LAPORAN TUGAS BESAR 1

IF3270 Pembelajaran Mesin
Feedforward Neural Network

Oleh:

Chelvadinda 13522154

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
2024/2025

DAFTAR ISI

DAFTAR ISI....2
BAB I DESKRIPSI PERSOALAN....3
BAB II PEMBAHASAN....4
A. Deskripsi Kelas....4
B. Forward Propagation....5
C. Backward Propagation dan Weight Update....5
BAB III HASIL PENGUJIAN....7
A. Pengaruh depth dan width....7
B. Pengaruh fungsi aktivasi....7
C. Pengaruh learning rate....8
D. Pengaruh Regularisasi....8
E. Perbandingan dengan library sklearn....8
BAB IV KESIMPULAN DAN SARAN....10
A. Kesimpulan....10
B. Saran....10
PEMBAGIAN TUGAS....11
REFERENSI....12

BAB I
DESKRIPSI PERSOALAN

Pada Tugas Besar 1 mata kuliah IF3270 Pembelajaran Mesin ini, mahasiswa diminta untuk mengimplementasikan Feedforward Neural Network (FFNN) menggunakan bahasa pemrograman Python. Implementasi ini bertujuan agar mahasiswa memahami secara mendalam bagaimana implementasi FFNN mulai dari inisialisasi bobot, forward propagation, backward propagation, dan update bobot.

FFNN yang dikembangkan harus dapat menerima jumlah neuron dan fungsi aktivasi untuk setiap layer. Fungsi aktivasi yang wajib diimplementasikan meliputi Linear, ReLU, Sigmoid, Tanh, dan Softmax. Selain itu, model juga harus mendukung penggunaan beberapa fungsi loss seperti Mean Squared Error (MSE), Binary Cross-Entropy, dan Categorical Cross-Entropy. Model harus mampu menangani beberapa metode inisialisasi bobot seperti Zero Initialization, Random Uniform, dan Random Normal. Untuk memperbaiki performa model, regularisasi L1 dan L2 juga perlu ditambahkan.

Perlu dilakukan pengujian untuk menganalisis:

1. Pengaruh variasi depth dan width
2. Pengaruh fungsi aktivasi pada performa jaringan,
3. Pengaruh learning rate terhadap konvergensi training,
4. Pengaruh metode inisialisasi bobot,
5. Pengaruh regularisasi terhadap overfitting.

Sebagai bagian dari evaluasi, hasil FFNN dibandingkan dengan hasil model MLPClassifier dari library scikit-learn pada dataset MNIST.

BAB II
PEMBAHASAN

A. Deskripsi Kelas

Feedforward Neural Network (FFNN) diimplementasikan dalam kelas bernama FFNN menggunakan library dasar Python dan NumPy. Kelas FFNN bertanggung jawab untuk menginisialisasi jaringan, melakukan proses forward propagation, backward propagation, serta menyimpan bobot.

1. Atribut Kelas

Atribut	**Tipe**	**Deskripsi**
layer_sizes	list of int	Jumlah neuron di setiap layer
activations	list of str	Daftar fungsi aktivasi
loss_function	str	Jenis fungsi loss
init_method	str	Metode inisialisasi bobot (zero, uniform, normal)
regularization	str	Jenis regularisasi (l1, l2, none)
reg_lambda	float	Koefisien regularisasi
weights	List of ndarray	Daftar bobot setiap layer
biases	List of ndarray	Daftar bias setiap layer

2. Method Kelas

Method	**Deskripsi**
__init__()	Konstruktor inisialisasi atribut dan bobot
init_weights()	Menginisialisasi bobot dan bias berdasarkan metode yang dipilih
activation(x, func)	Memberikan output dari fungsi aktivasi sesuai nama
activation_derivative(x, func)	Menghitung turunan dari fungsi aktivasi yang digunakan

forward(X)	Melakukan forward propagation dan menghitung output akhir
compute_loss(y_true, y_pred)	Menghitung nilai loss antara target dan prediksi
compute_loss_derivative(y_true, y_pred)	Menghitung turunan fungsi loss terhadap prediksi
backward(X, y)	Melakukan backward propagation untuk menghitung gradien bobot dan bias
save(path)	Menyimpan bobot dan bias
load(path)	Memuat bobot dan bias

B. Forward Propagation

Forward propagation adalah proses untuk menghitung output model berdasarkan input yang diberikan. Langkah-langkah forward propagation:

1. Input dikalikan dengan bobot dan ditambahkan dengan bias
2. Hasil dari langkah pertama dilewatkan ke fungsi aktivasi (ReLU, sigmoid, tanh, linear, softmax).
3. Output dari satu layer menjadi input ke layer berikutnya.
4. Proses ini berulang hingga mencapai output layer.

Persamaan matematis yang digunakan:

$$z^{(l)} = a^{(l-1)} \times W^{(l)} + b^{(l)}$$
$$a^{(l)} = activation(z^{(l)})$$

Dimana:

- a(l)a^{(l)}a(l) adalah output dari layer ke-l
- W(l)W^{(l)}W(l) adalah bobot layer ke-l
- b(l)b^{(l)}b(l) adalah bias layer ke-l

C. Backward Propagation dan Weight Update

Backward propagation adalah proses untuk menghitung gradien dari fungsi loss terhadap bobot dan bias, sehingga model bisa memperbaiki bobotnya melalui proses training. Langkah-langkah backward propagation:

1. Hitung turunan loss terhadap output menggunakan compute_loss_derivative().
2. Hitung turunan aktivasi menggunakan activation_derivative().
3. Gunakan chain rule untuk menghitung gradien bobot dan bias di setiap layer secara berurutan dari layer output ke layer input.
4. Simpan gradien bobot (grads_w) dan gradien bias (grads_b).

Persamaan matematis untuk backward propagation:

$$\delta^{(l)} = \frac{\delta loss}{\delta a^{(l)}} \times activation\_derivative(z^{(l)})$$

$$grads\_w^{(l)} = (a^{(l-1)})^T \times \delta^{(l)})$$

$$grads\_b^{(l)} = \sum \delta^{(l)}$$

Weight Update

Setelah gradien dihitung, bobot dan bias diperbarui menggunakan gradient descent dengan formula:

- Tanpa regularisasi:

  $$W = W - learning\,rate \times grads\_w$$

- Dengan L1 Regularization:

  $$W = W - learning\,rate \times (grads\_w + \lambda sign(W))$$

- Dengan L2 Regularization:

  $$W = W - learning\,rate \times (grads\_w + \lambda W)$$

BAB III
HASIL PENGUJIAN

A. Pengaruh depth dan width

Model	Struktur	Train Loss Akhir	Val Loss Akhir
Model 1	[784, 64, 10]	0.0259	0.0348
Model 2	[784, 128, 64, 10]	0.0509	0.0585
Model 3	[784, 256, 128, 64, 10]	0.0900	0.0900

Pada pengujian ini, saya menguji pengaruh depth dan width terhadap performa. Model 1 yang strukturnya paling sederhana, mendapatkan train loss dan val loss yang paling rendah. Ini menunjukkan bahwa model sederhana sudah cukup bagus untuk belajar dari data dengan ukuran kecil. Ketika saya menambah satu hidden layer lagi di Model 2, train loss dan validation loss justru naik dibandingkan Model 1. Ini terjadi karena modelnya lebih kompleks, sehingga butuh data lebih banyak untuk bisa belajar dengan optimal.

Karena data yang saya gunakan hanya 5000, model malah sedikit kesulitan. Di Model 3, saya buat model yang jauh lebih dalam dan lebih lebar. Hasilnya, loss-nya malah jadi paling tinggi. Ini menunjukkan bahwa model terlalu kompleks untuk dataset kecil ini, dan akhirnya malah overfitting. Kalau secara teori, sebenarnya semakin banyak hidden layer (depth) atau semakin banyak neuron (width), kapasitas model untuk belajar pola yang kompleks memang lebih besar. Tapi, model juga jadi lebih susah dilatih dan lebih gampang overfitting kalau datanya tidak cukup banyak.

B. Pengaruh fungsi aktivasi

Aktivasi	Train Loss Akhir	Val Loss Akhir
ReLU	0.0259	0.0348
Sigmoid	0.0594	0.0606
Tanh	0.0475	0.0572
Linear	0.0192	0.0285

Berdasarkan hasil pengujian, fungsi aktivasi Linear justru menghasilkan train loss dan val loss paling rendah. Setelah itu diikuti oleh ReLU, Tanh, dan terakhir Sigmoid. Linear sendiri sebenarnya bukan fungsi aktivasi "aktif" karena output-nya sama aja dengan input (alias nggak ada aktivasi).

Hasil Linear yang bagus di eksperimen ini mungkin karena model yang saya buat masih cukup sederhana, jadi linearitas saja sudah cukup untuk menyelesaikan masalah. Kalau dilihat dari teori, ReLU biasanya jadi pilihan utama di deep learning karena efektif dan cepat. Di pengujian ini, ReLU juga terbukti performanya cukup bagus. Sedangkan fungsi aktivasi klasik seperti Sigmoid dan Tanh performanya kurang dibanding ReLU.

C. Pengaruh *learning rate*

Learning Rate	Train Loss Akhir	Val Loss Akhir
0.1	0.0119	0.0242
0.01	0.0192	0.0285
0.001	0.0568	0.0582

Berdasarkan hasil pengujian, learning rate 0.1 dan 0.01 memberikan hasil yang lebih baik dibandingkan 0.001. Learning rate terlalu kecil menyebabkan training menjadi lambat dan *loss* lebih tinggi.

D. Pengaruh Regularisasi

Regularisasi	Train Loss Akhir	Val Loss Akhir
none	0.0259	0.0348
L1	0.0258	0.0343
L2	0.0249	0.0337

Berdasarkan hasil eksperimen, penggunaan regularisasi L1 dan L2 mampu menurunkan nilai *train loss* dan *validation loss* dibandingkan tanpa regularisasi. L2 *Regularization* memberikan performa terbaik dengan *train loss* dan *validation loss* paling kecil, menunjukkan bahwa regularisasi L2 lebih efektif dalam mencegah *overfitting* pada model ini.

E. Perbandingan dengan library sklearn

Model	Train Acc	Val Acc
FFNN	0.8482	0.8030
Sklearn	0.9932	0.9330

Sklearn memiliki akurasi lebih tinggi karena menggunakan optimisasi dan teknik yang lebih kompleks dibanding dengan ffnn yang telah saya implementasikan.

BAB IV
KESIMPULAN DAN SARAN

A. Kesimpulan

Pada tugas besar ini, telah diimplementasikan sebuah model Feedforward Neural Network (FFNN) dari awal menggunakan Python dan NumPy tanpa bantuan library deep learning modern. Model yang dibangun dapat menggunakan berbagai fungsi aktivasi seperti ReLU, Sigmoid, Tanh, Linear, dan Softmax, serta berbagai metode inisialisasi bobot seperti zero, uniform, dan normal. Selain itu, model juga mampu melakukan regularisasi L1 dan L2.

Dalam pengujian, FFNN dilatih pada subset dataset MNIST sebanyak 5000 data dan mampu mencapai akurasi training sekitar 84% dan akurasi validasi sekitar 80%.
Hasil ini menunjukkan bahwa model yang dibangun mampu belajar dengan baik dari data sederhana, walaupun performanya masih di bawah model yang dilatih menggunakan Scikit-Learn MLPClassifier yang mencapai akurasi validasi di atas 93%.

Seluruh proses training, forward propagation, backward propagation, hingga update bobot dilakukan secara manual. Eksperimen yang dilakukan meliputi variasi depth, width, fungsi aktivasi, learning rate, dan regularisasi, menunjukkan bahwa model sederhana dengan fungsi aktivasi ReLU dan regularisasi L2 cenderung memberikan hasil yang paling baik.

B. Saran

Masih terdapat beberapa bagian dari spesifikasi yang belum diimplementasikan. Salah satunya adalah visualisasi struktur jaringan saraf. Selain itu, visualisasi distribusi bobot dan gradien juga belum ditambahkan. Visualisasi ini penting untuk memahami bagaimana bobot dan gradien tersebar selama proses training. Terakhir, visualisasi learning curve berupa grafik perubahan loss terhadap epoch juga perlu ditambahkan. Grafik ini akan memudahkan dalam menganalisis konvergensi model selama training dan membantu dalam memilih hyperparameter yang lebih optimal.

PEMBAGIAN TUGAS

Nama / NIM	Tugas
Chelvadinda / 13522154	ALL

REFERENSI

[1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
[2] Nielsen, M. (2015). Neural networks and deep learning. Determination Press. Retrieved from
[3] Haykin, S. (1998). Neural networks: A comprehensive foundation (2nd ed.). Prentice Hall.