Overview
This document demonstrates a simple program that can recognize handwritten digits in MNIST dataset and run it optimally on widely available hardware using the machine learning platform TensorFlow and OpenVINO™ Toolkit.

Getting Started
First, you need to set up the Python environment for TensorFlow:

```bash
#setup the environment
!pip install -q openvino-dev[onnx,tensorflow2]==2022.1.0 &> error_log_openvino.txt
#pip install -q tensorflow==2.5.1
!pip install -q pandas==1.2.4 &> error_log_panda.txt
```

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only “../input/” directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory
import os
for dirname, _, filenames in os.walk('/kaggle/input'):    
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using “Save & Run All”
# You can also write temporary files to /kaggle/temp/, but they won’t be saved outside of the current session
```
Getting Started

```python
import tensorflow as tf

mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train, x_test = x_train / 255.0, x_test / 255.0
```

Loading the Database

For this demonstration, you will use the MNIST database to create a model that can recognize handwritten digits and train neural networks.

```python
model = tf.keras.models.Sequential(
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10)
)

predictions = model(x_train[:1]).numpy()
predictions

tf.nn.softmax(predictions).numpy()

loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)

loss_fn(y_train[:1], predictions).numpy()

model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=10)
```
model.evaluate(x_test, y_test, verbose=2)

probability_model = tf.keras.Sequential([model,
                                       tf.keras.layers.Softmax()])

probability_model(x_test[:5])

model_fname="mnist"
probability_model.save(model_fname)

---

Loading the Runtime

Since OpenVINO™ Toolkit works on models in its own “Intermediate Representation” (IR) format, it is necessary to declare where the model created with TensorFlow is, and the name and data type (FP16, that is floating point, 16-bit numbers) of the IR version. The Model Optimizer command will generate the IR model:

```python
from openvino.runtime import Core
from pathlib import Path
import json
import sys
import numpy as np
import matplotlib.pyplot as plt

# The paths of the source and converted models
model_name = "mnist"
model_path = Path(model_name)
ir_data_type = "FP16"
ir_model_name = "mnist_ir"

# Get the path to the Model Optimizer script

# Construct the command for Model Optimizer
mo_command = f"mo
    --saved_model_dir "{model_name}"
    --input_shape ":[28,28]"
    --data_type "{ir_data_type}\"
"
```
# Run the Model Optimizer (overwrites the older model)
mo_command = " '.join(mo_command.split())"

# Run the Model Optimizer (overwrites the older model)
print("Exporting TensorFlow model to IR... This may take a few minutes.")
mo_result = %sx $mo_command
print("\n'.join(mo_result))
```

---

**Loading the Network to the Plugin**

Now you will load the `(model_xml)` and weigh `(model_bin)` of the IR model into the OpenVINO™ Toolkit inference engine and start recognizing digit images from MNIST.

```python
model_xml = "mnist_ir.xml"
model_bin = "mnist_ir.bin"

# Load network to the plugin
ie = Core()
model = ie.read_model(model=model_xml)
compiled_model = ie.compile_model(model=model, device_name="CPU")

input_layer = compiled_model.input(0)
output_layer = compiled_model.output(0)

# Test against a few images from the dataset
input_list = x_test[:10]
for input_image in input_list:
    res = compiled_model([input_image])[output_layer]
    X = input_image
    X = X.reshape([28, 28]);
    plt.figure()
    plt.gray()
    plt.imshow(X)
    plt.text(0,-1, "The prediction is " + str(np.argmax(res[0])) + " @ " + str(max(res[0])*100) + "%")
```
The Results
Below you can see 10 digits properly identified with an accuracy of 99.99% or higher in most cases.

Now you’ve seen for yourself just how easy it is to write high-performing OCR code with OpenVINO™ Toolkit.

To learn more about OCR, read this blog and check out the Intel® AI Dev Team Adventures.

< Back to Home