

Perbandingan LSTM Dan GRU Untuk Prediksi Harga Saham BBKA

Denny Fernando ^a, Muhammad Siddik ^b

^aInstitut Bisnis dan Teknologi Pelita Indonesia, Jl, Ahmad Yani No.78-88, denny.fernando@student.pelitaindonesia.ac.id

^bInstitut Bisnis dan Teknologi Pelita Indonesia, Jl, Ahmad Yani No.78-88, muhammad.siddik@lecturer.pelitaindonesia.ac.id

INFORMASI ARTIKEL

Sejarah Artikel:

Diterima Redaksi: 19 Januari 2026

Revisi Akhir: 30 Maret 2026

Diterbitkan Online: 30 April 2026

KATA KUNCI

Deep Learning, GRU, LSTM, Prediksi Harga Saham, Time Series

KORESPONDENSI

E-mail:

denny.fernando@student.pelitaindonesia.ac.id

ABSTRACT

Ketidakpastian pergerakan harga saham menuntut adanya model prediksi yang akurat bagi para investor. Penelitian ini bertujuan membandingkan kinerja algoritma *Deep Learning*, yaitu *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU), dalam memprediksi harga saham PT Bank Central Asia Tbk (BBCA). Dataset yang digunakan mencakup 1.219 data historis harga penutupan dari periode 1 Juni 2020 hingga 30 Juni 2025. Proses pengolahan data meliputi pemilihan fitur, pembersihan, normalisasi menggunakan *Min-Max Scaler*, dan pembentukan *sequence*. Data dibagi menjadi 80% data latih, 10% data validasi, dan 10% data uji. Evaluasi kinerja model dilakukan menggunakan metrik MAE, RMSE, dan MAPE serta analisis tingkat konvergensi dan kecepatan komputasi. Hasil penelitian menunjukkan bahwa model GRU lebih unggul dibandingkan LSTM. Konfigurasi optimal pada model GRU (100 units, 100 *epoch*, *time step* 60, *batch size* 32, *optimizer Adam*, dan *dropout* 0,2) menghasilkan nilai RMSE sebesar 175,2050, MAE 140,6608 dan MAPE 1,6109%. Sementara itu, konfigurasi optimal LSTM (100 units, 100 *epoch*, *time step* 60, *batch size* 64, *optimizer Adam*, dan *dropout* 0,2) menghasilkan RMSE sebesar 214,7470, MAE 178,5560 dan MAPE 2,0372%. Meskipun waktu komputasi LSTM lebih cepat (290,66 detik) dibandingkan GRU (397,90 detik), arsitektur GRU terbukti memberikan akurasi dan stabilitas yang lebih baik untuk data deret waktu saham ini.

1. PENDAHULUAN

Investasi jangka panjang merupakan strategi keuangan yang melibatkan penempatan dana untuk periode waktu yang lama, salah satunya melalui pasar modal. Saham menjadi instrumen populer, namun pergerakan harganya memiliki ketidakpastian tinggi yang dipengaruhi oleh kinerja perusahaan, kondisi ekonomi, dan sentimen pasar. Kompleksitas ini menjadikan prediksi harga saham sebagai tantangan besar yang sulit dimodelkan dengan pendekatan tradisional.

Seiring perkembangan teknologi, *Artificial Intelligence* (AI) dan *Deep Learning* muncul sebagai solusi untuk pengolahan data besar dan kompleks. Dalam konteks data deret waktu (*time series*) seperti saham, *Recurrent Neural Network* (RNN) adalah arsitektur yang relevan. Namun, RNN memiliki kelemahan *vanishing gradient*, yang kemudian diatasi oleh variannya yaitu *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU). LSTM memiliki mekanisme *gate* yang kompleks untuk memori jangka panjang, sedangkan GRU menawarkan struktur yang lebih sederhana namun efisien.

Penelitian ini berfokus pada saham PT Bank Central Asia Tbk (BBCA) karena posisinya sebagai saham *blue chip* dengan fundamental kuat. Data harga saham BBCA yang akan digunakan dalam penelitian ini yaitu data selama 5 tahun terakhir dimulai

dari 1 Juni 2020 hingga 30 Juni 2025 yang didapatkan sekitar 1.219 data harga saham BBCA pada periode waktu tersebut.

Data saham BBCA meliputi riwayat tanggal (*date*), harga pembukaan (*opening price*), harga tertinggi (*high*), harga terendah (*low*), harga penutupan (*closing price*), harga penutupan yang telah disesuaikan (*adjusted closing price*) dan volume transaksi selama periode waktu tertentu. Dikarenakan dengan data transaksi selama periode waktu tertentu membuat data saham BBCA termasuk dalam kategori data deret waktu atau historis. Selain sebagai bank besar, data BBCA memiliki data historis yang panjang dan berkualitas, membuatnya ideal untuk pelatihan model *deep learning* yang membutuhkan volume data besar dan konsisten. Untuk data saham BBCA sendiri bisa dikumpulkan melalui sumber terpercaya yaitu dari *Yahoo Finance* yang menyediakan data-data yang diperlukan dalam penelitian.

Penelitian terdahulu menunjukkan hasil yang beragam; beberapa studi mengunggulkan LSTM dalam akurasi, sementara studi lain menemukan GRU lebih efisien dengan akurasi setara.

Tujuan penelitian ini adalah membandingkan kinerja algoritma LSTM dan GRU dalam memprediksi harga saham BBCA berdasarkan aspek *loss function*, metrik evaluasi (MAE, RMSE, MAPE), kecepatan komputasi, dan tingkat konvergensi untuk menentukan model yang paling optimal. Penelitian ini diharapkan dapat memberikan kontribusi berupa rekomendasi model prediksi yang lebih akurat bagi investor maupun peneliti.

2. TINJAUAN PUSTAKA

2.1. Saham

Saham (*stock*) merupakan salah satu instrumen pasar keuangan yang paling populer. Menerbitkan saham merupakan salah satu pilihan perusahaan ketika memutuskan untuk pendanaan perusahaan. Pada sisi yang lain, saham merupakan instrumen investasi yang banyak dipilih para investor karena saham mampu memberikan tingkat keuntungan yang menarik [1].

Harga saham dapat naik turun mengikuti hukum penawaran dan permintaan. Semakin banyak transaksi yang terjadi di mana mudah mencari investor yang menjual atau menawarkan saham, maka saham tersebut disebut likuid. Permintaan dan penawaran akan suatu saham bisa dipengaruhi oleh banyak hal, termasuk kinerja dan prospek bisnis perusahaan, kondisi ekonomi dan politik, sentimen dan masih banyak lagi [2].

2.2. Prediksi Harga Saham

Pergerakan harga saham sulit untuk ditebak akan arah kelajuannya. Dalam memprediksi dapat dilakukan dengan tiga faktor yang mempengaruhi, yaitu faktor teknikal, faktor fundamental, dan faktor sentimen. Faktor teknikal merupakan pergerakan dengan cara mengamati harga pada masa lalu, faktor fundamental merupakan teknik analisis pendekatan secara bisnis yang terjadi, faktor sentimen merupakan pergerakan harga saham yang dipengaruhi oleh faktor-faktor bisnis, berita dan pelaku operasional bisnis [3].

Prediksi harga saham adalah suatu proses menganalisis dan menentukan harga suatu saham di masa yang akan datang. Ada dua pendekatan dasar dalam menganalisis atau memilih saham, yaitu analisis fundamental dan analisis teknikal.

Pemikiran yang mendasari analisis teknikal adalah bahwa pola harga saham mencerminkan informasi yang relevan. Informasi tersebut ditunjukkan oleh perubahan harga saham di waktu yang lalu, dan karenanya perubahan harga saham mempunyai pola tertentu, dan pola tersebut berulang. Analisis teknikal digunakan untuk mencari pola harga saham yang berulang dan dapat diidentifikasi. Salah satu teori analisis terkenal menyatakan bahwa pergerakan harga dari suatu sekuritas mengandung semua informasi mengenai semua sekuritas tersebut [4].

Analisis fundamental adalah pendekatan yang menilai kondisi internal dan eksternal perusahaan dengan memperhatikan faktor-faktor yang dapat memengaruhi nilai saham dalam jangka panjang. Hal tersebut mencakup sejumlah elemen penting, seperti kondisi ekonomi global dan domestik, tren industri yang sedang berkembang, tingkat persaingan di pasar, serta kebijakan fiskal dan moneter yang diterapkan oleh pemerintah. Namun, elemen paling utama dalam analisis fundamental adalah menilai kinerja keuangan perusahaan yang bersangkutan. Kinerja ini dapat diukur melalui laporan keuangan yang mencakup berbagai aspek, seperti pendapatan, laba, utang, dan arus kas [5].

2.3. Deep Learning

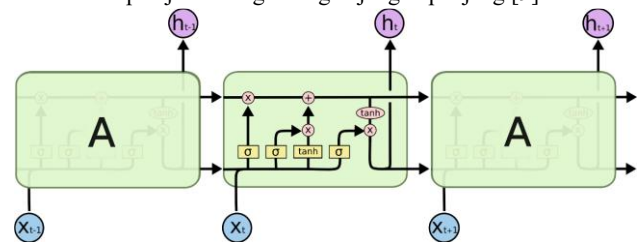
Deep learning merupakan percabangan dari bidang *machine learning* yang menggunakan saraf tiruan atau disebut dengan *artificial neural networks* yang memiliki beberapa *layers* dalam memproses atau mempelajari suatu data. *Deep learning*

memungkinkan suatu komputer untuk belajar secara otomatis melalui pengalaman yang diberikan agar sistem tersebut dapat mengidentifikasi pola yang kompleks dalam data [6].

Deep learning dapat digunakan dalam berbagai aplikasi, termasuk mengidentifikasi objek dan fitur dalam gambar, seperti manusia, hewan, dan tempat, membantu memahami makna teks, seperti dalam chatbot layanan pelanggan dan filter *spam*, membantu menganalisis data keuangan dan membuat prediksi tentang tren pasar, mengubah teks menjadi gambar, seperti pada generator seni AI atau alat pembuatan konten khusus [7].

2.4. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) adalah versi yang ditingkatkan dari *Recurrent Neural Network* (RNN) yang dirancang oleh Hochreiter dan Schmidhuber [8]. LSTM mampu menangkap ketergantungan jangka panjang dalam data berurutan, sehingga sangat ideal untuk tugas seperti penerjemahan bahasa, pengenalan suara, dan peramalan deret waktu (*time series forecasting*). Tidak seperti RNN tradisional yang hanya menggunakan satu *hidden state* (keadaan tersembunyi) yang diteruskan dari waktu ke waktu, LSTM memperkenalkan sel memori (*memory cell*) yang menyimpan informasi dalam jangka waktu yang lebih lama, sehingga mampu mengatasi tantangan dalam mempelajari ketergantungan jangka panjang [9].



Gambar 1. Arsitektur Long Short-Term Memory (LSTM)
(Sumber: Olah, 2025)

Kelebihan LSTM bila dibandingkan dengan RNN konvensional ialah LSTM mampu memodelkan urutan kronologis dan dependensi jarak jauh serta cenderung lebih baik untuk masalah memori pendek karena adanya modifikasi formula pada memori internal. Sedangkan kekurangan dari LSTM, yaitu terjadi peningkatan kompleksitas komputasi dibandingkan dengan RNN karena lebih banyak parameter untuk dipelajari, memori yang dibutuhkan lebih tinggi daripada RNN konvensional karena adanya beberapa memori *cell* serta cenderung terjadi masalah *overfitting* [10].

LSTM memiliki kemampuan untuk menambah atau menghapus informasi dari *cell state*, yang diatur dengan hati-hati oleh struktur yang disebut *gate* (gerbang). Gate adalah mekanisme yang memungkinkan informasi masuk atau keluar secara selektif. LSTM memiliki tiga gerbang yaitu *input gate*, *forget gate* dan *output gate* untuk melindungi dan mengontrol *cell state*. Langkah pertama dalam proses kerja LSTM adalah memutuskan informasi apa yang harus dibuang dari *cell state*. Keputusan ini dibuat oleh lapisan *sigmoid* yang disebut “*forget gate layer*” (lapisan gerbang lupa). Lapisan ini menerima dua input: h_{t-1} (output tersembunyi dari langkah sebelumnya) dan x_t (input saat ini). Lalu menghasilkan output berupa angka antara 0 dan 1 untuk setiap angka dalam keadaan sel C_{t-1} dimana jika nilai 1 berarti “sepenuhnya pertahankan informasi ini” sedangkan nilai 0 berarti “sepenuhnya buang informasi ini”. Berikut adalah rumus yang digunakan:

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

Langkah selanjutnya dalam proses kerja LSTM adalah memutuskan informasi baru apa yang akan disimpan dalam *cell state*. Proses ini terdiri dari dua bagian utama yaitu *Input Gate Layer* (Lapisan Gerbang Masuk), sebuah lapisan *sigmoid* yang bertugas menentukan nilai mana saja yang akan diperbarui dalam *cell state*. Nilai yang dihasilkan berada antara 0 dan 1, menunjukkan seberapa besar masing-masing elemen baru diizinkan masuk. Berikut adalah rumus yang digunakan:

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

Selanjutnya *Tanh Layer* (Lapisan Tanh) dimana lapisan ini menghasilkan vektor kandidat nilai baru yang dilambangkan sebagai $\tilde{C}_t$ vektor ini berisi informasi baru yang berpotensi untuk ditambahkan ke *cell state*. Fungsi aktivasi tanh memastikan nilainya berada antara -1 dan 1, membuat pembaruan data lebih stabil. Pada langkah berikutnya akan digabungkan lapisan *sigmoid* (*Gate Input*) dan lapisan tanh untuk membuat pembaruan ke *state*. Berikut adalah rumus yang digunakan:

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

Selanjutnya memperbarui *cell state* lama C_{t-1} ke *cell state* baru C_t . Kemudian untuk menghapus informasi lama perlu kalikan *cell state* lama C_{t-1} dengan *forget gate* output f_t . Ini akan menghapus bagian-bagian dari informasi lama yang diputuskan untuk dilupakan. Lalu menambahkan informasi baru dengan kalikan *candidate values* $\tilde{C}_t$ (hasil dari tanh layer) dengan *input gate* output i_t yang menentukan berapa banyak dari informasi baru yang akan dimasukkan. Hasilnya kemudian ditambahkan ke *cell state* yang sudah "dibersihkan". Berikut adalah rumus yang digunakan:

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

Langkah terakhir yaitu menentukan apa yang akan dikeluarkan (*output*) dari LSTM pada langkah waktu tersebut. *Output* ini akan berdasarkan *cell state*, tetapi dalam bentuk yang sudah difilter. Dengan jalankan input saat ini dan *hidden state* sebelumnya ke dalam lapisan *sigmoid*, yang akan memutuskan bagian mana dari *cell state* yang akan dikeluarkan sebagai *output*. Hasilnya adalah vektor o_t (*output gate*). Berikut adalah rumus yang digunakan:

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

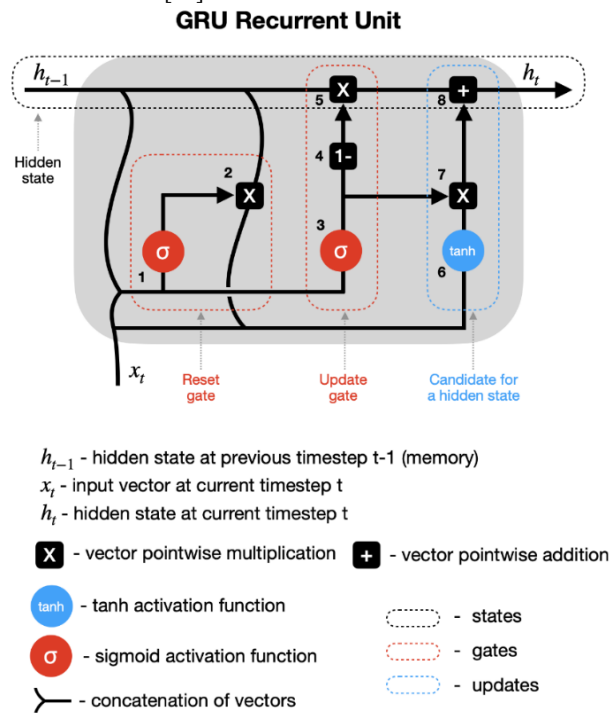
Kemudian cell state saat ini C_t diproses dengan fungsi aktivasi tanh, yang membatasi nilainya antara -1 dan 1 . Lalu Hasil dari tanh tersebut dikalikan dengan o_t agar hanya bagian tertentu dari cell state yang dikeluarkan sebagai output h_t . Berikut adalah rumus yang digunakan:

$$h_t = o_t * \tanh(C_t) \quad (2.6)$$

2.5. Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) adalah jenis dari *Recurrent Neural Network (RNN)* yang diperkenalkan oleh Kyunghyun Cho dan rekannya pada tahun 2014 [11]. Ide utama di balik GRU adalah penggunaan mekanisme gerbang (*gating mechanisms*) untuk secara selektif memperbarui *hidden state* pada setiap langkah waktu, sehingga jaringan dapat mengingat informasi penting dan membuang detail yang tidak relevan. GRU bertujuan untuk menyederhanakan arsitektur LSTM dengan cara menggabungkan beberapa komponennya dan hanya fokus pada dua gerbang utama yaitu: *Update Gate* (Gerbang Pembaruan), gerbang ini menentukan seberapa banyak informasi dari *hidden state* sebelumnya yang harus dipertahankan untuk langkah waktu

berikutnya. Kemudian *Reset Gate* (Gerbang Pengaturan Ulang) Gerbang ini menentukan seberapa banyak dari *hidden state* sebelumnya yang harus dilupakan saat menghitung kandidat *hidden state* baru [12].



Gambar 2. Arsitektur *Gated Recurrent Unit* (GRU) beserta
keterangannya
(Sumber: Dobilas, 2022)

Berikut adalah cara kerja algoritma Gated Recurrent Unit (GRU) [13]: Langkah pertama dalam mekanisme GRU adalah menghitung *Reset Gate* dimana *Hidden state* sebelumnya h_{t-1} dan *input* saat ini x_t digabung (dikalikan dengan bobot masing-masing dan ditambahkan bias), lalu dilewatkan ke dalam *reset gate*. Dikarenakan fungsi aktivasi *sigmoid* menghasilkan nilai antara 0 dan 1 sehingga jika nilai 0 maka informasi dibuang, jika nilai 1 maka informasi dipertahankan, jika nilai di antara 0 dan 1 maka informasi dipertahankan sebagian. Berikut adalah rumus yang digunakan:

$$r_t = \sigma(W_r * [h_{t-1}, x_t]) \quad (2.7)$$

Pada langkah ke-2, *hidden state* sebelumnya dikalikan dengan *output* dari langkah ke-1 (hasil *reset gate*).

Selanjutnya mencari *Update Gate*, dimana langkah ke-3 mirip dengan langkah ke-1, namun menggunakan bobot dan bias yang berbeda, sehingga menghasilkan *output sigmoid* yang berbeda pula. Berikut adalah rumus yang digunakan:

$$z_t = \sigma(W_z * [h_{t-1}, x_t]) \quad (2.8)$$

Setelah vektor gabungan dilewatkan ke fungsi *sigmoid* maka pada langkah ke-4 dimana hasilnya dikurangkan dari vektor yang semua elemennya 1, kemudian langkah ke-5 yaitu hasilnya dikalikan dengan hidden state sebelumnya. Ini menjadi bagian dari proses pembaruan hidden state dengan informasi baru.

Selanjutnya mencari kandidat *Hidden State*, setelah *hidden state* sebelumnya di-*reset* (langkah ke-2), hasilnya digabungkan dengan *input* baru x_t , lalu pada langkah ke-6, hasil digabung ini dikalikan bobot dan ditambah bias, lalu dilewatkan ke fungsi aktivasi tanh, menghasilkan kandidat *hidden state*. Berikut adalah rumus yang digunakan:

$$\tilde{h}_t = \tanh(W_h * [r_t * h_{t-1}, x_t]) \quad (2.9)$$

Kemudian pada langkah ke-7 kandidat ini dikalikan dengan hasil *update gate*. Lalu langkah ke-8 merupakan hasil dari

langkah 7 ditambahkan ke hasil dari langkah 5 sehingga inilah hidden state baru h_t . Berikut adalah rumus yang digunakan:

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

Proses ini akan berulang untuk timestep $t+1$, $t+2$ dan seterusnya hingga seluruh urutan data selesai diproses oleh unit rekuren GRU.

2.6. Python

Python merupakan bahasa pemrograman komputer yang biasa dipakai untuk membangun situs, *software/aplikasi*, mengotomatiskan tugas dan melakukan analisis data. Bahasa pemrograman ini termasuk bahasa tujuan umum. Artinya, bisa digunakan untuk membuat berbagai program berbeda, bukan khusus untuk masalah tertentu saja. Karena sifatnya yang serba guna dan mudah digunakan, ia menjadi bahasa pemrograman yang paling banyak digunakan [14].

Python dapat digunakan untuk membangun dan mengembangkan aplikasi berbasis *website*, *mobile*, maupun *desktop*. Python juga dapat digunakan untuk mengolah hal-hal yang berkaitan dengan data (*data science*) kemudian menganalisis data yang dihasilkan agar menjadi informasi yang lebih bermanfaat. Selain itu, Python menjadi salah satu opsi tepat untuk membuat sistem *machine learning* karena dinilai sederhana, mudah dimengerti, fleksibel, efisien, dan multi-implementasi [15].

Perintah-perintah dasar yang paling sering digunakan dalam bahasa pemrograman Python seperti *print()* untuk menampilkan data, *input()* untuk mengambil *inputan* data dari *user*, *type()* untuk mengecek tipe data variabel, *len()* untuk menghitung jumlah elemen atau panjang dari sebuah objek, *def* untuk membuat sebuah fungsi dan perintah-perintah lainnya.

2.7. Google Collaboratory

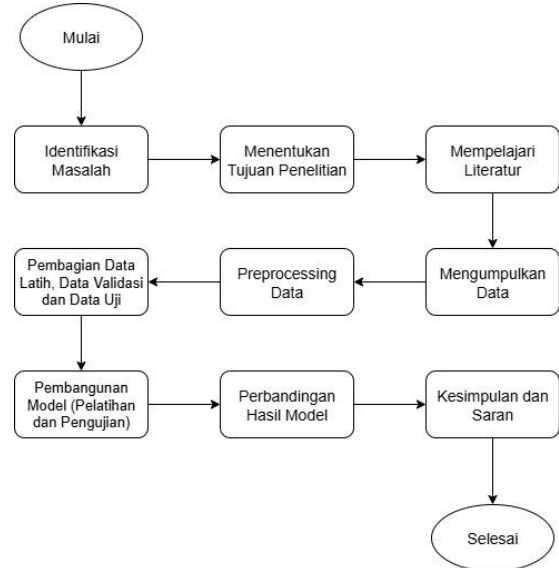
Google Collaboratory atau Google Colab adalah platform berbasis cloud untuk menulis, menjalankan, dan berbagi kode Python melalui *web browser*. Platform ini dirancang bagi *analyst*, *developer*, peneliti, dan pendidik yang bekerja di bidang *data science* dan *machine learning* dengan menyediakan *environment* komputasi yang fleksibel dan mudah diakses tanpa biaya.

Google Colab juga menawarkan kemampuan untuk menjalankan Jupyter Notebook (*web app open-source* untuk kombinasi kode, teks terformat, dan visualisasi data) langsung dari web browser tanpa perlu konfigurasi apa pun. Salah satu fitur utama dari Google Colab adalah kemampuannya untuk berkolaborasi secara *real-time*. Mirip dengan berbagi dokumen di Google Docs, user dapat berbagi *notebook* mereka dengan orang lain. Selain itu, karena *notebook* disimpan di Google Drive, file ini diakses dari perangkat mana pun dan kapan pun, mirip dengan dokumen lain di ekosistem Google. Dalam konteks analisis data, Google Colab memiliki akses ke *library* Python yang sudah terinstal seperti *TensorFlow*, *PyTorch*, dan *OpenCV*, memungkinkan *analyst* mengembangkan dan menjalankan model *machine learning* lebih cepat dan efisien [15].

3. METODOLOGI

3.1. Kerangka Penelitian

Dalam penelitian ini, diperlukan tahap-tahap penelitian yang merupakan proses dari penyelesaian penelitian ini. Berikut adalah kerangka penelitian yang dapat dilihat pada Gambar 3 dibawah ini:



Gambar 3. Kerangka Kerja Penelitian

3.1.1. Identifikasi Masalah

Pada tahap ini akan mengidentifikasi permasalahan yang ada dimana prediksi harga saham merupakan salah satu tantangan terbesar dalam dunia keuangan karena pergerakannya yang kompleks, tidak menentu, dan dipengaruhi oleh berbagai faktor, mulai dari kinerja perusahaan, kondisi ekonomi, hingga sentimen pasar sehingga pendekatan tradisional seringkali tidak memadai dalam memprediksi.

Seiring berkembangnya teknologi dengan adanya kemunculan kecerdasan buatan (*Artificial Intelligence*) yang kemudian bisa menghasilkan algoritma deep learning dalam memprediksi data deret waktu sehingga bisa menjadi solusi untuk permasalahan tersebut. Khususnya, algoritma *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU).

Meskipun keduanya memiliki tujuan yang sama, terdapat perbedaan arsitektur yang mendasar, di mana GRU memiliki struktur yang lebih sederhana daripada LSTM. Perbedaan ini memunculkan pertanyaan mengenai efisiensi komputasi dan akurasi yang dihasilkan oleh masing-masing algoritma. Oleh karena itu, penelitian ini berfokus pada analisis perbandingan kinerja kedua algoritma tersebut untuk memprediksi harga saham PT. Bank Central Asia Tbk., salah satu saham *blue chip* yang menjadi acuan penting bagi investor.

3.1.2. Menentukan Tujuan Penelitian

Berdasarkan identifikasi permasalahan maka didapatkan tujuan dari penelitian yaitu untuk mengetahui kinerja algoritma *Long Short-Term Memory* (LSTM) dan algoritma *Gated Recurrent Unit* (GRU) dalam memprediksi harga saham PT. Bank Central Asia Tbk. (BBCA). Serta untuk mengidentifikasi dan membandingkan model algoritma LSTM dan GRU yang menunjukkan performa kinerja yang lebih optimal dalam memprediksi harga saham PT. Bank Central Asia Tbk. (BBCA)

berdasarkan nilai *loss function*, metrik evaluasi, kecepatan komputasi, tingkat konvergensi dan visualisasi hasil prediksi dalam bentuk grafik.

3.1.3. Mempelajari Literatur

Tahap ini meliputi kajian teori dan penelitian terdahulu yang relevan dengan topik dengan pengumpulan dan analisis literatur dari berbagai sumber terpercaya seperti jurnal ilmiah, buku, dan laporan penelitian. Kajian literatur berfungsi sebagai landasan teori dan memperkuat metodologi yang digunakan dalam penelitian.

3.1.4. Mengumpulkan Data

Pengumpulan data merupakan tahap yang sangat penting dalam penelitian ini karena kualitas dan kelengkapan data sangat memengaruhi akurasi hasil prediksi model. Pada tahap ini akan mengumpulkan data historis harga saham PT. Bank Central Asia Tbk. (BBCA) dari sumber terpercaya yaitu dari *Yahoo Finance* dalam periode yang sudah ditentukan yaitu 5 tahun terakhir dari 1 Juni 2020 hingga 30 Juni 2025 dengan total 1.219 data harga saham BBCA.

3.1.5. Preprocessing Data

Setelah data sudah dikumpul akan masuk ke tahap preprocessing data, pada tahap ini akan dilakukan beberapa metode preprocessing data agar siap digunakan oleh model untuk memastikan kualitas data dan keberhasilan model. Tahap ini akan meliputi pembersihan data yang dimana penanganan data hilang jika ada seperti menghilangkannya, kemudian pemilihan fitur yang akan digunakan yaitu harga penutupan (*closing price*), normalisasi data menggunakan *Min-Max Scaler* dengan tujuan untuk meminimalisir redundansi dan mencegah anomali pada data yang akan digunakan kemudian akan dilanjutkan ke proses pembentukan urutan data (*Sequence/Time Step Shaping*) untuk mengubah data deret waktu (*time series*) menjadi format yang bisa dipahami dan dipelajari oleh model LSTM atau GRU.

3.1.6. Pembagian Data Latih, Data Validasi dan Data Uji

Setelah data selesai melalui tahap preprocessing, langkah selanjutnya adalah melakukan pembagian data menjadi tiga bagian, yaitu data latih (*training data*), data validasi (*validation data*), dan data uji (*testing data*). Pembagian ini bertujuan agar model dapat dilatih, dievaluasi, dan diuji secara objektif menggunakan data yang berbeda, sehingga bisa memberikan bahan belajar untuk model, mengurangi risiko *overfitting* dan meningkatkan kemampuan generalisasi model.

3.1.7. Pembangunan Model (Pelatihan dan Pengujian)

Tahap ini akan membangun arsitektur model algoritma deep learning yaitu *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) sesuai hiperparameter yang sudah ditentukan. Setelah tahap pembangunan pelatihan selesai akan diuji kedua model algoritma menggunakan data uji (*testing data*). Tahap ini adalah proses di mana model melakukan prediksi harga saham untuk data yang belum pernah dilihatnya.

3.1.8. Perbandingan Hasil Model

Setelah model selesai dilatih, akan dilakukan perbandingan hasil model kinerja antara model LSTM dan GRU berdasarkan

nilai *loss function*, metrik evaluasi, kecepatan komputasi, tingkat konvergensi dan visualisasi hasil prediksi dalam bentuk grafik. Tujuan dari tahap ini adalah untuk mengetahui model mana yang mampu memberikan hasil prediksi yang lebih optimal dan akurat dalam memprediksi harga saham PT. Bank Central Asia Tbk. (BBCA).

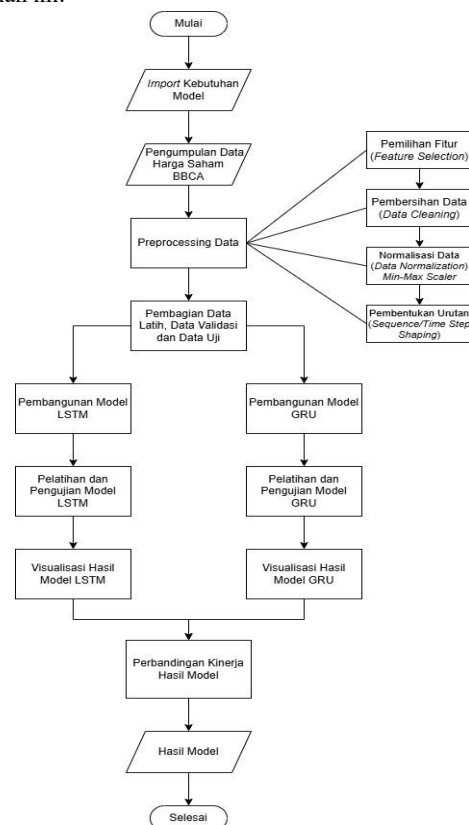
3.1.9. Kesimpulan dan Saran

Pada tahap ini akan dilakukan visualisasi hasil yang menampilkan hasil prediksi model dibandingkan dengan data aktual menggunakan grafik agar bisa melihat seberapa dekat hasil prediksi dengan kenyataan secara visual.

Selanjutnya akan dilakukan tahap pelaporan hasil, pada tahap ini akan dilakukan penyusunan laporan yang mencakup latar belakang, metodologi penelitian, hasil analisis yang diperoleh dengan pembahasan mendalam mengenai perbandingan kinerja kedua algoritma dan kesimpulan akhir berdasarkan penelitian yang dilakukan ini.

3.2. Metode Penelitian

Dalam penelitian ini, metode penelitian yang digunakan adalah algoritma *deep learning* dengan membandingkan dua algoritma, yaitu *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) yang dimana akan digunakan untuk memprediksi harga saham BBCA. Berikut adalah alur dari metode penelitian yang akan digunakan sebagai pedoman penelitian ini:



Gambar 4. Skema Alur Penelitian Algoritma LSTM dan GRU

3.2.1. Import Kebutuhan Model

Pada tahap ini akan diawali dengan mengimpor *library* dan *tools* yang dibutuhkan untuk pembuatan model agar proses pembangunan dan pelatihan model *deep learning* dapat berjalan

serta bisa membantu dalam proses menganalisis hasil model *deep learning* yang dibangun.

3.2.2. Pengumpulan Data Harga Saham BBKA

Selanjutnya akan dilakukan tahap pengumpulan data harga saham BBKA selama 5 tahun terakhir dari 1 Juni 2020 – 30 Juni 2025 yang bersumber dari *Yahoo Finance*, sehingga didapatkan sebanyak 1.129 data harga saham BBKA.

3.2.3. Preprocessing Data

Selanjutnya akan dilakukan proses preprocessing data harga saham BBKA yang didapatkan dari *Yahoo Finance* agar bisa digunakan dalam pembangunan model LSTM dan GRU. Proses preprocessing data berupa pemilihan fitur (*feature selection*), pembersihan data (*data cleaning*), normalisasi data (*data normalization*) menggunakan *Min-Max Scaler* dan pembentukan urutan (*sequence/time step shaping*).

3.2.4. Pembagian Data Latih, Data Validasi dan Data Uji

Pada penelitian ini akan menggunakan data latih yang merupakan sumber pengetahuan utama bagi model sebanyak 80%, data validasi untuk memantau performa model selama proses pelatihan sebanyak 10%, serta data uji untuk mengukur kemampuan generalisasi model terhadap data baru yang belum pernah dilihat sebelumnya setelah proses pelatihan model selesai sebanyak 10% sehingga didapatkan rasio 80:10:10.

3.2.5. Pembangunan Model LSTM dan GRU

Arsitektur perancangan arsitektur jaringan model *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) pada penelitian ini menggunakan hiperparameter seperti *optimizer adam*, *time step* sebanyak 60, *dropout* sebesar 0,2 dan *Loss Function* menggunakan *Mean Squared Error* (MSE). Kemudian untuk hiperparameter lainnya akan divariasikan seperti jumlah *neuron* (units) akan digunakan sebesar 50 dan 100, *epoch* sebanyak 50 dan 100, dan *batch size* sebesar 32 dan 64 dengan tujuan untuk mengevaluasi dan membandingkan performa prediksi masing-masing algoritma yang mana bisa memberikan hasil yang lebih optimal.

3.2.6. Pelatihan dan Pengujian Model LSTM dan GRU

Tahap pelatihan dan pengujian model LSTM dan GRU bertujuan agar model dapat mempelajari pola pergerakan harga saham berdasarkan data historis, sehingga mampu menghasilkan prediksi harga saham pada periode berikutnya secara akurat.

Setelah model selesai dilatih dan menunjukkan hasil pelatihan yang stabil maka akan dilakukan pengujian model, tahap ini bertujuan untuk mengukur sejauh mana model mampu melakukan generalisasi terhadap data baru yang belum pernah dilihat sebelumnya sehingga hasil dari modelnya bisa dibandingkan.

3.2.7. Visualisasi Hasil Model LSTM dan GRU

Pada tahap ini akan menampilkan hasil prediksi yang diperoleh dari model *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) dalam bentuk grafik agar dapat dianalisis secara visual. Visualisasi hasil model bertujuan untuk menunjukkan perbandingan antara nilai aktual (*real*) dengan nilai prediksi (*predicted*) dari model LSTM dan GRU serta mengamati

pola pergerakan harga saham apakah sudah berhasil diikuti oleh model yang dibangun dan dilatihkan.

3.2.8. Perbandingan Kinerja Hasil Model

Pada tahap ini akan dilakukan perbandingan kinerja hasil model LSTM dan GRU berdasarkan nilai *loss function*, metrik evaluasi, kecepatan komputasi, tingkat konvergensi dan visualisasi hasil prediksi dalam bentuk grafik yang bertujuan mengidentifikasi model mana yang paling optimal dan paling cocok untuk memprediksi harga saham BBKA.

Model LSTM dan GRU tersebut juga akan dibandingkan dengan proses hyperparameter tuning atau mencoba beberapa kombinasi nilai hiperparameter dengan kategori *Grid Search* sebagai berikut:

1. *Neuron* (Units): 50 dan 100
2. *Epoch*: 50 dan 100
3. *Time Step*: 60
4. *Batch Size*: 32 dan 64
5. *Optimizer*: Adam
6. *Dropout*: 0,2

3.2.9. Hasil Model

Tahap ini merupakan hasil akhir dari proses pelatihan dan pengujian model *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) menggunakan kombinasi hiperparameter yang telah ditentukan sebelumnya dengan menghasilkan model yang paling optimal. Tahap ini bertujuan untuk mengidentifikasi dan menyajikan konfigurasi model yang paling optimal untuk prediksi harga saham BBKA.

3.3. Metode Perbandingan Algoritma LSTM & GRU

Aspek yang digunakan dalam perbandingan meliputi *loss function*, metrik evaluasi, kecepatan komputasi, tingkat konvergensi, dan visualisasi hasil prediksi. Kelima aspek ini dipilih karena mampu memberikan gambaran menyeluruh terhadap akurasi, efisiensi, dan stabilitas model dalam konteks prediksi data deret waktu (*time series*). Berikut adalah penjelasan dari setiap aspek yang akan dibandingkan:

3.3.1. Loss Function

Loss function digunakan untuk mengukur tingkat kesalahan (*error*) antara nilai prediksi dan nilai aktual selama proses pelatihan model. Dalam penelitian ini digunakan *Mean Squared Error* (MSE) sebagai fungsi loss, yang dihitung dengan cara merata-ratakan kuadrat selisih antara nilai aktual dan nilai prediksi. Semakin kecil nilai MSE, semakin baik kemampuan model dalam menyesuaikan bobot (*weights*) terhadap pola data.

3.3.2. Metrik Evaluasi (Evaluation Metrics)

Metrik Evaluasi (*Evaluation Metrics*) digunakan untuk menilai sejauh mana model dapat melakukan prediksi secara akurat terhadap data yang diuji dengan memberikan angka spesifik yang mencerminkan seberapa baik atau buruk kinerja model. Dalam penelitian ini, metrik evaluasi yang digunakan yaitu *Mean Absolute Error* (MAE), *Root Mean Squared Error* (RMSE), *Mean Absolute Percentage Error* (MAPE). Semakin kecil nilai MAE, RMSE dan MAPE maka semakin baik dan akurat model tersebut.

3.3.3. Kecepatan Komputasi

Aspek kecepatan komputasi digunakan untuk mengukur efisiensi waktu yang dibutuhkan model dalam proses pelatihan (*training*) dan pengujian (*testing*). Semakin cepat model mencapai hasil yang optimal tanpa mengorbankan akurasi, semakin baik performa komputasinya.

3.3.4. Tingkat Konvergensi

Tingkat konvergensi merupakan ukuran seberapa cepat model pembelajaran mendekati kondisi optimal atau titik stabil, yaitu saat nilai *loss function* tidak mengalami penurunan signifikan lagi seiring bertambahnya *epoch*. Semakin cepat model mencapai konvergensi, semakin efisien proses pembelajarannya.

3.3.5. Visualisasi Hasil Prediksi

Visualisasi hasil prediksi memberikan gambaran secara visual mengenai seberapa baik hasil prediksi model dibandingkan dengan data aktual. Visualisasi menyajikan perbandingan antara harga aktual dengan harga prediksi pada data uji dalam bentuk grafik garis. Model yang paling optimal harus memiliki garis prediksi yang paling rapat dan paling konsisten mengikuti garis harga aktual.

4. HASIL DAN PEMBAHASAN

4.1. Implementasi Model

Implementasi model merupakan tahap pembangunan dan penerapan model *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) sebelum dilakukan pengujian hiperparameter. Tujuan dari implementasi ini adalah mempersiapkan model untuk menerima data latih yang telah diproses. Pada penelitian ini akan menggunakan Bahasa Pemrograman Python yang dibangun di platform Google Colab untuk melakukan seluruh implementasi model.

4.1.1. Import Kebutuhan Model

Tahap awal yang dilakukan yaitu menginisialisasi lingkungan kerja dengan mengimpor semua *library* Python yang diperlukan untuk implementasi *deep learning*, preprocessing data, dan visualisasi hasil.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error
from math import sqrt
import time
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, GRU, Dense, Dropout
from tensorflow.keras.optimizers import Adam
import warnings
```

Gambar 5. Library Python yang digunakan

Pada Gambar 5 dapat dilihat akan digunakan *library* Python berupa *numpy*, *pandas*, *yfinance*, *sklearn*, *tensorflow* dan lain sebagainya untuk mendukung proses implementasi model yang akan digunakan. Kemudian juga akan dilakukan pembuatan fungsi pembantu untuk mendukung proses penelitian seperti pada Gambar 6 berikut:

```
warnings.filterwarnings('ignore')

np.random.seed(42)
tf.random.set_seed(42)

def calculate_mape(y_true, y_pred):
    """Menghitung Mean Absolute Percentage Error (MAPE)"""
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    non_zero_mask = y_true != 0
    if np.any(non_zero_mask):
        return np.mean(np.abs((y_true[non_zero_mask] - y_pred[non_zero_mask]) /
                               y_true[non_zero_mask]) * 100)
    return np.nan

def calculate_all_metrics(y_true, y_pred):
    """Menghitung semua metrik evaluasi (MSE, RMSE, MAE, MAPE)"""
    mse = mean_squared_error(y_true, y_pred)
    rmse = sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    mape = calculate_mape(y_true, y_pred)
    return mse, rmse, mae, mape
```

Gambar 6. Pembuatan Fungsi Model

Pada Gambar 6 dapat dilihat bahwa adanya kodingan untuk mengabaikan *warnings* (peringatan) yang mungkin muncul saat menjalankan kode, sehingga *output* menjadi lebih bersih dan mengatur *seed* ke angka yang sama untuk mempertahankan kondisi awal yang identik dan mencegah variasi hasil yang disebabkan oleh proses inisialisasi acak model. Kemudian juga ada fungsi untuk menghitung metrik evaluasi yang dibutuhkan supaya proses penelitian menjadi lebih lancar dengan menggunakan fungsi tersebut.

4.1.2. Pengumpulan Data Harga Saham BBKA

Tahap ini akan dilakukan pengumpulan data harga historis harga saham PT. Bank Central Asia Tbk. (BBKA.JK). Data harga saham BBKA merupakan data berjenis *time series* yang diperoleh secara otomatis dari platform *Yahoo Finance* menggunakan *library* Python *yfinance*. Data dikumpulkan merupakan data selama 5 tahun terakhir dimulai dari 1 Juni 2020 hingga 30 Juni 2025. Kemudian akan dilakukan eksplorasi data secara menyeluruh terhadap *DataFrame* (tabel data) yang sebelumnya telah diunduh dari *Yahoo Finance* dengan tujuan untuk memahami struktur, kualitas, dan karakteristik dari data harga saham yang baru diambil, sebelum data tersebut diolah lebih lanjut. Hasil eksplorasi data bisa dilihat pada Gambar 7 berikut:

```
[*****] 1 of 1 completed
```

```
INFORMASI LENGKAP DATA YANG DIDOWNLOAD
```

```
Data berhasil diambil: 1219 baris
```

```
STRUKTUR DATA:
```

```
Shape: (1219, 5)
```

```
Kolom: ['Close', 'BBCA.JK'], ('High', 'BBCA.JK'), ('Low', 'BBCA.JK'), ('Open', 'BBCA.JK'), ('Volume', 'BBCA.JK')
```

```
Index type: <class 'pandas.core.indexes.datetimes.DatetimeIndex'>
```

```
Data type:
```

```
Price Ticker
```

```
Close BBCA.JK float64
```

```
High BBCA.JK float64
```

```
Low BBCA.JK float64
```

```
Open BBCA.JK float64
```

```
Volume BBCA.JK int64
```

```
dtype: object
```

```
5 DATA BARIS PERTAMA:
```

```
Price Ticker Close High Low Open Volume
```

```
Date BBCA.JK BBCA.JK BBCA.JK BBCA.JK BBCA.JK
```

```
2020-06-02 4713.167969 4836.273102 4687.649283 4625.235731 181369500
```

```
2020-06-03 5082.482910 5082.482910 4748.340435 4757.133658 279673500
```

```
2020-06-04 5091.276855 5267.141341 4924.205594 5082.483631 236158500
```

```
2020-06-05 5094.121552 5146.433513 4911.016425 5091.277546 80157500
```

```
2020-06-08 5188.001953 5209.585812 5060.508210 5060.508210 173557500
```

```
5 DATA BARIS TERAKHIR:
```

```
Price Ticker Close High Low Open Volume
```

```
Date BBCA.JK BBCA.JK BBCA.JK BBCA.JK BBCA.JK
```

```
2025-06-20 8642.865234 8816.715972 8568.357775 8642.865234 445915100
```

```
2025-06-23 8560.358398 8642.865863 8444.179291 8518.680756 87324500
```

```
2025-06-24 8717.373047 8791.880509 8642.865585 8667.701406 110158900
```

```
2025-06-25 8543.522461 8717.373209 8419.343355 8667.701566 117561000
```

```
2025-06-26 8593.194336 8593.194336 8469.015227 8469.015227 117281700
```

```
STATISTIK DESKRIPTIF:
```

```
Price Ticker Close High Low Open Volume
```

```
count BBCA.JK BBCA.JK BBCA.JK BBCA.JK BBCA.JK
```

```
1219.000000 1219.000000 1219.000000 1219.000000 1219.000000
```

```
mean 7564.253759 7632.185085 7497.074247 7564.905983 8.818107e+07
```

```
std 1496.033894 1504.153312 1490.933740 1495.162622 5.528508e+07
```

```
min 4713.167969 4836.273102 4687.649283 4625.235731 1.065390e+07
```

```
25% 6123.188721 6205.618395 6032.794510 6143.832699 5.790575e+07
```

```
50% 7774.618040 7858.265359 7698.821309 7767.154278 7.537160e+07
```

```
75% 8717.373047 8793.738632 8645.162356 8698.893586 1.012060e+08
```

```
max 10500.997070 10500.997070 10333.171875 10453.046259 7.564316e+08
```

```
INFORMASI DATA HILANG:
```

```
Price Ticker
```

```
Close BBCA.JK 0
```

```
High BBCA.JK 0
```

```
Low BBCA.JK 0
```

```
Open BBCA.JK 0
```

```
Volume BBCA.JK 0
```

```
dtype: int64
```

```
INFORMASI PERIODE DATA:
```

```
Periode data: 2020-06-02 00:00:00 hingga 2025-06-26 00:00:00
```

```
Jumlah hari perdagangan: 1219
```

```
Hari dalam seminggu: 1219
```

```
Friday 238
```

```
Monday 245
```

```
Thursday 241
```

```
Tuesday 251
```

```
Wednesday 244
```

```
Name: count, dtype: int64
```

Gambar 7. Hasil Eksplorasi Data

Pada Gambar 7 dapat dilihat dari hasil eksplorasi data bahwa data yang didapatkan selama periode 5 tahun dimulai dari 1 Juni 2020 hingga 30 Juni 2025 didapatkan sebanyak 1.219 data. Kemudian terdapat juga bentuk kolom data, tipe data setiap kolom, 5 data baris pertama, 5 data baris terakhir, statistik deskriptif data, informasi data hilang dan informasi periode data yang didapatkan.

4.1.3. Preprocessing Data

Preprocessing data merupakan tahap mengubah data mentah harga saham BBKA yang tidak lengkap, tidak konsisten dan memiliki format yang beragam menjadi format yang bersih, terstruktur dan dapat dipelajari oleh algoritma *machine learning*. Pada tahap awal preprocessing data akan dilakukan pemilihan fitur (*feature selection*) dan pembersihan data (*data cleaning*).

Pemilihan fitur (*feature selection*) adalah memilih satu kolom data yang paling relevan untuk digunakan analisis yaitu kolom harga penutupan (*close price*). Ini akan mencegah model bingung oleh variabel yang tidak relevan dan bisa mengurangi beban komputasi. Sedangkan pembersihan data (*data cleaning*) adalah proses menghilangkan nilai yang hilang (*missing values*) dengan dilakukan pembersihan data ini bisa membuat pembelajaran model lebih akurat. Untuk pemilihan fitur harga penutupan dan pembersihan data bisa dilihat pada Gambar 8 dibawah ini:

```
# Pilih fitur Close Price dan bersihkan data
df = data[['Close']].copy()
df = df.dropna()
print(f"Data setelah cleaning: {len(df)} data")
```

Data setelah cleaning: 1219 data

Gambar 8. Pemilihan Fitur dan Pembersihan Data

Dari hasil pembersihan data dapat dilihat bahwa data yang memiliki nilai yang hilang (*missing values*) itu tidak ada sehingga jumlah data yang didapatkan juga sama dengan data awal yang diunduh yaitu 1.219 data. Selanjutnya akan dilakukan normalisasi data dengan *Min-Max Scaler* dengan *source code* berikut:

```
# Normalisasi data dengan MinMaxScaler (0-1)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(df)
print(f"Contoh data yang sudah dinormalisasi:")
print(scaled_data)
```

Contoh data yang sudah dinormalisasi:

```
[[0.        ]
 [0.06380905]
 [0.06532827]
 ...
 [0.69183197]
 [0.66179471]
 [0.67037683]]
```

Gambar 9. Source Code dan Hasil Data dinormalisasi

Pada tahap terakhir preprocessing data penelitian ini adalah akan dilakukan pembentukan urutan (*sequence/time step shaping*). Pembentukan urutan diperlukan untuk mengubah data deret waktu 1-dimensi menjadi struktur data 3-dimensi agar bisa diproses model *Recurrent Neural Network* (RNN) seperti LSTM atau GRU. Pembentukan urutan (*sequence/time step shaping*) dapat dilakukan dengan *source code* sebagai berikut:

```
def create_sequences(data, sequence_length):
    X, y = [], []
    for i in range(sequence_length, len(data)):
        X.append(data[i-sequence_length:i, 0])
        y.append(data[i, 0])
    return np.array(X), np.array(y)
```

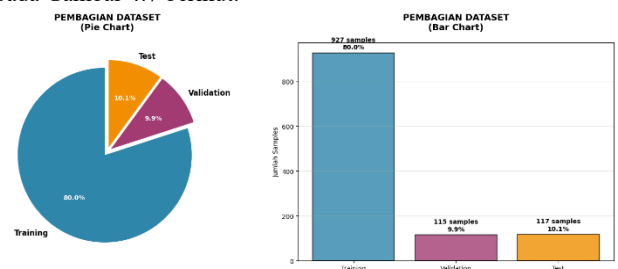
SEQUENCE_LENGTH = 60

```
X, y = create_sequences(scaled_data, SEQUENCE_LENGTH)
```

Gambar 10. Source Code Pembentukan Urutan

4.1.4. Pembagian Dataset

Setelah dilakukan preprocessing data akan dilanjutkan dengan pembagian dataset. Pembagian dataset dilakukan untuk menghindari model yang dilatih menjadi *overfitting* sehingga bisa menilai performa model secara objektif. Pembagian dataset pada penelitian ini dibagi menjadi 3 bagian yaitu 80% data latih, 10% data validasi, dan 10% data uji. Untuk visualisasi pembagian dataset dengan rasio 80:10:10 yang sudah dilakukan bisa dilihat pada Gambar 4.7 berikut:



Gambar 11. Visualisasi Pembagian Dataset

Selanjutnya akan dilakukan reshaping data. Reshaping data adalah proses mengubah bentuk (*shape*) data agar sesuai dengan kebutuhan model LSTM dan GRU yaitu menjadi 3 dimensi yang strukturnya adalah [*Samples, Time Steps, Features*]. *Reshaping*

data bisa dilakukan dengan menggunakan *source code* sebagai berikut:

```
# Reshape untuk LSTM/GRU [samples, time steps, features]
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_val = X_val.reshape((X_val.shape[0], X_val.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

print(f"RESHAPE PEMBAGIAN DATASET:")
print(f"X_train: {X_train.shape}")
print(f"X_val: {X_val.shape}")
print(f"X_test: {X_test.shape}")
```

```
RESHAPE PEMBAGIAN DATASET:
X_train: (927, 60, 1)
X_val: (115, 60, 1)
X_test: (117, 60, 1)
```

Gambar 12. Source Code dan Hasil Reshaping Data

4.1.5. Pembangunan Model

Tahap pembangunan model akan berfokus pada perancangan arsitektur model *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU). Tahap pembangunan model merupakan proses inti dalam penelitian karena menentukan arsitektur jaringan dibentuk, dan hiperparameter yang akan digunakan dalam penelitian. Pembangunan model *Long Short-Term Memory* (LSTM) dapat dilihat dengan pembuatan fungsi sebagai berikut:

```
def build_lstm_model(sequence_length, units=50, dropout_rate=0.2, learning_rate=0.001):
    model = Sequential([
        LSTM(units=units, return_sequences=True, input_shape=(sequence_length, 1)),
        Dropout(dropout_rate),
        LSTM(units=units, return_sequences=False),
        Dropout(dropout_rate),
        Dense(units=25, activation='relu'),
        Dense(units=1)
    ])
    optimizer = Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])
    return model
```

Gambar 13. Fungsi Model LSTM

Setelah fungsi pembangunan model untuk model *Long Short-Term Memory* (LSTM) dibuat, selanjutnya akan dilakukan juga pembuatan fungsi untuk model *Gated Recurrent Unit* (GRU) sebagai berikut:

```
def build_gru_model(sequence_length, units=50, dropout_rate=0.2, learning_rate=0.001):
    model = Sequential([
        GRU(units=units, return_sequences=True, input_shape=(sequence_length, 1)),
        Dropout(dropout_rate),
        GRU(units=units, return_sequences=False),
        Dropout(dropout_rate),
        Dense(units=25, activation='relu'),
        Dense(units=1)
    ])
    optimizer = Adam(learning_rate=learning_rate)
    model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])
    return model
```

Gambar 14. Fungsi Model GRU

4.1.6. Pelatihan Model

Setelah arsitektur model *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) selesai dibangun, akan dilakukan Optimasi Hiperparameter (*Hyperparameter Tuning*). Proses ini akan mengombinasikan nilai hiperparameter dengan kategori *grid search*. Hal ini bertujuan untuk menemukan konfigurasi parameter yang paling optimal bagi setiap algoritma dalam memprediksi harga saham BBKA. Berikut adalah kombinasi hiperparameter yang diuji dengan kategori *grid search*:

Tabel 1 : Kombinasi Hiperparameter yang diuji

Konfigurasi Hiperparameter	Neuron (Units)	Epoch	Time Step	Batch Size	Optimizer	Dropout
1	50	50	60	32	Adam	0,2
2	50	50	60	64	Adam	0,2
3	50	100	60	32	Adam	0,2

4	50	100	60	64	Adam	0,2
5	100	50	60	32	Adam	0,2
6	100	50	60	64	Adam	0,2
7	100	100	60	32	Adam	0,2
8	100	100	60	64	Adam	0,2

Setelah proses implementasi dan optimasi hiperparameter selesai akan dilanjutkan pelatihan model. Pelatihan model dilakukan pada data latih, sementara kinerja akhir dievaluasi secara objektif pada data uji yang belum pernah dilihat oleh model. Berikut adalah *source code* untuk pelatihan model *Long Short-Term Memory* (LSTM):

```
lstm_model = build_lstm_model(SEQUENCE_LENGTH, UNITS, DROPOUT_RATE, LEARNING_RATE)

start_time_lstm = time.time()
lstm_history = lstm_model.fit(
    X_train, y_train,
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    validation_data=(X_val, y_val),
    verbose=1,
    shuffle=False
)
training_time_lstm = time.time() - start_time_lstm
```

Gambar 15. Source Code Pelatihan Model LSTM

Setelah model *Long Short-Term Memory* (LSTM) sudah dilatih akan dilanjutkan pelatihan untuk model *Gated Recurrent Unit* (GRU) dengan *source code* sebagai berikut:

```
gru_model = build_gru_model(SEQUENCE_LENGTH, UNITS, DROPOUT_RATE, LEARNING_RATE)

start_time_gru = time.time()
gru_history = gru_model.fit(
    X_train, y_train,
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    validation_data=(X_val, y_val),
    verbose=1,
    shuffle=False
)
training_time_gru = time.time() - start_time_gru
```

Gambar 16. Source Code Pelatihan Model GRU

Model *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) yang telah berhasil dilatih dengan konfigurasi hiperparameter akan dilanjutkan ke tahap pengujian dan evaluasi model untuk mengukur tingkat generalisasi model pada data uji yang belum pernah dilihat sehingga dapat dilakukan perbandingan kinerja akhir antara kedua algoritma tersebut.

4.2. Pengujian Model

Pada tahap ini akan dilakukan proses pengujian terhadap model *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) yang telah dibangun pada tahap sebelumnya. Pengujian model dilakukan pada data uji (*testing data*) yang sebelumnya tidak digunakan pada proses pelatihan (*training*). Proses pengujian ini bertujuan untuk mengevaluasi kemampuan prediksi kedua model. Proses prediksi akan dilakukan dengan *source code* berikut ini:

```
# Prediksi
lstm_train_pred = lstm_model.predict(X_train, verbose=0)
lstm_val_pred = lstm_model.predict(X_val, verbose=0)
lstm_test_pred = lstm_model.predict(X_test, verbose=0)

gru_train_pred = gru_model.predict(X_train, verbose=0)
gru_val_pred = gru_model.predict(X_val, verbose=0)
gru_test_pred = gru_model.predict(X_test, verbose=0)
```

Gambar 17. Source Code Prediksi Data

Setelah dilakukan prediksi data seperti pada Gambar 17, akan dilanjutkan dengan mengembalikan nilai prediksi datanya yang dari angka normalisasi ke skala asli atau bentuk awal data sebenarnya sehingga bisa dilakukan analisis pada tahap

selanjutnya dan bisa tetap relevan. Berikut adalah *source code* untuk mengembalikan nilai prediksi yang sudah dinormalisasi sebelumnya ke bentuk skala asli atau data sebenarnya:

```
# Inverse transform ke skala asli
def inverse_transform_predictions(predictions, actual_y, scaler):
    pred_2d = predictions.reshape(-1, 1)
    actual_2d = actual_y.reshape(-1, 1)
    return scaler.inverse_transform(pred_2d), scaler.inverse_transform(actual_2d)

# Inverse transform semua prediksi
lstm_train_actual, y_train_actual = inverse_transform_predictions(lstm_train_pred, y_train, scaler)
lstm_val_actual, y_val_actual = inverse_transform_predictions(lstm_val_pred, y_val, scaler)
lstm_test_actual, y_test_actual = inverse_transform_predictions(lstm_test_pred, y_test, scaler)

gru_train_actual, _ = inverse_transform_predictions(gru_train_pred, y_train, scaler)
gru_val_actual, _ = inverse_transform_predictions(gru_val_pred, y_val, scaler)
gru_test_actual, _ = inverse_transform_predictions(gru_test_pred, y_test, scaler)
```

Gambar 18. Source Code Inverse Transform Data

Pada tahap terakhir pengujian akan dilakukan penyimpanan semua hasil evaluasi model LSTM dan GRU agar bisa dilakukan analisis mendalam pada penelitian ini. Penyimpanan hasil evaluasi ini terdiri dari perhitungan metrik evaluasi, waktu komputasi, nilai *loss function* terakhir dan nilai *loss minimum*. Berikut adalah *source code* untuk penyimpanan hasil evaluasi model:

```
results = {}

for model_name, actual_pred in [('LSTM', (lstm_train_actual, lstm_val_actual, lstm_test_actual)),
                                ('GRU', (gru_train_actual, gru_val_actual, gru_test_actual))]:

    train_pred, val_pred, test_pred = actual_pred

    train_metrics = calculate_all_metrics(y_train_actual, train_pred)
    val_metrics = calculate_all_metrics(y_val_actual, val_pred)
    test_metrics = calculate_all_metrics(y_test_actual, test_pred)

    results[model_name] = {
        'train_metrics': train_metrics,
        'val_metrics': val_metrics,
        'test_metrics': test_metrics,
        'training_time': training_time_lstm if model_name == 'LSTM' else training_time_gru,
        'final_train_loss': lstm_history.history['loss'][-1] if model_name == 'LSTM' else gru_history.history['loss'][-1],
        'final_val_loss': lstm_history.history['val_loss'][-1] if model_name == 'LSTM' else gru_history.history['val_loss'][-1],
        'min_train_loss': min(lstm_history.history['loss']) if model_name == 'LSTM' else min(gru_history.history['loss']),
        'min_val_loss': min(lstm_history.history['val_loss']) if model_name == 'LSTM' else min(gru_history.history['val_loss'])
    }
```

Gambar 19. Source Code Penyimpanan Hasil Evaluasi Model

4.3. Perbandingan Akhir Model

Perbandingan akhir antara model *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU) bertujuan untuk menentukan model yang memiliki performa paling optimal dalam memprediksi harga saham PT. Bank Central Asia Tbk. (BBCA). Perbandingan ini dilakukan dengan mempertimbangkan beberapa aspek utama, yaitu nilai metrik evaluasi sebagai ukuran akurasi prediksi, kualitas pelatihan yang berasal dari nilai *loss function* dan tingkat konvergensi, serta efisiensi waktu komputasi yang dibutuhkan oleh masing-masing model.

4.3.1. Perbandingan Model Berdasarkan Nilai Metrik Evaluasi

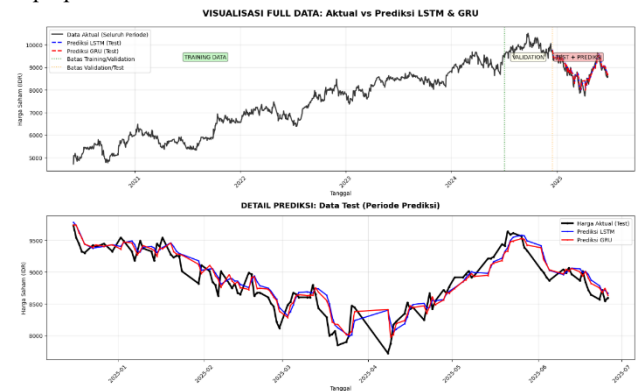
Perbandingan nilai metrik evaluasi dilakukan untuk mengukur tingkat akurasi dan kesalahan prediksi yang dihasilkan oleh model LSTM dan GRU berdasarkan konfigurasi hiperparameter yang sudah dilakukan. Berikut adalah hasil akhir model optimal dari masing-masing konfigurasi hiperparameter:

Tabel 2 : Perbandingan Model Optimal Nilai Metrik Evaluasi

Konfigurasi Hiperparameter	Algoritma	RMSE	MAE	MAPE
1	GRU	227.4599	185.6341	2.0884%
2	GRU	212.5119	175.2371	1.9832%
3	LSTM	215.5740	177.0311	2.0009%
4	GRU	222.5089	179.3415	2.0378%
5	LSTM	208.9044	173.7966	1.9772%

6	LSTM	258.0244	214.1619	2.4358%
7	GRU	175.2050	140.6608	1.6109%
8	GRU	183.6768	145.5977	1.6505%

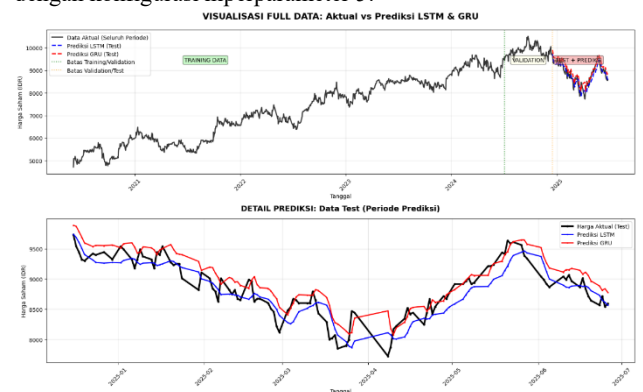
Berdasarkan Tabel 2 dapat dianalisis bahwa model yang paling optimal pada penelitian ini yaitu model GRU dengan konfigurasi hiperparameter 7. Nilai RMSE yaitu 175.2050 dan nilai MAE yaitu 140.6608 mengindikasikan bahwa kesalahan prediksi yang dihasilkan model relatif rendah, sedangkan nilai MAPE yaitu 1.6109% yang paling kecil dibanding konfigurasi lainnya menunjukkan tingkat kesalahan prediksi dalam persentase yang paling minimal. Berikut adalah visualisasi grafik harga aktual dan prediksi model dengan konfigurasi hiperparameter 7:



Gambar 20. Grafik Harga Aktual dan Prediksi Model Konfigurasi Hiperparameter 7

Pada Gambar 20, kedua model (LSTM dan GRU) menunjukkan kinerja yang sangat kompetitif dan seimbang pada konfigurasi hiperparameter 7. Hal ini bisa dilihat bahwa garis biru (LSTM) dan garis merah (GRU) sama-sama berhimpitan dan hampir selaras dengan garis hitam (aktual). Untuk menentukan model yang unggul bisa dilihat dari nilai metrik evaluasi yang didapatkan agar lebih akurat.

Sedangkan model LSTM mendapatkan hasil optimal pada konfigurasi hiperparameter 5 dengan nilai RMSE yaitu 208.9044, nilai MAE yaitu 173.7966 dan nilai MAPE sebesar 1.9772%. Berikut adalah visualisasi grafik harga aktual dan prediksi model dengan konfigurasi hiperparameter 5:



Gambar 21. Grafik Harga Aktual dan Prediksi Model Konfigurasi Hiperparameter 5

Pada detail prediksi Gambar 21 menunjukkan model LSTM (garis biru) justru terlihat jauh lebih akurat dan stabil dibandingkan model GRU (garis merah). Model GRU pada konfigurasi ini memprediksi harga yang jauh lebih tinggi daripada harga aktual, sementara model LSTM menunjukkan performa yang lebih mendekati harga aktual dapat dilihat saat harga jatuh di bulan April 2025, garis biru (LSTM) mampu turun

lebih dalam mendekati harga aktual dibandingkan garis merah (GRU). Hal ini didukung nilai metrik evaluasi model LSTM pada konfigurasi ini lebih baik dibanding model GRU.

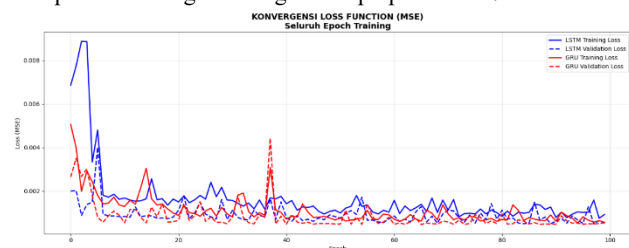
4.3.2. Perbandingan Model Berdasarkan Kualitas Pelatihan

Kualitas pelatihan mencerminkan kemampuan model dalam mempelajari pola data secara optimal selama proses pelatihan, yang dapat diamati melalui nilai *loss function*, stabilitas proses pembelajaran, serta tingkat konvergensi model. Analisis kualitas pelatihan bertujuan untuk mengevaluasi efisiensi model dalam pembelajaran data, dan menemukan gejala *overfitting* model yang dapat memengaruhi kemampuan generalisasi pada data pengujian. Berikut adalah hasil akhir model dengan tingkat konvergensi optimal pada masing-masing konfigurasi hiperparameter:

Tabel 3. Perbandingan Model Optimal Kualitas Pelatihan

Konfigurasi Hiperparameter	Algoritma	Min Train Loss	Final Train Loss	Min Val Loss	Final Val Loss
1	GRU	0.000942	0.001739	0.000580	0.003290
2	GRU	0.001344	0.001730	0.000574	0.001931
3	LSTM	0.000781	0.000900	0.000744	0.003720
4	GRU	0.000911	0.001422	0.000557	0.000675
5	GRU	0.000846	0.000992	0.000493	0.000744
6	LSTM	0.001185	0.001881	0.000780	0.000875
7	GRU	0.000525	0.000562	0.000461	0.000543
8	LSTM	0.000814	0.000854	0.000600	0.000622

Berdasarkan Tabel 3 dapat dianalisis bahwa kualitas pelatihan dengan konfigurasi yang paling optimal yaitu konfigurasi hiperparameter 7. Dengan algoritma GRU menggunakan konfigurasi hiperparameter *Neuron* (units) sebesar 100, *Epoch* sebanyak 100, *Time Step* sebanyak 60, *Batch Size* sebanyak 32, *Optimizer Adam*, dan *Dropout* sebesar 0,2 terbukti menunjukkan proses pelatihan yang stabil dan kemampuan generalisasi yang baik. Nilai *minimal train loss* sebesar 0.000525 dan *minimal validation loss* sebesar 0.000461 yang rendah menandakan model tersebut cepat dan konsisten mencapai titik optimal. Berikut adalah visualisasi grafik tingkat konvergensi saat pelatihan dengan konfigurasi hiperparameter 7:

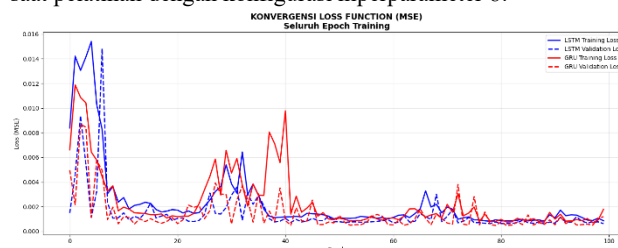


Gambar 22. Grafik Tingkat Konvergensi Model Konfigurasi Hiperparameter 7

Berdasarkan Gambar 22 model LSTM mendapatkan *minimum training loss* sekitar di *epoch* ke 99 dan *minimum validation loss* sekitar di *epoch* ke 98 sedangkan untuk model GRU mendapatkan *minimum training loss* sekitar di *epoch* ke 95

dan *minimum validation loss* sekitar di *epoch* ke 66. Pada konfigurasi ini, model GRU lebih menguasai data latih sedangkan model LSTM lebih presisi memprediksi data uji. Kedua model pada konfigurasi ini bisa dibilang aman untuk digunakan.

Sedangkan model LSTM mendapatkan konfigurasi hiperparameter yang paling optimal pada konfigurasi hiperparameter 8 dengan *Neuron* (units) sebesar 100, *Epoch* sebanyak 100, *Time Step* sebanyak 60, *Batch Size* sebanyak 64, *Optimizer Adam*, dan *Dropout* sebesar 0,2. Pada konfigurasi hiperparameter tersebut, model LSTM mendapat nilai *minimal train loss* sebesar 0.000814 dan *minimal validation loss* sebesar 0.000600. Berikut adalah visualisasi grafik tingkat konvergensi saat pelatihan dengan konfigurasi hiperparameter 8:



Gambar 23. Grafik Tingkat Konvergensi Model Konfigurasi Hiperparameter 8

Berdasarkan Gambar 23 didapatkan bahwa model LSTM menunjukkan performa yang sangat stabil dibanding model GRU sedangkan model GRU mengalami *overfitting* di akhir pelatihan. Hal ini bisa disebabkan gradien yang meledak (*Exploding Gradients*) yang terjadi pada model GRU.

Pada kualitas pelatihan juga menunjukkan model dengan algoritma GRU lebih unggul dibanding LSTM. Hal ini disebabkan algoritma GRU memiliki struktur lebih sederhana dengan jumlah parameter yang lebih sedikit sehingga proses pembelajaran menjadi lebih efisien dan stabil selama pelatihan model.

4.3.3. Perbandingan Kecepatan Komputasi

Perbandingan kecepatan komputasi dilakukan untuk mengevaluasi efisiensi waktu yang dibutuhkan model LSTM dan GRU dalam menyelesaikan proses pelatihan. Aspek kecepatan komputasi menjadi penting karena berkaitan dengan sumber daya komputasi yang digunakan serta kelayakan penerapan model dalam kondisi nyata. Berikut adalah hasil kecepatan komputasi model LSTM dan GRU sesuai konfigurasi hiperparameter yang ditentukan:

Tabel 4. Hasil Kecepatan Komputasi Model

Konfigurasi Hiperparameter	Waktu Komputasi	
	LSTM	GRU
1	97.87 detik	112.04 detik
2	64.04 detik	74.03 detik
3	200.70 detik	230.18 detik
4	131.30 detik	143.89 detik
5	188.82 detik	201.96 detik
6	167.45 detik	200.13 detik
7	363.12 detik	397.90 detik
8	290.66 detik	339.60 detik

Berdasarkan Tabel 4 dapat dilihat bahwa semakin besar nilai hiperparameter yang digunakan maka semakin lama juga waktu yang dibutuhkan dalam proses pembangunan model, sehingga pemilihan nilai hiperparameter yang tepat sangat disarankan untuk meningkatkan efisiensi waktu pelatihan model. Selain itu, meskipun algoritma GRU memiliki arsitektur yang

lebih sederhana dibanding LSTM, waktu komputasi GRU pada penelitian ini cenderung lebih lama. Hal tersebut dapat disebabkan oleh faktor implementasi dan perangkat yang digunakan, konfigurasi eksperimen, serta karakteristik konvergensi model selama proses pelatihan.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan penelitian yang telah dilakukan, dapat didapatkan kesimpulan bahwa prediksi harga saham PT. Bank Central Asia Tbk. (BBCA) selama periode 5 tahun dapat dilakukan menggunakan algoritma *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU). Kedua algoritma mampu memodelkan data deret waktu dan menghasilkan prediksi harga saham dengan nilai metrik evaluasi yang berbeda sesuai dengan karakteristik masing-masing algoritma.

Hasil penelitian menunjukkan algoritma *Gated Recurrent Unit* (GRU) memberikan performa yang lebih optimal dibanding algoritma *Long Short-Term Memory* (LSTM). Arsitektur GRU yang lebih sederhana mampu melakukan pembelajaran secara efektif dan memiliki kemampuan generalisasi yang baik terhadap data harga saham BBCA yang digunakan, sehingga menghasilkan prediksi yang lebih akurat dan konsisten.

Nilai konfigurasi hiperparameter mempengaruhi kecepatan komputasi model, dimana semakin besar nilai hiperparameter yang digunakan maka semakin lama juga waktu komputasi yang dibutuhkan. Selain itu, pengaruh faktor eksternal seperti implementasi kode dan spesifikasi perangkat juga mempengaruhi kecepatan komputasi model.

5.2. Saran

Berdasarkan hasil penelitian yang dilakukan, terdapat saran yang dapat digunakan untuk pengembangan penelitian selanjutnya yaitu model optimal yang dihasilkan dari penelitian ini bisa diterapkan ke sebuah aplikasi untuk memprediksi harga saham lainnya. Kemudian penelitian ini bisa dikembangkan dengan menggunakan dataset dengan periode waktu yang lebih lama serta menerapkan metode optimasi hiperparameter lebih lanjut seperti *Random Search* untuk memperoleh konfigurasi model yang lebih optimal.

DAFTAR PUSTAKA

- [1] BEI, "Saham," Bursa Efek Indonesia. [Online]. Available: <https://www.idx.co.id/id/produk/saham/>
- [2] Schroders, "Apa Itu Saham: Cara Beli, Jenis-Jenis Saham serta Risiko Investasi Saham," Schroders. [Online]. Available: <https://www.schroders.com/id-id/investasi-reksadana/resources/edukasi/kenalan-yuk-sama-saham/>
- [3] M. Abdul Dwiyanto Suyudi, E. C. Djamal, and A. Maspupah, "Prediksi Harga Saham menggunakan Metode Recurrent Neural Network," *Semin. Nas. Apl. Teknol. Inf.*, pp. 1907–5022, 2019.
- [4] C. Lubis, E. Sutedjo, and B. Setiadi, "Prediksi Harga Saham Dengan Menggunakan Algoritma Hybrid Neural Network," vol. 2005, no. Snati, pp. 17–19, 2005.
- [5] Team Brandcom BRIDS, "Semakin Paham Kualitas Saham dengan Analisis Fundamental," Brights. [Online]. Available: <https://www.brights.id/id/blog/analisis-fundamental>
- [6] Binus, "Mengenal Deep Learning Beserta Contoh Penerapannya," Binus. [Online]. Available: <https://sis.binus.ac.id/2023/07/18/mengenal-deep-learning-beserta-contoh-penerapannya/>
- [7] S. R. Noviah, "Mengenal Google Colab dan Manfaatnya," exabytes. [Online]. Available: <https://www.exabytes.co.id/blog/google-colab-adalah/>
- [8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, 1997, doi: 10.1162/neco.1997.9.8.1735.
- [9] GeeksForGeeks, "What is LSTM - Long Short Term Memory?," GeeksForGeeks. [Online]. Available: <https://www.geeksforgeeks.org/deep-learning/deep-learning-introduction-to-long-short-term-memory/>
- [10] Trivusi, "Mengenal Algoritma Long Short Term Memory (LSTM)," Trivusi. [Online]. Available: <https://www.trivusi.web.id/2022/07/algoritma-lstm.html>
- [11] K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," *arXiv Prepr.*, 2014, [Online]. Available: <https://arxiv.org/abs/1406.1078>
- [12] GeeksForGeeks, "Gated Recurrent Unit Networks," GeeksForGeeks. [Online]. Available: <https://www.geeksforgeeks.org/machine-learning/gated-recurrent-unit-networks/>
- [13] S. Dobilas, "GRU Recurrent Neural Networks – A Smart Way to Predict Sequences in Python," towards data science. [Online]. Available: <https://towardsdatascience.com/gru-recurrent-neural-networks-a-smart-way-to-predict-sequences-in-python-80864e4fe9f6/>
- [14] D. Intern, "Python: Pengertian, Contoh Penggunaan, dan Manfaat Mempelajarinya," Dicoding Indonesia. [Online]. Available: <https://www.dicoding.com/blog/python-pengertian-contoh-penggunaan-dan-manfaat-mempelajarinya/>
- [15] L. Hanif, "Apa Itu Python? Pengertian, Fungsi, Kelebihan, dan Contohnya," rumahweb. [Online]. Available: <https://www.rumahweb.com/journal/python-adalah/>