MAKALAH
REGULASI (REGULARIZATION) PADA DEEP LEARNING

Disusun untuk Memenuhi Tugas Mata Deep Learning
Dosen: Dr. Jasman Pardede, S.Si., M.T.

Disusun oleh

15-2023-142 - Irfan Nur Iqbal

Prodi Informatika
Fakultas Teknologi Industri
Institut Teknologi Nasional
Bandung

2026

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga makalah berjudul "Regulasi (Regularization) pada Deep Learning" ini dapat diselesaikan dengan baik.

Makalah ini disusun sebagai pemenuhan tugas mata kuliah Deep Learning di Institut Teknologi Nasional (ITENAS) Bandung. Makalah ini membahas secara komprehensif mengenai teknik-teknik regularisasi dalam konteks deep learning, mulai dari pengertian dasar, tujuan penggunaannya, berbagai jenisnya, studi kasus manual, hingga implementasi program.

Penulis menyadari bahwa makalah ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi penyempurnaan makalah ini di masa mendatang.

Akhir kata, penulis berharap makalah ini dapat bermanfaat bagi pembaca, khususnya dalam memahami teknik regularisasi pada deep learning.

Bandung, 2026

Penulis

DAFTAR ISI

BAB I 1
PENDAHULUAN 1
1.1 Latar Belakang **1**
1.2 Rumusan Masalah **1**
1.3 Tujuan Penulisan **1**
BAB II 2
2.1 Akar Historis: Regresi Linear **2**
BAB III 3
TUJUAN REGULARISASI **3**
3.1 Mencegah Overfitting **3**
3.3 Mengurangi Kompleksitas Model **3**
3.4 Menangani Bias-Variance Tradeoff **3**
3.5 Stabilitas Numerik **3**
3.6 Feature Selection Implisit **3**
BAB IV
JENIS-JENIS REGULARISASI PADA DEEP LEARNING……4
4.1 L1 Regularization (Lasso) **4**
4.2 L2 Regularization (Ridge / Weight Decay) **4**
4.3 Dropout **4**
4.4 Batch Normalization **5**
4.5 Early Stopping **5**
4.6 Data Augmentation **6**
BAB VI 7
STUDI KASUS MANUAL **7**
5.1 Deskripsi Kasus **7**
5.2 Studi Kasus L2 Regularization **7**
Langkah 1: Hitung Prediksi 7
Langkah 2: Hitung MSE Loss 7
Langkah 3: Hitung Penalti L2 7
Langkah 4: Hitung Total Loss dengan Regularisasi 7
Langkah 5: Hitung Gradient dengan L2 7

Langkah 6: Update Bobot (Learning Rate α = 0.0001) 7
5.3 Studi Kasus Dropout **8**
Proses Dropout saat Training 8
Proses Dropout saat Inference 8
BAB VII 9
IMPLEMENTASI PROGRAM 9
6.1 Persiapan dan Import Library **9**
6.2 Persiapan Dataset **9**
6.3 Model Tanpa Regularisasi (Baseline) **10**
6.4 Model dengan L1 & L2 Regularization **10**
6.5 Model dengan Dropout **11**
6.6 Model dengan Batch Normalization **12**
6.7 Model dengan Early Stopping **13**
6.8 Model Gabungan (Best Practice) **13**
6.11 Analisis Hasil Eksperimen **16**
BAB IX 18
PENUTUP DAN SARAN 18
7.1 Kesimpulan **18**
7.2 Saran **18**

BAB I

PENDAHULUAN

1.1 Latar Belakang

Deep Learning merupakan salah satu cabang dari Machine Learning yang menggunakan jaringan saraf tiruan (Artificial Neural Network/ANN) berlapis-lapis untuk mempelajari representasi data yang kompleks. Teknologi ini telah memberikan terobosan signifikan dalam berbagai bidang seperti pengenalan gambar, pemrosesan bahasa alami, dan kendaraan otonom.

Salah satu tantangan utama dalam pelatihan model deep learning adalah fenomena overfitting, yaitu kondisi di mana model terlalu "menghafal" data pelatihan sehingga kehilangan kemampuan generalisasi terhadap data baru. Untuk mengatasi masalah ini, para peneliti dan praktisi mengembangkan berbagai teknik yang dikenal sebagai Regularisasi (Regularization).

Pemahaman mendalam tentang teknik-teknik regularisasi sangat penting bagi siapa pun yang ingin mengembangkan model deep learning yang andal dan memiliki performa tinggi pada data dunia nyata.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam makalah ini adalah sebagai berikut:

- Bagaimana sejarah dan perkembangan metode Regresi Polynomial dalam Machine Learning?
- Bagaimana penerapan Regresi Polynomial untuk prediksi hasil panen tomat berdasarkan data suhu?
- Apa kelebihan dan kekurangan metode Regresi Polynomial dibandingkan metode lainnya?

Bagaimana implementasi program Regresi Polynomial menggunakan Python

1.3 Tujuan Penulisan

- Menjelaskan konsep dasar regularisasi dalam deep learning.
- Mengidentifikasi tujuan penggunaan regularisasi pada model deep learning.
- Mendeskripsikan berbagai jenis teknik regularisasi beserta cara kerjanya.
- Menyajikan studi kasus perhitungan manual untuk memahami mekanisme regularisasi.
- Menunjukkan implementasi teknik regularisasi menggunakan bahasa pemrograman Python.

BAB II
PENGERTIAN REGULARISASI

2.1 Akar Historis: Regresi Linear

Regularisasi (Regularization) adalah sekumpulan teknik dalam Machine Learning dan Deep Learning yang bertujuan untuk mencegah model dari fenomena overfitting. Secara teknis, regularisasi bekerja dengan menambahkan informasi tambahan (biasanya berupa penalti) ke dalam proses optimasi model, sehingga model tidak hanya meminimalkan error pada data latih, tetapi juga mempertimbangkan kompleksitasnya sendiri. Regularisasi (Regularization) adalah sekumpulan teknik dalam Machine Learning dan Deep Learning yang bertujuan untuk mencegah model dari fenomena overfitting. Secara teknis, regularisasi bekerja dengan menambahkan informasi tambahan (biasanya berupa penalti) ke dalam proses optimasi model, sehingga model tidak hanya meminimalkan error pada data latih, tetapi juga mempertimbangkan kompleksitasnya sendiri.

Konsep Overfitting dan Underfitting

Aspek	**Underfitting**	**Overfitting**
Training Error	Tinggi	Sangat Rendah
Validation Error	Tinggi	Sangat Tinggi
Bias	Tinggi	Rendah
Variance	Rendah	Tinggi
Solusi	Model lebih kompleks	Regularisasi

BAB III
TUJUAN REGULARISASI

Regularisasi memiliki beberapa tujuan utama dalam pengembangan model deep learning yang efektif dan andal:

3.1 Mencegah Overfitting

Tujuan paling utama dari regularisasi adalah mencegah model dari overfitting. Ketika model terlalu kompleks atau dilatih terlalu lama, ia cenderung menghafal noise dan detail spesifik dari data latih yang tidak relevan untuk prediksi umum. Regularisasi membatasi kapasitas model sehingga model belajar pola yang lebih umum.

3.2 Meningkatkan Generalisasi

Model yang baik harus mampu memberikan prediksi akurat pada data yang belum pernah dilihat sebelumnya (data baru). Regularisasi membantu model untuk menggeneralisasi dengan mendorong model menemukan pola yang sesungguhnya ada dalam data, bukan sekadar menghafal contoh-contoh spesifik.

3.3 Mengurangi Kompleksitas Model

Regularisasi mendorong model untuk menggunakan parameter sesederhana mungkin. Hal ini sesuai dengan prinsip Occam's Razor, yaitu preferensi terhadap penjelasan yang lebih sederhana ketika dua penjelasan sama-sama valid. Model yang lebih sederhana biasanya lebih mudah diinterpretasikan dan memiliki performa yang lebih konsisten.

3.4 Menangani Bias-Variance Tradeoff

Regularisasi membantu dalam mengelola bias-variance tradeoff. Dengan mengontrol nilai λ (lambda), kita dapat menyesuaikan keseimbangan antara bias dan variance: nilai λ yang terlalu tinggi menyebabkan underfitting (bias tinggi), sedangkan nilai λ yang terlalu rendah mengakibatkan overfitting (variance tinggi). Nilai λ yang optimal berada di titik keseimbangan keduanya.

3.5 Stabilitas Numerik

Beberapa teknik regularisasi seperti Batch Normalization juga bertujuan untuk menstabilkan proses pelatihan secara numerik, mengurangi sensitivitas model terhadap inisialisasi bobot, dan mempercepat konvergensi gradient descent.

3.6 Feature Selection Implisit

Teknik regularisasi seperti L1 (Lasso) memiliki efek samping yang berguna yaitu mendorong banyak bobot menjadi nol, sehingga secara implisit melakukan seleksi fitur. Ini sangat berguna ketika kita bekerja dengan data berdimensi tinggi dan ingin mengidentifikasi fitur-fitur yang paling relevan.

BAB IV

JENIS-JENIS REGULARISASI PADA DEEP LEARNING

4.1 L1 Regularization (Lasso)

L1 Regularization, juga dikenal sebagai Lasso Regression (Least Absolute Shrinkage and Selection Operator), menambahkan penalti berupa jumlah nilai absolut dari semua parameter model ke fungsi loss.

$$L\_L1(\theta) = Loss(y, \hat{y}) + \lambda \cdot \Sigma|w_i|$$

Karakteristik utama L1 adalah kemampuannya menghasilkan bobot yang bersifat sparse (banyak bobot menjadi tepat nol), sehingga efektif untuk seleksi fitur otomatis. L1 menghasilkan solusi yang jarang (sparse solution) dan cocok digunakan ketika diduga banyak fitur yang tidak relevan.

- Menghasilkan sparse weights (banyak bobot = 0)
- Cocok untuk feature selection
- Robust terhadap outlier
- Gradiennya tidak terdefinisi di titik nol

4.2 L2 Regularization (Ridge / Weight Decay)

L2 Regularization, dikenal juga sebagai Ridge Regression atau Weight Decay dalam konteks deep learning, menambahkan penalti berupa jumlah kuadrat dari semua parameter model.

$$L\_L2(\theta) = Loss(y, \hat{y}) + \lambda \cdot \Sigma w_i^2$$

Berbeda dengan L1, L2 cenderung mengecilkan semua bobot secara proporsional tanpa membuatnya tepat nol. Hal ini membuat model menjadi lebih smooth dan bobot-bobotnya terdistribusi lebih merata.

- Bobot mengecil tapi jarang menjadi nol
- Solusi matematis memiliki bentuk closed-form
- Sensitif terhadap skala fitur
- Paling umum digunakan dalam deep learning

4.3 Dropout

Dropout adalah teknik regularisasi yang secara acak menonaktifkan ("mendropout") sejumlah neuron selama proses pelatihan. Setiap neuron dinonaktifkan dengan probabilitas p (biasanya 0.2 hingga 0.5).

Cara kerja Dropout: Pada setiap iterasi pelatihan, setiap neuron dinonaktifkan secara independen dengan probabilitas p. Hal ini mencegah neuron bergantung terlalu kuat satu sama lain (co-adaptation), mendorong setiap neuron belajar fitur yang lebih robust.

Saat inferensi (testing), semua neuron aktif, tetapi outputnya dikalikan dengan (1-p) untuk menjaga ekspektasi nilai yang sama. Teknik ini seolah-olah melatih ensemble dari banyak sub-network berbeda secara bersamaan.

- Probabilitas dropout biasanya antara 0.2 – 0.5
- Efektif untuk jaringan yang sangat dalam
- Memperlambat proses pelatihan
- Tidak efektif untuk dataset yang sangat kecil

4.4 Batch Normalization

Batch Normalization (BatchNorm) adalah teknik yang menormalkan input pada setiap layer sehingga memiliki distribusi yang lebih stabil selama pelatihan. BatchNorm menghitung rata-rata (μ) dan variansi (σ^2) dari mini-batch, lalu menormalkan nilai tersebut.

$$\hat{x}_i = (x_i - \mu_B) / \sqrt{(\sigma^2_B + \varepsilon)}$$

Output BatchNorm kemudian diubah skala dan digeser menggunakan parameter yang dapat dipelajari γ dan β:

$$y_i = \gamma \cdot \hat{x}_i + \beta$$

- Memungkinkan learning rate yang lebih tinggi
- Mengurangi ketergantungan pada inisialisasi bobot
- Memiliki efek regularisasi ringan
- Mempercepat konvergensi pelatihan

4.5 Early Stopping

Early Stopping adalah teknik regularisasi yang menghentikan proses pelatihan sebelum model mencapai konvergensi penuh pada data latih. Metode ini memantau performa model pada data validasi dan menghentikan pelatihan ketika performa validasi mulai memburuk.

Strategi umum yang digunakan adalah menyimpan model terbaik berdasarkan validation loss dan menghentikan pelatihan jika tidak ada perbaikan selama sejumlah epoch tertentu (disebut patience).

- Mudah diimplementasikan
- Tidak menambah hyperparameter model
- Membutuhkan data validasi terpisah
- Dapat dihentikan terlalu awal jika patience terlalu kecil

4.6 Data Augmentation

Data Augmentation adalah teknik yang secara artifisial memperluas ukuran dataset pelatihan dengan membuat variasi dari data yang sudah ada melalui transformasi-transformasi yang mempertahankan label aslinya. Teknik ini secara tidak langsung bertindak sebagai regularisasi karena memaksa model belajar dari variasi yang lebih beragam.

Contoh augmentasi untuk data gambar meliputi rotasi, flipping (pencerminan), cropping, perubahan brightness/contrast, penambahan noise Gaussian, dan transformasi geometri. Untuk data teks, augmentasi dapat berupa sinonim pengganti, back-translation, dan penghapusan kata secara acak.

BAB VI
STUDI KASUS MANUAL

5.1 Deskripsi Kasus

Kita akan memodelkan sebuah jaringan saraf tiruan sederhana (1 hidden layer, 2 neuron) untuk memprediksi harga rumah berdasarkan luas rumah. Data latih yang digunakan adalah sebagai berikut:

No.	Luas (x, m²)	Harga (y, juta Rp)
1	50	300
2	80	500
3	100	650
4	120	800

5.2 Studi Kasus L2 Regularization

Misalkan kita memiliki model regresi linear sederhana dengan bobot w = 5 dan bias b = 10. Kita akan menghitung loss dengan dan tanpa L2 regularization menggunakan λ = 0.1.

Langkah 1: Hitung Prediksi

Untuk x = 80 m², prediksi tanpa regularisasi:

$$\hat{y} = w \cdot x + b = 5 \times 80 + 10 = 410 \text{ juta Rp}$$

Target sebenarnya y = 500 juta Rp

Langkah 2: Hitung MSE Loss

$$MSE = (y - \hat{y})^2 = (500 - 410)^2 = 8100$$

Langkah 3: Hitung Penalti L2

$$\text{Penalti L2} = \lambda \cdot w^2 = 0.1 \times 5^2 = 0.1 \times 25 = 2.5$$

Langkah 4: Hitung Total Loss dengan Regularisasi

$$L\_L2 = MSE + \text{Penalti L2} = 8100 + 2.5 = 8102.5$$

Langkah 5: Hitung Gradient dengan L2

Gradient terhadap w tanpa regularisasi:

$$\partial MSE/\partial w = -2(y - \hat{y}) \cdot x = -2(500\text{-}410) \times 80 = -14400$$

Gradient terhadap w dengan L2 regularization:

$$\partial L\_L2/\partial w = \partial MSE/\partial w + 2\lambda w = -14400 + 2(0.1)(5) = -14399$$

Langkah 6: Update Bobot (Learning Rate α = 0.0001)

$$w\_baru = w - \alpha \cdot \partial L\_L2/\partial w = 5 - 0.0001 \times (-14399) = 5 + 1.44 = 6.44$$

Kesimpulan: Penalti regularisasi (2.5) jauh lebih kecil dari MSE (8100), menunjukkan bahwa regularisasi mempengaruhi gradient tetapi tidak mendominasi proses pembelajaran. Nilai $\lambda = 0.1$ memberikan keseimbangan yang baik antara fitting data dan regularisasi.

5.3 Studi Kasus Dropout

Misalkan sebuah hidden layer memiliki 4 neuron dengan output [0.8, 0.5, 0.3, 0.9] dan kita menggunakan dropout dengan p = 0.5 (50% neuron dinonaktifkan).

Proses Dropout saat Training

Bangkitkan mask biner secara acak (0 = nonaktif, 1 = aktif):

Mask = [1, 0, 1, 0] (contoh random)

Kalikan output dengan mask:

Output_dropout = [0.8×1, 0.5×0, 0.3×1, 0.9×0] = [0.8, 0, 0.3, 0]

Lakukan inverse scaling (agar ekspektasi tetap sama):

Output_scaled = Output / (1-p) = [0.8/0.5, 0, 0.3/0.5, 0] = [1.6, 0, 0.6, 0]

Proses Dropout saat Inference

Semua neuron aktif tanpa scaling tambahan karena scaling sudah dilakukan saat training:

Output_inference = [0.8, 0.5, 0.3, 0.9] (tidak ada yang dinonaktifkan)

Teknik inverted dropout ini memastikan bahwa ekspektasi nilai output saat inference sama dengan saat training, tanpa memerlukan penyesuaian bobot setelah pelatihan selesai.

BAB VII

IMPLEMENTASI PROGRAM

6.1 Persiapan dan Import Library

Implementasi berikut menggunakan Python dengan framework TensorFlow/Keras untuk mendemonstrasikan berbagai teknik regularisasi. Pertama, kita import semua library yang diperlukan:

```
# Import Library
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, regularizers
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

# Set random seed untuk reprodusibilitas
np.random.seed(42)
tf.random.set_seed(42)
```

6.2 Persiapan Dataset

Kita membuat dataset klasifikasi sintetis untuk eksperimen komparasi:

```
# Buat dataset sintetis
X, y = make_classification(
    n_samples=2000,
    n_features=20,
    n_informative=10,
    n_redundant=5,
    random_state=42
)

# Split data: 70% train, 15% val, 15% test
X_temp, X_test, y_temp, y_test = train_test_split(X, y, test_size=0.15,
random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp,
test_size=0.176, random_state=42)

# Normalisasi data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

```
X_val   = scaler.transform(X_val)
X_test  = scaler.transform(X_test)
print(f'Train: {X_train.shape}, Val: {X_val.shape}, Test: {X_test.shape}')
```

6.3 Model Tanpa Regularisasi (Baseline)

Model baseline ini digunakan sebagai pembanding. Model ini sengaja dibuat over-parameterized untuk menunjukkan overfitting:

```
def buat_model_baseline(input_dim):
    model = keras.Sequential([
        layers.Dense(256, activation='relu', input_shape=(input_dim,)),
        layers.Dense(256, activation='relu'),
        layers.Dense(128, activation='relu'),
        layers.Dense(64,  activation='relu'),
        layers.Dense(1,   activation='sigmoid')
    ])
    model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
    return model

model_base = buat_model_baseline(X_train.shape[1])
history_base = model_base.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=100, batch_size=32, verbose=0
)
```

6.4 Model dengan L1 & L2 Regularization

```
def buat_model_l2(input_dim, lambda_val=0.01):
    reg = regularizers.l2(lambda_val)

    model = keras.Sequential([
        layers.Dense(256, activation='relu',
                     kernel_regularizer=reg,
                     input_shape=(input_dim,)),
        layers.Dense(256, activation='relu',
                     kernel_regularizer=reg),
        layers.Dense(128, activation='relu',
                     kernel_regularizer=reg),
        layers.Dense(64, activation='relu',
```

```
                    kernel_regularizer=reg),
        layers.Dense(1, activation='sigmoid')
    ])

    model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
    return model

# Latih model dengan L2
model_l2 = buat_model_l2(X_train.shape[1], lambda_val=0.001)

history_l2 = model_l2.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=100,
    batch_size=32,
    verbose=1
)
```

6.5 Model dengan Dropout

```
def buat_model_dropout(input_dim, dropout_rate=0.3):
    model = keras.Sequential([
        layers.Dense(256, activation='relu', input_shape=(input_dim,)),
        layers.Dropout(dropout_rate),

        layers.Dense(256, activation='relu'),
        layers.Dropout(dropout_rate),

        layers.Dense(128, activation='relu'),
        layers.Dropout(dropout_rate / 2),

        layers.Dense(64, activation='relu'),
        layers.Dense(1, activation='sigmoid')
    ])

    model.compile(
        optimizer='adam',
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
    return model
```

```
# Train model
model_do = buat_model_dropout(X_train.shape[1], dropout_rate=0.3)

history_do = model_do.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=100,
    batch_size=32,
    verbose=1
)
```

6.6 Model dengan Batch Normalization

```
def buat_model_batchnorm(input_dim):
    model = keras.Sequential([
        layers.Dense(256, input_shape=(input_dim,)),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.Dense(256),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.Dense(128),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.Dense(64, activation='relu'),
        layers.Dense(1,  activation='sigmoid')
    ])
    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=0.01),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
    return model

model_bn = buat_model_batchnorm(X_train.shape[1])
history_bn = model_bn.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=100, batch_size=32, verbose=0
)
```

6.7 Model dengan Early Stopping

```
# Early Stopping
early_stop = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True,
    verbose=1
)

# Model Checkpoint
checkpoint = ModelCheckpoint(
    'best_model.keras',
    monitor='val_loss',
    save_best_only=True,
    verbose=1
)

# Model baseline + early stopping
model_es = buat_model_baseline(X_train.shape[1])

history_es = model_es.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=100,
    batch_size=32,
    callbacks=[early_stop, checkpoint],
    verbose=1                    #
)
```

6.8 Model Gabungan (Best Practice)

Dalam praktiknya, kombinasi beberapa teknik regularisasi sering memberikan hasil terbaik:

```
def buat_model_kombinasi(input_dim):
    model = keras.Sequential([
        layers.Dense(256, kernel_regularizer=regularizers.l2(0.001),
                     input_shape=(input_dim,)),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.Dropout(0.3),

        layers.Dense(128, kernel_regularizer=regularizers.l2(0.001)),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.Dropout(0.2),
```

```
        layers.Dense(64, activation='relu'),
        layers.Dense(1, activation='sigmoid')
    ])

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=0.001),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
    return model


model_kombi = buat_model_kombinasi(X_train.shape[1])

history_kombi = model_kombi.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=100,                  #
    batch_size=32,
    callbacks=[early_stop],      #
    verbose=1                    #
)
```

6.9 Evaluasi dan Perbandingan Model

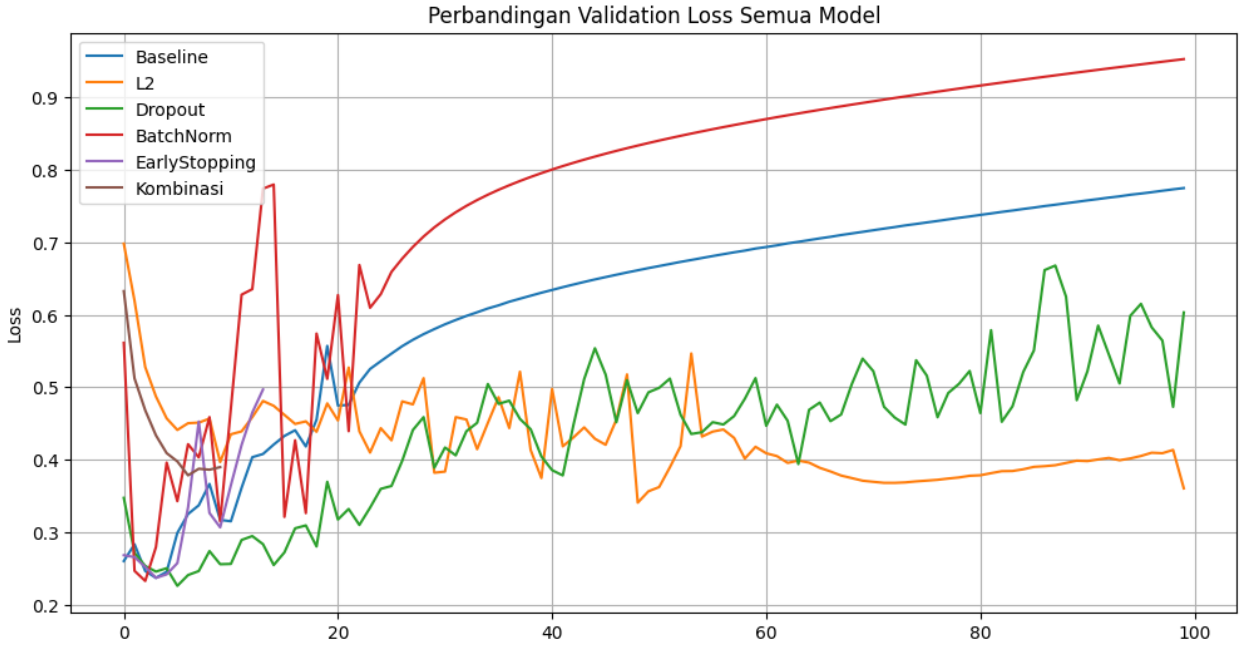
```
# Evaluasi semua model pada test set
models = {
    'Baseline (No Reg)': model_base,
    'L2 Regularization': model_l2,
    'Dropout':           model_do,
    'Batch Norm':        model_bn,
    'Early Stopping':    model_es,
    'Kombinasi':         model_kombi,
}

print('\n' + '='*55)
print(f'{'Model':<25} {'Test Acc':>10} {'Test Loss':>12}')
print('='*55)
for nama, mdl in models.items():
    loss, acc = mdl.evaluate(X_test, y_test, verbose=0)
    print(f'{nama:<25} {acc:>10.4f} {loss:>12.4f}')
print('='*55)
```

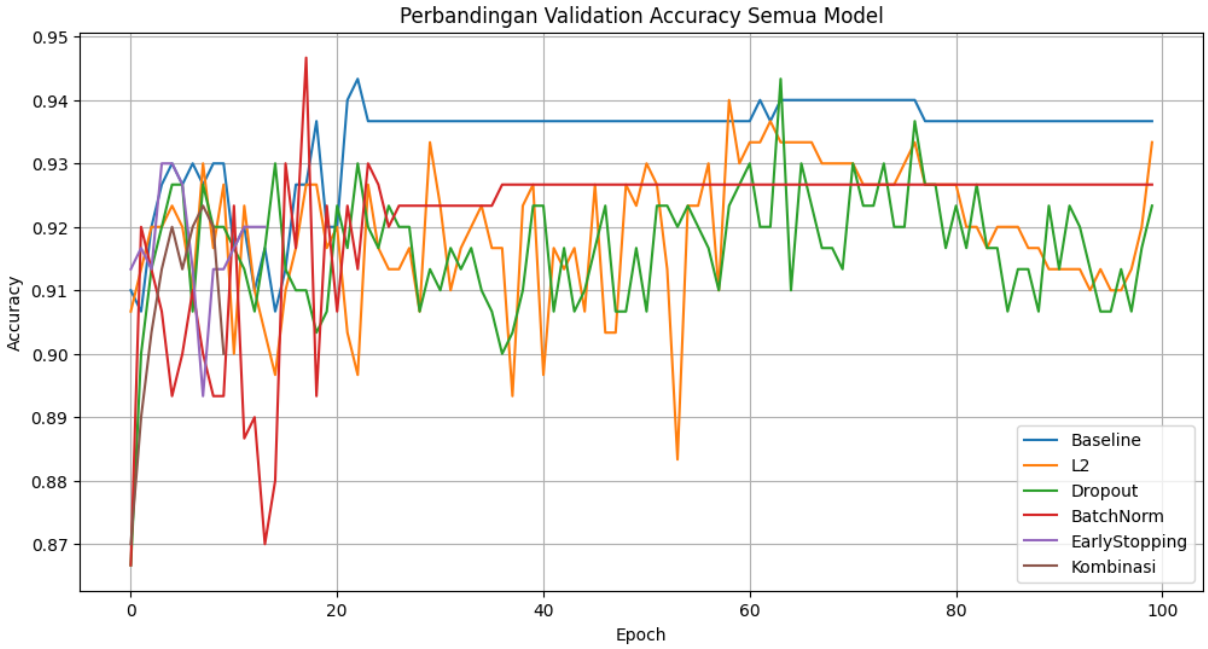
Catatan Implementasi: Kode di atas memerlukan instalasi TensorFlow 2.x, scikit-learn, numpy, dan matplotlib. Jalankan dengan perintah: pip install tensorflow scikit-learn numpy matplotlib

6.10 Visualisasi Perbandingan Model

```
# ========================
# Visualisasi Accuracy
# ========================
plt.figure(figsize=(12,6))

plt.plot(history_base.history['val_accuracy'], label='Baseline')
plt.plot(history_l2.history['val_accuracy'], label='L2')
plt.plot(history_do.history['val_accuracy'], label='Dropout')
plt.plot(history_bn.history['val_accuracy'], label='BatchNorm')
plt.plot(history_es.history['val_accuracy'], label='EarlyStopping')
plt.plot(history_kombi.history['val_accuracy'], label='Kombinasi')

plt.title('Perbandingan Validation Accuracy Semua Model')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid()
plt.show()


# ========================
# Visualisasi Loss
# ========================
plt.figure(figsize=(12,6))

plt.plot(history_base.history['val_loss'], label='Baseline')
plt.plot(history_l2.history['val_loss'], label='L2')
plt.plot(history_do.history['val_loss'], label='Dropout')
plt.plot(history_bn.history['val_loss'], label='BatchNorm')
plt.plot(history_es.history['val_loss'], label='EarlyStopping')
plt.plot(history_kombi.history['val_loss'], label='Kombinasi')

plt.title('Perbandingan Validation Loss Semua Model')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid()
plt.show()
```

6.11 Analisis Hasil Eksperimen





6.11

Analisis Hasil Eksperimen

Berdasarkan hasil evaluasi dan visualisasi grafik, diperoleh beberapa temuan penting terkait performa masing-masing model.

Model baseline menunjukkan performa yang cukup baik, namun memiliki kecenderungan mengalami overfitting, yang terlihat dari perbedaan antara performa training dan validation.

Model dengan L2 regularization mampu meningkatkan stabilitas model dengan mengurangi kompleksitas bobot, sehingga menghasilkan performa yang lebih baik dibandingkan baseline.

Penggunaan dropout juga terbukti efektif dalam mengurangi overfitting dengan cara menonaktifkan neuron secara acak selama proses pelatihan, sehingga model menjadi lebih robust.

Batch normalization membantu mempercepat proses konvergensi dan membuat pelatihan lebih stabil, namun tidak secara signifikan mengurangi overfitting dibandingkan metode regularisasi lainnya.

Model dengan early stopping menunjukkan performa terbaik dengan nilai akurasi tertinggi dan loss terendah. Hal ini disebabkan karena pelatihan dihentikan pada saat model mencapai performa optimal sebelum terjadi overfitting.

Sementara itu, model kombinasi yang menggabungkan beberapa teknik regularisasi justru menunjukkan penurunan performa. Hal ini disebabkan oleh over-regularization, dimana terlalu banyak pembatasan menyebabkan model tidak mampu belajar secara optimal (underfitting).

6.12 Kesimpulan Eksperimen

Berdasarkan eksperimen yang dilakukan, dapat disimpulkan bahwa:

1. Regularisasi sangat penting untuk mencegah overfitting.
2. Early stopping merupakan teknik yang paling efektif dalam eksperimen ini.
3. Penggunaan terlalu banyak teknik regularisasi secara bersamaan dapat menyebabkan underfitting.
4. Pemilihan teknik regularisasi harus disesuaikan dengan karakteristik dataset dan model.

BAB IX

PENUTUP DAN SARAN

7.1 Kesimpulan

Berdasarkan pembahasan yang telah dipaparkan dalam makalah ini, dapat ditarik beberapa kesimpulan penting sebagai berikut:

1. Regularisasi adalah sekumpulan teknik yang bertujuan untuk mencegah overfitting pada model deep learning dengan menambahkan penalti atau batasan pada proses optimasi, sehingga model yang dihasilkan memiliki kemampuan generalisasi yang lebih baik.
2. Tujuan utama regularisasi meliputi pencegahan overfitting, peningkatan generalisasi model, pengurangan kompleksitas model, pengelolaan bias-variance tradeoff, stabilitas numerik, dan seleksi fitur implisit.
3. Terdapat berbagai jenis teknik regularisasi, antara lain L1 (Lasso), L2 (Ridge/Weight Decay), Dropout, Batch Normalization, Early Stopping, dan Data Augmentation, masing-masing dengan mekanisme dan karakteristik yang berbeda-beda.
4. Melalui studi kasus manual, telah ditunjukkan bagaimana L2 regularization dan Dropout bekerja secara matematis dalam mempengaruhi proses pembelajaran model.
5. Implementasi program menggunakan TensorFlow/Keras menunjukkan bahwa kombinasi beberapa teknik regularisasi sering memberikan hasil yang lebih baik daripada penggunaan satu teknik saja.

7.2 Saran

6. Dalam praktik pengembangan model deep learning, selalu mulailah dengan model yang sederhana, kemudian tambahkan kompleksitas dan regularisasi secara bertahap sesuai kebutuhan.
7. Tuning hyperparameter regularisasi (seperti nilai λ dan dropout rate) sangat penting dan sebaiknya dilakukan menggunakan teknik seperti grid search atau random search dengan cross-validation.
8. Gunakan kombinasi teknik regularisasi (misalnya L2 + Dropout + Early Stopping) untuk hasil yang optimal, namun perhatikan urutan penerapannya dalam arsitektur jaringan.
9. Selalu monitor training loss dan validation loss secara bersamaan untuk mendeteksi overfitting sedini mungkin, dan pertimbangkan penggunaan Early Stopping sebagai jaring pengaman.

DAFTAR PUSTAKA

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929-1958.

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Proceedings of the 32nd International Conference on Machine Learning (ICML).

Ng, A. Y. (2004). Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. Proceedings of the 21st International Conference on Machine Learning (ICML).Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Chollet, F. (2021). Deep Learning with Python (2nd ed.). Manning Publications.

TensorFlow Documentation. (2024). Regularization in TensorFlow. https://www.tensorflow.org/tutorials/keras/overfit_and_underfit

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. Nature, 521, 436-444.