Massively Parallel Deep Learning with MMLSpark

Mark Hamilton
marhamil@microsoft.com

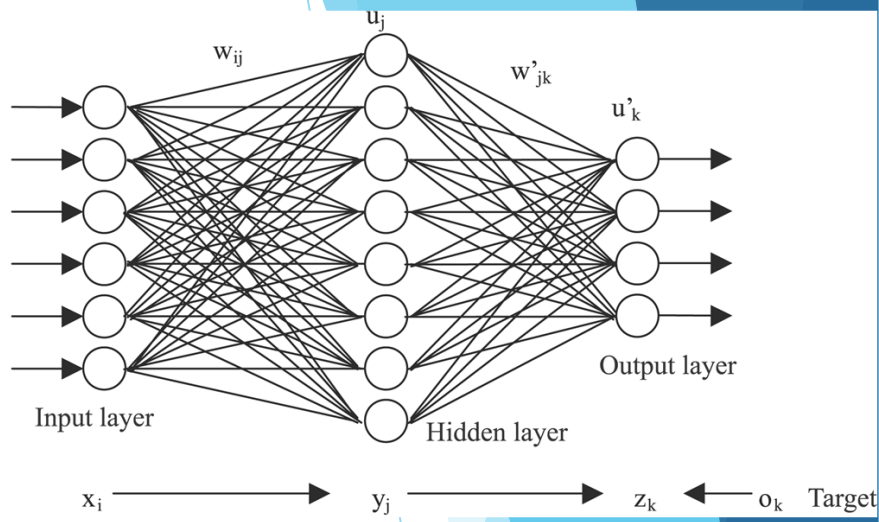
MMLSpark
aka.ms/mmlspark

Agenda

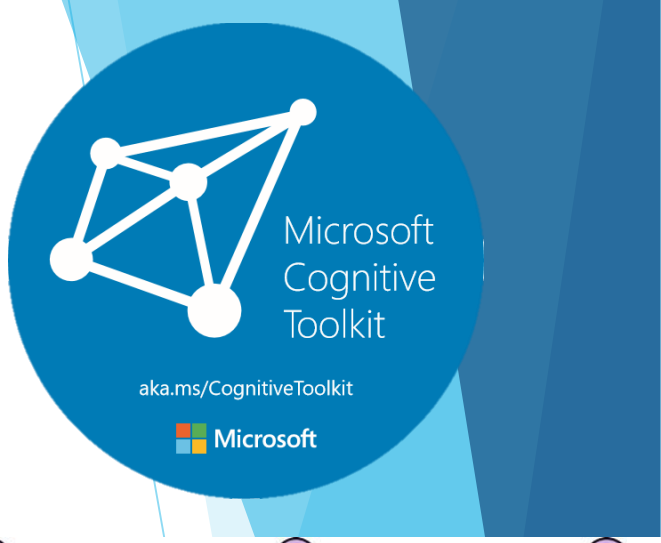
- Background
  - Deep Learning
  - Cognitive Toolkit (CNTK)
  - Apache Spark and SparkML
- MMLSpark
  - Integrating Cognitive Toolkit and Spark
  - OpenCV Spark integration
  - PySpark Wrapper Generation
- Snow Leopard Conservation
  - Transfer Learning
  - MMLSpark Serving
- Future Work
  - Local Interpretable Model Agnostic Explanations

Deep Learning

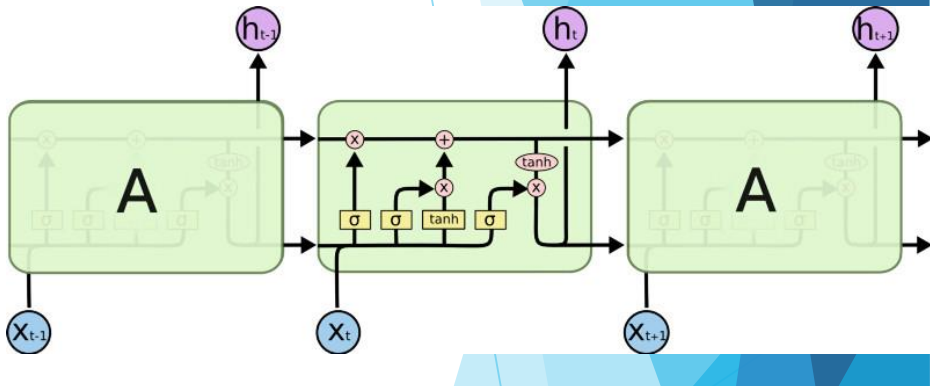
- Originally referred to learning neural networks with many layers
- Now refers to any suitably complicated statistical model trained with gradient descent
- Has become a recent favorite because:
  - Spectacular performance in many domains
  - Quick training w/ gradient descent
  - Low memory footprint w/ Stochastic Gradient Descent (SGD)
  - Large space of possible model architectures
  - Automatic differentiation software and APIs

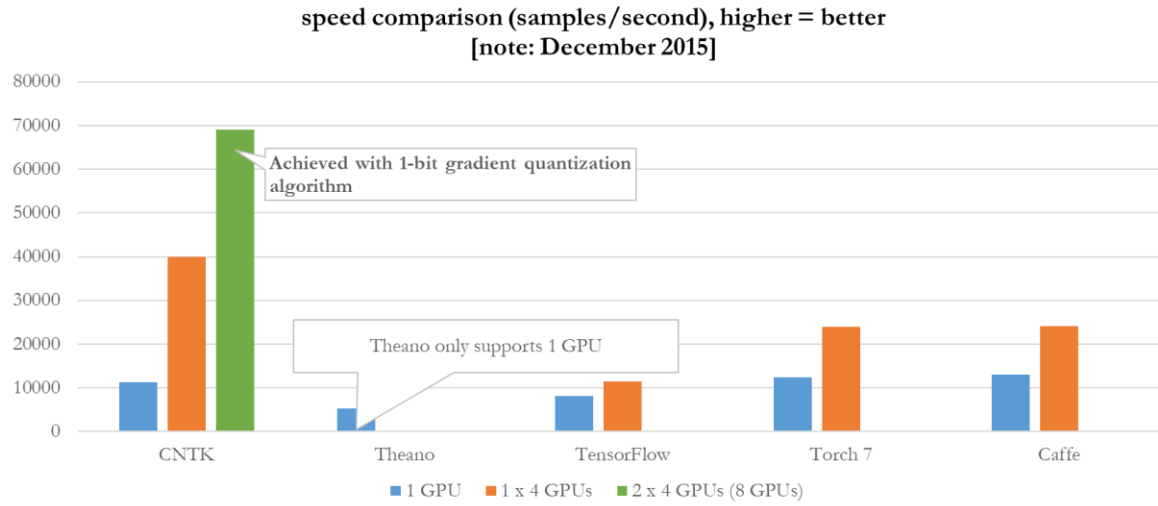


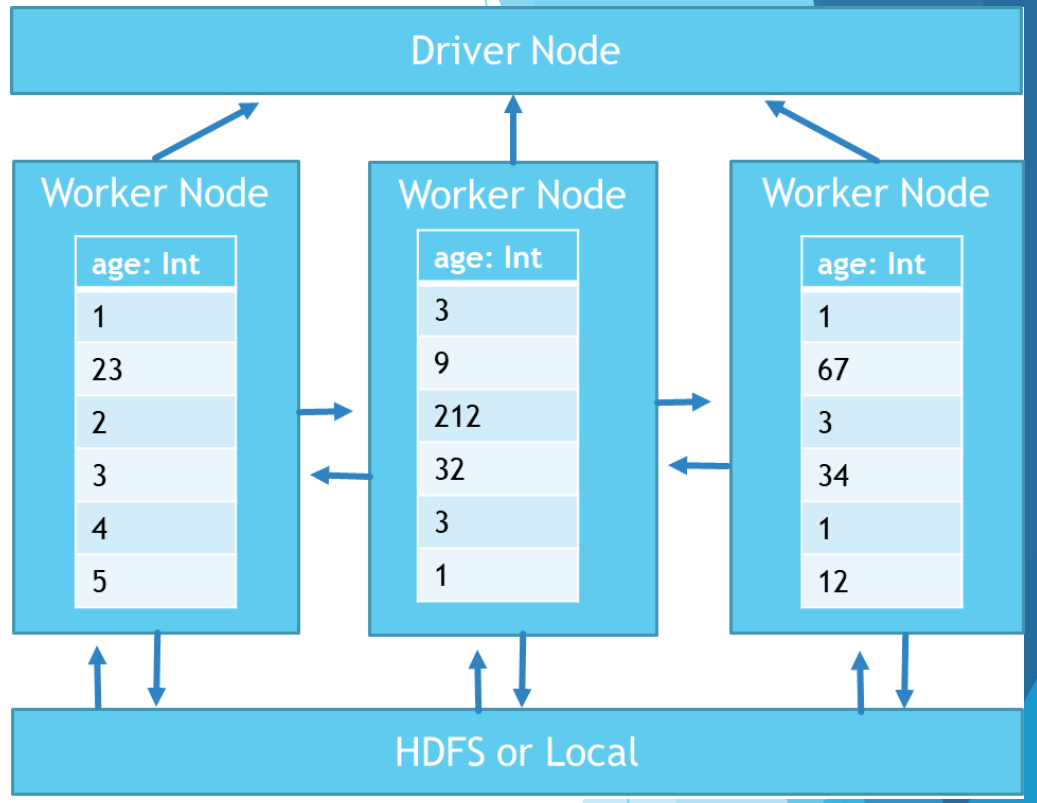
Microsoft Cognitive Toolkit (formerly CNTK) (Dong Yu et al. 2012)



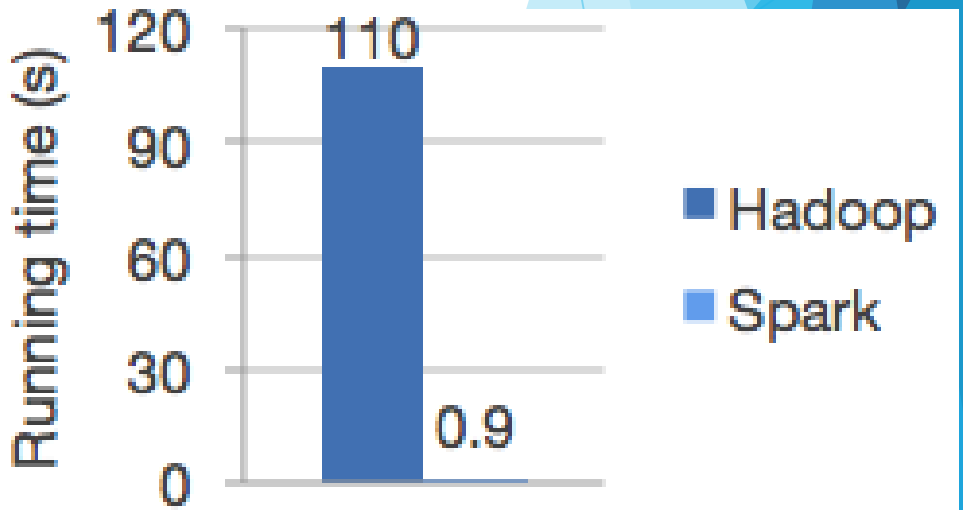
- Microsoft's open-source deep-learning toolkit
- https://github.com/Microsoft/CNTK
- Written in C++, bindings in Python, C#
- Runs over 80% Microsoft internal DL workload
- Can express a huge variety of deep architectures
  - LSTMS, ConvNets, RL, Gans, etc….
- Automatic differentiation
- Compute Agnostic: Compiles down to performant machine code
- Fast!







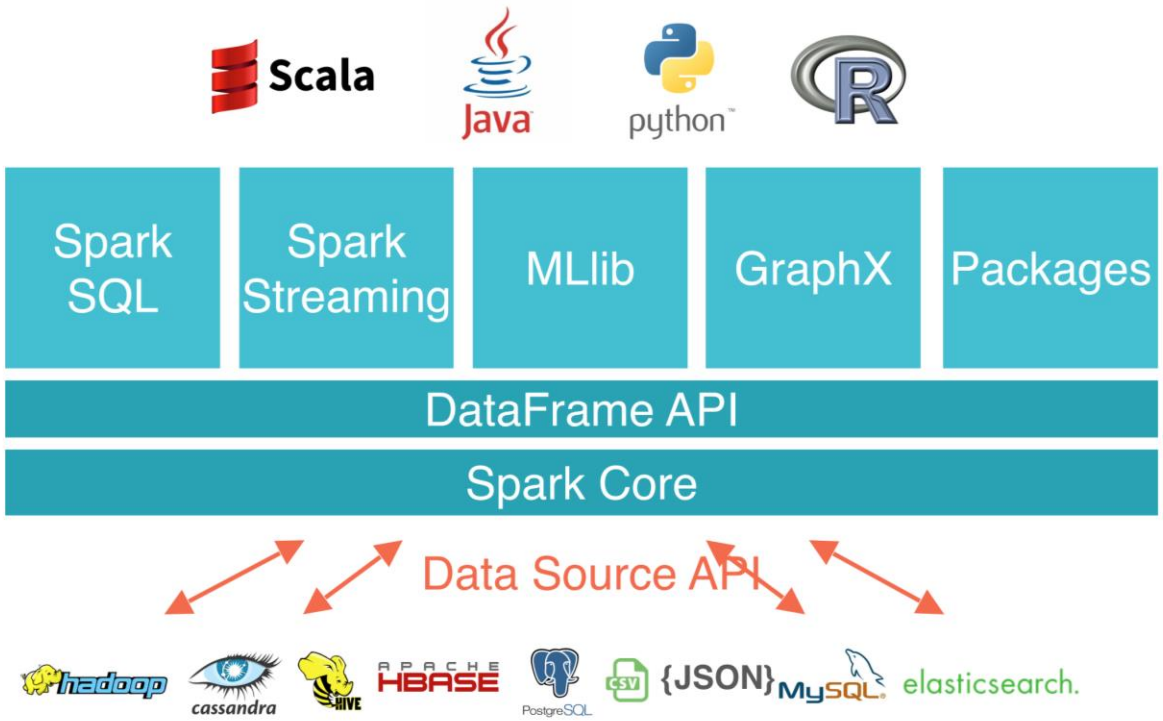
- A **fault-tolerant distributed** computing framework
- Generalizes, optimizes, and combines Map-Reduce and SQL style-operations
- Scales to thousands of machines
- Functional API
- Built in Scala, but has bindings in Python, R, Java, C#, F#
- Has a flourishing community
  - SparkML
  - GraphX
  - Structured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

MMLSpark

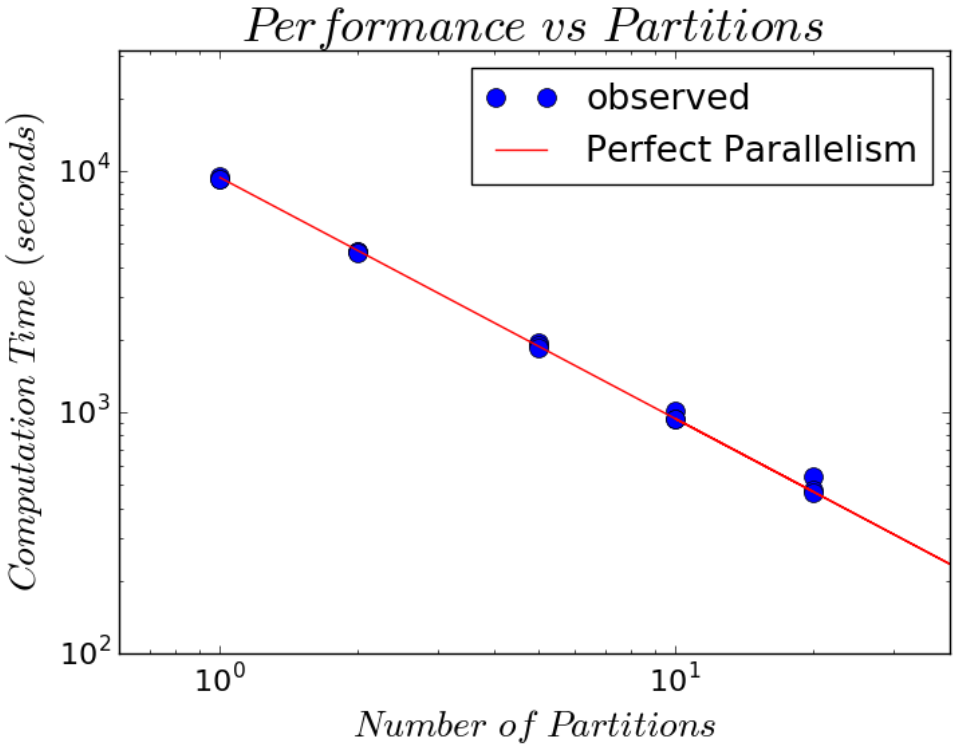
aka.ms/mmlspark

- Open Source
  - Contributions welcome!
  - Get started with our docker image + data science examples/course **aka.ms/mmlspark**
- **Integrates and unifies Spark, CNTK, and OpenCV, LightGBM, HTTP, and more**
- Tons of other useful abstractions and tools:
  - **Automatic Python/R code generation**, Auto-Featurization, Trained DNN Repository, High Performance Web Deployments

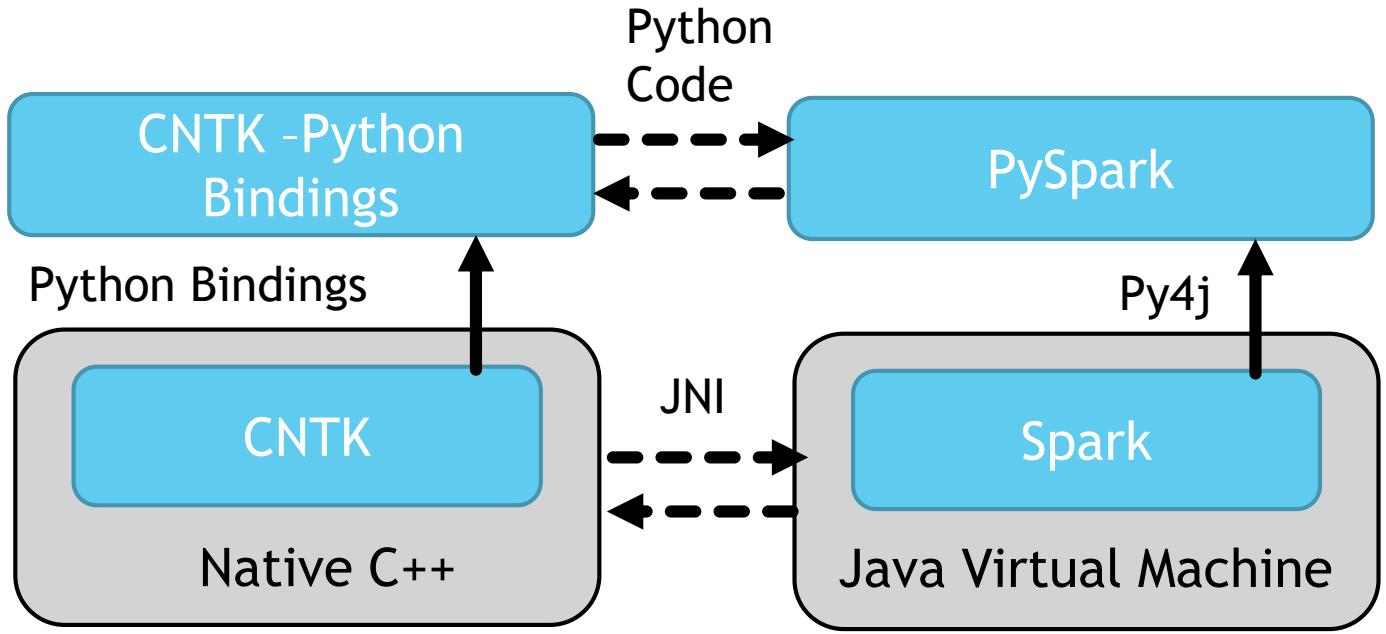
Cognitive Toolkit on Spark

- 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

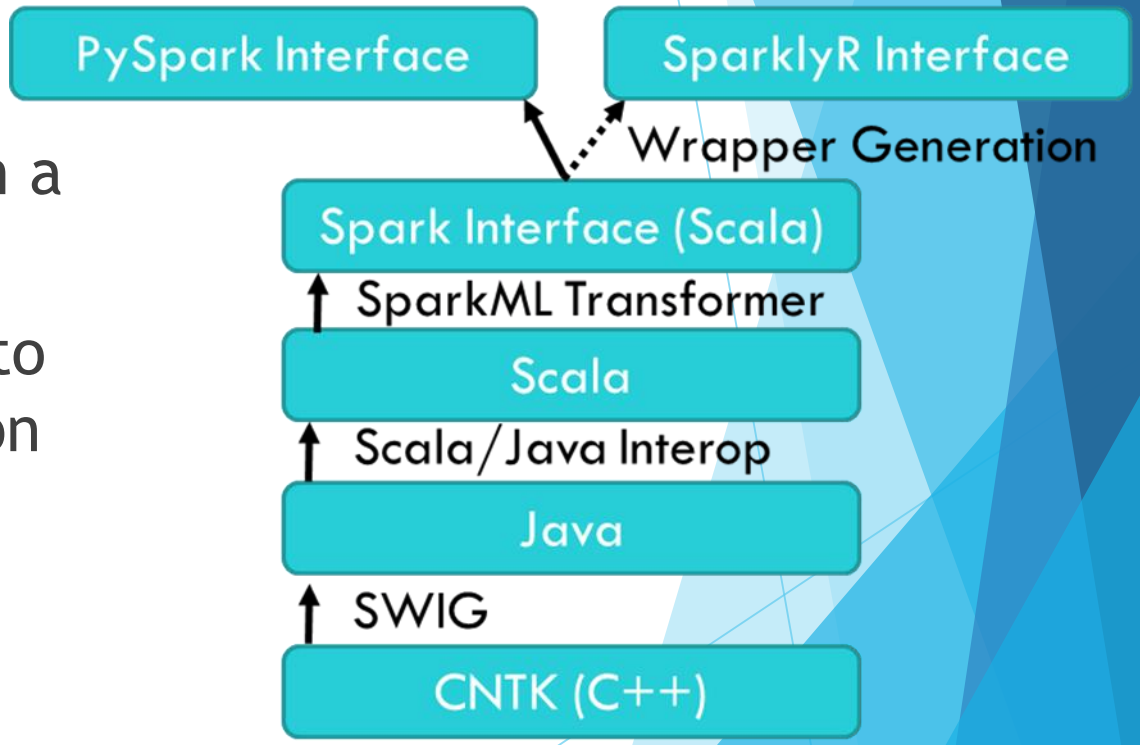
Cognitive Toolkit's Java Bindings

- 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);
```

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




Snow
Leopard
Trust

Snow Leopard Conservation

- 3,900-6,500 individuals left in the wild
- Little known about their ecology, behavior, movement patterns, survival rates
- More data required to influence survival

Mining

Poaching

Retribution Killing

Snow leopard no longer 'endangered'

14 September 2017 Share

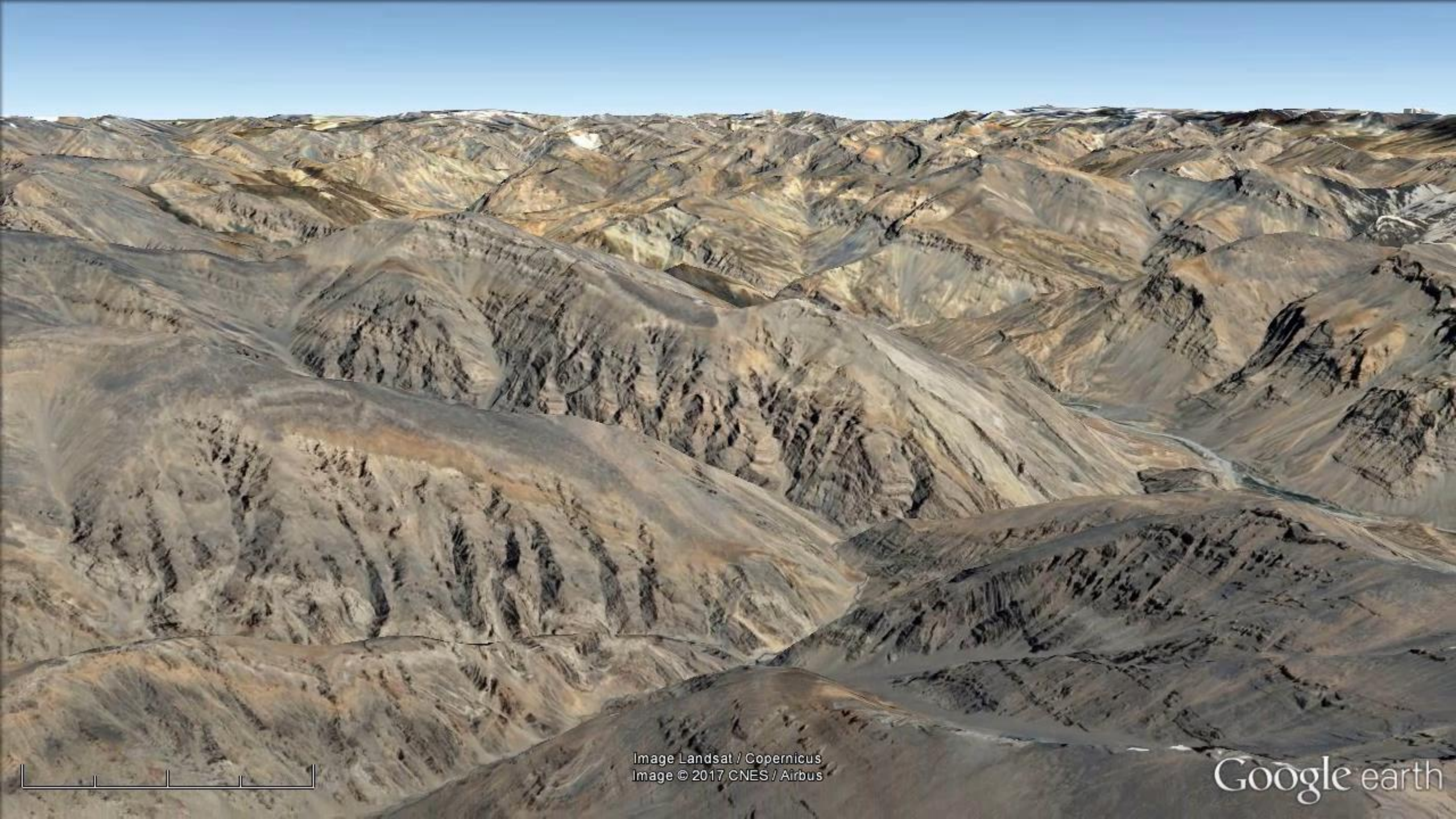
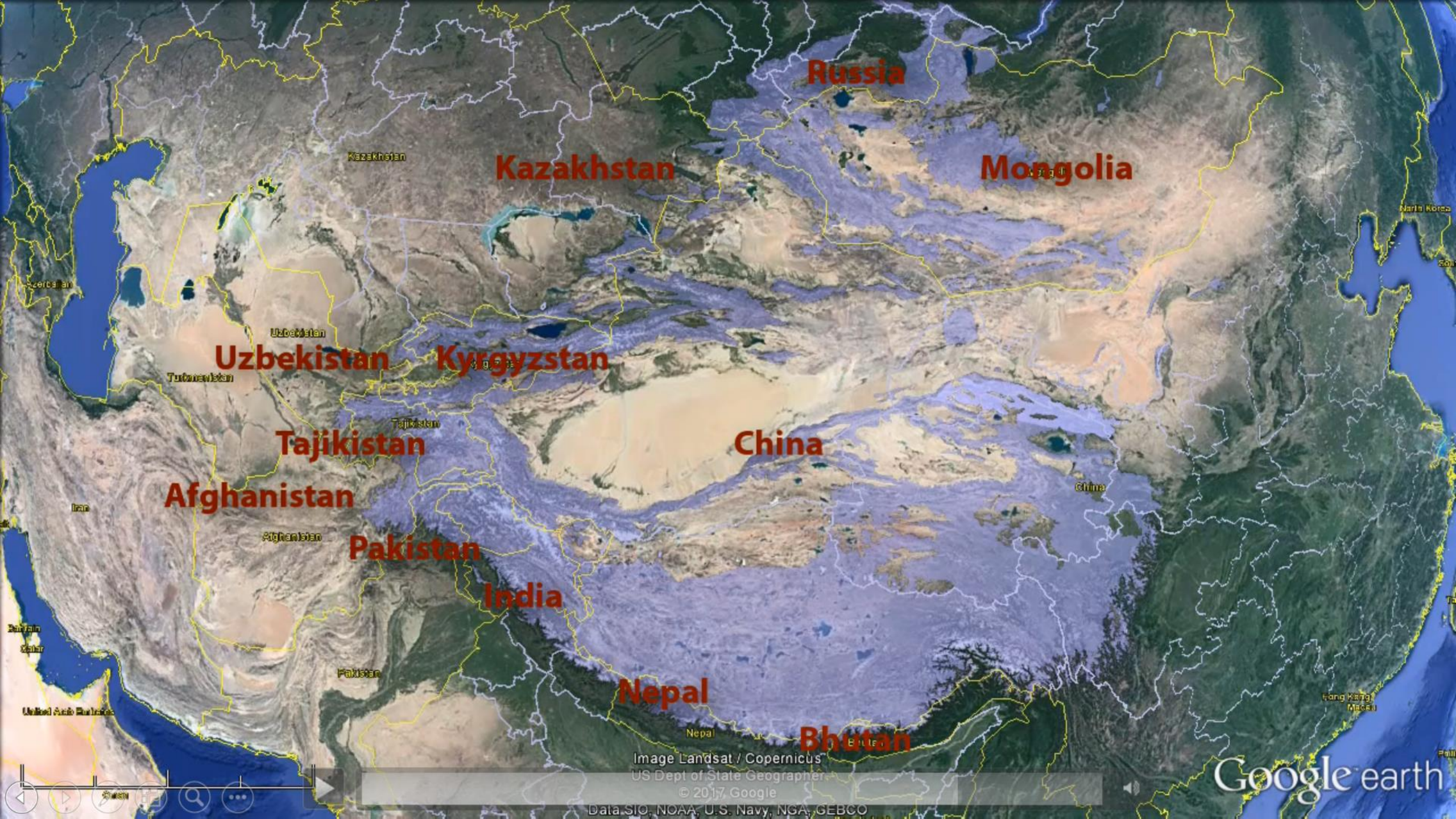
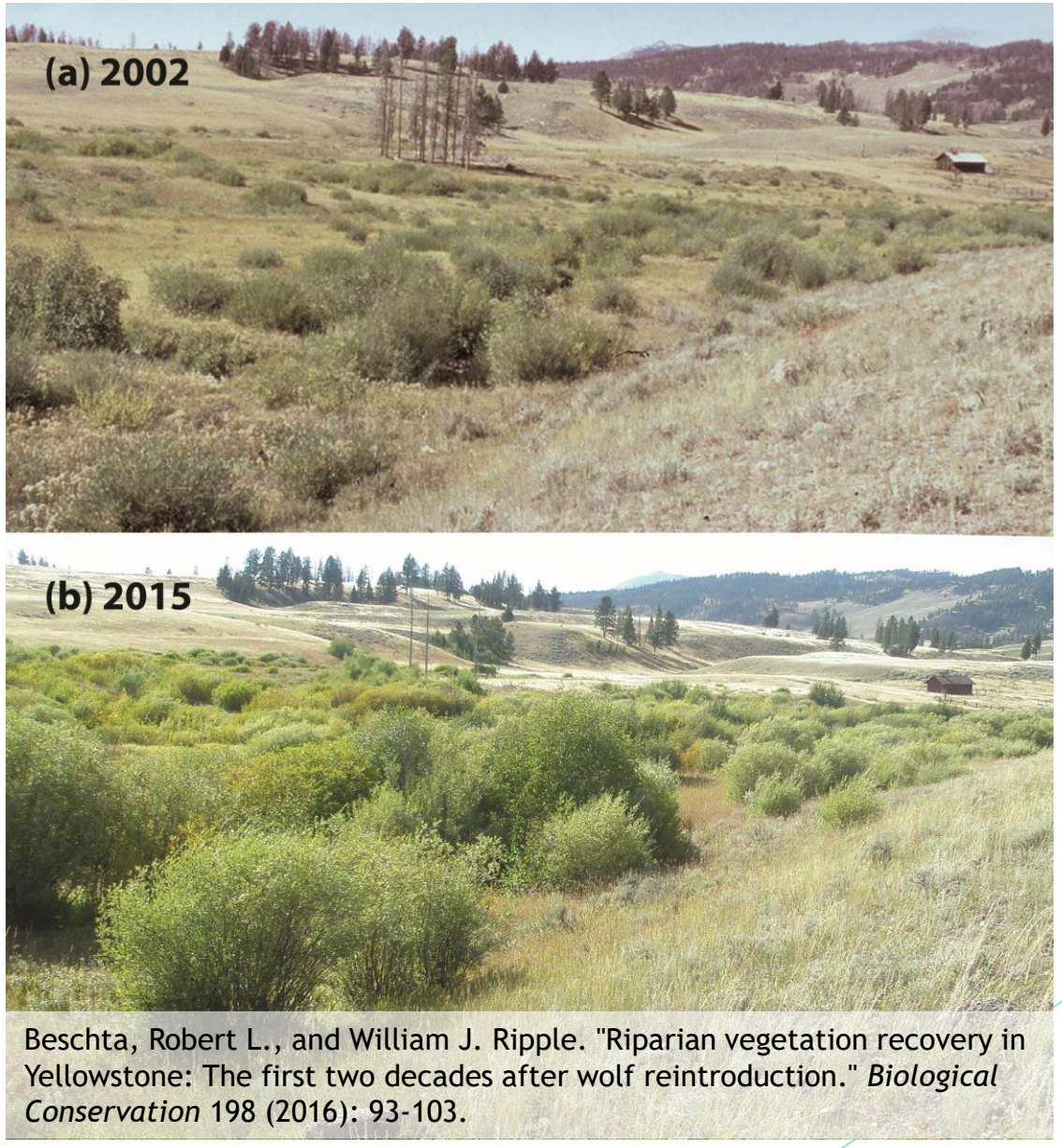


Image Landsat / Copernicus
Image © 2017 CNES / Airbus
Google earth


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

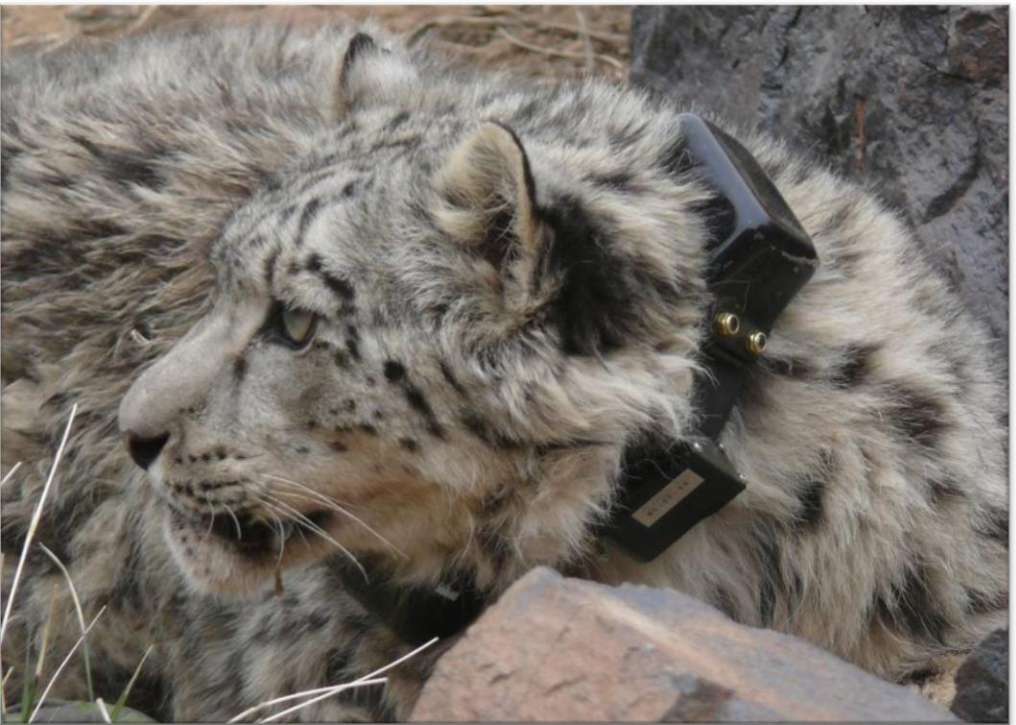
The Impact of Apex Predators



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.

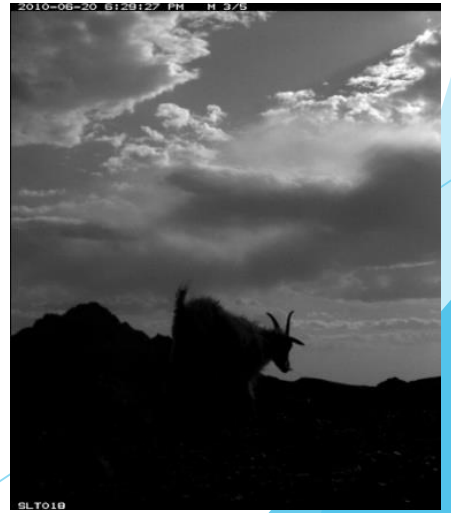
Gathering Leopard Data

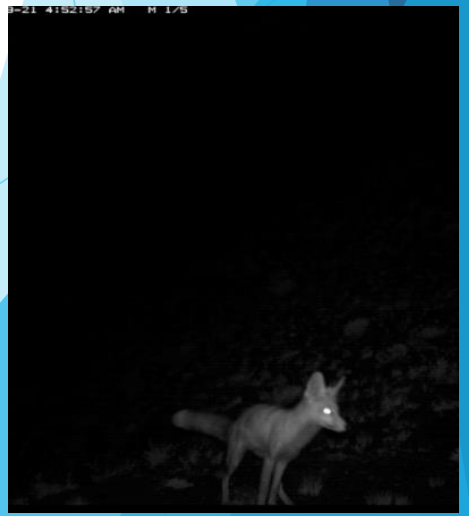
- 23 leopards collared in 9 years
- 42 camera traps over 1,700 sq km
- ~1.3 mil images

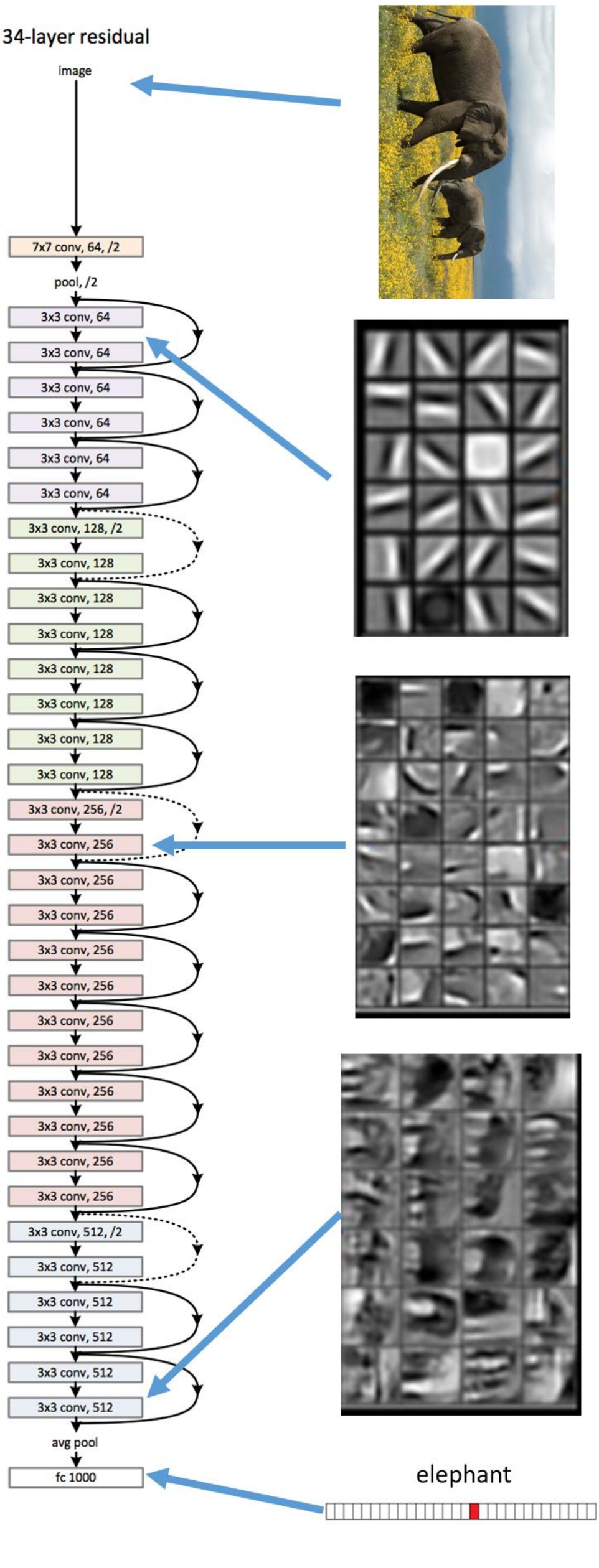


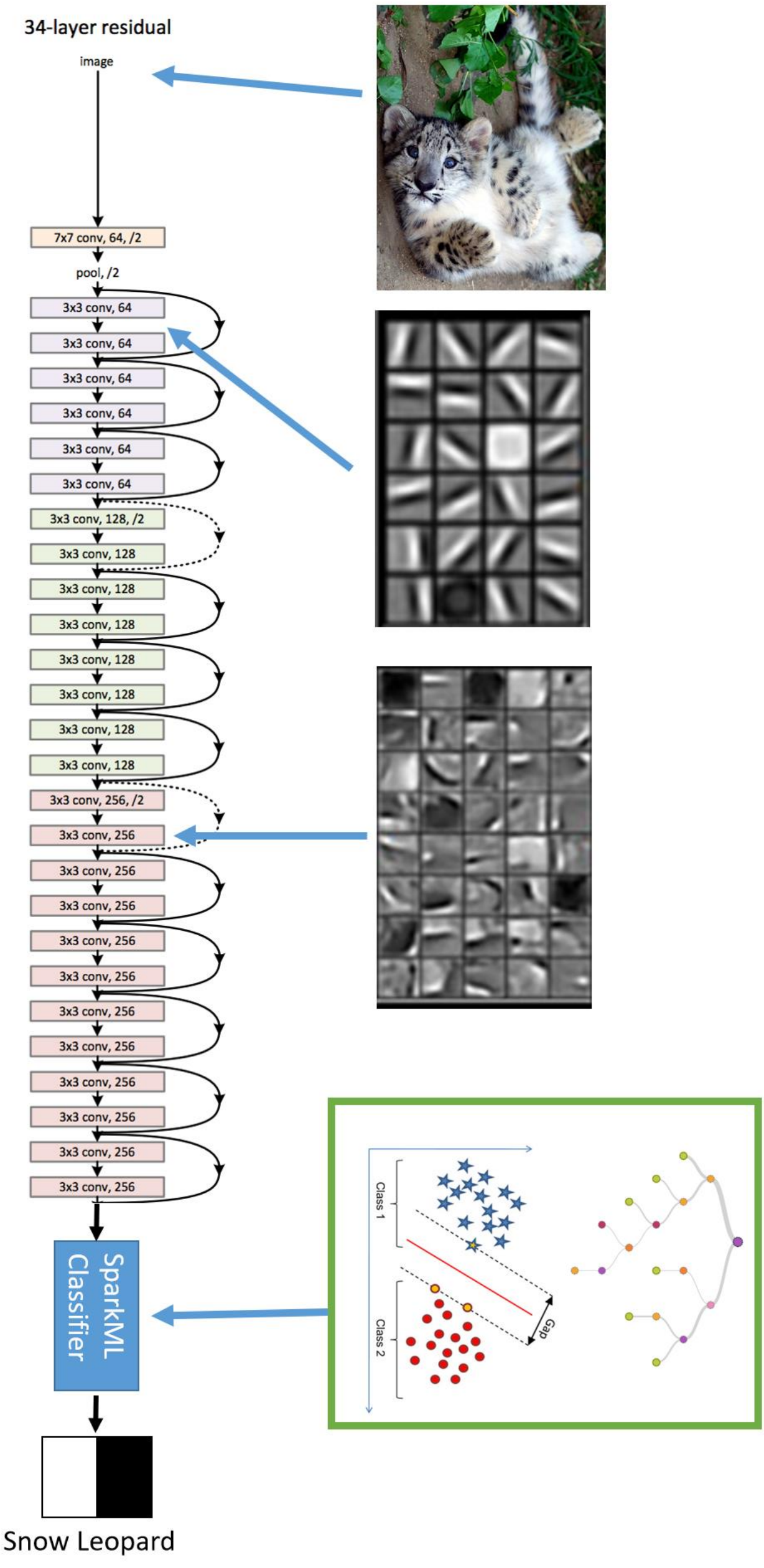
Camera Trap Images

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



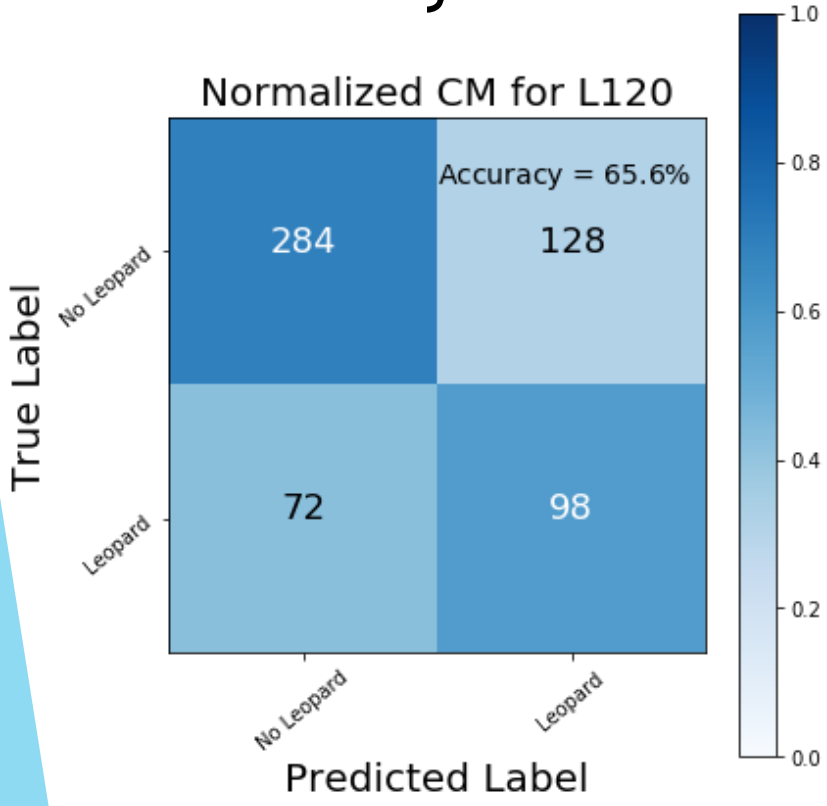



34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
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

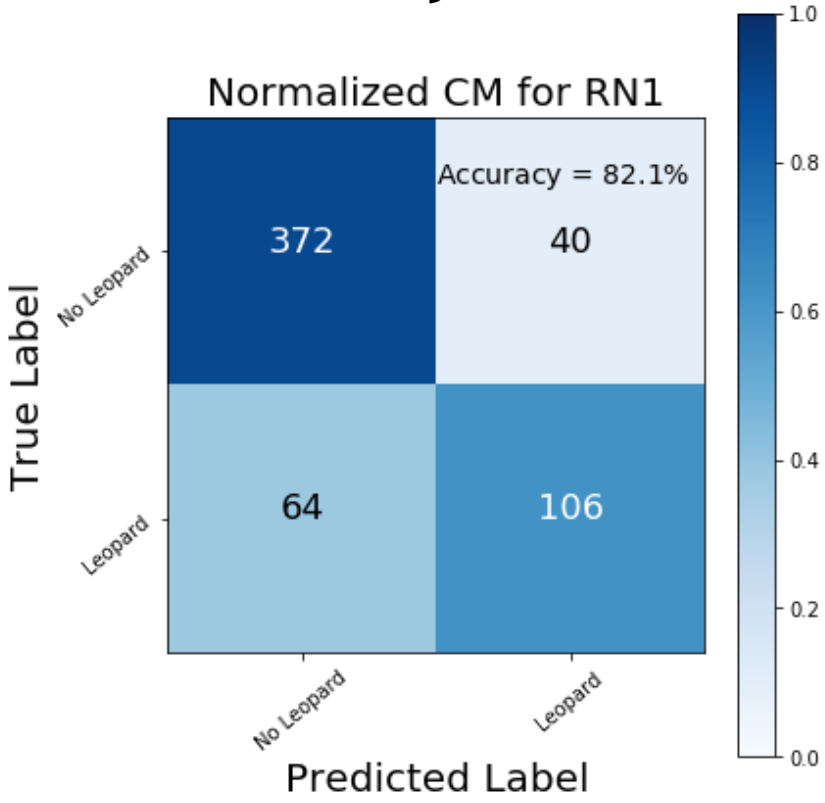

34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
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

Performance

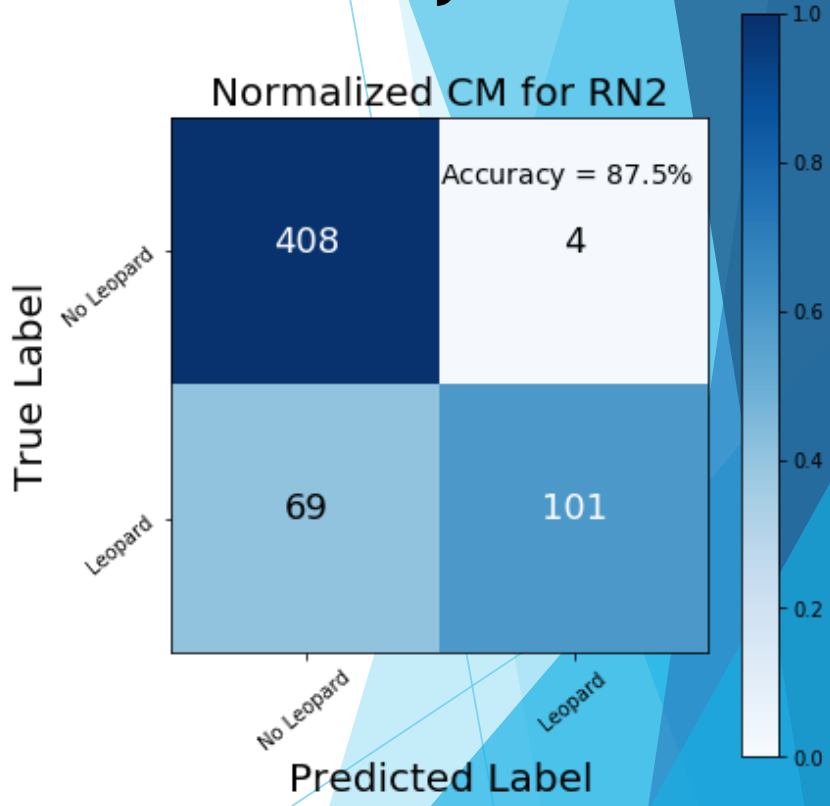
Accuracy 65.6%



Accuracy 82.1%



Accuracy 87.5%



Without Deep Featurization

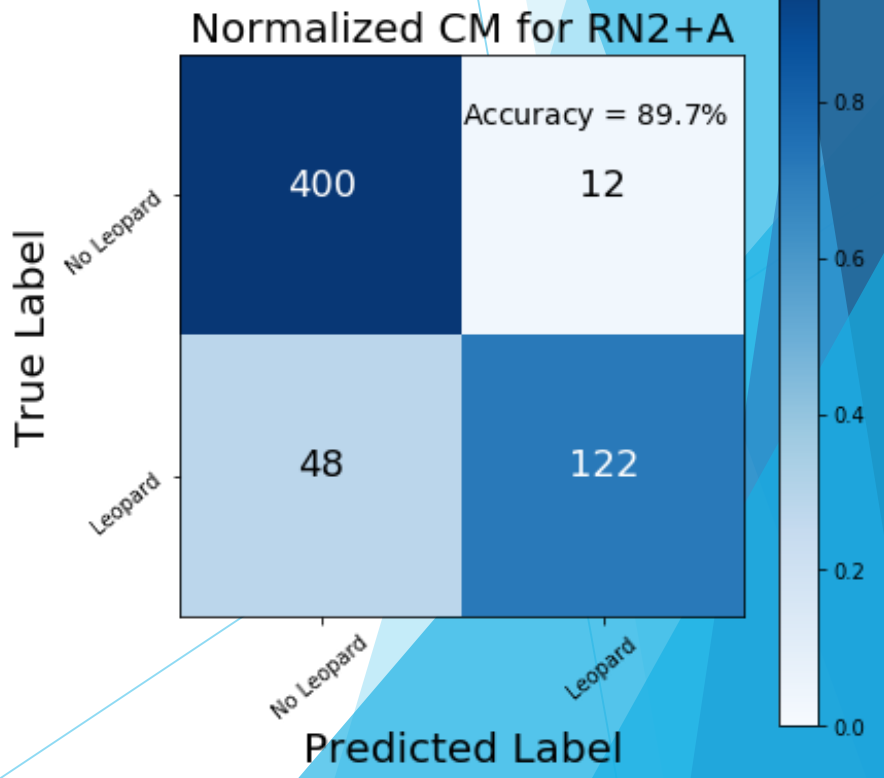
With Deep Featurization

Further Refinements

- Dataset Augmentation

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

Accuracy 89.7%



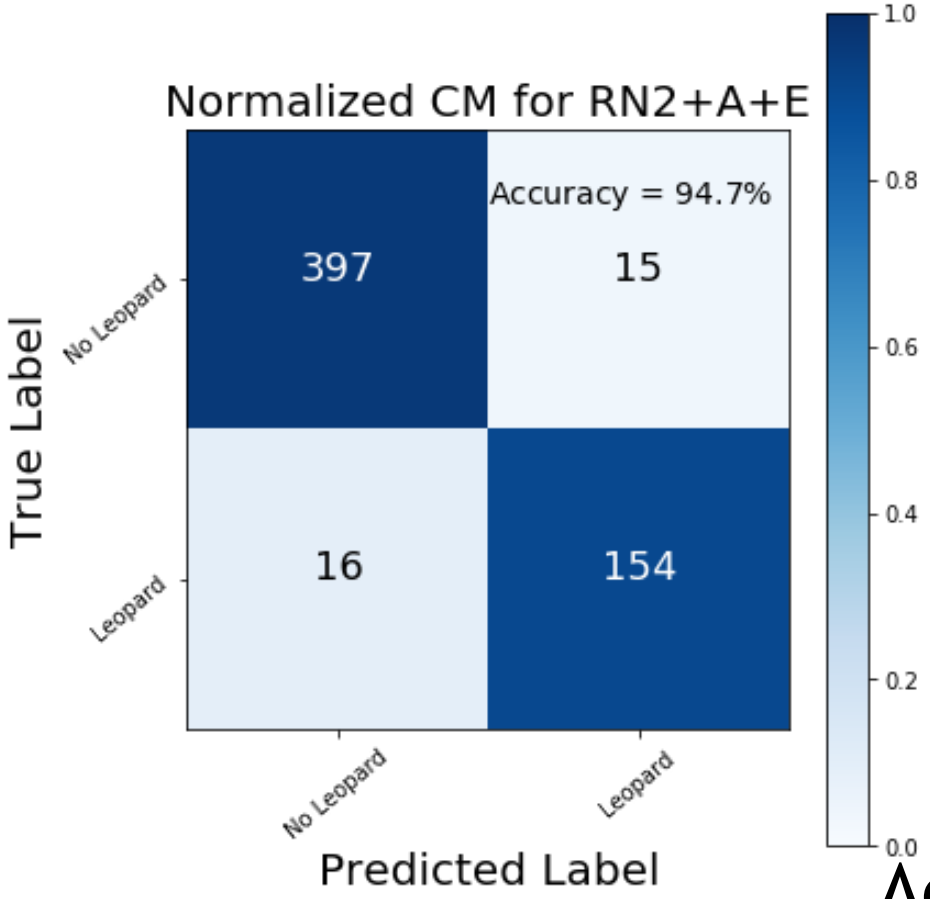

2012-06-30 9:03:10 PM M 1/5
24°C
RECONYX
SLT017

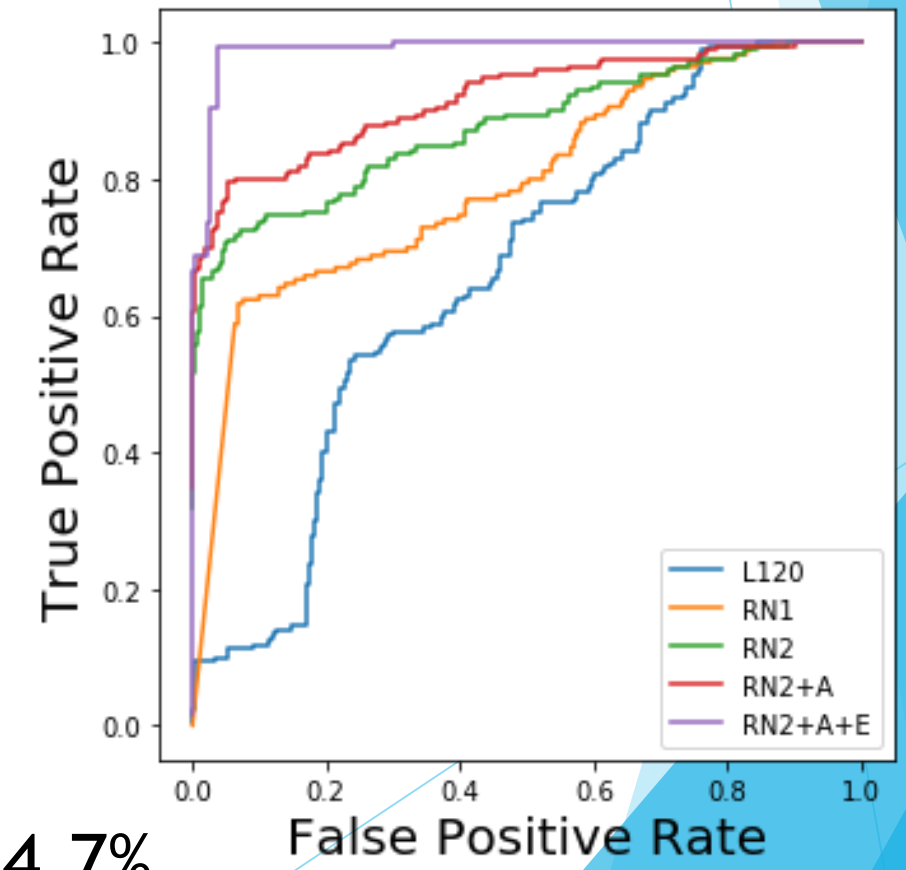

2012-06-30 9:03:11 PM M 2/5
24°C
RECONYX
SLT017


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])
```

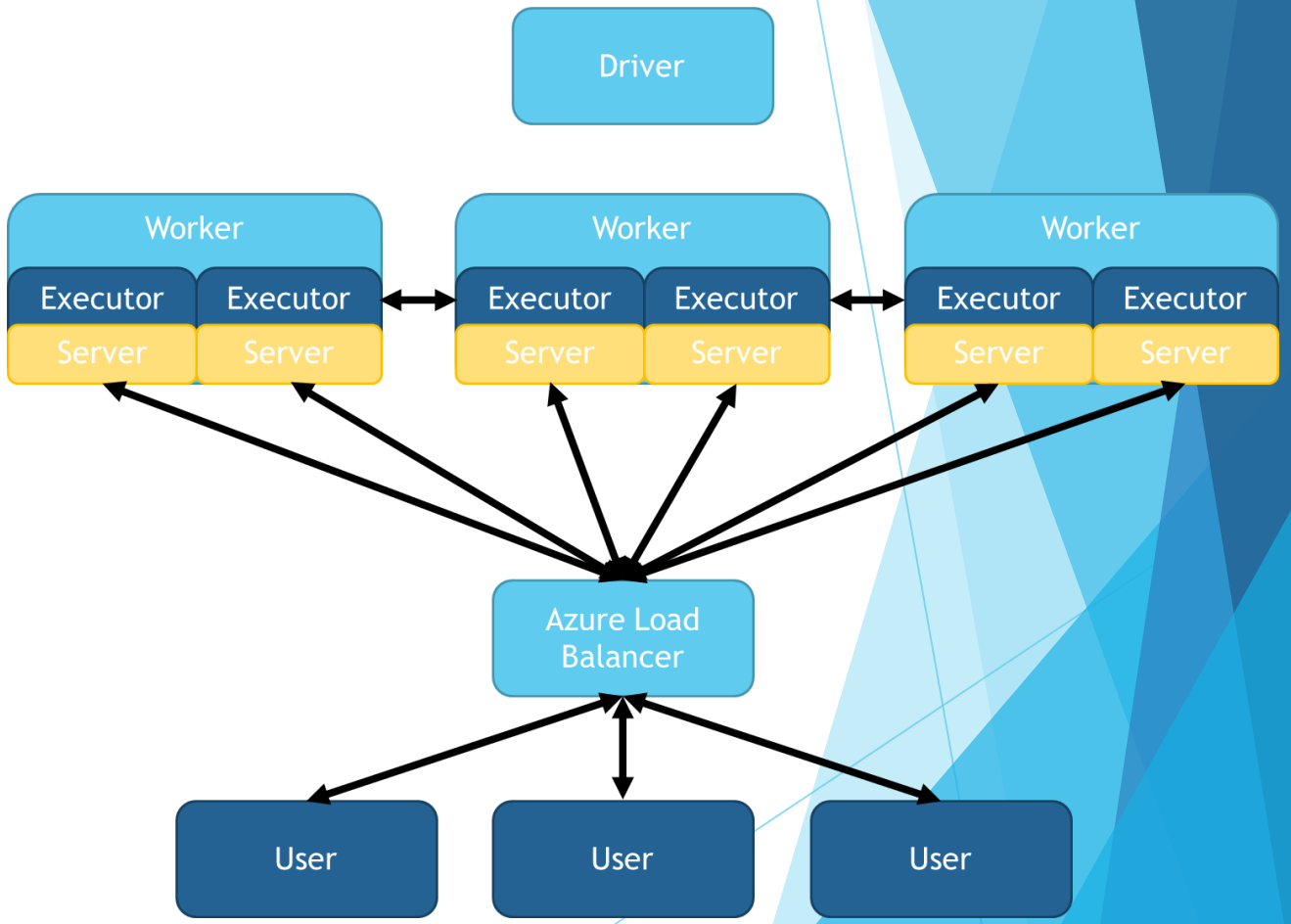




Accuracy 94.7%

MMLSpark Serving

- Scalable, fault-tolerant, and convenient RESTful services
- Simplest and most efficient way to operationalize spark models on any Spark platform
- Available in Python, Scala, and R
- Parallel at every level:
  - Multi-threaded, multi-JVM, multi-node

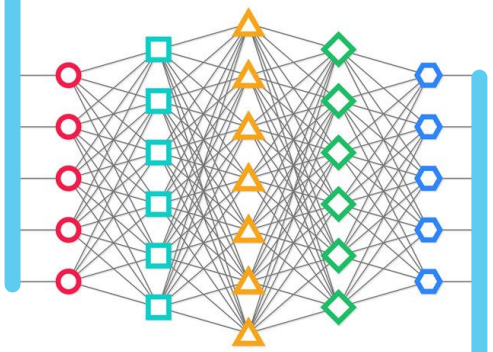


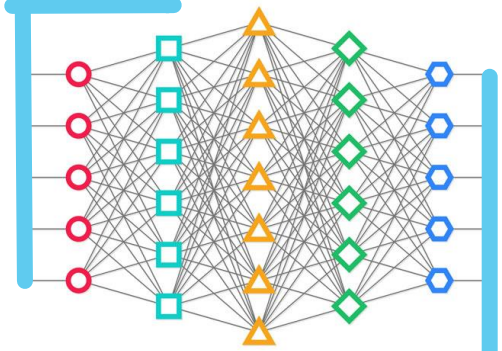



Time

1 2 3 4 5 6

CNTK Pipeline





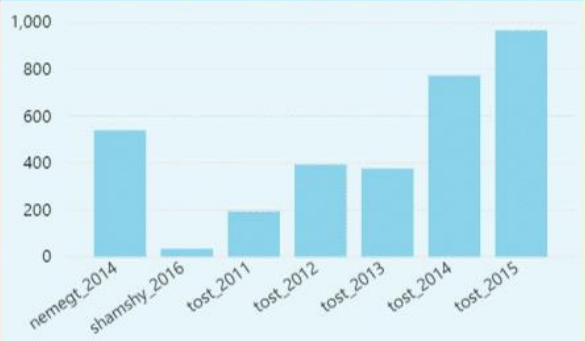
Time

1 2 3 4 5 6

Power BI | My Workspace > Snow Leopard Dashboard

File | View | Edit report | Explore | Refresh | Pin Live Page | Press F11 to exit full screen | Usage Metrics | View related | Subscribe

Total	Leopards	Other
51.88K	3274	48.61K



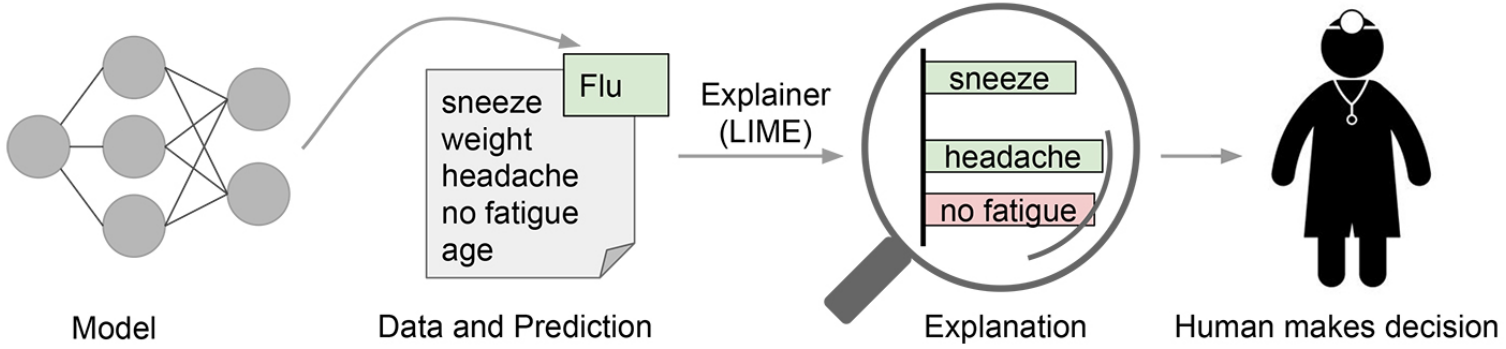
Survey Selector

- Select All
- nemegt_2014
- shamshy_2016
- tost_2011
- tost_2012
- tost_2013
- tost_2014

FILTERS

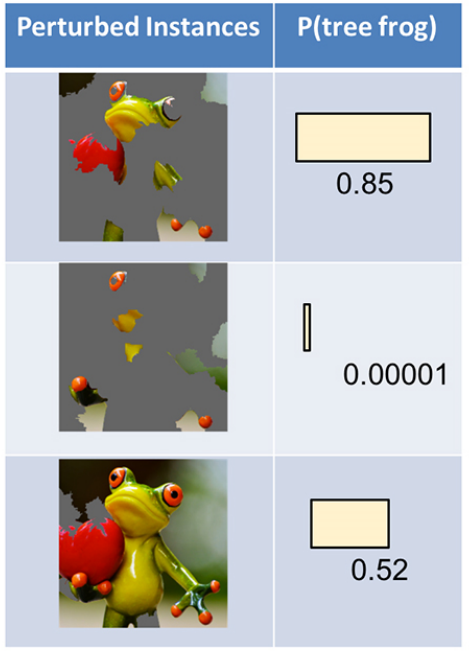
Page 1

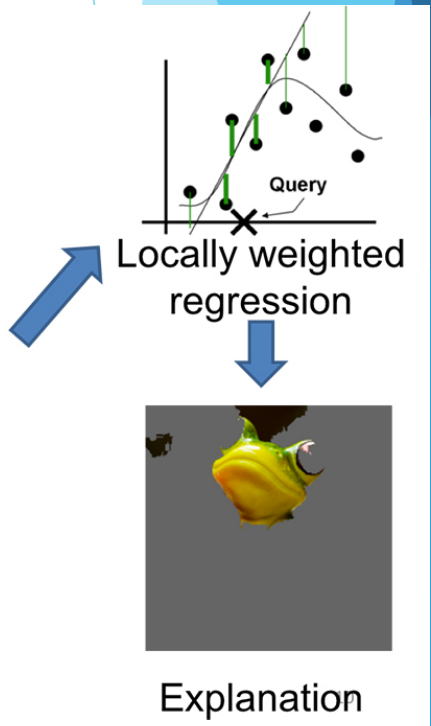
Ongoing work: Local Interpretable Model-Agnostic Explanations (LIME)



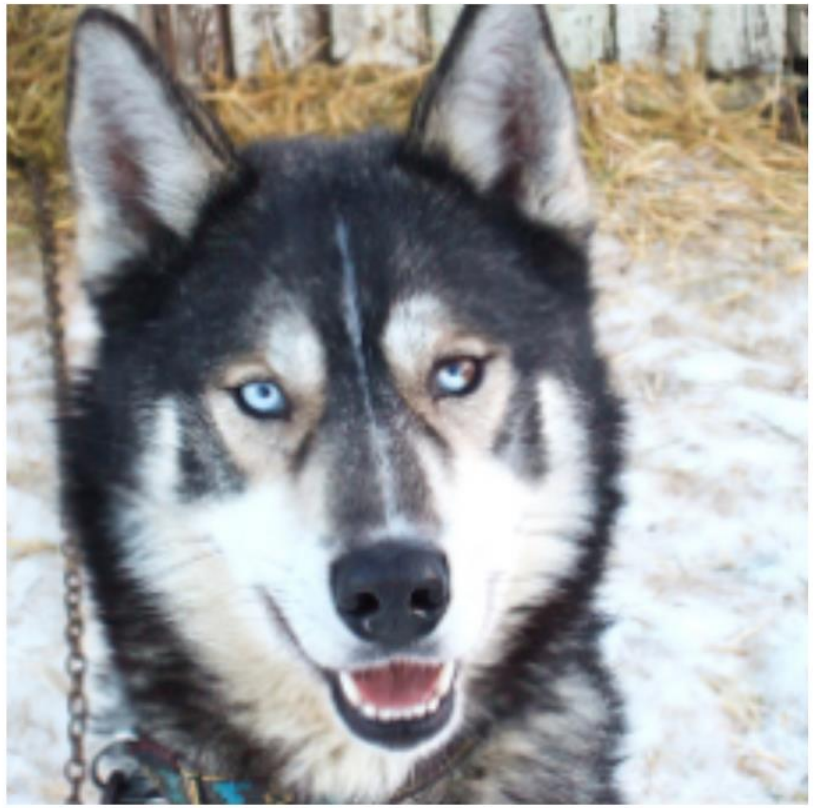
Interpretable Components

Original Image
P(tree frog) = 0.54





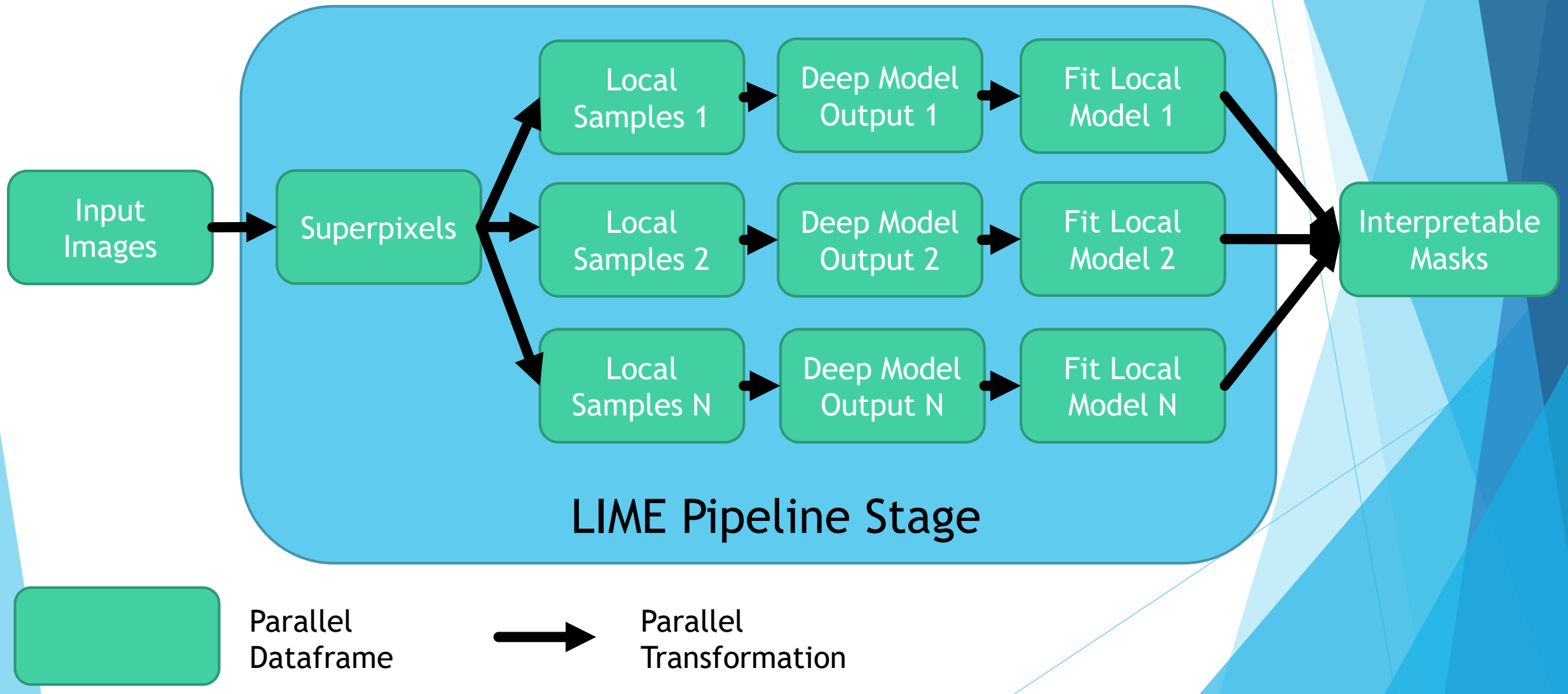
Error Detection



(a) Husky classified as wolf

(b) Explanation

LIME on Spark



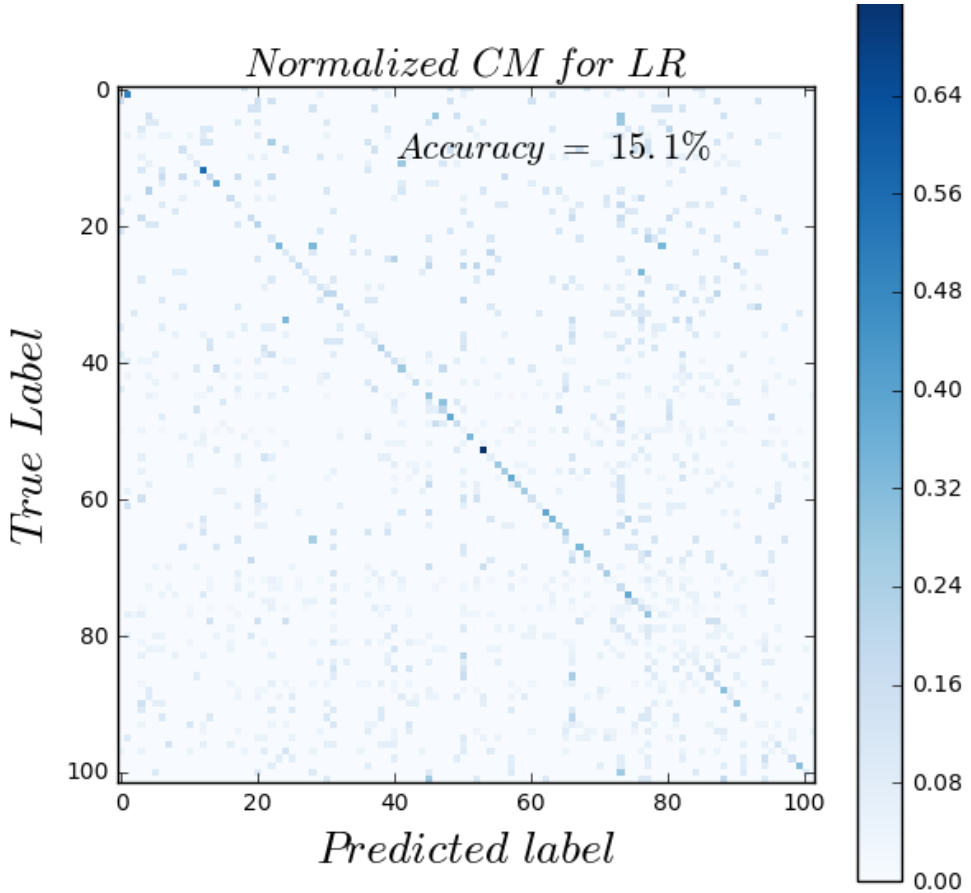
Thanks to

- MMLSpark Team:
  - **Ben Brodsky**, 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, Manik Jindall

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**
- Flexible and Scalable Deep Learning with MMLSpark (in review in PMLR):
  - https://arxiv.org/abs/1804.04031
- 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