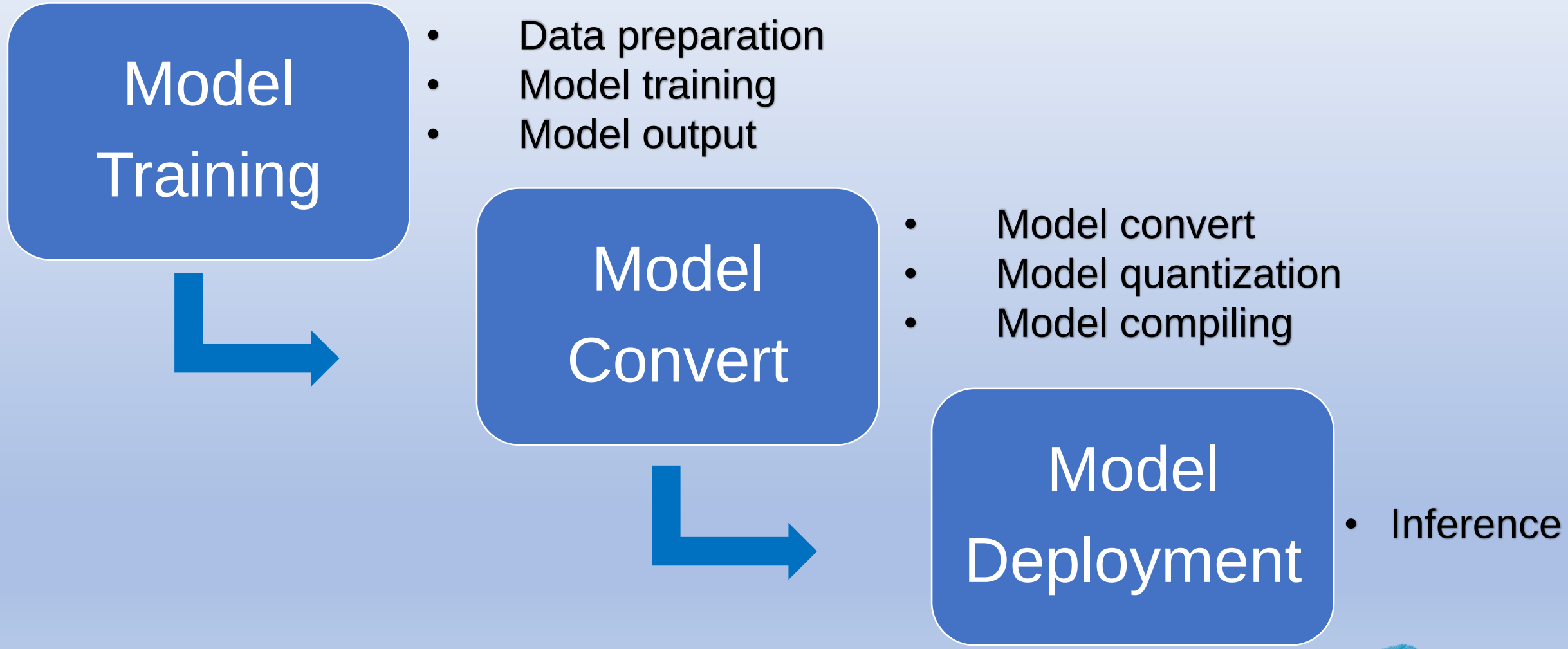
Edge Computing AI Application

OpenMMLab Kneron Edition

Presenter : Andy Hseih

Kneron Edge Computing AI Application Flow



Kneron Edge Computing AI Application Flow

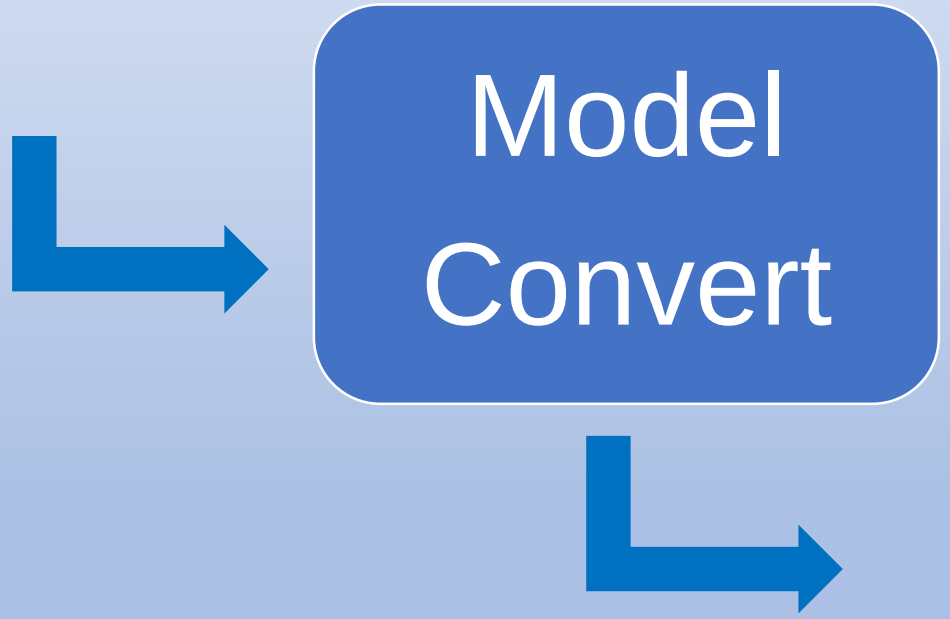
Model Training

- Data preparation
- Model training
- Model output

On Self Computer !

Kneron Edge Computing AI Application Flow

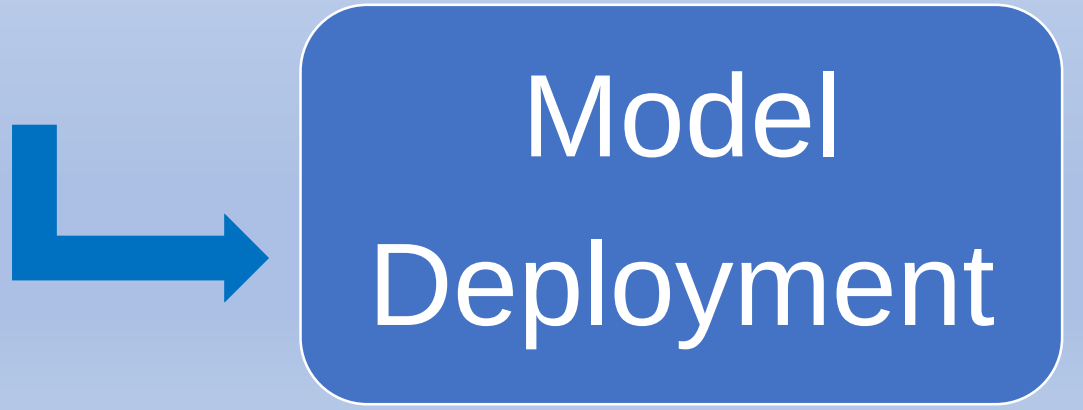
On Self Computer & Kneron ToolChain Docker!



- Model convert
- Model quantization
- Model compiling

Kneron Edge Computing AI Application Flow

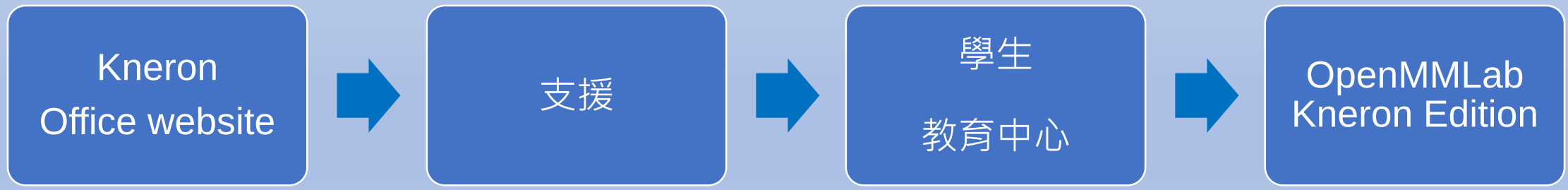
On Self Computer & Use dongle KL720!



- Inference

OpenMMLab Kneron Edition

- kneron-mmdetection (mmDetection-based)
- kneron-mmsegmentation (mmSegmentation-based)
- kneron-mmtracking (mmTracking-based)
- kneron-mmpose (mmPose-based)
- kneron-mmclassification (mmClassification-based)

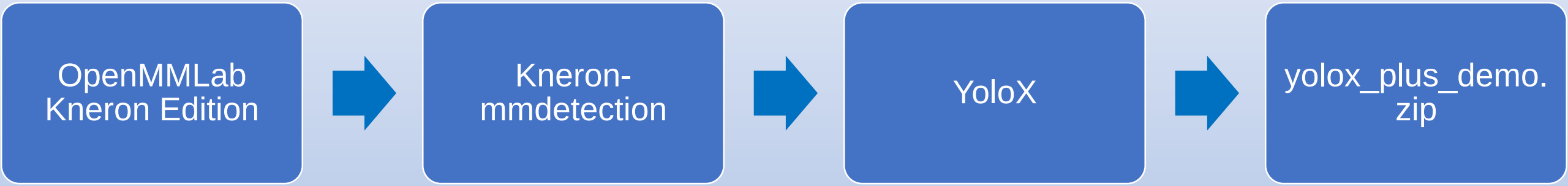


Link：https://www.kneron.com/tw/support/education-center/

Kneron office website: https://www.kneron.com/tw/

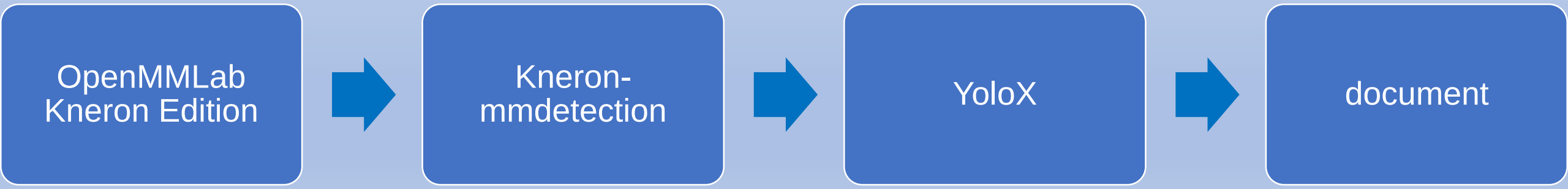
kneron-mmdetection (mmDetection-based)

- Kneron PLUS demo - YoloX



Link : download link

- YoloX step by step



Link : https://github.com/kneron/kneron-mmdetection/blob/main/docs_kneron/yolox_step_by_step.md

Model Training – YOLOX (YOLOX-s)

- Data preparation
- Model training
- Model output

On Self Computer !

Model Training – YOLOX

- **Environment**

(Python 3.6+、PyTorch 1.3+、CUDA 9.2+、mmcv-full)

- **kneron-mmdetection**

git clone https://github.com/kneron/kneron-mmdetection

pip install -r requirements/build.txt

pip install -v -e . # or "python setup.py develop"

pip install onnx onnxoptimizer onnx-simplifier

Pytorch versions download link : https://pytorch.org/get-started/previous-versions

mmcv-full install guide : https://github.com/open-mmlab/mmcv

Model Training – YOLOX (Data preparation)

- Data preparation

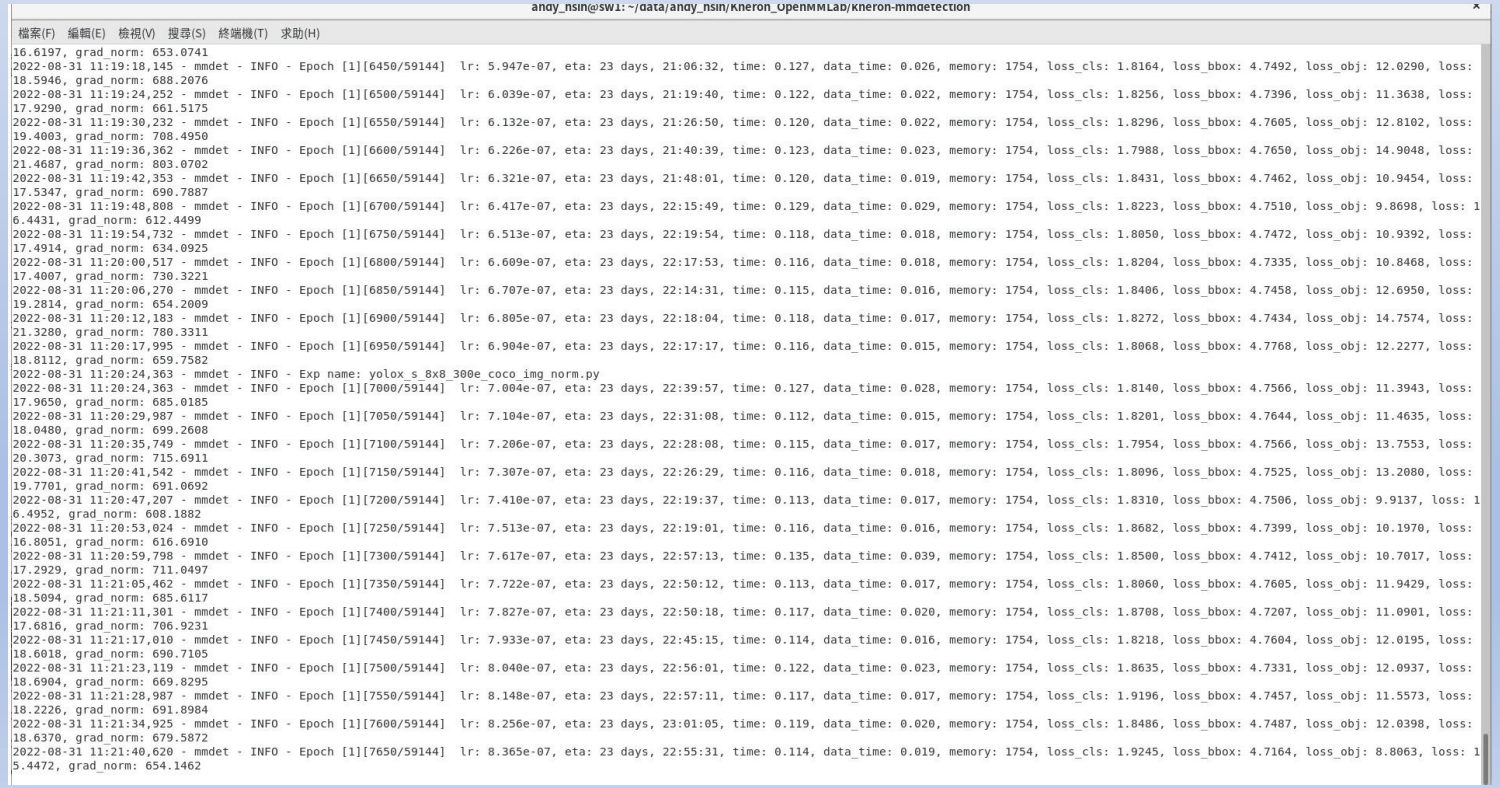
```
mmdetection
├── mmdet
├── tools
├── configs
├── data (this folder should be made beforehand)
│   ├── coco (symlink)
│   │   ├── annotations
│   │   ├── train2017
│   │   ├── val2017
│   │   ├── test2017
...
```

COCO Dataset link： https://cocodataset.org/#download

Model Training – YOLOX (Model training)

- Model training

```
python tools/train.py configs/yolox/yolox_s_8x8_300e_coco_img_norm.py
```



Model Training – YOLOX (Model output)

- Model output
  (work_dirs/yolox_s_8x8_300e_coco_img_norm/latest.pth)

- (Optional) Test trained pytorch model

```
python tools/test_kneron.py \
    configs/yolox/yolox_s_8x8_300e_coco_img_norm.py \
    work_dirs/latest.pth \
    --eval bbox \
    --out-kneron output.json
```

Model Convert – YOLOX

- Model convert
- Model quantization
- Model compiling

On Self Computer & Kneron ToolChain Docker!

Model Convert – .onnx (Model convert)

- Model convert (.pth -> .onnx)

```
python tools/deployment/pytorch2onnx_kneron.py \
    configs/yolox/yolox_s_8x8_300e_coco_img_norm.py \
    work_dirs/yolox_s_8x8_300e_coco_img_norm/latest.pth \
    --output-file work_dirs/latest.onnx \
    --skip-postprocess \
    --shape 640 640
```

Model Convert – .onnx (Model convert)

- (Optional) Test exported onnx model

```
python tools/test_kneron.py \
    configs/yolox/yolox_s_8x8_300e_coco_img_norm.py \
    work_dirs/latest.onnx \
    --eval bbox \
    --out-kneron output.json
```

Model Convert – .onnx (Model convert)

- Model optimizer (.onnx -> .onnx) -> Kneron Toolchain Docker

➢ Python API Work Flow

import ktc

```
onnx_path = '/data1/latest.onnx'
m = onnx.load(onnx_path)
m = ktc.onnx_optimizer.onnx2onnx_flow(m)
onnx.save(m,'latest.opt.onnx')
```

Kneron Toolchain Docker guide link : https://doc.kneron.com/docs/#toolchain/manual/#1-installation

Model Convert – .bie (Model quantization)

- Configure and load data necessary for ktc, and check onnx

```
# npu (only) performance simulation
km = ktc.ModelConfig(20008, "0001", "720", onnx_model=m)
eval_result = km.evaluate()
print("\nNpu performance evaluation result:\n" + str(eval_result))
```

- Model quantization

```
# fixed-point analysis
bie_model_path = km.analysis({"input": img_list})
print("\nFixed-point analysis done. Saved bie model to '" + str(bie_model_path) + "'")
```

Model Convert – .nef (Model compiling)

- Model compiling

```
# compile
nef_model_path = ktc.compile([km])
print("\nCompile done. Saved Nef file to '" + str(nef_model_path) + "'")
```

The .nef model will generate at /data1/batch_compile/models_720.nef

Model Deployment – KL720

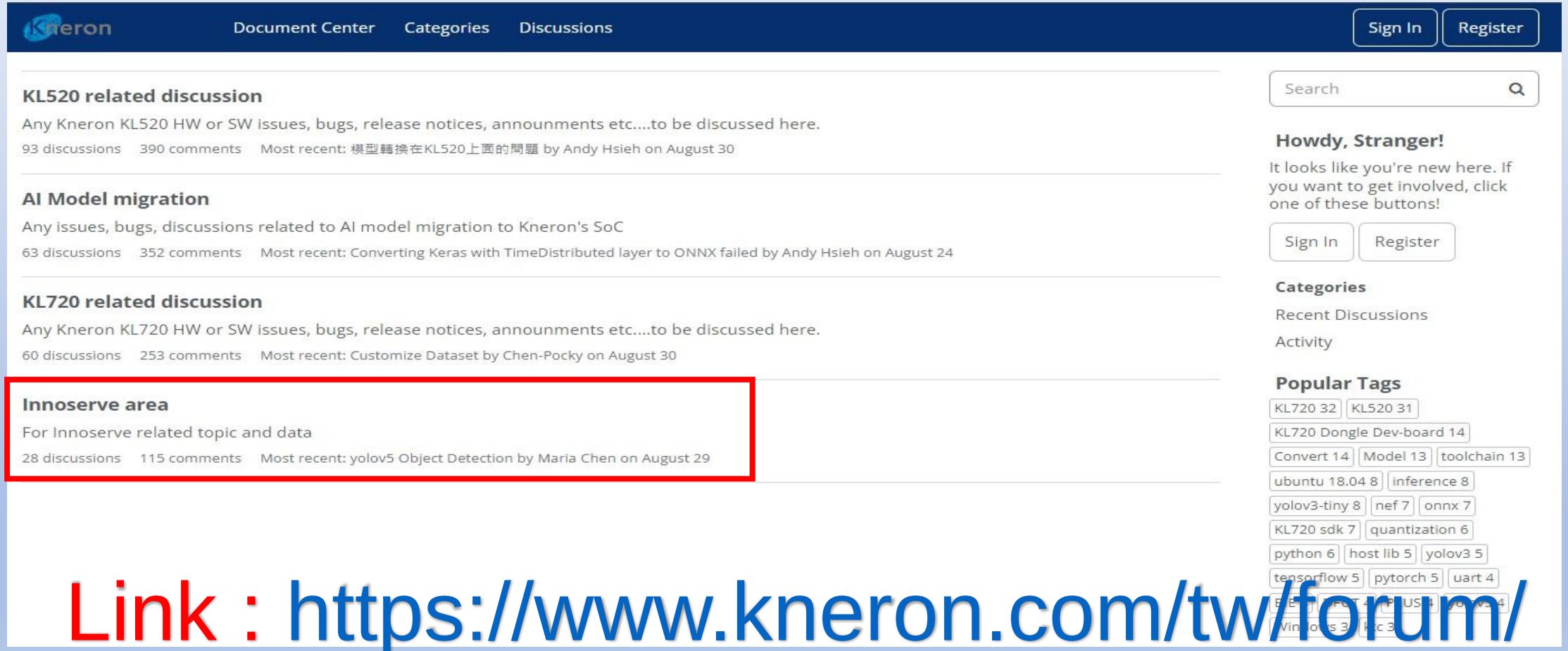
- yolox_plus_demo

```
python KL720DemoGenericInferenceYoloX_BypassHwPreProc.py -img ./000000000536.jpg -nef ./example_yolox_720.nef
```

```
(kneronmmlab) E:\kneron_plus\OpenMMLab_Kneron_Demo\yolox_plus_demo>python KL720DemoGenericInferenceYoloX_BypassHwPreProc.py
[Connect Device]
 - Success
[Set Device Timeout]
 - Success
[Upload Model]
 - Success
======== NEF Info ========

Toolchain ver= kneron/toolchain:v0.16.0

Schema ver   = v0.9.1

==========================
[Read Image]
 - Success
[Starting Inference Work]
 - Starting inference loop 1 times
 - .
[Retrieve Inference Node Output ]
 - Success
[Output Result Image]
 - Output bounding boxes on 'output_000000000536.jpg'
(291.7100891113281,135.5036407470703,444.81996459960936,331.6263565063477)
(86.29214324951171,75.08173713684083,178.1478744506836,331.3882736206055)
(168.07337951660156,81.79956779479981,274.36656799316404,331.9453491210938)

(kneronmmlab) E:\kneron_plus\OpenMMLab_Kneron_Demo\yolox_plus_demo>
```

On Self Computer & Use dongle KL720!

Kneron Forum



Link : https://www.kneron.com/tw/forum/

Kneron

Thank you for listening.