卒 業 研 究 報 告 題 目

オプティカルフローを用いたpix2pixによる2Dアニメーションの中割り画像の生成

Generation of In-between Images in 2D Animation by Pix2pix using Optical Flow

指 導 教 員　　出口 利憲 教授

岐 阜 工 業 高 等 専 門 学 校　電気情報工学科

2020E31 古田 陵矢

令 和 7 年（ 2 0 2 5 年 ）　　2 月 1 8 日 提 出

Abstract

The current anime industry is in a very unstable state. One of the main reasons for this is that the anime industry is a labor-intensive business. As a solution to this problem, this study proposes a method that combines motion estimation using optical flow and image generation using pix2pix. This method aims temporally consistent and highly accurate intermediate frame generation by adding the inter-frame motion calculated using optical flow as input to pix2pix, which has difficulty capturing temporal changes. The usefulness of this study is shown by comparing it with the method using only pix2pix without optical flow. When comparing the experimental results of the implementation with only pix2pix and the implementation using optical flow, it is only the latter that the generator G is learning to generate the intermediate image, which shows the usefulness of using optical flow estimation for GAN learning. However, the accuracy of the generated images is not necessarily good, so it is necessary to improve them. By improving it, the technology of this research is expected to be useful for automating the work of drawing in-between animation, which connects the original pictures in animation production.

目 次

Abstract **i**

第 1 章 序論 **1**

第 2 章 GAN **2**

2.1 識別モデル 2

2.2 生成モデル 2

2.3 GAN 4

2.4 DCGAN 6

2.5 CGAN 7

2.6 pix2pix 8

2.6.1 損失関数と学習について 10

2.6.2 PatchGAN 12

2.6.3 ネットワーク構造と学習の流れ 14

第 3 章 オプティカルフロー **15**

3.1 オプティカルフロー 15

3.2 Lucas-Kanade 法 15

3.3 Farneback 法 16

第 4 章 提案手法 **17**

4.1 概要 17

4.2 Farneback 法を用いたオプティカルフローの推定 17

4.3 pix2pix を用いた補間画像の生成学習 17

4.3.1 pix2pix のみによる実装 17

4.3.2 オプティカルフローを入力に用いた pix2pix の実装 17

第 5 章 実験 **20**

5.1 実験準備 20

5.2 pix2pix のみでの実装実験 20

5.2.1 結果 20

5.2.2 考察 23

5.3 オプティカルフローを入力に用いた pix2pix の実装実験 23

5.3.1 結果 23
5.3.2 考察 24
5.3.3 追加実験結果 27
5.3.4 考察 27
第 6 章 結論 **30**
参考文献 **31**

第1章　序論

人々から長く愛され続けてきた日本が誇るアニメ文化は、近年では海外からも注目を集め今でも多くの人に見続けられている。そんなアニメ産業は今、大きな課題を抱えている。それはアニメ業界の競争激化に伴った労働力、技術力の増加や、アニメーター、プロデューサーなどの人手不足、それによって生まれる過酷な労働環境、低賃金などによる人材の流出など、これらの悪循環によって今のアニメ業界はとても不安定な状態となっている。そしてその根本として挙げられるのはやはり、アニメ産業が労働集約型のビジネスだということである。

そこで、近年ではアニメーションにCGや3Dモデルを用いるなどの工夫がされているが、最も注目したいものはアニメーションのフレーム補間技術である。現在の手書きアニメーションにおいて作業の負担が大きい部分はやはり原画と原画をつなぐ中割りを描く作業であり、この作業の自動化が実現すればアニメーション制作にかかる負担がかなり軽減される。しかし、実写の映像と違ってフレームレートが低く、フレーム間の動きが大きくなりやすいこと、そしてアニメーション特有の作品によるデフォルメや絵柄などの違いから、フレームの補間が困難とされているのが現状である。

その解決策の一つとして本研究では、オプティカルフローを用いた動き推定と、pix2pixを用いた画像生成を組み合わせる手法を提案する。この手法は、時間的な変化をとらえることが難しいpix2pixに、オプティカルフローを用いて計算したフレーム間の動きを入力として加えることで、時間的に一貫性のある高精度な中間フレーム生成を実現する。オプティカルフローを用いないpix2pixのみで行う手法と比較して、本研究の有用性を示す。

第2章　GAN[1)]

2.1　識別モデル

識別モデルとは、入力されたデータがどのクラスに属するかを識別する教師あり学習モデルである。識別モデルのシンプルな例として画像分類を考える。Figure 2.1 に識別モデルの学習工程を示す。識別モデルには学習と推論の 2 つのプロセスが存在する。画像分類であれば、「学習」は、画像 x を識別モデルに入力して得られた分類ベクトル $h_0(x)$ と正解ベクトル y の誤差を損失関数で計算し、誤差を最小化するようにパラメータを最適化する。このときのパラメータは識別モデルの尤度 (もっともらしさを表す数値) を最適化する。

Figure 2.2 に識別モデルの推論工程を示す。「推論」は、未知の画像 x を、最適化したパラメータを持つ識別モデルに入力し、分類ベクトル $h_0(x)$ を出力する。この分類ベクトル $h_0(x)$ の中の最も確率が高い 1 クラスが推論結果となり、確率付きでクラスを予想できる。

識別モデルを数式的な表現で定式化すると、「データ x が与えられた時の正解ラベル y の条件付き確率 $p(y|x)$ を出力するモデル」となる。

GAN における識別モデル (Discriminator) は、入力されたデータが本物の訓練データか、生成モデルが生成した偽物かを判別する役割を担う。

2.2　生成モデル

生成モデルとは、データそのものを生成する学習モデルである。生成モデルのプロセスは識別モデルの逆変換のイメージであり、識別モデルは入力された 3 次元配列の画像

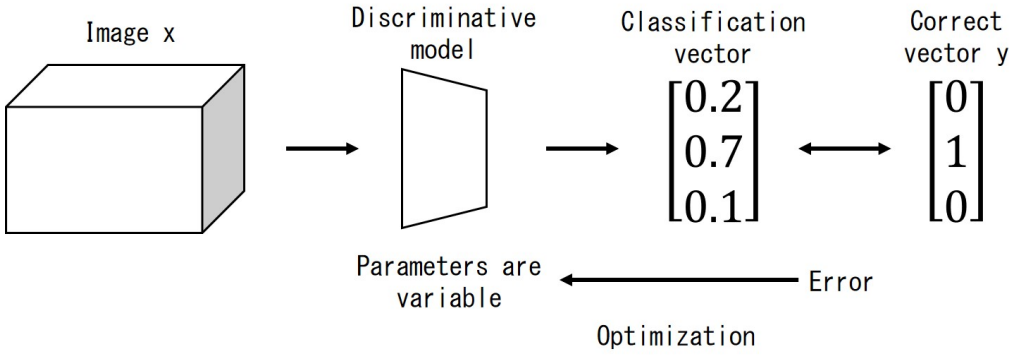

Figure 2.1　Image classification training.

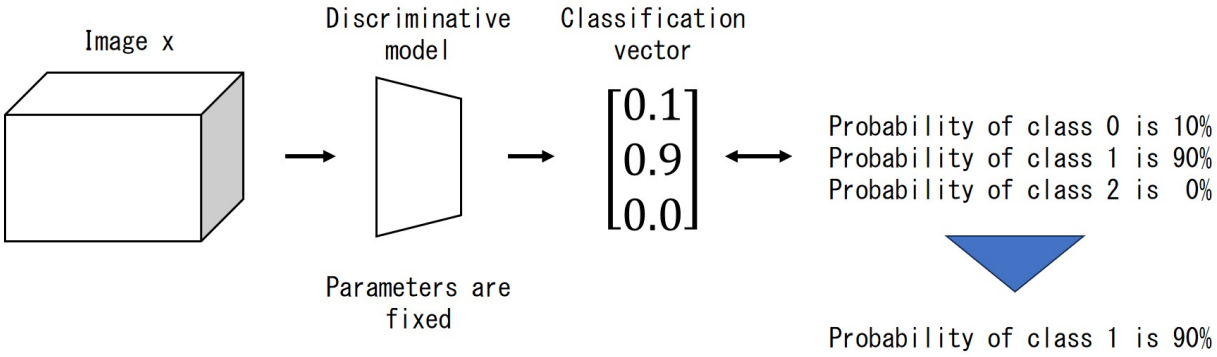

Figure 2.2　Image classification inference.

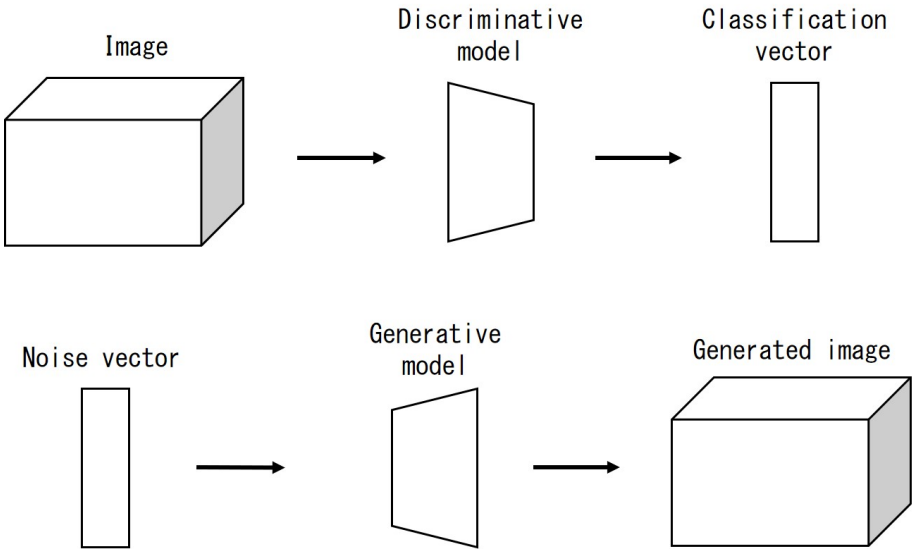

Figure 2.3　Comparison of discriminative and generative models.

を識別し、1 次元配列の分類ベクトルに変換するが、生成モデルは 1 次元配列のノイズベクトルを 3 次元配列の画像に変換する。Figure 2.3 に識別モデルと生成モデルの比較を示す。

生成モデルを数式的な表現で定式化すると、「ランダムノイズなどの確率変数から、データ x を生成する確率 $p(x)$ を出力し、正解ラベル y が与えられた場合、条件付き確率 $p(y|x)$ を出力するモデル」となる。

モデル分布 ($p(x)$) は、データ x が確率分布に従うと考えて、データが従う確率分布に近いほどいいモデルと考える。そのため、生成モデル、識別モデル共に最適化が重要となる。

GAN における生成モデル (Generator) は、乱数 (ノイズ) を入力として受け取り、訓練データに似た新しいデータを生成する役割を担う。

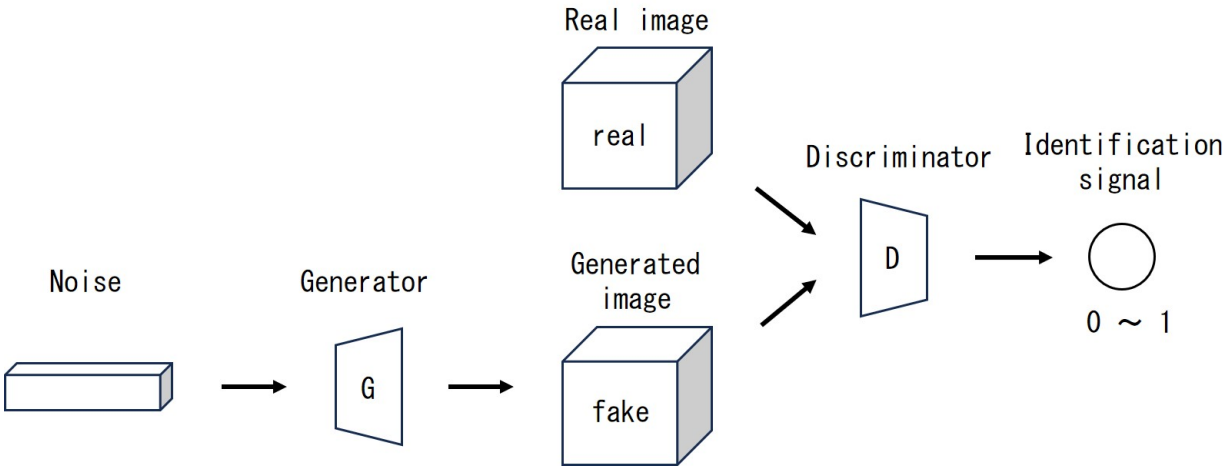

Figure 2.4　GAN Overview.

Table 2.1　Organizing the discriminator and generator.

Prosess	No	Image classified by discriminator	Parameters of generator	Parameters of discriminator	Correct label
Training a discriminator	1	real	fixed	variable	1(real)
	2	generate	fixed	variable	0(fake)
Training a Generator	3	generate	variable	fixed	1(real)

2.3　GAN

GAN は、敵対的生成ネットワーク (Generative Adversarial Networks) と呼ばれ、生成器 G(Generator) と識別器 D(Discriminator) の 2 つのネットワークを敵対的 (Adversarial) に競わせて交互に学習させるモデルである。Fiigure 2.4 に GAN のモデル構造を示す。生成器 G は、ランダムなノイズ z(一般的には標準正規分布) の確率変数から画像 x' を出力する生成モデルである。一方の識別器 D(Discriminator) は生成器 G の学習に使用し、画像の真贋を判定する識別信号を出力する識別モデルである。識別信号は画像が本物である確率を示し、0〜1 の値域になる。

Table 2.1 に識別器と生成器のパラメータの扱いを、Fiigure 2.5 に識別器 D の学習工程を示す。識別器 D は正解ラベルに 1(本物) と 0(偽物) の 2 値を使用し、本物画像の入力時は識別信号 1(本物)、生成画像の入力時は識別信号 0(偽物) を出力するよう学習する。

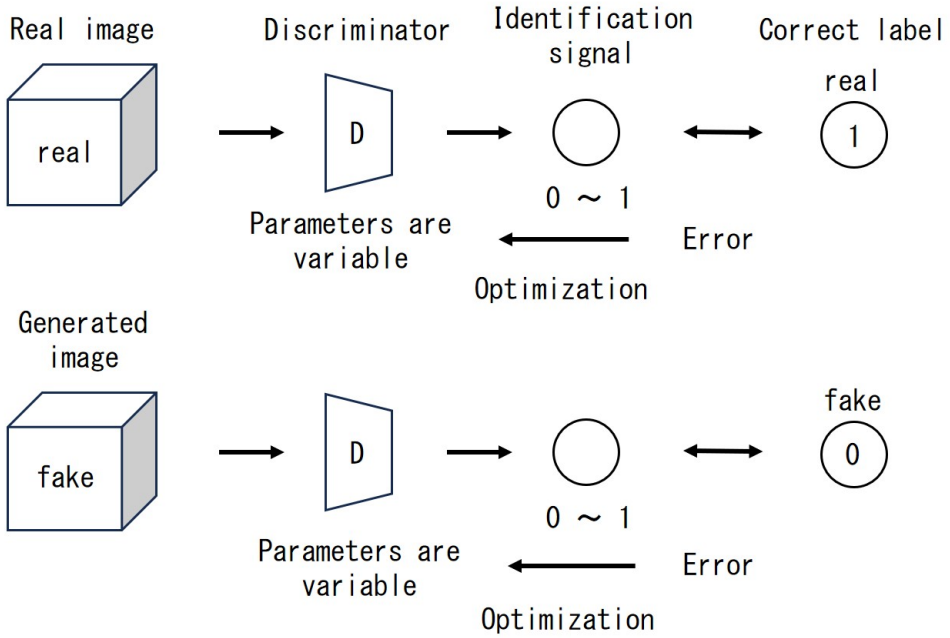

Figure 2.5　Training of Discriminator D.

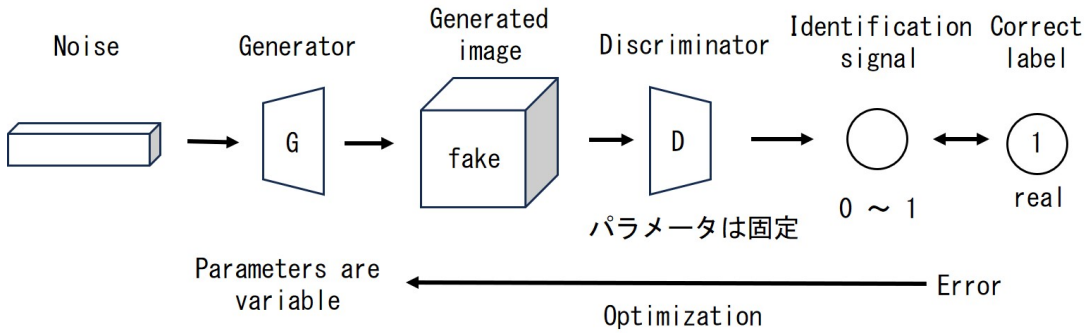

Figure 2.6　Training of Generator G.

Fiigure 2.6 に生成器 G の学習工程を示す。生成器 G は、識別器 D が識別信号 1(本物) を出力するよう、本物そっくりの画像の生成法を学習する。生成器の学習に必要な誤差は、識別器を経由して、逆伝播する。この時、識別器のパラメータを固定しておき、生成器のパラメータを最適化する。

つまり、生成画像の作成時には、識別器と生成器は逆の識別信号 (識別器の学習時は 0、生成器の学習時は 1) を出力するよう、最適化する。学習を繰り返すと、生成器 G と識別器 D は共に賢くなり、生成器 G はノイズから本物そっくりの画像を生成できるようになる。学習を終えると識別器は不要となり、生成器はノイズから本物そっくりの画像を生成する。Fiigure 2.7 に生成器 G の画像生成工程について示す。

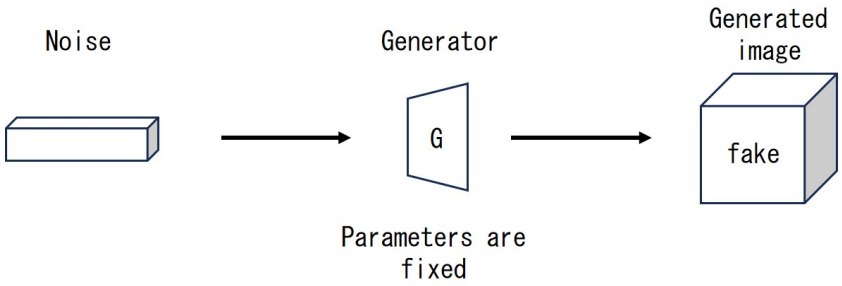

Figure 2.7　Image generation by Generator G

2.4　DCGAN

DCGAN(Deep Convolutional Generative Adversarial Networks) は、生成器と識別器に畳み込みニューラルネットワーク (Convolutional Neural Network、CNN) を利用した、オリジナル GAN の発展版である。

DCGAN は CNN の導入によって、オリジナル GAN に比べて画像が鮮明になり、安定した学習が可能になった。DCGAN の作成には試行錯誤を要し、作成の指針は論文 [2] によると次のとおりである。

1. 識別器にストライド 2 の畳み込み層、生成器にストライド 2 の転置畳み込み層を利用
2. 全結合層は利用しない
3. 出力層と生成器の入力層を除き、バッチ正規化 (Batch Normalization) を利用
4. 生成器は活性化関数に ReLU を利用し、出力層は tanh 関数を利用
5. 識別器はすべての活性化関数に Leaky ReLU を利用
6. 本物画像を −1〜1 の値域で正規化

指針 1 によると、識別器は畳み込み層を利用する。識別器の畳み込み層は Fiigure 2.8 イメージで、A の 3 × 3 カーネルがストライド 2 で移動する。また、パディング 1 の場合は点線のように画像の周囲が外側へ 1 だけ広がる。結果、B の 5 × 5 画像を畳み込み層に入力すると C の 3 × 3 の画像を出力し、画像サイズが縮小する。

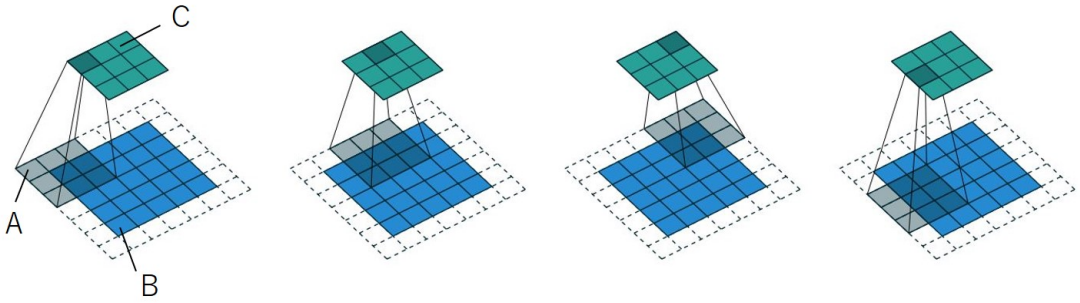

Figure 2.8　Image downsampling with a transposed convolution layer of stride 2.[3)]

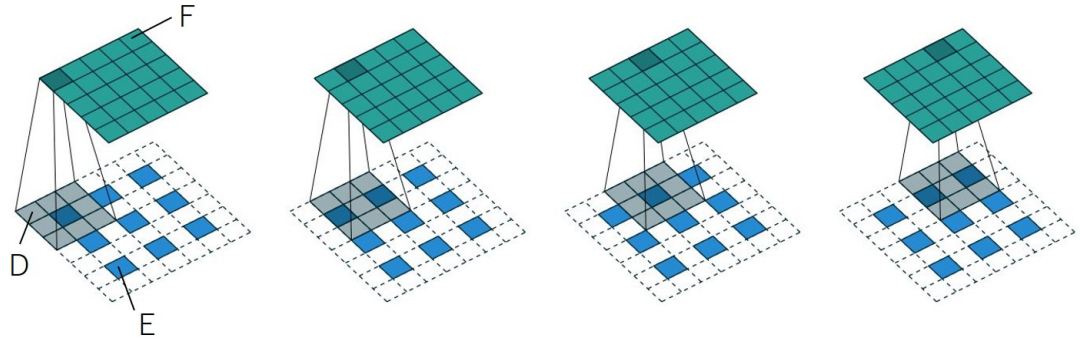

Figure 2.9　Image upsampling with a transposed convolution layer of stride 2.[3)]

生成器に使用する転置畳み込み層も Fiigure 2.9 のイメージで D の 3 × 3 カーネルがストライド 2 で移動する。また、パディングして画像の周囲を外側へ 1 だけ広げる。結果、E の 3 × 3 画像を転置畳み込み層に入力すると F の 5 × 5 画像を出力し、画像サイズが拡大する。

指針 4 によると、生成器の出力層に tanh 関数をしようする。tanh 関数は −1〜1 の値域であるため、生成画像の画素値は-1〜1 になる。識別器は本物画像と生成画像の真贋を判定するため、画素値の値域を揃える必要がある。そのため、指針 6 で本物画像の画素値を −1〜1 の値域に変換する。

指針 5 によると、識別器の活性化関数に Leaky ReLU を使用する。Fiigure 2.6 で示したとおり、生成器は識別器を経由して、誤差を逆伝播する。そのため、値がマイナスでも勾配損失しない Leaky ReLU を使用する。

2.5　CGAN

CGAN(Conditional Generative Adversarial Networks) は、オリジナル GAN と同じく生成器と識別器の 2 のネットワークで構成される。GAN との相違点は、クラスを指定す

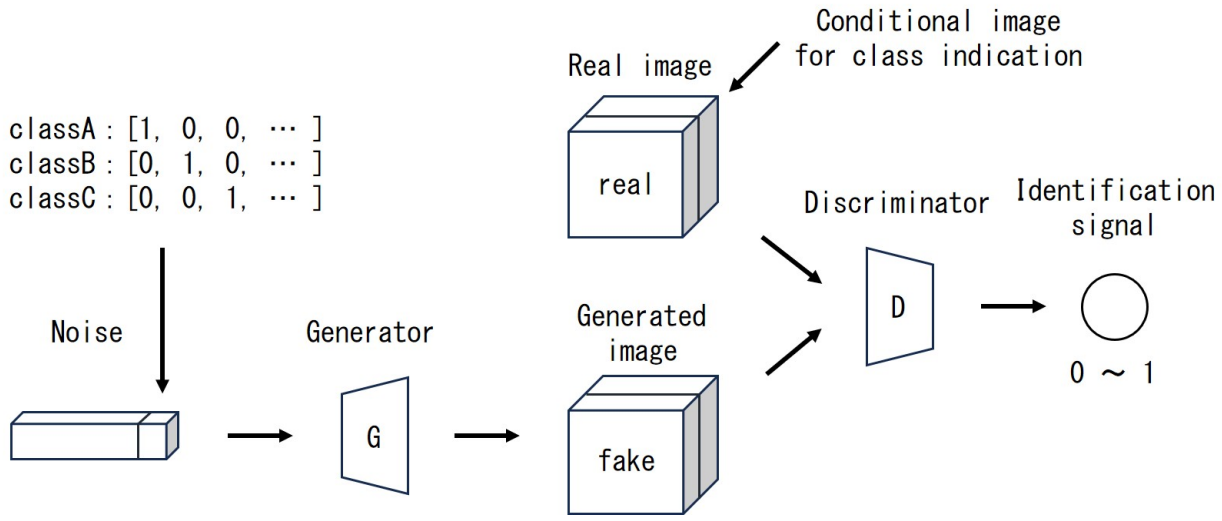

Figure 2.10　CGAN Overview.

る条件ベクトル y である。データ分布 $p_{data}(x)$ から取得される画像を x、条件を y、ランダムノイズを z とすると、損失関数は次の式 (2.1) になる。

$$minmaxL(D,G) = E_{x-pdatax}[\log D(x|y)] + E_{z-p_z(z)}[\log(1 - D(G(z|y)))] \tag{2.1}$$

第 1 項の $D(x|y)$ は条件 y を満たす画像 x の識別器への入力、第 2 項の $G(x|y)$ は条件 y を満たすランダムノイズ z の生成器への入力と解釈できる。

Fiigure 2.10 に CGAN のモデル構造を示す。GAN はノイズ z を生成器に入力したが、CGAN はノイズ z にクラスを指定する One-Hot ベクトルを結合したノイズを生成器に入力する。同様に、識別器に入力する本物画像にもクラスを指定する情報が必要になる。ただし、画像は 3 次元配列であるため、ノイズのように単純に 1 次元の One-Hot ベクトルを結合できない。そこで、One-Hot ベクトルを画像サイズに拡大したクラス数分の条件画像をチャンネル方向に結合する。このとき、「すべての要素が 0」の条件画像になる。

2.6　pix2pix

pix2pix モデルは、Conditional GAN のアイデアを拡張してノイズベクトル z の代わりに画像そのものを入力とすることで、GAN の技術でドメイン変換のタスクを実現することを可能にしたモデルである。pix2pix は共通化されたアーキテクチャを用いて複雑なドメイン変換のタスクを学習させることに成功しており、タスクに特化した従来のモデルでは得られなかった結果が、得られるようになっている。適用できる分野も多種多様であり、以下の例の他にも様々な応用が考えられる。Figure 2.11 に変換の例を示す。

- 夜の写真を昼の写真に変換する

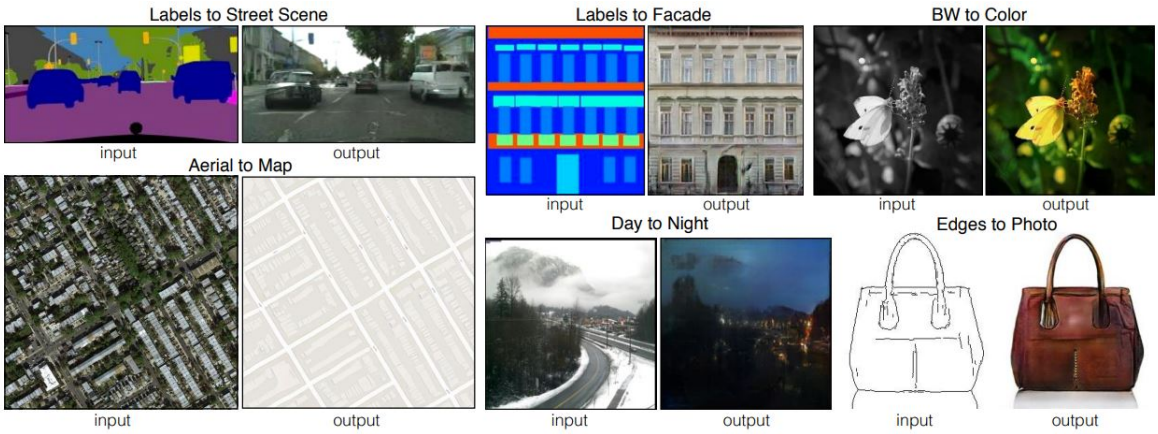

Figure 2.11　Examples of domain translation with pix2pix.[4)]

- 線画をリアルな実物写真に変換する
- セグメンテーション画像をリアルな道路画像に変換する
- 衛星写真から地図を生成する
- 顔の輪郭から肖像画を生成する

Figure 2.12 に pix2pix のモデル構造を示す。pix2pix モデルは、Conditional GAN と同様に、生成器 G と識別器 D から構成される。生成器 G は、ドメイン A の画像を入力して、ドメイン B の画像を生成するように学習を行う。また、識別器 D は、ドメイン A とドメイン B の 2 つの画像を 1 組の入力として、その画像の組が本物の組み合わせか、偽物の組み合わせかを識別するように学習する。また、Conditional GAN では、データセット以外のデータも生成できるよう、ノイズ z を生成器 G の入力としているが、pix2pix ではドロップアウト層に置き換えられる。

pix2pix と Conditional GAN で異なっている 3 点を以下に示す。

1. Conditional GAN は生成器 G の入力をノイズ z(洗剤空間のベクトル) としているが、pix2pix は画像そのものを種として生成器 G が画像を生成する。
2. Conditional GAN は、生成器 G の生成した偽物画像を識別器 D の入力としているが、pix2pix は 2 つのドメインの画像ペアを識別器の入力としている。ここで、識別器 D は、入力された画像ペアがデータセットに含まれる組み合わせ (本物) か、生成器 G が生成した画像を含む組み合わせ (偽物) かを識別する。
3. Conditional GAN において、生成器 G に入力されるノイズ z は、学習データ以外のペアも確率的に生成できるようにする役割がある。しかし、pix2pix は、ノイズを入力してもそれを打ち消すように生成器 G の学習が進んでしまうため、z は入

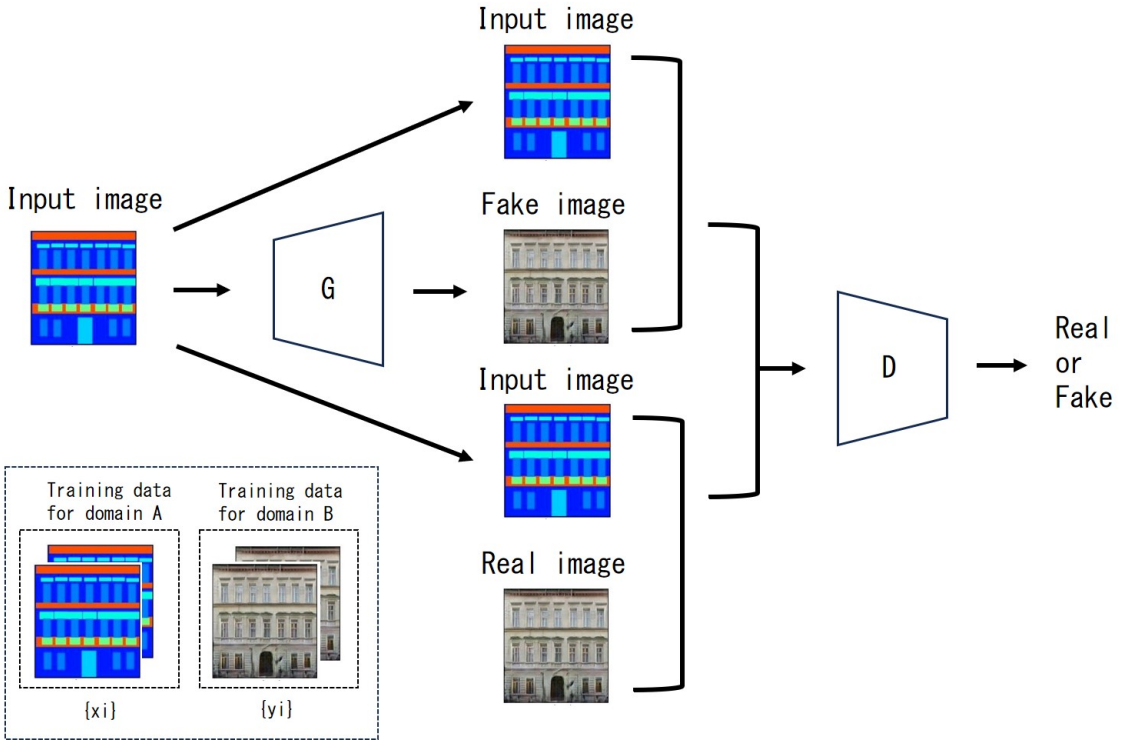

Figure 2.12　pix2pix model architecture.

力として使用されない。pix2pix では、生成器 G に含まれるドロップアウト層がノイズの役割を果たしている。

2.6.1　損失関数と学習について

モデルの学習をどのように進めるか決定するのが損失関数である。以下に pix2pix の損失関数と学習方法を示す。

pix2pix モデルでは、生成器 G と識別器 D の敵対的学習を行う。すなわち、生成器 G は、識別器 D が本物と間違えるような出力を生成するように学習し、識別器 D は、2 つのドメインの画像ペアが本物か偽物かを識別するように学習する。

この 2 つの学習は、以下の式 (2.2) の敵対性損失 (Adversarial loss) によって表せる。

$$L_{cGAN}(C, D) = E_{x,y}[\log D(x, y)] + E_{x,z}[\log(1 - D(x, G(x, z)))] \tag{2.2}$$

ここで $D(x, y)$ は、画像 x と画像 y の組み合わせが本物の場合 D=1、偽物の場合 D=0 として識別結果を出力する。

右辺の第 1 項は、学習データのペア画像 (x, y) に対して、識別器 D が正しく本物と識別できるほど大きくなる項である。また右辺の第 2 項は、画像 x および生成器 G による偽物画像 $\mathrm{G}(x, z)$ のペア画像に対して、識別器 D が正しく偽物と識別できるほど大きく

Table 2.2　Advantages and disadvantages of two loss functions for the generator G in pix2pix.

	Advantages	Disadvantages
Adversarial loss(L_{cGAN})	Sharp and colorful images are generated that are so realistic they are indistinguishable from real ones.	Extreme images are generated that disregard reality.
L_1 loss	Images are generated that appear realistic when viewed as a whole.	Images with blurred edges in shades of gray (midtones) are generated.

なる項である。

この損失関数に対して、識別器 D は L を最大化するように、生成器 G は L を最小化するように学習することで、1 つの損失関数を用いて敵対的学習を行うことができる。

この敵対性損失によって生成器 G は本物と見間違うようなリアルな画像を生成するようになるが、一方で、データセットの画像を無視した極端な画像が生成されやすいという問題も起こる。そこで pix2pix では、以下の式 (2.3) で表される L_1 損失を考慮している。

$$L_1(G) = E_{x,y,z}[|y - G(x,z)|_1] \tag{2.3}$$

L_1 損失は、本物画像 y と偽物画像 $G(x,y)$ の間の平均絶対誤差を表している。生成器 G の学習を、L_1 を最小化するように進めることで、よりリアルな画像を生成できるだけでなく、本物のターゲット画像により近い画像を生成できるようになる。なお、L_1 損失は識別器 D の学習時には考慮されない。

学習はこれらの 2 つの損失、すなわち敵対性損失と L_1 損失の重み付き合計損失によって進むが、相反する効果があるため、係数は慎重に決定する必要がある。Figure 2.2 に損失関数の生成器 G への効果についてのメリット・デメリットを示す。

L_1 損失を考慮すると生成器 G がより本物らしい、つまりデータセットに近い画像を生成する効果があるが、あらゆる可能な選択肢の中で平均的に損失の小さな画像、つまりぼやけた模様や中間色を生成してしまう傾向がある。pix2pix の論文 [4] では、敵対性損失と L_1 損失のトレードオフを実験的に調べており、最も本物らしい画像を生成した係数 λ =100 が選択されている。

Figure 2.13 に生成器 G の生成した画像の比較を示す。本物画像と比較すると、L_1 損失のみを考慮した場合は、全体が灰色でぼやけてしまう。一方、敵対性損失のみを考慮

Figure 2.13　Comparison of Images Generated by the Generator G.[3)]

した場合 (L_{cGAN}、λ=0) は鮮明な画像が生成されるが、本物を無視した極端な画像が生成される場合がある。この 2 つのバランスをとることで、よりリアルで鮮明な画像が生成される ($L_1 + L_{cGAN}$、λ=100)。

pix2pix のネットワーク構造について、識別器 D は単純な畳み込みニューラルネットワークである。異なるドメインのペア画像を結合したものを入力し、そのペアが本物かどうかを識別する。また、

2.6.2　PatchGAN

リアルな画像には鮮明さが求められる。しかし L_1 損失を考慮するだけでは、生成画像を本物に近いものにすることはできても、画像を鮮明にする効果はなかった。

そこで pix2pix で導入したのが PatchGAN と呼ばれるアーキテクチャである。これは、画像を N × N サイズのパッチに分割して、バッチ単位で識別器に本物か偽物化を判別させるようにする方法である。これによって、生成器 G は画像全体ではなく、より狭い範囲のパッチ単位でみて本物らしい画像を生成するように学習し、より鮮明な画像を生成できるようになった。

Figure 2.14 に通常の GAN と PatchGAN の比較を示す。PatchGAN のアイデアは以下のようなものである。識別器 D は、単純な畳み込みネットワークであるが、最終出力をあるサイズを持った特徴マップにしておき、各ピクセルごとに本物か偽物かを判別する。そうすると、識別器 D は入力画像 x の実効的な受容野 (Recepitive Field、バッチ) に対して本物か偽物かを判別することになるため、画像全体を入力するよりも、より鮮明な画像が生成されるようなる。

論文 [4)] では、パッチサイズの大きさを変えたときに生成器 G の生成画像がどう変化するかを検証している。Figure 2.15 に PatchGAN 導入による影響について示す。

パッチのサイズが 1 × 1 ピクセルとなる「PixelGAN」から、入力サイズ全体となる

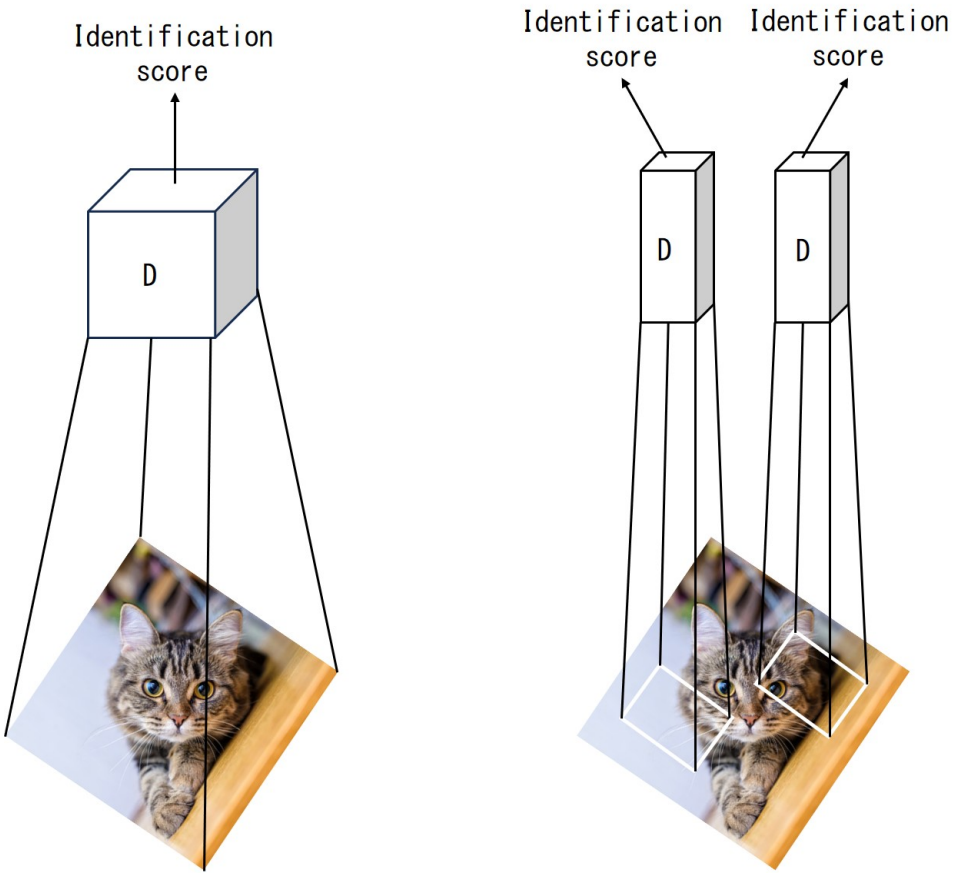

Figure 2.14　A comparison between a normal GAN and a PatchGAN.

Figure 2.15　Effects of using PatchGAN.[4)]

「ImageGAN」まで、徐々に変化させたときの生成画像を比較している。pix2pix では、最もカラフルで鮮明な画像が得られる 70 × 70 のパッチサイズが選択されている。

実際の実装では 70 × 70 という数字はどこにも現れてこない。その代わりに、256 × 256 サイズの入力画像に対して、実効的な受容野サイズが 70 × 70 となるよう、最終的な特徴マップサイズを 30 × 30 としている。最終出力の各ピクセルに対して真偽を識別し、各ピクセルに対するスコアから損失を計算、誤差逆伝播により識別器 D の重みを更新し、各ピクセルからの効果を重ね合わせて、識別器 D の学習を進めていく。

2.6.3　ネットワーク構造と学習の流れ

pix2pix のネットワーク構造について、識別器 D は単純な畳み込みニューラルネットワークである。異なるドメインのペア画像を結合したものを入力し、そのペアが本物かどうかを識別する。また、PatchGAN で実行的な受容野が 70 × 70 サイズになるようにネットワークが形成される。

一方、生成器のネットワーク構造は、少し複雑な構造をしており、「U-net」と呼ばれるネットワーク構造である。入力画像をボトルネック層までダウンサンプリングするエンコーダ部分と、ボトルネック表現を出力画像サイズまでアップサンプリングするデコーダ部分から構成される。

U-net は、エンコーダとデコーダとの間に Skip-connection が追加され、ピクセルレベルの詳細な情報を、ボトルネック層を飛び越して伝播する構造を持っている。例として、線画をカラー画像に変換するタスクの場合は、エンコーダで生成されたエッジ位置をデコーダと共有することで、より鮮明な画像を生成するようになる。また、層の間に Dropout レイヤーを追加することで、各レイヤーにノイズ要素を追加し、学習データにならないような入力に対しても、よりリアルな画像生成を可能にしている。

第3章　オプティカルフロー

3.1　オプティカルフロー [5)]

オプティカルフロー (optical flow) とは、物体やカメラの移動によって生じる隣接フレーム間の物体の動きの見え方のパターンである。各ベクトルが 1 フレーム目から 2 フレーム目への変位ベクトルを表す 2 次元ベクトル場で表現される。オプティカルフローは以下の仮定のもと計算される。

1. 連続フレーム間で物体の画像上の明るさは変わらない
2. 隣接する画素は似たような動きをする

1 枚目の画像中の画素 $I(x, y, t)$ を考える (t は時間軸方向を表す次元)。時刻 dt 後に撮影された次の画像中で (dx, dy) の距離を移動したとする。この二つの画素は同じものを見ていて、かつ明るさは変わらないと仮定したので、以下の関係式 (3.1) が成り立つ。

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{3.1}$$

式 (3.1) の右辺をテイラー展開し、共通する功を取り除き dt で割ると以下の式 (3.2) を得る。

$$f_x u + f_y v + f_t = 0 \tag{3.2}$$

ここで

$$f_x = \frac{\partial f}{\partial x};\ f_y = \frac{\partial f}{\partial y};\ f_t = \frac{\partial f}{\partial t};\ u = \frac{dx}{dt};\ v = \frac{dy}{dt} \tag{3.3}$$

この式を Optical Flow equation と呼ぶ。これにより画像の勾配である f_x と f_y、時間軸方向の勾配 f_t を計算できる。しかし、(u, v) が未知であり、二つの未知数があるのに式が一つしかないため、この問題を解くために様々な手法が提案されている。

3.2　Lucas-Kanade 法 [6)]

Lucas-Kanade 法 (ルカス・カナデ) は、画像のごく局所的な領域に対して、領域内すべてのピクセルが同じ速度で動き、物体の輝度は変化しないと仮定する。そのうえで最小二乗法を用いて動きのベクトルを計算する手法である。

最もシンプルなオプティカルフロー推定アルゴリズムで、小さな物体の動きを追跡する場合に有効である。あるピクセルの動きをその周囲のピクセルの動きと関連付けることで、滑らかなオプティカルフロー場を推定する。

Lucas-Kanade 法は局所的な計算のため、比較的高速でリアルタイムの映像解析に適し

ている利点がある。また、複数のピクセルの情報を使用するため、単一ピクセルベースの手法よりもノイズに強い特徴がある。

ただし、ピクセルの動きが大きい場合や、輝度が大きく変化する領域での精度が低下することがあり、限られた条件下で最適に機能する特徴を持っている。

3.3 Farneback 法[6)]

Farneback 法は、連続するフレーム間の動きを、画像を複数の解像度の画像に分解し、各レベルで光流を計算することで、各画素の周辺領域の動きを推定し、それを組み合わせて光流場を生成する手法である。フレーム内の局所的な領域を多項式で近似し、その領域の変形を解析することで各ピクセルの動きを算出する。

Farneback 法の利点は、計算精度が高く、画像全体の滑らかな動きの推定が可能である点である。ただ、滑らかな変位場を仮定しているため、不連続な動きを適切に捉えられない可能性がある。

第4章　提案手法

4.1　概要

連続する3枚のアニメーションの1枚目と3枚目のフレームと、オプティカルフローを用いて計算したベクトルをpix2pixの入力として、2枚目の補間画像を出力する手法を提案する。

4.2　Farneback法を用いたオプティカルフローの推定

学習に用いる連続した3枚のアニメーションフレーム (frame_t−1, frame_t, frame_t+1) から、frame_t−1からframe_t+1へのオプティカルフロー (flow_forward) と、frame_t+1からframe_t−1へのオプティカルフロー (flow_backward) を計算する。オプティカルフローの計算にはFaneback法を使用する。Figure 4.1にオプティカルフローの計算を示す。

4.3　pix2pixを用いた補間画像の生成学習

pix2pixを用いた補間画像の生成学習を行うが、オプティカルフローを入力に用いない場合と用いる場合の2種類の実装を行う。

4.3.1　pix2pixのみによる実装

Figure 4.2に入力に使用する画像について示す。pix2pixの実装をする上で、Generatorの入力にはframe_t−1とframe_t+1のアニメーション画像を1つのtensorに結合したものを使用する。この際、入力のチャネル数は、1つのRGB画像が3チャネルであるため、3+3=6で6チャネルの入力になる。

Figure 4.3にpix2pixのみによる実装のモデル構造を示す。Discriminatorの入力には、Generatorの入力で使用したtensorにframe_tのアニメーション画像を結合したものと、Generatorが生成した画像を結合したものの2種類を使用する。入力チャネル数は6+3=9で9チャネルになる。

4.3.2　オプティカルフローを入力に用いたpix2pixの実装

Figure 4.4に入力に使用する画像について示す。オプティカルフローを入力に用いたpix2pixの実装をするうえで、Generatorの入力にはframe_t−1とframe_t+1のアニメー

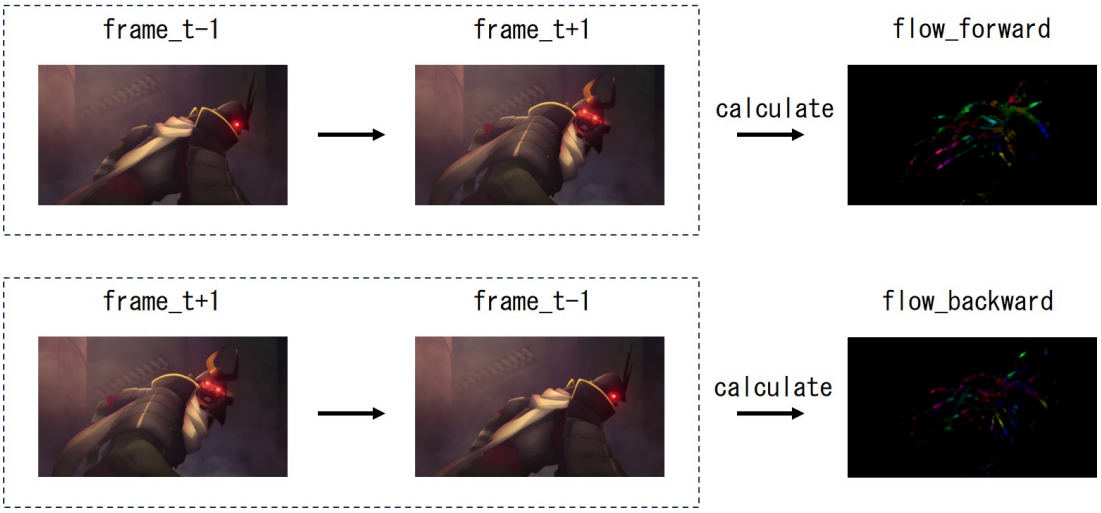

Figure 4.1　Optical flow Calculation.

Figure 4.2　Concatenating consecutive frames into a tensor.

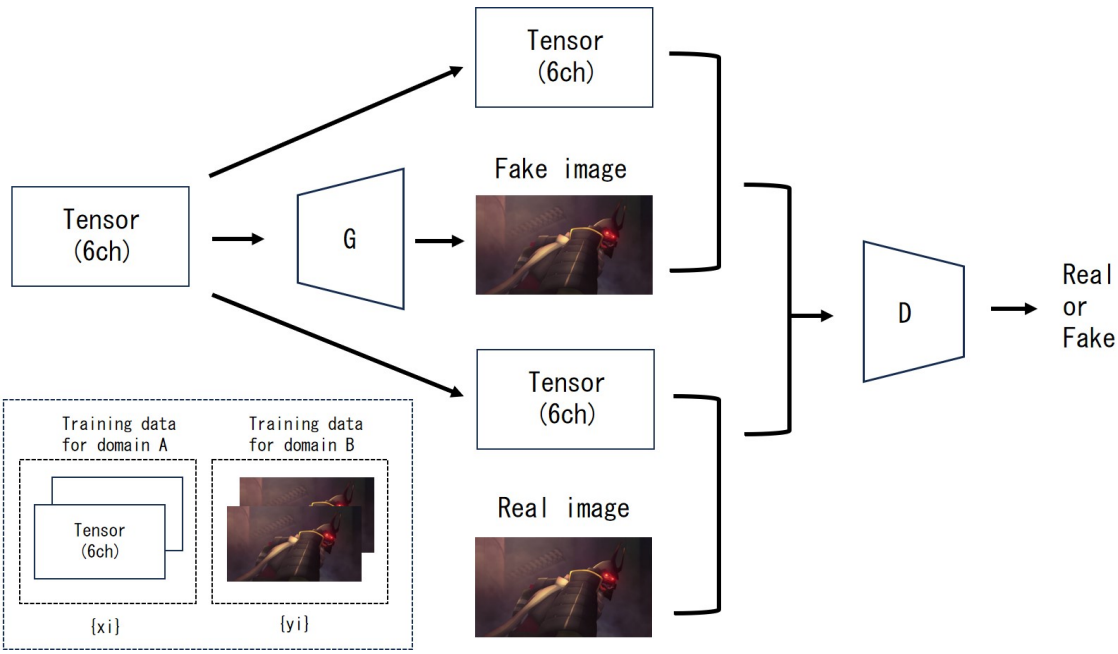

Figure 4.3　Model architecture of an implementation using only pix2pix.

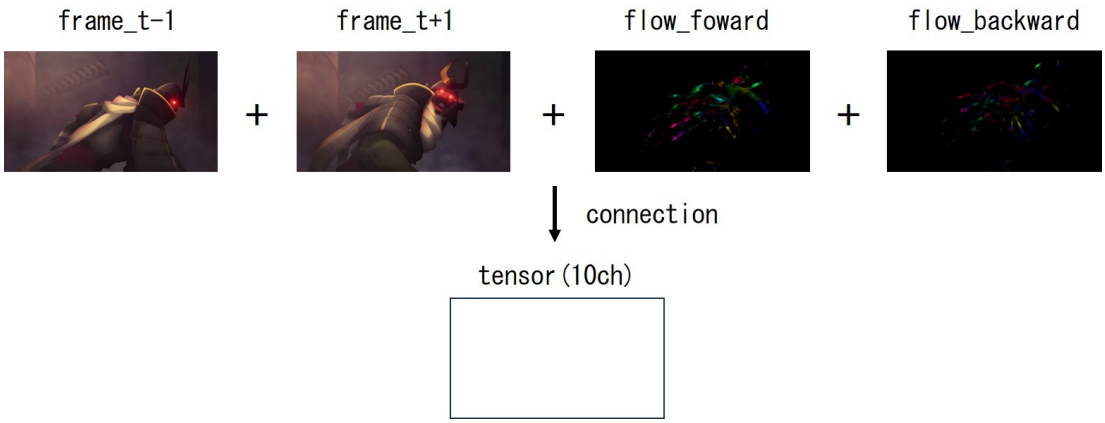

Figure 4.4　Concatenating frames and optical flow into a tensor.

ション画像と flow_forward と flow_backward のオプティカルフローベクトルを 1 つの tensor に結合したものを使用する。この際、入力のチャネル数は、1 つの RGB 画像が 3 チャネルで、1 つのベクトルが 2 チャネルのため、3+3+2+2=10 チャネルの入力となる。

Figure 4.5 にオプティカルフローを入力に用いた pix2pix の実装のモデル構造を示す。Discriminator の入力には Generator の入力で使用した tensor に frame_t のアニメーション画像を結合したものと、Generator が生成した画像を結合したものの 2 種類を使用する。入力チャネル数は 10+3=13 で 13 チャネルになる。

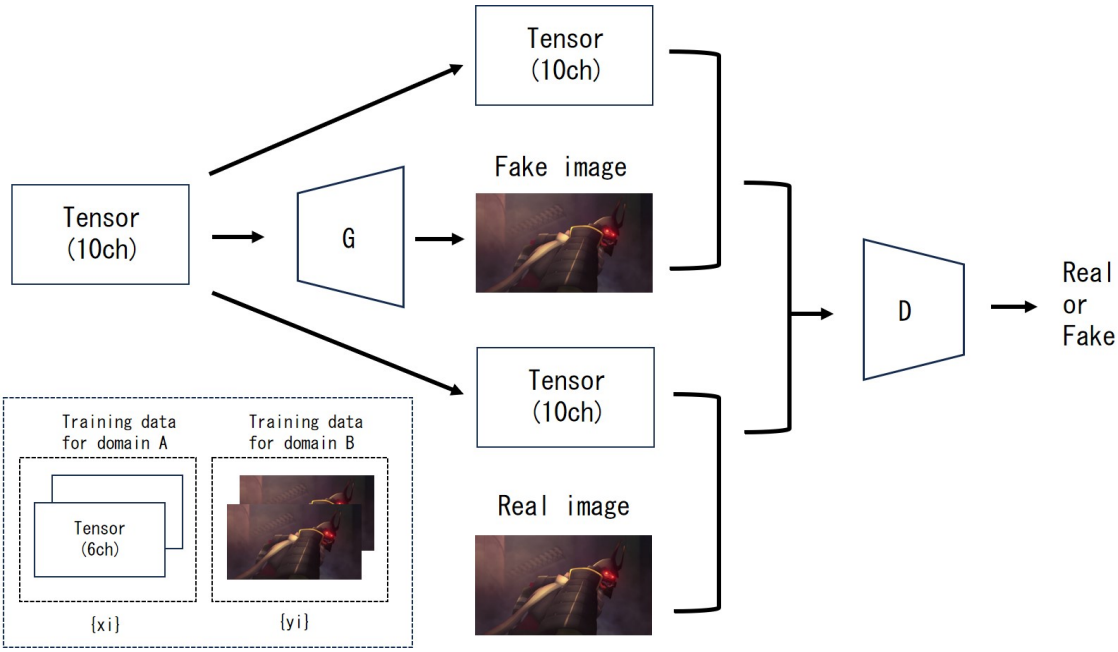

Figure 4.5　Model architecture of a pix2pix implementation using optical flow as input.

第5章　実験

5.1　実験準備

実験に向けて、3枚の連続したアニメーションフレームを集めたデータセットを用意する必要があるため、「ATD-12K」[7] というアニメーションデータセットを使用した。データセットの内容を Figure 5.1 示す。

研究設備と時間及び使用するネットワークの都合によりデータセットのサイズを変更する必要があるため、データセットの画像サイズを 508 × 286 サイズに変更して使用する。そして、pix2pix のネットワークには、参考文献 [1] で使用されているサンプルコードを雛形として利用する。

5.2　pix2pix のみでの実装実験

5.2.1　結果

4.3.1 項で提案した pix2pix のみによる実装をもとに、5.1 節で示したデータセットと pix2pix のサンプルコードを用いて実験を行った。この際、Generator に入力する tensor のチャネル数は 6、Discriminator に入力するチャネル数は 9 であるため、それぞれのネットワークの入力チャネル数を変更した。学習エポック数は 10 エポックで実行した。

学習後に生成器 G が生成した画像を Figure 5.2 に示す。左の画像が生成画像、右の画像が正解画像となっている。これらの生成画像からわかることとして、一部は比較的似た画像を生成できているが、ほとんどが 1 枚目と 3 枚目を重ねたような画像が生成されていることである。

Figure 5.3〜Figure 5.6 に学習に使用した損失関数 (LossD_real, LossD_fake, LossG_GAN, LossG_L1) の変化を示す。Figure 5.3、Figure 5.4 から、lossD_real、lossD_fake 共に早い段階ではほぼ 0 に近い値に推移していることがわかる。しかし何度か値が急上昇していることもわかる。Figure 5.5 から、LossG_GAN は何度か下がることはあっても、基本的には早い段階で値が上昇していることがわかる。Figure 5.6 から、LossG_L1 は比較的早い段階で値が減少しているが、値が 3.5 あたりで減少幅がほとんどなくなっていることがわかる。

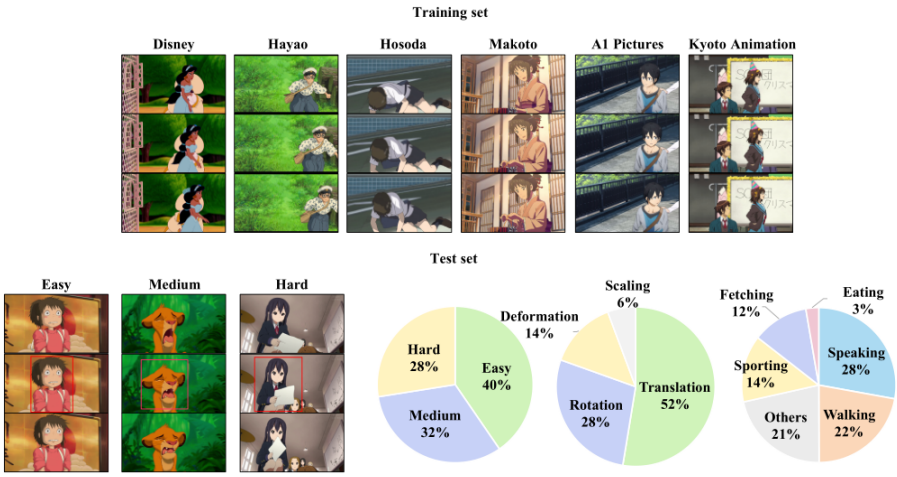

Figure 5.1　Dataset content.[7)]

Figure 5.2　Generated image.

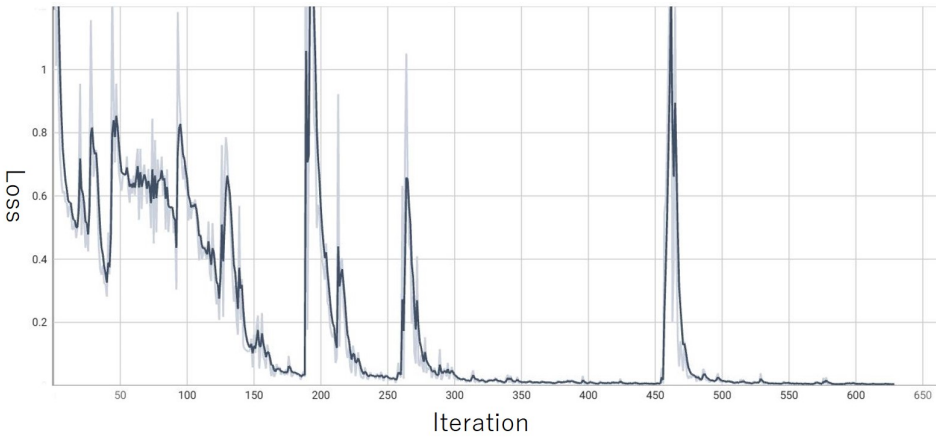

Figure 5.3　LossD_real.

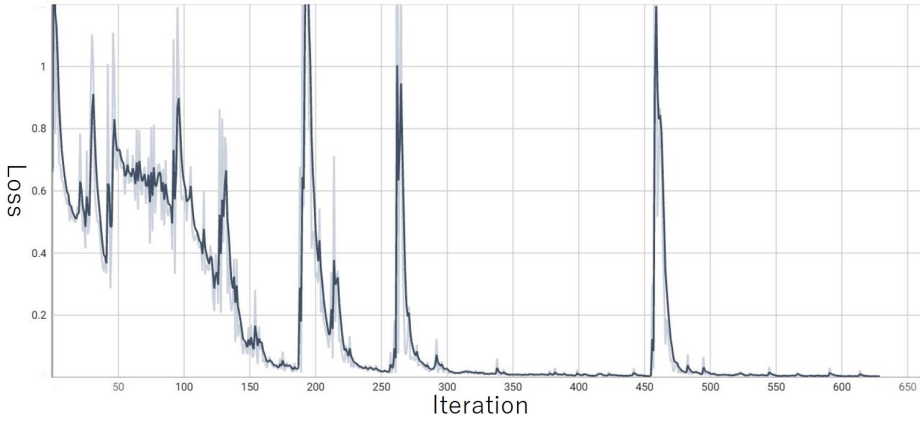

Figure 5.4 LossD_fake.

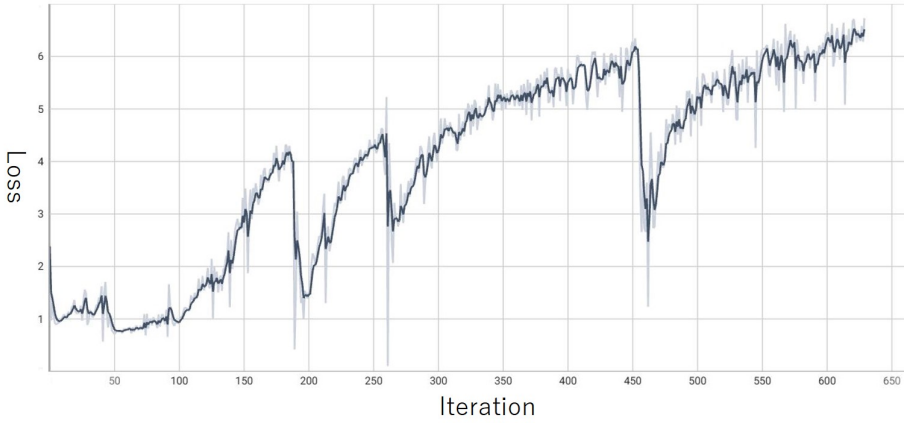

Figure 5.5 LossG_GAN.

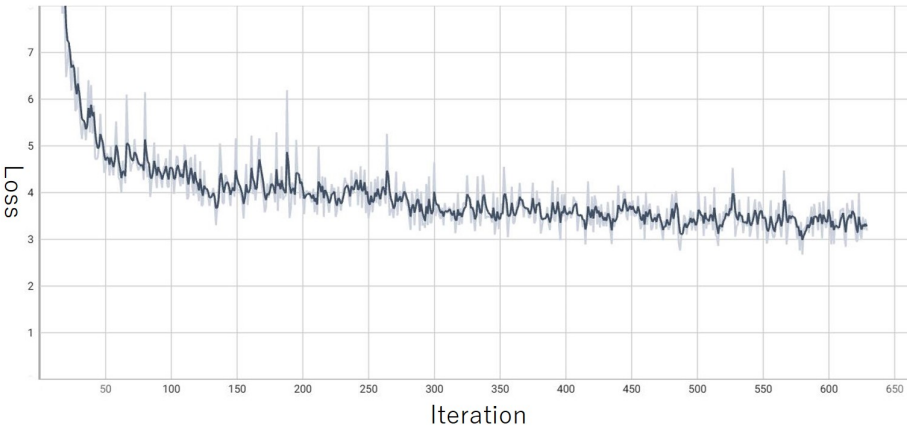

Figure 5.6 LossG_L1.

5.2.2　考察

学習後に生成器 G が生成した画像について、1 部の画像は比較的似た画像が生成された理由として、3 枚の連続したフレームにおいて画像に変化がほとんどないデータにおいてこの現象が起きていると考えられる。1 枚目と 3 枚目に変化がほとんどなければ、1 枚目と 3 枚目を重ねたような画像を生成したとしても、2 枚目とほぼ変わらない画像になるためである。そのため、この実験において学習させた生成器 G は、1 枚目と 3 枚目を重ねたような画像のみを生成するように学習しており、2 枚目を生成するようにする学習がうまくいっていないと考えらえる。これは、LossD_real と LossD_fake の値が学習後すぐに下がっており、識別器 D が強くなりすぎていることからもわかる。また、LossD_real と LossD_fake のグラフにおいて何度か値が急上昇した理由も、上記のほとんど動きに変化のない 3 フレームのデータを学習した場合に起こっていると考えられる。LossG_GAN のグラフが下がっているところが LossD_real と lossD_fake のグラフが上がっているところと一致しているため、ほとんど変化しない 3 フレームで生成した画像を生成したことで、lossG_GAN が一瞬だけ下がるという現象が起こったと考えられる。LossG_L1 は最初こそ減少していたものの、LossD_real と lossD_fake がほぼ 0 になったあたりでほとんど減少しなくなっているため、これ以上下がることはないと考えられる。

この実験の他にも、識別器 D の学習率を下げ、生成器 G の学習率を上げて学習させたり、生成器をあらかじめある程度 LossG_L1 のみを用いて学習させたうえで識別器と交互に学習させるなどを行ったが、最終的には LossD_real と LossD_fake 共に下がり、lossG_GAN が上がってしまい、1 枚目と 3 枚目を重ねたような画像を生成するという同じような結果となった。

以上より、pix2pix のみの実装では 2 枚目を生成するように生成器を学習させることは難しいと考えられる。

5.3　オプティカルフローを入力に用いた pix2pix の実装実験

5.3.1　結果

4.3.2 項で提案したオプティカルフローの入力を用いた pix2pix による実装をもとに、5.1 節で示したデータセットと pix2pix のサンプルコードを用いて実験を行った。この際、Generator に入力する tensor のチャネル数は 10、Discriminator に入力するチャネル数は 13 であるため、それぞれのネットワークの入力チャネル数を変更した。学習エポック数

は 100 エポックで実行した。

生成器 G が生成した画像を学習のエポック数 10, 50,100 の順に Figure 5.7〜Figure 5.9 に示す。Figure 5.7 から、10 エポック目では 1 枚目と 3 枚目を重ねたような画像の他に、動きに変化のある部分で少しぼやけたようになっている画像が生成されていることがわかる。Figure 5.8 から、50 エポック目では 1 枚目と 3 枚目を重ねたような画像がほぼなくなり、遠目では似たように見える画像が生成されていることがわかる。しかし、近くで見るとノイズが酷くかなりぼやけた画像となっている。Figure 5.9 から、100 エポック目では 50 エポックの時よりもぼやけが軽減され、より本物に近い画像がくっきりと生成されていることがわかる。ただ、動きが激しいフレームデータではまだ生成画像がぼやけてしまっている画像もある。

Figure 5.10〜Figure 5.13 に学習に使用した損失関数 (LossD_real, LossD_fake, LossG_GAN, LossG_L1) の変化を示す。Figure 5.10、Figure 5.11 から、lossD_real、lossD_fake 共に値が 0.7〜0.69 の範囲で安定していることがわかる。特に 50 エポックあたりからは外れ値がほぼ存在せず、かなり安定していることがわかる。同様に、Figure 5.12 から、LossG_GAN の値も 0.7〜0.69 の範囲で安定していることがわかる。Figure 5.13 から、LossG_L1 は安定して値が減少していっていることがわかるが、学習が進むにすれ減少率は下がっている。

5.3.2 考察

生成器 G が生成した画像について、pix2pix のみの実装実験の時は 1 枚目と 3 枚目の画像を重ねたような画像が生成され、画像がぼやけることがなかったが、10 エポック目の時点で 1 枚目と 3 枚目の画像を重ねたような画像の変化している部分がややぼやけていること、そして学習を進めるにつれてより鮮明に画像を生成できるようになっていることから、生成器 G は 2 枚目を生成するように正しく学習させることができたと考えられる。LossD_real、LossD_fake、LossG_GAN すべての値が 0.7〜0.69 あたりで安定して、かつ LossG_L1 が減少していっていることからも、学習は比較的良好だったと考えられる。

さらに学習を進めることで、Loss が安定しているうちは、生成器 G が生成する画像がより鮮明になると考えられる。しかし、100 エポック時点でも生成画像のぼやけが残っており、LossG_L1 の減少率が下がっていることから、これ以上学習を進めても意味がない可能性があるため、追加実験を行う。

Figure 5.7 Generated image at epoch 10.

Figure 5.8 Generated image at epoch 50.

Figure 5.9 Generated image at epoch 100.

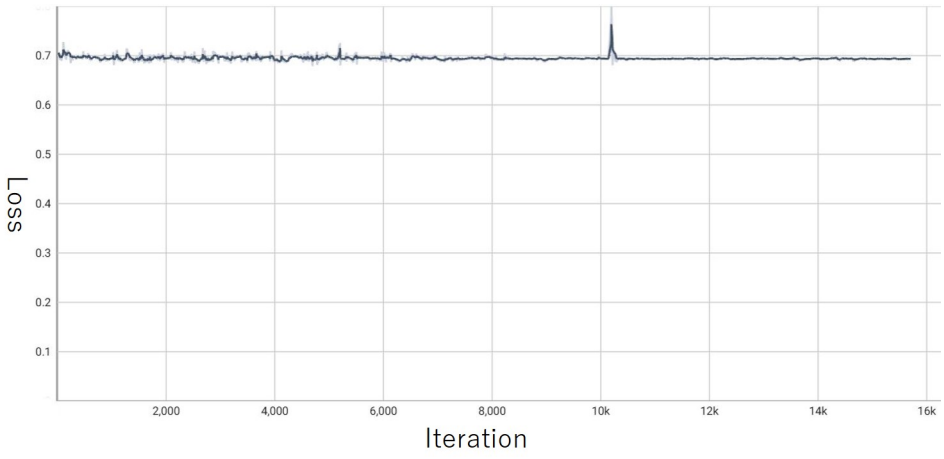

Figure 5.10 LossD_real.

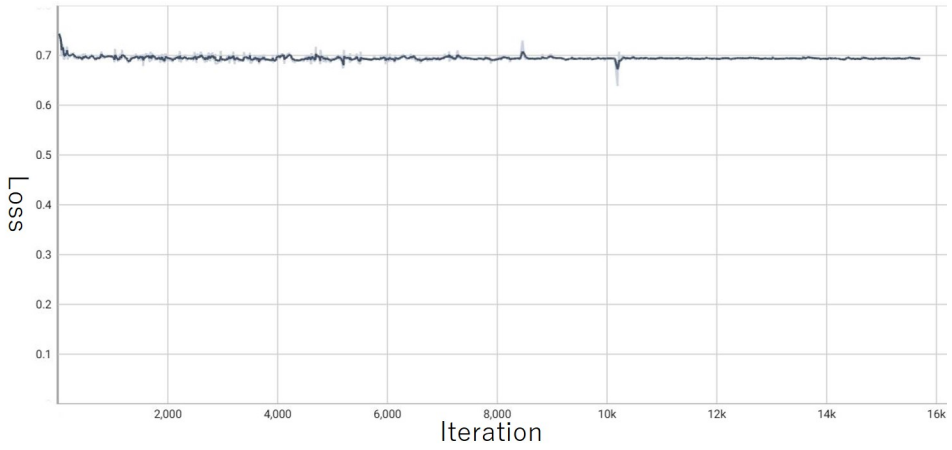

Figure 5.11 LossD_fake.

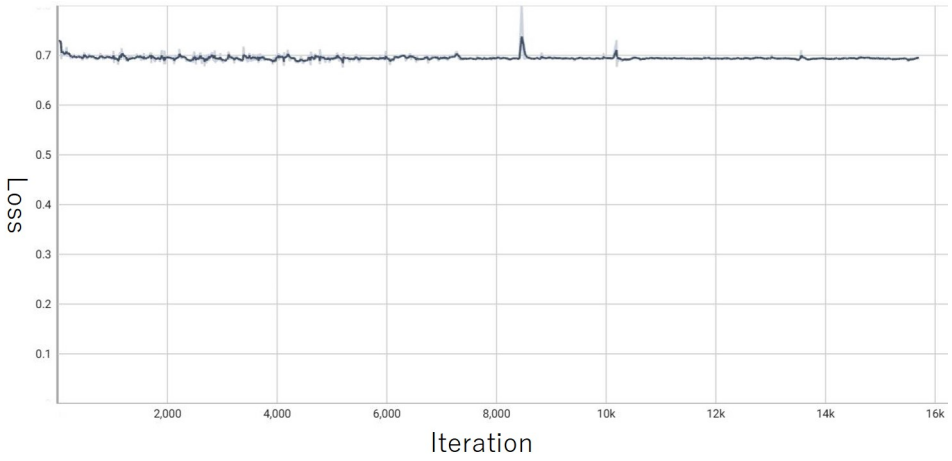

Figure 5.12 LossG_GAN.

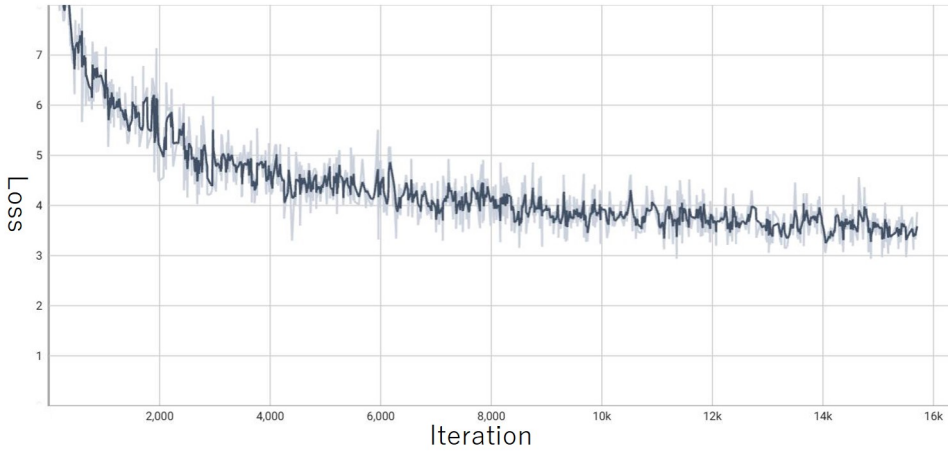

Figure 5.13 LossG_L1.

5.3.3　追加実験結果

初期条件を変えて、同様の学習実験を行った。生成される画像のぼやけをなくすために、LossG_L1 を算出する際にかけられている係数である lambda_L1 の値を 100 から 20 に変更した。また、前回の結果で LossD_real、LossD_fake 共に安定はしているが、逆にうまく学習できていない可能性を考えて、識別器 D の学習率を 0.0002 から 0.0001 に変更した。学習エポック数は前回同様 100 エポックで実行した。

生成器 G が生成した画像を学習のエポック数 10, 50,100 の順に Figure 5.14〜Figure 5.16 に示す。生成した画像からは、前回の実験から大きく変わったことはあまりなく、ぼやけが改善されることはなかった。むしろぼやけが少し激しくなったように見える。

Figure 5.17〜Figure 5.20 に学習に使用した損失関数 (LossD_real, LossD_fake, LossG_GAN, LossG_L1) を示す。Figure 5.17、Figure 5.18 から、40 エポックあたりまでは lossD_real、lossD_fake 共に値が 0.7 あたりで安定していたが、そのあとから急激に値が下がり始め、所々高くなることはあるが、0.05 あたりまで下がっていることがわかる。同様に、Figure 5.19 から、40 エポックあたりまでは LossG_GAN の値も 0.7 あたりで安定していたが、そのあとから急激に値が上がり始め、4〜6 あたりを推移していることがわかる。ただ、Figure 5.20 から、LossG_L1 は前回の実験と同様に安定して値が減少していっていることがわかる。

5.3.4　考察

生成器 G が生成した画像について、前回の実験で生成した画像より精度が上がることはなく、むしろ下がってしまった。この原因として、LossD_real、LossD_fake の値が 40 エポックあたりで急激に下がってしまい、それと同時に LossG_GAN の値が上がってしまって Generator が正しく学習できなくなってしまったことが考えられる。これは、Discriminator の学習率を下げることで、より正確に識別器 D が学習するようにした結果、逆に Discriminator が強くなりすぎてしまった。そのため、今回の実験で変更した学習率である 0.0001 は、適切な値ではなかったといえる。また、L1 損失の影響を抑えることで、画像のぼやけを軽減しより鮮明な画像を生成するために、Loss_L1 の係数である lambda_L1 の値を下げたが、結果としては軽減することはなく、効果的な結果は得られなかった。ただ、Loss_L1 の値は 40 エポック後も減少し続けており、生成器が正しく学習を最後まで行えていれば違う結果が得られた可能性があると考えられる。

Figure 5.14 Generated image at epoch 10.

Figure 5.15 Generated image at epoch 50.

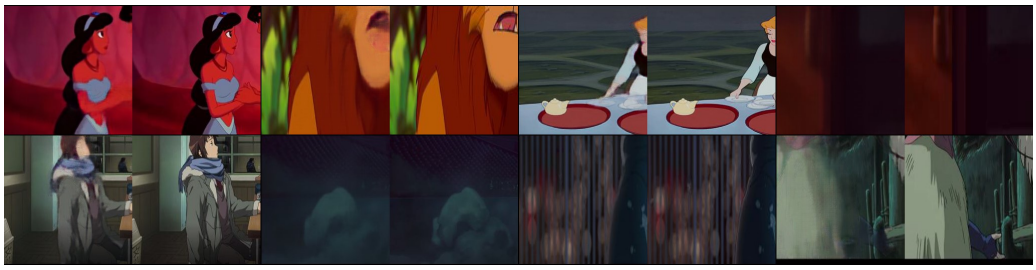

Figure 5.16 Generated image at epoch 100.

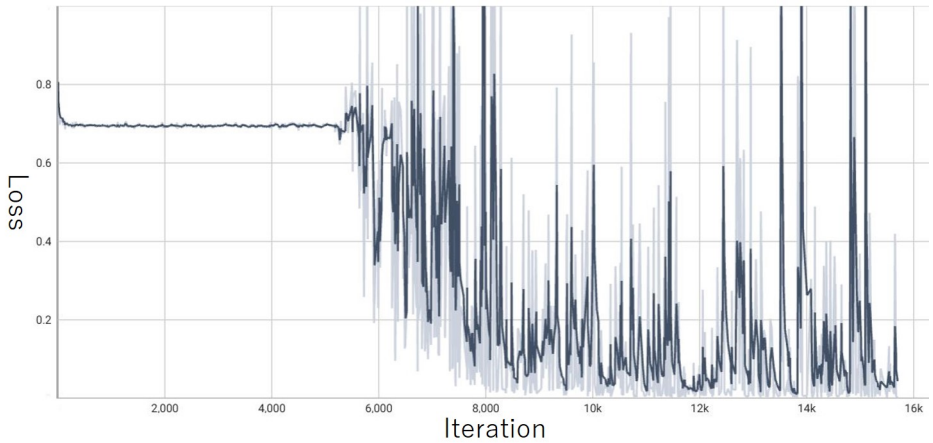

Figure 5.17 LossD_real.

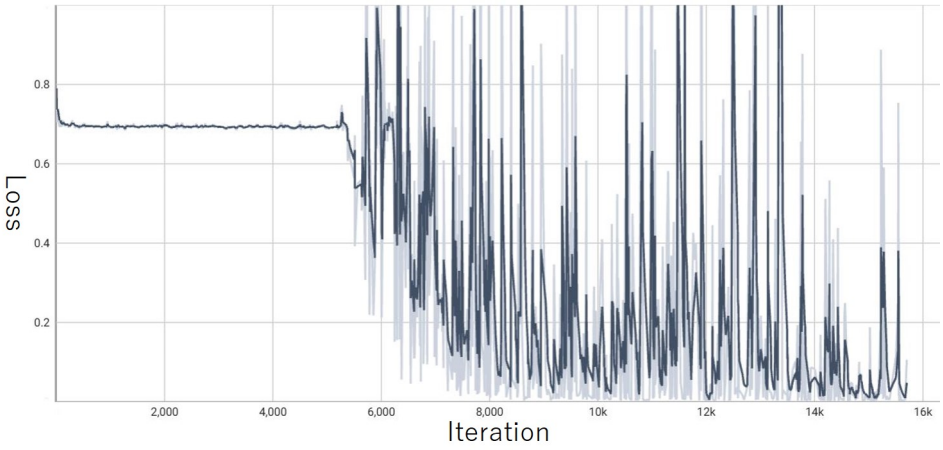

Figure 5.18 LossD_fake.

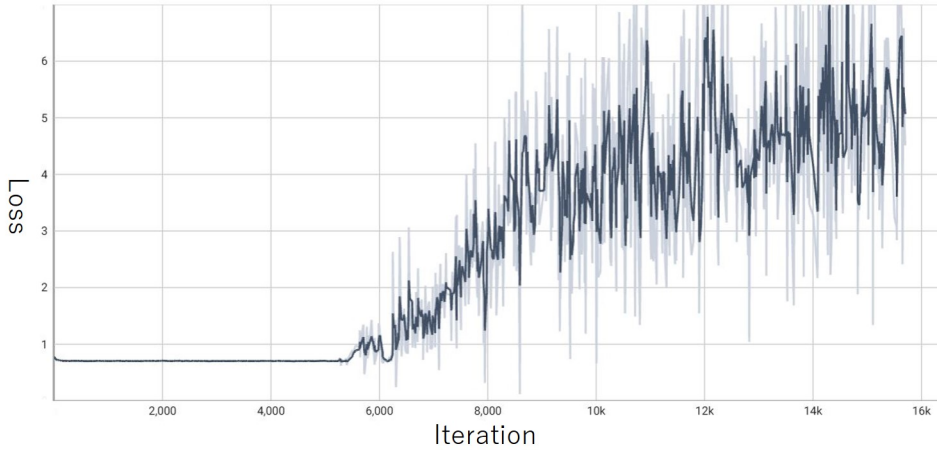

Figure 5.19 LossG_GAN.

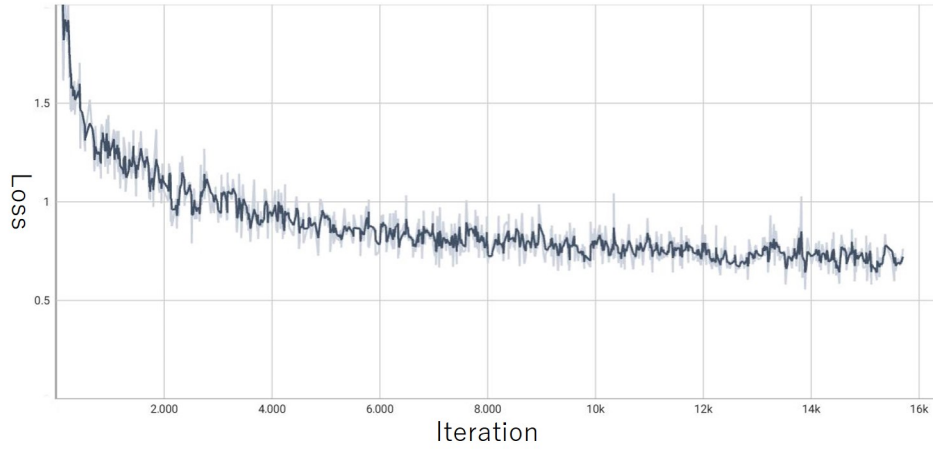

Figure 5.20 LossG_L1.

第6章　結論

本研究では、オプティカルフローを用いた動き推定と、pix2pix を用いた画像生成を組み合わせ、2D アニメーションの中割りの生成を行った。pix2pix のみでの実装とオプティカルフローを用いた実装の実験結果を比べた際に、生成器 G が 2 枚目を生成しようと学習しているのが後者のみであり、オプティカルフロー推定を GAN の学習に用いる有用性を示すことができた。

しかし、使用した pix2pix のネットワークと学習環境の制限により、学習させる際に使用する画像サイズが 256 × 256 であったため、フレームの正確なディテールを表現できず、オプティカルフローの推定精度が落ち、結果として pix2pix の生成器が学習する際の精度が落ちてしまって、識別器が強くなりすぎてしまうことに一部起因したとされる。また、使用したデータセットについて、「人が動いているフレームに限定する」や「ほぼ変化のない 3 フレームのデータを除外する」などの選別せずにすべて使用したことから、学習内容が汎化しすぎてしまい、学習がうまく進まないことや、生成器がうまく動きを捉えられておらず、生成画像がぼやけるなどが起こったと考えられる。今回の追加実験でのパラメータ変更では効果的な結果が得られなかったが、値次第ではより良い結果になる可能性が考えられる。

今後の展望として、パラメータを変えて再度実験し、最適なパラメータを探すこと、使用する画像を高画質にして、ネットワークをさらに複雑にすることで、より様々なフレームに対応できるようにすること、そして画像のぼやけをできるだけ無くして、より鮮明な画像が生成できることを目指したい。

参考文献

1) 毛利拓也, 大郷友海, 嶋田宏樹, 大政孝充, むぎたろう, 寅蔵, もちまる, GAN ディープラーニング実装ハンドブック, 秀和システム出版, 2021.

2) Alec Radford, Luke Metz, Soumith Chintala (2015), Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
https://arxiv.org/abs/1511.06434

3) Vincent Dumoulin, Francesco Visin, A guide to convolution arithmetic for deep learning, 2018
https://arxiv.org/pdf/1603.07285

4) Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros, Image-to-Image Translation with Conditional Adversarial Networks, 2018
https://arxiv.org/pdf/1611.07004

5) オプティカルフロー (Optical Flow) アクセス日：2025 年 2 月 4 日
`https://labs.eecs.tottori-u.ac.jp/sd/Member/oyamada/OpenCV/html/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html?highlight=optical%20flow`

6) オプティカルフロー (Optical Flow) とは？基本的な概要から仕組み、用途、欠点までを徹底解説！ アクセス日：2025 年 2 月 4 日
https://ai-market.jp/purpose/video-analysis-optical-flow/

7) ATD-12K アクセス日：2024 年 12 月 14 日
https://paperswithcode.com/dataset/atd-12k

謝辞

本研究を進めるにあたって、ご多忙中にも関わらず多大なご指導をしていただきました出口利憲先生、また、共に勉学に励んだ同研究室のメンバーに厚く御礼申し上げます。