Multi Style Transfer: 複数のスタイルの任意重み合成による モバイル上でのリアルタイム画風変換

丹野　良介†　　柳井　啓司†

† 電気通信大学 大学院情報理工学研究科 情報学専攻
〒 182–8585 東京都調布市調布ヶ丘 1-5-1
E-mail: †{tanno-r,yanai}@mm.inf.uec.ac.jp

あらまし　近年，画像の形状を保持したまま，画風を変化させることができる Neural Style Transfer が芸術的な画像をコンピュータが生成できるとして注目を集めている．この手法では，誤差逆伝播を複数回繰り返すため，画像の生成に時間がかかる．一方，特定のスタイルの変換を順伝播で行う CNN を学習しておくことで，非常に高速に画像を変換可能にする手法を Johnson ら [1] が提案した．しかし，この手法では，1 つのモデルで単一のスタイルしか学習できず，消費メモリの増大，学習に時間がかかる，変換の質が画像の質に依存する，などの問題点があった．そこで本研究では，ネットワークを拡張し，単一のモデルで複数のスタイルを任意の重みで合成して，リアルタイムで合成したスタイルを画像に転送する手法を提案し，それをモバイルアプリとして実装した．

キーワード　スタイル変換，モバイル，アプリケーション，Convolutional Neural Network，Neural Style Transfer

1. はじめに

2015 年，Gatys ら [2, 3] は Convolutional Neural Network(CNN) を用いて，2 種類の画像を合成するアルゴリズム “Neural Style Transfer”(スタイル変換) を提案した．この手法では画像のコンテンツ (形状) を保持したまま，スタイル (画風) を変化させることができ，例えば，写真を絵画風に変換可能であるなど，芸術的な画像をコンピュータが生成できるとして注目を集めている．Gatys らの手法では，CNN の特徴マップ間の相関行列であるグラム行列により表現されたスタイル行列を導入している．これにより，コンテンツ画像の信号が CNN の各層を順伝播している間に劣化する情報を，スタイル画像から抽出されたスタイル情報により置き換え，入力に与えたスタイル画像と同じスタイルに変換されたコンテンツ画像を生成することができる．

しかしながら，Gatys らの手法では，順伝搬及び誤差逆伝播を複数回繰り返すため，GPU を使う場合でも，画像の生成に数秒程度要するなど，処理に時間がかかりすぎる．この問題を解決するために，feed forward のみを使ってスタイル変換を高速に行う研究が世界中で多くなされている．その中の 1 つに Johnson ら [1] の研究が存在する．

Johnson ら [1] は feed forward style transfer network として ConvDeconvNetwork を学習する “perceptual loss” を提案し，特定のスタイルの変換を順伝播で行う CNN を学習しておくことで，非常に高速に画像を変換可能にする．しかし，この手法では，1 つのモデルで単一のスタイルしか学習できず，消費メモリの増大，学習に時間がかかる，変換の質が画像の質に依存する，などの問題点があった．

そこで本研究では，ネットワークを拡張し，単一のモデルで複数のスタイルを任意の重みで合成して，リアルタイムで合成したスタイルを画像に転送する手法を提案し，それをモバイルアプリとして実装した．図 1 として今回実装したリアルタイム画風変換アプリのデモ画面を示す．

図 1　リアルタイム画風変換アプリの動作例

2. 関連研究

本研究では CNN を用いたモバイル上でのリアルタイム動作を目的とするため，主に高速スタイル変換及び CNN の高速化，CNN のモバイルへの効率的な実装について概観する．

2.1 画風変換アルゴリズム

Gatys らが提唱した Convolutional Neural Network(CNN) を用いて 2 種類の画像を合成するアルゴリズム “Neural Style Transfer”(スタイル変換) を発端として，この分野に関する研究は急速に進んでいる．Gatys らの手法では，CNN の特徴マップ間の相関行列を式 1 で表されるグラム行列 $G^l \in R^{N_l \times N_l}$ により表現されたスタイル行列を導入し，

$$G^l_{i,j} = \sum_k F^l_{i,k} F^l_{j,k} \tag{1}$$

コンテンツ画像と生成される画像をそれぞれ，$\vec{p}, \vec{x}$ として，式

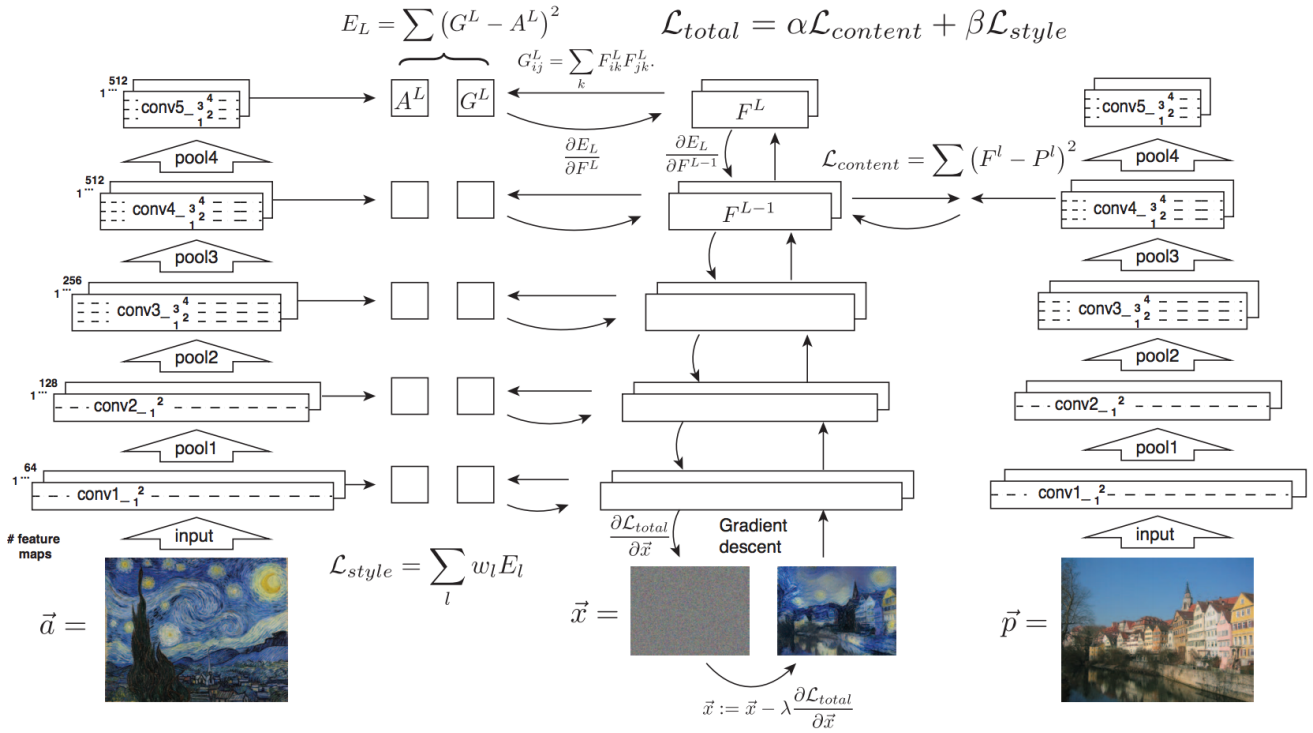

図 2　順伝搬及び誤差逆伝播を複数回繰り返すため計算コストが高い ([2] より引用)

2, 式 3, 式 4 を用いて，

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F^l_{i,j} - P^l_{i,j})^2 \tag{2}$$

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{i,j} - A^l_{i,j})^2 \tag{3}$$

$$L_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l \tag{4}$$

最小化したい損失関数を定義すると，式 5 として表現される．

$$L_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha L_{content}(\vec{p}, \vec{x}) + \beta L_{style}(\vec{a}, \vec{x}) \tag{5}$$

コンテンツ画像の信号が CNN の各層を順伝播している間に劣化する情報を，スタイル画像から抽出されたスタイル情報により置き換え，入力に与えたスタイル画像と同じスタイルに変換されたコンテンツ画像を生成することができる．

しかしながら，Gatys らの手法では，図 2 にあるように feed forward 及び back propagation を複数回繰り返すため，GPU を使う場合でも，画像の生成に数秒程度要するなど，処理に時間がかかりすぎる．到底モバイル上でのリアルタイム動作は叶わない．

この問題を解決するために，feed forward のみを使ってスタイル変換を高速に行う研究が世界中で多くなされている．

Johnson ら [1] は図 3 に挙げるような feed forward style transfer network として，Downsampling 層，複数の Residual block，Upsampling 層から構成される ConvDeconvNetwork f_w

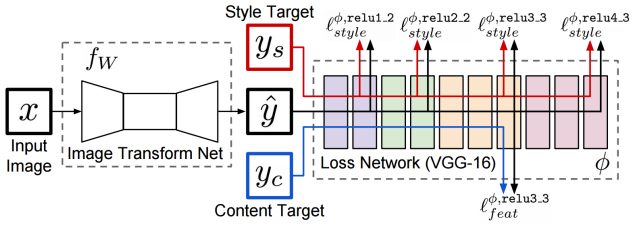

図 3　特定のスタイルの変換を feed forward で行う CNN を学習することで高速スタイル変換を実現 ([1] より引用)

を学習する “perceptual loss” を提案した．特定のスタイルの変換を feed forward で行う CNN を学習しておくことで，非常に高速に画像を変換可能にする．しかし，この手法では，スタイル変換ネットワーク f_w はスタイル毎に学習する必要があり，1 つのモデルで単一のスタイルしか表現することができない．そのため，消費メモリの増大，学習に時間がかかる，変換の質が画像の質に依存する，などの問題点があった．

2.2　CNN の効率的なモバイル実装

本研究では CNN を用いたスタイル変換アルゴリズムをモバイル端末上に実装し，モバイル端末上で処理が完結しリアルタイムに動作することを必要とする．昨今，モバイル端末の高性能化に伴い高演算性を必要とするアルゴリズムの動作が可能となってはきているが，アルゴリズムの更なる複雑化・高演算性の要求など，端末側に要求する処理能力が年々増してきている．更に，CNN を実装するには内部処理に畳込み演算が多数存在することから，畳込み処理の高速化は重要である．よって，CNN を効率的に実装するためには高速化の工夫は必須であり，この分野は世界中で盛んに研究が行われている．例えば，処理

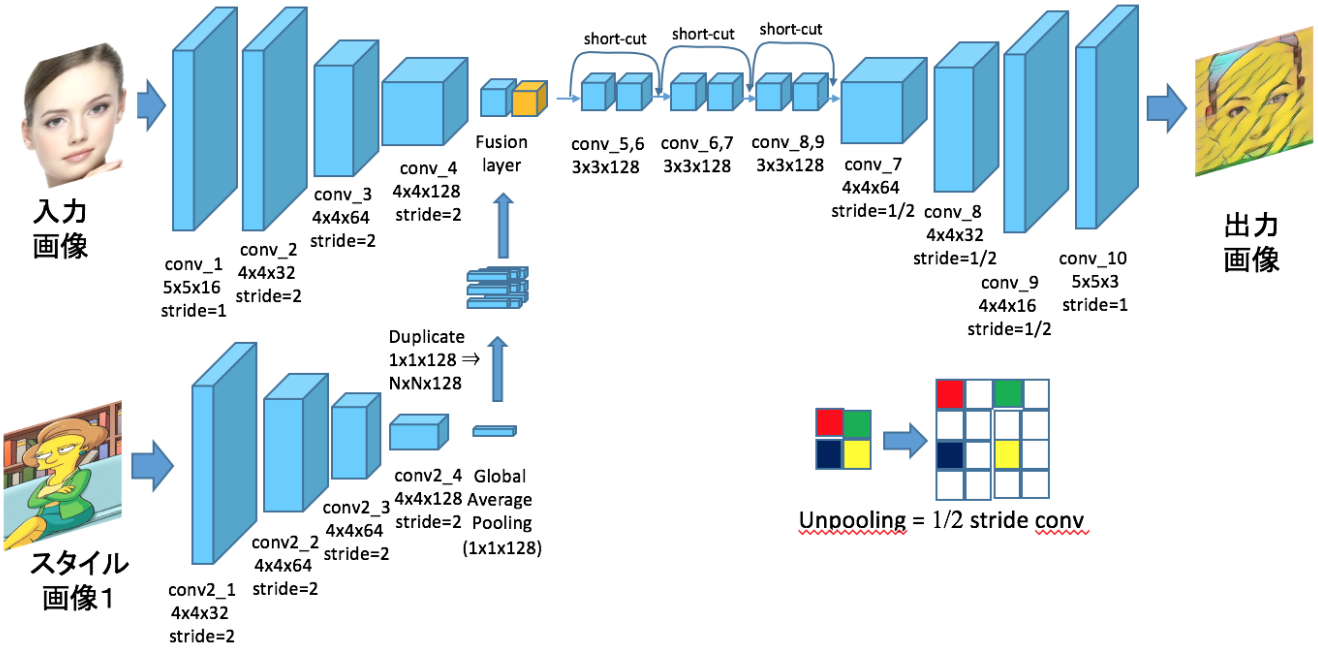

図 4　スタイル画像の入力を伴う ConvDeconvNetwork

に大部分の時間を要する畳込み層の計算の工夫に関する研究として [4–6] などがある．また，学習時のパラメーターにかかる重みに着目した高速化の研究として [7,8] などがある．

2.2.1　畳込み層の計算の工夫に関する研究

Liu ら [4] はスパース分解による DNN のパラメーター圧縮を行っている．また，CPU 上での効率的なスパース行列演算アルゴリズムとして Sparse Convolutional Neural Networks(SCNN) を提案し，物体検出に SCNN モデルを適用することで大幅な高速化を実現している．Jaderberg ら [5] は低ランク近似やフィルタ近似で，文字認識タスクにおいて精度低下 1%未満で 4.5 倍の高速化を実現している．Gong ら [6] は CNN をモバイル端末でも利用できるように，Vector 量子化によるパラメーター圧縮を行っている．1000 種類カテゴリ分類で，僅か 1%の損失に抑えて 16~24 倍の圧縮を達成する研究成果を残している．

2.2.2　学習時のパラメーター重みに着目した高速化の研究

Lin ら [7] は畳込み層の乗算をビットシフトに置き換える手法として “quantized back propagation” を提案し，先行研究である “BinaryConnect” を上回る性能を見せた．Courbariaux ら [8] は順方向及び逆方向伝搬時における重みを -1 or 1 に二値化することで，本来必要な演算の 2/3 を除去し，トレーニング時間を 3 倍にするという新しい高速化手法として “BinaryConnect” を提案している．

2.2.3　本研究の位置付け

本研究では，Johnson ら [1] が提案したネットワークを拡張し，単一のモデルで複数のスタイルを任意の重みで合成して，リアルタイムで合成したスタイルを画像に転送する手法を提案し，それをモバイルアプリとして実装した．また，CNN の具体的な実装の詳細については [9] に記載してある．

3.　システム概要

我々が提案する Neural Style Transfer Network を図 4 として以下に示す．本ネットワークは Convolutional Layer のみで構成されるため，入力画像の大きさは任意である．一般には，最後の 3 レイヤーは Convolutional Layer の逆演算を行う Deconvolutional Layer を用いることが多い．しかし，本研究ではモバイル上への実装を考慮し，モバイルでの高速 GEneral Matrix Multiplication(GEMM) 演算をそのまま利用できるように，図 4 に示す Unpooling を行い，特徴マップを縦横それぞれ 2 倍に拡大してから Convolution を行うことで，stride0.5 の Convolution を実現し，Deconvolutional Layer の代わりとした．一般には，これらの操作は等価である．また，今回作成した Neural Style Transfer アプリの GUI 構成を図 5 として以下に示す．画面右側にあるスライドバーを利用して複数のスタイルの重みを任意に合成し，リアルタイムに画風変換することが可能である．

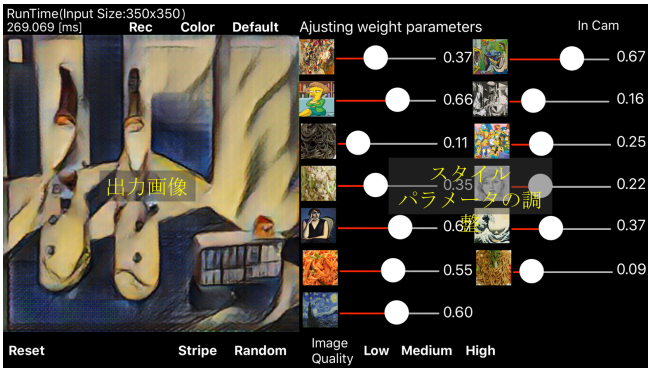

図 5　画風変換アプリの GUI 構成画面

4. システム詳細

4.1 Multi Style Transfer

図 6 に我々が提案するスタイル入力を伴う style transfer network を示す．学習の方法は [1] と基本的には同じである．よって，学習の詳細を知りたい場合は [1] を参照願いたい．本論文と [1] とで異なる点は以下のとおりである．

（1） ConvDeconvNetwork の縮小

（2） Fusion Layer の追加による複数の任意重みを合成可能 ([10] 参考)

また，ConvDeconvNetwork の縮小は主に以下のことを行うことで，モバイル上での計算コストの削減を行っている．実装上の高速化手法は，

（1） Deep Neural Network を直接 C コードに変換し，コンパイラ的に実行.

（2） Multithread 化による NEON/BLAS の効率的な利用.

（3） CNN に掛かる演算の可能な限りの事前計算の実行.

などであり，実装の詳細については [9] や [11] を参照願いたい．この時，[1] と比較してネットワークを縮小しているが，出力画像の質にあまり影響を与えない結果が得られる．

（1） down-sampling layer と up-sampling layer を追加

（2） 最初と最後の conv layer のカーネルを 9x9 を 5x5 に変更

（3） 5 つの residual element を 3 つに変更

表 1 [1] とのネットワーク構成の比較

Johnson [1]	Ours
Reflection Padding (40x40)	5x5x16 conv, stride 1
9x9x32 conv, stride 1	4x4x32 conv, stride 2
3x3x64 conv, stride 2	4x4x64 conv, stride 2
3x3x128 conv, stride 2	4x4x128 conv, stride 2
Residual block, 128 filters	Fusion layer
Residual block, 128 filters	Residual block, 128 filters
Residual block, 128 filters	Residual block, 128 filters
Residual block, 128 filters	Residual block, 128 filters
Residual block, 128 filters	4x4x64 conv, stride 1/2
3x3x64 conv, stride 1/2	4x4x32 conv, stride 1/2
3x3x32 conv, stride 1/2	4x4x16 conv, stride 1/2
9x9x3 conv, stride 1	5x5x3 conv, stride 1

5. 変 換 例

本アプリを用いて画風変換を行った例は図 7 として示した．

6. おわりに

本研究では，ネットワークを拡張し，単一のモデルで複数のスタイルを任意の重みで合成して，リアルタイムで合成したスタイルを画像に転送する手法を提案し，それをモバイルアプリとして実装した．

今後の課題としては，領域分割 [12] や物体検出 [13,14] などを組み合わせて，モバイル上での部分的な画風変換システムなどを構築したい．そのために，更なる高速化やスタイルを合成する際のスタイル重みの最適な組合せを調べるなどは重要である．

文 献

[1] J. Johnson, A. Alahi, and L.F. Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proc. of European Conference on Computer Vision*, 2016.

[2] L.A. Gatys, A.S. Ecker, and M. Bethge. Image Style Transfer Using Convolutional Neural Networks. In *Proc. of IEEE Computer Vision and Pattern Recognition*, 2016.

[3] L.A. Gatys, A.S. Ecker, and M. Bethge. A Neural Algorithm of Artistic Style. In *Proc. of arXiv:1508.06576*, 2015.

[4] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky. Sparse convolutional neural networks. In *Proc. of IEEE Computer Vision and Pattern Recognition*, 2015.

[5] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proc. of arXiv: 1405.3866*, 2014.

[6] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. In *Proc. of International Conference on Learning Representations*, 2015.

[7] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio. Neural networks with few multiplications. In *Proc. of arXiv:1510.03009*, 2015.

[8] M. Courbariaux, Y. Bengio, and J. P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, 2015.

[9] K. Yanai, R. Tanno, and K. Okamoto. Efficient Mobile Implementation of A CNN-based Object Recognition System. In *Proc. of ACM Multimedia*, 2016.

[10] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, Vol. 35, No. 4, 2016.

[11] R. Tanno and K. Yanai. Caffe2c: A framework for easy implementation of cnn-based mobile applications. In *Proc. of International Workshop On Mobile Ubiquitous Systems, Infras- tructures, Communications, And AppLications (MUSICAL 2016)*, 2016.

[12] W. Shimoda and K. Yanai. Distinct Class-specific Saliency Maps for Weakly Supervised Semantic Segmentation. In *Proc. of European Conference on Computer Vision*, 2016.

[13] J. A. Divvala, S. A. Girshick, R. A. Farhadi, and A. . You Only Look Once: Unified, Real-Time Object Detection. In *Proc. of IEEE Computer Vision and Pattern Recognition*, 2016.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. In *Proc. of European Conference on Computer Vision*, 2016.

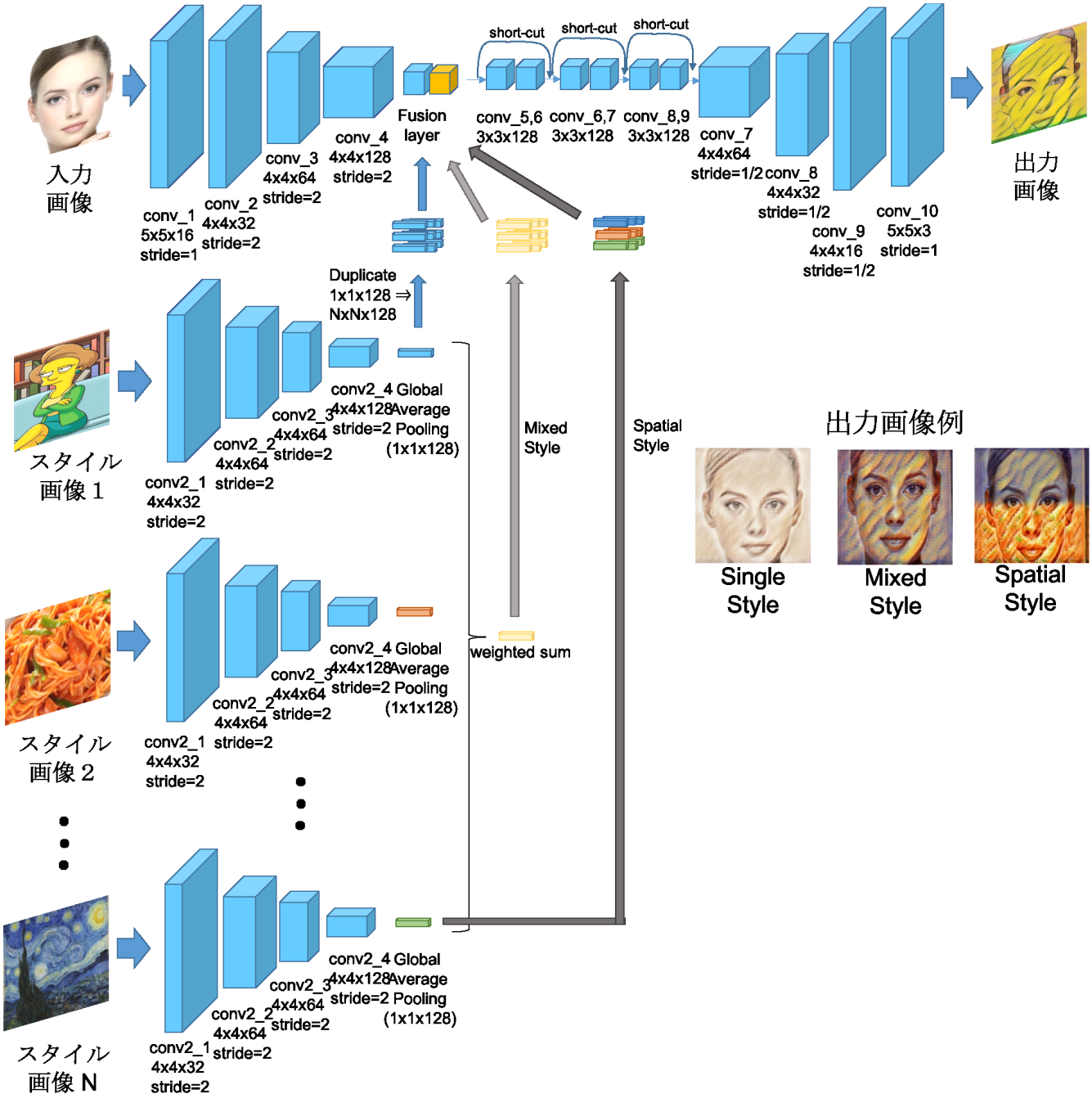

図 6　Multi Style Transfer 及び Spatial Style Transfer の概要

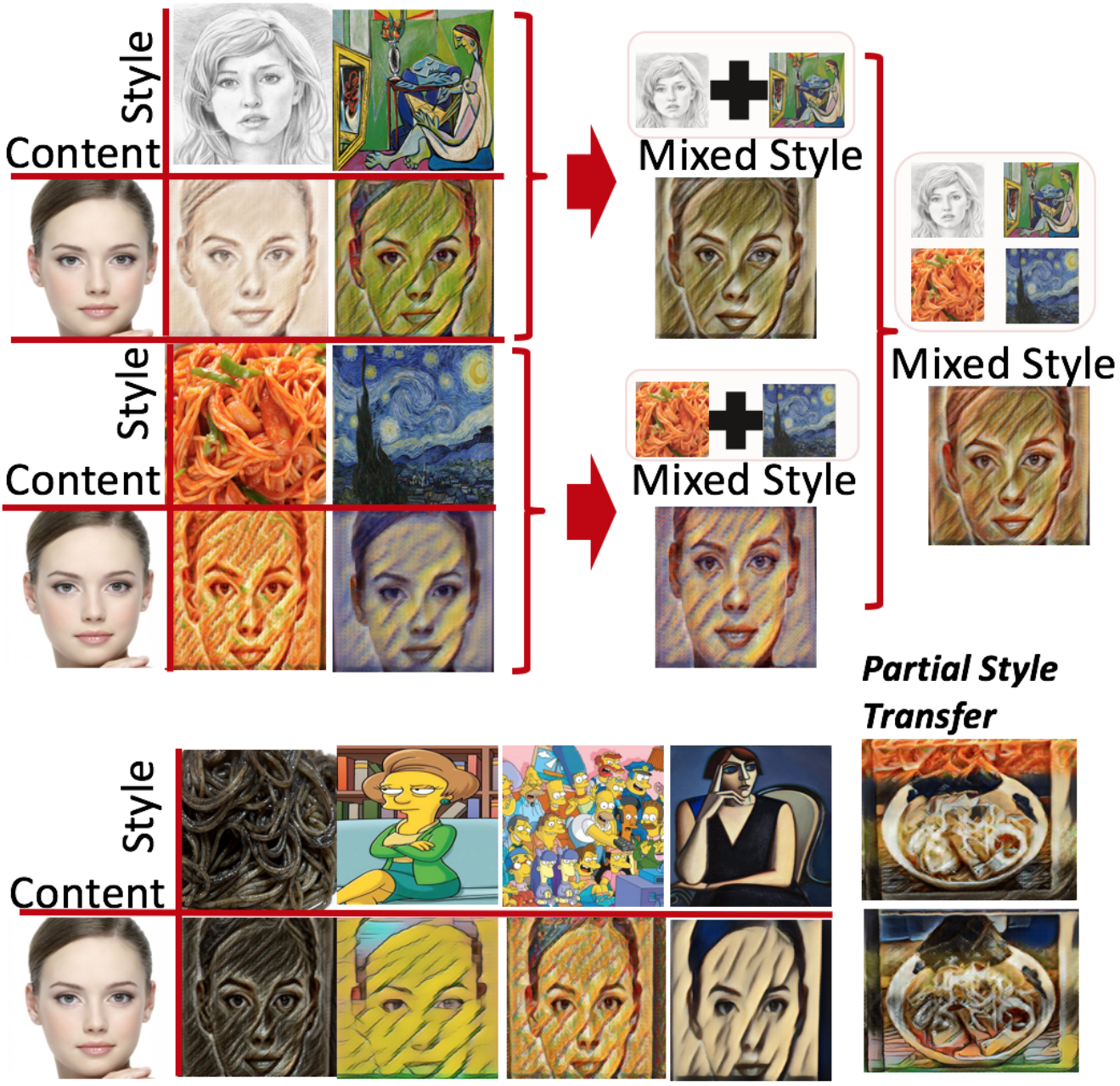
Style
Content
Mixed Style
Style
Content
Mixed Style
Mixed Style
Partial Style Transfer
Style
Content

図 7　スタイル変換の例