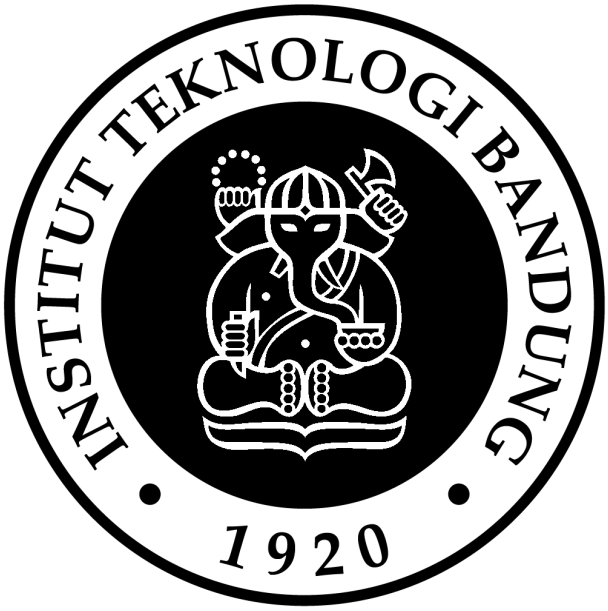
Laporan Tugas Besar 1 IF3270 Pembelajaran Mesin
Feedforward Neural Network

Semester II Tahun 2024/2025



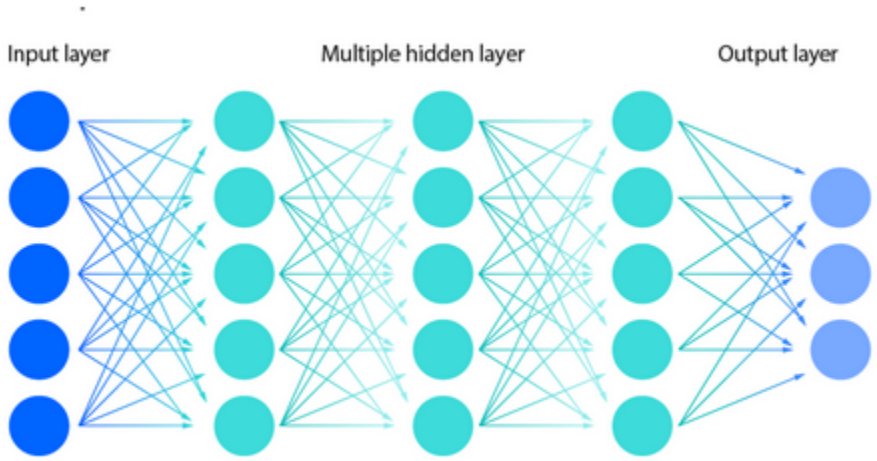
Disusun Oleh :

Maulana Muhamad Susetyo	13522127
Andi Marihot Sitorus	13522138
Muhammad Rasheed Qais Tandjung	13522158

INSTITUT TEKNOLOGI BANDUNG

2024

Bab I
Deksripsi Persoalan



Gambar 1.1 Feedforward Neural Network

1.1. Latar Belakang

Feedforward Neural Network (FFNN) merupakan salah satu arsitektur dasar dalam bidang kecerdasan buatan yang digunakan untuk pemrosesan data dan pengambilan keputusan berdasarkan pola yang telah dipelajari. FFNN banyak digunakan dalam berbagai aplikasi, seperti klasifikasi, regresi, dan pengenalan pola. Dengan berkembangnya teknologi komputasi dan kebutuhan analisis data yang semakin kompleks, implementasi FFNN menjadi semakin penting untuk menyelesaikan berbagai permasalahan di berbagai bidang, seperti kesehatan, keuangan, dan pengolahan citra.

Dalam laporan ini, akan dibahas mengenai implementasi FFNN dalam kode python, dengan tujuan memahami bagaimana model ini dapat diterapkan untuk menyelesaikan permasalahan yang dihadapi. Implementasi dilakukan dengan menggunakan dataset yang disediakan serta berbagai percobaan optimasi untuk menganalisis performa model.

1.2. Rumusan Masalah

Adapun permasalahan yang ingin dianalisis dalam implementasi ini adalah:

- Bagaimana cara kerja FFNN dalam memproses data dan menghasilkan prediksi?
- Bagaimana pengaruh arsitektur neuron terhadap performa model?
- Sejauh mana optimasi parameter dapat meningkatkan akurasi model?

1.3. Landasan Teori

Feedforward Neural Network (FFNN) adalah jenis jaringan saraf tiruan yang terdiri dari lapisan-lapisan neuron yang terhubung secara satu arah, dari lapisan input ke lapisan output melalui satu atau lebih lapisan tersembunyi.

1.3.1. Struktur FFNN

FFNN umumnya terdiri dari tiga jenis lapisan:

- Lapisan Input: Menerima data mentah sebagai masukan.
- Lapisan Tersembunyi: Mengandung neuron yang menerapkan fungsi aktivasi untuk memproses data.
- Lapisan Output: Menghasilkan hasil akhir berdasarkan bobot dan bias yang telah dipelajari.

1.3.2. Fungsi Aktivasi

Fungsi aktivasi dalam FFNN digunakan untuk memberikan non-linieritas pada model, sehingga dapat menangkap pola kompleks dalam data. Beberapa fungsi aktivasi yang akan diimplementasikan antara lain:

- Linear: Tidak memberikan non-linieritas
- Sigmoid: Cocok untuk kasus probabilitas (nilai output antara 0 dan 1).
- ReLU (Rectified Linear Unit): Banyak digunakan dalam deep learning karena mampu mengatasi masalah vanishing gradient.
- Tanh: Memetakan nilai input ke rentang -1 hingga 1.
- Softmax : Mengonversi skor atau logit dari setiap kelas menjadi distribusi probabilitas yang berjumlah satu.

1.3.3. Algoritma Training

FFNN belajar dengan menggunakan algoritma backpropagation, yang terdiri dari dua tahap utama:

- Forward Propagation: Data melewati jaringan dari input ke output, menghasilkan prediksi awal.
- Backward Propagation: Kesalahan dihitung menggunakan fungsi loss, lalu bobot diperbarui melalui optimasi gradien.

1.3.4. Evaluasi Model

Performa FFNN dievaluasi dengan beberapa metrik tergantung pada jenis masalah yang diselesaikan:

- Klasifikasi: Akurasi, Precision, Recall, F1-score.
- Regresi: Mean Squared Error (MSE), Mean Absolute Error (MAE), R-squared.

Bab II
Penjelasan Implementasi

A. Deskripsi Kelas

a. *Class FFNN*

```
class FFNN:
   def __init__(self, layer_neurons, X_train, y_train, X_val,
y_val, learning_rate, activation_functions=None):
       self.layer_neurons = layer_neurons
       self.input = X_train
       self.target = y_train
       self.learning_rate = learning_rate
       self.activations = activation_functions

       if activation_functions != None and
len(activation_functions) != len(layer_neurons):
           raise ValueError("Number of activation functions
must match number of layers")

       # For weight initialization
       self.sizes = [(self.layer_neurons[i] + 1,
self.layer_neurons[i + 1]) for i in
range(len(self.layer_neurons) - 1)]

       # For validation
       self.X_val = X_val
       self.y_val = y_val
```

Kelas FFNN terdiri dari beberapa atribut, yaitu:

- layer_neurons: jumlah neuron per layer – e.g., [2,2,2] berarti layer input dengan 2 neuron, 1 hidden layer dengan 2 neuron, dan 1 layer output dengan 2 neuron.
- input: dataset train
- target: label dari dataset train
- x_val: dataset validasi
- y_val: label dari dataset validasi
- learning_rate: learning rate per step

- activation_functions: array fungsi aktivasi per layer -e.g., jika layer neuron [2,2,2] maka activation_functions berisi jenis fungsi aktivasi sebanyak jumlah layer, pada contoh ini berarti 3 fungsi aktivasi
- sizes: ukuran matriks bobot per layer, menggunakan contoh layer neuron [2,2,2] maka ukurannya [(3,2),(3,2)] karena ada neuron bias.

b. *Method Setter dan Initializer FFNN*

```
    def setLearningRate(self, learning_rate):
        self.learning_rate = learning_rate

    def setActivationUniform(self, activation_function):
        self.activations = [activation_function for i in
range(len(self.layer_neurons))]

    def setWeights(self, weights):
        self.weights = weights

    def initializeWeightZeros(self):
        self.weights = [np.zeros(size) for size in self.sizes]

    def initializeWeightRandomUniform(self, lower_bound,
upper_bound, seed=None):
        if seed != None:
            np.random.seed(seed)
        self.weights = [np.random.uniform(lower_bound,
upper_bound, size) for size in self.sizes]

    def initializeWeightRandomNormal(self, mean, variance,
seed=None):
        if seed != None:
            np.random.seed(seed)
        self.weights = [np.random.normal(mean, variance, size)
for size in self.sizes]
```

Berikut adalah penjelasan *Setter* dan *Initializer* kelas FFNN:

- setLearningRate: mengubah learning rate
- setActivationUniform: mengubah fungsi aktivasi tiap layer menjadi fungsi aktivasi yang sama

- setWeights: mengubah bobot secara manual, ukuran matriks bobot harus sesuai dengan self.sizes. Sebagai Contoh, berikut adalah matriks bobot untuk layer_neutron [2,2,2] dengan size [(3,2),(3,2)];

```
        np.array(
            [[0.35, 0.35],
            [0.15, 0.25],
            [0.20, 0.30]]),
        np.array(
            [[0.60, 0.60],
            [0.40, 0.50],
            [0.45, 0.55]]),
```

  per np.array merepresentasikan layer pada FFNN, baris dalam satu np.array merupakan bobot dari suatu node, sebagai contoh [0.35,0.35] adalah bobot dari b1 ke h1_1 dan h2_2. kolom pada satu np.array merepresentasikan tujuan neuron, sebagai contoh [0.35, 0.15, 0.2] merupakan input untuk h1_1.

  NB: baris pertama adalah *node* bias
- initializeWeightZeros: Menginisialiasi bobot menjadi 0 sesuai self.size
- initializeWeightRandomUniform: Menginisialiasi bobot antara 0-1 dengan distribusi uniform sesuai self.size
- initializeWeightRandomNormal: Menginisialiasi bobot antara 0-1 dengan distribusi normal sesuai self.size

c. *Method Forward Propagation dan Backward Propagation*

Akan dijelaskan lebih lanjut pada bagian 2B. Penjelasan *Forward Propagation* dan 2C. Penjelasan *Backward Propagation* dan *Weight Update*.

d. *Method Train*

```
    def train(self, batch_size, learning_rate, epochs,
verbose=True, printResults=False):
        self.setLearningRate(learning_rate)
```

```
        self.training_loss_list = []
        self.validation_loss_list = []

        for epoch in range(epochs):
            training_loss = 0
            validation_loss = 0

            # Iterate through batches
            for i in range(0, len(self.input), batch_size):
                batch_end = (i + batch_size) if (i +
batch_size) < len(self.input) else len(self.input)
                X_batch = self.input[i:batch_end]
                y_batch = self.target[i:batch_end]

                self.FFNNForwardPropagation(X_batch)
                self.FFNNBackPropagation(y_batch)

                training_loss += Loss.mse(y_batch,
self.layer_results[-1])

            # Calculate epoch loss
            training_loss /= len(self.input)
            self.training_loss_list.append(training_loss)

            self.FFNNForwardPropagation(self.X_val)
            validation_loss = Loss.mse(self.y_val,
self.layer_results[-1])
            self.validation_loss_list.append(validation_loss)

            # Print epoch results
            if verbose:
                progress_bar = Color.progress_bar(epoch + 1,
epochs)
                print(Color.YELLOW + f" [Epoch {epoch + 1}]:"
+ Color.GREEN + f"\tTraining Loss: {training_loss}" +
Color.BLUE + f"\tValidation Loss: {validation_loss}" +
Color.YELLOW + f'\tProgress: [{progress_bar}]' + Color.RESET)
                if printResults:
                    print(f"{Color.CYAN}
```

```
Prediction:\t{self.layer_results[-1]}")
                    print(f"{Color.MAGENTA}
Target:\t\t{self.target}{Color.RESET}")

        return self.training_loss_list,
self.validation_loss_list
```

Metode *Train* melakukan pelatihan model. Terdapat beberapa parameter yang dapat diganti, yaitu:

- batch_size: besar batch yang akan dilakukan per *propagation*
- learning_rate: besar learning_rate untuk mengubah perubahan bobot
- epochs: berapa iterasi yang akan dilakukan
- verbose: opsi untuk menunjukkan progress bar, training loss, validation loss.
- printResult: opsi untuk menunjukkan prediksi model dan membandingkannya dengan label, hanya akan muncul jika verbose=true.

Untuk tahap2 metode *Train*:

- Inisialisasi penyimpanan loss training dan validation
- melakukan iterasi sebanyak epochs
- per epoch, akan membagi data *train* menjadi batch, dan melakukan *forward & backward propagation* per batch. Kemudian menghitung loss. setelah satu epoch selesai, akan menampilkan progress, loss, dan prediksi pada epoch tersebut
- *return* training_loss_list dan validation_loss_list

e. *Method printGraph*

```
    def printGraph(self):
        G = nx.DiGraph()
        positions = {}
        node_colors = {}
        node_labels = {}

        layer_spacing = 3
        neuron_spacing = 1.5
```

```
        num_layers = len(self.layer_neurons)
        max_neurons = max(self.layer_neurons)

        fig_width = layer_spacing * num_layers * 1.2
        fig_height = max_neurons * neuron_spacing * 1.5

        for layer_idx, num_neurons in
enumerate(self.layer_neurons):
            y_start = -(num_neurons - 1) * neuron_spacing / 2

            for i in range(num_neurons):
                if layer_idx == 0:
                    node_name = f"I{i+1}"
                    color = "lightgreen"
                elif layer_idx == num_layers - 1:
                    node_name = f"O{i+1}"
                    color = "yellow"
                else:
                    node_name = f"H{layer_idx}_{i+1}"
                    color = "lightblue"

                G.add_node(node_name)
                positions[node_name] = (layer_idx *
layer_spacing, y_start + i * neuron_spacing)
                node_colors[node_name] = color
                node_labels[node_name] = f"{node_name}"

            if layer_idx < num_layers - 1:
                bias_node = f"B{layer_idx+1}"
                G.add_node(bias_node)
                positions[bias_node] = (layer_idx *
layer_spacing, y_start - neuron_spacing)
                node_colors[bias_node] = "lightgray"
                node_labels[bias_node] = f"{bias_node}"

        edge_labels = {}
        for layer_idx, weight_matrix in
enumerate(self.weights):
            for src in range(len(weight_matrix) - 1):
```

```
                for dest in range(weight_matrix.shape[1]):
                    src_node = (
                        f"I{src+1}" if layer_idx == 0 else
f"H{layer_idx}_{src+1}"
                    )
                    dest_node = (
                        f"H{layer_idx+1}_{dest+1}" if
layer_idx + 1 < num_layers - 1 else f"O{dest+1}"
                    )
                    weight = weight_matrix[src + 1, dest]
                    G.add_edge(src_node, dest_node,
weight=weight)
                    edge_labels[(src_node, dest_node)] =
f"{weight:.2f}"

            for dest in range(weight_matrix.shape[1]):
                bias_node = f"B{layer_idx+1}"
                dest_node = (
                    f"H{layer_idx+1}_{dest+1}" if layer_idx +
1 < num_layers - 1 else f"O{dest+1}"
                )
                weight = weight_matrix[0, dest]
                G.add_edge(bias_node, dest_node,
weight=weight)
                edge_labels[(bias_node, dest_node)] =
f"{weight:.2f}"

        plt.figure(figsize=(fig_width, fig_height))
        nx.draw(
            G,
            pos=positions,
            with_labels=True,
            labels=node_labels,
            node_color=[node_colors[n] for n in G.nodes()],
            edge_color="black",
            node_size=2000,
            font_size=10,
            font_weight="bold",
            arrowsize=15,
```

```
        )

        for node_name, (x, y) in positions.items():
            if node_name.startswith("I") or
node_name.startswith("B"):
                continue

            if node_name.startswith("H"):
                layer_idx = int(node_name.split("_")[0][1:])
- 1
            elif node_name.startswith("O"):
                layer_idx = len(self.layer_neurons) - 2

            neuron_idx = int(node_name.split("_")[1]) - 1 if
"_" in node_name else int(node_name[1]) - 1

            if layer_idx < len(self.deltas) and neuron_idx <
self.deltas[layer_idx].shape[1]:
                delta_value = self.deltas[layer_idx][0,
neuron_idx]
                plt.text(x, y + 0.25, f"Δ={delta_value:.5f}",
fontsize=9, ha="center", color="green")

        nx.draw_networkx_edge_labels(G, pos=positions,
edge_labels=edge_labels, font_size=8, label_pos=0.75)

        plt.title("Feedforward Neural Network Graph",
fontsize=14)
        plt.show()
```

Metode printGraph() membuat visualisasi FFNN menggunakan networkX dan Matplotlib. Pertama melakukan definisi variabel untuk menyimpan lokasi, label, warna node serta menghitung ukuran graf. Kemudian pada *for loop* pertama, memasukan data *node*. Untuk *node* bias dipisah karena lokasinya dibawah *node* lainnya.

Pada *for loop* selanjutnya, melakukan penghubungan *node* serta data bobotnya. *Inner loop* untuk *node* bias dipisah karena *node* bias tidak mempunyai '*edge* masuk'.

Terakhir adalah menaruh delta pada *node hidden* dan *output.*

f. *Method plotWeightDistribution* dan *plotGradientDistribution*

```
    def plot_weight_distribution(self, layers):

        if not hasattr(self, 'weights'):
            raise ValueError("Bobot belum diinisialisasi")

        for layer in layers:
            if layer < 0 or layer >= len(self.weights):
                raise ValueError(f"Layer {layer} di luar
jangkauan. Indeks layer harus antara 0 dan {len(self.weights)
- 1}.")

            weights = self.weights[layer].flatten()
            plt.figure(figsize=(8, 5))
            plt.hist(weights, bins=30, alpha=0.7, color='b',
edgecolor='black')
            plt.xlabel("Nilai Bobot")
            plt.ylabel("Frekuensi")
            plt.title(f"Distribusi Bobot - Layer {layer}")
            plt.grid(axis='y', linestyle='--', alpha=0.7)
            plt.show()

    def plot_gradient_distribution(self, layers):

        if not hasattr(self, 'deltas'):
            raise ValueError("Gradien belum tersedia")

        for layer in layers:
            if layer < 0 or layer >= len(self.deltas):
                raise ValueError(f"Layer {layer} di luar
jangkauan. Indeks layer harus antara 0 dan {len(self.deltas)
- 1}.")

            gradients = self.deltas[layer].flatten()
            plt.figure(figsize=(8, 5))
```

```
            plt.hist(gradients, bins=30, alpha=0.7, color='r',
edgecolor='black')
            plt.xlabel("Nilai Gradien Bobot")
            plt.ylabel("Frekuensi")
            plt.title(f"Distribusi Gradien Bobot - Layer
{layer}")
            plt.grid(axis='y', linestyle='--', alpha=0.7)
            plt.show()

    # usage example : model.plot_gradient_distribution([0, 1])
```

Metode plot_weight_distribution dan plot_gradient_distribution adalah metode yang mem-visualisasikan distribusi bobot dan gradien bobot dari model. Tiap metode akan menampilkan histogram bobot / gradien bobot tiap layer serta mengecek apakah layer ada. input kedua metode adalah list of layers -e.g., [0,1,3,5] maka akan menampilkan bobot / gradien bobot pada layer pertama, kedua, keempat dan keenam.

g. *Method save dan load*

```
    def save(self, filename):
        try:
            with open(filename, "wb") as f:
                pickle.dump(self, f)
            print(f" File Successfuly Saved to: '{filename}'")
        except Exception as e:
            print(f"Error: {e}")

    @classmethod
    def load(cls, filename):
        if not os.path.exists(filename):
            print(f"Error: File '{filename}' doesn't exist.")
            return None

        try:
            with open(filename, "rb") as f:
                obj = pickle.load(f)
            if isinstance(obj, cls):
                print(f"Successfuly loaded '{filename}'")
```

```
                return obj
            else:
                print(f"Error: Loaded object is not an
instance of {cls.__name__}")
                return None
        except Exception as e:
            print(f"Error: {e}")
            return None
```

Metode *save* menyimpan model pada sebuah file dengan *extension* .pkl menggunakan library pickle. Metode save terlebih dahulu membuat sebuah file dan melakukan *dump* kelas FFNN yang berisikan informasi mengenai model.

Metode load, sesuai namanya, memuat model dari file .pkl. Metode akan memastikan apakah file ada, apakah file *pickle* tersebut merupakan *instance* kelas FFNN, serta *miscellaneous error* lainnya.

B. Penjelasan *Forward Propagation*

```
    def FFNNForwardPropagation(self, current_input):
        self.layer_results = [current_input]
        self.layer_results_before_activation = [current_input]
        input = Matrix.addBiasColumn(current_input)
        i = 1
        for layer in self.weights:
            # Get layer result
            initial_result = np.matmul(input, layer)

self.layer_results_before_activation.append(initial_result)

            # Apply activation function to result
            activation = self.activations[i]
            result = activation(initial_result)
            self.layer_results.append(result)

            # Change input to result
            biased_result = Matrix.addBiasColumn(result)
            input = biased_result
```

```
            i += 1

            # Return if at output layer
            if i > len(self.weights):
                return result
```

Forward propagation bekerja dengan mempropagasikan *input* dari layer pertama ke layer output jaringan. *Output* dari setiap layer, kecuali *layer* terakhir, akan menjadi input dari *layer* berikutnya. Input tersebut pertama akan dimasukkan ke fungsi aktivasi yang ditetapkan terlebih dahulu untuk setiap *layer*.

Sebuah *layer* terdiri atas N buah *node*. *Layer* tersebut akan menerima *input* dari *layer* sebelumnya. Sebutlah bahwa *layer* sebelumnya memiliki K buah *node*, dan seluruh *node* dari *layer* K akan menjadi *input* untuk seluruh *node* pada layer N. Sehingga akan terdapat K x N buah perpindahan data dari *layer* K ke *layer* N. Untuk setiap K x N perpindahan tersebut, akan dikalikan dengan sebuah bobot yang khusus untuk perpindahan dari *node* k ke *node n*.

Hubungan tersebut dapat dimodelkan sebagai perkalian matriks sebagai berikut.

$$N = Activation(K * W)$$

Pada persamaan tersebut, N merupakan matriks hasil *output* untuk *layer N*, K merupakan *output* dari *layer sebelumnya*, dan W merupakan matriks bobot. Matriks K memiliki ukuran D x K, dan matriks W memiliki ukuran K x N, sehingga matriks N akan memiliki ukuran D x N. Nilai D merupakan *batch size* atau jumlah data yang digunakan untuk melakukan *training*. Hasil dari perkalian K * W akan dimasukkan terlebih dahulu ke fungsi aktivasi.

C. Penjelasan *Backward Propagation* dan *Weight Update*

```
    def FFNNBackPropagation(self, current_target):
        deltas = []
        delta_weights = []
        n = len(self.weights)

        for i in range(n, 0, -1):
```

```
            # Calculate delta matrix
            output = self.layer_results[i]
# Output (Oj) matrix

            if i == n:
                # Output layer
                if self.activations[i] == Activation.softmax:
                    delta = current_target - output
                else:
                    delta =
Loss.getErrorDerivativeMatrix(self.loss_function, current_target,
output) * Activation.getDerivativeMatrix(self.activations[i],
output)
            else:
                # Hidden layer
                weight_ds = Matrix.removeBiasRow(self.weights[i])
# Downstream weight (Wkj) matrix
                delta_ds = deltas[0]
# Downstream delta (delta_k) matrix

                delta =
Activation.getDerivativeMatrix(self.activations[i], output) *
(np.matmul(delta_ds, np.transpose(weight_ds)))

            deltas = [delta] + deltas

            # Calculate new weights
            layer_input = Matrix.addBiasColumn(self.layer_results[i
- 1])   # Input (Xji) matrix
            weight_change = self.learning_rate *
(np.matmul(np.transpose(layer_input), delta))      # delta_w (n *
delta_j * xji)

            delta_weights = [weight_change] + delta_weights

        # Set current epoch's gradient array
        self.deltas = deltas

        # Update old weights after backpropagation has finished
```

```
        for i, weight_change in enumerate(delta_weights):
            self.weights[i] += weight_change
```

Backpropagation merupakan proses model untuk melakukan pembelajaran dari data dengan menghitung *error* dari masing-masing data dengan nilai aslinya, lalu mengubah bobot agar lebih sesuai dengan hasil yang diinginkan. Proses pembelajaran ini dilakukan berkali-kali, dalam banyak *epoch*, setiap iterasi melakukan pembelajaran dengan memperbagus bobot bergantung data *training*.

Proses modifikasi bobot dilakukan dengan konsep *gradient descent*, yaitu mencari perubahan bobot yang menghasilkan penurunan terbesar pada fungsi *error*, atau dapat diformulasikan sebagai mencari gradien terkecil dari fungsi tersebut:

$$\frac{\partial E}{\partial w} = \left(\frac{\partial E}{\partial a}\right) * \left(\frac{\partial a}{\partial z}\right) * \left(\frac{\partial z}{\partial w}\right)$$

Menggunakan formula tersebut, bobot baru dapat dihitung dengan persamaan berikut:

$$\Delta w = -\eta \frac{\partial E}{\partial z}$$

Dapat dilihat bahwa persamaan untuk perubahan bobot mengandung turunan fungsi *error* terhadap hasil *sum* pada suatu *node* yaitu z. Term ini dapat dipecah menjadi dua term pertama pada formula sebelumnya. Term turunan *error* terhadap z ini juga disebut sebagai *delta*, gradien, atau *error term*. Sehingga untuk mendapatkan perubahan bobot, cukup menghitung turunan tersebut.

Term delta dapat dipecah menjadi dua bagian, yaitu dE/da, yaitu turunan *error* terhadap fungsi aktivasi, dan da/dz, yaitu turunan fungsi aktivasi terhadap *sum* pada suatu *node*. Term dE/da hanyalah turunan dari fungsi *loss* yang digunakan, sedangkan term da/dz hanyalah turunan dari fungsi aktivasi. Sedangkan pengali dz/dw adalah turunan dari:

$$z = \Sigma w_k x_k$$

terhadap w, sehingga menghasilkan *sum* dari x_k saja. Berdasarkan kode, dihitung terlebih dahulu variabel *delta*, yaitu dE/da

(`getErrorDerivativeMatrix()`) dikalikan dengan da/dz, yaitu turunan fungsi aktivasi `getDerivativeMatrix(self.activations[i], output)`. Setelah itu, *delta* dikalikan dengan output dari *layer* sebelumnya, atau matriks x_k.

Terdapat perbedaan jika sedang melakukan *backpropagation* untuk *layer output* dengan *layer* lainnya. Untuk *output layer,* cukup dilakukan seperti yang sudah disebutkan. Untuk *layer* lainnya, perlu dilakukan rekursi dengan *output* dan *delta* dari layer setelahnya.

Bab III
Hasil Pengujian

Catatan: Pada saat *training*, terdapat sebuah *bug kecil* pada penyimpanan *training loss* dan *validation loss*, yang menyebabkan beberapa hasil pengujian tersebut memiliki nilai *loss* yang dibagi 10 dengan tidak sengaja. Kesalahan tersebut hanya berlaku pada *list loss*, tidak pada perhitungan model. Sehingga untuk pengujian-pengujian berikut, kecuali disebutkan di awal, maka seluruh nilai *loss* yang sebenarnya adalah L * 10.

A. *Base Model* dengan *Base Parameter*

Akan ditetapkan sebuah *base model*, yaitu sebuah model yang memiliki *base parameter* yang akan menjadi perbandingan terhadap model lainnya. Model lainnya akan memiliki variasi dari satu buah parameter, namun parameter lainnya akan sama seperti *base model*.

Parameter untuk *base model* adalah sebagai berikut:

- 2 hidden layer, [128 neuron, 64 neuron]
- Fungsi aktivasi tanh
- *Learning rate* = 0.01
- *Epoch = 20*
- *Batch size = 32*
- Inisialisasi bobot distribusi *uniform*

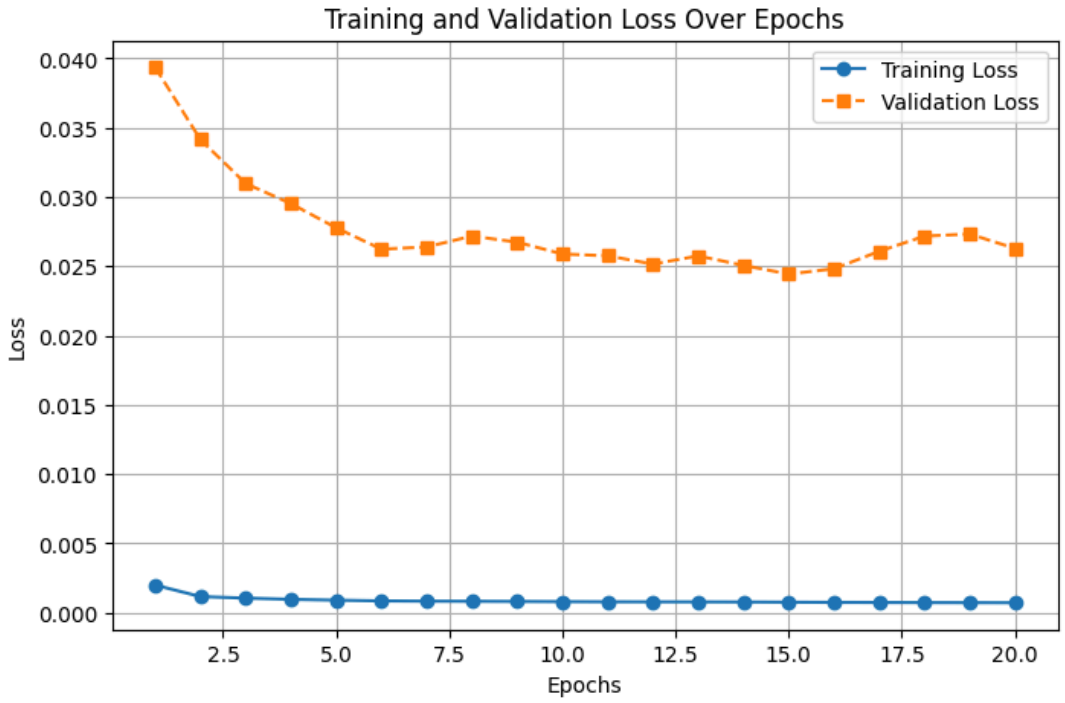
B. Hasil variasi *width* (jumlah neuron per layer)

a. *Hidden Layer [32, 32]*


```
[Epoch 1]:
Training Loss: 0.001993413826618404
Validation Loss: 0.03935695558488471
```

```
[Epoch 2]:
Training Loss: 0.0011637838878720379
Validation Loss: 0.03414250820120088

[Epoch 3]:
Training Loss: 0.0010535505204276692
Validation Loss: 0.030970632465083216

[Epoch 4]:
Training Loss: 0.0009638409042403008
Validation Loss: 0.02952029307909131

[Epoch 5]:
Training Loss: 0.0009037317409854668
Validation Loss: 0.0277752670834018

[Epoch 6]:
Training Loss: 0.0008520857372911044
Validation Loss: 0.026230477213554496

[Epoch 7]:
Training Loss: 0.0008326056054323164
Validation Loss: 0.026393538592013047

[Epoch 8]:
Training Loss: 0.0008247237037849512
Validation Loss: 0.02717660366249313

[Epoch 9]:
Training Loss: 0.0008122887963046698
Validation Loss: 0.026738745898679143

[Epoch 10]:
Training Loss: 0.0007980271563889162
Validation Loss: 0.025882119317394855

[Epoch 11]:
Training Loss: 0.0007855818603497057
Validation Loss: 0.025754465397094896

[Epoch 12]:
Training Loss: 0.0007823997630239525
Validation Loss: 0.02515641385306443

[Epoch 13]:
Training Loss: 0.0007785378621956142
Validation Loss: 0.025746055338441418

[Epoch 14]:
Training Loss: 0.0007716262744231401
Validation Loss: 0.02506290958668466
```

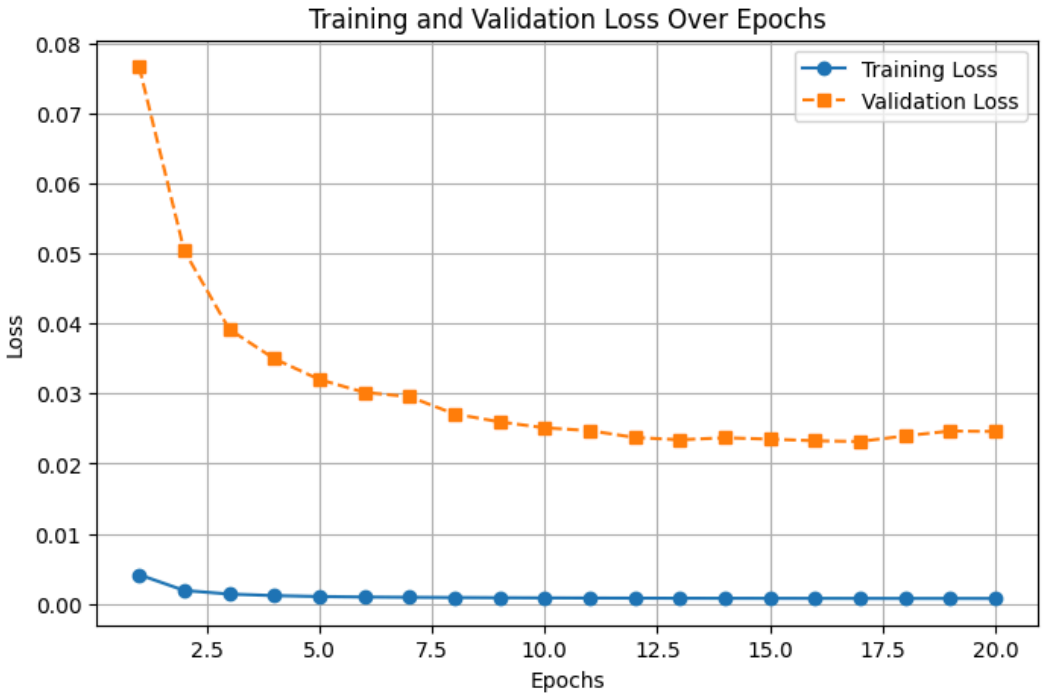
```
[Epoch 15]:
Training Loss: 0.0007572544289623868
Validation Loss: 0.024436580673141414

[Epoch 16]:
Training Loss: 0.0007453606796049657
Validation Loss: 0.024817553710657757

[Epoch 17]:
Training Loss: 0.0007387205047034845
Validation Loss: 0.026076039211360886

[Epoch 18]:
Training Loss: 0.0007312260832597877
Validation Loss: 0.027173825847552534

[Epoch 19]:
Training Loss: 0.0007250778410319475
Validation Loss: 0.027327100246117092

[Epoch 20]:
Training Loss: 0.0007200405055360144
Validation Loss: 0.026285202485103364
```



b. *Hidden Layer [32, 128]*


```
[Epoch 1]:
Training Loss: 0.004152205716794555
Validation Loss: 0.07659606130320927

[Epoch 2]:
Training Loss: 0.0018979399301274306
Validation Loss: 0.05037814282251015

[Epoch 3]:
Training Loss: 0.0013872796390054464
Validation Loss: 0.039154302853137056

[Epoch 4]:
Training Loss: 0.001165505313689666
Validation Loss: 0.03495597156071559

[Epoch 5]:
Training Loss: 0.0010353804291685091
Validation Loss: 0.031990815896122585

[Epoch 6]:
Training Loss: 0.0009642207330488024
Validation Loss: 0.030154031081920774

[Epoch 7]:
Training Loss: 0.0009193969530604944
Validation Loss: 0.02953742199172546

[Epoch 8]:
Training Loss: 0.0008855652520237144
Validation Loss: 0.027078715523143275

[Epoch 9]:
Training Loss: 0.0008576710250977216
Validation Loss: 0.025948071268743995

[Epoch 10]:
```

```
Training Loss: 0.0008338846724109448
Validation Loss: 0.025099603573621638

[Epoch 11]:
Training Loss: 0.00081524387328493
Validation Loss: 0.02469662231945965

[Epoch 12]:
Training Loss: 0.0008015002790754455
Validation Loss: 0.02372350395200112

[Epoch 13]:
Training Loss: 0.0007888775929411996
Validation Loss: 0.02339116113375394

[Epoch 14]:
Training Loss: 0.0007808323861343686
Validation Loss: 0.02367772578424636

[Epoch 15]:
Training Loss: 0.0007741653501880453
Validation Loss: 0.023487056804237117

[Epoch 16]:
Training Loss: 0.0007685395428348364
Validation Loss: 0.02324910235925231

[Epoch 17]:
Training Loss: 0.0007643512698918O8
Validation Loss: 0.023147030729887635

[Epoch 18]:
Training Loss: 0.0007584443907659197
Validation Loss: 0.023942020256361024

[Epoch 19]:
Training Loss: 0.0007580875861756302
Validation Loss: 0.02463810966115419

[Epoch 20]:
Training Loss: 0.0007553957552106237
Validation Loss: 0.024571809982456897
```

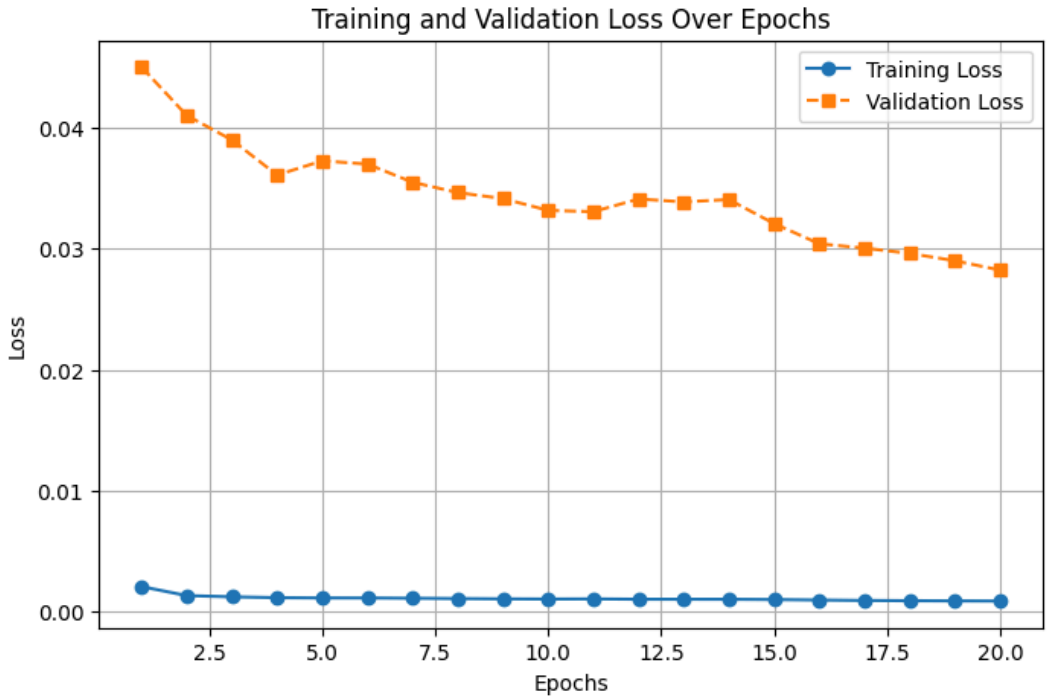


c. *Hidden Layer [128, 32]*


```
[Epoch 1]:
Training Loss: 0.002077437134208766
Validation Loss: 0.04501421240127375

[Epoch 2]:
Training Loss: 0.001331854094573999
Validation Loss: 0.04101886878803412

[Epoch 3]:
Training Loss: 0.0012306348011065572
Validation Loss: 0.03899009987159531

[Epoch 4]:
Training Loss: 0.001160946082317758
```

```
Validation Loss: 0.03609216064477764

[Epoch 5]:
Training Loss: 0.0011402637129094098
Validation Loss: 0.03726565827930403

[Epoch 6]:
Training Loss: 0.0011380357779321828
Validation Loss: 0.036997065003136924

[Epoch 7]:
Training Loss: 0.0011143029511433355
Validation Loss: 0.035493587884359815

[Epoch 8]:
Training Loss: 0.0010846305559428777
Validation Loss: 0.03464040216572321

[Epoch 9]:
Training Loss: 0.0010607471010312172
Validation Loss: 0.0341441310502751

[Epoch 10]:
Training Loss: 0.0010458064051471867
Validation Loss: 0.03317168062141958

[Epoch 11]:
Training Loss: 0.0010576970577907173
Validation Loss: 0.03306265719312221

[Epoch 12]:
Training Loss: 0.001031259277379374
Validation Loss: 0.034110655228960146

[Epoch 13]:
Training Loss: 0.0010290130004342187
Validation Loss: 0.03387364995684356

[Epoch 14]:
Training Loss: 0.0010275933951112276
Validation Loss: 0.03406327832986986

[Epoch 15]:
Training Loss: 0.0010064348684136809
Validation Loss: 0.03208281082642864

[Epoch 16]:
Training Loss: 0.0009658989035380056
Validation Loss: 0.030411880073568157

[Epoch 17]:
Training Loss: 0.0009301733618419505
Validation Loss: 0.03005289773159892
```

```
[Epoch 18]:
Training Loss: 0.0009101583293380081
Validation Loss: 0.029615133392796554

[Epoch 19]:
Training Loss: 0.0008963830186307586
Validation Loss: 0.02901013155923195

[Epoch 20]:
Training Loss: 0.0008835640904924482
Validation Loss: 0.0282370799636432
```



d. Perbandingan

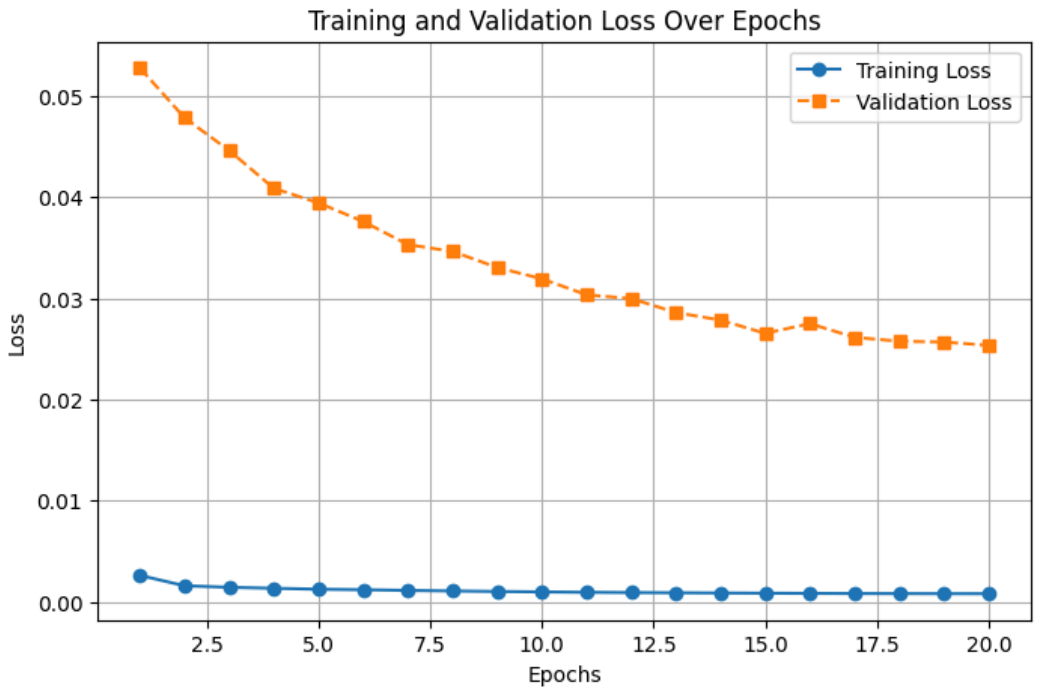
Untuk variasi jumlah neuron, didapat tidak terlalu banyak perubahan pada *validation loss*. Ketiga model rata-rata mendapatkan nilai *loss* sekitar 0.25 pada *epoch 10*. Sehingga dapat disimpulkan kelebaran layer tidak terlalu berpengaruh pada hasil model.

C. Hasil variasi *depth* (jumlah layer)

a. *Hidden Layer [64, 64]*


```
[Epoch 1]:
Training Loss: 0.0026024140462443898
Validation Loss: 0.05275876350533587

[Epoch 2]:
Training Loss: 0.0015758753768740966
Validation Loss: 0.04782995774966104

[Epoch 3]:
Training Loss: 0.001421601970182215
Validation Loss: 0.044594111492482914

[Epoch 4]:
Training Loss: 0.0013267727698628887
Validation Loss: 0.040856491565333414

[Epoch 5]:
Training Loss: 0.0012328011274299355
Validation Loss: 0.03941802862085311

[Epoch 6]:
Training Loss: 0.0011812089278585862
Validation Loss: 0.03758436176021608

[Epoch 7]:
Training Loss: 0.001114501342642192
Validation Loss: 0.03529651994048343

[Epoch 8]:
Training Loss: 0.0010607633535297213
Validation Loss: 0.03462186166712282

[Epoch 9]:
Training Loss: 0.0010109984708902861
Validation Loss: 0.03303632650823814
```

```
[Epoch 10]:
Training Loss: 0.000969920864973766
Validation Loss: 0.031913898883918095

[Epoch 11]:
Training Loss: 0.0009325179196381833
Validation Loss: 0.030331522097691497

[Epoch 12]:
Training Loss: 0.0009014721223760436
Validation Loss: 0.029964471479039984

[Epoch 13]:
Training Loss: 0.000881164275285813
Validation Loss: 0.028581381727087508

[Epoch 14]:
Training Loss: 0.0008576145666887885
Validation Loss: 0.02783412967772318

[Epoch 15]:
Training Loss: 0.0008377265108816899
Validation Loss: 0.026524311118812347

[Epoch 16]:
Training Loss: 0.0008227389556903813
Validation Loss: 0.027484953030546624

[Epoch 17]:
Training Loss: 0.000809377852084707
Validation Loss: 0.026129504477202685

[Epoch 18]:
Training Loss: 0.0008012199005438289
Validation Loss: 0.025746585978823275

[Epoch 19]:
Training Loss: 0.0007956388924245955
Validation Loss: 0.025665263919981515

[Epoch 20]:
Training Loss: 0.0007924509963084859
Validation Loss: 0.02533714925005167
```



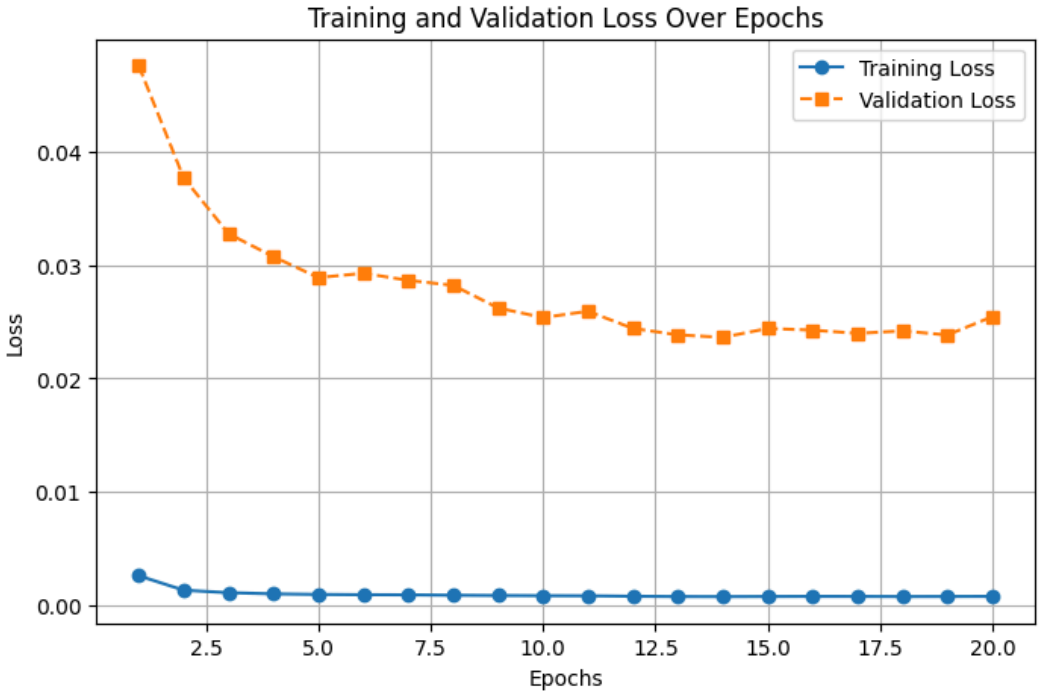
b. ***Hidden Layer [64, 64, 64]***

```
[Epoch 1]:    Training Loss: 0.0025757648657631946    Validation Loss: 0.04762870704645466    Progress: [ ]
[Epoch 2]:    Training Loss: 0.0012970182785610404    Validation Loss: 0.0377242993982438     Progress: [ ]
[Epoch 3]:    Training Loss: 0.0010780710824920463    Validation Loss: 0.03278432192151554    Progress: [ ]
[Epoch 4]:    Training Loss: 0.0009725169379307922    Validation Loss: 0.03075580297326961    Progress: [ ]
[Epoch 5]:    Training Loss: 0.0009238574218075808    Validation Loss: 0.028907762766974775   Progress: [ ]
[Epoch 6]:    Training Loss: 0.0008999815282139729    Validation Loss: 0.02929232684542674    Progress: [ ]
[Epoch 7]:    Training Loss: 0.0008883126446123712    Validation Loss: 0.028665265671606325   Progress: [ ]
[Epoch 8]:    Training Loss: 0.000854009048120397     Validation Loss: 0.02822534229413928    Progress: [ ]
[Epoch 9]:    Training Loss: 0.0008287791389858564    Validation Loss: 0.02622573414046804    Progress: [ ]
[Epoch 10]:   Training Loss: 0.000811757836993392     Validation Loss: 0.025377076419856013   Progress: [ ]
[Epoch 11]:   Training Loss: 0.0007997427728166396    Validation Loss: 0.025937780429282895   Progress: [ ]
[Epoch 12]:   Training Loss: 0.0007594807788735265    Validation Loss: 0.024419122974808086   Progress: [ ]
[Epoch 13]:   Training Loss: 0.0007343628286920653    Validation Loss: 0.023852711348685357   Progress: [ ]
[Epoch 14]:   Training Loss: 0.0007288777411049832    Validation Loss: 0.02362956546076934    Progress: [ ]
[Epoch 15]:   Training Loss: 0.0007396803321550683    Validation Loss: 0.0244151168514049     Progress: [ ]
[Epoch 16]:   Training Loss: 0.0007493488836860684    Validation Loss: 0.02426856193664142    Progress: [ ]
[Epoch 17]:   Training Loss: 0.0007459101948438364    Validation Loss: 0.02398212837067318    Progress: [ ]
[Epoch 18]:   Training Loss: 0.0007347824558500191    Validation Loss: 0.024198953937503667   Progress: [ ]
[Epoch 19]:   Training Loss: 0.0007412954595598481    Validation Loss: 0.023839936469698834   Progress: [ ]
[Epoch 20]:   Training Loss: 0.000750772277660027     Validation Loss: 0.025452640820588037   Progress: [ ]
File Successfully Saved to: 'models/cw2_model'
```

```
[Epoch 1]:
Training Loss: 0.0025757648657631946
Validation Loss: 0.04762870704645466

[Epoch 2]:
Training Loss: 0.0012970182785610404
Validation Loss: 0.0377242993982438

[Epoch 3]:
Training Loss: 0.0010780710824920463
Validation Loss: 0.03278432192151554

[Epoch 4]:
Training Loss: 0.0009725169379307922
Validation Loss: 0.03075580297326961
```

```
[Epoch 5]:
Training Loss: 0.0009238574218075808
Validation Loss: 0.028907762766974775

[Epoch 6]:
Training Loss: 0.0008999815282139729
Validation Loss: 0.02929232684542674

[Epoch 7]:
Training Loss: 0.0008883126446123712
Validation Loss: 0.028665265671606325

[Epoch 8]:
Training Loss: 0.000854009048120397
Validation Loss: 0.02822534229413928

[Epoch 9]:
Training Loss: 0.0008287791389858564
Validation Loss: 0.02622573414046804

[Epoch 10]:
Training Loss: 0.000811757836993392
Validation Loss: 0.025377076419856013

[Epoch 11]:
Training Loss: 0.0007997427728166396
Validation Loss: 0.025937780429282895

[Epoch 12]:
Training Loss: 0.0007594807788735265
Validation Loss: 0.024419122974808086

[Epoch 13]:
Training Loss: 0.0007343628286920653
Validation Loss: 0.023852711348685357

[Epoch 14]:
Training Loss: 0.0007288777411049832
Validation Loss: 0.02362956546076934

[Epoch 15]:
Training Loss: 0.0007396803321550683
Validation Loss: 0.0244151168514049

[Epoch 16]:
Training Loss: 0.0007493488836860684
Validation Loss: 0.02426856193664142

[Epoch 17]:
Training Loss: 0.0007459101948438364
Validation Loss: 0.02398212837067318
```

```
[Epoch 18]:
Training Loss: 0.0007347824558500191
Validation Loss: 0.024198953937503667

[Epoch 19]:
Training Loss: 0.0007412954595598481
Validation Loss: 0.023839936469698834

[Epoch 20]:
Training Loss: 0.000750772277660027
Validation Loss: 0.025452640820588037
```

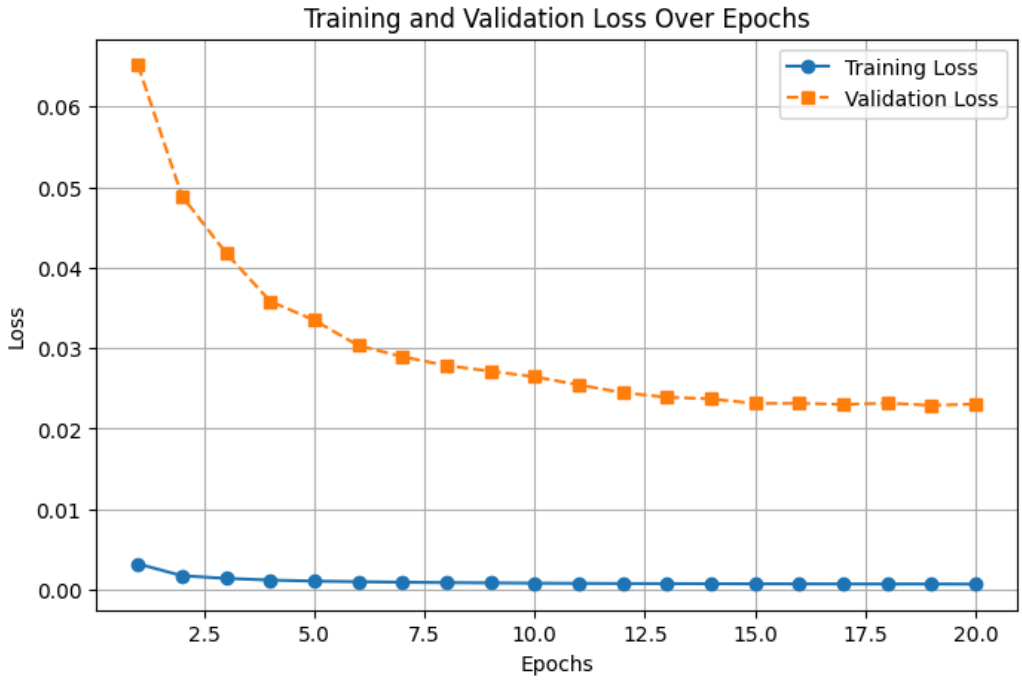


c. *Hidden Layer [64, 64, 64, 64]*


```
[Epoch 1]:
```

```
Training Loss: 0.003219876936702972
Validation Loss: 0.06518242143773711

[Epoch 2]:
Training Loss: 0.0017402468943384362
Validation Loss: 0.048913140642330696

[Epoch 3]:
Training Loss: 0.0014074903199812214
Validation Loss: 0.04182791539909665

[Epoch 4]:
Training Loss: 0.0012003176391017138
Validation Loss: 0.035837438756745534

[Epoch 5]:
Training Loss: 0.0010735679955670753
Validation Loss: 0.03348817852325981

[Epoch 6]:
Training Loss: 0.0009930843644133992
Validation Loss: 0.03033333664683513

[Epoch 7]:
Training Loss: 0.0009299888434350003
Validation Loss: 0.028947306526833715

[Epoch 8]:
Training Loss: 0.0008845361263567745
Validation Loss: 0.02785008887498348

[Epoch 9]:
Training Loss: 0.0008503788994577624
Validation Loss: 0.02714155705618374

[Epoch 10]:
Training Loss: 0.0008187548537062484
Validation Loss: 0.026455023023047704

[Epoch 11]:
Training Loss: 0.0007899218530440628
Validation Loss: 0.02545559251582547

[Epoch 12]:
Training Loss: 0.000769534174130053
Validation Loss: 0.024485441442782037

[Epoch 13]:
Training Loss: 0.0007588281178185752
Validation Loss: 0.023910594633398126

[Epoch 14]:
Training Loss: 0.0007470862857980929
```

```
Validation Loss: 0.023710263911666144

[Epoch 15]:
Training Loss: 0.0007360559802699162
Validation Loss: 0.023138084355297395

[Epoch 16]:
Training Loss: 0.0007297813830299468
Validation Loss: 0.02315783765297726

[Epoch 17]:
Training Loss: 0.0007206693668296858
Validation Loss: 0.0230096890407773

[Epoch 18]:
Training Loss: 0.0007171646594559269
Validation Loss: 0.02318528711288481

[Epoch 19]:
Training Loss: 0.000716280573253152
Validation Loss: 0.02289682800886071

[Epoch 20]:
Training Loss: 0.000707075559192638
Validation Loss: 0.02306021785889912
```



d. Perbandingan

Dari hasil variasi *depth* juga tidak didapat perbedaan yang signifikan antara ketiga model. Seluruh model mendapatkan sekitar 0.25 *loss* pada akhir *epoch* 10.

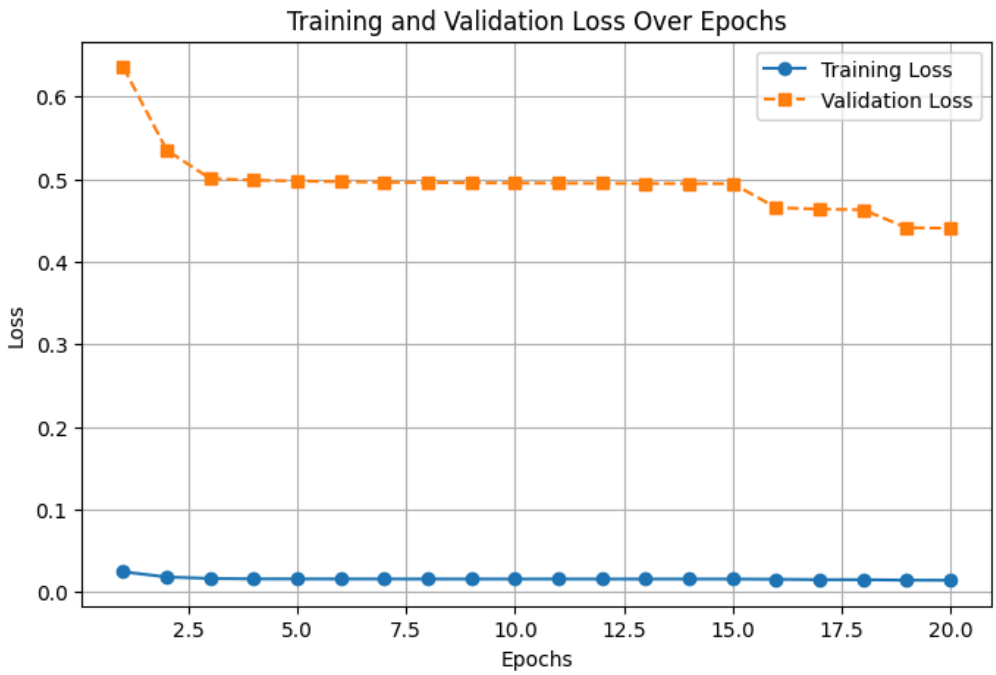
D. Hasil variasi *activation function*

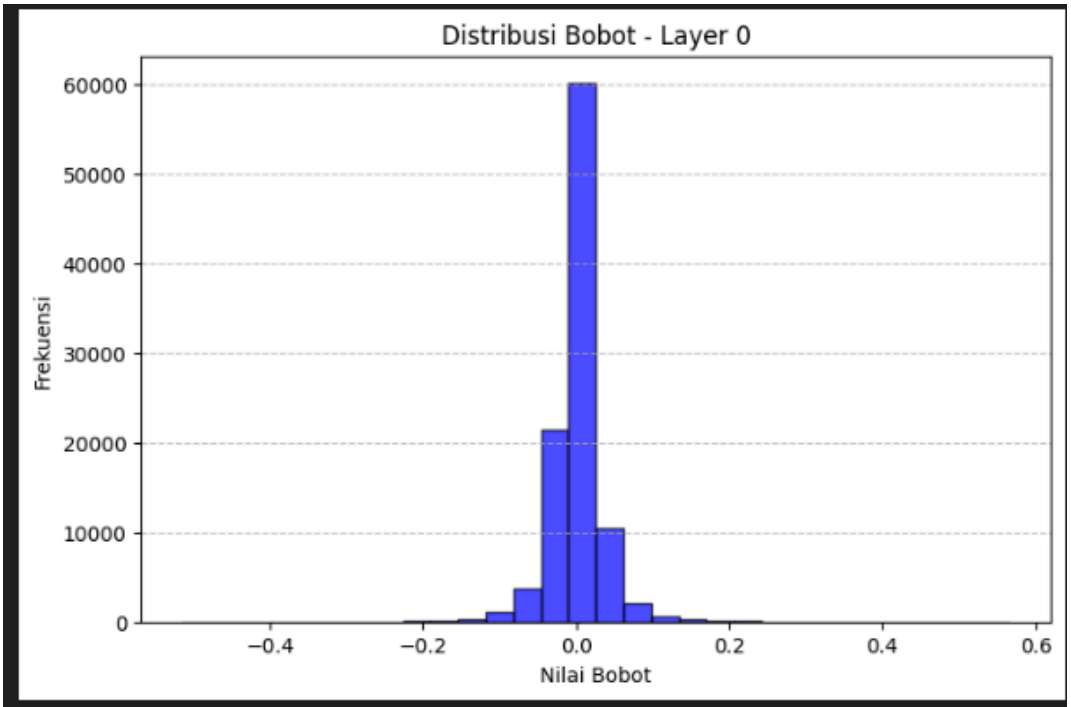
a. *Linear*

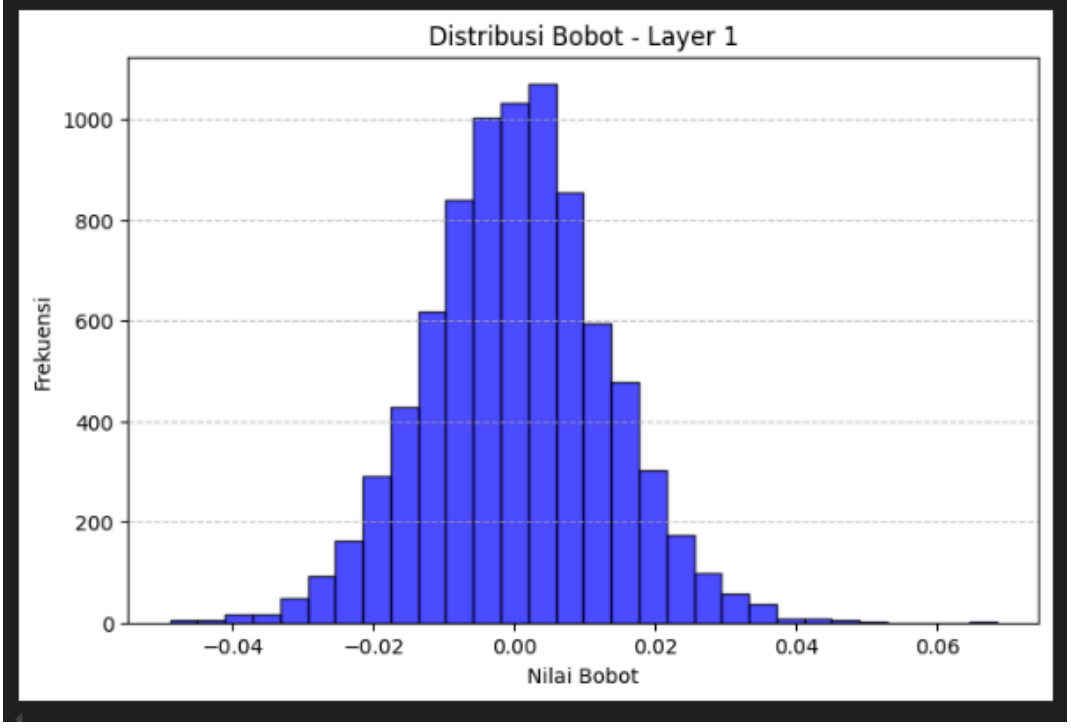
Untuk model ini, nilai *loss* yang ditampilkan adalah nilai aslinya.


```
[Epoch 1]:
Training Loss: 0.02423390703271593
Validation Loss: 0.6356625376424715

[Epoch 2]:
Training Loss: 0.01798719932011205
Validation Loss: 0.5349910737142741

[Epoch 3]:
Training Loss: 0.01618179599716462
Validation Loss: 0.5005774925223357

[Epoch 4]:
Training Loss: 0.015712331518940987
Validation Loss: 0.4987937365065114

[Epoch 5]:
Training Loss: 0.015638308587096493
Validation Loss: 0.49765280946441504

[Epoch 6]:
Training Loss: 0.015597402776547422
Validation Loss: 0.49685798419869653

[Epoch 7]:
Training Loss: 0.015571189364572484
Validation Loss: 0.496288950298649

[Epoch 8]:
Training Loss: 0.015552909186032122
Validation Loss: 0.49586668054159894

[Epoch 9]:
```

```
Training Loss: 0.01553941307571654
Validation Loss: 0.4955419000756656

[Epoch 10]:
Training Loss: 0.015529028489065264
Validation Loss: 0.4952839277739312

[Epoch 11]:
Training Loss: 0.015520783295579838
Validation Loss: 0.4950732784117403

[Epoch 12]:
Training Loss: 0.01551407349509121
Validation Loss: 0.49489725008916874

[Epoch 13]:
Training Loss: 0.015508504044601466
Validation Loss: 0.4947473277243721

[Epoch 14]:
Training Loss: 0.015503805875961644
Validation Loss: 0.4946176196653756

[Epoch 15]:
Training Loss: 0.015499787958259853
Validation Loss: 0.49450310553029503

[Epoch 16]:
Training Loss: 0.015207509988094793
Validation Loss: 0.46538319797327354

Training Loss: 0.014745572210722268
Validation Loss: 0.4636504629619188

[Epoch 18]:
Training Loss: 0.014695132761530564
Validation Loss: 0.46286052107644987

[Epoch 19]:
Training Loss: 0.014166691097897038
Validation Loss: 0.4411326420642001

[Epoch 20]:
Training Loss: 0.014036975043949879
Validation Loss: 0.4404627219721158
[Epoch 17]:
```


Training and Validation Loss Over Epochs
Training Loss
Validation Loss
Loss
Epochs
0.0
0.1
0.2
0.3
0.4
0.5
0.6
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0



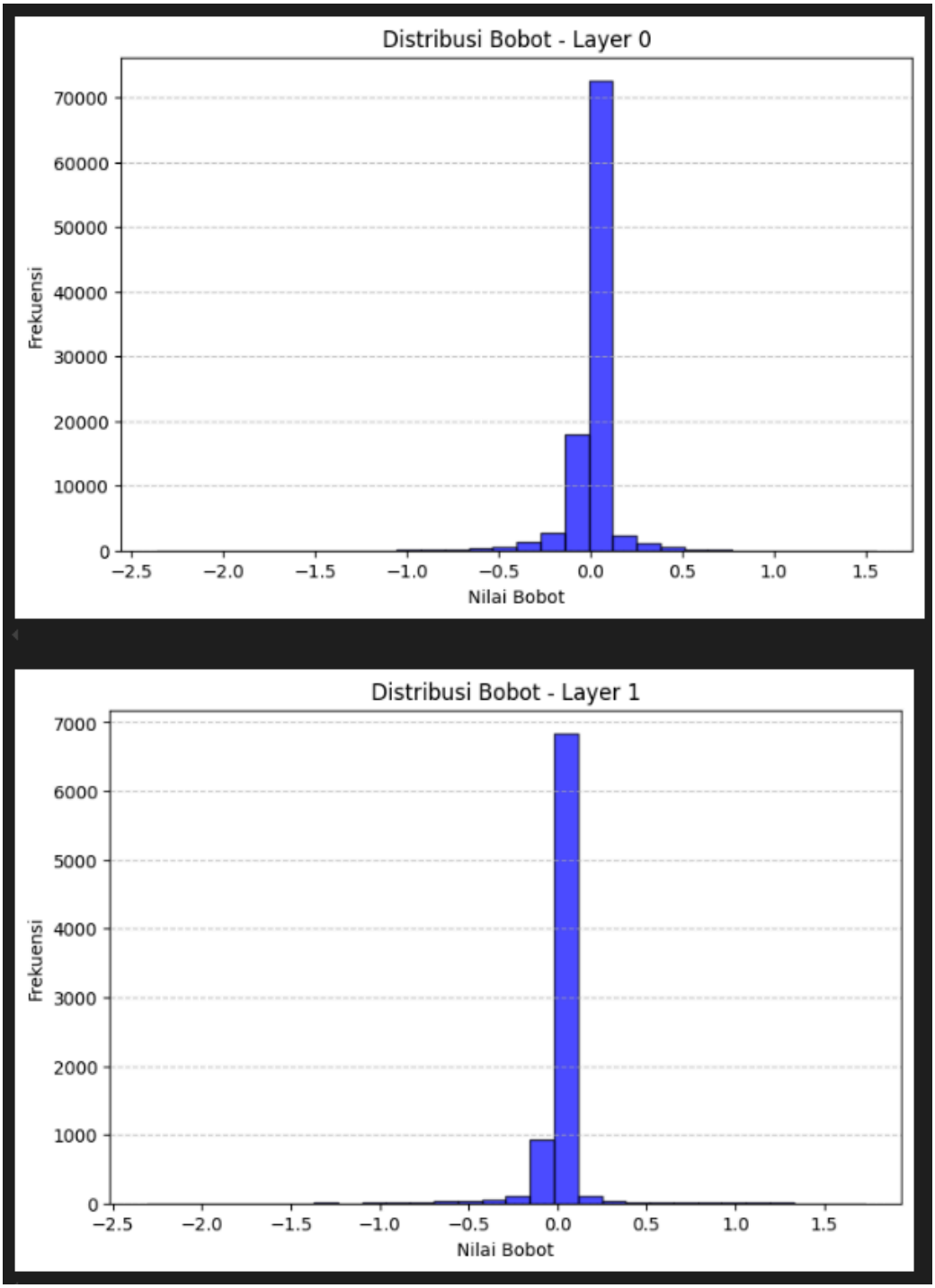


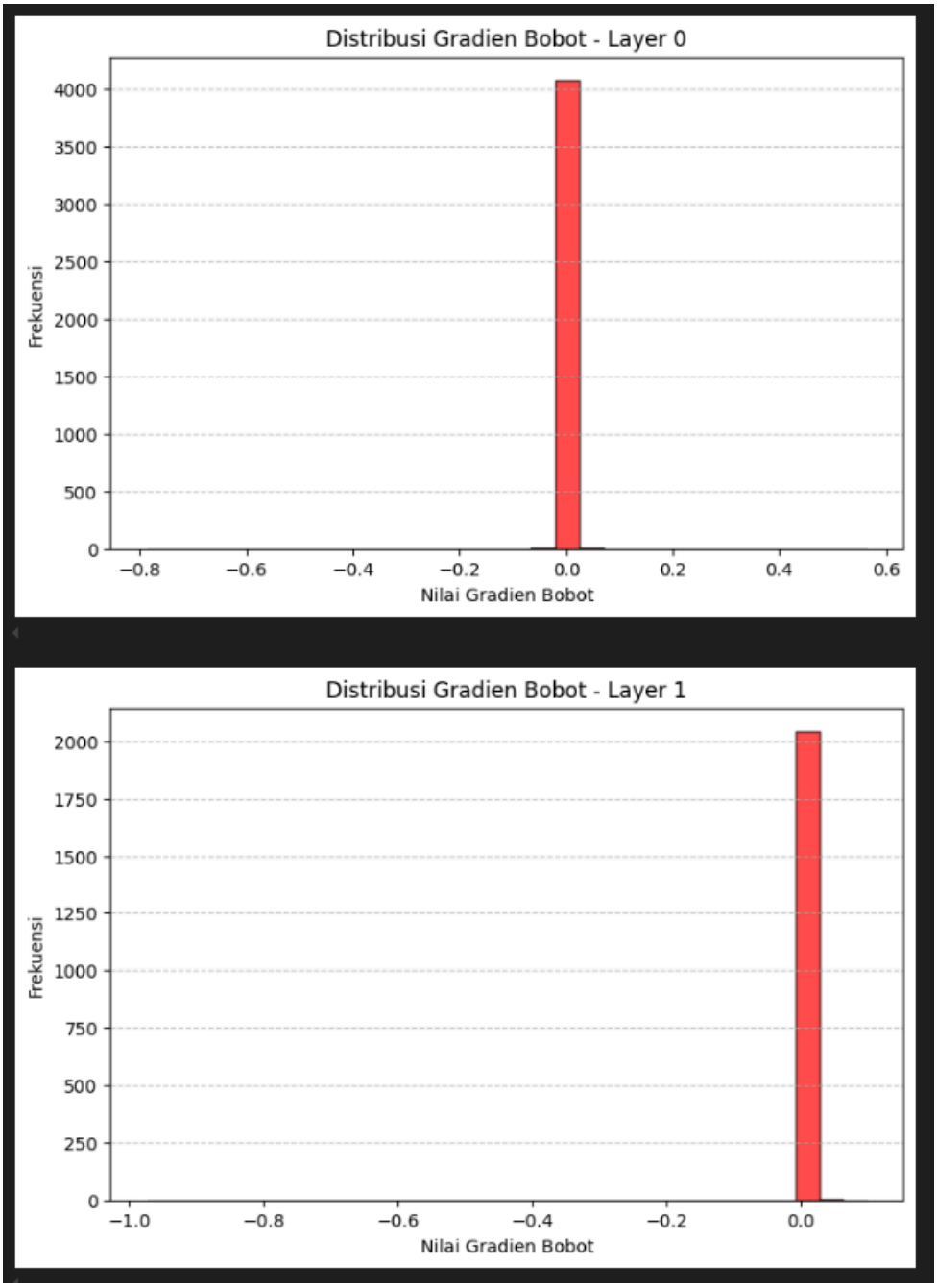
b. *ReLU*

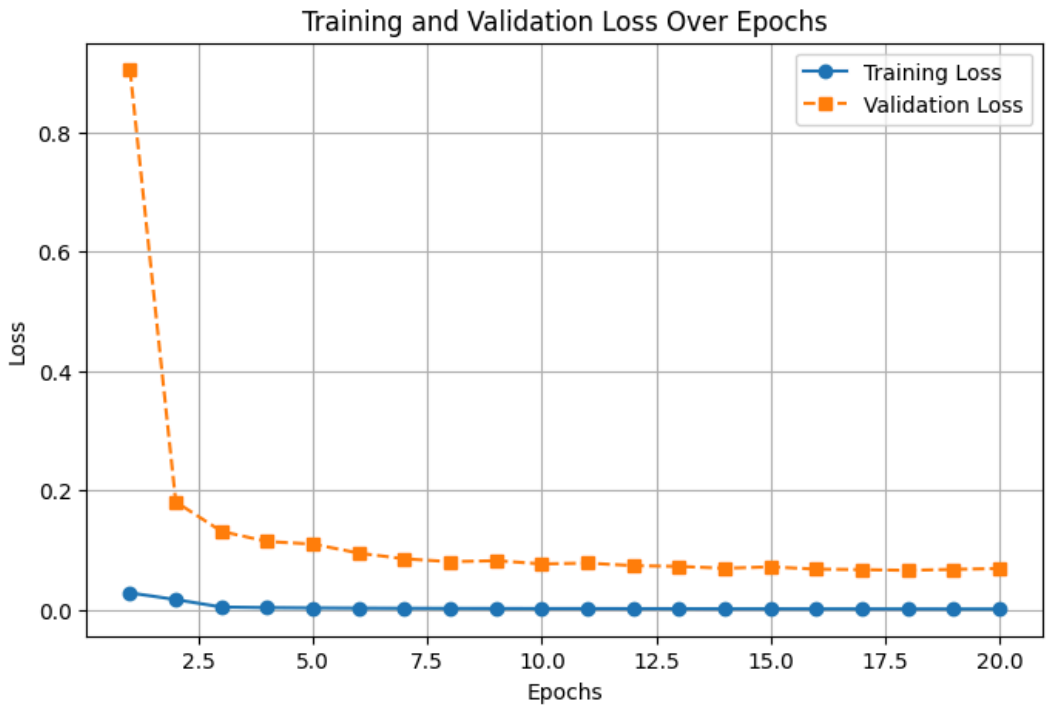
Untuk model ini, nilai *loss* yang ditampilkan adalah nilai aslinya. Catatan: Model ini menggunakan *softmax* sebagai *output activation function*, dan *loss function Categorical Cross Entropy*.


```
[Epoch 1]:
Training Loss: 0.028080227645648038
Validation Loss: 0.9044574171476628

[Epoch 2]:
Training Loss: 0.016540019102821896
Validation Loss: 0.17570714775562038

[Epoch 3]:
Training Loss: 0.0035953532601329047
Validation Loss: 0.11780476747672823

[Epoch 4]:
Training Loss: 0.0026112517895555396
Validation Loss: 0.10380239940138727

[Epoch 5]:
Training Loss: 0.0021753805646057374
Validation Loss: 0.08474610684602626

[Epoch 6]:
Training Loss: 0.0018866162577340297
Validation Loss: 0.08876943952343663

[Epoch 7]:
Training Loss: 0.001685207554312506
Validation Loss: 0.07803209335918801

[Epoch 8]:
Training Loss: 0.001597767762150984
Validation Loss: 0.06940085412131818

[Epoch 9]:
Training Loss: 0.0014383844480962981
Validation Loss: 0.06471231734311153

[Epoch 10]:
Training Loss: 0.0013628080690883013
Validation Loss: 0.06982215123259712
```

```
[Epoch 11]:
Training Loss: 0.0012811432976551989
Validation Loss: 0.06479167586189563

[Epoch 12]:
Training Loss: 0.001250756063049385
Validation Loss: 0.062115970154109745

[Epoch 13]:
Training Loss: 0.0011162141662214235
Validation Loss: 0.07717178790202693

[Epoch 14]:
Training Loss: 0.0011330706899868278
Validation Loss: 0.06682051542791655

[Epoch 15]:
Training Loss: 0.0010771213482955325
Validation Loss: 0.05577517967739363

[Epoch 16]:
Training Loss: 0.00101836724185728
Validation Loss: 0.058808830573734316

[Epoch 17]:
Training Loss: 0.0009914578099778357
Validation Loss: 0.057458544163414824

[Epoch 18]:
Training Loss: 0.0009445946969231559
Validation Loss: 0.06197063667822037

[Epoch 19]:
Training Loss: 0.000934601317760877
Validation Loss: 0.06371424153331688

[Epoch 20]:
Training Loss: 0.0008691516997128407
Validation Loss: 0.06034358217924947
```


Distribusi Bobot - Layer 0
Frekuensi
Nilai Bobot
70000
60000
50000
40000
30000
20000
10000
0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Distribusi Bobot - Layer 1
Frekuensi
Nilai Bobot
7000
6000
5000
4000
3000
2000
1000
0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Distribusi Gradien Bobot - Layer 0
Frekuensi
4000
3500
3000
2500
2000
1500
1000
500
0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
Nilai Gradien Bobot
Distribusi Gradien Bobot - Layer 1
Frekuensi
2000
1750
1500
1250
1000
750
500
250
0
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
Nilai Gradien Bobot



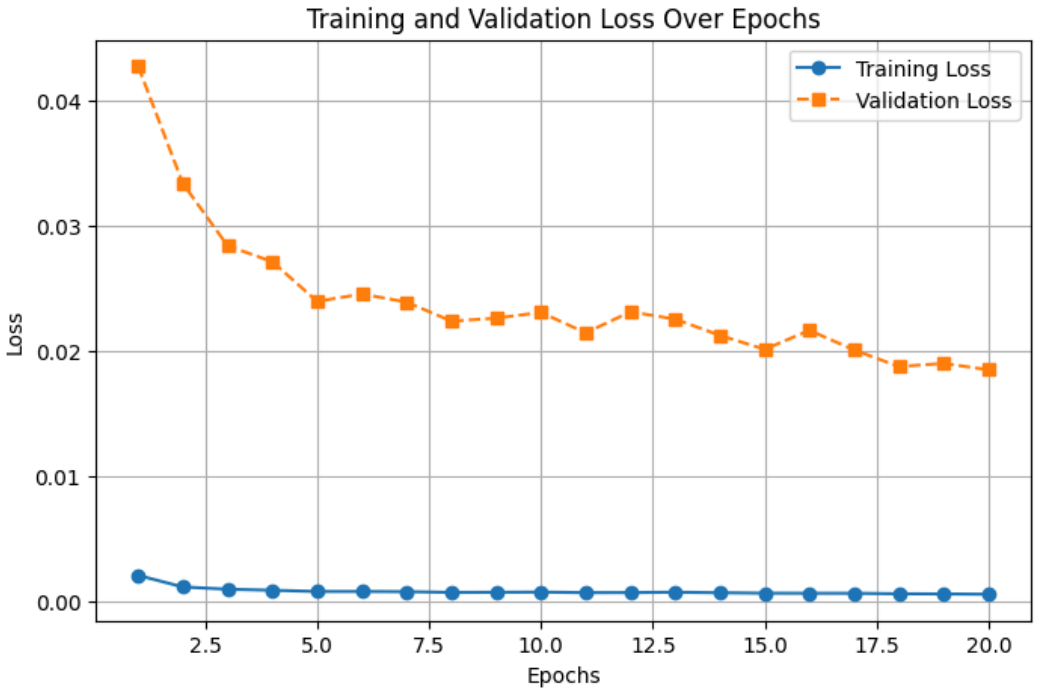
c. *Sigmoid*

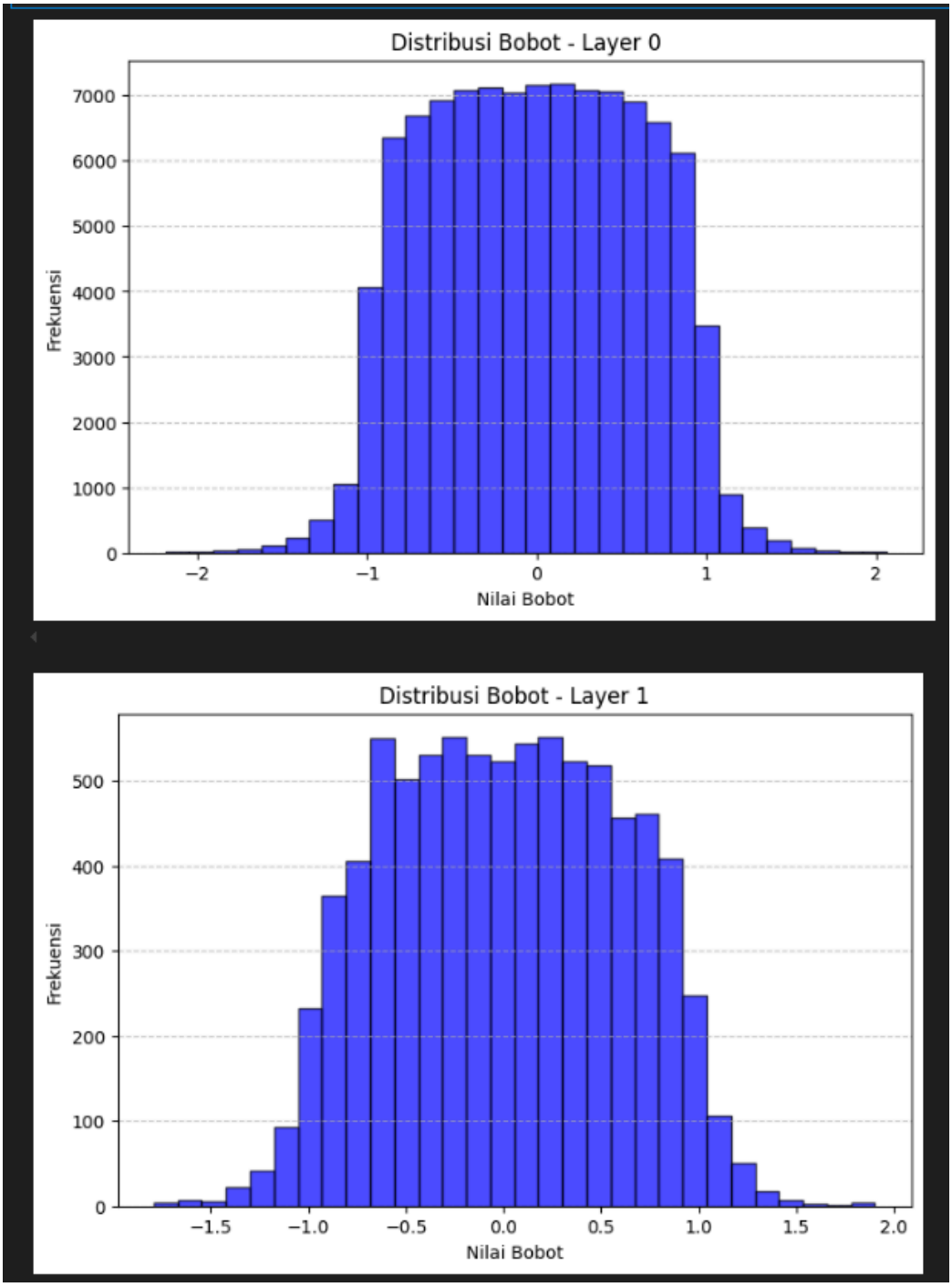
```
[Epoch 1]:     Training Loss: 0.002072876944039424     Validation Loss: 0.042733665617380116   Progress: [                    ]
[Epoch 2]:     Training Loss: 0.0011467926778048457    Validation Loss: 0.03336280948273385    Progress: [                    ]
[Epoch 3]:     Training Loss: 0.0009683211382385292    Validation Loss: 0.02839238021826575    Progress: [                    ]
[Epoch 4]:     Training Loss: 0.0008830960312775147    Validation Loss: 0.027122366999443176   Progress: [                    ]
[Epoch 5]:     Training Loss: 0.000786795217030591     Validation Loss: 0.023952391996774618   Progress: [                    ]
[Epoch 6]:     Training Loss: 0.0007868146906114134    Validation Loss: 0.024543150253779386   Progress: [                    ]
[Epoch 7]:     Training Loss: 0.000770604615132999     Validation Loss: 0.023897984593115675   Progress: [                    ]
[Epoch 8]:     Training Loss: 0.0007109624765322537    Validation Loss: 0.02237637278422594    Progress: [                    ]
[Epoch 9]:     Training Loss: 0.000721042678516216     Validation Loss: 0.022611920294121      Progress: [                    ]
[Epoch 10]:    Training Loss: 0.0007358447657568899    Validation Loss: 0.023076089529838202   Progress: [                    ]
[Epoch 11]:    Training Loss: 0.0006949125941840185    Validation Loss: 0.02144097500640312    Progress: [                    ]
[Epoch 12]:    Training Loss: 0.0006996060779712408    Validation Loss: 0.023123392036676474   Progress: [                    ]
[Epoch 13]:    Training Loss: 0.0007269895436231143    Validation Loss: 0.02253570313517756    Progress: [                    ]
[Epoch 14]:    Training Loss: 0.0006877229909702113    Validation Loss: 0.02124145124535848 5  Progress: [                    ]
[Epoch 15]:    Training Loss: 0.0006510100317650621    Validation Loss: 0.020123757075118376   Progress: [                    ]
[Epoch 16]:    Training Loss: 0.0006420913386038498    Validation Loss: 0.021644967608075282   Progress: [                    ]
[Epoch 17]:    Training Loss: 0.0006410563195289068    Validation Loss: 0.02006916612174665    Progress: [                    ]
[Epoch 18]:    Training Loss: 0.0006020613733731968    Validation Loss: 0.01875090077482039    Progress: [                    ]
[Epoch 19]:    Training Loss: 0.0005901738094031403    Validation Loss: 0.01900048199454859    Progress: [                    ]
[Epoch 20]:    Training Loss: 0.0005666921171598523    Validation Loss: 0.0184944284429768     Progress: [                    ]
File Successfully Saved to: 'models/sigmoid_model'
```

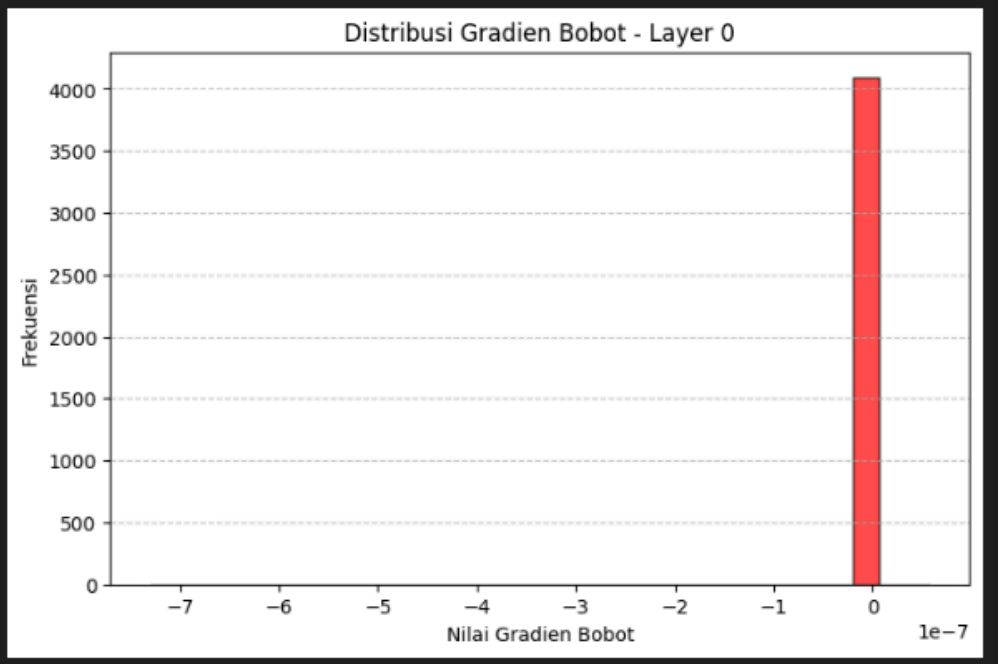
```
[Epoch 1]:
Training Loss: 0.002072876944039424
Validation Loss: 0.042733665617380116

[Epoch 2]:
Training Loss: 0.0011467926778048457
Validation Loss: 0.03336280948273385

[Epoch 3]:
Training Loss: 0.0009683211382385292
Validation Loss: 0.02839238021826575

[Epoch 4]:
```

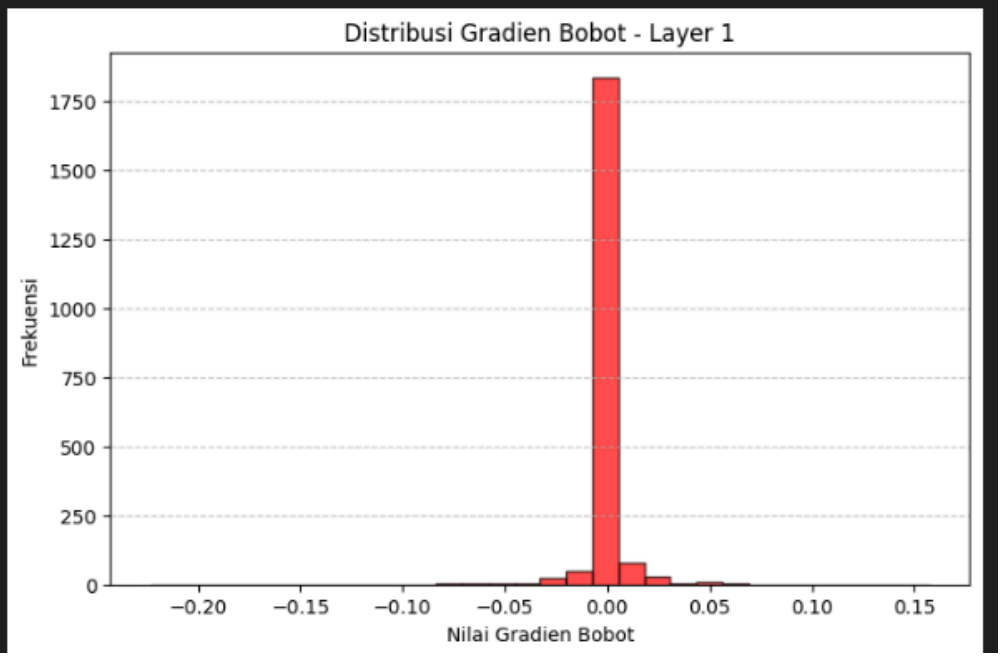
```
Training Loss: 0.0008830960312775147
Validation Loss: 0.027122366999443176

[Epoch 5]:
Training Loss: 0.000786795217030591
Validation Loss: 0.023952391996774618

[Epoch 6]:
Training Loss: 0.0007868146906114134
Validation Loss: 0.024543150253779386

[Epoch 7]:
Training Loss: 0.000770604615132999
Validation Loss: 0.023897984593115675

[Epoch 8]:
Training Loss: 0.0007109624765322537
Validation Loss: 0.02237637278422594

[Epoch 9]:
Training Loss: 0.000721042678516216
Validation Loss: 0.022611920294121

[Epoch 10]:
Training Loss: 0.0007358447657568899
Validation Loss: 0.023076089529838202

[Epoch 11]:
Training Loss: 0.0006949125941840185
Validation Loss: 0.02144097500640312

[Epoch 12]:
Training Loss: 0.0006996060779712408
Validation Loss: 0.023123392036676474

[Epoch 13]:
Training Loss: 0.0007269895436231143
Validation Loss: 0.02253570313517756

[Epoch 14]:
Training Loss: 0.0006877229909702113
Validation Loss: 0.021241451245358485

[Epoch 15]:
Training Loss: 0.0006510100317650621
Validation Loss: 0.020123757075118376

[Epoch 16]:
Training Loss: 0.0006420913386038498
Validation Loss: 0.021644967608075282

[Epoch 17]:
Training Loss: 0.0006410563195289068
```

```
Validation Loss: 0.02006916612174665

[Epoch 18]:
Training Loss: 0.0006020613733731968
Validation Loss: 0.01875090077482039

[Epoch 19]:
Training Loss: 0.0005901738094031403
Validation Loss: 0.01900048199454859

[Epoch 20]:
Training Loss: 0.0005666921171598523
Validation Loss: 0.0184944284429768
```




Distribusi Bobot - Layer 0
Frekuensi
Nilai Bobot
0
1000
2000
3000
4000
5000
6000
7000
−2
−1
0
1
2
Distribusi Bobot - Layer 1
Frekuensi
Nilai Bobot
0
100
200
300
400
500
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

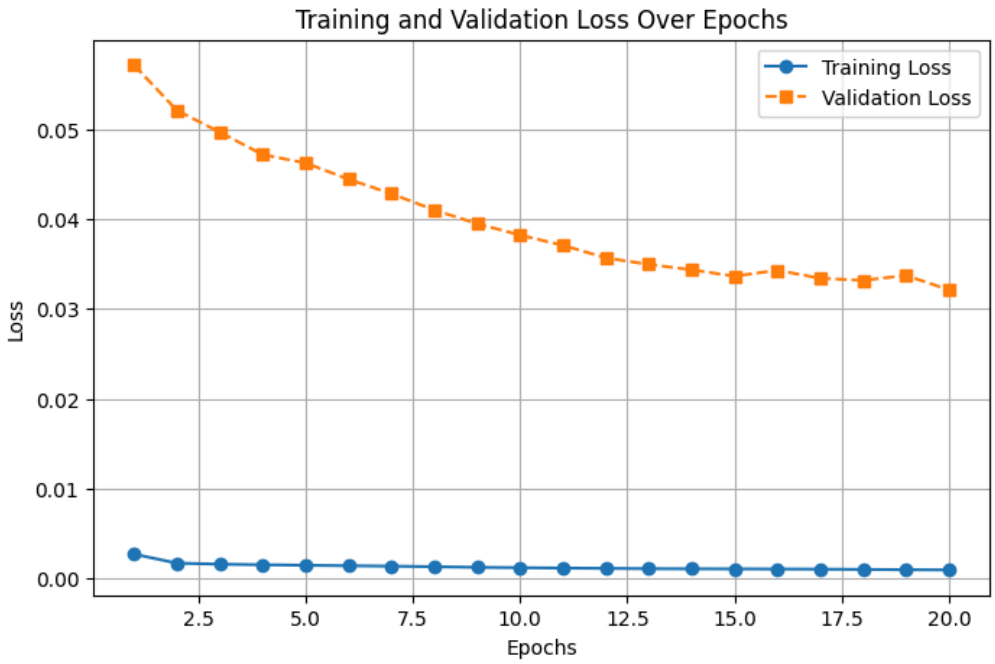

Distribusi Gradien Bobot - Layer 0
Frekuensi
4000
3500
3000
2500
2000
1500
1000
500
0
−7
−6
−5
−4
−3
−2
−1
0
1e−7
Nilai Gradien Bobot

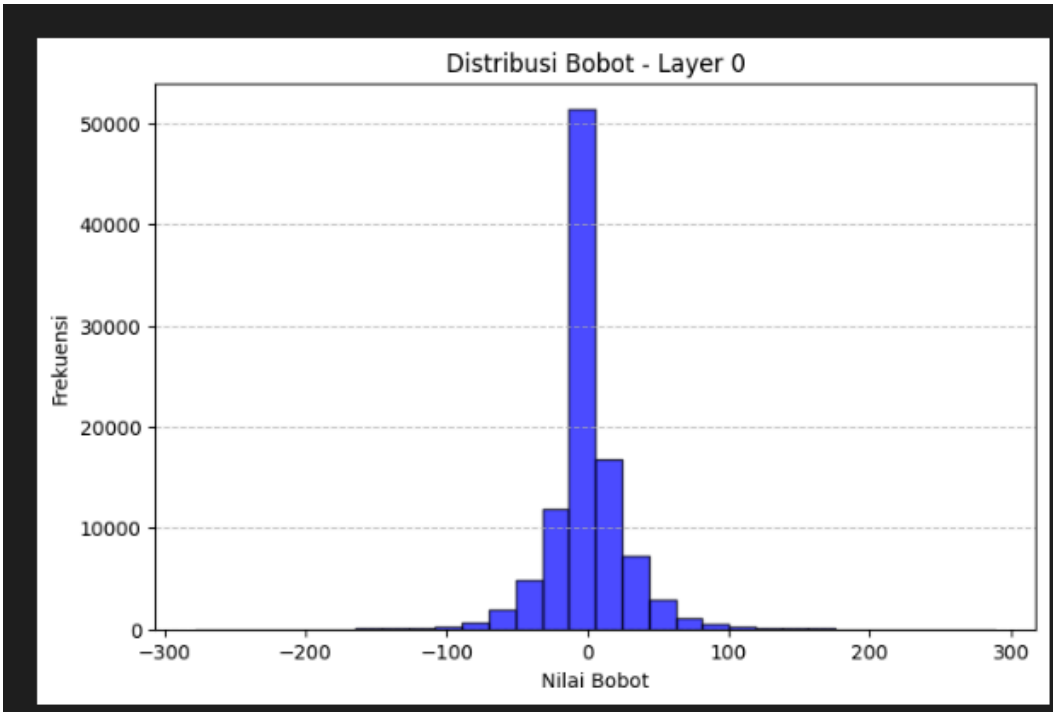

Distribusi Gradien Bobot - Layer 1
Frekuensi
1750
1500
1250
1000
750
500
250
0
−0.20
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
Nilai Gradien Bobot

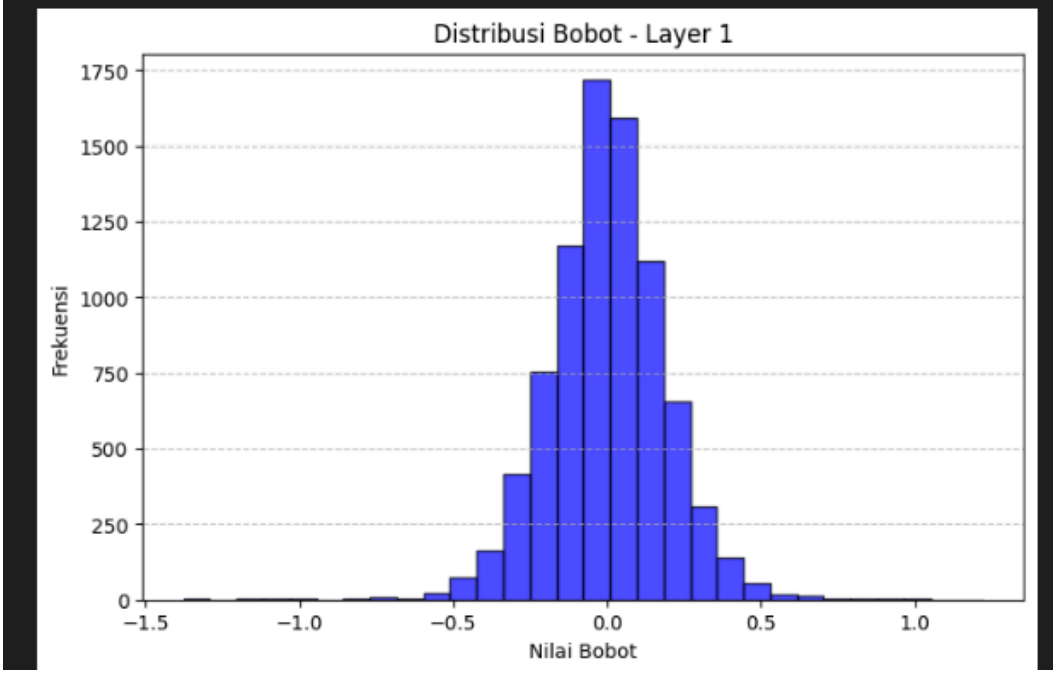
d. *tanh*

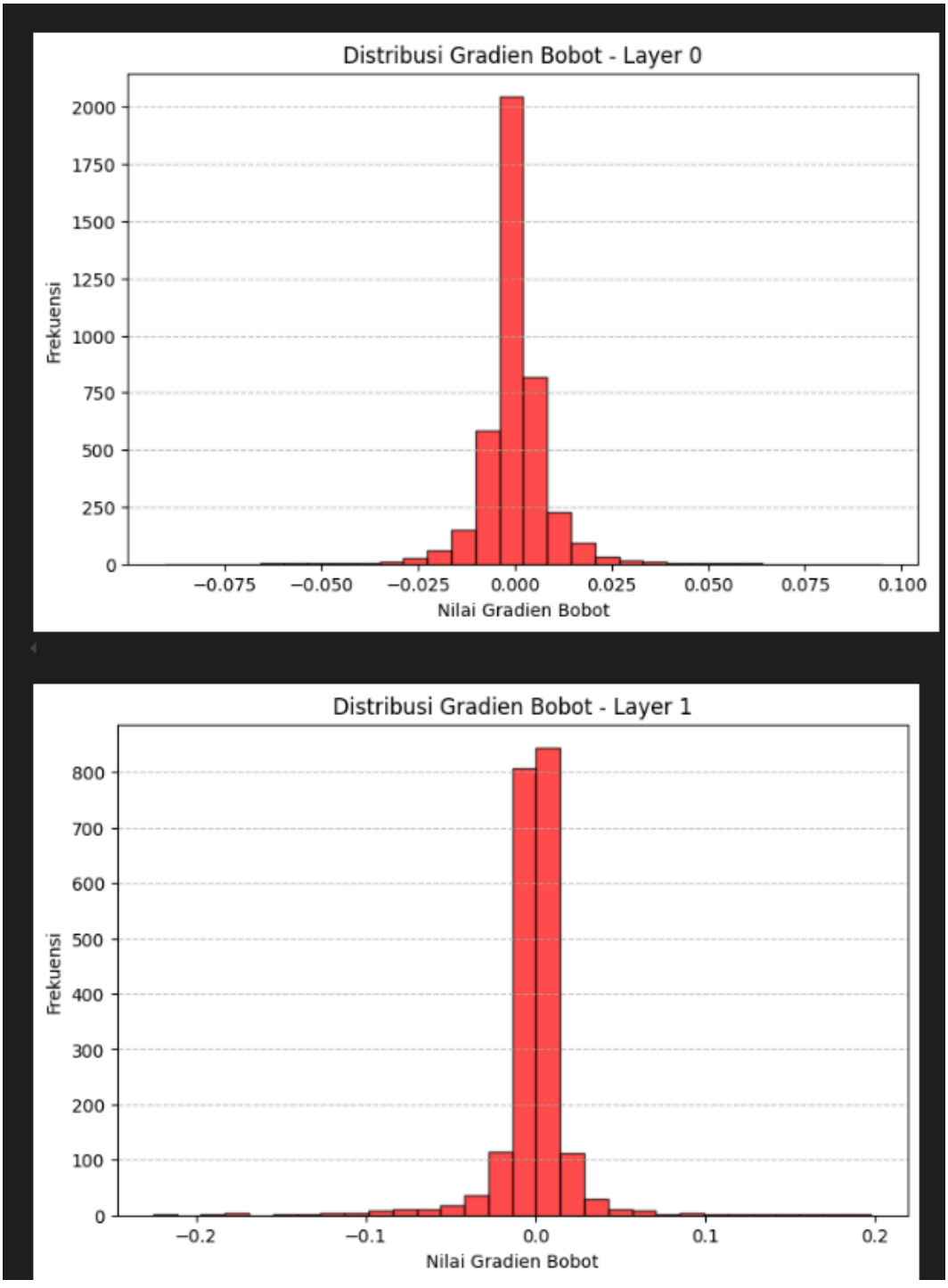

```
[Epoch 1]:
Training Loss: 0.0027099407614237245
Validation Loss: 0.0571544548208529

[Epoch 2]:
Training Loss: 0.001706889483943063
Validation Loss: 0.05210085521640847

[Epoch 3]:
Training Loss: 0.0015966776277596892
Validation Loss: 0.04967947677031759

[Epoch 4]:
Training Loss: 0.001524799263901508
Validation Loss: 0.04718012494877014

[Epoch 5]:
Training Loss: 0.0014714891992633086
Validation Loss: 0.04624765845043547

[Epoch 6]:
Training Loss: 0.0014176629959564682
Validation Loss: 0.04444701174706174

[Epoch 7]:
Training Loss: 0.001373547490030896
Validation Loss: 0.04282773304071671

[Epoch 8]:
Training Loss: 0.0013135754875201009
Validation Loss: 0.04101992988362089

[Epoch 9]:
Training Loss: 0.001244359535475869
Validation Loss: 0.039493487700896

[Epoch 10]:
Training Loss: 0.0012005088441360318
```

```
Validation Loss: 0.03822774648147555

[Epoch 11]:
Training Loss: 0.001156201363402212
Validation Loss: 0.03708439868191913

[Epoch 12]:
Training Loss: 0.0011292588930959223
Validation Loss: 0.035688982256479096

[Epoch 13]:
Training Loss: 0.0010893803837048834
Validation Loss: 0.034964862608091944

[Epoch 14]:
Training Loss: 0.0010757887409412609
Validation Loss: 0.034361206855257856

[Epoch 15]:
Training Loss: 0.0010693487356087789
Validation Loss: 0.03364834620603709

[Epoch 16]:
Training Loss: 0.001051777246899769
Validation Loss: 0.03430353493157926

[Epoch 17]:
Training Loss: 0.0010368413570126464
Validation Loss: 0.033415385826872104

[Epoch 18]:
Training Loss: 0.0010054481736810218
Validation Loss: 0.03318320388133522

[Epoch 19]:
Training Loss: 0.0009769407626764707
Validation Loss: 0.033735979472816105

[Epoch 20]:
Training Loss: 0.0009562444742805288
Validation Loss: 0.03212976849743355
```


Training and Validation Loss Over Epochs
Training Loss
Validation Loss
Loss
0.05
0.04
0.03
0.02
0.01
0.00
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epochs


Distribusi Bobot - Layer 0
Frekuensi
Nilai Bobot
50000
40000
30000
20000
10000
0
−300
−200
−100
0
100
200
300


Distribusi Bobot - Layer 1
Frekuensi
Nilai Bobot
1750
1500
1250
1000
750
500
250
0
−1.5
−1.0
−0.5
0.0
0.5
1.0



e. Perbandingan

Berdasarkan hasil variasi fungsi aktivasi, didapatkan perbedaan yang cukup signifikan antar fungsi.

- Model linear mendapatkan hasil yang cukup buruk, mendapatkan sekitar 0.5 untuk *loss*.

- Model ReLU mendapatkan hasil yang jauh lebih bagus dibanding model lainnya, mendapatkan *loss* sekitar 0.06 di akhir. Namun hal ini dapat diatribusikan ke penggunaan *softmax* pada *output layer*. Jika tidak menggunakan *softmax*, maka performa model cukup buruk, mendapatkan *loss* sekitar 0.9 dan tidak pernah berubah.
    - Hal ini dapat dijelaskan dengan *dying reLU problem*, yaitu ketika seluruh bobot menjadi 0, dan model tidak menghasilkan *output* apapun. Hal ini dapat dilihat pada grafik distribusi bobot dan gradien, yang memiliki *mean* dan variansi sangat dekat ke 0.
- Fungsi *sigmoid* memiliki hasil terbaik, menghasilkan *loss* hingga dibawah 0.2 pada *epoch 20*.
- Fungsi *tanh* tidak terlalu buruk dibanding *sigmoid*, mendapatkan sekitar 0.3 untuk *loss*.

E. Hasil variasi *learning rate*

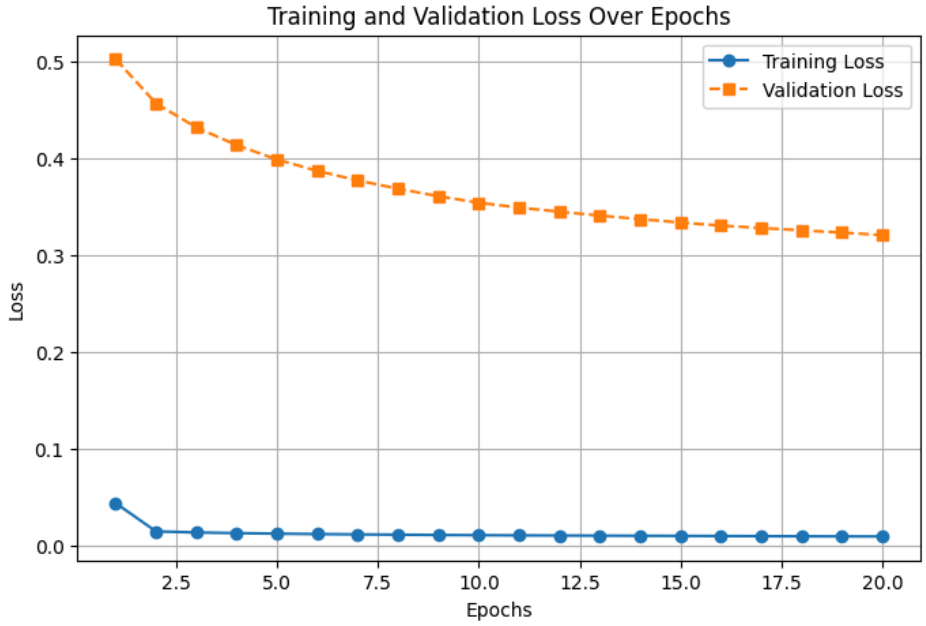
a. *n = 0.001*

Untuk model ini, nilai *loss* yang ditampilkan adalah nilai aslinya.


```
[Epoch 1]:
Training Loss: 0.04409952330234861
Validation Loss: 0.5021657158059052

[Epoch 2]:
Training Loss: 0.014926572861282118
Validation Loss: 0.45646696780965124

[Epoch 3]:
Training Loss: 0.013852389337843536
Validation Loss: 0.4316541108190246

[Epoch 4]:
Training Loss: 0.01314630466575115
```

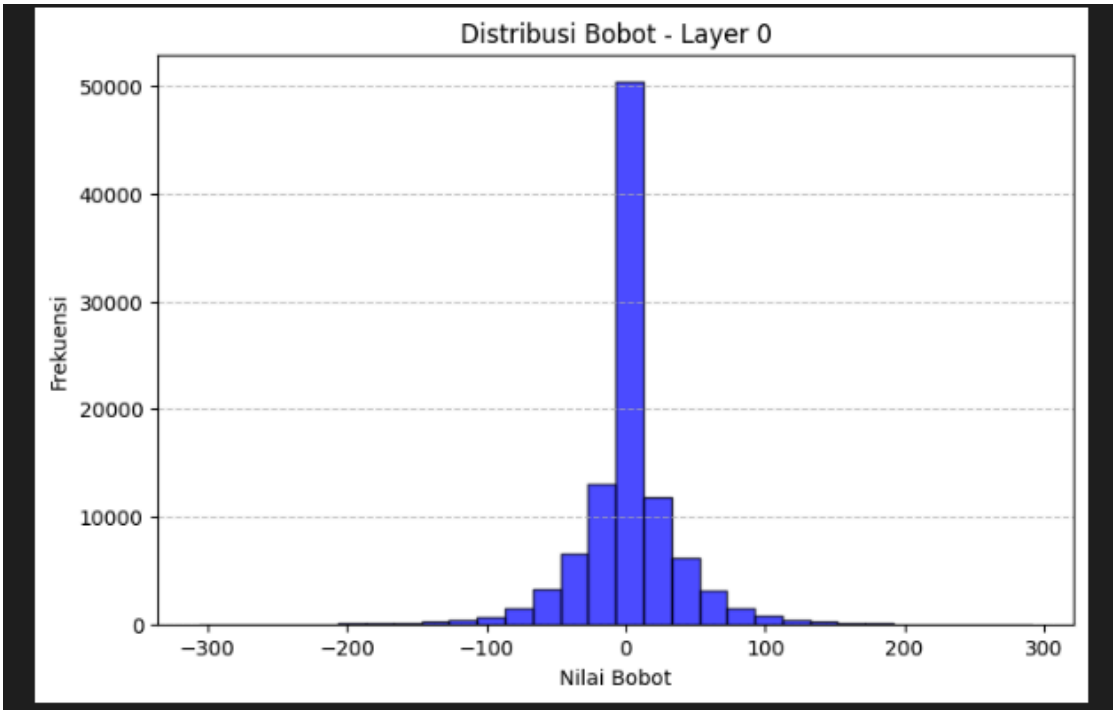
```
Validation Loss: 0.4134143322889296

[Epoch 5]:
Training Loss: 0.012619221720484885
Validation Loss: 0.39859459802908037

[Epoch 6]:
Training Loss: 0.012199382710103178
Validation Loss: 0.3867578240928198

[Epoch 7]:
Training Loss: 0.011851758384569075
Validation Loss: 0.3770221359326475

[Epoch 8]:
Training Loss: 0.011541364943786991
Validation Loss: 0.3684288651106108

[Epoch 9]:
Training Loss: 0.011279161257256856
Validation Loss: 0.36068437088754884

[Epoch 10]:
Training Loss: 0.01105593822666317
Validation Loss: 0.35402105331236616

[Epoch 11]:
Training Loss: 0.010859397217045458
Validation Loss: 0.34892137087896163

[Epoch 12]:
Training Loss: 0.010687715747111803
Validation Loss: 0.34475826755751937

[Epoch 13]:
Training Loss: 0.010536974645859398
Validation Loss: 0.34072867873331203

[Epoch 14]:
Training Loss: 0.010399205490784567
Validation Loss: 0.337078088574823

[Epoch 15]:
Training Loss: 0.010267288930663981
Validation Loss: 0.33357261277370115

[Epoch 16]:
Training Loss: 0.010140741793590615
Validation Loss: 0.3304353151743585

[Epoch 17]:
Training Loss: 0.010021433713825396
Validation Loss: 0.32791412417163346
```

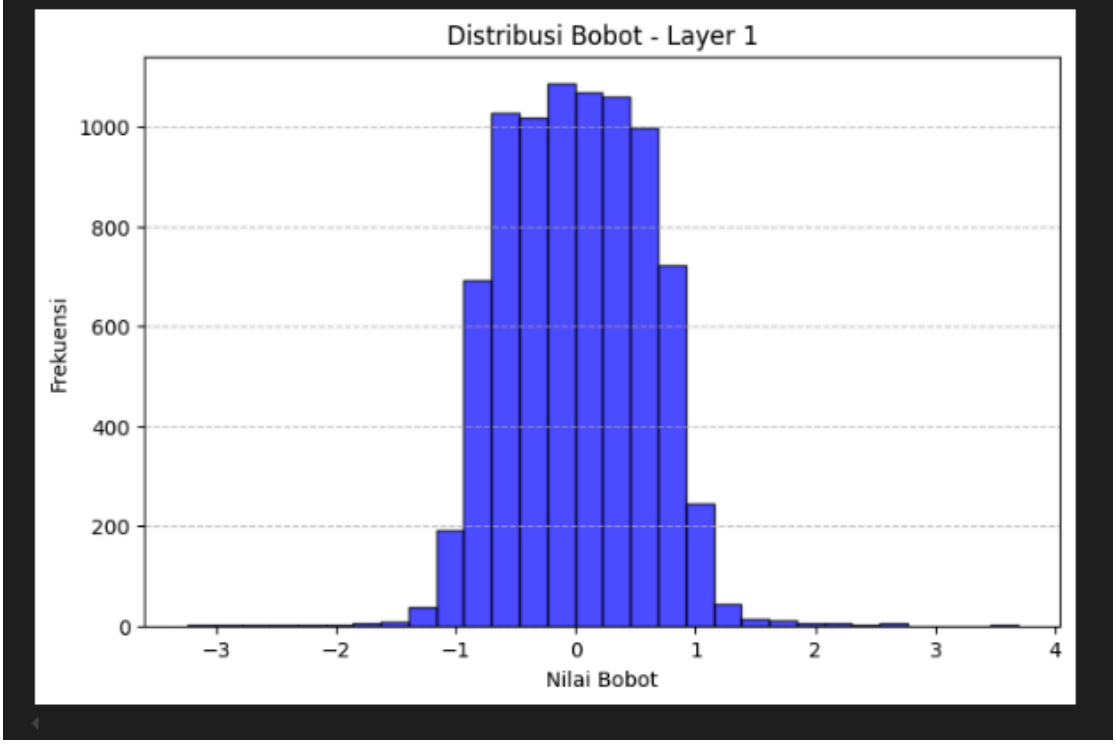
```
[Epoch 18]:
Training Loss: 0.009914133828151733
Validation Loss: 0.3255056844107895

[Epoch 19]:
Training Loss: 0.009813095279714853
Validation Loss: 0.32321155612617447

[Epoch 20]:
Training Loss: 0.009718053578809508
Validation Loss: 0.320642637957418
```

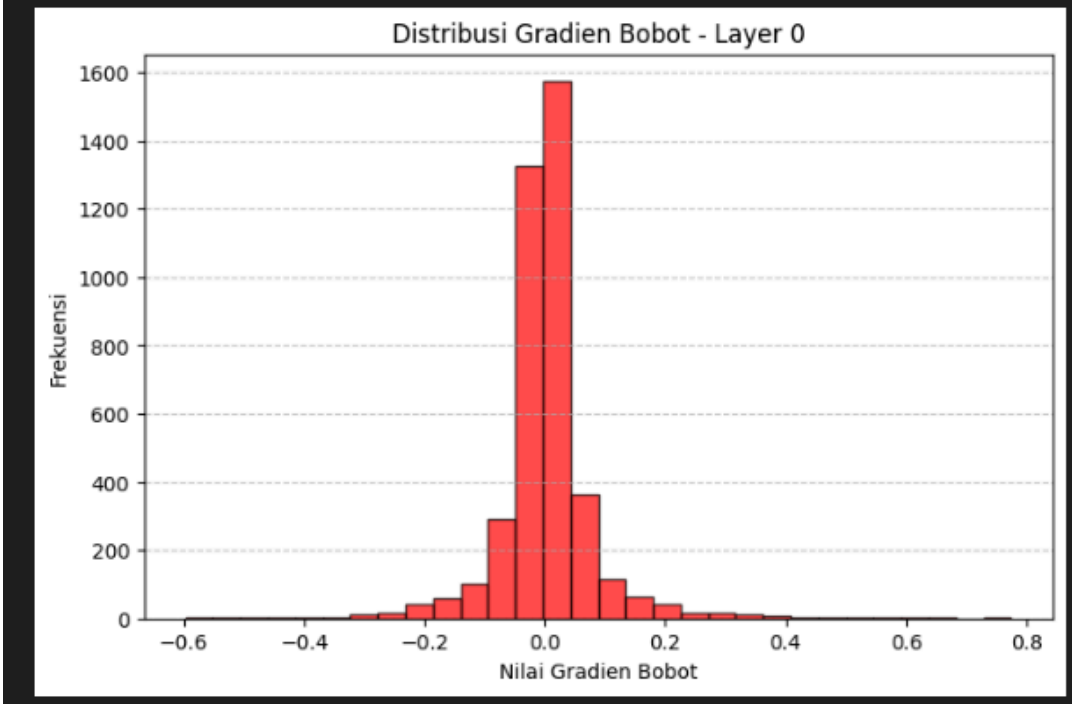


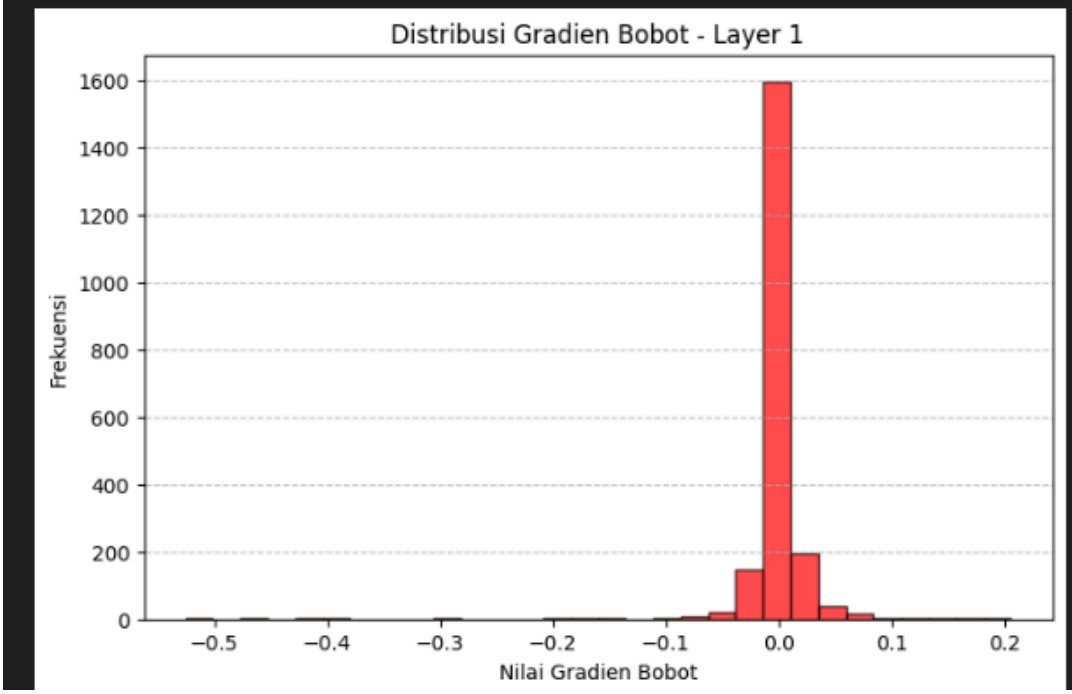
b. *n = 0.05*

```
[Epoch 1]:     Training Loss: 0.01832665058258044      Validation Loss: 0.9355505105004522     Progress: [ ]
[Epoch 2]:     Training Loss: 0.017851388982702596     Validation Loss: 0.4423243814352992     Progress: [ ]
[Epoch 3]:     Training Loss: 0.016716728498849785     Validation Loss: 0.4902867678351085     Progress: [ ]
[Epoch 4]:     Training Loss: 0.01851389613428372 5    Validation Loss: 0.47169322339967745    Progress: [ ]
[Epoch 5]:     Training Loss: 0.017083698849511175     Validation Loss: 0.4845581246615874     Progress: [ ]
[Epoch 6]:     Training Loss: 0.015617140364123195     Validation Loss: 0.46803318398118027    Progress: [ ]
[Epoch 7]:     Training Loss: 0.017614992329068158     Validation Loss: 0.7197269103432613     Progress: [ ]
[Epoch 8]:     Training Loss: 0.015680575799493068     Validation Loss: 0.3938348212684547     Progress: [ ]
[Epoch 9]:     Training Loss: 0.014738183601505568     Validation Loss: 0.4439440993731882     Progress: [ ]
[Epoch 10]:    Training Loss: 0.014385324503608401     Validation Loss: 0.4113154944885209     Progress: [ ]
[Epoch 11]:    Training Loss: 0.014070940247219764     Validation Loss: 0.5178644051560229     Progress: [ ]
[Epoch 12]:    Training Loss: 0.014198887477543743     Validation Loss: 0.3992493118699667     Progress: [ ]
[Epoch 13]:    Training Loss: 0.017551221613603115     Validation Loss: 0.5771117326263716     Progress: [ ]
[Epoch 14]:    Training Loss: 0.017364680948907966     Validation Loss: 0.6946592696776442     Progress: [ ]
[Epoch 15]:    Training Loss: 0.016137225171160167     Validation Loss: 0.4294976884257166     Progress: [ ]
[Epoch 16]:    Training Loss: 0.017100980172341823     Validation Loss: 0.502241301415287      Progress: [ ]
[Epoch 17]:    Training Loss: 0.017442380673602247     Validation Loss: 0.5306895411348346     Progress: [ ]
[Epoch 18]:    Training Loss: 0.01546658656789861      Validation Loss: 0.6325049842084443     Progress: [ ]
[Epoch 19]:    Training Loss: 0.014613712925805641     Validation Loss: 0.42841690985949227    Progress: [ ]
[Epoch 20]:    Training Loss: 0.013984213292798817     Validation Loss: 0.44350675692620095    Progress: [ ]
File Successfuly Saved to: 'models/lr1_model'
```


Distribusi Bobot - Layer 0
Frekuensi
Nilai Bobot
50000
40000
30000
20000
10000
0
−300
−200
−100
0
100
200
300

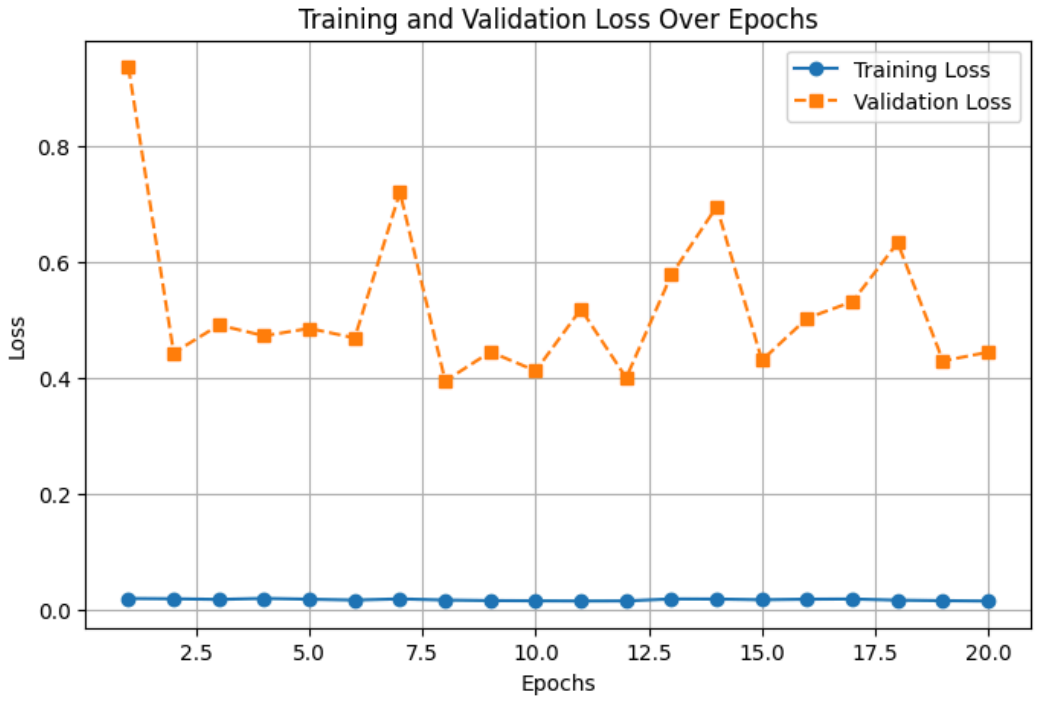

Distribusi Bobot - Layer 1
Frekuensi
Nilai Bobot
1000
800
600
400
200
0
−3
−2
−1
0
1
2
3
4

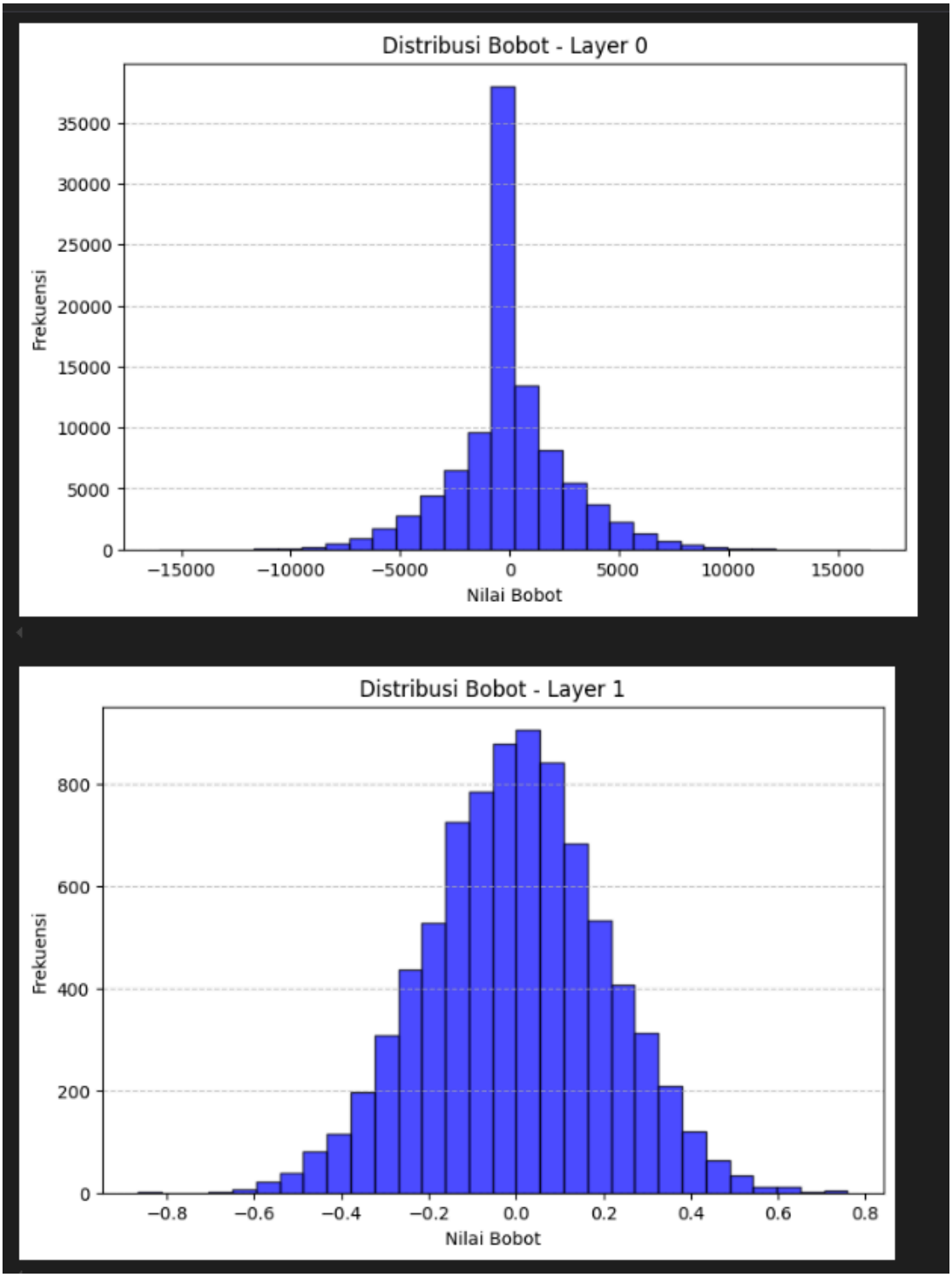

Distribusi Gradien Bobot - Layer 0
Frekuensi
Nilai Gradien Bobot
0
200
400
600
800
1000
1200
1400
1600
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8

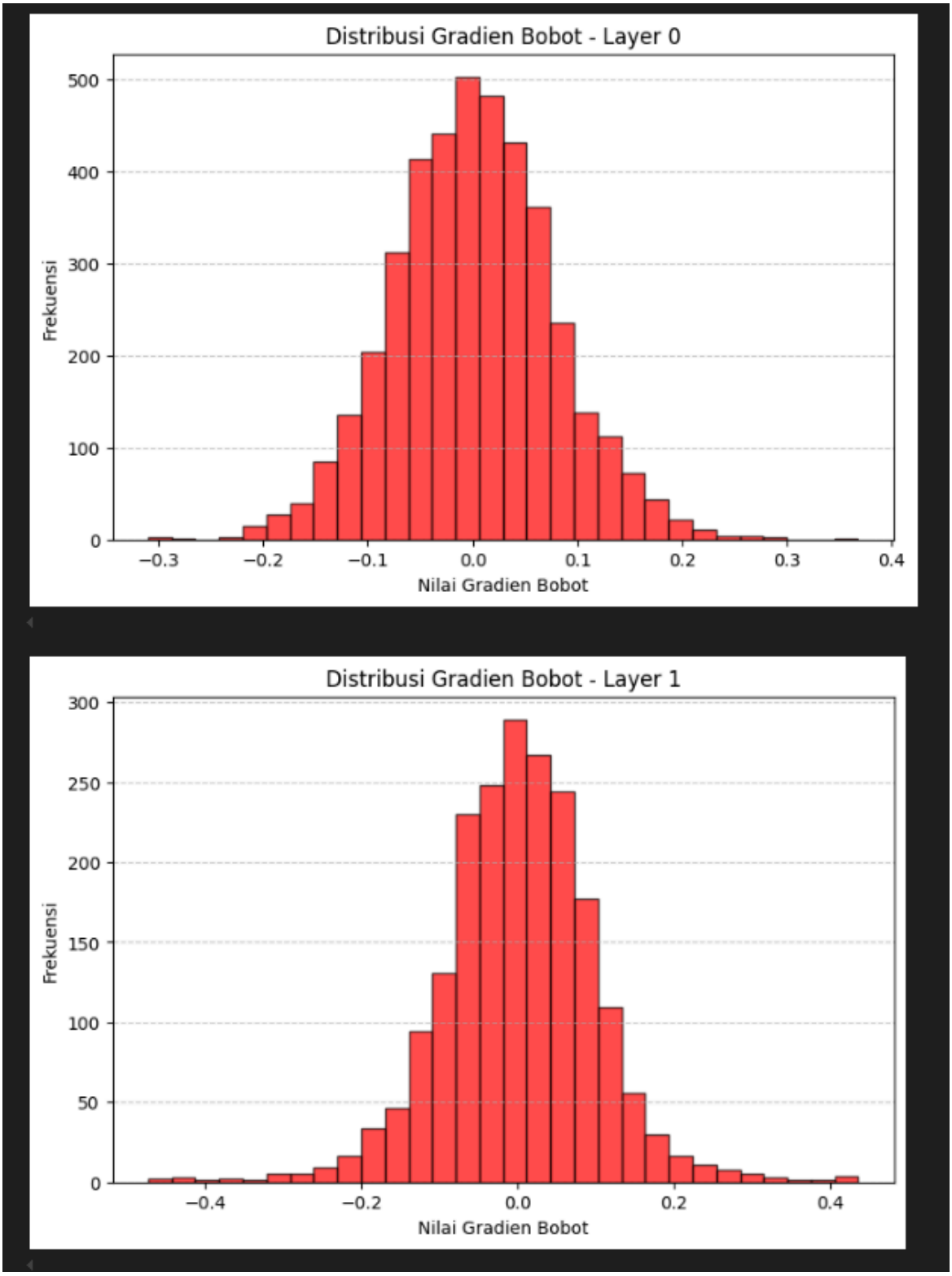

Distribusi Gradien Bobot - Layer 1
Frekuensi
Nilai Gradien Bobot
0
200
400
600
800
1000
1200
1400
1600
−0.5
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2

```
[Epoch 1]:
Training Loss: 0.01832665058258044
Validation Loss: 0.9355505105004522

[Epoch 2]:
Training Loss: 0.017851388982702596
Validation Loss: 0.4423243814352992

[Epoch 3]:
Training Loss: 0.016716728498849785
Validation Loss: 0.4902867678351085

[Epoch 4]:
Training Loss: 0.018513896134283725
Validation Loss: 0.47169322339967745

[Epoch 5]:
Training Loss: 0.017083698849511175
Validation Loss: 0.4845581246615874

[Epoch 6]:
Training Loss: 0.015617140364123195
Validation Loss: 0.46803318398118027

[Epoch 7]:
Training Loss: 0.017614992329068158
Validation Loss: 0.7197269103432613

[Epoch 8]:
Training Loss: 0.015680575799493068
Validation Loss: 0.3938348212684547

[Epoch 9]:
Training Loss: 0.014738183601505568
Validation Loss: 0.4439440993731882

[Epoch 10]:
Training Loss: 0.014385324503608401
Validation Loss: 0.4113154944885209

[Epoch 11]:
Training Loss: 0.014070940247219764
Validation Loss: 0.5178644051560229

[Epoch 12]:
Training Loss: 0.014198887477543743
Validation Loss: 0.3992493118699667

[Epoch 13]:
Training Loss: 0.017551221613603115
Validation Loss: 0.5771117326263716

[Epoch 14]:
```

```
Training Loss: 0.017364680948907966
Validation Loss: 0.6946592696776442

[Epoch 15]:
Training Loss: 0.016137225171160167
Validation Loss: 0.4294976884257166

[Epoch 16]:
Training Loss: 0.017100980172341823
Validation Loss: 0.502241301415287

[Epoch 17]:
Training Loss: 0.017442380673602247
Validation Loss: 0.5306895411348346

[Epoch 18]:
Training Loss: 0.01546658656789861
Validation Loss: 0.6325049842084443

[Epoch 19]:
Training Loss: 0.014613712925805641
Validation Loss: 0.42841690985949227

[Epoch 20]:
Training Loss: 0.013984213292798817
Validation Loss: 0.44350675692620095
```

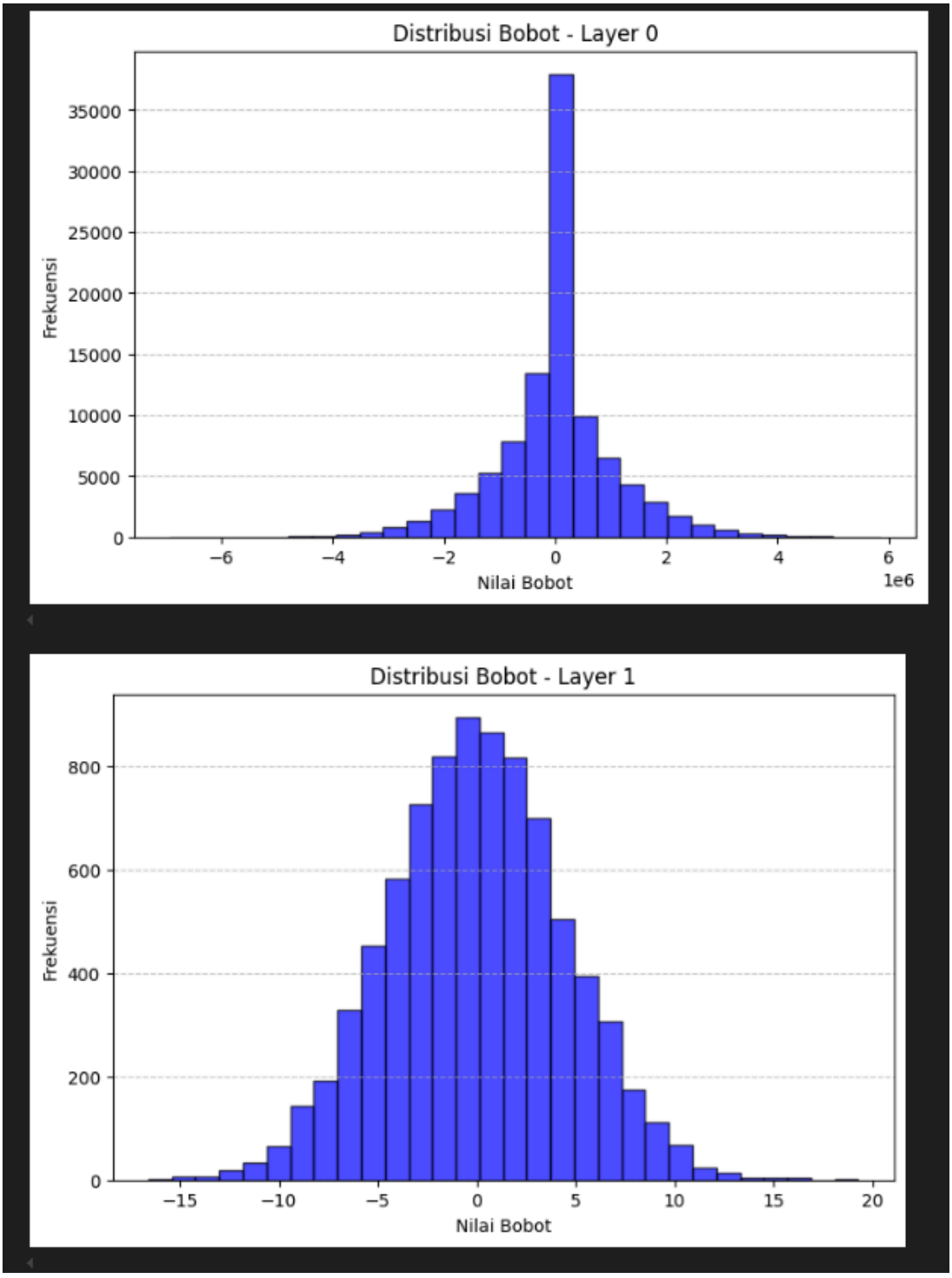


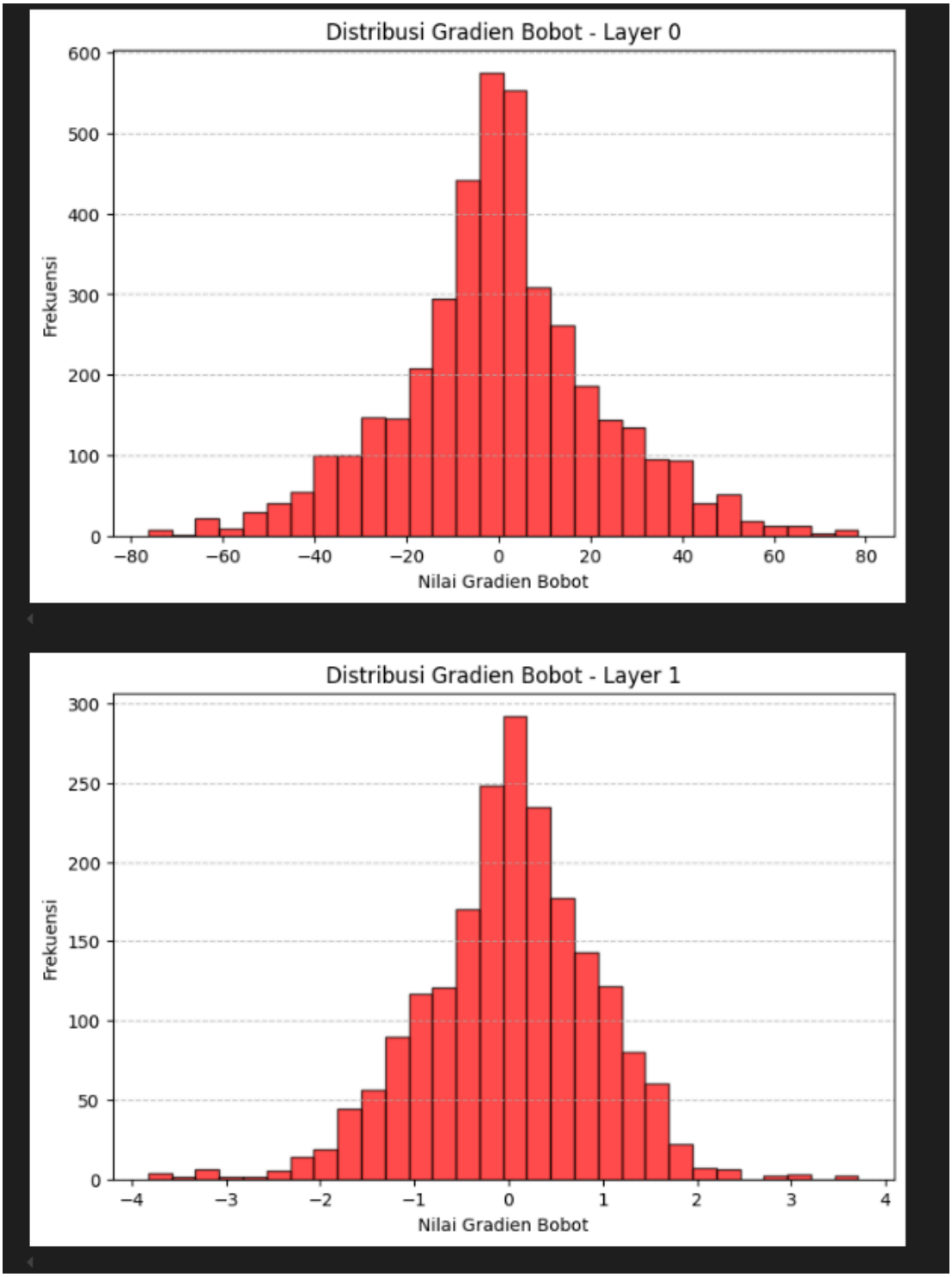

Distribusi Bobot - Layer 0
Frekuensi
Nilai Bobot
0
5000
10000
15000
20000
25000
30000
35000
−15000
−10000
−5000
0
5000
10000
15000
Distribusi Bobot - Layer 1
Frekuensi
Nilai Bobot
0
200
400
600
800
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8


Distribusi Gradien Bobot - Layer 0
Frekuensi
Nilai Gradien Bobot
500
400
300
200
100
0
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
Distribusi Gradien Bobot - Layer 1
Frekuensi
Nilai Gradien Bobot
300
250
200
150
100
50
0
−0.4
−0.2
0.0
0.2
0.4

c. *n = 0.20*

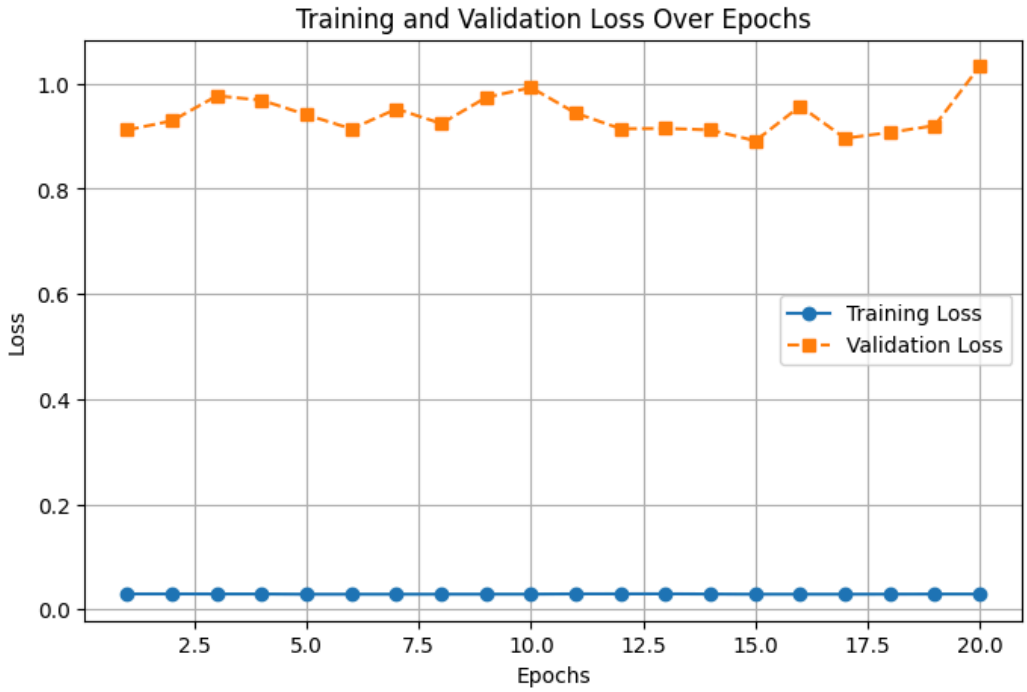

[Epoch 1]: Training Loss: 0.029520046810296106 Validation Loss: 0.9118799512386093 Progress: [
[Epoch 2]: Training Loss: 0.029587138474194124 Validation Loss: 0.92943932306647 Progress: [
[Epoch 3]: Training Loss: 0.029477668924649262 Validation Loss: 0.9768449224212542 Progress: [
[Epoch 4]: Training Loss: 0.02944238895803331 Validation Loss: 0.96838829862165 Progress: [
[Epoch 5]: Training Loss: 0.028978667257752687 Validation Loss: 0.9407026219885027 Progress: [
[Epoch 6]: Training Loss: 0.028997464839148602 Validation Loss: 0.9138975698882214 Progress: [
[Epoch 7]: Training Loss: 0.029012181631419835 Validation Loss: 0.9519523170330418 Progress: [
[Epoch 8]: Training Loss: 0.02904425911967591 Validation Loss: 0.9240890309803287 Progress: [
[Epoch 9]: Training Loss: 0.029103460341975206 Validation Loss: 0.9738196513793832 Progress: [
[Epoch 10]: Training Loss: 0.029146914122496474 Validation Loss: 0.992517072324671 Progress: [
[Epoch 11]: Training Loss: 0.029662836537782056 Validation Loss: 0.9437444169218517 Progress: [
[Epoch 12]: Training Loss: 0.029699305106165804 Validation Loss: 0.9141799837167444 Progress: [
[Epoch 13]: Training Loss: 0.029738996961922774 Validation Loss: 0.9148657842378528 Progress: [
[Epoch 14]: Training Loss: 0.02935744649787664 Validation Loss: 0.9119214795687495 Progress: [
[Epoch 15]: Training Loss: 0.02898032610649609 Validation Loss: 0.8912020016075614 Progress: [
[Epoch 16]: Training Loss: 0.029022988098566238 Validation Loss: 0.957144626941544 Progress: [
[Epoch 17]: Training Loss: 0.029026766340359396 Validation Loss: 0.8961599094894162 Progress: [
[Epoch 18]: Training Loss: 0.02912841942027166 Validation Loss: 0.9072787046743026 Progress: [
[Epoch 19]: Training Loss: 0.029344102481573632 Validation Loss: 0.9202675696502645 Progress: [
[Epoch 20]: Training Loss: 0.029295171497072368 Validation Loss: 1.0330938453297906 Progress: [
File Successfuly Saved to: 'models/lr2_model'


Distribusi Bobot - Layer 0
Frekuensi
35000
30000
25000
20000
15000
10000
5000
0
−6
−4
−2
0
2
4
6
1e6
Nilai Bobot
Distribusi Bobot - Layer 1
Frekuensi
800
600
400
200
0
−15
−10
−5
0
5
10
15
20
Nilai Bobot


Distribusi Gradien Bobot - Layer 0
Frekuensi
Nilai Gradien Bobot
0
100
200
300
400
500
600
−80
−60
−40
−20
0
20
40
60
80
Distribusi Gradien Bobot - Layer 1
Frekuensi
Nilai Gradien Bobot
0
50
100
150
200
250
300
−4
−3
−2
−1
0
1
2
3
4

```
[Epoch 1]:
Training Loss: 0.029520046810296106
Validation Loss: 0.9118799512386093

[Epoch 2]:
Training Loss: 0.029587138474194124
Validation Loss: 0.92943932306647

[Epoch 3]:
Training Loss: 0.029477668924649262
Validation Loss: 0.9768449224212542

[Epoch 4]:
Training Loss: 0.02944238895803331
Validation Loss: 0.96838829862165

[Epoch 5]:
Training Loss: 0.028978667257752687
Validation Loss: 0.9407026219885027

[Epoch 6]:
Training Loss: 0.028997464839148602
Validation Loss: 0.9138975698882214

[Epoch 7]:
Training Loss: 0.029012181631419835
Validation Loss: 0.9519523170330418

[Epoch 8]:
Training Loss: 0.02904425911967591
Validation Loss: 0.9240890309803287

[Epoch 9]:
Training Loss: 0.029103460341975206
Validation Loss: 0.9738196513793832

[Epoch 10]:
Training Loss: 0.029146914122496474
Validation Loss: 0.992517072324671

[Epoch 11]:
Training Loss: 0.029662836537782056
Validation Loss: 0.9437444169218517

[Epoch 12]:
Training Loss: 0.029699305106165804
Validation Loss: 0.9141799837167444

[Epoch 13]:
Training Loss: 0.029738996961922774
Validation Loss: 0.9148657842378528

[Epoch 14]:
```

```
Training Loss: 0.02935744649787664
Validation Loss: 0.9119214795687495

[Epoch 15]:
Training Loss: 0.02898032610649609
Validation Loss: 0.8912020016075614

[Epoch 16]:
Training Loss: 0.029022988098566238
Validation Loss: 0.957144626941544

[Epoch 17]:
Training Loss: 0.029026766340359396
Validation Loss: 0.8961599094894162

[Epoch 18]:
Training Loss: 0.02912841942027166
Validation Loss: 0.9072787046743026

[Epoch 19]:
Training Loss: 0.029344102481573632
Validation Loss: 0.9202675696502645

[Epoch 20]:
Training Loss: 0.029295171497072368
Validation Loss: 1.0330938453297906
```



d. Perbandingan

Hasil dari variasi *learning rate* menunjukkan bahwa parameter ini sangat mempengaruhi hasil kinerja model. Terdapat empat nilai *learning rate* yang dapat dibandingkan, yaitu *base model (0.01),* 0.001, 0.05, dan 0.02.

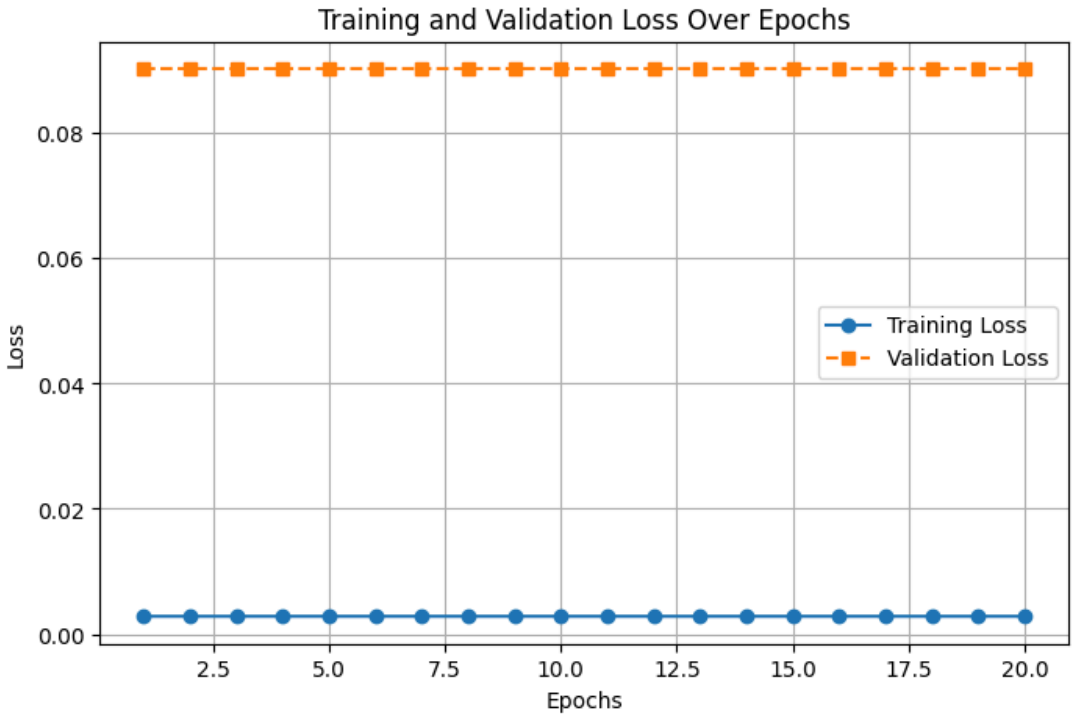
- Untuk *base model* didapatkan kinerja yang cukup bagus untuk nilai *learning rate* 0.01, yaitu mendapatkan sekitar 0.3 untuk *loss*.
- Untuk *learning rate* lebih kecil terlihat pada awalnya meningkatkan kinerja model, dengan model 0.001 langsung menghasilkan *loss* 0.37. Namun setelah beberapa *epoch* terlihat bahwa model ini memiliki penurunan yang jauh lebih lambat dibanding model 0.01.
- Ketika *learning rate* diperbesar, terlihat bahwa kinerja model menurun drastis. Untuk model 0.05, masih terlihat menurun, namun kinerjanya sangat tidak konsisten dan nilai *loss* sering naik.
- Untuk model 0.2, terlihat kinerja cukup buruk, nilai di atas 1.0 untuk *loss* dan tidak terlihat menurun.

F. Hasil variasi *weight initialization*

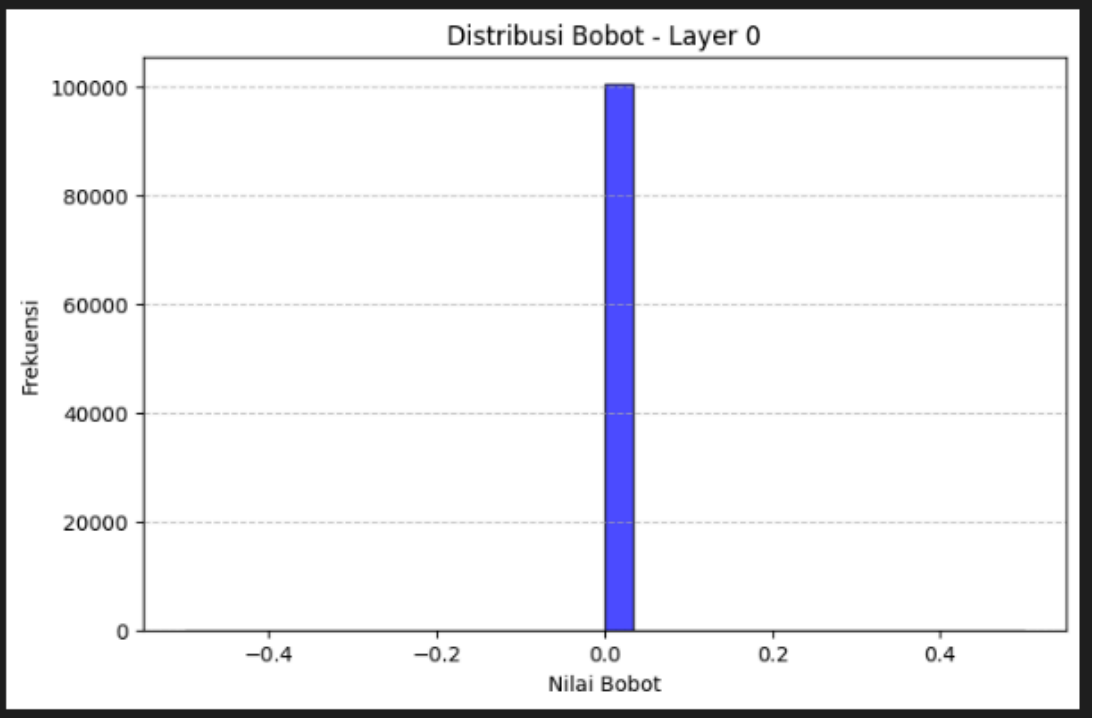
a. *Zero Weight Initialization*

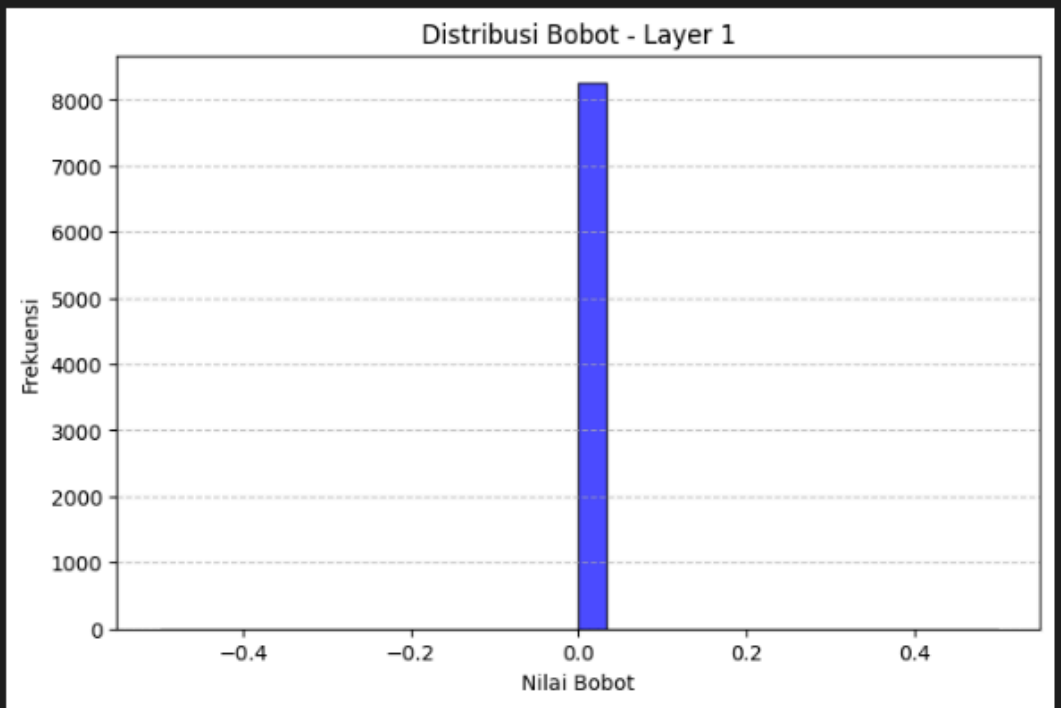
```
[Epoch 1]:
Training Loss: 0.002827781248851372
Validation Loss: 0.09024793327689644

[Epoch 2]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 3]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 4]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 5]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 6]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 7]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 8]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 9]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 10]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 11]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 12]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 13]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 14]:
```

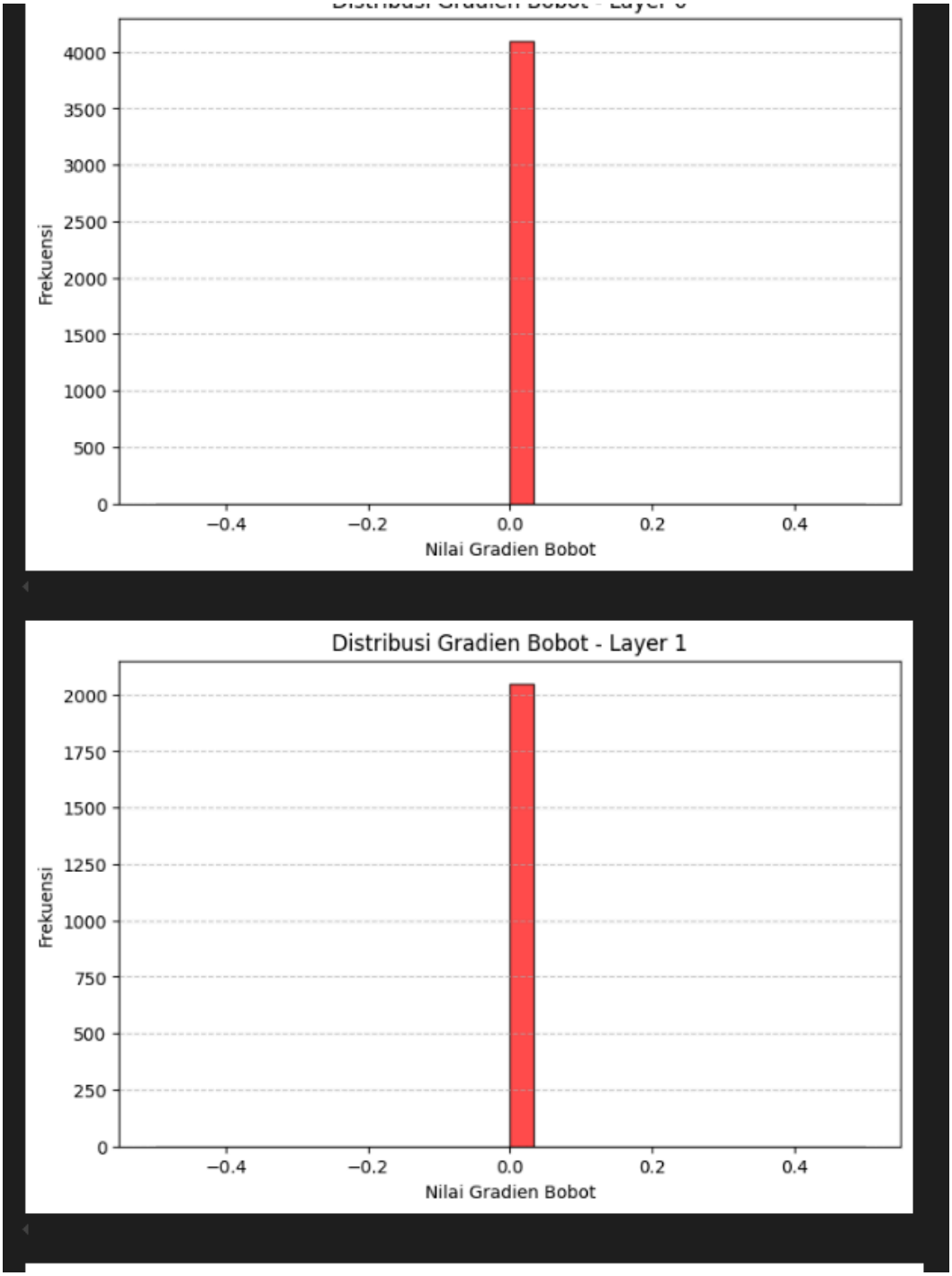
```
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 15]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 16]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 17]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 18]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 19]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644

[Epoch 20]:
Training Loss: 0.002827445735569724
Validation Loss: 0.09024793327689644
```



✓ 1.2s


Distribusi Bobot - Layer 0
Frekuensi
100000
80000
60000
40000
20000
0
−0.4
−0.2
0.0
0.2
0.4
Nilai Bobot


Distribusi Bobot - Layer 1
Frekuensi
8000
7000
6000
5000
4000
3000
2000
1000
0
−0.4
−0.2
0.0
0.2
0.4
Nilai Bobot



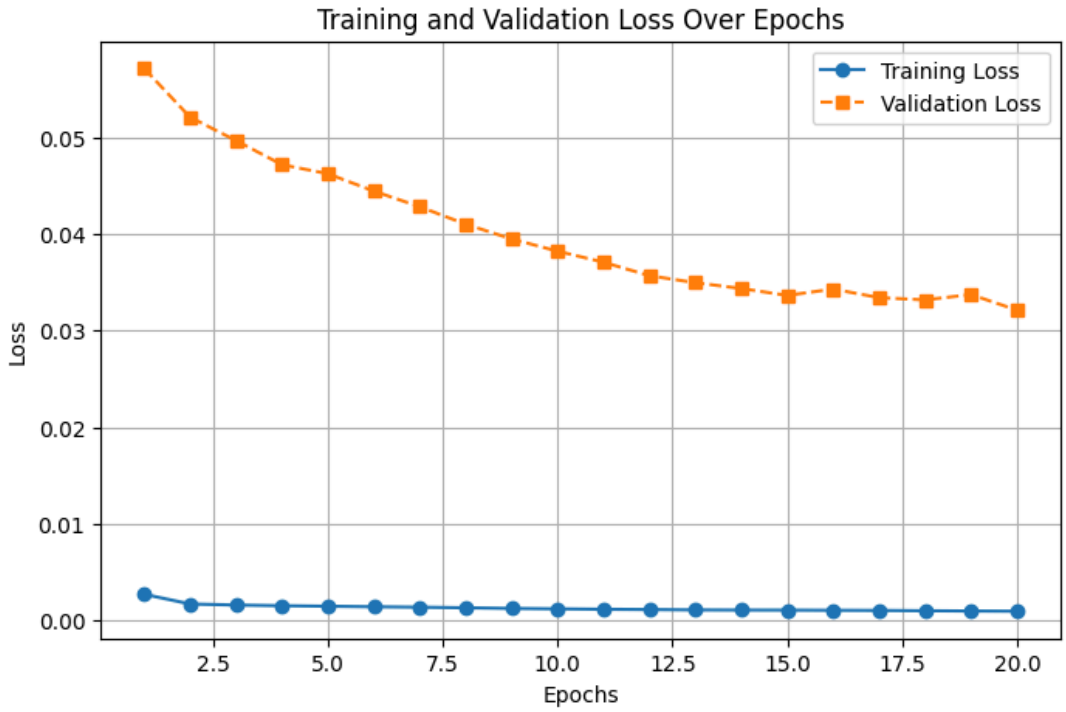
b. *Uniform Distribution Initialization*

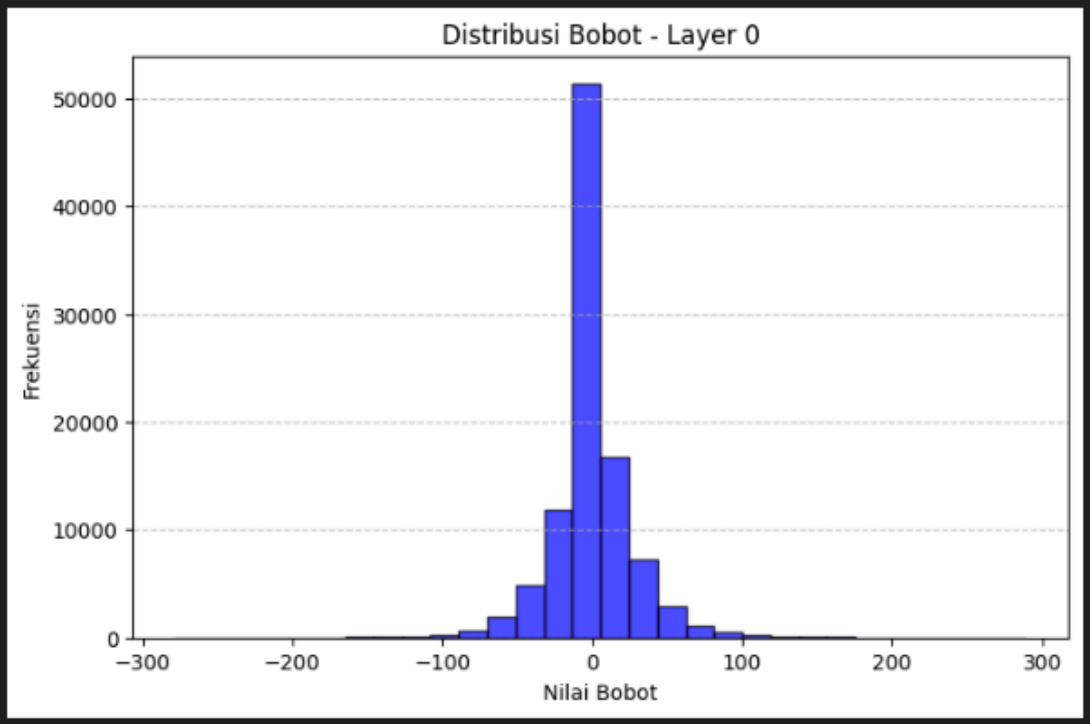
Untuk variasi ini menggunakan model yang sama dengan *base model*.

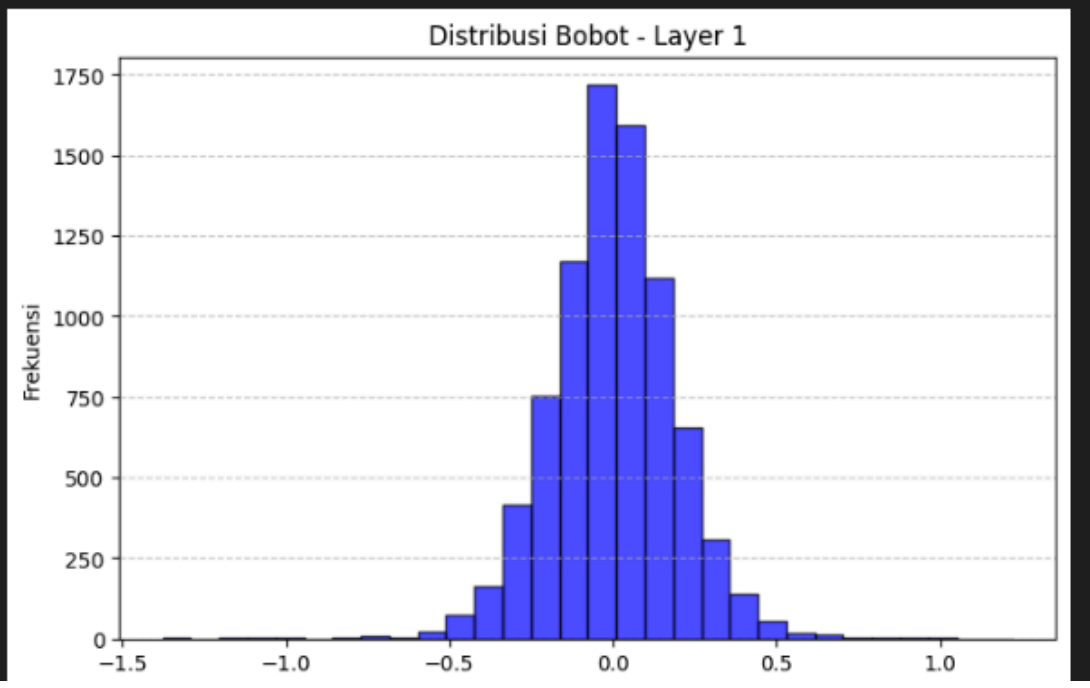

[Epoch 1]: Training Loss: 0.0027099407614237245 Validation Loss: 0.0571544548208529 Progress: [
[Epoch 2]: Training Loss: 0.001706889483943063 Validation Loss: 0.05210085521640847 Progress: [
[Epoch 3]: Training Loss: 0.0015966776277596892 Validation Loss: 0.04967947677031759 Progress: [
[Epoch 4]: Training Loss: 0.001524799263901508 Validation Loss: 0.0471801249487014 Progress: [
[Epoch 5]: Training Loss: 0.0014714891992633086 Validation Loss: 0.04624765845043547 Progress: [
[Epoch 6]: Training Loss: 0.0014176629959564682 Validation Loss: 0.04444701174706174 Progress: [
[Epoch 7]: Training Loss: 0.001373547490030896 Validation Loss: 0.04282773304071671 Progress: [
[Epoch 8]: Training Loss: 0.0013135754875201009 Validation Loss: 0.04101992988362089 Progress: [
[Epoch 9]: Training Loss: 0.001244359535475869 Validation Loss: 0.039493487700896 Progress: [
[Epoch 10]: Training Loss: 0.0012005088441360318 Validation Loss: 0.03822774648147555 Progress: [
[Epoch 11]: Training Loss: 0.001156201363402212 Validation Loss: 0.03708439868191913 Progress: [
[Epoch 12]: Training Loss: 0.0011292588930959223 Validation Loss: 0.035688982256479096 Progress: [
[Epoch 13]: Training Loss: 0.0010893803837048834 Validation Loss: 0.034964862608091944 Progress: [
[Epoch 14]: Training Loss: 0.0010757887409412609 Validation Loss: 0.034361206855257856 Progress: [
[Epoch 15]: Training Loss: 0.0010693487356087789 Validation Loss: 0.03364834620603709 Progress: [
[Epoch 16]: Training Loss: 0.001051777246899769 Validation Loss: 0.03430353493157926 Progress: [
[Epoch 17]: Training Loss: 0.0010368413570126464 Validation Loss: 0.033415385826872104 Progress: [
[Epoch 18]: Training Loss: 0.0010054481736810218 Validation Loss: 0.03318320388133522 Progress: [
[Epoch 19]: Training Loss: 0.0009769407626764707 Validation Loss: 0.033735979472816105 Progress: [
[Epoch 20]: Training Loss: 0.0009562444742805288 Validation Loss: 0.03212976849743355 Progress: [

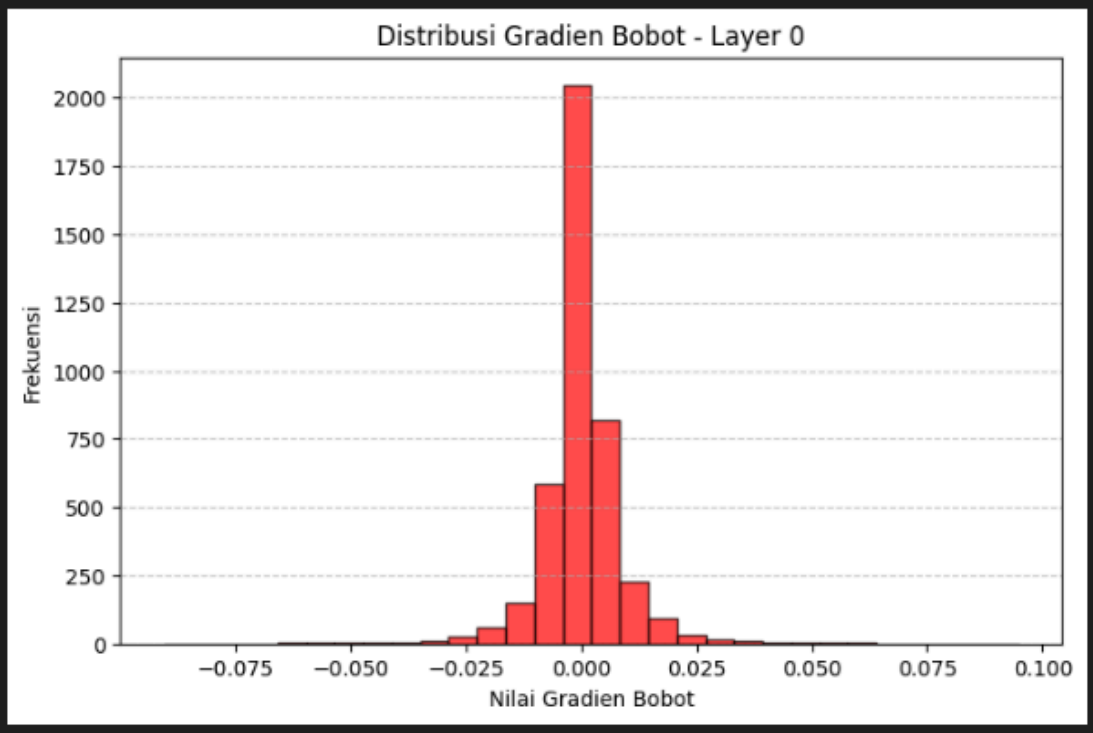
```
[Epoch 1]:
Training Loss: 0.0027099407614237245
Validation Loss: 0.0571544548208529

[Epoch 2]:
Training Loss: 0.001706889483943063
Validation Loss: 0.05210085521640847

[Epoch 3]:
Training Loss: 0.0015966776277596892
Validation Loss: 0.04967947677031759

[Epoch 4]:
Training Loss: 0.001524799263901508
Validation Loss: 0.04718012494877014

[Epoch 5]:
Training Loss: 0.0014714891992633086
Validation Loss: 0.04624765845043547

[Epoch 6]:
Training Loss: 0.0014176629959564682
Validation Loss: 0.04444701174706174

[Epoch 7]:
Training Loss: 0.001373547490030896
Validation Loss: 0.04282773304071671

[Epoch 8]:
Training Loss: 0.0013135754875201009
Validation Loss: 0.04101992988362089

[Epoch 9]:
Training Loss: 0.001244359535475869
Validation Loss: 0.039493487700896

[Epoch 10]:
Training Loss: 0.0012005088441360318
Validation Loss: 0.03822774648147555

[Epoch 11]:
Training Loss: 0.001156201363402212
Validation Loss: 0.03708439868191913

[Epoch 12]:
Training Loss: 0.0011292588930959223
Validation Loss: 0.035688982256479096

[Epoch 13]:
Training Loss: 0.0010893803837048834
Validation Loss: 0.034964862608091944

[Epoch 14]:
```

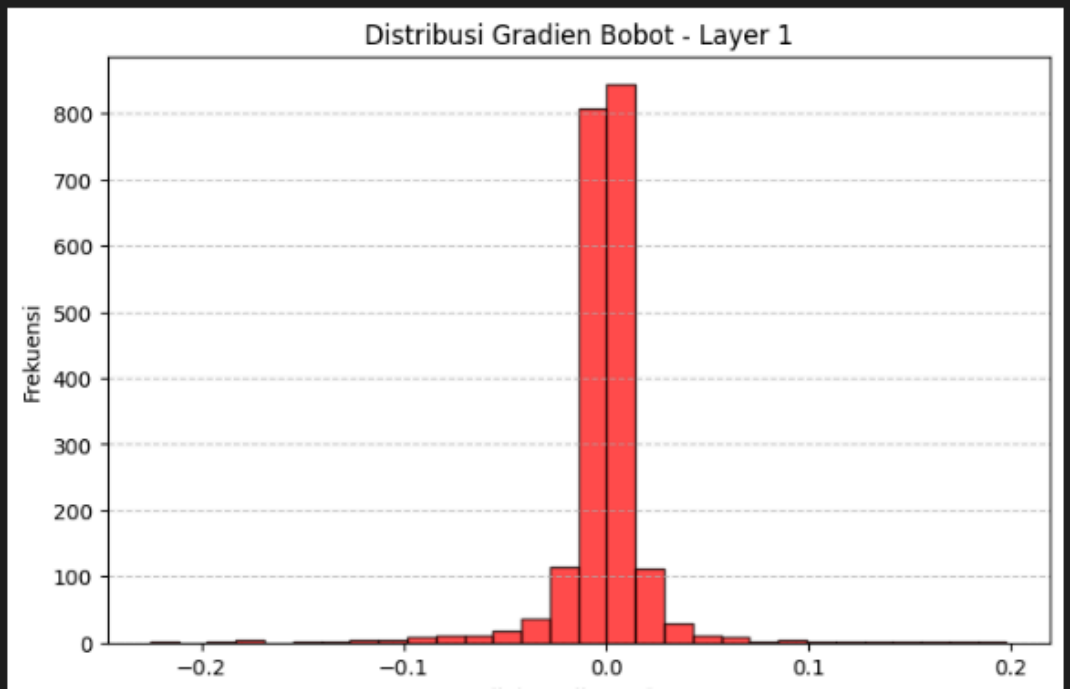
```
Training Loss: 0.0010757887409412609
Validation Loss: 0.034361206855257856

[Epoch 15]:
Training Loss: 0.0010693487356087789
Validation Loss: 0.03364834620603709

[Epoch 16]:
Training Loss: 0.001051777246899769
Validation Loss: 0.03430353493157926

[Epoch 17]:
Training Loss: 0.0010368413570126464
Validation Loss: 0.033415385826872104

[Epoch 18]:
Training Loss: 0.0010054481736810218
Validation Loss: 0.03318320388133522

[Epoch 19]:
Training Loss: 0.0009769407626764707
Validation Loss: 0.033735979472816105

[Epoch 20]:
Training Loss: 0.0009562444742805288
Validation Loss: 0.03212976849743355
```




Distribusi Bobot - Layer 0
Frekuensi
Nilai Bobot
50000
40000
30000
20000
10000
0
−300
−200
−100
0
100
200
300


Distribusi Bobot - Layer 1
Frekuensi
1750
1500
1250
1000
750
500
250
0
−1.5
−1.0
−0.5
0.0
0.5
1.0

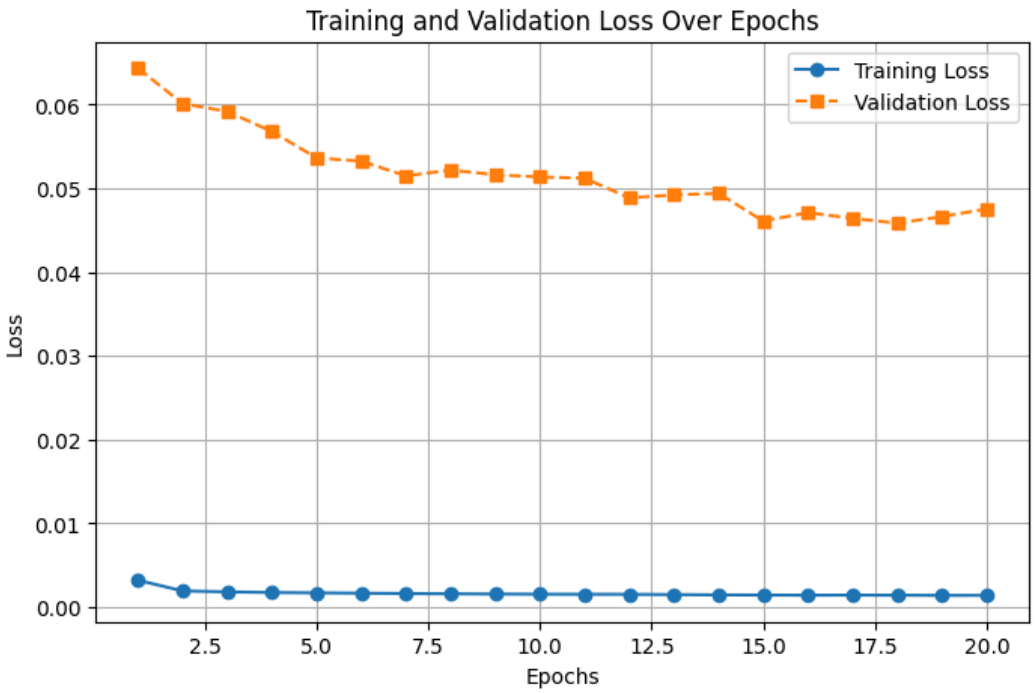

Distribusi Gradien Bobot - Layer 0
Frekuensi
2000
1750
1500
1250
1000
750
500
250
0
−0.075
−0.050
−0.025
0.000
0.025
0.050
0.075
0.100
Nilai Gradien Bobot

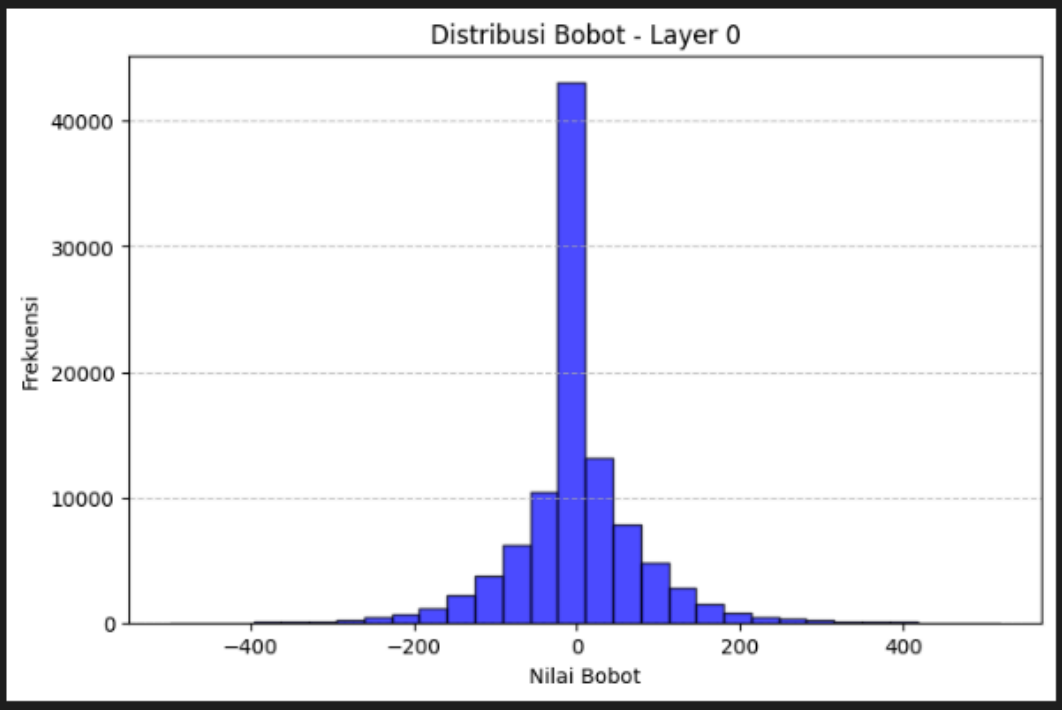

Distribusi Gradien Bobot - Layer 1
Frekuensi
800
700
600
500
400
300
200
100
0
−0.2
−0.1
0.0
0.1
0.2

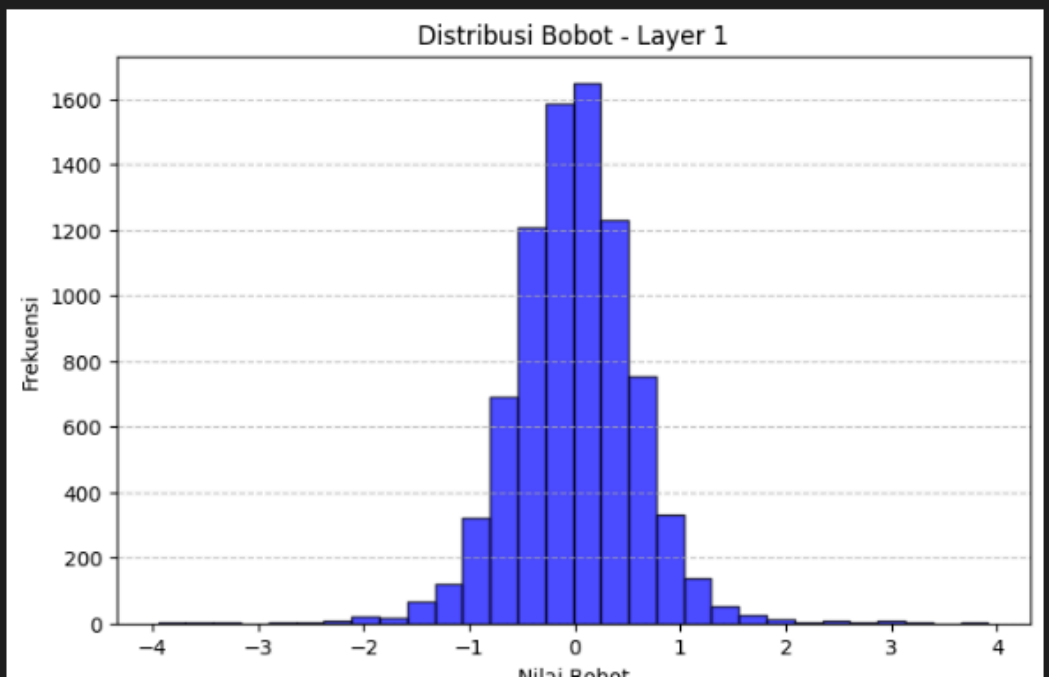
c. *Normal Distribution Initialization*

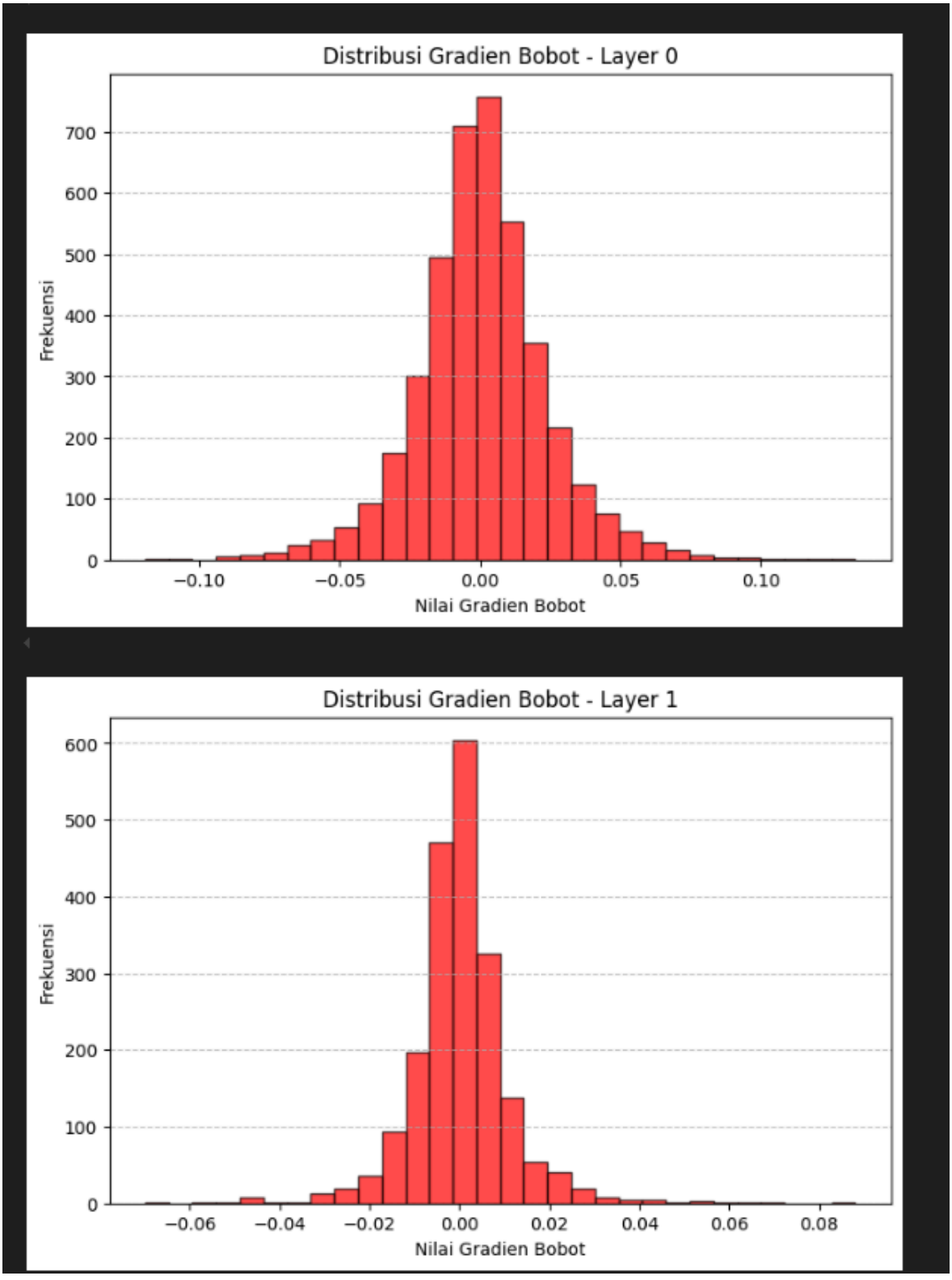

```
[Epoch 1]:
Training Loss: 0.003227543814136077
Validation Loss: 0.06436208008585514

[Epoch 2]:
Training Loss: 0.0019487975811172597
Validation Loss: 0.060138432774492115

[Epoch 3]:
Training Loss: 0.0018338003638948591
Validation Loss: 0.05918530001053791

[Epoch 4]:
Training Loss: 0.001762201944007271
Validation Loss: 0.05680446560003757

[Epoch 5]:
Training Loss: 0.0017202945959709987
Validation Loss: 0.05364807265487833

[Epoch 6]:
Training Loss: 0.0016740236533634419
Validation Loss: 0.05325718835678208

[Epoch 7]:
Training Loss: 0.0016366170420036511
Validation Loss: 0.05149971201804255

[Epoch 8]:
Training Loss: 0.0016085555944471974
Validation Loss: 0.05219497341825922

[Epoch 9]:
Training Loss: 0.0015792894873998104
Validation Loss: 0.05162880807864436

[Epoch 10]:
Training Loss: 0.0015563972040652538
```

```
Validation Loss: 0.05136510101486026

[Epoch 11]:
Training Loss: 0.001545981832794515
Validation Loss: 0.051224429788660306

[Epoch 12]:
Training Loss: 0.0015395766071034588
Validation Loss: 0.04887350044288162

[Epoch 13]:
Training Loss: 0.0015049978093172795
Validation Loss: 0.049228769389016186

[Epoch 14]:
Training Loss: 0.0014703659530307441
Validation Loss: 0.04940917152189027

[Epoch 15]:
Training Loss: 0.001453965751188711
Validation Loss: 0.04611346925216642

[Epoch 16]:
Training Loss: 0.0014406619757261708
Validation Loss: 0.0471114397875792

[Epoch 17]:
Training Loss: 0.0014481971243004532
Validation Loss: 0.046428756745155876

[Epoch 18]:
Training Loss: 0.0014401047861066906
Validation Loss: 0.04585411803164374

[Epoch 19]:
Training Loss: 0.0014168083882073708
Validation Loss: 0.046637604239895936

[Epoch 20]:
Training Loss: 0.0014190914621719309
Validation Loss: 0.047549579691087446
```


Training and Validation Loss Over Epochs
Training Loss
Validation Loss
Loss
Epochs
0.00
0.01
0.02
0.03
0.04
0.05
0.06
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Distribusi Bobot - Layer 0
Frekuensi
Nilai Bobot
40000
30000
20000
10000
0
−400
−200
0
200
400


Distribusi Bobot - Layer 1
Frekuensi
1600
1400
1200
1000
800
600
400
200
0
−4
−3
−2
−1
0
1
2
3
4


Distribusi Gradien Bobot - Layer 0
Frekuensi
0
100
200
300
400
500
600
700
−0.10
−0.05
0.00
0.05
0.10
Nilai Gradien Bobot
Distribusi Gradien Bobot - Layer 1
Frekuensi
0
100
200
300
400
500
600
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
0.08
Nilai Gradien Bobot

d. *Perbandingan*

- Untuk *zero weight initialization* terlihat bahwa bobot 0 membuat model sangat sulit untuk melakukan pembelajaran
- Untuk model dengan inisialisasi *uniform* terlihat memiliki kinerja paling bagus, memiliki penurunan *loss* yang cukup konsisten.
- Untuk model inisialisasi *normal* terlihat *loss*-nya menurun, namun lebih lambat dibanding model *uniform*.

G. Hasil Variasi Regularisasi

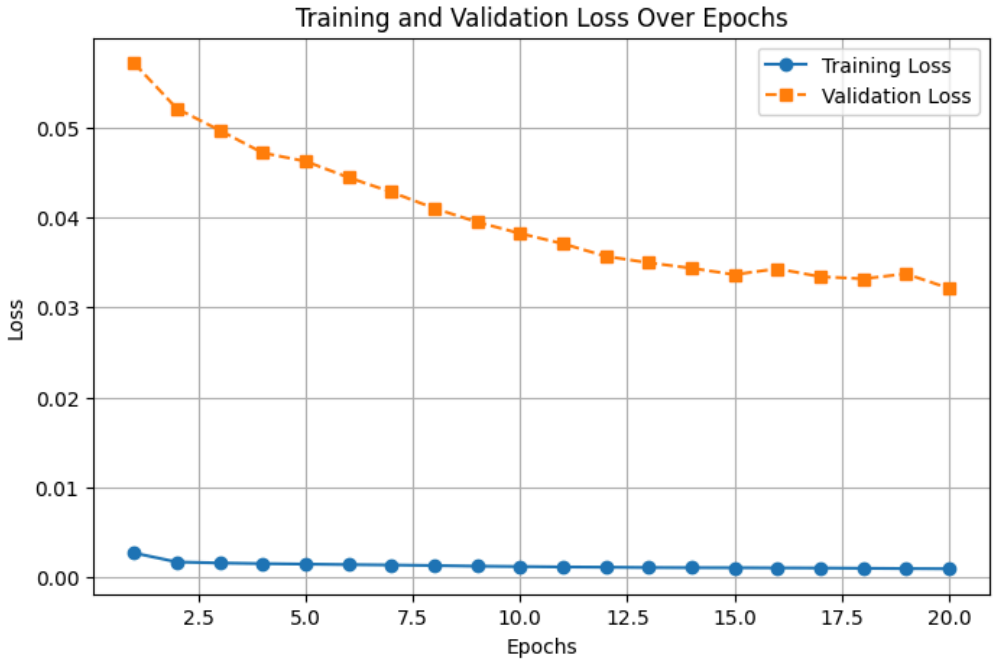
a. *No Regularization*

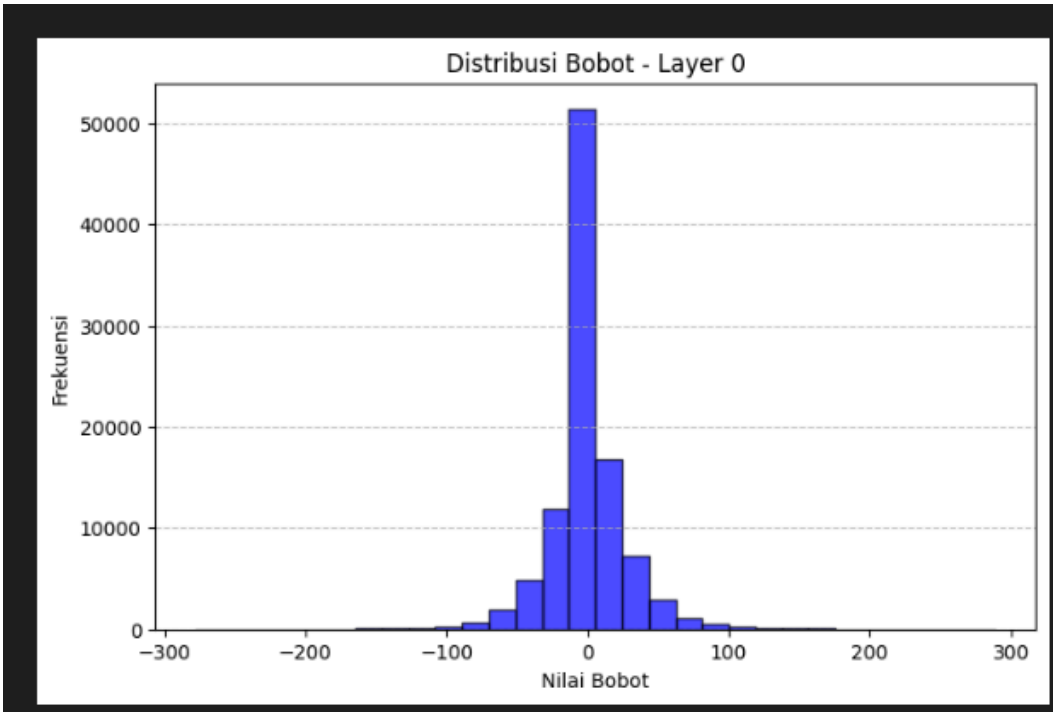
Untuk model ini menggunakan *base model.*

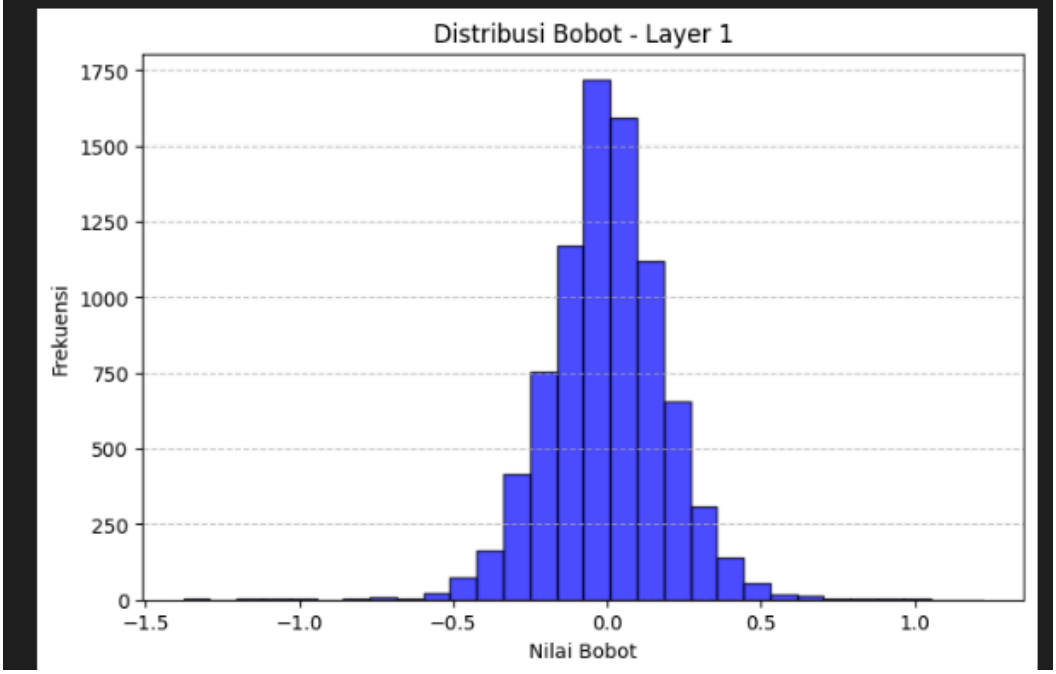

```
[Epoch 1]:
Training Loss: 0.0027099407614237245
Validation Loss: 0.0571544548208529

[Epoch 2]:
Training Loss: 0.0017068894839430 63
Validation Loss: 0.05210085521640847

[Epoch 3]:
Training Loss: 0.0015966776277596892
Validation Loss: 0.04967947677031759

[Epoch 4]:
Training Loss: 0.0015247992639015 08
Validation Loss: 0.04718012494877014

[Epoch 5]:
Training Loss: 0.0014714891992633086
Validation Loss: 0.04624765845043547

[Epoch 6]:
Training Loss: 0.0014176629959564682
Validation Loss: 0.04444701174706174
```

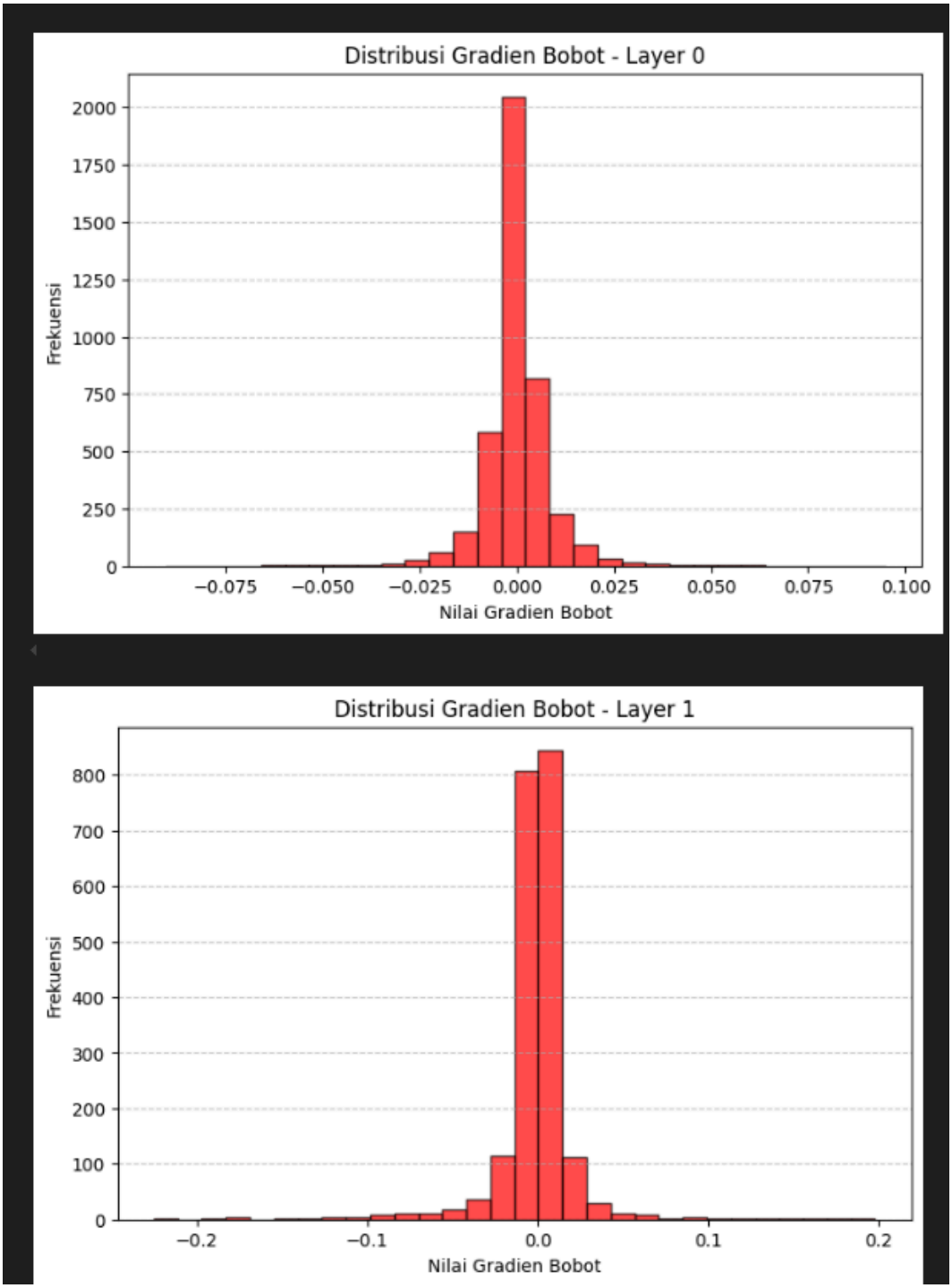
```
[Epoch 7]:
Training Loss: 0.001373547490030896
Validation Loss: 0.04282773304071671

[Epoch 8]:
Training Loss: 0.0013135754875201009
Validation Loss: 0.041019929883620 89

[Epoch 9]:
Training Loss: 0.0012443595354758 69
Validation Loss: 0.0394934877008 96

[Epoch 10]:
Training Loss: 0.0012005088441360318
Validation Loss: 0.03822774648147555

[Epoch 11]:
Training Loss: 0.001156201363402212
Validation Loss: 0.03708439868191913

[Epoch 12]:
Training Loss: 0.0011292588930959223
Validation Loss: 0.035688982256479096

[Epoch 13]:
Training Loss: 0.0010893803837048834
Validation Loss: 0.034964862608091944

[Epoch 14]:
Training Loss: 0.0010757887409412609
Validation Loss: 0.034361206855257856

[Epoch 15]:
Training Loss: 0.0010693487356087789
Validation Loss: 0.03364834620603709

[Epoch 16]:
Training Loss: 0.001051777246899769
Validation Loss: 0.03430353493157926

[Epoch 17]:
Training Loss: 0.0010368413570126464
Validation Loss: 0.033415385826872104

[Epoch 18]:
Training Loss: 0.0010054481736810218
Validation Loss: 0.03318320388133522

[Epoch 19]:
Training Loss: 0.0009769407626764707
Validation Loss: 0.033735979472816105

[Epoch 20]:
```

```
Training Loss: 0.0009562444742805288
Validation Loss: 0.03212976849743355
```



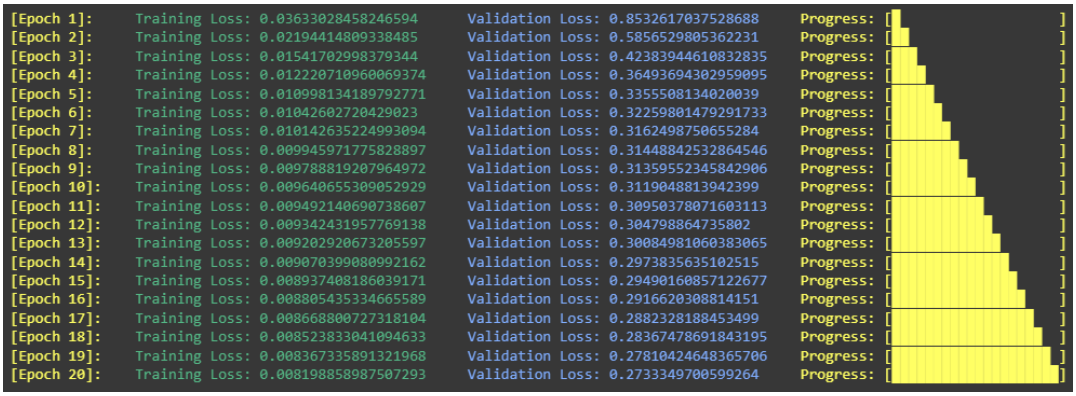

Distribusi Bobot - Layer 0
Frekuensi
Nilai Bobot
50000
40000
30000
20000
10000
0
−300
−200
−100
0
100
200
300


Distribusi Bobot - Layer 1
Frekuensi
Nilai Bobot
1750
1500
1250
1000
750
500
250
0
−1.5
−1.0
−0.5
0.0
0.5
1.0



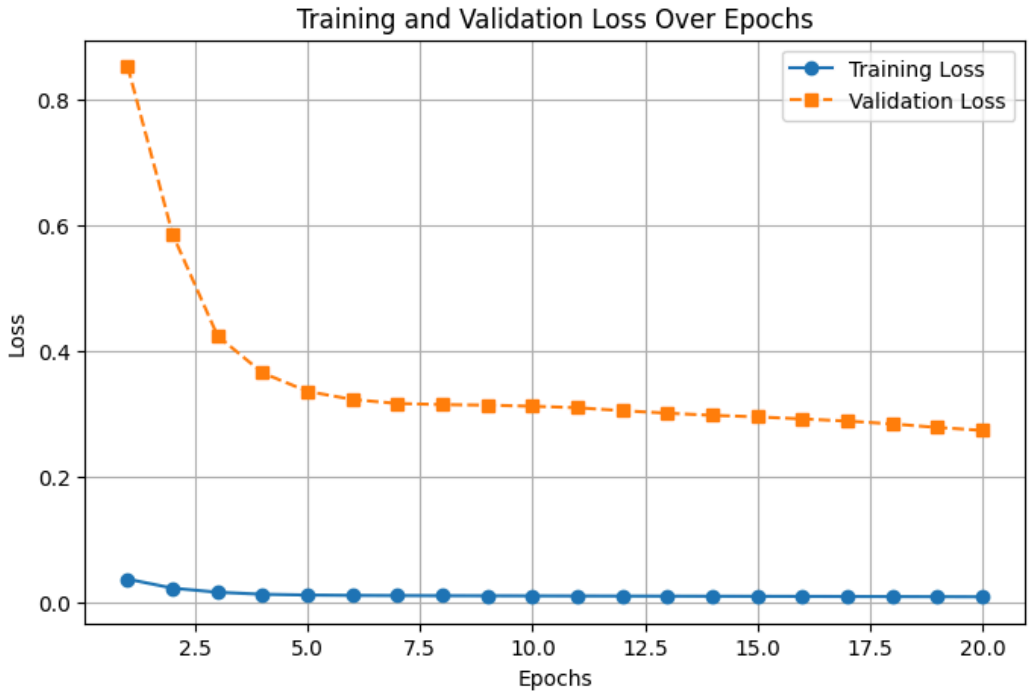
b. L1 Regularization (λ = 0.001)

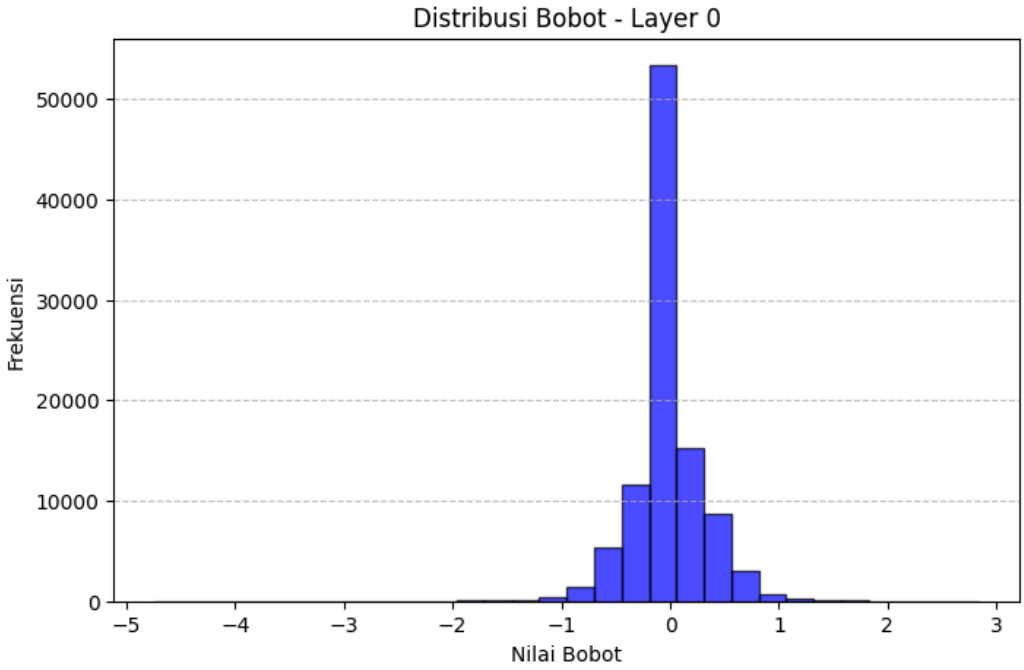
Untuk model ini, nilai *loss* yang ditampilkan adalah nilai aslinya.

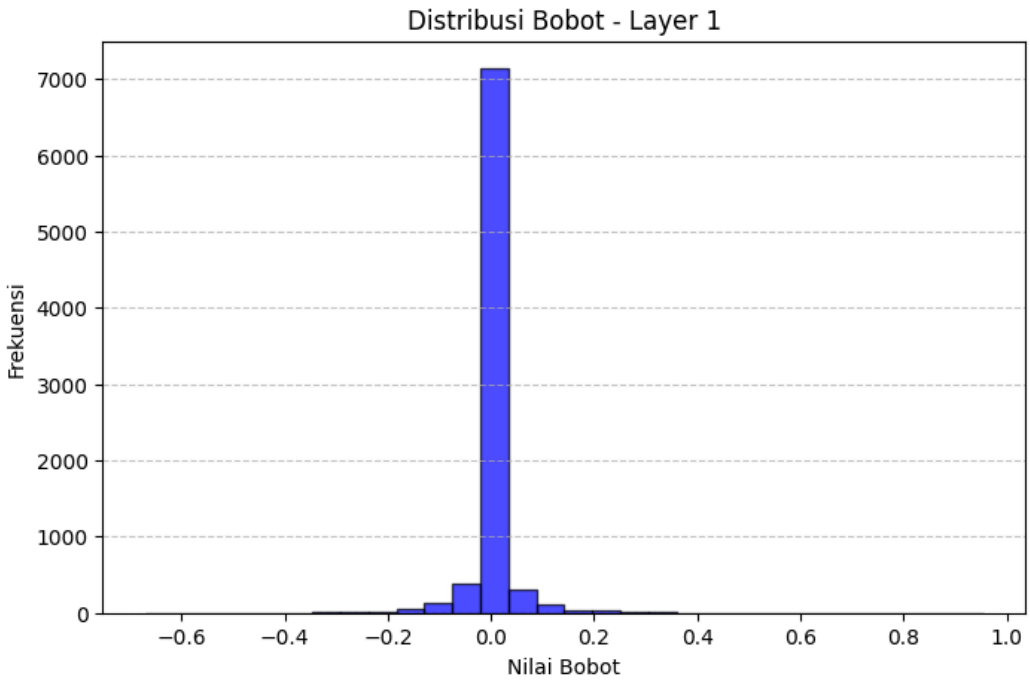


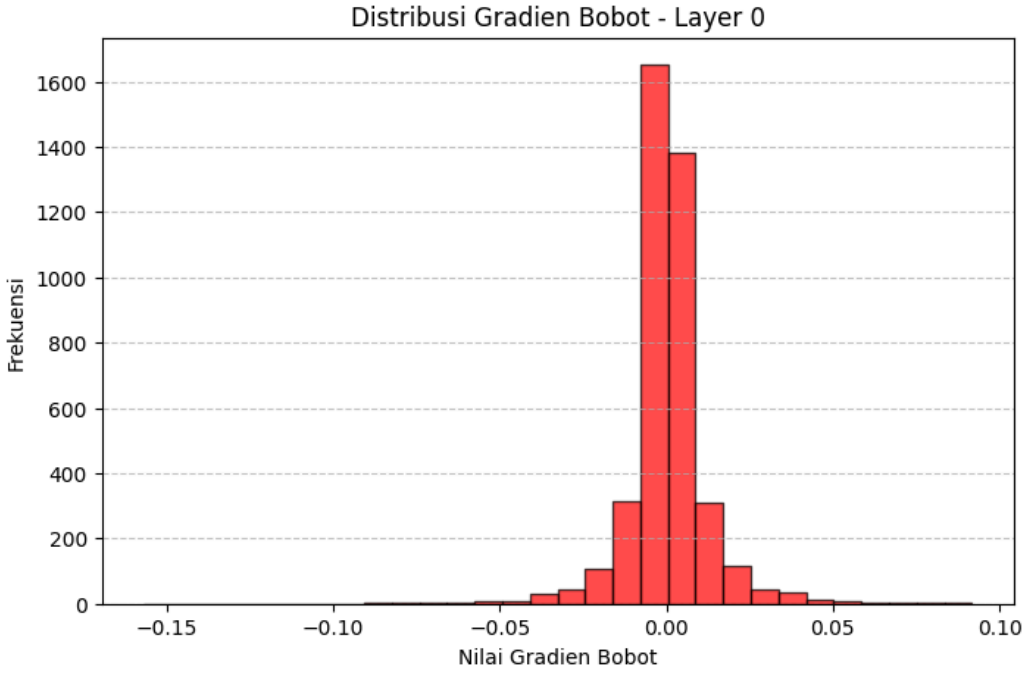

```
[Epoch 1]:
Training Loss: 0.03633028458246594
Validation Loss: 0.8532617037528688

[Epoch 2]:
Training Loss: 0.02194414809338485
Validation Loss: 0.5856529805362231

[Epoch 3]:
Training Loss: 0.01541702998379344
Validation Loss: 0.42383944610832835

[Epoch 4]:
Training Loss: 0.012220710960069374
Validation Loss: 0.36493694302959095

[Epoch 5]:
Training Loss: 0.010998134189792771
Validation Loss: 0.3355508134020039

[Epoch 6]:
Training Loss: 0.01042602720429023
Validation Loss: 0.32259801479291733

[Epoch 7]:
Training Loss: 0.010142635224993094
Validation Loss: 0.3162498750655284

[Epoch 8]:
Training Loss: 0.009945971775828897
Validation Loss: 0.31448842532864546

[Epoch 9]:
Training Loss: 0.009788819207964972
Validation Loss: 0.31359552345842906

[Epoch 10]:
Training Loss: 0.009640655309052929
Validation Loss: 0.3119048813942399
```

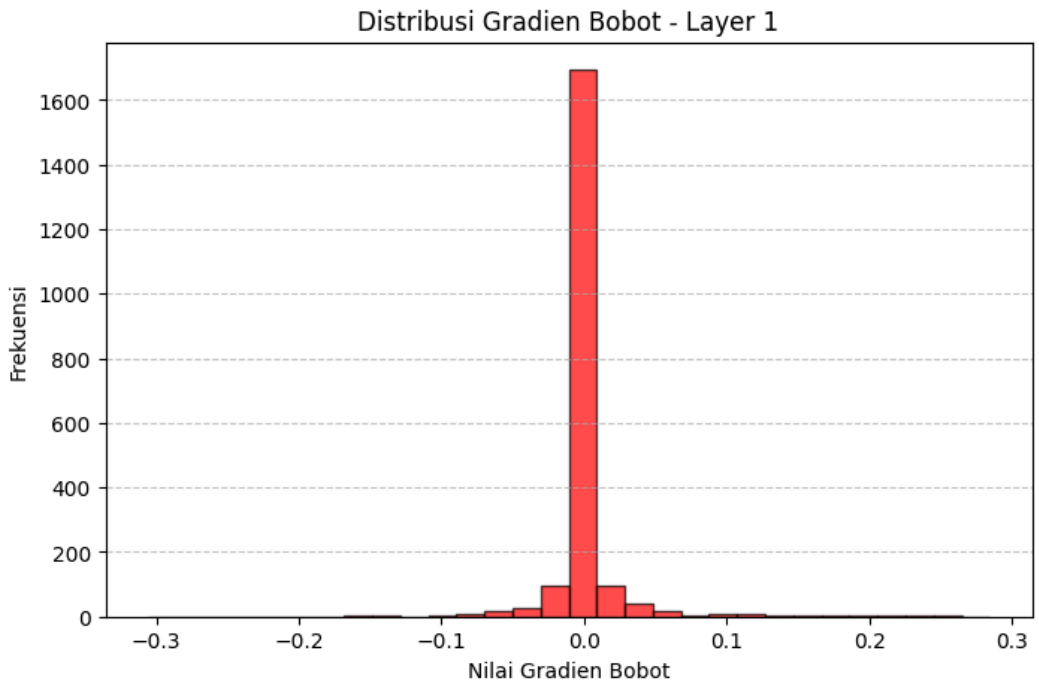
```
[Epoch 11]:
Training Loss: 0.009492140690738607
Validation Loss: 0.30950378071603113

[Epoch 12]:
Training Loss: 0.009342431957769138
Validation Loss: 0.304798864735802

[Epoch 13]:
Training Loss: 0.009202920673205597
Validation Loss: 0.30084981060383065

[Epoch 14]:
Training Loss: 0.009070399080992162
Validation Loss: 0.2973835635102515

[Epoch 15]:
Training Loss: 0.008937408186039171
Validation Loss: 0.29490160857122677

[Epoch 16]:
Training Loss: 0.008805435334665589
Validation Loss: 0.2916620308814151

[Epoch 17]:
Training Loss: 0.008668800727318104
Validation Loss: 0.2882328188453499

[Epoch 18]:
Training Loss: 0.008523833041094633
Validation Loss: 0.28367478691843195

[Epoch 19]:
Training Loss: 0.008367335891321968
Validation Loss: 0.27810424648365706

[Epoch 20]:
Training Loss: 0.008198858987507293
Validation Loss: 0.2733349700599264
```


Training and Validation Loss Over Epochs
Training Loss
Validation Loss
Loss
Epochs
0.0
0.2
0.4
0.6
0.8
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Distribusi Bobot - Layer 0
Frekuensi
Nilai Bobot
0
10000
20000
30000
40000
50000
−5
−4
−3
−2
−1
0
1
2
3


Distribusi Bobot - Layer 1
Frekuensi
Nilai Bobot
0
1000
2000
3000
4000
5000
6000
7000
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0


Distribusi Gradien Bobot - Layer 0
Frekuensi
Nilai Gradien Bobot
0
200
400
600
800
1000
1200
1400
1600
−0.15
−0.10
−0.05
0.00
0.05
0.10



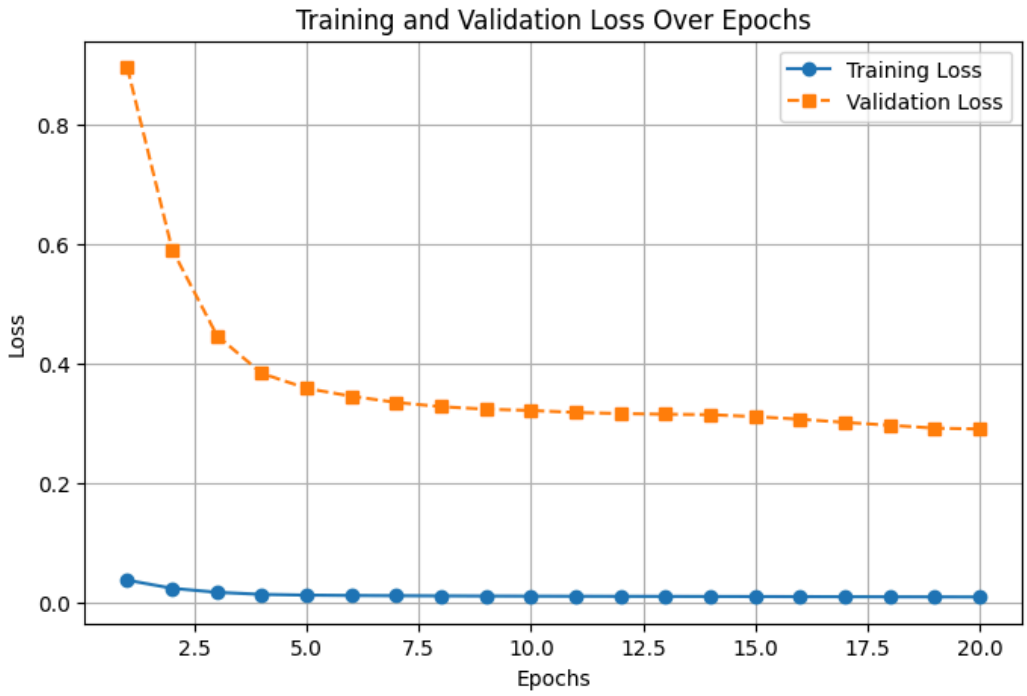
c. *L2 Regularization (λ = 0.001)*

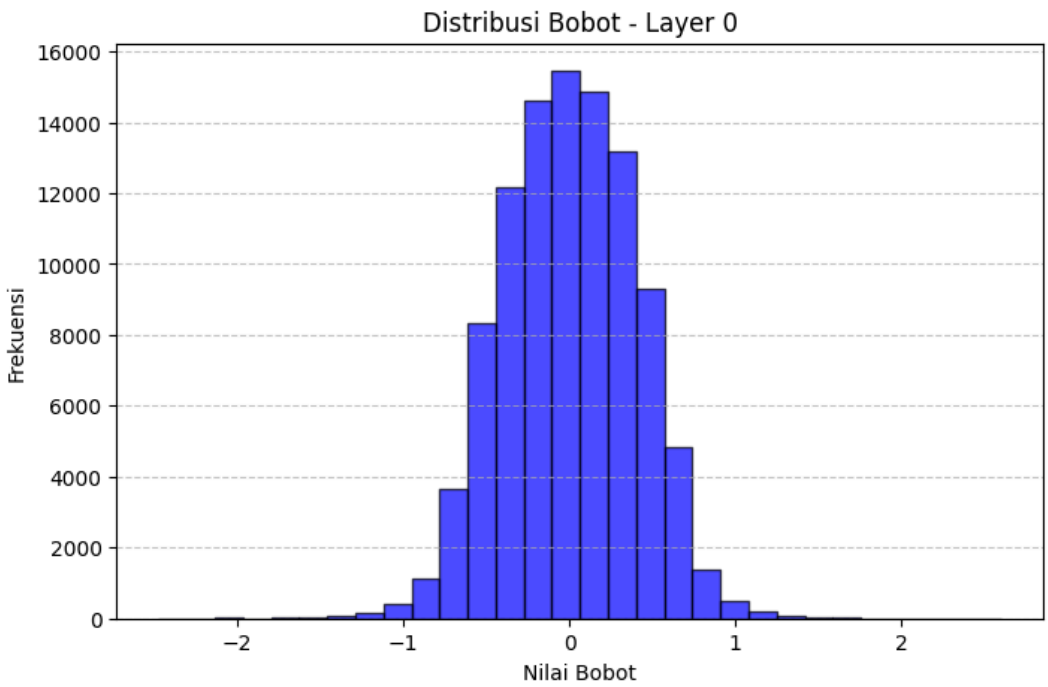
Untuk model ini, nilai *loss* yang ditampilkan adalah nilai aslinya.

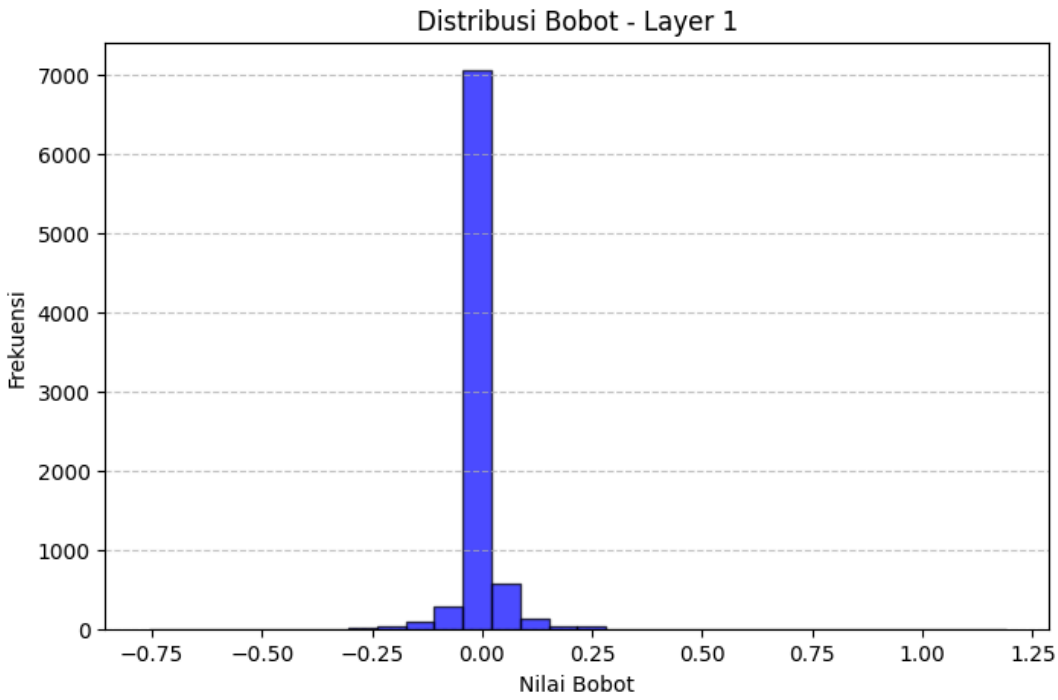

```
[Epoch 1]:
Training Loss: 0.03644435487086866
Validation Loss: 0.8967201568276306

[Epoch 2]:
Training Loss: 0.022815670189428844
Validation Loss: 0.5912592254984893

[Epoch 3]:
Training Loss: 0.016111643912606416
Validation Loss: 0.44674967372542657
```

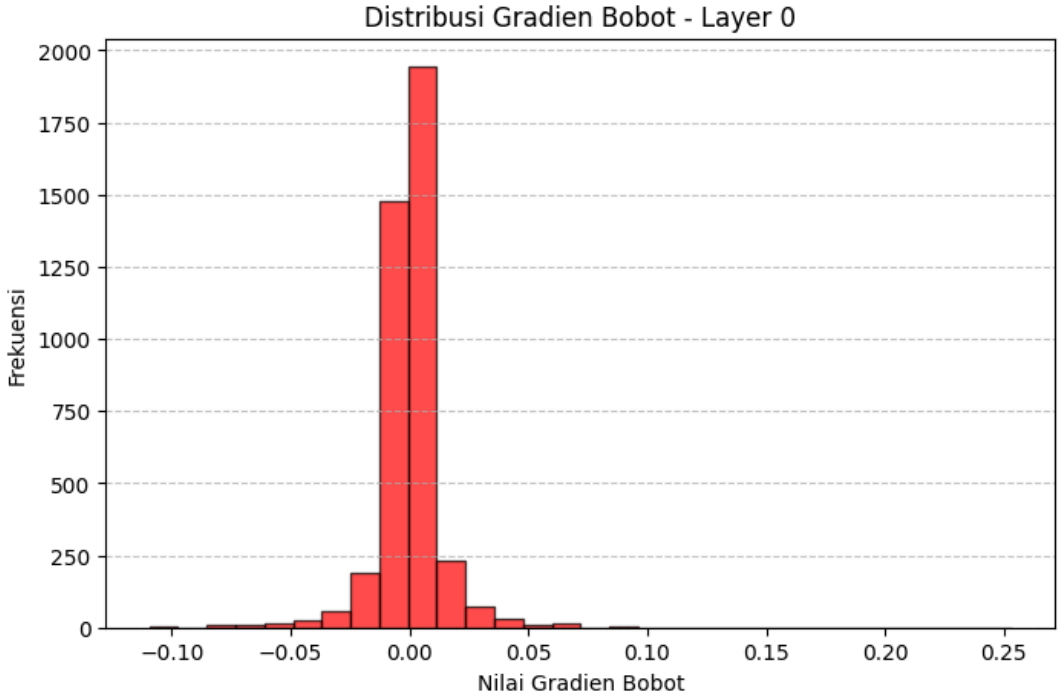
```
[Epoch 4]:
Training Loss: 0.012644737145540236
Validation Loss: 0.3834033684826814

[Epoch 5]:
Training Loss: 0.01137992338714097
Validation Loss: 0.3582266756353856

[Epoch 6]:
Training Loss: 0.010790468698800088
Validation Loss: 0.34506371144046455

[Epoch 7]:
Training Loss: 0.0104108268613163
Validation Loss: 0.3346432162335393

[Epoch 8]:
Training Loss: 0.010116133541470238
Validation Loss: 0.32752084438384066

[Epoch 9]:
Training Loss: 0.009865786783786747
Validation Loss: 0.3233543344743368

[Epoch 10]:
Training Loss: 0.009653303614423748
Validation Loss: 0.3212179444493647

[Epoch 11]:
Training Loss: 0.009497594911010192
Validation Loss: 0.3178141751679463

[Epoch 12]:
Training Loss: 0.009355873587590343
Validation Loss: 0.31602878190379896

[Epoch 13]:
Training Loss: 0.009226323474448428
Validation Loss: 0.3148368542772289

[Epoch 14]:
Training Loss: 0.009111602641521013
Validation Loss: 0.31410795502796524

[Epoch 15]:
Training Loss: 0.008974591828458776
```

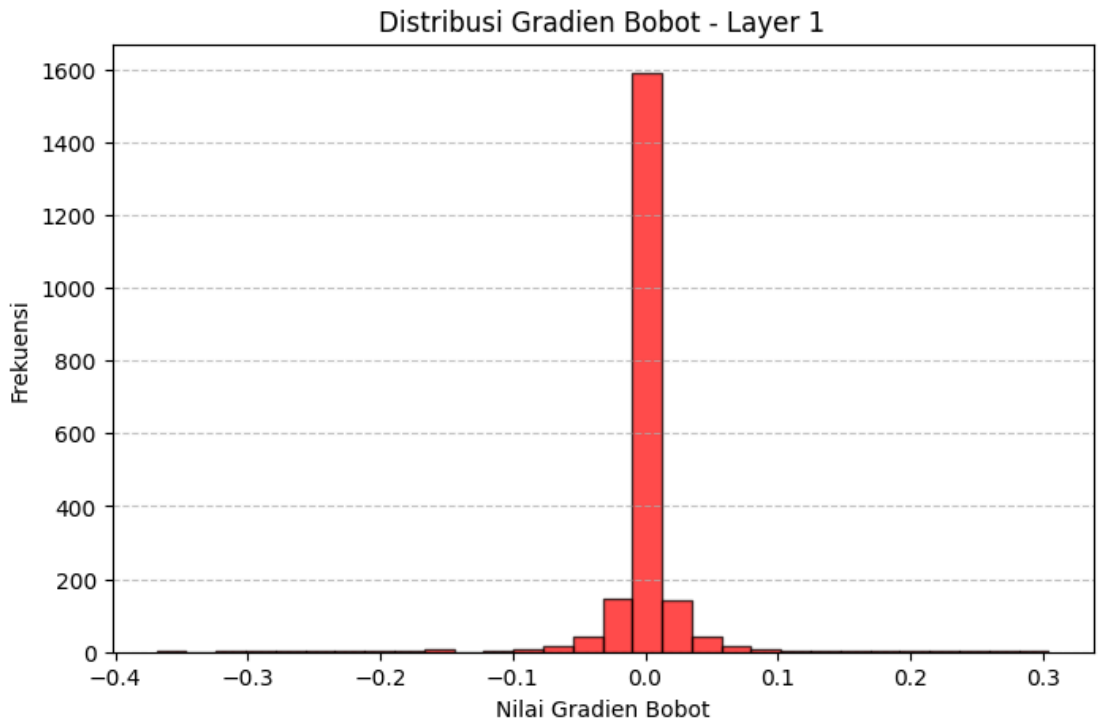
```
Validation Loss: 0.31083882646035543

[Epoch 16]:
Training Loss: 0.008840705827628245
Validation Loss: 0.3064696048033826

[Epoch 17]:
Training Loss: 0.008700671515553367
Validation Loss: 0.3011494450983544

[Epoch 18]:
Training Loss: 0.008578243056946174
Validation Loss: 0.29636395384688125

[Epoch 19]:
Training Loss: 0.008463429097478851
Validation Loss: 0.29150287672389613

[Epoch 20]:
Training Loss: 0.00834349235867331
Validation Loss: 0.2901262252218573
```




Distribusi Bobot - Layer 0
Frekuensi
Nilai Bobot
16000
14000
12000
10000
8000
6000
4000
2000
0
−2
−1
0
1
2


Distribusi Bobot - Layer 1
Frekuensi
Nilai Bobot
7000
6000
5000
4000
3000
2000
1000
0
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25


Distribusi Gradien Bobot - Layer 0
Frekuensi
Nilai Gradien Bobot
2000
1750
1500
1250
1000
750
500
250
0
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
0.25



d. Perbandingan

Secara keseluruhan, jika dibandingkan dengan Base Model, Validation Loss untuk L1 Regularization maupun L2 Regularization selalu menurun, dibandingkan Base Model dimana terjadi fluktuasi walaupun trennya menurun.

H. Perbandingan dengan *library sklearn*

Untuk perbandingan dengan library MLP sklearn, akan menggunakan hyperparameter berikut:

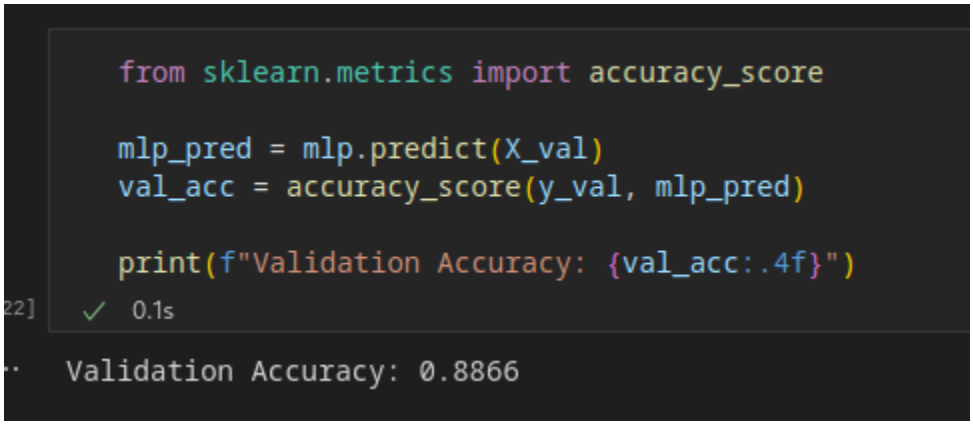
- 2 hidden layer, [128 neuron, 64 neuron]
- Fungsi aktivasi tanh
- *Learning rate* = 0.001
- *Epoch = 15*
- *Batch size = 32*
- Inisialisasi bobot distribusi *uniform*

Setelah dilakukan *training* pada model MLP *sklearn*, didapatkan hasil *loss* berikut:

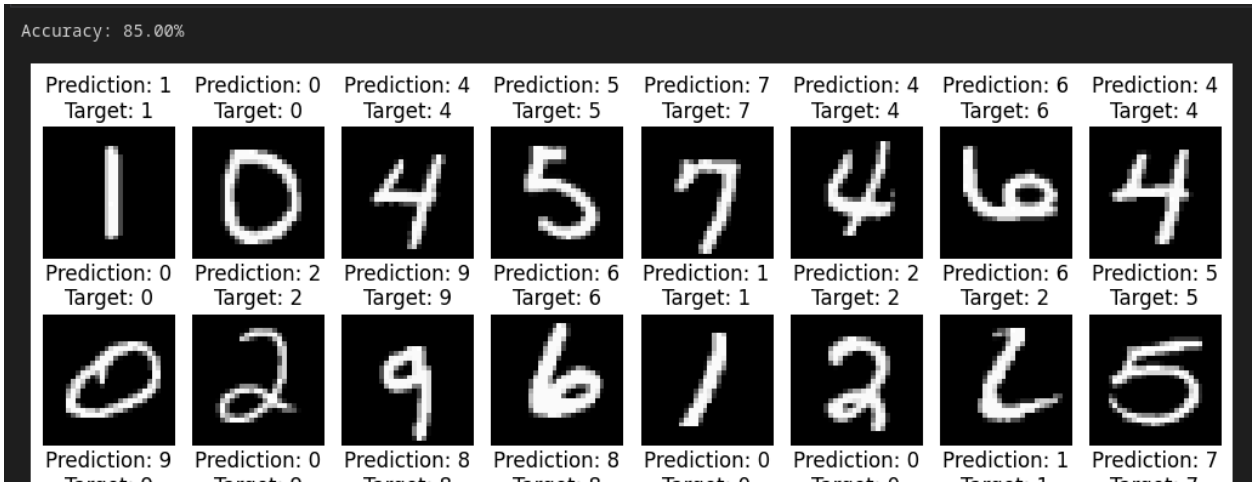
```
Iteration 1, loss = 1.15235224
Iteration 2, loss = 0.78783316
```

```
Iteration 3, loss = 0.70354855
Iteration 4, loss = 0.67189588
Iteration 5, loss = 0.64780021
Iteration 6, loss = 0.63269421
Iteration 7, loss = 0.59001100
Iteration 8, loss = 0.57485053
Iteration 9, loss = 0.53874835
Iteration 10, loss = 0.52509403
Iteration 11, loss = 0.50237153
Iteration 12, loss = 0.49755271
Iteration 13, loss = 0.51822235
Iteration 14, loss = 0.50488802
Iteration 15, loss = 0.45732488
```

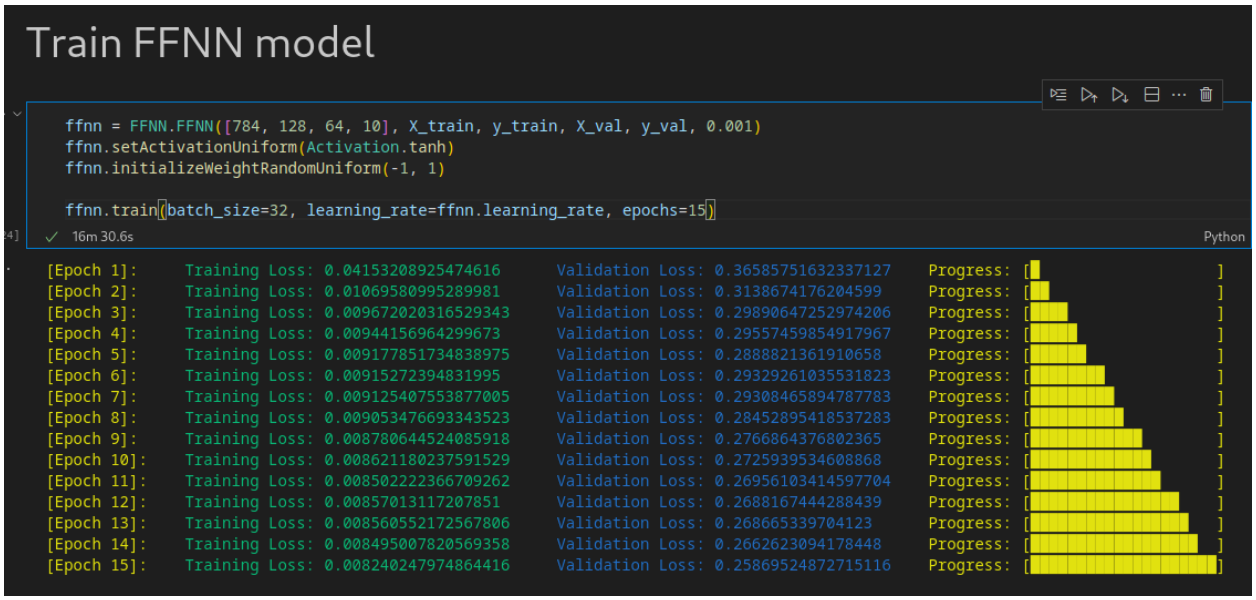
Performa model *sklearn* tidak jauh berbeda dari model-model FFNN yang sudah diuji sebelumnya.



Hasil pengecekan akurasi pada *validation set* menunjukkan model memiliki akurasi 86%. Ketika melakukan prediksi pada data *random* didapat hasil model sebagai berikut:



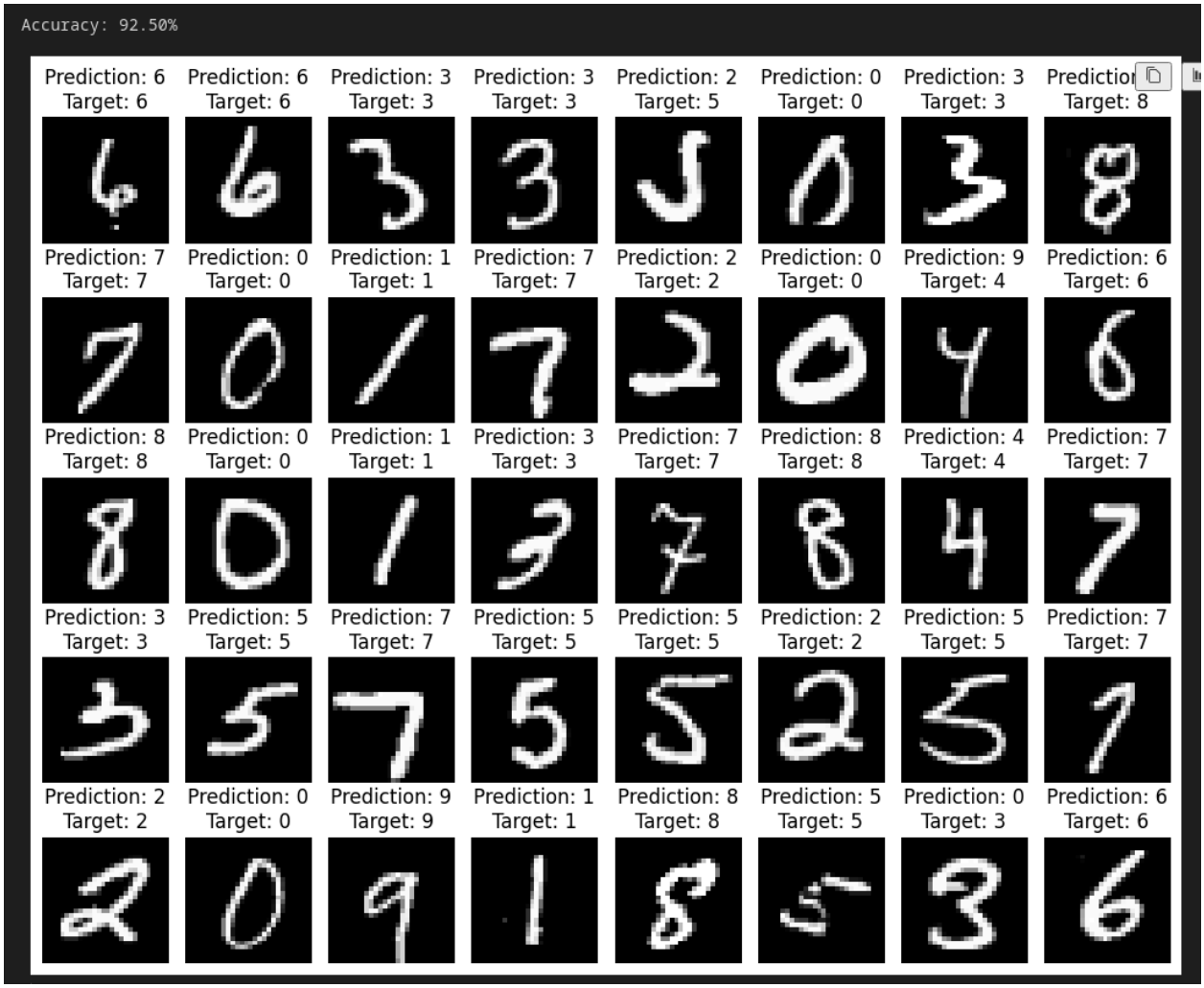
Pada subset data tersebut, model mendapatkan akurasi 85%. Pelatihan untuk model FFNN mendapatkan hasil seperti berikut:



```
from sklearn.metrics import accuracy_score

ffnn_pred = mlp.predict(X_val)
ffnn_val_acc = accuracy_score(y_val, ffnn_pred)

print(f"Validation Accuracy: {ffnn_val_acc:.4f}")
```

25] ✓ 0.6s

```
Validation Accuracy: 0.8866
```



Didapatkan bahwa model FFNN mendapatkan *loss* yang lebih kecil dari model MLP. Ini mungkin dapat terjadi karena terjadinya *one-hot encode*, yaitu perubahan output 1

kolom menjadi output n kolom, berdasarkan jumlah kelas. Ini membuat *error* menjadi jauh lebih rendah, karena banyak dari data target tersebut adalah 0 sehingga tidak berkontribusi ke *error*. Secara kinerja, model FFNN sudah cukup mirip kualitasnya dengan model *sklearn*.

Bab IV
Kesimpulan dan Saran

A. Kesimpulan

Pembelajaran mesin dapat dilakukan menggunakan *feedforward neural network*. Berdasarkan hasil yang didapatkan pada laporan ini, terlihat bahwa metode *backward propagation* pada FFNN merupakan metode yang efektif untuk mensimulasikan model yang dapat melakukan pembelajaran untuk melakukan prediksi angka hasil tulisan tangan.

Arsitektur FFNN yang terdiri atas banyak *neuron* yang tersambung ke banyak bobot, selain didasarkan atas analogi dengan cara kerja otak manusia, terlihat merupakan pendekatan yang bagus untuk melakukan pelatihan sebuah model prediksi. Banyaknya nilai bobot yang dapat dimodifikasi, dan banyaknya *neuron* yang berpengaruh pada hasil *output* menyebabkan fleksibilitas dan ruang lingkup hipotesis model sangat tinggi.

B. Saran

Berdasarkan hasil yang didapatkan pada pengerjaan pemodelan dan laporan ini, terdapat beberapa saran yang dapat diberikan penulis terkait pengerjaan pelatihan model untuk kedepannya:

- Menggunakan library yang dapat memanfaatkan GPU seperti PyTorch atau Cupy untuk melakukan pelatihan model yang jauh lebih cepat, dibanding menggunakan *library* Numpy yang hanya dapat menggunakan CPU.
- Melakukan normalisasi pada data untuk menghindari terjadinya overflow pada saat perkalian matriks.
- Mengimplementasikan metode inisialisasi bobot seperti *He* atau *Xavier initialization* sehingga kinerja model lebih efektif, terutama untuk model-model seperti ReLU.

Bab V
Pembagian Tugas

NIM	Nama	Tugas
13522127	Maulana Muhamad Susetyo	Tampilan struktur FFNN, *save* dan *load*, implementasi *softmax* dan turunan, *binary cross entropy, L1 and L2 Regularization*
13522138	Andi Marihot Sitorus	Distribusi bobot dan gradien, tampilan *loss epoch*, *categorical cross entropy*
13522158	Muhammad Rasheed Qais Tandjung	Rancangan algoritma *forward propagation* dan *backward propagation*, rancangan kelas FFNN, serta kelas-kelas dan fungsi bantuan, setup *notebook*

Bab VI
Referensi

- ***Repository* untuk *source code*:** https://github.com/trimonuter/ML_IF3270_TugasBesar1
- https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/
- Neural Networks Part 7: Cross Entropy Derivatives and Backpropagation
- Backpropagation for Softmax and Multi-Class Classification | Complete M…
- Neural Networks Part 5: ArgMax and SoftMax