BAB II
TINJAUAN PUSTAKA

2.1 Kajian Pustaka

Penelitian terkait prediksi dengan menggunakan Deep Learning sendiri sudah banyak dilakukan dan diterapkan secara luas di berbagai bidang kehidupan. Pada penelitian-penelitian sebelumnya yang menggunakan LSTM dan GRU, dijadikan dasar untuk penelitian sehingga menjadi lebih baik lagi dan dapat membantu penelitian selanjutnya. Berikut penelitian terdahulu menurut penulis relevan dengan penelitian, yang disajikan pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No	Peneliti	Objek	*Dataset*	Metode	Hasil	GAP
1	Pengtao Jia, Hangduo Liu, Sujian Wang dan Peng Wang (2020) [23]	Peramalan konsentrasi gas tambang	Data konsentrasi gas tambang dari 10419 titik pada Januari hingga Maret 2014	SVR, BPNN, RNN, LSTM dan GRU	Hasil penelitian menunjukkan bahwa model GRU lebih dominan dari model yang lain dengan nilai mse training 0.02836 dan testing 0.02405 dalam waktu 1172.87 detik.	Dalam penelitian ini melakukan prediksi harga komoditas pangan dengan menggunakan algoritma LSTM dan GRU
2	Anil Utku dan Umit Can (2022) [22]	Peramalan kualitas udara	Dataset konsentrasi PM2.5 dari 7 stasiun berbeda di London dari 1 Januari sampai 1 Mei 2019 (120 hari)	Decision Tree, Extra Tree, KNN, Random Forest, SVM, XGBoost, MLP, CNN, RNN, GRU, LSTM	Hasil penelitian menunjukan bahwa model LSTM lebih dominan dibandingkan model lainnya dengan nilai mse, rmse, mae dan R^2 sebesar 0.330, 0.574, 0.292 dan 0.988	Dalam penelitian ini melakukan prediksi harga komoditas pangan dengan menggunakan algoritma LSTM dan GRU dan di evaluasi dengan menggunakan mse.
3	Nanda Kurnia	Peramalan harga emas	Data harga emas dari	LSTM dan GRU	Hasilnya menunjukkan	Dalam penelitian ini

No	Peneliti	Objek	*Dataset*	Metode	Hasil	GAP
	Agusmawati, Fitwatul Khoiriyah, Abu Tholib (2023) [20]		Kaggle yang berjumlah 8277 hari (4 Januari 2000 sampai 2 September 2022) dengan total 5703 data.		bahwa model GRU lebih dominan untuk melakukan prediksi harga emas dibandingkan model LSTM dengan nilai mae, rmse dan mape sebesar 0.0447, 0.0545 dan 6.0688%	melakukan prediksi harga telur ayam ras dan menggunakan mse sebagai metrik evaluasinya.
4	Unix Izyah Arfianti, Dian Candra Rini Novitasari, Nanang Widodo, Moh. Hafiyusholeh, Wika Dianita Utami (2021) [21]	Prediksi bilangan sunspot	Dataset bilangan sunspot bulanan dari web SILSO (Sunspot Index and Long-Term Solar Observation) sebanyak 3240 data sejak Januari 1750 hingga Desember 2019.	LSTM dan GRU	Hasil menunjukkan bahwa model GRU lebih dominan untuk melakukan prediksi bilangan sunspot dengan mape sebesar 7.171% sementara LSTM sebesar 9.9557%	Dalam penelitian ini melakukan prediksi harga telur ayam ras yang dievaluasi menggunakan mse.
5	Lailan Sahrina Hasibuan dan Yanda Novialdi [16]	Prediksi harga minyak goreng curah dan kemasan	Dataset harga minyak goreng curah dan kemasan pada pasar tradisional Provinsi Jawa Barat dari 1 November 2021 sampai 31 Agustus 2022	LSTM	Hasil menunjukkan bahwa model LSTM mampu mengenali pola harga minyak goreng baik curah maupun kemasan dengan nilai NRMSE 0.019 pada minyak goreng curah dan 0.037 pada	Dalam penelitian ini melakukan prediksi harga telur ayam ras dengan menggunakan algoritma LSTM dan GRU, kemudian dievaluasi menggunakan mse.

No	Peneliti	Objek	*Dataset*	Metode	Hasil	GAP
					minyak goreng kemasan.	
6	Yadi Karyadi dan Handri Santoso (2022) [24]	Prediksi kualitas udara	Dataset kualitas udara harian kota Bandung dalam kurun waktu 2019 dengan 20 variabel	LSTM, LSTM-Bi dan GRU	Hasil menunjukkan bahwa model LSTM lebih dominan dibandingkan model lainnya dengan nilai rmse dari variabel suhu, kelembapan dan ISPU sebesar 3.18, 6.96 dan 1.84.	Dalam penelitian ini melakukan prediksi harga telur ayam ras dengan menggunakan algoritma LSTM dan GRU, kemudian dievaluasi menggunakan mse.
7	Matthew Oni, Manatap Dolok Lauro dan Teny Handhayani (2023) [14]	Prediksi harga pangan di kota Bandung	Dataset harga beras, daging ayam, telur ayam, bawang merah dan bawang putih	GRU	Hasil menunjukkan bahwa model GRU dapat melakukan prediksi dengan nilai mae dan mape 12.8 dan 0,10 untuk beras, 12.8 dan 0.10 untuk daging ayam, 244.5 dan 0.64 untuk telur ayam, 296.9 dan 1.05 untuk bawang merah serta 602.8 dan 1.32 untuk bawang putih.	Dalam penelitian ini melakukan prediksi harga telur ayam ras dengan menggunakan algoritma LSTM dan GRU, kemudian dievaluasi menggunakan mse.

Berdasarkan Tabel 2.1, peneliti mendapatkan bahwa pemakaian deep learning untuk prediksi sudah banyak dilakukan baik menggunakan algoritma LSTM maupun GRU. Penelitian mengenai prediksi di Tabel 2.1 menjadi literatur dalam pembuatan ataupun pengujian pada penelitian ini.

2.2 Landasan Teori

Landasan teori memuat tentang pengetahuan atau informasi yang berkaitan dengan penelitian yang dilakukan untuk membantu peneliti dalam melakukan penelitian. Berikut ini merupakan landasan teori yang berkaitan dengan penelitian ini.

2.2.1 Telur Ayam Ras

Telur ayam ras adalah sumber protein dari hewan yang mudah diperoleh dan memiliki harga terjangkau, memiliki peran penting dalam pemenuhan kebutuhan pangan, terutama bagi masyarakat dengan tingkat pendapatan rendah [15]. Komoditas ini menjadi favorit di kalangan penduduk Indonesia karena mengandung protein tinggi dan memiliki harga terjangkau dibandingkan dengan produk protein lainnya seperti daging. Meskipun demikian, harga telur ayam ras cenderung fluktuatif dibandingkan dengan komoditas lainnya [19].

2.2.2 Prediksi

Prediksi merupakan suatu proses estimasi mengenai kemungkinan kejadian di masa depan, yang didasarkan pada informasi dari masa lalu dan saat ini. Tujuan prediksi adalah untuk mendapatkan jawaban untuk mendapatkan jawaban yang seakurat mungkin terhadap perkiraan peristiwa yang akan datang [25].

2.2.3 Deep Learning

Deep learning adalah algoritma yang terinpirasi dari struktur dan fungsi otak manusia [24]. Metode ini membantu manusia dalam menjalankan aktivitas sehari-harinya dimana metode ini dapat mengolah data untuk dijadikan sebuah model yang akan digunakan untuk keperluan pengambilan keputusan, memutuskan atau melakukan sesuatu [26]. Deep learning merupakan cabang dari pembelajaran mesin yang khususnya menitikberatkan pada penggunaan jaringan saraf tiruan (JST) atau dapat dianggap sebagai pengembangan dari JST [27]. Deep learning mencakup sejumlah lapisan tersembunyi, yang merupakan komponen dari kecerdasan buatan (*artificial intelligence*) [28].

2.2.4 RNN (*Recurrent Neural Network*)

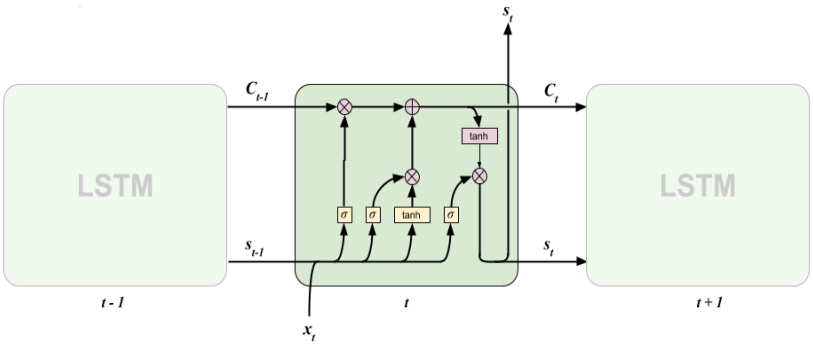
RNN (*Recurrent Neural Network*) merupakan metode pembelajaran mendalam yang digunakan untuk mengolah data secara berurutan dengan pemanggilan berulang [29]. RNN merupakan bentuk arsitektur jaringan saraf (*neural network*) dimana keluaran dari lapisan tersembunyinya menjadi masukan untuk proses lebih lanjut. RNN memiliki karakteristik dalam membuat prediksi data dengan memanfaatkan tidak hanya satu masukan pada satu waktu, namun juga meminta inputan dari inputan sebelumnya untuk membentuk keterkaitan dan memberikan informasi ke lapisan tersembunyi lainnya dalam RNN. RNN juga dilengkapi dengan memori yang menyimpan informasi dari hasil catatan sebelumnya [17].

2.2.5 LSTM (*Long Short Term Memory*)

RNN (*Recurrent Neural Network*) adalah model pembelajaran mesin awal dalam machine learning yang menggunakan jaringan saraf. Ketidakmampuan RNN dalam menjaga keterkaitan informasi dalam jangka panjang, menyebabkan masalah pada hilangnya nilai gradien selama iterasi pada dalam jaringan saraf (*backpropagation neural network*). Oleh karena itu diciptakanlah LSTM (*Long Short Term Memory*) untuk mengatasi permasalahan tersebut [30], dan LSTM ini cocok digunakan untuk kasus prediksi dan klasifikasi [20].

Model LSTM terdiri dari lapisan input, lapisan tersembunyi dan lapisan output. Setiap blok dalam LSTM memiliki sejumlah sel memori yang terpasang dan tiga unit pengontrol yaitu input gate, output gate dan forget gates [20]. *Input gate* mengambil keputusan untuk menentukan apakah informasi masukan tersebut harus disimpan dalam memori sel pada waktu tersebut. *Output gate* memutuskan apa yang akan dikeluarkan tergantung pada blok masukan dan blok memori. *Forget gate* digunakan untuk memutuskan secara kondisional apakah akan membuang informasi dari pemrosesan. *Cell state* bertindak sebagai lapisan penyimpanan. Nilai *cell state* dimanipulasi menggunakan sistem gerbang[17]. LSTM memiliki

tiga gerbang dan satu unit sel memori yang memiliki kemampuan untuk mengabaikan atau menyimpan memori guna menentukan seberapa informasi yang harus disampaikan ke sel berikutnya. Berdasarkan Gambar 2.1 terdapat bentuk persamaan LSTM yang dapat dilihat pada persamaan (2.1), (2.2), (2.3), (2.4), (2.5) dan (2.6) [20].



Gambar 2.1 Arsitektur LSTM [20]

$$f_t = \sigma\left(W_f . \left[h_{t-1,} x_t\right] + b_f\right) \quad (2.1)$$

$$i_t = \sigma\left(W_i . \left[h_{t-1,} x_t\right] + b_i\right) \quad (2.2)$$

$$\tilde{c}_t = \tanh(W_c . [h_{t-1}, x_t] + b_c) \quad (2.3)$$

$$C_t = f_t * c_{t-1} + i_t * \tilde{c}_t \quad (2.4)$$

$$O_t = \sigma\left(W_o . \left[h_{t-1,} x_t\right] + b_o\right) \quad (2.5)$$

$$h_t = O_t * \tanh\left(C_t\right) \quad (2.6)$$

Keterangan :

f_t = *forget gate*

σ = nilai dari fungsi *sigmoid*

W_f = matriks bobot pada *forget gate*

h_{t-1} = nilai output dari sel sebelumnya

x_t = nilai input pada waktu t

b_f = nilai vector bias pada *forget gate*

i_t = input *gate*

W_i = matrik bobot pada *input gate*

b_i = nilai vektor bias pada input *gate*

$\tilde{c}_t$ = vektor kandidat nilai baru

$tanh$ = nilai fungsi *tanh*

W_c = matriks bobot pada *memory cell*

b_c = nilai vektor bias pada *memory cell*

C_t = *Cell state* (*memory cell*)

c_{t-1} = *cell state* pada waktu *t-1*

O_t = output *gate*

W_o = matriks bobot pada output *gate*

b_o = nilai vektor bias pada output *gate*

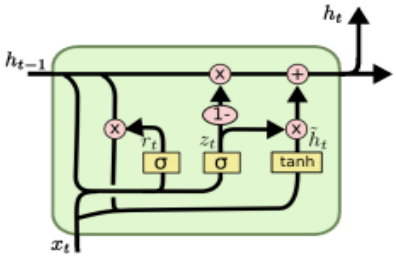
h_t = vektor keluaran hasil kombinasi (hasil akhir)

t = waktu

2.2.6 GRU (*Gated Recurrent Unit*)

Rangkaian LSTM yang kompleks, diperlukan waktu yang lebih lama dalam prosesnya. Sebagai alternatif yang lebih sederhana, diciptakan model yang kurang kompleks yaitu, model GRU (*Gated Recurrent Unit*) [30]. GRU merupakan salah satu algoritma turunan dari LSTM [24].

GRU adalah sel yang memiliki dua gerbang, yang mana jumlah gerbang dan fungsi aktivasinya terbatas [20]. GRU memiliki dua gerbang, yaitu *update gate* dan *reset gate* [21]. Dalam arsitektur GRU, fungsi dari *reset gate* adalah untuk menentukan apakah informasi yang ada harus dilupakan atau diperbarui, sementara *update gate* bertujuan untuk memutuskan apakah informasi tersebut harus diingat atau dipertahankan [31]. GRU dapat mempercepat pemrosesan data yang seringkali besar, sehingga kemampuannya lebih efisien dari pada LSTM, terutama saat menggunakan kumpulan data yang kecil. Berdasarkan proses pada Gambar 2.2 terdapat bentuk persamaan GRU yang dapat dilihat pada persamaan (2.7), (2.8), (2.9), (2.10) [20].



Gambar 2.2 Arsitektur GRU [20]

$$r = \sigma(W_r .\ X_t + U_r . h_{t-1}) + b_r) \quad (2.7)$$

$$z = \sigma(W_z . x_t + \ U_z . h_{t-1} + b_z) \quad (2.8)$$

$$\tilde{h} = tanh(W_h . X_t + r * U_h . h_{t-1} + b_z) \quad (2.9)$$

$$h = z * h_{t-1} + (1 - z) * \ \tilde{h} \quad (2.10)$$

Keterangan :

r = reset *gate*

σ = fungsi aktivasi sigmoid

W_r = matriks bobot terkait dengan input X_t

U_r = matriks bobot terkait *hidden state* sebelumnya

h_{t-1} = *hidden state* sebelumnya

X_t = Input pada waktu *t*

b_r = bias untuk reset *gate*

z = *update gate*

W_z = matriks bobot terkait dengan input X_t

U_z = matriks bobot terkait *hidden state* sebelumnya

b_z = bias untuk *update gate*

$\tilde{h}$ = *hidden state candidate*

$tanh$ = fungsi aktivasi *tanh*

W_h = matriks bobot terkait input X_t hubungan dengan *gate reset r*

U_h = matriks bobot *hidden state* sebelumnya h_{t-1} yang terkait dengan *hidden state candidate*

h = *hidden state* baru pada waktu *t*

t = waktu

2.2.7 Python

Penelitian ini memilih Python sebagai Bahasa pemrograman karena keberadaan beragam *library* yang mendukung operasi kompleks dengan vektor dan matriks (seperti *library keras, Tensorflow*, *NumPy* dan *Pandas*). Python juga menyediakan kemudahan dalam membuat visualisasi data menggunakan berbagai jenis plot yang menarik dan mudah dipahami, termasuk penggunaan *library* seperti *scikit-learn, matplotlib* dan *seaborn*

(*heatmap*) untuk menampilkan korelasi dalam bentuk peta warna dan nilai numeriknya [30].

2.2.8 Pra Proses data

Pra proses data merupakan fase kritis dalam rangkaian pemrosesan data [32]. Pada tahap ini, dilakukan seleksi atribut dalam dataset yang akan digunakan, *formatting data*, menangani data yang hilang dan normalisasi. Proses *formatting data* dilakukan untuk menangani format tipe data yang tidak sesuai yang mungkin terjadi selama proses pengumpulan data dan penyimpanan data. Dataset yang digunakan pada penelitian ini adalah dataset *time series*. Dataset times series merupakan data yang diurutkan menurut indeks waktu tertentu. Oleh karena itu, apabila ada data yang hilang pada waktu tertentu diperlukan penanganan data yang hilang [22]. Proses mengecek dan menangani nilai-nilai yang hilang dalam dataset dilakukan karena data yang hilang dapat berdampak buruk terhadap analisis statistik berdasarkan respon survei, seperti estimasi parameter yang bias dan kesalahan standar yang meningkat. Terdapat beberapa metode yang dapat diterapkan untuk menangani kekurangan data: hapus kolom yang datanya hilang, ganti data yang hilang dengan mean, median data, atau modus [33].

Penanganan nilai data yang hilang (*missing value*) pada penelitian ini dilakukan dengan mengisi nilai data yang hilang menggunakan nilai rata-rata. Nilai rata-rata (*mean*) adalah hasil dari perhitungan nilai rata-rata dari data yang ada. Perhitungan nilai rata-rata harga bulanan dapat dihitung menggunakan persamaan (2.11) [33].

$$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n} \qquad (2.11)$$

Keterangan :

$\bar{x}$ = Nilai rata-rata harga bulanan

x_i = Data harga per bulan

n = Banyaknya data harga per bulan

i = Data ke-1 sampai ke-n

Normalisasi adalah penskalaan data hingga berada dalam rentang tertentu[32]. Normalisasi ini mempengaruhi performa model yang dikembangkan dan stabilitas pelatihan [22]. Normalisasi dilakukan menggunakan *library scikit-learn* dan normalisasi *MinMaxScaler*. Dengan normalisasi *min-max*, ukuran data asli diubah sehingga semua nilai berada dalam rentang 0 hingga 1. Persamaan (2.12) merupakan rumus normalisasi *min-max* [34].

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (2.12)$$

Keterangan :

x_{scaled} = Besaran nilai pada variabel x setelah dinormalisasi

x_{min} = Nilai minimum variabel x

x_{max} = Nilai maximum variabel x

x = Nilai yang akan di normalisasi

2.2.9 Hypertuning Parameter

Hyperparameter merupakan parameter yang termasuk dalam model yang ditentukan sebelum proses pelatihan dilakukan. Kombinasi parameter yang optimal akan menghasilkan model yang sesuai [35]. Parameter yang digunakan dalam proses hypertuning adalah *dropout, batch size, epoch, neuron*, *learning rate* dan *optimizer*. *Dropout* merupakan Teknik yang digunakan untuk mencegah *overfitting* dan mempercepat proses pembelajaran. Proses ini melibatkan penghapusan *neuron* dalam lapisan tersembunyi atau terlihat dari jaringan [36]. *Batch size* merujuk pada jumlah data yang diproses bersamaan dalam satu batch [37]. *Batch size* menentukan jumlah sampel data yang digunakan untuk melakukan iterasi dalam satu proses pelatihan [38]. *Epoch* merupakan iterasi atau satu putaran lengkap saat sebuah algoritma jaringan *neural* melalui seluruh data latih. Jumlah *batch* yang dihasilkan pada setiap *epoch* ditentukan oleh ukuran *batch* atau *batch size* [37]. Penyetelan unit melibatkan penentuan jumlah unit atau *neuron* di lapisan tersembunyi. *Learning rate* adalah kecepatan atau kelambatan dimana model mempelajari informasi dari data yang disediakan.

Penyesuaian *learning rate* dilakukan dengan menetapkan nilai kecepatan pembelajaran pada *optimizer*. *Optimizer* adalah algoritma yang digunakan untuk mengoptimalkan atau meminimalkan fungsi kerugian (*loss function*) saat melatih jaringan *neural*. Algoritma ini memodifikasi bobot dan bias pada setiap iterasi pelatihan, yang menghasilkan peningkatan dalam prediksi model dan meningkatkan akurasi [38].

Pada penelitian ini menggunakan beberapa *optimizer* diantaranya yaitu *Adaptive Moment Estimation* (Adam), RMSprop, *Stochastic Gradient Descent* (SGD) dan Nesterov adam. *Adaptive Moment Estimation* (Adam) merupakan algoritma optimasi gradien orde pertama yang adaptif secara efisien secara komputasi dengan kebutuhan memori yang rendah. Algoritma ini menggabungkan elemen-elemen dari RMSprop dan momentum dalam pengaturan *learning rate* yang adaptif [39]. Penggunaan Adam sebagai optimizer karena gradient proses secara eksponensial [40]. RMSprop adalah fungsi optimisasi yang menggunakan nilai gradien terkini untuk menyesuaikan gradien secara proporsional. Metode ini memelihara rata-rata perhitungan kuadrat gradien dan menjaga nilai rata-rata bergerak di atas *gradien root mean square*, itulah sebabnya disebut RMS. RMSprop mempertahankan nilai rata-rata kuadrat gradien dari setiap bobot dalam proses optimasi [41]. *Stochastic Gradient Descent* (SGD) adalah algoritma optimasi yang umum digunakan dalam mengoptimalkan jaringan syaraf tiruan dan pengoptimal sederhana, namun membutuhkan waktu yang relatif lama untuk mendekati konvergensi. Algoritma ini untuk melakukan pembaruan pada parameter tertentu, seperti bobot (*weight*) dan bias [39]. Penggunaan SGD sebagai optimizer karena tidak melakukan perulangan pada saat update parameter sehingga sehingga lebih cepat untuk dataset yang besar [40]. Nadam atau Nesterov Adam merupakan varian algoritma optimisasi Adam yang telah ditingkatkan dengan menggunakan momentum *Nesterov Accelerated Gradient* (NAG), yang dikenal sebagai nesterov. Nadam beroperasi dengan memperbarui bobot agar dapat meratakan

gradien, sehingga mempercepat proses pelatihan dan meningkatkan tingkat akurasi [42].

2.2.10 Denormalisasi

Denormalisasi berarti mengubah data kembali ke nilai sebenarnya. Hal ini dikarenakan hasil data masih dalam bentuk interval rentang ketika dilakukan proses normalisasi data [35]. Persamaan denormalisasi dapat dilihat pada persamaan (2.13) [43].

$$x_t = y(x_{max} - x_{min}) + x_{min} \tag{2.13}$$

Keterangan :

x_t = nilai data asli

y = hasil output

x_{max} = nilai maksimal data aktual

x_{min} = nilai minimal data aktual

2.2.11 Evaluasi Metrik

Evaluasi performa model prediksi dalam analisis deret waktu menggunakan metrik evaluasi untuk mengukur kesalahan yang dihasilkan oleh model [20]. Dalam penelitian ini, evaluasi dilakukan dengan menggunakan MSE (*Mean Squared Error*), yang merupakan nilai yang diharapkan dari selisih kuadrat antara nilai prediksi dan nilai aktual [23]. Penggunaan MSE bertujuan untuk mengevaluasi seberapa dekat perkiraan nilai dengan data aktual, serta untuk memprediksi nilai data pada masa mendatang. Nilai MSE yang rendah atau hampir mendekati nol menandakan bahwa hasil prakiraan baik atau mendekati nilai sebenarnya. Persaman (2.14) merupakan rumus untuk menghitung nilai MSE [18].

$$MSE = \frac{\sum_{t=1}^{n}(A_t - F_t)^2}{n} \tag{2.14}$$

Keterangan :

A_t = nilai data aktual

F_t = nilai hasil prediksi

n = banyak data

Σ = *Summation* (total jumlah dari seluruh nilai)