BAB II

KAJIAN PUSTAKA

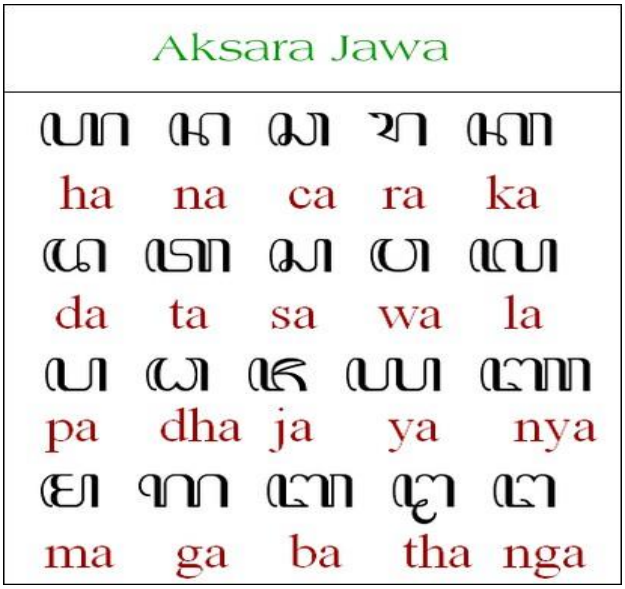
A. Kajian Teoritis

1. Pengertian Huruf Aksara Jawa

Aksara jawa hanacaraka merupakan salah satu aksara yang digunakan di Jawa dan sekitarnya. Aksara hancaraka sebenarnya diambil dari lima aksara pertama dalam aksara Jawa: “hana caraka”. Aksara Jawa sendiri berjumlah dua puluh aksara, yaitu ha – na – ca – ra – ka – da – ta – sa – wa – la – pa – dha – ja – ya – nya – ma – ga – ba – tha – nga. Seiring perkembangan zaman, aksara hanacaraka mengalami beragam perubahan bentuk dan komposisi hingga seperti yang kita kenal sampai saat ini. (Hidayat et al., 2020:507). Sedangkan menurut Linayanti, (2022:359) huruf jawa atau lebih dikenal dengan sebutan aksara Jawa merupakan huruf yang bersifat *syllabaic* (kesukukataan) yang mampu berbunyi walaupun berdiri sendiri.

Dari kedua pendapat di atas dapat disimpulkan bahwa aksara Jawa, atau yang lebih dikenal sebagai Hanacaraka, awalnya terdiri dari 20 aksara. Aksara ini diambil dari lima aksara pertama dalam aksara Jawa yang menyusun kata "hana caraka". Aksara Jawa memiliki sifat kesukukataan, yang berarti setiap hurufnya dapat berbunyi jika berdiri sendiri. Seiring dengan perkembangan zaman, aksara Hanacaraka mengalami berbagai perubahan bentuk dan komposisi hingga menjadi seperti yang kita kenal saat ini.

Huruf aksara jawa berjumlah 20 dapat dilihat pada Gambar 2.1:



Gambar 2.1 Huruf Aksara Jawa

Sumber: (BJ Widodo, 2020:22)

2. Pengertian Klasifikasi Citra

Klasifikasi adalah teknik yang digunakan untuk mengelompokkan data ke dalam berbagai kategori dengan tujuan memprediksi kategori dari data yang belum diketahui. Dalam konteks data mining, klasifikasi adalah proses mengelompokkan data baru berdasarkan kategori yang sudah ada sebelumnya. (Hakim et al., 2022:128). Widiastuti et al., (2023:222) klasifikasi adalah proses pengelompokan data berdasarkan karakteristik setiap objek untuk menetapkan kategori tertentu pada objek tersebut. Dalam klasifikasi citra, proses ini bertujuan mengelompokkan piksel dalam sebuah citra ke dalam beberapa kategori, di mana setiap kategori merepresentasikan entitas dengan ciri-ciri tertentu. Berdasarkan pendapat ini, dapat disimpulkan bahwa klasifikasi citra adalah teknik atau proses mengelompokkan data atau

piksel dalam citra ke dalam beberapa kategori berdasarkan karakteristik atau ciri-ciri tertentu, dengan tujuan memprediksi atau memberikan kategori pada data atau objek baru yang diklasifikasikan.

3. Pengertian *Machine learning*

Machine learning adalah bidang ilmu yang mempelajari cara sistem komputer menyelesaikan tugas-tugas tertentu tanpa instruksi eksplisit, dengan memanfaatkan algoritma dan metode statistik. Bidang ini merupakan bagian dari kecerdasan buatan dan melibatkan pembentukan model matematika dari data pelatihan untuk membuat prediksi atau keputusan tanpa memerlukan instruksi khusus. (Arbain et al., 2022:185). R.H. Zer et al., (2022):151) *machine learning* didefinisikan sebagai disiplin ilmu dalam kecerdasan buatan yang menggunakan bahasa pemrograman untuk membuat komputer berperilaku cerdas seperti manusia. Proses ini memerlukan analisis data dalam jumlah besar (Big Data) untuk mengidentifikasi pola-pola tertentu. *Machine learning* berfokus pada desain dan analisis algoritma yang memungkinkan komputer belajar, meskipun masih merupakan disiplin yang relatif baru dengan banyak hal yang belum ditemukan dibandingkan dengan pengetahuan yang ada saat ini. Berdasarkan beberapa definisi tersebut, dapat disimpulkan bahwa *machine learning* adalah cabang ilmu komputer dan kecerdasan buatan yang berfokus pada penerapan algoritma dan data untuk mengajar komputer belajar dari data, mengekstrak pola, dan membuat keputusan tanpa instruksi eksplisit. *Machine learning* mencakup metode pembelajaran terawasi (*supervised*), tak terawasi (*unsupervised*), semi-

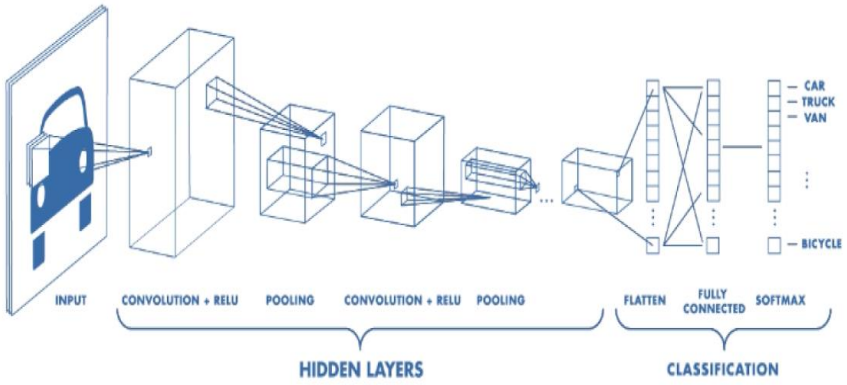
terawasi (*semi-supervised*), pembelajaran penguatan (*reinforcement learning*), dan pembelajaran mendalam (*deep learning*).

4. Konsep dasar *Convolusional Neural Network*

a) Pengertian *Convolusional Neural Network*

CNN atau *Convolutional Neural Network* adalah jenis jaringan saraf tiruan *feedforward* dengan struktur yang mendalam. *CNN* merupakan salah satu contoh algoritma *deep learning*. Keunggulannya terletak pada kemampuannya untuk mengidentifikasi informasi tersembunyi dari berbagai objek seperti gambar, suara, teks, dan lainnya, meskipun objek tersebut berada di posisi manapun dalam *input.* (Arif Faizin et al., 2022:343). Jaringan saraf tiruan *convolutional* adalah metode yang digunakan dalam pengenalan dan pemrosesan gambar. Jaringan ini meniru cara sel-sel saraf manusia berkomunikasi dengan neuron yang saling terhubung dan memiliki arsitektur serupa. (Kholik, 2021:12).

Sedangkan menurut Qotrunnada & Utomo, (2022:800) *Convolutional Neural Network* (*CNN*) adalah salah satu algoritma dalam *deep learning*. *CNN* digunakan untuk mengklasifikasikan gambar atau video, serta mendeteksi objek yang ada dalam gambar atau wilayah tertentu di dalam gambar. Ilustrasi proses CNN dapat dilihat pada Gambar 2.2:



Gambar 2.2 Contoh Proses *Convolutional Neural Network*

Sumber: (A. Kholik 2021:12)

Berikut adalah langkah-langkah umum untuk melakukan klasifikasi gambar menggunakan metode CNN:

1. Pengumpulan dan Pra-pemrosesan Data
   a. Kumpulkan dataset gambar yang relevan dengan tujuan klasifikasi.
   b. Lakukan pra-pemrosesan data seperti normalisasi, mengubah ukuran (*resizing*), pemotongan (*cropping*), augmentasi data, dan membagi dataset menjadi data latih dan data uji.
2. Membuat Arsitektur CNN
   a. Tentukan arsitektur CNN yang akan digunakan untuk klasifikasi gambar, termasuk jumlah lapisan, ukuran *kernel*, *pooling*, *dropout*, dan fungsi aktivasi yang digunakan.
   b. Gunakan arsitektur model standar seperti VGG, ResNet, atau AlexNet, atau buat model dari awal.

3. Pelatihan Model

a. Latih model CNN menggunakan data latih yang telah diproses dengan teknik *backpropagation* dan optimasi gradien untuk memperbarui bobot model.

b. Tentukan parameter pelatihan seperti jumlah *epoch*, ukuran batch (*batch size*), *learning rate*, dan *optimizer*.

4. Evaluasi Model

a. Evaluasi performa model menggunakan data uji yang belum pernah dilihat sebelumnya, dan hitung metrik seperti akurasi, presisi, *recall*, dan *f1-score* untuk mengevaluasi performa model.

b. Jika performa model tidak memuaskan, modifikasi parameter pelatihan atau ubah arsitektur model dan lakukan pelatihan ulang.

5. Penggunaan Model

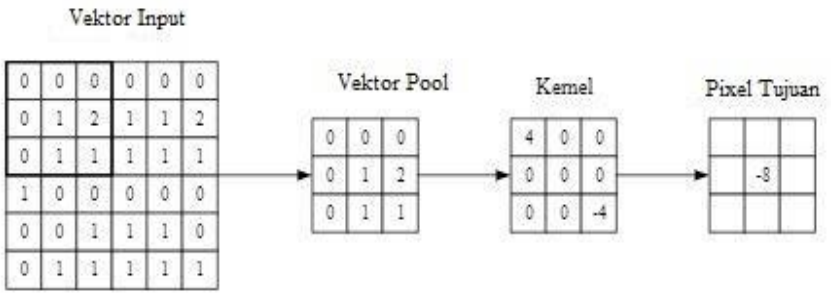
Setelah model dipelajari dan performanya dievaluasi dengan memuaskan, gunakan model tersebut untuk melakukan prediksi pada gambar baru dan dapatkan hasil klasifikasi yang diinginkan.

Langkah-langkah ini dapat diulang dan ditingkatkan dengan mengubah parameter model, augmentasi data, atau menggunakan teknik *transfer learning* untuk meningkatkan performa model.

b) *Convolution Layer*

Lapisan konvolusional adalah lapisan yang melakukan proses konvolusi pada *output* dari lapisan sebelumnya. Ini adalah bagian penting dari Jaringan Saraf Tiruan Konvolusional (CNN), mengandung filter yang

dipelajari secara acak untuk melakukan konvolusi dan mengekstrak *fitur* dari *input*. Tujuannya adalah mempelajari representasi *fitur* dari *input* melalui ekstraksi *fitur* pada citra *input.* (Alwanda et al., 2020:47). Sedangkan menurut Magdalena et al., (2021:336) lapisan konvolusi menggunakan filter yang disebut *kernel* untuk mengekstrak objek atau fitur dari citra *input*. *Kernel* ini berisi bobot yang digunakan untuk mengidentifikasi karakteristik objek. Selanjutnya, proses konvolusi dilakukan untuk menghasilkan transformasi linier dari citra *input*, mempertimbangkan informasi spasial dalam data. Ilustrasi lapisan konvolusi dapat dilihat pada Gambar 2.3:



Gambar 2.3 *Representasi visual layer konvolusi*

Sumber: (Magdalena et al., 2021:336)

Convolutional Layer adalah lapisan pertama dalam arsitektur CNN yang menerima input berupa gambar. Operasi di lapisan ini melibatkan operasi konvolusi, yaitu melakukan kombinasi linear filter pada area lokal dari gambar input. Filter ini mewakili bidang *reseptif* dari *neuron* yang terhubung ke area lokal pada input gambar. *Convolutional Layer* melakukan konvolusi pada *output* dari lapisan sebelumnya dan

merupakan proses utama yang membentuk sebuah CNN. Tujuan dari konvolusi pada data citra adalah untuk mengekstraksi fitur-fitur dari citra *input*. Secara umum, operasi konvolusi dapat ditulis dengan rumus:

$$FM[i]_{j,k} = \left(\sum_m \sum_n N_{[j-m,k-n]} F_{[m,n]}\right) + bF$$

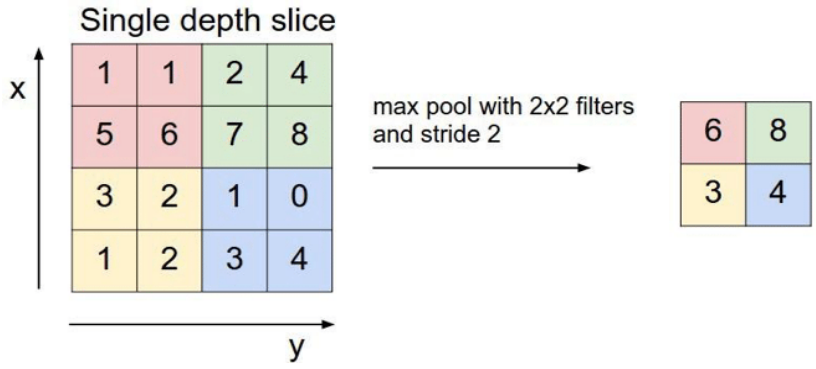
Keterangan

FM[i]	: Matriks *feature map* ke-i
j,k	: Posisi *pixel* pada matriks citra *input*
m,n	: Posisi *pixel* pada matriks *filter* konvulasi
N	: Matriks citra masukan
F	: Matriks *filter* konvulasi
bF	: Nilai bias pada *filter*

c) *Pooling Layer*

Lapisan *Pooling* berperan dalam mengurangi dimensi peta aktivasi dengan melakukan proses *downsampling*, di mana setiap peta aktivasi direduksi ukurannya secara terpisah. Metode ini umumnya menggunakan nilai terbesar melalui max-pooling atau rata-rata melalui *average-pooling*. *Downsampling* sendiri adalah teknik untuk mengurangi ukuran data seperti citra, suara, atau sinyal dengan mengambil subset dari data asli dan menghilangkan bagian-bagian data yang dianggap tidak penting. Dalam konteks pengolahan citra, *downsampling* digunakan untuk menurunkan resolusi citra tanpa mengorbankan informasi penting. Proses ini dapat mempercepat pengolahan citra atau mengurangi ukuran data sebelum

dilakukan analisis lebih lanjut. (Budi et al., 2021:5048). Gambaran proses *pooling layer* dapat dilihat pada Gambar 2.4 berikut:



Gambar 2.4 *Pooling Layer*

Sumber: (Peryanto et al., 2020:140)

Gambar tersebut menunjukkan bahwa dalam *CNN*, dilakukan operasi *max-pooling* pada citra berukuran 4x4 menggunakan *mask pooling* berukuran 2x2. Hasil dari *max-pooling* adalah matriks yang lebih kecil dari matriks asli. Proses *konvolusi* dan pooling ini diulang beberapa kali hingga menghasilkan peta fitur dengan ukuran yang diinginkan. Peta fitur tersebut kemudian digunakan sebagai input pada jaringan *neural fully connected*.

d) *Fully Connected Layer*

Fully Connected Layer (Lapisan Terhubung Secara Penuh) ini adalah lapisan dalam arsitektur *Convolutional Neural Network* (C*NN*) yang bertanggung jawab untuk melakukan klasifikasi. Lapisan ini menghubungkan setiap *neuron* dari lapisan sebelumnya dengan setiap *neuron* di dalamnya. Setiap *neuron* menerima input dari semua *neuron* pada lapisan sebelumnya dan menjalankan operasi matematis untuk

menghasilkan *output*. *Output* dari lapisan ini kemudian menjadi input untuk lapisan berikutnya. Lapisan ini memiliki peran penting dalam proses klasifikasi karena menggabungkan semua fitur yang diperoleh dari lapisan-lapisan sebelumnya dan menggunakan operasi *linier* dan *non-linier* untuk mengklasifikasikan data. (Denta Sukma & Mukhaiyar 2022:366)

e) *Softmax*

Fungsi aktivasi yang umum digunakan pada *output layer* dalam model deep-learning adalah *softmax*. *Softmax* berfungsi untuk menghasilkan probabilitas dari *input* yang diberikan. Input yang diberikan dalam bentuk vektor yang dihasilkan dari proses *Flatten*. *Output* dari fungsi *softmax* berupa nilai probabilitas dalam rentang antara 0 hingga 1. Fungsi *softmax* digunakan dalam model multi-class, di mana setiap kelas akan memiliki nilai probabilitasnya masing-masing. Nilai probabilitas tertinggi dari setiap kelas kemudian digunakan sebagai prediksi untuk input tersebut. (Ersyad et al., 2020:8215). Persamaan dari *softmax* dapat dituliskan sebagai berikut

$$f_j(Z) = \frac{e^{zJ}}{\sum_k e^{zk}}$$

Fungsi *Softmax* menghasilkan nilai fj untuk setiap elemen ke-j pada vektor keluaran kelas. Argumen z merupakan hipotesis yang diberikan oleh model pelatihan dan digunakan untuk melakukan klasifikasi menggunakan fungsi *Softmax*. Fungsi *Softmax* memiliki hasil yang lebih

mudah dipahami dan memiliki interpretasi probabilitas yang lebih baik dibandingkan dengan algoritma klasifikasi lainnya. Fungsi *Softmax* juga memungkinkan untuk menghitung probabilitas untuk semua label yang ada. Dalam proses ini, vektor nilai riil yang merepresentasikan label akan diubah menjadi vektor nilai antara 0 dan 1, yang total jumlahnya adalah satu.

f) *ReLU*

Berdasarkan pendapat Achmad et al., (2019:10596) ReLU (*Rectified Linear Unit*) adalah sebuah fungsi aktivasi yang digunakan dalam jaringan saraf untuk menormalisasi nilai yang dihasilkan dari *convolutional layer*. Fungsi aktivasi ini mengubah *input* yang negatif menjadi 0 dan *input* yang positif menjadi sama dengan *input* tersebut. Dalam konteks *convolutional layer*, ReLU digunakan untuk memastikan bahwa semua nilai *output* dari *layer* tersebut tidak bernilai negatif, sehingga membuat proses klasifikasi lebih mudah dan efektif. Persamaan ReLU ditunjukkan dibawah ini.

$$f = \max(0, x)$$

g) *Epoch*

Menurut Wasil et al., (2022:54) *Epoch* adalah langkah dalam pelatihan CNN, diterapkan untuk menjalankan proses berulang dalam klasifikasi gambar untuk mencari nilai los dan akurasi yang digunakan. Performa yang dihasilkan mungkin buruk karena parameter belum

sepenuhnya disesuaikan. Namun, menggunakan *epoch* yang terlalu besar dapat meningkatkan waktu komputasi dan berisiko mengalami *overfitting*.

Epoch dalam konteks *machine learning* dan *deep learning* merujuk pada satu putaran penuh dari pelatihan model, di mana setiap sampel dalam dataset telah digunakan satu kali untuk melakukan perhitungan pembelajaran (*learning*) dan pembaruan bobot (*weight update*) pada model. Secara lebih detail, selama satu *epoch*, model melakukan beberapa iterasi atau *batch*, di mana setiap *batch* adalah sejumlah sampel yang diproses secara bersamaan oleh model. Jumlah sampel dalam setiap *batch* ditentukan oleh parameter "*batch size*" yang diatur oleh pengguna saat memulai pelatihan model.

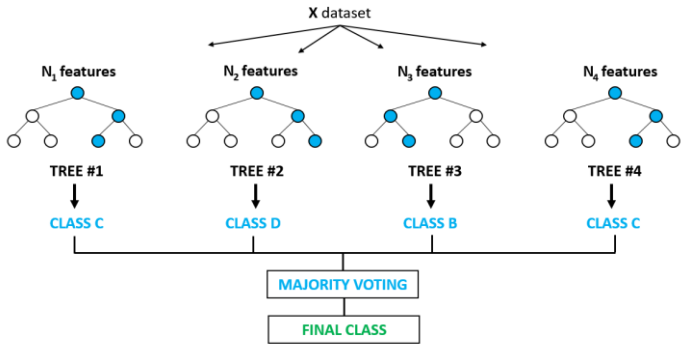
Setelah setiap *batch*, model melakukan perhitungan untuk mengevaluasi kinerja model terhadap sampel yang telah diproses dan memperbarui bobot model berdasarkan hasil evaluasi tersebut. Proses ini dikenal sebagai *backpropagation*, di mana model memperbarui bobotnya dengan mengurangi gradien dari fungsi kerugian (*loss function*) terhadap bobot. Setelah satu *epoch* selesai, model akan menghitung kinerja keseluruhan pada dataset dengan menggunakan model yang telah diperbarui, dan menampilkan nilai evaluasi seperti akurasi, presisi, dan *recall*. Jika nilai evaluasi masih belum memuaskan, maka pelatihan akan dilanjutkan dengan melakukan beberapa *epoch* tambahan hingga model cukup optimal.

Secara umum, jumlah *epoch* yang diperlukan untuk melatih model tergantung pada kompleksitas tugas, ukuran dataset, dan arsitektur model yang digunakan. Pada beberapa kasus, melatih model dengan terlalu banyak *epoch* dapat menyebabkan *overfitting*, di mana model menjadi terlalu spesifik untuk data latih dan tidak mampu menggeneralisasi dengan baik pada data baru. Oleh karena itu, pemilihan jumlah *epoch* yang tepat sangat penting untuk mendapatkan model yang optimal.

5. Konsep dasar *Random forest*

a) Pengertian *Random forest*

Random forest adalah model pembelajaran mesin yang menggunakan pembelajaran terawasi dan terdiri dari banyak pohon keputusan. Pohon keputusan bertindak sebagai klasifikasi dasar dalam hutan acak dan bekerja bersama sebagai sebuah kesatuan. Inilah sebabnya mengapa *random forest* kadang-kadang disebut sebagai *Classifier Ensembles.* (Dutta et al., 2020:536). Sedangkan berdasarkan pendapat Mekha & Teeyasuksaet (2021:166) *random forest* adalah sebuah metode pembelajaran mesin yang menggabungkan banyak pohon keputusan untuk klasifikasi dan metode lainnya. Dalam algoritma *random forest*, banyak pohon keputusan digunakan untuk melatih dan memberikan hasil prediksi klasifikasi (regresi) dari setiap pohon. Algoritma *random forest* berfungsi sebagai sebuah prediktor yang terdiri dari properti-properti yang didasarkan pada regresi acak dari pohon-pohon tersebut. Ilustrasi proses dalam algoritma Random Forest dapat dilihat pada Gambar 2.5:



Gambar 2.5 Ilustrasi *Random forest*

Sumber:(Khairani et al., 2022:535)

Berdasarkan gambar diatas, konsep tahapan persiapan dan estimasi menggunakan *random forest* adalah sebagai berikut:

a. Langkah awal adalah melakukan *bootstrap* hingga mencapai jumlah l, yaitu dengan mengambil sampel secara acak dari data latihan variabel independen yang dimiliki dengan ukuran kembali n.
b. Menggunakan contoh pada setiap *bootstrap* untuk membangun pohon hingga mencapai ukuran maksimum. Susun pohon berdasarkan data bootstrap. Setiap proses pemisahan memilih m < p sebagai variabel independen, dan pemisahan terbaik dilakukan pada tahap sub-setting secara acak.
c. Ulangi langkah 1-2 sebanyak l kali untuk membentuk sebuah hutan yang terdiri dari pohon-pohon.
d. Melakukan prediksi berdasarkan hasil prediksi pada setiap dataset hasil *Bootstrapping* dengan menggunakan mayoritas suara untuk kasus klasifikasi.

b) *Decision Tree*

Decision tree adalah struktur data yang dikenal sebagai pohon terdiri dari simpul dan rusuk. Simpul pohon dapat dibagi menjadi tiga jenis, yakni simpul akar (*root/node*), simpul percabangan/internal (*branch/internal node*), dan simpul daun (*leaf node*). (Nasrullah, 2021:46). Berdasarkan pendapat Mekha & Teeyasuksaet (2021:167) alat pengambilan keputusan berbentuk pohon, atau yang lebih dikenal dengan istilah "pohon keputusan", digunakan untuk membantu pengambilan keputusan dengan mempertimbangkan berbagai kemungkinan data seperti jenis acara dan jenis data. Pohon keputusan digunakan untuk menunjukkan cara kerja algoritma dalam kondisi yang terkendali, dan menjadi salah satu algoritma yang paling populer digunakan dalam penelitian dan operasi analisis data

c) *Bagging (Bootstrap Aggregating)*

Bagging merupakan teknik *ensemble learning* di mana beberapa model atau *decision tree* dibuat dengan menggunakan dataset yang sama namun diambil secara acak dengan pengambilan sampel yang dilakukan dengan penggantian. Teknik ini menggunakan metode pengambilan sampel acak dengan penggantian (*sampling with replacement*), di mana sampel-sampel tersebut dipilih secara acak dari data latih dan kemudian dimasukkan ke dalam setiap pohon dengan jumlah yang sama. (Wahyuningtyas et al., 2022:2975)

Bagging (*Bootstrap Aggregating*) adalah teknik *ensemble learning* di mana beberapa model dipelajari pada subset acak dari data latih dan

kemudian hasil prediksi mereka dikumpulkan dan digabungkan untuk membuat prediksi akhir yang lebih baik. Dalam *random forest*, sejumlah besar pohon keputusan dibuat pada subset acak dari data latih, dengan setiap pohon keputusan menggunakan subset yang berbeda dari fitur. Dalam setiap pohon keputusan, subset fitur dipilih secara acak untuk membuat pemilihan fitur yang lebih acak dan mengurangi ketergantungan pada fitur tertentu. Dengan menggunakan teknik *bagging* pada *random forest*, model ini mampu menghasilkan prediksi yang lebih stabil dan akurat daripada menggunakan satu pohon keputusan saja. Hal ini dikarenakan *random forest* dapat mengurangi *overfitting* dan meningkatkan kemampuan generalisasi model dengan menggunakan banyak pohon keputusan yang berbeda untuk membangun model.

d) Regresi *Random forest*

Menurut Fachid & Triayudi (2022:68) *Random forest Regression* adalah sebuah metode pembelajaran terawasi yang menggunakan teknik *ensemble learning* untuk melakukan regresi. Dalam teknik ini, beberapa prosedur pemecahan masalah dalam *machine learning* digabungkan untuk memberikan prediksi akurat, yang lebih baik daripada menggunakan model tunggal. Dengan demikian, *Random forest Regression* menggunakan metode pembelajaran ensemble untuk menggabungkan prediksi dari beberapa model dalam rangka menciptakan model regresi yang lebih efektif dan akurat.

Regresi *Random forest* merupakan sebuah teknik *machine learning* yang digunakan untuk melakukan regresi pada data numerik dengan menggunakan beberapa pohon keputusan yang dipelajari pada subset acak dari data latih. Setiap pohon keputusan pada *Random forest* dipelajari pada subset yang berbeda dari data latih dan menggunakan subset acak dari fitur. Selama proses prediksi, nilai keluaran dari setiap pohon keputusan digabungkan menggunakan rata-rata, yang kemudian menghasilkan prediksi akhir. Regresi *Random forest* sangat efektif dalam menangani data yang memiliki banyak fitur dan kompleksitas yang tinggi, serta dapat menghindari *overfitting* pada model. Oleh karena itu, teknik ini sering digunakan dalam berbagai aplikasi seperti pemodelan ekonomi, ilmu lingkungan, dan pengolahan citra.

e) *Ensemble Learning*

Berdasarkan pendapat Sudiyarno et al., (2021:4) *Ensemble learning* adalah gabungan dari beberapa learner atau *classifier* yang tidak begitu kuat dengan sebuah learner yang memiliki hasil prediksi yang baik. Model-model ini bekerja secara bersama-sama untuk menghasilkan prediksi yang lebih akurat daripada model tunggal. Teknik *ensemble learning* dapat diterapkan pada berbagai jenis model, seperti *decision tree, neural network*, dan *regression* model. Beberapa teknik *ensemble learning* yang populer adalah *bagging, boosting*, dan *stacking*.

f) *Variable Importance*

Variable importance (pentingnya variabel) merujuk pada ukuran yang digunakan untuk mengevaluasi seberapa signifikan suatu variabel dalam memprediksi variabel target dalam suatu model. Ini membantu dalam mengidentifikasi variabel mana yang paling berpengaruh dalam model dan dapat digunakan untuk seleksi variabel atau penafsiran model. Pada model *random forest*, beberapa metode digunakan untuk mengukur pentingnya variabel, seperti *Gini VIMP* dan *permutation VIMP*. (Lee et al., 2020)

Sementara itu, *permutation VIMP* adalah pengukuran pentingnya sebuah variabel dalam suatu model prediksi yang dihitung dengan menghitung penurunan rata-rata dalam akurasi OOB (*out-of-bag*) *decision trees* yang disebabkan oleh permutasi acak variabel tersebut. *Permutation VIMP* dianggap lebih dapat diandalkan daripada *Gini VIMP* karena tidak mengalami bias seperti *Gini VIMP*.

6. Pengertian *Confusion Matrix*

Confusion matrix adalah tabel yang berisi data uji yang telah diklasifikasikan dengan benar dan salah. *Confusion matrix* digunakan sebagai cara untuk mengevaluasi kinerja model yang telah dihasilkan. Terdapat empat istilah dalam *confusion matrix*, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Berikut adalah penjelasan dari empat istilah tersebut:

True Positive (TP) : kasus dimana objek diprediksi sesuai (*Positif*) kelasnya, dan sebenarnya memang sesuai (*True*) kelasnya.

True Negative (TN) : kasus dimana objek diprediksi tidak sesuai (*Negatif*) kelasnya dan sebenarnya memang tidak sesuai (*True*) kelasnya.

False Positive (FP) : kasus dimana objek diprediksi sesuai (*Positif*) kelasnya, dan ternyata tidak sesuai (*False*) kelasnya.

False Negative (FN) : kasus dimana objek diprediksi tidak sesuai (*Negatif*) kelasnya, dan dan ternyata sesuai (*True*) kelasnya.

Berdasarkan pendapat Pradika et al., (2020:134) *Confusion matrix* digunakan untuk melihat performa dari suatu model yang telah dibuat yaitu *accuracy, precision, recall,* dan *f1 score.*

Accuracy merupakan proporsi atau rasio dari jumlah prediksi yang benar dibandingkan dengan jumlah total data. Dalam hal ini, akurasi digunakan untuk menggambarkan seberapa banyak data objek yang telah diprediksi dengan benar sesuai dengan kelasnya dari keseluruhan data.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision dapat diartikan sebagai perbandingan antara hasil prediksi yang benar dengan keseluruhan hasil yang diprediksi sebagai positif. *Precision* mengukur seberapa banyak huruf hijaiyah yang sesuai dengan kelas yang diprediksi dari keseluruhan objek yang diprediksi sebagai positif.

$$Precision = \frac{TP}{TP + FP}$$

Recall merupakan rasio antara jumlah prediksi yang benar dengan keseluruhan data yang memang benar. *Recall* menjelaskan seberapa banyak persentase objek yang diprediksi sesuai dengan kelasnya dibandingkan dengan keseluruhan objek yang memang sesuai dengan kelasnya.

$$Recall = \frac{TP}{TP + FN}$$

F1 Score adalah sebuah metrik evaluasi yang mengkombinasikan rata-rata precision dan recall dengan pemberian bobot yang sama. Dalam *F1 Score*, nilai rata-rata *precision* dan *recall* digabungkan dan dihitung nilai *harmonic mean*-nya, sehingga memberikan nilai yang lebih representatif dari kedua metrik tersebut. Oleh karena itu, *F1 Score* dapat memberikan gambaran yang lebih baik tentang performa suatu model dalam mengklasifikasikan data.

$$F1\ Score = \frac{2 \times (Recall \times Precision)}{(Recall + Precision)}$$

7. *Python*

Python adalah bahasa pemrograman yang dinamis dan berorientasi objek yang dapat digunakan untuk berbagai jenis pengembangan perangkat lunak. Bahasa pemrograman ini menyediakan dukungan yang kuat untuk integrasi dengan bahasa pemrograman dan alat bantu lainnya. *Python* dilengkapi dengan pustaka standar yang dapat diperluas dan mudah dipelajari hanya dalam beberapa hari. *Python* dirancang dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode, dan diklaim sebagai bahasa yang menggabungkan kemampuan dan kapabilitas dengan sintaksis kode

yang sangat jelas serta fungsionalitas pustaka standar yang luas dan komprehensif. (Nugroho et al., 2020:16)

8. *Keras*

Berdasarkan pendapat Alwanda et al., (2020:49) *Keras* adalah perangkat lunak jaringan terbuka yang dibuat menggunakan bahasa pemrograman *Python*. Keras dapat dijalankan dengan *MXNet, Tensorflow, Deeplearning4j, theano,* atau *CNTK* untuk mempercepat eksperimen yang berhubungan dengan *deep learning*. Keras awalnya dibuat untuk proyek eksperimen bernama *Open-ended Neuro-Electronic Intelligent Robot Operating System*, yang ditulis dan dikembangkan oleh *Francois Chollet*, seorang insinyur di *Google*.

Keras adalah sebuah *library open-source* yang digunakan untuk membangun dan melatih model *neural network*. Keras menyediakan berbagai fitur yang memudahkan proses pembuatan model, termasuk *layer-layer neural network* yang telah diimplementasikan, fungsi aktivasi, *optimizer*, dan fungsi *loss function* yang dapat digunakan untuk menentukan tujuan pelatihan model. Keras juga menyediakan fitur untuk visualisasi performa model seperti *plot learning curves* dan *confusion matrix*.

9. *Scikit -learn*

Scikit-learn adalah sebuah modul *Python* yang mengintegrasikan berbagai macam algoritma *machine learning* terkini untuk masalah *medium-scale supervised* dan *unsupervised*. Modul ini fokus pada membawa *machine learning* ke non-spécialis dengan menggunakan bahasa pemrograman tingkat

tinggi yang umum digunakan. *Scikit-learn* mendukung *machine learning supervised* dan *unsupervised*, menyediakan berbagai algoritma untuk klasifikasi, regresi, *clustering*, dan reduksi dimensi. (Baranwal et al., 2019:2826)

Scikit-learn sangat diminati dalam penelitian akademik karena memiliki API yang terdokumentasi dengan baik, mudah digunakan, dan fleksibel. Pengguna dapat dengan mudah mencoba berbagai algoritma *machine learning* hanya dengan mengubah beberapa baris kode saja. *Scikit-learn* juga mengemas beberapa implementasi-algoritma *machine learning* terpopuler seperti LIBSVM dan LIBLINEAR. Beberapa pustaka *Python* lainnya, seperti NLTK, juga mengemas *scikit-learn* sebagai pembungkus. Selain itu, *scikit-learn* juga menyediakan berbagai dataset yang siap digunakan, sehingga pengguna dapat fokus pada pengembangan algoritma daripada menghabiskan waktu untuk mengumpulkan dan membersihkan data. (Hackeling, 2014:16).

10. *Tensorflow*

TensorFlow adalah sebuah perangkat lunak kerangka kerja komputasi yang dikembangkan dan didukung secara *open source* oleh *Google*. Kerangka kerja ini memungkinkan untuk mengimplementasikan berbagai jenis algoritma pembelajaran mendalam (*deep learning*) dengan dukungan yang baik untuk jaringan saraf konvolusional (*convolutional neural network*). Hal ini membuat *TensorFlow* menjadi salah satu pilihan utama para peneliti dan pengembang dalam mengembangkan aplikasi dan

sistem yang menggunakan kecerdasan buatan. (Ju et al., 2019:688). Sedangkan menurut Hikmatia A.E & Ihsan Zul (2021:76) *TensorFlow* adalah platform komputasi untuk membangun model pembelajaran mesin yang menyediakan berbagai *toolkit* untuk membuat model pada berbagai tingkat abstraksi, serta dapat menjalankan grafik pada berbagai *platform hardware* seperti CPU, GPU, dan TPU.

11. *Jupyter Notebook*

Jupyter Notebook adalah sebuah aplikasi *web open-source* yang memungkinkan pengguna untuk membuat dan berbagi dokumen yang mengandung kode, teks, gambar, dan visualisasi data interaktif. *Jupyter Notebook* dapat digunakan untuk berbagai keperluan, seperti analisis data, ilmu data, pembelajaran mesin, visualisasi data, dan banyak lagi. *Jupyter Notebook* berjalan pada lingkungan pemrograman *Python*, tetapi juga mendukung banyak bahasa pemrograman lainnya seperti R, Julia, dan Scala. Dalam *Jupyter Notebook*, pengguna dapat menulis dan menjalankan kode secara interaktif pada bagian-bagian yang disebut "sel". Setiap sel dapat berisi kode *Python* atau bahasa pemrograman lainnya, teks, atau visualisasi data.

Jupyter Notebook memiliki banyak kelebihan, antara lain:

1. Interaktif: Pengguna dapat mengeksekusi kode secara langsung dan melihat hasilnya secara interaktif di dalam dokumen.
2. Mudah digunakan: *Jupyter Notebook* memiliki antarmuka yang sederhana dan intuitif sehingga mudah digunakan oleh pemula.

3. Fleksibel: *Jupyter Notebook* dapat digunakan untuk berbagai keperluan, seperti analisis data, ilmu data, dan pembelajaran mesin.
4. Memudahkan berbagi: Pengguna dapat membagikan dokumen *Jupyter Notebook* secara mudah melalui berbagai media, seperti *email*, *GitHub*, dan lain-lain.
5. Mendukung banyak bahasa pemrograman: *Jupyter Notebook* mendukung banyak bahasa pemrograman, sehingga pengguna dapat menggunakan bahasa pemrograman yang paling sesuai untuk keperluan mereka.

Secara keseluruhan, *Jupyter Notebook* adalah alat yang sangat berguna dan populer di kalangan ilmuwan data dan pengembang perangkat lunak, yang dapat membantu mereka dalam melakukan analisis data, pembelajaran mesin, dan pengembangan perangkat lunak.

12. *Flask*

Flask merupakan kerangka kerja aplikasi website yang menggunakan bahasa pemrograman python dan menggunakan dependensi *Werkzeug* dan *Jinja2*. Dalam tahap pengemabangan sistem *hoax news detection* ini *flask* berfungsi sebagai pengontrol dalam pengembangan sistem *hoax news detection*. (Arbain et al., (2022:187).

Flask adalah sebuah kerangka kerja aplikasi web yang dibuat dengan bahasa pemrograman Python dan dirilis pertama kali pada tahun 2010 oleh *Armin Ronacher*. Flask dibangun dengan konsep sederhana dan fleksibel untuk membangun aplikasi web. *Framework* ini mengikuti

paradigma arsitektur web *Model-View-Controller* (MVC) dan tidak memerlukan peralatan atau pustaka tambahan untuk menjalankan aplikasi web. *Flask* bergantung pada dua pustaka *Python*, yaitu *Werkzeug* untuk pengembangan aplikasi web dan *Jinja2* untuk pemrosesan template HTML.

Flask adalah sebuah kerangka kerja aplikasi web yang sederhana dan fleksibel untuk membangun aplikasi web dengan menggunakan bahasa pemrograman *Python*. *Flask* memungkinkan pengembang untuk membangun aplikasi web yang ringan dan mudah di-maintain serta memperoleh fitur-fitur yang dibutuhkan melalui ekosistem ekstensi *Flask* yang kaya dan fleksibel.

13. UML (*Unified Modeling Language*)

UML adalah bahasa visual yang digunakan untuk pemodelan dan komunikasi mengenai sebuah sistem melalui diagram dan teks pendukung. Pemodelan UML mencakup beberapa jenis diagram seperti diagram *use case*, diagram *class*, diagram aktivitas, dan diagram urutan. Syarif & Nugraha (2020:65).

UML atau *Unified Modeling Language* merupakan sebuah bahasa yang digunakan untuk menentukan, visualisasi, konstruksi, dan dokumentasi suatu artifact dalam proses pembuatan perangkat lunak. Bahasa ini menyediakan notasi yang lengkap untuk membuat visualisasi model suatu sistem yang terdiri dari informasi dan fungsi, dan biasanya digunakan untuk pemodelan sistem komputer. UML tidak hanya digunakan dalam proses

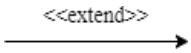
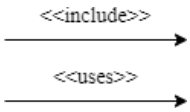
pemodelan perangkat lunak, tetapi juga dapat digunakan di hampir semua bidang yang memerlukan pemodelan. Febriani et al., (2020:125)

Maria & Lubis, (2020) Mengatakan bahwa beberapa diagram pada UML adalah sebagai berikut :

a) *Use Case Diagram*

Use Case Diagram adalah diagram untuk menunjukan perilaku bagaimana berbagai pihak yang akan menggunakan sistem akan berinteraksi satu sama lain.

Tabel 2.1 Simbol Use Case Diagram

Simbol	Notasi	Keterangan
	*Actor*	Pengguna sistem atau yang berinteraksi secara langsung dengan *usecase*.
	Assocation	Interaksi yang terjadi terhadap *actor* dan *user*.
	*Extend*	Relasi tambahan *use case* terhadap *use case* lain.
	Generalization	Menunjukan hubungan kearah *use case* yang lebih umum.
	*Uses/Include*	Relasi dua *use case, use case* yang dmembutuhkan tambahan untuk menjalankan *use case.*

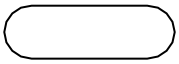
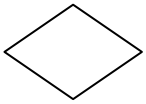
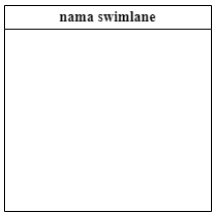
Sumber : Maria & Lubis, (2020)

b) *Activity Diagram*

Activity Diagram (diagram aktivitas) digunakan untuk menampilkan aliran aktifitas sistem atau proses bisnis, bagaimana proses dimulai,

kemungkinan keputusan yang mungkin diambil, dan bagaimana sistem akan berakhir

Tabel 2.2 Simbol Activity Diagram

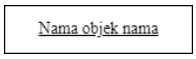
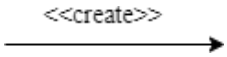
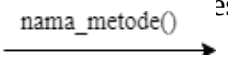
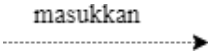
Simbol	Notasi	Keterangan
	Status Awal/akhir	Merupakan sebuah diagram aktifitas memiliki sebuah status awal.
	Aktivitas	Merupakan aktifitas yang dilakukan sistem, yang dimulai dengan kata kerja.
	*Decision*	Merupakan percabangan dimana ada pilihan aktivitas yang lebih dari satu.
	Join	Merupakan penggabungan yang mana lebih dari satu aktivitas lalu digabungkan menjadi satu.
	*Swimlane*	Merupakan pemisah organisasi bisnis yang memiliki tanggung jawab terhadap aktivitas yang terjadi.

Sumber : Maria & Lubis, (2020)

c) *Sequence Diagram*

Sequence Diagram digunakan untuk menujukan komunikasi antara objek dan menjelaskan interaksi pada sebuah sistem dalam bentuk urutan pesan yang dikirimkan antar objek.

Tabel 2.3 Simbol Sequence Diagram

Simbol	Notasi	Keterangan
	*Actor*	Merupakan peran orang atau sistem atau proses diluar sistem yang dibuat, yang berhubungan dengan sistem yang dibuat.
	Lifeline	Garis hidup objek yang menerangkan kehidupan dari objek tersebut.
	Objek	Merupakan interaksi pesan yang dilakukan oleh objek
	Waktu aktif	Menandakan interaksi obejek.
	Pesan tipe *create*	Pernyataan suatu objek membuatobjek lain, anak panah mengarah pada yang dibuat.
	esan *tipecall*	Merupakan penyataan satu objek memanggil metode atauproses pada objek lain.
	Pesan tipe *send*	Merupakan penyataan bahwa objek mengirimkan masukan atau data ke objek yang dikirim.

Sumber : Maria & Lubis, (2020)

14. *Blackbox Testing*

Pengujian fungsional atau *black box testing* adalah metode pengujian perangkat lunak yang digunakan untuk menguji fungsionalitas perangkat lunak tanpa memperhatikan detail struktur internal kode atau program yang

digunakan. Dengan metode ini, pengguna dapat menguji perangkat lunak dari luar tanpa perlu mengetahui bagaimana perangkat lunak tersebut diimplementasikan secara internal. Pengujian fungsional umumnya digunakan dalam tahap pengembangan perangkat lunak untuk memastikan bahwa fungsi-fungsi yang diharapkan dari perangkat lunak berjalan dengan baik. Metode ini sangat berguna untuk memastikan kehandalan dan kualitas perangkat lunak sebelum dirilis ke publik. (Wicaksono, 2021:48).

B. Kajian Empiris

Sebelum penelitian ini, telah ada beberapa penelitian yang dilakukan sebelumnya yang menggunakan Algoritma *Convolutional Neural Network* dan *Random Forest* untuk aplikasi tertentu. Penelitian tersebut dilakukan dengan menggunakan berbagai jenis data dan metode klasifikasi yang disesuaikan dengan kebutuhan peneliti. Penelitian mengenai metode *Convolutional Neural Network* dapat digunakan untuk menentukan mengklasifikasikan citra cuaca. Pada penelitian yang dilakukan oleh (Farid Naufal, 2021) dengan judul "Analisis Perbandingan Algoritma SVM, KNN, dan CNN untuk Klasifikasi Citra Cuaca" menyatakan bahwa CNN memiliki kinerja terbaik dalam mengklasifikasikan dataset cuaca dengan akurasi, presisi, recall, dan F1 score mencapai 0,942, 0,943, 0,942, dan 0,942 secara berturut-turut. Meski demikian, CNN memerlukan waktu eksekusi terlama, yakni sekitar 458,49 detik untuk mencapai kinerja terbaik.

Sebuah penelitian sejenis telah dilakukan untuk menggunakan metode *Convolutional Neural Network* dalam mengklasifikasikan buah

berdasarkan bentuknya. Penelitian tersebut dilakukan oleh (Kurniadi et al., 2021) dengan judul “Analisis Perbandingan Algoritma SVM dan CNN untuk Klasifikasi Buah” menyatakan bahwa warna dan tekstur merupakan fitur penting dalam klasifikasi jenis buah. Dalam penelitian ini, CNN berhasil mencapai akurasi sebesar 96,87%, lebih tinggi dibandingkan SVM yang hanya mencapai akurasi sebesar 93,09%. CNN menggunakan metode *subsampling* dengan *max pooling* dan dilatih pada data sebanyak 25 *epoch*. Hasil penelitian ini membuktikan bahwa CNN dapat digunakan untuk mengklasifikasikan jenis buah berdasarkan bentuknya dengan tingkat akurasi yang lebih tinggi daripada SVM.

Penelitian terkait penggunaan metode *Random forest* dalam analisis sentimen dilakukan oleh (Adrian et al., 2021) dengan judul “Perbandingan Metode Klasifikasi *Random forest* dan SVM Pada Analisis Sentimen PSBB” menyatakan bahwa bahasa yang beragam pada *Twitter* dapat menurunkan kemampuan model dalam memprediksi sentimen terkait PSBB. SVM dinilai lebih unggul dalam mengenali *tweet* yang dikategorikan sebagai "Positif". Kendala dalam jumlah data yang terbatas membatasi evaluasi penelitian ini. Peningkatan jumlah data diharapkan dapat membantu model dalam mengenali sentimen *tweet* dengan lebih baik. *Text vectorization* khusus bahasa Indonesia dan penggunaan algoritma *deep learning* dapat meningkatkan kualitas pelatihan model.

Pada penelitian yang dilakukan oleh (Nalatissifa et al., 2021) dengan judul “Perbandingan Kinerja Algoritma Klasifikasi *Naive Bayes, Support*

Vector Machine (SVM), dan *Random forest* untuk Prediksi Ketidakhadiran di Tempat Kerja" menyatakan bahwa ketiga algoritma tersebut dapat memberikan akurasi, presisi, dan recall di atas 96%. Selain itu, algoritma *Random forest* memberikan hasil akurasi, presisi, dan recall tertinggi dengan nilai akurasi mencapai 99,38%, presisi 99,42%, dan recall 99,39%. Hal ini menunjukkan bahwa algoritma *Random forest* lebih cocok digunakan untuk memprediksi ketidak hadiran di tempat kerja dibandingkan dengan algoritma Naïve Bayes dan SVM.

C. Kerangka Berpikir

Kerangka berpikir merupakan suatu model abstrak yang digunakan untuk membantu peneliti dalam merancang suatu penelitian. Kerangka berpikir tersebut dapat meliputi berbagai aspek seperti perumusan masalah, pendekatan penelitian, pengembangan sistem, dan hasil penelitian. Pada sub bab ini, akan dijelaskan kerangka berpikir yang berkaitan dengan variasi pada bentuk atau pola dari huruf aksara jawa dan solusi yang ditawarkan untuk mengatasinya.

Rumusan masalah pada penelitian ini adalah variasi pada bentuk atau pola dari huruf aksara jawa berpotensi untuk sulit dikenali maupun dihafalkan. Hal ini cukup menyulitkan bagi seseorang yang ingin belajar membaca aksara jawa khususnya bagi siswa sekolah dasar. Dalam konteks ini, penelitian bertujuan untuk mengembangkan suatu sistem yang dapat membantu seseorang dalam mengenali huruf aksara jawa dengan lebih mudah. Pendekatan penelitian yang digunakan dalam penelitian ini adalah perancangan dan pembangunan klasifikasi huruf aksara jawa mengacu pada algoritma *convolutional neural*

network dan *random forest*. Sistem dibangun dalam bentuk aplikasi *website* yang dapat diakses secara *online* oleh pengguna. Selain itu, pengembangan sistem dalam penelitian ini menggunakan pemodelan pengembangan sistem *Agile*. Pendekatan ini dipilih karena memungkinkan pengembangan sistem yang lebih *fleksibel* dan *responsif* terhadap perubahan kebutuhan pengguna.

Setelah tahap perancangan dan pembangunan, dilakukan implementasi sistem. Sistem ini dirancang agar memudahkan seseorang dalam mengenali huruf aksara jawa khususnya bagi siswa sekolah dasar maupun seseorang yang ingin belajar membaca aksara jawa. Implementasi dilakukan dengan menguji coba aplikasi pada sejumlah pengguna untuk mendapatkan umpan balik terkait kegunaan dan efektivitas sistem. Hasil penelitian menunjukkan bahwa aplikasi klasifikasi huruf aksara jawa berbasis *website* yang dikembangkan mampu mengenali dan mengklasifikasikan huruf aksara jawa dengan akurasi yang tinggi. Selain itu, aplikasi ini juga dianggap mudah digunakan dan dapat membantu pengguna dalam mempelajari huruf aksara jawa.

Dalam kesimpulan, penelitian ini mampu mengembangkan sistem klasifikasi huruf aksara jawa dengan metode *Convolutional Neural Network* dan *Random Forest* yang dapat membantu seseorang dalam mengenali huruf aksara jawa dengan lebih mudah. Hasil penelitian menunjukkan bahwa aplikasi klasifikasi huruf aksara jawa berbasis *website* yang dikembangkan mampu mengenali dan mengklasifikasikan huruf aksara jawa dengan akurasi yang tinggi serta dianggap mudah digunakan oleh pengguna.

Gambar kerangka berpikir dalam penelitian ini dapat dilihat pada gambar 2.6 dibawah ini:

Gambar 2.6 Kerangka berpikir