Haris Mahmood

AI Research Engineer

Islamabad, Pakistan | +92 335 5065740 | haris@harismahmood.dev
harismahmood.dev | linkedin.com/in/hmahmood24 | github.com/hmahmood24

Summary

AI research engineer with six years of work across document understanding, machine learning framework transpilers, and LLM agents, now focused on post-training, continual learning, and agent harnesses. At Unify (YC W23), from January 2025 to October 2026, built much of the platform under its AI teammates, from the RAG pipeline and generative UI to app integrations and agent workflows distilled into code, and led a ground-up redesign of its open-source agent harness, with an agent mesh and a memory built for both memoization, reusing work it had done before, and generalisation, transferring what it learned to new work. Before that, co-founded Ivy as CEO and technical lead and led its source-to-source ML transpiler, which rewrote PyTorch code as readable TensorFlow and JAX and was merged into Kornia; earlier, led the team on Ivy's graph tracer at Unify and built SmartRecruit's document understanding at CureMD.

Experience

Unify (YC W23) Jan 2025 – Oct 2026
Founding AI Engineer *Remote*
Startup that built Unify Interfaces, an observability and evaluation platform for LLM applications, then Unify Teammates, AI agents that companies onboarded like new employees and worked with over chat, email, and live calls.

- Led a ground-up redesign of Unify's open-source agent harness, where agents act by writing and running Python: a single code-execution tool, with every other capability a Python object in a persistent sandbox that the agent discovered progressively through on-demand help; model-written code moved out of the agent's process into a sealed sandbox; a lean context that compacted itself; and follow-up work forked from the agent's conversation to reuse the prompt cache. On Continual-ARC, an unpublished benchmark Unify was building on ARC-AGI to test continual learning, the rebuilt harness with memory on solved more puzzles than the harness before the rework at every model reasoning effort, 73.5% against 62.5% at low effort, and used less of its budget. Against OpenClaw, Hermes Agent, and Prime Agent with their own memory and the same model and settings, it solved more puzzles at every model reasoning effort and used less budget at medium and high effort, though more than OpenClaw and Prime Agent at low effort.
- Redesigned the agent's memory for both memoization, reusing work done before, and generalisation, transferring what it learned to new work. Each task was one commit in a git repository; after it the agent wrote up what it had done as a Python function, and a separate writer merged related experience into a library of functions and notes, trusting a function only once later tasks had used it. The acting agent, the writer, a librarian, and sandboxed sub-agents formed an agent mesh on one crash-safe, append-only log. On an internal office benchmark whose unwritten rules changed over time, the harness made 12 to 21 times fewer harmful mistakes than Prime Agent.
- Before the redesign, extended the harness's function and guidance stores so stored functions earned or lost trust from real reuse, were quarantined and patched in place when they broke, and were reviewed against the environment's checked outcome rather than the agent's own account; made the post-task review and context compression forks of the agent's conversation, so they read from the prompt cache.
- Built the engineering under the memory: a detached writer service that survived crashes and resumed model turns without paying for them twice, a spend cap that booked every call at its worst case so it could not be overshot, a prompt prefix kept byte-stable through forks, compaction, and provider routing, and model-written code confined with no network access or credentials.
- Built the RAG pipeline behind Unify Teammates, launched in March 2026. Documents were chunked along their own headings rather than fixed-size windows, tables that ran across pages were rejoined and kept whole as typed data the agent could query and total in code, and ingestion ran through one checkpointed engine with versioned leases that refused to report success while stored rows fell short of what the source declared.
- Ran ingestion on Google Cloud: uploads went straight to storage through single-object signed URLs before a job was published, so pods never held bucket credentials; parse and ingest workers on GKE scaled with the Pub/Sub backlog; and a private overlay per client, discovered by the open-source runtime through a plugin hook, kept client-specific code out of the core.
- Kept PostgreSQL fast under bulk ingestion: rebuilt the pipeline that embedded new records and wrote them into the vector search index, with maintenance that rebuilt the index without blocking searches; turned large deletions into batched phases that no longer held locks for minutes (one 20,000-embedding delete went from

15.9 s to 1.8 s in production); and replaced a uniqueness check that averaged 277 s in production with an indexed lookup.

- Built generative UI for the agents. From a plain-English request, a teammate wrote a React app, a canvas, on a component kit authored for it, passed it through lint, type, bundle, and real-browser render checks, and revised it from the screenshot; requests such as adding a section to email the report to a list of colleagues became forms wired to server-side actions. Each canvas ran in an opaque-origin frame with no network access, read data only through queries checked at creation, and could name an action but never see it; the server re-validated inputs against size-bounded schemas, asked for confirmation of anything irreversible, and ran each action exactly once.
- Built the agents' app integrations, connected through OAuth or API keys. Tools were disclosed progressively: an agent searched for the app and tool it needed from inside the code it wrote, loaded that tool's schema only when calling it, and called it there, keeping its context free of unused tools; every call passed through per-account permissions in which each action was allowed, asked about first, or blocked, risky actions asked by default, and an approval covered only the exact call it approved.
- Built, tested, and rolled out installable agent workflows: content installed at once and scheduled jobs switched on only when the required apps were connected; schedules ran on each user's own clock across daylight saving; and installs requested from the web app were claimed while the agent slept. A recurring task that had worked could be distilled into Python with model calls only for its judgement steps, making later runs cheaper and consistent, and a distilled version that kept failing was dropped so the agent planned the task afresh.
- Built the tables and dashboard editor through which teams explored their LLM traces and evaluations in Unify Interfaces, an observability and evaluation platform for LLM applications launched in March 2025, whose SDK captured detailed traces: nested column groups that dragged, pinned, and hid as units, grouping on the server with each group loading lazily and paging both ways, and a Zustand store hydrated from Next.js server components; and shaped the SQL generated from users' filters so PostgreSQL planned it well as logs grew into the millions of rows.
- In the Teammates web app, built the shell that kept chat, live activity, and calls running across navigation, the workflows and integrations screens, and responsive layouts with progressive loading; deployed pilots for prospective clients and supported their teams directly.

Ivy (Unify spin-out) 2024 – Jan 2025
Co-founder and CEO *Remote*
Company built around an open-source, source-to-source ML transpiler that rewrote PyTorch code as TensorFlow and JAX.

- Co-founded Ivy with colleagues from Unify and served as CEO and technical lead. Ran technical demonstrations for enterprise machine learning teams, including design partners whose platforms were built on TensorFlow while new models appeared in PyTorch, and set the roadmap from the operations their models needed.
- Led its source-to-source ML transpiler, rebuilt at the spin-out on an architecture first laid down at Unify in late 2023, after tracing had proved to bake control flow into converted models. It let PyTorch models and libraries use TensorFlow's and JAX's hardware and tooling, from TPUs to TensorFlow Lite, as editable source rather than the converted computation graph that tools such as ONNX produced. PyTorch code was parsed into syntax trees and lowered in stages, through Ivy's reimplementation of the PyTorch API and then Ivy itself, to readable TensorFlow or JAX source; because nothing was executed, loops and branches survived, and the output could be read, edited, and trained.
- Built the translation cache, which stored translations as serialisable syntax trees so shared code was translated once and regenerated from the cache, and the handling of PyTorch's in-place operations in immutable TensorFlow and JAX. Fixed the semantic gaps real libraries exposed, such as type checks that had to accept a translated parameter as a `tf.Variable`, and validated converted Hugging Face Transformers models by loading their pretrained weights, comparing outputs with the originals, and training some of them in the target framework.
- Worked through Kornia, a computer-vision library for PyTorch, until about 90% of a large set of its functions and classes transpiled; the maintainers merged the integration in October 2024 and shipped it in Kornia 0.7.4, giving Kornia TensorFlow, JAX, and NumPy support. Relicensed Ivy under Apache 2.0 in November 2024, ahead of the transpiler's source being published.

Noetico Ltd. Dec 2023 – Jun 2024
Senior Data and AI Engineer *Remote*

- Extended HunFlair, a biomedical entity recogniser, to new entity types by training on the sentences where it had missed an entity, and linked every entity to the client's own ontology of about 220,000 concepts, keeping a fact only when both of its entities linked, which filtered out false positives.
- Built an LLM agent for a biomedical question-answering service that turned a plain-English question into a TypeQL query from the graph's schema and worked examples, ran it against TypeDB, and wrote the answer from the result.

Unify (YC W23) Mar 2023 – Apr 2024
Machine Learning Research Engineer II *Remote*
Unify was then building Ivy, an open-source framework for making machine learning code portable across frameworks.

- Led the team building Ivy's graph tracer and transpiler, which traced a model into its computation graph and rebuilt that graph in PyTorch, TensorFlow, JAX, or NumPy through Ivy's frontends and backends, and interviewed engineering candidates for Unify. Made traces carry Python values as well as tensors, through proxies that behave like built-in types while recording what is done to them, so shapes and sizes stayed dynamic in converted models.
- Extended the Ivy framework that served as the common representation between frameworks: its functional API and backends, its PyTorch, TensorFlow, and JAX frontends, and its stateful API, where converted modules behaved as native ones, including PyTorch modules turned into trainable Keras models with tied weights and LSTMs.
- Adapted a fork of TensorFlow's AutoGraph to rewrite the user's source so built-ins such as `len`, which bypass the framework, appeared in the trace; made `jax.jit` work on transpiled Hugging Face models by generating pytree versions of their output classes; cut transpile time on very large graphs, such as FasterRCNN converted to JAX, from about 65 to about 25 minutes; and restructured an early prototype of source-to-source translation into the staged architecture the later transpiler was rebuilt on.

CureMD 2020 – Mar 2023
Machine Learning Engineer *Lahore, Pakistan*
Healthcare software company.

- Built SmartRecruit with one other engineer, a recruitment system built before large language models and used for CureMD's own hiring. It parsed resumes of any layout, length, or format, including multi-column and graphics-heavy documents, with over 90% accuracy, and scored candidates against the job, parsing and scoring each resume in 2–3 seconds against about three minutes for a recruiter to read one.
- Linked each extracted entity to its education or work entry by drawing the entities back onto the rendered page as boxes coloured by type and detecting entries with a custom YOLOv5 model, so dates and roles matched the right employer even when the layout scrambled the text. Trained and periodically fine-tuned the YOLOv5 detectors and spaCy entity-recognition models on labelled resume data.
- Chose headings by a vote among twelve candidate segmentations, from font rarity and underlines to a heading detector, and recovered columns and reading order by sweeping a vertical line from each heading until it crossed no text. Scored candidates with an explainable model over degree, institution, and grade, O*NET-normalised roles, and about 13,000 skill terms, and ran parsing and scoring as separate services on AWS behind RabbitMQ.

N-ovative Health Technologies Jul 2020 – Sep 2020
Embedded Developer *Islamabad, Pakistan*

- Designed the electronic circuit and control system of the NUST Powered Air Purifying Respirator and supported its quality testing and technical documentation. Worked in the team building the N-Saviour ventilator, on its software algorithms and hardware interfacing, during the COVID-19 equipment shortage.

Education

National University of Sciences and Technology (NUST) 2016 – 2020
Bachelor's in Electrical Engineering Islamabad, Pakistan
CGPA 3.69/4.00

Selected Open Source

- Unify agent harness: open-source harness in which agents act by writing code; led its ground-up redesign, with an agent mesh and a memory built for memoization and generalisation, and earlier worked in the full runtime on the agents' RAG pipeline, generative UI, integrations, and workflows.
- Ivy: led the open-source, source-to-source transpiler that rewrote PyTorch functions, models, and libraries as readable TensorFlow and JAX with the original control flow intact, published on PyPI.
- Kornia: worked through the library with Ivy's transpiler until about 90% of its functions and classes converted, and co-authored the integration, merged in October 2024 and shipped in Kornia 0.7.4, that gave it TensorFlow, JAX, and NumPy support.
- TensorLy: co-mentored the Google Summer of Code 2023 project to make TensorLy usable with every framework Ivy supported, letting its users switch backends through Ivy's backend handler instead of TensorLy's own.

Research Interests

Post-training of open models with reinforcement learning, verifiable rewards, and the environments that training needs; continual learning and continual post-training, including how to measure and limit forgetting; post-training on a company's own data; and agent harnesses that gather an agent's experience as training data.

Leadership and Collaboration

- **Leading engineers.** Led the Unify team that built Ivy's graph tracer and transpiler, interviewed engineering candidates for Unify's hiring, and co-mentored the Google Summer of Code 2023 project to make TensorLy usable with every framework Ivy supported. In 2026, led the redesign of Unify's agent harness.
- **Running a company.** Co-founded Ivy and served as CEO while remaining its technical lead, setting product direction from conversations with prospective users and taking the transpiler to a public release.
- **Working with clients.** Ran technical demonstrations of the transpiler for enterprise machine learning teams and turned their reports of unsupported operations into the roadmap. At Unify, deployed pilot environments for prospective clients, joined client meetings to turn business requirements into technical specifications, and supported clients directly with their questions, debugging, and training.
- **Experiments.** Works evidence-first, running the experiment or ablation before forming a hypothesis, comparing the result with what was expected, and tracing any gap to its cause in the agent's traces before reading the related work.

Technical Skills

Languages: Python, TypeScript, SQL, Bash
Machine learning: PyTorch, TensorFlow, JAX, Keras, Hugging Face Transformers, PEFT (LoRA), vLLM, YOLOv5, spaCy, HunFlair, scispaCy
LLM agents: Code-executing agents, multi-agent systems, agent memory, RAG, tool calling, prompt caching, sandboxing (bubblewrap, seccomp), OpenAI and OpenRouter APIs
Program transformation: Python AST transformation, static analysis, graph tracing, TensorFlow AutoGraph
Documents and data: Docling, PyMuPDF, Polars, openpyxl, OpenCV
Backend: FastAPI, Flask, PostgreSQL (query planning, JSONB), SQLAlchemy, Pydantic, MongoDB, TypeDB, RabbitMQ
Frontend: Next.js (App Router), React, generative UI, TanStack Query, TanStack Table, Zustand
Cloud and infrastructure: Google Cloud (GKE, Pub/Sub, Cloud Run, Cloud Build), AWS (ECS, S3), Kubernetes, Docker, MLflow

References

Daniel Lenton, Director and CEO, Unify AI Ltd dan@unify.ai