論文和訳

部分畳み込みを使用した不規則な穴の画像修復

概要。 既存の深層学習ベースの画像修復方法は、有効なピクセルとマスクされた穴の代替値（通常は平均値）の両方を条件とする畳み込みフィルター応答を使用して、破損した画像に対して標準の畳み込みネットワークを使用します。 これは、多くの場合、色の不一致やぼやけなどのアーティファクトにつながります。 通常、後処理はそのようなアーティファクトを減らすために使用されますが、高価で失敗する可能性があります。 部分畳み込みの使用を提案します。ここでは、畳み込みがマスクされ、正規化されて有効なピクセルのみに条件付けられます。 さらに、フォワードパスの一部として次のレイヤーの更新されたマスクを自動的に生成するメカニズムが含まれています。 このモデルは、不規則なマスクに対して他の方法を実行します。 アプローチを検証するために、他の方法との定性的および定量的な比較を示します。

キーワード：部分畳み込み、画像修復

1はじめに

（a）穴のある画像（b）パッチマッチ（c）飯塚ら[10] （d）Yu et al。[38]
（e）穴= 127.5（f）穴= IN平均（g）部分的な変換（h）オリジナル
一一図2.左から右、上から下：2（a）：穴のある画像。 2（b）：PatchMatch [2]の修復結果。 2（c）：飯塚らの修復結果[10]。 2（d）：Yu et al。[38]。 2（e）と2（f）はセクション3.2と同じネットワークアーキテクチャを使用していますが、一般的な畳み込みネットワークを使用しているため、2（e）はピクセル値127.5を使用して穴を初期化します。 2（f）は、平均ImageNetピクセル値を使用します。 2（g）：ホール値にとらわれない部分畳み込みベースの結果。

イメージの穴を埋めるタスクであるイメージの修復は、多くのアプリケーションで使用できます。 たとえば、結果のスペースをもっともらしい画像で埋めながら、画像編集で不要な画像コンテンツを削除するために使用できます。 従来の深層学習アプローチは、画像の中心付近に位置する長方形の領域に焦点を当てており、多くの場合、高価な後処理に依存しています。 この作業の目的は、不規則な穴パターンで堅牢に動作し、追加の後処理を必要とせずに画像の残りの部分にスムーズに組み込む意味的に意味のある予測を生成する画像修復のモデルを提案することです またはブレンド操作。
深層学習を使用しない最近の画像修復アプローチでは、残りの画像の画像統計を使用して穴を埋めます。 最先端の方法の1つであるPatchMatch [2]は、穴を埋めるために最適なパッチを繰り返し検索します。 このアプローチは一般にスムーズな結果を生成しますが、利用可能な画像統計によって制限され、視覚的なセマンティクスの概念はありません。 たとえば、図2（b）では、PatchMatchは周囲の影や壁からの画像パッチを使用して、ペイントの欠落したコンポーネントをスムーズに埋めることができましたが、セマンティックに対応したアプローチでは、代わりにペイントからのパッチを使用します。
ディープニューラルネットワークは、セマンティックプライオリティと意味のある隠された表現をエンドツーエンドで学習します。これらは、最近の画像修復作業に使用されています。 これらのネットワークは、画像に畳み込みフィルターを採用し、削除されたコンテンツを固定値に置き換えます。 その結果、これらのアプローチは初期の穴の値に依存する

ため、穴領域のテクスチャの欠如、明らかな色のコントラスト、または穴を囲む人工的なエッジ応答として現れることがよくあります。 図2（e）および2（f）に、さまざまなホール値の初期化を伴う典型的な畳み込み層でU-Netアーキテクチャを使用した例を示します。 （両方について、トレーニングとテストは同じ初期化スキームを共有します）。
穴の値で出力を調整すると、最終的には高価な後処理を必要とするさまざまな種類の視覚的アーチファクトが生じます。 例えば、飯塚ら。 [10]は高速マーチング[32]およびポアソン画像ブレンディング[23]を使用しますが、Yu et al。 [38]フォローアップ改良ネットワークを使用して、生のネットワーク予測を改良します。 ただし、これらの改良では、2（c）および2（d）として示されるすべてのアーティファクトを解決することはできません。 私たちの仕事の目的は、穴の初期化値とは無関係に、追加の後処理なしで、十分に組み込まれた穴の予測を達成することです。
多くの最近のアプローチの別の制限は、しばしば画像の中心であると想定される長方形の穴に焦点が当てられていることです。 これらの制限は、長方形の穴への過剰適合につながる可能性があり、最終的にこれらのモデルのアプリケーションでの有用性を制限することがあります。 Pathak et al。 [22] and Yang et al。 [36] 128×128画像の中心に64×64の正方形の穴があると仮定します。 飯塚ら [10] and Yu et al。 [38]中心の穴の仮定を削除し、不規則な形状の穴を処理できますが、不規則なマスクを含む多数の画像に対して広範な定量分析を実行しません（[8]の51のテスト画像）。 より実用的な不規則な穴のユースケースに焦点を当てるために、さまざまなサイズの不規則なマスクを使用して画像の大きなベンチマークを収集します。 分析では、穴のサイズだけでなく、穴が画像の境界に接触しているかどうかの影響も調べます。
不規則なマスクを適切に処理するために、マスクされて再正規化された畳み込み操作と、それに続くマスク更新ステップで構成される部分畳み込み層の使用を提案します。 マスクされ再正規化された畳み込みの概念は、画像セグメンテーションタスクの[7]でセグメンテーション対応畳み込みとも呼ばれますが、入力マスクを変更しませんでした。 部分畳み込みの使用では、バイナリマスクが与えられると、畳み込み結果はすべての層の非ホール領域のみに依存します。 主な拡張機能は、マスクの自動更新ステップです。このステップでは、部分畳み込みがマスクされていない値で処理できたマスキングを削除します。 連続した更新の十分なレイヤーが与えられると、最大のマスクされたホールでさえ最終的に縮小し、機能マップに有効な応答のみを残します。 部分的な畳み込み層により、最終的にモデルはプレースホルダーの穴の値に依存しなくなります。

要約すると、次の貢献を行います。
—最新の画像修復を実現するために、自動マスク更新ステップを使用した部分畳み込みの使用を提案します。
—これまでの研究では、典型的な畳み込みのあるU-Net [34]のスキップリンクで良好な修復結果を得ることができませんでしたが、部分畳み込みとマスク更新で畳み込み層を置き換えると、最先端の修復結果が得られることを示しています。
—私たちの知る限りでは、不規則な形状の穴で画像修復モデルをトレーニングすることの有効性を初めて実証しました。
—大規模な不規則なマスクデータセットを提案します。これは、ペイントモデルのトレーニングと評価における将来の取り組みを促進するために公開されます。

2.関連研究

３、アプローチ
提案モデルでは、スタックされた部分畳み込み演算とマスク更新ステップを使用して、画像修復を実行します。 最初に畳み込みおよびマスク更新メカニズムを定義してから、モデルアーキテクチャと損失関数について説明します。

3.1部分畳み込み層
部分畳み込み演算とマスク更新機能を合わせて、部分畳み込み層と呼びます。 Wを畳み込みフィルターのための畳み込みフィルターの重み、bを対応するバイアスとします。 Xは現在の畳み込み（スライド）ウィンドウの特徴値（ピクセル値）で、Mは対応するバイナリマスクです。 [7]で同様に定義されている、すべての位置での部分畳み込みは、次のように表されます。（式１）

ここで、⊙は要素ごとの乗算を示し、1はMと同じ形状ですが、すべての要素は1です。出力値は、マスクされていない入力のみに依存します。 スケーリング係数sum（1）/sum（M）は、適切なスケーリングを適用して、さまざまな量の有効な（マスクされていない）入力を調整します。
各部分畳み込み演算の後、マスクを次のように更新します。畳み込みが少なくとも1つの有効な入力値で出力を調整できた場合、その場所を有効としてマークします。 これは次のように表現されます。（式２）

フォワードパスの一部として、ディープラーニングフレームワークに簡単に実装できます。 部分畳み込み層を十分に連続して適用すると、入力に有効なピクセルが含まれていた場合、最終的にマスクはすべて1になります。

3.2ネットワークのアーキテクチャと実装
実装。 部分畳み込み層は、既存の標準PyTorch [21]を拡張することで実装されますが、カスタム層を使用して時間と空間の両方で改善できます。 簡単な実装では、サイズC×H×Wのバイナリマスクを定義し、関連する画像/機能と同じサイズにし、次に、部分更新と同じカーネルサイズの固定畳み込み層を使用してマスク更新を実装します。 たたみ込み演算、ただし重みは1に等しく設定され、バイアスはありません。 512×512画像のネットワーク全体の推定には、穴のサイズに関係なく、単一のNVIDIA V100 GPUで0.029秒かかります。

ネットワーク設計。 [11]で使用されるものと同様のUNetのようなアーキテクチャ[25]を設計し、すべての畳み込み層を部分畳み込み層に置き換え、デコード段階で最近傍のアップサンプリングを使用します。 スキップリンクは、2つの機能マップと2つのマスクをそれぞれ連結し、次の部分畳み込みレイヤーの機能およびマスク入力として機能します。 最後の部分畳み込み層の入力には、元の入力画像と穴と元のマスクの連結が含まれ、モデルが穴以外のピクセルをコピーできるようにします。 ネットワークの詳細は、補足ファイルに記載されています。

パディングとしての部分畳み込み。 典型的なパディングの代わりに画像境界で適切なマスキングを持つ部分畳み込みを使用します。 これにより、画像の境界で修復されたコンテンツが、画像外の無効な値の影響を受けなくなります。これは、別の穴として解釈される可能性があります。

3.3損失関数
損失関数は、ピクセルごとの再構成精度と組成、つまり予測された穴の値が周囲のコンテキストにどれだけスムーズに移行できるかを目的としています。
穴Iinのある入力画像、初期バイナリマスクM（穴の場合は0）、ネットワーク予測Iout、オリジナル画像Igtを与え、ピクセル損失Lholeを定義する（論文に式）
ここで、NIgtはIgtの要素数を表す。（NIgt = C ∗ H ∗ WおよびC、HおよびWは、画像Igtのチャネルサイズ、高さ、および幅です）。 これらは、それぞれ穴および非穴ピクセルのネットワーク出力でのL1損失です。

次に、Gatys et al。によって導入された知覚損失を定義します。（式３）
ここで、Icompは未編集の出力画像Ioutですが、穴出ないピクセル（穴でない部分）も一緒です。
知覚損失は、IoutとIcompの両方とオリジナルの間のL1距離を計算します知覚損失は、IoutとIcompの両方と地面の間のL1距離を計算しますが、後でこれらの画像を使用し、ImageNet事前トレーニング済みVGG-16 [29]w用いてより高いレベルの特徴空間に投影する。
さらに、知覚的損失[6]に似たスタイル損失項を含めますが、Lを適用する前に各特徴マップで最初に自己相関（グラム行列）を実行します。
(式４）（式５）
ここで、行列演算では、高レベルの特徴Ψ（x）pが形状（HpWp）×Cpであり、Cp×Cp Gram行列になると仮定し、Kpは選択された層であるpthの正規化係数1 / CpHpWpであることに注意してください 。 繰り返しますが、生の出力と合成された出力の両方に損失項を含めます。
最終的な損失項は、総変動（TV）損失Ltvです。これは、Rの平滑化ペナルティ[12]です。
ここで、Rは、ホール領域の1ピクセル膨張の領域です。
（式６）
NIcomp総損失Ltotalは、上記のすべての損失関数の組み合わせです。（式７）
損失項の重みは、100枚の検証画像に対してハイパーパラメーター検索を実行することにより決定されました。

さまざまな損失条件のアブレーション（切除）研究。
知覚的損失[12]は、チェッカーボードのアーティファクトを生成することが知られています。 ジョンソン他 [12]は、全変動（TV）損失を使用して問題を改善することを提案しています。 これは、このモデルには当てはまらないことがわかりました。 図3（b）は、LtotalからLstyleoutとLstylecompを削除してトレーニングしたモデルの結果を示しています。 このモデルでは、追加のスタイル損失条件が必要です。 ただし、スタイル損失のすべての損失重み付けスキームが妥当な結果を生成するわけではありません。 図3（f）は、小さなスタイルの損失ウェイトでトレーニングされたモデルの結果を示しています。 図3（g）の完全なLtotalでトレーニングされたモデルの結果と比較すると、多くの魚鱗のアーティファクトがあります。 ただし、知覚的損失も重要です。 グリッド状のアーティファクトは、知覚的損失のない結果（図3（j））よりも完全なLtotalの結果（図3（k））では目立ちません。 この議論が、VGGベースの高レベル損失の採用に関心のある読者に役立つことを願っています。

4実験
4.1不規則なマスクデータセット
以前の研究では、画像内の長方形の領域をランダムに削除して、データセットに穴を生成していました。 必要な多様な穴の形状とサイズを作成するには、これでは不十分だと考えています。 そのため、ランダムな線のマスクと任意の形状の穴を収集することから始めます。 [31]で説明されているビデオの2つの連続したフレーム間のオクルージョン/ディスオクルージョン（覆う・塞ぐ）マスク推定方法の結果が、このようなパターンの良いソースであることがわかりました。 トレーニング用に55,116個のマスク、テスト用に24,866個のマスクを生成します。 トレーニング中に、55,116個のマスクからランダムにマスクをサンプリングしてマスクデータセットを増強し、後でランダムな膨張、回転、およびトリミングを実行します。 トレーニングとテスト用のすべてのマスクと画像は、512 x 512のサイズです。
24,866の未加工マスクから開始し、ランダムな膨張、回転、およびトリミングを追加し

て、テストセットを作成します。 [10]などの以前の多くの方法では、画像の境界付近の穴でパフォーマンスが低下していました。 そのため、テストセットを2つに分割します。境界付近に穴があるマスクと穴がないマスクです。 境界線から離れた穴のある分割により、境界線から少なくとも50ピクセルの距離が確保されます。
また、マスクを穴のサイズでさらに分類します。 具体的には、穴と画像の面積比が異なる6つのカテゴリのマスクを生成します：（0.01、0.1]、（0.1、0.2]、（0.2、0.3]、（0.3、0.4]、（0.4、0.5]、（0.5 、0.6]。各カテゴリには、境界制約の有無にかかわらず1000個のマスクが含まれます。合計で6×2×1000 = 12、000個のマスクを作成しました。各カテゴリのマスクの例を図4に示します。

4.2トレーニングプロセス

トレーニングデータ
トレーニングとテストには、ImageNetデータセット[26]、Places2データセット[39]、CelebA-HQ [19,13]の3つの別個の画像データセットを使用します。 ImageNetとPlaces2には、元のtrain、test、val分割を使用します。 CelebA-HQの場合、トレーニング用の27Kイメージとテスト用の3Kイメージにランダムに分割します。

トレーニング手順。
[9]で説明されている初期化メソッドを使用して重みを初期化し、最適化のためにAdam [14]（optimaizer：最適化アルゴリズム）を使用します。 バッチサイズ6の単一のNVIDIA V100 GPU（16GB）でトレーニングします。

初期トレーニングと微調整。
穴は、穴のピクセルの平均と分散が計算されるため、バッチ正規化の問題を示します。したがって、マスクされた場所でそれらを無視するのが理にかなっています。 ただし、各アプリケーションで次第に穴が埋められ、通常はデコーダーステージによって完全に削除されます。
穴がある場合にバッチ正規化を使用するには、最初に0.0002の〜学習率〜を使用して初期トレーニングのバッチ正規化をオンにします。 次に、学習率0.00005を使用して微調整し、ネットワークのエンコーダー部分のバッチ正規化パラメーターをフリーズします。 デコーダでバッチ正規化を有効にしておきます。 これは、誤った平均と分散の問題を回避するだけでなく、より高速な収束を達成するのにも役立ちます。 ImageNetおよびPlaces2モデルは10日間トレーニングしますが、CelebA-HQは3日間トレーニングします。 すべての微調整は1日で実行されます。

4.3比較
4つの方法と比較します。
PatchMatch、飯塚、Yu、昆布（私たちの方法と同じネットワーク構造ですが、典型的な畳み込み層を使用しています。 損失重量は、ハイパーパラメーター検索を介して再決定されました。）

このメソッドはPConvと表記されます。 GLおよびGntIptとの公正な比較には、データに対するモデルの再トレーニングが必要です。 ただし、両方のアプローチのトレーニングでは、穴のローカルバウンディングボックス（穴を覆う最小限のボックス）が利用可能であると想定して、ローカル識別器を使用します。これは、マスクの形状には意味がありません。
そのため、リリース済みの事前トレーニングモデルを直接使用します。 PatchMatchに

は、サードパーティ（発表主でない）の実装を使用しました。 彼らのtrain-testの分割がわからないので、私たち自身の分割はおそらくそれらとは異なるでしょう。 12,000枚の画像を評価して、置換せずに画像にマスクをランダムに割り当てます。

定性的比較。（質）
図5と図6は、それぞれImageNetとPlaces2の比較を示しています。 GTはオリジナルを表します。 図8のCelebA-HQのGntIpt [38]と比較します。GntIptは256×256でCelebA-HQをテストしたため、モデルに入力する前に画像を256×256にダウンサンプリングします。 PMがセマンティック（意味のある）に不適切なパッチをコピーして穴を埋めることがありますが、GLおよびGntIptは、後処理または改良ネットワークを介して妥当な結果を達成できない場合があります。 図7は、Convの結果を示しています。Convには、ホールプレースホルダー（穴の仮初の値）値の条件付けからの明確なアーティファクトが含まれています。

質の比較
[38]で述べたように、多くの解決策が存在するため、画像修復の結果を評価するための適切な数値メトリックはありません。 それでも、以前の画像修復作業[36,38]に従って、l1エラー、PSNR、SSIM [35]、および開始スコア[27]を報告します。 l1エラー、PSNRおよびSSIMはPlaces2で報告されますが、Inceptionスコア（IS-core）はImageNetで報告されます。 [10]のリリースされたモデルは、すべての評価に使用するPlaces2でのみトレーニングされていることに注意してください。 表1に比較結果を示します。 この方法は、不規則なマスクでのこれらの測定において、他のすべての方法よりも優れていることがわかります。

ユーザー調査
定量的な比較に加えて、人間の主観的な調査を通じてアルゴリズムを評価します。
Amazon Mechanical Turk（MTurk）プラットフォームに展開された、穴の位置や穴のある元の入力画像を表示せずに、ペアワイズA / Bテストを実行します。 無制限の時間と制限時間の2種類の実験を実行します。 また、画像の境界付近に穴がある場合とない場合を別々に報告します。 状況ごとに、各メソッドに対して300個の画像をランダムに選択します。各画像は10回比較されます。
無制限の時間設定の場合、ワーカーには一度に2つの画像が与えられます。それぞれ異なる方法で生成されます。 作業者には、よりリアルに見える画像を選択するための無制限の時間が与えられます。 また、公平な比較を保証するために画像の順序をシャッフルします。 すべての異なる穴と画像の面積比にわたる結果を図9（a）にまとめています。 最初の行は、穴が画像の境界から少なくとも50ピクセル離れている結果を示し、2番目の行は、穴が画像の境界に近いか、画像の境界に触れる場合を示しています。 ご覧のように、どちらの場合も、このメソッドは他のすべてのメソッドよりもはるかに優れたパフォーマンスを発揮します（50％は2つのメソッドのパフォーマンスが同等であることを意味します）。
限られた時間の設定では、すべての方法（当社の方法を含む）をグラウンドトゥルースと比較します。 各比較では、1つの方法の結果が選択され、限られた時間、グラウンドトゥルースとともに作業者に表示されます。 次に、作業者は、より自然に見える画像を選択するように求められます。 これにより、画像間の違いがどれだけ早く認識されるかが評価されます。 異なる時間間隔の比較結果を図9（b）に示します。 繰り返しますが、最初の行は穴が画像の境界に触れない場合を示していますが、2番目の行はそれを許可しています。 私たちの方法は、ほとんどの場合、さまざまな項と穴と画像の面積比で他の方法よりも優れています。

5議論と延長

5.1議論

自動マスク更新メカニズムを備えた部分畳み込み層の使用を提案し、最新の画像修復結果を達成します。 このモデルは、あらゆる形状、サイズの場所、または画像の境界からの距離の穴を堅牢に処理できます。 さらに、図10に示すように、穴のサイズが大きくなってもパフォーマンスが壊滅的に低下することはありません。ただし、この方法の1つの制限は、図11のドアのバーなどのまばらな構造の画像では失敗することです。 ほとんどの方法は、最大の穴で苦労しています。

5.2画像の超解像度への拡張

また、ピクセルをオフセットし、穴を挿入することにより、フレームワークを超解像タスクに拡張します。 具体的には、高さHと幅W、アップスケーリング係数Kの低解像度画像Iが与えられた場合、以下を使用してネットワークの高さK * Hと幅K * Wの入力I 'を構築します：各ピクセル（x、 y）Iでは、I 'の（K * x + [K/2]、K * y +]K/2]）に配置し、この位置にマスク値が1になるようにマークします。 図12でK = 4を見つけることができます。 図13で、K = 4の2つの既知の画像超解像アプローチSRGAN [16]およびMDSR + [18]と比較します。