

ACCELERATE CNN PROGRAMS WITH AVX512

WHAT IS AVX512?

AVX512 is a more advanced instruction set architecture (ISA) compared to its predecessor AVX2. It was introduced in 2016 only for the Xeon processors but is now available on newer Intel processors. AVX512 has wider and more numerous vector registers (32 512-bit registers) compared to AVX2's (16 256-bit registers) and other specific instructions that can operate on larger amounts of data in parallel. The increased parallelism can significantly improve the performance for computationally intensive tasks, such as those in machine learning and other data intensive applications.

HOW DOES THIS BENEFIT CNN PROGRAMS?

1. **Convolutions:** Convolution operations are the main component in CNN programs and are computationally intensive. AVX512 provides wider vector registers, more numerous vector registers and efficient instructions for convolution.
2. **FMA and matrix operations:** Taking a closer look at CNN programs we see these can be broken down to matrix operations with connected layers, convolutional layers and filters. AVX512 has faster FMA operations and provides more efficient instructions for matrix operations.
3. **Memory access:** CNN programs require large amounts of memory and hence memory access is a major part in making it more efficient since this can easily become a bottle neck for performance if managed poorly. As we have seen earlier AVX512 has wider vector registers which means we have fewer memory accesses needed and can improve cache utilization.
4. **Activation functions:** Activation functions are generally used in CNNs to introduce nonlinearity into a model and AVX512 provides new instructions for such functions one such example being **ReLU**, which can improve the performance.
5. **Pooling:** Pooling operations are another computationally intensive task like the convolutions which we use in CNN programs to down sample feature maps. AVX512 provides new instructions for pooling operations, which helps improve the performance.
6. **Batch processing:** CNN programs use batch processing for processing multiple inputs at once. AVX512 provides more instructions for efficient batch processing, which can improve the speed of these operations.

Putting it all together we can summarise that AVX512 can improve the performance of CNN programs by providing wider vector registers and more efficient instructions for convolutions, FMA operations, pooling, memory access, activations functions and batch processing.

But to take advantage of all this one must fully understand the program they are working on and have the knowledge of all the various computations in the program with utmost

attention to detail so that they can optimise the code not to the best of their knowledge but to the best the computer architecture can support.

AVX512 INTRINSICS

I will be demonstrating the use of AVX512 intrinsic with the help of a working code.

These are two programs:

1. FMA operation on an image

```
1 #include <opencv2/opencv.hpp>
2
3 using namespace cv;
4
5 int main()
6 {
7     // Load image
8     Mat image = imread("input_image.jpg");
9
10    // Convert image to grayscale
11    cvtColor(image, image, COLOR_BGR2GRAY);
12
13    // FMA operations
14    float scale = 2.0;
15    float bias = 1.0;
16    image.convertTo(image, CV_32F);
17    image = image * scale + bias;
18
19    return 0;
20 }
```

2. Doing the same operation but using AVX512 instructions

```
1 #include <opencv2/opencv.hpp>
2 #include <immintrin.h>
3
4 using namespace cv;
5
6 int main()
7 {
8     // Load image
9     Mat image = imread("input_image.jpg");
10
11    // Convert image to grayscale
12    cvtColor(image, image, COLOR_BGR2GRAY);
13
14    // FMA operations using AVX512 intrinsics
15    const float scale = 2.0f;
16    const float bias = 1.0f;
17    __m512 scale_avx = _mm512_set1_ps(scale);
18    __m512 bias_avx = _mm512_set1_ps(bias);
19    image.convertTo(image, CV_32F);
20    int num_pixels = image.total() / 16;
21    for (int i = 0; i < num_pixels; i++) {
22        __m512 pixel_avx = _mm512_loadu_ps((const float*)(image.ptr<float>(0) + i * 16));
23        pixel_avx = _mm512_fmadd_ps(pixel_avx, scale_avx, bias_avx);
24        _mm512_storeu_ps((float*)(image.ptr<float>(0) + i * 16), pixel_avx);
25    }
26
27    return 0;
28 }
```

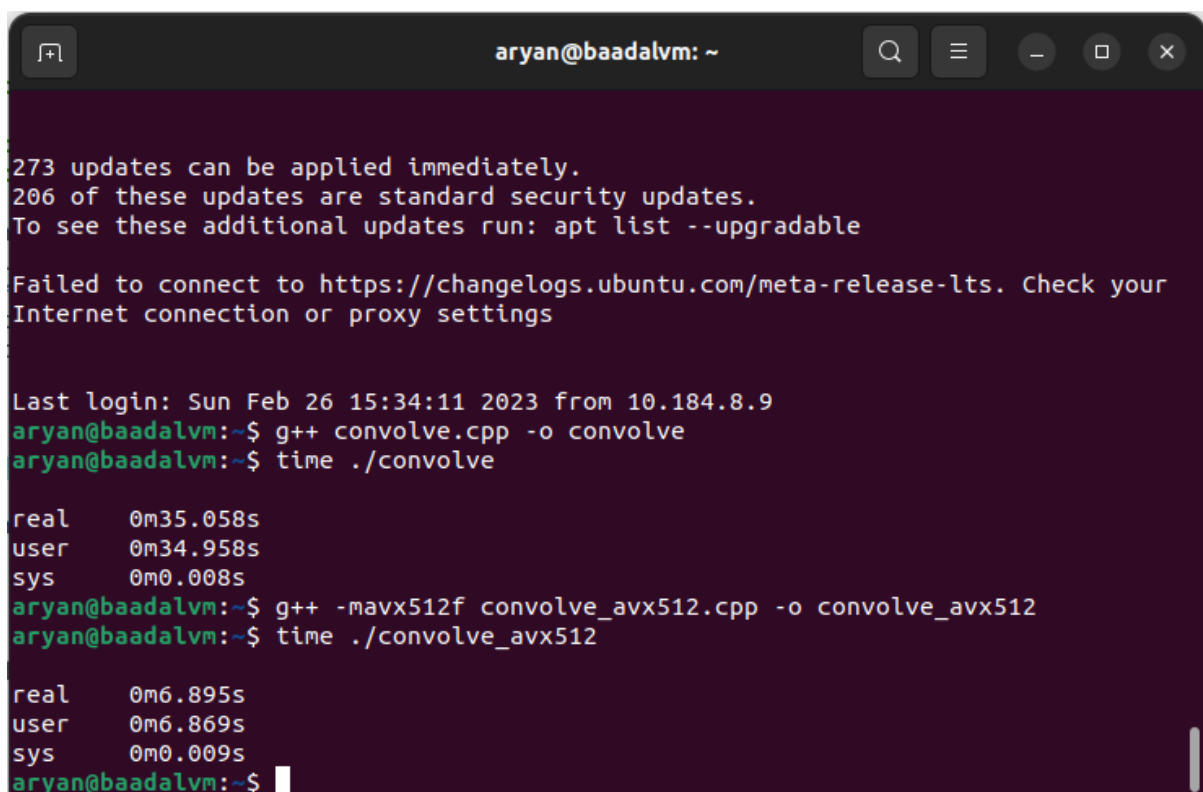
The code compiles but couldn't run it on Baadal VM to check the runtime as installing Opencv (C++) in a VM requires 'sudo'.

So instead wrote a code named convolve.cpp which performs convolution between a input matrix and a kernel matrix and its AVX512 optimised version convolve_avx512.cpp and ran it on Baadal VM to check improvement in runtimes.

Average Runtime of convolve.cpp: **35.2s**

Average Runtime of convolve_avx512.cpp: **6.8s**

The average was calculated over **10** observations one such observation is given below.



```
273 updates can be applied immediately.
206 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Last login: Sun Feb 26 15:34:11 2023 from 10.184.8.9
aryan@baadalvm:~$ g++ convolve.cpp -o convolve
aryan@baadalvm:~$ time ./convolve

real    0m35.058s
user    0m34.958s
sys     0m0.008s
aryan@baadalvm:~$ g++ -mavx512f convolve_avx512.cpp -o convolve_avx512
aryan@baadalvm:~$ time ./convolve_avx512

real    0m6.895s
user    0m6.869s
sys     0m0.009s
aryan@baadalvm:~$
```

This doesn't give an accurate estimate of how much the AVX512 optimised code is faster to the code that can be run on my non AVX512 system because the convolve.cpp was itself not optimised to run on my system it was a simple C++ code performing convolutions. To have it optimised on my system it should be done with AVX2 instructions.

But the overall observation is that we were able to reduce the runtime significantly by using AVX512. If we were to compare the runtimes between an AVX2 optimised program and AVX512 optimised program I would say we would see a **1.5x - 2.0x** boost in performance, which of course depends entirely on the implementation.

ARYAN SANTOSH SAHOO

2019CS10335