Nama : Ameliani Kusmayadi
NIM : 1103213044
Kelas : TK-45-06

WEEK 10 MLP REGRESSION

Library yang digunakan

```
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
import numpy as np
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[ ] # Memuat dataset
    df = pd.read_csv('/content/ObesityDataSet_raw_and_data_sinthetic.csv')
    df.head()
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NObeyesdad
0	Female	21.0	1.62	64.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	0.0	1.0	no	Public_Transportation	Normal_Weight
1	Female	21.0	1.52	56.0	yes	no	3.0	3.0	Sometimes	yes	3.0	yes	3.0	0.0	Sometimes	Public_Transportation	Normal_Weight
2	Male	23.0	1.80	77.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	2.0	1.0	Frequently	Public_Transportation	Normal_Weight
3	Male	27.0	1.80	87.0	no	no	3.0	3.0	Sometimes	no	2.0	no	2.0	0.0	Frequently	Walking	Overweight_Level_I
4	Male	22.0	1.78	89.8	no	no	2.0	1.0	Sometimes	no	2.0	no	0.0	0.0	Sometimes	Public_Transportation	Overweight_Level_II

Kode di atas digunakan untuk memunculkan 5 baris pertama dari dataset yang disimpan dalam bentuk csv. Ini akan menampilkan nama kolom dan beberapa nilai pertama dalam dataset.

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Gender                          2111 non-null   object
 1   Age                             2111 non-null   float64
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   family_history_with_overweight  2111 non-null   object
 5   FAVC                            2111 non-null   object
 6   FCVC                            2111 non-null   float64
 7   NCP                             2111 non-null   float64
 8   CAEC                            2111 non-null   object
 9   SMOKE                           2111 non-null   object
 10  CH2O                            2111 non-null   float64
 11  SCC                             2111 non-null   object
 12  FAF                             2111 non-null   float64
 13  TUE                             2111 non-null   float64
 14  CALC                            2111 non-null   object
 15  MTRANS                          2111 non-null   object
 16  NObeyesdad                      2111 non-null   object
dtypes: float64(8), object(9)
memory usage: 280.5+ KB
```

Kode tersebut digunakan untuk medapatkan informasi dari sebuah dataframe.

```
df.describe()
```

	Age	Height	Weight	FCVC	NCP	CH2O	FAF	TUE
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000
mean	24.312600	1.701677	86.586058	2.419043	2.685628	2.008011	1.010298	0.657866
std	6.345968	0.093305	26.191172	0.533927	0.778039	0.612953	0.850592	0.608927
min	14.000000	1.450000	39.000000	1.000000	1.000000	1.000000	0.000000	0.000000
25%	19.947192	1.630000	65.473343	2.000000	2.658738	1.584812	0.124505	0.000000
50%	22.777890	1.700499	83.000000	2.385502	3.000000	2.000000	1.000000	0.625350
75%	26.000000	1.768464	107.430682	3.000000	3.000000	2.477420	1.666678	1.000000
max	61.000000	1.980000	173.000000	3.000000	4.000000	3.000000	3.000000	2.000000

Kode tersebut digunakan untuk menampilkan statistik deskriptif dari sebuah dataframe. Output yang ditampilkan memberikan gambaran singkat tentang distribusi data pada setiap kolom numerik dalam dataframe tersebut.

```
df.isna().sum()
```

	0
Gender	0
Age	0
Height	0
Weight	0
family_history_with_overweight	0
FAVC	0
FCVC	0
NCP	0
CAEC	0
SMOKE	0
CH2O	0
SCC	0
FAF	0
TUE	0
CALC	0
MTRANS	0
NObeyesdad	0

dtype: int64

Kode ini digunakan untuk menghitung jumlah nilai yang hilang (missing value) pada setiap kolom dalam sebuah dataframe. Output tersebut menunjukkan nilai 0 yang artinya tidak ada satu pun nilai yang hilang pada semua kolom dalam dataframe.

```
df.columns
```

```
Index(['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweight',
       'FAVC', 'FCVC', 'NCP', 'CAEC', 'SMOKE', 'CH2O', 'SCC', 'FAF', 'TUE',
       'CALC', 'MTRANS', 'NObeyesdad'],
      dtype='object')
```

Kode tersebut digunakan untuk menampilkan semua nama kolom yang ada dalam dataframe.

```
[ ] # Melakukan encoding pada kolom kategorikal menggunakan LabelEncoder
    label_cols = ['Gender', 'family_history_with_overweight', 'FAVC', 'CAEC', 'SMOKE', 'SCC', 'CALC', 'MTRANS', 'NObeyesdad']
    label_encoder = LabelEncoder()

    for col in label_cols:
        df[col] = label_encoder.fit_transform(df[col])


    # Menentukan fitur (X) dan target (y)
    X = df.drop(['NObeyesdad'], axis=1)  # Menghapus kolom target
    y = df['NObeyesdad']  # Kolom target


    # Split data menjadi training dan testing (80% training, 20% testing)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Kode ini melakukan proses preprocessing data yang umum dilakukan sebelum membangun model machine learning.

```
[ ] # Standardisasi fitur numerik
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)


    # Konversi data ke tensor
    X_train_tensor = torch.tensor(X_train_scaled, dtype=torch.float32)
    y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32)
    X_test_tensor = torch.tensor(X_test_scaled, dtype=torch.float32)
    y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32)
```

Kode ini digunakan sebagai langkah preprocessing data sebelum melatih model deep learning, dengan neural network menggunakan framework seperti PyTorch atau TensorFlow.

```
# Menyusun model MLP untuk regresi
class MLPRegression(nn.Module):
    def __init__(self, input_size, hidden_layers, neurons, activation):
        super(MLPRegression, self).__init__()
        self.input_size = input_size
        self.hidden_layers = hidden_layers
        self.neurons = neurons
        self.activation = activation

        # Membuat layer input ke layer tersembunyi
        layers = []
        layers.append(nn.Linear(self.input_size, self.neurons))

        # Menambahkan hidden layers
        for _ in range(self.hidden_layers - 1):
            layers.append(self.activation())
            layers.append(nn.Linear(self.neurons, self.neurons))

        layers.append(nn.Linear(self.neurons, 1))  # Layer output

        self.model = nn.Sequential(*layers)

    def forward(self, x):
        return self.model(x).squeeze()
```

Kode ini memberikan kerangka dasar untuk membangun model regresi menggunakan MLP di PyTorch. Dengen memodifikasi parameter dan menambah fitur-fitur tambahan, model ini dapat disesuaikan untuk berbagai masalah regresi.

```python
# Setup perangkat (GPU jika tersedia)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Hyperparameter yang akan diuji
hidden_layers = [1, 2, 3]
neurons = [4, 8, 16, 32, 64]
activations = [nn.Sigmoid, nn.ReLU, nn.Softmax, nn.Tanh]  # Changed nn.RelU to nn.ReLU
epochs_list = [1,10,25,50,100,250]
learning_rates = [10, 1, 0.1, 0.01, 0.001, 0.0001]
batch_sizes = [16,32,64,128,256,512]

# Menyimpan hasil
results = []

# Melakukan eksperimen dengan kombinasi hyperparameter
for layers in hidden_layers:
    for neuron in neurons:
        for activation in activations:
            for epochs in epochs_list:
                for lr in learning_rates:
                    for batch_size in batch_sizes:
                        # Membuat dan memindahkan model ke perangkat (GPU atau CPU)
                        model = MLPRegression(input_size=X_train_tensor.shape[1],
                                              hidden_layers=layers,
                                              neurons=neuron,
                                              activation=activation).to(device)

                        # Mendefinisikan loss function dan optimizer
                        criterion = nn.MSELoss()
                        optimizer = optim.Adam(model.parameters(), lr=lr)

                        # Training loop
                        for epoch in range(epochs):
```

Kode diatas digunakan untuk melakukan hyperparameter tuning untuk model regresi MLPRegression yang telah didefinisikan sebelumnya. Hyperparameter tuning adalah proses mencoba kombinasi parameter yang berbeda untuk menemukan kombinasi yang menghasilkan performa terbaik pada model.

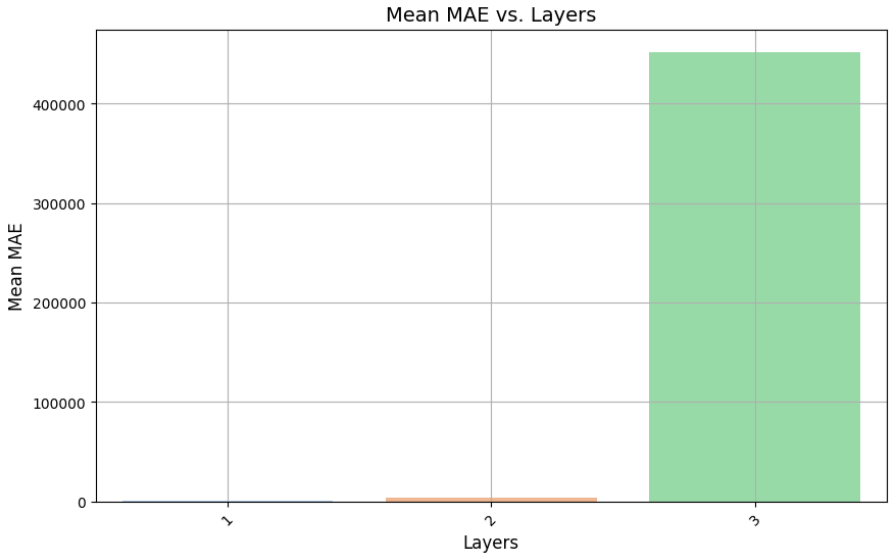
```python
# Mengonversi hasil ke DataFrame dan menyimpannya ke CSV
results_df = pd.DataFrame(results)
results_df.to_csv("mlp_regression_hidden layer 123.csv", index=False)
print("All results have been saved to 'mlp_regression_hidden layer 123.csv'.")
```

```
All results have been saved to 'mlp_regression_hidden layer 123.csv'.
```

Kode ini merupakan Langkah penting dalam proses hyperparameter tuning. Dengan menyimpan hasil eksperimen ke dalam file CSV.

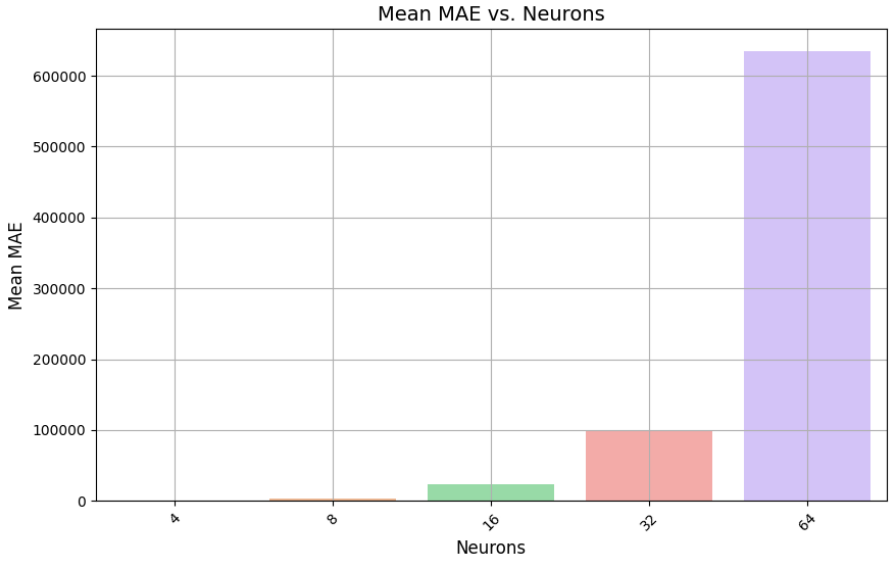
```python
# Select relevant hyperparameters and mean MAE
hyperparameters = ['layers', 'neurons', 'activation', 'epochs', 'lr', 'batch_size']
mean_mae_by_hyperparameter = results_df.groupby(hyperparameters)['mae'].mean().reset_index()

# Plot mean MAE against each hyperparameter
for param in hyperparameters:
    plt.figure(figsize=(10, 6))
    sns.barplot(data=mean_mae_by_hyperparameter, x=param, y='mae', ci=None, palette="pastel")
    plt.title(f'Mean MAE vs. {param.capitalize()}', fontsize=14)
    plt.xlabel(param.capitalize(), fontsize=12)
    plt.ylabel('Mean MAE', fontsize=12)
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.show()
```

Kode ini bertujuan untuk menganalisis hasil eksperimen hyperparameter tuning yang telah disimpan dalam dataframe results_df.



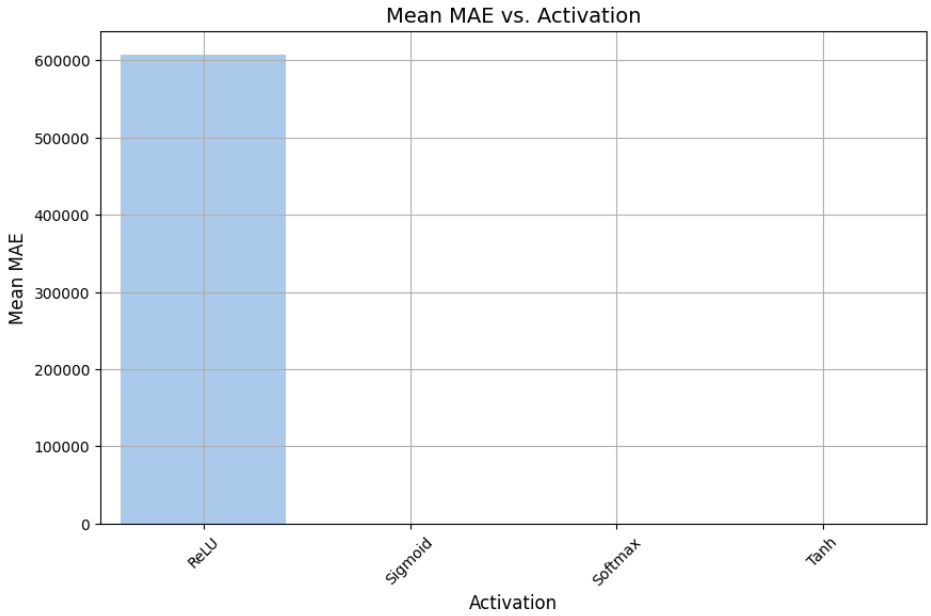
Dari grafik tersebut, dapat disimpulkan bahwa

- Jumlah layer berpengaruh signifikan terhadap nilai MAE: Terdapat perbedaan yang sangat besar antara nilai MAE untuk jumlah layer 1,2, dan 3.
- Model dengan 3 layer memiliki nilai MAE yang jauh lebih tinggi dibandingkan dengan model 1 atau 2 layer: Ini mengindikasikan bahwa model dengan nilai 3 layer memiliki kinerja yang lebih buruk dalam memprediksi nilai sebenarnya.



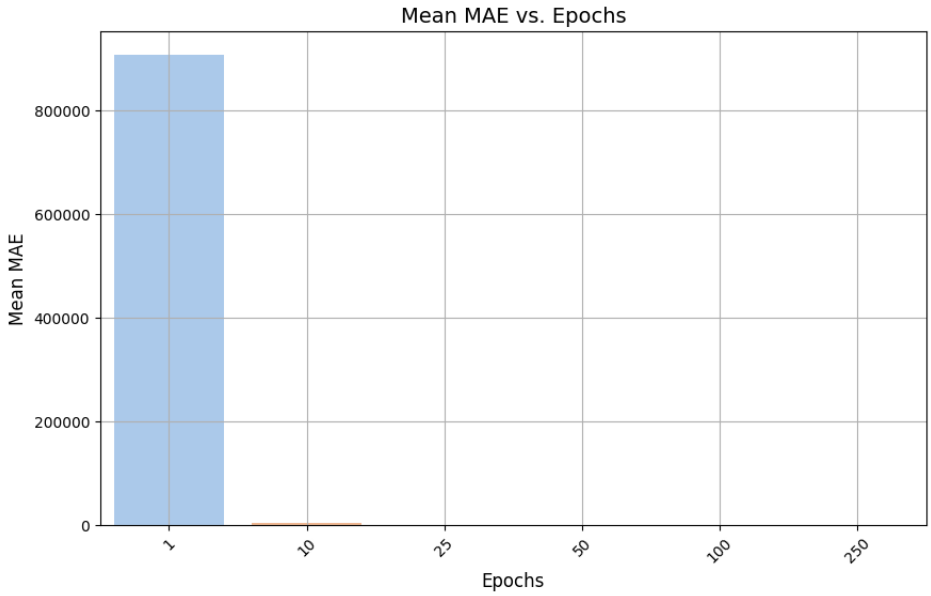
Berdasarkan grafik di atas, dapat disimpulkan bahwa:

- Terdapat perbedaan yang cukup besar pada nilai MAE untuk setiap jumlah neuron.
- Pada kasus ini, model dengan 64 neuron memiliki nilai MAE yang jauh lebih tinggi dibandingkan dengan model dengan jumlah neuron yang lebih sedikit. Ini mengindikasikan bahwa model dengan 64 neuron mungkin mengalami overfitting, yaitu model terlalu menghafal data pelatihan sehingga tidak dapat menggeneralisasi dengan baik pada data yang belum pernah dilihat sebelumnya.
- Model dengan sekitar 32 neuron memiliki kinerja yang cukup baik. Namun, kesimpulan ini perlu divalidasi dengan eksperimen lebih lanjut dan mempertimbangkan metrik evaluasi lainnya.



Berdasarkan grafik diatas, dapat disimpulkan bahwa:

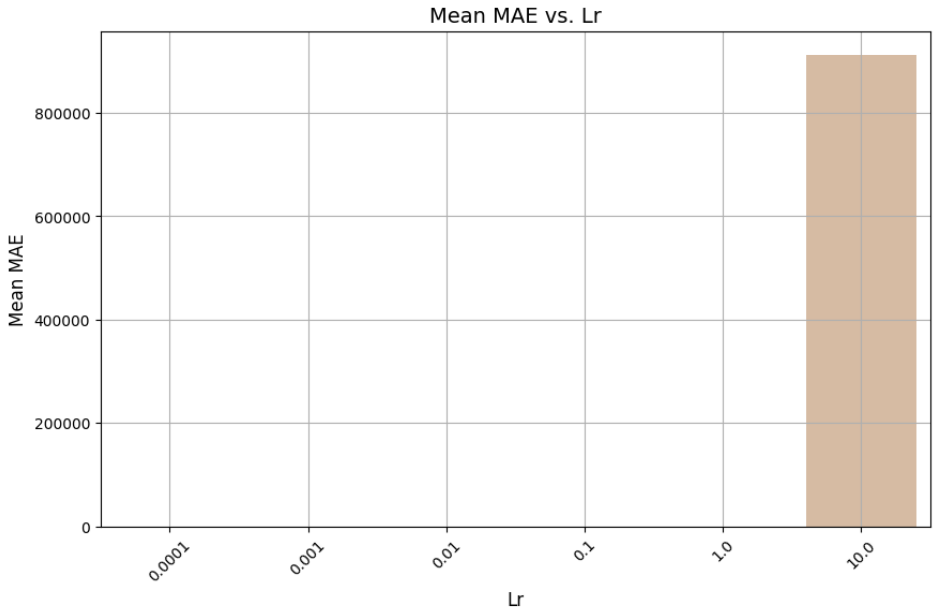
- Fungsi aktivasi berpengaruh signifikan terhadap kinerja model: Terdapat perbedaan yang cukup besar pada nilai MAE untuk setiap jenis fungsi aktivasi.
- ReLU memiliki kinerja terbaik: Fungsi aktivasi ReLU menghasilkan nilai MAE yang paling rendah dibandingkan dengan fungsi aktivasi lainnya. Ini mengindikasikan bahwa ReLU paling cocok untuk model neural network dalam kasus ini.
- Fungsi aktivasi lain kurang optimal: Fungsi aktivasi Sigmoid, Softmax, dan Tanh menghasilkan nilai MAE yang lebih tinggi, menunjukkan bahwa fungsi-fungsi ini kurang efektif dalam memodelkan data.



Berdasarkan grafik diatas, dapat disimpulkan bahwa:

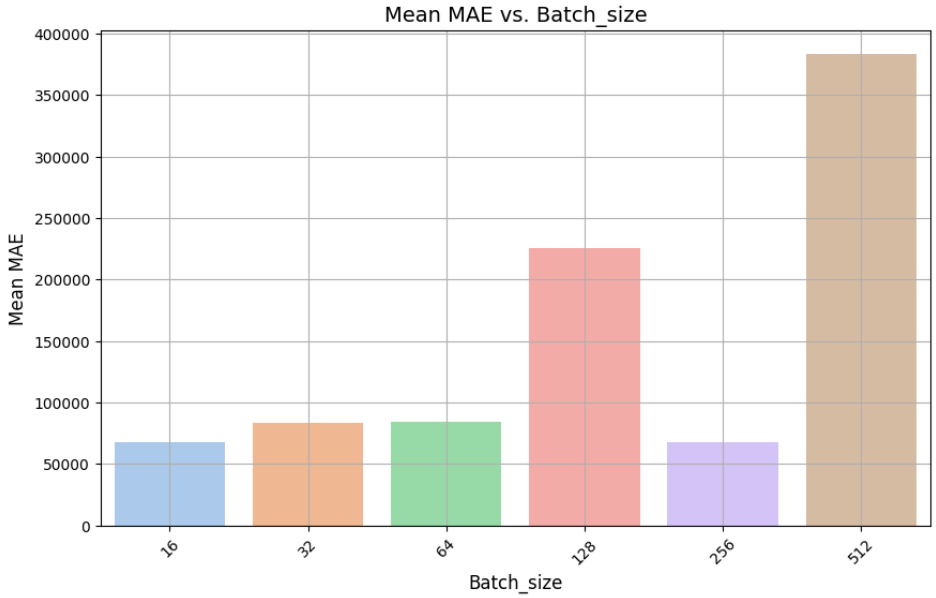
- Jumlah epoch berpengaruh signifikan terhadap kinerja model: Terdapat perbedaan yang sangat besar pada nilai MAE untuk setiap jumlah epoch.
- Model dengan 1 epoch memiliki kinerja yang sangat buruk: Nilai MAE yang sangat tinggi pada 1 epoch menunjukkan bahwa model belum cukup belajar untuk membuat prediksi yang akurat.

- Meningkatkan jumlah epoch tidak selalu meningkatkan kinerja: Setelah 1 epoch, nilai MAE menurun drastis, namun kemudian cenderung stagnan atau bahkan meningkat sedikit. Ini mengindikasikan bahwa model telah mencapai konvergensi atau bahkan overfitting.



Berdasarkan grafik diatas, dapat disimpulkan bahwa:

- Nilai learning rate sangat berpengaruh terhadap kinerja model: Terdapat perbedaan yang sangat signifikan pada nilai MAE untuk setiap nilai learning rate yang dicoba.
- Nilai learning rate yang terlalu besar menyebabkan kinerja model memburuk: Nilai MAE tertinggi terjadi pada learning rate sebesar 10.0. Ini mengindikasikan bahwa model dengan learning rate yang terlalu besar mengalami kesulitan untuk konvergen dan mungkin berosilasi.
- Nilai learning rate yang terlalu kecil juga tidak optimal: Meskipun tidak ditampilkan secara eksplisit pada grafik, umumnya nilai learning rate yang terlalu kecil akan menyebabkan proses pelatihan menjadi sangat lambat dan model membutuhkan waktu yang lama untuk konvergen.



Berdasarkan grafik diatas, dapat disimpulkan bahwa:

- Ukuran batch berpengaruh signifikan terhadap kinerja model: Terdapat perbedaan yang cukup besar pada nilai MAE untuk setiap ukuran batch yang dicoba.
- Ukuran batch yang terlalu kecil atau terlalu besar dapat menurunkan kinerja model: Nilai MAE terendah tidak selalu terjadi pada ukuran batch yang terbesar. Ini menunjukkan bahwa ada ukuran batch optimal yang menghasilkan kinerja terbaik.
- Ukuran batch yang terlalu kecil: Dapat menyebabkan estimasi gradien yang tidak akurat dan memperlambat proses pelatihan.
- Ukuran batch yang terlalu besar: Membutuhkan lebih banyak memori dan dapat menyebabkan generalisasi yang buruk (overfitting).