1. Artificial Neural Network (ANN) adalah model komputasi yang terinspirasi dari otak manusia, terdiri dari unit pemrosesan kecil yang saling terhubung bernama neuron yang tersusun dalam lapisan-lapisan (layers). Setiap koneksi antar neuron memiliki bobot yang akan disesuaikan selama proses pelatihan melalui algoritma seperti backpropagation. Tujuannya adalah untuk meminimalkan loss antara prediksi dan data sebenarnya, sehingga memungkinkan jaringan untuk belajar dan mengenali pola yang sangat kompleks dan non-linear. Komponen-komponen yang ada
    a. Activation Functions (Activation classes)
        i. Linear: Identity function (tidak melakukan transformasi)
        ii. Sigmoid: Transformasi non-linear untuk output probability (0-1)
        iii. ReLU: Rectified Linear Unit untuk menghindari vanishing gradient
        iv. Tanh: Hyperbolic tangent untuk output range (-1, 1)
        v. LeakyReLU: Variasi ReLU yang menghindari "dying neurons"
        vi. Softmax: Untuk multi-class classification probabilities
    b. Layer Types (DenseLayer)
        i. Fully connected layer dengan berbagai weight initialization
        ii. Support regularisasi L1 dan L2
        iii. Automatic forward dan backward propagation
    c. Loss Functions (Loss classes)
        i. MeanSquaredError: Untuk regression tasks
        ii. CrossEntropyLoss: Untuk classification tasks
    d. Optimization Algorithms (Optimizer classes)
        i. SGD: Stochastic Gradient Descent dengan learning rate decay
        ii. Adagrad: Adaptive gradient algorithm
        iii. Adam: Adaptive Moment Estimation (paling populer)
    e. Main Network Class (myANN)
        i. Manajemen multiple layers
        ii. Training loop dengan batch processing
        iii. Validation monitoring
        iv. Prediction functionality

    Cara kerja untuk setiap epoch
    a. Data Shuffling: Urutan data training diacak untuk memastikan model tidak belajar dari urutan data yang monoton

b. Pembagian Batch: Data yang sudah diacak dibagi menjadi beberapa kelompok kecil yang disebut batch
c. Untuk setiap batch
    i. Forward pass
        1. Satu batch data (X_batch) dimasukkan ke layer pertama
        2. Setiap DenseLayer menerima input, mengalikannya dengan weights, menambahkan biases, lalu melewatkannya ke Activation (misalnya ReLU atau Sigmoid) untuk menghasilkan output
        3. Output dari satu layer menjadi input untuk layer berikutnya, proses ini berlanjut hingga layer terakhir, menghasilkan y_pred
    ii. Menghitung loss
        1. y_pred dari forward pass dibandingkan dengan label sebenarnya (y_batch) menggunakan Fungsi Loss yang dipilih (misalnya Cross Entropy Loss)
        2. Loss dari regularisasi (jika ada) juga dihitung dan ditambahkan untuk mendapatkan total_loss
    iii. Backward Pass (Backpropagation)
        1. Proses dimulai dari loss. Kita menghitung turunan dari loss terhadap y_pred (ini adalah gradien awal).
        2. Gradien ini kemudian mundur melalui jaringan, dari layer terakhir ke layer pertama
        3. Setiap DenseLayer menerima gradien dari layer di depannya, lalu menggunakan chain rule (aturan rantai) untuk menghitung tiga hal
            a. dweights: Gradien untuk bobotnya
            b. dbiases: Gradien untuk biasnya
            c. dinputs: Gradien yang akan dilewatkan ke layer sebelumnya
    iv. Update Bobot
        1. Setelah semua gradien (dweights dan dbiases) dihitung untuk setiap layer, Optimizer dipanggil.
        2. Optimizer (misalnya Adam atau SGD) akan menggunakan gradien ini untuk memperbarui weights dan biases di setiap DenseLayer. Tujuannya adalah untuk menggeser bobot ke arah yang akan mengurangi loss.

2. Berdasarkan hasil evaluasi, model dari Sklearn memiliki nilai Recall 0.61 dan F1 score 0.62, sementara untuk model yang saya buat sendiri mendapatkan nilai Recall 0.69 dan F1 score 0.71. Hal ini menunjukan performa model yang saya buat lebih baik karena menggunakan regularisasi dan juga learning rate yang lebih kecil sehingga model dapat memiliki pemahaman yang lebih baik pada data.
3. Beberapa improvement yang dapat dilakukan
    a. Menambahkan early stopping untuk menghentikan proses training secara otomatis ketika performa model pada data validasi tidak lagi membaik
    b. Menerapkan Learning Rate Scheduling untuk meningkatkan kecepatan dan stabilitas training dengan cara mengubah learning rate secara dinamis saat training kepada setiap optimizer

Bonus

1. Convolution Layer dan Max Pooling Layer
    a. Convolution Layer: Ini adalah komponen inti dari sebuah CNN, yang bertugas sebagai pengekstrak fitur dari gambar. Layer ini bekerja dengan cara menggeser sebuah "jendela" kecil yang disebut kernel atau filter ke seluruh area gambar. Di setiap posisi, kernel (yang berisi bobot yang dapat dipelajari) melakukan operasi perkalian dot dengan bagian gambar yang ditutupinya untuk menghasilkan satu nilai. Hasil dari proses ini adalah sebuah feature map, yang merupakan representasi gambar yang telah disaring untuk menonjolkan fitur tertentu, seperti tepi, sudut, atau tekstur.
    b. Max Pooling Layer: Ini adalah layer yang berfungsi untuk mereduksi dimensi atau ukuran dari feature map sambil tetap mempertahankan informasi yang paling penting. Layer ini bekerja dengan cara mengambil jendela kecil (misalnya, 2x2 piksel) dan menggesernya di sepanjang feature map. Dari setiap jendela, ia hanya akan mengambil nilai piksel maksimum dan membuang sisanya. Proses ini membuat representasi fitur menjadi lebih ringkas dan memberikan ketahanan terhadap pergeseran kecil (translation invariance) pada gambar.
2. Keunggulan CNN Dibandingkan Fully Connected
    a. Mempertahankan Informasi Spasial: FC Layer memperlakukan gambar sebagai vektor datar, sehingga semua informasi tentang posisi piksel dan hubungan spasial antar piksel (mana yang di atas, bawah, kiri, atau kanan) menjadi hilang. Sebaliknya, convolution layer bekerja langsung pada grid 2D gambar, sehingga

secara alami mempertahankan dan memanfaatkan hubungan spasial antar piksel untuk mengenali struktur seperti bentuk dan objek.

b. Berbagi Parameter: Dalam FC Layer, setiap neuron terhubung ke setiap neuron di layer sebelumnya, menghasilkan jumlah parameter (bobot) yang sangat besar. Pada convolution layer, satu kernel yang sama (dengan set bobot yang sama) digunakan berulang kali di seluruh gambar untuk mendeteksi fitur yang sama di lokasi yang berbeda (misalnya, mendeteksi mata di kiri dan kanan wajah). Ini secara drastis mengurangi jumlah parameter, membuat model lebih efisien dan tidak rentan terhadap overfitting.
c. Invariansi Translasi: Berkat parameter sharing dan operasi pooling, CNN secara alami menjadi tahan terhadap pergeseran posisi objek. Artinya, model dapat mengenali sebuah objek (misalnya, kucing) tidak peduli apakah objek tersebut berada di pojok kiri atas atau di tengah bawah gambar. FC Layer tidak memiliki kemampuan ini. Jika posisi objek sedikit bergeser, ia akan melihatnya sebagai input yang sama sekali baru dan berbeda.