Run LLMs on your Phone

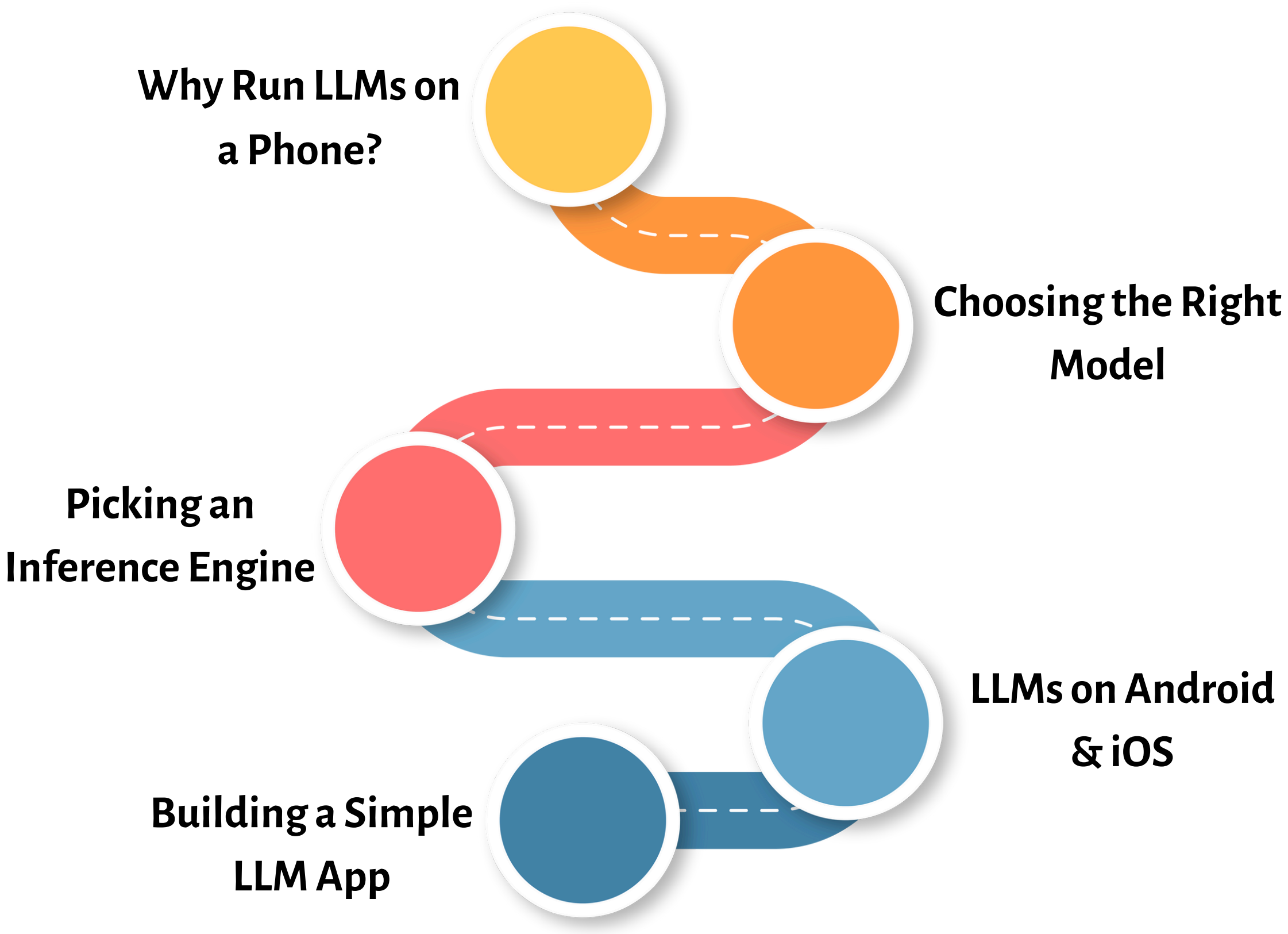
Your one stop guide to building LLM powered mobile apps

Bhavishya Pandit

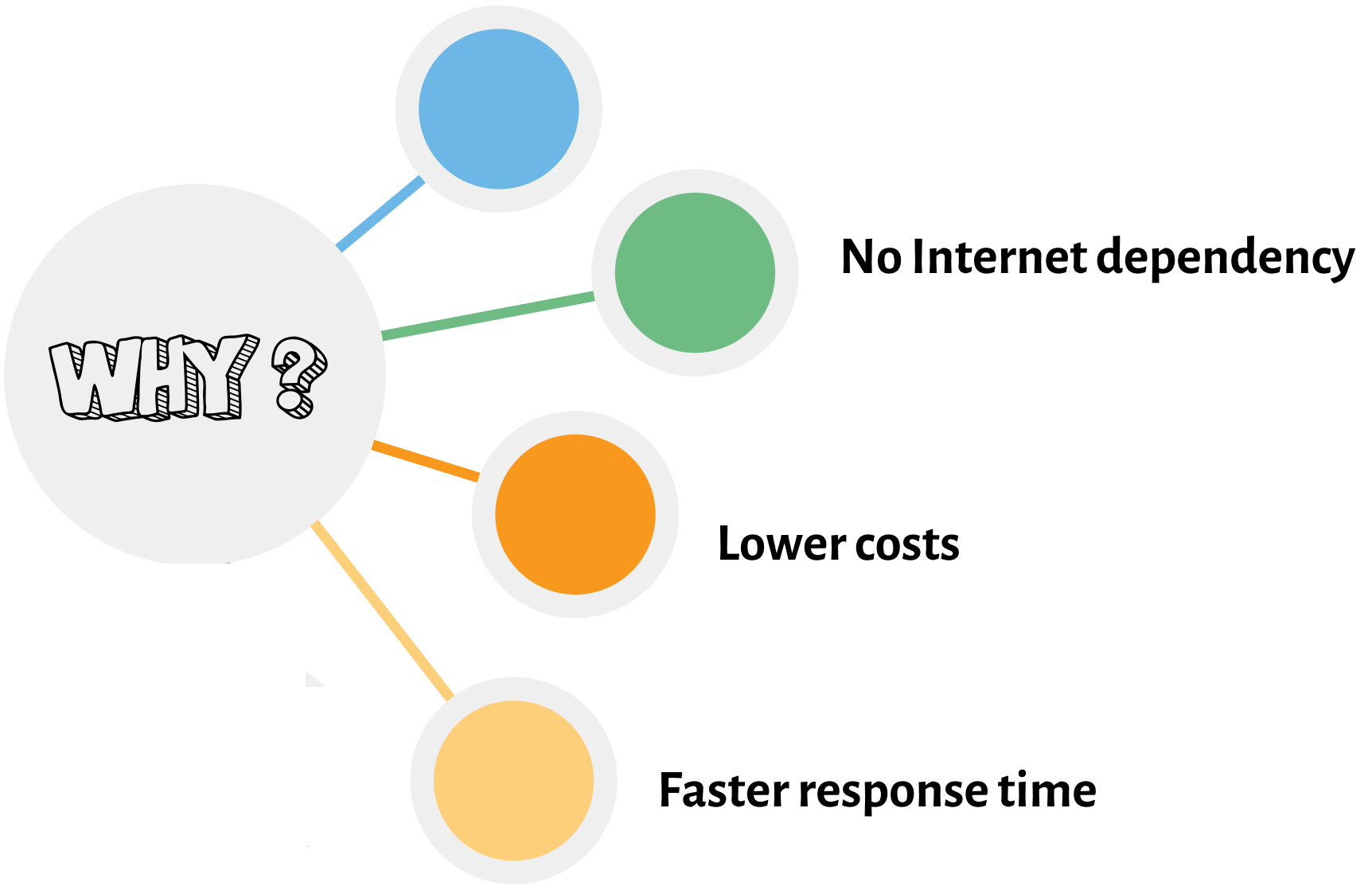
Introduction

Have you ever waited too long for LLM's response? Or are you worried about sending sensitive data to third party?

LLMs can be slow, expensive, and privacy-invasive. Running them locally on your phone unlocks faster, private, and offline AI interactions.
In this post, we'll cover:

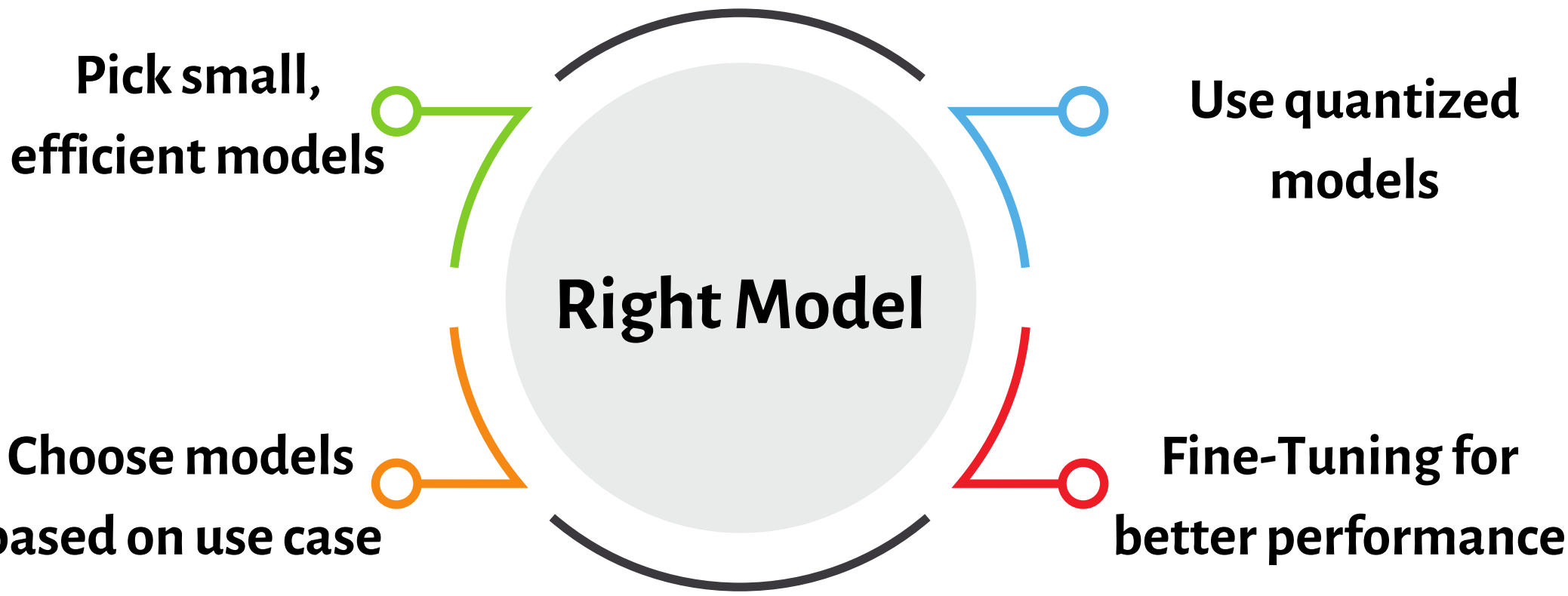


Why Run LLM on a Phone?



- **Privacy & Data security**: Cloud-based LLMs send your data to external servers, posing security risks. Running locally keeps data private, avoiding leaks.
- **No Internet dependency**: AI apps should work anywhere, anytime, even without internet. On-device LLMs enable offline assistants, useful for travel, remote areas.
- **Lower costs**: Cloud-based APIs charge per request, increasing costs over time. Running LLMs locally eliminates recurring expenses, making AI more affordable.
- **Faster response time**: Cloud LLMs suffer from latency due to server communication. Local execution ensures instant responses, essential for real-time applications.

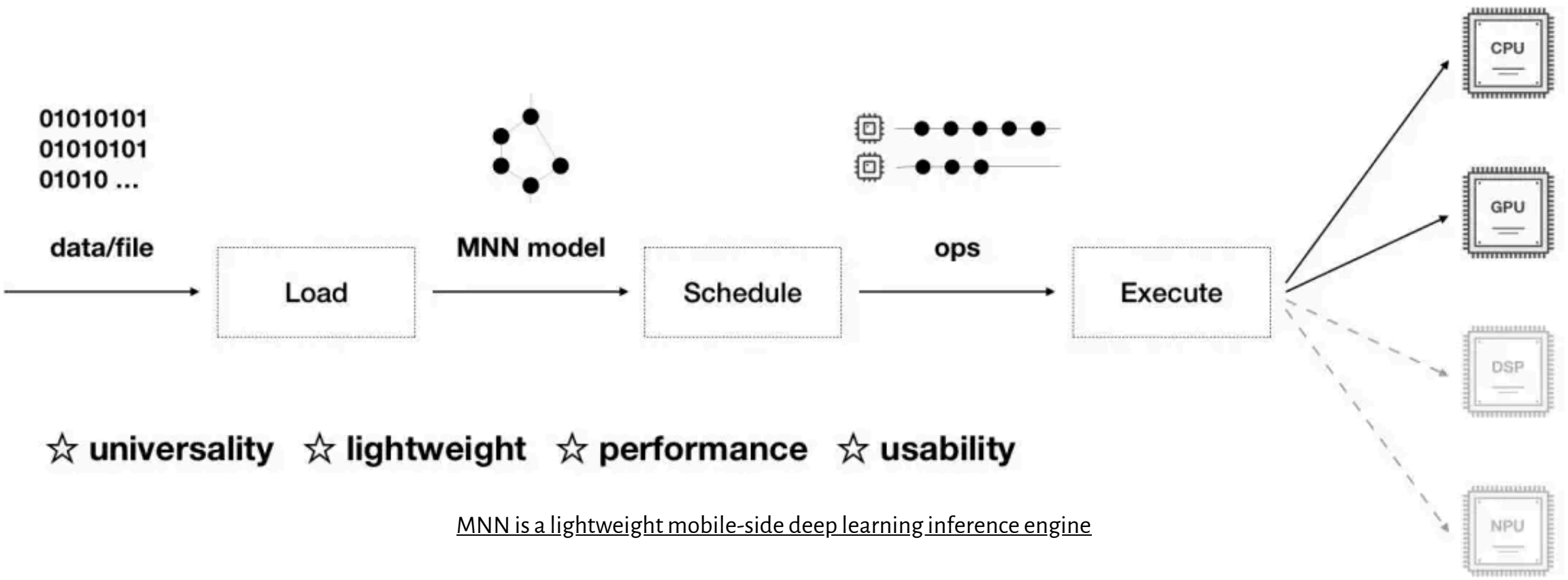
Choosing the Right Model



1. **Pick small, efficient models**: Large models like GPT-4 require too much power. Use lightweight models like Mistral-7B, Phi-2, or Gemma, optimized for mobile use.
2. **Use quantized models**: Quantization reduces model size and speeds up inference. Formats like GGUF and 4-bit int quantization make models run efficiently on phones.
3. **Fine-Tuning for better performance**: Pre-trained models can be optimized for tasks like summarization, and coding. LoRA fine-tuning makes models smaller and faster.
4. **Choose models based on use case**:

- Chatbots & general AI: Mistral-7B (GGUF)
- Code generation: Phi-2 (int4 quantized)

Picking an Inference Engine



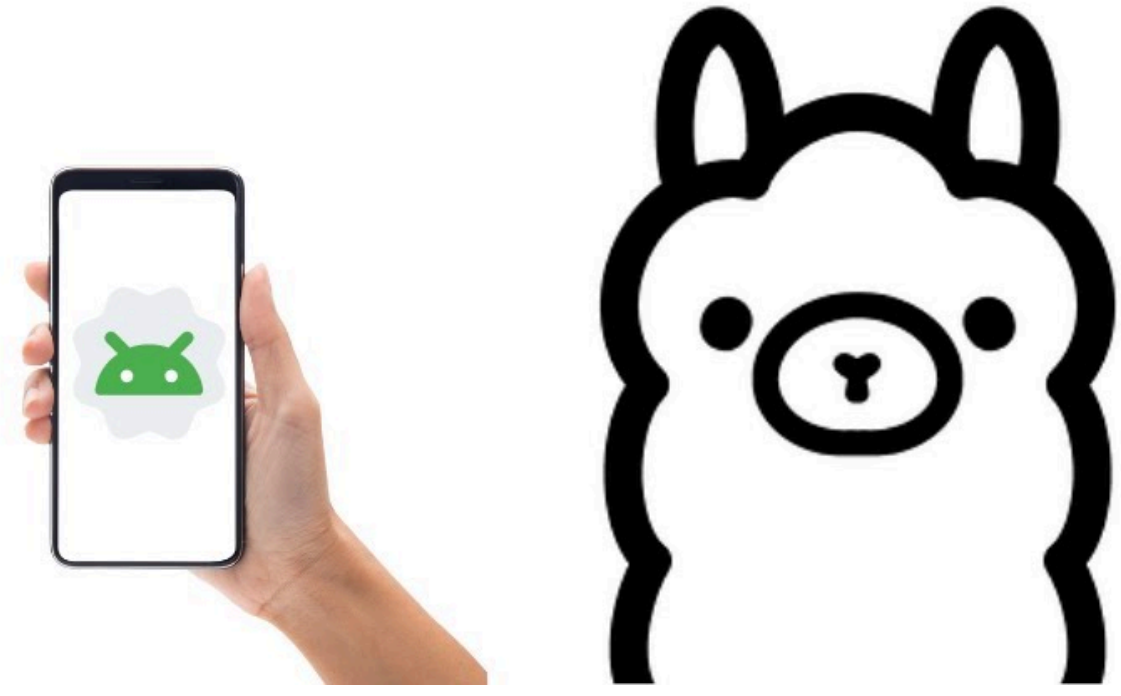
MNN is a lightweight mobile-side deep learning inference engine

1. **Use a lightweight engine for mobile**: General AI frameworks are too heavy for phones. Use llama.cpp (GGUF models), MobiLLM or ONNX Runtime.
2. **Leverage hardware acceleration**: AI inference is slow without acceleration. Use Core ML (Apple), MPS (iOS AI boost), CUDA/TensorRT (NVIDIA for Android), or NNAPI.
3. **Optimize for Low memory usage**: Phones have limited RAM, so use GGUF models with llama.cpp, TFLite , or ONNX Runtime Mobile to reduce memory needs.
4. **Match engine to your model:**

- Apple devices: Use Core ML or MPS for max efficiency.
- Android AI apps: Use TensorRT or TFLite for GPU speed-ups.

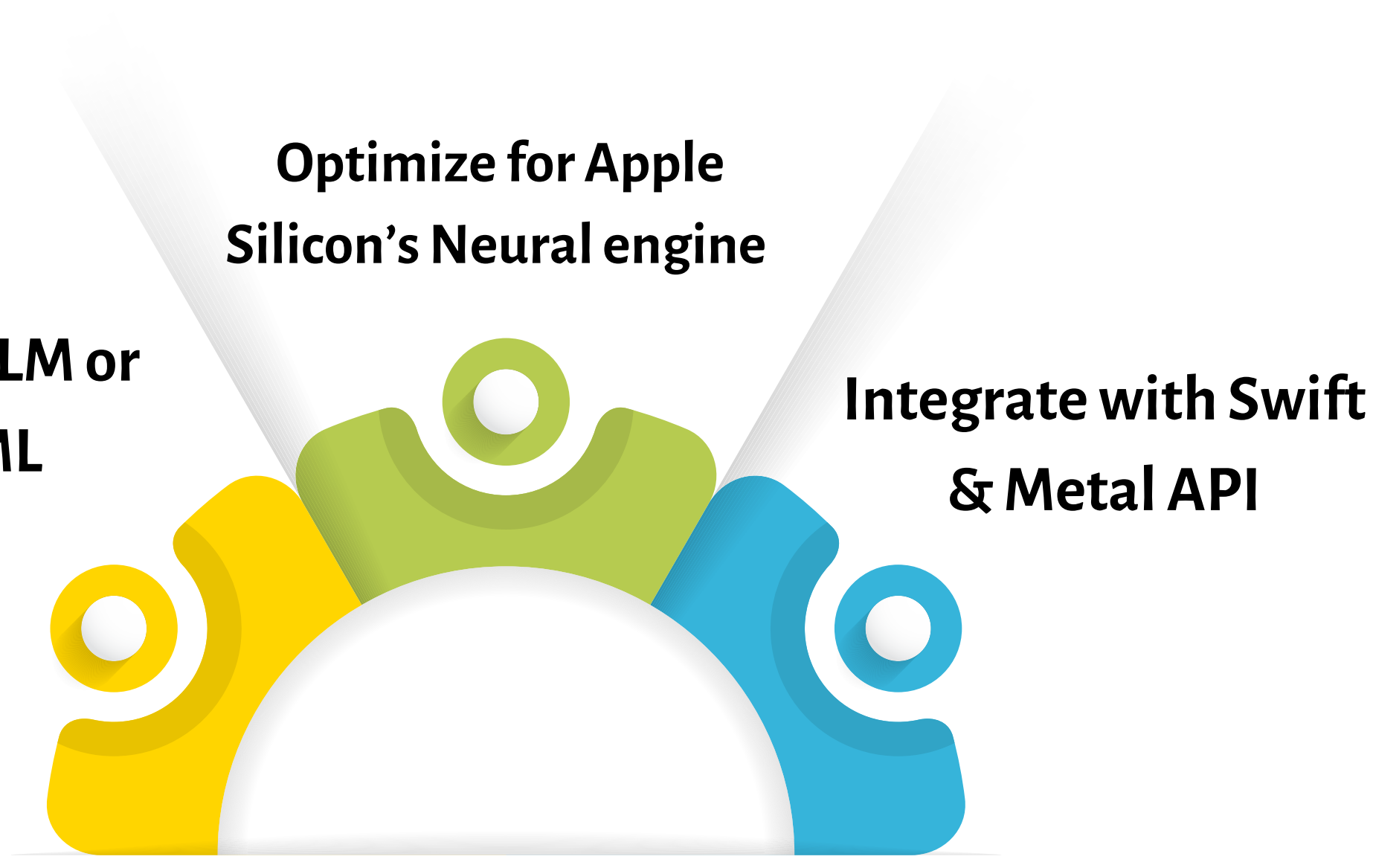
LLMs on Android

Run Offline LLMs on Android



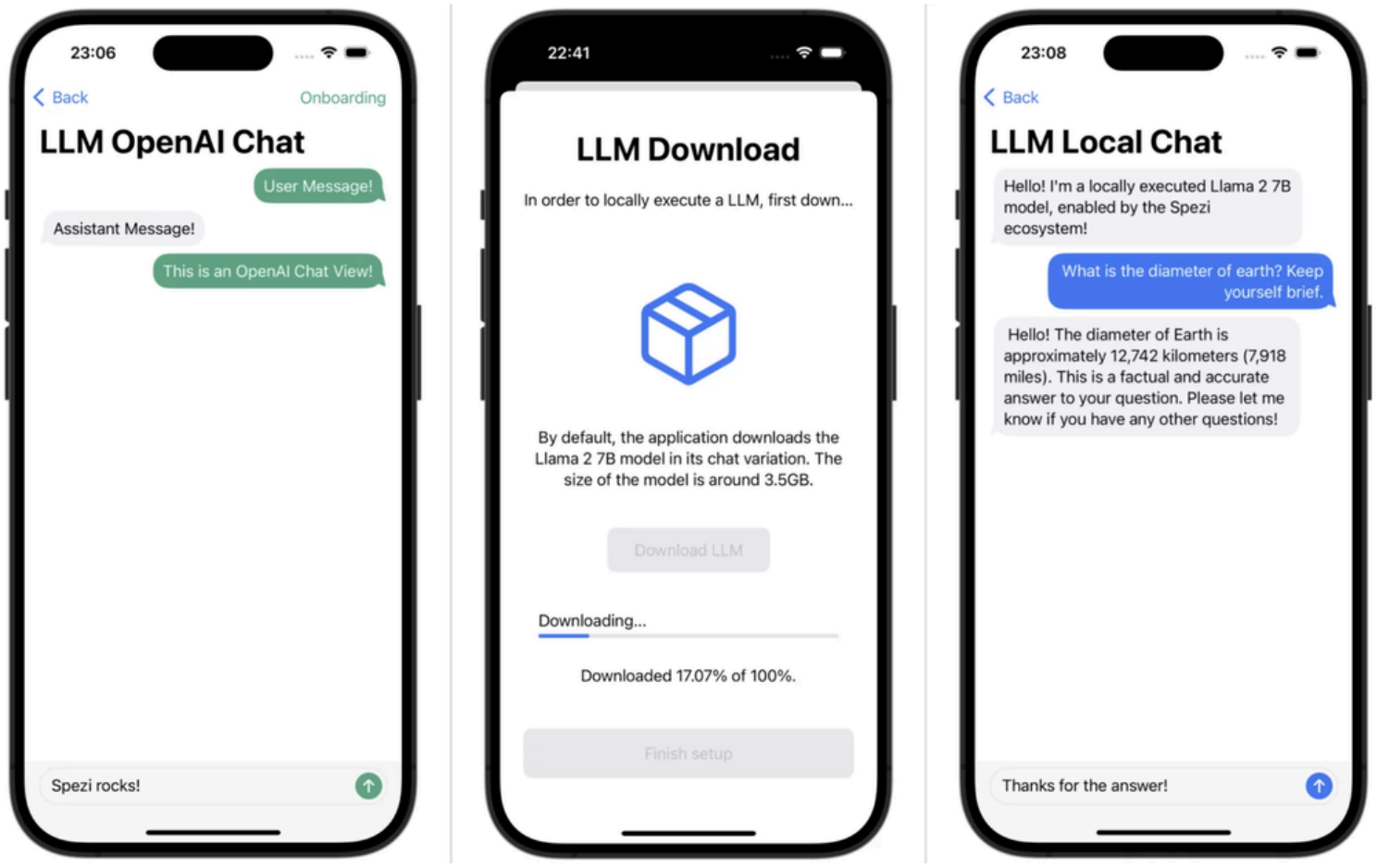
- **Use GGUF models with llama.cpp** – GGUF is a highly optimized format for running LLMs efficiently on mobile devices. llama.cpp enables on-device inference with minimal resource usage.
- **Deploy with Java/Kotlin + JNI bindings** – Since llama.cpp is written in C++, use JNI (Java Native Interface) bindings to call it from Java/Kotlin in Android apps.
- **Convert to TensorFlow Lite for speed** – TensorFlow Lite (TFLite) provides optimized execution for mobile hardware, using NNAPI or GPU acceleration for faster inference.

LLMs on iOS



- **Use MLC LLM or Core ML** – MLC LLM (from MLC AI) and Core ML enable efficient, Apple-optimized LLM execution directly on iOS devices.
- **Optimize for Apple Silicon’s Neural engine** – Apple's ANE (Apple Neural Engine) drastically improves inference speed and reduces battery drain compared to CPU-based execution.
- **Integrate with Swift & Metal API** – Use Swift for app development and Metal Performance Shaders (MPS) to accelerate model execution on the iPhone/iPad GPU.

Building a Simple LLM App



Source

1. **Choose your Tech stack**:
   - Android: Use React Native, Flutter, or native Kotlin for development.
   - iOS: Go with Swift (Core ML), React Native, or Flutter for cross-platform.
2. **Set up your backend:**
   - Local: Run models fully offline using llama.cpp, ONNX Runtime, or Core ML.
   - Server-Based: Host an LLM on a FastAPI/Flask backend if local execution is slow.
3. **Load the LLM in your App**:
   - Android: Use llama.cpp (GGUF models) or TensorFlow Lite for local inference.
   - iOS: Use Core ML models for optimized Apple Neural Engine (ANE) execution.
4. **Add a Chat interface**:
   - Flutter: Use the ChatBubble package for an interactive UI.

Follow to stay updated on Generative AI

LIKE

COMMENT

REPOST