Andrew M. Zhang

(510) 676-4193 | andrewmzhang@berkeley.edu | andrewmzhang.com | github.com/andrewmzhang

EDUCATION

University of California, Berkeley *Aug 2016 – Expected May 2020*

B. A. in Computer Science, GPA: 3.957

Relevant Coursework: Machine Learning, Computer Security, Algorithms, Concepts of Probability Theory, Concepts of Statistics, Upper Div. Linear Algebra, Machine Structures, Discrete Math and Probability Theory, Data Structures, Designing Information Devices and Systems I/II, Structure and Interpretation of Computer Programs

In Progress: Stochastic Processes, Real Analysis, Computational Photography, Deep Reinforcement Learning

Programming Skills: C/C++ (Nvidia CUDA) ∘ Java (Android) ∘ Python (Flask, Numpy, Sklearn, Tensorflow, Pytorch, Pywren) ∘ Javascript (Meteor, Nodejs) ∘ HTML/CSS ∘ MongoDB ∘SQL ∘Firebase framework

EXPERIENCE

RISELab, UC Berkeley – Disaggregated Machine Learning (C++): Cirrus Aug *2017 – present*

Undergraduate Researcher. Project Link: github.com/jcarreira/cirrus

- Low-cost, serverless, machine learning, and hyperparameter optimization framework in C++ that runs on AWS Lambdas
- Improved scalability without performance loss using multiple parameter servers (model sharding) for Logistic Regression
- Implemented collaborative filtering model using SGD, helped fix mathematical errors with original implementation
- Made Cirrus converge 70% faster than Spark on datasets, including Criteo Ad-Click Logs and Netflix Recommendations
- Created a Plotly Dash UI to visualize hyperparameter search and kill diverging experiments in Jupyter Notebook
- Implemented logistic regression with map reduce using Pywren (AWS S3, Lambdas) for a comparison baseline

uGSI, UC Berkeley - CS70: Discrete Math and Probability Jan *2017 - present*

Undergraduate Teaching Assistant. ~750 students

- Prepared mini-lectures, discussion questions, and test questions
- Worked with ~20 other uGSI's to answer students' questions, handle course logistics, and hold 1 on 1 sessions with students
- Ran officer hours to help students on homework and theory questions.

Geeni June *2016 - present*

Lead Android Developer

- Lead a team of 5 Android developers to create a Android app: https://youtu.be/9lTFQg4BV9g
- Designed and documented how the backend and frontend would work for Android, iOS, and NodeJS.
- Wrote a newsfeed for the app using Google Firebase Realtime DB, made a wallet that charged users using Stripe, and implemented user authentication, logout, and account deletion with Firebase Auth.
- Finalists in Berkeley's Big Ideas Startup Competition.

PROJECTS

Blinn: A C++ Raytracer for Metaballs - github.com/andrewmzhang/blinn - https://youtu.be/dMIecrYXcoE

- Created a raymarching metaball render from native C++
- Used Nvidia CUDA to aid in embarrassingly parallel renders, 1000x speedup over multithreaded CPU solution
- Renders video of several oscillating metaballs interconnected by springs with diffuse shading

Various Projects

- **Neural Net and Backpropagation:** Python - Used numpy to implement a Neural Net with fully connected and convolutional layers. Trained it successfully to recognize fruits and veggies.
- **Pseudo-Dropbox:** Python - Encrypted dropbox-like client that uses a public key-value store as a backend. Cryptographically secure, corruption-proof, file sharing (and unsharing). Fast updates on file changes with file sharding and Merkel Trees.
- **SQL:** Java - SQL implementation capable of basic SQL commands (select, update, delete, joins, insert into, etc.)
- **BearMaps**: Java - Used a QuadTree to effectively make a zoomable map of UC Berkeley
- **Text Editor** : Java - Wrote a custom data structure to implement a text editor using on JFrames. Supports copy-pasting, click navigation, saving, loading, and undo/redo.
- **Hog Dice Game Solver** : Implemented an optimal solve using an Expectimax Tree for a dice game, implemented in Python. 85% win rate vs naive strategies.
- **Collision Simulation**: C++ - 2D particle elastic collision simulation in real time using priority queues. Each frame updates in linearithmic time. https://vimeo.com/150040521
- **Boid Flocking Simulation**: Java - Created a flocking animation using an efficient k-nearest neighbor search with a k-d tree. Each frame updates in linearithmic time. Simulation of 1000 boids: https://vimeo.com/198900343
- **Yelp Rating Prediction:** Python - Employed MapReduce programming paradigm to parallelize a simple Naïve Bayes classifier with a Bag of Words model in Spark to predict Yelp review ratings
- **Scientific Computing Optimization:** C-Optimized a naïve version of NumPy using performance programming techniques (e.g. SIMD and OpenMP). Achieved >70x speedup compared to the naïve solution