Large Vocabulary Concatenative Resynthesis

Soumi Maiti[1], *Joey Ching*[2], *Michael Mandel*[1,2]

[1]Computer Science Program, the Graduate Center, City University of New York
[2]Computer and Information Science, Brooklyn College, City University of New York
smaiti@gradcenter.cuny.edu, mim@sci.brooklyn.cuny.edu

Abstract

Traditional speech enhancement systems reduce noise by modifying the noisy signal, which suffer from two problems: under-suppression of noise and over-suppression of speech. As an alternative, in this paper, we use the recently introduced concatenative resynthesis approach where we replace the noisy speech with its clean resynthesis. The output of such a system can produce speech that is both noise-free and high quality. This paper generalizes our previous small-vocabulary system to large vocabulary. To do so, we employ efficient decoding techniques using fast approximate nearest neighbor (ANN) algorithms. Firstly, we apply ANN techniques on the original small vocabulary task and get $5\times$ speedup. We then apply the techniques to the construction of a large vocabulary concatenative resynthesis system and scale the system up to $12\times$ larger dictionary. We perform listening tests with five participants to measure subjective quality and intelligibility of the output speech.

1. Introduction

Environmental noise is one of the biggest problems for speech processing systems. Traditional noise suppression algorithms modify the noisy signal to make it more like the original signal. In doing so, they also inevitably reduce the quality of the speech [1]. Using the recently proposed concatenative resynthesis approach [2, 3], we instead resynthesize the clean signal using clean speech segments to replace noisy segments. The resynthesized signal should be both high quality and noise free. Such a system works very well for small vocabulary tasks in both removing noise and increasing quality of distorted signals [2].

The core of such a system is a similarity network that can predict a similarity score between a clean and noisy segment of audio. Given a noisy segment and a dictionary of clean segments, we can replace the noisy segment with most similar clean dictionary segment. But in order to improve the quality of the resyntheses further, we utilize a transition affinity to encourage smoother transitions between chunks. The optimal resynthesis is thus found using the Viterbi algorithm to balance similarity between clean and noisy chunks and compatibility between consecutive clean chunks.

Our previous work [4] proposed a model configuration that embeds the clean and noisy speech into a shared low-dimensional space where matching clean and noisy chunks have similar embeddings and non-matching pairs have dissimilar embeddings. This configuration allows the clean speech to be processed offline so only the noisy speech needs to be processed at runtime. While faster, the time to identify an appropriate clean segment is still linear in the size of the dictionary.

In the current work, we thus propose to make this time sublinear by using approximate nearest neighbor algorithms in the decoding process to efficiently identify candidate clean segments for each noisy segment in this learned low-dimensional embedding space. In addition, we use ANN techniques to accelerate the computation of the affinity transition matrix. These two changes allow the system to be scaled to large vocabulary tasks. We experiment here on a new large noisy dataset by mixing audiobooks from the Blizzard 2013 speech synthesis challenge [5] with environmental noise from the CHiME-3 challenge [6].

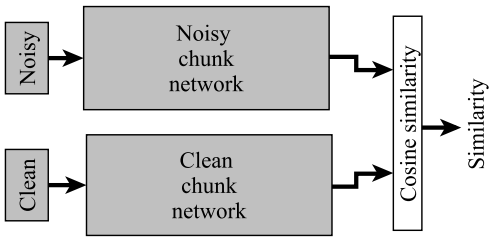

Figure 1: *Twin networks embed chunks of clean and noisy audio into a shared low dimensional space allowing fast search.*

Using these ANN techniques, we first show a $5\times$ (40 s vs 8 s) speedup compared to the brute force approach in a small vocabulary dataset. The time to construct the transition matrix also reduces by a factor of $40\times$ (1808 s vs 45 s). We then scale the system using different dictionary sizes (N) up to 746k ($12\times$ larger dictionary) and measure intelligibility and quality against two comparison models.

2. Technical Description

Noisy and clean utterances are divided into temporally overlapped "chunks" of duration 192 ms. We assume that each noisy chunk is constructed by adding noise to one clean chunk. A clean dictionary is built from all of the clean speech chunks $\{x_i\}_{i=1}^{N}$. We build noisy observation sets from noisy chunks $\{z_j\}_{j=1}^{M}$. Using the similarity network, we compute a similarity matrix between clean and noisy chunks ($S_{N\times M}$). In addition, We build a transition affinity matrix between clean dictionary chunks ($T_{N\times N}$). The similarity matrix and transition matrix are used to find an optimal resynthesis.

2.1. Similarity Network

The similarity network predicts a similarity score (between 0 and 1) between a clean and noisy chunk. We use twin networks [4] to separately map a clean and noisy chunk into a shared low-dimensional embedding space where matching chunks are close to each other. Then cosine similarity between low-dimensional embeddings is used as a similarity score (Figure 1). Twin networks have separate identical sub-networks for processing clean and noisy chunks. The subnetworks do not share weights and this allows them to process clean and noisy speech segments differently. The twin architecture allows the system to process the clean dictionary only once and store the embeddings. At

runtime it only needs to process the noisy observed chunks through the noisy network. The twin networks are trained with the contrastive loss according to [7].

$$L(y, \hat{y}) = (1-y)\frac{1}{2}\hat{y}^2 + y\frac{1}{2}\{\max(0, m-\hat{y})\}^2 \quad (1)$$

where $\hat{y}$ is the predicted similarity score, y is desired similarity score, and m is a margin parameter.

2.2. Efficient Decoding

The brute force approach for decoding is to compute the full similarity matrix, $S_{N\times M}$, between all pairs of clean and noisy chunks and then search for best path using the Viterbi algorithm. Since this matrix is generally sparse, we use fast ANN algorithms to compute an approximate version of $S_{N\times M}$ in much less time. Several efficient libraries exist for finding approximate nearest neighbors, including ANNOY [8] and NMSLib [9]. ANNOY builds a forest of random projection trees to index the data and searches all trees in parallel. NMSLib implements several algorithms, of which we use hierarchical navigable small world (HNSW) graphs. This approach creates a graph with small-world structure (where it is possible to travel between most points in a small number of hops) connecting indexed points and then uses a greedy search from random initialization points to search it. We compare these two ANN algorithms against each other and against brute-force computation.

2.3. Approximating transition affinities

The transition affinity matrix $T(i,j)_{i,j=1}^{N}$ is defined as the probability of transitioning from dictionary element x_i to dictionary element x_j. The affinity is currently computed based on acoustic features only. We calculate the euclidean distance, d_τ, between the log mel spectrogram of the last τ frames of x_i and first τ frames of x_j as the affinity,

$$T(i,j) = \frac{\exp(\frac{1}{\gamma}d_\tau(x_i, x_j))}{\sum_{j'}\exp(\frac{1}{\gamma}d_\tau(x_i, x_{j'}))} \quad (2)$$

where γ controls the mapping of distances to affinities and was tuned on development data.

Computing this matrix is very expensive (quadratic in N) and generally results in a very sparse matrix. Storing the whole transition matrix is not feasible, even for small vocabulary tasks. On one such task, with a $61k$-element dictionary, we have had success only storing the largest 10^7 entries, which is 0.28% of the full matrix. For large vocabularies, even computing the whole matrix is not feasible. Instead, we use ANN methods to find many of the non-zero entries in this matrix, specifically, the k most compatible transitions for each chunk. We explore different configurations of ANN algorithms and measure their trade-off between recall and speed.

Given a noisy speech chunk z_j, the concatenative resynthesis system identifies a matching clean chunk x_i from the dictionary. To evaluate the quality of the mapping, $z \rightarrow x$, we compute the accuracy of the frame-level phonetic transcriptions of x to z:

$$a_f(z, x) = \frac{1}{F}\sum_{k=1}^{F}\delta(p_k^{(z)}, p_k^{(x)}) \quad (3)$$

where p_k is the phonetic label of the k^{th} frame in a signal. Thus, a_f considers the frame-wise phonetic correspondence of the input and output chunks. We consider a_f as our objective accuracy metric.

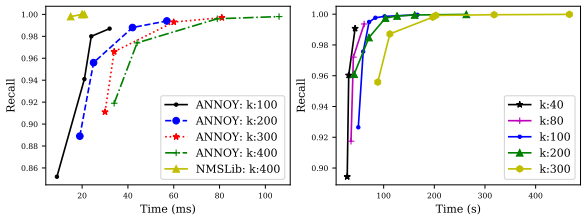

Figure 2: *(a) Recall versus time to compute a 500 × 61k similarity matrix using ANN. Brute-force computation takes 670 ms. (b) Recall versus time to build a 61k × 61k transition matrix using ANNOY. Brute force computation time is 1808.3 s.*

3. Small-vocabulary experiments

We first experiment with efficient decoding and approximate transition affinities on the small-vocabulary CHiME-2 Track 1 (GRID speech) dataset [10]. This enables us to directly measure the efficacy of both ANN algorithms on a task where brute force solutions are still possible.

This small dataset contains read speech simulated in a living room environment. Noises are mainly the speech of women and children, music, and various household activities. We train and test on different utterances with the same speaker (speaker 3, male), using 490 utterances for training and 10 utterances for testing. We calculate log mel spectrograms and extract 11 frame chunks that overlap with their neighbors by 10 frames. Each chunk contains 192 ms of audio. For resynthesis, we build the clean dictionary using 490 clean utterances, giving us a total of 60,801 clean chunks ($N \approx 61k$).

3.1. Efficient decoding using ANN

First, we compare efficient decoding techniques using ANN against brute force. We use two ANN algorithms: NMSLib and ANNOY. For each noisy chunk we retrieve the top k clean dictionary chunks and by varying k we can vary the sparsity of the similarity matrix (S).

For ANN algorithms, there is a trade-off between recall and computation time. It is possible to improve recall performance at the cost of higher computation time. Figure 2 shows this recall-computation trade-off for ANN algorithms for a random set of 500 noisy chunks in the shared embedding space. Using NMSLib gives nearly perfect recall in very little computation-time ($\sim$ 15 ms) over different k (in the figure, we show the highest $k = 400$). Hence, the computation is $44\times$ (670 ms vs 15 ms) faster than brute force. Although we must note that NMSLib needs to be tuned for good results. ANNOY can achieve good recall at the expense of considerable computation-time, but it is easier to configure and use than NMSLib.

3.2. Approximating the transition matrix

Next, we compare the ANN approximation of the transition matrix with brute force. The brute force approach takes 1808.3 s to build the matrix for $N = 61k$ and we store the largest 10^7 entries. We use ANNOY to retrieve top k dictionary chunks for each noisy chunk. Hence, the building time becomes linear in dictionary size compared to quadratic in brute force. Due to a bug in NMSLib we were unable to use NMSLib here. Figure 2 shows the recall-computation time trade-off for different values of k. At a recall of 97%, it takes 45 seconds to build the transition matrix for $k = 100$, which gives a $40\times$ speedup over brute force.

Table 1: *Effect of sparsity in similarity matrix on accuracy and per-file denoising time for small dictionary (size 61k) task*

Method	k	Acc (%)	Time (s)
Brute force	-	64.4	40.85
trans+dec	100	58.3	8.63
trans+dec	200	59.8	8.71
trans+dec	300	60.9	8.27
trans+dec	400	64.6	8.74
trans+dec	500	63.4	8.46
trans+dec	700	61.6	7.95

3.3. Denoising

After tuning ANN for both efficient decoding and transition matrix computation, we measure their effect on objective accuracy and denoising time of the system. We expect to achieve faster denoising with comparable accuracy to brute force. The transition matrix is built with ANN with $k = 100$.

Table 1 shows the average resynthesis time and accuracy for different choices of k. on an average, taken over 10 files, processing time for 2 s of speech is 8.7 s. This is $5\times$ speedup compared to brute force (40.8 s). The average accuracy of the resynthesized signals increases with k. At $k = 400$, it is interesting to note that the accuracy is similar to that of the brute force approach. Hence we can build fast concat systems with comparable accuracy for this small-vocabulary task.

4. Large-vocabulary experiments

In the second experiment, we use efficient decoding and approximate transition affinities to scale the system to large vocabularies. We conduct experiments on a new large vocabulary single-channel noisy speech dataset. The dataset is created from existing speech and noise datasets but unlike other speech enhancement datasets, ours uses a large amount of speech from a single speaker. We use the audio book narration speech from the 2013 Blizzard Challenge [5]. This dataset consists of approximately 50 audio books read by a single narrator, approximately 300 hours of speech. The speech is compressed using the MP3 codec at bit rates between 64 kbps and 128 kbps including frequencies up to 22 kHz. Preliminary listening tests on files at each bit rate showed that they all achieved high speech quality. Each audio book is broken into a separate MP3 file per chapter, with each chapter being 10-15 minutes long. Though there is no accompanying text for the books, many of them are available from Project Gutenberg. After obtaining the texts from Project Gutenberg, we segmented them into chapters corresponding to the audio files in a semi-manual way and used the Gentle forced alignment tool [11] to align the text with the recordings. The current experiments use the audio book narration of *Sense and Sensibility* by Jane Austin.

The noise comes from the CHiME-3 dataset [6], which consists of seven hours of 6-channel recordings. The noises are recorded in four environments: bus, café, pedestrian area, and street. We treat each of the six channels as a separate noise recording. For each audio book chapter, a random segment of a random environmental noise is selected and mixed using a constant gain of 0.95. The average signal-to-noise ratio (SNR) of the resulting noisy files is 3 dB, with a maximum SNR of 9 dB and a minimum of -4 dB.

Sense and Sensibility has 50 chapters and we use 40 for training/dictionary building and the other 10 for testing. Using 40 chapters results in 2,306,996 clean chunks. We select 1,000,000 matching clean-noisy pairs and 1,000,000 non-matching clean-noisy pairs, making a total training set of 2,000,000 pairs. To select representative test utterances, we searched for small to medium sentences or smaller parts of longer sentences from the test set that consist of words occurring more than 40 times in the training set. We use 62 of these sentences to test, each of which is 4–11 words long with a duration of 2–4 seconds.

Table 2: *Ranking performance test*

	Euclidean	Concat
Precision-at-1 (higher better)	12.6%	82.6%
Avg rank of correct chunk (lower better)	2522	10
Number of dictionary chunks	7972	7972

We compare the performance of the network directly by measuring their ranking performance on clean dictionary elements when the correct element is present. This approach allows us to directly quantify the performance of the system.

4.1. Ranking Test

We use the selected 62 test sentences and build a dictionary where for each noisy speech chunk there is exactly one matching clean chunk. We then randomly select 500 noisy chunks from the test sentences to evaluate. The exact matching clean chunk serves as our "ground truth". The baseline is the Euclidean distance between the log mel spectrum of clean and noisy chunks. We predict the similarity of all of clean speech dictionary elements for each noisy input and rank the clean chunks by their similarity. We measure the average precision-at-1, i.e., the percentage of queries where the top-ranked dictionary element is the actual matching clean element (higher better). We also measure the average rank (lower better) of the correct dictionary chunk for each model. Table 2 shows the ranking performance results.

The euclidean baseline gives 12.6% precision-at-1 and the average rank of the correct chunk is 2522 out of 7962. The precision-at-1 for our system is 82.6% with average rank of correct chunk at 10.

4.2. Denoising using increasing dictionary size

Next, we test the scalability of the system by increasing the dictionary size N to 746k and measuring the objective accuracy and denoising time, shown in Table 3. The system is more scalable if the denoising time is faster while objective accuracy is comparable. We select 8 test noisy sentences at random for denoising. To vary the dictionary size, we build two types of dictionaries. In the first type, we select small audio segments from the training set containing the words in the test set. We control the number of occurrence (n) of such words and change N, these are named Word$\langle n \rangle$. In the second type, we build a dictionary from a number of audio book chapters (m) directly, they are named Chp$\langle m \rangle$. Although, there is no guarantee that the Chp$\langle m \rangle$ dictionaries will have possible matching clean chunks for our test set, we can still measure the performance of the system. For large N, we first use the transition approximation only, so we expect to see higher denoising time, but also higher accuracy. Next, we apply both approximate transition and approximate similarity to decrease the denoising time.

For 8 speech files, each 2-4 s long, it is found that using trans approximation only average denoising time per file is high (14 mins) for the largest dictionary. On the other hand, by ap-

Table 3: *Effect of large dictionary size (N) on per-frame phonetic accuracy (Acc) and denoising time averaged over 8 files (Time). Using approximate transition (Trans) only and both approx transition and similarity (Trans+Sim)*

		Trans		Trans+Sim	
N	Dict	Acc (%)	Time (s)	Acc (%)	Time (s)
49k	Word10	45.9	11.0	42.8	7.8
178k	Chp4	48.9	79.3	48.5	15.5
186k	Word25	52.9	53.3	53.0	16.1
277k	Word40	50.3	103.7	53.1	39.1
475k	Chp10	50.9	403.8	49.7	36.6
746k	Chp15	51.0	837.6	50.7	75.0

proximating the transition and similarity matrices, the processing time reduces significantly to 75.04 seconds with minimal loss of frame-wise accuracy (speedup $= 837.6/75 \approx 11\times$). Hence, we were able to scale the dictionary up to 746k with denoising time of 75.04 s.

4.3. Listening Test

Finally, to evaluate subjective quality and intelligibility of the complete system, we perform listening tests. We resynthesize the clean speech from noisy test sentences using the Word25 dictionary and measure the subjective quality and intelligibility of the outputs compared with those of two models. One model predicts the ideal ratio mask from the log mel spectrum of the noisy speech as a classification task (Classification). The second one predicts the log mel spectrum of the clean speech from the noisy speech (Regression). These two comparison models were trained using same data as our concatenative resynthesis system (Concat). In total there are five versions of the test files, the original clean speech (Clean), the noisy mixture (Noisy), Concat, Classification, and Regression. There are 8 files from each of these systems, making a total of 40 files. Five listeners participated in both intelligibility and quality tests.

The speech quality test compared these systems under a Multiple Stimuli with Hidden Reference and Anchor (MUSHRA) paradigm [12]. For each mixture, listeners were first presented with the reference clean and noisy speech and then with the output of the systems. The systems are unlabeled and presented in a random order. Listeners were asked to rate each mixture in terms of speech quality, noise suppression, and overall quality each on a scale from 0 (poor) to 100 (excellent). Figure 4 shows the results of the speech quality test for each system, averaged across files and listeners. All three quality measures are highest for both the clean speech reference and the hidden clean speech. Speech quality is also very high for the noisy speech. Of the enhancement systems, Concat has better speech, noise suppression and overall quality than Classification and Regression.

For the intelligibility test, participants listened to all 40 files in different random orders. Subjects were given a copy of the possible vocabulary and asked to transcribe the sentences as best they could, noting that they did not necessarily have to adhere to the vocabulary. Figure 3 shows the results of the intelligibility test for each system averaged over all files and participants. Speech intelligibility for both clean and noisy utterances is very high. Intelligibility is worst for Concat and best for Classification. These results show that the large vocabulary Concat system achieves better quality with slightly lower intelligibility than the baseline systems. They also show that the system achieves comparable quality and intelligibility to earlier listening tests on

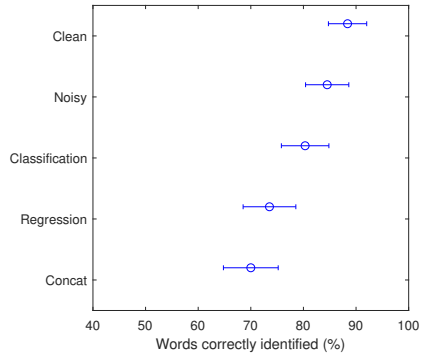

Figure 3: *Large vocabulary intelligibility test results showing the average percentage of 62 test words correctly identified per system. Error bars show 95% confidence intervals.*

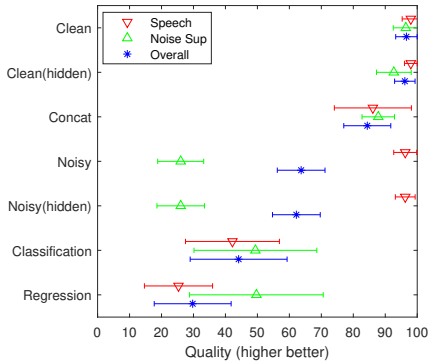

Figure 4: *Large vocabulary MUSHRA quality evaluation listening test results showing speech quality, noise suppression quality and overall quality judgments for each system.*

small vocabulary tasks [2], but with better runtime performance.

5. Conclusions

In this work, we introduce two ways to increase the efficiency of concatenative resyntheses systems using approximate nearest neighbor methods. In a small vocabulary system, we show a speedup of $40\times$ in transition matrix computation time and $5\times$ in denoising time. The system is thus scalable to large dictionaries and we show that using a $12\times$ larger dictionary, we can achieve a speedup of $11\times$ in resynthesis. With 8 files, each 2-4 s long, we have reduced the processing time considerably, from 14 minutes to 75 s per file. Finally, We perform listening tests to show that the large dictionary system has similar output quality and intelligibility to the small dictionary system. Future work will improve the phonetic accuracy of the large vocabulary system using more flexible training signals based on clean speech similarity, as opposed to the exact match criteria used here [13].

6. Acknowledgements

This material is based upon work supported by the National Science Foundation (NSF) under Grant IIS-1618061. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

7. References

[1] J. Chen, J. Benesty, Y. Huang, and S. Doclo, "New insights into the noise reduction wiener filter," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1218–1234, 2006.

[2] M. I. Mandel, Y.-S. Cho, and Y. Wang, "Learning a concatenative resynthesis system for noise suppression," in *Proceedings of the IEEE GlobalSIP conference*, 2014.

[3] M. I. Mandel and Y. S. Cho, "Audio super-resolution using concatenative resynthesis," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2015.

[4] S. Maiti and M. I. Mandel, "Concatenative resynthesis using twin networks," in *Proceedings of Interspeech*, 2017, pp. 3647–3651.

[5] S. King and V. Karaiskos, "The blizzard challenge 2013," in *Blizzard Challenge Workshop*, 2013.

[6] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third chime speech separation and recognition challenge: Dataset, task and baselines," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2015, pp. 504–511.

[7] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1735–1742.

[8] E. Bernhardsson, "ANNOY: Approximate nearest neighbors in C++/Python optimized for memory usage and loading/saving to disk," 2013, https://github.com/spotify/annoy.

[9] Y. A. Malkov and D. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *arXiv preprint arXiv:1603.09320*, 2016.

[10] E. Vincent, J. Barker, S. Watanabe, J. Le Roux, F. Nesta, and M. Matassoni, "The second 'CHiME' speech separation and recognition challenge: Datasets, tasks and baselines," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 126–130.

[11] R. Ochshorn and M. Hawkins, "Gentle: A forced aligner," 2016, https://github.com/lowerquality/gentle.

[12] "Method for the subjective assessment of intermediate quality level of audio systems," International Telecommunication Union Radiocommunication Standardization Sector (ITU-R), Tech. Rep. BS.1534-3, 2015.

[13] A. R. Syed, V. A. Trinh, and M. I. Mandel, "Concatenative resynthesis with improved training signals for speech enhancement," in *Proceedings of Interspeech*, 2018, to appear.