*International Journal of Engineering & Technology*, 7 (3.6) (2018) 229-233

**International Journal of Engineering & Technology**

*Website: www.sciencepubco.com/index.php/IJET*

***Research paper***

# Object Recognition Using Keras with Backend Tensor Flow

Raswitha Bandi [1,2]*, J Amudhavel[1]

[1] Department Of Computer Science And Engineering, K L University, Guntur, Andhra Pradesh.
[2] Department Of Information Technology, MLR Institute Of Technology, Hyderabad, India.
* *E-Mail:Raswitha.29reddy@Gmail.Com*

**Abstract**

Now a day's Machine Learning Plays an important role in computer vision, object recognition and image classification. Recognizing objects in images is an interesting thing, this recognization can be done easily by human beings but the computer cannot. The Problem with traditional neural networks is object recognition. So, to avoid difficulties in recognition of objects in images the deep neural networks especially Tensor flow under Keras Library is used and it will improve the Accuracy while recognizing objects. In this paper we present object recognition using Keras Library with backend Tensor flow.

***Keywords***: *Machine Learning, Deep neural networks, Tensor flow, Keras Library.*

## 1. Introduction

[1]Machine learning is about teaching computers to learn from Knowledge or predictions. For true machine learning, it is able to learn to spot patterns and it is programmed expressly. The different names and buzz words of machine Learning are Data Science, Big Data, AI, Analytics, Statistics, data processing etc.
While machine learning will heavily overlap with those fields, it should not beartlessly lumped at the side of them. As an example, machine learning is one tool for knowledge science. It is also one use of infrastructure that may handle huge knowledge. [1]The Different types of Machine learning algorithms are:
• Supervised Learning – In this type of Learning it is providing solutions to each and every Problem.

• Unsupervised Learning –In this type of Learning it is going to inferring a function to describe hidden structure from unlabeled data.

• Reinforcement Learning –This is mainly focus on Performance and which involves finding a balance between uncharted territory and current knowledge.

In these years Machine Learning will be used massively by data scientists and Business analysts because of two reasons [1]

1) Global Demand
Demand for Machine Learning is increased due to its future prediction
2) Power of Data
Data is transformed everywhere and everything we do. so, data is very important to reshape the technology and business.

### 1.1 Deep Learning

[2]Deep learning is the subfield of machine learning and computer science based on Learning data Representations. The major breakthrough of deep learning in the subjects of computer vision, audio process, and even self-driving cars.

[2]Deep learning is all the fashion these days, as corporations across industries ask for to use advanced process techniques to seek out helpful information hidden across knowledge. Much of the Deep Learning work happening currently involves application in many specific areas wherever rule-based programming has proven inadequate, like image recognition, speech recognition, and language understanding.

The Packages in deep learning support different types of architectures like feed-forward networks, auto encoders, recurrent neural networks (RNNs) and convolution neural networks (CNNs) [2].

By using the above mentioned packages the deep learning algorithms will recognize the objects in images very accurately, to do this object recognition deep learning uses popular python libraries. There are five python deep learning libraries that are most useful and popular [2].

They are

1) Theano [2]: It is a low level library and specializes in efficient computation, if the user need is to customize and flexible environment.
2) Tensor Flow: It is another low level library and it is supported by Google and it offers Distributed computing.
3) Lasagne[2]: It is a lightweight wrapper for Theano. Utilize this if require the adaptability of Theano however would prefer not to dependably compose neural system layers without any preparation.

4) Keras[2]: It is a heavyweight wrapper for both Theano and Tensor flow. It's moderate, particular, and amazing for quick experimentation. This is our most loved Python library for profound learning and the best place to begin for fledglings.
5) MXNet [2]: It is another high-level library like Keras. It offers ties for various dialects and support for Distributed Processing.

In this paper we focus mainly on heavy weight wrapper python Library Keras. In the first section we describe the introduction to Keras Library. In the second section we describe how to install and implement keras library with backend tensor flow. In the third section we implement object recognition using keras library with the backend tensor flow with the help of sample image dataset.

## 2. Introduction to Keras

[3]Keras is an High-level neural systems API, written in Python and fit for running over TensorFlow, CNTK, or Theano. It was created with an attention on empowering quick experimentation.
[3]Keras is a moderate Python library for Deep Learning that can keep running over Theano or TensorFlow.
It was created to make executing Deep learning models as quick and simple as feasible for innovative work.
[3]It keeps running on Python 2.7 or 3.5 and can flawlessly execute on GPUs and CPUs given the basic structures.
Utilize Keras[3] on the off chance that you require a Distributed learning library that includes the following Principles:

Takes into account simple and quick prototyping (through ease of use, measured quality, and extensibility).

Backings both convolution systems and repetitive systems, and also combination of the two systems.

Runs flawlessly on CPU and GPU.

### 2.1 Keras Library guiding Principles

[3, 4] Keras was created and kept up by François Chollet, a Google engineer Present four controlling standards:

- User Friendliness [3, 4]: Keras is an API intended for people, not machines. It puts client encounter up front. Keras takes accepted procedures for lessening subjective load: it offers predictable and straightforward APIs, it limits the quantity of client activities required for normal utilize cases, and it gives clear and significant criticism upon client mistake.
- Modularity [3, 4]: A model is comprehended as a grouping or a diagram of independent, completely configurable modules that can be stopped together with as meager limitations as would be prudent. Specifically, neural layers, cost capacities, analyzers, introduction plans, actuation capacities, regularization plans are all independent modules that you can consolidate to make new models.
- Easy extensibility [3, 4]: New modules are easy to include (as new classes and works), and existing modules give plentiful cases. To have the capacity to effortlessly make new modules add up to expressiveness, making Keras appropriate for cutting edge for computer vision.
- Work with Python [3, 4]: No different models setup documents in a revelatory arrangement. Models are depicted in Python code, which is minimal, less demanding to investigate, and takes into consideration simplicity of extensibility.

### 2.2 Organizing Layers in Keras:

[4]In Deep Learning algorithms the most commonly used architectures are convolution neural networks and recurrent neural networks to build hidden layers for the purpose of object recognition, classification and computer vision.
In this paper we present Convolution neural networks for object recognition and classification and it uses simplest model called Sequential model.
[4]In keras to organize layers we use a centralized core data structure called Model.
Basically two types of models used in keras library
1) Sequential model [4]: It is a simplest model consisting of linear stack of layers.
2) Keras Functional API [4]: This is used for Complex architectures, used to build arbitrary graph of layers.

**Sequential model:**
Step by step procedure to evaluate the model in sequential model
1) Import Sequential model from Model
The following command is used to import sequential model from keras models
[4]Import Sequential model = Sequential ();

2) Add stacking layers one by one using add () function.
The following command is used to add layers from keras.layers
[4]Import dense model. Add (dense (units=’units in number ‘, activation=’ specify activation function’, input dimension=’ dimension value’));

3) If the model is good then configure its process with [4].compile () function.
The following command is used to configure the process
[4] Model. Compile ( loss= ‘specify loss function’, optimizer =’ optimizer name’, Metrics =’ specify accuracy‘);
4) To make the model simple you can further configure its process with .compile () function.
5) Iterate On your training data with batches.
The following command is used to train the data
[4] Model.fit (train on x-axis, train on y-axis, epochs= number of epochs,batchsize=’sizeof batch’);

In this we can train the model manually by using the command
Model. Train on batch (batch on x-axis, batch on y-axis);
5) Evaluate the performance of your model.
The following command is used to evaluate the performance
[4]Loss metrics = model. Evaluate (test on x-axis, test on y-axis, size of batch = number);
6) Generate predictions on your data.
The following command is used to generate predictions
[4]Classes1= model. Predict (test on x-axis, size of batch = specify number);

### 2.2 Installing Tensor flow for java

[5][6]To implement keras first we need to install one of its backend engines i.e. tensor flow or theano. In this paper we focused mainly on keras library with tensor flow as backend.
[7][8]Tensor flow provides APIs for use in java programs and these APIs are particularly well suited to loading models created in python and execute them with in a java application.
Tensor flow installation [7][8]:
The following steps used to install tensor flow for java
1) Download libtensorflow.jar
2) Download the Java Native interface file appropriate for tensorflow for java
3) Extract the .zip file

### 2.4 Tensor Flow Backend for Keras[5,6,7,8]

After installing TensorFlow, you can configure the backend by using the following code snippet [5][6]
keras/keras.json

```
{
   "image_dim_ordering": "tf",    "epsilon": 1e-07,    "floatx": "float32", “backend": "tensor flow"
}
```

### 2.5 Object Recognition using Tensor Flow[5][6][7][8]

In this paper we recognize the objects from the images we collected in the dataset.
**Step1:** Collect the dataset from kaggle with images
The sample dataset consists of 2000 Images with different classes like Dogs, cats, nature etc...
**Step2:** Label the images with the features
**Step3:** Train the dataset ,For training from the 2000 images we use only 1500 images
**Step4:** Test the data
For testing the data we use 500 images
**Step5:** Evaluate the data using Language

#### 2.5.1 Creating the project xml file

```
<project>
 <modelVersion>4.0.0</modelVersion>
 <groupId>org.myorg</groupId>
 <artifactId>imgr</artifactId>
 <version>1.0</version>
<properties>
 <exec.mainClass>imgr</exec.mainClass>
   </properties>
   <dependencies>
     <dependency>
       <groupId>org.tensorflow</groupId>
       <artifactId>tensorflow</artifactId>
       <version>1.7.0</version>
     </dependency>
   </dependencies>
 </project>
```

**Create the source file imgr.java**

```
package com;
importjava.io.IOException;
importjava.io.PrintStream;
importjava.nio.charset.Charset;
importjava.nio.file.Files;
importjava.nio.file.Path;
importjava.nio.file.Paths;
importjava.util.Arrays;
importjava.util.List;
importorg.tensorflow.DataType;
importorg.tensorflow.Graph;
importorg.tensorflow.Output;
importorg.tensorflow.Session;
importorg.tensorflow.Tensor;
importorg.tensorflow.TensorFlow;
import org.tensorflow.types.UInt8;
public class imgr {
private static void printUsage(PrintStream s) {
final String url =

"https://storage.googleapis.com/download.tensorflow.org/models/";
s.println(
      "Java program that uses a pre-trained model (http://users/vijaychaitanya/documents/imagefile");
s.println("to label JPEG images.");
s.println("TensorFlow version: " + TensorFlow.version());
s.println();
s.println("Usage: label_image<model dir><image file>");
s.println();
s.println("Where:");
s.println("<model dir> is a directory containing the unzipped contents of the model");
s.println("          (from " + url + ")");
s.println("<image file> is the path to a JPEG image file");
  }
public static void main(String[] args) {
if (args.length != 2) {
printUsage(System.err);
System.exit(1);
   }
   String mod = args[0];
   String img = args[1];

byte[] graphDef = readAllBytesOrExit(Paths.get(mod, "tensorflow_graph1.pb"));
   List<String> labels =
readAllLinesOrExit(Paths.get(mod, "imagenet1_comp_graph_label_strings.txt"));
byte[] imgb = readAllBytesOrExit(Paths.get(img));

try (Tensor<Float> image = construct And Execute Graph To Normalize Image1(imgb)) {
float[] labelProbabilities = executeInceptionGraph1(graphDef1, imagefile);
intbestLabelIdx = maxIndex1(labelProbabilities1);
System.out.println(
String.format("BEST MATCH: %s (%.2f%% likely)",
labels.get(bestLabelIdx1),
labelProbabilities[bestLabelIdx1] * 100f));
   }
  }
private static Tensor<Float> construct And Execute Graph To Normalize Image1(byte[] imgb) {
try (Graph gr = new Graph()) {
GraphBuilderbr = new GraphBuilder(gr);

Final int h = 224;
finalint v = 224;
final float m = 117f;
final float s = 1f;

final Output<String> input = br.constant("input", imgb);
final Output<Float> output =
br.div(
br.sub(
br.resizeBilinear(
br.expandDims(
br.cast(b.decodeJpeg(input, 3), Float.class),
br.constant("make_batch", 0)),
br.constant("size", new int[] {h, v})),
br.constant("mean", m)),
br.constant("scale", s));
try (Session se = new Session(gr)) {
return
se.runner().fetch(output.op().name()).run().get(0).expect(Float.class);
     }
   }
  }
private static float[] executeInceptionGraph(byte[] graphDef, Tensor<Float> image) {
try (Graph gr = new Graph()) {
```

```
gr.importGraphDef(graphDef);
try (Session se = new Session(gr);
      Tensor<Float> result =
se.runner().feed("input", im-
age).fetch("output").run().get(0).expect(Float.class)) {
final long[] rs = result.shape();
if (result.numDimensions() != 2 || rs[0] != 1) {
throw new RuntimeException(
String.format(
          "Expected model to produce a [1 N] shaped
tensor where N is the number of labels, instead it pro-
duced one with shape %s",
Arrays.toString(rshape)));
      }
intnl = (int) rs1];
returnresult.copyTo(new float[1][nl])[0];
    }
   }
  }
private static intmaxIndex(float[] probabilities) {
int be = 0;
for (int i = 1; i <probabilities.length; ++i) {
if (probabilities[i] > probabilities[be]) {
be = i;
    }
   }
return be;
  }
private static byte[] readAllBytesOrExit(Path path) {
try {
returnFiles.readAllBytes(path);
   } catch (IOException e) {
System.err.println("Failed to read [" + path + "]: " +
e.getMessage());
System.exit(1);
   }
return null;
  }
private static List<String>readAllLinesOrExit(Path
path) {
try {
returnFiles.readAllLines(path, Charset.forName("UTF-
8"));
   } catch (IOException e) {
System.err.println("Failed to read [" + path + "]: " +
e.getMessage());
System.exit(0);
   }
return null;
  }

static class GraphBuilder {
GraphBuilder(Graph gr) {
    this.gr = gr;
   }

   Output<Float>div(Output<Float> x, Output<Float>
y) {
returnbinaryOp("Div", x, y);
   }
<T> Output<T>sub(Output<T> x, Output<T> y) {
returnbinaryOp("Sub", x, y);
   }
<T> Output<Float>resizeBilinear(Output<T> images,
Output<Integer> size) {
return binaryOp3("ResizeBilinear", images, size);
   }
<T> Output<T>expandDims(Output<T> input, Out-
put<Integer> dim) {
return binaryOp3("ExpandDims", input, dim);
   }
<T, U> Output<U>cast(Output<T> value, Class<U>
type) {
DataType dt = DataType.fromClass(type);
returng.opBuilder("Cast", "Cast")
       .addInput(value)
       .setAttr("DstT", dt)
       .build()
.<U>output(0);
   }
   Output<UInt8>decodeJpeg(Output<String> contents,
long channels) {
returng.opBuilder("DecodeJpeg", "DecodeJpeg")
      .addInput(contents)
      .setAttr("channels", channels)
       .build()
.<UInt8>output(0);
   }
<T> Output<T>constant(String name, Object value,
Class<T> type) {
try (Tensor<T> t1 = Tensor.<T>create(value, type)) {
returng.opBuilder("Const", name)
         .setAttr("dtype", DataType.fromClass(type))
         .setAttr("value", t1)
         .build()
.<T>output(0);
    }
   }
   Output<String>constant(String name, byte[] value) {
returnthis.constant(name, value, String.class);
   }

   Output<Integer>constant(String name, int value) {
returnthis.constant(name, value, Integer.class);
   }

   Output<Integer>constant(String name, int[] value) {
returnthis.constant(name, value, Integer.class);
   }
   Output<Float>constant(String name, float value) {
returnthis.constant(name, value, Float.class);
   }
private<T> Output<T>binaryOp(String type, Out-
put<T> in1, Output<T> in2) {
returng.opBuilder(type,
type).addInput(in1).addInput(in2).build().<T>output(0);
   }
private<T, U, V> Output<T> binaryOp3(String type,
Output<U> in1, Output<V> in2) {
returng.opBuilder(type,
type).addInput(in1).addInput(in2).build().<T>output(0);
   }
private Graph gr;
  }
}
```

### 2.5.2 Compile and execute the file

To compile a java program that uses Tensor Flow, the jar file must be downloaded and it should be the part of the program class path. The following command is used to compile the file [8][9][10]

Javac –cp libtensorflow-1.7.0.jar imgr.java

**Running:**

To execute a java program that depends on tensor flow, first we need to ensure the following two files are available to the JVM[8][9][10]

1) downloaded .jar file
2) Extracted JNI library

The Following command line is used to
Execute the imgr program

Java –cp libtensorflow-1.7.0.jar;. –Djava.library.path= jniimgr

## 3. Results and Screen Shots

After training and testing the above sample dataset using tensor flow in java by the keras library the model gives the following results as a best match to the images (Figure 1 and Figure 2).

**Fig** 1 Image1-Dog

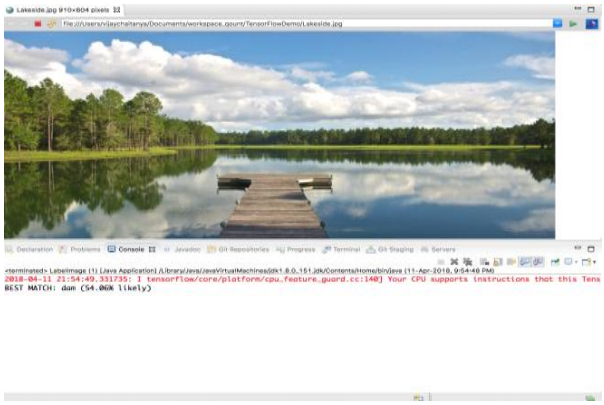

**Fig 2** Image2-Scenary

## 4. Conclusion

In this paper we present the introduction to keras library with backend Tensor Flow. In this we implement a model using tensor flow in java with the sample dataset of different classes to recognize objects in labeled images. The improvement we found using this is the accuracy in finding the objects in images is more than the traditional neural networks because the keras library using convolution neural networks.

## References


[1] https://elitedatascience.com/learn-machine-learning#what
[2] https://www.datanami.com/2017/01/30/deep-learning-now/
[3] https://elitedatascience.com/python-deep-learning-libraries#keras
[4] https://keras.io/
[5] https://www.tensorflow.org/install/install_java
[6] https://machinelearningmastery.com/introduction-python-deep-learning-library-keras/
[7] https://elitedatascience.com/keras-tutorial-deep-learning-in-python.
[8] Tensor Flow, Available online: https://www.tensorflow.org.
[9] B. Frederic, P. Lamblin, R. Pascanu et al., "Theano: new features and speed improvements," in Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012, http://deeplearning.net/software/theano/.
[10] Athanasius Voulodimos, Nikolaos Doulamis, Anastasias Doulamis, and Eftychios Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," Computational Intelligence and Neuroscience, vol. 2018, Article ID 7068349, 13 pages, 2018. doi:10.1155/2018/7068349.