

QAI AppBuilder - WoS

Accelerate Deployment of AI Models for WoS

80-94755-1 Rev. AA

Revision History

Revision	Date	Description
AA	Oct 2025	Initial Release

Contents

1 Overview	8
1.1 LLM	9
1.2 CV / LVM	13
2 Guide	18
3 Env Setup	20
4 Samples	22
5 Tools	28
5.1 LangFlow	30
5.2 QAI Launcher	33
5.3 Pixi (Python)	36
6 Models	38
7 GUI	43
8 Key Points	45
9 Resource	47

Agenda

■ Overview

- LLM
- CV / LVM

■ Guide

- Tutorial
- User Guide

■ Env Setup

- X64 Python
- ARM64 Python
- C++

■ Samples

- Python
- C++
- WebUI / FletUI Apps
- GenieAPIService

■ Tools

- LangFlow
- QAI Launcher
- Pixi (Python)

■ Models

- Compile Model
- Convert DLC to BIN
- Model Resource

■ GUI

- Python (WebUI, FletUI, PyQt)
- JS (Electron)
- C++ (Qt, MFC...)
- C# (WinForms, WPF...)

■ Resource

- Community
- Github

Introduction

QAI AppBuilder is a powerful toolkit designed for Snapdragon® AI PCs. It offers a set of optimized APIs to simplify the deployment of AI models on local NPUs, enabling high-performance inference and accelerating the development of AI applications. This article focuses on features related to the **Windows platform**, though QAI AppBuilder already supports **multi-platform deployment**, including **Windows, Android and Linux**, offering developers broad compatibility.

Running AI models locally brings several key advantages:

- Low latency and high responsiveness
- Enhanced data privacy and content security
- No cloud dependency and completely free

These benefits make it ideal for use cases such as chatbots, document summarization, writing assistants, and image enhancement.

QAI AppBuilder supports a wide range of models on local NPUs, including:

- Classic computer vision models
- Text-to-image models (e.g., Stable Diffusion)
- Popular large language models (LLMs) like Qwen, Llama

For LLMs, we offer flexible invocation methods to ease integration into various applications.

QAI AppBuilder significantly streamlines the deployment process. Developers only need to learn a few simple APIs for model loading and inference, allowing them to focus on designing and building their applications without getting bogged down in deployment complexities.

Our **Model Hub** provides hundreds of models optimized for Snapdragon AI PCs, ready for direct download and use. By combining QAI AppBuilder with these plug-and-play models, even AI development beginners can quickly deploy models on Snapdragon AI PC NPUs using **Python, C++, or LangFlow**. Local AI inference can be achieved in just a few hours.

QAI AppBuilder



For New AI Application Developer

QAI AppBuilder

Simple: Ready to use in ~2 hours, no AI experience needed

Flexible: Supports Python, C++, and low-code tools like LangFlow

Fast: Runs locally on NPU for ultra-low latency

Secure: All data stays on-device, privacy guaranteed

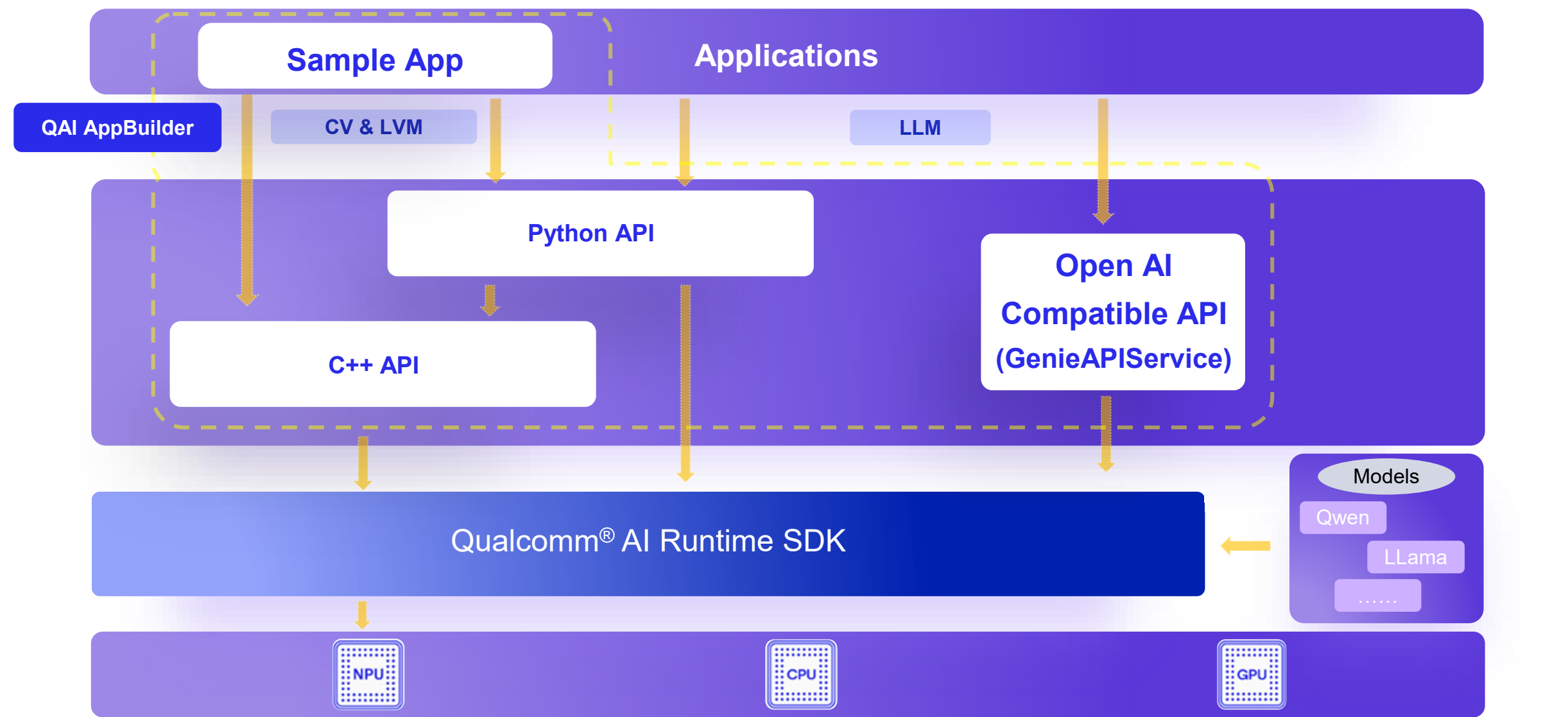
Free: No cloud cost, fully local and cost-effective

Compatible: Works with OpenAI-style APIs for easy integration

* Everything has been posted on public github.

<https://github.com/quic/ai-engine-direct-helper>

Architecture



<https://github.com/quic/ai-engine-direct-helper>

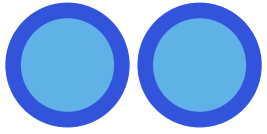


Section 1

Overview

- LLM
- CV / LVM

1.1 LLM	9
1.2 CV / LVM	13



Section 1.1

LLM

Deploying LLMs Models with QAI AppBuilder

To meet the growing demand for developing large language model (LLM) applications using Python, we provide a streamlined Python API(**GenieContext**). With just a few lines of code, developers can run LLMs locally on the NPU of a Snapdragon AI PC, enabling efficient and high-performance inference. This is one way to deploy LLMs using QAI AppBuilder.

Given that the mainstream approach to building LLM applications today is through OpenAI-compatible APIs, we've also implemented this interface(**GenieAPIService**). Developers can interact with locally running LLMs on the NPU using familiar methods. This approach supports flexible configuration and allows seamless switching between different LLMs (such as Qwen, Llama, etc.), making it adaptable to a wide range of application scenarios. This represents another way to deploy LLMs using QAI AppBuilder.

Many third-party applications that support the OpenAI API can seamlessly switch to local NPU-based models by simply updating the API endpoint—no code changes required, resulting in extremely low migration costs. Additionally, we provide Python and C++ client examples to help developers get started quickly and integrate model capabilities into their own applications.

Python API for LLM (GenieContext)

- Qualcomm **Genie** is a library created to simplify the deployment of Gen AI(LLM) pipelines.
- Encapsulate the Genie library to Python API.
 - ✓ **GenieContext** in QAI AppBuilder.
 - ✓ Easy to use with 10 lines Python code.

```
from qai_appbuilder import (GenieContext)
```

```
def response(text):
```

```
    # Print model generated text.  
    print(text, end="", flush=True)  
    return True
```

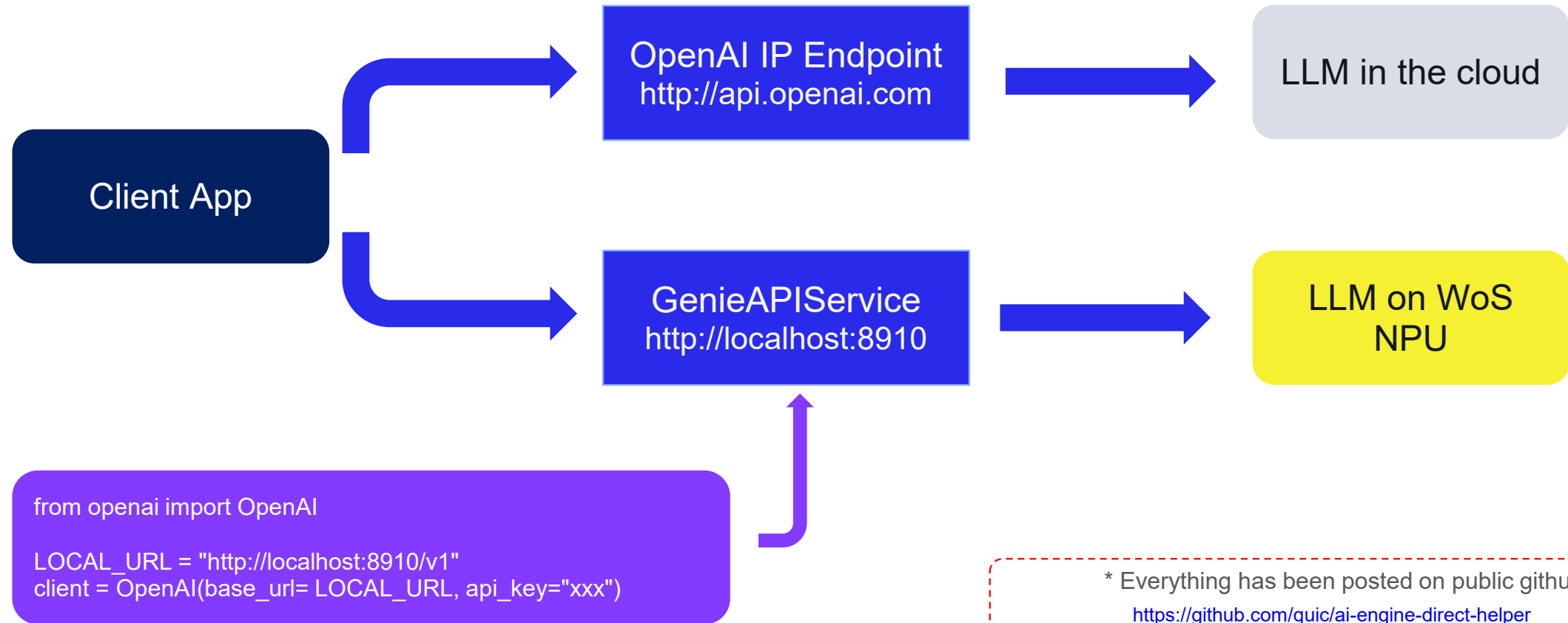
```
# Initialize Llama 3 model.
```

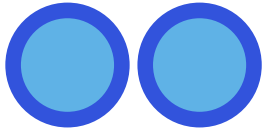
```
config = "models\\Llama3\\config.json"  
dialog = GenieContext(config)
```

```
# Ask question.
```

```
prompt = "How to fish?"  
dialog.Query(prompt, response)
```

GenieAPIService for LLM (OpenAI Compatible API Service)





Section 1.2

CV / LVM

Deploying Vision Models with QAI AppBuilder

So, how can developers run CV(computer vision) models or LVM(large vision model) models like text-to-image (e.g., Stable Diffusion) on the NPU of a Snapdragon AI PC?

The deployment process typically involves three core steps: model loading, inference, and resource release. QAI AppBuilder offers three concise APIs corresponding to these steps, enabling developers to quickly integrate and deploy various vision models.

In this article, we showcase an example of running an image restoration model on the NPU using QAI AppBuilder. It is a CV mode. This model performs image super-resolution, and the core logic requires only about 10 lines of code—making deployment simple and efficient.

For LVM models like Stable Diffusion, the usage pattern is largely similar, though it typically involves multiple models working together, making the workflow slightly more complex. We provide the complete local inference code for Stable Diffusion on GitHub, which developers can reference for fast and effective deployment.

QAI AppBuilder for CV models

```
from qai_appbuilder import (QNNContext, QNNConfig, ...)
from PIL import Image

class RealESRGan(QNNContext):
    def Inference(self, input_data):
        input_datas = [input_data]
        output_data = super().Inference(input_datas)
        return output_data

image = Image.open("flower.jpg")
image = preprocess(image)

QNNConfig.Config(qnn_lib_path=qnn_dir, ...)

realesrgan = RealESRGan("realesrgan.bin")

new_image = realesrgan.Inference([image])

del(realesrgan)

new_image = postprocess(new_image)
```



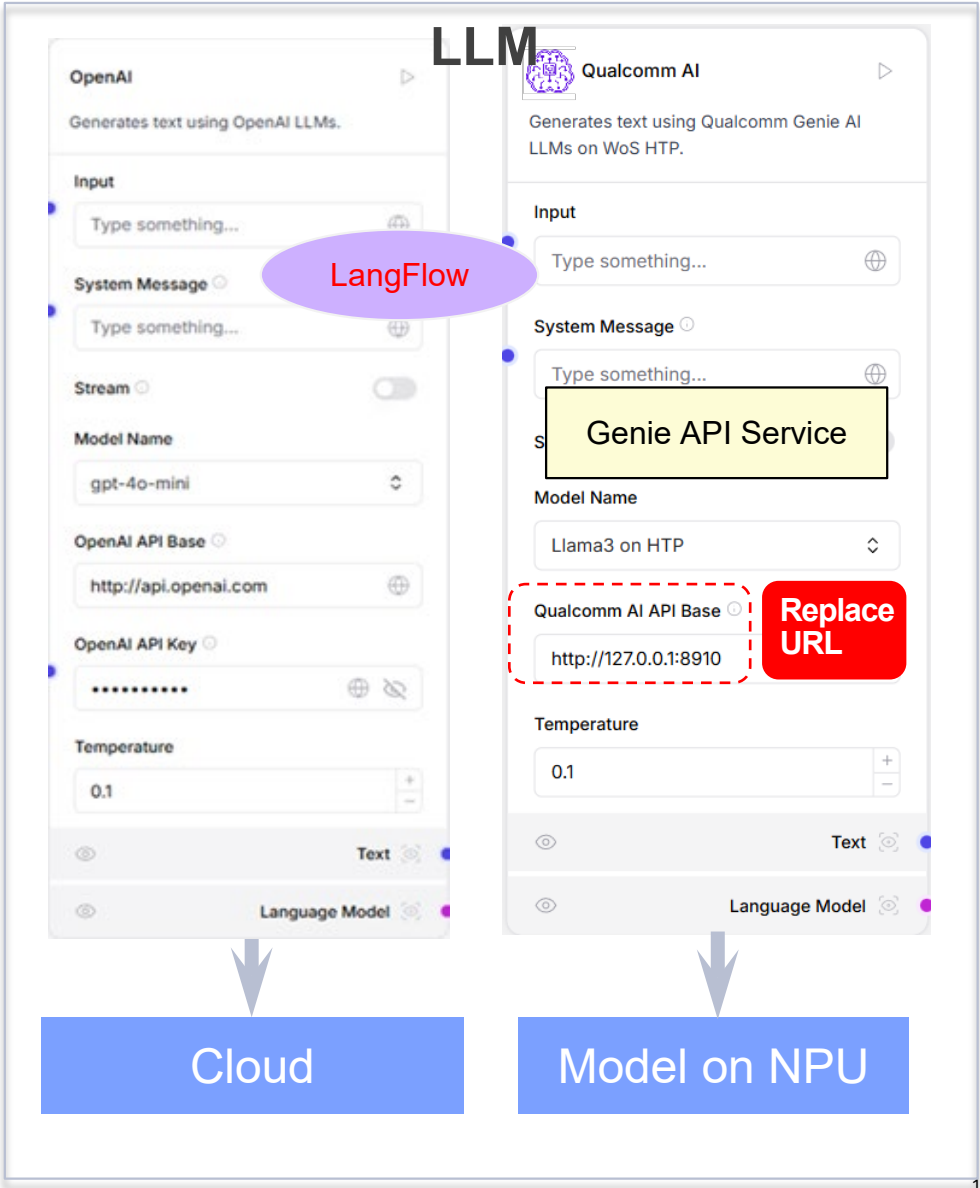
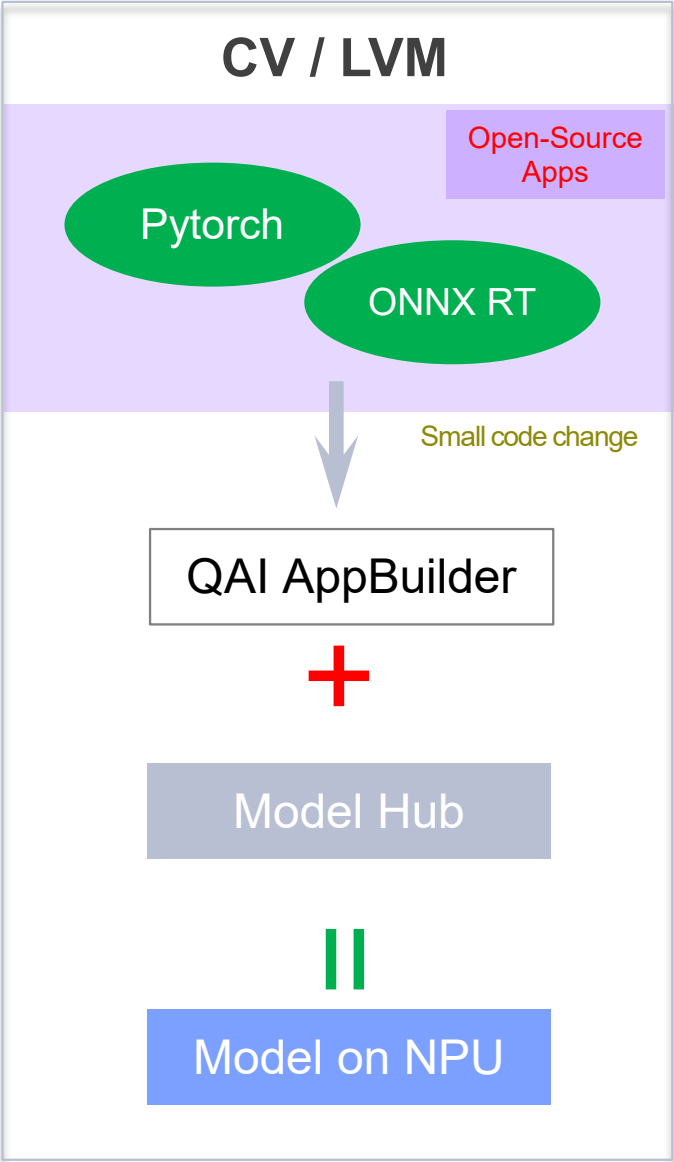
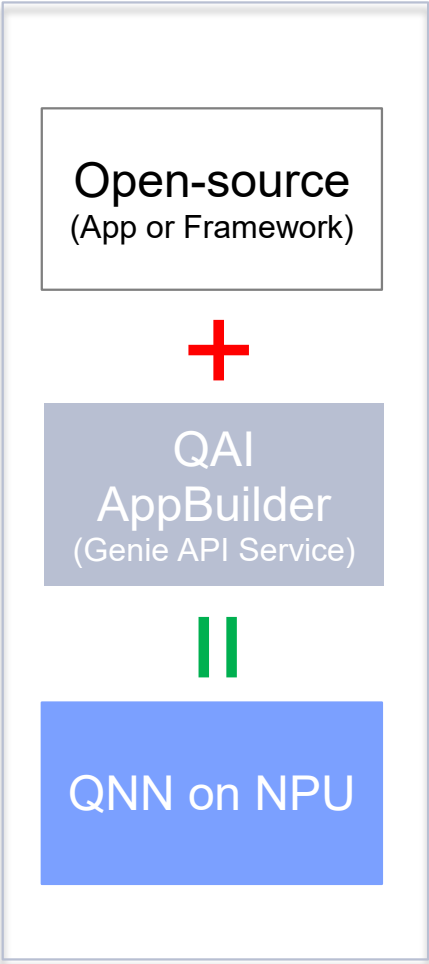
Open-source + QAI AppBuilder

As demonstrated, thanks to the design of QAI AppBuilder, developers can easily port open-source AI applications to the Snapdragon AI PC platform and deploy models to run on the local NPU, achieving **high-performance, low-latency inference**.

For vision models, only minimal modifications to existing PyTorch or ONNX Runtime code are needed. For LLMs, we offer an **OpenAI-compatible API**, enabling quick integration with mainstream applications. Models can be downloaded directly from our **Model Hub**, ready to use out of the box.

In tools like **LangFlow**, simply replacing the API endpoint allows components originally designed to call OpenAI cloud models to **seamlessly switch to local NPU models—without any code changes**.

Open-source + QAI AppBuilder





Section 2

Guide

- Tutorial
- User Guide

Guide

There're tutorial and detailed user guide on Github.

■ Tutorial:

- <https://github.com/quic/ai-engine-direct-helper/blob/main/docs/tutorial.ipynb>

■ User Guide:

- https://github.com/quic/ai-engine-direct-helper/blob/main/docs/user_guide.md

```
from qai_appbuilder import (QNNContext)
QNNConfig.Config(qnn_lib_path=qnn_dir, ...)
realesrgan = RealeSRGAN("realesrgan.bin")
new_image = realesrgan.Inference([image])
del(realesrgan)
new_image = postprocess(new_image)
```

■ Guide on how to prepare QAIRT basic libraries.

Before calling the model-related APIs, need to configure the environment first. We support both x64 and arm64 architecture applications. For programs with different architectures, we need to prepare different versions of QAIRT basic library.

Libraries in 'qnn_dir':
[QAIRT Runtime 2.38.0.zip](#)

X64 process: arm64x-windows-msvc
ARM64 process: aarch64-windows-msvc



Section 3

Env Setup

- X64 Python
- ARM64 Python
- C++

Env Setup

QAI AppBuilder supports calling API through Python and C++ languages. For Python, we support both X64 and ARM64 architectures. At present, the X64 version of Python has a better Python extension ecosystem, while the ARM64 version has better performance. Developers can choose according to their own needs. The following documents detail the steps of how to configure the environment.

- X64 Python:

- <https://github.com/quic/ai-engine-direct-helper/blob/main/docs/python.md>

- ARM64 Python:

- https://github.com/quic/ai-engine-direct-helper/blob/main/docs/python_arm64.md

- C++:

- <https://github.com/quic/ai-engine-direct-helper/tree/main/samples/c%2B%2B>



Section 4

Samples

- Python
- C++
- WebUI / FletUI Apps
- GenieAPIService

Samples

We provide a wealth of examples on Github. Python examples can be run directly without any modification, while C++ examples need to be compiled according to the steps in the document before running.

We provide two chat AI applications, which are implemented by WebUI and Flet respectively, to meet the usage habits and development needs of different users.

Please pay attention to the C++ version of [GenieAPIService](#), and we provide a [compiled version](#) that can be run directly. This is the implementation of the local LLM(large language model) service compatible with OpenAI API, and the large language model running on the local NPU can be directly called through OpenAI API.

■ Python:

- <https://github.com/quic/ai-engine-direct-helper/tree/main/samples/python>

■ C++:

- <https://github.com/quic/ai-engine-direct-helper/tree/main/samples/c%2B%2B/beit>

■ WebUI / FletUI App:

- <https://github.com/quic/ai-engine-direct-helper/tree/main/samples/webui>
- <https://github.com/quic/ai-engine-direct-helper/tree/main/samples/fletui>

■ GenieAPIService:

- <https://github.com/quic/ai-engine-direct-helper/tree/main/samples/genie>

GenieAPIService

■ Document:

- [Guide](#)
- [大语言模型系列\(1\)](#)
- [大语言模型系列\(2\)](#)

■ Service:

- [GenieAPIService \(v2.38\)](#)

■ LLM Model:

- [Qwen 2.0 SSD Model](#)

* Everything has been posted on public github.

<https://github.com/quic/ai-engine-direct-helper>

■ Qwen 2.0 SSD Model

- Download [Qwen 2.0 SSD Model](#) | [GenieAPIService \(v2.38\)](#)
- Unzip the Qwen model and GenieAPIService to below path:
ai-engine-direct-helper\samples\genie\python\models\Qwen2.0-7B-SSD\
ai-engine-direct-helper\samples\GenieAPIService\
□ Start the GenieAPIService:

```
cd ai-engine-direct-helper\samples  
GenieAPIService\GenieAPIService.exe -c "genie\python\models\Qwen2.0-7B-SSD\config.json" -l
```

■ Client Samples

□ C++

```
GenieAPIService\GenieAPIClient.exe --prompt "How to fish?" --stream --  
model "Qwen2.0-7B-SSD"
```

□ Python

• [GenieAPIClient.py](#)

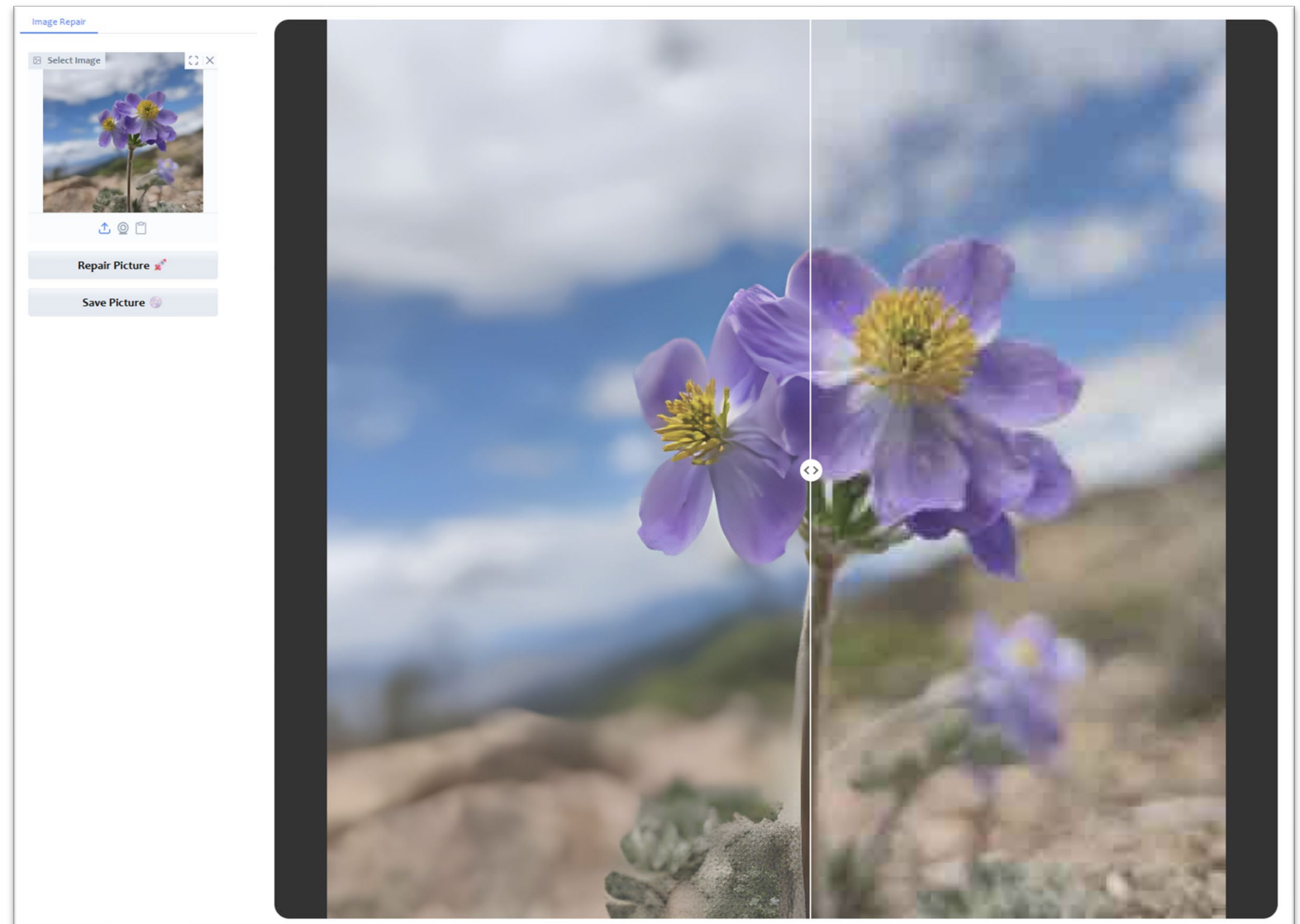
```
cd ai-engine-direct-helper\samples  
python genie\python\GenieAPIClient.py --prompt "How to fish?" -stream --  
model "Qwen2.0-7B-SSD"  
python GenieAPIClientTools.py --prompt "The whether in Shanghai today." -  
-stream
```

Image Repair

- Image Repair WebUI App
- Fast to repair image

* Everything has been posted on public github.

<https://github.com/quic/ai-engine-direct-helper>



Chat UI

- Chat WebUI App
- Chat FletUI App



Chat UI

- Chat WebUI App
- Chat FletUI App

请介绍竺可桢的生平

User

AI

分析生成中, 请耐心等待...

AI

根据提供的参考资料, 我们可以了解到竺可桢的生平大体如下:

- 出生背景:** 竺可桢于1890年3月7日出生在浙江省绍兴县东关镇, 出生在一个浸透了华夏文化乳汁的土地上。他出生时, 故乡已有着丰富的历史和文化。
- 教育与成长:** 他出生在一个充满科学和教育氛围的家庭, 这对他后来的科学和教育事业产生了重要影响。他从小就接受了良好的教育, 对科学产生了浓厚的兴趣。
- 学术成就:** 竺可桢在气象学领域做出了重要贡献, 他的研究被认为具有令人信服的、先进的和实践价值。他以在远东台风研究中的成就而走在了气象界的前头。
- 学术生涯:** 在1918年, 他成功通过了博士论文答辩, 并在哈佛大学获得了博士学位。他本有机会留在美国进行更深入的研究, 但为了国家的需要, 他决定回国。
- 社会活动:** 回国前, 他积极参与了中国留学生的活动, 并与朋友们告别。他不仅参与了学术交流, 还与朋友和学生保持了联系。
- 文化与旅行:** 在旅途中, 他不仅参观了美国的大学和博物馆, 还与在美中国学者和朋友交流, 了解了国内的情况。
- 文化与学术交流:** 在芝加哥大学期间, 他与学者们进行了学术交流, 并且参观了自然史博物馆, 对中国的文化有了更深入的了解。
- 学术考察:** 在芝加哥大学气象系进行学术考察时, 他不仅接受了美国学者的招待, 还与来自不同国家的学者交流, 这表明他的学术影响力和国际视野。
- 文化与教育:** 他参观了科罗拉多州立大学, 并与该校的教员进行了交流, 了解了中国学生在海外的状况。
- 文化与历史:** 在丹佛, 他参观了中国历史和文化相关的展览, 这显示了他对中华文化传统的兴趣和了解。
- 与文化参赞的交流:** 在与文化参赞费慰梅的交流中, 他讨论了去美国的活动安排, 并得到了文化参赞的支持和帮助。
- 与蒋介石的会面:** 他与蒋介石的会面, 可能与他作为科学家的身份和对国家的贡献有关, 这表明了他在国家层面上的影响力。
- 拒绝中大校长职位:** 尽管他与中大有深厚的历史渊源, 但他坚决拒绝了中大校长的职位, 表明了他对教育和学术的追求, 以及对中大传统风气的反感。

综上所述, 竺可桢的生平是一个充满学术探索、社会活动和文化交流的故事, 他以自己的方式为科学和教育事业做出了重要贡献。

Private Q&A

What is your question? ...

Solution Translate Code Analyze Doc Summary Private Q&A



Section 5

Tools

- LangFlow
- QAI Launcher
- Pixi (Python)

5.1 LangFlow	30
5.2 QAI Launcher	33
5.3 Pixi (Python)	36

Tools

There're supporting tools to help developers quickly deploy the development environment.

- LangFlow:

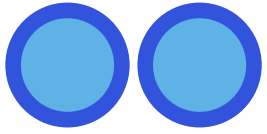
- <https://github.com/quic/ai-engine-direct-helper/tree/main/tools/langflow>

- QAI Launcher:

- <https://github.com/quic/ai-engine-direct-helper/tree/main/tools/launcher>

- Pixi (Python):

- 8.Start_PythonEnv.bat : Start Python environment with Pixi included in QAI Launcher.
 - <https://github.com/quic/ai-engine-direct-helper/tree/main/tools/launcher>



Section 5.1

LangFlow

LangFlow

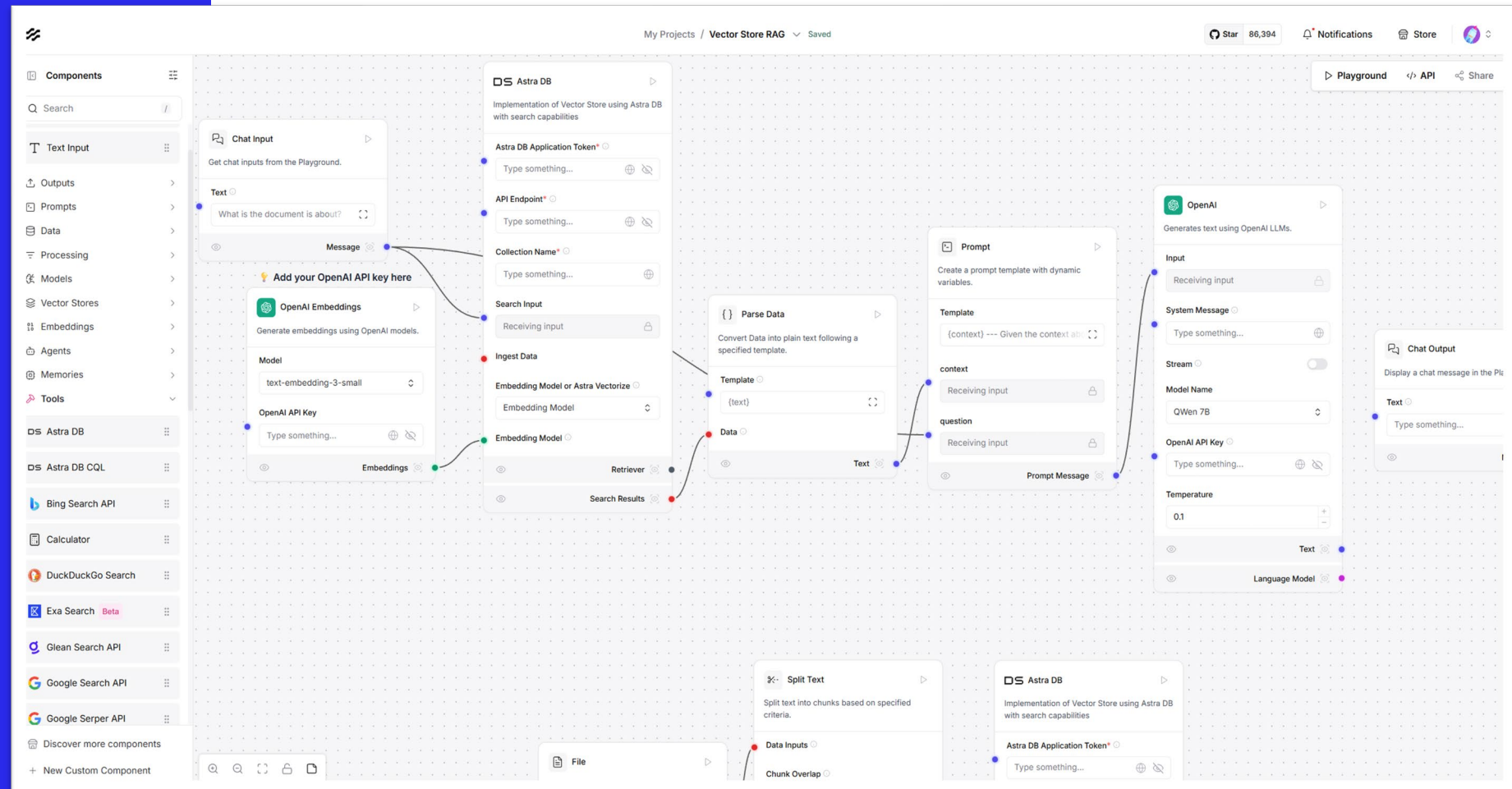
For users who are not good at programming but want to try their own AI application ideas, we recommend using zero-code AI application development tools such as LangFlow to realize their ideas.

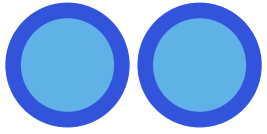
LangFlow is an open-source AI application building tool with intuitive operation and powerful functions. Even without any software development experience, it is easy to get started and quickly complete the construction of AI applications.

We provide a complete solution to connect the local NPU model to LangFlow on GitHub. Users can build an AI application based on the local model in LangFlow by simply dragging and configuring, and realize real local deployment, which is safe and efficient.

LangFlow

- Zero Code Build AI App
- Model on NPU





Section 5.2

QAI Launcher

QAI Launcher

To help users quickly experience the core functions we provide, we have specially prepared a set of automation scripts, which can be used to quickly deploy the running environment and start related applications. In a good network environment, the whole deployment process can usually be completed within two hours, and users can fully experience the local AI capabilities provided by QAI AppBuilder.

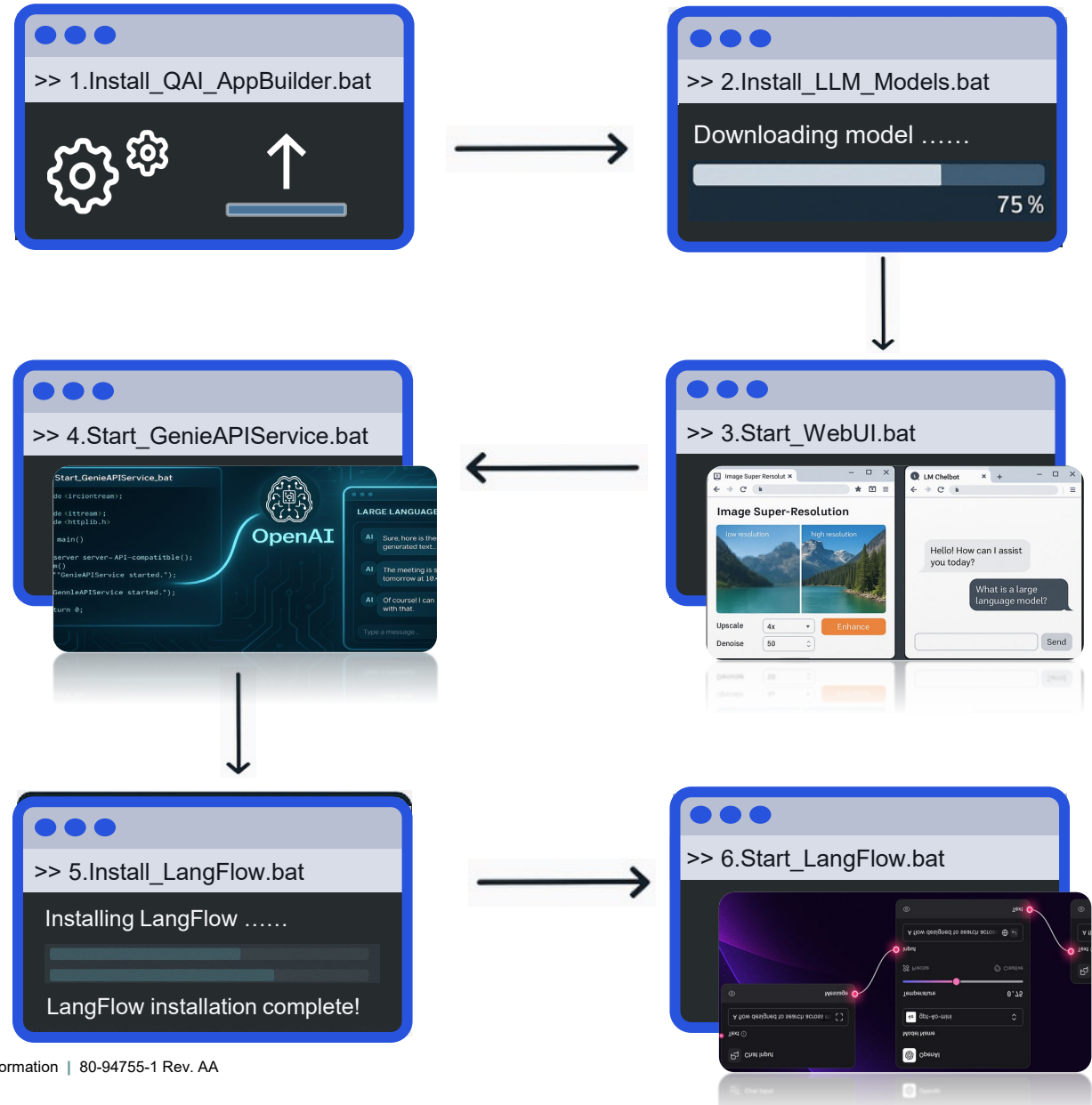
You are welcome to take the time to experience it yourself. We believe you will be deeply impressed by the performance and convenience of local AI applications on Snapdragon AI PC.

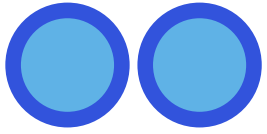
QAI Launcher

How to install?

- QAI Launcher
- 6 scripts to setup environment & launch apps easily

* Everything has been posted on public github.
<https://github.com/quic/ai-engine-direct-helper>





Section 5.3

Pixi (Python)

Pixi (Python)

Pixi (Python)

- Python Environment
- No additional dependency

* Everything has been posted on public github.

<https://github.com/quic/ai-engine-direct-helper>

We have established an independent Python environment without external dependence by using Pixi tool, which will not conflict with the locally installed Python environment and is very convenient to use.

```
>> 8.Start_PythonEnv.bat
```

```
Python 3.12.8 | packaged by conda-forge | (main, Dec 5  
2024, 14:06:27) [MSC v.1942 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more  
information.
```

```
>>>
```



Section 6

Models

- Compile Model
- Convert DLC to BIN
- Model Resource

Models

For AI applications, the model is very important and essential. This document provides several simple commands to demonstrate how to transform the model in the format of ONNX and DLC into the model in the format of QNN context binary that can be directly run on NPU.

The model of ONNX format can usually be converted from PyTorch or TensorFlow, while DLC format is a platform-independent format, which can be converted into QNN context binary format for different platforms and architectures, so that it can be loaded and run on the corresponding platforms with the best performance.

For detailed information about model transformation, please refer to the complete [QAIRT](#) document.

Models

■ QAIRT & QAI AppBuilder:

- [Qualcomm AI Runtime SDK \(v2.37.1\)](#)
- [QAI AppBuilder \(v2.38\)](#)

■ DLC & QNN:

- [Guide](#)
- [Convert Tools](#)

■ Model Resource:

- [Qwen 2.0 SSD Model](#)
- [CV Models](#)
- [Samples](#)

■ Compile Model

- Compile ONNX model to QNN
 - [Setup environment on WoS device](#)
 - [Convert model tutorial](#)
- Sample steps

Convert real_esrgan_x4plus ONNX model to FP16 QNN model(ARM64 windows target):

```
C:\Programs\Python\venv_310\Scripts\activate.bat
Set QNN_SDK_ROOT=C:\Qualcomm\AISTack\QAIRT\2.37.1.250807\
Set PYTHONPATH=%QNN_SDK_ROOT%lib\python
Set PATH=%QNN_SDK_ROOT%lib\aarch64-windows-msvc;%QNN_SDK_ROOT%bin\aarch64-
windows-msvc;%PATH%
```

```
python %QNN_SDK_ROOT%bin\x86_64-windows-msvc\qnn-onnx-converter --
input_network model\real_esrgan_x4plus.onnx --input_dim input 1,3,512,512 --
no_simplification --output_path output\qnn_model.cpp --float_bitwidth 16
```

```
python %QNN_SDK_ROOT%bin\aarch64-windows-msvc\qnn-model-lib-generator -c
C:\models\real-esrgan-x4plus\output\qnn_model.cpp -b C:\models\real-esrgan-
x4plus\output\qnn_model.bin -o output -t windows-aarch64
```

```
%QNN_SDK_ROOT%bin\aarch64-windows-msvc\qnn-context-binary-generator.exe --
backend QnnHtp.dll --model output\ARM64\qnn_model.dll --output_dir output --
config_file htp_backend_ext_fp16_v73.json --binary_file real_esrgan_x4plus
```

Models

■ QAIRT & QAI AppBuilder:

- [Qualcomm AI Runtime SDK \(v2.37.1\)](#)
- [QAI AppBuilder \(v2.38\)](#)

■ DLC & QNN:

- [Guide](#)
- [Convert Tools](#)

■ Model Resource:

- [Qwen 2.0 SSD Model](#)
- [CV Models](#)
- [Samples](#)

■ Compile Model

- Compile ONNX model to QNN
 - [Setup environment on WoS device](#)
 - [Convert model tutorial](#)
- Sample steps

Convert real_esrgan_x4plus ONNX model to quantify QNN model(ARM64 windows target), need additional parameters such as 'input_list', 'act_bw', 'weight_bw', 'bias_bw', refer to above document for detailed steps:

Default quantify:

```
python %QNN_SDK_ROOT%bin\x86_64-windows-msvc\qnn-onnx-converter --  
input_network model\real_esrgan_x4plus.onnx --input_dim input 1,3,512,512 --  
no_simplification --output_path output/qnn_model.cpp --input_list  
input_list.txt
```

A8w8b8 quantify:

```
python %QNN_SDK_ROOT%bin\x86_64-windows-msvc\qnn-onnx-converter --  
input_network model\real_esrgan_x4plus.onnx --input_dim input 1,3,512,512 --  
no_simplification --output_path output/qnn_model.cpp --input_list  
input_list.txt --act_bw 8 --weight_bw 8 --bias_bw 8
```

A16W8 quantify:

```
python %QNN_SDK_ROOT%bin\x86_64-windows-msvc\qnn-onnx-converter --  
input_network model\real_esrgan_x4plus.onnx --input_dim input 1,3,512,512 --  
no_simplification --output_path output/qnn_model.cpp --input_list  
input_list.txt --act_bw 16 --weight_bw 8
```

Models

■ QAIRT & QAI AppBuilder:

- [Qualcomm AI Runtime SDK \(v2.37.1\)](#)
- [QAI AppBuilder \(v2.38\)](#)

■ DLC & QNN:

- [Guide](#)
- [Convert Tools](#)

■ Model Resource:

- [Qwen 2.0 SSD Model](#)
- [CV Models](#)
- [Samples](#)

■ Convert DLC to BIN

- Install QAIRT SDK v2.37.1 ([Download](#))
 - Install to: C:\Qualcomm\AISTack\QAIRT\2.37.1.250807\
- Install QAI AppBuilder v2.38 ([Python x64](#) | [Python ARM64](#) | [C++ ARM64](#))
- Get Config Files
 - [DLC2BIN Config \(v73\)](#).
 - Rename the 'real_esrgan_x4plus' in 'extensions.conf' to your model's name.
- Copy QNN Files
 - Copy 'libqnnhtpv73.cat' & 'libQnnHtpV73Skel.so' from below path to DLC model folder.
 - C:\Qualcomm\AISTack\QAIRT\2.37.1.250807\lib\hexagon-v73\unsigned

```
Set QNN_SDK_ROOT=C:\Qualcomm\AISTack\QAIRT\2.37.1.250807\  
Set PATH=%QNN_SDK_ROOT%lib\aaarch64-windows-msvc;%QNN_SDK_ROOT%bin\aaarch64-  
windows-msvc;%PATH%
```

Fp16:

```
qnn-context-binary-generator.exe --log_level error --backend QnnHtp.dll --  
model QnnModelDlc.dll --dlc_path real_esrgan_x4plus-real-esrgan-x4plus-  
float.dlc --binary_file "real_esrgan_x4plus.fp16" --output_dir output --  
config_file config_local.json
```

Int8:

```
qnn-context-binary-generator.exe --log_level error --backend QnnHtp.dll --  
model QnnModelDlc.dll --dlc_path real_esrgan_x4plus-real-esrgan-x4plus-  
w8a8.dlc --binary_file "real_esrgan_x4plus.int8" --output_dir output --  
config_file config_local.json
```



Section 7

GUI

- Python (PyQt, WebUI, FletUI)
- JS (Electron)
- C++ (Qt, MFC...)
- C# (WinForms, WPF...)

GUI

There is not much difference between the development of GUI application on our Snapdragon AI PC and the traditional Windows platform. Python, Electron, QT, MFC, WinForms, WPF and other GUI development tools can all be used. For developing AI applications, it will be more efficient to develop through Python extensions like Gradio, Streamlit. These Python extensions, usually a few simple lines of code, can realize a relatively rich GUI interface. You can choose a suitable GUI development environment according to your own needs.

- Python (WebUI, FletUI, PyQt)
 - WebUI: [Gradio](#) / [Streamlit](#)
 - PyQt: [PyQt Tutorial](#)
- JS (Electron)
 - [Build cross-platform desktop apps with JavaScript, HTML, and CSS | Electron](#)
- C++ (Qt, MFC...)
- C# (WinForms, WPF...)



Section 8

Key Points

Key Points

- How to run CV / LVM model
- How to run LLM model
- How to get models

- How to run CV / LVM model
 - [Tutorial](#)
 - [User Guide](#)
 - [X64 Python](#)
 - [ARM64 Python](#)
 - [Python Samples](#) | [C++ Samples](#)
- How to run LLM model
 - [GenieAPIService \(v2.38\)](#)
 - [GenieAPIService Guide](#)
 - [C++ Sample](#) | [Python Sample](#)
 - [大语言模型系列\(1\)](#)
 - [大语言模型系列\(2\)](#)
- How to get models
 - [Qwen 2.0 SSD Model](#)
 - [CV Models](#)
 - [Samples](#)



Section 9

Resource

Resource

- [CSDN Qualcomm Zone](#)
- [CDSN AI PC Forum](#)



<https://bbs.csdn.net/forums/qualcomm?typeId=4318918>

Open Source

Everything Is Here:

<https://github.com/quic/ai-engine-direct-helper>

(Report Issues on GitHub Anytime)

Questions?

For additional information or to submit technical questions, go to: <https://www.qualcomm.com/support>

LEGAL INFORMATION

Your access to and use of this material, along with any documents, software, specifications, reference board files, drawings, diagnostics and other information contained herein (collectively this “Material”), is subject to your (including the corporation or other legal entity you represent, collectively “You” or “Your”) acceptance of the terms and conditions (“Terms of Use”) set forth below. If You do not agree to these Terms of Use, you may not use this Material and shall immediately destroy any copy thereof.

1) Legal Notice.

This Material is being made available to You solely for Your internal use with those products and service offerings of Qualcomm Technologies, Inc. (“**Qualcomm Technologies**”), its affiliates and/or licensors described in this Material, and shall not be used for any other purposes. If this Material is marked as “**Qualcomm Internal Use Only**”, no license is granted to You herein, and You must immediately (a) destroy or return this Material to Qualcomm Technologies, and (b) report Your receipt of this Material to qualcomm.support@qti.qualcomm.com. This Material may not be altered, edited, or modified in any way without Qualcomm Technologies’ prior written approval, nor may it be used for any machine learning or artificial intelligence development purpose which results, whether directly or indirectly, in the creation or development of an automated device, program, tool, algorithm, process, methodology, product and/or other output. Unauthorized use or disclosure of this Material or the information contained herein is strictly prohibited, and You agree to indemnify Qualcomm Technologies, its affiliates and licensors for any damages or losses suffered by Qualcomm Technologies, its affiliates and/or licensors for any such unauthorized uses or disclosures of this Material, in whole or part.

Qualcomm Technologies, its affiliates and/or licensors retain all rights and ownership in and to this Material. No license to any trademark, patent, copyright, mask work protection right or any other intellectual property right is either granted or implied by this Material or any information disclosed herein, including, but not limited to, any license to make, use, import or sell any product, service or technology offering embodying any of the information in this Material.

THIS MATERIAL IS BEING PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESSED, IMPLIED, STATUTORY OR OTHERWISE. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUALCOMM TECHNOLOGIES, ITS AFFILIATES AND/OR LICENSORS SPECIFICALLY DISCLAIM ALL WARRANTIES OF TITLE, MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, COMPLETENESS OR ACCURACY, AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MOREOVER, NEITHER QUALCOMM TECHNOLOGIES, NOR ANY OF ITS AFFILIATES AND/OR LICENSORS, SHALL BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY EXPENSES, LOSSES, USE, OR ACTIONS HOWSOEVER INCURRED OR UNDERTAKEN BY YOU IN RELIANCE ON THIS MATERIAL.

Certain product kits, tools and other items referenced in this Material may require You to accept additional terms and conditions before accessing or using those items.

Technical data specified in this Material may be subject to U.S. and other applicable export control laws. Transmission contrary to U.S. and any other applicable law is strictly prohibited.

Nothing in this Material is an offer to sell any of the components or devices referenced herein.

This Material is subject to change without further notification.

In the event of a conflict between these Terms of Use and the *Website Terms of Use* on www.qualcomm.com, the *Qualcomm Privacy Policy* referenced on www.qualcomm.com, or other legal statements or notices found on prior pages of the Material, these Terms of Use will control. In the event of a conflict between these Terms of Use and any other agreement (written or click-through, including, without limitation any non-disclosure agreement) executed by You and Qualcomm Technologies or a Qualcomm Technologies affiliate and/or licensor with respect to Your access to and use of this Material, the other agreement will control.

These Terms of Use shall be governed by and construed and enforced in accordance with the laws of the State of California, excluding the U.N. Convention on International Sale of Goods, without regard to conflict of laws principles. Any dispute, claim or controversy arising out of or relating to these Terms of Use, or the breach or validity hereof, shall be adjudicated only by a court of competent jurisdiction in the county of San Diego, State of California, and You hereby consent to the personal jurisdiction of such courts for that purpose.

2) Trademark and Product Attribution Statements.

Qualcomm is a trademark or registered trademark of Qualcomm Incorporated. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the U.S. and/or elsewhere. The Bluetooth® word mark is a registered trademark owned by Bluetooth SIG, Inc. Other product and brand names referenced in this Material may be trademarks or registered trademarks of their respective owners.

Snapdragon and Qualcomm branded products referenced in this Material are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patented technologies are licensed by Qualcomm Incorporated.