運用 Spark 與電腦視覺科技 協助瀕臨絕種的雪豹

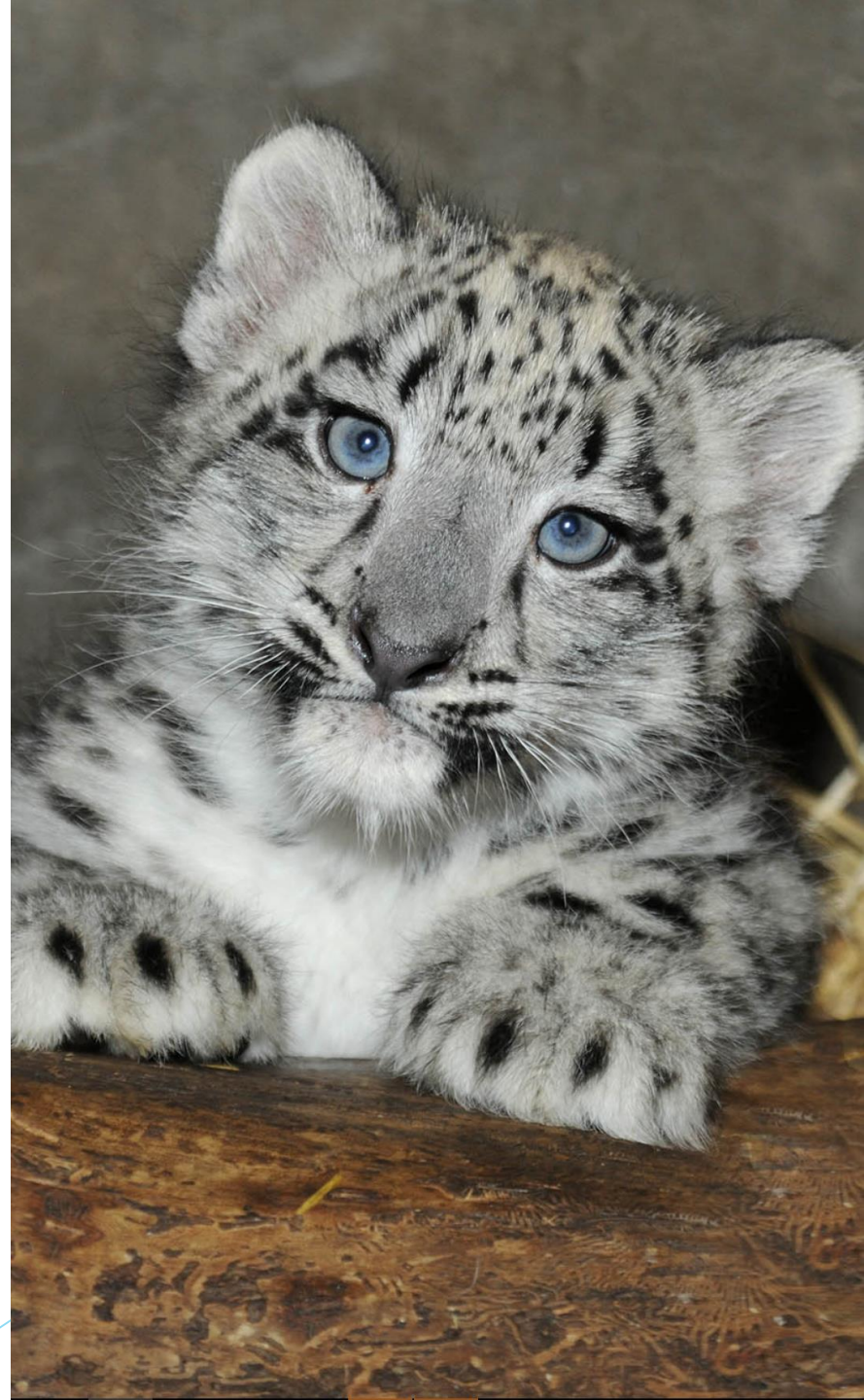
Mark Hamilton
SOFTWARE ENGINEER
Azure Machine Learning
Microsoft

Herman Wu
技術傳教士(Technical Evangelist)
前瞻技術合作事業部(CSE)
Microsoft

有「雪山之王」之稱，是一種重要的大型貓科食肉動物和旗艦種，由於其常在雪線附近和雪地間活動，故名「雪豹」

雪豹敏感、機警、喜歡獨行、夜間活動、遠離人跡和高海拔的生活特性使其行為特徵難以為人所知。到目前為止，**人類對雪豹的了解仍然十分有限**。因其處於高原生態食物鏈的頂端，雪豹亦被人們稱為「高海拔生態系統健康與否的氣壓計」。

而由於非法捕獵等多種人為因素，雪豹的數量正急劇減少，現已成為瀕危物種。**在中國，雪豹的數量甚至少於大熊貓。**



Snow Leopard Conservation

- 全球剩下3,900-6,500 隻
- 對雪豹的生態, 行為, 活動方式, 存活率了解有限
- 需要對他們有更多瞭解來協助雪豹存活

採礦

盜獵

誤入人類生活圈

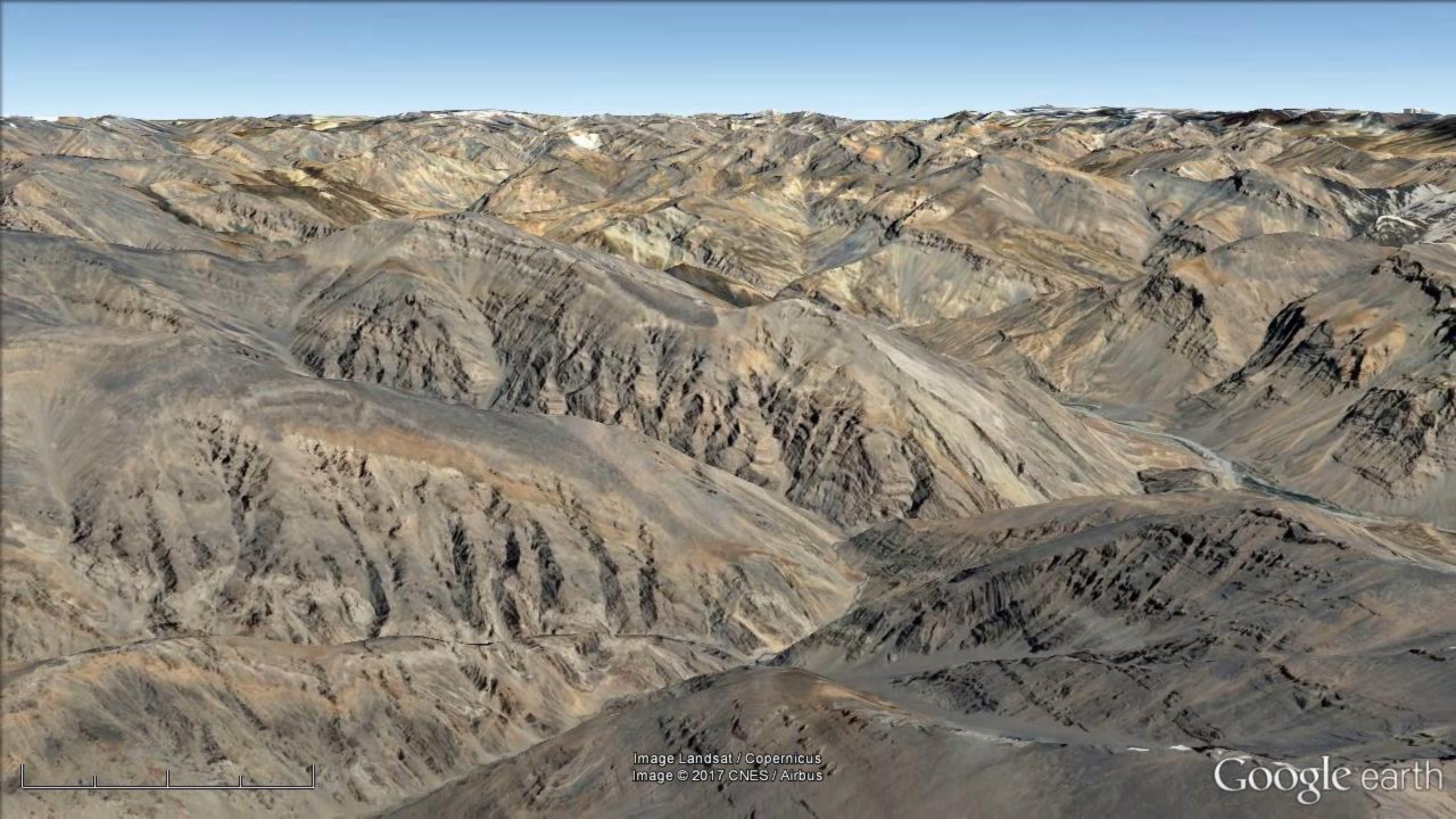
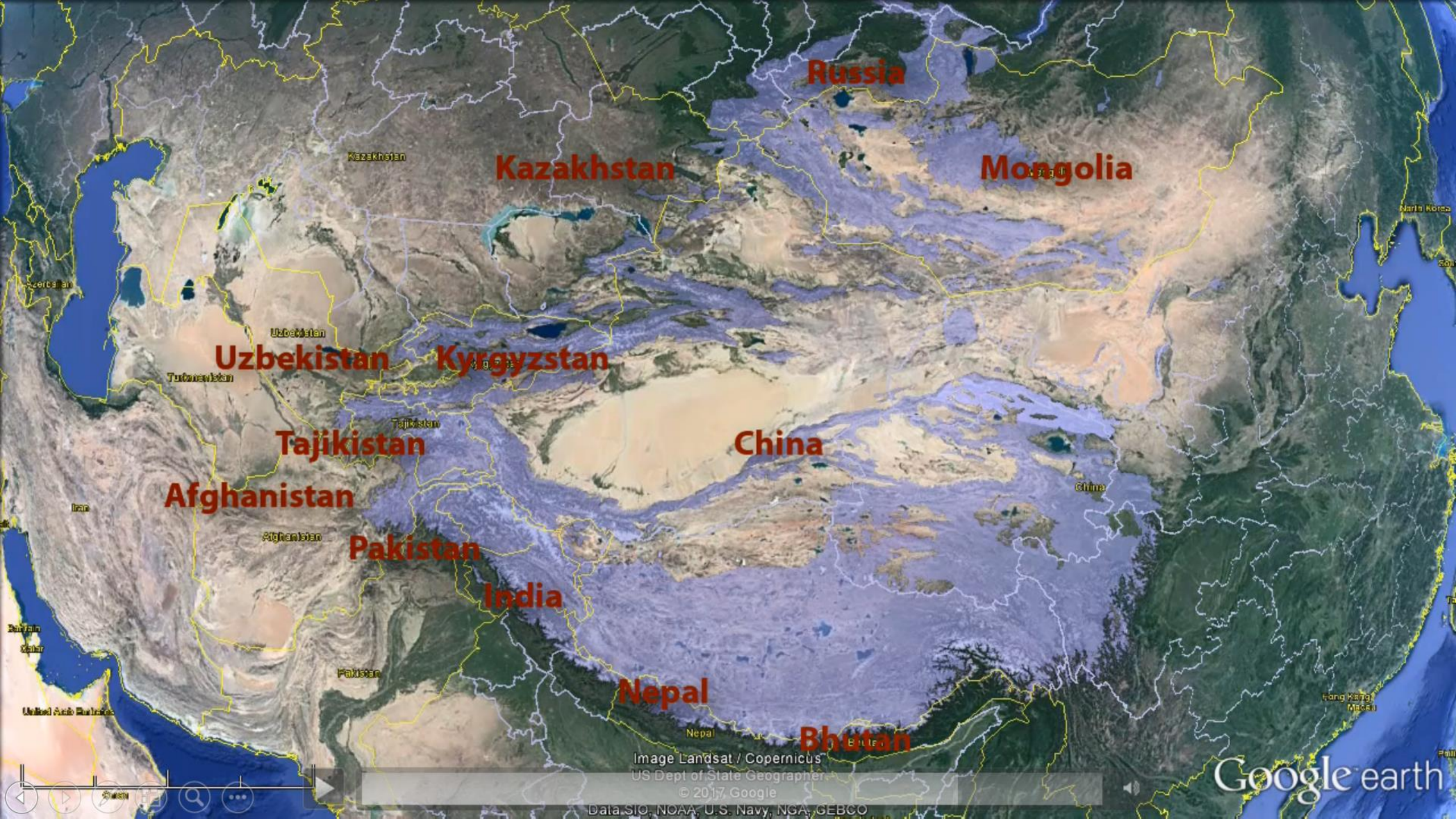
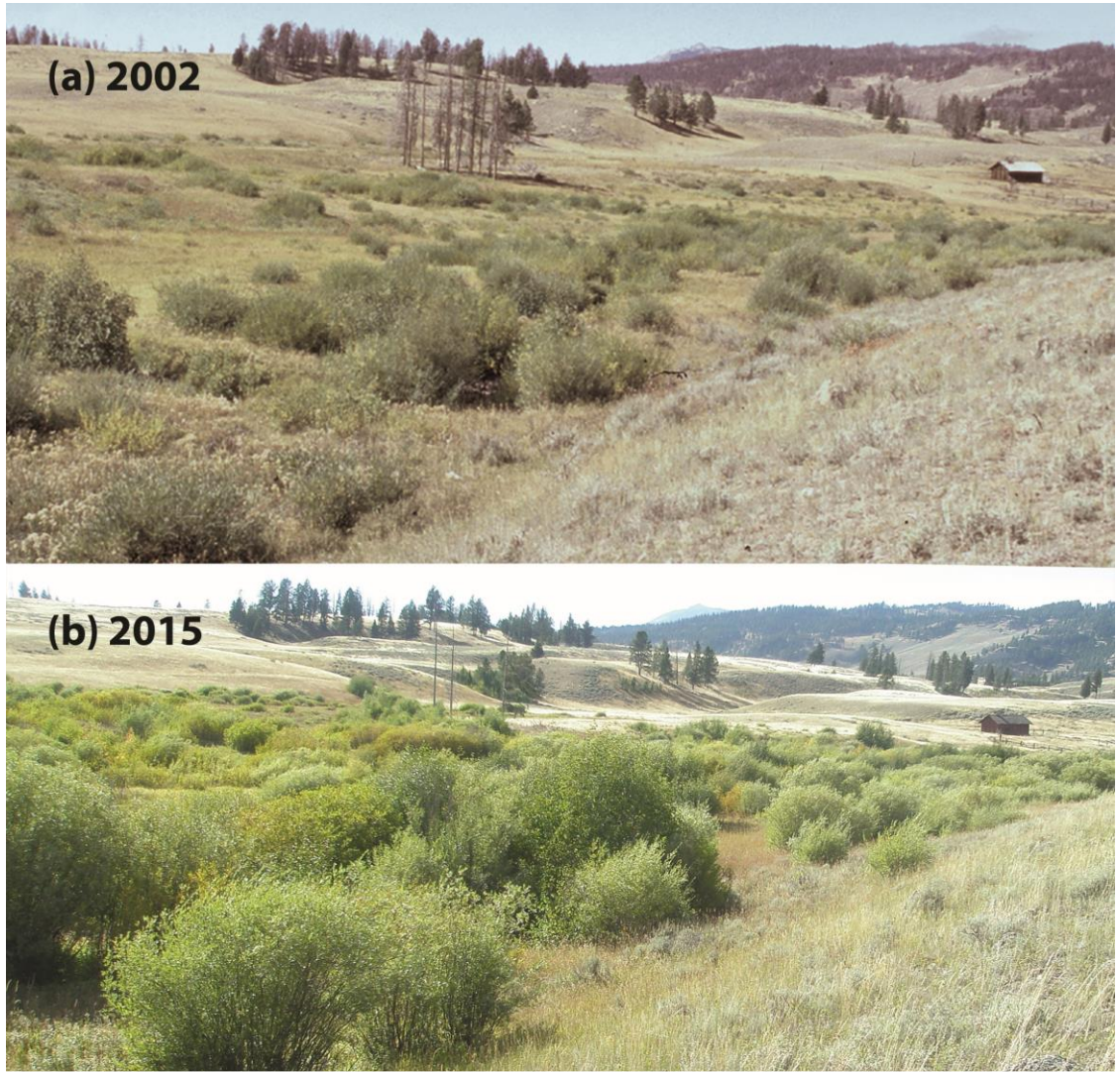


Image Landsat / Copernicus
Image © 2017 CNES / Airbus
Google earth


Russia
Kazakhstan
Mongolia
Uzbekistan
Kyrgyzstan
Tajikistan
China
Afghanistan
Pakistan
India
Nepal
Bhutan
Image Landsat / Copernicus
US Dept of State Geographer
© 2017 Google
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Google earth

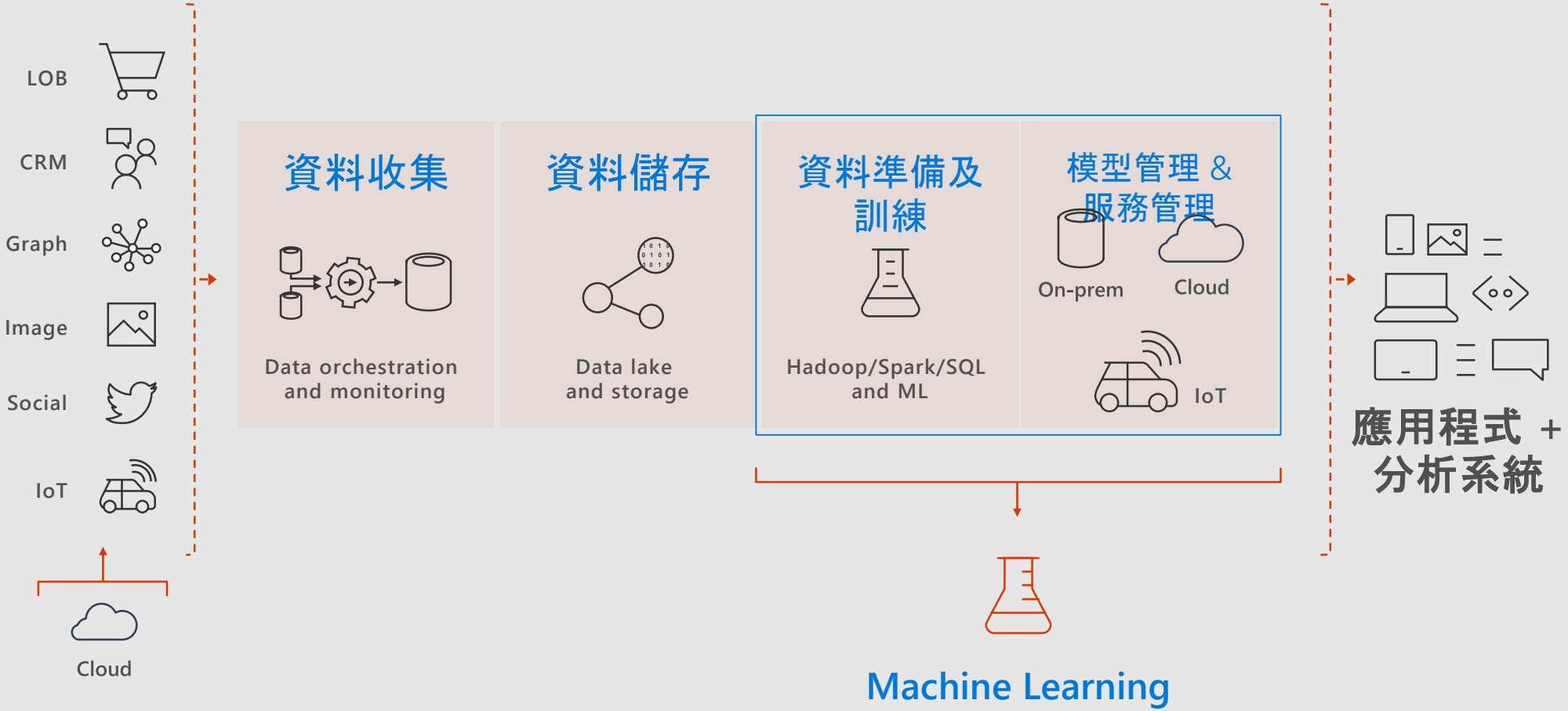
The Impact of Apex Predators



Beschta, Robert L., and William J. Ripple. "Riparian vegetation recovery in Yellowstone: The first two decades after wolf reintroduction." *Biological Conservation* 198 (2016): 93-103.


Snow
Leopard
Trust
https://www.snowleopard.org/

ML/AI 系統開發生命週期



VoTT 功能

可以標註整個目錄內或是單一檔案的影像.

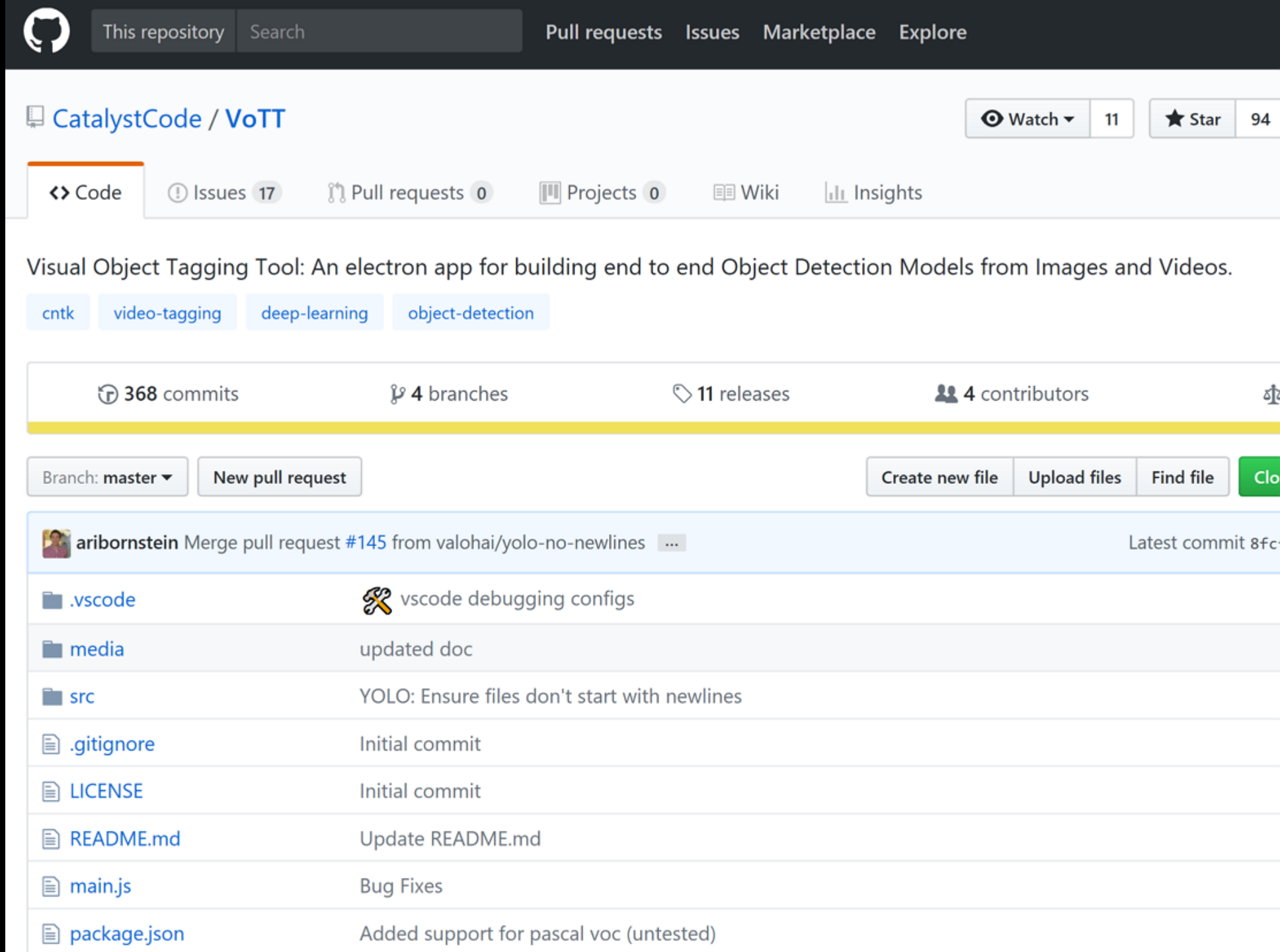
可透過Camshift tracking algorithm 透過電腦協助標註物件

匯出Tag 資料到CNTK, Tensorflow(PascalVOC), YOLO 格式來訓練物件辨識模型

可在新的影像檔中透過執行CNTK 物件辨識模型來驗證模型有效性

http://aka.ms/vott

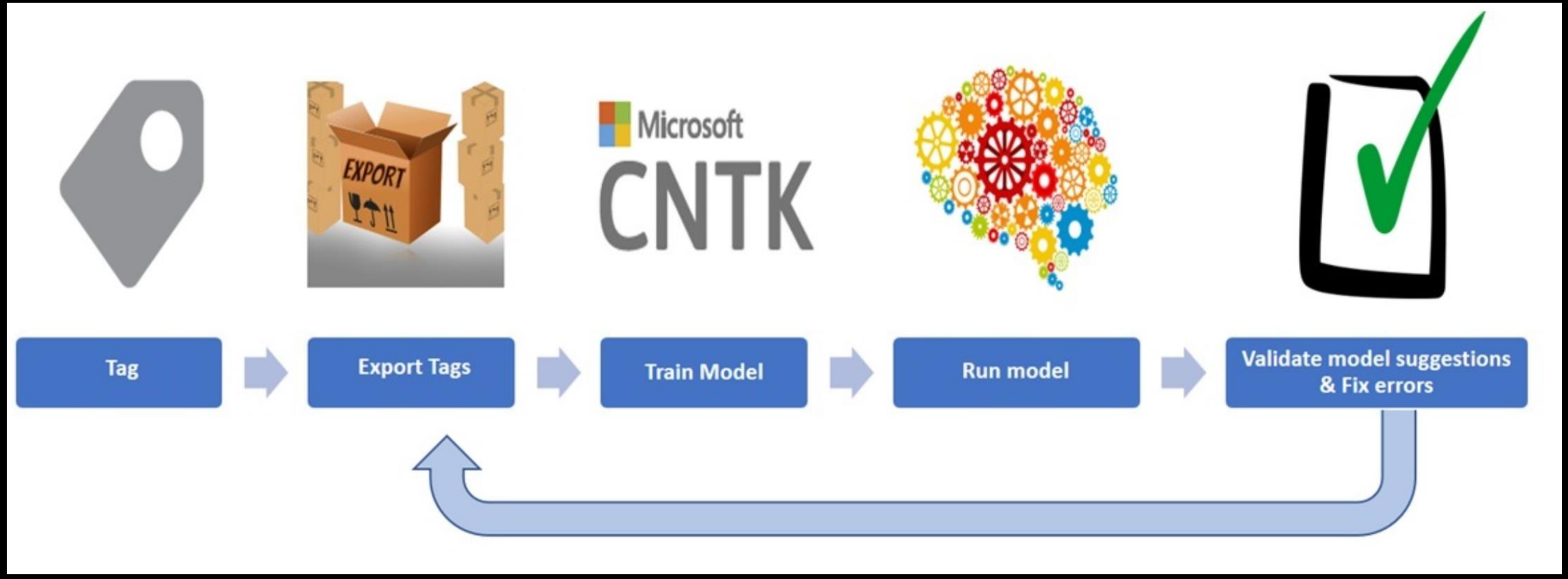
https://github.com/CatalystCode/VoTT



VoTT Demo


圈出要標示
區域
1
X

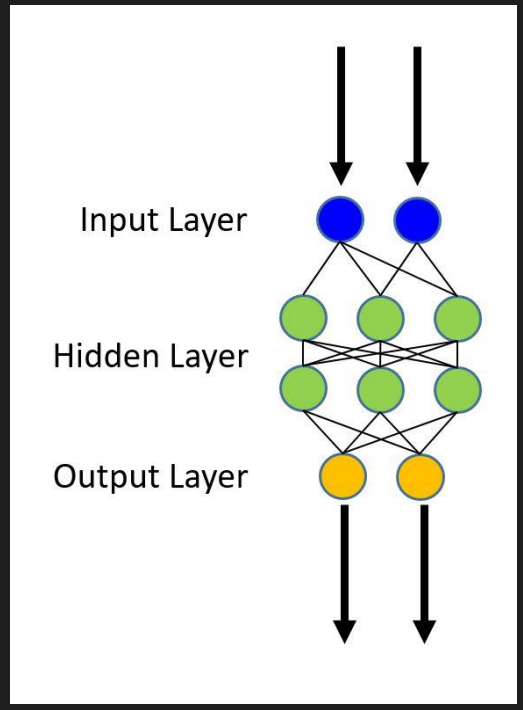

2016-07-15 12:00:00
50°F
H1


EXPORT
Microsoft
CNTK
Tag
Export Tags
Train Model
Run model
Validate model suggestions & Fix errors

Microsoft Cognitive Toolkit

- 執行微軟內部 80% Microsoft 深度學習工具
- 1st-class on **Linux** and **Windows**, **docker** support
- Training: Python, C++,
- Evaluation: C#, Java, Scale up evaluation in Spark
- New in GA:
  - Keras backend support (Beta)
  - Java support, Spark support
  - Model compression (Fast binarized evaluation)

```
def simple_mnist(tensorboard_logdir=None):
    input_dim = 784
    num_output_classes = 10
    num_hidden_layers = 1
    hidden_layers_dim = 200

    # Input variables denoting the features and label data
    feature = C.input_variable(input_dim, np.float32)
    label = C.input_variable(num_output_classes, np.float32)

    # Instantiate the feedforward classification model
    scaled_input = element_times(constant(0.00390625), feature)

    z = Sequential([For(range(num_hidden_layers), lambda i: Dense(hidden_layers_dim, activation=relu)),
                    Dense(num_output_classes)])(scaled_input)

    ce = cross_entropy_with_softmax(z, label)
    pe = classification_error(z, label)

    data_dir = os.path.join(abs_path, "..", "..", "..", "DataSets", "MNIST")

    path = os.path.normpath(os.path.join(data_dir, "Train-28x28_cntk_text.txt"))
    check_path(path)

    reader_train = create_reader(path, True, input_dim, num_output_classes)

    input_map = {
        feature  : reader_train.streams.features,
        label  : reader_train.streams.labels
    }
```



Cognitive Toolkit – The Fastest Toolkit for RNN

http://dlbench.comp.hkbu.edu.hk/
Benchmarking by HKBU, Version 8
Single Tesla K80 GPU, CUDA: 8.0 CUDNN: v5.1

Caffe: 1.0rc5(39f28e4)
CNTK: 2.0 Beta10(1ae666d)
MXNet: 0.93(32dc3a2)
TensorFlow: 1.0(4ac9c09)
Torch: 7(748f5e3)

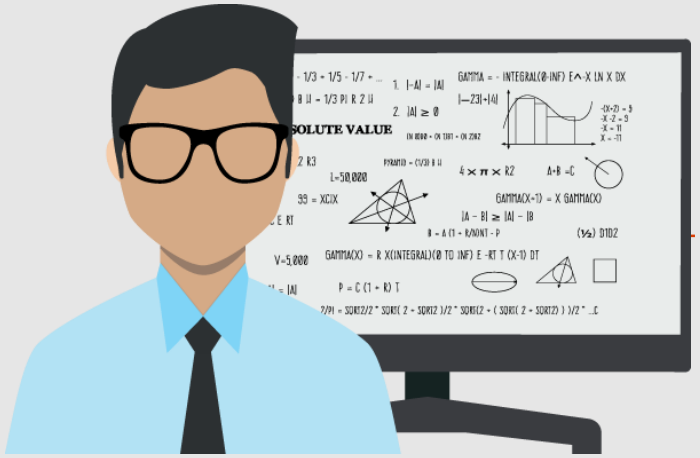
	Caffe	Cognitive Toolkit	MxNet	TensorFlow	Torch
FCN5 (1024)	55.329ms	**51.038ms**	60.448ms	62.044ms	52.154ms
AlexNet (256)	36.815ms	**27.215ms**	28.994ms	103.960ms	37.462ms
ResNet (32)	143.987ms	**81.470ms**	84.545ms	181.404ms	90.935ms
LSTM (256) (v7 benchmark)	-	**43.581ms (44.917ms)**	288.142ms (284.898ms)	- (223.547ms)	1130.606ms (906.958ms)

3.5 times faster then TF

2.2 times faster then TF

5 times faster then TF

對機器學習運算的需求



訓練 & 佈署

Cloud

- Spark
- RDBS
- Virtual machines
- GPUs
- Container services

ON-PREMISES

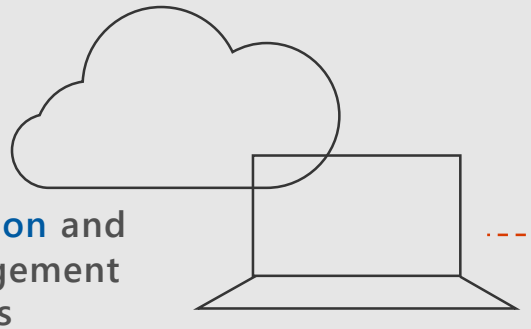
- RDBS
- Machine Learning Server
- FPGA

EDGE

- Azure IoT Edge

對機器學習運算的需求

訓練 & 佈署



Experimentation and **Model** Management Services

Notebooks
IDEs
Azure Machine Learning Workbench
VS Code Tools for AI

Cloud

- Spark
- SQL Server
- Virtual machines
- GPUs
- Container services

ON-PREMISES

- SQL Server
- Machine Learning Server
- FPGA

EDGE

- Azure IoT Edge

Conda_dependencies.yml

```
# Conda environment specification. The de
# automatically provisioned for runs agai

# Details about the Conda environment fil
# https://conda.io/docs/using/envs.html#u

# For Spark packages and configuration, s

name: project_environment
dependencies:
  - python=3.5.2
  - scikit-learn
  - pip:
    - notebook
    - matplotlib
```

Docker Hub

Azure ML Base Docker Image

User's python script (foo.py)

my_remote_vm.compute

```
address: 13.1.1.1
baseDockerImage: microsoft/mm
nativeSharedDirectory: ~/.azu
password: AzureMlSecret=my_re
sharedVolumes: true
type: remotedocker
username: usr
```

my_remote_vm.runconfig

```
ArgumentVector:
- $file
CondaDependenciesFile: aml_config/
EnvironmentVariables: null
Framework: PySpark
PrepareEnvironment: true
SparkDependenciesFile: aml_config/
Target: my_remote_vm
TrackedRun: true
```

az ml experiment submit –c my_remote_vm foo.py

Azure ML Execution Service

Execute script

Docker image definition

Remote VM

SSH

Docker Engine

User's program in Docker container

Other containers

Run results and artifacts

Azure ML Run History Service

Runs
Metrics
Artifacts
Status

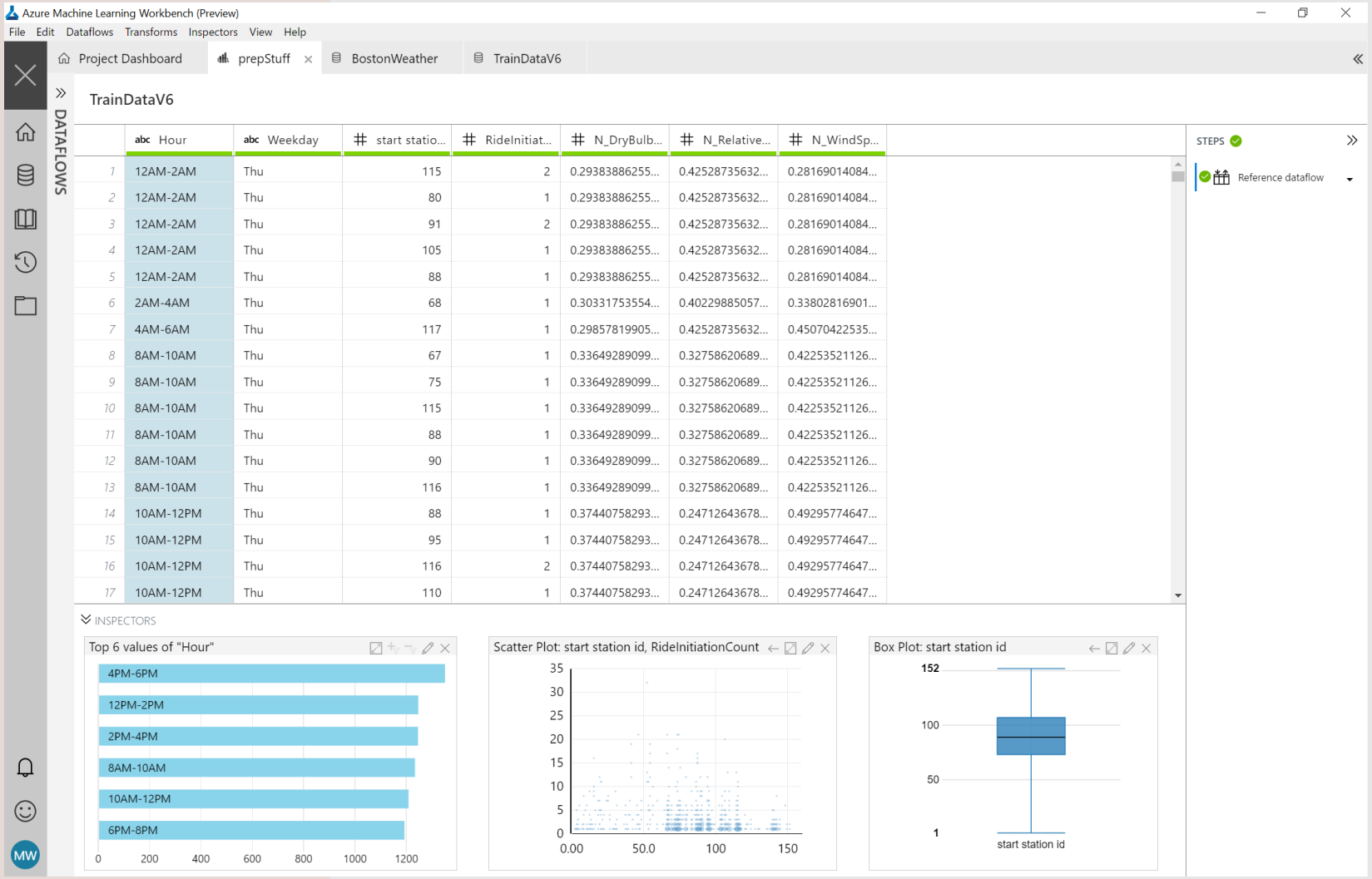
AI Powered Data Wrangling

Rapidly sample, understand, and prep data

Leverage PROSE and more for intelligent, data prep by example

Extend/customize transforms and featurization through Python

Generate Python and Pyspark for execution at scale



Workbench Demo

實驗服務
(Experimentation service)

Manage project dependencies

Manage training jobs locally, scaled-up or scaled-out

Git based checkpointing and version control

Service side capture of run metrics, output logs and models

Use your favorite IDE, and any framework

USE ANY FRAMEWORK OR LIBRARY

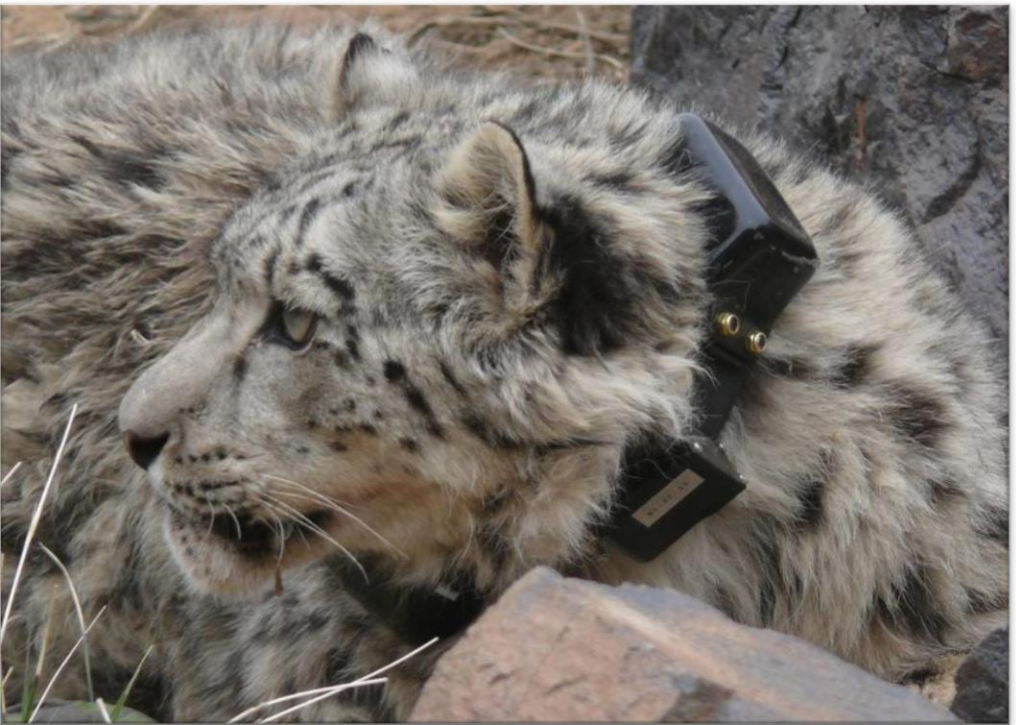
USE ANY TOOL

USE THE MOST POPULAR INNOVATIONS


Snow
Leopard
Trust
https://www.snowleopard.org/

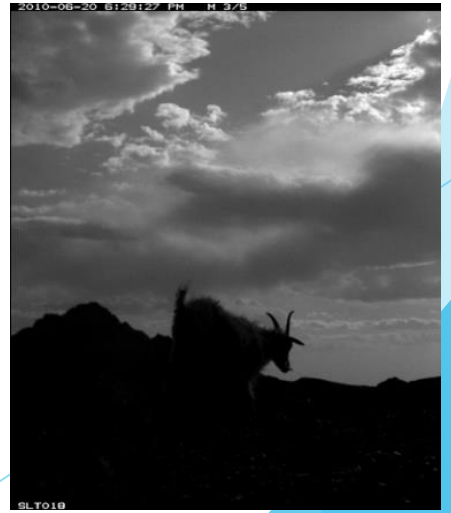
Gathering Leopard Data

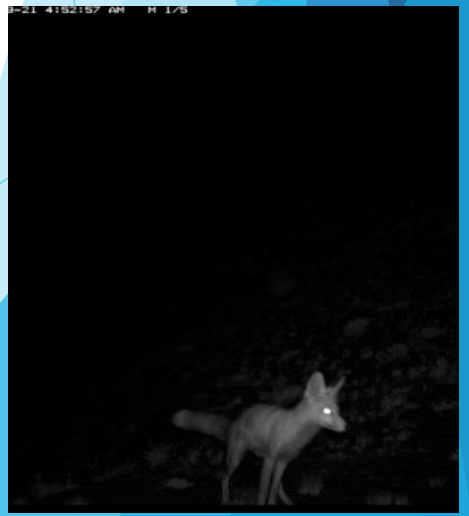
- 42 camera traps over 1,700 sq km
- ~1.3 mil images
- 23 leopards collared in 9 years

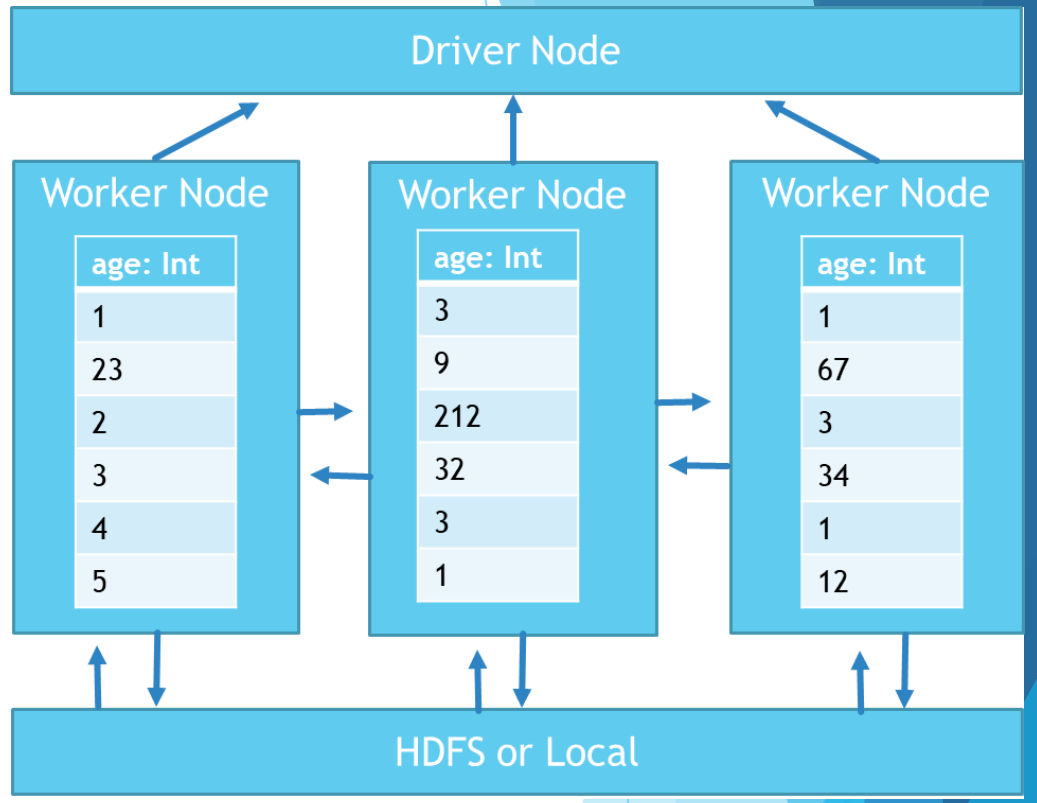


Camera Trap Images

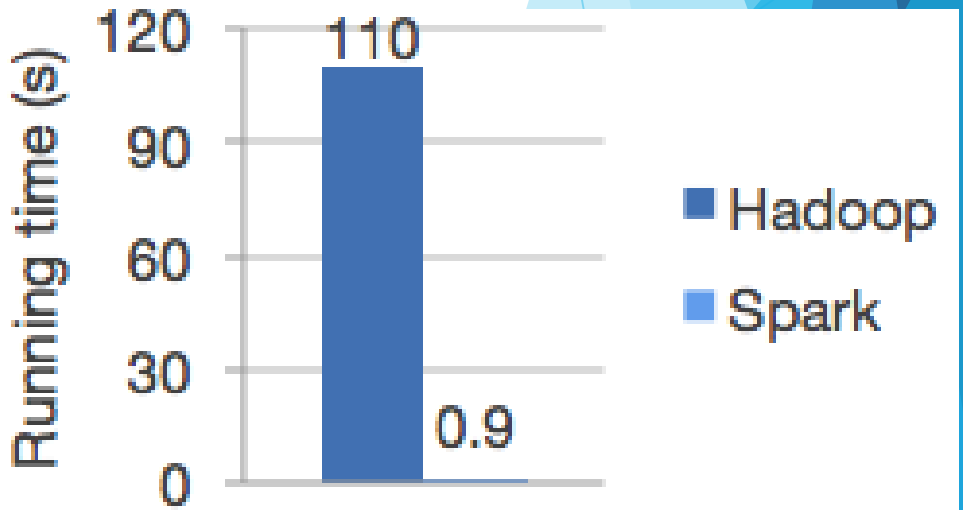
Manually classifying 20k images took 300 hours → 1.3 million will take 19,500 hours







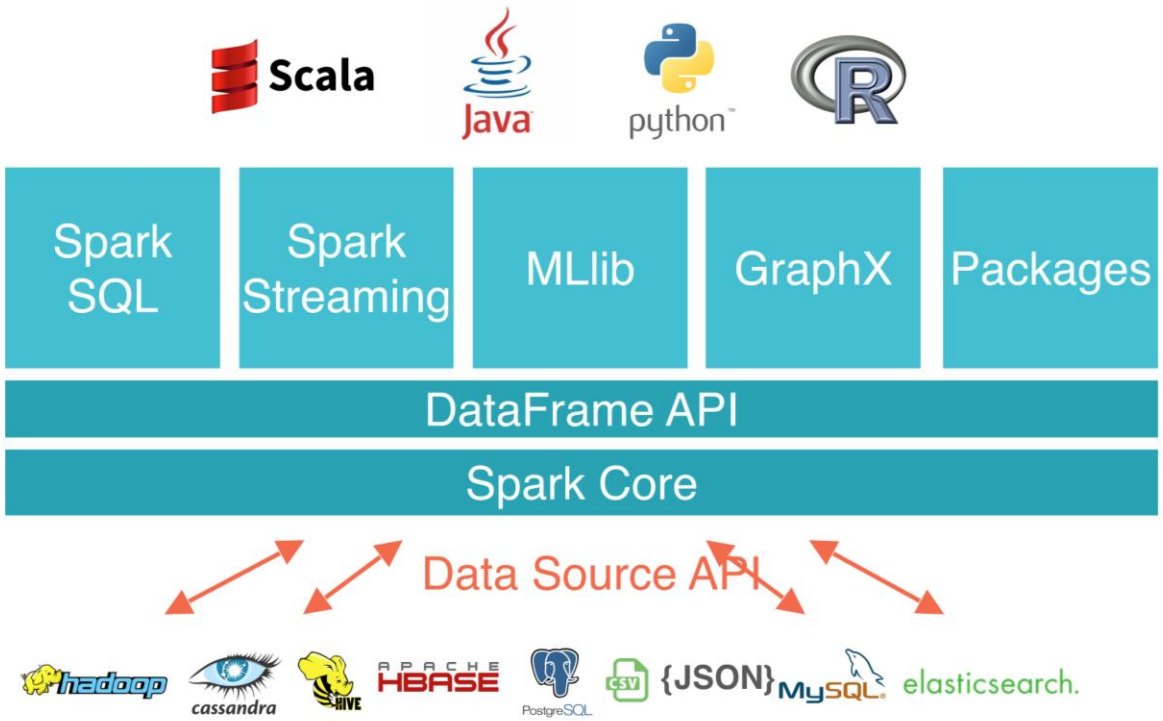
- A **fault-tolerant distributed** computing framework
- Generalizes, optimizes, and combines Map-Reduce and SQL style-operations
- Scales to thousands of machines
- Functional API
- Built in Scala, but has bindings in Python, R, Java, C#, F#
- Has a flourishing community
  - SparkML
  - GraphX
  - Structured Streaming



Spark ML: Machine Learning at Scale

- High level library for machine learning
- Similar abstraction to SciKit Learn (But waaaaay better API)
- Written in Scala but has wrappers in Python
- Models all have a uniform interface: PipelineStage
  - Classification: logistic regression, naive Bayes, ...
  - Regression: generalized linear regression, survival regression, ...
  - Decision trees, random forests, and gradient-boosted trees
  - Recommendation: alternating least squares (ALS)
  - Clustering: K-means, Gaussian mixtures (GMMs), ...
  - Topic modeling: Latent Dirichlet Aallocation (LDA)

```
data = spark.read.csv("hdfs://...")
train, test = data.randomSplit([.5,.5])
model = LogisticRegression().fit(train)
predictions = model.transform(test)
```



Spark/SparkML

- Large scale parallelism
- Fault tolerance
- Auto-scaling/ elastic
- High throughput streaming
- Operates in Scala (On the JVM) with bindings

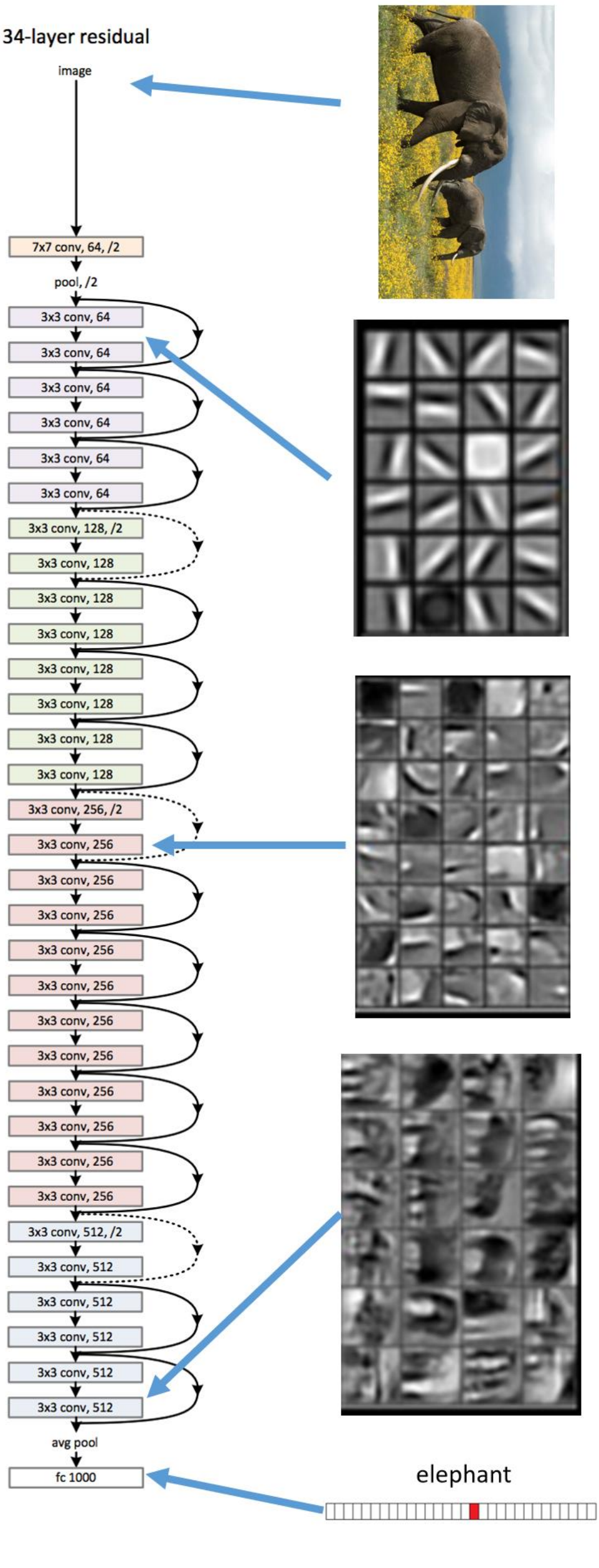
CNTK

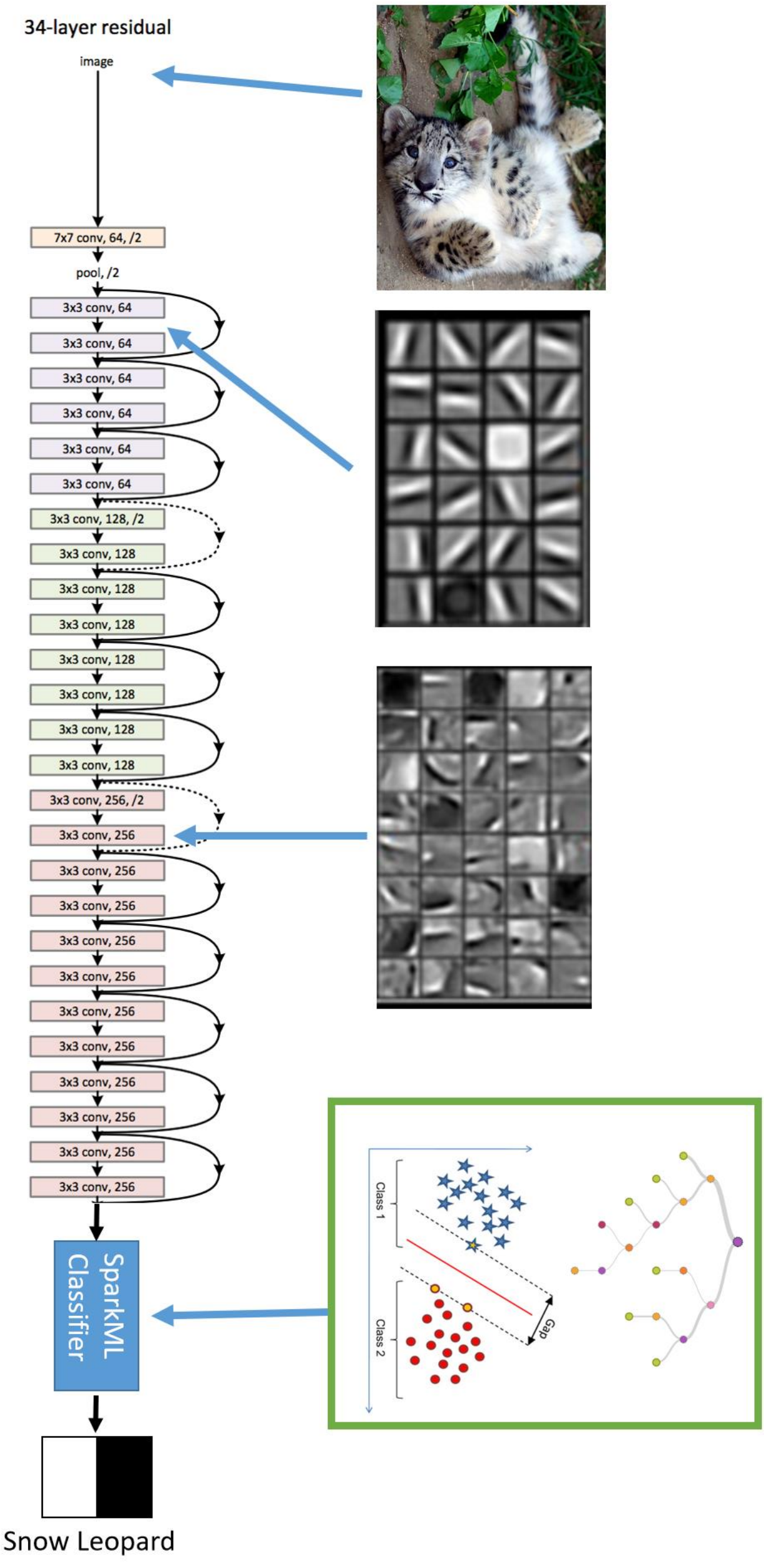
- High speed GPU/CPU computations
- Automated gradient calculations
- Flexible language for defining cutting edge models
- Operates in C++ with bindings

Microsoft Machine Learning For Apache Spark

Distributed Deep Learning, Image Analysis, Text Analytics, and Much More

https://aka.ms/mmlspark

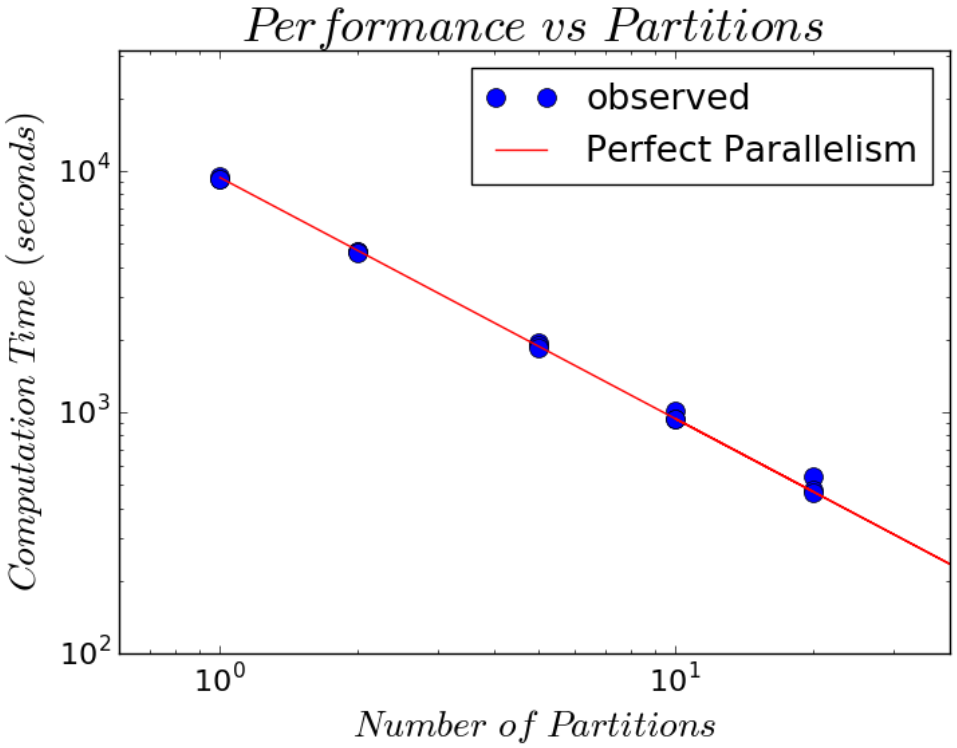

34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
elephant


34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
SparkML Classifier
Class 1
Class 2
Gap
Snow Leopard

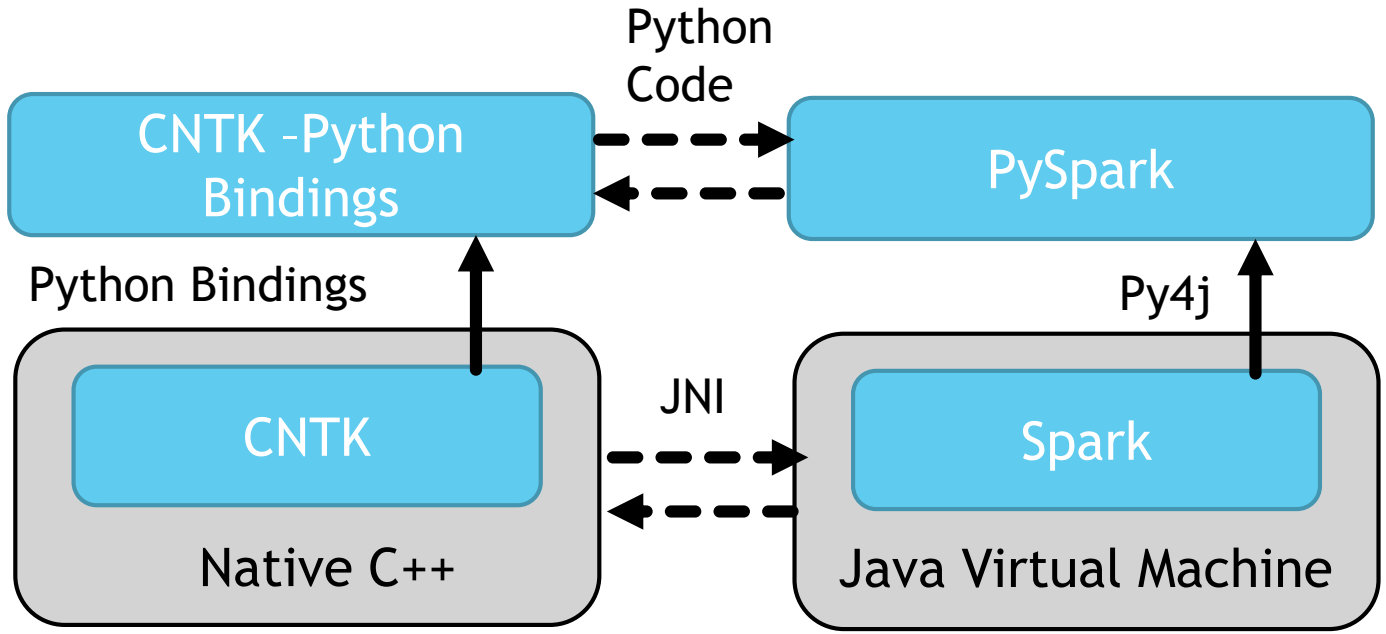
Cognitive Toolkit on Spark

- We have created a SparkML transformer for evaluating ***any*** CNTK computation graph on a spark cluster
- We take care of
  - Distributing + loading native libraries
  - Distributing model to all workers
  - Feeding the network using Spark's data
  - Minibatching for computational efficiency
- Performance scales near perfectly with Nodes
- Integrates naturally with SparkML + Spark Streaming ecosystem

```
1  val model = new CNTKModel()
2     .setModelLocation(session, modelPath)
3     .setInputCol("images")
4     .setOutputCol("features")
5     .setOutputNodeName("z")
6
7  val result = model.transform(data)
```



Unifying CNTK and Spark: What's under the hood



- Can either integrate at the python level
- Or at the deeper Java level
- Java level allows CNTK to target all spark bindings
- Direct Java integration means less data transfer

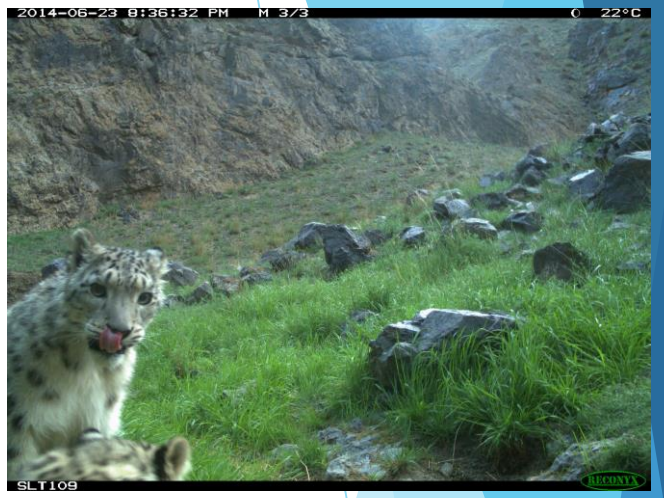
Cognitive Toolkit's Java Bindings

- Cognitive Toolkit is written in C++ but has bindings in Python, and C#
- We use the Simple Wrapper and Interface Generator (SWIG) to expose CNTK's Evaluation library to Java
- All Java bindings are machine generated so they require very little maintenance!
- Try them out in Cognitive Toolkit's > 2.0

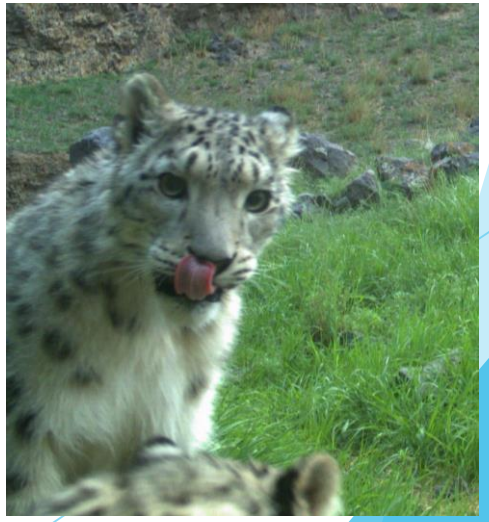
```
Function modelFunc = Function.load(new File("resnet20_cifar10_python.dnn"), device);
Variable outputVar = modelFunc.getOutputs().get(0);
Variable inputVar = modelFunc.getArguments().get(0);
```

Parallel Image Processing with OpenCV

- DNNs are often picky about their input data shape and normalization.
- We provide bindings to OpenCV image processing operations, exposed as SparkML PipelineStages:

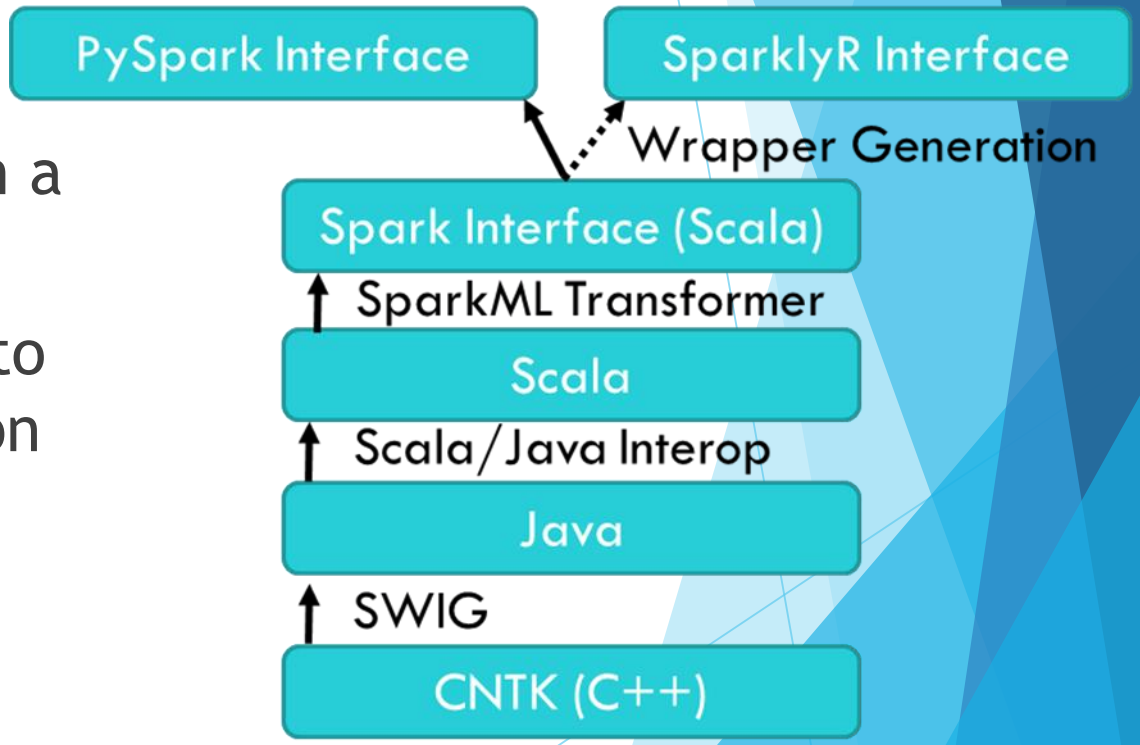



```
tr = ImageTransform().setOutputCol("transformed")
     .resize(height = 200, width = 200)
     .crop(0, 0, height = 180, width = 180)


smallImages =
tr.transform(images).select("transformed")
```



PySpark Bindings…For Free!

- Our core code is written in Scala
- Python is too hot to forget
- Spark has a exposed bindings to python in a package called PySpark
- We automatically expose all of our work to python through generating SparkML python APIs
- Now also support SparklyR, Spark's R bindings



Key Steps in Snow Leopard Analysis

- Prepare pipeline:

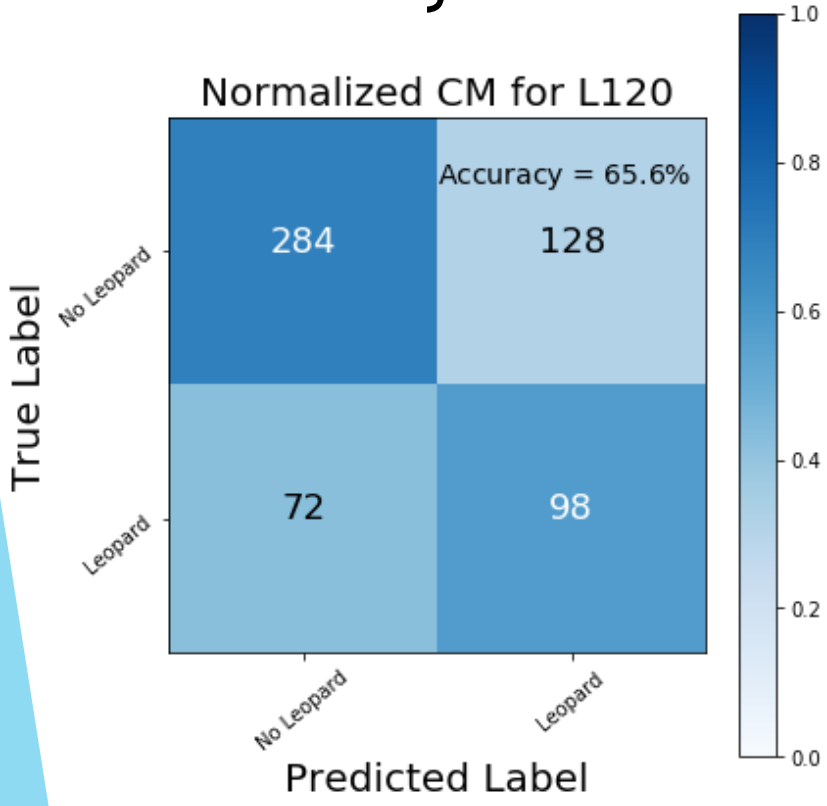
```
featurizer = mml.ImageFeaturizer(
    inputCol="image",
    outputCol="features",
    layerNames=localModel.layerNames,
    cutOutputLayers=2,
    miniBatchSize=100)\
    .setModelLocation(spark, localModel.uri)

classifier =  LogisticRegression(
        featuresCol = "features",
        labelCol = "labels")
pipe = Pipeline(stages=[featurizer,classifier])
```

- Fit and predict:

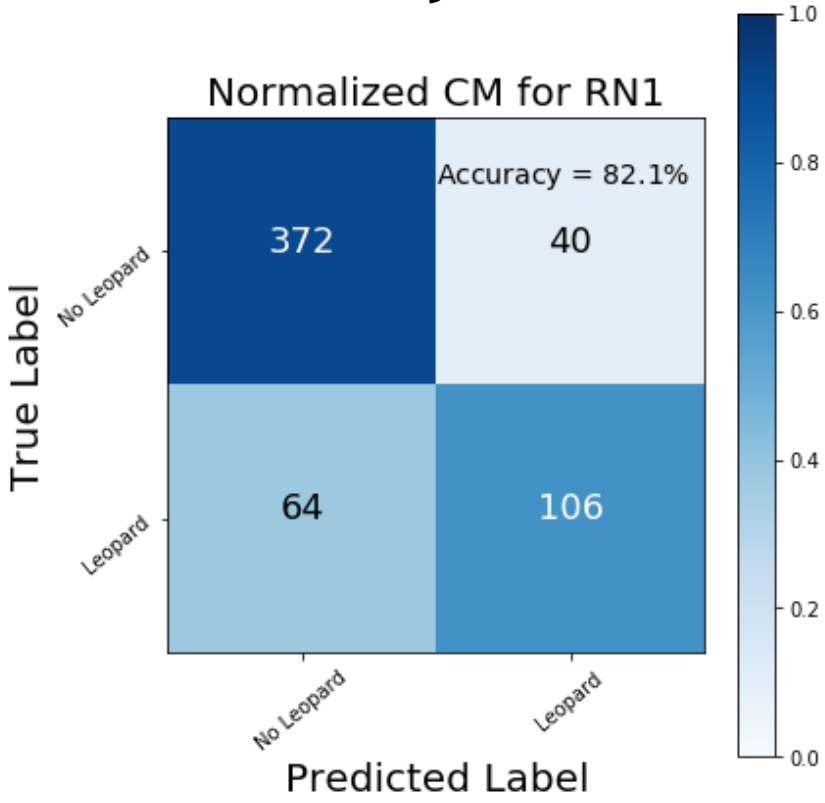
```
predictions = pipe.fit(train).transform(test)
```

Performance

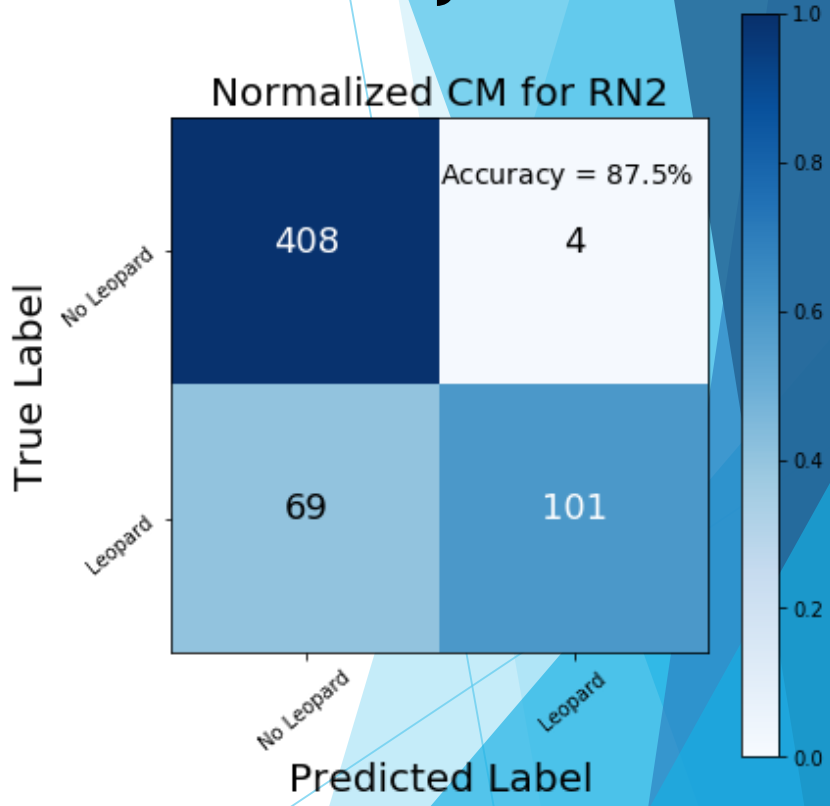
Accuracy 65.6%



Accuracy 82.1%



Accuracy 87.5%



Without Deep Featurization

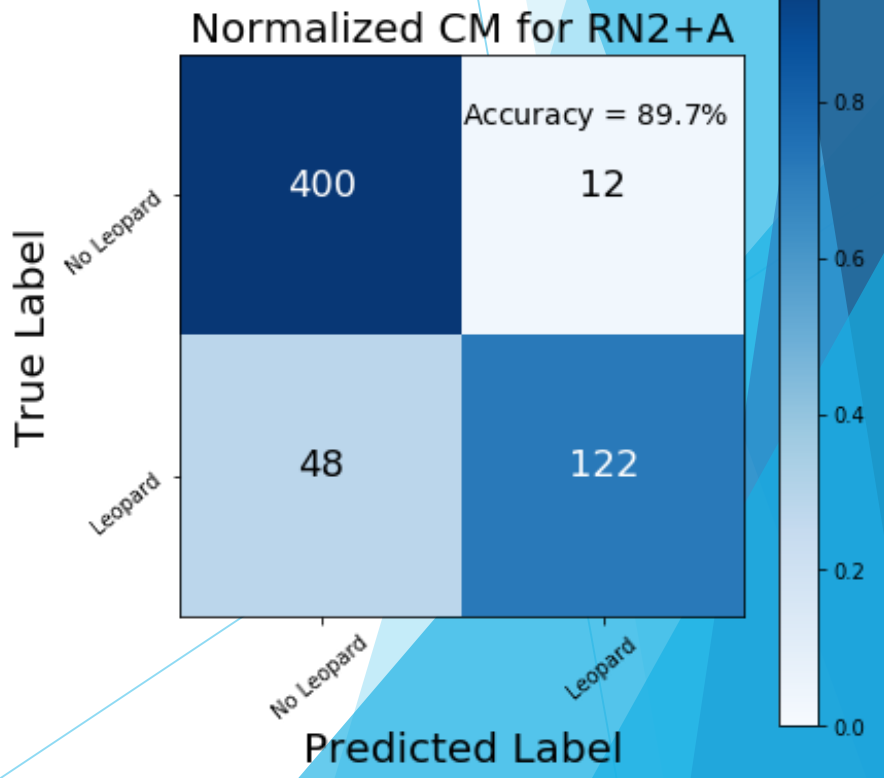
With Deep Featurization

Further Refinements

- Dataset Augmentation

```
ia = mml.ImageSetAugmenter(inputCol="images", outputCol="images")
pipe = Pipeline(stages=[ia, featurizer,classifier])
```

Accuracy 89.7%



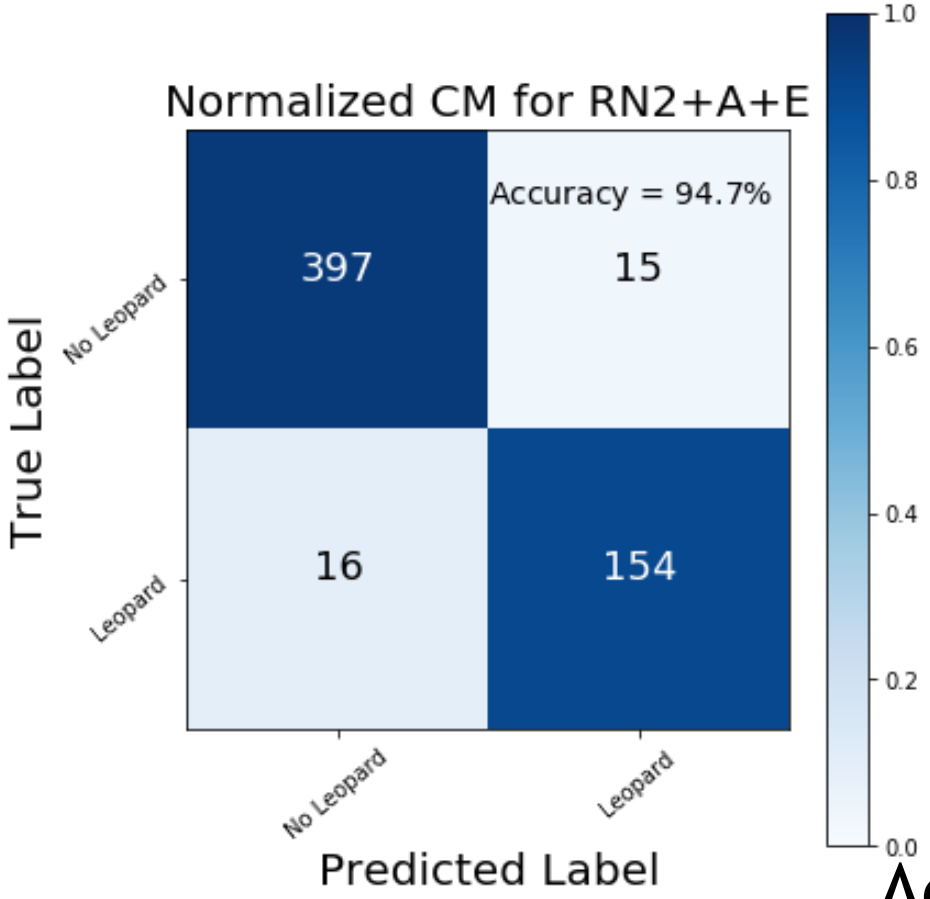

2012-06-30 9:03:10 PM M 1/5
24°C
RECONYX
SLT017

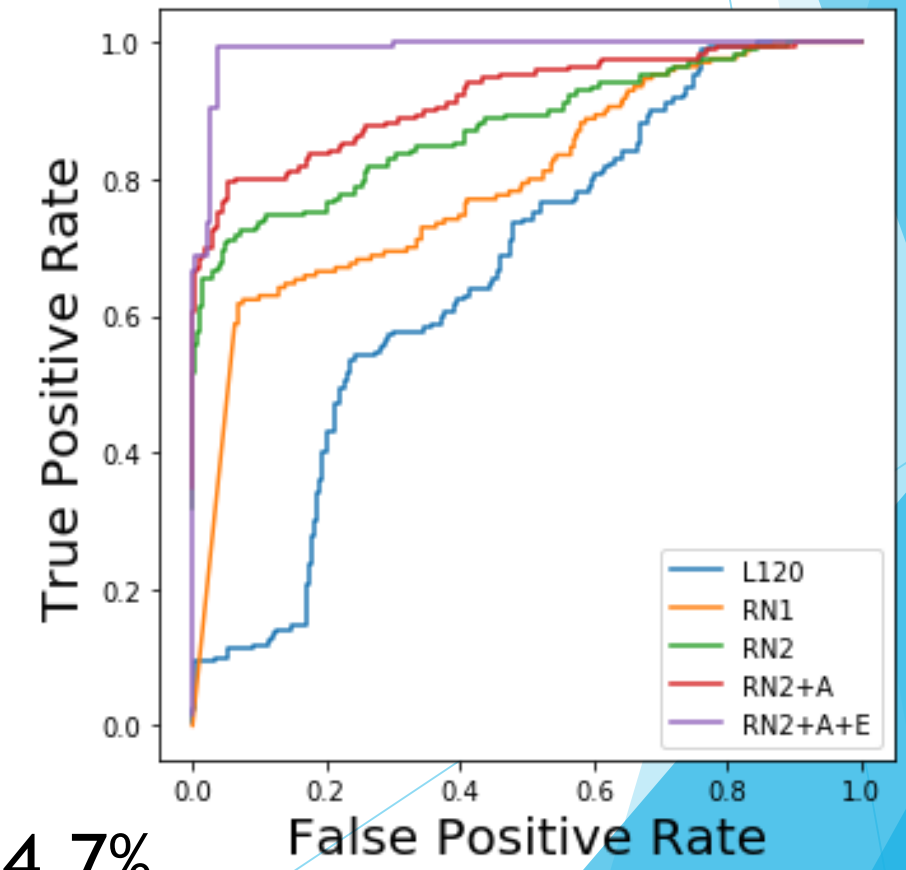

2012-06-30 9:03:11 PM M 2/5
24°C
RECONYX
SLT017


2012-06-30 9:03:12 PM M 3/5
24°C
RECONYX
SLT017

Ensembling over Sequences

```
ebk = mml.EnsembleByKey(
    keys=["group", "filename"],
    cols=["prob"],
    colNames=["mean(prob)"])
pipe = Pipeline(stages=[ia, featurizer, classifier, ebk])
```

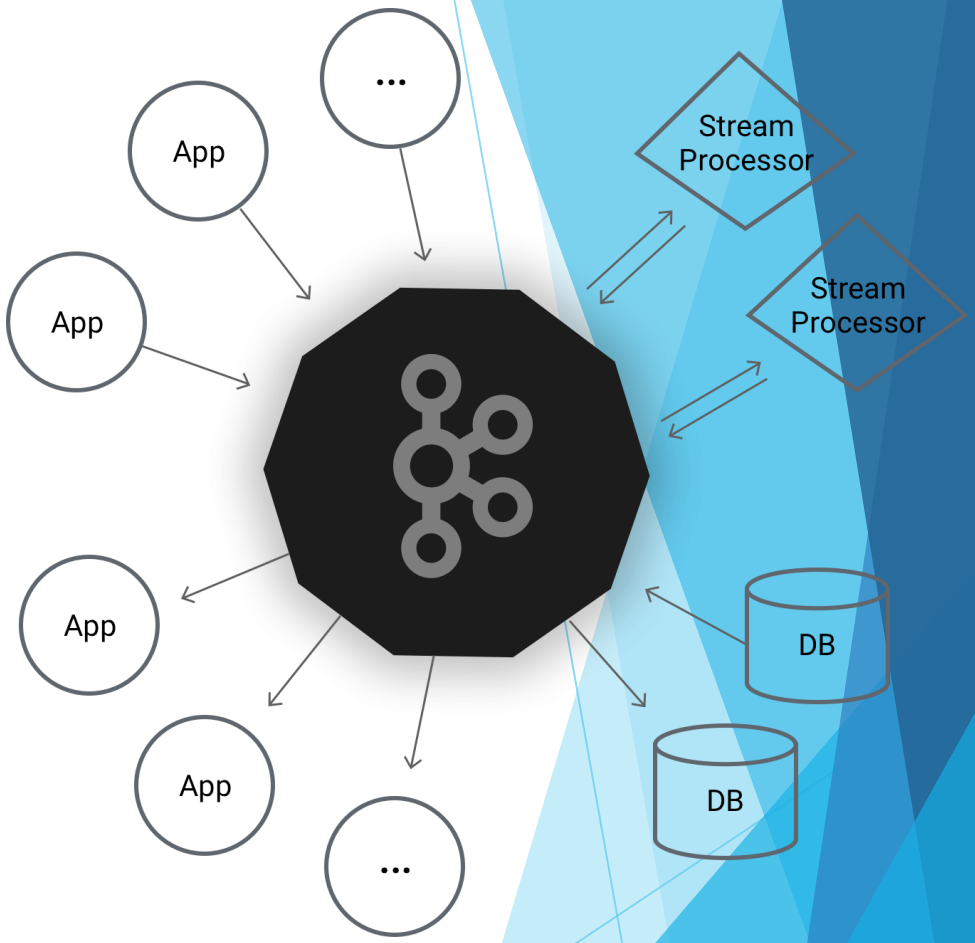


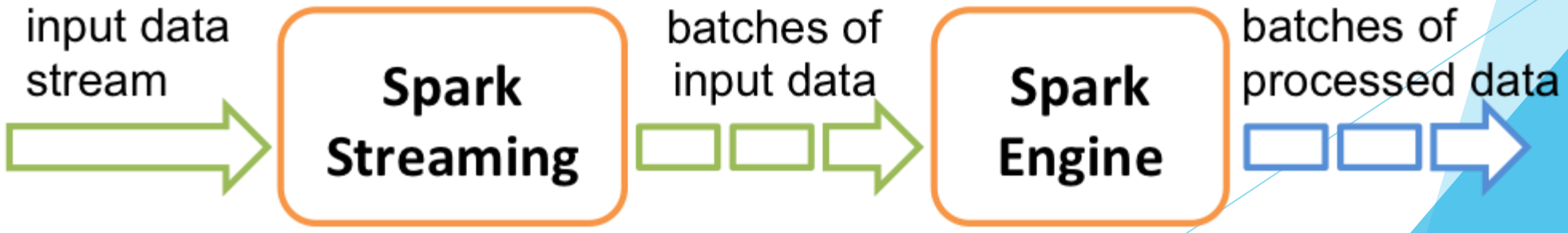


Accuracy 94.7%

Moving to Production

- Want to deploy trained models to large-scale production
- Spark Streaming:
  - high-throughput, fault-tolerant, parallel streaming.
  - Uses the same API as batch
  - Integrates tightly with Apache Kafka and



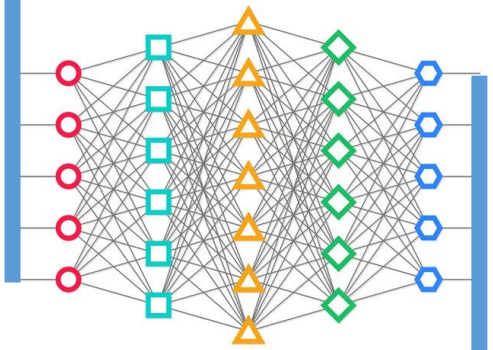
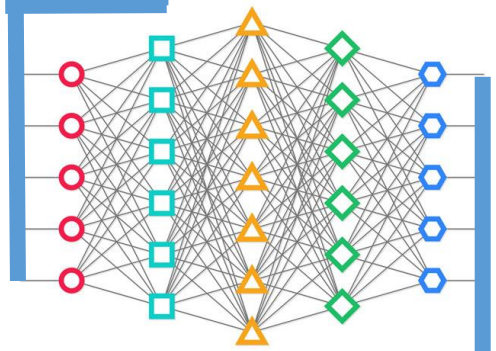





Time

1 2 3 4 5 6

CNTK Pipeline

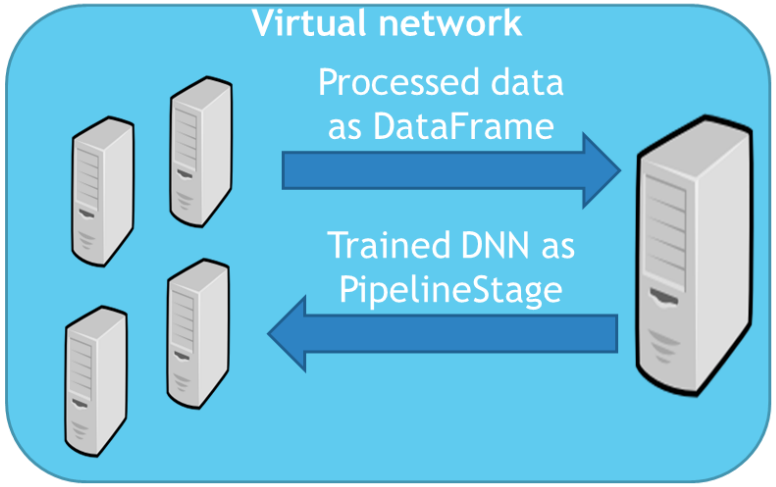
Time

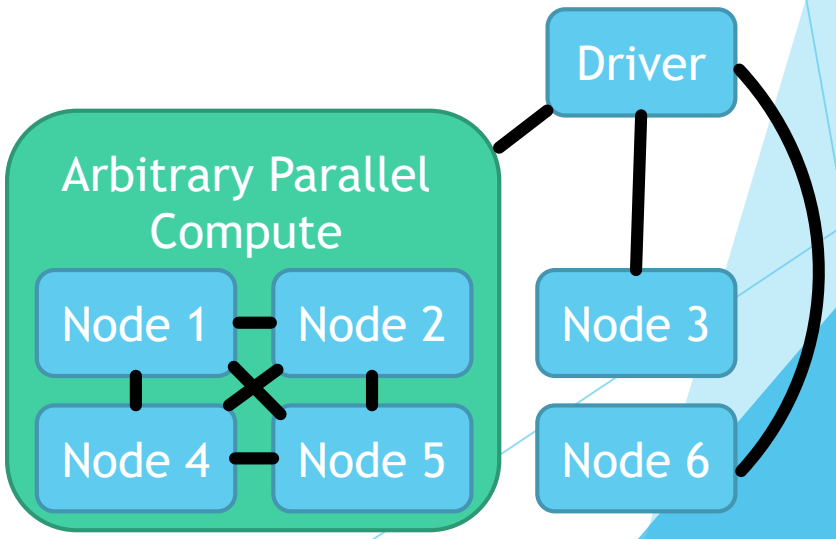
1 2 3 4 5 6

Demo

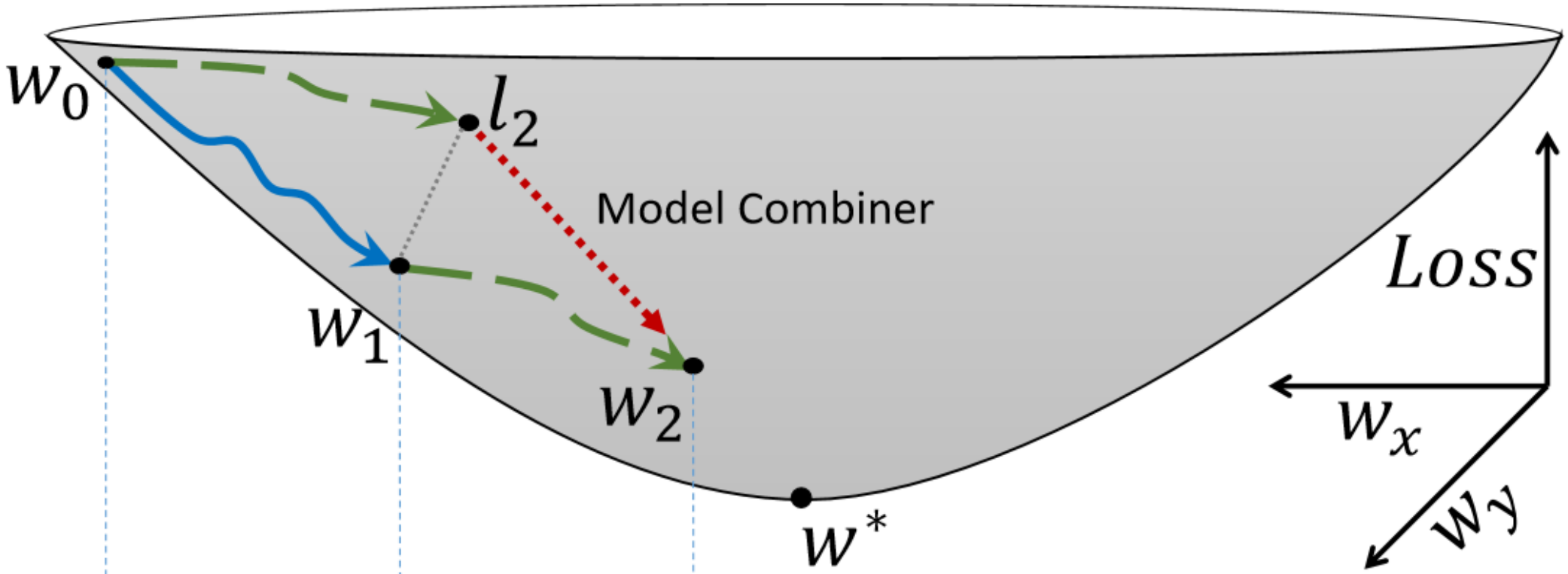
Training

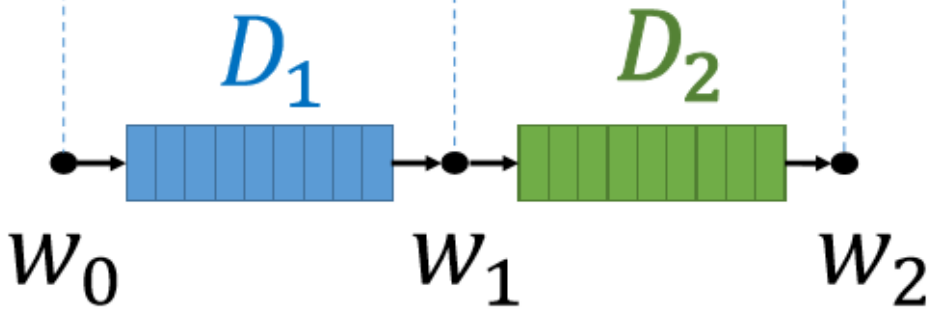
- https://github.com/Azure/mmlspark/tree/gpu
- GPU can give a 10x speedup for network training and eval
- Tether a GPU cluster to spark cluster
- Long Term Goal: Hook into Spark resource manager to request resources for custom jobs: CNTK, LightGBM, etc





Next Steps: Sym SGD and Parallel Training





Maleki, Saeed, Madanlal Musuvathi, and Todd Mytkowicz. "Parallel Stochastic Gradient Descent with Sound Combiners." *arXiv preprint arXiv:1705.08030* (2017).

How you can help

They need more camera surveys!

- 1,700 sq km surveyed of 1,500,000
- $500 will buy an additional camera
- $2,000 will fund a researcher
- Any amount helps!

www.snowleopard.org

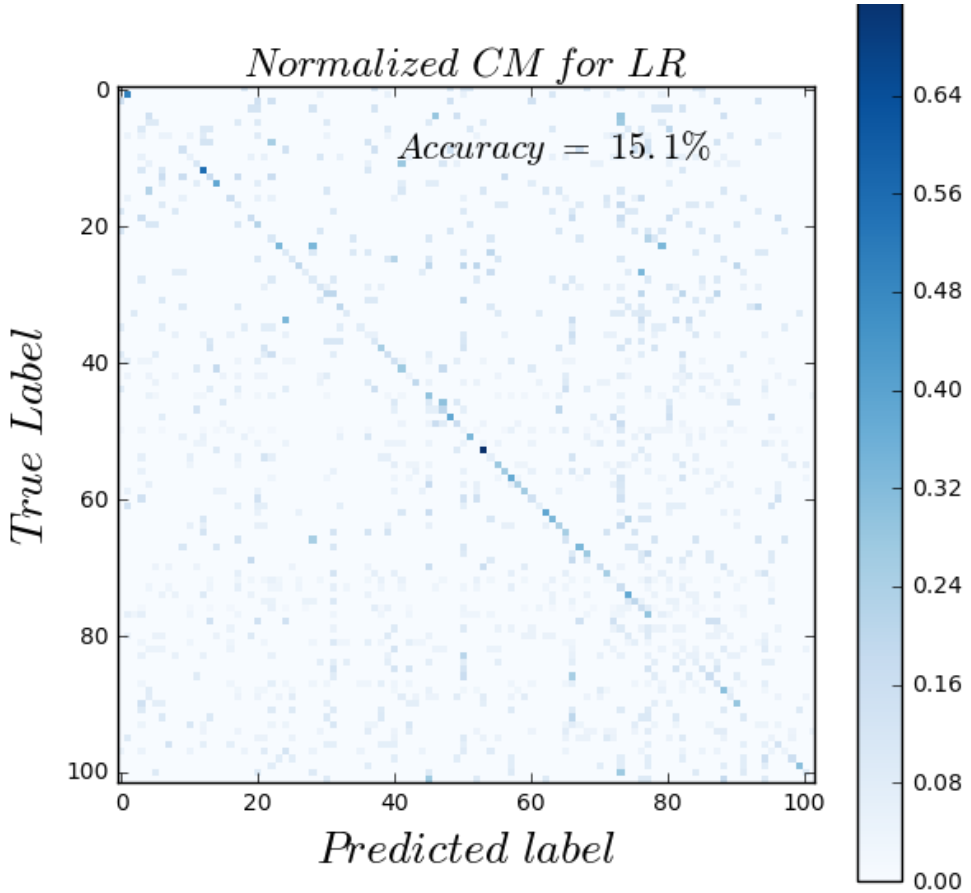
Thanks to

- MMLSpark Team:
  - Akshaya Annavajhala (AK), Danil Kirsanov, Eddie DeLeon, Eli Barzilay, Ilya Matiach, Joe Davison, Maureen Busch, Miruna Oprescu, Ratan Sur, Roope Astala, Sudarshan Raghunathan, Tong Wen, Young Park
- CNTK Team:
  - M. Hillebrand, N. Karampatziakis, W. Manousek, Z. Wang, C. Zhang, Liqun Fun

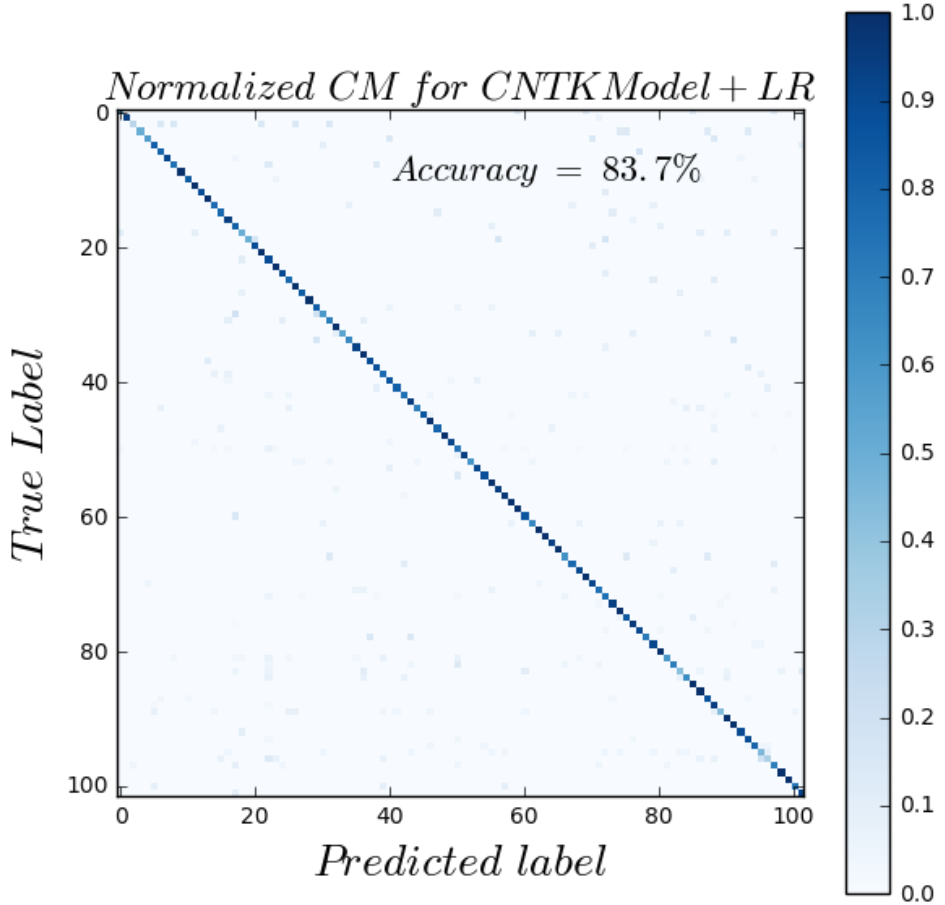
Where to Learn More

- Snow Leopard Blog Post:
  - https://blogs.technet.microsoft.com/machinelearning/2017/06/27/saving-snow-leopards-with-deep-learning-and-computer-vision-on-spark/
- MSJAR Article: “Massively Parallel Neural Networks with CNTK on Spark”
  - **http://aka.ms/msjar**
- Submitting to PMLR 2017: Flexible and Scalable Deep Learning with MMLSpark
- Sample Notebooks
  - https://github.com/mhamilton723/notebooks/blob/master/SnowLeopard.ipynb
  - https://github.com/Azure/mmlspark/blob/master/notebooks/samples/305%20-%20Flowers%20ImageFeaturizer.ipynb

Without Deep Featurization



With Deep Featurization



Home > Machine Learning Experimentation > hermanwumlv2a72a

hermanwumlv2a72a

Machine Learning Experimentation - PREVIEW

New
Dashboard
Resource groups
All resources
Recent
App Services
SQL databases
Virtual machines (class...
Virtual machines
Cloud services (classic)
Subscriptions
IoT Hub
Azure Cosmos DB
HDInsight clusters
Network security groups

Search (Ctrl+/)

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems

APPLICATION SETTINGS

Update seats
Add workspace
Resync Storage Keys

SETTINGS

Locks
Automation script
Properties

Delete
Add workspace
Update seats

Subscription
Microsoft Azure Internal Consumption

Subscription ID
[redacted]

Getting Started

Download Azure Machine Learning Workbench for Windows

Azure Machine Learning Workbench enables you to explore data, build experiments, and compare model performance.

Download Azure Machine Learning Workbench for Mac OS X

Azure Machine Learning Workbench enables you to explore data, build experiments, and compare model performance.

Create an account for Machine Learning Model Management

Machine Learning Model Management accounts allow you to easily deploy your models and manage them in Azure.

Learn how to configure Machine Learning Compute for Model Management

To take advantage of deploying and managing your models in production, you must first configure the environment where the models will be deployed. Learn how to set up your environment to deploy and manage your models in Azure.