kinara

PRACTICAL REQUIREMENTS FOR MISSION-CRITICAL EDGE AI APPLICATIONS

TECHNICAL ANALYSIS

Benchmarking exercises focus on individual neural network models for specific tasks, but there are many additional considerations and complexities when deploying real solutions. Most Edge AI applications use multiple AI models to derive the intelligence required for on-device decision-making. For example, our AI partner, Arcturus, specializes in smart city applications that center around detecting, characterizing, and tracking people, vehicles, behavior and emotion. These applications use multiple models to sufficiently analyze the data; we'll describe a few examples below. For now, let's highlight the fact that the Kinara architecture specializes in handling multiple models with no overhead or latency when switching from one model to the next – this contrasts with other AI/ML accelerators that can achieve reasonably high throughput when running the same model continuously but suffer from significant model switch overhead, making them impractical for real-time edge deployment.

Why Edge AI Applications Need to Run Multiple Models

Computer vision applications usually involve multiple AI tasks requiring a pipeline of multiple models. Furthermore, one model's output is often the next model's input. In other words, models in an application often depend on each other and must be executed sequentially. The exact set of models to execute may not be static and could vary dynamically, even on a frame-by-frame basis.

Motion prediction-based object tracking, an example where multiple models are required, relies on continuous detections to determine a vector which is used to identify the tracked object at a future position. This is a light-weight approach but its effectiveness is limited because it lacks true reidentification capability. With motion prediction, an object's track can be lost due to missed detections, occlusions, or the object leaving the field of view, even momentarily. Once lost, there is no way to re-associate the object's track. Adding reidentification solves

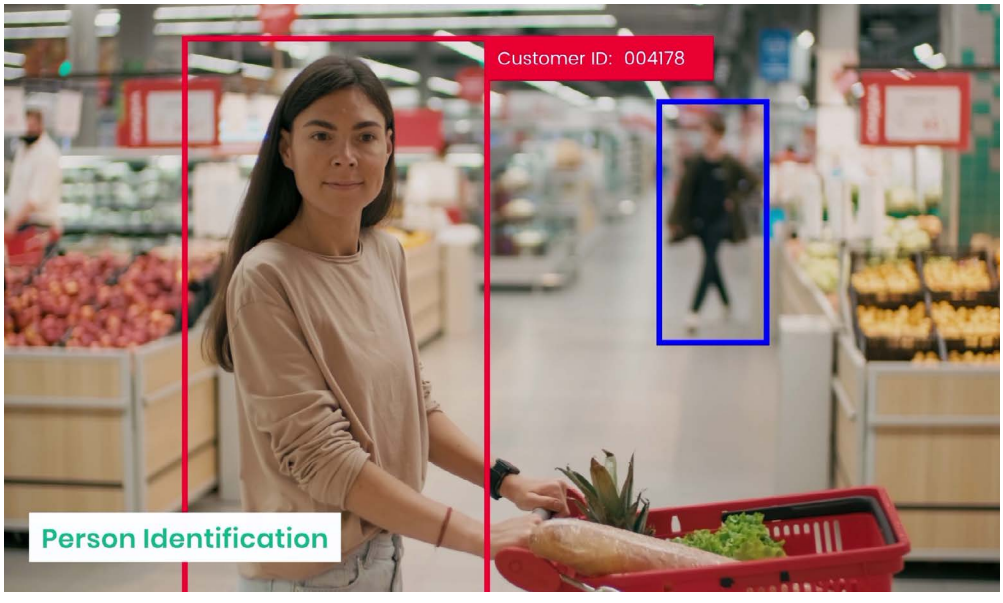

Figure 1.
Cashier-less checkout uses a combination of models for person identification, tracking, pose estimation, and product recognition.

this limitation but requires the addition of a visual appearance embedding –a sort of image fingerprint. Appearance embeddings require a second network to generate a feature vector by processing the image contained inside the bounding box of the object detected by the first network. This embedding can be used to reidentify the object again, irrespective of time or space. Since embeddings must be generated for each object detected in the field of view, processing requirements increase as the scene becomes busier. This drawback requires careful consideration between performing high-accuracy / high resolution / high-frame rate detection and reserving sufficient overhead for embeddings scalability. The key to finding this balance is through model optimization and off-loading model processing.

Face recognition is another example of a common edge AI application with multiple models in sequence. The first stage uses a face detection model on a full frame at a relatively high resolution, such as 720p or 1080p. The higher resolution allows the detection model to find faces far away and closeup. In the second stage, based on the number of faces found in the frame, the application crops out one or more regions of the frame and invokes a face recognition model on each face region. In this case, the models are cascaded such that until the face detection inference is complete, the subsequent face recognition inference can't be launched. Therefore, the AI accelerator must complete the detection with the shortest possible latency before it seamlessly switches to running the inference on the recognition model.

Cashier-less checkout is another example use case. When a person enters the store, the AI accelerator executes a person detection model on each frame to track them as they move through the store. As a person gets close to a shelf, the application invokes a second model, such as pose estimation, to determine if the person is reaching into the shelf to pick up a product. Finally, if a product was picked up, then a third product recognition model is used to determine the product and the quantity. Furthermore, by using a person re-identification model, a store with multiple cameras can track if the person moves from one camera's view to another. This application uses up to four different models that could be executed on a single video frame and decisions of which model executes on each frame varies dynamically based on the in-store activities.

"When a person enters the store, the AI accelerator executes a person detection model on each frame to track them as they move through the store."

What it Takes for AI Accelerator to Efficiently Run Multi-Model Applications

As the use cases above imply, all sequentially invoked inferences must be completed within a very short time (e.g., less than 33 milliseconds for a camera running at 30 frames per second). The Kinara AI accelerators have several design features that enables them to provide high-performance multi-model support with zero-overhead context switching – these include: a) use of a polymorphic architecture; b) dedicated local memory for direct access to models; c) compiler optimized execution plan; and d) command buffering.

a) The polymorphic architecture of the Kinara AI accelerators provide a very high degree of dynamic flexibility – in other words, the programmable on-chip compute resources can instantaneously accommodate all model operators and switch to a subsequent model. In contrast, some accelerators require a configuration phase where the on-chip interconnects and logic blocks are configured for a particular model. For such an architecture, switching from one model to another will incur a substantial overhead.

b) Dedicated local memory of Kinara devices stores all the application's model weights and parameters. The size and number of models that can be stored have practically no limit – the Ara-1 can access up to 2GB. To put it into perspective, models such as Resnet-50 (224x224) and a Yolo V3 (416x416) have only 26 million and 62 million parameters, respectively. But models can be much larger, especially when the input resolution is larger. Furthermore, as noted above, some applications require four or more models. However, 2GB is always sufficient for most Edge AI applications.

Some competing accelerators require the entire set of model weights to be stored on-chip (i.e., they lack external memory support). Supporting multiple models requires that each model is very small or the accelerator must pull the model weights from host memory and switch out the model weights every time a new model is needed for inference – this results in significant model switch overhead.

c) The Kinara compiler optimizes the execution plan such that model weights are continuously streamed in during the inference. Therefore, there is no model load overhead for each inference. The compiler strives to utilize all compute resources on the chip to accelerate a single inference with the lowest latency without requiring batching or pipelining.

On the other hand, other AI accelerators often require batching or pipelining with model inferences of the same model in flight at the same time to fully utilize the hardware compute resources. This often results in a perceived very high inferences per second, but the performance drops substantially in a single-inference, multi-model scenario.

Kinara Ara-1

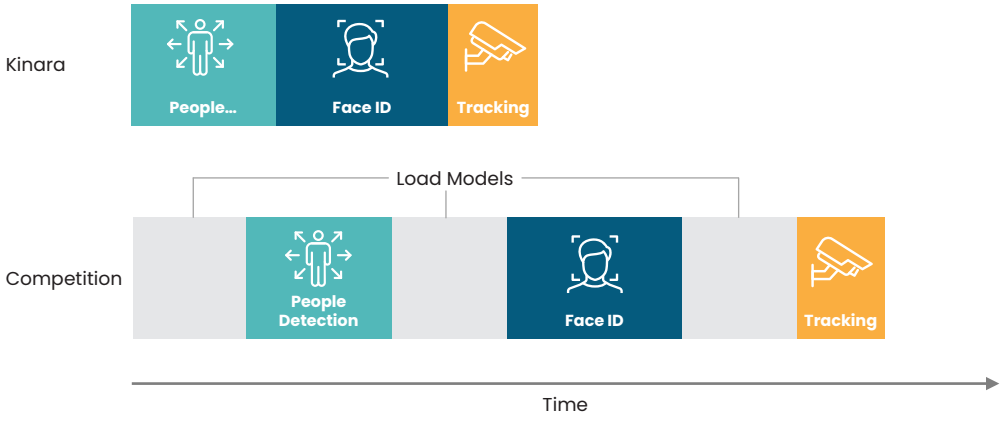

Figure 2.
Kinara Ara-1 executes multiple models with no switching time overhead.

d) The Kinara architecture supports command buffering which allows the Host to send over multiple commands, allowing the Kinara AI processor to buffer model execution requests, and even perform some model weight prefetching to start subsequent model execution.

Complex Edge AI Solutions – Putting the Pieces Together

We began this paper describing the single neural network model for research and benchmarking. But hopefully you're now convinced that real-world Edge AI applications are typically much more complex and require the execution of multiple models. In practice, building a complete system-level solution for these complex Edge AI applications is a concerted effort. The combination of a powerful host SoC, like the NXP i.MX 8M Plus, and the Kinara Ara AI accelerator, delivers the high-performance hardware capable of running complex, multi-model AI applications. And then we have to add in the vision analytics and specific components of the edge AI applications – here we bring in partners such as Arcturus with its Brinq™ products, and together we can provide our customers with class-leading detection, tracking and characterization solutions aimed at mission critical edge applications in public safety, transportation, healthcare, retail, and industrial applications.