After TensorFlow 2.0 Release: Is Keras.NET still relevant?

After being released in 2015, Keras has been the most popular package for developers to use when working within the field of Deep Learning. Throughout the years TensorFlow and Keras has been two separate packages that could interact with each other as TensorFlow served as backend for Keras. However, late in September 2019, a new version of TensorFlow has been released which has been a game changer for Keras users. Read this tutorial article to find out more about Keras, the history between Keras and TensorFlow and in conclusion what users should do in the future.

What is Keras.NET?

Keras**.**NET is a high-level neural networks API, written in C# with Python Binding and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation, where the user are able to go from idea to result with the least possible delay, which is key to doing good research. Keras is used when working with the artificial intelligence known as Deep Learning. The reason for this is that Keras allows for easy and fast prototyping (through user friendliness, modularity, and extensibility). Moreover, Keras supports both convolutional networks and recurrent networks, as well as combinations of the two. Lastly, it runs flawlessly on both CPU and GPU, which makes it very attractive for developers to use.

The history of Keras.NET

Keras was originally created and developed by Google AI Developer/Researcher, Francois Chollet, who committed and released the first version of Keras to his GitHub on March 27th, 2015. The idea behind Keras was that it should be easy to use for developers, which was one of the reasons why it became populated among its users. At that time, there were not too many deep learning libraries available — the popular ones included Torch, Theano, and Caffe, yet those were too time-consuming and inefficient according to many developers.

Keras backends

As mentioned earlier, Keras is providing building blocks for developing deep learning models. To do this Keras rely on a specialized, well optimized tensor manipulation library, which is serving as the backend engine of the function of Keras. Rather than

choosing one single library and making the implementation and code dependent on that, Keras allow for multiple backends, which can be "plugged" seamlessly into Keras. At this time, Keras has three backend libraries available, which are the TensorFlow backend, the Theano backend and the CNTK backend.

A Keras model and the code behind

Below will give you an example of one of the more simple models that developers can use when working with Keras. The easiest way of creating a model in Keras is by using the sequential API, which lets you stack one layer after the other. However, there is a problem with the sequential API, because it does not allow models to have multiple inputs or outputs, which are needed for some problems. Yet, for most problems the model will work perfectly – as in this example.

Firstly, it is important to import the necessary libraries to get the function to work. In this case the libraries are important from respectively keras.model and keras.layers. To create a convolutional neural network, one only need to create a Sequential object and use the *add* function to add layers.

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout

model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=x_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(10, activation='softmax'))
```

Figure 1: Sequential coding

The code above first of creates a Sequential object and adds a few convolutional, maxpooling and dropout layers. It then flattens the output and passes it to a last dense and dropout layer before passing it to the output layer. This model is an example of a simple model in Keras that allows for prediction.

Multiple data inputs in Keras

Keras can also be used for more advanced coding as Keras accept multiple inputs from mixed data where it is possible to train a multi-input model that accepts multiple types of input data in a single end-to-end network. In this case, mixed data means that the data can both be numerical, categorical etc. meaning having multiple types of independent data as shown in the figure below.

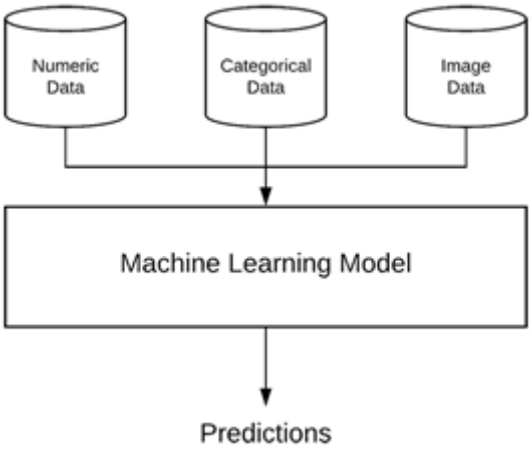

Figure 2: Multiple data inputs

Moreover, when working with machine learning systems that are capable of handling mixed data, it can be extremely challenging as each data type may require separate preprocessing steps, including scaling, normalization, and feature engineering, which is important that the developer keep in mind. This type of function can be relevant in cases with e.g. hospital data, where the information of different patients have different values.

In order to work with multiple inputs, a developer most use Functional API, as it makes it easier to manipulate large number of intertwined datasets.

keras vs. tf.keras

On September 30th 2019 a new package for deep learning users was released – TensorFlow 2.0. The release of this new package within Keras has both advantage and disadvantage. The advantage is that developer can access a new and improve package, which can be used to train their neural network data. However, the release of this new package has created confusion for many data scientist, since it is not clear what the release means for Keras users or which package (keras or tf.keras) should be used to

train the neural network. Furthermore, many have been wondering to what extend keras is still relevant to use.

The reason for this confusion is that Keras and TensorFlow has always had an intertwined history with lots of overlaps in features and abilities as shown in the figure below.

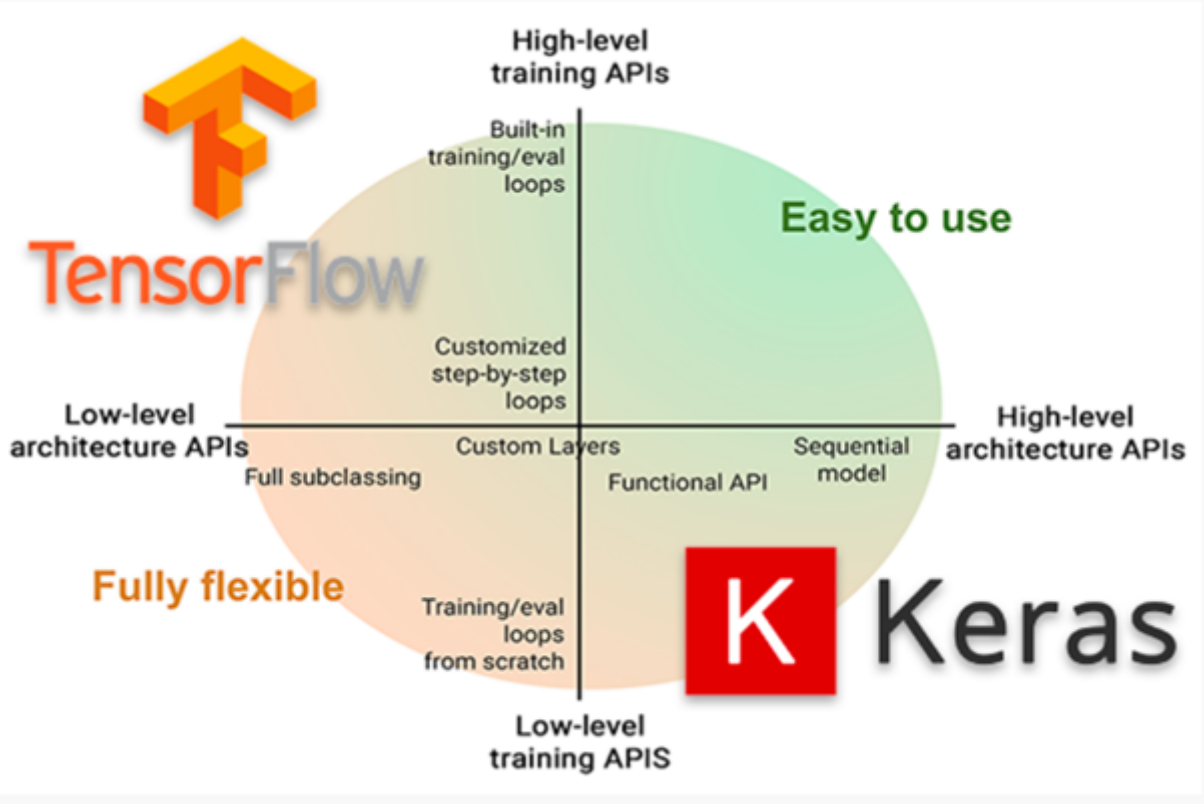

Figure 3: TensorFlow and Keras relations

There is a good reason for this confusion, since Keras and TensorFlow 2.0 have had several overlaps over the years. This will be elaborated in the next paragraph.

The intertwined history

When using Keras, there is one important criteria that must be fulfilled in order to use the function. In order to train the custom neural networks, Keras required a backend. A backend is a computational engine, which builds the network graph/topology, runs the optimizers, and performs the actual number crunching. In other words, the backend is the database and Keras the programming language used to access the database. Originally, Keras used Theano as default backend. Yet, at the same time, Google had released the first version of TensorFlow, a symbolic math library used for

machine learning and training neural networks. After this new release of TensorFlow, Keras started supporting TensorFlow as a backend, which resulted in the increased popularity of TensorFlow. As TensorFlow became the default backend of Keras, the two libraries grew together, which made it impossible to have the one without the other. This meant that if you installed one of them, you had to install the other, since they were still two separate libraries.
However, this all changed when Google released TensorFlow 2.0, where Keras was integrated directly in the TensorFlow package itself.
With TensorFlow 2.0, Google declared that "*Keras is now the official high-level API of TensorFlow for quick and easy model design and training*".

Which package should be used in the future?

Many have been wondered what this meant and which package they should be using in the future. With the release of the newest version of Keras, the creator and maintainer of Keras Francois Chollet stated that:

"This is also the last major release of multi-backend Keras. Going forward, we recommend that users consider switching their Keras code to tf.keras in TensorFlow 2.0.

It implements the same Keras 2.3.0 API (so switching should be as easy as changing the Keras import statements), but it has many advantages for TensorFlow users, such as support for eager execution, distribution, TPU training, and generally far better integration between low-level TensorFlow and high-level concepts like Layer and Model.

It is also better maintained."

This means that with new release of TensorFlow 2.0 and with keras and tf.keras being in sync, developers should start using tf.keras, since the keras package will only support bug fixes until April 2020. Therefore, tf.keras will be better maintained and has better integration with TensorFlow features such as eager execution, TBU training, distribution support and other that makes it more desirable to work with.