Mipsology

# Zebra®

## VMAccel™ FPGA Cloud

## AMD/Xilinx® Versal VCK5000

## User Guide

Zebra Version: V2022.versal.07

Mipsology

TABLE OF CONTENTS

# 1 Introduction & Scope

This document is a guide for *running* Mipsology® Zebra CNN inference acceleration software *on* AMD/Xilinx® VCK5000 PCIe Acceleration Card *hosted at* VMAccel® FPGA Cloud *with your neural network*.

Note that this is an Alpha quality release with following goals:

- Demonstrates Zebra functionality on VCK5000 board.
- Demonstrates Zebra software Ease-of-Use (EoU).
    - Accelerate trained Convolution Neural Network (CNN) model *without* any structural modification. NO pruning or re-training of the model.
    - Automatic and in-line quantization/calibration. NO offline or separate compilation tool.

For Performance and Accuracy, please refer to respective section in this document.

# 2 License

VMAccel Zebra virtual machines (VM) are pre-configured with software license. This license is **not** designed for production deployment. Any CNN inference running continuously for more than 15 minutes will experience significant slowdown in execution.

# 3 Contact & Support

Please email support@mipsology.com for questions, concerns, technical help or discuss your project's unique requirements.

Please email licenses@mipsology.com for questions related to Zebra License.

# 4 Requirements

VMAccel Zebra instances are pre-configured with all required software and hardware. Only requirement on client side is a computer with internet connection and web browser.

# 5 VMAccel Cloud Access

To gain access to VMAccel cloud, please fill the form https://www.vmaccel.com/zebrademo

For any questions or concerns, please contact support@vmaccel.com

# 6 Getting Started on VMAccel

## 6.1 Launching Zebra VM Instance

Once you have the access credentials from VMAccel, please follow “Getting Started” section from https://vmaccel.atlassian.net/wiki/spaces/docs/pages/59212022/Getting+Started

Summary:

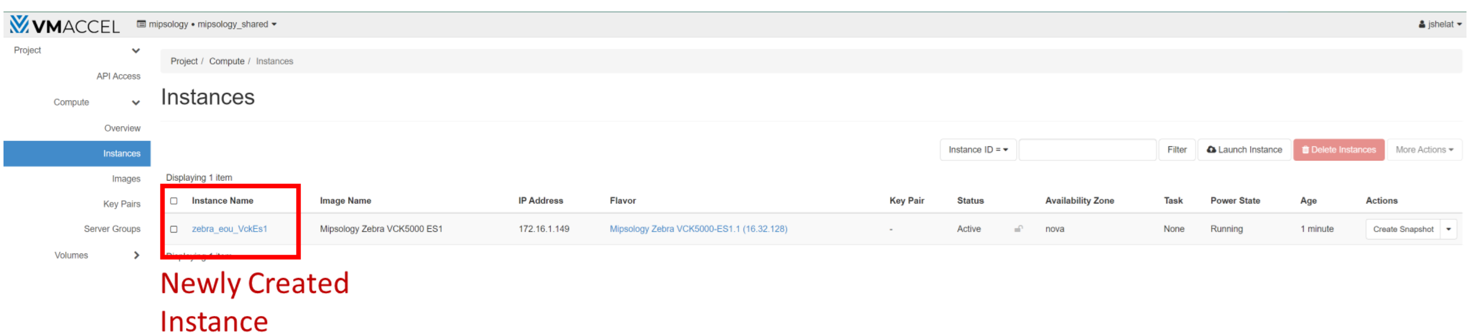
- In a web browser navigate to: https://xilinx2.vmaccel.com/dashboard/project/
- On left hand side: Click on “Instances”
- On right hand side: Click on “Launch Instance”
- Follow GUI instructions/options for configuration

**IMP:** Use following details when creating Zebra VM instance:

- Source : Secure Boot Image = “Mipsology Zebra VCK5000 ES1”
- Flavor = “Mipsology Zebra VCK5000-ES1.1 (16.32.128)”
- Network = “mipsology_local”

For advanced features like enabling external `ssh` access, follow instructions here : https://vmaccel.atlassian.net/wiki/spaces/docs/pages/38830174/Connect+to+Instance+via+SSH

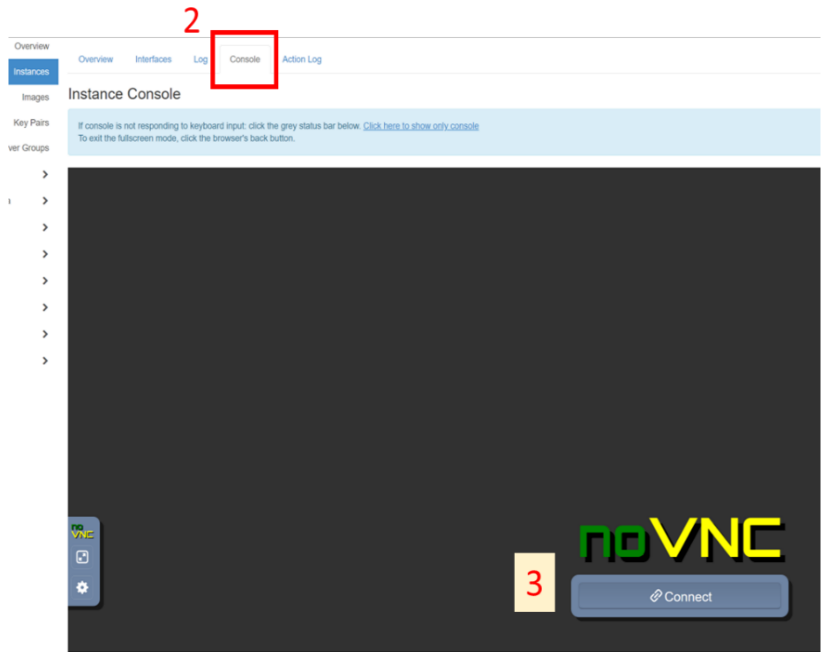
Once created, you should see the instance listed as a row on “Instances” web page/view. Example screenshot:



## 6.2 Starting Zebra on VM Instance

After the VM instance is created successfully:

1. Start the instance by clicking on the “Instance Name”
2. Click on “Console”
3. “Connect” to noVNC




**Inside VNC session:**

This User Guide is open by default.

Open “Terminal Emulator”
(Icon available on Desktop)

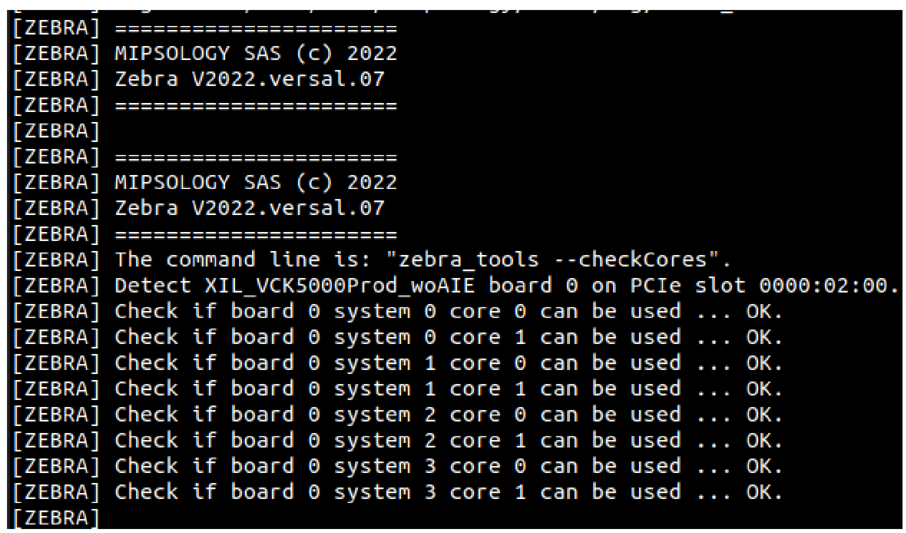
```
cd zebra
```

```
cd V2022.versal.07
```

```
./examples/docker/run.bash
```
(starts Zebra Docker)

```
cd zebra
. settings.sh
cd examples
zebra_tools --checkCores
```

Zebra is ready IF above command shows status <u>without</u> any errors.




*At this point, User is inside Zebra docker running on user-configured VM instance targeting VCK5000 board.*

# 7 Mipsology Examples and Demos

Zebra ships with many Examples and Demos provided inside Zebra Docker image. For all of these, the CNN model / Neural Network (NN) is from open-source community.

No modifications are done to the NNs. These models/NNs are used as-is and are exactly same as the ones executed on GPU/CPU. The models/NNs are trained in 32-bit Floating Point (FP32). NO training or retraining or pruning is required when running inference with Zebra software. Zebra starts from the trained model and automatically maps the neural network for the FPGA target, including operation like quantization.

Mipsology does not provide a model-zoo because Zebra software is designed to accelerate neural networks trained on GPU without modification.

## 7.1 Examples Quick Start

Inside Zebra docker

```
cd
cd zebra
. settings.sh
cd examples
```

```
./run_classification.sh -n resnet50 -f tensorflow
```

(runs TensorFlow1 ResNet-50v1 Inference on VCK5000)

- NOTE: when a network is executed for 1st time, Zebra will automatically perform Quantization/Calibration for INT8 inference. Result of calibration is saved and reused in future.

```
./run_classification.sh -n resnet50 -f pytorch
```

(runs PyTorch ResNet-50v1.5 inference on VCK5000)

```
./run_classification.sh -n resnet50 -f tensorflow2
```

(runs TensorFlow2 ResNet-50 Inference on VCK5000).

- NOTE: For Pytorch and TensorFlow2 : when executed 1st time, the model gets automatically downloaded from Internet. Downloaded models are saved and reused in future.

Results of inferences can be found in `predict.log` file.

## 7.2 Examples Details

Zebra provides example scripts and software to execute inference on various CNN post-trained models. All the models are open source – i.e. downloaded from internet (graph and weights/parameters). In case of Pytorch and TensorFlow2 frameworks, the models get automatically downloaded when a User executes for first time. Intent of Examples is to demonstrate seamless flow for FPGA acceleration of the CNN model. Intent is not to demonstrate full application execution.

User interface is "`run_classification.sh`" script located inside `examples` directory. The associated Python software is developed my Mipsology to let users run inference on these NNs easily. User's application can also be used if they are using supported framework and APIs.

Table below shows list of networks supported across frameworks in current release. Please use *network* and *framework* names in this table to run inference on associated model. For e.g.:

PyTorch ResNet-18 : `./run_classification.sh -f pytorch -n resnet18`
TensorFlow2 ResNet101 : `./run_classification.sh -f tensorflow2 -n resnet101`
TensorFlow1 Inceptionv4 : `./run_classification.sh -f tensorflow -n inceptionv4`

| **pytorch** | **tensorflow** | **tensorflow2** |
|---|---|---|
| Model Source: https://pytorch.org/vision/0.9/models.html | Model Source: Open-source models gathered from internet. | Model Source: https://tfhub.dev/ https://storage.googleapis.com/ |
| resnet50 | inceptionv4 | resnet50v1 |
| resnet18 | inceptionv3 | resnet50 |
| resnet34 | vgg16 | resnet50v2 |
| resnet101 | vgg19 | resnet101 |
| resnet152 | resnet50 | resnet101v2 |
| densenet121 | resnet50-v1.5 | resnet152 |
| densenet161 | resnet152 | resnet152v2 |
| densenet169 | mobilenet_v1 | inceptionv1 |
| densenet201 | mobilenet_v2 | inceptionv2 |
| inceptionv3 | yolov1 | inceptionv3 |
| mobilenet_v2 | yolov2 | inception_resnet_v2 |
| wide_resnet50_2 | yolov3 | mobilenet_v1 |
| wide_resnet101_2 | | mobilenet_v2 |
| squeezenet | | vgg16 |
| squeezenet1_1 | | vgg19 |
| vgg11 | | densenet121 |
| vgg11_bn | | densenet169 |
| vgg13 | | densenet201 |
| vgg13_bn | | |
| vgg16 | | |
| vgg16_bn | | |
| vgg19 | | |
| vgg19_bn | | |
| resnext50_32x4d | | |
| resnext101_32x8d | | |

`run_classification.sh` has many options that you can discover by adding the option "`--help`". Note that some options will force the mapping, optimization and/or quantization of the network to be redone as they impact the way the computation of the network is done on Zebra; then requiring more time than a simple inference execution.

Note that in the log file, all lines not preceded by "`[ZEBRA]`" are from libraries (like Python). "`[ZEBRA]`" is the header for lines printed by Zebra. "`[MIPSO]`" is the header printed by the application when Mipsology's general application is used.

### Expectation for Future Release

In upcoming releases, Zebra is expected to support and demonstrate increasing number of CNN models (and associated layers) across all three popular ML Frameworks.

## 7.3 Demos Quick Start

Inside Zebra docker

```
cd
source zebra/settings.sh
cd tensorflow-yolov4-tflite
```

(taking YOLO-v4 as an example)

```
zebra_config --system --add runSession.enableTimeStatistics=true
```

(enable printing performance table/statistics at end of Zebra run. By adding "`system`" flag, this option is enabled globally for all subsequent Zebra runs.)

```
./run ~/zebra/examples/VIDEO/paris_cut1.mkv
```

(run YOLO-v4 inference on VCK5000)

NOTES:

- By default, the application will show output video/pictures in a new GUI window.
- By default, inference is executed using `batch=1`.
  - Which means only 1 Zebra core is being utilized.
  - For VCK5000 this means throughput (FPS) is 1/8th of full Zebra performance.
- Summary table printed (in terminal window) by Zebra at end of inference execution provides many relevant details.
- This demo supports other pretrained CNN models – YOLO-v3, TinyYOLO-v4 and TinyYOLO-v3.
- Please study the '`run`' script and `detect.py` application for more options.

## 7.4 Demos Details

Zebra executes inside an ML framework. Zebra executes "in-line" with user application, including Quantization and Calibration. When accelerating CNN inference with Zebra, there is no additional tool for offline processing. Intent of Demos is to demonstrate effortless FPGA acceleration of applications. Intent is not to demonstrate a fully optimized and ready-to-deploy application.

Zebra aims to run inference with no change to application software and the model/NN. However, many GitHub repositories are not designed for easy execution on CPU or any accelerator (including GPU). Hence, Zebra makes following modifications for smooth User Experience (UX):

a) For repositories that only provide post-trained weights; Zebra generates a frozen graph before running the demo when required.
b) Provide ability to use videos OR image OR directory_of_images as input.
c) Provide a '`run`' script.

This is a wrapper script that enables all demos to run with similar command line. E.g.:

```
./run <input_source> [--batch B] [--out_file <file>] [--inputSize WxH]
```

- input_source = video file / image file / directory with images / usb cameras
- B = size of batch to use. Default = 1
- out_file = video file to save the output. Default = display output in new window.
- WxH = input image size to Neural Network for inference.
    - Unless the post-trained model has strict restrictions, the input image size can be user defined.

User is encouraged to study the '`run`' script and related application `*.py` code to understand various options.

Table below shows the various demos and associated CNN model along with how to enable Zebra.

| Demo Name | Framework | Supported CNN Models | Command to enable Zebra |
|---|---|---|---|
| darkflow | TF1 | YOLO-v2, TinyYOLO-v2 | `source settings.sh` |
| tensorflow-yolo3 | TF1 | YOLO-v3 | `source settings.sh` |
| tensorflow-yolov4-tflite | TF1 | YOLO-v4, TinyYOLO-v4,<br>YOLO-v3, TinyYOLO-v3 | `source settings.sh` |
| yolov5 | PT | YOLO-v5 N/S/M/L/N6/S6/M6<br>NOTE:<br>This demo supports 7 different models. Models X, L6 and X6 are not supported in this release. | `source settings.sh` |
| Ildoonet-tf-pose-estimation | TF1 | Pose-Estimation | `source settings.sh legacy` |
| EDSR | PT | Super Resolution | `source settings.sh legacy` |

Table below gives list of GitHub source link for the demos:

| Demo Name | GitHub Source Link |
|---|---|
| darkflow | https://github.com/thtrieu/darkflow |
| tensorflow-yolo3 | https://github.com/aloyschen/tensorflow-yolo3 |
| tensorflow-yolov4-tflite | https://github.com/hunglc007/tensorflow-yolov4-tflite |
| yolov5 | https://github.com/ultralytics/yolov5 |
| Ildoonet-tf-pose-estimation | https://github.com/jiajunhua/ildoonet-tf-pose-estimation |
| EDSR | https://github.com/thstkdgus35/EDSR-PyTorch |

### 7.4.1 Input for Docker Demos

Docker demos can accept input in either of the following formats:

- Video file
- Image file
- Directory of Images

Users are encouraged to use the input of their choice. Off course some demos may need appropriate input – for example Pose-Estimation demo needs input where it can detect human pose. Unless specified by the demo, the input can be of variable dimensions.

To make it easier for Users to run the demo on VMAccel Cloud instances, Mipsology provides sample inputs videos. These video files are located inside `~/zebra/examples/VIDEO` directory (let's call it `<VID_DIR>` in table below) which is automatically mounted when starting the docker container.

Table below gives example of command to start the demos.

| Demo Name | Basic Command | Optional Switches |
|---|---|---|
| darkflow | `./run <VID_DIR>/paris_cut1.mkv` | `--batch 8 --out_file YLv2_dk_out.mp4` |
| tensorflow-yolo3 | `./run <VID_DIR>/paris_cut2.mkv` | `--batch 8 --out_file YLv3_tf_out.mp4` |
| tensorflow-yolov4-tflite | `./run <VID_DIR>/paris_cut3.mkv` | `--batch 8 --out_file YLv4_tf_out.mp4` |
| yolov5 | `./run <VID_DIR>/paris_cut2.mkv` | `--batch 8 --out_file YLv5s_tf_out.mp4` |
| Ildoonet-tf-pose-estimation | `./run <VID_DIR>/kulam_dance_27sec.mp4` | `--batch 8 --out_file pose_tf_out.mp4` |

NOTE: the output videos will be lost when docker is closed and when saved inside docker the video cannot be viewed. User can choose to save the video outside docker by mounting a directory when starting the docker container. For e.g.: `./docker/run.bash -v <dir_of choice>:/VIDS`

## 7.5 List of Repositories that are known to work with Zebra

Please refer to Appendix 1 for a list of repositories and neural networks that were tested on Zebra, with notes on limitations or issues.

### Expectation for Future Release

In upcoming releases, Zebra is expected to enable more demos from open-source repositories covering wide variety of CNN models and end applications.

# 8 Release Details

## 8.1 Supported Frameworks and versions

Zebra supports the following frameworks:

| Framework | Recommended version |
|---|---|
| PyTorch | 1.9.0 |
| TensorFlow 1 | 2.8.0 |
| TensorFlow 2 | 2.8.0 |
| ONNX | 1.10.2 (opset 12) |

## 8.2 Release Limitations

### 8.2.1 Layers

In this alpha release:

- MobileNet-v3, EfficientDet and EfficientNet models are not supported.
- Custom layers in middle of NN graph are not supported by the automatic graph split feature.
  - For e.g. FB Detectron2 does not work because the graph includes custom layers.
  - However, in TensorFlow, a custom layer at the beginning or the end of the graph may be supported by execution on CPU.
- Layers that are not rightly supported by ONNX will cause Zebra to return an error when trying to convert the graph.

### 8.2.2 All Frameworks

In this alpha release:

- Some open-source models may experience an error during conversion to ONNX.
  - In such cases, the error message will mention which `opset_version` to use.
  - Please use Zebra's SW API to force this setting and re-run the application. For e.g.:

```
zebra_config --add debug.opset_version=<num_in_error_message>

<run_your_application_again>
```

- When a layer cannot be split into pieces that are smaller than 131072 pixels, an error will happen.
- Inputs wider than 4096 pixels are not supported. Consider doing a manual tiling for those large images.
- Calibration duration is limited by default to avoid wasting resources on wrong use of Zebra. However, on network using very large images or having many layers or very large layers, increasing the calibration time can allow to pass the stage and run. This is done only once and will not impact the eventual performance.
- Input in 8-bit integer format are supported only with legacy mode.

- Neural networks trained with FP16 must be converted into FP32 (changing type) before a quantization can be done. Alternatively, the legacy mode can be used if no automatic split is needed.
- The size of the batch must be constant over an execution
- The size of the input images must be constant over an execution

### 8.2.3 PyTorch

- This alpha release does not support the explicit "`forward`" inference API. For example:
  - The following code will FAIL with Zebra error

```
output = model.forward(input)
```

  - The following code will PASS

```
output = model(input)
```

- Upsample layer with ratio larger than 16 are not supported by Zebra and not automatically mapped on the CPU. Use the manual split for that purpose.
- ConvTranspose layer are not supported if an output padding is used.

### 8.2.4 TensorFlow 1 & 2

- Some demos may experience Zebra error related to automatic graph splitting.
  - We are still working on covering all ways in which TF developers train models and generate graphs.
  - The solution is to use Zebra Software API to 'manually' split the graph (a.k.a. 'Legacy' mode)
  - Details about graph splitting provided in a dedicated section of this document.
- Dilatation with ratio larger than 16 are not supported by Zebra and not automatically mapped on the CPU. Use the manual split for that purpose.
- Floating point NaN values in Python are not automatically converted into 8-bit integer and will result in a conversion error.
- In TensorFlow1, only 1 graph per session is supported. If you have multiple graphs attached to a single session, please use one session per graph and map the right graph to Zebra.

## Expectations for Future Release

In upcoming releases, Zebra is expected to improve automatic graph splitting for all frameworks and support models/graphs with custom layers (i.e. layers not supported by ONNX).

## 8.3 Accelerated layers

The following table summarize the layers that are accelerated on Zebra.

| Layer | Parameter | Values |
|---|---|---|
| 2D Convolution,<br>2D DepthWise Convolution,<br>2D Transpose Convolution,<br>2D Grouped Convolution,<br>2D Dilated Convolution,<br>2D Separable Convolution | Kernel Size | 1 <= W <=64<br>1 <= H <=64<br>D = unlimited;<br>1 <= W*H <= 256 |
| | Stride | 1 <= W <=256<br>1 <= H <= 256 |
| | Dilation | 1 <= dil <= 15 |
| | Padding | 0 <= P <= 15 |
| | Input size | 1 <= W <= 32767<br>1 <= H <= 4095<br>D = unlimited |
| | Groups | 1 <= G <= 2 |
| Max Pooling<br>Average Pooling | Kernel Size | 1 <= W <= 255<br>1 <= H <= 255 |
| | Stride | 1 <= W <=256<br>1 <= H <= 256 |
| | Padding | 0 <= P <= 15 |
| Eltwise Sum | Input size | 1 <= W <= 32767<br>1 <= H <= 4095<br>D = unlimited |
| Concat | | On output channels only<br>Input channels multiple of 4 |
| Reorg | Stride | 1 <= S <= 2 |
| Pad | Input size | 1 <= W <= 32767<br>1 <= H <= 4095<br>D=unlimited |
| | Padding | 1 <= P <= 15 |
| | Value | Constant |
| Global Pool, Mean | | No limit |
| Inner Product | Input size | 1 <= W <= 32767<br>1 <= H <= 4095<br>D = unlimited |
| Fully Connected | Input size | 1 <= W <= 32767<br>1 <= H <= 4095<br>D=unlimited |
| Matmul | Input size | 1 <= W <= 32767<br>1 <= H <= 4095<br>D = unlimited |
| Depth2Space | | |
| Split | Group | G = 2 |
| Crop & Resize | Input size | 1 <= W <= 32767<br>1 <= H <= 4095<br>D = unlimited |
| | Ratio | R = a/b with |

| Layer | Parameter | Values |
|---|---|---|
| | | 1 <= a <= 16<br>1 <= b <= 16<br>Nearest ratio used |
| Upsample | Factor | N = 2 |
| Clip by Value | Input size | 1 <= W <= 32767<br>1 <= H <= 4095<br>D = unlimited |
| BiasAdd | Input size | 1 <= W <= 32767<br>1 <= H <= 4095<br>D = unlimited |
| | Value | Any |
| Batch Normalisation | Input size | 1 <= W <= 32767<br>1 <= H <= 4095<br>D = unlimited |
| | Values | Any |
| Activation Functions | Kind | Relu, LRelu, Relu6, PRelu, Swish, Mish, Sigmoid, Tanh, hard-sigmoid, hard-swish, hard-mish. |
| | Position | Any position in the neural network. |
| Squeeze, Flatten, Reshape,<br>ExpandDims | | Accelerated only if can be merged with following layer |
| Const, Input | | No limitation |

# 9 Performance

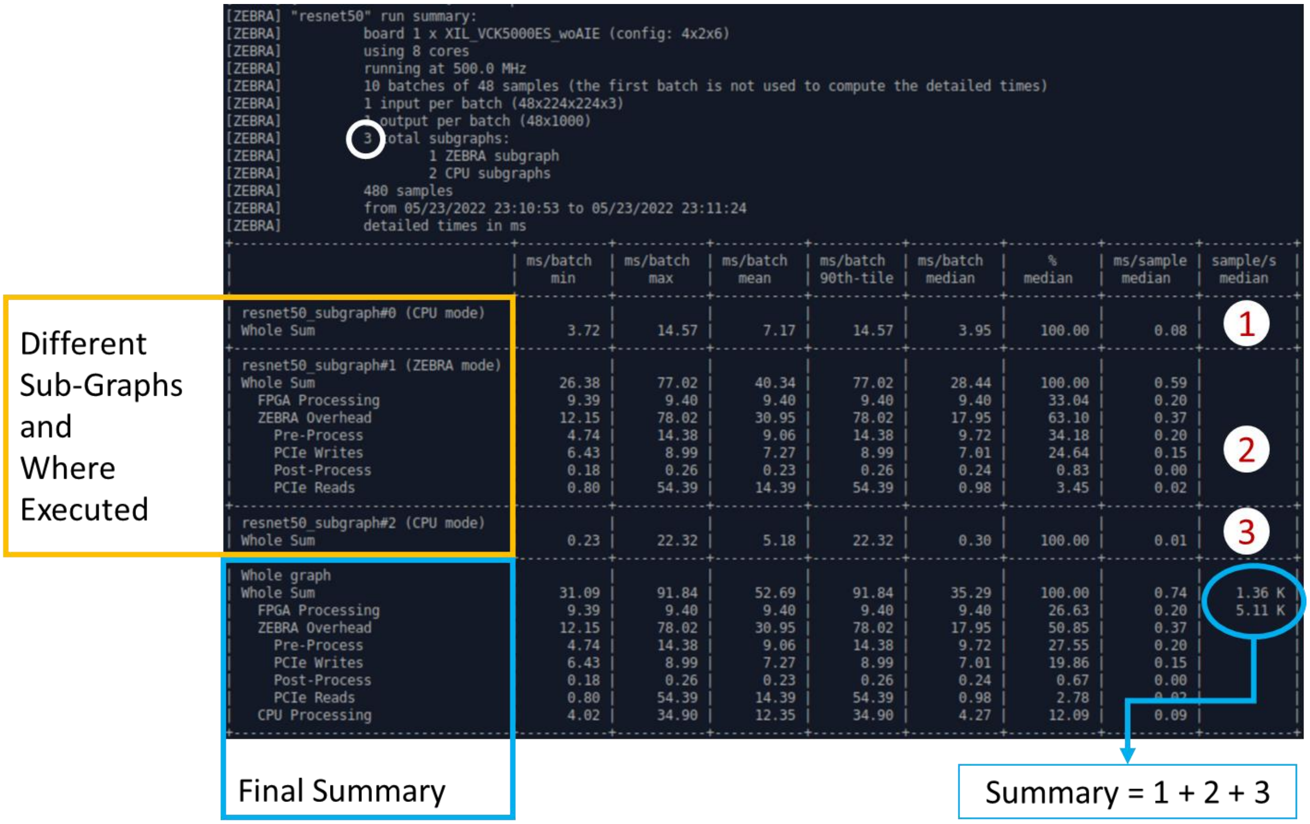
In this Alpha release, performance is medium-to-high depending on CNN. In near future we expect rolling releases/updates with significant performance improvements to all supported CNNs. For example, when targeting ResNet50 with a dedicated configuration (different VM instance available on VMAccel), Zebra today can achieve up to 2x higher FPS (depending on framework and model). And Zebra is expected to improve this number further. Similar increase in performance is expected for all supported CNNs.

Note: When analyzing performance from the summary table printed by Zebra, please note following:

- The main line of interest is the "FPGA" line – this is the CNN inference on FPGA
- Non-FPGA portion (i.e. Software running on CPU) will be pipelined in parallel of the FPGA
  - When fully pipelined, the gap between 'FPGA' and 'Total' FPS will be minimal.

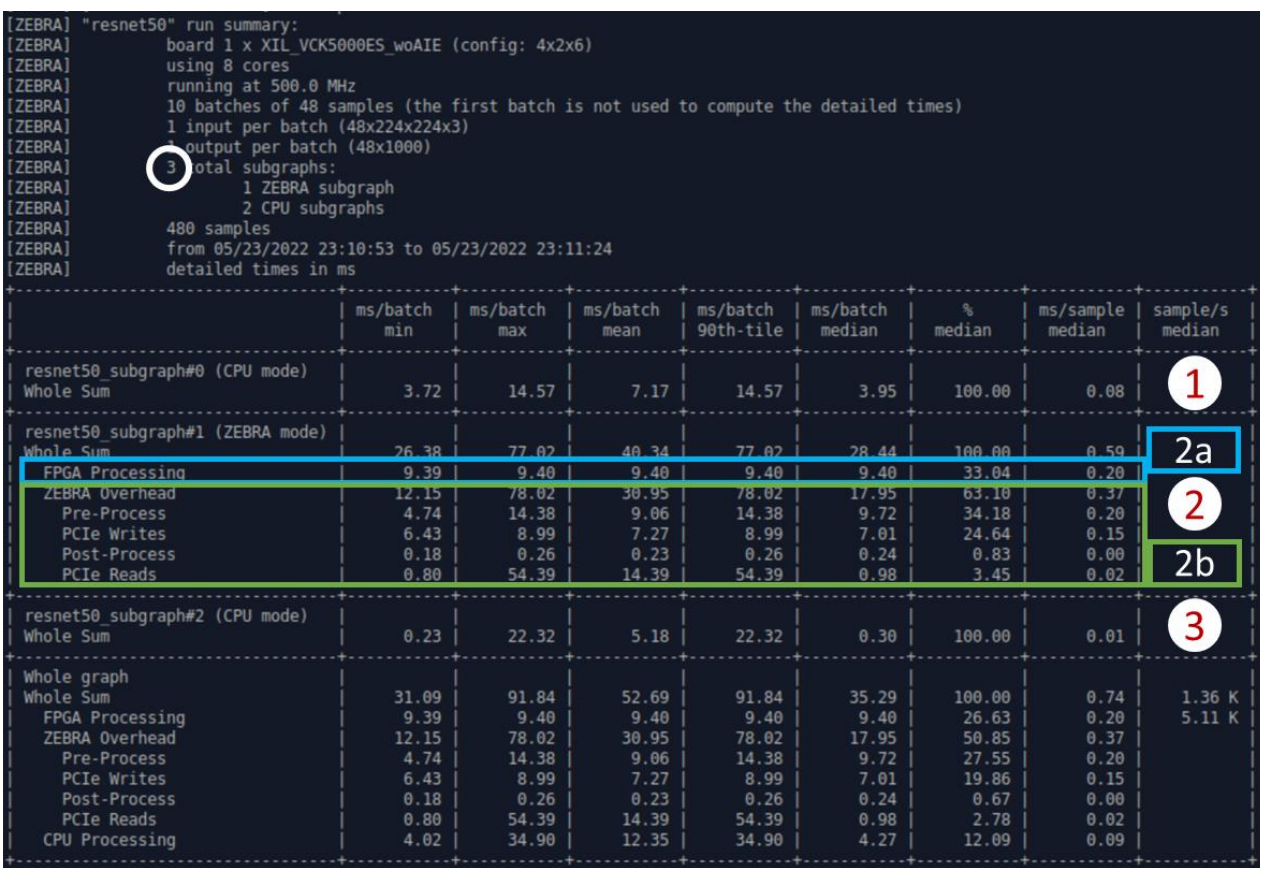
Each customer's requirement will be unique. For questions or concerns, please reach out to Mipsology.

The two figures below show examples of performance summary table printed by Zebra on terminal at the end of the inference execution when statistics are enabled. Note: these figures are for illustrative purposes. Each run's summary table will include details of that CNN execution.




```
[ZEBRA] "resnet50" run summary:
[ZEBRA]          board 1 x XIL_VCK5000ES_woAIE (config: 4x2x6)
[ZEBRA]          using 8 cores
[ZEBRA]          running at 500.0 MHz
[ZEBRA]          10 batches of 48 samples (the first batch is not used to compute the detailed times)
[ZEBRA]          1 input per batch (48x224x224x3)
[ZEBRA]          1 output per batch (48x1000)
[ZEBRA]          3 total subgraphs:
[ZEBRA]                  1 ZEBRA subgraph
[ZEBRA]                  2 CPU subgraphs
[ZEBRA]          480 samples
[ZEBRA]          from 05/23/2022 23:10:53 to 05/23/2022 23:11:24
[ZEBRA]          detailed times in ms
```

| | ms/batch min | ms/batch max | ms/batch mean | ms/batch 90th-tile | ms/batch median | % median | ms/sample median | sample/s median |
|---|---|---|---|---|---|---|---|---|
| resnet50_subgraph#0 (CPU mode) | | | | | | | | |
| Whole Sum | 3.72 | 14.57 | 7.17 | 14.57 | 3.95 | 100.00 | 0.08 | |
| resnet50_subgraph#1 (ZEBRA mode) | | | | | | | | |
| Whole Sum | 26.38 | 77.02 | 40.34 | 77.02 | 28.44 | 100.00 | 0.59 | |
| FPGA Processing | 9.39 | 9.40 | 9.40 | 9.40 | 9.40 | 33.04 | 0.20 | |
| ZEBRA Overhead | 12.15 | 78.02 | 30.95 | 78.02 | 17.95 | 63.10 | 0.37 | |
| Pre-Process | 4.74 | 14.38 | 9.06 | 14.38 | 9.72 | 34.18 | 0.20 | |
| PCIe Writes | 6.43 | 8.99 | 7.27 | 8.99 | 7.01 | 24.64 | 0.15 | |
| Post-Process | 0.18 | 0.26 | 0.23 | 0.26 | 0.24 | 0.83 | 0.00 | |
| PCIe Reads | 0.80 | 54.39 | 14.39 | 54.39 | 0.98 | 3.45 | 0.02 | |
| resnet50_subgraph#2 (CPU mode) | | | | | | | | |
| Whole Sum | 0.23 | 22.32 | 5.18 | 22.32 | 0.30 | 100.00 | 0.01 | |
| Whole graph | | | | | | | | |
| Whole Sum | 31.09 | 91.84 | 52.69 | 91.84 | 35.29 | 100.00 | 0.74 | 1.36 K |
| FPGA Processing | 9.39 | 9.40 | 9.40 | 9.40 | 9.40 | 26.63 | 0.20 | 5.11 K |
| ZEBRA Overhead | 12.15 | 78.02 | 30.95 | 78.02 | 17.95 | 50.85 | 0.37 | |
| Pre-Process | 4.74 | 14.38 | 9.06 | 14.38 | 9.72 | 27.55 | 0.20 | |
| PCIe Writes | 6.43 | 8.99 | 7.27 | 8.99 | 7.01 | 19.86 | 0.15 | |
| Post-Process | 0.18 | 0.26 | 0.23 | 0.26 | 0.24 | 0.67 | 0.00 | |
| PCIe Reads | 0.80 | 54.39 | 14.39 | 54.39 | 0.98 | 2.78 | 0.02 | |
| CPU Processing | 4.02 | 34.90 | 12.35 | 34.90 | 4.27 | 12.09 | 0.09 | |



Total execution time = 1 + 2a + 2b + 3 { 52.69 ms/batch (mean) == 7.17+9.40+30.95+5.18 }

- 2a == ALL / Majority CNN layers accelerated on FPGA
- 1, 2b, 3 == processes executing on CPU

Future Zebra release(s) will automatically pipeline/parallelize 1, 2b and 3

## 10 Accuracy

In this Alpha release, current Zebra delivers good accuracy. However, some CNNs may show larger than expected accuracy drop. It is recommended to compare Zebra accuracy with an inference execution done on CPU/GPU using the same training mode, dataset, image pre-processing, etc. Zebra makes switching between FPGA and CPU/GPU execution very easy : 1-line Linux command to enable and disable Zebra.

In case the inference accuracy is observed to be lower than expected in a default run, Zebra provides SW switches to improve the accuracy (the same FPGA bitstream is used).

Examples of Zebra SW switches/APIs to improve accuracy:

- `quantization.mode=dynamic` (default = constrainedCalibrationV1.5)
- `quantization.forceSatCheckOnLastLayer=false` (default = true)
- `quantization.algorithmVersion=1.0` (default = 3.1)
- `quantization.ignoreNegativeValuesOnLastLayer=false` (default = true)
- `runOptimization.addOptimizers=PrecisionRecovery:RUN`

Example : `zebra_config --add quantization.mode=dynamic`

In our experience, accuracy is a topic that usually attracts intellectual discussions. Please check this FAQ question for more information on understanding accuracy and Zebra Quantization/Calibration.

Note: achieving desired accuracy for CNN Inference is a 1-time R&D effort. Once achieved, Zebra saves the results of quantization/calibration process in a file and always re-use for future execution. The quantization results file can be deployed on target inference servers.

Quantization/Calibration is executed again in event of a change in inputs that can influence the quality of results – for e.g. change in model weights.

Please email Mipsology for any questions or concerns or unique requirements for Your CNN.

# 11 Neural Network (Graph) Management

## 11.1 Automatic splitting of Neural Networks

The default mode of Zebra is to read the full graph including the pre and post processing of images and to automatically create the proper execution on CPU and Zebra. Some pre/post processing cannot be accelerated on Zebra or don't need to be accelerated on Zebra. They are then mapped automatically on the CPU.

In case a layer is not accelerated by Zebra, it will be mapped by Zebra and executed by the framework or by ONNX-runtime. The format conversion between Zebra and the CPU are automatically handled. Therefore, in this mode, user does not need to perform any special modification to run a neural network.

By default, the split of a neural network will be automatically performed by Zebra and without any input from the user. The automatic split may result in:

- One or multiple graphs accelerated by Zebra based on the previously listed accelerated layers.
- A potential pre-processing graph. Typically, a pre-processing can read an image on a disk and do some pre-formatting before a core neural network is applied. Note that some pre-processing may be changed for Zebra to run them, like a resize, a crop or a color operation.
- A potential post-processing graph. Typically, a post-processing formats the output data to be used in a further processing, like a database or to draw boxes on an image.
- Some potential internal graphs kept on the CPU. Typically, this is because some layers are not possible to be accelerated by Zebra in the middle of the graph (for example selecting some parts of the data based on result of a first-level neural network), or because a layer is not supported yet by Zebra, or because the automatic split did not recognize a form of a layer.

Zebra creates a `graph_execution_report.json` file that contain information on how the mapping is performed. This information can be used to inspect the mapping or as a starting point for manual mapping if that is eventually required.

Frameworks offer a lot of flexibility to describe sometimes the same processing. Zebra can recognize layers or sets of layers to be a specific processing that can be accelerated. However, this recognition is not perfect for all forms of the same processing, leading to sub-optimal mapping. This can be fixed by using a manual mapping. In case a layer or structure is not properly recognized by Zebra, or if a layer has a limitation that the automatic mapping does not process yet correctly, a manual mapping can be done as described here after. Examples of such limitations are part of the neural network list provided in the Appendix 1.

Note that if a graph is split in many pieces, the performance can be highly impacted. Mipsology adds regularly new layers and more parameters to the accelerator. If you face a layer that is not accelerated and impact largely the performance, please contact us so we can investigate its acceleration.

## 11.2 Manually splitting a Neural Networks

This section describes how to split a graph between Zebra and the CPU. Zebra will honor user's explicit instructions. However it is possible Zebra may assign more layers to CPU in case some layer cannot be accelerated by Zebra. Alternatively, if a layer is mapped on CPU while it could be accelerated on Zebra, it may result in sub-optimal acceleration.

The execution and data management are performed automatically by Zebra without user intervention based on the provided commands.

### 11.2.1 Graph Splitting

Manual graph splitting is an advanced concept that assumes user is well aware of CNNs and their analysis using framework tools. To use this explicit API, user needs to identify appropriate layers for FPGA or CPU execution and instruct Zebra using software API/command. Zebra will then split the graph as per user directive.

Identification of layer names can be done using Framework APIs or using graphical tools like Netron. Layer names are case sensitive.

Format of the software API/command :

```
zebra_config --add runSession.subGraphs=<MODE>:<endLayer>
```

where:

- <MODE> == ZEBRA or CPU
- <endLayer> == Name of last layer in the model to be executed on MODE

"endLayer" can be composed of multiple layers separated by a ',' when the graph includes multiple branches.

Multiple subgraphs can be declared using the same command with "**|**" (pipe) or "@" to separate each of the subgraph. For example, declaring two subgraphs would look like (the '\' is from a usual shell syntax and is added here for clarity purpose only):

```
zebra_config --add \
runSession.subGraphs="<MODE1>:<endLayer1a,endLayer1b>|<MODE2>:<endLayer2>"
```

or

```
zebra_config --add \
runSession.subGraphs=<MODE1>:<endLayer1a,endLayer1b>@<MODE2>:<endLayer2>
```

Please email Mipsology for any questions around 'manual' graph splitting and use of Legacy mode.

If the last graph goes up to the last layer of the graph, the keyword "AUTO" can be used. Here, AUTO does not mean automatic split but "end of the graph":

```
zebra_config --add runSession.subGraphs=<MODE1><endLayer1a>@<MODE2>:AUTO
```

This command needs to be used only once. Once the syntax is verified, the command is saved in the file named `zebra.ini`, which is used at run time. It is also possible to modify the file.

For example:

```
zebra_config --add runSession.subGraphs= \
   "CPU:layer3a,layer3b,layer4c| \
    ZEBRA:layer73| \
    CPU:AUTO"
```

Means:

- The initial part of the graph is mapped on CPU, with 3 branches in parallel up to layer3a, layer3b and layer3c for the three branches.
- The middle of the graph is mapped on Zebra up to the layer layer73, which assumes that the 3 branches from the CPU execution were merged at some times during the normal processing of the graph.
- The end of the network, from the layer that follows the layer73 is executed on the CPU.

Note that each <MODE>:<layers> means a subgraph. If two subgraphs in a row are described on the same resource, implicit communications and conversions may be applied. For example, if two subgraphs are mapped on Zebra, there will be a communication with CPU added between the subgraphs, which is not optimal.

Please email Mipsology for any questions around 'manual' graph splitting and use of Legacy mode.

### 11.2.2 Zebra Legacy Mode

A legacy mode is included in this release for specific cases. Legacy mode refers to the support of the frameworks prior to the 2022 releases. Legacy mode does not support the automatic split of graphs, only the manual split. It can be used for specific cases like:

- Some specific forms of graphs are not yet properly supported by the latest SW but can be used with manual mapping in legacy mode.
- The training was done using FP16 instead of FP32. Alternatively, you can use the automatic split by converting your FP16-trained network into a FP32 model, that Zebra can process. This step will be automated in future release.
- The input data are provided as 8-bit integers. In many applications, the input data are provided as floating point, but the original type is 8-bit integer. As Zebra performs computation using 8-bit integer, the back-and-forth conversion is detrimental to performance. Using directly int8 data may be possible in some applications. This mode is not supported yet in the latest Zebra SW but can be used in legacy mode.

Please, contact Mipsology if you believe the legacy mode is useful for your application.

## 12 Running Your Neural Network on Zebra

Zebra uses the post-training Neural Network (NN) as-is. Zebra expects the training to be performed in 32-bit Floating Point (FP32) data type. No pruning, re-training, quantization, or any other specific prior operation is expected. Zebra executes within the ML Framework and in-line with the application. No specific software development or proprietary/external tool is required.

Once the NN inference executes successfully on the training hardware (CPU/GPU), switching to Zebra is 1-line Linux command. For example when inside Zebra docker on VMAccel cloud instance:

```
source ~/zebra/settings.sh
```

With this setting, Zebra will automatically intercept NN inference calls, execute the computation on FPGA+CPU and return output data in same format as the application expects. All communication and data conversions are automatically handled by Zebra.

When an application executes for first time, Zebra will perform some one-time initialization operations like neural network mapping on the Zebra accelerator, quantization and calibration of the parameters, optimization of performance, etc. The configuration related to the preparation are saved and reused if further runs are done with the same condition (neural network, weights, options). IF something changes, Zebra will automatically detect changes and re-run the initial operations then save the configuration.

Note that Zebra does not replace nVidia's CUDA but from a user point of view, it offers the same abstraction. So CUDA calls will not be caught by Zebra, but neural network layers in the frameworks will be.

As it is a "plug-and-play" solution, custom and open-source repositories can easily be executed by Zebra. This is one reason we don't provide a model-zoo. However, we strongly advise to check that a repository you plan to use works correctly on CPU before switching to Zebra. Many repositories don't include all elements required to run or their accuracy is wrong. In few cases, some minor modifications need to be done for running on Zebra, we have listed some in the Appendix 1. Please contact us if you think a repository is of general interest so we can add it to our tests.

Zebra software is evolving fast and tested thoroughly. But it is not perfect. Particularly, some ML frameworks contain rich API with many ways to achieve the same purpose. Zebra may not support all API calls. Please email support@mipsology.com for any limitations that prevents to use Zebra with your neural network.

We also advise to run Zebra demos and examples to get familiar with Zebra and its environment.

## 13 FAQ

### Can I control FPGA operating frequency?

On this alpha release for VCK5000, user cannot control FPGA operating frequency. We expect to enable this feature in next release.

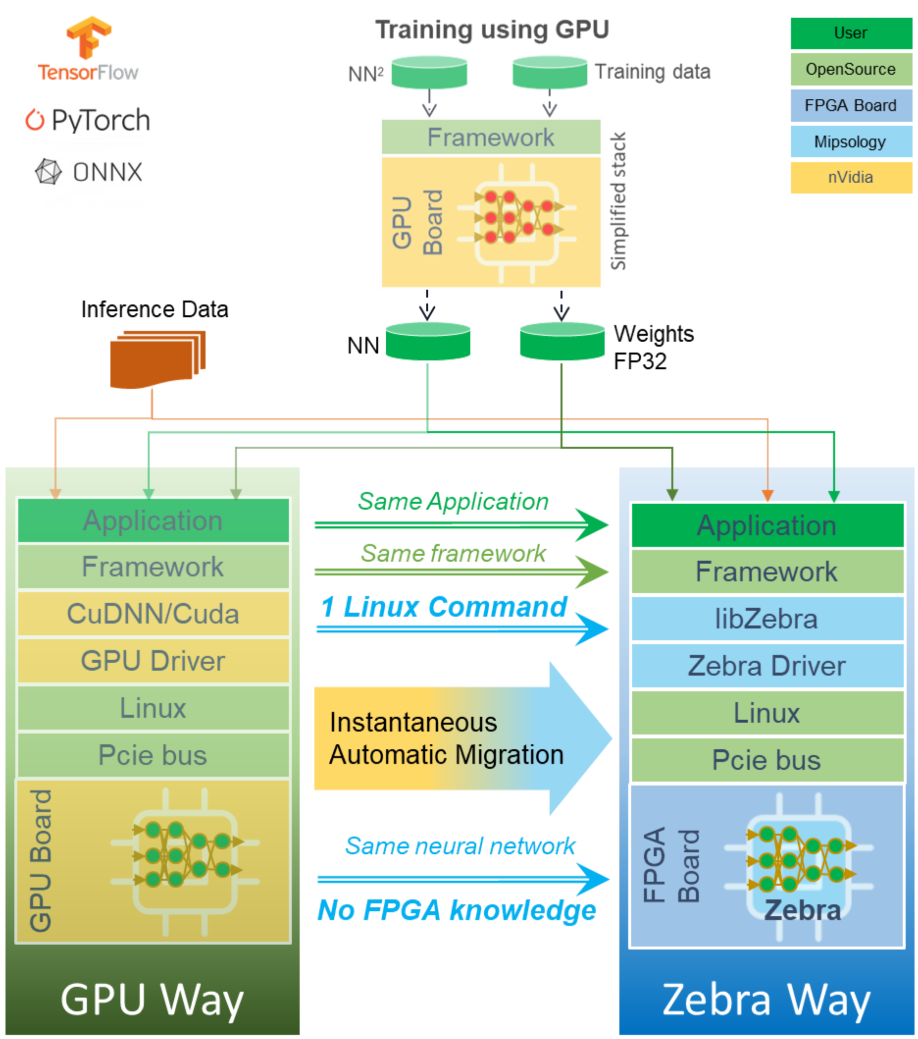
### Why do I get CUDA related messages when running some demos?

Depending on the demo, some CUDA related messages may be printed on terminal by the ML Framework. This should not result in any error during execution. This is not related to Zebra. If absolutely needed, these messages can be suppressed by compiling the framework from source.

### Does Mipsology provide a Model-Zoo?

Mipsology does not provide a model-zoo. This is because Zebra software is designed to accelerate neural networks (CNNs) trained on GPU without modification. In other words, Zebra accelerates post-training CNN graph as-is without any structural change. User does not need to prune the model. There are no offline tools to use before running Zebra.

Zebra works inside User's ML Framework and in-line with User's Application. Figure here shows simplified Zebra software stack. Please reach out to Mipsology for further questions or to discuss project's unique requirements.



### Can You give more information about Accuracy and Zebra Quantization?

High Accuracy for CNN Inference is important for production deployment. Inference accuracy depends on many factors like model training, dataset used, image pre-processing, etc. Based on our experience, it is not a correct practice to compare Zebra result with theoretical accuracy found in an article or on internet. Best practice is to compare results of 2 executions – one with CPU/GPU and one with Zebra on FPGA – using exact same weights/parameters, input data, pre-processing and application software.

Zebra does not need any offline tool for quantization. The process of FP32 to INT8 conversion happens in-line with user's application. From User's point of view, they run the inference application just as they would normally run on CPU/GPU.

Zebra makes switching between FPGA and CPU/GPU very easy – 1-line Linux command : `source settings.sh`.

Zebra aims to provide optimal accuracy by default. In case the accuracy is still observed to be lower than expected, Zebra provides SW switches to improve the accuracy (NOTE: the same FPGA bitstream is used). Some examples of this are shown in Accuracy section.

Most of the software options are influencing the calibration and quantization algorithms, and don't impact performance. The reason different algorithms may be required is that Zebra does not use the training data or expected results to map a model, which sometimes can influence the quality of the results. Typically, the options found for a given model will be reusable if the model goes through various training.

It is also a good practice that the first batch of images, which is used by Zebra for quantization/calibration, are diversified and of good quality. For example:

- Images should cover a good spread of target classes (classification) and objects (detection)
- Not all images should be very dark or very bright
- Not all images expected to give wrong result
- Not all images should be known outliers
- Not all images with extreme size (e.g. largely enlarged)

Users well versed with CNN model and intent of the application typically understand these requirements.

Note that achieving desired accuracy for CNN Inference is a 1-time R&D effort. Once achieved, Zebra saves the results of quantization/calibration process in a file and always re-use for future execution. The quantization results file can be deployed on target inference servers.

Quantization/Calibration is executed again in event of a change in inputs that can influence the quality of results – for e.g. change in model weights.

Please email Mipsology for any questions or concerns or unique requirements for Your CNN.

### How do I contact Mipsology for support or questions?

Please email support@mipsology.com with any questions or concerns or to discuss your unique requirements.

# 14 Appendix 1: List of Tested Neural Network Repositories

The following table provides a list of models that were tested at the time this manual was written. New repositories and models are tested daily by Mipsology to improve the coverage of frameworks and models. In table below:

- Repository: internet address where to find the model. Please consult the license of the repository. Typically, the models can be executed on CPU and/or GPU. Zebra reuses the same post-training neural network and application SW (if available) found in the repositories.
- Neural Network Kind: the name of the neural network or its kind. More details can be found in the repository.
- FWK: ML Framework used to describe the neural network model.
    - PT = PyTorch. TF1 = TensorFlow1. TF2 = TensorFlow2.
- Dataset: dataset used by the repository to our knowledge.
- Graph splitting: type of graph management used when executing on Zebra.
    - Auto: the automatic mode was used.
    - Manual: a manual splitting was done to run this model because of a limitation in the automatic mode. Limitations are addressed regularly in new releases to remove manual splitting.
    - Legacy: Zebra has a legacy software, which has some specific capabilities that may not be yet available with the automatic mode. It is expected that those models are supported in the automatic mode in a future release if not deprecated. See comments for more details.
- Software – from repo: the repository has an application that allows to do the computation. Zebra replaced transparently the CPU/GPU for those computation with a single line command.
    - Pass: functional.
    - Fail: the model fails with the default setup. See comments for more details. In some cases, a minor change or a work-around can be used to make the model functional. In many instances, the neural network is functional, but the application fails for minor issues.
- Software – Zebra App: the repository may not have an application that allows inference execution. Or the neural network was ported to the Zebra application to be easily executed.
- Comment: specific information that are useful to use the repository or model.
    - "Functional but lower-than-expected accuracy" : the model can be executed by Zebra and is known to work with different trainings. However, this specific repository version is not ideal and Zebra reduces accuracy more than what can be expected for such a model. You can use another repository for a similar model or perform your own training. The next version of Zebra software will look to improve accuracy for these cases – including new quantization algorithms – allowing to reduce the accuracy loss further.

Note that Mipsology does not control these open-source repositories and pointers may change at any time.

| Repository | Neural Network Kind | FWK | Dataset | Graph Split | Software | | Comment |
|---|---|---|---|---|---|---|---|
| | | | | | From repo | Zebra SW | |
| https://github.com/osmr/imgclsmob | Resnet18 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | Resnet34 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | Resnet50 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | Resnet101 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | Resnet152 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG11 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG13 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG16 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG19 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | mobileNet_W1 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | mobileNetV2_W1 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | denseNet121 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | inceptionv3 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | inceptionv4 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | xception | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | inceptionresnetv1 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | inceptionresnetv2 | PT | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnet_w18_small_v1 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnet_w18_small_v2 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w18 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w30 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w32 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w40 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w44 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w48 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w64 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hardnet39ds | PT | | manual | pass | | Graph splitting must be done manually to avoid a concat layer limitation |
| https://github.com/osmr/imgclsmob | hardnet68ds | PT | | manual | pass | | |
| https://github.com/osmr/imgclsmob | hardnet68 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hardnet85 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | vovnet27s | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | vovnet39 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | vovnet57 | PT | | auto | pass | | |
| https://github.com/osmr/imgclsmob | simplepose_resnet18_coco | PT | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/osmr/imgclsmob | simplepose_resnet50b_coco | PT | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/osmr/imgclsmob | simplepose_mobile_mobilenet_w1_coco | PT | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/osmr/imgclsmob | simplepose_mobile_mobilenetv2b_w1_coco | PT | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/osmr/imgclsmob | Resnet18 | TF1 | ImageNet | auto | pass | | Requires minor modifications in application:<br>- Graph has no inputs. Application is using an old TF1 API to get inputs without explicit graph connection for inputs.<br><br>Without modification: runs fully on CPU, Zebra does not intercept the graph.<br><br>With modification: functional on Zebra.<br><br>Modification involves addition of explicit input connections. |
| https://github.com/osmr/imgclsmob | Resnet34 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | Resnet50 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | Resnet101 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | Resnet152 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG11 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG13 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG16 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG19 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | mobileNet_W1 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | mobileNetV2_W1 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | DenseNet121 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | Resnet18 | TF2 | ImageNet | auto | pass | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| https://github.com/osmr/imgclsmob | Resnet34 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | Resnet50 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | Resnet101 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | Resnet152 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG11 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG13 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG16 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | VGG19 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | mobileNet_W1 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | mobileNetV2_W1 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | denseNet121 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | denseNet161 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | denseNet169 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | denseNet201 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | inceptionv3 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | inceptionv4 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | xception | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | inceptionresnetv1 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | inceptionresnetv2 | TF2 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnet_w18_small_v1 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnet_w18_small_v2 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w18 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w30 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w32 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w40 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w44 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w48 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hrnetv2_w64 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hardnet39ds | TF2 | | manual | pass | | Graph splitting must be done manually to avoid a concat layer limitation |
| https://github.com/osmr/imgclsmob | hardnet68ds | TF2 | | manual | pass | | Graph spliting must be done manually to avoid a concat layer limitation |
| https://github.com/osmr/imgclsmob | hardnet68 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | hardnet85 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | vovnet27s | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | vovnet39 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | vovnet57 | TF2 | | auto | pass | | |
| https://github.com/osmr/imgclsmob | simplepose_resnet18_coco | TF2 | | auto | | | |
| https://github.com/osmr/imgclsmob | simplepose_resnet50b_coco | TF2 | | auto | | | |
| https://github.com/osmr/imgclsmob | simplepose_resnet101b_coco | TF2 | | auto | | | |
| https://github.com/osmr/imgclsmob | simplepose_resnet152b_coco | TF2 | | auto | | | |
| https://github.com/osmr/imgclsmob | simplepose_resneta50b_coco | TF2 | | auto | | | |
| https://github.com/osmr/imgclsmob | simplepose_resneta101b_coco | TF2 | | auto | fail | | Zebra execution is successful. But application ends with error: conflict of type in Python/numpy ('list' object has no attribute 'numpy').<br><br>The Error is caused by ONNX operations.<br><br>Using different type for outputs in the Python test resolves the issue. |
| https://github.com/osmr/imgclsmob | simplepose_resneta152b_coco | TF2 | | auto | | | |
| https://github.com/osmr/imgclsmob | simplepose_mobile_resnet18_coco | TF2 | | auto | | | |
| https://github.com/osmr/imgclsmob | simplepose_mobile_resnet50b_coco | TF2 | | auto | | | |
| https://github.com/osmr/imgclsmob | simplepose_mobile_mobilenet_w1_coco | TF2 | | auto | | | |
| https://github.com/osmr/imgclsmob | simplepose_mobile_mobilenetv2b_w1_coco | TF2 | | auto | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| https://github.com/osmr/imgclsmob | simplepose_mobile_mobilenetv3_small_w1_coco | TF2 | | auto | | | |
| https://github.com/osmr/imgclsmob | simplepose_mobile_mobilenetv3_large_w1_coco | TF2 | | auto | | | |
| https://github.com/osmr/imgclsmob | mobilenetv3_large_w1 | TF2 | | auto | pass | | Functional but lower-than-expected accuracy |
| https://rwightman.github.io/pytorch-image-models/ | Resnet18 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | Resnet34 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | Resnet50 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | Resnet101 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | Resnet152 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | inception_resnet_v2 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | InceptionV3 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | InceptionV4 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | VGG11 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | VGG13 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | VGG16 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | VGG19 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | mobileNetV2 | PT | ImageNet | auto | pass | | |
| https://rwightman.github.io/pytorch-image-models/ | DenseNet121 | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | Resnet18 | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | Resnet34 | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | Resnet50 | PT | ImageNet | auto | pass | | Inference fails on CPU due to explicit CUDA calls in the application.<br><br>With minor modification of removing the '.cuda' in calls, Inference runs successfully on CPU and Zebra.<br><br>NOTE: Execution is correct in Zebra, however accuracy is not tested by this repository. |
| https://github.com/wang-xinyu/pytorchx | Resnext50 | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | VGG11 | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | mobilenetv2 | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | googlenet | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | InceptionV3 | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | lenet5 | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | mnasnet | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | shufflenet | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | squeezenet | PT | ImageNet | auto | pass | | |
| https://github.com/wang-xinyu/pytorchx | alexnet | PT | ImageNet | auto | pass | | |
| https://github.com/divamgupta/image-segmentation-keras | fcn_8 | TF2 | | auto | pass | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| https://github.com/divamgupta/image-segmentation-keras | fcn_32 | TF2 | | auto | fail | | |
| https://github.com/divamgupta/image-segmentation-keras | fcn_8_vgg | TF2 | | auto | pass | | |
| https://github.com/divamgupta/image-segmentation-keras | fcn_32_vgg | TF2 | | auto | fail | | |
| https://github.com/divamgupta/image-segmentation-keras | fcn_8_resnet50 | TF2 | | auto | pass | | |
| https://github.com/divamgupta/image-segmentation-keras | fcn_32_resnet50 | TF2 | | auto | fail | | Calibration exits due to memory allocation issue. It may pass on host with very large memory. |
| https://github.com/divamgupta/image-segmentation-keras | fcn_8_mobilenet | TF2 | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/divamgupta/image-segmentation-keras | fcn_32_mobilenet | TF2 | | auto | fail | | |
| https://github.com/divamgupta/image-segmentation-keras | vgg_pspnet | TF2 | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/divamgupta/image-segmentation-keras | resnet50_pspnet | TF2 | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/divamgupta/image-segmentation-keras | unet_mini | TF2 | | auto | pass | | |
| https://github.com/divamgupta/image-segmentation-keras | vgg_unet | TF2 | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/divamgupta/image-segmentation-keras | resnet50_unet | TF2 | | auto | pass | | |
| https://github.com/divamgupta/image-segmentation-keras | mobilenet_unet | TF2 | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/divamgupta/image-segmentation-keras | segnet | TF2 | | auto | pass | | |
| https://github.com/divamgupta/image-segmentation-keras | vgg_segnet | TF2 | | auto | pass | | |
| https://github.com/divamgupta/image-segmentation-keras | resnet50_segnet | TF2 | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/divamgupta/image-segmentation-keras | mobilenet_segnet | TF2 | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/divamgupta/image-segmentation-keras | pspnet_ade | TF2 | | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | efficientnet-edgetpu-L | TF1 | ImageNet | auto | pass | | TF application includes graphs/layers not related to the computing NN.<br><br>Need following Zebra option for successful inference execution: "rejectTfRunSession=<LayerName>:0".<br><br>With <LayerName> being one of: Mul, ExpandDims or ExpandDims_1.<br><br>Note: Lower-than-expected accuracy for mobilenetv1 and mobilenetv2. |
| https://github.com/Xilinx/Vitis-AI/ | efficientnet-edgetpu-M | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | efficientnet-edgetpu-S | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | inceptionresnetv2 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | inceptionv1 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | inceptionv2 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | inceptionv3 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | inceptionv4 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | mlperf_resnet50 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | mobilenetEdge0.75 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | mobilenetEdge1.0 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | mobilenetv1_0.25 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | mobilenetv1_0.5 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | mobilenetv1_1.0 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | mobilenetv2_1.0 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | mobilenetv2_1.4 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | resnetv1_101 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | resnetv1_152 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | resnetv1_50 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | resnetv2_101 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | resnetv2_152 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | resnetv2_50 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | vgg16 | TF1 | ImageNet | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | vgg19 | TF1 | ImageNet | auto | pass | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| https://github.com/Xilinx/Vitis-AI/ | refinedet | TF1 | | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | RefineDet-Medical | TF1 | | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | personreid-res50 | PT | | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | personreid-res18 | PT | | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | unet | PT | | auto | pass | | |
| https://github.com/Xilinx/Vitis-AI/ | FairMOT | | | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/jiajunhua/ildoonet-tf-pose-estimation | backbone: CMU | TF1 | | auto | | pass | |
| https://github.com/jiajunhua/ildoonet-tf-pose-estimation | backbone: mobilenet_thin | TF1 | | auto | | pass | |
| https://github.com/jiajunhua/ildoonet-tf-pose-estimation | backbone: mobilenet_v2_small | TF1 | | auto | | pass | |
| https://github.com/jiajunhua/ildoonet-tf-pose-estimation | backbone: mobilenet_v2_large | TF1 | | auto | | fail | Model does not get correctly converted by Zebra. May be supported in a future version. |
| https://github.com/thstkdgus35/EDSR-PyTorch | EDSR | PT | DIV2K | auto | | pass | |
| https://github.com/thstkdgus35/EDSR-PyTorch | EDSR | PT | DIV2K | auto | | pass | |
| https://github.com/thstkdgus35/EDSR-PyTorch | EDSR | PT | DIV2K | legacy | | pass | Legacy mode must be used for models with int8 inputs (instead of FP32).<br>Zebra SW automatic split currently supports NN trained with FP32 for data type. |
| https://github.com/charlesq34/pointnet | pointnet classification | TF1 | ShapeNetPart | manual | | pass | Graph splitting must be done manually to avoid a layer limitation |
| https://github.com/charlesq34/pointnet | pointnet sementic segmentation | TF1 | ShapeNetPart | manual | | pass | Graph splitting must be done manually to avoid a layer limitation |
| https://github.com/thangvubk/FEQE | FEQE | TF1 | DIV2K | manual | | pass | Graph splitting must be done manually to avoid a layer limitation |
| https://github.com/zhixuhao/unet | UNet | TF1 | membrane (isbi challenge) | auto | | pass | |
| https://github.com/ultralytics/yolov5 | YoloV5 | PT | COCO | auto | | pass | Graph correctly executed by Zebra.<br>Implicit PyTorch output type not detected by Zebra, requires explicit type declaration.<br>Passes with minor modification in application to define the correct type. |
| https://github.com/qfgaohao/pytorch-ssd | mobilenetV2 SSD lite | PT | COCO | auto | | pass | |
| https://github.com/qfgaohao/pytorch-ssd | mobilenetV1 SSD | PT | COCO | auto | | pass | |
| https://github.com/tensorflow/tpu/ | EfficientNet | TF1 | ImageNet | manual | | pass | EfficientNet and EfficientDet are not officially supported with this release.<br>Functional but lower-than-expected accuracy |
| https://github.com/osmr/imgclsmob | VGG16 | TF1 | ImageNet | auto | pass | | |
| https://github.com/osmr/imgclsmob | mobileNetV1 | PT | ImageNet | auto | pass | | |
| https://github.com/aloyschen/tensorflow-yolo3 | YoloV3 | TF1 | COCO | auto | | pass | |
| https://github.com/thunil/TecoGAN | TEcoGAN | TF1 | | legacy | | accuracy | Legacy mode must be used for this model. |
| https://github.com/marvis/pytorch-yolo2 (removed on github) | yolov2 | PT | COCO | auto | | pass | |
| https://github.com/hellochick/ICNet-tensorflow | ICNet | TF1 | Cityscape | auto | | pass | |
| https://github.com/matterport/Mask_RCNN | Mask_RCNN | TF1 | COCO | legacy | | pass | Legacy mode must be used for this model. |
| https://github.com/DevKiHyun/VDSR-Tensorflow | VDSR | TF1 | | auto | | pass | |
| https://github.com/kcosta42/Tensorflow-YOLOv3 | YOLOv3 | TF1 | COCO | manual | | pass | |
| https://github.com/longcw/yolo2-pytorch | YoloV2 | PT | COCO | auto | | pass | |
| https://github.com/twhui/SRGAN-PyTorch | SRResnet | PT | ImageNet | auto | | pass | |

Mipsology

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| https://github.com/twhui/SRGAN-PyTorch | SRGAN | PT | ImageNet | auto | | pass | |
| https://github.com/milesial/Pytorch-Unet | UNet | PT | Carvana | auto | | pass | |
| https://github.com/ultralytics/yolov5/releases | yolov5n.pt | PT | COCO | auto | pass | | |
| https://github.com/ultralytics/yolov5/releases | yolov5n.pb | TF1 | COCO | auto | fail | | Model currently failing for a problem of tensor conversion in Python. |
| https://github.com/ultralytics/yolov5/releases | yolov5n-fp16 | TF1 | COCO | auto | fail | | Zebra SW does not convert automatically FP16 into int8 for quantization.<br>This can be worked around by converting the FP16 models into FP32 prior to running in Zebra. |
| https://github.com/hunglc007/tensorflow-yolov4-tflite | YoloV3 | TF1 | COCO | auto | pass | | Functional on Zebra when using proper method:<br>- exporting the graph, as indicated in the repository, must be done using CPU, prior to enable Zebra. |
| https://github.com/hunglc007/tensorflow-yolov4-tflite | tinyYoloV3 | TF1 | COCO | auto | pass | | |
| https://github.com/hunglc007/tensorflow-yolov4-tflite | YoloV4 | TF1 | COCO | auto | pass | | |
| https://github.com/hunglc007/tensorflow-yolov4-tflite | tinyYoloV4 | TF1 | COCO | auto | pass | | |
| https://github.com/Megvii-BaseDetection/YOLOX | Yolo-X | PT | COCO | auto | pass | | Fully functional on Zebra.<br>However, the application is designed to run a profiling. This profiling is done with random weights, which forces Zebra to execute a useless quantization at start. The proper weights are quantized once and reloaded. Removing the profiling reduces the launch time. |
| https://github.com/WongKinYiu/yolor | YoloR | PT | | auto | pass | | The application does not resize automatically images to the same size. This repo is functional if all images computed are of the same size. |
| https://github.com/uvipen/SSD-pytorch | SSD ResNet50 | PT | COCO | auto | pass | | |
| https://github.com/dd604/refinedet.pytorch | refinedet resnet101_320 | PT | | auto | pass | | |
| https://github.com/dd604/refinedet.pytorch | refinedet resnet101_512 | PT | | auto | pass | | |
| https://github.com/dd604/refinedet.pytorch | refinedet_vgg16* | PT | | auto | fail | | ConvTranspose layer of PyTorch is not supported when output padding is used. Can be used with manual mapping those layers on CPU. |
| https://github.com/ultralytics/yolov3 | yolov3 | PT | COCO | auto | pass | | |
| https://github.com/ultralytics/yolov3 | yolov3_fixed | PT | COCO | auto | pass | | |
| https://github.com/ultralytics/yolov3 | yolov3_tiny | PT | COCO | auto | pass | | |
| https://github.com/ultralytics/yolov3 | yolov3_spp | PT | COCO | auto | pass | | |
| https://github.com/aloyschen/tensorflow-yolo3 | YoloV3 | TF1 | COCO | auto | pass | | |
| https://github.com/KleinYuan/tf-object-detection | ssdMobileNetV1 | TF1 | COCO | auto | pass | | |
| https://github.com/KleinYuan/tf-object-detection | ssdInceptionV2 | TF1 | COCO | auto | pass | | |
| https://github.com/thtrieu/darkflow | YoloV2-tiny frozen | TF1 | COCO | auto | pass | | |
| https://github.com/thtrieu/darkflow | YoloV2 frozen | TF1 | COCO | auto | pass | | |
| https://github.com/milesial/Pytorch-Unet | Unet scale 0.5 | PT | Carvana | auto | pass | | This repo does not test accuracy |
| https://github.com/milesial/Pytorch-Unet | UNet scale 1 | PT | Carvana | auto | pass | | This repo does not test accuracy |
| https://github.com/jiajunhua/ildoonet-tf-pose-estimation | backbone: CMU | TF1 | | auto | pass | | |
| https://github.com/jiajunhua/ildoonet-tf-pose-estimation | backbone: mobilenet_thin | TF1 | | auto | pass | | Functional but lower-than-expected accuracy |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| https://github.com/jiajunhua/ildoonet-tf-pose-estimation | backbone: mobilenet_v2_small | TF1 | | auto | fail | | Application does not work automatically with Zebra as some values in Python are float NaN, which cannot be quantized into int8 |
| https://github.com/jiajunhua/ildoonet-tf-pose-estimation | backbone: mobilenet_v2_large | TF1 | | auto | | | |
| https://github.com/barisbatuhan/FaceDetector | FaceDetector wf_r50 | PT | | auto | pass | | |
| https://github.com/barisbatuhan/FaceDetector | FaceDetector icf_r50 | PT | | auto | pass | | |
| https://github.com/barisbatuhan/FaceDetector | FaceDetector mixed_r50 | PT | | auto | pass | | |
| https://github.com/barisbatuhan/FaceDetector | FaceDetector mixed_r152 | PT | | auto | pass | | |
| https://github.com/mikel-brostrom/Yolov5_DeepSort_Pytorch | Yolov5_DeepSort | PT | | auto | pass | | Application working properly on Zebra with a minor modification in application SW:<br>- forced fixed batch-size for tracking,<br>- removing calibration of random weights done at each launch for some metric measurement makes launch faster. |
| https://github.com/clovaai/CRAFT-pytorch | craft mlt_25k | PT | SynthText | auto | pass | | |
| https://github.com/clovaai/CRAFT-pytorch | craft ic15_20k | PT | SynthText | auto | pass | | Functional but lower-than-expected accuracy |
| https://github.com/clovaai/CRAFT-pytorch | craft refiner_CTW1500 | PT | SynthText | auto | fail | | Test fails on ONNX conversion |
| Model provided in the Zebra examples | inceptionv2 | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | inceptionv3 | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | inceptionv4 | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | googlenet_no_lrn | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | googlenet | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | caffenet | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | vgg16 | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | vgg19 | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | nin | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | car_nin | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | resnet50 | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | resnet50-V1.5 | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | resnet50_reduce_mean | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | resnet152 | TF1 | ImageNet | auto | | pass | |
| Model provided in the Zebra examples | caffenet_no_lrn | TF1 | ImageNet | auto | | pass | |
| http://download.tensorflow.org/models/mobilenet_v1_2018_08_02/mobilenet_v1_1.0_224.tgz | mobilenet_v1 | TF1 | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://storage.googleapis.com/mobilenet_v2/checkpoints/mobilenet_v2_1.4_224.tgz | mobilenet_v2 | TF1 | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| Model provided in the Zebra examples | yolov1 | TF1 | PascalVOC | auto | | pass | |
| Model provided in the Zebra examples | yolov2 | TF1 | COCO | auto | | pass | |
| Model provided in the Zebra examples | yolov3 | TF1 | COCO | auto | | pass | |
| Model provided in the Zebra examples | edsr_x2 | TF1 | | auto | | pass | |
| https://tfhub.dev/google/imagenet | mobilenet_v1 | TF2 | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://tfhub.dev/google/imagenet | mobilenet_v2 | TF2 | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://keras.io/api/applications | VGG16 | TF2 | ImageNet | auto | | pass | |
| https://keras.io/api/applications | VGG19 | TF2 | ImageNet | auto | | pass | |
| https://tfhub.dev/google/imagenet | inceptionv1 | TF2 | ImageNet | auto | | pass | |
| https://tfhub.dev/google/imagenet | inceptionv2 | TF2 | ImageNet | auto | | pass | |
| https://tfhub.dev/google/imagenet | inceptionv3 | TF2 | ImageNet | auto | | pass | |
| https://keras.io/api/applications | xception | TF2 | ImageNet | auto | | pass | |
| https://tfhub.dev/google/imagenet | inception_resnet_v2 | TF2 | ImageNet | auto | | pass | |
| https://tfhub.dev/google/imagenet | resnet50 | TF2 | ImageNet | auto | | pass | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| https://tfhub.dev/google/imagenet | resnet50v1 | TF2 | ImageNet | auto | | pass | |
| https://tfhub.dev/google/imagenet | resnet101 | TF2 | ImageNet | auto | | pass | |
| https://tfhub.dev/google/imagenet | resnet152 | TF2 | ImageNet | auto | | pass | |
| https://tfhub.dev/google/imagenet | resnet50v2 | TF2 | ImageNet | auto | | pass | |
| https://tfhub.dev/google/imagenet | resnet101v2 | TF2 | ImageNet | auto | | pass | |
| https://tfhub.dev/google/imagenet | resnet152v2 | TF2 | ImageNet | auto | | pass | |
| https://keras.io/api/applications/ | densenet121 | TF2 | ImageNet | auto | | pass | |
| https://keras.io/api/applications/ | densenet169 | TF2 | ImageNet | auto | | pass | |
| https://keras.io/api/applications/ | densenet201 | TF2 | ImageNet | auto | | pass | |
| https://github.com/ultralytics/yolov5.git | yolov5n | TF2 | COCO | auto | | pass | |
| https://github.com/ultralytics/yolov5.git | yolov5s | TF2 | COCO | auto | | pass | |
| https://github.com/ultralytics/yolov5.git | yolov5m | TF2 | COCO | auto | | pass | |
| https://github.com/ultralytics/yolov5.git | yolov5l | TF2 | COCO | auto | | pass | |
| https://github.com/ultralytics/yolov5.git | yolov3 | TF2 | COCO | auto | | pass | |
| https://github.com/ultralytics/yolov5.git | yolov3-spp | TF2 | COCO | auto | | pass | |
| https://pytorch.org/hub/ | Resnet50 | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | alexnet | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | googlenet_no_lrn | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | inceptionv3 | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | Resnet18 | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | Resnet34 | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | Resnet101 | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | Resnet152 | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | resnext50_32x4d | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | resnext101_32x8d | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | wide_resnet50_2 | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | wide_resnet101_2 | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | shufflenet_v2_x0_5 | PT | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://pytorch.org/hub/ | shufflenet_v2_x1_0 | PT | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://pytorch.org/hub/ | squeezenet | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | squeezenet1_1 | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | VGG11 | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | VGG11_bn | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | VGG13 | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | VGG13_bn | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | VGG16 | PT | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://pytorch.org/hub/ | VGG16_bn | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | VGG19 | PT | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://pytorch.org/hub/ | VGG19_bn | PT | ImageNet | auto | | pass | |
| https://pytorch.org/hub/ | mobilenet_v2 | PT | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://pytorch.org/hub/ | densenet121 | PT | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://pytorch.org/hub/ | densenet161 | PT | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://pytorch.org/hub/ | densenet169 | PT | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://pytorch.org/hub/ | densenet201 | PT | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://pytorch.org/hub/ | mobilenet_v3_small | PT | ImageNet | auto | | fail | These networks are not supported with current Release |
| https://pytorch.org/hub/ | mobilenet_v3_large | PT | ImageNet | auto | | | |
| https://pytorch.org/hub/ | mnasnet0_5 | PT | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |
| https://pytorch.org/hub/ | mnasnet1_0 | PT | ImageNet | auto | | pass | Functional but lower-than-expected accuracy |

# Legal Notice

The information disclosed to you (the “User”) hereunder is provided solely for the selection and use of Mipsology products. The information, the applications, and the tools (together referred as the “Materials”) are made available “AS IS” and with all mistakes, errors, inconsistencies, or defects, without warranty of any kind. To the maximum extent permitted by applicable law:

(1) The Materials are provided "as is" and "as available", without warranty of any kind. Mipsology, its affiliates, its officers, its employees, or its suppliers and representatives, do not warrant in any way that the Materials is error free or satisfy licensee's specific requirements and disclaim any and all warranties of any kind or nature, whether express, implied, or statutory, relating to or arising with respect to the Materials, including but not limited to implied warranties of merchantability, warranty of fitness for a particular purpose, title, and noninfringement. Mipsology makes no warranty concerning the data, results or information resulting in using the Materials.

(2) To the maximum extent permitted by applicable law, in no event shall Mipsology, its affiliates, its officers, its employees, or its suppliers and representatives be liable for any special, exemplary, consequential, incidental, punitive, direct or indirect damages whatsoever including, but not limited, to loss of business profit, loss of use, loss of data, business interruption, loss of revenue, loss of orders, loss of business or profits, anticipated savings, loss of information and data, damage to brand image, or any other financial loss arising out of or in connection with the use of the Materials or the operation of the application or any other product or services, even if advised beforehand of the possibility of such damages. In no event will Mipsology total liability under or arising out of this agreement exceed the actual received payment from User, directly or through the cloud, in the last billing period or the duration of the incident, whichever is the lowest amount, reduced by any other amount Mipsology would have paid back to User. To the extent that the applicable jurisdiction limits licensee's ability to disclaim any implied warranties, this disclaimer shall be effective to the maximum extent permitted. Without limiting the foregoing, the User is responsible for determining and verifying that the Materials, its environment, and the hardware used to run the application are compatible. Mipsology further decline any warranties of any kind or nature on the hardware used in conjunction with the Materials. Mipsology shall not be liable to User nor any third parties (whether arising in contract, tort (including negligence), breach of statutory duty or otherwise) for failure of fitness or any of its or a third party's systems that results in the inability to process or use the Material, User's failure to meet any of its payment obligations, negligence, fraud or fraudulent misrepresentation of User or any other actions which result from misuse or inappropriate use of the Materials.

Without prior written agreement, User will not knowingly, or allow others, including internally, to copy, reproduce, modify, obliterate, distribute, or publicly display the Materials in any form, partially or fully, whatsoever except for the normal usage of the Materials.

Mipsology assumes no obligation to correct any errors contained in the Materials or to notify User of updates to the Materials. This document is subject to change without notice.

Please refer to Mipsology’s End User License Agreement (EULA.txt) and other legal notices available in the ‘doc’ directory of the provided release.

**Copyright**