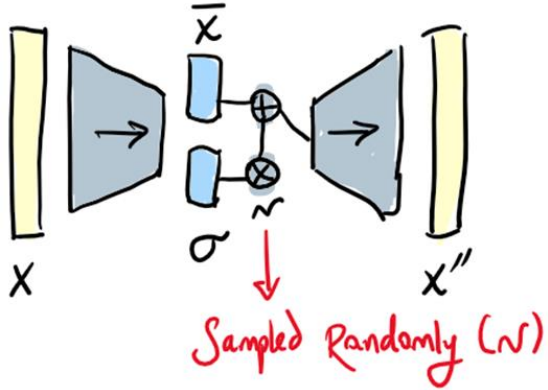
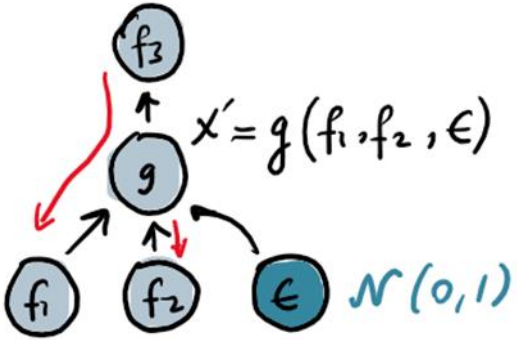
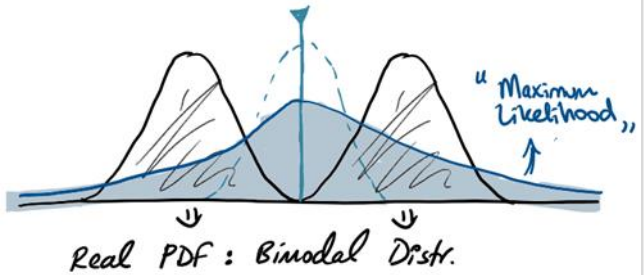
Data Driven Engineering I: Machine Learning for Dynamical Systems

Introduction to Generative Learning: VAEs and GANs

Institute of Thermal Turbomachinery
Prof. Dr.-Ing. Hans-Jörg Bauer

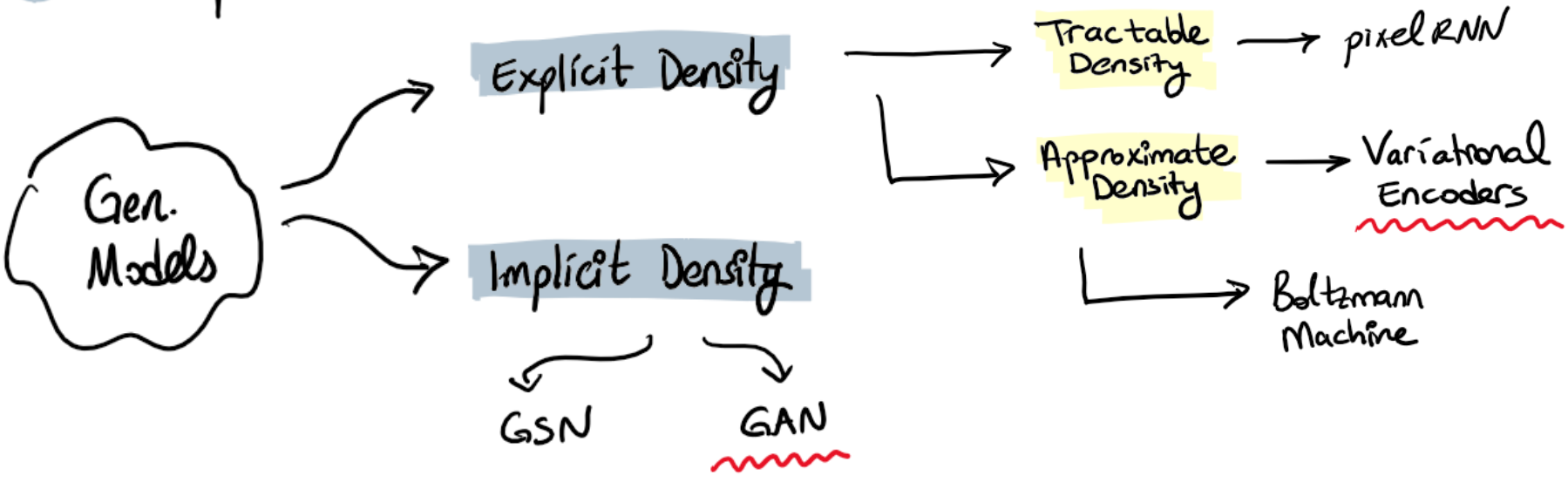






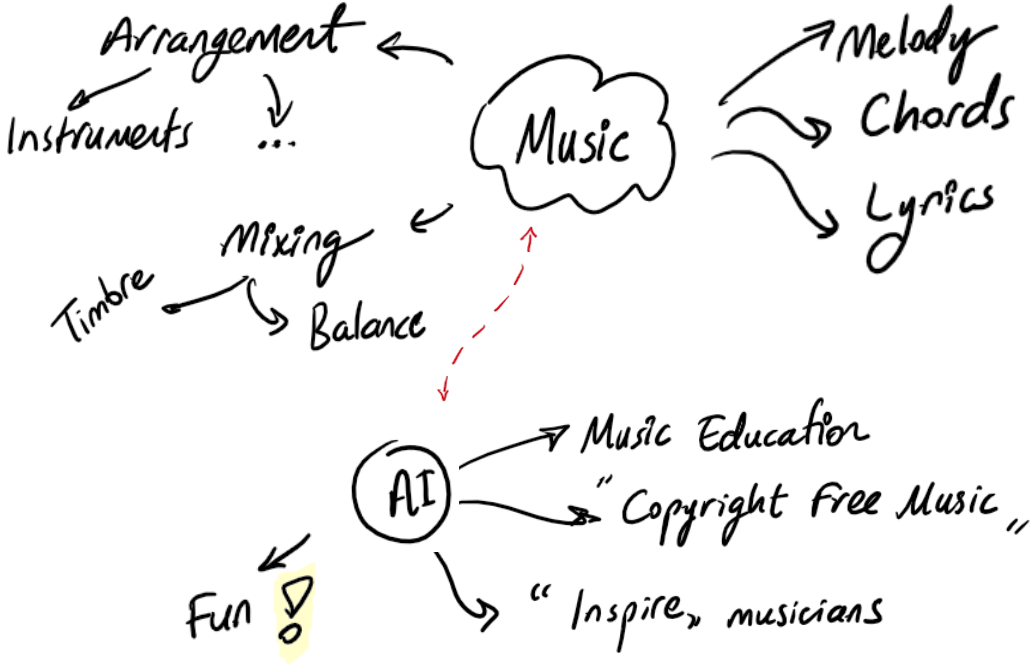
UL → Generative Models :

* It is probabilistic in nature



Case Study: Composing Music with Gen. Methods

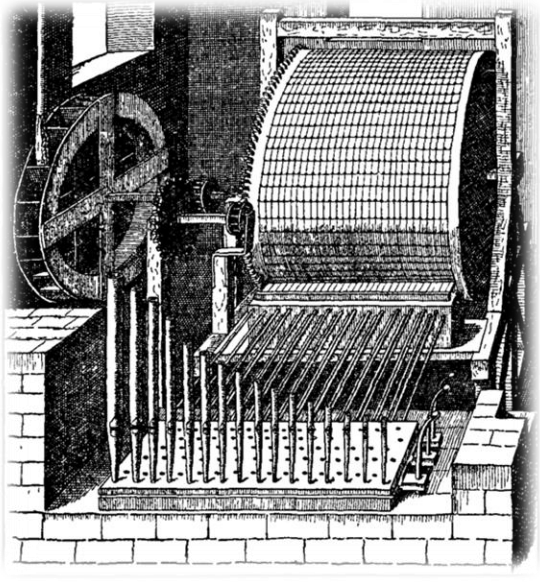
Bach
↓
Solo Violin



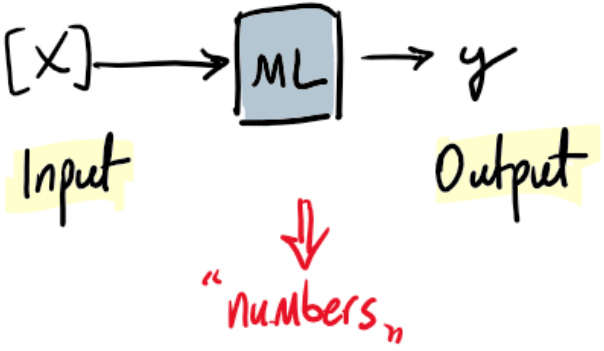
Industry:

- Google Magenta
- Spotify
- amper
- IBM
- Sony
- ...

Modeling Music



Our Models



? Representation of music

? Conversion of music

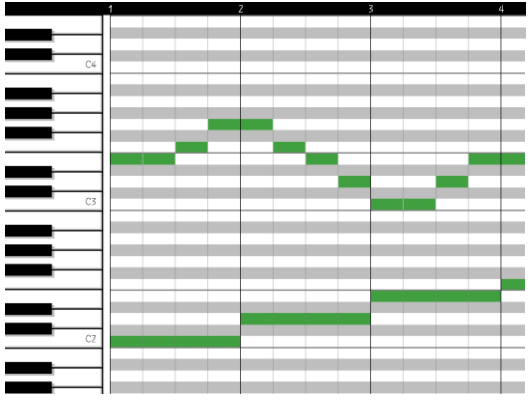
I/O in Music

① Symbolic Representation

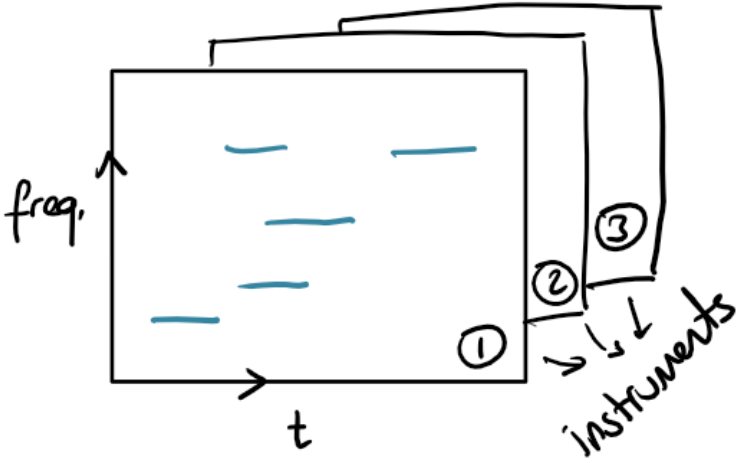
* Piano Rolls
  ↓
  "image like"

Eg:

* MuseGAN
* MidiNET



② Audio



I/O in Music

① Symbolic Representation

② Audio

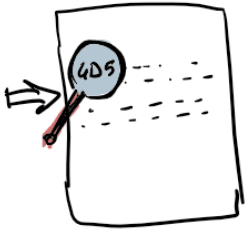
* MIDI events

↓

"text"

Musical
Instrument
Digital
Interface

" standard format "



music21

```
{0.0} <music21.instrument.Violin Violin>
{0.0} <music21.tempo.MetronomeMark Quarter=250.0>
{0.0} <music21.key.Key of D major>
{0.0} <music21.meter.TimeSignature 4/4>
{0.0} <music21.note.Rest rest>
{3.75} <music21.tempo.MetronomeMark largamente Qu
{3.75} <music21.chord.Chord F#5>
{4.0} <music21.chord.Chord B3 F#4 D5 F#5>
{4.75} <music21.chord.Chord B5>
{5.0} <music21.chord.Chord E5 G5>
```

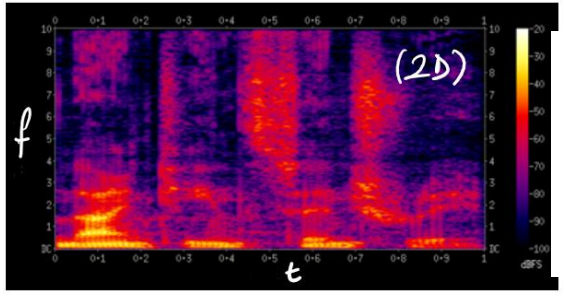
I/O in Music

① Symbolic Representation

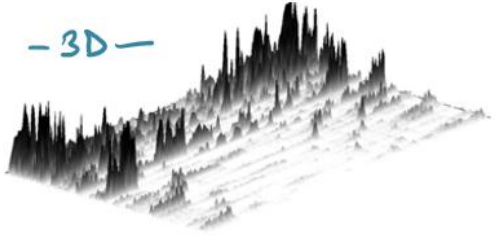
* Piano Rolls → "image like"
* MIDI events → "text"
* Score → <hybrid>

② Audio

* Spectrogram → "image-like"



* ~Wave form



Understanding the Data

① ~metadata

```
{0.0} <music21.instrument.Violin Violin>
{0.0} <music21.tempo.MetronomeMark Quarter=250.0>
{0.0} <music21.key.Key of D major>
{0.0} <music21.meter.TimeSignature 4/4>
{0.0} <music21.note.Rest rest>
{3.75} <music21.tempo.MetronomeMark largamente Quarter=32.0>
{3.75} <music21.chord.Chord F#5>
{4.0} <music21.chord.Chord B3 F#4 D5 F#5>
{4.75} <music21.chord.Chord B5>
{5.0} <music21.chord.Chord E5 G5>
```

- Starts @ beat 4
- Duration ⇒ 0.75
- Chords ⇒ (B, D, F)

Vast Music Space

② [X] := MIDI ⇒ Text ⇒ Numbers

? Multiple Notes

⇨ MIDI: 128 Single notes

↳ Combinations ⇒ [E5 G5]

Eg: Classes: A, B, C ⇨ {A}, {B}, {C}

↳

example	f_A	f_B	f_C
{A}	1	0	0
{C}	0	0	1
{B}	0	1	0

By TF

Vast Music Space

⇨ MIDI: 128 Single notes

↳ Combinations ⇒ [E5 G5]

Eg. {1 note, 2 notes}

```
{0.0} <music21.instrument.Violin Violin>
{0.0} <music21.tempo.MetronomeMark Quarter=250.0>
{0.0} <music21.key.Key of D major>
{0.0} <music21.meter.TimeSignature 4/4>
{0.0} <music21.note.Rest rest>
{3.75} <music21.tempo.MetronomeMark largamente Quarter=32.0>
{3.75} <music21.chord.Chord F#5>
{4.0} <music21.chord.Chord B3 F#4 D5 F#5>
{4.75} <music21.chord.Chord B5>
{5.0} <music21.chord.Chord E5 G5>
```

↳ $\frac{128!}{(128-2)!\,2!} = \frac{128 \times 127}{2} = 8128$ dimensions !

Solution: Notes = 128 ⟹ Reduce the number; (filter to C Major key)

③ One-hot encoding:

- text ⟺ number ; reversible
- Use dictionaries ~ look-up tables

```
{0.0} <music21.instrument.Violin Violin>
{0.0} <music21.tempo.MetronomeMark Quarter=250.0>
{0.0} <music21.key.Key of D major>
{0.0} <music21.meter.TimeSignature 4/4>
{0.0} <music21.note.Rest rest>
{3.75} <music21.tempo.MetronomeMark largamente Quarter=32.0>
{3.75} <music21.chord.Chord F#5>
{4.0} <music21.chord.Chord B3 F#4 D5 F#5>
{4.75} <music21.chord.Chord B5>
{5.0} <music21.chord.Chord E5 G5>
```

(i) (Chord list) → get unique chords ⇒ Give an integer to each unique case

(ii) (Duration list) → get unique durations ⇒ Give an integer to each unique case

(iii) Create reverse dict. as well.

```
{'A3': 0,
 'A3.A4': 1,
 'A3.A4.C#5': 2,
 'A3.A4.C5': 3,
 'A3.A4.C5.A5': 4,
 'A3.A4.C5.B5': 5,
 'A3.A4.C5.C6': 6,
 'A3.A4.C5.E5': 7,
 'A3.A4.C5.F#5': 8,
 'A3.A4.C5.F5': 9,
```

```
{0: 'A3',
 1: 'A3.A4',
 2: 'A3.A4.C#5',
 3: 'A3.A4.C5',
 4: 'A3.A4.C5.A5',
 5: 'A3.A4.C5.B5',
 6: 'A3.A4.C5.C6',
```

④ Creating [X] from Sequential Data

* Chord = [4, 11, 46, 122, 5, 7, ..]
  Duration = [2, 2, 1, 1, 4, 3, ...] ⇒ "Sliding windows"

* Sequence size = 32 ⇒ C_0 [0 → 31] D_0 [0 → 31]
  C_1 [1 → 32] D_1 [1 → 32]
  ....

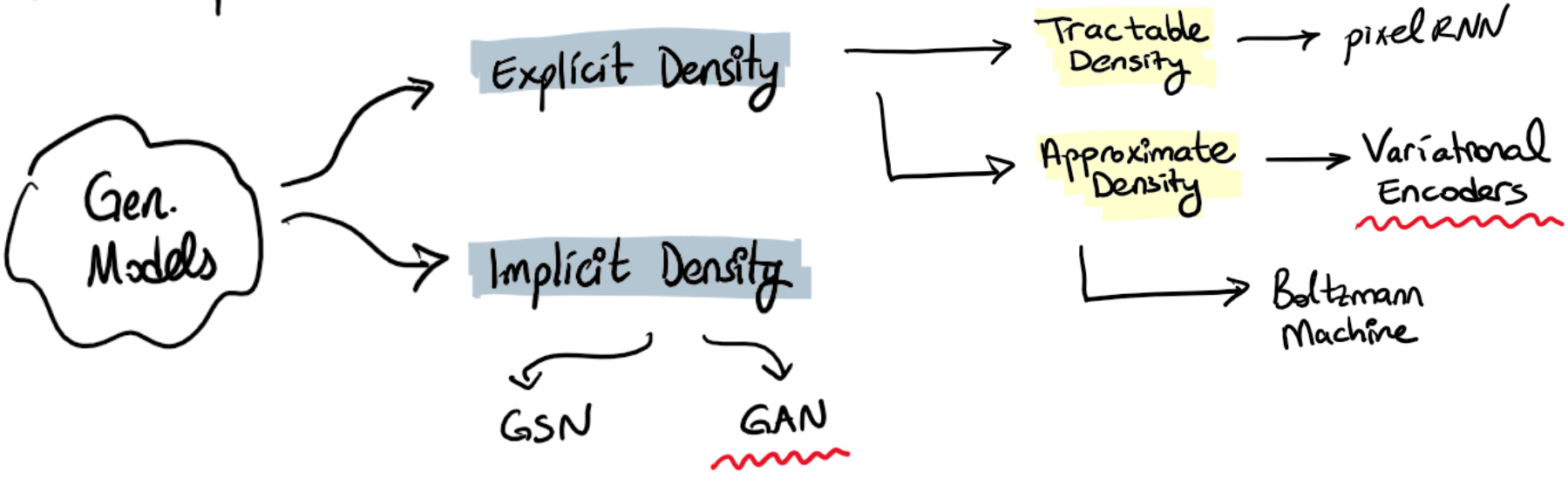
```
{'A3': 0,
 'A3.A4': 1,
 'A3.A4.C#5': 2,
 'A3.A4.C5': 3,
 'A3.A4.C5.A5': 4,
 'A3.A4.C5.B5': 5,
 'A3.A4.C5.C6': 6,
 'A3.A4.C5.E5': 7,
 'A3.A4.C5.F#5': 8,
 'A3.A4.C5.F5': 9,
```

Ateliers & Saveurs in Montreal

colab

UL → Generative Models :

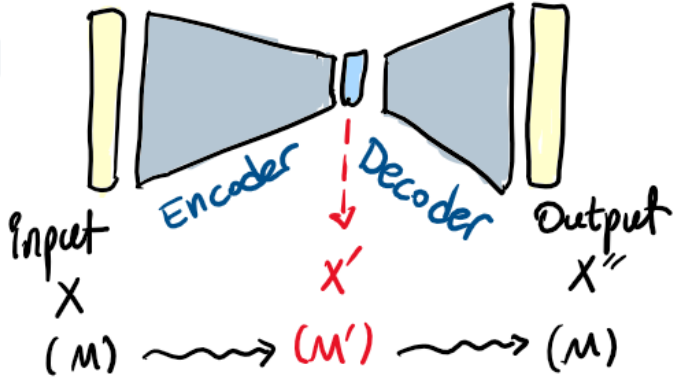
* It is probabilistic in nature



Variational Encoders:

* "AE + Probability"
* introduced in 2013

AE:



☑ Deterministic function

$\propto \; X \rightarrow X' \rightarrow X''$

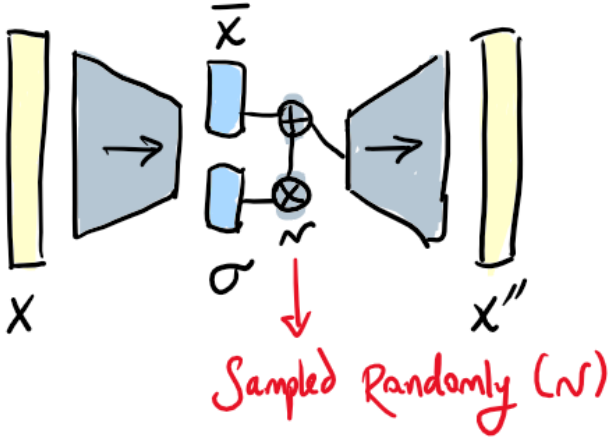
will always be the same

Variational Encoders:

* "AE + Probability"
* Introduced in 2013

VAE: Generative Models

☑ Probabilistic function



$x \rightarrow \begin{pmatrix} \bar{x} \\ \sigma \end{pmatrix} \rightarrow x' \rightarrow x''$

Two outputs: [mean, STD]

Stoc. Selection ==> x'

Encoder

$p_\theta(x'|x)$

Decoder

$p_{\theta'}(x''|x')$

Variational Encoders:

* How do we enforce learning?
  - #1: Creativity
  - #2: Backprop.

* Loss := Reconstruction loss + "Regularization"

$\|x - x''\|^2$

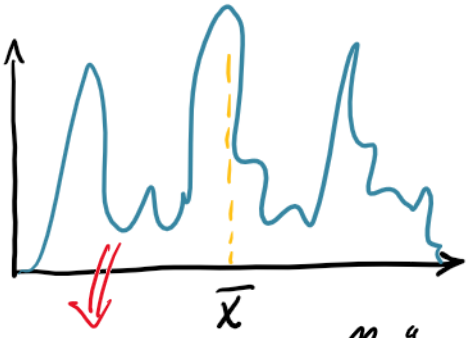
~ as usual ~

Confine Creativity

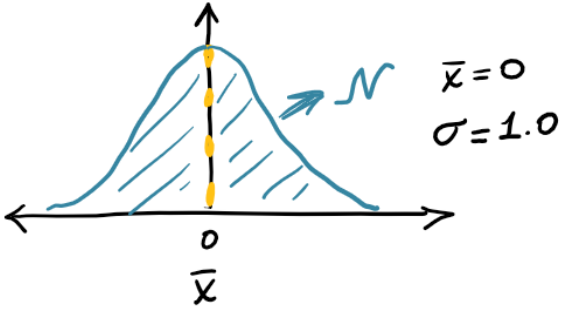
$G = H - TS$

* Solution: Force it to follow a prior distribution.

*

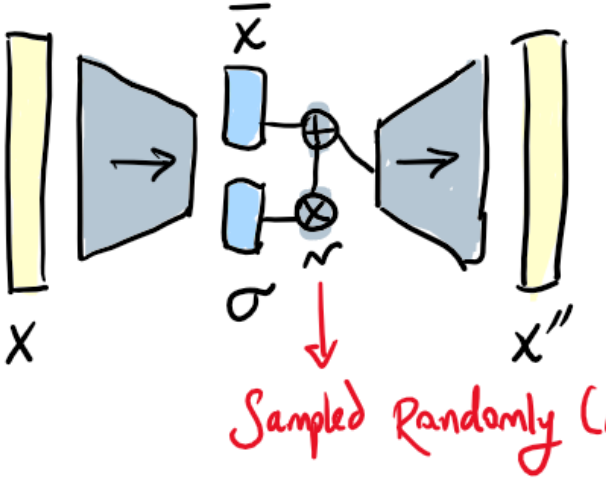


if you let it free, it usually "cheats".

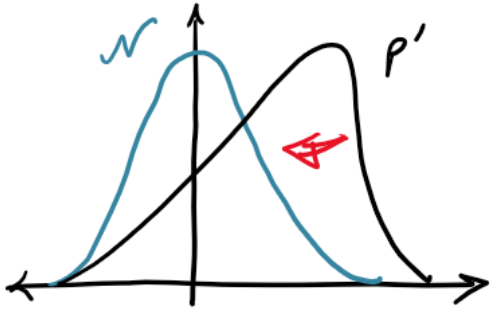


Variational Encoders:

* Gaussian Sampling $\Rightarrow$ [enforced via cost function]



Kullback – Leibler (KL) Divergence ($\mathcal{N}$ & p')



$$D_{KL}\left[p'(\bar{x},\sigma)\,||\,\mathcal{N}(0,1)\right]$$

- $\mathcal{L}_{KL} = -\frac{1}{2}\sum_i 1 + \gamma_i - \exp(\gamma_i) - \bar{x}_i^2$
- $\gamma_i = \ln(\sigma_i^2)$

Variational Encoders:

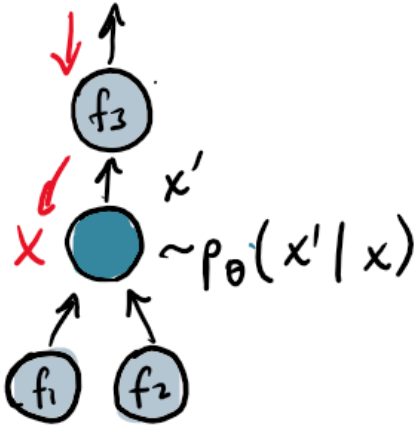
* How do we enforce learning? → #1: creativity
  → #2: Backprop.
* We cannot backprop. a stoc. layer!

$x' = \bar{x} + \sigma\epsilon$

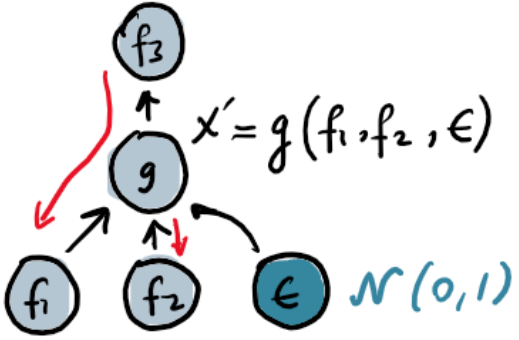
$\bar{x}$: mean vector

σ: std vector

↳ ϵ: point sampled in $\mathcal{N}$



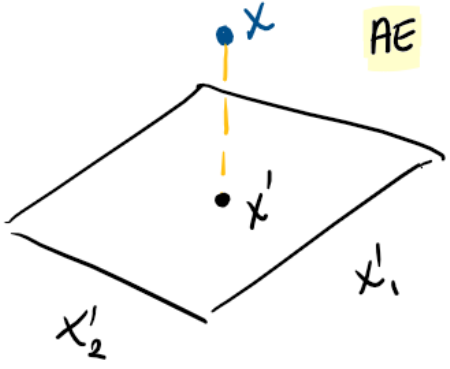
⇒

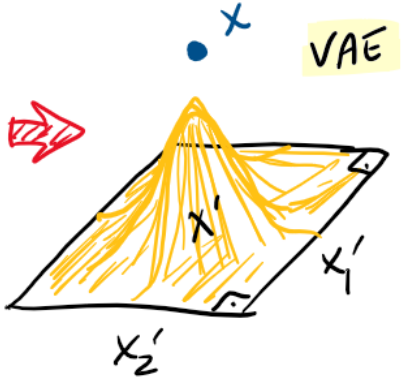


Variational Encoders:

* Gaussian Sampling

It creates a cont'd. latent space.





$[x_1', x_2']$

covariance matrix is diagonal.

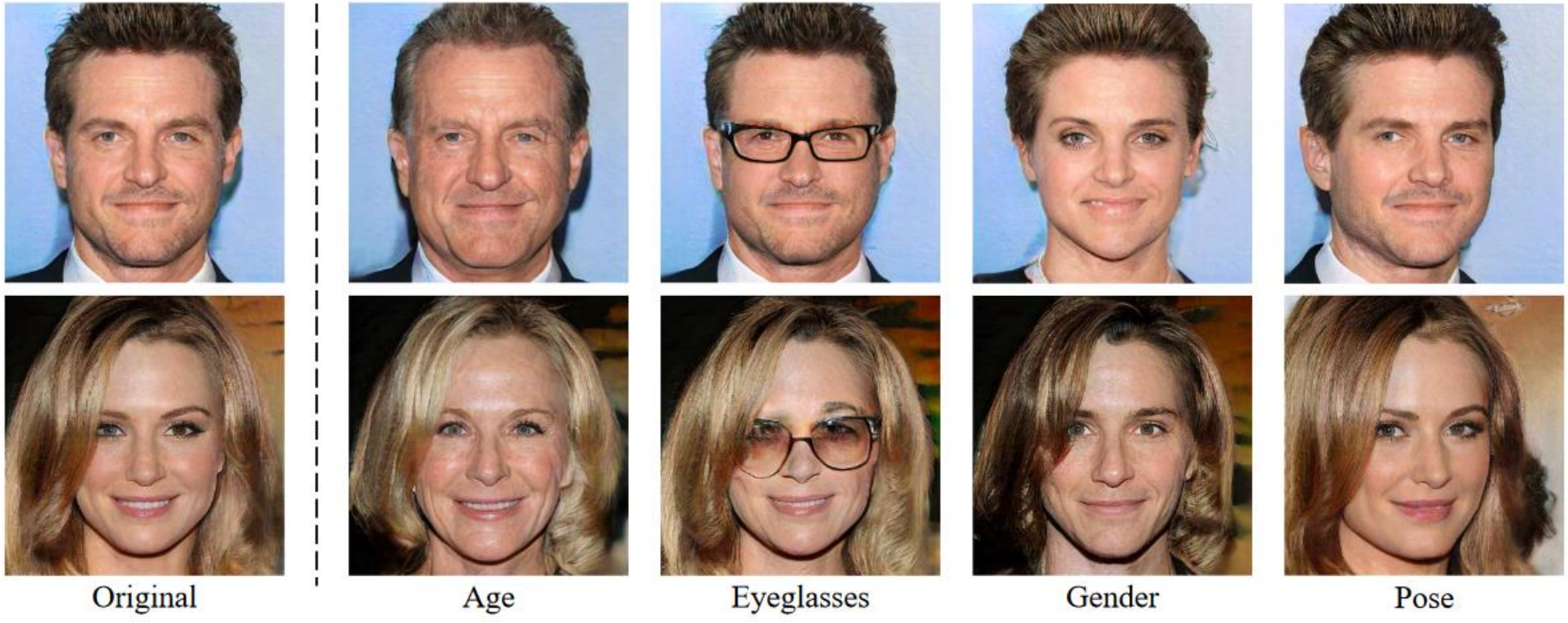
You can perform semantic interpolation.

* Case 1: $[\bar{x}, \sigma]_1$
  Case 2: $[\bar{x}, \sigma]_2$
  Case 3: $[\bar{x}, \sigma]_3$

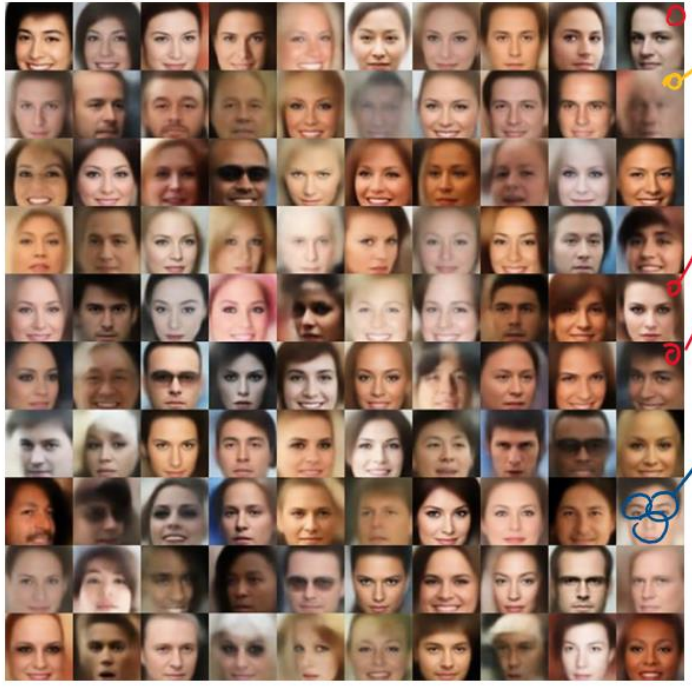
"Case 4 := Case 1 − Case 2 + Case 3"

Case 5 := α (Case 1) + $(1-\alpha)$ Case 2

Interpolation ⇨ Latent Space Arithmetics



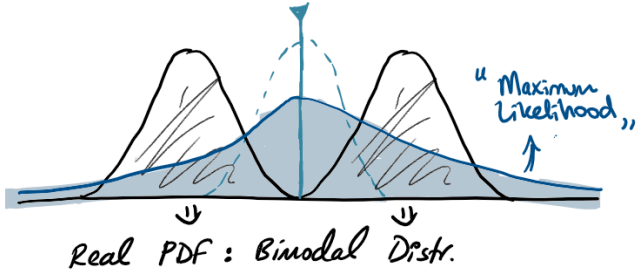
Variational Encoders:



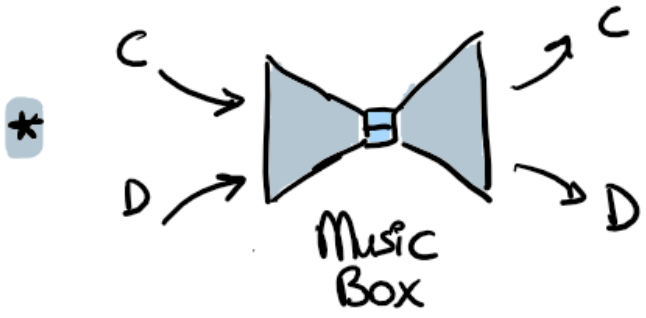
Blurry Edges

Blended in the background (safe pixel values)

Features are captured: eyes, nose, ...



Our Strategy:



II inputs

II outputs

Case #1: C, ~~D~~ ⇒ VAE:

= [[samples], features]

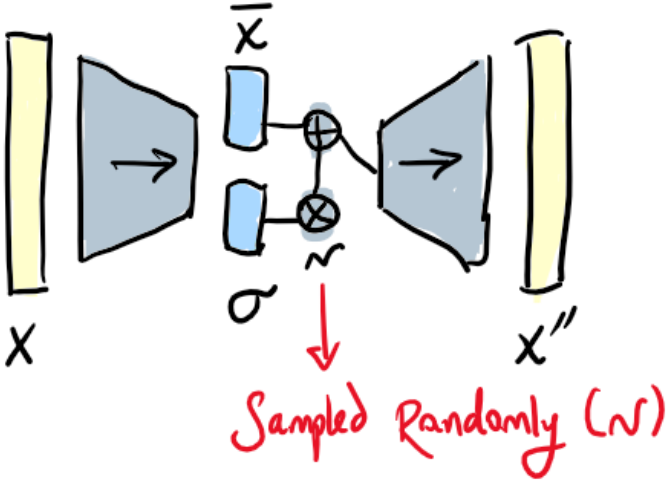
Case #2: C, D ⇒ RNN:

= [samples, seq., features]

* Building our Graph. Network :

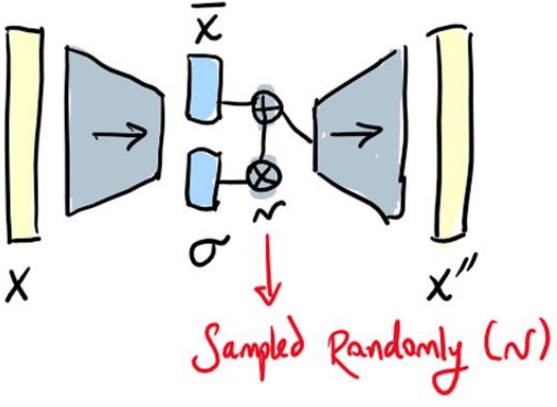
* Prev. we used Sequential API in TF.
  - Dense connections
    - Automated

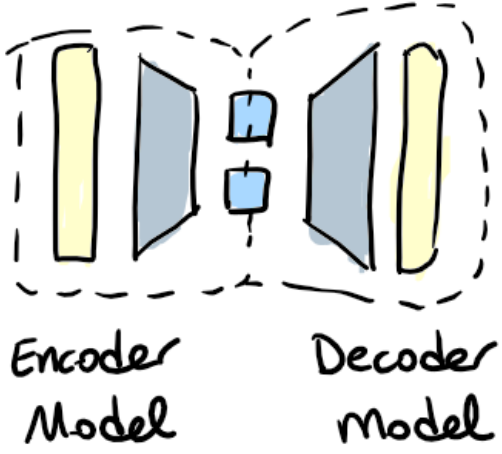
* Custom. Structure ⇒ Functional API



* Building our Graph. Network :

* Custom. Structure ⇒ Functional API



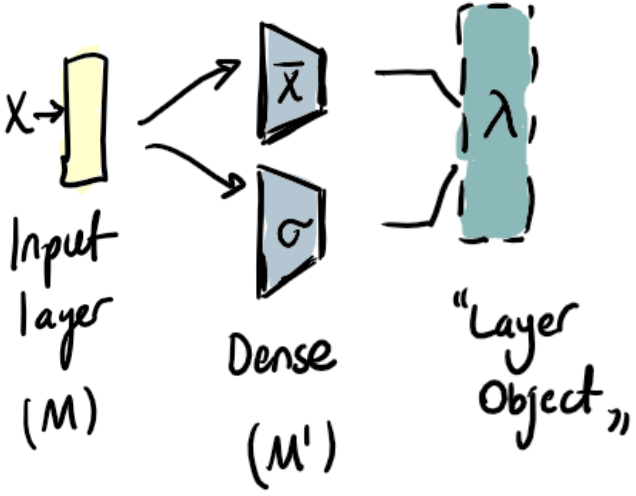


☑ Train ⇒ (E) + (D)

☑ Gen ⇒ (D)

* Building our Graph. Network :

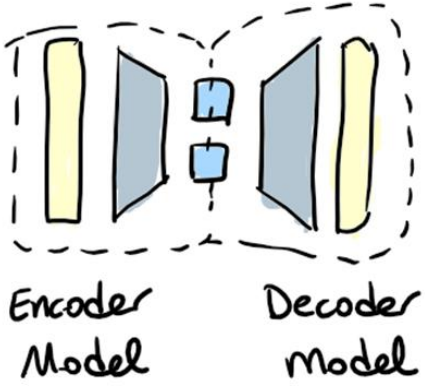
Encoder :

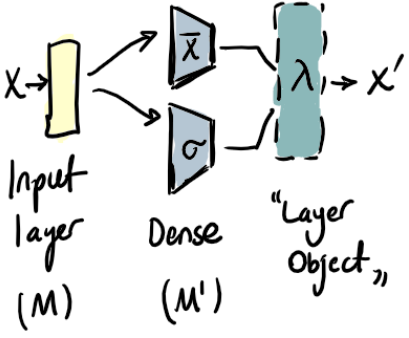


Sampling :

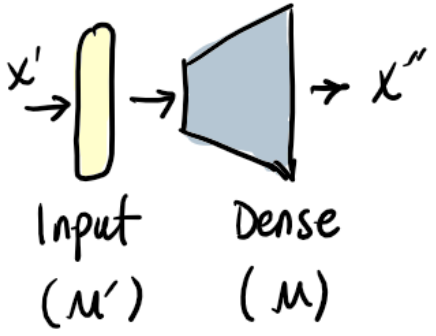
- $x' = \bar{x} + \sigma\epsilon \rightarrow x'$
- $\epsilon \leftarrow \mathcal{N}$

* Building our Graph. Network :





+



Encoder = [

④ { input, mean, logVar, lambda]

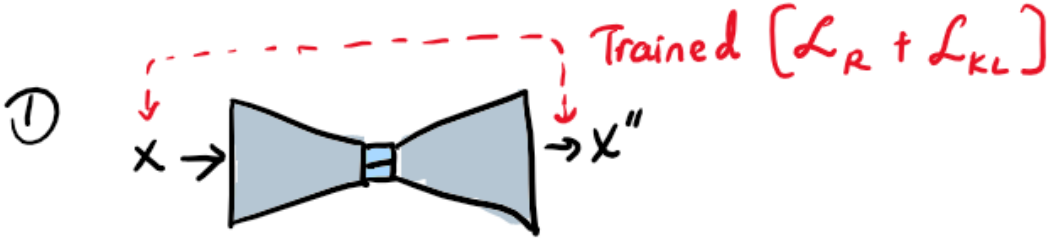
+

Decoder = [

② { input, decoder]

colab

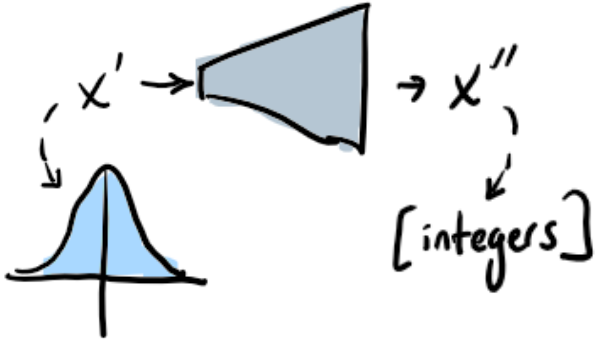
Generating Music :



(i) Generate noise via $\mathcal{N}$

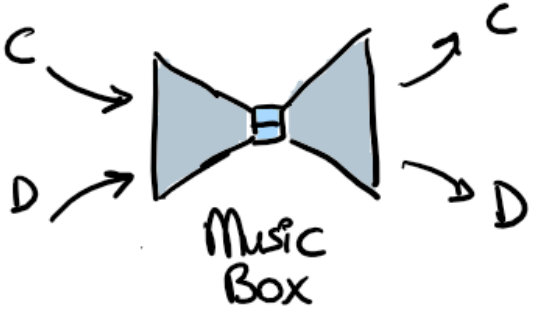
(ii) Get x'' [integers]

(iii) (int) → MIDI →



colab

Our Strategy:



* Case #2: C, D ⇒ RNN: (LSTM)

= [samples, seq., features]

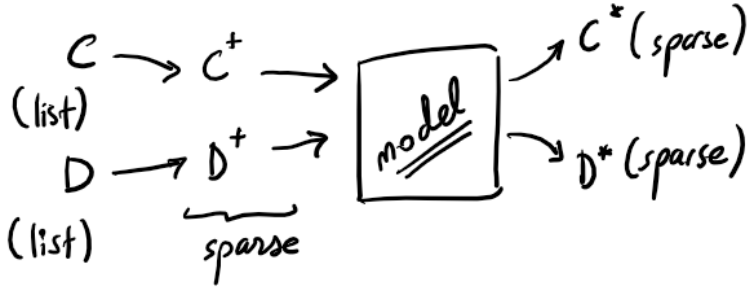
II inputs

II outputs

* LSTM → [X] data
→ y labels → next chord
→ next duration

Our Strategy:

* Idea:



Memory-intensive

Alternative ⇒ "Embeddings"

Eg: Possibility = [0, ... 1000]

(i) Encoding ⇒ "4" ⇒ Vector of size 1000
[1 → 1, 999 → 0]

(ii) Embedding ⇒ Integers ⇒ vector of fixed size (float)

Eg:

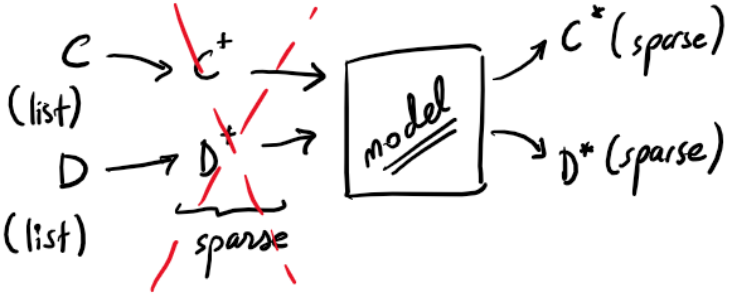
vector_size = 3

0 → [1.1, 2.6, 2.4]
1 → [0.1, 4.2, 1.5]
⋮

TF optimize it during the training.

Our Strategy:

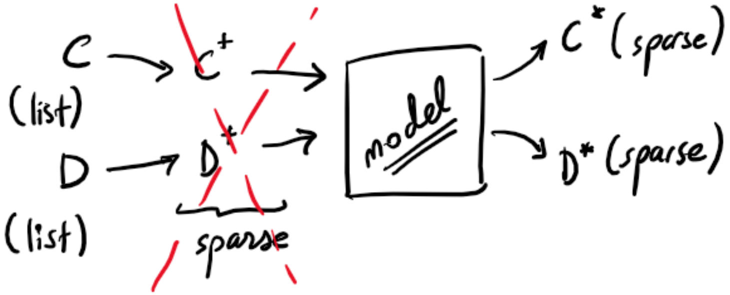
* New Idea:




```
chord input
[164 164 149 164 182 164 157 149 114  92 108 108  90 114 108  95  92 164
 184 150 154  48  48   1  16 255 208 224 233 422 417 451]
duration input
[4 4 4 4 2 2 2 2 2 2 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4]
```

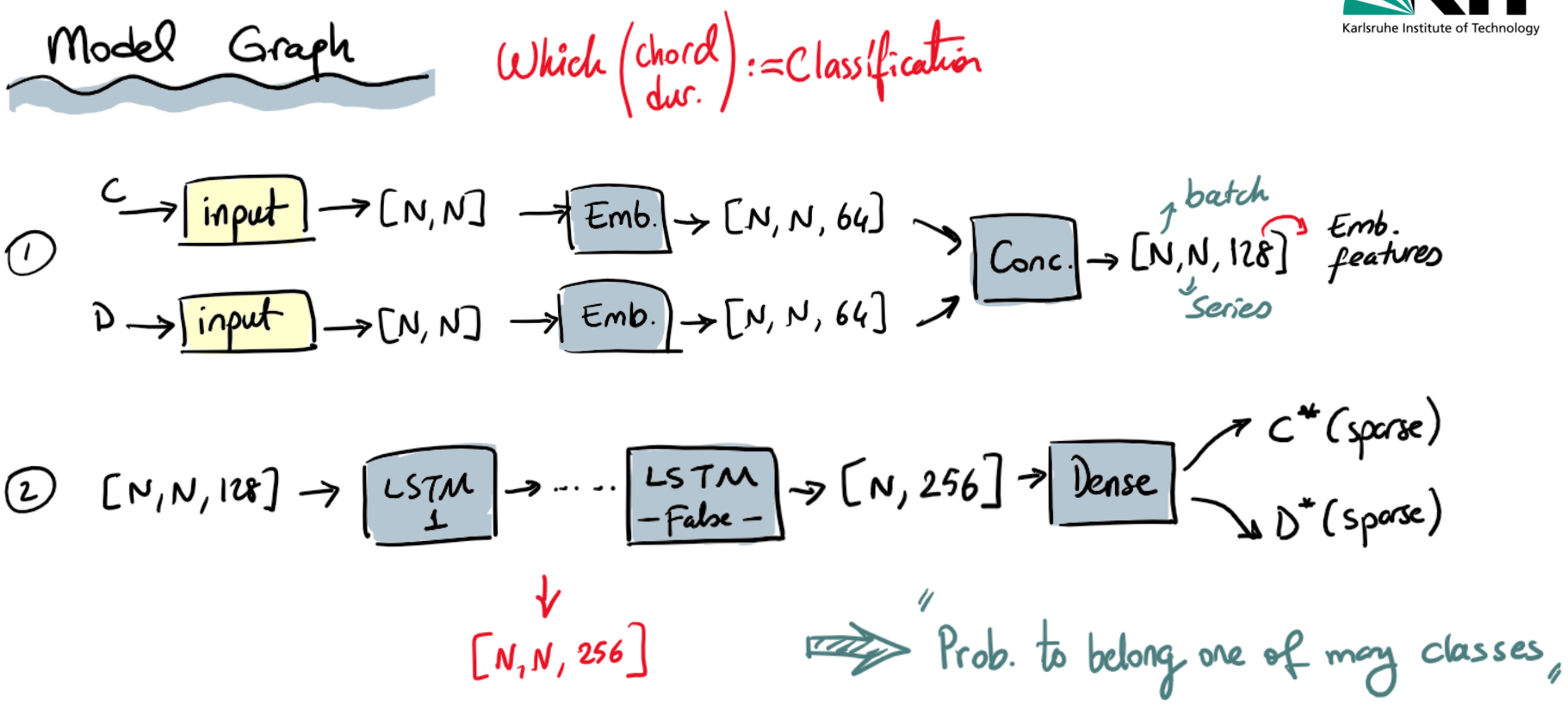
Our Strategy:

* New Idea:



* Still Sparse output?

```
chord output
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
duration output
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
```


Model Graph
Which (chord dur.) :=Classification
1
C
input
[N, N]
Emb.
[N, N, 64]
D
input
[N, N]
Emb.
[N, N, 64]
Conc.
batch
[N, N, 128]
series
Emb. features
2
[N, N, 128]
LSTM 1
LSTM -False-
[N, 256]
Dense
C* (sparse)
D* (sparse)
[N, N, 256]
"Prob. to belong one of many classes"

colab

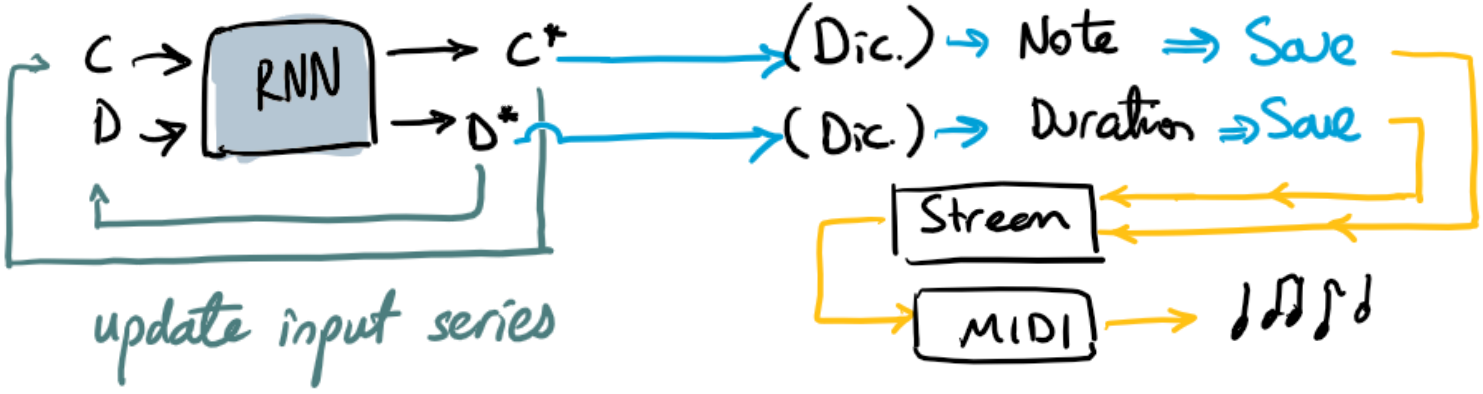
Back – conversion :

① C, D → Model → C* → ⇒ Pick most probable one
Model → D* → ⇒ Pick most prob. one

② C* → {Int_to_Chord} → Note (str.)
D* → {Int_to_Duration} → Duration (str.)

Back – conversion :

③ For generating long music;

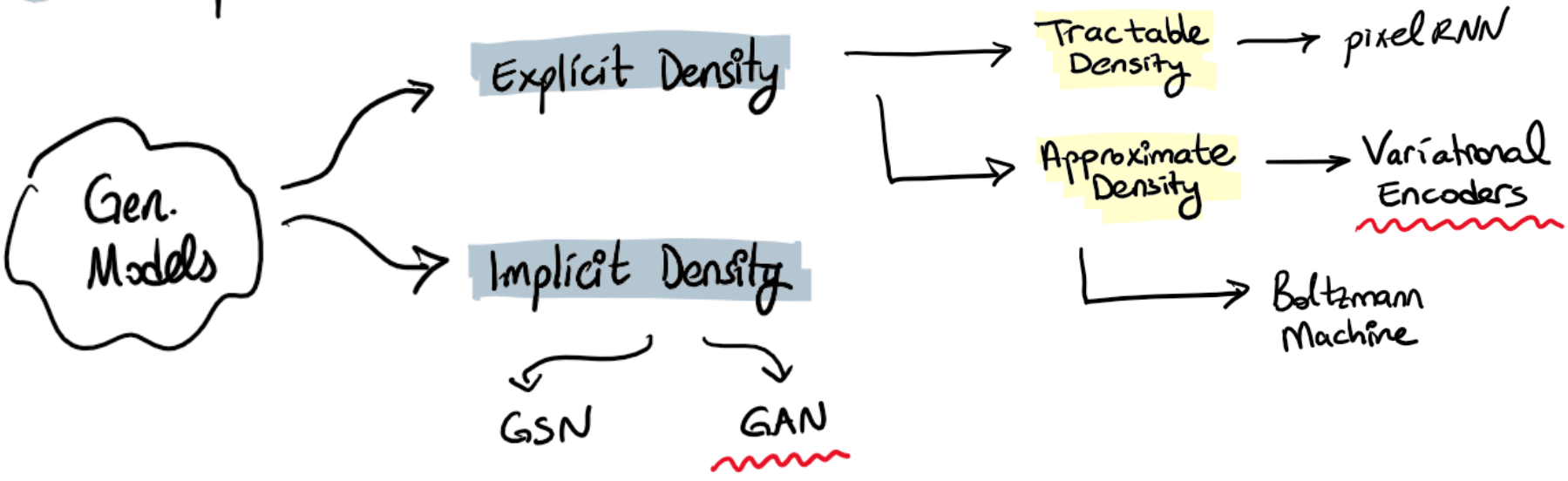


Ateliers & Saveurs in Montreal

colab

UL → Generative Models :

* It is probabilistic in nature



Generative Adversarial Networks: GANs

* Generative ⇒ creating non-existing data
* Adversarial ⇒ Competitive dynamics (game-like)
* Network ⇒ Neural networks
* GANS (2014)
  - Generator
  - Discriminator
* Random Numbers → Generator → Fake Examples

  Obj: Be as realistic as possible
* Real examples → Discriminator → probability ⇒ [0, 1]

  Fake Examples → Discriminator

  Obj: Distinguish Fake from Real data

Training Algorithm:

For each training do:

Train (D):

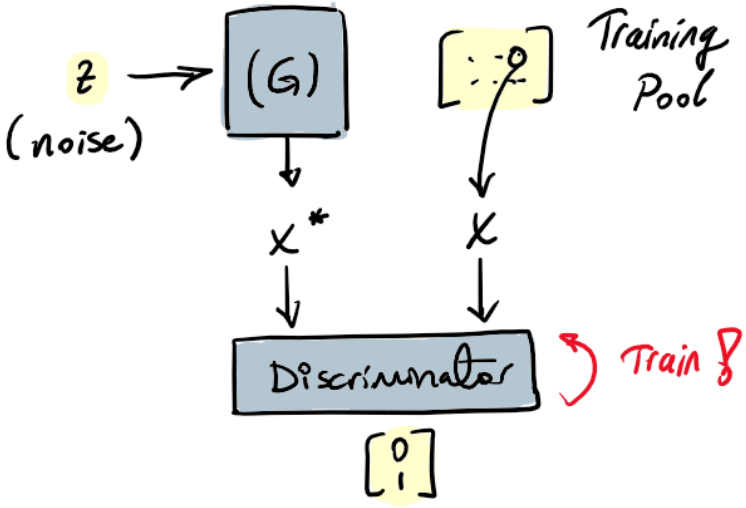
(1) Take a random real example from Training data, x

(2) Get a fake example from Generator, x^*

(3) Use Discriminator to classify x & x^*.

(4) Compute the class. error.

(5) Backprop. error & update Discriminator trainable parameters.



Training Algorithm:

#Train (G):

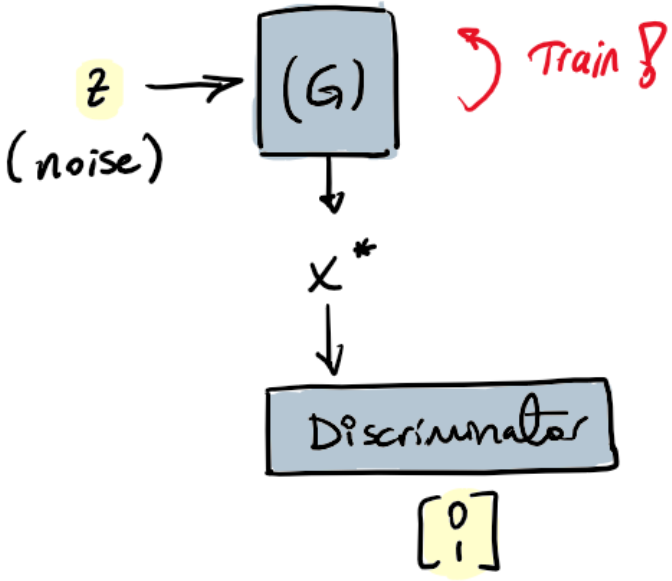
(6) Generate a new fake X^*.

(7) Use Discriminator to classify X^*.

(8). Compute the error.

(9). Update Generator 's trainable parameters via backprop.

end for



Training GANs

* In MLP, we have a clear goal & measure
  Ex Minimize Cross-entropy loss.

* In GANs, two networks have competing obj. !
  (G) ↑ ; (D) ↓ // (D) ↑ ; (G) ↓

Training GANs

Nash Equilibrium := Point where neither "player" can improve their situation

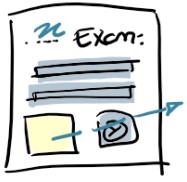
- (G) := Fakes are indistinguishable from Real data
- (D) := at best randomly guess (F/R $\Rightarrow$ 1)

In practice; ~ impossible to achieve Nash Eq.

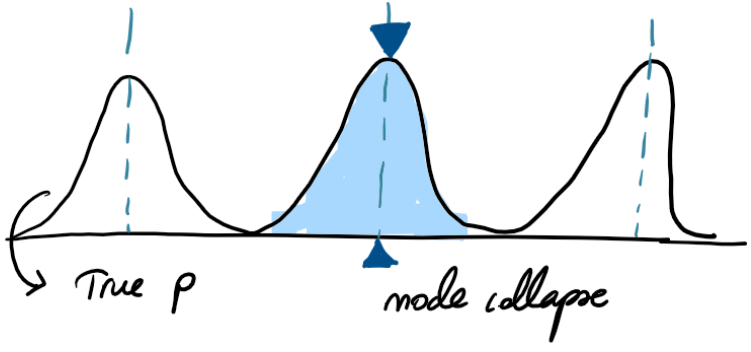
it still works ...

Training GANs

Training problems #1: Mode Collapse



"Abracadabra" ~"I create as I speak"



$$\begin{bmatrix} -4 \\ -2 \\ 0 \\ +2 \\ +4 \end{bmatrix}$$

True values

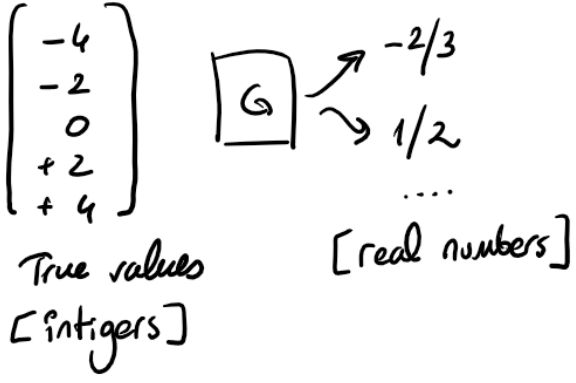
G ⇒ "0" (✓)

~~[-4, -2, 2, 4]~~

Training GANs

Training Problem #2 : Over generalization

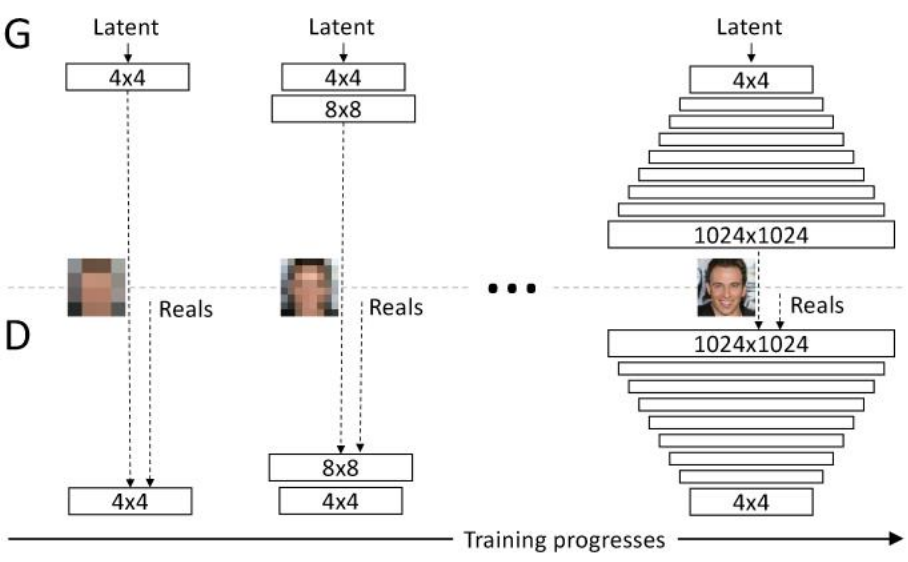
* Mods that should not exist, do exist.

* 

* Image generation

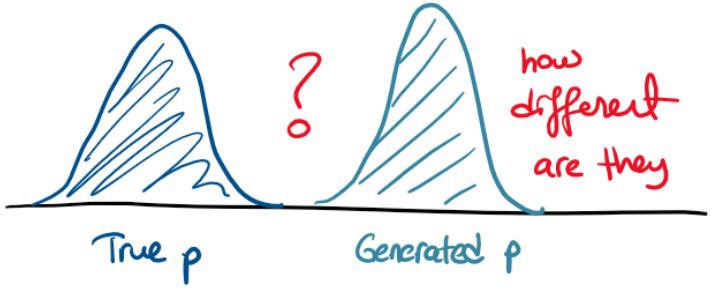
Possible Remedies

① Growing the network gradually



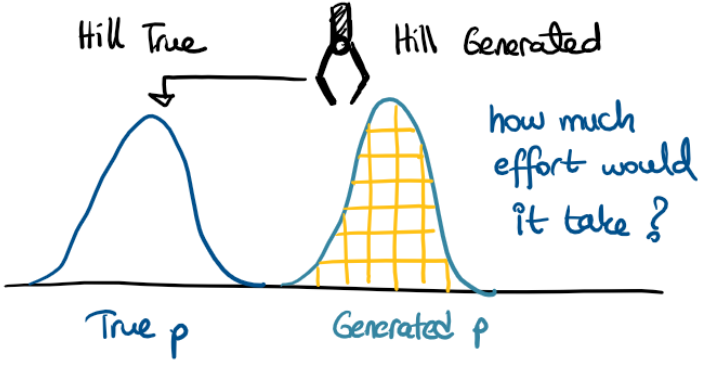
Possible Remedies

② Alternative loss definitions ⇒ Wasserstein GAN



* Distance := (Similarity)
- TV distance
- KL divergence
- JS divergence
- ...

⇒ Earth-mover Distance



Possible Remedies

② Alternative loss definitions ⇒ Wasserstein GAN

* Class. ⇨ [0, 1]
  ↳ Train... ⇨ 0.9971 ‖ 0.0004
  0.9965 ‖ 0.00013
  0.9981 ‖ 0.0002
  ...

⇩

vanishing gradient problem

* Stabilized learning