Miguel Millan

Report: Vision Bonnet Cannot Run Our Model

After doing extensive research/code review of the AIY Vision Kit and the code provided, I have determined that we will not be able to utilize the visual processing unit (VPU) of Google’s Vision Bonnet to run our *specific* models. In this report I shall detail the steps required to run a model on the Vision Bonnet and explain why we are unable to do so with our specific use case.

Steps to Run the model:

1. Save your trained model as a tensorflow ‘frozen graph’
   1. With some code modification, we should be able to do this.
2. Download and run Google’s ‘bonnet\_model\_compiler.par’ to compile the frozen graph of the model into a binary file that can be run on the Vision Bonnet hardware:

./bonnet\_model\_compiler.par \  
 --frozen\_graph\_path=<frozen\_graph\_path> \  
 --output\_graph\_path=<output\_graph\_path> \  
 --input\_tensor\_name=<input\_tensor\_name> \  
 --output\_tensor\_names=<output\_tensor\_names> \  
 --input\_tensor\_size=<input\_tensor\_size>

(note: this must be run on 64bit linux, NOT THE PI)

* 1. However, there are some constraints that we must adhere to when training our model if we want to use this:
     1. Model takes square RGB image and input image size must be a multiple of 8.
     2. Model's first operator must be tf.nn.conv2d.
     3. Model should be trained in NHWC order.
     4. Model's structure should be acyclic.
     5. When running inference, batch size is always 1.
  2. We’d have to adjust our training method in order to meet these constraints.

1. Assuming we compiled our model into a binary, we need to send it the pi and utilize the inference.py utilities to load the model onto the vision bonnet and run it.
   1. This is where our issue comes in:
      1. The Vision Bonnet is a passive listener: that is, the camera feed passes through the Vision Bonnet to the Pi.
      2. As frames pass through the Vision Bonnet they get run through whatever model is loaded, which then produces an inference result.
      3. The classes provided by inference.py allow us to load a model and retrieve the inference result of each frame that passes. The issue is **we want to pass a sequence of saved frames to the model** which correspond to a single event that needs to be classified, **but with the currently available code we can only pass real time frames from the camera to the model**.

If we were just doing a regular 2D convolutional neural network (CNN) to classify in real time there would be no problem, however we are using a Time Distributed 2D CNN that (with our current setup) requires us to pass a sequence of saved frames to it. It is POSSIBLE for us to write our own set of custom inference classes that would allow us to feed data to the Vision Bonnet. However, we lack the knowledge and time to write so much custom code. As such I believe that it would be best to avoid wasting time on a hardware specific issue like running our model on the Vision Bonnet, and instead focus on writing hardware independent code for classifying our events that can be implemented on more powerful hardware should our system be deployed.