Probabilistic generative models for audio-visual processing

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

Why unsupervised learning?

Imagine a system for {person tracking, speech denoising, body pose estimation, ...} in:



[Images under Creative Commons license]

We would like to avoid re-annotating for every new environment $\rightarrow$ Interest in unsupervised learning (and/or unsupervised domain adaptation).

And probabilistic learning?

Probabilistic generative models aim to learn a (parametric) *distribution* $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ that approximates the complex data distribution $p_{\text{data}}(\boldsymbol{x})$:

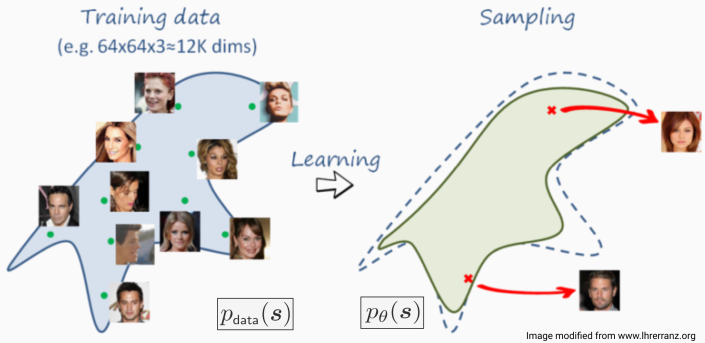


Image modified from www.lherranz.org

- Once learned, we can (ideally) sample new data.
- We can jointly learn them with other probabilistic models using *maximum likelihood*.

The Kullback-Leibler divergence and the ML formulation

The Kullback-Leilbler (KL) divergence between two distributions writes:

$$\underbrace{D_{\mathsf{KL}}(p(\boldsymbol{x})\|q(\boldsymbol{x})) = -\mathbb{E}_{p(\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right]}_{\text{Need your help!}} = -\int_{\mathcal{X}} p(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \begin{cases} \geq 0 \\ \neq D_{\mathsf{KL}}(q(\boldsymbol{x})\|p(\boldsymbol{x})) \end{cases}$$

$D_{\mathsf{KL}}(p(\boldsymbol{x})\|q(\boldsymbol{x})) = 0 \Leftrightarrow p(\boldsymbol{x}) = q(\boldsymbol{x})$.

The Kullback-Leibler divergence and the ML formulation

The Kullback-Leilbler (KL) divergence between two distributions writes:

$$\underbrace{D_{\mathsf{KL}}(p(\boldsymbol{x})\|q(\boldsymbol{x})) = -\mathbb{E}_{p(\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}\right]}_{\text{Need your help!}} = -\int_{\mathcal{X}} p(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \begin{cases} \geq 0 \\ \neq D_{\mathsf{KL}}(q(\boldsymbol{x})\|p(\boldsymbol{x})) \end{cases}$$

$D_{\mathsf{KL}}(p(\boldsymbol{x})\|q(\boldsymbol{x})) = 0 \Leftrightarrow p(\boldsymbol{x}) = q(\boldsymbol{x})$.

Given a training set $\{\boldsymbol{x}_i\}_{i=1}^N, \boldsymbol{x}_i \sim p_{\mathsf{data}}(\boldsymbol{x})$, ML minimizes the KL divergence:

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} \; D_{\mathsf{KL}}\Big(p_{\mathsf{data}}(\boldsymbol{x}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x})\Big) = \operatorname*{argmin}_{\boldsymbol{\theta}} \; -\mathbb{E}_{p_{\mathsf{data}}}\Big[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x})}{p_{\mathsf{data}}(\boldsymbol{x})}\Big]$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta}} \; \mathbb{E}_{p_{\mathsf{data}}}\Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})\Big] \approx \boxed{\operatorname*{argmax}_{\boldsymbol{\theta}} \; \frac{1}{N}\sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}$$

Interest of Latent Variables

- Speech enhancement: noisy speech (observation), clean speech (latent variable)
- Person tracking: detections (observation), person positions (latent variable)
- Representation learning: raw data (observation), representation (latent variable)

Interest of Latent Variables

- Speech enhancement: noisy speech (observation), clean speech (latent variable)
- Person tracking: detections (observation), person positions (latent variable)
- Representation learning: raw data (observation), representation (latent variable)

Let $\boldsymbol{z}$ denote the latent variable:

$$\begin{cases} \boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}) \\ \boldsymbol{x}|\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \end{cases} \rightarrow \; p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) \mathrm{d}\boldsymbol{z} = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}) \mathrm{d}\boldsymbol{z}$$

Interest of Latent Variables

- Speech enhancement: noisy speech (observation), clean speech (latent variable)
- Person tracking: detections (observation), person positions (latent variable)
- Representation learning: raw data (observation), representation (latent variable)

Let $\boldsymbol{z}$ denote the latent variable:

$$\begin{cases} \boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}) \\ \boldsymbol{x}|\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) \end{cases} \rightarrow \; p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) \mathrm{d}\boldsymbol{z} = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}) \mathrm{d}\boldsymbol{z}$$

New samples: Draw $\hat{\boldsymbol{z}} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z})$, then draw a new sample $\hat{\boldsymbol{x}} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|\hat{\boldsymbol{z}})$.

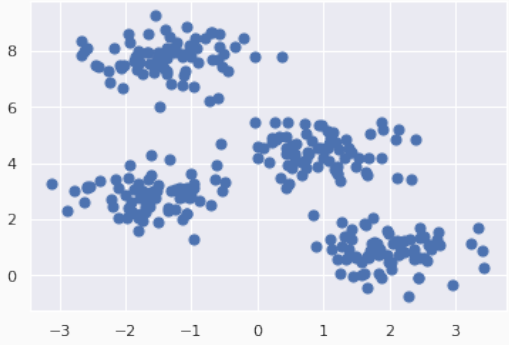
Learning: How to estimate $\boldsymbol{\theta}$ with the ML formulation?

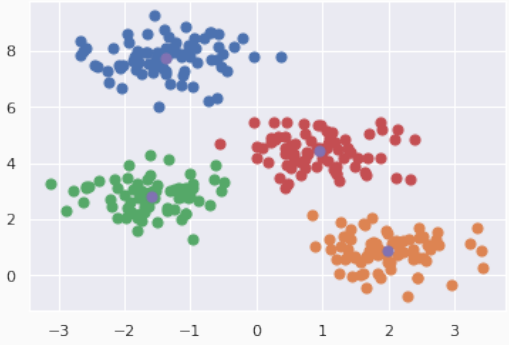
Inference: How to get the most likely latent $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$?

From shallow GMMs to deep VAEs

Simple example: clustering

Definition: find groups of data points without labels.





The Gaussian Mixture Model (GMM)

Definition:

- For each $\boldsymbol{x}_n$ there is a latent variable $\boldsymbol{z}_n$ taking values from 1 to K: $\boldsymbol{z}_n \in \{1, \ldots, K\}$.

The Gaussian Mixture Model (GMM)

Definition:

- For each $\boldsymbol{x}_n$ there is a latent variable $\boldsymbol{z}_n$ taking values from 1 to K: $\boldsymbol{z}_n \in \{1, \ldots, K\}$.
- Its prior probability is defined as: $p(\boldsymbol{z}_n = k) = \pi_k \geq 0$, with $\sum_{k=1}^{K} \pi_k = 1$.

The Gaussian Mixture Model (GMM)

Definition:

- For each $\boldsymbol{x}_n$ there is a latent variable $\boldsymbol{z}_n$ taking values from 1 to K: $\boldsymbol{z}_n \in \{1, \ldots, K\}$.
- Its prior probability is defined as: $p(\boldsymbol{z}_n = k) = \pi_k \geq 0$, with $\sum_{k=1}^{K} \pi_k = 1$.
- Given z_n, the data point is modeled as a multivariate Gaussian:

$$p(\boldsymbol{x}_n | \boldsymbol{z}_n = k) = \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Advantages:

1. Having $\pi_1, \ldots, \pi_K$ means that groups can be differently populated.
2. The shape of the groups is modeled by $\boldsymbol{\Sigma}_k$.
3. The parameters are: $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$.

Maximum likelihood for GMM

Let's compute $p(\boldsymbol{x}_n)$

$$p(\boldsymbol{x}_n) = \sum_{k=1}^{K} p(\boldsymbol{x}_n, \boldsymbol{z}_n = k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

The log-likelihood:

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{X}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

Computing directly ML by $\frac{\partial \mathcal{L}}{\partial \pi_k} = 0$, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = 0$ or $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = 0$ is very difficult.

(Expected complete-data) log-likelihood

We have seen that $\log p(\boldsymbol{x})$ does not work well with derivatives. However, $\log p(\boldsymbol{x}, \boldsymbol{z})$ does!

Problem: $\boldsymbol{z}$ is not observed, thus $\sum_n \log p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is a random variable.

(Expected complete-data) log-likelihood

We have seen that $\log p(\boldsymbol{x})$ does not work well with derivatives. However, $\log p(\boldsymbol{x}, \boldsymbol{z})$ does!

Problem: $\boldsymbol{z}$ is not observed, thus $\sum_n \log p(\boldsymbol{x}_n, \boldsymbol{z}_n)$ is a random variable.

Given an initial value of the parameters, $\boldsymbol{\theta}^0$, let's take the expectation w.r.t. the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}^0)$ [we will justify this choice later on]:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^0) = \sum_n \mathbb{E}_{p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^0)} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n; \boldsymbol{\theta})$$

This function is called: *expected complete-data log-likelihood*, and is the main mathematical object when working with EM algorithms.

The EM algorithm for GMM

Given $\boldsymbol{\theta}^0$, we use the *expected complete-data log-likelihood* $\mathcal{Q}$. For iteration $r = 1, \ldots, R$:

1. Expectation [E-step]:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{r-1}) = \mathbb{E}_{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^{r-1})} \log p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta})$$

2. Maximisation [M-step]:

$$\boldsymbol{\theta}^r = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{r-1})$$

(observations $\boldsymbol{x} = \{\boldsymbol{x}_n\}_{n=1}^N$, latent variables $\boldsymbol{z} = \{\boldsymbol{z}_n\}_{n=1}^N$, parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$)

The EM algorithm for GMM (II)

E-step The posterior distribution writes:

$$p(\boldsymbol{z}_n = k|\boldsymbol{x}_n; \boldsymbol{\theta}^0) = \frac{p(\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)}{\sum_\ell p(\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)} = \frac{\pi_k^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k^0, \boldsymbol{\Sigma}_k^0)}{\sum_\ell \pi_\ell^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_\ell^0, \boldsymbol{\Sigma}_\ell^0)} = \eta_{nk}$$

The EM algorithm for GMM (II)

E-step The posterior distribution writes:

$$p(\boldsymbol{z}_n = k|\boldsymbol{x}_n; \boldsymbol{\theta}^0) = \frac{p(\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)}{\sum_\ell p(\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)} = \frac{\pi_k^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k^0, \boldsymbol{\Sigma}_k^0)}{\sum_\ell \pi_\ell^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_\ell^0, \boldsymbol{\Sigma}_\ell^0)} = \eta_{nk}$$

The expected complete-data log-likelihood writes:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^0) = \sum_{n=1}^{N} \sum_{k=1}^{K} \eta_{nk} \log \pi_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

The EM algorithm for GMM (II)

E-step The posterior distribution writes:

$$p(\boldsymbol{z}_n = k|\boldsymbol{x}_n; \boldsymbol{\theta}^0) = \frac{p(\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = k; \boldsymbol{\theta}^0)}{\sum_\ell p(\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)p(\boldsymbol{x}_n|\boldsymbol{z}_n = \ell; \boldsymbol{\theta}^0)} = \frac{\pi_k^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k^0, \boldsymbol{\Sigma}_k^0)}{\sum_\ell \pi_\ell^0 \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_\ell^0, \boldsymbol{\Sigma}_\ell^0)} = \eta_{nk}$$

The expected complete-data log-likelihood writes:

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^0) = \sum_{n=1}^{N} \sum_{k=1}^{K} \eta_{nk} \log \pi_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

M-step The optimal value for $\boldsymbol{\mu}_k$ writes:

$$\boldsymbol{\mu}_k^* = \frac{1}{\sum_{n=1}^{N} \eta_{nk}} \sum_{n=1}^{N} \eta_{nk} \boldsymbol{x}_n$$

But why does the EM work?

The main mathematical object in EM is $\mathcal{Q}$ (The expected complete-data log-likelihood).

What is the relationship with the log-likelihood? Let's take any distribution of $\boldsymbol{z}$, $q(\boldsymbol{z})$:

$$\log p(\boldsymbol{x}) = \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})}\right]}_{\text{M-step - } \mathcal{Q}} + \underbrace{D_{\mathsf{KL}}\Big(q(\boldsymbol{z})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)}_{\text{E-step - KL}}$$

But why does the EM work?

The main mathematical object in EM is $\mathcal{Q}$ (The expected complete-data log-likelihood).

What is the relationship with the log-likelihood? Let's take any distribution of $\boldsymbol{z}$, $q(\boldsymbol{z})$:

$$\log p(\boldsymbol{x}) = \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})}\right]}_{\text{M-step - } \mathcal{Q}} + \underbrace{D_{\mathsf{KL}}\Big(q(\boldsymbol{z})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)}_{\text{E-step - KL}}$$

Another interpretation. Given $\boldsymbol{\theta}^0$:

1. Set $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^0)$, minimise KL (thus maximize $\mathcal{Q}$) w.r.t. $q(\boldsymbol{z})$. The E-step reduces the distance between log-likelihood and $\mathcal{Q}$.
2. Maximize $\mathcal{Q}$ w.r.t. $\boldsymbol{\theta}$:
$$\mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^0) = \mathbf{E}_{q(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\theta})}{q(\boldsymbol{z})}\right].$$
The M-step pushes $\mathcal{Q}$ and therefore pushes the log-likelihood.

The Exact EM

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) = \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})}\right]}_{\text{M-step - } \mathcal{Q}} + \underbrace{D_{\mathsf{KL}}\left(q(\boldsymbol{z})\middle\|p(\boldsymbol{z}|\boldsymbol{x})\right)}_{\text{E-step - KL}}$$

Given $\boldsymbol{\theta}^0$:

1. Set $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^0)$, and compute $\mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^0)$.
2. Maximize $\mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^0)$ w.r.t. $\boldsymbol{\theta}$.

The Exact EM

$$\log p(\boldsymbol{x};\boldsymbol{\theta}) = \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})}\right]}_{\text{M-step - } \mathcal{Q}} + \underbrace{D_{\mathsf{KL}}\left(q(\boldsymbol{z})\middle\|p(\boldsymbol{z}|\boldsymbol{x})\right)}_{\text{E-step - KL}}$$

Given $\boldsymbol{\theta}^0$:

1. Set $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}^0)$, and compute $\mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^0)$.
2. Maximize $\mathcal{Q}(\boldsymbol{\theta},\boldsymbol{\theta}^0)$ w.r.t. $\boldsymbol{\theta}$.

Several potential issues:

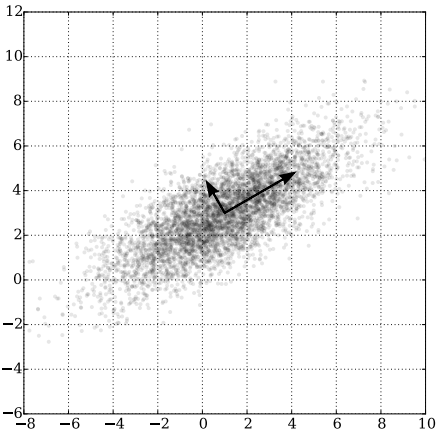
- The posterior distribution does not exist/is not computationally tractable.
- The expectation cannot be taken analytically.
- The maximization w.r.t. $\boldsymbol{\theta}$ does not have close-form solution.

What about continuous latent variables?

E.g.: extract a representation $(\boldsymbol{z})$ of each $(\boldsymbol{x})$.

Linear model (PPCA) $(D \ll F)$:

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \boldsymbol{I}), \quad \boldsymbol{z} \in \mathbb{R}^D.$$
$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{A}\boldsymbol{z} + \boldsymbol{b}, \nu\boldsymbol{I}), \quad \boldsymbol{x} \in \mathbb{R}^F.$$



[Image from Wikimedia Commons]

What about continuous latent variables?

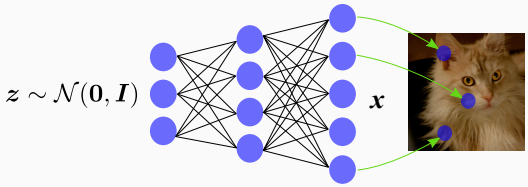
E.g.: extract a representation ($\boldsymbol{z}$) of each ($\boldsymbol{x}$).

Linear model (PPCA) ($D \ll F$):

$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I}), \quad \boldsymbol{z} \in \mathbb{R}^D.$$
$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{A}\boldsymbol{z} + \boldsymbol{b}, \nu\boldsymbol{I}), \quad \boldsymbol{x} \in \mathbb{R}^F.$$

Non-linear model (same $p(\boldsymbol{z})$):

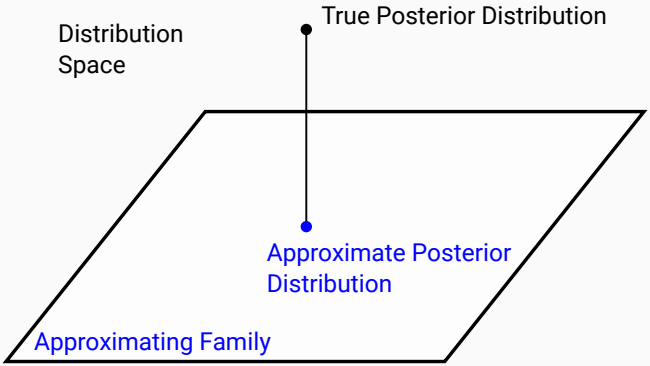
$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\Big(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z})\Big)$$



The posterior distribution

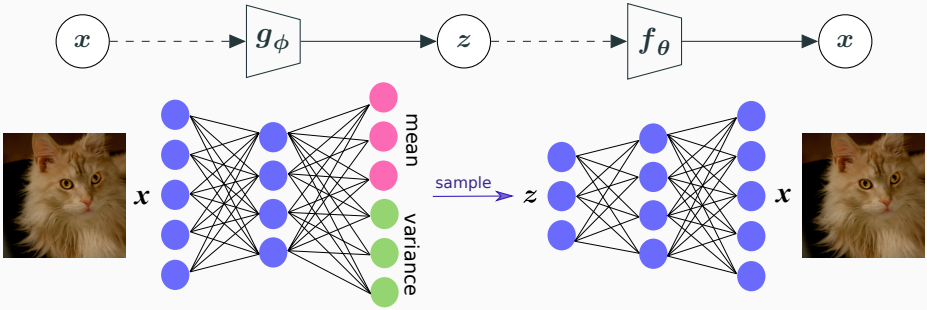
If $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is non-linear, $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ cannot be computed analytically, it needs to be approximated.

The idea is to find the best candidate within a family of distributions.



Overall architecture

If we "chain" the posterior (encoder) and the generative (decoder) model:



$$p(\boldsymbol{z}|\boldsymbol{x}) \approx q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\Big(\boldsymbol{z}; \tilde{\boldsymbol{\mu}}_{\boldsymbol{\phi}}(\boldsymbol{x}), \tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\phi}}(\boldsymbol{x})\Big) \qquad p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\Big(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z})\Big)$$

But how do we optimise for the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$?

Learning - ELBO

If we recall the formulation for the EM:

$$\log p(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\Big[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})}\Big] + D_{\mathsf{KL}}\Big(q(\boldsymbol{z}|\boldsymbol{x})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)$$

Learning - ELBO

If we recall the formulation for the EM:

$$\log p(\boldsymbol{x}) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\Big[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})}\Big] + D_{\mathsf{KL}}\Big(q(\boldsymbol{z}|\boldsymbol{x})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)$$

Problem: the second term cannot be computed! But it's positive:

$$\log p(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) \geq \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\Big[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\Big]$$

$$\log p(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{\phi}) \geq \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\Big]}_{\text{Reconstruction}} \underbrace{- D_{\mathsf{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\Big\|p(\boldsymbol{z})\Big)}_{\text{Regularisation}}$$

This is known as **Evidence Lower-BOund or ELBO**: $\mathcal{L}_{\mathsf{ELBO}}(\boldsymbol{\theta},\boldsymbol{\phi})$.

Be VERY careful with these expressions: They look alike, but they are NOT the same.

Learning - Practical comments

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})}\right] = \mathcal{L}_{\mathsf{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi})$$

- One needs to sample from $q(\boldsymbol{z}|\boldsymbol{x})$ to compute the expectation.
- This sampling needs the "reparametrisation trick" to be differentiable.
- Most libraries solve a minimisation problem $\rightarrow$ we need to provide $-\mathcal{L}_{\mathsf{ELBO}}(\boldsymbol{\theta}, \boldsymbol{\phi})$.

ML with latent variables: EM vs VAE?

ML with latent variable ($\boldsymbol{z}$) leads to EM[1] and VI[2] build from ($q(\boldsymbol{z})$ is an arbitrary distribution):

$$\log p(\boldsymbol{x}) = \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})}\right]}_{\text{M-step or VLB}} + \underbrace{D_{\mathsf{KL}}\Big(q(\boldsymbol{z})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)}_{\text{E-step}}$$

Exact EM	Variational AutoEncoder
Simple $p(\boldsymbol{z}\|\boldsymbol{x})$	$p(\boldsymbol{z}\|\boldsymbol{x})$???
$q(\boldsymbol{z}) = p(\boldsymbol{z}\|\boldsymbol{x})$	$q(\boldsymbol{z}) = q_{\boldsymbol{\phi}}(\boldsymbol{z})$
Closed-form!	$q_{\boldsymbol{\phi}}$ optimises ELBO/VLB
$D_{\mathsf{KL}} = 0$	$D_{\mathsf{KL}} > 0$ no closed form!

[1] Dempster, A.P., *et. al.*, (1977), Journal of the Royal Statistical Society.
[2] Jordan, M. I., *et. al.*, (1999),. Machine Learning.

ML with latent variables: EM vs VAE?

ML with latent variable ($\boldsymbol{z}$) leads to EM[1] and VI[2] build from ($q(\boldsymbol{z})$ is an arbitrary distribution):

$$\log p(\boldsymbol{x}) = \underbrace{\mathbb{E}_{q(\boldsymbol{z})}\left[\log \frac{p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})}\right]}_{\text{M-step or VLB}} + \underbrace{D_{\mathsf{KL}}\Big(q(\boldsymbol{z})\Big\|p(\boldsymbol{z}|\boldsymbol{x})\Big)}_{\text{E-step}}$$

Exact EM	Variational EM	Variational AutoEncoder
Simple $p(\boldsymbol{z}\|\boldsymbol{x})$	$p(\boldsymbol{z}\|\boldsymbol{x})$ comp. complex	$p(\boldsymbol{z}\|\boldsymbol{x})$???
$q(\boldsymbol{z}) = p(\boldsymbol{z}\|\boldsymbol{x})$	$q([\boldsymbol{z}_1, \boldsymbol{z}_2]) = q_1(\boldsymbol{z}_1)q_2(\boldsymbol{z}_2)$	$q(\boldsymbol{z}) = q_\phi(\boldsymbol{z})$
Closed-form!	$q_1, q_2 = \operatorname{argmin} D_{\mathsf{KL}}$	q_ϕ optimises ELBO/VLB
$D_{\mathsf{KL}} = 0$	$D_{\mathsf{KL}} > 0$ but closed form!	$D_{\mathsf{KL}} > 0$ no closed form!

[1] Dempster, A.P., *et. al.*, (1977), Journal of the Royal Statistical Society.
[2] Jordan, M. I., *et. al.*, (1999),. Machine Learning.

Audio-Visual Unsupervised Speech Enhancement with VAEs

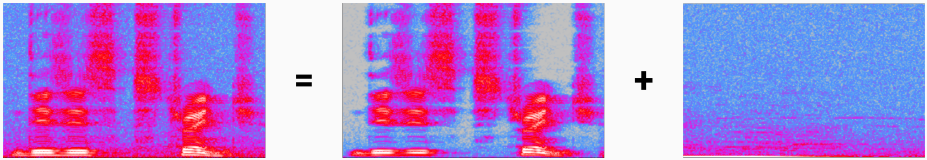
Mostafa Sadeghi

Laurent Girin

Radu Horaud

Speech Enhancement & Wiener Filter

Extract the latent clean speech signal from the observed noisy mixture. (STFT domain)



$\underbrace{\mathbf{s}}_{\text{Clean speech}} + \underbrace{\mathbf{b}}_{\text{Noise signal}} = \underbrace{\mathbf{x}}_{\text{Noisy mixture}}$

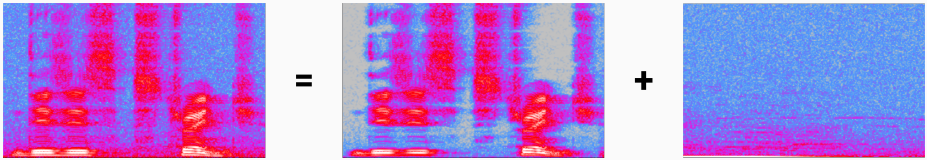
Speech Enhancement & Wiener Filter

Minimize MSE $\rightarrow$ Wiener Filter

(operations are element-wise)

$$\hat{s} = \frac{\sigma_s}{\sigma_s + \sigma_b} x$$

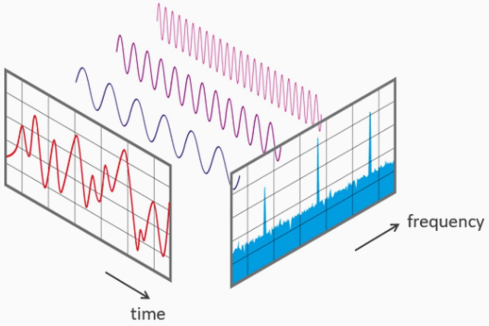
Extract the latent clean speech signal from the observed noisy mixture. (STFT domain)



$\underbrace{\mathbf{s}}_{\text{Clean speech}} + \underbrace{\mathbf{b}}_{\text{Noise signal}} = \underbrace{\mathbf{x}}_{\text{Noisy mixture}}$

Representing audio: time vs. frequency

Fourier domain decomposes the signal in frequencies:

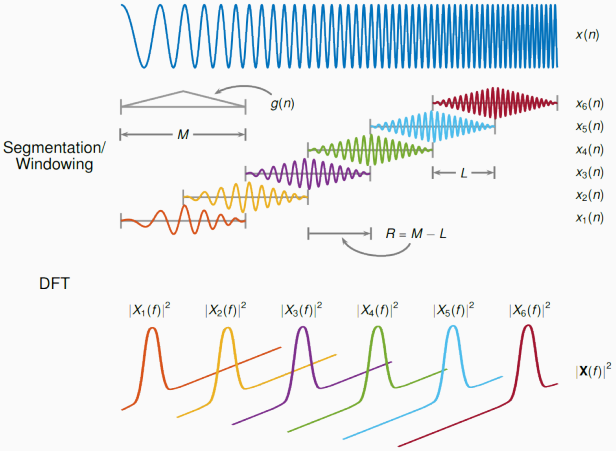


[Image from https://dev.to/trekhleb/]

Problem: we have to choose either time **or** frequency.

The short-time Fourier transform (STFT)

STFT: segment the input signal, and apply DFT to each segment.

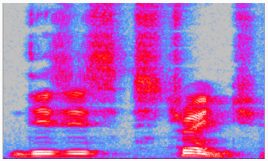


[Image from Mathworks]

Let's see it!

- Sine frequency sweep (pure sine of increasing frequency).
- Leyenda (piano, harmonics).
- O mio babbino caro (opera, vibrato).
- A few good men (speech, separation).
- ...

Unsupervised Probabilistic SE: paradigm



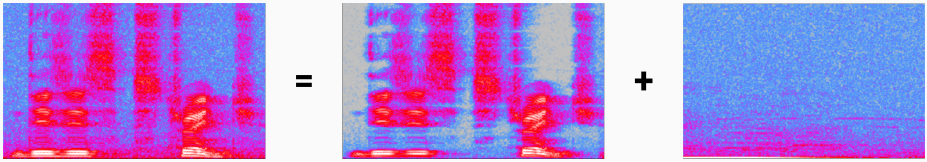
$\underbrace{\mathbf{s}}_{\text{Clean speech}}$ + $\underbrace{\mathbf{b}}_{\text{Noise signal}}$ = $\underbrace{\mathbf{x}}_{\text{Noisy mixture}}$

Train – Model for clean data: $\{\boldsymbol{s}_i\}_{i=1}^{N}$.

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

Then freeze $\boldsymbol{\theta}$ at test/adaptation time.

Unsupervised Probabilistic SE: paradigm

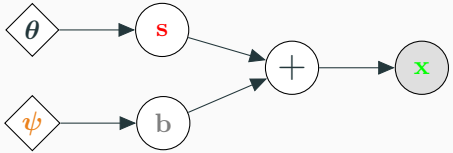


Train – Model for clean data: $\{\boldsymbol{s}_i\}_{i=1}^N$.

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

Then freeze $\boldsymbol{\theta}$ at test/adaptation time.

Test – learn the noise parameters from noisy samples $\mathbf{x}$ and estimate the clean speech $\hat{\boldsymbol{s}}$:



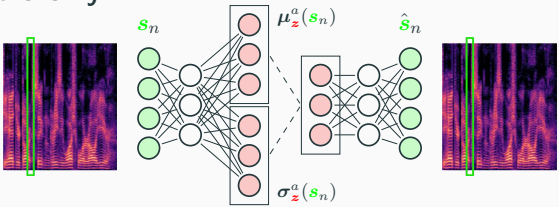
Audio-visual Speech Enhancement (AV-SE)

- Visual data (lip motion) provide complementary information about the unknown speech.
- For highly noisy audio recordings, visual information can be very helpful.

We investigate the VAE framework to fuse audio and visual data for speech enhancement.

Mono-modal VAEs: the baselines

Audio-only:[3]

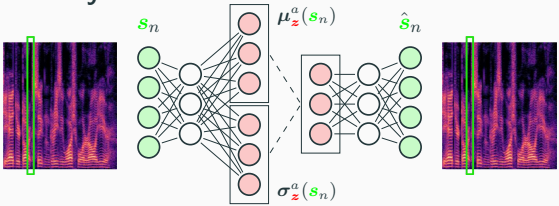


$$p_\theta(s_n|z_n) = \mathcal{N}_c\Big(\mathbf{0}, \text{diag}(\sigma_s(z_n))\Big)$$
$$q_\phi(z_n|s_n) = \mathcal{N}\Big(\mu_z^a(s_n), \text{diag}(\sigma_z^a(s_n))\Big)$$

[3] Leglaive, S., *et. al.*, (2018), IEEE MLSP.

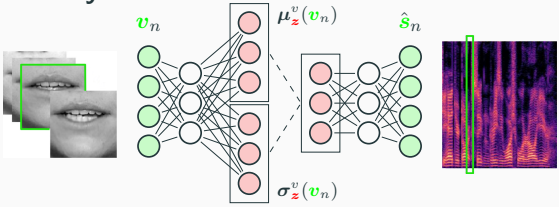
Mono-modal VAEs: the baselines

Audio-only:[3]



$$p_{\theta}(s_n|z_n) = \mathcal{N}_c\Big(\mathbf{0}, \text{diag}(\sigma_s(z_n))\Big)$$
$$q_{\phi}(z_n|s_n) = \mathcal{N}\Big(\mu_z^a(s_n), \text{diag}(\sigma_z^a(s_n))\Big)$$

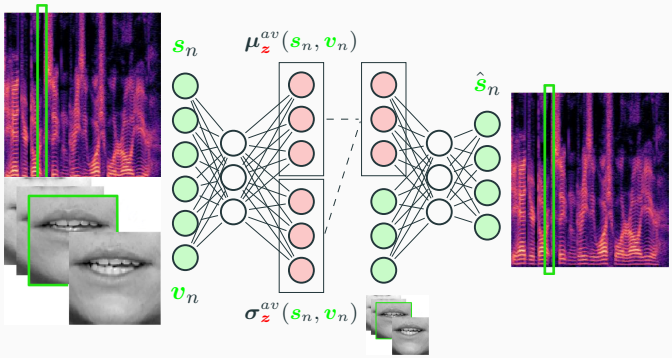
Video-only:



$$p_{\theta}(s_n|z_n) = \mathcal{N}_c\Big(\mathbf{0}, \text{diag}(\sigma_s(z_n))\Big)$$
$$q_{\phi}(z_n|v_n) = \mathcal{N}\Big(\mu_z^v(v_n), \text{diag}(\sigma_z^v(v_n))\Big)$$

[3] Leglaive, S., *et. al.*, (2018), IEEE MLSP.

Audio-visual Conditional VAE[4]

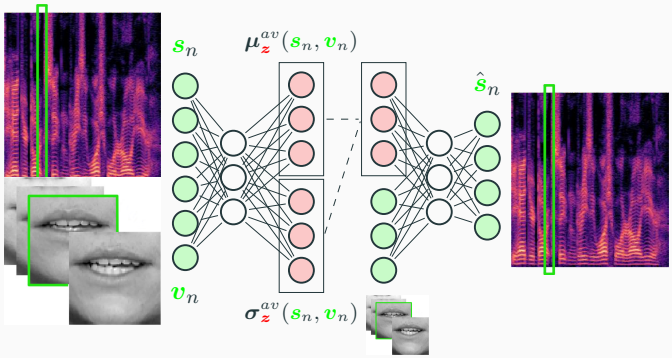


$$p_{\theta}(s_n|z_n, v_n) = \mathcal{N}_c\Big(\mathbf{0}, \text{diag}(\sigma_s(z_n, v_n))\Big)$$

$$q_{\phi}(z_n|v_n, s_n) = \mathcal{N}\Big(\mu_z^{av}(v_n, s_n), \text{diag}(\sigma_z^{av}(v_n, s_n))\Big)$$

[4] Sadeghi, M., *et. al.*, (2020), IEEE TASLP.

Audio-visual Conditional VAE[4]



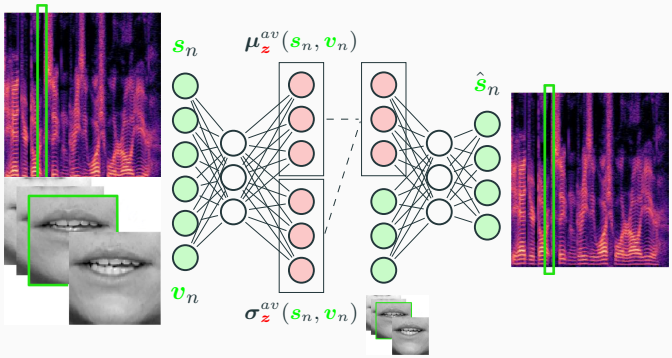
$$p_{\theta}(s_n|z_n, v_n) = \mathcal{N}_c\Big(\mathbf{0}, \text{diag}(\sigma_s(z_n, v_n))\Big)$$

$$q_{\phi}(z_n|v_n, s_n) = \mathcal{N}\Big(\mu_z^{av}(v_n, s_n), \text{diag}(\sigma_z^{av}(v_n, s_n))\Big)$$

What will this learn?

[4] Sadeghi, M., *et. al.*, (2020), IEEE TASLP.

Audio-visual Conditional VAE[4]



$$p_{\theta}(s_n|z_n, v_n) = \mathcal{N}_c\Big(\mathbf{0}, \text{diag}(\sigma_s(z_n, v_n))\Big)$$

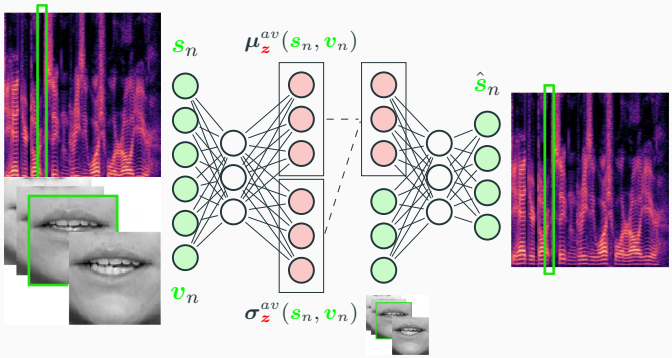
$$q_{\phi}(z_n|v_n, s_n) = \mathcal{N}\Big(\mu_z^{av}(v_n, s_n), \text{diag}(\sigma_z^{av}(v_n, s_n))\Big)$$

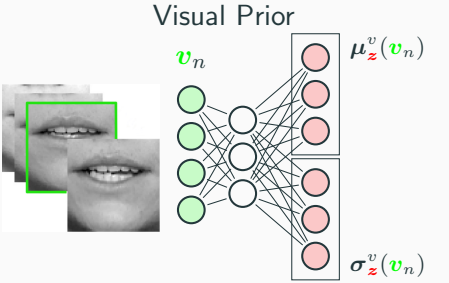
What will this learn?

To ignore the video!

[4] Sadeghi, M., *et. al.*, (2020), IEEE TASLP.

Audio-visual Conditional VAE[4]





$$p_{\theta}(s_n|z_n, v_n) = \mathcal{N}_c\Big(\mathbf{0}, \text{diag}(\sigma_s(z_n, v_n))\Big)$$

$$p_{\theta}(z_n|v_n) = \mathcal{N}\Big(\mu_z^v(v_n), \text{diag}(\sigma_z^v(v_n))\Big)$$

$$q_{\phi}(z_n|v_n, s_n) = \mathcal{N}\Big(\mu_z^{av}(v_n, s_n), \text{diag}(\sigma_z^{av}(v_n, s_n))\Big)$$

[4] Sadeghi, M., *et. al.*, (2020), IEEE TASLP.

Learning AV Conditional VAE

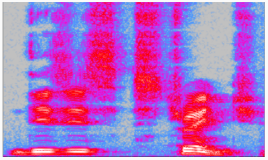
On top of adding a video-only prior, we optimize a modified ELBO:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_n (1-\alpha)\Big(\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{s}_n,\boldsymbol{v}_n;)}\Big[\ln p_{\boldsymbol{\theta}}(\boldsymbol{s}_n|\boldsymbol{z}_n,\boldsymbol{v}_n)\Big] - D_{\mathrm{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{s}_n,\boldsymbol{v}_n) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{z}_n|\boldsymbol{v}_n)\Big)\Big)$$
$$+\alpha\,\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z}_n|\boldsymbol{v}_n)}\Big[\ln p_{\boldsymbol{\theta}}(\boldsymbol{s}_n|\boldsymbol{z}_n,\boldsymbol{v}_n)\Big]$$

$\Rightarrow \alpha > 0$ gives some reconstruction power to the visual prior!

This provides the parameters of the clean speech model $\boldsymbol{\theta}$. Let's enhance!

AV Conditional VAE for Speech Enhancement



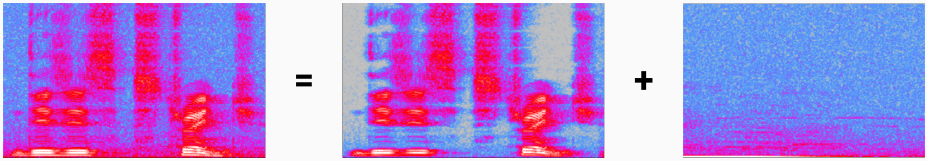
$\underbrace{\mathbf{s}}_{\text{Clean speech}}$ + $\underbrace{\mathbf{b}}_{\text{Noise signal}}$ = $\underbrace{\mathbf{x}}_{\text{Noisy mixture}}$

Train – Model for clean data: $\{\boldsymbol{s}_i\}_{i=1}^{N}$.

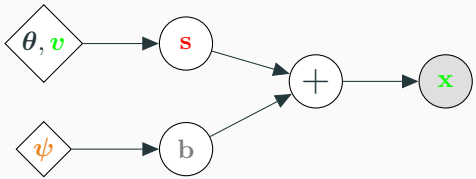
$$p_{\boldsymbol{\theta}}(\boldsymbol{s}|\boldsymbol{v}) \approx p_{\text{data}}(\boldsymbol{s}|\boldsymbol{v})$$

Then freeze $\boldsymbol{\theta}$ at test/adaptation time.

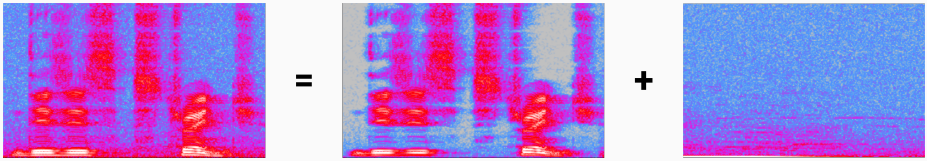
AV Conditional VAE for Speech Enhancement



Test – learn the noise parameters from noisy samples $\mathbf{x}$ and estimate the clean speech $\hat{s}$:



AV Conditional VAE for Speech Enhancement



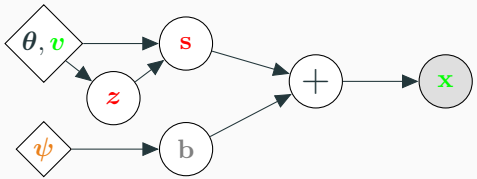
Speech Enhancement:

(Sample $\boldsymbol{z}$ from $p(\boldsymbol{z}|\boldsymbol{x}_n, \boldsymbol{v}_n)$)

$$\hat{\boldsymbol{s}}_n = \left(\frac{1}{R} \sum_{r=1}^{R} \frac{\boldsymbol{\sigma}_s(\mathbf{z}^{(r)}, \boldsymbol{v}_n)}{\boldsymbol{\sigma}_s(\mathbf{z}^{(r)}, \boldsymbol{v}_n) + \psi_n^*} \right) \boldsymbol{x}_n.$$

Wiener-like filter!

Test – learn the noise parameters from noisy samples $\mathbf{x}$ and estimate the clean speech $\hat{\boldsymbol{s}}$:



But how do we learn ψ^*?

We opt for an EM-like solution. Given an initial value of the parameters ψ^0:

$$Q(\psi, \psi^0) = \sum_n \mathbb{E}_{p(s_n, z_n | x_n, v_n; \psi^0, \boldsymbol{\theta})} [\log p(x_n, s_n, z_n | v_n; \psi, \boldsymbol{\theta})].$$

But how do we learn ψ^*?

We opt for an EM-like solution. Given an initial value of the parameters ψ^0:

$$Q(\psi, \psi^0) = \sum_n \mathbb{E}_{p(s_n, z_n | x_n, v_n; \psi^0, \theta)}[\log p(x_n, s_n, z_n | v_n; \psi, \theta)].$$

Stop! There is an *easier* way: $p(x_n, z_n | v_n; \psi) = \int p(x_n, s_n, z_n | v_n; \psi) \mathrm{d}s_n$

Approximate $\mathcal{Q}$ via Monte-Carlo sampling:

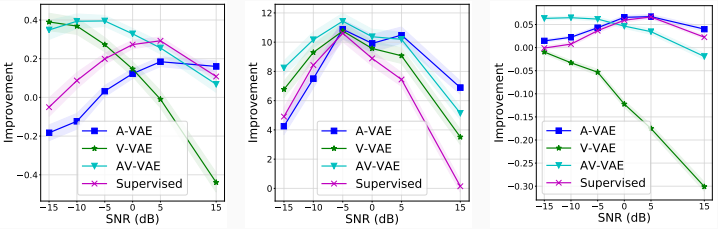
- **E-Step**: $Q(\psi; \psi^0) \approx \sum_{n=1}^N \frac{1}{R} \sum_{r=1}^R \ln p(x_n, z_n^{(r)} | v_n; \psi)$
  The samples $\left\{ z_n^{(r)} \right\}_{r=1,\dots,R}$ are i.i.d. and drawn from $p(z_n | x_n, v_n; \psi^0)$.

- **M-Step**: $\psi^1 \leftarrow \mathrm{argmax}_\psi \; Q(\psi; \psi^0) \Rightarrow$ multiplicative update rules.
  Standard formulae for NMF, not detailed here.

Results

Performance measures (the higher, the better) – improvement w.r.t. the noisy mixture:

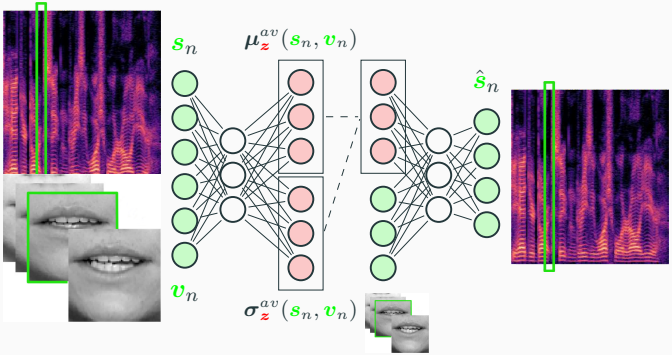
- Perceptual evaluation of speech quality (PESQ).
- Signal-to-distortion ratio (SDR).
- Short-time objective intelligibility (STOI).



(a) PESQ **(b)** SDR **(c)** STOI

Limitations: systematic AV fusion

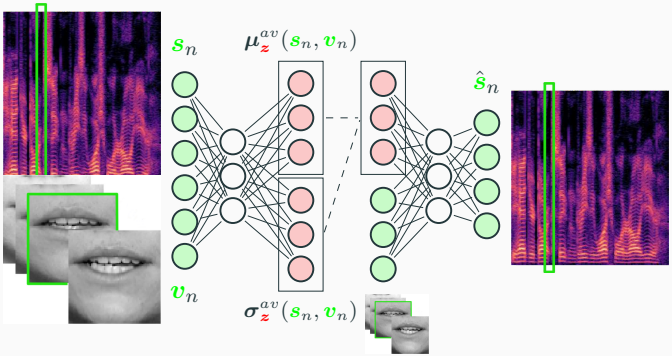
From the model design:



.

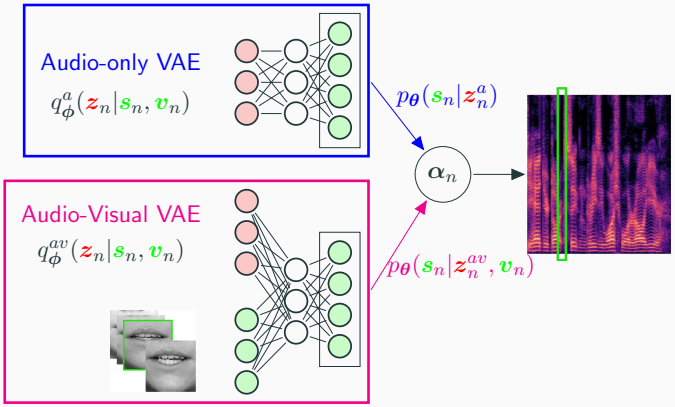
Limitations: systematic AV fusion

From the model design:



. What about clutter?

VAE Mixture Model[5]



$\boldsymbol{\alpha}_n$ is the "mixing" latent variable.

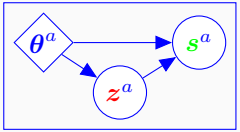
$$\begin{cases} \boldsymbol{\alpha}_n = 1 \leftrightarrow \text{Audio-only} \\ \boldsymbol{\alpha}_n = 0 \leftrightarrow \text{Audio-Visual} \end{cases}$$

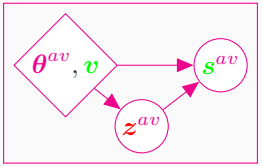
Prior: $p(\alpha_n) = \pi^{\alpha_n}(1-\pi)^{1-\alpha_n}$.

The Audio-only and the Audio-Visual VAE are **mixed without supervision**.

[5] Sadeghi, M., *et. al.*, (2021), IEEE ICASSP.

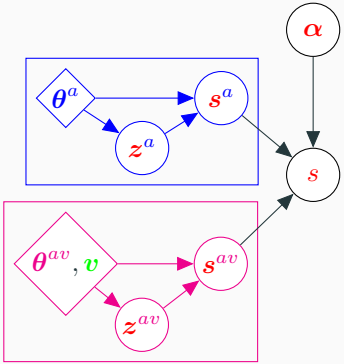
Train, adaptation & speech enhancement





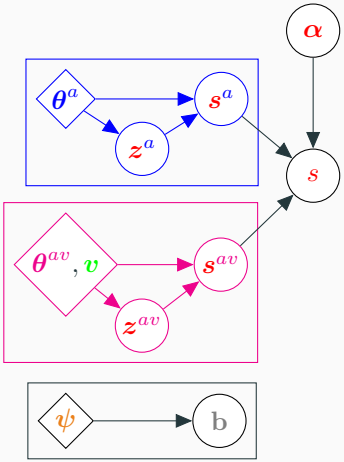
- Train A-VAE (with $\boldsymbol{s}$) and AV-VAE (with $(\boldsymbol{s}, \boldsymbol{v})$).

Train, adaptation & speech enhancement



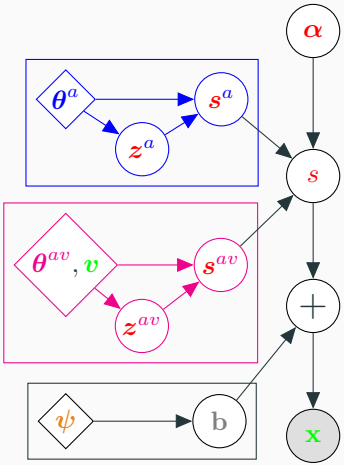
- Train A-VAE (with s) and AV-VAE (with (s, v)).
- The clean speeh is now latent (s^a, s^{av}), add the mixing latent variable (α).

Train, adaptation & speech enhancement



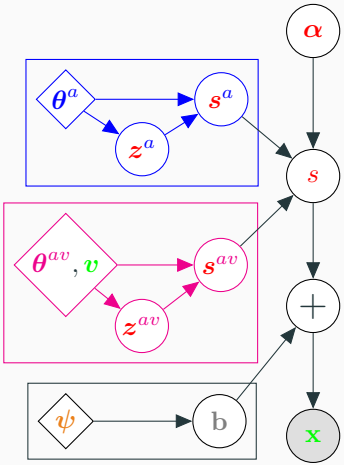
- Train A-VAE (with s) and AV-VAE (with (s, v)).
- The clean speeh is now latent (s^a, s^{av}), add the mixing latent variable (α).
- We also consider the noise variable and params.

Train, adaptation & speech enhancement



- Train A-VAE (with $\boldsymbol{s}$) and AV-VAE (with $(\boldsymbol{s}, \boldsymbol{v})$).
- The clean speeh is now latent $(\boldsymbol{s}^a, \boldsymbol{s}^{av})$, add the mixing latent variable $(\boldsymbol{\alpha})$.
- We also consider the noise variable and params.
- Learn $\boldsymbol{\psi}^*$ and estimate the clean speech. By defining $\gamma_n(\boldsymbol{z}_n, \boldsymbol{v}_n) = \left(\pi_n(\boldsymbol{\sigma}_s^a(\boldsymbol{z}_n))^{-1} + (1-\pi_n)(\boldsymbol{\sigma}_s^{av}(\boldsymbol{z}_n, \boldsymbol{v}_n))^{-1}\right)^{-1}$

Train, adaptation & speech enhancement



- Train A-VAE (with s) and AV-VAE (with (s, v)).
- The clean speeh is now latent (s^a, s^{av}), add the mixing latent variable (α).
- We also consider the noise variable and params.
- Learn ψ^* and estimate the clean speech. By defining $\gamma_n(z_n, v_n) = \left(\pi_n(\sigma_s^a(z_n))^{-1} + (1-\pi_n)(\sigma_s^{av}(z_n, v_n))^{-1}\right)^{-1}$:

$$\hat{s}_n = \frac{1}{R}\sum_{r=1}^{R} \frac{\gamma_n(z_n^{(r)}, v_n)}{\gamma_n(z_n^{(r)}, v_n) + \psi_n^*}\, x_n$$

Learning the noise parameters with VAE-MM

Ideally:

$$Q(\boldsymbol{\psi}, \boldsymbol{\psi}^0) = \sum_{n=1}^{N} \mathbb{E}_{p(\boldsymbol{s}_n, \boldsymbol{z}_n, \alpha_n | \mathbf{x}_n, \boldsymbol{v}_n; \boldsymbol{\psi}^0)} [\log p(\mathbf{x}_n, \boldsymbol{s}_n, \boldsymbol{z}_n, \alpha_n | \boldsymbol{v}_n; \boldsymbol{\psi})]$$

We need to approximate this posterior:

$$p(\boldsymbol{s}_n, \boldsymbol{z}_n, \alpha_n | \mathbf{x}_n, \boldsymbol{v}_n; \boldsymbol{\psi}^0) \approx q(\boldsymbol{s}_n, \boldsymbol{z}_n, \alpha_n) = q_{\boldsymbol{s}}(\boldsymbol{s}_n)\ q_{\boldsymbol{z}}(\boldsymbol{z}_n)\ q_{\alpha}(\alpha_n).$$

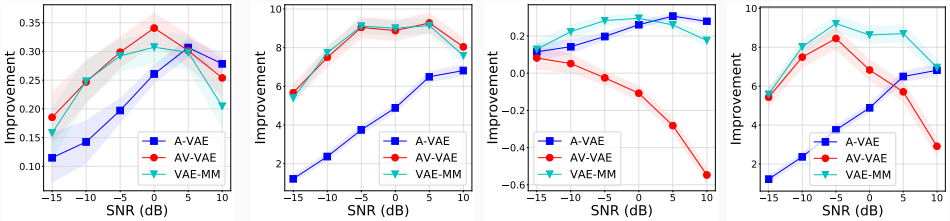
Of course we will need to sample from $q_{\boldsymbol{z}}$, but in that case:

1. q_{α} is a Bernoulli distribution.
2. $q_{\boldsymbol{s}}$ is a complex Gaussian distribution.

Settings & Results

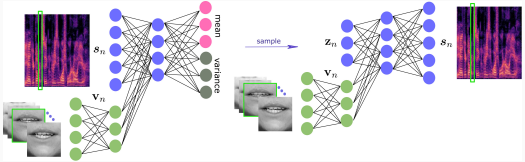
- **Noisy+clean speech**: NTCD-TIMIT database [Abdelaziz, 2017]
- **VAE models**: Pre-trained A-VAE and AV-VAE [Sadeghi et al., 2019]
- **Setup**: Very similar than in the previous experiments. Clean and noisy lips region visual information ($\sim$ one-third of total video frames per sample).

Improvement with respect to the input:



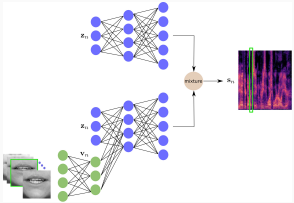
(a) PESQ (clean) **(b)** SDR (clean) **(c)** PESQ (noisy) **(d)** SDR (noisy)

Conditional VAE, VAE-MM and Mixture of Inference Networks VAE



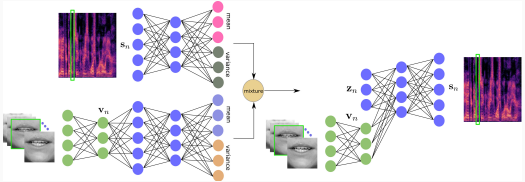
Conditional VAE:

- Training VAE via SGD.
- Systematic AV fusion.
- Enhancement via MECM.



VAE-MM:

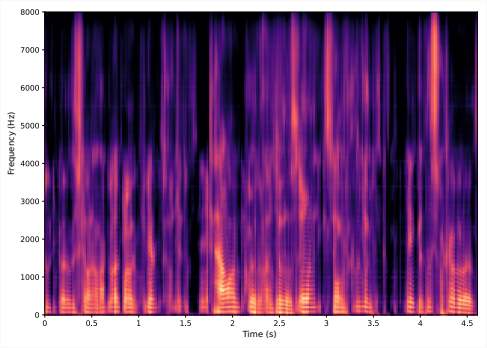
- Training two VAEs via SGD.
- Mixing (AV fusion) via two speech models.
- Enhancement via VEM + sampling.

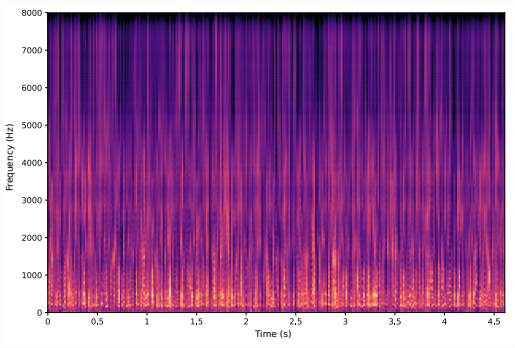


MIN-VAE:

- Training all 3 networks via VEM+SGD.
- Mixing (AV fusion) via a single speech model.
- Enhancement via VEM + sampling.

Strong limitation of VAEs: frame modeled independently.





Frames are modeled independently, we need time/sequential modeling!

▶ **Dynamical VAE (DVAE[6]) – "VAE for sequential modeling"**

(family of methods including existing literature)

Xiaoyu Bie

Laurent Girin

Simon Leglaive

Thomas Hueber

Julien Diard

[6] Girin, L., *et. al.*, (2021), FnT Machine Learning – `Warning` $\sim$ `150 pages!!`

Probabilistic Sequential Modeling (decoder network)

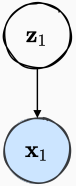
We would like to model sequences of observations ($\boldsymbol{x}_{1:T}$) and latent variables ($\boldsymbol{z}_{1:T}$):

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

Probabilistic Sequential Modeling (decoder network)

We would like to model sequences of observations ($\boldsymbol{x}_{1:T}$) and latent variables ($\boldsymbol{z}_{1:T}$):

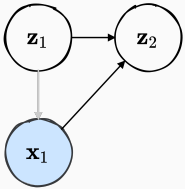
$$p_{\boldsymbol{\theta}}^{\mathrm{DVAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) \neq \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{z}_t) = p_{\boldsymbol{\theta}}^{\mathrm{VAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}).$$



Probabilistic Sequential Modeling (decoder network)

We would like to model sequences of observations ($\boldsymbol{x}_{1:T}$) and latent variables ($\boldsymbol{z}_{1:T}$):

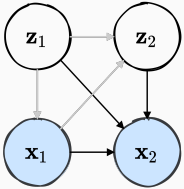
$$p_{\boldsymbol{\theta}}^{\mathrm{DVAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) \neq \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{z}_t) = p_{\boldsymbol{\theta}}^{\mathrm{VAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}).$$



Probabilistic Sequential Modeling (decoder network)

We would like to model sequences of observations ($\boldsymbol{x}_{1:T}$) and latent variables ($\boldsymbol{z}_{1:T}$):

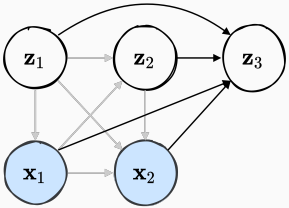
$$p_{\boldsymbol{\theta}}^{\mathrm{DVAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) \neq \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{z}_t) = p_{\boldsymbol{\theta}}^{\mathrm{VAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}).$$



Probabilistic Sequential Modeling (decoder network)

We would like to model sequences of observations ($\boldsymbol{x}_{1:T}$) and latent variables ($\boldsymbol{z}_{1:T}$):

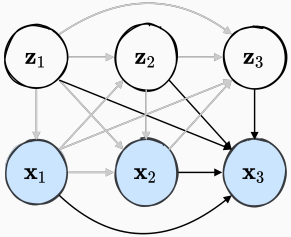
$$p_{\boldsymbol{\theta}}^{\mathrm{DVAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) \neq \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{z}_t) = p_{\boldsymbol{\theta}}^{\mathrm{VAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}).$$



Probabilistic Sequential Modeling (decoder network)

We would like to model sequences of observations ($\boldsymbol{x}_{1:T}$) and latent variables ($\boldsymbol{z}_{1:T}$):

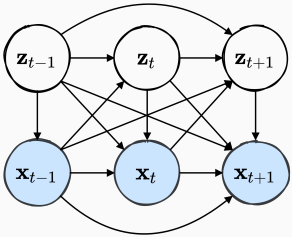
$$p_{\boldsymbol{\theta}}^{\mathrm{DVAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) \neq \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{z}_t) = p_{\boldsymbol{\theta}}^{\mathrm{VAE}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}).$$



Probabilistic Sequential Modeling (decoder network)

We would like to model sequences of observations ($\boldsymbol{x}_{1:T}$) and latent variables ($\boldsymbol{z}_{1:T}$):

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



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

The *prior* distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t+1}|\boldsymbol{z}_{1:t}, \boldsymbol{x}_{1:t})$ is now conditional, parametric, and might be auto-regressive (AR).

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

Probabilistic Sequential Inference (encoder network)

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

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

[7] Bishop, C. M., (2006).

Probabilistic Sequential Inference (encoder 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. Use D-separation[7] to find out the true dependencies, then choose whether to keep them (e.g. to ensure causality).

[7] Bishop, C. M., (2006).

Implementation (both decoder and encoder)

Let's take the general DVAE decoder:

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

[8] Several DVAEs @ `https://github.com/XiaoyuBIE1994/DVAE-speech`

Implementation (both decoder and encoder)

Let's take the general DVAE decoder:

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

The generative distributions can be implemented with an RNN:

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

There are many possible implementations[8] for the same probabilistic dependencies!!!

[8] Several DVAEs @ `https://github.com/XiaoyuBIE1994/DVAE-speech`

Learning: ELBO

The objective is build in the same way as VAEs, but looks different:

$$\mathcal{L}(\boldsymbol{x}_{1:T};\boldsymbol{\phi},\boldsymbol{\theta}) \stackrel{?}{=} \sum_{t=1}^{T} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:T})} \underbrace{\ln p_{\boldsymbol{\theta}_{\boldsymbol{x}}}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1},\boldsymbol{z}_{1:t})}_{\text{Reconstruction}}$$
$$- \sum_{t=1}^{T} \underbrace{D_{\mathsf{KL}}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1},\boldsymbol{x}_{1:T}) \,\Big\|\, p_{\boldsymbol{\theta}_{\boldsymbol{z}}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1},\boldsymbol{z}_{1:t-1})\Big)}_{\text{Regularization}}$$

- The reconstruction and regularisation terms are evaluated at every frame t.

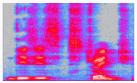
Learning: ELBO

The objective is build in the same way as VAEs, but looks different:

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

- 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!

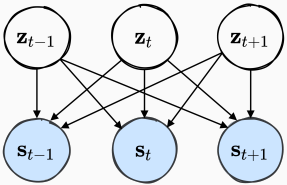
Application to Unsupervised Probabilistic SE (revisit)



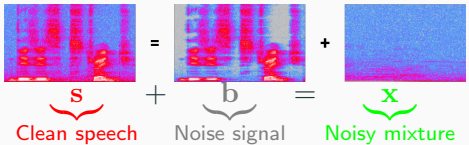
$\underbrace{\mathbf{s}}_{\text{Clean speech}} + \underbrace{\mathbf{b}}_{\text{Noise signal}} = \underbrace{\mathbf{x}}_{\text{Noisy mixture}}$

Train – Model $\boldsymbol{\theta}$ for clean data: $\{\boldsymbol{s}_{1:T}\}$.

$$p_{\boldsymbol{\theta}}(\boldsymbol{s}_{1:T}) \approx p_{\text{data}}(\boldsymbol{s}_{1:T})$$

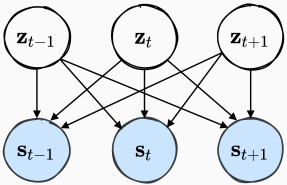


Application to Unsupervised Probabilistic SE (revisit)

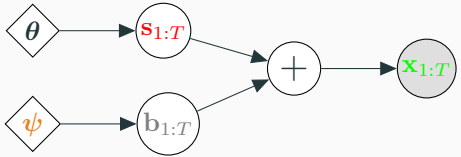


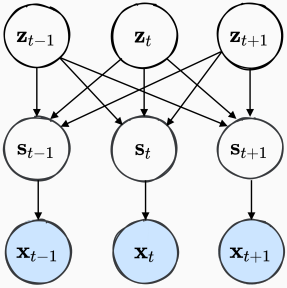
Train – Model $\boldsymbol{\theta}$ for clean data: $\{\boldsymbol{s}_{1:T}\}$.

$$p_{\boldsymbol{\theta}}(\boldsymbol{s}_{1:T}) \approx p_{\text{data}}(\boldsymbol{s}_{1:T})$$



Test – learn the noise parameters from noisy samples $\mathbf{x}$ and estimate the clean speech $\hat{\boldsymbol{s}}_{1:T}$:





Results on the Wall Street Journal (WSJ) and VoiceBank (VB)[9]

Method	Superv.	Test	Train	SI-SDR (dB)	Train	SI-SDR (dB)
Noisy mixture	-	WSJ+QUT	-	−2.6	-	−2.6
VAE-VEM (ICASSP'20)	None		Same	5.0		
RVAE-VEM (Proposed)	None			**5.8**		
MetricGAN-U (ICASSP'22)	Partial			N/A		
UMX* 2020	Full			<u>5.7</u>		
MetricGAN+* (Interspeech'21)	Full			3.6		

Noisy mixture	-
VAE-VEM (ICASSP'20)	None
RVAE-VEM (Proposed)	None
NyTT EUSIPCO'21	Partial/Xtra
MetricGAN-U (half) ICASSP'22	Partial
UMX 2020	Full
MetricGAN+ Interspeech'21	Full

Results in dB: **best** and <u>second best</u> per section.

[9] Bie, X., *et. al.*, (2022), IEEE TASLP.

Results on the Wall Street Journal (WSJ) and VoiceBank (VB)[9]

Method	Superv.	Test	Train	SI-SDR (dB)	Train	SI-SDR (dB)
Noisy mixture	-	WSJ+QUT	-	−2.6	-	−2.6
VAE-VEM (ICASSP'20)	None		Same	5.0		
RVAE-VEM (Proposed)	None			**5.8**		
MetricGAN-U (ICASSP'22)	Partial			N/A		
UMX* 2020	Full			<u>5.7</u>		
MetricGAN+* (Interspeech'21)	Full			3.6		
Noisy mixture	-	VB-DMD	-	8.4	-	8.4
VAE-VEM (ICASSP'20)	None		Same	16.4		
RVAE-VEM (Proposed)	None			<u>17.1</u>		
NyTT EUSIPCO'21	Partial/Xtra			**17.7**		
MetricGAN-U (half) ICASSP'22	Partial			8.2		
UMX 2020	Full			14.0		
MetricGAN+ Interspeech'21	Full			8.5		

Results in dB: **best** and <u>second best</u> per section.

[9] Bie, X., *et. al.*, (2022), IEEE TASLP.

Results on the Wall Street Journal (WSJ) and VoiceBank (VB)[9]

Method	Superv.	Test	Train	SI-SDR (dB)	Train	SI-SDR (dB)
Noisy mixture	-	WSJ+QUT	-	−2.6	-	−2.6
VAE-VEM (ICASSP'20)	None		Same	5.0	Different	3.8
RVAE-VEM (Proposed)	None			**5.8**		**4.3**
MetricGAN-U (ICASSP'22)	Partial			N/A		-1.6
UMX* 2020	Full			<u>5.7</u>		<u>4.1</u>
MetricGAN+* (Interspeech'21)	Full			3.6		1.8
Noisy mixture	-	VB-DMD	-	8.4	-	8.4
VAE-VEM (ICASSP'20)	None		Same	16.4		
RVAE-VEM (Proposed)	None			<u>17.1</u>		
NyTT EUSIPCO'21	Partial/Xtra			**17.7**		
MetricGAN-U (half) ICASSP'22	Partial			8.2		
UMX 2020	Full			14.0		
MetricGAN+ Interspeech'21	Full			8.5		

Results in dB: **best** and <u>second best</u> per section.

[9] Bie, X., *et. al.*, (2022), IEEE TASLP.

Results on the Wall Street Journal (WSJ) and VoiceBank (VB)[9]

Method	Superv.	Test	Train	SI-SDR (dB)	Train	SI-SDR (dB)
Noisy mixture	-	WSJ+QUT	-	−2.6	-	−2.6
VAE-VEM (ICASSP'20)	None		Same	5.0	Different	3.8
RVAE-VEM (Proposed)	None			**5.8**		**4.3**
MetricGAN-U (ICASSP'22)	Partial			N/A		-1.6
UMX* 2020	Full			<u>5.7</u>		<u>4.1</u>
MetricGAN+* (Interspeech'21)	Full			3.6		1.8
Noisy mixture	-	VB-DMD	-	8.4	-	8.4
VAE-VEM (ICASSP'20)	None		Same	16.4	Different	<u>15.0</u>
RVAE-VEM (Proposed)	None			<u>17.1</u>		**17.3**
NyTT EUSIPCO'21	Partial/Xtra			**17.7**		N/A
MetricGAN-U (half) ICASSP'22	Partial			8.2		N/A
UMX 2020	Full			14.0		10.4
MetricGAN+ Interspeech'21	Full			8.5		3.9

Results in dB: **best** and <u>second best</u> per section.

[9] Bie, X., *et. al.*, (2022), IEEE TASLP.

DVAEs are good (great!) at temporal modeling!

All implementations use RNN (or variants).

- **What about transformers?**


Xiaoyu Bie


Wen Guo

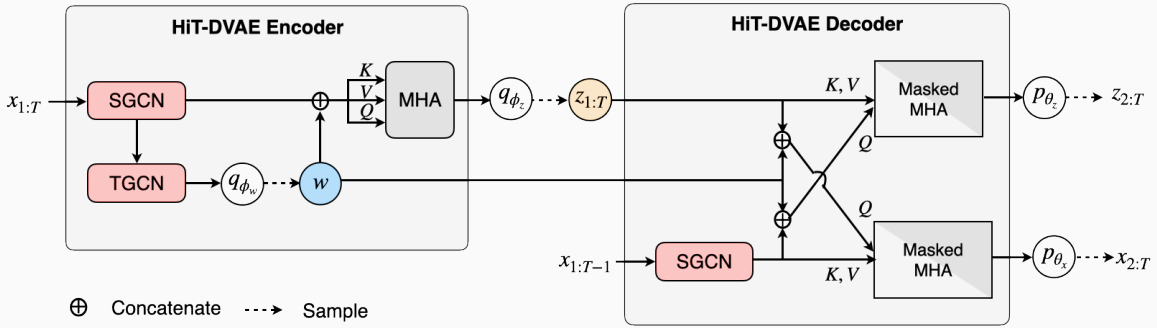

Simon Leglaive


Francesc Moreno


Laurent Girin

Introducing HIT-DVAE[10]

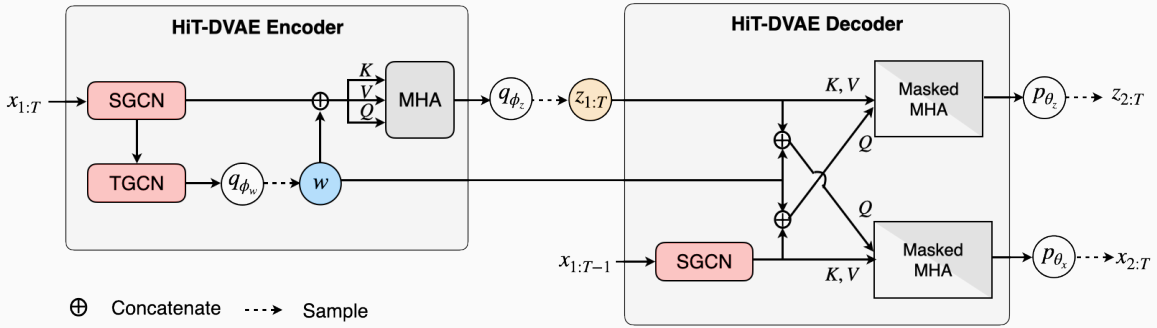
Hierarchical Transformer DVAE – two levels of latent variables: static w and dynamic $z_{1:T}$.



[10] Bie, X., *et. al.*, (2023), Pre-print.

Introducing HIT-DVAE[10]

Hierarchical Transformer DVAE – two levels of latent variables: static w and dynamic $z_{1:T}$.

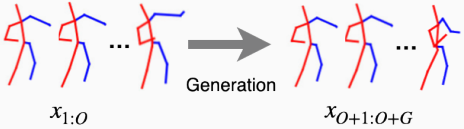


There is something odd in the way Q, K, V are used...

[10] Bie, X., *et. al.*, (2023), Pre-print.

HIT-DVAE: Modified transformer architecture

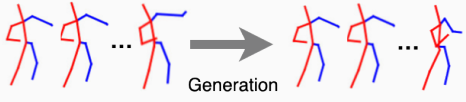
Task: human motion modeling/generation. Predicting $\boldsymbol{x}_{O+1}$ from $\boldsymbol{x}_{1:O}$ is very easy.[11]



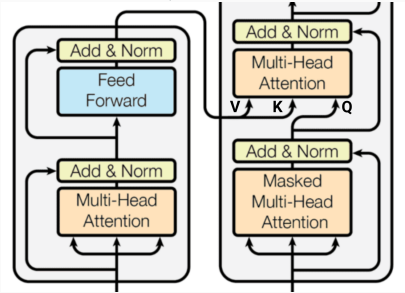
[11] Guo., W., *et. al.*, (2021), IEEE WACV.

HIT-DVAE: Modified transformer architecture

Task: human motion modeling/generation. Predicting $\boldsymbol{x}_{O+1}$ from $\boldsymbol{x}_{1:O}$ is very easy.[11]



Standard query/residual connection:



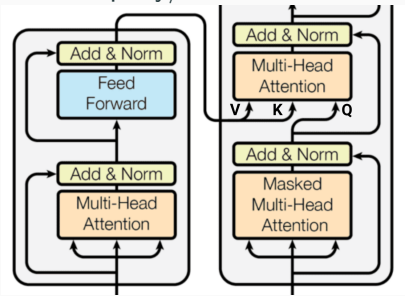
[11] Guo., W., *et. al.*, (2021), IEEE WACV.

HIT-DVAE: Modified transformer architecture

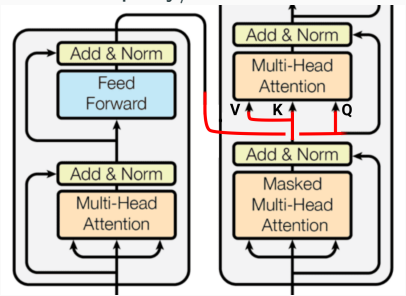
Task: human motion modeling/generation. Predicting $\boldsymbol{x}_{O+1}$ from $\boldsymbol{x}_{1:O}$ is very easy.[11]

$x_{1:O}$ $x_{O+1:O+G}$

Standard query/residual connection:

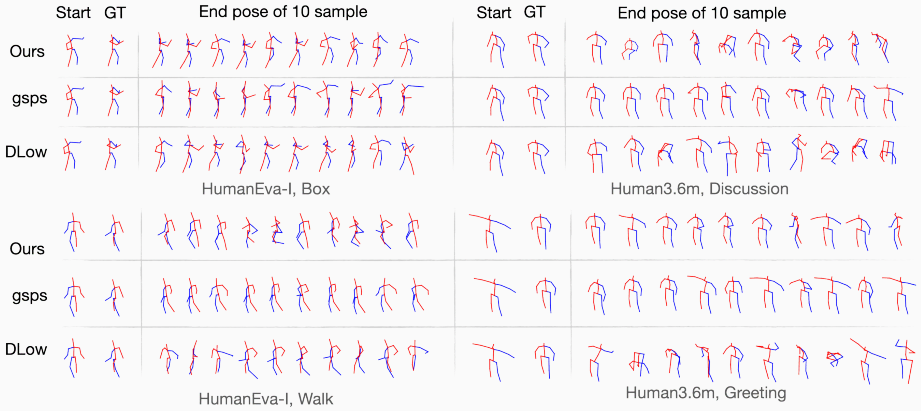


HIT-DVAE query/residual connection:



[11] Guo., W., *et. al.*, (2021), IEEE WACV.

HIT-DVAE: Results



Good trade-off between diversity, quality and accuracy. Also useful for audio modeling.[12]

[12] Lin, X., *et. al.*, (2023), IEEE ICASSP.

DVAEs can model mono-modal sequences.

- **What about multiple modalities?**

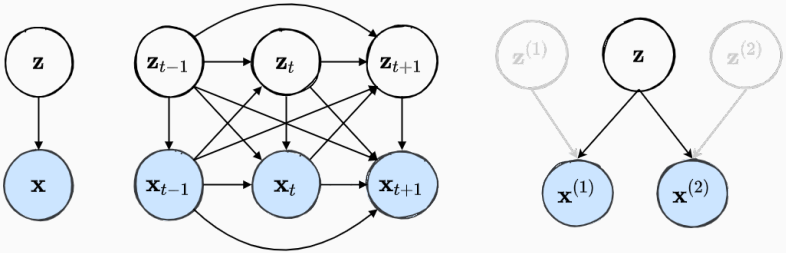
Samir Sadok

Simon Leglaive

Renaud Séguier

Laurent Girin

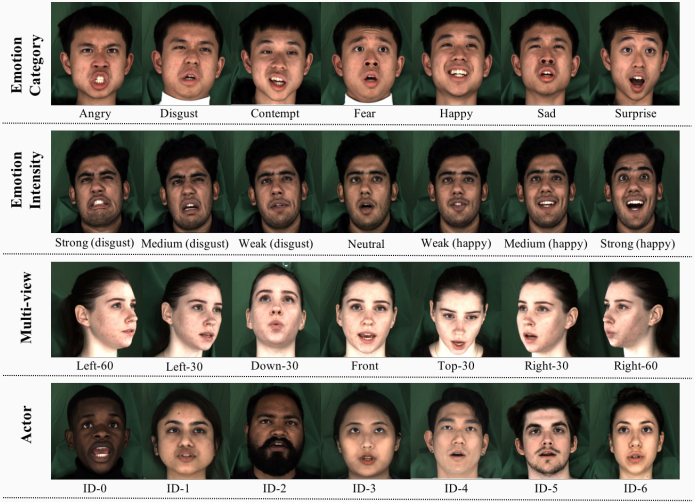
In other words:



VAEs (left) can be multi-modal[13] (right). Can DVAEs (middle) be multi-modal too?

[13] Sutter, T. M., *et. al.*, (2021), ICLR.

Task: emotional audio-visual speech modeling.[14]

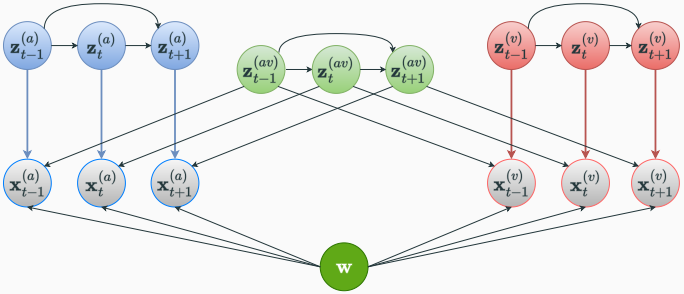


What should we model?

- Static AV (ID, emotion)
- Dynamic AV (lip-audio corr.)
- Dynamic A (other audio features)
- Dynamic V (eye AUs).

[14] Wang, K., *et. al.*, (2020), ECCV.

Introducing MDVAEs[15]

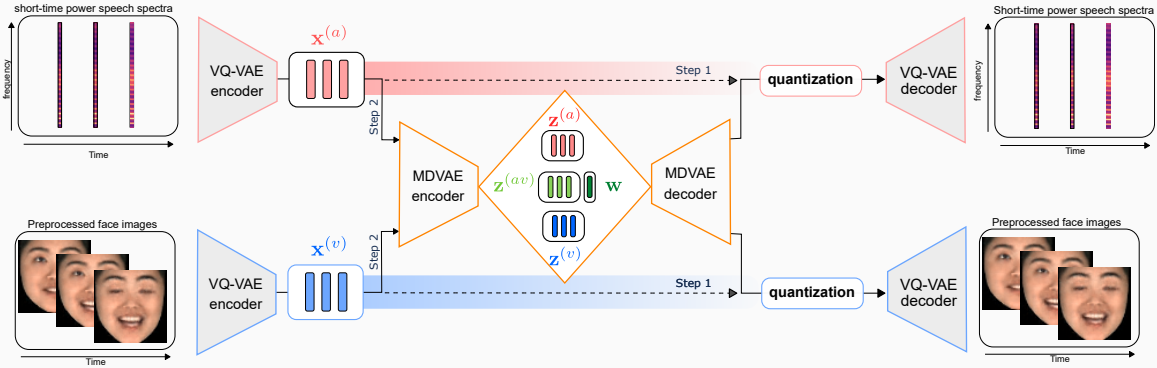


What should we model?

- Static AV – $\boldsymbol{w}$ (ID, emotion)
- Dynamic AV – $\boldsymbol{z}^{av}$ (lip-audio corr.)
- Dynamic A – $\boldsymbol{z}^{a}$ (other audio features)
- Dynamic V – $\boldsymbol{z}^{v}$ (eye AUs).

[15] Sadok, S., *et. al.*, (2024), Neural Networks.

The VQ-MDVAE Architecture



(i) Quantize auditory and visual features. (ii) Use MDVAE to model the quantized features.

MDVAE: Some fun things

Let's see a couple of videos on:

- latent variable transfer,
- latent variable interpolation.

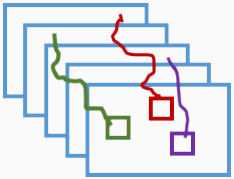
Earlier, we combined (static) VAEs with mixing latents.

- **Is it possible/useful with DVAEs?**

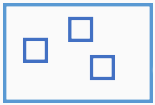
Xiaoyu Lin

Laurent Girin

Motivating application: unsupervised multiple object tracking



Tracklets @ $t-1$

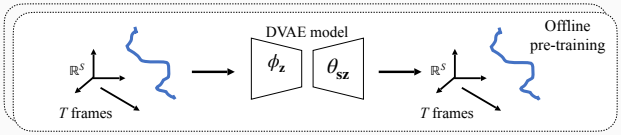


Detections @ t

Desired result

Question: Can we model the non-linear dynamics with a DVAE, and have an assignment mechanism within the same ML formulation?

Introducing MixDVAE[16]

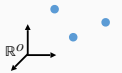


[16]Lin, X., *et. al.*, (2023), Under review.

Introducing MixDVAE[16]

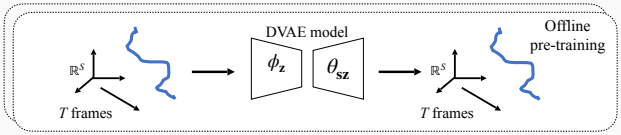
Observations at frame t

$\mathbf{o}_{t,1:K_t}$



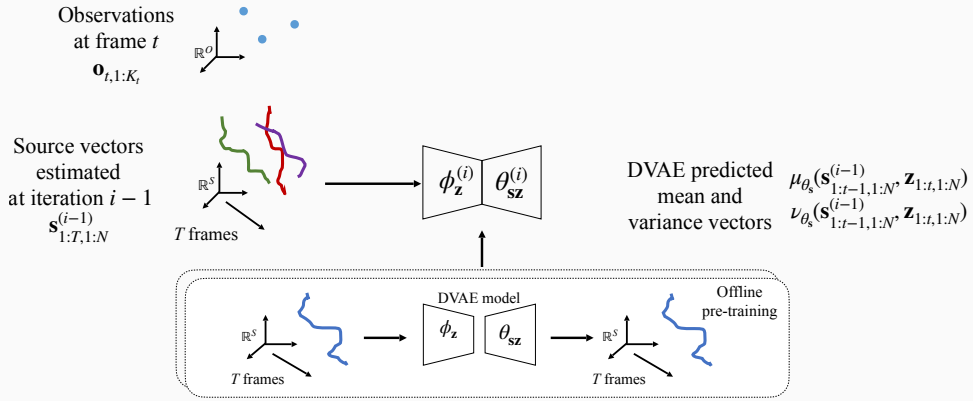
Source vectors estimated at iteration $i-1$

$\mathbf{s}^{(i-1)}_{1:T,1:N}$



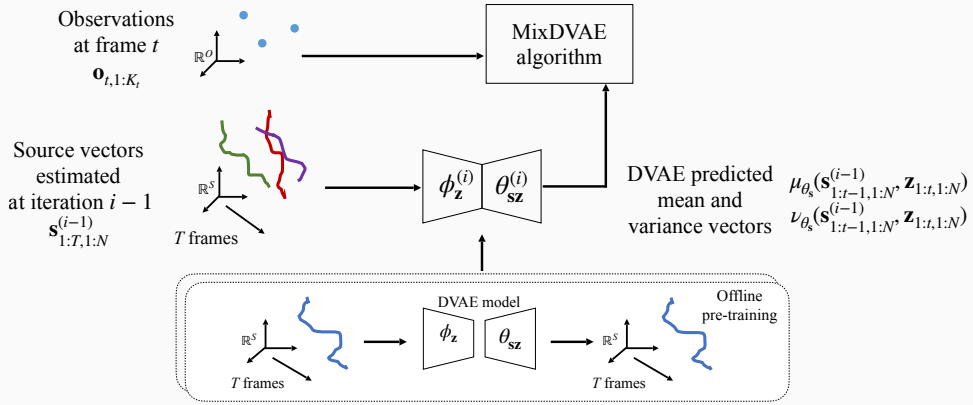
[16] Lin, X., *et. al.*, (2023), Under review.

Introducing MixDVAE[16]



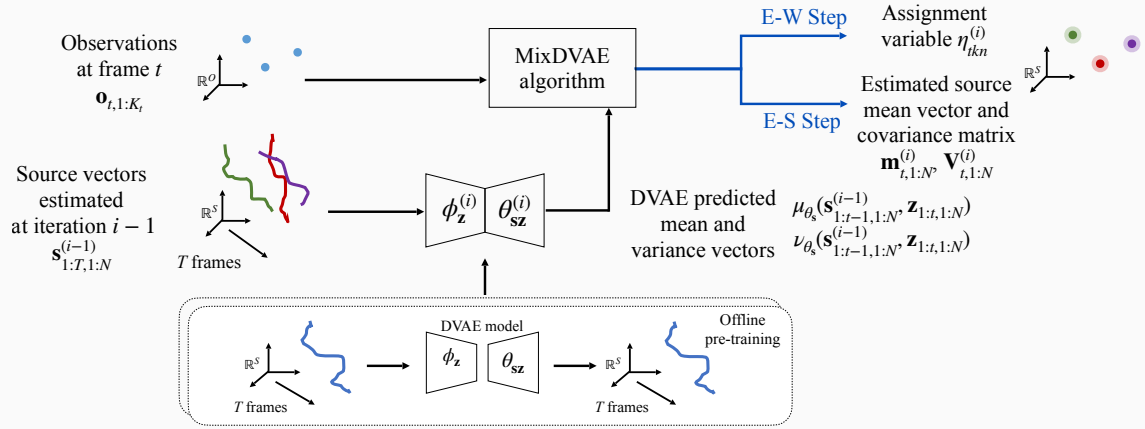
[16] Lin, X., *et. al.*, (2023), Under review.

Introducing MixDVAE[16]



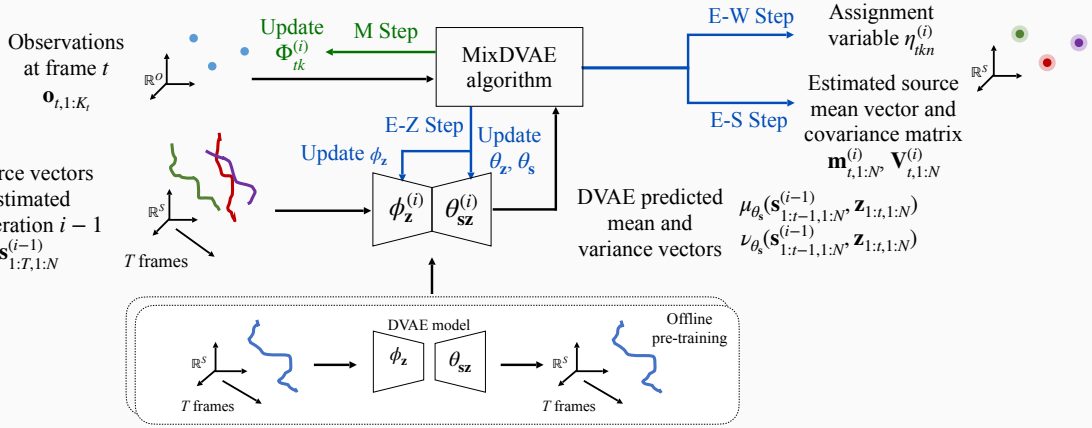
[16] Lin, X., *et. al.*, (2023), Under review.

Introducing MixDVAE[16]



[16] Lin, X., *et. al.*, (2023), Under review.

Introducing MixDVAE[16]



[16] Lin, X., *et. al.*, (2023), Under review.

Quick discussion & results

GMM update (K observations):

$$\boldsymbol{\mu}_n = \sum_k \underbrace{\eta_{kn}}_{\text{assign.}} \boldsymbol{o}_k$$

Quick discussion & results

GMM update (K observations):

$$\boldsymbol{\mu}_n = \sum_k \underbrace{\eta_{kn}}_{\text{assign.}} \boldsymbol{o}_k$$

Kalman update (sequence of T obs):

$$\boldsymbol{s}_t = \underbrace{\boldsymbol{P}_t \boldsymbol{o}_t}_{\text{update}} + \underbrace{\boldsymbol{T}_t \boldsymbol{s}_{t-1}}_{\text{prediction}}$$

Quick discussion & results

GMM update (K observations):

$$\boldsymbol{\mu}_n = \sum_k \underbrace{\eta_{kn}}_{\text{assign.}} \boldsymbol{o}_k$$

Kalman update (sequence of T obs):

$$\boldsymbol{s}_t = \underbrace{\boldsymbol{P}_t \boldsymbol{o}_t}_{\text{update}} + \underbrace{\boldsymbol{T}_t \boldsymbol{s}_{t-1}}_{\text{prediction}}$$

MixDVAE update is a combination:

$$\boldsymbol{s}_{tn} = \underbrace{\boldsymbol{P}_t \sum_k \eta_{kn} \boldsymbol{o}_{tk}}_{\text{assig. \& update}} + \underbrace{\boldsymbol{T}_t(\boldsymbol{s}_{1:t-1})}_{\text{non-lin. prediction}}$$

Quick discussion & results

GMM update (K observations):

$$\boldsymbol{\mu}_n = \sum_k \underbrace{\eta_{kn}}_{\text{assign.}} \boldsymbol{o}_k$$

Kalman update (sequence of T obs):

$$\boldsymbol{s}_t = \underbrace{\boldsymbol{P}_t \boldsymbol{o}_t}_{\text{update}} + \underbrace{\boldsymbol{T}_t \boldsymbol{s}_{t-1}}_{\text{prediction}}$$

MixDVAE update is a combination:

$$\boldsymbol{s}_{tn} = \underbrace{\boldsymbol{P}_t \sum_k \eta_{kn} \boldsymbol{o}_{tk}}_{\text{assig. \& update}} + \underbrace{\boldsymbol{T}_t(\boldsymbol{s}_{1:t-1})}_{\text{non-lin. prediction}}$$

- Results in unsupervised MOT and in semi-blind source separation.
- Work in progress: fine-tuning, complexity, learning from noise, ...

VAE/DVAE Summary

	Mono-modal	Multi-modal	Mixtures
Static	VAE [Kingma'14] VQ-VAE [van den Oord'17]		
Dynamic			

VAE/DVAE Summary

	Mono-modal	Multi-modal	Mixtures
Static	VAE [Kingma'14] VQ-VAE [van den Oord'17]	CVAE [Sadeghi'20] MVAE [Sutter'21]	
Dynamic			

VAE/DVAE Summary

	Mono-modal	Multi-modal	Mixtures
Static	VAE [Kingma'14] VQ-VAE [van den Oord'17]	CVAE [Sadeghi'20] MVAE [Sutter'21]	VAE-MM [Sadeghi'20] MIN-VAE [Sadeghi'21]
Dynamic			

VAE/DVAE Summary

	Mono-modal	Multi-modal	Mixtures
Static	VAE [Kingma'14] VQ-VAE [van den Oord'17]	CVAE [Sadeghi'20] MVAE [Sutter'21]	VAE-MM [Sadeghi'20] MIN-VAE [Sadeghi'21]
Dynamic	DVAE [Girin'21] Sw-VAE [Sadeghi'21]		

VAE/DVAE Summary

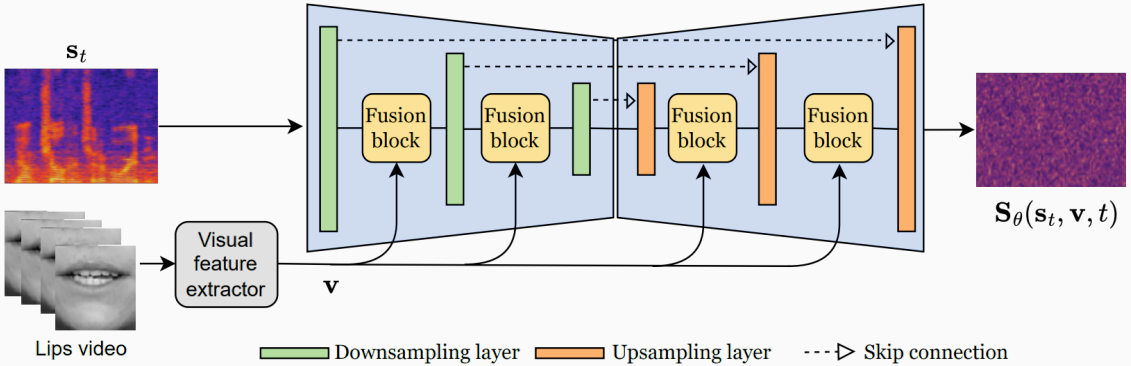
	Mono-modal	Multi-modal	Mixtures
Static	VAE [Kingma'14] VQ-VAE [van den Oord'17]	CVAE [Sadeghi'20] MVAE [Sutter'21]	VAE-MM [Sadeghi'20] MIN-VAE [Sadeghi'21]
Dynamic	DVAE [Girin'21] Sw-VAE [Sadeghi'21]	VQ-MDVAE [Sadok'23]	

VAE/DVAE Summary

	Mono-modal	Multi-modal	Mixtures
Static	VAE [Kingma'14] VQ-VAE [van den Oord'17]	CVAE [Sadeghi'20] MVAE [Sutter'21]	VAE-MM [Sadeghi'20] MIN-VAE [Sadeghi'21]
Dynamic	DVAE [Girin'21] Sw-VAE [Sadeghi'21]	VQ-MDVAE [Sadok'23]	MixDVAE [Lin'23]

Limited to VAE? NO!

We started investigated unsupervised AVSE with diffusion models:



Still need an EM algorithm at test time...

What else is there to be done?

Plenty!!!

- Systematic AV for diffusion works.
- Can we design mixtures of diffusion models?
- What about latent diffusion?
- Can we have mixtures combined with latent diffusion?
- What about temporal diffusion models? And their mixtures?
- And beyond diffusion? e.g. flow matching?

Summary & conclusions

Take-home messages

- Latent variable modlings has endless possiblities.
- Auto-grad cannot save you.
- Analytic results can bring interpretability.
- Many possibilities are yet to be explored.
- Uncommon skills/knowledge (e.g. VI) can make your profile unique.

Thanks...

for bearing with me!

to my colleagues & collaborators!

for your challenging questions & interesting discussion.

We are also interested in meta-learning, reinforcement learning, domain adaptation, etc.

I am around until Thursday!