BAB II
KAJIAN LITERATUR

2.1 Cryptocurrency

Cryptocurrency merupakan aset digital yang memanfaatkan teknik *cryptography* untuk menjamin keamanan transaksi, mengontrol penciptaan unit baru, serta memverifikasi transfer aset tanpa melibatkan otoritas terpusat [4]. Sistem *cryptocurrency* umumnya dibangun di atas teknologi *blockchain*, yaitu buku besar terdistribusi (*distributed ledger*) yang mencatat seluruh transaksi secara transparan, terdesentralisasi, dan bersifat tidak dapat diubah (*immutable*) [15]. Konsep ini pertama kali diwujudkan melalui Bitcoin yang diluncurkan pada Januari 2009 oleh Satoshi Nakamoto, ketika ia menambang genesis block yang menandai awal era baru sistem keuangan digital [16]. Sejak saat itu, ribuan cryptocurrency bermunculan dengan berbagai fungsi dan mekanisme konsensus, mulai dari *proof-of-work* hingga *proof-of-stake* [15]. Keamanan transaksi pada cryptocurrency bergantung pada algoritma kriptografi yang kuat seperti SHA-256 pada Bitcoin, sehingga transaksi sulit dipalsukan atau diubah setelah tercatat dalam blockchain [15].

Karakteristik utama cryptocurrency terletak pada sifatnya yang *decentralized*, sehingga tidak dikendalikan oleh lembaga keuangan tradisional seperti bank atau otoritas moneter. Selain itu, sistem ini memungkinkan transaksi dilakukan secara langsung antar pengguna (*peer-to-peer*) tanpa perantara, sehingga meningkatkan efisiensi dan transparansi dalam proses transaksi digital [15]. Litecoin (LTC), yang diluncurkan pada tahun 2011 oleh Charlie Lee, sering disebut sebagai "*silver to bitcoin's gold*" karena menawarkan waktu blok yang lebih cepat (2,5 menit) dibandingkan Bitcoin (10 menit) serta menggunakan algoritma *hashing Scrypt* yang berbeda [17]. Peluncuran Ethereum pada tahun 2015 membawa revolusi signifikan melalui kontrak pintar (*smart contract*), yang memungkinkan pengembangan aplikasi terdesentralisasi (dApps) di atas blockchain [18]. Teknologi kontrak pintar ini membuka jalan bagi munculnya ekosistem keuangan terdesentralisasi (DeFi) dan token *non-fungible* (NFT) yang memperluas fungsi blockchain dari sekadar transfer nilai menjadi ekosistem finansial yang lebih kompleks [15].

Seiring perkembangannya, *cryptocurrency* tidak hanya digunakan sebagai alat tukar digital, tetapi juga sebagai instrumen investasi yang memiliki potensi keuntungan tinggi, meskipun disertai risiko yang signifikan akibat volatilitas harga yang tinggi [4]. Volatilitas

harga *cryptocurrency*, terutama Bitcoin, sangat tinggi dibandingkan aset keuangan tradisional lainnya [13]. Tahun 2020–2021 menyaksikan ledakan DeFi yang menawarkan pinjaman, *staking*, dan pertukaran terdesentralisasi, serta NFT yang memperluas fungsi blockchain ke seni digital dan koleksi daring [4]. Perkembangan pesat ini menjadikan pemahaman tentang cryptocurrency sebagai fondasi penting bagi siapa pun yang ingin terjun ke pasar aset digital [15]. Kelebihan utama cryptocurrency adalah biaya transaksi yang relatif rendah, kecepatan transfer lintas negara tanpa perantara bank, serta aksesibilitas bagi siapa saja yang terhubung ke internet [15]. Namun demikian, fluktuasi nilai yang tinggi yang dapat mencapai puluhan persen dalam satu hari menjadikannya instrumen berisiko tinggi bagi investor konservatif [4]. Beberapa negara, seperti El Salvador, telah mengadopsi Bitcoin sebagai alat pembayaran sah, sementara negara lain seperti China melarang penambangan dan transaksi cryptocurrency karena risiko pencucian uang dan ketidakstabilan finansial [13].

2.2 Bitcoin

Bitcoin merupakan *cryptocurrency* pertama yang diperkenalkan oleh Satoshi Nakamoto pada tahun 2008 dan hingga saat ini menjadi aset kripto dengan kapitalisasi pasar terbesar [19], [13]. Bitcoin dirancang sebagai sistem pembayaran elektronik *peer-to-peer* yang memungkinkan transaksi dilakukan secara langsung antar pengguna tanpa perantara pihak ketiga seperti bank atau lembaga keuangan [16]. Setiap transaksi Bitcoin dicatat dalam blockchain yang dapat diakses publik, sehingga semua transfer bersifat transparan namun nama pengguna tetap anonim karena hanya alamat dompet digital yang terlihat [15]. Pasokan Bitcoin dibatasi hanya 21 juta koin melalui mekanisme halving, sehingga sifatnya mirip dengan emas digital keterbatasan ini menjadi salah satu pendorong utama nilainya di pasar [13]. Seiring dengan meningkatnya adopsi, Bitcoin tidak hanya berfungsi sebagai alat tukar, tetapi juga sebagai instrumen investasi dan aset spekulatif di pasar keuangan global [15]. Beberapa perusahaan bahkan menyimpan Bitcoin dalam neraca mereka sebagai aset strategis, dan nilai Bitcoin ditentukan oleh mekanisme permintaan dan penawaran di pasar, sehingga menyebabkan fluktuasi harga yang sangat dinamis [4].

Harga Bitcoin dikenal memiliki tingkat volatilitas yang tinggi dibandingkan aset keuangan tradisional. Volatilitas ini dipengaruhi oleh berbagai faktor, seperti sentimen pasar, berita finansial, kebijakan regulasi, serta kondisi ekonomi global [4]. Sensitivitas terhadap informasi eksternal menjadikan pergerakan harga Bitcoin sulit diprediksi menggunakan metode konvensional berbasis statistik linier seperti ARIMA atau GARCH [20]. Oleh karena

itu, diperlukan pendekatan berbasis machine learning dan deep learning yang mampu menangkap pola non-linear serta dependensi jangka panjang dalam data historis harga Bitcoin [13]. Berita tentang regulasi di negara besar seperti Amerika Serikat atau China sering memicu pergerakan harga tajam dalam hitungan jam [3]. Dengan kata lain, Bitcoin sangat sensitif terhadap sentimen eksternal, sehingga memprediksi harganya membutuhkan lebih dari sekadar data harga historis [4]. Pendekatan deep learning dinilai lebih efektif dalam memodelkan karakteristik data finansial yang kompleks dan fluktuatif dibandingkan metode statistik tradisional [20].

2.3 Prediksi Time Series

Prediksi *time series* merupakan proses peramalan nilai di masa depan berdasarkan pola historis data yang tersusun secara berurutan dalam dimensi waktu [20]. Dalam konteks pasar finansial, data yang digunakan umumnya memiliki karakteristik non-stasioner, non-linear, serta volatilitas yang tinggi, sehingga menimbulkan tantangan dalam proses peramalan [21]. Contoh penggunaan prediksi *time series* meliputi peramalan harga saham, nilai tukar mata uang, suhu udara, permintaan energi, dan lalu lintas jaringan [20]. Setiap titik data memiliki urutan waktu yang berkesinambungan, dan pola yang mungkin muncul dapat berupa tren (kenaikan/penurunan jangka panjang), musiman (pola berulang dalam periode tetap), siklus (fluktuasi tanpa periode tetap yang disebabkan oleh kondisi ekonomi), atau *noise* (komponen acak yang tidak memiliki pola jelas) [20].

Pendekatan statistik konvensional seperti ARIMA (*Autoregressive Integrated Moving Average*) telah banyak digunakan dalam prediksi *time series* selama beberapa dekade, terutama karena kesederhanaannya dan interpretabilitasnya yang baik [20]. Namun, metode tersebut memiliki keterbatasan signifikan dalam menangkap hubungan non-linear dan dependensi temporal jangka panjang yang sering muncul pada data finansial modern [22]. Kelemahan utama ARIMA terletak pada asumsi linearitas dan stasioneritas data, yang jarang terpenuhi dalam data finansial riil yang sarat dengan guncangan eksternal dan perilaku non-linear [20]. Akibatnya, prediksi yang dihasilkan cenderung kurang akurat pada kondisi pasar yang ekstrem seperti krisis keuangan atau gejolak harga tajam [4]. Dalam konteks cryptocurrency, khususnya Bitcoin, prediksi *time series* menghadapi tantangan tambahan akibat sensitivitas harga terhadap faktor eksternal seperti sentimen pasar, berita finansial, peristiwa politik global, serta perubahan kebijakan dan regulasi di berbagai negara [13].

Berdasarkan *horizon* waktunya, prediksi *time series* dapat dikategorikan menjadi tiga jenis utama: jangka pendek (*short-term*, misalnya beberapa jam hingga hari ke depan), jangka menengah (*medium-term*, misalnya minggu hingga bulan), dan jangka panjang (*long-term*, misalnya bulan hingga tahun ke depan) [13]. Perlu dicatat bahwa klasifikasi ini tidak bersifat universal dan bergantung pada granularitas data yang digunakan; Definisi horizon prediksi harus diinterpretasikan secara relatif terhadap frekuensi data, di mana untuk data harian, prediksi beberapa bulan ke depan sudah dikategorikan sebagai long-term forecasting, sementara definisi yang sama dapat berbeda untuk data mingguan atau kuartalan [23]. Dalam domain energi dan sistem tenaga listrik, *short-term forecasting* mencakup rentang dari beberapa menit hingga beberapa hari, sedangkan *long-term forecasting* mencakup rentang dari beberapa bulan hingga tahun ke depan [24]. Tabel 2.1 menyajikan klasifikasi horizon prediksi yang umum digunakan pada data *time series* harian.

Tabel 2.1 Jenis Klasifikasi Horizon

Jenis Forecasting	**Horizon Prediksi (Data Harian)**	**Tujuan Utama**	**Referensi**
Short-term Forecasting	Beberapa jam hingga beberapa hari	Keputusan operasional harian	[24]
Medium-term Forecasting	Beberapa minggu hingga beberapa bulan	Perencanaan taktis jangka menengah	[23], [24]
Long-term Forecasting	Beberapa bulan hingga lebih dari 1 tahun	Perencanaan strategis dan analisis tren	[23], [24]

Dalam penelitian ini, model dilatih menggunakan data historis harian Bitcoin selama kurang lebih sembilan tahun (Januari 2017 hingga Desember 2025) dan dievaluasi pada data uji sebesar 20% dari total dataset, yang mencakup 646 hari atau sekitar 1,8 tahun terakhir. Mengacu pada klasifikasi pada Tabel 2.1, horizon evaluasi yang digunakan dalam penelitian ini mencakup periode lebih dari satu tahun pada data harian, yang termasuk dalam kategori long-term forecasting karena panjang horizon harus disesuaikan dengan frekuensi data yang digunakan [23]. Pemilihan horizon ini didasarkan pada kebutuhan investor untuk perencanaan investasi strategis yang mempertimbangkan tren jangka panjang, bukan hanya fluktuasi harian atau mingguan [5]. Selain itu, prediksi jangka panjang pada pasar cryptocurrency merupakan tantangan yang lebih kompleks dibandingkan aset keuangan tradisional karena harga Bitcoin memiliki karakteristik volatilitas tinggi, non-stasioner, dan sangat sensitif terhadap sentimen pasar serta perubahan kondisi makroekonomi dan regulasi

[13]. Model deep learning seperti LSTM dan GRU dirancang khusus untuk kebutuhan tersebut karena mereka mempertahankan informasi dari langkah waktu sebelumnya dalam memori internal yang dapat diupdate secara adaptif [25]. Lim dan Zohren (2021) dalam survei komprehensif mereka mengenai arsitektur *deep learning* untuk *time series forecasting* menyatakan bahwa LSTM dan GRU secara konsisten mengungguli metode statistik tradisional seperti ARIMA dalam menangkap dependensi temporal jangka panjang, terutama pada data yang bersifat non-stasioner dan non-linear seperti data finansial [26]. Metode *deep learning* seperti LSTM, GRU, dan Transformer telah terbukti lebih unggul dibandingkan ARIMA dalam berbagai studi prediksi finansial, terutama ketika data memiliki pola musiman dan tren yang tidak linear [20]. Dalam penelitian ini, LSTM dan GRU dipilih sebagai model utama karena kemampuannya yang telah terbukti secara objektif dalam memprediksi harga Bitcoin pada berbagai kondisi pasar [13], [27].

2.4 Preprocessing Data untuk Prediksi Time Series

Sebelum data *time series* dapat diproses oleh model *deep learning* seperti LSTM dan GRU, data harus melalui beberapa tahap *preprocessing* untuk memastikan data memiliki kualitas yang baik, terbebas dari *noise* dan inkonsistensi, serta sesuai dengan format input yang diharapkan oleh model [28]. Tanpa *preprocessing* yang tepat, model dapat menghasilkan prediksi yang berat sebelah, tidak stabil, atau bahkan gagal untuk konvergen karena data yang tidak sistematis akan mendominasi proses pembelajaran [20]. Dalam penelitian ini, data harga Bitcoin dan skor sentimen diproses secara berurutan melalui tahapan pembersihan data (*data cleaning*), normalisasi (*normalization*), transformasi *sliding window*, dan pembagian data latih-uji [28]. Seluruh tahapan ini dirancang untuk memastikan bahwa data yang masuk ke model telah siap untuk pembelajaran yang efektif dan evaluasi yang adil [28].

2.4.1 Pembersihan Data

Dalam data *time series* finansial seperti harga Bitcoin, nilai yang hilang (*missing values*) dapat terjadi karena berbagai alasan seperti liburan pasar (market holidays), gangguan teknis API (*Application Programming Interface*), periode tanpa transaksi karena likuiditas rendah, atau kesalahan saat proses scraping data [28]. *Missing values* harus diidentifikasi dan ditangani sebelum proses normalisasi karena dapat mengganggu perhitungan statistik dan mengganggu pola temporal data [20]. Metode umum untuk menangani *missing values* pada data *time series* meliputi interpolasi linear (mengisi nilai

berdasarkan titik sebelum dan sesudah secara proporsional), *forward fill* (mengisi dengan nilai sebelumnya), *backward fill* (mengisi dengan nilai setelahnya), atau imputasi dengan metode statistik seperti *mean imputation* [28]. Penanganan yang tidak tepat dapat menghasilkan pola palsu yang membingungkan model, misalnya interpolasi pada loncatan harga yang tajam dapat menciptakan pola transisi gradual yang sebenarnya tidak terjadi di pasar nyata [20]. Dalam penelitian ini, nilai yang hilang diisi menggunakan metode *forward fill* pada data harga, karena harga cenderung tidak berubah drastis dalam satu hari libur pasar atau akhir pekan [28]. Untuk skor sentimen, *forward fill* digunakan untuk menghindari bias akibat tidak adanya berita pada hari tersebut dengan asumsi bahwa sentimen pasar tidak berubah secara fundamental tanpa adanya informasi baru [28]. Selain itu, data duplikat dan *outlier* yang tidak wajar misalnya harga negatif atau loncatan harga melebihi 200% dalam satu hari tanpa alasan juga dihapus untuk menjaga konsistensi dataset sebelum masuk ke tahap selanjutnya [28].

2.4.2 Normalisasi Data

Normalisasi data merupakan proses penskalaan nilai fitur ke dalam rentang tertentu agar tidak ada satu fitur yang mendominasi proses pelatihan model *deep learning* [20]. Harga Bitcoin dapat mencapai puluhan ribu dolar (hingga $100,000), sementara skor sentimen hanya berkisar antara -1 hingga 1 [4]. Tanpa normalisasi, fitur harga akan mendominasi perhitungan *loss function*, sehingga kontribusi sentimen menjadi tidak berarti karena perbedaan skala yang terlalu besar [20]. Dalam penelitian ini, data harga Bitcoin dinormalisasi ke rentang [0, 1] menggunakan metode MinMaxScaler, sedangkan data sentimen dinormalisasi ke rentang [-1, 1] untuk merepresentasikan polaritas sentimen secara proporsional (negatif murni = -1, netral = 0, positif murni = 1) [20]. Rumus MinMaxScaler adalah sebagai berikut [28]:

$$X' = \frac{(X - X_{min})}{(X_{max} - X_{min})} \quad \text{.....(2.1)}$$

Dimana:

X' : Nilai hasil normalisasi

X : Nilai asli (sebelum normalisasi)

X_{min} : Nilai minimum dari seluruh dataset

X_{max} : Nilai maksimum dari seluruh dataset

Alternatif lain adalah StandardScaler yang menghasilkan data dengan mean 0 dan standar deviasi 1 (distribusi Z-score), namun MinMaxScaler lebih cocok untuk data dengan

batas bawah dan atas yang jelas seperti harga yang tidak pernah negatif [20]. Salah satu keunggulan utama MinMaxScaler adalah kemampuannya mempertahankan hubungan relatif antar nilai (nilai yang lebih besar sebelum normalisasi tetap lebih besar setelah normalisasi) sambil membawa data ke dalam rentang yang seragam [28]. Setelah proses pelatihan selesai dan prediksi dihasilkan dalam bentuk skala ternormalisasi, *invers* normalisasi diterapkan untuk mengembalikan nilai ke skala harga asli (USD) [28]. Proses *invers* normalisasi ini penting agar metrik evaluasi seperti MAPE dan RMSE dapat dihitung pada skala harga yang bermakna dan mudah diinterpretasikan oleh investor dan praktisi keuangan [29].

2.4.3 Transformas Sliding Window

Model RNN seperti LSTM dan GRU memerlukan input dalam bentuk urutan (*sequence*) data historis yang berurutan, bukan nilai observasi tunggal yang terisolasi [25]. Tanpa teknik sliding window, model RNN tidak dapat memanfaatkan dependensi temporal karena setiap input hanya berisi satu titik data tanpa konteks historis [30]. Transformasi time series dilakukan menggunakan teknik *sliding window* untuk membentuk urutan data (sequence) yang dapat diproses oleh model RNN [28]. Setiap sampel input (X) dibentuk dari n data historis sebelumnya, dengan nilai target (y) adalah nilai pada waktu ke-t (langkah waktu berikutnya setelah akhir window) [28]. Rumus transformasinya adalah sebagai berikut [28]:

$X_t = [x_{t-n}, x_{t-n+1}, \dots, x_{t-1}]$ .....(2.2a)

$y_t = x_t$ .....(2.2b)

Dimana:

X_t : Vektor input (sequence) pada waktu ke-t

x_{t-n} : Nilai data pada n langkah waktu sebelum t (awal window)

x_{t-n+1} : Nilai data pada n – 1 langkah waktu sebelum t

x_{t-1} : Nilai data pada satu langkah waktu sebelum t (akhir window)

y_t : Nilai target yang diprediksi (nilai aktual pada waktu ke-t)

t : indeks waktu (time step)

n : Ukuran jendela (window size)

Window size n = 60 dipilih karena mewakili sekitar 2 bulan data harian perdagangan (asumsi 30 hari per bulan), yang dianggap cukup untuk menangkap berbagai pola temporal: pola jangka pendek (1-7 hari), pola siklus mingguan, serta tren jangka menengah (2 bulan)

[28]. Jika *window* terlalu kecil (misalnya 5-10 hari), model akan kehilangan konteks historis yang cukup untuk memahami pola jangka panjang seperti siklus bulanan [20]. Sebaliknya, jika *window* terlalu besar (misalnya 200-300 hari), jumlah sampel yang dihasilkan akan berkurang drastis sehingga data menjadi terbatas, dan risiko *overfitting* (model terlalu menghafal data latih) meningkat karena model memiliki terlalu banyak parameter relatif terhadap ukuran data [28]. Dalam penelitian ini, setiap sampel input memiliki dimensi (n, jumlah_fitur), yaitu (60, 1) untuk model tanpa integrasi sentimen, dan (60, 2) untuk model dengan integrasi sentimen (harga ternormalisasi + skor sentimen) [28]. Setelah transformasi sliding window, total 3.287 observasi harian yang tersedia menghasilkan 3.227 pasang sequence (karena 60 hari pertama tidak dapat membentuk *window* penuh), yang semuanya siap digunakan dalam tahap pelatihan dan pengujian model [28].

2.4.4 Pembagian Data

Pembagian data untuk prediksi *time series* harus memperhatikan urutan temporal (*temporal order*), dan tidak boleh dilakukan secara acak (*random sampling*) seperti pada data non-sekuensial klasifikasi gambar [28]. Jika pembagian dilakukan secara acak, akan terjadi kebocoran data (*data leakage*) yang serius, di mana informasi dari masa depan (data uji) digunakan secara tidak sah untuk melatih model karena urutan waktu menjadi kacau [20]. Model yang dilatih dengan kebocoran data akan menunjukkan performa yang terlalu optimis pada pengujian, tetapi akan gagal total ketika diterapkan pada data masa depan yang sesungguhnya [28]. Praktik yang benar dan standar dalam *time series forecasting* adalah membagi data berdasarkan urutan kronologis: proporsi pertama (awal periode) digunakan untuk melatih model, dan proporsi sisanya (akhir periode) digunakan untuk menguji model [20]. Dalam penelitian ini, dataset sekuensial yang telah melalui transformasi *sliding window* dibagi menjadi 80% data latih (*training*) yang diambil dari 80% pertama urutan *window*, dan 20% data uji (*testing*) yang diambil dari 20% terakhir urutan *window* (akhir periode) [28]. Tidak digunakan data validasi terpisah karena optimasi hyperparameter dilakukan menggunakan Optuna dengan validasi internal 5-*fold cross validation* yang secara otomatis membagi data latih menjadi subset training dan validation [31]. Total 3.227 pasang *sequence* menghasilkan 2.581 data latih (≈ 2.581 hari ≈ 7 tahun data) dan 646 data uji (≈ 646 hari ≈ 1,8 tahun data) [28]. Penggunaan data latih sebesar 80% (sekitar 7 tahun) memberikan cukup data untuk model mempelajari berbagai kondisi pasar (bull market, bear market, sideways), sementara 20% data uji (1,8 tahun) cukup untuk mengevaluasi performa model pada data yang belum pernah dilihat sebelumnya dalam periode yang cukup panjang untuk menangkap

siklus pasar [28]. Prinsip ini memastikan bahwa model diuji pada data yang secara temporal lebih baru dari data latih, tepatnya pada kondisi pasar yang belum terjadi pada saat model dilatih, sehingga mensimulasikan secara realistis kondisi dunia nyata di mana prediksi harus dibuat berdasarkan data historis saja tanpa pengetahuan tentang masa depan [28].

2.5 Arsitektur Model Deep Learning untuk Prediksi *Time Series*

Pendekatan *deep learning* merupakan bagian dari *machine learning* yang menggunakan arsitektur jaringan saraf dengan banyak lapisan tersembunyi (*multi-layer neural networks*) untuk mempelajari representasi data secara hierarkis dan otomatis tanpa rekayasa fitur manual [20]. Dalam prediksi *time series*, *deep learning* memiliki keunggulan fundamental dalam memodelkan hubungan non-linear yang kompleks serta pola temporal yang dinamis, tanpa memerlukan asumsi statistik yang ketat seperti stasioneritas dan linearitas yang diperlukan oleh metode konvensional [22]. Berbeda dengan pendekatan statistik tradisional yang umumnya bergantung pada asumsi distribusi normal dan hubungan linear antar variabel, model *deep learning* mampu melakukan pembelajaran adaptif secara *end-to-end* dari data mentah [20]. Kemampuan adaptif ini menjadikan *deep learning* sangat efektif untuk data yang bersifat non-stasioner (statistik berubah seiring waktu) dan dinamis, seperti yang umum terjadi pada pasar finansial [22]. Akibatnya, deep learning banyak digunakan dalam berbagai aplikasi prediksi *time series* finansial, termasuk peramalan harga saham, nilai tukar mata uang, volatilitas aset, dan tentu saja harga aset digital seperti Bitcoin [20].

Salah satu keunggulan utama *deep learning* untuk data sekuensial adalah kemampuannya yang superior dalam menangkap dependensi temporal jangka panjang (*long-term dependency*) [25]. Kemampuan ini sangat krusial pada data finansial yang sering kali memiliki keterkaitan antar waktu yang tidak hanya terbatas pada periode pendek (misalnya beberapa hari), tetapi juga pola yang terbentuk dari kejadian yang terjadi berminggu-minggu atau berbulan-bulan sebelumnya [30]. Dalam konteks Bitcoin, peristiwa *halving* yang terjadi setiap 4 tahun sekali memiliki dampak harga yang muncul secara bertahap selama 6-12 bulan setelah peristiwa, sehingga model harus mampu "mengingat" informasi dari 200-300 langkah waktu sebelumnya [13]. Model *deep learning* berbasis *Recurrent Neural Network* (RNN) dan variannya dirancang secara khusus untuk mengatasi keterbatasan model statistik tradisional dengan cara mempertahankan memori internal (*hidden state*) yang membawa informasi historis melalui proses pembelajaran secara berulang [25]. Namun, RNN standar memiliki keterbatasan serius yaitu masalah *vanishing*

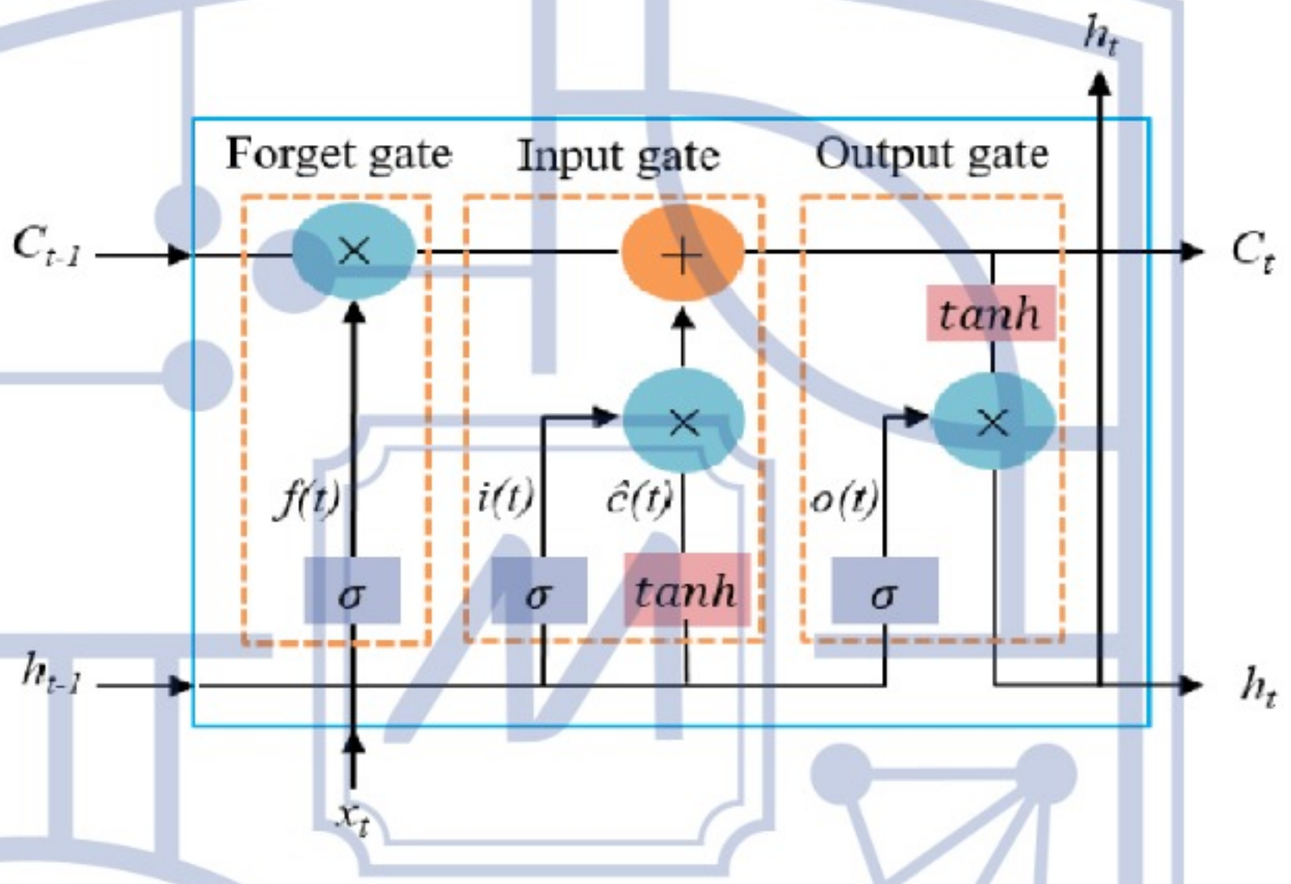
gradient (gradien yang sangat kecil saat propagasi mundur melalui banyak langkah waktu), sehingga sulit mempelajari dependensi yang sangat panjang (lebih dari 10-20 langkah waktu) [32]. LSTM (*Long Short-Term Memory*) dan GRU (*Gated Recurrent Unit*) secara eksplisit dirancang untuk mengatasi masalah *vanishing gradient* ini melalui mekanisme gerbang (*gates*) yang mengatur aliran informasi secara adaptif, seperti yang akan dijelaskan lebih detail pada subbab selanjutnya [25], [30].

Seiring berkembangnya penelitian di bidang keuangan dan komputasi, deep learning semakin banyak diterapkan pada prediksi harga *cryptocurrency* karena volatilitas dan kompleksitasnya yang ekstrem [13]. Tingginya volatilitas dan kompleksitas pergerakan harga aset kripto yang sering kali tidak mengikuti pola statistik sederhana menjadikan pendekatan *deep learning* lebih unggul dibandingkan metode konvensional dalam menangkap pola pergerakan harga yang dinamis dan non-linear [20]. Banyak penelitian empiris melaporkan bahwa LSTM dan GRU secara konsisten mengungguli ARIMA, GARCH, dan model ekonometrik tradisional lainnya dalam memprediksi harga saham dan indeks keuangan [22]. Sebagai bagian dari *deep learning*, model RNN dan variannya seperti LSTM dan GRU dirancang khusus untuk memproses data sekuensial dengan mempertahankan informasi dari waktu ke waktu melalui mekanisme memori internal [25]. Kedua model ini merupakan pengembangan dari RNN standar yang secara fundamental mampu mengatasi permasalahan vanishing gradient di mana gradien menjadi sangat kecil saat propagasi mundur sehingga lapisan awal tidak belajar serta secara signifikan meningkatkan kemampuan dalam menangkap dependensi jangka panjang pada data time series yang kompleks [30], [32]. Dalam penelitian ini, LSTM dan GRU digunakan sebagai model utama untuk memprediksi harga Bitcoin karena keduanya telah terbukti efektif dalam berbagai studi prediksi time series finansial dan memiliki arsitektur yang berbeda sehingga memungkinkan perbandingan performa yang menarik [13], [20].

2.6 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) merupakan salah satu varian dari *Recurrent Neural Network* (RNN) yang dirancang untuk mengatasi permasalahan *vanishing gradient* yang menjadi kelemahan fundamental RNN standar pada data sekuensial panjang [25]. LSTM pertama kali diperkenalkan oleh Hochreiter dan Schmidhuber pada tahun 1997 untuk memungkinkan model mempertahankan informasi dalam jangka waktu yang sangat panjang tanpa degradasi gradien, sehingga secara signifikan lebih efektif dalam memodelkan dependensi temporal jangka panjang pada data time series yang kompleks [25]. Berbeda

dengan node tersembunyi biasa pada RNN yang hanya memiliki satu fungsi aktivasi sederhana, setiap unit dalam lapisan tersembunyi LSTM digantikan oleh sebuah memory cell (sel memori) yang dirancang khusus untuk menyimpan informasi secara stabil dalam rentang waktu yang panjang [32]. Stabilisasi ini dicapai melalui mekanisme yang disebut constant error carousel (CEC), yaitu sebuah koneksi rekuren internal dengan bobot tetap bernilai 1 yang secara matematis mencegah gradien menghilang maupun meledak saat propagasi balik dilakukan berkali-kali [32]. Arsitektur LSTM ditunjukkan pada Gambar 2.1 [25].



Gambar 2.1 Arsitektur LSTM *[25]*

Komponen utama dalam arsitektur LSTM terdiri dari *input node*, *input gate*, *forget gate*, *cell state*, dan *output gate* [32]. *Forget gate* yang diperkenalkan oleh Gers et al. pada tahun 2000 merupakan inovasi penting yang memungkinkan model secara adaptif mengosongkan memori yang tidak lagi relevan, sebuah kemampuan yang tidak dimiliki oleh LSTM versi awal 1997 [32]. *Input gate* bertugas mengontrol seberapa banyak informasi baru yang akan disimpan ke dalam *cell state* berdasarkan kandidat yang dihasilkan oleh input node, sementara input node menghasilkan kandidat informasi baru menggunakan fungsi aktivasi tanh [32]. *Cell state* (C_t) kemudian diperbarui dengan menggabungkan kontribusi dari forget gate (mengalikan memori lama dengan faktor antara 0 dan 1) dan input gate (menambahkan informasi baru yang telah disaring) [32]. Mekanisme gerbang ini memberikan LSTM kemampuan adaptif untuk memutuskan secara dinamis informasi mana yang perlu diingat, diperbarui, atau dilupakan pada setiap langkah waktu sebuah properti yang sangat berharga untuk data finansial yang berubah-ubah secara dinamis [25]. sebagaimana dirumuskan sebagai berikut [25], [32]:

Mencari Forget Gate :

$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$ .....(2.3)

Dimana :

f_t : *Forget gate*

σ : Fungsi aktivasi *Sigmoid*

W : Bobot Jaringan

h_t : *Hidden state*

x_t : Input pada waktu t

b : Bias

Mencari Input Gate :

$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$*.....(2.4)

Dimana :

i_t : *Input gate*

σ : Fungsi aktivasi *Sigmoid*

W : Bobot Jaringan

h_t : *Hidden state*

x_t : Input pada waktu t

b : Bias

Mencari Candidate Cell State :

$\tilde{C}_t = tanh(W_c[h_{t-1}, x_t] + b_c$ .....(2.5)

Dimana :

$\tilde{C}_t$: *Candidate cell state*

$tanh$: Fungsi aktivasi *hyperbolic tangent*

W : Bobot Jaringan

h_t : *Hidden state*

x_t : Input pada waktu t

b : Bias

Mencari Update Cell State :

$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$ .....(2.6)

Dimana :

C_t : *Cell state*

f_t : *Forget gate*

i_t : *Input gate*

$\tilde{C}_t$: *Candidate cell state*

Mencari Output Gate :

$o_t = \sigma(W_f[h_{t-1}, x_t] + b_o$ .....(2.7)

Dimana :

o_t : *Output gate*

σ : Fungsi aktivasi *Sigmoid*

W : Bobot Jaringan

h_t : *Hidden state*

x_t : Input pada waktu t

b : Bias

Mencari Hidden State:

$h_t = o_t \times tanh(C_t)$ .....(2.8)

Dimana :

h_t : *Hidden state*

o_t : *Output gate*

$tanh$: Fungsi aktivasi *hyperbolic tangent*

C_t : *Cell state*

Alur di LSTM :

1. *Forget gate* : menentukan informasi lama yang dibuang
2. *Input gate* : menentukan informasi baru yang diterima
3. *Candidate cell state* $\tilde{C}_t$: menghasilkan calon informasi baru
4. *Cell state* $C_t$diperbarui
5. *Output gate* menghasilkan *hidden state*

Mekanisme kelima komponen ini (forget gate, input gate, candidate, cell state, output gate) bekerja secara sinergis: forget gate memutuskan apa yang dibuang, input gate memutuskan apa yang disimpan dari kandidat baru, cell state diperbarui dengan kombinasi keduanya, dan output gate memutuskan bagian mana dari memori yang dikeluarkan sebagai hidden state [32]. Dengan mekanisme tersebut, LSTM mampu mempelajari pola temporal yang sangat kompleks tanpa kehilangan informasi penting dari periode sebelumnya yang mungkin berlangsung ratusan langkah waktu lalu [25]. Proses pembelajaran pada LSTM bersifat adaptif terhadap pola data yang berbeda karena bobot-bobot gerbang dipelajari dari data, bukan ditetapkan secara manual [32]. Dalam konteks prediksi time series finansial,

LSTM telah banyak digunakan dan menjadi standar emas (gold standard) karena kemampuannya yang superior dalam menangkap pola non-linear serta hubungan jangka panjang antar data historis [20]. Model ini terbukti secara empiris lebih unggul dibandingkan RNN konvensional dalam memodelkan data finansial yang non-stasioner dan fluktuatif, sehingga menghasilkan performa prediksi yang lebih stabil dan akurat pada berbagai metrik evaluasi [22]. Pada pasar cryptocurrency, LSTM sering diterapkan untuk prediksi harga Bitcoin karena karakteristik data yang volatil dan kompleks, dan sejumlah penelitian menunjukkan LSTM mampu memberikan performa prediksi yang kompetitif, baik pada prediksi jangka pendek (daily) maupun jangka panjang hingga beberapa bulan ke depan [13]. Oleh karena itu, LSTM dipilih sebagai salah satu model utama dalam penelitian ini untuk mengevaluasi kinerja prediksi harga Bitcoin pada horizon waktu yang lebih panjang dibandingkan dengan GRU [13].

2.6.1 Stacked LSTM

Stacked LSTM adalah arsitektur yang menggunakan lebih dari satu lapisan LSTM yang disusun secara bertingkat (stacked), di mana output hidden state dari suatu lapisan menjadi input sequence bagi lapisan berikutnya bukan hanya input ke lapisan dense, tetapi input ke seluruh rangkaian timestep lapisan LSTM berikutnya [33]. Susunan lapisan pada Stacked LSTM bersifat sekuensial (*sequential*), bukan paralel (*parallel*), yang berarti bahwa setiap lapisan LSTM diproses secara berurutan satu per satu, dan hidden state dari lapisan ke-l pada setiap timestep menjadi input bagi lapisan ke-(l+1) pada timestep yang sama [34], sebagaimana ditunjukkan pada Persamaan (2.7). Pendekatan bertingkat ini memungkinkan jaringan untuk membangun representasi fitur yang lebih kompleks dan hierarkis dari data sekuensial, karena setiap level abstraksi dipelajari oleh lapisan yang berbeda [33]. Lapisan pertama (bawah) cenderung menangkap pola dasar urutan data misalnya fluktuasi harga harian atau pola musiman sederhana sementara lapisan kedua dan seterusnya (atas) mempelajari representasi yang lebih abstrak dan kompleks, seperti pola interaksi antar fitur atau tren jangka panjang yang tidak terlihat secara kasat mata [35]. Struktur hierarkis ini terbukti secara empiris membantu model dalam memahami dependensi jangka panjang pada data time series yang kompleks, karena setiap lapisan dapat fokus pada skala waktu yang berbeda [35]. Persamaan umum *Stacked* RNN [33]:

Mencari Untuk lapisan ke-l:

$$h_t^{(l)} = f(W^{(l)} h_t^{(l-1)} + U^{(l)} h_{t-1}^{(l)} + b^{(l)}) \quad \text{..... (2.9)}$$

Dimana :

$h_t^{(l)}$: hidden state pada waktu t di lapisan ke-l

f : fungsi aktivasi (LSTM atau GRU)

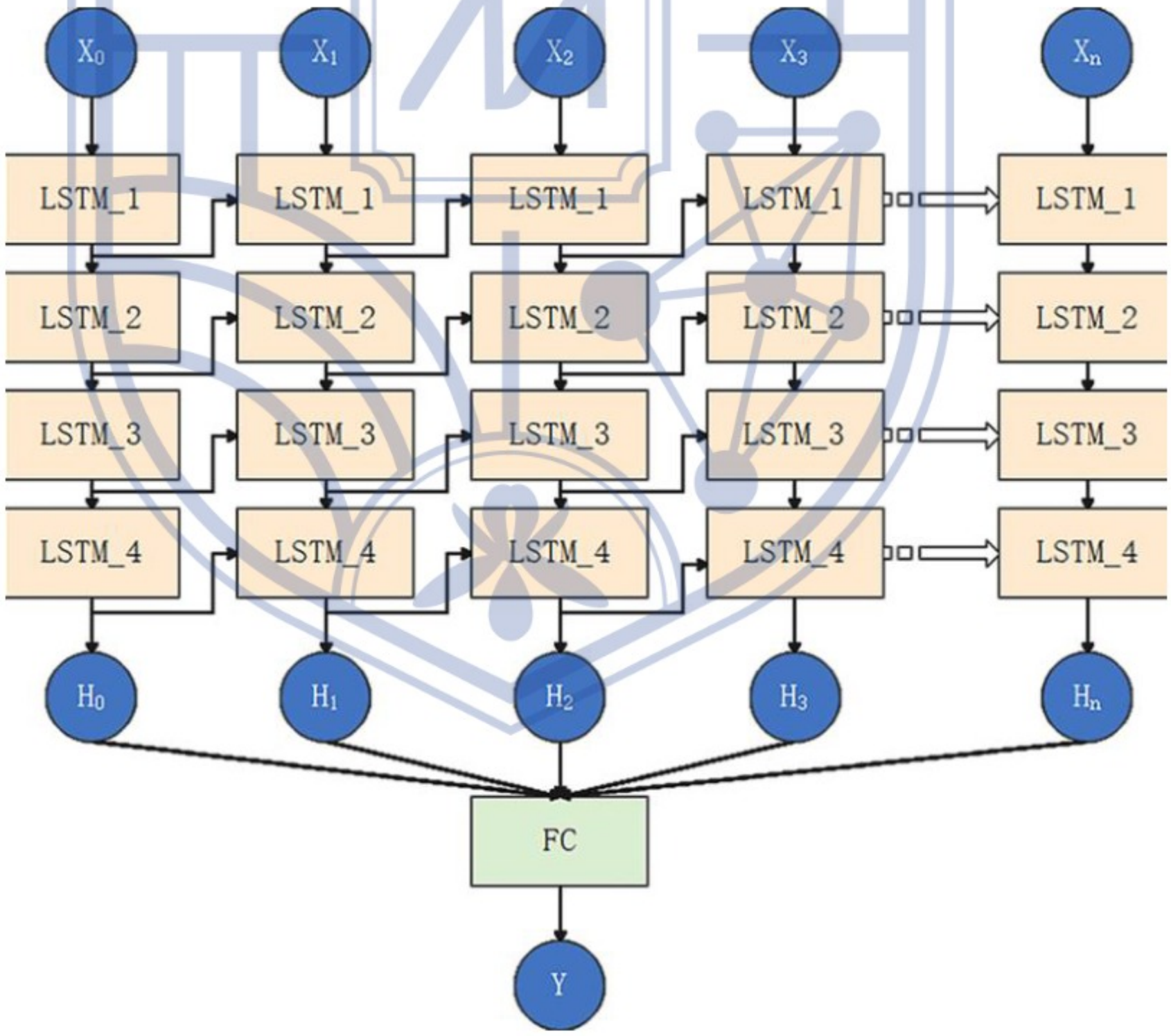
$W^{(l)}, U^{(l)}$: matriks bobot

$h_t^{(l-1)}$: output dari lapisan sebelumnya

$h_{t-1}^{(l)}$: hidden state sebelumnya pada lapisan yang sama

$b^{(l)}$: bias

Pada *Stacked* LSTM, penumpukan beberapa lapisan LSTM secara berurutan menjadikan jaringan lebih dalam, sehingga setiap lapisan dapat memproses representasi dari lapisan sebelumnya secara hierarkis. Lapisan bawah cenderung menangkap pola dasar urutan data, sementara lapisan lebih tinggi mempelajari representasi yang lebih abstrak. Struktur ini terbukti membantu model dalam memahami dependensi jangka panjang pada data *time series* [35]. Arsitektur stacked LSTM ditunjukkan pada Gambar 2.2 [32].



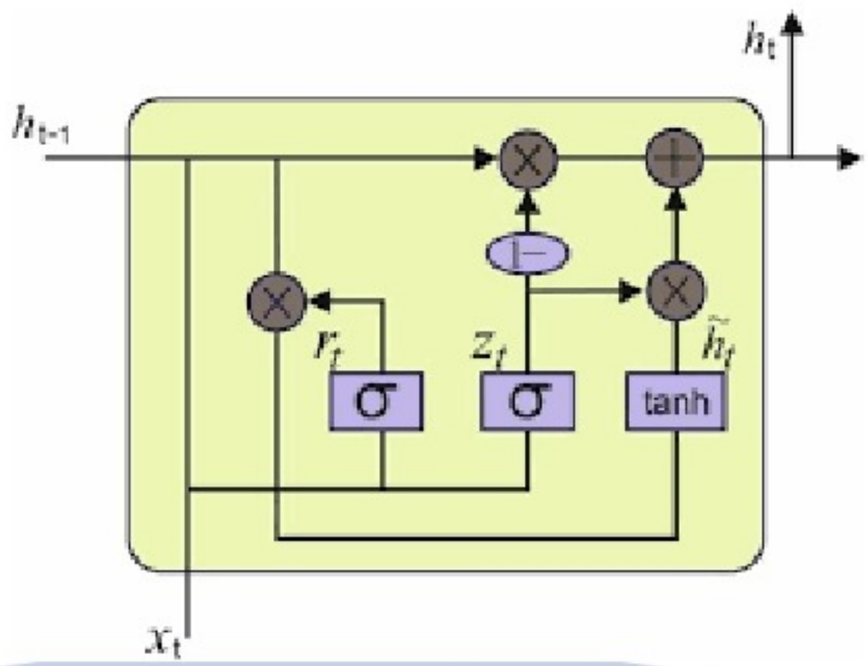
Gambar 2.2 Arsitektur Stacked LSTM [32]

Salah satu konfigurasi yang umum diterapkan pada Stacked LSTM untuk tugas prediksi adalah pola *many-to-one*, di mana model menerima urutan beberapa langkah waktu sebagai masukan (sequence input) dan menghasilkan satu nilai keluarkan (prediksi harga pada suatu langkah ke depan) [34]. Pada konfigurasi ini, bentuk sequence masukan berupa tensor tiga dimensi dengan dimensi [jumlah sampel, panjang window (n), jumlah fitur], yang diproses secara bertingkat oleh setiap lapisan LSTM [33]. Setiap lapisan LSTM memproses seluruh urutan timestep dan menghasilkan hidden state pada setiap timestep, tetapi hanya hidden state terakhir (pada akhir window) yang diteruskan ke lapisan dense untuk menghasilkan prediksi tunggal [34]. Konfigurasi many-to-one ini sangat cocok untuk tugas prediksi harga karena data historis dalam suatu window (60 hari terakhir) dimanfaatkan untuk memprediksi hanya satu nilai di masa mendatang (hari ke-61) [34]. Dengan menggunakan stacked LSTM, model dapat mengeksploitasi kedalaman jaringan untuk menangkap pola yang lebih kaya dan kompleks dibandingkan LSTM satu lapisan [33]. Dalam penelitian ini, stacked LSTM digunakan untuk mengevaluasi apakah penambahan kedalaman lapisan dapat secara signifikan meningkatkan kapasitas model dalam menangkap pola kompleks pada harga Bitcoin yang sarat dengan volatilitas dan noise, meskipun dengan risiko overfitting yang lebih tinggi jika tidak diatur dengan regularisasi yang tepat seperti dropout [34].

2.7 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) merupakan salah satu varian dari *Recurrent Neural Network* (RNN) yang diperkenalkan oleh Cho et al. pada tahun 2014 sebagai alternatif yang lebih sederhana dan efisien secara komputasi dibandingkan LSTM, tanpa mengorbankan kemampuan menangkap dependensi temporal jangka panjang secara signifikan [18]. GRU dirancang untuk mengatasi permasalahan vanishing gradient sama seperti LSTM namun melakukannya dengan struktur arsitektur yang lebih ringkas dan jumlah parameter yang lebih sedikit, sehingga lebih cepat dilatih dan lebih hemat memori [22].

Berbeda dengan LSTM yang memiliki tiga gerbang utama (*input gate*, *forget gate*, dan *output gate*), GRU hanya menggunakan dua mekanisme gerbang, yaitu *update gate* dan *reset gate*. Penyederhanaan ini memungkinkan GRU untuk mengatur aliran informasi historis secara efisien, sekaligus mengurangi kompleksitas komputasi dan jumlah *parameter* yang harus dipelajari selama proses pelatihan [30]. Arsitektur GRU ditunjukkan pada Gambar 2.3 [22].



Gambar 2.3 Arsitektur GRU [22]

Arsitektur *Gated Recurrent Unit* (GRU) diperkenalkan oleh Cho et al. sebagai varian yang lebih sederhana dari LSTM, dengan mengusulkan unit tersembunyi baru yang terdiri dari dua komponen gerbang utama yaitu *update gate* dan *reset gate* yang berfungsi mengatur aliran informasi secara adaptif pada setiap *timestep* [22]. Berbeda dengan LSTM yang memiliki *cell state* terpisah sebagai memori jangka panjang, GRU menyederhanakan arsitektur ini dengan menjadikan *hidden state* sebagai satu-satunya memori dalam proses pembelajaran, sehingga jumlah parameter yang perlu dilatih menjadi lebih sedikit dan proses pelatihan dapat berlangsung lebih efisien [22].

Pada setiap *timestep* t, GRU menerima input $x_t$serta *hidden state* sebelumnya h_{t-1}. Proses pertama adalah menghitung *reset gate* $r_t$yang berfungsi menentukan seberapa banyak informasi dari hidden state sebelumnya yang akan dilupakan ketika menghasilkan kandidat *hidden state* baru [22]. *Reset gate* dirumuskan sebagai berikut [22]:

Mencari Update Gate :

$$z_t = \sigma(W_z[h_{t-1}, x_t]) \quad (2.10)$$

Dimana :

z_t: *Update gate*

σ : Fungsi aktivasi *Sigmoid*

W : Bobot jaringan

h_t : *Hidden state*

x_t : Input pada waktu t

Mencari Reset Gate :

$$r_t = \sigma(W_r[h_{t-1}, x_t]) \quad (2.11)$$

Dimana :

r_t : *Reset gate*

σ : Fungsi aktivasi *Sigmoid*

W : Bobot jaringan

h_t : *Hidden state*

x_t : Input pada waktu t

Mencari Candidate Hidden State :

$$\tilde{h}_t = tanh(W_h[r_t \times h_{t-1}, x_t]) \quad (2.12)$$

Dimana :

$\tilde{h}_t$: *Candidate hidden state*

$tanh$: Fungsi aktivasi *hyperbolic tangent*

W : Bobot jaringan

r_t : *Reset gate*

h_t : *Hidden state*

x_t : Input pada waktu t

Mencari Hidden State Baru :

$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h}_t \quad (2.13)$$

Dimana :

h_t : *Hidden state*

z_t : *Update gate*

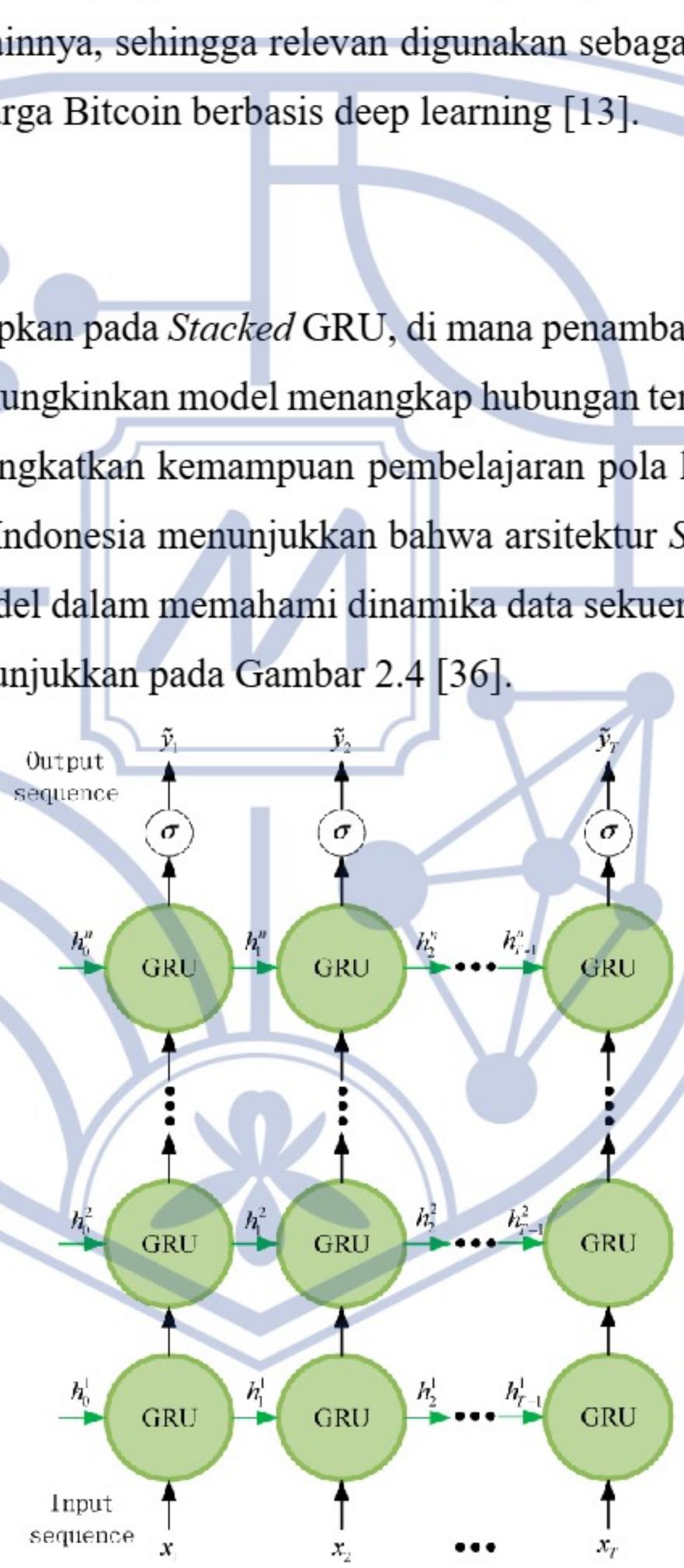
$\tilde{h}_t$: *Candidate hidden state*

Interpretasi dari mekanisme GRU adalah: ketika update gate z_t mendekati 1, hidden state baru didominasi oleh candidate $h_t$yang kaya informasi baru; ketika z_t mendekati 0, hidden state baru didominasi oleh hidden state lama h_{t-1}, sehingga informasi lama dipertahankan hampir tanpa perubahan [22]. Reset gate r_t mengontrol seberapa besar hidden state sebelumnya diperhitungkan dalam candidate: jika r_t mendekati 0, maka candidate h_t hampir hanya bergantung pada input saat ini x_t, mengabaikan konteks historis [18]. Dalam konteks prediksi time series finansial, GRU menunjukkan performa yang sangat kompetitif dalam memodelkan data yang non-linear dan fluktuatif, seringkali menyamai atau bahkan melampaui LSTM dalam berbagai tugas peramalan harga [20]. Beberapa penelitian melaporkan bahwa GRU mampu menghasilkan akurasi prediksi yang sebanding, dan dalam beberapa kasus lebih baik, dibandingkan LSTM, terutama ketika diterapkan pada dataset dengan ukuran relatif besar (jutaan sampel) dan horizon prediksi yang lebih panjang (minggu

hingga bulan) [22]. Keunggulan GRU terletak pada efisiensi komputasinya sekitar 25-30% lebih cepat dari LSTM dengan ukuran hidden yang sama karena jumlah parameter yang lebih sedikit sehingga propagasi forward dan backward memerlukan operasi matriks yang lebih sedikit [22]. Pada prediksi harga cryptocurrency, GRU banyak digunakan karena kemampuannya menangkap dinamika temporal data harga yang volatil tanpa memerlukan sumber daya komputasi yang besar [13]. Studi terkini menunjukkan bahwa GRU dapat memberikan performa prediksi yang stabil dan akurat pada pemodelan harga Bitcoin, Ethereum, dan aset kripto lainnya, sehingga relevan digunakan sebagai model pembanding dalam penelitian prediksi harga Bitcoin berbasis deep learning [13].

2.7.1 Stacked GRU

Prinsip serupa diterapkan pada *Stacked* GRU, di mana penambahan beberapa lapisan GRU secara bertingkat memungkinkan model menangkap hubungan temporal pada berbagai skala waktu sekaligus meningkatkan kemampuan pembelajaran pola kompleks. Penelitian pada prediksi harga saham Indonesia menunjukkan bahwa arsitektur *Stacked* GRU mampu meningkatkan performa model dalam memahami dinamika data sekuensial yang rumit [36]. Arsitektur stacked GRU ditunjukkan pada Gambar 2.4 [36].



Gambar 2.4 Arsitektur Stacked GRU [36]

Sama halnya dengan stacked LSTM, susunan lapisan pada stacked GRU juga bersifat sekuensial, di mana output hidden state di setiap timestep dari lapisan GRU pertama diteruskan sebagai input sequence (untuk seluruh timestep) ke lapisan GRU berikutnya

secara berurutan [37]. Setiap lapisan yang lebih tinggi (deeper) memproses representasi abstrak yang dihasilkan lapisan sebelumnya, sehingga lapisan yang lebih dalam dapat menangkap pola temporal pada tingkat abstraksi yang lebih tinggi dan dalam skala waktu yang lebih panjang [24]. Stacked GRU dapat dimanfaatkan untuk melakukan temporal encoding (pengkodean temporal) terhadap data hasil ekstraksi fitur, sehingga model mampu menangkap long-range dependencies (ketergantungan jarak jauh) dari representasi sekuensial yang dihasilkan oleh lapisan-lapisan sebelumnya [37]. Dalam arsitektur yang diusulkan, keluaran dari setiap lapisan GRU tetap berbentuk urutan (sequence) yang mempertahankan dimensi waktu, yang selanjutnya diproses kembali oleh lapisan GRU berikutnya, sehingga menghasilkan representasi sekuensial yang lebih mendalam, kaya informasi, dan adaptif terhadap berbagai skala waktu [37]. Dalam konteks penelitian ini, penggunaan arsitektur Stacked LSTM dan Stacked GRU bertujuan untuk meningkatkan kapasitas model dalam mempelajari pola fluktuasi harga Bitcoin yang memiliki volatilitas tinggi dan dinamika temporal yang kompleks [14]. Dengan memanfaatkan beberapa lapisan rekuren yang disusun secara hierarkis, model diharapkan mampu menangkap dependensi jangka panjang serta hubungan temporal yang rumit pada data historis harga Bitcoin, sehingga menghasilkan prediksi yang lebih akurat dibandingkan model single-layer [24].

2.8 Analisis Sentimen

Analisis sentimen merupakan cabang fundamental dari Natural Language Processing (NLP) yang secara sistematis bertujuan untuk mengidentifikasi, mengekstraksi, dan mengklasifikasikan opini, emosi, atau sikap yang terkandung dalam teks, umumnya ke dalam tiga kategori: positif, negatif, atau netral [38]. Dalam konteks finansial, analisis sentimen digunakan secara luas untuk memahami persepsi dan reaksi pasar terhadap berbagai informasi yang disampaikan melalui berita keuangan, laporan tahunan perusahaan (annual reports), transkrip earnings call, maupun postingan media sosial dan platform daring [38]. Metode analisis sentimen dapat dibagi menjadi tiga pendekatan utama: (1) pendekatan berbasis leksikon (lexicon-based) seperti VADER (Valence Aware Dictionary and sEntiment Reasoner) dan TextBlob yang menggunakan kamus kata bermuatan sentimen dengan skor yang telah ditentukan sebelumnya; (2) pendekatan machine learning tradisional seperti Support Vector Machine (SVM), Naïve Bayes, dan Logistic Regression yang memerlukan ekstraksi fitur manual; serta (3) pendekatan deep learning seperti LSTM, CNN, dan Transformer yang secara otomatis mempelajari representasi fitur dari data [25]. Pendekatan leksikon menawarkan kesederhanaan implementasi namun seringkali gagal

menangkap konteks kalimat yang kompleks, terutama ketika kata bermuatan positif/negatif dibalik oleh negasi (misalnya "not good") atau digunakan secara sarkastik [38]. Sebaliknya, pendekatan deep learning, terutama yang berbasis Transformer, mampu mempelajari representasi semantik teks secara hierarkis dan kontekstual (memperhatikan kata-kata di sekitarnya), sehingga jauh lebih unggul, khususnya pada domain finansial yang memiliki terminologi teknis dan konteks yang sangat spesifik [39].

Pada pasar keuangan tradisional maupun pasar cryptocurrency, sentimen investor memiliki peran yang sangat penting dalam memengaruhi pergerakan harga aset secara fundamental [38]. Informasi yang bersifat positif seperti laporan laba yang melebihi ekspektasi analis, akuisisi strategis, atau persetujuan regulasi yang menguntungkan dapat memicu kenaikan harga yang signifikan dalam hitungan jam atau bahkan menit [38]. Sebaliknya, sentimen negative misalnya berita tentang kebangkrutan mitra bisnis, investigasi regulator, atau serangan siber dapat menyebabkan penurunan harga yang tajam dan cepat [38]. Efek ini sering terjadi bahkan sebelum perubahan fundamental (seperti arus kas atau profitabilitas) tercermin pada data harga historis, karena pelaku pasar bereaksi terhadap berita dan informasi baru secara cepat, seringkali secara irasional atau berlebihan (overreaction) [38]. Dalam pasar cryptocurrency, pengaruh sentimen pasar cenderung jauh lebih kuat dibandingkan pasar keuangan tradisional karena beberapa faktor: (a) pasar kripto masih relatif muda dan belum matang, sehingga investor lebih mudah terpengaruh oleh berita; (b) kurangnya regulasi yang seragam di berbagai negara menyebabkan ketidakpastian tinggi; (c) tingginya proporsi investor ritel (individual) dibandingkan institusi, yang cenderung lebih emosional; dan (d) tingginya ketergantungan investor terhadap informasi yang beredar di media sosial (Twitter, Reddit, Telegram) serta platform berita daring karena kurangnya laporan fundamental yang terstruktur seperti di perusahaan publik [4]. Sejumlah penelitian empiris menunjukkan bahwa sentimen berita dan opini publik memiliki korelasi yang signifikan, baik positif maupun negatif, dengan pergerakan harga Bitcoin dan aset kripto lainnya, terutama pada periode volatilitas tinggi [40]. Oleh karena itu, analisis sentimen sering dimanfaatkan sebagai indikator tambahan (feature) untuk melengkapi analisis berbasis data historis harga semata, yang seringkali tidak cukup untuk menangkap lonjakan atau kejatuhan harga yang dipicu oleh berita [38].

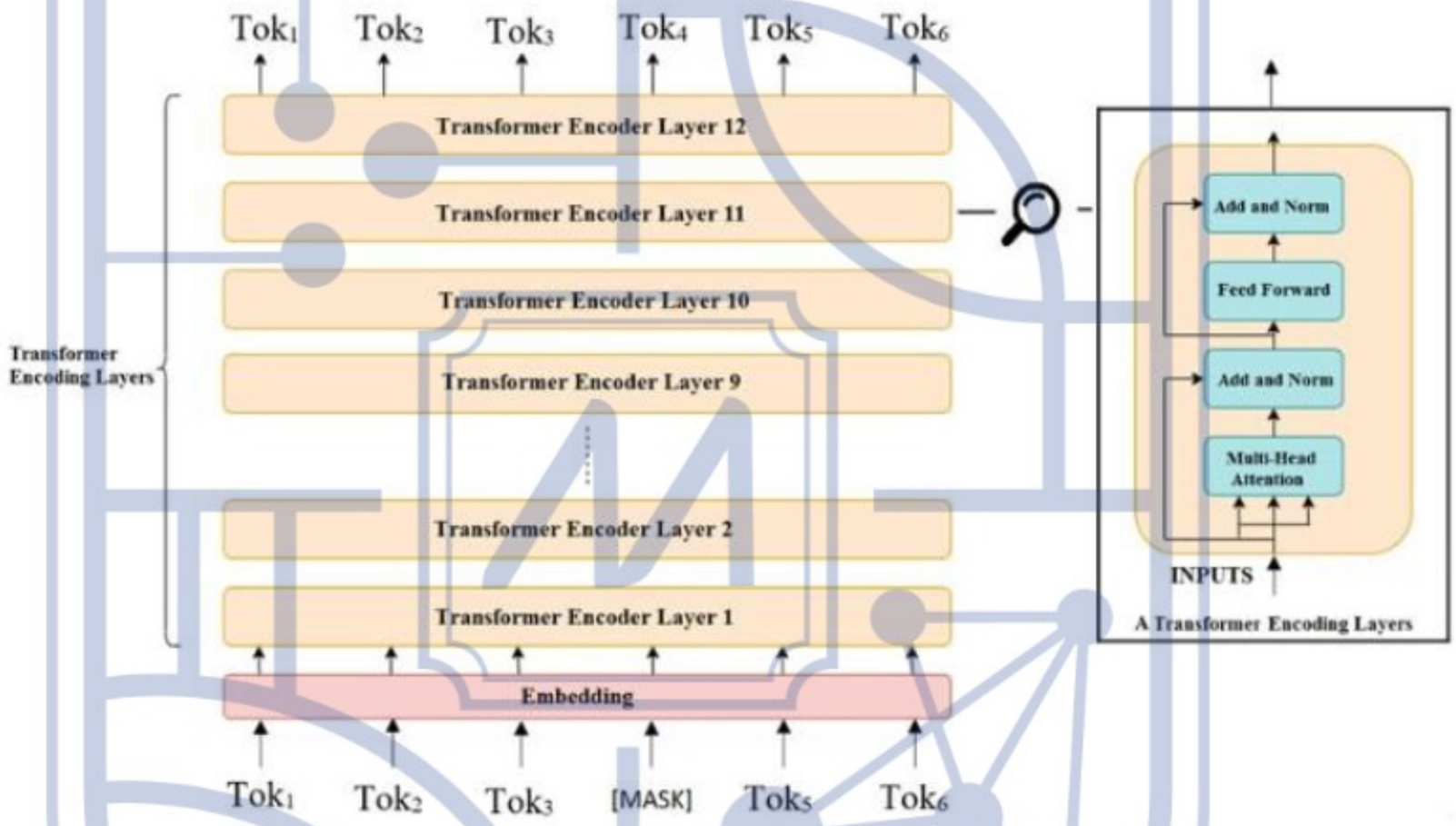
Secara umum, pipeline lengkap NLP untuk analisis sentimen teks finansial terdiri dari enam tahapan utama yang dilakukan secara berurutan: (1) pengumpulan teks (text collection) dari sumber berita atau media daring melalui API atau web scraping; (2) praproses teks (text preprocessing) meliputi cleaning (penghapusan HTML tags, URL,

angka), case folding (konversi ke huruf kecil), dan penghapusan karakter khusus; (3) tokenisasi (tokenization) yaitu pemecahan teks menjadi unit-unit token (kata, subword, atau karakter); (4) representasi teks (text representation) berupa konversi token menjadi vektor numerik berdimensi tinggi melalui mekanisme embedding seperti Word2Vec, GloVe, atau contextual embedding BERT; (5) pemodelan (modeling) yaitu pemrosesan representasi teks menggunakan arsitektur machine learning/deep learning (SVM, LSTM, CNN, atau Transformer) untuk mengekstraksi makna semantik dan pola linguistik; dan (6) klasifikasi sentimen (sentiment classification) berupa prediksi label sentimen menggunakan fungsi softmax pada layer output [39]. Dalam konteks penelitian ini, FinBERT (Financial BERT) menempati posisi sentral pada tahap representasi teks (embedding) dan pemodelan (Transformer encoder), di mana model ini memproses token hasil tokenisasi WordPiece melalui 12 lapisan encoder Transformer untuk menghasilkan representasi kontekstual yang kaya, yang kemudian diteruskan ke lapisan klasifikasi (dense layer + softmax) untuk menghasilkan label sentimen berita cryptocurrency (positif/negatif/netral) [39]. Dengan demikian, analisis sentimen menjadi jembatan kritis antara data tekstual tidak terstruktur (berita cryptocurrency mentah) dan data numerik terstruktur (harga Bitcoin) dalam model prediksi hibrida yang diusulkan dalam penelitian ini [40].

2.9 Bidirectional Encoder Representations from Transformers (BERT)

BERT (Bidirectional Encoder Representations from Transformers) adalah model representasi bahasa berbasis arsitektur Transformer yang diperkenalkan oleh Devlin et al. dari Google pada tahun 2018, dan sejak saat itu menjadi fondasi bagi hampir semua model NLP mutakhir (state-of-the-art) [41]. BERT secara fundamental dirancang untuk menghasilkan representasi kata yang bersifat kontekstual (contextual word embeddings), yaitu representasi vektor numerik yang dinamis dan bergantung pada konteks penggunaan kata dalam kalimat berbeda dengan embedding statis tradisional seperti Word2Vec yang memberikan vektor yang sama untuk kata yang sama di semua konteks [42]. Keunggulan utama BERT terletak pada sifatnya yang bidirectional (dua arah), yang memungkinkan model untuk memperhatikan dan memanfaatkan informasi dari kedua sisi suatu kata—baik konteks yang berada di sebelah kiri (kata-kata sebelumnya) maupun di sebelah kanan (kata-kata sesudahnya) secara simultan dalam satu lapisan yang sama [41]. Sebelum BERT, model seperti ELMo (Embeddings from Language Models) menggunakan pendekatan dua arah tetapi secara terpisah (forward LSTM dan backward LSTM yang kemudian digabungkan), sehingga tidak sepenuhnya bidirectional karena kedua arah tidak berinteraksi dalam lapisan

yang sama [41]. BERT dibangun berdasarkan arsitektur Transformer encoder yang disusun secara berlapis, di mana setiap encoder terdiri dari dua komponen utama: (a) mekanisme multi-head self-attention yang memungkinkan setiap token untuk "memperhatikan" token lain dalam kalimat; dan (b) feed-forward network (position-wise) yang memproses output attention [42]. BERT hadir dalam dua ukuran utama: BERT_BASE memiliki 12 lapisan encoder, 768 unit hidden size, 12 kepala attention, dan total sekitar 110 juta parameter; sementara BERT_LARGE memiliki 24 lapisan encoder, 1024 hidden size, 16 kepala attention, dan total sekitar 340 juta parameter [41]. Arsitektur BERT ditunjukkan pada Gambar 2.5 [42].



Gambar 2.5 Arsitektur BERT [41]

Pada tahap input, BERT merepresentasikan teks melalui tiga jenis embedding yang dijumlahkan (element-wise addition) untuk menghasilkan representasi input yang kaya informasi [41]. Token embedding merepresentasikan setiap token individual menggunakan lookup table yang dipelajari selama pre-training; positional embedding memberikan informasi tentang posisi token dalam urutan karena self-attention sendiri tidak memiliki konsep urutan; dan segment embedding membedakan antara kalimat A dan kalimat B pada tugas Next Sentence Prediction (NSP) [41]. Token khusus [CLS] (classification token) ditambahkan di awal sekuens, yang representasi finalnya dari lapisan encoder terakhir digunakan sebagai representasi agregat seluruh kalimat untuk tugas klasifikasi [41]. Token khusus [SEP] (separator token) ditambahkan di akhir setiap kalimat untuk memisahkan kalimat A dan B [41]. Setelah melalui seluruh lapisan encoder Transformer, representasi token [CLS] dari lapisan terakhir (biasanya berupa vektor berdimensi 768 untuk BERT_BASE) digunakan sebagai input ke lapisan klasifikasi tambahan untuk tugas

downstream [41]. Mekanisme self-attention adalah inti matematis dari BERT rumus dasarnya adalah sebagai berikut [42]:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.14)$$

Dimana:

Q : Query matrix
K : Key matrix
V : Value matrix
$softmax$: Fungsi probabilitas
K^T : Transpose dari matrix key
d_k : Dimensi vector

BERT dilatih (*pre-training*) secara tidak terawasi (*unsupervised*) menggunakan dua tugas utama pada korpus teks besar yang terdiri dari 3.3 miliar kata dari Wikipedia dan BookCorpus [41]. Tugas pertama adalah *Masked Language Model* (MLM): sekitar 15% token dalam setiap sekuens input dipilih secara acak untuk dimasking (disembunyikan) dengan *special token* [MASK]; model kemudian dilatih untuk memprediksi token asli berdasarkan konteks di sekitarnya (kiri dan kanan) [41]. Dari 15% token yang dipilih untuk masking, 80% diganti dengan token [MASK], 10% diganti dengan token acak dari *vocabulary* (untuk mengurangi *mismatch* antara *pre-training* dan *fine-tuning*), dan 10% dipertahankan tidak berubah [41]. Pendekatan ini memungkinkan pembelajaran representasi bahasa yang benar-benar *bidirectional* karena token yang diprediksi dapat "melihat" konteks dari sisi kiri dan sisi kanan secara bersamaan [41]. *Loss function* untuk MLM adalah *categorical cross-entropy* antara distribusi probabilitas output model (dari *softmax* atas *vocabulary*) dan token asli [41]. Tugas kedua adalah *Next Sentence Prediction* (NSP) yang melatih model untuk memahami hubungan antar kalimat: model diberikan dua kalimat A dan B (dipisahkan oleh token [SEP]) dan diprediksi apakah kalimat B merupakan kelanjutan yang logis setelah kalimat A (*label IsNext*) atau bukan (*label NotNext*) [41]. NSP sangat berguna untuk tugas-tugas yang memerlukan pemahaman hubungan antar kalimat, seperti *question answering* dan *natural language inference* [41].

Setelah proses *pre-training* selesai yang memakan waktu berhari-hari hingga berminggu-minggu pada TPU/GPU cluster BERT dapat diadaptasi untuk berbagai tugas *downstream* (tugas hilir) melalui proses *fine-tuning* dengan menambahkan lapisan keluaran sederhana pada arsitektur yang sama tanpa mengubah parameter utama secara drastis [41].

Fine-tuning dilakukan dengan melatih seluruh parameter model (termasuk semua lapisan *encoder* dan *embedding*) pada dataset berlabel yang relatif kecil (ratusan hingga ribuan sampel) untuk tugas tertentu seperti klasifikasi sentimen, *named entity recognition* (NER), atau *question answering* [41]. Proses *fine-tuning* biasanya hanya memerlukan 2-4 epoch karena model *pre-training* sudah memiliki pengetahuan bahasa yang sangat kuat dari korpus berukuran 3.3 miliar kata [41].

Kemampuan *fine-tuning* untuk mencapai *state-of-the-art* dengan hanya ribuan sampel adalah keunggulan utama BERT dibandingkan model NLP sebelumnya yang memerlukan ratusan ribu hingga jutaan sampel berlabel [43]. Namun, *fine-tuning* pada dataset yang sangat kecil (ratusan sampel) berisiko *overfitting*, sehingga diperlukan strategi seperti *discriminative fine-tuning* (*learning rate* berbeda per *layer*), *gradual unfreezing*, dan regularisasi yang tepat [41]. BERT telah menjadi fondasi bagi berbagai model domain-spesifik, termasuk FinBERT untuk keuangan, BioBERT untuk biomedis, dan LegalBERT untuk hukum, dengan melakukan further *pre-training* atau *fine-tuning* pada korpus domain tertentu [43].

2.10 FinBERT

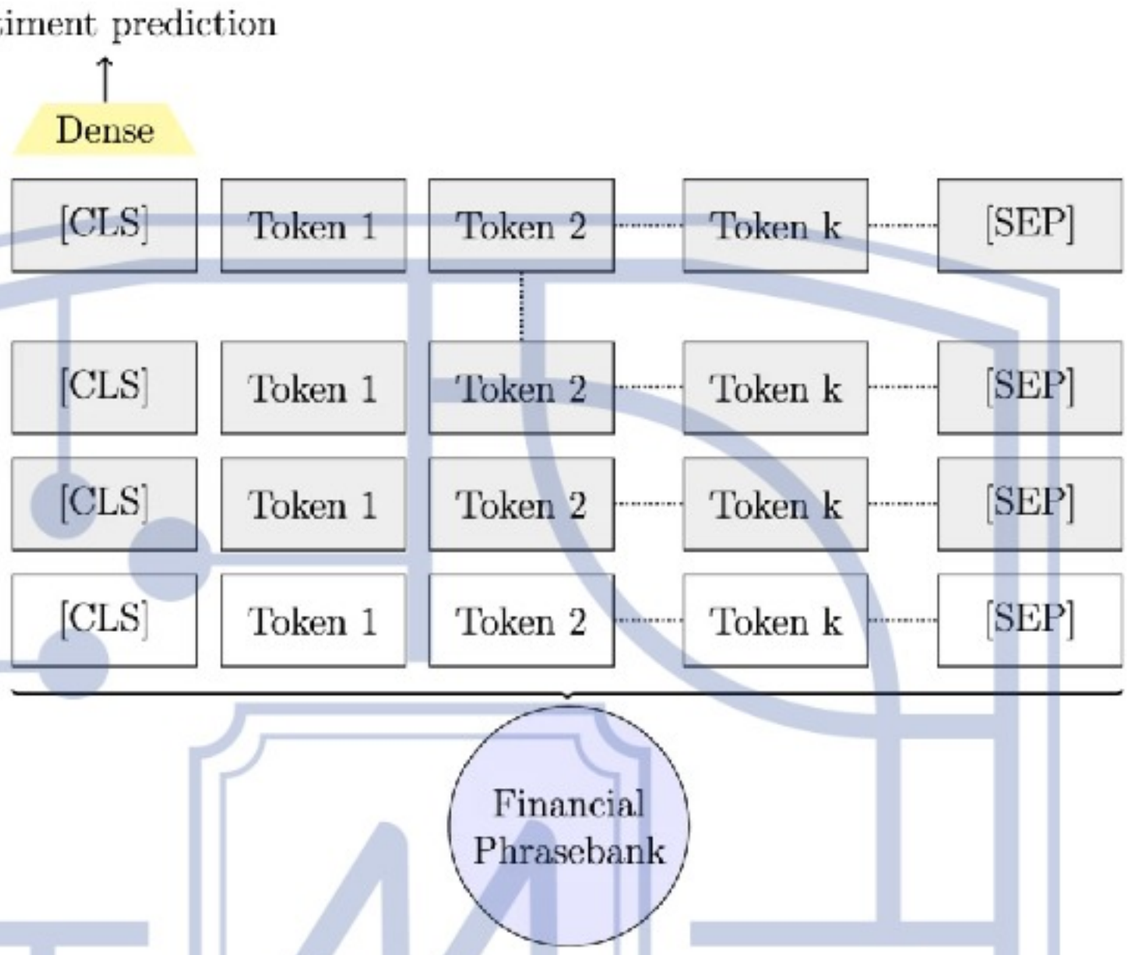
FinBERT adalah model *Natural Language Processing* (NLP) berbasis arsitektur Transformer yang dikembangkan secara khusus untuk domain keuangan, dan merupakan hasil *fine-tuning* lanjutan dari model BERT (*Bidirectional Encoder Representations from Transformers*) menggunakan korpus teks finansial yang besar [25]. FinBERT dirancang secara fundamental untuk memahami konteks, terminologi teknis, dan pola bahasa yang khas pada dokumen keuangan seperti berita pasar keuangan, laporan keuangan perusahaan, transkrip *earnings call*, analisis ekonomi oleh para ahli, dan dokumen regulasi [31]. Berbeda dengan model NLP umum yang dilatih pada korpus teks non-spesifik seperti Wikipedia dan BookCorpus (kumpulan buku), FinBERT mampu menangkap istilah-istilah teknis keuangan dengan tepat misalnya kata "*bullish*" (optimis terhadap pasar), "*bearish*" (pesimis), "*liquidity*" (likuiditas), "*volatility*" (volatilitas), "*price-to-earnings ratio*" (rasio harga terhadap laba), "*yield curve*" (kurva imbal hasil), dan "*quantitative easing*" dalam konteks yang benar, bukan sekedar kata biasa [39]. FinBERT dikembangkan dengan melakukan *further pre-training* (pelatihan lanjutan) pada BERT-base menggunakan korpus teks finansial TRC2-*financial* yang terdiri dari 46.143 dokumen berita Reuters dengan lebih dari 29 juta kata dari berbagai sektor keuangan [39]. Setelah *further pre-training* selesai, model dilanjutkan dengan *fine-tuning* untuk tugas klasifikasi sentimen keuangan menggunakan

dataset Financial PhraseBank yang berisi kalimat-kalimat dari berita keuangan Eropa yang telah dilabeli sentimen secara manual oleh para ahli finansial [39]. Hasilnya, FinBERT berhasil melampaui metode *state-of-the-art* sebelumnya (termasuk SVM dengan *feature engineering manual*) dengan peningkatan akurasi yang signifikan hingga 15% pada dataset Financial PhraseBank [39]. Model hasil penelitian Araci tersebut selanjutnya dipublikasikan secara terbuka di platform Hugging Face, sehingga dapat diakses dan digunakan kembali oleh peneliti lain (termasuk penelitian ini) tanpa perlu melatih dari awal [38].

Perkembangan model berbasis Transformer dalam bidang Natural Language Processing (NLP) tidak berhenti pada BERT saja. Sejumlah peneliti kemudian mengadaptasi BERT untuk domain-domain tertentu agar model lebih memahami terminologi dan konteks yang spesifik pada bidang tersebut, pendekatan ini dikenal sebagai domain-specific BERT [41]. FinBERT merupakan salah satu hasil dari pendekatan tersebut. Araci [39] mengembangkan FinBERT dengan mengambil BERT-base sebagai titik awal, kemudian melakukan further *pre-training* menggunakan korpus teks finansial TRC2-financial yang terdiri dari 46.143 dokumen berita Reuters dengan lebih dari 29 juta kata, lalu dilanjutkan dengan *fine-tuning* untuk tugas klasifikasi sentimen keuangan. Melalui pendekatan ini, FinBERT berhasil melampaui metode *state-of-the-art* sebelumnya dengan peningkatan akurasi hingga 15% pada dataset Financial PhraseBank [39]. Model hasil penelitian Araci tersebut selanjutnya dipublikasikan secara terbuka sehingga dapat diakses dan digunakan kembali oleh peneliti lain, termasuk dalam penelitian ini.

Arsitektur FinBERT pada dasarnya identik dengan BERT dalam hal jumlah lapisan *encoder* (12 layer untuk *model base*), *hidden size* (768), dan mekanisme *multi-head self-attention* (12 heads) [39]. Perbedaan utama antara FinBERT dan BERT standar terletak pada bobot (*weights*) model yang telah disesuaikan (*fine-tuned*) dengan domain keuangan [39]. Dengan kata lain, FinBERT menggunakan arsitektur yang sama dengan BERT (keduanya berbasis *Transformer encoder*), tetapi nilai parameter-parameter numeriknya (bobot dan bias) berbeda karena FinBERT telah melihat data keuangan dalam jumlah besar [39]. Proses klasifikasi sentimen dengan FinBERT dimulai dengan tokenisasi teks menggunakan WordPiece tokenizer (*subword tokenization*) yang memecah kata-kata langka menjadi *subword unit* untuk generalisasi yang lebih baik [39]. Setelah tokenisasi, teks diproses melalui lapisan *embedding*, kemudian melalui 12 lapisan *Transformer encoder* yang menggunakan mekanisme *multi-head self-attention* untuk menangkap hubungan antar token secara kontekstual [39]. Representasi dari token khusus [CLS] (yang berada di awal sekuens) setelah melewati seluruh 12 lapisan encoder diambil sebagai representasi agregat dari

keseluruhan kalimat, karena token [CLS] telah "memperhatikan" semua token lain dalam kalimat melalui mekanisme *self-attention* [39]. Representasi token [CLS] ini kemudian diteruskan ke lapisan klasifikasi (*classification layer*) berupa *dense layer* dengan fungsi aktivasi *softmax* untuk memprediksi probabilitas setiap kategori sentimen: positif, negatif, atau netral [39]. Arsitektur FinBERT ditunjukkan pada Gambar 2.6 [39].



Gambar 2.6 Arsitektur FinBERT [39]

Untuk keperluan analisis sentimen, proses dimulai dengan tokenisasi teks menggunakan *WordPiece tokenizer*. Pada salah satu varian FinBERT, dikembangkan pula kosakata finansial khusus bernama FinVocab menggunakan pustaka SentencePiece, sebagai alternatif dari kosakata bawaan BERT (BaseVocab) [44]. Sementara itu, pendekatan lain menggunakan korpus finansial TRC2-financial untuk melakukan further pre-training pada model BERT dasar sebelum proses fine-tuning [39].

Setelah melalui seluruh lapisan *encoder*, representasi dari token [CLS] digunakan sebagai representasi keseluruhan kalimat, lalu diteruskan ke lapisan klasifikasi berupa *dense layer* dengan fungsi aktivasi *softmax* untuk memprediksi kategori sentimen positif, negatif, atau netral [39], [44]. Araci juga menerapkan sejumlah strategi pelatihan tambahan guna mencegah *catastrophic forgetting*, yakni *slanted triangular learning rates*, *discriminative fine-tuning*, dan *gradual unfreezing*, yang secara keseluruhan terbukti menghasilkan performa klasifikasi terbaik [39].

Mencari Persamaan Klasifikasi Sentimen :

$$y = softmax(W_h h_{CLS} + b) \quad (2.15)$$

Dimana :

y : Output probabilitas klasifikasi sentimen

$softmax$: Fungsi probabilitas

W_h : Bobot matrix pada layer klasifikasi

h_{CLS} : Representasi token CLS

b : Bias

Selain arsitektur dasar FinBERT, terdapat varian FinBERT lain yang menggunakan kosakata finansial khusus bernama FinVocab yang dibangun dengan pustaka SentencePiece dari awal, sebagai alternatif dari kosakata bawaan BERT (BaseVocab) yang berbasis WordPiece [44]. Namun, versi Araci yang diakses melalui repositori "ProsusAI/finbert" pada platform Hugging Face lebih banyak digunakan oleh peneliti dan praktisi karena ketersediaannya (*ready-to-use*), kemudahan integrasi dengan *libraries* Python seperti Transformers, dan dokumentasi yang lengkap [45]. Araci juga menerapkan sejumlah strategi pelatihan tambahan untuk mencegah catastrophic forgetting fenomena di mana model deep learning melupakan pengetahuan yang sudah dipelajari pada *pre-training* ketika di-*fine-tune* pada dataset baru yang kecil [39]. Strategi-strategi tersebut meliputi: (1) *Slanted triangular learning rates* (STLR) yang memberikan *learning rate* lebih besar pada *epoch* awal untuk adaptasi cepat ke domain baru, kemudian *learning rate* menurun secara gradual; (2) *Discriminative fine-tuning* yang memberikan *learning rate* yang berbeda untuk setiap lapisan (lapisan yang lebih dalam/dekat output mendapat *learning rate* lebih besar, lapisan yang lebih dangkal/dekat input mendapat *learning rate* lebih kecil), karena lapisan bawah cenderung menangkap fitur yang lebih umum dan universal yang sebaiknya tidak terlalu banyak berubah; dan (3) *Gradual unfreezing* yang secara bertahap membuka kunci (*unfreeze*) lapisan-lapisan BERT dari yang paling atas hingga ke bawah selama proses *fine-tuning*, sehingga lapisan bawah tetap mempertahankan pengetahuan *pre-training* lebih lama dan terlindung dari perubahan drastis yang dapat menyebabkan *overfitting* [39]. Ketiga strategi ini terbukti menghasilkan performa klasifikasi terbaik pada dataset Financial PhraseBank, dengan peningkatan metrik evaluasi yang konsisten [39].

Dalam penelitian finansial dan kini juga di pasar cryptocurrency, FinBERT banyak digunakan untuk analisis sentimen berita pasar karena kemampuannya yang superior dalam mengklasifikasikan sentimen ke dalam kategori positif, negatif, dan netral secara kontekstual [46]. Beberapa studi menunjukkan bahwa integrasi hasil analisis sentimen berbasis FinBERT ke dalam model prediksi harga dapat memberikan informasi tambahan yang relevan dan signifikan informasi yang tidak sepenuhnya tercermin dalam data harga historis semata [40]. Pada konteks pasar cryptocurrency yang sangat dipengaruhi oleh berita dan

media sosial, FinBERT semakin banyak diterapkan untuk menganalisis sentimen berita kripto karena karakteristik bahasa yang dinamis, penuh dengan slang (misalnya "moon", "dump", "pump", "whale", "FOMO", "ATH"), dan sangat berbasis informasi daring yang berubah cepat [40]. Analisis sentimen berbasis FinBERT dinilai efektif dalam menangkap persepsi pasar terhadap peristiwa dan berita yang berkaitan dengan Bitcoin, sehingga relevan digunakan sebagai fitur tambahan dalam prediksi harga Bitcoin berbasis deep learning [40]. Meskipun FinBERT unggul dalam domain keuangan umum, terdapat keterbatasan yang perlu diakui dalam konteks cryptocurrency. FinBERT dilatih pada korpus berita keuangan konvensional yang tidak mencakup kosakata khas komunitas kripto seperti "moon", "FOMO", "dump", "whale", dan "ATH", sehingga berpotensi salah mengklasifikasikan sentimen pada berita yang sarat terminologi kripto informal. Keterbatasan ini diakui dalam penelitian ini sebagai salah satu faktor yang dapat memengaruhi kualitas skor sentimen yang dihasilkan, dan menjadi rekomendasi untuk penelitian selanjutnya agar mempertimbangkan model NLP yang dilatih secara khusus pada korpus cryptocurrency. Dalam penelitian ini, model FinBERT yang digunakan adalah versi "ProsusAI/finbert" dari platform Hugging Face, yang langsung digunakan tanpa proses fine-tuning ulang karena model ini sudah memiliki kemampuan klasifikasi sentimen yang memadai untuk domain keuangan termasuk berita cryptocurrency [39].

2.11 Integrasi Data Historis dan Sentimen Berita

Integrasi data historis harga dengan analisis sentimen berita merupakan pendekatan multimodal yang mengombinasikan informasi numerik (*time series* harga) dan informasi tekstual (sentimen berita) untuk meningkatkan kualitas prediksi *time series* finansial dibandingkan model yang hanya menggunakan satu sumber data [38]. Data historis harga merepresentasikan dinamika pasar yang telah terjadi berdasarkan data transaksi aktual, mencerminkan perilaku kolektif pelaku pasar melalui mekanisme permintaan dan penawaran [4]. Sementara itu, sentimen berita mencerminkan persepsi, ekspektasi, dan reaksi emosional pelaku pasar terhadap informasi eksternal (berita, regulasi, peristiwa global) yang berpotensi memengaruhi pergerakan harga aset di masa depan, seringkali sebelum perubahan tercermin dalam data harga [38]. Dengan kata lain, data harga bersifat *lagging* (tertinggal) karena mencerminkan apa yang sudah terjadi, sedangkan sentimen berita bersifat leading (memimpin) karena dapat menangkap sentimen kolektif yang akan mendorong pergerakan harga di menit atau jam berikutnya [40].

Dalam praktiknya, integrasi sentimen berita dilakukan dalam beberapa langkah sistematis [38]. Pertama, ekstraksi skor sentimen numerik dari teks berita menggunakan model *Natural Language Processing* (NLP) yang canggih seperti FinBERT (atau BERT untuk domain umum) [39]. Skor sentimen biasanya dinormalisasi ke dalam rentang [-1, 1] atau [0, 1] tergantung pada kebutuhan model *downstream* [28]. Kedua, skor sentimen tersebut digabungkan (*concatenate* atau ditambahkan sebagai fitur tambahan) dengan fitur numerik harga historis (*open, high, low, close, volume*) yang sudah dinormalisasi [40]. Ketiga, dataset gabungan (harga + sentimen) digunakan sebagai input ke model *deep learning*, di mana model mempelajari interaksi kompleks antara kedua jenis fitur secara simultan dan otomatis [40]. Pendekatan multimodal ini memungkinkan model *deep learning* memanfaatkan konteks informasi yang kaya dan beragam yang tidak sepenuhnya tercermin dalam data harga semata, khususnya pada pasar yang sangat sensitif terhadap berita dan informasi publik seperti *cryptocurrency* [40].

Terdapat beberapa strategi integrasi data yang umum digunakan dalam literatur, yaitu *early fusion, late fusion, dan hybrid fusion* [38]. *Early fusion* (fusi awal) menggabungkan fitur dari berbagai sumber pada tingkat input sebelum dimasukkan ke model, sehingga model dapat mempelajari interaksi antar modalitas secara *end-to-end* dari awal hingga akhir [40]. Pendekatan ini sederhana, efisien secara komputasi, dan telah terbukti efektif dalam banyak studi prediksi harga saham dan *cryptocurrency* [40]. *Late fusion* (fusi akhir) melatih model terpisah untuk setiap modalitas secara independen, kemudian menggabungkan prediksi akhir dari masing-masing model (misalnya melalui rata-rata tertimbang atau voting) [38]. *Late fusion* lebih fleksibel (setiap modalitas dapat menggunakan arsitektur model yang berbeda) tetapi tidak dapat menangkap interaksi antar modalitas selama pemrosesan internal [38]. *Hybrid fusion* menggabungkan kedua pendekatan, misalnya dengan mengekstraksi fitur dari setiap modalitas secara terpisah pada lapisan awal, kemudian menggabungkannya pada lapisan yang lebih dalam [38]. Dalam penelitian ini, digunakan *early fusion* karena paling sederhana, paling hemat komputasi, dan telah terbukti efektif dalam berbagai studi prediksi harga saham dan *cryptocurrency* [40]. Skor sentimen hasil ekstraksi FinBERT yang telah dirata-rata per hari digabungkan dengan data harga (*close price*) yang telah dinormalisasi, lalu dibentuk menjadi *sequence* (*window*) dengan *sliding window* [28]. Input model menjadi matriks 3D dengan dimensi (*batch_size, window_size*=60, jumlah_fitur), di mana jumlah_fitur adalah 1 (model *baseline* hanya harga) atau 2 (model dengan integrasi sentimen: harga + skor sentimen) [28]. Model LSTM atau GRU kemudian memproses kedua fitur secara simultan pada setiap timestep, mempelajari pola interaksi temporal yang

kompleks antara pergerakan harga masa lalu dan sentimen pasar yang terjadi pada periode yang sama [40]. Pendekatan early fusion dipilih dalam penelitian ini karena paling sederhana, paling hemat komputasi, dan telah terbukti efektif dalam berbagai studi prediksi harga cryptocurrency yang menggabungkan data harga dengan fitur sentimen [40]. Zhao et al. (2022) menunjukkan bahwa penambahan fitur sentimen pada model prediksi cryptocurrency berbasis deep learning secara konsisten memberikan peningkatan akurasi dibandingkan model yang hanya menggunakan data harga historis [40].

2.12 Optimasi *Hyperparameter* Menggunakan Optuna

Optimasi *hyperparameter* merupakan tahapan krusial dalam pengembangan model *machine learning* dan *deep learning* karena parameter ini ditentukan sebelum proses pelatihan dimulai dan tidak dipelajari secara otomatis oleh model [31]. Dalam penelitian ini, terdapat empat *hyperparameter* utama yang dioptimasi, yaitu jumlah unit tersembunyi, *learning rate, dropout rate*, dan jumlah layer. Keempat parameter ini sangat menentukan kemampuan model dalam melakukan generalisasi terhadap data baru.

1. Jumlah Unit Tersembunyi (*Hidden Units*)

Hidden units menentukan kapasitas representasi model dalam menangkap pola dari data sekuensial. Semakin banyak unit tersembunyi, semakin besar kemampuan model untuk memodelkan hubungan yang kompleks dan non-linear. Namun, jumlah yang terlalu besar berisiko menyebabkan *overfitting*, sementara jumlah yang terlalu kecil dapat mengakibatkan *underfitting* sehingga model gagal menangkap pola penting dalam data [28].

2. Laju Pembelajaran (*Learning Rate*)

Learning rate merupakan parameter yang mengendalikan besaran pembaruan bobot (*weight*) pada setiap iterasi pelatihan melalui algoritma optimasi seperti Adam maupun SGD. Apabila nilai *learning rate* ditetapkan terlalu tinggi, proses pelatihan berpotensi tidak konvergen atau mengalami osilasi yang tidak stabil. Sebaliknya, nilai yang terlalu rendah akan memperlambat laju konvergensi secara signifikan dan meningkatkan risiko model terjebak pada *local minimum*, sehingga solusi optimal sulit untuk dicapai [28].

3. Tingkat Dropout (*Dropout Rate*)

Dropout merupakan teknik regularisasi yang bekerja dengan cara menonaktifkan sejumlah neuron secara acak selama proses pelatihan berdasarkan suatu nilai probabilitas tertentu. Mekanisme ini bertujuan untuk mencegah ketergantungan yang

berlebihan antarneuron, sehingga secara efektif dapat menekan terjadinya *overfitting*. Penentuan *dropout rate* yang tepat sangat penting agar model tidak kehilangan terlalu banyak informasi selama pelatihan, sekaligus tetap memiliki kemampuan generalisasi yang memadai terhadap data yang belum pernah dilihat [28].

4. **Jumlah Lapisan (*Stacked Layers*)**

Kedalaman arsitektur model ditentukan oleh jumlah lapisan LSTM atau GRU yang disusun secara bertumpuk (*stacked*). Konfigurasi ini memengaruhi kemampuan model dalam mengekstraksi fitur hierarkis dari data sekuensial secara bertahap. Semakin banyak lapisan yang digunakan, semakin tinggi pula kemampuan model untuk mempelajari representasi abstrak pada tingkat yang lebih kompleks. Namun demikian, penambahan lapisan yang berlebihan dapat memperburuk masalah *vanishing gradient* serta memperpanjang durasi pelatihan secara keseluruhan [28].

5. **Ukuran Batch (*Batch Size*)**

Batch size menentukan jumlah sampel data yang diproses oleh jaringan sebelum bobot internal model diperbarui pada setiap *epoch*. Pengaturan ukuran *batch* memengaruhi stabilitas estimasi gradien dan efisiensi penggunaan memori perangkat keras selama pelatihan. Ukuran *batch* yang kecil memberikan estimasi gradien yang lebih fluktuatif (*noisy*), yang secara tidak langsung dapat membantu model keluar dari *local minimum*, namun memperpanjang total durasi komputasi. Sebaliknya, ukuran *batch* yang terlalu besar dapat mempercepat proses pelatihan per *epoch*, tetapi berisiko menurunkan kemampuan generalisasi model karena pembaruan gradien cenderung terlalu halus (*smooth*) dan dapat menyebabkan model terjebak pada solusi suboptimal [28].

6. **Tingkat Regulasi L2 (L2 *rate*)**

Tingkat regulasi L2 (L2 *rate*) atau *weight decay* merupakan *hyperparameter* regularisasi yang bekerja dengan menambahkan penalti berupa kuadrat dari nilai bobot (L2 *norm*) ke dalam fungsi kerugian (*loss function*). Pendekatan ini bertujuan untuk membatasi pertumbuhan bobot jaringan selama proses pelatihan sehingga parameter model tidak menjadi terlalu besar. Dengan menjaga bobot tetap kecil, kompleksitas model dapat dikurangi dan kemampuan generalisasi terhadap data yang belum pernah dilihat sebelumnya menjadi lebih baik, sehingga risiko *overfitting* dapat diminimalkan. Sebaliknya, nilai *weight decay* yang terlalu besar dapat menyebabkan model mengalami *underfitting* karena kapasitas pembelajaran menjadi terlalu terbatas, sedangkan nilai yang terlalu kecil atau mendekati nol akan mengurangi efek regularisasi sehingga meningkatkan potensi *overfitting*. Oleh karena itu, pemilihan

nilai *weight decay* yang optimal menjadi salah satu *hyperparameter* penting yang perlu dioptimasi, misalnya menggunakan metode *hyperparameter optimization* seperti Optuna[47].

Penelitian ini memanfaatkan Optuna, sebuah kerangka kerja (*framework*) optimasi *hyperparameter* generasi terbaru [31]. Optuna dirancang dengan prinsip *define-by-run*, yang memungkinkan pengguna untuk membangun ruang pencarian parameter secara dinamis selama masa transisi (*runtime*). Keunggulan utama Optuna dibandingkan metode konvensional seperti *Grid Search* atau *Random Search* terletak pada efisiensi strateginya yang menggabungkan pencarian berbasis *Bayesian Optimization,* khususnya algoritma *Tree-structured Parzen Estimator* (TPE) yang membangun model probabilistik untuk memprediksi kombinasi *hyperparameter* paling menjanjikan. Selain itu, Optuna dilengkapi dengan mekanisme *pruning* (penghentian dini) yang secara otomatis menghentikan percobaan dengan performa yang kurang menjanjikan, sehingga menghemat sumber daya komputasi secara signifikan [31]. Hal ini sangat efektif untuk mengoptimalkan penggunaan sumber daya komputasi dan mempercepat waktu eksekusi, terutama dalam melatih model kompleks seperti LSTM dan GRU pada lingkungan terbatas. Dengan integrasi Optuna, penelitian ini bertujuan untuk menemukan kombinasi *hyperparameter* paling optimal guna meminimalkan galat prediksi pada peramalan harga Bitcoin jangka panjang.

2.12.1 Fungsi Kerugian (Loss Function) pada Pelatihan Model

Fungsi kerugian (loss function) merupakan komponen fundamental dalam proses pelatihan model deep learning yang mengukur seberapa besar selisih antara nilai prediksi model dengan nilai aktual pada setiap iterasi pelatihan [20]. Nilai fungsi kerugian inilah yang menjadi sinyal bagi algoritma optimasi seperti Adam untuk memperbarui bobot-bobot model melalui proses backpropagation, dengan tujuan agar selisih tersebut semakin mengecil seiring berjalannya pelatihan [28].

Pada permasalahan regresi time series finansial, terdapat beberapa pilihan fungsi kerugian yang umum digunakan. Mean Squared Error (MSE) merupakan yang paling umum, dengan cara mengkuadratkan seluruh selisih prediksi sehingga kesalahan besar mendapat penalti yang jauh lebih besar dibandingkan kesalahan kecil [20]. Namun, sifat ini menjadikan MSE sangat sensitif terhadap nilai pencilan (outlier) yang pada data harga cryptocurrency sangat sering terjadi akibat lonjakan harga ekstrem dalam waktu singkat sehingga model dapat terdistorsi untuk meminimalkan kesalahan pada outlier dengan

mengorbankan akurasi pada mayoritas data [28]. Mean Absolute Error (MAE) di sisi lain memperlakukan semua kesalahan secara linear tanpa pengkuadratan, sehingga lebih tahan terhadap outlier, tetapi gradiennya konstan dan tidak informatif di sekitar nilai nol, sehingga konvergensi pelatihan dapat menjadi tidak stabil [20].

Huber loss merupakan fungsi kerugian yang menggabungkan keunggulan MSE dan MAE secara adaptif [28],[48]. Untuk kesalahan yang kecil (di bawah ambang batas δ), Huber loss berperilaku seperti MSE sehingga gradiennya informatif dan konvergensi lancar. Untuk kesalahan yang besar (di atas δ), Huber loss berperilaku seperti MAE (linear) sehingga tidak memberikan penalti berlebihan terhadap outlier [20]. Secara matematis, Huber loss dirumuskan sebagai berikut [48]:

$$L_\delta(y,\hat{y}) = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & jika\ |y-\hat{y}| \le \delta \\ \delta \cdot |y-\hat{y}| - \frac{1}{2}\delta^2 & |y-\hat{y}| > \delta \end{cases} \quad \text{.....(2.16)}$$

Dimana :

y : nilai aktual

$\hat{y}$: nilai prediksi

δ : parameter ambang batas yang menentukan titik peralihan antara perilaku kuadratik dan linear

Dalam penelitian ini, Huber loss dipilih sebagai fungsi kerugian untuk seluruh dua belas model karena karakteristik harga Bitcoin yang memiliki volatilitas ekstrem dan sering mengalami lonjakan harga dalam waktu singkat (spike), sehingga fungsi kerugian yang tahan terhadap outlier lebih sesuai dibandingkan MSE murni [20],[28].

2.12.2 Mekanisme Callback pada Pelatihan Model

Callback adalah mekanisme pada *framework deep learning* yang memungkinkan eksekusi fungsi-fungsi tertentu secara otomatis pada titik-titik tertentu selama proses pelatihan, seperti di akhir setiap *epoch* [49]. Penggunaan *callback* yang tepat sangat penting untuk mencegah *overfitting*, menghemat sumber daya komputasi, dan memastikan model yang dihasilkan adalah model dengan performa terbaik pada data validasi, bukan model pada *epoch* terakhir [28].

EarlyStopping adalah *callback* yang menghentikan proses pelatihan secara otomatis apabila metrik yang dipantau umumnya *validation loss* tidak mengalami perbaikan selama sejumlah *epoch* berturut-turut yang ditentukan oleh parameter *patience* [49]. Tanpa *EarlyStopping*, model akan terus dilatih hingga batas *epoch* maksimum meskipun performa pada data validasi sudah mulai memburuk (*overfitting*), yang hanya membuang waktu

komputasi dan menghasilkan model yang lebih buruk pada data baru [28]. Dalam penelitian ini, *EarlyStopping* dikonfigurasi dengan *patience* sebesar 15 *epoch* terhadap *validation loss*, yang berarti pelatihan dihentikan apabila *validation loss* tidak membaik selama 15 *epoch* berturut-turut. Nilai *patience* 15 dipilih sebagai keseimbangan antara memberikan model kesempatan yang cukup untuk keluar dari *local minimum* sementara, tanpa membuang waktu komputasi yang berlebihan [28].

ReduceLROnPlateau adalah *callback* yang secara otomatis menurunkan *learning rate* apabila metrik yang dipantau tidak menunjukkan perbaikan selama sejumlah *epoch* tertentu [49]. Penurunan *learning rate* secara adaptif ini berguna ketika model sudah mendekati solusi optimal tetapi langkah pembaruan bobot masih terlalu besar, sehingga model berosilasi di sekitar minimum tanpa bisa konvergen lebih dalam [28]. Dalam penelitian ini, ReduceLROnPlateau dikonfigurasi dengan faktor pengurangan 0,5 (artinya learning rate dikurangi menjadi setengahnya), patience 7 epoch, dan batas minimum learning rate sebesar $1\times1\ \ 0^{-6}$. Kombinasi *EarlyStopping* dan *ReduceLROnPlateau* ini memastikan bahwa pelatihan berjalan secara efisien: *ReduceLROnPlateau* memperbaiki konvergensi dengan menyesuaikan langkah pembaruan, sementara *EarlyStopping* menghentikan pelatihan tepat waktu sebelum model mengalami *overfitting* yang berlebihan [28], [49].

2.12.3 Mekanisme Callback pada Pelatihan Model

Selain *dropout* dan regularisasi L2, terdapat dua teknik lain yang digunakan dalam penelitian ini untuk menjaga kestabilan proses pelatihan pada arsitektur LSTM dan GRU, yaitu *Batch Normalization* dan *Gradient Clipping*.

Batch Normalization bekerja dengan menormalisasi keluaran setiap lapisan pada tiap *batch* data selama pelatihan, sehingga distribusi inputnya tetap stabil dari satu iterasi ke iterasi berikutnya [50]. Tanpa adanya normalisasi ini, perubahan bobot di lapisan awal dapat menggeser distribusi input yang diterima lapisan berikutnya secara terus-menerus sepanjang pelatihan. Fenomena ini dikenal sebagai *internal covariate shift,* yang memperlambat laju konvergensi dan membuat model lebih sulit dilatih. Dengan menstabilkan distribusi tersebut, pelatihan dapat berjalan lebih lancar dan lebih stabil, terutama pada data yang memiliki variasi besar antar-*batch* seperti harga Bitcoin yang bersifat sangat volatil [50]. Dalam penelitian ini, *Batch Normalization* diterapkan pada setiap blok LSTM maupun GRU sebagai lapisan tambahan setelah keluaran rekuren, baik pada arsitektur *single-layer* maupun *stacked*.

Gradient Clipping menangani permasalahan yang berbeda namun sama pentingnya. Pada jaringan rekuren seperti LSTM dan GRU, proses *backpropagation* dilakukan melewati banyak langkah waktu (*timestep*) secara berurutan, dan perkalian gradien yang berulang pada setiap langkah berpotensi membuat nilainya membesar secara tidak terkendali — dikenal sebagai *exploding gradient* [51]. Ketika ini terjadi, pembaruan bobot menjadi sangat besar dan tidak stabil, sehingga pelatihan bisa gagal konvergen atau bahkan menghasilkan nilai *NaN*. *Gradient Clipping* mengatasi ini dengan membatasi besaran gradien agar tidak melampaui suatu ambang batas tertentu sebelum digunakan untuk memperbarui bobot, tanpa mengubah arah gradien secara keseluruhan [51]. Dalam penelitian ini, *Gradient Clipping* diterapkan melalui parameter *clipvalue* pada *optimizer* Adam dengan nilai 0,5, yang berarti setiap komponen gradien dipotong ke rentang [−0,5, 0,5] apabila nilainya melampaui batas tersebut.

2.12.4 Hyperband Pruner

Selain mekanisme *pruning* dasar seperti *Median Pruner*, Optuna juga menyediakan *Hyperband Pruner* sebagai strategi *pruning* yang lebih adaptif dalam mengalokasikan sumber daya komputasi antar *trial* [31]. *Hyperband Pruner* bekerja dengan menjalankan beberapa instansi *Successive Halving Pruner* secara paralel, yang masing-masing disebut sebagai *bracket*, dengan konfigurasi jumlah *trial* awal (n) dan alokasi sumber daya per *trial* (B/n) yang berbeda-beda [31]. Pendekatan multi-*bracket* ini bertujuan mengatasi *trade-off* antara banyaknya *trial* yang dijalankan dan besar sumber daya (misalnya jumlah *epoch*) yang dialokasikan untuk setiap *trial*, sehingga peneliti tidak perlu menentukan secara manual keseimbangan optimal antara kedua nilai tersebut [31].

Pada setiap *bracket*, *trial* dengan performa terbaik pada *checkpoint* tertentu dipertahankan dan dilanjutkan ke tahap alokasi sumber daya yang lebih besar, sedangkan *trial* dengan performa buruk dihentikan lebih awal (di-*prune*), mengikuti prinsip *successive halving* dengan faktor reduksi (*reduction_factor*) tertentu. Mekanisme ini dikendalikan oleh tiga parameter utama, yaitu *min_resource* (sumber daya minimum per *trial*), *max_resource* (sumber daya maksimum, umumnya disetel setara jumlah *epoch* maksimum pelatihan), dan *reduction_factor* (rasio pemangkasan *trial* pada setiap tahap) [31]. Dengan menjalankan banyak *bracket* sekaligus, *Hyperband Pruner* dapat secara otomatis mengeksplorasi berbagai strategi alokasi sumber daya tanpa memerlukan penyetelan manual, sehingga lebih *robust* dibandingkan *Successive Halving Pruner* tunggal apabila kombinasi n dan B/n yang optimal tidak diketahui sejak awal.

Dalam praktiknya, *Hyperband Pruner* umumnya dikombinasikan dengan *TPE Sampler*, karena kombinasi keduanya terbukti kompetitif dibandingkan pendekatan pencarian acak (*random search*) maupun *grid search* konvensional [31]. Namun demikian, karena *TPE Sampler* memerlukan sejumlah *trial* awal (secara baku 10 *trial* per *bracket*) untuk membangun model probabilistiknya, penggunaan *Hyperband Pruner* dengan banyak *bracket* menuntut jumlah total *trial* yang cukup besar agar keunggulan strategi penyetelannya dapat dimanfaatkan secara optimal [31].

2.13 Evaluasi Model

Evaluasi model merupakan tahap krusial dalam penelitian prediksi harga aset finansial untuk mengukur secara objektif seberapa akurat model dalam memperkirakan nilai di masa mendatang, dengan metrik yang tepat dan dapat dibandingkan antar penelitian [29]. Berbagai penelitian di bidang prediksi pasar keuangan berbasis *machine learning* dan *deep learning* umumnya menerapkan beberapa metrik evaluasi secara bersamaan, karena data finansial bersifat volatil (perubahannya cepat dan ekstrem) dan non-linear (tidak mengikuti pola linear sederhana), sehingga tidak cukup hanya mengandalkan satu metrik untuk menilai performa secara menyeluruh [52]. Keempat metrik yang digunakan secara konsisten dalam penelitian ini adalah *R-squared* (R^2), *Root Mean Squared Error* (RMSE), *Mean Absolute Percentage Error* (MAPE), dan *Directional Accuracy* (DA) [29], [52]. Masing-masing metrik memberikan informasi yang berbeda dan saling melengkapi: MAPE mengukur kesalahan relatif dalam bentuk persentase (mudah diinterpretasi lintas aset dengan harga berbeda), RMSE memberikan penalti yang lebih besar terhadap *outlier* (kesalahan besar yang sangat berbahaya dalam perdagangan), DA mengukur kemampuan model memprediksi arah pergerakan harga (naik atau turun) yang seringkali lebih berharga daripada akurasi nilai absolut bagi trader, dan R^2 menjelaskan seberapa baik model mampu menjelaskan varians dalam data aktual (proporsi variasi yang dapat dijelaskan oleh model) [29], [52].

MAPE (*Mean Absolute Percentage Error*) mengukur rata-rata kesalahan prediksi dalam bentuk persentase terhadap nilai aktual, sehingga bersifat *scale-invariant* dan dapat membandingkan kesalahan prediksi antar aset dengan tingkat harga yang berbeda-beda secara langsung, tanpa perlu normalisasi tambahan [29]. Al Ridhawi et al. (2026) menerapkan MAPE sebagai metrik utama dalam evaluasi model *Node Transformer* berbasis BERT untuk prediksi saham S&P 500, di mana model yang diusulkan mencapai MAPE sebesar 0.80% untuk prediksi satu hari ke depan, melampaui ARIMA (1.20%) dan LSTM (1.00%) [29]. Secara matematis, MAPE dirumuskan sebagai berikut [29], [52]:

Mencari Mean Absolute Percentage Error (MAPE) :

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| \times 100\% \quad (2.17)$$

Dimana :

Y_i : nilai actual

$\hat{Y}_i$: nilai prediksi

n : jumlah data

Interpretasi praktis dari MAPE dalam konteks finansial adalah: nilai MAPE < 10% menunjukkan model sangat akurat, 10-20% menunjukkan akurasi baik, 20-50% menunjukkan akurasi wajar (masih berguna untuk analisis tren), dan >50% menunjukkan model tidak akurat dan tidak dapat diandalkan untuk pengambilan keputusan investasi [29].

RMSE (Root Mean Square Error) mengukur tingkat kesalahan prediksi dengan memberikan penalti yang lebih besar (*quadratic penalty*) terhadap kesalahan yang besar (*outlier*), karena kesalahan dikuadratkan sebelum diakarkan [52]. RMSE sangat sensitif terhadap outlier lonjakan harga ekstrem akibat berita besar yang sering terjadi pada data finansial volatil seperti *cryptocurrency*, sehingga metrik ini sangat berguna untuk menilai stabilitas model dalam kondisi pasar ekstrem [52]. Oak et al. (2024) menggunakan RMSE bersama *Mean Absolute Error* (MAE) dalam mengevaluasi model *Bidirectional Multivariate* LSTM untuk prediksi harga saham India, di mana model yang diusulkan memperoleh rata-rata RMSE sebesar 0.0103955 yang sangat kecil (karena data telah dinormalisasi) [52]. RMSE dirumuskan sebagai berikut [29], [52]:

Mencari *Root Mean Square Error* (RMSE) :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \quad (2.18)$$

Dimana :

Y_i : nilai actual

$\hat{Y}_i$: nilai prediksi

n : jumlah data

RMSE memiliki satuan yang sama dengan data asli (dalam USD untuk harga Bitcoin), sehingga interpretasinya bergantung pada skala harga [29]. Untuk Bitcoin yang harganya dapat mencapai puluhan ribu dolar, RMSE beberapa ribu dolar mungkin masih dianggap wajar, sementara untuk saham berharga rendah (puluhan dolar), RMSE yang sama akan sangat besar secara relatif [29].

Directional Accuracy (DA) mengukur proporsi prediksi yang berhasil menentukan apakah harga akan naik (*increase*) atau turun (*decrease*) secara benar, terlepas dari seberapa besar pergerakannya [29]. DA seringkali lebih bernilai untuk strategi trading jangka pendek daripada meminimalkan kesalahan absolut harga, karena banyak strategi investasi (seperti momentum *trading, mean reversion, breakout trading*) hanya memerlukan sinyal arah (*long/short*) yang benar, bukan target harga yang presisi [29]. Al Ridhawi et al. (2026) menegaskan bahwa untuk aplikasi perdagangan (*trading*) di pasar finansial, memprediksi arah pergerakan harga dengan tepat (*directional accuracy*) kerap kali lebih bernilai ekonomis daripada meminimalkan kesalahan absolut harga, mengingat banyak strategi bergantung pada sinyal arah saja daripada target harga yang sangat presisi [29]. Model *Node Transformer* yang mereka usulkan berhasil mencapai *directional accuracy* sebesar 65% untuk prediksi satu hari ke depan, melampaui seluruh model pembanding yang berkisar 50-55% (hampir setara tebakan acak) [29]. DA dirumuskan sebagai berikut [29]:

Mencari *Directional Accuracy* (DA) :

$$DA = \frac{1}{n}\sum_{i=1}^{n} 1\left[\, sign(Y_i - Y_{i-1}) = sign(\hat{Y}_i - Y_{i-1})\right] \times 100\% \quad \text{.....(2.19)}$$

Dimana :

Y_i : nilai aktual pada waktu ke-i

$\hat{Y}_i$: nilai prediksi pada waktu ke-i

Y_{i-1} : nilai aktual pada waktu sebelumnya

n : jumlah data

$1[\cdot]$: fungsi indikator, bernilai 1 jika arah prediksi benar dan 0 jika sebaliknya

Al Ridhawi et al. menegaskan bahwa untuk aplikasi perdagangan, memprediksi arah pergerakan harga dengan tepat kerap kali lebih bernilai daripada meminimalkan kesalahan absolut harga, mengingat banyak strategi bergantung pada sinyal arah daripada target harga yang presisi [29]. Model *Node Transformer* yang mereka usulkan berhasil mencapai *directional accuracy* sebesar 65% untuk prediksi satu hari ke depan, melampaui seluruh model pembanding [29].

R^2 (*Coefficient of Determination*) menilai sejauh mana model mampu menjelaskan variasi dalam data aktual (proporsi *variance* yang dijelaskan oleh model), dengan nilai 1 menunjukkan model sempurna (semua variasi data dapat dijelaskan), nilai 0 menunjukkan model tidak lebih baik daripada hanya menggunakan rata-rata data (baseline paling sederhana), dan nilai negatif menunjukkan model lebih buruk daripada *baseline* rata-rata

[52]. Oak et al. (2024) menjadikan R^2 sebagai salah satu tolak ukur utama dalam perbandingan model, di mana model *Bidirectional Multivariate* LSTM yang diusulkan mencapai rata-rata R^2 sebesar 99.4779% pada empat saham yang diuji, yang berarti model tersebut dapat menjelaskan 99.48% variasi harga saham, hanya 0.52% variasi yang tidak dapat dijelaskan oleh model [52]. R^2 dirumuskan sebagai berikut [52]:

Mencari *Coefficient of Determination* (R^2)

$$R^2 = 1 - \frac{SS_R}{SS_T} \quad \text{.....(2.20)}$$

Dimana :

SS_R : *Sum of Squared Residuals*, yaitu jumlah kuadrat selisih antara nilai aktual dan nilai prediksi.

SS_T : *Total Sum of Squares*, yaitu jumlah kuadrat selisih antara nilai aktual dan rata-rata nilai aktual.

Satuan R^2 tidak memiliki dimensi (*unitless*), berkisar antara negatif tak terhingga hingga 1, dan semakin mendekati 1 semakin baik model dalam menjelaskan data [52]. Namun, untuk data finansial yang sangat *noise*, R^2 di atas 0.90 (90%) sudah dianggap sangat baik [52]. Dengan menggabungkan keempat metrik secara bersamaan dalam evaluasi, kita dapat melakukan penilaian yang komprehensif dan holistik terhadap performa model [29]. MAPE dan RMSE memberikan informasi tentang akurasi prediksi nilai absolut; DA memberikan informasi tentang kemampuan model dalam memprediksi arah pergerakan (yang seringkali lebih relevan untuk trading jangka pendek); dan R^2 memberikan informasi tentang seberapa baik model menjelaskan *varians* data secara keseluruhan (yang relevan untuk analisis fundamental) [29], [52]. Dalam penelitian ini, metrik evaluasi dihitung pada data uji (*hold-out test set*) sebesar 20% dari total data (646 sampel terakhir), dan performa dari keduabelas skenario eksperimen model (*baseline* LSTM/GRU, model dengan sentimen, *stacked models*, dan *optimized models*) dibandingkan secara sistematis dan disajikan dalam bentuk tabel dan grafik untuk mengidentifikasi model dengan performa terbaik [29], [52]. Model terbaik dipilih berdasarkan prioritas pada MAPE terendah (untuk investor institusi yang membutuhkan akurasi nilai absolut) dan DA tertinggi (untuk trader ritel yang membutuhkan akurasi arah), dengan mempertimbangkan juga RMSE untuk stabilitas terhadap *outlier* [29].

2.14 Penelitian Terdahulu

Tabel berikut merangkum penelitian terdahulu yang relevan beserta posisi penelitian ini sebagai research gap:

No	Peneliti & Tahun	Model	Sumber Sentimen	Aset	Horizon	Keterbatasan
1	Hamayel & Owda [6]	GRU, LSTM, Bi-LSTM	Tidak ada	BTC, LTC, ETH	Harian	Tidak mengintegrasikan sentimen; merekomendasikan eksplorasi faktor eksternal
2	Andromeda et al. [7]	LSTM, GRU	Tidak ada	Cryptocurrency	Harian	Tidak mengintegrasikan sentimen, hanya perbandingan arsitektur dasar
3	Arslan [9]	LSTM + EMD	Sentimen umum	BTC	Harian	Tidak menggunakan FinBERT, sentimen tidak terstruktur
4	Jaquart et al. [10]	LSTM, GRU	Twitter	BTC	1–60 menit	Horizon sangat pendek, sentimen dari sosial media
5	Otabek & Choi [11]	Review (LSTM, GRU, dll.)	Twitter, Reddit	BTC, ETH, LTC	Jangka pendek	Tidak ada studi sentimen berita finansial untuk jangka panjang
6	Hossain et al. [12]	FinBERT + Bi-LSTM	Sentimen pasar (FinBERT)	BTC, ETH	Harian	Tidak mengevaluasi jangka panjang, tidak ada optimasi hyperparameter, tidak membandingkan multi-skenario
7	Zhao et al. [35]	Transformer, LSTM	Twitter	BTC, ETH	Harian	Sentimen dari Twitter bukan berita finansial

8	Farrugia & Deguara [1]	Korelasi statistik	Berita & sosial media	Cryptocurrency	Harian	Hanya analisis korelasi, tidak membangun model prediksi
9	Gu & Yan [41]	FinBERT + LSTM	Berita finansial (Benzinga)	Saham NASDAQ-100	Harian	Domain saham bukan cryptocurrency, tidak ada optimasi hyperparameter
10	**Penelitian ini**	**LSTM, GRU, Stacked + Optuna**	**Berita finansial (FinBERT)**	**BTC**	**Jangka panjang (minggu–bulan)**	**—**