نام : پیام امانت

استاد : دکتر محمد زارع

موضوع : سیستم هوشمند نظارت بر کشاورزی

درس : رباتیک

سیستم هوشمند نظارت بر کشاورزی

سیستم نظارت بر کشاورزی با هوش مصنوعی برای اتومات سازی فرایند ها

ظرفیت پیشگیری و تشخیص آفات و بیماری ها

وسایل مورد نیاز این پروژه :

قطعات سخت افزاری

1-AMD Kria KV260 Vision AI Starter Kit (x1)

2-AMD PYNQ Grove Adapter(x1)

3-Seeed Studio Grove- Relay(x1)

4-Seeed Studio Grove-I2c High accuracy Temp&Humi sensor (SHT35) (x1)

5-Seeed Studio Grove-Water Sensor (x1)

6-Webcam ELIPENICO(x1)

برنامه های نرم اقزاری و سرویس های انلاین

1-Google Cloud IOT Core

2-AMD Vitis Unified Software Platform

3-Jupyter Notebook

4-AMD PetaLinux

5-Tensorflow

AMD Kria چیست ؟

توسعه پلتفرم برای Kria K26 SOMs که در برنامه های حرفه ای دید استفاده می شود که نیاز به داشتن داتش زیاد در طراحی سخت افزاری ندارد.منبع تقذیه به صورت جداگانه به فروش می رسد.

AMD PYNQ چیست ؟

یک اداپتور با 4 کانکنتور که دارای استادارد Grove می باشد.

Seeed Studio Grove چیست؟

بخشی از سیستم Grove هست که از ابزارهای تعدیل شده و اماده برای استغاده می باشد.سنسورها از یک پایه و ماؤول های مختلف برای اتصالات استاندارد استفاد می شود.محافظه پایه امکان اتصال اسان هر ورودی یا خروجی ریزپردازنده از ماژول های grove را فراهم میکند .

Seeed Studio Grove-Water Sensor چیست؟

ماژول سنسور اب یم بخشی از Grove هست که با اندازه گیری رسانایی نشان می دهد که سنسور خشک یا مرطوب یا کاملا در اب غوطه ور هست.ردیابی سنسور دارای یک مقاومت کششی ضعیف 1m اهم هست.

Webcam ELIPENICO چیست ؟

یک وبکم که دارای 1080 پیکسل رزولوشن و 264 فشرده سازی ویدیویی و میدان فوق عریض 110 درجه می باشد.

چیست؟-Google Cloud IOT Core

یک پلتفرم مدیریت شده برای که به دستگاهای متصل اجازه می دهد به راحتی و ایمن با برنامه های ابری و سایر دستگاه ها در تعامل باشد.

AMD Vitis Unified Software Platform چیست؟

یک محیط توسعه برای توسعه طرح هایی که شامل FPGA، زیرسیستم های پردازنده Arm® و موتورهای هوش مصنوعی می شود.

چیست ؟Jupyter Notebook

برای ایجاد اسناد نوت بوک تعاملی که می تواند حاوی کد زنده، معادلات، تجسم ها، رسانه ها و سایر خروجی های محاسباتی باشد.

AMD PetaLinux چیست ؟

ابزارهایی برای سفارش سازی، ساخت و استقرار راه حل های لینوکس جاسازی شده در سیستم های پردازش جاسازی شده در AMD

Tensorflow چیست ؟

یک کتابخانه نرم افزاری متن باز برای یادگیری ماشین که انواع مختلف مفاهیم مفهومی و زبانی هست.

مقدمه :

اثرات تغییرات آب و هوایی به گسترش آفات مخرب کمک میکند و میزانش را افزایش می دهد و بقای مهم ترین گیاهان و محصولات زراعی را از نقطه نظر اقتصادی تهدید می کند. وضعیتی که تهدیدی فزاینده برای امنیت غذایی و محیط زیست است.

موضوع:

تغییر اقلیم باعث تغییر دما، رطوبت و گازهای جو می شود و به خصوص انباشت GHG که می تواند به رشد قارچ ها و حشرات کمک کند و برهمکنش مثلث بیماری (میزبان - پاتوژن - محیط) را تغییر دهد و در نتیجه تولید آنها را کاهش دهد. بررسی های مختلف نوسانات بروز آفات را در مناطق معتدل و گرمسیری نشان داده است که با رویدادهای دوره خشک و ترکیبی از خشکسالی و رطوبت نسبی بالا همراه است.سازمان خواربار و کشاورزی ملل متحد (فائو) تخمین می زند که آفات هر ساله تا 40 درصد از تولید جهانی محصولات را از بین می برند، در حالی که بیماری های گیاهی سالانه بیش از 220 هزار دلارو حشرات مهاجم حداقل 70000 میلیون دلار برای اقتصاد جهان هزینه دارد. منبع (UN)

مدیرکل سازمان در ارائه این مطالعه گفت: «نتیجه‌گیری اصلی این ارزیابی باید به همه ما هشدار دهد که چگونه تغییرات آب و هوایی می‌تواند بر میزان سرایت، گسترش و شدت آفات در سراسر جهان تأثیر بگذارد.

فائو:سازمان غذا و کشاورزی ملل متحد از سازمان های بین المللی است که در زمینه ی توسعه ی کشاورزی فعالیت دارد.

راه حل :

طراحی و ساخت یک سیستم نظارت بر کشاورزی با هوش مصنوعی با قابلیت شناسایی آفات و آفات در محصولات به گونه‌ای که به سیستم اجازه می‌دهد علف‌کش و مقدار مناسب را برای ریشه‌کن کردن آن به‌طور مستقل و فوری انتخاب کند. به همین ترتیب، قادر به ثبت و تجزیه و تحلیل متغیرهای کشاورزی اقلیمی که امکان تجزیه و تحلیل و تصمیم گیری در مورد محصولات را فراهم می کند تا بتوان از تلفات ناشی از تغییرات آب و هوایی جلوگیری کرد.

قدم های اولیه:

پیکربندی Petalinux

1: تنظیم تصویر کارت SD

ما باید در برنامه توسعه Xilinx ثبت نام کنیم که قادر باشیم تصاویرPetalinux 2021 را دانلود کنیم.

هنگامی که تصویر در Balena Etcher بارگذاری شد، آن را فلش می کنیم.

Balena Etcher : یک ابزار رایگان و متن باز است که این قابلیت را به ما میدهد که فایل های تصویری را بنویسیم و بخوانیم و همچنین یک رسانه ذخیره سازی کارت های اس دی می باشد.

2-ارتباط تمام وسایل

ما مطمئن می شویم که همه کابل ها به خوبی به هم وصل شده اند.

3-راه اندازی

ما اتصال را از طریق پورت COM با کارت پیکربندی می کنیم. بنابراین ما به استفاده از PuTTy ادامه می دهیم و مطمئن می شویم که پارامترهای زیر را پیکربندی کرده ایم :

Port Com: این پورت به معنی ارتباطات است و برای اتصال انواع ورودی و خروجی ها استفاده می شود.

Putty: یک سرویس گیرنده تلنت برای ویندوز است که این امکان را میدهد که سیستم های محلی از راه دور به سرور ابری به صورت ایمن دسترسی پیدا کنند.

* Baud rate = 115200
* Data bits = 8
* Stop bit = 1
* Flow control = None
* Parity = None

Baud rate : برای توصیف حداکثر نرخ نوسان یک سیگنال الکترونیکی استفاده می شود.  به عنوان مثال، اگر یک سیگنال 1200 بار در یک ثانیه تغییر کند (یا می تواند تغییر کند)، با 1200 باود اندازه گیری می شود.

Flow control :  در ارتباطات داده‌ای، کنترل جریان یک فرآیند مدیریت نرخ انتقال بین دو گره (دو عضو در یک شبکه) برای جلوگیری از ارسال از طرف فرستنده (سریع) به دریافت‌کننده (کند) است.

Parity : در علم کامپیوتر Parity یا بیت افزونه یا بیت توازن تکنیکی است برای بررسی اینکه اطلاعات و داده ها در طول جابجایی از مکانی در حافظه بین کامپیوترها،  دچار تغییر و خطا شده است یا نه.

پیکربندی VITES AI

توجه به این نکته ضروری است که اگر سیستم عامل لینوکس ندارید، می توانید نصب زیر را در ماشین مجازی انجام دهید.

1. Docker را روی دستگاه نصب و راه اندازی کنید.

Docker : یک پلتفرم نرم افزاری برای ساخت اپلیکیشن‌های مبتنی بر Container است. محیط‌های اجرایی کوچک و سبک که به طور مشترک از هسته سیستم عامل استفاده می‌کنند اما در عین حال در یک محیط ایزوله و کاملا جدا از هم قرار دارند.

2. مخزن Vitis-AI را کلون کنید تا نمونه ها، کد مرجع و اسکریپت ها را به دست آورید.

git clone --recurse-submodules https://github.com/Xilinx/Vitis-AI   
  
cd Vitis-AI

3. آخرین نسخه Vitis AI Docker را با دستور زیر دانلود کنید. این کانتینر روی CPU اجرا می شود.

docker pull xilinx/vitis-ai-cpu:latest

4. برای اجرای docker، از دستور استفاده کنید:

./docker\_run.sh xilinx/vitis-ai-cpu:latest

اموزش دادن مدل :

برای آموزش مدل از TensorFlow برای ایجاد شبکه عصبی استفاده می کنیم. واضح است که برای استفاده از یک پردازنده گرافیکی رم 30 گیگابایتی باید طرح PRO Colab را خریداری کنید.

## گوگل کولب (Google Colab) چیست؟

Google Colab یا Colaboratory Pro یک [**سرویس ابری**](https://license-market.ir/category/cloud-services) ارائه شده از طرف گوگل است که به شما اجازه برنامه نویسی با زبان پایتون را داده و امکان نصب و کار با پکیج های مختلف زبان پایتون و فریم ورکهای مطرح یادگیری عمیق همانند Pytorch، Keras، Tensorflow و غیره را فراهم می آورد.

[**اکانت Colab Pro**](https://license-market.ir/product/Google-Colab) امکان نوشتن و اجرای کدهای پایتون دلخواه از طریق مرورگر را فراهم می کند و برای [**یادگیری ماشینی**](https://fa.wikipedia.org/wiki/%DB%8C%D8%A7%D8%AF%DA%AF%DB%8C%D8%B1%DB%8C_%D9%85%D8%A7%D8%B4%DB%8C%D9%86%DB%8C)، آموزش و تحلیل داده بسیار مفید و مناسب است و با سریع ترین GPU گوگل، کار را برای کاربران بسیار آسان می کند.

1ما کتابخانه ها را وارد کرده و دیتاست را دانلود می کنیم.

import tensorflow as tf  
import tensorflow\_datasets as tfds  
datos, metadatos = tfds.load('plant\_village', as\_supervised = True, with\_info = True)  
metadatos.features  
print(metadatos.features["label"].names)

2. تغییر اندازه تصاویر: از آنجایی که همه تصاویر دارای ابعاد متفاوتی هستند و می توانند با TensorFlow در تضاد باشند، به تغییر اندازه همه تصاویر می پردازیم:

import matplotlib.pyplot as plt  
import cv2  
plt.figure(figsize=(20,20))  
tamaño = 50  
for i, (imagen, etiqueta) in enumerate(datos['train'].take(25)):  
imagen = cv2.resize(imagen.numpy(), (tamaño, tamaño))  
plt.subplot(5, 5, i+1)  
plt.imshow(imagen)

3. دیتابیس را برای آموزش تقسیم کنید:

train\_data = []  
for i, (imagen, etiqueta) in enumerate(datos['train']):  
imagen = cv2.resize(imagen.numpy(), (tamaño, tamaño))  
imagen = imagen.reshape(tamaño, tamaño, 3)  
train\_data.append([imagen, etiqueta])  
  
#Prepare my variables X (inputs) and y (labels) separately  
X\_data = [] #imagenes de entrada (pixeles)  
y\_data = [] #etiquetas  
for imagen, etiqueta in train\_data:  
X\_data.append(imagen)  
y\_data.append(etiqueta)

4. Xdata و Ydata را به آرایه تبدیل کنید

import numpy as np  
X\_data = np.array(X\_data).astype(float) / 255  
  
y\_data = np.array(y\_data)

5. ساخت شبکه عصبی: لایه اولیه، لایه های پنهان و لایه های خروجی با تابع فعال سازی مربوطه خود

modeloCNN = tf.keras.models.Sequential([  
tf.keras.layers.Conv2D(32, (3,3), activation='relu', input\_shape=(50, 50, 3)),  
tf.keras.layers.MaxPooling2D(2, 2),  
tf.keras.layers.Conv2D(64, (3,3), activation='relu'),  
tf.keras.layers.MaxPooling2D(2, 2),  
tf.keras.layers.Conv2D(128, (3,3), activation='relu'),  
tf.keras.layers.MaxPooling2D(2, 2),  
tf.keras.layers.Flatten(),  
tf.keras.layers.Dense(100, activation='relu'),  
tf.keras.layers.Dense(38, activation='softmax')  
])

6. مدل را کامپایل می کنیم:

modeloCNN.compile(optimizer='adam',  
loss='sparse\_categorical\_crossentropy',  
metrics=['accuracy'])  
modeloCNN2.compile(optimizer='adam',  
loss='sparse\_categorical\_crossentropy',  
metrics=['accuracy'])

7. مدل را دوباره آموزش می دهیم:

from tensorflow.keras.callbacks import TensorBoard  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
datagen = ImageDataGenerator(  
rotation\_range=30,  
width\_shift\_range=0.2,  
height\_shift\_range=0.2,  
shear\_range=15,  
zoom\_range=[0.7, 1.4],  
horizontal\_flip=True,  
vertical\_flip=True  
)  
datagen.fit(X\_data)  
plt.figure(figsize=(20,8))  
for imagen, etiqueta in datagen.flow(X\_data, y\_data, batch\_size=10, shuffle=False):  
for i in range(10):  
plt.subplot(2, 5, i+1)  
plt.xticks([])  
plt.yticks([])  
plt.imshow(imagen[i].reshape(50, 50, 3))  
break  
  
from keras.models import Sequential  
from keras.layers import Convolution2D, MaxPooling2D, Dropout  
from keras.layers import Flatten, Dense  
from keras.layers import Conv2D, GlobalAveragePooling2D  
from keras.models import Sequential  
from keras.layers import Conv2D, MaxPooling2D  
from keras.layers import Activation, Dropout, Flatten, Dense  
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D  
from keras.layers import Dropout, Flatten, Dense  
from keras.models import Sequential

8. ما مدل های اولیه را ایجاد می کنیم:

modeloCNN\_AD = tf.keras.models.Sequential([  
tf.keras.layers.Conv2D(32, (3,3), activation='relu', input\_shape=(50, 50, 3)),  
tf.keras.layers.MaxPooling2D(2, 2),  
tf.keras.layers.Conv2D(64, (3,3), activation='relu'),  
tf.keras.layers.MaxPooling2D(2, 2),  
tf.keras.layers.Conv2D(128, (3,3), activation='relu'),  
tf.keras.layers.MaxPooling2D(2, 2),  
tf.keras.layers.Flatten(),  
tf.keras.layers.Dense(100, activation='relu'),  
tf.keras.layers.Dense(38, activation='softmax')  
])  
X\_train = X\_data[:40000]  
X\_valid = X\_data[40000:]  
y\_train = y\_data[:40000]  
y\_valid = y\_data[40000:]  
  
modeloCNN\_AD.compile(optimizer='adam',  
loss='sparse\_categorical\_crossentropy',  
metrics=['accuracy'])  
  
data\_gen\_train = datagen.flow(X\_train, y\_train, batch\_size=32)  
  
tensorboardCNN\_AD = TensorBoard(log\_dir='logs/cnn\_AD')  
modeloCNN\_AD.fit(  
data\_gen\_train,  
epochs=100, batch\_size=32,  
validation\_data=(X\_valid, y\_valid),  
steps\_per\_epoch=int(np.ceil(len(X\_train) / float(32))),  
validation\_steps=int(np.ceil(len(X\_valid) / float(32))),  
callbacks=[tensorboardCNN\_AD]  
)

10. ما مدل را دانلود می کنیم:

modeloCNN\_AD.save('my\_model-cnn-ad.h5')

کوانتیزاسیون مدل

فرایند دیجیتالی کردن یک سیگنال آنالوگ شامل گرد کردن مقادیری است که تقریبا برابر با مقدار آنالوگ هستند. با استفاده از [نمونه‌برداری](https://blog.faradars.org/%D9%86%D9%85%D9%88%D9%86%D9%87-%D8%A8%D8%B1%D8%AF%D8%A7%D8%B1%DB%8C-%D8%B3%DB%8C%DA%AF%D9%86%D8%A7%D9%84/) چند نقطه روی سیگنال آنالوگ اصلی انتخاب می‌شود و سپس این نقاط پس از [گرد کردن](https://blog.faradars.org/rounding-up-numbers-in-excel/) مقادیر به نزدیک‌ترین مقدار پایدار، به یکدیگر متصل می‌شوند. به این فرایند «کوانتیزاسیون» (Quantization) می‌گویند.

  1. مدل حاصل از آموزش و پایگاه داده را به کد کوانتیزاسیون ایجاد شده در Vitis AI وارد کنید.

2. مدل کوانتیزاسیون خود را تعریف می کنیم:

def quantize(train\_generator, model):  
   
 # run quantization  
 quantizer = vitis\_quantize.VitisQuantizer(model)  
 #quantizer = tfmot.quantization.keras.quantize\_model(model)  
 quantized\_model = quantizer.quantize\_model(calib\_dataset=train\_generator, calib\_batch\_size=10)

3. مدل کوانتیزاسیون را ذخیره می کنیم:

# save quantized model  
 quantized\_model.save('quantized\_model.h5')  
 return (quantized\_model)

تلفیق

1. یک فایل به نام arch.json ایجاد کنید که در آن تنظیمات DPU را اختصاص می دهیم

{  
 "fingerprint":"0x1000020F6014406"  
}

**DPU چیست؟**

سیستمی روی یک تراشه که ترکیبی ست از:

* پردازنده چند هسته ای استاندارد، با کارایی بالا ، قابل برنامه ریزی و با استاندارد صنعتی
* رابط شبکه با کارایی بالا
* موتورهای شتاب دهنده انعطاف پذیر و قابل برنامه ریزی

**CPU vs GPU vs DPU چه چیزی DPU را متفاوت می‌کند؟**

DPU کلاس جدیدی از پردازنده‌های قابل برنامه ریزی است که سه عنصر اصلی را با هم ترکیب می‌کند. DPU سیستمی است بر روی تراشه یا SoC که ترکیبی ست از:

* یک پردازنده مرکزی چند هسته ای با قابلیت کارایی بالا، قابل برنامه ریزی و قابل استفاده در صنعت، و معمولاً مبتنی بر معماری widely used Arm که با سایر اجزای SoC کاملاً سازگار است.
* یک رابط شبکه با کارایی بالا که قادر به تجزیه، پردازش و انتقال کارآمد داده ها با سرعت بالا
* مجموعه ای غنی از موتورهای شتاب دهنده انعطاف پذیر و قابل برنامه ریزی که باعث تخلیه بار و بهبود عملکرد برنامه ها برای هوش مصنوعی و یادگیری ماشین، امنیت، ارتباطات از راه دور و ذخیره سازی می‌شوند.

تمام این قابلیت های DPU برای فعال کردن یک سیستم محاسباتی ایزوله، bare-metal و بومی ابر cloud-native که نسل بعدی محاسبات در مقیاس ابری را تعریف می‌کند ، بسیار مهم است.

2. دستور source compile.sh را در جایی که محل فایل arch.json ظاهر می شود اعمال می کنیم.

compile() {  
 vai\_c\_tensorflow2 \  
 --model quantized\_model.h5 \  
 --arch $ARCH \  
 --output\_dir build/compiled\_$TARGET \  
 --net\_name customcnn  
}  
  
  
compile 2>&1 | tee build/logs/compile\_$TARGET.log

3. فایل های This.xmodel تولید شده توسط مدل ما از طریق اتصال SFTP روی برد آپلود می شوند.

sftp petalinux@ <ip adreess>

4- دستور زیر را اجرا می کنیم:

lcd ..  
put -r compiled\_$TARGET

5. در مرحله بعد، petalinux روی برد شروع می شود و فایل های زیر را ایجاد می کنیم:

aiiference.json

drawresult.json

preprocess.json

6. این سه فایل باید در دایرکتوری قرار داشته باشند:

sudo cp yolov2tiny/aiinference.json /opt/xilinx/share/ivas/smartcam/ssd/aiinference.json  
sudo cp yolov2tiny/preprocess.json /opt/xilinx/share/ivas/smartcam/ssd/preproces.json  
sudo cp yolov2tiny/drawresult.json /opt/xilinx/share/ivas/smartcam/ssd/drawresult.json

7. ما آزمایش می کنیم که پروژه ما با دوربین خوب کار می کند و مدل یادگیری ماشین ما کار می کند:

sudo xmutil unloadapp  
sudo xmutil loadapp kv260-smartcam  
sudo smartcam --usb 0 -W 1920 -H 1080 --target rtsp --aitask ssd

پیکربندی سنسورها با Pynq

برای این پروژه باید پورت Pmod را فعال و استفاده کنیم و سنسورها را با آداپتور Pynq Grove متصل کنیم.

در ابتدا تحلیل خواهیم کرد:

درجه حرارت

رطوبت

سطح آب

1. یک دایرکتوری ایجاد کنید که حاوی تمام فایل های مورد نیاز ما باشد:

project-spec/meta-user/recipes-apps

2. فایل a.bb ایجاد کنید:

> vim python3-pynq-temp&hum.bb

3. پکیج سنسور را مستقیماً از دایرکتوری ریشه نصب می کنیم.

SRC\_URI = "https://pynq.readthedocs.io/en/v2.0/\_modules/pynq/lib/pmod/pmod\_tmp2.html#:~:text=lib.pmod.pmod\_tmp2-,Edit%20on%20GitHub,-Note"   
SRC\_URI[md5sum] = "ac1bfe94a18301b26ae5110ea26ca596"  
SRC\_URI[sha256sum] = "f522c54c9418d1b1fdb6098cd7139439d47b041900000812c51200482d423460"   
SRCREV = "0e10a7ee06c3e7d873f4468e06e523e2d58d07f8"S = "${WORKDIR}/git"  
  
inherit xilinx-pynq setuptools3

4. Xilinx-pynqclass یک متغیر PYNQ\_NOTEBOOK\_DIR ایجاد می‌کند که در بسته‌بندی فرعی نوت‌بوک بسته‌بندی می‌شود، اما ما همچنان باید مطمئن شویم که محیط برای اجرای صحیح دستور درست است. در این مورد باید متغیرهای محیطی PYNQ\_JUPYTER\_NOTEBOOK را تنظیم کنیم. BOARD setup.py همچنین انتظار دارد فهرست نوت بوک وجود داشته باشد، بنابراین باید آن را ایجاد کنیم. برای این کار، می‌توانیم دستورالعمل‌هایی را برای مراحل مختلف کامپایل قرار دهیم.

do\_compile\_prepend() { export BOARD=KV260 export PYNQ\_JUPYTER\_NOTEBOOKS=${D}${PYNQ\_NOTEBOOK\_DIR}}   
do\_install\_prepend() { export BOARD=KV260 export PYNQ\_JUPYTER\_NOTEBOOKS=${D}${PYNQ\_NOTEBOOK\_DIR} install -d ${PYNQ\_JUPYTER\_NOTEBOOKS}}   
do\_configure\_prepend() { export BOARD=KV260 export PYNQ\_JUPYTER\_NOTEBOOKS=${D}${PYNQ\_NOTEBOOK\_DIR} install -d ${PYNQ\_JUPYTER\_NOTEBOOKS}}

5. باید مستقل ها را تعریف کنیم.

RDEPENDS\_${PN} += "\   
python3-pynq \   
python3-pillow \   
pynq-overlay \   
libstdc++ \ "   
  
RDEPENDS\_${PN}-notebooks += "\   
python3-jupyter \ "

6. اجرا کنید:

> petalinux-build -c python3-pynq-temp&hum

7. ممکن است یک خطای سازگاری دریافت کنیم که باید با ایجاد یک وصله که رشته قالب را با یک اضافه ساده جایگزین می کند، آن را برطرف کنیم.

8. باید پچ را در یک زیر پوشه قرار دهیم تا مشخص شود.

> mkdir python3-pynq-temp&hum   
> cp $patch\_file python3-pynq-temp¬hum/build-fixes.patch

لازم بود با یک صفحه به ما مشاوره بدهید.

9. حال باید دستور را اجرا کنیم تا Petalinux بسته های نصب شده جدید را شناسایی کند.

> cd ../../   
> vim conf/user-rootfsconfig  
CONFIG\_python3-pynq-temp¬hum   
CONFIG\_python3-pynq-temp¬hum-notebooks  
  
> petalinux-config -c rootfs

10. اکنون می توانیم تصویر کامل را ایجاد کنیم:

> petalinux-build  
> petalinux-package --boot --u-boot --atf --pmufw

11. ما می توانیم برد را آزمایش کنیم و آدرس ترمینال را به ما می دهد تا اکنون مستقیماً در Jupyter بوت شود.

12. ما کد سنسورهای خود را اجرا می کنیم، آنها را می توان در انتهای این پروژه پیدا کرد.

پیکربندی Google Cloud IoT

برای تجسم محصولات خود در زمان واقعی و به دست آوردن نمودارهای سوابق حسگر، Google Cloud را با Jupyter Notebook پیکربندی کردیم.

پایگاه داده sheet برای مدل CNN

از بیش از 5000 تصویر مرجع از بیش از 20 نوع مختلف محصول استفاده می کند.پیشرفت‌های پس از مسابقه: امیدواریم بتوانیم سیستم‌های پمپ الکتریکی را برای سمپاشی علف‌کش‌ها، آب برای آبیاری محصولات و توسعه یک برنامه تلفن همراه ادغام شده با Google Cloud یکپارچه کنیم تا مزارع بتوانند خودکار شوند. به همین ترتیب، ما می‌خواهیم همراه با Xilinx بتوانیم توسط Pynq حسگرهای بسیار بیشتری را با مقاومت بیشتر در برابر شرایط شدید برای نظارت لحظه‌ای و دائمی متغیرهای جدید مانند سطوح فسفر، پتاسیم، وجود گازها، پیاده‌سازی کنیم. دوربین های مادون قرمز در سیستم ما نظارت بر اقلیم کشاورزی. با تشکر فراوان از کل تیم Xilinx به خاطر فرصتی که برای توسعه پروژه‌ها با محصولات پیشرفته به دست آوردند و همیشه از ما پشتیبانی می‌کنند تا توسعه این پروژه محقق شود.

Link to the article : <https://www.hackster.io/gabogiraldo/ia-agricultural-monitoring-system-cda68c>