BAB II

TINJAUAN PUSTAKA

II.1 Penelitian Terkait

Berdasarkan penelitian yang telah dilakukan terkait prediksi curah hujan, yang penelitiannya dilakukan oleh (Fitriyanti, 2022) yang mana Proses untuk memprediksi menggunakan sistem Jaringan Syaraf Tiruan dengan menggunakan metode *Backpropagation* , Nilai keluaran berupa perbandingan data aktual yaitu tahun 2017 – 2020 dan data prediksi untuk 2021. Arsitektur terbaik JST yang digunakan untuk metode *Backpropagation* yaitu menggunakan fungsi *aktivasi sigmoid* biner (logsig), yang menggunakan 5 *hidden* layer, dengan *iterasi* sebanyak 1000, dengan algoritma *Levenber-Marquart (trainlm).* Nilai keluaran yang diperoleh dari 3 pos curah hujan di Kabupaten Wajo Sulawesi – Selatan yaitu Sokkalia,, Paria/Majennang, dan Anabanua kemudian dirata ratakan sehingga menghasilkan nilai rata-rata curah hujan prediksi. Data curah hujan tertinggi terdapat pada bulan mei – juni dengan nilai intensitas sebesar 449 mm untuk pos hujan Sakkoli, 401 pada pos Pari/Majenneng dan 594 untuk pos Anabanua yang sesuai dengan data BMKG tahun 2021, sedangkan nilai rata-rata curah hujan bulanan dalam setahun pada pos hujan Sokkalia yaitu 174,875, Paria/Majennang yakni 175,04 mm dan nilai rata-rata curah hujan pada pos hujan Anabanua yakni 210,52 mm. Berdasarkan pola grafik yang diperoleh dengan meng*input* data aktual dan data prediksi diperoleh hasil pola grafik atau trend prediksi mendekati pola grafik dari data aktual. Sehingga dapat disimpulkan bahwa pola curah hujan yang didapatkan dalam penelitian dapat digunakan sebagai pola untuk pemodelan curah hujan untuk tahun selanjutnya dengan arsitektur algoritma JST yang telah dibangun dalam penelitian. Nilai keakuratan untuk data berdasarkan hasil perbandingan nilai validitas berupa nilai *RMSE*, untuk pos hujan Sakkoli, Paria/Majennang dan Anabanua masingmasing memiliki nilai yaitu 0,150., 0,107 dan 0,024. ,

Sehingga, dapat disimpulkan bahwa Jaringan Syaraf Tiruan metode *Backpropagation* , mampu melakukan prediksi curah hujan Kabupaten Wajo, Sulawesi – Selatan dengan cukup baik. Melalui penelitian ini, kita dapat menghasilkan suatu pola prediksi dengan menggunakan sistem Jaringan Syaraf Tiruan metode *Backpropagation* .

Penelitian yang dilakukan (Setti & Wanto, 2018) Tentang *"Analysis of Bacpropagation in Predicting the Most Number of Internet Users in the World"*, Analisa Prediksi peningkatan jumlah pengguna internet, khususnya di Indonesia. Maka perlu dilakukan prediksi untuk tahun-tahun mendatang sehingga pemerintah dapat menyediakan sarana dan pra-sarana yang memadai guna untuk mengimbangi pertumbuhan jumlah pengguna internet dan sebagai langkah antisipasi saat terjadi penurunan jumlah pengguna internet. Data yang digunakan pada penelitian ini fokus pada data jumlah pengguna internet di 25 negara tahun 2013-2017. Algoritma yang digunakan yaitu Jaringan Syaraf Tiruan *Backpropagation* .

Pada tahun 2019 telah dilakukan penelitian (Wong et al., 2019). Mengenai , prediksi tingkat *inflasi* dengan judul penelitian "Prediksi Tingkat Inflasi Dengan Menggunakan Metode *Backpropagation Neural Network*". Analisa prediksi tingkat inflasi di Kota Samarinda, Kalimantan Timur dengan menggunakan metode *Backpropagation* Neural Network (BPNN) telah diimplementasikan. Berdasarkan hasil percobaan, metode BPNN dengan parameter seperti fungsi pembelajaran (trainlm), fungsi aktivasi (logsig, tansig) dan learning rate 0.1 mampu menghasilkan tingkat kesalahan prediksi yang cukup baik dengan nilai *MSE* sebesar 0.00000424. Hal ini menunjukkan bahwa metode BPNN dapat menjadi alternative metode dalam meramalkan tingkat inflasi di Kota Samarinda, Kalimantan Timur. Masukan yang digunakan adalah 99 data dari BPS, dimulai dari periode 4 april 2012 hingga agustus 2019, dengan rincian 70 data pertama digunakan sebagai training data dan 29 data sisanya digunakan sebagai testing data. Untuk hasil *Output*nya dengan menggunakan metode *Backpropagation*

memberikan hasil peramalan yang cukup baik untuk peramalan data Inflasi. Hasilnya dilihat dari nilai *MSE* dan *MAPE* yang cukup kecil.

Pada tahun 2019 telah dilakukan penelitian (Kurniawan et al., 2019) mengenai *implementasi metode Backpropagation* dengan inisialisasi bobot *nguyen widrow* untuk meramalkan harga saham. Proses *implementasi* melalui 3 tahapan, yaitu preprosesing data, pelatihan jaringan, dan pengujian jaringan. Hasil dari penelitian ini menunjukkan bahwa pelatihan jaringan saraf tiruan dengan jumlah dataset yang banyak membutuhkan perhitungan yang kompleks, sehingga jaringan saraf tiruan dengan arsitektur jaringan yang sederhana kurang efektif dan dapat terjebak pada titik lokal minimum. Hasil peramalan untuk harga close saham BBCA.JK memiliki nilai *MAPE* 0,85% dan untuk harga close saham AALI.JK memiliki nilai MAPE sebesar 1,84%.

Pada peneltian yang berjudul "Modified *Backpropagation* with Added White Gaussian Noise in Weighted Sum for Convergence Improvement (Sapkal & Kulkarni, 2018) menggunakan metode Noise Injection dengan tambahan White Gaussian Noise, pada data 8 Signal to noise pada 2 bit iris dataset (iris setosa, iris versicolour, iris virginica), menghasilkan 1. BP = 6050. 2. NNBP = 602.5. Penelitian ini menunjukkan bahwa *Backpropagation* dengan tambahan Noise Net manghasilkan waktu konvergensi lebih sedikit.

Pada Penilitian yang berjudul "Plate Recognition Using *Backpropagation* Neural Network and Genetic Algorithm" (Tarigan et al., 2017), menggunakan metode Genetic Algorithm dengan data 220 Foto Data Plat Nomor Kendaraan dan menghasilkan 1. Genetic Algorithm *Backpropagation* Neural Network (GABPNN): a. Epoch = 2230. b. Training Time = 3 Menit 9 Detik. 2. *Backpropagation* Neural Network (BPNN): a. Epoch = 3530. b. Training Time = 3 Menit 43 Detik. Penelitian ini menunjukkan bahwa GABPNN menghasilkan epoch dan training time lebih sedikit.

Pada Penelitian yang berjudul " Optimization of Smooth Support Vector achine Algorithm Using *Backpropagation* Nguyen Widrow Algorithm in Classification of Mellitus Diabetes Disease" (Dewi et al., 2020), dengan menggunakan metode Smooth Vector Support Machine (SSVM) dan Nguyen Widrow *Backpropagation* dengan menggunakan dataMellitus Diabetes Disease dan Menghasilkan *MSE* dan akurasi sebagai berikut :

1. *MSE* = 0.7569 Akurasi = 81.33%
2. *MSE* = 0. 667235 Akurasi = 96.6%
3. *MSE* = 0. 7521 Akurasi = 82.89%
4. *MSE* = 0. 6228 Akurasi = 97.633%

Pada penelitian yang berjudul " *Nguyen-Widrow* Neural Network for Distribution Transformer Lifetime Prediction " (Rosmaliati et al., 2018), menggunakan metode *Nguyen-Widrow*; learning rate, dengan data Weight and Adaptive Learning Rate. Menghasilkan: *Error* : 0238498338869393, Accuracy : 92%, Hasil : Metode *Nguyen-Widrow* mampu memberikan nilai yang mendekati target.

Adapun rekapitulasi dari penelitian terkait diatas dapat dilihat pada **Tabel II.1** sebagai berikut :

Tabel II.1 Rekapitulasi Penelitian Terkait

No	Judul	Data	Metode	Hasil
1	Aplikasi Jaringan Syaraf Tiruan *Backpropagation* dalam Prediksi Curah Hujan (Fitriyanti, 2022).	Objek: prakiraaan curah hujan bulanan di Kabupaten Wajo, Sulawesi – Selatan *Input*: mengetahui nilai hasil prediksi curah hujan bulanan dengan melihat data prediksi dengan curah hujan aktual dengan	Metode: *Backpropagation* . Proses: pengolahan datanya dilakukan pada *software Octave-7.3.0. RMSE* menggunakan *software Microsoft excel-2010.*	Hasil : nilai R*MSE*, untuk pos hujan Sakkoli, : 0,150 Paria/Majennang : 0,107 Anabanua : 0,024

		menggunakan JST		
2	*Analysis of Backpropagation Algorithm in Predicting the Most Number of Internet Users in the World* (Setti & Wanto, 2018).	Objek: Pengguna Internet di Indonesia *Input*: metode jaringan syaraf tiruan *Backpropagation* untuk meprediksi pengguna internet di dunia	Metode: *Backpropagation* Proses: Fase maju *(feed forward)* fase mundur *(Backpropagation* fase modifikasi bobot	Dengan model arsitektur 3-50-1 dapat melakukan prediksi dengan akurasi 92%.
3	Prediksi Tingkat Inflasi Dengan Menggunakan Metode *Backpropagation Neural Network* (Wong et al., 2019)	Objek : Data tingkat inflasi Kota Samarinda tahun 2012-2017 (72 bulan)	Metode : Metode *Backprogation Neural Network (BPNN)*	datasets : 720 BPNN arsitektur: 4-8-1; learning rate adalah 0.9, epoch adalah 1000. *MSE* : 0.002748. BPNN dengan model arsitektur 5-5-5-1, parameter: fungsi aktivasi trainlm dan learning rate sebesar 0.1 telah menghasilkan nilai *MSE* sebesar 0.00000424 Hasil : dengan nilai *MSE* sebesar 0.00000424. Hal ini menunjukkan bahwa metode BPNN dapat menjadi alternative metode dalam meramalkan tingkat inflasi di Kota Samarinda

4	Implementasi Metode *Bacpropagation* dengan Inisialisasi Bobot *Nguyen Widrow* Untuk Peramalan Harga Saham ((Kurniawan et al., 2019)	Harga Saham	Metode : *Backprogation Neural Network* (*BPNN*)	Hasil dari penelitian ini menunjukkan 1. Peramalan harga closesaham AALI.JK memiliki nilai MAPE : 1.84% 2. Peramalan harga closesaham BBCA.JK memiliki nilai MAPE : 0,85%.
5	"Modified *Backpropagation* with Added White Gaussian Noise in Weighted Sum for Convergence Improvement" (Sapkal & Kulkarni, 2018)	8 Signal to noise pada 2 bit iris dataset (iris setosa, iris versicolour, iris virginica)	Noise Injection dengan tambahan White Gaussian Noise	1. BP = 6050. 2. NNBP = 602.5. Penelitian ini menunjukkan bahwa *Backpropagation* dengan tambahan Noise Net manghasilkan waktu konvergensi lebih sedikit.
6	"Plate Recognition Using *Backpropagation* Neural Network and Genetic Algorithm" (Tarigan et al., 2017)	220 Foto Data Plat Nomor Kendaraan	Genetic Algorithm	1. Genetic Algorithm *Backpropagation* Neural Network (GABPNN): a. Epoch = 2230. b. Training Time = 3 Menit 9 Detik. 2. *Backpropagation* Neural Network (BPNN): a. Epoch = 3530. b. Training Time = 3 Menit 43 Detik. Penelitian ini

				menunjukkan bahwa GABPNN menghasilkan epoch dan training time lebih sedikit.
7	Optimization of Smooth Support Vector Machine Algorithm Using *Backpropagation* Nguyen Widrow Algorithm in Classification of Mellitus Diabetes Disease (Dewi et al., 2020)	Mellitus Diabetes Disease	Smooth Vector Support Machine (SSVM) dan Nguyen Widrow *Backpropagation*	Menghasilkan *MSE* dan akurasi sebagai berikut : 1. *MSE* = 0.7569 Akurasi = 81.33% 2. *MSE* = 0. 667235 Akurasi = 96.6% 3. *MSE* = 0. 7521 Akurasi = 82.89% 4. *MSE* = 0. 6228 Akurasi = 97.633%
8	*Nguyen-Widrow* Neural Network for Distribution Transformer Lifetime Prediction (Rosmaliati et al., 2018)	Distribution Transformer Lifetime Prediction	*Nguyen-Widrow*	Menghasilkan *MSE* dan akurasi sebagai berikut : 5. *MSE* = 0.7569 Akurasi = 81.33% 6. *MSE* = 0. 667235 Akurasi = 96.6% 7. *MSE* = 0. 7521 Akurasi = 82.89% 8. *MSE* = 0. 6228 Akurasi = 97.633%
9	Optimization *Backpropagation* Algorithm Based on Nguyen-Widrom Adaptive Weight and Adaptive	Weight and Adaptive Learning Rate	*Nguyen-Widrow*; learning rate	Menghasilkan : *Error* : 0.0238498338869393 Accuracy : 92% Hasil : Metode *Nguyen-Widrow* mampu

	Learning Rate (Andayani et al., 2017)			memberikan nilai yang mendekati target

II.2

Inflasi

Inflasi memiliki dampak positif dan negatif terhadap perekonomian. Apabila perekonomian suatu negara mengalami suatu kelesuan, maka Bank Indonesia dapat melakukan kebijakan *moneter* yang ekspansif dengan cara menurunkan tingkat suku bunga. Inflasi yang tinggi dan tidak stabil merupakan cerminan dari ketidakstabilan perekonomian yang berakibat pada naiknya tingkat harga barang dan jasa secara umum dan terus menerus, dan berakibat pada makin tingginya tingkat kemiskinan di Indonesia. Karena semakin tinggi tingkat inflasi, maka masyarakat yang awalnya dapat memenuhi kebutuhan sehariharinya dengan adanya harga barang dan jasa yang tinggi tidak dapat memenuhi kebutuhannya sehingga menimbulkan kemiskinan dan tingkat inflasi di Indonesia mengalami fluktuasi dari tahun ke tahun (Salim et al., 2021).

II.3 Peramalan

Peramalan *(forecasting)* bisa disebut juga sebagai suatu teknik atau berupa alat untuk memprediksi atau memperkirakan skala nilai dengan mengenali data masa lalu dan masa depan baik berupa informasi atau data *real*. Peramalan bisa di bilang bagian utama dalam pengembalian keputusan dan perancangan untuk jangka panjang bagi perusahaan atau instansi tertentu (Fitriyanti, 2022).

II.4 Jaringan Syaraf Tiruan

Neural Network / Jaringan Syaraf Tiruan (JST) adalah suatu gagasan model yang mirip dengan ilmu di bidang biologi dalam pemrosesan informasi pada otak manusia. Bagian proses utama pada model ini adalah sebagian besarnya saling terhubung (*neuron*) seperti sel pada otak

manusia yang nantinya akan bekerja sama untuk pemecahan berbagai masalah yang rumit. Cara kerjanya sangatlah mirip dengan manusia belajar dengan melihat contoh. Lapisan-lapisan penyusun JST dibagi menjadi 3, yaitu lapisan *input (input layer),* lapisan tersembunyi *(hidden layer),* dan lapisan *Output (Output layer)*(Wuryandari & Afrianto, 2012).

II.5 *Normalisasi*

Normalisasi data dilakukan karena range nilai *input* tidak sama, yaitu bernilai puluhan hingga ribuan. Persamaan normalisasi yang digunakan secara berturut – turut didefinisikan pada persamaan dibawah ini yang dapat dilihat pada persamaan 2.18 sebagai berikut (Ekojono et al., 2019):

$$x^1 = \frac{x\ -\ min}{max\ -\ min}(0.8) + 0.1$$

Keterangan :

x^1 = Data hasil normalisasi

x = Data yang akan dinormalisasi atau data asli

min = Nilai minimum semua data asli

max = Nilai maximum semua data asli

II.6 *Denormalisasi*

Jika proses *normalisasi* dilakukan, pada akhirnya akan dilakukan juga proses *denormalisasi* untuk mengembalikan data pada range semula, yaitu dengan mencari nilai y. Berikut merupakan rumus denormalisasi yang dapat dilihat pada persamaan 2.19 sebagai berikut (Ekojono et al., 2019):

$$x^{11} = \frac{x^1 - (0.1)}{0.8}(max\ -\ min) + min$$

Keterangan:

x^1 = Data hasil normalisasi

max = Nilai maximum data asli

min = Nilai minimum data asli

x^{11} = Data hasil denormalisasi

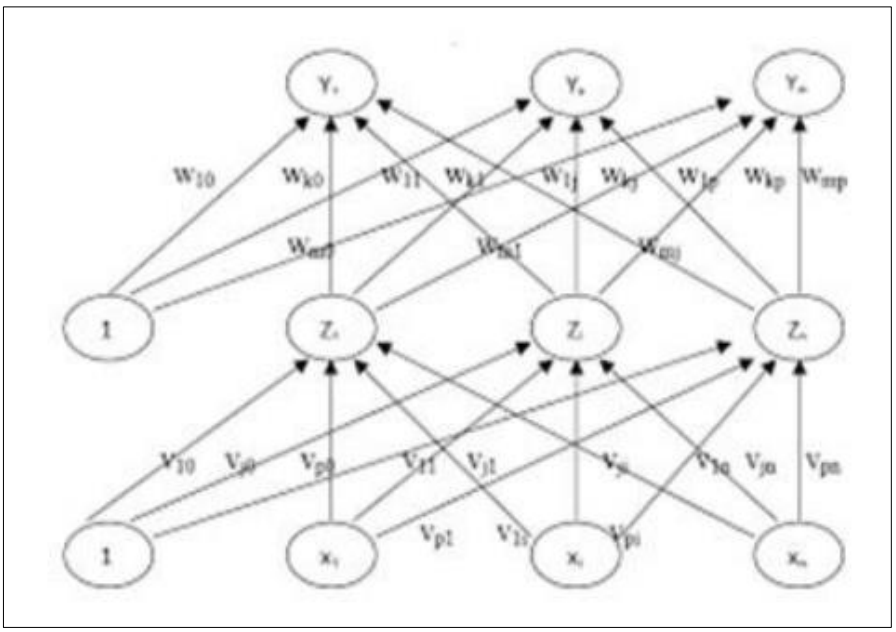
II.7 *Backpropagation*

Metode JST *Backpropagation* (propagasi balik) merupakan metode untuk mengenali suatu pola yang rumit dan menghasilkan pengenalan pola yang sangat baik. *Backpropagation* merupakan salah satu metode yang memiliki lapisan tersembunyi *(hidden layer)* bernilai satu atau lebih dan memiliki proses *feed forward* yang digunakan untuk peramalan dan proses kedua *feed backward* yang digunakan untuk pengecekan *error* yang nantinya akan dilakukan perbaikan. Metode ini memberikan suatu respon terhadap pola *input* atau fitur-fitur data pada proses *learning* yang berfungsi melatih setiap jaringan yang saling terhubung (Setti & Wanto, 2018). Metode ini memiliki proses yang menarik dimana di awali dengan
feed forward terlebih dahulu setelahnya dilakukan proses *feed backward* untuk memperbaiki *error* dari bobot dan biasnya. Proses ini mempunyai syarat dalam memberhentikan proses pelatihan jika batas *error* terpenuhi dan *iterasi* pada mencapai batas yang sudah ditentukan (Aminulloh et al., 2019)

II.8 Arsitektur *Backpropagation*

Backpropagation memiliki beberapa unit *layer* yang saling terhubung dengan di awali *layer input*, layer tersembunyi, dan terakhir *layer Output.* Gambar dibawah adalah arsitektur *Backpropagation* dengan n sebagai jumlah *input* masukan (ditambah bias *input*), sebuah layer tersembunyi yang terdiri dari p unit jumlah dari *neuron hidden layer* (ditambahkan sebuah

bias *Output*) serta m sebagai jumlah unit keluaran. Arsitektur *Backpropagation* dapat dilihat pada **Gambar II.1** sebagai berikut :



Gambar II.1 Arsitektur *Backpropagation*

Vj merupakan bobot dari unit masukan Xi ke unit *layer* tersembunyi Zj ($Vj0$ merupakan bobot garis yang menghubungkan bias di unit masukan ke unit *layer* tersembunyi Zj). Wkj merupakan bobot dari unit *layer* tersembunyi Zj ke unit keluaran Yk ($Wk0$ merupakan bobot dari bias di *layer* tersembunyi ke unit keluaran Zk) (Putri et al., 2021).

Neuron layer terbagi menjadi 3 yaitu :

1. Lapisan *input (Input Layer)*

   Lapisan *input* memiliki fungsi untuk menerima pola data *input*an atau bisa disebut fitur-fitur data yang akan mempengaruhi hasil dari *target (Output).*

2. Lapisan Tersembunyi (*Hidden Layer*)

   Lapisan tersembunyi merupakan unit-unit tersembunyi yang menghubungkan lapisan *input* dan lapisan *Output*.

3. Lapisan *Output (Output Layer)*

Lapisan *Output* merupakan hasil dari proses pemecahan masalah yang berupa target yang akan dicari.

II.9 Fungsi Aktivasi

Dalam *Backpropagation* , fungsi aktivasi yang dipakai harus memenuhi beberapa syarat yaitu : kontinu, terdeferensial dengan mudah dan merupakan fungsi yang tidak turun. Salah satu fungsi yang memenuhi ketiga syarat tersebut sehingga sering dipakai adalah fungsi *sigmoid biner* yang memiliki *range* (0,1). Adapun persamaannya dapat dilihat pada persamaan 2.3 dan 2.4 sebagai berikut :

$f(x) = \frac{1}{1+e^{-x}}$ *dengan turunan* $f'(x) = f(x)(1 - f(x))$ (2.3)

Fungsi lain yang sering dipakai adalah fungsi *sigmoid bipolar* yang bentuk fungsinya mirip dengan fungsi *sigmoid biner*, tapi dengan *range* (-1,1).

$f(x) = \frac{2}{1+e^{-x}}$ dengan turunan $f'(x) = \frac{(1+f(x))(1-f(x))}{2}$ (2.4)

Fungsi *sigmoid* memiliki nilai maksimum = 1. Maka untuk pola yang targetnya > 1, pola masukan dan keluaran harus terlebih dahulu ditransformasi sehingga semua polanya memiliki *range* yang sama seperti fungsi *sigmoid* yang dipakai. Alternatif lain adalah menggunakan fungsi *aktivasi sigmoid* hanya pada *layer* yang bukan *layer* keluaran, pada *layer* keluaran, fungsi aktivasi yang dipakai adalah fungsi identitas : f(x) = x.

II.10 Algoritma *Nguyen-Widrow*

Nguyen-Widrow adalah sebuah algoritma yang digunakan untuk inisialisasi bobot pada jaringan syaraf tiruan untuk mengurangi waktu pelatihan. Algoritma inisialisasi *Nguyen-Widrow* dapat dilihat pada persamaan 2.5 sampai dengan 2.7 sebagai berikut :

1. Set :

   N = Jumlah unit *input*

P = Jumlah unit tersembunyi

β = Faktor skala = $0.7(p)^{1/n} = 0.7\ \sqrt[n]{p}$ (2.5)

2. Untuk setiap unit tersembunyi ($j = 1, \dots, p$)

3. Untuk $i = 1, \dots, n$ (semua unit *input*), $vij(old)$ = bilangan acak antara -0.5 dan 0.5

4. Hitung nilai $\left|\left|v_j(old)\right|\right| = \sqrt{v_1^2 j + v_1^2 ja + \cdots + v_n^2 j}$ (2.6)

5. Hitung bobot baru $= V_{IJ} = \frac{\beta V_{ij}(old)}{||v_j||}$ (2.7)

6. Bias yang dipakai sebagai inisialisasi:

   v_{oj} = bilangan acak antara $-\beta\ dan\ \beta$.

II.11 Pelatihan *Backpropagation*

Proses pelatihan *Backpropagation* memiliki 3 fase. Fase pertama adalah fase maju atau disebut juga fase peramalan. Pada fase ini pola *input* akan dikirim menuju *layer input* dan akan dilakukan perhitungan menuju *hidden layer* dan *terakhir Output layer* dengan menggunakan aktivasi yang ditentukan. Fase kedua adalah fase mundur dengan dilakukan perbandingan hasil *Output* dengan target akan mendapatkan kesalahan yang nantinya akan dilakukan perbaikan pada proses selanjutnya. Fase ketiga adalah perbaikan bobot dan bias untuk menemukan kesalahan yang terjadi.

1. Fase 1 : Propagasi Maju

   Pada proses propagasi maju, lapisan *input* (Xi) dipropagasikan ke *layer* tersembunyi. Hasil keluaran dari setiap unit *layer* tersembunyi (Zj) tersebut selanjutnya dipropagasikan maju lagi ke *Output layer* dengan menggunakan fungsi aktivasi yang ditentukan. Demikian dilakukan secara berulang hingga menghasilkan keluaran jaringan (yk). Berikutnya, keluaran jaringan (yk) dibandingkan dengan target yang harus dicapai (tk). Selisih $tk - yk$ adalah kesalahan yang terjadi. Jika kesalahan ini lebih kecil dari

batas toleransi yang ditentukan, maka *iterasi* dihentikan. Akan tetapi apabila kesalahan masih lebih besar dari batas *error* yang ditentukan, maka bobot sebelumnya akan diperbarui untuk mengurangi kesalahan untuk mendapatkan hasil keluaran dengan batas *error* yang lebih rendah.

2. Fase 2 : Propagasi Mundur

   Berdasarkan kesalahan $tk - yk$, dihitung faktor δk (k = 1,2, … , m) yang dipakai untuk mengirimkan kesalahan di unit yk ke semua unit tersembunyi yang terhubung langsung dengan yk. k juga dipakai untuk memperbarui bobot garis yang terhubung langsung dengan unit keluaran. Dengan cara yang sama, dihitung faktor *j* di setiap unit di *layer* tersembunyi sebagai dasar pengubah yang digunakan untuk faktor menghitung *error* di *hidden layer* dan *Output layer*.

3. Fase 3 : Perubahan Bobot

   Setelah semua faktor dihitung, bobot semua garis dimodifikasi bersamaan. Perubahan bobot didasarkan atas faktor x *neuron* di *layer* atasnya. Sebagai contoh, perubahan bobot garis yang menuju ke *layer* keluaran didasarkan atas yang ada di unit keluaran. Dari ketiga fase yang sudah dijelaskan akan dilakukan perulangan sampai maksimal *iterasi* dan batas *error* yang ditentukan (Mahfuzh et al., 2020).

Berikut algoritma pelatihan *Backpropagation* menurut Seng Hansun (Setti & Wanto, 2018).

1. **Langkah 1 :** Inisialisasi bobot dan bias

   Bobot dan bias dapat di inisialisasi dengan sembarang angka (acak) Dan biasanya terletak antara 0 dan 1 atau -1

2. **Langkah 2 :** Jika kondisi STOP belum terpenuhi, lakukan langkah 3 – 10

3. **Langkah 3 :** Untuk setiap data *training*, lakukan langkah 4 – 9

   Propagasi Maju *(feedforward)*

4. **Langkah 4 :** Setiap unit *input* (Xi, i = 1, … . , n) menerima sinyal *input* xi dan menyebarkan sinyal tersebut ke seluruh unit pada *hidden units*.

   Catatan : sinyal *input* xi yang digunakan adalah *input data training* yang sudah diskalakan. Pertama, dicari nilai terendah dan tertinggi dari *input* data *training.* Kemudian skala yang digunakan tergantung pada fungsi aktivasinya. Jika yang digunakan adalah fungsi *sigmoid Biner*. Yang mempunyai nilai terendah 0 dan tertinggi 1, maka nilai *input* terendah juga dianggap 0 dan tertinggi dianggap 1. Bila fungsi *aktivasi* yang digunakan adalah *Sigmoid Bipolar*, maka *range* nilainya juga bervariasi mulai dari -1 sampai dengan 1.

5. **Langkah 5 :** Pada setiap *hidden unit* (Zj , j = 1, … , p) jumlahkan sinyal-sinyal *input* yang sudah berbobot (termasuk biasnya) yang dapat dilihat pada persamaan 2.8 sebagai berikut :

   $$z\_in_j = v_{oj} + \sum_{i=1}^{n} x_i v_{ij} \qquad (2.8)$$

   Lalu hitung sinyal *Output* dari unit *Output* bersangkutan dengan menggunakan fungsi *aktivasi* yang ditentukan yang dapat dilihat pada persamaan 2.9 sebagai berikut :

   $$z_{j=f\left(z_{in_j}\right)} = \frac{1}{1+\exp\left(-zin_j\right)} \qquad (2.9)$$

   Sinyal *Output* ini selanjutnya dikirim ke seluruh unit pada *Output* jaringan.

   Keterangan :

   z_inj = sinyal masukan pada *hidden layer* ke – j

   voj = bias ke *hidden layer* ke – j (untuk bobot awalnya, nilainya random antara $-\beta \; dan \; \beta$)

   vij = bobot antara unit *input layer* ke – i dan *hidden layer* ke – j (untuk bobot awalnya, nilainya random antara -0,5 dan 0,5)

   x_i = unit *input layer* ke – i

   x_j = aktivasi *hidden layer* ke – j

i = urutan unit *input layer*

j = urutan unit *hidden layer*

Propagasi Mundur *(Backpropagation of error)*

6. **Langkah 6 :** Pada setiap unit *Output* (Yk, K = 1, … , m), jumlahkan sinyal-sinyal *input* yang sudah berbobot (termasuk biasnya) yang dapat dilihat pada persamaan 2.10 sebagai berikut :

$$y_{in_k} = w_{ok} + \sum_{j=1}^{p} z_j w_{jk} \quad (2.10)$$

Lalu hitung sinyal *Output* dari unit *Output* bersangkutan dengan menggunakan fungsi aktivasi yang telah ditentukan pada persamaan 2.11 sebagai berikut :

$$y_k = f(y\_in_k) = \frac{1}{1+\exp(-y\_in_k)} \quad (2.11)$$

Sinyal *Output* ini selanjutnya dikirim ke seluruh unit pada *Output* jaringan

Keterangan :

y_ink = sinyal masukan *Output* ke – k

wok = bias ke *hidden layer* ke – k (untuk bobot awalnya, nilainya random antara -0,5 dan 0,5)

wjk = bobot antar *hidden layer* ke – j dan *Output* ke – k (untuk bobot awalnya, nilainya random antara -0,5 dan 0,5)

yk = aktivasi *Output layer* ke - k

K = urutan unit *Output layer*

m = jumlah maksimum unit pada *Output layer*

p = jumlah maksimum unit pada *hidden layer*

7. **Langkah 7 :** Setiap unit *Output* (Yk, k = 1, … , m) menerima suatu target pattern *(desired Output)* yang sesuai dengan *input training pattern* untuk menghitung kesalahan *(error)* antara target dengan *Output* yang dihasilkan jaringan yang dapat dilihat pada persamaan 2.12 sebagai berikut :

$\delta_k = (t_k - y_k) f'(y_k)$ (2.12)

Faktor δk digunakan untuk menghitung koreksi *error* Δwjk yang nantinya akan dipakai untuk meng-*update* wjk, yang dapat dilihat pada persamaan 2.13 sebagai berikut :

$\Delta w_{jk} = a\delta_k z_j$ (2.13)

Selain itu juga dihitung koreksi bias Δwok yang akan dipakai untuk meng-update wok, yang dapat dilihat pada persamaan 2.14 sebagai berikut :

$\Delta w_{ok} = a\delta_k$ (2.14)

Faktor δk kemudian dikirimkan ke layer yang berada pada langkah ke-8

Keterangan :

δk = faktor koreksi *error* bobot wjk

tk = target *Output* (*desired Output*) ke – k

a = laju percepatan (*learning rate*)

Δwjk = nilai koreksi *error* bias wok

zj = aktivasi *hidden layer* ke - j

8. **Langkah 8 :** Setiap *hidden* unit (Zj , j = 1, … , p) menerima *input* delta (dari langkah ke-7) yang sudah berbobot yang dapat dilihat pada persamaan 2.15 sebagai berikut :

$\delta\_in_j = \sum^{m}_{k=1} \delta_k w_{jk}$ (2.15)

Kemudian hasilnya dikalikan dengan turunan dari fungsi aktivasi yang digunakan jaringan untuk menghasilkan faktor koreksi *error* δj, yang dapat dilihat pada persamaan 2.16 sebagai berikut :

$\delta_j = \delta\_in_j f'(z_j)$ (2.16)

Faktor δj digunakan untuk menghitung koreksi *error* (Δvij) yang nantinya akan dipakai untuk meng-update vij, yang dapat dilihat pada persamaan 2.17 sebagai berikut :

$\Delta v_{ij} = a\delta_j x_i$ (2.17)

Selain itu juga dihitung koreksi bias Δvoj yang akan dipakai untuk meng-update voj, yang dapat dilihat pada persamaan 2.18 sebagai berikut :

$\Delta v_{0j} = a\delta_j$ (2.18)

Keterangan :

δ_in_j = jumlah nilai kesalahan bobot *hidden layer* ke – j

z_j = aktivasi *hidden layer* ke – j

δ_j = faktor koreksi *error* bobot v_{ij}

x_i = unit *input* ke – i

Δv_{ij} = nilai koreksi *error* bobot v_{ij}

Δv_{0j} = nilai koreksi *error* bias v_{0j}

Update Bobot dan Bias (*adjustment*)

9. **Langkah 9 :** Setiap unit *Output* (Y_k , k = 1, … , m) akan memperbarui bias dan bobot dari setiap *hidden* unit (j = 0 , … , p) yang dapat dilihat pada persamaan 2.19 dan 2.20 sebagai berikut :

$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$ (2.19)

$w_{0k}(baru) = w_{0k}(lama) + \Delta w_{0k}$ (2.20)

Demikian pula untuk setiap *hidden unit* (Z_j, j = 1, … , p) akan memperbarui bias dan bobot dari setiap unit *input* (i = 0, … , n) yang dapat dilihat pada persamaan 2.21 dan 2.22 sebagai berikut :

$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$ (2.21)

$v_{0j}(baru) = v_{0j}(lama) + \Delta v_{0j}$ (2.22)

Keterangan :

$w_{jk}(baru)$ = bobot antara *hidden layer* ke – j dan *Output* ke – k baru

$w_{jk}(lama)$ = bobot antara *hidden layer* ke – j dan *Output* ke – k lama

Δw_{jk} = nilai koreksi *error* bobot w_{jk}

$v_{ij}(baru)$ = bobot antara unit *input* ke – i dan *hidden layer* ke – j baru

$v_{ij}(lama)$ = bobot antara unit *input* ke – i dan *hidden layer* ke – j lama

Δv_{ij} = nilai koreksi *error* bobot v_{ij}

10. **Langkah 10 :** Memeriksa kondisi *STOP*

Jika kondisi *STOP* telah terpenuhi, maka pelatihan jaringan dapat dihentikan.

II.12 Mean Squared *Error*

Mean Squared Error (MSE) adalah metode yang digunakan untuk mengevaluasi hasil peramalan. Masing-masing kesalahan dikuadratkan dan rata - rata selisih kuadrat antara nilai yang diramalkan dan yang diamati (Aisyah et al., 2018). Rumus untuk perhitungan *MSE* dapat dilihat pada persamaan 2.23 sebagai berikut :

$$MSE - \sum_{t=1}^{n} \frac{(target-y)^2}{n} \quad (2.23)$$

Keterangan :

MSE = *Mean Square Error*

N = Jumlah Sampel

target = Nilai actual

y = Nilai Prediksi