Image Processing Ops as first class citizens in MLIR: write once, vectorise everywhere!

Prathamesh Tagore

Speaker: Prathamesh Tagore (VJTI)
Authors: Prathamesh Tagore (VJTI), Hongbin Zhang (ISCAS), Rishabh Bali (VJTI), Yuchen Li (ISCAS)

prathameshtagore@gmail.com
Challenge

Make Image Processing faster 🚄 to make life easier

A possible solution

Use Vectorisation?
CPU Vectorisation in Image Processing

Single Instruction Multiple Data (SIMD) intrinsics in C++

Hardware Specific
Variable length registers

Vectorised low level code

```c
__m128 dot_product_high = _mm256_extractf128_ps(dot_product, 1);
__m128 dot_product_low = _mm256_castps256_ps128(dot_product);
dot_product_low = _mm_add_ps(dot_product_low, dot_product_high);
dot_product_low = _mm_hadd_ps(dot_product_low, dot_product_low);
dot_product_low = _mm_hadd_ps(dot_product_low, dot_product_low);
```
Universal intrinsics
Potential problems while dealing with SIMD intrinsics

- Reliance on a wrapper to provide unified API for targeting each abstraction (Ex. Universal Intrinsics)

- More Code Maintenance
- Separate support for emerging technologies
- Non-trivial design decisions for novel features
Potential problems while dealing with SIMD intrinsics

- Manual reimplementation of existing algorithms for each hardware technology

- More Code Maintainance

- Separate support for emerging technologies
Potential Solution

- Compiler Lowering Passes for Vectorisation in IP
  - Generate vectorisable lower level IR
  - Add support for novel hardware in compiler stack only
  - IP specific stuff is confined in one lowering pass
Why we chose MLIR

- Modular and Reusable Dialects
- Infrastructure for simple rewriting
- Low level codegen via LLVM IR is widely used and tested

MLIR dialect for Image Processing
DIP (Digital Image Processing) Dialect Overview
DIP Dialect Examples

Original Image

Rotation (45°) Output

Laplacian Filter Output

Resize Output
DIP Dialect Examples

Original Image

Dilation Output

Erosion Output
```cpp
#include <buddy/Container.h>
#include <buddy/DIP.h>
#include <opencv2/opencv.hpp>
...

// Read image by using OpenCV API.
cv::Mat inputImage = cv::imread("/path/to/dog.png");

// Convert image data into MemRef container.
buddy::Img<float, 4> image(inputImage);

// Define the size of input and output.
intptr_t inputSize[4] = {1, 224, 224, 3};
intptr_t outputSize[2] = {1, 1001};

// Pre-processing for the input image.
buddy::Img<float, 4> input = buddy::resize(image, inputSize);

// Define the output container.
buddy::MemRef<float, 2> output(outputSize);

// The C++ interface wrapped from custom DIP dialect operations for deep learning pre-processing.
_mlir_ciface_mobilenet_v3(&output, &input);

// Decode the output and print the result.
printResult(output.getData());
```

MemRef container in Buddy Compiler is compatible with existing libraries.

The C++ interface is wrapped from custom DIP dialect operations for deep learning pre-processing.

Deep learning model optimized using buddy-mlir.

Classification: Samoyed
Probability: 0.529544
Single Channel 2D Image Filtering

Normal Convolution Algorithm (9 iterations give 1 element) vs Coefficients Broadcasting Algorithm (9 iterations give N elements)
Considered boundary extrapolation strategies for 2D Image Filtering

Constant Boundary Extrapolation

Replicate Boundary Extrapolation
Anchor point position and padding
Performance Data

Time comparison for kernel size 3x3

Time comparison for kernel size 5x5

Time comparison for kernel size 7x7

Time comparison for kernel size 9x9
Performance data
Operations currently supported by the DIP dialect:

- 1D and 2D Image Filtering
- Image Resizing
- IFFT on Images
- Image Rotation
- FFT on Images
- Image Morphology
Potential Scope

- Target more hardware?
- Integrate with existing MLIR based ML workflows
- Interoperability with linalg dialect
Networking

- Email: prathameshtagore@gmail.com
- GitHub: https://github.com/meshtag
- CV: [Link](https://www.linkedin.com/in/prathamesh-tagore-61aa1a1b1/)
- LinkedIn: https://www.linkedin.com/in/prathamesh-tagore-61aa1a1b1/
- Twitter: https://twitter.com/PrathameshTago1
References

- https://docs.opencv.org/4.x/d6/dd1/tutorial_univ_intrin.html
- https://github.com/buddy-compiler/buddy-mlir
- https://github.com/buddy-compiler/buddy-benchmark
- https://carbon.now.sh/
- Lucidchart