Art'Em – 絵画風加工を VR で実現する: パート 1

この記事は、インテル® デベロッパー・ゾーンに公開されている「Art'Em – Artistic Style Transfer to Virtual Reality Week 2 Updat」の日本語参考訳です。

Art'Em (英語) は、バーチャル・リアリティー (VR) で絵画風加工を利用できるようにするアプリケーションです。低精度ネットワークを利用することで、スタイライゼーション (様式化) を高速化することを目的としています。私は、この早期イノベーション・プロジェクトで、低精度ネットワークにより乗算を加算とビット単位の否定排他的論理和 (XNOR) に置き換えたいと考えています。

$$\begin{bmatrix} -1 & +1 & +1 \\ -1 & -1 & -1 \\ +1 & +1 & -1 \end{bmatrix} \times \begin{bmatrix} -1 \\ +1 \\ +1 \end{bmatrix} = \begin{matrix} (-1*-1)+(1*1)+(1*1) \\ (-1*-1)+(1*-1)+(1*-1) \\ (-1*1)+(1*1)+(1*-1) \end{matrix} = \begin{bmatrix} 3 \\ -1 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & +1 & +1 \\ 0 & 0 & 0 \\ +1 & +1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{matrix} pcnt(xnor(011,011)) \\ pcnt(xnor(011,000)) \\ pcnt(xnor(011,110)) \end{matrix} = \begin{bmatrix} 3 \\ -1 \\ -1 \end{bmatrix}$$

上記のイメージから、この行列乗算をビット単位の XNOR popcount 操作に置き換えられることは明らかです。pcnt は popcount 操作を表します。

概念を説明する前に、まずこのアプリケーションの目的を理解しましょう。

絵画風加工とは?

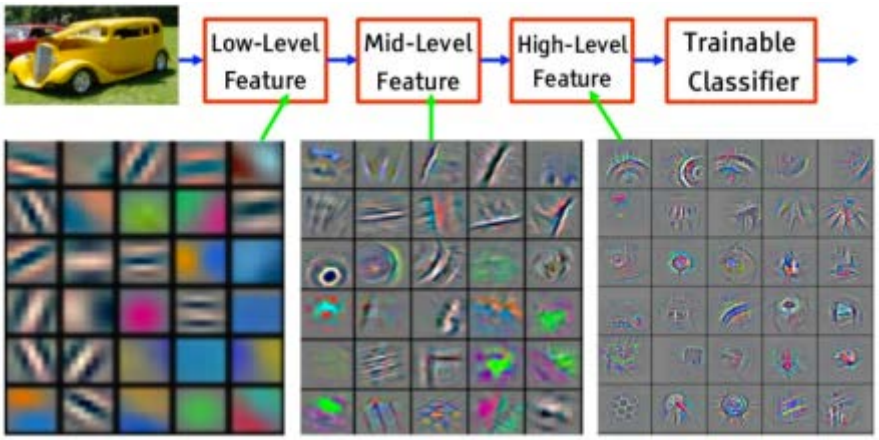
Content　Style　Pastiche

(https://goo.gl/6T67VY (英語))

すでに、皆さんは Prisma* アプリケーションや YouTube* (英語) で目にしているかもしれません。絵画風加工 (アーティスティックフィルターなどとも呼ばれます) とは、基本的に入力イメージを別のリファレンス・イメージの「スタイル」に変換することです。上記の例では、content (コンテンツ) が入力イメージで、style (スタイル) がリファレンス・イメージです。

アイデアを話すことは簡単ですが、イメージの「スタイル」と「コンテンツ」を定義し、イメージから選択したスタイルを抽出するには、どうしたら良いのでしょうか? この質問に答えるには、畳み込みネットワークについて理解する必要があります。

畳み込みニューラル・ネットワーク (CNN) は、動物の映像処理を応用した畳み込み操作を利用する、ディープ・ニューラル・ネットワークの一種です。CNN についてはここで詳しく述べませんが、これらのネットワークには特定のイメージの特徴に反応するニューロンがあります。これらの特徴は、次のイメージから想像することができるでしょう。



(http://www.iro.umontreal.ca/~bengioy/talks/DL-Tutorial-NIPS2015.pdf (英語))

イメージの各セルには、最も反応の高かった特定の特徴が入っています。ネットワークが深くなるにつれ、分類器の特徴は抽象的になります。

これは、絵画風加工でどのような処理が行われているか理解する上で重要です。ネットワークが深くなるにつれ、各層はイメージのより抽象化された特徴に反応することが分かります。そこで必要になるのが、スタイルイメージから抽出する特徴のレベルと、コンテンツイメージから残すべきものの適切なバランスを見つけることです。

$$F = \alpha L_{content}(c, x) + \beta L_{style}(s, x) + \gamma L_{TV}(x)$$

ここで、c はコンテンツイメージ、s はスタイルイメージ、そして x は変換中のイメージを表します。

これは、F を最小化する必要がある、最適化の問題です。アルファ、ベータ、ガンマはそれぞれ、コンテンツの損失、スタイルの損失、全体的な変動の損失に関連付けられる重みです。全体的な変動の損失については、まだ触れていませんが、これは正規化するための単純な関数で、スタイライズしたイメージ全体の滑らかさを保ちます。

$L_{content}$ と L_{style} を計算するには、訓練済みの VGG 16 ネットワークから層を選択し、(c,x) と (s,x) を比較します。通常は、Adam オプティマイザーやメモリー制限 BFGS 法 (L-BFGS) などのオプティマイザーを使用します。

私が知っている最速のスタイライゼーション・レートは、Nvidia* TITAN Xp グラフィックス・カード上で約 1 秒あたり 1 フレームというものです。処理自体が非常に複雑ではあるものの、これはリアルタイムでは遅すぎます。

絵画風加工について理解したところで、私がやろうとしていることについて述べたいと思います。

Visual Geometry Group (VGG) 16 のバイナリー化

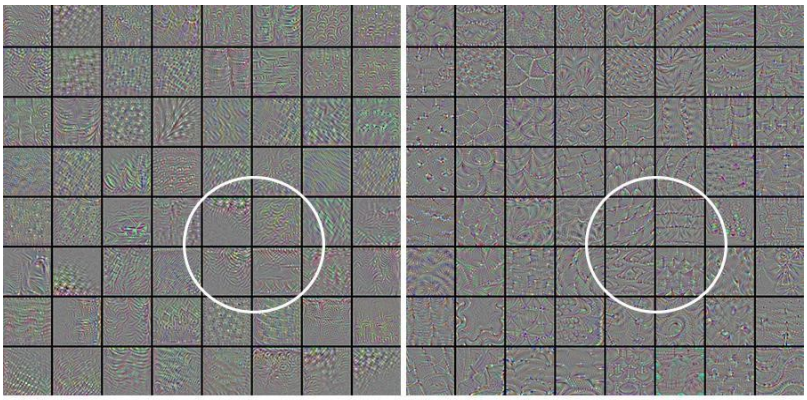
バイナリー化 (2 進化) には、こちらの論文 (英語) にある次のアルゴリズムを使用しました。

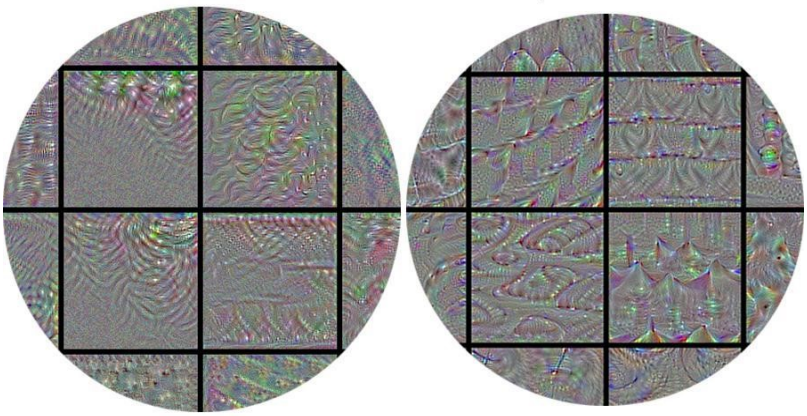
$$
\begin{aligned}
&\textbf{for } l = 1 \text{ to } L \textbf{ do} \\
&\quad \textbf{for } k^{\text{th}} \text{ filter in } l^{\text{th}} \text{ layer } \textbf{do} \\
&\qquad \mathcal{A}_{lk} = \tfrac{1}{n} \|\mathcal{W}_{lk}^{t}\|_{\ell 1} \\
&\qquad \mathcal{B}_{lk} = \operatorname{sign}(\mathcal{W}_{lk}^{t}) \\
&\qquad \widetilde{\mathcal{W}}_{lk} = \mathcal{A}_{lk}\mathcal{B}_{lk}
\end{aligned}
$$

これを VGG16 ネットワークに適用しました。A_{lk} は、重み行列の絶対値の合計の平均です。

バイナリー化は破壊的なプロセスであり、バイナリー化した VGG 16 の結果全体の精度は大幅に低下します。現在、損失を考慮したバイナリー化などのより優れた手法を調査中です。

上記のバイナリー化アルゴリズム適用前後のネットワークの各層の最大活性化を視覚化するのに、インテル® Xeon Phi™ プロセッサー・ベースのクラスターでの関数の実行が役立ちました。





上記のイメージから、バイナリー・ネットワークでは、特徴が非常に抽象化されていることが分かります。フィルター品質が大幅に低下しており、バイナリー・ネットワークでは高レベルの特徴を抽出することは不可能です。

良い文献を信頼することとプランジャーの問題

バイナリー化の結果はあまり良いものではありませんでしたが、バイナリー・ネットワークを訓練すれば、特徴の抽出が可能となり、並列化へと進むことができると私は確信しています。

バイナリー・ネットワークで分類タスクを実行したところ、すべてが犬や猫そしてプランジャーのように分類されました。これは想定していた結果とは異なりますが、VGG 16 分類モデルと VGG 16 no_top モデルは複雑さとはかけ離れています。私はいくつかの論文を信頼して、バイナリー・ネットワークをさらに訓練し、特徴検出の改善を図ることにしました。分類タスクにおいて低精度ネットワークは、最先端のアルゴリズムに近いパフォーマンスを発揮しました。

スピードアップ

通常、ほとんどのニューラル・ネットワークでは 32 ビットの浮動小数点乗算が使用されます。しかし、これは非常に時間のかかる操作です。バイナリー・ニューラル・ネットワーク (BNN) では、これらの乗算をビット単位の XNOR と左/右ビットシフトに置き換えることができます。ネットワークの精度にはさほど影響しないため、これは非常に良い手法と考えられます。BNN 操作については、こちらの記事 (英語) で詳しく説明されています。

ここでは、XNOR ドット積を使用する畳み込みの単純な CUDA カーネル実装を作成します。こちらの記事 (英語) に記述されているように、ビット単位のパック操作により追加の計算が生じますが、フォワード・プロパゲーション・スループットは、最適化されていない GEMM の CUDA ベースライン・カーネルよりも理論的に 32 倍高速になると考えられます。

絵画風加工では、スタイライゼーション・イメージの変更にバックプロパゲーション (誤差逆伝播法) が必要なため、L-BFGS/Adam オプティマイザーに加えて、バックプロパゲーションでこの手法を利用することが重要です。

バックプロパゲーション

$$\frac{\partial E'}{\partial x^{l}_{i',j'}} = \delta^{l+1}_{i',j'} * rot_{180^\circ}\left(w^{l+1}_{m,n}\right) f'\left(x^{l}_{i',j'}\right)$$

説明:

1. l は、l 番目の層を示します。
2. 入力 x は、次元 H×W でイテレーター i と j を持ちます。
3. フィルターまたはカーネル: w と次元のイテレーター m と n。
4. f は、活性化関数です。

 は畳み込み操作であるため、上記の操作によるネットワークのバックプロパゲーションは、並列化することができます。畳み込みについては、こちらの記事 (英語) が参考になります。

まとめ

完全な精度のネットワークと比較して、低精度ネットワークはスタイライズに優れているわけではありませんが、これは訓練されていないバイナリー・ネットワークでは想定されることです。ここでは、インテル® Xeon Phi™ プロセッサー・ベースのクラスター上でバイナリー・ネットワークをトレーニングし、XNOR フォワード/バックワード・プロパゲーションを設定することに注目しました。畳み込みの CUDA 実装をフレームワークに統合したら、グラフィックス・プロセシング・ユニット (GPU) 上でその効果をテストする予定です。

まだ、多くの手法を検討中です。超解像度、ダウンサンプリング、より良い損失アルゴリズム (知覚損失)、より良いオプティマイザー、より小さなネットワークなどです。

さらに、より優れた損失関数を利用して、入力イメージを小さなセグメントに分割して、GPU でスタイライゼーションを並列化することも検討しています。

コンテンツ損失の計算では、プライマリー・ネットワークを GPU で実行し、コンテンツ・スループットをビジョン・プロセシング・ユニット (インテル® Movidius™ Neural Compute Stick) にリダイレクトできるでしょう。これにより、絵画風加工がさらに高速になります。

可能性は、絵画風加工以外の分野にも広がります。ネットワーク・サイズの軽減と高速なスループットにより、適切に最適化することで、これらのネットワークを低消費電力のデバイスで実装できるでしょう。

私は、このプロジェクトについて非常に楽観的です。絵画風加工を VR で実現しましょう。

次のステップ: パート 2

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。