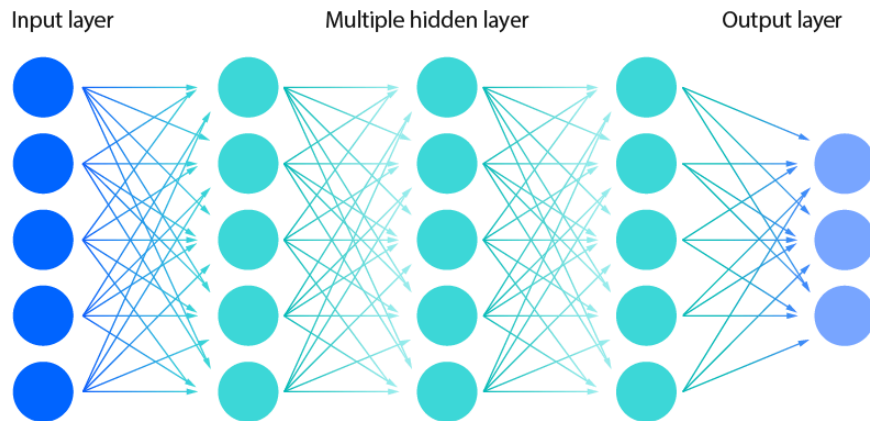


Tugas Besar 1 IF3270 Pembelajaran Mesin Feedforward Neural Network

Dipersiapkan Oleh Tim Asisten IF3270 2024/2025

Versi: 1.0 06/03/2025

Versi: 1.1 07/03/2025



Deadline:

Jumat, 28 Maret 2025 23.59 WIB

Tujuan

Tugas Besar I pada kuliah IF3270 Pembelajaran Mesin agar peserta kuliah mendapatkan wawasan tentang bagaimana cara mengimplementasikan Feedforward Neural Network (FFNN). Pada tugas ini, peserta kuliah akan ditugaskan untuk mengimplementasikan FFNN *from scratch*.

Spesifikasi

Implementasikan suatu modul FFNN yang memenuhi ketentuan-ketentuan berikut:

- FFNN yang diimplementasikan dapat **menerima jumlah neuron dari tiap layer** (termasuk input layer dan output layer)
- FFNN yang diimplementasikan dapat **menerima fungsi aktivasi dari tiap layer**. Pilihan fungsi aktivasi yang harus diimplementasikan adalah sebagai berikut:

Nama Fungsi Aktivasi	Definisi Fungsi
Linear	$Linear(x) = x$
ReLU	$ReLU(x) = \max(0, x)$
Sigmoid	$\sigma(x) = \frac{1}{1 + e^{-x}}$
Hyperbolic Tangent (tanh)	$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
Softmax	Untuk vector $\vec{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$, $softmax(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$

- FFNN yang diimplementasikan dapat **menerima fungsi loss** dari model tersebut. Pilihan loss function yang harus diimplementasikan adalah sebagai berikut:

Nama Fungsi Loss	Definisi Fungsi
MSE	$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$

Binary Cross-Entropy	$\mathcal{L}_{BCE} = -\frac{1}{n} \sum_{i=1}^n (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i))$ <p> y_i = Actual binary label (0 or 1) \hat{y}_i = Predicted value of y_i n = Batch size </p>
Categorical Cross-Entropy	$\mathcal{L}_{CCE} = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^C (y_{ij} \log \hat{y}_{ij})$ <p> y_{ij} = Actual value of instance i for class j \hat{y}_{ij} = Predicted value of y_{ij} C = Number of classes n = Batch size </p>

- Catatan: Binary cross-entropy merupakan kasus khusus categorical cross-entropy dengan kelas sebanyak 2
- Terdapat mekanisme untuk **inisialisasi bobot** tiap neuron (termasuk bias). Pilihan metode inisialisasi bobot yang harus diimplementasikan adalah sebagai berikut:
 - **Zero initialization**
 - Random dengan distribusi **uniform**.
 - Menerima parameter lower bound (batas minimal) dan upper bound (batas maksimal)
 - Menerima parameter seed untuk reproducibility
 - Random dengan distribusi **normal**.
 - Menerima parameter mean dan variance
 - Menerima parameter seed untuk reproducibility
- Instance model yang diinisialisasikan harus bisa **menyimpan bobot** tiap neuron (termasuk bias)
- Instance model yang diinisialisasikan harus bisa **menyimpan gradien bobot** tiap neuron (termasuk bias)
- Instance model memiliki method untuk **menampilkan model** berupa **struktur jaringan** beserta **bobot** dan **gradien bobot** tiap neuron. (Format dibebaskan)
- Instance model memiliki method untuk **menampilkan distribusi bobot** dari tiap layer.

- Menerima masukan berupa list of integer (bisa disesuaikan ke struktur data lain sesuai kebutuhan) yang menyatakan layer mana saja yang distribusinya akan di-plot
- Instance model memiliki method untuk **menampilkan distribusi gradien bobot** dari tiap layer.
 - Menerima masukan berupa list of integer (bisa disesuaikan ke struktur data lain sesuai kebutuhan) yang menyatakan layer mana saja yang distribusinya akan di-plot
- Instance model memiliki method untuk **save** dan **load**
- Model memiliki implementasi **forward propagation** dengan ketentuan sebagai berikut:
 - Dapat menerima input berupa **batch**.
- Model memiliki implementasi **backward propagation** untuk menghitung perubahan gradien:
 - Dapat menangani perhitungan perubahan gradien untuk input data **batch**.
 - Gunakan konsep **chain rule** untuk menghitung gradien tiap bobot terhadap loss function.
 - Berikut merupakan **turunan pertama** untuk setiap fungsi aktivasi:

Nama Fungsi Aktivasi	Turunan Pertama
Linear	$\frac{d(\text{Linear}(x))}{dx} = 1$
ReLU	$\frac{d(\text{ReLU}(x))}{dx} = \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$
Sigmoid	$\frac{d(\sigma(x))}{dx} = \sigma(x)(1 - \sigma(x))$
Hyperbolic Tangent (tanh)	$\frac{d(\tanh(x))}{dx} = \left(\frac{2}{e^x - e^{-x}} \right)^2$

Softmax	<p>Untuk vector $\vec{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$,</p> $\frac{d(\text{softmax}(\vec{x}))}{d\vec{x}} = \begin{bmatrix} \frac{\partial(\text{softmax}(\vec{x}))_1}{\partial x_1} & \dots & \frac{\partial(\text{softmax}(\vec{x}))_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial(\text{softmax}(\vec{x}))_n}{\partial x_1} & \dots & \frac{\partial(\text{softmax}(\vec{x}))_n}{\partial x_n} \end{bmatrix}$ <p>Dimana untuk $i, j \in \{1, \dots, n\}$,</p> $\frac{\partial(\text{softmax}(\vec{x}))_i}{\partial x_j} = \text{softmax}(\vec{x})_i (\delta_{i,j} - \text{softmax}(\vec{x})_j)$ $\delta_{i,j} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$
---------	---

- Model memiliki implementasi **weight update** dengan menggunakan **gradient descent** untuk memperbarui bobot berdasarkan gradien yang telah dihitung, berikut persamaannya:

$$W_{new} = W_{old} - \alpha \left(\frac{\partial \mathcal{L}}{\partial W_{old}} \right)$$

α = Learning rate

- Implementasi untuk pelatihan model harus memenuhi ketentuan berikut:
 - Dapat menerima parameter berikut:
 - Batch size
 - Learning rate
 - Jumlah epoch
 - Verbose
 - Verbose 0 berarti tidak menampilkan apa-apa selama pelatihan
 - Verbose 1 berarti hanya menampilkan progress bar beserta dengan kondisi training loss dan validation loss saat itu
 - Proses pelatihan mengembalikan **histori dari proses pelatihan** yang berisi **training loss** dan **validation loss tiap epoch**.
- Lakukan **pengujian** terhadap implementasi FFNN dengan ketentuan sebagai berikut:
 - Analisis pengaruh beberapa hyperparameter sebagai berikut:
 - Pengaruh **depth (banyak layer)** dan **width (banyak neuron per layer)**
 - Pilih 3 variasi kombinasi width (depth tetap) dan 3 variasi depth (width semua layer tetap)


- Bandingkan hasil akhir prediksinya
- Bandingkan grafik loss pelatihannya
- Pengaruh **fungsi aktivasi** hidden layer
 - Lakukan untuk setiap fungsi aktivasi yang diimplementasikan **kecuali softmax**.
 - Bandingkan hasil akhir prediksinya
 - Bandingkan grafik loss pelatihannya
 - Bandingkan distribusi bobot dan gradien bobot dari beberapa/semua layer pada model
- Pengaruh **learning rate**
 - Lakukan 3 variasi learning rate (nilainya dibebaskan)
 - Bandingkan hasil akhir prediksinya
 - Bandingkan grafik loss pelatihannya
 - Bandingkan distribusi bobot dan gradien bobot dari beberapa/semua layer pada model
- Pengaruh **inisialisasi bobot**
 - Lakukan untuk setiap metode inisialisasi bobot yang diimplementasikan
 - Bandingkan hasil akhir prediksinya
 - Bandingkan grafik loss pelatihannya
 - Bandingkan distribusi bobot dan gradien bobot dari beberapa/semua layer pada model
- Analisis perbandingan hasil prediksi dengan [library sklearn MLP](#)
 - Lakukan satu kali pelatihan dengan hyperparameter yang sama untuk kedua model
 - Hyperparameter yang digunakan dibebaskan
 - Bandingkan hasil akhir prediksinya saja
- Gunakan dataset berikut untuk menguji model: [mnist_784](#)
 - Gunakan method [fetch_openml](#) dari sklearn untuk memuat dataset
 - Berikut contoh untuk memuat dataset: [Contoh](#)
- Akan ada beberapa **test case** yang akan diberikan oleh tim asisten (menyusul).
- Pengujian dilakukan di file .ipynb terpisah

Spesifikasi Bonus

Berikut merupakan beberapa spesifikasi bonus yang dapat Anda kerjakan:

- Implementasikan FFNN dengan menggunakan **automatic differentiation**.

- Metode ini merupakan metode yang umum digunakan untuk perhitungan gradien pada library-library untuk deep learning seperti PyTorch atau TensorFlow.
- Jika ingin mengimplementasikan bonus ini, sangat disarankan untuk mengimplementasikan dari sejak awal mengerjakan tugas supaya tidak terlalu banyak perubahan dari implementasi biasa.
- Referensi video:

 The spelled-out intro to neural networks and backpropagation: building mi...
- Implementasikan minimal 2 fungsi aktivasi lain yang sering digunakan.
- Implementasikan 2 metode inisialisasi bobot berikut:
 - Xavier initialization
 - He initialization
- Implementasikan metode regularisasi L1 dan L2, kemudian lakukan analisis untuk beberapa hal berikut:
 - Lakukan eksperimen masing-masing 1 kali untuk model tanpa regularisasi, dengan regularisasi L1, dan dengan regularisasi L2.
 - Bandingkan hasil akhir prediksinya
 - Bandingkan grafik loss pelatihannya
 - Bandingkan distribusi bobot dan gradien bobot dari beberapa/semua layer pada model
- Implementasikan metode normalisasi RMSNorm, kemudian lakukan analisis untuk beberapa hal berikut:
 - Lakukan eksperimen masing-masing 1 kali untuk model tanpa normalisasi dan dengan normalisasi.
 - Bandingkan hasil akhir prediksinya
 - Bandingkan grafik loss pelatihannya
 - Bandingkan distribusi bobot dan gradien bobot dari beberapa/semua layer pada model

Kelompok

Pembagian kelompok ditentukan sendiri oleh mahasiswa dengan mengisi [sheets kelompok IF3270 Pembelajaran Mesin](#) berikut ini dengan 1 kelompok terdiri dari **3 orang** dan **tidak lintas kelas**. Batas waktu pengisian kelompok adalah **Minggu, 9 Maret 2025 pukul 23:59 WIB**. Setelah waktu yang ditentukan, mahasiswa yang belum mengisi sheets kelompok akan diacak.

QnA

Pertanyaan dapat ditanyakan pada [link QnA](#) berikut. Pastikan pertanyaan yang ditanyakan tidak berulang.

Aturan

Terdapat beberapa hal yang harus diperhatikan dalam pengerjaan tugas ini, yakni:


1. Jika terdapat hal yang tidak dimengerti, silahkan ajukan pertanyaan kepada asisten melalui **link QnA** yang telah diberikan di atas. Pertanyaan yang diajukan secara personal ke asisten **tidak akan dijawab** untuk menghindari perbedaan informasi yang didapatkan oleh peserta kuliah.
2. Dilarang melakukan **plagiarisme, menggunakan AI dalam bentuk apapun secara tidak bertanggungjawab, dan melakukan kerjasama antar kelompok**. Pelanggaran pada poin ini akan menyebabkan pemberian **nilai E** pada setiap anggota kelompok.
3. Tugas diimplementasikan dalam bahasa **Python**.
4. Implementasi FFNN *from scratch* hanya boleh menggunakan library untuk perhitungan matematika (Contoh: NumPy, dll).

Deliverables

- Tugas dikumpulkan dalam bentuk link ke *repository* GitHub yang **minimal** berisi beberapa hal berikut (boleh ditambahkan jika dirasa perlu):
 - Folder **src**, digunakan untuk menyimpan source code implementasi FFNN beserta dengan notebook pengujian.
 - Folder **doc**, digunakan untuk menyimpan laporan dalam bentuk **.pdf** yang terdiri atas komponen berikut:
 - Cover
 - Deskripsi Persoalan
 - Pembahasan
 - Penjelasan implementasi
 - Deskripsi kelas beserta deskripsi atribut dan methodnya
 - Penjelasan forward propagation
 - Penjelasan backward propagation dan weight update
 - Hasil pengujian
 - Pengaruh depth dan width
 - Pengaruh fungsi aktivasi
 - Pengaruh learning rate
 - Pengaruh inisialisasi bobot

- Pengaruh regularisasi (jika mengerjakan)
 - Pengaruh normalisasi RMSNorm (jika mengerjakan)
 - Perbandingan dengan library sklearn
 - Kesimpulan dan Saran
 - Pembagian tugas tiap anggota kelompok
 - Referensi
- **README.md**, yang berisi deskripsi singkat repository, cara setup dan run program, dan pembagian tugas tiap anggota kelompok.
- Pengumpulan dilakukan melalui form dengan tautan sebagai berikut:
[Form Pengumpulan IF3270 Pembelajaran Mesin](#)
- Batas akhir pengumpulan adalah hari **Jumat, 28 Maret 2025 23.59 WIB**. Tugas yang terlambat dikumpulkan tidak akan diterima.
- Pengumpulan dilakukan oleh NIM terkecil.

Referensi

-  The spelled-out intro to neural networks and backpropagation: building micrograd
- <https://www.jasonosajima.com/forwardprop>
- <https://www.jasonosajima.com/backprop>
- <https://numpy.org/doc/2.2/>
- https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
- [https://math.libretexts.org/Bookshelves/Calculus/Calculus_\(OpenStax\)/14%3A_Differentiation_of_Functions_of_Several_Variables/14.05%3A_The_Chain_Rule_for_Multivariable_Functions](https://math.libretexts.org/Bookshelves/Calculus/Calculus_(OpenStax)/14%3A_Differentiation_of_Functions_of_Several_Variables/14.05%3A_The_Chain_Rule_for_Multivariable_Functions)
- <https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/>
- <https://douglasorr.github.io/2021-11-autodiff/article.html>