

TikaAndVisionDL4J

Tika and Computer Vision powered by Deeplearning4j(DL4J)

- [Tika and Computer Vision powered by Deeplearning4j\(DL4J\)](#)
- [Java/Groovy/Scala example](#)
- [For Python and Tika Server users](#)
- [Large Scale Image Recognition In Spark](#)
- [Configuration options](#)
- [Questions / Suggestions / Improvements / Feedback ?](#)

This page describes a way to make Tika perform image recognition. Tika has many implementations of image recognition parsers. Specifically, this page provides information for an implementation powered by [Deeplearning4j](#), [InceptionNet](#)-V3 model pre-trained on [ImageNet](#) dataset and now with [TIKA-2298](#), the VGG-16 model. Both models can detect a thousand different objects in the images.

The advantage of this particular setting is, this implementation runs inside a Java Virtual Machine (JVM) stack without dependence on any external services. So it is perfect for the users who are trying to run image recognition on a distributed setup like Apache Hadoop or Apache Spark.

Note:

1. This is a work in progress. Inception-V3 was added in Tika 1.15 and VGG-16 in 1.16. 2. At the time of writing, Tika 1.16 was not released. You have to [clone Tika repository](<https://github.com/apache/tika>) and do **mvn clean install**. 3. The rest of the page uses version **1.15-SNAPSHOT**, however, if you are reading this after release, please use **1.15**, **1.16** or newer version.

Java/Groovy/Scala example

For maven users:

Add Tika-parsers and tika-dl to your project

Here is an example for Apache Maven users:

```
{{{#!highlight xml
```

```
<dependencies>
<dependency>
<groupId>org.apache.tika</groupId>
<artifactId>tika-parsers</artifactId>
<version>1.15-SNAPSHOT</version>
</dependency>
<dependency>
<groupId>org.apache.tika</groupId>
<artifactId>tika-dl</artifactId>
<version>1.15-SNAPSHOT</version>
</dependency>
</dependencies>
}}}
```

A configuration, tika-config.xml, to activate Inception-V3 image recognition model:

```
{{{#!highlight xml
```

```
<properties>
<parsers>
<parser class="org.apache.tika.parser.recognition.ObjectRecognitionParser">
<mime>image/jpeg</mime>
<params>
<param name="topN" type="int">10</param>
<param name="minConfidence" type="double">0.015</param>
<param name="class" type="string">org.apache.tika.dl.imagerec.DL4JInceptionV3Net</param>
</params>
</parser>
</parsers>
</properties>
}}}
```

A configuration, tika-config.xml, to activate VGG-16 image recognition model:

```

{{{#!highlight xml
<?xml version="1.0" encoding="UTF-8"?>
<properties>
<parsers>
<parser class="org.apache.tika.parser.recognition.ObjectRecognitionParser">
<mime>image/jpeg</mime>
<params>
<param name="topN" type="int">2</param>
<param name="minConfidence" type="double">0.015</param>
<param name="class" type="string">org.apache.tika.dl.imagerec.DL4JVG16Net</param>
<param name="modelType" type="string">VGG16</param>
<param name="serialize" type="string">yes</param>
</params>
</parser>
</parsers>
</properties>
}}}
```

Note: Refer to a later section for customizing the config.

Sample Java code:

```

{{
// Retrieve meta-data from the metadata system
getClosestLoader().getResourceAsStream("tika-config.xml").openStream().use {TikaConfigFactory::fromStream}
// Create a TikaConfig object
TikaConfig config = new TikaConfigFactory().fromStream(inputStream).create();
// Create a File object
File file = new File("/data/sample.jpg");
// Create a Metadata object
Metadata meta = new Metadata();
// Parse the image file
Parser.parse(imageFile, meta); //retrieve objects from the metadata System.out.
println(Arrays.toString(meta.getValues("OBJECT"))); // This should print}
}
```

For Python and Tika Server users

1. Create file with name **tika-config.xml** by using the content shown above. 2. Clone tika repository and build the tika project 3. Add **tika-dl** to classpath of tika server

```

{{{#!highlight bash
alias tika="java -cp tika-server/target/tika-server-1.15-SNAPSHOT.jar:tika-dl/target/tika-dl-1.15-SNAPSHOT-jar-with-dependencies.jar org.apache.tika.server.TikaServerCli " tika --config=tika-config.xml # Starts server port 9998
}}}
```

Refer to [Tika-python](#) for an example usage.

Large Scale Image Recognition In Spark

Coming soon! It is being tested here <https://github.com/thammegowda/tika-dl4j-spark-imgrec>

Configuration options

Previously, we have used the three parameters to this parser:

```

{{{#!highlight xml
<params>
<param name="topN" type="int">10</param>
<param name="minConfidence" type="double">0.015</param>
<param name="class" type="string">org.apache.tika.dl.imagerec.DL4JInceptionV3Net</param>
</params>
}}}
```

The other important parameters are:

```

{{{#!highlight xml
<param name="modelWeightsPath" type="string">VALUE</param>
<param name="modelJsonPath" type="string">VALUE</param>
}}}
```

The VALUE string can be:

1. File path relative to class loader (note that these could also be inside jar files) 2. Absolute file path anywhere on the local filesystem 3. Any remote URL, including HTTP or HTTPS

For example:

```
1. <param name="modelWeightsPath" type="string">inception-model-weights.h5</param> 2. <param name="
modelWeightsPath" type="string">/usr/share/apache-tika/models/tikainception-model-weights.h5</param> 3. <para
m name="modelWeightsPath" type="string">https://myserver.com/files/apache-tika/models/tikainception-model-weights.h5</param>
```

Questions / Suggestions / Improvements / Feedback ?

1. If it was useful, let us know on twitter by mentioning [@ApacheTika](#) 2. If you have questions, let us know by [using Mailing Lists](#) 3. If you find any bugs, [use Jira to report them](#)