



Novaic Convertor User Guide

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

Table of Content

Novaic Convertor User Guide	1
Table of Content.....	2
1 Introduction	4
1.1 Introduction of Flow	4
1.2 Required Environment.....	5
1.3 Installation	7
1.3.1 Installation of Docker.....	7
1.3.2 Use Docker Image at First Time	8
1.3.3 Installation of Tool Package.....	8
2 Support Function	10
2.1 Support Framework.....	10
2.2 Verified Networks	10
2.3 Layer Fusion.....	14
2.4 Support Layer List	15
2.4.1 Limitation Table of Tool.....	19
2.4.2 Limitation Table of HW	19
3 Gen-tool Introduction	20
3.1 Config Setting.....	20
3.2 Input Data.....	41
3.3 Output Data	43
4 Sim-tool Introduction	44
4.1 Config Setting.....	44
4.2 Input Data.....	50
4.3 Output Data	52
5 Example Introduction	55
5.1 Preprocess of Configure Files	55
5.2 ONNX Model Conversion	56
5.2.1 Pytorch to ONNX.....	57

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

5.2.2	ONNX to ONNX	59
5.3	Run Gen-tool Program	60
5.4	Run Sim-tool Program	63
6	The Method of Adding Custom Layer	66
6.1	The Process of Setting Up A Custom Layer (Caffe)	66
6.1.1	Caffe.....	66
6.1.2	Gen-tool	66
6.1.3	Sim-tool	74
6.1.4	AI2 SDK	80
6.2	The Process of Setting Up A Custom Layer (ONNX).....	82
6.2.1	ONNX.....	82
6.2.2	Gen-tool	85
6.2.3	Sim-tool	91
6.2.4	AI2 SDK	91
7	The usage of DSP ACL.....	92
8	Network Adjustment Suggestions	95
8.1	Precision Adjustment.....	95
8.2	Automatic Network Correction.....	101
8.3	Network Efficiency.....	103
8.4	Quantized Param. importment.....	111
9	Tool Debugging.....	115
9.1	LOG Printing Specification	115
9.1.1	Compiler.....	115
9.1.2	Simulator	116
10	Revision History.....	117

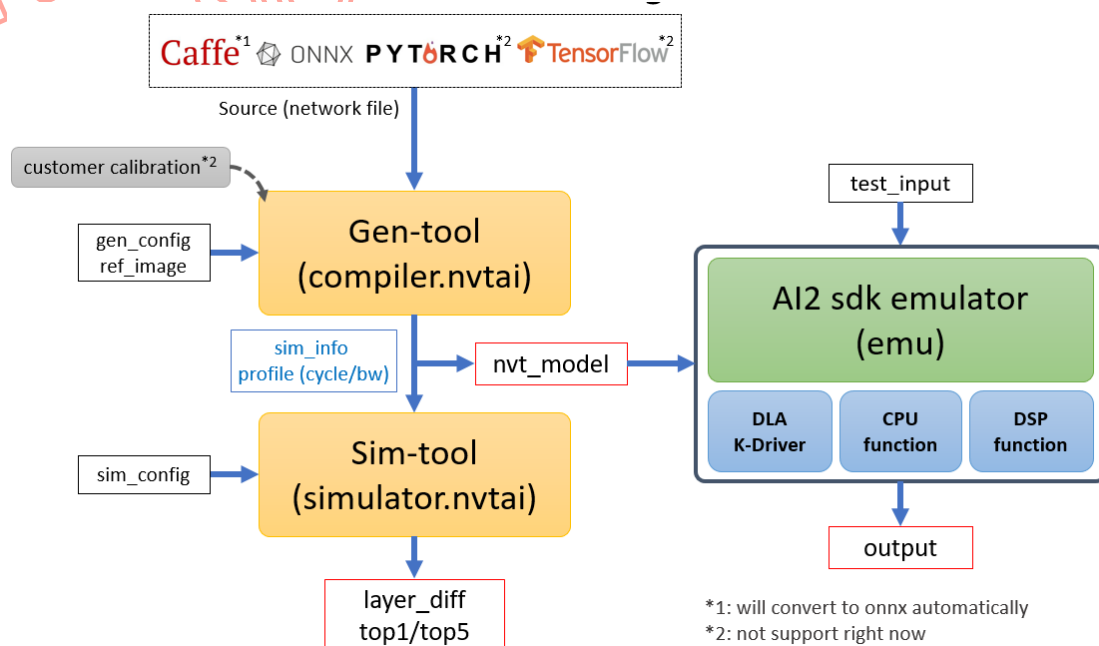
1 Introduction

1.1 Introduction of Flow

Novaic Convertor adopts a new compiler architecture to accommodate more frameworks to convert Caffe/Tensorflow/Pytorch/ONNX trained models into binary files (nvt_model.bin) for intelligent applications of deep learning networks in 32bit/64bit environments. This version is developed in C/C++ language, which greatly improves the portability of the platform.

Novaic Convertor currently contains two modules:

1. Gen-tool: Converts the trained model of Caffe/Tensorflow/Pytorch/ONNX to a binary loadable file (nvt_model.bin) °
2. Sim-tool: Prepares the test data and input the binary file about hardware parameter (e.g. nvt_model.bin) generated by Gen-tool to simulate its effect, and then compare it with the result of runtime to see if it is correct. If it is incorrect, the Layer_diff info can provide be used as a basis for adjusting parameters.

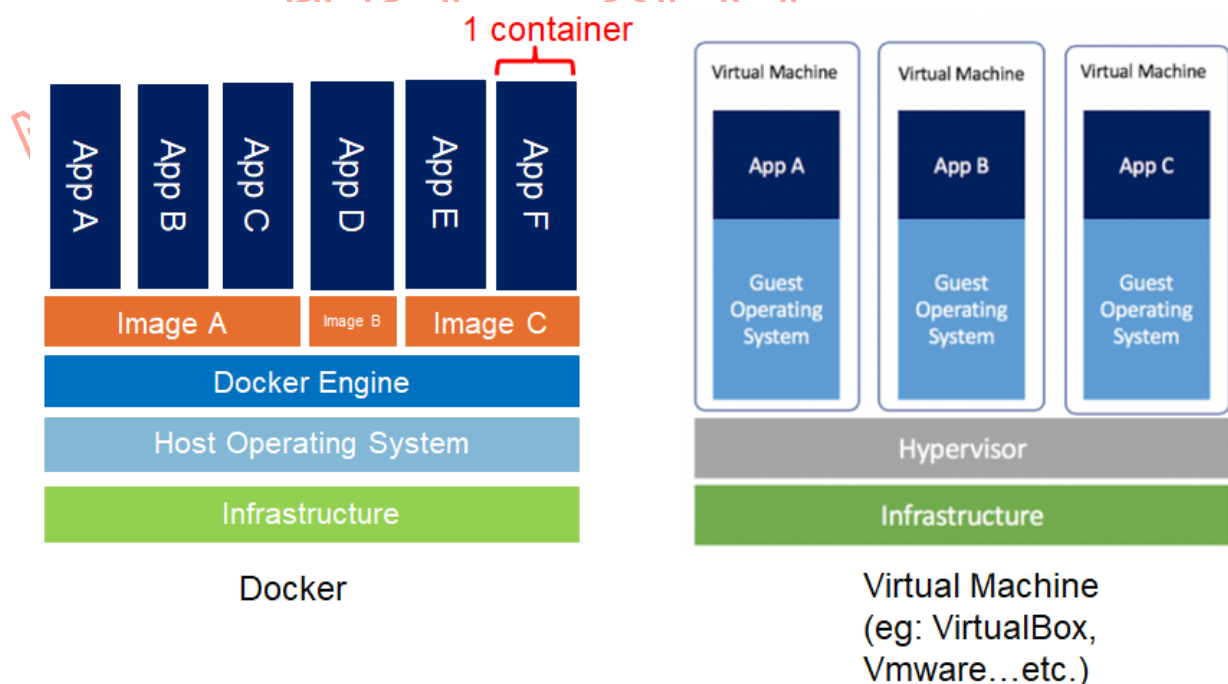


1.2 Required Environment

- Software:

Item	Version description	Note
Tool Version	v01.07.2211296	
Docker Version	v00.07.2108260	Support ubuntu16.04

In order to avoid the problem of environment variables or insufficient installation packages caused by the OS environment during Tool compilation, the method of operating in Docker with its own package is introduced to make the use environment uniform. Docker is an "application". If you have a Docker image (packaged environment), as long as you have an OS with Docker installed and the Docker Engine is operating normally, you can run this Docker like an application. The environment constructed by the image, in which the program is compiled and verified



● **Server**

Item	Version description	Note
Ubuntu 64-bit	Ubuntu Hirsute 21.04 or Ubuntu Groovy 20.10 or Ubuntu Focal 20.04 (LTS) or Ubuntu Bionic 18.04 (LTS) or Ubuntu Xenial 16.04 (LTS)	x86_64/amd64 or ARM or ARM64/AARCH64
CentOS	CentOS 7 or CentOS 8	x86_64/amd64 or ARM64/AARCH64
Debian 64-bit	Debian Buster 10 or Raspbian Buster 10	Debian: x86_64/amd64 or ARM or ARM64/AARCH64 Raspbian: ARM or ARM64/AARCH64
Fedora 64-bit	Fedora 32 or Fedora 33 or Fedora 34	x86_64/amd64 or ARM64/AARCH64

● **Hardware:**

Item	Description of requirement	Note
PC Hardware Requirements	Memory RAM needs at least 4G	
Supported IC Version	<ul style="list-style-type: none"> ● NT98336 (64bit) ● NT9852x (32bit) ● NT9832x (32bit) ● NT9856x (32bit)) ● NT98331 (32bit/64bit) ● NT98530 (64bit) 	

1.3 Installation

1.3.1 Installation of Docker

The following describes the installation method in the Ubuntu environment. If the environment is other OS, please refer to the steps in the Docker official website (<https://docs.docker.com/engine/install/>).

1. Uninstall the old Docker version first (the name of the old version is docker, docker.io or docker-engine)
[host]\$ sudo apt-get remove docker docker-engine docker.io containerd runc
2. Install the packages you will use first
[host]\$ sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release
3. Added Docker's official GPG key
[host]\$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
[host]\$ echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
4. Install Docker
[host]\$ sudo apt-get install docker-ce docker-ce-cli containerd.io
5. Check if Docker Engine is working
[host]\$ systemctl status docker
6. If Docker Engine is not working, execute the following commands
[host]\$ systemctl start docker.service
7. Verify that Docker is installed correctly
[host]\$ sudo docker run hello-world
This command will download a test image from DockerHub and execute a container. When the container runs, it will print out some messages before leaving

1.3.2 Use Docker Image at First Time

1. Load Docker image

```
[host]$ sudo docker load -i ubuntu16p04_v00_07_2108260.tar
```

2. Confirm the currently loaded Docker image

```
[host]$ sudo docker image ls
```

3. Use Docker image ubuntu16p04:v00.07.2108260 to create a new container, and treat <host_shared_folder_path> and <docker_shared_folder_path> as the same folder, and then the tool installation package can be put into docker via

```
<host_shared_folder_path>
```

```
[host]$ sudo docker run -it -v
```

```
<host_shared_folder_path>:<docker_shared_folder_path>
```

```
ubuntu16p04:v00.07.2108260
```

4. Confirm the container on the current system

```
[host]$ sudo docekr ps -a
```

5. Leave the current container

```
[docker]# exit
```

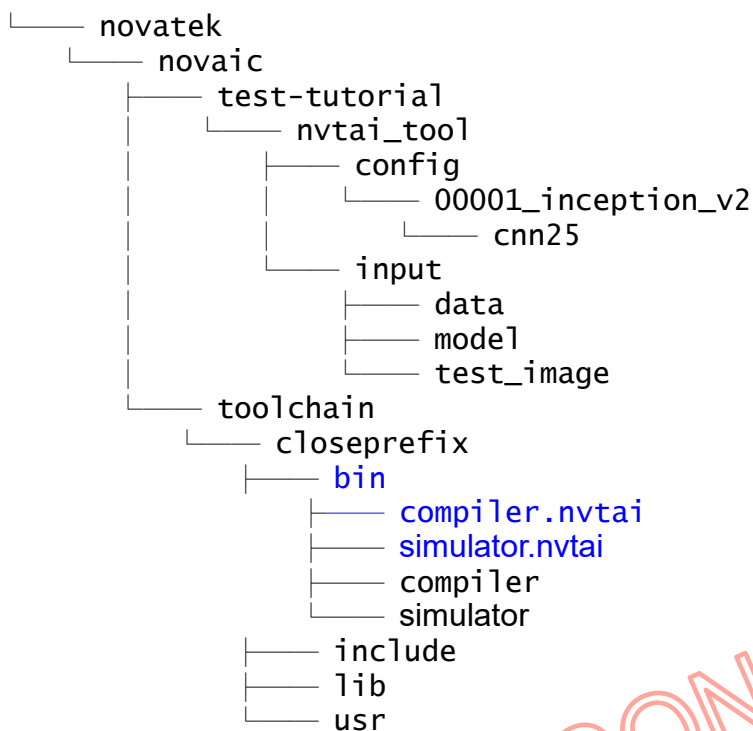
1.3.3 Installation of Tool Package

1. Put the installation package of the tool into docker via <host_shared_folder_path>

2. unzip the tool package

```
[docker]# tar zxvf release.tar.gz
```

3. After decompression, the main directory structure is roughly as shown in the figure below. The main programs compiler.nvtai and simulator.nvtai will be placed in the following location (closeprefix/bin/), such as the yellow box, the built-in example will be placed in test- in the tutorial



Notice

If there is a need to move executable files such as `compiler.nvtai`, `simulator.nvtai`, etc., the two directories `compiler` and `simulator` need to be moved to maintain at the same level.

2 Support Function

2.1 Support Framework

The following describes the framework and support status that the tool can currently support:

Framework name	Support status
Caffe	Supported, the built-in tool automatically converts to ONNX format
ONNX	Partially supported, can directly support ONNX format
Pytorch	Partially supported, the model needs to be converted to ONNX format in advance
Tensorflow	Partially supported, the model needs to be converted to ONNX format in advance

2.2 Verified Networks

At present, more than 200 Caffe network patterns have passed the test, and the information of 18 of them is extracted as shown in the table below

Network name	Input size of test
Inceptionv2	224x224x3
Inceptionv3	299x299x3
Inceptionv4	299x299x3
Mobilenetv1	224x224x3
Mobilenetssdv1	300x300x3
Vgg16	224x224x3
Resnet18	112x112x3
Resnet50	224x224x3
Resnet101	224x224x3
Yolov2	960x540x3

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

Yolov3	416x416x3
Yolov3_mobilenet_lite	512x512x3
Squeezenet_v1.1	227x227x3
Pfld	112x112x3
Enet	256x128x3
PoseNet	144x192x3
RetinaFaceNet	960x540x3
SphereFace	96x112x3

The Pytorch model information that has been tested so far is as follows:

Network name	Download URL
inceptionresnetv2	https://github.com/Cadene/pretrained-models.pytorch/blob/master/pretrainedmodels/models/inceptionresnetv2.py
inception_v3	https://github.com/pytorch/vision/tree/main/torchvision/models
inception_v4	https://github.com/Cadene/pretrained-models.pytorch/blob/master/pretrainedmodels/models/inceptionv4.py
resnet18	https://github.com/pytorch/vision/tree/main/torchvision/models
resnet50	https://github.com/pytorch/vision/tree/main/torchvision/models
resnet101	https://github.com/pytorch/vision/tree/main/torchvision/models
mobilenet_v1	https://github.com/zhaoyuzhi/PyTorch-MobileNet-v123
mobilenet_v2	https://github.com/pytorch/vision/tree/main/torchvision/models
Mobilenet V1 SSD	https://github.com/qfgaohao/pytorch-ssd
mobilefacenet	https://github.com/xuexingyu24/Pruning_MTCNN_MobileFaceNet_Using_Pytorch
yolov2	https://github.com/tztztztzt/yolov2.pytorch
yolov3	https://github.com/zldrobit/onnx_tflite_yolov3
vgg16	https://github.com/pytorch/vision/tree/main/torchvision/models
squeezenet_v1	https://github.com/pytorch/vision/tree/main/torchvision/models
retinaface	https://github.com/bubbliiiing/retinanet-pytorch
wide_resnet50_2	https://github.com/pytorch/vision/tree/main/torchvision/models
wide_resnet101_2	https://github.com/pytorch/vision/tree/main/torchvision/models
yolo_v3_tiny	https://github.com/zldrobit/onnx_tflite_yolov3
yolo_v3_spp	https://github.com/zldrobit/onnx_tflite_yolov3

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

pflid	https://github.com/polarisZhao/PFLD-pytorch
lanenet	Private network
res14_clean1029_col or	Private network
denoiser	Private network
unet	https://github.com/bubbliiiing/unet-pytorch
ghostnet	Private network
mnasnet0_5	https://github.com/pytorch/vision/tree/main/torchvision/models
resnext50_32x4d	https://github.com/pytorch/vision/tree/main/torchvision/models
DTLN	Private network
ssd	Private network (Use the DSP of NT98530 to support op)
deeplab_resnet50	https://pytorch.org/hub/pytorch_vision_deeplabv3_resnet101/ (Use the DSP of NT98530 to support op)
Note	
The post-processed and the pre-process of some of the above models has been removed or integrated into the NUE2 HW before inference.	

The Tensorflow model information that has been tested so far is as follows:

Network name	Download URL
inception_v2	Private network
inception_v3	Private network
inception_v4	https://www.deepdetect.com/models/tf/
resnet50	Private network
resnet101	Private network
mobilenet_v1	https://github.com/tensorflow/models/tree/master/research/slim#pre-trained-models/
mobilenet_v2	https://github.com/tensorflow/models/tree/master/research/slim#pre-trained-models/
mobilenet_ssd	https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md/
mobilenet_yolov 3/yolov3-lite	PINTO_model_zoo/download_saved_model.sh at main · PINTO0309/PINTO_model_zoo (github.com)

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

mobilefacenet	https://github.com/sirius-ai/MobileFaceNet_TF/tree/master/arch/pretrained_model
vgg16	https://www.deepdetect.com/models/tf/
squeezenet_v1	https://github.com/oracle/graphpipe/tree/master/docs/models
pflid	Private network
densnet	https://tfhub.dev/tensorflow/lite-model/densenet/1/default/1
mnasnet1_0	https://tfhub.dev/tensorflow/lite-model/mnasnet_1.0_224/1/default/1
efficientnet	https://tfhub.dev/tensorflow/lite-model/efficientnet/lite0/fp32/2
Note	
<p>The post-processed and the pre-process of some of the above models has been removed or integrated into the NUE2 HW before inference.</p>	

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

2.3 Layer Fusion

In terms of hardware design, to save the bandwidth and speed up the performance, the following fusion combinations are effective for common architectures in the network

CNN Engine	BN_Scale	BN_Scale BN_Scale + Eltwise BN_Scale + Eltwise + ReLU BN_Scale + Eltwise + ReLU + Pool
	Eltwise	Eltwise Eltwise + ReLU Eltwise + ReLU + Pool
	ReLU	ReLU ReLU + Pool
	Pool	Pool
	Convolution	Convolution Convolution + BN_Scale Convolution + BN_Scale + Eltwise Convolution + BN_Scale + Eltwise + ReLU Convolution + BN_Scale + Eltwise + ReLU + Pool Convolution + Eltwise Convolution + Eltwise + ReLU Convolution + Eltwise + ReLU + Pool Convolution + ReLU Convolution + ReLU + Pool Convolution + Pool
	Deconvolution	Deconvolution Deconvolution + BN_Scale Deconvolution + BN_Scale + Eltwise Deconvolution + BN_Scale + Eltwise + ReLU Deconvolution + BN_Scale + Eltwise + ReLU + Pool Deconvolution + Eltwise Deconvolution + Eltwise + ReLU Deconvolution + Eltwise + ReLU + Pool Deconvolution + ReLU Deconvolution + ReLU + Pool Deconvolution + Pool
NUE Engine	Fully Connection	FC FC + ReLU

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

2.4 Support Layer List

The following introduces the list of deep learning functions supported by the tools and hardware and the restricted specifications:

Caffe layer name	Description	52x	32x	528	56x	336	331	530
Common Layer	InnerProduct	✓	✓	✓	✓	✓	✓	✓
	Convolution	✓	✓	✓	✓	✓	✓	✓
	Deconvolution	✓	✓	✓	✓	✓	✓	✓
Normalization	BatchNorm + Scale	✓	✓	✓	✓	✓	✓	✓
Pooling	Local Pooling (only ceil mode)	✓	✓	✓	✓	✓	✓	✓
	Global Pooling	✓	✓	✓	✓	✓	✓	✓
ROI Pooling	ROI Pooling (max)	✓	✓	✓	✗	✓	✓	✓
	PSROI Pooling (max)	△	△	△	✗	△	△	△
Activation	ReLU	✓	✓	✓	✓	✓	✓	✓
	cReLU	△	△	△	△	△	△	△
	PreLU	✓	✓	✓	✓	✓	✓	✓
	Leaky ReLU	✓	✓	✓	✓	✓	✓	✓
	Abs	✓	✓	✓	✓	✓	✓	✓
	tanH	✓	✓	✓	✓	✓	✓	✓
	Sigmoid	✓	✓	✓	✓	✓	✓	✓
Recurrent	LSTM	✓	✓	✓	✓	✓	✓	✓
MatrixMultiplication	MatrixMultiplication	✓	✓	✓	✓	✓	✓	✓
Utility	Reshape	✓	✓	✓	✓	✓	✓	✓
	Crop	✓	✓	✓	✓	✓	✓	✓
	Clip	✓	✓	✓	✓	✓	✓	✓
	Threshold	✓	✓	✓	✓	✓	✓	✓
	Dummy (only support dummy → crop struc)	✓	✓	✓	✓	✓	✓	✓
	Flatten	✓	✓	✓	✓	✓	✓	✓
	Permute	✓	✓	✓	✓	✓	✓	✓
	Reorganize (stride=2)	✓	✓	✓	✓	✓	✓	✓
	Reverse (CPU Layer)	✓	✓	✓	✓	✓	✓	✓
	Passthrough (stride=2)	✓	✓	✓	✓	✓	✓	✓
	Concat	✓	✓	✓	✓	✓	✓	✓

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

	Slice	✓	✓	✓	✓	✓	✓	✓
	Eltwise (2-input)	✓	✓	✓	✓	✓	✓	✓
	Axpy	✓	✓	✓	✓	✓	✓	✓
	Upsample (1x, 2x, 4x, 8x) (Nearest)	✓	✓	✓	✓	✓	✓	✓
	Softmax	✓	✓	✓	✓	✓	✓	✓
	Cross Correlation	✓	✓	✓	✓	✓	✓	✓
	LRN	✗	△	✗	△	△	△	△
	Feature Compression Decoder (FCD)	✓	✓	✓	✓	✓	✗	✓
Preprocess	Mean Subtraction	✓	✓	✓	✓	✓	✓	✓
	Image Crop/Pad	△	△	△	△	△	△	△
	Image Scaling Down (Bilinear)	✓	✓	✓	✓	✓	✓	✓
	Image Scaling Up (Bilinear)	✗	△	✗	△	✓	✓	✓
	Image Rotate	△	△	△	△	△	△	△
	Image Flip	✗	△	△	△	△	△	△
	YUV2RGB	✓	✓	✓	✓	opt	opt	opt
	RGB2HSV	△	△	△	△	✗	✗	✗
	Normalize Scale	✓	✓	✓	✓	✓	✓	✓
Postprocess	Proposal (CPU Layer)	✓	✓	✓	✓	✓	✓	✓
	Priorbox (CPU Layer)	✓	✓	✓	✓	✓	✓	✓
	Detectionout (CPU Layer)	✓	✓	✓	✓	✓	✓	✓

Note:

✓: Indicates that this feature is supported by tool

△: Indicates that this feature is not currently supported and is expected to be supported in the future

✗: Indicates that it does not support

opt: the coef. Of color space convert can be set optionally.

The following describes the list of Onnx framework functions supported by tool:

Onnx layer name	52x	32x	528	56x	336	331	530
Abs	✓	✓	✓	✓	✓	✓	✓
Add	✓	✓	✓	✓	✓	✓	✓
AveragePool	✓	✓	✓	✓	✓	✓	✓
BatchNorm	✓	✓	✓	✓	✓	✓	✓

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

Clip	✓	✓	✓	✓	✓	✓	✓
Conv	✓	✓	✓	✓	✓	✓	✓
Add	✓	✓	✓	✓	✓	✓	✓
ConvTranspose	✓	✓	✓	✓	✓	✓	✓
Concat	✓	✓	✓	✓	✓	✓	✓
Flatten	✓	✓	✓	✓	✓	✓	✓
Gemm	✓	✓	✓	✓	✓	✓	✓
GlobalAveragePool	✓	✓	✓	✓	✓	✓	✓
GlobalMaxPool	✓	✓	✓	✓	✓	✓	✓
LeakyRelu	✓	✓	✓	✓	✓	✓	✓
LSTM	✓	✓	✓	✓	✓	✓	✓
MatMul	✓	✓	✓	✓	✓	✓	✓
Max	✓	✓	✓	✓	✓	✓	✓
MaxPool	✓	✓	✓	✓	✓	✓	✓
MaxRoiPool	✓	✓	✓	x	✓	✓	✓
Mul	✓	✓	✓	✓	✓	✓	✓
Pad	✓	✓	✓	✓	✓	✓	✓
Prelu	✓	✓	✓	✓	✓	✓	✓
ReduceMean	✓	✓	✓	✓	✓	✓	✓
Relu	✓	✓	✓	✓	✓	✓	✓
Resize	✓	✓	✓	✓	✓	✓	✓
Sigmoid	✓	✓	✓	✓	✓	✓	✓
Softmax	✓	✓	✓	✓	✓	✓	✓
Sub	✓	✓	✓	✓	✓	✓	✓
Tanh	✓	✓	✓	✓	✓	✓	✓
Transpose	✓	✓	✓	✓	✓	✓	✓
Upsample	✓	✓	✓	✓	✓	✓	✓
Reshape	✓	✓	✓	✓	✓	✓	✓
Slice	✓	✓	✓	✓	✓	✓	✓
Split	✓	✓	✓	✓	✓	✓	✓
Neg (Only support Neg + Relu struc.)	✓	✓	✓	✓	✓	✓	✓
Sub	✓	✓	✓	✓	✓	✓	✓
Tanh	✓	✓	✓	✓	✓	✓	✓
Resize*	✓	✓	✓	✓	✓	✓	✓
Sqrt*	✓	✓	✓	✓	✓	✓	✓
Exp* **	✓	✓	✓	✓	✓	✓	✓
Div* **	✓	✓	✓	✓	✓	✓	✓
Log*	✓	✓	✓	✓	✓	✓	✓
Pow*	✓	✓	✓	✓	✓	✓	✓
Sin*	✓	✓	✓	✓	✓	✓	✓

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

Floor*	✓	✓	✓	✓	✓	✓	✓
Round*	✓	✓	✓	✓	✓	✓	✓
Layer Norm**	✗	✗	✗	✗	✗	✗	✓

Note:

✓: Indicates that this feature is supported by tool

△: Indicates that this feature is not currently supported and is expected to be supported in the future

✗: Indicates that it does not support

*: Cpu layer inference by ACL (ARM Compute Library)

** : Dsp layer inference by ACL (ARM Compute Library)


Notice

ROI-pooling: the main road features are divided into single-batch and multi-batch usage:

- single-batch: can be multiple ROIs, but only static settings (the number of ROIs specified in the prototxt)
- multi-batch: the batch idx of ROI [batch idx, x1, y1, x2, y2] is not supported. Input feature batch and ROI have a one-to-one relationship (one batch corresponds to one ROI), so the number of input feature batches = number of ROI = number of output feature batches.

Pay attention when placing the last layer in the following:

- Concat
- Slice
- Reshape
- Flatten
- Permute1x1 (whcn → cwhn)

The problem with these layers in the last layer is , If you want to get the last layer of information (especially the dimensional information) , what you get is the upper level information, which can cause misunderstandings (just causes misunderstanding, not means cannot output layer)

2.4.1 Limitation Table of Tool

The following table describes the detailed limitations of the functions supported by the deep learning network with tool. For a detailed layer support list, please refer to the following two documents; Novaic_SupportSpec_Caffe_vxx.xx_en and Novaic_SupportSpec_ONNX_vxx.xx_en

2.4.2 Limitation Table of HW

Please refer to the Novaic_SupportSpec_CNN25_HW_vxx.xx_en document for IC HW restrictions of each generation, and network parameters can be planned according to this document to meet the effective operation of the hardware.

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

3 Gen-tool Introduction

Gen-tool is used to convert ONNX, Caffe, Pytorch, and Tensorflow architecture models, and after optimization, generate a binary model file that can be interpreted by the hardware.

3.1 Config Setting

The Config file is placed in the nvta_tool folder by default. Take 00001_inception_v2 as an example. The data structure setting is shown in the figure below

```

├─ config
│   └─ 00001_inception_v2
│       └─ cnn25
│           ├── gen_config.txt
│           └─ sim_config.txt

```

1. For single input, open config\00001_inception_v2\cnn25\gen_config.txt and make the following settings

[1] Set model and label path: (Support relative to the path set by `-config-dir`, or absolute path)

```
### [GENERAL]
```

```
## model
```

```
[path/model_dir] = ..\nvta_tool\input\model\0001_inception_v2
```

- [2] Set mean data path: (Support relative to the path set by `-config-dir`, or absolute path)

```
## mean
```

```
[path/mean_path] = ..\nvta_tool\input\model\0001_inception_v2\mean_data.txt
```

- [3] Set reference data path: (Support relative to the path set by `-config-dir`, or absolute path)

```
## img
```

```
[path/ref_img_dir] = ..\nvta_tool\input\data\fakeset\jpg
```

```
## list
```

```
[path/ref_list_path] = ..\nvta_tool\input\data\fakeset\ref_img_list.txt
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

- [4] Set the output folder after Gen-tool is executed: (Support relative to the path set by – config-dir, or absolute path)

```
### [GENERATOR]
```

```
## gen output root
```

```
[path/out_dir] = ..\nvtai_tool\output
```



Notice

Unlike the setting of sim_config, when setting the output path in gen_config, remember not to add the network name, such as ..\nvtai_tool\output\00001_inception_v2

- [5] Number of reference images [1, 1000]:

```
## img num [1, 1000]
```

```
[ref_data/num] = 20
```



Notice

Since the type and quantity of ref_data used sometimes affect the quantified result, if the user feels that cos or avg.diff. is not up to the standard during simulation or testing, the user can try to adjust ref_img_list or [ref_data/num]

- [6] Set whether to turn on the function of multi-scale input:

```
## model mode
```

```
# 0: single-scaled net input, 1 nvt_model
```

```
# 1: multi-scaled net input, multiple nvt_model, sdk dynamically choose proper model to use
```

```
[multiscale/en] = 0
```

- [7] If [multiscale/en] = 1, set the size of each resolution:

```
## net input sizes(preproc output sizes) for multi-scaled model
```

```
# please modify the scale in descending order
```

```
# ex. [multiscale/width] = 1920, 1280
```

```
# [multiscale/height] = 1080, 720
```

```
# there are 2 sizes of net input => 1920x1080, 1280x720
```

```
[multiscale/width] = 1920, 1280
```

```
[multiscale/height] = 1080, 720
```



Notice

1. The number of [multiscale/width] and [multiscale/height] settings should be the same.
2. The number of [multiscale/width] and [multiscale/height] settings is limited to [1, 64]
3. For NT98530, the value of [multiscale/width] and [multiscale/height] is limited to [1, 4096]
4. For other chips, the value of [multiscale/width] and [multiscale/height] should be ≥ 1

[8] ### [DYNAMIC_BATCH]

model mode

0: static-batch-size net, 1 nvt_model

1: dynamic-batch-size net, multiple nvt_model, sdk dynamically choose proper model to use

[dynamic_batch/en] = 1

[dynamic_batch/batch_size] = 2, 1



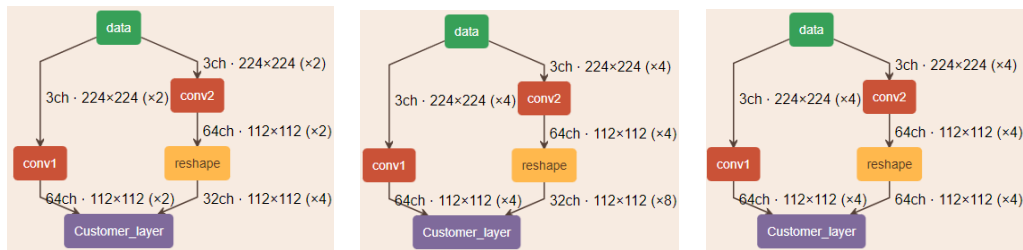
Notice

1. All the batch number using at AISDK should be setting at gentool. For example, if the dynamic_batch_size = 8, 4, 1 at AISDK, the setting in gen_config.txt should be [dynamic_batch/batch_size] = 8, 4, 1 too. Therefore, in this case, the input can not use batch_size = 6
2. [dynamic_batch/batch_size] must be set in descending order and must not contain duplicate batch values, e.g. [dynamic_batch/batch_size] = 16, 10, 8, 2, 1
3. ref_list.txt settings are based on the largest batch number. For example, the max. batch number = 16, it could be set 16 file names of rer. Images in ref_list.txt
4. All the combination of batch size could be known and static at gentool.
7. When using Customer layer and Reshape layer, and there exists certain layers which contains multiple input blobs or multiple output blobs in the network, the batch size of the inputs/outputs of these layer must changed in a proportional way. For example, in the left image below, when the batch size of network input is 2, the shape of two inputs of the Customer_layer are [2, c1, h1, w1] and [4, c2, h2, w2]. In the center image below, if we change the batch size of network input to 4, the shape of the two inputs of Customer_layer changed to [4, c1, h1, w1] and [8, c2, h2, w2]. The batch size of these 2 input blobs become 2 times larger comparing to original case. However, In the right image below, the shape of these 2 blobs become [4, c1, h1, w1] and [4, c2, h2, w2]. The batch size of first blob is 2 times larger, while the second blob is 1 time the same size, which did

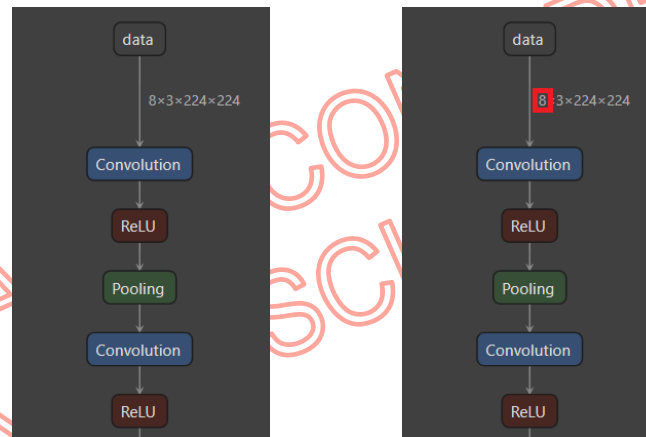
Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

not meet the proportional constraint.



6. When $[\text{dynamic_batch}/\text{en}] = 0$, the gentool will use the original strategy of setting for batch. Take left image below for example. The batch size of network input equals to 8, and the batch size of all the other layers is calculated using their input shape. When $[\text{dynamic_batch}/\text{en}] = 1$, the original static batch_size will be replaced by $[\text{dynamic_batch}/\text{batch_size}]$. For the case in right image below, the batch size 8 of data blob will be overwritten to the values set in $[\text{dynamic_batch}/\text{batch_size}]$.



[9] Set the input format of pre-processing layer(NUE2):

NUE2 input format

0: FMT_YONLY

1: FMT_RGB

2: FMT_YUV420

3: FMT_FEAT

4: FMT_BGR

5: x

[preproc/in/fmt] = 2

[10] If $[\text{preproc}/\text{in}/\text{fmt}] = 3$, Set the input type of FMT_FEAT:

input type for FMT_FEAT

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

0: TYPE_INT8
1: TYPE_UINT8
2: TYPE_INT16
3: TYPE_UINT16
6: TYPE_FLOAT32
[preproc/in/type] = 1

[11] If **[preproc/in/fmt] = 3**, set frac bit of FMT_FEAT(fixedpoint):

input frac bit num for FMT_FEAT
[preproc/in/frac_bit_num] = 0

[12] Set the input size of pre-process (NUE2):

input size[1, 4096]
input size to network[1, 4096]
[preproc/in/width] = 1920
[preproc/in/height] = 1080
[preproc/in/channel] = 2
[preproc/in/batch] = 1
[preproc/in/time] = 1

[13] Set the resize mode of reference img:

0: RESIZE_WH (resize to **[preproc/resize/width]** & **[preproc/resize/height]**)
1: x
2: x
3: RESIZE_WH_WITH_TILING (Keep the aspect ratio and repeatedly concat to **[preproc/resize/width]** and **[preproc/resize/height]**)
[preproc/resize/mode] = 0

[14] Set the output size of Resize :

RESIZE_WH PARAMETERS
width & height [1, 1023]
[preproc/resize/width] = 224
[preproc/resize/height] = 224

[15] Set whether to run mean subtraction:

[MEAN SUB]

0: disable meansub

1: enable meansub

[preproc/meansub/en] = 1

[16] Enable 16-bit (high precision) output of meansub:

0: disable meansub

1: enable meansub

[preproc/meansub_hp/en] = 0

[17] Mean subtraction execution mode (DC or Planar): DC means that each channel only subtracts one value (refer to ~input\model\customer\inception_v2\mean_data.txt), Planar refers to each channel subtracts the value related channel of a mean image (usually configured in the form of mean_data.binaryproto)

mean data mode

0: MEANSUB_DC

1: MEANSUB_PLANAR

[preproc/meansub/mode] = 0

[18] Set the mean data file format of mean subtraction :

mean data format

0: TXT

1: BINARYPROTO

[preproc/meansub/fmt] = 0



Notice

If [preproc/meansub/mode] = 1 (MEANSUB_PLANAR), currently only [preproc/meansub/fmt] = 1 (BINARYPROTO) is supported

[19] Set the data order of mean in mean_path :

mean_path mean data fmt type

0: FMT_YONLY

1: FMT_RGB

2: x

3: FMT_FEAT

4: FMT_BGR

5: x

[preproc/meansub/fmt_type] = 1



Notice

If [preproc/meansub/mode] is set to Planar mode, since the .binaryproto format read is BGR format, [preproc/meansub/fmt_type] also remember to modify it to the corresponding format (4)

[20] Set whether to run the normalization process (multiply the constant ($1/\sigma$) after mean subtraction)

[NORMALIZE]

0: disable normalization

1: enable normalization

[preproc/normalize/en] = 0

[21] Set the constant ($1/\sigma$), and the value is [0, 1]:

normalize scale

[preproc/normalize/scale] = 0.017



Notice

1. [preproc/normalize/scale] can also set 3 float values, separated by commas ',', these three values correspond to the [preproc/out_fmt] layout settings, for example:

[preproc/normalize/scale] = 0.017, 0.018, 0.019

[preproc/out_fmt] = 3

The norm scale of channel_B is 0.017, and the norm scale of channel_R is 0.019

2. When [preproc/normalize/scale] is set to 3 float values, [preproc/normalize/en] must be enabled, and [preproc/out_fmt] must be set to PREPROC_OUT_FMT_RGB or PREPROC_OUT_FMT_BGR

[22] Set the format of pre-process (NUE2) output:

[NUE2 OUT FORMAT]

0: PREPROC_OUT_FMT_YONLY

1: x

```
# 2: PREPROC_OUT_FMT_RGB
# 3: PREPROC_OUT_FMT_BGR
# 4: PREPROC_OUT_FMT_FEAT


```

[23] NUE2 csc mode for yuv2rgb:

```
# 0: CSC_YUV2RGB (default value)
# 1: CSC_BT601_YUV2RGB_LIMIT (VIDEO MODE) /* video transfer mode, RGB value
range [16, 235]*/
# 2: CSC_BT709_YUV2RGB_LIMIT (VIDEO MODE) /* video transfer mode, RGB value
range [16, 235]*/
# 3: CSC_BT601_YUV2RGB_FULL (PIC MODE) /* picture transfer mode, RGB value
range [0, 255]*/
# 4: CSC_BT709_YUV2RGB_FULL (PIC MODE) /* picture transfer mode, RGB value
range [0, 255]*/


```



Notice

[This parameter is only supported by chip 530/331/336](#)

[24] Set whether to print the result of classification (Top-5):

```
### [POSTPROCESS]
# 0: disable post process (classify accuracy)
# 1: enable post process (classify accuracy)
[postproc/en] = 1
```

[25] Set the bit of DRAM input feature to 16:

```
### [INPUT PRECISION]
## 16bit input to functions, this priority is higher than [precision/mode]
# 0: disable 16bit
# 1: enable 16bit


```

[precision/in_hp/28ltwise_en] = 0

[precision/in_hp/roipool_en] = 0

0: Adjusted by tool (default)

1: eltwise enabled separately: 16bit (int16), eltwise not enabled separately: 16bit (sign is determined by the tool)

2: eltwise enabled separately: 8bit (int8), eltwise not enabled separately: 8bit (sign is determined by the tool)

[precision/in_hp/eltwise_mode] = 0

0: 16bit (default)

1: 8bit

[precision/in_hp/sigmoid_mode] = 0

Since sigmoid is directly set to 8bit in some cases, it may affect the accuracy, so [precision/sigmoid_hp_int_thd] is used to control whether sigmoid is adjusted back to 16bit (even if sigmoid_mode is set to 1, sigmoid_hp_int_thd may still change sigmoid in bit back to 16bit), The default is 3, and the value range is 3 - 8. The larger the value, the easier it is to adjust the in bit to 16bit. If sigmoid_mode is 0, this option has no effect.

[precision/sigmoid_hp_int_thd] = 3

[26] Set the bit of DRAM output feature to 16:

[OUTPUT PRECISION]

16bit output to functions, this priority is higher than [precision/mode]

0: disable 16bit

1: enable 16bit

[precision/out_hp/eltwise_en] = 0

0: Adjusted by tool (default)

1: eltwise enabled separately: 16bit (int16), eltwise not enabled separately: 16bit (sign bit is determined by the tool)

2: eltwise enabled separately: 8bit (int8), eltwise not enabled separately: 8bit (sign bit is determined by the tool)

[precision/out_hp/eltwise_mode] = 0



Notice

[precision/out_hp/eltwise_mode] is mainly to adjust the priority of setting bitdepth to achieve this function, but it is not an absolute setting. In the following cases, the tool may still consider the hardware limit configuration

1. When this eltwise is split out for sigmoid
2. When the confusion where eltwise is located contains bnmul
3. When this eltwise is split out for lstm
4. When the eltwise is removed from the hp-conv_add mode
5. When this eltwise is dismantled from the prelu scheme
6. When this eltwise is hp_meansub
7. When the op followed by eltwise requires the input to be 16bit, such as sigmoid, tanh, bnmul, and 16bit hpconv
8. When eltwise is followed by a slice, the slice will bypass the bit setting. As long as one of the ops connected to the slice requires 16bit input, eltwise will output 16bit, and the input of other layers connected to the slice will also be 16bit.
9. When concat is followed by eltwise, concat will also bypass the bit setting. Different from slice, it is affected by two aspects when connected to concat
 - a. When the op behind concat requires the input to be 16bit, the input of all layers connected to concat must be 16bit at this time, and eltwise_mode does not work at this time
 - b. As long as one of all ops connected to concat requires an output of 16bit, the output of all ops connected to concat will be 16bit, and eltwise_mode will not work

- [27] Balance the threshold magnification of the weight distribution between conv+bn. When the conv+bn structure appears, if the weight of conv or bn is higher or lower than the average of all channels by out_range_ratio times, then adjust the weight of conv/bn (this function has not been extensively verified , If this value is set to a maximum value, the balance mechanism will not be activated):

```
### [BALANCE WEIGHT]
```

```
## outrange rate [1, 1023]
```

```
[precision/balance_weight/out_range_ratio] = 1000.000000
```

- [28] If there is a func+relu structure, this switch determines whether func refers to the relu output to allocate the current layer bit (relu negative slope = 0 or minimum value, it is recommended to turn on):

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

[REF RELU OUTPUT]

0: current reference its layer out

1: current reference next relu out

[precision/ref_relu_outval_en] = 1

- [29] When encountering the Mobile-Net series, or the feature map Std. is too small and the Scale is too large, you can open CLE to adjust to the appropriate range, but if you encounter a layer with too small parameter value, if CLE is used, the Bias value may increase sharply. To increase the output feature map value range, it is recommended to turn on when the network contains Depthwise-convolution and the accuracy of the network result is insufficient

[CROSS LAYER EQUALIZATION]

0: turn off CLE

1: turn on CLE

[precision/cle_en] = 0

- [30] Set calibration method

[CALIBRATION]

0: GEN_REF_CALIB_MAX

1: GEN_REF_CALIB_L2

2: x

3: x

4: GEN_REF_CALIB_PERCENTILE

[precision/calib_mode] = 0

When [precision/calib_mode] is not GEN_REF_CALIB_MAX, and the number of feature points in this layer is less than [precision/calib_min_actnum], it will automatically switch back to GEN_REF_CALIB_MAX implementation, which is to avoid L2 or PERCENTILE for some small feature maps.

[precision/calib_min_actnum] = 1

When [precision/calib_mode] = GEN_REF_CALIB_PERCENTILE, set the percentile threshold. This layer will sort the feature values of the reference image from large to small, and find the value at the [precision/percentile] scale in the corresponding distribution as the calibration value

[precision/percentile] = 0.9999

- [31] When avg_pooling encounters that the output frac_bit is larger than the input frac_bit, start the optimization algorithm to avoid result overflow. The default is to enable
- # 0: disable avg_pooling optimization
 - # 1: enable avg_pooling optimization
- [precision/avgpool_noclamp_en] = 1**
- [32] ### [convBias_bitdepth] only available on 530 & 331, this priority is lower than [precision/quant_injection]
- # 0: 8 bits
 - # 1: 12 bits , default
 - # 2: 32 bits
- [precision/bias_bitdepth_mode] = 1**
- [33] ### [preproc_clamp_en]
- # 0 : no preproc clamp en, bit info is determined by layer output
 - # 1: is same as in bit info , default
- [precision/preproc_clamp_en] = 1**
- [34] Support conv with 16bit high-precision
- # 0 : Automatically configure bit operations according to tools (default)
 - # 1 : Split all conv/deconv layers to support high-precision
 - # 2 : Split the specified conv/deconv layer to support high-precision (according to hpconv_config.txt in the config folder)
- [precision/in_hp/conv_op] = 0**



Notice

1. The conv/deconv layer specified in hpconv_config.txt is written as follows (you can fill in the layer that requires high-precision calculation according to the name of the golden_layer_name field in proc_bit_info.txt)
Convolution_conv_1_Y
Convolution_conv_2_Y
2. The impact of enabling high-precision computing will be the increase in time consumption and bandwidth, which needs to be used in conjunction with application requirements

- [35] Control the split or combination method of Prelu layer from tool, which default value is 0
- # 0: current method with 8bit
 - # 1: current method with 16bit
 - # 2: original method which is similar as old tool
- [precision/prelu_ctrl_mode] = 0**
- [36] When testing the performance of an untrained model on EVB, an error will often be reported due to the wrong weight value range exceeding the HW limit. At this time, you can use this switch to generate a test model with the parameter clamp
- # 0: assert when the quan. parm. are over hw spec.(default)
 - # 1: clamp when the quan. parm. are over hw spec.
- [precision/clamp_quan_parm] = 0**
- [37] Set whether to enable 8bit quantization model compression to reduce memory and model size: (accuracy lossy)
- #[WEIGHT COMPRESSION]
- # 0: disable quatization
 - # 1: enable quatization
- [compression/method/quant_en] = 0**
- [38] Set whether to enable k-means weight loss compression. This function must be enable at the same time as the VLC switch, and cannot be enable at the same time as quant_en. NT98331 does not support this function (accuracy loss)
- # 0: disable kmeans quantization (Default)
 - # 1: enable kmeans quantization
- [compression/method/kmeans_en] = 0**
- [39] Set whether to enable VLC weight lossless compression: (accuracy lossless)
- # 0: disable variable length coding
 - # 1: enable variable length coding
- [compression/method/vlc_en] = 0**
- [40] When **[compression/method/vlc_en] = 1**, user can set this flag to open the mix-compression mode to avoid the situations of some side effects.
- # 0: disable mixed compression
 - # 1: enable mixed compression

[compression/use_mixed_compression] = 0



Notice

Generally, user can turn on the `quan_en` flag to reduce the size and decrease the bandwidth effects. If there are the increasion of time cost during integrating progress or the requirement of reducing more model size, user can turn on `vlc_en` or `use_mixed_compression` to get the best case.

[41] In addition, some parameters listed as follows are aligned with those cmd config in Chapter 5.3. If they are set with cmd config at the same time, the cmd config will be the main one

0: Don't MSB Shrink for DW Conv (Default)

1: Do MSB Shrink for DW Conv

[performance/msb_shrink_en] = 0

Thld for Calculating Complexity for setting job or buf opt => 65535 is debug mode

[performance/buf_thld] = 312

Thld for Calculating Complexity for setting job or buf opt => 65535 is debug mode

[performance/graph_thld] = 128

0: don't use tiling (Default)

1: use tiling (only support by chip 530)

2: tiling + preload weight + 33ovaic33r33s33e output of tiling group in TCM (only support by chip 530)

[performance/use_tiling] = 0

0: debug at linear mode

1: linear mode

10: debug at graph mode

11: graph mode

[performance/job_opt] = 1

0: debug mode

1: shrink mode

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

2: shrink reorder(O1)
 # 3: shrink reorder(O2)
 # 4: non-tiling + shrink (only support by chip 530)
 # 5: non-tiling + shrink reorder(O2) (only support by chip 530)
[performance/buf_opt] = 1

[42] # 0: It is up to the tool to use the CPU layer or the HW layer (default)
 # 1: HW Softmax
 # 2: CPU Softmax
[platform/softmax] = 0

[43] When using the NT98530 platform, set {output_tensor_name} to use backend {backend_type} for the corresponding layer, where backend_type can be set as follows
 # EXT: means that both DSP or CPU can run, the priority will be determined by the tool
 # DSP: This layer is run by the DSP
 # CPU: This layer is run by the CPU
[tensor_backend/{output_tensor_name}] = {backend_type}



Notice

When [platform/softmax] = 2 (all softmax in this network runs by CPU), the [tensor_backend/{output_tensor_name}] setting will be forcibly cleared. If you want to set the CPU to run according to the single-layer softmax, [platform/softmax] must be 0 or 1 or not set

[44] # 0: Remove the reshape of the output layer
 # 1: Keep the reshape of the output layer (default)
[special/keepdim/reshape] = 1

[45] # 0: The nuesoftmax of the output layer does not add permute to switch back to the original dimension
 # 1: The nuesoftmax of the output layer needs to add two nuepermutes to return to the original dimension. If the nuepermute cannot be added, the softmax will become cpusoftmax) (default value)
[special/keepdim/softmax] = 1

[46] add extra upsample in the end if the last op is reshape and do not produce diff

0 : do not add upsample
1 : add upsample (default)
[special/keepdiff] = 1

- For multiple inputs, the reference image of each input and the variables of the preproc block need to be set separately, please use `[blob] = n`, `[blob_name] = datan` to separate different inputs. The following example is a dual digital input setting method



Notice

Each input needs to have a corresponding config setting, and blob_name must be consistent with the name of the onnx model input, otherwise the model cannot be converted smoothly

```
#####
#[PATH]
#####
### [GENERAL]
## model
[path/model_dir] = ..\nvtai_tool\input\model\customer\elt_multiblob

### [GENERATOR]
## gen output root
[path/out_dir] = ..\nvtai_tool\output

#####
#[REFERENCE DATA]
#####
## img num [1, 1000]
[ref_data/num] = 1

#####
#[FUNCTION]
#####
### [MULTISCALE]
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```
## model mode
# 0: single-scaled net input, 1 nvt_model
# 1: multi-scaled net input, multiple nvt_model, sdk dynamically choose proper model to use
[multiscale/en] = 0

## net input sizes(preproc output sizes) for multi-scaled model
# please modify the scale in descending order
# ex. [multiscale/width] = 1920, 1280
#     [multiscale/height] = 1080, 720
# there are 2 sizes of net input => 1920x1080, 1280x720
[multiscale/width] = 1920, 1280
[multiscale/height] = 1080, 720

[blob] = 0
[blob_name] = data0
### [REFERENCE DATA]
## mean
[path/mean_path] = ..\nvtai_tool\input\model\customer\elt_multiblob\mean_data.txt
## img
[path/ref_img_dir] = ..\nvtai_tool\input\data\feature\uint8_224x224x3x1\bin
## list
[path/ref_list_path] = ..\nvtai_tool\input\data\feature\uint8_224x224x3x1\ref_img_list.txt

### [PREPROCESS]
## NUE2 input format
# 0: FMT_YONLY
# 1: FMT_RGB
# 2: FMT_YUV420
# 3: FMT_FEAT
# 4: FMT_BGR
# 5: x
[preproc/in/fmt] = 3

## input type for FMT_FEAT
# 0: TYPE_INT8
# 1: TYPE_UINT8
# 2: TYPE_INT16
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.


```
# 3: TYPE_UINT16
[preproc/in/type] = 1

... (Omission)

## [NUE2 OUT FORMAT]
# 0: PREPROC_OUT_FMT_YONLY
# 1: x
# 2: PREPROC_OUT_FMT_RGB
# 3: PREPROC_OUT_FMT_BGR
# 4: PREPROC_OUT_FMT_FEAT
[preproc/out_fmt] = 4

[blob] = 1
[blob_name] = data1
### [REFERENCE DATA]
## mean
[path/mean_path] = ..\nvtai_tool\input\model\customer\elt_multiblob\mean_data.txt
## img
[path/ref_img_dir] = ..\nvtai_tool\input\data\feature\uint8_224x224x3x1\bin
## list
[path/ref_list_path] = ..\nvtai_tool\input\data\feature\uint8_224x224x3x1\ref_img_list.txt

### [PREPROCESS]
## NUE2 input format
# 0: FMT_YONLY
# 1: FMT_RGB
# 2: FMT_YUV420
# 3: FMT_FEAT
# 4: FMT_BGR
# 5: x
[preproc/in/fmt] = 3

## input type for FMT_FEAT
# 0: TYPE_INT8
# 1: TYPE_UINT8
# 2: TYPE_UINT16
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```
# 3: TYPE_UINT16
[preproc/in/type] = 1

... (Omission)

## [NUE2 OUT FORMAT]
# 0: PREPROC_OUT_FMT_YONLY
# 1: x
# 2: PREPROC_OUT_FMT_RGB
# 3: PREPROC_OUT_FMT_BGR
# 4: PREPROC_OUT_FMT_FEAT
[preproc/out_fmt] = 4

### [POSTPROCESS]
# 0: disable post process (classify accuracy)
# 1: enable post process (classify accuracy)
[postproc/en] = 0

#####
#[FEATURE PRECISION]
#####
### [INPUT PRECISION]
## 16bit input to functions, this priority is higher than [precision/mode]
# 0: disable 16bit
# 1: enable 16bit
[precision/in_hp/conv_en] = 0
[precision/in_hp/bnscale_en] = 0
[precision/in_hp/deconv_en] = 0
[precision/in_hp/fc_en] = 0
[precision/in_hp/eltwise_en] = 1
[precision/in_hp/roipool_en] = 0

### [BALANCE WEIGHT]
## outrange rate [1, 1023]
[precision/balance_weight/out_range_ratio] = 1000.000000

### [REF RELU OUTPUT]
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```
# 0: current reference its layer out
# 1: current reference next relu out
[precision/ref_relu_outval_en] = 1
```

```
### [CROSS LAYER EQUALIZATION]
```

```
# 0: turn off CLE
# 1: turn on CLE
[precision/cle_en] = 0
```

```
#####
```

```
#[WEIGHT COMPRESSION]
```

```
#####
```

```
# 0: disable quatization
# 1: enable quatization
[compression/method/quant_en] = 0
```

```
# 0: disable variable length coding
# 1: enable variable length coding
[compression/method/vlc_en] = 0
```

```
#####
```

```
#[PERFORMANCE MODE]
```

```
#####
```

```
### [MEMORY MODE]
```

```
# 0: disable shrink memory
# 1: enable shrink memory
[performance/shrink_en] = 1
```

3. The following table lists

- [1] [preproc/in_fmt]
- [2] [preproc/meansub_fmt_type]
- [3] [preproc/out_fmt]

Effective combination of settings , **Please do not exceed the configuration set in this table to avoid errors:**

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

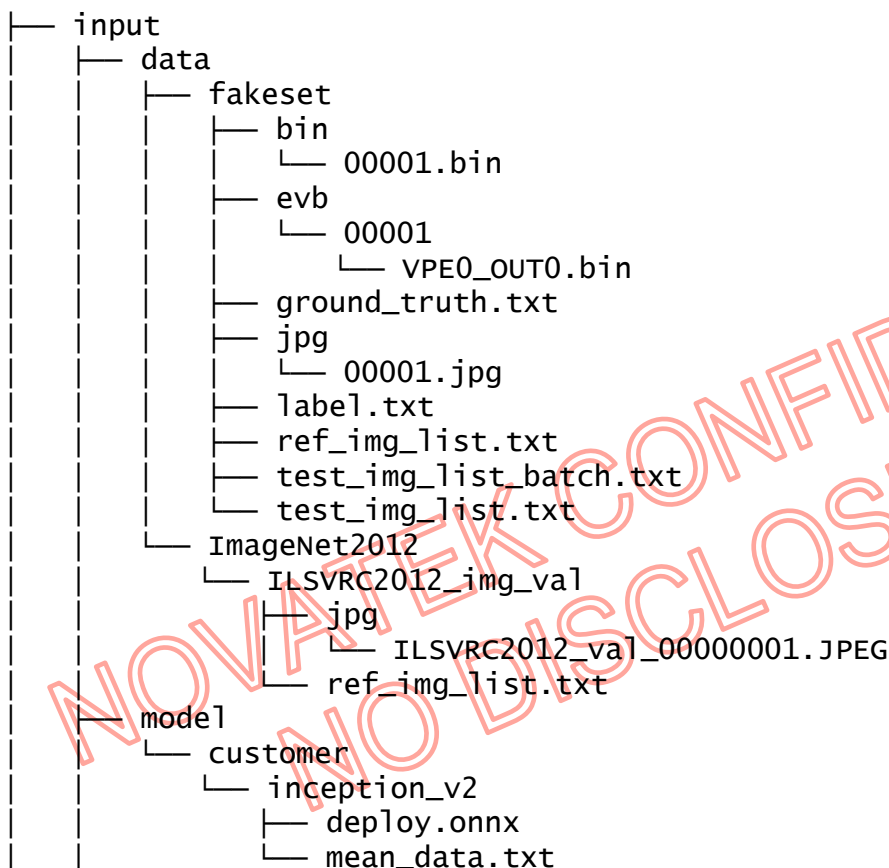
With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

preproc/in/fmt	preproc/meansub/fmt_type	preproc/out_fmt
FMT_YONLY	FMT_YONLY	PREPROC_OUT_FMT_YONLY
FMT_RGB	FMT_RGB FMT_BGR	PREPROC_OUT_FMT_RGB PREPROC_OUT_FMT_BGR
FMT_BGR	FMT_RGB FMT_BGR	PREPROC_OUT_FMT_RGB PREPROC_OUT_FMT_BGR
FMT_YUV420	FMT_RGB FMT_BGR	PREPROC_OUT_FMT_RGB PREPROC_OUT_FMT_BGR
FMT_FEAT	FMT_FEAT	PREPROC_OUT_FMT_FEAT

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

3.2 Input Data

The input data file of Gent-tool is placed in the nvtai_tool folder by default. Take 00001_inception_v2 as an example. The data structure setting is shown in the figure below



Described as follows:

- ~\input\data\
 - File\Folder: [ImageNet2012, fakeset...](#)
 - Description: Refer to the image in advance and analyze the relative value of dynamic fixed point. If the input format is not feature bin, the reference image can be read in JPEG/jpg format. If the input format is feature bin, the reference image can be read in bin format.
 - File\Folder: [ref_img_list.txt](#)
 - Description: Each test data is separated by blank lines, as shown in the figure below

```
ref_img_list.txt
1 ILSVRC2012_val_00000001.JPEG
2 ILSVRC2012_val_00000002.JPEG
3 ILSVRC2012_val_00000003.JPEG
4 ILSVRC2012_val_00000004.JPEG
5 ILSVRC2012_val_00000005.JPEG
6 ILSVRC2012_val_00000006.JPEG
7 ILSVRC2012_val_00000007.JPEG
8 ILSVRC2012_val_00000008.JPEG
9 ILSVRC2012_val_00000009.JPEG
10 ILSVRC2012_val_00000010.JPEG
11 ILSVRC2012_val_00000011.JPEG
12 ILSVRC2012_val_00000012.JPEG
13 ILSVRC2012_val_00000013.JPEG
14 ILSVRC2012_val_00000014.JPEG
15 ILSVRC2012_val_00000015.JPEG
16 ILSVRC2012_val_00000016.JPEG
17 ILSVRC2012_val_00000017.JPEG
18 ILSVRC2012_val_00000018.JPEG
19 ILSVRC2012_val_00000019.JPEG
20 ILSVRC2012_val_00000020.JPEG
```



Notice

If the number of items listed in `ref_img_list.txt` is less than the number set in `[ref_data/num]`, the following error will be reported

Fatal `ref_data/num = 4 > the image number = 3` in `~/test-tutorial/nvtai_tool/~/nvtai_tool/input/data/cnet_cus1_reference_image/ref_img_list.txt`

- `~/input/model/customer/inception_v2`
 - File\Folder: `deploy.onnx`
 - Description: Place the network model to be converted. If it is a Caffe model (e.g. `deploy.prototxt`, `deploy.caffemodel`), it can also be placed in this folder
 - File\Folder: `mean_data.txt`
 - Description: Mean Image can be in the following formats
 - ◆ Binaryproto, the same as the Binaryproto format supported by general Caffe
 - ◆ Txt format

3.3 Output Data

In gen_config.txt, users can set the result location that Gen-tool wants to output by giving [path/out_dir] (default is ..\nvtai_tool\output), and the output folder structure is shown in the figure below

```
|— gentool
|   |— siminfo.bin
|   |— sdk
|       |— nvt_model.bin
```

Described as follows:

- ~\output\network_name\gentool\
 - File\Folder: [siminfo.bin](#)
 - Description: Reference file for Sim-tool simulation
- ~\output\network_name\sdk\
 - File\Folder: [nvt_model.bin](#)
 - Description: The converted binary data. It is used in the board and simulation test

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

4 Sim-tool Introduction

Sim-tool is used to simulate the behavior of the board. After the Gen-tool runs, users can use Sim-tool to simulate the results of the board on the PC and compare the results of the fixed-point model with the floating-point model.

4.1 Config Setting

The Config file is placed in the nvtai_tool folder by default. Take 00001_inception_v2 as an example. The data structure setting is shown in the figure below

```

|— config
|   |— 00001_inception_v2
|       |— cnn25
|           |— gen_config.txt
|           |— sim_config.txt

```

1. The content of config can be divided into single input and multiple input formats. Here, [config\00001_inception_v2\cnn25\sim_config.txt](#) is used as a single input example to illustrate each parameter setting

- [1] Set the path of Label/Model ([Support relative to the path set by –config-dir, or absolute path](#))

```
### [GENERAL]
```

```
## all class name
```

```
[path/label_path] = ..\nvta_tool\input\data\fakeset\label.txt
```

```
## model
```

```
[path/model_dir] = ..\nvta_tool\input\model\0001_inception_v2
```

- [2] Set test data path: ([Support relative to the path set by –config-dir, or absolute path](#))

```
### [TEST DATA]
```

```
## image
```

```
[path/test_img_dir] = ..\nvta_tool\input\data\fakeset\evb\0001_inception_v2
```

```
## ground truth
```

```
[path/test_gt_path] = ..\nvta_tool\input\data\fakeset\ground_truth.txt
```

```
## list
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

[path/test_list_path] = ..\nvtai_tool\input\data\fakeaset\test_img_list.txt

- [3] Set the output folder after Sim-tool is executed: (Support relative to the path set by – config-dir, or absolute path)

[SIMULATION]

simulation output root

[dataset] will store the test image results in a folder with the name [dataset] under [path/out_dir]/results and [path/out_dir]/simtool. It is recommended to have the same name as the test image folder

[dataset] = fakeaset

[path/out_dir] = ..\nvtai_tool\output\0001_inception_v2



Notice

Different from the setting of gen_config, when setting the output path in sim_config, remember to add the network name, such as ..\nvtai_tool\output\0001_inception_v2

- [4] Set the format of input test data:

src format

JPEG = 0,

BIN = 1,

[src/imgfmt] = 1

- [5] Whether the input data has been 45ovaic45r45s

preproc out bin

0: is preproc input bin

1: is preproc output bin

[src/is_preproc_out_bin] = 1

- [6] Type of input data

src bin file type

BLOB_TYPE_INT8 = 0,

BLOB_TYPE_UINT8 = 1,

BLOB_TYPE_INT16 = 2,

BLOB_TYPE_UINT16 = 3,

[src/bintype] = 0

- [7] Set the size of the input test data, between [1, 4096]:

src size [1, 4096]

[src/width] = 224

[src/height] = 224

[src/channel] = 3

[src/batch] = 1

[src/time] = 1

- [8] Set the size of the input test data, between [1, 4096]:

src size [1, 4096]

[src/width] = 224

[src/height] = 224

[src/channel] = 3

[src/batch] = 1

[src/time] = 1

- [9] After enabling, the input image file will be resized into src size (width/height) set by sim_config through opencv first, and then converted into fmt input by nue2 (default disable):

[JPG INPUT RESIZE]

0: disable jpg resize

1: enable jpg resize

[src/resize_en] = 0

- [10] After enabling, if the width or height of the input image file is odd, it will padding a row/column of 0 pixels on the bottom/right side:

[PAD JPG W/H to EVEN]

0: disable jpg padding

1: enable jpg padding

[src/padding_en] = 1

- [11] After enabling, if the width or height of the input image file is odd, the lower/right row/column will be discarded (default disabled), and cannot be enabled at the same time as [src/padding_en]:

[CROP JPG W/H to EVEN]

0: disable jpg cropping

1: enable jpg cropping

[src/crop_en] = 1

[12] On NT98530 platform, set what kind of backend runs in simulation.

DSP: This layer is run by the DSP

CPU: This layer is run by the CPU

[backend/convert_ext_to] = CPU

2. Multi-input network settings need to use these two tags (**[blob]** and **[blob_name]**) to distinguish different input settings. **[blob_name]** sets the tensor name of the original network input. For the setting method, please refer to the following example:



Notice

Each input needs to have a corresponding config setting, and blob_name should be consistent with the input of onnx model

```
[blob] = 0
[blob_name] = data0
### [TEST DATA]
## image
[path/test_img_dir] = ..\nvtai_tool\input\test_image\20094_elt_multiblob_dc
## list
[path/test_list_path] = ..\nvtai_tool\input\test_image\20094_elt_multiblob_dc\test_img_list.txt

## src format
# JPEG = 0,
# BIN = 1,
[src/imgfmt] = 1

## preproc out bin
# 0: is preproc input bin
# 1: is preproc output bin
[src/is_preproc_out_bin] = 0
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```
## src bin file type
# BLOB_TYPE_INT8 = 0,
# BLOB_TYPE_UINT8 = 1,
# BLOB_TYPE_INT16 = 2,
# BLOB_TYPE_UINT16 = 3,
[src/bin_type] = 1

## src size [1, 4096]
[src/width] = 224
[src/height] = 224
[src/channel] = 3
[src/batch] = 1
[src/time] = 1

[blob] = 1
[blob_name] = data1
### [TEST DATA]
## image
[path/test_img_dir] = ..\nvtai_tool\input\test_image\20094_elt_multiblob_dc
## list
[path/test_list_path] = ..\nvtai_tool\input\test_image\20094_elt_multiblob_dc\test_img_list.txt

## src format
# JPEG = 0,
# BIN = 1,
[src/imgfmt] = 1

## preproc out bin
# 0: is preproc input bin
# 1: is preproc output bin
[src/is_preproc_out_bin] = 0

## src bin file type
# BLOB_TYPE_INT8 = 0,
# BLOB_TYPE_UINT8 = 1,
# BLOB_TYPE_INT16 = 2,
# BLOB_TYPE_UINT16 = 3,
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

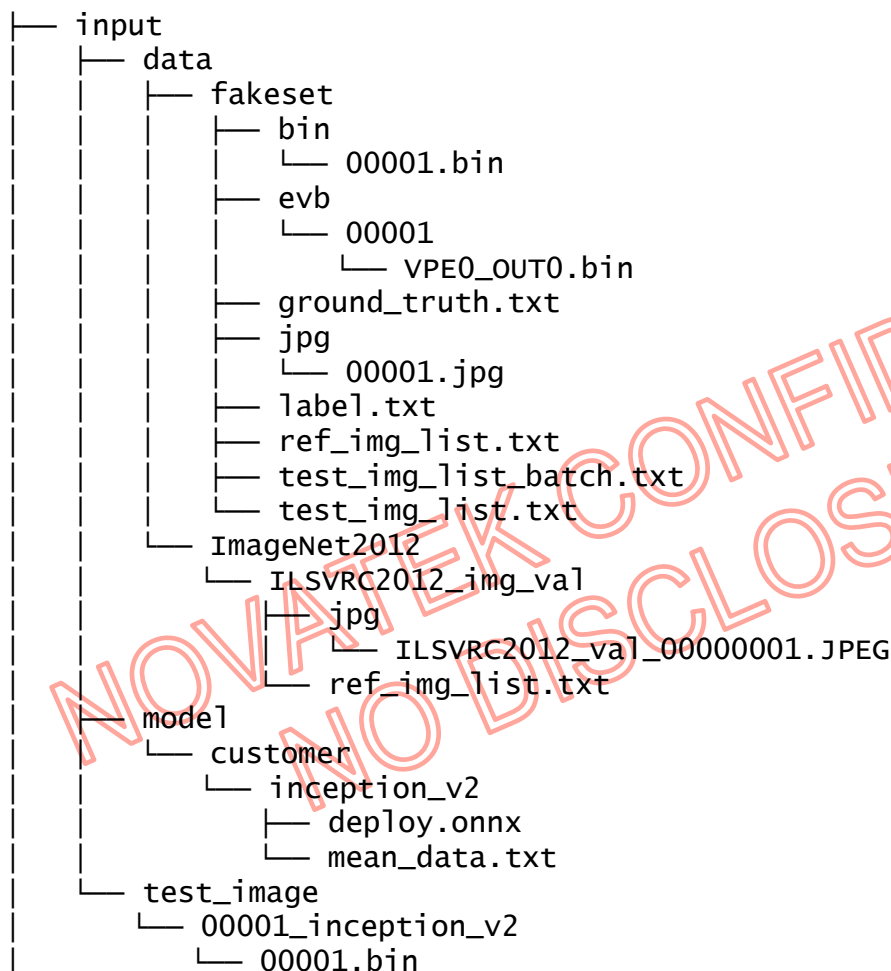
With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```
[src/bintype] = 1  
  
## src size [1, 4096]  
[src/width] = 224  
[src/height] = 224  
[src/channel] = 3  
[src/batch] = 1  
[src/time] = 1
```

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

4.2 Input Data

The input data file of Sim-tool is placed in the nvtai_tool folder by default. Take 00001_inception_v2 as an example. The data structure setting is shown in the figure below



Described as follows:

- ~\input\model\customer\inception_v2
 - File\Folder: [deploy.onnx](#)
 - Description: Place the network model to be converted. If it is a Caffe model ([deploy.prototxt](#), [deploy.caffemodel](#)), it can also be placed in this folder
- ~\input\test_image\00001_inception_v2
 - File\Folder: [00001.bin](#)

- Description: Place the input image bin file to be tested which is generated from the board

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

4.3 Output Data

The user can set the result location that Sim-tool wants to output by giving [path/out_dir] in sim_config.txt (default is ..\nvtai_tool\output\00001_inception_v2), and the output folder structure is shown in the figure below

```

├── debug
│   ├── golden_tensor
│   │   ├── data_nue2_output.bin
│   │   ├── InnerProduct_fc1_Gemm_Y.bin
│   │   ├── Pooling_avg_pool_3a_pool_Y.bin
│   │   └── ReLU_relu_5b_proj_Y.bin
│   └── sim_layer
│       ├── fixed
│       │   ├── CNN_1_OUT1.bin
│       │   ├── CNN_2_OUT0.bin
│       │   ├── NUE2_0_OUT0.bin
│       │   ├── NUE2_0_OUT1.bin
│       │   ├── NUE2_0_OUT2.bin
│       │   └── NUE_86_OUT0.bin
│       └── float
│           ├── data_nue2_output.bin
│           ├── InnerProduct_fc1_Gemm_Y.bin
│           ├── Pooling_avg_pool_3a_pool_Y.bin
│           └── ReLU_relu_5b_proj_Y.bin
├── gentool
│   ├── siminfo.bin
│   └── siminfo.txt
├── sdk
│   ├── fp32_input
│   │   └── NUE2_OUT.bin
│   ├── input_bin
│   │   └── 00001.bin
│   └── nvt_model.bin
└── simtool
    ├── layer_diff.json
    └── layer_diff.txt

```

Described as follows:

- ~\output\network_name\debug\
 - File\Folder: golden_tensor
 - Description: ONNX native network float operation result, used as golden answer
 - File\Folder: sim_layer\fixed
 - Description: The fixed buffer result obtained by Sim-tool running the binary

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

nvt_model.bin

- File\Folder: [sim_layer\float](#)
- Description: Restore sim_layer\fixed to float result, then compare with ONNX golden float result and produce layer_diff file
- Debug (golden_layer v.s sim_layer) process
 - ◆ To compare golden_layer and sim_layer, first open ~\simtool\fakeaset\layer_diff.txt, confirm the layer name to be queried, and search for the same layer name bin file under sim_layer\float and golden_layer\.
- ~\output\network_name\gentool\
 - File\Folder: [siminfo.txt](#)
 - Description: Produced by Sim-tool, which is a converted text file of siminfo.bin, used for debugging, and will be automatically produced during operation
- ~\output\ network_name \sdk\
 - File\Folder: [input_bin](#)
 - Description: Input file used to run Sim-tool
 - File\Folder: [fp32_input](#)
 - Description: Input file used to input ONNX native network
- ~\output\ network_name \simtool\
 - File\Folder: [layer_diff.txt & layer_diff.json](#)
 - Description: The difference between Sim-tool and ONNX feature values in each layer is recorded, as shown in the figure below, where the judgment criterion is as follows:
 - ◆ average absolute difference ratio (avg_diff): Its normal value is usually **< 0.02**
 - ◆ cosine similarity (cosine): Its normal value is usually **> 0.99**

Proc. No.	Layer name	COS similarity	Average absolute difference ratio
[45]	ReLU_relu_4c_3x3_reduce_Y	cosine(0.998469)	avg_diff(0.001848)
[46]	ReLU_relu_4c_3x3_Y	cosine(0.998393)	avg_diff(0.002223)
[47]	ReLU_relu_4c_double_3x3_reduce_Y	cosine(0.999036)	avg_diff(0.001719)
[48]	ReLU_relu_4c_double_3x3_0_Y	cosine(0.999110)	avg_diff(0.001577)
[49]	ReLU_relu_4c_double_3x3_1_Y	cosine(0.999054)	avg_diff(0.001434)
[50]	Pooling_avg_pool_4c_pool_Y	cosine(0.998455)	avg_diff(0.003116)
[51]	ReLU_relu_4c_proj_Y	cosine(0.997786)	avg_diff(0.004972)
[52]	ReLU_relu_4d_1x1_Y	cosine(0.998669)	avg_diff(0.002846)
[53]	ReLU_relu_4d_3x3_reduce_Y	cosine(0.998532)	avg_diff(0.001679)
[54]	ReLU_relu_4d_3x3_Y	cosine(0.998310)	avg_diff(0.001389)
[55]	ReLU_relu_4d_double_3x3_reduce_Y	cosine(0.999010)	avg_diff(0.001641)
[56]	ReLU_relu_4d_double_3x3_0_Y	cosine(0.999157)	avg_diff(0.001784)
[57]	ReLU_relu_4d_double_3x3_1_Y	cosine(0.999355)	avg_diff(0.001740)
[58]	Pooling_avg_pool_4d_pool_Y	cosine(0.998936)	avg_diff(0.003073)
[59]	ReLU_relu_4d_proj_Y	cosine(0.998894)	avg_diff(0.004990)
[60]	ReLU_relu_4e_3x3_reduce_Y	cosine(0.998320)	avg_diff(0.001450)
[61]	ReLU_relu_4e_3x3_Y	cosine(0.998078)	avg_diff(0.003267)
[62]	ReLU_relu_4e_double_3x3_reduce_Y	cosine(0.998879)	avg_diff(0.001392)
[63]	ReLU_relu_4e_double_3x3_0_Y	cosine(0.998773)	avg_diff(0.001278)
[64]	ReLU_relu_4e_double_3x3_1_Y	cosine(0.998659)	avg_diff(0.002214)
[65]	Pooling_max_pool_4e_pool_Y	cosine(0.999229)	avg_diff(0.002959)
[66]	ReLU_relu_5a_1x1_Y	cosine(0.998314)	avg_diff(0.002854)
[67]	ReLU_relu_5a_3x3_reduce_Y	cosine(0.998363)	avg_diff(0.002539)
[68]	ReLU_relu_5a_3x3_Y	cosine(0.997971)	avg_diff(0.001950)

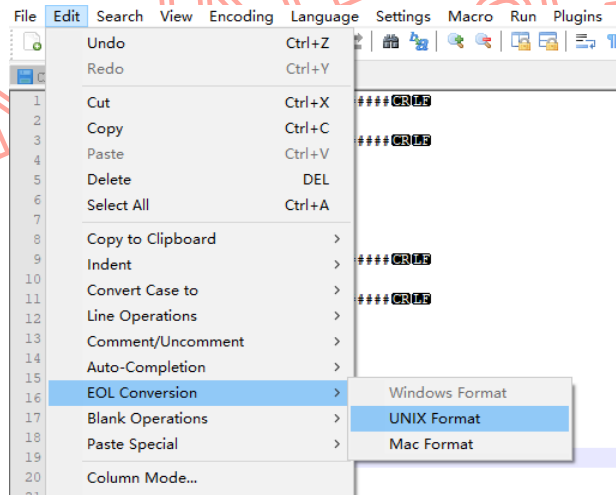
NOVATEK CONFIDENTIAL
NO DISCLOSURE!

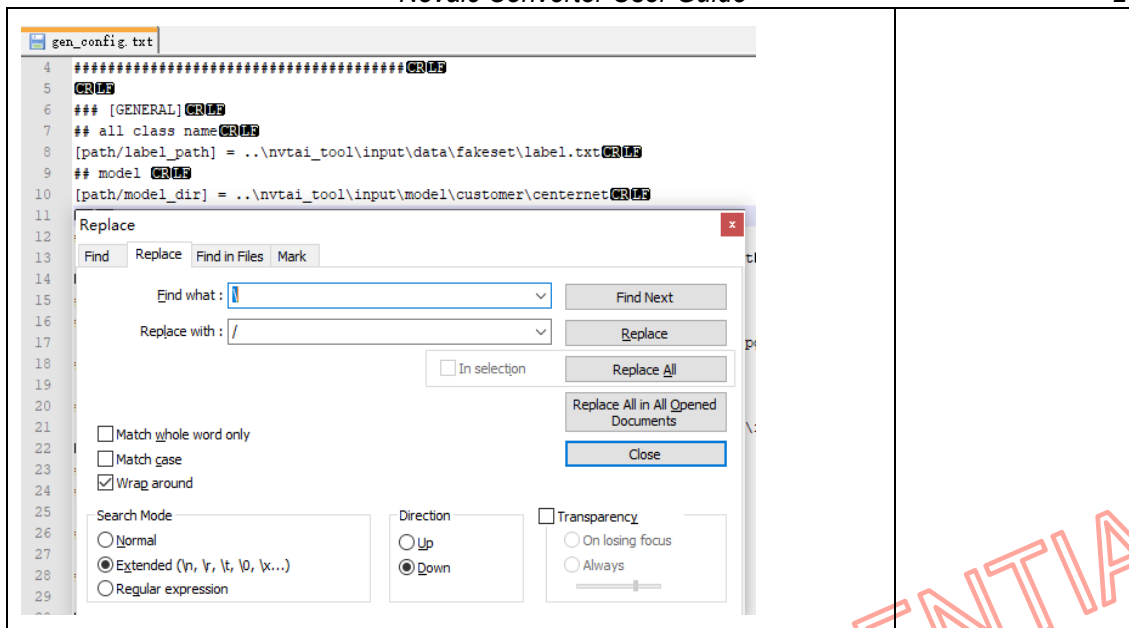
5 Example Introduction

Here, we use 00001_inception_v2 following is an example of test-tutorial to introduce the process tool to convert the network.

5.1 Preprocess of Configure Files

1. The user can use the gen/sim config file under tes-tutorial\nvtai_tool\config\00001_inception_v2, or the user can generate the config file by himself and overwrite the original file

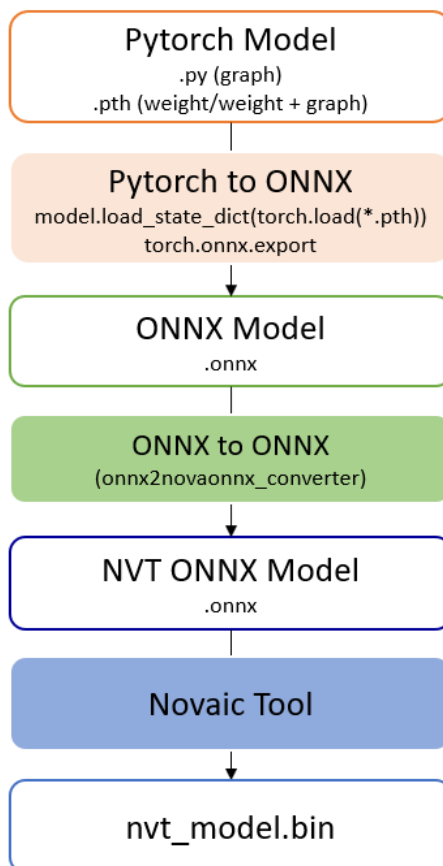
Notice	
Description	Note
<p>If the Win version is used on the Linux system, all .txt files in the input and config folders must be converted to UNIX format in advance.</p> 	<p>Notepad++ can be used for this modification</p>
<p>If Win version of config and input are used on Linux system, the internal path semicolon is changed from '\ ' to '/ '.</p>	<p>Notepad++ can be used for this modification</p>



5.2 ONNX Model Conversion

Since most of the current algorithm models are trained using the Pytorch framework, Novaic Tool can also support the conversion of customers from Pytorch to ONNX model. The following describes the conversion method, which will be divided into two parts

- Pytorch to ONNX
- ONNX to ONNX



5.2.1 Pytorch to ONNX

When Pytorch is converted to ONNX, the first thing to do is to convert the trained Pytorch model file using Pytorch's `torch.onnx.export()`. The API parameter description can refer to the following link:

<https://pytorch.org/docs/stable/onnx.html#torch.onnx.export>

```

torch.onnx.export(model, args, f, export_params=True, verbose=False, training=
<TrainingMode.EVAL: 0>, input_names=None, output_names=None, operator_export_type=None,
opset_version=None, _retain_param_name=None, do_constant_folding=True,
example_outputs=None, strip_doc_string=None, dynamic_axes=None,
keep_initializers_as_inputs=None, custom_opsets=None, enable_onnx_checker=None,
use_external_data_format=None) [SOURCE]
  
```

Here is a simple example:

```

model.eval()

input_names = ["input_1"]
output_names = ["output_1"]

# Input to the model
dummy_input = torch.randn(1, 3, 299, 299, device='cpu')

# Export the model
torch.onnx.export(model, dummy_input, "model_opset12.onnx",
                  opset_version=12, verbose=False, input_names=input_names,
                  export_params=True, do_constant_folding=False)

```

The description of parameter setting is as follows:

- `export_params=True` #When converting the model, the weights must also be saved to the file
- `do_constant_folding=False` #Close here to avoid models with `bn_folding` settings
- `opset_version=12` #The opset version supported by the tool needs to be set here
- `training` #Use default values. Because the onnx model is generally used for inference, it does not need to be set

It should be noted that `eval()` or `train(False)` must be run before using `torch.onnx.export()` to avoid conversion errors. In addition, it should be noted that the opset version supported by Novaic Tool is opset=12. If the Pytorch version does not support opset=12, it can be converted to opset=8 ~ opset=11 first. In the subsequent ONNX to ONNX process, it can be converted to opset=12 through `onnx2novaonnx_converter.py`



Notice

If the model of opset12 is used directly, the onnx version can use 1.7.0, but if the network is not opset12, it will be converted to opset 12 by the onnx converter. At present, the tool needs onnx version 1.9.0 to fully support the function of switching to opset12 , so using models other than opset12 needs to upgrade onnx to 1.9.0, otherwise you may encounter errors that some ops do not support

For pooling op, the parameter `countIncludePad` will not be included when converting Pytorch to ONNX (even if `count_includ_pad` is set). When the ONNX model does not have this parameter, the default value in the tool is 0. At this point, if the calculation

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

area covers the pad, this layer of pooling will become the cpu layer. Although it does not affect the output result, it will affect the performance. Therefore, users need to modify the ONNX model after converting the ONNX model and add the attribute `countIncludPad` to let the tool know whether to include 0 padding in the pool calculation.

5.2.2 ONNX to ONNX

Since the Pytorch converted ONNX model still lacks some information on Novaic Tool, or the opset version needs to be adjusted, the ONNX to ONNX conversion tool is provided in the following path:

```
release\ai_tool\novatek\novaic\toolchain\closeprefix\bin\compiler\frontend\onnx-onnx\  
onnx2novaonnx_converter.py
```

There are two parameters here, input specifies the path of the input onnx model, and output specifies the path of the output onnx model. The instructions are as follows

```
python3 onnx2novaonnx_converter.py -input path/to/input_model -output path/to/output_model
```

After execution, you can get the `deloy.onnx` model that can be used by Novaic Tool in the specified path

If the input is a model other than `opset=12`, it will try to convert it to `opset=12` with onnx converter. Currently, it is determined to support `opset=8 ~ opset=12`. Other opset versions may report errors related to onnx converter. In addition, `opset=8` or `opset=9` to convert `opset=12` requires onnx python lib version 1.9.0 or above

Also note that in order to avoid `bn_folding`, `skip_fuse_bn=True` (line: 159) needs to be enabled in `onnx2novaonnx_converter.py`

```
simplify(onnx_model, skip_fuse_bn=True)
```


5.3 Run Gen-tool Program

1. User can also choose to modify the parameters of tes-tutorial\nvtai_tool\config\00001_inception_v2 to test the example.



Notice

The default test image here is preset to the output bin file (3-Channel, RGB or BGR format) after NUE2 pre-processing

2. Convert caffe model to onnx model

```
python3 /home/[user]/ai_tool/novatek/60ovaic/toolchain/closeprefix/bin/compiler
/frontend/caffe-onnx/convert2onnx.py /home/[user]/ai_tool/novatek/60ovaic/test-tutorial
/nvtai_tool/input/model/customer/inception_v2/deploy.prototxt
/home/[user]/ai_tool/novatek/60ovaic/test-tutorial
/nvtai_tool/input/model/customer/inception_v2/deploy.caffemodel deploy
/home/[user]/ai_tool/novatek/60ovaic/test-
tutorial/nvtai_tool/input/model/customer/inception_v2
```

3. Use Novaic Convertor's Gen-tool (compiler) to convert onnx model to nvt_model.bin
closeprefix/bin/compiler.nvtai -config-dir /home/[user]/ai_tool/novatek/60ovaic/test-
tutorial/nvtai_tool -pattern-name 00001_inception_v2 -chip 336



Notice

The above two points 3. And 4. Can be simplified into the following commands,
which can be completed at one time
closeprefix/bin/compiler.nvtai -config-dir /home/[user]/ai_tool/novatek/60ovaic/test-
tutorial/nvtai_tool -pattern-name 00001_inception_v2 -chip 336 -use-caffe

- [1] The following table summarizes the parameter descriptions used in Gen-tool (compiler)

Para. Name	Value of parameter	Description
------------	--------------------	-------------

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

--config-dir	Set the path where config files are placed	This parameter must be set, you can set an absolute path or a relative path relative to the toochian\ folder
--pattern-name	Set the network name to be converted	This parameter must be set
--use-caffe	-	Setting this parameter can simplify the process and automatically convert the caffe model to the onnx model for tool conversion.
--verbose	0: FATAL 1: ERROR 2: WARNING 3: INDEX 4: USER	The printing level can be set dynamically, the smaller the level value set, the less printing will be, 0 means no printing, and the default level is 4
--quan-en	0: Turn off 8-bit weight compression 1: Turn on 8-bit weight compression	The default value is 0 (turn-off)
--chip	52x: NT9852x 528: NT98528 56x: NT9856x 32x: NT9832x 336: NT98336 331: NT98331 (32bit) 331_A64:NT98331 (64bit) 530: NT98530	No default value, this parameter must be set, and it must be consistent with the chip setting used by the simulator
--msb-shrink-en	0: Original format (complete output) 1: simplify the format at depthwise based net	The default is 0 (turn-off), the simplified format is more efficient, and the model size is smaller, but it may not run in older versions of the AI SDK
--job-opt	0: Debug at linear mode 1: Linear mode	1. When not set, the job-opt option is configured by the EVB

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

	10: Debug at graph mode 11: Graph mode	2. This parameter should be set together with --buf-opt 3. Currently only supports setting 0 or 1 (linear mode)
--buf-opt	0: Do not shrink buff size 1: Shrink buff size 2: Shrink buff size + reorder (O1) 3: Shrink buff size + reorder (O2) 11: Automatically find the best settings	1. When not set, the buf-opt option is configured by the EVB 2. This parameter should be set together with --job-opt
--use-tiling	0: Disable 1: Enable tiling func. (Only consider the setting of iobuf)	The tiling func. Can reduce the side effect of time latency when bandwidth is heavy. The default is 0, It should be noted that when this parameter is not 0, --chip can only be set to 530
--use-heteroge	0: Disable 1: Enable	The default ACL mode is disable. The user can use this flag to enable the ACL mode, and print error message when ONNX model contains the ACL op.
--buf-thld	1 ~ 65535	The threshold of buf configuration complexity when analyzing graph complexity, the default is 312
--graph-thld	1 ~ 65535	The threshold for analyzing graph complexity, the default is 128
--dump-onnx-en	NA	The Dump onnx function is disabled by default. When enabled and --use-tiling is disabled, the user will get 4 debug onnx files in the \gentool folder: onnc_model.onnx, legal_nvt_model.onnx, calib_nvt_model.onnx, codeemit_nvt_model.onnx . If --use-tiling is enabled, there will be an additional

		tiling_nvt_model.onnx file. The user can confirm the results of each module from these files
--	--	--



Notice

Graph analysis is used to estimate if the network is too complicated or not. If the user sets graph mode, buf-opt = 3, there is a risk of too long configuration memory or too large nvt_model. Therefore, it is necessary to reconfigure reasonable parameters by tool. The threshold for judging whether the graph is complex is controlled by buf-thld and graph-thld. Force reconfiguration to linear mode when both are set to 1, and buf-opt = 1, cannot be reconfigured on the board. When both are set to 65535, the graph analysis function is turned off



Notice

If the parameter setting conflicts with the gen/sim config setting, for example, quan_en is set to 1 in cmd line, but when it is set to 0 in gen_config, the setting of cmd line will prevail

5.4 Run Sim-tool Program

1. Use Novaic Convertor's Sim-tool (simulator) and nvt_model.bin to get float & fixed cosine similarity and avg_diff difference
closeprefix/bin/simulator.nvtai --config-dir /home/[user]/ai_tool/novatek/63ovaic/test-tutorial/nvtai_tool --pattern-name 00001_inception_v2 --chip 336

[1] The following table summarizes the parameter descriptions used in Sim-tool (simulator)

Para. Name	Value of parameter	Description
--config-dir	Set the path where config files are placed	This parameter must be set, you can set an absolute path or a relative path relative to the toochian\ folder
--pattern-name	Set the network name to	This parameter must be set

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

	be converted	
--verbose	0: FATAL 1: ERROR 2: WARNING 3: INDEX 4: USER	The printing level can be set dynamically, the smaller the level value set, the less printing will be, 0 means no printing, and the default level is 4
--chip	52x: NT9852x 528: NT98528 56x: NT9856x 32x: NT9832x 336: NT98336 331: NT98331 (32bit) 331_A64:NT98331 (64bit) 530: NT98530	No default value, this parameter must be set, and it must be consistent with the chip setting used by the compiler Note: If the user uses the 32x setting, the input jpg will convert to NV12 format. If the NV21 format used on the 32x board, please convert it to NV12 for the simulator to avoid losing precision.



Notice

Sometimes the network used is large, and the unoptimized io buff size may not meet the hardware limit and cannot generate cos/diff results. You can add --emu-bufopt 1 to cmd to use the optimized io buff size to simulate.

2. After performing the above steps, you can confirm whether the result is correct according to the following methods

- [1] Confirm whether deploy.onnx has been generated
/home/[user]/ai_tool/novatek/64ovaic/test-tutorial/nvtai_tool/input/model/customer/inception_v2/deploy.onnx
- [2] Confirm whether nvt_model.bin is generated
/home/[user]/ai_tool/novatek/64ovaic/test-tutorial/nvtai_tool/output/00001_inception_v2/sdk/nvt_model.bin
- [3] Confirm whether there is a layer_diff file
/home/[user]/ai_tool/novatek/64ovaic/test-tutorial/nvtai_tool/output/00001_inception_v2/simtool/layer_diff.json
/home/[user]/ai_tool/novatek/64ovaic/test-

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

6 The Method of Adding Custom Layer

Tool will classify unsupported layers as custom layers. The parameters of the custom layer are divided into the following three types, which will be passed to the SDK in a predetermined format and can be parsed by themselves.

1. .prototxt document description
2. Weight in .caffemodel file
3. Customized cust_bin file: can bring other parameters and Table information



Notice

Currently supports 66ovaic66r layer is limited to Caffe framework

6.1 The Process of Setting Up A Custom Layer (Caffe)

6.1.1 Caffe

1. First we need to modify the `caffe_nvt.proto` in `./toolchain/closeprefix/bin/compiler/frontend/caffe-onnx/proto/` and add the definition of the custom layer
2. Execute the following commands in the `./toolchain/closeprefix/bin/compiler/frontend/caffe-onnx` directory to update `caffe_nvt_pb2.py`

```
protoc proto/caffe_nvt.proto --python_out ./
```

6.1.2 Gen-tool

Refer to the example of `00001_custom_layer_inception_v2`. This example is the inception_v2 network and specifies the two layers of pool_1 & pool_2 as custom layers.

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

3. First, we need to modify/add the custom layer dedicated config. File (cust_config.txt). You need to set the type, name, platform, do_compare, has_parm, in_num, in_bitdepth, in_sign, out_num, out_bitdepth, out_sign here, the format is as shown in the figure below:

```
##[type]
##[name]
##[platform]
##[do_compare]
##[has_parm]
##[in_num]
##[in_bitdepth]
##[in_sign]
##[out_num]
##[out_bitdepth]
##[out_sign]
```


Take [00001_custom_layer_inception_v2](#) as an example, the result after setting the value is as follows

```
[type]=Pooling
[name]=pool_1;pool_1b;pool_2;
[platform]=CPU
[do_compare]=1
[has_parm]=0
[in_num]=1
[in_bitdepth]=8
[in_sign]=0
[out_num]=1
```

The following is a description of the default value of each item, please refer to the notes to set

Item	Description
[type]	No default value, must be set
[name]	The initial default is all layers of the same type
[platform]	it can be filled with CPU or DSP. The initial value is CPU
[do_compare]	The initial is 0
[has_parm]	The initial value is determined by the judgment of the prototxt parser

[in_num]	The initial value is as the same as the output parameters of the previous layer
[in_bitdepth]	The initial value is as the same as the output parameters of the previous layer
[in_sign]	The initial value is as the same as the output parameters of the previous layer
[out_num]	The initial value is as the same as the output parameters of the previous layer
[out_bitdepth]	The initial value is as the same as the output parameters of the previous layer
[out_sign]	The initial value is as the same as the output parameters of the previous layer

 Notice	
1.	No blank cells are allowed in all fills
2.	[type] is a required item, <u>it must be in the first line</u> , other options can be filled or not filled in, if it is not filled in, it is default value
3.	If "[name]" is filled in, only the custom layer with a specific layer_name will be modified, if not, all the same layer type will be defined as custom layer. You can fill multiple layer_name at once, separated by ",".
4.	Support setting multiple custom layers of different types in the same csut_config file

4. Modify the functions of the custom layer: Open

ai_tool/novatek/68ovaic/toolchain/repo/proprietary/onnc-backends/NvtAI/Source_pub/CustomOP/NvtAICustomerUtility.cpp. The functions that need to be modified are listed below.

[1] Prototxt parameters:

In the prototxt description of the custom layer, the parameters in xx_param{} (as shown below) will be passed in as a string. The user must parse the custom layer parameters by himself and store the parameters in layer_parm (custom struct form , Such as CUSTNN_POOL_PROTO_PARM in NvtAICustomerUtility.h), please refer to the implementation of parsing_proto_parm().

NvtAICustomerUtility.h is saved at the following path:

novatek/68ovaic/toolchain/repo/proprietary/onnc-backends/NvtAI/

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

NvtAlCustomerUtility.h

```
layer {
  bottom: "relu_1"
  top: "pool_1"
  name: "pool_1"
  top: "Pooling"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
    pad: 0
  }
}
```

For example, call the processPooling function in parsing_proto_parm() to parse the above content. After parsing, it will be stored in the Public variable (op.layer_parm.data()) of IR (NvtAlCustomer& op), which can be obtained directly

```
int processPooling(std::string protoParam, std::vector<unsigned char>& layer_parm) {
  CUSTNN_POOL_PROTO_PARM custPoolParm = {0};
  size_t idx1, idx2;
  std::vector<std::string> keyword = {"pool:", "kernel_size:", "stride:", "pad:", };
  std::string value;

  for(auto key : keyword) {
    idx1 = protoParam.find(key);
    idx2 = protoParam.find('/');
    if ((idx1 != std::string::npos) && (idx2 != std::string::npos)) {
      value = protoParam.substr(idx1+key.size(), (idx2-idx1-key.size()));
    }
    protoParam.erase(0, idx2 + 1);

    if (key == "pool:") {
      if (value == "MAX") {
        custPoolParm.pool_type == 0;
      } else {
        custPoolParm.pool_type == 1;
      }
    } else if (key == "kernel_size:") {
      custPoolParm.kerl_sz = std::stoi(value);
    } else if (key == "stride:") {
      custPoolParm.stride = std::stoi(value);
    } else if (key == "pad:") {
      custPoolParm.pad = std::stoi(value);
    }
  }
  auto ptr = reinterpret_cast<unsigned char*>(&custPoolParm);
  layer_parm = std::vector<unsigned char>(ptr, ptr + sizeof(CUSTNN_POOL_PROTO_PARM));
  return 0;
}
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

[2] Reshape function: (similar to Reshape in Caffe xxx_layer.cpp)

The user can calculate the dimensional information (Tensor Shape) of each output of the private layer by himself, please refer to the implementation of ReshapeCustomer(). If there are multiple outputs, you need to set multiple. Take the following code as the example:

```
else if (strcmp(op.layer_type.c_str(), "Slice") == 0) {
    Tensor* input = dynamic_cast<Tensor*>(op.getInput(0));
    int output_num = op.getNumOfOutputs();
    std::vector<int64_t> new_dim = input->dimensions();

    CUSTNN_SLICE_PROTO_PARM*p_custSliceParm =
    (CUSTNN_SLICE_PROTO_PARM*)op.layer_parm.data();
    int32_t axis = p_custSliceParm->axis;
    int32_t* slice_point = p_custSliceParm->slice_point;

    for(int i = 0; i < output_num; i++) {
        Tensor* output_i = dynamic_cast<Tensor*>(op.getOutput(i));
        if (i == 0) {
            new_dim.at(axis) = slice_point[i];
        }
        else if (i == output_num - 1) {
            new_dim.at(axis) = input->dimension(axis) - slice_point[i-1];
        }
        else {
            new_dim.at(axis) = slice_point[i] - slice_point[i-1];
        }
        output_i->setDimensions(new_dim);
    }
}
```

[3] fp32 inference function: (similar to Forward_cpu in Caffe xxx_layer.cpp)

For the user-defined fp32 inference calculation of the custom layer, please refer to the implementation of computeCustomer().

```
Else if (strcmp(op.layer_type.c_str(), "Eltwise") == 0) {
    const int idx = 1;
    CalibrationTensor<float> secondInput{*op.getInput(idx),
addressTable().at(op.getInput(idx))};
    Tensor::Dimensions second_dims(secondInput.dims().begin(),
secondInput.dims().end());
    auto *secondDataPtr = secondInput.data();

    auto len = 1;
    for (auto dim : out_dims) {
        len *= dim;
    }
}
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```

for (auto idx=0; idx<len; idx++) {
    outDataPtr[idx] = inDataPtr[idx] + secondDataPtr[idx];
}
}

```

[4] tmp_buffer information:

The user calculates the required size of tmp_buffer. The size must be greater than or equal to the total buffer size required by the input feature (add up if there are multiple inputs). Refer to the implementation of getCustomerTempBufferSize().

```

Size_t getCustomerTempBufferSize(const NvtAICustomer& customer) {
    size_t bufferSize = 0, InputSize = 0;

    for (size_t in = 0; in < customer.getNumOfCustInputs(); in++) {
        const Tensor* input = static_cast<const Tensor*> (customer.getInput(in));
        assert(input != NullTensor::singleton());
        // calculate each input size
        InputSize = 1;
        for (size_t idx=0; idx<input->dimensions().size(); idx++) {
            InputSize *= input->dimension(idx);
        }
        bufferSize += InputSize * customer.in_bitdepth >> 3;
    }
    return bufferSize;
}

```

Since getCustomerTempBufferSize is implemented after Calibration, if users want to get the real quantized result, they can get it here.

[5] Get the quantized information of the current custom layer:

If the user wants to obtain the information of the current private layer such as

i. Input/output size [n, c, h, w]. It can be obtained by

```

void ReshapeCustomer(NvtAICustomer& op) {
    Tensor* input = dynamic_cast<Tensor*>(op.getInput(0));
    Tensor* output = dynamic_cast<Tensor*>(op.getOutput(0));
}

```

ii. Bitdepth, sign_bit_num, int_bit_num, frac_bit_num, isf/osf: Those info. Are saved in the Public variable of IR (NvtAICustomer& op), which can be got directly.

```
Size_t getCustomerTempBufferSize(const NvtAICustomer& customer) {
    .
    .
    bufferSize += InputSize * customer.in_bitdepth >> 3;
}
return bufferSize;
};
```

- iii. The input size [n, c, h, w] of the entire network is recorded in the public variable `std::vector<TensorInfo> net_input_infos` of `NvtAICustomer`;

The structure definition is as follows:

```
struct TensorInfo final
{
    std::string name;
    std::vector<int64_t> dims;
};
```

The case of usage is as follows:

```
size_t getCustomerTempBufferSize(const NvtAICustomer& customer) {
    .
    .
    for(auto& net_input_info : customer.net_input_infos) {
        std::string net_input_name = net_input_info.name;
        std::int64_t n = net_input_info.dims[0];
        std::int64_t c = net_input_info.dims[1];
        std::int64_t h = net_input_info.dims[2];
        std::int64_t w = net_input_info.dims[3];
        outs()<<"Net input name = "<<net_input_name<<", n = "<<n<<", c = "<<c<<", h =
        "<<h<<", w = "<<w<<std::endl;
    }
    .
    .
}
```

5. After the above function is modified, return to the toolchain directory to execute *make rd-onnc-app*



Notice

1. The weight in caffemodel will be automatically read for packing, no additional settings are required
2. If you have additional parameters you want to pass, you can use `custom_bin` (not

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

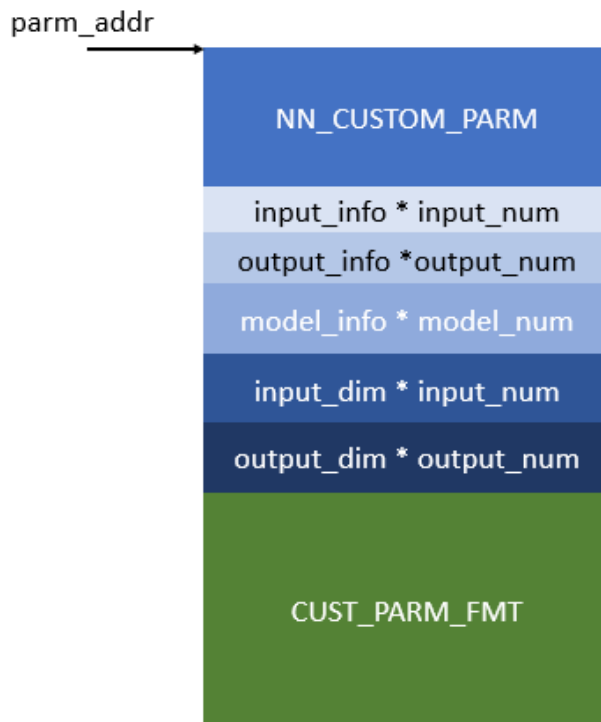
With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

necessary). Each value is packaged into a bin file in units of 4 bytes. The file name is layer_name for each layer and placed in the specified location (..\nvtai_tool\input\model\[model_name]\custom_bin) For examples, please refer to: pool_1.bin & pool_2.bin in ..\nvtai_tool\input\model\ inceptionv2_custom\custom_bin.

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

6.1.3 Sim-tool

1. Modify the source code vendor_ai_cpu_custnn.c, and write a custom function vendor_ai_cust() according to the definition parameters passed by Gen-tool. The format of custom parameters is as follows:



[1] NN_CUSTOM_PARM structure is as follows

```
typedef struct _NN_CUSTOM_PARM {
    UINT32 input_num;
    UINT32 output_num;
    UINT32 model_num;
    ai_ptr_32 temp_buf_addr;
    UINT32 temp_buf_size;
    UINT32 parm_size;
    /*
    NN_DATA* input;    // size = sizeof(NN_DATA)*input_num
    NN_DATA* output;  // size = sizeof(NN_DATA)*output_num
    NN_DATA* model;   // size = sizeof(NN_DATA)*model_num
    NN_CUSTOM_DIM* input_dim; // size = sizeof(NN_CUSTOM_DIM)*input_num
    NN_CUSTOM_DIM* output_dim; // size = sizeof(NN_CUSTOM_DIM)*output_num
    ...
    custom parameters
    */
} NN_CUSTOM_PARM;
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

[2] CUST_PARM_FMT format is as follows:

Item	Type	Description
layer_type_id	UINT32	idx of the custom layer defined by ai_gen, starting from 0
weight_num weight_size_1 weight_size_2 ...	UINT32	The number of weights of this custom layer in caffemodel, followed by the size of each weight
prototxt_parm_len	UINT32	The length of the string passed from prototxt
isf_len	UINT32	The length of the input scale-shift
isf_mul_1	UINT32	The multiplier factor of input
isf_shift_1	UINT32	The shift of input
osf_len	UINT32	The length of the output scale-shift
osf_mul_1	UINT32	The multiplier factor of output
osf_shift_1	UINT32	The shift of output
bin_parm_len	UINT32	The length of cust_bin
bin_parm		cust_bin data

- File path:
 - 32bits:

ai_tool/novat75ovaicaic/toolchain/repo/proprietary/runtime/hdal/vendor/a

i2/source_pub/vendor_ai_cpu/vendor_ai_cpu_custnn.c
 - 64bits:

ai_tool/novat75ovaicaic/toolchain/repo/proprietary/runtime64/hdal/vendo

r/ai2/source_pub/vendor_ai_cpu/vendor_ai_cpu_custnn.c

[3] The sample code for obtaining necessary information is as follows:

- i. Obtain the parameters required for the current custom layer

```
HD_RESULT vendor_ai_cpu_cust(uintptr_t parm_addr, UINT32 net_id)
{
    /*
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```

        custom parm format:
        1. custom_layer_type_id(UINT32)
        2. weight_num(UINT32), weight_size_1(UINT32), weight_size_2(UINT32), ...
           (weights at p_head->model.va)
        3. prototxt_parm_len(UINT32),
        4. isf_len(INT32), isf_mul_1(INT32), isf_shift_1(INT32), isf_mul_2(INT32),
           isf_shift_2(INT32), ...
        5. osf_len(INT32), osf_mul_1(INT32), osf_shift_1(INT32), osf_mul_2(INT32),
           osf_shift_2(INT32), ...
        6. bin_parm_len(UINT32), p_prototxt_parm, p_bin_parm
        7. tmp_buf addr(NN_DATA)

    */
    NN_CUSTOM_PARM *p_head = (NN_CUSTOM_PARM *) (parm_addr);
#if CUST_SUPPORT_MULTI_IO
    UINT32 input_num = p_head->input_num;
    UINT32 output_num = p_head->output_num;
    UINT32 model_num = p_head->model_num;
    NN_DATA* input_info = (NN_DATA*) (parm_addr + sizeof(NN_CUSTOM_PARM));
    NN_DATA* output_info = (NN_DATA*) (parm_addr + sizeof(NN_CUSTOM_PARM) +
input_num*sizeof(NN_DATA));
    uintptr_t dim_addr = (uintptr_t) (parm_addr + sizeof(NN_CUSTOM_PARM) +
(input_num+output_num+model_num)*sizeof(NN_DATA));
    NN_CUSTOM_DIM* input_dim = (NN_CUSTOM_DIM*) (dim_addr);
    NN_CUSTOM_DIM* output_dim = (NN_CUSTOM_DIM*) (dim_addr +
sizeof(NN_CUSTOM_DIM)*input_num);
    UINT32 *p_layer_type_id = (UINT32 *) (dim_addr +
sizeof(NN_CUSTOM_DIM)*(input_num+output_num));
#else
    UINT32 *p_layer_type_id = (UINT32 *) (p_head + 1);
#endif
    INT32 layer_type_id = (INT32) (*p_layer_type_id);
    UINT32 *p_weight_num = (UINT32 *) (p_layer_type_id + 1);
    UINT32 weight_num = *p_weight_num;
    UINT32 *p_prototxt_parm_len = (UINT32 *) (p_weight_num + 1 + weight_num);

    UINT32 *p_isf_len = (UINT32 *) (p_prototxt_parm_len + 1);
    UINT32 isf_num = (*p_isf_len) / 8;
    INT32 *p_isf_parm = (INT32 *) (p_isf_len + 1);
    UINT32 *p_osf_len = (UINT32 *) (p_isf_parm + isf_num*2);
    UINT32 osf_num = (*p_osf_len) / 8;
    INT32 *p_osf_parm = (INT32 *) (p_osf_len + 1);

```

- ii. The input size [n, c, h, w] of the entire network, use HD_RESULT
_vendor_ais_get_net_input_info_list() to store it in
VENDOR_AI_NET_INPUT_INFO

```

typedef struct _VENDOR_AI_NET_INPUT_INFO {
    CHAR   name[192];           ///< buffer name, e": "mylayer.o"t0"
    UINT32 width;               ///< width
    UINT32 height;              ///< height
    UINT32 channel;             ///< channel
    UINT32 batch;               ///< number of batch

```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.


```
} VENDOR_AI_NET_INPUT_INFO;
```

The usage example is as follows:

```
HD_RESULT custnn_cpu_elt_process(CUSTNN_ELT_PARM* p_parm, UINT32 in0_addr, UINT32
in1_addr, UINT32 out_addr)
{
    . . .
    VENDOR_AI_NET_INPUT_INFO net_input_infos[32] = {0};
    _vendor_ais_get_net_input_info_list(0, net_input_infos);
    printf("net_input_infos[0].name = %s\n", net_input_infos[0].name);
    printf("net_input_infos[0].width = %d\n", net_input_infos[0].width);
    printf("net_input_infos[0].height = %d\n", net_input_infos[0].height);
    printf("net_input_infos[0].channel = %d\n", net_input_infos[0].channel);
    printf("net_input_infos[0].batch = %d\n", net_input_infos[0].batch);
    printf("net_input_infos[1].name = %s\n", net_input_infos[1].name);
    printf("net_input_infos[1].width = %d\n", net_input_infos[1].width);
    printf("net_input_infos[1].height = %d\n", net_input_infos[1].height);
    printf("net_input_infos[1].channel = %d\n", net_input_infos[1].channel);
    printf("net_input_infos[1].batch = %d\n", net_input_infos[1].batch);
    . . .
}
```

- [4] After obtaining the input/output info., the user can use a self-defined structure; take CUSTNN_POOL_PARM as an example, this structure can be described or adjusted in the following file

i. 32bits:

/ai_tool/novat77ovaicaic/toolchain/repo/proprietary/runtime/hdal/vendor/ai2/s
ource_pub/vendor_ai_cpu/vendor_ai_cpu_custnn_sample.h

ii. 64bits:

/ai_tool/novat77ovaicaic/toolchain/repo/proprietary/runtime64/hdal/vendor/ai2
/source_pub/vendor_ai_cpu/vendor_ai_cpu_custnn_sample.h

範例代碼如下:

```
HD_RESULT vendor_ai_cpu_cust(uintptr_t parm_addr, UINT32 net_id)
{
    . . .

    #if 0
        UINT32 *p_bin_parm_len = (UINT32 *) (p_osf_parm + osf_num*2);
        CUSTNN_ELT_PARM *p_elt_parm = (CUSTNN_ELT_PARM *) (p_bin_parm_len + 1);
        CUSTNN_POOL_PARM *p_pool_parm = (CUSTNN_POOL_PARM *) (p_bin_parm_len + 1);
    #if CUST_SUPPORT_MULTI_IO
        CUSTNN_CONCAT_PARM *p_concat_parm = (CUSTNN_CONCAT_PARM *) (p_bin_parm_len + 1);
        CUSTNN_SLICE_PARM *p_slice_parm = (CUSTNN_SLICE_PARM *) (p_bin_parm_len + 1);
    #endif
}
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```
#else
    UINT32 *p_bin_parm_len = (UINT32 *) (p_osf_parm + osf_num * 2);
    CHAR *p_prototxt_parm = (CHAR *) (p_bin_parm_len + 1);
    CHAR *p_bin_parm = (CHAR *) (p_prototxt_parm + *p_prototxt_parm_len);

    CUSTNN_ELT_PARM *p_elt_parm = (CUSTNN_ELT_PARM *) (p_bin_parm);
    CUSTNN_POOL_PARM *p_pool_parm = (CUSTNN_POOL_PARM *) (p_bin_parm);
#if CUST_SUPPORT_MULTI_IO
    CUSTNN_CONCAT_PARM *p_concat_parm = (CUSTNN_CONCAT_PARM *) (p_bin_parm);
    CUSTNN_SLICE_PARM *p_slice_parm = (CUSTNN_SLICE_PARM *) (p_bin_parm);
#endif
#endif
```

- [5] After retrieving the self-defined structure parameters and input/output address, it will call the calculation API for processing to complete the calculation of the custom layer, as described in the following example:

```
if (layer_type_id == get_hashco("e("Pool"ng")) {
    // pooling
#if CUST_SUPPORT_MULTI_IO
    custnn_cpu_scaleshift_process(input_info[0].va, p_head->temp_buf_addr,
input_info[0].size, p_pool_parm->in_type, p_isf_parm[0], p_isf_parm[1]);
    custnn_cpu_pool_process(p_pool_parm, p_head->temp_buf_addr,
output_info[0].va);
    custnn_cpu_scaleshift_process(output_info[0].va, output_info[0].va,
output_info[0].size, p_pool_parm->out_type, p_osf_parm[0], p_osf_parm[1]);
#else
    custnn_cpu_scaleshift_process(p_head->input.va, p_head->input.va, p_head->
input.size, p_pool_parm->in_type, p_isf_parm[0], p_isf_parm[1]);
    custnn_cpu_pool_process(p_pool_parm, p_head->input.va, p_head->output.va);
    custnn_cpu_scaleshift_process(p_head->output.va, p_head->output.va, p_head->
output.size, p_pool_parm->out_type, p_osf_parm[0], p_osf_parm[1]);
#endif
} else if (layer_type_id == get_hashco("e("Eltw"se")) {
```

2. Compile related libraries of custom layer

Under /toolchain, please execute

make clean; make;



Notice

When using make cmd, the following commands will be executed automatically

1. make custom_op (generate libCustomOP.a, libCustomOP.so to
ai_tool/novat78ovaicaic/toolchain/closeprefix/lib)
2. make runtime_ai2_pub (generate libvendor_ai2_pub.so for 32-bits to

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```
ai_tool/novat79ovaicaic/toolchain/closeprefix/bin/simulator/emu52X)  
3. make runtime64_ai2_pub (generate libvendor_ai2_pub.so for 64-bits to  
ai_tool/novat79ovaicaic/toolchain/closeprefix/bin/simulator/emu33X)
```

3. The modified vendor_ai_cpu_custnn.c can be directly copied to the file code/hdal/vendor/ai2/source_pub/vendor_ai_cpu/vendor_ai_cpu_custnn.c under the corresponding AI2 SDK path.

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

6.1.4 AI2 SDK

1. Modify the source code `vendor_ai_cpu_custnn.c`, and write a custom function `vendor_ai_cust()` according to the definition parameters passed by Gen-tool; its steps must be consistent with Step 1 of Sim-tool

Take the pooling layer of 00001_custom_layer_inception_v2 as an example: In the `vendor_ai_cpu_cust()` on the AI2 sdk code, some custom information will be captured first in the first half, where the structure of CUSTNN_POOL_PARM is defined in `~\hda\vendor\ai2\source_pub\source_pub\ vendor_ai_cpu_custnn_sample.h`

2. ,

Users can adjust the structure according to their own needs. After obtaining some information, users can add calculations and processing according to their own needs. In this example, after we retrieve the pooling parameters and input/output address, we call The calculation API performs processing to complete the pooling calculation, the example is as follo

3. ws

```
HD_RESULT vendor_ai_cpu_cust(UINT32 parm_addr, UINT32 net_id)
{
    /*
     custom parm format:
     1. custom_layer_type_id(UINT32)
     2. weight_num(UINT32), weight_size_1(UINT32), weight_size_2(UINT32), ...
        (weights at p_head->model.va)
     3. prototxt_parm_len(UINT32),
     4. isf_len(INT32), isf_mul_1(INT32), isf_shift_1(INT32), isf_mul_2(INT32),
        isf_shift_2(INT32), ...
     5. osf_len(INT32), osf_mul_1(INT32), osf_shift_1(INT32), osf_mul_2(INT32),
        osf_shift_2(INT32), ...
     6. bin_parm_len(UINT32), p_prototxt_parm, p_bin_parm
     7. tmp_buf addr(NN_DATA)
    */
    NN_CUSTOM_PARM *p_head = (NN_CUSTOM_PARM *) (parm_addr);
    #if CUST_SUPPORT_MULTI_IO
        UINT32 input_num = p_head->input_num;
        UINT32 output_num = p_head->output_num;
        UINT32 model_num = p_head->model_num;
        NN_DATA* input_info = (NN_DATA*) (parm_addr + sizeof(NN_CUSTOM_PARM));
        NN_DATA* output_info = (NN_DATA*) (parm_addr + sizeof(NN_CUSTOM_PARM) +
        input_num*sizeof(NN_DATA));
        UINT32 dim_addr = (UINT32) (parm_addr + sizeof(NN_CUSTOM_PARM) +
        (input_num+output_num+model_num)*sizeof(NN_DATA));
        NN_CUSTOM_DIM* input_dim = (NN_CUSTOM_DIM*) (dim_addr);
    #endif
}
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```
    NN_CUSTOM_DIM* output_dim = (NN_CUSTOM_DIM*)(dim_addr + sizeof(NN_CUSTOM_DIM)*input_num);
    UINT32 *p_layer_type_id = (UINT32 *)(dim_addr +
sizeof(NN_CUSTOM_DIM)*(input_num+output_num));
#else
    UINT32 *p_layer_type_id = (UINT32 *)(p_head + 1);
#endif
    INT32 layer_type_id = (INT32)(*p_layer_type_id);
    UINT32 *p_weight_num = (UINT32 *)(p_layer_type_id + 1);
    UINT32 weight_num = *p_weight_num;
    UINT32 *p_prototxt_parm_len = (UINT32 *)(p_weight_num + 1 + weight_num);

    UINT32 *p_isf_len = (UINT32 *)(p_prototxt_parm_len + 1);
    UINT32 isf_num = (*p_isf_len) / 8;
    INT32 *p_isf_parm = (INT32 *)(p_isf_len + 1);
    UINT32 *p_osf_len = (UINT32 *)(p_isf_parm + isf_num*2);
    UINT32 osf_num = (*p_osf_len) / 8;
    INT32 *p_osf_parm = (INT32 *)(p_osf_len + 1);
    UINT32 *p_bin_parm_len = (UINT32 *)(p_osf_parm + osf_num * 2);
    CHAR *p_prototxt_parm = (CHAR *)(p_bin_parm_len + 1);
    CHAR *p_bin_parm = (CHAR *)(p_prototxt_parm + *p_prototxt_parm_len);

    CUSTNN_ELT_PARM *p_elt_parm = (CUSTNN_ELT_PARM *)(p_bin_parm);
    CUSTNN_POOL_PARM *p_pool_parm = (CUSTNN_POOL_PARM *)(p_bin_parm);
#if CUST_SUPPORT_MULTI_IO
    CUSTNN_CONCAT_PARM *p_concat_parm = (CUSTNN_CONCAT_PARM *)(p_bin_parm);
    CUSTNN_SLICE_PARM *p_slice_parm = (CUSTNN_SLICE_PARM *)(p_bin_parm);
#endif
```

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

6.2 The Process of Setting Up A Custom Layer (ONNX)

6.2.1 ONNX

1. Prepare the ONNX model containing the ONNX custom layer, which can be generated in the following way:

- [1] According to official documents, generate ONNX model with ONNX private layer from pytorch
(<https://github.com/onnx/tutorials/tree/main/PyTorchCustomOperator>)
- [2] Compared with the general ONNX model, the self-built ONNX model has two additional places that need to be added. The first is that the opset_import field of ModelProto needs to add the name and version of the custom domain as follows:



In this example, the domain name is defined as mydomain, and the version is 1. The way to add domain name and version can refer to the following python script

```
import onnx
import copy

onnx_model = onnx.load('in_model.onnx')

copy_op_set_import = copy.deepcopy(onnx_model.opset_import[0])
copy_op_set_import.domain = "mydomain"
copy_op_set_import.version = 1
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

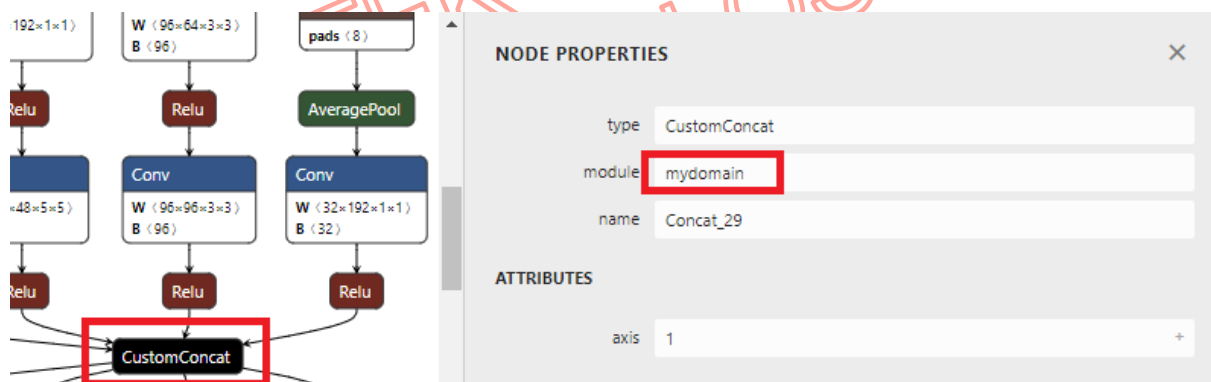
```
onnx_model.opset_import.append(copy_op_set_import)
print(onnx_model.opset_import)

onnx.save(onnx_model, 'out_model.onnx')
```

If you print model.opset_import, you can see the following results, the empty string domain is the onnx standard opset domain (empty string or ai.onnx are both considered as the onnx standard opset)

```
nvt04631@oaaalnx53:~/onnx_tool$ python3 DumpOnnxDomain.py --input custom_inception_v3.onnx
input: custom_inception_v3.onnx
[domain: ""
version: 12
, domain: "mydomain"
version: 1
]
```

In addition, the domain field needs to be added to the NodeProto of each Custom OP, which is the same as the name of the above custom domain



How to add can refer to the following python script

```
import onnx
import copy

onnx_model = onnx.load('in_model.onnx')

graph = onnx_model.graph
for i in range(len(graph.node)):
    if graph.node[i].op_type == 'CustomConcat':
```

```
graph.node[i].domain = "mydomain"  
print(graph.node[i])
```

```
onnx.save(onnx_model, 'out_model.onnx')
```

The result of printing NodeProto is as follows

```
nvt04631@oaalnx53:~/onnx_tool$ python3 DumpCustomNode.py --input custom_inception_v3.onnx  
input: custom_inception_v3.onnx  
input: "600"  
input: "606"  
input: "615"  
input: "621"  
output: "622"  
name: "Concat_29"  
op_type: "CustomConcat"  
attribute {  
  name: "axis"  
  i: 1  
  type: INT  
}  
domain: "mydomain"
```



Notice

The ONNX model with custom layer prepared above still needs to be converted to deploy.onnx with onnx2novaonnx_converter before using Novaic Tool to convert nvt_model.bin

6.2.2 Gen-tool

1. The setting of cust_config.txt is the same as that of caffe custom layer, please refer to the setting of cust_config.txt in 6.1.2.
2. In order to use onnxruntime to obtain the custom layer output shape and inference results, it is necessary to implement the custom layer on onnxruntime and register the custom layer OP with onnxruntime, please build according to the official document (<https://github.com/onnx/tutorials/tree/main/PyTorchCustomOperator>), open ai_tool/novatek/novaic/toolchain/repo/proprietary/onnc-backends/NvtAI/Source_pub/CustomOP, modify custom_op.h and custom_op.cc here, the example of Concat custom layer is as follows

```
//custom_op.h
template <typename T>
struct CustomConcatKernel {
    private:
        int64_t axis_;
        Ort::CustomOpApi ort_;

    public:
        CustomConcatKernel(Ort::CustomOpApi ort, const OrtKernelInfo* info) : ort_(ort) {
            axis_ = ort_.KernelInfoGetAttribute<int64_t>(info, "axis");
        }

        void Compute(OrtKernelContext* context);
};

struct CustomConcatCustomOp : Ort::CustomOpBase<CustomConcatCustomOp,
CustomConcatKernel<float>> {
    void* CreateKernel(const Ort::CustomOpApi api, const OrtKernelInfo* info) const { return new
CustomConcatKernel<float>(api, info); };
    const char* GetName() const { return "CustomConcat"; };

    size_t GetInputTypeCount() const { return 4; };
    ONNXTensorElementType GetInputType(size_t /*index*/) const { return
ONNX_TENSOR_ELEMENT_DATA_TYPE_FLOAT; };
};
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```
size_t GetOutputTypeCount() const { return 1; };  
ONNXTensorElementType GetOutputType(size_t /*index*/) const { return  
ONNX_TENSOR_ELEMENT_DATA_TYPE_FLOAT; };  
};
```

```
//custom_op.cc  
template <typename T>  
void CustomConcatKernel<T>::Compute(OrtKernelContext* context) {  
  
    const OrtValue* input_0 = ort_.KernelContext_GetInput(context, 0);  
    OrtTensorDimensions out_dim(ort_, input_0);  
  
    //Do reshape  
    for (int i = 1; i < ort_.KernelContext_GetInputCount(context); ++i) {  
        const OrtValue* input_i = ort_.KernelContext_GetInput(context, i);  
        OrtTensorDimensions dimensions(ort_, input_i);  
        out_dim[axis_] += dimensions[axis_];  
    }  
  
    //Setup output  
    OrtValue* output = ort_.KernelContext_GetOutput(context, 0, out_dim.data(), out_dim.size());  
    float* out = ort_.GetTensorMutableData<float>(output);  
  
    // Do computation  
    int64_t slices = shape_size(out_dim, axis_, 0);  
    int64_t block = shape_size(out_dim, out_dim.size(), axis_ + 1);  
  
    for (int64_t j = 0; j < slices; ++j) {  
        for (int64_t i = 0; i < ort_.KernelContext_GetInputCount(context); ++i) {  
            const OrtValue* input_i = ort_.KernelContext_GetInput(context, i);  
            const T* i_data = reinterpret_cast<const T*>(ort_.GetTensorData<T>(input_i));  
            OrtTensorDimensions dimensions(ort_, input_i);  
            int64_t length = dimensions[axis_] * block;  
            memcpy(out, i_data + j * length, length * sizeof(float));  
            out += length;  
        }  
    }  
}
```

```

    }
  }
}

```

3. Modify self-defined custom layer functions:

Open `ai_tool/novatek/novaic/toolchain/repo/proprietary/onnc-backends/NvtAI/Source_pub/CustomOP/NvtAICustomerUtility.cpp`, and the functions that need to be modified are listed below.

[1] AttributeProto parameters:

In the AttributeProto description of the custom layer (the following example), all parameters will be passed in in the form of a composite string. Users must parse the custom layer parameters by themselves, and store the parameters in `layer_parm` (customized struct form, such as `NvtAICustomerUtility.CUSTNN_CONCAT_PROTO_PARM` in `h`)

The file location of `NvtAICustomerUtility.h` is in `novatek/novaic/toolchain/repo/proprietary/onnc-backends/NvtAI/NvtAICustomerUtility.h`

Please refer to the implementation of `parsing_proto_parm()`.

```

attribute {
  name: "axis"
  i: 1
  type: INT
}

```

For example, call the `processConcat` function in `parsing_proto_parm()` to parse the above content. After parsing, it will be stored in the Public variable (`op.layer_parm.data()`) of IR (`NvtAICustomer& op`), which can be obtained directly

```

int processConcat(std::string protoParam, std::vector<unsigned char>& layer_parm) {
  CUSTNN_CONCAT_PROTO_PARM custConcatParm = {0};
  std::vector<std::string> keyword = {"axis:"};

  for(auto key : keyword) {

```

```

size_t idx1 = protoParam.find(key);
size_t idx2 = protoParam.find('/');
std::string value;
if ((idx1 != std::string::npos) && (idx2 != std::string::npos)) {
    value = protoParam.substr(idx1+key.size(), (idx2-idx1-key.size()));
    protoParam.erase(0, idx2 + 1);
}

if (key == "axis:") {
    custConcatParm.axis = value.empty() ? 1 : std::stoi(value);
}
}
auto ptr = reinterpret_cast<unsigned char*>(&custConcatParm);
layer_parm = std::vector<unsigned char>(ptr, ptr + sizeof(CUSTNN_CONCAT_PROTO_PARM));
return 0;
}

```

[2] Reshape function:

The custom layer op has been implemented on onnxruntime before, and this custom layer op is added to the self-defined custom op domain during Reshape (the domain on the onnx model, refer to 6.2.1), and then call ReshapeWithCustomOpDomain (custom_op_domain, op), you can Refer to the implementation of ReshapeCustomer()

```

else if (strcmp(op.layer_type.c_str(), "CustomConcat") == 0) {

    CustomConcatCustomOp custom_op;
    Ort::CustomOpDomain custom_op_domain("mydomain");
    custom_op_domain.Add(&custom_op);

    ReshapeWithCustomOpDomain(custom_op_domain, op);
}

```

[3] fp32 inference function:

The custom layer op has been implemented on the onnxruntime before. When fp32 inference, add this custom layer op to the self-defined custom op domain (the domain on the onnx model, refer to 6.2.1), and then call

ComputeWithCustomOpDomain (custom_op_domain, op), Refer to the implementation of ComputeCustomer()

```
else if (strcmp(op->layer_type.c_str(), "CustomConcat") == 0) {

    CustomConcatCustomOp custom_op;
    Ort::CustomOpDomain custom_op_domain("mydomain");
    custom_op_domain.Add(&custom_op);
    ComputewithCustomOpDomain(custom_op_domain, op);
}
```

[4] tmp_buffer data:

Users can calculate the required size of tmp_buffer by themselves, and the size must be greater than or equal to the total buffer size required by the input feature (if there are multiple inputs, they must be added together). Please refer to the implementation of getCustomerTempBufferSize().

```
Size_t getCustomerTempBufferSize(const NvtAICustomer& customer) {
    size_t bufferSize = 0, InputSize = 0;

    for (size_t in = 0; in < customer.getNumOfCustInputs(); in++) {
        const Tensor* input = static_cast<const Tensor*> (customer.getInput(in));
        assert(input != nullptr::singleton());
        // calculate each input size
        InputSize = 1;
        for (size_t idx=0; idx<input->dimensions().size(); idx++) {
            InputSize *= input->dimension(idx);
        }
        bufferSize += InputSize * customer.in_bitdepth >> 3;
    }
    return bufferSize;
}
```

Since getCustomerTempBufferSize is implemented after Calibration, if users want to obtain the real quantized results, they can obtain them here.

[5] Get information about the quantization of the current custom layer:

If the user wants to obtain the quantitative information of the current private layer such as

- i. The input and output size [n, c, h, w] can be obtained by the following methods

```
void ReshapeCustomer(NvtAICustomer& op) {
    Tensor* input = dynamic_cast<Tensor*>(op.getInput(0));
    Tensor* output = dynamic_cast<Tensor*>(op.getOutput(0));
```

- ii. Bitdepth, sign_bit_num, int_bit_num, frac_bit_num, isf/osf are stored in the Public variables of IR (NvtAICustomer& op), and can be obtained directly

```
size_t getCustomerTempBufferSize(const NvtAICustomer& customer) {
    .
    .
    bufferSize += InputSize * customer.in_bitdepth >> 3;
}
return bufferSize;
};
```

- iii. The input size [n, c, h, w] of the entire network is recorded in the public variable `std::vector<TensorInfo> net_input_infos` of `NvtAICustomer`

Its structure is as follows:

```
struct TensorInfo final
{
    std::string name;
    std::vector<int64_t> dims;
};
```

The usage example is as follows:

```
size_t getCustomerTempBufferSize(const NvtAICustomer& customer) {
    .
```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```
.  
for(auto& net_input_info : customer.net_input_infos) {  
    std::string net_input_name = net_input_info.name;  
    std::int64_t n = net_input_info.dims[0];  
    std::int64_t c = net_input_info.dims[1];  
    std::int64_t h = net_input_info.dims[2];  
    std::int64_t w = net_input_info.dims[3];  
    outs()<<"Net input name = "<<net_input_name<<", n = "<<n<<", c = "<<c<<", h = "<<h<<", w  
= "<<w<<std::endl;  
    }  
.  
.  
}
```

6.2.3 Sim-tool

The usage is the same as that of caffe custom layer, please refer to 6.1.3

6.2.4 AI2 SDK

The usage is the same as that of caffe custom layer, please refer to 6.1.4

7 The usage of DSP ACL

NT98530 has DSP hardware support. When DSP is idle, it can actually be used to execute some network layers that originally is operated in the CPU. The powerful DSP can reduce the time-cost of network inference and CPU loading.

In addition, the DSP supports the OP through the ACL interface. This chapter will introduce how to use DSP ACL. Currently, the DSP supports OPs as follows

- Softmax
- Dilated Conv

The following parameters need to be set first to use DSP ACL function.

- Compiler:
 - cmd parameter: the following parameters must be set to enable ACL DSP
 - ◆ --chip **530** : Chip that supports DSP ACL
 - ◆ --use-heteroge **1** : Enable ACL
 - ◆ --backend-order **DSP_CPU** : Using the ACL DSP backend
 - If one of the above parameters is set incorrectly, the execution of the OP that requires DSP ACL will be blocked, similar to the error reported as follows. At this time, please confirm again whether the parameters are set correctly, or the specifications of the op are not currently supported

`[nvt][gen][legal] [check fail]: do not support dilations != {1,1}, op info is as follows:`

- If the chip is not 530 and --backend-order is specified, the tool will return the following warning (the example takes DSP_CPU as an example, the corresponding number is 1). In this case, please set --chip to 530.

`war'--backend-or'er' 1 can only be used with chip 530, s'--backend-or'er' to 0`

- The OP layer can be set to run by DSP in gen config, and the following parameters format are provided

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

- ◆ [tensor_backend/{output_tensor_name}] = {backend_type}

To set a specific backend layer, please fill in the output tensor name of the layer in the {output_tensor_name} field, and the specified backend type in the {backend_type} field. Currently there are three options of backend_type: EXT/DSP/CPU. EXT means that DSP or CPU can be operated. Here is the example:



```
[tensor_backend/Softmax_prob11_Y] = EXT
[tensor_backend/Softmax_prob8_Y] = DSP
```

- If simulation is required, the backend that needs to be set first. In sim config, it is set through parameters [backend/convert_ext_to]. There are currently two options for DSP/CPU. The default is CPU.

```
[backend/convert_ext_to] = DSP
```

If this parameter is not specified, or the wrong parameter is provided, the tool will return the following warning and continue with the default value (CPU).

```
[backend/convert_ext_to] can be only set to DSP, CPU. Current setting is: . Auto fix
[backend/convert_ext_to] = CPU
```

After the setting is completed, the OP set to DSP will be marked with different words in the following places, which can confirm whether DSP ACL is used successfully or not: the word "DLI" will appear at executing log.

```
[nvt][gen][cemit] opId:81 fId:59 [DLI-CONV] Conv_317_Y
```

- The eng_type in process_bit_info.txt will be DSP-DLI

```
golden_layer_name = Conv_317_Y
origal_name = Conv_317_Y
golden_layer_type = None
eng_type = DSP-DLI
```

- The description of layer in profile.txt will have prefix "DLI"

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```
===== opId=81 fid=59 [DLI-CONV] conv_317_Y =====
```

- eng_type in siminfo.txt will be set to 4

```
|-- tensor_name : Conv_317_Y  
...  
|-- eng_type : 4
```

- When executing the simulator, the engine type of the output fixed bin file will be “DSP”

```
[nvt][sim][emu] _kflow_dsp_dump_out_buf() kflow_i - dump buf <dsp>  
...  
[nvt][sim][emu] _kflow_dsp_dump_out_buf() write file: ../DSP_81_OUT0.bin
```

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

8 Network Adjustment Suggestions

8.1 Precision Adjustment

- gen_config adjustment
 - 8/16bit input for selecting different functions
 - [precision/in_hp/conv_en] = 0
 - [precision/in_hp/bnscale_en] = 0
 - [precision/in_hp/deconv_en] = 0
 - [precision/in_hp/fc_en] = 0
 - [precision/in_hp/eltwise_en] = 0It is recommended to turn on the above switch to achieve higher precision
 - When selecting the next layer is relu, whether to unconditionally refer to relu as the fixed-point reference value range of the current layer
 - ### [REF RELU OUTPUT]
 - # 0: current reference its layer out
 - # 1: current reference next relu out
 - [precision/ref_relu_outval_en] = 1When relu 的 negative_slope = 0 , It is recommended to turn on this switch.If it is other value, you need to test to see the effect of this switch
 - Choose whether to enable weight 8bit quantization compression
 - # 0: disable quatization
 - # 1: enable quatization
 - [compression/method/quant_en] = 0Turn on this switch, the weight will be quantized to 8bit, which can save bandwidth, but may also reduce the accuracy. If the weight training period in the caffemodel is 8bit, this switch setting should have no difference. If the training weight is 32bit, you can test whether this switch Influence accuracy
 - Choose whether to enable VLC weight lossless compression
 - # 0: disable variable length coding
 - # 1: enable variable length coding

[compression/method/vlc_en] = 0

Turn on this switch, weight will perform lossless compression, which can save model size and bandwidth under most network structures. If the model is too large and there are memory problems, you can turn on this switch for testing.

- Choose whether to open the mechanism of balancing the value ranges of each layer

[CROSS LAYER EQUALIZATION]

0: turn off CLE

1: turn on CLE

[precision/cle_en] = 0

If conv / bn / scale / relu / leaky relu / prelu / global_ave pool / innerproduct appear continuously and there is no branching, this switch will adjust the weight weight according to the value range and the average maximum value distribution. You can turn on this switch to see the accuracy there has any improvement.



Notice

The intermediate cos similarity of continuous func is not indicative, and it depends on the accuracy at the end.

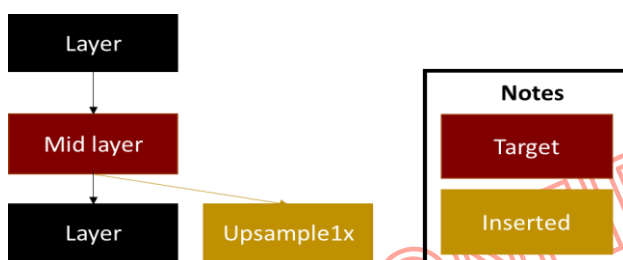
- Restrictions on the full range of scale factor mechanism
 - Tool will open the full range mechanism by default to improve accuracy.
- Selection of reference images (validation dataset)
 - Prepare multiple pictures as a reference for parameter configuration. It is recommended that there be no less than 3 pictures in each category. The pictures selected should cover all training target categories, and the target objects should be clear.

- Prototxt adjustment

- If you want to get the output features of the following layers

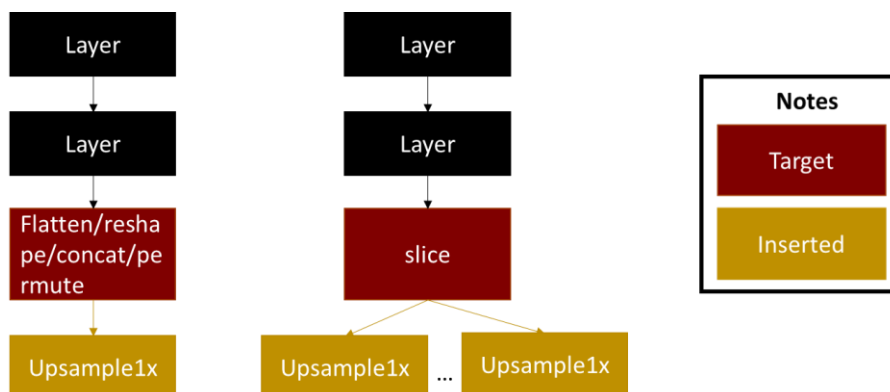
- mid layer

- description: To save memory, unused output buffers of layers are overwritten, we cannot get the output of mid layers.
 - solution: Layers without next layers are regarded as the tail of a network, and their features are reserved. Therefore, inserting upsample1x at the back of the mid layer(without following layers) can reserve its output feature.



- end layer that belongs to flatten/reshape/slice/concat/permute

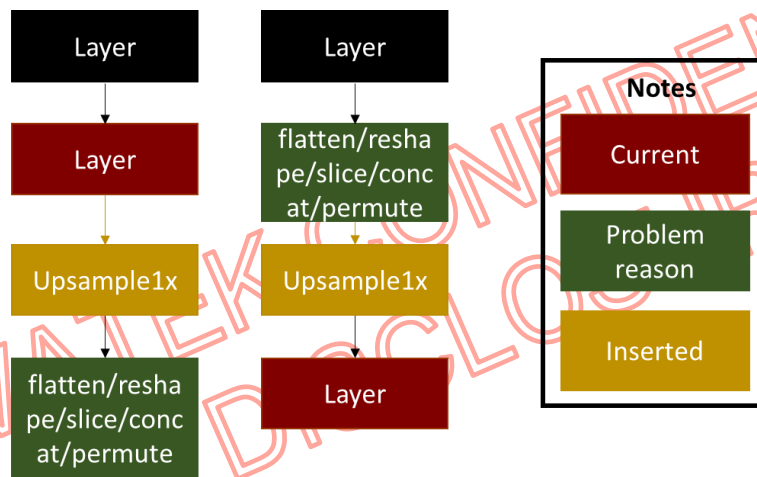
- **description:** Flatten/reshape/slice/concat/permute are removed in the loadable network, user might get output feature of the network with wrong size.
 - **solution:** Insert upsample1x at the end of the network, so that it becomes the tail of a network and its output shape corresponds to that of flatten/reshape/slice/concat/permute.
 - **note:** the abovementioned permute means those which does not change data ordering



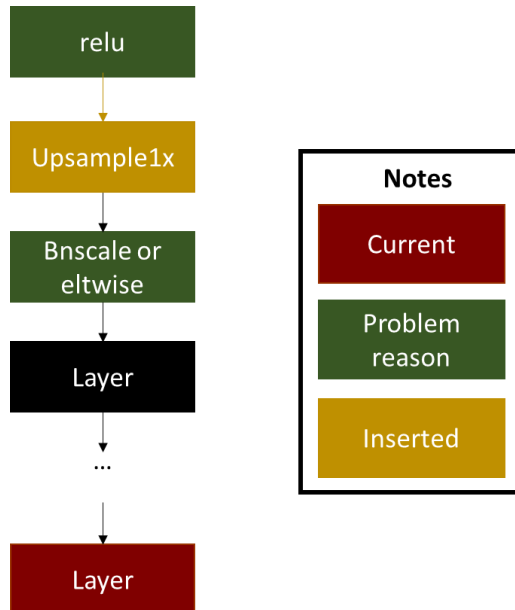
Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

- If current layer is with low precision and
 - previous or next layer is flatten/reshape/slice/concat/permute
 - **possible reason:** Flatten/reshape/slice/concat/permute are removed in the loadable network, and some parameters related to precision need to be modified (e.g. scale factor or bit depth).
 - **solution:** Insert upsample1x between current layer and flatten/reshape/slice/concat/permute. Upsample1x can handle parameters related to precision and the structure of upsample1x with flatten/reshape/slice/concat/permute has been verified in many networks.

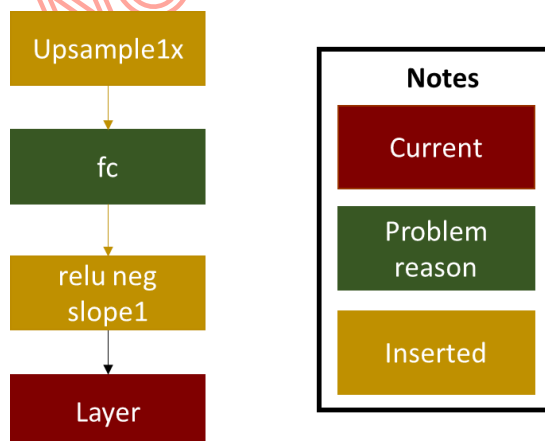


- previous layers include relu+bnscale/relu+eltwise
 - **possible reason:** relu+bnscale or relu+eltwise are fused in loadable network.
 - **solution:** Upsample1x will not be fused with relu/bnscale/eltwise in the generation tool, inserting upsample1x between relu and bnscale/eltwise to break the fusion.



■ previous layer is fc

- **possible reason:** fc layer has following constraints
 - (1) input/output address must be 4-aligned
 - (2) input/output w*h*c must be 4-aligned
- **solution:** Insert upsample1x at the front of fc and relu with negative slope 1(relu1x) at the back of fc. The address/pixel alignment handling of upsample1x and relu1x has been verified in many networks.



■ previous layer has multiple next layers (branches)

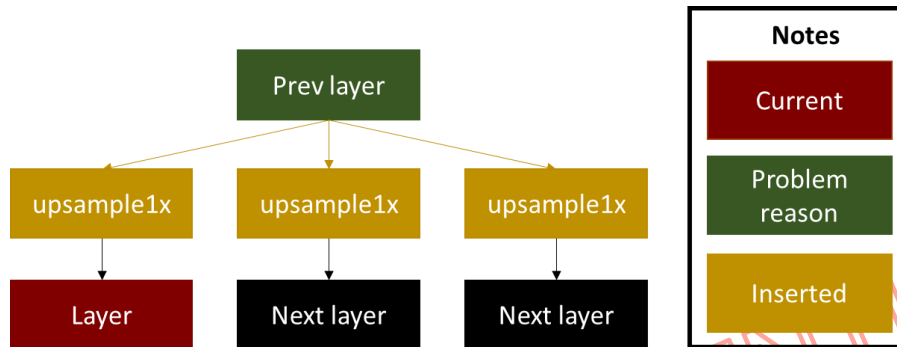
- **possible reason:** If previous layer has multiple next layers, the constraints of all the next layers will be considered and affect the

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

parameter settings. Therefore, precision of some next layers might be affected.

- **solution:** Insert upsample1x between previous layer and each of the next layers. Upsample1x can handle constraints of next layers respectively.



Notice

1. If upsample1x is added, the error of size or dimension inconsistency is encountered in the caffe stage of the conversion tool. The reason is that upsample can only accept 4-dimensional input. At this point you can

- (1) Insert reshape in front of upsample1x to convert the input features into 4 dimensions
- (2) Do not insert upsample1x, but replace it with relu negative slope: 1

2. Please make the above changes on prototxt

8.2 Automatic Network Correction

In the previous chapter, it was mentioned that some situations will need to be modified in prototxt, but in fact the tool already has an automatic correction mechanism for some situations. The following table sorts out the adding method and adding reason.

Add reason	Add method
Support the layer with the same name as bottom	By adding upsample, change the bottom of the same name to another name and then connect it back to the original layer
Support one top to connect multiple concats	This layer top can be directly connected to the first concat layer. When connecting to the second concat layer, you need to add upsample first and then connect to the second concat layer.
1. Support reshape, slice, reverse connect to concat(axis=1) 2. Support not conv, upsample, deconv, relu, tanh, sigmoid, elwise, bn/scale layer connect to concat (axis!=1) 3. Support priorboxconnect to concat (axis=2)	Add upsample after these layers and then connect to the concat layer
1. Support slice connect to reshape, slice, concat 2. Support slice(axis!=1) connect to not conv, upsample, deconv, relu, tanh, sigmoid, elwise, bn/scale layer	Add the upsample layer after the slice layer and then connect these layers
Support two scale layers with different bottom dimensions	Turn the scale layer into the following combination layer :reshape + upsample*n + crop + elwise(mul)
Support axpy layer	Turn axpy layer into the following combination layer: reshape + upsample*n + crop + elwise(mul)

	+ eltwise(add)
Support split layer	The role of the split layer is to copy the bottom to each top, where split is replaced with multiple upsamples
Support setting norm scale in gen_config when the first layer is not conv or upsample	When norm scale is turned on in gen_config and norm scale is not 1, and the first layer is not conv or upsample, add upsample before the first layer
<p>Solve the problem of incorrect size of pooling (HW) in the following situations</p> <p>[1]. When ceil_mode = true , and W_{out} meets</p> $W_{out} = \text{ceil}\left(\frac{(W_{in} + 2 * Pad - Kernel)}{Stride}\right) + 1$ <p>[2]. When W_{out} meets</p> $W_{out} \geq \frac{(W_{in} + Pad)}{Stride} + 1$	<p>Add upsample after the problematic pool</p>
Support eltwise, fc(+relu), sigmoid to adjust accuracy when connected to concat	Add upsample after these layers and then connect to concat
Solve the problem of (HW) FC + Conat because of HW align 4, if (FC / FC / FC)(in parallel) → Concat output to the large buffer is not all multiples of 4, it will output an error problem	If (FC / FC / FC)(in parallel) → Concat, add upsample layer after fc that does not meet batch*channel*height*width align4

8.3 Network Efficiency

1. Optimize UT Rate

- Convolution

- Image mode

Parameters	Condition/Example
In_Width	Condition:
	multiples of 8
	Example:
	In_Width = 64
In_Height	Condition:
	multiples of 16
	Example:
	In_Height = 32
Kernel 11x11 or Kernel 9x9	Condition:
	Out_Width = multiples of 2
	Out_Height = multiples of 2
	$\frac{\text{In_Width} * \text{In_Height}}{\text{Stride}^2 * \text{CEIL}(\text{Kw} * \text{Kh} * \text{In_Channel} / 5)} > 4$
	Example:
	Input (n, c, h, w) = (1, 3, 224, 224) Output (n, c, h, w) = (1, 32, 112, 112) Kernel (w, h) = (11, 11) or (9, 9) Pad (w, h) = (5, 5) or (4, 4) Stride = 2
Kernel 7x7	Condition:
	Out_Width = multiples of 2
	Out_Height = multiples of 4
	If Stride = 4
	$\frac{\text{In_Width} * \text{In_Height}}{\text{Stride}^2 * \text{CEIL}(\text{Kw} * \text{Kh} * \text{In_Channel} / 5)} > 4$
	If Stride != 4
	$\frac{\text{In_Width} * \text{In_Height}}{\text{Stride}^2 * \text{CEIL}(\text{Kw} * \text{Kh} * \text{In_Channel} / 5)} > 8$
	Example: If Stride = 4

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

	Input (n, c, h, w) = (1, 3, 224, 224) Output (n, c, h, w) = (1, 32, 56, 56) Kernel (w, h) = (7, 7) Pad (w, h) = (3, 3) Stride = 4
	Example: If Stride != 4 Input (n, c, h, w) = (1, 3, 224, 224) Output (n, c, h, w) = (1, 32, 112, 112) Kernel (w, h) = (7, 7) Pad (w, h) = (3, 3) Stride = 2
Kernel 3x3	Condition: Out_Width = multiples of 4 Out_Height = multiples of 4 $\frac{\text{In_Width} * \text{In_Height}}{\text{Stride}^2 * \text{CEIL}(\text{Kw} * \text{Kh} * \text{In_Channel} / 5)} > 16$
	Example: Input (n, c, h, w) = (1, 3, 224, 224) Output (n, c, h, w) = (1, 32, 112, 112) Kernel (w, h) = (3, 3) Pad (w, h) = (1, 1) Stride = 2

■ Feature mode

Parameters	Condition/Example
In_Width	Condition:
	multiples of 4
	Example:
	In_Width = 36
In_Height	Condition:
	multiples of 8
	Example:
	In_Height = 32
Kernel 5x5	Condition:
	If In_Channel > 16 (should be a multiple of 16)

	$\frac{\text{In_Width} * \text{In_Height} * \text{In_Batch}}{\text{Stride}^2 * \text{CEIL}(\text{Kw} * \text{Kh} * 16 / 5)} > 1$ <p>In_Channel = 16, it's always efficient In_Channel < 16, it's always inefficient</p>
	Example:
	Input (n, c, h, w) = (1, 128, 32, 32) Output (n, c, h, w) = (1, 64, 16, 16) Kernel (w, h) = (5, 5) Pad (w, h) = (2, 2) Stride = 2
Kernel 3x3	Condition:
	Out_Width = multiples of 2 If In_Channel > 32 (should be a multiple of 32) $\frac{\text{In_Width} * \text{In_Height} * \text{In_Batch}}{\text{Stride}^2 * \text{CEIL}(\text{Kw} * \text{Kh} * 32 / 5)} > 2$ <p>In_Channel = 32, it's always efficient In_Channel < 32, it's always inefficient</p>
	Example:
	Input (n, c, h, w) = (1, 128, 32, 32) Output (n, c, h, w) = (1, 64, 16, 16) Kernel (w, h) = (3, 3) Pad (w, h) = (1, 1) Stride = 2
Kernel 1x1	Condition:
	Out_Width = multiples of 4 Out_Height = multiples of 2 In_Channel > 128 (should be a multiple of 64) If Stride = 2 and $\frac{\text{CEIL}(\text{In_Width}/8) * \text{In_Height} * \text{In_Batch}}{\text{CEIL}(\text{Kw} * \text{Kh} * 64 / 5)} > 4$ If Stride != 2 and $\frac{\text{In_Width} * \text{In_Height} * \text{In_Batch}}{\text{CEIL}(\text{Kw} * \text{Kh} * 64 / 5)} > 8$ <p>In_Channel = 128, it's always efficient In_Channel < 128, it's always inefficient</p>
	Example:
	Input (n, c, h, w) = (1, 128, 32, 32) Output (n, c, h, w) = (1, 64, 32, 32)

	Kernel (w, h) = (1, 1) Pad (w, h) = (0, 0) Stride = 1
Kernel 1x7 (Kw = 1, Kh = 7)	Condition: Out_Width = multiple of 4 In_Channel > 16 (should be a multiple of 16) If Stride = 2 and $\frac{\text{CEIL}(\frac{\text{In_Width}}{8}) * \text{In_Height} * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 16 / 5)} > 1$ If Stride != 2 and $\frac{\text{In_Width} * \text{In_Height} * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 16 / 5)} > 4$ In_Channel = 16, it's always efficient In_Channel < 16, it's always inefficient
	Example: If Stride = 2 Input (n, c, h, w) = (1, 128, 32, 32) Output (n, c, h, w) = (1, 64, 16, 16) Kernel (w, h) = (1, 7) Pad (w, h) = (0, 3) Stride = 2 If Stride != 2 Input (n, c, h, w) = (1, 128, 32, 32) Output (n, c, h, w) = (1, 64, 32, 32) Kernel (w, h) = (1, 7) Pad (w, h) = (0, 3) Stride = 1
	Condition: Out_Width = multiple of 2 In_Channel > 32 (should be a multiple of 32) If Stride = 2 and $\frac{\text{In_Width} * \text{In_Height} * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 32 / 5)} > 4$ If Stride != 2 and $\frac{\text{In_Width} * \text{In_Height} * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 32 / 5)} > 2$ In_Channel = 32, it's always efficient In_Channel < 32, it's always inefficient
Kernel 7x1 (Kw = 7, Kh = 1)	Example:

	<p>If Stride = 2</p> <p>Input (n, c, h, w) = (1, 128, 32, 32)</p> <p>Output (n, c, h, w) = (1, 64, 16, 16)</p> <p>Kernel (w, h) = (7, 1)</p> <p>Pad (w, h) = (3, 0)</p> <p>Stride = 2</p> <p>If Stride != 2</p> <p>Input (n, c, h, w) = (1, 128, 32, 32)</p> <p>Output (n, c, h, w) = (1, 64, 32, 32)</p> <p>Kernel (w, h) = (7, 1)</p> <p>Pad (w, h) = (3, 0)</p> <p>Stride = 1</p>
<p>Kernel 1x5 (Kw = 1, Kh = 5)</p>	<p>Condition:</p> <p>Out_Width = multiple of 4</p> <p>In_Channel > 16 (should be a multiple of 16)</p> <p>If Stride = 2 and $\frac{\text{CEIL}(\frac{\text{In_Width}}{8}) * \text{In_Height} * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 16 / 5)} > 1$</p> <p>If Stride != 2 and $\frac{\text{In_Width} * \text{In_Height} * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 16 / 5)} > 4$</p> <p>In_Channel = 16, it's always efficient</p> <p>In_Channel < 16, it's always inefficient</p>
	<p>Example:</p> <p>If Stride = 2</p> <p>Input (n, c, h, w) = (1, 128, 32, 32)</p> <p>Output (n, c, h, w) = (1, 64, 16, 16)</p> <p>Kernel (w, h) = (1, 5)</p> <p>Pad (w, h) = (0, 2)</p> <p>Stride = 2</p> <p>If Stride != 2</p> <p>Input (n, c, h, w) = (1, 128, 32, 32)</p> <p>Output (n, c, h, w) = (1, 64, 32, 32)</p> <p>Kernel (w, h) = (1, 5)</p> <p>Pad (w, h) = (0, 2)</p> <p>Stride = 1</p>

Kernel 5x1 (Kw = 5, Kh = 1)	Condition: Out_Width = multiple of 2 In_Channel > 32 (should be a multiple of 32) If Stride = 2 and $\frac{\text{In_Width} * \text{In_Height} * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 32 / 5)} > 4$ If Stride != 2 and $\frac{\text{In_Width} * \text{In_Height} * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 32 / 5)} > 2$ In_Channel = 32, it's always efficient In_Channel < 32, it's always inefficient
	Example: If Stride = 2 Input (n, c, h, w) = (1, 128, 32, 32) Output (n, c, h, w) = (1, 64, 16, 16) Kernel (w, h) = (5, 1) Pad (w, h) = (2, 0) Stride = 2 If Stride != 2 Input (n, c, h, w) = (1, 128, 32, 32) Output (n, c, h, w) = (1, 64, 32, 32) Kernel (w, h) = (5, 1) Pad (w, h) = (2, 0) Stride = 1
Kernel 1x3 (Kw = 1, Kh = 3)	Condition: Out_Width = multiple of 4 Out_Height = multiple of 3 In_Channel > 48 (should be a multiple of 16) If Stride = 2 and $\frac{\text{CEIL}(\frac{\text{In_Width}}{8}) * \text{CEIL}(\frac{\text{CEIL}(\frac{\text{In_Height}}{3})}{\text{Stride}}) * \text{In_Batch}}{\text{CEIL}(\text{Kw} * \text{Kh} * 16 / 5)} > 1$ If Stride != 2 and $\frac{\text{In_Width} * \text{CEIL}(\frac{\text{CEIL}(\frac{\text{In_Height}}{3})}{\text{Stride}}) * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 16 / 5)} > 4$ In_Channel = 48, it's always efficient In_Channel < 48, it's always inefficient
	Example: If Stride = 2

	<p>Input (n, c, h, w) = (1, 128, 32, 32)</p> <p>Output (n, c, h, w) = (1, 64, 16, 16)</p> <p>Kernel (w, h) = (1, 3)</p> <p>Pad (w, h) = (0, 1)</p> <p>Stride = 2</p> <p>If Stride != 2</p> <p>Input (n, c, h, w) = (1, 128, 32, 32)</p> <p>Output (n, c, h, w) = (1, 64, 32, 32)</p> <p>Kernel (w, h) = (1, 3)</p> <p>Pad (w, h) = (0, 1)</p> <p>Stride = 1</p>
<p>Kernel 3x1 (Kw = 3, Kh = 1)</p>	<p>Condition:</p>
	<p>Out_Width = multiple of 2</p> <p>Out_Height = multiple of 3</p> <p>In_Channel > 64 (should be a multiple of 32)</p> <p>If Stride = 2 and $\frac{\text{In_Width} * \text{CEIL}(\frac{\text{In_Height}}{6}) * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 32 / 5)} > 2$</p> <p>If Stride != 2 and $\frac{\text{In_Width} * \text{CEIL}(\frac{\text{In_Height}}{3}) * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 32 / 5)} > 2$</p> <p>In_Channel = 64, it's always efficient</p> <p>In_Channel < 64, it's always inefficient</p>
	<p>Example:</p> <p>If Stride = 2</p> <p>Input (n, c, h, w) = (1, 128, 32, 32)</p> <p>Output (n, c, h, w) = (1, 64, 16, 16)</p> <p>Kernel (w, h) = (3, 1)</p> <p>Pad (w, h) = (1, 0)</p> <p>Stride = 2</p> <p>If Stride != 2</p> <p>Input (n, c, h, w) = (1, 128, 32, 32)</p> <p>Output (n, c, h, w) = (1, 64, 32, 32)</p> <p>Kernel (w, h) = (3, 1)</p> <p>Pad (w, h) = (1, 0)</p> <p>Stride = 1</p>

- Fully Connection

Parameters	Condition/Example
Feature dimension	Condition:
	multiples of 32
	Example:
	Feature dimension = 64

- Matmul (A (1x1xNxK) * B (1x1xKxM))

Parameters	Condition/Example
Feature dimension	Condition:
	M: multiples of 4
	K: multiples of 64
	Example:
	M: multiples of 64 K: multiples of 128

2. Data capture efficiency

- DRAM starting address
 - 4-word aligned

8.4 Quantized Param. importment

For the features and weights of each layer, if you want to adjust the value range and reassign the bit settings, you can adjust the following config settings related to quantization injection

1. Add the following settings to gen_config

[1] Set the quantized param

```
#[FEATURE PRECISION]
# 0: normal gentool process which will not import the quan_info_XXXXX.json
# 1: quantization info (quan_info_XXXXX.json) is written to [path/cust_quan/path]
# 2: quantization info (quan_info_XXXXX.json) is read from [path/cust_quan/path] which will
read the ref_image
# 3: quantization info (quan_info_XXXXX.json) is read from [path/cust_quan/path] which will
not read the ref_image
[precision/cust_quan/mode] = 1
```

[2] Set the path of json file of quantized parameters (Support relative to the path set -- config-dir, or absolute path)

```
## weight_quantization
[path/cust_quan/path] = ..\nvtai_tool\config\00001_inception_v2 \cnn25
```

[3] Set the quantization mode

```
### [CUSTOMIZED QUANTIZATION]
## feature & weight val range format
# 0: max+min
# 1: zeropoint+scale
# 2: bit info
[precision/cust_quan/val_mode] = 0
```



Notice

The quantization mode is used to control the expression of the value range in the interface file. Currently, the zeropoint+scale is under development and will be supported in the future.

[4] Generate model and quantization parameter json file

Obtain quantization information quan_info_maxmin.json. Set [precision/cust_

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

quan/mode] to 1, and execute gentool, the initial quantization information quan_info_maxmin.json (when [precision/cust_quan/val_mode] = 0) or quan_info_bitinfo.json (when [precision/cust_quan/val_mode] = 2) will be generated under the quantization injection path ([path/cust_quan/path]) specified in gen_config



Notice

When [precision/cust_quan/mode] is set to 1, only the quan_info_maxmin json file will be generated, and nvt_model.bin will not be generated

2. Introduction to output files

[1] Quantization information (quan_info_maxmin.json):

Write the quantization information of the features and weights of each layer in the model. The file will print the information with the output tensor_name of each layer as the name. The printing content includes: the onnx op name of each layer, the type of each layer (layer_type), the maximum and minimum values of the output features of each layer (output0_max/ output0_min). If the layer is convolution or innerproduct, the layer will be printed. The maximum and minimum weights (input1_max/ input1_min/ input2_max/ input2_min), where input1 represents Weight and input2 represents Bias

```
{
  "Convolution_conv_Y": {
    "layer_type": "C"nv",
    "input1_max": 2.9798028469085693,
    "input1_min": -2.95339298248291,
    "input2_max": 4.032540914522542e-07,
    "input2_min": -3.6120823665442003e-07,
    "output0_max": 3528.323974609375,
    "output0_min": -3416.013671875
  },
  "BatchNorm_Scale_bn_1_scale_Y": {
    "layer_type": "BatchNormalizat"on",
    "output0_max": 13.681736946105957,
    "output0_min": -8.931833267211914
  },
  "ReLU_relu_Y": {
    "layer_type": "R"lu",
    "output0_max": 13.681736946105957,
    "output0_min": 0.0
  },
  "Pooling_pool_Y": {
    "layer_type": "MaxP"ol",
    "output0_max": 13.681736946105957,

```

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

```

    "output0_in": 0.0
  },
  "InnerProduct_fc1_Gem"_Y": {
    "layer_type": "Gemm",
    "input1_ax": 1.7419883012771606,
    "input1_in": -0.6007164120674133,
    "input2_ax": 0.9874842762947083,
    "input2_in": -0.8291921019554138,
    "output0_ax": 12.809922218322754,
    "output0_in": -7.271744728088379
  }
}

```



Notice

At present, the quantization injection of weight only supports the conv and fc layers (the gemm layer in the onnx model)

[2] Quantization information (quan_info_bitinfo.json):

Dump the bit information of the features and weights of each layer in the model. When "conv1" is the network layer, a "layer_type" is the type of the network layer, which is aligned with that of ONNX. "weight_bitdepth", "weight_sign_bit", "weight_frac_bit" are the bit depth, sign bit, quantized floating-point bit of this layer of weight, respectively. "bias_bitdepth", "bias_sign_bit", "bias_frac_bit" are the bit depth of this layer of bias, sign bit, quantized floating point bit. "output0_bitdepth", "output0_sign_bit", "output0_frac_bit" are the bit depth, sign bit, and quantized floating point bit of the first output of this layer, respectively.

```

{
  "conv1": {
    "layer_type": "Conv",
    "weight_bitdepth": 8,
    "weight_sign_bit": 1,
    "weight_frac_bit": 16,
    "bias_bitdepth": 12,
    "bias_sign_bit": 1,
    "bias_frac_bit": 12,
    "output0_bitdepth": 8,
    "output0_sign_bit": 1,
    "output0_frac_bit": 3
  },
}

```

3. Modify the quantitative information and import it into the original network model

[1] Modify quan_info_maxmin.json:

Modify the max and min values of features and weights of each layer according to the needs

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.



Notice

If the layer is near the concat layer, gentool will automatically adjust the reference layer to the concat output, so adjusting the output of these layers does not affect the accuracy adjustment

- [2] Turn on quantization injection, and execute gentool to set [precision/cust_quan/mode] to 2, and execute gentool. In addition, whether to switch the full value range can be set with the following parameters:

Output value reference mode (non-full range or full range): Non-full range means that scale factor of the quantization is 1.0. Full range means that max value of outputs is used to compute the scale factor.

0: Enable non-full range

1: Enable full range (default)

[precision/ref_outval_en] = 0

4. Output the maximum and minimum values of each layer

- [1] If you do not use quantization injection ([precision/cust_quan/mode = 0]), or after quantization injection ([precision/cust_quan/mode = 2]), you can refer to the following parameter settings when you want to know the maximum and minimum output values of each layer:

Dump tensor information. If this option is enabled, the maximum and minimum values of each fusion layer will be dumped to layer_maxmin.json, and stored in nvtai_tool/output/{MODEL}/gentool/, which is disabled by default

0: disable dump tensor max min

1: enable dump tensor max min

[precision/cust_quan/dump_tensor_en] = 0

9 Tool Debugging

9.1 LOG Printing Specification

When the tool is running, it will print out the current operating status. The specifications for printing are listed below. You can use this table to know which module has the problem by the printed log. It can reduce the scope of troubleshooting.

9.1.1 Compiler

Name	Method	Class	Description
nvt	gen	cfg	Configuration and analysis
nvt	gen	parser	ONNC IR parsing and conversion
nvt	gen	legal	ONNC IR operation converted to nvt IR and graph optimization
nvt	gen	cali	ONNC float inference + quantization parameter optimization
nvt	gen	binfo	Bit information processing
nvt	gen	wpart	OP converts work domain for scheduling
nvt	gen	schd	OP scheduling and optimization
nvt	gen	malloc	Tensor memory configuration and optimization
nvt	gen	lib-weight	Target hardware weight parameter generation
nvt	gen	lib-stripe	Target hardware stripe parameter generation
nvt	gen	lib-lcmd	Target hardware link-list command parameter generation
nvt	gen	lib-perf	Target hardware performance bandwidth and time estimation
nvt	gen	cemit	Loadable archive generation

9.1.2 Simulator

Name	Method	Class	Description
nvt	sim	cfg	Configuration and analysis
nvt	sim	target	sim info parsing and emu preparation
nvt	sim	lib-nn	Target hardware operation simulation
nvt	sim	lib-perf	Target hardware performance bandwidth and time estimation
nvt	sim	emu	Emu runs and produces target fixed inference results
nvt	sim	golden	ONNC fp32 runs and produces golden float inference results
nvt	sim	diff	Target fixed/golden float results generation & diff report generation

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

10 Revision History

Revision	Date	Author	Changes
0.0.1	2021/06/29	Edward Tseng	First formal version
0.0.2	2021/07/05	Edward Tseng	Modify the path of folder struc. By new version of tool.
0.0.3	2021/07/21	Edward Tseng	6.1 add statement for customlayer in caffe
0.0.4	2021/07/23	Edward Tseng	3.1 modify the gen_config setting
0.0.5	2021/08/02	Edward Tseng	3.1 modify the gen_config setting
0.0.6	2021/08/03	Edward Tseng	6.1.2, 6.1.3 add the method of getting info. for custom layer 5 add statement of config. parameters in cmd line
0.0.7	2021/08/18	Edward Tseng	5, 6.1.1, 6.1.3 modify the path of scripts folder
0.0.8	2021/08/24	Edward Tseng	5.2, 5.3 a-- --config description
0.0.9	2021/08/31	Edward Tseng	3.1 add [preproc/meansub_hp/en] description 6.1 modify the path of adding caffe_nvt.proto
0.1.0	2021/10/25	Edward Tseng	6.1.3 modify usage of the make cmd
1.0.0	2021/11/15	Edward Tseng	2.4 Modify the info of supporting layer 2.4.1 Add the supporting spec. table of tool 6.1.2 Add the config for turning on/off openMP 6.1.3 Add the config for turning on/off openMP
1.0.1	2022/01/05	Edward Tseng	2.4.1 Modify the stride condition of pooling
1.0.2	2022/01/13	Edward Tseng	2.1 Modify the description of supporting framework
1.0.3	2022/01/13	Edward Tseng	2.4, 2.4.1 Add clip layer and threshold layer 5.2, 5.3 remove openmp-en 8 Add the debug log description
1.0.4	2022/02/21	Alex	7.4 Add the description of inserting quan. Info.
1.0.5	2022/02/24	Edward Tseng	2.4 Add support layer for 331 5.2, 5.3 Add 331 chip setting 3.1 Add the setting of L2 clip threshold
1.0.6	2022/03/31	Edward Tseng Jack Wei	2.1 Add the supported layer of onnx 2.2 Add the test Pytorch network

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or usefulness of any such information.

			3.1 Add the out_hp setting of eltwise Modify the description of dynamic batch 5.2 Add the description of converting Pytorch model to ONNX model
1.0.7	2022/04/25	Edward Tseng	3.1 Add [precision/avgpool_noclamp_en] 5.2.1 Add supported onnx version description related to opset12
1.0.8	2022/05/17	Edward Tseng	2.1 ~ 2.4 Add the info of NT98530 5.3 Add the new cmd config. 7.4 Add the new gen_config for quantization.
1.0.9	2022/06/30	Edward Tseng	2.2 Add supported network list of tensorflow & Pytorch 2.4 Add correlation layer sepc. 3.1 Add more gen-config for precision 4.1 Add more sim-config for JPEG-in 5.3 ~ 5.4 Add more cmd-config 7.4 Add the bit_info mode
1.0.10	2022/09/05	Edward Tseng	2.1, 2.2 Add more nets and supported ops 3.1 Add more configs 7 Add DSP ACL of NT98530
1.0.11	2022/12/02	Edward Tseng	2.1, 2.2 Add more nets and supported ops 3.1 Add more configs 5.3 Add NT98331 64bit setting 6.2 Add ONNX custom layer flow
1.0.12	2023/02/05	Lisa Huang	8.3 Modify conv and matmul spec with good hardware efficiency 2.4.2 Move the detail to CNN25_HW_SupportSpec.pdf
1.0.13	2023/03/08	Edward Tseng	2.4.1 Move the detail of tool limitation to Novaic_SupportSpec_Caffe & Novaic_SupportSpec_ONNX