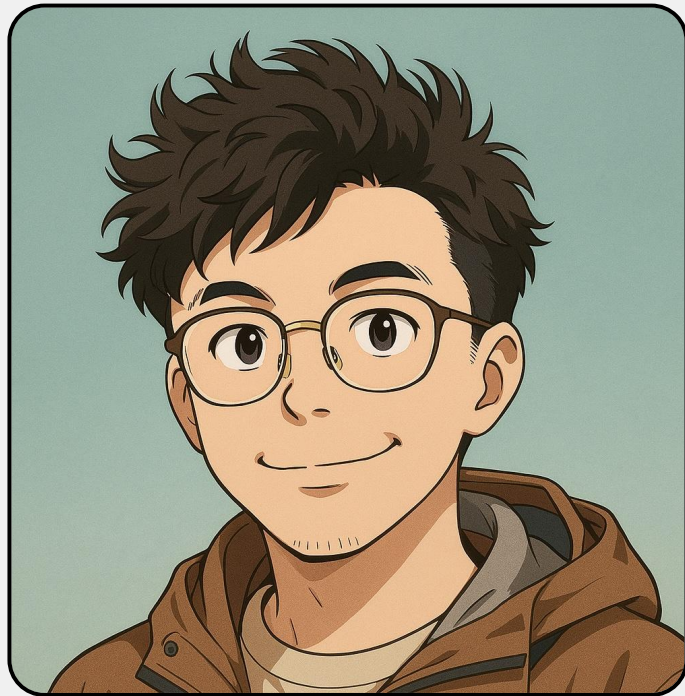


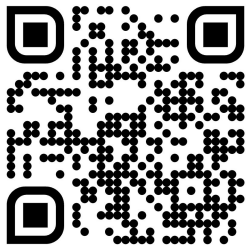
# On-device LLMs & Intelligent Agents on Android

El Zhang (2BAB)

Android GDE / Tech Lead at PixAI



**Build**  **with AI**



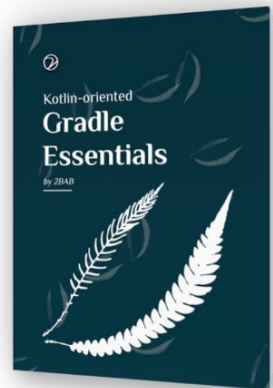
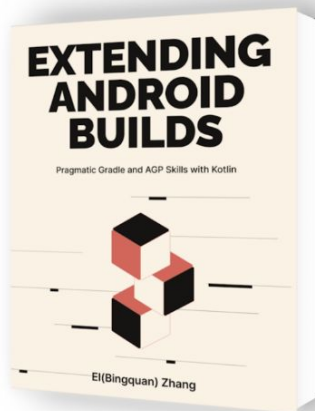
[2bab.com](https://2bab.com)

# EI

- ◆ Google Developer Expert (GDE) for Android
- ◆ 9+ YOE on Mobile Infrastructures
- ◆ Enthusiast of AI, Build Tools (Gradle), Compiler (Kotlin & KSP), Kotlin Multiplatform
- ◆ Pursuer of Elegant UI/UX Effect
- ◆ Residing in Singapore 🇸🇬
- ◆ E-Mail: [xx2bab@gmail.com](mailto:xx2bab@gmail.com)

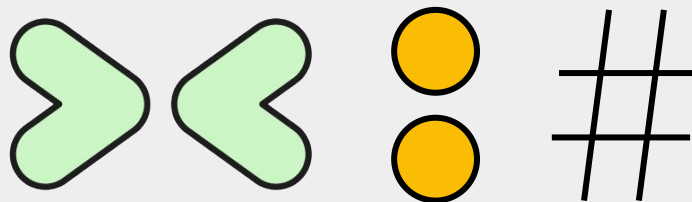
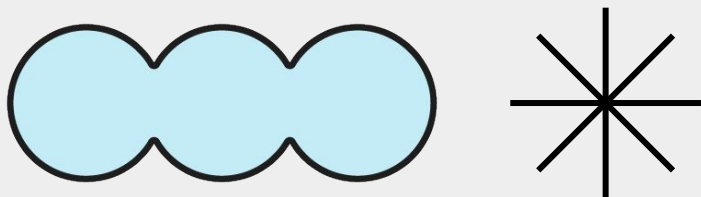
## My Thoughts

Posts / Conference Talks / Podcasts



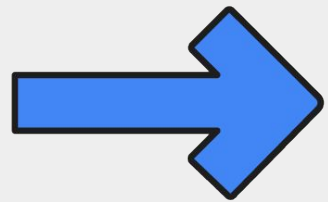
# ToC

1. Recent Advances in On-Device Models on Mobile.



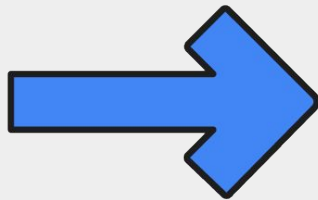
2. A Real-Estate Agent demo from scratch without using any frameworks.

On-Device Models (< 4B)



## Notice

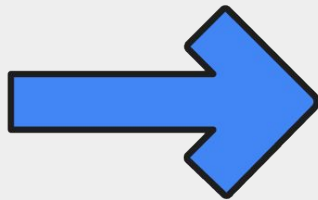
Understand the limitations of small LLMs,  
and leverage their strength - inference  
capability.



## Notice

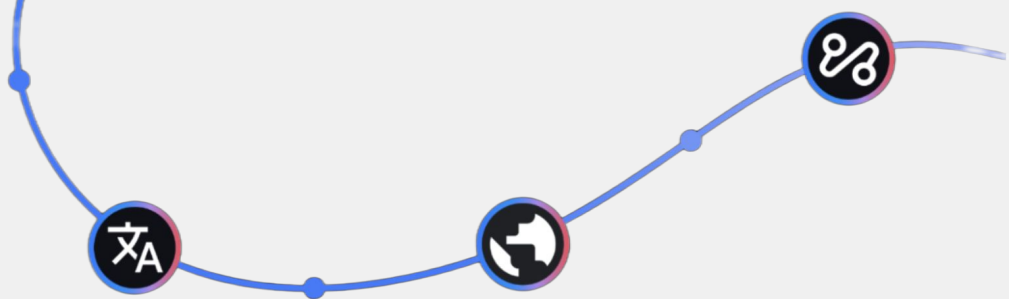
Small LLMs ( $\leq 7B$ ) in production (Cloud) are being applied to tasks such as:

- text classification
- sentiment analysis
- retrieval-augmented generation (RAG)
- quality scoring





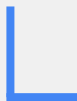
# Gemma3 1B



Small 687M memory  
footprint for q4\_0

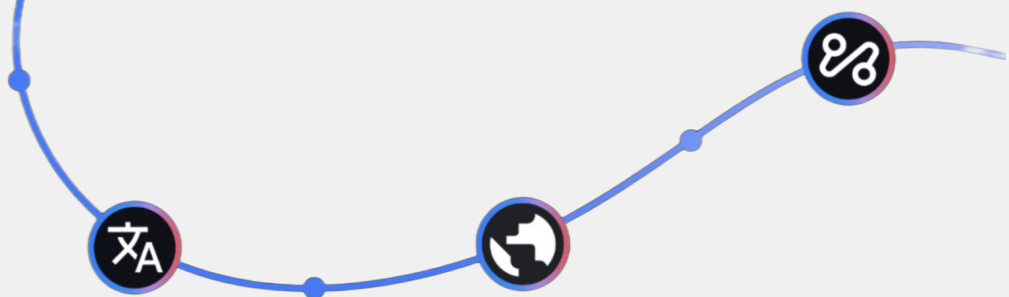
Optimized for strong  
english  
and multilingual  
performance

32k context length





# Gemma3 1B



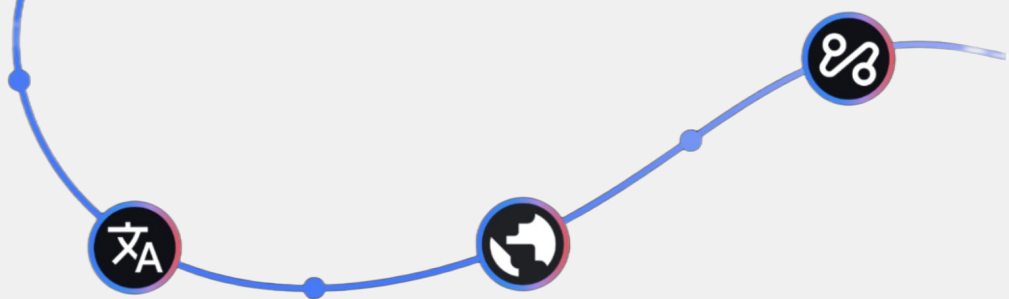
Note that all benchmark stats are from a Samsung S24 Ultra with 2048 KV cache size, 1024 tokens prefill, 256 tokens decode.

Weight Quantization	Backend	Prefill (tokens/sec)	Decode (tokens/sec)	Time to first token (sec)	Model size (MB)	Peak RSS Memory (MB)	GPU Memory (MB)
dynamic_int4	CPU	322.5	47.4	3.1s	529	1138.31	-
dynamic_int4	GPU	2585.9	56.4	4.5s	529	1205.28	585.66





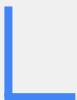
# Gemini Nano 1.8B



Approximately 1.2 GB  
memory usage

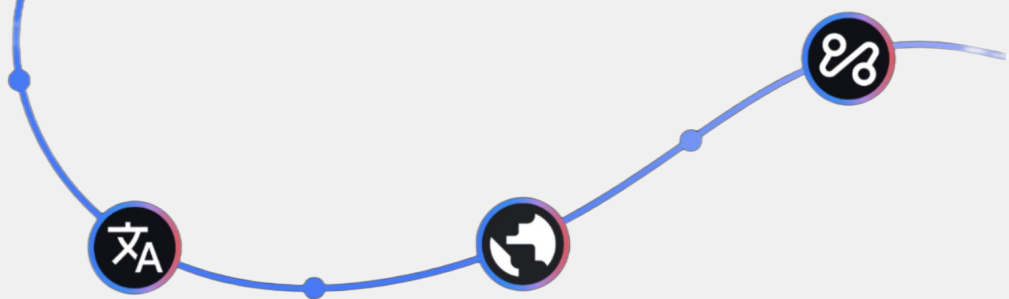
TPU/NPU acceleration

Approximately 4096  
context length






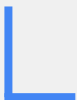
# Gemini Nano 1.8B



Based on AICore 0.0.1-exp01 with Pixel 9

- Decode ~45t/s
- Max Tokens in Decoding is ~250 tokens

 <b>Aicore » 0.0.1-exp02</b> Aicore	
Categories	Android Packages
Tags	google ai aar mobile android
Date	Mar 19, 2025



# Google x On-Device Model

(On Mobile Devices)

Model	Inference Framework	Backend	OpenSource
Gemma (1/2/3) 1B/2B	<b>MediaPipe</b> ( <b>LiteRT</b> + <b>XNNPack</b> )	CPU/GPU	Y
Gemini Nano	<b>AICore</b>	TPU/NPU	N

Actually, MediaPipe LLM Task supports more than one models...




Google Developer Group


 **Models** 5



↑↓ Sort: Recently updated

 `litert-community/Gemma3-1B-IT`


Updated 16 days ago • ❤️ 17

 `litert-community/Gecko-110m-en`

 Question Answering • Updated 17 days ago • ❤️ 1

 `litert-community/Phi-4-mini-instruct`

Updated 23 days ago • ❤️ 1

 `litert-community/Gemma2-2B-IT`

Updated 23 days ago

 `litert-community/DeepSeek-R1-Distill-Qwen...`

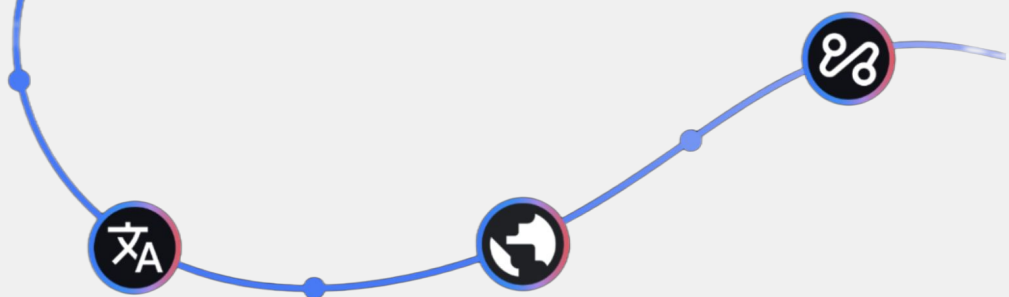
Updated 23 days ago • ❤️ 1

[huggingface.co/litert-community](https://huggingface.co/litert-community)



# Deepseek-R1 1.5B

(distill-qwen)



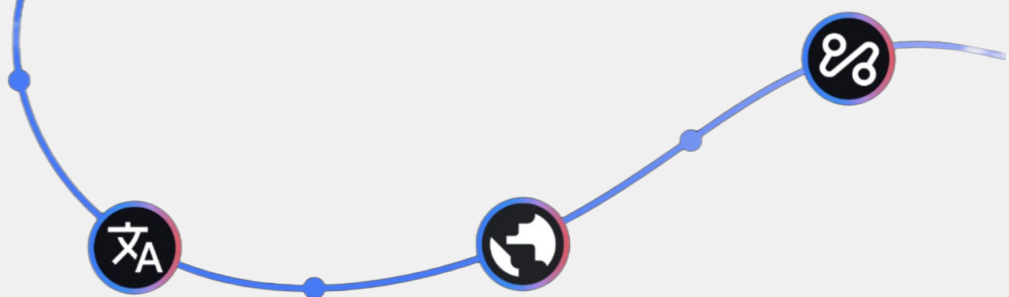
Note that all benchmark stats are from a Samsung S24 Ultra with 1280 KV cache size, 512 tokens prefill, 128 tokens decode.

	Backend	Prefill (tokens/sec)	Decode (tokens/sec)	Time-to-first- token (sec)	Memory (RSS in MB)	Model size (MB)
fp32 (baseline)	CPU	45	6	8	6,213	7,124
dynamic_int8		261	23	2	1,936	1,861

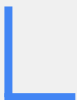


# Gecko 110M en

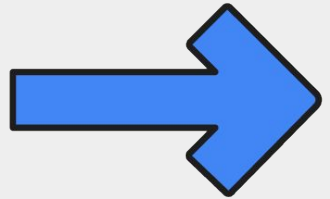
(Embedder Model)



	Backend	Max sequence length	Init time (ms)	Inference time (ms)	Memory (RSS in MB)	Model size (MB)
dynamic_int8	CPU	512	30.04	353.1	225.6	120
dynamic_int8	CPU	1024	79.17	954	619.5	145



Let's take Gemma3 and  
Deepseek as examples

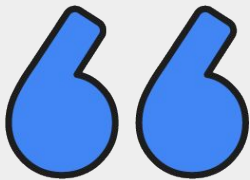


# Gemma3 1B - Best practice: Text generation

A 1B parameter is great at app context -> short text:

- Data Captioning
- Summarization
- Smart Reply/Detection
- Simple In-Game Dialog





8:50 P

MediaPipe LLM Inference

Responses generated by user-provided model

GEMMA\_CPU

X

Can u tell if this is a scam message:  
Your parcel ending 4069 is now at your  
condo's Parcel Santa Locker @ Leeeeee  
Hills, B1 Blk. 17 Behind Carpark Lot 486.  
Please collect by 03/March/2025 09:27  
with OTP code: 674500.

Model

Yes, this message is highly likely a  
scam. Here's why:  
  
\* \*\*Vague Delivery Information:\*\* The  
message gives a general location  
(Leeeeee Hills) and a complex address  
without proper street and building  
names. Real delivery services would  
have official, clear-cut addresses.  
\* \*\*Unreliable and Inconsistent  
Language:\*\* Phrases like "Parcel Santa  
Locker" suggest a fictional and  
overly-casual approach to a formal  
shipping process.  
\* \*\*Unsolicited OTP Code:\*\* You are  
being asked for an OTP code that's

8:52 P

MediaPipe LLM Inference

Responses generated by user-provided model

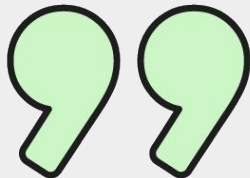
DEEPSEEK\_CPU

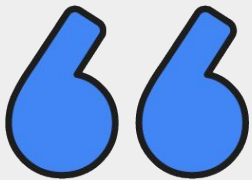
X

Can u tell if this is a scam message:  
Your parcel ending 4069 is now at your  
condo's Parcel Santa Locker @ Leeeeee  
Hills, B1 Blk. 17 Behind Carpark Lot 486.  
Please collect by 03/March/2025 09:27  
with OTP code: 674500.

Model

Okay, so I need to figure out if this  
message is a scam. Let me read it again  
carefully.  
  
The message says: "Your parcel ending  
4069 is now at your condo's parcel Santa  
Locker @ Leeeeee Hills, B1 Blk. 17 Behind  
Carpark Lot 486. Please collect by 03/  
March/2025 09:27 with OTP code:  
674500."  
  
Hmm, first, I notice the address includes  
a lot of abbreviations and symbols like  
"Leeeeee Hills" and "Carpark Lot 486."  
Those might be typos or abbreviations  
for longer words. For example, "Leeeeee





## MediaPipe LLM Inference

Responses generated by user-provided model

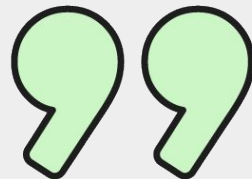
DEEPSEEK\_CPU



Message

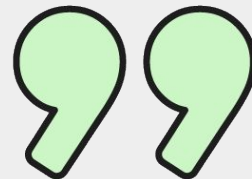
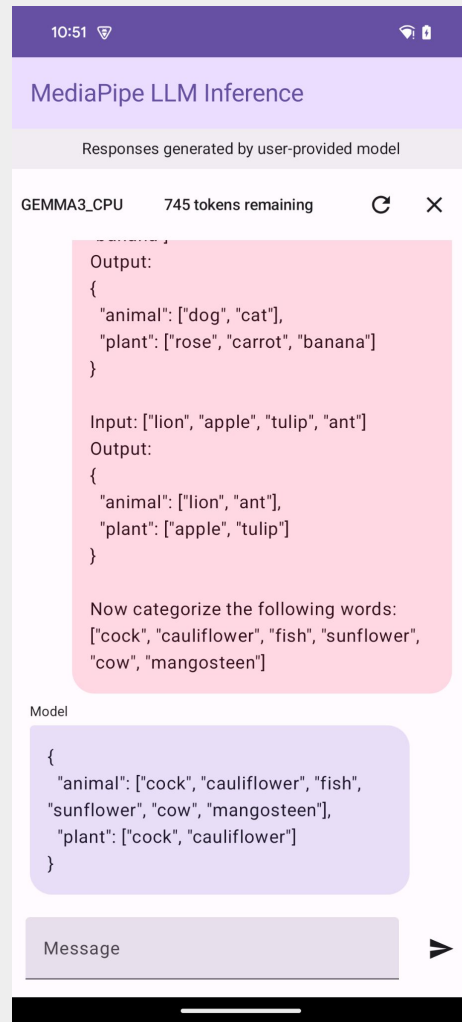
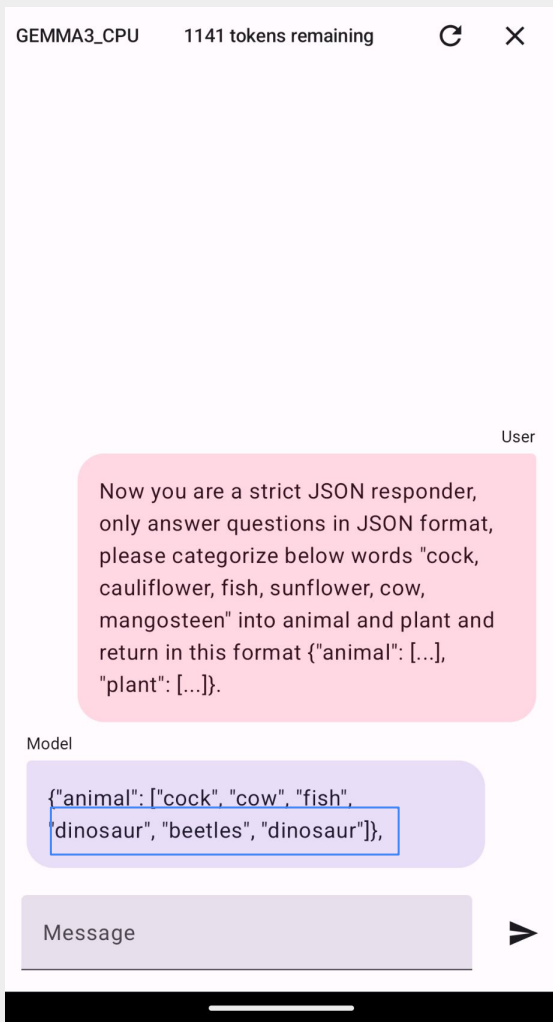
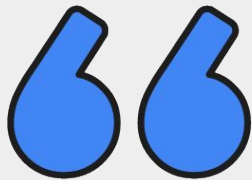
帮我检查者这是不是 fishing:

<Singtel>Dear customer, your latest bill for Singtel a/c 72888888 is ready. The total amount is SGD\$56.29 due on 20 Mar 2025. You can view and pay this bill via My Singtel app at [www.singtel.com/viewbill](http://www.singtel.com/viewbill)

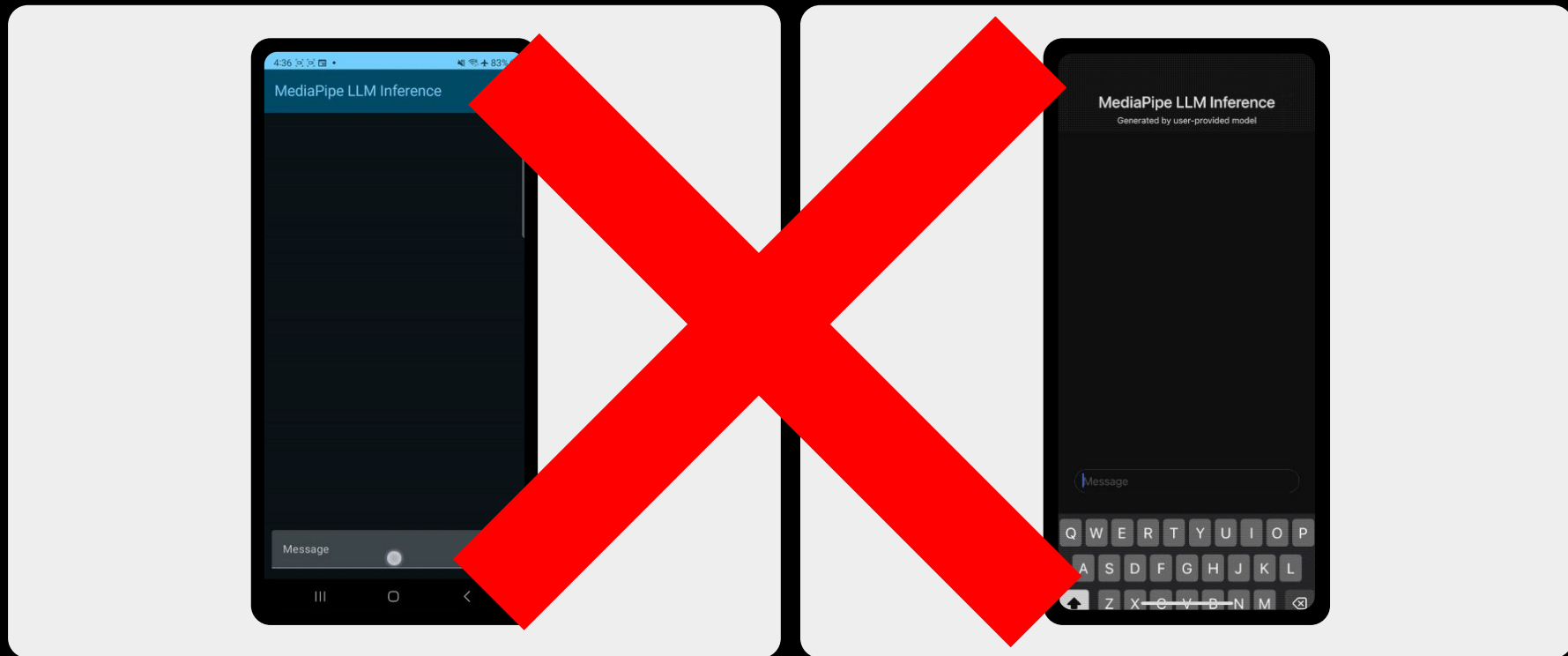


# Best practice: Customize Gemma3 1B

- Few shot examples
- Fine-tune (LoRA or SFT)
- Use Retrieval Augmented Generation (RAG) to consume more content

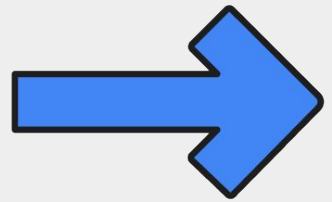


# Worst practice: Gemma3 1B as chatbot



Results for illustrative purpose

The Agent





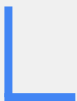
# Limitations



- When using the MediaPipe LLM Task, a **context window** of around **1500-2000 tokens** is optimal.
- With small models, **limited attention spans** requires **concise prompts**, even with effective prompt engineering.
- Without fine-tuning, achieving ReAct-style **reasoning** with pure prompts is **challenging**.

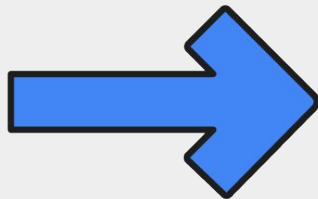
Note that all benchmark stats are from a Samsung S24 Ultra with 2048 KV cache size, 1024 tokens prefill, 256 tokens decode.

(Google applied 2048 for testing)

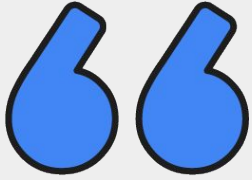


## What is the Agent?

1. *Autonomous decision-making*: planning and executing multi-step tasks independently
2. *Tool utilization*: leveraging external APIs, web crawlers & databases to extend capabilities
3. *Environmental interaction*: operating with real-world systems including file manipulation and code execution
4. *Feedback processing*: self-evaluating results and adjusting strategies accordingly
5. *Memory management*: maintaining context coherence and task progress across multiple interactions
6. ...

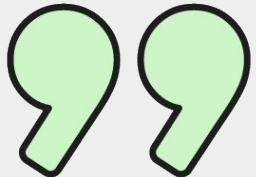


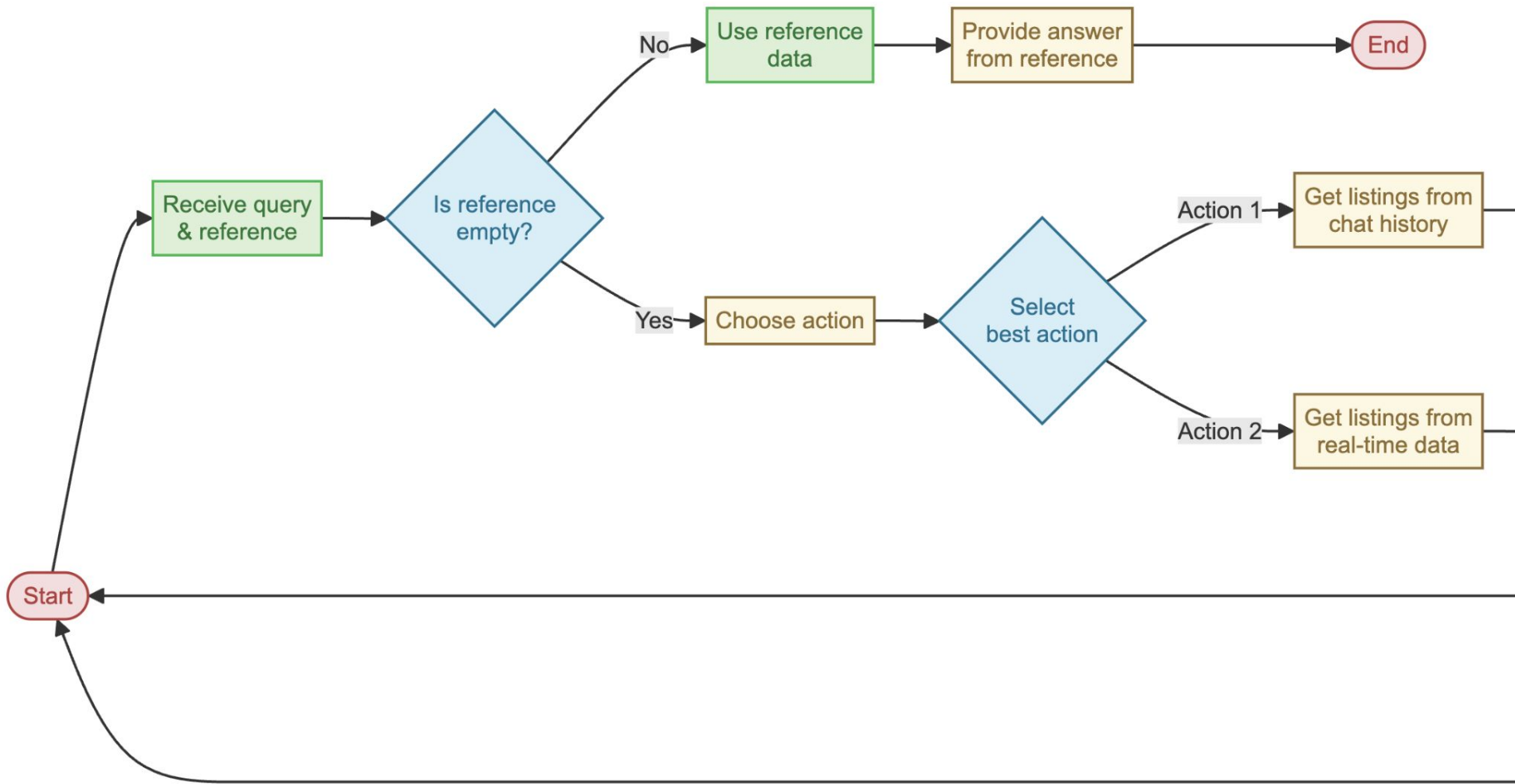


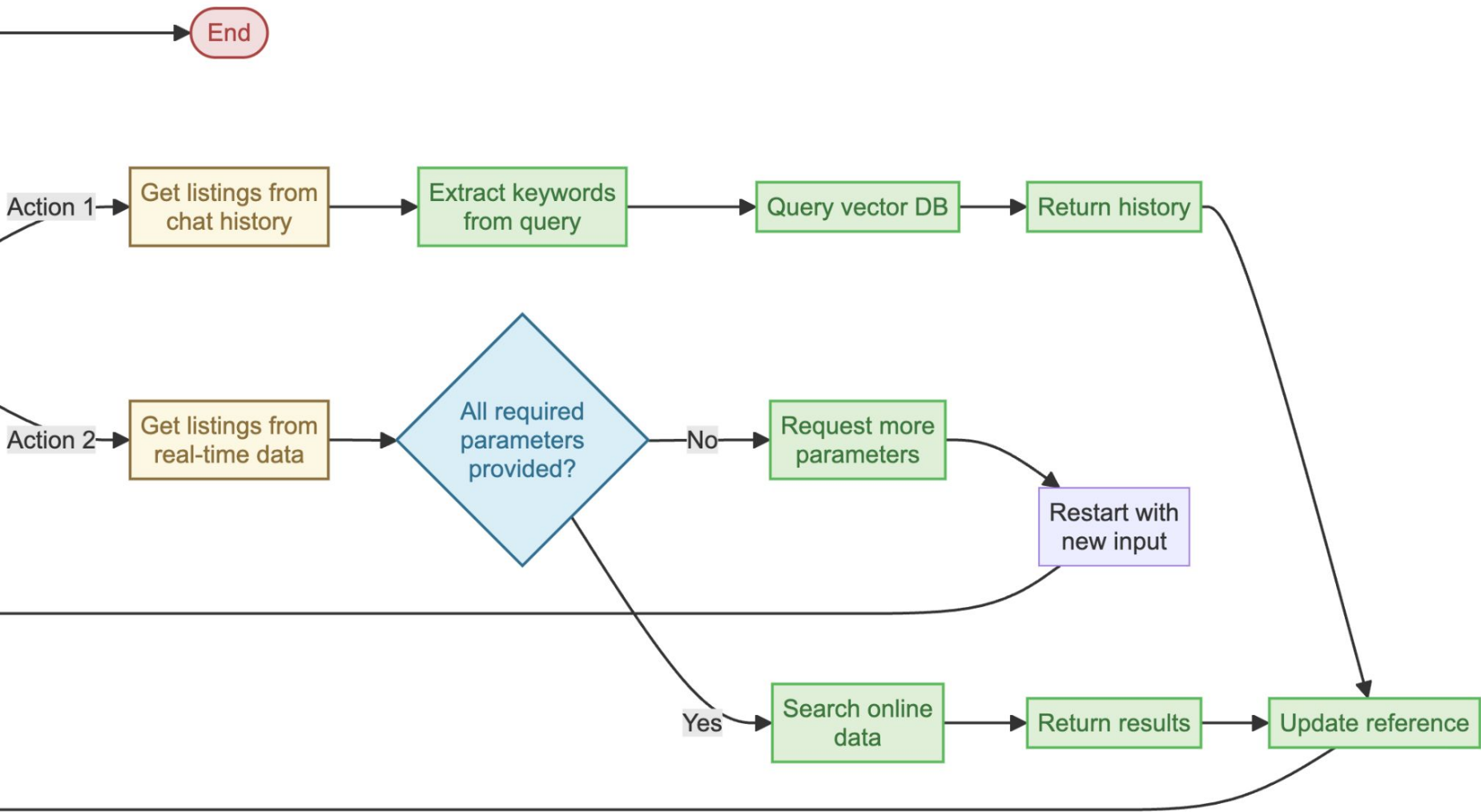


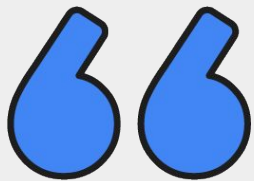
You're a real estate agent with access to a property search tool. This tool requires location, property type, and price as mandatory parameters, with bedrooms as optional. Follow the below process:

1. Receive user query
2. If any required parameters are missing, inform the user
3. With all required parameters, make an API call: `{"action": "search", ...}`
4. Process API results: `[{"name": "propertyA", "subway": "0.5", "price": "600k" ...}]`
5. Analyze each property for proximity to subway, price value, etc.
6. Rate properties and generate a report









'ReAct' style prompt (lite ver.), to achieve Prompt Chaining

You are a professional real estate agent ...

Settle down basics

You are given:

<question>...</question>

Using special marks to split

<ref>...</ref>

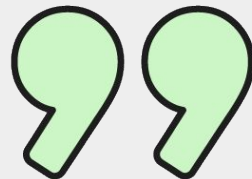
Please output with below:

Explicit thinking process can

<think>..if ref is not empty.. if it is..</think>

improve reasoning ability

<answer>...<answer>



# OnDevice Agent Example

A prototype Android application demonstrating an on-device LLM agent for property search. The app uses MediaPipe for LLM inference and implements a chat-like interface for user interactions.

## 🔗 Features

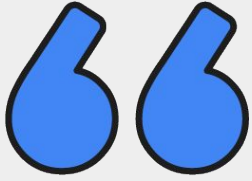
- On-device LLM inference using MediaPipe
- Property search functionality
- Chat-like interface with:
  - Message types (System, User, Function, Result)
  - Message animations
  - Auto-scrolling to latest messages
  - Expandable message content
  - Clear conversation history
- Material 3 design with custom styling

## Tech Stack

- Kotlin
- Jetpack Compose
- MediaPipe Tasks for LLM inference (Gemma3 1B/DS 1.5B)
- Koin for dependency injection
- Ktor for network requests
- Kotlin Coroutines & Flow

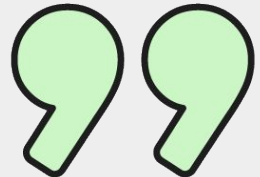


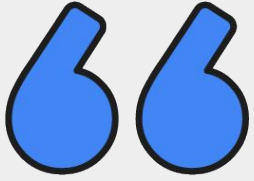
<https://github.com/2BAB/OnDeviceAgent>



1. *Role prompting*: “You are a professional real estate agent...”
2. *Explicit instructions*: “Carefully extract ... without including ANY numerical details.”
3. *Instruction reinforcement*: “(NO numbers at all!)”
4. *Few-shot prompting*: “here are 3 examples”
5. *Task decomposition*
6. *Output constraints*: “in JSON format strictly”
7. ...

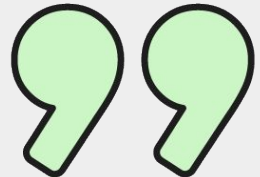
<https://www.promptingguide.ai/>





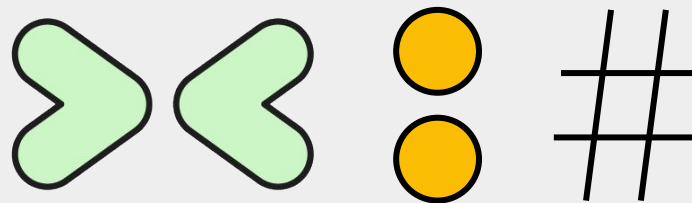
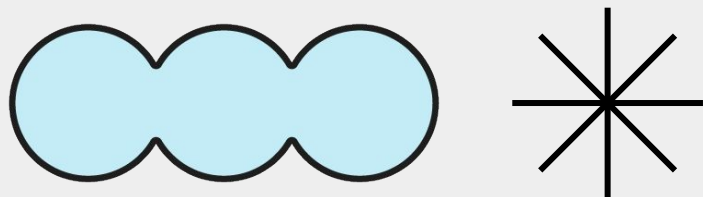
Combine with Android APIs to setup Schedule Tasks

1. WorkManager (~10 mins)
2. Foreground Service (as long as the app is active)
3. Widgets (< 5mins)



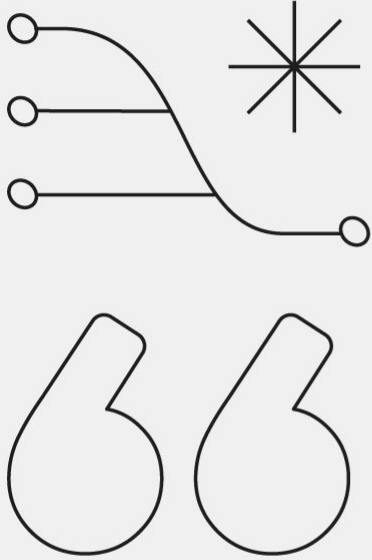
# Summary

1. Advancements in models, hardware, system-level model sharing, and open data protocols will create new opportunities.



2. OEM and IoT companies can lead the early adoption of on-device LLMs within specialized industries.

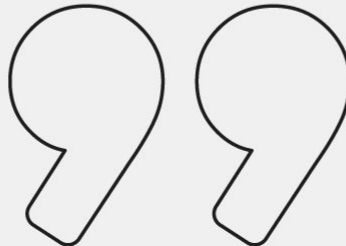
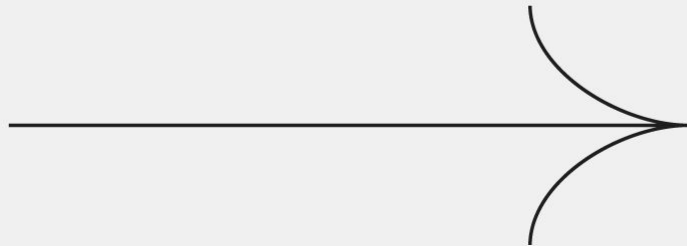




**Thank you!**



**Q&A**



**{ Build  with AI }**