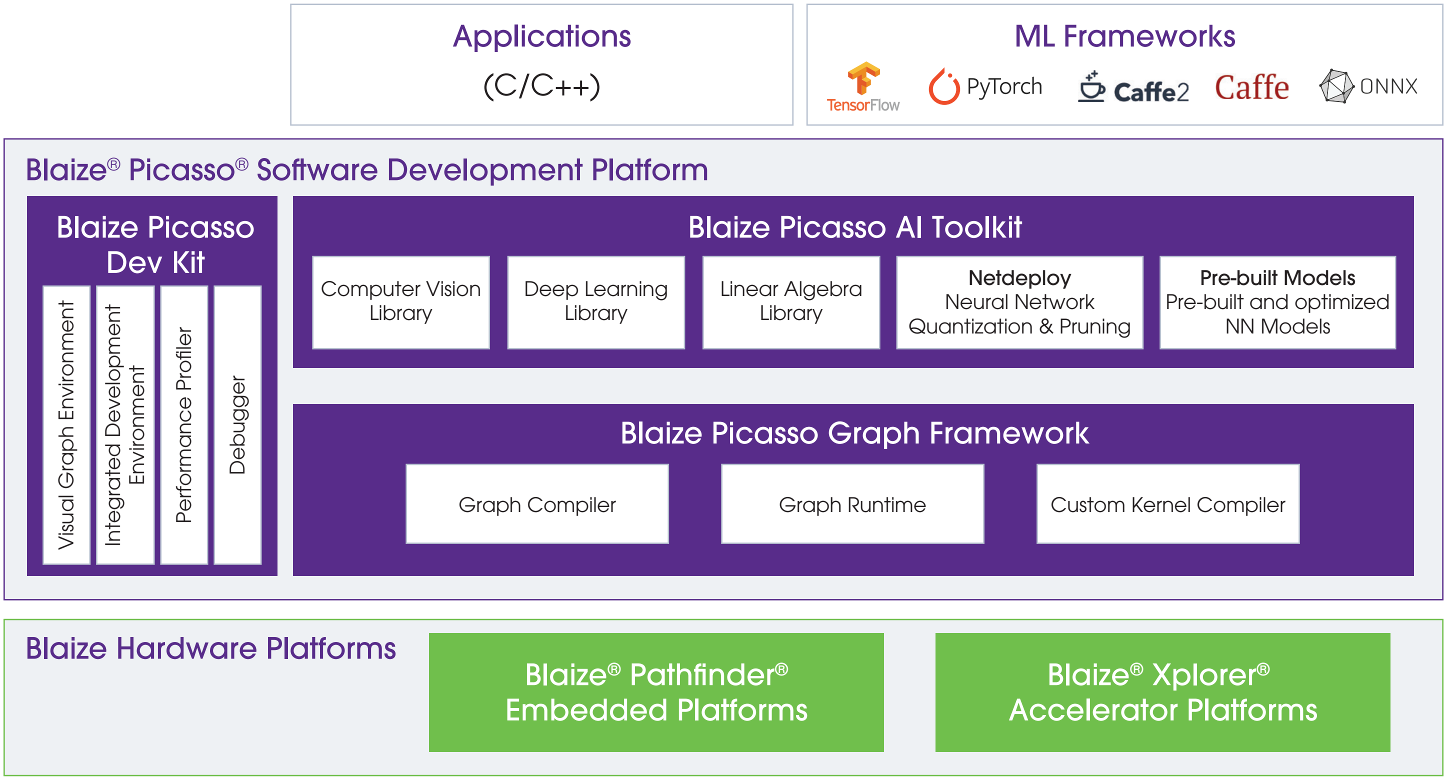
WHITE PAPER

Blaize® Picasso® Software Development Platform for Graph Streaming Processors® (GSP)

Graph-native Software Platform

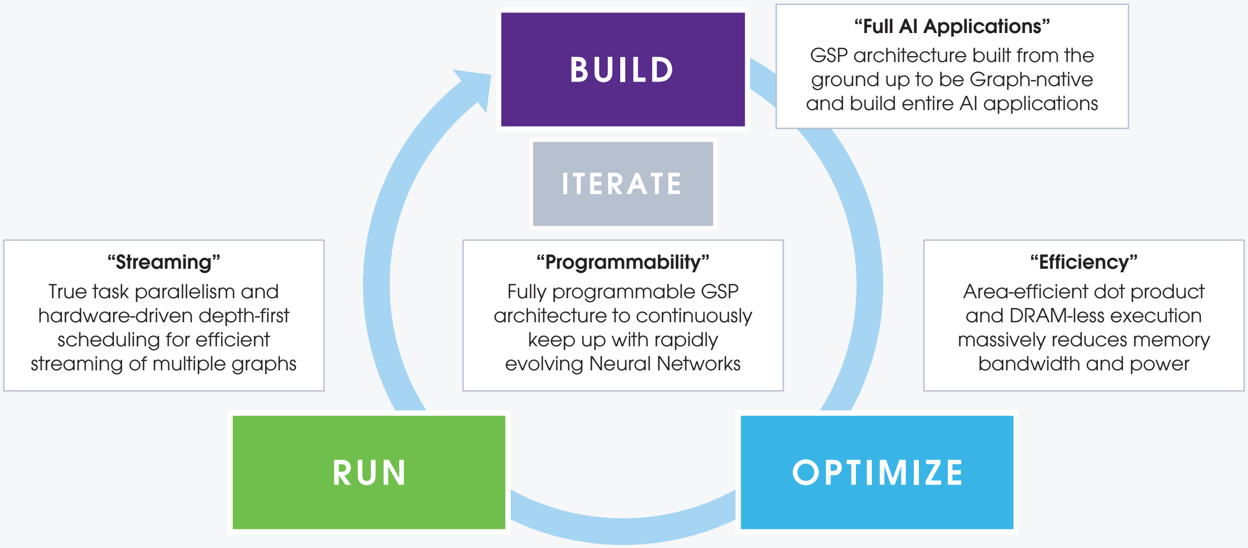
Efficiently Build and Optimize Complete Artificial Intelligence Applications

Blaize® Graph Streaming Processor®, or GSP, is a new architectural paradigm for building and accelerating Artificial Intelligence (AI) applications. In order to take full advantage of this new architecture, Blaize has developed the industry's first truly graph-native software platform that enables customers to build and optimize many AI application from end-to-end as well as deep in hardware.


Applications
(C/C++)
ML Frameworks
TensorFlow
PyTorch
Caffe2
Caffe
ONNX
Blaize® Picasso® Software Development Platform
Blaize Picasso Dev Kit
Visual Graph Environment
Integrated Development Environment
Performance Profiler
Debugger
Blaize Picasso AI Toolkit
Computer Vision Library
Deep Learning Library
Linear Algebra Library
Netdeploy Neural Network Quantization & Pruning
Pre-built Models Pre-built and optimized NN Models
Blaize Picasso Graph Framework
Graph Compiler
Graph Runtime
Custom Kernel Compiler
Blaize Hardware Platforms
Blaize® Pathfinder® Embedded Platforms
Blaize® Xplorer® Accelerator Platforms

Blaize's Picasso® Software Development Platform enables customers to accelerate their entire AI Application development cycle — build, integrate, optimize, run and continuous improvement.

Graph Streaming Processor Architecture Enables Graph-Native AI Application Development



"Full AI Applications"
GSP architecture built from the ground up to be Graph-native and build entire AI applications

"Streaming"
True task parallelism and hardware-driven depth-first scheduling for efficient streaming of multiple graphs

"Programmability"
Fully programmable GSP architecture to continuously keep up with rapidly evolving Neural Networks

"Efficiency"
Area-efficient dot product and DRAM-less execution massively reduces memory bandwidth and power

Blaize's Picasso Platform consists of three major components:

- Blaize Picasso AI Toolkit
- Blaize Picasso Graph Framework
- Blaize Picasso Dev Kit

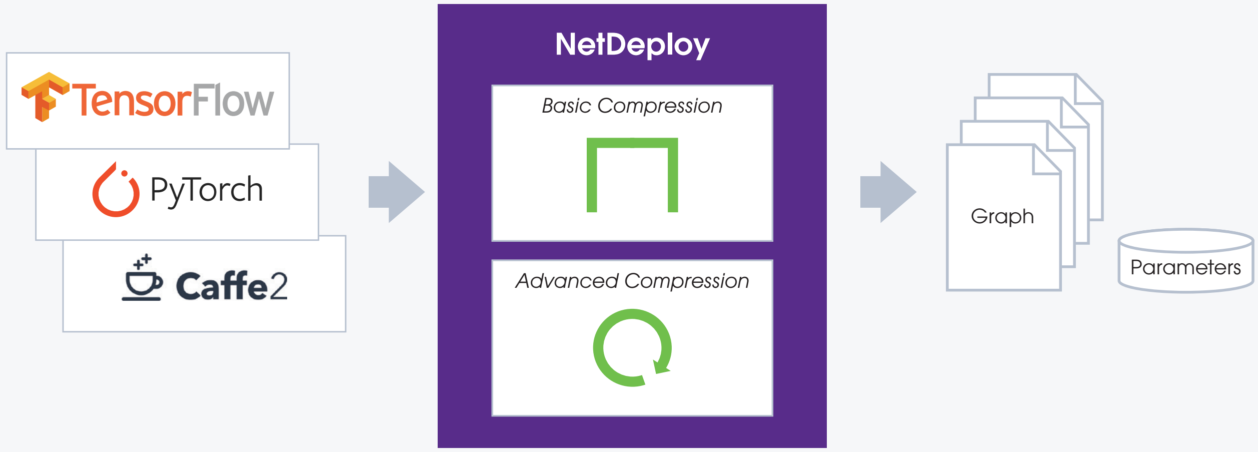
Blaize Picasso AI Toolkit: Libraries and Tools to Build Entire AI Apps

Blaize Picasso AI Toolkit consists of libraries and tools to build entire AI Applications, including neural network and non-neural network functions. AI application pipelines typically include both neural network operations for object detection, classification, segmentation, etc. as well as functions for Sensor Fusion and Image Signal Processing (ISP) such as noise filtering, color conversion, etc. The Blaize Picasso AI toolkit consists of multiple libraries for Computer Vision, Deep Learning and Linear Algebra to build these non-neural network functions in a graph-native manner. Other existing functions, written in C/C++, can be readily converted into a graph-native format with full GSP hardware acceleration.

NetDeploy: Automated Optimization and Compression

Neural network functions, which are typically trained in Machine Learning frameworks such as Tensorflow or Pytorch, need to be optimized to run efficiently on hardware. Blaize NetDeploy tool automates the optimization and compression process through techniques such as quantization, block sparsity induction and pruning. The core feature of NetDeploy is its ability to perform hardware-aware optimization of neural networks by balancing hardware performance with neural network model accuracy. Using sensitivity analysis, NetDeploy determines which operations are good candidates for compression, selectively removes minimally-contributing parameters, evaluates the resulting change in accuracy, and, if the accuracy loss exceeds a threshold, backs off that particular optimization and tries elsewhere. Other fusing and compression steps perform in a similar fashion. The data scientist need not be concerned with the details of how this optimization is proceeding, focusing only on target performance and accuracy, although complete control is available if desired. NetDeploy can also do more advanced neural network optimization during training to get even more accuracy at a given performance. Optimizations typically take weeks to perform and require deep knowledge of the nuances of neural network optimization. NetDeploy can automate this optimization step in achieving excellent results.

Blaize Picasso NetDeploy Automates Hardware-Aware Neural Network Optimization



Blaize Picasso Graph Framework: Complete Graph-native Framework

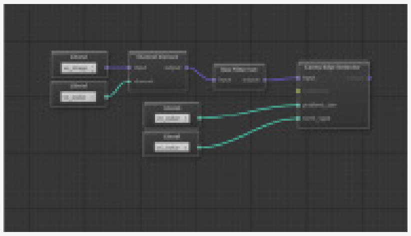
The Blaize Picasso Graph Framework is a complete Graph-native framework built from the ground up for compiling and running graphs natively. Once the entire AI application has been built and the graphs integrated using the tools in the AI toolkit, the graphs are compiled for efficient execution on GSP hardware. The Graph Compiler generates the kernel code needed for each node of a graph as well as a representation of the data dependencies between the nodes. This allows the hardware-implemented GSP scheduler to make optimal use of the internal processors, keeping them busy while eliminating or minimizing the the need to store and receive data off-chip. Libraries for the GSP provide GSP-specific implementations of many common computer vision, signal processing, linear algebra and neural network operations. Developers can also create custom graphs and custom programs — neural or non-neural — using standard programming languages in graph-native format to run fully accelerated on the GSP. This enables the development of AI applications at multiple levels of abstractions, ranging from high-level machine learning frameworks to the node level for real-world graphs that benefit from non-standard neural-network structures. Custom kernels can be written for any node of a graph and compiled using the Kernel Compiler. The Graph Runtime operates in real time, managing various buffers (tensors, images, etc.) and scheduling the graphs for execution on the GSP.

Blaize Picasso Dev Kit: Tools to Verify & Debug

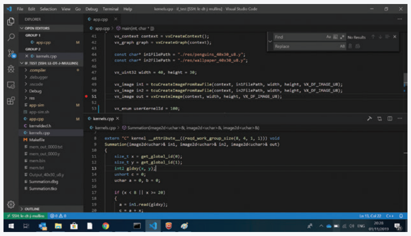
Blaize Picasso Dev Kit provides a comprehensive suite of tools for developers to verify and debug applications running on GSP hardware. These tools include:

- Visual graph environment front end to build and optimize graphs visually
- Integrated development environment
- Performance profiler
- Source-level symbolic debugger that understands graphs and can work at the node level
- Software simulator for high-level behavioral checkout and transaction-accurate application behavior analysis.

Blaize Picasso Dev Kit



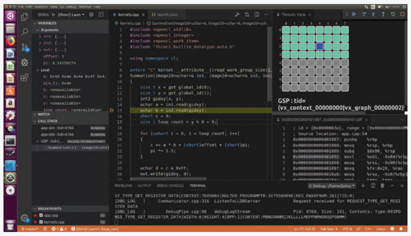
Creating a Graph Visually



```
    vx_context context = vxCreateContext();
    vx_graph graph = vxCreateGraph(context);

    const char* in1FilePath = "./res/penguins_40x30_u8.y";
    const char* in2FilePath = "./res/wallpaper_40x30_u8.y";

    vx_uint32 width = 40, height = 30;

    vx_image in1 = tcuCreateImageFromRawFile(context, in1FilePath, width, height, VX_DF_IMAGE_U8);
    vx_image in2 = tcuCreateImageFromRawFile(context, in2FilePath, width, height, VX_DF_IMAGE_U8);
    vx_image out = vxCreateImage(context, width, height, VX_DF_IMAGE_U8);

    vx_enum userKernelId = 100;
```

```
extern "C" kernel __attribute__((reqd_work_group_size(8, 4, 1, 1))) void
Summation(image2d<uchar>& in1, image2d<uchar>& in2, image2d<uchar>& out)
{
    size_t x = get_global_id(0);
    size_t y = get_global_id(1);
    int2 gidxy(x, y);
    ushort c = 0;
    uchar a = 0, b = 0;

    if (x < 8 || x >= 20)
    {
      a = in1.read(gidxy);
      c = a + x;
```

Integrated Development Env



```
in1: {...}
in2: {...}
out: {...}
offset: 1
pi: 3.14159274
a: 0x4d 0x4e 0x4e 0x4f 0x4...
a[4,2]: 0x4e
b: <unavailable>
c: <unavailable>
y: <unavailable>
loop_count: <unavailable>
```

```
app-sim : tid=4760    RUNNING
app-sim : tid=4766    RUNNING
GSP : tid=(...    PAUSED ON BREAKPOINT
  ::Summation(cl::image2d<uns
```

```
#include <opencl_stdlib>
#include <opencl_integer>
#include <opencl_work_item>
#include "Thinci_builtin_datatype.auto.h"

using namespace cl;

extern "C" kernel __attribute__((reqd_work_group_size(8,
Summation(image2d<uchar>& in1, image2d<uchar>& in2, imag
{
    size_t x = get_global_id(0);
    size_t y = get_global_id(1);
    int2 gidxy(x, y);
    uchar a = in1.read(gidxy);
    uchar b = in2.read(gidxy);
    short c = 0;
    size_t loop_count = y % 8 + 8;

    for (ushort i = 0; i < loop_count; i++)
    {
        c += a * b + (short)offset + (short)pi;
        pi *= 1.5;
    }

    uchar d = c & 0xFF;
    out.write(gidxy, d);
```

```
0x0000000000401887..0x000000000000401a9f
; id = {0x0000003e}, range = [0x00000000000040
; Source location: app.cpp:54
0x0000000000401887: pushq    %rbp
0x0000000000401888: movq     %rsp, %rbp
0x000000000040188b: subq     $0x90, %rsp
0x0000000000401892: movl     %edi, -0x84(%rbp
0x0000000000401898: movq     %rsi, -0x90(%rbp
0x000000000040189f: movq     %fs:0x28, %rax
0x00000000004018a8: movq     %rax, -0x8(%rbp)
```

```
ST_TYPE_GET_REGISTER_DATA|CONTEXT:7605664|HALTED_PROGRAMPTR:1075580096|REG_DWARFNUM:261|TID:0|
[DBG_LOG  ]    Communicator.cpp:316  ListenToLLDBServer          Request received for REQUEST_TYPE_GET_REGI
STER_DATA
[DBG_LOG  ]      DebugPipe.cpp:96    debugLogStream              Pid: 4760, Size: 101, Contents: type:RESPO
NSE_TYPE_GET_REGISTER_DATA|WIDTH:8|HEIGHT:4|BPP:1|CONTENT:MNNOONNMJJKLLLLLMOPPNMNORQQPONMM|
```

Symbolic Debugger Session

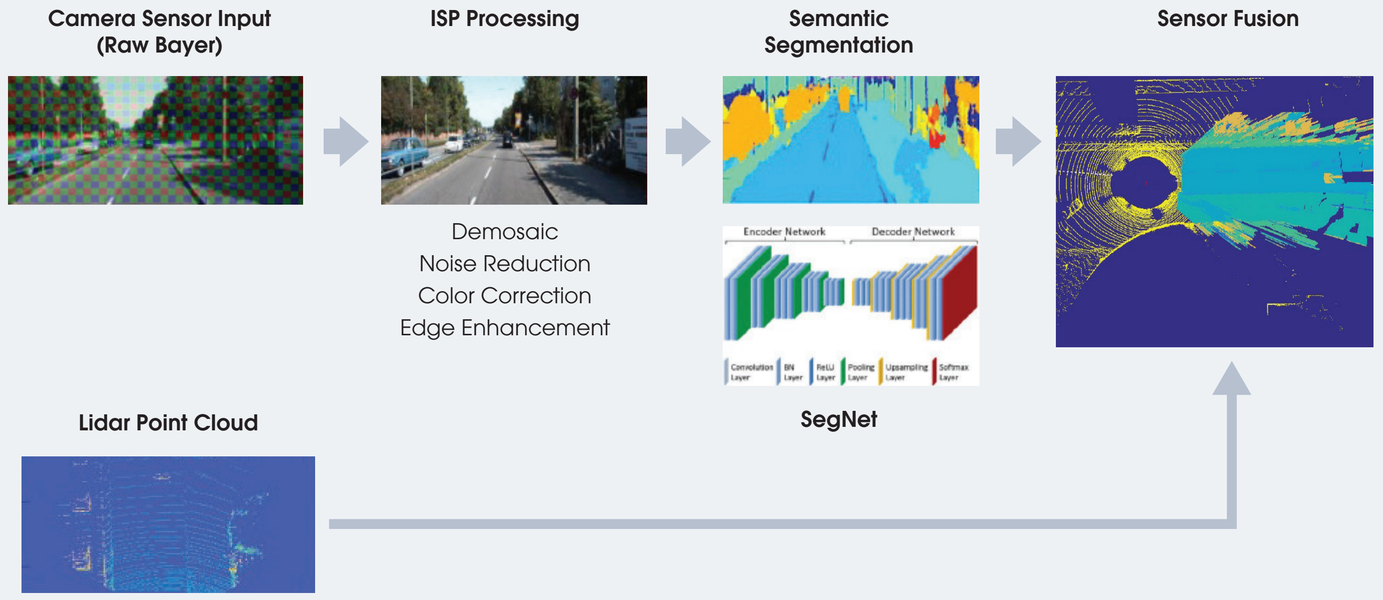
USE CASE EXAMPLE

GSP Delivers Greater Performance and Power Efficiency for Automotive Sensor Fusion

The Blaize Picasso Software Platform is a powerful Graph-native platform for customers to build complete AI applications and take advantage of the programmable and efficient nature of the GSP architecture. A representative complete automotive sensor fusion application consists of non-neural network Image Signal Processing (ISP) functions, neural network processing for semantic segmentation and functions for sensor fusion as follows:

1. ISP functions take raw Bayer input from the cameras and convert into an RGB image. Typical ISP functions include demosaic, noise reduction, color correction, edge enhancement and many others.
2. The RGB image is then processed using a neural network to perform semantic segmentation in order to identify and classify objects in the image at a pixel level. SegNet is a typical semantic segmentation neural network that is based on an Encoder-Decoder architecture and is chosen for its simplicity to meets the need of this example. Other more sophisticated semantic segmentation neural networks such as ICNet and SwiftNet may be selected based on the complexity of the applications.
3. The output of the semantic segmentation network is then fused with LiDAR point cloud data in order to get a more complete understanding of the scene around the car. Sensor fusion of the LiDAR samples and image data allows cars to understand not only what objects are in the scene but also their scale, distance and velocity in 3D space. This information is placed in an occupancy grid-map, where the details of relevant objects surrounding the vehicle are captured.

Complete Automotive Sensor Fusion Application



Camera Sensor Input (Raw Bayer)

ISP Processing

Demosaic
Noise Reduction
Color Correction
Edge Enhancement

Semantic Segmentation

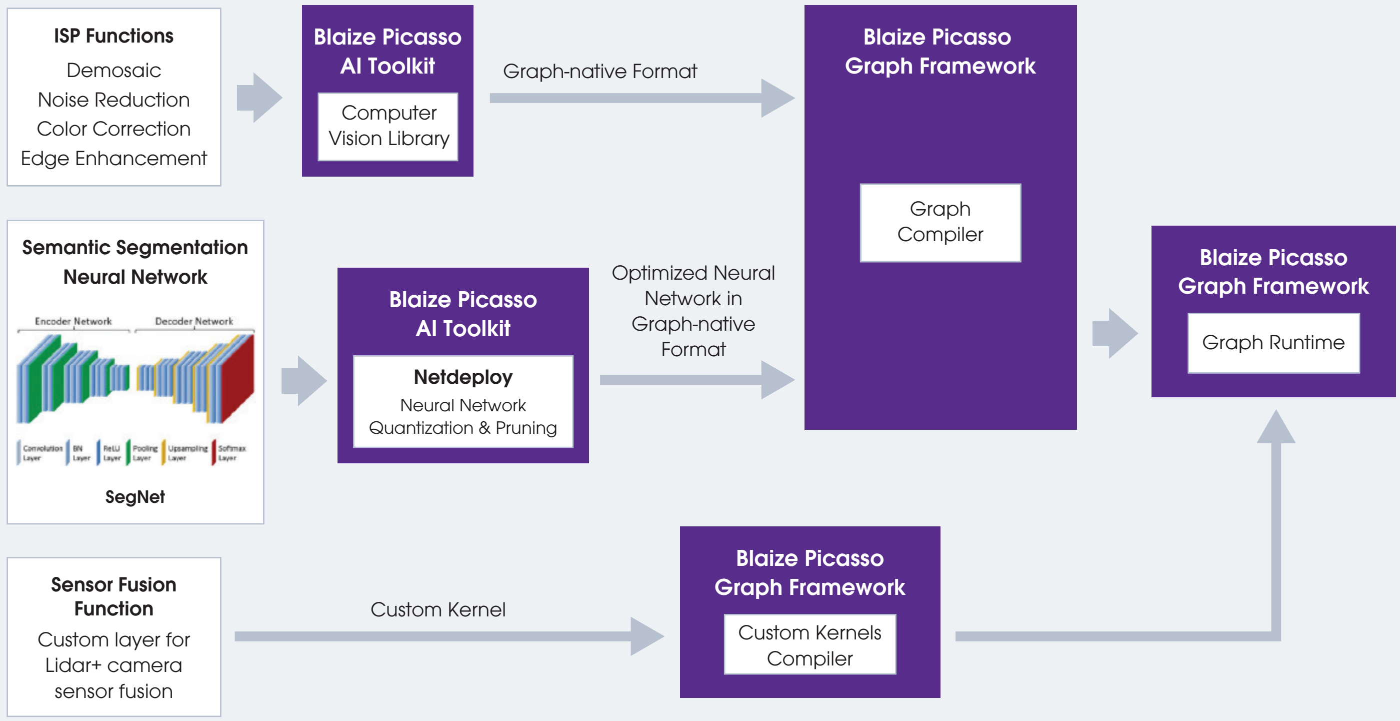
SegNet

Sensor Fusion

Lidar Point Cloud

The different tools in the Blaize Picasso Software Development Platform enable developers to build, optimize and run this complete automotive sensor fusion application on the GSP.

Automotive Sensor Fusion Software Workflow on Blaize Picasso Software Development Platform



ISP Functions
Demosaic
Noise Reduction
Color Correction
Edge Enhancement

Blaize Picasso AI Toolkit
Computer Vision Library

Graph-native Format

Blaize Picasso Graph Framework
Graph Compiler

Semantic Segmentation Neural Network

SegNet

Blaize Picasso AI Toolkit
Netdeploy
Neural Network Quantization & Pruning

Optimized Neural Network in Graph-native Format

Blaize Picasso Graph Framework
Graph Runtime

Sensor Fusion Function
Custom layer for Lidar+ camera sensor fusion

Custom Kernel

Blaize Picasso Graph Framework
Custom Kernels Compiler

The Graph-native nature of the GSP architecture enables customers to run this entire sensor fusion application, both neural network and non-neural network functions, efficiently on the GSP cores in a streaming fashion, providing great advantages in terms of performance and power efficiency. Further, as the overall sensor fusion workflow and neural network architectures evolve, customers can easily update the entire application, taking advantage of the extensive software tools in the Blaize Picasso Platform and the programmable nature of the GSP architecture.

4659 Golden Foothill Parkway, Suite 206, El Dorado Hills, CA 95762 USA
www.blaize.com | info@blaize.com

MWP00070
v1.0, Jan 2024