

Ubiquitous and Mobile Computing

CS 528: Keras support for Android Deep learning

*Presented By,
Bhoomi Patel, Sanika Patki, Srinarayan Srikanthan*

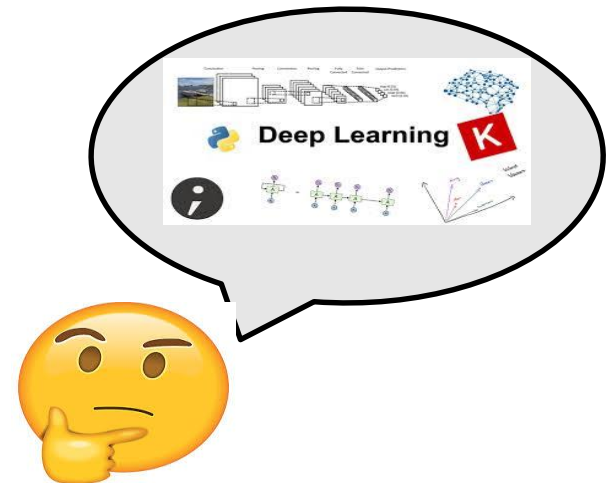
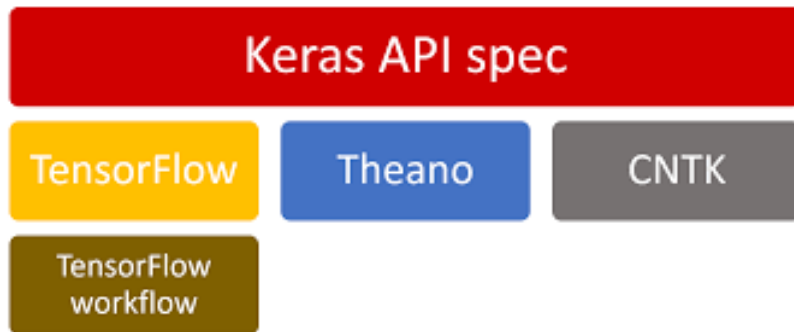
*Computer Science Dept.
Worcester Polytechnic Institute (WPI)*





What is Keras?

- **Keras** is high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
- Keras enables fast experimentation with deep neural networks.
- It is an interface rather than a standalone machine learning framework.





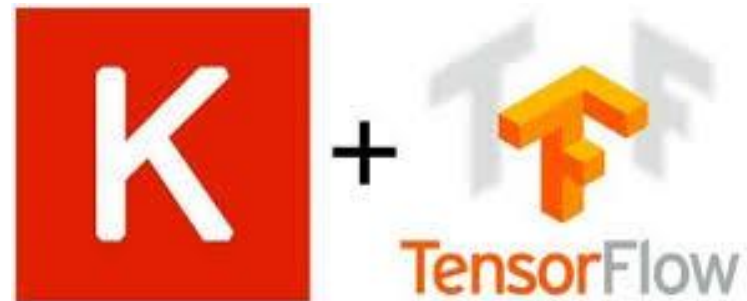
What is Keras?

- Benefits: User Friendliness, Modularity, Easy extensibility
- Keras allows users to productize deep models on smartphones (Android) using its android support for deep learning

If you're asking "Keras or TensorFlow?"



Then you're asking the *wrong* question (and here's why...)



Deep Learning with Keras



Keras Overview...

Keras Overview

8/30/2017

What is Keras?

- Neural Network library written in python
- Design to be simple and straightforward
- Built on top of different deep learning libraries such as Tensorflow, Theano and CNTK

Why Keras?

- Simple
- Highly modular
- Deep enough to build models

Background



- Keras was initially developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System).
- In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library.
- Keras models can be easily deployed across a greater range of platforms like Android, iOS, Google Cloud etc. On Android, Keras can be deployed using the TensorFlow Android runtime.
- Today, Keras has broad adoption in the industry and the research community.



Motivation- Why Keras?

- **User friendliness**- Offers consistent & simple APIs
- **Modularity**- neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes are all standalone modules that can be plugged together to create new models.
- **Easy extensibility**- New modules are simple to add.
- **Work with Python**. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility. Keras is compatible with: **Python 2.7-3.6**.

Why Keras?

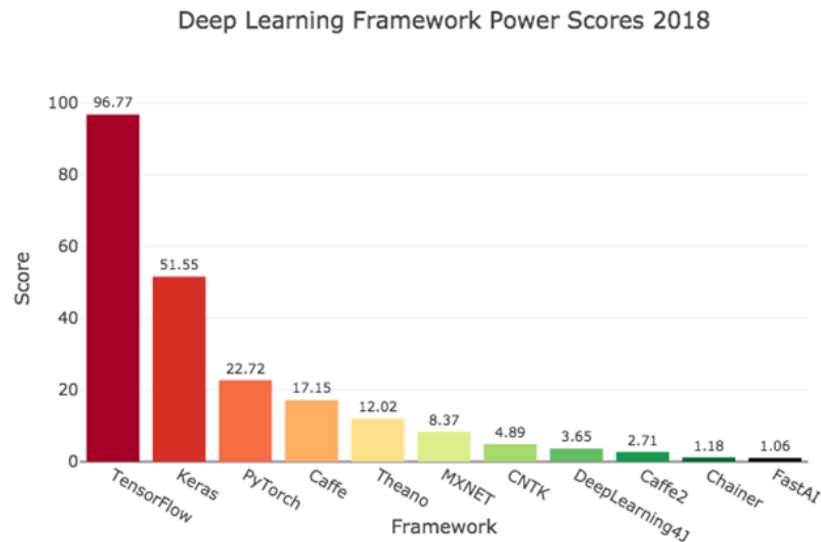


- Keras models can be easily deployed across platforms like Android, iOS, Google Cloud, JVM or Raspberry Pi than any other deep learning framework.
- Keras models can be developed with a range of different deep learning backends like:
 - The TensorFlow backend (from Google)
 - The CNTK backend (from Microsoft)
 - The Theano backend
- Keras model can be trained on a number of different hardware platforms beyond CPUs. Eg. NVIDIA GPUs , Google TPUs etc.
- Keras has strong multi-GPU support and distributed training support
- Keras development is backed by key companies in the deep learning ecosystem

Why Keras?



- Keras has broad adoption in the industry and the research community.

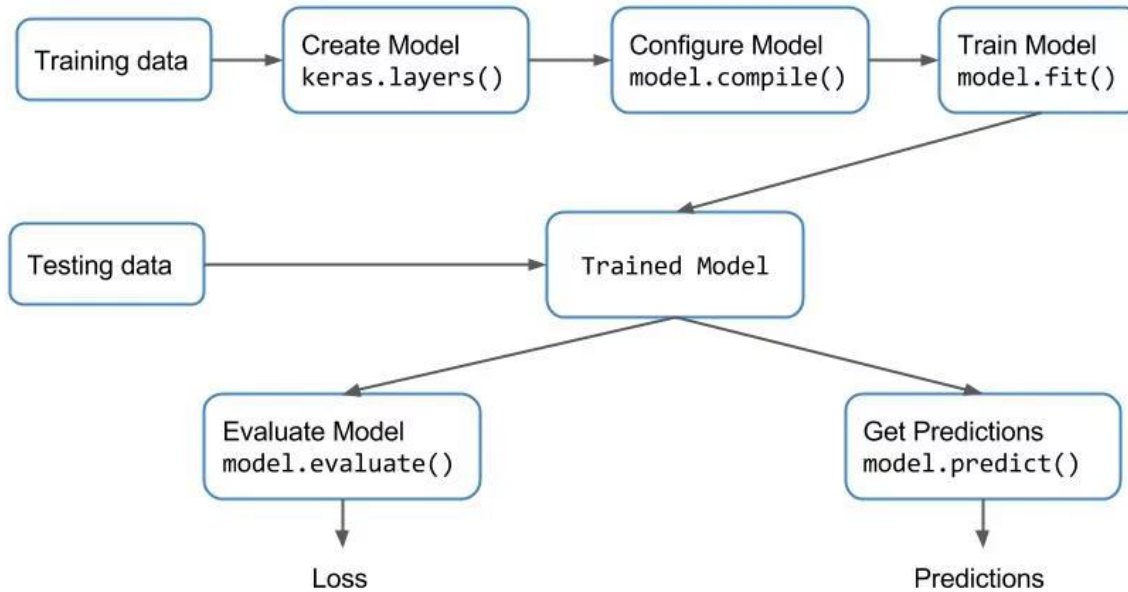


When to use keras?



- We use Keras if we need a deep learning library that:
 - Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
 - Runs seamlessly on CPU and GPU.
 - Supports both convolutional networks and recurrent networks, as well as combinations of the two.

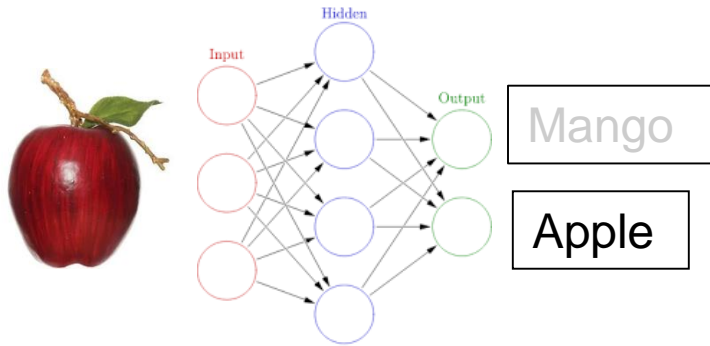
Keras Workflow





Specific Problems it solves

Image Recognition



Speech Recognition



Google Translate



Medical Image Analysis

Deep Learning and Medical Image Analysis for Malaria Detection



Medical Image Analysis with Deep Learning

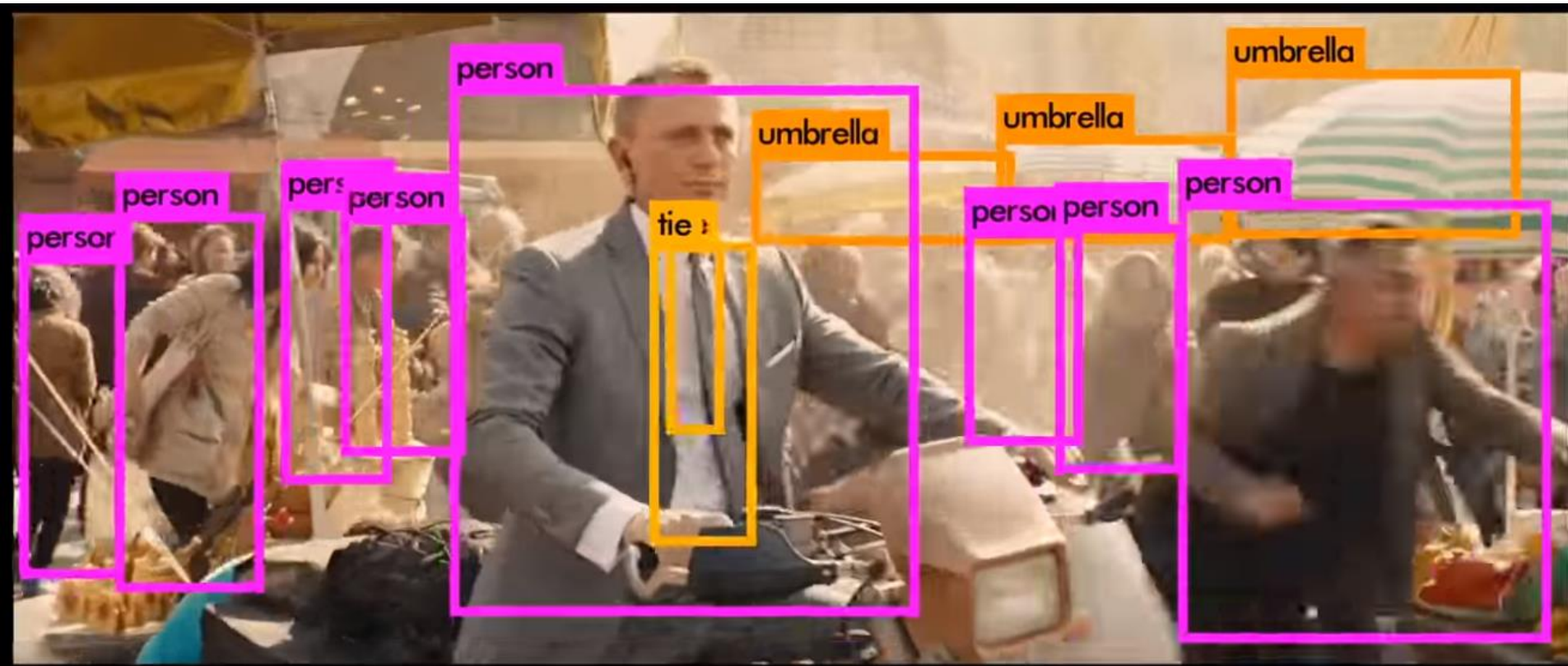


Real World Implementation example



- Using YOLO weights on keras

<https://www.youtube.com/watch?v=VOC3huqHrss>

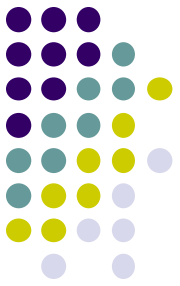


Real world Use

- Companies using keras:




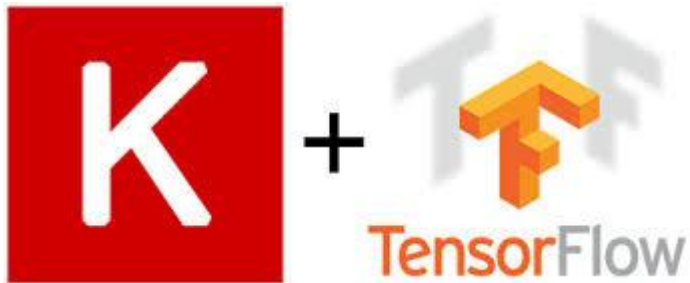
- Keras has also been adopted by researchers at large scientific organizations, in particular CERN and NASA.





Machine Learning in Android

- Keras + Android  Powerful Android Apps
- Object Detection
- Human Activity Recognition
- Recommendation System





Keras Workflow In Android

End to End: tf.Keras to TFLite to Android



- Train your model
(tf.keras)
3 choices for model building:
- Sequential
 - Functional
 - Model subclassing

- Convert to tflite
(tf.lite.TFLiteconverter)
2 choices for conversion:
- Python code (recommended)
 - Command line

- Run on Android:
1. Model file under /assets
 2. Update build.gradle
 3. Input image
 4. Preprocessing
 5. Classify with the model
 6. Post processing
 7. Display result in UI



How Keras Works on Android?

Step 1 : Keras Setup

- Install Python and following libraries - Keras, tensorflow
- Since we're going to convert our file to .tflite, we need to install toco using the following command:

```
(base) C:\Users\Bhoomi Kalpesh Patel>pip install keras
```

```
(base) C:\Users\Bhoomi Kalpesh Patel>pip install tensorflow
```

```
(base) C:\Users\Bhoomi Kalpesh Patel>pip install toco
```




- **Step 2:- Generate Data** – Generate random numbers from x1 to x6. Using a certain equation generate value of y. Append the values to list and store the data in csv file.

```
import random
import csv
def getRandom():
    return int(random.random()*1000)
def generateData():
    output = []
    output.append(['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'y'])
    for i in range(0, 50000):
        x1=getRandom()
        x2=getRandom()
        x3=getRandom()
        x4=getRandom()
        x5=getRandom()
        x6=getRandom()
        #####
        #This is the equation imprinted to generated data
        y=x1+x2+x3+x4+x5-x6
        #####
        output.append([x1,x2,x3,x4,x5,x6,y])

    with open('generated_dataset.csv', 'w', newline='') as fp:
        a = csv.writer(fp, delimiter=',')
        a.writerows(output)
```

```
generateData()
```

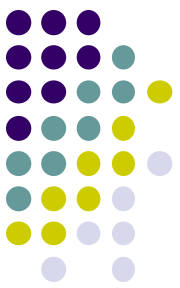


- **Step 3: Train Model using Keras and Predict Values**
- Read & split data, Define model as sequential, Add layers using **model.add()** function. Configure model using **model.compile()**, Train model using **model.fit()**, Perform Prediction, save model as filename.h5 file
- **Model.add()** – Add layers
 - The first layer in a Sequential model needs to receive information about its input shape.
 - Some 2D layers, such as Dense, support the specification of their input shape via the argument `input_dim`



- **Configure the learning process:**
- **Use compile() method.** It receives three arguments:
 - ✓ An optimizer. This could be the string identifier of an existing optimizer (such as rmsprop or adagrad), or an instance of the Optimizer class
 - ✓ A loss function. This is the objective that the model will try to minimize
 - ✓ A list of metrics. For any classification problem you will want to set this to `metrics=['accuracy']`.

```
#Model definition
model = Sequential()
model.add(Dense(16, input_dim=6, activation='relu'))
model.add(Dense(1, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
```



```
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
df = pd.read_csv("D:\WPI\Fall'19\Android\TechTalk\generated_dataset.csv")

X = df.drop(labels=['y'], axis=1).values
Y = df[['y']].values

X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2, random_state=42)

model = Sequential()
model.add(Dense(16, input_dim=6, activation='relu'))
model.add(Dense(1, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])

model.fit(X_train, Y_train, epochs=40, shuffle=True, verbose=2)

X_prediction= pd.DataFrame([{'x1':1, 'x2':2, 'x3':3, 'x4':4, 'x5':5, 'x6':6}])
print( model.predict(X_prediction) )

model.save("trained_model.h5", include_optimizer=False)
```



- **Step 4: Convert keras to Tensorflow lite**

```
toco \  
  --output_file=trained_model.tflite\  
  --keras_model_file=trained_model.h5
```

```
(base) D:\WPI\Fall'19\Android\TechTalk>tflite_convert \ --output_file=/tmp/androidKeras.tflite \ --keras_model_file=trained_model.h5
```



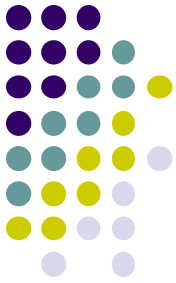
- **Step 5: Import Model To Android Studio**

i. Add the following dependency to the app/build.gradle:

```
implementation 'org.tensorflow:tensorflow-lite:0.0.0-nightly'
```

ii. Create a folder assets where we place kerasAndroid.tflite. This folder should be at the same level of src, res

iii. Using Kotlin, import the model, and run the android project to obtain the output.

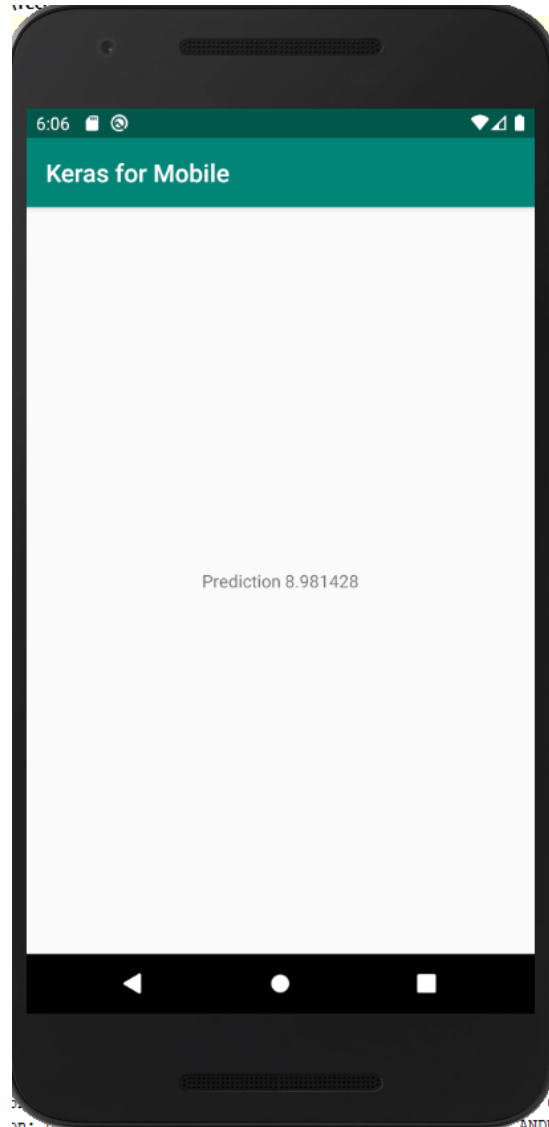


- **Step 6: Create xml file. Create layout. Create Android App.**

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        thread(start = true) {  
            runInference()  
        }  
    }  
  
    fun runInference() {  
        val tfliteModel = loadModelFile( activity: this)  
        val tflite = Interpreter(tfliteModel, Interpreter.Options())  
  
        var inputData: ByteBuffer = ByteBuffer.allocateDirect(  
            capacity: 1 // 1 dimension  
            * 6 //6 attributes/columns  
            * 1 //1 row  
            * 4 //4 bytes per number as the number is float  
        )  
        inputData.order(ByteOrder.nativeOrder())  
  
        floatArrayOf(1.0f, 2.0f, 3.0f, 4.0f, 5.0f, 6.0f).forEach { it:Float  
            | inputData.putFloat(it)  
        }  
  
        val labelProbArray: Array<FloatArray> = Array( size: 1 ) { FloatArray( size: 1 ) }  
  
        tflite.run(inputData, labelProbArray)  
  
        var prediction = (labelProbArray[0][0])  
        runOnUiThread { result.text = "Prediction $prediction" }  
    }  
  
    private fun loadModelFile(activity: Activity): MappedByteBuffer {  
        val fileDescriptor = activity.getAssets().openFd( fileName: "androidKeras.tflite")  
        val inputStream = FileInputStream(fileDescriptor.fileDescriptor)  
        val fileChannel = inputStream.channel  
        val startOffset = fileDescriptor.startOffset  
        val declaredLength = fileDescriptor.declaredLength  
        return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength)  
    }  
}
```



- Output:





References

- <https://en.wikipedia.org/wiki/Keras>
- <https://keras.io/>
- *medium.com*
- <https://towardsdatascience.com/what-can-deep-learning-bring-to-your-app-fb1a6be63801>
- <https://blog.venturepact.com/10-examples-of-machine-learning-mobile-apps/>