BAB 2
LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Pada bagian ini dijelaskan referensi yang digunakan dalam menyelesaikan penelitian ini. Referensi yang dipilih adalah referensi tentang permasalahan yang dibahas dan juga tentang metode yang digunakan. Masalah yang dibahas adalah penentuan durasi nyala lampu lalu lintas dan metode yang digunakan adalah metode *backpropagation*.

Penelitian pertama adalah penelitian dari Yohannes et al. (2015). Penelitian ini memiliki masukan berupa tingkat inflasi kota Malang pada tahun 2005-2014. Inflasi sendiri memiliki 8 kategori. Kategori-kategori ini dijadikan nilai masukan, sehingga masukan dari *backpropagation* ada 8. Keluaran sistem berupa jumlah upah minimum kota malang. Dalam penelitian ini terdapat 3 variabel yang diuji, yaitu *learning rate*, jumlah neuron *hidden* layer, dan jumlah iterasi. Pengujian ini bertujuan untuk mendapatkan nilai optimal di ketiga variabel tersebut dan diharapkan nilai *Root Mean Square Error* (*RMSE*) sistem juga optimal. Penelitian kedua adalah penelitian dari Wicaksana et al. (2014). Penelitian ini menggunakan 2 metode gabungan dari logika *fuzzy* dan *neural network*. Logika *fuzzy* dilakukan untuk menentukan waktu lampu hijau dan neural network untuk proses pelatihannya. Algoritme dari *neural network* yang dipakai adalah algoritme *Levenberg-Marquardt Backpropagation.* Penelitian ketiga adalah penelitian dari Royani et al. (2013). Penelitian ini diuji dengan menggunakan simulasi persimpangan pada MATLAB 7.8 dan Toolbox Algoritme Genetika. Pengujian ini nantinya akan dibandingkan dengan sistem yang tidak menggunakan *Fuzzy Neural Network*.

Tabel 2.1 Kajian Pustaka

Judul	Masukan	Metode	Keluaran
		Proses	**Analisis**
Penentuan Upah Minimum Kota Berdasarkan Tingkat Inflasi Menggunakan Backpropagation Neural Network (BPNN) (Yohannes, et al., 2015)	• Data tingkat inflasi.	*Backpropagation.*	• Upah minimum kota.
		• Normalisasi Data. • Prose algoritme Backpropagation. • Denormalisasi Data.	• Nilai RMSE sebesar 0.0728.

Tabel 2.1 Kajian Pustaka

Judul	Masukan	Metode Proses	Keluaran Analisis
Sistem Pengambilan Keputusan Waktu Perpindahan Lampu Lalu Lintas Menggunakan Metode Neuro-Fuzzy Pada Sistem Tranportasi Cerdas (Wicaksana, et al., 2014)	• Jumlah Kendaraan.	Neuro *Fuzzy.*	• Durasi nyala lampu hijau.
		• Pelatihan *Neural Network.* • Pengujian Nilai *Error (RMSE).* • Jika RMSE < 10^{-6}, maka data masuk ke metode *fuzzy.*	• Dengan Menggunakan neuro fuzzy, kendaraan yang tidak berhasil dilewatkan sebesar 2.93%. • Tanpa menggunakan neuro fuzzy, kendaraan yang tidak berhasil dilewatkan sebesar 12.71%. • Nilai RMSE sebesar 0.00452.
Control of Traffic Light in Isolated Intersections Using *Fuzzy Neural Network and Genetic Algorithm* (Royani, et al., 2013)	• Antrian kendaraan pada fase lampu .hijau saat ini • Antian Kendaraan pada fase lampu merah saat ini. • Kendaraan yang mendekati fase lampu hijau saat ini.	Fuzzy *Neural Network (FNN)* dan Algoritme Genetika.	• Waktu tunggu setiap ruas jalan.
		• Memodelkan *FNN.* • Pelatihan *FNN* dengan Algoritme Genetika.	• Rata-rata waktu tunggu kendaraan tanpa menggunakan *FNN* adalah 330 detik. • Rata-rata waktu tunggu kendaraan menggunakan *FNN* adalah 178 detik.

Dari Tabel 2.1 dapat dilihat bahwa ketiga penelitian berhasil diterapkan. Pada penelitian Wicaksana et al. (2014) Dan Royani et al. (2013) dapat disimpulkan bahwa jaringan syaraf tiruan yang digabung dengan algoritme fuzzy memiliki hasil yang lebih baik dari pada menggunakan jaringan syaraf tiruan murni. Tetapi pada penelitian Yohannes et al. (2015) dapat dilihat bahwa jaringan syaraf tiruan murni yaitu *Backpropagation Neural Network (BPNN)* juga bisa diterapkan untuk

masalah penentuan upah minimum kota dengan nilai RMSE yang mencapai 0.0728. Kelebihan algoritme *Backpropagation Neural Network (BPNN)* adalah mudah dipahami dan mudah untuk diterapkan dengan hasil yang tetap optimal.

2.2 Lalu Lintas

Menurut Undang-undang No 22 tahun 2009, lalu lintas didefinisikan sebagai gerak Kendaraan dan orang di Ruang Lalu Lintas Jalan. Ruang Lalu Lintas Jalan sendiri diartikan sebagai prasarana yang diperuntukkan bagi gerak pindah Kendaraan, orang, dan/atau barang yang berupa Jalan dan fasilitas pendukung. Lalu lintas memegang peranan penting dalam dalam masyarakat karena seluruh aktifitas masyarakat banyak yang terjadi disana.

2.2.1 Permasalahan Lalu Lintas

Permasalahan lalu lintas di kota kecil tidak sekomplit masalah lalu liintas dikota besar. Masyarakat di kota kecil yang cenderung homogen dan jumlah penduduknya yang sedikit membuat suasana lalu lintasnya cenderung stabil. Namun tidak menutup kemungkinan terjadinya masalah lalu lintas. Data dari POLRI, 2006, menyatakan bahwa setiap 1 jam terjadi 10 kecelakaan, setiap 10 menit terdapat 1 orang cidera ringan, setiap 15 menit terdapat 1 orang cidera parah, setiap 30 menit terdapat 1 orang meninggal dunia dan kerugian nasional mencapai Rp 87 miliyar (Kusmagi, 2010).

Sedangkan permasalahan di kota besar dengan penduduk yang heterogen dan jumlah penduduknya yang besar adalahh kemacetan. Masalah tersebut terjadi karena arus urbanisasi yang tinggi di kota besar sehingga kebutuhan kendaraan pribadi ataupun umum yang semakin meningkat. Di Jakarta, pertumbuhan kendaraan setiap taunnya mencapai 11% sedangkan pertumbuhan jalan tidak sampai 1% (Kusmagi, 2010).

2.2.2 Rambu Lalu Lintas

Rambu lalu lintas adalah salah satu alat perlengkapan di jalan dalam bentuk tertentu yang memuat lambang, huruf, angka, kalimat dan/atau perpaduan di antaranya, yang digunakan untuk memberikan peringatan, larangan, perintah, dan petunjuk bagi pemakai jalan (SB, 2011). Rambu lalu lintas dikelompokkan sebagai berikut:

a. Rambu Peringatan

Rambu yang memperingatkan adanya bahaya, agar para pengemudi berhati-hati dalam menjalankan kendaraannya.

b. Rambu Petunjuk

Rambu ini memberikan petunjuk atau keterangan kepada pengemudi atau pemakai halan lainnya, tentang arah yang harus ditempuh atau letak kota yang akan dituju lengkap dengan nama dan arah .

c. Rambu Tetap

Rambu tetap adalah semua jenis rambu yang diletakkan menurut Surat Keputusan Menteri Perhubungan yang di pasang secara tetap. Sedangkan rambu tidak tetap adalah rambu yang waktunya hanya beberapa waktu saja.

d. Rambu Tambahan

Rambu tambahan adalah rambu-rambu juga membatu para pengguna jalan dan penggunaannya ditetapkan melalui UU yang berlaku.

e. Rambu Larangan dan Perintah

Rambu untuk melarang atau mengijinkan mellewati suatu jalan tertentu.

2.3 Lampu Lalu Lintas

Menurut UU no. 22/2009 tentang Lalu lintas dan Angkutan Jalan: alat pemberi isyarat lalu lintas atau APILL) Lampu lalu lintas adalah lampu yang mengendalikan arus lalu lintas yang terpasang di persimpangan jalan, tempat penyeberangan pejalan kaki (zebra cross), dan tempat arus lalu lintas lainnya. Lampu ini digunakan sebagai penanda kendaraan dari berbagai arah apakah harus jalan atau berhenti secara bergantian sehingga arus lalu lintas lebih tertib. Warna lampu lalu lintas yang sering digunakan adalah merah, kuning, dan hijau seperti yang ada pada Gambar 2.1. Merah menandakan berhenti, hijau menandakan berjalan dan kuning menandakan berhati-hati.

Gambar 2.1 Lampu Lalu Lintas

2.3.1 Sistem Lampu Lalu Lintas

Sistem pengendali lampu lalu lintas dikatakan baik, jika lampu-lampu lalu lintas yang terpasang dapat berjalan baik secara otomatis dan dapat menyesuaikan diri dengan kepadatan lalu lintas tiap-tiap jalur atau disebut *actuated controller*. Namun akademisi Indonesia menemukan sistem baru yang membuat penentuan waktu lalu lintas berdasarkan volume kendaraan yang sedang mengantri. Sistem tersebut menggunakan metode *logika fuzzy.*

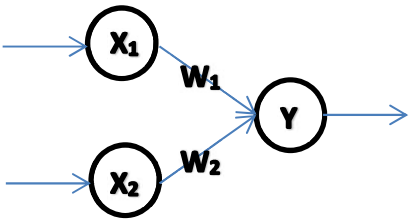
2.4 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan adalah salah satu algoritme pembelajaran yang ada dalam ilmu komputer. Jaringan syaraf tiruan sendiri merupakan sebuah model yang menirukan sistem kerja neuron dalam ilmu biologi yang mempelajari

tentang syaraf otak. Jaringan syaraf tiruan banyak diterapkan pada aplikasi-aplikasi seperti *voice recognition, handwriting recognition*, serta *customer churn analysis.*

Secara umum, jaringan syaraf tiruan menggambarkan relasi antara layer output dengan layer input seperti pada Gambar 2.2. Namun dalam pengembangannya jaringan syaraf tiruan terdiri dari *node* (*neuron*) dan relasi seperti pada Gambar 2.3 dan Gambar 2.4. Ada 3 tipe *node* yaitu *input node*, *hidden node*, dan *output node*. Input node merupakan layer pertama dalam jaringan syaraf tiruan. Setiap input merepresentaskan parameter atau variabel masukan dari sistem. *Hidden node* merupakan merupakan *node* yang berada di tengah. *Node* ini menerima *input* dari *node* baris sebelumnya. *Output* node umumnya merupakan representasi hasil dari sistem jaringan saraf tiruan.

Gambar 2.2 Gambaran Umum Jaringan Syaraf Tiruan



Gambar 2.3 Jaringan Syaraf Tiruan

Pada Gambar 2.3, *neuron* y mendapatkan masukan dari *x1* dan *x2* dengan bobot *x1* dan *x2*. Sehingga persamaan untuk menentukan nilai *y* adalah sebagai berikut:

$$y = \frac{1}{1+e^{-z}} \quad (2.1)$$

$$z = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n \quad (2.2)$$

Keterangan:

$x_1, x_2, x_3, \ldots, x_n$	: Masukan / *input*
$w_1, w_2, w_3, \ldots, w_n$	: Bobot
y	: keluaran / *output*
e	: Bilangen eural yaitu 2.71828
z	: Hasil penjumlahan dari sinyal – sinyal input

2.5 Backpropagation

Backpropagation merupakan salah satu metode pembelajaran dalam jaringan syaraf tiruan. Proses pembelajaran dalam *backpropagation* dilakukan

dengan penyesuaian bobot-bobot (w_n) dengan arah mundur berdasarkan nilai error dalam proses pembelajaran. *Backpropagation* biasanya digunakan dalam kasus-kasus yang membutuhkan prediksi dan penentuan didalam penyelesaiannya.

Proses pertama sebelum masuk ke algoritme backpropagation adalah proses normalisasi. Proses ini digunakan untuk menyetarakan *range* nilai pada setiap fitur menjadi antara 0 sampai dengan 1. Proses normalisasi penting dilakukan agar salah satu fitur yang memiliki tipe nilai yang tinggi tidak menjadi dominan dan juga sebaliknya.

Rumus normalisasi dapat dilihat pada persamaan berikut:

$$x' = \left(0.8 \times \frac{x-min}{max-min}\right) + 0.1 \quad (2.3)$$

Keterangan:

x' : nilai x hasil normalisasi
x : nilai x awal
max : nilai maksimum x
min : nilai minimum x

Backpropagation memiliki 3 fase utama yaitu fase *feedforward*, fase *backpropagation*, dan fase *weight updating*. Fase feedforward adalah proses untuk menghitung keluaran tiap *neuron* mulai dari masukan awal, maju ke layer depannya untuk mendapatkan keluaran *neuron* yang digunakan sebagai masukan *neuron* depannya, dan berakhir pada keluaran pada *output* layer atau disebut *output* sistem. Rumus untuk menghitung keluaran setiap neuron seperti pada persamaan 2.2.

Fase berikutnya adalah fase *backpropagation*, pada fase ini dilakukan proses pencarian error setiap neuron. *Error neuron* pada *hidden* layer dan *error neuron* pada *output* layer memiliki persamaan yang berbada. Persamaan error output layer adalah seperti berikut:

$$\boldsymbol{\delta_k = o_k(1-o_k)(t_k-o_k)} \quad (2.4)$$

Keterangan:

δ_k : Nilai error ouput layer k
o_k : Nilai output k sistem
t_k : Nilai output k target

Sedangkan persamaan error hidden layer adalah seperti berikut:

$$\boldsymbol{\delta_h = o_h(1-o_h)(\sum_{k \in output} w_{h,k}\delta_k)}$$
(2.5)

Keterangan:

δ_h : Nilai *error hidden* layer *h*
o_h : Nilai *output hidden* layer *h*
$w_{h,k}$: Nilai bobot penghubung *hidden* layer *h* dan *output* layer *k*
δ_k : Nilai *error ouput* layer *k* yang terhubung dengan *hidden* layer

Selanjutnya adalah fase perubahan bobot atau *weight updating.* Pada fase ini, bobot dirubah sesuai dengan nilai *error* yang di dapat pada fase *backpropagation*. Adapun persamaan perubahan bobot adalah sebagai berikut:

$$\boldsymbol{w'_{i,j} = w_{i,j} + (\alpha \delta_j x_{i,j})} \quad (2.6)$$

Keterangan:
$w_{i,j}{}'$: Nilai bobot baru
$w_{i,j}$: Nilai bobot awal
δ_j : Nilai error

Setelah bobot dirubah seperti pada persamaan 2.6, selanjutnya dilakukan lagi prosesyang ada mulai dari fase 1 sampai ke fase 3. Proses ini dilakukan sebanyak data yang dilatihkan. Setelah semua data di proses, dilakukan pencarian *Root Means Square Error* (*RMSE*). Nilai tersebut menunjukan nilai *error* sesuai nilai keluaran mulai dari data ke-1 sampai data ke-*n* pada iterasi tertentu. Nilai *RMSE* setiap iterasi berbeda dan cenderung turun. Persamaan untuk mencari nilai *RMSE* adalah sebagai berikut:

$$\boldsymbol{RMSE = \sqrt{\frac{\sum_{i=1}^{n} d_i^2}{n}}} \quad (2.7)$$

Keterangan:
$RMSE$: Nilai *RMSE*
d : Selisih *output* sistem pada data ke-*i* dan *output* target pada data ke-*i*
n : Jumlah Data

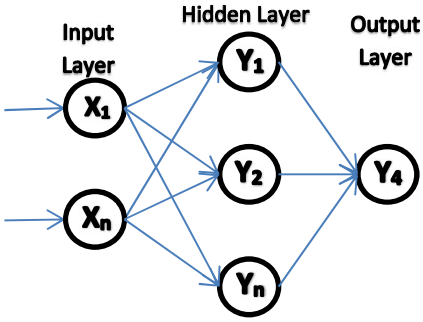
Karena diawal proses data dinormalisasi, maka di akhir proses perlu juga dilakukan denormalisasi. Denormalisasi dilakukan untuk mengembalikan range nilai data kembali ke representasi awalnya, sehigga makna dari nilai tersebut tidak berubah. Persamaan dari proses denormalisasi adalah seperti berikut:

$$\boldsymbol{x'' = \frac{(max - min) \times (x' - 0{,}1)}{0.8} + min} \quad (2.8)$$

Keterangan:
x'' : Nilai denormalisasi

2.5.1 Arsitektur Backpropagation

Karena *backpropagation* merupakan metode pembelajaran dari jaringan syaraf tiruan maka *backpropagation* secara umum memiliki 3 komponen utama, yaitu *input* layer, *hidden* layer, dan *output* layer. Jumlah *Hidden* layer tidak hanya terdapat satu layer, tetapi bisa saja lebih dari satu layer. Tetapi menurut Panchal, et al. (2011), dalam banyak masalah, neural network hanya membutuhkan satu *hidden layer*, sedangkan dua *hidden layer* diperlukan untuk pemodelan data dengan diskontinuitas seperti pola gelombang gigi gergaji namun memiliki resiko tercapainya *local optimum* sehingga tidak ada alasan yang untuk memakai lebih dari dua *hidden layer*. Dan menurut Heaton (2005), jumlah *node* pada *hidden layer* adalah sebanyak 2/3 *input* ditambah dengan jumlah *output*. Arsitektur *backpropagation* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Arsitektur Backpropagation

2.6 Fungsi Aktivasi

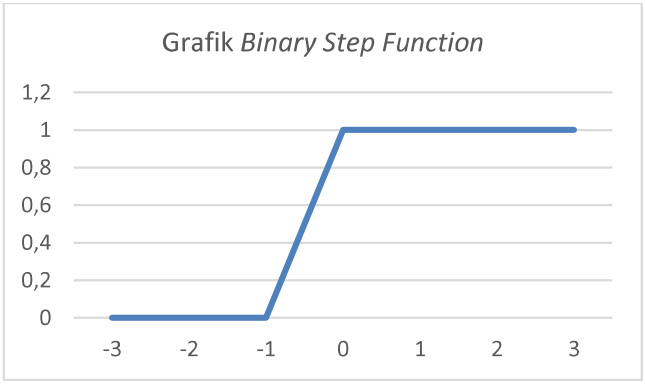
Fungsi aktivasi merupakan salah satu fitur yang penting dalam jaringan syaraf tiruan. Fungsi aktivasi berfungsi untuk memutuskan apakah neuron harus diaktivkan atau tidak. Jika informasi yang diterima oleh sebuah neuron relevan, maka neuron akan diaktifkan, sedangkan jika informasi yang diterima tidak relevan maka akan diabaikan. Fungsi aktivasi memiliki beberapa tipe yang populer, diantaranya adalah *Binary Step, Linear, Sigmoid, Tanh, Relu,* dan *Leaky ReLU* (Gupta, 2017).

2.6.1 Binary Step Function

Binary Step Function merupakan fungsi aktivasi paling sederhana daripada yang lainnya (Gupta, 2017). Seperti namanya, yaitu "*Binary*", fungsi ini hanya membagi nilai masukan menjadi 2 nilai keluaran, yaitu 0 jika nilai masukan kurang dari 0 dan 1 jika masukn lebih dari 0. Persamaan fungsi aktivai *binary step* bisa dilihat pada Persamaan 2.9 dan Persamaan 2.10, sedangkan grafik hasil Persamaan 2.9 dan 2.10 bisa dilihat pada Gambar 2.5.

$$\boldsymbol{y = f(x) = 1, x \geq 0} \quad (2.9)$$

$$\boldsymbol{y = f(x) = 0, x < 0} \quad (2.10)$$



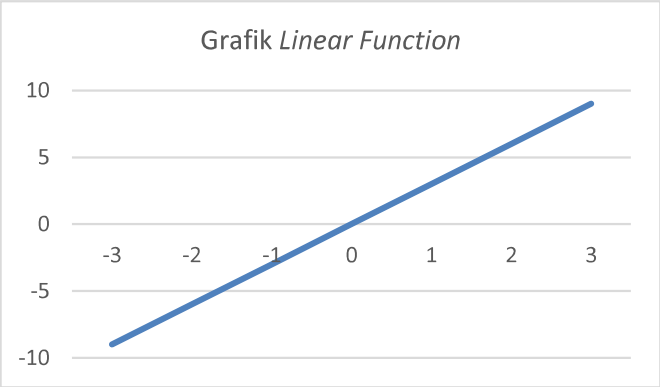
Gambar 2.5 Grafik Fungsi *Binary Step*

2.6.2 Linear Function

Pada fungsi *binary step*, jika Persamaan 2.9 diturunkan akan menjadi $f'(x) = 0$, gradien akan menjadi nol sehigga ketika proses *backpropagation*,

gradiennya tidak akan diperbarui. Solusi dari masalah tersebut adalah menggunakan fungsi *linear* (Gupta, 2017). Input yang masuk pada *node* akan diubah menjadi *ax* seperti pada Persamaan 2.11. Dengan memisalkan nilai a = 3, grafik hasil Persamaan 2.11 dapat dilihat dari pada Gambar 2.6.

$$\boldsymbol{f(x) = ax} \tag{2.11}$$

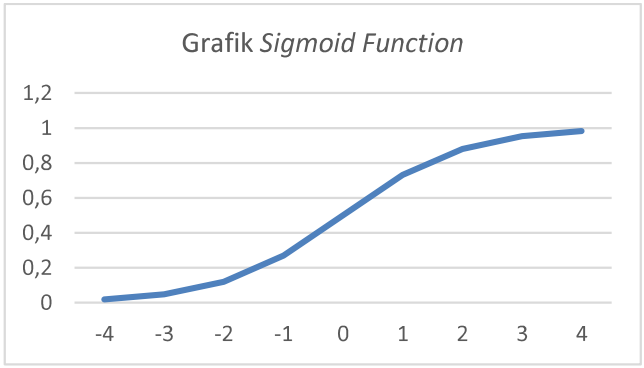


Gambar 2.6 Grafik Fungsi *Linear*

Tetapi fungsi linear memiliki masalah ketika fungsi diturunkan menjadi $f'(x) = a$, menunjukan bahwa derivativ fungsi linear tidak bergantung pada nilai *x* atau inputan melainkan bergantung pada variabel *a*. Sehigga ketika proses backpropagation dilakukan, gradiennya akan sama dan error tidak benar benar diselesaikan berdasarkan *x* melainkan *a*. Sehingga fungsi linear cocok digunakan untuk masalah yang sederhana.

2.6.3 Sigmoid

Fungsi aktivasi sigmoid merupakan fungsi aktivasi yang paling halus (Gupta, 2017). Hal ini bisa dilihat pada Gambar 2., yaitu grafik yang dihasilkan dari Persamaan 2.1. Fungsi sigmoid memiliki rentang antara 0 sampai dengan 1. Sehingga berapapun masukannya, hasilnya akan dihaluskan kedalam rentang 0 sampai dengan 1. Persamaan fungsi aktivasi sigmoid dapat dilihat pada Persamaan 2.1, sedangkan grafik hasil dari Persamaan 2.1 dapat dilihat pada Gambar 2.7.

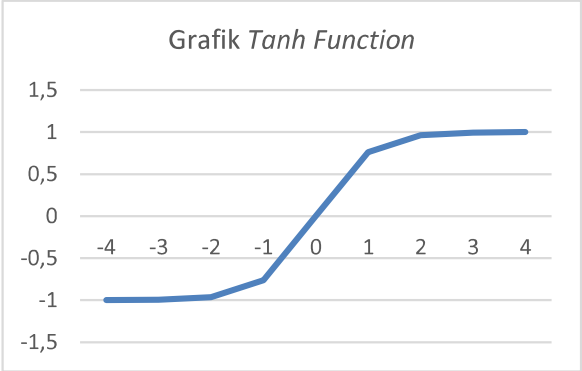


Gambar 2.7 Grafik Fungsi *Sigmoid*

2.6.4 Tanh

Tanh merupakan fungsi aktivasi yang sedikit mirip dengan fungsi aktivasi sigmoid. Yang membedakan kedua fungsi tersebut selain persamaannya yaitu rentang nilai yang dihasilkannya. Rentang nilai yang dihasilkan pada fungsi aktivasi sigmoid berkisar antara 0 sampai dengan 1, sedangkap fungsi aktivasi tanh berkisar antara -1 sampai dengan 1. Persamaan fungsi aktivasi tanh dapat dilihat pada Persamaan 2.12, sedangkan grafik hasil dari Persamaan 2.12 dapat dilihat pada Gambar 2.8.

$$f(x) = \frac{2}{1+e^{-2x}} - 1 \tag{2.12}$$



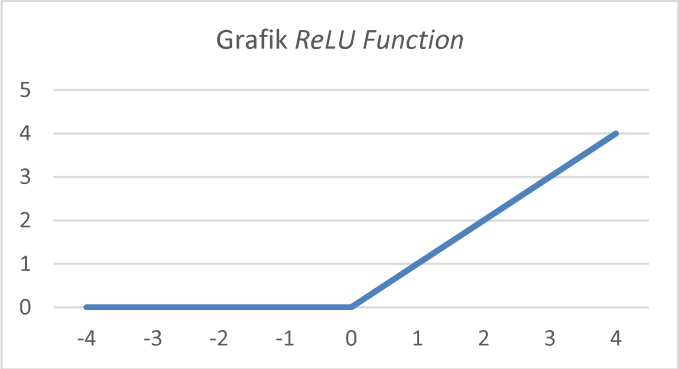
Gambar 2.8 Grafik Fungsi *Tanh*

2.6.5 *Rectified Linear Unit* (*ReLU*)

ReLU merupakan fungsi aktivasi yang saat ini paling sering digunakan ketika merancang jaringan (Gupta, 2017). Fungsi *ReLU* adalah non linier, yang berarti kita dapat dengan mudah melakukan *backpropagation* kesalahan dan memiliki banyak lapisan *node* yang diaktifkan oleh fungsi *ReLU*. Keuntungan utama penggunaan fungsi *ReLU* pada fungsi aktivasi lainnya adalah tidak mengaktifkan semua neuron pada saat bersamaan. Jika input negatif maka fungsi ReLU akan mengubahnya menjadi nol dan *node* tidak diaktifkan (Persamaan 2.14). Ini berarti bahwa pada suatu waktu hanya sedikit *node* yang diaktifkan sehingga membuat jaringan menjadi mudah untuk dihitung. Persamaan fungsi aktivai *ReLU* bisa dilihat pada Persamaan 2.13 dan Persamaan 2.14, sedangkan grafik hasil Persamaan 2.13 dan 2.14 bisa dilihat pada Gambar 2.9.

$$f(x) = \max(0, x), x \geq 0 \tag{2.13}$$

$$f(x) = 0, x < 0 \tag{2.14}$$

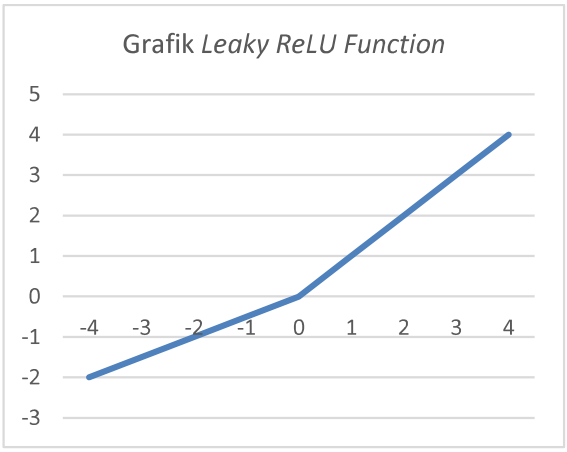


Gambar 2.9 Grafik Fungsi *ReLU*

2.6.6 *Leaky Rectified Linear Unit* (*Leaky ReLU*)

Fungsi *Leaky ReLU* merupakan versi perbaikan dari fungsi *ReLU*. Bagian yang diperbaiki pada fungsi sebelumnya adalah ketika nilai *x* dibawah 0, *node* tidak akan di non-aktifkan, melainkan tetap diaktifkan dengan memberinya nilai yang lebih kecil dengan bantuan variabel tambahan *a*, seperti pada persamaan 2.15. Dengan memisalkan niali a = 0,5 maka grafik hasil Persamaan 2.13 dan 2.15 bisa dilihat pada Gambar 2.10.

$$f(x) = ax, x < 0 \tag{2.15}$$



Gambar 2.10 Grafik Fungsi *Leaky ReLU*

2.7 *Root Means Square Error* (*RMSE*)

RMSE merupakan salah satu metode evaluasi yang digunakan untuk mengukur hasil perkiraan dari suatu metode peramalan. *RMSE* mengukur berapa banyak kesalahan yang terjadi antara dua dataset. Dengan kata lain, *RMSE* membandingkan nilai prediksi atau perkiraan dan nilai yang diamati atau diketahui (Barnston, 1992). *RMSE* dengan nilai yang semakin rendah menunjukan bahwa hasil dari peramalan semakin baik. Persamaan *RMSE* dapat dilihat pada Persamaan 2.7.