TensorFlow Library

Introduction to TensorFlow

TensorFlow is an open-source library developed by Google to simplify the process of building and training machine learning (ML) and deep learning models. It is designed to work efficiently with large datasets and complex algorithms, making it a powerful tool for AI developers and researchers.

The library supports a wide range of tasks, from simple data processing to advanced neural network designs, and provides pre-built functions to speed up development. TensorFlow is flexible, allowing developers to work in multiple programming environments, including Python, and scale their models from small devices like smartphones to large servers.

With TensorFlow, you can create applications like image recognition, natural language processing, recommendation systems, and more. It also comes with extensive documentation, tutorials, and community support to help both beginners and experts in their AI journey.

Install TensorFlow 2

TensorFlow is tested and supported on the following 64-bit systems:

- Python 3.8–3.11
- Ubuntu 16.04 or later
- Windows 7 or later (with C++ redistributable)
- macOS 10.12.6 (Sierra) or later (no GPU support)
- WSL2 via Windows 10 19044 or higher including GPUs (Experimental)

Command line or Powershell commands :

```
# Requires the latest pip
pip install --upgrade pip
# Current stable release for CPU and GPU
pip install tensorflow
# Or try the preview build (unstable)
pip install tf-nightly
```

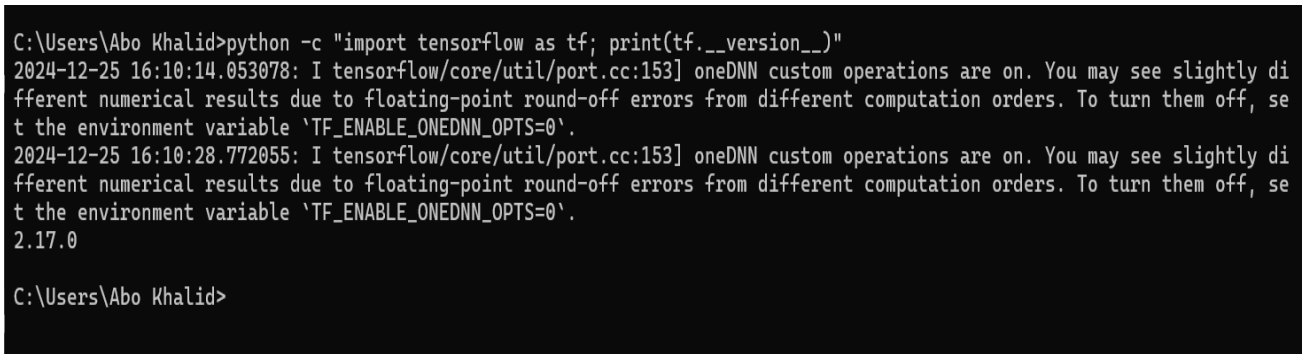
Run a TensorFlow container

The TensorFlow Docker images are already configured to run TensorFlow. A Docker container runs in a virtual environment and is the easiest way to set up GPU support.

```
docker pull tensorflow/tensorflow:latest  # Download latest stable image
docker run -it -p 8888:8888 tensorflow/tensorflow:latest-jupyter  # Start Jupyter server
```

To verify the installation: cmd ↙️

python -c "import tensorflow as tf; print(tf.__version__)"



Key Features Offered to Developers

1. **Flexibility**: Developers can easily design and customize models with flexible programming interfaces.
2. **Cross-platform Support**: TensorFlow runs on various platforms, from mobile devices to supercomputers.
3. **Visualization Tools**: Includes tools like TensorBoard for training visualization and performance analysis.
4. **High Performance**: Supports GPU and TPU acceleration to enhance training speed.
5. **Large Community**: TensorFlow is highly popular, offering numerous tutorials and ongoing community support.
    a. **Support sections**
        i. Issue tracker
        ii. Release notes
        iii. Stack Overflow
        iv. Brand guidelines
        v. Cite TensorFlow
    b. **Stay connected**
        i. Blog
        ii. Forum
        iii. GitHub
        iv. Twitter
        v. YouTube
6. Distributed training with keras | TensorFlow Core
7. Distribute your model training across multiple GPUs, multiple machines or TPUs.

A Simple Practical Example Using TensorFlow

The following example demonstrates how to build a simple linear regression model for prediction:

```
import tensorflow as tf
import numpy as np
# Training data
x_train = np.array([1.0, 2.0, 3.0, 4.0], dtype=float)
y_train = np.array([2.0, 4.0, 6.0, 8.0], dtype=float)
```

```
# Building the model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=1, input_shape=[1])
])
```

الطبقة **Dense** تعني أن جميع الوحدات العصبية في هذه الطبقة متصلة بالمدخلات.#

#input_shape=[1] يحدد أن النموذج يتوقع مدخلاً واحدًا (قيمة واحدة في كل مرة).

```
# Compiling the model (choosing loss function and optimizer)
model.compile(optimizer='sgd', loss='mean_squared_error')
#Stochastic Gradient Descent (SGD)
# Training the model
model.fit(x_train, y_train, epochs=500, verbose=0)
# Testing the model

print("Predicted value for 5.0:",
model.predict([5.0])[0][0])
```

- **Code Explanation**:
  - A simple model with a single layer was created.
  - sgd was used as the optimizer and mean_squared_error as the loss function.
  - The model was trained using the data, and then predictions were made.

Conclusion:

TensorFlow is a powerful and comprehensive library for developing machine learning solutions. Its ease of use, flexible interfaces, and robust support make it the top choice for many advanced AI projects.

A Practical Example In The File main.ipynb At The Package Of This Task In Folder CV_S_Task.
The Mean Source at TensorFlow Site https://www.tensorflow.org

الطبقات الأساسية في الشبكات العصبية (Conv2D، MaxPooling2D، Dense):

---

1. طبقة Conv2D (Convolutional Layer):

- تُعتبر طبقة Conv2D حجر الأساس في الشبكات العصبية التلافيفية (Convolutional Neural Networks - CNNs).
- وظيفتها استخراج المزايا (features) المهمة من الصور مثل الحواف، الأشكال، أو الأنماط.

الآلية:

- يتم تمرير صورة الإدخال عبر مصفوفة تُسمى kernel (أو filter).
- يتم تطبيق عمليات حسابية تُسمى الالتفاف(convolution) على الصورة.
- النتيجة هي "خريطة مزايا" تمثل المزايا المستخرجة (Feature Map).

المتغيرات الأساسية:

1. عدد الفلاتر (Filters):
   - يحدد عدد مصفوفات الالتفاف (kernels) التي سيتم استخدامها.
   - كل فلتر يتعلم نوعًا معينًا من الأنماط (مثل الخطوط أو الزوايا).
   - مثال: `filters=32` يعني استخدام 32 فلترًا.
2. حجم الفلتر (Kernel Size):
   - يحدد أبعاد كل فلتر (e.x 3x3 or 5x5).
   - مثال: `kernel_size=(3, 3)`.
3. وظيفة التنشيط (Activation Function):

- تُطبق لتحويل القيم الناتجة إلى شكل غير خطي.
- الأكثر شيوعًا: ReLU (Rectified Linear Unit).
- مثال: `activation='relu'`.

4. حجم الخطوة (Strides):
   - يحدد عدد البكسلات التي يتحرك بها الفلتر في كل خطوة.
   - مثال: `strides=(1, 1)` يعني خطوة واحدة أفقية وعمودية.
5. الحشو (Padding):
   - يتحكم في كيفية معالجة الحواف (الزوايا) للصورة.
   - `valid`: بدون حشو (يقل حجم الصورة).
   - `same`: يتم إضافة حشو للحفاظ على نفس أبعاد الصورة.

MaxPooling2D (Pooling Layer) طبقة :

- تُستخدم لتقليل أبعاد خريطة المزايا (Feature Map) الناتجة عن Conv2D .
- وظيفتها تقليل عدد العمليات الحسابية وتقليل احتمالية الإفراط في التخصيص (Overfitting).
- تقوم باختيار القيمة القصوى في نافذة معينة.

الآلية:

- يتم تقسيم خريطة المزايا إلى أقسام صغيرة مثل (2x2) .
- يتم أخذ القيمة القصوى من كل قسم.

المتغيرات الأساسية:

1. حجم النافذة (Pool Size):
   - يحدد حجم النافذة المستخدمة لاختيار القيم القصوى.
   - مثال: `pool_size=(2, 2)`.
2. حجم الخطوة (Strides):
   - يحدد عدد البكسلات التي تتحرك بها النافذة.
   - مثال: `strides=(2, 2)`.

طبقة Dense (Fully Connected Layer):

- تُستخدم في الشبكات العصبية التقليدية أو عند نهاية CNN للقيام بتصنيف أو توقع.

- كل وحدة(Neuron) في هذه الطبقة متصلة بجميع وحدات الطبقة السابقة .

المتغيرات الأساسية:

1. عدد الوحدات (Units):
   - يحدد عدد الخلايا العصبية في الطبقة.
   - مثال: `units=128`.
2. وظيفة التنشيط (Activation Function):
   - تُطبق لتحويل المخرجات.
   - الأكثر شيوعًا: ReLU، Softmax (للتصنيف متعدد الفئات).
   - مثال: `activation='softmax'`.

المتغيرات المشتركة بين الطبقات:

1. الأوزان (Weights) والتحيزات (Biases):
   - الأوزان: معاملات تتعلمها كل طبقة أثناء التدريب.
   - التحيزات: قيم إضافية تُضاف إلى المخرجات لزيادة المرونة.
2. معدل التعلم (Learning Rate):
   - يتحكم في مقدار التحديث للأوزان أثناء كل خطوة تدريب.
   - يتم ضبطه غالبًا باستخدام المُحسّن مثل (Adam، SGD)
   - مثال: `learning_rate=0.001`.

-

مثال عملي :

```python
from tensorflow.keras.models import Sequential # type: ignore
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense # type: ignore

# إنشاء نموذج باستخدام الطبقات الأساسية
model = Sequential([
    Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(units=128, activation='relu'),
    Dense(units=10, activation='softmax')  # للتصنيف إلى 10 فئات
])
model.summary()
```

إنشاء نموذج متسلسل باستخدام `Sequential`:

- **يضيف الطبقات بشكل تسلسلي.**

استخدام طبقة `Dense`:

- **طبقة Dense الأولى تحتوي على `128` وحدة مع وظيفة تنشيط `ReLU`.**
- **طبقة Dense الثانية (النهائية) تحتوي على `10` وحدات مع وظيفة تنشيط `Softmax` للتصنيف إلى 10 فئات.**

تبسيط الفكرة بمثال يومي:

1. **Conv2D:**
   - **لنفكر في مصفاة مياه تقوم باستخراج الأوساخ من الماء. هنا، الفلتر هو "المصفاة" والصورة هي "الماء".**
2. **MaxPooling2D:**
   - **فكر في تصغير صورة كبيرة إلى صورة مصغرة عن طريق أخذ التفاصيل الأهم فقط.**
3. **Dense:**
   - **تخيل جدولًا يتم جمع بياناته من مصادر مختلفة وربطها لإعطاء النتيجة النهائية (مثل متوسط درجات الطلاب).**

بهذه الطريقة، تتعاون الطبقات معًا لتشكيل شبكة عصبية يمكنها التعلم والتصنيف.

- **CNN**

Comprehensive Report on Convolutional Neural Networks (CNNs)

Definition of CNNs: Convolutional Neural Networks (CNNs) are a specialized class of artificial neural networks primarily designed for processing structured grid data such as images. CNNs excel at tasks involving visual data processing, including image classification, object detection, and segmentation. Unlike traditional neural networks, CNNs use convolutional layers to automatically and adaptively learn spatial hierarchies of features, enabling the extraction of relevant patterns from images or other structured data.

Key Features of CNNs:

1. **Convolutional Layers:**
    - **These layers apply a convolution operation, extracting features like edges, textures, and shapes from input data.**
    - **Filters (kernels) slide over the input data to create feature maps.**
2. **Pooling Layers:**
    - **Pooling reduces the dimensionality of feature maps while retaining critical information.**
    - **Common pooling operations include max pooling and average pooling.**
3. **Fully Connected Layers:**
    - **These layers connect every neuron from the preceding layer, aggregating extracted features for final classification or regression.**
4. **Activation Functions:**

    - **Non-linear functions like ReLU (Rectified Linear Unit) introduce non-linearity, enabling the model to capture complex patterns.**
5. **Regularization Techniques:**
    - **Methods such as dropout and weight decay help prevent overfitting.**

Applications of CNNs:

- **Image Classification: Recognizing objects in images (e.g., cat vs. dog).**
- **Object Detection: Identifying and localizing objects within images.**
- **Segmentation: Partitioning images into meaningful regions.**
- **Medical Imaging: Diagnosing diseases from radiological scans.**
- **Natural Language Processing: In tasks like text classification and sentence parsing.**

Steps in Building a CNN Model:

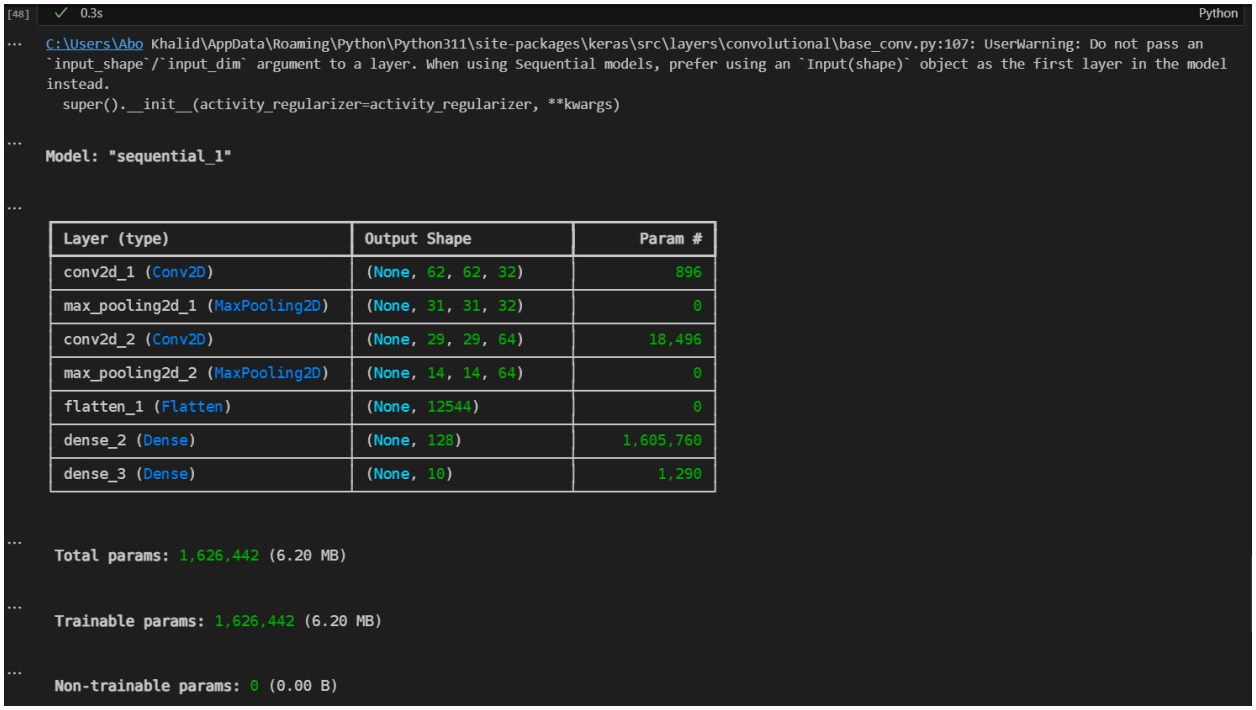
1. **Define Input Data: Images are typically formatted as tensors with dimensions (height, width, channels).**
2. **Construct the Model: Combine convolutional, pooling, and fully connected layers.**
3. **Compile the Model: Specify loss functions, optimizers, and metrics.**
4. **Train the Model: Feed labeled data into the model, iterating to minimize error.**
5. **Evaluate the Model: Test the model on unseen data.**
6. **Deploy: Use the trained model for inference tasks.**

Code Example: The following code snippet demonstrates building a simple CNN using TensorFlow:

```
from tensorflow.keras.models import Sequential # type: ignore
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense # type: ignore

# Initialize the CNN
model = Sequential([
    Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(units=128, activation='relu'),
    Dense(units=10, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Summary of the model
model.summary()
```

References:

- **TensorFlow Official Documentation: https://www.tensorflow.org**
- **Deep Learning Book by Ian Goodfellow: https://www.deeplearningbook.org**
- **Stanford CS231n Convolutional Neural Networks for Visual Recognition: http://cs231n.stanford.edu/**



أغلب الاعتماد على المراجع التالية :

Deep Learning