BAB II
TINJAUAN PUSTAKA

Tinjauan pustaka terdiri dari pencapaian terdahulu dan tinjauan teoritis yang diperoleh peneliti. Penjabaran tinjauan pustaka sebagai berikut ini.

2.1 Pencapaian Terdahulu

Tinjauan pustaka merupakan bagian dari laporan penelitian yang memuat tinjauan terhadap hasil-hasil penelitian terdahulu yang terkait dengan masalah yang diteliti. Hal ini bertujuan untuk memberikan gambaran mengenai pencapaian terdahulu dalam menangani masalah yang sama, serta untuk menjelaskan bagaimana hasil penelitian ini akan memperluas atau mengembangkan pengetahuan yang ada.

Tabel 2. 1. Pencapaian Terdahulu

No	Nama (Tahun)	Judul	Hasil
1	Asthana, S., Pandit, A., & Bhardwaj, A. (2017).	*Analysis of Multiple Hidden Layer vs. Accuracy in Performance using Back Propagation Neural Network*	Bahwa penggunaan multiple hidden layer pada *Back Propagation Neural Network* (BPNN) dapat meningkatkan akurasi dan tingkat pengenalan karakter secara signifikan dibandingkan dengan metodologi lainnya
2	Muh. Ibnu Choldun R., Santoso, J., & Surendro, K. (2019).	*Determining the number of hidden layer and hidden neuron of neural network for wind speed prediction*	Hasil penelitiannya menunjukkan bahwa metode PCA dan K-means clustering berhasil menentukan jumlah lapisan tersembunyi paling efektif.
3	Zair, M., & Jamil, N. (2020)	*Effects of Hidden Layers on the Efficiency of Neural Networks.*	Hasil dari penelitian ini menunjukkan bahwa penggunaan tiga atau kurang hidden layers dalam jaringan saraf dapat mengatasi masalah kompleks atau data besar dengan akurasi terbaik dan kompleksitas waktu yang wajar

4	Chen, G., & Liu, Z. (2022)	*Artificial neural network-based feed-forward and feedback control design and convergence analysis*	Hasil dari jurnal ini menunjukkan bahwa skema kontrol yang diusulkan dapat mencapai tracking yang lebih baik dari trajectory yang diinginkan dan mengurangi efek dari gangguan.

Referensi penelitian pertama yang berjudul "Analysis of Multiple Hidden Layer vs. Accuracy in Performance using Back Propagation Neural Network" yang ditulis oleh Stuti Asthana, Amitkant Pandit, dan Anil Bhardwaj pada tahun 2017. Metode yang digunakan dalam jurnal ini adalah Back Propagation Neural Network (BPNN) untuk menganalisis pengaruh jumlah lapisan tersembunyi (hidden layer) terhadap akurasi dalam pengenalan karakter tulisan tangan. Penelitian ini dilakukan dengan menggunakan numerik kode pos campuran sebagai contoh. Hasil penelitian menunjukkan bahwa dengan menggunakan dua lapisan tersembunyi, tingkat pengenalan sebesar 66,6% dapat dicapai, sedangkan dengan menggunakan lima lapisan tersembunyi, tingkat pengenalan mencapai 100%.

Referensi penelitian kedua dengan judul “Determining the number of hidden layer and hidden neuron of neural network for wind speed prediction” disusun oleh Rachmatullah, M.I.C., Santoso, J., & Surendro, K. tahun 2021. Metode yang digunakan dalam penelitian ini mencakup analisis komponen utama (PCA) dan pengelompokan K-means untuk menentukan jumlah lapisan tersembunyi dan neuron tersembunyi yang optimal dalam jaringan saraf. Penelitian ini menggunakan regresi sebagai fungsi tujuan untuk memprediksi kecepatan angin. Hasil penelitian ini memberikan wawasan yang berharga dalam menentukan topologi jaringan saraf yang optimal untuk prediksi kecepatan angin. Dengan menggunakan metode PCA dan K-means clustering, penelitian ini berhasil menentukan jumlah lapisan tersembunyi dan neuron tersembunyi yang paling efektif dalam memprediksi kecepatan angin. Hasil ini dapat digunakan sebagai panduan dalam pengembangan model jaringan saraf untuk prediksi kecepatan angin di berbagai aplikasi.

Referensi penelitian ketiga dengan judul “Effects of Hidden Layers on the Efficiency of Neural Networks” yang ditulis oleh Muhammad Uzair, Noreen Jamil pada tahun 2020. Metode yang digunakan dalam jurnal ini adalah melakukan

tinjauan terhadap berbagai dampak dari hidden layers pada jaringan saraf. Penelitian ini memberikan gambaran penggunaan tiga jumlah hidden layers yang ditemukan optimal dalam hal mengurangi kompleksitas waktu dan mencapai akurasi yang memadai. Hasil dari penelitian ini menunjukkan bahwa penggunaan tiga atau kurang hidden layers dalam jaringan saraf dapat mengatasi masalah kompleks atau data besar dengan akurasi terbaik dan kompleksitas waktu yang wajar. Namun, jika kriteria yang diinginkan adalah mencapai akurasi 100 persen, pendekatan ini tidak akan berhasil dalam setiap kasus, karena bekerja dengan akurasi 90 persen. Selain itu, penelitian juga menunjukkan bahwa menggunakan satu atau maksimal dua hidden layers dalam jaringan saraf dapat memberikan hasil yang sangat baik dengan data sampel atau data kecil, memberikan akurasi yang sangat baik dalam waktu yang singkat.

Referensi keempat yang berjudul "Artificial Neural Network-Based Feed-Forward and Feedback Control Design and Convergence Analysis" yang tulis oleh Guoshao Chen and Zhiping Liu pada tahun 2018, dengan metode yang digunakan dalam jurnal ini adalah menggabungkan iterative learning control dan artificial neural network untuk mencapai tracking yang lebih baik dari trajectory yang diinginkan dan mengurangi efek dari gangguan. ANN digunakan untuk mensimulasikan plant, dan bobotnya diperbarui secara online untuk konsistensi. Konvergensi dan kekokohan sistem dianalisis, dan simulasi menunjukkan kelayakan dari skema kontrol yang diusulkan. Hasil dari jurnal ini menunjukkan bahwa skema kontrol yang diusulkan dapat mencapai tracking yang lebih baik dari trajectory yang diinginkan dan mengurangi efek dari gangguan. Selain itu, konvergensi dan kekokohan sistem dianalisis dan simulasi menunjukkan kelayakan dari skema kontrol yang diusulkan. Oleh karena itu, jurnal ini dapat menjadi referensi yang baik untuk penelitian lebih lanjut dalam bidang kontrol sistem menggunakan artificial neural network.

2.2 Tinjauan Teoritis

Artificial Neural Network (ANN) adalah jaringan yang menyerupai atau meniru fungsi-fungsi otak manusia untuk menyelesaikan tugas tertentu. Dalam jaringan saraf, terdapat koneksi antara satu neuron dengan neuron lainnya yang memiliki kekuatan tertentu yang disebut sebagai bobot atau synaptic weight.

Keadaan aktif dan non-aktif suatu neuron ditentukan oleh fungsi ambang batas. Konsep perceptron diperkenalkan pada tahun 1958 oleh Frank Rosenblatt, yang merupakan kemampuan untuk belajar dengan jaringan satu lapisan. Batasan dari perceptron adalah jika titik data tidak dapat dipisahkan secara linear, maka perceptron tidak dapat menyelesaikan masalah tersebut. Biasanya, input pada ANN berupa nilai biner, bipolar, dan nilai waktu nyata dari lingkungan." (Manogaran Madhiarasan & Mohamed Louzazni, 2022).

2.2.1 Konsep Dasar Artificial Neural Network

Artificial Neural Network (ANN) atau Jaringan Saraf Tiruan adalah model komputasi yang terinspirasi dari struktur dan fungsi neuron pada sistem saraf manusia. Konsep dasar dari ANN adalah menggabungkan beberapa elemen pengolah sederhana yang disebut "*neuron*" atau "*node*" dalam jaringan yang saling terhubung untuk menghasilkan output yang lebih kompleks. Lapisan *input*, lapisan tersembunyi, dan lapisan *output* adalah tiga jenis lapisan yang membentuk jaringan saraf tiruan. Setiap lapisan terdiri dari beberapa *neuron* atau simpul yang saling terhubung melalui koneksi antar *neuron* atau bobot.

Pada jaringan saraf tiruan, data input pertama-tama diteruskan melalui lapisan input, kemudian melalui lapisan tersembunyi, dan akhirnya menghasilkan output pada lapisan output. Selama proses ini, bobot pada koneksi sinaptik antara setiap *neuron* diatur sedemikian rupa sehingga jaringan dapat mempelajari pola-pola yang ada pada data input dan menghasilkan output yang akurat. Salah satu teknik yang paling sering digunakan pada jaringan saraf tiruan adalah backpropagation, yang merupakan algoritme pelatihan yang memperbarui bobot dan bias pada setiap *neuron* dalam jaringan.

Backpropagation menghitung gradien kesalahan pada output yang dihasilkan oleh jaringan dan mengalirkan gradien kesalahan tersebut mundur melalui jaringan untuk memperbarui bobot dan bias. Proses pelatihan ini berulang-ulang hingga jaringan dapat menghasilkan output yang akurat.

Jaringan saraf tiruan digunakan dalam berbagai aplikasi seperti pengenalan pola, klasifikasi data, prediksi, pengolahan bahasa alami, dan pengenalan suara. ANN telah membuktikan keefektifannya dalam banyak kasus, dan terus

dikembangkan dengan model-model yang lebih kompleks seperti *deep learning* dan *convolutional neural networks.*

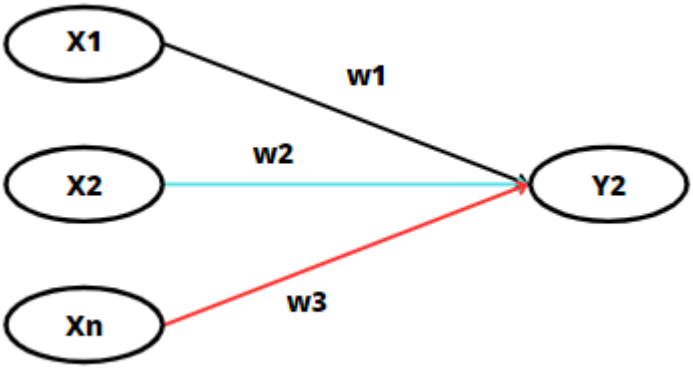
Secara tradisional, jaringan saraf digunakan untuk merujuk pada jaringan atau rangkaian neuron biologis, tetapi penggunaan istilah tersebut saat ini sering mengacu pada Jaringan Saraf Tiruan (Artificial Neural Network, ANN). ANN merupakan model matematika atau komputasi yang terinspirasi oleh sistem saraf biologis, seperti sistem informasi dalam otak. ANN terdiri dari neuron buatan yang saling terhubung dan diprogram untuk meniru sifat-sifat neuron biologis. Neuron-neuron ini bekerja bersama-sama untuk memecahkan masalah tertentu. ANN dikonfigurasi untuk memecahkan masalah kecerdasan buatan tanpa menciptakan model sistem biologis yang sebenarnya. ANN digunakan untuk pengenalan suara, analisis citra, pengendalian adaptif, dan lain sebagainya. Aplikasi-aplikasi ini dilakukan melalui proses pembelajaran, seperti pembelajaran pada sistem biologis, yang melibatkan penyesuaian antara neuron melalui koneksi sinaptik. Hal yang sama terjadi dalam ANN." (Maind, S.B. & Wankar, P., 2014)

2.2.2 Arsitektur Artificial Neural Network

Artificial neural network (ANN) adalah model pemrosesan informasi yang mengambil desain dari cara kerja otak manusia.. ANN dapat dibagi menjadi beberapa arsitektur jaringan, di antaranya adalah jaringan layer tunggal dan jaringan layer jamak.

a. Jaringan layer tunggal, juga dikenal sebagai perceptron, terdiri dari satu layer input dan satu layer output, serta lapisan bobot koneksi di antara keduanya. Jaringan ini digunakan untuk memecahkan masalah klasifikasi biner, di mana keluarannya hanya memiliki dua kemungkinan nilai. Misalnya, membedakan antara gambar yang berisi kucing atau tidak berisi kucing. Keuntungan dari jaringan ini adalah mudah dipahami dan diimplementasikan, serta memiliki waktu komputasi yang cepat. Namun, jaringan ini terbatas dalam menyelesaikan masalah yang kompleks.

   Jaringan yang ditunjukkan pada gambar memiliki n unit input (x1, x2, ..., xn) dan m output (y2).

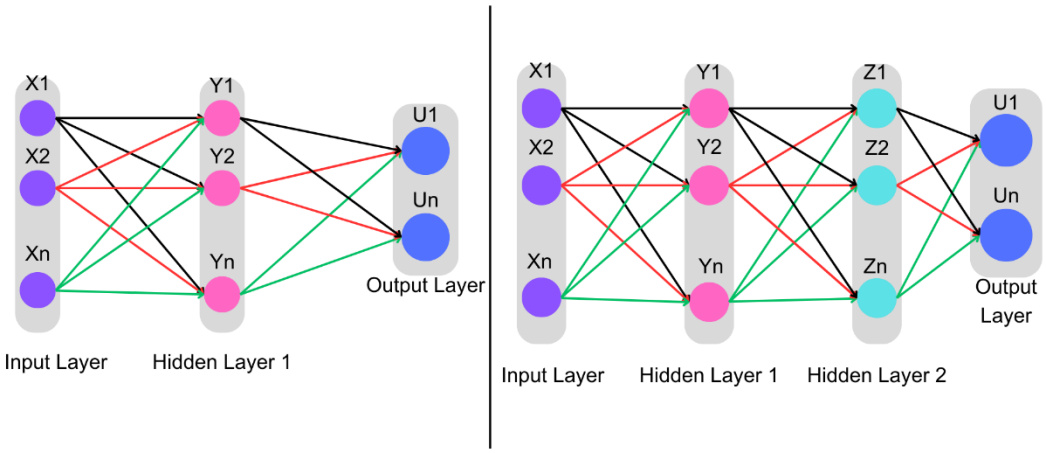


Keterangan:
X..: Neuron Input
Y..: Neuron Output
W..: Bobot

Gambar 2. 1 Jaringan Layer Tunggal

b. Sementara itu, jaringan layer jamak memiliki beberapa layer, termasuk layer input, layer output, dan layer tersembunyi. Masalah kompleks seperti klasifikasi multi-kelas dan prediksi nilai numerik dapat dipecahkan dengan menggunakan jaringan ini. Keuntungan dari jaringan layer jamak adalah kemampuannya dalam menyelesaikan masalah yang kompleks dan mampu mengenali pola yang lebih rumit. Namun, jaringan ini membutuhkan waktu pelatihan yang lebih lama dan dapat mengalami *overfitting*, di mana jaringan terlalu memperhatikan data pelatihan sehingga tidak dapat menggeneralisasi dengan baik pada data baru.

Pada gambar dibawah ini adalah jaringan dengan n unit dari masukan (x1, x2, xn) dan *hidden layer* yang terdiri dari m (y1, y2,...., yn) dan 1 unit keluaran.



Jika gambar berukuran 128 x 128 piksel, maka : Xn = 163824
Jika output adalah 52 huruf kapital dan huruf kecil, maka: Un = 52

Gambar 2. 2 Jaringan dengan Satu dan Dua Lapisan Tersembunyi

Pada tahun 2020, sebuah penelitian oleh Muhammad Faisal dan rekannya membahas penggunaan jaringan layer jamak untuk memprediksi inflasi di Indonesia. Mereka menggunakan jaringan *feedforward* dengan tiga layer, yaitu satu layer input, satu layer tersembunyi, dan satu *layer output*. Hasil penelitian menunjukkan bahwa jaringan tersebut mampu menghasilkan prediksi yang lebih baik daripada model regresi linear. Penelitian lain oleh Mohammad Ali Wardana dan rekannya pada tahun 2021 membahas penggunaan jaringan layer jamak untuk memprediksi harga saham. Mereka menggunakan jaringan dengan dua layer tersembunyi dan berhasil menghasilkan hasil yang lebih baik daripada model regresi linear dan model ARIMA.

Secara keseluruhan, jaringan layer tunggal dan jaringan layer jamak merupakan dua arsitektur ANN yang umum digunakan. Jaringan layer tunggal cocok untuk memecahkan masalah yang sederhana, sedangkan jaringan layer jamak digunakan untuk masalah yang lebih kompleks. Keduanya memiliki kelebihan dan kekurangan masing-masing, sehingga pemilihan jenis jaringan yang tepat tergantung pada jenis masalah yang ingin dipecahkan.

2.2.3 Fungsi Aktivasi

Dalam penelitian oleh Purnomo dkk. (2019, hlm. 96), dijelaskan bahwa fungsi aktivasi berperan penting dalam memetakan masukan pada keluaran yang diinginkan. Setiap neuron dalam jaringan saraf tiruan diberikan fungsi aktivasi, dimana fungsi ini cenderung memiliki karakteristik yang *non-linear*. Sigmoid merupakan salah satu fungsi aktivasi yang paling sering digunakan dalam jaringan saraf tiruan. Fungsi aktivasi menggunakan argumen net masukan, yang merupakan kombinasi linier dari masukan dan bobotnya yang dinyatakan dalam persamaan sebagai berikut:

$$\text{net} = \sum x_i w_i \tag{2.1}$$

dengan fungsi aktivasi

$$F(net) = f(\sum x_i w_i) \tag{2.2}$$

Persamaan yang diberikan menggambarkan bagaimana fungsi aktivasi digunakan dalam jaringan saraf tiruan untuk memetakan masukan ke keluaran yang diinginkan. Fungsi ini mengambil net masukan sebagai argumen, yang didefinisikan sebagai kombinasi linier dari masukan dan bobotnya. Dalam persamaan, simbol xi merepresentasikan masukan ke-i dan wi merepresentasikan bobot yang terhubung dengan masukan ke-i. Persamaan tersebut menghasilkan net masukan, yang kemudian digunakan sebagai argumen dalam fungsi aktivasi. Fungsi aktivasi memproses *net* masukan dan menghasilkan keluaran yang diinginkan, dengan mengaplikasikan sifat *non-linear* pada *net* masukan. Keluaran ini kemudian diteruskan ke lapisan berikutnya di dalam jaringan saraf tiruan, atau sebagai output dari jaringan. Oleh karena itu, persamaan ini memainkan peran penting dalam memahami bagaimana jaringan saraf tiruan bekerja untuk memproses informasi.

Salah satu fungsi aktivasi dalam *artificial neural network* yang paling banyak digunakan yaitu:

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.3}$$

Fungsi aktivasi ANN yang diberikan dalam persamaan diatas, dikenal sebagai *fungsi sigmoid* atau *logistic*. Fungsi ini sering digunakan dalam jaringan

saraf tiruan karena memiliki sifat *non-linear* dan memetakan nilai input ke dalam rentang nilai antara 0 dan 1. Persamaan ini mengambil nilai input x, kemudian memasukkannya ke dalam persamaan eksponensial dengan pangkat negatif dari x. Nilai output fungsi *sigmoid* selalu berada di antara 0 dan 1, dan ketika nilai input x semakin besar, nilai output fungsi akan semakin mendekati 1. Sebaliknya, ketika nilai input x semakin kecil, nilai output fungsi akan semakin mendekati 0. Fungsi *sigmoid* digunakan dalam jaringan saraf tiruan untuk memetakan nilai input ke dalam probabilitas kelas target.

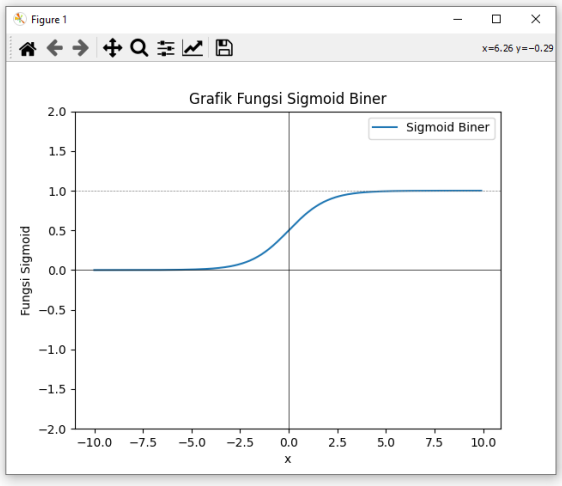
Fungsi *Sigmoid* terbagi menjadi 2, yaitu:

1. Fungsi *Sigmoid Biner*

Fungsi *sigmoid* terdiri dari dua jenis, salah satunya adalah fungsi *sigmoid biner*. Fungsi *sigmoid biner* ini secara khusus dapat menguntungkan dalam penggunaan jaringan syaraf tiruan yang dilatih dengan metode backpropagation, karena hubungan yang sederhana di antara nilai fungsi pada suatu titik mengurangi beban perhitungan selama pelatihan. Fungsi ini seringkali digunakan untuk jaringan syaraf yang membutuhkan nilai output yang terletak pada interval 0 sampai 1. Nilai rangenya berkisar dari 0 sampai 1. (Eka Mala Sari Rochman & Aeri Rachmad, 2021)

$$y = f(x) = \frac{1}{1 + e^{-\sigma x}} \tag{2.4}$$

Dengan $f(x) = \sigma f(x)[1 - f(x)]$



Gambar 2. 3 Fungsi Sigmoid Biner

2. Fungsi *Sigmoid Bipolar*

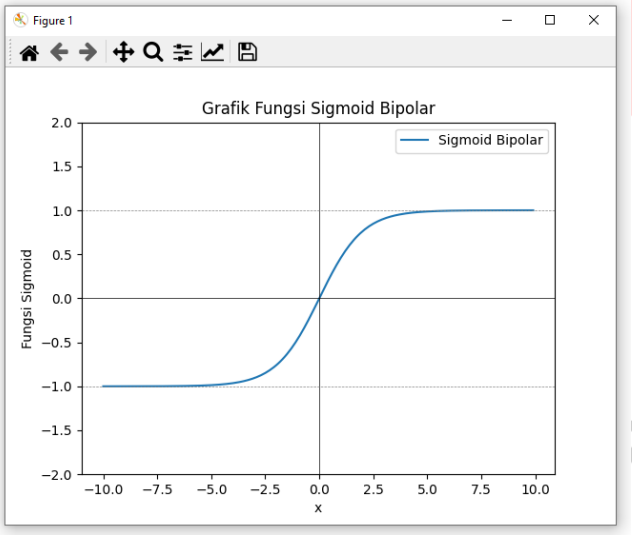
Sigmoid bipolar adalah salah satu jenis fungsi sigmoid yang sering digunakan dalam jaringan saraf tiruan. Berbeda dengan fungsi *sigmoid biner* yang memiliki range nilai antara 0 hingga 1, fungsi *sigmoid bipolar* memiliki *range* nilai antara -1 hingga 1. Fungsi *sigmoid bipolar* adalah sebagai berikut:

$$y = f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{2.5}$$

$$f(x) = \left(\frac{1 - e^{-2x}}{1 + e^{-2x}}\right) \tag{2.6}$$

$$f(x) = \left(\frac{e(x) - e(-x)}{e(x) + e(-x)}\right) \tag{2.7}$$

$$= \left(\frac{1 - e(-2x)}{1 + e(-2x)}\right)$$



Gambar 2. 4 Fungsi Sigmoid Bipolar

2.2.4 *Feed Forward Artificial Neural Network*

Artificial neural network juga merupakan suatu unit dasar dari *feed forward artificial neural network* dalam *neuron* yang formal. Model *neuron* sederhana ini diusulkan oleh Mc Culloch dan Pitts pada tahun 1943. Pembuatan blok sederhana dari jaringan dengan satu lapisan jaringan dengan x sebagai sinyal input

(x1,x2,…..,xi) dan y (y1,y2,…..,yj) merupakan output dari *neuron*, dimana jaringan di sini merupakan perpindahan dari input x ke output y.

Feed forward artificial neural network adalah sebuah jaringan yang terdiri dari titik-titik yang dihubungkan melalui jalur numerik, yang memungkinkan adanya hubungan antara titik-titik tersebut dalam urutan numerik yang lebih tinggi. Dalam penggunaan sebenarnya, jalur-jalur ini dibagi menjadi beberapa tingkatan dan hanya terhubung dengan lapisan yang lebih tinggi. Secara khusus, jaringan ini memiliki *neuron-neuron* yang terhubung dengan neuron lain melalui satu jalur saja. Setiap *neuron* dapat diberi label atau nomor mulai angka kecil pada input sampai angka terbesar pada output, sehingga setiap *neuron* hanya terhubung dengan *neuron* yang mempunyai angka lebih tinggi. Seperti dijelaskan di atas dengan notasi yang lebih sederhana, setiap *neuron* menjumlahkan input dan menambahkan konstanta (bias) untuk membentuk input total dan menerapkan fungsi aktivasi pada input total untuk mendapatkan suatu output. Dimana tiap jalur dihubungkan dengan pembobot (Fauset, 1994:115).

Jaringan pembobotan didasarkan pada perbedaan target dengan hasil *output*. Kesalahan dari *output layer* akan diinteraksikan mundur oleh jaringan dengan "*link weight*". Prosedur iterasi ini akan berulang sampai diperoleh nilai yang konvergen dengan nilai *error* yang minimum.

$$y_k = f_k\left(\alpha_k + \sum_{j\to k} w_{jk} f_j\left(a_j + \sum_{i\to j} w_{ij} x_i\right)\right) \qquad (2.8)$$

dimana :

x : *signal input*

y : signal output

fk : fungsi aktivasi

αk : nilai bias untuk output ke-k

αj : nilai bias untuk hidden node ke-j

$i = 1,2,\dots,l; j = 1,2,\dots,m;$ dan $k = 1,2,\dots,n$

$\sum_{j\to k} w_{jk}$: jumlah bobot dari hidden ke output

$\sum_{i\to j} w_{ij}$: jumlah bobot dari input ke hidden

Artificial neural network adalah sebuah struktur yang terdiri dari satu atau beberapa layer tersembunyi dimana semua sel saraf dalam layer tersebut memiliki fungsi yang serupa, yaitu fh atau f0. Hal ini secara umum menggambarkan arsitektur dari jaringan saraf buatan.

$$y_k = f_k\left(a_k + \sum_{i \to k} w_k x_i + \sum_{j \to k} w_{jk} f_k\left(a_j + \sum_{j \to j} w_j x_i\right)\right) \quad (2.9)$$

Arsitektur *artificial neural network* yang menggunakan fungsi logistik pada output dan memiliki struktur "*skip layer*" dapat dianggap sebagai bentuk regresi logistik non-linier. Jika jumlah output *neuron* sangat banyak, maka arsitektur ini berkaitan dengan "*linked logistic regression*" yaitu sebuah metode regresi logistik yang memiliki kaitan dengan arsitektur *neural network* dengan banyak *output neuron*. Dalam struktur jaringan, setiap neuron memiliki input dan output yang berbeda, dimana input dari seluruh jaringan menjadi input untuk neuron input dan output dari seluruh jaringan menjadi output untuk *neuron output*. Persamaan untuk setiap titik *neuron* dapat dinyatakan sebagai berikut: (2.10)

$$y_j = f_j(x_j)$$

dan

$$x_j = \sum_{i \to j} w_{ij} y_j \quad (2.11)$$

Di sini, y_j adalah output dari *neuron* j, x_j adalah input dari *neuron* j, dan wij adalah bobot antara neuron i dan j. Persamaan tersebut menunjukkan bahwa output y_j dihitung dengan menggunakan fungsi aktivasi $f_j(x_j)$ yang dihitung dari input x_j dan dikalikan dengan bobot yang diterima dari *neuron-neuron* input. Bobot ini ditentukan selama proses pelatihan dengan menggunakan algoritma yang disebut *backpropagation*

Dalam arsitektur jaringan neural, penggunaan fungsi aktivasi dapat berbeda-beda tergantung pada permasalahan yang dihadapi dan teori yang digunakan. Penentuan parameter dan pembobotan serta optimisasi jumlah *layer* dan *neuron*

dalam suatu jaringan merupakan masalah utama dalam penerapannya. Proses *artificial neural network* terdiri dari tiga tahapan utama, yang meliputi langkah *feed forward, backpropagation,* dan pembaruan bobot nilai. Tahap *feed forward* merupakan proses dari input hingga *output* dihasilkan, sementara pada tahap b*ackpropagation*, nilai *output* dari tahap *feed forward* dibandingkan dengan nilai target yang telah ditentukan dan diproses ke belakang hingga layer input untuk mendapatkan nilai *error*. Pada tahap update nilai bobot, nilai bobot di-*update* hingga *error* yang minimal diperoleh.

2.2.5 Feed Forward

Dalam tahap ini, input data dimasukkan ke dalam input layer dan nilai pada *node input* akan disebarkan ke seluruh *node* pada lapisan tersembunyi (kecuali bias). Selanjutnya, nilai tersebut akan diteruskan ke seluruh *output layer* dan diaktivasi menggunakan fungsi aktivasi. Fungsi aktivasi yang digunakan pada penelitian ini adalah fungsi *logistik sigmoid*.

Algoritma proses *feed forward* terdiri dari beberapa langkah, yaitu:

1. Langkah pertama yang akan dilakukan yaitu penentuan inisialisasi bobot, yaitu dengan random atau ditentukan nilainya.
2. Masukan input signal (i = 1,2,….,l) pada node input.
3. Kalikan signal yang masuk dengan nilai bobot link dan berikan hasilnya ke lapisan tersembunyi.
4. Nilai yang diterima oleh hidden node (zj, j = 1,2,….,m) dijumlahkan dan dilakukan suatu proses aktivasi.
5. Lakukan proses seperti tahap 3 dalam penyampaian signal dari lapisan tersembunyi ke lapisan *output*.
6. Bandingkan hasilnya dengan nilai target pada tahap *backpropagation*.

2.2.6 Backpropagation

Tahap *backpropagation* dalam jaringan saraf tiruan adalah proses kalkulasi nilai kesalahan dan penyesuaian bobot agar jaringan dapat memberikan hasil prediksi yang lebih tepat. Tahap ini dimulai dengan membandingkan output

jaringan dengan output yang diharapkan, kemudian menghitung kesalahan antara kedua nilai tersebut. Kemudian, kesalahan tersebut disebarkan kembali ke setiap layer dalam jaringan (dikenal sebagai *backpropagation*) untuk menghitung kontribusi setiap *neuron* terhadap kesalahan dan menyesuaikan bobot agar kesalahan dapat diminimalkan.

Menurut jurnal "Pengaruh Nilai *Hidden Layer* dan *Learning Rate* Terhadap Kecepatan Pelatihan Jaringan Syaraf Tiruan *Backpropagation*" oleh Supriyanto, Sunardi, dan Imam Riadi. (2022), backpropagation adalah sebuah metode pembelajaran yang digunakan untuk memperkecil tingkat error dalam jaringan syaraf tiruan. Algoritma ini bekerja dengan cara menyesuaikan bobot berdasarkan perbedaan antara output yang dihasilkan oleh jaringan dan target yang diinginkan. *Backpropagation* merupakan pengembangan dari *single layer network* menjadi *multilayer network.*

Proses pembelajaran *backpropagation* melibatkan tiga lapisan, yaitu lapisan *input*, lapisan tersembunyi *(Hidden Layer)* dan lapisan *output.* Lapisan *Input* menerima data masukan, lapisan tersembunyi berfungsi untuk memproses informasi secara internal, dan lapisan *output* menghasilkan *output* yang diinginkan. Dalam proses pembelajaran, bobot antar *neuron* diperbarui berdasarkan perbedaan antara output yang dihasilkan dan target yang diinginkan.

Dengan menggunakan algoritma *backpropagation,* jaringan syaraf tiruan dapat belajar dan mengoptimalkan bobotnya sehingga dapat menghasilkan output yang mendekati target yang diinginkan. Hal ini memungkinkan jaringan syaraf tiruan untuk digunakan dalam berbagai aplikasi seperti pengenalan pola, prediksi, dan pengolahan data.

2.2.7 Pemilihan Bobot dan Bias Awal

Pilihan bobot awal memengaruhi pencapaian jaringan ke titik minimum lokal dan global serta kecepatan konvergensinya. Untuk menghindari bobot yang menghasilkan nilai turunan aktivasi yang kecil, disarankan untuk menghindari perubahan bobot yang sangat kecil. Demikian pula, bobot awal tidak boleh terlalu

besar agar nilai turunan fungsi aktivasi target tidak terlalu kecil. Algoritma backpropagation menggunakan jumlah acak untuk memulai bias dan bobot.

2.2.8 Metode Optimasi Error

Optimasi error adalah suatu metode yang digunakan untuk meminimalkan atau mengoptimalkan kesalahan (error) dalam suatu sistem atau model. Tujuannya adalah untuk mencapai performa yang lebih baik atau mendekati nilai yang diinginkan. Metode optimasi error dapat berbeda-beda tergantung pada konteks dan jenis sistem yang sedang dipelajari. Salah satu metode umum yang digunakan dalam optimasi error adalah *Gradient Descent*. *Gradient descent* adalah metode optimasi yang umum digunakan dalam *machine learning* dan *deep learning*. Metode ini berdasarkan pada perhitungan gradien fungsi error terhadap parameter model. Dengan menggunakan gradien, metode ini mencoba untuk menemukan minimum lokal atau global dari fungsi error.

2.2.9 Pengenalan Huruf Alfabet

Pengenalan huruf alfabet merupakan suatu bentuk pengenalan pola pada sistem komputer untuk mengenali huruf-huruf pada abjad bahasa tertentu. Tujuan dari pengenalan huruf alfabet adalah untuk memungkinkan komputer untuk mengidentifikasi, mengklasifikasikan, dan memproses teks dengan lebih akurat dan efisien. Dengan menggunakan pengenalan huruf alfabet, sistem komputer dapat memahami dan memproses teks dengan cara yang mirip dengan cara manusia memahami dan memproses tulisan.

Terdapat berbagai metode yang telah digunakan dalam pengenalan huruf alfabet, di antaranya adalah metode *template matching*, metode *decision tree*, metode *hidden Markov models* (HMM), dan *metode neural network*. Salah satu metode yang sering digunakan dalam pengenalan huruf alfabet adalah metode neural network, khususnya *artificial neural network* (ANN).

Dataset yang digunakan dalam penelitian pengenalan huruf alfabet dapat berupa gambar huruf yang telah dikumpulkan dari berbagai jenis *fonts*. Dalam memilih dataset, perlu diperhatikan bahwa dataset yang digunakan harus mencakup variasi huruf yang cukup sehingga dapat merepresentasikan keseluruhan huruf pada abjad bahasa tertentu.

Kriteria pemilihan dataset meliputi jumlah data, variasi data, kualitas data, dan kesesuaian data dengan tujuan penelitian.

Dalam melakukan pengenalan huruf alfabet menggunakan ANN, langkah pertama yang dilakukan adalah melakukan *preprocessing* pada data gambar huruf, seperti pengubahan ukuran, penajaman gambar, dan konversi gambar ke dalam bentuk matriks. Selanjutnya, data gambar huruf yang telah diproses tersebut digunakan sebagai input pada model ANN. Model ANN kemudian dilatih menggunakan algoritma pembelajaran tertentu dengan menggunakan data *training*. Setelah model ANN telah dilatih, model tersebut dapat digunakan untuk melakukan pengenalan pada data uji.

Dalam penelitian tentang pengenalan huruf alfabet dengan ANN, salah satu fokus penelitian adalah pada pengujian akurasi model ANN dengan menggunakan satu dan dua lapisan tersembunyi. Pengujian ini bertujuan untuk mengetahui apakah penambahan lapisan tersembunyi pada model ANN dapat meningkatkan akurasi pengenalan huruf alfabet atau tidak.

Dalam melakukan pengujian akurasi model ANN, data yang digunakan dibagi menjadi dua bagian, yaitu data training dan data testing. Data *training* digunakan untuk melatih model ANN, sedangkan data testing digunakan untuk menguji akurasi model ANN. Pada pengujian ini, perlu dilakukan evaluasi performa model ANN dengan menggunakan metrik evaluasi seperti *accuracy*, *precision*, *recall*, dan *F1-score*.

Dalam rangka meningkatkan akurasi model ANN dalam pengenalan huruf alfabet, perlu dilakukan optimasi parameter pada model ANN seperti memilih jumlah *hidden layers* yang tepat, menentukan jumlah *neuron* pada setiap lapisan tersembunyi, dan memilih algoritma pembelajaran yang sesuai dengan data. Selain itu, perlu juga dilakukan peningkatan kualitas dataset dengan melakukan augmentasi data atau pengambilan data dari berbagai sumber yang berbeda.

2.2.10 Hidden Layers ANN

Artificial Neural Network (ANN) terdiri dari lapisan *input*, lapisan *output* dan satu atau lebih lapisan tersembunyi. Lapisan tersembunyi merupakan bagian

tersembunyi dari ANN yang menghubungkan lapisan *input* dengan lapisan *output*, dan berfungsi untuk memproses informasi dalam ANN.

Setiap lapisan tersembunyi terdiri dari sejumlah node, dan setiap *node* memiliki bobot yang terhubung dengan node di layer sebelumnya dan layer setelahnya. Ketika informasi masuk ke lapisan tersembunyi, setiap node melakukan perhitungan terhadap informasi tersebut dengan menggunakan bobotnya. Selanjutnya, informasi tersebut akan diproses oleh *node* lain pada lapisan tersembunyi tersebut atau diteruskan ke lapisan *output*.

ANN dengan satu lapisan tersembunyi disebut dengan *Single Layer Perceptron* (SLP), sedangkan ANN dengan dua atau lebih lapisan tersembunyi disebut dengan *Multi-Layer Perceptron* (MLP). SLP biasanya digunakan untuk masalah klasifikasi yang sederhana dan memiliki struktur yang sederhana pula. Sementara itu, MLP digunakan untuk masalah klasifikasi yang lebih kompleks dan memiliki struktur yang lebih kompleks pula.

Kelebihan dari menggunakan dua atau lebih lapisan tersembunyi pada MLP adalah kemampuan untuk memproses informasi yang lebih kompleks dan memiliki kemampuan untuk menghasilkan output yang lebih akurat. Namun, penggunaan dua atau lebih lapisan tersembunyi juga akan memperpanjang waktu proses *training* dan memperbesar kompleksitas model.

2.2.12 Library

Library pada pemrograman *Python* merupakan kumpulan modul atau fungsi yang telah dikodekan sebelumnya untuk memudahkan pengembangan program. *Library* ini berisi berbagai macam fungsi dan modul untuk melakukan tugas tertentu seperti manipulasi data, visualisasi, pengolahan citra, atau pemrosesan bahasa alami. *Library* dalam *Python* sangat membantu karena dapat menghemat waktu dan usaha dalam pengembangan aplikasi dan mempercepat proses pembuatan aplikasi. Beberapa contoh *library* yang populer dalam *Python* adalah *NumPy, Pandas, Matplotlib*. Berikut adalah beberapa *library* yang digunakan:

A. *Numpy*

Numpy adalah *library Python* untuk melakukan perhitungan numerik. Ini menawarkan implementasi *array* dan matriks yang efektif serta sejumlah fungsi matematika untuk digunakan pada mereka. Fabio Nelli (2018) dalam bukunya *"Python Data Analytics With Pandas, NumPy, and Matplotlib"* memperkenalkan *NumPy* sebagai inti dari banyak pustaka *Python* lainnya yang berasal darinya. Nelli menjelaskan bahwa *NumPy* merupakan pustaka yang menyediakan struktur data dan fungsi berperforma tinggi yang tidak dapat disediakan oleh paket dasar *Python.* Dalam buku ini, *NumPy* mendefinisikan struktur data khusus yang disebut ndarray, yaitu array N-dimensi.

Nelli menekankan pentingnya pengetahuan tentang *NumPy* dalam perhitungan numerik karena penggunaannya yang benar dapat sangat mempengaruhi kinerja komputasi. *NumPy* menawarkan beberapa fitur yang tidak ada pada *Python* standar. Pertama, *ndarray* merupakan array multidimensi yang lebih cepat dan efisien daripada array yang disediakan oleh paket dasar *Python*. Kedua, *NumPy* menyediakan kumpulan fungsi untuk melakukan perhitungan elemen antara array serta operasi matematika lainnya antara array. Selain itu, *NumPy* juga menyediakan alat-alat untuk membaca dan menulis dataset yang disimpan di hard disk.

B. *Pandas*

Pandas adalah pustaka *Python* untuk analisis dan manipulasi data yang menawarkan struktur data yang kuat untuk menyimpan dan memproses data serta berbagai fungsi untuk mengubah dan memanipulasi data. Fabio Nelli (2018) dalam bukunya "*Python Data Analytics With Pandas, NumPy, and Matplotlib*" memperkenalkan *Pandas* sebagai pustaka yang menyediakan struktur data kompleks dan fungsi yang dirancang khusus untuk memudahkan, mempercepat, dan efektif dalam bekerja dengan data. *Pandas* merupakan inti dari analisis data di *Python*. Oleh karena itu, mempelajari dan mengaplikasikan pustaka ini menjadi tujuan utama yang akan dikerjakan sepanjang buku (terutama dalam Bab 4, 5, dan 6). Pengetahuan mendalam tentang setiap detail *Pandas*, terutama saat diterapkan dalam analisis data, adalah tujuan utama dari buku ini.

Konsep dasar dari *Pandas* adalah *DataFrame*, struktur data tabular dua dimensi dengan label baris dan kolom. Pandas mengaplikasikan sifat kinerja tinggi dari pustaka *NumPy* dalam manipulasi data dalam *spreadsheet* atau basis data relasional (basis data SQL). Dengan menggunakan indeks yang canggih, Pandas memudahkan untuk melakukan berbagai operasi pada jenis struktur data ini, seperti perubahan bentuk, pemotongan, agregasi, dan pemilihan subset data.

C. *Matplotlib*

Sub-library dari paket matplotlib, yang menawarkan berbagai fungsi untuk membuat visualisasi dengan *Python*, adalah matplotlib.pyplot. Modul matplotlib.pyplot dapat lebih mudah dirujuk dalam kode dengan menggunakan sintaks as plt untuk membuat alias plt untuk modul tersebut. Fabio Nelli (2018, hal. 231-233) menjelaskan bahwa matplotlib merupakan sebuah pustaka Python yang mengkhususkan diri dalam pengembangan grafik dua dimensi (termasuk grafik tiga dimensi). Dalam beberapa tahun terakhir, matplotlib telah banyak digunakan dalam lingkungan ilmiah dan rekayasa. Beberapa fitur yang membuat matplotlib menjadi alat yang paling banyak digunakan dalam representasi grafik data antara lain:

1. Kemudahan penggunaan yang sangat tinggi
2. Pengembangan secara bertahap dan visualisasi data interaktif
3. Ekspresi dan teks dalam *LaTeX*
4. Kontrol yang lebih besar terhadap elemen grafik
5. Ekspor ke banyak format, seperti *PNG, PDF, SVG, dan EPS.*

Matplotlib dirancang untuk mereproduksi lingkungan yang sebanyak mungkin mirip dengan *MATLAB* dalam hal tampilan grafis dan bentuk sintaksisnya. Pendekatan ini terbukti sukses karena dapat mengambil manfaat dari pengalaman perangkat lunak *MATLAB* yang telah beredar selama beberapa tahun dan kini luas digunakan dalam lingkaran profesional teknis-sains. Tidak hanya itu, matplotlib berdasarkan pada skema yang dikenal dan cukup familiar bagi sebagian besar ahli di bidang tersebut, dan juga mengoptimalkan penggunaannya selama bertahun-tahun sehingga menjadikan pustaka ini pilihan yang sangat baik bagi mereka yang baru mengenal visualisasi data, terutama bagi mereka yang tidak

memiliki pengalaman dengan aplikasi seperti *MATLAB* atau sejenisnya.

D. *Tqdm*

Tqdm adalah modul *Python* untuk membuat bilah progres dan bentuk indikator progres lainnya. Fungsi *tqdm* diimpor dari modul tqdm menggunakan sintaks *from tqdm import tqdm*.

E. *Pickle*

Pickle paket *Python* digunakan untuk membuat serialisasi dan de-serialisasi objek *Python* menjadi aliran *byte* yang dapat dikirim melalui jaringan atau disimpan di disk. Modul *pickle*, yang menawarkan fungsi untuk serialisasi dan de-serialisasi objek *Python*. Fabio Nelli (2018, hal. 168) menjelaskan bahwa modul *pickle* dalam Python mengimplementasikan algoritma yang kuat untuk serialisasi dan deserialisasi struktur data yang diimplementasikan dalam Python. Pickling adalah proses di mana hierarki suatu objek dikonversi menjadi aliran byte.

F. *Tkinter*

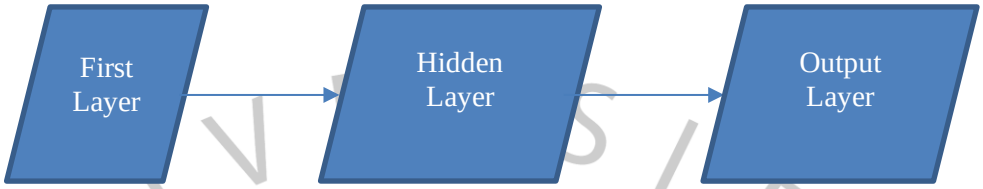
Pustaka *Python* bawaan yang umum untuk antarmuka pengguna grafis (GUI) adalah *Tkinter*. Ini menawarkan kumpulan utilitas untuk menambahkan jendela, kotak dialog, tombol, label, kotak teks, menu, dan elemen visual lainnya ke aplikasi *Python*. *Toolkit* antarmuka pengguna grafis lintas *platform* yang dibangun dalam bahasa pemrograman Tcl, toolkit Tk adalah pembungkus untuk *Tkinter*. *Tkinter* menawarkan antarmuka yang mudah dan ramah pengguna untuk membuat program GUI di beberapa sistem operasi.

G. *Artificial Neural Network*

Artificial Neural Network (ANN), merupakan salah satu jenis metode *machine learning*. Itu terdiri dari jaringan neuron yang saling berhubungan yang diatur berlapis-lapis. Setiap *neuron* mengambil informasi dari *neuron* lain, memprosesnya, dan kemudian mengirimkan kembali informasi ke *neuron* lain di lapisan di bawahnya. Bobot dan bias neuron diubah selama fase pelatihan *Artificial Neural Network* (ANN) untuk mengurangi perbedaan antara keluaran yang diproyeksikan dan keluaran sebenarnya. *Backpropagation*, sebuah metode yang

menentukan gradien dari fungsi kerugian sehubungan dengan bobot dan bias, sering digunakan untuk mencapai hal ini.

H. *One Hidden Layer* / Lapisan Tersembunyi



Gambar 2. 5 Ilustrasi satu lapisan tersembunyi

Setiap node dalam *Artificial Neural Network* (ANN) satu lapis terhubung ke setiap karakteristik input, menjadikannya jenis arsitektur jaringan saraf tertentu. Lapisan ini, juga disebut sebagai lapisan tersembunyi, mengubah data masukan secara nonlinier untuk menghasilkan keluaran. Keluaran utama jaringan dibuat dengan mengumpankan keluaran lapisan tersembunyi ke dalam lapisan keluaran. *Artificial Neural Network* (ANN) satu lapis, sering menangani tugas klasifikasi sederhana, seperti klasifikasi biner atau masalah klasifikasi multi-kelas dengan fitur input minimal.

Rumus:

Dalam arsitektur model satu lapisan, kami telah mengevaluasi rumus berikut

a) Untuk propagasi maju untuk arsitektur satu lapis:

$$z = w^{*}x + b$$
$$a = g(z) \qquad (2.12)$$

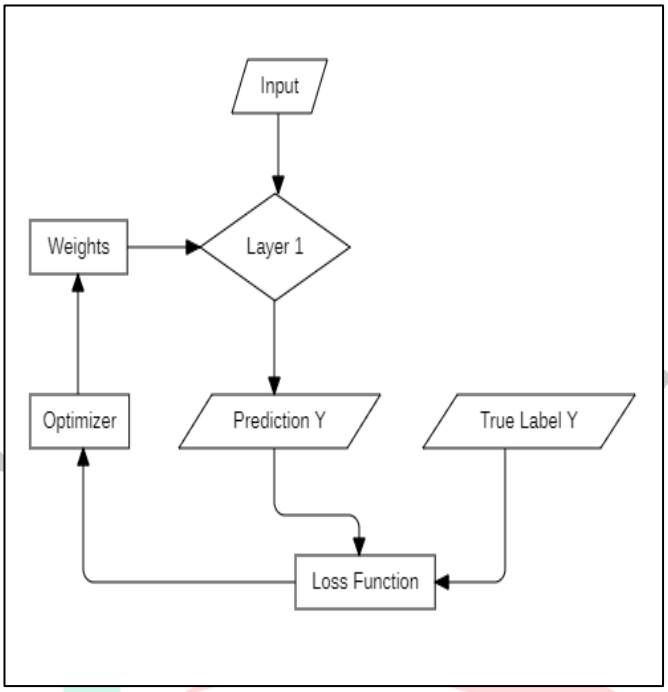
di mana "w" adalah bobot, "x" adalah input lapisan, "b" adalah vektor bias dan "g" adalah fungsi aktivasi.

b) Untuk *backpropagatio*n untuk arsitektur satu layer:

$$\partial L/\partial w = \partial L/\partial y * \partial y/\partial z * \partial z/\partial w \qquad (2.13)$$

di mana "y" adalah keluaran dari *neuron* atau lapisan, "z" adalah jumlah bobot dari masukan ke *neuron* atau lapisan, dan "∂" adalah operator turunan

parsial. "L" adalah fungsi kerugian.



Gambar 2. 6 Flowchart ANN Satu Lapisan Tersembunyi

Dalam *flowchart* ini, kami memiliki gambar masukan. Vektor gambar ini dilewatkan melalui layer1 dan kerugian dievaluasi melalui label yang diprediksi dan benar. Berdasarkan hal ini, kami akan memperbarui bobot untuk menghindari prediksi yang salah dan mengurangi kerugian (*loss)*.

I. *Two Hidden Layer* / Dua Lapisan Tersembunyi

Kami telah menggunakan optimasi penurunan gradien untuk membangun *Artificial Neural Network* (ANN) dua lapis. Dataset pelatihan dan pengujian, kecepatan pembelajaran, jumlah zaman, dan ukuran lapisan tersembunyi semuanya adalah input untuk jaringan saraf. Daftar yang disebut Ukuran Lapisan Tersembunyi, yang memiliki dua item yang mewakili jumlah neuron di setiap lapisan tersembunyi, digunakan untuk menentukan lapisan tersembunyi.

Neural network kemudian dilatih menggunakan pengoptimalan penurunan gradien pada data pelatihan, dan akurasi model dihitung menggunakan data dari dataset pelatihan dan pengujian. Selain itu, 'model_2_layers.pickle' digunakan untuk menyimpan data pelatihan dan pengujian serta parameter model yang dilatih.

Keakuratan pelatihan dan pengujian serta hilangnya keseluruhan model yang dipelajari termasuk dalam keluaran kode. *Final loss* merupakan nilai numerik yang merepresentasikan error model pada *training epoch* terakhir, sedangkan akurasi *training* dan *testing* disajikan sebagai nilai persentase.

Rumus:

Dalam model arsitektur dua lapis kami telah mengevaluasi rumus berikut

c) Untuk propagasi maju untuk arsitektur dua lapis:

Lapisan 1

$$Z1 = w1 * x1 + b1$$
$$\mathrm{A1} = \mathrm{g(z1)} \quad (2.14)$$

di mana "w" adalah bobot, "x" adalah input lapisan, "b" adalah vektor bias dan "g" adalah fungsi aktivasi.

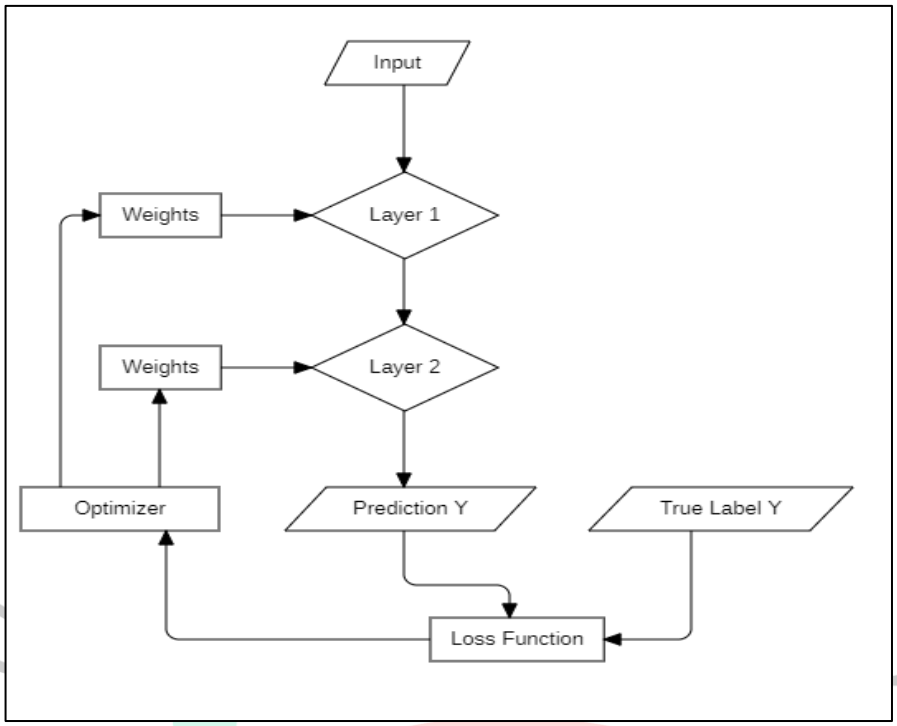
Lapisan 2

$$Z2 = w2 * x2 + b2$$
$$\mathrm{A2} = \mathrm{g(z2)} \quad (2.15)$$

d) Untuk *backpropagation* untuk arsitektur satu layer:

$$\partial \mathrm{L}/\partial \mathrm{w} = \partial \mathrm{L}/\partial \mathrm{y} * \partial \mathrm{y}/\partial \mathrm{z} * \partial \mathrm{z}/\partial \mathrm{w} \quad (2.16)$$

di mana "y" adalah keluaran dari *neuron* atau lapisan, "z" adalah jumlah bobot dari masukan ke *neuron* atau lapisan, dan "∂" adalah operator turunan parsial. "L" adalah fungsi kerugian (*Loss)*.



Gambar 2. 7 Flowchart ANN Dua Lapisan Tersembunyi

Dalam diagram alir ini, kami memiliki gambar masukan, vektor gambar ini melewati layer1, layer2 dan kerugian dievaluasi melalui label yang diprediksi dan benar sesuai dengan ini kami akan memperbarui bobot dari lapisan satu dan lapisan dua untuk menghindari prediksi yang salah dan mengurangi kerugian (*Loss*). Untuk memperbarui bobot, kami akan menggunakan pengoptimal untuk mengurangi kerugian dan meningkatkan akurasi model.