BAB II
LANDASAN TEORI

2.1 Pengertian Prediksi

Prediksi adalah proses untuk meramalkan suatu variable di masa mendatang dengan berdasarkan pertimbangan data pada masa lampau. Data yang sering digunakan untuk melakukan prediksi adalah data yang berupa data kuantitatif. Prediksi tidak harus memberikan jawaban secara pasti kejadian yang akan terjadi, melainkan berusaha untuk mencari jawaban sedekat mungkin yang akan terjadi [5].

2.2 Pengertian *Game*

Game atau permainan merupakan media hiburan yang sudah dikenal sejak lama. *Game* dapat dimainkan berbagai umur tua maupun muda. *Game* juga sudah berkembang dengan pesat sesuai dengan perkembangan teknologi, dari *game* sederhana sampai *game* modern saat ini. Hal ini terbukti dengan adanya perkembangan jenis, produk, serta alat yang digunakan.

2.3 *Game* DoTA 2

Game DoTA 2 adalah permainan kompetitif dari aksi dan strategi yang dikembangkan oleh Valvesoftware. *Game* DoTA 2 melibatkan dua regu yang saling bertanding yang terdiri dari masing-masing lima pemain yang memilih salah satu *Hero* dari lebih dari seratus *Hero* yang disediakan. *Game* DoTA 2 merupakan permainan dengan kompleksitas yang tidak terbatas, setiap *Hero* memiliki beberapa kemampuan dan keterampilan yang dapat dikombinasikan dengan kemampuan regu masing-masing dengan cara yang tidak terduga, maka tidak akan ada pertandingan yang sama persis [1].

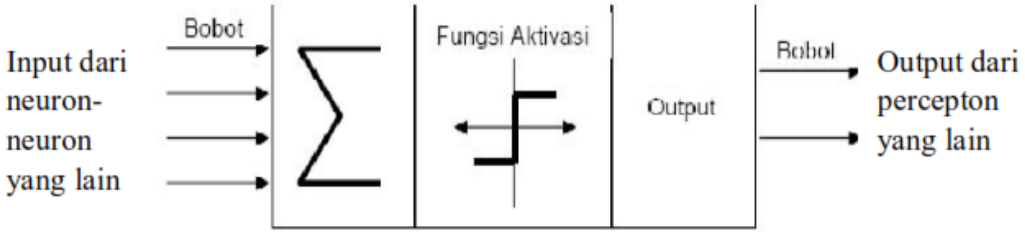
2.4 Kecerdasan Buatan

Kecerdasan buatan atau *Artificial Intelegence* adalah suatu ilmu yang mempelajari tentang bagaimana membuat computer melakukan hal-hal yang pada saat ini dilakukan manusia lebih baik. Dengan adanya kecerdasan buatan diharapkan computer menjadi lebih cerdas, mengerti tentang kecerdasan dan membuat mesin lebih berguna [6].

2.4.1 Neural Network

Neural Network atau Jaringan syaraf tiruan adalah salah satu kajian pada kecerdasan buatan. *Neural Network* digunakan untuk menyelesaikan masalah-masalah yang kompleks dan sulit dipahami, dimanan sejumlah besar data mengenai masalah tersebut telah dikumpulkan. *Neural Network* mencari pola dan hubungan dalam data yang sangat besar yang terlalu rumit dan sulit dianalisis manusia menggunakan perangkat keras dan piranti lunak yang menyerupai pola-pola pemrosesan dalam otak manusia.

Komponen Neural Network terdiri dari *neuron-neuron* yang saling berhubungan. *Neuron – neuron* ini akan mentransformasikan informasi yang diterima melalui sambungan keluarannya menuju *neuron-neuron* yang lain. Pada Neural Network hubungan *neuron-neuron* tersebut dikenal dengan nama bobot [7]. Komponen Neural Network seperti terlihar pada gambar 2.1 terdiri dari input (informasi) bobot (nilai- nilai tertentu), fungsi aktivasi berfungsi apabila input yang dimasukan sesuai dengan nilai ambang *(threshold)* yang ditentukan jika tidak sesuai maka fungsi aktivasi tidak diaktifkan, dan apabila neuron – neuron tersebut diaktifkan maka neuron akan mengirimkan output melalui bobot – bobot.

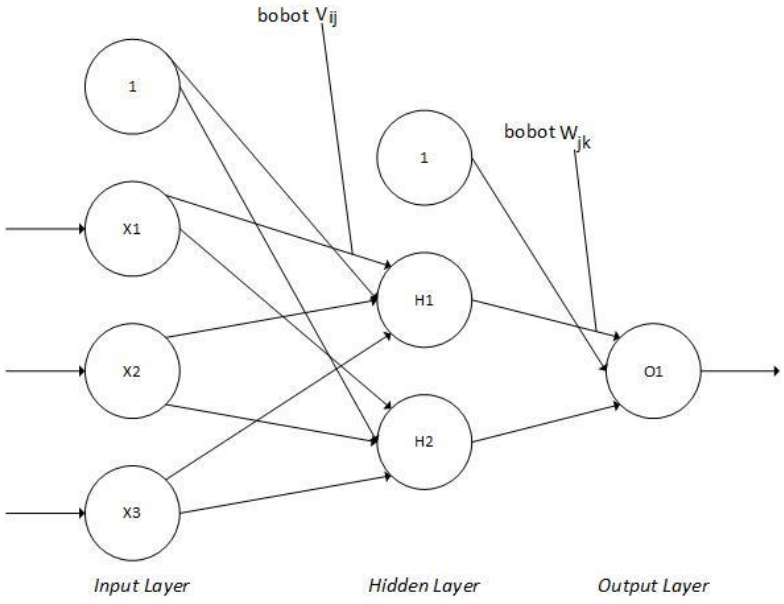


Gambar 2.1 Komponen Neural Network

Karakterisitik yang dimiliki Neural Network adalah sebagai berikut:

a. Arsitektur Jaringan

Merupakan gambaran pola hubungan antara neuron. Pada algoritma Neural Network neuron-neuron disusun pada suatu lapisan (layer), umumnya lapisan-lapisan tersebut terdiri dari lapisan masukan (input layer), lapisan tersembunyi (hidden layer) dan lapisan keluaran (output layer). Neuron-neuron yang terletak pada lapisan yang sama akan memiliki keadaan yang sama. Factor terpenting dalam menentukan kelakuan suatu neuron adalah fungsi aktivasi dan pola bobotnya. Pada setiap lapisan yang sama, neuron-neuron akan memiliki fungsi aktivasi yang sama. Arsitektur Jaringan Neural Network yang digunakan pada penelitian ini adalah arsitektur jaringan dengan banyak lapisan (*multilayer net*) yang memiliki satu lapisan tersembunyi (hidden layer) diantara lapisan masukan dan lapisan keluaran. Arsitektur yang akan digunakan dapat dilihat pada gambar 2.1 berikut:



Gambar 2.2 Arsitektur Neural Network [7]

Dimana :

X = input neuron, merupakan variabel lapisan masukan

H = hidden neuron, merupakan output dari lapisan masukan

O = output neuron, merupakan output dari lapisan tersembunyi

V_{ij} = bobot yang menghubungkan lapisan masukan menuju lapisan tersembunyi

W_{jk} = bobot yang menghubungkan lapisan tersembunyi menuju lapisan Keluaran

b. Pembelajaran

Algoritma Neural Network mempuyai dua macam proses pembelajaran yaitu pembelajaran terawasi (*supervised learning*) dan pembelajaran tidak terawasi (*unsupervised learning*).

1. Pembelajaran terawasi (*supervised learning*)

   Pada Pembelajaran terawasi output yang diharapakan telah diketahui sebelumnya. Pada proses pembelajaran, satu pola input akan diberikan ke satu neuron pada lapisan input. Pola ini akan dirambatkan di sepanjang jaringan syaraf hingga sampai ke neuron pada lapisan output. Lapisan output ini akan membangkitkan pola output yang nantinya akan dicocokkan dengan pola output targetnya. Apabila terjadi perbedaan antara pola output hasil pembelajaran dengan pola target, maka disini akan muncul error. Apabila nilai error ini masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi. Metode pembelajaran terawasi diantaranya *Hebb Rule, Perceptron, Delta Rule, Backpropagation, Learning Vektor Quantization, Heteroassociative Memory* [7].

2. Pembelajaran tak terawasi (*unsupervised learning*)

   Pada pembelajaran tak terawasi tidak memerlukan target output. Pada pembelajaran ini, tidak dapat ditentukan hasil yang seperti apakah yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu range tertentu tergantung pada nilai input yang diberikan. Tujuan pembelajaran ini

adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu [7].

c. Fungsi aktivasi

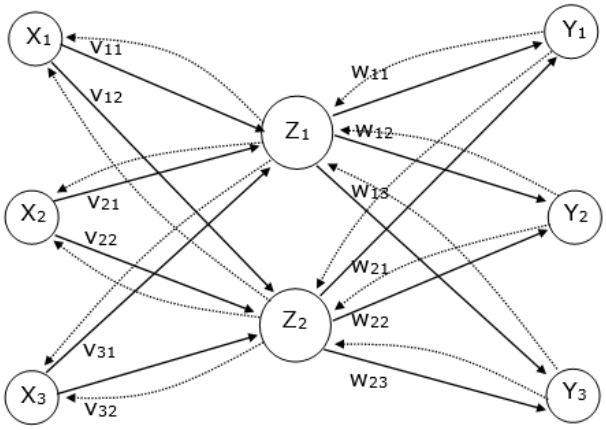
Fungsi aktivasi yang mentranformasikan total input Neural Network pada suatu *neuron* untuk menghasilkan sinyal keluaran (*outgoing activity*). Fungsi aktivasi yang digunakan pada penelitian ini adalah fungsi *sigmoid.* Fungsi ini digunakan untuk Neural Network yang dilatih dengan menggunakan metode *backpropagation*. Fungsi sigmoid memiliki nilai pada range 0 sampai 1. Fungsi ini sering digunakan untuk jaringan yang membutuhkan nilai output yang terletak pada interval 0 sampai 1. Namun, fungsi ini bisa juga digunakan oleh jaringan syaraf yang nilai outputnya 0 atau 1. Fungsi sigmoid terlihat pada persamaan 2.1 berikut [7].

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

2.4.1.1 Backpropagation Neural Network

Backpropagation neural network merupakan tipe jaringan saraf tiruan yang menggunakan metode pembelajaran terawasi (*supervised learning*). Pada *supervised learning* terdapat pasangan data *input* dan *output* yang dipakai untuk melatih Neural Network hingga diperoleh bobot yang diinginkan. Metode Backpropagation menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward).* Untuk mendapatkan error ini, tahap perambatan maju (*forward*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron-neuron diaktifkan dengan menggunakan fungsi aktivasi sigmoid, seperti terlihat pada persamaan 2.1.

Arsitektur Backpropagation Neural Network terdiri atas tiga lapis (*layer*) yaitu lapis masukan (*input layer*) x_i, lapis tersembunyi (*hidden layer*) z_j, dan lapis keluaran (*output keluaran*) y_k. Lapis masukan dan lapis tersembunyi dihubungkan dengan bobot v_{ij} dan antara lapis tersembunyi dan lapis keluaran dihubungkan oleh bobot w_{jk}. Arsitektur dapat dilihat pada gambar 2.3



Gambar 2.3 Arsitektur Backpropagation

Jumlah lapis tersembunyi dan jumlah *neuron* tiap lapisan tidak ada rumus tertentu untuk menghitungnya, namun terdapat aturan yang sering dipakai baik untuk menentukan jumlah lapis tersembunyi maupun jumlah *neuron* tersembunyi didalamnya. Permasalahan yang membutuhkan dua lapis tersembunyi jarang ditemukan, meskipun *neural network* yang terdiri dari dua lapis tersembunyi dapat merepresentasikan segala jenis bentuk fungsi. Tidak terdapat alasan teoritis untuk menggunakan *neural network* lebih dari dua lapis tersembunyi. Faktanya untuk berbagai macam permasalahan tidak terdapat alasan untuk menggunakan lebih dari satu lapis tersembunyi [8]. Kemudian untuk menentukan jumlah *neuron* pada lapis tersembunyi sangat penting dalam membangun arsitektur jaringan. Menggunakan terlalu sedikit neuron pada lapis tersembunyi kemungkinan akan menghasilkan *underfittin,* sebaliknya menggunakan terlalu banyak *neuron* pada lapis tersembunyi mungkin akan menghasilkan *overfitting* dan apabila data latih sangat banyak akan sangat lama untuk melakukan pelatihan [8].

Aturan yang sering dipakai dalam menentukan jumlah *neuron* pada lapis tersembunyi adalah sebagai berikut:

1. Jumlah *neuron* pada lapis tersembunyi adalah antara jumlah neuron lapis masukan dan neuron lapis keluaran [8].

2. Jumlah *neuron* pada lapis tersembunyi adalah 2/3 jumlah neuron lapis masukan, ditambah jumlah neuron lapis keluaran [8].
3. Jumlah *neuron* pada lapis tersembunyi harus lebih sedikit dari dua kali lipat jumlah neuron lapis tersembunyi [8].
4. Jumlah *neuron* pada lapis tersembunyi dipengaruhi jumlah data latih [9]. Persamaan yang digunakan untuk aturan ini adalah sebagai berikut :

$$N_h = \frac{jumlah\ data\ latih}{\alpha(N_i + N_o)}$$

Keterangan:

N_h = jumlah neuron lapis tersembunyi

N_i = jumlah neuron lapis masukan

N_o = jumlah neuron lapis keluaran

α = skala nomor antara 2 – 10

Untuk melatih *Backpropagation, Neural Network* diberi pola masukan sebagai pola pelatihan kemudian pola tersebut akan menuju ke unit pada lapis tersembunyi untuk diterusan pada unit yang berada pada lapis keluaran. Keluaran sementara pada lapis tersembunyi akan diteruskan pada lapis keluaran dan lapis keluaran akan memberikan tanggapan yang disebut sebagai keluaran sementara y_k. ketika $y_k \neq t_k$ dimana t_k adalah keluaran yang diharapkan, maka selisih (*error*) keluaran sementara u'_k akan disebarkan mundur (*backward*) pada lapis tersembunyi dan diteruskan ke unit pada lapis masukan. Oleh karena itu proses tersebut disebut propagasi balik (*backpropagation*) [10].

Pada metode *Backpropagation* terdapat dua tahapan proses, yaitu :

1. Tahapan pelatihan, dimana pada tahap ini metode backpropagation diberikan sejumlah data pelatihan dan target.
2. Tahap pengujian atau penggunaan, pengujian dilakukan setelah melakukan tahap belajar atau tahap pelatihan.

Kondisi pelatihan berhenti atau *stopping condition* dapat menggunakan dua cara yang biasa dipakai, yaitu:

1. Membatasi iterasi yang ingin dilakukan.
   Misalnya jaringan akan dilatih sampai iterasi yang ke-500.
2. Membatasi *error*.
   Misalnya menentukan besar *Mean Square Error*.

Secara umum langkah dalam pelatihan Neural Network menggunakan Backpropagation adalah sebagai berikut :

1. Menentukan jumlah *input* (pola masukan), hidden layer, dan output (target pelatihan).
2. Memberi nilai awal secara random bagi seluruh bobot antara *input-hidden layer* dan *hidden-output layer*.
3. Melakukan *feedforward*
   a. Tiap unit input (x_i) menerima sinyal input dan sinyal tersebut dikirimkan pada seluruh unit pada *hidden layer*.
   b. Tiap unit pada *hidden layer* (Z_in_j) menjumlahkan sinyal-sinyal input terbobot:
   $$z_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (2.2)$$
   gunakan fungsi aktivasi untuk menghitung sinyal outputnya:
   $$z_j = f(z_in_j) \quad (2.3)$$
   kirimkan sinyal tersebut ke semua unit di *output layer*
   c. Tiap unit pada *output layer* (Y_in_k) menjumlahkan sinyal-sinya input terbobot:
   $$y_in_k = w_{0j} + \sum_{i=1}^{n} z_i w_{jk} \quad (2.4)$$
   gunakan fungsi aktivasi untuk menghitung sinyal outputnya:
   $$y_k = f(y_in_k) \quad (2.5)$$
4. Melakukan *Backpropagation*
   a. Tiap unit pada *output layer* (y_k) menerima target pola yang dengan berhubungan dengan pola input pembelajaran, hitung informasi errornya :

$$\delta_k = (t_k - y_k)\, f'(y_in_k) \quad (2.6)$$

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai w_{jk}):

$$\Delta w_{jk} = \alpha\, \delta_k\, z_j \quad (2.7)$$

Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai w_{0k})

$$\Delta w_{0k} = \alpha\, \delta_k \quad (2.8)$$

Kirimkan δ_k ke unit yang berada dilapisan bawahnya.

b. Tiap unit pada *hidden layer* (Z_in_j) menjumlahkan delta inputnya (dari unit yang berada pada lapisan atasnya):

$$\delta_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad (2.9)$$

kalikan nilai ini dengan turunan dari fungsi aktivasinya untu menghitung informasi error:

$$\delta_j = \delta_in_j f'(z_in_j) \quad (2.10)$$

kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij}):

$$\Delta v_{ij} = \alpha\, \delta_j\, x_i \quad (2.11)$$

Hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai v_{0j}):

$$\Delta v_{0j} = \alpha\, \delta_j \quad (2.12)$$

c. Tiap unit output (y_in_k) memperbaiki bias dan bobotnya:

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2.13)$$

tiap unit tersembunyi (z_in_j) memperbaiki bias dan bobotnya:

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad (2.14)$$

5. Tes Kondisi berhenti
   a. Apakah batas *error* minimum telah tercapai.
   b. Apakah batas iterasi maksimum telah tercapai.

Keterangan Persamaan :

z_in	:	Bobot sinyal unit tersembunyi
z	:	Keluaran unit tersembunyi
y_in	:	Bobot sinyal lapisan keluaran
y	:	Keluaran unit keluaran
δ	:	Kesalahan
Δw	:	Koreksi bobot unit keluaran
Δv	:	Koreksi bobot unit tersembunyi
x_i	:	Unit (neuron) pada lapisan input
n	:	Jumlah neuron (unit) pada lapisan input
v	:	Bobot awal masuk ke hidden
v_0	:	Bobot bias yang menuju ke hidden
w	:	Bobot awal hidden ke output
w_0	:	Bobot bias yang menuju ke output
α	:	Learning rate / rasio pembelajaran

2.5 C#

Bahasa C# dikembangkan oleh tim kecil di Microsoft di bawah pimpinan Anders Hejlsberg dan Scott Wiltamuth. Anders juga dikenal sebagai pernbuat Turbo Pascal dan pimpinan tim yang mengembangkan Borland Delphi salah satu IDE (Integrated Development Environment) yang cukup populer untuk pengembangan aplikasi client/server.

C# adalah pengembangan dari bahasa pemrograman sebelumnya, yaitu C/C++, Java, dan Visual Basic. C# memiliki kehandalan yang dimiliki C/C++, mendukung pemrograman berbasis objek, seperti Java, serta kemudahan rapid application development, seperti Visual Basic.

Tujuan bahasa C# adalah untuk menyediakan tool untuk pengembangan program yang simpel, aman, berbasis objek, inrerner-centric, dan mempunyai performa yang handal [11].

2.6 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau yang dikenai dengan DBMS(database management system), database ini multithread, multi-user. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU General Public License (GPL), tetapi juga dijual di bawah lisensi komersial untuk kasus-kasus yang bersifat khusus.

MySQL adalah Relational Database Management System (RDBMS) yang didistribusikan secara gratis di bawah lisensi GPL (General Public License). Di mana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat *closed source* atau komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (Structured Query Language}. SQL adalah sebuah konsep pengoperasian database, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoprasian data dikerjakan dengan mudah secara otomatis.

Sebagai database server, MySQL dapat dikatakan lebih unggul dlbandingkan database server lainnya, terutama dalam kecepatan. Berikut ini beberapa keunggulan MySQL, antara lain:

1. Portability

   MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.

2. Multiuser

   MySQL dapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

3. Security

   MySQL memiliki beberapa lapisan sekuritas seperti level subnetmask, nama host, dan izin akses user dengan system perizinan yang mendetail serta password terenkripsi.

4. Scalability dan limits

   MySQL mampu menangani database dalam skala besar, dengan jumlah records lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya [12].

2.7 Ufield Modelling Language

Unified Modelling Language (UML) merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menemukan atau menggambarkan sebuah system *software* yang terkait dengan objek.

UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk baku [11].

2.7.1 Use Case Diagram

Use case diagram digunakan untuk memodelkan aspek dinamis dari sistem. Fungsi diagram Use Case adalah untuk menggambarkan fungsionalitas yang disediakan oleh sebuah sistem dan interaksi *user* atau *actor* dengan sistem dengan menggunakan fungsionalitas yang tersedia. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merupakan sekumpulan urutan dari aksi atau langkah-langkah tertentu, misalnya login ke sistem atau meng*create* peminjaman pada perpustakaan. Seorang/sebuah actor merupakan sekumpulan peran yang berkaitan yang dimainkan pada saat berinteraksi dengan use case bersangkutan. Biasanya seorang aktor mewakili peran yang dimainkan oleh manusia, perangkat keras, atau bahkan sistem lain [11]

2.7.2 Activity Diagram

Activity Diagram menunjukkan flow aktifitas ke aktifitas (bukan status ke status). *Activity diagram* memodelkan workflow proses bisnis dan urutan aktifitas dalam sebuah prose. Diagram ini sangat mirip dengan flowchart. Membuat *activity diagram* pada awal pemodelan proses sangat membantu memahami keseluruhan proses. *Activity diagram* juga bermanfaat untuk menggambarkan *paralel behavior* atau menggambarkan interaksi antara beberapa *use case* [11].

2.7.3 Class Diagram

Class adalah deskripsi sekumpulan *object* yang memiliki kesamaan atribut, operasi, *relationship* dan *semantics.* Dengan kata lain, sebuah *class* merupakan *blueprint* atau *template* cetakan dari satu atau lebih *object.* Class Diagram dapat membantu dalam memvisualkan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak ditemui dalam pemodelan system berbasis orientasi objek. Pada UML, class digambarkan dengan segi empat yang dibagimenjadi 3 bagian. Bagian yang pertama merupakan 2 nama dari kelas. Bagian yang tengah merupakan strukur dari class (atribut) dan bagian ketiga merupakan *behaviour* operasi dari class tersebut [13].

Class diagram mempunyai beberapa macam hubungan sebagai berikut :

a. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memilikiatribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
b. Agregasi, yaitu hubungan yang menyatakan bagian.
c. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga disebut turunan dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

d. Komposisi, Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut.

2.7.4 Squence Diagram

Sequence diagram menggambarkan interaksi antara sejumlah objek dalam urutan waktu. Sequence diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Umumnya sebuah sequence diagram menangkap behavior dari suatu skenario (*best case*) sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan [11].

2.8 K-Fold Cross Validation

K-fold cross validation adalah pengujian dengan membagi *dataset* menjadi *k* buah subset yang berukuran sama secara acak, ketika k subset menjadi data uji maka subset yang lain menjadi data latih, proses ini berulang sebanyak k kali sampai semua subset telah digunakan baik sebagai data latih maupun sebagai data uji. Pada permasalahan estimasti statistik jumlah k biasanya 10, sedangkan pada neural network tidak terdapat aturan yang pasti namun umumnya jumlah k berkisar antara 5 sampai dengan 10 buah [14]