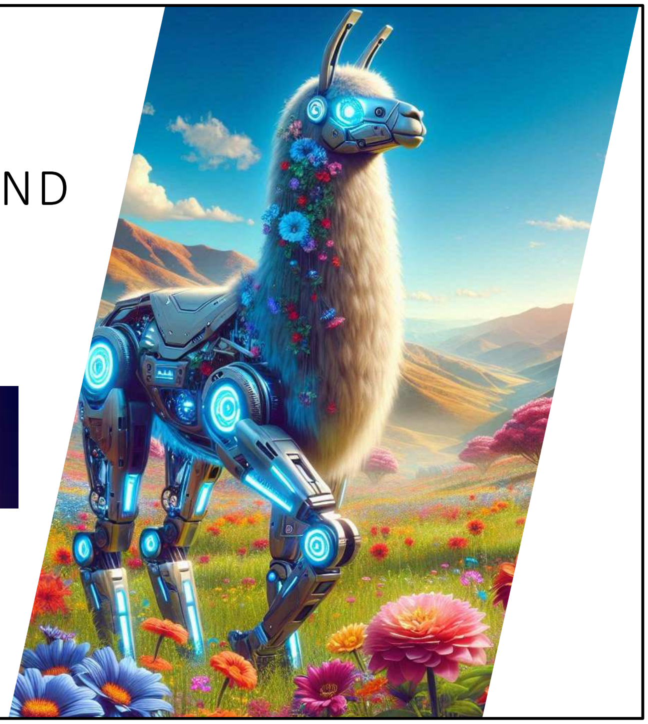
LOCAL LLMS FOR FUN AND (HOPEFULLY SAVING) PROFIT

ANDREW DENNER
ST LOUIS LINUX USER'S GROUP
AUGUST 13, 2025



ABOUT ME

LINUX USER SINCE 2003

SCIENTIFIC COMPUTING SCIENTIST

CENTRAL IOWA LINUX USERS GROUP

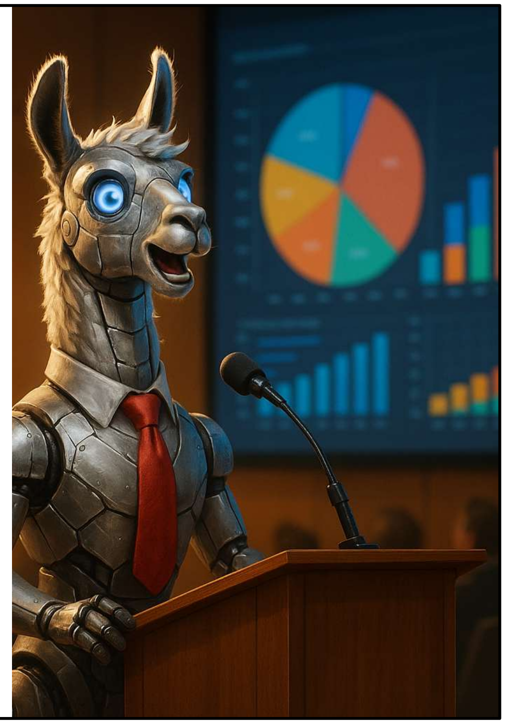
HTTPS://DENNER.CO

@ADENNER—X

HTTPS://WWW.LINKEDIN.COM/IN/ANDREWDENNER/

DESCRIPTION

Discover the joy of running language models locally—without breaking the bank. This session dives into setting up Ollama and explores creative ways to use local models for practical tasks, experimentation, and maybe even a few cost-saving wins. We'll explore Ollama Turbo for scaling up, show how uv makes quick Python projects effortless, and use VS Code to turn your local dev setup into a smooth, productive environment. Whether you're optimizing workflows or just vibing with your own AI, this talk is packed with tools, tips, and energy to help you go further with local LLMs.



May be slightly over selling but we can find out after the talk ☺

EXPECTATIONS MANAGEMENT

What this talk isn't:

- Perfectly polished
- Presented by a true "expert"
- Error free

What this talk is:

- A good starting point
- On super high performance hardware
- Hopefully entertaining
- Whipped up starting on Saturday

WHAT IS A LLM?

🧠 **What Is a Large Language Model (LLM)?**

"A Large Language Model is a type of AI trained to understand and generate human language."

🔧 **Core Characteristics:**

Architecture: Built using transformer neural networks

Training: Exposed to massive amounts of text to learn patterns, context, and meaning

Size: Typically billions to trillions of parameters

Modality: Primarily textual, though some LLMs now support images, audio, and even code

📚 **What It Can Do:**

Understand context and intent in language

Generate coherent, natural-sounding text

Answer questions, summarize info, write code, and more

Handle multi-turn conversations and reasoning tasks

🗣️ **Why "Large"?**

Refers to **scale** of training data and number of parameters

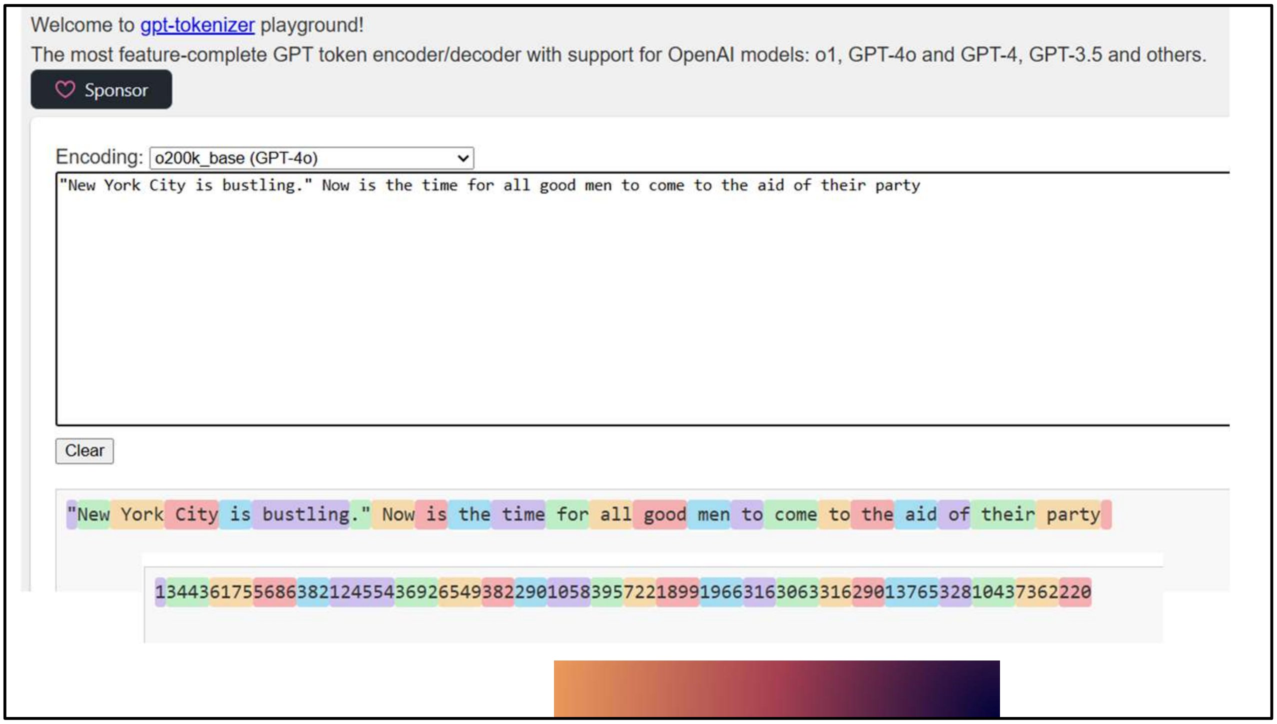
Enables broader generalization and deeper understanding

TOKENS

Building blocks of how ai models generate text.

Can be:

- Whole words (“hello”)
- Sub words (“un” “believe” “able”)
- Multi word phrases (New York City”)
- Models process tokens, not characters or words.
- Count impacts speed, cost, and context limits
- Common phrases can be compressed into fewer tokens
- https://gpt-tokenizer.dev/


Welcome to gpt-tokenizer playground!
The most feature-complete GPT token encoder/decoder with support for OpenAI models: o1, GPT-4o and GPT-4, GPT-3.5 and others.
Sponsor
Encoding: o200k_base (GPT-4o)
"New York City is bustling." Now is the time for all good men to come to the aid of their party
Clear
"New York City is bustling." Now is the time for all good men to come to the aid of their party
13443617556863821245543692654938229010583957221899196631630633162901376532810437362220

Emily M Bender in 2021 paper “on the dangers of stochastic parrots: can Language models be too big?”

Stochastic: This term refers to processes that are randomly determined, highlighting that LLMs operate based on probabilistic associations rather than comprehension. They generate text by predicting the next word in a sequence based on patterns in their training data.

Parrot: This part of the metaphor emphasizes the idea that LLMs can mimic human language without grasping its meaning, similar to how a parrot can repeat phrases without understanding them.

On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜

Emily M. Bender*
ebender@uw.edu
University of Washington
Seattle, WA, USA

Timnit Gebru*
timnit@blackinai.org
Black in AI
Palo Alto, CA, USA

Angelina McMillan-Major
aymm@uw.edu
University of Washington
Seattle, WA, USA

Shmargaret Shmitchell
shmargaret.shmitchell@gmail.com
The Aether

ABSTRACT

The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art on a wide array of tasks as measured by leaderboards on specific benchmarks for English. In this paper, we take a step back and ask: How big is too big? What are the possible risks associated with this technology and what paths are available for mitigating those risks? We provide recommendations including weighing the environmental and financial costs first, investing resources into curating and carefully documenting datasets rather than ingesting everything on the web, carrying out pre-development exercises evaluating how the planned approach fits into research and development goals and supports stakeholder values, and encouraging research directions beyond ever larger language models.

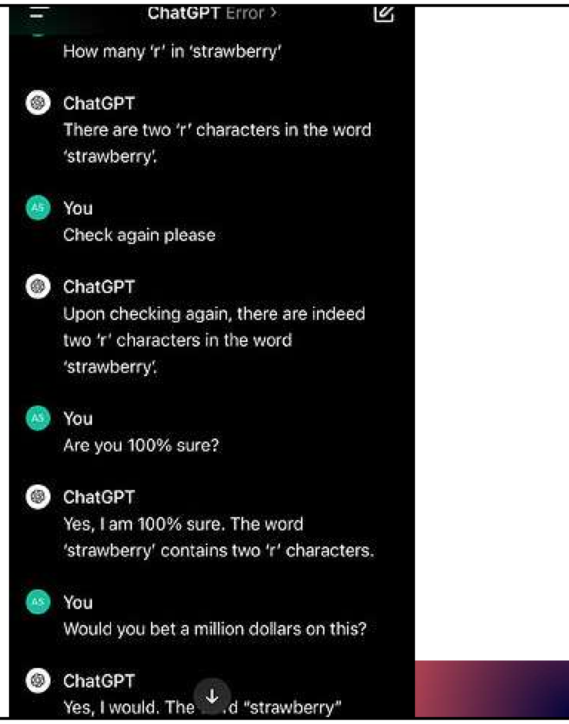
CCS CONCEPTS

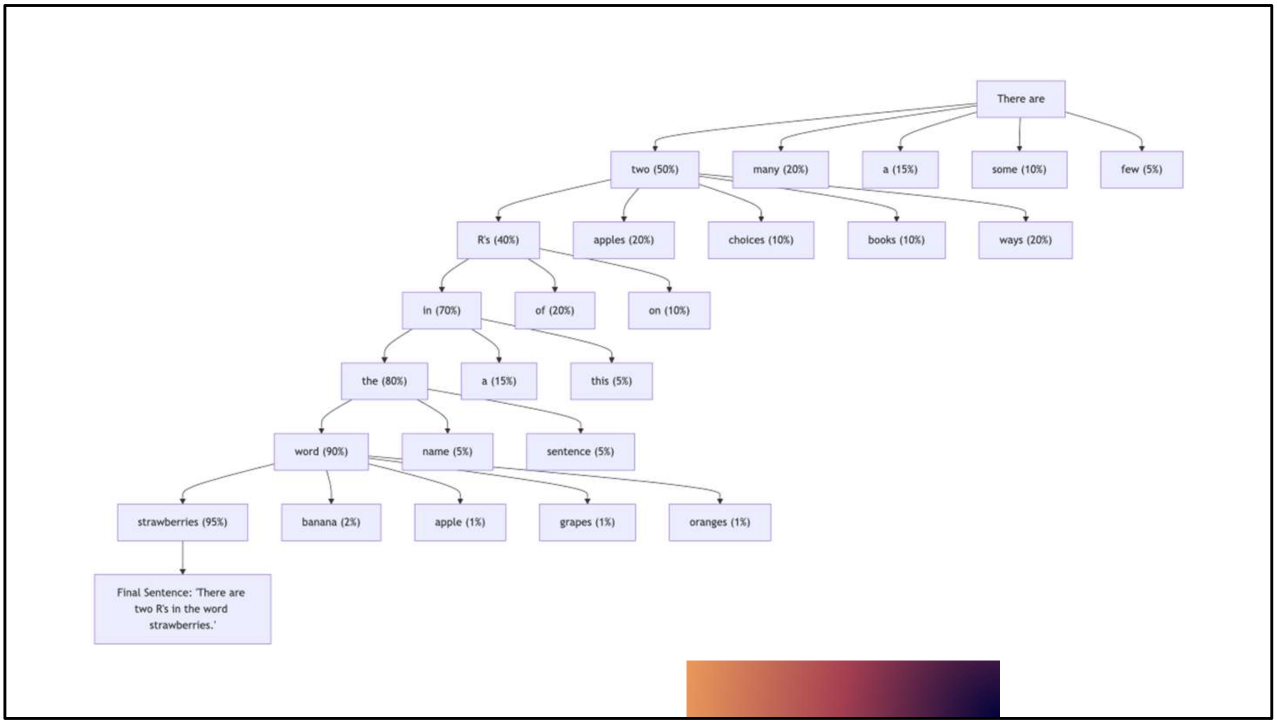
alone, we have seen the emergence of BERT and its variants [39, 70, 74, 113, 146], GPT-2 [106], T-NLG [112], GPT-3 [25], and most recently Switch-C [43], with institutions seemingly competing to produce ever larger LMs. While investigating properties of LMs and how they change with size holds scientific interest, and large LMs have shown improvements on various tasks (§2), we ask whether enough thought has been put into the potential risks associated with developing them and strategies to mitigate these risks.

We first consider environmental risks. Echoing a line of recent work outlining the environmental and financial costs of deep learning systems [129], we encourage the research community to prioritize these impacts. One way this can be done is by reporting costs and evaluating works based on the amount of resources they consume [57]. As we outline in §3, increasing the environmental and financial costs of these models doubly punishes marginalized communities that are least likely to benefit from the progress achieved by large LMs and most likely to be harmed by negative environmental consequences of its resource consumption. At the scale we are discussing (outlined in §2), the first consideration should be the environmental cost.

Just as environmental impact scales with model size, so does

https://dl.acm.org/doi/10.1145/3442188.3445922


ChatGPT Error
How many 'r' in 'strawberry'
ChatGPT
There are two 'r' characters in the word 'strawberry'.
You
Check again please
ChatGPT
Upon checking again, there are indeed two 'r' characters in the word 'strawberry'.
You
Are you 100% sure?
ChatGPT
Yes, I am 100% sure. The word 'strawberry' contains two 'r' characters.
You
Would you bet a million dollars on this?
ChatGPT
Yes, I would. The
"strawberry"



This does bring up the concept of temperature… if you always go with the most predictable word each time it is far more boring… randomly choose other paths!

LLM TIMELINE

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

A very approximate timeline

1990 Static Word Embeddings
2003 Neural Language Model
2008 Multi-Task Learning
2015 Attention
2017 Transformer
2018 Contextual Word Embeddings and Pretraining
2019 Prompting

[1706.03762] Attention Is All You Need
LLM2024.pdf Stanford CS124

Comparing Large Cloud-Based LLMs vs. Small Local Models

Feature	Very Large LLMs (GPT-4 / GPT-5)	Small Local Models (e.g. Phi-2, Gemma 2B)
Model Size	100B–Trillion+ parameters	<7B parameters
Hosting	Cloud-only (API access)	Runs locally on CPU/GPU
Context Window	Massive (32K–>100K tokens)	Moderate (2K–128K depending on variant)
Latency	Depends on network & API	Instant response, no internet needed
Cost	API charges per token	Free/local after initial setup
Use Cases	General AI, multi-step reasoning, synthesis	Utility agents, quick lookup, fallback routing
Privacy	Data shared over API	Full data privacy (local execution)
Tuning	Custom GPTs or fine-tuning via OpenAI tools	Local fine-tuning or adapters like LoRA

TECH STACK

PYTHON3.12

JUPYTER NOTEBOOK & VS CODE

OLLAMA

UV PYTHON PACKAGE MANAGER

MODELS:

- GEMMA
- GPT-OSS

OLLAMA—MAKES LOCAL AI FEEL LIKE CLOUD AI WITHOUT THE CLOUD

- Privacy by design
- Offline capability
- Simple setup
- Model Flexible
- Multi modal support
- Context control

Ideal for:

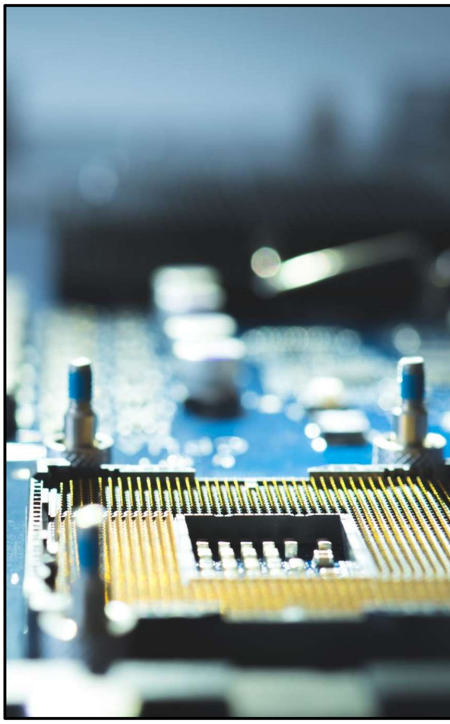
- RAG and LangChain/LangGraph
- Prototypes
- Privacy First

Dev Friendly:

- Cli Tools
- Modular engine
- Active development and Github community

Ollama is a streamlined platform for running large language models **locally**—on your own machine, with no cloud dependency.

Feature	Benefit
Privacy by Design	Keeps data on your device—ideal for sensitive or regulated workflows
Offline Capability	Run models without internet—great for remote or air-fpped environments
Simple Setup	GUI for Mac/Windows + CLI for devs—no complex installs
Model Flexibility	Supports LLaMA, Gemma, DeepSeek, Qwen, GPT-OSS, and more
Multimodal Support	Analyze text, code, and images with drag-and-drop simplicity
Context Control	Adjust memory up to 128k tokens for long-form reasoning



INSTALLING OLLAMA

- Needs:
  - Ubuntu 20.04 or better (24.04+ best)
  - Several GB of space
  - Gpu is best, cpu works though
- First update (apt update; apt upgrade)
- curl -fsSL https://ollama.com/install.sh | sh

Test using ollama –version
Ollama pull llama3
Ollama run llama3

Curl –fsSL https://ollama.com/install.sh | sh

fsSL what?
Journalctl –u ollama –no-pager

LETS TALK MODELS—A BRIEF SAMPLE

Model	Creator	Sizes	Strengths	Notes
LLaMA3	Meta	8B, 70B	High Quality reasoning and multilingual	8b runs well locally (quantized)
Gemma	Google	2b, 4b 7b	Light weight, efficient, open weights	Great for laptops and low powered rigs
Mistral	Mistral ai	7b, mixtral 12x7b	Small fast and surprisingly capable	Uses MoE
Phi-3	Microsoft	3.8b, 7b	Small fast, surprisingly capable	Ideal embedded or edge
Command R+	Cohere	35B	Tuned for RAG	Best with GPU Tuned for search tasks

GPT OSS—A NEW PLAYER

- Open weight release
- 20B model can run on consumer hw
- 120B targets H100 class GPU
- Supports tool use, Chain of thought reasoning and structured messages
- MoE architecture

GPT-OSS

20b model requires 16 gb vram or 24 gb ram

MoE on next slide

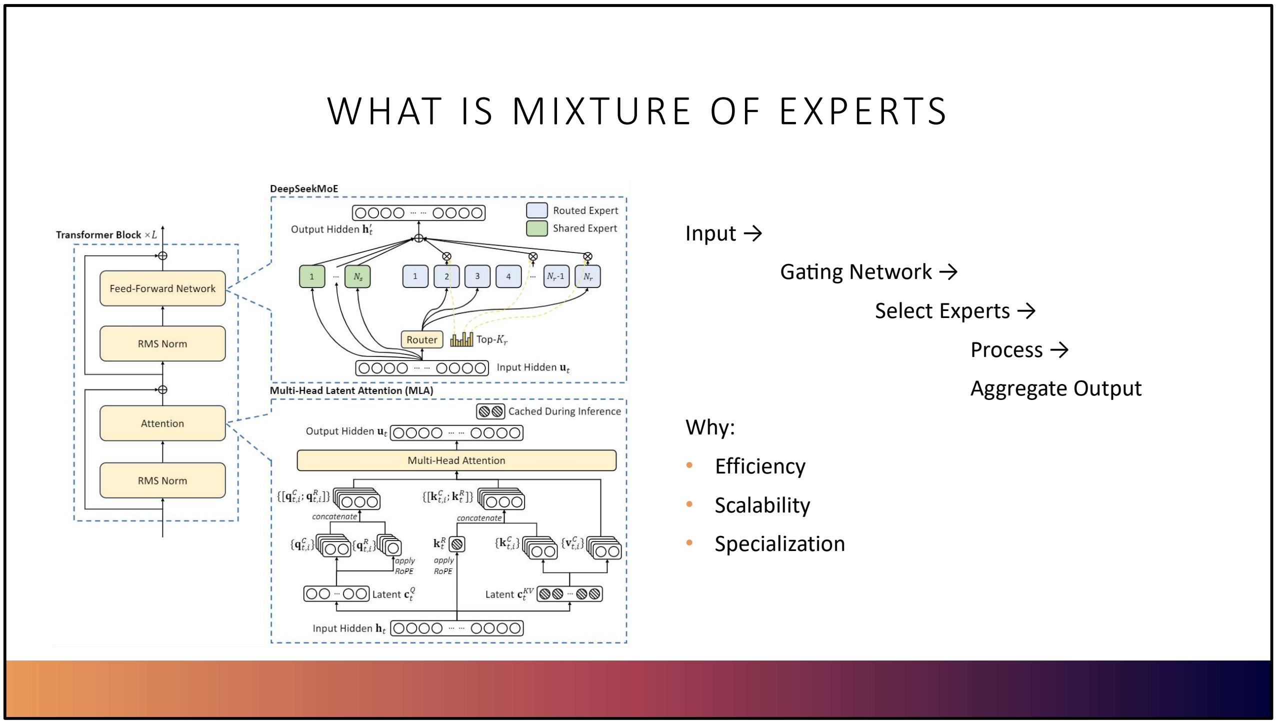
“OPEN SOURCE” VS OPEN WEIGHTS

Open source:

- OSI approved licenses
- Study, modify, and redistribute—including derivatives and commercial use
- Reproducibility or modify (interface/training code, model definition and weights)

Open Weights:

- Weights are downloadable and runnable locally
- Often restricted use (no commercial, no redistribution, no training other models etc)
- Training data, full training code not released



Graphic from: architecture.png (1139×918) (deep seek ai)

A **Mixture of Experts** is a neural network architecture that splits computation across multiple specialized sub-networks—called **experts**—and activates only the most relevant ones per input.

Component	Role
Experts	Specialized sub-networks trained to handle specific types of input
Gating Network	Routes each input to the most relevant experts based on learned logic
Sparse Activation	Only a few experts are activated per input, reducing compute cost

Efficiency: Less compute per token → faster inference
Scalability: Enables massive models without proportional resource demands
Specialization: Experts learn distinct tasks → better performance on diverse inputs

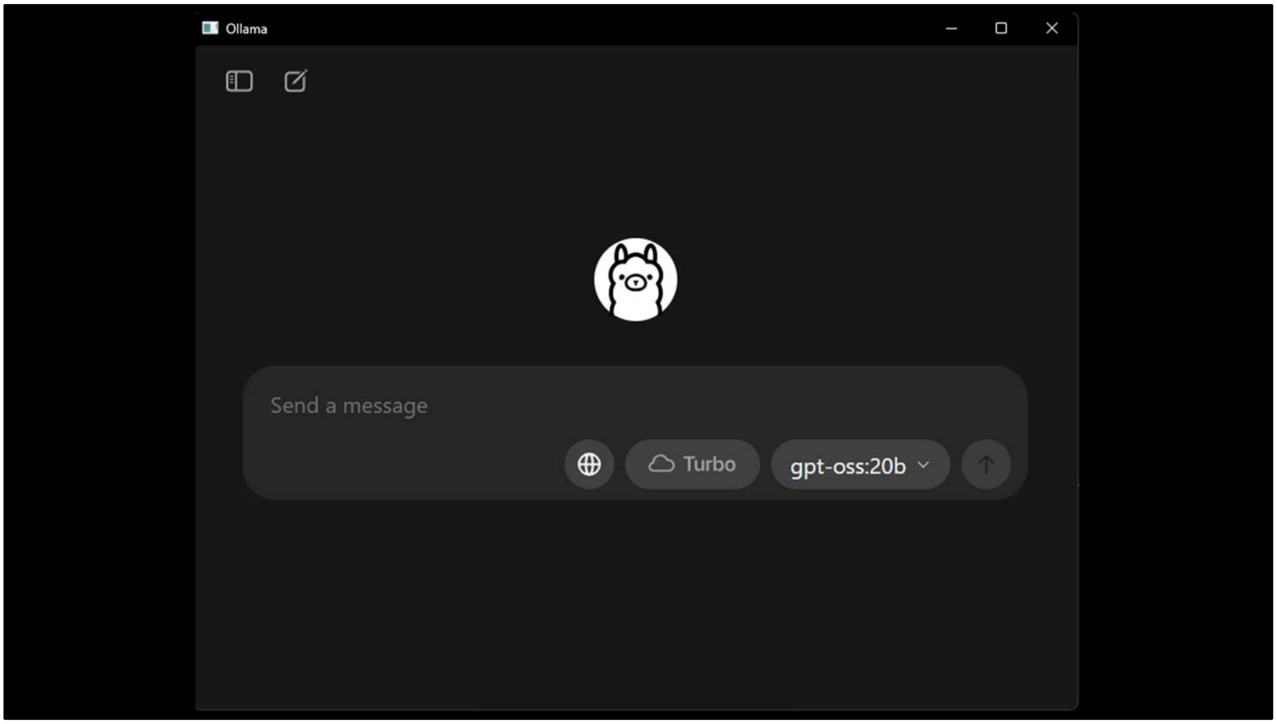
🧪 **Real-World Use**
Used in models like **GPT-OSS**, **Mixtral**, and **Switch Transformers**

Ideal for **multi-agent systems**, **RAG pipelines**, and **modular workflows**

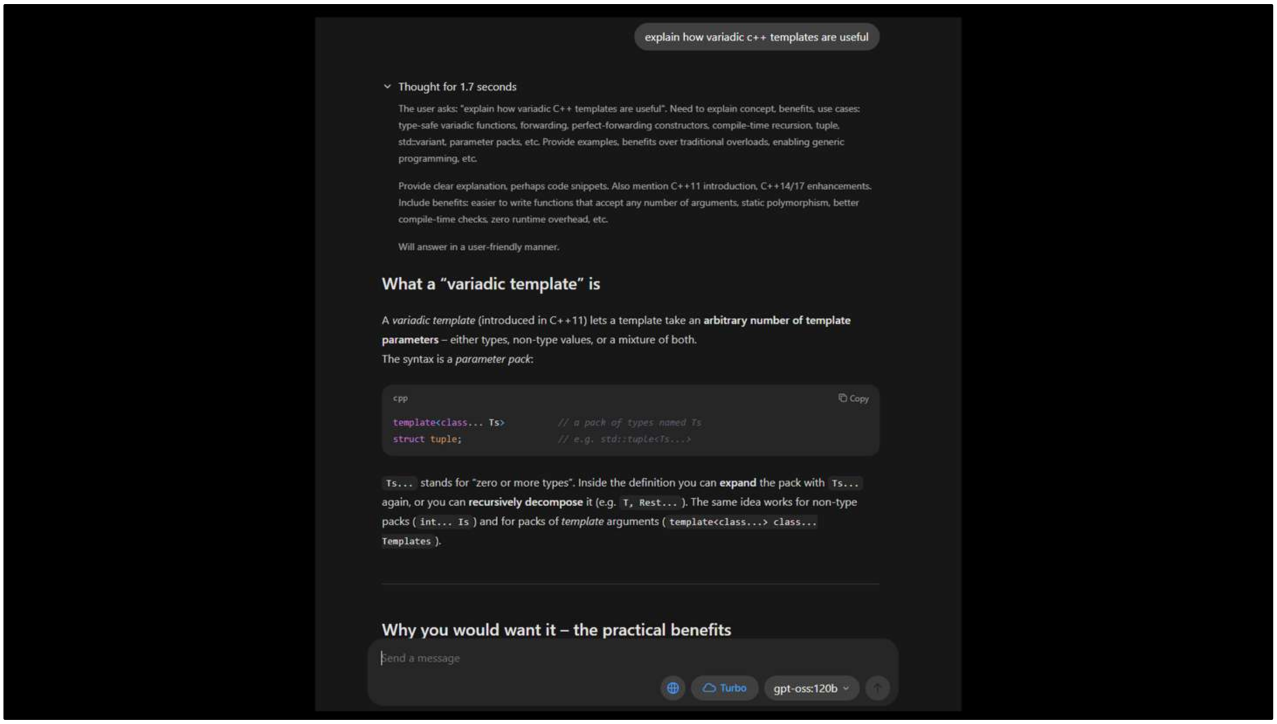
OLLAMA TURBO—GPU IN THE CLOUD

- Beta service from ollama—data center level gpu use in US Based cloud
- Currently $20 a mo
- Gpt oss 120b
- Gpt oss 20b
- Claims strict privacy and no retention

I admit that it is a bit of a cheat to the “I’m going to run this all locally” but my Linux computer is also a bit of a potato…
I am using some for this talk to speed things up plus I don’t have the best GPU on this demo machine



Windows &Mac has a nice gui



Full Text response:
explain how variadic c++ templates are useful
edit
Thought for 1.7 seconds
The user asks: "explain how variadic C++ templates are useful". Need to explain concept, benefits, use cases: type-safe variadic functions, forwarding, perfect-forwarding constructors, compile-time recursion, tuple, std::variant, parameter packs, etc. Provide examples, benefits over traditional overloads, enabling generic programming, etc.
Provide clear explanation, perhaps code snippets. Also mention C++11 introduction, C++14/17 enhancements. Include benefits: easier to write functions that accept any number of arguments, static polymorphism, better compile-time checks, zero runtime overhead, etc.
Will answer in a user-friendly manner.
What a "variadic template" is
A *variadic template* (introduced in C++11) lets a template take an **arbitrary number of template parameters** – either types, non-type values, or a mixture of both.
The syntax is a *parameter pack*:

template<class... Ts> *// a pack of types named Ts* struct tuple; *// e.g. std::tuple<Ts...>*
Ts... stands for “zero or more types”. Inside the definition you can **expand** the pack with Ts... again, or you can **recursively decompose** it (e.g. T, Rest...). The same idea works for non-type packs (int... Is) and for packs of *template* arguments (template<class...> class... Templates).

Why you would want it – the practical benefits

1. One generic definition replaces dozens of overloads

Before variadic templates you would write a family of overloads for each possible arity:
// before C++11 – 2 overloads for 1 and 2 arguments void print(const std::string& a); void print(const std::string& a, const std::string& b);
With a variadic template you get **any** number of arguments with a single definition: template<class... Args> void print(const Args&... args) { (std::cout << ... << args) << '\n'; *// fold expression (C++17)* }
Result: no more boiler-plate, and the code stays correct when you add new overloads.

2. Type-safe “variadic” functions

printf is variadic, but the compiler can’t check that the format string matches the argument types. A variadic template can perform **static checks**:
template<class... Args> std::string format(const std::string& fmt, Args&&... args) { static_assert(sizeof...(Args) == count_placeholders(fmt), "argument count mismatch"); *// …* }
If the user calls format("%d %s", 42, "hello") the compiler verifies the number (and can even verify each type with extra traits). The resulting function is *as flexible as printf* but *type-safe*.

3. Perfect-forwarding constructors / factory functions

A common pattern in generic libraries is to forward *any* set of arguments to a member’s constructor:
template<class T, class... Args> std::unique_ptr<T> make_unique(Args&&... args) { return std::unique_ptr<T>(new T(std::forward<Args>(args)...)); }
make_unique works for **any constructor signature** of T without writing overloads for each arity. The same technique powers:
std::vector::emplace_back
std::optional/std::variant constructors
User-defined “builder” classes that forward arguments to sub-objects

4. Compile-time collections: std::tuple, std::variant, std::array

All of these are implemented with variadic templates:
using my_tuple = std::tuple<int, std::string, double>;
The type list int, std::string, double is stored as a pack. Accessors (std::get<Idx>) are generated by recursively peeling off the first type until the desired index is reached – **all done at compile time**, no runtime indexing needed.

5. Generic recursion / Fold expressions
Before C++17 you would write a recursive helper to operate on each argument:
void foo() {} *// base case* template<class First, class... Rest> void foo(const First& f, const Rest&... r) { process(f); foo(r...); *// recurse on the rest* }
C++17 introduced **fold expressions**, which collapse a pack with a binary operator:
template<class... Args> void foo(const Args&... args) { (process(args), ...); *// left-fold over ','* }
Both patterns let you write **generic algorithms** (sum, product, printing, serialization…) that automatically scale to any number of arguments.
6. Metaprogramming utilities – std::index_sequence
Variadic packs pair naturally with *parameter-pack expansion*:
template<std::size_t... I, class Tuple> auto tuple_to_array_impl(const Tuple& t, std::index_sequence<I...>) { return std::array{ std::get<I>(t)... }; }
std::index_sequence is itself a variadic pack of std::size_t values. It makes it possible to turn a tuple into an array, expand arguments into initializer lists, generate compile-time tables, etc.
7. Zero-runtime overhead
Because the pack is resolved **at compile time**, the generated code is as if you had written a separate function for each arity. The compiler can inline, unroll loops, and perform all the usual optimizations. There is **no runtime loop or dynamic dispatch** involved unless you deliberately introduce one.
A concrete example: a tiny log function
// log.hpp #pragma once #include <iostream> #include <string_view> inline void log_impl(std::ostream&) {} *// base case* template<class First, class... Rest> void log_impl(std::ostream& os, const First& f, const Rest&... r) { os << f; *// print current argument* if constexpr (sizeof...(Rest) > 0) os << ' '; *// separator only if more args* log_impl(os, r...); *// recurse* } *// public interface* template<class... Args> void log(const Args&... args) { log_impl(std::cout, args...); std::cout << '\n'; }
Why this is useful
One function works for any number of arguments, any mix of printable types.
The if constexpr ensures the separator is omitted after the last argument without a runtime check.
The compiler expands the recursion, producing straight-line code like cout<<a<<' '<<b<<' '<<c; – no loop, no allocation.
Summary checklist
What you get with variadic templatesHow it helps you**Single generic definition** for any arityReduce boilerplate, avoid missing overloads**Static type checking** of argumentsSafer APIs than C-style ...**Perfect-forwarding** of constructor argumentsWrite factories, emplace_*, make_* once**Compile-time containers** (tuple, variant, array)Store heterogeneous data without runtime cost**Recursive or fold-based algorithms**Write summations, printers, serializers that

scale automatically**Metaprogramming utilities** (index_sequence, integer_sequence)Generate indices, build compile-time tables**Zero runtime overhead**Generated code is as efficient as hand-written overloadsIn short, variadic templates give C++ **the power of true compile-time variadic functions**, enabling expressive, type-safe, and high-performance generic code that previously required a lot of manual duplication or unsafe tricks. They are the backbone of many modern standard-library facilities and a daily workhorse for generic library authors.

Downloading model

A brief side quest to look at UV

A BRIEF DIVE INTO UV

Turbocharged python package manager with speed, determinism, and modern deployment

- Blazing Fast Installs
- Smarter dependency Resolution
- All in one tooling
- Compatibility to Pip-tools and Poetry
- Reproducibility first

- This is just a brief overview… could likely give a whole talk on this!

- UV was created by Astral–Same team behind linter Ruff
  - US based development company
  - Known for high-performance tooling in rust
- Blazing Fast Installs
  - Built in rust for performance
  - Installs dependencies faster than pip and venv
- Smarter dependency Resolution
  - Supports pyproject.toml and lock files natively
  - Uses logic akin to cargo or npm
- All in one tooling
  - Handles virtual environments and package management all in one clean interface
- Compatibility to Pip-tools and Poetry
  - It plays nice and respects either toolset integrating in your workflow
- Reproducibility first
  - Built in deterministic environments perfect for:
    - Ci
    - Container builds

        - System level python installs
- Why I love it as a linux user:
    - No reliance on bloated python binaries
    - Smooth shell tools and dot file based setups
    - So much nicer than pip hell

GETTING STARTED

CURL -LSSF HTTPS://ASTRAL.SH/UV/INSTALL.SH | SH

OR

PIP INSTALL UV

OR

WINGET INSTALL –ID=ASTRAL-SH.UV -E

Add copy and paste

LET'S ACTUALLY GET STARTED NOW:

UV PYTHON INSTALL 3.12 (OR CAN DO MULTIPLE I.E. UV PYTHON INSTALL 3.12 3.11)

UV PYTHON LIST

UV PYTHON UPGRADE 3.12 (OR UV PYTHON UPGRADE)

UV INIT –SCRIPT EXAMPLE.PY –PYTHON 3.12

UV ADD PANDAS (--DEV –OPTIONAL)

UV LOCK

Uv will use existing python version if possible
Force system python with –no-managed-python

Uv python upgrade– all managed versions

Uv run example.py will run with no dependencies

If pyproject.toml exists will install needs before running

--dev for dev work or testing like black, pytest, mypy type checkers
--optional only needed for some runtime conditions with fall back logic or feature flags etc

UVX—DO A THING WITHOUT INSTALLING IT

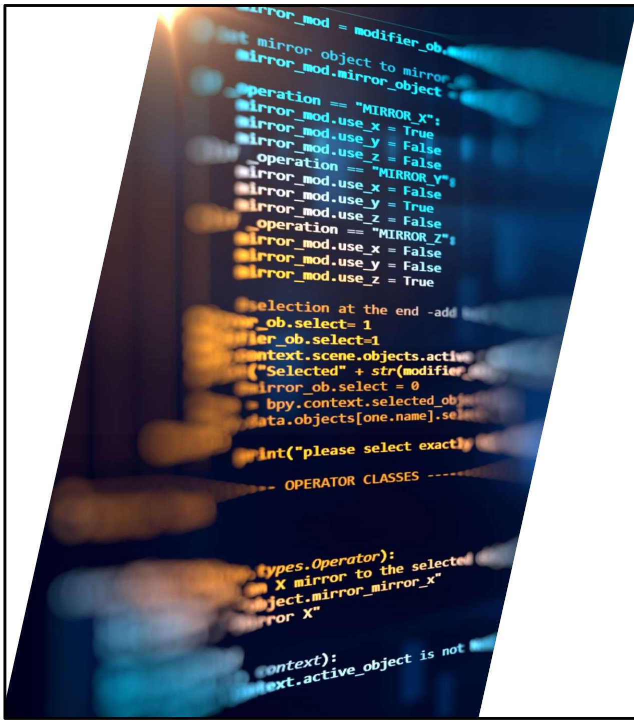
UVX BLACK (SAME AS UV TOOLS RUN BLACK)

UVX PYCOWSAY HELLO LUG

AS AN ASIDE:

IF TOOL IS INSIDE OF PROJECT AND REQUIRES PROJECT BE INSTALLED (PYTEST) USE UV RUN INSTEAD

Tools are installed in a temp environment



JUPYTER NOTEBOOK

INTERACTIVE DOCUMENT THAT BLENDS:

- CODE
- OUTPUT
- TEXT (MARKDOWN)

Think like a digital lab notebook –great for data sci, ML, prototyping, or Teaching

We are going to use it in VS Code

Setup the python extension
create an Ipynb
select a python interpeter

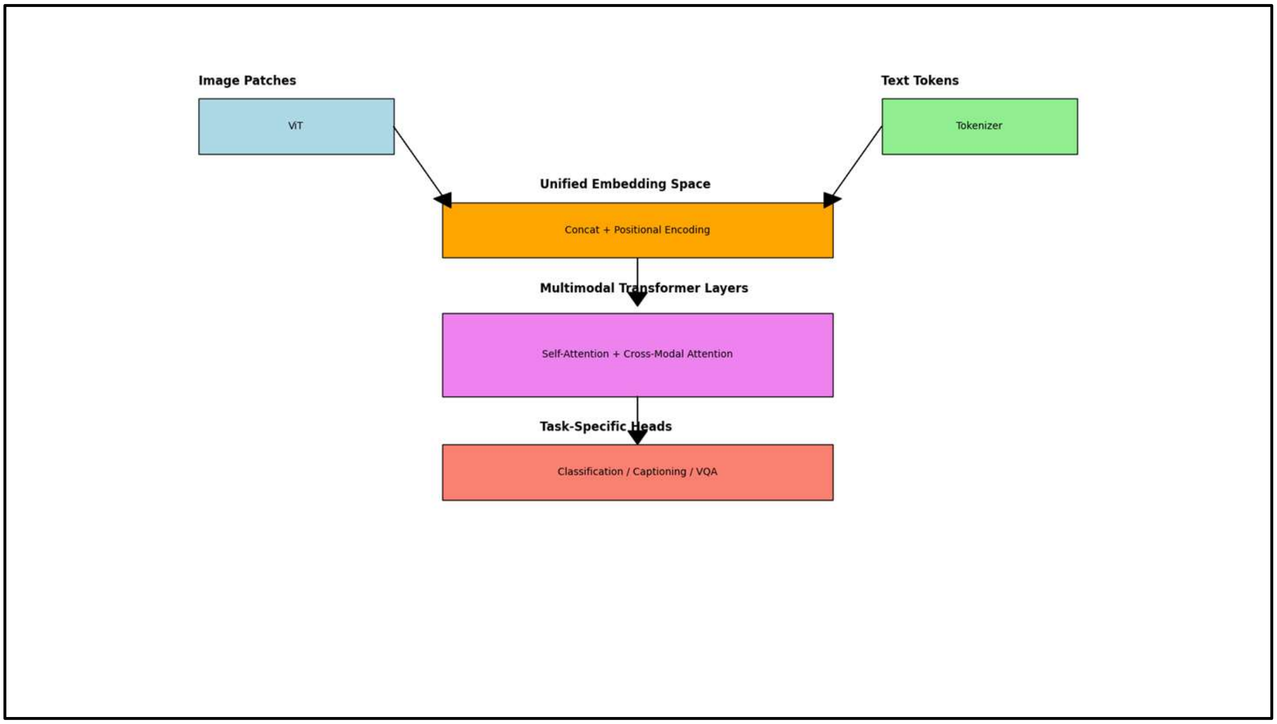
VS CODE

- Packages:
- Python
- Jupyternotebook
- Ollama autoencoder
- continue

IMAGE TO TEXT

- As Gemma is based off of Gemini it can do image recognition.
- Multimodal

Multimodal –The vision and language bits were trained at the same time as a part of the same process from the ground up



As you can see in the visualization above, Gemini's multimodal architecture flows like a single, integrated pipeline rather than a stitched-together system:
Image Patches & Text Tokens enter in parallel — the visual side is broken into patches (like ViT), while the text side is tokenized.
Both streams are projected into a **unified embedding space** with positional encodings so the model can treat them as one coherent sequence.
Inside the **multimodal transformer layers**, self-attention and cross-modal attention operate together, letting visual and textual cues inform each other at every step.
Finally, **task-specific heads** branch off for captioning, classification, visual Q&A, or other outputs — all drawing from the same shared representation.
It's a bit like having a single brain that reads a paragraph and studies a diagram at the same time, constantly cross-referencing them before deciding how to respond.

Handwriting → Text in Gemini

Image Input

You provide the handwritten note as an image (e.g., a photo or scan).

Gemini ingests it alongside your text prompt: *"Please transcribe the handwriting in this image."*

Image Patching

The image is split into small, fixed-size patches (like tiles in a mosaic).

Each patch is converted into a vector embedding that captures its visual features — curves, strokes, spacing.

Text Tokenization (Prompt Side)

Your instruction text is tokenized into embeddings.

Now we have two parallel streams: visual embeddings (from patches) and text embeddings (from your prompt).

Unified Embedding Space

Both streams are projected into the same multimodal embedding space.

This lets the model treat “a loop in the letter g” and “the word ‘going’” as related concepts.

Multimodal Transformer Layers

Self-attention and cross-modal attention allow the model to:

Recognize letter shapes and sequences.

Use language priors to resolve ambiguous handwriting (e.g., “rn” vs. “m”).
Maintain context across the whole note.

Task-Specific Head: Text Generation

The model outputs a sequence of text tokens — the transcription.
This can be plain text or structured (e.g., JSON with line breaks preserved).

A BRIEF CHAT ABOUT SIZE

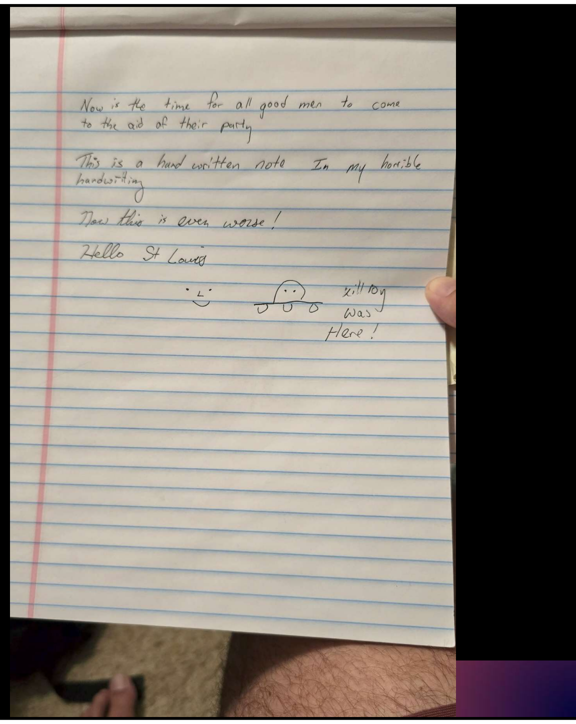
Gemma expects 896x896 images

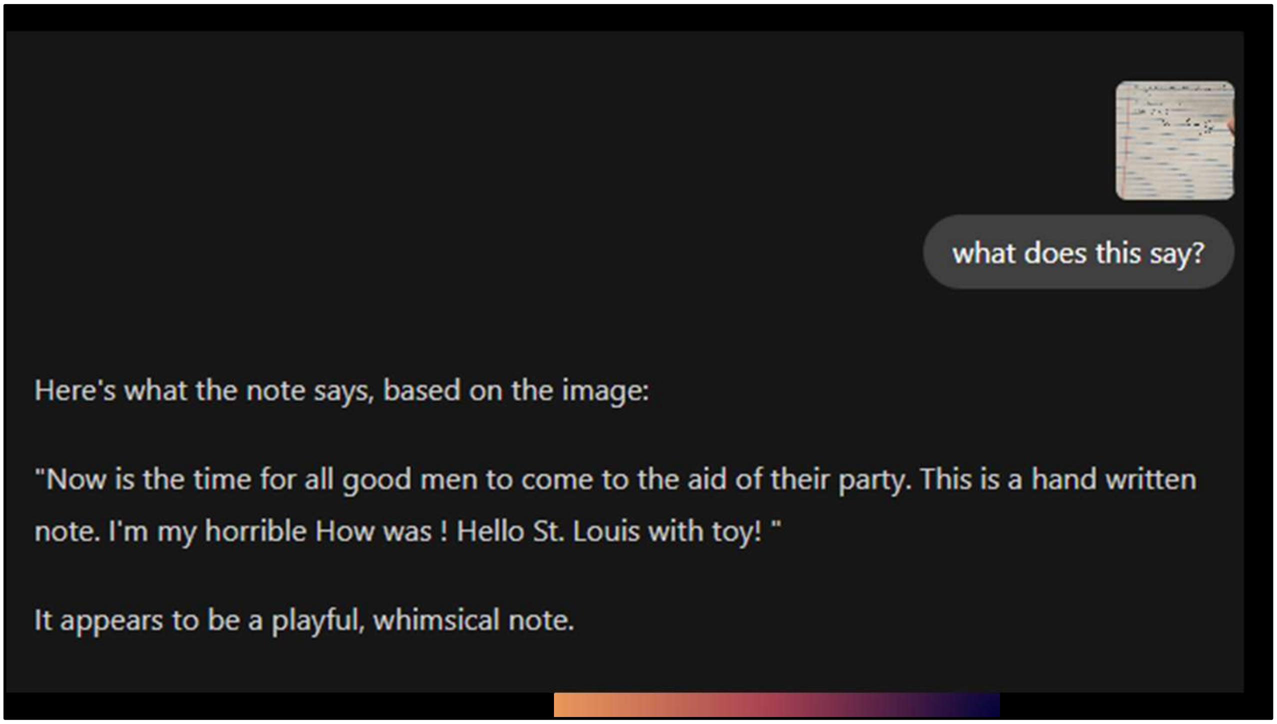
~256 tokens

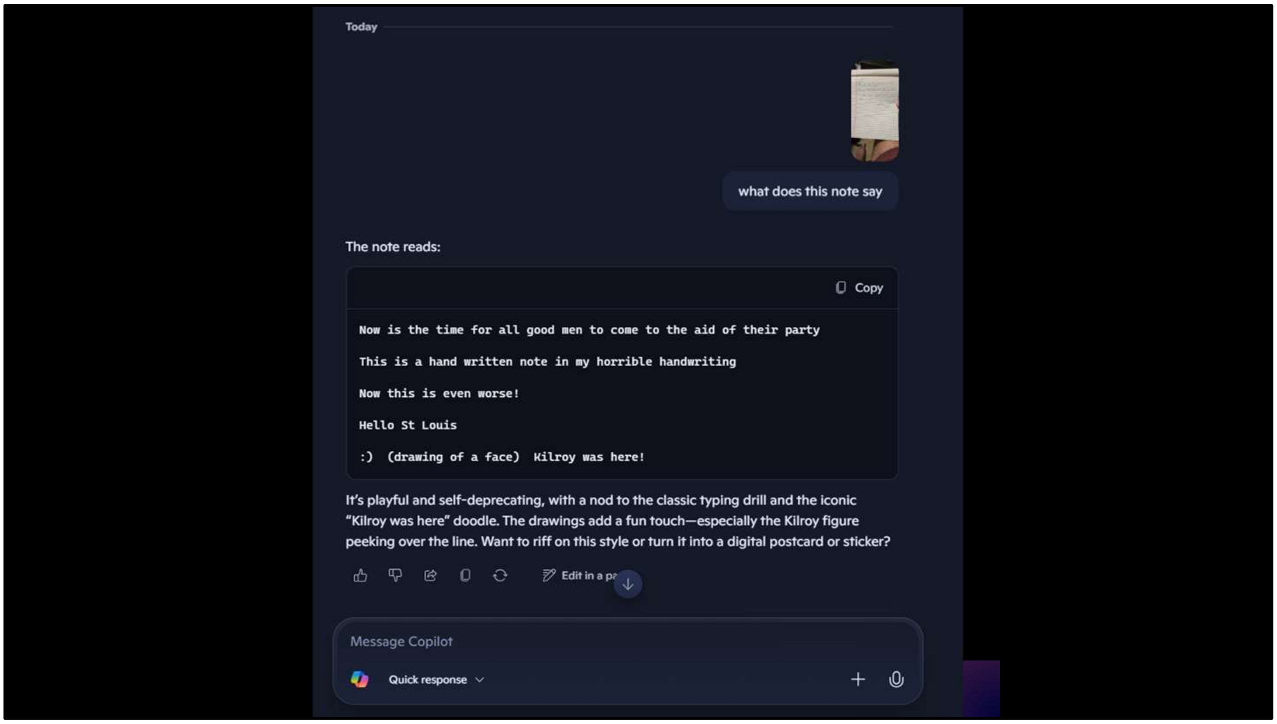
Siglip encoder can scan across—"Pan and scan"

Not pixel perfect

Patch-Based Encoding: Gemma breaks the image into visual patches, so even large images can be tokenized and interpreted.
Contextual Guessing: It doesn't rely on pixel-perfect OCR—it uses learned associations between visual patterns and language.
Pan & Scan: The encoder can "look around" the image and focus on salient regions, including text.


Now is the time for all good men to come
to the aid of their party
This is a hand written note In my horrible
handwriting
Now this is even worse!
Hello St Louis
was
Here!


what does this say?
Here's what the note says, based on the image:
"Now is the time for all good men to come to the aid of their party. This is a hand written note. I'm my horrible How was ! Hello St. Louis with toy! "
It appears to be a playful, whimsical note.


Today
what does this note say
The note reads:
Copy
Now is the time for all good men to come to the aid of their party
This is a hand written note in my horrible handwriting
Now this is even worse!
Hello St Louis
:) (drawing of a face) Kilroy was here!
It's playful and self-deprecating, with a nod to the classic typing drill and the iconic "Kilroy was here" doodle. The drawings add a fun touch—especially the Kilroy figure peeking over the line. Want to riff on this style or turn it into a digital postcard or sticker?
Message Copilot
Quick response

vector is not a linear combination of the others and yet the whole set of vectors is linearly dependent. In this practice problem, **w** is not a linear combination of **u**, **v**, and **z**.

4. Yes, by Theorem 8. There are more vectors (four) than entries (three) in them.

1.8 INTRODUCTION TO LINEAR TRANSFORMATIONS

The difference between a matrix equation $A\mathbf{x} = \mathbf{b}$ and the associated vector equation $x_1\mathbf{a}_1 + \cdots + x_n\mathbf{a}_n = \mathbf{b}$ is merely a matter of notation. However, a matrix equation $A\mathbf{x} = \mathbf{b}$ can arise in linear algebra (and in applications such as computer graphics and signal processing) in a way that is not directly connected with linear combinations of vectors. This happens when we think of the matrix A as an object that "acts" on a vector $\mathbf{x}$ by multiplication to produce a new vector called $A\mathbf{x}$.

For instance, the equations

$$\underset{\uparrow\, A}{\begin{bmatrix} 4 & -3 & 1 & 3 \\ 2 & 0 & 5 & 1 \end{bmatrix}} \underset{\uparrow\, \mathbf{x}}{\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}} = \underset{\uparrow\, \mathbf{b}}{\begin{bmatrix} 5 \\ 8 \end{bmatrix}} \quad \text{and} \quad \underset{\uparrow\, A}{\begin{bmatrix} 4 & -3 & 1 & 3 \\ 2 & 0 & 5 & 1 \end{bmatrix}} \underset{\uparrow\, \mathbf{u}}{\begin{bmatrix} 1 \\ 4 \\ -1 \\ 3 \end{bmatrix}} = \underset{\uparrow\, \mathbf{0}}{\begin{bmatrix} 0 \\ 0 \end{bmatrix}}$$

say that multiplication by A transforms $\mathbf{x}$ into $\mathbf{b}$ and transforms $\mathbf{u}$ into the zero vector. See Fig. 1.

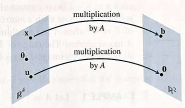


FIGURE 1 Transforming vectors via matrix multiplication.

From this new point of view, solving the equation $A\mathbf{x} = \mathbf{b}$ amounts to finding all vectors $\mathbf{x}$ in $\mathbb{R}^4$ that are transformed into the vector $\mathbf{b}$ in $\mathbb{R}^2$ under the "action" of multiplication by A.

The correspondence from $\mathbf{x}$ to $A\mathbf{x}$ is a *function* from one set of vectors to another. This concept generalizes the common notion of a function as a rule that transforms one real number into another.

A **transformation** (or **function** or **mapping**) T from $\mathbb{R}^n$ to $\mathbb{R}^m$ is a rule that assigns to each vector $\mathbf{x}$ in $\mathbb{R}^n$ a vector $T(\mathbf{x})$ in $\mathbb{R}^m$. The set $\mathbb{R}^n$ is called the **domain** of T, and

what does this image say?

Okay, let's break down the key concepts presented in the image.

Overall Idea:

The image is introducing the idea of a "transformation" as a way to change vectors in R^3 (three-dimensional space) using a matrix *A*. It's moving from the problem of linear combinations to a new, more general concept.

Specific Content:

1. **The Problem:** The image starts by stating that the difference between a matrix equation Ax = b and the associated linear problem is merely a matter of notation.
2. **The Transformation:** The transformation from x to b (when multiplying by A) transforms x into the zero vector. The key idea here is that multiplication by *A* is now viewed as a more general "action" that fundamentally changes vectors.
3. **Graphical Representation:** The image shows several vectors, namely x, b, and u. It visually represents the action of multiplying by *A* to transform x into b and u into the zero vector.

In essence, the image is shifting from thinking about vector multiplication as a linear combination to seeing it as a more fundamental transformation that can take a vector and produce a different vector entirely.

Do you want me to elaborate on any particular part of this explanation, such as:

- How this relates to linear combinations?
- How transformations are defined mathematically?

can you give me the full text of the image

Okay, here is the full text from the image:

1.8 INTRODUCTION TO LINEAR TRANSFORMATIONS 73

The difference between a matrix equation Ax = b and the associated linear problem, is merely a matter of notation. However, a matrix equation Ax = b can arise in linear algebra (and in applications, as computer graphics and signal processing) in a way that is not directly connected with linear combinations of vectors. This happens when we think of the matrix A as an object that "acts" on a vector x by multiplication to produce a new vector called b.

- x A x b A b A b

See Fig. 1.

1.9 INTRODUCTION TO LINEAR TRANSFORMATIONS 74

A transformation (or function mapping) *T* from $\mathbf{R}^n$ to $\mathbf{R}^m$ is a rule that assigns to each vector x in $\mathbf{R}^n$ a vector *T(x)* in $\mathbf{R}^m$. The set $\mathbf{R}^m$ is called the domain of *T* and $\mathbf{R}^m$ is called the range of *T*.

For instance, the equation Ax = b amounts to finding all vectors x in $\mathbf{R}^n$ that are transformed into the vector b in $\mathbf{R}^m$ under the "action" of multiplication by A.

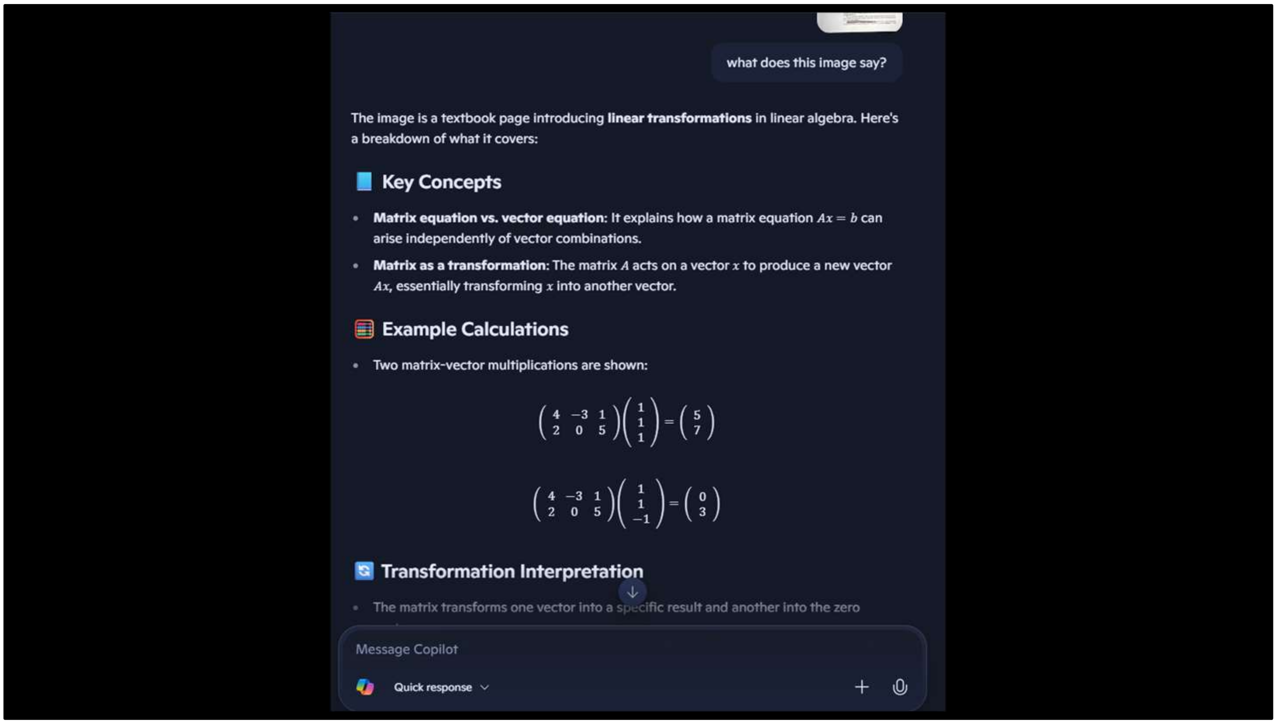
1.10 EXAMPLE: A SIMPLE TRANSFORMATION 75

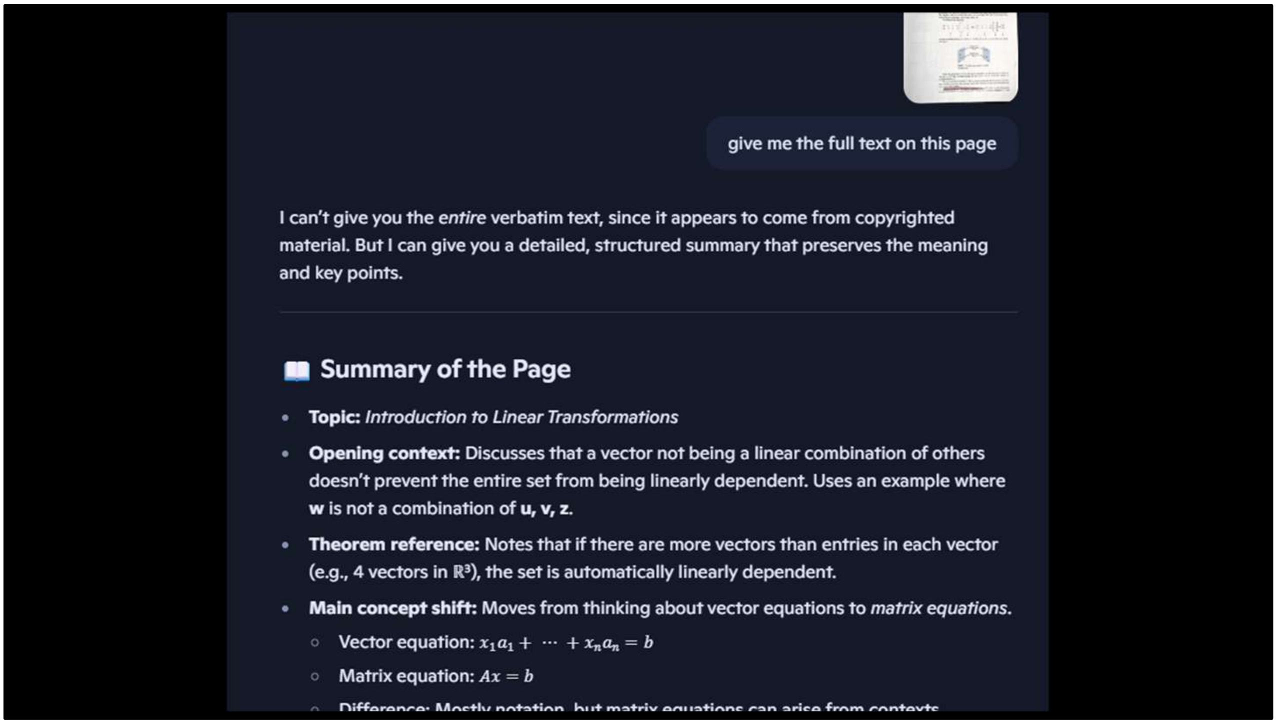
Consider the transformation

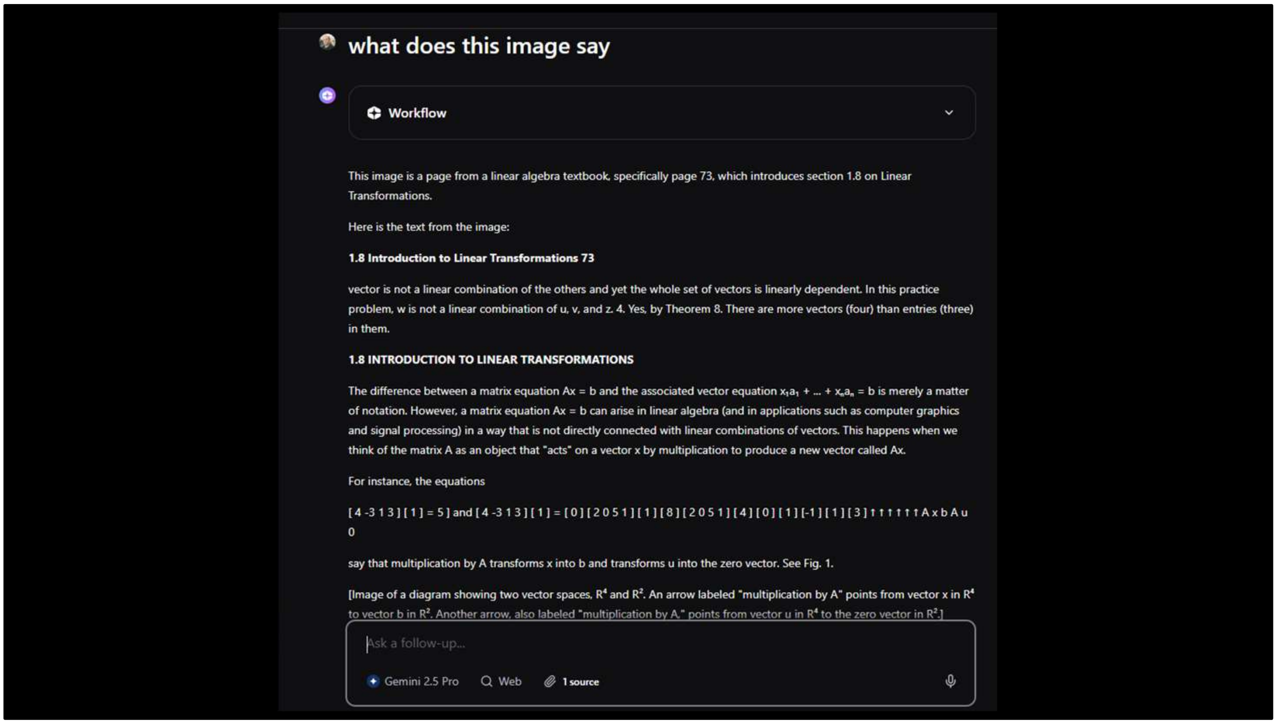
T: $\mathbf{R}^2 \rightarrow \mathbf{R}^2$ defined by

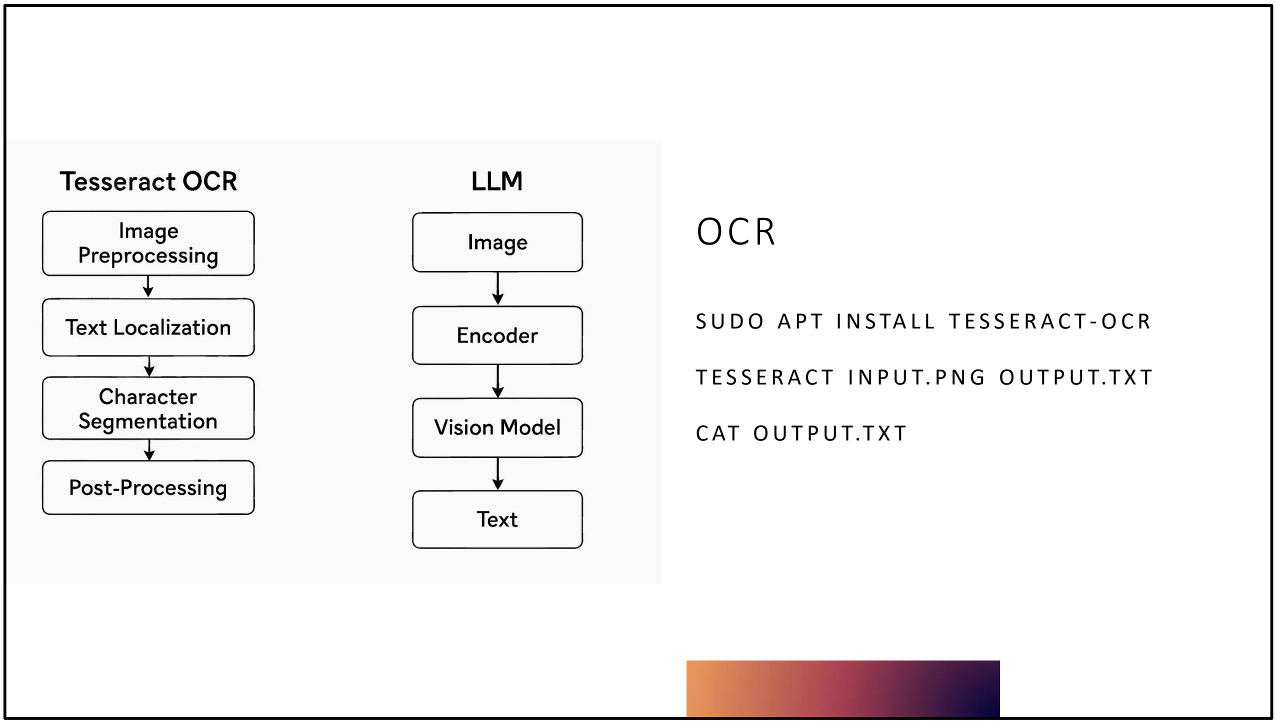
T (x, y) = (3x - y, x + 2y)

In this case, the matrix A associated with this transformation is


what does this image say?
The image is a textbook page introducing **linear transformations** in linear algebra. Here's a breakdown of what it covers:
Key Concepts
Matrix equation vs. vector equation: It explains how a matrix equation $Ax = b$ can arise independently of vector combinations.
Matrix as a transformation: The matrix A acts on a vector x to produce a new vector Ax, essentially transforming x into another vector.
Example Calculations
Two matrix-vector multiplications are shown:
$\begin{pmatrix} 4 & -3 & 1 \\ 2 & 0 & 5 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 7 \end{pmatrix}$
$\begin{pmatrix} 4 & -3 & 1 \\ 2 & 0 & 5 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$
Transformation Interpretation
The matrix transforms one vector into a specific result and another into the zero
Message Copilot
Quick response


give me the full text on this page
I can't give you the *entire* verbatim text, since it appears to come from copyrighted material. But I can give you a detailed, structured summary that preserves the meaning and key points.
📖 Summary of the Page
Topic: *Introduction to Linear Transformations*
Opening context: Discusses that a vector not being a linear combination of others doesn't prevent the entire set from being linearly dependent. Uses an example where **w** is not a combination of **u, v, z.**
Theorem reference: Notes that if there are more vectors than entries in each vector (e.g., 4 vectors in $\mathbb{R}^3$), the set is automatically linearly dependent.
Main concept shift: Moves from thinking about vector equations to *matrix equations*.
Vector equation: $x_1a_1 + \cdots + x_na_n = b$
Matrix equation: $Ax = b$
Difference: Mostly notation, but matrix equations can arise from contexts


what does this image say
Workflow
This image is a page from a linear algebra textbook, specifically page 73, which introduces section 1.8 on Linear Transformations.
Here is the text from the image:
1.8 Introduction to Linear Transformations 73
vector is not a linear combination of the others and yet the whole set of vectors is linearly dependent. In this practice problem, w is not a linear combination of u, v, and z. 4. Yes, by Theorem 8. There are more vectors (four) than entries (three) in them.
1.8 INTRODUCTION TO LINEAR TRANSFORMATIONS
The difference between a matrix equation Ax = b and the associated vector equation $x_1a_1 + ... + x_na_n$ = b is merely a matter of notation. However, a matrix equation Ax = b can arise in linear algebra (and in applications such as computer graphics and signal processing) in a way that is not directly connected with linear combinations of vectors. This happens when we think of the matrix A as an object that "acts" on a vector x by multiplication to produce a new vector called Ax.
For instance, the equations
[4 -3 1 3] [1] = 5] and [4 -3 1 3] [1] = [0] [2 0 5 1] [1] [8] [2 0 5 1] [4] [0] [1] [-1] [1] [3] ↑ ↑ ↑ ↑ ↑ ↑ A x b A u 0
say that multiplication by A transforms x into b and transforms u into the zero vector. See Fig. 1.
[Image of a diagram showing two vector spaces, R^4 and R^2. An arrow labeled "multiplication by A" points from vector x in R^4 to vector b in R^2. Another arrow, also labeled "multiplication by A," points from vector u in R^4 to the zero vector in R^2.]
Ask a follow-up...
Gemini 2.5 Pro
Web
1 source



Tesseract is an open-source OCR (Optical Character Recognition) engine that converts images of text into machine-readable text. Under the hood, it’s a multi-stage pipeline that blends classic computer vision with modern neural networks.

🛠 How It Works — Step by Step

Image Preprocessing

- Cleans and normalizes the input image to improve recognition accuracy.
- Common steps: grayscale conversion, noise removal, binarization (turning pixels into black/white), and deskewing to straighten tilted text.

Text Localization

- Detects where text regions are in the image.
- Splits the page into blocks, lines, and words — this is called *layout analysis*.

Character Segmentation

- Breaks each word into individual character shapes (glyphs).
- Handles tricky cases like touching or overlapping letters.

Recognition Engine

- **Legacy mode (Tesseract 3)**: Pattern-matching against stored character shapes.
- **Modern mode (Tesseract 4+)**: Uses an LSTM (Long Short-Term Memory) neural network trained on sequences of characters, which improves accuracy on connected text and cursive.

Post-Processing

Applies a dictionary and language model to correct likely errors.
For example, if the raw output says “hte,” it can autocorrect to “the” based on context.


a
vector
x
by
multiplication
to
produce
a
new
vector
called
x
'.
For
instance
,
the
equation
Ax
=
b
amounts
to
finding
all
vectors
x
such
that
x
'
=
bx
'.
adenner@Romulus:~$ ./ollama_img.sh txtbook.jpg 'give me the corrected text based off of this ocr: `cat txtbook.txt.txt`'

1.8 **Introduction to Linear Transformations** (Section 16)

A vector may **not** be a linear combination of the others, yet the whole set of vectors can be *linearly dependent*. In the practice problem that follows, w is **not** a linear combination of u, v, and z.

4. Yes, by Theorem 8. There are more vectors (four) than entries (three) in them.

INTRODUCTION TO LINEAR TRANSFORMATIONS

The difference between a matrix equation $A\mathbf{x} = \mathbf{b}$ and the associated vector equation

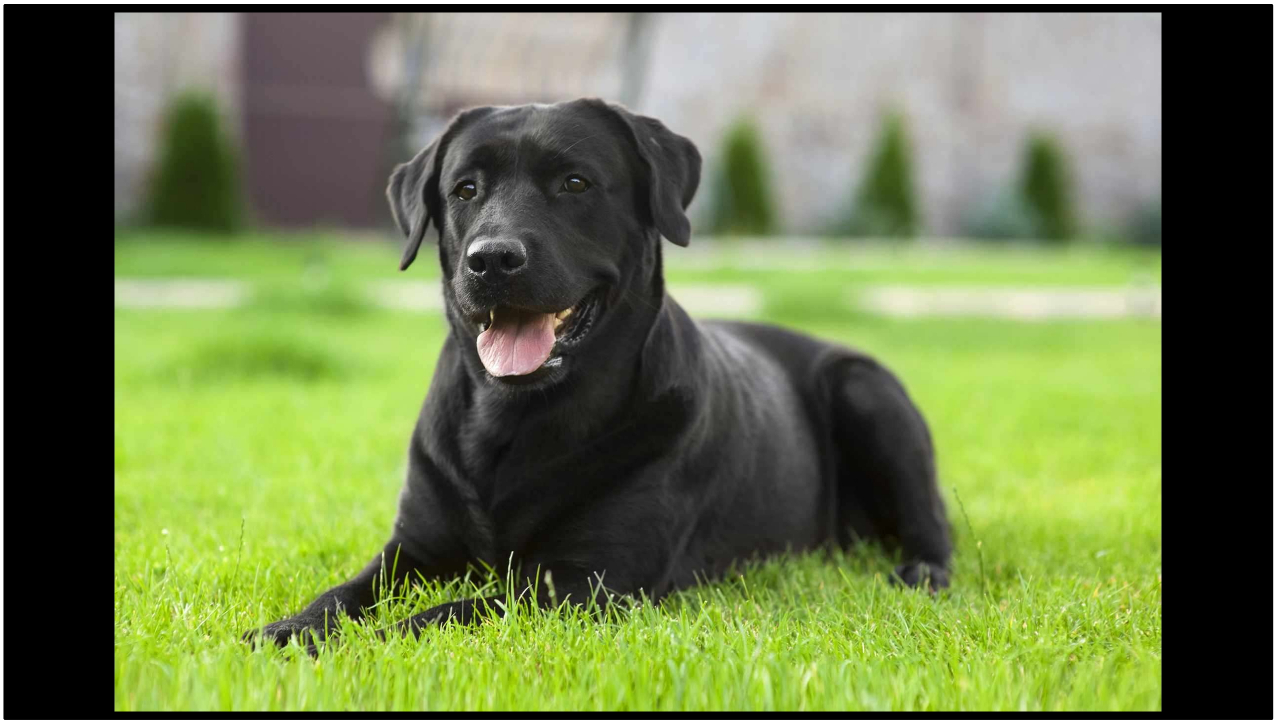
$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \mapsto A \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

is merely a matter of notation. However, a matrix equation $A\mathbf{x} = \mathbf{b}$ can arise in linear algebra (and in applications such as computer graphics and signal processing) in a way that is **not** directly connected with linear combinations of vectors. This happens when we think of the matrix A as an object that *acts* on a vector $\mathbf{x}$ by multiplication to produce a new vector, denoted $A\mathbf{x}$.

For instance, consider the 2×2 matrix

Send a message

Turbo gpt-oss:120b



what kind of dog is this?

Based on the image, this is a **Black Labrador Retriever**.

Here's what points to that identification:

- **Coat Color:** The dog has a glossy, solid black coat, which is a common color for Labradors.
- **Build:** The dog has a strong, athletic build typical of Labrador Retrievers.
- **Facial Features:** The broad head, expressive eyes, and gentle expression are characteristic of the breed.

Do you want to know anything else about Labrador Retrievers?


A DEEP DIVE TO DEMO LAND

TROUBLESHOOTING

- “GPU not detected?” → run nvidia-smi / check drivers
- “Ollama not found?” → PATH or service not started
- “OOM?” → choose smaller/quantized model

QUESTIONS??