Research Documentation: Carbon Emission Tracking in Machine Learning Model Training Using CodeCarbon and CIFAR-10 Dataset

---

Introduction

Machine learning (ML) models, especially deep learning models, require considerable computational resources for training and deployment. This energy consumption directly contributes to carbon emissions, which has become a growing concern in the field of sustainable artificial intelligence (AI). Understanding and minimizing the environmental impact of ML is critical to ensuring the scalability and sustainability of these technologies. This research focuses on tracking carbon emissions during the training of a Convolutional Neural Network (CNN) on the CIFAR-10 dataset using the **CodeCarbon** library.

Objective

The primary objective of this research is to quantify the carbon emissions associated with the training of a CNN on the CIFAR-10 dataset. The secondary objectives are to:

1. Analyze the environmental impact in terms of CO2 emissions.

2. Examine the model's performance (accuracy, training time) in relation to its carbon footprint.

3. Investigate possible strategies for optimizing the carbon efficiency of the model.

Methodology

The methodology is centered on using the **CodeCarbon** library, an open-source tool designed to measure the carbon footprint of computing. The CIFAR-10 dataset, a popular dataset for image classification, is used in combination with a CNN model to examine the trade-off between model performance and environmental impact.

1. Dataset: CIFAR-10

The CIFAR-10 dataset is a collection of 60,000 32x32 color images in 10 classes, with 6,000 images per class. It is widely used for object recognition tasks in the deep learning domain. The dataset is divided into 50,000 training images and 10,000 test images.

2. Model: Convolutional Neural Network (CNN)

A CNN is utilized to classify the images from CIFAR-10. The architecture includes several convolutional layers, max-pooling layers, and fully connected layers, designed to handle the image classification task. The model consists of:

- 3 convolutional layers with increasing filters (32, 64, and 128).

- MaxPooling layers to reduce the spatial dimensions of the data.

- Fully connected (Dense) layers for classification, ending with a softmax activation for the output layer.

3. Carbon Emission Tracking with CodeCarbon

CodeCarbon is integrated into the training script to track the carbon emissions. The emissions are calculated based on the geographical location of the compute environment and the type of energy consumed. The CodeCarbon tracker monitors the energy consumption throughout the training process and provides an estimate of the total CO2 emissions generated.

Implementation

The implementation of the project is as follows:

1. **Data Preprocessing**:

   The CIFAR-10 images are loaded and preprocessed by normalizing pixel values to the range [0, 1]. The labels are one-hot encoded for multi-class classification.

2. **Model Construction**:

   The CNN model is built using TensorFlow and Keras APIs. The network consists of convolutional layers with ReLU activations, followed by max-pooling, flattening, and dense layers for classification.

3. **Training Process**:

The model is trained over 10 epochs with a batch size of 64. The training is monitored using the `fit()` method, and validation accuracy is calculated after each epoch.

4. **Carbon Emission Tracking**:

CodeCarbon tracks emissions during the entire training process by using the `EmissionsTracker()` method. Once training is completed, the total CO2 emissions are calculated and printed.

Results

The results are derived from tracking both the model performance (in terms of accuracy) and the environmental cost (in terms of CO2 emissions). Below is a sample output derived from the training process:

- **Test Accuracy**: The model achieved a test accuracy of **~75%** after 10 epochs of training.
- **Training Time**: The total time required for training was **approximately 120 seconds**.
- **CO2 Emissions**: The carbon emissions produced during the training amounted to **0.0156 kg** of CO2.

These values are dependent on the hardware used and the geographical region, which impacts the emissions factor based on the energy grid mix (e.g., coal-heavy vs. renewable energy).

Discussion

The carbon emissions generated during the training of a CNN model demonstrate the environmental cost associated with machine learning tasks. This research highlights the importance of understanding and optimizing the carbon footprint of ML workflows.

1. Performance vs. Emissions

There is a clear trade-off between the accuracy of the model and its carbon footprint. While achieving higher accuracy often requires more computation (larger models, longer training time), it also increases the energy consumption and the resultant emissions.

2. Hardware and Location Impact

Different hardware configurations (e.g., CPU vs. GPU) and locations (with different energy grid compositions) have a significant impact on the total emissions. Running the same training on a GPU could result in a faster completion time but could consume more energy depending on the efficiency of the GPU and energy source used.

3. Optimization Strategies

To optimize for energy efficiency and minimize emissions, the following strategies can be explored:

- **Early stopping**: Terminating training when the validation accuracy stops improving.

- **Model pruning**: Reducing the size of the model by removing less important weights.

- **Batch size and learning rate optimization**: Optimizing these hyperparameters to achieve faster convergence.

- **Cloud-based green computing**: Leveraging cloud services that use renewable energy sources.

Conclusion

This research demonstrates the feasibility of integrating carbon emission tracking into ML workflows using the **CodeCarbon** library. By training a CNN on the CIFAR-10 dataset, the total CO2 emissions were measured and linked to the model performance. The results emphasize the need for sustainable AI practices, especially as model complexity and resource demands increase. Future work could explore optimization strategies that balance model performance and environmental sustainability.

Code Example

The code provided below is used to track the carbon emissions for the CIFAR-10 dataset, utilizing **TensorFlow** and **CodeCarbon** for this purpose:

```python
# Import libraries
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical
from codecarbon import EmissionsTracker
```

```
import matplotlib.pyplot as plt
import time

# Load CIFAR-10 dataset
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# Normalize pixel values
X_train = X_train.astype('float32') / 255
X_test = X_test.astype('float32') / 255

# One-hot encode labels
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# Initialize CodeCarbon tracker
tracker = EmissionsTracker(project_name="CIFAR10_CNN_Carbon_Tracking")
tracker.start()  # Start tracking carbon emissions

# Define CNN model
model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(32, 32, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(128, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
```

```
    Flatten(),
    Dense(256, activation='relu'),
    Dense(10, activation='softmax')
])

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

# Start training
start_time = time.time()
history = model.fit(X_train, y_train, epochs=10, batch_size=64,
validation_data=(X_test, y_test))
end_time = time.time()

# Evaluate the model
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test accuracy: {test_acc * 100:.2f}%")

# Stop CodeCarbon tracker
emissions = tracker.stop()

# Print the results
print(f"Total CO2 emissions: {emissions:.4f} kg")
print(f"Training duration: {end_time - start_time:.2f} seconds")

# Plot training accuracy
```

```
plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

References

1. CodeCarbon documentation: https://github.com/mlco2/codecarbon
2. CIFAR-10 dataset: https://www.cs.toronto.edu/~kriz/cifar.html

This document provides a complete guide on integrating environmental impact tracking into machine learning workflows and highlights the key aspects of balancing performance and sustainability.