Abstract

These are specifications, guidelines and recommendations for the design and implementation of single source audio separation. This is a work in progress that will evolve as the system is developed.

SSAS Model Specs

Recommendations

GAN Generator–Used Separator

# LSTM Denoising Autoencoder feature-mapping network (DAE)

If it is necessary to use this separator I recommend using LSTM Denoising Autoencoder mask-learning network which has proven better performance than feature-mapping networks.

#### 

## Assumptions and Constraints

* This separator requires the assumption that the number of sources are fixed.

## Motivation

### Pros

* RNN based autoencoders can deal with sequences while regular AE can’t.
* RNN share their parameters/weights across all timesteps because what they are trying to learn is time-invariant. A speech signal can show-up in any moment of time.
* RNN are designed for remembering long-time dependencies. Can be leveraged to incrementally update the learnt models for both noise and speech sources over time and learn context.
* LSTM uses potentially longer context. RNN hard to learn the parameters. Weights learned with back propagation through time (BPTT). BPTT gradients get too small (or too large) as we backpropagate from *t* to *t-T* where *T* is large. Effectively, Network forgets previous “events” due to shrinkage of earlier hidden node activations as time progresses. To preserve the earlier (*t-T*) hidden node activations and to be able to use them in predictions at time *t* we use Long short-term memory (LSTM) RNNs.
* Mask-learning network: Since the mask is bounded (e.g. the mask usually has value in [0; 1]), the mask learning network has the fixed dynamic range. Therefore, it is easier for the mask learning network to generalize different noises, conditions. Moreover, during the separation since the mixture is re-introduced to the computation, the network only needs to filter out the noisy part. Compared with the feature mapping network, where the system need to both remove the noise and remember the clean reference, the learning task for mask based system is easier, and thus usually leads to better performance.

### Cons

* It fails to handle the permutation problem which occurs when separating more than two sources and it since the order of targets of the sources is irrelevant.
* It fails to handle the dimension mismatch problem which requires the number of sources that is used in the training to be the same number of sources in testing.
* Feature-mapping network: it suffers from the unbounded dynamic range. Since the network directly output the clean spectrogram, which is unbounded, the output dynamic range has to be large enough to cover all possible volumes. Such procedure will largely increase the redundancy for the learning task. For example, for the same utterance with different amplitude, the feature mapping network need to generate completely different result, which will make the network more difficult to converge. Also, prediction of the spectra may also yield over-smoothing effects in general due to the regression-to-the-mean effect (regardless of predicting log-spectra or not)

### The Recommendation

I think this separator fails to satisfy the requirements for this model, i.e., the number of sources can be more than 2 and it is variable. Also, the model can be trained on n sources while in testing, the model can be fed a mixture of m sources and it can separate them successfully.

However, if LSTM, mask learning network is used, I recommend the following:

* The use of phase-sensitive loss function
* Use ASR alignment
* Use BLSTM (already used)

## Dataset

Recommended one of the CHiME speech separation and recognition challenge I recommend the 2nd release of the challenge to compare it to other benchmarks.

### Common Input Format

All references seem to agree that using features similar to log-magnitude-spectra is a good type of input to the network. However, It was found that using log-mel-filterbank features with 100 Mel filters gave the best result.

In ASR typically less filters are used, but in enhancement, it looks like a larger number like 100 is necessary.

For DNN: concatenate features from neighboring frames for contextual information (splicing, super-frames, sliding window). For RNN: use single-frame features, it handles context directly.

The input features are the 100 dimension log mel-filterbank,

#### The Recommendation

Log-mel-filterbank features with 100 Mel filters, use single-frame features. No context window is used for the BLSTM network. All the input feature is normalized with zero mean and unit variance..

### Common Loss

**Feature-mapping network.** best performance using Euclidean distance (least-square)

**Mask-learning Network**, best performance for:

* Mask approximation (MA): least-squares loss.
* Binary mask: binary-cross entropy loss.
* Magnitude spectrum approximation (MSA) loss. MSA is better than MA as its final target is directly related with the source signal. Recommended in case of LSTM Denoising Autoencoder mask-learning network.
* Use phase-sensitive loss function

### Standalone Training

* Layer-by-layer supervised pre-training (aka. Microsoft style)
* Whenever possible, initialize from an earlier trained network
* Initially train with mask approximation in Mel-domain, then switch to signal approximation in spectral domain
* Stochastic gradient with mini-batch size of 50 utterances
* Learning rate (per whole training data) 1e-6
* Momentum with momentum weight 0.9
* Sequence shuffling
* Normalize input data to mean 0 and variance 1
* Add Gaussian noise to input with stdev 0.1 for robustness
* Validation using monitoring of development set loss
* Wait 20 epochs before no more improvement on validation loss to stop training

GAN Generator–Alternative Separators

# Alternative Separator - Deep Clustering based separator (DC)

## Motivation

### Pros

* DC can solve both permutation and output dimension problem to produce the state of the art separation performance.

### Cons

* The main drawback of DC is its inefficiency to perform end-to-end mapping, because the objective function is the affinity between the sources in the embedded space and not the separated signals themselves. Minimizing the separation error is done with an unfolding clustering system and a second network, which is trained iteratively and stage by stage to ensure convergence.

# Alternative Separator - Permutation Invariant Training (PIT)

## Motivation

### Pros

* PIT algorithm solves the permutation problem by pooling over all possible permutations for N mixing sources (N! permutations), and use the permutation with lowest error to update the network.
* PIT was shown to have comparable performance as DC.

### Cons

* PIT approach suffers the output dimension mismatch problem because it assumes a fixed number of sources.
* PIT also suffers from its computation efficiency, where the prediction window has to be much shorter than context window due to the inconsistency of the permutation both across and within sample segments.

GAN Generator–Recommended Separator

# Deep Attractor based separator (DANet)

## Motivation

### Pros

* DANet solves the permutation problem and does not suffer from the output dimensions mismatch problem.
* DANet can potentially be extended to arbitrary number of sources without the permutation problem.
* the mask learning enables a very efficient end-to-end training scheme and highly reduces the computation complexity compared with DC and PIT
* DANet removes the stepwise pre-training required in DC method to enable end-to-end training.
* Another big advantage of DANet arises from the flexibility in source dependent training, where the source-dependent knowledge could be easily incorporated by the attractor (e.g. speaker identity).

### Cons

* PIT

## Assumptions and Constraints

* This separator does not assume anything about the number of sources during training or test times.
* No special constraint on vocabulary and grammar
* The separator performs a speaker-independent separation

## Notes on the workings of the method

### Estimation of attractor points

#### Training-time

Attractor points can be estimated using various methods including the average. One possibility is to use weighted average. Since the attractors represents the source center of gravity, we can include only the embeddings of the most salient T-F bins, which leads to more robust estimation. We investigate this strategy by using an amplitude

threshold in the estimation of the attractor.

Alternatively, a neural network model may also be used to pick the representative embedding for each source, an idea which shares similarities with encoder-decoder attention networks

#### Test-time

During test time, because the true assignment Y is unknown, we incorporate two strategies to form the attractor points. The first find the centers using post K-means algorithm. The second method is based on the observation that the location of the attractors in the embedding space is relatively stable.

## DANet Dataset

We should be using the dataset used in DANet which is basically introduced in the Deep clustering sound source separation paper:

It contains a 30 h training set and a 10 h validation set generated by randomly selecting utterances from

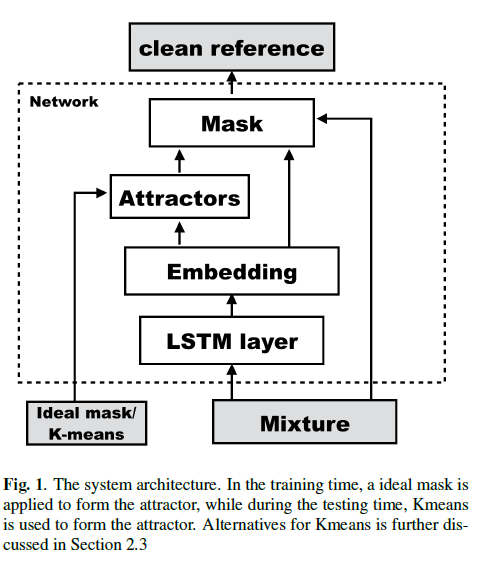
different speakers in the Wall Street Journal (WSJ0) training set si\_tr\_s, and mixing them at various signal-to-noise ratios (SNR) randomly chosen between 0 dB and 10 dB. 5 h evaluation set is generated similarly as above, using utterances from 16 unseen speakers from si\_ dt\_05 and si\_et\_05 in WSJ0 dataset. Additionally, the authors constructed a three-speaker mixture dataset for three speaker separation evaluation from same WSJ set, which has 30h training, 10 hours validation and 5 hours testing data, with mixing SNR at -5 5 dB. They ensure that in each mixture, there exist both female and male speakers. All data are resampled to 8 kHz to reduce computational and memory costs.

### DANet Input Format

The log spectral magnitude is served as input feature, computed using short-time Fourier transform (STFT) with 32 ms window length, 8 ms hop size, and the square root of hanning window. We split the input features into non-overlapping chunks of 100-frame length as the input to the network.

## DANet Architecture

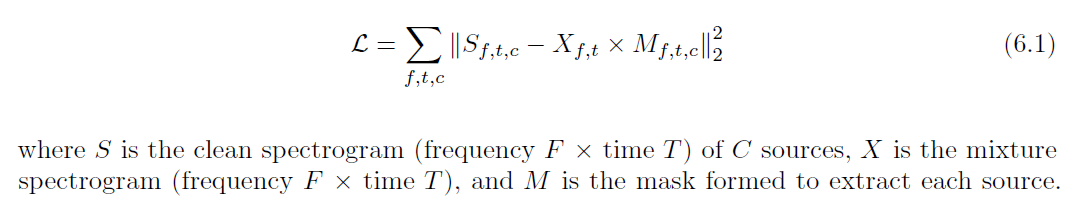
* The network contains 4 BLSTM layers with 600 hidden units in each layer.
* The embedding dimension is set to 20, resulting in a fully-connected feed-forward layer of 2580 hidden units (20 x 129) after the BLSTM layers.



## DANet Standalone Training

RMSprop algorithm is used for training with an exponential learning rate decaying strategy, where the learning rate starts at 10-4 and ends at 3 x 10-6.

The total number of epochs was set to be 150, and we will use the cost function in Equation 6.1 on the validation set for early stopping.



The criteria for early stopping is no decrease in the loss function on validation set for 10 epochs. Moreover, by applying curriculum training strategy, i.e. continue training the network with 400-frame length input, DANet achieves the best overall performance.

### Metrics

signal-to-distortion ratio (SDR, which is defined as scale-invariant SNR here), signal-to-artifacts ratio (SAR), and signal-to-interference ratio (SIR).

### Reported results

#### 

GAN Generator–Speech Recognizer

More details on the speech recognizer module later.

Concern: without supervised training, Is speech recognizer going to be effective with solely GAN training?

Optional: we can alternatively consider giving speech recognizer standalone supervised training objective.

GAN Generator–Discriminator

Discriminator is a classifier, and its inputs are either real samples, coming from the dataset that G is imitating, or fake samples, made up by G.

The adversarial characteristic comes from the fact that D must classify the samples coming from X as real, whereas the samples coming from G, X^, must be classified as fake. This leads to G trying to fool D, and the way to do so is that G adapts

its parameters such that D classifies G’s output as real. During back-propagation, D gets better at finding realistic features in its input and, in turn, G corrects its parameters to move towards the real data manifold described by the training data

## Discriminator Architecture

We will use two BLSTM networks, processing text and signal respectively. Effectively, we obtain two embedding vectors. Then we feed concatenated embedding vector to a fully connected layer, which output a 3 dimensional categorical probability distribution. Belonging to classes {true\_signal, true\_noise, fake\_signal}

### Signal Subnetwork

The signal BLSTM sub-network is similar to DANet.

* It contains 4 BLSTM layers with 600 hidden units in each layer.
* The embedding dimension is set to 20, resulting in a fully-connected feed-forward layer of 2580 hidden units (20 x 129) after the BLSTM layers.

### Text Subnetwork

I am recommending a minor change to the architecture of this sub-network. The 2 BLSTM layers will have 600 and 300 hidden units respectively. In addition to the use of a dropout layer after every layer.

We should try other variants such as the use of a conv layer before the BLSTM or the use of max pooling layer after the first BLSTM and removing the second in case of bad performance for this sub-network.

#### Alternative Architecture

* Embedding layer, length 32
* 1D Convolutional Layer with 32 filters and 3x3 kernel size, and an activation of Relu and 1D Max Pooling
* It contains 1 BLSTM layers with 300 hidden units.

### Fully-connected Layer for classification

The FC is going to output 3 dimensional logits that are going to be fed into a SoftMax activation function which produces estimated categorical probability distribution for 3-way classification.

GAN

# Architecture

* We will use Relu and Leaky Relu to avoid sparse gradients.
* We should use Dropouts (%50) in several layers of the Generator in both train and test phases

## [Wasserstein GAN](https://www.google.se/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiw4_rmmqbWAhXSmLQKHeRtC74QFggoMAA&url=https%3A%2F%2Farxiv.org%2Fabs%2F1701.07875&usg=AFQjCNEzsMTnaaks-q6t04LPTMfIsS1pWA)

We will work with a WGANs, that uses a different loss/cost functions that perform well in training better than regular GANs. The generator is basically the separator component and the speech recognizer while the discriminator is composed of two subnetworks, a discriminator subnetwork for the speech signal and another one for the text. The output of both are then fused using a fully-connected layer.

# Training

The generator initially generates some random noise (because it’s weights will be random). After training our discriminator to discriminate this random noise and real images, we’ll connect our generator to our discriminator and backprop only through the generator with the constraint that the discriminator output should be 1 (i.e, the discriminator should classify the output of the generator as real images).

We’ll again train our discriminator to now tell apart the new fake images from our generator and the real ones from our database. This is followed by training the generated to generate better fake looking images.

We’ll continue this process until the generator becomes so good at generating fake images that the discriminator is no longer able to tell real images from fake ones.

At the end, we’ll be left with a generator which can produce real looking fake images given a random set of numbers as its input.

### Choice of activation

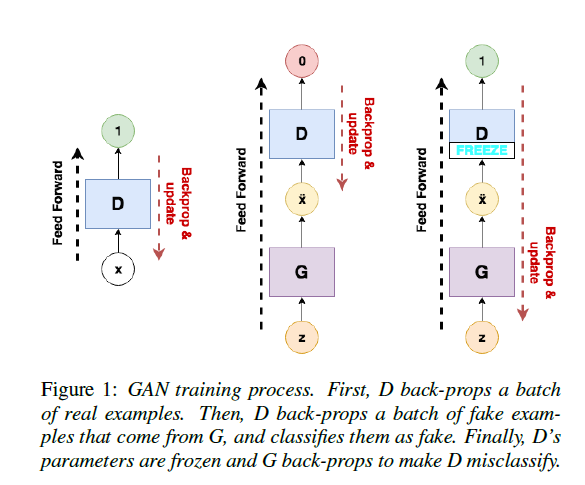
relu or lrelu

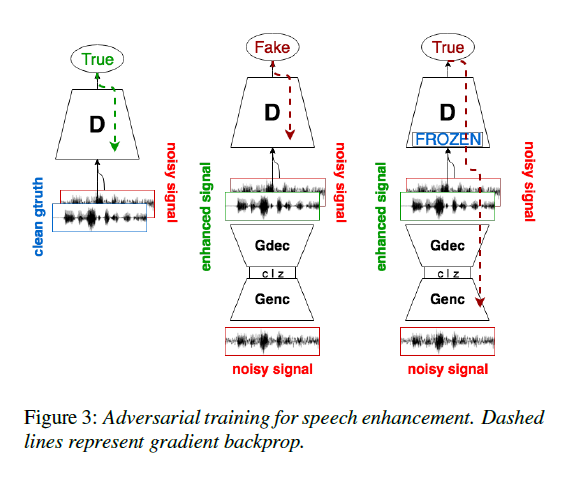
### Choice of nonlinear function

sigmoid, tanh (tanh is preferable specially for the last layers in both the generator and the discriminator

### Dropout

Aggressive and widespread use of dropout.





## Tricks to stabilize training

* Adding gaussian noise to every layer of generator.
* Use SGD for discriminator and ADAM for generator
* Use stability tricks
  + Maintain a replay buffer of past generations and occasionally show them
  + Maintain checkpoints from the past of G and D and occasionally swap them out for a few iterations
* Label Smoothing
  + Instead of using labels: Real=1 and Fake=0, we use a random number between 0.7 and 1.2 for real samples, and if it is a fake sample, we use 0.0 and 0.3.
  + We will occasionally flip the labels when training the discriminator

### Hyperparameters tuning

GANs are very sensitive to hyperparameters. A lot of experimentation goes into finding the best hyperparameters such that the generator and discriminator don’t overpower each other.

We will train the model for 100 epochs with RMSprop and a learning rate of 2x10-4, using an effective batch size of 500.

### Optimization

We want to update the generator and discriminator variables separately. Note that when minimizing the discriminator loss, we want optimizer to only be updating the discriminator variables and similar for generator.

# Dataset

We will use the same dataset we used for the standalone training for the Deep Attractor Network.