Evan Zimmerman

evanzimm@umich.edu | (703) 304-2447 |LinkedIn: linkedin.com/in/evan-zimmerman-3817541aa
Github: https://github.com/EvanZimmerman307?tab=repositories
U.S. Citizen — Top Secret Clearance (DODCAF Tier 5 Investigation: 11/30/22, Access Granted: 12/27/22)

EDUCATION

University of Michigan - Ann Arbor, MI *August 2024 - December 2025 (Expected)*
Master of Science in Computer Science and Engineering (GPA: 4.0/4.0)
Coursework: Foundations of AI, Advanced Compilers, Advanced Database Systems, Probability & Random Processes, Machine Learning, Advanced Scalable Systems for Generative AI, Natural Language Processing, Engineering Interactive Systems

University of Virginia - Charlottesville, VA *August 2020 - May 2024*
Bachelor of Science in Computer Science, *Minor in Data Science* (GPA: 3.9/4.0)
Coursework: Data Structures and Algorithms, Machine Learning, Regression Analysis, Database Systems, AI, Advanced Software Development, Computer Architecture, Operating systems, Distributed Systems, Data Science Systems, Probability, Mathematics of Statistics, Linear Algebra, Algorithmic Economics

RESEARCH EXPERIENCE

UMich Computer Aided Diagnosis AI Lab, *Machine Learning Research Assistant* – Ann Arbor, MI *Feb 2025 - Present*

- Designed and implemented a machine learning pipeline to predict the presence of Chagas disease from ECG signals using Dynamic Bayesian Networks (DBNs), modeling temporal dependencies among cardiac wave characteristics.
- Engineered interpretable time-series features from ECG signals, including onset and offset times of the P wave, QRS complex, and T wave, extracted via NeuroKit2-based delineation.
- Developed a novel discretization strategy by selecting one wave feature (e.g., R-peak) as a temporal anchor and encoding the occurrence of other features within time windows relative to it—enabling structured, interpretable modeling of temporal relationships in the DBN.
- Tuned model structure and trained conditional probability distributions using pgmpy; evaluated performance using patient-level cross-validation to prevent data leakage.
- Achieved 0.87 recall on the best-performing model, demonstrating high sensitivity in detecting true positive cases of Chagas disease.
- Published reproducible, modular codebase in Python with clear documentation, enabling future research extensions and potential integration with clinical screening tools.

UVA Biocomplexity Institute, ***High Performance Data Engineering Researcher*** – Charlottesville, VA *May 2024 - Present*

- Integrated Nvidia Collective Communications Library into the communication layer of Cylon, a high performance data engineering framework, to optimize GPU utilization for data engineering operations
- Implemented an all-to-all collective communication operation with a custom channel to optimize distributed join

UVA, *Machine Learning Stock Price Forecasting Researcher* – Charlottesville, VA *September 2023 - December 2023*

- Under the guidance of a UVA Professor, analyzed the predictive accuracy of various machine learning models in TensorFlow for forecasting next-day stock closing prices using historical price data
- Evaluated the accuracy of Long Short-Term Memory networks, random forest regressors, and linear regression, and compared various training features, such as volume and price data from similar companies

WORK EXPERIENCE

Amazon Web Services, *Software Development Engineer Intern* – New York, NY *May 2025 – August 2025*

- Designed and deployed a scalable architecture to host tokenizers from third-party foundation model providers (e.g., Anthropic), enabling customers to retrieve token counts without performing inference.

- Used TypeScript and AWS CDK to define infrastructure stacks with configurable compute profiles, autoscaling policies, and robust deployment pipelines; architecture supported flexible tokenizer descriptor parsing for dynamic service configurations.
- Chose SageMaker over ECS after performance and usability evaluation—favoring provider-side testing and operational simplicity despite slightly higher latency.
- Addressed cross-account networking and VPC isolation, enabling secure service-to-service calls between tokenizer hosting in a protected provider-facing account and customer-facing inference systems.
- Built a distributed request visibility tool to reconstruct the full lifecycle of requests across the multi-layer Bedrock–Anthropic hosting stack, reducing operator debugging time from hours to minutes.
- Gained hands-on experience working with LLM inference systems deployed on Kubernetes (EKS), and built the tool with an understanding of how requests traverse containerized microservices across Kubernetes clusters/layers.
- Implemented core logic in Python, and used TypeScript CDK to provision infrastructure for securely accessing logs across services; handled fine-grained IAM and resource policies for traversing highly isolated layers of the Bedrock stack.
- Surfaced structured operator-facing summaries, enriched with request metadata and links to CloudWatch log streams, improving incident triage, root cause analysis, and operational observability.
- Assisted with E2E integration tests for an automated model onboarding service
- Assisted with the deployment of image updates to Sonnet 3.5 v2, 3.7, and the release of Sonnet 4.1 across 1000s of GPU/Tranium instances running in Kubernetes Clusters in AWS regions in the world

AnalystKit, ***Data Science Intern*** – Charlottesville, VA *September 2023 – September 2024*

- Prototyped a RAG-powered AI chatbot with Python, Langchain, Chroma DB, and OpenAI API to explain investing concepts to users
- Developed a model in Python that leveraged historic stock return data to forecast the covariance of stock returns using a weighted, multi-predictor IEWMA methodology
- Utilized covariance forecasts to develop a Python program that constructed optimal portfolios for select portfolio strategies

EY, ***Forensic Technology Intern*** – New York, NY *June 2023 – August 2023*

- Implemented an efficient NLP model for generating word embeddings in Python through probabilistic methods, pointwise mutual information, and SVD
- Applied the model to execute similarity searches to identify whether a given text contained information connecting a client to financial crime

Allied Associates International, *Software Engineer Intern* – Gainesville, VA May 2022 – July 2022

- Wrote a Python program that performed network protocol analysis on packet captures of over 10 GB, sent from IoT devices, which led to discovering device-identifying patterns in network data
- Fixed a program, written in C#, that was failing in production to automate packet capture analysis for an IoT device
- Conducted packet forensics using Wireshark on network data sent from IoT devices that were producing incomplete results in production

PROJECT EXPERIENCE

UVA, *NCAA Basketball Focus*

- Designed a Polynomial Support Vector Machine and a Random Forest Classifier with Scikit-learn to recommend a basketball player's ideal position, yielding accuracies of 91% and 88% respectively - won 1st place in the 2022 Machine Learning for Virginia competition
- Analyzed position groupings of over 19,000 unique players across 10 seasons using K-means clustering, identifying key performance metrics that differentiate each position

UVA, *Schedule Advisor*

- Built a web application with Django and PostgreSQL that enabled UVA students to search for courses, create a schedule, and send it to an advisor for approval
- Facilitated team progress as Scrum Master by leading biweekly stand-ups to set clear sprint goals and define responsibilities for each team member
- Developed a course search feature with an API that interfaced with the UVA course database

UVA, *AI 2048*
- Created a Python-based clone of the puzzle game 2048 that was playable by both humans and AI agents
- Implemented Expectimax and Monte Carlo Tree Search algorithms to beat 2048
- Through experimentation and testing, achieved a 65% win rate and 3 minute mean game time with the Monte Carlo Tree Search algorithm

UVA, *Job Board*
- Worked in a team of four to create a web application that presented thousands of real job opportunities across all industries and experience levels, and enabled users to create and apply to job postings as well
- Designed the database schema, hosted the MySQL database on Google Cloud Platform, and populated the database with a Kaggle dataset containing thousands of LinkedIn job postings
- Implemented features to update, delete, and search for job postings with TypeScript and Prisma to query the database hosted on Google Cloud Platform

UVA, *Raft*
- Implemented the Raft consensus algorithm in Go to achieve distributed consensus in a fault-tolerant system, enhancing the reliability and consistency of data across multiple nodes through heartbeats, leader election, log-replication, and persistence

UMich, *ReviewGenie* – *Python, Flask, JavaScript, React, MongoDB, Llama 3, Beautiful Soup*
- Developed an LLM-powered web application to help businesses solicit meaningful product reviews and enable users to perform natural language searches for relevant product reviews – led backend development and competed in MHacks 17
- Scraped product websites and utilized Llama 3 to generate comprehensive product profiles and targeted review prompts.
- Engineered backend functionalities with Flask to categorize review submissions, store reviews in MongoDB, and search for meaningful reviews based on user queries by processing queries with Llama 3 and filtering reviews in MongoDB accordingly

UMich, *Energy Optimized LLVM* – *Python, LLVM, CompilerGym, PyTorch*
- Developed a reinforcement learning framework for optimizing LLVM compiler pass ordering to reduce code size and statically estimated dynamic energy consumption, outperforming traditional optimization levels (-O3, -Oz) on multiple benchmarks.
- Implemented a modular static energy estimation model based on ARM Cortex-A7 assembly and LLVM IR analysis, enabling static reward calculations for change in energy consumption during training
- Trained reinforcement learning agents using Proximal Policy Optimization (PPO) to iteratively refine compiler optimizations, balancing trade-offs between energy efficiency and code size
- Extended CompilerGym's LLVM reinforcement learning environment to customize reward evaluation, integrating energy and code size reduction metrics into the reward function

UMich, *Reinforcement Learning Stock Trading Bot* – *Python, Pandas, Numpy, Gymnasium, Stable-Baselines3, Xgboost*
- Designed and implemented a reinforcement learning framework that trained actor-critic algorithms to optimize stock trading strategies for maximizing portfolio returns in dynamic financial markets.
- Developed a custom stock trading environment with a continuous action space, incorporating technical indicators, volatility measures, and risk-adjusted reward functions to guide agent decision-making.
- Explored five actor-critic algorithms, including PPO, SAC, and TD3, and created ensemble strategies (weighted average and meta-agent) to aggregate agent actions for enhanced performance and risk mitigation.

- Achieved significant returns above the Dow Jones Industrial Average baseline, while reducing annual volatility and maximum drawdown through ensemble decision-making.

UMich, *PyDuck –* *Python, DuckDB, Pandas, SQL, NumPy, Polars*

- Co-developed PyDuck, a dataframe library that compiles Pandas-style operations to DuckDB SQL, allowing data scientists to interact with an intuitive DataFrame abstraction while benefiting from vectorized execution, multithreading, and out-of-core performance.
- Designed the Quack abstraction—an immutable, chainable object representing virtual views over DuckDB tables—that supports lazy evaluation and SQL query lineage tracking inspired by Spark's RDD model.
- Built a modular SQL compiler to translate chained operations like filter(), groupby(), dropna(), and get_dummies() into optimized DuckDB SQL, with full transparency via .to_sql() for inspection and debugging.
- Implemented core preprocessing operations in a composable architecture, enabling extensibility while decoupling transformation logic from execution.
- Conducted comprehensive performance benchmarking on TPC-H workloads and real-world preprocessing tasks across Pandas, DuckDB, and PyDuck at scales up to 10GB, showing >100x speedups over Pandas on common aggregation and filtering tasks.
- Analyzed limitations of SQL-based engines for certain row-wise operations (fillna, get_dummies) and proposed future improvements including hybrid eager execution and optimized SQL translation for non-relational patterns.
- Demonstrated that PyDuck can serve as a drop-in replacement for Pandas for scalable ML data pipelines, bridging the gap between familiar APIs and high-performance analytics engines.

UMich, *Sparse Action Spotting with Video-Language Models* - *Python, PyTorch, Hugging Face Transformers & TRL*

- Developed a token-based vision-language model (VLM) pipeline for semantic highlight detection in broadcast soccer videos, addressing the sparse action spotting task using video and natural language commentary as input.
- Fine-tuned Qwen2-VL-2B, a lightweight open-source VLM, on 28,939 1-minute gameplay clips from the SoccerNet v2 dataset using associated commentary transcripts and action labels (e.g., goals, free-kicks, fouls).
- Constructed a supervised training dataset and implemented a finetuning pipeline using QLoRA for memory-efficient training on constrained GPU environments. Employed token-level cross-entropy loss via Hugging Face's TRL SFTTrainer.
- Designed and evaluated a prompt-driven inference system that takes a video + text query (e.g., "show all goals") and outputs timestamp intervals for trimming highlights, using IoU-based precision, recall, and F1 as evaluation metrics.
- Compared the fine-tuned VLM against Mistral-7B and OpenAI o4-mini, showing that despite having fewer parameters, the VLM achieved higher recall (0.40) and F1 (0.11)—demonstrating improved video grounding over larger LLMs.
- Contributed to dataset construction, fine-tuning design, model evaluation, and comparative analysis across all three model architectures. Results suggest that token-based, multimodal models are a promising alternative to CNN-based or feature-extractor approaches in long-form video understanding.

UMich, *Realized Volatility ML Pipeline -* *Python, PyTorch, CUDA/C++, ONNX, Polars, FastAPI, Numpy, Typer*

- Developed an end-to-end machine learning pipeline for predicting realized volatility in financial time-series data, with a focus on low-latency inference suitable for high-frequency trading scenarios. The pipeline spans data ingestion, model training, model export, deployment, and performance benchmarking.
- GPU-accelerated training: Built a PyTorch training workflow with a custom CUDA kernel for the RMSPE loss function (Root Mean Squared Percentage Error), achieving ~2× faster loss computation than a pure PyTorch implementation.
- Implemented a lightweight Transformer model (~400K parameters) for log-volatility prediction and leveraged mixed precision and PyTorch 2.0 compilation for speedups.
- ONNX model serving: Exported trained models to ONNX format for efficient deployment. Implemented a FastAPI inference server using ONNX Runtime with GPU support, featuring dual inference endpoints (standard JSON and a binary NumPy interface) to balance integration ease and maximum throughput. The optimized binary endpoint achieved ~11× higher throughput (~948 QPS vs 85 QPS) and ~1 ms per-sample latency in batch inference tests.
- Full CLI pipeline: Developed a flexible CLI tool (Python Typer) to orchestrate all pipeline stages – from data

indexing & feature generation to model training, evaluation, and serving. This command-line interface (with YAML configuration files) enables reproducible experiments and streamlined end-to-end execution (e.g. one command each for index, build, train, export, serve, evaluate, bench).

- Efficient data handling: Optimized data ingestion through sharding and parallel feature extraction (using Polars DataFrames and Python multiprocessing), enabling streaming of large-scale time-series data into the training pipeline. Ensured reproducible train/validation/test splits and on-the-fly data normalization for robust model training and evaluation.

SKILLS

Programming Languages: Python, C++, Java, Go, C#, SQL, JavaScript, TypeScript, HTML/CSS, R, Bash, LaTeX
Cloud Platforms & Infrastructure: AWS (Bedrock, EC2, ECS, EKS, SageMaker, Lambda, CloudFormation, CloudWatch, S3, DynamoDB, VPC), GCP, Azure, Docker, Kubernetes, Slurm
Databases & Storage: PostgreSQL, MySQL, DynamoDB, Chroma DB, DuckDB, MongoDB
ML/AI Frameworks & Libraries: PyTorch, TensorFlow, Hugging Face Transformers, Hugging Face TRL, Stable-Baselines3, Scikit-learn, LangChain, CompilerGym, QLoRA, pgmpy, NeuroKit2, Pandas, NumPy, SciPy, NLTK
Developer Tools & Testing: Git, AWS CDK, AWS SDK, TestNG, PyTest, Zod, FFmpeg, unittest
Web Frameworks: Django, Flask, React,