Bab 13: Loading and Preprocessing Data with TensorFlow

Tujuan Bab

Menjelaskan cara memuat, memproses, dan menyiapkan data secara efisien menggunakan TensorFlow's tf.data API, untuk mendukung pelatihan skala besar yang cepat dan dapat diskalakan.

Konsep Utama

1. Kebutuhan Pipeline Data yang Efisien

Untuk deep learning berskala besar:

- Data harus diproses dengan cepat
- Harus bisa diparalelkan dan *prefetched*
- Harus mendukung transformasi kompleks dan augmentasi

TensorFlow menyediakan modul tf.data untuk ini.

2. Membuat Dataset dari Numpy

```
import tensorflow as tf

dataset = tf.data.Dataset.from_tensor_slices((features, labels))
```

Dapat digunakan dalam model.fit() langsung jika features dan labels adalah array atau tensor.

3. Method Penting pada tf.data.Dataset

Method	Fungsi
batch(n)	Membagi data ke batch ukuran n
shuffle(n)	Mengacak data dengan buffer size n
repeat(n)	Mengulang dataset n kali

map(func)	Transformasi data baris per baris
prefetch(n)	Menyiapkan batch di latar belakang
cache()	Menyimpan dataset di memori/disk setelah satu epoch

Contoh penggunaan:

```
dataset = dataset.shuffle(1000).batch(32).prefetch(1)
```

4. Membaca Data dari File

tf.data mendukung banyak format:

- Text files (CSV, txt)
- Binary files (TFRecord)
- Image files

Contoh: Membaca file CSV

```
def parse_csv_line(line):
   defs = [0.] * 8 + [0]  # kolom fitur dan label
   parsed = tf.io.decode_csv(line, record_defaults=defs)
   return tf.stack(parsed[:-1]), parsed[-1]


dataset = tf.data.TextLineDataset("data.csv").skip(1).map(parse_csv_line)
```

5. TFRecord Format

Format biner terkompresi dan efisien untuk data skala besar.

- Menggunakan tf.io.TFRecordWriter() untuk menulis
- Dibaca dengan TFRecordDataset()
- Untuk menyimpan objek: gunakan *protocol buffers*

6. Pipelines yang Kompleks

Transformasi dapat dirangkai seperti pipeline Scikit-Learn:

```
def preprocess(features, label):
    # normalisasi, augmentasi, dsb
    return transformed_features, label

train_set = dataset.map(preprocess).shuffle(1000).batch(32).prefetch(1)
```

7. Performance Tips

- Gunakan .cache() untuk dataset kecil
- Gunakan .prefetch(tf.data.AUTOTUNE)
- Gunakan num_parallel_calls=tf.data.AUTOTUNE di map()

Contoh:

```
dataset.map(func, num_parallel_calls=tf.data.AUTOTUNE).prefetch(tf.data.AUTOTUNE)
```

Proyek / Notebook Praktik

Isi:

- Membaca dataset CSV menggunakan TextLineDataset
- Membuat pipeline transformasi menggunakan map()
- Menambahkan shuffle, batch, prefetch, dan cache
- Membaca TFRecord files
- Menyusun input pipeline efisien untuk pelatihan

Inti Pelajaran

Konsep	Penjelasan

tf.data.Dataset	Abstraksi pipeline data efisien dan scalable
.map()	Transformasi tiap elemen dataset
.shuffle()	Meningkatkan generalisasi model
.batch()	Mengelompokkan data untuk pelatihan
.prefetch()	Meminimalkan bottleneck I/O
.cache()	Menyimpan data agar dibaca hanya sekali
TFRecord	Format data biner efisien untuk produksi

Kesimpulan

Bab ini membekali dengan kemampuan untuk menyiapkan pipeline input yang efisien dan andal menggunakan tf.data. Pipeline yang baik sangat krusial untuk mempercepat pelatihan dan menghindari bottleneck saat memproses data besar atau kompleks, terutama untuk image dan video.