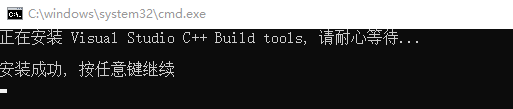
1. **Prerequisites** 
   1. Rust

Rust needs to be installed in order to cross-compile the HuggingFace tokenizer for Android. To get started with Rust, first install the Visual Studio C++ Build Tools. The following steps are based on a Windows environment installation.

1.1.1 Installing Visual Studio C++ Build Tools Download the msvc-buildtools-with-sdk.zip file, extract it, and run the install.bat file.A screenshot of a computer

Description automatically generated

After successful installation.



A screenshot of a computer

Description automatically generated

1.1.2 Installing Rust

Setting Environment Variables

RUSTUP\_DIST\_SERVER <https://mirrors.tools.huawei.com/rustup/>

RUSTUP\_UPDATE\_ROOT <https://mirrors.tools.huawei.com/rustup/rustup/>

A screenshot of a computer program

Description automatically generated

After downloading rustup-init.exe (change the file extension if necessary), simply double-click it.

A screenshot of a computer program

Description automatically generated

Enter 2/x86\_64-pc-windows-msvc/ enter/ enter/ y/ 1, in sequence, as shown below.

A screenshot of a computer program

Description automatically generated

After a successful installation, the following message will be displayed

A screenshot of a computer program

Description automatically generated

#Configure rust environment variables

*PATH=C:\Users\y60044858\.rustup\toolchains\innersource-distribution-x86\_64-pc-windows-msvc\bin*

To check the installation information, execute rustc --version.

A screenshot of a computer code

Description automatically generated

* 1. **JDK**

Based on the Windows operating system, install JDK 1.8 (jdk-8u201-windows-x64.msi) and configure the environment variables.

Execute jdk-8u201-windows-x64.msi to complete the installation.

JDK Environment Variable Configuration:

*JAVA\_HOME=D:\D\Android\Java\jdk1.8.0\_201*

*CLASSPATH=.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar*

*PATH=%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin*

To check the installation information, execute java -version.

**A black screen with white text

Description automatically generated**

* 1. **Git**

Download Git

Extract Git-2.31.1-64-bit.rar and complete the installation.

Setting Environment Variables

*PATH=D:\D\Git\bin*

To check the installation information, execute git -version

**A black screen with white text

Description automatically generated**

*#* GIT Network Proxy Configuration:*：*

*#* View global configuration variables

*git config --list*

*#* Configure using commands

*git config --global http.proxy http://y60044858:password@proxyhk.huawei.com:8080/*

*git config --global https.proxy https://y60044858:password@proxyhk.huawei.com:8080/*

*git config --global http.sslverify false*

*#* To remove the configuration, execute the following commands

*git config --global --unset http.proxy*

*git config --global --unset https.proxy*

* 1. **Android SDK,NDK and CMake**

Download android-sdk\_r24.4.1-windows.zip and extract it.

* + 1. **adb Environment Variable Configuration**

*PATH=D:\D\Android\androidSDK\android-sdk\_r24.4.1-windows\platform-tools*

* + 1. **NDK Environment Variable Configuration**

*ANDROID\_NDK=D:\D\Android\androidSDK\android-sdk\_r24.4.1-windows\ndk\25.1.8937393*

*TVM\_NDK\_CC=%ANDROID\_NDK%/toolchains/llvm/prebuilt/windows-x86\_64/bin/aarch64-linux-android24-clang*

* + 1. **CMake Environment Variable Configuration**

*PATH=D:\D\Android\androidSDK\android-sdk\_r24.4.1-windows\cmake\3.22.1\bin*

1.5 Android Studio

Download android-studio-2023.2.1.23-windows.exe and complete the installation.

* 1. Conda

Download Anaconda3-2024.02-1-Windows-x86\_64.exe and complete the installation. Use conda to manage isolated Python environments to avoid missing dependencies, version incompatibilities, and package conflicts.

1. **MLC-LLM Source code build for Android application**
   1. **mcl-ai/mlc-llm source code download**

# Clone the repository with the specified branch

git clone -b docs\_typo\_mlc\_chat --single-branch https://github.com/mlc-ai/mlc-llm.git

# Enter the mlc-llm project

cd mlc-llm

# Clone the submodule code

git submodule update --init --recursive

# Enter the MLCChat directory

cd ./android/MLCChat

**Environment variable configuration for the code**

# mlc-llm code path

MLC\_LLM\_SOURCE\_DIR=D:\mlc-llm

# TVM Unity runtime is located under MLC LLM's 3rdparty/tvm, so no additional installation is required. Set the following environment variable

TVM\_SOURCE\_DIR=D:\mlc-llm\3rdparty\tvm

* 1. **Install MLC LLM python package**

The MLC LLM Python package can be installed directly from pre-built developer packages or built from source.

Below are the steps to set up build dependencies using pre-built packages in Conda.

*# make sure to start with a fresh environment*

*conda env remove -n mlc-chat-venv*

*# create the conda environment with build dependency*

*conda create -n mlc-chat-venv -c conda-forge "cmake>=3.24" rust git python=3.11*

*A screenshot of a computer program

Description automatically generated*

*# enter the build environment*

*conda activate mlc-chat-venv*

*# install zstd*

*conda install zstd*

*# install vulkan loader、clang、git and git-lfs*

*conda install -c conda-forge clang libvulkan-loader git-lfs git*

*A screenshot of a computer program

Description automatically generated*

*Install mlc-llm-nightly and mlc-ai-nightly*

*python -m pip install --pre -U -f https://mlc.ai/wheels mlc-llm-nightly mlc-ai-nightly*

A screenshot of a computer screen

Description automatically generated

If there is a timeout or the package cannot be found, you can manually download the whl package from the website <https://mlc.ai/wheels> and install it using:

python -m pip install \*.whl

Download mlc\_ai\_nightly-0.15.dev404-cp311-cp311-win\_amd64.whl and mlc\_llm\_nightly-0.1.dev1404-cp311-cp311-win\_amd64 based on your Python version.

TVM Unity Editor Installation and Verification

*# enter the folder of \*.whl*

*d:*

*cd D:\D\download*

*# TVM Unity installation：*

*python -m pip install mlc\_ai\_nightly-0.15.dev404-cp311-cp311-win\_amd64.whl*

*# TVM verification：*

*python -c "import tvm; print(tvm.\_\_file\_\_)"*

A screenshot of a computer program

Description automatically generated

A black background with white text

Description automatically generated

**mlc-llm Installtion and Verification**

*# Install mlc\_llm\_nightly*

*python -m pip install mlc\_llm\_nightly-0.1.dev1404-cp311-cp311-win\_amd64.whl*

*# mlc\_llm Verification*

*mlc\_llm --help*

*python -c "import mlc\_llm; print(mlc\_llm)"*

A screenshot of a computer screen

Description automatically generated

A screen shot of a computer program

Description automatically generated

**2.3 Converting Model Weights**

To run the model using MLC LLM, the model weights need to be converted. The Hugging Face model is used as input and quantized into weights compatible with MLC.

Download the MiniCPM-2B-dpo-bf16-llama-format model library from Hugging Face.

Download openbmb/MiniCPM-2B-dpo-bf16-llama-format from the official Hugging Face website and place it in the dist/models directory.

A screenshot of a computer

Description automatically generated

*convert\_weight*

*# Enter mlc-llm\android\MLCChat*

*cd D:\mlc-llm\android\MLCChat*

*# MiniCPM-2B-dpo-bf16-llama-format*

*mlc\_llm convert\_weight ./dist/models/MiniCPM-2B-dpo-bf16-llama-format/ --quantization q4f16\_1*

*-o dist/bundle/MiniCPM-2B-dpo-bf16-llama-format-q4f16\_1*

A screenshot of a computer screen

Description automatically generated

After successful execution, ndarray-cache.json and params\_sh will be generated in the dist/bundle/MiniCPM-2B-dpo-bf16-llama-format-q4f16\_1 directory

A screenshot of a computer

Description automatically generated

**Generating MLC Chat Configuration**

*# Generate MLC Chat Configurations*

*mlc\_llm gen\_config ./dist/models/MiniCPM-2B-dpo-bf16-llama-format/ --quantization q4f16\_1 -*

*-conv-template redpajama\_chat -o dist/bundle/MiniCPM-2B-dpo-bf16-llama-format-q4f16\_1/*

A screenshot of a computer

Description automatically generated

*After Sucessful execution，****Four files:*** **mlc-chat-config.json、tokenizer.json, tokenizer.model、tokenizer\_config.json**will be generated under dist/bundle/MiniCPM-2B-dpo-bf16-llama-format-q4f16\_1

2.4 Compiling **tvm4j\_core.jar and libtvm4j\_runtime\_packed.so** Dependency Library

1. Modify the MLCChat/mlc-package-config.json file to customize the model built into the Android application.

A screenshot of a computer

Description automatically generated

2. Copy the converted MiniCPM-2B-dpo-bf16-llama-format-q4f16\_1 model to the following directory on your local machine: C:\Users\y60044858\AppData\Local\mlc\_llm\model\_weights\hf\mlc-ai

**Note:** During the compilation process, the system will first search for the model locally in the specified directory. If the model is not found locally, it will be downloaded from the official website: <https://huggingface.co/mlc-ai>.

A screenshot of a computer

Description automatically generated

3. Run the mlc\_llm package command. The execution process might be slightly slow, so please be patient. Use the Git Bash interface to execute the mlc\_llm command. First, configure the environment variables for Python and mlc\_llm:

*PATH=D:\anaconda3\envs\mlc-chat-venv*

*PATH=D:\anaconda3\envs\mlc-chat-venv\Scripts*

Run the mlc\_llm package command：

*mlc\_llm package*

*A screenshot of a computer program

Description automatically generated*

Upon successful execution, the following files will be generated in the /dist/lib/mlc4j directory.

A screenshot of a computer program

Description automatically generated

2.5 Generate APK

Click on "Build → Generate Signed Bundle / APK". If this is your first time generating an APK, you will need to create a key according to the official Android guidelines. This APK will be placed in android/MLCChat/app/release/app-release.apk.

A screenshot of a computer program

Description automatically generated

**2.6 Install ADB and Enable USB Debugging**

Add Platform-Tools to the PATH Environment Variable

*# adb tools variable configuration*

*PATH=D:\D\Android\androidSDK\android-sdk\_r24.4.1-windows\platform-tools*

Enable "USB Debugging" in Developer Mode in your phone settings. Run the following command, and if ADB is installed correctly, your phone will show up as a device:

*adb devices*

A black screen with white text

Description automatically generated

**2.7 Install the APK and Weights on Your Phone**

Open a CMD window and enter the following commands:

*# Open mlc—llm/android/MLCChat*

*cd D:\mlc-llm\android\MLCChat*

*python bundle\_weight.py --apk-path app/release/app-release.apk*

A screenshot of a computer program

Description automatically generated

**2.8 Run the MLCChat Application**

Open the MLCChat application on your phone and run it.

A white background with black dots

Description automatically generated A white background with black dots

Description automatically generated