

# Akshay Trikha

akshaytrikha@berkeley.edu | 510-301-0042 | akshaytrikha.github.io | US Permanent Resident

## EDUCATION

<b>University of California, Berkeley</b> <i>Master of Engineering in Materials Science &amp; Engineering</i>	2023 – 2025 Berkeley, CA
<b>Harvey Mudd College</b> <i>Bachelor of Science in Computer Science</i>	2017 – 2021 Claremont, CA
Selected coursework: Machine Learning, Natural Language Processing, Materials Science of Energy Conversion & Storage, Quantum Physics, Microprocessors.	

## SKILLS

Technical: Python (PyTorch, TensorFlow, NumPy, SciPy, Scikit-learn, Pandas, OpenCV), C++, C, JavaScript (TensorFlow.js), React, Vue, SQL, HTML/CSS, Java, Google Cloud Platform  
Natural Language: Hindi (fluent), Mandarin (conversational), Sanskrit (learning), English (fluent).

## EXPERIENCE

<b>QuantumScape</b> <i>Machine Learning Engineer</i>	09/21 – Present San Francisco, CA
<ul style="list-style-type: none"><li>Design &amp; manage ML-based image processing pipelines to detect defects, make manufacturing scrapping decisions, and support materials research.</li><li>Use Landing AI for segmentation, object detection, and classification model development. My 10 models in production run inference ~25,000 times / day.</li><li>Develop features for a Vue.js dashboard able to efficiently handle ~100GBs / day worth of image data.</li><li>Created a REST API using Flask used in our dashboard as a part of a data engine that feeds into models.</li></ul>	
<b>Sandia National Laboratories</b> <i>Researcher, 9-person team</i>	09/20 – 05/21 San Francisco, CA
<ul style="list-style-type: none"><li>Investigated link between diameter of ferroelectric barium titanate nanoparticles and dielectric constant.</li><li>Created a Jupyter Notebook / Python image processing pipeline using OpenCV, NumPy, and Matplotlib to extract particle sizes and distribution from transmission electron microscope images. Then optimized runtime 25x by using Numba library.</li><li>Presented at Materials Research Society '21 Spring Meeting &amp; published in MRS Advances, link at <a href="https://tinyurl.com/sandia-paper">tinyurl.com/sandia-paper</a>.</li></ul>	
<b>AMISTAD Lab</b> <i>Researcher, 6-person team</i>	05/19 – 12/19 Claremont, CA
<ul style="list-style-type: none"><li>Explored why machine learning works from an information theory and search perspective.</li><li>Co-authored <i>The Bias-Expressivity Tradeoff</i>, won best paper award for ICAART2020 in Valletta, Malta.</li><li>Co-authored <i>The Futility of Bias Free Learning</i>, which team presented at AI2019 in Adelaide, Australia.</li><li>Created <a href="https://tinyurl.com/amistad-futility">tinyurl.com/amistad-futility</a> to communicate research findings in more accessible manner.</li></ul>	
<b>Coinhako</b> <i>Software Engineer Intern</i>	07/18 – 08/18 Singapore
<ul style="list-style-type: none"><li>Helped develop SmartWallet, a crypto to crypto exchange platform that is in production.</li><li>Wrote smart contracts in Solidity for handling ERC20 token transactions. Two are now in production with &gt;100k users.</li></ul>	

## PROJECTS

<b>Neural Style Transfer</b>   <i>JavaScript, React, HTML/CSS</i>	07/21 San Francisco, CA
<ul style="list-style-type: none"><li>Created a neural style transfer web app that generates stylized images of webcam input in near real time.</li><li>Used a pretrained TensorFlow.js model, link at <a href="https://styletransfer.art">styletransfer.art</a>.</li></ul>	
<b>Flow Battery Simulation</b>   <i>Jupyter Notebook</i>	05/20 Singapore
<ul style="list-style-type: none"><li>Characterized single cell vanadium redox flow battery discharging by numerically integrating a system of governing differential equations in a Jupyter notebook.</li><li>Python packages: SciPy, NumPy, Matplotlib. Link at <a href="https://tinyurl.com/flow-battery-sim">tinyurl.com/flow-battery-sim</a>.</li></ul>	
<b>AES Encryption</b>   <i>C, SystemVerilog</i>	06/2020 Claremont, CA
<ul style="list-style-type: none"><li>Built a hardware implementation of AES FIPS 197 encryption specification using an FPGA that ran in 300 nanoseconds (excluding SPI transfer from a microcontroller)</li><li>Software implementation using C ran on average 13715.3 ns, or 45x slower. Link at <a href="https://tinyurl.com/akshay-aes">tinyurl.com/akshay-aes</a>.</li></ul>	