Nama : Eko Putra Nugraha

NIM : 1103213212

Classification Dummy

Proses pengembangan dan eksperimen model deep learning dalam kode yang telah dibuat dimulai dari tahap persiapan data. Dataset dummy dihasilkan menggunakan make_classification, yang menciptakan data sintetik dengan fitur relevan untuk tugas klasifikasi. Fitur-fitur ini kemudian distandardisasi menggunakan StandardScaler untuk memastikan bahwa semua fitur memiliki skala yang sama, yaitu mean 0 dan standar deviasi 1. Langkah ini penting untuk meningkatkan stabilitas dan kecepatan konvergensi model selama pelatihan. Untuk mempercepat proses eksperimen, hanya 50% dari dataset yang digunakan dengan memilih subset data secara acak. Setelah itu, dataset dibagi menjadi data latih (80%) dan data uji (20%), memastikan model dapat dilatih dan dievaluasi secara terpisah.

Data yang telah diproses kemudian dikonversi ke dalam format tensor menggunakan PyTorch, yang merupakan format standar untuk pemrosesan data dalam deep learning. Tensor ini dipindahkan ke GPU jika tersedia, karena GPU mampu mempercepat proses komputasi dengan mendukung operasi paralel. Data ini kemudian dimasukkan ke dalam DataLoader, sebuah fitur dari PyTorch yang memungkinkan data diambil dalam batch-batch kecil. Ukuran batch disesuaikan dengan berbagai parameter untuk melihat pengaruhnya terhadap kinerja model.

Model yang dirancang adalah Multilayer Perceptron (MLP), yang memiliki struktur fleksibel dengan berbagai kombinasi jumlah hidden layer, jumlah neuron per layer, dan fungsi aktivasi. Fungsi aktivasi seperti ReLU, Sigmoid, dan Tanh digunakan untuk memperkenalkan non-linearitas ke dalam model, yang penting untuk menangkap hubungan kompleks antara fitur dan target. Lapisan terakhir memiliki dua neuron, sesuai dengan jumlah kelas dalam dataset.

Pelatihan model dilakukan menggunakan AdamW, sebuah optimizer modern yang dikenal karena kemampuannya menangani berbagai skenario dengan baik. Early stopping diterapkan untuk menghentikan pelatihan jika model tidak menunjukkan peningkatan akurasi selama beberapa epoch berturut-turut. Ini membantu mencegah pelatihan berlebih (overfitting) dan menghemat waktu pelatihan. Evaluasi dilakukan dengan menghitung akurasi pada data uji, yang memberikan gambaran tentang kemampuan model untuk menggeneralisasi pada data yang tidak terlihat sebelumnya.

Eksperimen dilakukan dengan mencoba berbagai kombinasi hyperparameter, termasuk jumlah hidden layer, jumlah neuron, fungsi aktivasi, jumlah epoch, learning rate, dan ukuran batch. Hasil dari semua eksperimen ini dicatat dalam sebuah tabel untuk dianalisis lebih lanjut. Akurasi terbaik dibandingkan untuk menentukan kombinasi hyperparameter yang paling optimal. Proses ini menunjukkan pentingnya eksplorasi parameter dalam membangun model deep learning yang efektif dan efisien.

Dengan struktur ini, proses pengembangan berjalan sistematis, mulai dari pemrosesan data, desain model, pelatihan, hingga analisis hasil. Pendekatan ini memberikan wawasan yang kuat tentang bagaimana setiap keputusan desain dan konfigurasi memengaruhi performa model.

Heart Disease

Kode tersebut bertujuan untuk mengevaluasi performa model *Multilayer Perceptron* (MLP) dengan berbagai konfigurasi parameter menggunakan dataset penyakit jantung (*heart.csv*). Model dirancang untuk tugas klasifikasi biner, di mana outputnya menentukan apakah seseorang memiliki penyakit jantung (1) atau tidak (0). Kode ini menggunakan PyTorch untuk implementasi jaringan saraf tiruan, dengan kombinasi parameter seperti jumlah *hidden layers*, fungsi aktivasi, *learning rate*, *batch size*, dan jumlah *epochs*.

Dataset pertama-tama diproses dengan normalisasi menggunakan *StandardScaler* untuk memastikan semua fitur memiliki distribusi seragam, yang penting untuk stabilitas pelatihan jaringan saraf. Data dibagi menjadi data latih dan data uji dengan rasio 80:20. Setelah itu, data diformat menjadi tensor PyTorch untuk digunakan dalam proses pelatihan model.

Model MLP diimplementasikan sebagai kelas MLPModel, dengan arsitektur yang fleksibel untuk menambahkan sejumlah *hidden layers* dan jenis fungsi aktivasi berdasarkan parameter input. Proses pelatihan dilakukan dalam fungsi train_model_with_logging, yang mencatat nilai *loss* (baik pada data latih maupun uji) dan akurasi pada setiap epoch. Informasi ini disimpan untuk visualisasi, sehingga pengguna dapat memantau perkembangan model dari waktu ke waktu.

Visualisasi hasil pelatihan sangat penting untuk memahami performa model. Grafik pertama menampilkan perubahan nilai *loss* untuk data latih dan uji selama *epochs*. Penurunan *loss* pada kedua grafik menunjukkan bahwa model berhasil mempelajari pola dari data latih dan mampu menggeneralisasi pada data uji. Grafik kedua menunjukkan perubahan akurasi model pada data uji, yang mengindikasikan bagaimana kemampuan prediksi model meningkat seiring pelatihan.

Melalui kombinasi parameter yang diuji, kode ini memungkinkan pengguna untuk mengidentifikasi konfigurasi optimal dari jumlah *hidden layers*, fungsi aktivasi, dan *hyperparameter* lainnya. Dengan demikian, pengguna dapat memperoleh wawasan tentang bagaimana setiap parameter memengaruhi performa model secara keseluruhan. Output berupa log pelatihan dan visualisasi memberikan cara intuitif untuk menganalisis hasil dan membuat keputusan berbasis data.