Analisis Penerapan ANN dan RNN dengan Inisialisasi Nguyen-Widrow pada Aplikasi Monitoring Banjir dan Gempa

1st Triyo Krismantoro
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
krismantoro@student.telkomuniversity.ac.id

2nd Asep Suhendi
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
suhendi@telkomuniversity.ac.id

3rd Casmika Saputra
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
casmika@telkomuniversity.ac.id

Abstrak— Banjir dan gempa memiliki *parameter-parameter* yang dapat diamati dengan *Internet of Things* (IoT). Hasil dari pemantauan dengan IoT adalah berupa data yang dapat diolah untuk mendapatkan peringatan banjir dan gempa. Dalam penelitian ini, model *neural network* digunakan untuk mengolah data tersebut. Model *neural network* yang digunakan yaitu ANN dan RNN. Dalam penelitian ini juga, model LSTM yang merupakan jenis dari RNN digunakan sebagai pembanding RNN. Inisialisasi bobot dan bias model-model tersebut menggunakan Nguyen-Widrow. ANN, RNN, dan LSTM berturut-turut digunakan untuk deteksi banjir dan gempa, prediksi tinggi air, dan sebagai pembanding RNN dalam memprediksi tinggi air. Model-model tersebut dirancang melalui *trial* and *error* hingga menemukan parameter model yang optimal. Hasil eksperimen menunjukkan bahwa model ANN untuk deteksi banjir memiliki rata-rata akurasi training 0,9969 dan testing 0,9991. Model ANN untuk deteksi gempa memiliki rata-rata akurasi training 0,9967 dan testing 0,9987. Model RNN untuk prediksi tinggi air memiliki hasil yang lebih baik dibandingkan model LSTM. Model-model yang sudah dilatih tersebut kemudian diterapkan pada aplikasi *monitoring* banjir dan gempa. Namun, adanya keterbatasan sehingga data pelatihan model tidak objektif, maka model *neural network* yang telah dibangun saat ini belum dapat untuk digunakan secara publik. Walaupun demikian, penelitian ini diharapkan bisa menjadi referensi untuk penelitian selanjutnya.

Kata kunci— banjir, gempa bumi, internet of things (IoT), artificial neural network, recurrent neural network, long short-term memory, nguyen-widrow, API

I. PENDAHULUAN

Indonesia termasuk ke dalam negara dengan iklim tropis dengan curah hujan yang tinggi. Semakin sempitnya lahan terbuka yang tersedia dapat menyebabkan bencana banjir. Dampak dari terjadinya banjir bisa menyebabkan masalah pada kesehatan, ekonomi, hingga menimbulkan korban jiwa [1]. Selain banjir, gempa bumi juga merupakan bencana alam yang sering melanda Indonesia. Hal tersebut karena Indonesia terletak pada pertemuan lempeng tektonik dengan jajaran gunung api aktif. Pada 11 April 2012 di Simeuleu memiliki kekuatan magnitudo 8,4 SR pada kedalaman 12,93 kilometer. Dengan besaran magnitudo dan kedalaman tersebut, gempa tersebut mengakibatkan tsunami yang menghantam Meulaboh, Sabang, dan Lahewa [2].

Mitigasi bencana adalah upaya yang termasuk ke dalam tahap awal penanggulangan bencana [3]. Tahapan penanggulangan bencana terdiri dari tiga tahapan [4], tahap awal bencana, tahap saat bencana, dan tahap setelah terjadinya bencana. Sistem mitigasi bencana dapat memanfaatkan teknologi *Internet of Things* (IoT). Dengan menggunakan teknologi IoT, pengamat dapat memonitor dari jarak jauh. Dari IoT didapatkan data sensor yang dapat dilakukan proses pengolahan data. Dalam metode pengolahan data sensor untuk peringatan bencana, penelitian sebelumnya ada yang menggunakan *fuzzy logic*. Kelemahan *fuzzy logic* yaitu tidak dapat untuk mengolah data sekuensial dengan jumlah yang banyak. Selain *fuzzy logic*, ada juga yang menggunakan model *neural network*.

Pada penelitian ini, pengolahan data sensor menggunakan model *neural network* dalam membangun program prediksi banjir, deteksi banjir, dan deteksi gempa bumi. Model *neural network* dipilih karena model *neural network* memiliki sifat *self-learning*. Sifat *self-learning* ini dibutuhkan karena model pengolahan data yang sudah dibangun bisa saja mengalami penurunan performa akurasi akibat perubahan data lapangan. Untuk membangun program prediksi tinggi air menggunakan model *Recurrent Neural Network* (RNN). Selain model RNN, dalam penelitian ini juga menggunakan model *Long Short-Term Memory* (LSTM) yang merupakan jenis dari RNN sebagai pembanding model RNN. Sedangkan untuk membangun program deteksi banjir dan gempa bumi menggunakan model *Artificial Neural Network* (ANN). Program prediksi banjir dibentuk dengan prediksi tinggi air oleh model RNN kemudian diklasifikasi bersamaan parameter lain dengan model ANN. Dalam penelitian ini juga menggunakan inisialisasi Nguyen-Widrow untuk mencari nilai awal bobot dan bias di dalam model *neural network*. Proses penerapan model ANN dan RNN ke

aplikasi dengan menghubungkan model dan aplikasi menggunakan API.

II. KAJIAN TEORI

A. 2.1 Banjir

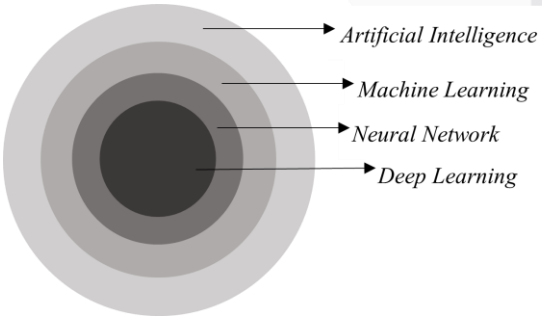
Banjir adalah meluapnya permukaan air pada aliran air hingga menggenangi daerah di sekitarnya. Meluapnya aliran air tersebut biasanya disebabkan aliran air tersebut melebihi kapasitas sistem drainase [12]. Banjir merupakan suatu fenomena bencana alam yang sering terjadi di Indonesia, terutama bagi kawasan rawan banjir. Hal tersebut karena Indonesia termasuk ke dalam negara beriklim tropis, sehingga memiliki curah hujan yang tinggi.

B. 2.2 Gempa Bumi

Indonesia terletak pada daerah pertemuan 3 lempeng, yaitu lempeng Eurasia, lempeng Indo-Australia, dan lempeng samudera Pasifik dan termasuk ke dalam kawasan *ring of fire*. Ketiga lempeng tersebut saling memberikan tekanan satu sama lain. Ketika pinggiran salah satu lempeng sudah tidak kuat, maka terjadi pelepasan energi. Hasil pelepasan energi ini menimbulkan pergerakan bumi yang kemudian dikenal sebagai gempa bumi [14]. Gempa bumi akan menimbulkan gelombang seismik. Gelombang seismik terdiri dari dua jenis, yaitu gelombang permukaan dan gelombang badan. Pada gelombang badan dapat dibedakan menjadi gelombang *Primary* (P) dan gelombang *Secondary* (S) [14]. Poisson adalah salah satu ilmuwan yang menemukan rasio kecepatan gelombang P terhadap gelombang S. Menurut Poisson, gelombang P memiliki kecepatan yang lebih tinggi daripada kecepatan gelombang S. Sehingga terdapat kemungkinan bisa mendeteksi gempa lebih dini dengan menangkap akselerasi pergerakan tanah.

C. 2.3 *Artificial Intelligence*

Sesuai dengan namanya, *artificial intelligence* atau kecerdasan buatan merupakan suatu implementasi pada komputer atau mikro-komputer, seperti gawai, agar dapat meniru kecerdasan manusia dalam memecahkan masalah [15]. *Artificial intelligence* memiliki beberapa bagian seperti digambarkan sebagai berikut [16].



GAMBAR 2.1
BAGAN BAGIAN DARI *ARTIFICIAL INTELLIGENCE*

D. 2.4 *Machine Learning*

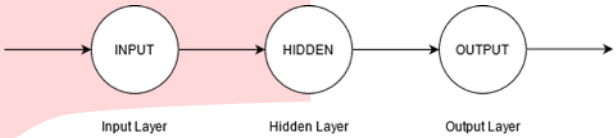
Machine learning adalah suatu konsep yang merupakan bagian dari *artificial intelligence*. Konsep dari *machine learning* yaitu membuat mesin dapat mempelajari data masukan dan data keluaran yang diberikan. Keluaran dari *machine learning* adalah suatu formula yang menggambarkan hubungan antara data masukan dan data keluaran.

E. 2.5 *Deep Learning*

Deep learning merupakan sebuah konsep pembelajaran bagian dari *machine learning*. Struktur model *neural network* pada *deep learning* lebih kompleks dibandingkan model *neural network* pada *machine learning*. Contoh model *neural network* pada *deep learning* seperti *Recurrent Neural Network* (RNN), *Convolutional Neural Network* (CNN), *Generative Adversarial Network* (GAN), dan lain-lain.

F. 2.6 *Artificial Neural Network* (ANN)

Dalam ANN terdiri dari tiga jenis lapisan, yaitu lapisan masukan, lapisan tersembunyi, dan lapisan keluaran. Secara sederhana, arsitektur dari ANN dapat dilihat pada gambar berikut.



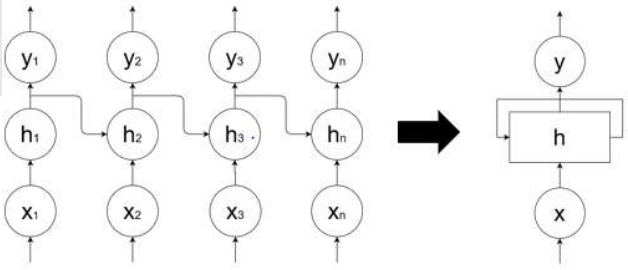
GAMBAR 2.2
ARSITEKTUR SEDERHANA ANN

1. Algoritma ANN

Pada model *neural network* terdapat dua jenis tahapan dalam membentuk sistem, yaitu *forward pass* dan *backward pass*. *Forward pass* adalah proses memasukkan data masukan dan keluaran yang alurnya dari lapisan masukan ke lapisan keluaran. *Backward pass* adalah proses mengubah nilai bobot dan bias yang alurnya dimulai dari lapisan keluaran ke lapisan masukan. Pengubahan nilai bobot dan bias saat *backward pass* menggunakan *optimizer*. Tujuan dari dua tahap ini yaitu agar model memiliki kemampuan untuk mempelajari pola data masukan dan keluaran yang diberikan.

G. 2.7 *Recurrent Neural Network* (RNN)

RNN memiliki tahapan pelatihan yang sama dengan ANN, karena sama-sama merupakan model *neural network*. RNN biasanya digunakan untuk menganalisis data yang berupa *time-series*, sehingga dibutuhkan struktur lapisan tersembunyi yang dapat menyimpan informasi dari data sebelumnya.



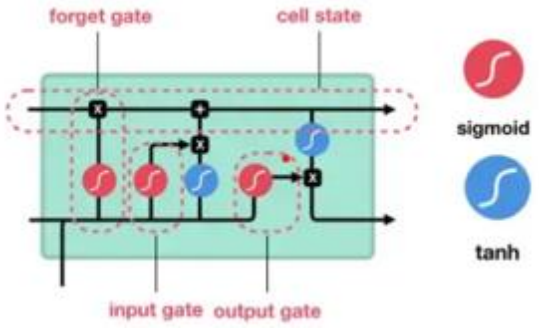
GAMBAR 2.3
ARSITEKTUR SEDERHANA RNN

Model RNN memiliki kelemahan dalam memproses dataset *time-series* yang cukup panjang. Jika dataset *time-series* yang dimasukkan ke dalam RNN cukup panjang, maka bisa terjadi dua masalah yaitu *vanishing* atau *exploding gradient*. Salah satu cara untuk mengatasi masalah tersebut

yaitu bisa menggunakan varian dari RNN yang lebih kompleks, seperti LSTM atau GRU.

H. 2.8 *Long Short-Term Memory* (LSTM)

Jika RNN diberikan data sekuensial yang banyak, maka bisa jadi RNN mengalami *vanishing gradient* atau *exploding gradient*. Sehingga RNN tidak dapat menangkap informasi penting dari *time-step* sebelumnya [19]. Solusi untuk permasalahan tersebut yaitu *Long Short-Term Memory* (LSTM). LSTM memiliki struktur yang berbeda dibandingkan dengan RNN. Dalam LSTM terdapat *cell state* dan tiga gerbang yaitu *forget gate*, *input gate*, dan *output gate*. Berikut adalah diagram struktur dari LSTM.



GAMBAR 2.4
DIAGRAM STRUKTUR LSTM [19]

I. 2.9 Algoritma Nguyen-Widrow

Algoritma Nguyen-Widrow dapat digunakan untuk menginisialisasi bobot dan bias pada model *neural network*. Berikut adalah algoritma dari Nguyen-Widrow [20].

1. Menentukan bilangan pada jarak [-0.5, 0.5] sebanyak jumlah unit masukan dikali jumlah unit tersembunyi dan disusun menjadi sebuah *array* bobot awal. Tetapi selain dalam bentuk *array*, bentuk bobot dan bias bisa dalam bentuk satu angka.
2. Menghitung bobot mutlak dengan persamaan (2.5)

$$\| V_{ij} \| = \sqrt{v_1^2 j + v_2^2 j + \cdots + v_n^2 j} \quad (2.1)$$

3 Menghitung faktor skala dengan persamaan (2.6)

$$\beta = 0.7\sqrt[n]{p} = 0.7(p)^{\frac{1}{n}} \quad (2.2)$$

Keterangan:
β : faktor skala
n : jumlah unit masukan
p : jumlah unit tersembunyi

4. Menghitung nilai bobot baru dengan persamaan (2.7)

$$V_{ij} = \frac{\beta V_{ij}(lama)}{\| V_{ij} \|} \quad (2.3)$$

Keterangan:
i : neuron *input* ke-i (i = 1,2,3,…,n)
j : neuron tersembunyi ke-j (j = 1,2,3,…,n)
V_{ij} : bobot yang baru
V_{ij} (lama) : bobot yang lama
$\| V_{ij} \|$: bobot mutlak
β : faktor skala

5. Menentukan bias menggunakan bilangan random dengan interval [-β,β].

J. Epoch dan *iteration*

Proses memasukkan data ke dalam model tidak memasukkan data langsung secara keseluruhan, tetapi mengikuti *batch size* yang telah ditentukan. Proses satu kali *epoch* terjadi ketika semua data telah selesai dimasukkan ke dalam model, atau semua *batch* sudah dimasukkan ke dalam model. Sedangkan satu kali *iteration* terjadi ketika proses *forward pass* dan *backward pass* terjadi satu kali.

K. *Batch size*

Jumlah data yang dimasukkan untuk satu kali *iteration* mengikuti *batch size* yang ditentukan. Jumlah *batch size* tidak dapat ditentukan secara pasti dan dibutuhkan proses *trial and error*. Pada umumnya untuk kasus data yang kompleks, dibutuhkan *batch size* yang banyak agar model dapat melakukan perubahan bobot dan bias lebih presisi terhadap pola data.

L. *Activation function*

Fungsi aktivasi *ReLU* akan menghasilkan keluaran 0 jika nilai masukan bernilai negatif, dan menghasilkan keluaran sesuai nilai masukan jika nilai masukan bernilai positif. Fungsi aktivasi *sigmoid* memiliki rentang keluaran 0 hingga 1 dan nilai masukan akan dikonversi sesuai skala 0 hingga 1. Fungsi aktivasi *tanh* memiliki rentang keluaran -1 hingga 1 dan nilai masukan akan dikonversi sesuai skala -1 hingga 1. Fungsi aktivasi *softmax* memiliki skala keluaran yang sama seperti fungsi aktivasi *sigmoid*. Biasanya fungsi aktivasi *sigmoid* digunakan pada klasifikasi 2 kelas, jika lebih dari 2 bisa menggunakan fungsi aktivasi *softmax*.

M. *Optimizer*

Untuk menghitung bobot dan bias yang baru, menggunakan *optimizer*. Tujuan dari *optimizer* yaitu untuk mengurangi nilai hasil dari *loss function* dan menaikkan akurasi [22]. *Optimizer* yang sering digunakan yaitu *gradient descent* dan *adaptive moment estimation* (adam).

N. *Loss function*

Untuk mencari nilai selisih antara target dan hasil klasifikasi atau hasil prediksi menggunakan *loss function*. Jenis-jenis *loss function* dapat dibedakan sesuai model *neural network* [23]. Untuk model *neural network* untuk regresi atau prediksi biasanya menggunakan *mean squared error*, *mean absolute percentage error*, atau *mean squared logarithmic error*. Sedangkan model *neural network* untuk klasifikasi biasanya menggunakan *binary crossentropy*, *categorical crossentropy*, atau *sparse categorical crossentropy*.

O. *Mean Squared Error* (MSE)

Metode yang umum digunakan untuk mengevaluasi sistem prediksi yaitu *Mean Square Error* (MSE). MSE adalah penjumlahan kuadrat *error* atau selisih antara nilai aktual dan nilai prediksi, kemudian membagi jumlah tersebut dengan banyaknya waktu data peramalan.

$$MSE = \frac{\sum(Aktual - Prediksi)^2}{n} \quad (2.4)$$

P. *Smoothing*

Dalam pengolahan data gambar untuk mengurangi gangguan biasanya dilakukan proses *smoothing*. Salah satu metode *smoothing* yang bisa digunakan yaitu Gaussian filter. Gaussian filter adalah suatu filter yang mengeluarkan keluaran berupa rata-rata dari sekelompok nilai masukan.

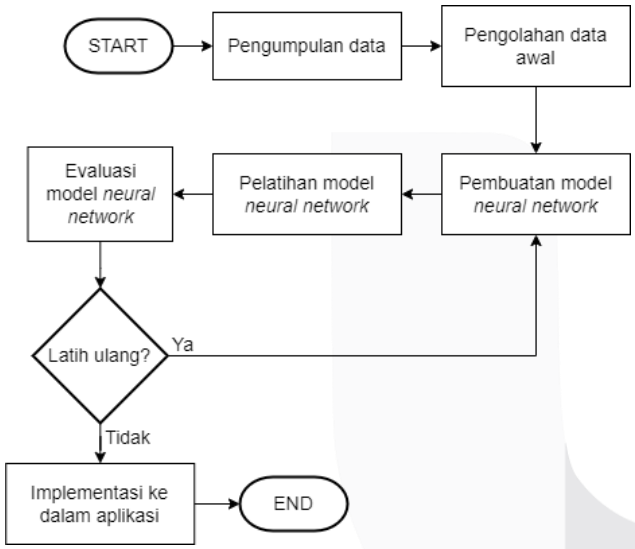
Pada sebuah baris angka, efek penggunaan Gaussian filter akan terlihat pada bentuk grafiknya.

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2.5)$$

III. METODE

A. Desain Sistem

Dalam aplikasi monitoring banjir dan gempa bumi menerapkan gabungan dari sistem prediksi banjir dan sistem deteksi banjir dan gempa bumi. Dalam penelitian ini, model RNN dan model LSTM dibandingkan dan yang terbaik digunakan untuk memprediksi ketinggian air. Hasil dari prediksi ketinggian air tersebut kemudian digabung dengan prakiraan cuaca dari laman penyedia prakiraan cuaca. Kemudian data-data tersebut diklasifikasi dengan menggunakan model ANN. Sedangkan model ANN digunakan untuk mengklasifikasi banjir dan gempa bumi sesuai data. Dalam pembuatan model *neural network* dan penerapan pada aplikasi dalam penelitian ini, terdapat diagram alir seperti pada gambar berikut.



GAMBAR 3.1
DIAGRAM ALIR PEMBUATAN DAN PENERAPAN MODEL *NEURAL NETWORK*

B. Dataset Model RNN dan LSTM Banjir

Dalam pembuatan model RNN dan LSTM, proses yang pertama kali dilakukan yaitu pengumpulan data. *Outlier* adalah suatu nilai pada data yang di luar "kebiasaan" data-data lainnya yang dapat menyebabkan gangguan dalam proses pembuatan model *neural network*. Data ketinggian air yang digunakan untuk pembuatan model RNN dan LSTM ini bersumber dari Patriot-net. Jumlah data yang berhasil terhimpun dari Patriot-net yaitu 9792 data (catatan: data tersusun tidak sesuai dengan tanggal dan waktu pada data asli dari Patriot-net, karena terdapat beberapa data kosong yang tidak dapat dilakukan interpolasi linier). Untuk menyiapkan data, dalam penelitian ini diperlukan proses modifikasi data. Beberapa data yang merupakan *outlier* dimodifikasi dengan interpolasi linier. Sebelum data dimasukkan ke dalam model, dataset dibagi 70% sebagai data *training*, 20% sebagai data *validating*, dan 10% sebagai data *testing*. Data yang dimasukkan ke dalam model RNN dan LSTM, selain data ketinggian air yaitu data *date time*. Data *date time* ditransformasi ke dalam nilai sinus dan cosinus.

C. Dataset Model ANN Banjir

Dataset yang digunakan untuk membuat model ANN banjir berbeda dengan dataset model RNN dan LSTM banjir. Jumlah dataset yang digunakan ada 2730 data, yang terdiri dari variabel *water level*, *temperature*, *humidity*, *precipitation*, dan *cloudcover*. Rentang waktu dari data yang diambil yaitu mulai pukul 00.00 1 Juli 2022 hingga pukul 22.50 19 Juli 2022. Selain itu, ada juga variabel status banjir yang digunakan untuk mengklasifikasi status tingkat kerawanan banjir. Proses pelabelan status tingkat kerawanan banjir berdasarkan asumsi pada tinggi air, dengan kriteria label aman jika tinggi air ≤ 10 cm, label siaga 1 jika 10 cm < tinggi air ≤ 15 cm, dan label siaga 2 jika tinggi air > 15 cm. Hal tersebut karena, data pada Patriot-net tidak tersedia informasi terkait adanya kejadian banjir. Selain itu, tidak ditemukan juga berita banjir pada rentang waktu dari data yang diambil di Padang, Sumatera Barat (lokasi dari sensor ketinggian air oleh Patriot-net). Variabel selain ketinggian air diambil dari laman penyedia data riwayat cuaca. Lokasi yang digunakan untuk pengambilan variabel tersebut sesuai dengan lokasi sensor ketinggian air di Patriot-net.

D. Dataset Model ANN Gempa

Dataset yang dijadikan sebagai bahan pelatihan model ANN gempa bersumber dari penelitian yang dilakukan oleh Duggal dan kawan-kawan [24]. Dataset tersebut terdiri dari variabel 3 sumbu koordinat untuk *accelerometer* yang menunjukkan percepatan pergerakan tanah, 3 sumbu koordinat untuk *gyroscope* yang menunjukkan kecepatan sudut akibat pergerakan tanah, dan hasil klasifikasi gempa. Jumlah data yang terdapat pada dataset tersebut yaitu 59.276 data.

E. Implementasi Model Pada Aplikasi

Model *neural network* yang sudah terlatih untuk dapat digunakan dalam aplikasi, butuh dikonversi format berkasnya. Pada penelitian ini model *neural network* dikonversi format berkasnya menjadi tflite. Skema dari pengimplementasiannya yaitu data-data baru yang dimasukkan ke dalam model *neural network* dibuatkan API. Model-model *neural network* yang sudah terlatih juga dibuatkan API. Pembuatan API pada data dan model tersebut menggunakan bantuan FastAPI yang menggunakan bahasa Python. Tujuan dari dibuatkan API pada model yaitu untuk mempermudah proses *maintenance* jika model perlu dilatih kembali.

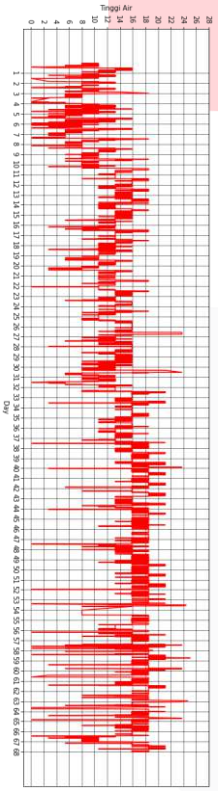
IV. HASIL DAN PEMBAHASAN

A. Model RNN Banjir

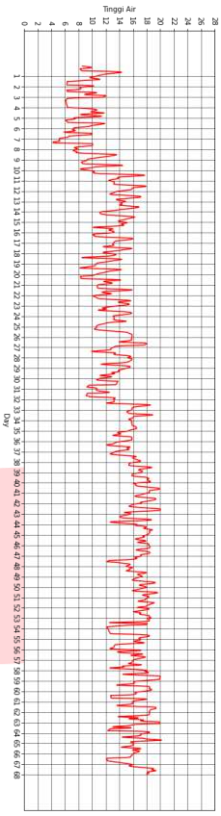
1. Pengolahan Data Awal dan Perancangan RNN Banjir

Dataset yang digunakan untuk membangun model RNN untuk prediksi banjir hanya menggunakan data ketinggian air. Hal tersebut karena untuk data prediksi cuaca sudah tersedia di dalam laman Visual Crossing. Pada Gambar 4.1 merupakan grafik ketinggian air yang didapat dari laman Patriot-net. Dari grafik tersebut, terlihat perubahan data tinggi air pada tiap waktunya sangat signifikan. Dengan

jumlah data yang hanya tersedia 9792, bisa saja pola data tersebut dapat mempengaruhi hasil prediksi RNN. Meskipun begitu, jenis data ini tetap digunakan sebagai bahan perbandingan dengan jenis data lainnya. Selanjutnya dibuat jenis data lain dengan menerapkan *smoothing* menggunakan Gaussian filter dan interpolasi linier. *Smoothing* data dengan Gaussian filter pada Gambar 4.2 menunjukkan hasil yang baik, data hasil *smoothing* cukup halus dan masih bisa mengikuti kecenderungan data asli. Setelah dilakukan proses *smoothing*, dilakukan proses pembandingan nilai rata-rata dan standar deviasi sebelum dan setelah dilakukan *smoothing*. Hal tersebut bertujuan untuk mengetahui jika hasil *smoothing* terlalu jauh dari data aslinya. Pada Gambar 4.3 dan Gambar 4.4 nilai rata-rata dan standar deviasi sebelum dan sesudah *smoothing* tidak jauh berbeda. Interpolasi linier juga dilakukan pada dataset yang sudah di-*smoothing*, tujuannya agar pola data lebih halus.



GAMBAR 4.1
DATA ASLI KETINGGIAN AIR DARI PATRIOT-NET



GAMBAR 4.2
DATA HASIL *SMOOTHING* DENGAN GAUSSIAN FILTER

```
data = df['Tinggi Air']

mean_result = mean(data)
stdev_result = stdev(data)

print("Rata-rata:", mean_result)
print("Standar deviasi:", stdev_result)
```

```
Rata-rata: 11.589766178388277
Standar deviasi: 3.5901904033504377
```

GAMBAR 4.3
NILAI RATA-RATA DAN STANDAR DEVIASI SEBELUM *SMOOTHING*

```
data = df['smoothed_data 8']

mean_result = mean(data)
stdev_result = stdev(data)

print("Rata-rata:", mean_result)
print("Standar deviasi:", stdev_result)
```

```
Rata-rata: 11.589506903127752
Standar deviasi: 2.9733744907021764
```

GAMBAR 4.4
NILAI RATA-RATA DAN STANDAR DEVIASI SESUDAH *SMOOTHING*

Sebelum dataset dimasukkan ke dalam model RNN dan LSTM, dataset juga dilakukan proses normalisasi. Proses normalisasi dilakukan untuk menyederhanakan variasi dari suatu variabel. Model RNN memiliki keterbatasan pada jumlah data optimal. Sehingga pada kasus tertentu jika terdapat banyak data, model RNN bisa gagal dalam mempelajari pola. Model LSTM juga dibentuk sebagai bahan pembanding model RNN. Parameter model LSTM yang dibentuk mengikuti parameter dari model RNN. Setelah melalui proses *trial and error*, berikut adalah parameter

model RNN dan LSTM yang digunakan untuk prediksi banjir.

TABEL 4.1
PARAMETER MODEL RNN DAN LSTM PREDIKSI BANJIR

Jumlah unit masukan	3
Jumlah unit keluaran	3
Jumlah lapisan tersembunyi	2
Jumlah unit tersembunyi	256 + 128
Batch size	512
Fungsi *loss*	*Mean Squared Logarithmic Error*
Optimizer	*Adam*
Rentang masukan	5 hari atau 720 data
Rentang keluaran	2 hari atau 288 data
Fungsi aktivasi	*sigmoid*
Split data	*training* : 70% *validating* : 20% *testing* : 10%
Stop condition	MSE < 0,1100
Max epoch	500

Untuk melakukan pelatihan model RNN dan LSTM prediksi banjir, dalam penelitian ini menggunakan sebuah fungsi *callback* yang berfungsi untuk memberhentikan proses *training*.

```
hidden_neuron = 256
hidden_neuron_2 = 128
output_neuron = 3

# Nguyen-Widrow
faktor_skala = 0.7*((hidden_neuron_2)**(1/hidden_neuron))

bobot_lama = []
bobot_lama_kuadrat = []
x = round(random.uniform(-0.5, 0.5),2)

for i in range((hidden_neuron_2*hidden_neuron)):
  bobot_lama.append(x)

for i in bobot_lama:
  i = i**2
  bobot_lama_kuadrat.append(i)

bobot_mutlak = math.sqrt(sum(bobot_lama_kuadrat))
bobot = (faktor_skala*x)/(bobot_mutlak)
bias = random.uniform(-(faktor_skala), faktor_skala)
```

GAMBAR 4.5
ALGORITMA NGUYEN-WIDROW PADA MODEL RNN DAN LSTM BANJIR

```
# menggunakan SimpleRNN
rnn_model = Sequential([
  tf.keras.layers.SimpleRNN(256, return_sequences=True),
  tf.keras.layers.SimpleRNN(128, return_sequences=False,
                            kernel_initializer=tf.keras.initializers.Constant(bobot),
                            bias_initializer=tf.keras.initializers.Constant(bias),
                            activation='sigmoid'),
  tf.keras.layers.Dense(OUT_STEPS*3),
  tf.keras.layers.Reshape([OUT_STEPS, 3])
])
```

GAMBAR 4.6
ARSITEKTUR MODEL RNN BANJIR PADA TENSORFLOW

```
# menggunakan LSTM
rnn_model = Sequential([
  tf.keras.layers.LSTM(256, return_sequences=True),
  tf.keras.layers.LSTM(128, return_sequences=False,
                       kernel_initializer=tf.keras.initializers.Constant(bobot),
                       bias_initializer=tf.keras.initializers.Constant(bias),
                       activation='sigmoid'),
  tf.keras.layers.Dense(OUT_STEPS*3),
  tf.keras.layers.Reshape([OUT_STEPS, 3])
])
```

GAMBAR 4.7
ARSITEKTUR MODEL LSTM BANJIR PADA TENSORFLOW

B. Pengujian Model

Model RNN dan model LSTM yang sudah dibentuk kemudian dimasukkan dataset dan dilakukan pengujian. Masing-masing model tersebut dimasukkan dataset asli dan dataset hasil *smoothing*. Berikut adalah hasil dari pengujian model RNN dan model LSTM tersebut.

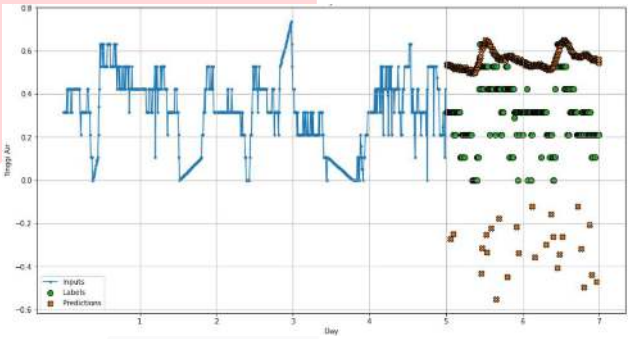
a. Model RNN dengan Data Asli

Nilai MSE pada hasil *training* dan hasil *testing* menunjukkan terjadinya *overfitting*. Hal tersebut karena nilai MSE hasil *training* memiliki nilai yang lebih rendah dibanding hasil *testing*. Meskipun nilai MSE kecil, tetapi banyak data hasil prediksi yang kurang dekat dengan target. Sehingga menyebabkan grafik hasil prediksi yang dihasilkan kurang bagus. Berikut adalah nilai MSE dan grafik hasil prediksi model RNN dengan data asli.

```
Epoch 500/500
12/12 [==============================] - 16s 1s/step - loss: 0.0258 - mean_squared_error: 0.1158 - val_loss: 0.0277 - v
al_mean_squared_error: 0.1219

Testing...
2/2 [==============================] - 0s 137ms/step - loss: 0.0277 - mean_squared_error: 0.1219
```

GAMBAR 4.8
NILAI MSE MODEL RNN DENGAN DATA ASLI



GAMBAR 4.9
GRAFIK HASIL PREDIKSI MODEL RNN DENGAN DATA ASLI

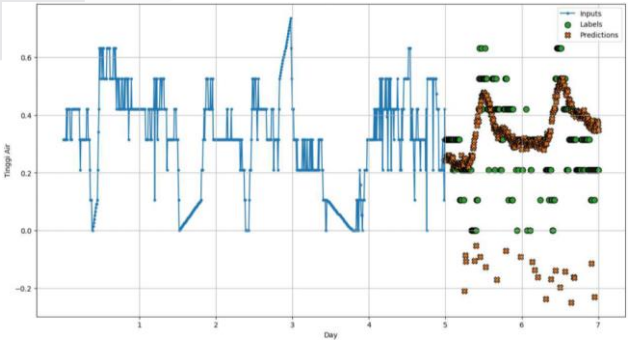
b. Model LSTM dengan Data Asli

Karakteristik dari nilai MSE *training* dan *testing* juga menunjukkan terjadi *overfitting*. Tetapi dari grafik hasil prediksi, model LSTM memiliki grafik hasil prediksi yang lebih baik. Hal tersebut bisa terjadi mungkin karena model LSTM memiliki *forget gate*. Hasil pelatihan model RNN dan model LSTM dengan data asli menunjukkan bahwa model membutuhkan data untuk di-*smoothing*. Berikut adalah nilai MSE dan grafik hasil prediksi model LSTM dengan data asli.

```
Epoch 304/500
12/12 [==============================] - 149s 12s/step - loss: 0.0262 - mean_squared_error: 0.1090 - val_loss: 0.0302 -
val_mean_squared_error: 0.1214

Testing...
2/2 [==============================] - 8s 4s/step - loss: 0.0302 - mean_squared_error: 0.1214
```

GAMBAR 4.10
NILAI MSE MODEL LSTM DENGAN DATA ASLI



GAMBAR 4.11
GRAFIK HASIL PREDIKSI MODEL LSTM DENGAN DATA ASLI

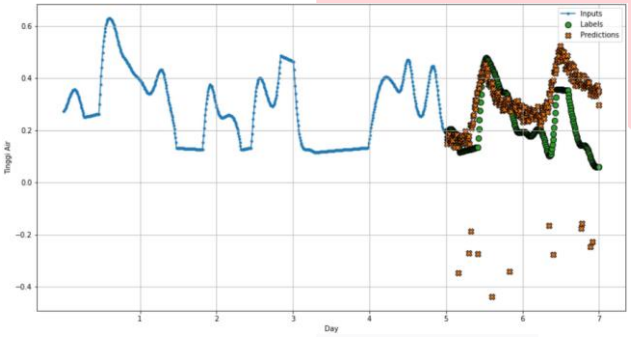
3. Model RNN dengan Data *Smoothing*

Jika melihat nilai MSE, nilai MSE hasil *training* dan hasil *testing* menunjukkan terjadinya *overfitting*. Pada hasil pengujian nilai MSE model RNN dengan data *smoothing* memiliki nilai yang cukup besar, tetapi grafik hasil prediksinya bagus. Hal tersebut menunjukkan terdapat nilai prediksi yang sangat jauh dari nilai aktual meskipun hanya ada beberapa. Selain menggunakan data asli dan data *smoothing*, pada Gambar 4.14 dan Gambar 4.15 merupakan hasil pengujian model RNN dengan data sinus. Hasil tersebut sebagai pembanding jika data yang digunakan untuk pelatihan model RNN memiliki periodik yang bagus. Berikut adalah nilai MSE dan grafik hasil prediksi model RNN dengan data *smoothing*.

```
Epoch 500/500
12/12 [==============================] - 14s 1s/step - loss: 0.0443 - mean_squared_error: 0.3534 - val_loss: 0.0455 - v
al_mean_squared_error: 0.3716

Testing...
2/2 [==============================] - 0s 129ms/step - loss: 0.0455 - mean_squared_error: 0.3716
```

GAMBAR 4.12
NILAI MSE MODEL RNN DENGAN DATA *SMOOTHING*

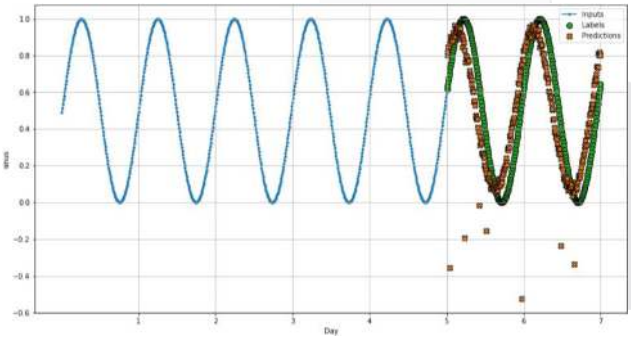


GAMBAR 4.13
GRAFIK HASIL PREDIKSI MODEL RNN DENGAN DATA *SMOOTHING*

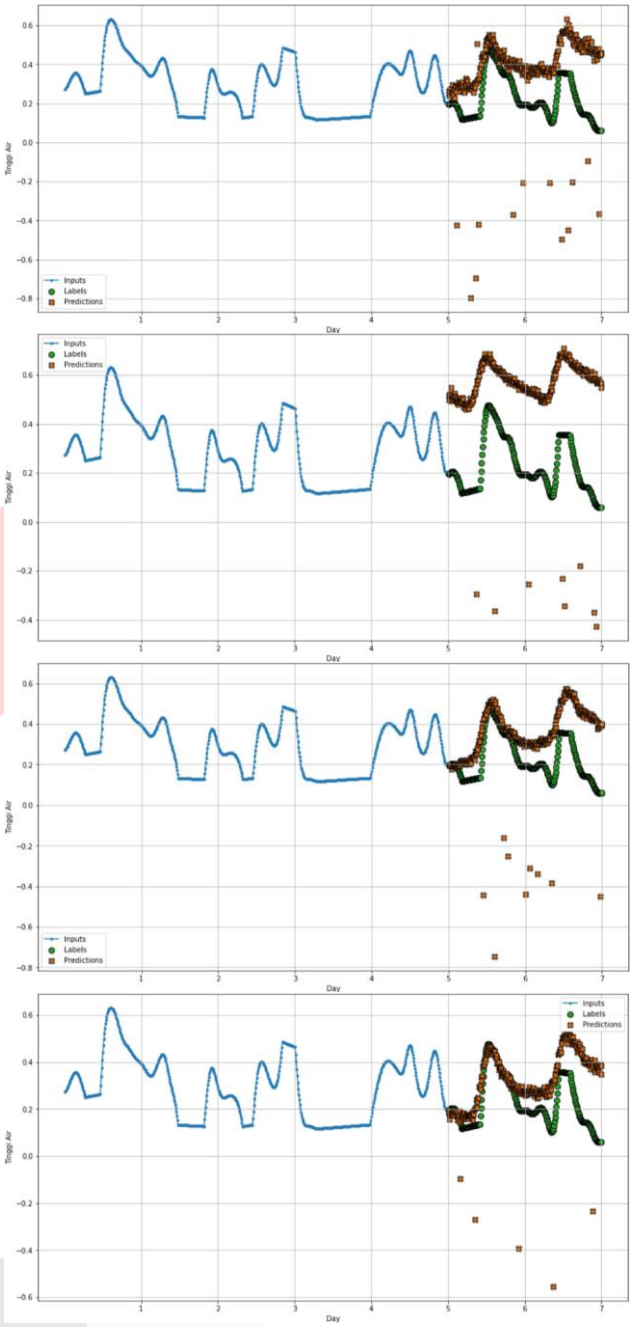
```
Epoch 500/500
12/12 [==============================] - 14s 1s/step - loss: 0.0213 - mean_squared_error: 0.1351 - val_loss: 0.0303 - v
al_mean_squared_error: 0.1655

Testing...
2/2 [==============================] - 0s 113ms/step - loss: 0.0303 - mean_squared_error: 0.1655
```

GAMBAR 4.14
NILAI MSE MODEL RNN DENGAN DATA SINUS



GAMBAR 4.15
GRAFIK HASIL PREDIKSI MODEL RNN DENGAN DATA SINUS



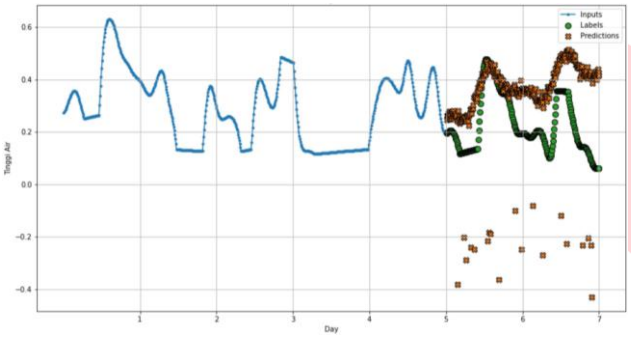
GAMBAR 4.16
GRAFIK HASIL PREDIKSI MODEL RNN DALAM 4 KALI PERCOBAAN

Dilakukan pengujian model RNN dengan data *smoothing* dengan 4 kali percobaan lagi. Hasil 4 kali percobaan pada model RNN dengan data *smoothing* terdapat pada Gambar 4.16. Dalam 5 kali percobaan, model RNN dengan data *smoothing* berhasil 3 kali menghasilkan grafik hasil prediksi yang baik. Model ini yang diterapkan pada aplikasi, karena model ini memiliki kemampuan yang baik sesuai percobaan. Data hasil prediksi kemudian dilakukan proses denormalisasi untuk mengembalikan nilai hasil prediksi sesuai skala aslinya. Data hasil prediksi yang memiliki nilai negatif diubah ke nilai terendah dari data tinggi air.

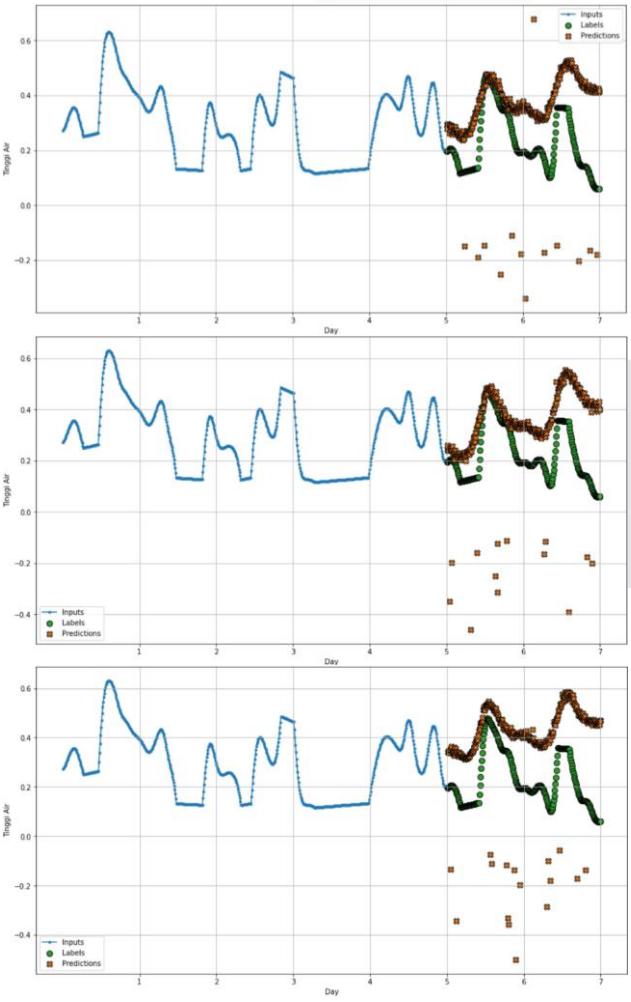
4. Model LSTM dengan Data *Smoothing*

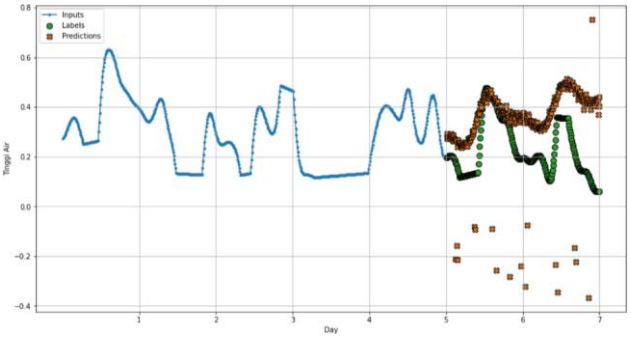
Karakteristik nilai MSE masih menunjukkan terjadinya *overfitting*. Dibandingkan dengan model RNN dengan data *smoothing*, model LSTM dengan data *smoothing* memiliki grafik hasil prediksi yang kalah baik. Hal tersebut mungkin terjadi karena *forget gate* pada model LSTM membuang beberapa informasi yang sebenarnya dibutuhkan untuk proses perubahan nilai bobot dan bias. Berikut adalah nilai MSE dan grafik hasil prediksi model LSTM dengan data *smoothing*.

GAMBAR 4.17
NILAI MSE MODEL LSTM DENGAN DATA *SMOOTHING*



GAMBAR 4.18
GRAFIK HASIL PREDIKSI MODEL LSTM DENGAN DATA *SMOOTHING*





GAMBAR 4.19
GRAFIK HASIL PREDIKSI MODEL LSTM DALAM 4 KALI PERCOBAAN

Dilakukan pengujian model LSTM dengan data *smoothing* dengan 4 kali percobaan lagi. Hasil 4 kali percobaan pada model LSTM dengan data *smoothing* terdapat pada Gambar 4.19. Dalam 5 kali percobaan, model LSTM dengan data *smoothing* hanya berhasil 1 kali menghasilkan grafik hasil prediksi yang baik.

B. Model ANN Banjir

1. Pengolahan Data Awal dan Perancangan ANN Banjir

Dataset yang digunakan untuk membuat model ANN pada penelitian ini menggunakan data dari Patriot-net dan Visual Crossing. Bentuk pengolahan data awal yang dilakukan yaitu proses normalisasi. Proses normalisasi dilakukan untuk menyederhanakan variasi dari suatu variabel. Setelah melalui proses *trial and error*, berikut adalah parameter model ANN yang digunakan untuk deteksi banjir.

TABEL 4.2
PARAMETER MODEL ANN DETEKSI BANJIR

Jumlah unit masukan	5
Jumlah unit keluaran	3
Jumlah lapisan tersembunyi	1
Jumlah unit tersembunyi	8
Batch size	1
Fungsi *loss*	*Categorical Crossentropy*
Optimizer	*Adam*
Fungsi aktivasi	*relu* : pada lapisan tersembunyi *softmax* : pada lapisan keluaran
Split data	*training* : 50% *testing* : 50%
Stop condition	*Accuracy* > 0,9900
Max epoch	200

Untuk melakukan pelatihan model ANN deteksi banjir, dalam penelitian ini menggunakan sebuah fungsi *callback* yang memberhentikan proses *training* sesuai syarat kondisi.

```
input_neuron = 5
hidden_neuron = 8
output_neuron = 3

# Tentukan nilai beta
beta = 0.7 * np.power(hidden_neuron, 1/input_neuron)

# Inisialisasi bobot dan bias
bobot = np.random.uniform(-0.5, 0.5, size=(hidden_neuron, input_neuron))
bias = np.random.uniform(-(beta), beta, size=(hidden_neuron, 1))

# Hitung nilai bobot mutlak
bobot_norm = np.linalg.norm(bobot)

# Inisialisasi ulang bobot-bobot dari unit input
bobot = bobot / bobot_norm * beta
```

GAMBAR 4.20
ALGORITMA NGUYEN-WIDROW PADA MODEL ANN BANJIR

```
model = Sequential()
model.add(Dense(8, input_dim= 5,
                activation= 'relu',
                kernel_initializer= tf.keras.initializers.Constant(bobot),
                bias_initializer= tf.keras.initializers.Constant(bias)))
model.add(Dense(3, activation= 'softmax'))
```

GAMBAR 4.21
ARSITEKTUR MODEL ANN BANJIR PADA TENSORFLOW

2. Pengujian Model

Arsitektur model ANN banjir yang sudah dibentuk dilakukan pengujian untuk melihat performa model. Pengujian dilakukan dengan melakukan percobaan berulang dengan parameter yang sama sebanyak 10 kali. Tujuan pengujian dengan cara tersebut yaitu untuk melihat performa model. Berikut adalah tabel hasil pengujian model ANN banjir dalam 10 kali percobaan.

TABEL 4.3
HASIL PENGUJIAN MODEL ANN BANJIR

Percobaan ke-	***accuracy train***	***accuracy test***
percobaan 1	0,9949	0,9993
percobaan 2	0,9934	0,9993
percobaan 3	0,9934	0,9941
percobaan 4	0,9912	0,9993
percobaan 5	0,9993	1,0000
percobaan 6	0,9993	1,0000
percobaan 7	0,9993	1,0000
percobaan 8	1,0000	0,9993
percobaan 9	0,9993	0,9993
percobaan 10	0,9985	1,0000
Rata-rata	**0,9969**	**0,9991**

Tabel di atas hasil pengujian model ANN banjir dengan parameter yang sudah ditentukan. Dari tabel tersebut terlihat bahwa terjadi 2 kali *overfitting* tepatnya pada percobaan ke-8 dan 9. Model ini memiliki rata-rata akurasi *training* 0,9969 dan akurasi *testing* 0,9991 dari 10 kali percobaan.

C. Model ANN Gempa

1. Pengolahan Data Awal dan Perancangan ANN Gempa

Dataset yang digunakan untuk membentuk model ANN gempa diambil dari jurnal penelitian oleh Duggal dan kawan-kawan [24]. Untuk model ANN gempa hanya dilakukan proses normalisasi sebagai bentuk pengolahan data awal. Setelah melalui proses *trial and error*, berikut adalah parameter model ANN yang digunakan untuk deteksi gempa.

TABEL 4.4
PARAMETER MODEL ANN DETEKSI GEMPA

Jumlah unit masukan	6
Jumlah unit keluaran	2
Jumlah lapisan tersembunyi	1
Jumlah unit tersembunyi	8
Batch size	1
Fungsi *loss*	*Categorical Crossentropy*
Optimizer	*Adam*
Fungsi aktivasi	*relu* : pada lapisan tersembunyi *softmax* : pada lapisan keluaran
Split data	*training* : 50% *testing* : 50%
Stop condition	*Accuracy* > 0,9900
Max epoch	200

Untuk melakukan pelatihan model ANN deteksi gempa, dalam penelitian ini menggunakan sebuah fungsi *callback* yang memberhentikan proses *training* sesuai syarat kondisi.

```
input_neuron = 6
hidden_neuron = 8
output_neuron = 2

# Tentukan nilai beta
beta = 0.7 * np.power(hidden_neuron, 1/input_neuron)

# Inisialisasi bobot dan bias
bobot = np.random.uniform(-0.5, 0.5, size=(hidden_neuron, input_neuron))
bias = np.random.uniform(-(beta), beta, size=(hidden_neuron, 1))

# Hitung nilai bobot mutlak
bobot_norm = np.linalg.norm(bobot)

# Inisialisasi ulang bobot-bobot dari unit input
bobot = bobot / bobot_norm * beta
```

GAMBAR 4.22
ALGORITMA NGUYEN-WIDROW PADA MODEL ANN GEMPA

```
model = keras.Sequential([
    keras.layers.Dense(8, input_dim=6,
                       activation='relu',
                       kernel_initializer=tf.keras.initializers.Constant(bobot),
                       bias_initializer=tf.keras.initializers.Constant(bias)),
    keras.layers.Dense(2, activation='softmax')
])
```

GAMBAR 4.23
ARSITEKTUR MODEL ANN GEMPA PADA TENSORFLOW

2. Pengujian Model

Arsitektur model ANN gempa yang sudah dibentuk dilakukan pengujian untuk melihat performa model. Pengujian dilakukan dengan melakukan percobaan berulang dengan parameter yang sama sebanyak 10 kali. Tujuan pengujian dengan cara tersebut yaitu untuk melihat performa model. Berikut adalah tabel hasil pengujian model ANN gempa dalam 10 kali percobaan.

TABEL 4.5
HASIL PENGUJIAN MODEL ANN GEMPA

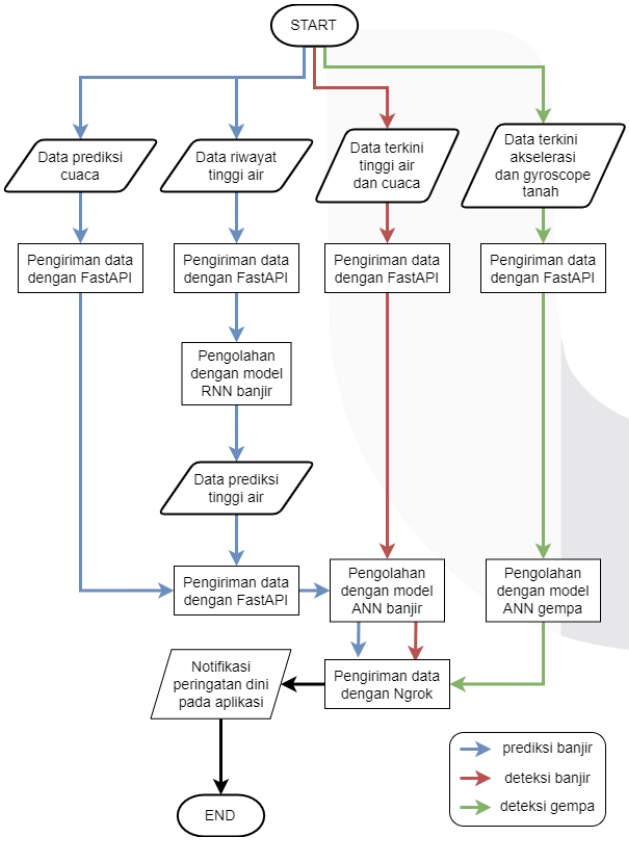
Percobaan ke-	***accuracy train***	***accuracy test***
percobaan 1	0,9982	0,9991
percobaan 2	0,9977	0,9992
percobaan 3	0,9978	0,9981
percobaan 4	0,9901	0,9977
percobaan 5	0,9983	0,9991

percobaan 6	0,9925	0,9979
percobaan 7	0,9978	0,9989
percobaan 8	0,9982	0,9987
percobaan 9	0,9982	0,9992
percobaan 10	0,9982	0,9989
Rata-rata	**0,9967**	**0,9987**

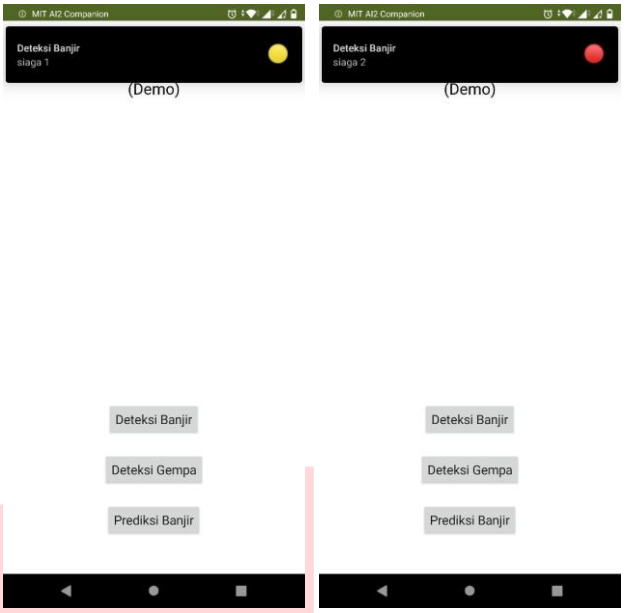
Tabel di atas merupakan hasil pengujian model ANN gempa dalam 10 kali percobaan. Model ini memiliki kemampuan yang sangat baik, karena dalam pengujian tidak ditemukan terjadinya kondisi *overfitting*.

D. Penerapan Model *Neural Network* Pada Aplikasi

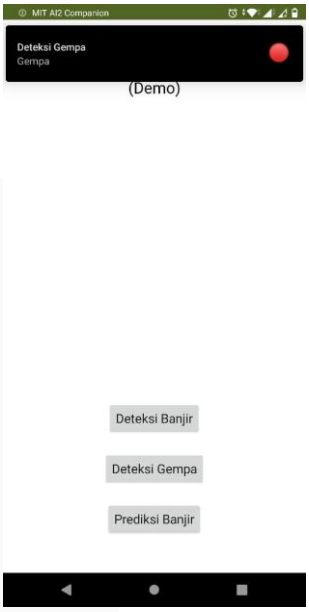
Penerapan model *neural network* ke dalam aplikasi tidak bisa langsung menggunakan FastAPI. API model tidak bisa langsung terhubung ke dalam aplikasi, karena API dari FastAPI masih bersifat *localhost*. Dengan begitu dibutuhkan Ngrok yang membuat API bisa diakses secara publik. Proses pembentukan aplikasi dilakukan melalui MIT App Inventor. Untuk pembentukan sistem prediksi banjir, model yang digunakan adalah model RNN yang dilatih dengan data hasil modifikasi. Gambar 4.24 adalah gambar skema penerapan model *neural network* ke aplikasi. Untuk tampilan aplikasi terdapat pada Gambar 4.25, Gambar 4.26, dan Gambar 4.27.



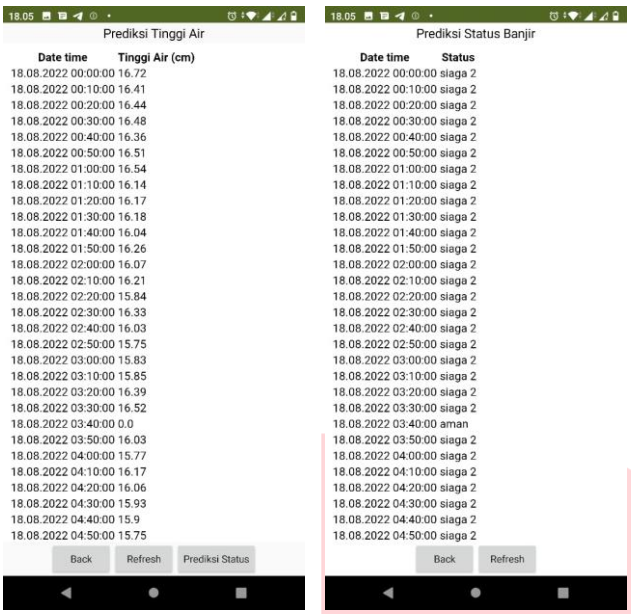
GAMBAR 4.24
DIAGRAM ALIR KERJA APLIKASI



GAMBAR 4.25
TAMPILAN DETEKSI BANJIR



GAMBAR 4.26
TAMPILAN DETEKSI GEMPA



GAMBAR 4.27
TAMPILAN PREDIKSI BANJIR

V. KESIMPULAN

Berdasarkan pembahasan pada bab sebelumnya didapatkan beberapa kesimpulan. Pertama, model ANN, RNN, dan LSTM dalam penelitian ini dirancang dengan proses *trial and error*. Hasil pengujian model ANN untuk deteksi banjir memiliki kemampuan yang baik. Dalam 10 kali percobaan, hanya terjadi 2 kali kondisi *overfitting*. Model ANN untuk deteksi banjir memiliki rata-rata akurasi *training* 0.9969 dan akurasi *testing* 0.9991. Hasil pengujian model ANN untuk deteksi gempa memiliki kemampuan yang sangat baik. Dalam 10 kali percobaan, tidak ditemukan terjadinya kondisi *overfitting*. Model ANN untuk deteksi gempa memiliki rata-rata akurasi *training* 0.9967 dan akurasi *testing* 0.9987. Hasil pengujian antara model RNN dan LSTM, model RNN memiliki kemampuan yang lebih baik daripada model LSTM. Grafik hasil prediksi model RNN menghasilkan hasil yang lebih baik daripada model LSTM. Dalam 5 kali percobaan, model RNN berhasil menghasilkan 3 kali grafik hasil prediksi yang baik berbanding 1 kali. Kedua, model ANN dan RNN dapat diterapkan ke dalam aplikasi dengan skema data sensor dihubungkan ke model *neural network* dengan bantuan FastAPI, sedangkan model *neural network* dihubungkan ke aplikasi dengan bantuan Ngrok. Ketiga, karena keterbatasan pada data pelatihan yang digunakan untuk melatih model *neural network* yang dibangun. Model *neural network* yang sudah terbentuk belum dapat diterapkan pada aplikasi untuk digunakan secara publik. Meskipun begitu, diharapkan penelitian ini bisa menjadi referensi untuk penelitian selanjutnya.

REFERENSI

[1] D. Danang, S. Suwardi and I. A. Hidayat, "Mitigasi Bencana Banjir dengan Sistem Informasi Monitoring dan Peringatan Dini Bencana menggunakan Microcontroller Arduino Berbasis IoT," *TEKNIK,* vol. 1, no. 40, pp. 55-60, 2019.

[2] U. Setiyono, I. Gunawan, Priyobudi, T. Yatimantoro, R. T. Imananta, M. Ramdhan, Hidayanti, S. Anggraini, R. H. Rahayu, P. Hawati, D. S. Yogaswara, A. M. Julius, M. Apriani, M. Harvan, G. Simangunsong and T. Kriswinarso, Katalog Gempabumi Signifikan dan Merusak 1821-2018, 1st ed., Jakarta Pusat: Pusat Gempabumi dan Tsunami Kedeputian Bidang Geofisika Badan Meteorologi Klimatologi dan Geofisika , 2019.

[3] D. Amalia, "StudioBelajar," 2021. [Online]. Available: https://www.studiobelajar.com/mitigasi-bencana/. [Accessed 29 Maret 2021].

[4] Y. C. Ginanjar, "Kebencanaan Babel," 2018. [Online]. Available: https://bpbd.babelprov.go.id/proses-penanggulangan-bencana/. [Accessed 29 Maret 2021].

[5] B. M. Ramageri, "DATA MINING TECHNIQUES AND APPLICATIONS," *Indian Journal of Computer Science and Engineering,* vol. 1, no. 4, pp. 301-305.

[6] R. N. Putri and D. Setiawan, "PROTOTIPE JARINGAN SYARAF TIRUAN UNTUK MENDETEKSI BANJIR MENGGUNAKAN METODE BACKPROPAGATION," *Journal Of Information System And Informatics Engineering,* vol. 1, no. 2, pp. 144-149, 2017.

[7] M. A. Azizulhaq, "DASHBOARD SISTEM PERINGATAN DINI PREDIKSI BANJIR MENGGUNAKAN METODE RADIAL BASIS FUNCTION BERBASIS WEB," *e-Proceeding of Engineering,* vol. 8, no. 1, pp. 334-341, 2021.

[8] S. Al-Ayubi, "ESTIMASI MAGNITUDO GEMPA BUMI DARI SINYAL SEISMIK GELOMBANG P MENGGUNAKAN METODE JST BACKPROPAGATION," *e-Proceeding of Engineering ,* vol. 7, no. 2, pp. 4624-4632, 2020.

[9] A. Pranesthi, "PROTOTYPE SISTEM PERINGATAN DINI GEMPA BUMI BERDASARKAN SINYAL GEOMAGNETIK DAN ANALISA POLA WAKTU MUSIM KEMARAU DENGAN ALGORITMA BACKPROPAGATION NETWORK BERBASIS INTERNET OF THINGS," *e-Proceeding of Engineering,* vol. 7, no. 1, pp. 1676-1683, 2020.

[10] F. N. Elrizki, "PROTOTYPE SISTEM PERINGATAN DINI GEMPA BUMI BERDASARKAN SINYAL GEOMAGNETIK DAN ANALISA POLA WAKTU MUSIM KEMARAU DENGAN ALGORITMA RADIAL BASIS FUNCTION NETWORK BERBASIS INTERNET OF THINGS," *e-Proceeding of Engineering ,* vol. 7, no. 1, pp. 1668-1675, 2020.

[11] Suyanto, K. N. Ramadhani and S. Mandala, Deep Learning Modernisasi Machine Learning Untuk Big Data, Bandung: Informatika, 2019.

[12] A. Rosyidie, "Banjir: Fakta dan Dampaknya, Serta Pengaruh dari Perubahan Guna Lahan," *Jurnal*

Perencanaan Wilayah dan Kota, vol. 24, no. 3, pp. 241-249, 2013.

[13] P. N. Rahardjo, "7 PENYEBAB BANJIR DI WILAYAH PERKOTAAN YANG PADAT PENDUDUKNYA," *JAI,* vol. 7, no. 2, pp. 205-213, 2014.

[14] Sunarjo, M. T. Gunawan and S. Pribadi, GEMPA BUMI EDISI POPULER, Jakarta: Badan Meteorologi Klimatologi dan Geofisika, 2012.

[15] J.-P. Haton, "A brief introduction to artificial intelligence," *IFAC Proceedings Volumes,* vol. 39, no. 4, pp. 8-16, 2006.

[16] E. Kavlakoglu, "IBM," 27 Mei 2020. [Online]. Available: https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks. [Accessed 27 November 2022].

[17] antjef, "deepomatic," 14 Desember 2017. [Online]. Available: https://deepomatic.com/introduction-to-deep-learning-ai-for-dummies. [Accessed 27 November 2022].

[18] Y. Bohra, "Analytics Vidhya," 23 Juni 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/. [Accessed 27 November 2022].

[19] M. Phi, "Towards Data Science," 25 September 2018. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. [Accessed 27 November 2022].

[20] K. Mishra, N. K. Mittal and M. H. Mirja, "Image Compression Using Multilayer Feed Forward Artificial Neural Network with Nguyen Widrow Weight Initialization Method," *International Journal of Emerging Technology and Advanced Engineering,* vol. 4, no. 4, pp. 475-480, 2014.

[21] H. F. Mahfuzh, D. Widiyanto and N. Chamidah, "PENGARUH ALGORITMA INISIALISASI NGUYEN-WIDROW TERHADAP ALGORITMA BACKPROPAGATION DALAM PREDIKSI INDEKS HARGA KONSUMEN (IHK)," *Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA),* pp. 707-720, 2020.

[22] A. Gupta, "Analytics Vidhya," 24 Mei 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/. [Accessed 28 November 2022].

[23] D. Mwiti, "Neptune Labs," 16 Desember 2022. [Online]. Available: https://neptune.ai/blog/keras-loss-functions. [Accessed 20 Desember 2022].

[24] R. Duggal, N. Gupta, A. Pandya, P. Mahajan, K. Sharma, T. Kaundal and P. Angra, "Building structural analysis based Internet ofThings network assisted earthquake detection," *Elsevier,* vol. 19, 2022.