Qualcomm Developer Project ObjectDetectionSample

Project Submission

|  |  |  |
| --- | --- | --- |
| **Attribution:** |  | |
| **Email address** | [sunzhen@thundersoft.com](mailto:sunzhen@thundersoft.com)  [yangrong0925@thundersoft.com](mailto:yangrong0925@thundersoft.com)  [wangjie0508@thundersoft.com](mailto:wangjie0508@thundersoft.com)  [Kouzw0723@thundersoft.com](mailto:Kouzw0723@thundersoft.com)  [yansh0810@thundersoft.com](mailto:yansh0810@thundersoft.com) | |
| **Project Title**\* | **ObjectDetectionSample** | |
| **Images**  *Upload up to 5 images of your project*  *Please submit/send the original JPEG/PNG files for all images included in the document* | **Turbox C610.png**  turbox30-c610 **[alt tag: “ObjectDetectionSample using** t**he Turbox C610 development board.**”] **Type-C.png**   |  | | --- | | typc |   [alt tag: “**using the type-c line to develop on Turbox C610 development board.** ”]  Orange:  orange  Result:  result | |
| **Description**\*  *High level description of the project* ***(75 words or less)*** | Using Turbox C610 with Linux Embedded OS ，The project consists of two parts, first using SNPE SDK and tensorflow trans the object detection network model into DLC file which can be loaded into the SNPE runtime, second making a secondary porting of SNPE SDK for testing the model’s results. | |
| **Objective**   * *What inspired you to create this project?* * *What is your desired outcome?* | The project includes the full use of snpe SDK ，it can help developers quickly trans the network to dlc file and load models into the GPU/DSP/CPU of Qualcomm chips through SNPE SDK. | |
| **Materials Required / Parts List / Tools** | Part Name | Link to purchase |
| Turbox C610 | https://www.thundercomm.com/app\_en/product/1593776185472315?index=0&categoryId=categorynull |
| Neural Processing SDK for AI | <https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk> |
| TensorFlow | https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.0.0-cp27-none-linux\_x86\_64.whl |
| Type-c line | https://detail.tmall.com/item.htm?id=44425281296&ali\_refid=a3\_430582\_1006:1103572855:N:8BFxSxK119dzkfQCc2yGI2us815vvcUHETWnj5g1swo=:6399b40850a40201c56536531a885bcf&ali\_trackid=1\_6399b40850a40201c56536531a885bcf&spm=a230r.1.14.11 |
|  |  |
|  |  |
|  |  |
| **Source Code / Source Examples / Application Executable**  *Link to open source / shareable code repository* | Description | Link |
| [Source Code](https://github.com/canyudeguang/Home_Automation) | [https://github.com/ThunderSoft-XA/](https://github.com/ThunderSoft-XA/Neural-Processing-SDK-for-AI) |
|  |  |
|  |  |
|  |  |
| **Additional Resources**  *List related links or resources such as websites, videos, presentations, or other materials* | Resource Title | Link or File Name (and provide file) |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Build / Assembly Instructions** | Sample outline:  Below are the items used in this project.   |  | | --- | | **Turboxc610**  **turbox30-c610** |   [alt tag: “**Parts used for developing.** ”]   1. Turbox C610 with Linux Embedded OS 2. PC with Ubuntu 16.04. 3. Type-C data cable. 4. Neural Process SDK for AI. 5. TensorFlow.   Model conversion instruction:  According to document《model\_conversion\_instruction.md》.  Project compilation instructions:   1. Purchase the equipment and configure the cross compilation environment according to the Technical Documents . 2. Configure algorithm model conversion environment According to 《model\_conversion\_instruction.md》 3. Software Build Instructions ( Reference yocto compilation method ) 4. Edit BB file in a directory of pokey 5. Bitbake objsample. | |
|  | Sample outline:   1. How does it work?   By Using the Open source Inception\_v3 network , the project can detect 1001 kinds of items on the Turbox C610 .  Detect method:  std::vector<int> ObjectDetection::doDetect(cv::Mat img) {  std::vector<int> result;  if (NULL == snpe) {  //TS\_LOGE("can not init err!");  return result;  }  cv::Mat input;  long start\_pre = getCurrentTime\_ms();  // cv resize  cv::Mat resize\_mat;  cv::resize(img, resize\_mat, cv::Size(299, 299));  cv::cvtColor(resize\_mat, input, CV\_BGR2RGB);  cv::Mat input\_norm(MODEL\_INPUT\_H, MODEL\_INPUT\_H, CV\_32FC3, inTensor.get()->begin().dataPointer());  input.convertTo(input, CV\_32F);  cv::normalize(input, input\_norm, -1.0f, 1.0f, cv::NORM\_MINMAX);  long diff\_pre = getCurrentTime\_ms() - start\_pre;  zdl::DlSystem::ITensor\* outputTensor = nullptr;  long start = getCurrentTime\_ms();  bool ret = snpe->execute(inMap, outMap);  long diff = getCurrentTime\_ms() - start;  if (!ret) {  const char\* const err = zdl::DlSystem::getLastErrorString();  std::cout<<"!!!!!!ERROR code:"<<err<<std::endl;  return result;  }  zdl::DlSystem::StringList tensorNames = outMap.getTensorNames();  for( auto& name: tensorNames ){  std::cout<< "tensor name: "<<name<<std::endl;  outputTensor = outMap.getTensor(name);  }  zdl::DlSystem::TensorShape shape = outputTensor->getShape();  size\_t rank = shape.rank();  int input\_size = 1;  for (size\_t i=0; i<rank; i++) {  input\_size \*= shape[i];  std::cout<<"output shape"<<i<<":"<<shape[i]<<std::endl;  }  const float\* final\_result = &\*(outputTensor->cbegin());  for(int i =0; i<shape[1]; i++){  float confidence = \*final\_result++;  if(confidence>mConfidenceThreshold){  result.push\_back(i);  std::cout<<"i: "<<i<<std::endl;  std::cout<<"confidence: "<< confidence<<std::endl;  }  }  long start\_post = getCurrentTime\_ms();  outMap.clear();  long diff\_post = getCurrentTime\_ms() - start\_post;  long diff\_sum = getCurrentTime\_ms() - start\_pre;  std::cout<<"sum cost time="<<diff\_sum<<std::endl;  return result;  }  Main test:  int main(int argc, char\*\* argv) {  std::string\* labels = new std::string[1001];  std::ifstream in("../test/imagenet\_slim\_labels.txt");  std::string line;  int count=0;  while (getline(in,line)){  labels[count] = line;  count++;  }  ObjectDetection\* detector = new ObjectDetection();  detector->init(GPU);  detector->setConfidence(0.1f);  cv::Mat img = cv::imread("../test/orange.jpeg");  if(img.empty()){  std::cout<<"empty"<<std::endl;  }  //test  std::vector<int> result = detector->doDetect(img);  std::cout<<"data size: "<< result.size()<<std::endl;  if(result.size()>0){  for(int i =0;i< result.size();i++){  std::cout<<"name : "<< labels[result[i]]<<std::endl;  }  }  return 0;  }  Items :  cauliflower  zucchini  spaghetti squash  acorn squash  butternut squash  cucumber  artichoke  bell pepper  cardoon  mushroom  Granny Smith  strawberry  orange  lemon  fig  pineapple  banana  jackfruit  custard apple  pomegranate  hay  carbonara  chocolate sauce  dough  meat loaf  And so on.... | |
| **Usage Instructions** | Sample outline:  Demo running steps：   1. push the project that builded to the Turbox C610 devices   Adb push ObjectDetectionSample /data/   1. Excute the bin   Cp ObjectDetectionSample/lib/aarch/lib\* /usr/lib  Cd ObjectDetectionSample/bin  ./object\_test  3.result：  orange | |
| **Contributor(s) Info**  *Feel free to include headshots!* | Name | Title  Company |
|  |  |
|  |  |
|  |  |

––– Continued on next page –––

Filters and Tags for QDN projects page

|  |  |  |
| --- | --- | --- |
| **Platform/Hardware** | CSR 101x/102x Bluetooth  DragonBoard 410c  mangOH Red/Yellow | MDM920x LTE for IoT  QCA-402x WiFi/BLE/Zigbee     Qualcomm Robotics RBx Dev Kit |
| **Software Tools** | 3D Audio Plugin for Unity  Adreno GPU SDK  Hexagon DSP SDK | √ Neural Processing SDK for AI  Snapdragon Profiler |
| **Operating System** | Android  √ Linux  ThreadX RTOS | Ubuntu Core  Windows 10 IoT Core |
| **Cloud Services/Platform** | Sierra Wireless AirVantage  Gizwits Cloud Platform  AT&T M2X  IBM Bluemix | IBM Watson IoT  Microsoft Azure IoT  Amazon AWS IoT |
| **Skill Level Required** | √ Advanced  Beginner  Intermediate |  |
| **Areas of Focus** | 3D Printing & Modeling  Alexa Voice Service  Artificial Intelligence  Bluetooth  Computer Vision  Digital Signage  √ Education  √ Embedded  Gaming | Healthcare  √ IoT  √ Robotics  √ Security  Sensors  Smart Cities  √ Smart Home  √ Toys |

*By submitting your content (“Submission”), you are granting Qualcomm a royalty-free, perpetual, non-exclusive, unrestricted, worldwide license to: (a) post, use, copy, sublicense, adapt, transmit, publicly perform or display any such Submission, (b) use, reproduce, modify, adapt, publish, translate, create derivative works from, distribute, perform, play, host, communicate, make available and publish your Submission without restriction and (c) sublicense to third parties the unrestricted right to exercise any of the foregoing rights granted with respect to the Submission. The foregoing grants shall include the right to exploit any ideas, concepts, intellectual property, or proprietary rights in such Submission, including but not limited to rights under copyright, trademark, servicemark or patent laws under any relevant jurisdiction without Qualcomm owing any monies to you whatsoever. You represent and warrant that you own all right, title and interest in and to the Submission, or you have been granted sufficient rights in and to the Submission allowing the foregoing use of such Submission.*