



AI & Machine Learning @ AWS

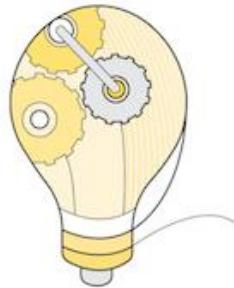
An Introduction

Daniel ZivKovic, Cloud Solutions Architect
TriNimbus, an Onica company
Office Lunch & Learn, September 28, 2018



TODAY WE WILL COVER

1. WHY LISTEN
TO AMAZON?



2. MACHINE
LEARNING
OVERVIEW



3. AWS ML STACK:
APPLICATION
SERVICES



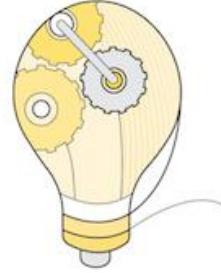
4. AWS ML STACK:
PLATFORM SERVICES



5. KEEP LEARNING



1. WHY LISTEN TO AMAZON?





Two Decades of Machine Learning at Amazon.com



UI – PERSONALIZED RECOMMENDATIONS

The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses Hardcover – September 13, 2011

by Eric Ries • (Author)
★★★★★ • 2,082 customer reviews
#1 Best Seller in New Business Enterprises

See all 7 formats and editions

Kindle \$10.05	Hardcover \$15.62	Paperback \$10.52	Audiobook \$0.00
Read with Our Free App	99 Used from \$9.00	33 Used from \$10.52	Free with your Audible trial
	50 New from \$14.09	19 New from \$11.49	
	6 Collectible from \$10.46		

Arrives before Christmas. Choose delivery option in checkout.

Most startups fail. But many of those failures are preventable. The Lean Startup is a new approach being adopted across the globe, changing the way companies are built and new products are launched.

Eric Ries defines a startup as an organization dedicated to creating something new under conditions of extreme uncertainty. This is just as true for one person in a garage or a group of seasoned professionals in a Fortune 500 boardroom. What they have in common is a mission to

Read more

Spend \$15 on books, get \$5 off
Use code: BOOKGIFT17
Learn more

Frequently bought together

THE LEAN STARTUP + ERIC RIES THE STARTUP WAY + PRINCIPLES: LIFE AND WORK	Total price: \$51.38
Add all three to Cart	
Add all three to List	

This item: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful... by Eric Ries Hardcover \$15.62

The Startup Way: How Modern Companies Use Entrepreneurial Management to Transform Culture and Drive... by Eric Ries Hardcover \$17.77

Principles: Life and Work by Ray Dalio Hardcover \$17.99

Customers who bought this item also bought

THE STARTUP WAY	THE FOUNDER'S DILEMMA	THE STARTUP OWNER'S MANUAL	ZERO TO ONE	THE INNOVATOR'S DILEMMA	THE HARDEST THING ABOUT HARD THINGS	LEAN ANALYTICS	CROSSING THE CHASM	RUNNING LEAN	BUSINESS MODEL GENERATION
ERIC RIES	NOAM WISEMAN	STEVE BLANK	NOAM WISEMAN	CLAYTON M. CHRISTENSEN	ROBERT H. SOUTHERN	ALISTAIR CROLL	GEORGE GARDNER	JEFF HORN	ALEX OSTERWALDER
Hardcover	Hardcover	Hardcover	Hardcover	Hardcover	Hardcover	Hardcover	Hardcover	Hardcover	Hardcover
\$17.77 .prime	\$13.04 .prime	\$26.15 .prime	\$18.84 .prime	\$26.64 .prime	\$19.89 .prime	\$25.65 .prime	\$12.48 .prime	\$16.64 .prime	\$26.05 .prime

Page 1 of 10

Amazon.com Recommendations – Item-to-Item Collaborative Filtering algorithm, did to Barnes & Noble what Google PageRank algorithm did to Yahoo's search dominance – which was based on professional web surfers curating the web.

– Jeremy Howard

ALEXA – AI ENABLED #VoiceFirst UI



With voice, for the first time in human evolution ever – we can operate “tools” out of our hand’s reach!

– Paul Cutsinger

Alexa was my “wake-up call” for AI, as before it no speech recognition program could handle my thick accent, hoarseness, and pace of speech.

– Daniel ZivKovic

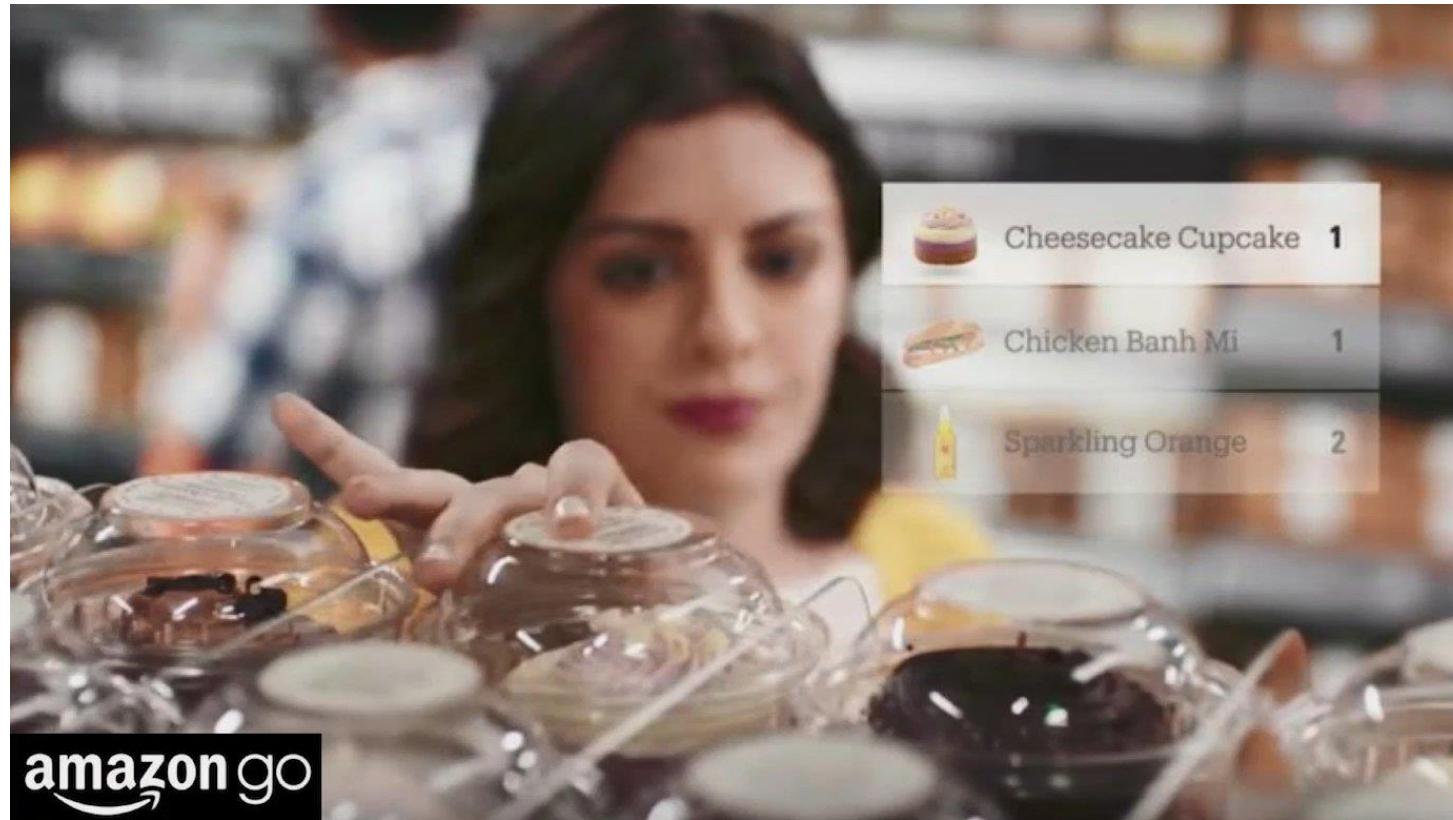
50K ROBOTS ACROSS DELIVERY CENTRES



DRONES – ALGOS LIKE IN SELF-DRIVING CARS



AMAZON GO – NO CHECKOUT, NO CASHIERS



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



ML @ AWS MISSION

Put Machine Learning in the
hands of **every** developer
and data scientist

This mission statement feels as big as Bill Gates' - "A computer on every desk and in every home", or Steve Jobs' - "1000 Songs In Your Pocket" mission statements before.

AWS ML STACK

APPLICATION SERVICES

Amazon Rekognition

Amazon Rekognition Video

Amazon Polly

Amazon Transcribe

Amazon Lex

Amazon Translate

Amazon Comprehend

PLATFORM SERVICES

Amazon SageMaker

AWS DeepLens

Amazon EMR

FRAMEWORKS AND INTERFACES

Deep Learning
AMI

Apache MXNet

Caffe2

CNTK

PyTorch

TensorFlow

Theano

Torch

Keras

Gluon

ML USE CASES BY VERTICALS



MANUFACTURING

- Preventative maintenance
- Demand prediction
- Defect detection



FINANCE

- Fraud detection
- Customer UX/UI experience optimization
- Loan risk assessment



RETAIL

- Targeting
- Recommendation engines
- Dynamic pricing



HEALTHCARE

- Preventative care
- Diagnosis in medical imagery
- Drug discovery



OIL & GAS

- Reservoir analysis and modeling
- Alarm management and investigation
- Well test analysis and prediction



AGRICULTURE

- Predict beneficial plant genes
- Livestock IoT efficiency / optimization
- Crop health monitoring

2. MACHINE LEARNING OVERVIEW



DEFINITIONS

ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.



MACHINE LEARNING

Machine learning begins to flourish.

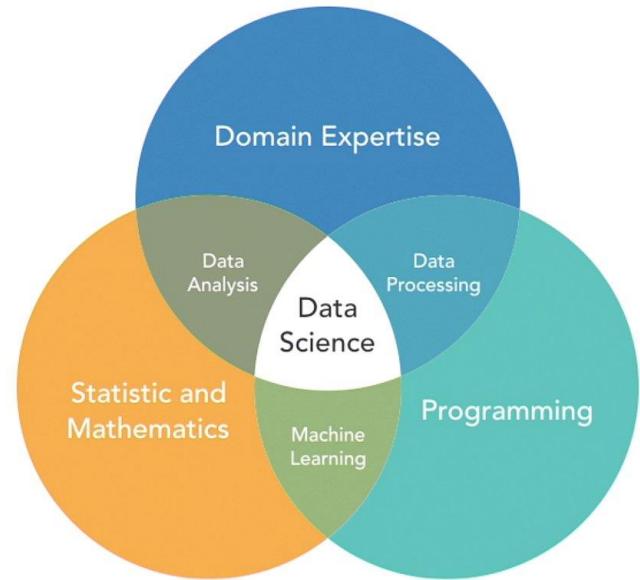
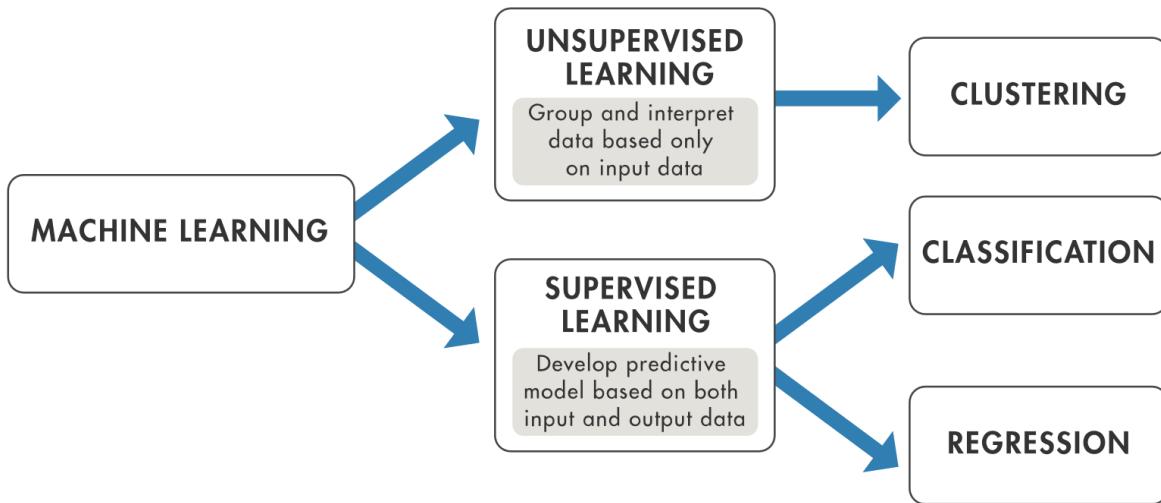


DEEP LEARNING

Deep learning breakthroughs drive AI boom.



MORE DEFINITIONS

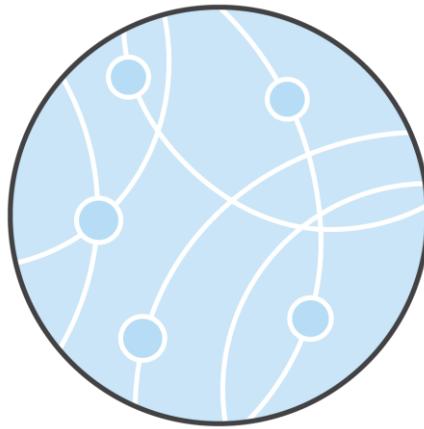


- Unsupervised – **clustering** finds hidden patterns in your data; draws inferences (conclusions) from datasets without labeled data
- Supervised – **regression** predicts future value of continuous variable like temperature or stock price
- Supervised – **classification** predict the likelihood of belonging to a specific group (e.g. cat or a dog)
- Supervised – most commonly includes algorithm, plus **labeled data**
- Data Science...

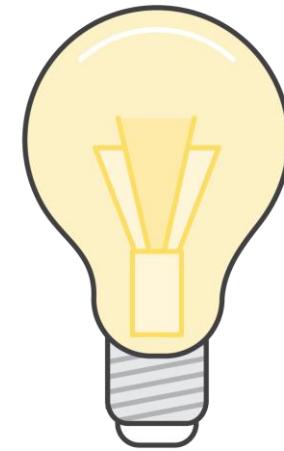
OVERVIEW OF DEEP LEARNING



Data

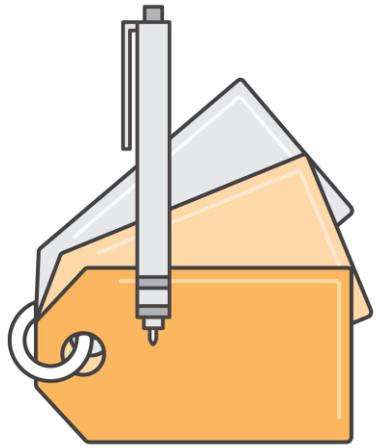


Model training

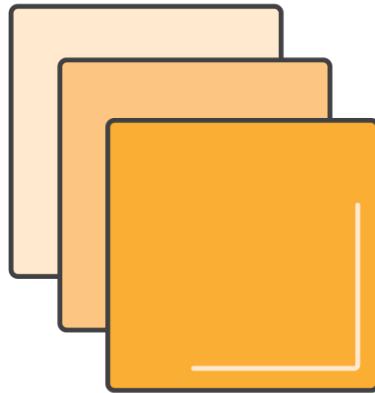


Inference

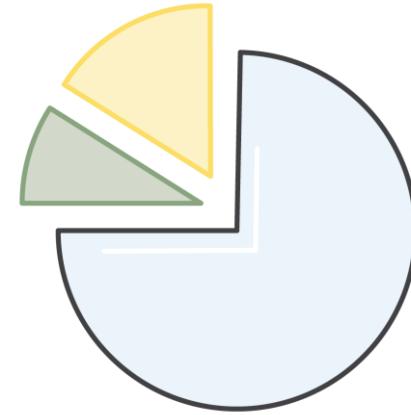
DATA



Annotate
Labeling



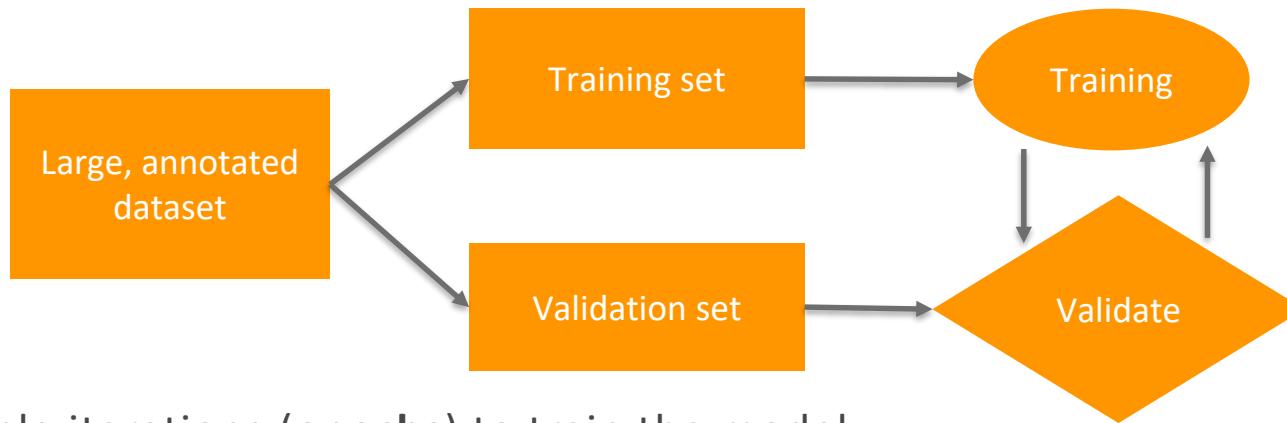
Pre-process
Sizing, formatting...



Data split
Training set vs
Validation set

MODEL DEVELOPMENT & TRAINING

- Define model architecture
- Input the annotated and cleaned data into the model



- Multiple iterations (**epochs**) to train the model
- Validate with held back dataset

INFERENCE

It's where the magic happens!

1. Preprocess **new** data/image just like training set.
2. Feed image back to the trained model to get a predicted output.



THE OLD WAY VS. THE NEW WAY

**SELECT c.cust_name, o.order_id, p.product_name
FROM customers c
JOIN orders o
ON c.ID = o.customer_id
LEFT JOIN products p
ON o.ID = o.product
GROUP BY c.ID
HAVING o.category = 'Toys'
((p.description LIKE '%HELICOPTER%'
AND o.date > GETDATE() - 30)
(COUNT(*) > 2
AND SUM(o.price)
> date > GETDATE() - 30)
)**

Use machine learning technology to learn your business rules from data!

If you want a computer to do something, the old way to do it is to write a program. The new way is you tell the computer to pretend to be in neural network with a learning algorithm in it - that's programming, but then after that, if you want to solve a particular problem you just show it examples.

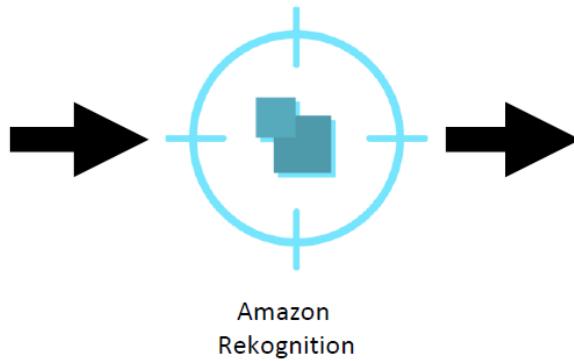
– Geoffrey Hinton



4. AWS ML STACK: APPLICATION SERVICES



AMAZON RECOGNITION



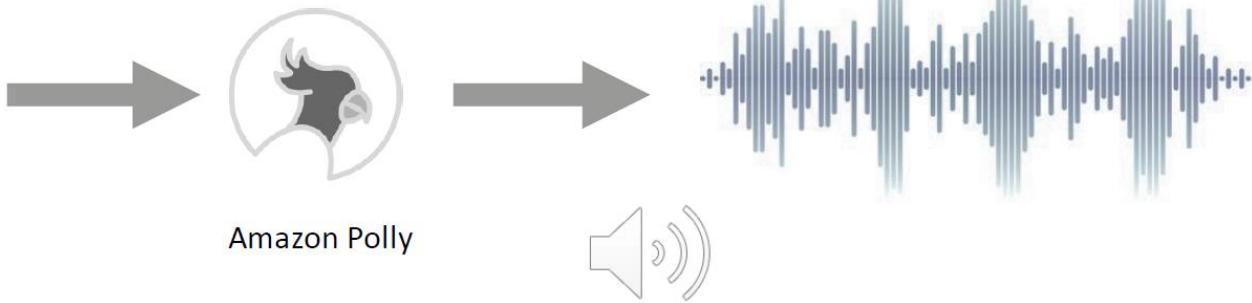
Object and scene detection
Facial analysis
Face comparison
Celebrity recognition
Image moderation
Text in image

AMAZON RECOGNITION VIDEO



AMAZON POLLY

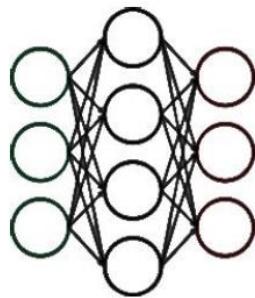
“Hejsan! Jag heter
Astrid och läser upp
det som skrivs här.”



AMAZON TRANSLATE



Real-time translation



Powered by Deep
Learning

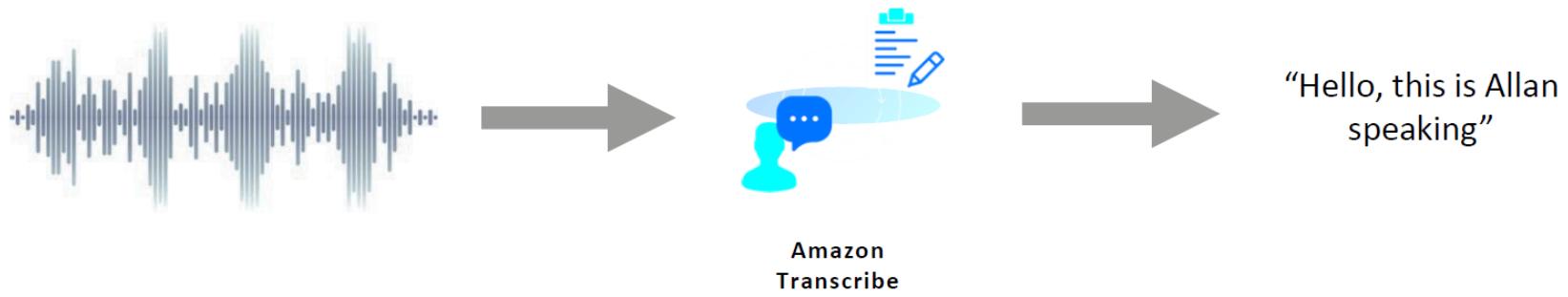


12 Language pairs
(more to come)



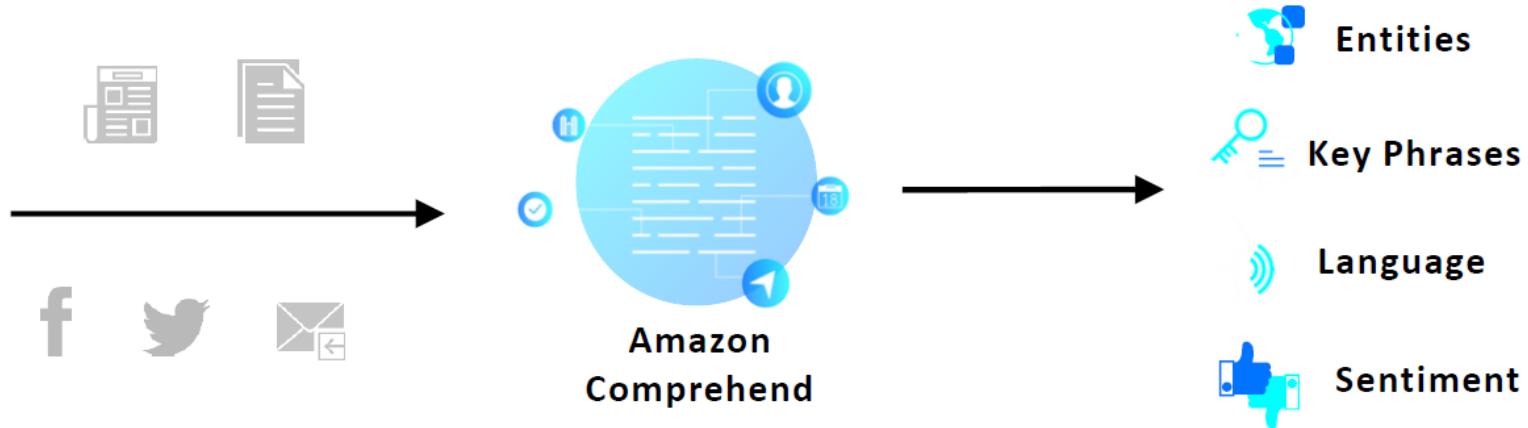
Language detection

AMAZON TRANSCRIBE

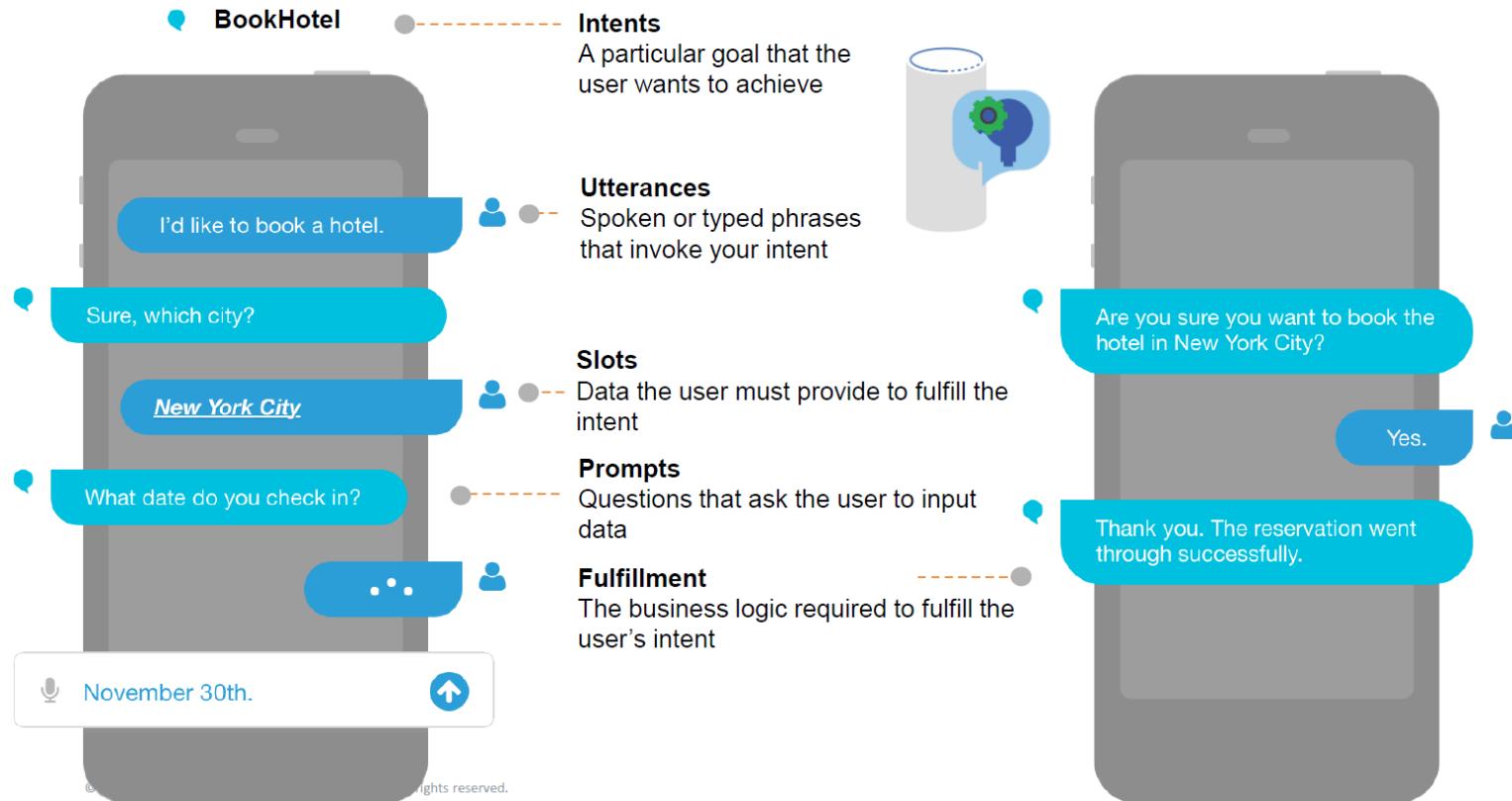


AMAZON COMPREHEND

Discover valuable insights from text



AMAZON LEX



GO EXPERIMENT

“No Shame in Console Game”
– Abby Fuller

4. AWS ML STACK: PLATFORM SERVICES



AWS ML STACK

APPLICATION SERVICES

Amazon Rekognition

Amazon Rekognition Video

Amazon Polly

Amazon Transcribe

Amazon Lex

Amazon Translate

Amazon Comprehend

PLATFORM SERVICES

Amazon SageMaker

AWS DeepLens

Amazon EMR

FRAMEWORKS AND INTERFACES

Deep Learning
AMI

Apache MXNet

Caffe2

CNTK

PyTorch

TensorFlow

Theano

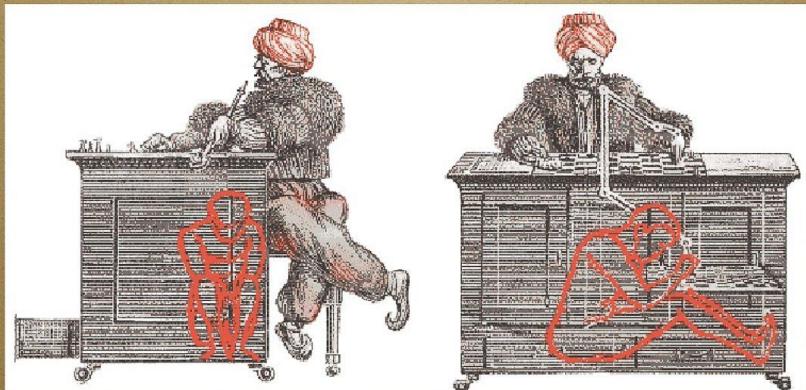
Torch

Keras

Gluon

PLUS: AMAZON MECHANICAL TURK

Peeking into the Machine



and Finding it Full of People

Pic source: <http://www.codinghorror.com/blog/archives/000828.html>

It is “artificial Artificial Intelligence” also counted as a Platform Service:
<https://docs.aws.amazon.com/mturk/>

Crowdsourced labor directly into applications using an API to complete jobs that humans can do better than computers.

AMAZON ELASTIC MAP REDUCE (EMR)



Amazon EMR



Spark 1.2.2 Overview Programming Guides API Docs Deploying More

Machine Learning Library (MLlib) Programming Guide

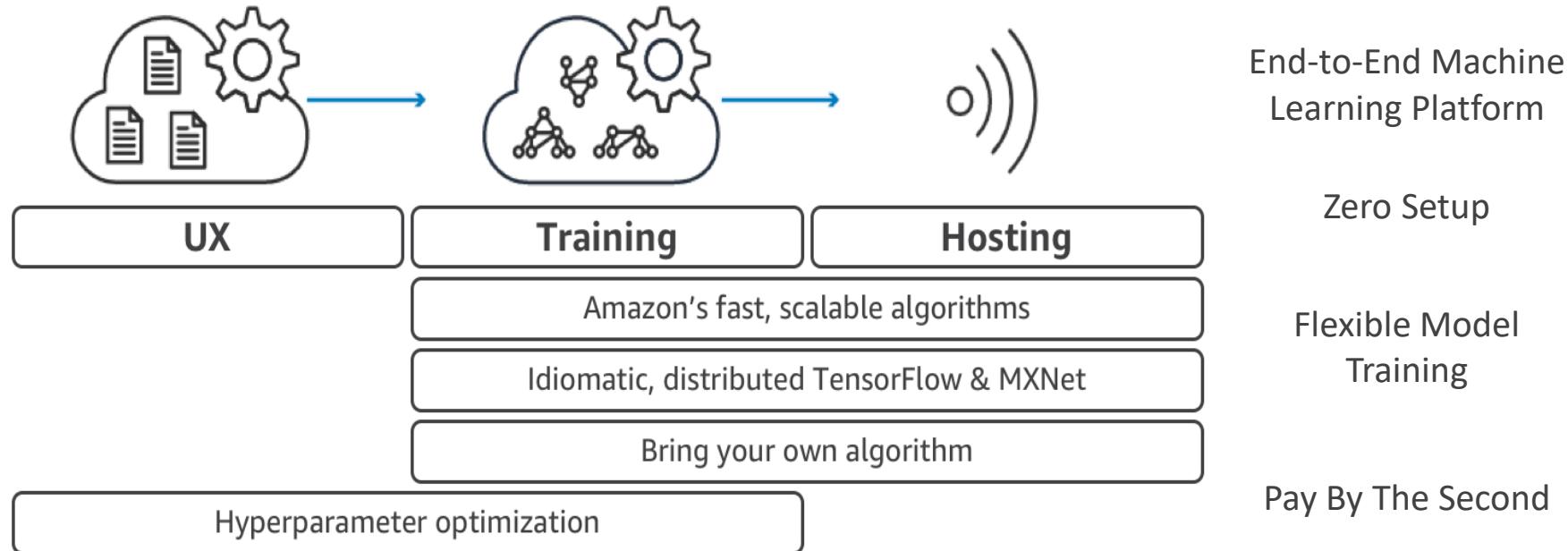
MLlib is Spark's scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives, as outlined below:

- [Data types](#)
- [Basic statistics](#)
 - summary statistics
 - correlations
 - stratified sampling
 - hypothesis testing
 - random data generation
- [Classification and regression](#)
 - linear models (SVMs, logistic regression, linear regression)
 - naive Bayes
 - decision trees
 - ensembles of trees (Random Forests and Gradient-Boosted Trees)
- [Collaborative filtering](#)
 - alternating least squares (ALS)
- [Clustering](#)
 - k-means
- [Dimensionality reduction](#)
 - singular value decomposition (SVD)
 - principal component analysis (PCA)
- [Feature extraction and transformation](#)
- [Optimization \(developer\)](#)
 - stochastic gradient descent
 - limited-memory BFGS (L-BFGS)

MLlib is under active development. The APIs marked Experimental/DeveloperApi may change in future releases, and the migration guide below will explain all changes between releases.

SAGEMAKER – FULLY-MANAGED PLATFORM

The quickest and easiest way to get ML models from idea to production



DEEPLENS



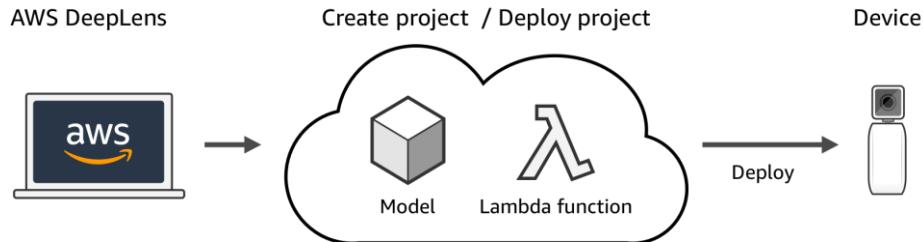
SPECIFICATIONS:

- Intel Atom Processor
- Gen9 graphics
- Ubuntu OS- 16.04 LTS
- 100 GFLOPS performance
- Dual band Wi-Fi
- 8 GB RAM
- 16 GB Storage (eMMC)
- 32 GB SD card
- 4 MP camera with MJPEG
- H.264 encoding at 1080p resolution
- 2 USB ports
- Micro HDMI
- Audio out
- AWS Greengrass preconfigured
- cIDNN Optimized for MXNet

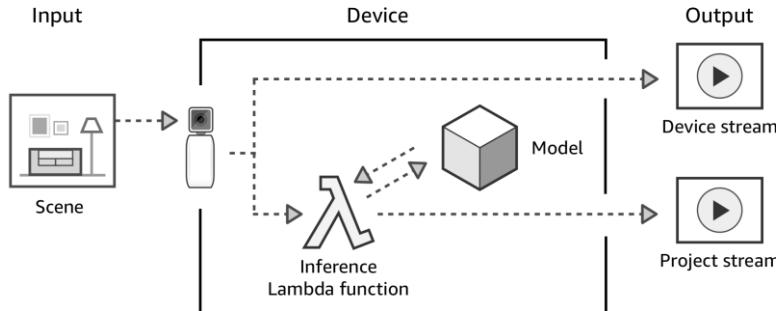


DEEPLENS – UNDER THE COVERS

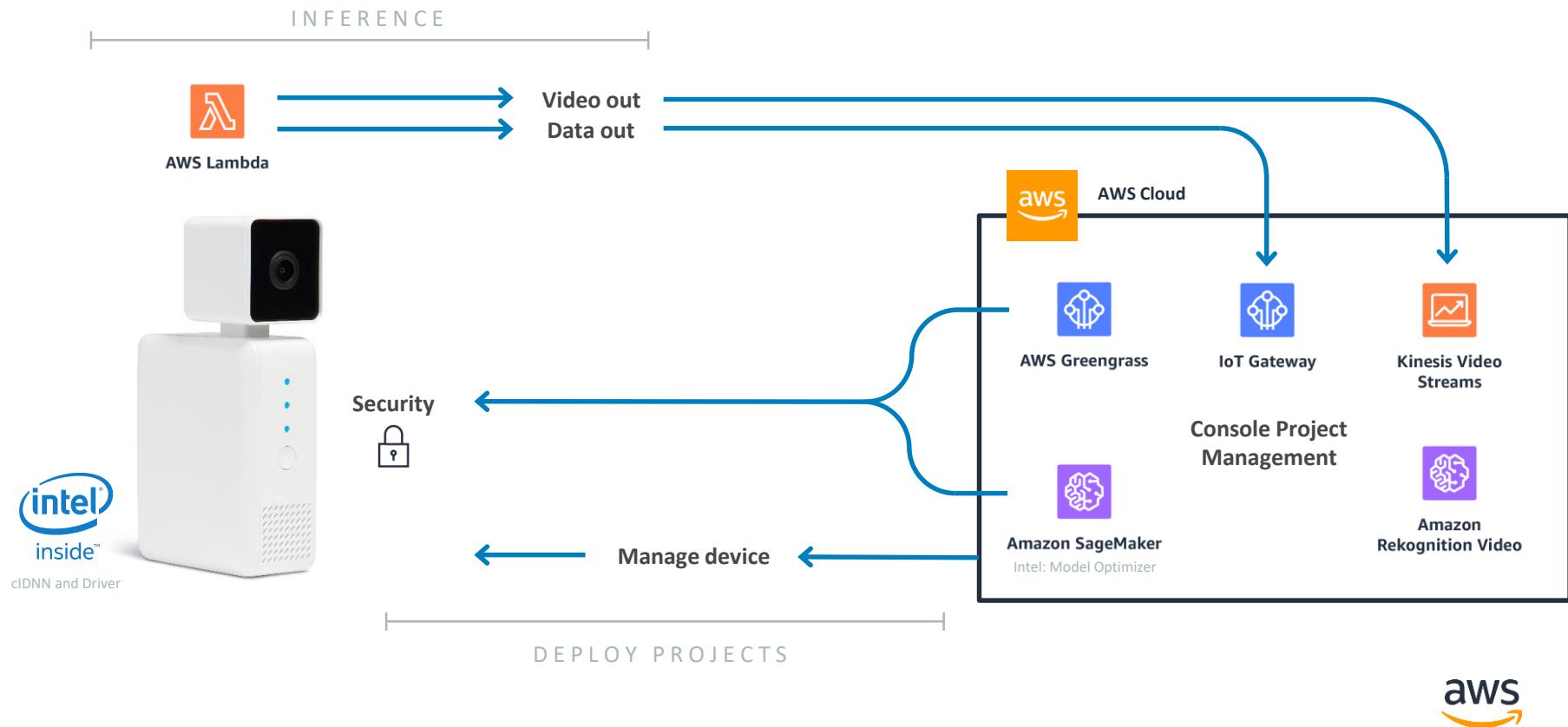
- Cloud to device



- On the device



AWS DEEPLENS ARCHITECTURE



GET STARTED WITH SAMPLE PROJECTS

HOT DOG / NOT HOT DOG



OBJECT DETECTION



FACE DETECTION



ACTIVITY DETECTION



HEAD POSE DETECTION



ARTISTIC STYLE TRANSFER



CAT VS. DOG



Or build **custom** deep learning models in the cloud using
Amazon SageMaker

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



ML BEHIND OBJECT DETECTION PROJECT

```
object-detection.py x
75 def greengrass_infinite_infer_run():
76     """ Entry point of the lambda function"""
77     try:
78         # This object detection model is implemented as single shot detector (ssd), since
79         # the number of labels is small we create a dictionary that will help us convert
80         # the machine labels to human readable labels.
81         model_type = 'ssd'
82         output_map = {1: 'aeroplane', 2: 'bicycle', 3: 'bird', 4: 'boat', 5: 'bottle', 6: 'bus',
83                       7: 'car', 8: 'cat', 9: 'chair', 10: 'cow', 11: 'dinning table',
84                       12: 'dog', 13: 'horse', 14: 'motorbike', 15: 'person',
85                       16: 'pottedplant', 17: 'sheep', 18: 'sofa', 19: 'train',
86                       20: 'tvmonitor'}
87
88         # Create an IoT client for sending messages to the cloud.
89         client = greengrassdk.client('iot-data')
90         iot_topic = '$aws/things/{}/infer'.format(os.environ['AWS_IOT_THING_NAME'])
91
92         # Create a local display instance that will dump the image bytes to a FIFO
93         # file that the image can be rendered locally.
94         local_display = LocalDisplay('480p')
95         local_display.start()
96
97         # The sample projects come with optimized artifacts, hence only the artifact
98         # path is required.
99         model_path = '/opt/awscam/artifacts/mxnet_deploy_ssd_resnet50_300_FP16_FUSED.xml'
100
101        # Load the model onto the GPU.
102        client.publish(topic=iot_topic, payload='Loading object detection model')
103        model = awscam.Model(model_path, {'GPU': 1})
104        client.publish(topic=iot_topic, payload='Object detection model loaded')
105
106        # Set the threshold for detection
107        detection_threshold = 0.25
108
109        # The height and width of the training set images
110        input_height = 300
111        input_width = 300
112
113        # Do inference until the lambda is killed.
114        while True:
115            # Get a frame from the video stream
116            ret, frame = awscam.getLastFrame()
117            if not ret:
118                raise Exception('Failed to get frame from the stream')
119
120            # Resize frame to the same size as the training set.
121            frame_resize = cv2.resize(frame, (input_height, input_width)) 300
122
123            # Run the images through the inference engine and parse the results using
124            # the parser API, note it is possible to get the output of doInference
125            # and do the parsing manually, but since it is a ssd model,
126            # a simple API is provided.
127            parsed_inference_results = model.parseResult(model_type,
128                                              model.doInference(frame_resize))
129
130            # Compute the scale in order to draw bounding boxes on the full resolution
131            # image.
132            yscale = float(frame.shape[0]/input_height)
133            xscale = float(frame.shape[1]/input_width)
134
135            # Dictionary to be filled with labels and probabilities for MQTT
136            cloud_output = {}
137
138            # Get the detected objects and probabilities
139            for obj in parsed_inference_results[model_type]:
140                if obj['prob'] > detection_threshold:
141                    # Add bounding boxes to full resolution frame
142                    xmin = int(xscale * obj['xmin']) \
143                          + int((obj['xmin'] - input_width/2) + input_width/2)
144                    ymin = int(yscale * obj['ymin'])
145                    xmax = int(xscale * obj['xmax']) \
146                          + int((obj['xmax'] - input_width/2) + input_width/2)
147                    ymax = int(yscale * obj['ymax'])
148
149                    # See https://docs.opencv.org/3.4.1/d6/d6e/group_imgproc_draw.html
150                    # for more information about the cv2.rectangle method.
151                    # Method signature: image, point1, point2, color, and thickness.
152                    cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), (255, 165, 20), 10) 4
153
154                    # Amount to offset the label/probability text above the bounding box.
155                    text_offset = 15
156
157                    # See https://docs.opencv.org/3.4.1/d6/d6e/group_imgproc_draw.html
158                    # for more information about the cv2.putText method.
159                    # Method signature: image, text, origin, font face, font scale, color,
160                    # and thickness
161                    cv2.putText(frame, "{}: {:.2f}%".format(output_map[obj['label']],
162                                              obj['prob'] * 100),
163                                (xmin, ymin-text_offset),
164                                cv2.FONT_HERSHEY_SIMPLEX, 2.5, (255, 165, 20), 6)
165
166                    # Store label and probability to send to cloud
167                    cloud_output[output_map[obj['label']]]= obj['prob']
168
169                    # Set the next frame in the local display stream.
170                    local_display.set_frame_data(frame)
171
172                    # Send results to the cloud
173                    client.publish(topic=iot_topic, payload=json.dumps(cloud_output))
174
175            except Exception as ex:
176                client.publish(topic=iot_topic, payload='Error in object detection lambda: {}'.format(ex))
177
178        greengrass_infinite_infer_run()
```

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



REUSABLE CODE

The image shows two code editors side-by-side. The left editor contains 'object-detection.py' and the right editor contains 'hotdog-detection.py'. Both files are annotated with yellow numbers 1 through 6 pointing to specific lines of code.

object-detection.py:

```
107     while True:
108         # Get a frame from the video stream
109         ret, frame = awscam.getLastFrame()
110         if not ret:
111             raise Exception('Failed to get frame from the stream')
112         # Resize frame to the same size as the training set.
113         frame_resize = cv2.resize(frame, (input_height, input_width))
114         # Run the images through the inference engine and parse the results using
115         # the parser API, note it is possible to get the output of doInference
116         # and do the parsing manually, but since it is a ssd model,
117         # a simple API is provided.
118         parsed_inference_results = model.parseResult(model_type,
119                                                       model.doInference(frame_resize))
120         # Compute the scale in order to draw bounding boxes on the full resolution
121         # image.
122         yscale = float(frame.shape[0]/input_height)
123         xscale = float(frame.shape[1]/input_width)
124         # Dictionary to be filled with labels and probabilities for MQTT
125         cloud_output = {}
126         # Get the detected objects and probabilities
127         for obj in parsed_inference_results[model_type]:
128             if obj['prob'] > detection_threshold:
129                 # Add bounding boxes to full resolution frame
130                 xmin = int(xscale * obj['xmin']) \
131                     + int((obj['xmin'] - input_width/2) + input_width/2)
132                 ymin = int(yscale * obj['ymin'])
133                 xmax = int(xscale * obj['xmax']) \
134                     + int((obj['xmax'] - input_width/2) + input_width/2)
135                 ymax = int(yscale * obj['ymax'])
136                 # See https://docs.opencv.org/3.4.1/d6/d6e/group__imgproc__draw.html
137                 # for more information about the cv2.rectangle method.
138                 # Method signature: image, point1, point2, color, and thickness.
139                 cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), (255, 165, 20),
140                               # amount to offset the label/probability text above the bounding box
141                               text_offset = 15
142                               # See https://docs.opencv.org/3.4.1/d6/d6e/group__imgproc__draw.html
143                               # for more information about the cv2.putText method.
144                               # Method signature: image, text, origin, font face, font scale, color
145                               # and thickness
146                               cv2.putText(frame, "{}: {:.2f}%".format(output_map[obj['label']],
147                                                               obj['prob'] * 100),
148                                         (xmin, ymin-text_offset),
149                                         cv2.FONT_HERSHEY_SIMPLEX, 2.5, (255, 165, 20), 6)
150                               # Store label and probability to send to cloud
151                               cloud_output[output_map[obj['label']]]=obj['prob']
152                               # Set the next frame in the local display stream.
153                               local_display.set_frame_data(frame)
154                               # Send results to the cloud
155                               client.publish(topic=iot_topic, payload=json.dumps(cloud_output))
156               except Exception as ex:
```

hotdog-detection.py:

```
95     # Since this is a binary classifier only retrieve 2 classes.
96     num_top_k = 2
97     # The height and width of the training set images
98     input_height = 224
99     input_width = 224
100    # Do inference until the lambda is killed.
101    while True:
102        # Get a frame from the video stream
103        ret, frame = awscam.getLastFrame()
104        if not ret:
105            raise Exception('Failed to get frame from the stream')
106        # Resize frame to the same size as the training set.
107        frame_resize = cv2.resize(frame, (input_height, input_width))
108        # Run the images through the inference engine and parse the results using
109        # the parser API, note it is possible to get the output of doInference
110        # and do the parsing manually, but since it is a classification model,
111        # a simple API is provided.
112        parsed_inference_results = model.parseResult(model_type,
113                                                       model.doInference(frame_resize))
114        # Get top k results with highest probabilities
115        top_k = parsed_inference_results[model_type][0:num_top_k]
116        # Get the probability of 'hotdog' label, which corresponds to label 934 in SqueezeNet.
117        prob_hotdog = 0.0
118        for obj in top_k:
119            if obj['label'] == 934:
120                prob_hotdog = obj['prob']
121                break
122        # Compute the probability of a hotdog not being in the image.
123        prob_not_hotdog = 1.0 - prob_hotdog
124        # Add two bars to indicate the probability of a hotdog being present and
125        # the probability of a hotdog not being present.
126        # See https://docs.opencv.org/3.4.1/d6/d6e/group__imgproc__draw.html
127        # for more information about the cv2.rectangle method.
128        # Method signature: image, point1, point2, color, and thickness.
129        cv2.rectangle(frame, (0, 0), (int(frame.shape[1] * 0.2 * prob_hotdog), 80),
130                      (0, 0, 255), -1)
131        cv2.rectangle(frame, (0, 90), (int(frame.shape[1] * 0.2 * prob_not_hotdog), 170), (0, 255, 0), -1)
132        font = cv2.FONT_HERSHEY_SIMPLEX
133        # See https://docs.opencv.org/3.4.1/d6/d6e/group__imgproc__draw.html
134        # for more information about the cv2.putText method.
135        # Method signature: image, text, origin, font face, font scale, color,
136        # and thickness
137        cv2.putText(frame, 'Not hotdog', (10, 70), font, 3, (225, 225, 225), 8)
138        cv2.putText(frame, 'Hotdog', (10, 160), font, 3, (225, 225, 225), 8)
139        # Send the top k results to the IoT console via MQTT
140        client.publish(topic=iot_topic, payload=json.dumps({'Hotdog': prob_hotdog,
141                                                               'Not hotdog': prob_not_hotdog}))
142        # Set the next frame in the local display stream.
143        local_display.set_frame_data(frame)
144    except Exception as ex:
```

SEE WHAT OTHERS HAVE BUILT

VIEW ALL 23 PROJECTS: aws.amazon.com/deeplens/community-projects

AWS DEEPLENS HACKATHON WINNERS:

First Place



ReadToMe

Created by Alex Schultz

ReadToMe is a deep learning enabled application that is able to read books to kids. In this case, reading Green Eggs and Ham, by Dr. Seuss.

Second Place



Dee

Created by Matthew Clark

Dee is a fun AWS DeepLens interactive device for children. The device asks children to answer questions by showing a picture of the answer.

Third Place



SafeHaven

Created by Nathan Stone and Peter McLean

SafeHaven uses Alexa and AWS DeepLens to bring peace of mind for vulnerable people and their families.

DEAR demo



▶ ⏪ 0:22 / 2:14

CC YouTube

SUBMITTED TO



AWS DeepLens Challenge

CREATED BY



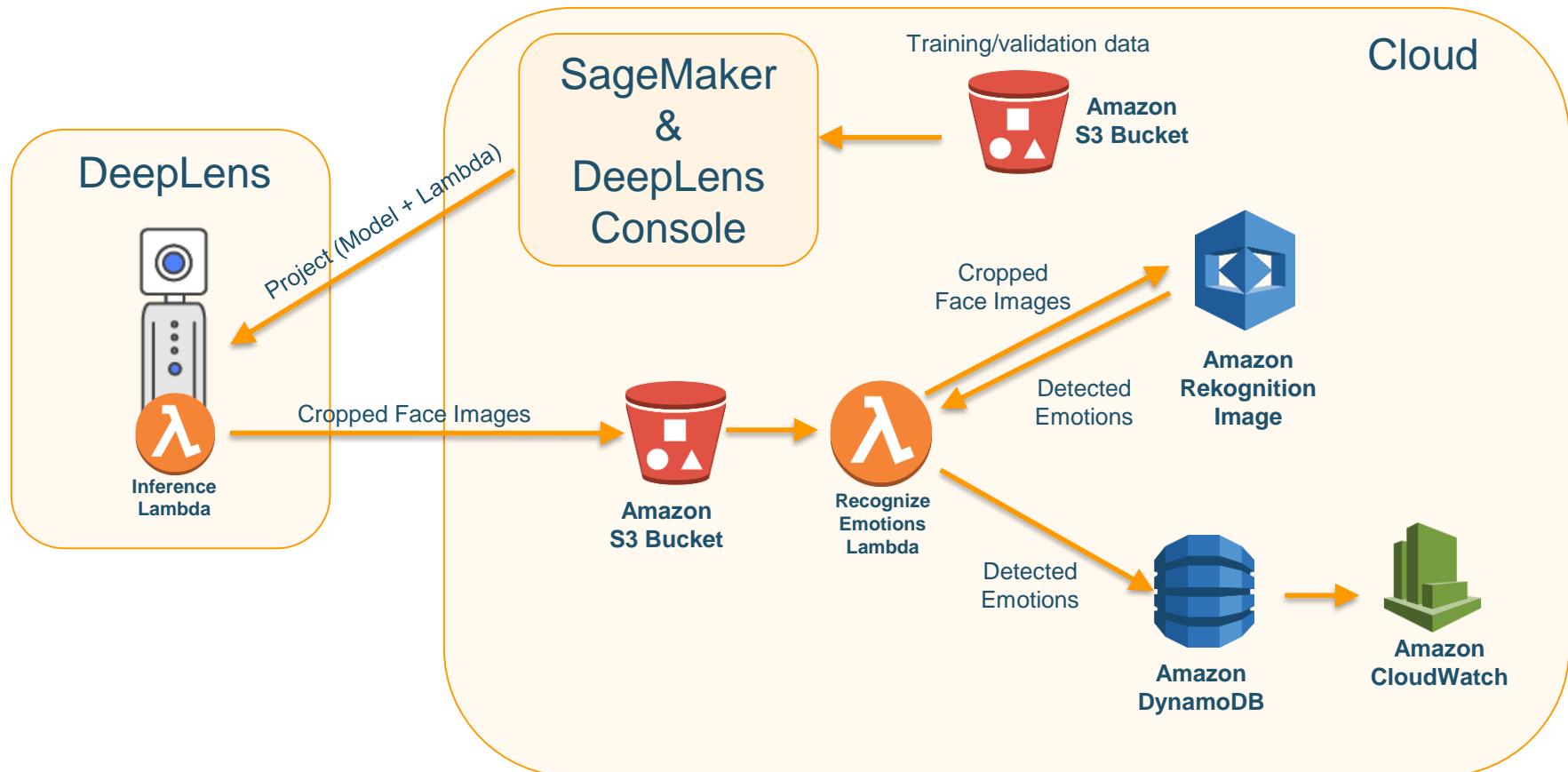
Ricardo Mota [+](#)



Jidesh
Veeramachaneni [+](#)

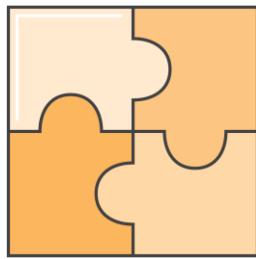
Report inappropriate content

Watch the demo:
<https://youtu.be/2oMGkWc46Lw>



AMAZON SAGEMAKER

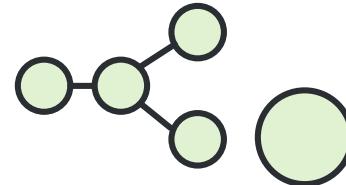
The quickest and easiest way to get ML models from idea to production



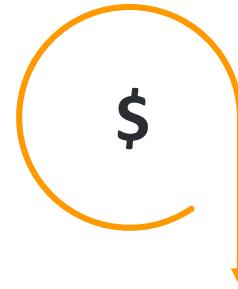
End-to-End
Machine Learning
Platform



Zero Setup



Flexible Model
Training



Pay By The Second

5. KEEP LEARNING



CHOOSE YOUR MENTORS WISELY

- Unlike 25+ years ago when there was a lack of information, Internet overwhelms us with information. New problem is finding good mentors (not just those who sound good) – because nowadays everyone writes, records videos, and brags.
- **Join local communities!** (“Collaborative Filtering” the old fashioned way ;)
- Follow the links in my curated list of **awesome** people and resources where to go for knowledge next...

And let's make the world a better place through AI & ML!

DEEPLENS & SAGEMAKER

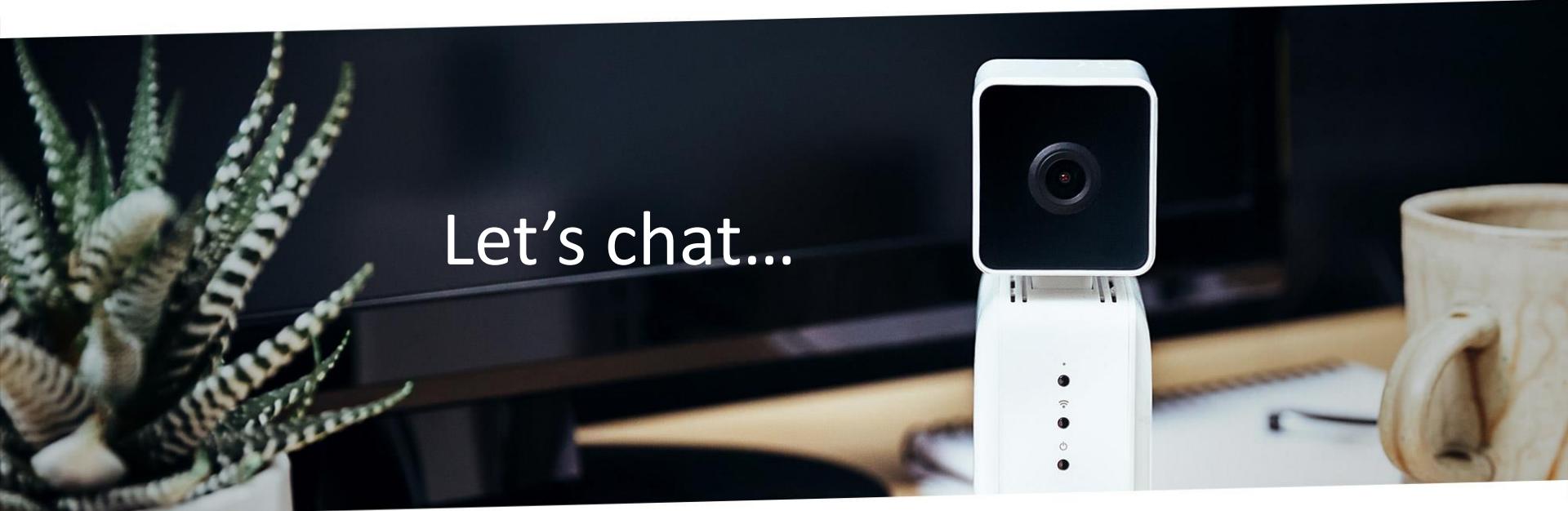
- **Julien Simon**, AI & Machine Learning @ AWS <https://medium.com/@julsimon>
 - Deep Learning at the edge with AWS DeepLens - Julien Simon (AWS)
<https://www.youtube.com/watch?v=Vw-bX1RRZGQ>
 - An overview of Amazon SageMaker <https://www.youtube.com/watch?v=ym7NEYEx9x4>
- Dev Guide <https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-dg.pdf>
- **Lynn Langit**: “Amazon Web Services Machine Learning Essential Training” 3hr course on Lynda.com or
<https://www.linkedin.com/learning/amazon-web-services-machine-learning-essential-training>
- **Randall Hunt**, Sr Technical Evangelist @ AWS <https://aws.amazon.com/blogs/aws/author/randhunt/>

ALEXA & VOICE-FIRST AI

- **Paul Cutsinger**, Head of Alexa Voice Design Education @ AWS <https://twitter.com/PaulCutsinger>
- Alexa Blogs <https://developer.amazon.com/blogs/alexa/>
- ML uses case “Chatbots & Voice User Interfaces (the Amazon way)” – samples at <http://goo.gl/H5CEpW> slides at <https://www.slideshare.net/dzivkovi/3in1-talk-on-serverless-chatbots-alexa-skills-voice-ui-best-practices-the-amazon-way>

ML & DEEP LEARNING

- **Robert Newman** @ Beyondsoft “How to Leverage Machine Learning”
<https://www.brighttalk.com/webcast/16661/321515/how-to-leverage-machine-learning>
- **Geoffrey Hinton** “The Foundations of Deep Learning” <https://www.youtube.com/watch?v=zI99IZvW7rE>
- Deep Learning Crash Course https://gluon.mxnet.io/chapter01_crashcourse/introduction.html
- **3Blue1Brown** playlist “Neural Networks”
https://www.youtube.com/playlist?list=PLZHQBObOWTQDNU6R1_67000Dx_ZCJB-3pi
- **Jeremy Howard** [https://en.wikipedia.org/wiki/Jeremy_Howard_\(entrepreneur\)](https://en.wikipedia.org/wiki/Jeremy_Howard_(entrepreneur))
 - Preview our Deep Learning Certificate with a lecture by Jeremy Howard <https://www.usfca.edu/data-institute/certificates/deep-learning-part-one>
 - The wonderful and **terrifying** implications of computers that can learn
<https://www.youtube.com/watch?v=t4kyRyKyOpo>
 - Practical Deep Learning for Coders <http://course.fast.ai/start.html> (2018 edition of 7-week course)
- **UN Sustainable Development Goals** <https://www.globalgoals.org/> (17 goals for a better world by 2030)
- AI grounded in human values, empowering people to live better lives <https://www.PartnershipOnAI.org>
- Make the world a better place through AI <https://ai4good.org/>



Let's chat...