Build, Scale, and Deploy Deep Learning Pipelines Using Apache Spark

Sue Ann Hong, Databricks
Tim Hunter, Databricks

#EUdd3

databricks

About Us

Sue Ann Hong

- Software engineer @ Databricks
- Ph.D. from CMU in Machine Learning

Tim Hunter

- Software engineer @ Databricks
- Ph.D. from UC Berkeley in Machine Learning
- Very early Spark user

This talk

- Deep Learning at scale: current state
- Deep Learning Pipelines: the vision
- End-to-end workflow with DL Pipelines
- Future

Deep Learning at Scale

: current state

What is Deep Learning?

- A set of machine learning techniques that use layers that transform numerical inputs
  - Classification
  - Regression
  - Arbitrary mapping
- Popular in the 80's as Neural Networks
- Recently came back thanks to advances in data collection, computation techniques, and hardware.

Success of Deep Learning

Tremendous success for applications with complex data

- AlphaGo
- Image interpretation
- Automatic translation
- Speech recognition

But requires a lot of effort

- No exact science around deep learning
- Success requires many engineer-hours

- Low level APIs with steep learning curve
- Not well integrated with other enterprise tools
- Tedious to distribute computations

What does Spark offer?

Very little in Apache Spark MLlib itself (multilayer perceptron)

Many Spark packages

Integrations with existing DL libraries

- Deep Learning Pipelines (from Databricks)
- Caffe (CaffeOnSpark)
- Keras (Elephas)
- mxnet
- Paddle
- TensorFlow (TensorFlow on Spark, TensorFrames)
- CNTK (mmlspark)

Implementations of DL on Spark

- BigDL
- DeepDist
- DeepLearning4J
- MLlib
- SparkCL
- SparkNet

Deep Learning in industry

- Currently limited adoption
- Huge potential beyond the industrial giants
- How do we accelerate the road to massive availability?

Deep Learning Pipelines

Deep Learning Pipelines: Deep Learning with Simplicity

- Open-source Databricks library
- Focuses on *ease of use* and *integration*
  - without sacrificing *performance*
- Primary language: Python

- Uses Apache Spark for *scaling out* common tasks
- Integrates with MLlib Pipelines to capture the ML workflow concisely

A typical Deep Learning workflow

- **Load data** (images, text, time series, …)
- **Interactive work**
- **Train**
  - Select an architecture for a neural network
  - Optimize the weights of the NN
- **Evaluate** results, potentially re-train
- **Apply**:
  - Pass the data through the NN to produce new features or output

A typical Deep Learning workflow

Load data

Interactive work

Train

Evaluate

Apply

- Image loading in Spark
- Pre-trained models
- Transfer learning — Part 1
- Distributed tuning — Part 2
- Distributed batch prediction
- Deploying models in SQL

End-to-End Workflow

with Deep Learning Pipelines

t

Deep Learning Pipelines

- **Load data**
- Interactive work
- Train
- Evaluate model
- Apply

Adds support for images in Spark

```
from sparkdl import readImages
image_df = readImages(sample_img_dir)
```

filePath	image
dbfs:/tmp/flower_photos/sample/100080576_f52e8ee070_n.jpg	{"mode":"RGB","height":263,"width":320,"nChannels":3,"data":"h4eFioqIjo6OkZ(
dbfs:/tmp/flower_photos/sample/100930342_92e8746431_n.jpg	{"mode":"RGB","height":209,"width":320,"nChannels":3,"data":"Ey4PEC8QDi8QI

- ImageSchema, reader, conversion functions to/from numpy arrays
- Most of the tools we'll describe work on ImageSchema columns

Upcoming: built-in support in Spark

- Spark-21866
- Contributing image format & reading to Spark
- Targeted for Spark 2.3
- Joint work with Microsoft

Deep Learning Pipelines

- Load data
- **Interactive work**
- Train
- Evaluate model
- Apply

Applying popular models

- Popular pre-trained models accessible through MLlib Transformers

```
predictor = DeepImagePredictor(inputCol="image",
                               outputCol="predicted_labels",
                               modelName="InceptionV3")

predictions_df = predictor.transform(image_df)
```

Applying popular models

```
predictor = DeepImagePredictor(inputCol="image",
                               outputCol="predicted_labels",
                               modelName="InceptionV3")

predictions_df = predictor.transform(image_df)
```

filePath	predicted_labels
dbfs:/tmp/flower_photos/sample/100080576_f52e8ee070_n.jpg 	[{"class":"n11939491","description":"daisy","probability":0.8805494}, {"class":"n02219486","description":"ant","probability":0.0020712544}, {"class":"n02206856","description":"bee","probability":0.00084249553}, {"class":"n02165456","description":"ladybug","probability":0.000675952}, {"class":"n03691459","description":"loudspeaker","probability":0.00063085736}, {"class":"n02190166","description":"fly","probability":0.0006220306}, {"class":"n02281406","description":"sulphur_butterfly","probability":0.0006154293}, {"class":"n07930864","description":"cup","probability":0.00055486656}, {"class":"n02112018","description":"Pomeranian","probability":0.0004993835}, {"class":"n07745940","description":"strawberry","probability":0.00048750438}]
dbfs:/tmp/flower_photos/sample/100930342_92e8746431_n.jpg	[{"class":"n03930313","description":"picket_fence","probability":0.18473865},

s

Deep Learning Pipelines

- Load data
- Interactive work
- **Train**
  - Transfer learning
  - Hyperparameter tuning
- Evaluate model
- Apply

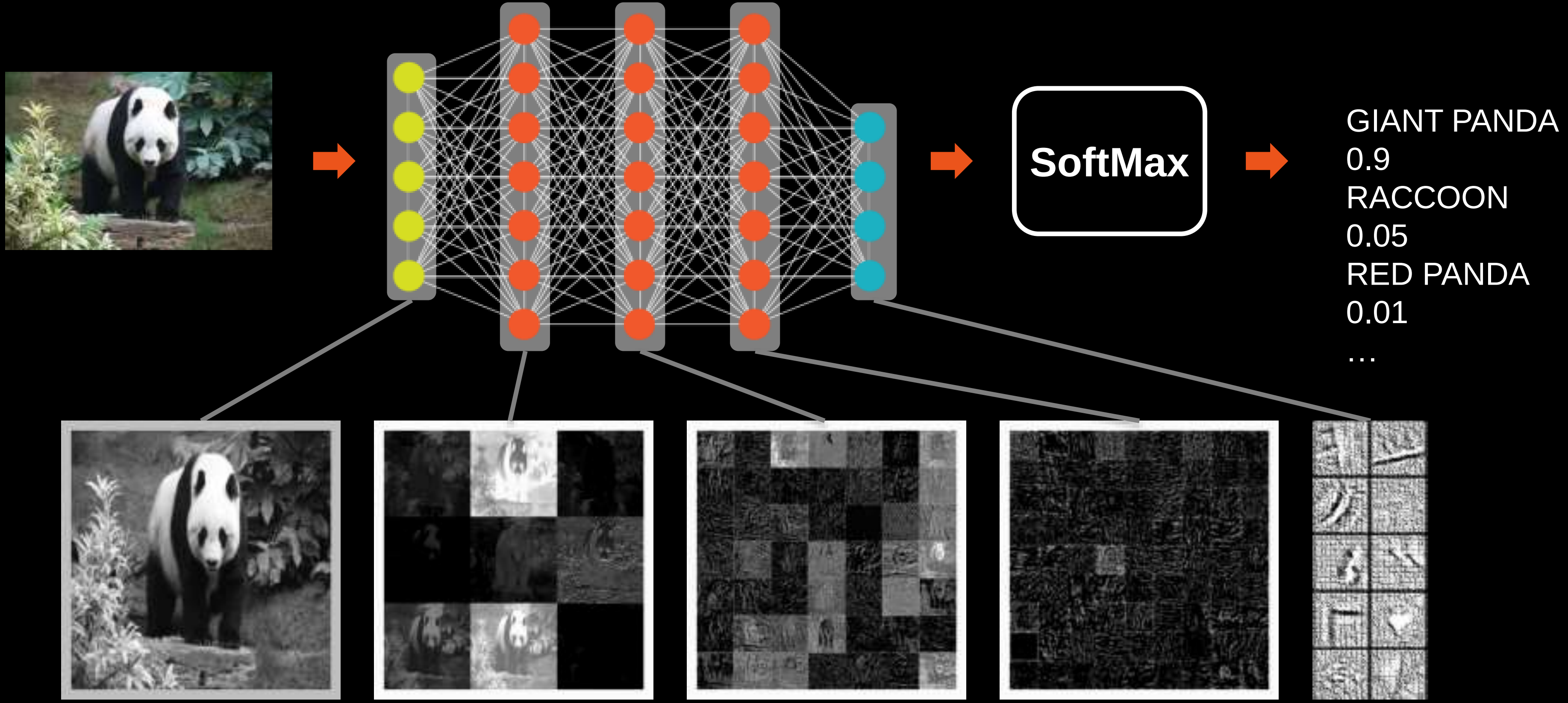
Deep Learning Pipelines

- Load data
- Interactive work
- **Train**
  - **Transfer learning**
  - Hyperparameter tuning
- Evaluate model
- Apply

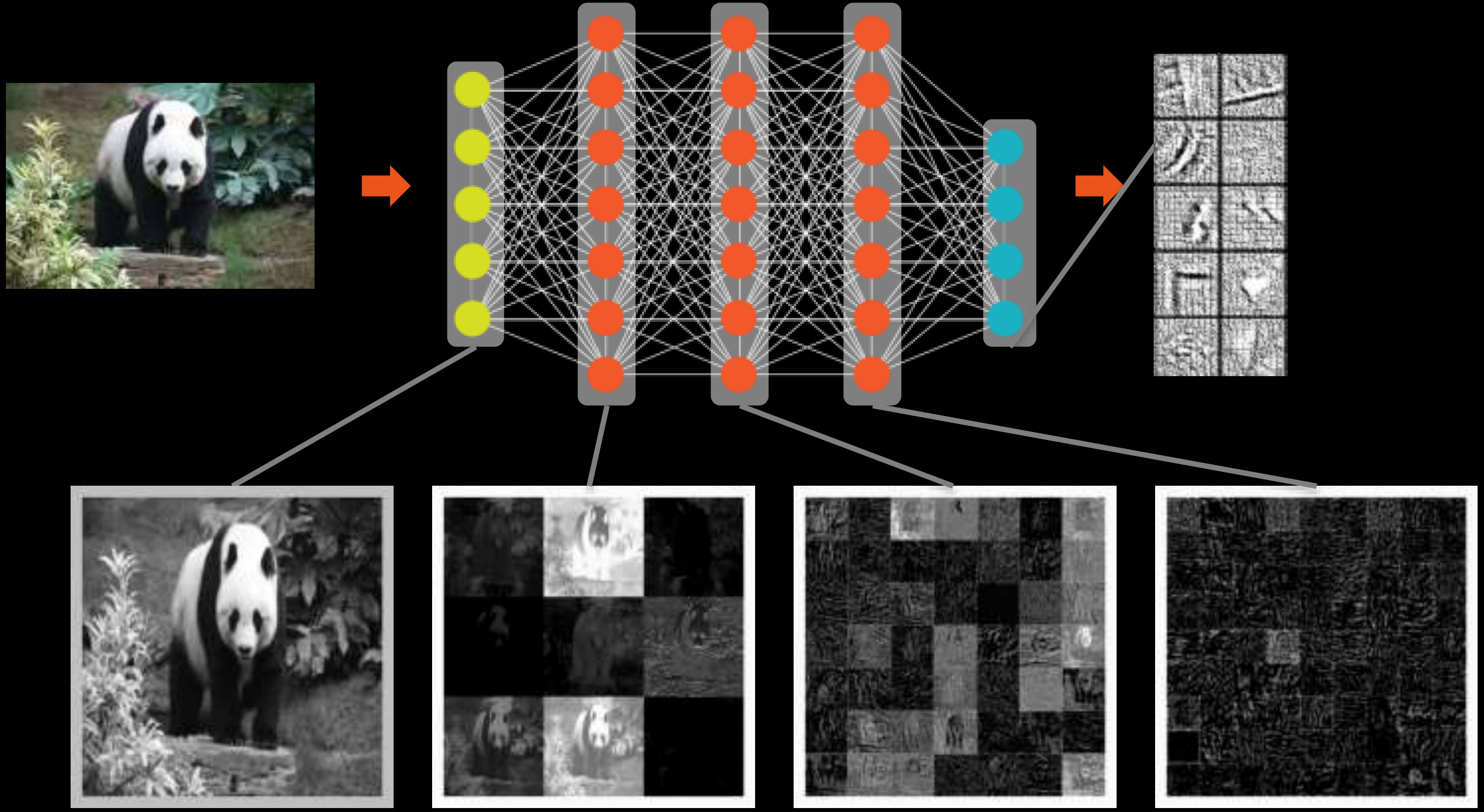
Transfer learning

- Pre-trained models may not be directly applicable
  - New domain, e.g. shoes
- Training from scratch requires
  - Enormous amounts of data
  - A lot of compute resources & time
- Idea: intermediate representations learned for one task may be useful for other related tasks

Transfer Learning



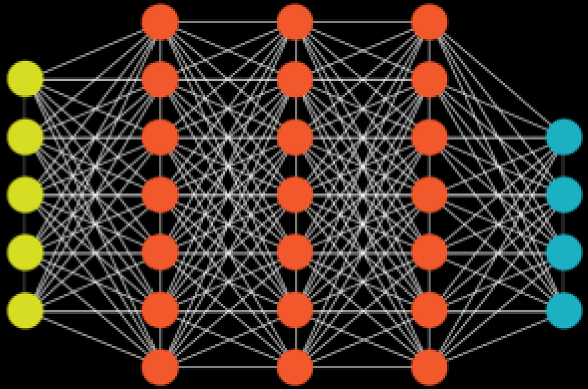
Transfer Learning



0.01

...

Transfer Learning

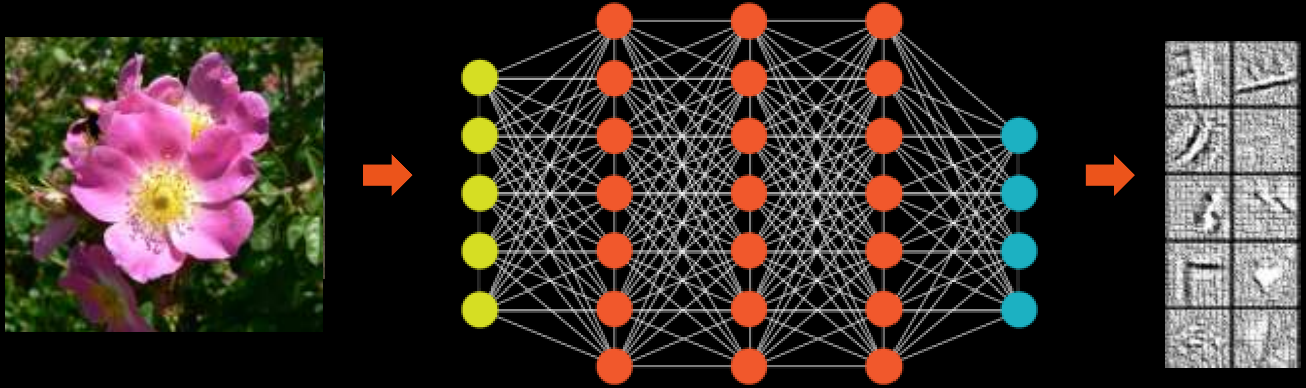


Classifie r

0.01

...

Transfer Learning



Classifie
r

Rose: 0.7

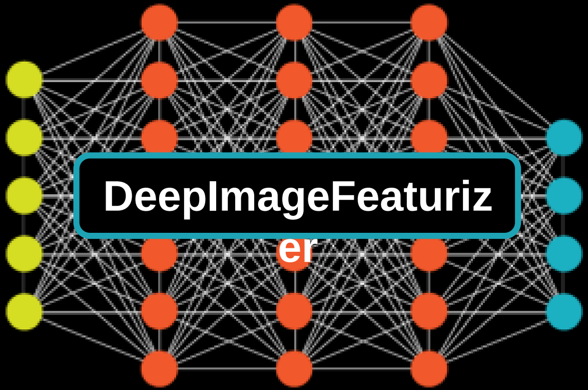
Daisy: 0.3

MLlib Pipelines primer

- MLlib: the machine learning library included with Spark
- `Transformer`
  - Takes in a Spark dataframe
  - Returns a Spark dataframe with new column(s) containing “transformed” data
  - e.g. a `Model` is a `Transformer`
- `Estimator`
  - A learning algorithm, e.g. `lr = LogisticRegression()`
  - Produces a `Model` via `lr.fit()`
- `Pipeline`: a sequence of Transformers and Estimators

0.01
…

Transfer Learning as a Pipeline

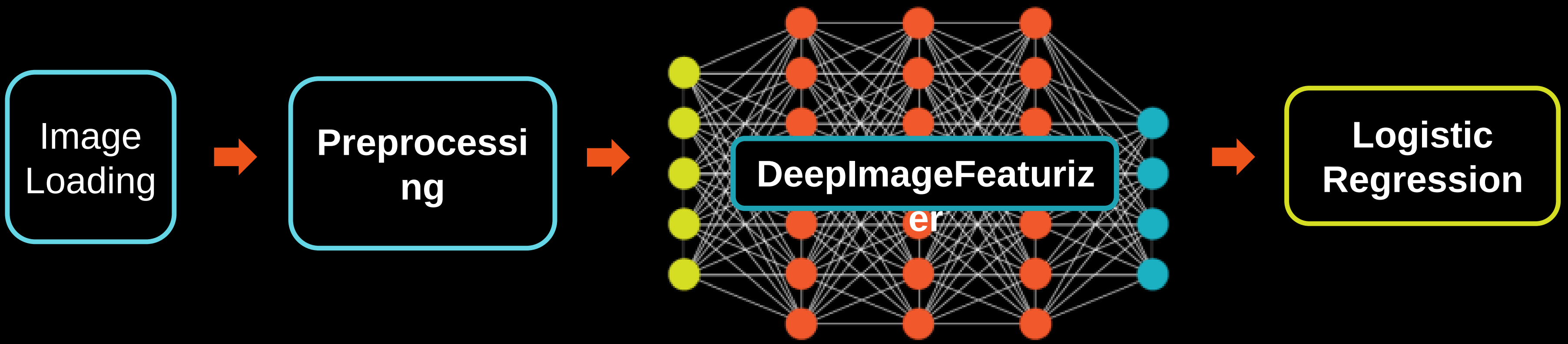


**Classifie
r**

Rose / Daisy

Transfer Learning as a Pipeline

0.01
...



MLlib Pipeline

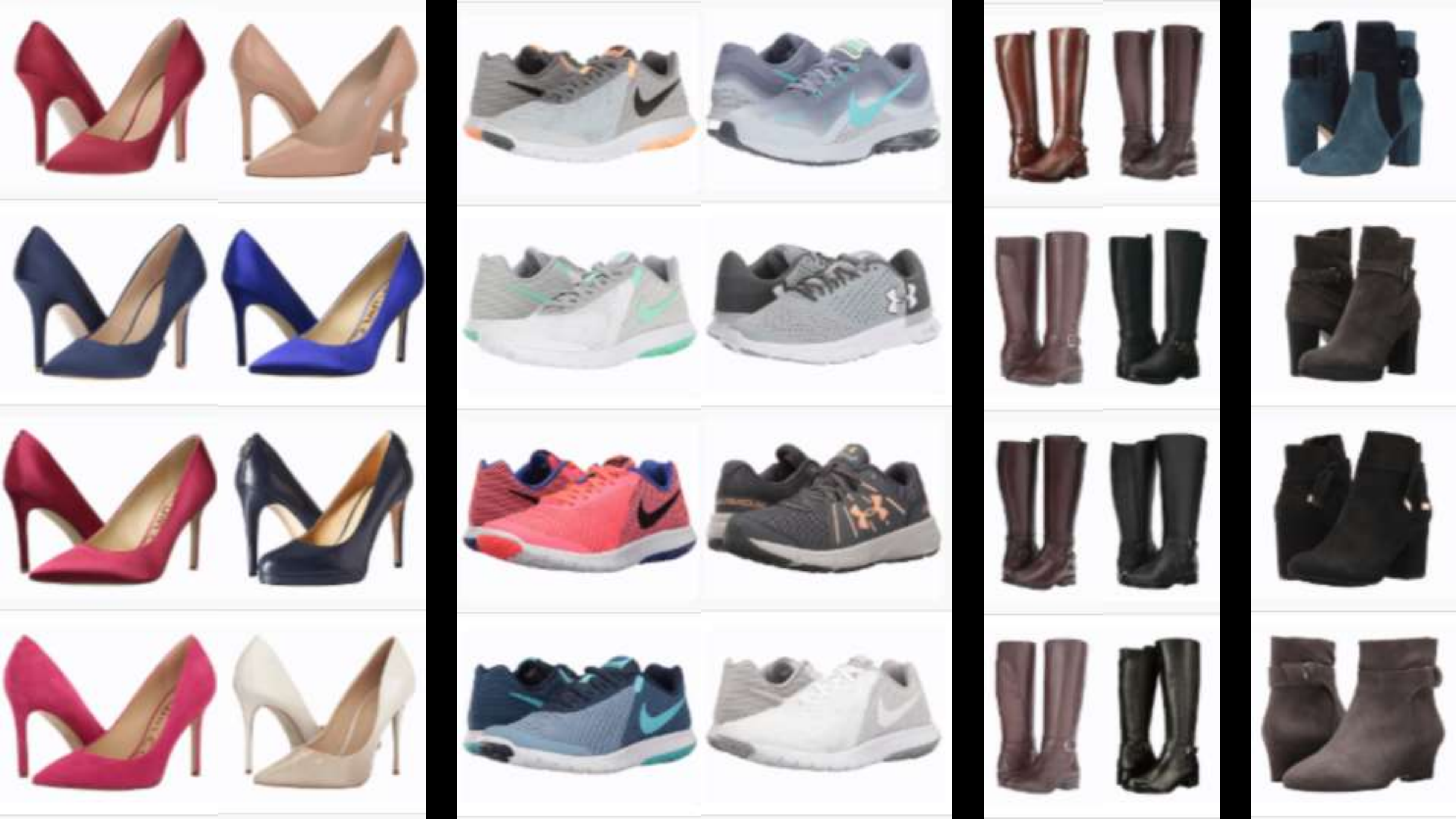
Transfer Learning as a Pipeline

```
featurizer = DeepImageFeaturizer(inputCol="image",
                                 outputCol="features",
                                 modelName="InceptionV3")

lr = LogisticRegression(labelCol="label")

p = Pipeline(stages=[featurizer, lr])

p_model = p.fit(train_df)
```

Transfer Learning

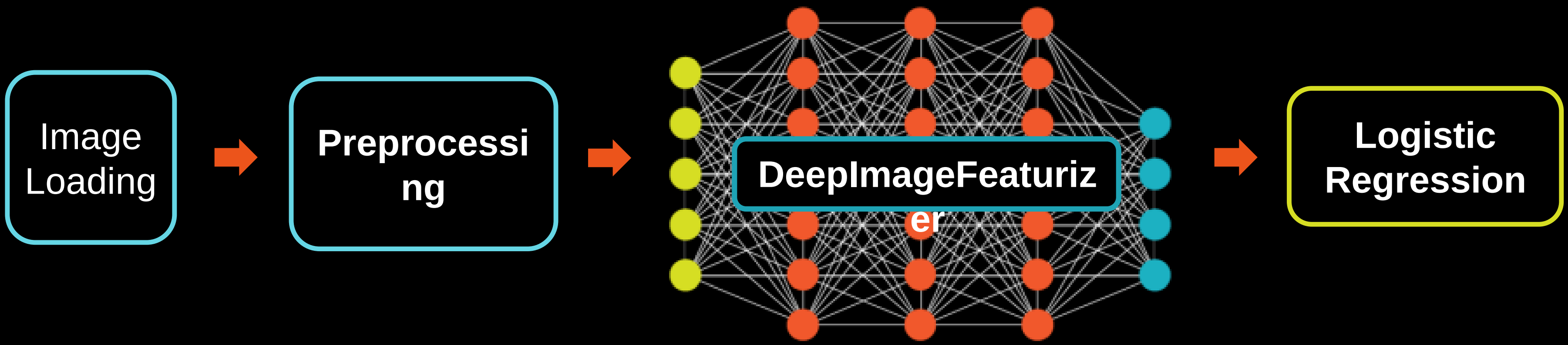
- Usually for classification tasks
  - Similar task, new domain

- But other forms of learning leveraging learned representations can be loosely considered transfer learning



0.01

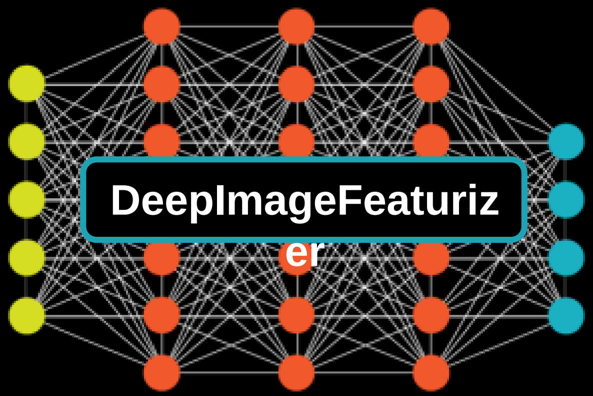
Featurization for similarity-based ML



0.01

Featurization for similarity-based ML

Image Loading → **Preprocessing** →



→

Nearest Neighbor
`KNN LSH`

Clustering
`KMeans`
`GaussianMixture`

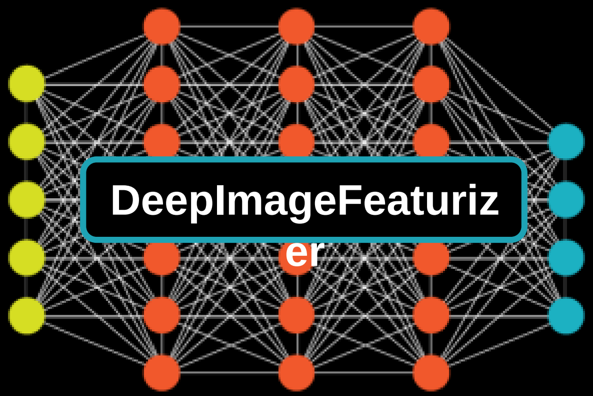
Distance computation

0.01

Featurization for similarity-based ML

Image
Loading

Preprocessi
ng



DeepImageFeaturiz
er

Nearest Neighbor
`KNN LSH`

Clustering
`KMeans`
`GaussianMixture`

Distance computation

Duplicate Detection

Anomaly Detection

Recommendatio
n

Search result diversification

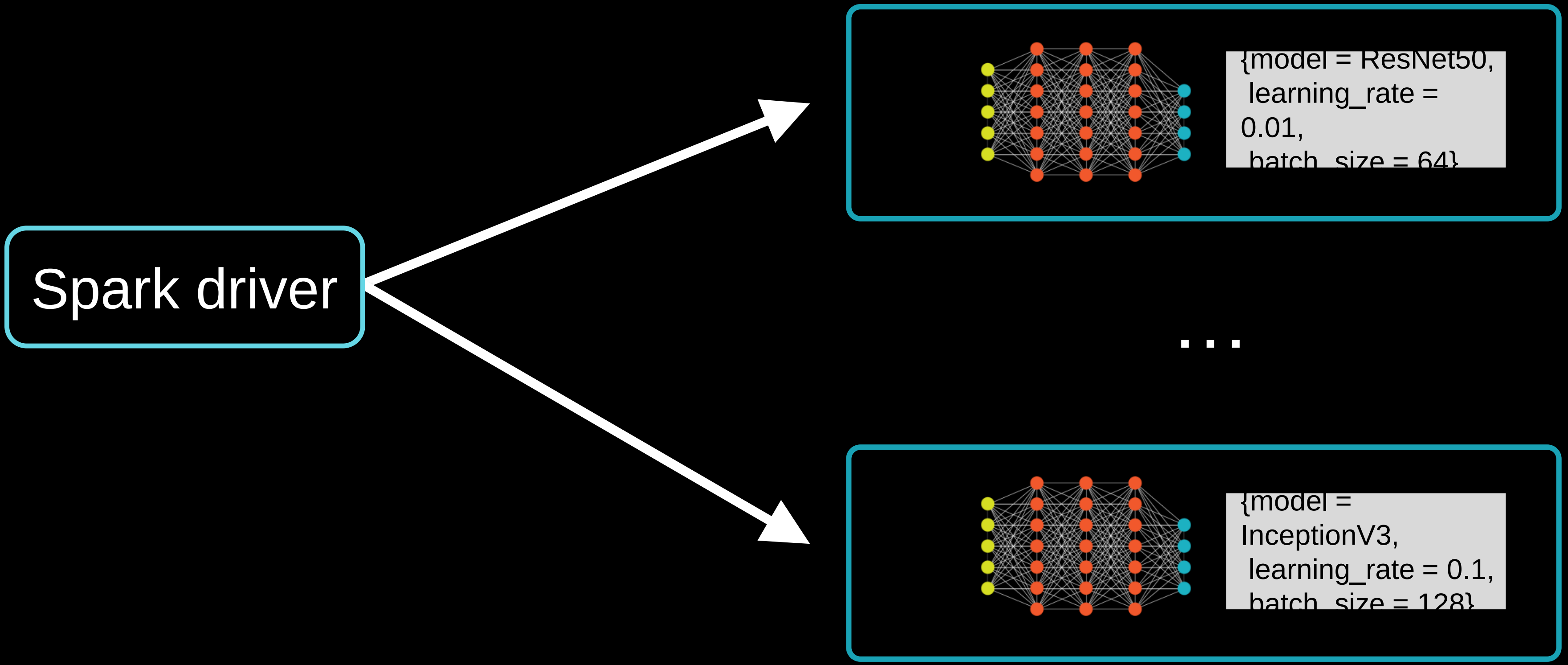
Break?

t

Deep Learning Pipelines

- Load data
- Interactive work
- **Train**
  - Transfer learning
  - $\boxed{\textbf{Hyperparameter tuning}}$
- Evaluate model
- Apply

Distributed Hyperparameter Tuning



MLlib Components

Estimators

- Learning algorithms
- Implement

```
model = fit(train_df, params)
```

e.g. learning rate

Evaluators

- Metric used to measure model's goodness on validation data
- e.g. `BinaryClassificationEvaluator` computes accuracy

Model Selection (Parameter Search)

Estimators

Map of `params`

Evaluators

```
paramGrid = ( ParamGridBuilder()
  .addGrid(hashingTF.numFeatures, [10, 100])
  .addGrid(lr.regParam, [0.1, 0.01])
  .build() )
```

```
cv = CrossValidator(
    estimator=pipeline,
    estimatorParamMaps=paramGrid,
    evaluator=BinaryClassificationEvaluator())

best_model = cv.fit(train_df)
```

Deep Learning Estimators

- Transfer learning Pipelines

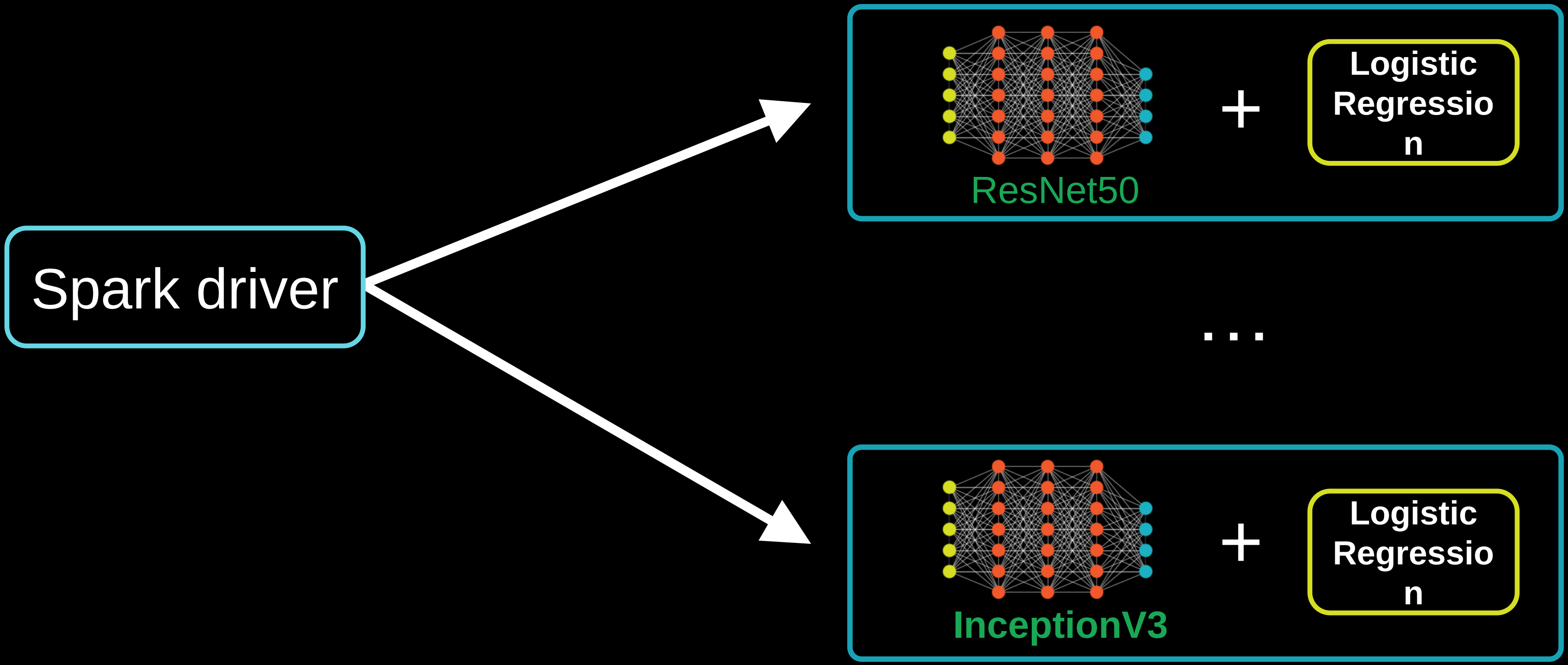
- KerasImageFileEstimator

Deep Learning Estimators

- **Transfer learning Pipelines**

- KerasImageFileEstimator

Distributed Tuning Transfer Learning



databricks

Distributed Tuning Transfer Learning

```
pipeline = Pipeline([DeepImageFeaturizer(), LogisticRegression()])

paramGrid = ( ParamGridBuilder()
     .addGrid(modelName=["ResNet50", "InceptionV3"]) )

cv = CrossValidator(estimator= pipeline,
                    estimatorParamMaps=paramGrid,
                    evaluator=BinaryClassificationEvaluator(),
                    numFolds=3)

best_model = cv.fit(train_df)
```

Deep Learning Estimators

- Transfer learning Pipelines

- **KerasImageFileEstimator**

Keras

- A popular, declarative interface to build DL models
- High level, expressive API in python
- Executes on TensorFlow, Theano, CNTK

```
model = Sequential()
model.add(Dense(32, input_dim=784))
model.add(Activation('relu'))
```

Keras Estimator

```
model = Sequential()
model.add(...)
model.save(model_filename)

estimator = KerasImageFileEstimator(
    kerasOptimizer="adam",
    kerasLoss="categorical_crossentropy",
    kerasFitParams={"batch_size":100},
    modelFile=model_filename)
model = model.fit(dataframe)
```

Keras Estimator in Model Selection

```
estimator = KerasImageFileEstimator(
    kerasOptimizer=“adam“,
    kerasLoss=“categorical_crossentropy“,
    kerasFitParams={“batch_size“:100},
    modelFile=model_filename)
model = model.fit(dataframe)
```

```
estimator = KerasImageFileEstimator(
    kerasOptimizer=“adam“,
    kerasLoss=“categorical_crossentropy“)

paramGrid = ( ParamGridBuilder()
    .addGrid(kerasFitParams=[{“batch_size“:100}, {“batch_size“:200}])
    .addGrid(modelFile=[model1, model2]) )
```

Keras Estimator in Model Selection

```
estimator = KerasImageFileEstimator(
    kerasOptimizer="adam",
    kerasLoss="categorical_crossentropy")

paramGrid = ( ParamGridBuilder()
    .addGrid(kerasFitParams=[{"batch_size":100}, {"batch_size":200}])
    .addGrid(modelFile=[model1, model2]) )
```

```
cv = CrossValidator(estimator=estimator,
                    estimatorParamMaps=paramGrid,
                    evaluator=BinaryClassificationEvaluator(),
                    numFolds=3)

best_model = cv.fit(train_df)
```

Deep Learning Pipelines

- Load data
- Interactive work
- Train
- **Evaluate model**
- **Apply**

Deep Learning Pipelines

- Load data
- Interactive work
- Train
- Evaluate model
- **Apply**
  - Batch prediction
  - Spark SQL

Deep Learning Pipelines

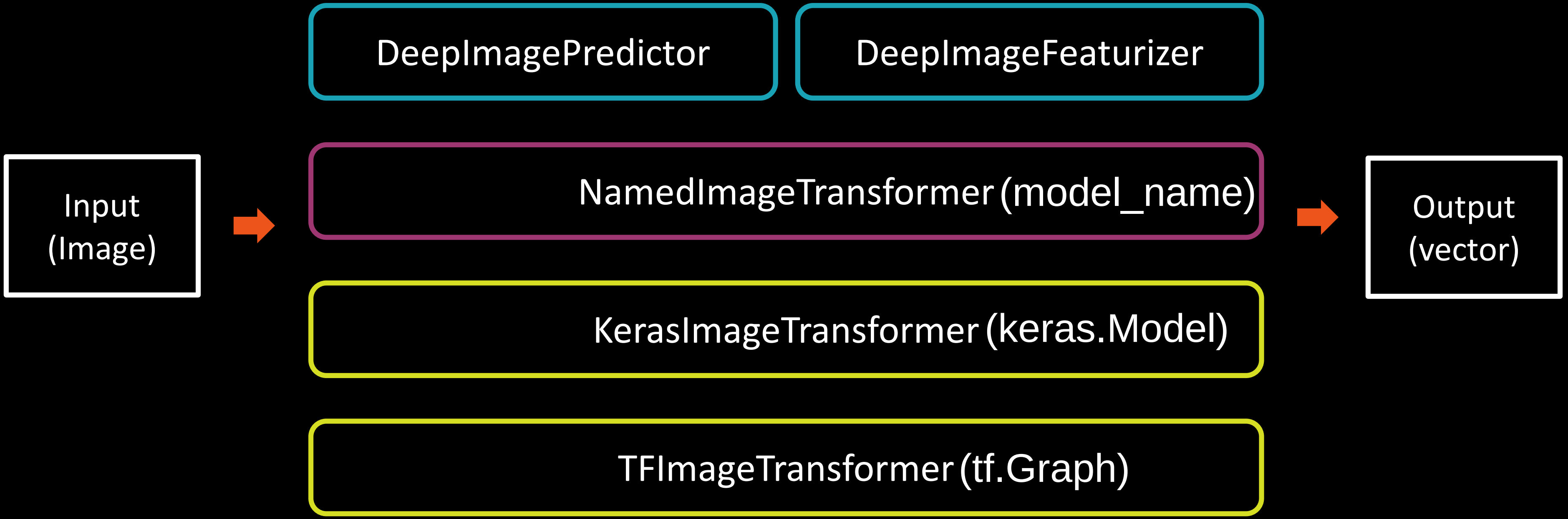
- Load data
- Interactive work
- Train
- Evaluate model
- **Apply**
  - **Batch prediction**
  - Spark SQL

Batch prediction as an MLlib Transformer

- Recall a model is a Transformer in MLlib

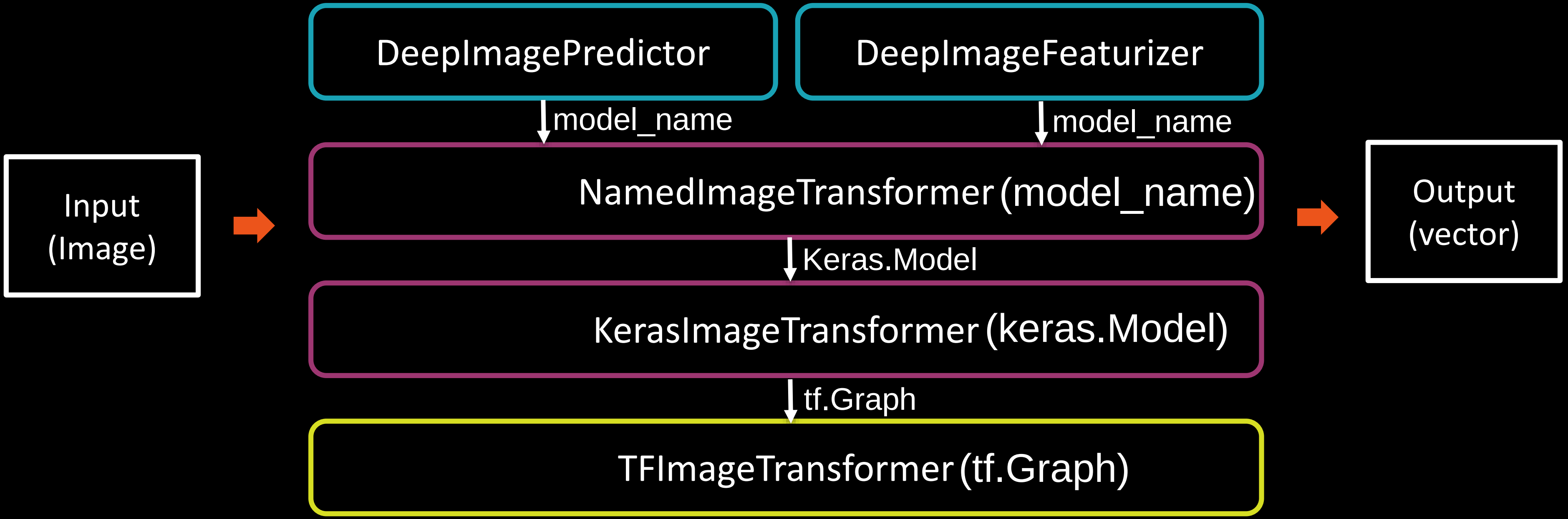
```
predictor = XXTransformer(inputCol="image",
                          outputCol=”predictions",
                          modelSpecification={…})
predictions = predictor.transform(test_df)
```

Hierarchy of DL transformers for images

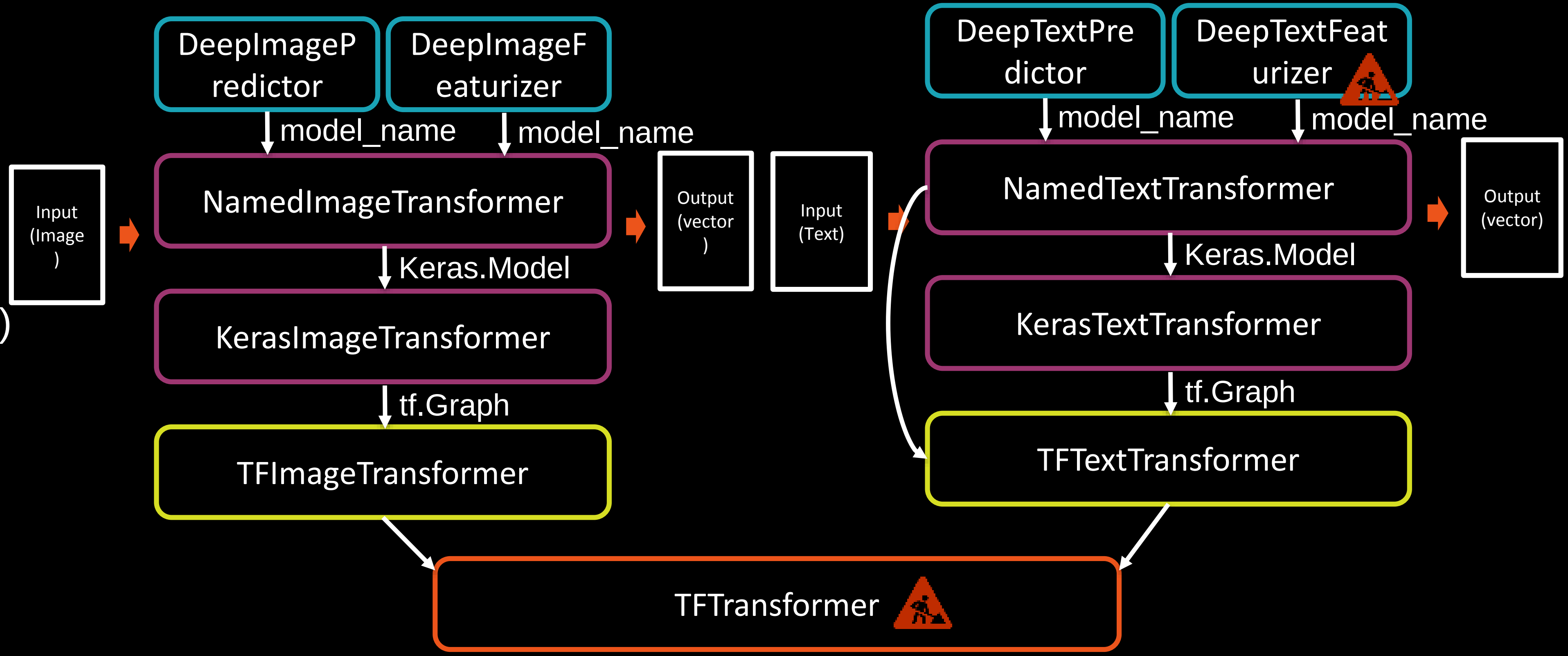


databricks

Hierarchy of DL transformers for images



Hierarchy of DL transformers



databricks

TFImageTransformer

Defined by tf.Graph

1. Convert ImageSchema data into a vector
2. Use tensorframes to
   - Efficiently distribute tf.Graph to workers
   - Apply the graph to the partitioned data
3. Return the result as a spark.ml.Vector

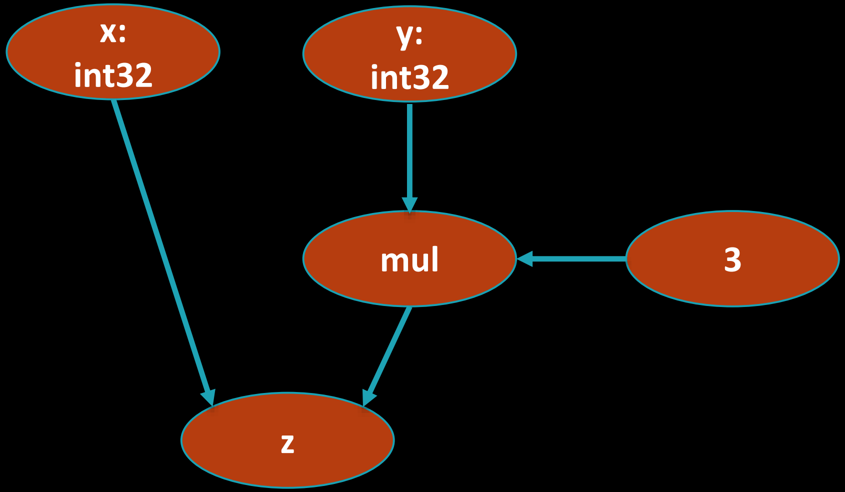
t

Aside: high performance with Spark

- TensorFlow (and other frameworks) have 2 mechanisms to ingest data:
  - memory-based API (tensors)
  - file-based API (Queue, …)
- DLP makes all data transfers in memory
- Spark responsible for reading and assembling data
- Low-level transformations handled by TensorFrames

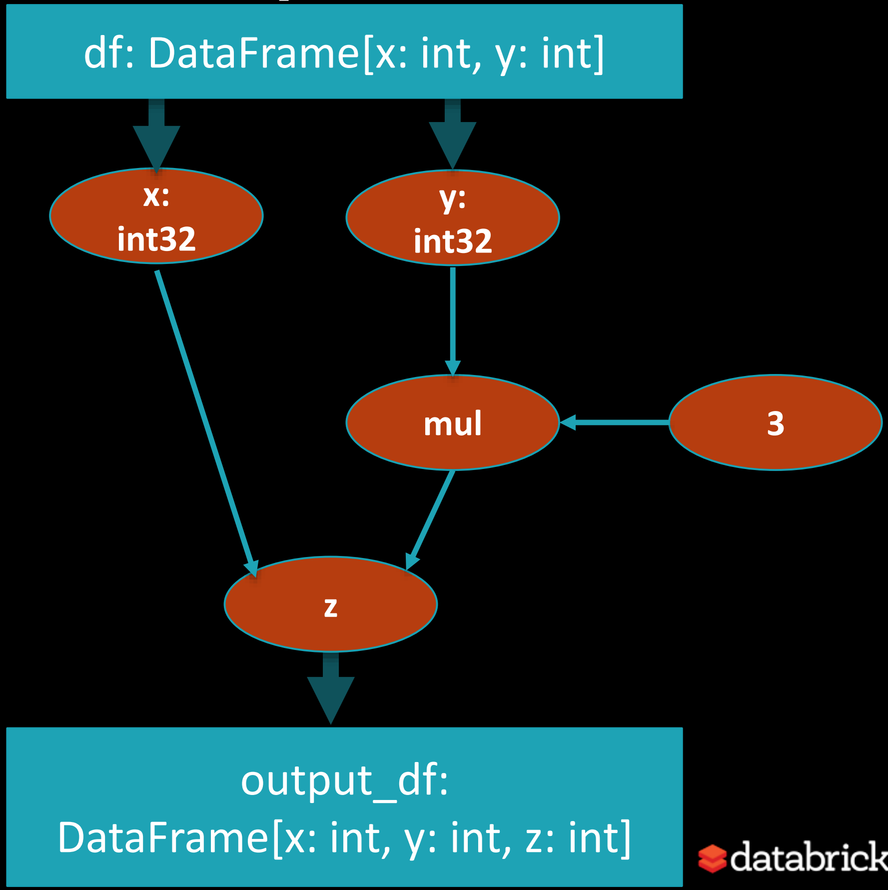
High performance with Spark

- Every DL transform is a TensorFlow graph



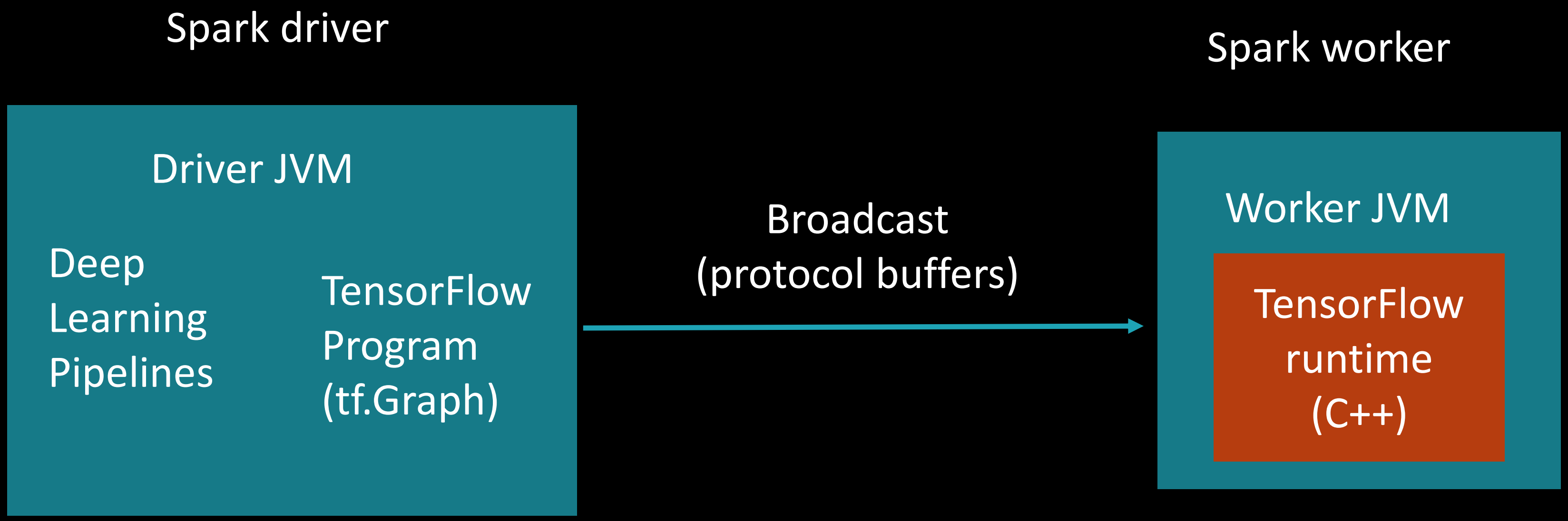
High performance with Spark

- Every DL transform is a TensorFlow graph
- Applied on each of the rows of the dataframe



databricks

Example: running batch inference



s

Deep Learning Pipelines

- Load data
- Interactive work
- Train
- Evaluate model
- **Apply**
  - Batch prediction
  - **Spark SQL**

Shipping predictors in SQL

Take a trained model / Pipeline, register a SQL UDF usable by *anyone* in the organization

```
registerKerasUDF(”my_object_recognition_function",
                 keras_model_file="/mymodels/007model.h5")
```

In Spark SQL:

```
select image, my_object_recognition_function(image) as objects
from traffic_imgs
```

This means you can apply deep learning models in streaming!

databricks

Almost done

Deep Learning Pipelines : Future

In progress

- Scala API for DeepImageFeaturizer
- Text featurization (embeddings)
- TFTransformer for arbitrary vectors

Future

- Distributed training
- Support for more backends, e.g. MXNet, PyTorch, BigDL

Deep Learning without Deep Pockets

- Simple API for Deep Learning, integrated with MLlib
- Scales common tasks with transformers and estimators
- Embeds Deep Learning models in MLlib and SparkSQL
- Check out https://github.com/databricks/spark-deep-learning !

Thank you!

Questions?

Resources

Blog posts & webinars (http://databricks.com/blog)

- Deep Learning Pipelines
- GPU acceleration in Databricks
- BigDL on Databricks
- Deep Learning and Apache Spark

Docs for Deep Learning on Databricks (http://docs.databricks.com)

- Getting started
- Deep Learning Pipelines Example
- Spark integration