Unsupervised Probabilistic Learning with Latent Variables

Lecture 3: Dynamical variational autoencoders, application to speech enhancement.

MLSS Africa 2023, Cape Town **Slides:** https://xavirema.eu/MLSS2023/

Xavier Alameda-Pineda

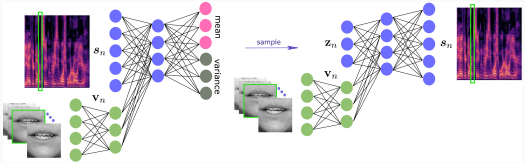
RobotLearn team, Inria at University Grenoble-Alpes,
Jean-Kuntzman CNRS Laboratory, Multidisciplinary Institute of Artificial Intelligence

Inria

UGA Université Grenoble Alpes

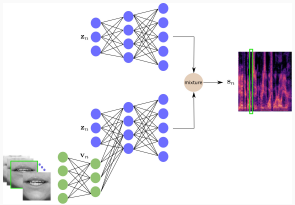
MIAI Grenoble Alpes

Summary of the three VAE-based models for AV-SE



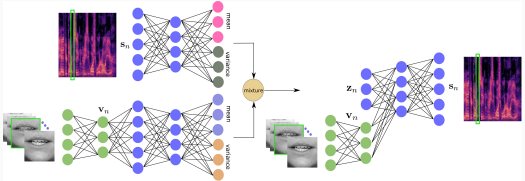
Conditional VAE:

- Training VAE via SGD.
- Systematic AV fusion.
- Learning noise parameters via MCEM.



VAE-MM:

- Training two VAEs via SGD.
- Mixing AV fusion - two speech models.
- Learning noise parameters via VEM.



MIN-VAE:

- Training all 3 networks via VEM+SGD.
- Mixing AV fusion - single speech model.
- Learning noise parameters via VEM.

Motivation for DVAE

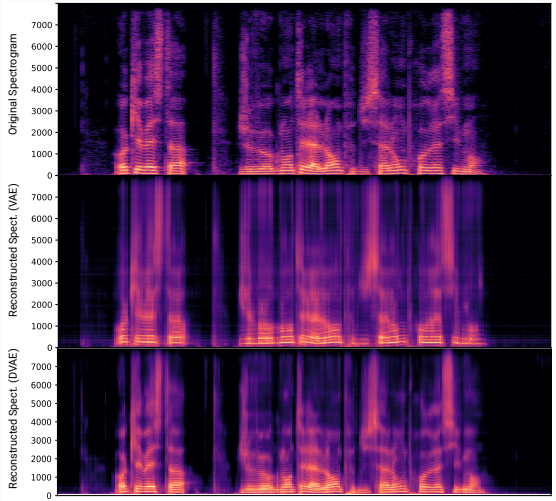
All these models process speech data *independently* per each frame,
that is regardless of the frame order.

But speech is sequential, and order matters. What can we do?
(we will forget about visual data in this lecture)

Are there VAE-like models able to deal with sequential data?
Yes, we will call them **dynamical variational autoencoders (DVAE)**.

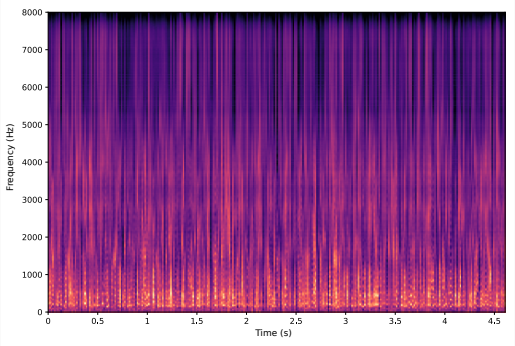
We'll also discuss their use in unsupervised speech enhancement?

Reconstruction?

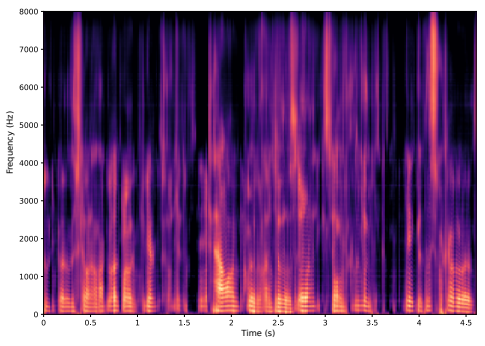


Original (top) | Reconstruction VAE (middle) | Reconstruction DVAE (bottom)

Generation?



Generation VAE (left)



Generation DVAE (right)

Lecture's Outline

1. Formalising Dynamical Variational Autoencoders
2. Inference network: conditional independence and Markov blankets
3. DVAE: general case, implementation and learning
4. DVAEs for unsupervised speech enhancement

Formalising Dynamical Variational Autoencoders

Probabilistic Sequential Modeling

We would like to model sequences of observations and latent variables:
Observed sequence: $\boldsymbol{x}_{1:T} = \{\boldsymbol{x}_t \in \mathbb{R}^F\}_{t=1}^T$
Latent sequence: $\boldsymbol{z}_{1:T} = \{\boldsymbol{z}_t \in \mathbb{R}^D\}_{t=1}^T$

Generative sequential modeling consists in defining the joint distribution with temporal dependencies, rather than the frame-wise joint distribution (as a vanilla VAE does):

$$p_{\boldsymbol{\theta}}^{\mathrm{DVAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) \neq p_{\boldsymbol{\theta}}^{\mathrm{VAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{z}_t).$$

Chain rule

Using the chain rule we can write the joint distribution as a product of conditionals:

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_1)p(\boldsymbol{x}_1|\boldsymbol{z}_1)\prod_{t=2}^{T} p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1})p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t})$$

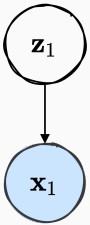
Generative process:

Chain rule

Using the chain rule we can write the joint distribution as a product of conditionals:

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_1)p(\boldsymbol{x}_1|\boldsymbol{z}_1)\prod_{t=2}^{T} p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1})p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t})$$

Generative process:

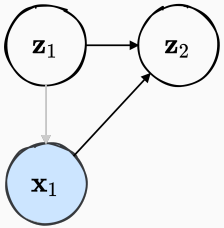


Chain rule

Using the chain rule we can write the joint distribution as a product of conditionals:

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_1)p(\boldsymbol{x}_1|\boldsymbol{z}_1) \prod_{t=2}^{T} p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1})p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t})$$

Generative process:

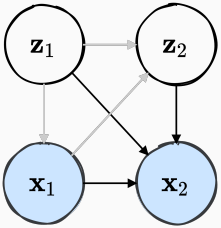


Chain rule

Using the chain rule we can write the joint distribution as a product of conditionals:

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_1)p(\boldsymbol{x}_1|\boldsymbol{z}_1)\prod_{t=2}^{T} p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1})p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t})$$

Generative process:

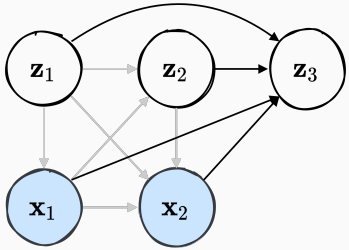


Chain rule

Using the chain rule we can write the joint distribution as a product of conditionals:

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_1)p(\boldsymbol{x}_1|\boldsymbol{z}_1) \prod_{t=2}^{T} p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1})p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t})$$

Generative process:

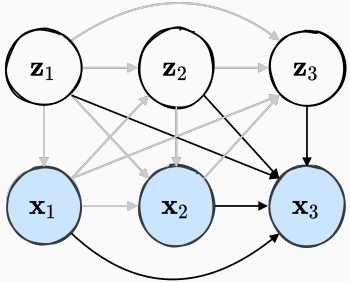


Chain rule

Using the chain rule we can write the joint distribution as a product of conditionals:

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_1)p(\boldsymbol{x}_1|\boldsymbol{z}_1)\prod_{t=2}^{T} p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1})p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t})$$

Generative process:

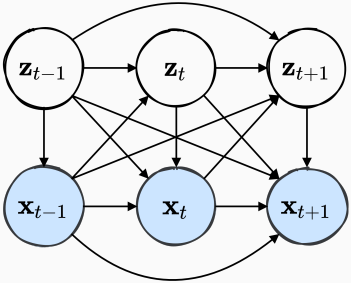


Chain rule

Using the chain rule we can write the joint distribution as a product of conditionals:

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_1)p(\boldsymbol{x}_1|\boldsymbol{z}_1) \prod_{t=2}^{T} p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1})p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t})$$

Generative process:



The distribution $p(\boldsymbol{z}_{t+1}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{1:t})$ is not a *prior* distribution anymore, ...

...it is parametric and it might be auto-regressive (AR).

$p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{1:t+1}, \boldsymbol{x}_{1:t})$ might be AR as well.

Very well-known particular case: Kalman

A particular case of the above paradigm are linear dynamical systems (a.k.a. Kalman filter). Assuming the following conditional independence hypothesis:

$$p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}) = p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) \qquad p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t}) = p(\boldsymbol{x}_t|\boldsymbol{z}_t),$$

Very well-known particular case: Kalman

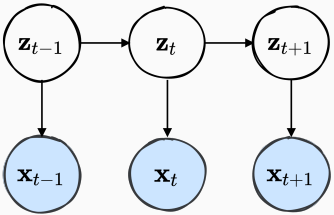
A particular case of the above paradigm are linear dynamical systems (a.k.a. Kalman filter). Assuming the following conditional independence hypothesis:

$$p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}) = p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) \qquad p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t}) = p(\boldsymbol{x}_t|\boldsymbol{z}_t),$$

the joint distribution simplifies as:

$$p(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = p(\boldsymbol{z}_1)p(\boldsymbol{x}_1|\boldsymbol{z}_1) \prod_{t=2}^{T} p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})p(\boldsymbol{x}_t|\boldsymbol{z}_t).$$

This is the family of state-space models (SSMs) introduced by Kalman in 1960 (linear-Gaussian SSMs with continuous state and hidden Markov models with discrete state).



(Deep?) Kalman/Markov model

The standard Kalman (HMM) models, the dependency is linear (left),
while in the deep counterparts it is non-linear (right)!

Kalman filter:

$$p_{\boldsymbol{\theta_z}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \mathcal{N}\Big(\boldsymbol{z}_t; \boldsymbol{A}_t\boldsymbol{z}_{t-1}, \boldsymbol{Q}_t\Big)$$

$$p_{\boldsymbol{\theta_x}}(\boldsymbol{x}_t|\boldsymbol{z}_t) = \mathcal{N}\Big(\boldsymbol{x}_t; \boldsymbol{B}_t\boldsymbol{z}_t, \boldsymbol{R}_t\Big),$$

Deep Kalman filter:

$$p_{\boldsymbol{\theta_z}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \mathcal{N}\Big(\boldsymbol{z}_t; \boldsymbol{\mu}_{\boldsymbol{\theta_z}}(\boldsymbol{z}_{t-1}), \boldsymbol{\Sigma}_{\boldsymbol{\theta_z}}(\boldsymbol{z}_{t-1})\Big)$$

$$p_{\boldsymbol{\theta_x}}(\boldsymbol{x}_t|\boldsymbol{z}_t) = \mathcal{N}\Big(\boldsymbol{x}_t; \boldsymbol{\mu}_{\boldsymbol{\theta_x}}(\boldsymbol{z}_t), \boldsymbol{\Sigma}_{\boldsymbol{\theta_x}}(\boldsymbol{z}_t)\Big),$$

The non-linearities can be implemented with feed-forward neural networks, as in VAEs.

Can we learn it?

Recall that, for learning VAE, we needed an auxiliary inference (encoder) network, that approximates the posterior:

This is our next objective!

Inference network: conditional independence and Markov blankets

Sequential Inference Network

As in the VAE, we need to approximate the posterior distribution:

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$$

Sequential Inference Network

As in the VAE, we need to approximate the posterior distribution:

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$$

We can always use the Bayes theorem to write:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T})$$

Can we simplify each of the terms further? It depends on generative model.

Sequential Inference Network

As in the VAE, we need to approximate the posterior distribution:

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$$

We can always use the Bayes theorem to write:

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T})$$

Can we simplify each of the terms further? It depends on generative model.
For instance, for (deep) kalman filter we have the following result:

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t:T})$$

But how to obtain it?

Conditional Independence

You might very well know what "independence" means:

$$\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y} \Leftrightarrow p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x})p(\boldsymbol{y}) \Leftrightarrow p(\boldsymbol{x}|\boldsymbol{y}) = p(\boldsymbol{x}).$$

Conditional Independence

You might very well know what "independence" means:

$$\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y} \Leftrightarrow p(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x})p(\boldsymbol{y}) \Leftrightarrow p(\boldsymbol{x}|\boldsymbol{y}) = p(\boldsymbol{x}).$$

Conditional independence is defined in a similar way, but conditionned to $\boldsymbol{z}$:

$$\boldsymbol{x} \perp\!\!\!\perp \boldsymbol{y} \,|\, \boldsymbol{z} \Leftrightarrow p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{y}|\boldsymbol{z}) \Leftrightarrow p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{z}).$$

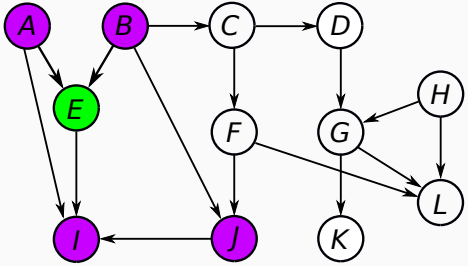
This means that $\boldsymbol{z}$ is *breaking* the link between $\boldsymbol{x}$ and $\boldsymbol{y}$: if we know $\boldsymbol{z}$, $\boldsymbol{y}$ does not bring any extra information to describe $\boldsymbol{x}$.

In models with lots of variables, we cannot test all possible subsets of variables $\boldsymbol{z}$, we need an *automatic* way to check that.

Markov Blankets

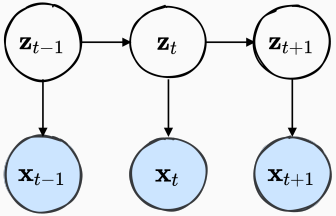
(Disclaimer: we are taking a shorcut. Other key concepts are path blocking and D-separation.)

The **Markov Blanket** of $\boldsymbol{x}$, $\mathcal{B}(\boldsymbol{x})$ is the set of nodes isolating $\boldsymbol{x}$ from the rest of the graph.



In the case above: $\mathcal{B}(\boldsymbol{E}) = \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{I}, \boldsymbol{J}\}$, that is the set of **parents**, **children** and **co-parents** (other parents of its children).

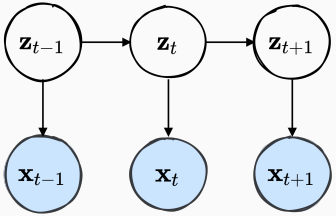
Let's take a look back at DKF



We relise that $\boldsymbol{z}_{t-1}$ is isolating $\{\boldsymbol{z}_t, \boldsymbol{x}_{t:T}\}$ from the past:
($\boldsymbol{z}_{t-1}$ is the Markov blanket of $\{\boldsymbol{z}_t, \boldsymbol{x}_{t:T}\}$)

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t:T})$$

Let's take a look back at DKF



We relise that $\boldsymbol{z}_{t-1}$ is isolating $\{\boldsymbol{z}_t, \boldsymbol{x}_{t:T}\}$ from the past:
($\boldsymbol{z}_{t-1}$ is the Markov blanket of $\{\boldsymbol{z}_t, \boldsymbol{x}_{t:T}\}$)

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t:T})$$

It seems natural to impose the same dependencies in the inference network:

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t:T}) \approx \prod_{t=1}^{T} q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_{t:T})$$

Note: It is OK to simplify even further, but you need to know what you are doing.

DVAE: general case, implementation and learning

The general (causal) case

General umbrella for all (causal) DVAEs:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}) p_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t}),$$

where

$$p_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}) = \mathcal{N}\Big(\boldsymbol{z}_t; \boldsymbol{\mu}_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\ldots), \mathsf{diag}\{\boldsymbol{v}_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\ldots)\}\Big),$$

$$p_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t}) = \mathcal{N}\Big(\boldsymbol{x}_t; \boldsymbol{\mu}_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\ldots), \mathsf{diag}\{\boldsymbol{v}_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\ldots)\}\Big),$$

and $\{\boldsymbol{\mu}_{\boldsymbol{\theta}_{\boldsymbol{z}}}, \boldsymbol{v}_{\boldsymbol{\theta}_{\boldsymbol{z}}}\}$, and $\{\boldsymbol{\mu}_{\boldsymbol{\theta}_{\boldsymbol{x}}}, \boldsymbol{v}_{\boldsymbol{\theta}_{\boldsymbol{x}}}\}$ are **non-linear functions** of the conditionning variables.

Which kind of networks can be use to construct these functions?

The general (causal) case

General umbrella for all (causal) DVAEs:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}) p_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t}),$$

where

$$p_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}) = \mathcal{N}\Big(\boldsymbol{z}_t; \boldsymbol{\mu}_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\ldots), \mathsf{diag}\{\boldsymbol{v}_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\ldots)\}\Big),$$

$$p_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t}) = \mathcal{N}\Big(\boldsymbol{x}_t; \boldsymbol{\mu}_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\ldots), \mathsf{diag}\{\boldsymbol{v}_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\ldots)\}\Big),$$

and $\{\boldsymbol{\mu}_{\boldsymbol{\theta}_{\boldsymbol{z}}}, \boldsymbol{v}_{\boldsymbol{\theta}_{\boldsymbol{z}}}\}$, and $\{\boldsymbol{\mu}_{\boldsymbol{\theta}_{\boldsymbol{x}}}, \boldsymbol{v}_{\boldsymbol{\theta}_{\boldsymbol{x}}}\}$ are **non-linear functions** of the conditionning variables.

Which kind of networks can be use to construct these functions? RNN, Transformers, ...

Example with vanilla RNN

Recall the DVAE generative model:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}) p_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t}).$$

The conditional distributions are parametrized, for instance:

- $\boldsymbol{h}_t = \sigma(\boldsymbol{W}_{xh}\boldsymbol{x}_{t-1} + \boldsymbol{W}_{zh}\boldsymbol{z}_{t-1} + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$,
- $p_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t-1}) = \mathcal{N}\Big(\boldsymbol{z}_t; \boldsymbol{\mu}_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\boldsymbol{h}_t), \mathsf{diag}\{\boldsymbol{v}_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\boldsymbol{h}_t)\}\Big)$,
- $p_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{z}_{1:t}) = \mathcal{N}\Big(\boldsymbol{x}_t; \boldsymbol{\mu}_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{z}_t, \boldsymbol{h}_t), \mathsf{diag}\{\boldsymbol{v}_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{z}_t, \boldsymbol{h}_t)\}\Big)$.

In this example, the same RNN is used for $\boldsymbol{x}$ and for $\boldsymbol{z}$. This is an arbitrary choice.

Inference for the general case

In the general case, we cannot simplify the dependencies of the inference model:

$$q_{\phi}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} q_{\phi}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T}),$$

where typically we have ($\tilde{\boldsymbol{\mu}}_{\phi}$, $\tilde{\boldsymbol{v}}_{\phi}$ are non-linear functions of $\boldsymbol{z}_{1:t-1}$, $\boldsymbol{x}_{1:T}$):

$$q_{\phi}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T}) = \mathcal{N}\Big(\boldsymbol{z}_t; \tilde{\boldsymbol{\mu}}_{\phi}(\ldots), \mathrm{diag}\{\tilde{\boldsymbol{v}}_{\phi}(\ldots)\}\Big).$$

Inference for the general case

In the general case, we cannot simplify the dependencies of the inference model:

$$q_{\phi}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} q_{\phi}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T}),$$

where typically we have ($\tilde{\boldsymbol{\mu}}_{\phi}$, $\tilde{\boldsymbol{v}}_{\phi}$ are non-linear functions of $\boldsymbol{z}_{1:t-1}$, $\boldsymbol{x}_{1:T}$):

$$q_{\phi}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T}) = \mathcal{N}\Big(\boldsymbol{z}_t; \tilde{\boldsymbol{\mu}}_{\phi}(\ldots), \mathrm{diag}\{\tilde{\boldsymbol{v}}_{\phi}(\ldots)\}\Big).$$

One possible parametrization of the conditional posterior of $\boldsymbol{z}_t$ is given as follows:

$$q_{\phi}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T}) = \mathcal{N}\Big(\boldsymbol{z}_t; \tilde{\boldsymbol{\mu}}_{\phi}\Big(\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t\Big), \mathrm{diag}\Big\{\tilde{\boldsymbol{v}}_{\phi}\Big(\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t\Big)\Big\}\Big),$$

where:

- $\overrightarrow{\boldsymbol{h}}_t = \sigma(\overrightarrow{\boldsymbol{W}}_{xh}\boldsymbol{x}_{t-1} + \overrightarrow{\boldsymbol{W}}_{zh}\boldsymbol{z}_{t-1} + \overrightarrow{\boldsymbol{W}}_{hh}\overrightarrow{\boldsymbol{h}}_{t-1} + \overrightarrow{\boldsymbol{b}}_h)$ – encodes causal dependencies.
- $\overleftarrow{\boldsymbol{h}}_t = \sigma(\overleftarrow{\boldsymbol{W}}_{xh}\boldsymbol{x}_t + \overleftarrow{\boldsymbol{W}}_{hh}\overleftarrow{\boldsymbol{h}}_{t+1} + \overleftarrow{\boldsymbol{b}}_h)$ – encodes non-causal dependencies.

Learning: ELBO

The objective function is slightly different compared with standard VAEs:

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}_{1:T};\boldsymbol{\phi},\boldsymbol{\theta}) = &\sum_{t=1}^{T}\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:t}|\boldsymbol{x}_{1:T})}[\ln p_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1},\boldsymbol{z}_{1:t})] \\ &-\sum_{t=1}^{T}\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:t-1}|\boldsymbol{x}_{1:T})}\left[D_{\mathsf{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1},\boldsymbol{x}_{1:T})\,\Big\|\,p_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1},\boldsymbol{z}_{1:t-1})\Big)\right]\end{aligned}$$

Learning: ELBO

The objective function is slightly different compared with standard VAEs:

$$\mathcal{L}(\boldsymbol{x}_{1:T};\boldsymbol{\phi},\boldsymbol{\theta}) = \sum_{t=1}^{T} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:t}|\boldsymbol{x}_{1:T})}[\ln p_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1},\boldsymbol{z}_{1:t})] - \sum_{t=1}^{T} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_{1:t-1}|\boldsymbol{x}_{1:T})} \left[D_{\mathsf{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1},\boldsymbol{x}_{1:T}) \,\Big\|\, p_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1},\boldsymbol{z}_{1:t-1}) \Big) \right]$$

- The reconstruction and regularisation terms are evaluated at every frame t.
- Because of the model, the KL term depends on previous latent variables $\Rightarrow$ sampling.
- The sampling occurs sequentially and cannot be paralelized!

DVAE Review & Code

Six (of the many) different models in the literature belong to the DVAE family:

STORN	J. Bayer and C. Osendorfer, Learning Stochastic Recurrent Networks, arXiv, 2014
VRNN	J. Chung et al., A recurrent latent variable model for sequential data, NeurIPS, 2015
SRNN*	M. Fraccaro et al., Sequential neural models with stochastic layers, NeurIPS, 2016
DMM*	R. Krishnan et al., Structured Inference Networks for Nonlinear State Space Models, AAAI, 2017
DSAE	Y. Li and S Mandt, Disentangled sequential autoencoder, ICML, 2018.
RVAE*	S. Leglaive et al., A recurrent variational autoencoder for speech enhancement, ICASSP 2020

Different conditional independence assumptions.

* the encoder is compliant with the true posterior.

DVAE Review & Code

Six (of the many) different models in the literature belong to the DVAE family:

STORN	J. Bayer and C. Osendorfer, Learning Stochastic Recurrent Networks, arXiv, 2014
VRNN	J. Chung et al., A recurrent latent variable model for sequential data, NeurIPS, 2015
SRNN*	M. Fraccaro et al., Sequential neural models with stochastic layers, NeurIPS, 2016
DMM*	R. Krishnan et al., Structured Inference Networks for Nonlinear State Space Models, AAAI, 2017
DSAE	Y. Li and S Mandt, Disentangled sequential autoencoder, ICML, 2018.
RVAE*	S. Leglaive et al., A recurrent variational autoencoder for speech enhancement, ICASSP 2020

Different conditional independence assumptions.

* the encoder is compliant with the true posterior.

Check "Dynamical Variational Autoencoders: A Comprehensive Review", FnT ML:

- review and discuss these models (and more) with unified notations,
- compare their performance for analysis/resynthesis of speech signals,
- provide an implementation: `https://github.com/XiaoyuBIE1994/DVAE-speech`.

Results on speech analysis-resynthesis (reconstruction)

RESULTS OF THE SPEECH ANALYSIS-RESYNTHESIS EXPERIMENT, AVERAGED OVER THE TEST SUBSET OF WSJ0 AND VOICEBANK.

Models	Dataset	SI-SDR (dB)	PESQ MOS	PESQ WB	PESQ NB	ESTOI
VAE	WSJ0	8.0	3.33	2.95	3.31	0.89
DKF	WSJ0	9.0	3.55	3.39	3.61	0.91
RVAE	WSJ0	9.8	3.65	3.57	3.75	0.92
SRNN	WSJ0	8.2	3.48	3.24	3.52	0.90
VAE	VB	8.6	3.22	2.79	3.15	0.88
DKF	VB	9.4	3.35	2.96	3.34	0.90
RVAE	VB	9.6	3.41	3.00	3.42	0.90
SRNN	VB	9.1	3.39	2.99	3.39	0.89

Various metrics, the SI-SDR is the scale-invariant signal-to-noise ratio.

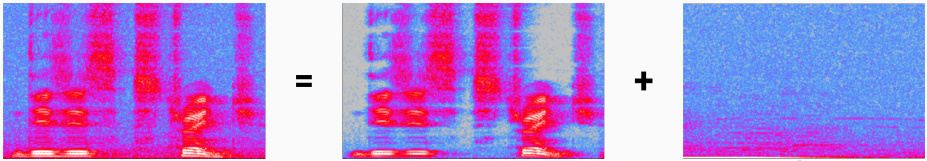
DKF/RVAE/SRNN outperform the VAE, which does not model cross-frame dependencies.

DVAEs for unsupervised speech enhancement

Unsupervised Speech Enhancement

Speech Enhancement: Remove the background noise from the observed mixture speech.

Short-time Fourier transform (STFT) is a time-frequency (matrix) representation.



Unsupervised (our work): Learn a generative audio-visual model for clean speech and combine it with an unsupervised noise model at test time

Unsupervised SE is **more flexible** since it adapts to various noises.

Unsupervised speech enhancement: overview

Train a generative speech model with clean data: $\{\boldsymbol{s}_i\}_{i=1}^{N}$.

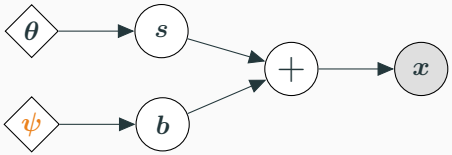
$$p_{\boldsymbol{\theta}}(\boldsymbol{s}) \approx p_{\text{data}}(\boldsymbol{s})$$

Unsupervised speech enhancement: overview

Train a generative speech model with clean data: $\{\boldsymbol{s}_i\}_{i=1}^N$.

$$p_{\boldsymbol{\theta}}(\boldsymbol{s}) \approx p_{\text{data}}(\boldsymbol{s})$$

Test: learn the noise parameters of $\boldsymbol{x}$:



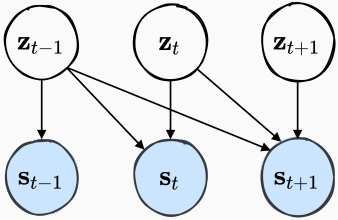
Note: at test time, $\boldsymbol{\theta}$ is frozen, and $\boldsymbol{s}$ becomes a latent variable. Let's detail a bit more RVAE!

Recurrent Variational Autoencoder (RVAE)

Causal version (C-RVAE):

$$p_{\theta_s}(\boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{z}_t) p_{\theta_s}(\boldsymbol{s}_t | \boldsymbol{z}_{1:t}),$$

where $p(\boldsymbol{z}_t) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $p_{\theta_s}(\boldsymbol{s}_t | \boldsymbol{z}_{1:t}) = \mathcal{N}_c\Big(\boldsymbol{s}_t; \mathbf{0}, \mathrm{diag}\{\boldsymbol{v}_{\theta_s}(\boldsymbol{z}_{1:t})\}\Big)$

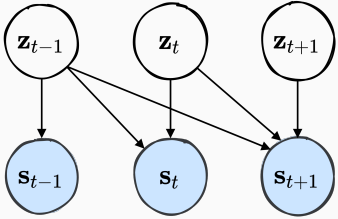


Recurrent Variational Autoencoder (RVAE)

Causal version (C-RVAE):

$$p_{\theta_s}(\boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{z}_t) p_{\theta_s}(\boldsymbol{s}_t | \boldsymbol{z}_{1:t}),$$

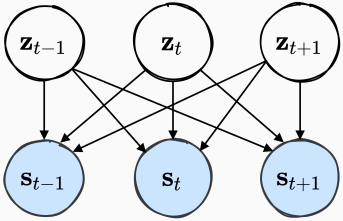
where $p(\boldsymbol{z}_t) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $p_{\theta_s}(\boldsymbol{s}_t | \boldsymbol{z}_{1:t}) = \mathcal{N}_c\Big(\boldsymbol{s}_t; \boldsymbol{0}, \text{diag}\{\boldsymbol{v}_{\theta_s}(\boldsymbol{z}_{1:t})\}\Big)$



Non-causal version (NC-RAVE)

$$p_{\theta_s}(\boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{z}_t) p_{\theta_s}(\boldsymbol{s}_t | \boldsymbol{z}_{1:T}),$$

where $p(\boldsymbol{z}_t) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $p_{\theta_s}(\boldsymbol{s}_t | \boldsymbol{z}_{1:T}) = \mathcal{N}_c\Big(\boldsymbol{s}_t; \boldsymbol{0}, \text{diag}\{\boldsymbol{v}_{\theta_s}(\boldsymbol{z}_{1:T})\}\Big)$



Noisy speech model

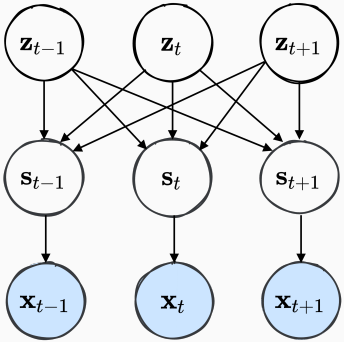
At test time, the clean speech signal is **latent**, and we consider the following generative model:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:T}, \boldsymbol{s}_{1:T}, \boldsymbol{z}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{z}_t) p_{\boldsymbol{\theta}_{\boldsymbol{s}}}(\boldsymbol{s}_t|\boldsymbol{z}_{1:T}) p_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{x}_t|\boldsymbol{s}_t),$$

where $\boldsymbol{x}_t$ is a **noisy speech** frame, whose likelihood is defined by:

$$p_{\theta_{\boldsymbol{x}}}(\boldsymbol{x}_t|\boldsymbol{s}_t) = \mathcal{N}_c\Big(\boldsymbol{x}_t; \boldsymbol{s}_t, \mathrm{diag}\big\{(\boldsymbol{W}\boldsymbol{H})_{:,t}\big\}\Big),$$

with $\theta_{\boldsymbol{x}} = \{\boldsymbol{W} \in \mathbb{R}_+^{F\times K}, \boldsymbol{H} \in \mathbb{R}_+^{K\times T}\}$.



EM Algorithm for Parameter Learning

Expectation-maximization (EM)-like algorithm:

- Approximate $p_{\boldsymbol{\theta}}(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$
- Estimate $\boldsymbol{\theta}_{\boldsymbol{x}} = \{\boldsymbol{W} \in \mathbb{R}_{+}^{F\times K}, \boldsymbol{H} \in \mathbb{R}_{+}^{K\times T}\}$

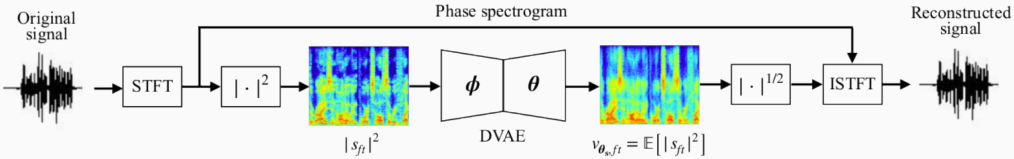
Three strategies for the intractable E-step:

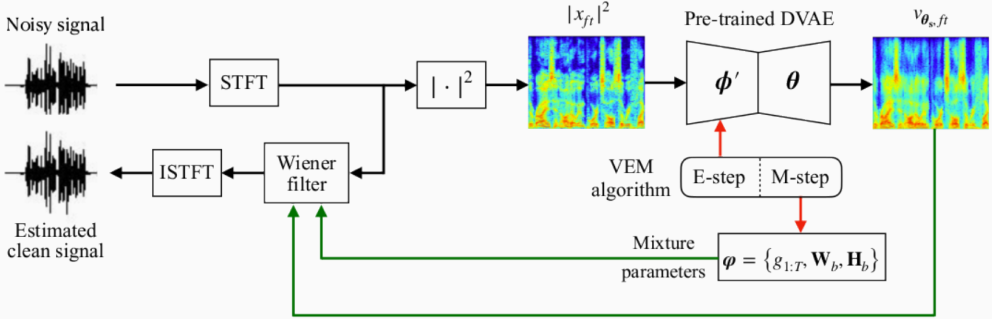
- Markov Chain Monte Carlo
- Variational inference]: $p_\theta(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) \approx q_\phi(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})$
- $p_\theta(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T}) \approx \delta(\boldsymbol{z}_{1:T} - \boldsymbol{z}_{1:T}^{\text{MAP}})$

Posterior mean estimate of the speech signal with Wiener-like filtering (element-wise ops):

$$\hat{\boldsymbol{s}}_t = \mathbb{E}_{p_\theta(\boldsymbol{s}_t|\boldsymbol{x}_{1:T})}[\boldsymbol{s}_t] = \mathbb{E}_{p_\theta(\boldsymbol{z}_{1:T}|\boldsymbol{x}_{1:T})}\left[\frac{\boldsymbol{v}_{\theta_{\boldsymbol{s}}}(\boldsymbol{z}_{1:T})}{\boldsymbol{v}_{\theta_{\boldsymbol{s}}}(\boldsymbol{z}_{1:T}) + (\boldsymbol{W}\boldsymbol{H})_{:,t}}\right]\boldsymbol{x}_t.$$

Pre-training and Parameter Learning





Results on the Wall Street Journal (WSJ) and VoiceBank (VB)

Method	Superv.	Test subset	Train subset	SI-SDR (dB)	Train subset	SI-SDR (dB)
Noisy mixture	-	WSJ0-QUT	-	−2.6	-	−2.6
VAE-VEM ICASSP'20	UA	WSJ0-QUT	WSJ0	5.0		
Proposed DKF-VEM	UA	WSJ0-QUT	WSJ0	5.1		
Proposed RVAE-VEM	UA	WSJ0-QUT	WSJ0	**5.8**		
Proposed SRNN-VEM	UA	WSJ0-QUT	WSJ0	5.2		
MetricGAN-U (full) ICASSP'22	UD	WSJ0-QUT	WSJ0-QUT	N/A		
MetricGAN-U (half) ICASSP'22	UD	WSJ0-QUT	WSJ0-QUT	N/A		
UMX* 2020	S	WSJ0-QUT	WSJ0-QUT	**5.7**		
MetricGAN+* Interspeech'21	S	WSJ0-QUT	WSJ0-QUT	3.6		
Noisy mixture	-					
VAE-VEM ICASSP'20	UA					
Proposed DKF-VEM	UA					
Proposed RVAE-VEM	UA					
Proposed SRNN-VEM	UA					
NyTT EUSIPCO'21	UD					
NyTT EUSIPCO'21	UD					
MetricGAN-U (full) ICASSP'22	UD					
MetricGAN-U (half) ICASSP'22	UD					
UMX 2020	S					
MetricGAN+ Interspeech'21	S					

Results on the Wall Street Journal (WSJ) and VoiceBank (VB)

Method	Superv.	Test subset	Train subset	SI-SDR (dB)	Train subset	SI-SDR (dB)
Noisy mixture	-	WSJ0-QUT	-	−2.6	-	−2.6
VAE-VEM ICASSP'20	UA	WSJ0-QUT	WSJ0	5.0		
Proposed DKF-VEM	UA	WSJ0-QUT	WSJ0	5.1		
Proposed RVAE-VEM	UA	WSJ0-QUT	WSJ0	**5.8**		
Proposed SRNN-VEM	UA	WSJ0-QUT	WSJ0	5.2		
MetricGAN-U (full) ICASSP'22	UD	WSJ0-QUT	WSJ0-QUT	N/A		
MetricGAN-U (half) ICASSP'22	UD	WSJ0-QUT	WSJ0-QUT	N/A		
UMX* 2020	S	WSJ0-QUT	WSJ0-QUT	**5.7**		
MetricGAN+* Interspeech'21	S	WSJ0-QUT	WSJ0-QUT	3.6		
Noisy mixture	-	VB-DMD	-	8.4	-	8.4
VAE-VEM ICASSP'20	UA	VB-DMD	VB	16.4		
Proposed DKF-VEM	UA	VB-DMD	VB	16.9		
Proposed RVAE-VEM	UA	VB-DMD	VB	17.1		
Proposed SRNN-VEM	UA	VB-DMD	VB	14.2		
NyTT EUSIPCO'21	UD	VB-DMD	VB-DMD (Xtra)	**17.7**		
NyTT EUSIPCO'21	UD	VB-DMD	VB-DMD	12.1		
MetricGAN-U (full) ICASSP'22	UD	VB-DMD	VB-DMD	6.5		
MetricGAN-U (half) ICASSP'22	UD	VB-DMD	VB-DMD	8.2		
UMX 2020	S	VB-DMD	VB-DMD	**14.0**		
MetricGAN+ Interspeech'21	S	VB-DMD	VB-DMD	8.5		

Results on the Wall Street Journal (WSJ) and VoiceBank (VB)

Method	Superv.	Test subset	Train subset	SI-SDR (dB)	Train subset	SI-SDR (dB)
Noisy mixture	-	WSJ0-QUT	-	−2.6	-	−2.6
VAE-VEM ICASSP'20	UA	WSJ0-QUT	WSJ0	5.0	VB	3.8
Proposed DKF-VEM	UA	WSJ0-QUT	WSJ0	5.1	VB	3.5
Proposed RVAE-VEM	UA	WSJ0-QUT	WSJ0	**5.8**	VB	4.3
Proposed SRNN-VEM	UA	WSJ0-QUT	WSJ0	5.2	VB	**4.6**
MetricGAN-U (full) ICASSP'22	UD	WSJ0-QUT	WSJ0-QUT	N/A	VB-DMD	-2.3
MetricGAN-U (half) ICASSP'22	UD	WSJ0-QUT	WSJ0-QUT	N/A	VB-DMD	-1.6
UMX* 2020	S	WSJ0-QUT	WSJ0-QUT	**5.7**	VB-DMD	**4.1**
MetricGAN+* Interspeech'21	S	WSJ0-QUT	WSJ0-QUT	3.6	VB-DMD	1.8
Noisy mixture	-	VB-DMD	-	8.4	-	8.4
VAE-VEM ICASSP'20	UA	VB-DMD	VB	16.4		
Proposed DKF-VEM	UA	VB-DMD	VB	16.9		
Proposed RVAE-VEM	UA	VB-DMD	VB	17.1		
Proposed SRNN-VEM	UA	VB-DMD	VB	14.2		
NyTT EUSIPCO'21	UD	VB-DMD	VB-DMD (Xtra)	**17.7**		
NyTT EUSIPCO'21	UD	VB-DMD	VB-DMD	12.1		
MetricGAN-U (full) ICASSP'22	UD	VB-DMD	VB-DMD	6.5		
MetricGAN-U (half) ICASSP'22	UD	VB-DMD	VB-DMD	8.2		
UMX 2020	S	VB-DMD	VB-DMD	**14.0**		
MetricGAN+ Interspeech'21	S	VB-DMD	VB-DMD	8.5		

Results on the Wall Street Journal (WSJ) and VoiceBank (VB)

Method	Superv.	Test subset	Train subset	SI-SDR (dB)	Train subset	SI-SDR (dB)
Noisy mixture	-	WSJ0-QUT	-	−2.6	-	−2.6
VAE-VEM ICASSP'20	UA	WSJ0-QUT	WSJ0	5.0	VB	3.8
Proposed DKF-VEM	UA	WSJ0-QUT	WSJ0	5.1	VB	3.5
Proposed RVAE-VEM	UA	WSJ0-QUT	WSJ0	**5.8**	VB	4.3
Proposed SRNN-VEM	UA	WSJ0-QUT	WSJ0	5.2	VB	**4.6**
MetricGAN-U (full) ICASSP'22	UD	WSJ0-QUT	WSJ0-QUT	N/A	VB-DMD	-2.3
MetricGAN-U (half) ICASSP'22	UD	WSJ0-QUT	WSJ0-QUT	N/A	VB-DMD	-1.6
UMX* 2020	S	WSJ0-QUT	WSJ0-QUT	**5.7**	VB-DMD	**4.1**
MetricGAN+* Interspeech'21	S	WSJ0-QUT	WSJ0-QUT	3.6	VB-DMD	1.8
Noisy mixture	-	VB-DMD	-	8.4	-	8.4
VAE-VEM ICASSP'20	UA	VB-DMD	VB	16.4	WSJ0	15.0
Proposed DKF-VEM	UA	VB-DMD	VB	16.9	WSJ0	16.8
Proposed RVAE-VEM	UA	VB-DMD	VB	17.1	WSJ0	textbf17.3
Proposed SRNN-VEM	UA	VB-DMD	VB	14.2	WSJ0	16.8
NyTT EUSIPCO'21	UD	VB-DMD	VB-DMD (Xtra)	**17.7**	WSJ0-QUT	N/A
NyTT EUSIPCO'21	UD	VB-DMD	VB-DMD	12.1	WSJ0-QUT	N/A
MetricGAN-U (full) ICASSP'22	UD	VB-DMD	VB-DMD	6.5	WSJ0-QUT	N/A
MetricGAN-U (half) ICASSP'22	UD	VB-DMD	VB-DMD	8.2	WSJ0-QUT	N/A
UMX 2020	S	VB-DMD	VB-DMD	**14.0**	WSJ0+QUT	**10.4**
MetricGAN+ Interspeech'21	S	VB-DMD	VB-DMD	8.5	WSJ0+QUT	3.9

Conclusions & Resources

- Unsupervised speech enhancement methods can be more "flexible".
- Useful for unsupervised sequential deep modeling.
- Can be combined with other probabilistic models (at training or test time).
- Sequential sampling is not very efficient.

Samples: `https://team.inria.fr/robotlearn/unsup-se-dvae/`
Code: `https://github.com/XiaoyuBIE1994/DVAE_SE`
Paper: `https://arxiv.org/abs/2106.12271`

DVAE's can be used for other data/tasks: check
`https://team.inria.fr/robotlearn/dvae/`!