BAB II

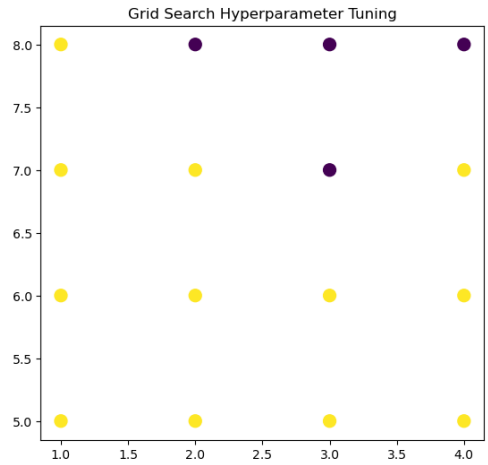
KAJIAN PUSTAKA

A. Kajian Teoritis

1. *Hyperparameter Tuning*

Hyperparameter tuning mempunyai peranan yang sangat penting dalam mengoptimalkan kinerja *machine learning* apa pun (Nugraha & Sasongko, 2022). *Hyperparameter* beserta nilainya harus ditetapkan sebelum model menjalani proses pembelajarannya (Bartz et al., 2023). Penetapan *hyperparameter* beserta nilainya tergantung dari intuisi atau perkiraan pengembang model kecerdasan buatan dengan berlandaskan dari pengalaman atau ilmu yang didapat.

Perlu digaris bawahi pemilihan *hyperparameter* dan nilainya harus berkesinambungan dikarenakan ini adalah proses yang cukup lama jika terlalu banyak *hyperparameter* dan nilainya (Decastro-García et al., 2019). Memilih teknik yang tepat untuk pengoptimalan *hyperparameter tuning* juga merupakan kunci untuk menghasilkan penyetelan *hyperparameter* yang lebih optimal (Putra Utama et al., 2022). Selain itu pemilihan model juga perlu diperhatikan karena pemilihan *hyperparameter* dan nilainya, pemilihan teknik dari *hyperparameter tuning*, dan pemilihan model kecerdasan buatan bergantung pada tujuan dan isi dari dataset. Ilustrasi *hyperparameter tuning* dapat dilihat pada gambar 2.1.



Gambar 2. 1 Ilustrasi *hyperparameter tuning* teknik *grid search*

2. *Progressive Web Application (PWA)*

Progressive Web Application (PWA) adalah aplikasi web yang bisa berjalan layaknya aplikasi native pada umumnya. *Progressive Web Application (PWA)* memiliki keunggulan dapat diinstall diberbagai *platform* mulai dari *android*, *ios*, dan *dekstop* (Hikmah et al., 2020). *Progressive Web Application (PWA)* bekerja dengan adanya *Service Worker*, *service worker* adalah skrip yang dijalankan di latar belakang *browser*. Dengan *service worker*, *website* dapat berjalan secara *offline* baik di mobile ataupun dekstop sehingga memberikan pengalaman menggunakan *website* menjadi lebih mudah dan cepat.

Progressive Web Application (PWA) tidak seperti aplikasi asli khusus pada suatu platform, namun sifat asli dari *website* pada dasarnya bersifat agnostik platform yang berarti tidak terikat atau tidak bergantung pada platform (Li, 2021). Aplikasi *web* berjalan pada *browser*, sehingga dapat diakses di perangkat apa pun yang memiliki browser, selain itu semua platform pastinya sudah memiliki aplikasi

browser bawaannya masing-masing seperti Chrome, Edge, Mozila, dan lain-lain. Sebaliknya, aplikasi android didistribusikan melalui platform distribusi aplikasi berpemilik yaitu *Google Play Store*. Hal ini membatasi jangkauan aplikasi untuk melewati platform lain, seperti sistem operasi *IOS, MAC OS,* dan *Dekstop* kecuali versi dari setiap sistem operasi juga diterapkan. *Progressive Web Application (PWA)* mampu menjembatani kesenjangan yang merupakan perbedaan dalam jangkauan dan kemampuan dalam aplikasi.

3. *Grid Search*

Tenik *grid search* biasanya digunakan untuk optimasi *hyperparameter. Grid search* adalah teknik yang populer untuk menentukan semua kombinasi *hyperparameter* (Ahmad et al., 2022). *Learning rate* dan jumlah *layer* adalah dua parameter terpenting dalam teknik *grid sarch*.

Prinsip *grid searh* adalah pencarian menyeluruh. *Grid search* akan melatih model kecerdasan buatan dengan mengkombinasikan tiap-tiap *hyperparameter* dan nilainya dan mengevaluasi model sehingga akan menciptakan beberapa model pelatihan tergantung dari pengaturan *hyperparameter* dan nilainya (Wu et al., 2019). Hasil evaluasi dari tiap tiap model inilah yang akan menjadi perbandingan kombinasi manakah yang terbaik dari optimasi *hyperparameter* dan nilainya. Meskipun teknik ini bisa mencari kombinasi terbaik untuk

pelatihan model tetapi jika pengaturan *hyperparameter* dan nilainya terlalu banyak akan memakan waktu yang sangat lama.

4. *Neural Netwrok*

Neural Network (NN) merupakan salah satu metode prediksi atau sebuah model kecerdasan buatan yang didasarkan pada cara kerja neuron pada otak manusia (Fanny, 2023). Neuron tiruan atau buatan tersebut dibangun oleh beberapa elemen seperti *input, hidden layer,* dan *output* (Daqiqil, 2021). Proses yang dilakukan dimulai dari *input* yang nilainya dipengaruhi oleh *weight* (bobot) untuk setiap input yang ada.

Dari penjelasan diatas Neural Network (NN) adalah model tiruan yang didasarkan pada kerja otak manusia. Neural Network (NN) memiliki 3 pondasi utama yaitu *input, hidden layer,* dan *output*. Tetapi pengaturan dari 3 pondasi ini juga perlu diperhatikan dari dataset yang ingin diolah dan dipelajari oleh mesin dengan metode ini. Seperti contoh, jika pengaturan *hidden layer* tidak diperhitungkan maka bisa terjadi *overfitting* atau *underfitting.*

5. *Typescript*

Javascript telah menjadi bahasa pemrograman lintas platform yang sangat popular dan digunakan untuk pengembangan aplikasi *web* baik *frontend* maupun *backend* (Goldberg, 2022). Namun, *Javascript* mempunyai keterbatasan dalam menangani tipe variabelnya dikarenakan *Javascript* bersifat *loosely typed* atau bertipe dinamis. Artinya, tipe variabel dalam *Javascript* tidak perlu didefinisikan secara

ekplisit saat deklarasi. *Typescript* hadir sebagai pelengkap dan solusi dengan memperkelankan pengecekan tipe statis dalam *Javascript.*

6. *Python*

Python adalah bahasa pemrograman yang paling populer terlebih lagi dalam pengembangan kecerdasan buatan (Anam et al., 2023). *Python* memiliki berbagai paket atau *package* untuk membantun mengembangkan kecerdasan buatan seperti *tensorflow, keras, skitlearn,* dan masih banyak lagi (Benjamin Dicken, 2023). Itulah mengapa *python* adalah bahasa yang paling populer dikalangan para pengembang kecerdasan buatan.

7. *Tensorflow*

Tensorflow adalah *framework* komputasi serbaguna yang digunakan untuk membangun dan melatih berbagai model pembelajaran mesin (Nasha Hikmatia A.E. & Zul, 2021). Dengan menyediakan beragam *toolkit* yang memudahkan pengguna dalam merancang model sesuai dengan keinginan dan kemampuan (Yudistira et al., 2023). Selain itu, *tensorflow* juga dapat menjalankan model tersebut di berbagai perangkat keras, sehingga memberikan fleksibilitas dan skalabilitas dalam penerapannya.

8. *Next.Js*

Next.Js merupakan *framework* fleksibel yang dapat digunakan untuk membuat aplikasi web dengan cepat. *Next.Js* memerlukan *react* sebagai *library Javascirpt* untuk membuat antarmuka pengguna yang interaktif (Phie Joarno et al., 2022). Terlebih lagi *Next.Js* juga memiliki *library* tambahan yang digunakan untuk membangun *Progressive Web Application(PWA).*

9. *Train-Test-Validation(TTV)*

Metode *train-test-validation (TTV)* digunakan untuk melatih model prediktif dan mengevaluasi akurasi serta stabilitasnya. Proses ini melibatkan pembagian sampel secara acak menjadi tiga kelompok: pelatihan, pengujian, dan validasi. Sampel pelatihan digunakan untuk membuat dan melatih model, sampel pengujian melacak kesalahan selama pelatihan untuk mencegah *overfitting* dan sampel validasi memvalidasi kekuatan prediktif dan stabilitas model (Ballestar et al., 2019). Pada penelitian ini sampel pelatihan terdiri dari 80% sampel yang digunakan untuk melatih model. Sampel pengujian adalah kumpulan data independen yang mencakup 10% sampel yang digunakan untuk melacak kesalahan selama pelatihan Model *Neural Network (NN)* untuk mencegah *overfitting*. Terakhir, sampel validasi juga merupakan kumpulan data independen yang mencakup 10% sampel. Ini digunakan untuk memvalidasi kekuatan prediksi dan stabilitas model.

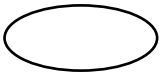
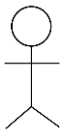
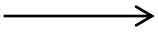
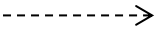
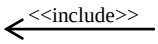
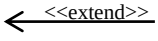
10. *UML (Unified Modeling Language)*

UML *(Unified Modeling Language)* adalah bahasa pemodelan standar yang digunakan dalam rekayasa perangkat lunak untuk memvisualisasikan, menspesifikasikan, membangun, dan mendokumentasikan artefak sistem (Syarif & Nugraha, 2020). UML menyediakan notasi grafis dan aturan semantik yang komprehensif untuk menggambarkan struktur statis, perilaku dinamis, dan interaksi dalam sistem yang kompleks (Destriana et al., 2022). Dengan demikian, UML memfasilitasi komunikasi yang efektif antara pemangku kepentingan teknis dan non-teknis, serta mendukung seluruh siklus hidup pengembangan sistem, mulai dari analisis kebutuhan hingga implementasi dan evolusi. Berikut adalah beberapa diagram pada UML:

a. *Use Case Diagram*

Use case diagram adalah salah satu dari berbagai jenis diagram UML yang menggambarkan hubungan interaksi antara sistem dan aktor (Prasetya et al., 2022). *Use case* dapat mendeskripsikan tipe interaksi antara pengguna sistem dengan sistemnya. *Use case diagram* digunakan untuk mengkomunikasikan interaksi manusia *(actor)* dengan apa yang bisa dilakukan oleh sistem. Simbol-simbol *Use Case Diagram* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Simbol-simbol *Use Case*

Simbol	Nama	Deskripsi
	*Use Case*	Merepresentasikan fungsionalitas atau tindakan yang dilakukan sistem dari sudut pandang pengguna.
	Aktor	Merepresentasikan pengguna atau sistem eksternal yang berinteraksi dengan sistem yang sedang dimodelkan.
	Asosiasi	Menghubungkan aktor dengan *use case* yang dilakukannya. Menunjukkan interaksi antara aktor dan sistem.
	Generalisasi	Menunjukkan spesialisasi aktor untuk mendapatkan partisipasi dengan *use case*.
	*Include*	Menunjukkan bahwa suatu *use case* menyertakan fungsionalitas dari *use case* lain. *Use case* yang di-include harus selesai agar *use case* yang menyertakannya dapat selesai.
	*Extend*	Menunjukkan bahwa suatu *use case* memperluas atau menambahkan fungsionalitas pada *use case* lain. *Use case* yang di-*extend* bersifat opsional dan dapat terjadi dalam kondisi tertentu.

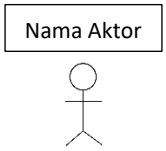
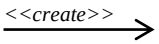
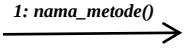
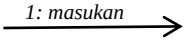
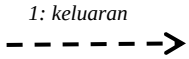
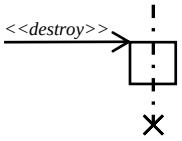
Sumber: (Noviantoro et al., 2022)

b. *Sequence Diagram*

Sequence diagram atau diagram urutan adalah sebuah diagram yang digunakan untuk menjelaskan dan menampilkan interaksi antar objek-objek dalam sebuah sistem secara terperinci (Abdillah et al., 2019). Selain itu, *sequence diagram* juga akan menampilkan pesan atau perintah yang dikirim, beserta waktu pelaksanaannya. *Sequence diagram* mengilustrasikan objek-objek yang terdapat pada *use case* dan menggambarkan arus pesan antara satu sama lain pada objek *use case*. *Sequence diagram* bersifat dinamis dan lebih banyak menampilkan aktivitas objek berdasarkan urutan waktu.

Diagram ini sangat berguna dalam menggambarkan skenario spesifik dari sebuah use case, menunjukkan bagaimana objek berkolaborasi untuk mencapai tujuan tertentu. Dengan menggunakan sequence diagram, pengembang dan analis sistem dapat memahami alur eksekusi, mengidentifikasi potensi masalah desain, dan memastikan bahwa interaksi antar objek berjalan sesuai dengan yang diharapkan. Simbol-simbol *Sequence Diagram* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Simbol-simbol *Sequence Diagram*

Simbol	Nama	Deskripsi
	*Actor*	Menunjukkan pengguna atau sistem eksternal yang berinteraksi dengan sistem.
	Life line (Garis hidup)	Menunjukkan keberadaan objek selama interaksi.
	*Activation*	Menunjukkan periode waktu ketika objek sedang aktif dan melakukan operasi.
	Pesan tipe *create*	Merupakan pernyataan satu objek membuat objek lain.
	Pesan tipe call	Merupakan pernyataan satu objek memanggil metode atau operasi pada objek lain atau diri sendiri
	Pesan tipe send	Merupakan pernyataan bahwa objek mengirimkan informasi atau masukan atau data ke objek lain.
	Pesan tipe return	Merupakan pernyataan bahwa objek menjalankan metode atau operasi yang memberi hasil suatu pengembalian atau keluaran ke objek tertentu.
	Pesan tipe destroy	Merupakan pernyataan bahwa satu objek mengakhiri hidup dari objek lain, jika ada create lebih baik ada *destroy*.

Sumber: (Hutabri et al., 2019)

c. *Class Diagram*

Class diagram atau diagram kelas adalah salah satu jenis diagram struktur pada UML yang menggambarkan dengan jelas struktur serta deskripsi *class*, atribut, metode, dan hubungan dari setiap objek (Abdillah et al., 2019). *Class diagram* adalah model statis yang mendukung tampilan data dan informasi dari keseluruhan sistem. Penggunaan *class diagram* dikaitkan dengan struktur basis data sistem atau dapat menggantikan ERD pada proses penggambaran diagram rekayasa perangkat lunak yang konvensional.

Diagram ini sangat penting dalam perancangan sistem berorientasi objek karena memungkinkan pengembang untuk memodelkan struktur kelas dan hubungan antar kelas secara visual. Dengan menggunakan class diagram, pengembang dapat mengidentifikasi kelas-kelas yang diperlukan, atribut-atribut yang relevan, dan metode-metode yang harus diimplementasikan. Selain itu, class diagram juga membantu dalam memahami hierarki pewarisan dan hubungan asosiasi antar kelas. Simbol-simbol *Class Diagram* dapat dilihat pada Tabel 2.3.

Tabel 2. 3 Simbol-simbol *Class Diagram*

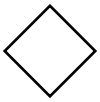
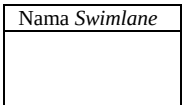
Simbol	Nama	Deskripsi
Nama_kelas +atribut +operasi()	*Class*	Merupakan kelas yang ada pada struktur sistem. Memiliki atribut dan operasi dalam kelas.
Nama _interface	*Interface*	Merupakan kemiripan dengan kelas tetapi memiliki metode yang di deklarasikan tanpa isi dan tanpa atribut kelas.
	Association	Merupakan relasi antarkelas (umum), biasanya dilengkapi dengan *miltiplicity*.
	Directed association	Merupakan relasi antara kelas bermakna satu kelas digunakan oleh kelas yang lain, biasanya dilengkapi dengan *multiplicity*.
	Generalisasi	Merupakan relasi antara kelas bermakna generalisasi-spesialisasi (umum ke khusus).
	Dependency	Merupakan relasi kebergantungan (*dependency*) antara kelas
	Aggregation	Merupakan relasi antara kelas bermakna semua-bagian (*whole-part*).

Sumber: (Hutabri et al., 2019)

d. *Activity Diagram*

Activity diagram adalah diagram yang dapat memodelkan proses-proses yang terjadi pada sebuah sistem. Runtutan proses dari suatu sistem digambarkan secara vertikal. *Activity diagram* merupakan pengembangan dari *use case* yang memiliki alur aktivitas (Gimnastiar et al., 2023). *Activity diagram* menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Simbol-simbol *Activity Diagram* dapat dilihat pada Tabel 2.4.

Tabel 2. 4 Simbol-simbol *Activity Diagram*

Simbol	Nama	Deskripsi
	Status awal/ akhir	Merupakan status awal atau akhir keadaan dari sistem, setiap diagram aktivitas memiliki satu status awal.
	Aktivitas	Merupakan kegiatan yang dilakukan sistem, sering dimulai dengan kata kerja.
	*Decision*	Merupakan hubungan percabangan untuk keputusan aktivitas yang memiliki lebih dari satu pilihan.
	Join	Merupakan hubungan penggabungan jika satu atau lebih aktivitas menjadi satu.
	*Swimlane*	Merupakan yang memisahkan organisasi bisnis. Memiliki tanggung jawab untuk aktivitas yang terjadi.

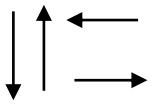
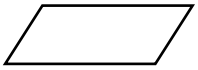
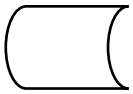
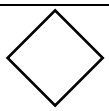
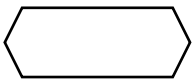
Sumber: (Hutabri et al., 2019)

11. *Flowchart*

Flowchart adalah gambar yang terdiri dari simbol-simbol tertentu yang berfungsi untuk menggambarkan proses secara detail dan berurutan, serta hubungan antar proses dalam program (Yusril et al., 2022). *Flowchart* adalah cara untuk menjelaskan tahap-tahap pemecahan masalah dengan merepresentasikan simbol-simbol tertentu yang mudah dipahami, mudah digunakan dan standar (Java et al., 2021). *Flowchart* dapat membantu dalam menjelaskan alur suatu sistem secara logis dan terstruktur.

Flowchart sangat berguna dalam pengembangan perangkat lunak karena memungkinkan pengembang untuk memvisualisasikan alur logika program secara jelas. Dengan menggunakan flowchart, pengembang dapat mengidentifikasi langkah-langkah yang diperlukan, percabangan kondisi, pengulangan, dan titik akhir dari suatu proses. Selain itu, flowchart juga membantu dalam menganalisis efisiensi algoritma dan menemukan potensi kesalahan atau perbaikan yang dapat dilakukan. Simbol-simbol *Flowchart* dapat dilihat pada Tabel 2.5.

Tabel 2. 5 Simbol-simbol *Flowchart*

Simbol	Nama	Deskripsi
	*Flow Direction*	Untuk menunjukan garis alir proses dan menghubungkan simbol satu dengan simbol lain
	*Terminator*	Untuk permulaan (*start*) dan akhir (*stop*) dari suatu proses.
	*Input/Output*	Untuk menggambarkan proses masukkan (*input*) dan keluaran (*output*) pada sistem.
	*Process*	Untuk menggambarkan proses pengolahan yang ada pada sistem.
	*Disk Storage*	Untuk menyatakan masukan dan keluaran yang berasal dari *disk*.
	*Decision*	Untuk memilih proses atau keputusan berdasarkan kondisi yang ada.
	*Connector on page*	Untuk menghubungkan atau menyambungkan proses dalam satu lembar kerja yang sama.
	*Connector off page*	Untuk menghubungkan atau menyambungkan proses pada lembar kerja yang berbeda.
	*Document*	Untuk input berasal dari dokumen dalam bentuk kertas atau output yang perlu dicetak diatas kertas.
	*Preparation*	Untuk mempersiapkan penyimpanan di dalam *storege*.

12. Metode *Agile*

Agile adalah suatu metodologi pengembangan perangkat lunak yang mengedepankan fleksibilitas dan kolaborasi tim dalam menghadapi perubahan kebutuhan pengguna (Susilo & Azimah, 2023). Metodologi ini menekankan pada pengembangan iteratif dan inkremental, di mana perangkat lunak dikembangkan dalam siklus pendek yang disebut sprint. Setiap sprint menghasilkan peningkatan fungsionalitas sistem yang dapat segera diuji dan digunakan oleh pengguna. *Agile* juga mendorong komunikasi yang terbuka dan umpan balik yang berkelanjutan antara tim pengembang dan pemangku kepentingan untuk memastikan bahwa produk yang dihasilkan sesuai dengan harapan dan kebutuhan pengguna.

13. *Blackbox Testing*

Pengujian *black box testing* merupakan metode pengujian yang berfokus pada fungsionalitas perangkat lunak. Artinya, pengujian ini dilakukan untuk melihat apakah semua fungsi dari sistem yang dirancang dapat berjalan sesuai dengan yang diharapkan, tanpa perlu melihat atau memahami kode program yang mendasari fungsi-fungsi tersebut (Wicaksono et al., 2021). Dengan kata lain, penguji hanya memperhatikan input yang dimasukkan ke dalam sistem dan output yang dihasilkan, tanpa mengetahui bagaimana proses di dalamnya terjadi.

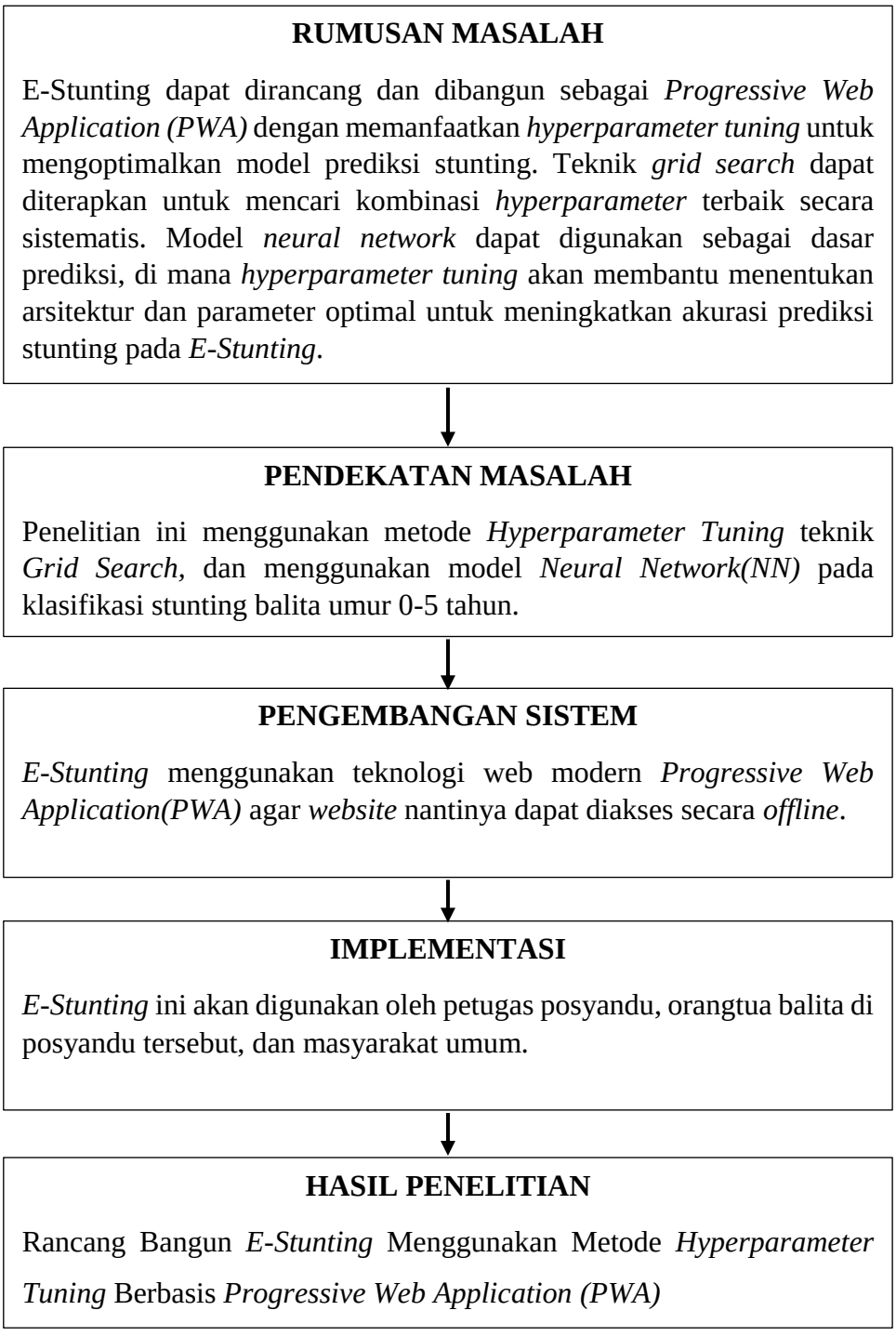
B. Kajian Empiris

Berikut ini merupakan penelitian yang relevan untuk dijadikan acuan dalam penelitian ini:

1. Penelitian yang dilakukan oleh Nugraha & Sasongko, (2022) berfokus pada optimasi *hyperparameter* dalam algoritma klasifikasi *machine learning* untuk prediksi diabetes, sedangkan penelitian ini mengembangkan aplikasi web untuk klasifikasi stunting pada balita menggunakan model *Neural Network* dan teknik *grid search*.
2. Penelitian yang dilakukan oleh Phie Joarno et al., (2022) berhasil mengimplementasikan *Progressive Web Apps (PWA)* pada situs web *GetHelp* dengan *Next.js*, memungkinkan akses seperti aplikasi *Android* tanpa *browser*. Penelitian ini menginspirasi pengembangan website *E-Stunting* menggunakan PWA.
3. Penelitian yang dilakukan oleh Kunang et al., (2021) dan penelitian ini memiliki perbedaan mendasar dalam fokus dan metodologi. Pada penelitian yang sudah disebutkan sebelumnya penelitian tersebut berfokus pada klasifikasi serangan dalam sistem deteksi intrusi (IDS) menggunakan *deep learning* dan optimasi *hyperparameter*, sedangkan penelitian ini mengembangkan aplikasi web (PWA) untuk klasifikasi stunting pada balita menggunakan model *Neural Network* (NN) dengan dataset stunting balita dan teknik *grid search*.

C. Kerangka Berfikir

Berdasarkan latar belakang dan kajian teori diatas maka kerangka berfikir penelitian dapat dilihat pada gambar 2.2 sebagai berikut:



Gambar 2. 2 Kerangka berfikir