SPARK+AI SUMMIT EUROPE

Deep Reality Simulation For Automated Poacher Detection

Mark Hamilton, Microsoft
Anand Raman, Microsoft

#SAISDD2

npr set station news arts & life music programs shop

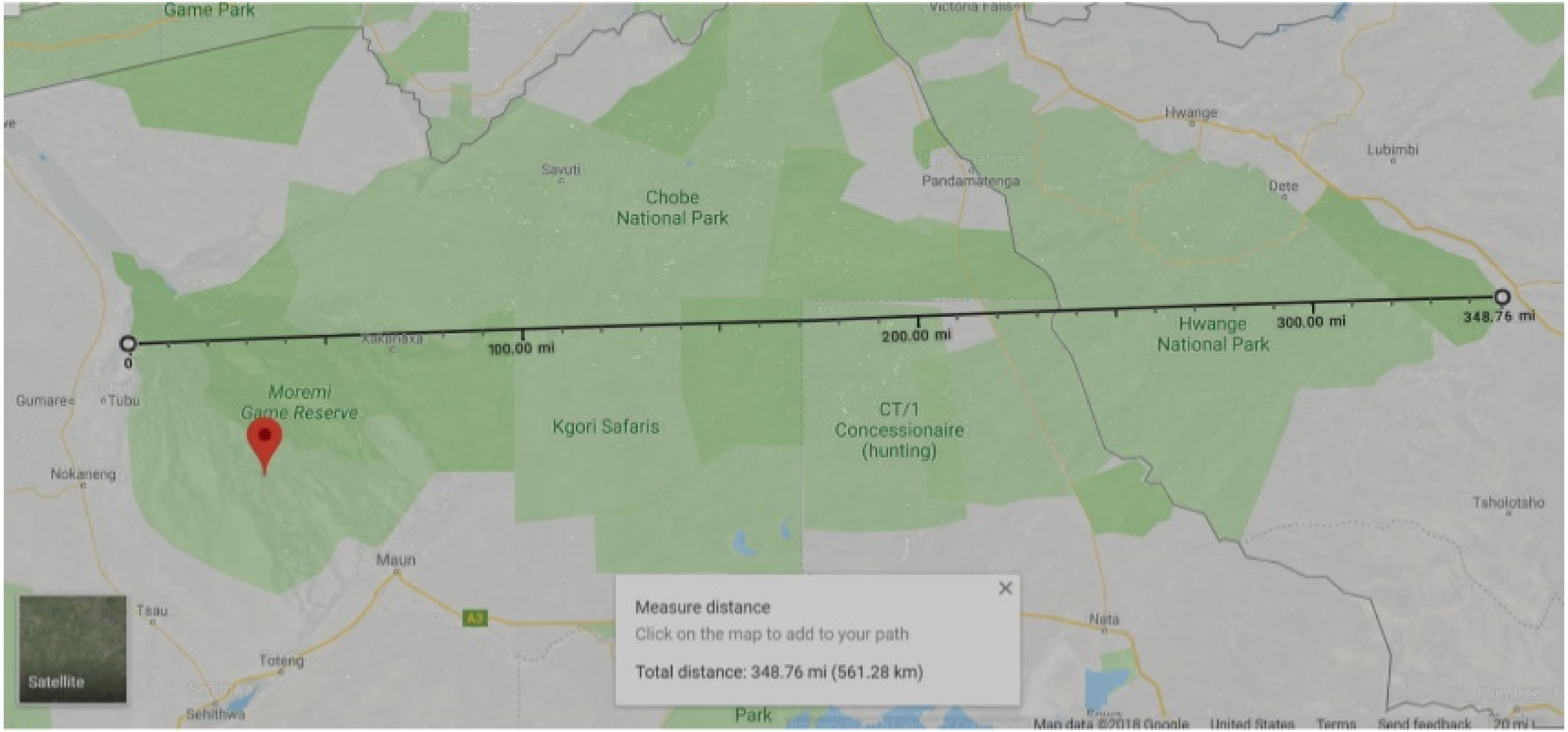
AFRICA

Nearly 90 Elephants Found Dead Near Botswana Sanctuary, Killed By Poachers

September 3, 2018 · 5:57 PM ET

EMILY SULLIVAN

Okavango Delta Wildlife Sanctuary



The Economics of Poaching

Average Elephant Tusk Wight:
20-40 lbs

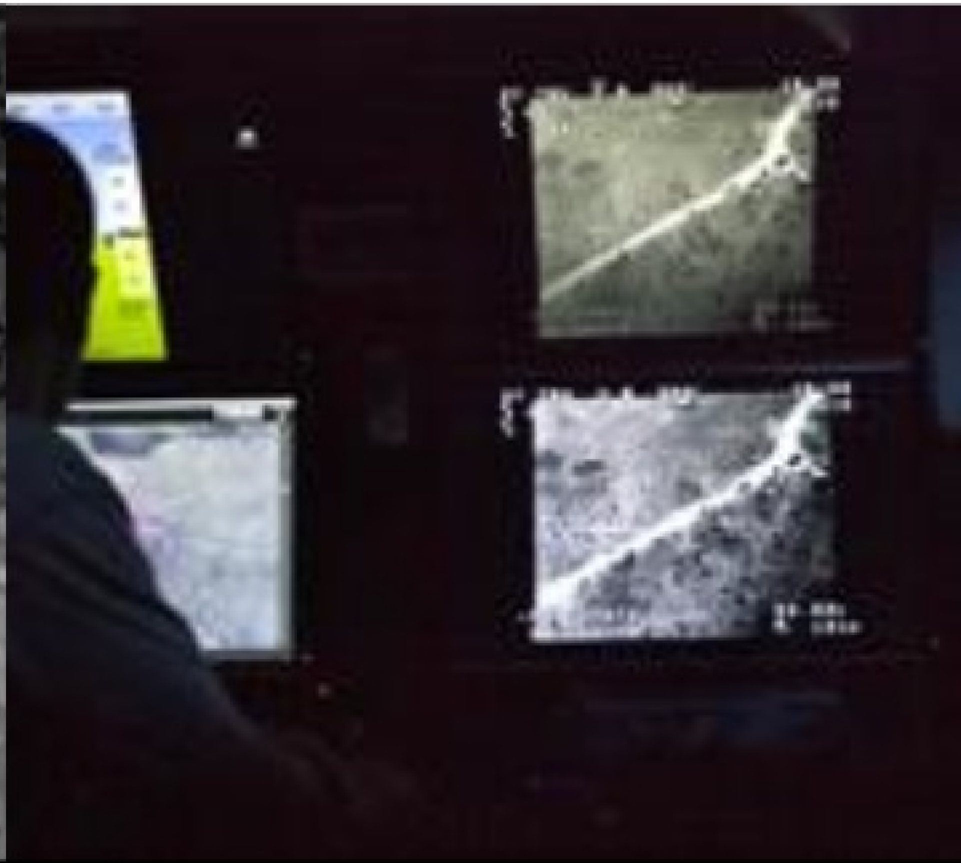
Average Per Capita Income Across
Sub-Saharn Africa (2008):
$762

Years of Average Income from one
Poached Elephant:
52 – 104 years

Air Shepherd

The Lindbergh Foundation

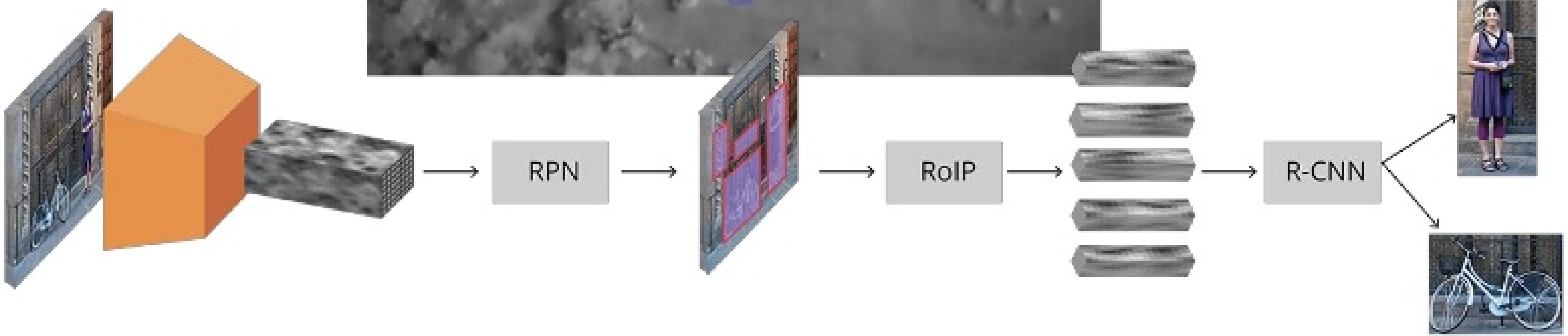




Previous System: EyeSpy

SPOT Poachers in Action: Augmenting Conservation Drones with Automatic Detection in Near Real Time

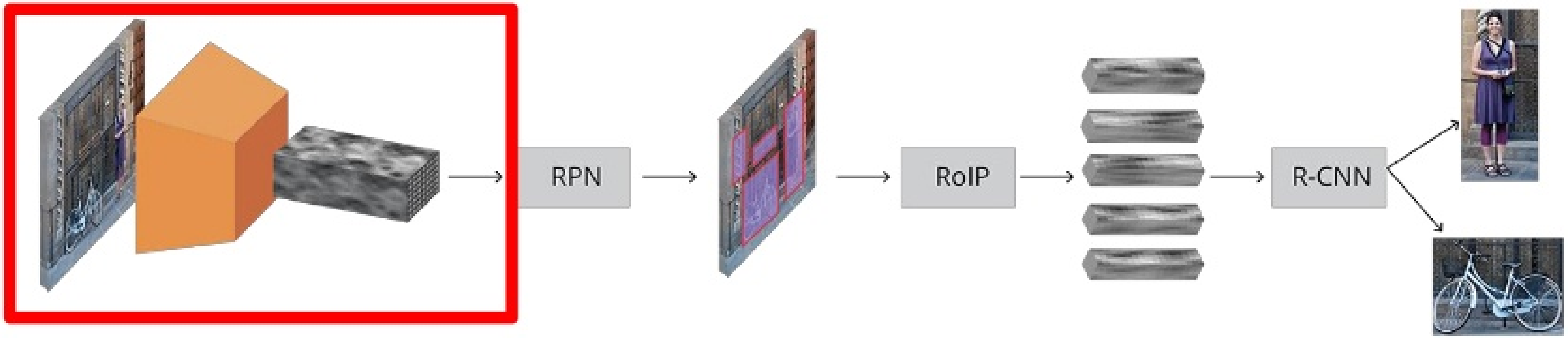
Elizabeth Bondi[1], Fei Fang[2], Mark Hamilton[3], Debarun Kar[1], Donnabell Dmello[1], Jongmoo Choi[1] Robert Hannaford[4], Arvind Iyer[4], Lucas Joppa[3], Milind Tambe[1], Ram Nevatia[1]
[1]University of Southern California, Los Angeles, CA, 90089 USA
{bondi, dkar, ddmello, jongmooc, tambe, nevatia}@usc.edu
[2]Carnegie Mellon University, Pittsburgh, PA, 15213 USA, feifang@cmu.edu
[3]Microsoft, Redmond, WA, 98052 USA, {marhamil, lujoppa}@microsoft.com
[4]AirShepherd, Berkeley Springs, WV, 25411 USA
rob@coolideassolutions.com, arvind.iyer@lindberghfoundation.org



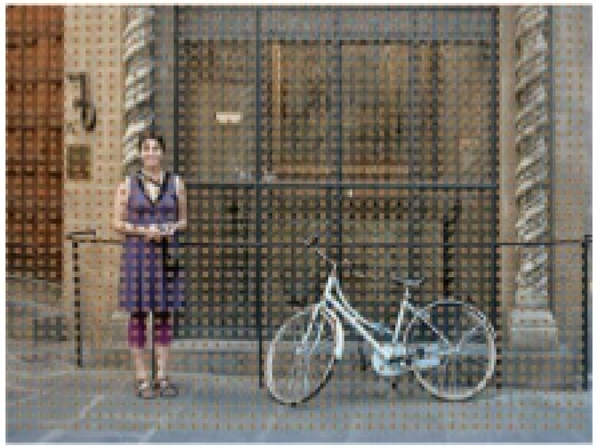
Faster R-CNN: Convolutional Featurization

Step 1

- A Convolutional network "featurizes" the image while maintaining its spatial structure

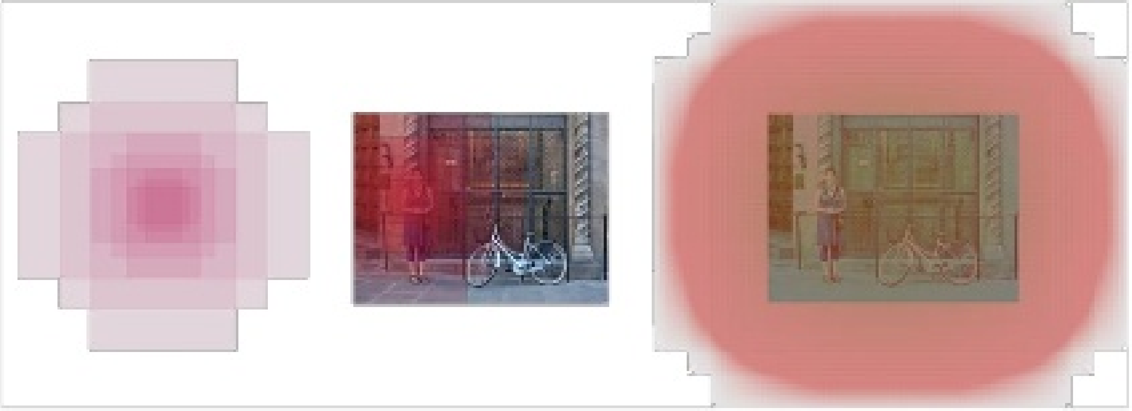


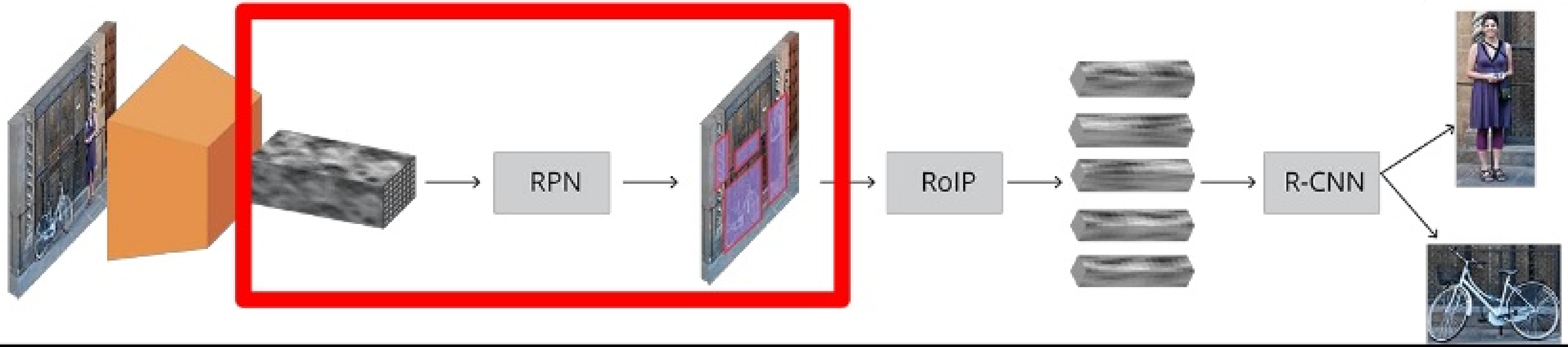
Faster R-CNN: Region Proposal Network



Step 2

- A "Region Proposal network" generates the probability that each "Anchor" has an object
- The RPN also predicts how to "adjust" the anchors to make them fit the object

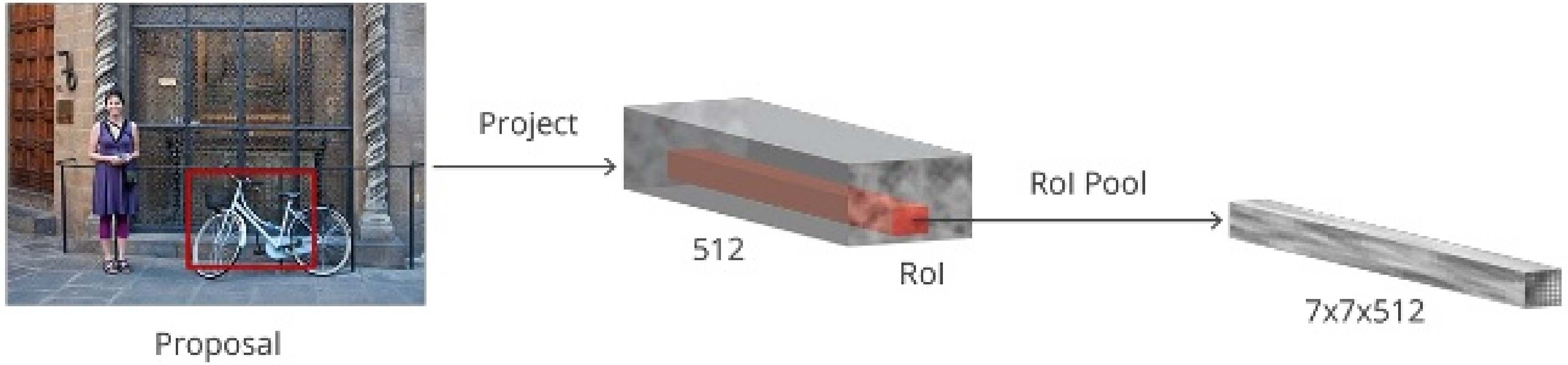


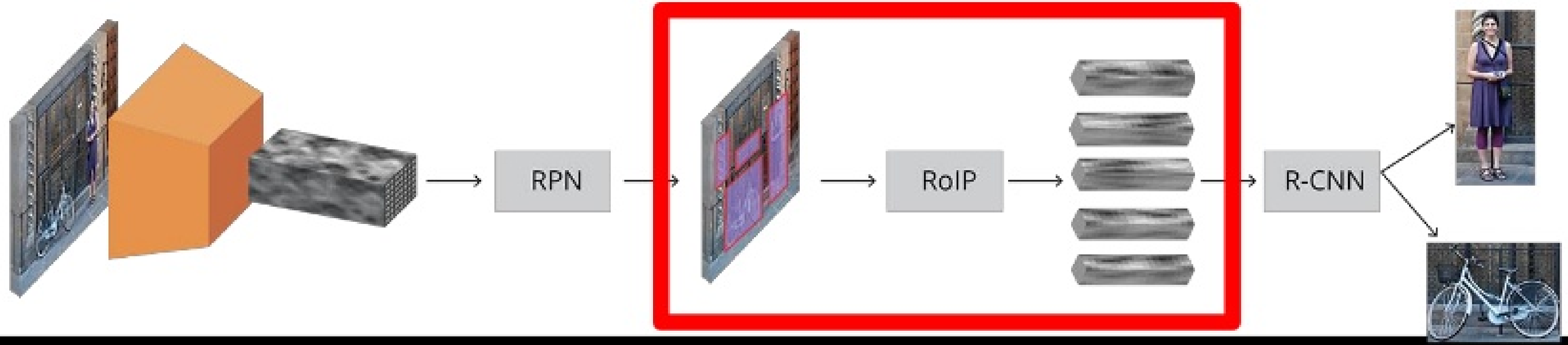


Faster R-CNN: Region of Interest Pooling

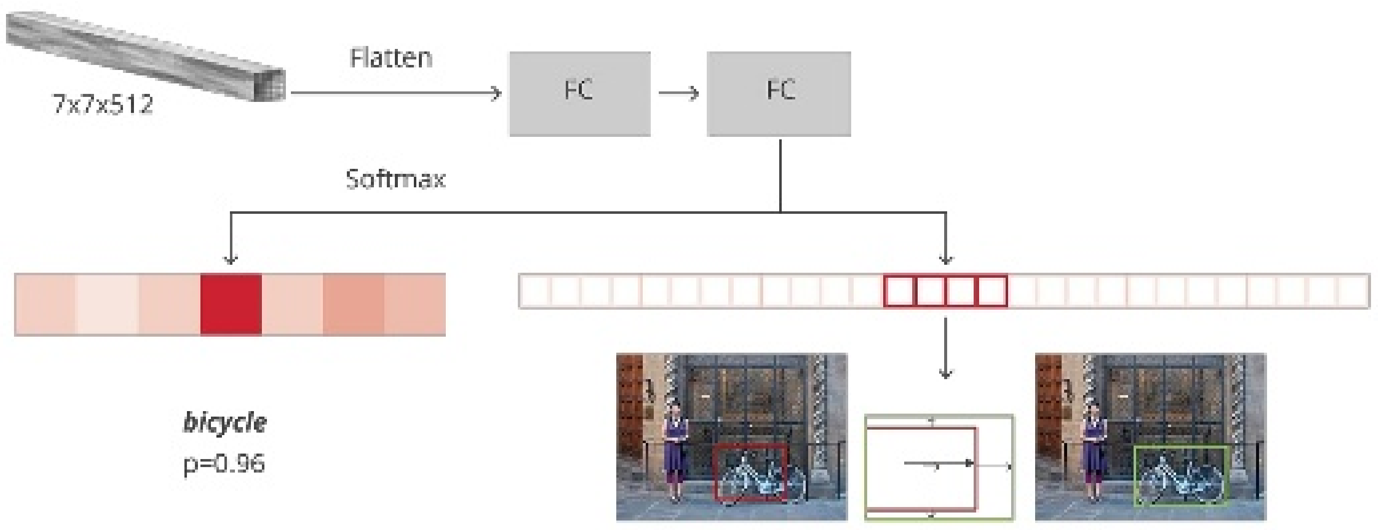
Step 3

- A “Region of Interest Pooling” stage condenses the convolutional features into a fixed size



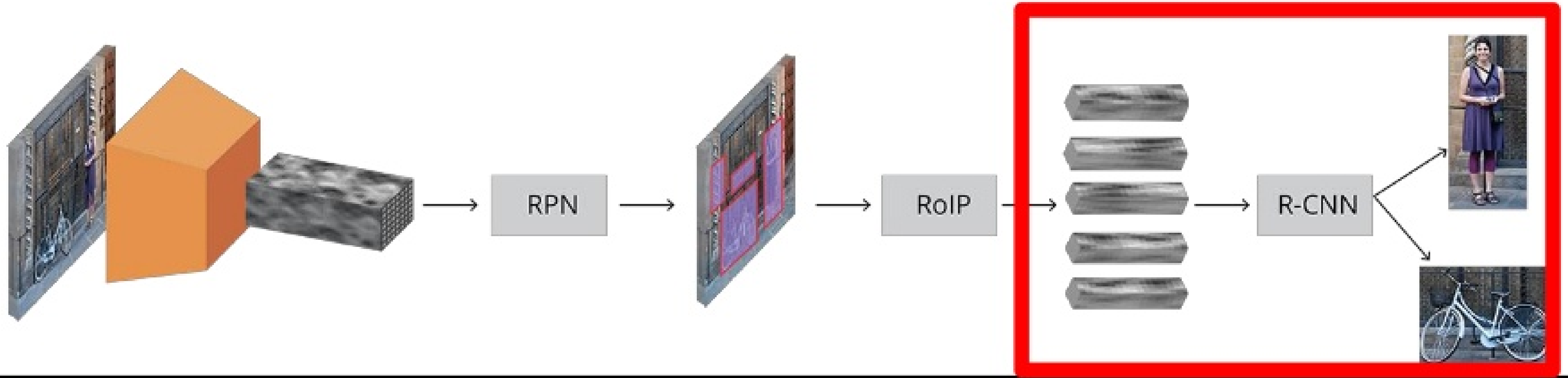


Faster R-CNN: Region Based CNN

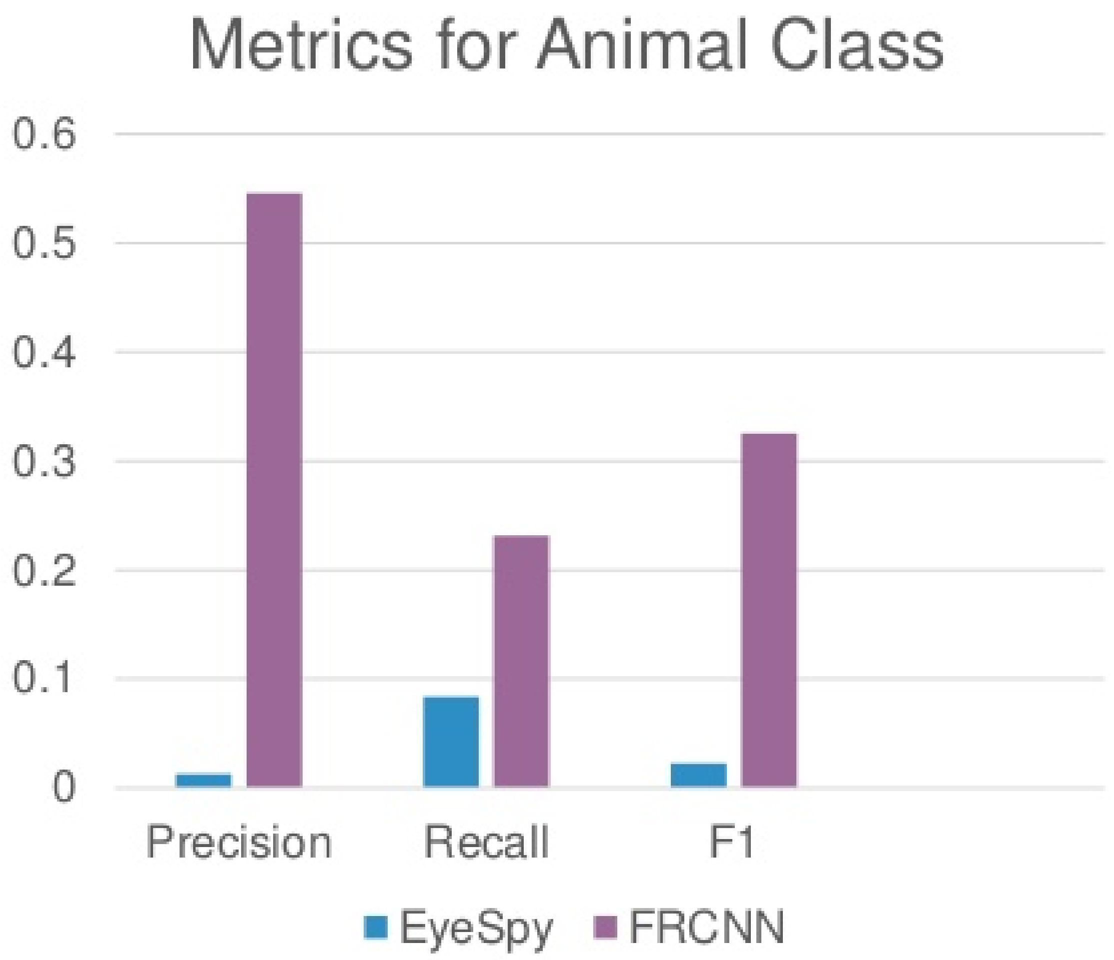


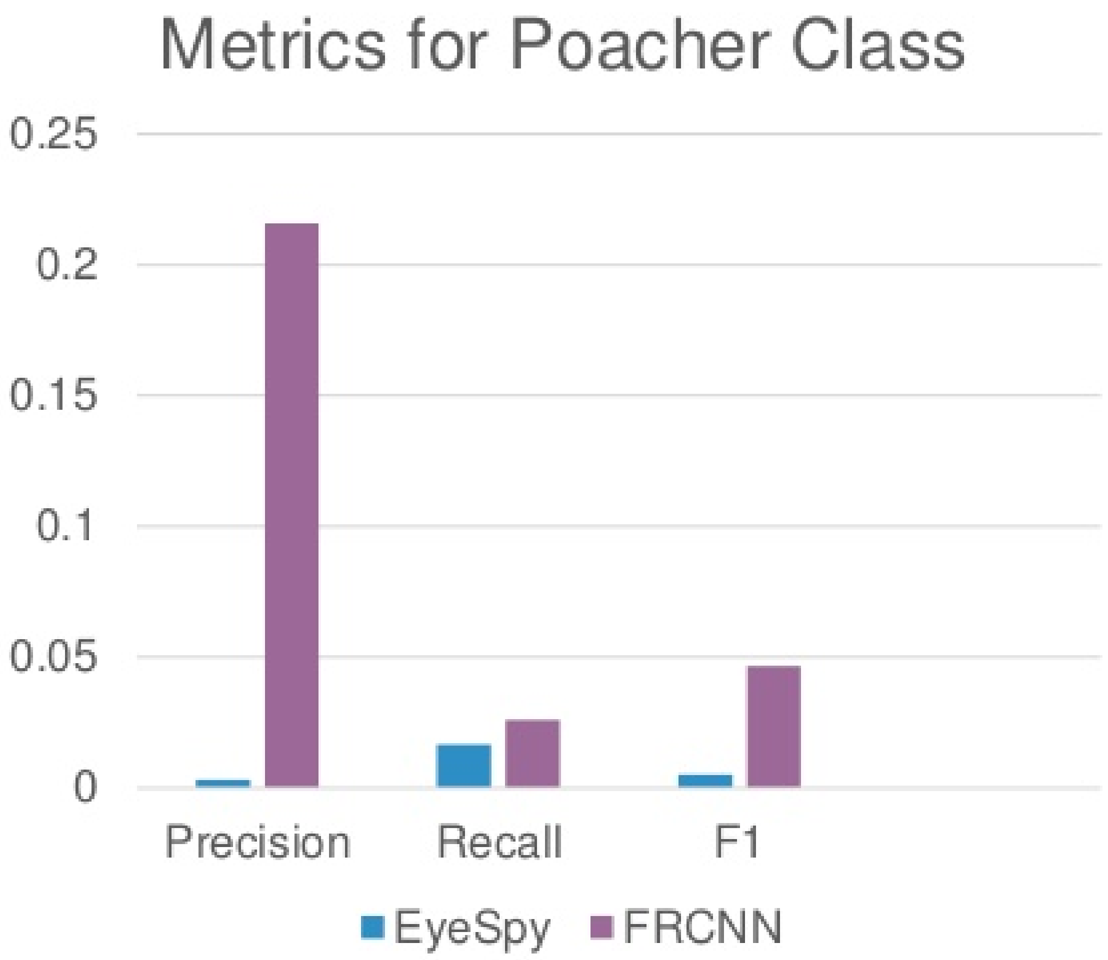
Step 4

- A convolutional network emits the class of the object and the bounding box “adjustments”



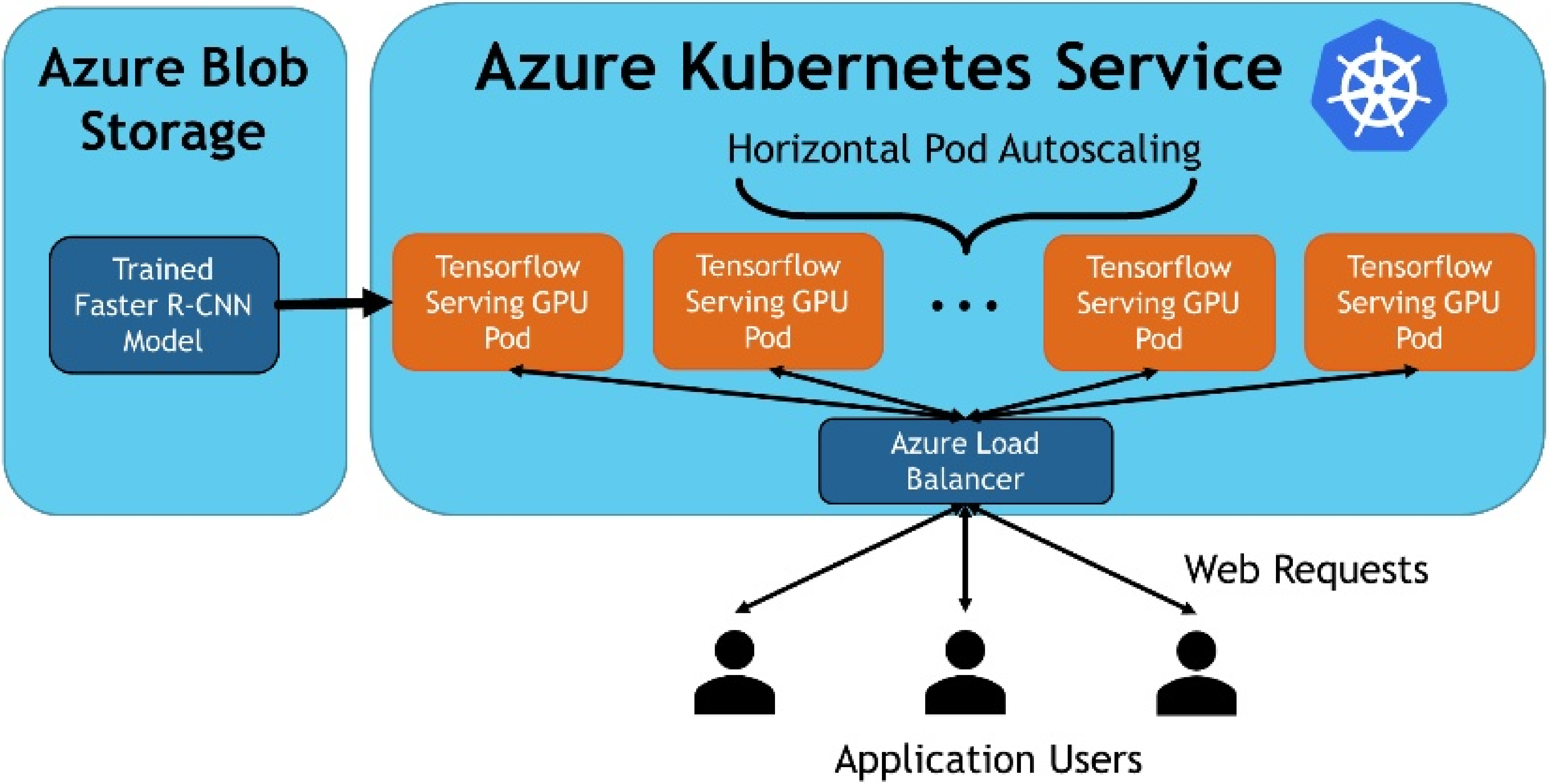
Results





"*AirSim-W: A Simulation Environment for Wildlife Conservation with UAVs*" – E. Bondi et al

Deployment of SPOT

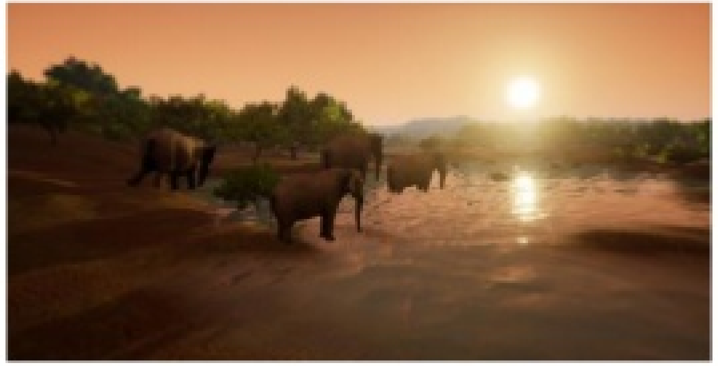


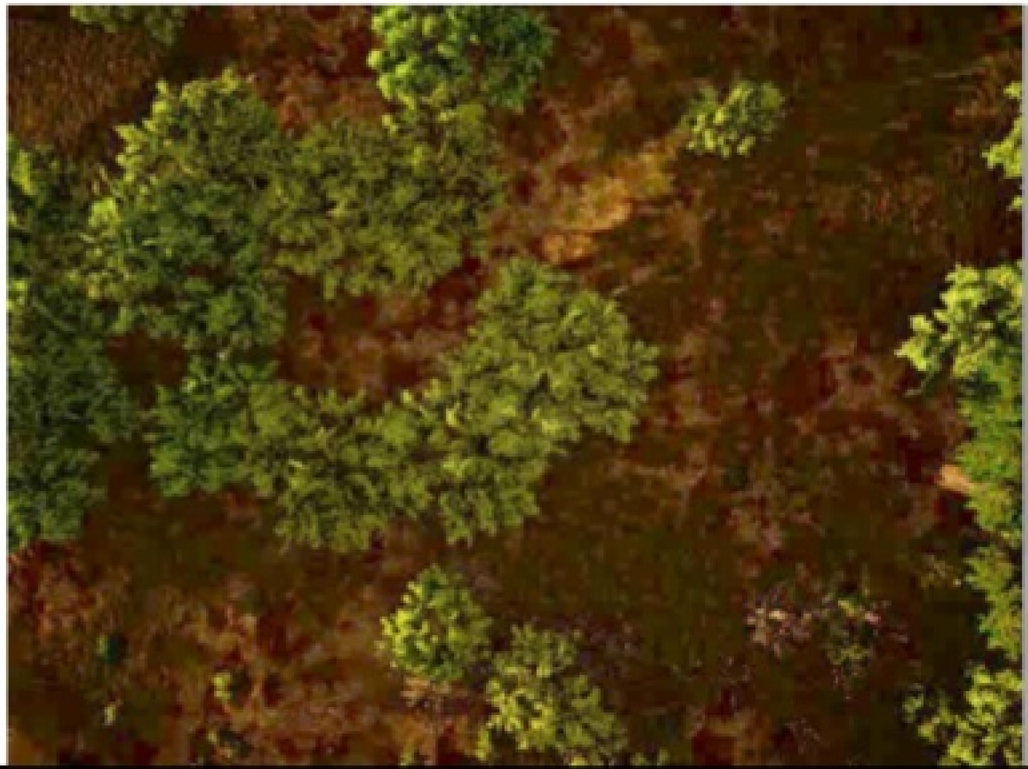
Sidestepping Labeling Woes with AirSim

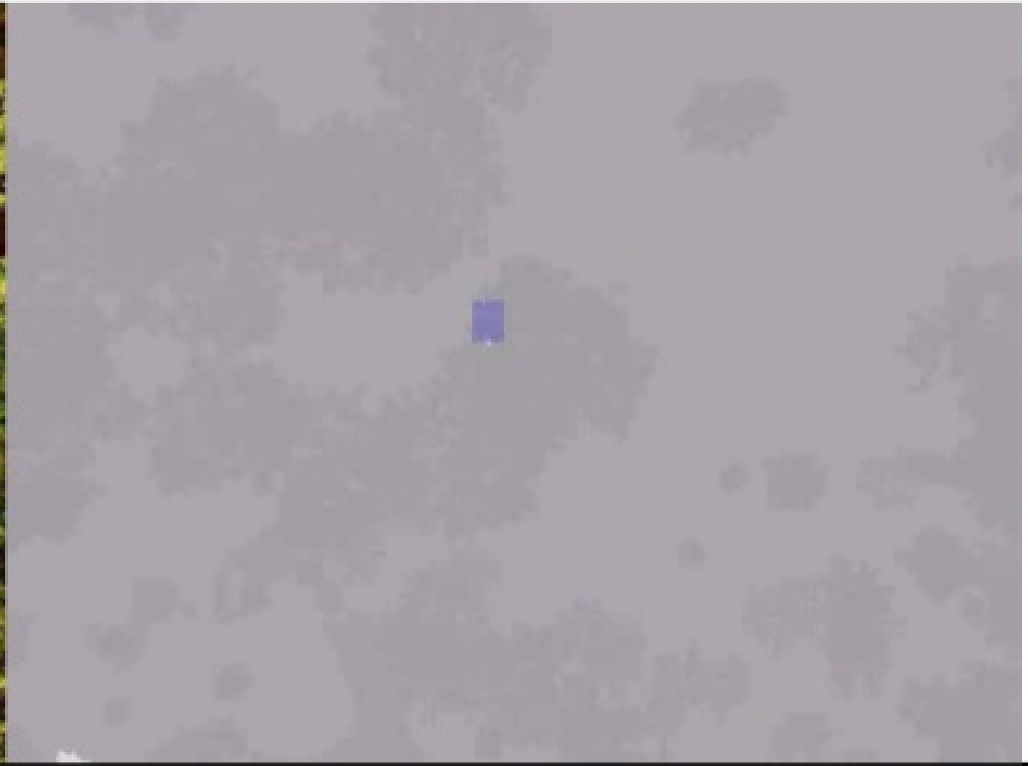
- 70 videos (~180,000 individual bounding boxes, 39,380 frames) labeled with VIOLA took **6 months**
- Goal: Bypass the need for human labelling

Shah, S., Dey, D., Lovett, C., & Kapoor, A. (2017). AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics* (pp. 621-635). Springer, Cham.

AirSim-W: Africa Environment







"*AirSim-W: A Simulation Environment for Wildlife Conservation with UAVs*" – E. Bondi et al

Results

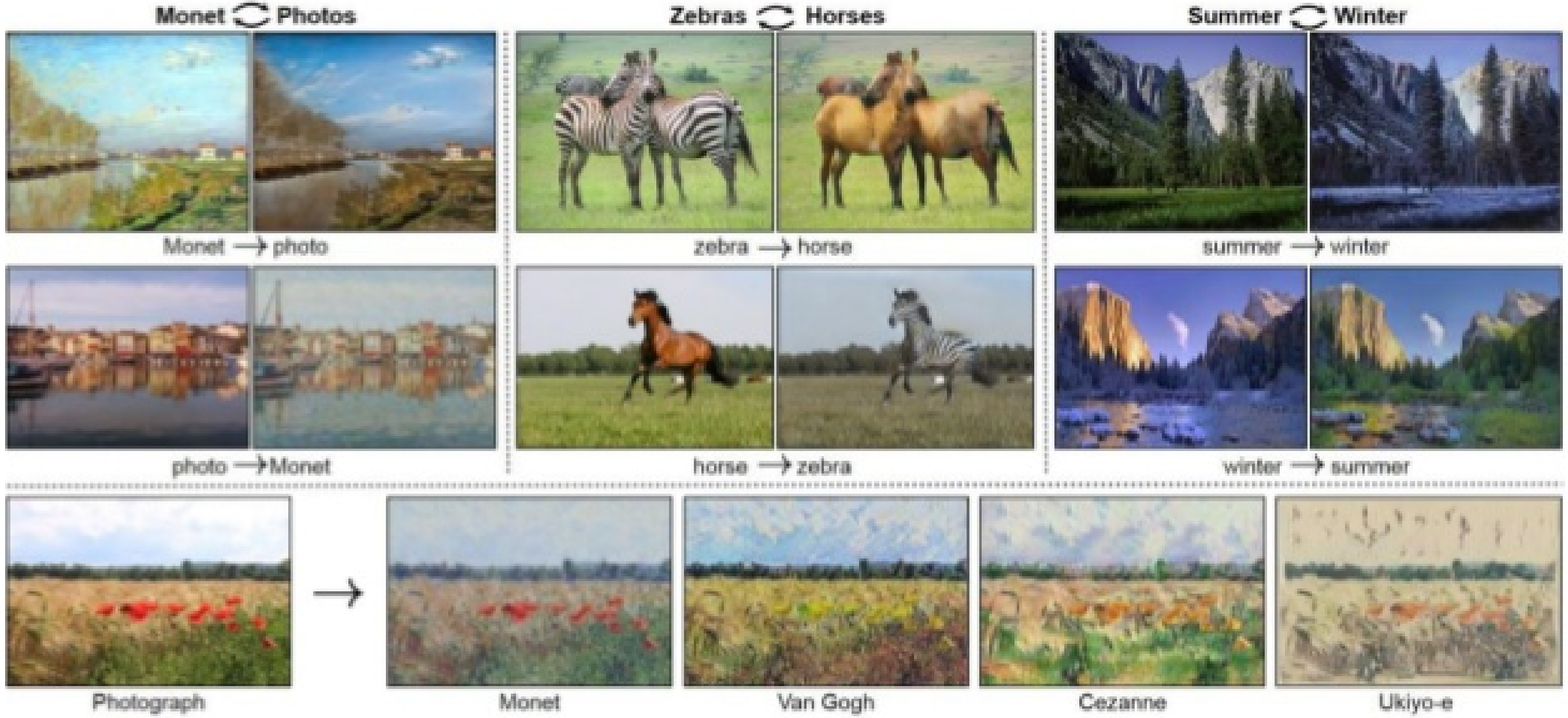
	Precision			
	EyeSpy	Only Real	Only Sim.	Sim. + Real
Animals	0.0128	**0.5463**	0.0202	0.5352
Poachers	0.0031	0.2160	0.0051	**0.3329**

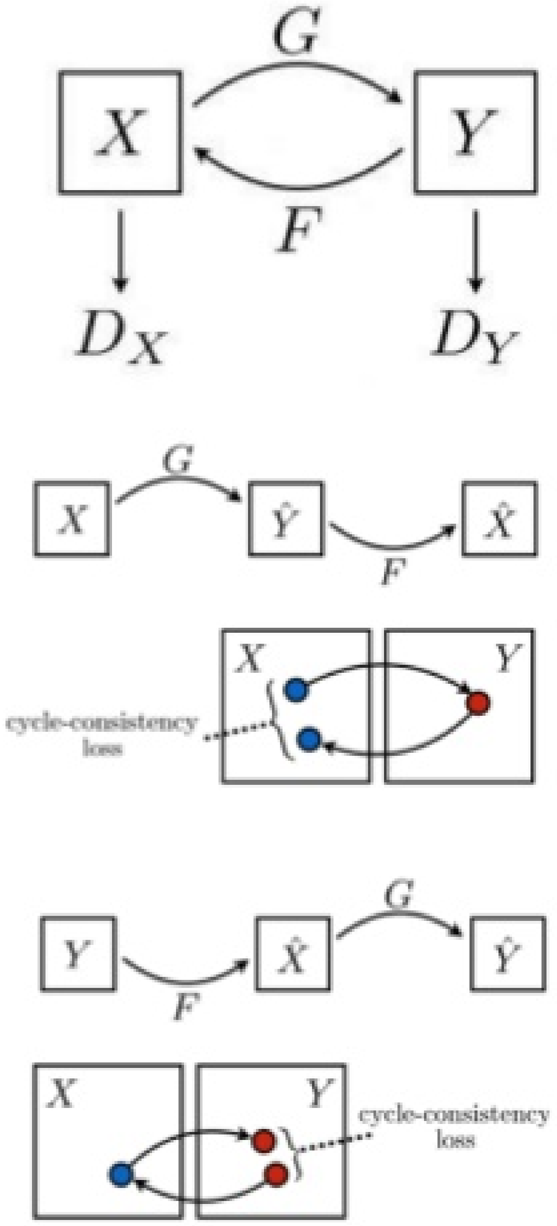
Recall			
EyeSpy	Only Real	Only Sim.	Sim. + Real
0.0839	0.2316	0.1458	**0.3207**
0.0167	0.0261	0.0038	**0.0485**

F1			
EyeSpy	Only Real	Only Sim.	Sim. + Real
0.0222	0.3253	0.0355	**0.4011**
0.0052	0.0466	0.0044	**0.0847**

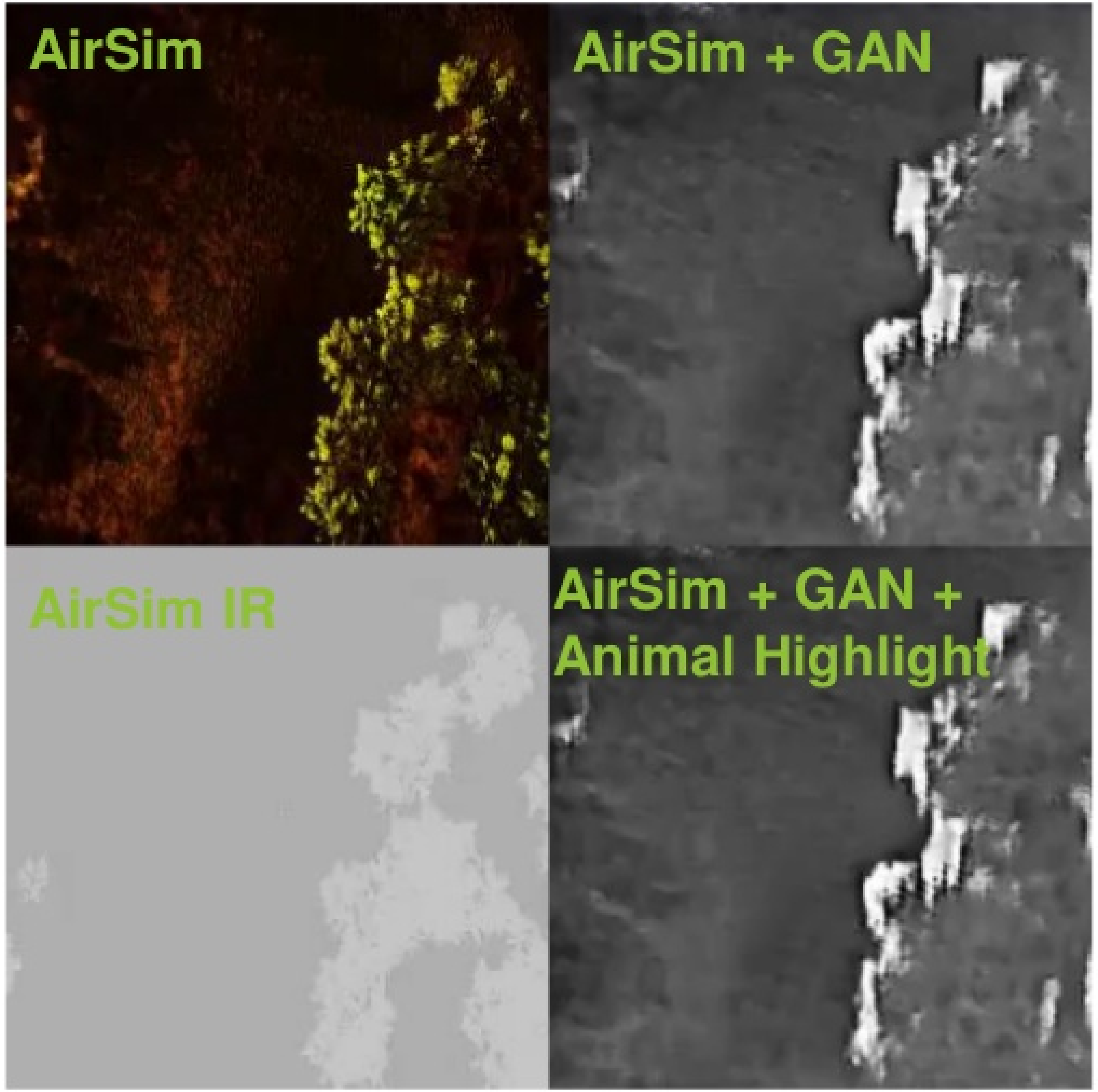
"*AirSim-W: A Simulation Environment for Wildlife Conservation with UAVs*" – E. Bondi et al

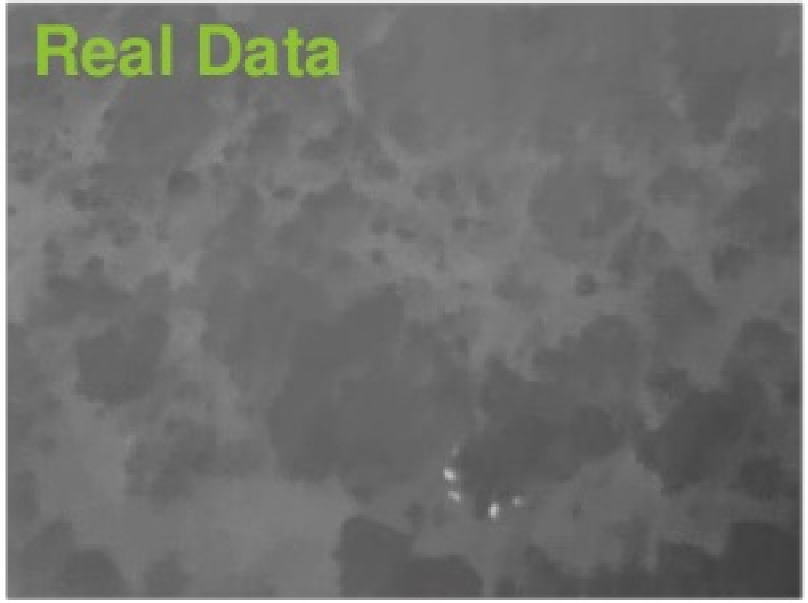
Deep Domain Adaptation

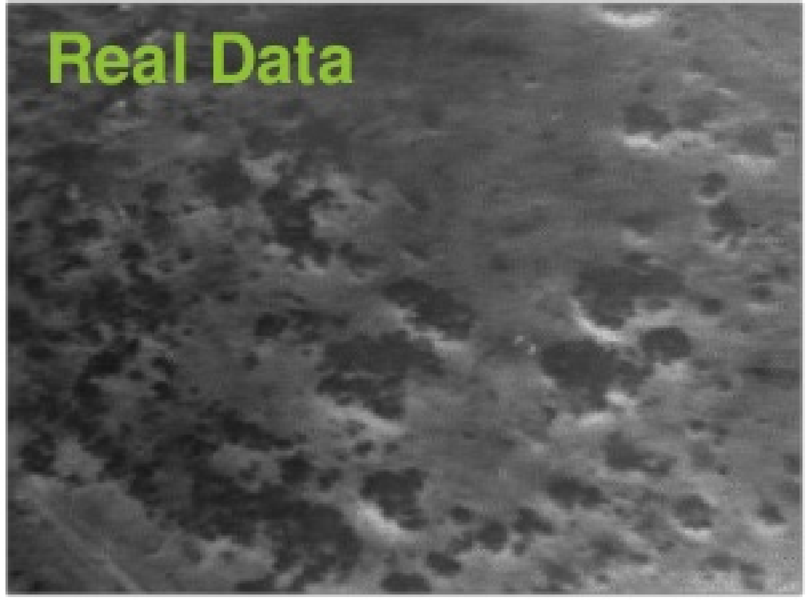




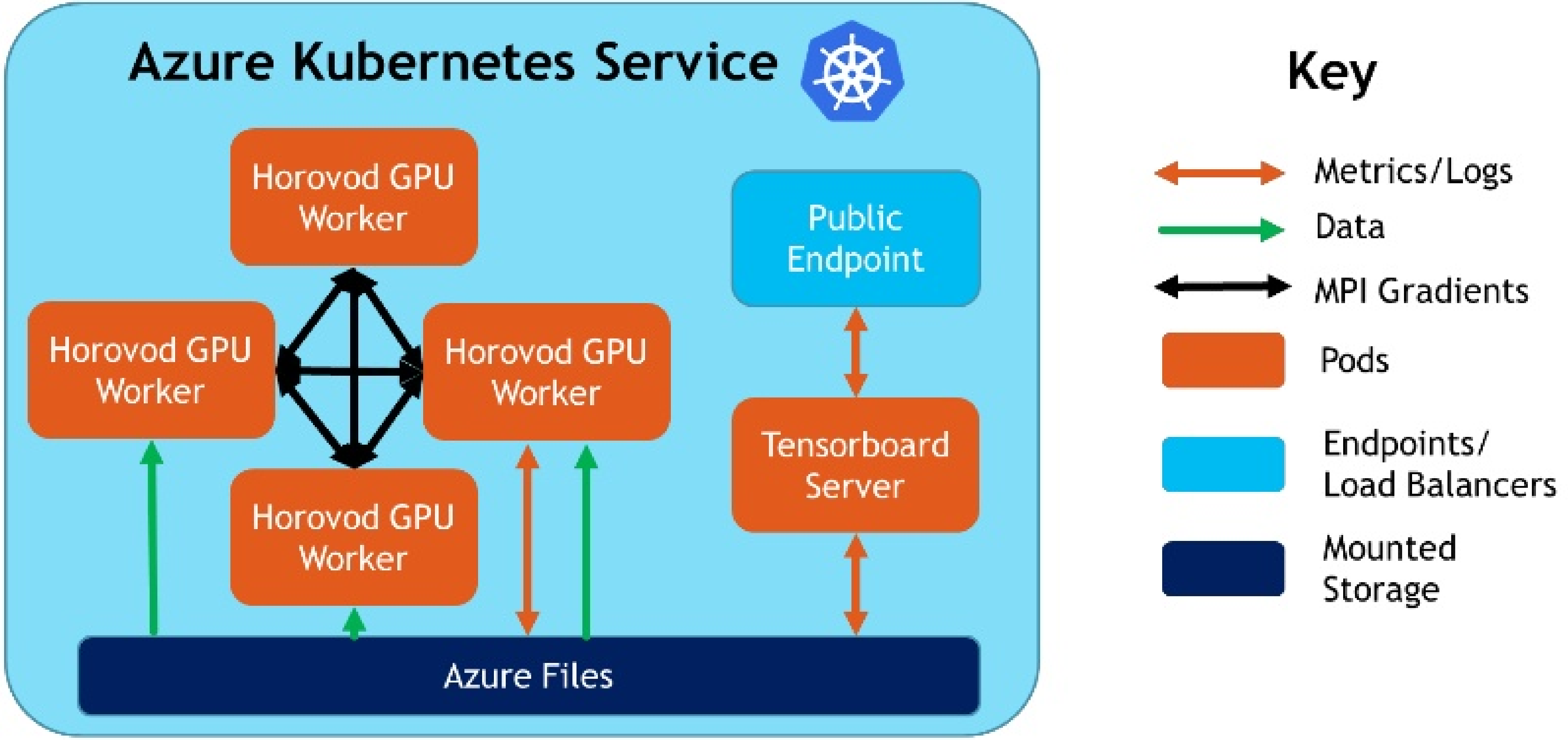
"*Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*" Jun-Yan Zhu*, Taesung Park*, Phillip Isola, Alexei A. Efros
Berkeley AI Research Lab, UC Berkeley


AirSim
AirSim + GAN
AirSim IR
AirSim + GAN +
Animal Highlight


Real Data


Real Data

Distributed Training Architecture



Easy Deployment with HELM

- Helm acts as a "package manager" for Kubernetes
- Easily deploy and share recipes for complex architectures
  - Uses parameters for easy CLI configuration
- Works on any Kubernetes Cluster (AKS, ACI, On-Prem)
- Open Source "Helm Chart" Coming Soon!

```
charts/horovod$ helm install ./horovod -n cycle-gan
```

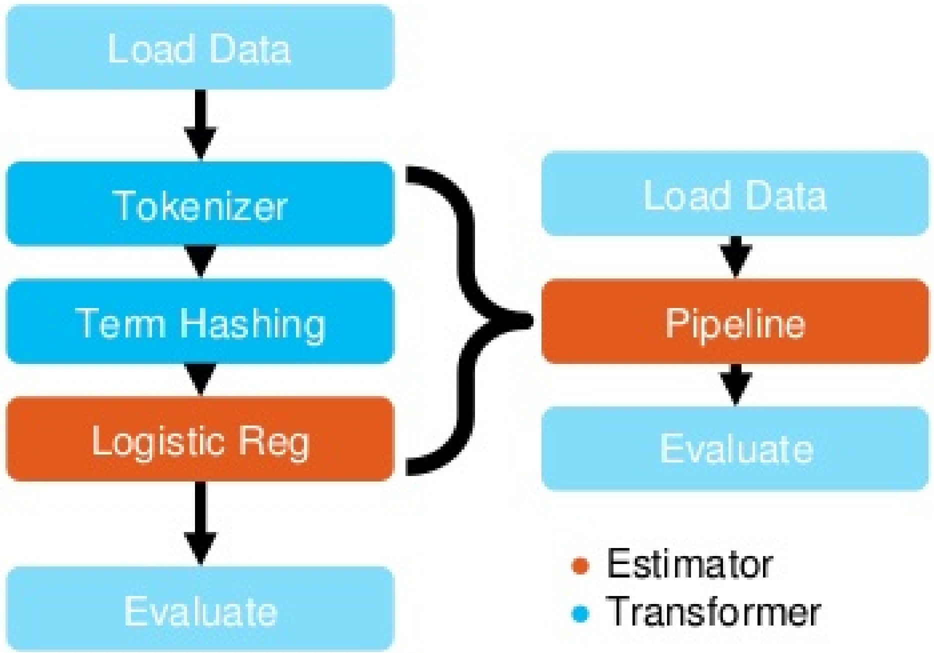
Challenges

- No unified procedures for storing data
- Serving and training architectures fundamentally different
  - Inconsistent APIS
  - Inconsistent architectures (different base pods)
- No single point of entry to run entire workflow

Apache Spark ML

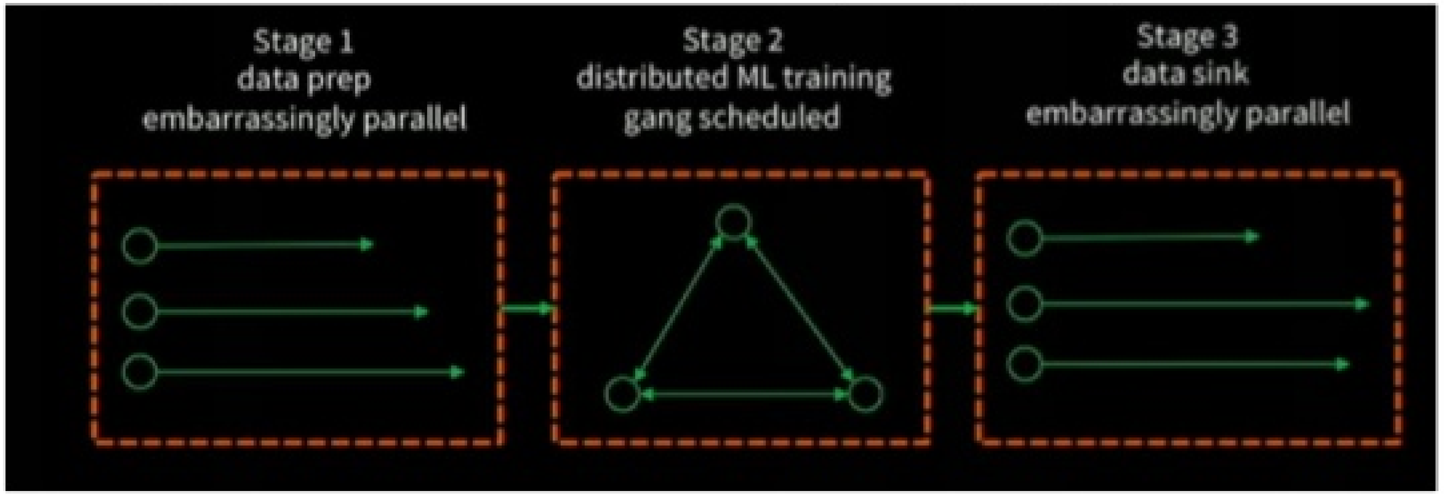
```
data = spark.read.csv("hdfs://...")
train, test = data.randomSplit([.5,.5])
model = LogisticRegression().fit(train)
predictions = model.transform(test)
```

- High level library for distributed machine learning
- More general than SciKit-Learn
- All models have a uniform interface
  - Can compose models into complex pipelines
  - Can save, load, and transport models



Horovod on Spark

- High level SparkML API for launching distributed deep learning jobs on the spark cluster
- Built on Project Hydrogen: Gang Scheduled Job
- Enables integration with data prep pipeline on Spark

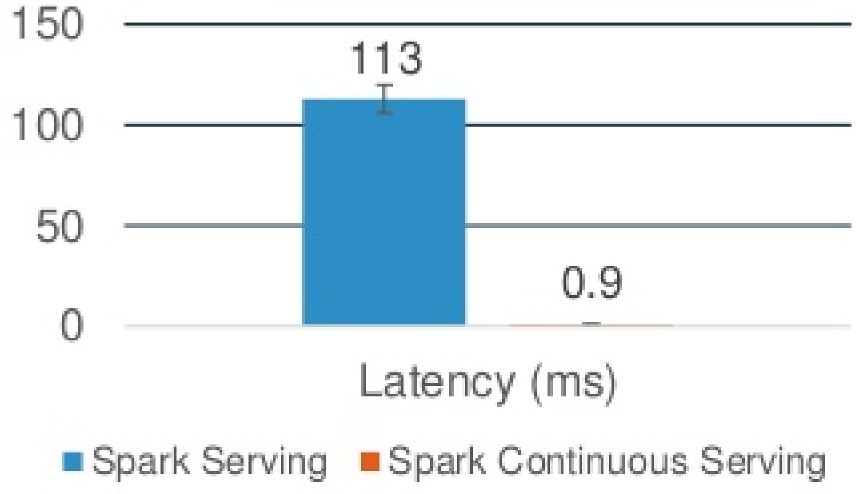


Apache Spark Serving

Lightning Fast Web Services on ***Any*** Spark Cluster

- Sub-millisecond latencies
- Fully Distributed
- Spins up in seconds
- Same API as Batch and Streaming
- Scala, Python, R and Java
- Fully Open Source

Announcing: **100x Latency Reduction** with MMLSpark v0.14



www.aka.ms/spark

JIRA: SPARK-25350

The Spark Serving API

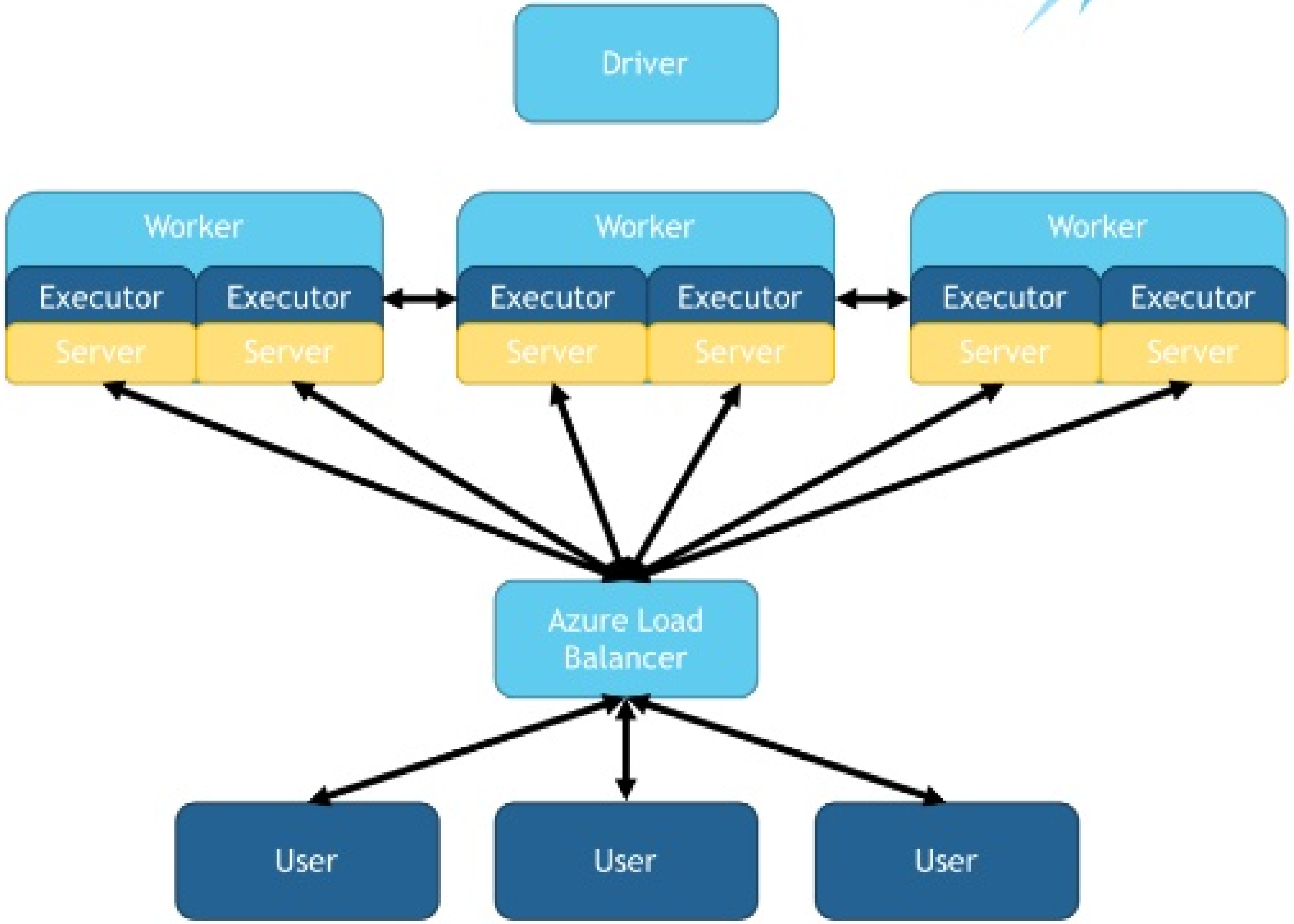
```
# Create a new "Serving" Dataframe
serving_df = (spark.readStream.server()
  .address("0.0.0.0", 8889, "danger_detector").load()
  .parseRequest(BinaryType())
  .withColumnRenamed("bytes", "image"))

# Manipulate your dataframe using ANY Spark/SparkML operations
replies = fitModel.transform(serving_df).makeReply("probability")

# Write your serving dataframe to the api to reply
server = (replies.writeStream.server()
  .replyTo("danger_detector")
  .queryName("my_service")
  .start())
```

Spark Serving Architecture

- 3 main modes:
  - server: 1 service on the head node
  - distributedServer: 1 service per executor
  - continuousServer: 1 service per partition
- Built on top of Spark Streaming
- Each worker keeps a running service, and a routing table
- Use HTTP on Spark objects to represent requests and responses in Spark SQL



Ideal Architecture

Azure Kubernetes Service or Azure Databricks

APACHE Spark™

Preprocess Image Data with OpenCV on Spark → Train the CycleGAN with Horovod on Spark → Train FasterRCNN with Horovod on Spark → Deploy FasterRCNN as a Web Service with Spark Serving

In Conclusion

- Big Releases in MMLSpark v0.14:
  - Sub-Millisecond latency distributed web services
  - 2 more large announcements in our next Session!
    - [Semi-Supervised Object Detection Using the Azure Cognitive Services on Spark](): **Oct 3, 14:00**
- We aim to give all work back to the community!
- Easy to Get Started on Databricks:
  - 16 Jupiter notebook guide

MMLSpark v0.14

www.aka.ms/spark

Contributions Welcome!
github.com/Azure/mmlspark

Thanks to

- You all!
- MMLSpark Team: Sudarshan Raghunathan, Ilya Matiach, Eli Barzilay, Tong Wen, Ben Brodsky
- Microsoft AI Development Acceleration Program: Abhiram Eswaran, Ari Green, Courtney Cochrane, Janhavi Suresh Mahajan, Karthik Rajendran, Minsoo Thigpen, Casey Hong, Soundar Srinivasan
- University of Southern California: Elizabeth Bondi, Millind Timbe
- Microsoft: Joseph Sirosh, Lucas Joppa, WeeHyong Tok

MMLSpark Website: aka.ms/spark

Get in touch: marhamil@microsoft.com