# IMPLEMENTASI SISTEM

Pada bab ini, akan dibahas implementasi sistem sesuai analisa dan desain sistem pada bab sebelumnya.

Tabel 4.1 Daftar Segmen Program dan Flowchart

|  |  |
| --- | --- |
| SEGMEN PROGRAM | GAMBAR (FLOWCHART) |
| 4.1 | 3.3 |
| 4.2, 4.3 | 3.5, 3.6 |
| 4.4, 4.5 | 3.7 |
| 4.6 | 3.8 |
| 4.8, 4.9, 4.10 | 3.10 |
| 4.11 | 3.11 |

1. **Instalasi *Open Source Library***

Dalam implementasi sistem ini, digunakan dua *open source library*, yaitu *NEAT-Python* dan *Neataptic.js.*

1. **Instalasi NEAT-Python**

*NEAT-Python* merupakan sebuah *open source library* dari bahasa pemrograman *python*. *Library* ini nantinya akan digunakan untuk menjalankan proses NEAT.

Untuk meng-*install* *NEAT-Python*, dibutuhkan *pip,* yang merupakan *package installer* untuk bahasa pemrograman *python*. Jika *pip* sudah ter-*install*, cukup ketikkan **pip install neat-python** pada *terminal* jika menggunakan *linux* atau *macOS,* atau *command prompt* jika menggunakan *windows.*

1. **Instalasi *Neataptic.js***

Untuk melakukan *backpropagation*, dibutuhkan *open source library* lainnya. Pada penelitian ini, library yang digunakan untuk melakukan *backpropagation* adalah *Neataptic.js*, yang merupakan *library* dari bahasa pemrograman *javascript* yang pada penelitian ini akan berjalan diatas *nodejs*.

Sama seperti *NEAT-Python*, *Neataptic.js* juga membutuhkan *npm*, yaitu sebuah *package installer* pada bahasa pemrograman *javascript,* untuk melakukan instalasi. Jika *npm* sudah terinstall, cukup ketikkan **npm install neataptic** pada *terminal* jika menggunakan *linux* atau *macOS,* atau *command prompt* jika menggunakan windows

1. ***Data Pre-processing***

Sebelum data dapat digunakan, harus dilakukan *pre-processing* terlebih dahulu untuk mengekstrak *features* yang akan digunakan. Seperti yang sudah dijelaskan pada Bab 3, pada awalnya format data adalah *json*. Proses ini menggunakan bahasa pemrograman *php*. *Output* dari proses *pre-processing* ini berbentuk *.txt* yang nanti akan digunakan pada proses NEAT.

Segmen Program 4.1 Pre-processing pada data dengan feature berupa player ratings dan team ratings

|  |
| --- |
| $data = json\_decode(file\_get\_contents("dataset/datafile/season14-15/season\_stats.json"),true);  $fInput = array();  $fOutput = array();  $readCount = 0;  foreach($data as $key => $values){      $input = array();      $output = array();      $readCount++;      foreach($values as $key2 => $values2){          foreach($values2 as $key3 => $values3){              if($key3 == 'team\_details'){                  foreach($values3 as $key4 => $values4){                      if($key4 == 'team\_rating'){                          array\_push($input,$values4/10);                      }                  }              }              if($key3 == 'aggregate\_stats'){                  $noGoal = false;                  foreach($values3 as $key4 => $values4){                      if($key4 == 'goals'){                          array\_push($output,$values4);                          $noGoal = true;                      }                    }                  if($noGoal == false){                      array\_push($output,0);                  }              }              if($key3 == 'Player\_stats'){                  foreach($values3 as $key4 => $values4){                      foreach($values4 as $key5 => $values5){                          if($key5 == 'player\_details'){                              foreach($values5 as $key6 => $values6){                                  if($key6 == 'player\_rating' ){                                      if($values6 == 'Sub'){                                          break;                                      }else if($key6 == 'player\_rating'){                                          array\_push($input,$values6/10);                                      }                                  }                              }                          }                      }                  }              }            }      }      array\_push($fInput,$input);      array\_push($fOutput,$output);  }  $input\_str = '[';  $inputCount = 0;  $outputCount = 0;  foreach($fInput as $key => $values){      $inputCount++;      $input\_str =  $input\_str.'(';      foreach($values as $key2 => $values2){          $input\_str = $input\_str.$values2.', ';      }      $input\_str =  $input\_str.'),';    }  $input\_str =  $input\_str.']';  $output\_str = '[';  foreach($fOutput as $key => $values){      $outputCount++;      $output\_str =  $output\_str.'(';      foreach($values as $key2 => $values2){          $output\_str = $output\_str.$values2.', ';      }      $output\_str =  $output\_str.'),';    }  $output\_str =  $output\_str.']';  $myfile = fopen("inputTestWithTeam.txt", "w") or die("Unable to open file!");  fwrite($myfile, $input\_str);  $myfile = fopen("OutputTest.txt", "w") or die("Unable to open file!");  fwrite($myfile, $output\_str); |

Setelah proses *pre-processing* selesai, barulah data siap digunakan untuk proses selanjutnya, yaitu NEAT.

1. **Implementasi NEAT**

Proses NEAT dimulai dengan melakukan pengaturan *config*, yang merupakan aturan-aturan yang berlaku selama proses NEAT berlangsung, seperti kemungkinan terjadinya mutasi, jumlah populasi, dan lain-lain. Contoh *config* pada NEAT dapat dilihat pada Gambar 4.1.

Setelah melakukan pengaturan *config,* akan dijalankan fungsi load\_data() untuk menginputkan data yang telah diproses melalui proses *pre-processing*.

Akan ada 4 data yang akan digunakan, yaitu data untuk *input* dan *output* untuk proses *training* dan data *input* dan *output* untuk proses *testing*.

Jika data sudah diterima dan *config* sudah ditetapkan, proses neat akan berjalan dengan menjalankan fungsi start(), yang menerima *config* dan mengaplikasikannya kepada sistem.

A picture containing outdoor, sitting, monitor, building

Description automatically generated

Gambar 4.1 Contoh *file* *config* pada NEAT

Segmen Program 4.2 Fungsi load\_data()

|  |
| --- |
| def load\_data():      train\_inputs = f=open("inputTestWithTeam.txt","r")      if(f.mode == 'r'):          xor\_inputs = eval(f.read())      train\_outputs = f=open("outputTest.txt","r")      if(f.mode == 'r'):          xor\_outputs = eval(f.read())      valid\_inputs = f=open("validInputWithTeam.txt","r")      if(f.mode == 'r'):          valid\_inputs = eval(f.read())      valid\_outputs = f=open("validOutputTest.txt","r")      if(f.mode == 'r'):          valid\_outputs = eval(f.read()) |

Segmen Program 4.3 Fungsi start() untuk menjalankan proses NEAT

|  |
| --- |
| def start(config\_file):      config = neat.Config(neat.DefaultGenome, neat.DefaultReproduction,                           neat.DefaultSpeciesSet, neat.DefaultStagnation,                           config\_file)      p = neat.Population(config)      p.add\_reporter(neat.StdOutReporter(True))      stats = neat.StatisticsReporter()      p.add\_reporter(stats)      p.add\_reporter(neat.Checkpointer(50))      pe = neat.ParallelEvaluator(4,fitness\_function2)      winner = p.run(pe.evaluate,5000)      print('\nBest genome:\n{!s}'.format(winner))      print('\nOutput:')      calculate\_winner(winner,config)      node\_names = {-1:'A', -2: 'B', 0:'A XOR B'}  visualize.extract\_net(config,winner)      visualize.draw\_net(config, winner, view=False, node\_names=node\_names)      visualize.plot\_stats(stats, ylog=False, view=False)      visualize.plot\_species(stats, view=False) |

Fungsi neat.ParallelEvaluator() menerima sebuah *fitness function* yang akan digunakan untuk mengevaluasi populasi pada NEAT. Akan ada 2 *fitness function* yang akan digunakan pada penelitian ini.

Segmen Program 4.4 *Fitness* *function* pertama

|  |
| --- |
| def eval\_genome1(genome, config):      fitness = 1140      net = neat.nn.FeedForwardNetwork.create(genome, config)      for xi, xo in zip(xor\_inputs, xor\_outputs):          output = net.activate(xi)          output[0] = round((output[0]\*10))          output[1] = round((output[1]\*10))          if(xo[0]\*10 == output[0] and xo[1]\*10 == output[1] ):              genome.fitness += 1          else :              if(xo[0]\*10 > xo[1]\*10 and output[0] > output[1] ):                  genome.fitness += 0.5              if(xo[0]\*10 < xo[1]\*10 and output[0] < output[1] ):                  genome.fitness += 0.5              if(xo[0]\*10 == xo[1]\*10 and output[0] == output[1] ):                  genome.fitness += 0.5      return fitness |

Segmen Program 4.5 *Fitness* *function* kedua

|  |
| --- |
| def eval\_genome2(genome, config):      fitness = 1140      net = neat.nn.FeedForwardNetwork.create(genome, config)      for xi, xo in zip(xor\_inputs, xor\_outputs):          output = net.activate(xi)          fitness -= (output[0] - xo[0]) \*\* 2          fitness -= (output[1] - xo[1]) \*\* 2      return fitness |

Pada *fitness function* pertama, masing masing *genome* akan diberikan nilai fitness sebesar 1 jika skor yang dihasilkan benar, sedangkan jika skor salah tetapi pemenang yang diprediksi benar, nilai fitness yang diberikan adalah 0.5. Sedangkan pada *fitness function* kedua, nilai *fitness* dihitung berdasarkan *sum of squared errors*.

Fungsi neat.parallelEvaluator() kemudian akan di*passing*kan kepada fungsi run() untuk menjalankan proses NEAT. Fungsi run() juga menerima parameter berupa jumlah iterasi yang ingin dijalankan.

Setelah fungsi run() selesai dijalankan, yang menandakan proses NEAT telah selesai, fungsi ini akan meng*outputkan* genome terbaik dengan nilai *fitness* terbaik. Dari genome terbaik ini kemudian akan di*construct* sebuah *neural network* yang akan di*passing*kanke fungsi calculate\_winner()

Fungsi calculate\_winner() bertujuan untuk melakukan proses *testing* terhadap *genome* terbaik yang dihasilkan oleh NEAT.

Segmen Program 4.6 Fungsi calculate\_winner()

|  |
| --- |
| def calculate\_winner(winner,config):  winner\_net = neat.nn.FeedForwardNetwork.create(winner,config)      correct\_predict = 0      correct\_winner = 0      for xi, xo in zip(valid\_inputs, valid\_outputs):          output = winner\_net.activate(xi)          output[0] = round((output[0]))          output[1] = round((output[1]))          if(xo[0] == output[0] and xo[1] == output[1] ):              correct\_predict += 1          if(xo[0] > xo[1] and output[0] > output[1] ):              correct\_winner += 1          if(xo[0] < xo[1] and output[0] < output[1] ):              correct\_winner += 1          if(xo[0] == xo[1] and output[0] == output[1] ):              correct\_winner += 1          print("input {!r}, expected output {!r}, got {!r}".format(xi, xo, output))      print("Prediction accuracy = {!r} ".format(correct\_predict))      print("Winner accuracy = {!r} ".format(correct\_winner)) |

Setelah semua proses pada NEAT selesai, genome terbaik akan diekstrak untuk dioptimasi menggunakan backpropagation. Proses ekstraksi ini terjadi pada fungsi visualise.extract\_net()

Segmen Program 4.7 Fungsi extract\_net()

|  |
| --- |
| def extract\_net(genome,config):      node\_dict = dict()      node\_obj = []        for k in config.genome\_config.input\_keys:          node\_dict = collections.defaultdict(dict)          node\_dict['key'] = k          node\_dict['bias'] = 0          node\_dict['activation'] = ''          node\_dict['type'] = 'input'          node\_obj.append(node\_dict)      for i in genome.nodes:          node\_dict = collections.defaultdict(dict)          if genome.nodes[i].key > 1 :              node\_dict['key'] = genome.nodes[i].key              node\_dict['bias'] = genome.nodes[i].bias              node\_dict['activation'] = genome.nodes[i].activation              node\_dict['type'] = 'hidden'              node\_obj.append(node\_dict)      for i in genome.nodes:          node\_dict = collections.defaultdict(dict)          if genome.nodes[i].key == 1 or genome.nodes[i].key == 0:              node\_dict['key'] = genome.nodes[i].key              node\_dict['bias'] = genome.nodes[i].bias              node\_dict['activation'] = genome.nodes[i].activation              node\_dict['type'] = 'output'              node\_obj.append(node\_dict)        with open('node\_data.json', 'w') as outfile:          json.dump(node\_obj, outfile)      conn\_dict = dict()      conn\_obj = []      for i in genome.connections:          if genome.connections[i].enabled == True:              conn\_dict = collections.defaultdict(dict)              conn\_dict['from'] = genome.connections[i].key[0]              conn\_dict['to'] = genome.connections[i].key[1]              conn\_dict['weight'] = genome.connections[i].weight              conn\_obj.append(conn\_dict)      with open('conn\_data.json', 'w') as outfile:          json.dump(conn\_obj, outfile) |

Fungsi extract\_net() akan menghasilkan 2 *output* berformat *json,* yaitu *node\_data.json* yang berisi informasi dari *node* dan *conn\_data.json­* yang berisi informasi koneksi yang menghubungkan satu *node* dengan *node* lainnya.

1. **Implementasi *Backpropagation***

Setelah semua proses NEAT selesai, proses selanjutnya adalah *backpropagation*. Langkah pertama pada prose *backpropagation* ialah menyiapkan data untuk *training* dan *testing*, serta melakukan konsturksi *network* berdasarkan output dari extract\_net() pada NEAT. Semua proses *backpropagation* menggunakan bahasa pemrograman *javascript*

Segmen Program 4.8 fungsi prepare\_data() untuk menyiapkan data berdasarkan tipe *feature* yang digunakan

|  |
| --- |
| prepareData(type){    if(type == 1){        this.trainingData = fs.readFileSync('./dataset/inputTestPlayerOnly.json');        this.validData = fs.readFileSync('./dataset/validTestPlayerOnly.json');     }else if(type == 2){        this.trainingData = fs.readFileSync('./dataset/inputTestWithTeam.json');        this.validData = fs.readFileSync('./dataset/validTestWithTeam.json');     }else if(type == 3){        this.trainingData = fs.readFileSync('./dataset/inputTestWithTeamAndPos.json');        this.validData = fs.readFileSync('./dataset/validTestWithTeamAndPos.json');          }else{              console.log("unknown type");          }      } |

Segmen Program 4.9 fungsi build() untuk mengkonstuksi *network*

|  |
| --- |
| build(node\_json , conn\_json){          let contents = fs.readFileSync(node\_json);          this.node\_data = JSON.parse(contents);            this.node\_count = 0;          for(let i in this.node\_data){              this.node\_count++;          }            this.nodes = Array(this.node\_count);          for(let i = 0 ; i < this.nodes.length ; i++){              this.nodes[i] = new Node(this.node\_data[i].type , this.node\_data[i].key , this.node\_data[i].bias)          }          contents = fs.readFileSync(conn\_json);          this.conn\_data = JSON.parse(contents);            for(let i in this.conn\_data){              for(let j = 0 ; j < this.nodes.length ; j++){                  if(this.conn\_data[i].from == this.nodes[j].key){                      for(let h = 0 ; h < this.nodes.length ; h++){                          if(this.conn\_data[i].to == this.nodes[h].key){                              if(j != h){                                  this.nodes[j].connect(this.nodes[h] , this.conn\_data[i].weight);                              }                            }                      }                  }              }          }          this.network = architect.Construct(this.nodes);      } |

Setelah *network* berhasil dikonsturksi dan data telah disiapkan, proses *training* dan proses *testing* dapat dijalankan.

Segmen Program 4.10 fungsi backprop() untuk menjalankan proses *training*

|  |
| --- |
| backprop(){          let backprop = true;          let input = JSON.parse(this.trainingData)          let opt = {              log: 100,              error: 0,              iterations: 100000,              rate: 0.001,          }          this.network.train(input,opt,propagate);  } |

Pada segmen ini, backprop diatur ke true sebagai tanda untuk melakukan proses *training*. Object opt berisi konfirgurasi pada proses *training*. Selanjutnya, fungsi network.train() yang berasal dari *library neataptic.js* akan dijalankan untuk memulai proses *training*.

Setelah proses *training* selesai dijalankan, akan dilakukan proses *testing* menggunakan fungsi calculate\_accuracy() yang bertujuan untuk mengukur tingkat akurasi dari *network*. Fungsi ini menggunakan data *testing* yang tidak dikenali oleh *network*.

Segmen Program 4.11 fungsi calculateAccuracy() untuk mengukur akurasi dari *network*

|  |
| --- |
| calculateAccuracy(){          let set = JSON.parse(this.testingData);          for(let i = 0 ; i < set.length ; i++){              let input = set[i].input;              let target = set[i].output;              let output = this.network.activate(input, true);              let temOutput = output;                temOutput[0] = Math.round(temOutput[0]);              temOutput[1] = Math.round(temOutput[1]);              console.log('output = '+temOutput+' target = '+target);              if(Math.round(output[0]) == target[0] && Math.round(output[1]) == target[1]){                  correct++;              }              if(Math.round(output[0]) > Math.round(output[1]) && target[0] > target[1]){                  win++;              }              if(Math.round(output[0]) < Math.round(output[1]) && target[0] < target[1]){                  win++;              }              if(Math.round(output[0]) == Math.round(output[1]) && target[0] == target[1]){                  win++;              }          }            console.log("Prediction correct = "+correct)          console.log("winner correct = "+win)      } |