Chapter 19: Melatih dan Mendeploy Model TensorFlow dalam Skala Besar

Pendahuluan

Setelah berhasil membangun dan melatih sebuah model Machine Learning, pekerjaan belum selesai. Model tersebut perlu "dihidupkan" agar dapat memberikan nilai, yaitu dengan cara di-*deploy* ke lingkungan produksi di mana ia dapat menerima data baru dan membuat prediksi. Selain itu, seiring dengan pertumbuhan data, kita seringkali dihadapkan pada tantangan di mana dataset menjadi terlalu besar atau pelatihan memakan waktu terlalu lama untuk ditangani oleh satu mesin. Chapter ini membahas dua aspek krusial dalam siklus hidup Machine Learning: deployment (penyebaran model) dan pelatihan dalam skala besar (*training at scale*), menggunakan ekosistem TensorFlow.

1. Mendeploy Model ke Produksi

Deployment adalah proses mengintegrasikan model Machine Learning ke dalam lingkungan produksi untuk membuat prediksi pada data baru yang belum pernah dilihat sebelumnya.

TensorFlow Serving

Daripada membuat server web manual dengan skrip Python untuk menjalankan model.predict(), pendekatan yang jauh lebih andal dan dapat diskalakan adalah dengan menggunakan sistem khusus seperti TensorFlow Serving. Ini adalah sistem *serving* fleksibel dan berkinerja tinggi yang dirancang khusus untuk lingkungan produksi.

Alur Kerja dengan TF Serving:

1. Menyimpan Model: Langkah pertama adalah mengekspor model Keras yang telah dilatih ke dalam format SavedModel TensorFlow. Format ini bersifat universal, independen dari bahasa pemrograman, dan berisi semua informasi yang diperlukan untuk menjalankan model: arsitektur, bobot, dan "tanda tangan" (*signature*) komputasi.
2. Menjalankan TF Serving: TF Serving, yang ditulis dalam C++ untuk performa maksimal, biasanya dijalankan menggunakan Docker. Kita cukup menunjuk TF Serving ke direktori yang berisi versi-versi model kita.
3. Manajemen Model: Salah satu keunggulan TF Serving adalah kemampuannya mengelola beberapa versi model secara bersamaan. Ia dapat secara otomatis mendeteksi versi model baru dan memuatnya tanpa *downtime*, memungkinkan pembaruan model yang mulus.
4. Membuat Permintaan (Querying): Aplikasi klien berinteraksi dengan model yang di-*serve* dengan mengirimkan permintaan prediksi melalui API. Ada dua protokol utama:
    - REST API: Menggunakan permintaan HTTP standar. Lebih mudah diintegrasikan dan dipahami.
    - gRPC API: Sebuah protokol RPC (Remote Procedure Call) berkinerja tinggi yang dikembangkan oleh Google. gRPC lebih efisien dan memiliki latensi lebih rendah daripada REST, menjadikannya pilihan yang lebih baik untuk lingkungan produksi dengan volume permintaan tinggi.

TensorFlow Lite (TFLite) untuk Perangkat Mobile dan Embedded

Untuk mendeploy model pada perangkat dengan sumber daya terbatas (seperti smartphone, Raspberry Pi, atau mikrokontroler), kita memerlukan format model yang sangat ringan dan efisien. TensorFlow Lite (TFLite) adalah kerangka kerja yang dirancang untuk tujuan ini.

Proses Konversi TFLite:

1. Model TensorFlow atau Keras yang sudah dilatih diubah menggunakan TFLite Converter.
2. Converter ini mengubah model menjadi format file .tflite yang sangat dioptimalkan. Selama konversi, beberapa teknik optimisasi dapat diterapkan, yang paling penting adalah kuantisasi (quantization).
3. Kuantisasi: Teknik ini mengurangi presisi dari bobot dan aktivasi model, biasanya dari 32-bit *floating-point* menjadi 8-bit *integer*. Hal ini secara drastis mengurangi ukuran model (sekitar 4x lebih kecil) dan mempercepat inferensi secara signifikan pada perangkat keras yang mendukungnya, seringkali dengan penurunan akurasi yang minimal.
4. Di perangkat, kita menggunakan TFLite Interpreter yang ringan untuk memuat file .tflite dan menjalankan inferensi.

2. Melatih Model dalam Skala Besar

Ketika dataset terlalu besar untuk muat di memori satu mesin atau ketika pelatihan membutuhkan waktu berminggu-minggu, kita perlu mendistribusikan proses pelatihan ke beberapa mesin atau beberapa GPU.

Data Parallelism

Strategi yang paling umum untuk pelatihan terdistribusi adalah Data Parallelism. Konsepnya sederhana:

1. Replikasi: Model yang sama direplikasi di setiap perangkat (disebut *worker*), yang bisa berupa GPU di satu mesin atau beberapa mesin di sebuah kluster.
2. Partisi Data: Dataset pelatihan dibagi menjadi beberapa bagian (*shards*).
3. Pelatihan Paralel: Setiap *worker* memproses *batch* data yang berbeda dari *shard*-nya masing-masing secara paralel dan menghitung gradien secara independen.
4. Agregasi Gradien: Gradien dari semua *worker* dikumpulkan, dirata-ratakan, dan kemudian pembaruan bobot diterapkan secara sinkron ke semua replika model.

Distribution Strategy API di TensorFlow

TensorFlow menyediakan Distribution Strategy API tingkat tinggi untuk menyederhanakan implementasi pelatihan terdistribusi. Kita cukup mendefinisikan sebuah strategi dan membuat serta mengompilasi model di dalam lingkup (scope) strategi tersebut. Keras dan TensorFlow akan menangani semua komunikasi dan sinkronisasi yang rumit di belakang layar.

Strategi Utama yang Tersedia:

- MirroredStrategy: Digunakan untuk pelatihan pada beberapa GPU di satu mesin. Model dan variabelnya "dicerminkan" di setiap GPU. Agregasi gradien dilakukan dengan sangat efisien menggunakan algoritma *all-reduce*.
- MultiWorkerMirroredStrategy: Versi multi-mesin dari MirroredStrategy. Digunakan untuk pelatihan sinkron di beberapa mesin, di mana setiap mesin dapat memiliki satu atau lebih GPU.
- ParameterServerStrategy: Mendukung pelatihan asinkron di beberapa mesin. Dalam skema ini, beberapa mesin bertindak sebagai Parameter Servers (yang menyimpan variabel model) dan yang lain bertindak sebagai Workers (yang melakukan komputasi). *Worker* secara independen mengambil bobot, menghitung gradien, dan mengirim pembaruan kembali ke server. Pelatihan asinkron lebih toleran terhadap *worker* yang lambat tetapi bisa lebih rumit untuk disetel.

Kesimpulan

TensorFlow menyediakan ekosistem yang matang untuk seluruh siklus hidup proyek Machine Learning, melampaui sekadar desain model. Dengan TF Serving, model dapat di-*deploy* secara andal untuk melayani jutaan pengguna. Dengan TF Lite, model dapat dijalankan secara efisien di perangkat seluler dan IoT. Dan dengan Distribution Strategy API, tantangan dalam melatih model pada dataset berukuran masif dapat diatasi dengan mendistribusikan beban kerja ke banyak GPU atau mesin, memungkinkan kita untuk membangun model yang lebih besar dan lebih akurat dari sebelumnya.