BAB II
KAJIAN LITERATUR

2.1 Tinjauan Pustaka

Bagian ini berisi landasan terkait teori-teori yang digunakan dan pekerjaan yang sudah dilakukan oleh penelitian sebelumnya untuk mendukung penyelesaian penelitian yang akan dilakukan.

2.1.1 Prediksi Harga Cryptocurrency Bitcoin

Perdagangan *cryptocurrency* telah mendapatkan pengakuan luas sebagai salah satu pilihan investasi paling populer dan menjanjikan bagi para investor [17]. *Cryptocurrency* pertama yang ada di dunia adalah Bitcoin, yang mulai diperdagangkan pada Januari 2009. Dalam beberapa tahun terakhir, Bitcoin telah mengalami kenaikan dan menjadi investasi menarik bagi para trader. Berbeda dengan saham atau valuta asing, harga Bitcoin bersifat fluktuatif, terutama karena perdagangan berlangsung 24 jam sehari tanpa penutupan. Untuk mengurangi risiko yang terlibat dan memaksimalkan keuntungan modal, para trader dan investor membutuhkan cara yang dapat memprediksi tren harga Bitcoin dengan akurat. Namun, banyak penelitian sebelumnya mengenai prediksi harga *cryptocurrency* lebih fokus pada jangka pendek, memiliki tingkat akurasi yang rendah, dan belum divalidasi secara menyeluruh [18]. Prediksi harga *cryptocurrency* sering dianggap sebagai salah satu tantangan terbesar dalam memprediksi *time series* data, penyebab utamanya adalah volatilitas harga yang signifikan dan jumlah elemen yang sulit diprediksi yang terlibat [17].

Prediksi Bitcoin oleh AI terbagi menjadi dua kategori. Kategori pertama adalah klasifikasi memprediksi naik atau turunnya Bitcoin, dengan standar kesalahan *Decision Accuracy* dan *F1-Score*. Kategori kedua adalah regresi memprediksi harga Bitcoin, dengan kesalahan *Root Mean Square Error* dan *Mean Absolute Percentage Error*. Namun, hanya memahami arah pergerakan harga Bitcoin di masa depan saja tidak cukup membantu investor, sehingga mendapatkan harga Bitcoin yang spesifik sebagai harga referensi akan lebih berguna [6]. Untuk menciptakan model yang mampu memprediksi harga *cryptocurrency*, terutama Bitcoin, dapat dicapai menggunakan serangkaian teknik dan metodologi *machine learning* [19].

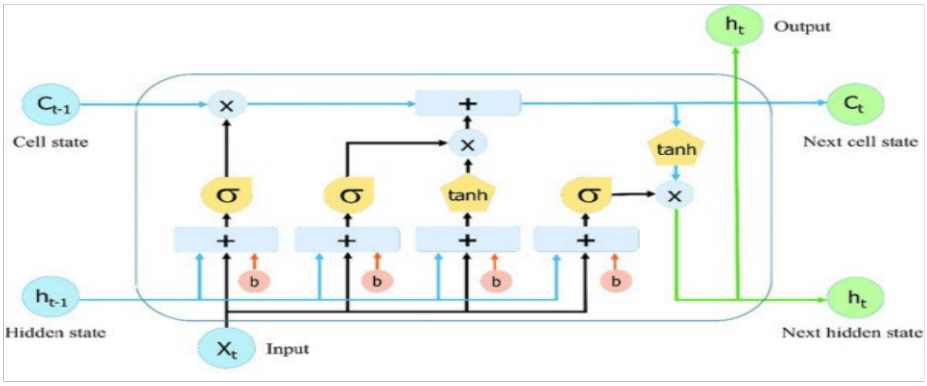
2.1.2 Machine Learning

Machine learning merupakan bagian dari ilmu/teknologi kecerdasan buatan yang telah menjadi komponen kunci dalam solusi digitalisasi yang telah menarik perhatian besar dalam dunia digital. Pada *machine learning*, sebuah program komputer diberi tugas untuk melakukan beberapa pekerjaan, program dikatakatan telah belajar dari pengalamannya jika kinerjanya dapat diukur dalam tugas-tugas ini dan meningkat seiring dengan bertambahnya pengalaman. Jadi, *machine learning* ini mengambil keputusan dan melakukan prediksi/ramalan berdasarkan data [20].

Secara umum, efektivitas dan efisiensi dari solusi *machine learning* bergantung pada sifat dan karakteristik data serta performa dari *learning algorithms*. Dalam ranah algoritma *machine learning*, terdapat berbagai teknik seperti *classification analysis*, *regression*, *data clustering*, *feature engineering* dan *dimensionality reduction*, *association rule learning*, dan *reinforcement learning* untuk membangun sistem berbasis data secara efektif [21]. Model algoritma *machine learning* yang digunakan untuk prediksi harga *cryptocurrency* Bitcoin diantaranya adalah *Support Vector Regression* (SVR), Prophet, *Autoregressive Integrated Moving Average* (ARIMA), *Long Short-Term Memory* (LSTM), dan XGBoost [22].

2.1.3 Long Short Term Memory (LSTM)

Long Short Term Memory merupakan pengembangan dari *Recurrent Neural Network* (RNN). LSTM dirancang khusus untuk menghindari *long-term dependencies*, sekaligus mengatasi masalah *vanishing gradient* dengan mekanisme tambahan untuk mengatur informasi, sehingga informasi tersebut dapat tetap dipertahankan dalam jangka waktu yang lama [11]. LSTM menggunakan mekanisme *gate control* yang terdiri dari *forget gate*, *input gate*, *cell state*, dan *output gate* [15], [16]. Model jaringan LSTM memiliki fungsi memori yang dapat dilihat pada Gambar 2.1.



Gambar 2.1 Model Jaringan Long Short Term Memory [23]

Dimana x_t merupakan masukan/input dari node saat ini, C_t mewakili vektor nilai kandidat di proses perhitungan, σ mewakili fungsi aktivasi sigmoid *feedforward* dari lapisan jaringan, dan *tanh* mewakili fungsi aktivasi dari lapisan jaringan *feedforward*. Adapun langkah-langkah algoritma LSTM sebagai berikut [15], [16]:

1. Hitung *input gate* menggunakan persamaan :

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i) \quad (2.1)$$

2. Hitung waktu kalkulasi t pada saat memasukkan kandidat nilai *cell state* $\tilde{C}_t$ menggunakan persamaan :

$$\tilde{C}_t = \tanh(W_c . [h_{t-1}, x_t] + b_c) \quad (2.2)$$

3. Hitung fungsi aktivasi *forget gate* menggunakan persamaan :

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

4. Berdasarkan kalkulasi di atas, diperoleh nilai perubahan *cell state* pada waktu t dan dirumuskan dengan :

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

5. Nilai *output gate* diperoleh dengan persamaan :

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

6. Lapisan *hidden layer* berikutnya dapat diperoleh dengan persamaan :

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

dimana W_f, W_i, W_c, dan W_o mewakili *weight matrices* dan b_f, b_i, b_c, dan b_o mewakili *bias terms* dari setiap *gate* [24].

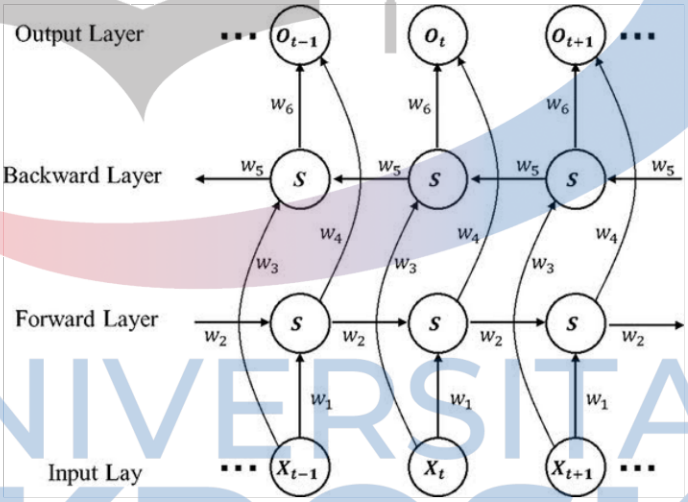
2.1.4 Bidirectional Long Short Term Memory (BiLSTM)

Bidirectional Long Short Term Memory (BiLSTM) adalah struktur deformasi dari LSTM yang memiliki lapisan *forward* LSTM maju dan *backward* LSTM. Dengan memanfaatkan konsep sebelum dan sesudahnya saat memahami konteks, BiLSTM dapat mempertimbangkan *future information* dan *past information* dari data yang ada secara bersamaan [25]. BiLSTM menggunakan data yang sudah melewati tahap *pre-process* sebagai masukan, melewati lapisan jaringan saraf LSTM *forward layer* dan *backward layer*,

kemudian menuju ke *full connection layer* dan mengeluarkan hasil prediksi seperti yang ditunjukkan pada Gambar 2.2 [15], [16].

Gambar 2.2 Struktur Diagram BiLSTM [15], [16]

Pada lapisan *forward*, perhitungan dilakukan dari awal hingga akhir, dan hasil dari lapisan tersembunyi setiap saat diperoleh dan disimpan. Pada lapisan *backward*, perhitungan dibalik dari akhir ke awal untuk mendapatkan dan menyimpan hasil dari lapisan *backward* setiap saat. Pada Gambar 2.3, terlihat bahwa masing-masing dari enam bobot unik digunakan berulang-ulang setiap saat. Masukan ke lapisan tersembunyi *forward* dan *backward layer* (w1, w3), lapisan tersembunyi ke dirinya sendiri (w2, w5), dan lapisan tersembunyi *forward* dan *backward layer* ke *output layer* (w4, w6). *Output layer* adalah hasil dari penggabungan *forward* dan *backward layer* [15], [16].



Gambar 2.3 Model Jaringan BiLSTM [26]

Adapun langkah-langkah dari algoritma BiLSTM sebagai berikut [15], [16] :

1. Hitung masukan ke lapisan tersembunyi *forward layer* dengan persamaan :

$$\overrightarrow{h_t} = f(w_1 x_t + w_2 h_{t-1}) \quad \text{.....(2.7)}$$

2. Hitung lapisan tersembunyi *backward layer* dengan persamaan :

$$\overleftarrow{h_t} = f(w_3 x_t + w_5 h_{t+1}) \quad \text{.....(2.8)}$$

3. Hitung lapisan *output* dengan menggabungkan keluaran dari *forward layer* dan *backward layer* seperti pada persamaan berikut :

$$o_t = g(w_4 \overrightarrow{h_t} + w_6 \overleftarrow{h_t}) \quad \text{.....(2.9)}$$

dimana f merupakan fungsi aktivasi dan g merupakan *optimizer* yang digunakan.

2.1.5 Firefly Algorithm

Algoritma *firefly* adalah algoritma metaheuristik yang terinspirasi oleh perilaku kedipan kunang-kunang (*firefly*). Algoritma *firefly* adalah algoritma pencarian kelompok secara acak. Algoritma ini menggunakan pencahayaan *firefly* untuk mensimulasikan titik solusi dalam ruang solusi. Nilai kilauan pada setiap *firefly* menunjukkan kualitas solusi. *Firefly* menggunakan perilaku kedipan untuk menarik *firefly* lainnya secara berulang sehingga menemukan posisi yang optimal [13], [27].

Kecerahan *firefly* terkait dengan nilai fungsi objektif. Jika *firefly* memiliki kecerahan dan posisi yang lebih baik, mereka akan menarik lebih banyak *firefly*. *Firefly* paling terang bergerak secara acak karena mereka tidak dapat ditarik oleh *firefly* lain [28]. Dapat diasumsikan bahwa daya tarik *firefly* bergantung pada kecerahan mereka, yang pada gilirannya ditentukan oleh nilai-nilai *fitness* dalam ruang pencarian. Artinya, I(x) berkaitan dengan f(x), dimana I(x) menggambarkan intensitas cahaya *firefly* di lokasi x, dan f adalah fungsi objektif [29].

Parameter masukan untuk algoritma *firefly* adalah N, yaitu jumlah *firefly* yang akan membentuk populasi, parameter acak α, daya tarik β_0 ketika jarak r dari sumber adalah 0, koefisien pencahayaan γ, dan jumlah maksimum iterasi dari algoritma [30].

Intensitas cahaya (I) dari *firefly* ditunjukkan dalam persamaan 2.10. Nilai ini bergantung pada intesitas cahaya awal (I_0), gamma (γ), dan jarak (r) [31].

$$I = I_0 e^{-\gamma r} \quad \text{.....(2.10)}$$

Daya tarik β(r) dari *firefly* sumber terhadap *firefly* yang mengamatinya pada jarak r dari sumber dapat didefinisikan pada persamaan 2.11 [29]. Daya tarik ini bergantung pada kuadrat jarak, gamma, dan nilai awal [31].

$$\beta = \beta_0 e^{(-\gamma r^2)} \quad \text{.....(2.11)}$$

Jarak antara satu *firefly* dengan yang lainnya penting, karena intensitas cahaya dan daya tarik keduanya berubah sesuai dengan jarak. Oleh karena itu, perubahan ini akan menentukan pergerakan *firefly* [31]. Jarak ini diukur sebagai *cartesian distance* yang terlihat pada persamaan 2.12 [30].

$$r_{ij} = \|x_i - x_j\| = \sum_{k=1}^{d} \left(x_i^k - x_j^k\right)^2 \quad \text{.....(2.12)}$$

dimana x_k^i adalah komponen ke-k dari posisi *firefly* ke-i pada posisi x_i.

Firefly bergerak menuju *firefly* yang lebih terang dan lebih menarik. Pergerakan ini dapat diungkapkan melalui persamaan 2.13. Ekspresi kedua dalam persamaan tersebut berasal dari rumus daya tarik. Dengan kata lain, hal ini diungkapkan melalui perkalian jarak dan daya tarik antara dua *firefly* [31]. Ekspresi ketiga adalah komponen acak dengan α sebagai parameter acak. ε adalah vektor acak yang mengikuti *Gaussian Distribution* atau *Uniform Distribution*. Dalam algoritma ini, ε diimplementasikan dengan (*rand* - $\frac{1}{2}$), dimana *rand* adalah generator angka acak yang terdistribusi secara seragam antara [0,1], parameter $\beta_0 = 1$ dan α berada di antara [0,1] [29].

$$x_i^{t+1} = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha\varepsilon_i^t \quad \text{.....(2.13)}$$

Parameter γ memiliki peran yang sangat penting pada kecepatan konvergensi. Nilai parameter ini secara teoritis dapat menggunakan nilai pada rentang $\gamma \in [0, \infty)$. Optimisasi bervariasi tergantung pada masalahnya. Biasanya antara 0.1 dan 10 [31].

Adapun pseudocode algoritma *firefly* terlihat pada Gambar 2.4.

```
Begin
    1. Initialisation max iteration, α, β_o, γ
    2. Generate initial population
    3. Define the Objective function f(x),
    4. Determine Intensity (I) at cost (x) of each individual determined by f(x_i)
    5. While (t < Iter max)
                For i=1 to n
                    For j=1 to n
                        if (I_j > I_i)
                            Move firefly i towards j in K dimension
                        end if
                    Evaluate new solutions and update light intensity
                end for j
            end for i
            Rank the fireflies and find the current best
        end while
    6. Post process results and visualization
End procedure
```

Gambar 2.4 Pseudocode Algoritma Firefly [32]

2.1.6 Improved Firefly Algorithm (IFA)

Algoritma *firefly* cukup sederhana, namun memiliki banyak kekurangan [27]. Parameter dalam FA standar yang diatur sebelumnya dapat menyebabkan konvergensi dini dari algoritma atau algoritma tidak dapat konvergen karena pengaturan parameter yang tidak tepat. Oleh karena itu, perlu adanya perbaikan pada FA standar untuk mencapai kinerja optimisasi yang lebih baik [28]. Untuk meningkatkan akurasi solusi dan kinerjanya, perlu dilakukan perubahan iterasi pembaruan posisi dan strategi pembaruan faktor panjang langkah algoritma [27].

2.1.6.1 The Minimum Attraction

Dengan menganalisis parameter FA, ketika koefisien serapan cahaya γ terlalu besar atau memiliki nilai tetap, interval optimisasi menjadi terlalu besar, yang akan mengakibatkan daya tarik antar *firefly* individual mendekati nol dan *firefly* individual kehilangan daya tarik satu sama lain. Untuk menghindari jarak antar *firefly* individual menjadi terlalu jauh sehingga daya tarik antara mereka hampir nol, diperkenalkan konsep daya tarik minimum guna mencegah *firefly* individual bergerak secara acak [28] dengan persamaan :

$$\beta_{ij}(r_{ij}) = \beta_{min} + (\beta_0 - \beta_{min})e^{-\gamma r_{ij}^2} \quad \text{.....(2.14)}$$

dimana $\beta_0 = 1$, $\beta_{min} \in [0, 1]$.

Jadi, jika jarak antara *firefly* terlalu jauh, $e^{-\gamma r_{ij}^2}$ mendekati 0, daya tarik antar *firefly* dapat menjadi β_{min}.

2.1.6.2 Adaptive Step Length Factor

Pada algoritma *firefly*, faktor *step length* α adalah konstan. Pada tahap awal, α dapat membantu algoritma dalam pencarian optimal global, tetapi ketika algoritma mendekati jumlah iterasi maksimum, karena nilai konstan α, sebagian besar *firefly* bergerak bolak-balik di sekitar solusi terbaik dan tidak dapat menyatu dengan yang lain.

Untuk menyeimbangkan kontradiksi antara tahap awal dan tahap akhir proses konvergensi seluruh perhitungan, penyesuaian dinamis pada faktor *step length* dirumuskan [13], [27] dengan :

$$\alpha = \alpha_0 * P \quad \text{.....(2.15)}$$

$$P = \frac{Max - t}{Max} \quad \text{.....(2.16)}$$

dimana Max adalah jumlah iterasi maksimum dan t adalah jumlah iterasi saat ini.

2.1.6.3 Inertia Weight

Belajar dari peningkatan pada *Particle Swarm Optimization* (PSO) yang menggunakan *inertia weight*, untuk meningkatkan algoritma dasar dapat dilakukan dengan mengurangi *inertia weight* secara linear, dijelaskan dalam persamaan 2.17 [13].

$$w = w_{max} - (w_{max} - w_{min}) * \frac{t}{MaxT} \quad \text{.....(2.17)}$$

Dimana w_{max} dan w_{min} mewakili *weight* maksimum dan minimum. t dan MaxT mewakili jumlah iterasi saat ini dan jumlah iterasi maksimum. Strategi baru untuk memperbaharui posisi *firefly* adalah sebagai berikut [13] :

$$x_i(t+1) = wx_i(t) + \beta\left(x_j(t) - x_i(t)\right) + \alpha\varepsilon_i \quad \text{.....(2.18)}$$

2.1.7 Penelitian Terdahulu

Prediksi harga *cryptocurrency* sering dianggap sebagai salah satu tantangan terbesar dalam memprediksi *time series* data, penyebab utamanya adalah volatilitas harga yang signifikan dan jumlah elemen yang sulit diprediksi yang terlibat [17]. Dalam beberapa tahun terakhir, ada banyak penelitian yang dilakukan untuk melakukan prediksi terhadap *time series* data, termasuk harga *cryptocurrency* Bitcoin.

Penelitian pendahuluan pertama ini mengacu pada prediksi harga Bitcoin yang menggunakan model *Recurrent Neural Network* (RNN) dengan analisis sentimen pada Twitter. *Dataset* yang digunakan diambil dari Twitter Account seperti BitcoinNews (@BTCTN), CryptoCurrency (@cryptocurrency), CryptoYoda (@CryptoYoda1338), BitcoinMagazine (@BitcoinMagazine), Bitcoin Forum (@BitcoinForums), CoinDesk (@coindesk) dan Roger Ver (@rogerkver) pada periode 1 Januari 2015 s/d 31 Desember 2017. *Dataset* harga dengan periode yang sama diambil dari Coinmarketcap. Hasilnya, akurasi prediksi harga dengan model RNN adalah 77.62% [8]. Namun, RNN memiliki keterbatasan seperti masalah *vanishing* dan *exploding gradient descent*, serta ketidakefisienan dalam melacak *long-term dependencies* [9].

Pada penelitian lain, peneliti menggunakan model yang berbeda untuk dibandingkan dalam melakukan prediksi *cryptocurrency*, yaitu *Long Short Term Memory* (LSTM) dan *Autoreggresive Moving Integrated Average* (ARIMA). *Cryptocurrency* yang coba diprediksi adalah BTC/USD, ETH/BTC, dan lain sebagainya. *Dataset* yang digunakan diambil dari

website Bitfinex sebanyak 10000 informasi harga *cryptocurrency* dengan interval 5 detik. *Dateset* kemudian di-*split* menjadi data *training* dan data *testing* yang masing-masing berjumlah 8000 dan 2000 data.). Hasil penelitian yang telah dilakukan menunjukkan bahwa metode LSTM lebih efisien dan menghasilkan tingkat akurasi yang lebih tinggi dibandingkan metode ARIMA [10]. Pada penelitian lain di tahun yang berbeda, peneliti melakukan hal yang sama, yaitu membandingkan metode LSTM dan ARIMA pada prediksi harga *cryptocurrency*. *Cryptocurrency* yang diprediksi adalah Bitcoin, Ethereum, Binance Coin, Tether, dan Cardano. *Dataset* harga diambil dari website Yahoo Finance pada periode 9 November 2017 s/d 28 Juni 2021. *Dataset record* terdiri dari 1238 baris dan 7 kolom (*date, open, high, low, close, adj. close,* dan *volume*). Hasilnya LSTM memiliki akurasi yang lebih baik untuk memprediksi harga Bitcoin jika dibandingkan dengan ARIMA dilihat dari *Root Mean Square Error* (RMSE) dan *Mean Absolute Percentage Error* (MAPE). LSTM mendapatkan RMSE = 2879,051 dan MAPE = 5,202 sedangkan ARIMA mendapatkan RMSE = 0.797 dan MAPE = 6,952 [33]. LSTM dirancang khusus untuk menghindari *long-term dependencies*, sekaligus mengatasi masalah *vanishing gradient* dengan mekanisme tambahan untuk mengatur informasi, sehingga informasi tersebut dapat tetap dipertahankan dalam jangka waktu yang lama [11].

Namun, model LSTM juga rentan terhadap kinerja yang buruk apabila konfigurasi *hyperparameter* yang dilakukan tidak tepat [12]. Untuk mengatasi hal tersebut, pada penelitian lain, peneliti menerapkan *Improved Firefly Algorithm* (IFA) pada model LSTM dalam memprediksi *network traffic*. Peneliti melakukan perbandingan model IFA-LSTM dengan metode *Back Propagation* (BP) dan *Gate Recurrent Unit* (GRU). Hasil penelitian tersebut, IFA berhasil meningkatkan akurasi nilai prediksi dan lebih cepat dibandingkan BP dan GRU [27].

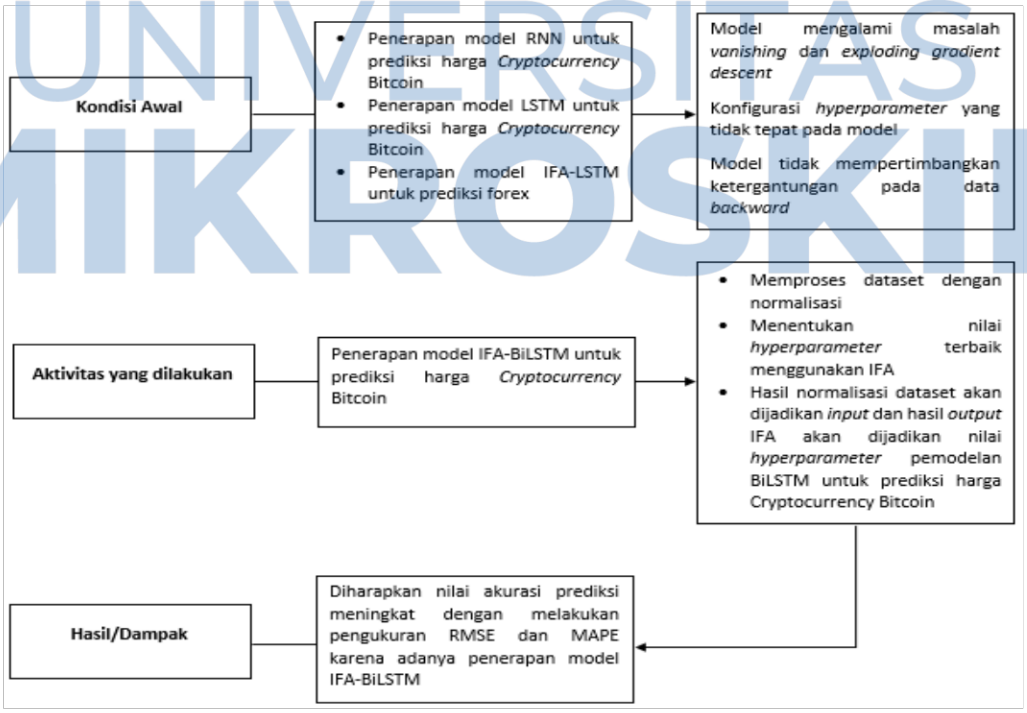
Penelitian yang menerapkan *Improved Firefly Algorithm* (IFA) juga dilakukan pada model CEEMDAN-LSTM untuk mengoptimalkan *hyperparameter* LSTM dalam prediksi forex. *Dataset* yang digunakan diambil dari dukascopy.com, yaitu data historis pasangan mata uang EUR/USD, AUD/USD, GBP/USD mulai dari 1 Januari 2010 s/d 30 Desember 2019 dengan atribut *high*, *open*, *close*, dan *low* karena lebih berkorelasi dengan harga penutupan. Hasil penelitian membuktikan IFA berhasil menyelesaikan masalah optimisasi dan konfigurasi *hyperparameter* LSTM [13].

Namun, penelitian sebelumnya [13], [27] belum membahas lebih lanjut model LSTM yang memiliki keterbatasan hanya menggunakan ketergantungan pada data *forward* saja dan tidak secara efisien mempertimbangkan ketergantungan pada data *backward* untuk

mendapatkan informasi yang berguna [15]. Model *Bidirectional Long Short-Term Memory* (BiLSTM) pun diusulkan dan dibandingkan dengan LSTM dan GRU pada sebuah penelitian untuk memprediksi harga *cryptocurrency*. *Dataset* yang digunakan diambil dari https://finance.yahoo.com/ dalam format CSV. Berkas CSV tiga lembar terpisah : BTC, ETH, dan LTC. Data *training* mulai tanggal 1 Januari 2018 hingga 31 Desember 2021 (80% dari data), sedangkan data *testing* mulai tanggal 1 Januari 2022 hingga 1 Januari 2023 (20% dari data). Dari penelitian tersebut diperoleh hasil bahwa model BiLSTM memiliki nilai RMSE dan MAPE paling akurat untuk tiga *cryptocurrency* yang diteliti, yaitu Bitcoin, Ethereum, dan Litecoin [11]. Pada penelitian lain yang membandingkan LSTM dan BiLSTM untuk prediksi harga saham, juga diperoleh hasil bahwa model BiLSTM memiliki nilai RMSE, MAE, dan Loss yang lebih baik dibandingkan dengan LSTM [16]. Bidirectional LSTM menggunakan kombinasi dari *forward* LSTM dan *backward* LSTM untuk memprediksi, sehingga memungkinkan model untuk mendapatkan data historis dan masa depan secara menyeluruh untuk analisis, dan hasil prediksinya menjadi lebih akurat [15], [16].

2.2 Kerangka Pikir Pemecahan Masalah

Kerangka ini menggambarkan bagaimana masalah penelitian dapat diselesaikan melalui solusi yang diusulkan dan memberi dampak yang dapat menyelesaikan permasalahan penelitian. Kerangka ini dapat dilihat pada Gambar 2.5.



Gambar 2.5 Kerangka Pikir Pemecahan Masalah

Pada gambar 2.5 di atas, kondisi awal dimulai dengan adanya penelitian yang menerapkan model RNN untuk melakukan prediksi terhadap harga *cryptocurrency* Bitcoin. Namun, RNN memiliki keterbatasan seperti masalah *vanishing* dan *exploding gradient descent*, serta ketidakefisienan dalam melacak *long-term dependencies* [9]. Dengan ranah yang sama yaitu prediksi harga *cryptocurrency*, dilakukan penelitian yang membandingkan model LSTM dan ARIMA. Diperoleh hasil bahwa model LSTM memiliki tingkat akurasi yang lebih baik dibandingkan ARIMA. LSTM mampu menghindari *long-term dependencies*, sekaligus mengatasi masalah *vanishing gradient* dengan mekanisme tambahan untuk mengatur informasi, sehingga informasi tersebut dapat tetap dipertahankan dalam jangka waktu yang lama [11]. Namun, model LSTM rentan terhadap kinerja yang buruk apabila konfigurasi *hyperparameter* yang dilakukan tidak tepat [12]. Dilanjutkan dengan penelitian untuk prediksi forex yang menerapkan model IFA-LSTM. Hasil penelitian membuktikan IFA berhasil menyelesaikan masalah optimisasi dan konfigurasi *hyperparameter* LSTM [13]. Namun, penelitian yang sudah dijabarkan belum membahas lebih lanjut model LSTM yang memiliki keterbatasan hanya menggunakan ketergantungan pada data *forward* saja dan tidak secara efisien mempertimbangkan ketergantungan pada data *backward* untuk mendapatkan informasi yang berguna [15].

Berdasarkan kondisi awal tersebut, maka penelitian yang akan dilakukan adalah Menggunakan model IFA-BiLSTM. *Dataset* yang digunakan akan melewati proses normalisasi kemudian hasil dari normalisasi akan digunakan sebagai *input* model. Model BiLSTM digunakan untuk mengatasi keterbatasan pada LSTM yang hanya menggunakan data *forward* saja dan tidak secara efisien mempertimbangkan data *backward*. Metode IFA diterapkan untuk mencari dan menentukan nilai *hyperparameter* terbaik pada BiLSTM untuk menghasilkan prediksi harga *cryptocurrency* Bitcoin dengan nilai akurasi yang tinggi.