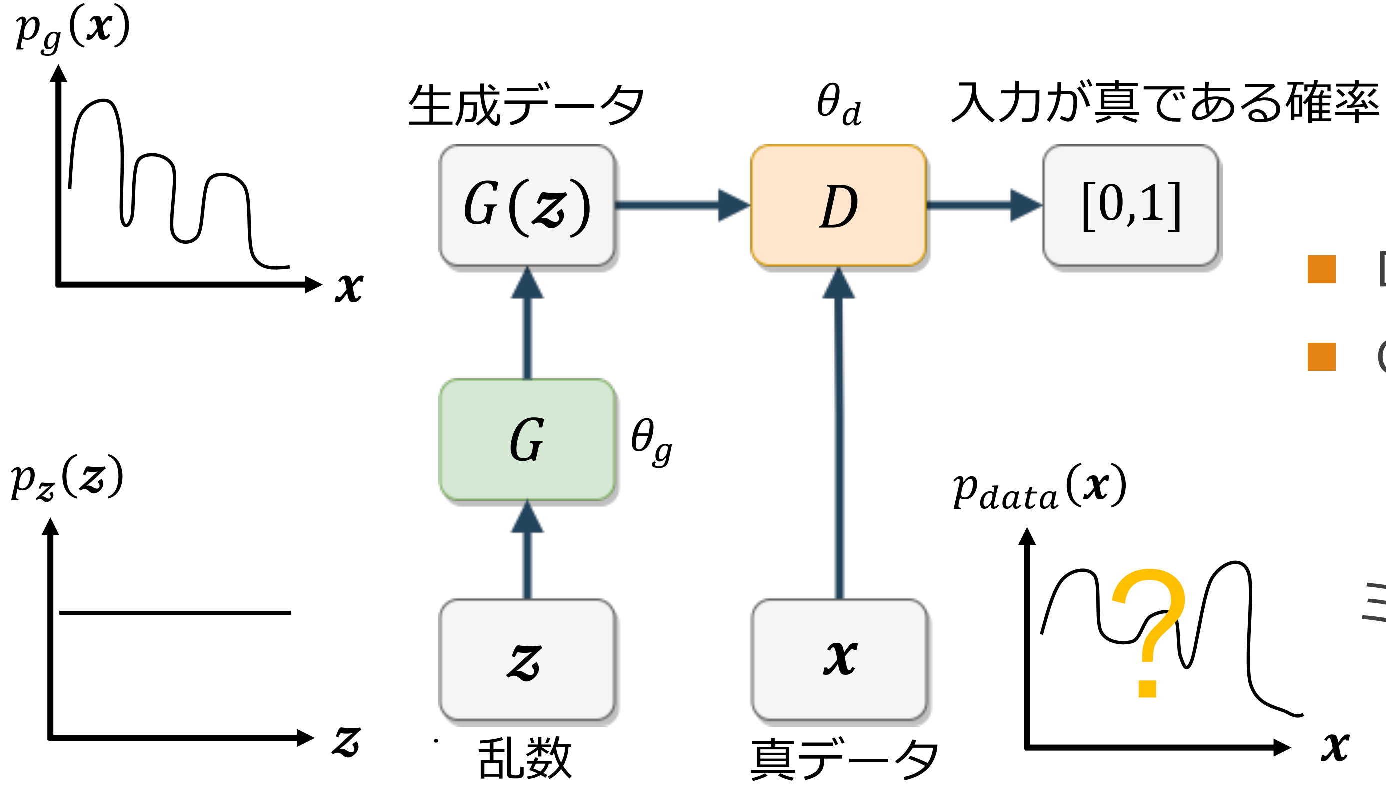
数式とソースコードによるDCGANの解説

目次

- GANについて
  - GANの構造
  - ミニマックスゲームと価値関数
  - GANの最適化方法
  - 本物のようなデータを生成できる理由
- DCGANについて
  - 具体的なネットワーク構造
- 応用技術の紹介
  - 概要

GAN(Generative Adversarial Nets)とは[1]

- 生成器と識別器を競わせて学習する生成&識別モデル
  - Generator: 乱数からデータを生成
  - Discriminator: 入力データが真データ(学習データ)であるかを識別



- D: 正しく判別したい
- G: Dに誤判断させたい

2プレイヤーの
ミニマックスゲーム

[1] I. Goodfellow, *et al.*, NIPS, 2014.

2プレイヤーのミニマックスゲームとは？

- 1人が自分の勝利する確率を最大化する作戦を取る
- もう一人は相手が勝利する確率を最小化する作戦を取る

- GANでは価値関数Vに対し, Dが最大化, Gが最小化を行う.

$$\min_G \max_D V(D, G)$$

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

- バイナリークロスエントロピーと似ている？

$$L = -\sum y \log \hat{y} + (1 - y) \log(1 - \hat{y})$$

GANの価値関数はバイナリークロスエントロピー

- 単一データのバイナリークロスエントロピー

$$L = -y\log\hat{y} + (1-y)\log(1-\hat{y})$$

y: 真値（ラベル）
$\hat{y}$: 予測値（確率）

- 真データを扱う時: $y = 1,\ \hat{y} = D(x)$ ➡ $L = -\log[D(\boldsymbol{x})]$
- 生成データを扱う時: $y = 0,\ \hat{y} = D(G(\boldsymbol{z}))$ ➡ $L = -\log[1 - D(G(\boldsymbol{z}))]$

- 2つを足し合わせる

$$L = -\left(\log[D(\boldsymbol{x})] + \log[1 - D(G(\boldsymbol{z}))]\right)$$

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

- 複数データを扱うために期待値を取る
- 期待値: 何度も試行する際の平均的な結果値 $\sum xp(x)$

最適化方法

- Generatorのパラメータθ_gを固定
  - 真データと生成データをm個ずつサンプル
  - θ_dを勾配上昇法(Gradient Ascent)で更新

$$\frac{\partial}{\partial \theta_d} \frac{1}{m} \left[\log[D(\boldsymbol{x})] + \log\left[1 - D\left(G(\boldsymbol{z})\right)\right]\right]$$

θ_dをk回更新

- Discriminatorのパラメータθ_dを固定
  - 生成データをm個ずつサンプル
  - θ_gを勾配降下法(Gradient Descent)で更新

$$\frac{\partial}{\partial \theta_g} \frac{1}{m} \left[\log\left[1 - D\left(G(\boldsymbol{z})\right)\right]\right]$$

θ_gを1回更新

なぜGeneratorは本物のようなデータを生成するのか？

- 生成データが本物とそっくりな状況とは
  - $p_g = p_{data}$であるはず

- 価値関数が$p_g = p_{data}$の時に最適化されていることを示せばよい

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

- 二つのステップにより確認する

1. Gを固定し、価値観数が最大値を取るときの$D(\boldsymbol{x})$を算出
2. 上記の$D(\boldsymbol{x})$を価値関数に代入し、 Gが価値観数を最小化する条件を算出

ステップ1: 価値関数を最大化する$D(\boldsymbol{x})$の値は?

➢Generatorを固定

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

$$= \int_{\boldsymbol{x}} p_{data}(\boldsymbol{x})\log\left(D(\boldsymbol{x})\right)dx + \int_{\boldsymbol{z}} p_{z}(\boldsymbol{z})\log\left(1 - D\left(G(\boldsymbol{z})\right)\right)dz$$

$$= \int_{\boldsymbol{x}} \underline{p_{data}(\boldsymbol{x})\log\left(D(\boldsymbol{x})\right) + p_{g}(\boldsymbol{x})\log\left(1 - D(\boldsymbol{x})\right)}dx$$

$\mathrm{y} = D(\boldsymbol{x}), a = p_{data}(\boldsymbol{x}), b = p_{g}(\boldsymbol{x})$ と置けば

$$a\log(y) + b\log(1-y)$$

➢$a\log(y) + b\log(1-y)$の極値を求めよう

ステップ1: 価値関数を最大化する$D(\boldsymbol{x})$の値は?

- $a\log(y) + b\log(1-y)$をyで微分

$$\frac{a}{y} + \frac{b}{1-y}(-1) = 0$$

$$\frac{a}{y} = \frac{b}{1-y}$$

$$a - ay = by$$

$$a = (a+b)y$$

$$y = \frac{a}{a+b}$$

- $y = D(\boldsymbol{x}), a = p_{data}(\boldsymbol{x}), b = p_g(\boldsymbol{x})$なので

$$D(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

- 価値関数が最大値を取るときの$D(\boldsymbol{x})$が判明

ステップ2: 価値関数はいつ最小化するか?

- 価値関数の$D(\boldsymbol{x})$を$\frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x})+p_g(\boldsymbol{x})}$で置き換え

$$V = \mathbb{E}_{\boldsymbol{x}\sim p_{data}} \log\left[\frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x})+p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g} \log\left[1 - \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x})+p_g(\boldsymbol{x})}\right]$$

$$= \mathbb{E}_{\boldsymbol{x}\sim p_{data}} \log\left[\frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x})+p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g} \log\left[\frac{p_g}{p_{data}(\boldsymbol{x})+p_g(\boldsymbol{x})}\right]$$

- 二つの確率分布がどれぐらい近いのか調べる必要がある
  - 有名な指標としてJSダイバージェンスがある
  - $JS(p_1 \parallel p_2) = \frac{1}{2}\left(\mathbb{E}_{\boldsymbol{x}\sim p_1} \log\left(\frac{2p_1}{p_1+p_2}\right) + \mathbb{E}_{\boldsymbol{x}\sim p_2} \log\left(\frac{2p_2}{p_1+p_2}\right)\right)$
  - JSダイバージェンスは非負で、分布が一致する時のみ0の値を取る

ステップ2: 価値関数はいつ最小化するか?

- 価値関数を変形

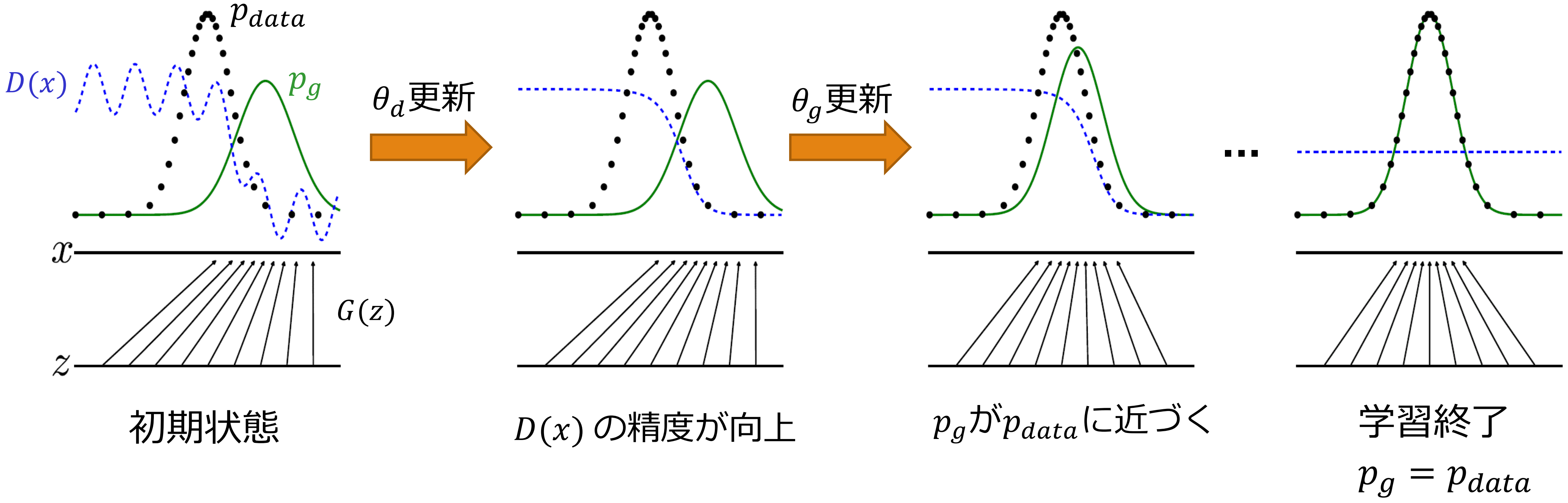
$$V = \mathbb{E}_{\boldsymbol{x}\sim p_{data}} \log\left[\frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x})+p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g} \log\left[\frac{p_g}{p_{data}(\boldsymbol{x})+p_g(\boldsymbol{x})}\right]$$

$$= \mathbb{E}_{\boldsymbol{x}\sim p_{data}} \log\left[\frac{2p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x})+p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g} \log\left[\frac{2p_g}{p_{data}(\boldsymbol{x})+p_g(\boldsymbol{x})}\right] - 2\log 2$$

$$= 2JS\left(p_{data} \parallel p_g\right) - 2\log 2$$

- $\min_G V$は$p_{data} = p_g$のときに最小値となる ($-2\log 2 \approx -1.386$)
- GANの学習によりGは本物のようなデータを生成できる

学習ステップの可視化



p_{data}
$D(x)$
p_g
θ_d更新
θ_g更新
…
x
z
$G(z)$
初期状態
$D(x)$ の精度が向上
p_gがp_{data}に近づく
学習終了
$p_g = p_{data}$

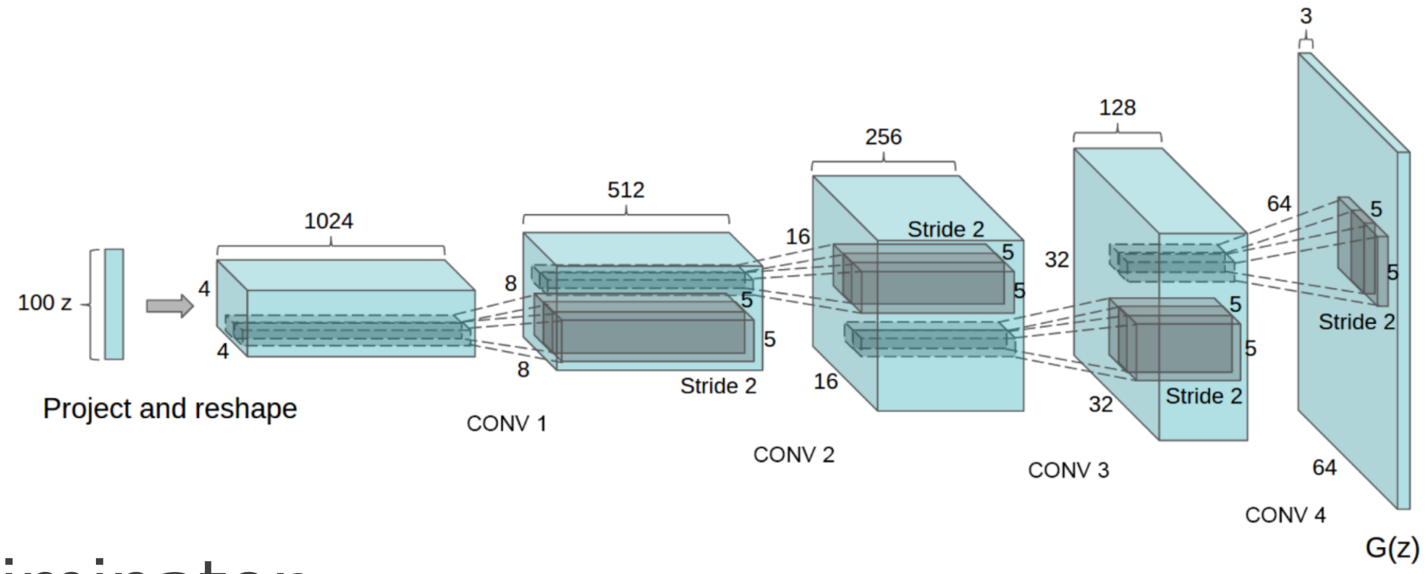
DCGAN(Deep Convolutional GAN)とは[2]

- GANを利用した画像生成モデル
- いくつかの構造制約により生成品質を向上
  - Generator
    - Pooling層の代わりに転置畳み込み層を使用
    - 最終層はtanh、その他はReLU関数で活性化
  - Discriminator
    - Pooling層の代わりに畳み込み層を使用
    - Leaky ReLU関数で活性化
  - 共通事項
    - 中間層に全結合層を使わない
    - バッチノーマライゼーションを適用

[2] A. Radfold, *et al.*, ICLR, 2016.

DCGANのネットワーク構造

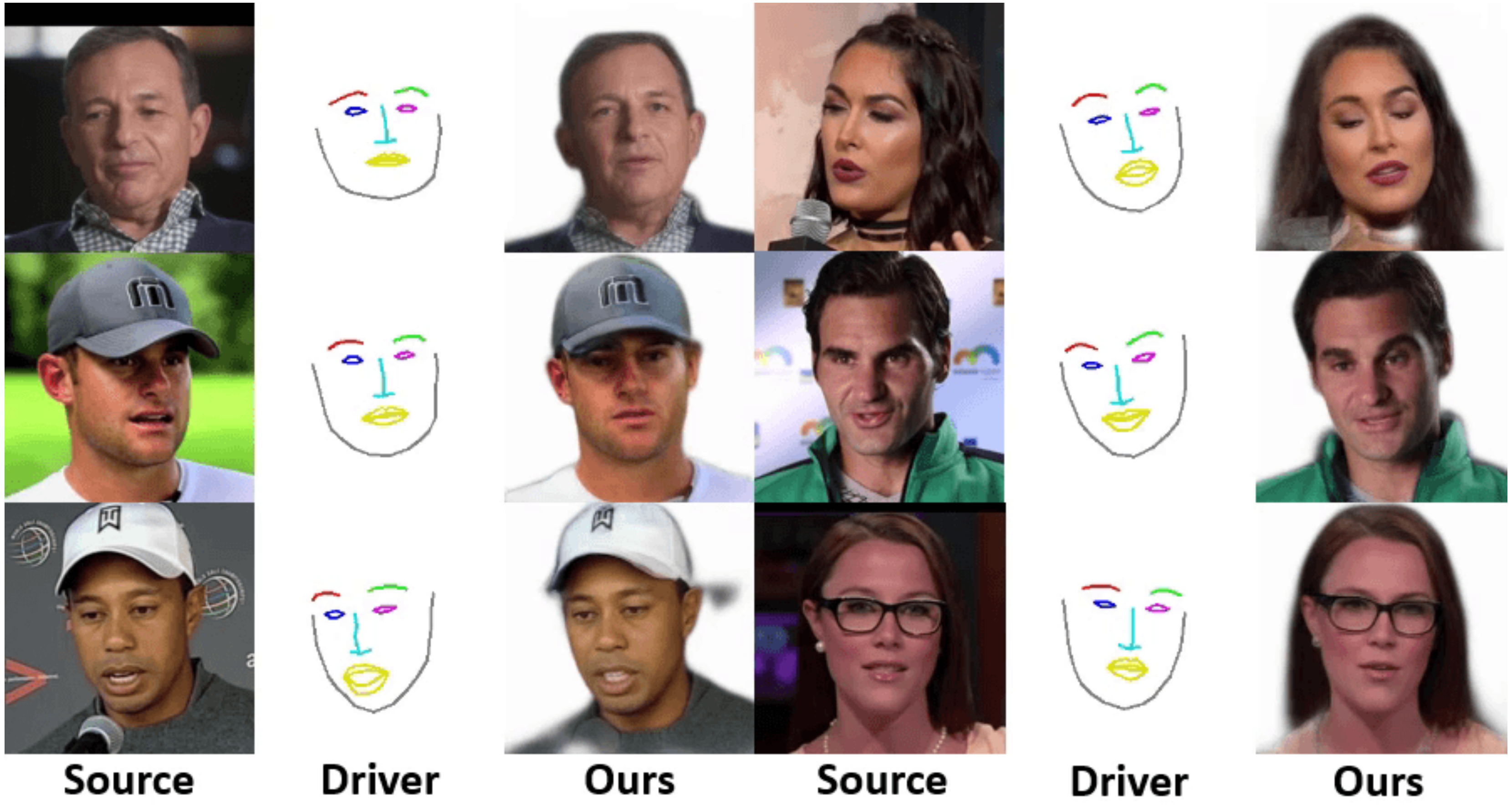
- Generator
  - 転置畳み込み層により乱数を画像にアップサンプリング



- Discriminator
  - 畳み込み層により画像から特徴量を抽出し、最終層をsigmoid関数で活性化

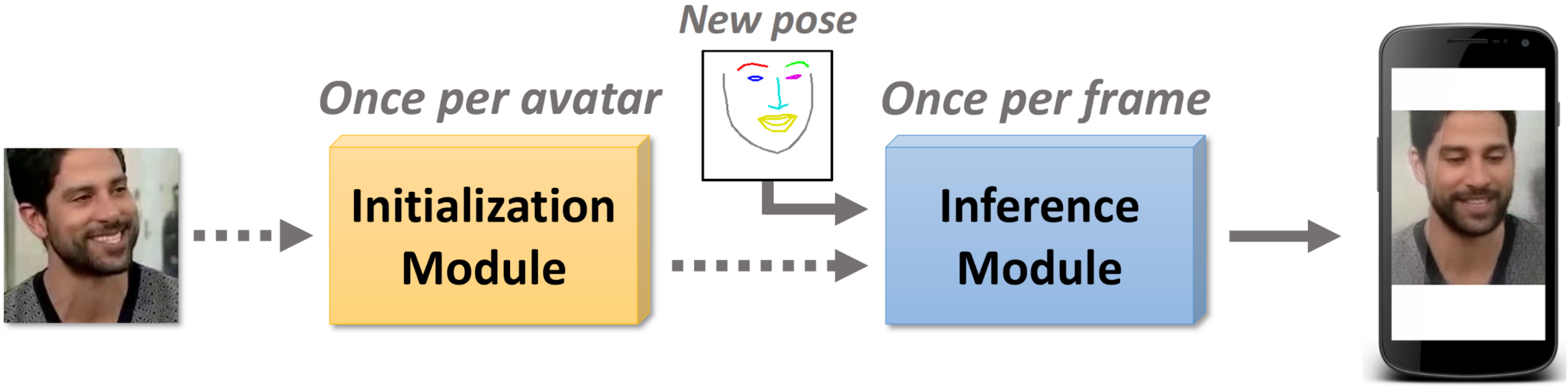
応用技術紹介

- Fast Bi-layer Neural Synthesis of One-Shot Realistic Head Avatars[3]
- 1枚の顔画像から動画像(Avatar)を高速に生成するモデル



一般的な顔アバター生成フロー

- 初期化部と推論部から成る
  - 初期化: 人物の特徴を抽出、1アバターにつき一回の計算コスト
  - 推論: 所望の動きを付ける、時間フレーム分だけの計算コスト

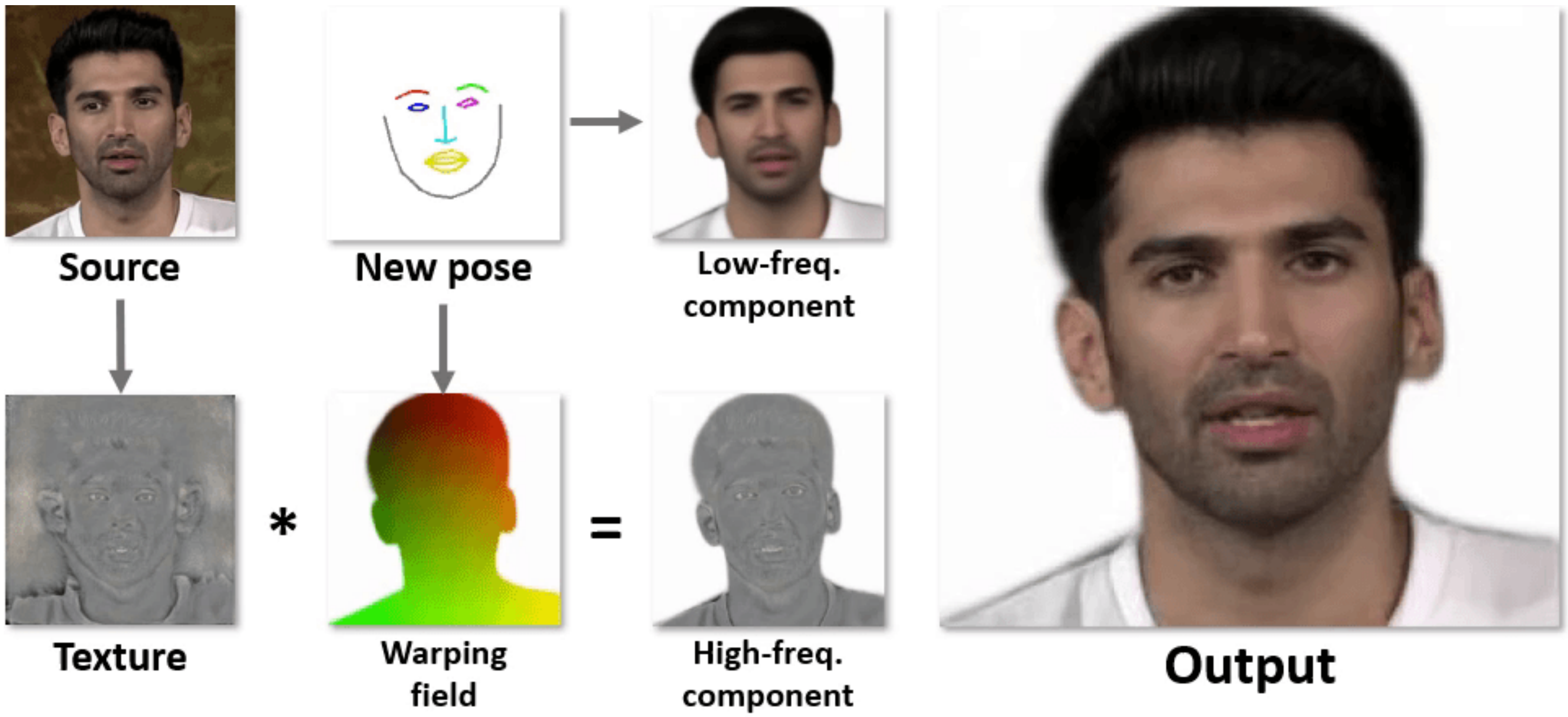


- 計算コストの比較
  - 従来: 初期化の計算コストが小さく、推論部の計算コストが大きい
  - 提案:初期化の計算コストが大きく、推論部の計算コストが小さい

リアルタイムで推論できる

推論部の計算コスト削減方法

- 緻密な輪郭と粗い顔画像を別々に生成し結合する
  - 初期化時に輪郭情報を生成（ポーズに非依存）
  - 推論時に粗い動画像を生成（ポーズに依存）



ネットワーク構造

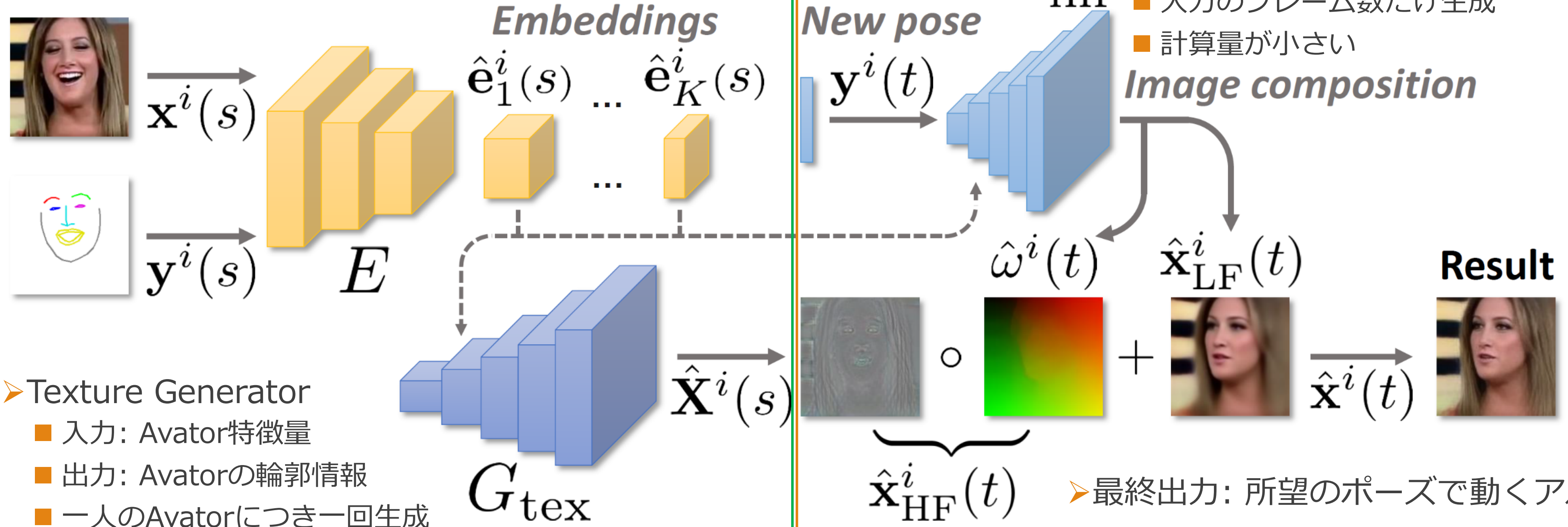
➢輪郭画像と低周波動画像を別々に生成する

初期化部

➢Embedder
- 入力：画像, ポーズ情報
- 出力：特徴量

推論部

➢Inference Generator
- 入力：所望のポーズ
- 出力：Warping Field, 低周波動画像
- 入力のフレーム数だけ生成
- 計算量が小さい



➢Texture Generator
- 入力: Avator特徴量
- 出力: Avatorの輪郭情報
- 一人のAvatorにつき一回生成
- 計算量が大きい

➢最終出力: 所望のポーズで動くアバター