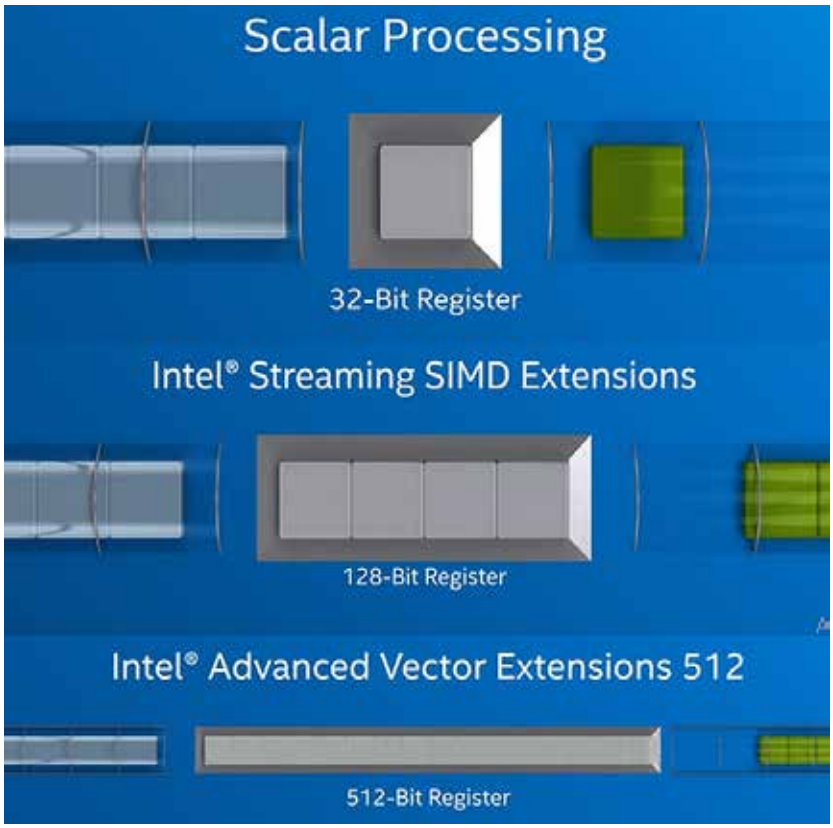
Art'Em – 絵画風加工を VR で実現する: パート 3

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 7 Update」の日本語参考訳です。

Art'Em 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を高速化することを目的としています。

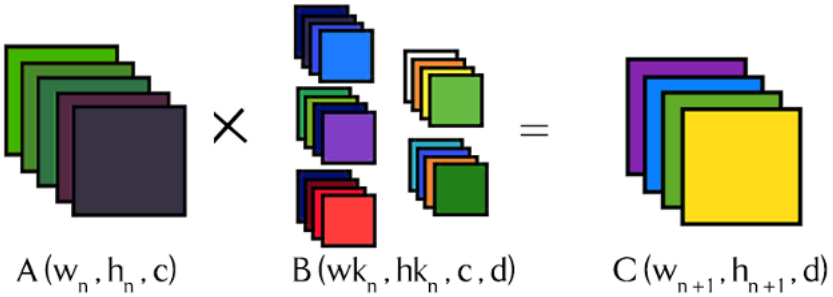
パート 2 では、XNOR (否定排他的論理和) と popcount 操作による乗算の基本的な概念実証を行い、いくつかのベンチマークを測定しました。バイナリー・ネットワークで可能と予測される速度に達しましたが、本格的な実装については取り上げませんでした。ここでは、畳み込み関数を効率良く実装する方法と、想定される課題について詳しく述べます。

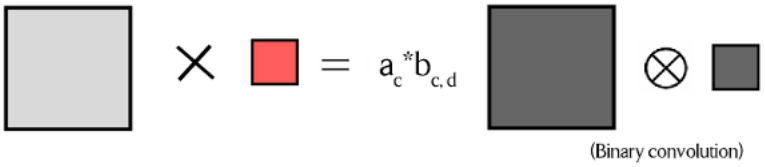
最初に、インテル® Xeon Phi™ x200 製品ファミリーについて見てみましょう。インテル® Xeon Phi™ x200 製品ファミリーは、インテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) 命令をサポートします。そのため、アプリケーションは、512 ビット・ベクトル内で 1 クロックサイクルごとに 1 秒あたり 32 個の倍精度または 64 個の単精度浮動小数点演算をパックできます。ここでは、ビット単位の論理演算や popcount などの組込み関数を利用するため、FMA (Fused Multiply Add) ユニットについては考慮する必要がありません。以下のイメージから、インテルの命令セット・アーキテクチャー (ISA) のベクトル演算への進化が分かります。



出典: https://www.intel.in/content/www/in/en/architecture-and-technology/avx-512-animation.html (英語)

まず、畳み込み操作がどのように行われるか視覚化することが重要です。そして、コードのベクトル化とフォワード・プロパゲーションを並列化する方法について解説します。





Convolution for each d in depth =>

$$C_d = \sum_c a_{[:,:,c]} * b_{[:,:,c,\ d]} (A_{b[:,:,c]} \otimes B_{b[:,:,c,\ d]})$$

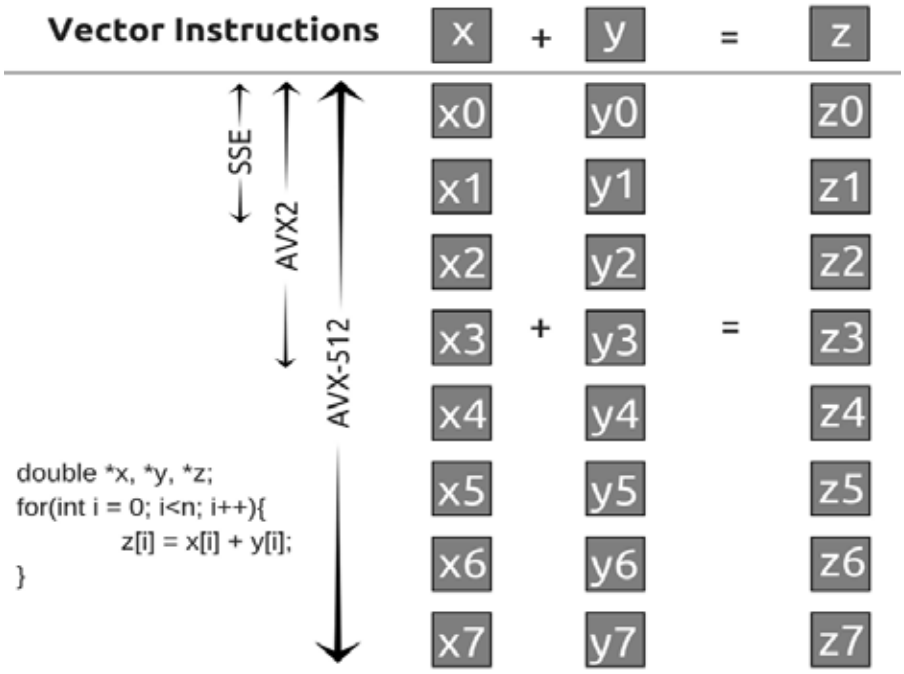
上記の図で、**X** 演算子は通常の畳み込みを示し、**丸で囲んだ X** はバイナリー (2 進) 形式畳み込みを示します。**a** と **b** は、オリジナルの行列を得るためバイナリー化 (2 進化) した行列に掛ける係数です。これらは、完全な精度の浮動小数点数です。ここで、A_b と B_b はバイナリー化した行列です。上記の図の色分けは、各カーネルと重みに関連付けられた異なる係数を表します。

これは、畳み込みにおける基本的な概念をうまく説明しています。標準的な行列乗算操作は、ネットワークの精度を下げることで比較的簡単に並列化できますが、畳み込みは、各部分行列をビット単位操作のデータ型にパックするオーバーヘッド時間が生じるため、並列化では追加の処理が困難です。パート 2 で説明したように、完全な精度のドット積の必要性を排除するには、XNOR を使用して popcount を実行する必要があります。

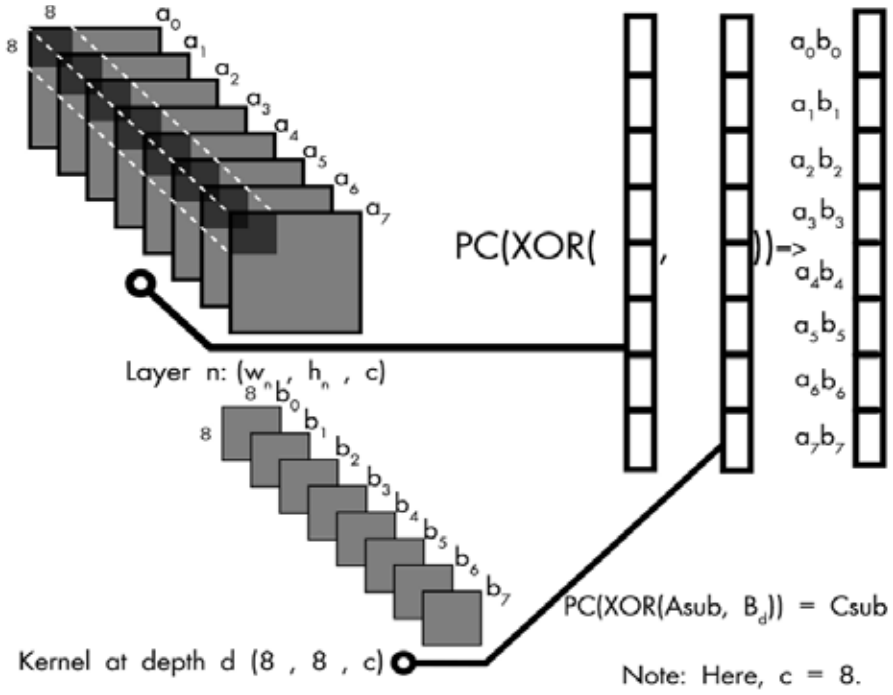
ここでは、スループットを最大化するため、いくつかアーキテクチャーに関する変更を考慮する必要があります。このネットワークで最も重要なことは、インテル® AVX-512 ISA のビット単位の論理演算組込み関数のデータ型 (Int32 と Int64) に部分行列をマスクできることです。ここでは正方カーネルについてのみ考えます。$2^n x 2^n$ のカーネル (n > 1) を使用します。n の値が小さいほうが、カーネルの繰り返し回数が増えます。深い層を作成する場合、n を適切にスケーリングする必要があります。

インテル® Xeon Phi™ x200 製品ファミリーは、32 ビット/64 ビット整数および浮動小数点データのさまざまな操作においてインテル® AVX-512 命令をサポートします。インテル® Xeon®スケーラブル・プロセッサーでは、これが 8 ビット/16 ビット整数に拡張されています。そのため、今後 XNOR-net はより広範にサポートされることが予想されます。

カーネルサイズについては以上です。次に、異なる ISA でループがどのようにベクトル化されるか見てみましょう。



インテル® AVX-512 は、ベクトル化の大きな可能性を示しています。私は、8 つの部分行列を 1 つの _m512i データ型にパックして、ビット単位の論理演算を実行することで、畳み込み操作をスピードアップしたいと考えています。現在直面している課題の 1 つは、インテル® Xeon Phi™ x200 製品ファミリーがインテル® AVX-512 のダブルワードおよびクワッドワードのベクトル popcount (AVX512VPOPCNTDQ) 命令をサポートしていないため、_mm512_popcnt_epi32 組込み命令をインテル® Xeon Phi™ プロセッサー上で利用できないことです。別の popcount 関数の実装を試みましたが、あるボトルネックが開発コード名 Knights Mill (英語) または開発コード名 Ice Lake (英語) プロセッサーがリリースされるまで解消されないことが分かりました。また、ネットワーク実行中の部分行列のビット単位のパック操作の並列化もボトルネックになります。



上記のイメージは、入力の重み付けによりバイナリー化したそれぞれの部分行列のドット積がベクトル化される基本概念を示しています。深さ 8 では、8x8x8 = 512 となり、すべて 1 または -1 になります。これらは、_m512 データ型の Asub と B_d にパックされます。そして、Asub と B_d を XOR して popcount (PC) を実行します。

ここでは、2 つのことを考慮する必要があります。パート 2 では、行列の XNOR を使用しました。しかし、ここでは直接 XOR 組込み関数を使用できます。そのため、Asub と B_d の排他的論理和 (XOR) を使用し、PC (popcount) 関数を調整します。また、PC (popcount) 関数は、上位ビットを実際にカウントした数ではなく、下位ビットの数からセットされている上位ビットの数を引いた値です。各部分行列は Int64 値にパックされ、ビット単位の操作は組込み関数によってベクトル化されます。

これは、8x8 カーネルでは動作しますが、4x4 カーネルでは Int32 データ型に 16 ビットをロードすることになります。32 ビットの残り半分は、最終結果に影響しない値で拡張する必要があります。このため、PC (popcount) 関数も調整する必要があります。この調整は簡単です。Int32 データ型の 16 ビットのみを利用するため、スピードアップの可能性の半分が失われます。しかし、前述のように、将来のインテル® Xeon® プロセッサーでは 8 ビットと 16 ビットの整数がサポートされており、将来はさまざまなカーネルサイズで利用できるでしょう。

行列乗算の並列化は、_m512 データ型でアライメントしたメモリー領域をロードする必要がないことを除いて、GPU でも同様に行われます。popcount 組込み関数があるため、CUDA による実装は難しくないでしょう。唯一のボトルネックは、部分行列をビット単位の論理演算のデータ型に変換することだけです。

インテル® Xeon Phi™ 製品ファミリーでの並列化と畳み込みの GPU 実装については以上です。次に、並列化に適したカスタム・ネットワークについて考える必要があります。インテル® Xeon Phi™ プロセッサー・ベースのクラスター上でのネットワークの訓練では、4x4 カーネルまたは 8x8 カーネルを使用します。XNOR-nets (英語) は、比較的単純な画像認識モデルである AlexNet において約 43.3% という最高の精度を達成しているため、ネットワークの訓練がこのプロジェクトの最後のフェーズになります。私は、このネットワークではイメージ・セマンティクスが比較的良く理解されると信じています。

ここで考察したことにより、畳み込みと汎用行列-行列乗算を効率良くベクトル化できることを願っています。順調にいけば、数週間で良く最適化された畳み込みカーネルが完成するでしょう。その後、Neon などのフレームワークのバックエンドとの統合を試すことができます。これらの実装に取り掛かれるのを楽しみにしています。

前のステップ: パート 2

次のステップ: パート 4

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。