Computing paradigm shift waves

- **Centralized**
- **Distributed**

- **Mainframe**
- **Personal computing**
- **Azure**
- **Intelligent cloud**
- **Intelligent cloud + edge**
- **IoT in Action**
- **Microsoft**
Intelligent cloud & intelligent edge, infused by AI

“A new technology paradigm is emerging, one with an intelligent cloud and an intelligent edge.”

-- Satya at Build 2017
Driving forces towards intelligent edge

Data explosion from fast growing edge devices
E.g., smart surveillance cameras, self-driving cars

Strong needs of intelligence on edge
Lower latency
Higher availability
Better privacy
Lower cost

Edge devices becoming increasingly powerful
Emerging high-perf, low-power, low-cost AI ASIC
Agenda

Research Areas in our group
DeepCache: Principled Cache for Mobile Deep Vision
Enabling Efficient DNN Inference on Customized Hardware
Other on-going research efforts

Platform and Tools for AI
Open Platform for AI
Neural Network Intelligent
MMdnn
DeepCache: Principled Cache for Mobile Deep Vision

Continuous mobile vision apps

- Cognitive assistance
- Street navigation
- Mixed reality
Deep vision using on-device DNN models

- Convolution
- Pooling
- Fully-Connect

(Cat, 0.67)
(Dog, 0.21)
(Pig, 0.06)
Deep vision models are expensive.

Image recognition

- AlexNet (2012)
  - 8 layers
  - 1.4 GFLOPS
  - ~16% error

  - 152 layers
  - 22.6 GFLOPS
  - ~3.5% error

16X
CNN layers are primary performance hotspots

<table>
<thead>
<tr>
<th>Model</th>
<th>Lib</th>
<th>conv</th>
<th>fc</th>
<th>pl</th>
<th>act</th>
<th>rest</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet [40]</td>
<td>TF</td>
<td>79.2%</td>
<td>6.4%</td>
<td>11.1%</td>
<td>2.7%</td>
<td>0.6%</td>
</tr>
<tr>
<td></td>
<td>ncnn</td>
<td>77.9%</td>
<td>7.1%</td>
<td>12.1%</td>
<td>1.8%</td>
<td>1.1%</td>
</tr>
<tr>
<td>GoogLeNet [55]</td>
<td>TF</td>
<td>80.2%</td>
<td>0.1%</td>
<td>7.5%</td>
<td>8.1%</td>
<td>4.3%</td>
</tr>
<tr>
<td></td>
<td>ncnn</td>
<td>78.8%</td>
<td>0.7%</td>
<td>8.6%</td>
<td>9.3%</td>
<td>2.6%</td>
</tr>
<tr>
<td>ResNet-50 [33]</td>
<td>TF</td>
<td>91.8%</td>
<td>5.8%</td>
<td>0.5%</td>
<td>1.7%</td>
<td>0.2%</td>
</tr>
<tr>
<td></td>
<td>ncnn</td>
<td>93.7%</td>
<td>4.9%</td>
<td>0.8%</td>
<td>0.4%</td>
<td>0.2%</td>
</tr>
<tr>
<td>YOLO [51]</td>
<td>TF</td>
<td>82.4%</td>
<td>12.8%</td>
<td>2.1%</td>
<td>1.8%</td>
<td>0.9%</td>
</tr>
<tr>
<td></td>
<td>ncnn</td>
<td>84.1%</td>
<td>12.2%</td>
<td>2.6%</td>
<td>0.9%</td>
<td>0.2%</td>
</tr>
<tr>
<td>Dave-orig [22]</td>
<td>TF</td>
<td>58.8%</td>
<td>28.6%</td>
<td>4.8%</td>
<td>2.9%</td>
<td>5.2%</td>
</tr>
<tr>
<td></td>
<td>ncnn</td>
<td>62.7%</td>
<td>25.9%</td>
<td>5.8%</td>
<td>3.7%</td>
<td>1.9%</td>
</tr>
</tbody>
</table>
Observation: content redundancy in frames

Key idea: explore the **temporal locality** in mobile video streams (reuse the computation results of the same/similar blocks in previous frames)
Architecture

- No cloud offloading
- Support all CNN models we know
- App transparent
- Minor accuracy loss
Evaluation results

Datasets
UCF101
Nvidia driving

Five CNN models
AlexNet, GoogleNet, Res-Net50, YOLO, Dave-orig

Results
18.2% saving on average (up to 47.1%)
2x saving of DeepMon
Enabling Efficient DNN Inference on Customized Hardware

Deep learning requires an amazing amount of computing power.
Online service: latency sensitive

Online Service → Latency sensitive → No batching → Limited Parallelism
<table>
<thead>
<tr>
<th>Algorithm: Model Compression</th>
<th>Hardware: General-purpose Processors</th>
</tr>
</thead>
</table>
| 10~50 × Compression          | 2~10 × Speedup                      

**Hardware:**
- **General-purpose Processors**
- **GPU**
Algorithm: Model Compression
10~50 × Compression

Hardware: Customized Accelerator
100~1000 × Speedup

High Accuracy
Energy Efficient
Low Cost
Fast & Low latency
Speedup and Accuracy Tradeoff

- **Fine-grained Sparsity**
  - High model accuracy
  - High compression rate
  - Irregular pattern
  - Difficult to accelerate

- **Vector-level Sparsity**
  - Low model accuracy
  - Low compression rate
  - Regular pattern
  - Easy to accelerate

- **Kernel-level Sparsity**
  - Regular pattern

- **Neuron-level Sparsity**
  - Irregular pattern

---

Microsoft
Speedup and Accuracy Tradeoff

Can FPGA (Customized Hardware) go beyond this? Yes!

Coarse-grained: Regular but not accurate
Fine-grained: Accurate but irregular

Model Loss
the smaller

Model Accuracy
the higher

Sparsity
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

- Original
- Fine-grained
- Block Size (4*4)
- Block Size (8*8)
- Block Size (16*16)

Can FPGA (Customized Hardware) go beyond this? Yes!

- Fine-grained: Accurate but irregular
- Coarse-grained: Regular but not accurate

Model Accuracy
the higher

Model Loss
the smaller

Sparsity
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
FPGA: Efficient buffer + MIMD
### Driving Case: 3 layer LSTM for Text to Speech

<table>
<thead>
<tr>
<th></th>
<th>Matrix-Vector Mul</th>
<th>Element Wise Op</th>
<th>Compt. Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input ffdnn</td>
<td>630x512</td>
<td>0</td>
<td>0.6 Million Ops</td>
</tr>
<tr>
<td>LSTM 1</td>
<td>640x 1024 x4+1024*128</td>
<td>1024*4</td>
<td>5.2 Million Ops</td>
</tr>
<tr>
<td>LSTM 2</td>
<td>256x 1024 x4+1024*128</td>
<td>1024*4</td>
<td>2.1 Million Ops</td>
</tr>
<tr>
<td>LSTM 3</td>
<td>256x 1024 x4+1024*128</td>
<td>1024*4</td>
<td>2.1 Million Ops</td>
</tr>
<tr>
<td>Output ffdnn</td>
<td>128x508</td>
<td>0</td>
<td>0.26 Million Ops</td>
</tr>
</tbody>
</table>
Other on-going research efforts

Empower edge devices with AI
- Model compression
- System optimizations

New techniques and scenarios
- Distributed, collaborative learning
- Continuous, incremental learning

Privacy and security
- User data protection
- Model protection
The Importance of AI Platform and Tools

Infrastructure support for the advance of artificial intelligence
A platform to boost AI innovation and productivity

Massive labelled data  
Deeper and Larger model

Platform&Tools for AI  
Powerful computing power

- Efficient and customizable
- Scalable and compatible
- Intelligent and adaptable
Deep Learning Incubation Investments

Automatic DL experimental process
• Neural network intelligence (aka. nni). Open sourced https://github.com/Microsoft/nni

DL job management and scheduling system
• Open platform for AI (aka. OpenPAI). Open sourced https://github.com/Microsoft/pai

Tools for AI
• An extension to build, test, and deploy Deep Learning / AI solutions https://github.com/Microsoft/vs-tools-for-ai

DL framework language, compiler and optimization
• Language: Julia, MMdnn; Compiler: Wolong.
MMdnn
Model Management for deep neural networks

https://github.com/Microsoft/mmdnn
Want to try DNNs?

You can choose:

- ONNX
- theano
- TENS OF THEM!
- torch
- Caffe2
- PyTorch
- Keras

IoT in Action
How to interoperate and collaborate? People keep asking
Looking for Model Interoperation

We mine search queries in Bing

- Universal Visualization and Editing
- Training/Inference Code Snippet Gen
- Pre-trained Model Converter
- Compatibility Testing

* Bigger the arrow, more popular the system is.
Code Gen and Model Zoo

Generate code from one framework to others

Caffe → TensorFlow → CNTK
# Code Gen and Model Zoo

<table>
<thead>
<tr>
<th>Models</th>
<th>Caffe</th>
<th>Keras</th>
<th>Tensorflow</th>
<th>CNTK</th>
<th>MXNet</th>
<th>PyTorch</th>
<th>CoreML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception V1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>x (No LRN)</td>
</tr>
<tr>
<td>Inception V3</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Inception V4</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ResNet V1 50</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ResNet V2 152</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>VGG 19</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MobileNet_v1</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓ (only Relu)</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Xception</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
</tbody>
</table>
## Model Conversion

Before MMdnn, we have to make it one by one.

<table>
<thead>
<tr>
<th>converter</th>
<th>mxnet</th>
<th>caffe sparsify</th>
<th>caffe2</th>
<th>CNTK</th>
<th>theano/laforge</th>
<th>neon</th>
<th>pytorch sparsify</th>
<th>torch</th>
<th>keras sparsify</th>
<th>chain</th>
<th>coreML/ROS sparsify</th>
</tr>
</thead>
<tbody>
<tr>
<td>mxnet</td>
<td>-</td>
<td>MMdnn</td>
<td>None</td>
<td>MMdnn</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>MMdnn</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>caffe sparsify:</td>
<td>none</td>
<td>caffe2toCaffe2</td>
<td>None</td>
<td>-</td>
<td>ONNX</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>caffe2</td>
<td>None</td>
<td>None</td>
<td>-</td>
<td>ONNX</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>CNTK</td>
<td>MMdnn</td>
<td>MMdnn</td>
<td>ONNX</td>
<td>-</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>theano/laforge</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>-</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>neon</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>-</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>pytorch sparsify:</td>
<td>None</td>
<td>pytorchcaffe</td>
<td>None</td>
<td>-</td>
<td>ONNX</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>torch</td>
<td>None</td>
<td>fbcaffe-conda</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>keras sparsify:</td>
<td>MMdnn</td>
<td>MMdnn</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>darknet</td>
<td>None</td>
<td>pytorch-darknet-convert</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>tensorflow</td>
<td>MMdnn</td>
<td>MMdnn</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>chain</td>
<td>None</td>
<td>MMdnn</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>coreML/ROS sparsify:</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>-</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

O(n²)
Model Conversion

With MMdnn, we make it much simpler
nni
Neural Network Intelligence

https://github.com/Microsoft/nni
nni Design Conceptual View

Command line tool: **nnictl**

Visualized UI: **nni board**

**User Code + SDK (import nni)**
- CNTK
- TensorFlow
- PyTorch
- ...
  (Python based frameworks)

**Tuning Algorithm Extensions**
- TPE
- Random
- Anneal
- Naïve Evolution
- SMAC
- Batch
- Grid
- Hyperband
- ENAS *
- ...

**Training Service Extensions**
- Local machine
- Remote servers
- OpenPAI
- K8S services (AKS etc.)
- ...

* Under development feature, coming soon
How to Use nni

Step 1. Define SearchSpace

Step 2. Update Codes

Step 3. Define Experiment
Trial and Algorithm SDK

Trial SDK

```python
trial_config = get_parameter()
report_intermediate_result(metric)
report_final_result(metric)
```

Tuner and Assessor SDK

```python
class Tuner(Recoverable):
    def update_search_space(self, search_space)
    def generate_parameters(self, parameter_id)
    def receive_trial_result(self, parameter_id, parameters, reward)
    def receive_customized_trial_result(self, parameter_id, parameters, reward)

class Assessor(Recoverable):
    def assess_trial(self, trial_job_id, trial_history)
    def trial_end(self, trial_job_id, success)
```
Training Service SDK

General training service interface
Could easily support other platforms, such as K8S, AWS.

```java
abstract class TrainingService {
    public abstract listTrialJobs(): Promise<TrialJobDetail[]>;
    public abstract getTrialJob(trialJobId: string): Promise<TrialJobDetail>;
    public abstract addTrialJobMetricListener(listener: (metric: TrialJobMetric) => void): void;
    public abstract removeTrialJobMetricListener(listener: (metric: TrialJobMetric) => void): void;
    public abstract submitTrialJob(form: JobApplicationForm): Promise<TrialJobDetail>;
    public abstract cancelTrialJob(trialJobId: string): Promise<void>;
    public abstract setClusterMetadata(key: string, value: string): Promise<void>;
    public abstract getClusterMetadata(key: string): Promise<string>;
    public abstract cleanUp(): Promise<void>;
    public abstract run(): Promise<void>;
}
```
Search Results
Annotation for Better Experience

Advantages of nni annotation
Exist as comments in Python code, be able to run both independently and on nni

```python
"""@nni.variable(nni.choice(2, 3, 5, 7), name=self.conv_size)"
self.conv_size = conv_size
"""@nni.variable(nni.choice(124, 512, 1024), name=self.hidden_size)"
self.hidden_size = hidden_size
self.pool_size = pool_size
"""@nni.variable(nni.uniform(0.0001, 0.1), name=self.learning_rate)"
self.learning_rate = learning_rate

"""@nni.function_choice(max_pool(h_conv1, self.pool_size), avg_pool(h_conv1, self.pool_size), name=max_pool)"
h_pool1 = max_pool(h_conv1, self.pool_size)

test_acc = mnist_network.accuracy.eval(
    feed_dict={mnist_network.images: mnist.test.images,
               mnist_network.labels: mnist.test.labels,
               mnist_network.keep_prob: 1.0})

"""@nni.report_intermediate_result(test_acc)"

"""@nni.report_final_result(test_acc)"
```
Summary

Exciting time for intelligent edge computing!
New apps, scenarios, software, hardware, tools ...

Bring intelligence to edge devices
Sensing the rich context: users, external & internal states
Taking actions accordingly: better performance & user experience

Many research and engineering problems to solve
DRS, RAVEN, DeepCache
More on system optimizations, collaborative learning, private AI ...
OpenPAI
Open Platform for AI

https://github.com/Microsoft/pai
Background and Motivation

Systems research empowers future AI innovations
Compiler optimization for deep learning application
System primitives for deep learning scheduling

Innovating in an open way
Internal and external collaborations (with MSR labs and Universities)
An open source ecosystem to innovate with the community
Gandiva: Introspective Cluster Scheduling for Deep Learning

A new scheduler architecture catering to the key characterizations of deep learning training

System innovations bring an order of magnitude efficiency gains

The first deep learning scheduling system in the top system conference (OSDI’18)

Joint work with MSR India
## Deep Learning Training vs. Big Data Processing

<table>
<thead>
<tr>
<th>Deep Learning Training</th>
<th>Big Data Processing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal</strong></td>
<td>Compute a set of data</td>
</tr>
<tr>
<td><strong>Process</strong></td>
<td>Run one job to completion</td>
</tr>
<tr>
<td>Trial-and-error: see the training accuracy of hyper-parameters</td>
<td>- may have sub-tasks (e.g., MapReduce)</td>
</tr>
<tr>
<td>- a trial job can last for hours or days</td>
<td>- sub-tasks are short-lived (minutes)</td>
</tr>
<tr>
<td>- it might take lots of jobs (~100)</td>
<td></td>
</tr>
<tr>
<td>May stop a job early if not promising</td>
<td></td>
</tr>
<tr>
<td>AutoML: automate the process</td>
<td></td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Run the job as fast as possible</td>
</tr>
<tr>
<td>Find the model as fast as possible</td>
<td></td>
</tr>
<tr>
<td>- Performance of one job might not matter</td>
<td></td>
</tr>
</tbody>
</table>

Implication – More Parallel Jobs the Better

Good trial jobs surface earlier

Bad trial jobs stop earlier

More effective search in hyper-parameter space

Need *time-slicing* of jobs – GPU not efficiently virtualizable
Opportunity – Computation Boundary of Deep Learning Training
Deep learning training runs in *mini-batches*

Bulk synchronous parallel (BSP)
Computation divided by super-steps (i.e., mini-batches)
Mini-batches separated by global synchronization

Mini-batch as the computation boundary – for *time-slicing* and *migration*
Opportunity – Predictability of Mini-Batches

Mini-batch has identical timespan
  – Light-weight profiling

Low memory usage at barrier point
  – Time-slicing and migration at the barrier

ResNet50 on ImageNet data
Our Approach

*Time-slicing and migration* as the *primitives* for scheduling (similar to OS)
Mitigate head-of-line blocking
Explore more trials in parallel

Introspection: Application-aware profiling (*time-per-minibatch*)
Continuous and introspective scheduling to adapt quickly to the changing environment

Efficient implementation by exploiting the predictability
Checkpointing at the mini-batch boundary with minimum memory overhead
## A Comparison to Big Data Scheduler

<table>
<thead>
<tr>
<th>Big Data Scheduler</th>
<th>Deep Learning Scheduler</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Granularity</strong></td>
<td></td>
</tr>
<tr>
<td>MapReduce task or DFG</td>
<td>Mini-batch boundary</td>
</tr>
<tr>
<td><strong>Decision</strong></td>
<td></td>
</tr>
<tr>
<td>One-time</td>
<td>Continuous/introspective</td>
</tr>
<tr>
<td><strong>Profiling</strong></td>
<td></td>
</tr>
<tr>
<td>System-level</td>
<td>Application-level</td>
</tr>
<tr>
<td>- CPU/GPU Util., disk I/O</td>
<td>- Time-per-mini-batch</td>
</tr>
</tbody>
</table>
Performance Highlights

Time-slicing
- Less than 2% overhead

Migration
- 50x faster
- <1s migration cost
  (incl. multi-GPU, multi-Node)
Performance Highlights

AutoML model exploration

- 1.5hrs vs. 18.5hrs
- $10\times$ speedup

VGG-like model, $>90\%$ accuracy, 40 trials
Two AutoML sessions, each uses 8 GPUs
Background DLT in a 100-GPU cluster
### Beyond Research: An Open Source Stack for AI Innovation

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenPAI platform</td>
<td>Cluster management for AI training and Marketplace for AI asset sharing</td>
</tr>
<tr>
<td>NNI – Neural Network Intelligence</td>
<td>A toolkit for automated machine learning experiments</td>
</tr>
<tr>
<td>MMdnn</td>
<td>A tool to convert, visualize and diagnose deep neural network models</td>
</tr>
<tr>
<td>Tools for AI</td>
<td>An extension to build, test, and deploy deep learning/AI solutions</td>
</tr>
</tbody>
</table>
A Modularized (containerized) AI Platform

**PAI marketplace**
AI asset sharing

**PAI protocol -- Resource Specification**
data, code, docker image
Hardware requirement

**Job Launcher**
Understand PAI protocol and execute the job accordingly
Onboard new framework w/o modifications to platform

**Deployment in different environment**
Single-box, cloud, on-prem, hybrid
USTC OpenPAI Deployment

1040 GPU Deployment

- GPU card types: K80, P40, V100, MLU
- Total power: 20P Flops (including half-precision)

Developer-Oriented Customization

- Rich sample gallery
- Convenient debugging support

250000 GPU·Hour/Month

- Resource monitoring of multiple dimensions
- Notification and discovery of resource utilization bottlenecks

Various types of Running AI Jobs

- ACM Multimedia 2018 基于内容的视频推荐挑战赛 冠军
- ECCV 2018 光谱图像超分辨率挑战赛 冠军
- ECCV 2018 视觉对话挑战赛 冠军
- ICFHR 2018 东南亚视域叶面分析挑战赛 冠军
- CVPR 2018 光谱重构挑战赛 季冠军
- 下一代视觉编码技术国际评比第三（全球高校第一）
- ImageNet 国际物体检测竞赛 冠军
- ActivityNet 国际动作识别挑战赛 冠军
- TRECVID 国际视频检索评测 冠军
- 全国音视频检索识别竞赛 冠军

330 high-freq users from: Multimedia, NLP, Basic Theory, etc.
Contest support: Kuaisou recoContest, ChinaMM image compression
Neural Network Intelligence with OpenPAI

A toolkit for AutoML experiments and research
Include popular AutoML algorithms
Focus on AI research w/o worrying about system issues

Training service as the abstraction to run experiments on the platform (OpenPAI)

Gandiva Planned on OpenPAI + NNI
Thank you!