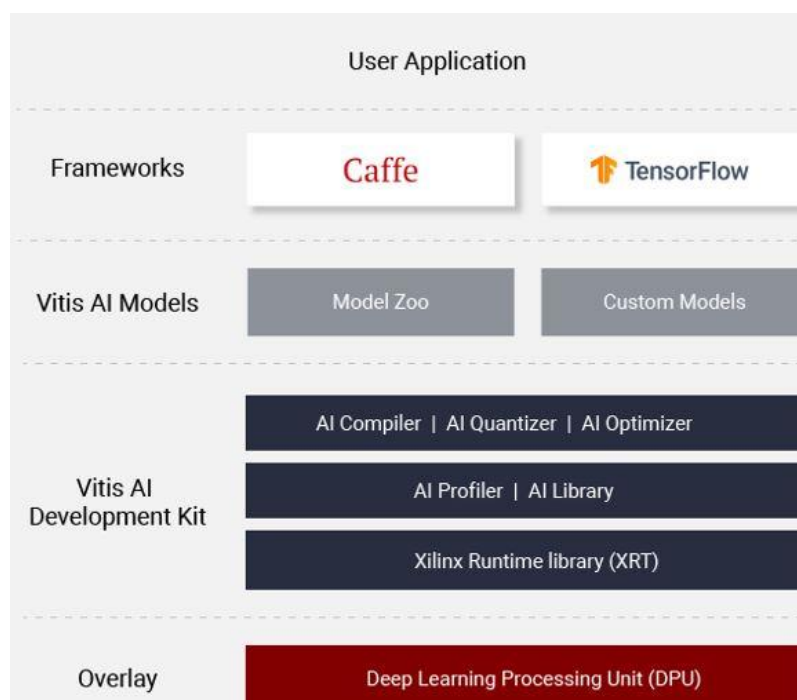


Guide Book

Deep Neural Network Inference on FPGA using TensorFlow



מבצעים : אלון נמירובסקי ועמית שטאובר
מנחים : אינה ריבקין, עוז שמואלי

אביב תשפ"א / אפריל 2021

תוכן עניינים

3.....	מבוא
4.....	הקמת עמדה
8.....	צריבת SD-Card
9.....	Setup של כרטיס FPGA
13.....	הקמת רשת, קומפילציה והרצת הרשת: Partial Flow For Resnet-50
13.....	• שלב הקוונטיזציה
15.....	• שלב הקומפילציה
18.....	הקמת רשת, קומפילציה, אופטימיזציה וסיווג: Full Flow For Mnist
19.....	• שלב הקוונטיזציה
19.....	• שלב הקומפילציה
22.....	• שלב הסיווג/הרצה על הboard
23.....	• אופטימיזציה וביצועים
26.....	טיפים, בעיות ופתרון בעיות
27.....	תובנות כלליות

רקע כללי

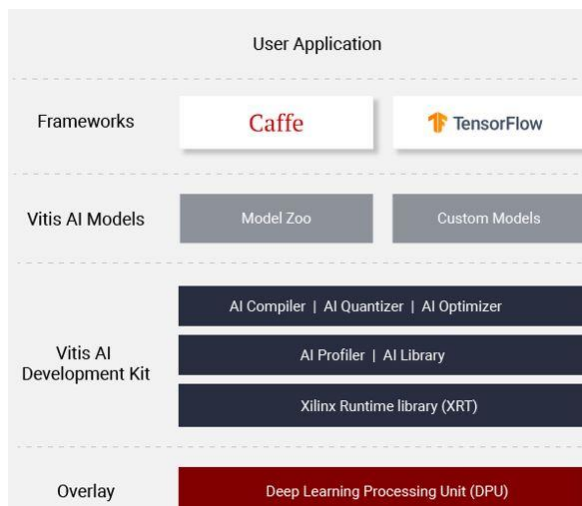
- תחום המערכות הלומדות המבוסס על רשתות נוירונים צובר תאוצה בשנים האחרונות, בעקבות ההצלחה של מערכות אלו בפתרון בעיות מורכבות.
- יישום של רשתות מתבצע ב-2 שלבים – שלב האימון ושלב הסיווג.
- שלב האימון הוא שלב מקדים המתבצע פעם אחת לכל רשת נתונה ולרוב מבוצע על מערכת בעלת עוצמת חישוב גבוהה כגון GPU. כיוון ששלב זה מבוצע פעם אחת לכל מערכת אין חשיבות עקרונית לזמן וכוח חישוב ששלב זה דורש.
- לעומת זאת בהרבה אפליקציות כגון רובוטים, מל"טים, רשתות חיישנים, טלפונים סלולאריים, רכבים אוטונומיים וכו' שלב הסיווג הוא שלב המתבצע בשטח ע"י רכיבים המוגבלים בכוח החישוב, בהספק, ולעתים גם יש דרישה לעבודה בזמן אמת ולמחיר זול.
- כיוון שאחד המאפיינים הבולטים של שלב הסיווג ברשתות נוירונים הוא מקביליות, מימוש שלב זה דווקא על חומרה כגון FPGA מאפשר להגיע לביצועים גבוהים מאוד תוך שמירה על צריכת הספק נמוכה ומחיר זול. כמו-כן, מימוש החומרה על FPGA נותן גמישות ויכולת התאמה של הרשתות עבור בעיות שונות.
- כדי לקדם מימוש רשתות נוירונים בחומרה על רכיבי FPGA יצרני רכיבים אלו פתחו סט כלים חדש שמאפשר מימוש שלב הסיווג ואופטימיזציה של רשת החל משלב ההגדרה עד שלב המימוש בפועל, תוך שימוש בכלים סטנדרטיים.

מטרת הפרויקט

לחקור ולהעריך סט כלים חדש **Vitis AI** עבור DNN inference (סיווג ע"י רשת נוירונים) על רכיבי FPGA של חברת Xilinx.

בניית הרשת תעשה באמצעות TensorFlow

דרישות הפרויקט:



1. הקמת המערכת על מחשב ייעודי.
2. בחירה והרצה של מודל שאומן מראש תוך שימוש בכל כלי התוכנה של חברת Xilinx.
3. מימוש סביבת הבדיקה.
4. ביצוע שינויים והתאמות במודלים במטרה לבדוק את גמישותם ויכולתם של הכלים.

לדוגמא שינוי מבנה הרשת (model zoo vs Custom), טיפוס המשתנים (FP vs Fixed point) וכו'.

1. השוואה בין המימושים השונים תוך בדיקה של דיוק, throughput, גודל ואתגרים במימוש ושימוש בכלים.
2. הסקת מסקנות ודיווח על מגבלות, יכולות ובעיות של הכלים השונים בהם נעשה שימוש.

הקמת עמדה

חברת Xilinx מספקת לנו תשתית שלמה להרצת מערך Vitis-AI הדרוש להרצת מערכות בינה מלאכותית על הלוח ZCU104. נתאר כאן את תהליך הקמת התשתית על גבי המחשב האישי הכולל התקנת Virtual Machine ותוכנות שונות.

סדר פעולות ראשוני:

הקמת VM על גבי המחשב האישי:

הורדת קובץ ubuntu-18.04-desktop-amd64.iso ויצירת VM באמצעות VMware.

פירוט נוסף – חלק מהשלבים הראשוניים בוצעו באמצעות VirtualBox ולא שונו לאחר שביצענו המרה לVirtualBox. **VirtualBox אין לנסות לעבוד עם VirtualBox כי אנחנו קיבלנו הודעת שגיאה שיש פקודות שלא נתמכות בVirtualBox.** התוכנה בה ביצענו שימוש: VMware player workstation.

הקצנו 4GB ראם לתוכנה ו-350GB להארד דיסק, 4 cpu.

התקנת דוקר:

לפי המדריך [Program/References/PDFs_Archive/1](#)

```
E: Could not get lock /var/lib/dpkg/lock - open (11: Resource temporarily unavailable)
E: Unable to lock the administration directory (/var/lib/dpkg/), is another process using it?
projecta@projecta-VirtualBox:~$ sudo rm /var/lib/dpkg/lock
projecta@projecta-VirtualBox:~$ sudo apt-get install \apt-transport-https \ca-c
certificates \curl \gnupg-agent \software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  dirngr glr1.2-snapd-1 gnupg gnupg-l10n gnupg-utils gpg gpg-agent
  gpg-wks-client gpg-wks-server gpgconf gpgsm gpgv libcurl4 libgnapd-glib1
  libssl1.1 python3-dateutil python3-software-properties
  software-properties-gtk
Suggested packages:
```

הערה: ההרצה של GPU העלתה שגיאה שכרגע נראית לא רלוונטית.

גרסת הדוקר:

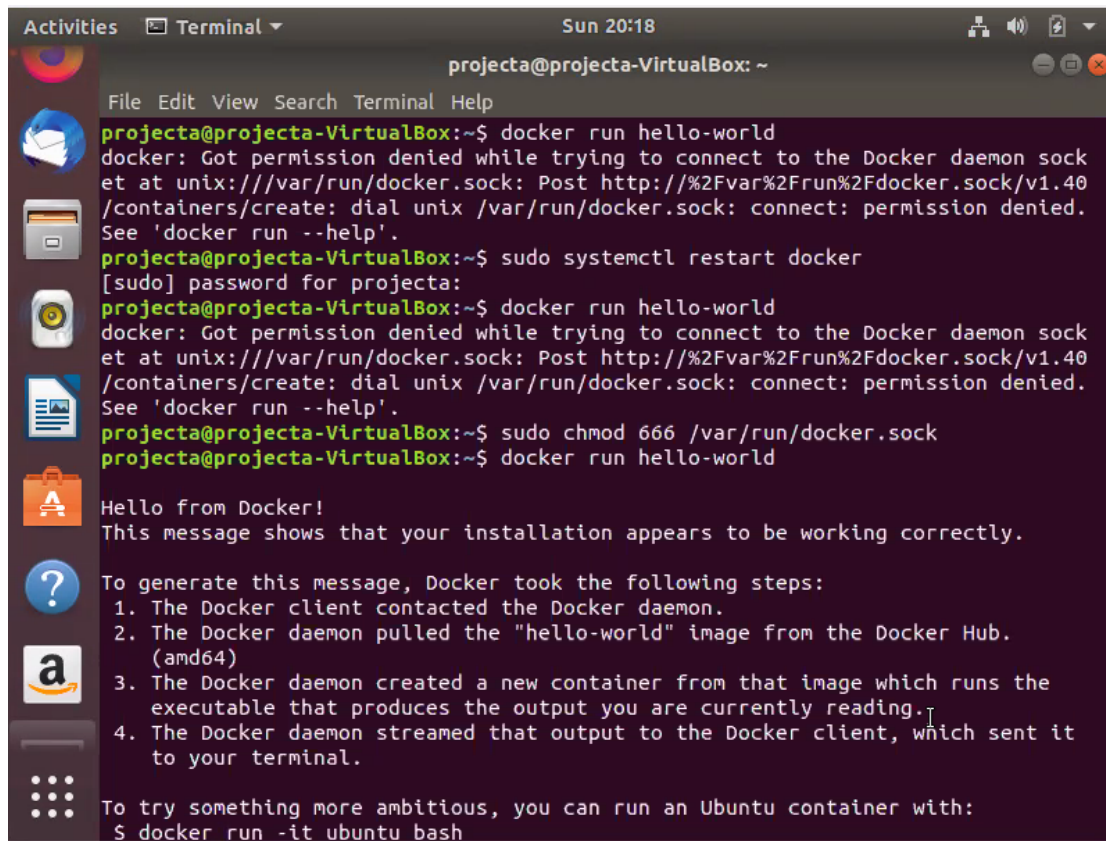
```
Setting up docker-ce (5:19.03.13-3-0-ubuntu-bionic) ...
```

פקודות להרצת docker:

```
./docker_run.sh xilinx/vitis-ai-cpu: latest
```

```
conda activate vitis-ai-tensorflow
```

תקלה שנתקלנו בה (ופתרון כמפורט בתמונה):



```
projecta@projecta-VirtualBox: ~
File Edit View Search Terminal Help
projecta@projecta-VirtualBox:~$ docker run hello-world
docker: Got permission denied while trying to connect to the Docker daemon sock
et at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.40
/containers/create: dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
projecta@projecta-VirtualBox:~$ sudo systemctl restart docker
[sudo] password for projecta:
projecta@projecta-VirtualBox:~$ docker run hello-world
docker: Got permission denied while trying to connect to the Docker daemon sock
et at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.40
/containers/create: dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
projecta@projecta-VirtualBox:~$ sudo chmod 666 /var/run/docker.sock
projecta@projecta-VirtualBox:~$ docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash
```

- Four steps to set up and verify the Vitis AI environment on a host machine
 - Step 1: Install the Docker engine
 - Step 2: Ensure that the Linux user is in the group Docker
 - Step 3: Clone the Vitis AI repository
 - Step 4: Run the Docker container

קישורים חשובים:

Program/References/PDFs_Archive/2

Program/References/PDFs_Archive/3

Program/References/PDFs_Archive/4

התקנת VART:

Program/References/PDFs_Archive/5

הודעה לאחר סיום ההתקנה :

.SDK has been successfully set up and is ready to be used

Each time you wish to use the SDK in a new shell session, you need to source the environment setup script e.g.

```
/ . $ home/projecta/petalinux_sdk/environment-setup-aarch64-xilinx-linux
```

: pathn

```
/~petalinux_sdk
```

הורדת qBittorrent :

```
sudo apt install qbittorrent
```

[/https://itsfoss.com/best-torrent-ubuntu](https://itsfoss.com/best-torrent-ubuntu)

Petalinux התקנת

כחלק מהפרויקט בדקנו היתכנות של קימפול קוד ה RTL של רכיב ה DPU וצריבתו על ה ZCU-104. נעזרנו במדריך הבא :

Program/References/PDFs_Archive/17

הערה: יש לשים לב שבעת ביצוע השלבים, אתם פועלים בהתאם לדרישות המקדימות בתחילת המדריך (גרסת petalinux, גרסת מכשיר וכו').

ראשית יש לבצע קימפול של קוד ה design של חומרת ה-DPU בעזרת כלי Xilinx.

בהמשך, מורידים מפה את שני הקבצים :

<https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/embedded-design-tools.html>

 **PetaLinux 2020.2 Installer (TAR/GZIP - 1.85 GB)**

MD5 SUM Value : 687b018f7502a4258bd633dc483bde79

 **ZCU104 BSP (BSP - 1.35 GB)**

MD5 SUM Value : d5d1b32de55f9d67f59e9d5593238884

יש לפעול לפי הוראות ההדרכה. נציין כי בכמה מקרים ביצענו התאמות. נפרט :

פקודות בהן נעזרנו (בוצעו התאמות לכרטיס שלנו בהתאם לסוגו) :

```
petalinux-config --get-hw-description=~/.Vitis-AI/BootExp/VivadoOutput/prj --silentconfig
```

```
petalinux-build
```

נדרשנו לשנות את הקונפיגורציה בקובץ הבא :

project-spec/meta-user/conf/petalinuxbsp.conf

לערכים הבאים :

```
#User Configuration

#OE_TERMINAL = "tmux"

IMAGE_BOOT_FILES_zynqmp = "BOOT.BIN boot.scr Image rootfs.cpio.gz.u-boot"
IMAGE_INSTALL_remove = "gststreamer-vcu-examples"
CONFIG_PRE_MIRROR_URL="http://petalinux.xilinx.com/sswreleases/rel-v2020/downloads"
CONFIG_YOCTO_NETWORK_SSTATE_FEEDS_URL="http://petalinux.xilinx.com/sswreleases/rel-v2020/aarch64/ssstate-cache"
IMAGE_INSTALL_append = "opencv"
```

באופן כללי לאחר יצירת הפרויקט :

```
projecta@ubuntu:~/Vitis-AI/BootExp/Petalinux$ petalinux-create -t project -s ../xilinx-zcu104-v2020.2-final.bsp
INFO: Create project:
INFO: Projects:
INFO: * xilinx-zcu104-2020.2
INFO: Has been successfully installed to /home/projecta/Vitis-AI/BootExp/Petalinux/
INFO: New project successfully created in /home/projecta/Vitis-AI/BootExp/Petalinux/
```

מקרים ותגובות:

במקרה של השגיאה הבאה :

petalinux-create: command not found

יש להסתכל בדיון הבא :

Program/References/PDFs_Archive/18

במהלך הבניה נתקלנו בשגיאה, פתרנו את השגיאה בעזרת הקישור הבא :

Program/References/PDFs_Archive/19

העברה לכרטיס זיכרון יש לבצע על פי המדריך הבא, עמוד 61 :

Program/References/PDFs_Archive/20

צריבת SD-Card

צריבת הכרטיס הינה פעולה הכרחית לטעינה קוד התכן אל הלוח. אנו נעזרנו בתוכנת צריבה אשר נראית etcher ובקובץ מוכן שניתן על ידי Xilinx. יש להיעזר באתחול במדריך הבא :

Program/References/PDFs_Archive/14

הערה – יש לבצע הרצה של התוכנה etcher כמנהל.

Setup של כרטיס FPGA

חבר טכני על חיבור הכבלים:

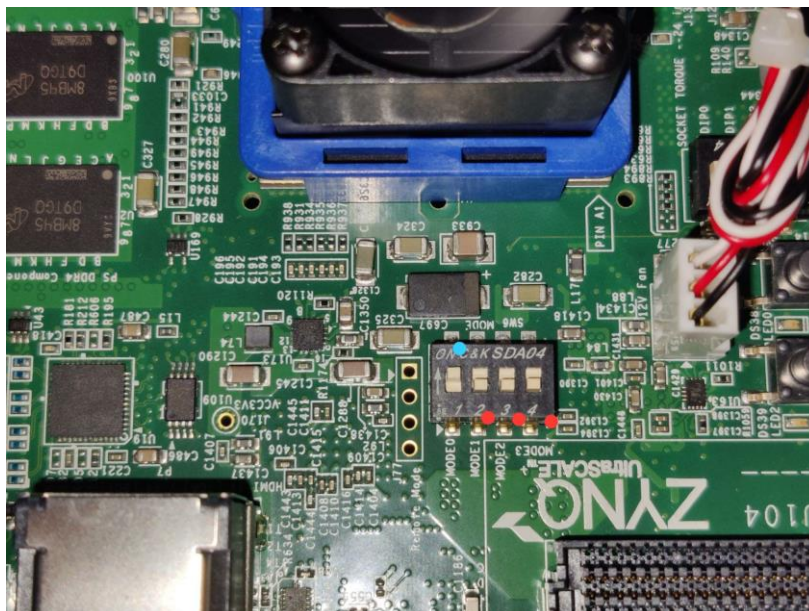
יש לחבר את הכבלים בצורה הבאה (יש להיעזר בתמונה הראשונה):

הכבל העליון מתחבר לחשמל, הכבל התחתון מתחבר לרשת והכבל הימני מתחבר למחשב. לאחר החיבור יש לבצע הדלקה של הלוח ולחכות כדקה.

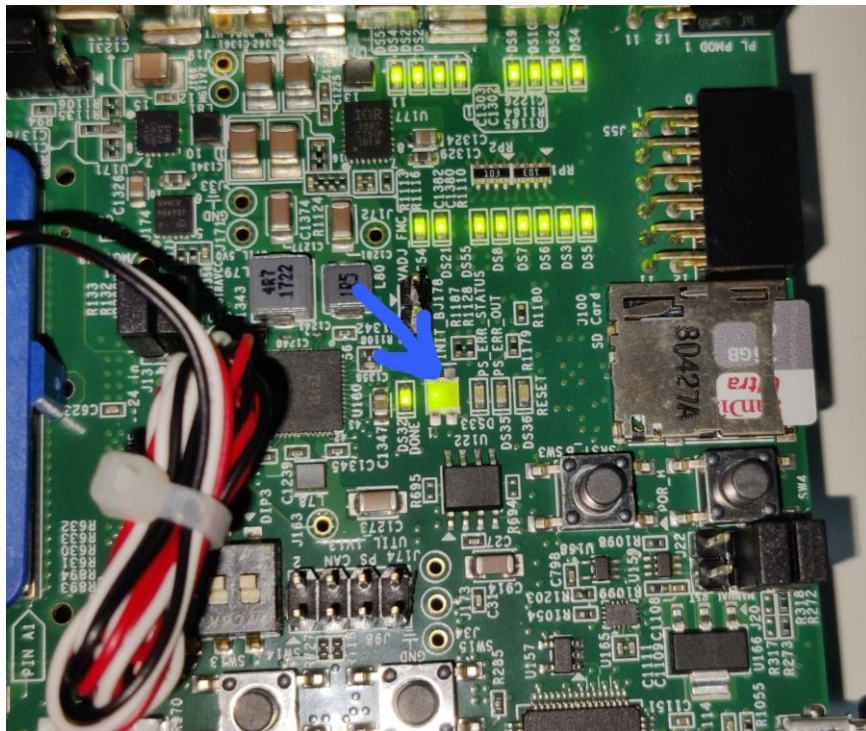
תמונה כללית של הלוח לאחר הדלקה (ZCU104):



יש לסדר את המתגים כמתואר בתמונה, כאשר כחול מעיד על למעלה ואדום על למטה.



יש לוודא כי הנורה המסומנת דלוקה :



כעת נחבר את הכרטיס למחשב האישי באמצעות כבל micro-usb. חיבור זה יוצר שלושה חיבורי COM: יש לבחור את הCOM בעל הערך המינימלי (לחפש במנהל ההתקנים).

כדי להתחבר לכרטיס ניעזר בתוכנת Putty שמאפשרת חיבור במגוון פרוטוקולים.

בתוכנה נבחר בחיבור טורי (סריאלי). המסך המתקבל בחיבור הטורי:

יש להגדיר baud rate: 115200 bps

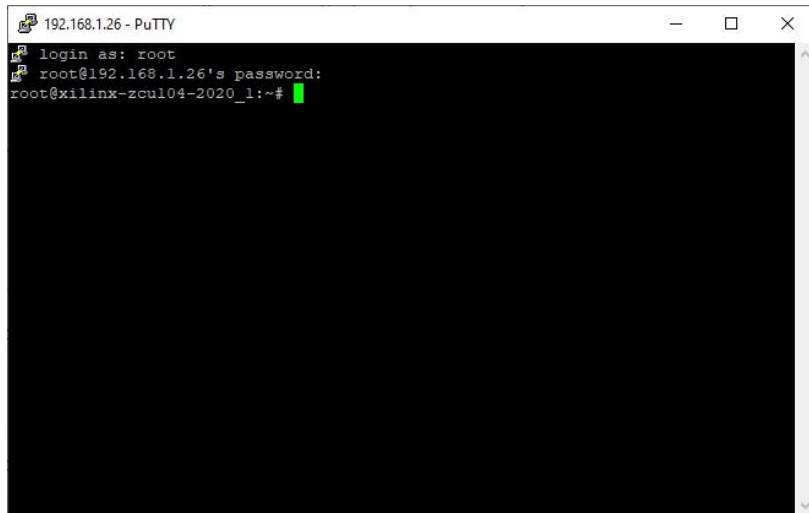
```
COM7 - PuTTY
Xilinx Zynq MP First Stage Boot Loader
Release 2020.1 Jun 22 2020 - 11:05:38
NOTICE: ATF running on XCZU7EV/silicon v4/RTL5.1 at 0xffffea000
NOTICE: BL31: v2.2(release):v1.1-5588-g5918e656e
NOTICE: BL31: Built : 10:53:46, Jun 22 2020

U-Boot 2020.01 (Jun 22 2020 - 10:56:22 +0000)

Model: ZynqMP ZCU104 RevC
Board: Xilinx ZynqMP
DRAM: 2 GiB
PMUFW: v1.1
EL Level: EL2
Chip ID: zu7ev
NAND: 0 MiB
MMC: mmc@ff170000: 0
In: serial@ff000000
Out: serial@ff000000
Err: serial@ff000000
Bootmode: LVL_SHFT_SD MODE1
Reset reason: EXTERNAL
Net:
ZYNQ GEM: ff0e0000, mdio bus ff0e0000, phyaddr 12, interface rgmii-id
```

לאחר ההתחברות אפשר לרשום ifconfig ולהתחבר ללוח עם כבל רשת בשם:

(שם משתמש וסיסמא – root)



ניתן להוריד תוכנה בשם – WinSCP – תוכנה אשר משמשת להעברת קבצים מהמחשב הלוכלי למחשב המרוחק עם תצוגה למשתמש.

לחלופין, ניתן להשתמש בפקודות הבאות:

פקודה להעביר תיקייה מהמחשב הלוכלי למחשב המרוחק:

```
scp -r c:\move2fpga\ root@192.168.1.26: ~/Vitis-AI/VART/samples/images
```

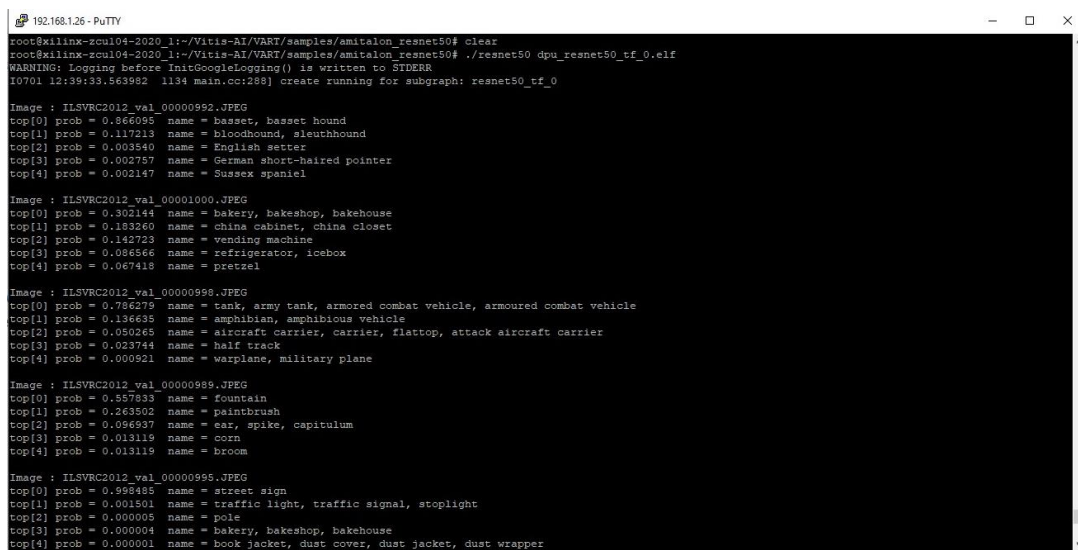
במקרה שמתקבלת תשובה לתמונה אחת בלבד יש לפעול לפי האתר הבא:

Program/References/PDFs_Archive/15

פקודת להעברת קובץ:

```
scp c:\move2fpga\dpu_resnet50_tf_0.elf root@192.168.1.26: ~/Vitis-AI/VART/samples/images
```

להלן דוגמה להרצה מוצלחת בFPGA:



קבצי הקוד:

Program/References/PDFs_Archive/16

```
root@xilinx-zcu104-2020_1: ~/Vitis-AI/VART/samples/amitalon_resnet50# ./resnet50
dpu_resnet50_tf_0.elf
```

WARNING: Logging before InitGoogleLogging() is written to STDERR

I0701 12:39:33.563982 1134 main.cc:288] create running for subgraph: resnet50_tf_0

הקמת רשת, קומפילציה והרצת הרשת: Partial Flow For Resnet-50

- שלב הקוונטיזציה

הדרכה כללית מאתר Xilinx:

Program/References/PDFs_Archive/6

הורדת validation set (שימושי באופן כללי בשלב זה):

Program/References/PDFs_Archive/7

הורדת מודל:

Program/References/PDFs_Archive/8

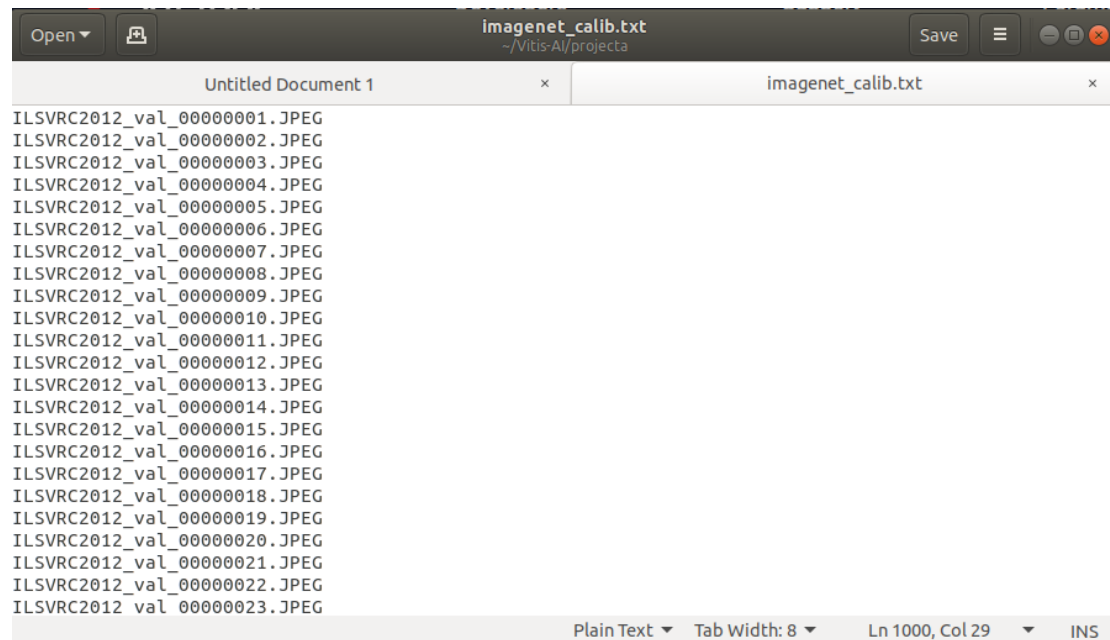
בחרנו במודל הבא:

47	tf_resnetv1_50_imagenet_224_224_6.97G_1.2	207MB	55576ff6afdc700ee00664642e19a6fa
----	---	-------	----------------------------------

זה המודל שהונחנו להשתמש בהחלטה משותפת עם המנחים.

נשים לב כי כעת אנו פועלים על גבי סביבת הUNIX (אלא אם נאמר אחרת).

יצרנו תיקייה בשם imagenet_images עם אלף תמונות מה validation set. יצרנו קובץ פייתון כפי שמוגדר במדריך ויצרנו קובץ imagenet_calib.txt. הוא מכיל אלף שורות מן הצורה:



פקודה להעתקת אלף הקבצים הראשונים:

```
cp `ls | head -1000` ../../Vitis-AI/projecta/imagenet_images
```


הפקודה שהרצנו לבצע את תהליך הקוונטיזציה:

```
vai_q_tensorflow quantize --input_frozen_graph resnet_v1_50_inference.pb --input_nodes
input --input_shapes ?,224,224,3 --output_nodes resnet_v1_50/predictions/Reshape_1 --
input_fn input_fn.calib_input --method 1 --gpu 0 --calib_iter 20 --output_dir ./quantize_results
```

ההודעה בסיום ההרצה:

```
projecta@ubuntu: /workspace/projecta
File Edit View Search Terminal Help
2020-11-07 02:50:50.280900: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 160563
200 exceeds 10% of system memory.
2020-11-07 02:50:50.623561: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 160563
200 exceeds 10% of system memory.
2020-11-07 02:50:50.818927: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 160563
200 exceeds 10% of system memory.
2020-11-07 02:50:51.604160: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 160563
200 exceeds 10% of system memory.
2020-11-07 02:50:51.864837: W tensorflow/core/framework/cpu_allocator_impl.cc:81] Allocation of 802816
00 exceeds 10% of system memory.
INFO: Float Graph Check Done.
2020-11-07 02:50:54.387672: W tensorflow/contrib/decent_q/utils/quantize_utils.cc:538] Convert mean no
de resnet_v1_50/pool5 to AvgPool
2020-11-07 02:50:54.392173: W tensorflow/contrib/decent_q/utils/quantize_utils.cc:628] Scale output of
avg_pool node resnet_v1_50/pool5 to simulate DPU.
INFO: Calibrating for 20 iterations...
100% (20 of 20) |#####| Elapsed Time: 0:22:59 Time: 0:22:59
INFO: Calibration Done.
INFO: Generating Deploy Model...
[DEPLOY WARNING] Node resnet_v1_50/predictions/Reshape_1(Type: Reshape) is not quantized and cannot be
deployed to DPU,because it has unquantized input node: resnet_v1_50/predictions/Softmax. Please deplo
y it on CPU.
INFO: Deploy Model Generated.
***** Quantization Summary *****
INFO: Output:
quantize_eval_model: ./quantize_results/quantize_eval_model.pb
deploy_model: ./quantize_results/deploy_model.pb
(vitis-ai-tensorflow) projecta@ubuntu: /workspace/projecta$ ll
```

שתי תקלות שחשוב להכיר ולהבין:

1. תקלת core dumped – הסיבה לתקלה זו נבעה אצלנו כתוצאה מהקצאה של מקום בכוון שאין בו מקום, לכן חשוב להקצות מקום בכוון עם הרבה מקום, במקרה של המחשב הנוכחי זה היה כוון D.
2. תקלת FMA Instructions not supported – תקלה אשר התקבלה כתוצאה מעבודה עם virtual box, חשוב לעבוד בVMARE אחרת לא ניתן לעבוד.

• שלב הקומפילציה

מדריך כללי מאתר Xilinx :

Program/References/PDFs_Archive/9

הפקודה לביצוע קומפילציה (הרצנו אותה מתוך אותה תיקייה שהרצנו את הפקודה של הquantization) :

```
vai_c_tensorflow --frozen_pb ./quantize_results/deploy_model.pb --arch  
/opt/vitis_ai/compiler/arch/dpuv2/ZCU104/ZCU104.json --output_dir model --net_name  
resnet50_tf
```

הערה : הפקודה באתר איננה מדויקת, הפקודה שרשמנו כאן היא המדויקת.

הפלט שקיבלנו :

```
projecta@ubuntu: /workspace/projecta$ vai_c_tensorflow --frozen_pb  
./quantize_results/deploy_model.pb --arch  
/opt/vitis_ai/compiler/arch/dpuv2/ZCU104/ZCU104.json --output_dir model --net_name  
resnet50_tf
```

.VITIS_AI Compilation - Xilinx Inc *

[VAI_C][Warning] layer [resnet_v1_50_SpatialSqueeze] (type: Squeeze) is not supported in
.DPU, deploy it in CPU instead

[VAI_C][Warning] layer [resnet_v1_50_predictions_Softmax] (type: Softmax) is not supported
.in DPU, deploy it in CPU instead

"Kernel topology "resnet50_tf_kernel_graph.jpg" for network "resnet50_tf

"kernel list info for network "resnet50_tf

Kernel ID : Name

resnet50_tf_0 : 0

resnet50_tf_1 : 1

Kernel Name : resnet50_tf_0

Kernel Type : DPUKernel

Code Size : 0.59MB

Param Size : 24.35MB

Workload MACs : 6964.51MOPS

IO Memory Space : 2.25MB

,Mean Value : 0, 0, 0

Total Tensor Count : 59

Boundary Input Tensor(s) (H*W*C)

input: 0(0) : 224*224*3

Boundary Output Tensor(s) (H*W*C)

resnet_v1_50_logits_Conv2D: 0(0) : 1*1*1000

Total Node Count : 58

Input Node(s) (H*W*C)

resnet_v1_50_conv1_Conv2D(0) : 224*224*3

Output Node(s) (H*W*C)

resnet_v1_50_logits_Conv2D(0) : 1*1*1000

Kernel Name : resnet50_tf_1

Kernel Type : CPUPKernel

Boundary Input Tensor(s) (H*W*C)

resnet_v1_50_SpatialSqueeze: 0(0) : 1*1*1000

Boundary Output Tensor(s) (H*W*C)

resnet_v1_50_predictions_Softmax: 0(0) : 1*1*1000

Input Node(s) (H*W*C)

resnet_v1_50_SpatialSqueeze : 1*1*1000

Output Node(s) (H*W*C)

resnet_v1_50_predictions_Softmax : 1*1*1000

נציין כי קיבלנו תיקיית מודל ולהלן הפירוט שלה:

```
(vitis-ai-tensorflow) projecta@ubuntu:/workspace/projecta/model$ ll
total 25604
drwx----- 2 projecta vitis-ai-users  4096 Nov  7 04:07 ./
drwxr-xr-x  7 projecta      1000    4096 Nov  7 04:06 ../
-rw-r--r--  1 projecta vitis-ai-users 26204736 Nov  7 04:07 dpu_resnet50_tf_0.elf
-rw-r--r--  1 projecta vitis-ai-users    325 Nov  7 04:07 resnet50_tf_kernel_graph.gv
```

הקמת רשת, קומפילציה, אופטימיזציה וסיווג: Full Flow For Mnist

בשלב זה אנו יוצרים מודל בצורה מלאה, החל מכתובתו ב-TF ועד הרצת שלב הסיווג על גבי הכרטיס.

התבססנו על המדריך הבא :

Program/References/PDFs_Archive/10

מיקום בגיט: [References/Vitis-AI-Tutorials-MNIST-Classification-TensorFlow.zip](#)

מצוין באדום הנקודות בהן השתמשנו במדריך.

ראשית, כתבנו קוד בcolab בשפת python בעזרת ספריות tensorflow, keras.

הקוד הסופי מצורף (נעזרנו במודל של המדריך):

מיקום בגיט: [References/Start.ipynb](#)

לאחר מכן הרצנו קוד על הטנזור וקיבלנו גרף, לאחר מכן ביצענו "הקפאה" של הגרף על מנת שנוכל לעבוד איתו במכונה הוירטואלית.

Program/References/PDFs_Archive/11

הערות :

- לא ניתן להעזר בגרסה tensorflow 2 ומעלה (דיון בפורום בנוגע להתאמה בין גרסאות של טנזור):

Program/References/PDFs_Archive/12

- לא ניתן לעבוד עם שכבות FC, יש לעבוד עם שכבות קונבולוציה.

Program/References/PDFs_Archive/13

- נוצרה בעיה כאשר נעזרנו בשכבת dense-softmax, המרנו את השכבה לשכבת סיגמואיד והגענו לתוצאה מעולה.

לאחר מכן יצרנו קבצי קליברציה :

מיקום בגיט: [References/trainimages_calib_original.zip](#)

סקריפטים ליצירת התמונות :

מיקום בגיט: [References/4_quant.zip](#)

- שלב הקוונטיזציה

בשלב הקוונטיזציה נעזרנו בסקריפט (הועתק מהמדריך): `image_input_fn.py`

תוך כדי שהגדרנו בסקריפט מחדש את השם של ה-Input node (נלקח מהלוגים של הקפאת המודל בקולאב). השורה מסומנת בכתום.

```
calib_image_list = './calib.txt'
calib_batch_size = 10

def calib_input(iter):
    images = []
    line = open(calib_image_list).readlines()
    for index in range(0, calib_batch_size):
        curline = line[iter * calib_batch_size + index]
        calib_image_name = curline.strip()

        # open image as grayscale
        image = cv2.imread(calib_image_name, cv2.IMREAD_GRAYSCALE)

        # reshape
        image = image.reshape(28,28,1)

        # normalize
        image = image/255.0

        images.append(image)
    return {"flatten_input_1": images}
```

פקודת להרצת תהליך הקוונטיזציה:

```
vai_q_tensorflow quantize --input_frozen_graph frozen_graph.pb --input_nodes
conv2d_input_1 --input_shapes ?,28,28,1 --output_nodes flatten_1/Reshape --input_fn
image_input_fn.calib_input --method 1 --gpu 0 --calib_iter 10 --output_dir ./quantize_results
```

- שלב הקומפילציה

פקודה להרצת תהליך הקומפילציה:

```
vai_c_tensorflow --frozen_pb ./quantize_results/deploy_model.pb --arch
/opt/vitis_ai/compiler/arch/dpuv2/ZCU104/ZCU104.json --output_dir model --net_name
mnist_tf
```

הפלט לאחר קומפילציה:

: INFO: Output

```
quantize_eval_model: ./quantize_results/quantize_eval_model.pb
deploy_model: ./quantize_results/deploy_model.pb
(vitis-ai-tensorflow) projecta@ubuntu: /workspace/projecta-mnist$ vai_c_tensorflow --
frozen_pb ./quantize_results/deploy_model.pb --arch
/opt/vitis_ai/compiler/arch/dpuv2/ZCU104/ZCU104.json --output_dir model --net_name
mnist_tf
```

.VITIS_AI Compilation - Xilinx Inc *

[VAI_C][Warning] layer [activation_1_Sigmoid] (type: Sigmoid) is not supported in DPU,
 .deploy it in CPU instead

"Kernel topology "mnist_tf_kernel_graph.jpg" for network "mnist_tf

"kernel list info for network "mnist_tf

Kernel ID : Name

mnist_tf_0 : 0

mnist_tf_1 : 1

Kernel Name : mnist_tf_0

Kernel Type : DPUKernel

Code Size : 3.46KB

Param Size : 0.08MB

Workload MACs : 3.08MOPS

IO Memory Space : 5.86KB

,Mean Value : 0, 0, 0

Total Tensor Count : 5

Boundary Input Tensor(s) (H*W*C)

conv2d_input_1:0(0) : 28*28*1

Boundary Output Tensor(s) (H*W*C)

conv2d_3_1_Conv2D:0(0) : 1*1*10

Total Node Count : 4

Input Node(s) (H*W*C)

conv2d_4_Conv2D(0) : 28*28*1

Output Node(s) (H*W*C)

conv2d_3_1_Conv2D(0) : 1*1*10

Kernel Name : mnist_tf_1

Kernel Type : CPUKernel

Boundary Input Tensor(s) (H*W*C)

activation_1_Sigmoid: 0(0) : 1*1*10

Boundary Output Tensor(s) (H*W*C)

activation_1_Sigmoid: 0(0) : 1*1*10

Input Node(s) (H*W*C)

activation_1_Sigmoid : 1*1*10

Output Node(s) (H*W*C)

activation_1_Sigmoid : 1*1*10

- שלב הסיווג/הרצה על board

יצרנו תיקייה חדשה על גבי הרכיב.

העברנו את הקובץ הבא (מועתק מהמדריך):

References/ app_mt.py : מיקום בגיט

בנוסף העברנו תמונות לסיווג ואת קובץ ELF של הקומפילציה.

/home/root/Vitis-AI/VART/samples/projecta-mnist/				
Name	Size	Changed	Rights	Owner
..		12/07/2020 07:21:33	rw-r-xr-x	root
images		12/07/2020 08:17:31	rw-r-xr-x	root
dpu_mnist_tf_0.elf	87 KB	19/12/2020 17:00:49	rw-r--r--	root
app_mt.py	6 KB	20/12/2020 12:04:07	rw-r--r--	root

כעת ביצענו את הרצת test set:

`python3 app_mt.py -m dpu_mnist_tf_0.elf`

לשינוי threads להוסיף -t

```

root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m dpu_mnist_tf_0.elf
Command line options:
  --image_dir : images
  --threads   : 1
  --model     : dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 1 threads...
AmitAlon log: dpu function
FPS=2898.11, total frames = 10000 , time=3.4505 seconds
Correct: 9819 Wrong: 181 Accuracy: 0.9819
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m dpu_mnist_tf_0.elf -t 6
Command line options:
  --image_dir : images
  --threads   : 6
  --model     : dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 6 threads...
AmitAlon log: dpu function
AmitAlon log: dpu function
AmitAlon log: dpu function
AmitAlon log: dpu function
AmitAlon log: dpu function
AmitAlon log: dpu function
FPS=3700.52, total frames = 10000 , time=2.7023 seconds
Correct: 9819 Wrong: 181 Accuracy: 0.9819
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist#

```

בוצע על מס' threads שונה, 1 ו-6.

בדיקת ביצועים לאחר שינוי פרמטרים:

ביצועים	תיקיה	פרמטר שינוי	שלב ארכיטקטוני	מבוצע על ידי
Accuracy 98.25% 2.221[Sec] 0.02221 [$\frac{\mu Sec}{Pic}$]	-	-	-	Colab
Accuracy 98.19% 3.4864[Sec] 0.034864 [$\frac{\mu Sec}{Pic}$]	Modifications/Quantization/ Original	-	-	Board
ביצועים ירודים כתוצאה מחוסר תמיכה של המערכת	Modifications/Quantization/ Weight_bit_16 INT8 is only supported for DPU deployment. You can use 16 and 32 bits for simulation, but you won't be able to deploy this on DPU hw.	--weight_bit Bit width for quantized weight and bias. הערך הדיפולטי : .8 שינוי שביצענו : מעבר ל16.	– Quantization קוונטיזציה	Board
ביצועים ירודים כתוצאה מחוסר תמיכה של המערכת	Modifications/Quantization/ Activation_bit_16 INT8 is only supported for DPU deployment. You can use 16 and 32 bits for simulation, but you won't be able to deploy this on DPU hw.	--activation_bit Bit width for quantized activation. הערך הדיפולטי : .8 שינוי שביצענו : מעבר ל16.	– Quantization קוונטיזציה	Board
Accuracy 98.22% 3.4645[Sec] 0.034645 [$\frac{\mu Sec}{Pic}$]	Modifications/Quantization/ Calib_Iter_100	--calib_iter The iterations of calibration. Total number of images for calibration = calib_iter * batch_size.	– Quantization קוונטיזציה	Board
Accuracy 98.33% 3.4648[Sec] 0.034648 [$\frac{\mu Sec}{Pic}$]	Modifications/Quantization/ Method_0	שתי שיטות על פיהן ניתן לקוונט. שיטה 0 יוצרת טווח רחב של ערכים קוונטיזציה ושיטה 1 מאפשרת טווח צר יותר.	– Quantization קוונטיזציה	Board
Accuracy 98.19% 2.5925[Sec] 0.025925 [$\frac{\mu Sec}{Pic}$]	-	--threads Number of threads. Default is 1	– Inference סיווג	Board

ללא שינויים:

```
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m dpu_mnist_tf_0.elf
Command line options:
--image_dir : images
--threads   : 1
--model     : dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 1 threads...
AmitAlon log: dpu function
FPS=2868.33, total frames = 10000 , time=3.4864 seconds
Correct: 9819 Wrong: 181 Accuracy: 0.9819
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist#
```

Method 0

```
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m Method_0/model/dpu_mnist_tf_0.elf
Command line options:
--image_dir : images
--threads   : 1
--model     : Method_0/model/dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 1 threads...
AmitAlon log: dpu function
FPS=2886.14, total frames = 10000 , time=3.4648 seconds
Correct: 9833 Wrong: 167 Accuracy: 0.9833
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist#
```

Calib:

```
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m Calib_Iter_100/model/dpu_mnist_tf_0.elf
Command line options:
--image_dir : images
--threads   : 1
--model     : Calib_Iter_100/model/dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 1 threads...
AmitAlon log: dpu function
FPS=2885.50, total frames = 10000 , time=3.4656 seconds
Correct: 9822 Wrong: 178 Accuracy: 0.9822
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist#
```

: Activation

```
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m Activation_bit_16/model/dpu_mnist_tf_0.elf
Command line options:
--image_dir : images
--threads   : 1
--model     : Activation_bit_16/model/dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 1 threads...
AmitAlon log: dpu function
FPS=2841.37, total frames = 10000 , time=3.5194 seconds
Correct: 980 Wrong: 9020 Accuracy: 0.098
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist#
```

: Weight

```
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m Weight_bit_16/model/dpu_mnist_tf_0.elf
Command line options:
--image_dir : images
--threads   : 1
--model     : Weight_bit_16/model/dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 1 threads...
AmitAlon log: dpu function
FPS=2885.55, total frames = 10000 , time=3.4655 seconds
Correct: 980 Wrong: 9020 Accuracy: 0.098
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist#
```



```

root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m dpu_mnist_tf_0.elf
Command line options:
--image_dir : images
--threads   : 1
--model     : dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 1 threads...
FPS=2884.90, total frames = 10000 , time=3.4663 seconds
Correct: 9819 Wrong: 181 Accuracy: 0.9819
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m dpu_mnist_tf_0.elf -t 3
Command line options:
--image_dir : images
--threads   : 3
--model     : dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 3 threads...
FPS=3803.81, total frames = 10000 , time=2.6289 seconds
Correct: 9819 Wrong: 181 Accuracy: 0.9819
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m dpu_mnist_tf_0.elf -t 6
Command line options:
--image_dir : images
--threads   : 6
--model     : dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 6 threads...
FPS=3709.51, total frames = 10000 , time=2.6958 seconds
Correct: 9819 Wrong: 181 Accuracy: 0.9819
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m dpu_mnist_tf_0.elf -t 7
Command line options:
--image_dir : images
--threads   : 7
--model     : dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 7 threads...
FPS=3816.50, total frames = 10000 , time=2.6202 seconds
Correct: 9819 Wrong: 181 Accuracy: 0.9819
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m dpu_mnist_tf_0.elf -t 8
Command line options:
--image_dir : images
--threads   : 8
--model     : dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 8 threads...
FPS=3857.26, total frames = 10000 , time=2.5925 seconds
Correct: 9819 Wrong: 181 Accuracy: 0.9819
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist# python3 app_mt.py -m dpu_mnist_tf_0.elf -t 9
Command line options:
--image_dir : images
--threads   : 9
--model     : dpu_mnist_tf_0.elf
Pre-processing 10000 images...
Starting 9 threads...
FPS=3809.07, total frames = 10000 , time=2.6253 seconds
Correct: 9819 Wrong: 181 Accuracy: 0.9819
root@xilinx-zcu104-2020_1:~/Vitis-AI/VART/samples/projecta-mnist#

```

טיפים, בעיות ופתרון בעיות

הבעיה	מה היא אומרת	איך ניתן לתקן
שגיאת core dumped	בתחילה ניסינו ליצור מכונה עם גודל אחסון של כמה עשרות גיגה, הגענו במהירות לאחסון מלא.	נתנו למכונה הוירטואלית מרחב אחסון גדול בהרבה של 350 גיגה.
Virtual box לא תומך בפקודות FMA		יצרנו מכונה וירטואלית ב-VMARE.
המכשור לא מסוגל לעבוד עם גרסה 2 tensorflow ומעלה		ניתן לבצע המרה בתחילת הקוד ב colab לגרסה ישנה יותר כפי שהראנו בקוד.
המכשור לא מסוגל לעבוד עם שכבות FC אחת אחרי השניה.	באופן כללי מתבצעת המרה משכבת FC לשכבת קונבולוציה. כאשר משתמשים בשתי שכבות כאלו אחת אחרי השניה נוצר מצב בו גודל הכניסה לשכבת הקונבולוציה גדול מהגודל המקסימלי (16x16).	מעבר לארכיטקטורה של שכבות קונבולוציה בלבד כפי שהראנו בקוד שהכנו.
שכבת dense softmax נכשלת בשלב הקוונטיזציה	הקוונטיזציה לא יודעת איך לעבוד עם סוג המשתנה שיוצא משכבה זו (int32)	עברנו לשכבת אקטיביזציה אחרת בשם סיגמואיד.
שכבת סיגמואיד לא עובדת על ה DPU	מכיוון שהמשתנה היוצא משכבה זו הינו int16 ה DPU לא יכול להריץ אותו ולכן הוא מורץ על ה CPU.	זוהי אינה בעיה. ה DPU לא יכול להריץ את כל סוגי השכבות. אין בעיה שחלק מהקוד יורץ על ה CPU.
ה DPU לא תומך ברוחב ביט שאינו 8		הפתרון הינו כמובן לעבוד ברוחב ביט זה בלבד.

תובנות כלליות

מסקנות שעלו בעקבות תהליך הביצועים והאופטימיזציה

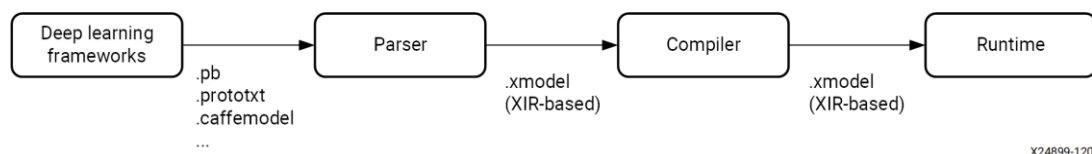
- ביצועים אופטימליים מבחינת סיווג – יש לבצע קוונטיזציה על פי שיטה 0 אשר עובדת עם טווח רחב של ערכים.
- ביצועים אופטימליים מבחינת זמנים – במקרה של עבודה עם הלוח יש לבצע חלוקה למספר חוטים אשר נותן תוצאה אופטימלית. במקרה שלנו זה התקבל עבור 8 חוטים.
- יש לעבוד עם רוחב ביט של 8.
- יש לבצע שינויים בפרמטרים בשלב הקוונטיזציה, בשלב הקומפילציה לא נמצאו פרמטרים רלוונטיים.

הרצת שכבות על גבי DPU ועל גבי CPU

כפי שניתן לראות בתוצאות הקומפילציה, ישנן שכבות אשר מיועדות לריצה על גבי המעבד (ARM) וחלקן מיועדות לריצה על גבי הDPU. נרצה טיפה להסביר על המניע וכיצד המנגנון הפנימי עובד.

התשתיות בפרויקט מתבססות על פרוטוקול ייצוג XIR - Xilinx Intermediate Representation. זהו פרוטוקול מבית XILINIX של ייצוג גרף לאלגוריתמי AI. הוא נועד להצגה איכותית של רשתות נוירונים ולאחר מכן גם לפרישה נוחה ומהירה על גבי פלטפורמות חומרה שונות (במקרה שלנו DPU על גבי הFPGA).

באופן כללי התהליך נראה כך :



X24899-120920

כאשר המודל (רשת הנוירונים הסופית) מכילה פעולות שהDPU אינו יכול לבצע ברמת החומרה, נוצרים Sub-graph לשכבות הספציפיות הללו והם ממופים לביצוע על גבי ה-CPU. המידע לגבי איזה שכבות ניתן להריץ על גבי הCPU ואילו לא, מוחלט בשלב הקומפילציה על ידי הקומפיילר. בסופו של דבר, כאשר נעזרים ב-Vitis AI unified high-level API המודל שלנו ינסה לרוץ על ה-DPU וכאשר לא יוכל יבצע חלק מהפעולות על גבי הCPU. באופן כללי, הData Flow הוא מן הצורה: ...CPU--DDR--DPU--DDR...

קימפול ה-RTL לכרטיס (בדיקת היתכנות)

מתוך המחקר שביצענו, אנו ראינו כי אכן יש אפשרות להצליח לקמפל את קוד הDPU ולהורידו לכרטיס. עם זאת, מדובר בתהליך מורכב, הכולל שלבים רבים של התאמות וכולל בעיות רבות של סנכרון בין גרסאות שונות של תוכנות שונות ושימוש בקבצי תלויות מסוימים. כפי שניתן לראות, המדריך המקורי (מדריך 17) יועד לכרטיס אחר - ZCU-102. אנו מאמינים שבמידה וניתן יהיה להשיג כרטיס זה, עבודת הקימפול והצריבה לכרטיס לאחר מודיפיקציות שונות תתבצע בקלות יותר (אך עדיין מורכבת לאור התלויות המשתנות).

אחרית דבר

- במקרים מסוימים, אכן הכרטיס בשילוב סביבת Vitis-AI מצליחים להשיג אחוזי דיוק טובים יותר מאשר מאשר סיווג על גבי מחשב ממוצע וזאת הודות לחומרה הייעודית שלו.
- חברת Xilinx הצליחה להעניק בצורה יחסית פשוטה יכולות של בינה מלאכותית לכל החפץ בכך - גם כאלה בעלי מעט מאוד ניסיון בעולם זה.
- יצירת מודל חדש הינו תהליך מורכב מאוד. לאורך התהליך מהרגע שבו יצרנו מודל חדש ועד להרצתו על גבי הכרטיס, נדרשו מספר רב של התאמות לאורך הדרך.
- לצערנו המידע על תשתיות Xilinx ברשת הינו מוגבל ולעיתים קרובות קשה מאוד להתמודד עם בעיות שצפות במהלך תהליך הפיתוח של מודל חדש.