Concurrent Systems (ComS 527)

Ali Jannesari

Department of Computer Science

Iowa State University, Spring 2023

PROJECT IDEAS

Project Ideas – Remarks!

- Talk to me if you want to propose your own project idea
- Read these project ideas as **"get this application running, then speed it up"**
- Some of the ideas come from deep learning (ML) applications.
- But you are free to choose **any other domain** to develop an application either from scratch (e.g. a parallel version of a zip program) or speeding up an existing open-source application (e.g. speeding up an existing desktop search program or an encryption application, etc.)

Video blurry barcode detection @ 100FPS

Find barcodes in video stream → Output overlay of bounding rotated rectangles.
Blurry? No problem! (minimally: do better than SoA)
Overlay:

- GREEN=good (show code)
- YELLOW=maybe (show failure diagnostic -- too blurry, truncated, too small, etc..)
- RED=bad (show failure reason -- bad/unknown code, etc.)

Bonus: BarCodeSphere (Photosphere-ish thing for barcodes):

- follow/track detections/overlays persistently across frames
- use model: 'sweep' environment until all readable codes are decoded.

- Determine state of the art (too hard? already solved?)
- make problem harder/easier as needed
- Determine target platform (CPU/GPU/mobile)
- Determine if/where ||ism is helpful
- If you are (or aspire to be) cool: use Deep Learning and/or Big Data

Mini-Photoshop

- **Idea 1:** How fast can you Sharpen and Blur photos?
- **Idea 2:** Pick your favorite photoshop functionality, we can help you frame a project around implementing it efficiently

Speeding up audio event detection

Task: Classify and detect certain sound events in an audio file

Goal:

1) Learn to use machine learning tools (i.e. theano, R) to classify sound.

2) Understand performance implications; how to make it faster.

3) Understand memory/power budget; deployment in mobile devices.

What kind of party is Obama attending today?

Speeding up Surveillance Camera Anomaly Detection

Task: Detect Anomaly in Surveillance Video.

Goal:

1) Learn to use machine learning tools (i.e. theano, R) to detect anomaly in surveillance videos.

2) Understand performance implications; how to make it faster.

3) Understand memory/power budget; deployment in mobile devices.

Sorry burglar, you are detected today. Try hard next time.

Speeding up LSTM/Transformers for sequence Learning

Task: LSTMs/Transformers are the leading machine learning tool for analyzing time series data.

Goal:

1) Learn to use machine learning tools (i.e. theano, R, Pytorch) to run LSTM/Transformer.

2) Understand performance implications; how to make it faster.

3) Understand memory/power budget; deployment in mobile devices.

Speeding up Finding Celebrity in a Video

Task: Given a youtube video URL, find out if a certain celebrity is in the video or not.

Goal:

1) Learn to use machine learning tools (i.e. theano, R, PyTorch) to identify a speaker.

2) Learn to mine data from social media.

3) Understand performance implications;

how to make it faster.

Thats Alan Turing!!!

RNN/ViT for Automated, Real-Time Esports Spectator Camera Control

‘Good’ Spectator Cam Control is (currently) a Full Time Job:

- Automated Cam Control has issues:
  - “The spectator mode needs some adjustments. [...]. While trying to watch the VODs for RNG, I found myself nauseated by the constant bouncing of the camera. This problem is exacerbated by Kalista's jumpiness, but has problems beyond that.”
    -- BlueThingamajig
- Machine Learning / Deep Learning / RNNs /ViT are “unreasonably effective” at sequence translation/learning/prediction: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- Project:
  - View gameplay as sequence of events (use spectator/stats APIs) → RNN /ViT Input
  - View professional observer camera work as sequence of positions → RNN/ViT Output
  - Train RNN/ViT to emulate professional observer
  - Speed up RNN/ViT Training: port CPU RNN training to GPU / improve existing GPU alg.

Emotion analysis on reviews (Speeding up NLP)

Task: Perform NLP on online wine reviews to find out the best wineries in the Napa valley.

Goal:

1) Learn to use machine learning tools (i.e. theano, R) to do NLP.

2) Learn to mine text data from the social media.

3) Understand performance implications; how to make it faster. How to analyze larger data sets.

Ohh the wine was awesome!!!

Bitcoin Mining

Task: Implement a parallel Bitcoin (or any other <X>-coin) miner.

Goals:

1. Learn how Bitcoin (or <X>-coin) works. Understand proof-of-work, cryptographic hash functions (e.g., SHA-256), and digital signatures (elliptic curve DSA).
2. Optimize the heck out of a GPU miner.
3. Optional: Mine Bit/<X>-coins and make $$$ (net electricity costs).

Parallel Garbage Collection

Task: Implement parallel garbage collection on the GPU for a virtual machine environment of your choice.

Goals:

1. Understand how garbage collection work and how to adapt it on the GPU. For background, see a paper by Maas, Reames et. al from 2012 [1]. They implemented a mark-and-sweep algorithm.
2. Starting off, it will be best to implement GC on a toy memory layout of your design. Time permitting, you can choose a small VM, e.g., Lua, Ruby, Python, in which to embed your parallel GC.

[1] http://www.philipreames.com/publications/ismm2012-gpugc.html

Parallel/Distributed Database

Task: Implement a parallel or distributed database management system, or an important component of a database.

Goals:

1. Understand how modern databases are engineered.
2. Carefully implement a database or an important piece of it (queries, memory management, transactions, locking, etc.).

Note: It is recommended that you have taken and done well a database course (or an equivalent) if you want to work on this.

Global Address Space Runtime

Task: Implement a global address space (GAS) runtime for running parallel programs.

Goals:

1. Understand how GAS systems work.
2. Possibly implement a GAS runtime for running "jobs" on CPUs and GPUs (significant freedom on what counts as a "job").
3. Possibly implement a distributed GAS runtime.

Parallel Linear Algebra

Task: Implement some common BLAS or LAPACK routines in parallel for CPU or GPU.

Goals:

1. Learn more than you ever wanted to about linear algebra!
2. Understand and implement the core concerns of parallel programming.

Page image segmentation

Isolate and extract:

pictures

line art

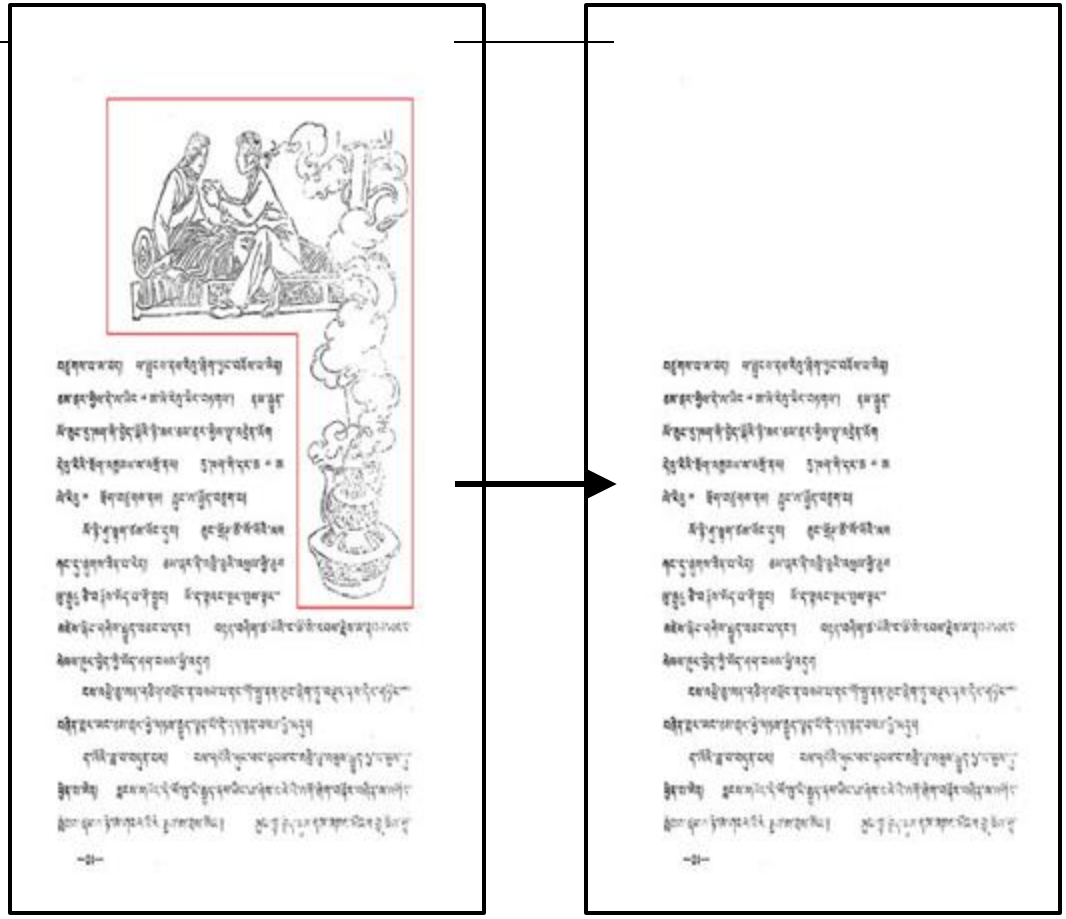
tables

from scanned book pages and historical documents

Inputs: High resolution BW/grayscale/color images

Output: bounding box coordinates and/or transformed image with pictures removed.

Learn and make fast common image processing algorithms. Help improve automated text recognition (OCR)



For more info, check out Bukhari, S. (2011). "Improved document image segmentation algorithm using multiresolution morphology". *Imaging*.

TensorKart (RL – Reinforcement Learning)

- This project uses TensorFlow to train an agent to play the popular game MarioKart 64. In this project, the model has been trained using MNIST dataset for character recognition, and Nvidia's Autopilot was developed for creating self-driving vehicles.
- Add a reinforcement layer based on lap time or other metrics so that the AI can start to teach itself now that it has a baseline. The environment currently provides a reward signal of -1 per time-step, which gives the AI agent a metric to calculate its performance during each race (episode), the goal being to maximize reward and therefore, minimize overall race duration.

Match passages of text in a large corpus

Task: identify matching (or nearly matching) passages of text across a large corpus

Useful for: detecting plagiarism, correcting computer recognized (OCR'd) text, studying intertextuality among authors

Task: Implement and and look for parallelism in existing recursive algorithms

Scalable parameter sharing for Tensorflow (or PyTorch)

Sharing parameters is how neural networks can become more robust and how we can enable model parallelism

Task: code a framework to make enable model parallelism with Tensorflow on Amazon EC2 and a high performance super computer (100 Knights Landing nodes). Ultimately implement asynchronous learning methods and tweak them.

Useful for: learning to code distributed and parallel systems, learning about deep learning with an emphasis on time series analysis

Classifying Unit Test Functions

Use deep learning for program analysis

Task:
Derive essential metadata such as the test pattern used to write a unit test and the function under test (FUT).
(e.g. arrange–act–assert pattern in the example)

Goals:

- Learn code analysis with machine learning
- Use deep learning tools (Tensorflow, Azure,...)
- Get knowledge of unit testing
- Create a set of unit tests from common open source projects as training data
- Extract features from unit tests as the input set
- Detect the test pattern and FUT for a given unit test using your model

```
[TestFunction Example]
public void GetCount_Empty_NoMessages()
{
            // arrange
            Mailbox mailbox = new Mailbox();
            Mailbox.Init();

            // act: call the function under
test (FUT)
            var result = mailbox.GetCount();

            // assert
            Assert.AreEqual(0, result);
}
```

Distributed Machine Learning using OpenMP tasking

- Implement distributed machine learning models using OpenMP on multi-GPU nodes
- Master-Worker distributed training model using OpenMP tasking
- Compare the performance against existing distributed training methods (e.g. pytorch, MPI implementation)

Distributed Machine Learning using MPI

- MPI programming model is used for implementing distributed machine learning approaches across the nodes of High Performance Computing (HPC) clusters
- Implement your own version of distributed machine learning models using MPI on a HPC cluster with multi-GPU nodes
- Compare the performance against existing distributed training methods (e.g. pytorch, OpenMP/MPI implementation)

OpenMP on Python

- Python is a popular language for ML/DL based research. Using OpenMP on Python would be interesting to evaluate the same code from C and checking for performance variations between programming constructs.

Parallelizing Bzip/Gzip

- Use pipeline parallelism for parallelizing Bzip/Gzip in OpenMP or any other parallel programming model

Parallelizing VLC Media Player

- Use data and task parallelism for parallelizing VLC Media Player in OpenMP or any other parallel programming model

Decentralize Learning and Federated Learning (FL)

- Develop a scalable method for privacy-preserved training in FL (IID and Non-IID settings) and decentralize learning
- Reduce communication costs between nodes/clients while training

Scalable Neural Architecture Search (NAS)

- Develop a scalable method for Neural architecture search (NAS) using distributed training for supernetwork/subnetwork training