BAB II
TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Penelitian yang menerapkan *deep learning* dengan menggunakan CNN untuk klasifikasi teks telah banyak dilakukan dan diterapkan dalam berbagai bidang untuk menyelesaikan berbagai permasalahan. Berdasarkan penelitian sebelumnya, diketahui bahwa *deep learning* merupakan suatu kemajuan teknologi yang meniru fungsi otak manusia untuk menangani berbagai permasalahan. Dengan kemampuan tersebut, diperoleh peningkatan akurasi dalam pengenalan dan klasifikasi teks. Penjelasan hasil tinjauan dari penelitian sebelumnya dapat diuraikan sebagai berikut.

Penelitian [11] mengenai penerapan Analisis Sentimen dengan CNN. Tujuan penelitian tersebut adalah untuk menganalisis pendapat masyarakat terkait produk di restoran tertentu. Penelitian ini menggunakan kombinasi model CNN dan *contextualized word embedding*, yang kemudian dibandingkan dengan kombinasi model CNN dan *traditional word embedding*. Pengujian klasifikasi dilakukan menggunakan tiga model, yaitu BERT-CNN, ELMo-CNN, dan Word2vec-CNN. Hasil penelitian menyatakan bahwa klasifikasi sentimen memberikan hasil terbaik pada model BERT-CNN, dengan nilai *precision* sebesar 0.89, *recall* sebesar 0.89, dan *f1-score* sebesar 0.91.

Penelitian [12] tentang penerapan analisis sentimen pada gunung semeru menggunakan menggunakan data dari *Google Maps User Reviews.* Model klasifikasi yang digunakan adalah menggunakan *SVM, complement naïve bayes, logistic regression,* dan *transfer learning* dari *pre-trained BERT, IndoBERT* dan *mBERT.* Kemudian untuk aspek yang digunakan pada penelitian tersebut adalah atraksi, fasilitas, akses, dan harga. Berdasarkan hasil percobaan menunjukkan bahwa *transfer learning* dari model IndoBERT menghasilkan hasil yang baik dengan akurasi dan F1-Score berturut-turut mencapai 91,48% dan 71,56%. Selain itu, di antara berbagai model lainnya yang digunakan, model SVM memberikan

hasil terbaik dengan akurasi sebesar 89,16% dan *f1-score* sebesar 62,23% [12].

Penelitian [13] analisis sentimen menggunakan metode STM, LSTM-CNN, CNN-LSTM dan *Word2Vec* pada media *online* dengan *dataset* yang diambil dari artikel, yaitu dari *Detik Finance.* Penelitian tersebut bertujuan untuk melakukan klasifikasi berdasarkan sentimen positif dan negatif. Hasil dari penelitian tersebut menunjukkan bahwa pengujian dengan metode LSTM (*Long Short Term Memory*) mempunyai akurasi 62%, LSTM-CNN 65% dan CNN-LSTM mempunyai 74% [13].

Penelitian [14] tentang perbandingan pengaruh panjang kalimat dalam analisis sentimen menggunakan SVM dan CNN. Dalam penelitian [14] menyatakan bahwa penggunaan panjang kalimat pada dataset akan mempengaruhi performa pada algoritma SVM dan CNN jika menggunakan *word2vec,* sedangkan jika menggunakan model SVM dengan TFIDF performa tidak begitu terpengaruh oleh panjang kalimat. Meskipun begitu, kombinasi metode SVM+TFIDF memiliki waktu proses yang cepat dibandingkan metode lainnya. Adapun metode CNN+Word2vec menghasilkan performansi yang baik dengan nilai akurasi sebesar 0,94%, presisi sebesar 0,95%, *recall* sebesar 0,96%, dan *f1-score* sebesar 0,95%.

Penelitian [15] tentang optimisasi *sentiment analysis* dengan CNN dengan kombinasi *text preprocessing* dan *word2vec* yang dilakukan [15]. Penelitian tersebut bertujuan untuk melakukan identifikasi model yang paling akurat dalam melakukan *sentiment analysis,* data yang dipakai menggunakan 20.986 ulasan dari 720 produk di *marketplace shopee,* pada penelitian tersebut menemukan bahwa kombinasi teknik *preprocessing* ketiga *(case folding, punctuation removal, word normalizer, dan stemming)*, menyatakan bahwa kombinasi parameter *word2vec* kedua (size 50, window 2, hs 0, dan negatif 10), dan kombinasi parameter CNN keempat (*kernel size 2, dropout 0.2, dan learning rate 0.01)* memiliki akurasi terbaik sebesar 99.00%, presisi 98.96%, dan recall 98.96%.

Penelitian [23] membahas tentang penggunaan NLP untuk keperluan analisis sentimen. Penelitian [23] menggunakan beberapa model *deep learning*, yaitu CNN, RNN (*Recurrent Neural Network*) dan LSTM. Tujuan dari penelitian ini adalah mengetahui perbedaan performa dari model *deep learning*. Hasil dari penelitian ini menunjukkan bahwa penggunaan CNN menempati posisi terbaik dengan akurasi *training* sebesar 97%, kemudian untuk posisi kedua ada pada bidirectional LSTM dengan nilai akurasi *training* mencapai 94% [23].

Berbagai penelitian telah menunjukkan bahwa penggunaan CNN meningkatkan akurasi model, seperti pada analisis sentimen berbasis aspek pada ulasan restoran di Indonesia [11], analisis sentimen produk di pasar *online* [15] dan perbandingan model *deep learning* untuk klasifikasi *sentiment analysis* [23].

Selain itu, penggunaan *Word2Vec* sebagai metode *word embedding* juga terbukti efektif dalam meningkatkan akurasi model klasifikasi, seperti pada analisis sentimen dengan LSTM-CNN pada media *online* [13]. Temuan ini mengindikasikan potensi penggunaan CNN dan *Word2Vec* dalam meningkatkan akurasi model dalam berbagai konteks analisis sentimen.

Tabel 2.1 Penelitian Terdahulu

No	Peneliti, Tahun	Judul	Tujuan	Metode	Hasil Dan Kesimpulan	Perbedaan Dengan Penelitian Ini
1.	P. R. Amalia and E. Winarko, 2021	Aspect-Based Sentiment Analysis on Indonesian Restaurant Review Using a Combination of Convolutional Neural Network and Contextualized Word Embedding	penelitian ini melakukan aspek-analisis sentimen berdasarkan ulasan restoran Indonesia menggunakan kombinasi Convolutional Neural Network dan Contextualized Word Embedding	*klasifikasi menggunakan Convolutional Neural Network* dan Vektorisasi teks menggunakan *Contextualized Word Embedding*	Hasil dari penelitian tersebut adalah menyatakan bahwa klasifikasi sentimen memberikan hasil terbaik pada model BERT-CNN dengan nilai precision sebesar 0.89, recall sebesar 0.89, dan f1-score sebesar 0.91.	Datasets, aspek yang diklasifikasi.
2.	C. A. Bahri and L. H. Suadaa, 2023	Aspect-Based Sentiment Analysis in Bromo Tengger Semeru National Park Indonesia Based on Google Maps User Reviews	Melakukan analisis sentimen pada obyek wisata di gunung bromo berdasarkan data dari ulasan di *Google Maps*	SVM, *Complement Naïve Bayes, Logistic Regression, dan transfer learning*	Hasil pengujian dengan metode CNN, didapatkan nilai akurasi sebesar 98.66% dibanding dengan algoritma Naive Bayes yang mendapatkan nilai akurasi sebesar 94.66%.	Datasets, aspek yang diklasifikasi, model machine learning.
3.	D. T. Hermanto,	Algoritma LSTM-CNN untuk	Melakukan klasifikasi sentimen pada dataset	LSTM, LSTM-CNN,	Berdasarkan hasil pengujian, Double Max	Perbedaan yang ada pada penelitian ini ada

No	Peneliti, Tahun	Judul	Tujuan	Metode	Hasil Dan Kesimpulan	Perbedaan Dengan Penelitian Ini
	A. Setyanto, and E. T. Luthfi, 2021	Sentimen Klasifikasi dengan Word2vec pada Media Online	media online	CNN-LSTM	CNN mendapatkan hasil klasifikasi dengan baik dengan akurasi sebesar 98%, Recall 97%, Precision 98% dan F1-Score 98%.	pada datasets yang digunakan untuk melakukan klasifikasi.
4.	A. Pambudi and S. Suprapto, 2021	Effect of Sentence Length in Sentiment Analysis Using Support Vector Machine and Convolutional Neural Network Method	Melakukan percobaan untuk mengetahui apakah panjang kalimat mempengaruhi performa SVM dan CNN	*Support Vector Machine* dan *Convolutional Neural Network Method*	Hasil pengujian metode CNN+Word2vec menghasilkan performansi terbaik dengan nilai akurasi sebesar 0,94%, presisi sebesar 0,95%, recall sebesar 0,96%, dan f1-score sebesar 0,95%.	Pada penelitian ini menggunakan menggunakan SVM dan CNN sebagai percobaannya.
5.	E. Utami, 2023	Optimizing Sentiment Analysis of Product Reviews on Marketplace Using a Combination of Preprocessing Techniques, Word2vec, and Convolutional	Melakukan optimisasi analisis sentimen menggunakan CNN dan Word2Vec	*Convolutional Neural Network* dan *Word2Vec*	Hasil penelitian tersebut menemukan bahwa kombinasi teknik preprocessing ketiga (case folding, punctuation removal, word normalizer, dan stemming), menyatakan	Pada penelitian ini perbedaannya ada pada datasets yang digunakan.

No	Peneliti, Tahun	Judul	Tujuan	Metode	Hasil Dan Kesimpulan	Perbedaan Dengan Penelitian Ini
		Neural Network			bahwa kombinasi cnn dan word2vec mendapatkan hasil yang baik, yaitu akurasi 99.00%, presisi 98.96% dan *recall* sebesar 98.96%.	
6.	F. P. Rachman, H. Santoso, 2021	Perbandingan Model *Deep Learning* untuk Klasifikasi Sentiment Analysis dengan Teknik Natural Language Processing	Mengetahui performa dari CNN, LSTM dan RNN dalam klasifikasi teks.	CNN, LSTM dan RNN	Hasil menunjukkan bahwa penggunaan CNN dalam klasifikasi teks lebih unggul dari pada model *deep learning* lainnya	Dataset dan tujuan penelitian.

Tabel 2.1 merangkum hasil penelitian yang telah dilakukan sebelumnya, dengan data yang terdokumentasi dalam jurnal-jurnal terkait. Informasi yang terkandung dalam jurnal-jurnal tersebut memberikan pandangan yang berharga untuk mendukung penelitian yang akan dilaksanakan. Rentang topik yang dicakup melibatkan berbagai aspek yang terkait dengan subjek penelitian yang akan dijalankan. Analisis dari penelitian sebelumnya memberikan pemahaman yang lebih mendalam mengenai pemanfaatan cnn dan Word2Vec dalam konteks klasifikasi sentimen berbasis aspek. Metode yang digunakan memberikan gambaran hasil penelitian yang telah dilakukan sebelumnya, membuka pintu untuk peluang-peluang baru dalam penelitian masa depan.

Pada penelitian yang telah dilakukan oleh [11], membandingkan 3 algoritma cnn, yaitu BERT-CNN, ELMO-CNN dan Word2vec-CNN. Dari ketiga model cnn tersebut didapatkan bahwa model dengan menggunakan BERT-CNN mendapatkan akurasi yang bagus, yaitu *precision* sebesar 0.89, *recall* sebesar 0.89, dan *f1-score* sebesar 0.91. Kemudian penelitian yang dilakukan oleh [14], menjelaskan bahwa penggunaan algortima cnn dengan word2vec mendapatkan hasil yang cukup baik, yaitu mendapatkan nilai akurasi nilai akurasi sebesar 0,94%, presisi sebesar 0,95%, *recall* sebesar 0,96%, dan *f1-score* sebesar 0,95%. Penelitian [15] tentang kombinasi *preprocessing,* cnn dan word2vec mendapatkan hasil yang cukup baik dalam melakukan klasifikasi pada *dataset marketplace,* yaitu mendapatkan akurasi 99.00%, presisi 98.96% dan *recall* sebesar 98.96%.

2.2. Landasan Teori

2.2.1 *Sentiment Analysis*

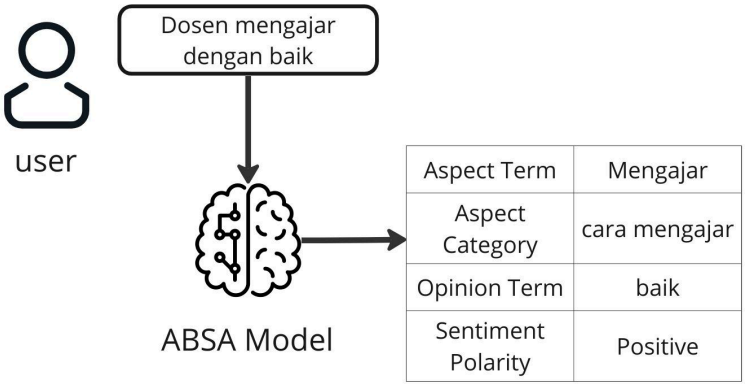
Sentiment analysis (analisis sentimen) adalah sebuah teknik pengolahan bahasa alami (*natural language processing*) yang bertujuan untuk mengidentifikasi dan mengekstrak sentimen, opini, atau perasaan dari teks atau dokumen tertentu, seperti ulasan produk, postingan media sosial, atau email. Dalam analisis sentimen, komputer menggunakan algoritma dan metode klasifikasi untuk mengenali, mengelompokkan, dan memperkirakan sentimen yang terkandung dalam dokumen teks. Sentimen yang dianalisis bisa berupa positif atau negatif tergantung pada konteks dan kata-kata yang digunakan [8].

2.2.2 *Aspect Based Sentiment Analysis*

Aspect-based sentiment analysis (ABSA) adalah metode untuk mengidentifikasi sentimen yang diungkapkan pada aspek atau fitur-fitur pada teks tertentu. Ini bertujuan untuk memberikan pemahaman sentimen yang lebih baik dengan mengaitkannya dengan berbagai aspek yang disebutkan dalam ulasan, umpan balik atau postingan media sosial[3]. Ada empat kategori dalam ABSA, yaitu :

a. *Aspect Category*: Mendefinisikan aspek unik dari suatu entitas dan yang seharusnya termasuk dalam kumpulan kategori, yang telah ditentukan sebelumnya untuk setiap domain aspek tertentu.
b. *Aspect Term*: Target opini dapat muncul dengan jelas dalam teks yang diberikan, seperti contohnya adalah kata "mengajar" dalam kalimat "Dosen mengajar dengan baik." Namun, terkadang target opini dinyatakan secara tidak langsung, kita dapat mengidentifikasi istilah aspek sebagai istilah khusus.
c. *Opinion Term*: Merupakan cara bagi pemegang opini untuk menyampaikan sentimen mereka terhadap subjek tertentu. Sebagai contoh ada pada gambar 2.1.
d. *Sentiment Polarity*: Merupakan orientasi sentimen terhadap suatu kategori aspek atau istilah aspek, yang biasanya tergolong positif, negatif, dan

netral. Ilustrasi pada gambar 2.1 merupakan contoh dari ABSA.



Gambar 2.1 Ilustrasi ABSA [3]

2.2.3 *Convolutional Neural Network* (CNN)

CNN adalah jenis arsitektur jaringan saraf yang terinspirasi oleh cara kerja sistem syaraf khususnya manusia. CNN dirancang khusus untuk memproses data yang memiliki struktur spasial, seperti data gambar atau data video[6][9]. Menurut[6] arsitektur CNN terdiri dari sejumlah lapisan (atau biasa disebut *multi-building blocks*). Setiap lapisan dalam arsitektur CNN, termasuk fungsinya, dijelaskan secara rinci di bawah ini.

2.2.3.1 *Convolutional Layer* (CONV)

Convolutional Layer (CONV) merupakan bagian inti dari arsitektur CNN, adanya layer tersebut terdiri dari sekumpulan filter atau biasa disebut *kernel*. Setiap kernel dalam CONV memiliki dimensi panjang x tinggi, dengan ukuran dimensi pada umumnya adalah 3x3, 5x5 dan 7x7. Dalam CNN yang digunakan untuk *Natural Language Processing* (NLP), lapisan convolutional menggunakan kernel (juga disebut sebagai *filter*) untuk mengekstraksi fitur yang berarti dari teks masukan. Kernel ini memainkan peran penting dalam menangkap pola lokal dan mendeteksi struktur linguistik yang relevan. Selama operasi CONV, *kernel* akan mengalami pergeseran pada bagian *text input*. Pergeseran akan mulai melakukan

operasi AND pada setiap fitur di antara nilai filter dan input. Sehingga akan menghasilkan nilai skalar tunggal yang dikenal sebagai *feature map* atau *activation map*.

2.2.3.2 *Activation Function*

Activation Function memiliki peran utama dalam memetakan *input* ke *output*. Nilai *input* dihasilkan dengan mengalikan bobot *input* dengan jaringan dan menambahkan bias (jika ada). Fungsi aktivasi kemudian bertanggung jawab untuk mengambil keputusan apakah harus mengaktifkan atau tidak mengaktifkan jaringan berdasarkan *input* tertentu, serta menghasilkan *output* yang sesuai. Fungsi aktivasi diaplikasikan setelah lapisan-lapisan awal seperti lapisan *fully-connected* dan lapisan konvolusi.

a. *Sigmoid*

Sigmoid adalah jenis fungsi aktivasi yang menerima *input* bilangan *real* dan menghasilkan output terbatas antara nol dan satu. Fungsi *sigmoid* bersifat monotonik dengan rentang *output* yang konstan, sehingga cocok digunakan dalam proses klasifikasi [20]. Namun, penggunaan *sigmoid* membutuhkan waktu komputasi yang lebih lama. Bentuk kurva *sigmoid* menyerupai huruf S dan dapat dijelaskan secara matematis dengan rumus berikut:

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

Keterangan :

e = Euler / 2.71828

x = Output dari *Neural Network*

b. Tanh

Hampir sama dengan fungsi aktivasi *sigmoid* karena fungsi ini sama-sama menerima masukan dengan bilangan *real,* tetapi output dari fungsi ini dibatasi dengan jumlah antara -1 dan 1. Perhitungan *softmax* dapat dirumuskan sebagai berikut:

$$f(x)tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (2.2)$$

Keterangan :

e = Euler / 2.71828

x = Output dari *Neural Network*

c. **ReLU** (*Rectified Linear Unit*)

ReLU adalah fungsi aktivasi yang paling umum digunakan dalam CNN. ReLU mengubah nilai input secara keseluruhan menjadi nilai positif. Meskipun probabilitas yang dihasilkan hampir sama dengan *sigmoid*, ReLU memiliki kinerja yang lebih baik. Rumus yang digunakan sangat sederhana, yaitu:

$$f(x)ReLu = max(0, z) \quad (2.3)$$

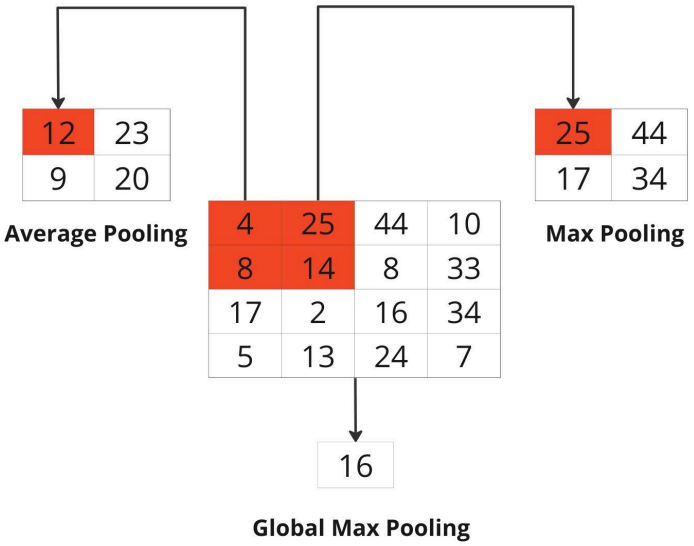
Keterangan :

$f(x)Relu$ = Ini mewakili output dari fungsi aktivasi ReLU dengan nilai input x.

$max(0, z)$ = Ini memilih nilai maksimum antara 0 dan input x,

2.2.3.3 *Pooling Layer*

Fungsi utama dari *pooling layer* adalah melakukan *sub-sampling* pada *feature map* yang dihasilkan oleh *convolutional layer*. *Pooling layer* akan mengurangi ukuran *feature map* yang besar, sambil tetap mempertahankan fitur-fitur yang dominan. Beberapa jenis *pooling layer* yang umum digunakan antara lain *tree pooling*, *gated pooling*, *average pooling*, *min pooling*, *max pooling*, *global average pooling* (GAP), dan *global max pooling*. Di antara jenis-jenis tersebut, *min pooling, max pooling*, dan *global average pooling* (GAP) merupakan yang paling populer dan sering digunakan. Ilustrasi dari ketiga jenis *pooling layer* ada pada Gambar 2.3.



Gambar 2.2 Ilustrasi dari ketiga jenis pooling layer [6]

a. *Average Pooling*

Proses *Average Pooling* melibatkan pemilihan jendela (biasanya berukuran 2x2 atau 3x3) yang bergerak secara bersilangan di seluruh gambar atau fitur. Di setiap jendela, nilai rata-rata dari elemen-elemen di dalamnya dihitung. Hasilnya kemudian digunakan sebagai nilai yang akan mewakili jendela tersebut dalam lapisan yang lebih dalam dari jaringan saraf [31].

b. *Global Max Pooling*

Global Max Pooling adalah variasi dari *Max Pooling*, seluruh matriks atau fitur *map* direduksi menjadi satu nilai tunggal. Berbeda dengan *Max Pooling* yang menggunakan jendela bergerak, *Global Max Pooling* mengambil nilai maksimum dari seluruh matriks atau fitur *map*. Hal ini menghasilkan satu nilai untuk setiap *channel* atau fitur yang merepresentasikan fitur paling dominan dari seluruh *input* [31].

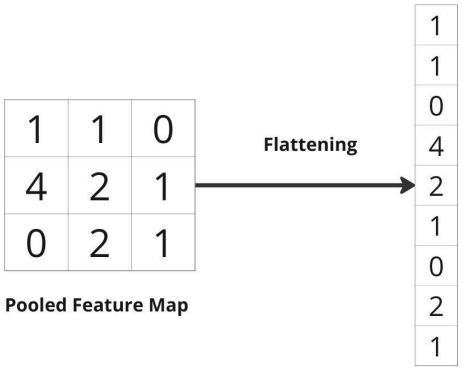
c. *Max Pooling*

Max Pooling adalah operasi *pooling* yang umum digunakan dalam jaringan saraf tiruan. Tujuannya adalah untuk mereduksi dimensi spasial dari representasi gambar atau fitur dengan memilih nilai maksimum di dalam jendela bergerak. Prosesnya mirip dengan *Average Pooling*, namun yang diambil bukan nilai rata-rata, melainkan nilai maksimum dari setiap

jendela [31].

2.2.3.4 *Flatten*

Lapisan *Flatten* memiliki dua fungsi utama, yaitu mengubah citra dua dimensi menjadi satu dimensi dan menghapus beberapa bias yang dapat memperlambat proses pembelajaran (*training*) [21]. Gambar 2.3 merupakan ilustrasi dari flatten dalam CNN.



Gambar 2.3 Ilustrasi Flattening [21]

2.2.3.5 *Fully Connected Layer* (FC)

FC (Fully Connected) Layer, yang terletak di akhir arsitektur CNN, memiliki perbedaan dengan lapisan CONV. Pada FC Layer, semua neuron terhubung penuh ke semua aktivitas di lapisan sebelumnya, berbeda dengan CONV *layer* yang hanya terhubung ke lapisan *input*. FC *Layer* umumnya berperan sebagai pengklasifikasi dalam CNN. Input pada FC *Layer* berupa vektor, yang dapat berasal dari lapisan *pooling* atau lapisan konvolusi terakhir dalam CNN. FC Layer mengolah *input* ini dengan melakukan operasi matematika seperti perkalian matriks dan penjumlahan bobot. *Output* dari FC *Layer* merupakan output akhir dari arsitektur CNN, yang digunakan untuk tujuan mengklasifikasi atau prediksi[6].

2.2.4 TensorFlow

TensorFlow adalah sebuah *library* sumber terbuka yang digunakan untuk membangun dan melatih model pembelajaran mesin (*machine learning*) dan

jaringan saraf (*neural network*). Dikembangkan oleh tim Google *Brain*, TensorFlow menyediakan antarmuka pemrograman yang kaya dan fleksibel untuk mengimplementasikan berbagai jenis algoritma pembelajaran mesin. Pada dasarnya, TensorFlow memodelkan komputasi sebagai aliran data melalui grafik komputasi. Grafik tersebut terdiri dari *node-node* yang mewakili operasi matematika dan perhitungan, serta *edge-edge* yang mewakili aliran data antara *node-node* tersebut. Data yang digunakan dalam TensorFlow diwakili sebagai tensor, yang merupakan kumpulan nilai yang memiliki dimensi dan bentuk tertentu [18].

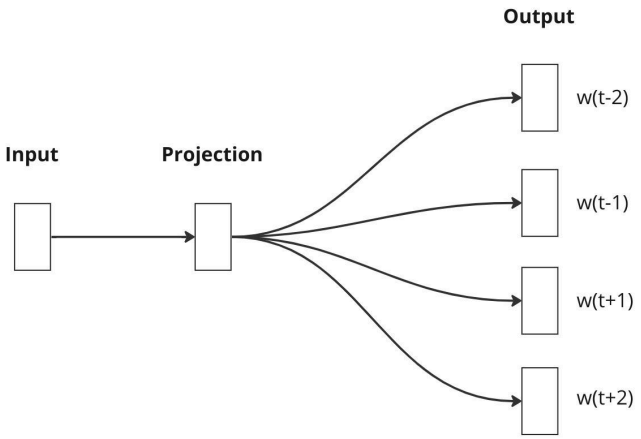
2.2.5 Keras

Keras adalah sebuah *library open-source* untuk Python yang digunakan untuk membangun dan melatih jaringan saraf tiruan (*neural networks*). Keras sangat populer dan banyak digunakan dalam komunitas machine learning karena mudah digunakan, efisien, dan fleksibel. Dalam Keras, kita dapat membangun model *neural network* dengan mudah dengan menentukan arsitektur model menggunakan beberapa layer atau lapisan (seperti *layer input*, *layer* tersembunyi, dan *layer output*) dan fungsi aktivasi yang sesuai. Kita juga dapat menentukan jenis optimisasi dan fungsi *loss* yang digunakan selama proses pelatihan model [19].

2.2.6 Word2vec

Word2Vec adalah sebuah metode populer dalam NLP yang digunakan untuk mempelajari representasi vektor kata. Tujuannya adalah untuk mengubah kata-kata menjadi vektor numerik yang dapat digunakan untuk menganalisis dan memahami makna kata-kata dalam konteks yang lebih luas. Ide dasar dibalik Word2Vec adalah kata-kata yang sering muncul dalam konteks yang serupa memiliki makna yang mirip. Dengan menggunakan model Word2Vec, kata-kata yang memiliki kemunculan dan konteks serupa akan memiliki representasi vektor yang berdekatan dalam ruang vektor. Dengan kata lain, kata-kata yang serupa secara semantik akan berada dalam kedekatan spasial dalam representasi vektor

[10].



Gambar 2.4 Ilustrasi Word Embedding dengan CBOW [10]

2.2.7 *Confusion Matrix*

Confusion matrix adalah sebuah tabel yang digunakan untuk menggambarkan performa dari sebuah model klasifikasi. Matriks ini memberikan gambaran tentang sejauh mana model tersebut mampu mengklasifikasikan *instance* dari setiap kelas dengan benar. *Confusion matrix* dapat digunakan dalam evaluasi model klasifikasi yang memiliki lebih dari dua kelas target dan juga dapat digunakan dalam klasifikasi biner [17]. *Confusion matrix* dapat didefinisikan sebagai tabel yang merangkum jumlah deteksi yang benar dan deteksi yang salah yang dihasilkan oleh model. *Confusion matrix* ini berbentuk matriks persegi dengan ukuran *n x n*, dengan *n* menunjukkan jumlah kelas target. Biasanya, setiap kolom pada *confusion matrix* mewakili kelas aktual atau kelas sebenarnya, sedangkan setiap baris pada confusion matrix mewakili kelas hasil deteksi [22]. Gambar 2.6 merupakan ilustrasi dari *confusion matrix*.

		Kelas Hasil Deteksi	
		Positif	Negatif
Kelas Aktual	Positif	TP	FN
	Negatif	FP	TN

Gambar 2.5 Confusion Matrix [22]

Keterangan :

1. TP (*True Positive*) merupakan jumlah data yang kelas aktual dan kelas hasil deteksinya adalah kelas positif.
2. FN (*False Negative*) merupakan jumlah data yang kelas aktualnya adalah kelas positif tapi terdeteksi sebagai kelas negatif.
3. FP (*False Positive*) adalah jumlah data kelas aktualnya adalah kelas negatif namun terdeteksi kelas positif.
4. TN (*True Negative*) merupakan jumlah data kelas aktual dan kelas hasil yang terdeteksi adalah negatif.

Untuk mengevaluasi keakuratan dan kinerja model dalam proses klasifikasi, dapat menggunakan *confusion matrix* untuk menghitung beberapa metrik seperti akurasi (*accuracy*), presisi (*precision*), *recall*, dan skor F1[22].

a. Accuracy

Akurasi (*accuracy*) adalah ukuran yang menggambarkan sejauh mana model yang dibuat dapat mengklasifikasikan data dengan benar, atau seberapa dekat nilai prediksi dengan nilai sebenarnya. Akurasi dapat dihitung sebagai rasio antara jumlah prediksi benar positif (*true positive*) dengan jumlah total data yang ada, dalam konteks *confusion matrix* di

mana hasil klasifikasi model dikelompokkan menjadi empat kategori (*True Positive, True Negative, False Positive,* dan *False Negative*), akurasi dapat dihitung dengan rumus [22].

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.4)$$

b. Precision

Precision merupakan ukuran yang menggambarkan tingkat keakuratan antara data yang benar positif (*true positive*) yang diminta dengan hasil deteksi yang diberikan oleh model klasifikasi, atau dengan kata lain seberapa banyak data yang terdeteksi sebagai kelas positif secara tepat. Nilai *precision* berguna dalam menentukan kehandalan model yang telah dibuat. Secara matematis, precision dapat dihitung sebagai rasio antara *true positive* dengan total hasil yang terdeteksi positif [22].

$$Precision = \frac{TP}{TP + FP} \quad (2.5)$$

c. Recall

Recall, juga dikenal sebagai *sensitivity* atau *true positive rate*, menggambarkan seberapa baik model dapat mendeteksi data kelas aktual yang positif. Nilai *recall* mencerminkan kemampuan model dalam mengidentifikasi kembali informasi yang relevan. Secara matematis, *recall* dapat dihitung sebagai rasio antara *true positive* dengan total data yang sebenarnya positif [22].

$$Recall = \frac{TP}{TP + FN} \quad (2.6)$$

d. *F1-Scor*e

F1-Score adalah nilai yang mencerminkan perbandingan rata-rata antara presisi (*precision*) dan recall secara bersamaan. *F1-Score* memberikan gambaran tentang seberapa baik nilai presisi dan recall dapat dicapai oleh model. Nilai maksimum *F1-Score* dapat tercapai jika nilai presisi dan recall memiliki nilai yang sama. Dengan demikian, semakin mendekati nilai *F1-Score* dengan 1, semakin baik kinerja model klasifikasi tersebut [22]. *F1-score* dapat dirumuskan sebagai berikut :

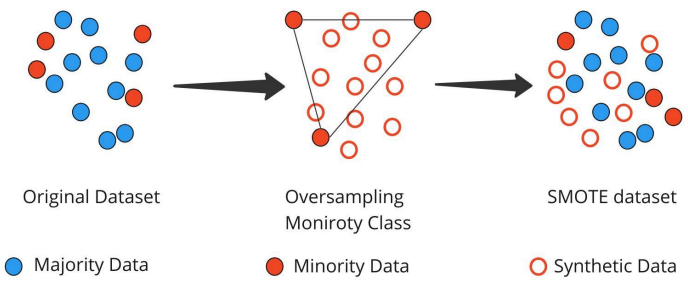
$$F1 - Score = \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \quad (2.7)$$

2.2.8 *Oversampling*

Oversampling adalah suatu metode dalam analisis data yang bertujuan untuk menyeimbangkan distribusi kelas pada dataset. Dalam konteks klasifikasi, *oversampling* dilakukan dengan meningkatkan jumlah sampel dari kelas minoritas agar sebanding dengan jumlah sampel dari kelas mayoritas. Hal ini membantu mencegah model machine learning cenderung memihak kelas mayoritas dan meningkatkan performa dalam mengenali kelas minoritas. Salah satu cara *oversampling* yang umum adalah

a. SMOTE (*Synthetic Minority Over-sampling Technique*)

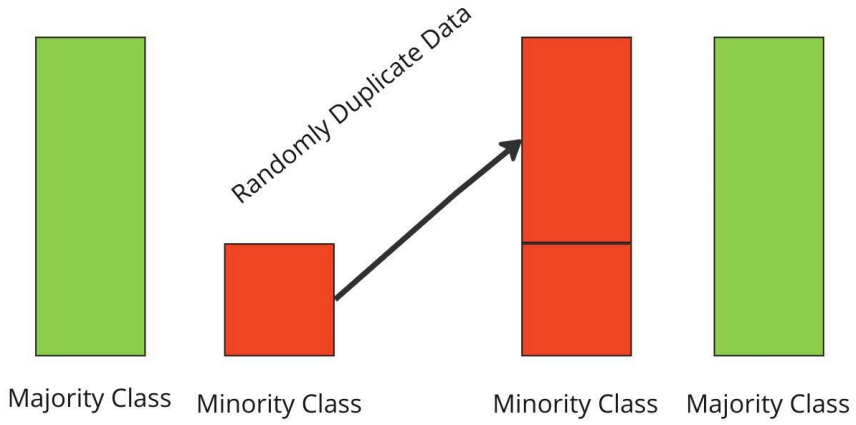
SMOTE adalah teknik dalam *machine learning* untuk menangani ketidakseimbangan kelas. Cara kerjanya melibatkan pemilihan instansi minoritas, identifikasi tetangga terdekat, dan pembuatan instansi sintetis dengan mempertimbangkan perbedaan nilai fitur. Langkah terakhirnya adalah menghubungkan instansi asli dengan instansi sintetis, menciptakan dataset yang seimbang. SMOTE membantu mencegah bias model terhadap kelas mayoritas dan meningkatkan kemampuan model untuk menggeneralisasi pada contoh kelas minoritas. Meskipun berguna, penggunaan SMOTE memerlukan pertimbangan dan evaluasi hati-hati pada dampaknya terhadap kinerja model [26]. Berikut merupakan ilustrasi dari SMOTE:



Gambar 2.6 Ilustrasi SMOTE [26]

b. *Random Oversampling* (RO)

RO adalah teknik *oversampling* dengan instansi dari kelas minoritas secara acak dipilih dan disalin untuk meningkatkan jumlahnya. Tujuannya adalah menciptakan keseimbangan antara kelas mayoritas dan minoritas dengan meningkatkan proporsi instansi kelas minoritas. Meskipun sederhana, metode ini dapat membantu model machine learning untuk memahami dan memprediksi kelas minoritas dengan lebih baik [24]. Berikut merupakan ilustrasi dari *random oversampling*:



Gambar 2.7 Ilustrasi Random Oversampling [24]

c. ADASYN (*Adaptive Synthetic Sampling*)

ADASYN adalah varian dari teknik SMOTE yang dirancang untuk menangani ketidakseimbangan kelas dalam *machine learning*. ADASYN memperkenalkan elemen adaptif dengan memberikan bobot pada setiap instansi minoritas berdasarkan tingkat kesulitannya dalam klasifikasi. Ini berarti instansi yang lebih sulit untuk diklasifikasikan menerima lebih banyak perhatian dalam pembuatan instansi sintetis [26][29]. Berikut merupakan algoritma dari ADASYN:

I. Misalkan kita memiliki dataset pelatihan D dengan m sampel (Xi, Yi), di mana $i = 1..m$. Setiap X_i adalah contoh dalam ruang

fitur berdimensi X, dan Y_i adalah kelas label yang terkait dengan X_i , dengan $Yi \in Y = \{1, -1\}$. Kita definisikan ms sebagai kelas minoritas dan ml sebagai kelas mayoritas, sementara jumlah total sampel adalah $m = ms + ml$.

II. Perkirakan tingkat ketidakseimbangan sebagai $d = ms/ml$.

III. Jika nilai d kurang dari d_{th} (d_{th} adalah batas ambang yang dapat ditoleransi untuk rasio ketidakseimbangan), maka jumlah contoh data sintetis yang perlu dihasilkan untuk kelas minoritas dapat ditentukan oleh $G = (ml - ms) \times \beta$. Di sini, $\beta \in [0, 1]$ adalah parameter yang digunakan untuk menentukan tingkat keseimbangan yang diinginkan setelah pembangkitan data sintetis. Nilai $\beta = 1$ menunjukkan bahwa set data yang sepenuhnya seimbang akan dibuat setelah proses generalisasi.

IV. Untuk setiap contoh xi yang termasuk dalam kelas minoritas, cari k dengan tetangga terdekat berdasarkan jarak *Euclidean* dalam ruang berdimensi n. Hitung rasio ri yang didefinisikan sebagai $ri = \Delta i / k$, di mana $i = 1..ms$. Di sini, Δi adalah jumlah sampel dalam k tetangga terdekat dari x_i yang termasuk dalam kelas mayoritas, dan nilai ri terletak dalam rentang [0, 1].

V. Normalisasikan r_i menurut :

$$\widehat{r_i} = r_i / \sum_{i=1}^{m_s} r_{i,} \tag{2.8}$$

sehingga $\widehat{r_i}$ distribusi kepadatan (*density distribution*)

$$\left(\sum_i \widehat{r_i} = 1 \right) \tag{2.9}$$

VI. Hitung jumlah contoh data sintetis yang perlu dihasilkan untuk setiap contoh minoritas X_i dengan menggunakan $g_i = \widehat{r} \, x \, G$ di mana G adalah total jumlah contoh data sintetis yang perlu

dihasilkan untuk kelas minoritas.

VII. Hitung jumlah contoh data sintetis yang perlu dibuat untuk setiap contoh minoritas x_i menghasilkan g_i.

d. *Borderline*-SMOTE

Borderline-SMOTE adalah variasi dari teknik SMOTE yang difokuskan pada pembuatan instansi sintetis hanya di sekitar batas keputusan (*borderline*) antara kelas minoritas dan mayoritas. Berbeda dengan SMOTE yang secara acak memilih instansi minoritas, *Borderline*-SMOTE lebih selektif dalam menghasilkan instansi sintetik. Secara singkat, *Borderline*-SMOTE bertujuan untuk meningkatkan kualitas pembuatan instansi sintetis dengan memfokuskan pada titik-titik batas antara kelas. Hal ini dapat membantu mengatasi ketidakseimbangan kelas dengan lebih efektif karena hanya memperkenalkan instansi sintetis di area yang penting untuk memperbaiki masalah klasifikasi pada kelas minoritas [26]. Berikut merupakan algoritma dari Borderline-SMOTE:

I. Set pelatihan adalah T, kelas minoritas adalah P, dan mayoritas adalah M.

II. Untuk setiap pi $(i = 1, 2, 3,..., pnum)$ dalam kelas minoritas P: mencari tetangga terdekat m dari seluruh set pelatihan T (minoritas dan mayoritas).

III. Untuk setiap Pi, s tetangga terdekat dari k tetangga terdekatnya dalam P dipilih secara acak.

IV. Sistem menghitung perbedaan difj antara P'i dan s tetangga terdekat dari P, kemudian mengalikan perbedaan ini dengan angka acak rj antara 0 dan 1, dan hasil perkalian ditambahkan ke P'i. Data sintetis baru ditandai sebagai:

$$Synthetic_j = p'_i + r_j \, x \, dif_{j,} \; j = 1, 2,..., s \tag{2.10}$$

2.2.9 *Text to Sequences*

Text to Sequences adalah metode dalam pemrosesan teks yang umum

digunakan, terutama dengan menggunakan *library* seperti TensorFlow atau Keras. Fungsi ini berguna untuk mengonversi teks menjadi urutan angka (*sequences*) berdasarkan token yang telah dihasilkan sebelumnya oleh objek *Tokenizer*. Saat melakukan tokenisasi, setiap kata dalam teks diberikan token numerik sesuai dengan urutan kemunculannya dalam dataset. Proses ini memungkinkan representasi teks yang semula bersifat nominal berubah menjadi bentuk numerik, yang dapat digunakan sebagai input untuk model machine learning, terutama pada tugas-tugas yang melibatkan pemahaman dan analisis teks [27].

2.2.10 *Padding*

Padding dalam konteks CNN adalah proses menambahkan nilai nol atau nilai tetap ke tepi data *input* sehingga ukuran data tetap konsisten. Pada CNN, ini dilakukan untuk memastikan bahwa *feature maps* yang dihasilkan selama proses konvolusi memiliki dimensi yang sesuai. *Padding* berguna untuk mempertahankan informasi di tepi data, mencegah informasi yang penting hilang selama konvolusi, dan memastikan bahwa ukuran *output feature maps* tetap sesuai dengan ukuran *input* [28].

2.2.11 *Imbalance Data*

Berdasarkan penelitian yang dilakukan oleh Japkowicz dkk [32], tantangan ketidakseimbangan kelas sangat bergantung pada empat faktor utama: tingkat ketidakseimbangan kelas, ukuran keseluruhan kumpulan data pelatihan, dan jenis pengklasifikasi yang digunakan. Japkowicz dan rekan-rekannya [32] melakukan eksperimen menggunakan teknik pengambilan sampel, membandingkan hasil berbagai metode klasifikasi untuk mengevaluasi dampak ketidakseimbangan kelas. Eksperimen ini menyoroti betapa krusialnya tingkat ketidakseimbangan kelas, menunjukkan bahwa besarnya ketidakseimbangan kelas yang substansial mungkin tidak secara signifikan mempengaruhi klasifikasi suatu domain jika konsep yang mendasarinya mudah dipelajari. Oleh karena itu, untuk mengatasi tantangan ini, perhatian utama harus difokuskan pada pemahaman terhadap tingkat ketidakseimbangan kelas. Tingkat ketidakseimbangan kelas dapat

diukur dengan menetapkan rasio antara jumlah sampel kelas positif dan kelas negatif. Pendekatan lain untuk mengukur tingkat ketidakseimbangan adalah dengan menghitung *Imbalanced Ratio* (IR) [33], yang rumusnya diberikan di bawah ini: [33].

$$IR = \frac{Jumlah\ total\ kelas\ negatif}{Jumlah\ total\ kelas\ positif} \tag{2.11}$$