Chapter 12: Summary

Diedrick Darrell Darmadi - 1103223031

Bab 12 membahas bagaimana kita dapat membuat model machine learning yang benar-benar fleksibel menggunakan TensorFlow dengan membangun *custom model*, *custom layer*, dan *custom training loop*. Jika pada bab sebelumnya kita terbiasa menggunakan API tingkat tinggi seperti Sequential, Functional API, serta model.fit(), maka pada bab ini kita "membuka kap mesin" TensorFlow untuk memahami bagaimana proses pelatihan sebenarnya bekerja. Pendekatan ini memberi kebebasan penuh ketika arsitektur yang ingin dibangun tidak cocok dengan struktur standar Keras.

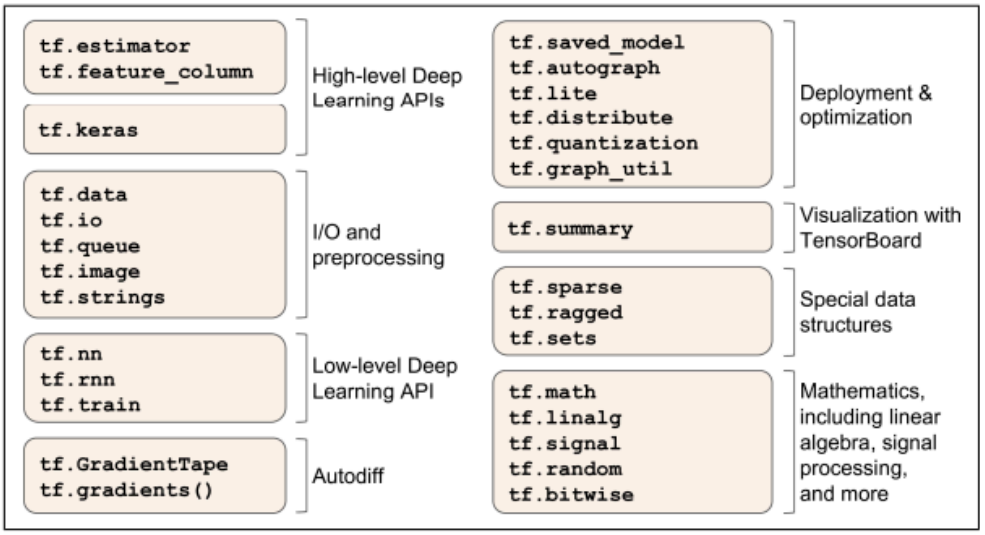

Figure 12-1. TensorFlow's Python API

Pembahasan dimulai dengan cara membuat **model kustom** menggunakan turunan dari tf.keras.Model. Dengan pendekatan ini, kita sendiri mendefinisikan apa yang terjadi pada *forward pass* melalui method call(). Teknik ini bermanfaat ketika arsitektur memiliki alur perhitungan yang tidak beraturan—misalnya model dengan beberapa jalur seperti *Wide & Deep model*, model dengan percabangan logika, ataupun model yang membutuhkan manipulasi tensor yang tidak umum. Dengan membuat model secara manual, kita memiliki kontrol total atas setiap langkah komputasi.

Selanjutnya, bab ini menjelaskan bagaimana membuat **layer kustom** menggunakan tf.keras.layers.Layer. Di sini kita belajar mendefinisikan bobot secara manual menggunakan add_weight() serta mengatur sendiri operasi matematis yang terjadi pada layer tersebut. Layer kustom sangat penting jika kita membutuhkan operasi unik yang tidak tersedia pada layer bawaan TensorFlow, seperti normalisasi khusus, transformasi matematis tertentu, atau teknik regularisasi baru.

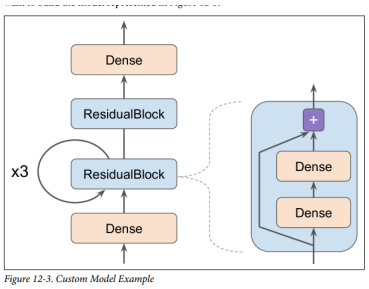

Figure 12-3. Custom Model Example

Bab ini juga membahas tentang pembuatan **loss function kustom**. Meskipun TensorFlow menyediakan banyak loss umum seperti MSE, MAE, dan cross-entropy, ada banyak kasus di mana kita membutuhkan loss khusus yang mengikuti kebutuhan riset atau struktur model tertentu. Loss kustom dapat dibuat sebagai fungsi biasa atau sebagai kelas turunan dari tf.keras.losses.Loss. Contoh seperti *Huber loss* digunakan untuk menjelaskan bagaimana loss

hybrid dapat dirancang agar lebih stabil terhadap outlier.

Bagian terpenting dari bab ini adalah pembahasan tentang **custom training loop**. Dengan memanfaatkan tf.GradientTape(), kita dapat mengatur secara manual bagaimana forward pass dihitung, bagaimana loss diperoleh, dan bagaimana gradien dihitung serta diaplikasikan ke bobot model. Pendekatan ini memberikan fleksibilitas maksimal, terutama untuk model seperti GAN, *reinforcement learning*, *meta-learning*, atau situasi lain yang memerlukan dua optimizer, langkah pembaruan berbeda, atau manipulasi gradien secara manual. Kita juga belajar cara menambahkan metrik sendiri serta melakukan logging selama proses training.

Selain itu, bab ini memperkenalkan penggunaan dekorator **@tf.function**, yang mengubah fungsi Python menjadi grafik TensorFlow yang dapat dioptimalkan. Dengan mengompilasi fungsi training, proses pelatihan menjadi jauh lebih cepat dan efisien, terutama ketika dijalankan di GPU atau TPU. TensorFlow dapat melakukan optimasi seperti *graph pruning*, *constant folding*, dan eksekusi paralel yang tidak bisa dilakukan pada mode eager execution biasa.

Secara keseluruhan, Bab 12 memberikan pemahaman fundamental tentang bagaimana TensorFlow bekerja di balik layar. Bab ini mempersenjatai pembaca dengan kemampuan untuk membangun arsitektur deep learning yang kompleks dan tidak konvensional, menjalankan skenario training yang penuh penyesuaian, serta mengoptimalkan performa model dengan kontrol penuh terhadap setiap tahapan komputasi. Pemahaman dari bab ini sangat penting terutama bagi mereka yang ingin mengembangkan model penelitian, arsitektur baru, atau membuat sistem yang memerlukan modifikasi khusus dalam mekanisme training.