the audio in local, then sends the features in encrypted form to the acoustic model on the cloud. We proposed a special form of DNNs, deep polynomial networks (DPN), as acoustic models, which can make predictions over the encrypted input features and yields the posteriors also in the encrypted form. Note that the server does not have the private key, so it is impossible for the server to decrypt the frame-level scores (posteriors) [6, 7]. Also to keep the cloud server away from the final recognition results, a cloud-local joint decoding framework is proposed. In this framework the encrypted frame-level scores from cloud will be sent back to the local owner who has the secret key for decryption. The decoding is done in local after the decryption. During the entire process no information is exposed to the cloud. It just performed a frame-level prediction on behalf of audio providers and learn nothing about the customer's data or the recognition results.

The main contributions of this work include: 1) a cloud-local joint decoding framework that enables practical privacy preserving speech recognition. 2) a deep polynomial network that can be trained on unencrypted data and make predictions over the encrypted inputs in real time. The remainder of the paper is organized as follows. In Section 2 we introduce the homomorphic encryption and the challenges of applying it to speech recognition. In Section 3 we describe the structures of polynomial networks for efficient homomorphic encryption. The experimental results will be discussed in Section 4.

2. ENCRYPTED SPEECH RECOGNITION

2.1. Homomorphic Encryption

One way to preserve the privacy of data is to use encryption. Traditional encryption lock data down in a way that makes it impossible to use, or compute on, unless we decrypt it. The homomorphic encryption (HE) allows people to use data in computations even while that data are still encrypted [8, 9] as illustrated in Figure 1. The encryption is called "homomorphic" because the transformation has the same effect on both the unencrypted and encrypted data. For example, suppose an encryption algorithm requires multiplying input numbers by 10 and the decryption requires dividing them by 10. This encryption is homomorphic for simple addition because "$1 + 2$"

978-1-5386-4658-8/18/$31.00 ©2019 IEEE

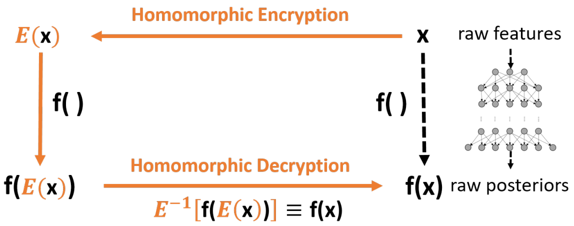


Fig. 1. The illustration of homomorphic encryption. The full homomorphic encryption is applicable to any arbitrary function $f(\cdot)$. For speech recognition, the $\mathbf{x}$ can be the input features and $f(\cdot)$ can be the acoustic models. The main challenge is how to run everything efficiently.

would be encrypted to "10 + 20", and decrypting the result by dividing it by 10 which would get to "3", as expected. A standard encryption algorithm, though, might turn the "1" and "2" into a smiley face and a semicolon. Adding such symbols is nonsensical, making computation impossible.

There are different degrees of homomorphic encryption, sometimes referred to as *fully* homomorphic compared with *partly*. In the previous paragraph, the example is only partly homomorphic because it works only for additions. The well known RSA algorithm [10] is also partly homomorphic. The fully homomorphic encryption allows for any arbitrary function $f(\cdot)$ to be performed on the encrypted data [11],

$$\mathsf{Enc}_k^{-1}\left[f(\mathsf{Enc}_k(\mathbf{x}))\right] \equiv f(\mathbf{x}) \tag{1}$$

where Enc_k and Enc_k^{-1} are homomorphic encryption and decryption respectively. The subscript k denotes the secret key. In theory, we could treat the acoustic models such as DNNs as the arbitrary function $f(\cdot)$ and apply the homomorphic encryption. In practice, high degree polynomial function $f(\cdot)$ requires the use of large parameters, which results in larger encrypted messages and slower computation [12]. Therefore, for efficiency, it is desired to restrict the acoustic model to degree-bounded polynomials, which only includes additions and multiplications. Thus, to make DNNs compatible with homomorphic encryption some modifications are needed. Details are discussed in Section 3.

Another limitation of homomorphic encryption is that it does not support floating-point numbers in practice. We have to use fixed-point real numbers and convert them to integers using the encoding approach in [6] for efficiency. We use 4-8 bits of precision on the weights of the network and inputs. To compensate the performance loss caused by low-bit quantization, a special training algorithm is proposed in Section 3.

The detailed description of encryption scheme, including the key generation algorithm, encryption and decryption algorithms, are beyond the scope of this paper and can be found in [7]. In the experiments, we use the implementation of *Simple Encrypted Arithmetic Library (SEAL)* [13] by Microsoft.[1]

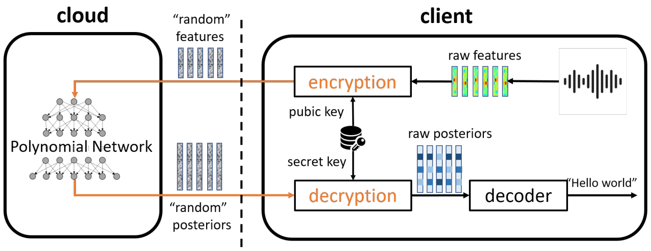


Fig. 2. The framework of encrypted speech recognition. The deep polynomial network is trained on unencrypted data with 8-bit quantization. During decoding the network is operated on encrypted space without knowing secret key.

2.2. Decoding Framework

In this section we propose a practical framework that enables clients and server to collaboratively decode the speech while satisfying their privacy constraints. During the entire process, the client has no access to the server model and the server has no idea about the input speech and recognition results as illustrated in Figure 2. First, the clients extract features from the audio in local. Second, the features are encrypted and sent to the cloud. Third, the acoustic models on the cloud evaluate the encrypted features and return the frame-level posteriors in encrypted form. Since the server does not have the secret key, it can not decrypt the posteriors nor decode the utterance. Finally, the client decrypt the frame-level scores and using (personal) language model to decode the utterance in local. These procedures are summarized in Table 1.

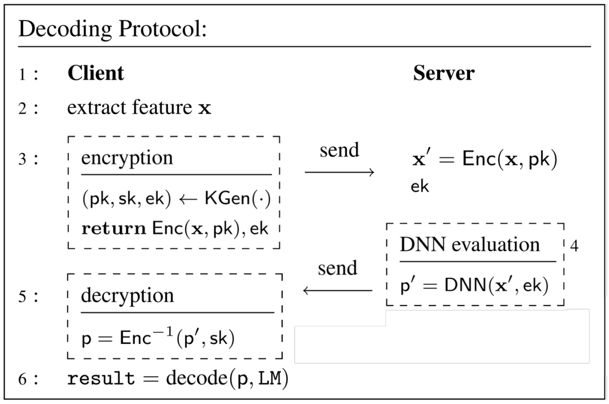
2.2.1. Why not run everything on local

In theory we could keep everything on local to maintain the privacy. In practice, the state-of-art acoustic models in production could be very large (for example up to 1 GB) and the computation can be very intensive. Some models even require specific hardware in order to decode speech in real time. It is impossible to run these models on local devices in real time. In addition, the acoustic models in production usually got updated fairly often. It could be a model update retrained with more recent data, or with more powerful structures. It is much easier to deploy the new model if it is run on cloud.

2.2.2. Why not run everything on cloud

For efficiency we would like to move the decoder and language model to the cloud as well, so that the client only need to do the decryption to get the recognition results. Some recent work shown that it is possible to apply the Viterbi search operation over homomorphic encrypted data [2, 14], however, all these papers assume that no pruning is applied while searching in encrypted domain. In practice, speech recognition is never preformed in this manner. On the other hand, pruning may restrict the hypothesis set and reveal information about the recognition output. How to prune in encrypted domain and hide this information is still an open problem.

[1] The SEAL library is available at `http://sealcrypto.org/`.



Decoding Protocol:

	Client		Server
1 :	**Client**		**Server**
2 :	extract feature $\mathbf{x}$		
3 :	encryption $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{KGen}(\cdot)$ **return** $\mathsf{Enc}(\mathbf{x}, \mathsf{pk}), \mathsf{ek}$	send $\longrightarrow$	$\mathsf{x}' = \mathsf{Enc}(\mathbf{x}, \mathsf{pk})$ ek
4			DNN evaluation $\mathsf{p}' = \mathsf{DNN}(\mathsf{x}', \mathsf{ek})$
5 :	decryption $\mathsf{p} = \mathsf{Enc}^{-1}(\mathsf{p}', \mathsf{sk})$	send $\longleftarrow$	
6 :	$\mathtt{result} = \mathrm{decode}(\mathsf{p}, \mathtt{LM})$		

Table 1. The decoding protocol for encrypted speech recognition. $(\mathsf{pk}, \mathsf{sk})$ denotes for a public/secret key pair, ek is the evaluation key, and p is the frame-level posteriors.

3. DEEP POLYNOMIAL NETWORKS

In this section we propose a special form of DNNs as acoustic models to make predictions over the encrypted input features. In Section 2.1 we conclude that certain polynomial functions can be computed over encrypted data given that their degree is not too large. However, some operations in neural networks are not polynomials, such as sigmoids and ReLU activation functions [15] and max pooling [16]. Here we listed some of common operations in DNNs (including CNNs) and their approximations in polynomials.

Dense Layer This is just linear multiplication and addition. It can be directly implemented for homomorphic encryption without approximation. Note the weights $\mathbf{W}$ in dense layer are fixed and not encrypted during decoding. Given encrypted inputs $\mathsf{Enc}(\mathbf{x})$, a naive way to compute dense layers is to first encrypt the weights and then perform the multiplication in encrypted domain $\mathsf{Enc}(\mathbf{W})^\mathsf{T}\mathsf{Enc}(\mathbf{x})$, so that after decryption we can get the exact value of $\mathbf{W}^\mathsf{T}\mathbf{x}$. However, this process is computationally intensive and not necessary. Instead, we use a more efficient plain operation $\mathbf{W}^\mathsf{T}\mathsf{Enc}(\mathbf{x})$. Because of this, during decryption, we need to keep in mind that the random noise bias in encryption scheme has been scaled by $\mathbf{W}$.

Batch Norm This is also multiplications and additions [17] $\mathsf{BN}(\mathbf{x}) = \boldsymbol{\gamma}\frac{\mathbf{x}-\boldsymbol{\mu}}{\boldsymbol{\sigma}} + \boldsymbol{\beta}$, which can be directly implemented for homomorphic encryption without approximation. Note since $\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are fixed during decoding, there is no need to encrypt these parameters or to compute batch norm explicitly. Instead, we merge these parameters to the preceding dense layer, which results to new weights $\mathbf{W}^{\mathtt{new}} = \mathtt{diag}(\frac{\boldsymbol{\gamma}}{\boldsymbol{\sigma}})\mathbf{W}$ and bias $\boldsymbol{b}^{\mathtt{new}} = \boldsymbol{b} + \mathbf{W}^\mathsf{T}\boldsymbol{\beta} - \mathbf{W}^\mathsf{T}\frac{\boldsymbol{\mu}\cdot\boldsymbol{\gamma}}{\boldsymbol{\sigma}}$ for the preceding layer.

ReLU This activation function $z \mapsto \max(0, z)$ can be approximated with $\mathsf{p}(z) := z^2$. Note there is a theoretical study of the problem of learning neural networks with polynomial activation functions in [18], where the above square function was also used to replace ReLU.

Algorithm 1: Quantized training for DPN

```
Data: unencrypted training data.
Result: quantized deep polynomial network.
convert pretrained DNN to DPN: $\mathbf{W}_{\mathtt{DPN}} \leftarrow \mathbf{W}_{\mathtt{DNN}}$
for each minibatch do
    update $\mathbf{W}_{\mathtt{DPN}}$ in floating point
for each layer $[l]$ do
    optimize codebook: $\boldsymbol{c}^{[l]}_{1:256} \leftarrow \mathsf{distribution}(\mathbf{W}^{[l]}_{\mathtt{DPN}})$
    quantization: $\tilde{\mathbf{W}}^{[l]}_{\mathtt{DPN}} \leftarrow \mathsf{Quant}(\mathbf{W}^{[l]}_{\mathtt{DPN}}, \boldsymbol{c}^{[l]}_{1:256})$
for each minibatch do
    update $\tilde{\mathbf{W}}_{\mathtt{DPN}}$ with on-the-fly quantization based on $\boldsymbol{c}$
```

Sigmoid This activation function $z \mapsto \frac{1}{1+e^{-z}}$ can be approximated with low-degree polynomials $\mathsf{p}(z) := \frac{1}{2} + \frac{1}{4}z - \frac{1}{48}z^3$.

Convolution The operation is essentially a dot product of the weight vector (kernel) and the vector of feeding layer outputs. Therefore it is compatible with homomorphic encryption.

Max Pooling This operation cannot be computed directly as it is non-polynomial. However, it can be approximated using $\max(z_1, \ldots, z_n) = \lim_{d\to\infty}(\sum_{i=1}^{n} z_i^d)^{1/d}$. For efficiency, we use $d = 1$ which leads to a scaled mean pooling.

Using the above approximations all the layers and operations in the traditional DNNs and CNNs becomes polynomial. We named the resulting models as deep polynomial networks.

3.1. Training with Quantization

In Section 2.1 we discussed that the homomorphic encryption in practice only supports fixed-precision real numbers or integers.[2] However the state-of-the-art DNNs/CNNs are typically trained on GPUs with 32-bit floating point. Applying low-bit fixed-point quantization directly to the model parameters during decoding could cause substantial performance drop. Here a quantized retraining algorithm is proposed. In Section 4 We show that the DNNs, CNNs and DPNs can all be trained using 8-bit quantization with almost no WER increase.

Unlike floating point has an universal standard, the fixed-point numbers are domain specific. Each task has to design its own scheme for fixed-point quantization [19, 20]. Our quantization scheme has three features. 1) $0.\mathtt{f}$ is always kept as one of 256 (8-bit) quantized values. This is because 0 has a special significance in CNNs such as zero padding. 2) By analyzing the histogram of parameters, we found most values are concentrated in a small range. This implies instead of using uniform quantization, we should put more codebooks for these concentrated region. The Lloyd-max quantization $\mathsf{Quant}(\cdot)$ is adopt to find the optimal codebooks $\{c_1, \ldots, c_{256}\}$ and bin-boundaries $\{l_1, \ldots, l_{257}\}$ [21]. 3) When training deep neural networks, the parameters, activations and gradients have very

[2] The SEAL library can encode the fixed-precision real numbers into integers for efficient encryption during decoding [7].

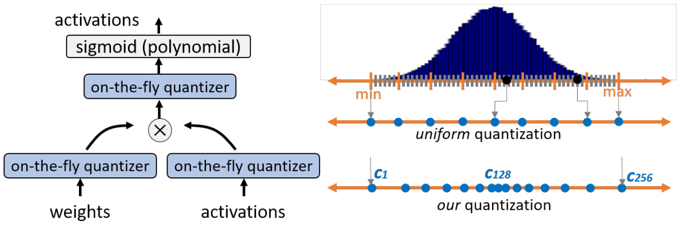


Fig. 3. Quantization scheme in training over raw data.

different ranges. For example gradient's ranges slowly diminish during the training. Therefore, we draw the distributions and compute the codebook for each layer's weights, bias, activations and respective gradients separately. The training process is illustrated in Algorithm 1 and Figure 3.

3.1.1. *Why not train on encrypted data*

Note one important feature of this framework is that the DPN can be trained on unencrypted data and applied to encrypted data. In theory, it is possible to also train the DPN over encrypted data as its gradients are polynomials as well. However, in practice this is not scalable as it is very expensive to encrypt the entire training corpus and compute the gradient in encrypted domain. Quantization will also be an challenge if training on encrypted data. Besides, the learning algorithm does not have access to the secret key for decryption, we will never know what these trained weights are.

4. EXPERIMENTS

In this section we present the details of network architectures, practical considerations for training and decoding, and experimental results. All the models investigated in this work were trained using the computational network toolkit (CNTK) [22]. The homomorphic encryption is implemented using the SEAL library [13]. We evaluate the effectiveness of the DPN proposed in Section 3 on the Switchboard and Cortana voice assistant tasks.

In Switchboard task, we use 309hr training set and NIST 2000 Hub5 as test set. The features used in this set up is 40-dimensional LFB with utterance-level CMN. The outputs of network are 9000 tied triphone states. We verified the polynomial approximation on two models, DNN and CNN. The DNN is a 6-layer ReLU network with batch normalization and 2048 units on each layer. The CNN is a 17-layer VGG network [23, 24], including $3\times\texttt{conv}(3,3,96)$[3], max-pooling, $4\times\texttt{conv}(3,3,192)$, max-pool, $4\times\texttt{conv}(3,3,384)$, max-pool, followed by two dense layer with 4096 units and softmax layer. Both models use $[t-30, t+10]$ as the input context. The LM used in this task is from [25]. The vocabulary size is 226k. The WERs of above DNN and CNN and the corresponding DPNs are shown in Table 2. All models are trained using CE criteria. We leave the sequence training [26, 27] for these models as the future work.

WER in %		16-bit	8-bit	4-bit	2-bit
DNN	quantization	14.7%	14.9%	16.6%	100.4%
	+ retrain	–	14.7%	14.9%	30.3%
DNN→DPN (Alg. 1)		15.8%	15.8%	16.1%	30.8%
CNN	quantization	12.2%	12.7%	15.4%	–
	+ retrain	–	12.3%	12.7%	–
CNN→DPN (Alg. 1)		13.5%	13.6%	14.0%	–

Table 2. The results of DNN, CNN and DPN on switchboard. Note we could not make recurrent models work for homomorphic encryption.

In Cortana voice assistant task, we used about 3400 hours US-English data in training and 6 hours data (5500 utterances) for testing. The features used in this setup is 87-dim LFB (including 29-dim static, Δ and $\Delta\Delta$) with utterance-level CMN. The networks used in this setup have the same structure as above, but with 9404 tied triphone states. Table 3 summarizes the WER and the average latency per utterance (including encryption, AM scoring, decryption and decoding) on this Cortana task.

			avg. latency per utterance		
	16-bit	4-bit	encryption	decryption	overall
DNN	12.9%	13.4%	–	–	177ms
DPN	14.8%	15.5%	202ms	16ms	373ms

Table 3. The performance of DNN and DPN on Cortana task. The overall latency of DPN includes encryption, AM scoring, decryption and decoding based on 4-bit model.

5. CONCLUSION AND FUTURE WORK

The cloud-based SR service empowers users or third-parties to try state-of-art speech recognition easily in their own tasks. However, sending audios about personal or company internal information to the cloud, raises concerns about privacy. The main contributions of this work include: 1) a cloud-local joint decoding framework that enables privacy preserving speech recognition. It allows users to send their data in an encrypted form to ensure that their data remains confidential, at mean while the server can still do speech recognition on their behalf without knowing the content. 2) a deep polynomial network that can be trained efficiently on unencrypted data and make predictions over the encrypted speech in real time. We illustrate the effectiveness of model and framework on the Switchboard and Cortana voice assistant tasks with acceptable performance degradation and latency increased comparing with the traditional cloud-based DNNs. Future works will include 1) making the decoder also works on encrypted domain so that we could move everything to the cloud. 2) investigating training on encrypted data so that multiple parties (e.g. Microsoft, Google and Amazon) can encrypt and combine their data together to train models without sacrificing users privacy.

[3] $\texttt{conv}(3,3,96)$ means kernel size is $(3,3)$ and that layer has 96 kernels.

6. REFERENCES

[1] Xuedong Huang, James Baker, and Raj Reddy, "A historical perspective of speech recognition," *Communications of the ACM*, vol. 57, no. 1, pp. 94–103, 2014.

[2] Manas A Pathak, Bhiksha Raj, Shantanu D Rane, and Paris Smaragdis, "Privacy-preserving speech processing: cryptographic and string-matching frameworks show promise," *IEEE signal processing magazine*, vol. 30, no. 2, pp. 62–74, 2013.

[3] Paris Smaragdis and Madhusudana Shashanka, "A framework for secure speech recognition," in *ICASSP*. IEEE, 2007, vol. 4, pp. IV–969.

[4] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N Wright, "Privacy-preserving machine learning as a service," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 3, pp. 123–142, 2018.

[5] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in bioinformatics*, 2017.

[6] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *ICML*, 2016, vol. 48, pp. 201–210.

[7] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing, "Manual for using homomorphic encryption for bioinformatics," *Proceedings of the IEEE*, vol. 105, no. 3, pp. 552–567, 2017.

[8] Louis JM Aslett, Pedro M Esperança, and Chris C Holmes, "A review of homomorphic encryption and software tools for encrypted statistical machine learning," *arXiv preprint arXiv:1508.06574*, 2015.

[9] Junfeng Fan and Frederik Vercauteren, "Somewhat practical fully homomorphic encryption.," *IACR Cryptology ePrint Archive*, vol. 2012, pp. 144, 2012.

[10] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[11] Craig Gentry and Dan Boneh, *A fully homomorphic encryption scheme*, vol. 20, Stanford University Stanford, 2009.

[12] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan, "Can homomorphic encryption be practical?," in *ACM Cloud Computing Security Workshop CCSW 2011*. January 2011, vol. 2011, p. 405, ACM.

[13] "Simple Encrypted Arithmetic Library (release 3.0.0)," `http://sealcrypto.org`, Oct. 2018, Microsoft Research, Redmond, WA.

[14] Mehrdad Aliasgari, Marina Blanton, and Fattaneh Bayatbabolghani, "Secure computation of hidden markov models and secure floating-point arithmetic in the malicious model," *International Journal of Information Security*, vol. 16, no. 6, pp. 577–601, 2017.

[15] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, 2013, vol. 30, p. 3.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[17] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[18] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir, "On the computational efficiency of training neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 855–863.

[19] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy, "Fixed point quantization of deep convolutional networks," in *International Conference on Machine Learning*, 2016, pp. 2849–2858.

[20] Song Han, Huizi Mao, and William J Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[21] Paul Scheunders, "A genetic lloyd-max image quantization algorithm," *Pattern Recognition Letters*, vol. 17, no. 5, pp. 547–556, 1996.

[22] Dong Yu, Adam Eversole, Mike Seltzer, Kaisheng Yao, Zhiheng Huang, Brian Guenter, Oleksii Kuchaiev, Yu Zhang, Frank Seide, Huaming Wang, et al., "An introduction to computational networks and the computational network toolkit," *Microsoft Technical Report MSR-TR-2014–112*, 2014.

[23] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al., "English conversational telephone speech recognition by humans and machines," *arXiv preprint arXiv:1703.02136*, 2017.

[24] Shi-Xiong Zhang, Zhuo Chen, Yong Zhao, Jinyu Li, and Yifan Gong, "End-to-end attention based text-dependent speaker verification," in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 171–178.

[25] Dong Yu, Wayne Xiong, Jasha Droppo, Andreas Stolcke, Guoli Ye, Jinyu Li, and Geoffrey Zweig, "Deep convolutional neural networks with layer-wise context expansion and attention.," in *Interspeech*, 2016, pp. 17–21.

[26] Shi-Xiong Zhang, Chaojun Liu, Kaisheng Yao, and Yifan Gong, "Deep neural support vector machines for speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4275–4279.

[27] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey, "Sequence-discriminative training of deep neural networks.," in *Interspeech*, 2013, pp. 2345–2349.