1. Cara kerja algoritma Artificial Neural Network dari scratch pada repository ini

a. Inisialisasi parameter dan struktur jaringan

Algoritma dimulai dengan inisialisasi struktur jaringan dan parameter-parameter yang diperlukan. Model ANN yang diimplementasikan adalah jenis *Feedforward Neural Network* yang terdiri dari beberapa layer yang terhubung penuh (*fully connected*), termasuk *input layer*, *hidden layers*, dan *output layer*. Setiap layer didefinisikan dengan jumlah neuron spesifik, fungsi aktivasi (seperti sigmoid, ReLU, linear, atau softmax), dan metode inisialisasi bobot (misalnya *He initialization* untuk fungsi ReLU). Bobot (*weights*) diinisialisasi secara acak sesuai dengan metode inisialisasi yang dipilih, sedangkan bias (*biases*) diinisialisasi dengan nilai nol.

Selain itu, parameter penting seperti learning rate, jumlah iterasi (*epochs*), ukuran batch (*batch size*), jenis regularisasi (L1, L2, atau tidak ada), fungsi loss (MSE atau cross-entropy), dan jumlah neuron di setiap layer diatur selama inisialisasi. Pengaturan ini memungkinkan jaringan untuk fleksibel dalam menangani berbagai jenis masalah.

b. Forward Propagation

Pada tahap forward propagation, input data melewati setiap layer jaringan secara berurutan dimulai dari *input layer* hingga *output layer* untuk menghasilkan prediksi akhir. Proses ini mencakup beberapa langkah berikut:

- **Kalkulasi Linear**

  Untuk setiap neuron di layer saat ini, input dari layer sebelumnya dikalikan dengan bobot neuron dan ditambahkan bias

$$z = X \cdot W + b$$

  Di sini, *X* adalah input ke layer tersebut, *W* adalah bobot, dan b adalah bias.

- **Aktivasi Non-Linear**

  Nilai z kemudian dilewatkan melalui fungsi aktivasi yang dipilih untuk layer tersebut untuk menghasilkan output non-linear

$$A = \text{activation}(z)$$

  Aktivasi ini bisa berupa sigmoid (untuk output antara 0 dan 1), ReLU (untuk menangani non-linearity), linear (untuk regresi), atau softmax (untuk klasifikasi multi kelas).

- **Output Layer**
  Pada *output layer*, fungsi aktivasi seperti softmax digunakan untuk menghasilkan probabilitas dari kelas prediksi.

c. Perhitungan Loss

Setelah forward propagation, jaringan menghasilkan output prediksi yang kemudian digunakan untuk menghitung loss atau galat terhadap nilai aktual. Fungsi loss yang digunakan adalah mean squared error (MSE) untuk regresi dan binary cross-entropy untuk klasifikasi biner.

d. Backward Propagation

Tahap berikutnya adalah backward propagation. Backward propagation adalah tahap di mana gradien dari fungsi loss dihitung terhadap bobot dan bias jaringan, mulai dari output layer ke input layer.

- Pertama, gradien dari loss function terhadap output layer dihitung.
- Gradien ini kemudian digunakan untuk menghitung gradien terhadap bobot, bias, dan input di setiap layer dengan bergerak mundur melalui jaringan. Misalnya, untuk setiap layer, gradien bobot dihitung dengan mempertimbangkan kontribusi error terhadap perubahan pada bobot tersebut.

Jika regularisasi diterapkan, penalti regularisasi ditambahkan ke gradien bobot selama backward propagation. Untuk regularisasi L1, penalti berdasarkan absolut nilai bobot ditambahkan, sedangkan untuk L2, penalti berdasarkan kuadrat dari nilai bobot ditambahkan.

d. Pembaruan Bobot dan Bias

Setelah semua gradien dihitung selama backward propagation, bobot dan bias diperbarui menggunakan metode batch gradient descent. Pembaruan dilakukan dengan mengurangi nilai gradien yang sudah dihitung dari bobot dan bias saat ini, dikalikan dengan learning rate.

e. Iterasi dan Pembelajaran

Proses forward propagation, perhitungan loss, backward propagation, dan pembaruan bobot diulang untuk setiap batch data dalam dataset selama sejumlah epoch yang ditentukan. Tujuan dari proses ini adalah untuk mengurangi nilai loss secara bertahap, yang pada akhirnya meningkatkan akurasi model. Setiap beberapa epoch, loss dihitung ulang dan dicetak untuk memonitor kinerja model dan memastikan konvergensi.

f. Prediksi

Setelah model selesai dilatih, model dapat digunakan untuk memprediksi kelas atau nilai pada data baru (X_test). Proses prediksi melibatkan forward propagation saja, di mana input data baru dimasukkan ke dalam jaringan dan hasil akhirnya adalah prediksi model.

4. Perbandingan hasil evaluasi model

Berdasarkan hasil evaluasi model pada set uji dan K-Fold Cross-Validation, model ANN_Scratch dan ANN_Keras menunjukkan performa yang berbeda. ANN_Keras memiliki akurasi yang sedikit lebih tinggi (0.81) dibandingkan dengan ANN_Scratch (0.80) pada set uji, sementara F1 Score keduanya sama di angka 0.59. Pada K-Fold Cross-Validation, ANN_Keras juga unggul dengan akurasi rata-rata 0.82 ± 0.02 dan F1 Score 0.62 ± 0.05, dibandingkan dengan ANN_Scratch yang memiliki akurasi 0.81 ± 0.01 dan F1 Score 0.60 ± 0.03.

Perbedaan ini mungkin disebabkan oleh penggunaan optimizer yang berbeda. ANN_Keras menggunakan optimizer Adam, yang lebih canggih dan mampu menyesuaikan learning rate secara dinamis, memungkinkan konvergensi lebih cepat dan stabil dibandingkan batch gradient descent sederhana yang digunakan di ANN_Scratch. Selain itu, implementasi internal Keras untuk regularisasi mungkin juga lebih optimal sehingga embantu mengurangi overfitting dan meningkatkan performa pada data yang tidak terlihat.

5. Improvement yang bisa dilakukan

Untuk meningkatkan kinerja model ANN yang telah diimplementasikan, ada beberapa perbaikan yang mungkin bisa dilakukan. Pertama adalah mengganti metode optimasi yang digunakan. Alih-alih menggunakan *batch gradient descent* yang sederhana, mencoba metode optimasi lain yang lebih canggih seperti *Adam* atau *RMSProp* mungkin akan membantu model mencapai hasil yang lebih cepat dan stabil. Selain itu, melakukan penyesuaian atau tuning pada parameter-parameter penting seperti jumlah neuron di setiap layer, jumlah hidden layer, learning rate, dan ukuran batch mungkin akan sangat bermanfaat. Teknik-teknik ini mungkin akan membantu model untuk lebih baik dalam menangkap pola dari data tanpa berlebihan atau kurang memadai. Ide yang lain adalah menambahkan jenis fungsi aktivasi yang lebih variatif, seperti Leaky ReLU atau ELU. Fungsi aktivasi yang lebih kompleks ini mungkin dapat membantu model mengatasi beberapa masalah yang lebih rumit dalam data, sehingga meningkatkan kemampuan prediksi model. Dengan menggabungkan beberapa pendekatan ini, kinerja model ANN bisa saja menjadi lebih baik, efektif, dan efisien.