Real Estate Search Ranking with BigDL Framework on Microsoft Azure Platform

Dave Wetzel, COO and CTO, MLS Listings
Sergey Ermolin, Solutions Architect, Intel

#ExpSAIS16

MLSListings Inc, Sunnyvale, California

MLSListings Business Use-Case: Personalized Visual Search Ranking

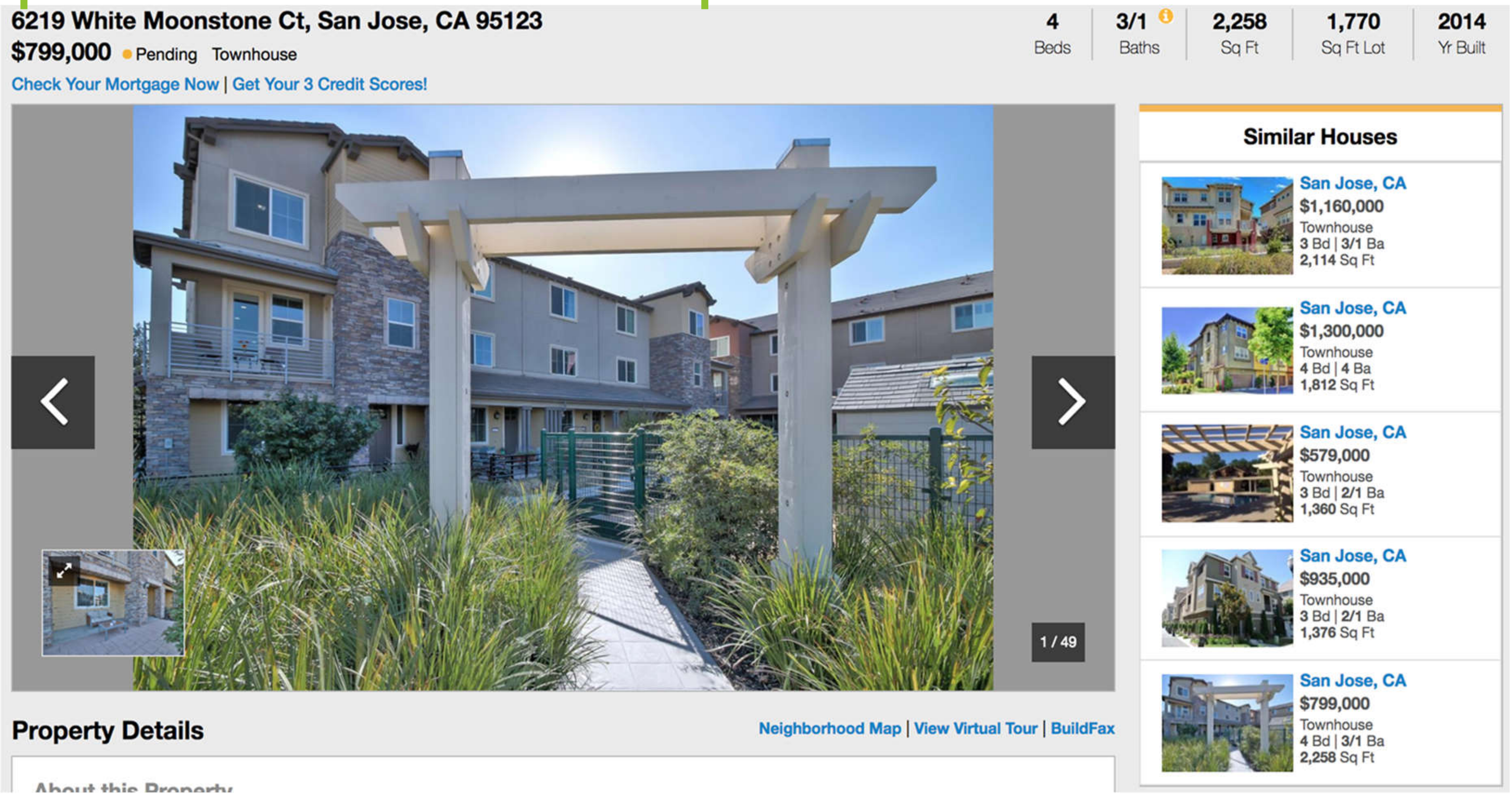
If you looked at this house…..

You will want to look at this one, too

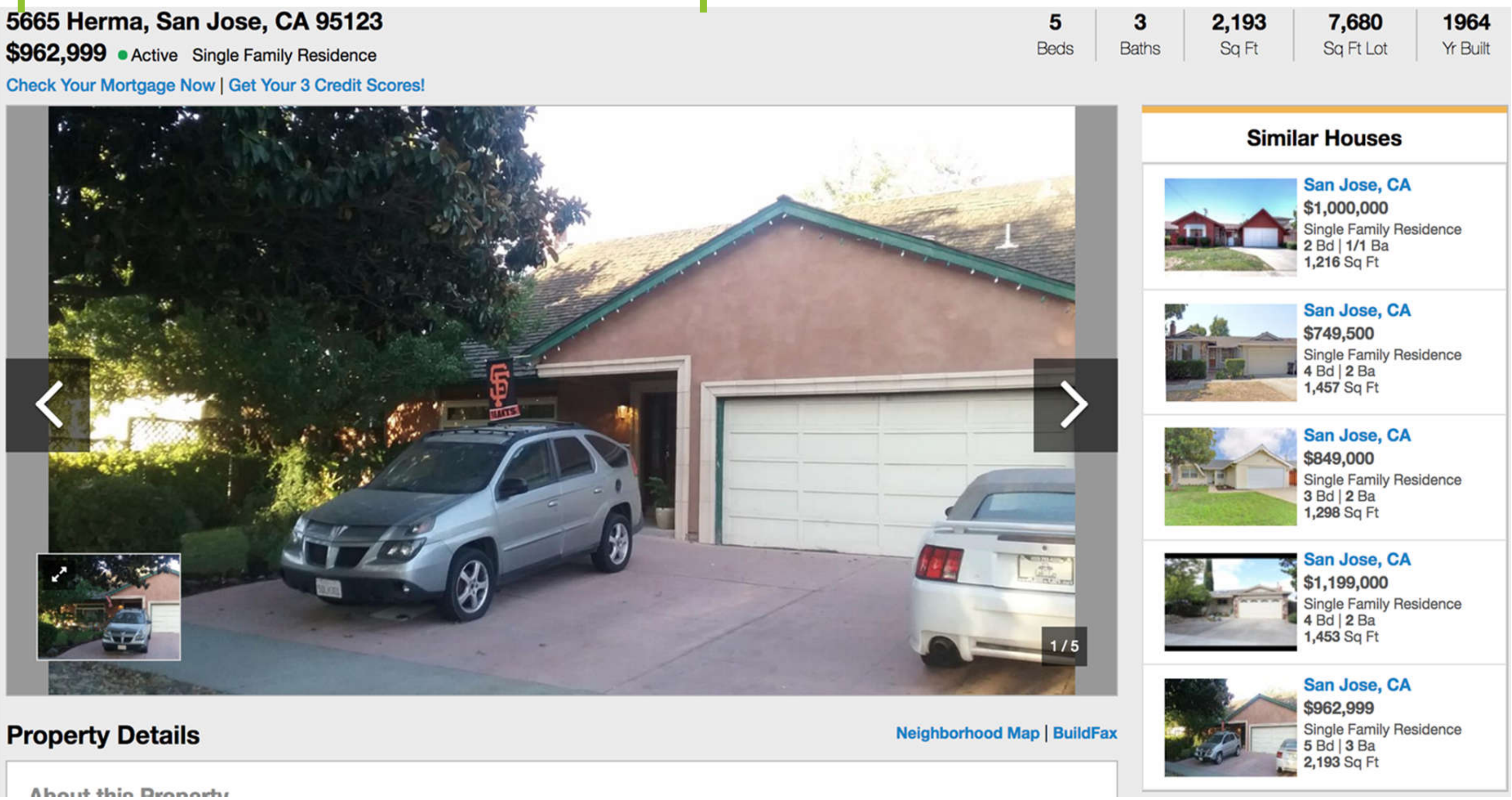
Image similarity is an extra search parameter along with area, location, size, price, etc.

Business need: real estate search results need to be sorted based on image similarities of attached photos

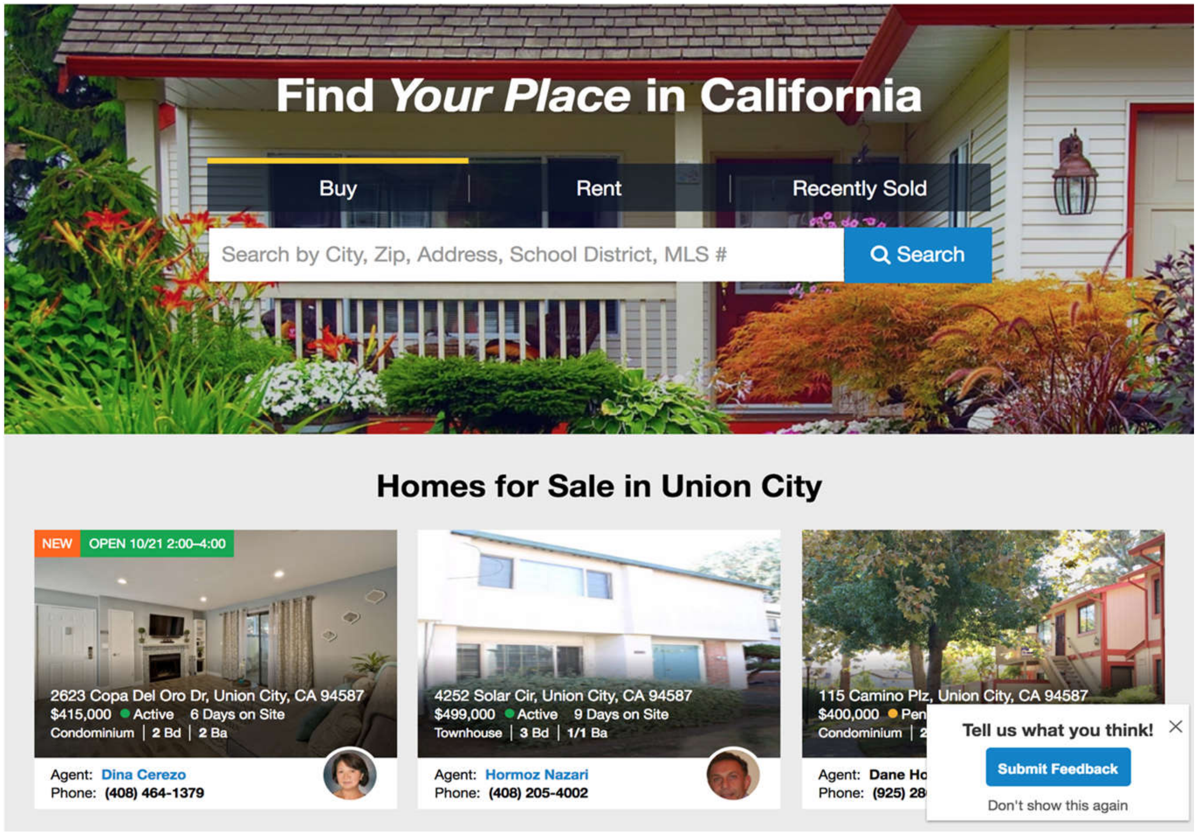
Implementation Example - 1



Implementation Example - 2

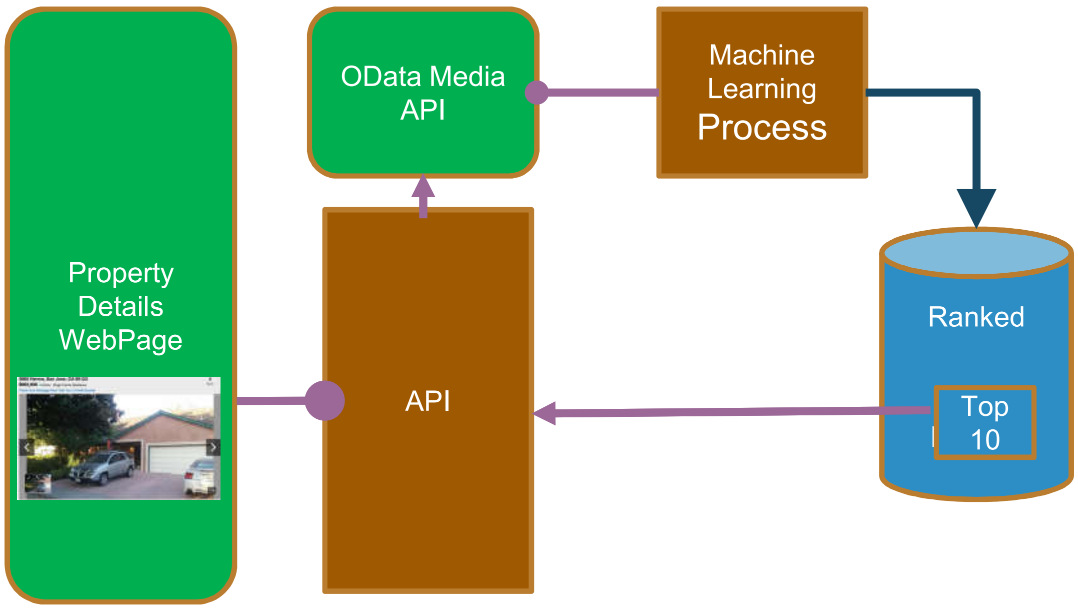


LIVE DEMO



Find Your Place in California
Buy
Rent
Recently Sold
Search by City, Zip, Address, School District, MLS #
Search
Homes for Sale in Union City
NEW
OPEN 10/21 2:00–4:00
2623 Copa Del Oro Dr, Union City, CA 94587
$415,000 Active 6 Days on Site
Condominium | 2 Bd | 2 Ba
Agent: Dina Cerezo
Phone: (408) 464-1379
4252 Solar Cir, Union City, CA 94587
$499,000 Active 9 Days on Site
Townhouse | 3 Bd | 1/1 Ba
Agent: Hormoz Nazari
Phone: (408) 205-4002
115 Camino Plz, Union City, CA 94587
$400,000
Condominium
Agent:
Phone:
Tell us what you think!
Submit Feedback
Don't show this again

High-Level Data Workflow



Deep Learning Data Flow

Training

Data Engineering | **Deep Learning** | **Compute**

Labeled Dataset of Real Estate Images (Bing Search) + BigDL VGG Arch = BigDL Trained Model

- Long Compute. 8500 images
- 2 nodes, 28 cores/node. 3 minutes for a one single pass
- Model parameters are changing.
- Repeat until convergence
- But: only do once !

    Note-1: Images are *not* stored in the model

    Note-2: you can trade compute resources for time.

Deep Learning Data Flow

Inference

Compute Only

+ BigDL Trained Model = Feature Vector

- Image Class (Front, Bdr, Bath,…)
- House Style Tag (Ranch, Victorian,…)
- House Levels (1, 2..)
- Latent Features (25k entries)

- Short Compute. Real Time
- 1 node, 1 core/node
- Model parameters unchanged.
- Only run once per image
- But: need to do for every image in the searched dataset !

Deep Learning Data Flow – putting it together

Training

Labeled Dataset of Real Esate Images (Bing Search) + BigDL VGG CNN = BigDL Trained Model

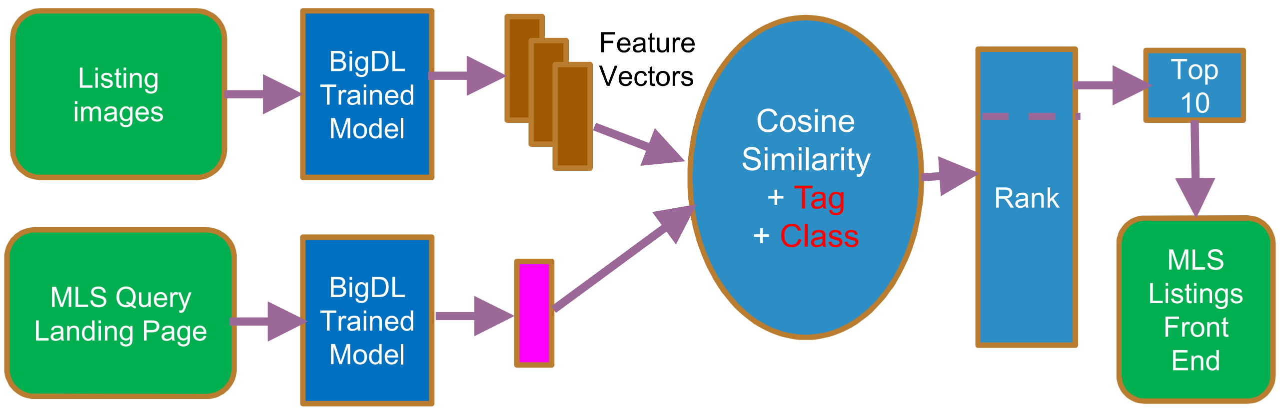
Inference

 + BigDL Trained Model = Feature Vector

- Image Class (Front, Bdr, Bath,…)
- House Style Tag (Ranch, Victorian,…)
- House Levels (1, 2..)
- Latent Features (25k entries)

Deep Learning Data Flow

Real-time Ranking

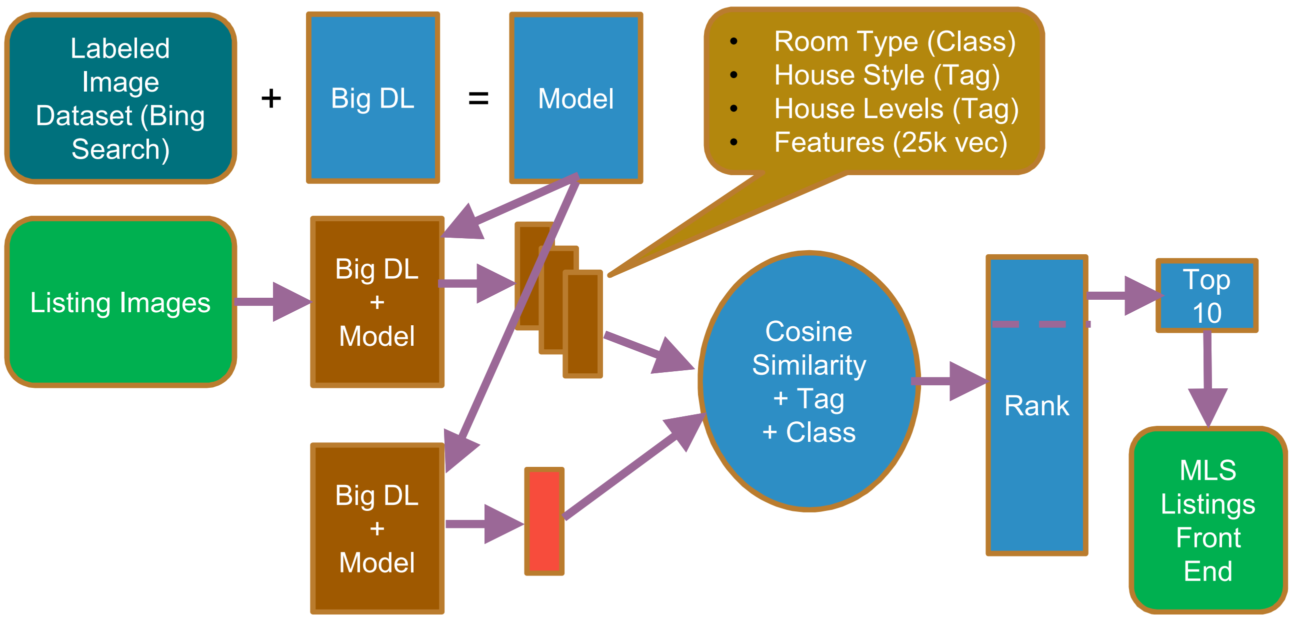


Cosine similarity measure: (Weighed)

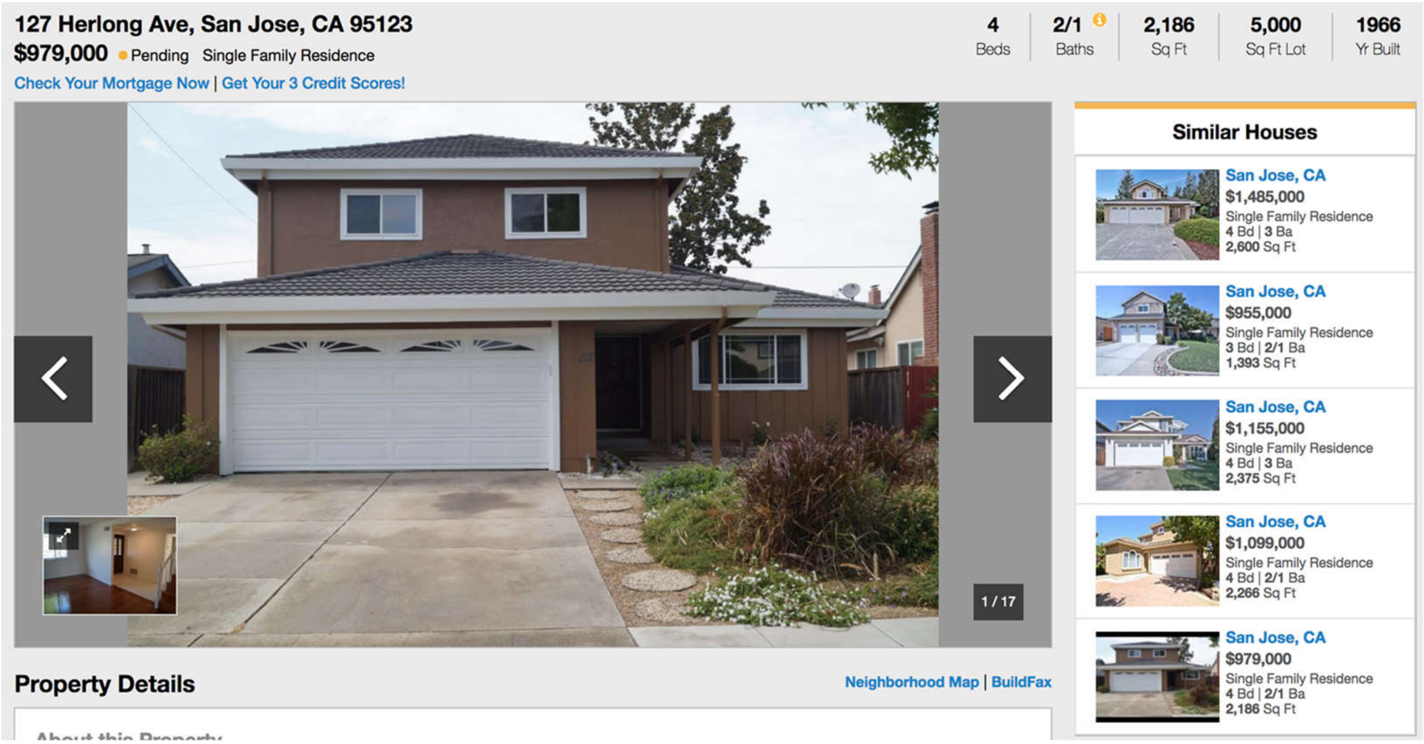
- $\text{sim}(\boldsymbol{x}, \boldsymbol{y}) = \cos(\boldsymbol{r}_x, \boldsymbol{r}_y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$

r_x, r_y as points:
$r_x = \{1, 0, 0, 1, 3\}$
$r_y = \{1, 0, 2, 2, 0\}$

Deep Learning Data Flow



Implementation Example - 3



Implementation - BigDL

Building BigDL Graph

```
(trainingDF, validationDF) = labelDF.randomSplit([0.9, 0.1])
numClasses = 4
transformer = NNImageTransformer(
    image.Pipeline([Resize(256, 256), CenterCrop(224, 224),
    ChannelNormalize(123.0, 117.0, 104.0)]))
     .setInputCol("image").setOutputCol("features")
caffeModel = Model.load_caffe_model(def_path, weight_path)
preTrainedNNModel = NNModel(caffeModel, [3,224,224])
     .setPredictionCol("embedding")
```

- Prepare Training/Validation data.
- Image Transformer:
  - Image scale/crop
  - Channel color normalizing
- Caffe Model Import
- Render BigDL as SparkML Transformer

```
lrModel = Sequential().add(Linear(1024, numClasses)) \
    .add(LogSoftMax())
classifier = NNClassifier(lrModel, ClassNLLCriterion(), [1024]) \
    .setLearningRate(1e-3).setBatchSize(40) \
    .setMaxEpoch(100).setFeaturesCol("embedding")
```

- Create BigDL Linear SoftMax model
- Define Classifier, SparkML Transformer

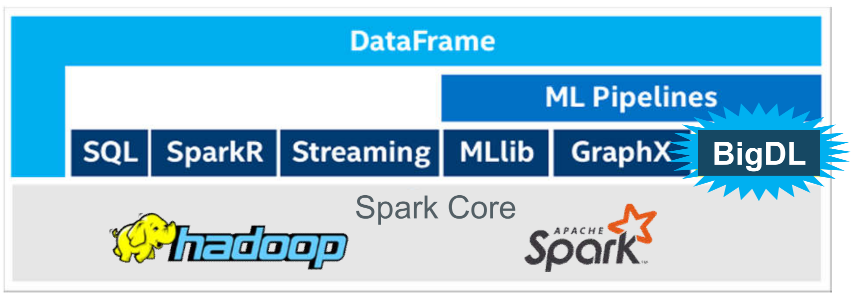
```
pipeline = Pipeline(stages=[transformer, preTrainedNNModel,\
                            classifier])
HouseStyleModel = pipeline.fit(trainingDF)
```

- Set up SparkML Pipeline

Executing BigDL Graph

```
predictionDF = HouseStyleModel.transform(validationDF)
predictionDF.show()
evaluator = MulticlassClassificationEvaluator(
    labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictionDF)
```

BigDL: Performance Deep Learning for Apache Spark* on CPU Infrastructure



BigDL is an **open-source** distributed deep learning library for Apache Spark* that can run directly on top of existing Spark or Apache Hadoop* clusters

Ideal for DL Models TRAINING and INFERENCE

Designed and Optimized for Intel® Xeon®

No need to deploy costly GPUs, duplicate data, or suffer through scaling headaches!

Feature Parity & Model Exchange with TensorFlow*, Caffe*, Keras, Torch*

Lower TCO and improved ease of use with existing infrastructure

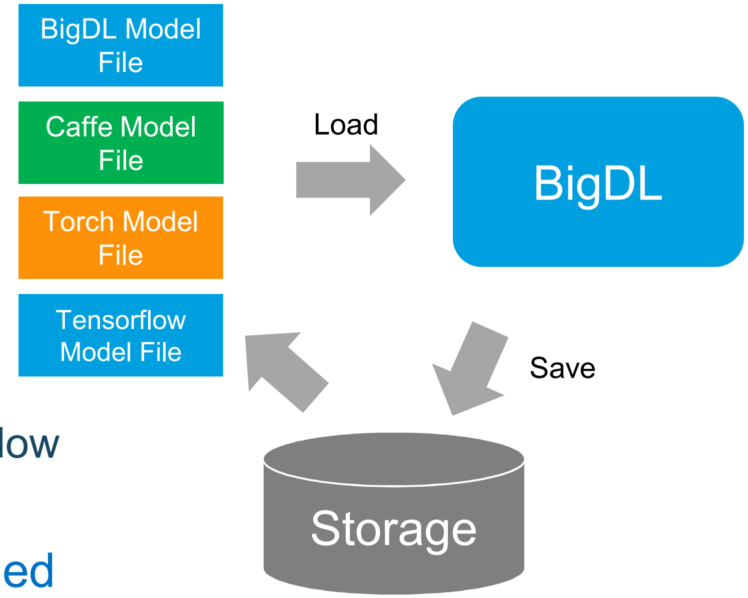
Deep Learning on Big Data Platform, Enabling **Efficient Scale-Out**

Powered by Intel® MKL and multi-threaded programming

https://github.com/intel-analytics/analytics-zoo

Models Interoperability Support

- Model Snapshot
  - Long training work checkpoint
  - Model deployment and sharing
  - Fine-tune
- Caffe/Torch/Tensorflow Model Support
  - Model file load
  - Easy to migrate your Caffe/Torch/Tensorflow code base to Spark
- **NEW** - BigDL supports loading pre-defined Keras models (Keras 1.2.2)

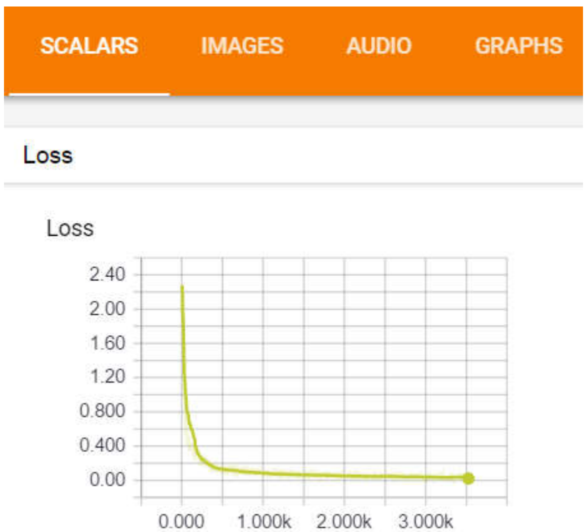


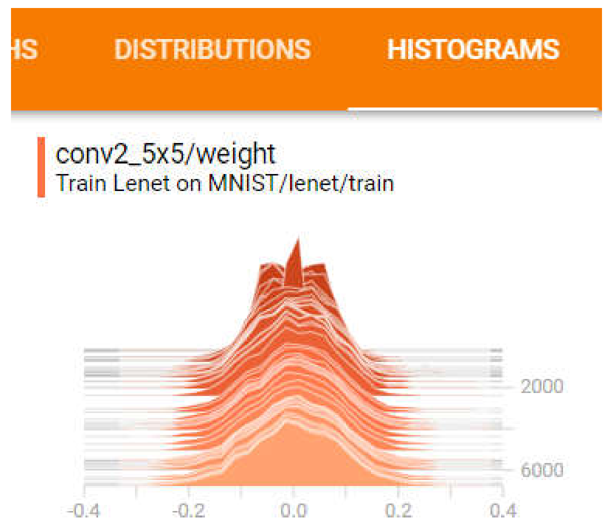
https://github.com/intel-analytics/analytics-zoo

Visualization for Learning

BigDL integration with TensorBoard

- TensorBoard is a suite of web applications from Google for visualizing and understanding deep learning applications





2018 - BIGDL ANALYTICS ZOO STACK

Reference Use Cases	Anomaly detection, sentiment analysis, fraud detection, chatbot, sequence prediction, etc.
Built-In Algorithms and Models	Image classification, object detection, text classification, recommendations, GAN, etc.
Feature Engineering and Transformations	Image, text, speech, 3D imaging, time series, etc.
High-Level Pipeline APIs	DataFrames, ML Pipelines, Autograd, Transfer Learning, etc.
Runtime Environment	Spark, BigDL, Python, etc.

Making it easier to build end-to-end analytics + AI applications

https://github.com/intel-analytics/analytics-zoo

Engineering Team

- Data scientist, proficient in Machine Learning / Deep Learning
- Software Engineer, experience with Apache Spark.
- Technical project manager

Domain Expertise:

- Machine Learning / Deep Learning,
- Python, Scala

- Software Engineer, Web API
- Software Engineer, Web UI

Domain Expertise:

- OData, .net Core MSSQL
- C#, HTML, JavaScript

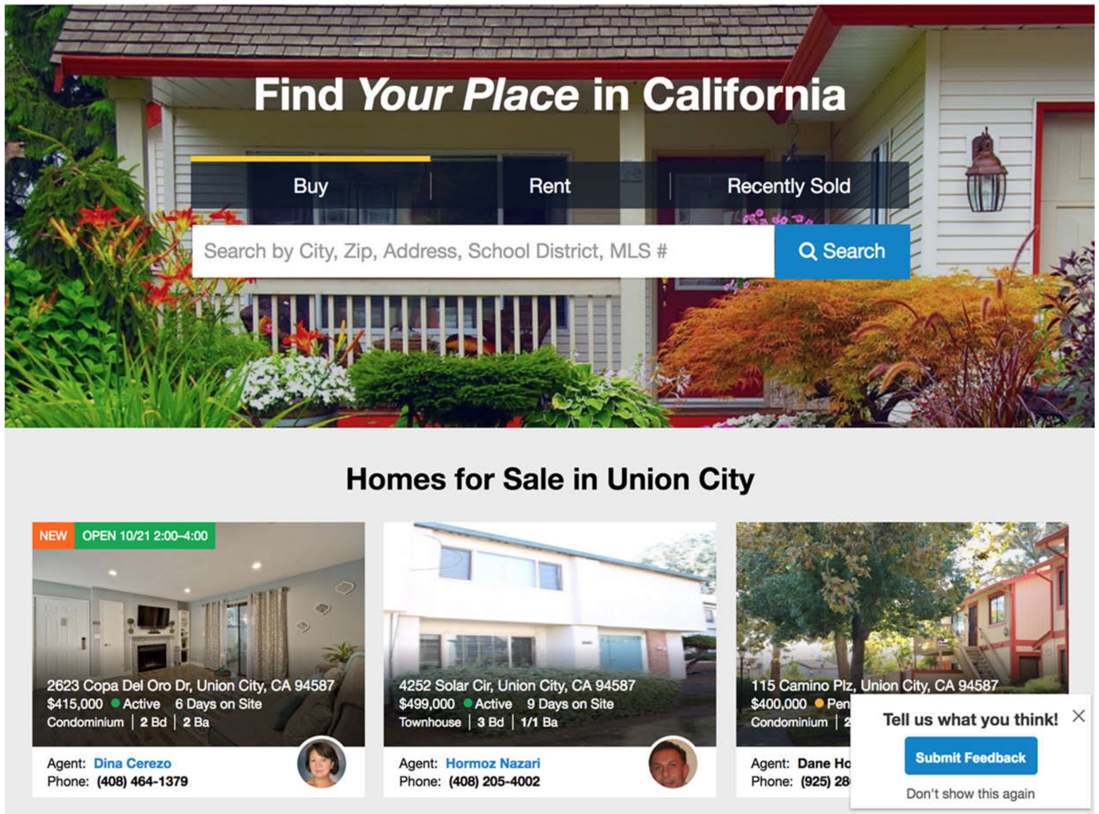
“ROAD AHEAD”

“Fireplace in the living room”

1. Feature extraction and tagging.
2. Image-based listings search
3. Feature verification based on listing images.
4. Image-based compliance and quality check

Thank You
and
Questions

LIVE DEMO



Find Your Place in California
Buy
Rent
Recently Sold
Search by City, Zip, Address, School District, MLS #
Search
Homes for Sale in Union City
NEW
OPEN 10/21 2:00–4:00
2623 Copa Del Oro Dr, Union City, CA 94587
$415,000 Active 6 Days on Site
Condominium | 2 Bd | 2 Ba
Agent: Dina Cerezo
Phone: (408) 464-1379
4252 Solar Cir, Union City, CA 94587
$499,000 Active 9 Days on Site
Townhouse | 3 Bd | 1/1 Ba
Agent: Hormoz Nazari
Phone: (408) 205-4002
115 Camino Plz, Union City, CA 94587
$400,000
Condominium
Tell us what you think!
Submit Feedback
Don't show this again

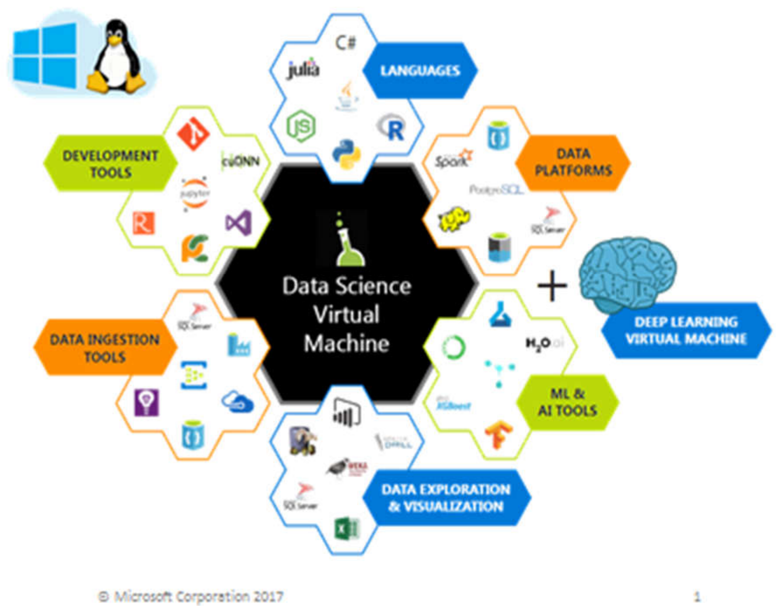
Infrastructure

Microsoft Data Science Virtual Machines

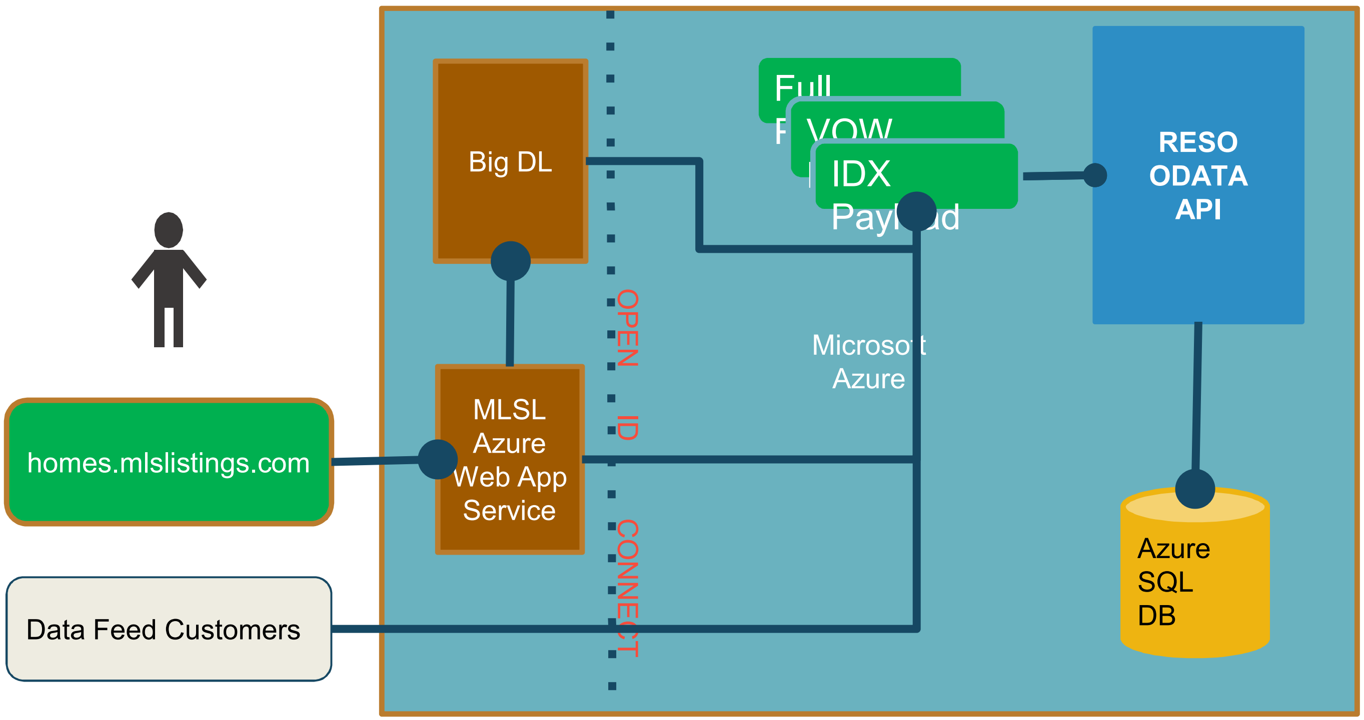
(DSVM)

Pre-Configured environments in the cloud for

Data Science & AI Modeling, Development & Deployment.



MLS Listings Apps and Services



Roles and Responsibilities

- Microsoft: Microsoft's data science team in Mountain View, CA participated in project discussions and provided Azure Data Science VM to deploy and train the deep learning model.
- • Microsoft - Apache Spark Cloud Service Provider
- • Intel - BigDL distributed Deep Learning Library
- • MLSListings - RESO Web API Provider
- Intel: Team members worked to integrate MLSListings's OData Media Services to deploy a custom real estate image similarity comparison solution on Azure using Big DL.
- MLSListings : MLSListings's team working on new web portal provided Media API and worked on the user interface to integrate with Big DL API.

BigDL: Python API

- Support deep learning model training, evaluation, inference
- Support Spark v1.6 - 2.2
- Support **Python 2.7/3.5/3.6**
- Based on PySpark, **Python API** in BigDL allows use of existing Python libs (Numpy, Scipy, Pandas, Scikit-learn, NLTK, Matplotlib, etc)

```
train_data = get_minst("train").map(
    normalizer(mnist.TRAIN_MEAN, mnist.TRAIN_STD))
test_data = get_minst("test").map(
    normalizer(mnist.TEST_MEAN, mnist.TEST_STD))
state = {"batchSize": int(options.batchSize),
         "learningRate": 0.01,
         "learningRateDecay": 0.0002}
optimizer = Optimizer(
    model=build_model(10),
    training_rdd=train_data,
    criterion=ClassNLLCriterion(),
    optim_method="SGD",
    state=state,
    end_trigger=MaxEpoch(100))
optimizer.setvalidation(
    batch_size=32,
    val_rdd=test_data,
    trigger=EveryEpoch(),
    val_method=["top1"]
)
optimizer.setcheckpoint(EveryEpoch(), "/tmp/lenet5/")
trained_model = optimizer.optimize()
```