



# Deep Learning Pipelines for Apache Spark

CS570 - Big Data Processing and Analytics  
Saron Haile 20069

# Table of contents

**01**

**Introduction**

**02**

**Design**

**03**

**Implementation**

**04**

**Test**

**05**

**Enhancement**

**06**

**Conclusion**

# 01

# Introduction

## Overview

- Deep Learning Pipelines library by Databricks
- High-level APIs for scalable deep learning and transfer learning
- Integration with popular deep learning libraries (TensorFlow, Keras) and Spark MLlib

## Objectives

- Demonstrate how to set up and use Deep Learning Pipelines
- Provide examples of TensorFlow and Keras integration
- Showcase practical applications and enhancements

# Design



02

## Cluster Setup

- **Installation:**
  - Add spark-deep-learning as a Maven coordinate library
  - Attach to the cluster
- **Compatibility:**
  - Spark versions 2.0 or higher
  - CPU or GPU instance types
- **Additional Libraries:**
  - TensorFlow, Keras, h5py via PyPI



**03**

# **Implementation and Testing**

# Step I: Run code on databricks

The screenshot shows the Databricks web interface. On the left is a dark sidebar with navigation options: New, Workspace, Recents, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, Delta Live Tables, Machine Learning, and Experiments. The main area displays a notebook titled 'Deep Learning Pipelines for Apache Spark' in Python. The notebook content includes a title 'Introducing Deep Learning Pipelines for Apache Spark', a paragraph about the library, and a section titled 'Cluster set-up' with a note and a list of requirements.

**databricks** Search data, notebooks, recents, and more... workspace

**Deep Learning Pipelines for Apache Spark** Python ☆

File Edit View Run Help Provide feedback Interrupt Starting Schedule Share

## Introducing Deep Learning Pipelines for Apache Spark

Deep Learning Pipelines is a new library published by Databricks to provide high-level APIs for scalable deep learning model application and transfer learning via integration of popular deep learning libraries with MLlib Pipelines and Spark SQL. For an overview and the philosophy behind the library, check out the Databricks [blog post](#). This notebook parallels the [Deep Learning Pipelines README](#), detailing usage examples with additional tips for getting started with the library on Databricks.

### Cluster set-up

Deep Learning Pipelines is available as a Spark Package. To use it on your cluster, create a new library with the Source option "Maven Coordinate", using "Search Spark Packages and Maven Central" to find "spark-deep-learning". Then [attach the library to a cluster](#).

**Note:**

- This notebook works with spark-deep-learning release `0.1.0-spark2.1-s_2.11`. Future releases may introduce breaking changes. Please refer to the project's [github page](#) for the latest examples and docs.
- To run this notebook, also create and attach the following libraries via PyPI: tensorflow, keras, h5py.

Deep Learning Pipelines is compatible with Spark versions 2.0 or higher and works with any instance type (CPU or GPU).

# Step 1: Run code on databricks

Let us first get some images to work with in this notebook. We'll use the flowers dataset from the [TensorFlow retraining tutorial](#).

Waiting Starting Spark

5

```
%sh
curl -O http://download.tensorflow.org/example_images/flower_photos.tgz
tar xzf flower_photos.tgz
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload Upload	Total	Spent	Left	Speed
0	0	0	0	0	0	--:--:--	0
0	0	0	0	0	0	--:--:--	0
12	218M	12 28.0M	0	0	24.0M	0 0:00:09 0:00:01 0:00:08	24.0M
100	218M	100 218M	0	0	114M	0 0:00:01 0:00:01 --:--:--	114M

```
tar: flower_photos/roses/14810868100_87eb739f26_m.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/1446090416_f0cad5fde4.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/15319767030_e6c5602a77_m.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/15032112248_30c5284e54_n.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/7211616670_2d49ecb3a5_m.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/15674450867_0ced942941_n.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/17158274118_00ec99a23c.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/14019883858_e5d2a0ec10_n.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/8035908422_87220425d2_n.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/14747962886_2bfff6bb323_m.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/4356781875_92c5cd93c0.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/8524505546_b242bd4928_n.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/9406573080_60eab9278e_n.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
tar: flower_photos/roses/6039330368_c30ed224c4_m.jpg: Cannot change ownership to uid 270850, gid 5000: Invalid argument
```

# Step 1: Run code on databricks

<>

+ Code + Text

Waiting 6

```
display(dbutils.fs.ls('file:/databricks/driver/flower_photos'))
```

Table ▾ +

🔍 🏠

	A <sup>B</sup> <sub>C</sub> path	A <sup>B</sup> <sub>C</sub> name	1 <sup>2</sup> <sub>3</sub> size
1	file:/databricks/driver/flower_photos/daisy/	daisy/	32768
2	file:/databricks/driver/flower_photos/dandelion/	dandelion/	49152
3	file:/databricks/driver/flower_photos/sunflowers/	sunflowers/	36864
4	file:/databricks/driver/flower_photos/LICENSE.txt	LICENSE.txt	418049
5	file:/databricks/driver/flower_photos/tulips/	tulips/	40960
6	file:/databricks/driver/flower_photos/roses/	roses/	36864

⌵

↓ 6 rows



## Step I: Run code on databricks

Waiting

7

# The 'file:/...' directory will be cleared out upon cluster termination. That doesn't matter for this example notebook, but in most cases we'd want to store the images in a more permanent place. Let's move the files to dbfs so we can see how to work with it in the use cases below.

```
img_dir = '/tmp/flower_photos'
dbutils.fs.mkdirs(img_dir)
dbutils.fs.cp('file:/databricks/driver/flower_photos/tulips', img_dir + "/tulips", recurse=True)
dbutils.fs.cp('file:/databricks/driver/flower_photos/daisy', img_dir + "/daisy", recurse=True)
dbutils.fs.cp('file:/databricks/driver/flower_photos/LICENSE.txt', img_dir)
display(dbutils.fs.ls(img_dir))
```

Table

	path	name	size
1	dbfs:/tmp/flower_photos/LICENSE.txt	LICENSE.txt	418049
2	dbfs:/tmp/flower_photos/daisy/	daisy/	0
3	dbfs:/tmp/flower_photos/tulips/	tulips/	0

3 rows

## Step I: Run code on databricks

Waiting

8

```
# Let's create a small sample set of images for quick demonstrations.
sample_img_dir = img_dir + "/sample"
dbutils.fs.mkdirs(sample_img_dir)
files = dbutils.fs.ls(img_dir + "/tulips")[0:1] + dbutils.fs.ls(img_dir + "/daisy")[0:2]
for f in files:
    dbutils.fs.cp(f.path, sample_img_dir)
display(dbutils.fs.ls(sample_img_dir))
```

Table

	$A_C^B$ path	$A_C^B$ name	$i_3^2$ size
1	dbfs:/tmp/flower_photos/sample/100080576_f52e8ee070_n.jpg	100080576_f52e8ee070_n.jpg	26797
2	dbfs:/tmp/flower_photos/sample/100930342_92e8746431_n.jpg	100930342_92e8746431_n.jpg	26200
3	dbfs:/tmp/flower_photos/sample/10140303196_b88d3d6cec.jpg	10140303196_b88d3d6cec.jpg	117247

3 rows

## Step 2: Working with images in Spark

**Deep Learning Pipelines for Apache Spark** Python ☆

File Edit View Run Help [Provide feedback](#)

Interrupt S Y's Personal Compu... Schedule

Working with images in Spark

The first step to applying deep learning on images is the ability to load the images. Deep Learning Pipelines includes utility functions that can load millions of images into a Spark DataFrame and decode them automatically in a distributed fashion, allowing manipulation at scale.

Waiting 10

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

image\_df: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct]

/databricks/python/local/lib/python2.7/site-packages/h5py/\_\_init\_\_.py:36: FutureWarning: Conversion of the second argument of issubclass from 'float' to 'np.floating' is deprecated. In future, it will be treated as 'np.float64 == np.dtype(float).type'.

from .\_conv import register\_converters as \_register\_converters

Using TensorFlow backend.

## Step 2: Working with images in Spark



The resulting DataFrame contains a string column named "filePath" containing the path to each image file, and a image struct (" SpImage ") column called "image" containing the decoded image data.



Waiting

12

```
display(image_df)
```

Table

	filePath	image
1	dbfs:/tmp/flower_photos/sample/100080576_f52e8ee070_n.jpg	> {"mode":"RGB","height":263,"width":320,"nChannels":3,"data":"h4eFioqljo6OkZGRkpKSk5C
2	dbfs:/tmp/flower_photos/sample/100930342_92e8746431_n.jpg	> {"mode":"RGB","height":209,"width":320,"nChannels":3,"data":"Ey4PEC8QDi8QDi8SES4SE

Download

2+ rows | Truncated data due to byte limit

# Step 3: Transfer Learning

## Deep Learning Pipelines for Apache Spark

Python



File Edit View Run Help [Provide feedback](#)

Interrupt

S Y's Personal Compu...

Schedule

Share

Waiting

14

```
# Create training & test DataFrames for transfer learning - this piece of code is longer than transfer learning itself
below!
from sparkdl import readImages
from pyspark.sql.functions import lit

tulips_df = readImages(img_dir + "/tulips").withColumn("label", lit(1))
daisy_df = readImages(img_dir + "/daisy").withColumn("label", lit(0))
tulips_train, tulips_test, _ = tulips_df.randomSplit([0.05, 0.05, 0.9]) # use larger training sets (e.g. [0.6, 0.4] for
non-community edition clusters)
daisy_train, daisy_test, _ = daisy_df.randomSplit([0.05, 0.05, 0.9]) # use larger training sets (e.g. [0.6, 0.4] for
non-community edition clusters)
train_df = tulips_train.unionAll(daisy_train)
test_df = tulips_test.unionAll(daisy_test)

# Under the hood, each of the partitions is fully loaded in memory, which may be expensive.
# This ensure that each of the paritions has a small size.
train_df = train_df.repartition(100)
test_df = test_df.repartition(100)
```

- ▶ tulips\_df: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct ... 1 more field]
- ▶ daisy\_df: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct ... 1 more field]
- ▶ tulips\_train: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct ... 1 more field]
- ▶ tulips\_test: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct ... 1 more field]
- ▶ daisy\_train: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct ... 1 more field]
- ▶ daisy\_test: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct ... 1 more field]
- ▶ \_: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct ... 1 more field]
- ▶ train\_df: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct ... 1 more field]
- ▶ test\_df: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct ... 1 more field]

## Step 3: Transfer Learning

Deep Learning Pipelines for Apache Spark

File Edit View Run Help Provide feedback

Python ☆

Interrupt

Waiting

```
from pyspark.ml.classifi
from pyspark.ml import P
from sparkdl import Deep

featurizer = DeepImageFe
lr = LogisticRegression(
p = Pipeline(stages=[fea

p_model = p.fit(train_df

Downloading data from https
ering_tf_kernels_notop.h5

16384/87910968 [.....
24576/87910968 [.....
57344/87910968 [.....
73728/87910968 [.....
106496/87910968 [.....
139264/87910968 [.....
196608/87910968 [.....
262144/87910968 [.....
352256/87910968 [.....
458752/87910968 [.....
557056/87910968 [.....
696320/87910968 [.....
860160/87910968 [.....
1032192/87910968 [.....
1220608/87910968 [.....
1433600/87910968 [.....
1671168/87910968 [.....
1933312/87910968 [.....
```

Note: the training step may take a while on Community Edition - try making a smaller train

< > + Code + Text

Let's see how well the model does:

Waiting

18

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

tested_df = p_model.transform(test_df)
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
print("Test set accuracy = " + str(evaluator.evaluate(tested_df.select("predict

tested_df: pyspark.sql.dataframe.DataFrame
```

# Step 3: Transfer Learning

## Deep Learning Pipelines for Apache Spark

Python



File Edit View Run Help [Provide feedback](#)

Interrupt

S Y's Personal Compu..



Note: the training step may take a while on Community Edition - try making a smaller training set in that case.



+ Code

+ Text

Let's see how well the model does:

Waiting

18

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

tested_df = p_model.transform(test_df)
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
print("Test set accuracy = " + str(evaluator.evaluate(tested_df.select("prediction", "label"))))
```

tested\_df: pyspark.sql.dataframe.DataFrame

```
INFO:tensorflow:Froze 376 variables.
Converted 376 variables to const ops.
INFO:tensorflow:Froze 0 variables.
Converted 0 variables to const ops.
Test set accuracy = 0.971014492754
```

# Step 3: Transfer Learning

Deep Learning Pipelines for Apache SparkPython ☆

File Edit View Run Help Provide feedback

e/Users/saron54lo@gmail.com/Deep Learning Pipelines for Apache Spark

Waiting20

```
from pyspark.sql.types import DoubleType
from pyspark.sql.functions import expr
def _p1(v):
    return float(v.array[1])
p1 = udf(_p1, DoubleType())

df = tested_df.withColumn("p_1", p1(tested_df.probability))
wrong_df = df.orderBy(expr("abs(p_1 - label)", ascending=False))
display(wrong_df.select("filePath", "p_1", "label").limit(10))
```

df: pyspark.sql.dataframe.DataFrame

wrong\_df: pyspark.sql.dataframe.DataFrame

Table +

	filePath	p_1	label
1	dbfs:/tmp/flower_photos/daisy/9345273630_af3550031d.jpg	0.804202936186138	0
2	dbfs:/tmp/flower_photos/daisy/530738000_4df7e4786b.jpg	0.6165413243876156	0
3	dbfs:/tmp/flower_photos/tulips/113902743_8f537f769b_n.jpg	0.5153939149058596	1
4	dbfs:/tmp/flower_photos/tulips/14053292975_fdc1093571_n.jpg	0.5376996385350966	1
5	dbfs:/tmp/flower_photos/tulips/2294116183_a30d2aa2c1_m.jpg	0.6245144314249896	1
6	dbfs:/tmp/flower_photos/daisy/15813862117_dedcd1c56f_m.jpg	0.3507493964314712	0
7	dbfs:/tmp/flower_photos/tulips/14674071872_2df55466d5_m.jpg	0.6689220612741658	1
8	dbfs:/tmp/flower_photos/tulips/4591323356_030d8b6967_m.jpg	0.6965201230884019	1
9	dbfs:/tmp/flower_photos/tulips/4590703575_6371c0a186_n.jpg	0.7213367030778205	1
10	dbfs:/tmp/flower_photos/tulips/7166567320_0a2beb6d42.jpg	0.7230501393191048	1

10 rows



Deep Learning Pipelines for Apache Spark

Python

File Edit View Run Help

/Users/saron54lo@gmail.com/Deep Learning Pipelines for Apache Spark

Interrupt

S Y's Personal Compu... ▾

Schedule

# Applying popular image models

There are many well-known deep learning models for images. If the task at hand is very similar to what the models provide (e.g. object recognition with ImageNet classes), or for pure exploration, one can use the Transformer `DeepImagePredictor` by simply specifying the model name.

Waiting

23

```
from sparkdl import readImages, DeepImagePredictor

image_df = readImages(sample_img_dir)

predictor = DeepImagePredictor(inputCol="image", outputCol="predicted_labels", modelName="InceptionV3",
                                decodePredictions=True, topK=10)
predictions_df = predictor.transform(image_df)

display(predictions_df.select("filePath", "predicted_labels"))
```

▶ image\_df: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct]

▶ predictions\_df: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct ... 1 more field]

	filePath	predicted_labels
1	dbfs:/tmp/flower_photos/sample/100080576_f52e8ee070_n.jpg	> [{"class": "n11939491", "description": "daisy", "probability": 0.8805494}, {"class": "n02219486", "description": "petaluma rock", "probability": 0.0001001}]
2	dbfs:/tmp/flower_photos/sample/100930342_92e8746431_n.jpg	> [{"class": "n03930313", "description": "picket_fence", "probability": 0.18473865}, {"class": "n11939491", "description": "daisy", "probability": 0.0001001}]
3	dbfs:/tmp/flower_photos/sample/10140303196_b88d3d6cec.jpg	> [{"class": "n11939491", "description": "daisy", "probability": 0.9535933}, {"class": "n02219486", "description": "petaluma rock", "probability": 0.0001001}]

3 rows

## Step 4: Applying Deep learning Models at scale

### Deep Learning Pipelines for Apache Spark

Python ☆

Interrupt S Y's Personal Compu... Schedule

File Edit View Run Help [Provide feedback](#)

Notice that the `predicted_labels` column shows "daisy" as a high probability class for all sample flowers using this base model. However, as can be seen from the differences in the probability values, the neural network has the information to discern the two flower types. Hence our transfer learning example above was able to properly learn the differences between daisies and tulips starting from the base model.

Waiting 25

```
df = p_model.transform(image_df)
display(df.select("filePath", (1-p1(df.probability)).alias("p_daisy")))
```

df: pyspark.sql.dataframe.DataFrame

Table +

	filePath	1.2 p_daisy
1	dbfs:/tmp/flower_photos/sample/100080576_f52e8ee070_n.jpg	0.968875532556084
2	dbfs:/tmp/flower_photos/sample/100930342_92e8746431_n.jpg	0.13976502414604564
3	dbfs:/tmp/flower_photos/sample/10140303196_b88d3d6cec.jpg	0.9786112803439984

3 rows

# Step 4: Applying Deep learning Models at scale

**Deep Learning Pipelines for Apache Spark** Python ☆

File Edit View Run Help Provide feedback

Interrupt S Y's Personal Compu... Schedule Sha

**For TensorFlow users**

Deep Learning Pipelines provides a MLlib Transformer that will apply the given TensorFlow Graph to a DataFrame containing a column of images (e.g. loaded using the utilities described in the previous section). Here is a very simple example of how a TensorFlow Graph can be used with the Transformer. In practice, the TensorFlow Graph will likely be restored from files before calling `TFImageTransformer`.

Waiting 27

```
from sparkdl import readImages, TFImageTransformer
from sparkdl.transformers import utils
import tensorflow as tf

image_df = readImages(sample_img_dir)

g = tf.Graph()
with g.as_default():
    image_arr = utils.imageInputPlaceholder()
    resized_images = tf.image.resize_images(image_arr, (299, 299))
    # the following step is not necessary for this graph, but can be for graphs with variables, etc
    frozen_graph = utils.stripAndFreezeGraph(g.as_graph_def(add_shapes=True), tf.Session(graph=g), [resized_images])

transformer = TFImageTransformer(inputCol="image", outputCol="transformed_img", graph=frozen_graph,
                                inputTensor=image_arr, outputTensor=resized_images,
                                outputMode="image")
tf_trans_df = transformer.transform(image_df)
```

image\_df: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct]

tf\_trans\_df: pyspark.sql.dataframe.DataFrame = [filePath: string, image: struct ... 1 more field]

INFO:tensorflow:Froze 0 variables.  
Converted 0 variables to const ops.  
INFO:tensorflow:Froze 0 variables.  
Converted 0 variables to const ops.

# Step 4: Applying Deep learning Models at scale

## Deep Learning Pipelines for Apache Spark

Python



Interrupt

S Y's Personal Compu...

Schedule

File Edit View Run Help Provide feedback

### For Keras users

For applying Keras models in a distributed manner using Spark, [KerasImageFileTransformer](#) works on TensorFlow-backed Keras models. It

- Internally creates a DataFrame containing a column of images by applying the user-specified image loading and processing function to the input DataFrame containing a column of image URIs
- Loads a Keras model from the given model file path
- Applies the model to the image DataFrame

The difference in the API from `TFImageTransformer` above stems from the fact that usual Keras workflows have very specific ways to load and resize images that are not part of the TensorFlow Graph.

To use the transformer, we first need to have a Keras model stored as a file. For this notebook we'll just save the Keras built-in InceptionV3 model instead of training one.

Waiting

30

```
from keras.applications import InceptionV3

model = InceptionV3(weights="imagenet")
model.save('/tmp/model-full.h5') # saves to the local filesystem
# move to a permanent place for future use
dbfs_model_path = 'dbfs:/models/model-full.h5'
dbutils.fs.cp('file:/tmp/model-full.h5', dbfs_model_path)
```

Out[14]: True

## Step 4: Applying Deep learning Models at scale

**Deep Learning Pipelines for Apache Spark** Python ☆

File Edit View Run Help [Provide feedback](#)

Interrupt S Y's Personal Compu... Schedule Share

Waiting 32

```
from keras.applications.inception_v3 import preprocess_input
from keras.preprocessing.image import img_to_array, load_img
import numpy as np
from pyspark.sql.types import StringType
from sparkdl import KerasImageFileTransformer

def loadAndPreprocessKerasInceptionV3(uri):
    # this is a typical way to load and prep images in keras
    image = img_to_array(load_img(uri, target_size=(299, 299))) # image dimensions for InceptionV3
    image = np.expand_dims(image, axis=0)
    return preprocess_input(image)

dbutils.fs.cp(dbfs_model_path, 'file:/tmp/model-full-tmp.h5')
transformer = KerasImageFileTransformer(inputCol="uri", outputCol="predictions",
                                       modelFile='/tmp/model-full-tmp.h5', # local file path for model
                                       imageLoader=loadAndPreprocessKerasInceptionV3,
                                       outputMode="vector")

files = ["dbfs" + str(f.path)[5:] for f in dbutils.fs.ls(sample_img_dir)] # make "local" file paths for images
uri_df = sqlContext.createDataFrame(files, StringType()).toDF("uri")

keras_pred_df = transformer.transform(uri_df)
```

uri\_df: pyspark.sql.dataframe.DataFrame = [uri: string]

keras\_pred\_df: pyspark.sql.dataframe.DataFrame

/databricks/python/local/lib/python2.7/site-packages/keras/models.py:255: UserWarning: No training configuration found in save file: the model was \*not\* compiled. Compile it manually.  
warnings.warn('No training configuration found in save file: '  
INFO:tensorflow:Froze 378 variables.  
Converted 378 variables to const ops.  
INFO:tensorflow:Froze 0 variables.  
Converted 0 variables to const ops.

## Step 4: Applying Deep learning Models at scale

<>

+ Code + Text

Waiting

33

```
display(keras_pred_df.select("uri", "predictions"))
```

Table

+

Q Y

	uri	predictions
1	/dbfs/tmp/flower_photos/sample/100080576_f52e8ee070_n.jpg	> [1,1000,[],[0.00007469155389117077,0.00007630845357198268,0.000193578365724
2	/dbfs/tmp/flower_photos/sample/100930342_92e8746431_n.jpg	> [1,1000,[],[0.0002563267189543694,0.0028356011025607586,0.0001203265346703
3	/dbfs/tmp/flower_photos/sample/10140303196_b88d3d6cec.jpg	> [1,1000,[],[0.00003744077548617497,0.000053084160754224285,0.0000970438049

3 rows

## Step 5: Cleanup Data Generated for this notebook

### Clean up data generated for this notebook

Waiting

35

```
dbutils.fs.rm(img_dir, recurse=True)
dbutils.fs.rm(dbfs_model_path)
```

Out[17]: True

A decorative graphic in the top-left corner consisting of a cluster of white-outlined hexagons of various sizes, some of which are slightly offset to create a 3D effect.

# Enhancement

A black, multi-pointed starburst or explosion-like shape that frames the number 05.

**05**

- **SQL function deployment (upcoming feature)**
- **Distributed hyper-parameter tuning (upcoming feature)**
- **Integration with additional deep learning frameworks**



# Conclusion



06

## Summary:

- Deep Learning Pipelines streamline deep learning model deployment on Spark
- Integration with TensorFlow and Keras provides flexibility
- Practical applications in image processing and beyond

## Next Steps:

- Explore additional features and updates on the Deep Learning Pipelines GitHub page
- Stay updated with new releases and enhancements

# GITHUB LINK

<https://github.com/Sharon20222/Cloud-Computing/tree/main/Machine%20Learning/Apache%20Spark%20+%20Deep%20Learning>

# References

07

- <https://github.com/databricks/spark-deep-learning>
- <https://docs.databricks.com/en/machine-learning/train-model/deep-learning.html>
- <https://www.databricks.com/blog/2017/06/06/databricks-vision-simplify-large-scale-deep-learning.html>