BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Berikut merupakan penelitian-penelitian yang sudah dilakukan sebelumnya terkait prediksi saham menggunakan *machine learning* yang juga menjadi acuan dalam penelitian ini.

Tabel 2. 1 Penelitian Terdahulua

Penulis	Judul	Jurnal	Model	Kesimpulan
• Yu Chen • Et al	Stock Price Forecast Based on CNN-BiLSTM-ECA Model [5]	Scientific Programming, Vol 2021	CNN-BiLSTM-ECA	Algoritma CNN-BiLSTM-ECA memiliki performa terbaik dalam memprediksi harga Shanghai Composite Index dibandingkan dengan algoritma LSTM biasa dan varian-variannya.
• Haiyao Wang • Jianxuan Wang • Lihui Cao • Yifan Li • Qiuhong Sun • Jingyang Wang	A Stock Closing Price Prediction Model Based on CNN-BiSLSTM [6]	Complexity, Vol 2021	CNN-BiSLSTM	Algoritma CNN-BiSLSTM memiliki akurasi tertinggi dalam memprediksi harga Shenzhen Component Index dibandingkan dengan algoritma LSTM, BiLSTM, dan CNN-LSTM.
• Widiputra, H • Mailangkay, A • Gautama, E	A novel hybrid deep learning model for price prediction [9]	International Journal of Electrical and Computer Engineering (IJECE), Vol 13, No 3	CNN-LSTM	Algoritma CNN-LSTM memiliki nilai MSE dan MAE yang paling rendah sehingga menjadikannya yang paling baik dibandingkan dengan algoritma LSTM biasa dan ARIMA menggunakan data harga saham, nilai tukar mata uang, dan mata uang kripto.
• Aldhyani, T H H • Alzahrani, A	Framework for Predicting and Modeling Stock Market Prices Based on Deep Learning Algorithms [10]	Electronics 2022, 11, 3149	CNN-LSTM	Menunjukkan bahwa penggabungan CNN dengan LSTM dapat meningkatkan akurasi prediksi harga saham dengan data time series, memberikan landasan bagi pengembangan model hybrid dalam penelitian ini.
• Widiputra, H • Mailangkay, A • Gautama, E	Multivariate CNN-LSTM Model for Multiple Parallel Financial Time-Series Prediction [11]	Complexity, Vol 2021	CNN-LSTM	CNN-LSTM menjadi algoritma terbaik dalam memprediksi harga indeks saham di Shanghai, Jepang, Singapore, dan Indonesia dibandingkan dengan algoritma CNN biasa dan algoritma LSTM biasa.
• Pin Lv • Qinjuan Wu • Jia Xu • Yating Shu	Stock Index Prediction Based on Time Series Decomposition and Hybrid Model [12]	Entropy 2022, 24, 146	CAL	Konfigurasi hyperparameter yang digunakan untuk melatih algoritma CAL adalah batch size dengan jumlah 10, time step berjumlah 10, epoch berjumlah 200, dengan algoritma *optimizer* Adam, dan *loss function* MSE. Hasilnya algoritma CAL memiliki tingkat kesalahan paling rendah dibandingkan dengan algoritma lainnya. Penggunaan mekanisme *attention* berhasil menjadi

Penulis	Judul	Jurnal	Model	Kesimpulan
• Gao, Shangzhi	Research on Stock Price Prediction Based on CNN-LSTM Combined Model [13]	Advances in Computer, Signals and Systems (2023) Vol. 7: 73-79	CNN-LSTM with Attention	CNN-LSTM dengan mekanisme Attention mampu memberikan bobot yang lebih baik pada fitur penting sehingga meningkatkan akurasi dan stabilitas prediksi. Hasil eksperimen menunjukkan bahwa model CNN-LSTM hybrid ini mampu mengatasi kompleksitas dan non-linearitas pasar saham dengan performa yang lebih baik dibandingkan model CNN atau LSTM.
• Lee, Sanghyuk • Yang, Yeongwook • Aiyanyo, Imatitikua • Sang, Shuai • Li, Lu	A Novel Variant of LSTM Stock Prediction Method Incorporating Attention Mechanism [14]	Mathematics 2024, 12, 945	AMV-LSTM	Model varian LSTM ini memodifikasi struktur *forget gate* dan input dengan menambahkan *simple forget gate* pada sel *long-term* untuk mengatasi masalah *overfitting* dan ketidakstabilan yang sering dialami LSTM standar. Selanjutnya, mekanisme *attention* diterapkan untuk memperkuat kemampuan model dalam menyorot fitur penting dan meningkatkan generalisasi. Hasil eksperimen menunjukkan bahwa model Attention Mechanism Variant LSTM (AMV-LSTM) ini memberikan akurasi prediksi yang lebih tinggi dan konvergensi lebih baik dibandingkan dengan model LSTM biasa.
• Wang, Jingyang • Wang, Xiaoxiao • Li, Jiazheng • Wang, Haiyao	A Prediction Model of CNN-TLSTM for USD/CNY Exchange Rate Prediction [15]	IEEE Access 2021	CNN-TLSTM	CNN-TLSTM merupakan sebuah model yang hibrida yang menggabungkan CNN dengan TLSTM yang merupakan sebuahvarian LSTM yang dimodifikasi dengan menggunakan fungsi aktivasi tanh pada gate inputnya. Model ini menunjukkan kinerja prediksi yang lebih baik dibandingkan metode lain seperti MLP, RNN, dan LSTM biasa. Hasil evaluasi dengan metrik MAPE, MSE, dan R2 mengonfirmasi bahwa CNN-TLSTM memberikan prediksi yang paling akurat dan efektif untuk nilai tukar USD/CNY.
• Jilin Zhang • Lishi Ye • Yongzeng Lai	Stock Price Prediction Using CNN-BiLSTM-Attention Model [16]	Mathematics 2023, 11(9), 1985	CNN-BiLSTM-Attention	Penelitian ini mengembangkan model prediksi harga saham yang menggabungkan Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM), dan mekanisme Attention untuk meningkatkan akurasi prediksi. CNN berperan dalam mengekstraksi fitur temporal dari data harga saham, sementara BiLSTM mampu menangkap ketergantungan waktu dua arah dalam deret data. Mekanisme Attention kemudian memberikan bobot dinamis pada fitur-fitur penting yang berpengaruh terhadap harga saham, sehingga model dapat fokus pada informasi krusial. Uji coba pada berbagai indeks saham, termasuk CSI300 dan indeks internasional, menunjukkan bahwa model CNN-BiLSTM-Attention memberikan hasil prediksi yang lebih akurat dibandingkan model lain seperti LSTM dan CNN-LSTM

Sudah terdapat beberapa penelitian yang menyebutkan bahwa LSTM dapat memberikan performa yang lebih baik dibanding RNN atau Recurrrent Neural Network dalam tugas memprediksi *timeseries*. Salah satu alasannya adalah karena LSTM tidak memiliki masalah vanishing gradient seperti RNN. Selain itu, LSTM juga bisa digabungkan dengan algoritma lainnya untuk dapat meningkatkan performanya lebih baik lagi. Contoh algoritma tersebut adalah Convolutional Neural Network atau CNN. Biasanya CNN digabungkan dengan LSTM dengan tujuan untuk menentukan fitur-fitur utama dari data yang diberikan kepada model untuk dilatih. Selain menggabungkan algoritma, LSTM juga dapat dimodifikasi pada bagian dalamnya, seperti salah satu bentuknya adalah dengan mengaplikasi formula 1-tanh pada *output gate* LSTM. Terdapat dua studi dimana masing-masing mengusulkan sebuah algoritma hibrida baru, yaitu CNN-BiLSTM-ECA dan CNN-BiSLSTM. Masing-masing studi menunjukkan bahwa model yang diusulkan masing-masing studi adalah yang terbaik dibandingkan dengan model yang lebih sederhana seperti LSTM. Permasalahannya adalah kedua studi ini menggunakan objek penelitian yang berbeda sehingga sulit untuk menentukan algoritma terbaik diantara keduanya. Penelitian kali ini bertujuan untuk menjawab permasalahan ini dengan melakukan uji coba menggunakan model yang telah diusulkan oleh masing-masing studi dengan objek penelitian yang sama. Cara ini diharapkan dapat dengan objektif menentukan algoritma terbaik diantara CNN-BiLSTM-ECA dan CNN-BiSLSTM dalam memprediksi harga saham.

2.2 Tinjauan Teori

2.2.1 Saham

Saham merupakan instrumen pasar keuangan yang sangat diminati, memberikan perusahaan pilihan yang signifikan untuk mendapatkan dana. Dari perspektif investor, saham dianggap sebagai investasi menarik karena memiliki potensi untuk memberikan keuntungan yang menggiurkan [1]. Saham dapat dijelaskan sebagai kontribusi modal dari individu atau entitas perusahaan ke dalam suatu Perseroan Terbatas (PT). Dengan menyumbangkan modal tersebut, pihak

tersebut memperoleh hak terhadap pendapatan perusahaan, klaim atas aset, dan keikutsertaan dalam Rapat Umum Pemegang Saham (RUPS). Berikut ini adalah keuntungan dan risiko yang terkait dengan melakukan investasi dalam saham:

Keuntungan:

- *Capital Gain*: Merupakan kondisi di mana pemegang saham dapat menjual saham dengan harga lebih tinggi daripada harga beli awal.
- Dividen: Dividen merupakan pembagian keuntungan oleh perusahaan yang diberikan kepada pemegang saham yang berasal dari laba yang diperoleh. Pemegang saham berhak menerima dividen jika mereka memegang saham tersebut untuk jangka waktu tertentu [17].

Risiko:

- Likuidasi: Risiko terkait dengan kemungkinan likuidasi perusahaan ketika pemegang saham telah membeli saham dari perusahaan yang kemudian dinyatakan bangkrut atau dibubarkan. Klaim hak pemegang saham menjadi prioritas terakhir setelah seluruh kewajiban perusahaan dilunasi. Jika tidak ada sisa kekayaan perusahaan setelah penjualan aset, pemegang saham tidak akan menerima hasil likuidasi [18].
- *Capital Loss*: Pemegang saham menjual sahamnya dengan harga yang lebih rendah daripada harga pembelian sehingga mengalami kerugian.

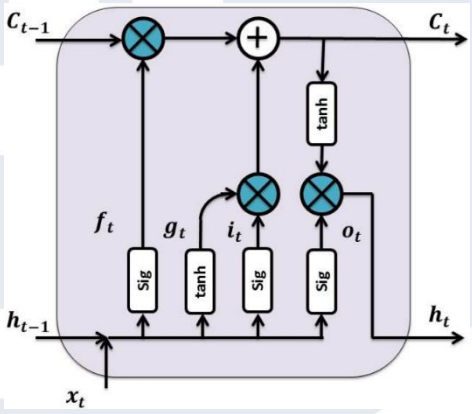
2.3 Framework, Algoritma, dan Metode Evaluasi yang Digunakan

2.3.1 Algoritma

2.3.1.1 Long-Short Term Memory

Long-Short Term Memory atau LSTM adalah salah satu algoritma Recurrent Neural Network (RNN) untuk *deep learning* yang dapat

mempelajari data *time-series* yang lebih panjang. LSTM dibuat untuk mengatasi masalah yang sering muncul di dalam model RNN seperti *vanishing and exploding gradients* [19]. *Vanishing and exploding gradients* adalah sebuah masalah dimana pada jumlah data yang besar gradien yang diperoleh akan sangat kecil jika *weight* kurang dari 0 sedangkan gradien data akan terlalu besar jika *weight* data tersebut lebih besar dari 1. Berikut adalah gambar dari arsitektur sebuah cell LSTM.



Gambar 2. 1 Arsitektur *cell* LSTM [20]

Jaringan LSTM sendiri terdiri dari *input layer*, *hidden layer(s),* dan sebuah *output layer*. Di dalam *hidden layer* ini terdapat beberapa *memory cell* yang menjadikan LSTM unik. Terdapat tiga *gate* utama di dalam setiap *memory cell* yaitu *forget gate, input gate*, dan *output gate* [20]. Berikut adalah penjelasan untuk masing-masing *gate*.

1. Forget Gate

   Forget gate berfungsi untuk menentukan informasi apa yang perlu dihilangkan [20]. Semakin dekat nilainya dengan 1 maka informasi yang ada akan benar-benar diingat dan jika semakin dekat nilainya dengan 0 maka informasi yang ada akan lebih sedikit diingat. jika nilai yang diperoleh adalah sama dengan 0, maka informasi

tersebut akan benar-benar dilupakan. Rumus matematika yang digunakan dalam bagian ini adalah sebagai berikut.

$$f_t = \sigma\left(W_{x_t}^{(f)} + U_{ht-1}^{(f)} + b^{(f)}\right)$$

Rumus 2. 1 Rumus Forget Gate

Keterangan:

- f_t : Nilai $forget\ gate$
- σ : sigmoid
- $W_{x_t}^{(f)}$: *weight* pada *time step* t
- $U_{ht-1}^{(f)}$: *weight* pada *time step* t-1
- $b^{(f)}$: bias

2. Input Gate

Input gate berfungsi untuk menambahkan informasi baru ke dalam *memory cell* [20]. Di dalam *input gate* terdapat *sigmoid activation layer* dan juga *tanh activation layer*. Awalnya, informasi yang mungkin akan ditambahkan pada *cell state* akan dihitung terlebih dahulu. Nilai-nilai yang dibutuhkan untuk operasi ini adalah *hidden state* dari *time step* sebelumnya dan *input* dari *time step* sekarang. *Value* ini kemudian akan dimasukan kedalam *tanh activation layer*. Formula matematika pada *input gate* adalah sebagai berikut.

$$g_t = tanh\left(W_{x_t}^{(g)} + U_{ht-1}^{(g)} + b^{(g)}\right)$$

Rumus 2. 2 Rumus *Input Gate*

Keterangan:

- g_t : Nilai $input\ node$
- $W_{x_t}^{(g)}$: *weight* pada *time step* t
- $U_{ht-1}^{(g)}$: *weight* pada *time step* t-1

- $b^{(g)}$: bias

Kemudian, jumlah dari informasi yang akan ditambahkan ini akan ditentukan melalui *gate* sigmoid dengan formula matematika sebagai berikut.

$$i_t = \sigma\left(W_{x_t}^{(i)} + U_{ht-1}^{(i)} + b^{(i)}\right)$$

Rumus 2. 3 Rumus *Input Gate*

Keterangan:

- i_t : Nilai $input\ gate$
- σ : sigmoid
- $W_{x_t}^{(i)}$: *weight* pada *time step* t
- $U_{ht-1}^{(i)}$: *weight* pada *time step* t-1
- $b^{(i)}$: bias

Kedua hasil operasi tersebut akan dikalikan dan kemudian ditambahkan kepada *cell state*. Setelah itu hasil dari kedua operasi dari *forget gate* akan digabungkan untuk menjadi nilai dari *cell state* ini dengan rumus matematika sebagai berikut.

$$C_t = g_t \circ i_t + f_t \circ C_{t-1}$$

Rumus 2. 4 Rumus *Cell State*

Keterangan:

- C_t : *cell state* pada *time step* t
- g_t: nilai dari hasil operasi pada *input node*
- i_t: nilai dari hasil operasi *input gate*
- f_t: nilai dari hasil operasi *forget gate*
- C_{t-1}: *cell state* sebelumnya
- $\circ$: *Hadamard product*

Nilai dari *cell state* ini juga kemudian akan diteruskan ke *cell* pada *time step* selanjutnya.

3. Output Gate

Output gate berfungsi untuk menentukan output dari *memory cell* [20]. Mirip seperti *input gate*, *output gate* juga memiliki *sigmoid activation layer* dan juga *tanh activation layer*. Kemudian menggunakan input dari *time step* sekarang dan nilai *hidden state* dari *time step* sebelumnya akan dihitung jumlah informasi *cell state* yang perlu menjadi *output* dari *cell* dengan rumus matematika sebagai berikut.

$$o_t = \sigma\left(W_{x_t}^{(o)} + U_{ht-1}^{(o)} + b^{(o)}\right)$$

Rumus 2. 5 Rumus Output Gate

Keterangan:

- o_t : Nilai $output\ gate$
- σ : sigmoid
- $W_{x_t}^{(o)}$: *weight* pada *time step* t
- $U_{ht-1}^{(o)}$: *weight* pada *time step* t-1
- $b^{(o)}$: bias

Untuk menentukan nilai yang akan menjadi *output* dari *cell* ini, berikut rumus matematika yang digunakan.

$$h_t = o_t \circ \tanh\,(C_t)$$

Rumus 2. 6 Rumus Output Cell

Keterangan:

- h_t : nilai dari *output cell* tersebut
- o_t : nilai dari *output gate*
- C_t : nilai *cell state*

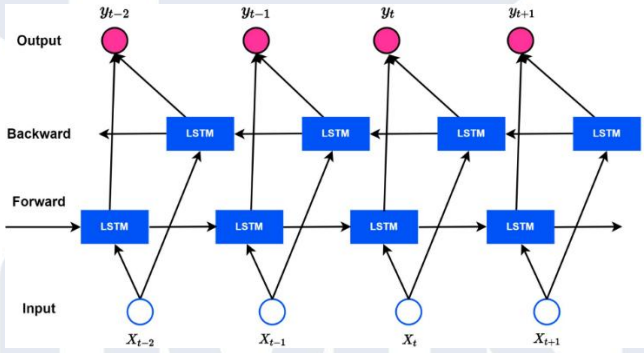
Pada setiap *gate*, terdapat *function tanh* atau *sigmoid*. Tanh (Hyperbolic Tangent) Activation Function merupakan *function* yang memastikan nilai akhir berada di antara -1 hingga 1. Hal ini berbeda dengan Sigmoid Activation Function yang memiliki nilai akhir yang berada di rentang 0

hingga 1. Dengan kedua *function* ini, nilai dari *activation value* tidak akan menjadi terlalu besar atau kecil.

Keterbatasan dari arsitektur ini jika berdiri sendiri adalah ketidakmampuannya untuk mempelajari data dari dua arah, maju dan mundur. Arsitektur ini dapat dimodifikasi sehingga bisa mendapatkan performa yang lebih baik. Dalam penelitian ini akan digunakan dua modifikasi dari LSTM, yaitu Bidirectional-LSTM dan Bidirectional-Special-LSTM.

2.3.1.2 Bidirectional-LSTM

Bidirectional-LSTM memungkinkan model untuk melihat pola-pola baru yang tidak dapat dilihat dengan model LSTM biasa. Hal ini dikarenakan data diberikan kepada model dari kedua arah, maju dan mundur.



Gambar 2. 2 Arsitektur *cell* Bidirectional-LSTM [21]

Gambar 2.2 menggambarkan arsitektur dari Bi-LSTM. Arsitektur memiliki tambahan lapisan LSTM yang terbalik sehingga hasil akhir dari *cell* ini adalah sebuah vektor yang yang menghubungkan dua *hidden state* antar kedua lapisan LSTM [21] Berikut adalah rumus yang digunakan untuk menghitung nilai dari output.

$$y_t = W_{hy}^f h_t^f + W_{hy}^b h_t^b + b_y$$

Rumus 2. 7 Rumus Output

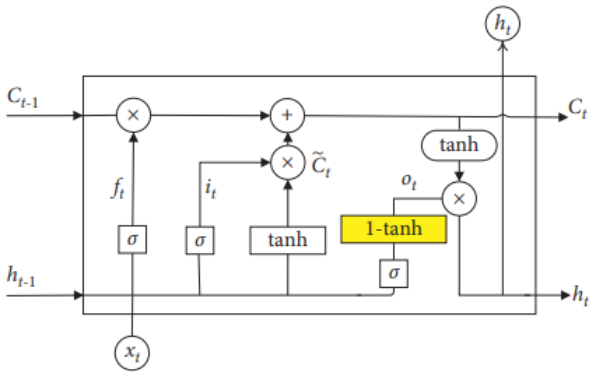
Keterangan:

- $W_{hy}^{f} h_t^{f}$: *Weight* matriks untuk koneksi antara *hidden state* maju dengan *output*
- $W_{hy}^{b} h_t^{b}$: *Weight* matriks untuk koneksi antara *hidden state* mundur dengan *output*
- b_y : bias

Meskipun BiLSTM dapat meningkatkan kinerja dalam beberapa tugas, ada beberapa tantangan yang perlu dipertimbangkan. Salah satunya adalah kompleksitas komputasi yang lebih tinggi. BiLSTM memerlukan dua lapisan LSTM: satu untuk memproses informasi dari awal ke akhir, dan satu lagi untuk memproses dari akhir ke awal. Hal ini dapat meningkatkan penggunaan sumber daya komputasi, baik dari segi waktu maupun memori. Selain itu, BiLSTM mungkin juga lebih rentan terhadap *overfitting* pada dataset yang lebih kecil, karena lebih banyak parameter yang perlu dipelajari.

2.3.1.3 Bidirectional-Special-LSTM

Bidirectional-Special-LSTM merupakan modifikasi dari arsitektur Bidirectional-LSTM. Berikut adalah struktur cell Special-LSTM.



Gambar 2. 3 Arsitektur *cell* Special-LSTM [6]

Dapat dilihat pada gambar 2.3 terdapat tambahan fungsi 1-tanh(x) pada *output gate*. Dengan begini hasil dari output gate akan berada direntang yang lebih besar dari 0 hingga 1. Hal ini dapat meningkatkan sensitivitas model terhadap untuk meningkatkan sensitivitas model terhadap perubahan sinyal input. Modifikasi ini memungkinkan model untuk lebih efektif menangkap pola non-linear dan dinamis pada data deret waktu yang *volatile*, seperti harga saham [6]. Fungsi modifikasi ini bertujuan memperbaiki kemampuan *gating* dalam mengontrol aliran informasi dengan cara menekan sinyal yang kurang relevan dan memperkuat sinyal penting, sehingga dapat mengurangi *noise* dalam data. Rumus 2.8 merupakan perhitungan yang dilakukan untuk mendapatkan nilai dari *output gate.*

$$o_t = 1 - \tanh(\sigma\left(W_{x_t}^{(o)} + U_{ht-1}^{(o)} + b^{(o)}\right))$$

Rumus 2. 8 Rumus Output

Keterangan:

- o_t : Nilai $output\ gate$
- σ : sigmoid
- $W_{x_t}^{(o)}$: *weight* pada *time step* t
- $U_{ht-1}^{(o)}$: *weight* pada *time step* t-1
- $b^{(o)}$: bias

2.3.1.4 Convolutional Neural Network

Convolutional neural network adalah *neural network* untuk mengekstrak fitur dari data melalui struktur konvolusi [6]. CNN mengambil sejumlah parameter untuk mempelajari fitur dari data. Hasilnya kemudian digunakan untuk melakukan klasifikasi. Jika berdiri sendiri, CNN biasanya kurang optimal untuk melakukan tugas regresi. Biasanya, CNN memiliki beberapa lapisan, yaitu input *layer*, *convolutional layer*, *pooling layer*, *fully connected layer*, dan *output layer*. *Convolutional layer* merupakan lapisan

dimana operasi konvolusi dilakukan melalui kernel untuk menjadi input pada lapisan selanjutnya. Ada pun *pooling layer* yang berfungsi untuk mengurangi jumlah parameter pada model sehingga kompeksitas model juga akan berkurang. Berikut adalah rumus yang digunakan dalam *convolutional layer.*

$$x_j^l = f\left(\sum_{i=1}^{I} x_i^{l-1} \otimes k_{ij}^l + b_j^l\right)$$

Rumus 2. 9 Rumus *Convolutional Layer*

Keterangan:

- x_j^l: *Output feature map*
- l : Jumlah fitur input pada lapisan ke- l
- x_i^{l-1} : *Output feature map* pada lapisan sebelumnya sekaligus *input* pada lapisan ke- l
- $\otimes$: Operasi *convolutional*
- k_{ij}^l: *convolutional kernel* dari neuron ke-*i*
- b_j^l: nilai standar deviasi dari neuron ke- *j* dan lapisan ke- l
- f: *Activation function*

Adapun rumus yang digunakan dalam *pooling layer* sebagai berikut.

$$y_i^l = max\_pooling(x_i^{l-1}, s_{scale}, s_{stride})$$

Rumus 2. 10 Rumus *Pooling Layer*

Keterangan:

- y_i^l : *Output* dari neuron ke- i pada lapisan ke- l
- $max\_pooling$: *down-sampling function*
- s_{scale}: skala dari *pooling*
- s_{stride}: panjang *step pooling*

2.3.1.5 Effective Channel Attention

Effective Channel Attention (ECA) merupakan unit arsitektural yang didasarkan pada blok *squeeze-and-excitation* yang mengurangi kompleksitas model tanpa mengurangi dimensionalitas [22]. Sebagian besar metode yang ada untuk *channel attention* berfungsi untuk mengembangkan modul *channel attention* yang lebih canggih untuk mencapai kinerja yang lebih baik sehingga secara otomatis juga akan meningkatkan kompleksitas dari model. Modul ECA tidak mengurangi dimensi namun membuat interaksi antar *channel* yang tepat sehingga dapat menjaga kinerja sambil mengurangi kompleksitas model secara signifikan. Di sisi lain, modul ini menambahkan kompleksitas terhadap model yang juga berarti memperbesar sumber daya yang dibutuhkan untuk melakukan training. Tidak hanya itu, model yang kompleks juga memperbesar kemungkinan terjadinya *overfitting*.

Proses modul ECA dimulai dengan tahap Squeeze, di mana operasi *pooling*, seperti Global Average Pooling (GAP), digunakan untuk mereduksi dimensi spasial dan menangkap informasi global dari tiap *channel*. Tahap Excitation menghasilkan bobot saluran berdasarkan fungsi aktivasi non-linier, yaitu sigmoid, yang mengubah hasil *pooling* menjadi bobot yang mencerminkan relevansi tiap *channel*. Kemudian dilakukan tahap Scale, di mana bobot yang dihasilkan pada tahap Excitation diterapkan pada setiap channel untuk memperkuat informasi yang relevan dan mengurangi pengaruh dari *channel* yang kurang penting, sehingga memperbaiki representasi fitur dan meningkatkan kinerja model secara keseluruhan.

Dalam konteks data *timeseries*, modul ECA akan memberikan bobot yang lebih besar kepada fitur-fitur yang penting sekaligus mengurangi bobot untuk fitur-fitur yang kurang penting. Struktur ECA dimulai dengan proses Global Average Pooling (GAP) pada tiap *channel*, kemudian menggunakan konvolusi satu dimensi dengan kernel adaptif yang mencakup *channel*

tersebut beserta *k channel* tetangga di sekitarnya untuk menangkap interaksi lokal antar *channel*. Bobot channel ω dihitung melalui fungsi aktivasi sigmoid terhadap hasil konvolusi 1D.

$$\omega = \sigma(C1D_k(\mathrm{y}))$$

Rumus 2. 10 Rumus Perhitungan Bobot *Channel*

Keterangan:

- ω: Bobot channel
- $C1D_k$: operasi konvolusi 1D dengan kernel berukuran k
- y: rata-rata nilai dari setiap channel

Kernel k yang bersifat adaptif ditentukan menggunakan Rumus 2.11.

$$k = \psi(C) = \left|\frac{\log_2(C)}{\gamma} + \frac{b}{\gamma}\right|_{odd}$$

Rumus 2. 11 Rumus Menghitung Jumlah Kernel

Keterangan:

- k : jumlah kernel
- $|\ \ |_{odd}$: angka ganjil terdekat
- γ: bernilai 2 [5]
- b: bernilai 1 [5]
- C: dimensi *channel*

2.4 Evaluasi

2.4.1 Root Mean Squared Error

Root Mean Square Error atau yang biasa disingkat RMSE adalah salah satu metode untuk menghitung error yang diprediksi oleh sebuah model. Metode RMSE ini digunakan karena sifatnya yang sensitif terhadap besar atau kecilnya suatu error. Semakin kecil nilai RMSE, maka semakin kecil tingkat kesalahan yang dibuat oleh model dimana nilai RMSE 0

mengindikasikan bahwa model yang dibuat *fit* dengan sempurna [5]. Rumus dari teknik evaluasi RMSE adalah sebagai berikut.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(yi - y)^2}{n}}$$

Rumus 2. 12 Rumus RMSE

Keterangan:

- yi : nilai yang diprediksi oleh model
- y : nilai sesungguhnya
- n : banyak data

Adapun beberapa alasan yang mendasari peneliti dalam memilih teknik evaluasi RMSE adalah sebagai berikut.

1. RMSE mudah untuk dipahami.
2. Berfungsi sebagai heuristik untuk training model.
3. Sederhana secara komputasi.
4. Mudah membedakan data yang menginginkan optimasi algoritma.
5. RMSE cenderung lebih sensitif terhadap kesalahan prediksi yang besar atau *outliers* daripada MSE sehingga dapat memberikan refleksi yang lebih akurat tentang seberapa besar kesalahan tersebut

2.4.2 Mean Squared Error

Mean Squared Error (MSE) adalah ukuran rata-rata dari kuadrat kesalahan antara nilai yang diestimasi dan nilai sebenarnya dalam statistika [5]. Ini digunakan untuk menilai jumlah kesalahan dalam model statistik dan mengukur seberapa dekat garis regresi dengan sekelompok titik. Rumus untuk MSE adalah:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - y1)^2$$

Rumus 2. 13 Rumus MSE

Keterangan:

- yi : nilai yang diprediksi oleh model
- y : nilai sesungguhnya
- n : banyak data

Kelebihan menggunakan MSE adalah:

1. Bagus untuk memastikan bahwa model yang dilatih tidak memiliki prediksi *outlier* dengan kesalahan besar. Hal ini dikarenakan MSE memberikan bobot yang lebih besar pada kesalahan-kesalahan ini.
2. Memiliki grafik yang dapat dihitung turunannya, sehingga cocok digunakan sebagai fungsi kerugian.

2.4.3 Mean Absolute Percentage Error

Mean Absolute Percentage Error atau yang biasa disingkat MAPE adalah salah satu metode untuk menghitung akurasi dari sebuah model [23]. Nilai MAPE biasanya berbentuk persentase. Semakin rendah nilainya, maka semakin tinggi akurasi dari model. Berikut adalah rumus matematika dari MAPE.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y - y1}{y}\right| . 100\ \%$$

Rumus 2. 14 Rumus MAPE

Keterangan:

- yi : nilai yang diprediksi oleh model
- y : nilai sesungguhnya
- n : banyak data

Adapun beberapa alasan yang mendasari peneliti dalam memilih teknik evaluasi MAPE adalah sebagai berikut.

- Seluruh error di normalisasi dalam 1 skala.

- Tidak terjadi masalah error positif dan error negatif yang kemudian saling menetralkan error.

2.4.4 Mean Absolute Error

Mean Absolute Error (MAE) adalah ukuran dari rata-rata ukuran kesalahan dalam kumpulan prediksi, tanpa memperhitungkan arah kesalahan. Ini dihitung sebagai rata-rata dari selisih absolut antara nilai yang diprediksi dan nilai sebenarnya [5]. Rumus untuk MAE adalah:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y - y1|$$

Rumus 2. 15 Rumus MAE

Kelebihan dari metrik ini meliputi:

- Tidak terganggu terhadap *outlier* seperti MSE.
- MAE adalah skor linear, artinya semua perbedaan individual memberikan kontribusi yang sama pada rata-rata
- Mudah dipahami, dapat diinterpretasikan, dan dapat diandalkan untuk menilai akurasi prediksi.

2.5 Tools

2.5.1 Python

Python adalah bahasa pemrograman tingkat tinggi, terinterpretasi, dan bersifat umum yang berfokus pada keterbacaan kode. Diciptakan pada tahun 1991 oleh pengembang Guido Van Rossum, Python secara luas digunakan di banyak organisasi karena mendukung berbagai paradigma pemrograman. Berikut adalah beberapa keunggulan Python:

Keunggulan:

- Mudah dibaca, dipelajari, dan ditulis: Python memiliki sintaksis mirip bahasa Inggris, sehingga lebih mudah dibaca dan dipahami. Sangat mudah diambil dan dipelajari, itulah mengapa banyak orang

merekomendasikan Python untuk pemula. Kode yang perlu ditulis lebih sedikit untuk melakukan tugas yang sama dibandingkan dengan bahasa lain seperti C/C++ dan Java [24].

- Produktivitas yang tinggi: Python adalah bahasa yang sangat produktif. Kesederhanaanya memungkinkan pengembang untuk dapat fokus menyelesaikan masalah tanpa harus menghabiskan waktu terlalu lama memahami sintaksis atau perilaku bahasa pemrograman.
- Fleksibilitas: Python adalah bahasa yang fleksibel dan bersifat umum, sepenuhnya mendukung pemrograman prosedural dan mendukung Object Oriented Programming (OOP). Berkat *package* bawaan dan *library* pihak ketiga, Python cocok untuk berbagai tugas. Python mendominasi di bidang ilmu data dan pembelajaran mesin.
- Komunitas besar dan mendukung: Python memiliki komunitas pengembang yang besar yang berkontribusi pada *library* dan peralatan *open-source* [24].

Namun, Python juga memiliki beberapa kekurangan, seperti:

Kekurangan:

- Performa: Python adalah bahasa terinterpretasi, yang berarti dapat lebih lambat dibandingkan dengan bahasa terkompilasi seperti C atau Java [24].
- Konsumsi memori: Python dapat mengonsumsi banyak memori, terutama ketika bekerja dengan dataset besar atau menjalankan algoritma kompleks.
- Tipe variabel dinamis: Python adalah bahasa tipe dinamis, yang berarti jenis variabel dapat berubah pada saat *runtime*. Ini dapat membuat lebih sulit mencari *error* dan dapat menyebabkan *bug*.

2.5.2 Microsoft Visual Studio Code

Microsoft Visual Studio Code adalah editor kode yang ringan dan lintas-platform. Berikut adalah beberapa keunggulan dari Microsoft Visual Studio Code:

Keunggulan:

- Ringan dan cepat: Visual Studio Code adalah editor kode ringan yang dapat dijalankan dengan cepat dan berjalan lancar, menjadikannya ideal untuk penggunaan sehari-hari [25].
- *Cross-platform*: Visual Studio Code dapat dijalankan di beberapa platform, termasuk Windows, macOS, dan Linux [25].
- Dapat dipersonalisasi: Visual Studio Code sangat dapat disesuaikan, dengan berbagai ekstensi dan tema yang tersedia untuk memenuhi kebutuhan pengguna.
- IntelliSense: Visual Studio Code mendukung fitur IntelliSense, pemahaman dan navigasi kode semantik yang kaya, dan refaktor kode, sehingga memudahkan penulisan kode yang baik.
- *Open-source*: Visual Studio Code sepenuhnya gratis dan bersifat *open-source*, sehingga dapat ditingkatkan oleh siapa saja.