https://chicago.ml
The AI World converges on Chicago!

AI Ideas, Careers and Impact March 12th | Midwest Applied AI Conference May 20-21st

The Chicago Artificial Intelligence community invites you to:

- **Discover** Learn about AI applications
- **Learn** Dig into data & algorithms
- **Network** Meet other AI professionals
- **Start an AI Business** Attend AI startup boot camp
Great!

@guildai
Introduction

What is machine learning?

Theory

Tools
What is machine learning?
What is machine learning?
What is machine learning engineering?

**Infrastructure**
- Facilities and tools for research and engineering
- Continuous integration and continuous development

**Research**
- Data analysis
- Data processing and preparation
- Model selection
- Training a model

**Production**
- Model inference
- Model optimization
- Deployment
Introduction

Why machine learning engineering?

Use Cases
- Anomaly detection (e.g. fraud)
- Optimization (e.g. minimize cost, maximize yield)
- Market analysis
- Risk analysis
- Prediction
## Machine learning vs traditional data analytics

<table>
<thead>
<tr>
<th></th>
<th>Traditional Data Analytics / BI</th>
<th>Machine Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data suited for</td>
<td>Structured</td>
<td>Structured and unstructured</td>
</tr>
<tr>
<td>Typical application</td>
<td>Summary/reports, some prediction</td>
<td>Prediction, some summary/reports</td>
</tr>
<tr>
<td>Artifacts</td>
<td>Reports, graphs</td>
<td>Trained models, applications</td>
</tr>
<tr>
<td>Used by</td>
<td>Human decision makers</td>
<td>Application developers</td>
</tr>
</tbody>
</table>
What are the roles in an ML engineering team?

**Research Scientist**
- Pure and applied research
- Some programming
- Budget for publishing

**Research Engineer**
- Support research scientist
- More programming
- Implement papers
- Requires in-depth knowledge of science

**Software/Systems Engineer**
- Support ML systems
- Custom development
- Systems integration
Tools of the trade

First instruments for galvanocautery introduced by Albrecht Middendorpf in 1854 (source)
## Tools of the trade

### Programming languages

<table>
<thead>
<tr>
<th>Language</th>
<th>When to Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>General ML, data processing, systems integration</td>
</tr>
<tr>
<td>R</td>
<td>Stats, general data science</td>
</tr>
<tr>
<td>C/C++</td>
<td>System software, HPC</td>
</tr>
<tr>
<td>JavaScript</td>
<td>Web based applications</td>
</tr>
<tr>
<td>Java/Scala</td>
<td>Enterprise integration</td>
</tr>
<tr>
<td>bash</td>
<td>Systems integration</td>
</tr>
</tbody>
</table>
## Computational libraries and frameworks

<table>
<thead>
<tr>
<th>Library</th>
<th>Sweet Spot</th>
<th>When to Look Elsewhere</th>
</tr>
</thead>
<tbody>
<tr>
<td>TensorFlow</td>
<td>Deep learning, production systems including mobile</td>
<td>New to ML, no production requirements</td>
</tr>
<tr>
<td>PyTorch</td>
<td>Ease of use, popular among researchers</td>
<td>Production requirements beyond simple serving</td>
</tr>
<tr>
<td>Keras</td>
<td>Ease of use, production backend with TensorFlow</td>
<td>Affinity with another library (e.g. colleagues use something else),</td>
</tr>
<tr>
<td>MXNet</td>
<td>Performance, scalability, stability</td>
<td>Seeking larger community or features not available in MXNet</td>
</tr>
<tr>
<td>Caffe 2</td>
<td>Computer vision heritage</td>
<td>Seeking larger community or need features not available in Caffe</td>
</tr>
<tr>
<td>scikit-learn</td>
<td>General purpose ML</td>
<td>Deep learning, need GPU</td>
</tr>
</tbody>
</table>
## Modules and toolkits - Prepackaged models

<table>
<thead>
<tr>
<th>Name</th>
<th>Application</th>
<th>Language and Libraries Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>spaCy</td>
<td>Natural language processing</td>
<td>Python, TensorFlow, PyTorch</td>
</tr>
<tr>
<td>TF-Slim</td>
<td>Image classification</td>
<td>TensorFlow</td>
</tr>
<tr>
<td>TF-object detection</td>
<td>Object detection</td>
<td>TensorFlow</td>
</tr>
<tr>
<td>TensorFlow Hub</td>
<td>Various</td>
<td>TensorFlow</td>
</tr>
<tr>
<td>Caffe Model Zoo</td>
<td>Various</td>
<td>Caffe</td>
</tr>
<tr>
<td>TensorFlow models</td>
<td>Various</td>
<td>TensorFlow</td>
</tr>
<tr>
<td>Keras applications</td>
<td>Various</td>
<td>Keras</td>
</tr>
</tbody>
</table>
## Tools of the trade

### Scripting tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>When to Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python + argparse</td>
<td>Create reusable scripts with well defined interfaces</td>
</tr>
<tr>
<td>Guild AI</td>
<td>Capture script output as ML experiments</td>
</tr>
<tr>
<td>Paver</td>
<td>Python make-like tool</td>
</tr>
<tr>
<td>Traditional build tools</td>
<td>General purpose build automation</td>
</tr>
<tr>
<td>(make, cmake, ninja)</td>
<td></td>
</tr>
</tbody>
</table>
## Tools of the trade

### Workflow automation

<table>
<thead>
<tr>
<th>Tool</th>
<th>When to Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLFlow</td>
<td>Enterprise wide machine learning workflow</td>
</tr>
<tr>
<td>Guild AI</td>
<td>Ad hoc workflows, integration with other automation systems</td>
</tr>
<tr>
<td>Polyaxon</td>
<td>Kubernetes based job scheduling</td>
</tr>
<tr>
<td>Airflow</td>
<td>General workflow automation</td>
</tr>
<tr>
<td>Traditional scripting</td>
<td>Ad hoc automation</td>
</tr>
</tbody>
</table>
Data analysis

Chart showing quarterly value of wheat, 1821 (source)
Structured vs unstructured data

**Structured Data**
*Classification chart of Factory Ledger Accounts, 1919 (source)*

**Unstructured Data**
*Darwin’s Finches, 1837 (source)*
Data analysis

Visualization

Matplotlib
Plotly
Visdom
H20.ai
Seaborn

Many, many more!
Model selection
(Representation)
Model selection

Standard architectures

CNN, RNN, LSTM, GAN, NAT, AutoML, SVM etc...
Model selection

Hand engineered or learned?

**Hand Engineered**
- Rely on experience and recommendation of experts
- Experiment with novel changes to hyperparameters and architecture
- Best place to start

**Learned**
- AutoML for hyperparameter and simple architectural optimization
- Neural architecture search to learn entire architecture on data
- Advanced technique
Model selection

Runtime performance criteria

Accuracy/Precision
- Various measurements (e.g. accuracy, precision, recall)
- Metrics depend on prediction task

Speed/Latency
- Inference time per example
- Inference time per batch
- Model and runtime environment interaction

Resource Constraints
- Required memory and power
- Model/runtime environment interaction
- Mobile and embedded devices severely constrained
Model selection

Training performance criteria

- Training Progress
  - Training and validation loss/accuracy
  - Time/epochs to convergence
  - Vanishing/exploding gradient

- Time to Train
  - Model training time can vary by order of magnitude
  - Longer runs mean fewer trials
  - Direct impact on time-to-market

- Cost
  - GPU / HPC time is expensive
  - Opportunity cost of not training other models
### Model selection

#### Sample trade off comparison

<table>
<thead>
<tr>
<th>Task: image classification</th>
<th>Logistic Regression</th>
<th>3 Layer CNN</th>
<th>ResNet-50</th>
<th>NASNET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Very High</td>
</tr>
<tr>
<td>Inference Memory</td>
<td>Very Low</td>
<td>Low</td>
<td>High</td>
<td>Very High</td>
</tr>
<tr>
<td>Inference Latency</td>
<td>Very Low</td>
<td>Low</td>
<td>High</td>
<td>Very High</td>
</tr>
<tr>
<td>Training Time</td>
<td>Very Low</td>
<td>Low</td>
<td>High</td>
<td>Very High</td>
</tr>
<tr>
<td>Training Cost</td>
<td>Very Low</td>
<td>Very Low</td>
<td>Medium</td>
<td>Medium</td>
</tr>
</tbody>
</table>
Training

Wanderer above the Sea of Fog,
Caspar David Friedrich, 1818 (source)
Training

Primary training patterns

- Train from scratch
- Transfer learn
- Fine tune
- Retrain
Train from Scratch

Wooden frame construction in Sabah, Malaysia
(source)
Training

Transfer Learn

“The Barge” at PolarTrec Northeast Scientific Station, Siberia Russia (source)
Fine Tune

WTC under construction, April 2012 (source)
Framing for new addition to home
(source)
## Training techniques

<table>
<thead>
<tr>
<th>When</th>
<th>Train from Scratch</th>
<th>Transfer Learn</th>
<th>Fine Tune</th>
<th>Retrain</th>
</tr>
</thead>
<tbody>
<tr>
<td>No pretrained models</td>
<td>No pretrained models</td>
<td>Pretrained models for different task</td>
<td>Pretrained model for same task</td>
<td>Pretrained model same task, different number of output classes</td>
</tr>
<tr>
<td>Data Requirements</td>
<td>Highest</td>
<td>Reduced</td>
<td>Reduced</td>
<td>Reduced</td>
</tr>
<tr>
<td>Training Time</td>
<td>Highest</td>
<td>Reduced</td>
<td>Reduced to Unchanged</td>
<td>Reduced</td>
</tr>
<tr>
<td>Domains/tasks involved</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>When Used</td>
<td>No pretrained model, lots of data and compute resources, highest accuracy required</td>
<td>Pretrained model, limited data and compute resources</td>
<td>Pretrained model, additional data or compute resources to improve accuracy</td>
<td>Pretrained model for same task, need to remove or add classes</td>
</tr>
</tbody>
</table>
TF Slim transfer learn example

$ python train_image_classifier.py
   --model_name resnet-50
   --dataset_dir ./prepared-data
   --train_dir train
   --checkpoint_path checkpoint/resnet_v1_50.ckpt
   --checkpoint_exclude_scopes resnet_v1_50/logits
   --trainable_scopes resnet_v1_50/logits

https://github.com/tensorflow/models/tree/master/research/slim
TF Slim transfer learn example

$ python train_image_classifier.py
  --model_name resnet-50
  --dataset_dir ./prepared-data
  --train_dir train
  --checkpoint_path checkpoint/resnet_v1_50.ckpt
  --checkpoint_exclude_scopes resnet_v1_50/logits
  --trainable_scopes resnet_v1_50/logits

Model architecture (network)
Training

TF Slim transfer learn example

```
$ python train_image_classifier.py
   --model_name resnet-50
   --dataset_dir ./prepared-data
   --train_dir train
   --checkpoint_path checkpoint/resnet_v1_50.ckpt
   --checkpoint_exclude_scopes resnet_v1_50/logits
   --trainable_scopes resnet_v1_50/logits
```

New data for new task
TF Slim transfer learn example

$ python train_image_classifier.py
    --model_name resnet-50
    --dataset_dir ./prepared-data
    --train_dir train
    --checkpoint_path checkpoint/resnet_v1_50.ckpt
    --checkpoint_exclude_scopes resnet_v1_50/logits
    --trainable_scopes resnet_v1_50/logits

Model weights from source task (ImageNet)
Training

TF Slim transfer learn example

$ python train_image_classifier.py
   --model_name resnet-50
   --dataset_dir ./prepared-data
   --train_dir train
   --checkpoint_path checkpoint/resnet_v1_50.ckpt
   --checkpoint_exclude_scopes resnet_v1_50/logits
   --trainable_scopes resnet_v1_50/logits

Layer weights to not initialize from checkpoint (unfrozen)
TF Slim transfer learn example

$ python train_image_classifier.py
   --model_name resnet-50
   --dataset_dir ./prepared-data
   --train_dir train
   --checkpoint_path checkpoint/resnet_v1_50.ckpt
   --checkpoint_exclude_scopes resnet_v1_50/logits
   --trainable_scopes resnet_v1_50/logits

Layer weights to train (freeze all others)
Training

Hyperparameters and tuning

What combination of hyperparameters will train the best model on our data?

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Search Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>learning-rate</td>
<td>uniform from 1e-4 to 1e-1</td>
</tr>
<tr>
<td>activation</td>
<td>choice of “relu” or “sigmoid”</td>
</tr>
<tr>
<td>dropout</td>
<td>uniform from 0.1 to 0.9</td>
</tr>
</tbody>
</table>

```
$ python train.py
   --learning-rate=0.01
   --activation=relu
   --dropout=0.2
```

Bayesian Optimization

Credit: Hutter & Vanschoren (source)
Hyperparameter tuning example

During the development of AlphaGo, its many hyperparameters were tuned with Bayesian optimization multiple times.

This automatic tuning process resulted in substantial improvements in playing strength. For example, prior to the match with Lee Sedol, we tuned the latest AlphaGo agent and this improved its win-rate from 50% to 66.5% in self-play games. This tuned version was deployed in the final match.

Of course, since we tuned AlphaGo many times during its development cycle, the compounded contribution was even higher than this percentage.
Training

Architecture search (advanced topic)

Typical Model Layout (layers)

Layers with branches and skip connections

Fom Automatic Machine Learning (AutoML): A Tutorial at NeurIPS 2018 (source)
Distributed training

Credit: Lim, Andersen, and Kaminsky (source)
Motivations for distribution

Training

Too Much Data
Large model (e.g. ResNet-200)
Large batch size (effects accuracy and total training time)

Not Enough Wall Time
Use data or task parallelism to distribute training over multiple GPUs
Data preparation and processing

Women lumberjacks at Pityoulish lumber camp, 1941 (source)
Role of data preparation and processing

Data preparation and processing

Data Source -> Data Retrieval

Data Preparation and Processing

Manual Feature Selecting and Engineering

Automate Feature Engineering

Modeling

Machine Learning Algorithm

Model Evaluation and Tuning

Deployment and Monitoring

Data preparation and processing

Feature detection in neural networks

Credit: Sootla (source)
Data preparation and processing

Feature selection and engineering

- Features available
- Features to manually create
- Features to auto-generate

When you don't have enough data for deep learning

Train Model

Model Performance Results
Data preparation and processing

Data splitting and test rules

- Training algorithm never, ever sees validation and test data.
- Training orchestrator never, ever sees test data.
- Test data is used for final scoring and once used becomes validation data.
- Validation and test data much have the same distribution as training data.
The Roquefavour bridge-aqueduct over the Canal de Marseille (source)
Environment isolation

Development -> Test -> Production
Workflow management and job scheduling

Infrastructure

Apache Airflow
- Automate data pipelines
- ETL
- General workflow

Jenkins
- Continuous integration
- Automate software production pipelines
- General workflow

Kubernetes
- Container orchestration
- General purpose application platform
Cloud services and accelerators

**AWS**
- General purpose IaaS
- Standard GPU options
- Track record of improving performance while lowering prices

**GCP**
- General purpose IaaS
- Standard GPU options and TPUs
- Complement to TensorFlow ecosystem

**Azure**
- General purpose IaaS
- Standard GPU options

**Kubernetes**
- Container orchestration
- General purpose application platform

**Other GPU**
- Dedicated GPU servers - on prem or hosted in datacenter
- Paperspace
- FloydHub
Reproducibility

Early wooden printing press, 1568 (source)
Reproducibility

Source code revisions

<table>
<thead>
<tr>
<th>Branch</th>
<th>New pull request</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td></td>
</tr>
</tbody>
</table>

Open source experiment tracking and optimization for machine learning [https://guild.ai](https://guild.ai)

Manage topics

- **2,365 commits**
- **17 branches**
- **0 releases**
- **1 contributor**

This branch is 269 commits ahead, 7 commits behind master.

<table>
<thead>
<tr>
<th>User</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>gar1t</td>
<td>Better default for poly example</td>
<td>Latest commit cf0751f 9 days ago</td>
</tr>
<tr>
<td>.circleci</td>
<td>Don't upgrade brew</td>
<td>22 days ago</td>
</tr>
<tr>
<td>guild</td>
<td>Better default for poly example</td>
<td>9 days ago</td>
</tr>
<tr>
<td>guildai.dist-info</td>
<td>Move optimizers to plugins</td>
<td>20 days ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Missing pip files + cleanup gitignores</td>
<td>6 months ago</td>
</tr>
<tr>
<td>.pypi-creds.gpg</td>
<td>Store encrypted PyPi credentials</td>
<td>a year ago</td>
</tr>
</tbody>
</table>
Reproducibility

Data versioning and auditability

<table>
<thead>
<tr>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple, universal interface</td>
<td>Batch oriented</td>
</tr>
<tr>
<td>Wide range of tooling</td>
<td>Highly latent</td>
</tr>
<tr>
<td>Secure</td>
<td></td>
</tr>
<tr>
<td>Easily auditable</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real time oriented</td>
<td>Complex</td>
</tr>
<tr>
<td>Low latency</td>
<td></td>
</tr>
<tr>
<td>Checkpointing depending on DBMS</td>
<td></td>
</tr>
</tbody>
</table>
Reproducibility

Experiment automation and management

$ guild run train.py lr=0.1
Refreshing project info...
You are about to run train.py
  batch_size: 100
  epochs: 10
  lr: 0.1
Continue? (Y/n)

$ guild ls
~/.guild/runs/072817ee348d11e98c6cc85b764bbf34:
  data/
    data/t10k-images-idx3-ubyte.gz
    data/t10k-labels-idx1-ubyte.gz
    data/train-images-idx3-ubyte.gz
    data/train-labels-idx1-ubyte.gz
  model/
    model/checkpoint
    model/export.data-00000-of-00001
    model/export.index
    model/export.meta
  train/
    train/events.out.tfevents.1550611600.omaha
  validate/
    validate/events.out.tfevents.1550611600.omaha

Experiment

- Metadata (unique ID, model, operation, hyperparameters, time)
- Source code snapshot
- Output (stdio)
- Logs
- Metrics (e.g. loss, accuracy)
- Generated files (e.g. checkpoints)
Reproducibility

Experiment automation and management

- No matter how good the result, if it’s not reproducible, it’s not ready to ship.
- Code review equivalent: *can another engineer easily reproduce this result?* It’s a pass/fail grade.
- Without reproducibility, organization is exposed to enormous risk.
- Runs counter to traditional data science tendencies to keep results, tools, and knowledge private.
Production

Water-wheel at London Bridge, 1749 (source)
<table>
<thead>
<tr>
<th>Serving systems</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Batch Inference</strong></td>
</tr>
<tr>
<td>Accumulate examples in a file system directory or other similar container (e.g. S3 bucket)</td>
</tr>
<tr>
<td>Run batch job to process examples and perform inference (e.g. predict image class)</td>
</tr>
<tr>
<td>Simple and effective but highly latent - not suitable for low latency applications</td>
</tr>
<tr>
<td><em>Start here if possible</em></td>
</tr>
<tr>
<td><strong>Online Inference</strong></td>
</tr>
<tr>
<td>Requires a serving system (e.g. TF Serving, Deep Detect, or cloud server like Google Cloud ML) - or a Python based system like Flask</td>
</tr>
<tr>
<td>Process examples as they are submitted or accumulate a batch of min size (efficiency)</td>
</tr>
<tr>
<td>Python not suitable for performance critical applications - triggers need for native execution</td>
</tr>
<tr>
<td><em>Complex at scale</em></td>
</tr>
</tbody>
</table>
Production

TensorFlow Serving

Serve models in production with TensorFlow Serving

Source
Production
Mobile and embedded platforms

TensorFlow Lite (ML Kit)
- By Google
- Closely tied to TensorFlow ecosystem
- Android and iOS
- 8 bit quantization

TensorRT
- By NVIDIA
- Embedded and datacenter
- Support for
  - 8 bit quantization

CoreML
- By Apple
- iOS only
- 8, 4, 2, or 1 bit quantization

Embedded
- By Intel, ARM, Samsung, lots more
- Varied applications and platform support
Production

Monitoring model and application performance

Open Source
- Prometheus
- Kibana
- Sensu
- Nagios
- Zabbix

Hosted
- Data Dog
- New Relic
- App Dynamics
- AWS Cloud Watch
- Google Stack Driver
Ongoing development

Chicago in 1820 (source)
Ongoing development

Upgrading production systems

1. Blue service active
2. Stage green service
3. Test green service, comparing performance to blue
4. When ready, promote green to active in router
5. Green service active
6. Users and services happily unaware of upgrade (zero downtime, zero faults)
Ongoing development

Acquiring more data

- Build data acquisition into the application
- Run as a long running, continual process
- Look for new applications to collect new data
- Can bootstrap an application with limited data provided application can collect more
General guidelines

- Commonly revisit: model architecture, data acquisition, data processing, and retraining
- Production systems rely heavily on traditional systems engineering practices that cannot be short-circuited
- Again, without measuring, you’re guessing - even a nominal data collection facility is better than nothing
- Stress collaboration between researchers and engineers
The Story of a Little Gray Mouse, 1945 (source)