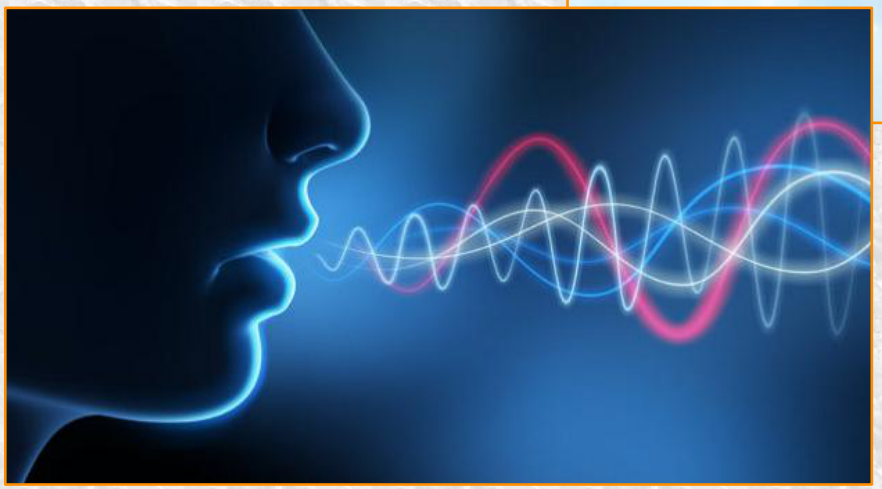
APNNS/ IEEE
Deep Learning and Artificial Intelligence
Summer School (DLAI3)
29 June – 3 July 2020

Doing Deep Learning:
The hype and the reality

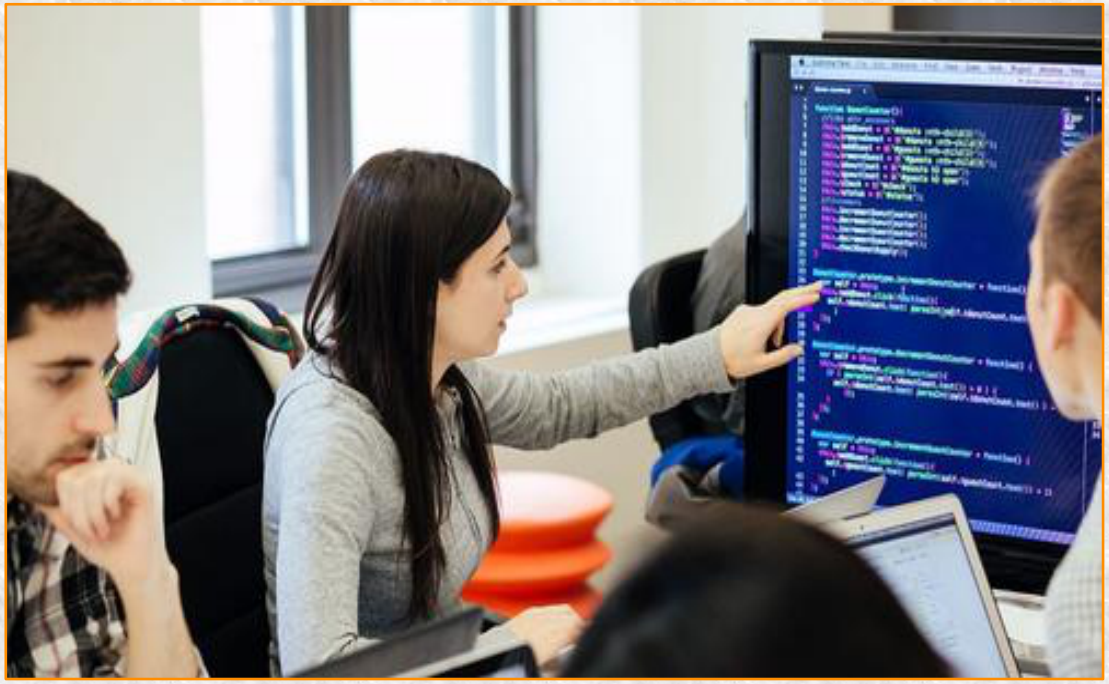
Sally E. Goldin
Department of Computer Engineering
King Mongkut's University of Technology Thonburi

The brave new world of deep learning



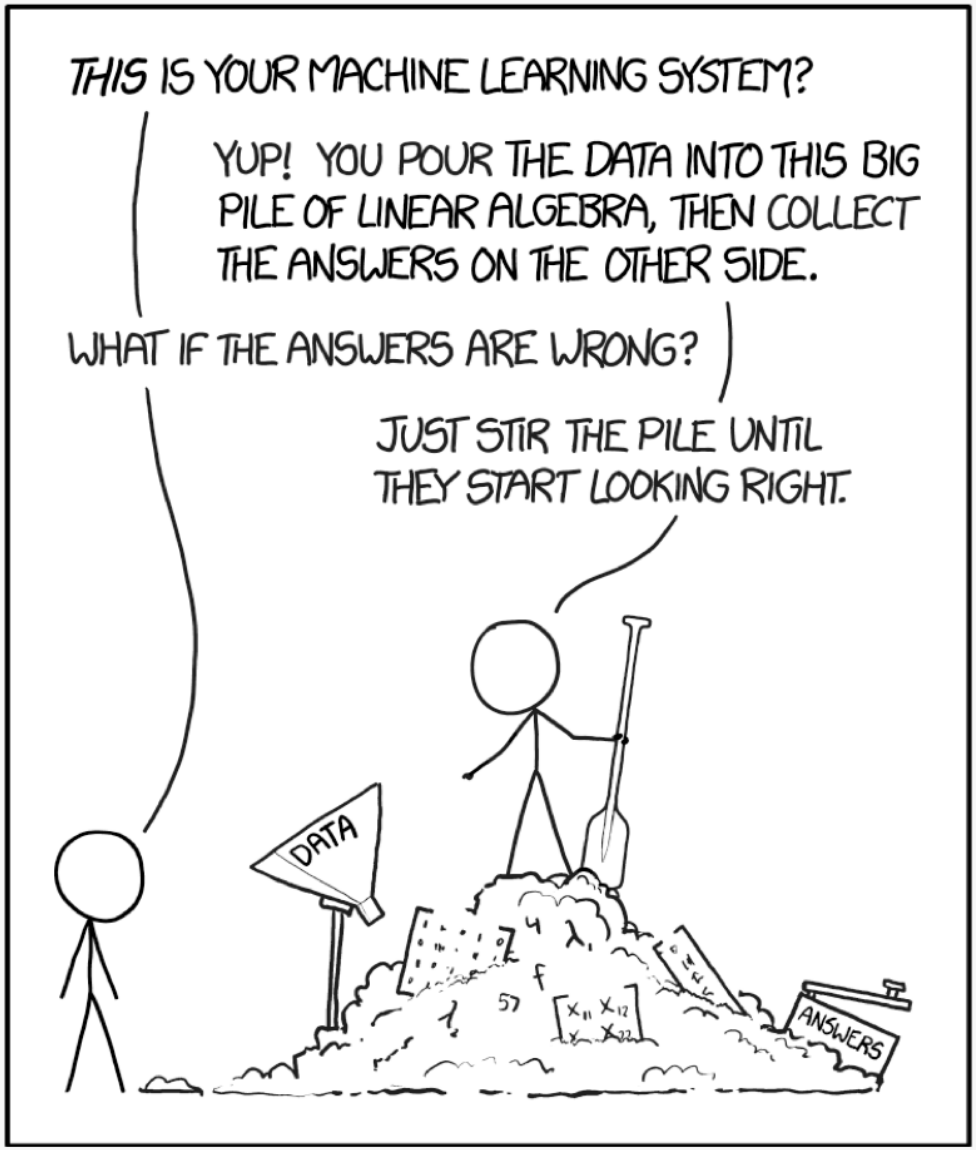
Accessible technologies and tools

- ***Hardware / infrastructure***
  - Cloud computing
  - GPGPU
- ***Software frameworks***
  - Tensorflow
  - PyTorch
  - Keras
  - Caffe
- ***Data sources***
  - Mobile devices
  - IOT
- ***Pre-trained models***



Anyone can build a DL model

Right?



About my background

PhD from Carnegie-Mellon University

During the first "golden age" of AI: Herb Simon, Allen Newell, Marvin Minsky, Patrick Winston, John McCarthy...

Research Scientist at the Rand Corporation

Software engineer, architect, project leader

Lecturer & researcher at KMUTT

Two dozen programming languages
Many publications
1500+ students...

Case study: My first real DL project

- ***Successes***
- ***Challenges***
- ***Learning***

In collaboration with Professor Yuanjie Xiao
High Speed Railway Engineering Laboratory
Central South University, Changsha, China

The problem

Given a photo of a railway embankment under construction, estimate the level of compaction

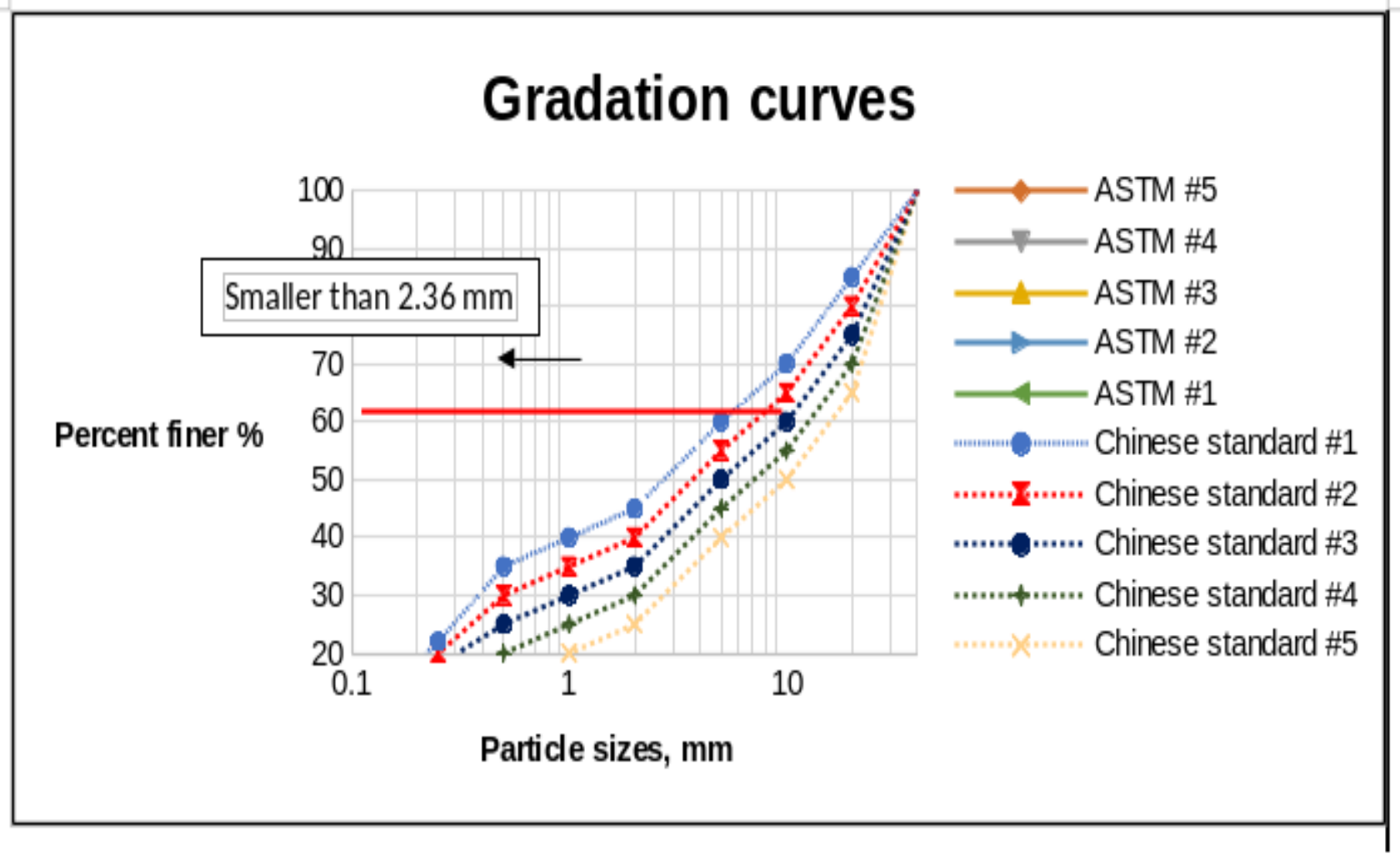
Approach: Train a DL model to discriminate between different known gradations which can be mapped to estimates of compaction

Gradation 1

Gradation 3

Gradation 5

Different mixes of particle size



Challenge 1: Development environment

- **Google Cloud: costs impossible to predict**
- **Configured dedicated virtual machine**
  - Kubuntu 18
  - Python 3.7
  - PyTorch
  - No GPGPU
- **PyTorch “getting started” tutorial crashed!**
- **Had to move VM from AMD to Intel-based host**
- **Continued to see other bugs (e.g. processes spontaneously killed)**

Exploratory data set

- Only two categories, not standard gradations
- 200 images of each, 4896 x 3264 pixels – extracted multiple 500 x 500 subsets
- ~6300 training images per class, 1600 validation images

Class 1

Class 2

Best DL model

- **Model architecture**
  - 3 convolutional layers (6,16,32) with ReLU and max pooling
  - 3 fully connected layers (128,64,32) – 2 outputs
  - Cross-entropy loss error function, stochastic gradient descent back propagation, learning rate 0.001, momentum 0.9
- **Data loading / transformation**
  - Image load direct from disk
  - Gray scale conversion, sub-sample to 32 x 32 for input
  - Random 25% brightness and contrast “jitter”

Note: very similar to the generic model used in the PyTorch tutorials.

Success 1: Acceptable accuracy

- **92% accuracy on validation images after 150 epochs**
- **Minimum loss still declining**
- **Probably could push accuracy higher**
- **Quite slow to train**
  - *About 5 minutes/epoch*
  - *Total time 13 hours*
- **These were not our real target classes**

Standardized data set

- ***Five categories***
- ***Standard gradations***
- ***Pre-processed like exploratory data set***
  - 200 images of each, 4896 x 3264
  - Extracted multiple 500 x 500 subsets
  - ~5600 training images per class, 1400 validation images
- ***Total data volume approximately 10 GB (8GB training)***

Challenge 2: Learning speed

- Modified previous model (5 classes, 0.01 learning rate)
- 120 epochs -> loss 1.45, **accuracy 27%**
- Each epoch takes > 10 minutes, > 20 hours
- ***Experiments***
  - Bigger subsets (1000 x 1000) – more representative texture
  - Two gradations only (1 and 5) – easier discrimination – 240 epochs, loss 0.062, **accuracy 62%**
  - This took days! And still just two classes

New development environment

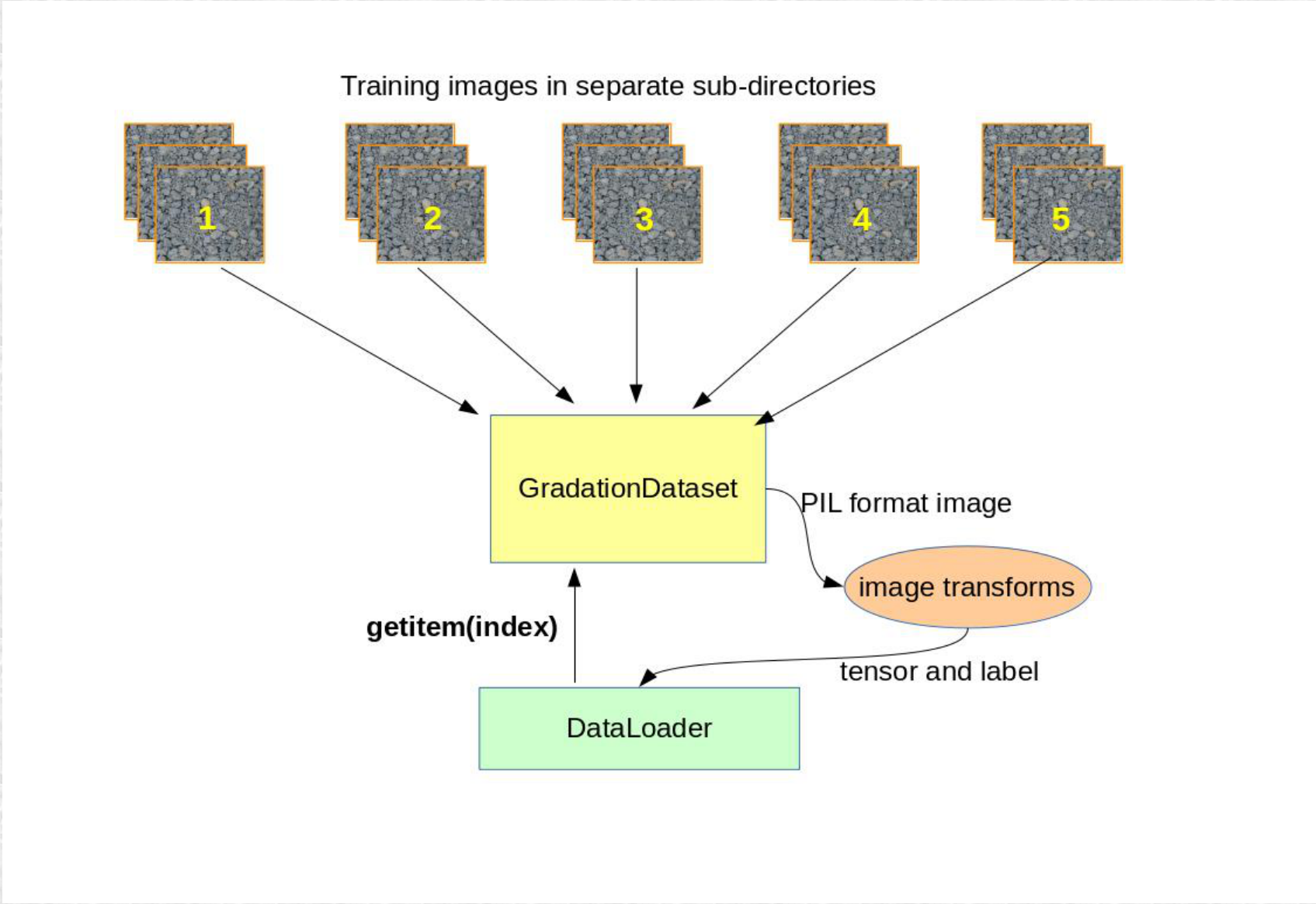
- Physical machine: AMD Ryzen 1850 MHz processor, 8 cores, 16 GB memory
- CentOS 8, Python 3.7.4
- PyTorch with CUDA 10.2 support
- Gigabyte RTX 2060 GPGPU
- 2 TB disk space

Added code to move model and data to the GPGPU device (cuda) during training

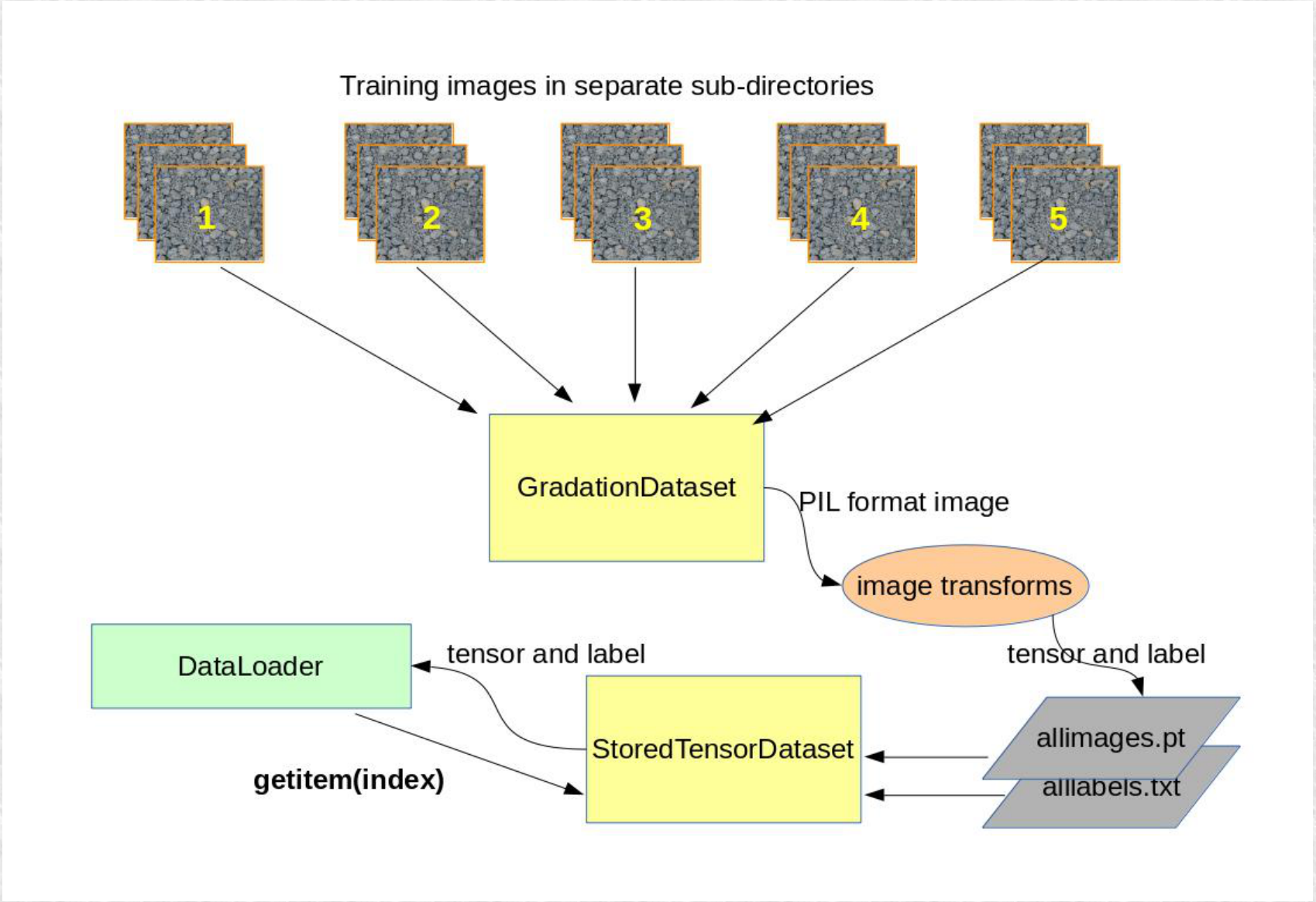
Challenge 3: GPGPU does not help

- Each epoch **still requires more than 9 minutes**
- 100 epochs, 15 hours total, loss down to only 1.203
- **Why?**
  - Sending all training data (8 GB) to the cuda device, each epoch
  - Data transfer time swamping computation time
- **Solution**
  - Pre-process (gray transform, subsample, color jitter) all training images and save as tensors in PyTorch serialization format
  - Create a new data loader class that loads all training data as tensors at the start of training
  - Each sample still sent to cuda device separately, but this is a small amount of data

Original data loading strategy



New data loading strategy



Success 2: Greatly improved speed

Each epoch now takes about 10 seconds

Challenge 4: Accuracy is still poor

1500 epochs – loss 0.657 – accuracy 30%

Analysis

- Is this a model problem?
- Or a data problem?
- Given the success this model had with the exploratory data, look at the data first
- ***New image subsets***
  - 2400 x 2400 (instead of 1000 x 1000) to better capture texture
  - Increased model input size from 32x32 to 64x64
  - Note training data images now require > 50 GB
  - However, we only need to pre-process this once, to create the serialized tensor data, which is about 120 MB

Success 3: Improved accuracy

- After 1100 epochs, loss 0.000, ***accuracy 61%***
- Model is clearly over-fit (classifies the training data perfectly)
- Consulted PyTorch forum for suggestions
  - "Make sure you use Softmax as final layer"
  - "Try using dropouts"
  - "Add more training data"
  - "Optimize hyper-parameters"

More challenges...

- Added Softmax layer and tried dropouts
- Performance got much worse (no learning after 6000 epochs)
- Did not improve after removing the dropouts
- After some research I discovered that the Cross Entropy Loss class in PyTorch incorporates Softmax already

When you're a new to a field, it's easy to be led astray!

Latest success

- ***Doubled the number of training images***
  - 2600 total per class
  - More than 100 GB of raw data
- ***Removed Softmax layer***
- ***200 iterations, 0.000 loss, 90% accuracy***

More work to be done

- Increase accuracy (hyper-parameters...)
- Test against field images
- Integrate integrate into real time environment

Lessons learned

- **PyTorch** – powerful, flexible, multiple levels of abstraction. well-supported – but not bug free
- **Data** may be more important than model structure
- Deep learning makes heavy demands on **hardware,** both computational power and disk space
- Deep learning requires time to configure the **software** environment
- **Detailed notes** on experiments are essential
- **Incorrect advice** can lead you astray

Deep learning is powerful...

but it’s not magic!

Questions?

Contact information

Email: *seg@goldin-rudahl.com*

LinkedIn: *https://www.linkedin.com/in/sally-e-goldin-b77595b/*

Website: *http://windu.cpe.kmutt.ac.th*

YouTube: *https://tinyurl.com/youtubeseg*