Date: 18/01/2021

Practical 2

2CSDE61 - Deep Learning

Name: Shrey Viradiya

Roll No: 18BCE259

Exploring

TensorFlow Keras API

What is TensorFlow Keras API?

TensorFlow is an open-source platform for deep learning and machine learning, developed by Google. TensorFlow is used for research, development, and production. TensorFlow can efficiently execute low-level tensor operations on CPU, GPU, or TPU. TensorFlow 1.0 was released in 2017 and the next major release was in 2019 i.e., TensorFlow 2.0.

Keras is a high-level API built upon TensorFlow 2.0 for a faster deep learning solution and a highly-productive interface.

Advantages of TensorFlow Keras API

1. User-Friendly, Faster Development, and Deployment
2. Pretrained Models
3. Multiple GPUs Support
4. Well written documentation

Disadvantages of TensorFlow Keras API

1. Low-level API Error
2. Some implementation not available

Exploring TensorFlow Keras API

With TensorFlow Keras API, creating a model and training becomes a piece of cake. We can create a model using the Sequential model or the functional API. A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor. With functional API, we can create a more complex model with more flexibility. After compiling the model, `fit()` method with train the model with given data.

The Sequential Model

```
model = keras.Sequential([
        layers.Dense(2, activation="relu", name="layer1"),
        layers.Dense(3, activation="relu", name="layer2"),
        layers.Dense(4, name="layer3"),
    ]
)
```

OR

```
model = keras.Sequential()
model.add(layers.Dense(2, activation="relu"))
model.add(layers.Dense(3, activation="relu"))
model.add(layers.Dense(4))
```

The Functional API

```
inputs = keras.Input(shape=(32, 32, 3))
dense = layers.Dense(64, activation="relu")
x = dense(inputs)
x = layers.Dense(64, activation="relu")(x)
outputs = layers.Dense(10)(x)
model = keras.Model(inputs=inputs, outputs=outputs,
name="mnist_model")
```

Compiling, Training, Evaluating, and Predicting

With TensorFlow Keras API, one line code is enough to train the model, evaluate the trained model, and predict using trained. First, `compile()` method is used to compile the model with optimizer, loss, and metrics.

```
model.compile(
    optimizer=keras.optimizers.RMSprop(),
    loss=keras.losses.SparseCategoricalCrossentropy(),
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)
```

After this, we train the model with the `train()` method. The returned "history" object holds a record of the loss values and metric values during training.

```
history = model.fit(x_train, y_train, batch_size=64, epochs=2,
    validation_data=(x_val, y_val),
)
```

Using `evaluate()` method, the model is evaluated.

```
model.evaluate(test_dataset)
```

We get a prediction from a model by passing the data in the `predict()` method.

```
model.predict(x_test[:3])
```

Save and load Keras models

All the details of the Keras model i.e., the architecture, the weights, the optimizer and etc. are can be saved with `save()` method. Saved method can be loaded with `load_model()`.

```
model.save('path/to/location')
model = keras.models.load_model('path/to/location')
```

Datasets

TensorFlow Keras API provides some vectorized datasets for creating simple code and debugging. These datasets include MNIST, CIFAR10, CIFAR100, IMDB reviews, Fashion MNIST and, Boston Housing data, etc.

References:

1] TensorFlow Core

2] Python Keras Advantages and Limitations - DataFlair (data-flair.training)

3] Advantages and Drawbacks of Keras - TechVidvan