



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 介紹	4
1.1 簡介	4
1.2 環境需求	5
1.3 安裝介紹	7
1.3.1 Docker 安裝說明	7
1.3.2 第一次使用 Docker Image	8
1.3.3 工具包安裝說明	8
2 支持項目介紹	10
2.1 模型框架支持列表	10
2.2 驗證網絡列表	10
2.3 層融合組合列表	14
2.4 網絡層支持列表	15
2.4.1 工具網絡層支持規格限制	19
2.4.2 硬件網絡層支持規格限制	19
3 Gen-tool 介紹	20
3.1 Config 設置	20
3.2 輸入資料	38
3.3 輸出資料	40
4 Sim-tool 介紹	41
4.1 Config 設置	41
4.2 輸入資料	46
4.3 輸出資料	47
5 工具運行樣例說明	50
5.1 檔案預處理	50
5.2 ONNX 模型轉換	51
5.2.1 Pytorch to ONNX	52

5.2.2	ONNX to ONNX	53
5.3	運行 Gen-tool 程序.....	54
5.4	運行 Sim-tool 程序	57
6	添加私有層方式.....	59
6.1	建置私有層的流程 (Caffe).....	59
6.1.1	Caffe.....	59
6.1.2	Gen-tool	59
6.1.3	Sim-tool	66
6.1.4	AI2 SDK	72
6.2	建置私有層的流程 (ONNX)	74
6.2.1	ONNX.....	74
6.2.2	Gen-tool	77
6.2.3	Sim-tool	81
6.2.4	AI2 SDK	81
7	DSP ACL 使用方式.....	82
8	網路調整建議	84
8.1	精度調整	84
8.2	自動化網路修正	89
8.3	網路效率.....	91
8.4	量化注入.....	99
9	工具除錯	104
9.1	LOG 打印規範.....	104
9.1.1	Compiler.....	104
9.1.2	Simulator	105
10	Revision History.....	106

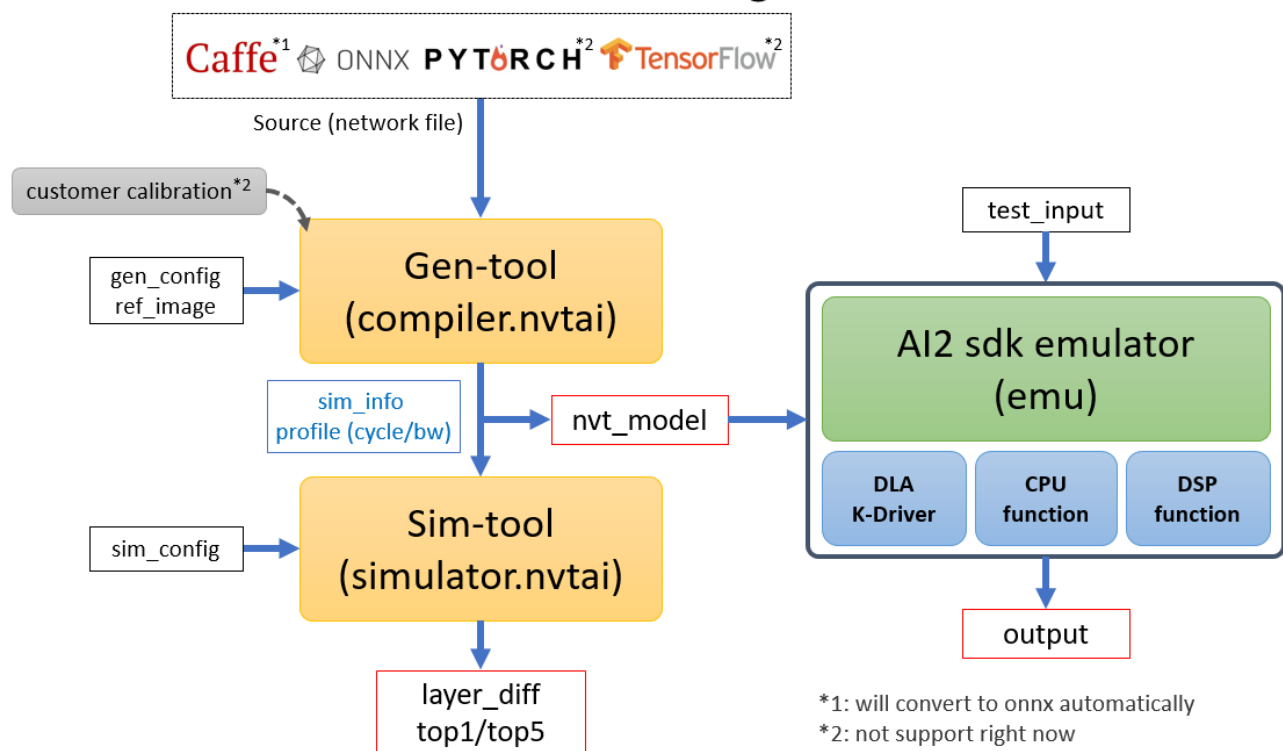
1 介紹

1.1 簡介

Novaic Convertor 採用新的 Compiler 架構，容納更多 Framework 將 Caffe/Tensorflow/Pytorch/ONNX 訓練完成的 Model 轉換成二進制檔案 (nvt_model.bin)，以供版端 32bit/64bit 環境下的深度學習網絡做智能檢測。此版本採用 C/C++ 語言開發，對於平台的移植性有大幅的提升。

Novaic Convertor 目前包含兩個核心模塊

1. Gen-tool: 將 Caffe/Tensorflow/Pytorch/ONNX 訓練完的模型轉換成二進制的 Loadable File (nvt_model.bin)。
2. Sim-tool: 準備測試資料以及輸入 Gen-tool 產生之硬體參數檔 Loadable File (e.g. nvt_model.bin)，仿真其效果，再與 Runtime 答案作比較，看產出是否正確。若是不正確其提供的 Layer_diff 也可做為調整參數之依據。

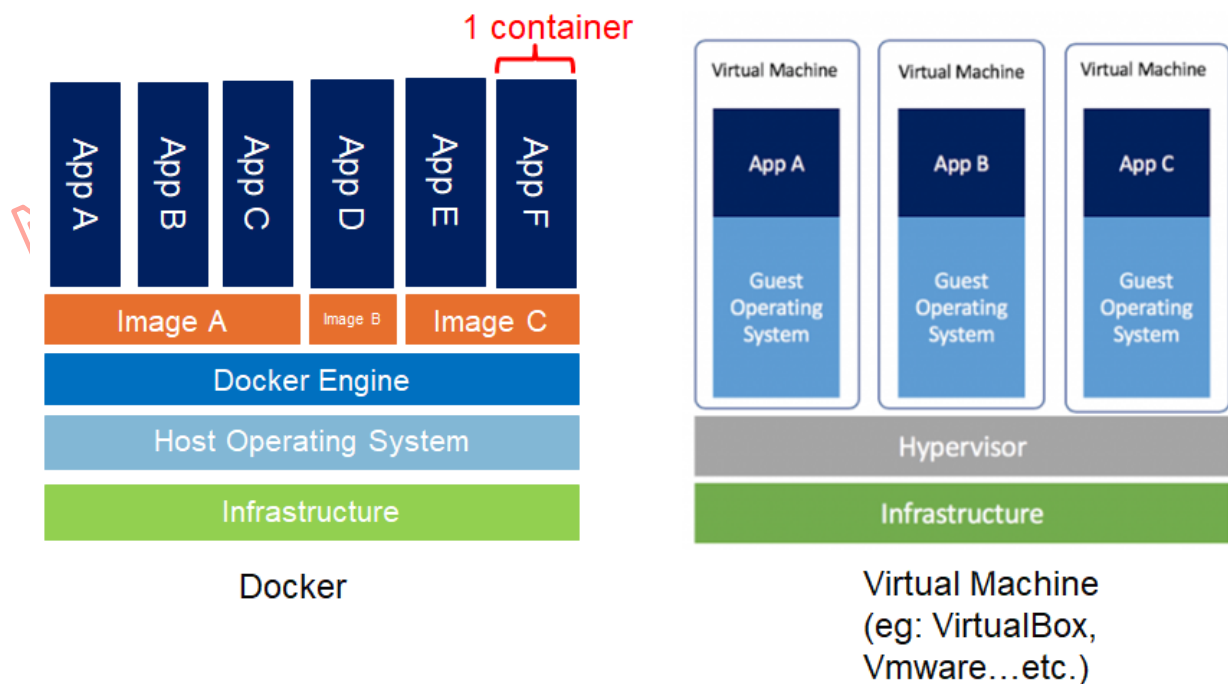


1.2 環境需求

- Software:

項目	版本描述	備註
Tool 版本號	v01.07.2211296	
Docker 版本號	v00.07.2108260	Support ubuntu16.04

由於避免 OS 環境對 Tool 編譯時帶來環境變數或是安裝套件不足的困擾，所以引進在自帶套件的 Docker 中操作的方式讓使用環境統一，Docker 是一個”應用程式”，假如您手邊有個 Docker image(已打包好的環境)，只要有安裝 Docker 的 OS 下，且 Docker Engine 運作正常的情況，就可以像執行應用程式那樣執行此 Docker image 所建構的環境，在此環境下編譯與驗證程式



● Server

項目	版本描述	備註
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:

項目	需求內容	備註
PC 硬件需求	Memory RAM 至少需要 4G 以上	
支持的 IC 型號	<ul style="list-style-type: none"> ● NT98336 (64bit) ● NT9852x (32bit) ● NT9832x (32bit) ● NT9856x (32bit) ● NT98331 (32bit/64bit) ● NT98530 (64bit) 	

1.3 安裝介紹

1.3.1 Docker 安裝說明

以下介紹 Ubuntu 環境下的安裝方式，若環境為其他 OS，請參考 Docker 官方網站 (<https://docs.docker.com/engine/install/>) 內的步驟安裝。

1. 先卸除舊的 Docker 版本(舊版本的名稱為 docker, docker.io 或 docker-engine)
`[host]$ sudo apt-get remove docker docker-engine docker.io containerd runc`
2. 先安裝會使用到的 package
`[host]$ sudo apt-get install apt-transport-https ca-certificates curl gnupg lsb-release`
3. 新增 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 $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null`
4. 安裝 Docker
`[host]$ sudo apt-get install docker-ce docker-ce-cli containerd.io`
5. 確認 Docker Engine 是否運作
`[host]$ systemctl status docker`
6. 若 Docker Engine 沒有運作，執行下列指令
`[host]$ systemctl start docker.service`
7. 驗證 Docker 是否被正確的安裝
`[host]$ sudo docker run hello-world`
此指令會從 DockerHub 下載一個測試 image 並跑起一個 container，當 container 跑起來後，會打印出一些訊息後再離開

1.3.2 第一次使用 Docker Image

1. 載入 Docker image

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

2. 確認目前已載入的 Docker image

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

3. 使用 Docker image ubuntu16p04:v00.07.2108260 創建一個新的 container，並把 < host_shared_folder_path > 與 <docker_shared_folder_path> 視為視同個資料夾，之後可將工具的安裝包透過 < host_shared_folder_path > 放到 docker 中

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

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

```
ubuntu16p04:v00.07.2108260
```

4. 確認目前的系統上的 container

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

5. 離開目前的 container

```
[docker]# exit
```

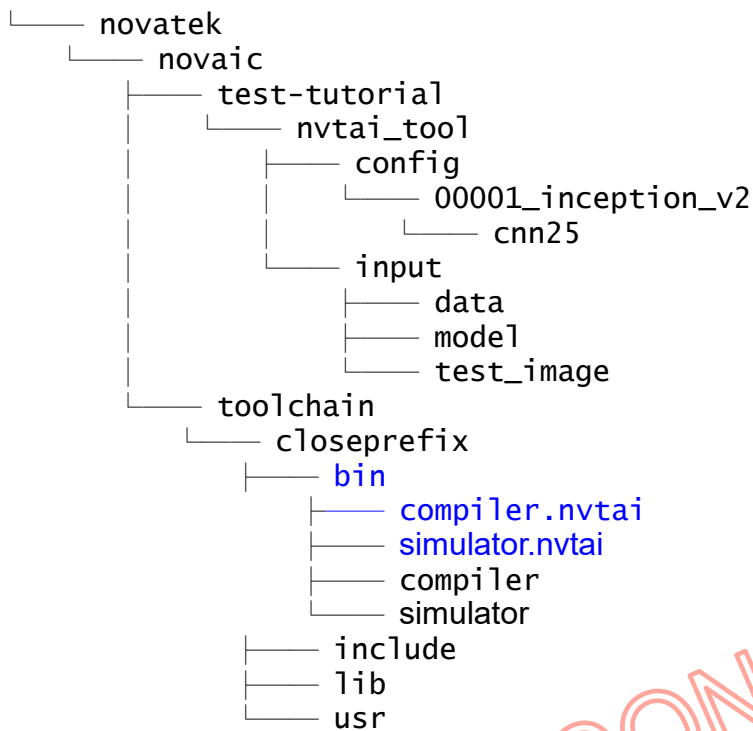
1.3.3 工具包安裝說明

1. 將工具的安裝包透過 < host_shared_folder_path > 放到 docker

2. 解壓工具包

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

3. 解壓後主要目錄結構大致如下圖所示，其中主要程式 compiler.nvtai、simulator.nvtai 會放在以下位置 (closeprefix/bin/)，內建的 example 會放在 test-tutorial 中



注意事項

若有移動 `compiler.nvtai`, `simulator.nvtai` 等執行檔的需求，`compiler` 與 `simulator` 兩個目錄需跟著移動，維持在同一層目錄下

2 支持項目介紹

2.1 模型框架支持列表

以下介紹工具目前能支持的框架以及支持狀況：

框架名稱	支持狀況
Caffe	已開放支持，工具內建自動轉換成 ONNX 格式
ONNX	已部分支持，可直接支持 ONNX 格式
Pytorch	已部分支持，需預先將 Pytorch 轉成 ONNX 格式
Tensorflow	已部分支持，需預先將 Tensorflow 轉成 ONNX 格式

2.2 驗證網絡列表

目前測試通過的 Caffe 網絡架構已超過 200 支，摘錄其中 18 支網絡資訊如下表

網絡名稱	測試時輸入尺寸
Inceptionv2	224x224x3
Inceptionv3	299x299x3
Inceptionv4	299x299x3
Mobilenetv1	224x224x3
Mobilenetssdv1	300x300x3
Vgg16	224x224x3
Resnet18	112x112x3
Resnet50	224x224x3
Resnet101	224x224x3
Yolov2	960x540x3
Yolov3	416x416x3
Yolov3_mobilenet_lite	512x512x3
Squeezenet_v1.1	227x227x3
Pfld	112x112x3

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.

Enet	256x128x3
PoseNet	144x192x3
RetinaFaceNet	960x540x3
SphereFace	96x112x3

目前測試過的 Pytorch 模型資訊如下表

網絡名稱	下載網址
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/bubbliiiiing/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
pflD	https://github.com/polarisZhao/PFLD-pytorch
lanenet	Private network
res14_clean1029_col or	Private network

denoiser	Private network
unet	https://github.com/bubbliiiiing/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	
上述模型有做過拿掉後處理或是將前處理合到 NUE2 HW，做完後再進網路的調整	

目前測試過的 Tensorflow 模型資訊如下表

網絡名稱	下載網址
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_yolov3/yolov3-lite	PINTO_model_zoo/download_saved_model.sh at main · PINTO0309/PINTO_model_zoo (github.com)
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

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	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	
上述模型有做過拿掉後處理或是將前處理合到 NUE2 HW，做完後再進網路的調整	

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

2.3 層融合組合列表

在硬體設計上，為了節省頻寬及加快速度，針對網絡中常見的架構有以下連接組合方式：

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 網絡層支持列表

以下介紹工具與硬件所支援的 Caffe 框架功能列表以及限制規格:

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	✓	✓	✓	✓	✓	✓	✓
Matrix Multiplication	Matrix Multiplication	✓	✓	✓	✓	✓	✓	✓
Utility	Reshape	✓	✓	✓	✓	✓	✓	✓
	Crop	✓	✓	✓	✓	✓	✓	✓
	Clip	✓	✓	✓	✓	✓	✓	✓
	Threshold	✓	✓	✓	✓	✓	✓	✓
	Dummy (only support dummy → crop struc)	✓	✓	✓	✓	✓	✓	✓
	Flatten	✓	✓	✓	✓	✓	✓	✓
	Permute	✓	✓	✓	✓	✓	✓	✓
	Reorganize (stride=2)	✓	✓	✓	✓	✓	✓	✓
	Reverse (CPU Layer)	✓	✓	✓	✓	✓	✓	✓
	Passthrough (stride=2)	✓	✓	✓	✓	✓	✓	✓
	Concat	✓	✓	✓	✓	✓	✓	✓
	Slice	✓	✓	✓	✓	✓	✓	✓

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.

	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.

以下介紹工具所支援的 Onnx 框架功能列表:

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

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.

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)


注意事項

ROI-pooling 主路特徵分為單 batch 和多 batch 用法:

- 單 batch: ROI 可多個，但只可靜態設置 (prototxt 內指明 ROI 數量)

多 batch: 不支援 ROI [batch idx, x1, y1, x2, y2] 的 batch idx 任意指定，輸入特徵 batch 與 ROI 為一對一關係 (一個 batch 對應一個 ROI)，因此輸入特徵 batch 數 = ROI 數量 = 輸出特徵 batch 數。

以下 Layer 放最後一層時須特別注意

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

這些層在最後一層的問題是，如果要拿最後一層的資訊（特別是維度資訊），他實際上拿到的會是上一層的資訊，這樣會造成誤解（只是會造成誤解，不是說不可以做輸出層）。

2.4.1 工具網絡層支持規格限制

以下表格介紹深度學習網絡搭配工具時所支援的功能的細部限制，詳細的層支持列表可參考 Novaic_SupportSpec_Caffe_vxx.xx_en 與 Novaic_SupportSpec_ONNX_vxx.xx_en 兩份文檔

2.4.2 硬件網絡層支持規格限制

各代 IC HW 限制可參考 Novaic_SupportSpec_CNN25_HW_vxx.xx_en 文檔，可根據該文檔規劃網絡參數以符合硬件有效運行。

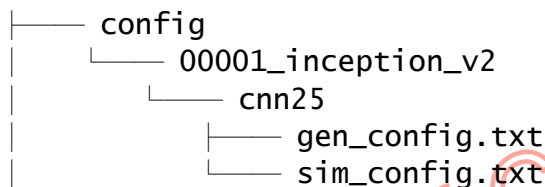
NOVATEK CONFIDENTIAL
NO DISCLOSURE!

3 Gen-tool 介紹

Gen-tool 是用來將 ONNX，Caffe，Pytorch，Tensorflow 等架構的模型轉換，最佳化後產生版端硬體可以解讀的二進制參數模型檔。

3.1 Config 設置

Config 檔案默認是放置在 `nvtai_tool` 資料夾下，以 `00001_inception_v2` 為例，其資料結構的設置如下面所示



- 對於單輸入，開啟 `config\00001_inception_v2\cnn25\gen_config.txt` 後做以下設置

[1] 設置模型與標籤路徑: (支持相對於 `--config-dir` 設置的路徑之下，或是絕對路徑)

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

```
## model
```

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

[2] 設置 mean 資料路徑: (支持相對於 `--config-dir` 設置的路徑之下，或是絕對路徑)

```
## mean
```

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

[3] 設置參考資料路徑: (支持相對於 `--config-dir` 設置的路徑之下，或是絕對路徑)

```
## img
```

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

```
## list
```

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

[4] 設置 Gen-tool 執行完的輸出資料夾: (支持相對於 `--config-dir` 設置的路徑之下，或是絕對路徑)

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

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.

```
## gen output root
```

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



注意事項

與 `sim_config` 設置時不同，在 `gen_config` 設置輸出路徑時，記得不可以加上網路名稱，例如 `..\nvtai_tool\output\00001_inception_v2`

[5] 參考影像數量 [1, 1000]:

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

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



注意事項

由於 `ref_data` 使用的類別與數量有時會影響量化的結果，若在仿真或測試時覺得 `cos` or `avg.diff` 與標準有點落差時可以試著調整 `ref_img_list` 或是 `[ref_data/num]`

[6] 設置是否要使用多尺度輸入模式:

```
## 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] 當多尺度設置 `[multiscale/en]` 開啟時，設置每個尺度的寬高資訊:

```
## 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
```



注意事項

1. `[multiscale/width]` 設定的個數和 `[multiscale/height]` 需要一致

2. [multiscale/width] 和 [multiscale/height] 設定的個數限制在 [1, 64] 區間
3. 在 530 上，限制 [multiscale/width] 和 [multiscale/height] 的範圍為 [1, 4096]
4. 在非 530 上，限制 [multiscale/width] 和 [multiscale/height] 需要大於等於 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

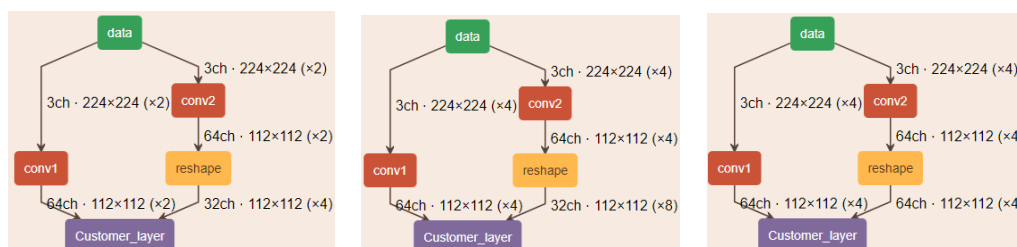
[dynamic_batch/en] = 1

[dynamic_batch/batch_size] = 2, 1



注意事項

1. 在 SDK 需使用的所有 batch_size 皆須設置；ex 若需使用 batch = 8, 4, 1，則 gen config 設置必須包含 batch_size = 8, 4, 1，若 gen config 沒設置 batch_size = 6，則 SDK 也無法使用 batch_size = 6。
2. [dynamic_batch/batch_size] 設置必須採數值遞減方式且數值不重複，以下為一個符合規範的設定：[dynamic_batch/batch_size] = 16, 10, 8, 2, 1。
3. ref_list.txt 設置以最大 batch 為主；例如設置的最大 batch_size = 16，則 ref_list.txt 每一行需設置滿 16 張 ref_image 檔名。
4. 所有 batch_size 組合必須在 gentool 階段可預先知道且可靜態描述
5. 當使用私有層或是 Reshape 層，且網路中某些層有多個輸入或是多個輸出時，需特別注意多個輸入/輸出的 batch size 變化必須成比例增加或減少。例如下圖的範例，Customer_layer 私有層在網路端輸入 batch=2 時，該私有層的兩個輸入 shape 分別是 [2, c1, h1, w1] 與 [4, c2, h2, w2] (如下圖左)；當網路端輸入 batch=4 時，該私有層的兩個輸入 shape 改變為 [4, c1, h1, w1] 與 [8, c2, h2, w2] (如下圖中，batch 都變為原始的 2 倍)，但不允許 batch 變化為 [4, c1, h1, w1] 與 [4, c2, h2, w2] (如下圖右，batch 變化分別為 2 倍與 1 倍)。

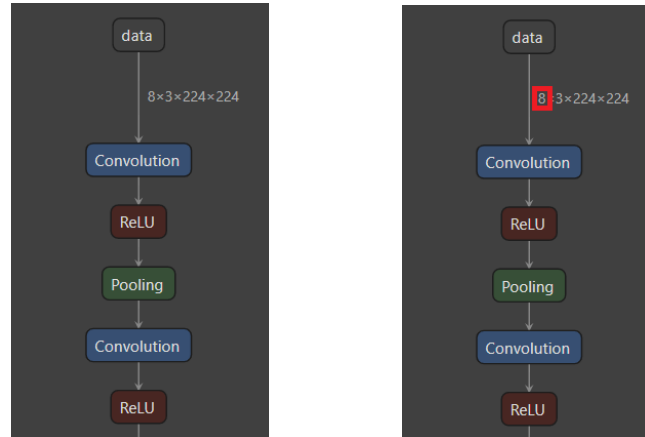


6. 當 [dynamic_batch/en] = 0 時，為原始多 batch 的設置方法：batch 數由網路輸入 shape 決定。如下左圖中的“data”為 batch=8，對於後續每一層的 batch size 就會依照輸入 shape

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.

以及該層的運算行為得到；當 `[dynamic_batch/en] = 1` 時，原本的 `batch` 數設置會由 `[dynamic_batch/batch_size]` 的設置所改寫變成無效，例如下圖右紅色框的 `batch=8` 不會起作用，而會被 `config` 內指定的 `[dynamic_batch/batch_size]` 覆寫作為網路輸入的 `batch size`。



[9] 前處理硬體層(NUE2)輸入格式:

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] 設置當 **[preproc/in/fmt] = 3** 時 FMT_FEAT 輸入型態:

input type for FMT_FEAT

0: TYPE_INT8

1: TYPE_UINT8

2: TYPE_INT16

3: TYPE_UINT16

6: TYPE_FLOAT32

[preproc/in/type] = 1

[11] 設置當 **[preproc/in/fmt] = 3** 時 FMT_FEAT(fixedpoint 表示法)輸入 frac bit:

input frac bit num for FMT_FEAT

[preproc/in/frac_bit_num] = 0

[12] 前處理硬體層(NUE2)輸入大小:

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] 參考影像 resize 的模式:

0: RESIZE_WH (resize 到[preproc/resize/width]和[preproc/resize/height])

1: x

2: x

3: RESIZE_WH_WITH_TILING

(維持長寬比重複拼貼至[preproc/resize/width]和[preproc/resize/height]大小)

[preproc/resize/mode] = 0

[14] Resize 的輸出大小:

RESIZE_WH PARAMETERS

width & height [1, 1023]

[preproc/resize/width] = 224

[preproc/resize/height] = 224

[15] 是否要執行 mean subtraction:

[MEAN SUB]

0: disable meansub

1: enable meansub

[preproc/meansub/en] = 1

[16] 開啟 16-bit (高精度) 的 meansub 輸出:

0: disable meansub

1: enable meansub

[preproc/meansub_hp/en] = 0

- [17] Mean subtraction 的執行模式 (DC or Planar): DC 指的是每個 Channel 只減一個值 (可以參考 ~input\model\customer\inception_v2\mean_data.txt) , Planar 指的是每個 Channel 減掉 mean 值圖 (通常會以 mean_data.binaryproto 形式配置)

```
## mean data mode
```

```
# 0: MEANSUB_DC
```

```
# 1: MEANSUB_PLANAR
```

```
[preproc/meansub/mode] = 0
```

- [18] Mean subtraction 的 mean 資料檔案格式:

```
## mean data format
```

```
# 0: TXT
```

```
# 1: BINARYPROTO
```

```
[preproc/meansub/fmt] = 0
```



注意事項

若 [preproc/meansub/mode] = 1 (MEANSUB_PLANAR) 時，目前只支持
[preproc/meansub/fmt] = 1 (BINARYPROTO)

- [19] mean_path 的 Mean 的資料順序:

```
## 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
```



注意事項

若 [preproc/meansub/mode] 設置為 Planar mode, 由於讀入的 .binaryproto 格式為 BGR 格式，故 [preproc/meansub/fmt_type] 也記得要修改成對應的 format (4)

- [20] Mean subtraction 後所乘上的常數 (1/σ) 是否要開:

```
## [NORMALIZE]
```

0: disable normalization

1: enable normalization

[preproc/normalize/en] = 0

[21] Mean subtraction 後所乘上的常數 ($1/\sigma$)，其值域為 [0, 1]:

normalize scale

[preproc/normalize/scale] = 0.017



注意事項

1. [preproc/normalize/scale] 還可以設置 3 個 float 數值，之間用逗號 ',' 隔開，此三個值對應的是 [preproc/out_fmt] 排布設置，例如：

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

[preproc/out_fmt] = 3

那 channe_B 的 norm scale 為 0.017, channel_R 的 norm scale 為 0.019

2. 當 [preproc/normalize/scale] 設置為 3 個 float 數值時，前提需開啓

[preproc/normalize/en]，且 [preproc/out_fmt] 必須設置為 PREPROC_OUT_FMT_RGB 或 PREPROC_OUT_FMT_BGR

[22] 前處理硬體層(NUE2)的輸出格式:

[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] = 3

[23] 前處理硬體層(NUE2) 的 YUV2RGB 套用參數組設置

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]*/

[preproc/yuv2rgb/csc] = 0



注意事項

此參數只支援 530 / 331 / 336 平台

[24] 是否要打印 top5 分類結果:

[POSTPROCESS]

0: disable post process (classify accuracy)

1: enable post process (classify accuracy)

[postproc/en] = 1

[25] DRAM 輸入特徵調為 16bit:

[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] = 0

[precision/in_hp/roipool_en] = 0

0: 由工具調節 (默認值)

1: eltwise 單獨開啟: 16bit (int16), eltwise 非單獨開啟: 16bit (sign 由工具決定)

2: eltwise 單獨開啟: 8bit (int8), eltwise 非單獨開啟: 8bit (sign 由工具決定)

[precision/in_hp/eltwise_mode] = 0

0: 16bit (默認值)

1: 8bit

[precision/in_hp/sigmoid_mode] = 0

由於在某些情況下 sigmoid 直接設為 8bit 可能會影響精度，因此通過 [precision/sigmoid_hp_int_thd] 來控制 sigmoid 是否調回 16bit（即使 sigmoid_mode 設為 1，sigmoid_hp_int_thd 仍有可能將 sigmoid in bit 改回 16bit），默認是 3，可以選擇的範圍是{3, 4, 5, 6, 7, 8}，此值越大，越容易將 in bit 調 16bit。若 sigmoid_mode 為 0，此選項不起作用。

[precision/sigmoid_hp_int_thd] = 3

[26] DRAM 輸出特徵調為 16bit

[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: 由工具調節 (默認值)

1: eltwise 單獨開啟: 16bit (int16)，eltwise 非單獨開啟: 16bit (sign 由工具決定)

2: eltwise 單獨開啟: 8bit (int8)，eltwise 非單獨開啟: 8bit (sign 由工具決定)

[precision/out_hp/eltwise_mode] = 0



注意事項

[precision/out_hp/eltwise_mode] 主要是調整設置 bitdepth 的優先級來實現該功能，但並非是絕對性的設置，在以下情況下可能仍是由工具考量硬體限制配置

1. 當此 eltwise 為 sigmoid 拆分出來時
2. 當 eltwise 所在的 confusion 中包含 bnmul 時
3. 當此 eltwise 為 lstm 拆分出來時
4. 當此 eltwise 是 hp-conv_add 模式拆出來時
5. 當此 eltwise 是 prelu 方案拆出來時
6. 當此 eltwise 為 hp_meansub 時
7. 當 eltwise 後面接的 op 要求輸入為 16bit 時，例如 sigmoid、tanh、bnmul、16bit hpconv 時
8. 當 eltwise 後面接 slice 時，slice 會 bypass 傳遞 bit 設定，只要 slice 所接的 op 中有一個要求輸入 16bit，eltwise 就輸出 16bit，且 slice 所接的其他 layer 的輸入也會是 16bit
9. 當 eltwise 後面接 concat 時，concat 也會 bypass 傳遞 bit 設定，與 slice 不同的時，接 concat 時受到兩個方面的影響

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.

- a. 當 concat 後面的 op 要求輸入為 16bit 時，此時所有接入 concat 的 layer 的輸入都要為 16bit，此時 eltwise_mode 不起作用
- b. 當接入 concat 的所有 op 中只要有一個要求輸出為 16bit，則所有接入 concat 的 op 的輸出都為 16bit，eltwise_mode 不起作用

[27] 平衡 conv+bn 之間的 weight 分布的閾值倍率，當 conv+bn 結構出現，若 conv 或 bn 的 weight 比所有 channel 平均高或低 out_range_ratio 倍，則調整 conv/bn 的 weight (此功能尚未大量驗證，此值若設為極大值，則平衡機制不會啟動)

[BALANCE WEIGHT]

outrange rate [1, 1023]

[precision/balance_weight/out_range_ratio] = 1000.000000

[28] 若有 func+relu 結構出現，此開關決定 func 是否參考 relu 輸出來分配當層 bit (relu negative slope = 0 或極小值時，建議開啟):

[REF RELU OUTPUT]

0: current reference its layer out

1: current reference next relu out

[precision/ref_relu_outval_en] = 1

[29] 當遇到 Mobile-Net 系列，或是特徵圖 Std. 過小且 Scale 過大時，可以打開 CLE 調適至適當的值域，但遇到參數值過小的層若做 CLE 反而會可能使 Bias 值暴增而使輸出特徵圖值域暴增，建議在網絡包含 Depthwise-convolution 且網絡結果精度不足情況開啟

[CROSS LAYER EQUALIZATION]

0: turn off CLE

1: turn on CLE

[precision/cle_en] = 0

[30] 設置校調精度作法的開關組合

[CALIBRATION]

0: GEN_REF_CALIB_MAX

1: GEN_REF_CALIB_L2

2: x

3: x

4: GEN_REF_CALIB_PERCENTILE

[precision/calib_mode] = 0

當 [precision/calib_mode] 不是 GEN_REF_CALIB_MAX，且該層特徵點數小於 [precision/calib_min_actnum] 時會自動轉回 GEN_REF_CALIB_MAX 實現，這是避免 L2 或是 PERCENTILE 對於一些小的 feature map 情況結果反而不佳的微調方案

[precision/calib_min_actnum] = 1

當 [precision/calib_mode] = GEN_REF_CALIB_PERCENTILE，設置 percentile 的閾值，當層會針對參考影像特徵值，由大而小排序，並找對應分布中位於 [precision/percentile] 比例的值當 calibration 值

[precision/percentile] = 0.9999

[31] 設置當 avg_pooling 遇到輸出的 frac_bit 比輸入的 frac_bit 大時，是否啟動優化算法，避免結果溢出，默認是開啟

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] 使用拆分支持 conv 16bit 高精度運算

0 : 根據工具自動配置原始 bit 運算 (默認值)

1 : 拆分所有 conv/deconv 層支持高精度運算

2 : 拆分指定的 conv/deconv 層支持高精度運算 (根據在 config 文件夾下的 hpconv_config.txt)

[precision/in_hp/conv_op] = 0



注意事項

1. 在 `hpconv_config.txt` 中指定的 `conv/deconv` 層寫法如下 (可根據 `proc_bit_info.txt` 內 `golden_layer_name` 欄位名稱填寫自定義高精度運算的層)

`Convolution_conv_1_Y`

`Convolution_conv_2_Y`

2. 開啟高精度運算的影響會是耗時與帶寬的增加，需搭配應用需求使用

[35] 控制工具對於 PRelu 層的拆分方式，默認值為 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] 當要測試未訓練過的模型在板端上的效能時往往會因為 `weight` 值域不對導致超出 HW 限制報錯，這時可以使用此開關進行參數 `clamp` 生成測試用模型

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] 設置是否開啟 8bit 量化模型壓縮來減少 memory 及 model size: (精度有損)

#[WEIGHT COMPRESSION]

0: disable quatization

1: enable quatization

[compression/method/quan_en] = 0

[38] 設置是否開啟 k-means 有損壓縮，此功能必須同時打開 VLC 開關，且不能跟 `quan_en` 同時開啟，NT98331 不支持此功能 (精度有損)

0: disable kmeans quantization (Default)

1: enable kmeans quantization

[compression/method/kmeans_en] = 0

[39] 設置是否開啟 VLC weight 無損壓縮: (精度無損)

0: disable variable length coding

1: enable variable length coding

[compression/method/vlc_en] = 0

[40] 當設置 **[compression/method/vlc_en] = 1** 時，是否開啟混合模型壓縮方法來避免有些模型在壓縮上的負面影響

0: disable mixed compression

1: enable mixed compression

[compression/use_mixed_compression] = 0



注意事項

一般還是先使用 **quan_en = 1** 即可，若是有 **model size** 需縮小或是整合業務時有耗時增加太多情況，再建議開啟 **vlc_en** 或搭配 **use_mixed_compression** 來測試最佳情況

[41] 另外列出一些對齊 5.3 章 cmd config 的設置參數如下，若同時與 cmd config 設置會以 cmd config 的為主

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 + intermediate 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
 # 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: 由工具決定要使用 CPU 層或是 HW 層 (默認值)
 # 1: 使用 HW 切分 Softmax
 # 2: 強制使用 CPU Softmax
[platform/softmax] = 0

[43] 當使用 NT98530 平台時，指定 {output_tensor_name} 對應層使用 backend {backend_type}，其中 backend_type 可設置值如下
 # EXT: 代表 DSP 或 CPU 皆可運行，優先級會由工具自行判斷
 # DSP: 此 layer 由 DSP 運行
 # CPU: 此 layer 由 CPU 運行
[tensor_backend/{output_tensor_name}] = {backend_type}



注意事項

當 [platform/softmax] = 2 (該網路所有 softmax 都使用 CPU 運行) 時會強制清除掉 [tensor_backend/{output_tensor_name}] 設定，若想根據單層 softmax 設置成 CPU 運行時 [platform/softmax] 必須為 0 or 1 or 不設置

[44] # 0: 拿掉输出层的 reshape
 # 1: 保留输出层的 reshape (默認值)
[special/keepdim/reshape] = 1

[45] # 0: 輸出層的 nuesoftmax 不加 permute 轉回原始維度
 # 1: 輸出層的 nuesoftmax 需要加兩個 nuepermute 轉回原始維度，如果無法加 nuepermute，該 softmax 會變成 cpusoftmax (默認值)
[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

- 對於多輸入，每個輸入的參考影像和 **preproc** 區塊的變數需分開設置，請以 **[blob] = n**, **[blob_name] = datan** 區隔不同輸入，以下範例為一雙數入的設置方法



注意事項

每個輸入都需要有對應的 **config** 設置，另外 **blob_name** 要與 **onnx model** 的輸入的名稱一致，不然沒法順利轉換模型

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

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.

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

... (中間省略)

## [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_INT16
# 3: TYPE_UINT16
[preproc/in/type] = 1

... (中間省略)

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

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: 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. 下表列出

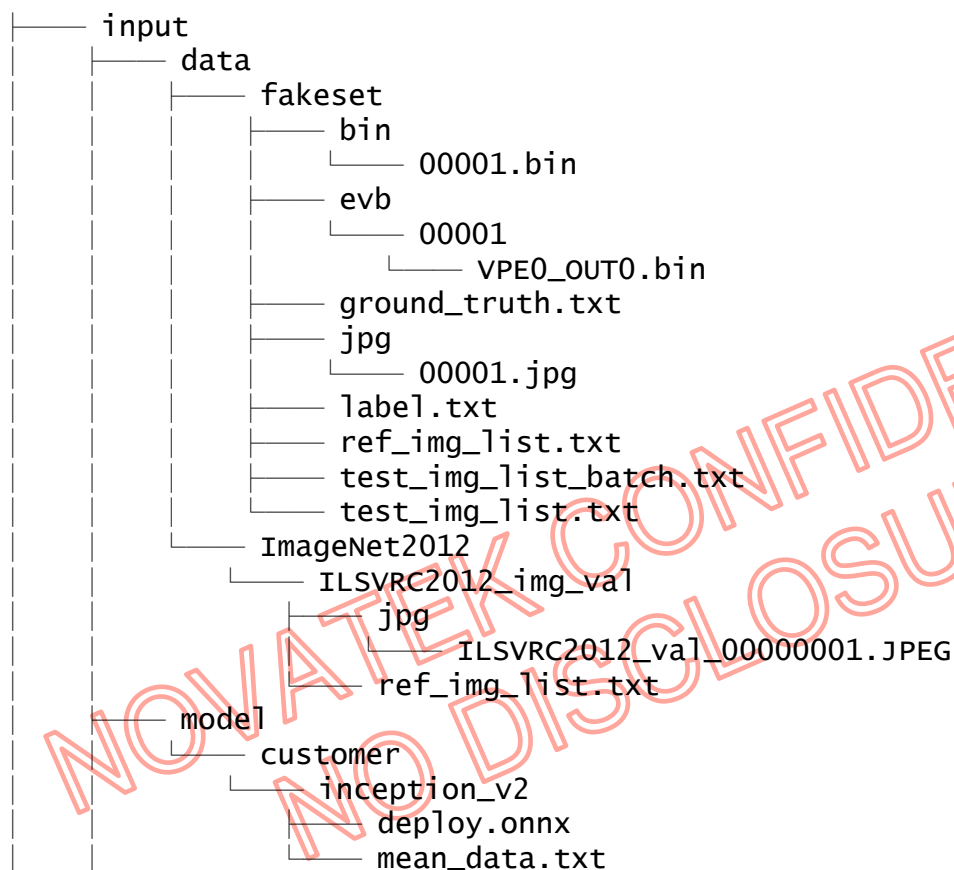
- [1] [preproc/in_fmt]
- [2] [preproc/meansub_fmt_type]
- [3] [preproc/out_fmt]

有效的設定組合，請勿超出此表所設之配置避免出錯：

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

3.2 輸入資料

Gen-tool 的輸入資料檔案默認是放置在 `nvtai_tool` 資料夾下，以 `00001_inception_v2` 為例，其資料結構的設置如下面所示



說明如下：

- `~\input\data\`
 - 檔案\資料夾: [ImageNet2012](#), [fakeset...](#)
 - 說明: 預先參考影像，分析 `dynamic fixed point` 相關數值，目前若輸入格式為 `feature bin`，參考影像可讀 `JPEG/jpg` 格式，若輸入格式為 `feature bin`，參考影像可讀 `bin` 格式
 - 檔案\資料夾: [ref_img_list.txt](#)
 - 說明: 每筆測試資料以空行方式隔開，如下圖

```
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
```



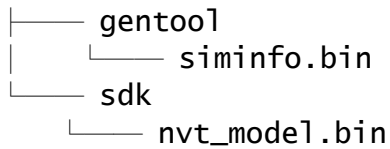
注意事項

ref_img_list.txt 內所列筆數若是小於 [ref_data/num] 所設置的數目，會報出以下錯誤
 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
 - 檔案\資料夾: [deploy.onnx](#)
 - 說明: 放置要轉換的網絡模型，若是 Caffe model ([deploy.prototxt](#), [deploy.caffemodel](#)) 也可放到此資料夾
 - 檔案\資料夾: [mean_data.txt](#)
 - 說明: Mean Image 可為下列形式
 - ◆ Binaryproto, 同一般 Caffe 可支持的 Binaryproto 格式
 - ◆ txt 格式

3.3 輸出資料

在 gen_config.txt 使用者可以透過給定 [path/out_dir] 來設置 Gen-tool 想要輸出的結果位置 (默認為 ..\nvtai_tool\output)，其輸出文件夾結構如下圖所示



說明如下：

- ~\output\網路名稱\gentool\
 - 檔案\資料夾: [siminfo.bin](#)
 - 說明: 用於 Sim-tool 仿真時的參考檔
- ~\output\網路名稱\sdk\
 - 檔案\資料夾: [nvt_model.bin](#)
 - 說明: 轉換後的二進制資料，使用在板端與仿真測試上

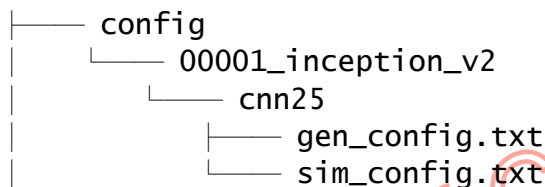
NOVATEK CONFIDENTIAL
NO DISCLOSURE!

4 Sim-tool 介紹

Sim-tool 是用來模擬版端的行為，當 Gen-tool 跑完後，使用者可以透過 Sim-tool 在 PC 端仿真在板端的運行結果並一併與浮點模型比較了解定點模型的效果。

4.1 Config 設置

Config 檔案默認是放置在 nvta_tool 資料夾下，以 00001_inception_v2 為例，其資料結構的設置如下面所示



1. 開啟 `config\00001_inception_v2\cnn25\sim_config.txt` 後，其內容可分為單輸入以及多輸入格式，在此先使用單輸入為範例說明各個參數設置

- [1] 設置 Label/Model 的路徑 (支持相對於 `--config-dir` 設置的路徑之下，或是絕對路徑)

```

### [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] 設置測試資料路徑: (支持相對於 `--config-dir` 設置的路徑之下，或是絕對路徑)

```

### [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
[path/test_list_path] = ..\nvta_tool\input\data\fakeset\test_img_list.txt
  
```

- [3] 設置 Sim-tool 執行完的輸出資料夾: (支持相對於 `--config-dir` 設置的路徑之下，或是絕對

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.

路徑)

[SIMULATION]

simulation output root

[dataset] 會在 [path/out_dir]/results 與 [path/out_dir]/simtool 下各產生 [dataset] 名稱的資料夾存放測試影像的結果，建議與測試影像資料夾同名

[dataset] = fakeset

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



注意事項

與 [gen_config](#) 設置時不同，在 [sim_config](#) 設置輸出路徑時，記得要加上網路名稱，例如 [..\nvtai_tool\output\0001_inception_v2](#)

- [4] 設置輸入測試資料的格式:

src format

JPEG = 0,

BIN = 1,

[src/imgfmt] = 1

- [5] 輸入資料是否做過 preprocess

preproc out bin

0: is preproc input bin

1: is preproc output bin

[src/is_preproc_out_bin] = 1

- [6] 輸入資料的型態

src bin file type

BLOB_TYPE_INT8 = 0,

BLOB_TYPE_UINT8 = 1,

BLOB_TYPE_INT16 = 2,

BLOB_TYPE_UINT16 = 3,

[src/bintype] = 0

- [7] 設置輸入測試資料的尺寸，介於[1, 4096]:

src size [1, 4096]

[src/width] = 224

[src/height] = 224

[src/channel] = 3

[src/batch] = 1

[src/time] = 1

- [8] 開啓后，輸入圖檔會先經由 `opencv resize` 成 `sim_config` 設定的 `## src size (width/height)` 後再轉成 `nue2` 輸入的 `fmt` (默認關閉)：

[JPG INPUT RESIZE]

0: disable jpg resize

1: enable jpg resize

[src/resize_en] = 0

- [9] 開啓后，輸入圖檔如果 `width` 或 `height` 為奇數，會在下側/右側 `padding` 一行/一列為 0 的像素：

[PAD JPG W/H to EVEN]

0: disable jpg padding

1: enable jpg padding

[src/padding_en] = 1

- [10] 開啓后，輸入圖檔如果 `width` 或 `height` 為奇數，會捨棄下側/右側一行/一列 (默認關閉)，不能與 `[src/padding_en]` 同時開啟：

[CROP JPG W/H to EVEN]

0: disable jpg cropping

1: enable jpg cropping

[src/crop_en] = 1

- [11] 在 NT98530 平台上靜態指定 `EXT` 於 `sim` 以何種 `backend` 運行

DSP: 此 layer 由 DSP 運行

CPU: 此 layer 由 CPU 運行 (默認值)

[backend/convert_ext_to] = CPU

2. 多輸入網路設定需要用這兩種標籤 (`[blob]`以及`[blob_name]`)，來區別不同 `input` 的設定
`[blob_name]` 設定的是原始網路 `input` 的 `tensor name`，設置方式可以參考以下範例：



注意事項

每個輸入都需要有對應的 **config** 設置，另外 **blob_name** 要與 **onnx model** 的輸入一致

```
[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

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

Copyright © 2018 Novatek Microelectronics Corp. All Rights Reserved.

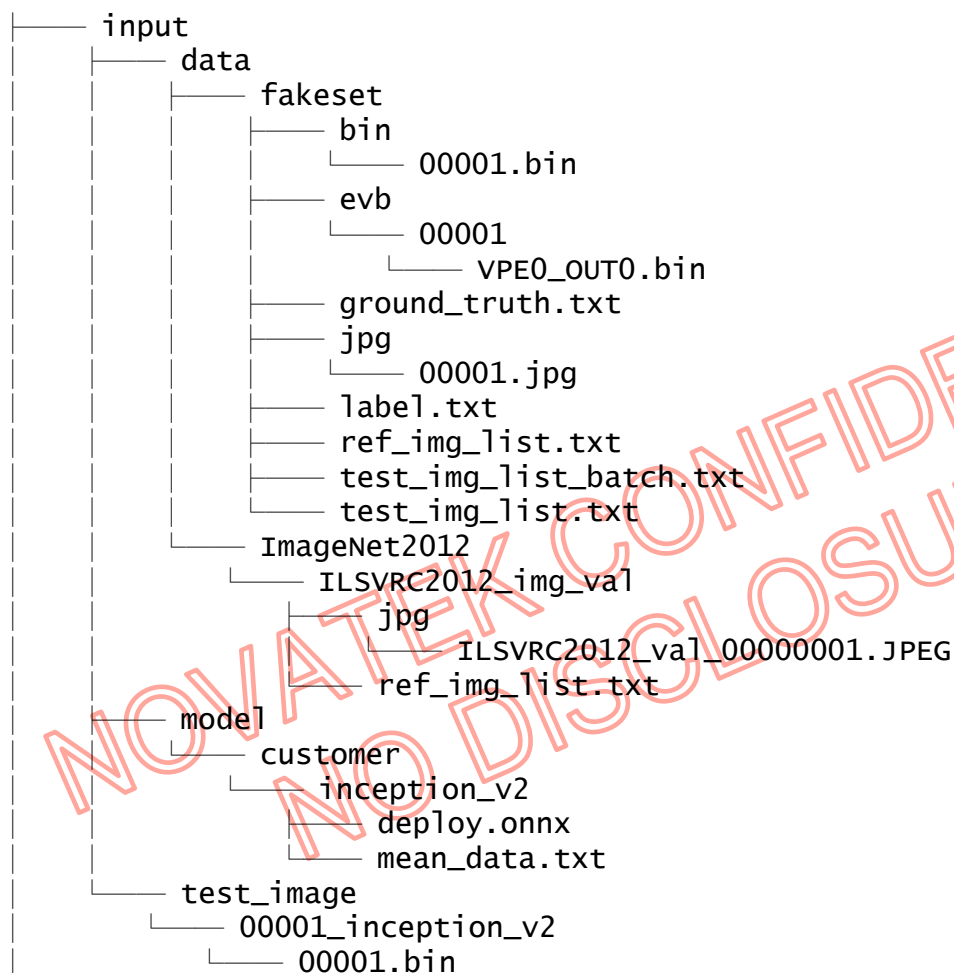
With respect to the information represented in this document, Novatek makes no warranty, expressed or implied, including the warranties of merchantability, fitness for a particular purpose and non-infringement, and does not assume any legal liability or responsibility for the accuracy, completeness or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


```
# BLOB_TYPE_INT8 = 0,  
# BLOB_TYPE_UINT8 = 1,  
# BLOB_TYPE_INT16 = 2,  
# BLOB_TYPE_UINT16 = 3,  
[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 輸入資料

Sim-tool 的輸入資料檔案默認是放置在 `nvtai_tool` 資料夾下，以 `00001_inception_v2` 為例，其資料結構的設置如下面所示

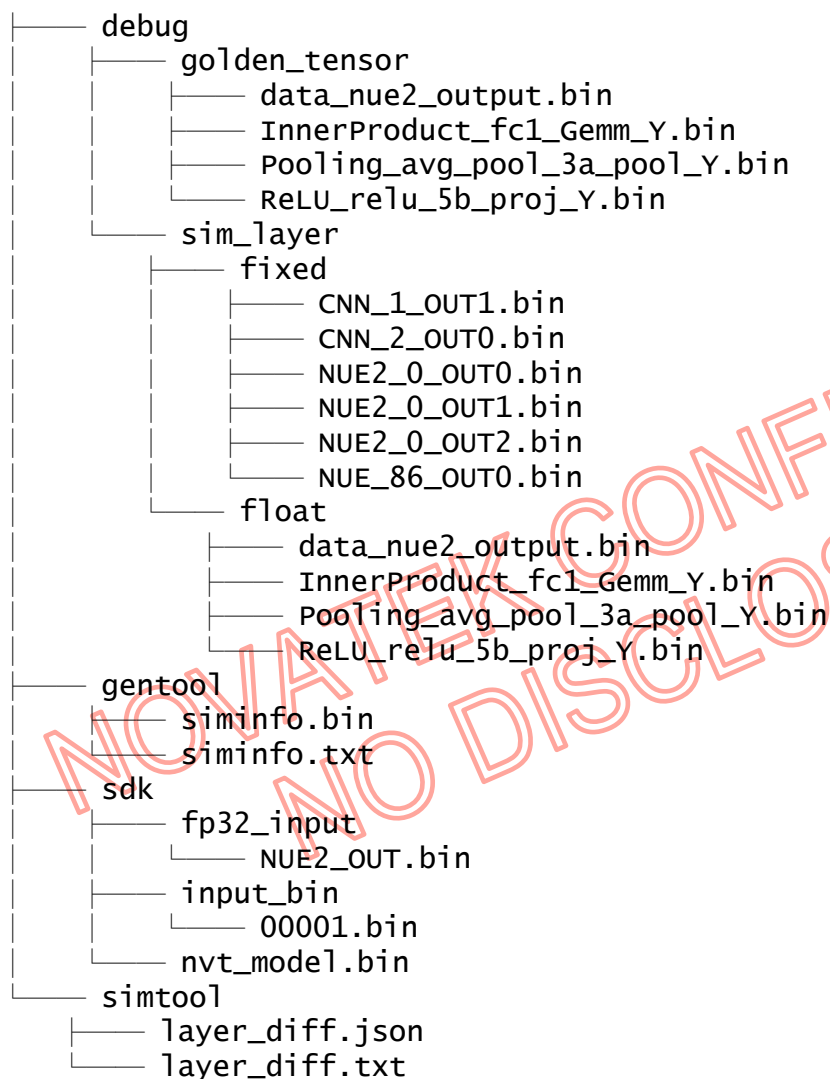


說明如下：

- `~\input\model\customer\inception_v2`
 - 檔案\資料夾: `deploy.onnx`
 - 說明: 放置要轉換的網絡模型，若是 Caffe model (`deploy.prototxt`, `deploy.caffemodel`) 也可放到此資料夾
- `~\input\test_image\00001_inception_v2`
 - 檔案\資料夾: `00001.bin`
 - 說明: 放置要測試的輸入影像檔案，由板端產出

4.3 輸出資料

使用者可以在 `sim_config.txt` 透過給定 `[path/out_dir]` 來設置 Sim-tool 想要輸出的結果位置 (默認為 `..\nvtai_tool\output\00001_inception_v2`)，其輸出文件夾結構如下圖所示



說明如下：

- `~\output\網絡名稱\debug\`
 - 檔案\資料夾: `golden_tensor`
 - 說明: ONNX 原生網路 float 運算結果，用來當作 golden 答案
 - 檔案\資料夾: `sim_layer/fixed`
 - 說明: Sim-tool 運行二進制 `nvt_model.bin`，跑出 fixed buffer 結果
 - 檔案\資料夾: `sim_layer/float`

- 說明: 將 `sim_layer\fixed` 還原為 `float` 結果，後續跟 ONNX golden float 結果做比對並產出 `layer_diff`
- Debug (golden_layer v.s sim_layer) 過程
 - ◆ 比對 `golden_layer` 與 `sim_layer` 的方法首先會先開啟
 `~\simtool\fakeaset\layer_diff.txt` 確認要查詢的 layer name 後在
 `sim_layer\float` 與 `golden_layer\` 下搜尋相同的 layer name bin file 即可
- `~\output\網路名稱\gentool\`
 - 檔案\資料夾: `siminfo.txt`
 - 說明: Sim-tool 產出，為 `siminfo.bin` 的轉換文字檔，debug 用，運行中自動產出
- `~\output\網路名稱\sdk\`
 - 檔案\資料夾: `input_bin`
 - 說明: 放置用來給 Sim-tool 運行的 input 檔
 - 檔案\資料夾: `fp32_input`
 - 說明: 放置用來給 ONNX 原生網路的 input 輸入檔
- `~\output\網路名稱\simtool\`
 - 檔案\資料夾: `layer_diff.txt & layer_diff.json`
 - 說明: 記錄了每層中 Sim-tool 與 ONNX 特徵值的差異，如下圖所示，其中判斷標準為
 - ◆ average absolute difference ratio (avg_diff): 其正常值通常要 **< 0.02**
 - ◆ cosine similarity (cosine): 其正常值通常要 **> 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)


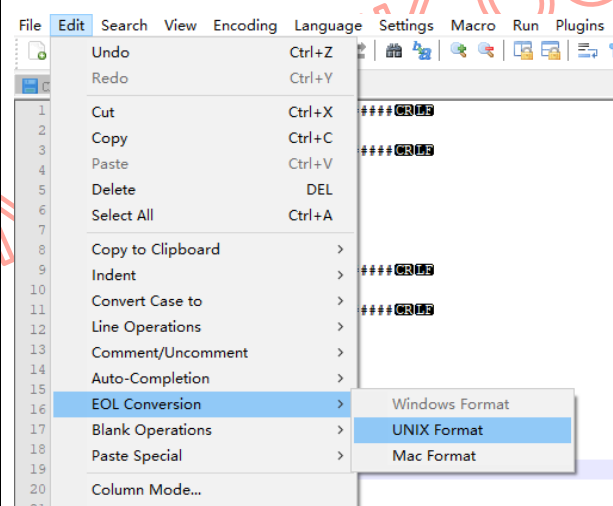
NOVATEK CONFIDENTIAL
NO DISCLOSURE

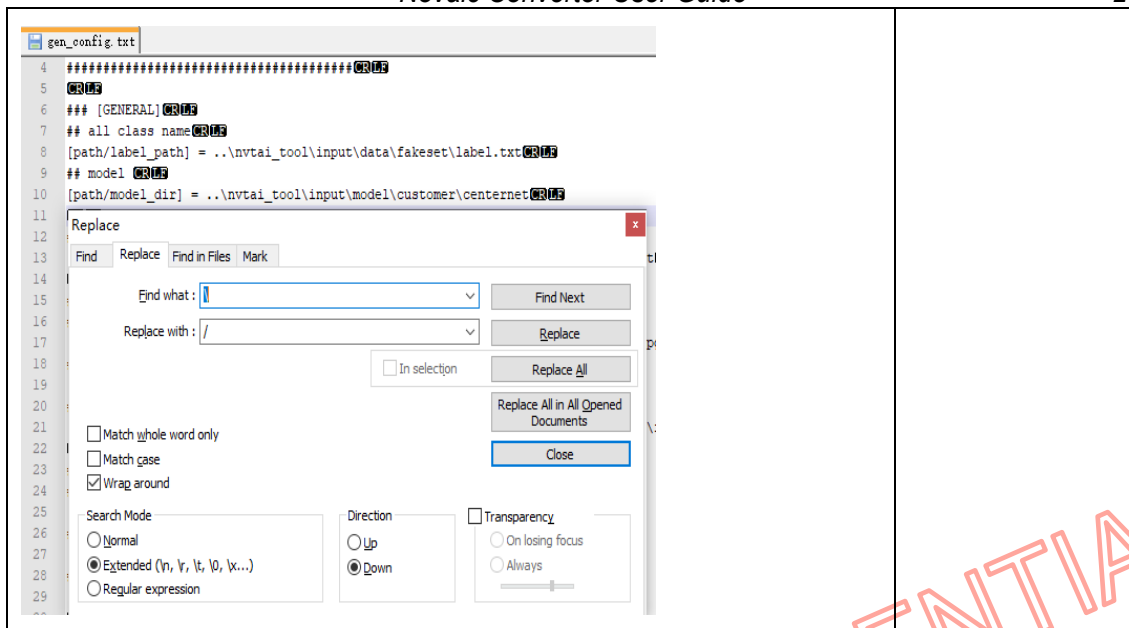
5 工具運行樣例說明

在此，我們使用 `test-tutorial` 下的 `00001_inception_v2` 範例來介紹工具轉換網絡的流程。

5.1 檔案預處理

1. 使用者可以使用 `tes-tutorial\nvtai_tool\config\00001_inception_v2` 底下的 `gen/sim config` 檔案，或是可以自行產生 `config` 檔案並將其原生的檔案覆蓋過去

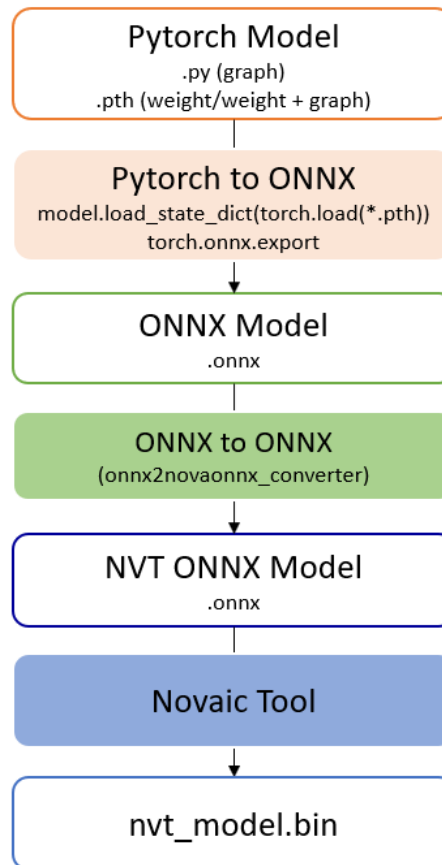
 注意事項	
描述	備註
<p>若在 Linux 系統上使用到 Win 版本，<code>input</code> 與 <code>config</code> 資料夾下所有 <code>.txt</code> 檔案需事先轉成 UNIX 格式。</p> 	<p>此項修改可使用 <code>notepad++</code></p>
<p>若在 Linux 系統上使用到 Win 版本的 <code>config</code> 與 <code>input</code>，內部的路徑分號由 <code>\</code> 改成 <code>/</code>。</p>	<p>此項修改可使用 <code>notepad++</code></p>



5.2 ONNX 模型轉換

由於目前大部分的算法模型皆是使用 **Pytorch** 框架訓練，所以 **Novaic Tool** 也開始可以支持客戶由 **Pytorch** 轉換成 **ONNX** 模型的轉換，以下介紹轉換的方法，會分成兩個部分

- **Pytorch to ONNX**
- **ONNX to ONNX**



5.2.1 Pytorch to ONNX

Pytorch 轉 ONNX 時首先是需要將訓練完成的 Pytorch 模型文件使用 Pytorch 的 `torch.onnx.export()` 進行轉換。此 API 參數說明可以參考以下連結：

<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]
  
```

下面是一個簡單的示例

```

model.eval()
input_names = ["input_1"]
output_names = ["output_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.


```
# 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)
```

參數設置的說明如下：

- `export_params=True` #轉換模型時須將 `weights` 也存至檔案中
- `do_constant_folding=False` #在這裏關閉，避免模型帶有 `bn_folding` 設置
- `opset_version=12` #這邊需設置工具可支持的 `opset` 版本
- `training` #使用默認值。因為 `onnx` 模型一般爲了推斷，所以不需要設置

要注意的是在使用 `torch.onnx.export()` 前必須先運行 `eval()` or `train(False)` 避免轉換時報錯。另外要注意的是 Novaic Tool 支持的 `opset` 版本為 `opset=12`，若是 Pytorch 版本不支持 `opset=12`，可以先轉爲 `opset=8 ~ opset=11`。在後續 ONNX to ONNX 過程時可透過 `onnx2novaonnx_converter.py` 轉成 `opset=12`



注意事項

如果直接用 `opset12` 的 `model`, `onnx` 版本使用 1.7.0 即可，但如果網路不是 `opset12` 會先透過 `onnx converter` 轉成 `opset 12`，目前工具需要 `onnx` 版本使用 1.9.0 才有完整支援轉到 `opset12` 的功能，所以使用 `opset12` 以外的 `model` 需要將 `onnx` 升級到 1.9.0，否則可能會遇到某些 `op` 不支援的報錯

對於 `pooling op`, 在 Pytorch 轉 ONNX 時不會帶有 `countIncludePad` 這個參數 (既有設定 `count_includ_pad`)。當 ONNX model 不帶有此參數時，工具內的默認值為 0。此時，如果計算區域涵蓋到 `pad`，這層 `pooling` 就會變成 `cpu layer`。雖不影響輸出結果，但會影響到效能。所以使用者需要在轉完 ONNX model 後自行修改 ONNX model 並加入 `countIncludPad` 這個 `attribute` 讓工具知道是否要對 `pool` 計算包括 0 填充

5.2.2 ONNX to ONNX

由於 Pytorch 轉換後的 ONNX model 對於 Novaic Tool 上還缺少一些資訊，或是 `opset` 版本需要調整，所以在以下路徑提供了 ONNX to ONNX 的轉換工具

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

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.

onnx2novaonnx_converter.py

共兩個參數，input 指定輸入 onnx model 路徑，output 指定輸出 onnx model 路徑，指令如下

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

執行完後即可在指定路徑下獲得轉好 Novaic Tool 可以轉換的模型: deploy.onnx

若輸入的是 opset12 以外的 model 會用 onnx converter 嘗試轉換成 opset12，目前確定支援的是 opset=8 ~ opset=12，其餘 opset 版本可能會有 onnx converter 相關報錯
另外 opset=8 或 opset=9 轉 opset=12 需要 onnx python lib 為 1.9.0 版以上

另外要注意的是，為了避免 bn_folding，需要在 onnx2novaonnx_converter.py 中將 skip_fuse_bn=True (line: 159) 開啓

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

5.3 運行 Gen-tool 程序

1. 也可以選擇自行修改 tes-tutorial\invtai_tool\config\00001_inception_v2 的 參數來測試範例。



注意事項

這邊預設測試影像預設為過 NUE2 前處理後的輸出 bin 檔 (3-Channel, RGB 或 BGR 形式)

2. 把 caffe model 轉成 onnx model

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

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.

```
/home/[user]/ai_tool/novatek/novaic/test-tutorial
/nvtai_tool/input/model/customer/inception_v2/deploy.caffemodel  deploy
/home/[user]/ai_tool/novatek/novaic/test-
tutorial/nvtai_tool/input/model/customer/inception_v2
```

3. 使用 Novaic Convertor 的 Gen-tool (compiler) 把 onnx model 轉成 nvt_model.bin
closeprefix/bin/compiler.nvtai --config-dir /home/[user]/ai_tool/novatek/novaic/test-
tutorial/nvtai_tool --pattern-name 00001_inception_v2 --chip 336



注意事項

以上 3. 與 4. 兩點，可以簡化為底下命令，一次完成

```
closeprefix/bin/compiler.nvtai --config-dir /home/[user]/ai_tool/novatek/novaic/test-
tutorial/nvtai_tool --pattern-name 00001_inception_v2 --chip 336 --use-caffe
```

- [1] 以下表格整理在 Gen-tool (compiler) 使用到的參數說明

參數名稱	參數值	參數說明
--config-dir	設置放置 config 的路徑	必須設置此參數，可設置絕對路徑或是相對於 toochian\ 文件夾的相對路徑
--pattern-name	設置要轉換的網絡名稱	必須設置此參數
--use-caffe	-	設置此參數可簡化流程，自動將 caffe 模型轉成 onnx 模型後進工具轉換。
--verbose	0: FATAL 1: ERROR 2: WARNING 3: INDEX 4: USER	可動態設置打印的等級，設置的等級值越小，打印越少，0 為不打印，默認等級為 4
--quan-en	0: 關閉 8-bit weight 壓縮 1: 開啟 8-bit weight 壓縮	默認為 0 (關閉)
--chip	52x: NT9852x 528: NT98528 56x: NT9856x 32x: NT9832x 336: NT98336 331: NT98331 (32bit)	無默認值，一定要設置此參數，且要跟 simulator 使用的 chip 設置一致

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.

	331_A64:NT98331 (64bit) 530: NT98530	
--msb-shrink-en	0: 原格式(完整輸出) 1: depthwise 會精簡格式	默認為 0 (關閉)，精簡格式效率較好，model size 較小，但在較舊版本的 SDK 會無法運行
--job-opt	0: Debug at linear mode 1: Linear mode 10: Debug at graph mode 11: Graph mode	1. 不設置時是由板端配置 job-opt 選項 2. 要注意的是此參數要與 --buf-opt 一起設置 3. 目前只支持設置 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. 不設置時是由板端配置 buf-opt 選項 2. 要注意的是此參數要與 --job-opt 一起設置
--use-tiling	0: 關閉 1: 開啟 tiling 功能 (只考慮 iobuf 的配置)	Tiling 功能可以減緩當帶寬吃緊時所帶來的運算耗時增加影響 默認為 0，要注意的是此參數不為 0 時，--chip 只能設置 530
--use-heteroge	0: 關閉 1: 開啟	ACL 默認是關閉的，使用者可以使用這個 flag 將其開啟，且遇到帶有 ACL op 的 ONNX 模型會擋住報錯
--buf-thld	1 ~ 65535	在分析 graph 複雜度時的 buf 配置複雜度的閾值，默認是 312
--graph-thld	1 ~ 65535	在分析 graph 複雜度時的 graph 複雜度閾值，默認是 128
--dump-onnx-en	NA	Dump onnx 功能默認是關閉的，當開啟時，且 --use-tiling 關閉時，會在 \gentool 資料夾下看到 onnc_model.onnx, legal_nvt_model.onnx, calib_nvt_model.onnx, codeemit_nvt_model.onnx 四個除錯 onnx 檔案，若 --use-tiling 開

		啟時，則是會多一個 tiling_nvt_model.onnx 檔案，可以由這些 檔案確認每個模組的結果
--	--	------------------------------------------------------------



注意事項

Graph 分析是用於判斷網路因為太過複雜，在板端配置 graph mode, buf-opt = 3 或是帶有 graph 資訊時有配置內存過久或是 nvt_model 過大的風險。這時，需要重新配置合理的參數。而判斷 graph 是否複雜的閥值則是由 buf-thld 與 graph-thld 控制。當兩者皆設置 1 時強制重新配置為 linear mode, 且 buf-opt = 1, 板端不能配置。當兩者皆設置 65535 時則是強制關閉 graph 分析功能



注意事項

若參數設置與 gen/sim config 設置衝突時，例如 quan_en 在 cmd line 設置為 1，但 gen_config 內設置為 0 時，會以 cmd line 的設置為主

5.4 運行 Sim-tool 程序

1. 使用 Novaic Convertor 的 Sim-tool (simulator) 與 nvt_model.bin 得到 float & fixed 的 cosine 相似性與 avg_diff 差異

```
closeprefix/bin/simulator.nvtai --config-dir /home/[user]/ai_tool/novatek/novaic/test-tutorial/nvtai_tool --pattern-name 00001_inception_v2 --chip 336
```

[1] 以下表格整理在 Sim-tool (simulator) 使用到的參數說明

參數名稱	參數值	參數說明
--config-dir	設置放置 config 的路徑	必須設置此參數，可設置絕對路徑或是相對於 toochian\ 文件夾的相對路徑
--pattern-name	設置要轉換的網絡名稱	必須設置此參數
--verbose	0: FATAL 1: ERROR 2: WARNING	可動態設置打印的等級，設置的等級值越小，打印越少，0 為不打印，默認等級為 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.

	3: INDEX 4: USER	
--chip	52x: NT9852x 528: NT98528 56x: NT9856x 32x: NT9832x 336: NT98336 331: NT98331 (32bit) 331_A64:NT98331 (64bit) 530: NT98530	無默認值，一定要設置此參數，且要跟 compiler 使用的 chip 設置一致 Note: 若使用 32x 設定，輸入 jpg 時是轉成 NV12 格式，所以若 32x 板端上使用的 NV21 格式要記得再轉成 NV12 給 sim 避免掉點



注意事項

有時使用的網路較大，所配置的未優化 iobuf size 可能踩到硬件限制而無法生成 cos/diff 結果時可在 cmd 加上 --emu-bufopt 1 使用優化後 iobuf size 來仿真。

2. 執行完上面步驟後可依照以下方式確認結果是否正確

- [1] 確認是否有產生 deploy.onnx
/home/[user]/ai_tool/novatek/novaic/test-tutorial/nvtai_tool/input/model/customer/inception_v2/deploy.onnx
- [2] 確認是否有產生 nvt_model.bin
/home/[user]/ai_tool/novatek/novaic/test-tutorial/nvtai_tool/output/00001_inception_v2/sdk/nvt_model.bin
- [3] 確認是否有產生 layer_diff 檔案
/home/[user]/ai_tool/novatek/novaic/test-tutorial/nvtai_tool/output/00001_inception_v2/simtool/layer_diff.json
/home/[user]/ai_tool/novatek/novaic/test-tutorial/nvtai_tool/output/00001_inception_v2/simtool/layer_diff.txt

6 添加私有層方式

Tool 會將不支援的 layer 歸類為 custom layer。而 custom layer 的參數劃分為以下三種，這三種會以既定格式傳遞到 SDK，可自行解析。

1. .prototxt 檔案描述
2. .caffemodel 中的 weight
3. 自訂的 cust_bin 檔案: 可自帶其他參數以及 Table 資訊



注意事項

目前私有層的支持只限於 Caffe 框架

6.1 建置私有層的流程 (Caffe)

6.1.1 Caffe

1. 首先我們需要修改 `./toolchain/closeprefix/bin/compiler/frontend/caffe-onnx/proto/` 裡的 `caffe_nvt.proto`，加上私有層的定義
2. 在 `./toolchain/closeprefix/bin/compiler/frontend/caffe-onnx` 目錄下執行以下指令來更新 `caffe_nvt_pb2.py`

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

6.1.2 Gen-tool

可參考 [00001_custom_layer_inception_v2](#) 這個範例 pattern。此範例為 inception_v2 網路並指定 pool_1 & pool_2 這兩層為 custom layer。

3. 首先我們需要修改/新增私有層專用的 config.檔案 `cust_config.txt`，需在此設訂 type, name, platform, do_compare, has_parm, in_num, in_bitdepth, in_sign, out_num,

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.

out_bitdepth, out_sign，格式如下圖所示：

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


以 00001_custom_layer_inception_v2 為例，填完值後的結果如下圖

```
[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
```

以下是每個 item 的默認值說明，請參考注意事項填寫

項目	說明
[type]	無默認值，必填
[name]	初始預設為所有相同 type 的 layer
[platform]	可填 CPU 或 DSP，初始為 CPU
[do_compare]	默認值為 0
[has_parm]	初始值由 prototxt parser 判斷決定
[in_num]	默認為上層輸出結果
[in_bitdepth]	默認為上層輸出結果
[in_sign]	默認為上層輸出結果
[out_num]	默認為上層輸出結果
[out_bitdepth]	默認為上層輸出結果

[out_sign]	默認為上層輸出結果
------------	-----------

 注意事項
1. 所有的填寫不允許空白格
2. [type] 為必填項目， <u>必須在第一行</u> ，其他選項可不填，不填則為默認值
3. [name] 如果有填取的話，則只會針對特定 layer_name 的私有層修改，不填的話所有相同 layer type 都會被定義成私有層，可以一次填多個 layer name ，中間用 “；” 區隔。
4. 在同一個 csut_config 內支持設置多個不同 type 的私有層

4. 修改私有層自訂函式: 打開 `ai_tool/novatek/novaic/toolchain/repo/proprietary/onnc-backends/NvtAI/Source_pub/CustomOP/NvtAICustomerUtility.cpp`，以下列出需要修改的函式。

[1] **prototxt** 參數:

自訂層的 **prototxt** 描述中，標示 `xx_param{}` 中的參數 (如下圖)，會以 **string** 的形式傳入，使用者須自行解析自訂義層參數，並將參數存在 **layer_parm**(自訂的 **struct** 形式，如 `NvtAICustomerUtility.h`，中 `CUSTNN_POOL_PROTO_PARM`)

`NvtAICustomerUtility.h` 檔案位置在
`novatek/novaic/toolchain/repo/proprietary/onnc-backends/NvtAI/`
`NvtAICustomerUtility.h`

可參考 `parsing_proto_parm()` 的實現。

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

例如在 `parsing_proto_parm()` 調用 `processPooling` 函式來解析上述內容，解析完會存在 `IR (NvtAICustomer& op)` 的 `Public` 變數 (`op.layer_parm.data()`)裡，可以直接取得

```
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;
}
```

[2] Reshape 函式: (類似 Caffe xxx_layer.cpp 中的 Reshape)

使用者自行計算私有層每個輸出的維度資訊(Tensor Shape)，可參考 ReshapeCustomer() 的實現，若有多個輸出需設定多個。例如以下代碼:

```
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) {
```

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.

```

        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 函式: (類似 Caffe xxx_layer.cpp 中的 Forward_cpu)

使用者自訂私有層的 fp32 inference 計算，可參考 computeCustomer() 的實現。

```

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;
    }

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

```

[4] tmp_buffer 資訊:

使用者自行計算 tmp_buffer 所需要的大小，大小需大於等於 input feature 總共所需要的 buffer 大小 (多 input 的話要相加)，可參考 getCustomerTempBufferSize() 的實現。

```

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;
}

```

由於 `getCustomerTempBufferSize` 實現的位置是在 **Calibration** 之後，使用者若想取得真實量化後的結果，可以在這邊取得。

[5] 獲取當前私有層量化的資訊：

若使用者想獲取當前私有層的量化資訊如

- i. 輸入輸出 **size [n, c, h, w]**，可透過以下方式取得

```
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** 存在 **IR (NvtAICustomer& op)** 的 **Public** 變數裡，可以直接取得

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

- iii. 整個網路的輸入 **size [n, c, h, w]**，紀錄在 **NvtAICustomer** 的 **public** 變數 **std::vector<TensorInfo> net_input_infos;**

其結構體如下：

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

使用範例如下：

```
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];
```

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.

```
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. 上述函式修改完後回到 **toolchain** 目錄底下執行

make rd-onnc-app

重新編譯。

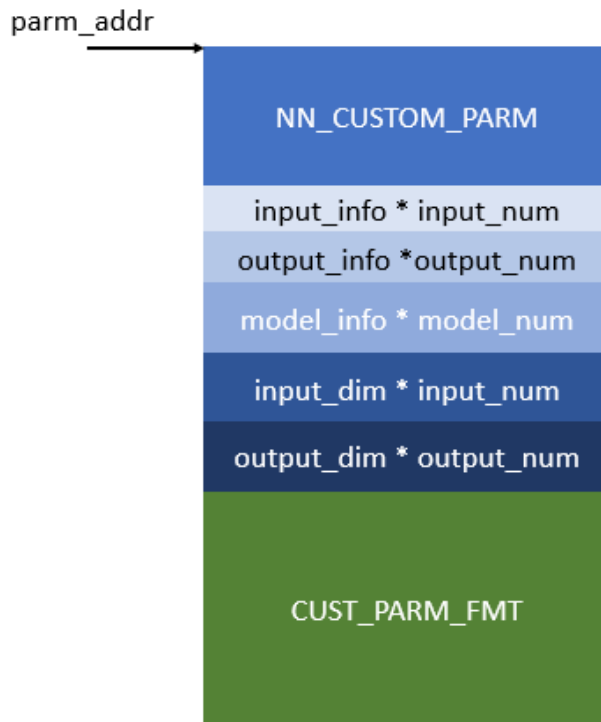


注意事項

1. **caffemodel** 中的 **weight** 會自動讀取進行 **packing**，不須額外設置
2. 如果有額外想要傳遞的參數可以使用 **custom_bin** 傳遞(非必要)，每個值以 **4byte** 為單位打包成 **bin** 檔，檔名為每層 **layer_name**，放置在指定位置 (位置: `..\nvtai_tool\input\model\[model_name]\custom_bin`) 範例請參考: `..\nvtai_tool\input\model\inceptionv2_custom\custom_bin` 中 `pool_1.bin` & `pool_2.bin`。

6.1.3 Sim-tool

1. 修改源碼 vendor_ai_cpu_custnn.c，根據 Gen-tool 傳來的訂義參數，撰寫自訂義函式 vendor_ai_cust()；自訂義參數排列格式如下圖



[1] NN_CUSTOM_PARM 結構如下

```
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;
```

[2] CUST_PARM_FMT 格式如下表:

Item	Type	Description
layer_type_id	UINT32	ai_gen 定義的 custom layer 的 idx，從 0 開始。
weight_num weight_size_1 weight_size_2 ...	UINT32	此自訂義層在 caffemodel 中的 weight 數量，後面接著每個 weight 的 size。
prototxt_parm_len	UINT32	prototxt 中傳過來的 string 長度
isf_len	UINT32	輸入 scale-shift 的長度
isf_mul_1	UINT32	輸入的 multiply
isf_shift_1	UINT32	輸入的 shift
osf_len	UINT32	輸出 scale-shift 的長度
osf_mul_1	UINT32	輸出的 multiply
osf_shift_1	UINT32	輸出的 shift
bin_parm_len	UINT32	為 cust_bin 長度
bin_parm		為 cust_bin data

● 檔案路徑

■ 32bits:

ai_tool/novatek/novaic/toolchain/repo/proprietary/runtime/hdal/vendor/ai2/source_pub/vendor_ai_cpu/vendor_ai_cpu_custnn.c

■ 64bits:

ai_tool/novatek/novaic/toolchain/repo/proprietary/runtime64/hdal/vendor/ai2/source_pub/vendor_ai_cpu/vendor_ai_cpu_custnn.c

[3] 獲取必要資訊的範例代碼如下:

i. 獲取當前私有層需要參數

```

HD_RESULT vendor_ai_cpu_cust(uintptr_t 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)
    */
}

```

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. 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);
#ifdef 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. 整個網路的輸入 size [n, c, h, w]，使用 HD_RESULT
_vendor_ais_get_net_input_info_list() 存放至
VENDOR_AI_NET_INPUT_INFO

```

typedef struct _VENDOR_AI_NET_INPUT_INFO {
    CHAR   name[192];           ///< buffer name, eg: "mylayer.out0"
    UINT32 width;               ///< width
    UINT32 height;              ///< height
    UINT32 channel;              ///< channel
    UINT32 batch;               ///< number of batch
} VENDOR_AI_NET_INPUT_INFO;

```

使用範例如下：

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.

```

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 = %lu\n",net_input_infos[0].width);
    printf("net_input_infos[0].height = %lu\n",net_input_infos[0].height);
    printf("net_input_infos[0].channel = %lu\n",net_input_infos[0].channel);
    printf("net_input_infos[0].batch = %lu\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 = %lu\n",net_input_infos[1].width);
    printf("net_input_infos[1].height = %lu\n",net_input_infos[1].height);
    printf("net_input_infos[1].channel = %lu\n",net_input_infos[1].channel);
    printf("net_input_infos[1].batch = %lu\n",net_input_infos[1].batch);
    . . .
}

```

[4] 在獲取完 input/output info 之後，使用者可以用自定義的結構體；在此以 CUSTNN_POOL_PARM 為例，此結構可以在以下檔案下描述或調整

i. 32bits:

/ai_tool/novatek/novaic/toolchain/repo/proprietary/runtime/hdal/vendor/ai2/source_pub/vendor_ai_cpu/vendor_ai_cpu_custnn_sample.h

ii. 64bits:

/ai_tool/novatek/novaic/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
    #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);
    #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.

```
#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] 擷取完自定義結構參數及 input/output address 後, 便會呼叫運算 API 做處理, 來完成私有層的運算, 如下範例所述:

```
if (layer_type_id == get_hashcode("Pooling")) {
    // 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_hashcode("Eltwise")) {
```

2. 編譯私有層相關庫

在 toolchain 底下執行
make clean; make;



注意事項

用 **make** 時會自動執行以下指令

1. **make custom_op** (產生 libCustomOP.a, libCustomOP.so 到
ai_tool/novatek/novaic/toolchain/closeprefix/lib)
2. **make runtime_ai2_pub** (產生 libvendor_ai2_pub.so for 32-bits 到
ai_tool/novatek/novaic/toolchain/closeprefix/bin/simulator/emu52X)
3. **make runtime64_ai2_pub** (產生 libvendor_ai2_pub.so for 64-bits 到
ai_tool/novatek/novaic/toolchain/closeprefix/bin/simulator/emu33X)

3. simulation tool 底下完成的 vendor_ai_cpu_custnn.c, 可直接複製到對應的 AI2 SDK 路徑下的檔案 code/hdal/vendor/ai2/source_pub/vendor_ai_cpu/vendor_ai_cpu_custnn.c

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

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.

6.1.4 AI2 SDK

1. 修改源碼 `vendor_ai_cpu_custnn.c`，根據 Gen-tool 傳來的訂義參數，撰寫自訂義函式 `vendor_ai_cust()`；其步驟與 Sim-tool 的 步驟 1 需一致
2. 以 `00001_custom_layer_inception_v2` 的 pooling 層為例：在 AI2 sdk code 上的 `vendor_ai_cpu_cust()` 裡頭，前半段會先擷取一些 custom 資訊，其中 `CUSTNN_POOL_PARM` 的 structure 是定義在 `~\hda\vendor\ai2\source_pub\vendor_ai_cpu\vendor_ai_cpu_custnn_sample.h`，使用者可以依據自身需求調整 structure。在取得若干資訊後，使用者便可依據自身需求，來加入運算及處理。在此例中，我們擷取的 pooling 參數及 input/output address 後，便會呼叫運算 API 做處理來完成 pooling 運算，範例如下

```

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);
        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);

```

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.

```
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 建置私有層的流程 (ONNX)

6.2.1 ONNX

1. 準備含有 ONNX 私有層之 ONNX 模型，可按照以下方式生成:

- [1] 根據官方文檔，從 pytorch 生成含有 ONNX 私有層之 ONNX 模型
(<https://github.com/onnx/tutorials/tree/main/PyTorchCustomOperator>)
- [2] 自行建置 ONNX 模型，相較於一般的 ONNX 模型，有兩個地方需要額外添加，第一個是 ModelProto 的 opset_import 欄位需額外添加 custom domain 的 name 和 version 如下:



此範例定義 domain name 為 mydomain，version 為 1，添加方式可以參考以下 python 腳本

```
import onnx
import copy

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

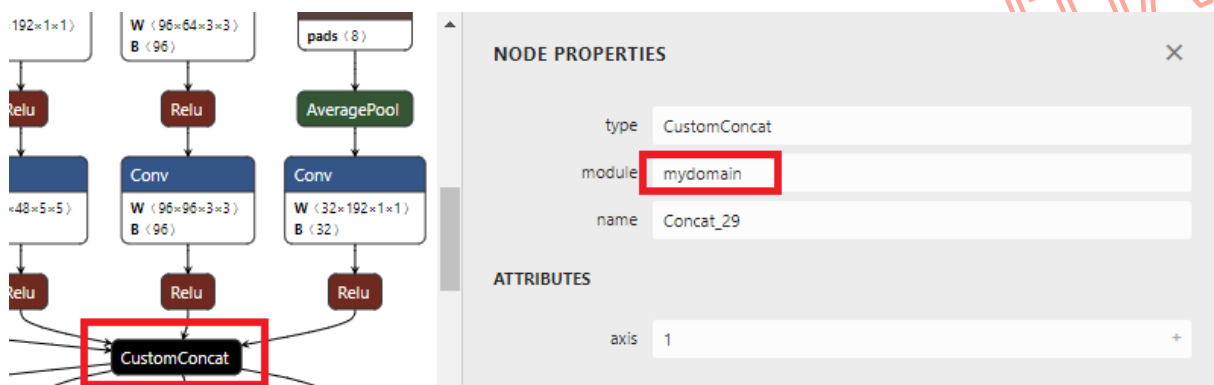
copy_op_set_import = copy.deepcopy(onnx_model.opset_import[0])
copy_op_set_import.domain = "mydomain"
copy_op_set_import.version = 1
onnx_model.opset_import.append(copy_op_set_import)
print(onnx_model.opset_import)

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


若打印 `model.opset_import` 可以看到如下，空字串 `domain` 為 `onnx` 標準的 `opset domain` (空字串或 `ai.onnx` 都被視為 `onnx` 標準 `opset`)

```
nvt04631@oaalnx53:~/onnx_tool$ python3 DumpOnnxDomain.py --input custom_inception_v3.onnx
input: custom_inception_v3.onnx
[domain: ""
 version: 12
 , domain: "mydomain"
 version: 1
]
```

另外，每個 `Cutom OP` 的 `NodeProto` 中需添加 `domain` 欄位同上述的 `custom domain` 的 `name`



添加方式可以參考以下 `python` 腳本

```
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')
```

打印 `NodeProto` 如下

```
nvt04631@oaaalnx53:~/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"
```

**注意事項**

上述準備之含私有層 ONNX 模型仍需使用 onnx2novaonnx_converter 轉成 deploy.onnx 後方可使用 Novaic Tool

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

6.2.2 Gen-tool

1. cust_config.txt 的設置同 caffe 私有層，請參考 6.1.2 前面關於 cust_config.txt 的設置。
2. 為了使用 onnxruntime 取得私有層輸出 shape 及 inference 結果，需在 onnxruntime 上實現私有層並向 onnxruntime 註冊私有層 OP，請根據官方文檔進行建置 (<https://github.com/onnx/tutorials/tree/main/PyTorchCustomOperator>)，打開 ai_tool/novatek/novaic/toolchain/repo/proprietary/onnc-backends/NvtAI/Source_pub/CustomOP，修改此處的 custom_op.h 及 custom_op.cc，私有層 Concat 的範例如下

```
//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; };

    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
```

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 (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. 修改私有層自訂函式: 打開 `ai_tool/novatek/novaic/toolchain/repo/proprietary/onnc-backends/NvtAI/Source_pub/CustomOP/NvtAICustomerUtility.cpp`，以下列出需要修改的函式。

[1] **AttributeProto** 參數:

自訂層的 **AttributeProto** 描述中(如下範例)，所有參數會以合成一個 **string** 的形式傳入，使用者須自行解析自訂義層參數，並將參數存在 **layer_parm**(自訂的 **struct** 形式，如 **NvtAICustomerUtility.h** 中的 **CUSTNN_CONCAT_PROTO_PARM**)

NvtAICustomerUtility.h 檔案位置在

`novatek/novaic/toolchain/repo/proprietary/onnc-backends/NvtAI/
NvtAICustomerUtility.h`

可參考 `parsing_proto_parm()`的實現。

```

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

```

例如在 `parsing_proto_parm()` 調用 `processConcat` 函式來解析上述內容，解析完會存在 `IR (NvtAICustomer& op)` 的 `Public` 變數 (`op.layer_parm.data()`)裡，可以直接取得

```
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 函式:

先前已經在 `onnxruntime` 上實現私有層 `op`，`Reshape` 的時候把此 私有層 `op` 加入自訂義 `custom op domain` 之中(`onnx` 模型上的 `domain`，參照 6.2.1)，然後呼叫 `ReshapeWithCustomOpDomain(custom_op_domain, op)`，可參考 `ReshapeCustomer()` 的實現

```
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 函式:

先前已經在 `onnxruntime` 上實現私有層 `op`，`fp32 inference` 的時候把此 私有層 `op` 加入自訂義 `custom op domain` 之中(`onnx` 模型上的 `domain`，參照 6.2.1)，然

後呼叫 `ComputeWithCustomOpDomain(custom_op_domain, op)`，可參考 `ComputeCustomer()` 的實現

```
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 資訊:

使用者自行計算 `tmp_buffer` 所需要的大小，大小需大於等於 `input feature` 總共所需要的 `buffer` 大小 (多 `input` 的話要相加)，可參考 `getCustomerTempBufferSize()` 的實現。

```
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;
}
```

由於 `getCustomerTempBufferSize` 實現的位置是在 `Calibration` 之後，使用者若想取得真實量化後的結果，可以在這邊取得。

[5] 獲取當前私有層量化的資訊:

若使用者想獲取當前私有層的量化資訊如

- i. 輸入輸出 `size [n, c, h, w]`，可透過以下方式取得

```
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 存在 IR

(NvtAICustomer& op) 的 Public 變數裡，可以直接取得

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

- iii. 整個網路的輸入 size [n, c, h, w]，紀錄在 NvtAICustomer 的 public 變數 `std::vector<TensorInfo> net_input_infos;`

其結構體如下：

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

使用範例如下：

```
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;
    }
    .
    .
}
```

6.2.3 Sim-tool

使用方式同 `caffe` 私有層，請參照 6.1.3。

6.2.4 AI2 SDK

使用方式同 `caffe` 私有層，請參照 6.1.4。

7 DSP ACL 使用方式

NT98530 有加上 DSP 硬件支持，若當 DSP 閒置時其實是可以拿來作執行一些原本需要在 CPU 處理的網路層，一來透過 DSP 強大的運算可以減少網路推裡耗時，二來原本需要透過 CPU 來處理的層改使用 DSP 推裡能降低 CPU loading。

另外，DSP 在支持某些 OP 是透過 ACL 介面在 DSP 上做加速運行，所以在本章節會介紹 DSP ACL 使用方式，目前 DSP 支持 OP 如下

- Softmax
- Dilated Conv

以下說明使用 DSP ACL 需設置的參數以及 gen_config 額外提供的參數

- Compiler:

- cmd 參數：以下幾個參數必設定才能使用 ACL DSP
 - ◆ --chip 530：支援 DSP ACL 的 chip
 - ◆ --use-heteroge 1：開啟 ACL
 - ◆ --backend-order DSP_CPU：指定使用 ACL DSP backend

- 若上述參數有其中一種設置錯誤，執行須使用 DSP ACL 的 OP 會被攔下，類似下方例子的報錯。此時請再次確認參數是否設置正確，或是該 OP 的規格目前尚未支援

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

- 若 chip 設置非 530 且指定 --backend-order，則會看到以下警告〔範例以 DSP_CPU 為例，其對應編號為 1〕。此時請將 --chip 設回 530。

```
Warn: '--backend-order' 1 can only be used with chip 530, set '--backend-order' to 0
```

- 在 gen config 透過參數設定想要運行 DSP 的 op 層：提供以下幾種參數供設定
 - ◆ [tensor_backend/{output_tensor_name}] = {backend_type}欲指定特定 backend 層，請將該層的 output_tensor_name 填入 {output_tensor_name} 欄位，指定的 backend type 填入 {backend_type} 欄位。目前提供 EXT/DSP/CPU 三種選項。EXT 代表 DSP 或 CPU 運行皆可。範例如下：

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

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.

- 若需要仿真，則是需在 `sim config` 透過參數設定仿真時調用的 `backend` 為 `CPU` 或是 `DSP`。目前提供 `DSP/CPU` 兩種選項。默認值為 `CPU`。

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

若未指定該參數，或是提供錯誤參數，會有以下警告並以默認值繼續運行。

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

- 設置完畢後，執行 `compiler` 時有使用 `ACL` 的 `op` 會於以下幾處顯示不同字樣，可確認是否成功使用 `DSP ACL`：執行 `compiler` 會多出 `DLI` 字樣的前綴

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

- `process_bit_info.txt` 的 `eng_type` 會是 `DSP-DLI`

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

- 在 `profile.txt` 中描述層的字串會有 `DLI` 字樣的前綴

```
===== opId=81 fId=59 [DLI-CONV] Conv_317_Y =====
```

- `siminfo.txt` 的 `eng_type` 為 `4`

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

- 執行 `simulator` 時可確認 `job` 的 `engine type` 與最後寫出的 `fixed bin` 檔案名稱是否有包含 `DSP` 字樣

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

8 網路調整建議

8.1 精度調整

- gen_config 調整

- 選擇不同功能輸入要 8/16bit

- [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] = 0

- 建議可以開啟上述的開關，以達到較高精度

- 選擇下一層是 relu 時，是否無條件參考 relu 作為當層定點參考值域

- ### [REF RELU OUTPUT]

- # 0: current reference its layer out

- # 1: current reference next relu out

- [precision/ref_relu_outval_en] = 1

- 當 relu 的 negative_slope = 0 時，建議開啟此開關，若為其他數值，需要測試看看此開關效果

- 選擇是否開啟 weight 8bit 量化壓縮

- # 0: disable quatization

- # 1: enable quatization

- [compression/method/quan_en] = 0

- 開啟此開關，weight 會量化為 8bit，可節省頻寬，但可能也會降低精度，如果 caffemodel 內的 weight 訓練時期為 8bit，這個開關設定應該沒差，如果訓練 weight 為 32bit，可測試此開關是否有影響精度。

- 選擇是否開啟 VLC weight 無損壓縮

- # 0: disable variable length coding

- # 1: enable variable length coding

- [compression/method/vlc_en] = 0

- 開啟此開關，weight 會做無損壓縮，可在大部分網路結構下，節省 model size 與頻寬，

如果模型太大有內存問題可以打開此開關測試。

- 選擇是否開啟平衡各層值域機制

[CROSS LAYER EQUALIZATION]

0: turn off CLE

1: turn on CLE

[precision/cle_en] = 0

如果有 conv / bn / scale / relu / leaky relu / prelu / global_ave pool / innerproduct 連續出現，且沒有 branching，此開關會視值域調整 weight 權重，平均最大值分布，可開啟看看此開關，看精度是否有改善。



注意事項

連續 func 的中間 cos similarity 不具參考性，要看最後面的精度才行

- 全值域 scale factor 機制限制

- Tool 會預設開啟全值域(full range)的機制，使精度提升。

- 參考影像的挑選(validation dataset)

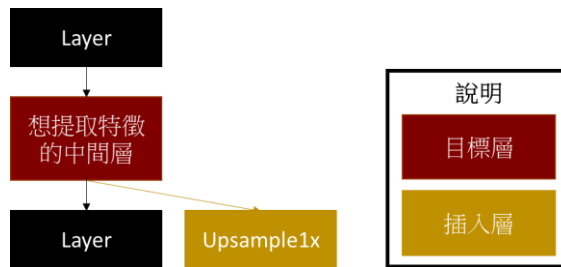
- 準備多張圖片當作參數配置的參考，建議每個類別不要少於 3 張圖，圖片選擇要涵蓋所有訓練目標類別，且目標物清晰。

- Prototxt 調整

- 若想要得到以下層輸出特徵

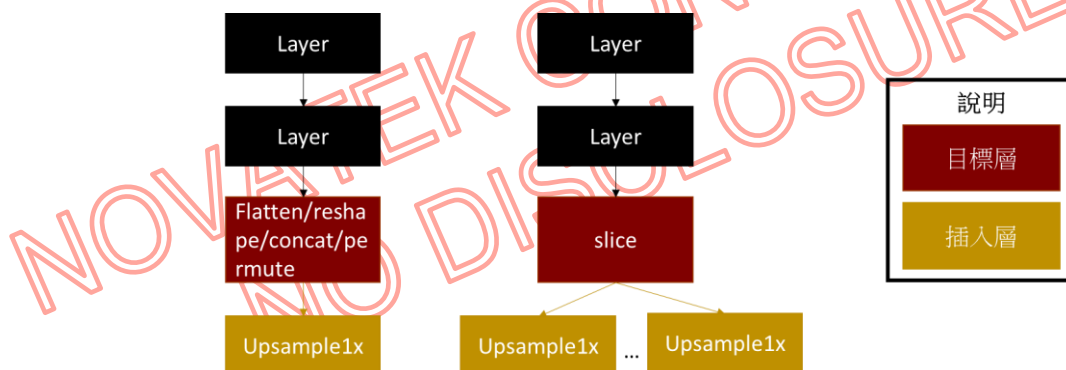
- 中間層

- **描述:** 為了節省 memory，中間層的 output buffer 若不再使用會被覆寫，因此無法提取其輸出特徵。
- **解法:** 由於轉換工具會將無下一層的層視為網路輸出層，並保留其特徵，因此在某中間層後方插入 upsample1x，此層不改變輸入內容，且為網路輸出層，便能提取該中間層特徵。



■ 最後層，且為 `flatten/reshape/slice/concat/permute`

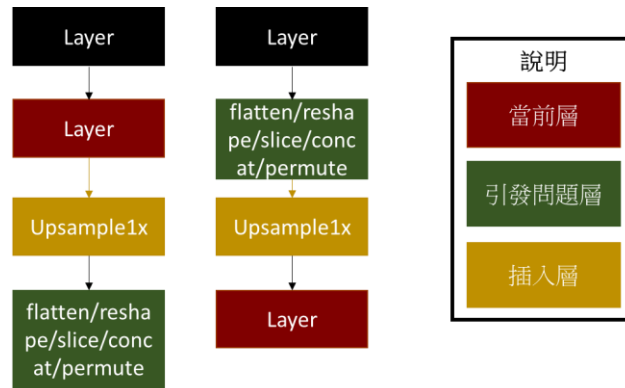
- **描述:** `flatten/reshape/slice/concat/permute` 作為最後一層，在網路轉換時會被移除，使用者無法提取 `flatten/reshape/slice/concat/permute` 的特徵形狀
- **解法:** 可以在 `flatten/reshape/slice/concat/permute` 後方插入 `upsample1x`，使其不改變輸入內容，且為網路輸出層，其形狀會等於 `flatten/reshape/slice/concat/permute` 輸出形狀，如此便能提取這些最後層的特徵。



■ 當前層精度有問題，且符合

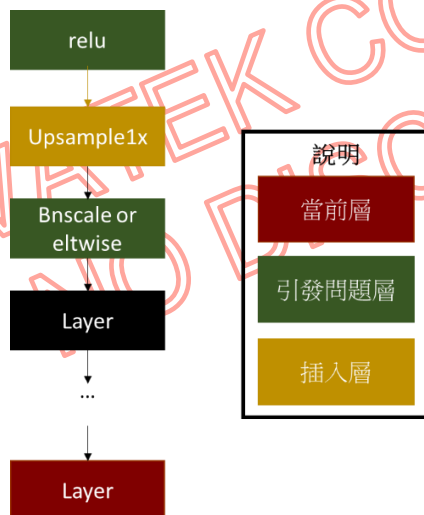
■ 前後連接層有包含 `flatten/reshape/slice/concat/permute` 層

- **可能原因:** `flatten/reshape/slice/concat/permute` 在網路轉換時會被移除，移除後，需特別調整前後層精度相關參數(例如 `scale factor` 或位寬)
- **解法:** 由於 `upsample1x` 不改變輸入內容，且有精度調整機制，與 `flatten/reshape/slice/concat/permute` 連接的結構已在多個網路驗證過，因此可在目前層和 `flatten/reshape/slice/concat/permute` 中間插入 `upsample1x`。



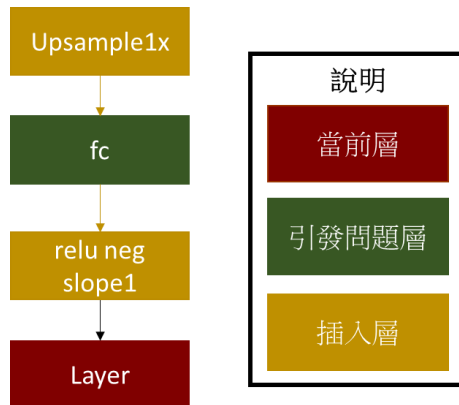
■ 前層有包含 relu+bnscale/relu+eltwise 結構

- 可能原因: relu+bnscale 或 relu+eltwise 會被轉換工具融合
- 解法: upsample1x 不改變輸入內容，且在轉換工具內不會與 relu 或 bnscale/eltwise 融合，將 upsample1x 插入於 relu 和 bnscale/eltwise 之間可以斷掉融合



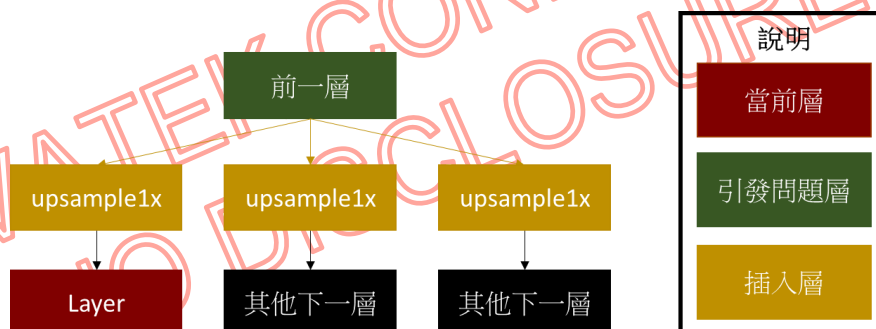
■ 前後連接層有包含 fc 層

- 可能原因: fc 層有以下限制
 - (1) 輸入/輸出點數對齊 4
 - (2) 輸入/輸出位址對齊 4
- 解法: upsample1x 與 relu negative slope: 1 不改變輸入內容，由於 upsample1x + fc 或 fc + relu negative slope: 1，有處理 fc 位址與點數 4 對齊問題，可以在 fc 前後插入 upsample1x 與 relu。



■ 有連結多個 branch

- **可能原因:** 不同層有不同限制，前一層會根據所有 branch 層的限制，調整精度相關參數，但部分後面層為了配合其他層限制，精度會受影響。
- **解法:** 由於 **upsample1x** 不改變輸入內容，且有調整限制機制，若在前層與後面層之間各插入一個 **upsample1x**，可讓 **upsample1x** 分別針對不同限制做調整，每個後面層也就互不影響了。



注意事項

1. 若 **upsample1x** 加入後，在轉換工具的 **caffe** 階段遇到尺寸或維度不符合錯誤，原因為 **upsample** 只能接受 4 維輸入，此時可以

- (1) 在 **upsample1x** 前方插入 **reshape**，將輸入特徵轉為四個維度
- (2) 不插入 **upsample1x**，而是取代為 **relu negative slope: 1**

2. 以上改動請在 **prototxt** 上進行

8.2 自動化網路修正

上一章提到有些情況會需要修改到 **Prototxt** 來達到目地，但其實工具對於一些情況已經有自動修正機制，以下表格整理出添加方式與添加原因

添加原因	添加方式
支援存在同名 bottom 的 layer	通過加 upsample 將同名的 bottom 改一個名字再接回原來的 layer
支援一個 top 接多個 concat	該層 top 可以直接接第一個 concat 層，接第二個 concat 層時需要先加 upsample 再接第二個 concat 層
1. 支援 reshape , slice , reverse 接 concat(axis=1) 2. 支援非 conv , upsample , deconv , relu , tanh , sigmoid , eltwise , bn/scale 層接 concat (axis!=1) 3. 支援 priorbox 接 concat (axis=2)	在這些層後加 upsample 再接 concat 層
1. 支援 slice 接 reshape , slice , concat 2. 支援 slice(axis!=1) 接非 conv , upsample , deconv , relu , tanh , sigmoid , eltwise , bn/scale 層	在 slice 層後加 upsample 層再接這些層
支援兩個 bottom 維度不同的 scale layer	將該 scale layer 變成以下組合層 reshape + upsample*n + crop + eltwise(mul)
支援 axpy layer	將 axpy layer 變成以下組合層 reshape + upsample*n + crop + eltwise(mul) + eltwise(add)
支援 split layer	split layer 作用就是將 bottom 拷貝給各個 top ，這裡將 split 換成多個 upsample
支援第一層不是 conv 或 upsample 時在 gen_config 設 norm scale	當 gen_config 中開啟 norm scale 且 norm scale 不為 1，同時第一層不是 conv 或 upsample ，在第一層前加 upsample
解決 pooling (HW) 在以下情況下會有尺寸不對的問題 [1]. 當 ceil_mode = true ，且 W_{out} 符合以下條件	在有問題的 pool 後加 upsample

$W_{out} = \text{ceil}\left(\frac{(W_{in} + 2 * Pad - Kernel)}{Stride}\right) + 1$ <p>[2]. 當 W_{out} 符合以下條件</p> $W_{out} \geq \frac{(W_{in} + Pad)}{Stride} + 1$	
支援 eltwise, fc(+relu), sigmoid 接 concat 時調整精度	當 eltwise, fc(+relu), sigmoid 接到 concat 層時，在這些層後面加 upsample 再接 concat
解決 (HW) FC + Conat 因為 HW align 4, 如果 (FC / FC / FC)(並聯) → Concat 輸出到大 Buffer 並非都是 4 倍數, 則會輸出錯誤的問題	如果 (FC / FC / FC)(並聯) → Concat, 在不滿足 batch*channel*height*width align4 的 fc 后加 upsample 層

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

8.3 網絡效率

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

	<p>Kernel (w, h) = (1, 1)</p> <p>Pad (w, h) = (0, 0)</p> <p>Stride = 1</p>
<p>Kernel 1x7 (Kw = 1, Kh = 7)</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, 7)</p> <p>Pad (w, h) = (0, 3)</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, 7)</p> <p>Pad (w, h) = (0, 3)</p> <p>Stride = 1</p>
<p>Kernel 7x1 (Kw = 7, Kh = 1)</p>	<p>Condition:</p>
	<p>Out_Width = multiple of 2</p> <p>In_Channel > 32 (should be a multiple of 32)</p> <p>If Stride = 2 and $\frac{\text{In_Width} * \text{In_Height} * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 32 / 5)} > 4$</p> <p>If Stride != 2 and $\frac{\text{In_Width} * \text{In_Height} * \text{In_Batch}}{\text{Stride} * \text{CEIL}(\text{Kw} * \text{Kh} * 32 / 5)} > 2$</p> <p>In_Channel = 32, it's always efficient</p> <p>In_Channel < 32, 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) = (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>
Kernel 1x5 (Kw = 1, Kh = 5)	<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 ($A (1 \times 1 \times N \times K) * B (1 \times 1 \times K \times M)$)

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

2. 抓資料效率

- DRAM starting address
 - 4-word aligned

8.4 量化注入

對於各層特徵以及權重，若想自行調整值域並重新分配 **bit** 設定，可調整以下與量化注入相關 **config** 設定

1. 在 **gen_config** 加入以下設置

[1] 設置量化注入開關

```
#[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] 設置量化注入路徑 (支持相對於 **--config-dir** 設置的路徑之下，或是絕對路徑)

```
## weight_quantization
[path/cust_quan/path] = ..\nvtai_tool\config\00001_inception_v2_cnn25
```

[3] 設置量化模式

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



注意事項

量化模式開關是用於控制介面檔內值域的表達方式，目前 **zeropoint+scale** 正在開發中，將在未來支援

[4] 生成模型與量化參數 **json** 檔:

獲取量化資訊: 將 **[precision/cust_quan/mode]** 設定為 **1**，並執行 **gentool**，會在 **gen_config** 中指定的量化注入路徑下 (**[path/cust_quan/path]**) 生成初始的量化資訊 **quan_info_maxmin.json** (當 **[precision/cust_quan/val_mode] = 0**) 或是 **quan_info_bitinfo.json** (當 **[precision/cust_quan/val_mode] = 2**)

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

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.

注意事項

當 [precision/cust_quan/mode] 設定為 1 時只會生成 quan_info_maxmin.json 文檔，並不會生成 nvt_model.bin

2. 輸出檔案介紹

[1] 量化資訊 quan_info_maxmin.json:

打印模型中各層的特徵以及權重的量化資訊。該檔案會以各層輸出的 tensor_name 作為名稱打印資訊，打印內容包括：各層 onnx op 名稱、各層類型 (layer_type)、各層輸出特徵最大最小值 (output0_max/ output0_min) 若該層為 convolution 或 innerproduct，將會打印該層權重的最大最小值 (input1_max/ input1_min/ input2_max/ input2_min)，其中 input1 代表 Weight, input2 代表 Bias

```
{
  "Convolution_conv_1_Y": {
    "layer_type": "Conv",
    "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": "BatchNormalization",
    "output0_max": 13.681736946105957,
    "output0_min": -8.931833267211914
  },
  "ReLU_relu_1_Y": {
    "layer_type": "ReLU",
    "output0_max": 13.681736946105957,
    "output0_min": 0.0
  },
  "Pooling_pool_1_Y": {
    "layer_type": "MaxPool",
    "output0_max": 13.681736946105957,
    "output0_min": 0.0
  },
  "InnerProduct_fc1_Gemm_Y": {
    "layer_type": "Gemm",
    "input1_max": 1.7419883012771606,
    "input1_min": -0.6007164120674133,
    "input2_max": 0.9874842762947083,
    "input2_min": -0.8291921019554138,
    "output0_max": 12.809922218322754,
    "output0_min": -7.271744728088379
  }
}
```

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.

注意事項

目前 **weight** 的量化注入只支持 **conv** 與 **fc** 層 (在 **onnx** 模型中圍 **gemm** 層)

[2] 量化資訊 **quan_info_bitinfo.json**:

打印模型中各層的特徵以及權重的 **bit** 資訊。其中 **"conv1"** 為網絡層名稱，**"layer_type"** 為網絡層類型，對齊 **onnx** 的網絡層類型。**"weight_bitdepth"**，**"weight_sign_bit_num"**，**"weight_frac_bit_num"**分別為這一層 **weight** 的 **bit** 位深，符號位，量化浮點位，**"bias_bitdepth"**，**"bias_sign_bit_num"**，**"bias_frac_bit_num"**分別為這一層 **bias** 的 **bit** 位深，符號位，量化浮點位，**"output0_bitdepth"**，**"output0_sign_bit_num"**，**"output0_frac_bit_num"**分別為這一層第一路輸出的 **bit** 位深，符號位，量化浮點位。

```
{
  "conv1": {
    "layer_type": "Conv",
    "weight_bitdepth": 8,
    "weight_sign_bit_num": 1,
    "weight_frac_bit_num": 16,
    "bias_bitdepth": 12,
    "bias_sign_bit_num": 1,
    "bias_frac_bit_num": 12,
    "output0_bitdepth": 8,
    "output0_sign_bit_num": 1,
    "output0_frac_bit_num": 3
  },
}
```

3. 修改量化資訊並注入到原本的網路模型中

[1] 修改 **json** 檔:

根據需求修改檔案中各層的特徵以及權重的最大最小值，若是注入 **bit_info**，則是修改各層的 **bit** 數

注意事項

如果當層為 **concat** 附近層，**gentool** 會自動調整參考層到 **concat** 輸出，因此調整這些層的輸出，並不影響精度調整

- [2] 開啟量化注入，並執行 **gentool** 將 **[precision/cust_quan/mode]** 設定為 2，並執行 **gentool**，此外，是否開關全值域可以用以下參數設置：

輸出值域參考模式 (非全值域模式 或 全值域模式)：非全值域模式時，比例因子為 1.0；全值域模式，比例因子會由輸出資料的最大值計算

0: Enable non-full range

1: Enable full range (default)

[precision/ref_outval_en] = 0

4. 輸出每層的最大最小值

- [1] 若平時不使用量化注入時 **[precision/cust_quan/mode = 0]**，或是量化注入後 **[precision/cust_quan/mode = 2]** 想知道每層的最大輸出最大最小值時可以參考以下參數設置：

轉儲張量資訊，若開啟此選項，每融合層的最大與最小值會轉存到 **layer_maxmin.json**，並存在 **nvtai_tool/output/{MODEL}/gentool/** 下，默認不開啟

0: disable dump tensor max min

1: enable dump tensor max min

[precision/cust_quan/dump_tensor_en] = 0

9 工具除錯

9.1 LOG 打印規範

當工具運行時，會打印出目前的運行狀況，以下整理出打印的規範，可以透過此表了解當問題發生時，是發生在哪個模組，縮小問題排查的範圍

9.1.1 Compiler

Name	Method	Class	Description
nvt	gen	cfg	compiler 初期配置與解析
nvt	gen	parser	compiler onnc IR 解析與轉換
nvt	gen	legal	compiler onnc IR 運算轉換成 nvt IR 與算圖優化
nvt	gen	cali	compiler 執行 onnc float 推論結果 + 量化參數優化
nvt	gen	binfo	compiler 位元資訊處理
nvt	gen	wpart	compiler OP 轉換 work domain 供排程使用
nvt	gen	schd	compiler OP 排程處理及優化
nvt	gen	malloc	compiler Tensor 記憶體配置及優化
nvt	gen	lib-weight	compiler 產生 target 硬件 weight 參數
nvt	gen	lib-stripe	compiler 產生 target 硬件 stripe 參數
nvt	gen	lib-llcmd	compiler 產生 target 硬件 link-list command 參數
nvt	gen	lib-perf	compiler 估算 target 硬件效能頻寬與時間
nvt	gen	cemit	compiler 產生 loadable 檔案

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.

9.1.2 Simulator

Name	Method	Class	Description
nvt	sim	cfg	simulator 初期配置與解析
nvt	sim	target	simulator 進行 sim info 解析與 emu 運行準備
nvt	sim	lib-nn	simulator 模擬 target 硬件運算
nvt	sim	lib-perf	simulator 估算 target 硬件效能頻寬與時間
nvt	sim	emu	simulator 運行 emu 產生 target fixed 推論結果
nvt	sim	golden	simulator 運行 onnc fp32 產生 golden float 推論結果
nvt	sim	diff	simulator 產生 target fixed 與 golden float 結果, 並比對產生 diff report

NOVATEK CONFIDENTIAL
NO DISCLOSURE!

10 Revision History

Revision	Date	Author	Changes
0.0.1	2021/06/25	Edward Tseng	First formal version
0.0.2	2021/06/29	Edward Tseng	4.2 modify the description of test_image
0.0.3	2021/07/05	Edward Tseng	Modify the path of folder struc. By new version of tool.
0.0.4	2021/07/21	Edward Tseng	6.1 add statement for custom layer in caffe
0.0.5	2021/07/23	Edward Tseng	3.1 modify the gen_config setting
0.0.6	2021/08/03	Edward Tseng	3.1 modify the gen_config setting 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 add --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 Add custom-layer description
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	2.1 Add the supported layer of onnx

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.

		Jack Wei	2.2 Add the test Pytorch network 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