When he finished high school, DiPrima took a job repairing televisions and other consumer electronics, which turned out to be an education in itself. "You get to see how everybody screwed up. I think it's a pretty invaluable lesson. You know what RCA did wrong.... You know what Samsung got wrong every time.... Things are super-refined now, but back in the day everything had some peculiar glitch or problem."

When the TV repair business dried up because it became cheaper to replace a TV than fix one, DiPrima worked at a music store repairing guitar amplifiers—"people will still pay $150 to repair a $900 Marshall amp," he says—before getting a job around 2005 at the University of Texas doing work like designing and building lecterns in classrooms.

That's when he and his brother got interested in Tesla coils. "We started building our own Tesla coils for the sole purpose that we wanted them to be musical instruments. We posted some [YouTube videos] as a kind of a hobby." The videos started attracting an audience and the brothers started performing live shows. Eventually, they realized they were making more money doing things like selling T-shirts at their shows than they were earning from their day jobs, and they took the plunge and formed ArcAttack as a full-time occupation.

DiPrima continues to spend a lot of his time reading and researching for his projects. "I kind of regret now that I didn't have more college education, but at the same time I feel like I have a pretty solid foundation of engineering knowledge," he says.

—STEPHEN CASS

POST YOUR COMMENTS at https://spectrum.ieee.org/diprima0519

TAKING AI TO THE EDGE

GOOGLE'S TPU NOW COMES IN A MAKER-FRIENDLY PACKAGE

There's a steady drumbeat about how deep learning is going to touch nearly every area of technology, not least from us here at *IEEE Spectrum.* So, like a lot of folks, I've been interested in getting my hands dirty and learning some of the tools and techniques. But, like a lot of folks, I've been stymied by the difficulty of getting up and running.

I've tried to use TensorFlow—Google's open-source machine-learning library—and Keras, another library that acts as a high-level interface between Python programs and machine-learning back ends like TensorFlow. But it's been a discouraging exercise in going down software-dependency rabbit holes and sifting through fragmented and often obsolete documentation. And then, just when everything finally seems to be working, something breaks thanks to a system upgrade. As the old hacker lament goes, "You are in a maze of twisty little passages, all alike."

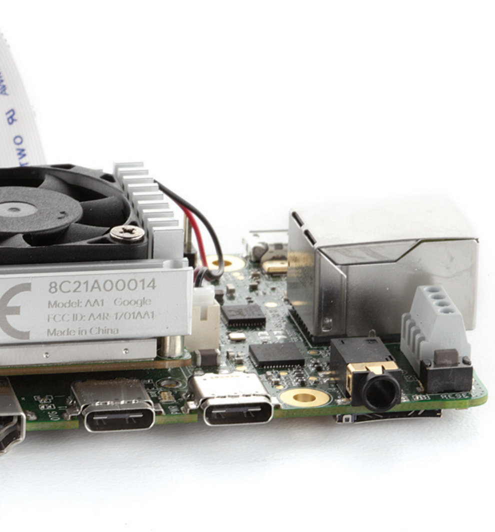
Then Google offered a way out of the maze, with the US $150 Coral Dev Board. The Dev Board looks a lot like a Raspberry Pi, albeit one with a great big heat sink bolted on top. But under that heat sink is a system-on-module built to support Google's Edge Tensor Processor Unit (TPU).

The TPU is a coprocessor optimized for handling neural networks, intended to push artificial intelligence out from centralized clouds to embedded devices. It's not meant for the actual learning phase of machine learning, when models are compiled from sample data sets to determine what output should correspond to a given input. The TPU is designed for the performance phase, when systems with compiled models are presented with real-world data and are expected to behave appropriately using a version of TensorFlow called TensorFlow Lite. In this Hands On, I'm going to be focusing on the Dev Board's hardware and setup, and in a later article I will dive deeper into using models and interfacing with some external hardware.

The Dev Board is designed to make hardware experimentation easy, with a Pi-like general-purpose input/output (GPIO) connector, SD-card reader, HMDI video output, a Wi-Fi radio, an Ethernet port, a port for attaching a camera module, and a USB port for peripherals. Like the Pi 3, it has 1 gigabyte of RAM and uses an Arm-based processor as its CPU.

There are differences, however: The Dev Board's GPIO pinout is similar to the Pi's, but there are fewer general-purpose pins available because the Dev Board's GPIO also supports things like a serial audio interface. Other differences include the presence of 8 GB of onboard flash storage, which hosts the operating system. This leaves the

RANDI KLETT (3)



SD-card reader free for additional storage, unlike the Pi's reader, which is reserved for the OS. The board is powered via a USB-C connector instead of a Micro-USB connector. And be warned: Google recommends using a 2- to 3-ampere 5-volt power supply, while many USB power adapters top out at 1.5 amps.

There's also only one USB-A port, compared with the Pi 3's four. The paucity of USB-A ports is because Google envisions the Dev Board being used to prototype embedded devices, so there's less need to support peripherals (to that end, the core system-on-module can be detached so that a device maker can provide its bespoke supporting hardware). Instead the board provides a lot of support for headless operation, sans keyboard and screen.

The board has a Micro-USB port for a dedicated serial console interface, which can be used to monitor the system and supervise flashing an OS (a customized version of Debian Linux called Mendel) to the onboard storage. There's a second USB-C port, intended for connecting the board to a machine running Linux. From there, users are expected to use the SSH protocol to log in to the Dev Board for normal use.

Because of my prior experience with machine learning, I was a little apprehensive when I unwrapped my Dev Board, not least because it was a preproduction unit, where rough edges often abound.

But I needn't have worried. An insert directed me to a Web address to get started. I was pleasantly surprised to find a set of well-illustrated step-by-step instructions (albeit for folks comfortable with using a command-line interface), including directions for attaching the specially designed camera module Google provided.

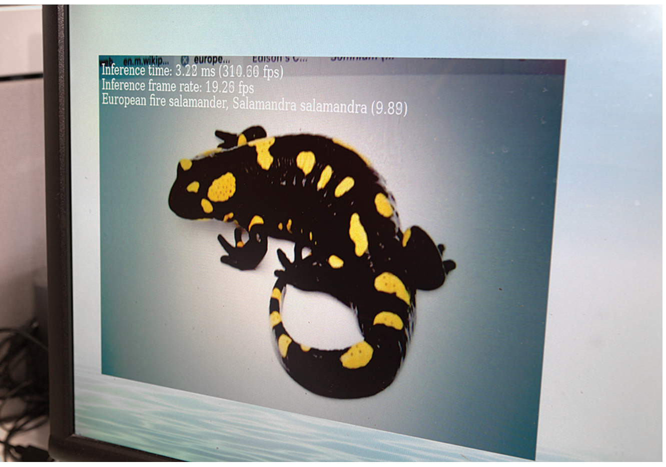
I used a Raspberry Pi as my front-end Linux computer for the setup and was quickly able to download to the Dev Board the precompiled models required for some demos. Once everything was running, I also had no problem SSHing in from my MacBook Pro.

There are two camera demos, both of which use live video: One can detect when faces come into view, while the other can recognize a somewhat eclectic collection of 1,000 objects, including "coffee mug," "garbage truck," and "European fire salamander." Bounding boxes (for faces) or text labels (for objects) are overlaid on the video feed. Google gives instructions on how to stream the overlaid feed to your front-end computer, but I didn't have much luck in getting smooth results. However, with a monitor plugged directly into the Dev Board it ran impressively well.

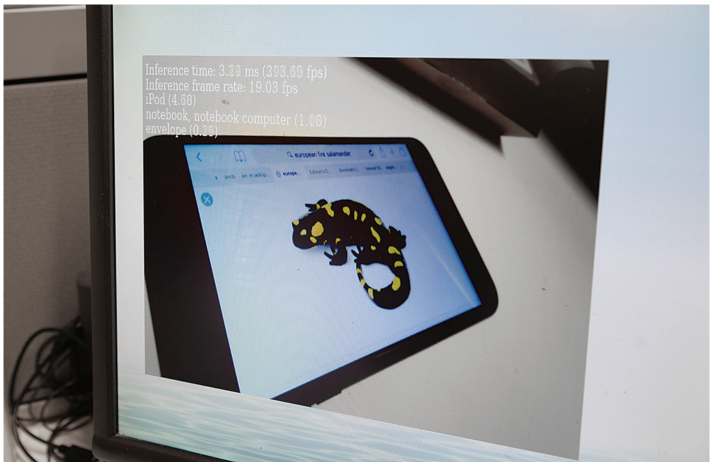
In addition to the camera-demo data, Google offers a small selection of other models that can be used with a Python application programming interface (API) to communicate with the TPU, and it's possible to create and upload your own. However, currently the Python API is limited to processing static images rather than the camera feed. At press time, the Dev Board team said they were considering releasing an extended API, which I hope to use as the basis for my follow-up article. I can finally see the maze's exit.

—STEPHEN CASS

POST YOUR COMMENTS at https://spectrum.ieee.org/devboard0519



```
Inference time: 3.22 ms (310.60 fps)
Inference frame rate: 19.26 fps
European fire salamander, Salamandra salamandra (9.89)
```



```
Inference time: 3.29 ms (303.69 fps)
Inference frame rate: 19.03 fps
iPod (4.68)
notebook, notebook computer (1.00)
envelope (0.26)
```

REAL-TIME RECOGNITION: For one demo, I put a picture of a salamander on my iPhone's screen. When I held the Dev Board's camera module so that the onscreen salamander filled the view, the board took just a few milliseconds to identify it. As I pulled the module back so that the iPhone's body came into view, the board instantly corrected its guess about what it was seeing, despite the salamander still being prominent.