INTRODUCTION

Many aging people, either due to physical weakness or an onset of an illness, suffer from falls. Quick detection and intervention can result in a greater chance of recovery. We utilized Intel’s OpenVINO toolkit to process live video footage and obtain a decision on whether or not a fall has been detected.

To improve performance, we used Intel’s Neural Compute Stick v2 FPGA (Vision Processing Unit). This stick is designed to provide improved performance compared to purely CPU implementations on low-cost hardware for computer vision applications. One of the major advantages of this approach over wearable devices is that it is non-intrusive. None of the data is sent to the cloud, it’s all processed locally thereby respecting the user’s privacy.

AIM

We expand on the work done by Adrián Núñez-Marcos et al. [1] by developing a trained model to a low-cost edge device (Raspberry Pi 4). The trained model architecture is VGG-16 modified to accept optical flow stacks for generic feature extraction. The classifier is a fully-connected neural network fine-tuned for fall detection on embedded platform using depth maps and wireless accelerometer.

METHOD

Training

To train the model, we conduct transfer learning on an RGB I3D model that has been pre-trained on full ImageNet and Kinetics 400 datasets [5]. We do so by freezing all convolutional layers and removing the logits output layer. We replace this layer with a classification top with 2 dense layers and 3 dropout layers, with a softmax output layer. The datasets we use to train are the URFD dataset [2] and the multiple cameras fall dataset [3]. For preprocessing, we split the dataset into falls and no falls based on the corresponding labels. We load images, normalize and arrange them in arrays with a depth of 1. Those of the arrays differed from falls and no falls.

We faced a severe class imbalance for falls at a rate of 1:50. So, we took two approaches to help balance the dataset. The first was to undersample falls by converting image stacks from a video at a stride of 20, and oversample falls by converting them at a stride of 1. Despite these measures, we still faced a rate of 1:33. The next step was to apply class weights to the training function by modifying the loss function. Penalizing incorrect fall predictions and lessening the importance of correct non-fall predictions. Due to time constraints, we trained for two epochs, using Adam for optimization at a learning rate of 0.001.

Deployment

To deploy the app on the Pi we first created a Python program, grant #1725729, as well as the University of Illinois at Urbana-Champaign.

RESULTS

Model

At the end of two epochs, we achieved an accuracy of 97.7% on our test set. However, this result would lend itself to the accuracy paradox, as due to our class imbalance, outputting constant no-fall predictions would still yield an accuracy of about 95%. Examining further on our test set, our model predicts non-falls at an accuracy of 97%. On purely falls, we achieve an accuracy of 65.4%.

While less than ideal, this does not render the model ineffective. An average fall lasts about 1 second, and at a rate of 25 FPS, the model will infer 25 times in the “fall” region. Given this same opportunity with changing frame information, the model has an effectively better chance to catch a fall. As the model was made under a time constraint, there are many avenues for improvement. Hyperparameter optimization and data bootstrapping methods could be the key to achieving the maximum potential of the network. Additionally, incorporating additional fall datasets would be critical for increasing the diversity and balance of the dataset.

Timing Info

Surprisingly inferencing took place far more quickly than anticipated. Ideal (20 frame @ 43FPS) Image Capture and Processing: 40 ms Inference: 800 ms Observed: Image Capture and Processing: ~32 ms = 31.25 fps, 74 ms max (Reinitializing the buffer) 25 ms first run inference: ~74 ms = 41.67 fps Delay added later to avoid data races and maintains 25 FPS as the model expects.

CONCLUSIONS

This project shows that Deep Neural Network based inferencing can be effectively performed on low cost, low powered devices such as the Raspberry Pi 4 along with an Intel NCS Vision Processing Unit.

ACKNOWLEDGEMENTS

This work utilizes resources supported by the National Science Foundation’s Major Research Instrumentation program, grant #1752729, as well as the University of Illinois at Urbana-Champaign.

REFERENCES