Practical 2:
Introduction to KERAS

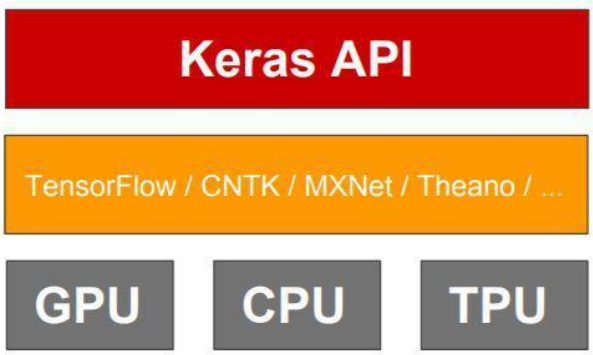
2CSDE61
Deep Learning

18BCE239

Sunidhi Tandel

What is Keras?

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

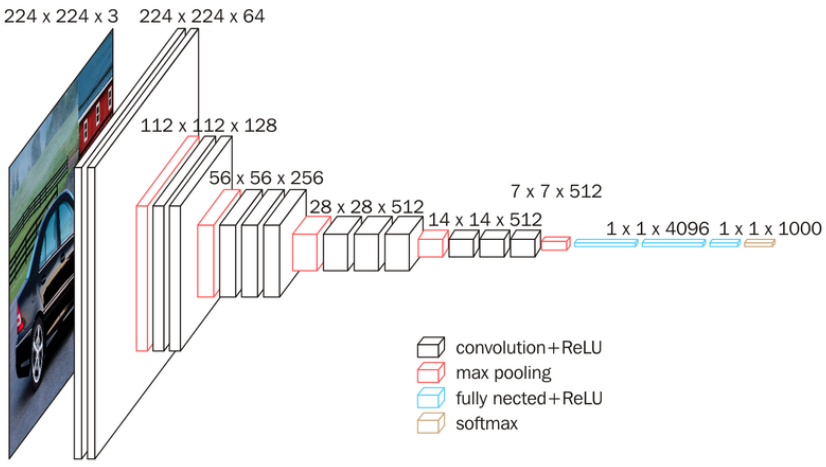


Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

Models API

There are three ways to create Keras models:

- The **Sequential model**,which is very straightforward (a simple list of layers), but is limited to single-input, single-output stacks of layers (as the name gives away).



- The **Functional API**,which is an easy-to-use, fully-featured API that supports arbitrary model architectures. For most people and most use cases, this is what you should be using. This is the Keras "industry-strength" model.

Model subclassing,where you implement everything from scratch on your own. Use this if you have complex, out-of-the-box research use cases.

Keras Fundamentals for Deep Learning

The main structure in Keras is the Model that defines the complete graph of a network. You can add more layers to an existing model to build a custom model that you need for your project.

Here's how to make a Sequential Model and a few commonly used layers in deep learning

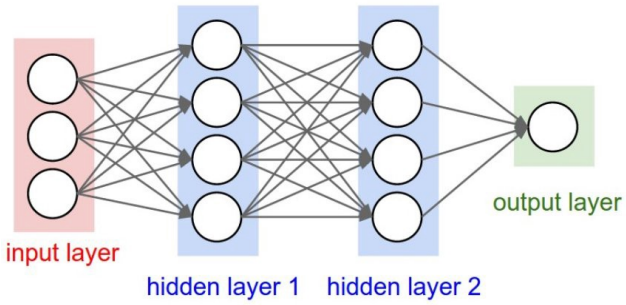
1. Sequential Model

```
from keras.models import Sequential

from keras.layers import Dense,

Activation,Conv2D,MaxPooling2D,Flatten,Dropout

model = Sequential()
```



2. Convolutional Layer

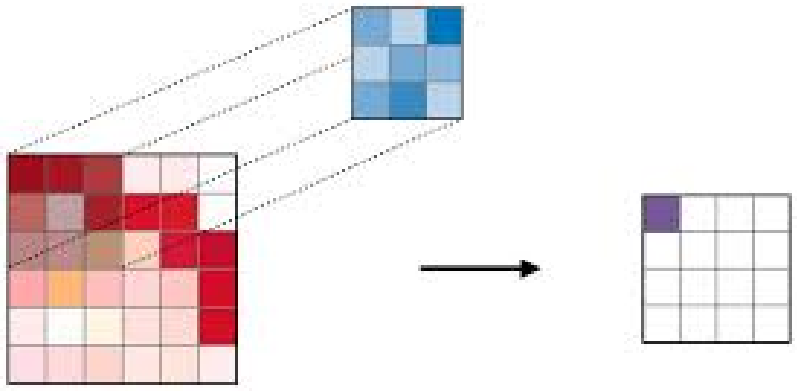
This is an example of a convolutional layer as the input layer with the input shape of 320x320x3, with 48 filters of size 3x3 and uses ReLU as an activation function.

```
input_shape=(320,320,3) #this is the input shape of an
image 320x320x3

model.add(Conv2D(48, (3, 3), activation='relu',
input_shape= input_shape))
```

another type is

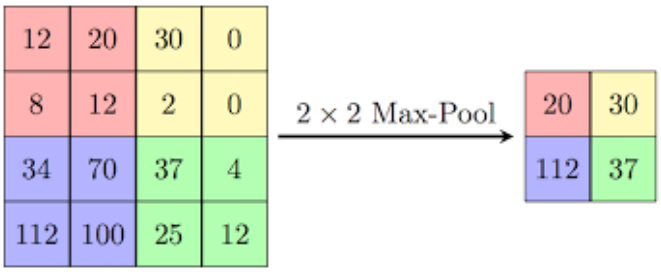
```
model.add(Conv2D(48, (3, 3), activation='relu'))
```



3. MaxPooling Layer

To downsample the input representation, use MaxPool2d and specify the kernel size

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```



4. Dense Layer

adding a Fully Connected Layer with just specifying the output Size

```
model.add(Dense(256, activation='relu'))
```

5. Dropout Layer

Adding dropout layer with 50% probability

```
model.add(Dropout(0.5))
```

Compiling, Training, and Evaluate

After we define our model, let's start to train them. It is required to compile the network first with the loss function and optimizer function. This will allow the network to change weights and minimize the loss.

```
model.compile(loss='mean_squared_error', optimizer='adam')
```

Now to start training, use fit to fed the training and validation data to the model. This will allow you to train the network in batches and set the epochs.

```
model.fit(X_train, X_train, batch_size=32, epochs=10,
validation_data=(x_val, y_val))
```

Our final step is to evaluate the model with the test data.

```
score = model.evaluate(x_test, y_test, batch_size=32)
```

ADVANTAGES

1. **Fast Deployment and Easy to understand**

Because of the friendly API, we can easily understand the process. Writing the code with a simple function and no need to set multiple parameters.

2. **Large Community Support**

There are lots of AI communities that use Keras for their Deep Learning framework. Many of them publish their codes as well as tutorials to the general public.

3. **Have multiple Backends**

You can choose **Tensorflow, CNTK, and Theano**as your backend with Keras. You can choose a different backend for different projects depending on your needs. Each backend has its own unique advantage.

4. **Cross-Platform and Easy Model Deployment**

With a variety of supported devices and platforms, you can deploy Keras on any device like

- iOS with CoreML
- Android with Tensorflow Android,
- Web browser with .js support
- Cloud engine
- Raspberry Pi

5. **Multi GPUs Support**

You can train Keras on a single GPU or use multiple GPUs at once. Because Keras has built-in support for data parallelism so it can process large volumes of data and speed up the time needed to train it.

DISADVANTAGES

- **Cannot handle low-level API**

Keras only handles high-level API which runs on top of other framework or backend engines such as Tensorflow, Theano, or CNTK. So it's not very useful if you want to make your own abstract layer for your research purposes because Keras already have pre-configured layers.

Pre-Trained Models in Keras and How to Use Them?

(TRANSFER LEARNING)

For the pre-trained model, we can load a variety of models that Keras already has in its library such as:

VGG16, InceptionV3, ResNet, MobileNet, Xception, InceptionResNetV2

```
from keras.applications.vgg16 import VGG16
base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(image_size, image_size, 3))

# Freeze the layers
 for layer in base_model.layers:

    layer.trainable = False
# # Create the model
model = keras.models.Sequential()
# Add the vgg convolutional base model
model.add(base_model)
# # Add new layers
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(num_class, activation='softmax'))
```