JDLA E資格認定プログラム

全人類がわかるE資格コース

生成モデル

© AVILEN Inc.

目次

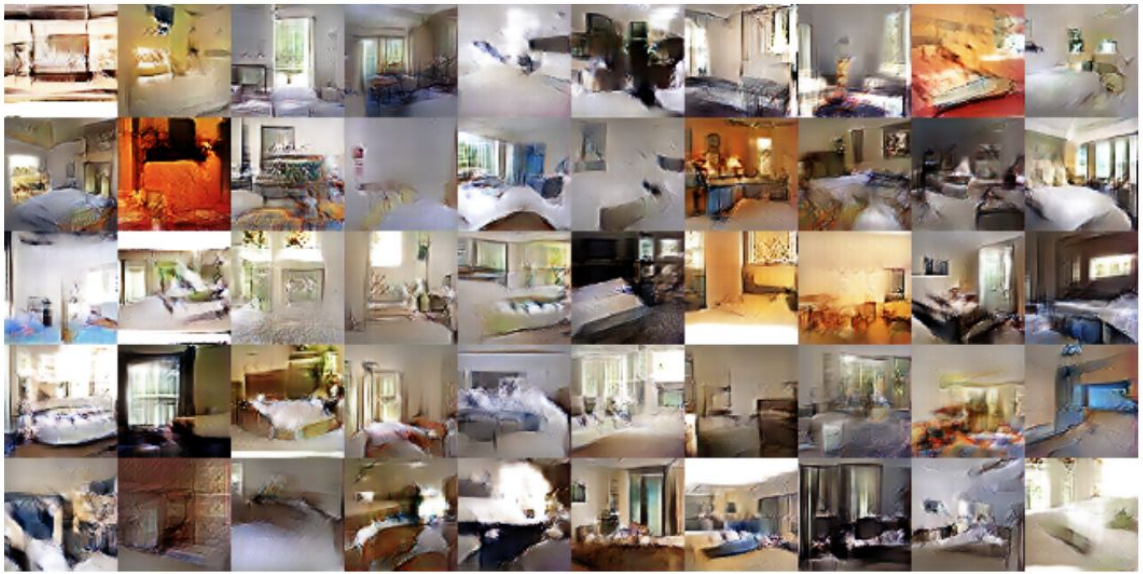
© AVILEN Inc.

【Chapter07】生成モデル

生成モデルの考え方

生成モデルの応用例

すべて実在しない寝室であり、生成モデルが作った



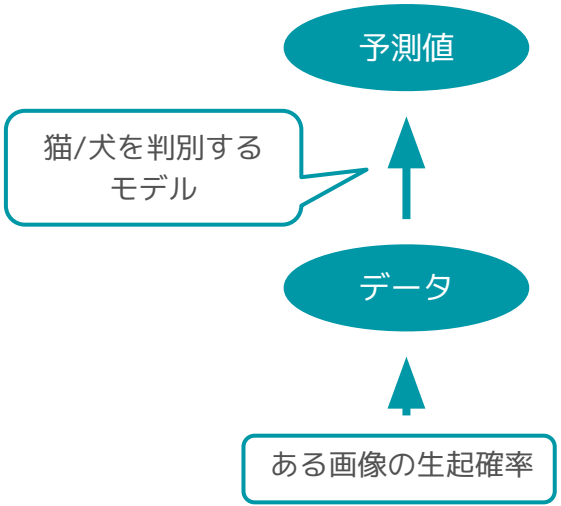
UNSUPERVISED REPRESENTATION LEARNING
WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

© AVILEN Inc.

識別モデルと生成モデル

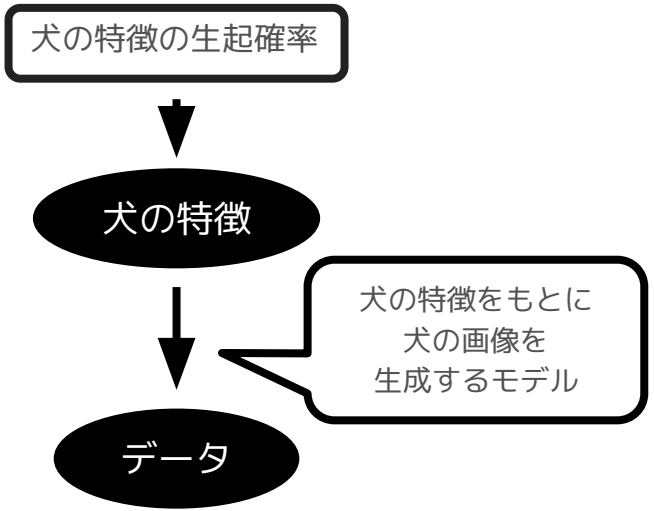
識別モデル

与えられたデータから
「ある可能性」を予測することが目的



生成モデル

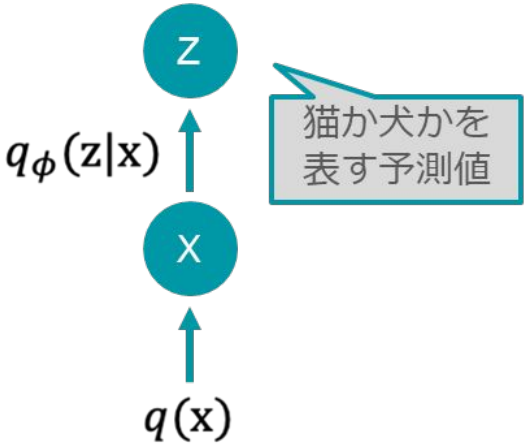
与えられたデータから
「新たなデータ」を生成することが目的



© AVILEN Inc.

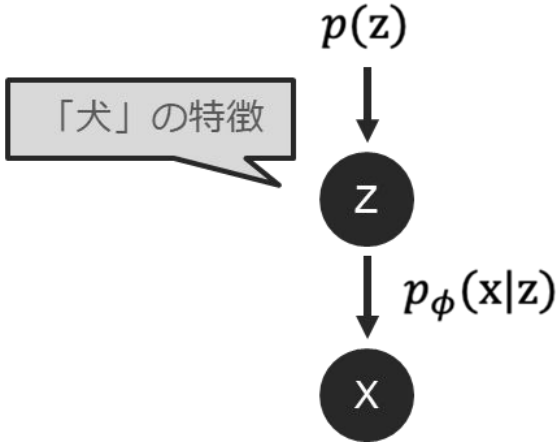
識別モデルと生成モデル

識別モデル



$q(x)$：ある画像データの生起確率
$q_\phi(z|x)$：猫/犬を判別するモデル：
ϕモデルのパラメータ

生成モデル

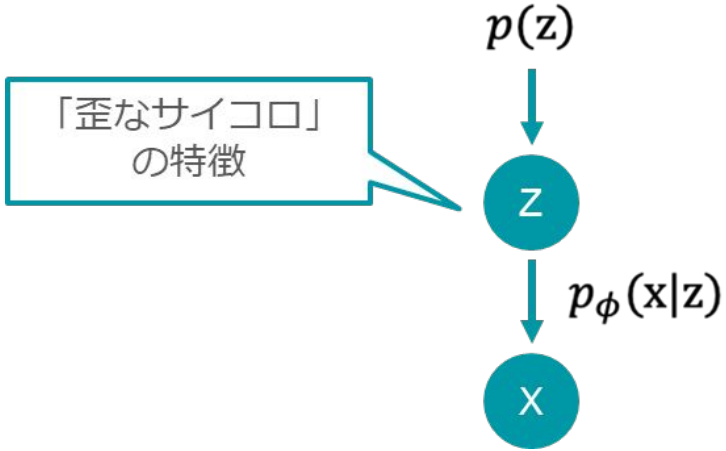


$p(z)$：犬の特徴の生起確率
$p_\phi(x|z)$：犬の特徴から画像を生成するモデル
ϕ：モデルのパラメータ

© AVILEN Inc.

生成モデルとサイコロ

歪なサイコロの観測データからそのサイコロの「形」を生成することで、
歪なサイコロと近いシミュレーションが可能に



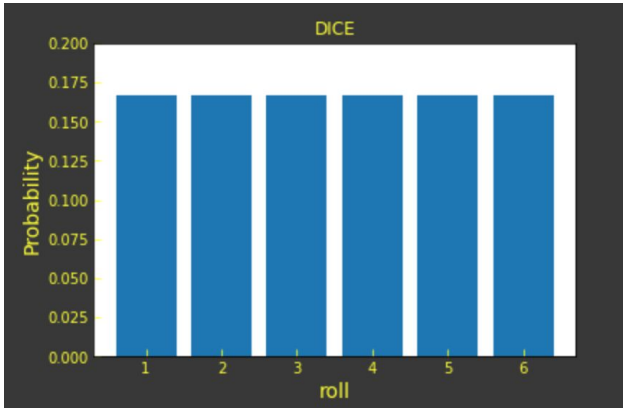
$p(z)$：歪なサイコロの特徴の生起関数
$p_\phi(x|z)$：歪なサイコロの特徴をもとに、擬似サイコロを生成するモデル

© AVILEN Inc.

【補足】サイコロの別の見方

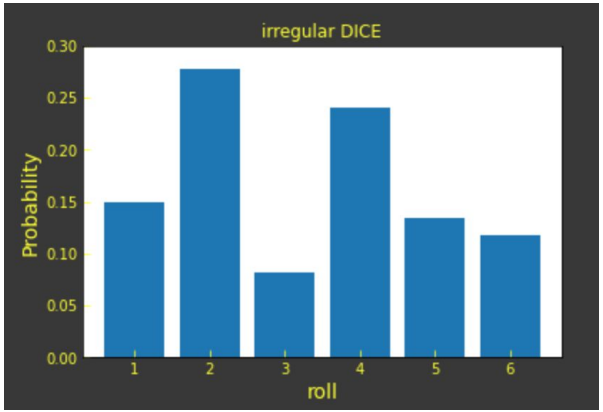
サイコロ＝「それぞれの出目の出現確率が割り当てられたモデル」

通常のサイコロ



https://illust8.com/contents/2692

歪なサイコロ



http://torito.jp/shopping/item.cgi?_skewdice

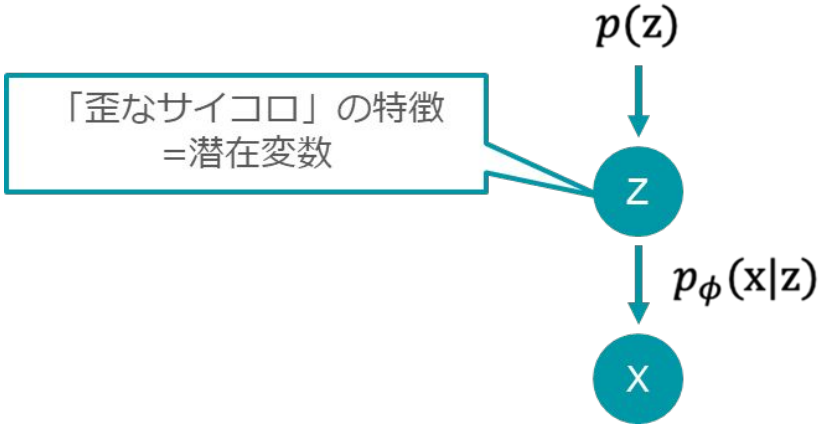
© AVILEN Inc.

【Chapter07】生成モデル

潜在変数

潜在変数とは

潜在変数 z：潜在的な特徴を表す変数



x：データ

$p(z)$：歪なサイコロの特徴の生起関数

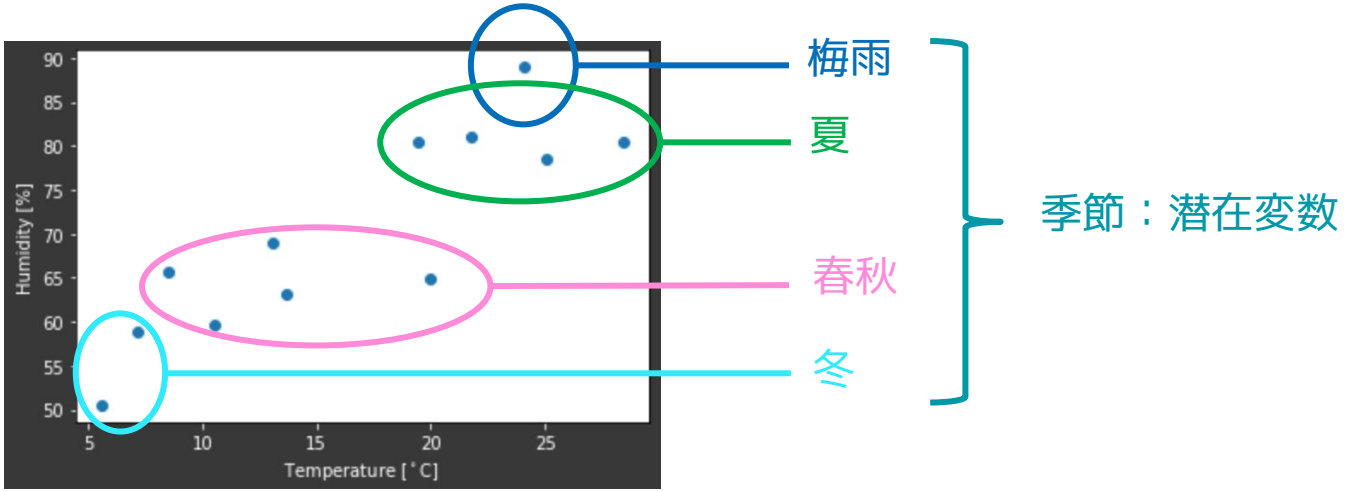
$p_\phi(x|z)$：歪なサイコロの特徴をもとに、擬似サイコロを生成するモデル

© AVILEN Inc.

潜在変数とは

（例）季節

「月間平均気温(横軸)」と「月間平均湿度(縦軸)」の２つの軸から
「季節」という１つの軸でデータを捉える

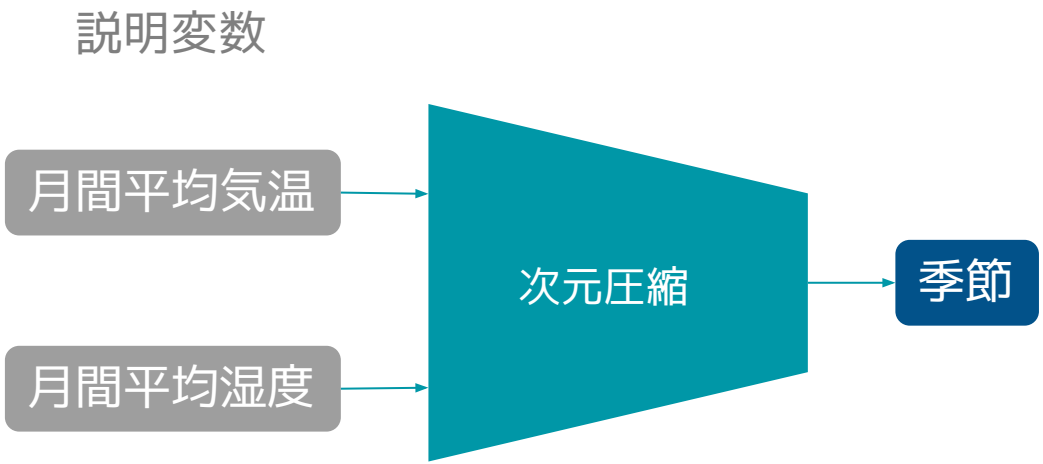


気象庁 2019年東京の月間気象データ

© AVILEN Inc.

潜在変数とは

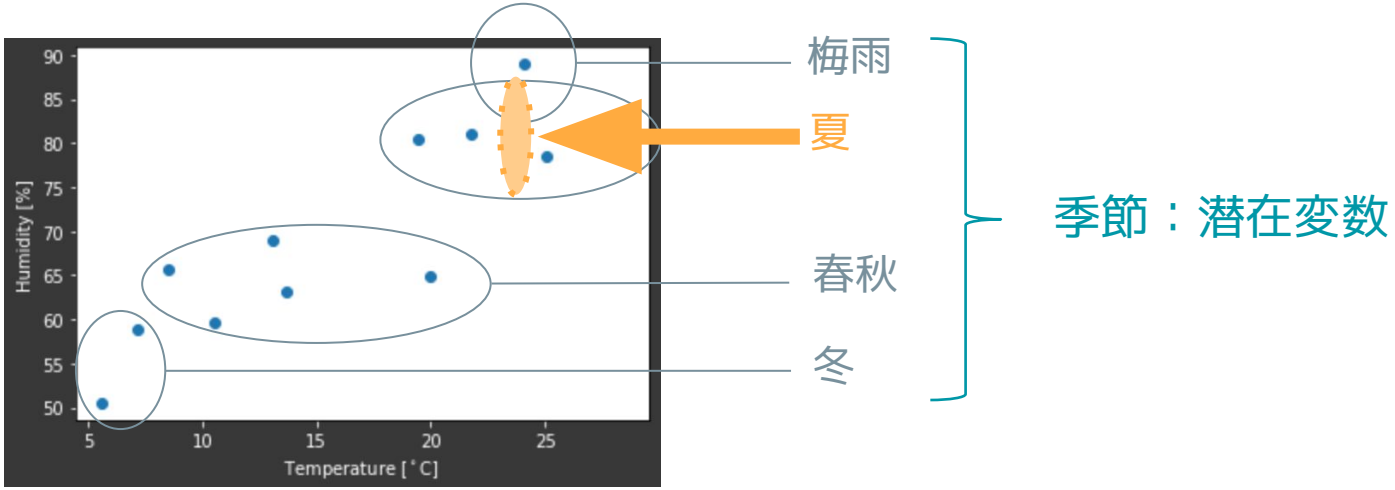
潜在変数は「次元圧縮後の世界」



© AVILEN Inc.

潜在変数の効果

潜在変数からデータを生成できる
（例）「夏」という潜在変数から
「湿度・気温が高い」データを生成できる可能性が高い



気象庁 2019年東京の月間気象データ

© AVILEN Inc.

【Chapter07】生成モデル

エンコーダ・デコーダモデル

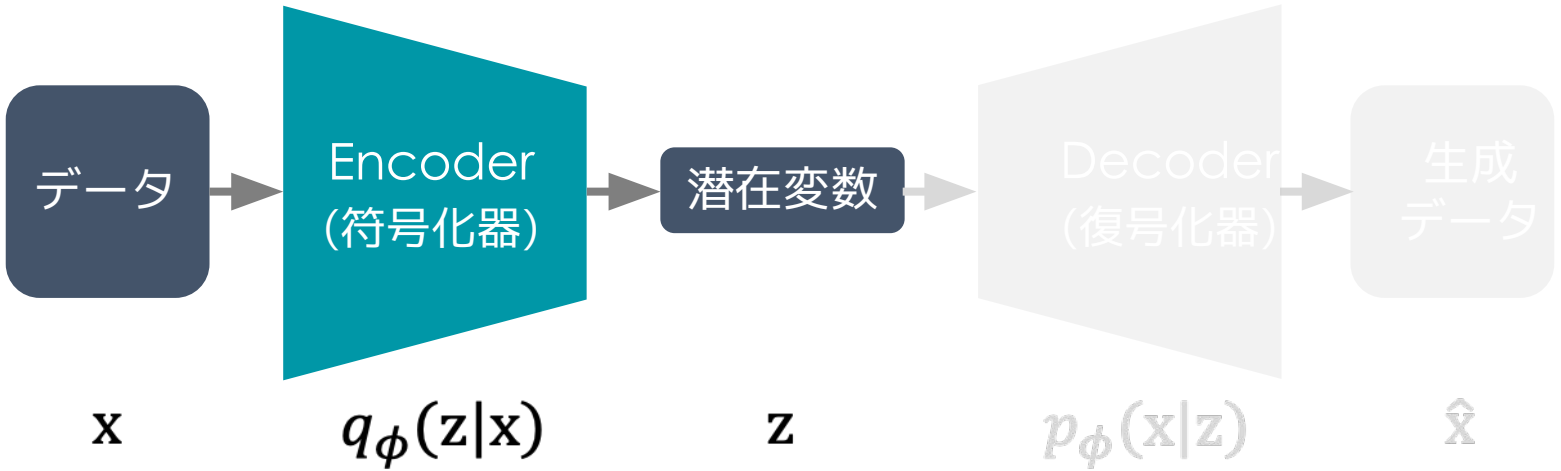
概念図

1. Encoderが潜在変数に落とし込み、
2. Decoderは潜在変数をもとにデータを生成する

© AVILEN Inc.

エンコーダ

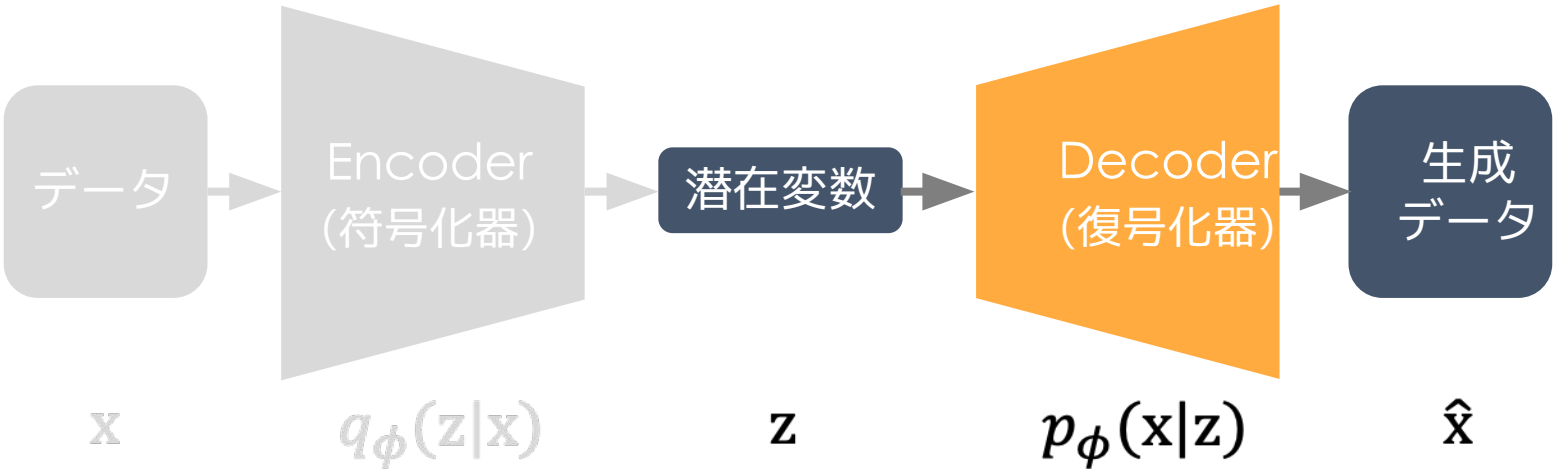
エンコーダ$q_\phi(\mathrm{z}|\mathrm{x})$：高次元データから潜在変数を推測するモデル



© AVILEN Inc.

デコーダ

デコーダ$p_\phi(\mathrm{x}|\mathrm{z})$：潜在変数からデータを生成するモデル



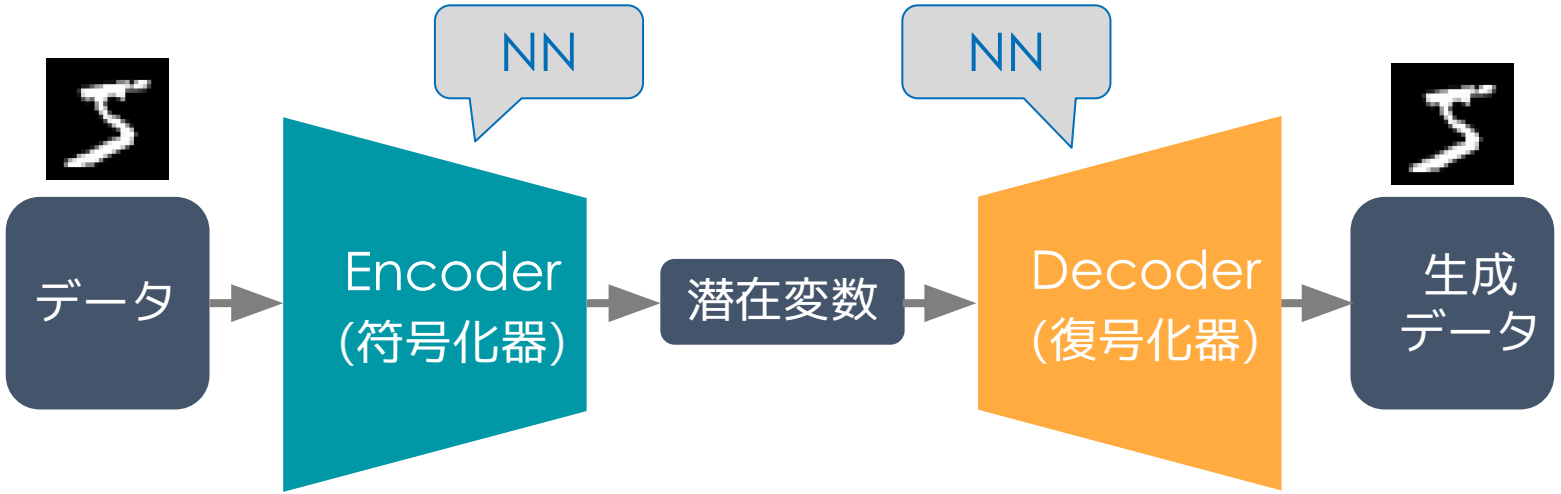
© AVILEN Inc.

【Chapter07】生成モデル

AE

オートエンコーダ(AE)

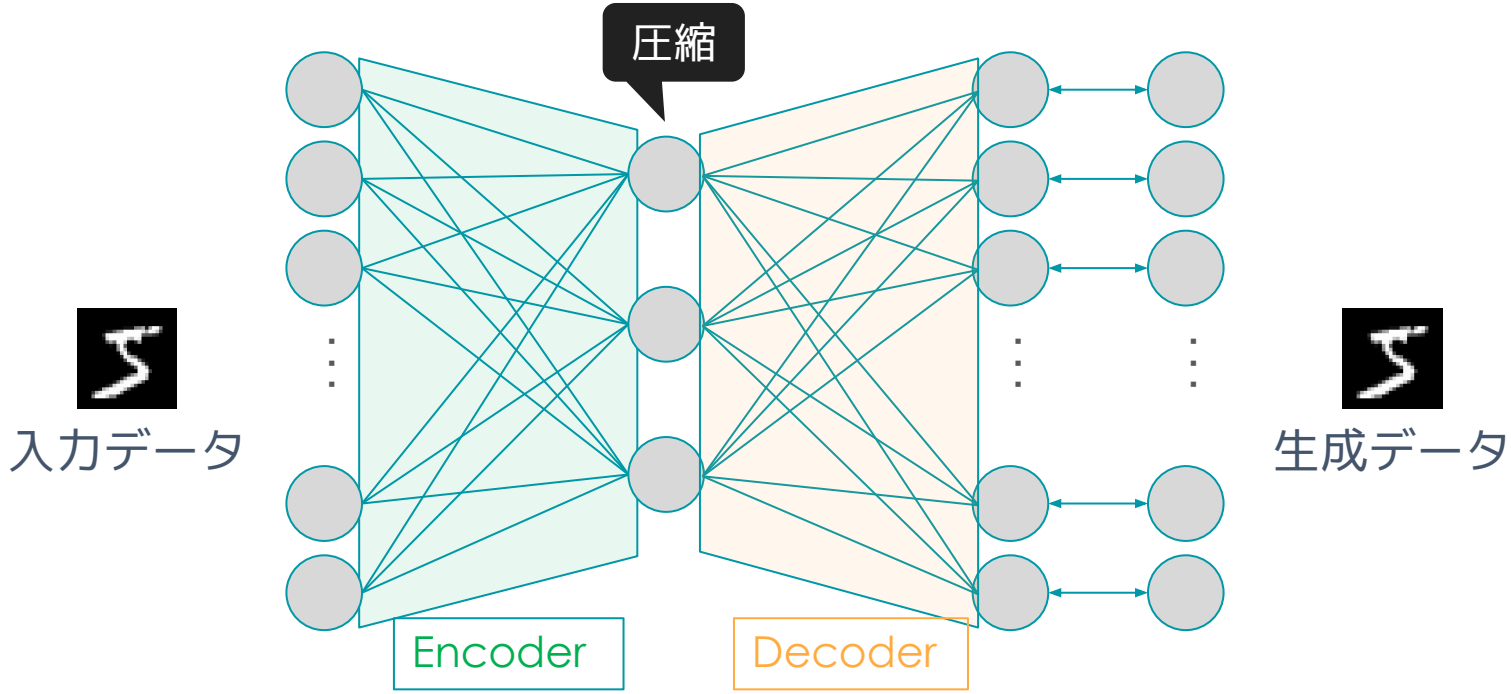
- EncoderとDecoderはニューラルネットワーク
- 入力と同じものを出力するように学習



© AVILEN Inc.

オートエンコーダ(AE)

隠れ層のノード数は入力層・出力層より少なくする

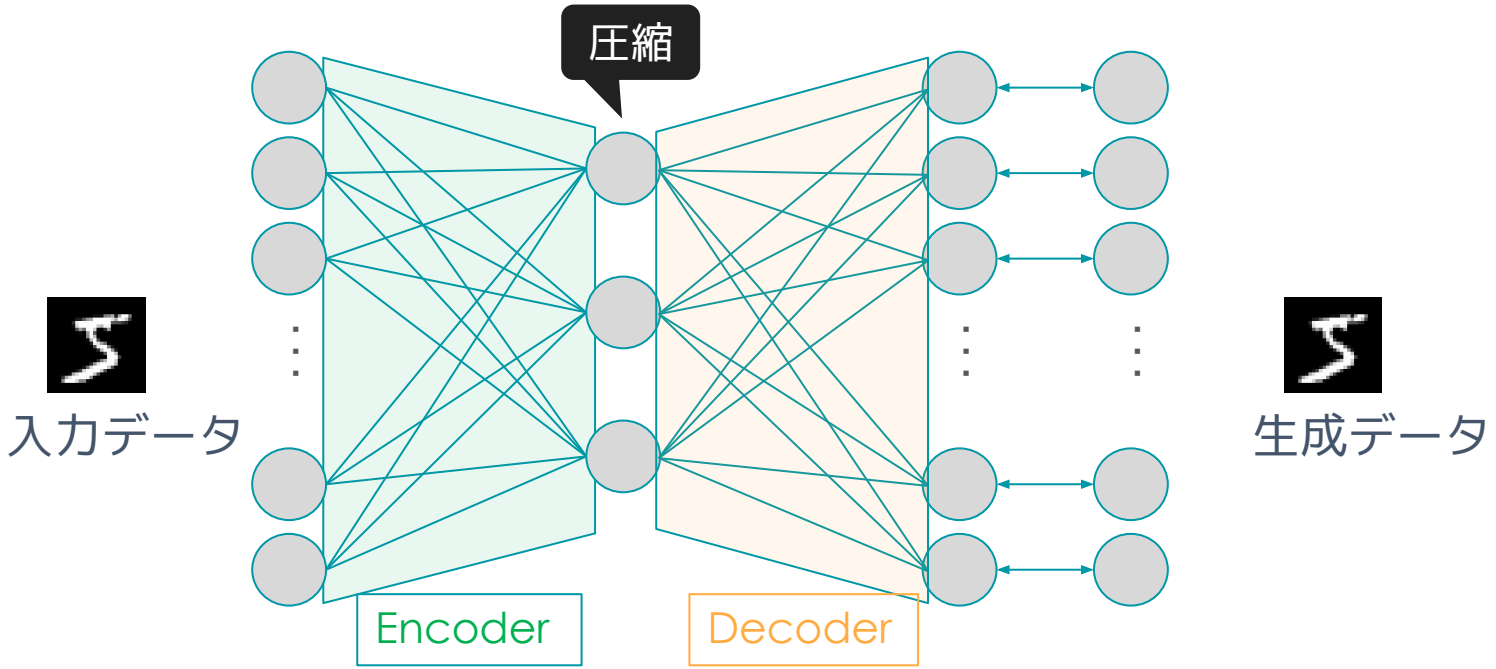


© AVILEN Inc.

AEの次元圧縮効果

入力と同じものを、入力の次元より少ない次元の潜在変数から出力させる

⇨**効果的な潜在変数が作られる**



© AVILEN Inc.

【応用例】AEの次元圧縮効果

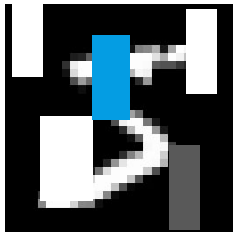
予測型NNモデル(識別モデル)とは全く異なる目的で使われる

- 画像のノイズ除去
  ：Encoderで次元圧縮で抽象化してからDecoderで元の次元に戻す

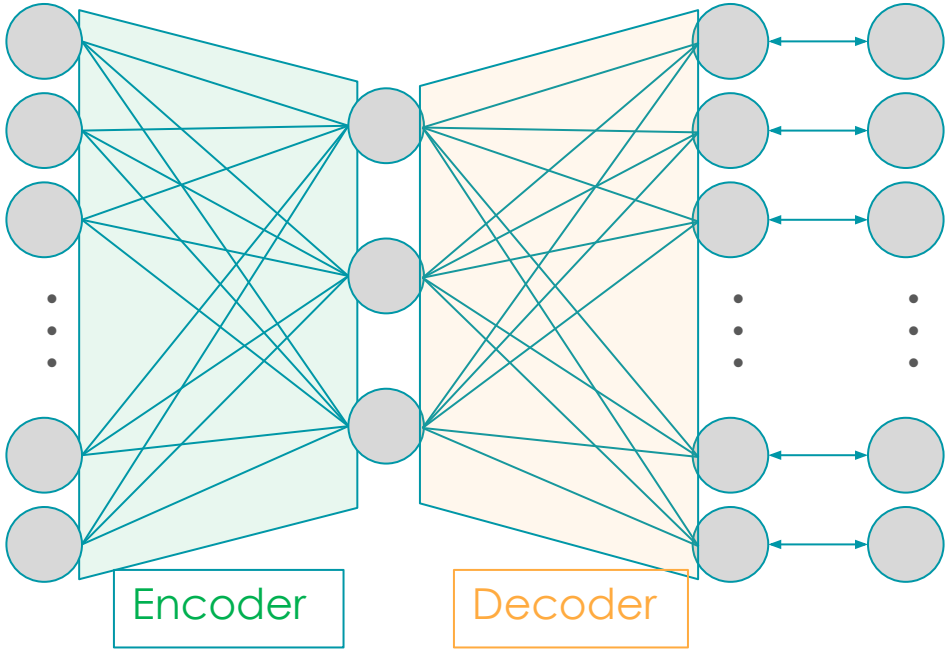
- クラスタリング
  ：Encoderで次元圧縮された特徴分布上でクラスタリングすることで、
  データ中のより重要な特徴をもとにクラスタリングできる

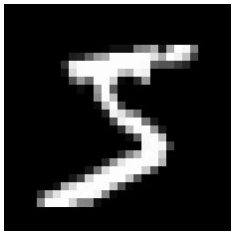
© AVILEN Inc.

入力にノイズを加え、ノイズが無いデータを生成することを目指すモデル



ノイズ付き
入力データ





生成データ

© AVILEN Inc.

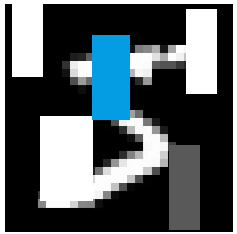
ノイズの種類

- マスキングノイズ
  ・・・入力の一部を0にする
- ソルト＆ペッパーノイズ
  ・・・入力の一部を0か1にする
- ガウシアンノイズ
  ・・・入力に平均0分散σ^2(固定値)のガウス分布からサンプルした値を加える

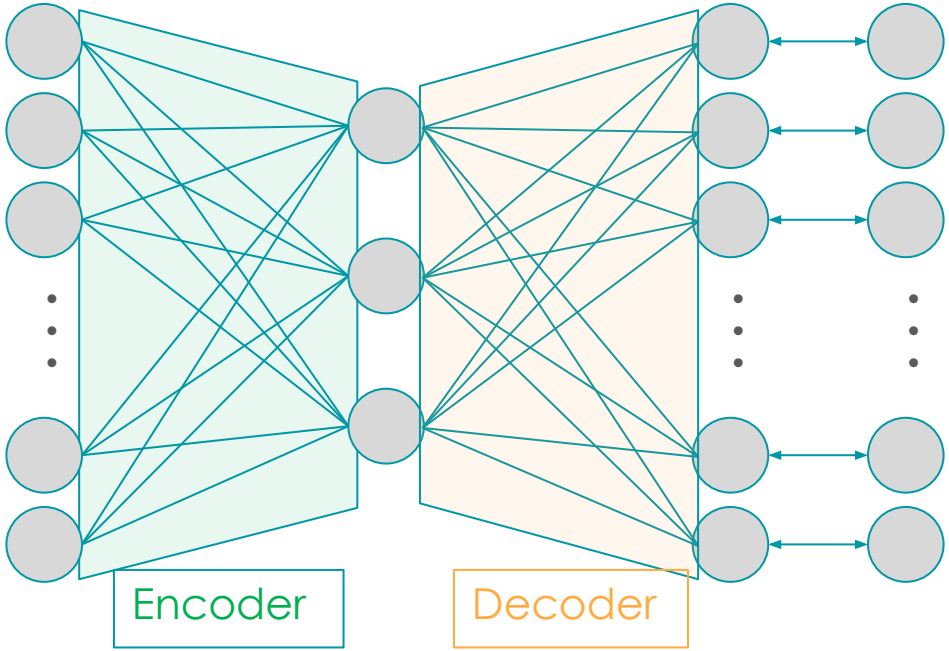
© AVILEN Inc.

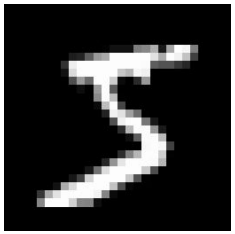
DAEの効果

データのノイズ除去機能としての性能が上がる



ノイズ付き
入力データ





生成データ

© AVILEN Inc.

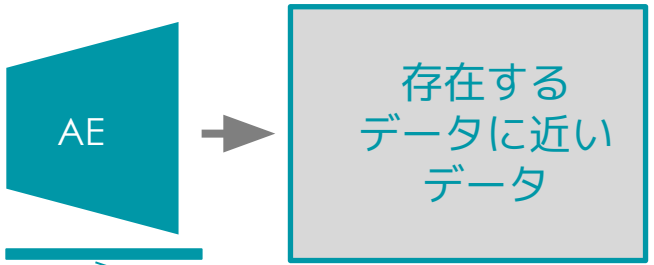
【Chapter07】生成モデル

VAE

AEとVAE(Variational Auto Encoder)の目的

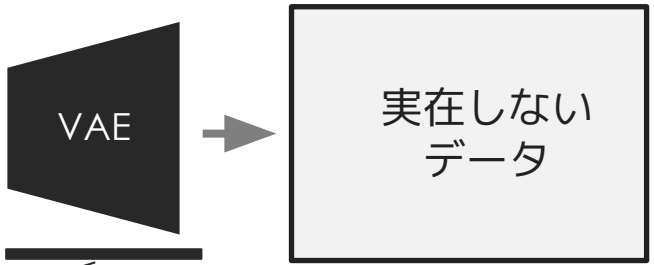
AE

もともと存在するデータを忠実に再現したい



VAE(Variational Auto Encoder)

実在しないデータを生み出したい



※それぞれのDecoderでモデルを生成している

© AVILEN Inc.

AEの問題点と対策

【前提】エンコーダとは「元のデータにどんな特徴があるか」を表す
潜在変数をもとにデータ分布を生成するモデル

問題点

AEはデータを忠実に再現しようとして「遊び」がないイメージ

対策

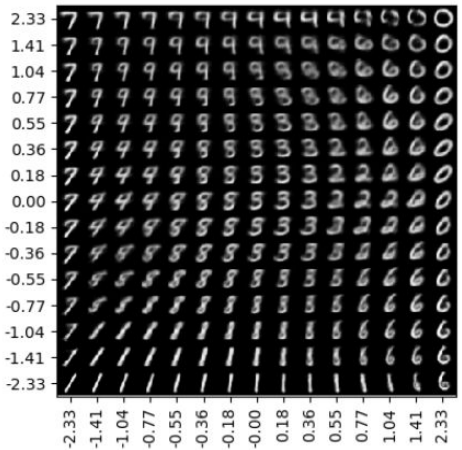
VAEは潜在変数に「ランダム性」と「連続性」を持たせている

© AVILEN Inc.

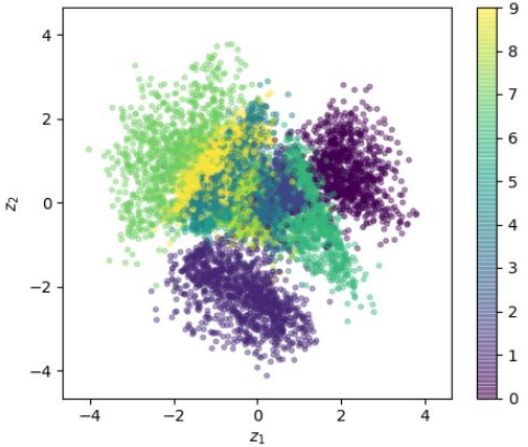
VAEの発想

ランダム性と連続性を持つ「正規分布」を潜在変数に導入

そのために潜在変数が正規分布に従うように調整する



潜在変数から生成された
mnist画像群

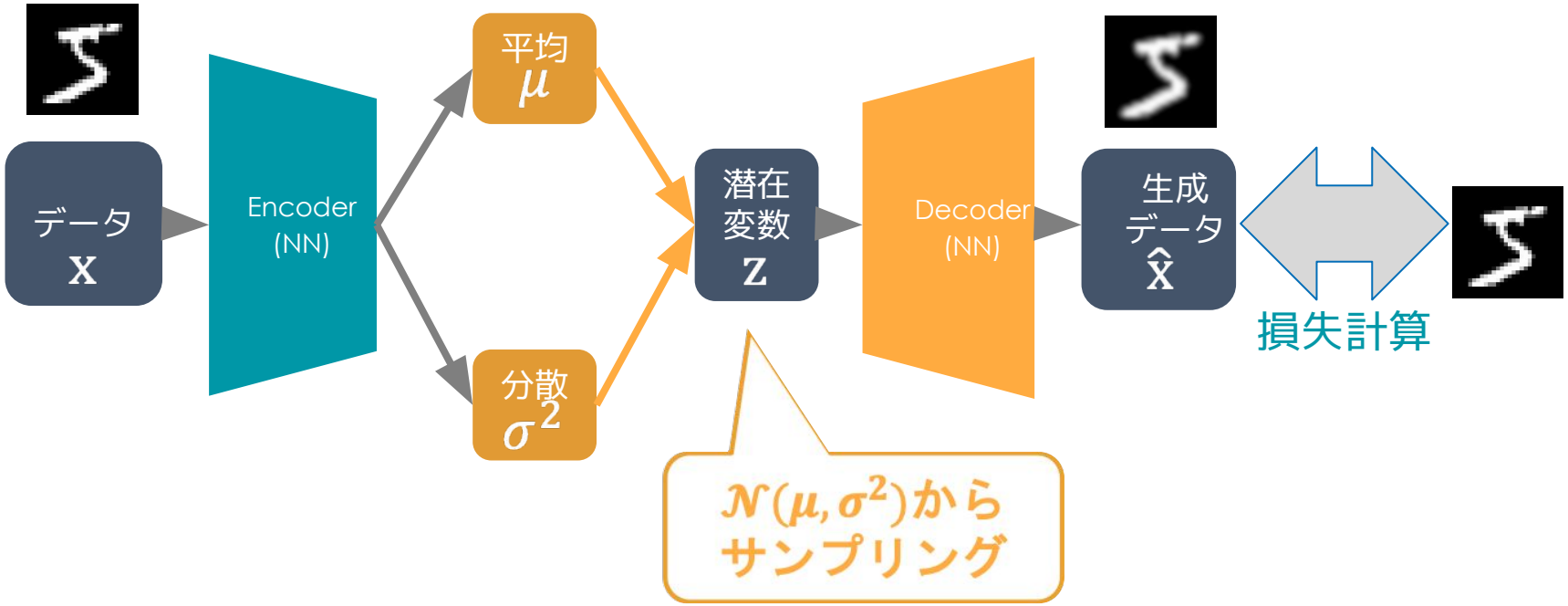


潜在空間

Implementing Variational Autoencoders in Keras: Beyond the Quickstart Tutorial

© AVILEN Inc.

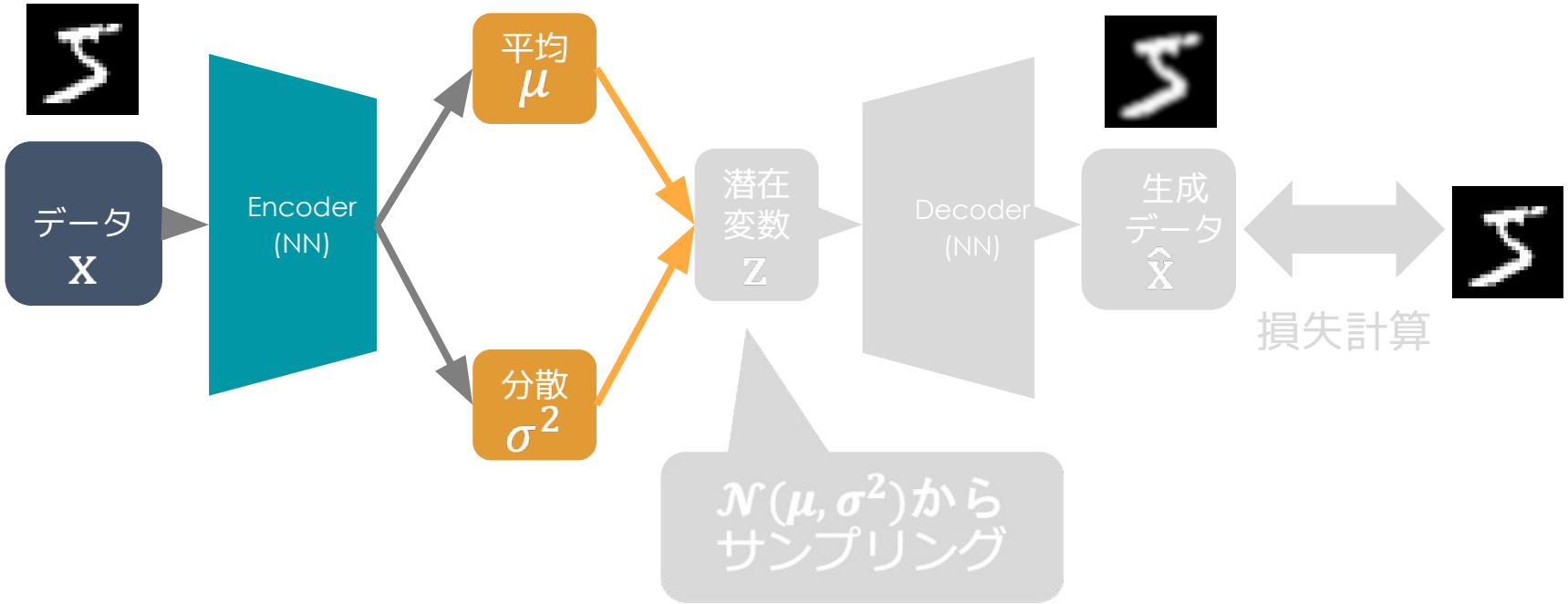
【学習時】VAEのアーキテクチャ(概観)



© AVILEN Inc.

【学習時】VAEのアーキテクチャ(1/4)

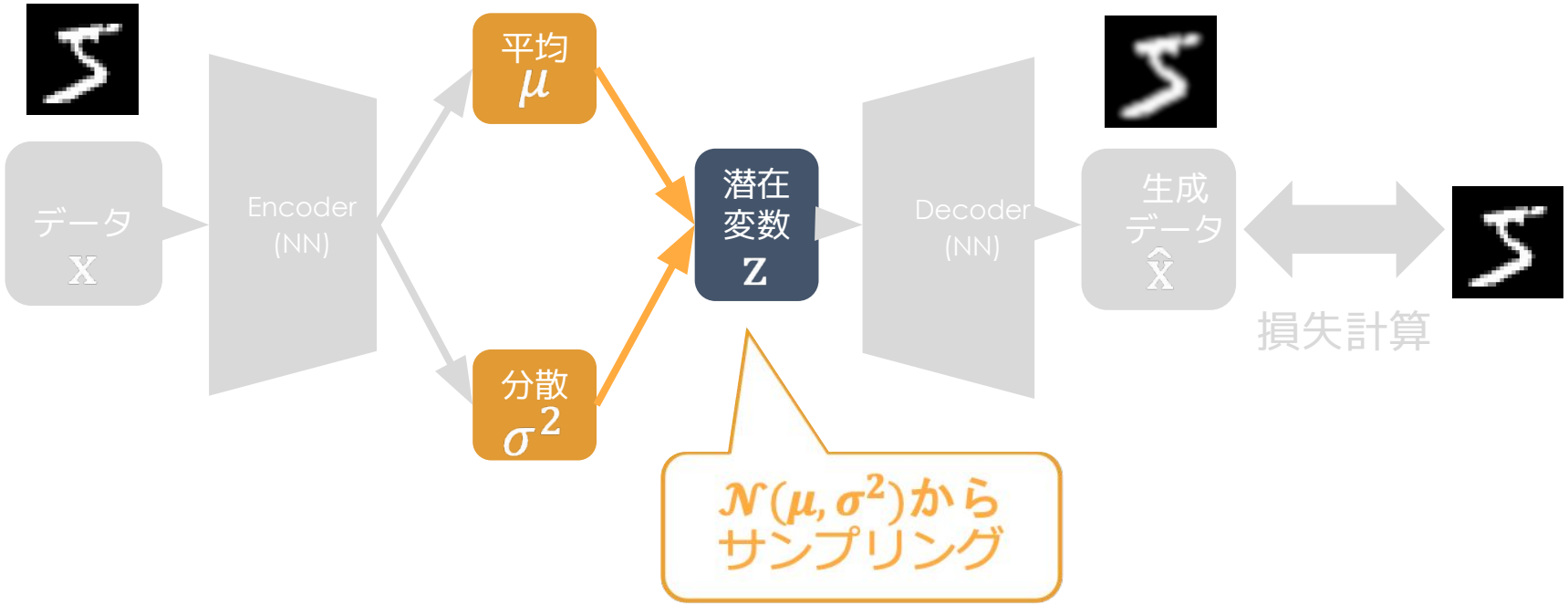
1. Encoderにより、データXから潜在変数Zの平均μと分散σ^2を出力



© AVILEN Inc.

【学習時】VAEのアーキテクチャ(2/4)

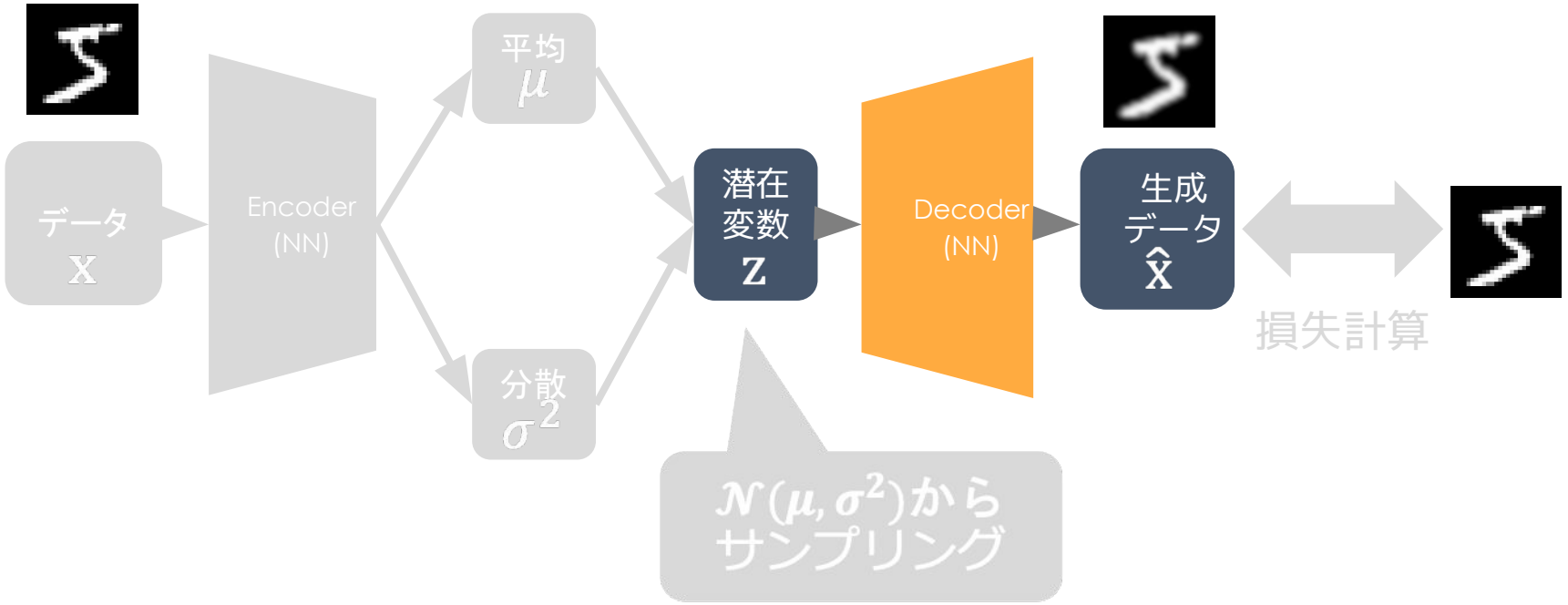
2. 出力された平均μと分散σ^2の正規分布に従う潜在変数Zをサンプリング



© AVILEN Inc.

【学習時】VAEのアーキテクチャ(3/4)

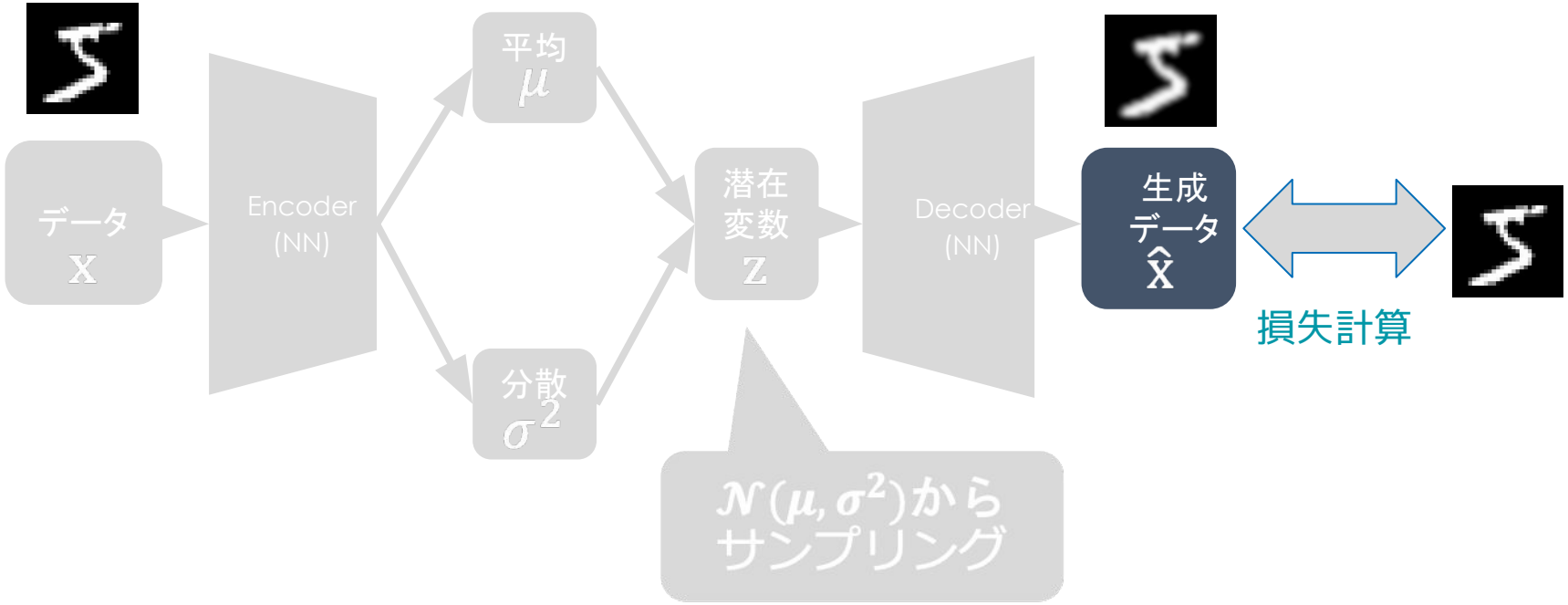
3. Decoderにより、潜在変数Zから新たなデータ$\hat{X}$を作成



© AVILEN Inc.

【学習時】VAEのアーキテクチャ(4/4)

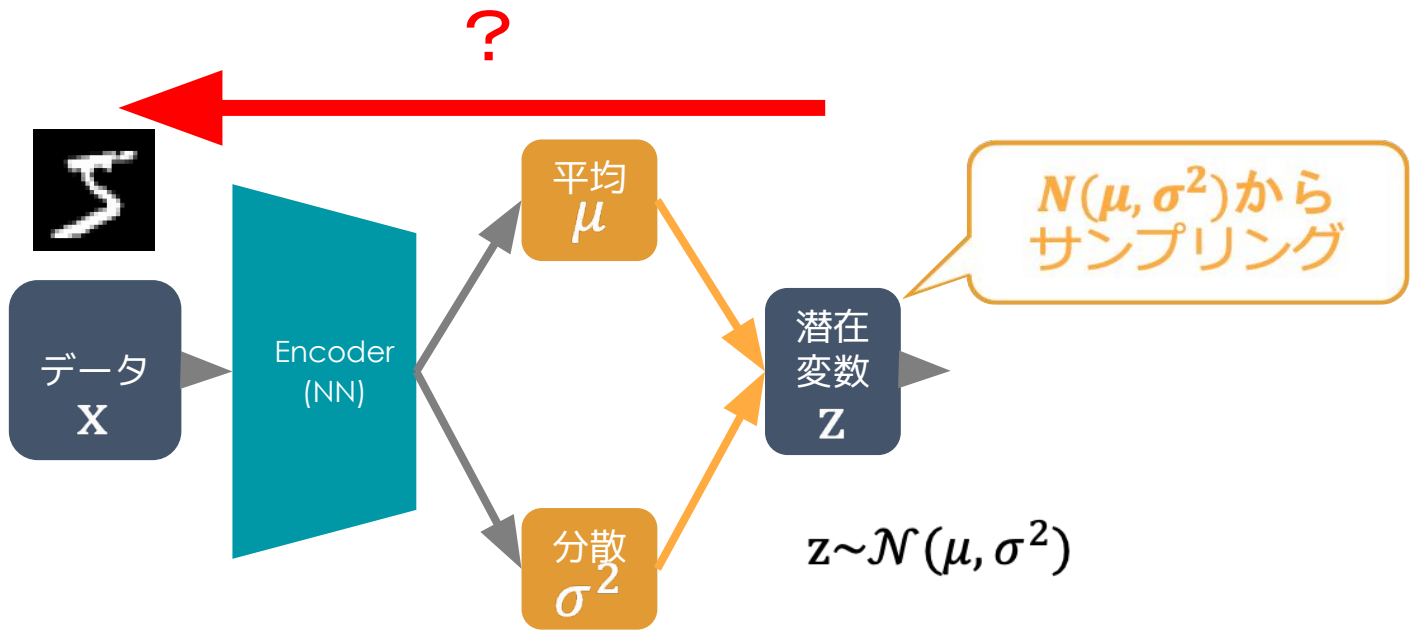
4. 後述の損失関数により、損失計算



© AVILEN Inc.

【補足】 Reparametrization Trick

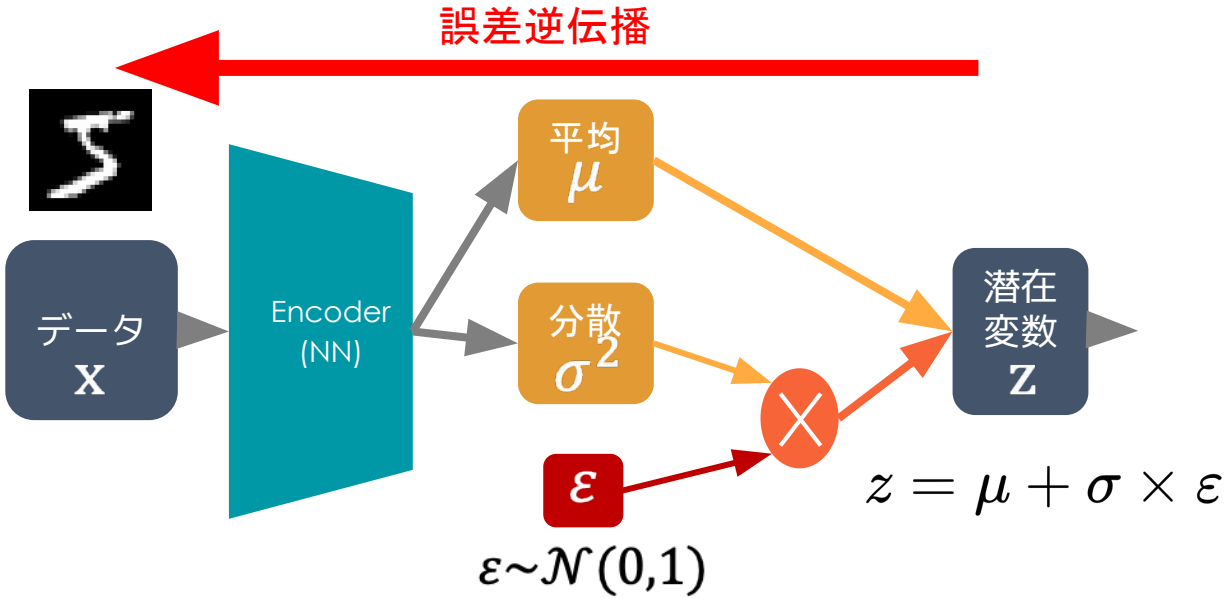
課題：潜在変数Zのランダム生成は、誤差逆伝播法ではどう表現できるのか？



© AVILEN Inc.

【補足】 Reparametrization Trick

$Z \sim \mathcal{N}(\mu, \sigma^2)$ のかわりにガウシアンノイズ $\varepsilon \sim \mathcal{N}(0, I)$ を使い、順伝播のときに得た ε の値を記憶しておけば誤差逆伝播が可能となる



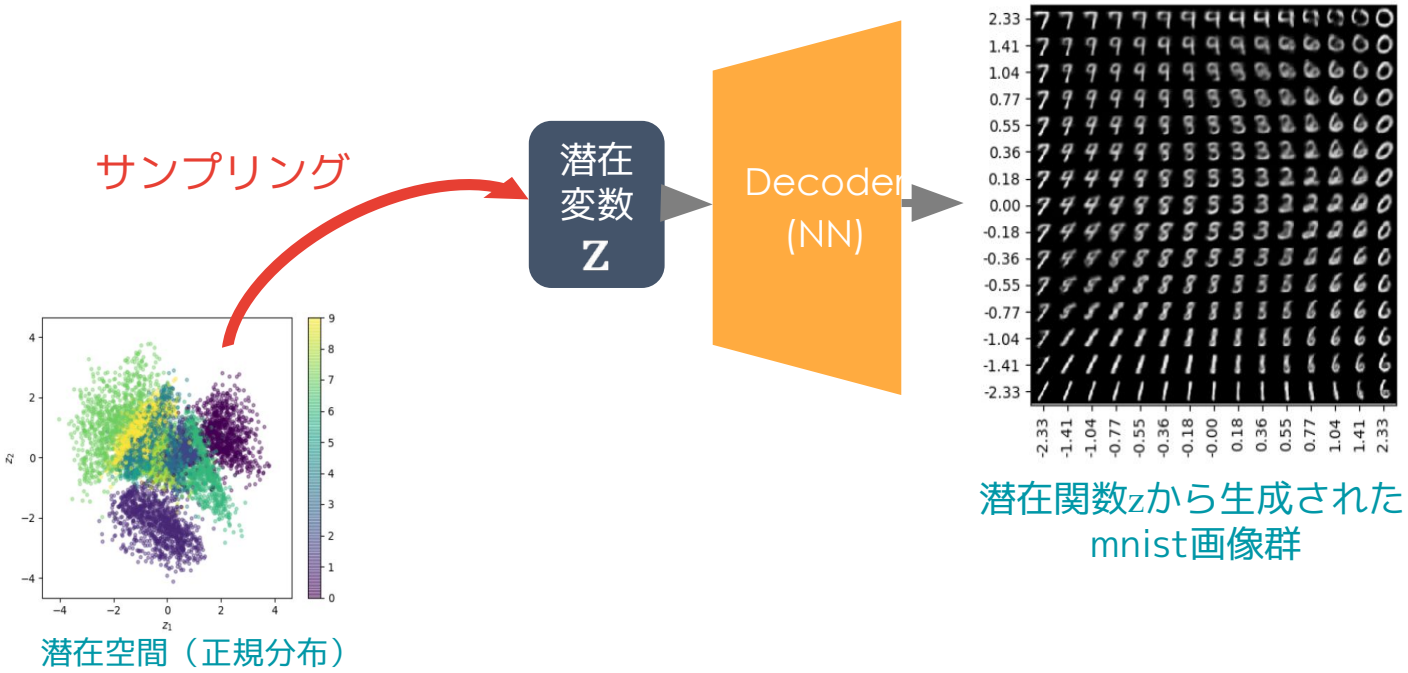
© AVILEN Inc.

【生成時】VAEのアーキテクチャ

1. 平均μと分散σ^2の正規分布に従う潜在変数Zをサンプリング
2. Decoderにより、潜在変数Zから新たなデータ$\hat{X}$を生成

© AVILEN Inc.

【生成時】VAEのアーキテクチャ



潜在空間（正規分布）

潜在関数zから生成された
mnist画像群

Implementing Variational Autoencoders in Keras: Beyond the Quickstart Tutorial

© AVILEN Inc.

VAEの損失関数

VAEでは、以下で与えられる変分下界 $\mathcal{L}$ を最大化することを考える

$$\begin{aligned}\mathcal{L} &= -D_{KL}[\mathcal{N}(\mu(\boldsymbol{x}), \Sigma(\boldsymbol{x}))||\mathcal{N}(0, I)] + \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] \\ &= -D_{KL}[\mathcal{N}(\mu(\boldsymbol{x}), \Sigma(\boldsymbol{x}))||\mathcal{N}(0, I)] + \beta\|\boldsymbol{Y} - \boldsymbol{X}\|^2\end{aligned}$$

したがって、最小化したい損失関数は以下のようになる

$$\begin{aligned}L &= -\mathcal{L} \\ &= D_{KL}\left[\mathcal{N}(\mu(\boldsymbol{X}), \Sigma(\boldsymbol{X}))||\mathcal{N}(0, I)\right] - \beta\|\boldsymbol{Y} - \boldsymbol{X}\|^2\end{aligned}$$

© AVILEN Inc.

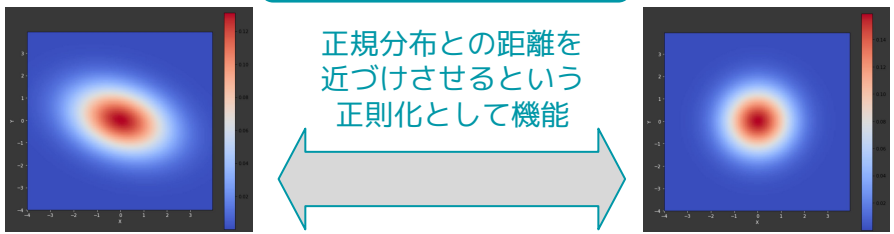
VAEの損失関数

$$
\begin{aligned}
L &= -\mathcal{L} \\
&= D_{KL}\left[q(z|\boldsymbol{X})||p(z)\right] - \mathbb{E}_{q(z|\boldsymbol{X})}\left[\log p(\boldsymbol{X}|z)\right] \\
&= D_{KL}\left[\mathcal{N}(\mu(\boldsymbol{X}), \Sigma(\boldsymbol{X}))||\mathcal{N}(0, I)\right] - \beta\|\boldsymbol{Y} - \boldsymbol{X}\|^2
\end{aligned}
$$

Encoderが求めた分布と
N(0,I)との近さ

入力データと
出力データの近さ(MSE)

第１項



正規分布との距離を
近づけさせるという
正則化として機能

Encoder出力

$\mathcal{N}(0, I)$

第２項

入力データ

出力データ

© AVILEN Inc.

VAEの課題：posterior collapse

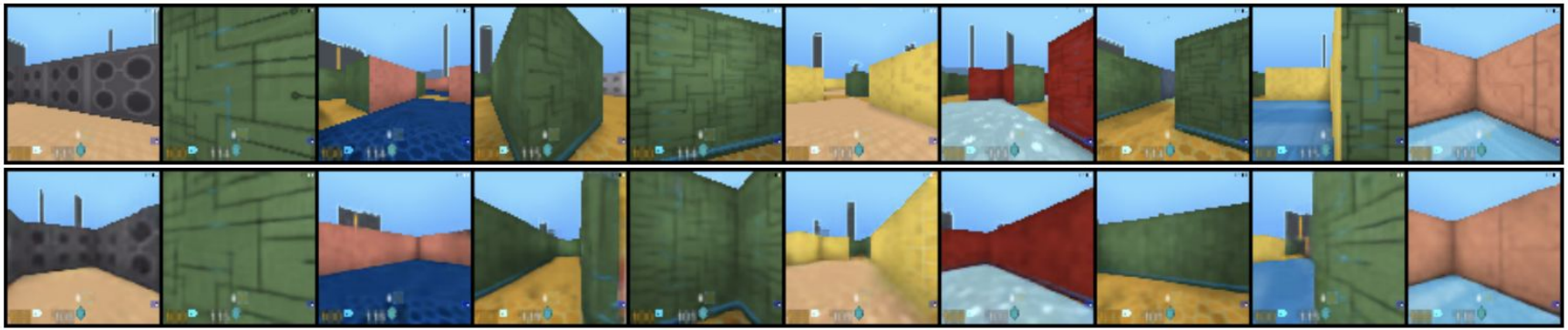
PixelCNNなどの高い表現力があるDecoderを使うときに、

潜在変数を無視した生成が行われてしまう現象

© AVILEN Inc.

VQ(Vector Quantised)-VAEによる解決策

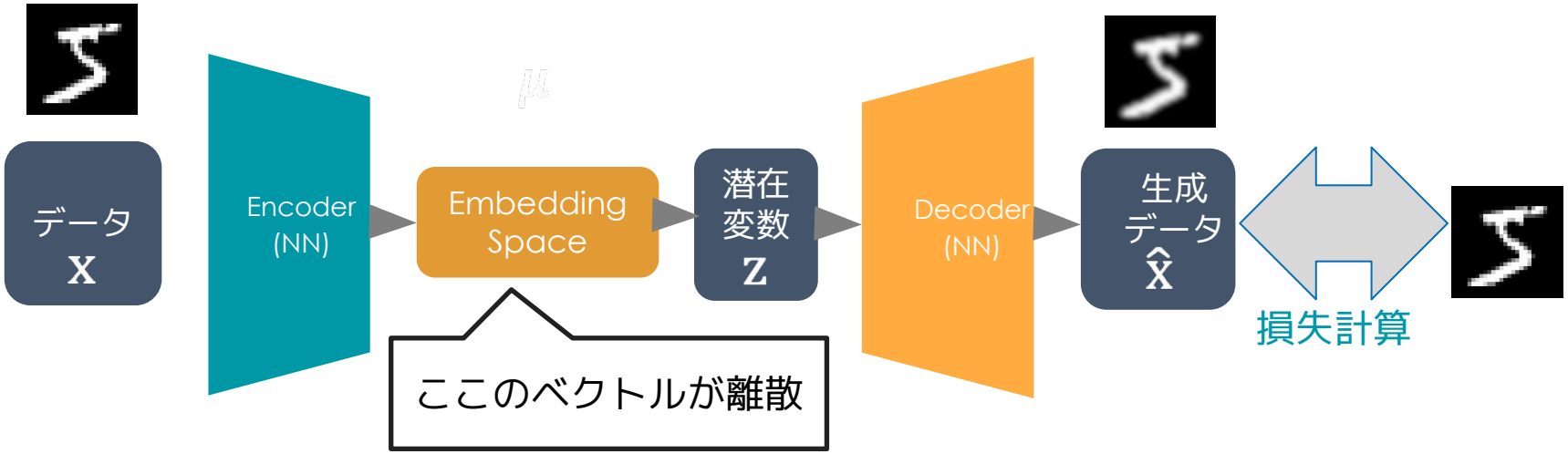
下図はVQ-VAEにPixelCNNを用いて画像を生成した時、ピクセル値そのものを
保存しているのではなく、潜在変数を考慮できていることを示している



上の行が実際の画像で、下の行がVQVAEの潜在変数を使って生成した画像
実際の画像に近いような画像が生成できていることから、
潜在変数が意味を持った変数であることがわかる

© AVILEN Inc.

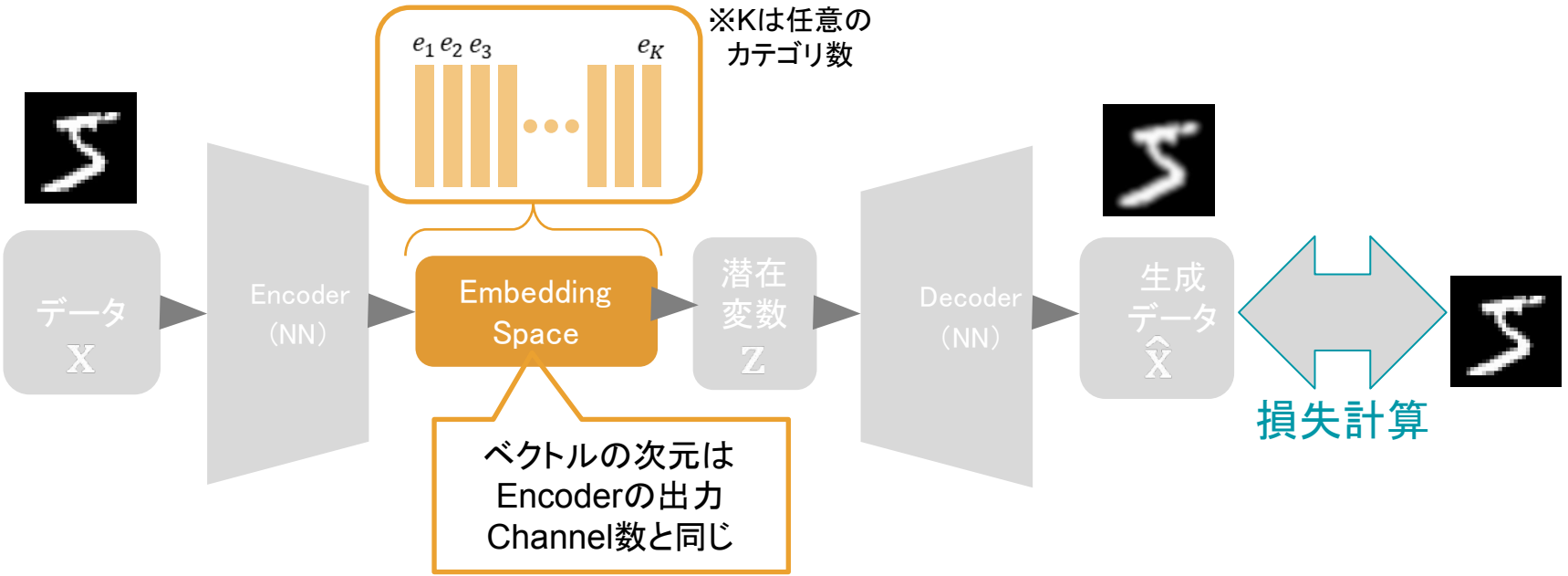
潜在変数をVAEの時とは異なり、離散的なベクトルとして扱う
本来文章や画像は離散的で、犬が猫に変化することはないため、
離散的なベクトルで扱うことが本質である
VQ-VAEは離散を扱うにも関わらず、学習可能としたアーキテクチャである



© AVILEN Inc.

【学習時】VQ-VAEのアーキテクチャ

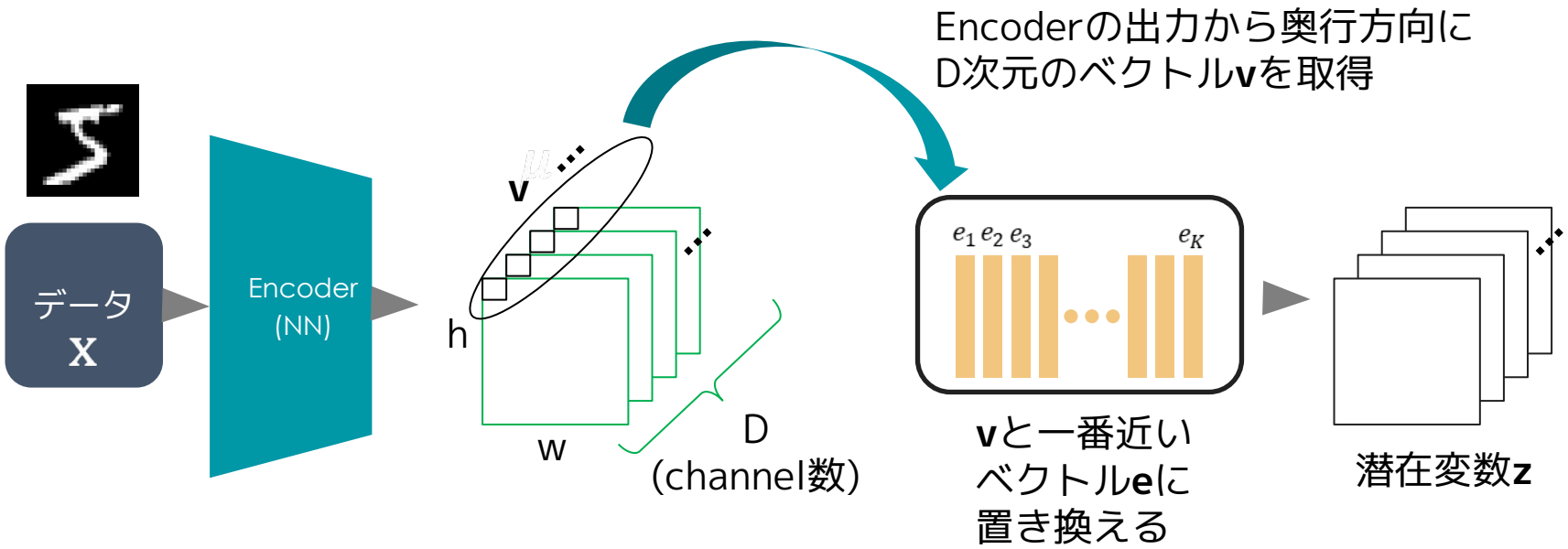
1. Embedding Spaceを設定



© AVILEN Inc.

【学習時】VQ-VAEのアーキテクチャ

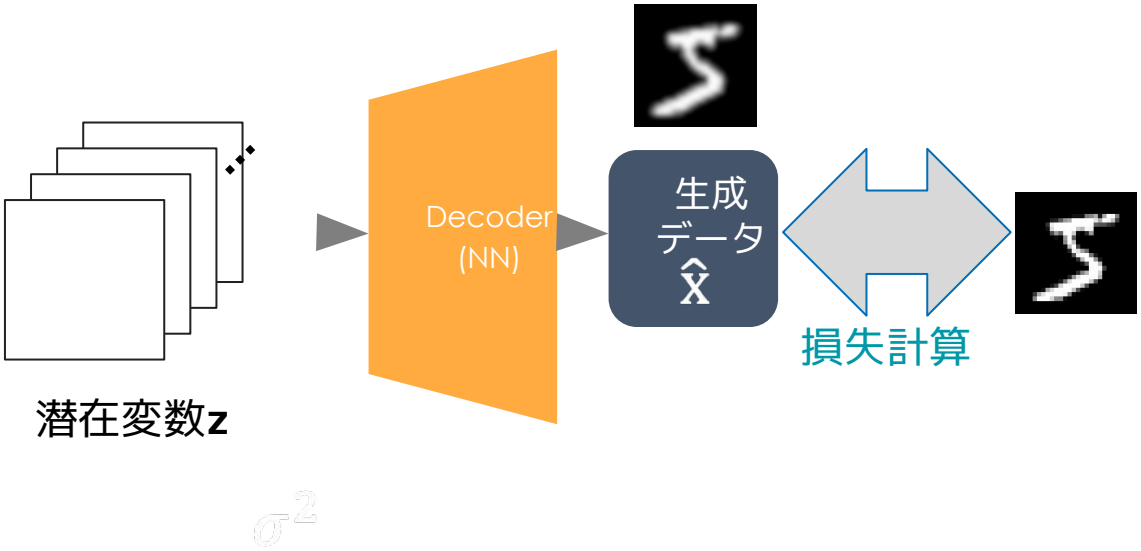
2. Encoderによって得られたベクトルとの距離が最も近いEmbedding Space内の離散ベクトルが潜在変数として得られる（これをh×w個やる）



© AVILEN Inc.

潜在変数の離散化モデリング

3. Decoderにより、潜在変数から新たなデータ$\hat{x}$を作成



© AVILEN Inc.

VQ-VAEの損失関数

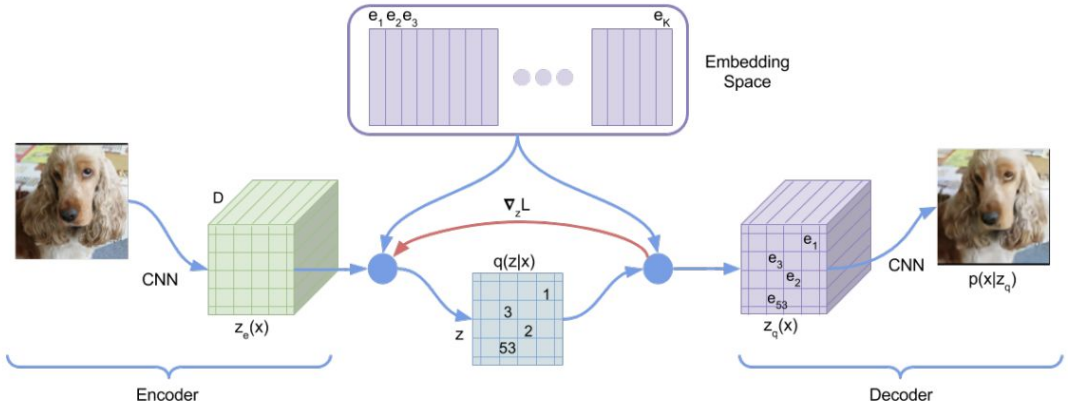
$$L = \log p(x|z_q(x)) + ||\mathrm{sg}[z_e(x)] - e||_2^2 + \beta||z_e(x) - \mathrm{sg}[e]||_2^2$$

入力データと
出力データの近さ

潜在変数ベクトルをエンコーダ出力$z_e(x)$に近づける

エンコーダ出力$z_e(x)$を潜在変数ベクトルに近づける

※$sg()$はstop gradientを意味し,勾配更新を行わないことを意味する



出典：https://arxiv.org/pdf/1711.00937.pdf

© AVILEN Inc.

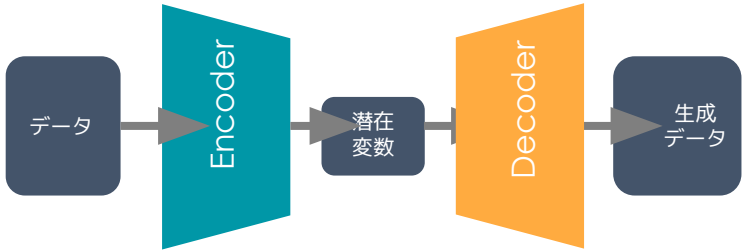
AE系まとめ

AE

損失関数

$$\mathcal{L} = \beta||\boldsymbol{Y} - \boldsymbol{X}||^2$$

アーキテクチャ

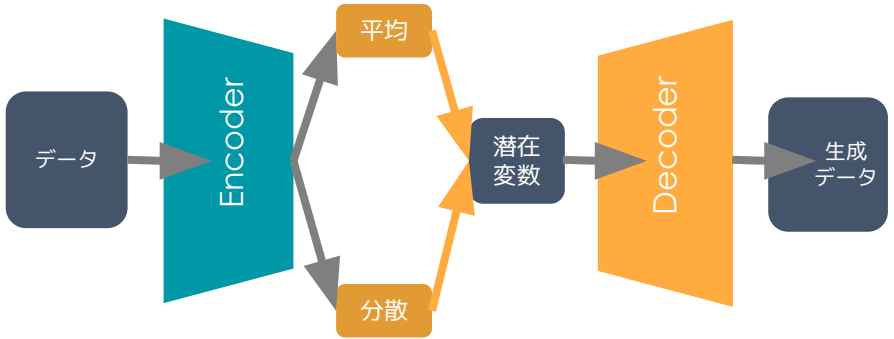


VAE

損失関数

$$\mathcal{L} = D_{KL}[\mathcal{N}(\mu(\boldsymbol{X}), \Sigma(\boldsymbol{X}))||\mathcal{N}(0, I)] + \beta||\boldsymbol{Y} - \boldsymbol{X}||^2$$

アーキテクチャ



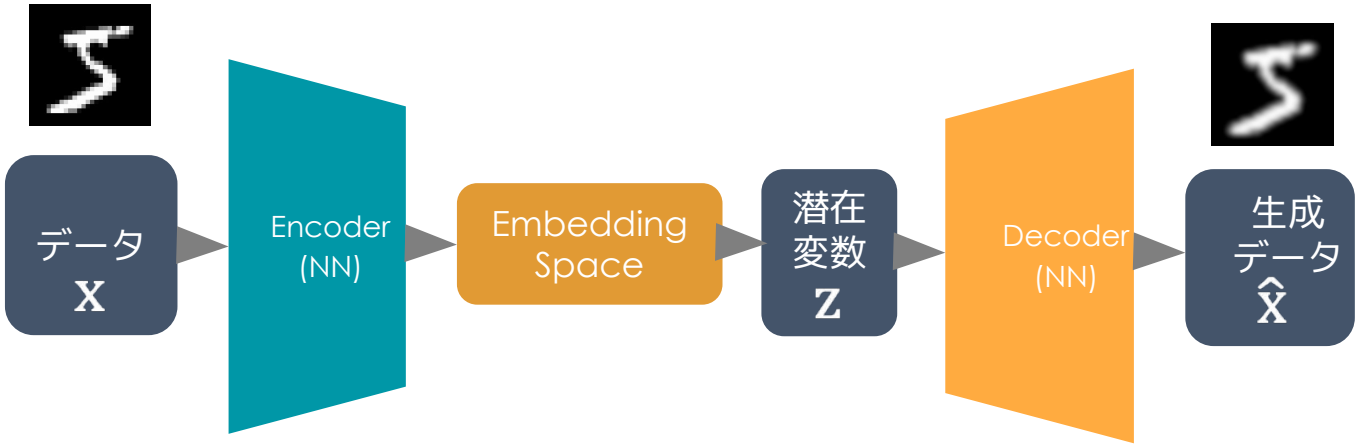
© AVILEN Inc.

VQ-VAE

損失関数

$$L = \log p(x|z_q(x)) + ||\mathrm{sg}[z_e(x)] - e||_2^2 + \beta||z_e(x) - \mathrm{sg}[e]||_2^2$$

アーキテクチャ



© AVILEN Inc.

【Chapter07】生成モデル

GAN

VAE実用と課題

- VAEは理論がしっかり整っており、実装が簡単な上、好まれる
- しかし、画像を入力として訓練したVAEのサンプルはややぼやける傾向にあり、その原因はまだよくわかっていない

  - [原因1] D_{KL}の最小化において、ぼやけた画像に高い確率を与えることが挙げられる。ガウス分布による符号化は、わずかなピクセルの変化しかもたらさない入力特徴量を無視する傾向があることが理由。
  - [原因2]鮮明に出力する画像よりも、全体的にぼやかした画像のほうが「入力画像と出力画像の差(MSE)」が下がることが挙げられる。

© AVILEN Inc.

VAE課題と対策

VAEで生成した画像はややぼやける

- [原因1]ガウス分布による符号化により、データに対して制約をかけているために出力画像にノイズが生まれるのではないか？
- [原因2]鮮明に出力する画像よりも、全体的にぼやかした画像の方が「入力画像と出力画像の差(MSE)」が下がることが挙げられる

より鮮明な画像を生成するためには…

- [原因1]の対策：ノイズの除去
- [原因2]の対策：MSEをやめる

→ 今までとは全く異なるアプローチが必要

© AVILEN Inc.

GAN(Generative Adversarial Network)

敵対的生成ネットワーク

Generative Adversarial Network

生成　　敵対的

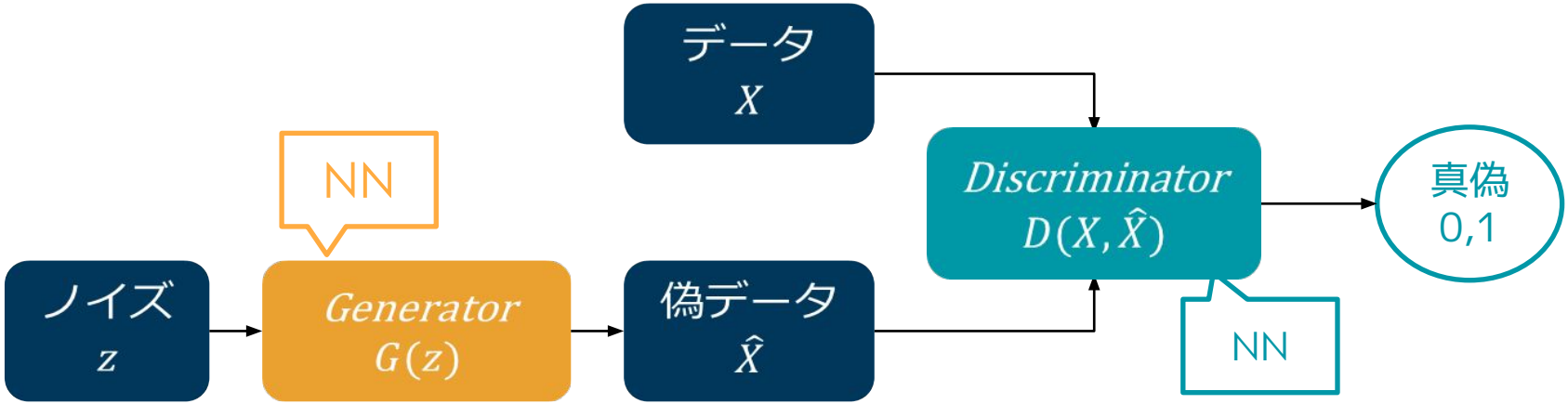
Discriminatorにバレないように
訓練データそっくりの画像を生成する

Generatorが生成したサンプルか、訓練データ
として与えられたサンプルかを識別する

© AVILEN Inc.

【訓練時】GANのアーキテクチャ

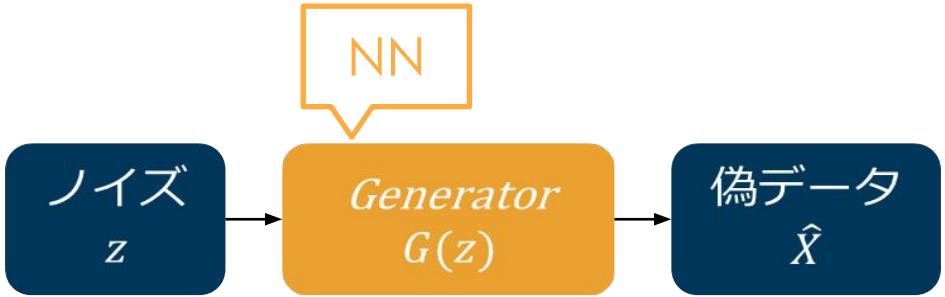
1. 毎回同じものを生成しないように、一様乱数からノイズzをサンプリング
2. Generatorによって偽のデータを生成
3. 混ぜられた観測データ（真）と生成データ（偽）を、Discriminatorが識別
4. Discriminatorは審議を判別できるように、
   GeneratorはDiscriminatorを騙すように学習する



© AVILEN Inc.

【生成時】GANのアーキテクチャ

1. 一様乱数からノイズzをサンプリング
2. **Generator**によって新たなデータを生成する



© AVILEN Inc.

学習目標

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

$D(x)$: Dが予測した、訓練データxが実在データである確率
$G(z)$: ノイズ z を入力としてGが生成するデータ

D(Discriminator)は訓練データxと生成データG(z)に対して、
正しくラベル付けを行う確率を**最大化**しようとする

G(Generator)はlog(1-D(G(z)))を**最小化**しようとする
つまり、G自身が生成したデータG(z)をDに本物だと思わせる

© AVILEN Inc.

DはV(D,G)を最大化しようとする

$$\min_G \max_D V(D,G) = \underbrace{\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)]}_{\text{第❶項}} + \underbrace{\mathbb{E}_{z\sim p_z(z)}\left[\log\left(1-D\left(G(z)\right)\right)\right]}_{\text{第❷項}}$$

$D(x)$: Dが予測した、訓練データxが実在データである確率
$G(z)$: ノイズ z を入力としてGが生成するデータ

第①項：訓練データ分布 $p_{data}(x)$から得られたデータ x が「訓練データである」と判断する確率 $D(x)$(の自然対数の期待値)を最大化しようとしている

第②項：生成データ分布$p_z(z)$から得られたデータ $G(z)$ が「生成データである」と判断する確率 $(1-D(G(z)))$ (の自然対数の期待値)を最大化しようとしている

© AVILEN Inc.

GはV(D,G)を最小化しようとする

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

第❶項　　第❷項

$D(x)$: Dが予測した、訓練データxが実在データである確率
$G(z)$: ノイズ z を入力としてGが生成するデータ

第①項：Dが訓練データをどう判断するかは、Gには関係ない

第②項：生成データ$p_z(z)$から得られたデータ$G(z)$が「生成データである」とDに判断させる確率 $(1 - D(G(z)))$(の自然対数の期待値) を最小化しようとしている

© AVILEN Inc.

m：ミニバッチひとまとまりのサイズ

1-1. m個のノイズzとデータxをサンプリング

1-2. 確率的勾配降下法でDiscriminatorを更新

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$

2-1. m個のノイズzをサンプリング

2-2. 確率的勾配降下法でGeneratorを更新

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

3. 1-1～2-2を全てのミニバッチに対して繰り返す

© AVILEN Inc.

GANは学習が困難

問題	対策
Discriminator(D)とGenerator(G)が拮抗しなければいけないのに、Dが圧勝して勾配消失する	• Dは小さいネットワークにする • DのDropout rateを大きめに • Unrolled GAN
mode collapse	• Minibatch Discrimination • Wasserstein GAN

© AVILEN Inc.

Unrolled GAN：

GよりDの方が学習が進みすぎてしまう問題に対する解決策の1つ。

Gを1ステップ学習させる際、あらかじめKステップ学習させたDのパラメータを用いる。

これによりGはKステップ分の勾配情報を先取りして学習したことになる。

効果：Generatorに「質のいい勾配情報(hint)」を与える

⇒学習のバランスが取りやすくなる

© AVILEN Inc.

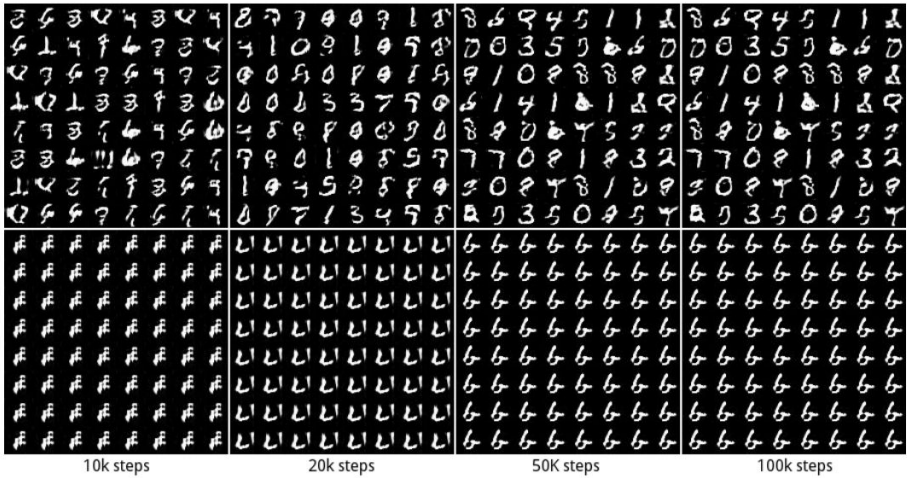
【問題点2】モード崩壊(mode collapse)

mode collapse：

Generatorの能力が不足していて全体の分布を近似しきれないので、
苦肉の策として、ある1つのデータ(最頻値=mode)だけを出力しようとしている状態

対策：

- Minibatch Discrimination
- Wasserstein GAN



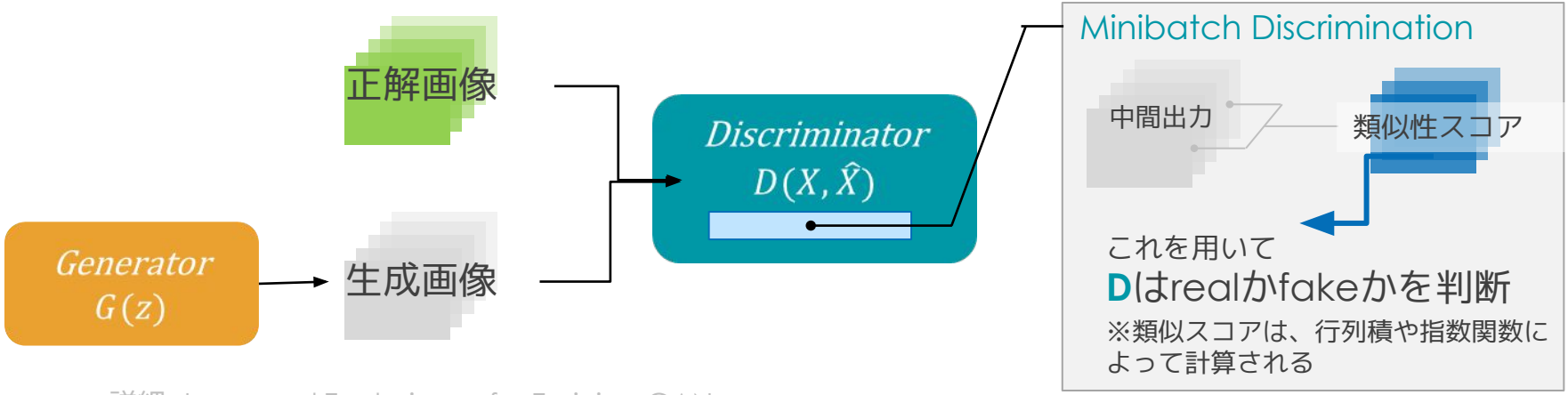
UNROLLED GENERATIVE ADVERSARIAL NETWORKS

© AVILEN Inc.

Minibatch Discrimination

【mode collapse への対応策１】

Discriminatorに「mode collapseが起きているか」を
見分けやすくするためのヒント💡を与える処理
具体的には、同バッチ内の複数の画像同士の相似性を表すスコアを用いる



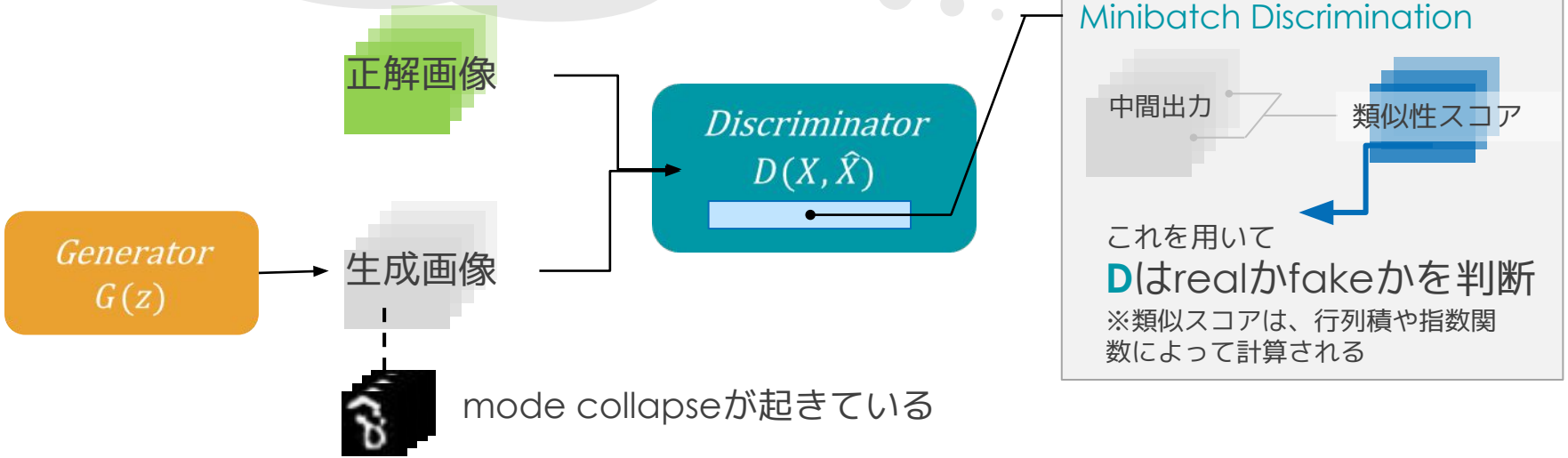
詳細: Improved Techniques for Training GANs

© AVILEN Inc.

Minibatch Discriminationの具体例

Minibatch Discriminationは、Discriminatorにヒントを与える

これは生成された画像(fake)の可能性が高い！
なぜなら「同バッチ内の画像同士の類似度が高く、mode collapseが起きている」と推測できるから



© AVILEN Inc.

➢ Gの目的は訓練データの確率分布に近づいていくこと

▼ mode collapse への対応策 2

通常のGAN

JSD(イェンセンシャノンダイバージェンス)を使って近づいていく

➔ 分布の裾が狭いときに、全く異なる形状の分布同士でもJSDは低い値となってしまう問題がある

改良

Wasserstein GAN

EMDという距離を使う
EMD: 分布を砂山とみなしたときに、別の分布(砂山)に移動するためには「どれくらいの量の砂を運ばなければならないか」を数値化した指標

✓ EMDならばある分布と異なる形状の分布間の距離は大きいと判断する
→ Gの目的を補助してくれる

© AVILEN Inc.

JSDとは

通常のGANはJSD(イェンセンシャノンダイバージェンス)を最小にすることで、真のデータ分布を学習する
以下例では P_r、P_g をそれぞれ確率分布とする

$$JSD(P_r||P_g) = \frac{1}{2}KL(P_r||P_A) + \frac{1}{2}KL(P_g||P_A)$$

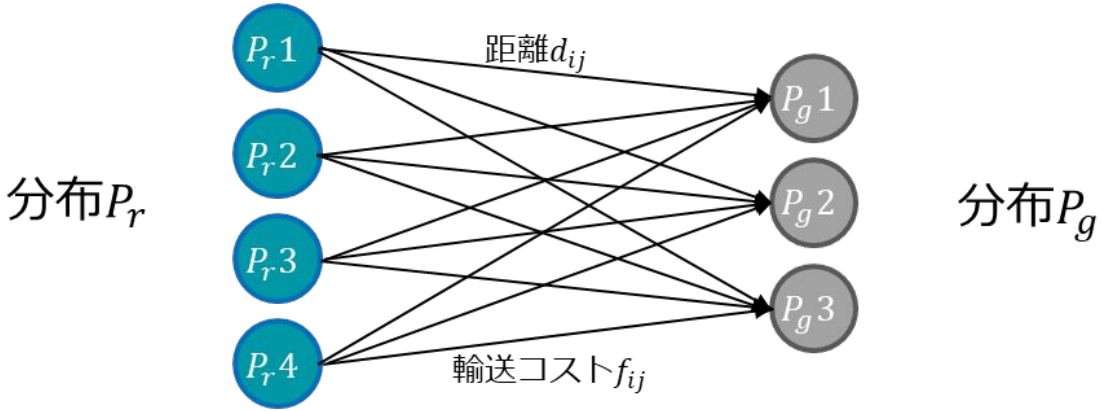
$$P_A = \frac{P_r + P_g}{2}$$

© AVILEN Inc.

EMDとは

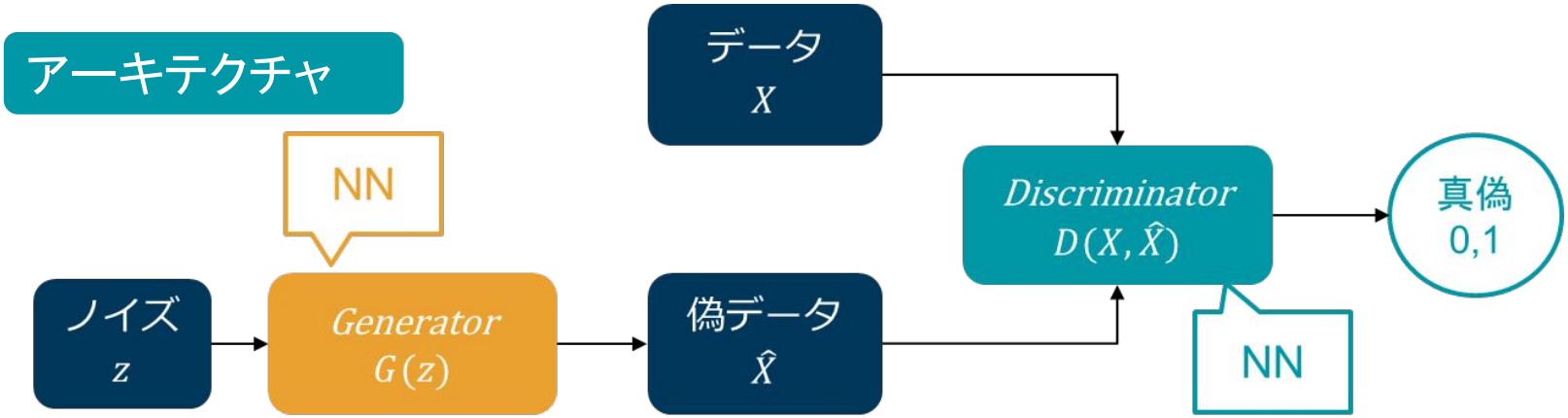
EMD(Earth Mover's Distance)は下の式で定義されており、
直感的にはxからyにどれだけの「質量」を最適に輸送するコストと考えられる

$$EMD(P_r||P_g) = \inf_{\gamma \in (P_r \times P_g)} E_{(x,y)\sim\gamma}[||x-y||]$$



© AVILEN Inc.

GANまとめ



損失関数

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\left[\log D(x)\right] + \mathbb{E}_{x\sim p_z(x)}\left[\log(1 - D(G(z)))\right]$$

$D(\boldsymbol{x})$: データ $\boldsymbol{x}$ が訓練データである確率
$G(\boldsymbol{z})$: Gはノイズ $\boldsymbol{z}$ を入力として生成されるデータ

D(Discriminator)は訓練データ$\boldsymbol{x}$と生成データ$G(\boldsymbol{z})$に対して、正しくラベル付けを行う確率を最大化しようとする

G(Generator) は$\log(1 - D(G(z)))$を最小化しようとする

© AVILEN Inc.

この章で学習したこと

1. 生成モデルの考え方
2. 潜在変数
3. エンコーダ・デコーダモデル
4. AE
5. VAE
6. GAN

© AVILEN Inc.

AVILEN