On-Device ML

Needs, Challenges, Deployment, Low-Level Concepts

Shubham Panchal

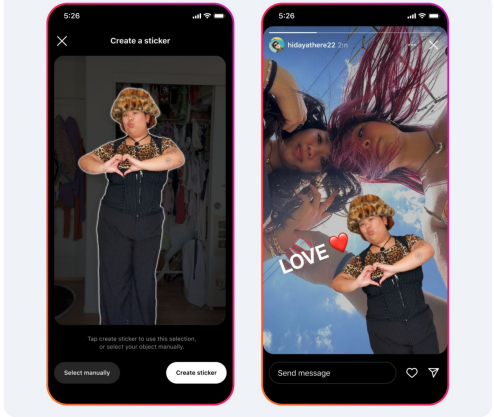
FOSS Meetup Pune - January 2026

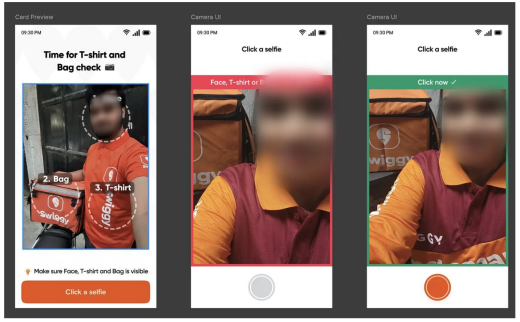
What is On-Device Machine Learning?

"Techniques that enable ML models to execute locally on the user's device i.e. without requiring communication to a hosted service across the Internet"

Enabling Cutouts on Instagram

Cutouts is one of Instagram's latest features for creative expression and storytelling. It lets people transform photos and videos of their favorite moments into animated, personalized stickers that they can share via Reels or Stories. We migrated the Cutouts feature in Instagram to run with ExecuTorch by enabling SqueezeSAM, a lightweight version of the Meta Segment Anything Model (SAM). For both Android and iOS, ExecuTorch was significantly faster compared to the older stack, translating into increases in Cutouts' daily active users (DAU).

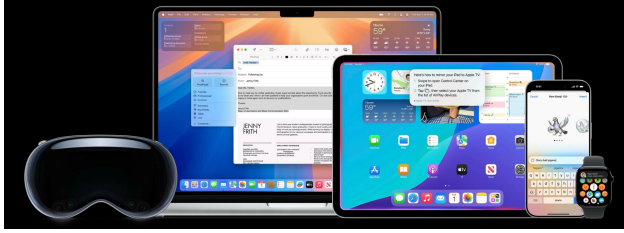




Screenshots of image capture flow where we check for Swiggy bag & t-shirt

Apple Intelligence

Apple Intelligence is the personal intelligence system that puts powerful generative models right at the core of your iPhone, iPad, Mac, Apple Vision Pro, and Apple Watch with incredible new features to help people communicate and work better. You can bring Apple Intelligence features right into your apps, integrate into places across the system via App Intents, and tap into models directly to build your own intelligent experiences.



Speech and Natural Language Processing

Hey Siri: An On-device DNN-powered Voice Trigger for Apple's Personal Assistant

---

Shrey Desai · 2nd
Staff Research Engineer at Meta
3mo · + Follow

Excited to share message translations on WhatsApp are live!

To seamlessly communicate across languages, users can opt-in to translate individual messages or entire threads into their preferred language by long-pressing a message and selecting 'Translate'. Critically, this feature is privacy-preserving by design: all translations are performed on-device ensuring messages remain end-to-end encrypted.

Check out more details here:
- WhatsApp Blog: https://lnkd.in/eK9za7PC
- TechCrunch: https://lnkd.in/emqBAgcH

On-Device ML Is Best For...

Tasks that require minimum latency: next-word prediction, portrait mode in the device camera.
How: The model runtime and the app's code are in the same process, hence no inter-process communication overhead.

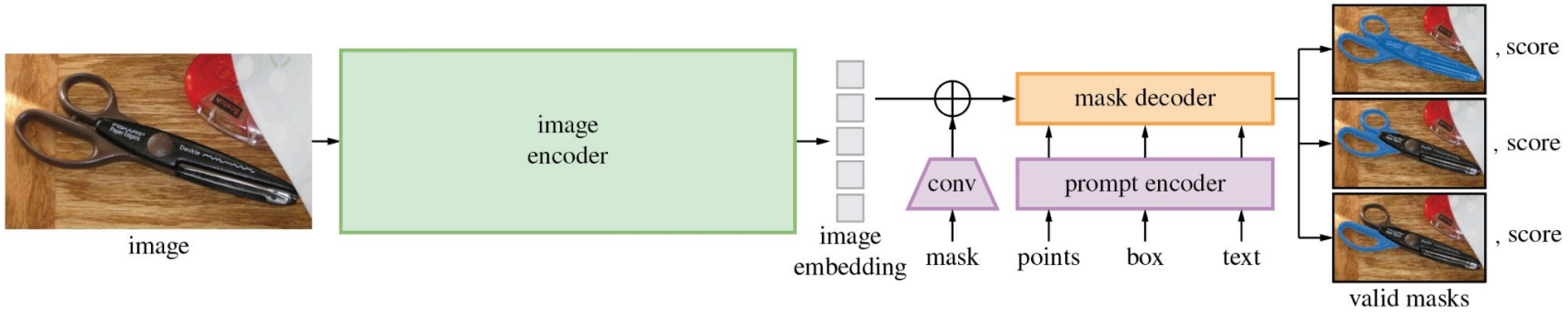
Tasks that ingest sensitive/private data: chat-message translation, face recognition in the photos app
How: No data flows outside the app's process, hence no data leakage. (BONUS[1])

On-Device ML Is Not The Best For...

Tasks that require larger models: image generation, larger chat models, ‘circle to search’
Why: Larger models yield better results, at the cost of increased memory and CPU usage at inference

Tasks that require a special/complex inference: If the model requires special steps that the runtime does not offer, they need to be implemented in the app.

Example: Deploying SAM (Segment Anything)



Pre-processing:

- Standardize image pixels
- Tokenize text input for the decoder
- Binarize mask input for the decoder
- Normalize point coordinates

Post-processing:

- Resize output mask and apply to input image

Model Optimizations

- Quantization: Reduce the computational and memory costs of running inference by representing the weights and activations with low-precision data types like 8-bit integer (int8) instead of the usual 32-bit floating point (float32).

  - `float32` to `int8` implies 4x reduction in size and memory footprint

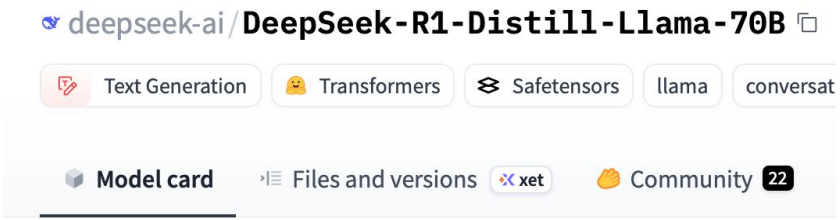
  - `int8` operations might be accelerated on certain microprocessors

  - theoretically, the model loses on quality as it holds less information about the data on which it was trained. Hence, the efficiency vs. performance tradeoff needs to be considered.

Model Optimizations

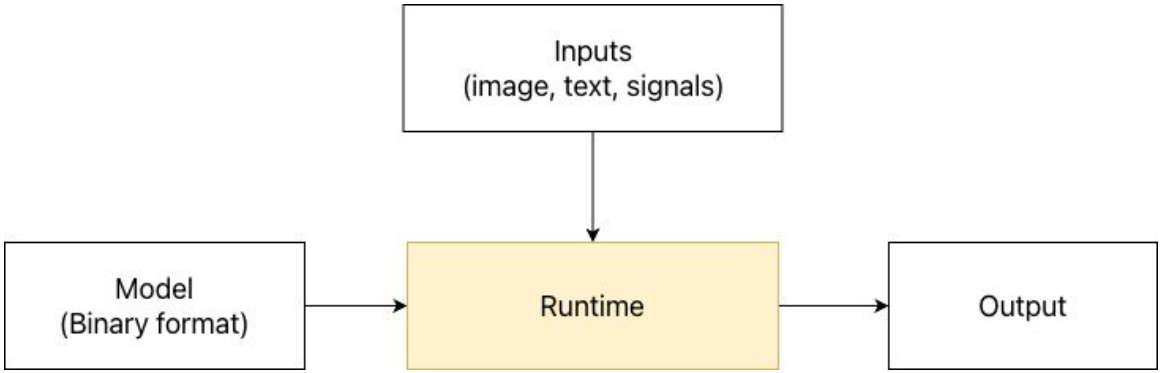
- Distillation: Training a smaller (student) model on the outputs of a bigger (teacher) model.
  - Smaller model, lesser parameters, lesser resource footprint

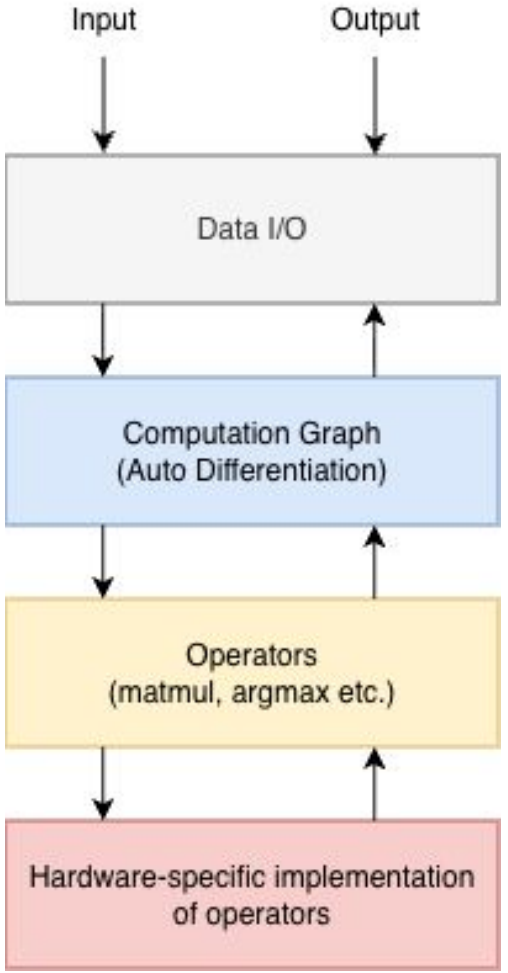
- Model Pruning: Determine ‘insignificant’ weights from the NN and remove them. Reduces size and model latency with minimum impact on the quality of the outputs.



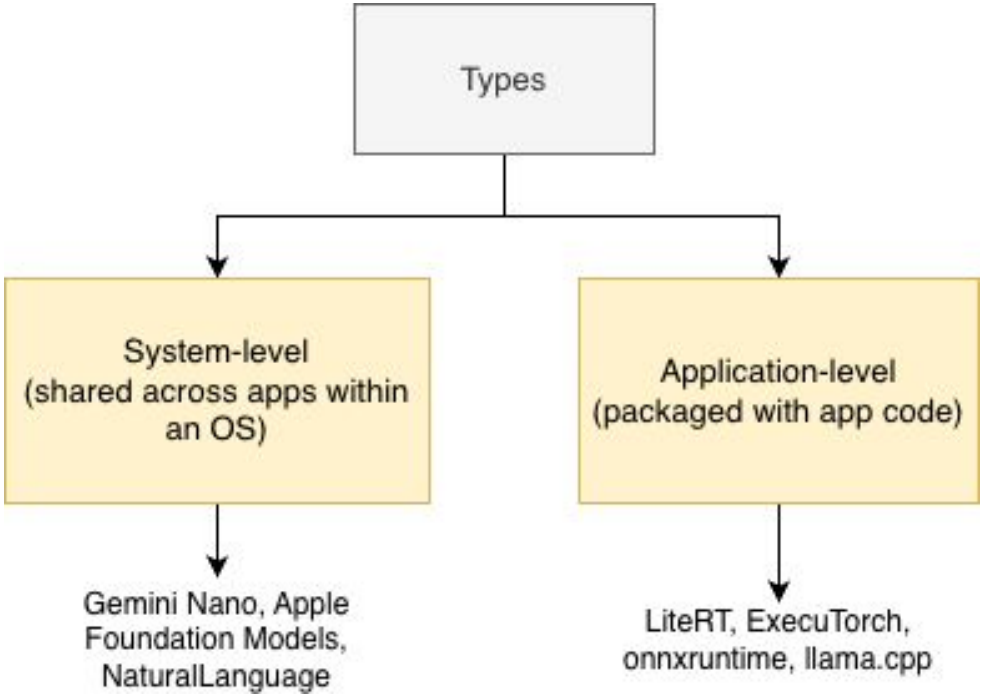
What is a Model Runtime?

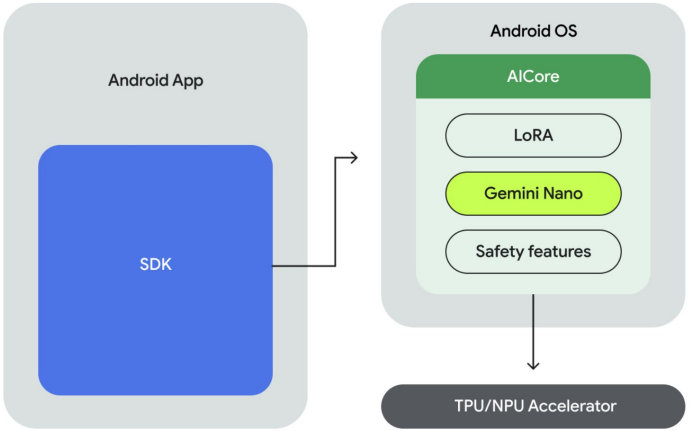
A software that ingests a model file (definition and weights) along with the user input, performs the task, and provides the output.





Types of Model Runtimes





AICore manages model, runtime and safety features.

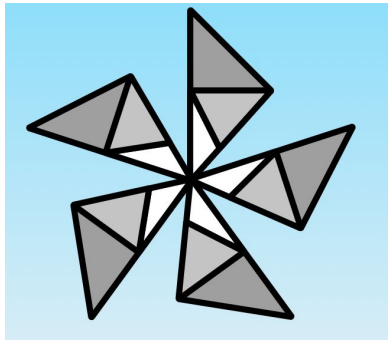
Popular Model Runtimes

- LiteRT (prev. TensorFlow Lite) is a runtime and model format from Google.
- Supports deployment on Android, iOS, web, desktop, and embedded devices with GPU/NPU acceleration.
- Ready-to-use wrapper APIs for common ML tasks like image/text classification, embeddings, and chat models.

- llama.cpp is an open-source pure C/C++ runtime for large-language models (vision/multimodal models) developed initially by Georgi Gerganov.
- Supports inference for most LLM architectures and implements accelerators with CUDA, Vulkan, Arm Kleidi etc.
- Rapidly growing community. Tools like LM-Studio, Docker Models, ollama internally use llama.cpp

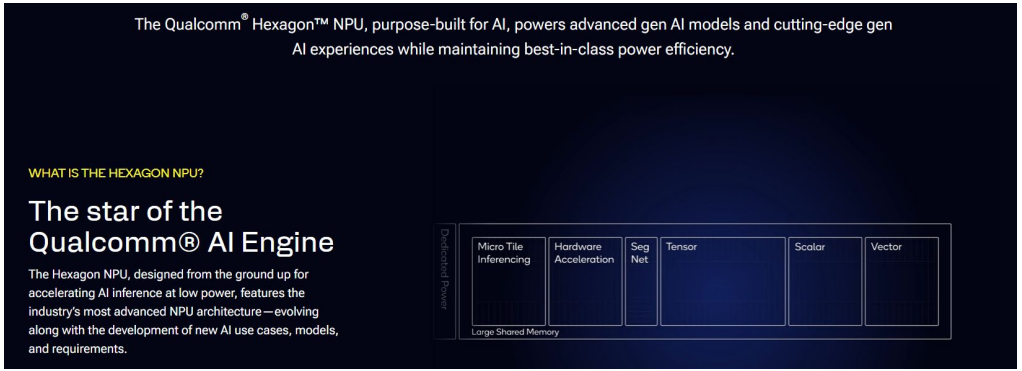
Popular Model Runtimes

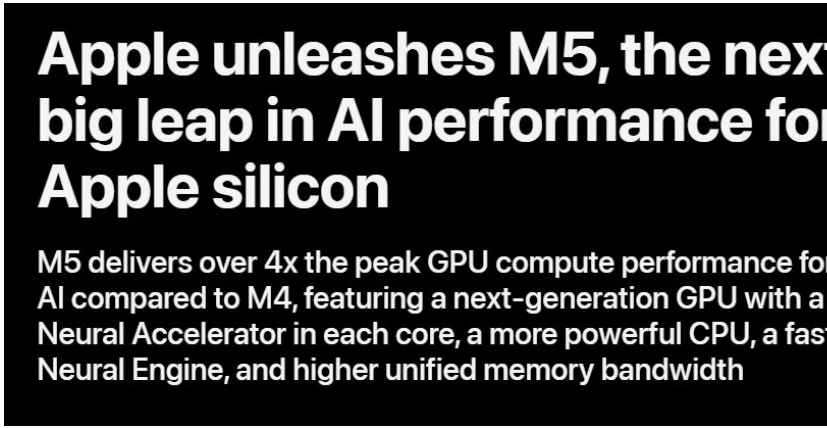
- onnxruntime is an open-source runtime developed by Microsoft for models defined in the ONNX (Open Neural Network Exchange) format.
- Supports inference on web, mobile, and desktop.
- Unique: Supports various model optimization schemes and on-device training

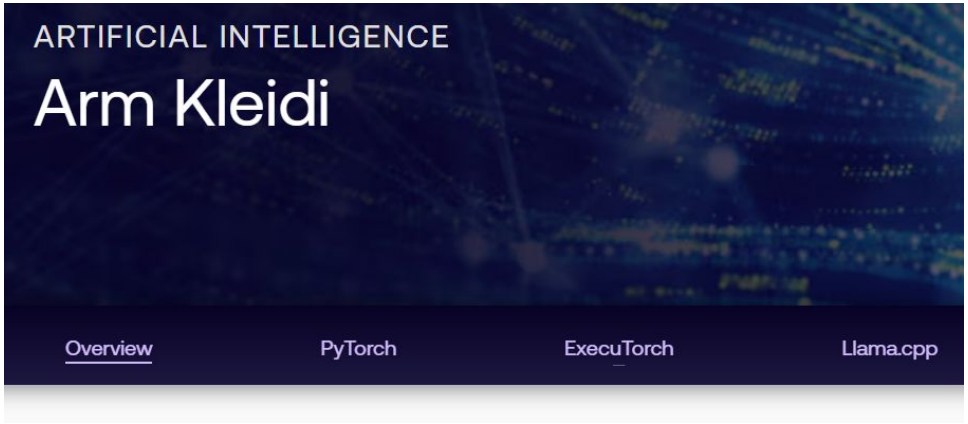


- ExecuTorch is an open-source runtime developed by Meta and is closest to PyTorch.
- Supports GPU/NPU acceleration for Android, Apple devices, and embedded devices.
- Unique: Tight integration with PyTorch, a runtime used by most modern ML models.

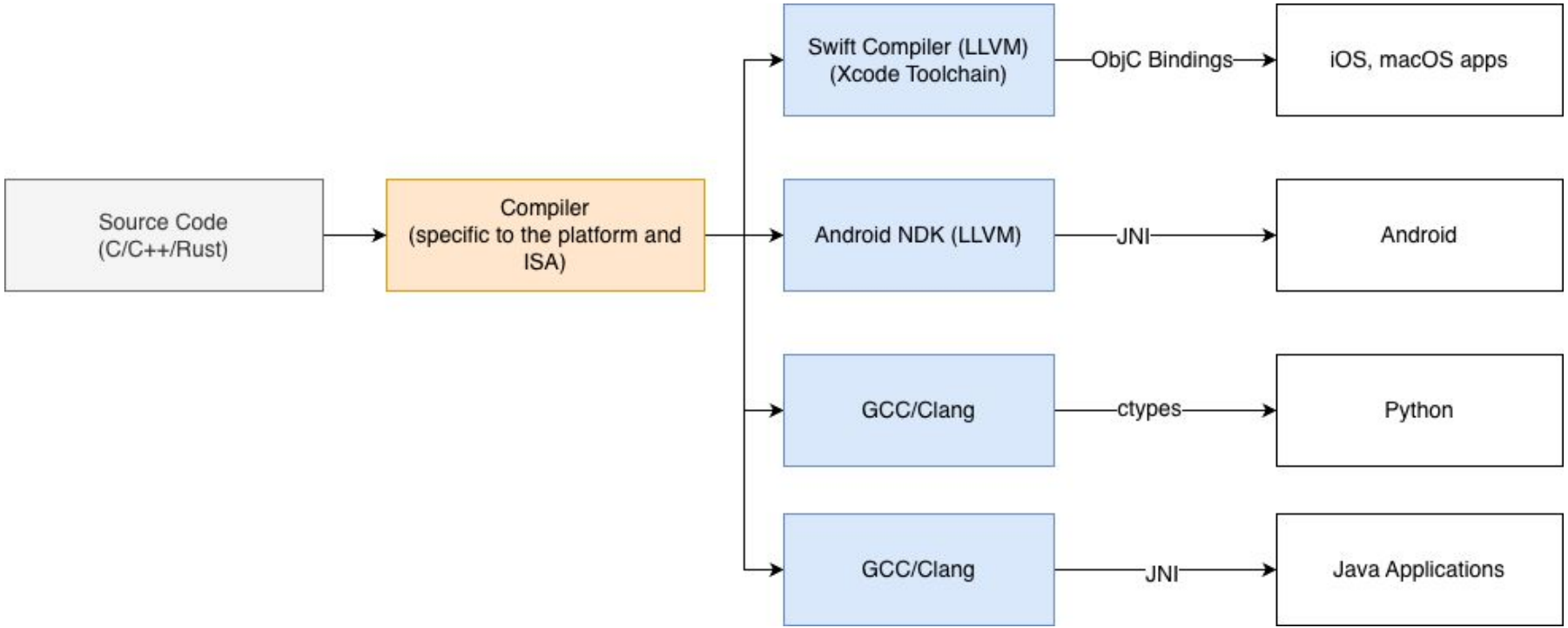
Hardware Improvements







How Does C/C++/Rust Code Integrate with Python, Java or Swift codebases?



How to integrate with JS for Web Apps?
(BONUS[2])

Kotlin/Java - JNI and the Memory API

Write special JNI functions and export them as symbols to the shared libraries (.so/dylib) files

JVM searches for external or native methods and matches them with symbols present in the shared library.

If a match is found, the Java method is bind to the native JNI function.

Newer alternative that does not require writing ‘special’ JNI functions: [Memory API (OpenJDK Project Panama)]

```
extern "C" JNIEXPORT void JNICALL
Java_io_shubham0204_smollm_SmolLM_addChatMessage(
  JNIEnv* env,
  jobject thiz,
  jlong modelPtr,
  jstring message,
  jstring role
) {
  // Use C library functions here...
}
```

```
package io.shubham0204.smollm

class SmolLM {

  private external fun addChatMessage(
    modelPtr: Long,
    message: String,
    role: String
  )

}
```

Python - `ctypes` and `Cython`

The Python VM can bind shared libraries and symbols/objects from the shared libraries can be accessed via `ctypes`.

```c
// C source code
float test_add(float x, float y) {
    return x + y;
}
```

`Cython` allows writing C extensions for Python. The routines are written in a `.pyx` file that are compiled by Cython to a Python extension module. This module, like any other Python modules, like be used with an `import` statement.

```python
import ctypes

mylib = ctypes.CDLL("mylib.so")

test_add = mylib.test_add
test_add.argtypes = [ctypes.c_float, ctypes.c_float]
test_add.restype = ctypes.c_float

print(mylib.test_add(34.55, 23))
```

Swift - `clang` Modules

The Swift compiler can detect symbols from a C/C++ header files guided by a `clang` module map.

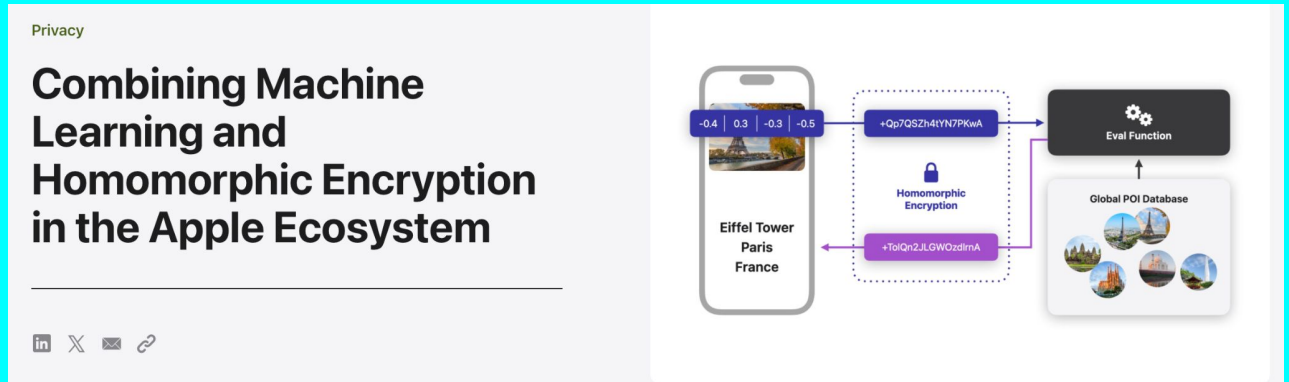
This `clang` module can be used in the Swift code.

```
import forestLib

let tree = Tree(.Oak)
```

```
module forestLib {
    header "forest.h"
    header "tree.h"

    export *
}
```

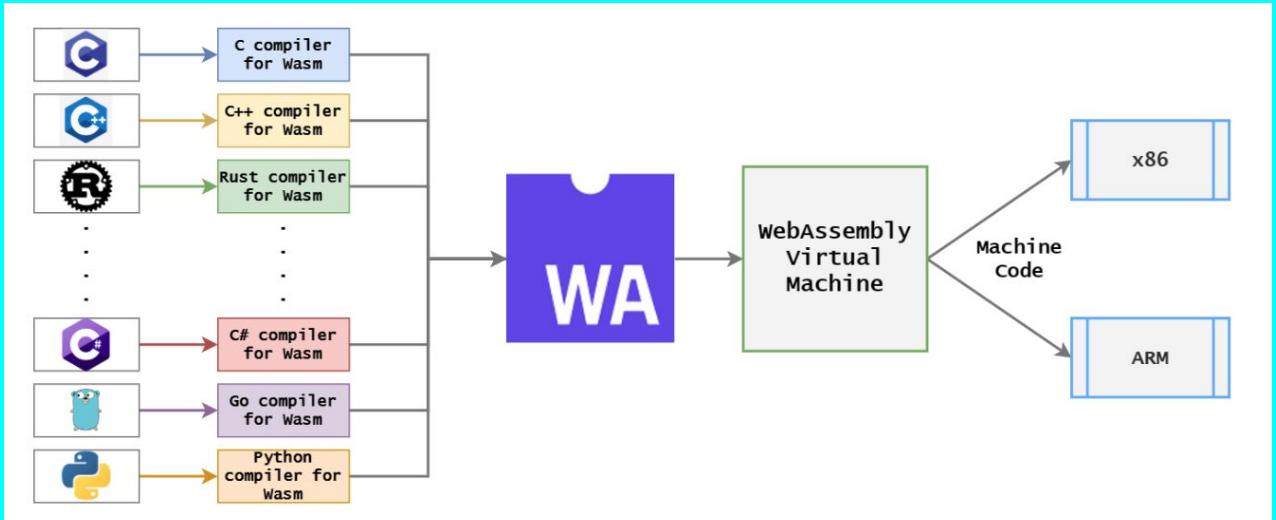
BONUS #1 - ML Inference on Encrypted Data

- ML models need data to be decrypted before performing computations over it, even if the data was encrypted when at rest.
- Homomorphic Encryption is a form of encryption that allows computations to be performed on encrypted data without first having to decrypt it.



BONUS #2 - Using C/C++/Rust in Web Apps

- Compile native code to a WebAssembly module and download it when the page loads.
- Browser's WASM runtime loads the module; JS can now calls functions from the module directly.
- Example



Questions
or
Thoughts!