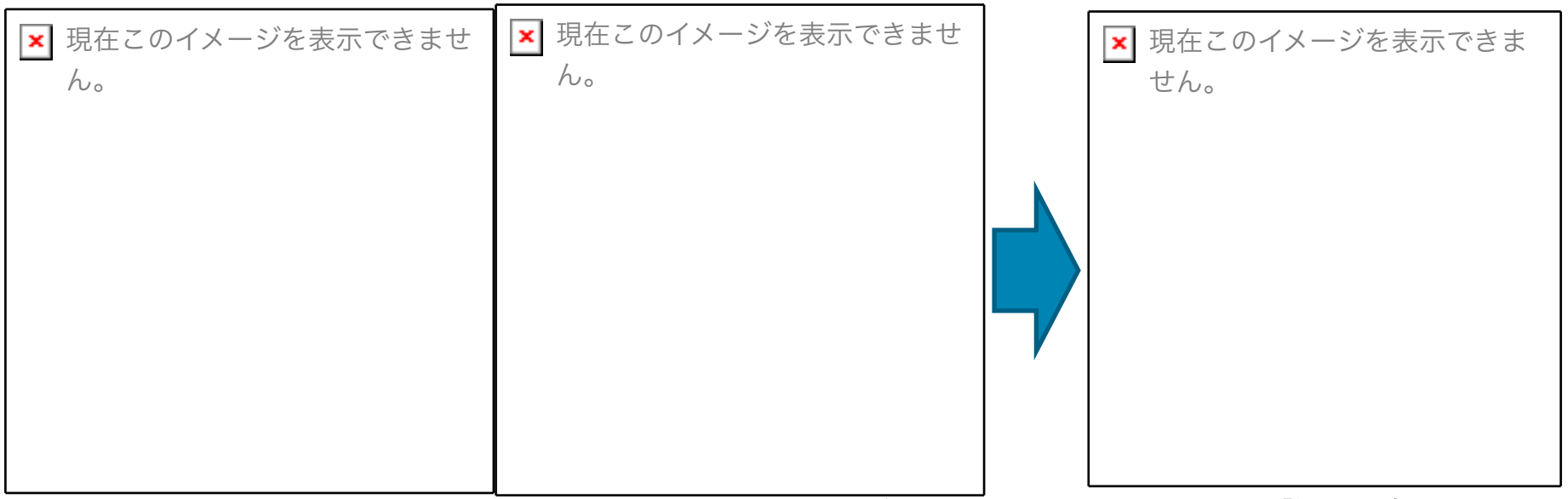
複数スタイルの融合と部分的適用を可能とするMulti-style Feed-forward Networkの提案

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

Style Transferとは

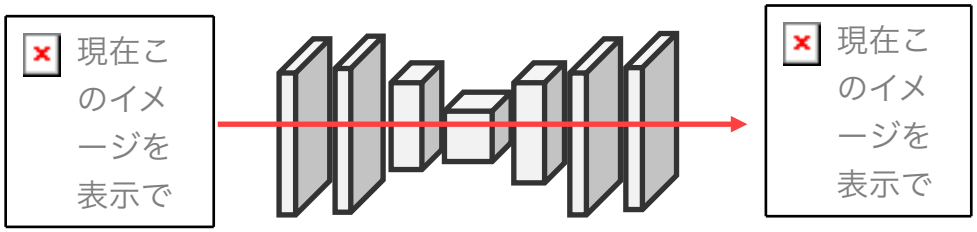
- 画風を変換するアルゴリズムのこと(2015年8月に公開)
  - 2枚の画像を入力として，片方をコンテンツ画像，片方をスタイル画像とする
  - コンテンツ画像に書かれた物体の配置をそのままにして，画風をスタイル画像に変換した画像を生成



本研究の目的

「Neural Style Transferをモバイル上に実装」

- 既存手法
  - **1つのモデルで1つのスタイル**
  - **学習に時間を要する**
  - **消費メモリの増大**



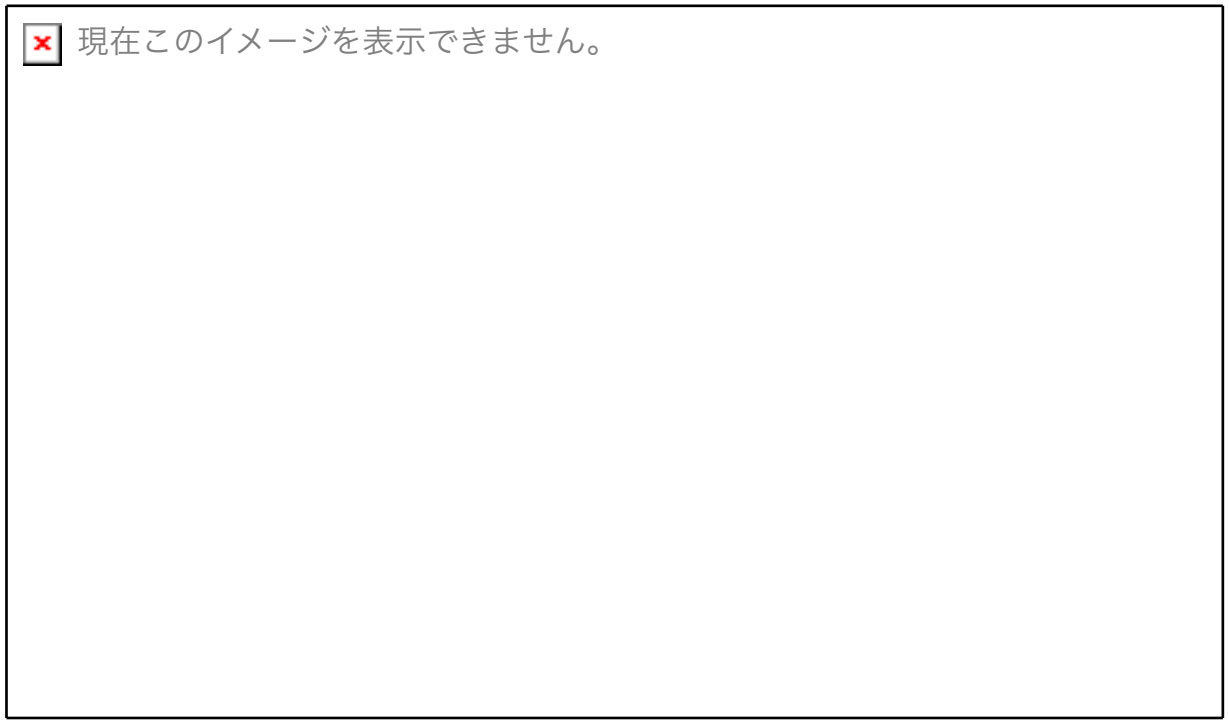
- 提案手法
  - **1つのモデルで複数のスタイルを変換可能**
  - **モバイル上でのリアルタイム画風変換**

関連研究

現在このイメージを表示できません。

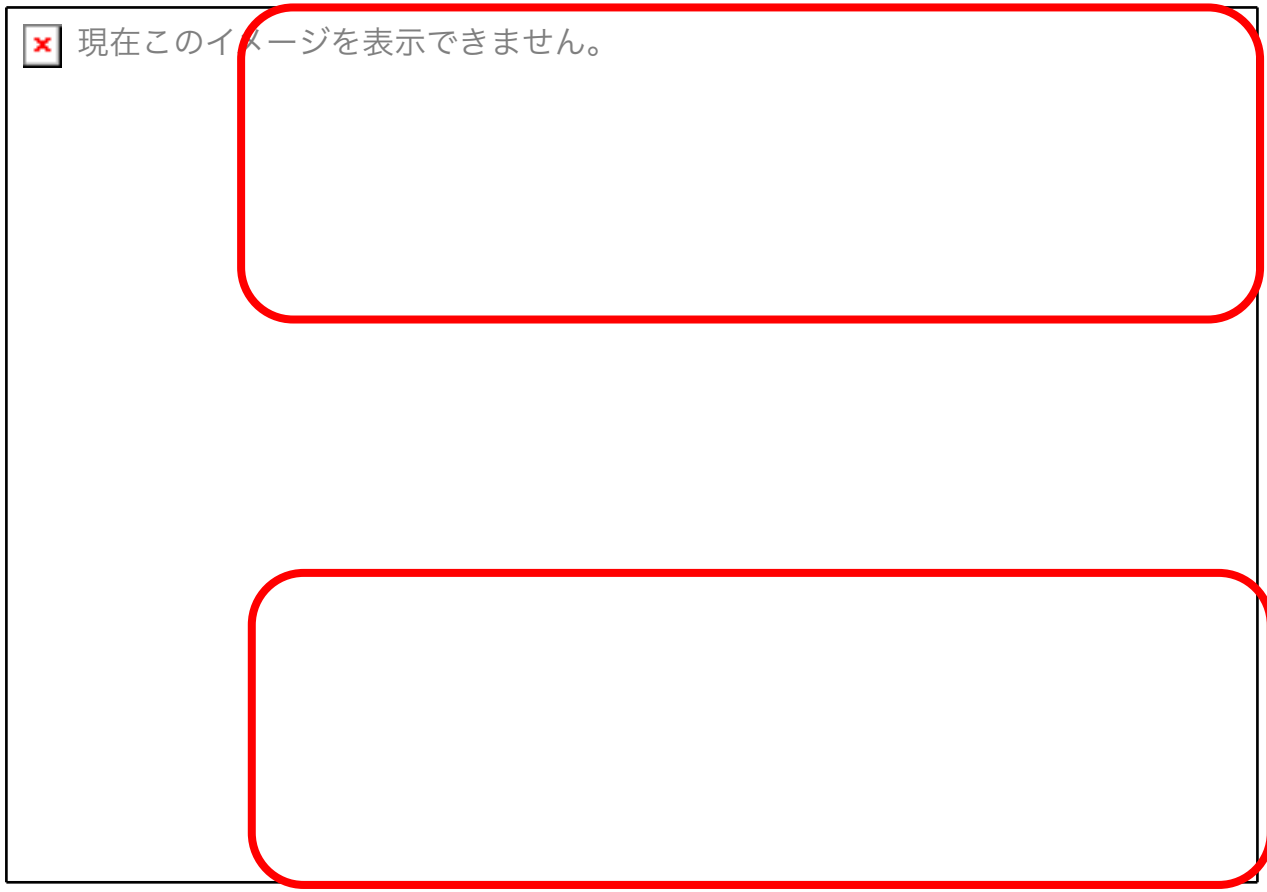
現在このイメージを表示できません。

- Neural Style Transferを提案 [Gatys+ 2015]
  - 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
  - AIによる芸術的画像の生成



各レイヤの特徴マップから画像を再構築

conv1 conv2 conv3 conv4 conv5

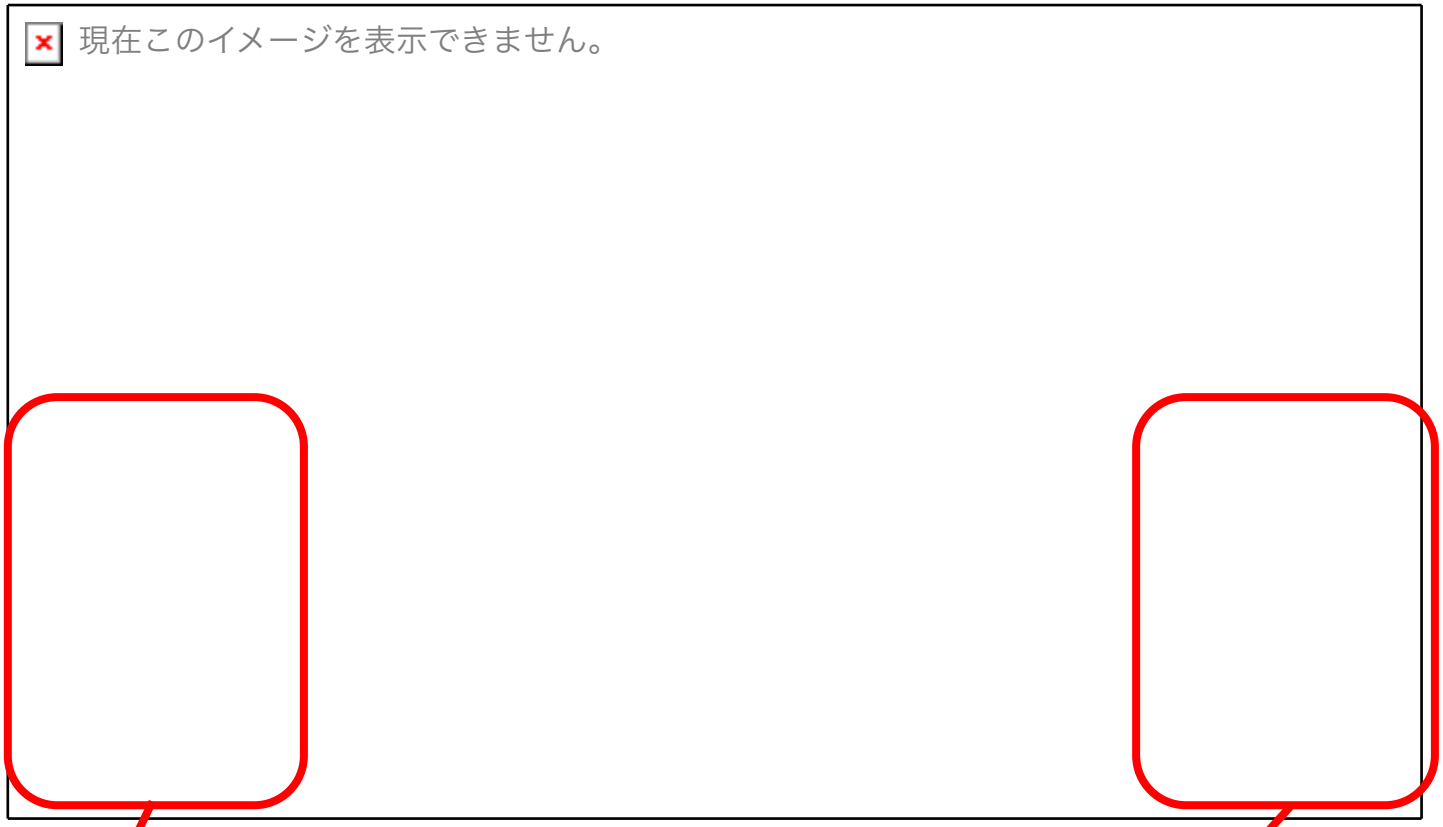


スタイル画像再構築

コンテンツ画像再構築

conv1 conv2 conv3 conv4 conv5

コンテンツ画像の特徴マップ再構築



浅いレイヤ
・正確な元画像の情報を保持

深いレイヤ
・空間的な位置関係はそのまま
→画像が持つ重要な情報を保持.
・正確な形などの情報は弱まる

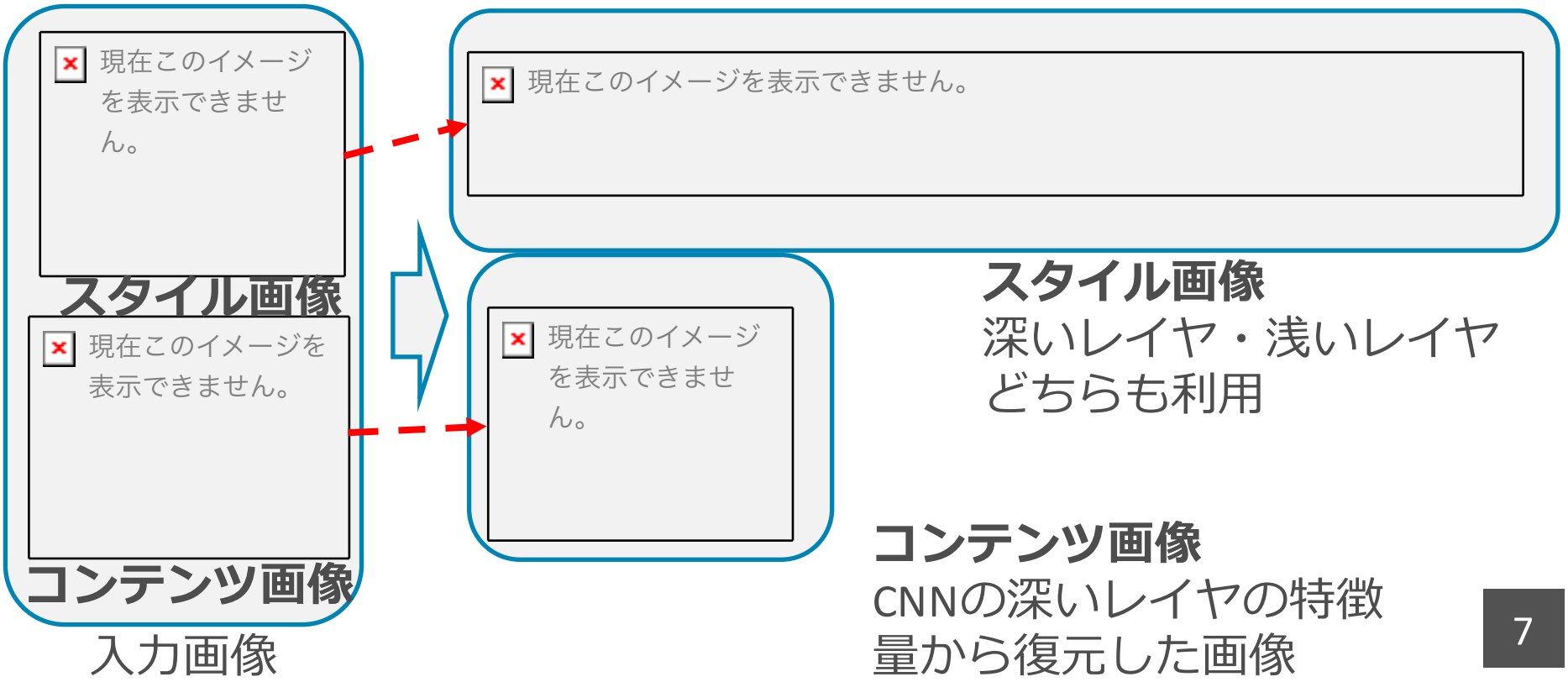
スタイル画像の特徴マップ再構築

浅いレイヤ
・画像の細かなパターン情報を保持

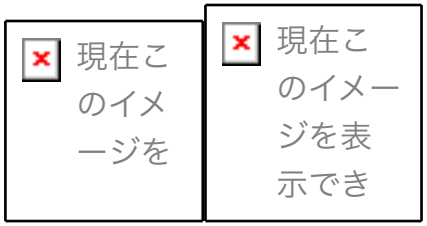
深いレイヤ
・画像の大きめの空間パターンを保持

Gatysらの手法のポイント

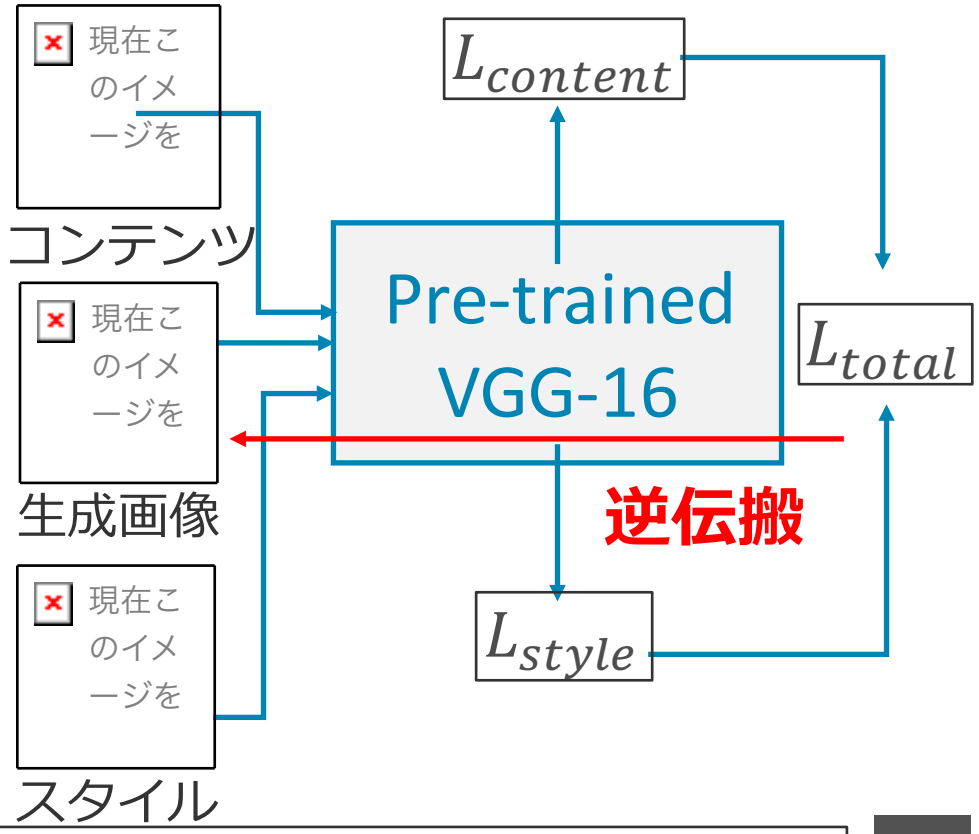
- **情報が弱まっている部分**を，**別の画像の画風(スタイル)に変換**すれば，**コンテンツ画像の形状を保持**したまま，別のスタイルに画像を変換できるのでは？



Gatysらの手法まとめ



- 損失関数に学習済みCNNを利用
- 生成画像は一様乱数で初期化
- 順伝搬＆逆伝搬を反復



- **問題点**
  - 生成に**時間**がかかる
  - GPU利用で**数十秒**程度

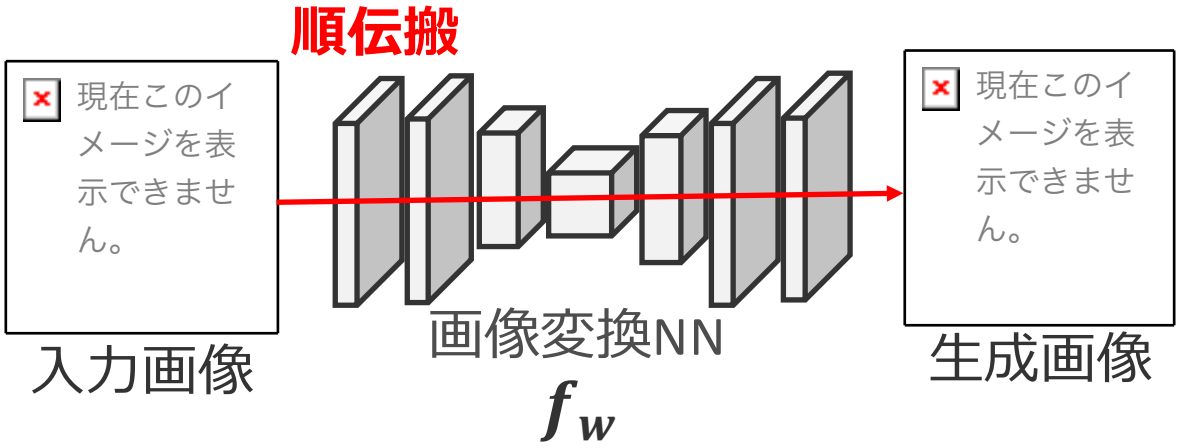
$$\text{argmin } L_{total} = \alpha L_{content} + \beta L_{style}$$

関連研究

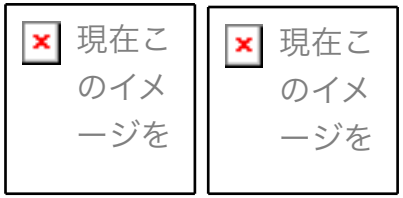
現在このイメージを表示できません。

現在このイメージを表示できません。

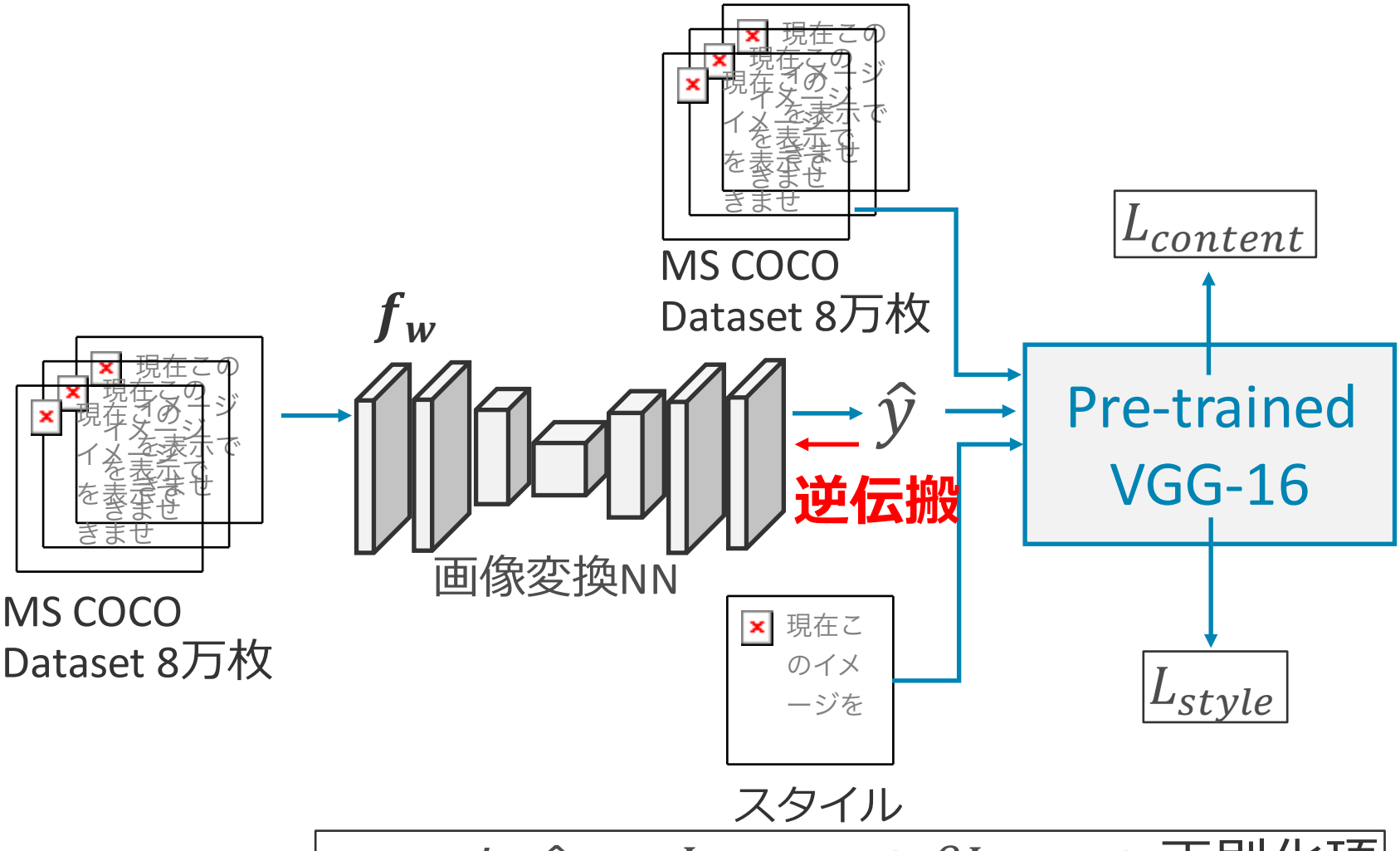
- 順伝搬でスタイル変換できる手法を提案 [Johnson+ 2016]
  - **入力画像のスタイルを変換するNN (f_w) を学習**
  - 生成時は順伝搬のみなので高速



Johnsonらの手法



- MSCOCO 8万枚 + スタイル画像でf_wを学習



$$\mathrm{argmin}\ \hat{y} = \alpha L_{content} + \beta L_{style} + \text{正則化項}$$

Johnsonらの手法まとめ

現在このイメージを

現在このイメージを

- 特定スタイルを変換可能なCNNを学習
- 順伝搬でスタイル変換可能
  - 非常に高速に画像生成可能に

現在このイメージを表示できません。

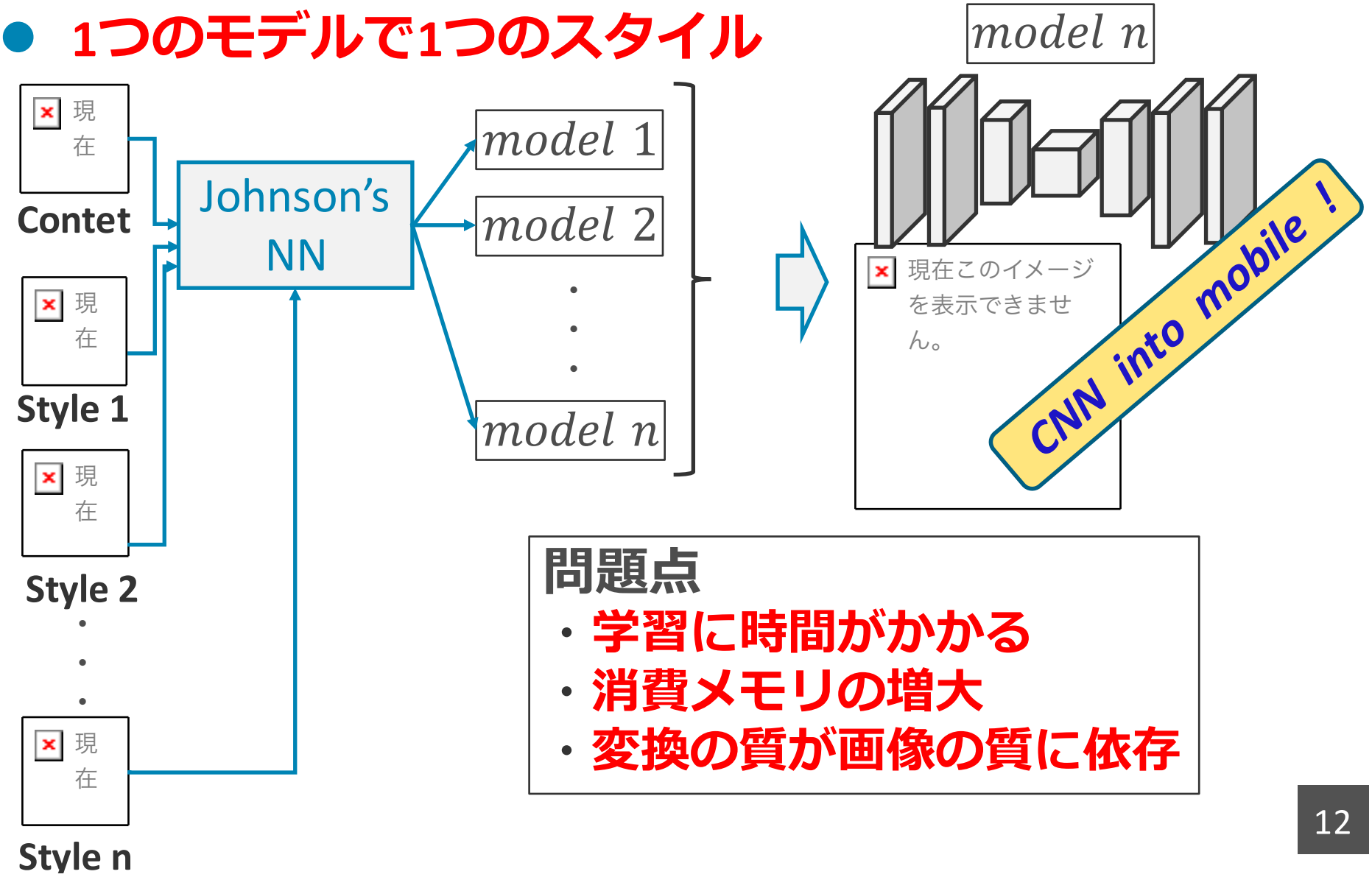
- **問題点**
  - **1つのモデルで1つのスタイル**

Johnsonらの手法の問題点

現在このイメージを

現在このイメージを

- 1つのモデルで1つのスタイル



問題点

- 学習に時間がかかる
- 消費メモリの増大
- 変換の質が画像の質に依存

Johnsonらの手法まとめ(再掲)

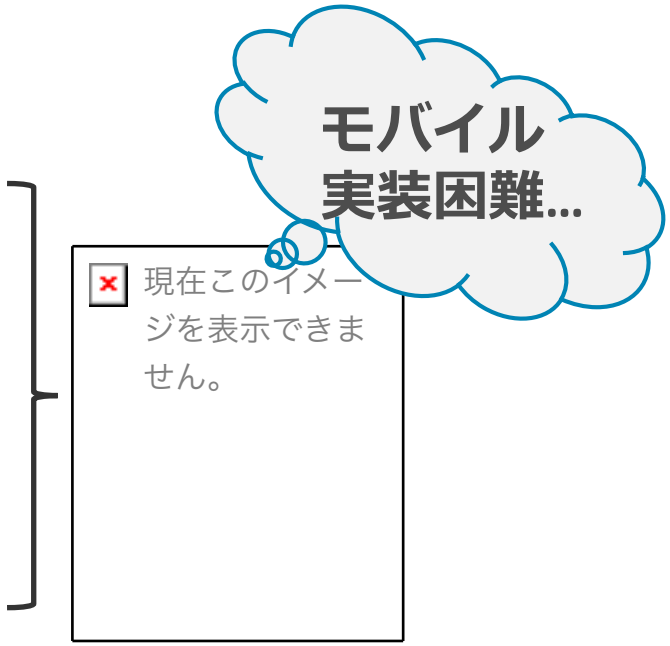
現在このイメージを

現在このイメージを

- 特定スタイルを変換可能なCNNを学習
- 順伝搬でスタイル変換可能
  - 非常に高速に画像生成可能に

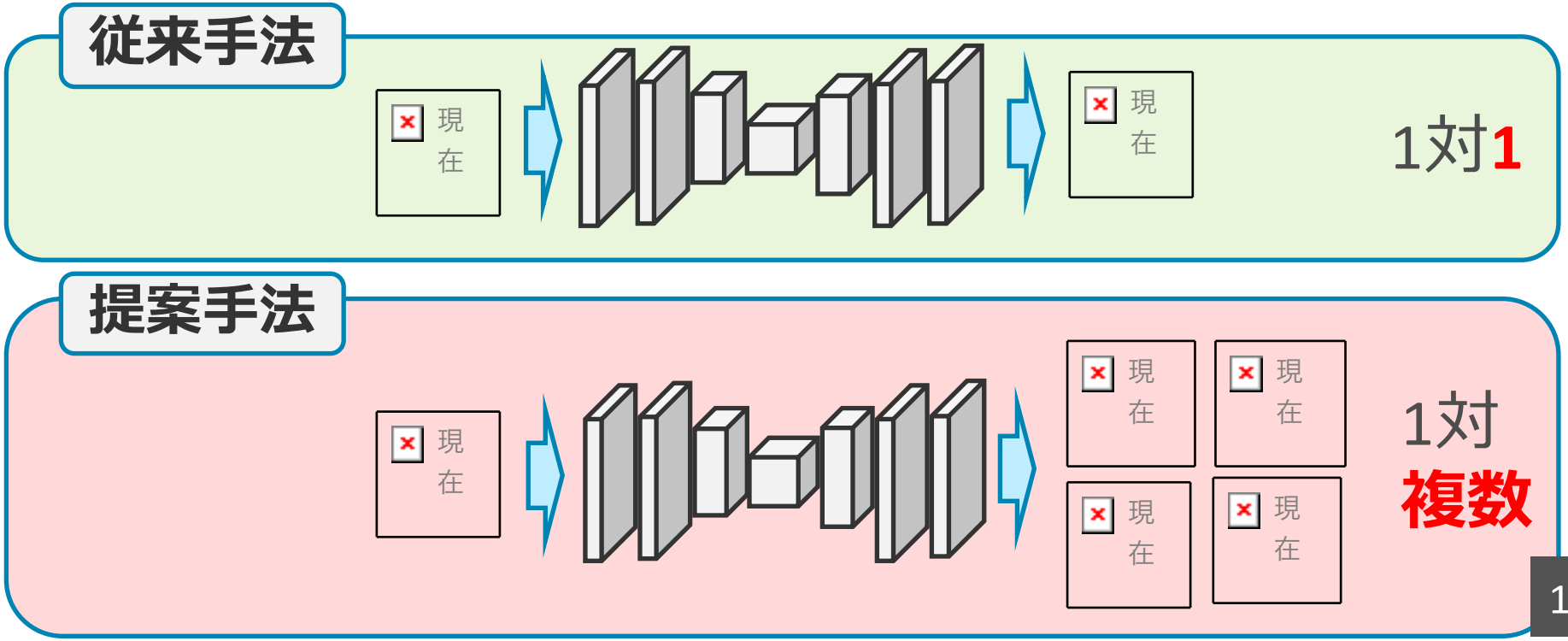
現在このイメージを表示できません。

- **問題点**
  - **1つのモデルで1つのスタイル**
  - **学習に時間を要する**
  - **消費メモリの増大**



目的

- Johnsonらの手法を拡張し，1つのモデルで**複数**のスタイルの同時学習を可能に
  - 学習時間の削減
  - 消費メモリの削減



本研究の貢献

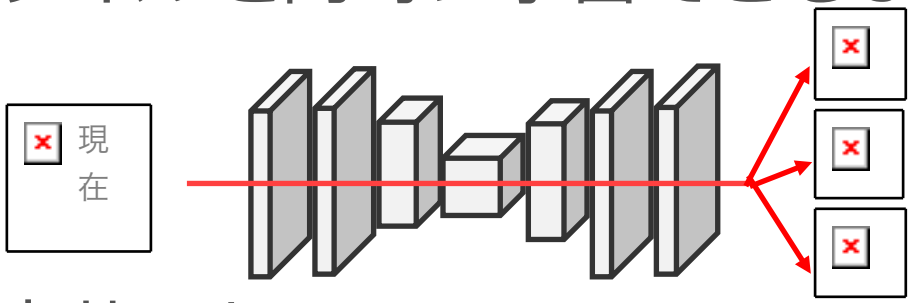
① 単一のモデルで複数のスタイルを任意の重みで合成

② リアルタイムで合成したスタイルを画像に転送

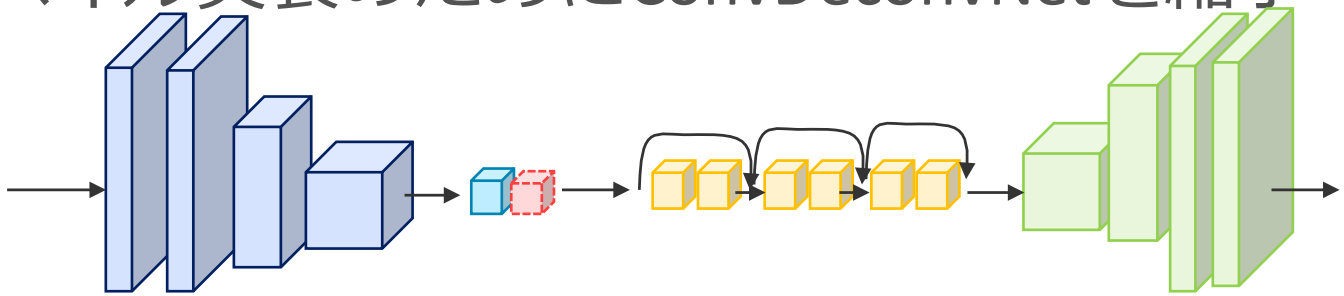
③ モバイルアプリとして実装

提案手法

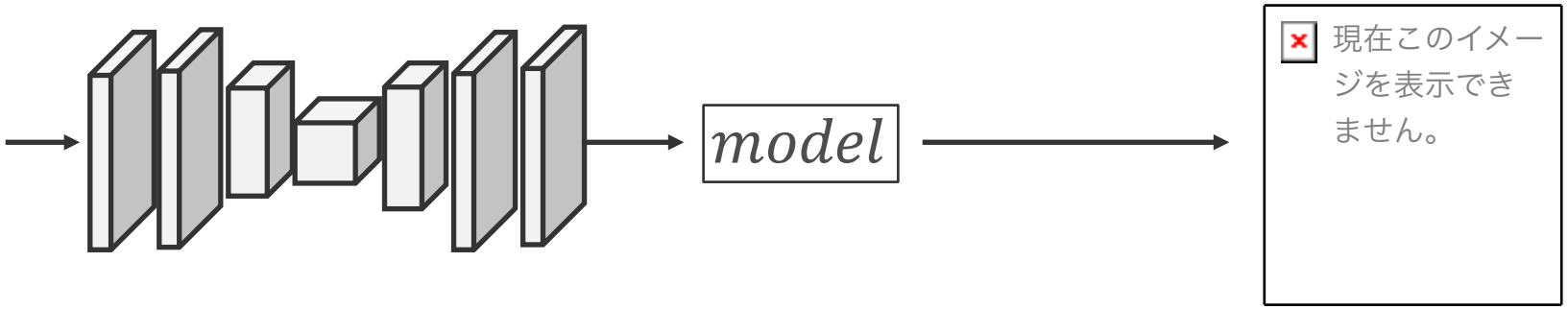
① 複数のスタイルを同時に学習できるように拡張



② モバイル実装のためにConvDeconvNetを縮小

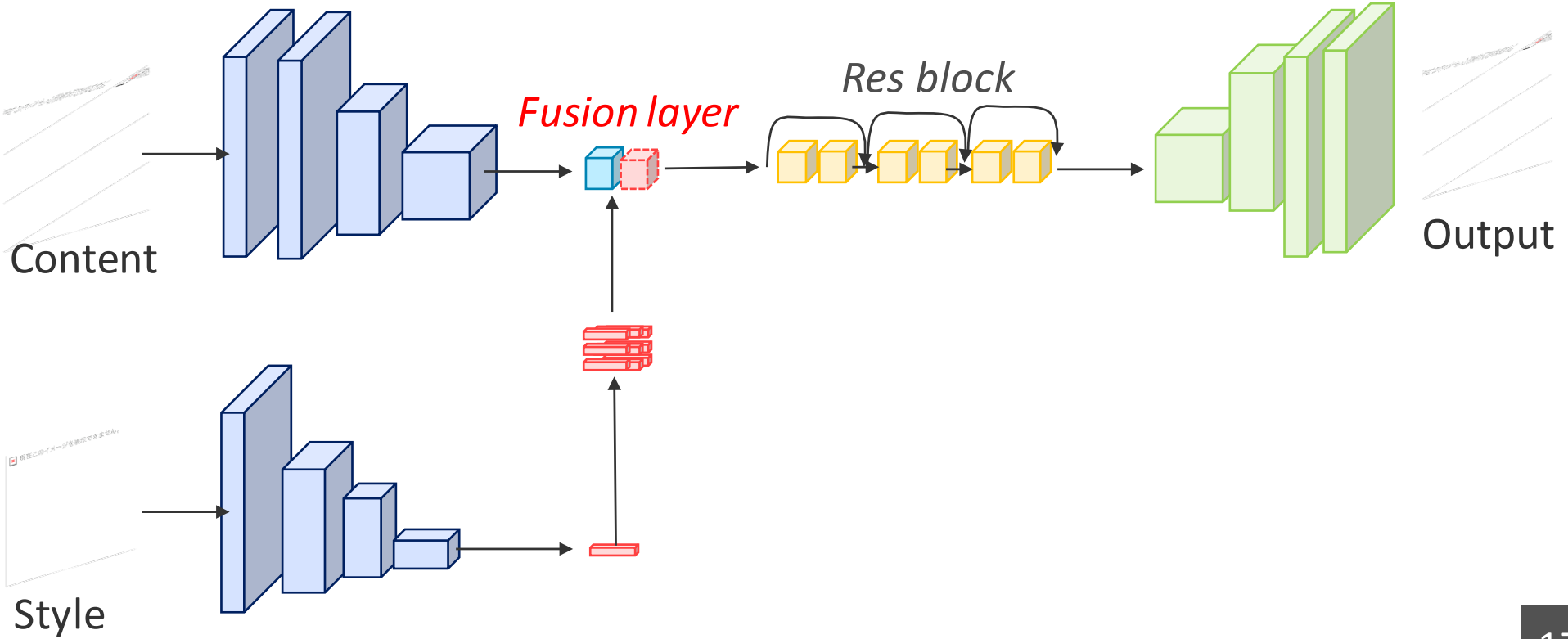


③ モバイル上での効率的なCNN実装及び工夫



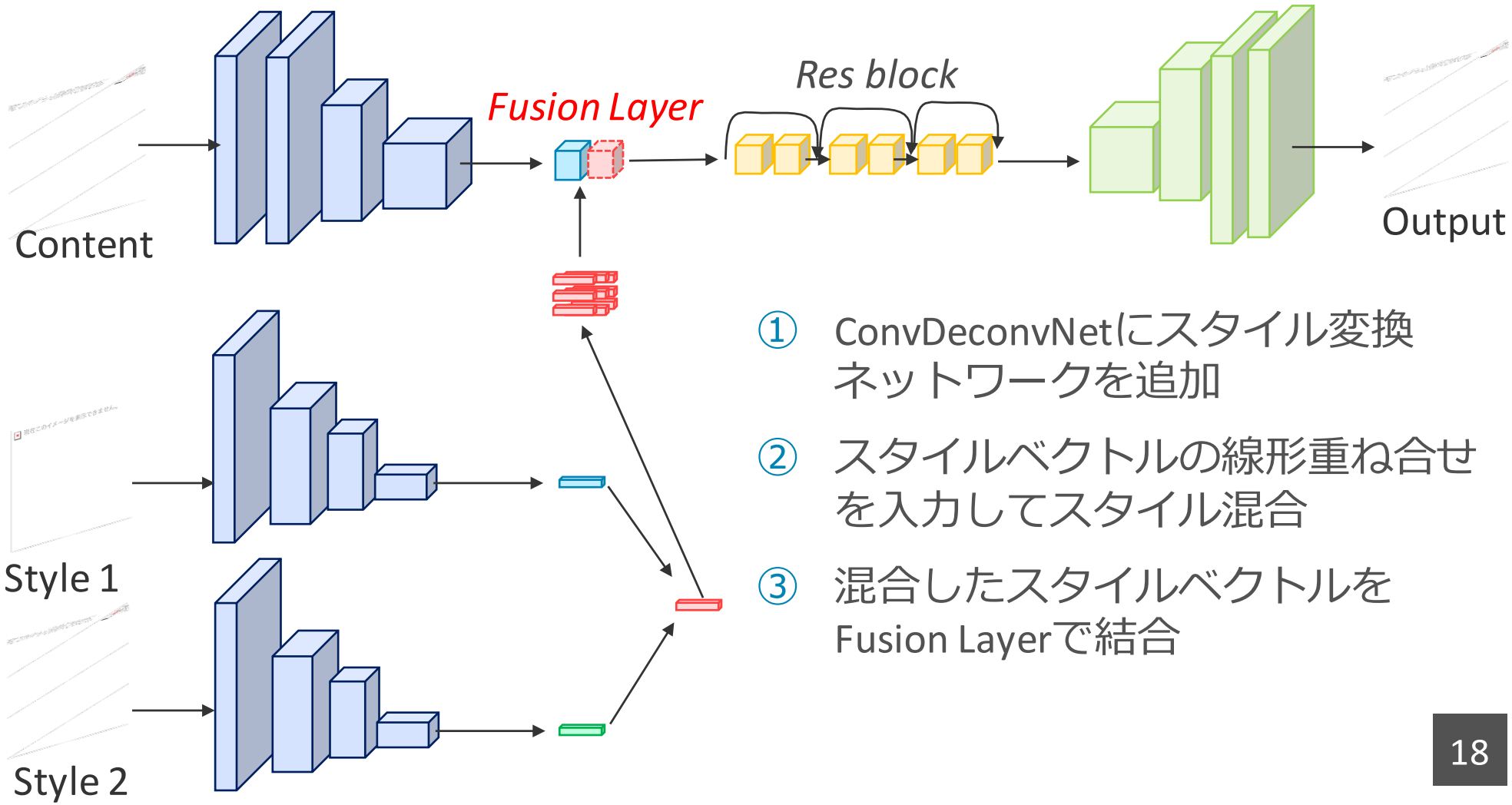
提案手法①

- **複数のスタイルを同時に学習できるように拡張**
  - Fusion Layerの追加
  - 複数の任意重みを合成可能



提案手法①

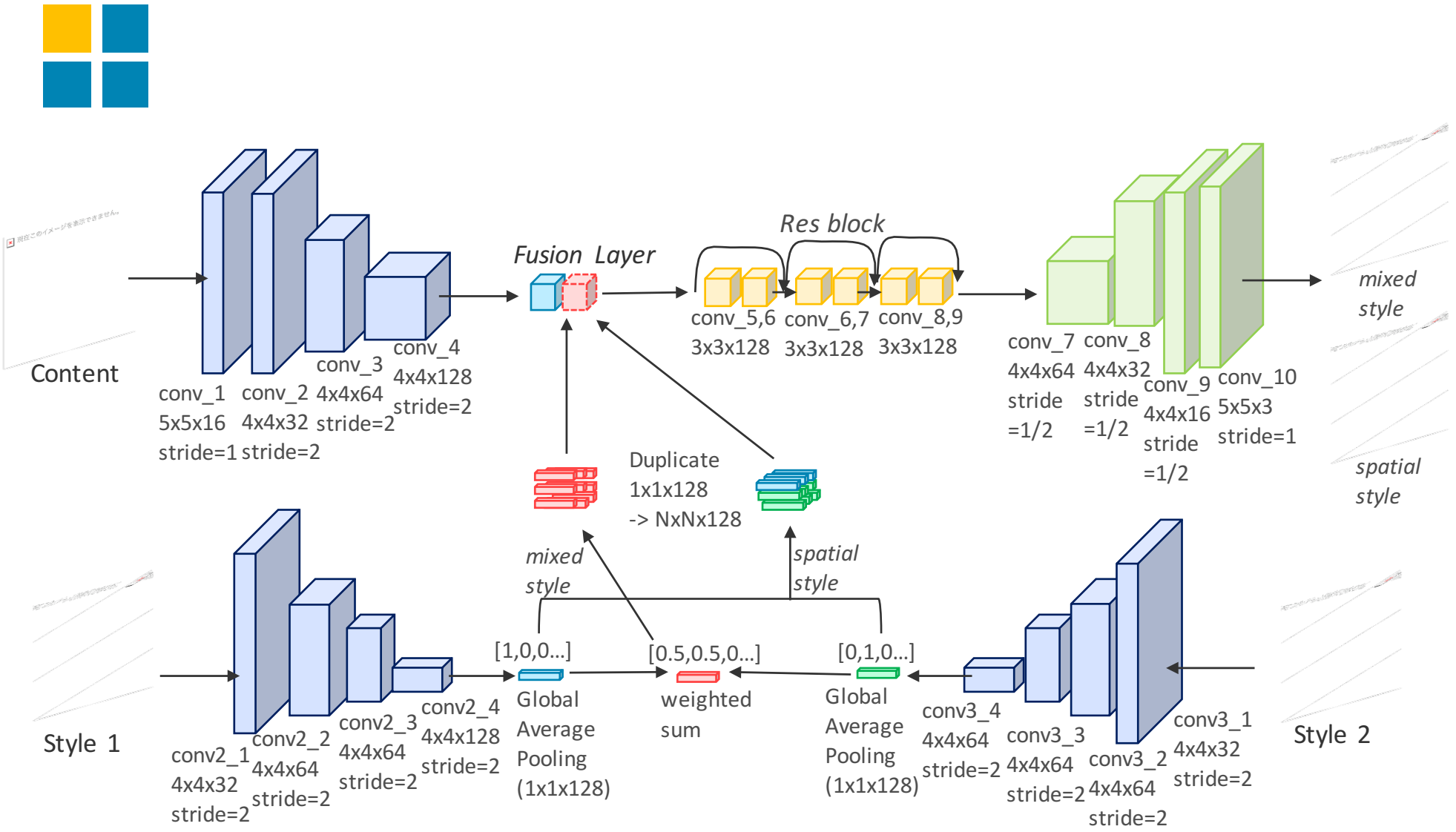
- 複数のスタイルを同時に学習できるように拡張



① ConvDeconvNetにスタイル変換ネットワークを追加

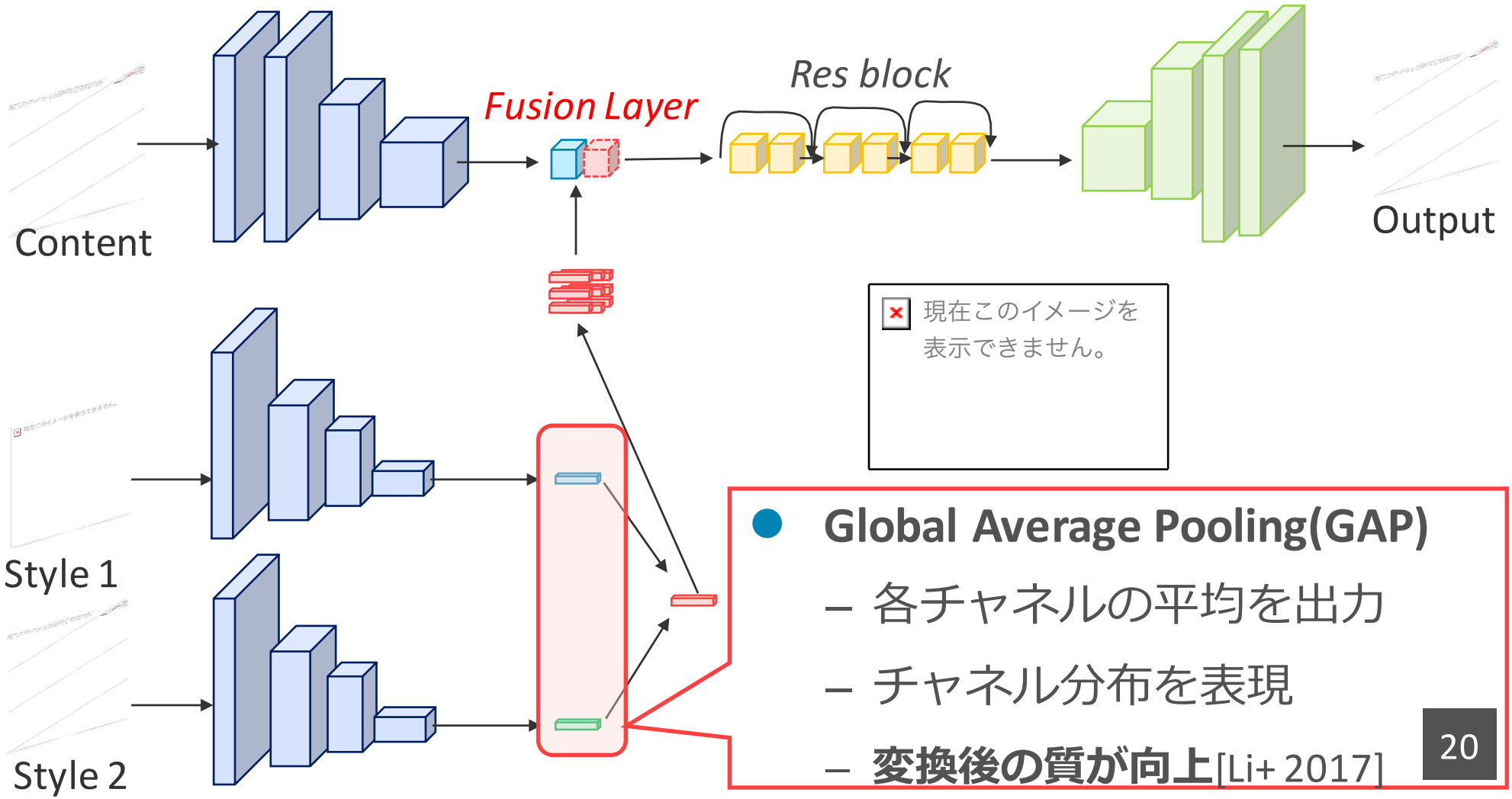
② スタイルベクトルの線形重ね合せを入力してスタイル混合

③ 混合したスタイルベクトルをFusion Layerで結合


Content
conv_1
5x5x16
stride=1
conv_2
4x4x32
stride=2
conv_3
4x4x64
stride=2
conv_4
4x4x128
stride=2
Fusion Layer
Res block
conv_5,6
3x3x128
conv_6,7
3x3x128
conv_8,9
3x3x128
conv_7
4x4x64
stride
=1/2
conv_8
4x4x32
stride
=1/2
conv_9
4x4x16
stride
=1/2
conv_10
5x5x3
stride=1
mixed
style
spatial
style
Duplicate
1x1x128
-> NxNx128
mixed
style
spatial
style
Style 1
conv2_1
4x4x32
stride=2
conv2_2
4x4x64
stride=2
conv2_3
4x4x64
stride=2
conv2_4
4x4x128
stride=2
[1,0,0...]
Global
Average
Pooling
(1x1x128)
[0.5,0.5,0...]
weighted
sum
[0,1,0...]
Global
Average
Pooling
(1x1x128)
conv3_4
4x4x64
stride=2
conv3_3
4x4x64
stride=2
conv3_2
4x4x64
stride=2
conv3_1
4x4x32
stride=2
Style 2

提案手法①

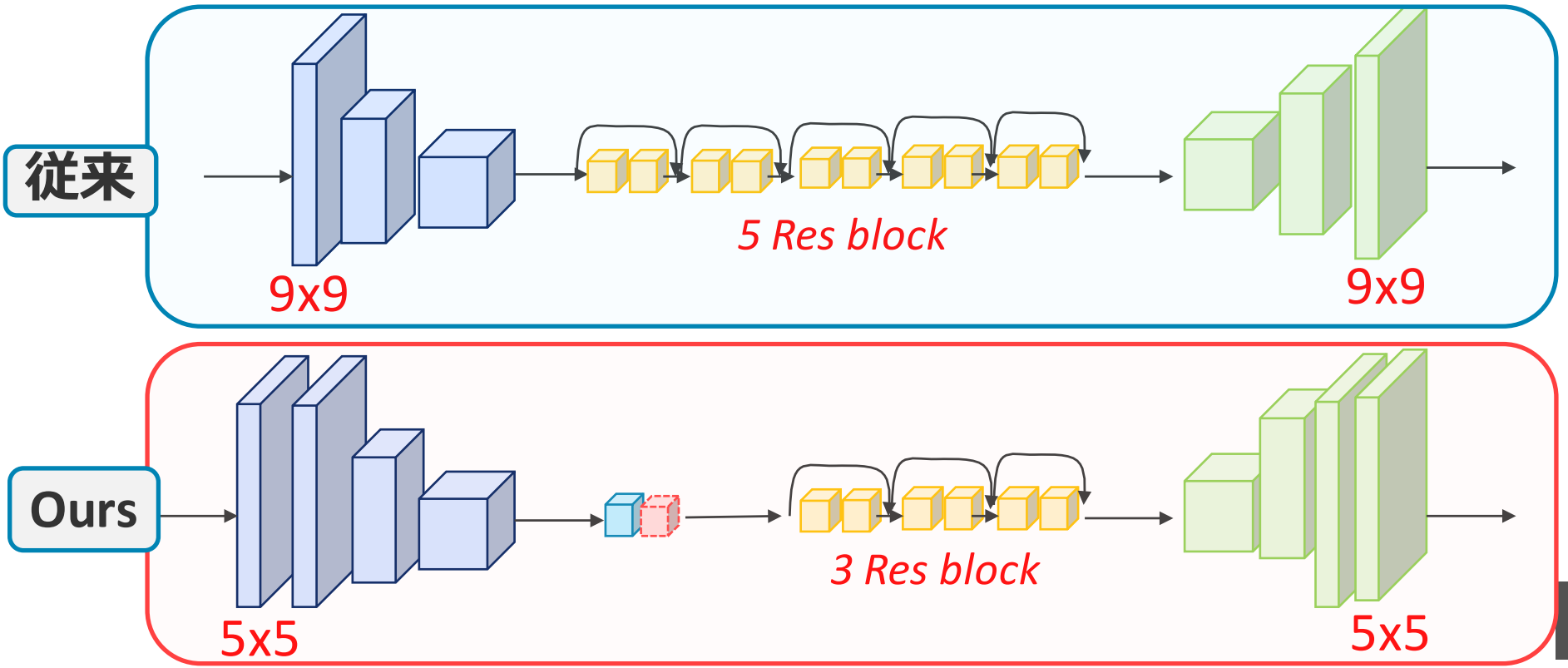
- 複数のスタイルを同時に学習できるように拡張



- **Global Average Pooling(GAP)**
  - 各チャネルの平均を出力
  - チャネル分布を表現
  - **変換後の質が向上**[Li+ 2017]

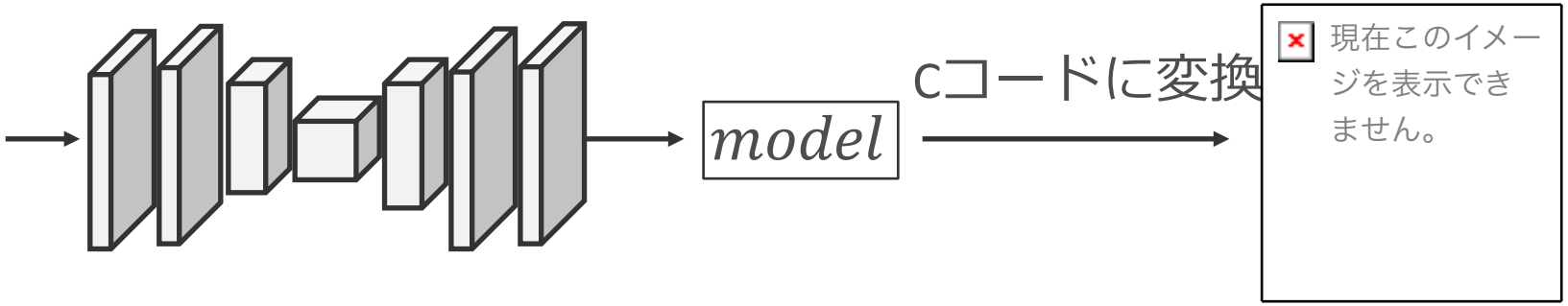
提案手法②

- モバイル実装のためにConvDeconvNetを縮小
  - 最初と最後のConv Layerのカーネルを 9x9 -> 5x5 に変更
  - Residual Elementsを 5 -> 3 に変更



提案手法③

- **モバイル上での効率的なCNN実装及び工夫**



① DNNを直接Cコードに変換(コンパイラ的に実行)

② Multithread化によるNEON / BLAS の効率的な利用

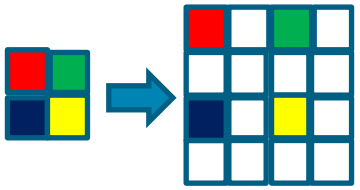
③ CNNに掛かる演算の可能な限りの事前計算の実行

④ **Unpooling+Conv -> 1/2 strideを実現**

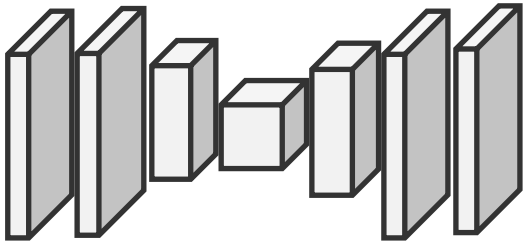
- ✓ **Deconvolutional layerの代替**

Unpooling + Conv -> 1/2 strideを実現

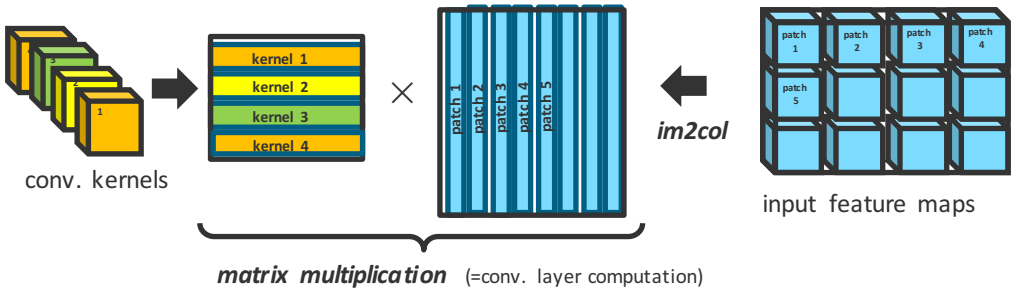
- Convの逆演算を行うDeconvをConvで表現したい…
  - **高速GEMM演算(BLAS)を利用したい**

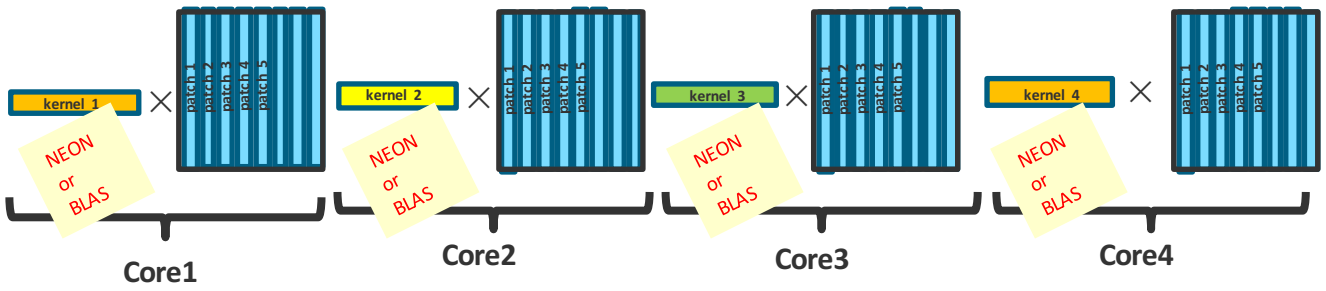


Unpooling =
1/2 stride Conv



- **Unpoolingで縦横2倍に拡大 -> Conv**



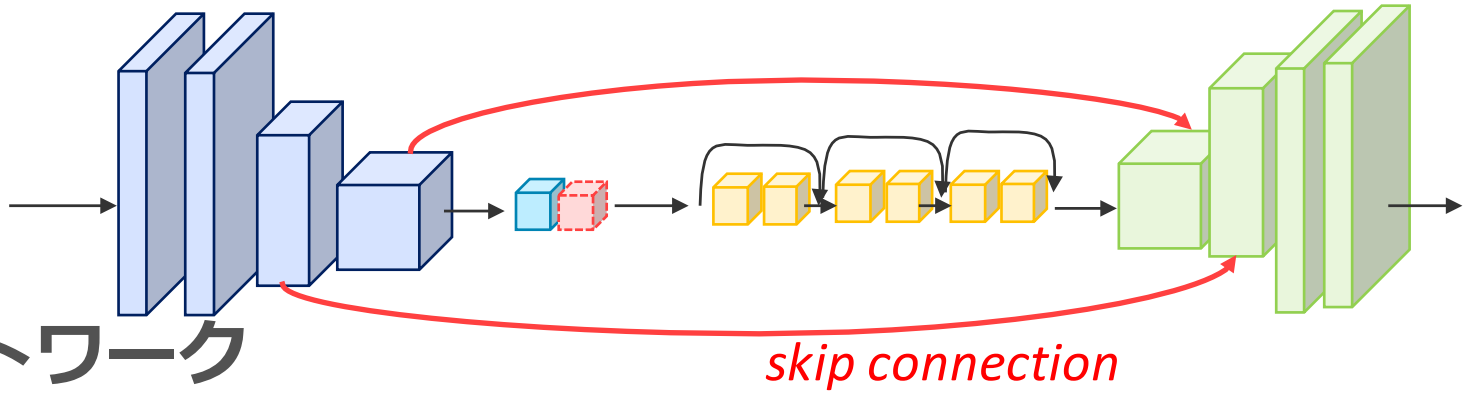


デモ動画

入力: 250x250
演算: 13億回
時間: 175ms(iphone 7+)

現在このイメージを表示できません。

考察



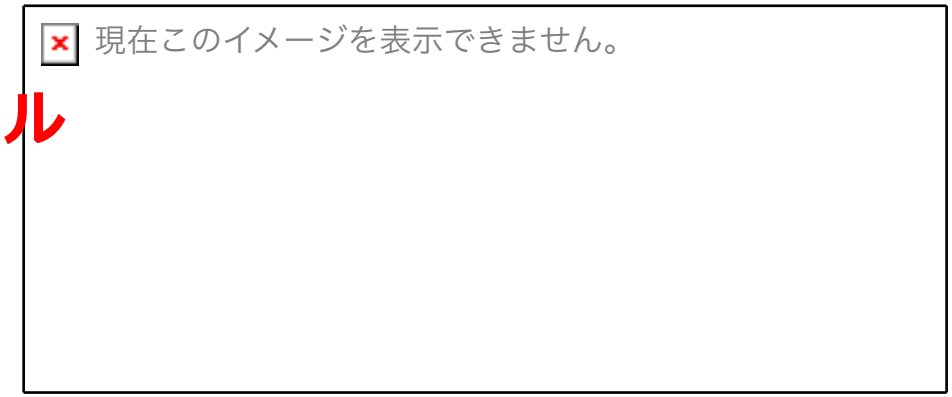
- **変換ネットワーク**
  - Res block 3個は過剰？
  - skip connectionの追加

- **モバイル実装**
  - 画像の解像度を上げたい -> Conv処理を高速化必須
  - 明示的にGPUを利用？

現在このイメージを表示できません。

まとめ

- 目的
  - 「Neural Style Transferをモバイル上に実装」
- 既存手法
  - **1つのモデルで1つのスタイル**
  - **学習に時間を要する**
  - **消費メモリの増大**
- 提案手法
  - **1つのモデルで複数のスタイルを変換可能**
  - **モバイル上でのリアルタイム画風変換**
- 今後の課題
  - **線画のモバイル上でのリアルタイム着色**



ご清聴ありがとうございました.

損失関数

スタイル画像
深いレイヤ・浅いレイヤ
どちらも利用