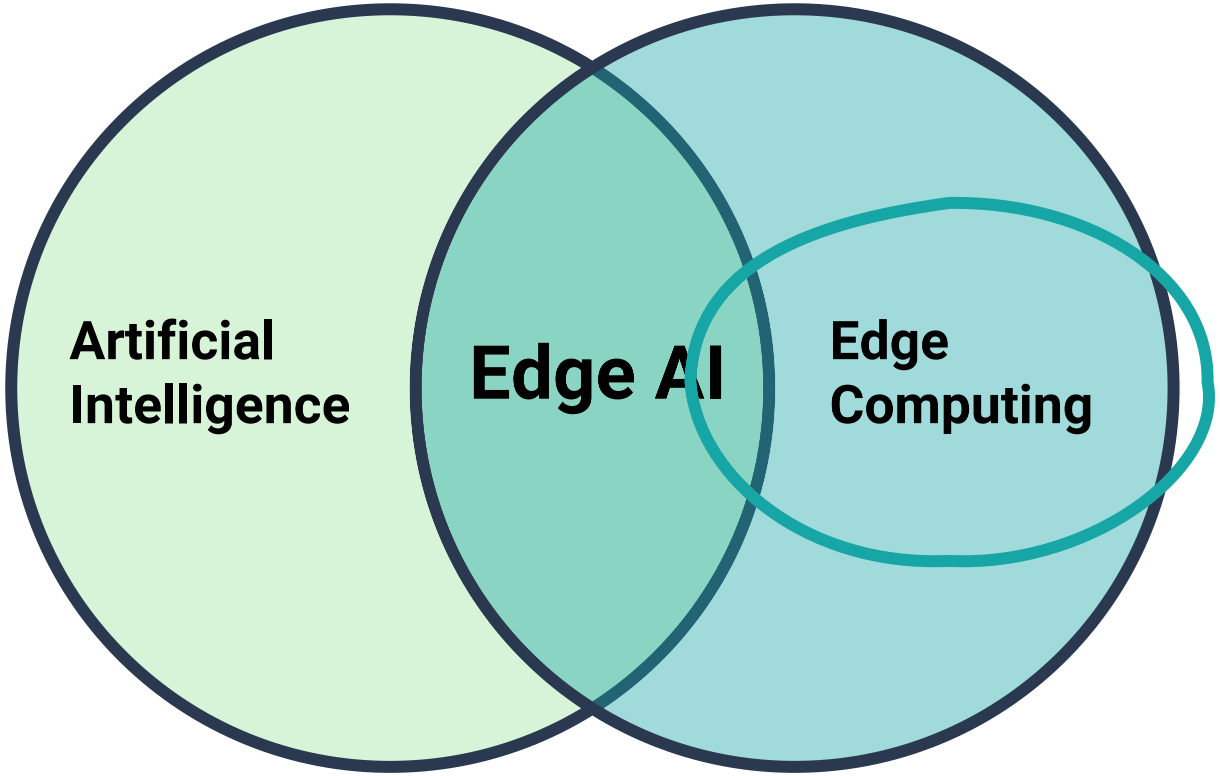
AI anywhere | anytime

The promise of Edge AI…

but...
is it already relevant for you?

Edge AI: Two megatrends converging: Edge Computing + AI



Data is being created everywhere

„Locally“

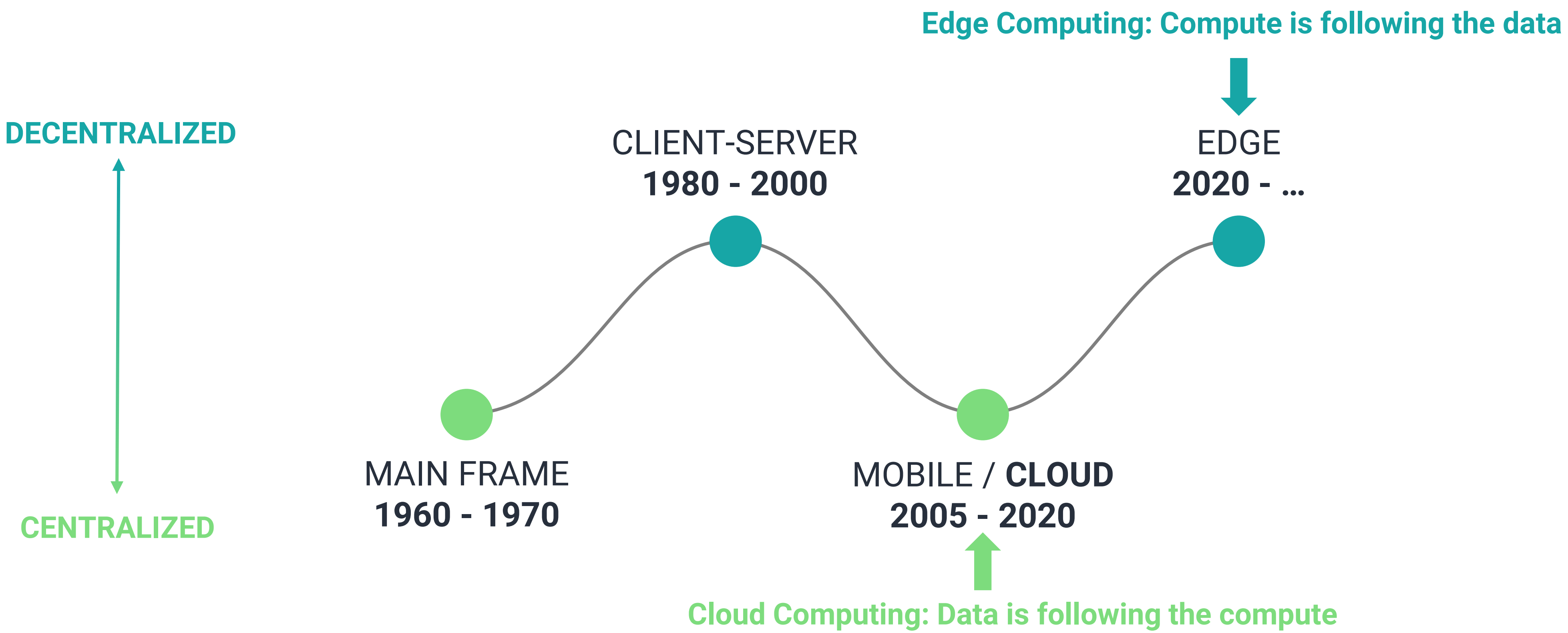
„On the Edge“

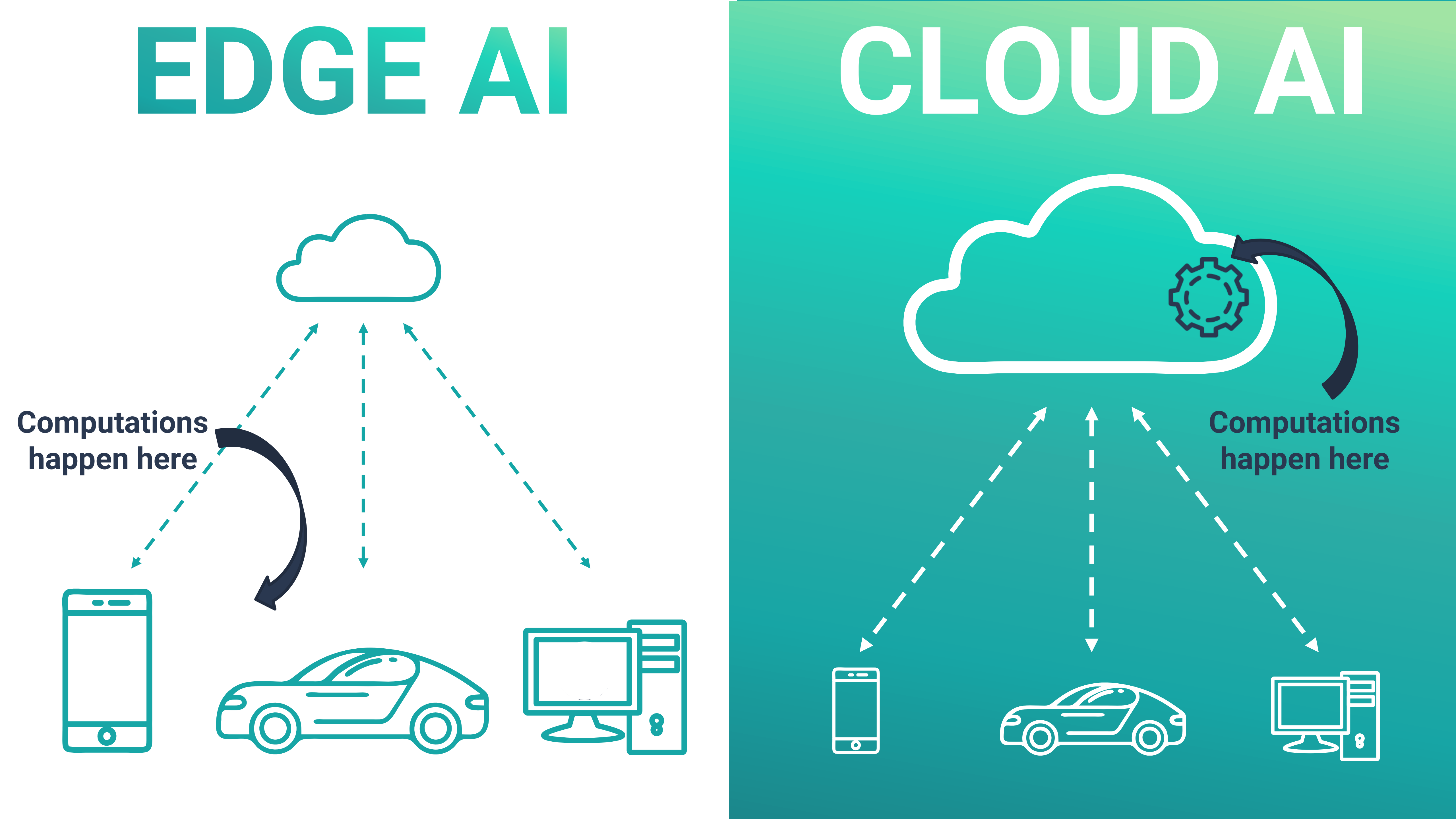
„On-device“

When we process data where it is created
(locally, on-device, on the edge vs. in a distant cloud)

Edge Computing

Just a “business” term for a decentralized computing paradigm




EDGE AI
CLOUD AI
Computations happen here
Computations happen here

Why Edge AI?

Works Offline

Privacy

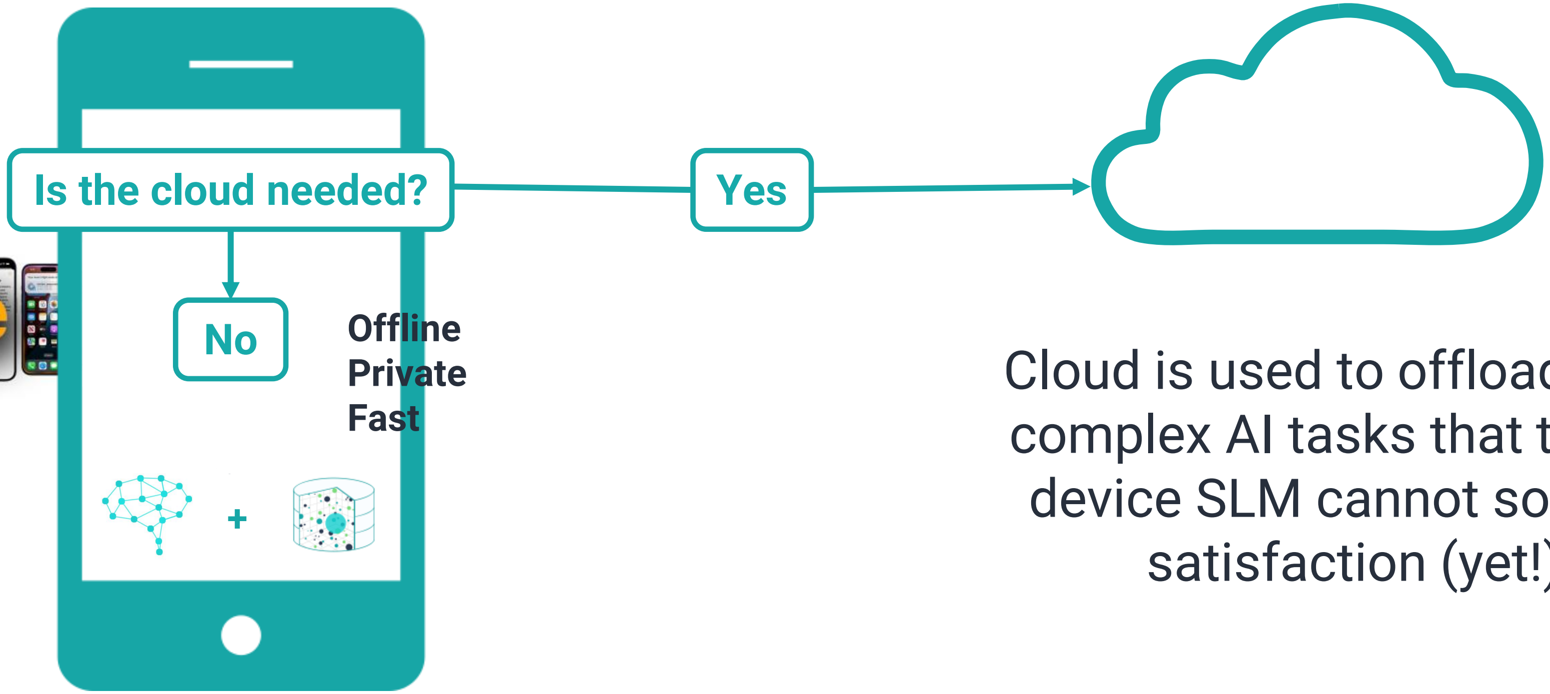
No Bandwidth Bottleneck

Realtime possible

Less costs

More sustainable

The reality is "hybrid": best of both worlds



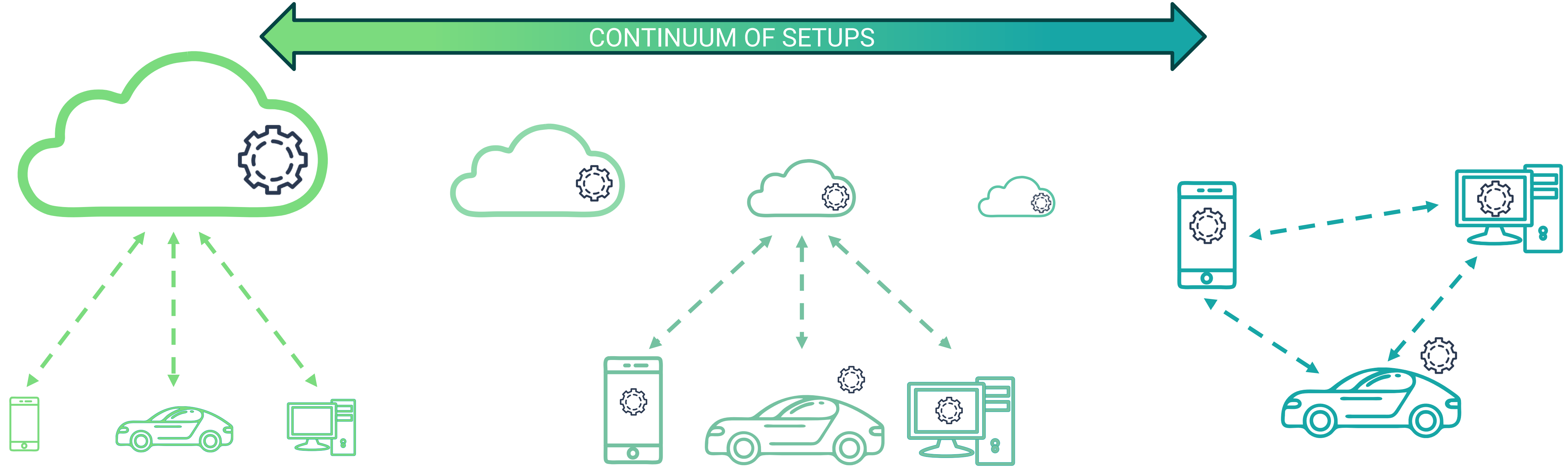
(MicroNN)

Cloud is used to offload more complex AI tasks that the on-device SLM cannot solve to satisfaction (yet!)

Adapted from https://www.edge-ai-vision.com/2023/05/the-future-of-ai-is-hybrid-how-on-device-ai-is-enabling-generative-ai-to-scale/

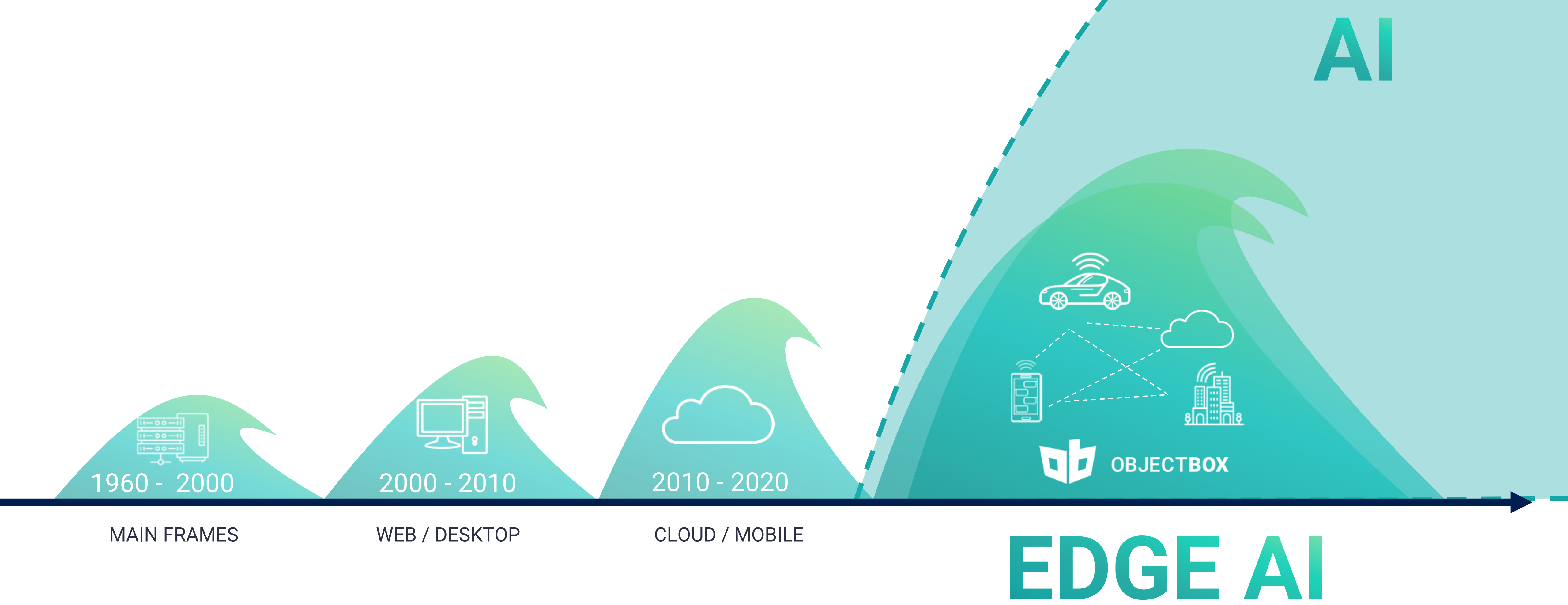
The question is: How much edge / cloud do you need?

++	← Cloud & networking costs →	--
++	← Energy consumption →	--
--	← Offline-Ability, Speed, Performance →	++
--	← Privacy and Security →	++

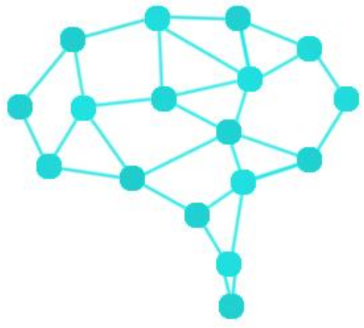


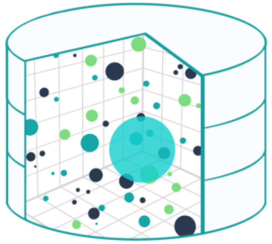
Cloud AI ← **Hybrid AI** → **On device AI**

Every **Megashift in Computing** is **empowered** by enabling **core technologies**



Two core technologies + Sync for hybrid AI apps





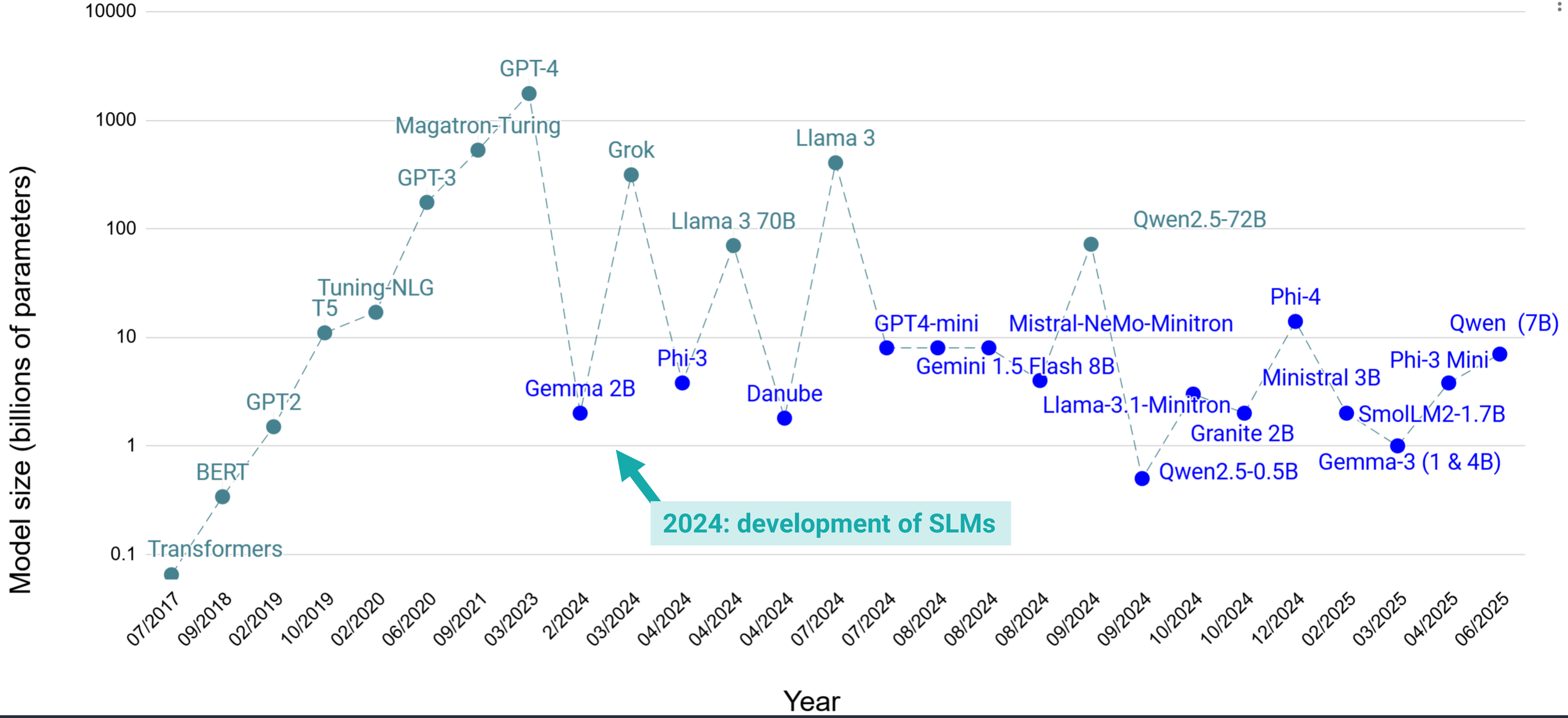
On-device (Local / Edge)
AI Models
(LLMs / SLMs)

On-device (Local / Edge)
Vector Databases
(Semantic Index, Vector Search)

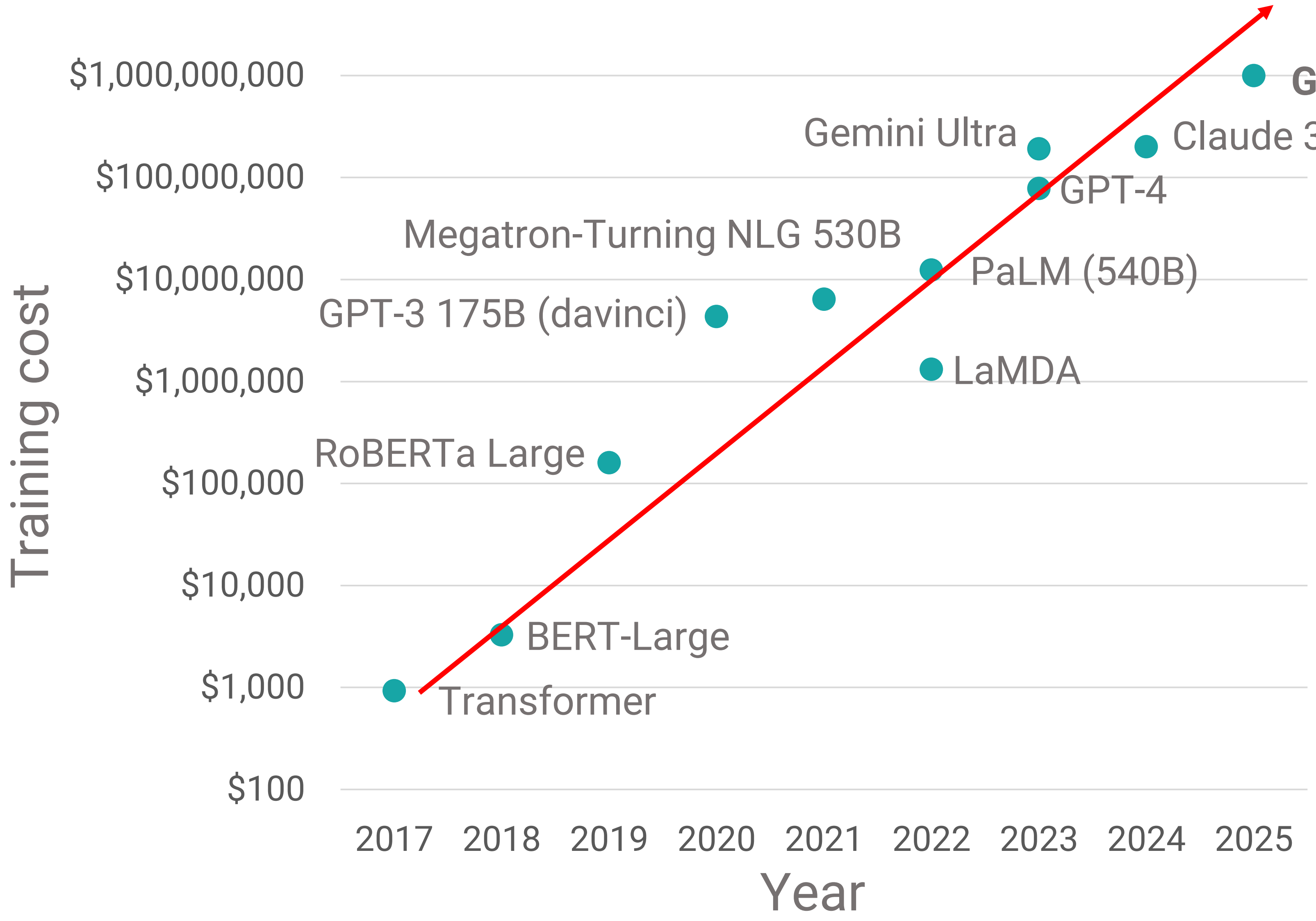
Data Sync

On-device AI models
((Small) Language Models)

The rise of small language models (SLMs)



Frontier AI model training costs (cloud…) are skyrocketing



Frontier AI model training costs (→ cloud costs) have **skyrocketed** since 2020

Power consumption too…

Data from Epoch 2023 [https://epoch.ai/blog/how-much-does-it-cost-to-train-frontier-ai-models], updated with data from Epoch AI 2025

Estimated **Loss** in Market Value **due to Cloud Impact on Margins**

$100B

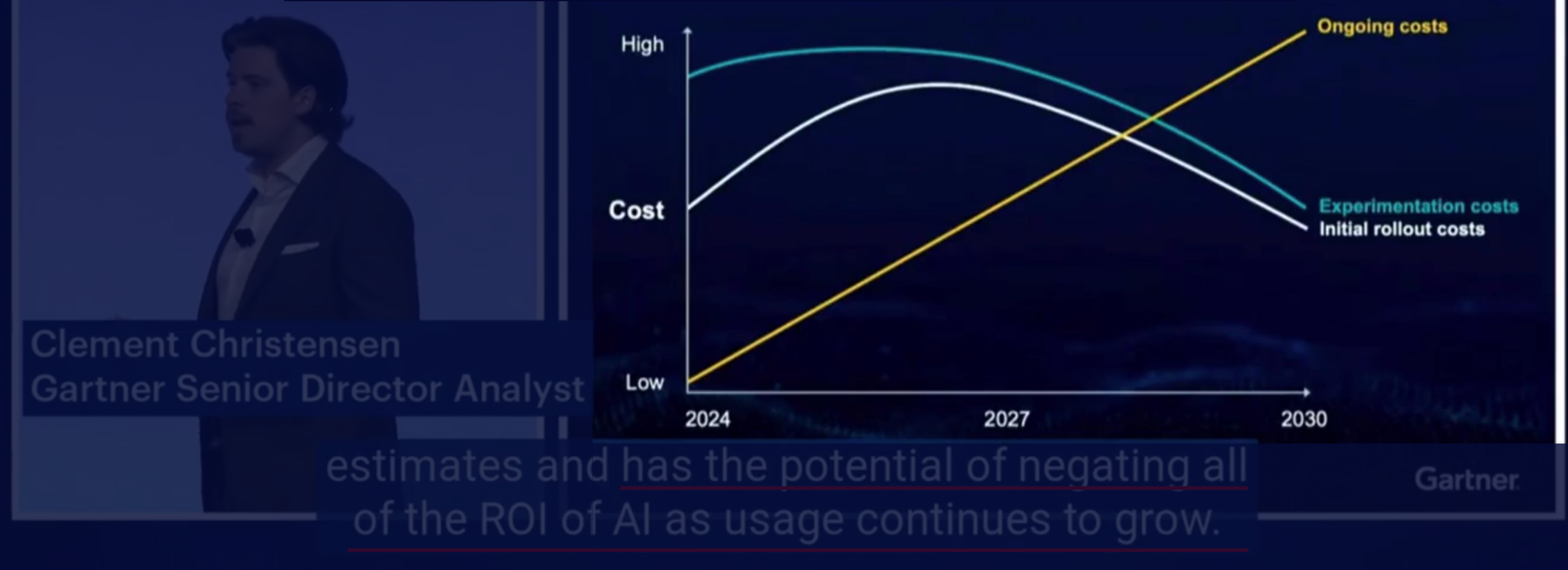
across 50 of the top public software companies

$500B

across all public companies

Andresseen Horrowitz (2021): The cost of the cloud: A trillion dollar paradox

The cost of using AI has the potential of
negating all of the ROI of AI as usage continues to grow
[Gartner (2025)]



Cloud AI is driving **exploding demand for data center capacity**

+20% data center capacity is needed every year to satisfy the demand

300GW projected demand for data center capacity by 2030

70% of this will be for **AI workloads** (Cloud AI, hosted)

McKinsey (2024): https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/ai-power-expanding-data-center-capacity-to-meet-growing-demand

Why this matters for you (as a developer)

- Only few can train („few tech giants hold all AI models“)
  → not your playground
  → building on them is a relevant dependency
- S.o. needs to pay the bill: AI and Cloud costs add up…
  → how can you create margins?
- Most thin layer apps won‘t work longterm
  → how can you create defendable market value?
- Privacy, data security, compliance
  → sending data around always adds risk
  → sharing data too - what are you agreeing to?
- Sustainability – if you care
  → sending data around unnecessarily consumes more energy (and CO2) → and with billions of edge devices this adds up!

But what about open source? Free as in freedom or free beer?

A valid alternative for sure, however…

- Check what has been open sourced
  - Model weights?
  - Training code?
  - Training data (or exact recipe)?
- Check the license (is it really a (permissive) open source license?)
- If your using a „blackbox“ model, you might be
  - liable for copyright infringement
  - liable for accidental disclosure of sensitive data
  - facing compliance issues (EU Act, special requirements in e.g. finance, healthcare etc.)
  - facing unwanted / biased output

For Mobile: Often you want specialized models

- **Small Language (General) Models**, e.g. Phi-3 Mini (3.8B), Gemma 2B, Mistral 7B, Qwen 7B, SmolLM 1–2B, will fit well when **quantized**
- 1 bill+ parameters – while comparitively small – is already a consideration on mobile **e.g. 1B params:**
  - Storage (weights only): FP32 ≈ 4.0 GB, FP16 ≈ 2.0 GB, INT8 ≈ 1.0 GB, INT4 ≈ 0.5 GB
  - With pruning / weight sharing < 0.5 GB (more accuracy loss)
  - Runtime RAM ca. 1.5 – 3X storage, e.g. for 1B INT4 model ≈ 0.7–1.1 GB (weights + cache).
- **Instead often: Specialized models** (speech, vision, translation, text classification) → smaller, faster and better for their task → typically: More than one
- **Latency**: Smaller models can respond in near-real time on device CPUs/NPUs
- **Battery**: Smaller models → reduced compute cost, less battery drain

Fine tuning can well be needed

- **Training (from scratch)**
  - start with random weights
  - needs tons of data/compute → time-consuming, costly
    → only for specific cases

- **Fine-tuning**
  - start from **pretrained models** (e.g. MobileBERT, already “understands” language)
  - Add a small head (or format outputs)
  - train it on labeled data, e.g. DBpedia dataset (ready-made), or your own

- **“Linear probing”**
  - **Freeze base model, train only the** classifier layer
    → faster, less accurate

Advantages of Small Language Models

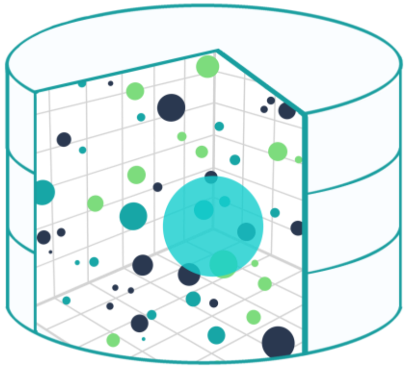
More efficient → faster and cheaper

Better Accessibility (less resource-hungry, no cloud limitations / restrictions)

Cheaper and more sustainable
- in training & in use!

Specialization and Fine-tuning is more efficient (ideal for domain-specific tasks)

Can run on-device → Heightened Privacy & Security, no dependence on the cloud / a network necessary



(On-Device)
Vector Databases

What are vector databases? In a nutshell

- Vector databases = “AI databases”
- AI models use vector embeddings
- Vector databases store vector embeddings
    - Powerful vector search and querying capabilities
    - Add. context and filtering mechanisms
    - Longterm memory

Vector Databases Uses

Speeding up LLM responses
Vector databases use various techniques to speed up the responses, e.g. by using compression and filtering

Enhancing LLMs reponses, e.g. RAG
With a vector database you have additional knowledge to enhance the responses and decrease hallucinations; real-time updates to knowledge becomes possible.

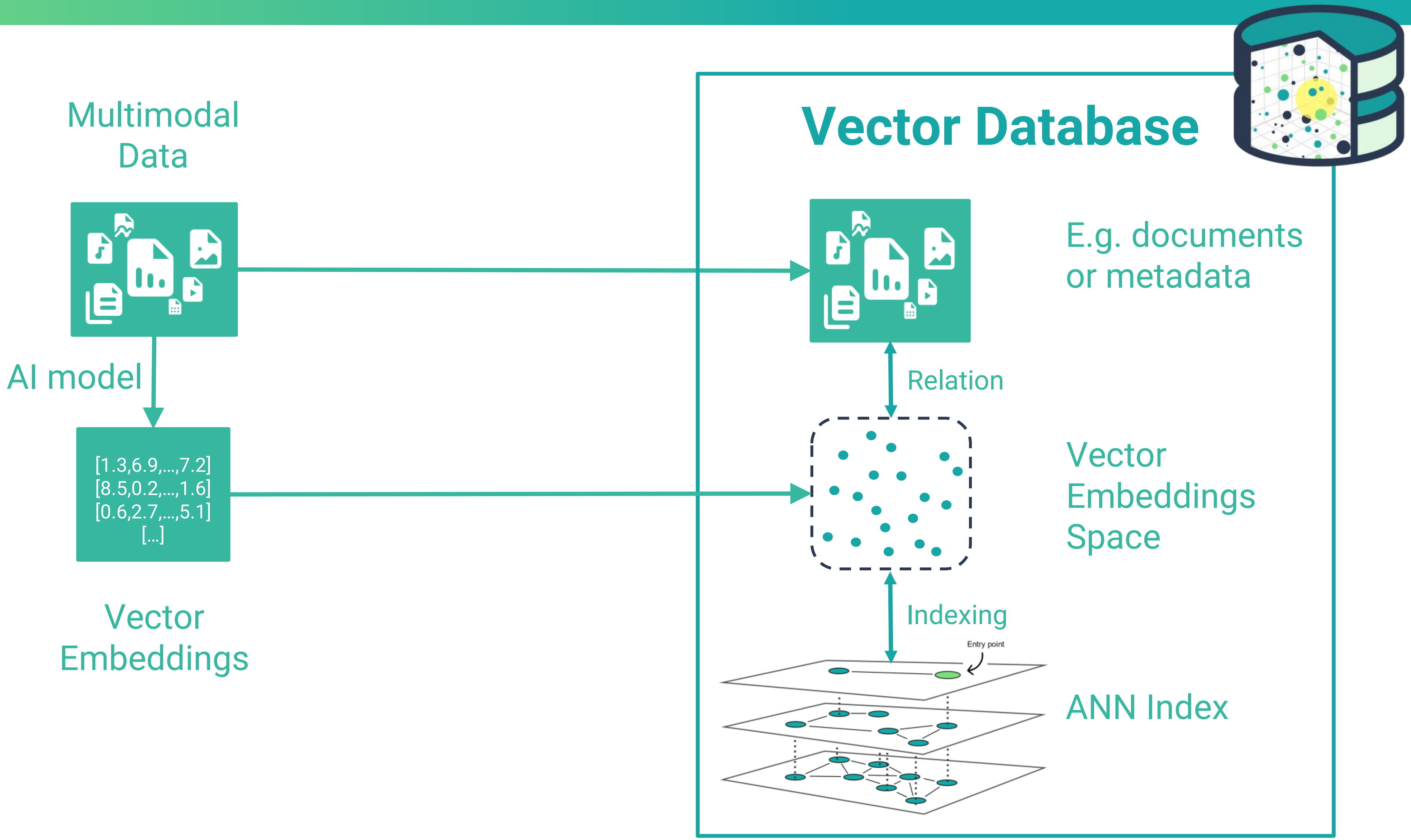
Adding Long-term memory
Persist the conversation history and search for relevant conversation pieces as needed.

Similarity Search / Semantic Retrieval
Often works better than „full text search“ (FTS)

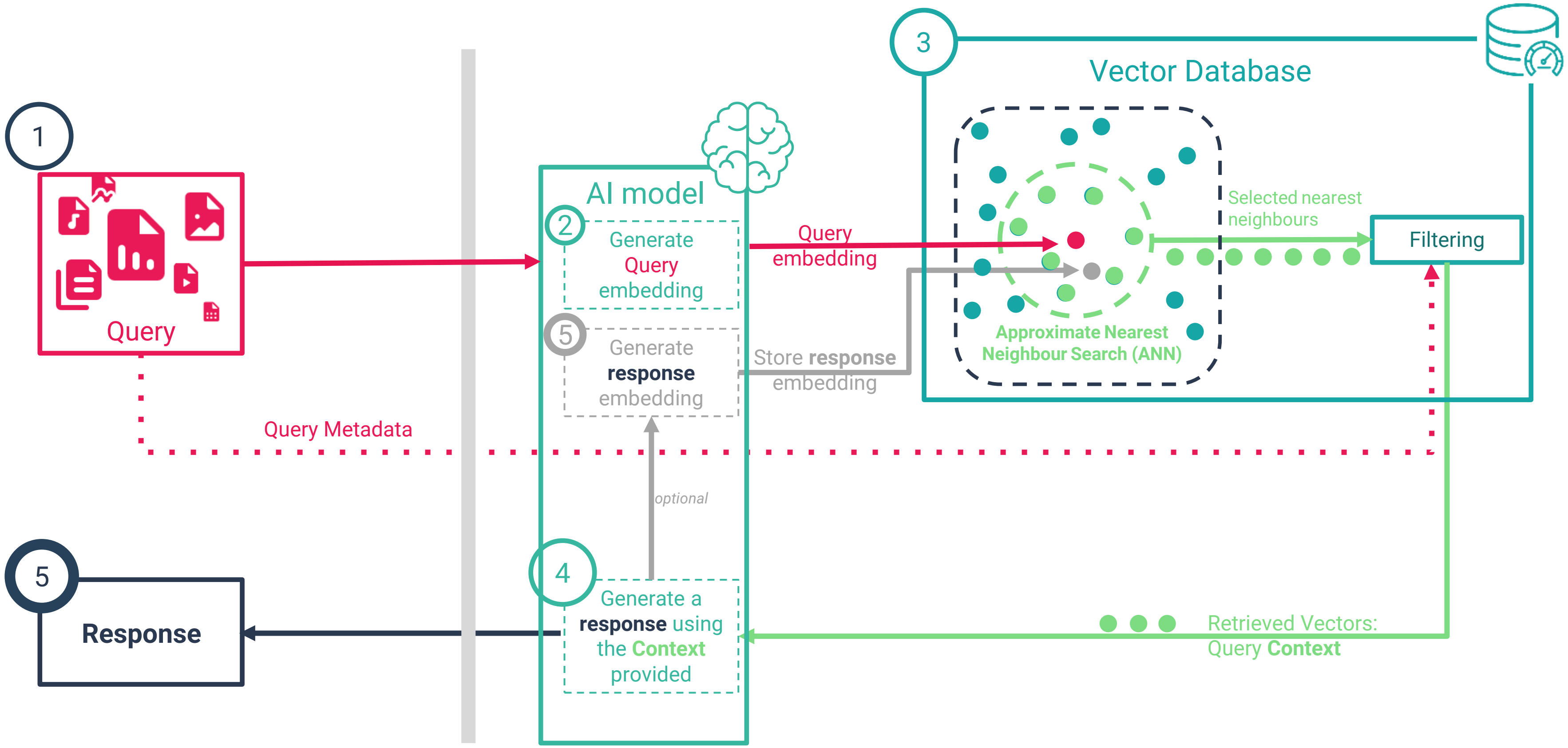
Enable Multimodel Search
Vector databases serve as the backbone to jointly analyze vectors from multimodal data for unified multimodal search and analytics.

Caching: Reducing LLM calls
Vector databases are used to cache similar queries and responses can be used as a lookup prior to calling the LLM (saving time and costs)

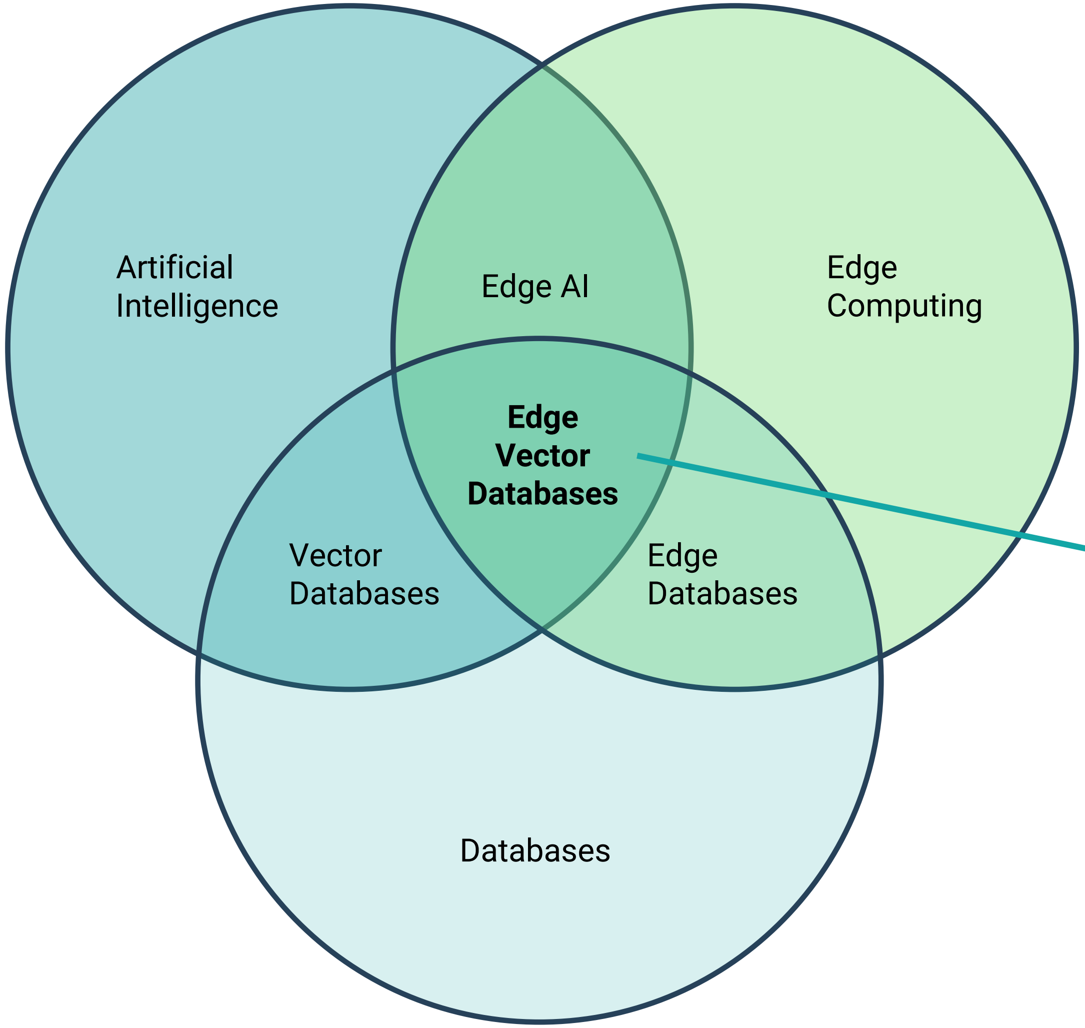
Vector Databases – Generation of Context



Vector Databases in Use: Efficient access to vector embeddings



The intersection of AI, Edge Computing & Databases



„on-device“, „local“, „Edge“
Vector Databases

On-device Vector Search Options

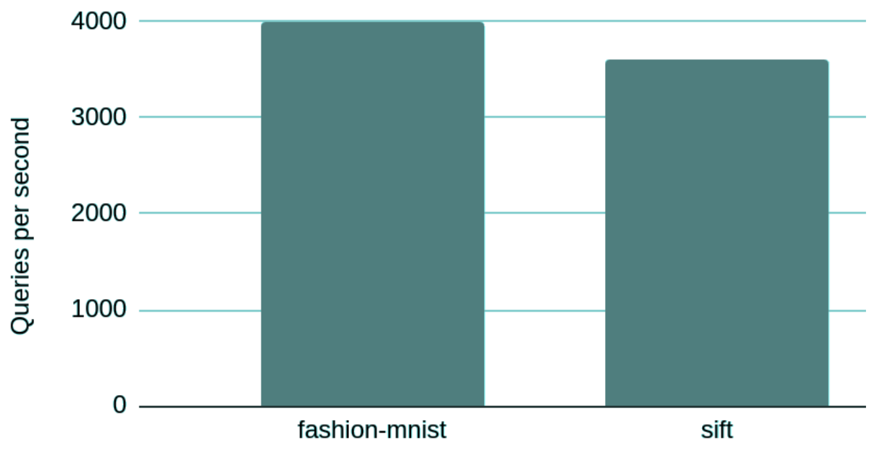
- Vector Search Libs (Pure Vector Indexes) like FAISS, Annoy, HNSW, …
- Build it yourself
- On-device: ObjectBox Vector Database (afaik)

➔ For fast protoytpes and demos, vector search libs can be great; for the long run, vector databases are better suited

Benchmarking ObjectBox Vector Search on Mobile

Tested on a 5(!) year old phone (LG G8S, ARM Cortex-A76)

- Tested using 2 well-known datasets
  - fashion-mnist: 768 dim., 60,000 vectors
  - sift: 128 dimensions, 1,000,000 vectors
- Query for the 10 nearest neighbors
- Time for one query: 0.25 / 0.27 milliseconds
  → Up to 4000 queries per second
- Enables „real-time“ use cases,
  e.g. for sensor or camera data:
  at 30 frames/sec → 100 queries/frame



How easy is it? Or: Can you do Edge AI with Vibe Coding?

- A Screenshot Searcher App
- Feature Set
  - Offline-first, on-device
  - Private (photos, screenshots never leave the device)
  - Extract texts, search texts
  - Search for semantic similarity
  - Search for image similarity (just for fun / comparison)
  - Categorize screenshots

The beginning was a breeze…

What was easy

- OCR - Extracting text from screenshots using ML Kit and doing text search
- Semantic Search – Text embedding search using MediaPipe & ObjectBox
- Image Embeddings – Image-embedding based search & ObjectBox
- Categorization of the screenshots with ML Kit Image Labeling

- Took less than a day with vibe coding
- Keep in mind: I only wanted a working example to learn (and share the learning))
- The code…is very likely not suited for a real release…
- Still great for testing!

So, what else could we do?

The Feature Creep

- Object Detection with MediaPipe, which supposedly has a ready-to-use ObjectDetector task (the AI thought: „ It's very simple to integrate into your existing MediaPipe workflow”)
- Categorization of text (search query) to search based on category match using MediaPipe TextClassifier with DBpedia model
- Sounded easy enough….

It was definetely TOO EASY

- All hell broke loose
- Nothing worked

- I discarded the object detection
- Focused on the text classification

Why not add a classic SLM on top and see if we can enhance the project a bit?

I decided to try Gemma…

Text classification: The model

- On Android, TensorFlow Lite (TFLite) is the standard deployment format
- I chose MobileBERT because it is lightweight, well-documented, and supported in tutorials/tools like TensorFlow Lite Model Maker
- However: MobileBERT is a general pretrained language model; it usually needs fine-tuning on a labeled dataset!
- Prebuilt TFLite models I found were typically for sentiment analysis → easy to run, but not what I needed…
- I couldn't find a ready-to-use TFLite model for the text categories I needed / wanted (e.g. based on DBPedia)

➡ to get what I needed, I would need to **fine-tune MobileBERT myself** and then convert/optimize it for Android (e.g. with quantization)

Text classification: Finetuning in practice (what it is supposed to be)

Basically: "**Run a Python script with a pretrained model + your labeled data + training settings**"

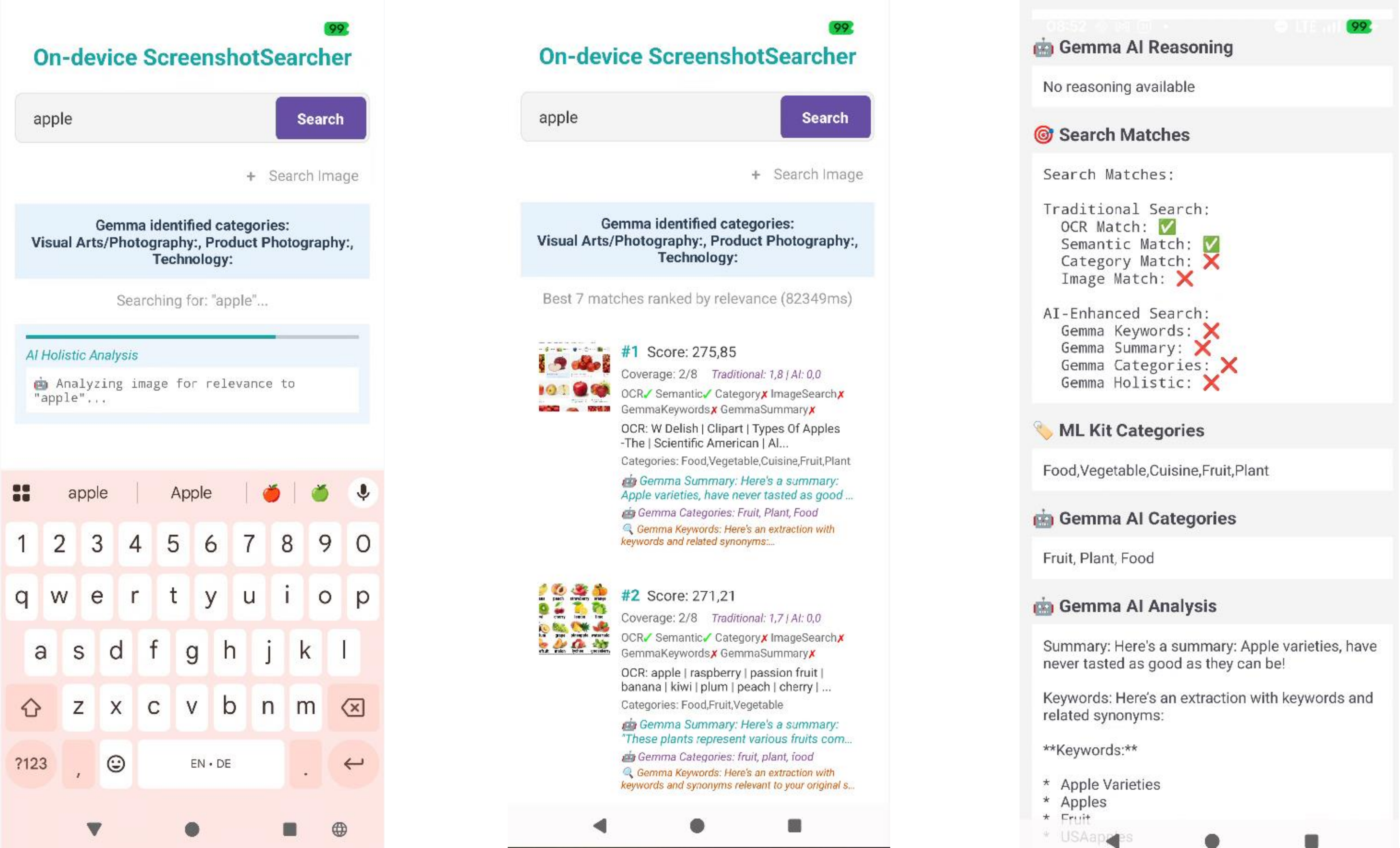
- Choose a framework (**TensorFlow/Keras** or **PyTorch**)
- Attach a **small classification head** (or reuse the model's head, or format outputs) and train for a few epochs
- Feed in your **labeled texts**, e.g. DBPedia (ready-made) or own data
- **Set a couple of parameters: batch size, learning rate, sequence length**

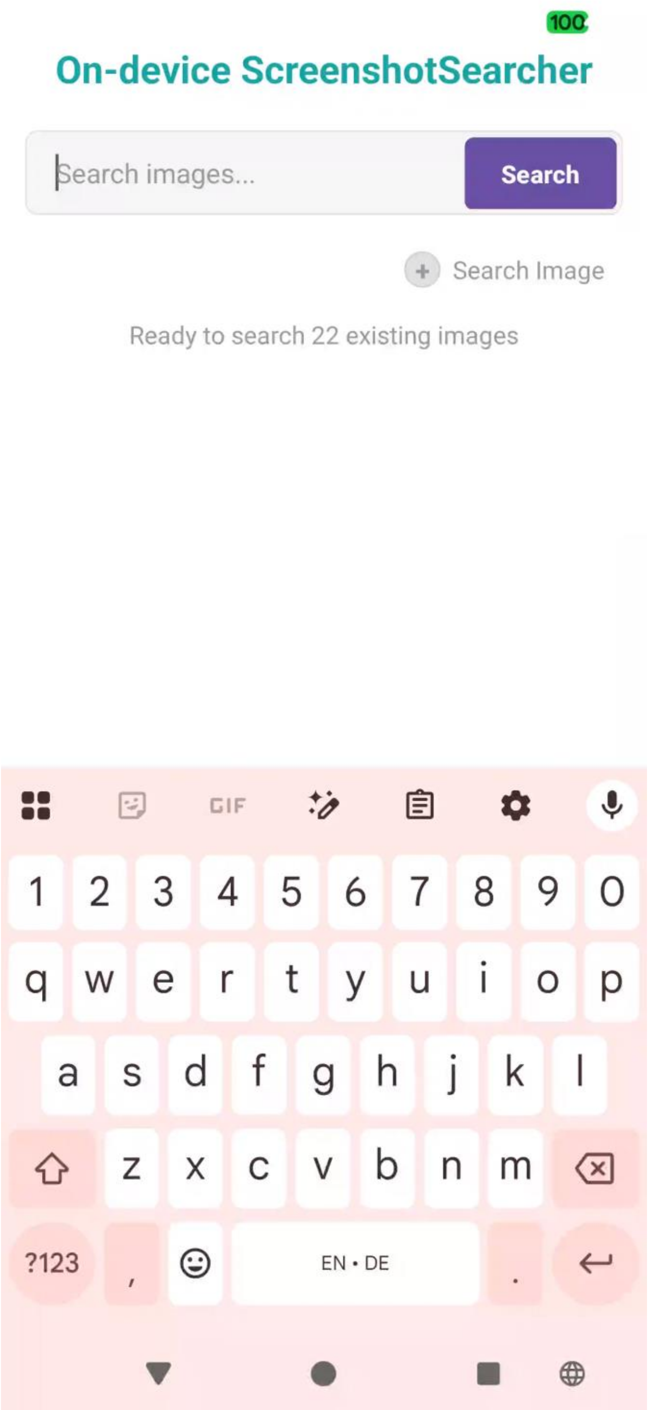
Text classification: Finetuning in practice (the reality)

- Expect trial-and-error: downgrade Python, swap library versions, repeat…
- Don't mix stacks, e.g., Keras 3 ↔ TF-Hub often don't play nicely together
- Keep a working „requirements.txt“
- When it works: **pin versions**, **export requirements**, **snapshot the env (venv)**
- Do a 1% smoke run before long training (!)
- Save checkpoints often (so an 8-hour run isn't lost) → check how long an epoch takes… if it is too long, choose another metric, e.g. each X steps, or Y minutes
  - Always save architecture separately
  - Use save_weights_only=True for frequent checkpoints
  - Periodic SavedModel exports for safety
- Watch basics: disk space, GPU driver/CUDA, and Python 3.10/3.11.

Gemma

- It’s easy to download the ready-made model → but there are many models, finding the right one is more of a challenge
- You must **accept the Gemma terms** on each related repo
- Trying to run Gemma on the Android emulator is not recommended! I tried and failed: Even though, if you try: Give the Android **emulator (AVD)** more space, (and make sure your host machine has enough resources too…)
- Once I went for testing with a real device, it was fairly easy to get Gemma to run… and then also to do what I wanted (however, some more testing and prompt engineering would be needed)
- Using Gemma: Be aware that you cannot reshare the model (e.g. you cannot share it as part of an example repo)


On-device ScreenshotSearcher
apple
Search
+ Search Image
Gemma identified categories:
Visual Arts/Photography:, Product Photography:, Technology:
Searching for: "apple"...
AI Holistic Analysis
Analyzing image for relevance to "apple"...
apple
Apple
EN • DE
On-device ScreenshotSearcher
apple
Search
+ Search Image
Gemma identified categories:
Visual Arts/Photography:, Product Photography:, Technology:
Best 7 matches ranked by relevance (82349ms)
#1 Score: 275,85
Coverage: 2/8
Traditional: 1,8 | AI: 0,0
OCR✓ Semantic✓ Category✗ ImageSearch✗
GemmaKeywords✗ GemmaSummary✗
OCR: W Delish | Clipart | Types Of Apples -The | Scientific American | Al...
Categories: Food,Vegetable,Cuisine,Fruit,Plant
Gemma Summary: Here's a summary: Apple varieties, have never tasted as good ...
Gemma Categories: Fruit, Plant, Food
Gemma Keywords: Here's an extraction with keywords and related synonyms:...
#2 Score: 271,21
Coverage: 2/8
Traditional: 1,7 | AI: 0,0
OCR✓ Semantic✓ Category✗ ImageSearch✗
GemmaKeywords✗ GemmaSummary✗
OCR: apple | raspberry | passion fruit | banana | kiwi | plum | peach | cherry | ...
Categories: Food,Fruit,Vegetable
Gemma Summary: Here's a summary: "These plants represent various fruits com...
Gemma Categories: fruit, plant, food
Gemma Keywords: Here's an extraction with keywords and synonyms relevant to your original s...
Gemma AI Reasoning
No reasoning available
Search Matches
Search Matches:
Traditional Search:
OCR Match: ✅
Semantic Match: ✅
Category Match: ❌
Image Match: ❌
AI-Enhanced Search:
Gemma Keywords: ❌
Gemma Summary: ❌
Gemma Categories: ❌
Gemma Holistic: ❌
ML Kit Categories
Food,Vegetable,Cuisine,Fruit,Plant
Gemma AI Categories
Fruit, Plant, Food
Gemma AI Analysis
Summary: Here's a summary: Apple varieties, have never tasted as good as they can be!
Keywords: Here's an extraction with keywords and related synonyms:
Keywords:
* Apple Varieties
* Apples
* Fruit


100
On-device ScreenshotSearcher
Search images...
Search
Search Image
Ready to search 22 existing images
GIF
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123
EN • DE

The final tech stack – for this minimal example

Name	Publisher	License	Link	Notes	Weights published	Training data published
TensorFlow (2.20.0)	Google	Apache 2.0	https://github.com/tensorflow/tensorflow/blob/master/LICENSE	Permissive	NA	NA
TensorFlow Hub (0.16.1)	Google	Apache 2.0	https://github.com/tensorflow/hub/blob/master/LICENSE	Attribution required.	NA	NA
TensorFlow Datasets (4.9.9)	Google	Apache 2.0	https://github.com/tensorflow/datasets/blob/master/LICENSE	Attribution required.	NA	NA
TensorFlow Metadata (1.17.2)	Google	Apache 2.0	https://github.com/tensorflow/metadata/blob/master/LICENSE	Permissive.	NA	NA
Keras (3.11.3)	Keras Team	Apache 2.0	https://github.com/keras-team/keras/blob/master/LICENSE	Permissive.	NA	NA
Transformers (4.56.2)	Hugging Face	Apache 2.0	https://github.com/huggingface/transformers/blob/main/LICENSE	Safe, but weights may differ (check model card)	NA	NA
Datasets (4.1.1)	Hugging Face	Apache 2.0	https://github.com/huggingface/datasets/blob/master/LICENSE	Attribution needed	NA	NA
Torch (2.7.0)	Meta AI	BSD-style	https://github.com/pytorch/pytorch/blob/main/LICENSE	Very permissive	NA	NA
TensorFlow Lite (2.16.1)	Google	Apache 2.0	https://github.com/tensorflow/tensorflow/blob/master/LICENSE	Permissive	NA	NA
Google ML Kit Text Recognition (v19.0.0)	Google	Proprietary (ML Kit ToS)	https://developers.google.com/ml-kit/terms	Not open source, but ML Kit APIs run on-device, no data is sent back to Google	NA	NA
Google ML Kit Image Labeling (v17.0.8)	Google	Proprietary (ML Kit ToS)	https://developers.google.com/ml-kit/terms	Not open source, but ML Kit APIs run on-device, no data is sent back to Google	NA	NA
MediaPipe Tasks Text / Vision	Google	Apache 2.0	https://github.com/google-ai-edge/mediapipe/blob/master/LICENSE	Fully open source.	NA	NA
ObjectBox (4.3.0)	ObjectBox	Bindings: Apache 2.0	https://objectbox.io/faq/	Permissive for typical apps; hosting ObjectBox as a service is not allowed	NA	NA
MobileBERT (uncased)	Google	Apache 2.0	https://tfhub.dev/google/mobilebert_uncased_L-24_H-128_B-512_A-4_F-4_OPT/1	Base model checkpoint	Yes	The training datasets (BooksCorpus + Wikipedia) are publicly known datasets; however exact dataset contents and preprocessing used in pretraining are not published
MobileBERT Tokenizer	Google (via Hugging Face)	Apache 2.0	https://huggingface.co/google/mobilebert-uncased	Tokenizer distribution		
MediaPipe text_embedder.tflite	Google	Apache 2.0	https://developers.google.com/mediapipe/solutions/text/text_embedder	Open source	Yes	No
Image embedder (MobileNetV3 .tflite)	Google / TF Hub	Apache 2.0	https://tfhub.dev/google/imagenet/mobilenet_v3_large_100_224/feature_vector/5	Feature extractor	Yes	Much was disclosed, but not fully, so: No
Gemma3-1B-IT (INT4, LiteRT-LM .litertlm)	Google	Gemma Terms of Use	https://huggingface.co/litert-community/Gemma3-1B-IT/tree/main	"Responsible commercial use (per terms)", hosting or sharing the model for download is not allowed	Yes	No (only high-level description published)

My assessment

- MediaPipe, MLKit, and ObjectBox were super easy; many basic on-device features ran within a day
- Trying to finetune a model (nothing fancy mind you!) was a major fail (but tons of learnings already on the way there ;)) → and I'm not giving up
- Gemma was fairly easy to set up → quality is ok, but needs time for testing and enhancing
- Overall: OCR Texts are pretty good, if you want to find screenhots…
- But it's great that doing AI on-device with vibe coding is so easy, you really can get (some) results
- At the speed at which AI is progressing… in 6 months, you can probably already do pretty advanced apps without being a developer (?)

AI anywhere | anytime

is already very possible

even with vibe coding

Feedback and Questions

Connect

https://www.linkedin.com/in/vivien-dollinger

Feedback

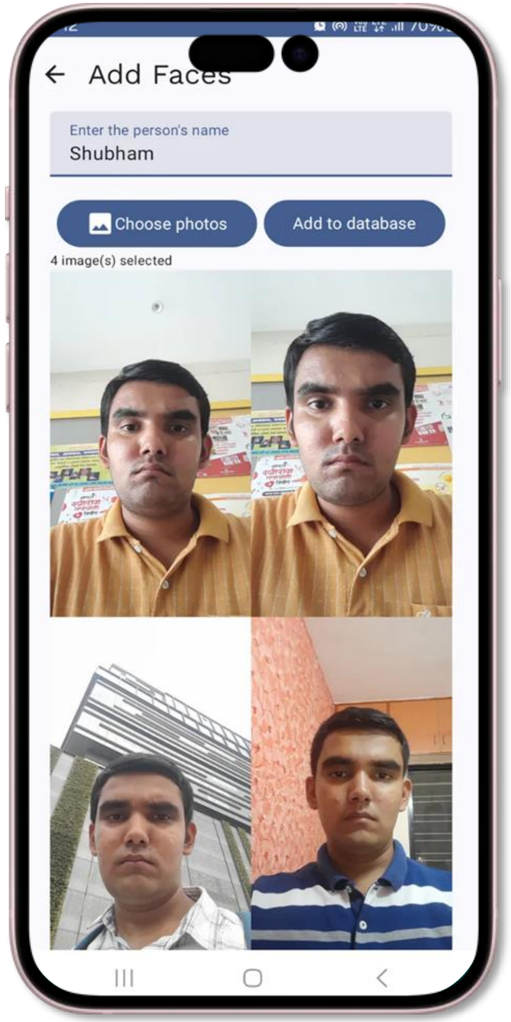
Thoughts? Questions? Comments?

Share

#ObjectBox #Database

Backup Slides

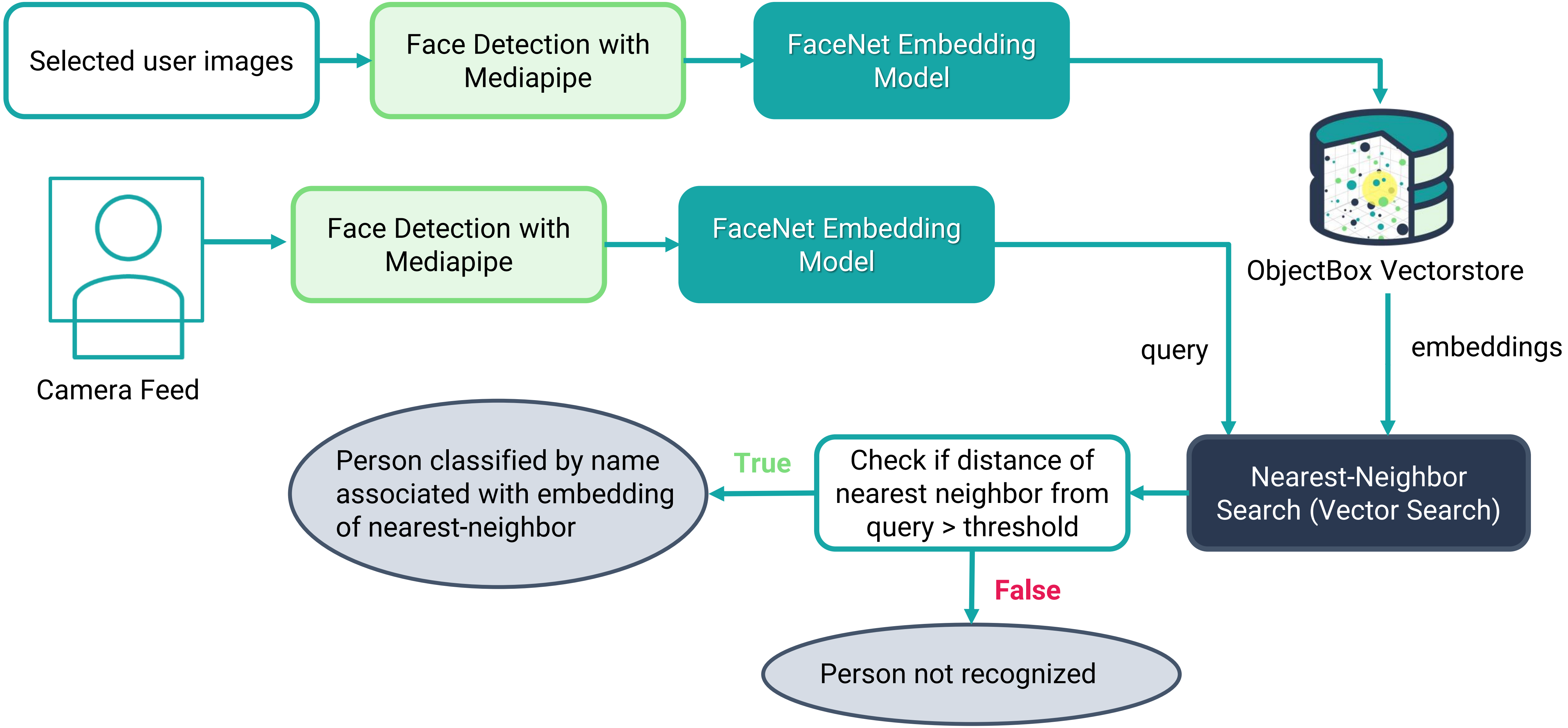
Use Case: On-device Face Recognition



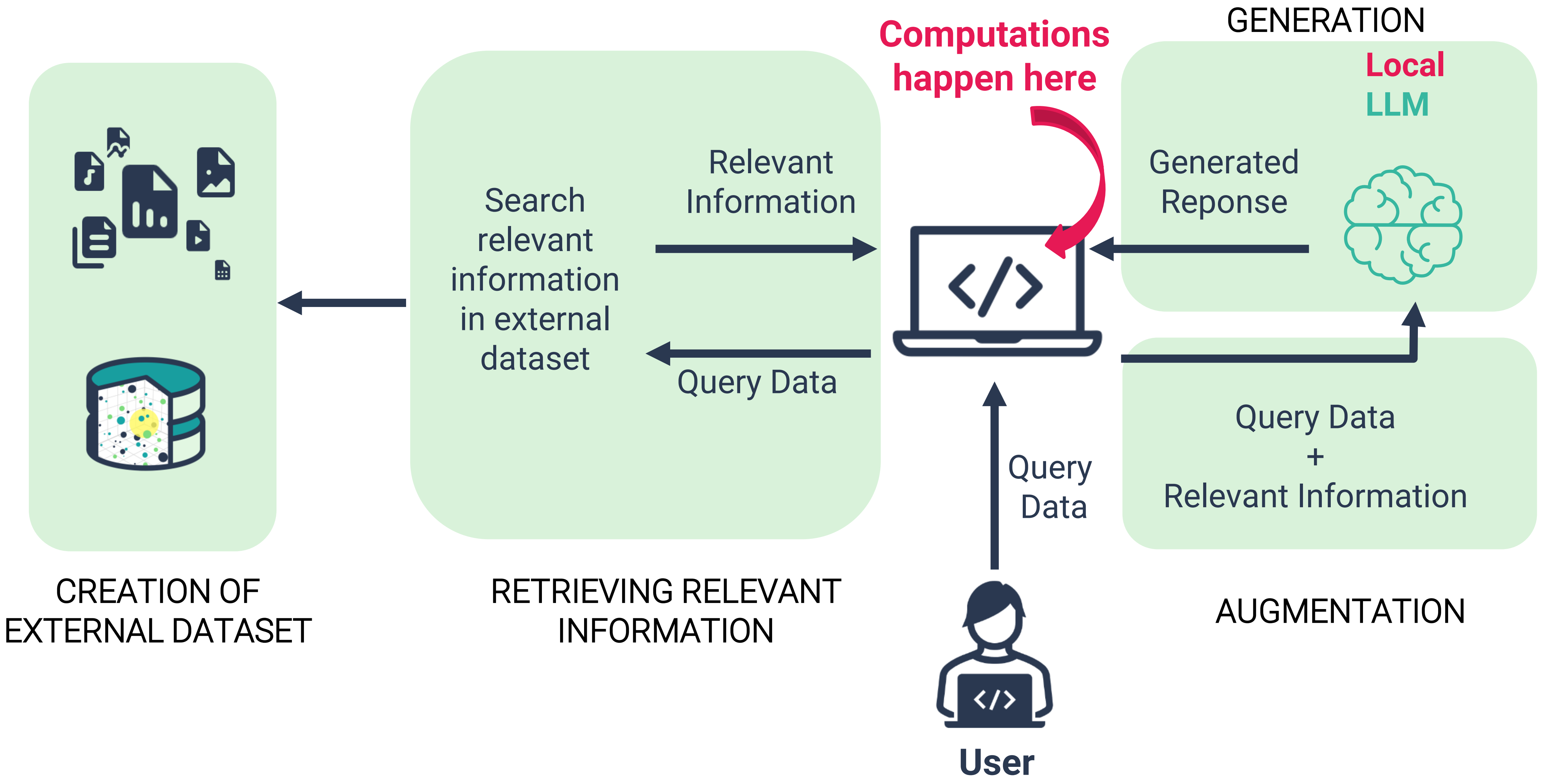
Add images to database

Real-time recognition

Use Case: On-device Face Recognition



On-device Retrieval Augmented Generation (RAG)

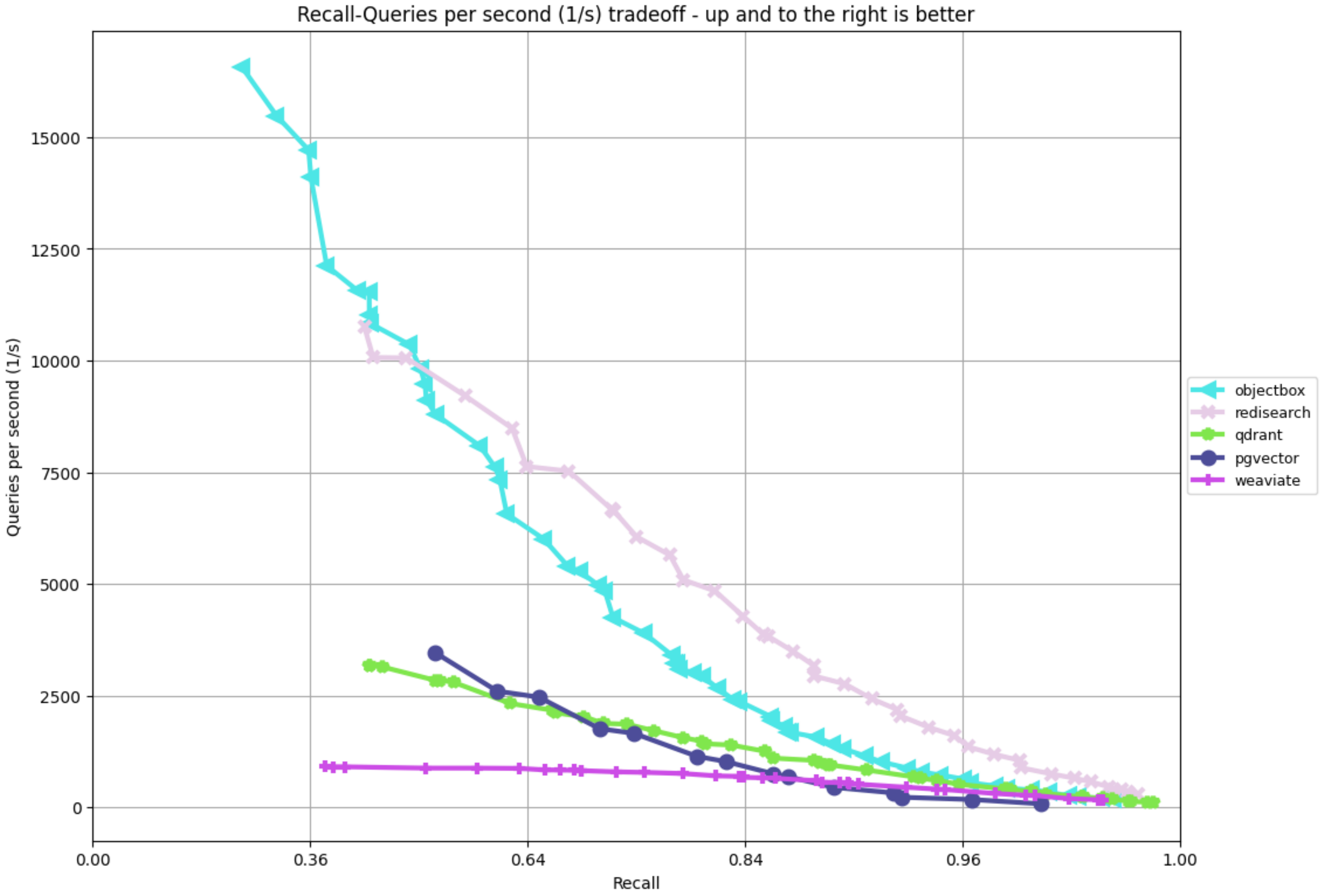


On-device Vector Search options compared

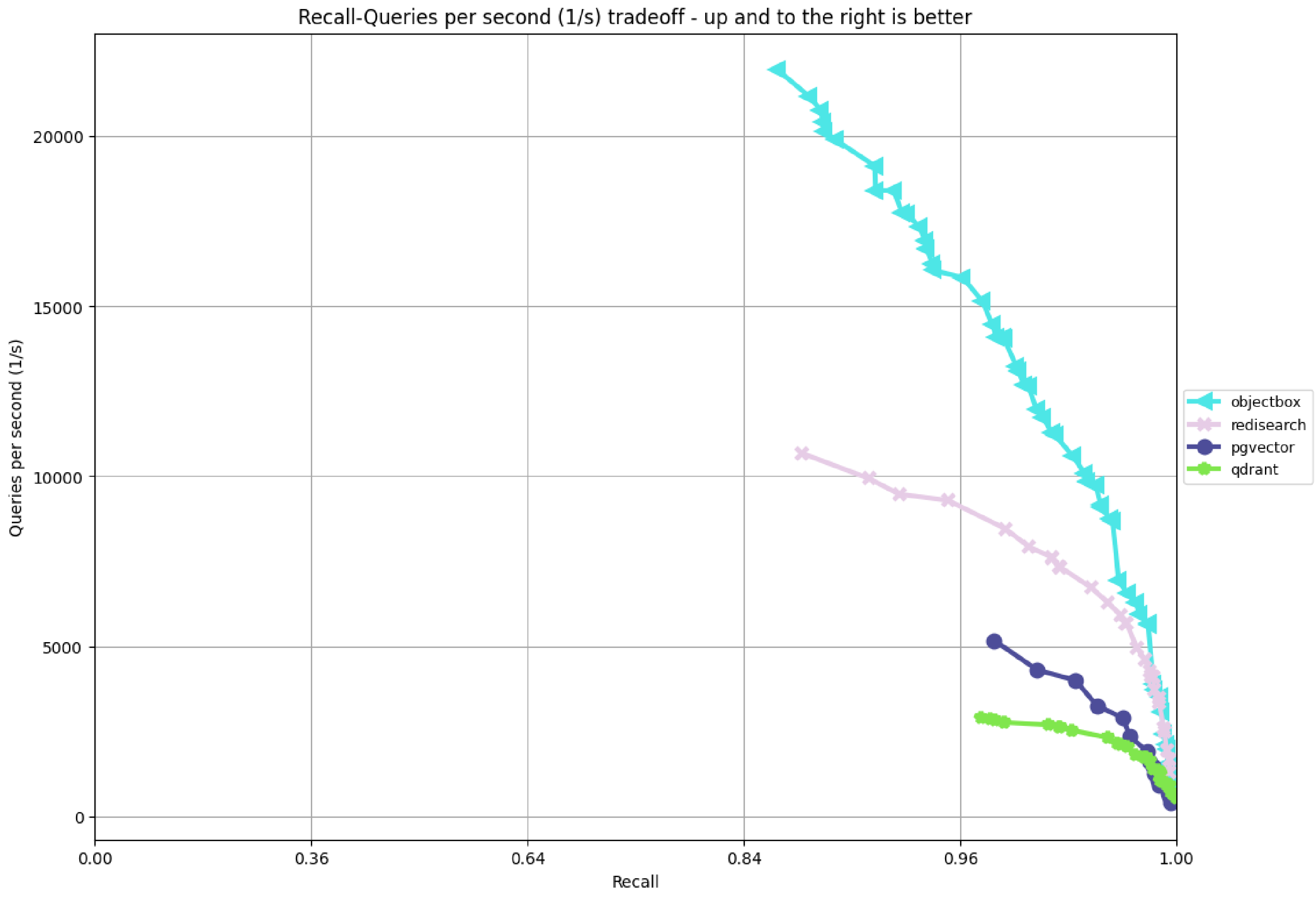
	Vector Search Lib (e.g. FAISS, HNSW-lib)	On-device Vector-Database (e.g. ObjectBox)
Persistence	Snapshot (save / load all)	Continual persistency
RAM consumption	All data (vectors) in RAM (in memory only!)	Disk + RAM (typically)
Minimum HW requirements	High RAM requirements, no disk	Less RAM req, needs disk
Persistent Data updates	Inefficient: a new snapshot (all data) must be saved	Efficient: „regular" database operation, only the changes updated
Data types	Only vectors; no other data types are stored	Other data, e.g. metadata can be stored together with the vector data, which is a useful feature e.g. for RAG use cases
Feature set	Pure vector search, nothing more	Vector databases often come with build-in DB-functionalities like backup, recovery, security
Scalability	Limited by RAM	Typically, superior to a pure Vector Search lib

➡ For fast protoytpes and demos, vector search libs can be great; for the long run, vector databases are better suited

Vector Search: Glove-100-Angular



Vector Search: fashion-mnist-784-euclidean



On **myriads of devices across verticals**