IE120R Single Chip Image Encoder

Introduction

The IE120R is a high-performance, Verilog-based image encoder designed to convert 896x896 RGB images into 7x7x512 feature maps with a throughput of 100 frames per second and a latency of 11.5 milliseconds. Implementing a VGG-style convolutional neural network (CNN) backbone with 21 layers and 10 million weights, the IE120R utilizes pipelining and parallel execution to achieve efficient, low-latency operation without requiring external components or flow control signals.

The encoder supports AXI-S-compatible real-time streaming interfaces and uses pretrained weights distilled from a ResNet-18 model available on HuggingFace. Its self-contained architecture minimizes intermediate storage requirements by processing images on a per-row basis. The IE120R is optimized for integration into hardware systems, with a clock rate of 250 MHz and a maximum input rate of 100,000 rows per second.

This specification outlines the module's architecture, implementation, and performance evaluation. Key topics include its PyTorch and Verilog implementations, pretraining methodology, functional verification flow, and system integration use cases. The IE120R is designed to address the demands of real-time vision systems, offering a robust solution for applications ranging from robotic control to human imitation learning.

PyTorch Model

The IE120R is implemented as a CNN backbone using PyTorch, consisting of 21 layers composed of Conv2d, BatchNorm2d, and ReLU operations as shown in Figure 1. Five intermediate feature maps are projected into a ResNet-compatible feature space using 1×1 convolution layers with linear activation functions. These projections are employed during pretraining for knowledge distillation.

Installation of the IE120R PyTorch model is supported via the siliconperception package. The pretrained weights are hosted on HuggingFace, enabling straightforward initialization for further development or evaluation. The model supports conversion to its Verilog implementation, including operations for weight quantization and batch normalization fusion.

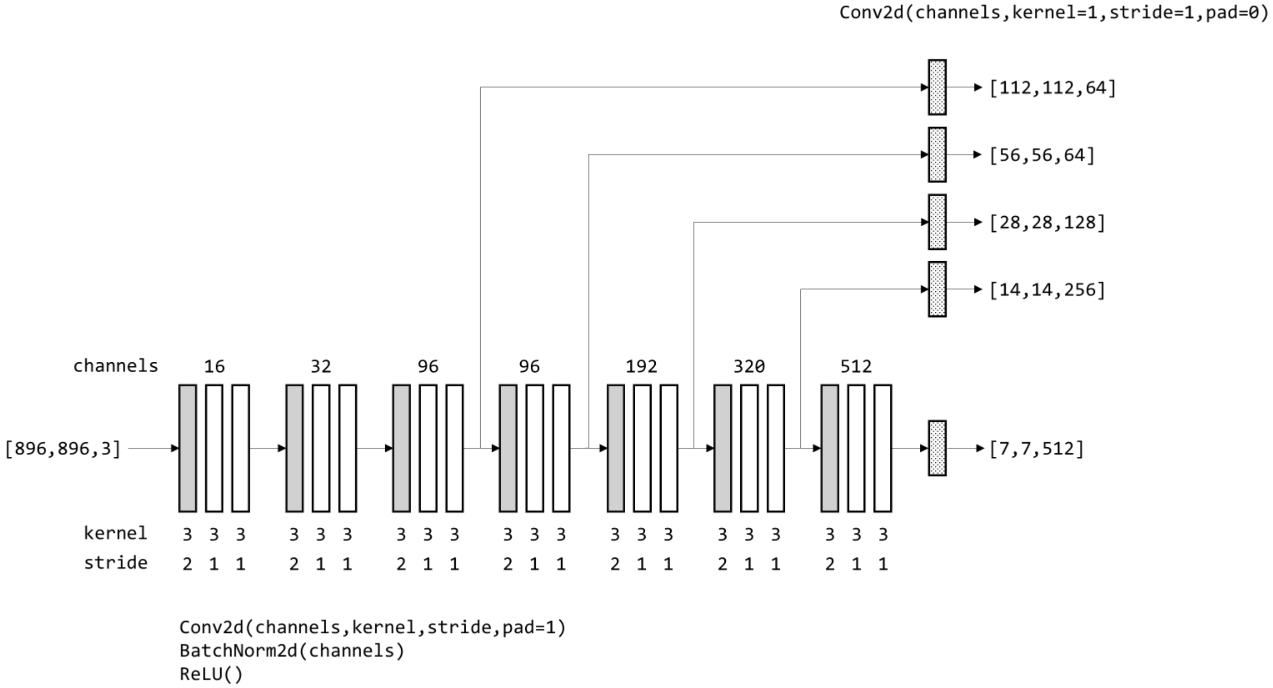

Figure 1

To install the IE120R.

```
pip install --upgrade siliconperception
git clone git@github.com:siliconperception/IE120R.git
```

Model Accuracy Evaluation

The accuracy of the IE120R was evaluated using a single linear layer trained to map the 7x7x512 output feature map to 1000 ImageNet class logits. By freezing the encoder weights and training only the linear classifier layer, we can benchmark the accuracy of the IE120R against a pretrained Resnet-18 feature extractor. This procedure is shown in Figure 2. Note that the IE120R accepts an 896x896 resolution image compared to the 224x224 Resnet-18 input resolution. The output feature maps are the same shapes.

To compare the accuracy of the IE120R and Resnet-18 pretrained encoders, we train for 10000 batches on the Imagenet training distribution and then measure the top-1 accuracy using the validation set. The code below produces the accuracy validation test results.

```
cd IE120R/scripts
./dataset.bash
python classify.py –backbone resnet18
python classify.py -backbone ie120r
```

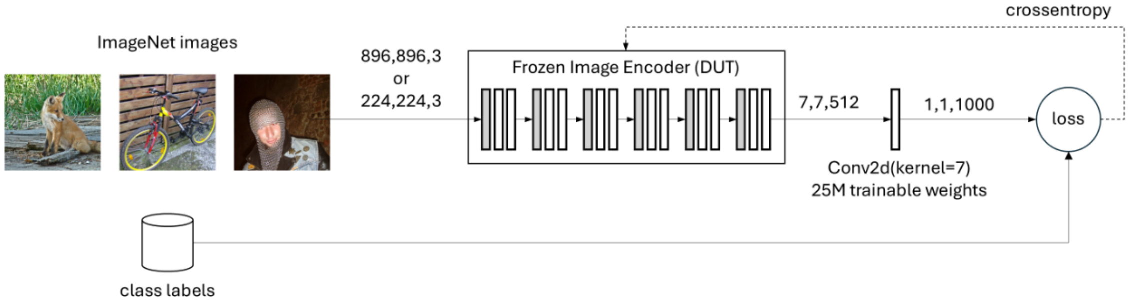

Figure 2

The results from this comparison are shown in Table 1, demonstrating that the IE120R is a functional replacement for the Resnet-18 backbone.

Table 1

	IE120R_HW	**Resnet-18**
Top-1 accuracy @10K batches	45.5%	47.3%
Weights	10.1M, bfloat18	11.1M, float32
FLOPS	0.684T	0.181T
Input shape	[896,896,3]	[224,224,3]
Output shape	[7,7,512]	[7,7,512]

Verilog Module

The IE120R Verilog module is a streaming dataflow implementation of the Pytorch `IE120R()` model, with the following modifications:

- only the 7x7x512 projected feature map is emitted
- BatchNorm2d() layers are merged with the previous layer Conv2d() parameters
- weights are quantized to bfloat18 (1 sign, 8 exponent, 9 mantissa bits)

Note that the bias values and intermediate tensor results use float32 precision.

The module ports are shown in Figure 3. The clock runs at 250Mhz and the reset signal is synchronous to clk.

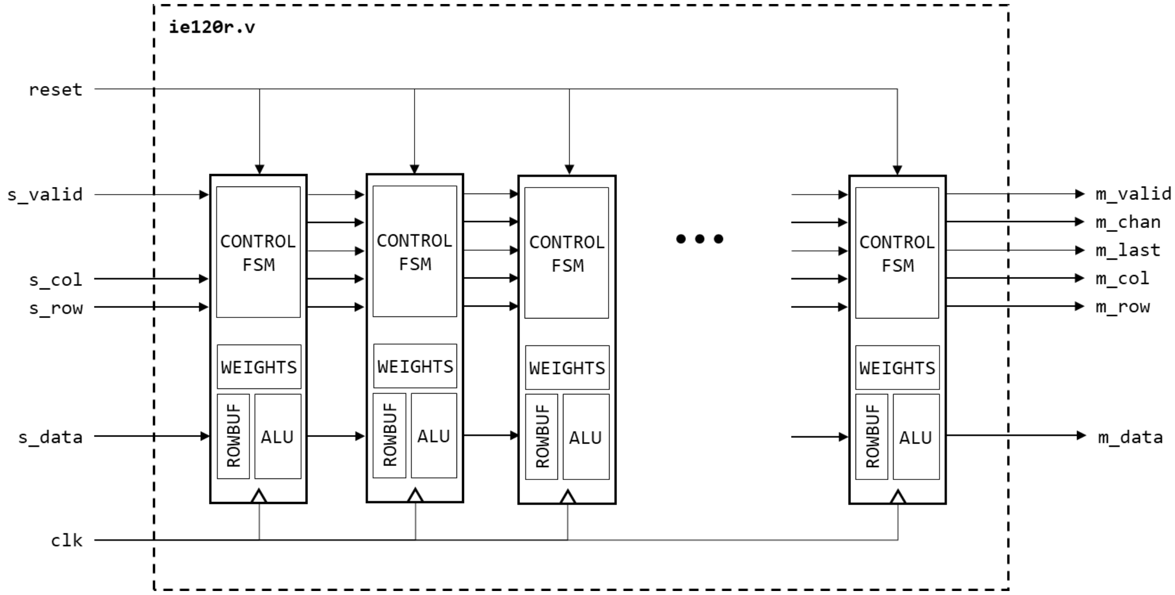

Figure 3

The input signals are s_valid, s_col, s_row and s_data. The s_data bus contains 3*32 signals containing the float32 values of the RGB pixels. The s_valid signal is used to qualify s_data, s_row, and s_col. The s_row and s_col signals range from 0..895. After reset is deasserted, the input interface should receive 896*896 clocks with valid asserted per frame, with corresponding RGB pixels on s_data with s_row, s_col incrementing in 'C' order, corresponding to the pixels in raster order. The row time is designed to run at a fixed 10 microsecond period, which requires s_row to increment with row time T_{row}>= 10us. This rate is equivalent to 100000 rows/s, or 100 frames/s with 12% blanking time at 896 rows/frame.

The ie120r() top level interface protocol is shown in Figure 4 and Figure 5.

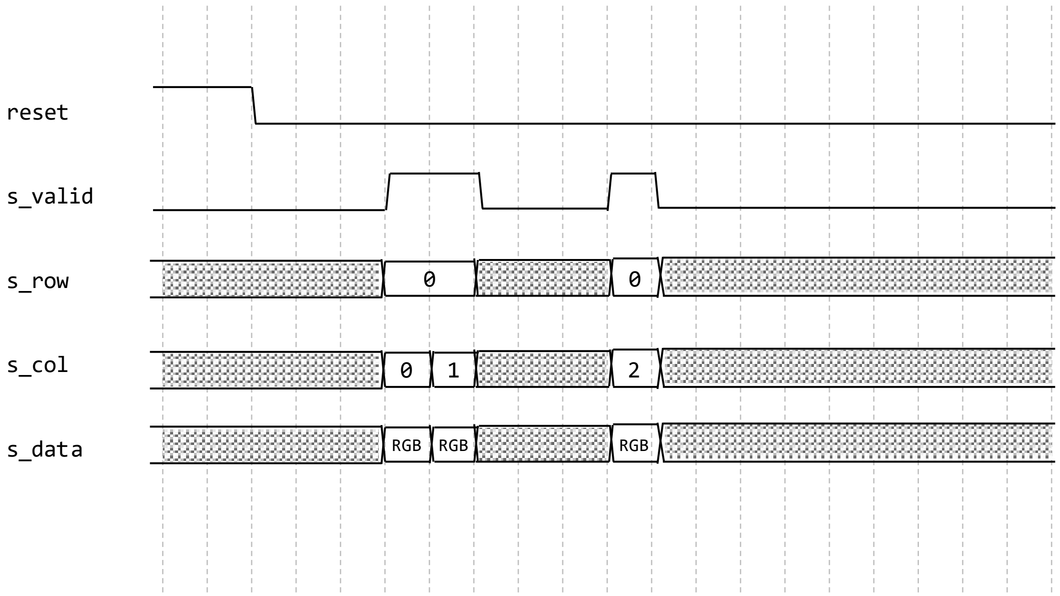

Figure 4

The output signals are m_valid, m_row, m_col, m_chan, m_last and m_data. The output features are interleaved in the channel dimension. The equation to map the ie120r() outputs m_chan[5:0] and m_data[8*32-1:0] to the deinterleaved m_data_flat[512*32-1:0] is:

```
for i=0…7, mchan=0..63 :
    m_data_flat[(511-i*64-m_chan)*32 +:32] = m_data[i*32 +=32]
```

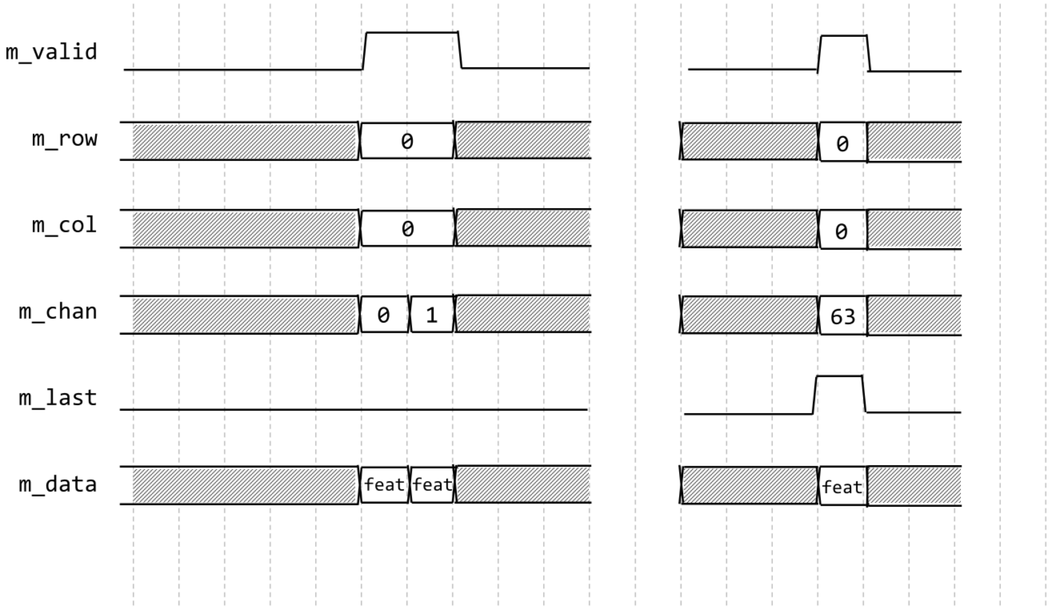

Figure 5

To prepare a trained Pytorch IE120R model to run on the Verilog hardware, the Batchnorm2d() layers must be merged with Conv2d() and the weights quantized to bfloat18 format. The IE120R_HW() model is a Pytorch model which contains methods to perform these operations. Here is code to create a hardware model using the pretrained weights from HuggingFace and then generate files containing the layer weight and bias values.

```
encoder = IE120R.from_pretrained(‘siliconperception/IE120R’)
hw_encoder = IE120R_HW(encoder)
hw_encoder.write_memh(x)
hw_encoder.write_mif(x)
```

Functional Verification

The `IE120R/verilog/sim` directory contains a testbench which verifies that the Verilog code matches the Pytorch reference implementation. The `test.py` program first loads the `IE120R_HW()` model with pretrained weights, then evaluates the model using a test image, and finally writes Verilog `.memh` files for the weight and bias values, the input image, and the intermediate layer activations from Pytorch. These `.memh` files are read by the testbench `test.v`, which first preloads `weight*.memh` and `bias*.memh` into per-layer memories, then injects `input.mem` serially into the encoder `ie120r.v`, and finally checks the output against the Pytorch reference outputs `activation*.memh`.

The verification flow is shown in Figure 6.

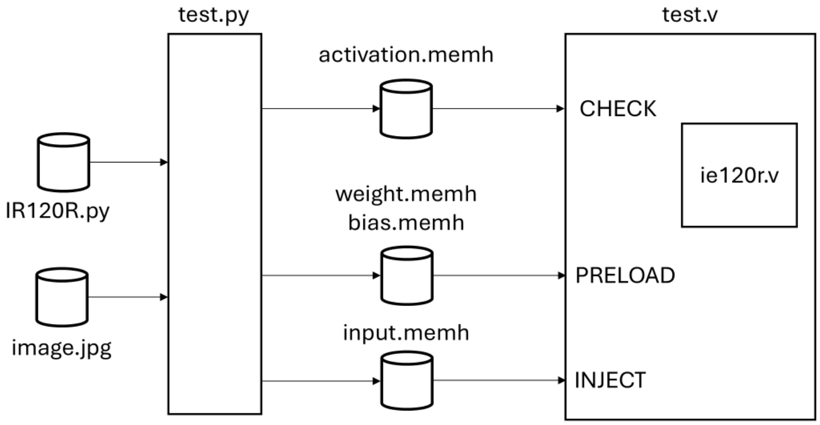

Figure 6

To run the simulation

```
pip install numpy torch torchinfo cv2
pip install --upgrade siliconperception
```

```
python3 test.py
iverilog -v -g2012 -o sim.vvp ie120r_func.v test.v -s tb
vvp sim.vvp -lxt2
```

The waveform in Figure 7 was produced by `test.v` and shows the data flow through 22 layers, the 11.572 ms latency from last pixel in to last feature out, and continuous pipelined operation from frame 0 to frame 1.

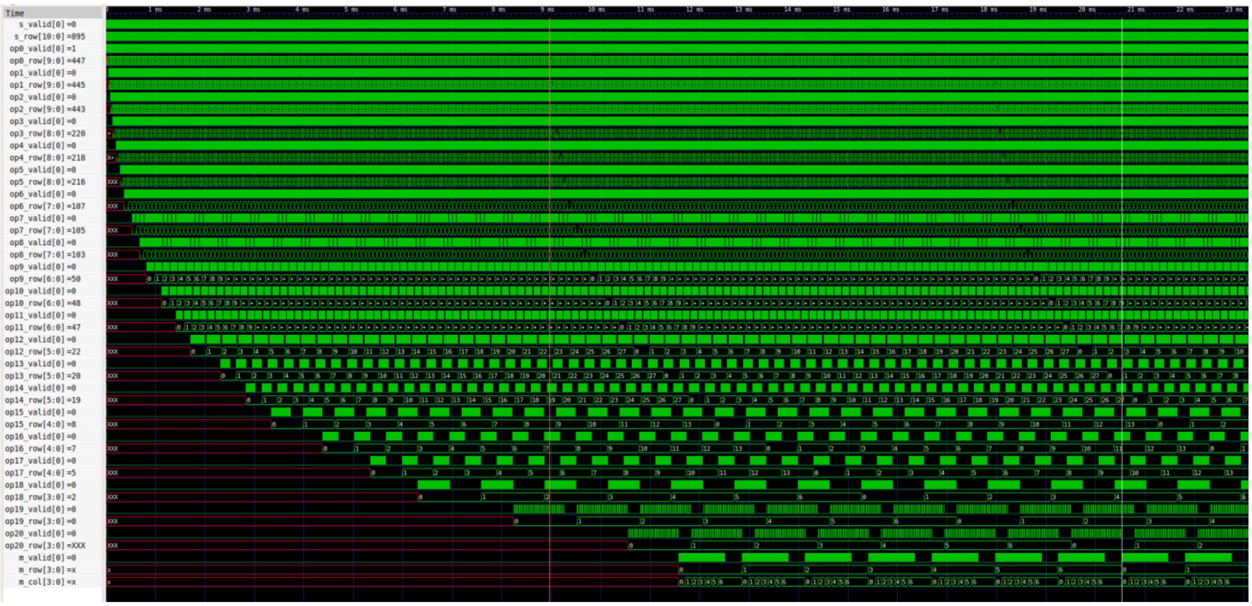

Figure 7

Quartus Compilation

The `IE120R/verilog/agilex` directory contains the files necessary to compile the IE120R using Quartus.

```
top.v, top.sdc, ie120r_agilex.v *.ip
```

The weight and bias values are stored in Agilex ROM IP, which is initialized using a memory initialization file and compiled into the bitstream. IE120R.py currently only produces .memh format files for weight and bias via write_memh(). TBD add support to produce .mif or .hex format files for bitstream generation.

```
Flow Status Successful - Thu Dec 26 12:55:07 2024
Quartus Prime Version   23.4.0 Build 79 11/22/2023 SC Pro Edition
Revision Name     top
Top-level Entity Name   top
Family      Agilex 7
Device      AGIB027R29A1E1VB
```

```
Timing Models     Preliminary
Power Models      Preliminary
Device Status     Preliminary
Logic utilization (in ALMs)   266,302 / 912,800 ( 29 % )
Total dedicated logic registers     435942
Total pins  394 / 1,040 ( 38 % )
Total block memory bits 216,203,136 / 271,810,560 ( 80 % )
Total RAM Blocks  12,978 / 13,272 ( 98 % )
Total DSP Blocks  3,960 / 8,528 ( 46 % )
Total HSSI HPS    0 / 1 ( 0 % )
Total HSSI R-Tiles      0 / 3 ( 0 % )
Total PLLs  0 / 36 ( 0 % )
```

Pretraining

The IE120R has been pretrained using knowledge distillation from the timm resnet18 weights. The pretraining starts with unlabeled images drawn from the Imagnet training set. The unlabeled images are resized to 224x224 and evaluated using the pretrained timm resnet18 model to produce target labels for the output feature maps. These dense labels are used to train the output of the IE120R using a pairwise distance loss function. This approach produces a pretrained IE120R student model which is effectively a clone of the timm resnet 18 teacher model. This procedure is shown in Figure 8.

`IE120R/scripts/pretrain.py`

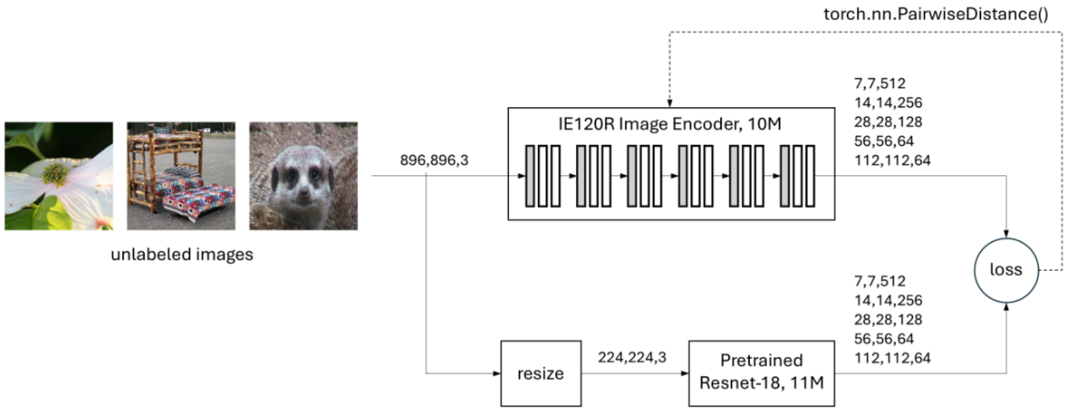

Figure 8

Reference Designs

The IE120R can be used as a component in a real time system. To highlight the high speed and low latency capabilities, the following concepts are potential reference design projects.

Figure 9 shows a simple active vision demo with a low latency vision to action loop. The images are captured using an image sensor with 896x896 RGB resolution, 100 frames/s, and low latency PCIe connectivity. For example, the XIMEA camera with PCIE adapter board. A passive PCIe backplane connects the camera interface board to the Agilex development board using the PCIe gold finger connector. The Agilex board reads pixels from the camera over PCIe, sends them to the IE120R module, receives the encoded 7x7x512 feature map, and executes a small decoder to predict servo pose for high speed, low latency keypoint tracking e.g. "follow my finger".

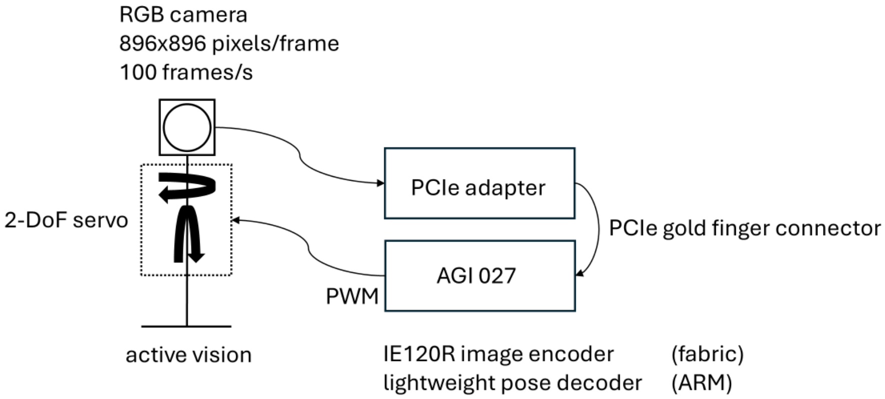

Figure 9

In Figure 10, the stateless pose decoder is replaced with a transformer model running on another device, for example from the Jetson Orin family. The transformer is a sequence-to-sequence model, which enables the pose decoder to perform complex actions guided by the perception tokens from the IE120R. The Agilex board is responsible for reading pixels from the camera and running the IE120R image encoder. The feature maps are sent as a token stream to the downstream transformer model over either PCIe or Ethernet. This type of system is capable of imitation learning using shadowing. For example, it could learn how to control a marble maze game using human imitation. Another example is a "pinball wizard" demo which controls a commercial pinball machine, including ball launch and flippers, also trained from human example. A system with IE120R encoders and a small, fast transformer-based pose decoder can perform sophisticated actions. Multiple IE120R encoders allow full perception with binocular vision, 3D audio, and inertial sensors. The transformer pose decoder predicts the target angle for each degree of freedom DoF in the robot.

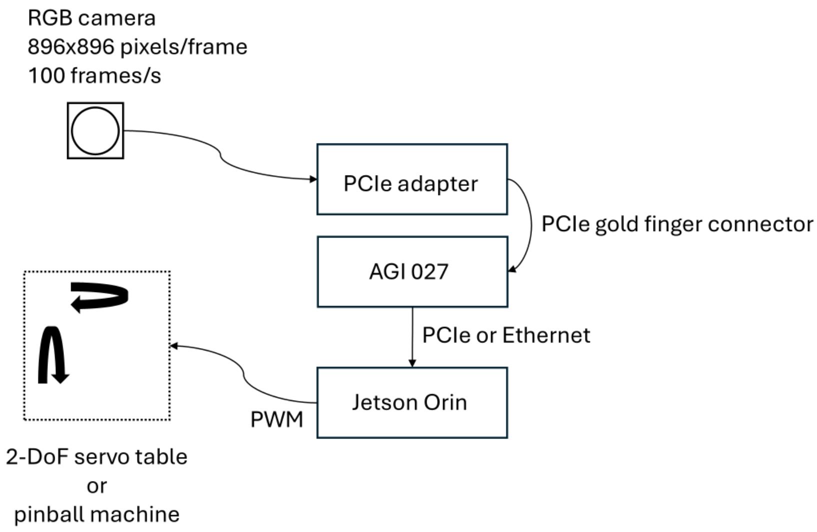

Figure 10

The ultimate application for the IE120R is in a humanoid robot, converting binocular images and spectrogram encoded signals for audio, tactile, inertial and other sensors into a sequence of dense tokens. Intermediate milestones toward this goal include low latency keypoint detection for human shadowing, low latency shadowing using a seated humanoid robot, and fast pose decoding using a small transformer or CNN.