





[Sign up](#)[Xilinx](#) / [Vitis-AI](#)[Code](#)[Issues](#) 121[Pull requests](#) 18[Actions](#)[Projects](#)[Security](#)

master ▾

[Vitis-AI](#) / [models](#) / [AI-Model-Zoo](#) /[History](#)

..

 <a href="#">caffe-xilinx</a>	Dec 16, 2020
 <a href="#">images</a>	Dec 16, 2020
 <a href="#">model-list</a>	Mar 26, 2021
 <a href="#">LICENSE</a>	Dec 16, 2020
 <a href="#">README.md</a>	Mar 4, 2021



README.md



## Vitis AI Model Zoo

### Introduction

This repository includes optimized deep learning models to speed up the deployment of deep learning inference on Xilinx™ platforms. These models cover different applications, including but not limited to ADAS/AD, medical, video surveillance, robotics, data center, etc. You can get started with these free pre-trained models to enjoy the benefits of deep learning acceleration.



## 🔗 Vitis AI 1.3 Model Zoo New Features !

1.New added 28 models. Total 92 models from different deep learning frameworks, Caffe, TensorFlow, TensorFlow 2 and PyTorch are supported in Vitis AI Model Zoo.

2.The variety of Model Zoo has been significantly improved for a wider range of applications.

For medical applications, we added CT image segmentation, medical robot instrument segmentation, Covid-19 chest radiograph segmentation and other reference models. For autonomous driving and ADAS applications, we added 3D point cloud detection and point cloud segmentation models.

3.Provided accuracy evaluation and quantization scripts for all the released models.

4.Restructured model list show all versions of models that running on all supported Xilinx platforms in a clearer and more explicit way, and users could download the specific version they need for every model.

## 🔗 Model Details

The following table includes comprehensive information about each model, including application, framework, training and validation dataset, backbone, input size, computation as well as float and quantized precision.

► [Click here to view details](#)

## 🔗 Naming Rules

Model name: `F_M_(D)_H_W_(P)_C_V`

- `F` specifies training framework: `cf` is Caffe, `tf` is Tensorflow, `tf2` is Tensorflow 2, `dk` is Darknet, `pt` is PyTorch
- `M` specifies the model
- `D` specifies the dataset. It is optional depending on whether the dataset is public or private
- `H` specifies the height of input data
- `W` specifies the width of input data
- `P` specifies the pruning ratio, it means how much computation is reduced. It is optional depending on whether the model is pruned or not
- `C` specifies the computation of the model: how many Gops per image
- `V` specifies the version of Vitis AI

For example, `cf_refinedet_coco_360_480_0.8_25G_1.3` is a `RefineDet` model trained with `Caffe` using `coco` dataset, input data size is `360*480`, `80%` pruned, the computation per image is `25Gops` and Vitis AI version is `1.3`.

## 🔗 caffe-xilinx

This is a custom distribution of caffe. Please use [caffe-xilinx](#) to test and finetune the caffe models listed in this page.

## 🔗 Model Download

Please visit [model-list](#) in this page. You will get downloadlink and MD5 of all the released models, including pre-compiled models running on different platforms.

## 🔗 Model Directory Structure

Download and extract the model archive to your working area on the local hard disk. For details on the various models, their download link and MD5 checksum for the zip file of each model, see [model-list](#).

## 🔗 Caffe Model Directory Structure

For a caffe model, you should see the following directory structure:

```
|— code                                     # Contains test , training and quantization s
|
|
```

— readme.md	# Contains the environment requirements, data Refer this to know that how to test and tra
— data	# Contains the dataset that used for model te When test or training scripts run successfu In some cases, you may also need to manuell
— quantized	
— deploy.caffemodel	# Quantized weights, the output of vai_
— deploy.prototxt	# Quantized prototxt, the output of vai_
— quantize_test.prototxt	# Used to run evaluation with quantize_ Some models don't have this file if t Pytorch (ReID) or there is no Caffe T
— quantize_train_test.caffemodel	# Quantized weights can be used for qua
— quantize_train_test.prototxt	# Used for quantized-point training and on GPU when datalayer modified to use
— DNNC	
— deploy.prototxt	# Quantized prototxt for dnnc. It's the
— float	
— trainval.caffemodel	# Trained float-point weights.
— test.prototxt	# Used to run evaluation with python te
— trainval.prortotxt	# Used for training and testing with ca when datalayer modified to user's dat have this file if they are converted Pytorch (ReID) or there is no Caffe T

## 🔗 Tensorflow Model Directory Structure

For a Tensorflow model, you should see the following directory structure:

— code	# Contains test code which can run demo and e
— readme.md	# Contains the environment requirements, data Refer this to know that how to test the mod
— data	# Contains the dataset that used for model te When test or training scripts run successfu
— quantized	
— deploy.model.pb	# Quantized model for the compiler (extended
— quantize_eval_model.pb	# Quantized model for evaluation.
— float	
— frozen.pb	# Float-point frozen model, the input to the The pb name of different models may be diff

## Pytorch Model Directory Structure

For a Pytorch model, you should see the following directory structure:

```
├── code                                # Contains test and training code.
├── readme.md                          # Contains the environment requirements, data
                                      # Refer this to know that how to test and tra
├── data                              # Contains the dataset that used for model te
                                      # When test or training scripts run successfu
├── quantized
│   ├── bias_corr.pth                 # Quantized model.
│   ├── quant_info.json               # Quantization steps of tensors got. Please k
│   ├── _int.py                       # Converted vai_q_pytorch format model.
│   └── _int.xmodel                   # Deployed model. The name of different model
└── float
    └── _int.pth                      # Trained float-point model. The.pth name of
                                      # Path and name in test scripts could be modi
```

**Note:** For more information on `vai_q_caffe` and `vai_q_tensorflow`, see the [Vitis AI User Guide](#).

## Model Performance

All the models in the Model Zoo have been deployed on Xilinx hardware with [Vitis AI](#) and [Vitis AI Library](#). The performance number including end-to-end throughput and latency for each model on various boards with different DPU configurations are listed in the following sections.

For more information about DPU, see [DPU IP Product Guide](#).

**Note:** The model performance number listed in the following sections is generated with Vitis AI v1.3 and Vitis AI Library v1.3. For different boards, different DPU configurations are used. Vitis AI and Vitis AI Library can be downloaded for free from [Vitis AI Github](#) and [Vitis AI Library Github](#). We will continue to improve the performance with Vitis AI. The performance number reported here is subject to change in the near future.

## Performance on ZCU102 (0432055-04)

This version of ZCU102 is out of stock. The performance number shown below was measured with the previous AI SDK v2.0.4. Now this form has stopped updating.

► [Click here to view details](#)

### ↻ Performance on ZCU102 (0432055-05)

Measured with Vitis AI 1.3 and Vitis AI Library 1.3

► [Click here to view details](#)

### ↻ Performance on ZCU104

Measured with Vitis AI 1.3 and Vitis AI Library 1.3

► [Click here to view details](#)

### ↻ Performance on U50

Measured with Vitis AI 1.3 and Vitis AI Library 1.3

► [Click here to view details](#)

### ↻ Performance on U50 lv9e

Measured with Vitis AI 1.3 and Vitis AI Library 1.3

► [Click here to view details](#)

### ↻ Performance on U50 lv10e

Measured with Vitis AI 1.3 and Vitis AI Library 1.3

► [Click here to view details](#)

### ↻ Performance on U200

Measured with Vitis AI 1.3 and Vitis AI Library 1.3

► [Click here to view details](#)

### ↻ Performance on U250

Measured with Vitis AI 1.3 and Vitis AI Library 1.3

► [Click here to view details](#)

## 🔗 Performance on U280

Measured with Vitis AI 1.3 and Vitis AI Library 1.3

► [Click here to view details](#)

## 🔗 Performance on VCK190

Measured with Vitis AI 1.3 and Vitis AI Library 1.3

► [Click here to view details](#)

## 🔗 Performance on Ultra96

The performance number shown below was measured with the previous AI SDK v2.0.4 on Ultra96 v1. The Vitis platform of Ultra96 v2 has not been released yet. So the performance numbers are therefore not reported for this Model Zoo release.

► [Click here to view details](#)

## 🔗 Contributing

---

We welcome community contributions. When contributing to this repository, first discuss the change you wish to make via:

- [GitHub Issues](#)
- [Forum](#)
- [Email](#)

You can also submit a pull request with details on how to improve the product. Prior to submitting your pull request, ensure that you can build the product and run all the demos with your patch. In case of a larger feature, provide a relevant demo.

## 🔗 License

---

Xilinx AI Model Zoo is licensed under [Apache License Version 2.0](#). By contributing to the project, you agree to the license and copyright terms therein and release your contribution under these terms.

