TUGAS AKHIR – IF184802

DETEKSI DEPRESI BERDASARKAN DATA TWITTER MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK*

FARIS SALBARI
NRP 05111440000119

Dosen Pembimbing I
Dini Adni Navastara, S. Kom., M.Sc.

Dosen Pembimbing II
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – IF184802

DETEKSI DEPRESI BERDASARKAN DATA TWITTER MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK*

FARIS SALBARI
NRP 05111440000119

Dosen Pembimbing I
Dini Adni Navastara, S.Kom., M.Sc.

Dosen Pembimbing II
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – IF184802

DEPRESSION DETECTION ON TWITTER DATA USING CONVOLUTIONAL NEURAL NETWORK

FARIS SALBARI
NRP 05111440000119

Advisor I
Dini Adni Navastara, S.Kom., M.Sc.

Advisor II
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

Departement of Informatics
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

DETEKSI DEPRESI BERDASARKAN DATA TWITTER MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh
FARIS SALBARI
NRP: 0511144000119

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Dini Adni Navastara, S.Kom., M.Sc. .....
NIP: 198510172015042001 (Pembimbing 1)

Dr. Eng. Nanik Suciati, S.Kom., M.Kom. .....
NIP: 197104281994122001 (Pembimbing 2)

SURABAYA
JANUARI, 2019

[Halaman ini sengaja dikosongkan]

DETEKSI DEPRESI BERDASARKAN DATA TWITTER MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK*

Nama Mahasiswa : FARIS SALBARI
NRP : 0511144000119
Departemen : Informatika FTIK-ITS
Dosen Pembimbing 1 : Dini Adni Navastara, S. Kom., M.Sc.
Dosen Pembimbing 2 : Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

ABSTRAK

Twitter adalah layanan jejaring sosial yang memiliki pengguna cukup banyak di Indonesia. Karena Twitter berbentuk *microblogging*, pengguna banyak memanfaatkannya untuk bercerita tentang masalah di kehidupan sehari-harinya. Dari *tweet* ini dapat dideteksi apakah pengguna ada indikasi depresi atau tidak.

Data yang digunakan pada tugas akhir ini diambil dengan metode *webcrawling* dan didapatkan 1000 *tweet* Twitter yang kemudian dilabeli secara manual menjadi 500 *tweet* depresi dan 500 *tweet* bukan depresi. Kemudian pada data tersebut dilakukan *text preprocessing*. Diterapkan juga *word embedding* dengan metode *Word2Vec Skip-gram* dan *word encoding* seperti *sequencing* dan *padding* agar data dapat diproses oleh algoritma CNN. Selanjutnya proses *training* data menggunakan model-model CNN yaitu model AlexNet, ZFNet, VGG16, VGG19, LeNet-5, serta model yang dirancang oleh penulis dan menggunakan *Stratified 5-Fold cross validation* sebagai metode validasi.

Hasil uji coba menunjukkan bahwa nilai rata-rata akurasi tertinggi sebesar 75,33% yang diperoleh dari arsitektur model yang dirancang oleh penulis. Arsitektur model tersebut terdiri dari 3 *layer* dengan jumlah *filter* sebanyak 16 pada tiap

layer dan menggunakan *optimizer* Nadam dengan *learning rate* 0,0001. Arsitektur model tersebut memiliki rata-rata akurasi yang lebih tinggi dari lima arsitektur model lainnya.

Kata Kunci: convolutional neural network, deteksi depresi, text preprocessing, twitter, word2vec, word embedding, word encoding

DEPRESSION DETECTION ON TWITTER DATA USING CONVOLUTIONAL NEURAL NETWORK

Student's Name	**: FARIS SALBARI**
Student's ID	**: 0511144000119**
Departement	**: Informatics FICT-ITS**
First Advisor	**: Dini Adni Navastara, S. Kom., M.Sc.**
Second Advisor	**: Dr. Eng. Nanik Suciati, S.Kom., M.Kom.**

ABSTRACT

Twitter is a social networking service that has many users from Indonesia. Because twitter is a kind of microblogging service, most of their users use twitter to tell their problems in their daily life.

The data that is used in this undergraduate thesis is retrieved using webcrawling method and obtained 1000 tweets from Twitter which is labelled and divided manually to 500 depressed tweets and 500 non-depressed tweets. Then text preprocessing is applied to the data. Word embedding using the method of Word2Vec Skip-gram and word encoding such as sequencing and padding are applied so the data can be processed by CNN algorithm. Then the data will be trained using CNN models such as AlexNet, ZFNet, VGG16, VGG19, LeNet-5, and a model designed by the writer of this undergraduate thesis. Stratified 5-Fold cross validation is applied as validation method.

The result of the training shows the best average accuracy score is 75,33% resulted from the model designed by the writer. The model architecture is consisting of 3 layers with 16 filters on every layer and using Nadam optimizer with learning rate of 0,0001. and using Stratified 5-Fold cross

validation as validation method. The model resulted accuracy average higher than the other five models.

Keywords: convolutional neural network, deteksi depresi, text preprocessing, twitter, word2vec, word embedding, word encoding

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"DETEKSI DEPRESI BERDASARKAN DATA TWITTER MENGGUNAKAN *CONVOLUTIONAL NEURAL NETWORK*".** Bagi penulis, pengerjaan Tugas Akhir ini merupakan sebuah pengalaman yang berharga. Selama pengerjaan Tugas Akhir, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menjalani perkuliahan di Informatika ITS dan Tugas Akhir ini adalah implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT yang maha pengasih lagi maha penyayang serta selalu memberikan rahmat-Nya dan Nabi Muhammad SAW yang telah membimbing umatnya menuju jalan yang benar.
2. Keluarga tercinta, Mama dan Papa yang telah memberikan dukungan moral dan material serta doa yang tidak pernah berhenti dipanjatkan dan selalu memberikan nasehat dan motivasi kepada penulis dalam mengerjakan Tugas Akhir ini.
3. Ibu Dini Adni Navastara, S.Kom., M.Sc., selaku pembimbing I dan dosen wali yang telah membimbing dan membantu penulis serta memberikan nasehat dalam menyelesaikan Tugas Akhir dan memberikan pembelajaran serta arahan saat masa kuliah.

4. Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom., selaku pembimbing II yang juga telah membantu, membimbing, dan memberikan nasehat kepada penulis dalam mengerjakan Tugas Akhir ini.
5. Brilian Widya, Paul Aldy, Buyung Abiyoso, Hanendyo Indira, dan Mochammad Baruno yang masih berjuang bersama dalam menyelesaikan tugas akhir. Serta Rizky Fenaldo, Abdul Majid Hasani, Glleen Allan, Nezar Mahardika, Andre Abdirrosyid, Petrus Damianus dan semua teman-teman Warkop x Jojoran yang selalu memberi dukungan moral dan saran dalam pengerjaan tugas akhir serta pengalaman pada masa kuliah.
6. Nabilah Zaki Lismia yang memberikan dukungan dan semangat serta pengalaman indah saat masa kuliah.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga, penulis mengharapkan kritik dan saran yang membangun dari pembaca untuk perbaikan ke depannya.

Surabaya, Januari 2019

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR PERSAMAAN

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Twitter adalah sebuah jejaring sosial yang hingga saat ini masih banyak digunakan oleh masyarakat Indonesia. Terbukti hingga Januari 2018, 27% dari pengguna jejaring sosial di Indonesia yang berjumlah 130 juta pengguna memiliki akun Twitter [1]. Twitter masih digunakan karena fiturnya yang memungkinkan pengguna untuk dapat bercerita dan berbagi pengalaman pada konten yang disebut "*tweet*" kepada pengguna lainnya.

Menganalisis isi dari *tweet* dewasa ini sudah menjadi cara yang cukup sering digunakan untuk memahami dan memprediksi perilaku sosial manusia. Karena frekuensi penggunaan yang cukup tinggi dan pengguna tersebar di masyarakat luas, Twitter menjadi sumber yang cukup kaya untuk memberikan data yang dapat digunakan untuk menganalisis berbagai macam perilaku tersebut.

Dewasa ini, banyak berita tentang kematian dengan cara bunuh diri. Menurut data statistik WHO, pada tahun 2014 sebanyak 23% dari jumlah kematian di dunia diakibatkan oleh gangguan jiwa dan penyalahgunaan obat-obatan. Kemudian menurut data statistik yang dikeluarkan oleh situs *World Population Review*, tiga dari seratus ribu penduduk Indonesia melakukan bunuh diri pada tahun 2018 [2].

Tahun 1980, Yann LeCun menemukan sebuah metode untuk melakukan pengenalan obyek gambar,

metode ini kemudian disebut *Convolutional Neural Network* atau CNN [3]. Walaupun metode ini lebih terkenal untuk melakukan klasifikasi obyek pada gambar, dewasa ini metode CNN sudah mulai digunakan untuk klasifikasi *Natural Language Processing*, seperti *Sentiment Analysis* dan *Spam Detection*.

Dalam Tugas Akhir ini, dilakukan pengambilan data teks dari jejaring sosial Twitter menggunakan *crawler*, untuk kemudian diterapkan ke sebuah model CNN untuk mendeteksi depresi berdasarkan data Twitter. Diharapkan model yang dibangun dapat mendeteksi secara akurat.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana melakukan *preprocessing* yang tepat untuk mengolah data hasil *crawling* pada Twitter?
2. Bagaimana membangun model CNN untuk mendeteksi depresi dengan memanfaatkan data hasil *crawling* pada Twitter?
3. Bagaimana melakukan proses klasifikasi data teks hasil *crawling* pada Twitter menggunakan CNN?
4. Bagaimana mengevaluasi kinerja model?

1.3. Tujuan

Tujuan dari pembuatan Tugas Akhir ini yaitu untuk mendeteksi depresi menggunakan data Twitter menggunakan CNN.

1.4. Manfaat

Tugas Akhir ini diharapkan dapat membantu memberi acuan *tweet* seperti apa yang dapat diklasifikasikan depresi.

1.5. Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Percobaan akan dilakukan dengan Bahasa pemrograman Python.
2. Data yang digunakan untuk percobaan adalah data dengan bahasa Indonesia yang diambil dari Twitter melalui program *crawler*.
3. Hasil klasifikasi dibagi menjadi dua kelas, yaitu depresi dan tidak depresi.

1.6. Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir

   Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri dari latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Subbab

metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

2. Studi literatur

   Pada studi literatur dipelajari sejumlah referensi yang diperlukan dalam pembuatan tugas akhir. Metode CNN yang akan diterapkan untuk klasifikasi teks dipelajari dari penelitian oleh Ahmed Husseini Orabi, Prasadith Buddhita, Mahmoud Husseini Orabi, dan Diana Inkpen [4] serta Diveesh Singh dan Aileen Wang [5].

3. Analisis dan desain perangkat lunak

   Fitur yang terdapat pada aplikasi klasifikasi ini adalah:

   1) Menampilkan jumlah input data *tweet* Twitter yang dimasukkan pengguna.
   2) Menampilkan proses jalannya algoritma CNN.
   3) Menampilkan hasil klasifikasi depresi dari data *tweet* Twitter.

4. Implementasi perangkat lunak

   Aplikasi pada tugas akhir ini akan dibangun dengan bahasa pemrograman Python. *Input* aplikasi merupakan data teks dari *tweet* Twitter. Output aplikasi berupa hasil klasifikasi depresi dari *tweet* Twitter.

5. Pengujian dan evaluasi

Pada pengerjaan tugas akhir ini dilakukan pengujian dengan menghitung nilai akurasi dari proses klasifikasi yang dijalankan pada setiap model CNN yang dibangun.

6. Penyusunan buku Tugas Akhir

   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7. Sistematika Penulisan Laporan

Buku Tugas Akhir ini merupakan laporan secara lengkap mengenai Tugas Akhir yang telah dikerjakan baik dari sisi teori, rancangan, maupun implementasi sehingga memudahkan bagi pembaca dan juga pihak yang ingin mengembangkan lebih lanjut. Sistematika penulisan buku Tugas Akhir secara garis besar antara lain:

Bab I Pendahuluan

Bab ini berisi penjelasan latar belakang, rumusan masalah, batasan masalah dan tujuan pembuatan Tugas Akhir. Selain itu, metodologi pengerjaan dan sistematika penulisan laporan Tugas Akhir juga terdapat di dalamnya.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Perancangan Sistem

Bab ini berisi penjelasan tentang rancangan dari sistem yang akan dibangun. Rancangan digambarkan dalam bentuk diagram alir.

Bab IV Implementasi

Bab ini berisi penjelasan implementasi dari rancangan yang telah dibuat pada bab sebelumnya. Implementasi disajikan dalam bentuk kode sumber secara keseluruhan disertai dengan penjelasannya.

Bab V Uji Coba Dan Evaluasi

Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai hasil percobaan yang telah dilakukan.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Dalam bab ini akan dijelaskan mengenai teori-teori yang merupakan dasar dari pembangunan sistem. Selain itu terdapat penjelasan yang menunjang pengerjaan Tugas Akhir ini sehingga dapat memberikan gambaran secara umum sistem yang akan dibangun.

2.1. Twitter

Twitter merupakan sebuah layanan jejaring sosial yang memungkinkan penggunanya untuk mengirim dan membaca pesan berbasis teks yang disebut "*tweet*". *Tweet* pada awalnya memiliki batas sebanyak 140 karakter. Namun, pada 7 November 2017 batasan tersebut dinaikkan hingga 280 karakter untuk semua Bahasa kecuali Bahasa Jepang, Korea, dan Mandarin.

Twitter didirikan pada Maret 2006 oleh Jack Dorsey, Noah Glass, Biz Stone, serta Evan Williams dan situsnya diluncurkan pada Juli 2006. Pada tahun 2012, lebih dari 100 juta pengguna mengirim *tweet* sebanyak 340 juta tiap harinya dan Twitter menangani 1,6 miliar pencarian. Pada tahun 2013, Twitter menjadi salah satu dari sepuluh situs yang paling sering dikunjungi. Twitter disebut juga sebagai "*the SMS of Internet*" [6]. Tercatat hingga awal tahun 2018, Twitter memiliki 336 juta pengguna aktif [7].

Pengguna yang sudah terdaftar dapat mengirim dan membaca *tweet*. Sedangkan pengguna yang tidak terdaftar hanya dapat membaca *tweet* yang dikirim pengguna lain. Pengguna dapat mengakses Twitter melalui sistus Twitter,

pesan singkat (SMS), atau melalui aplikasi pada perangkat seluler.

Twitter memiliki fitur untuk saling mengikuti (*follow*) antar penggunanya yang memungkinkan *tweet* pengguna yang diikuti ditampilkan oleh sistem pada lini masa para pengikut pengguna tersebut. Fitur *retweet*, *reply*, *like*, serta *direct message* juga tersedia sebagai sarana untuk berhubungan antar pengguna.

Beberapa fitur utama lainnya yang dimiliki Twitter adalah *mention*, *hashtag*, dan *trending topic*. *Mention* adalah fitur yang memungkinkan pengguna untuk membalas ataupun mengirim *tweet* pengguna lain dengan cara menggunakan tanda "@" yang diikuti nama pengguna. Fitur *hashtag* adalah fitur yang memudahkan pengguna untuk mengelompokkan dan mencari *tweet* berdasarkan topik bahasan dari *tweet* tersebut dengan cara menggunakan tanda "#" yang diikuti kata atau frasa topik yang diinginkan. Kemudian kata atau frasa topik yang sedang hangat dibicarakan akan ditampilkan pada fitur *trending topic*. Fitur ini membantu Twitter dan penggunanya untuk memahami apa yang sedang terjadi di dunia [6].

2.2. Depresi

Depresi adalah gangguan jiwa yang umum dan dapat memengaruhi diri secara negatif, baik dalam hal perasaan, pikiran, dan tindakan. Depresi dapat menyebabkan perasaan sedih dan/atau kehilangan minat dari suatu aktivitas yang sebelumnya disukai. Depresi juga dapat menyebabkan bermacam masalah emosi dan fisik serta dapat mengurangi kemampuan beraktivitas dalam pekerjaan ataupun di rumah.

Gejala depresi dapat bermacam-macam , contohnya

- Merasakan sedih terus menerus
- Kehilangan minat dan kesenangan saat melakukan hal yang sebelumnya dinikmati
- Perubahan nafsu makan yang menyebabkan turun atau naiknya berat badan tidak teratur yang bukan disebabkan oleh diet
- Kurang waktu tidur atau tidur terlalu lama yang menyebabkan waktu tidur tidak teratur
- Mudah kehilangan energi dan lelah
- Sering melakukan kegiatan yang tidak bertujuan
- Sering merasa bersalah dan tidak berguna
- Sulit untuk berpikir, berkonsentrasi, dan memilih keputusan
- Memiliki pikiran untuk mati atau bunuh diri

Depresi dapat menimpa siapapun, bahkan seseorang yang menurutnya sudah tinggal di lingkungan yang ideal. Ada beberapa faktor yang dapat memengaruhi depresi, di antaranya adalah faktor kepribadian dan lingkungan. Orang dengan tingkat kepercayaan diri yang rendah dan mudah mengalami stress serta orang yang bersifat pesimis adalah orang-orang yang mudah mengalami depresi. Sering menerima tindak kekerasan, ditolak oleh lingkungan, dan mengalami kemiskinan juga dapat membuat orang menjadi rentan terhadap depresi [8].

Tidak sedikit orang yang mengalami gejala depresi mengutarakannya di jejaring sosial seperti Twitter. Contoh *tweet* pengguna yang memiliki indikasi depresi dapat dilihat pada Tabel 2.1.

Tabel 2.1 Contoh *tweet* depresi

No	*Tweet*
1	Ya.. gue sering merasa sangat down. Dan ingin mati.
2	Apa lebih baik saya mati? Kerja ndak dapat-dapat, stress berat, siklus haid berantakan sejak lulus kuliah. AKU JUGA BISA DEPRESI
3	Saat ini lah titik terendah semasa hidup saya, jiwa dan otak saya tiap hari mendidih..mempertanyakan sebenernya siapa saya? Apa yg bisa saya lakukan? Saya depresi karena diri sendiri... pekerjaan saya amburadul, bahkan saya tidak bisa berpikir atau melakukan hal kecil...
4	berkutik dengan depresi akhir akhir ini sampai benar benar ingin mati. ah aku butuh teman.
5	Pernah suatu waktu ketika dihadapkan pada masalah yang menentuka masa depan membuat ku sangat stress dan depresi. Aku pernah memiliki keinginan untuk mengakhiri hidup ini karena merasa tidak ada seorang pun yang dapat mengerti aku.

2.3. Text Preprocessing

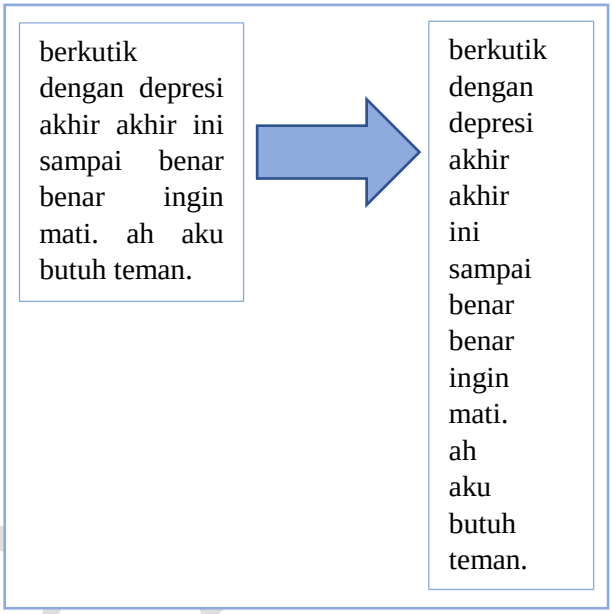
Sebelum melakukan pengolahan data teks, dibutuhkan pemrosesan data terlebih dahulu untuk mengubah data teks yang tidak terstruktur jadi terstruktur. Proses ini disebut *Text Preprocessing*. Proses ini dilakukan karena penyimpanan data yang terstruktur dapat membantu pengolahan data, karena dapat lebih mudah menghasilkan algoritma yang efisien.

Ada beberapa metode yang umum digunakan dalam *text preprocessing*, antara lain:

a. Tokenizing

Tokenizing adalah tahap untuk merubah teks yang panjang menjadi beberapa bagian kecil (*token*). Teks yang panjang dapat menjadi kalimat, kalimat

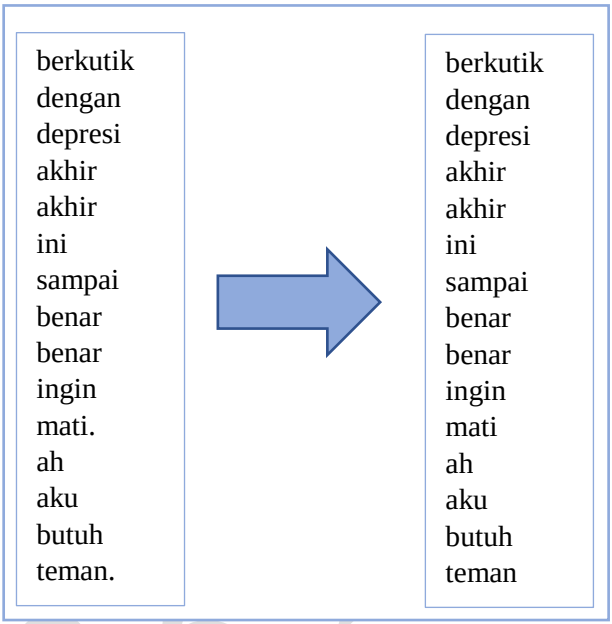
dapat menjadi kata, dan seterusnya. Contoh penerapan *tokenizing* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Contoh penerapan *tokenizing*

b. Cleansing

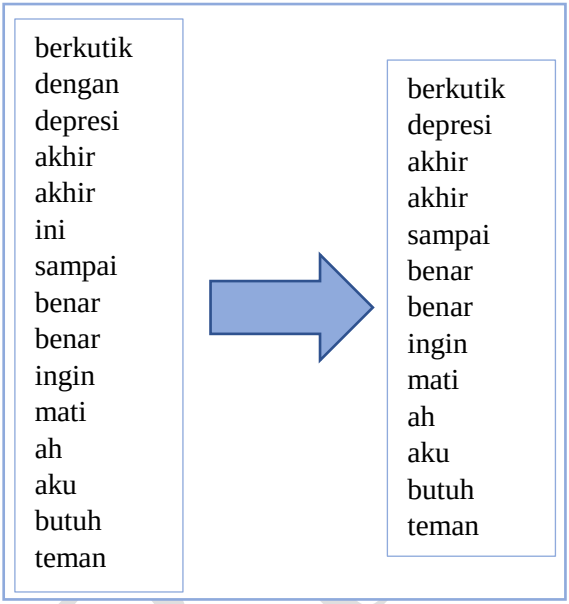
Cleansing adalah tahap untuk menghilangkan karakter-karakter yang tidak dibutuhkan untuk proses selanjutnya. C*leansing* akan menghilangkan angka, tanda baca, dan *emoticon* pada sebuah teks. Contoh proses *cleansing* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Contoh penerapan *cleansing*

c. *Stop Word Removal*

Stop word removal bertujuan untuk menghapus kata-kata yang tidak dianggap memiliki makna penting pada sebuah kalimat atau *stop word*. *Stop word* dapat berupa kata hubung maupun kata depan seperti kata “di”, “ke”, “dan”, “dengan”, serta “ini”. Contoh penerapan *stop word removal* dapat dilihat pada Gambar 2.3.



Gambar 2.3 Contoh penerapan *stop word removal*

d. Stemming

Stemming adalah tahap untuk menghapus imbuhan pada suatu kata dalam teks sehingga dapat menjadi kata dasar. Seperti kata "berkutik" menjadi "kutik". Contoh penerapan *stemming* dapat dilihat pada Gambar 2.4.

Gambar 2.4 Contoh penerapan *stemming*

2.4. *Word2Vec*

Word2Vec adalah salah satu teknik *word embedding* yang dibangun oleh Tomas Mikolov, Kai Chen, Greg Corrado, dan Jeffrey Dean pada tahun 2013. *Word2Vec* bertujuan untuk melakukan ekstraksi fitur menjadi bentuk vektor [9]. Vektor ini dapat mewakili makna dari sebuah kata yang kemudian dapat mengenali kemiripan antar kata. Terdapat dua metode *Word2Vec* yang dibangun oleh Tomas Mikolov, di antaranya:

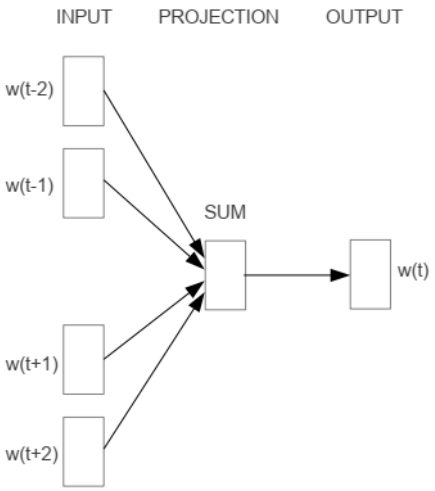
1. *Continuous Bag-of-Words*
   Metode ini mengambil vektor dari setiap kata pada suatu kalimat, kemudian dirata-rata untuk diproyeksikan pada satu tempat sebagai *input*. Metode ini berfungsi untuk memprediksi kata yang sesuai dalam konteks

suatu kalimat dengan kompleksitas *training* sebagai berikut [9].

$$Q = N \times D + D \times \log_2(V)$$

(2.1)

Di mana N adalah jumlah kata yang sedang di-*training*, D adalah dimensi atau jumlah vektor representasi dari tiap kata, dan V adalah jumlah kata dalam *vocabulary*. Arsitektur model *Continuous Bag-of-Words* dapat dilihat pada Gambar 2.5.



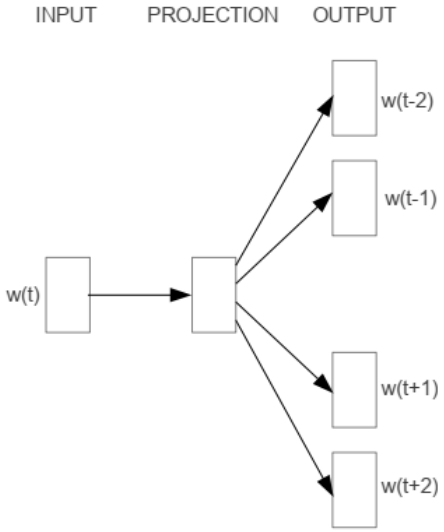
Gambar 2.5 Arsitektur model CBOW

2. *Skip-gram*
   Metode ini mengambil vektor dari kata di sekitar kata yang sedang di-*training* untuk

memahami konteks dari kata pada suatu kalimat. Metode ini memiliki kompleksitas *training* sebagai berikut [9].

$$Q = C \times (D + D \times \log_2(V)) \tag{2.2}$$

Di mana C adalah jarak maksimal antar kata yang akan diprediksi, D adalah dimensi atau jumlah vektor representasi dari tiap kata, dan V adalah jumlah kata dalam *vocabulary*. Arsitektur model *Skip-gram* dapat dilihat pada Gambar 2.6.



Gambar 2.6 Arsitektur model *Skip-gram*

Pada Gambar 2.5 dan Gambar 2.6 yang menunjukkan arsitektur CBOW dan *Skip-gram* dapat dilihat bahwa kedua arsitektur masing-masing memiliki tiga *layer* yaitu *input* yang berisi kata, *projection* yang menghasilkan matriks *word embedding* berukuran jumlah kata dalam *vocabulary* dikali jumlah dimensi vektor representasi tiap kata, dan *output* yang berisi hasil kata yang diprediksi.

Pada model CBOW, *layer input* berisikan kata-kata dari corpus, kemudian akan diubah menjadi vektor representasi dari kata-kata tersebut pada *layer projection*. Pada *layer output*, akan dihasilkan prediksi kata yang memiliki konteks yang sama dengan kata-kata pada *layer input* [9].

Sedangkan pada model *skip-gram*, *layer input* berisikan kata dari korpus yang akan diubah menjadi vektor representasi kata tersebut pada *layer projection*. Kemudian pada *layer output* akan dihasilkan kata-kata tetangga yang memiliki konteks sama dengan kata pada *layer input* [9].

Convolutional Neural Network atau CNN tidak bisa menerima data berbetuk teks menjadi data masukan atau *input*. Pada tahap inilah *word embedding* seperti *Word2Vec* berperan untuk mengubah data teks menjadi vektor sehingga dapat dijadikan data masukan pada CNN.

2.5. *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah salah satu jenis *neural network* yang biasa digunakan pada data gambar, seperti untuk mendeteksi dan mengenali obyek pada sebuah gambar. Namun CNN juga dapat digunakan pada data teks, dan sudah teruji memiliki akurasi yang cukup baik. Secara garis besar, CNN tidak jauh berbeda

dengan *neural network* biasanya. CNN terdiri dari *neuron* yang memiliki *weight*, *bias*, dan *activation function* yang bersifat *non-linear*.

Pada sebuah CNN umumnya terdapat *convolutional layer*, *pooling layer*, dan *fully connected layer* [10].

1. *Convolution Layer*
   Lapisan ini adalah inti dari CNN yang melakukan sebagian besar dari komputasi. Parameter dari lapisan ini berisi *filter*. *Filter* tersebut akan dilakukan komputasi *dot product* dengan *input*. Kemudian komputasi tersebut akan menghasilkan fungsi aktivasi 2 dimensi. Kemudian fungsi aktivasi tersbut akan didapatkan *output volume* [10].
2. *Pooling Layer*
   Lapisan ini dapat disebut juga *subsampling layer* dan diterapkan setelah *convolution layer*. Fungsi dari lapisan ini adalah untuk mengurangi parameter. Lapisan ini biasanya beroperasi menggunakan operasi *max pooling* [10].
3. *Fully Connected Layer*
   Fully Connected Layer berfungsi untuk mengklasifikasi data masukan dengan cara mentransformasi *output* dari *convolution* dan *pooling layer* menjadi data satu dimensi. Pada dasarnya *fully connected layer* merupakan sebuah *multi layer perceptron* (MLP) yang menggunakan sebuah fungsi aktivasi untuk proses klasifikasi [10].

   MLP merupakan kumpulan *neuron* yang saling terhubung dan umumnya terdiri dari tiga

jenis *layer* yaitu *input layer*, *hidden layer*, dan *output layer*. Sebuah *neuron* yang terhubung dengan *input* dan *output* membentuk sebuah sistem yang disebut *perceptron*. Variabel h_j merupakan sebuah *perceptron* yang menerima *input* sejumlah *d* yang dapat berasal dari data atau *output* dari *layer* sebelumnya. Variabel-variabel (w_{ij} , . . . , w_{dj}) merupakan bobot (*weights*) dari koneksi *input* dengan *neuron*. Terdapat kondisi khusus untuk x_0 dimana nilai dari x_0 selalu bernilai 1 dan w_{0j} dinamakan sebagai *bias*. Operasi pada sebuah *perceptron* terdiri dari dua buah operasi terpisah, yaitu operasi linear dan operasi non-linear (fungsi aktivasi) [10]. Berikut adalah persamaan untuk operasi linear dan persamaan untuk operasi aktivasi.

a. Persamaan operasi linear

$$z_j = \sum_{i=0}^{d} w_{ij} x_i \tag{2.3}$$

b. Persamaan operasi aktivasi

$$h_j = \sigma(z_j) \tag{2.4}$$

Ada beberapa pilihan operasi yang dapat diterapkan pada operasi non-linear, antara lain:

a. Linear

$$\sigma(a) = a \tag{2.5}$$

b. Sigmoid Logistic

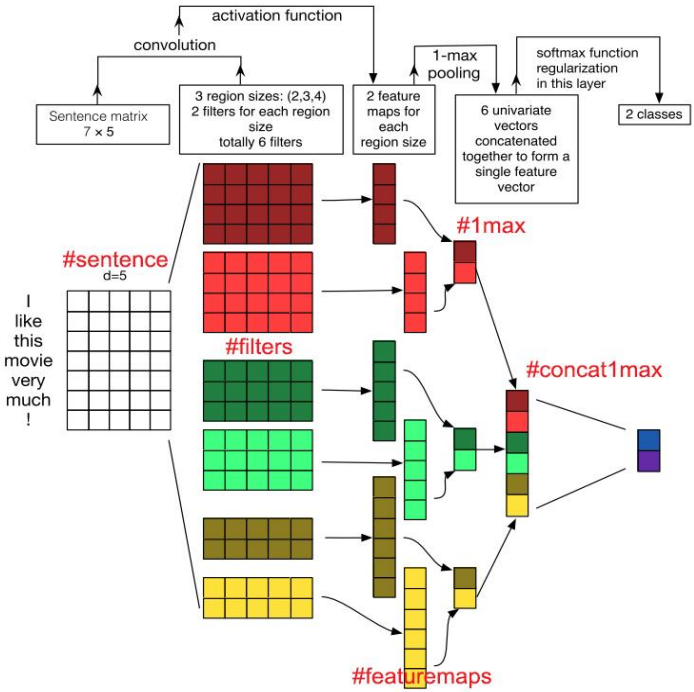
$$\sigma(a) = \frac{1}{1 + \exp(-a)} \tag{2.6}$$

c. Hyperbolic Tangent

$$\sigma(a) = \tanh(a) \tag{2.7}$$

d. Recitified Linear Unit (ReLU)

$$\sigma(a) = \max(0, a) \tag{2.8}$$



Gambar 2.7 Contoh arsitektur CNN pada NLP [11]

Pada NLP, *input* yang digunakan CNN adalah kalimat atau dokumen yang direpresentasikan sebagai matriks. Setiap baris matriks adalah vektor yang merepresentasikan *token* yang biasanya adalah sebuah kata. Vektor-vektor tersebut adalah hasil *word embeddings* seperti *word2vec*. Nantinya, *filter* akan dilakukan penggeseran ke seluruh baris matriks yang berisikan kata. Sehingga *width* dari *filter* biasanya sama dengan *width* dari matriks *input* namun *height* mungkin bervariasi. Gambar 2.7 adalah contoh arsitektur CNN yang digunakan untuk NLP [11].

2.6. *Deep Learning Optimization*

Algoritma optimisasi membantu *deep learning* untuk meminimalisir nilai *loss* pada sebuah model. Algoritma ini melakukan *training* dan memperbarui parameter yang ada pada *deep learning* untuk mencari solusi optimal dengan menggunakan variabel *learning rate*. *Learning rate* adalah variabel dengan nilai sangat kecil yang akan memastikan perubahan pada bobot model akan sangat kecil sehingga dapat meminimalisir *loss* [12].

Pada model CNN yang dibangun pada Tugas Akhir ini akan digunakan beragam jenis *optimizer* yang tersedia pada *library* Keras. Jenis *optimizer* yang digunakan di antaranya adalah:

1. Adam (*Adaptive Moment Estimation*)
   Adam adalah metode optimisasi yang dibangun Diederik Kingma dan Lei Ba pada 2015. Metode ini berbasis gradien dan momentum yang hanya membutuhkan memori kecil [12]. Metode ini melakukan kalkulasi *learning rate* pada tiap

parameter. Berikut adalah formula dari metode Adam.

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{2.9}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{2.10}$$

m_t adalah perkiraan dari momentum pertama (*mean*) dari gradien dan v_t adalah perkiraan dari momentum kedua (*variance*). Kemudian metode ini memperbarui parameter-parameter dengan formula berikut.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}} + \epsilon} \widehat{m_t} \tag{2.11}$$

Di mana θ adalah parameter yang akan diperbarui, η adalah *learning rate*, β_1 bernilai 0,9 dan β_2 bernilai 0,999 serta ϵ yang memiliki nilai 10^{-8}.

Metode ini bekerja cukup cepat dan dapat mengatasi masalah pembaruan parameter yang terlalu sering yang dapat mengakibatkan *loss* fluktuatif.

2. *Stochastic Gradient Descent* (SGD)

Metode optimisasi ini melakukan pembaruan parameter pada tiap *epoch*. Berikut adalah formula dari metode SGD.

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{(i)}; y^{(i)}) \tag{2.12}$$

Di mana θ adalah parameter yang akan diperbarui, η adalah *learning rate* yang akan dikalikan dengan parameter gradien dengan $x^{(i)}$ dan $y^{(i)}$ sebagai data *training* dan labelnya.

Karena pembaruan parameter yang cukup sering, hal ini mengakibatkan *loss* menjadi fluktuatif [12].

3. Adagrad

Adagrad dibangun oleh John Duchi, Elad Hazan, dan Yoram Singer pada tahun 2011. Metode ini memungkinkan *learning rate* untuk menyesuaikan diri terhadap parameter. Pembaruan yang bernilai lebih besar akan diterapkan pada parameter yang jarang, dan pembaruan bernilai lebih kecil akan diterapkan pada parameter yang sering. *Learning rate* akan berbeda-beda untuk setiap parameter pada tiap *epoch*. Berikut adalah formula dari Adagrad.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \tag{2.13}$$

Di mana θ adalah parameter yang akan diperbarui, η adalah *learning rate* dan g_t adalah gradien *loss* dari θ.

Kelebihan dari metode ini adalah tidak perlu mengubah *learning rate* secara manual. Sebagian besar implementasi menggunakan *learning rate* sebesar 0,01. Sedangkan kekurangan dari metode ini adalah *learning rate* yang selalu berkurang sehingga membuat model *deep learning* untuk tidak mempelajari data lebih banyak karena *learning rate* yang terlalu kecil [12].

4. Nadam

Metode ini dibangun oleh Timothy Dozat berdasarkan penggabungan algoritma *Nesterov Accelerated Gradient* yang dibangun oleh Yurii Nesterov pada tahun 1983 dan metode optimisasi Adam. Metode ini bertujuan untuk menggunakan algoritma yang lebih kuat dengan cara membagi tiap perubahan parameter dengan normalisasi *L2* dari perubahan sebelumnya pada parameter tersebut [12]. Berikut adalah formula yang digunakan.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}} + \epsilon} \left(\beta_1 \widehat{m_t} + \frac{(1 - \beta_1) g^t}{1 - \beta_1^t}\right) \tag{2.14}$$

$$\widehat{m_t} = (1 - \beta_1) g^t + \beta_1 m_t \tag{2.15}$$

Di mana θ adalah parameter yang akan diperbarui, η adalah *learning rate*, β_1 bernilai 0,9, dan ϵ yang memiliki nilai 10^{-8}. m_t adalah perkiraan dari momentum pertama (*mean*) dari gradien dan v_t adalah perkiraan dari momentum kedua (*variance*) serta g_t adalah gradien *loss* dari θ.

5. RMSprop
   RMSprop adalah metode optimisasi yang dibangun oleh Geoffrey Hinton. Metode ini akan membagi *learning rate* dengan gradien *loss* yang dikuadratkan [12]. Berikut adalah formula yang digunakan pada metode ini.

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \tag{2.16}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{2.17}$$

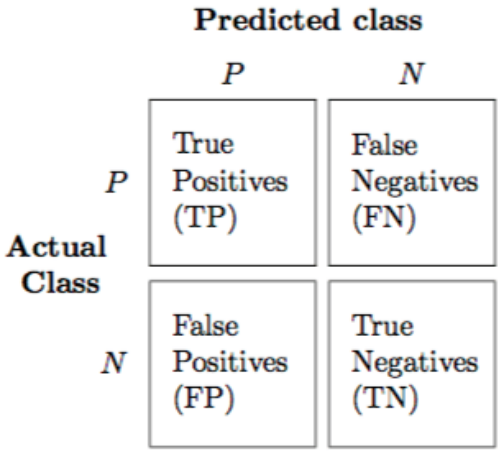
   Di mana θ adalah parameter yang akan diperbarui, η adalah *learning rate*, g_t adalah gradien *loss* dari θ, dan ϵ yang memiliki nilai 10^{-8}.

2.7. Metode Pengukuran Evaluasi

Setiap *machine learning* membutuhkan evaluasi untuk menilai seberapa baik metode yang digunakan terhadap data yang dilakukan *learning*. Beberapa metode pengukuran evaluasi yang paling sering digunakan adalah dengan menghitung akurasi ataupun *loss*.

Pada Tugas akhir ini pengukuran evaluasi yang digunakan adalah akurasi dan *loss* dengan metode *cross entropy*. Berikut adalah penjelasan dari masing-masing metode pengukuran evaluasi.

1. Akurasi
   Untuk mencari akurasi dari sebuah model *machine learning* dilakukan operasi matematika yang melibatkan jumlah data yang mendapatkan prediksi yang benar dan jumlah total data yang diprediksi.



Gambar 2.8 *Confusion matrix*

$$ACC = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.18}$$

Persamaan (**2.18** menunjukkan operasi matematika untuk mendapatkan nilai akurasi dari hasil klasifikasi biner sebuah model *machine*

learning terhadap data. Di mana TP adalah *True Positive* yaitu jumlah data yang diprediksi dengan benar memiliki kelas *positive*, TN adalah *True Negative* yaitu jumlah data yang diprediksi dengan benar memiliki kelas *negative*, FP adalah *False Positive* yaitu jumlah data yang diprediksi dengan salah memiliki kelas *positive*, dan FN adalah *False Negative* yaitu jumlah data yang diprediksi dengan salah memiliki kelas *negative*. Penggambaran dari penjelasan tersebut dapat dilihat pada Gambar 2.8.

2. *Cross Entropy*
   Loss pada *machine learning* adalah nilai yang mengindikasikan seberapa buruk hasil prediksi dari *machine learning* pada sebuah data. Semakin besar nilai *loss*, maka semakin buruk hasil prediksi terhadap data tersebut. *Cross Entropy* adalah salah satu fungsi untuk mendapatkan nilai *loss* dari klasifikasi biner. Fungsi ini menggunakan probabilitas kelas dari suatu data dalam operasinya.

$$E_N = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot \log\left(p(y_i)\right) + (1-y_i) \cdot \log\left(1-p(y_i)\right)\right] \tag{2.19}$$

   Persamaan di atas adalah operasi penghitungan *loss* dengan fungsi *Cross Entropy*. Di mana N adalah jumlah data, y_i adalah kelas aktual yang dimiliki data tersebut, $p(y_i)$ probabilitas data

tersebut memiliki kelas dengan label 1, dan $1 - p(y_i)$ adalah probabilitas data tersebut memiliki kelas dengan label 0.

BAB III
PERANCANGAN SISTEM

Bab ini menjelaskan tentang perancangan dan pembuatan sistem perangkat lunak. Sistem perangkat lunak yang dibuat pada Tugas Akhir ini adalah klasifikasi data hasil pencarian pada Twitter berupa data teks dengan menggunakan algoritma *Convolutional Neural Network*. Pada bab ini pula akan dijelaskan gambaran umum sistem dalam bentuk diagram alir.

3.1. Data

Pada sub bab ini akan dijelaskan mengenai data yang digunakan sebagai masukan perangkat lunak untuk selanjutnya dilakukan klasifikasi sehingga menghasilkan data keluaran yang diharapkan.

3.1.1. Data Masukan

Data masukan atau *input* yang digunakan dalam studi kerja klasifikasi ini adalah teks dari *tweet* Twitter yang diambil menggunakan program *crawler*. Terdapat 1000 data yang digunakan pada pengerjaan tugas akhir ini. Data tersebut kemudian dibagi dua kategori yaitu depresi dan normal. Penentuan kategori pada setiap data dilakukan secara manual dan mengacu pada arahan dari diskusi dengan psikolog dan beberapa teman mahasiswa jurusan psikologi. Parameter pencarian data teks dari *tweet* Twitter adalah sebagai berikut.

- *Tweet* dikirim oleh pengguna yang berlokasi di Indonesia

- *Tweet* yang telah dikirim sebelum tanggal 25 November 2018
- *Tweet* mengandung kata ataupun frasa yang telah ditentukan berdasarkan hasil dari diskusi oleh psikolog dan beberapa teman mahasiswa jurusan psikologi

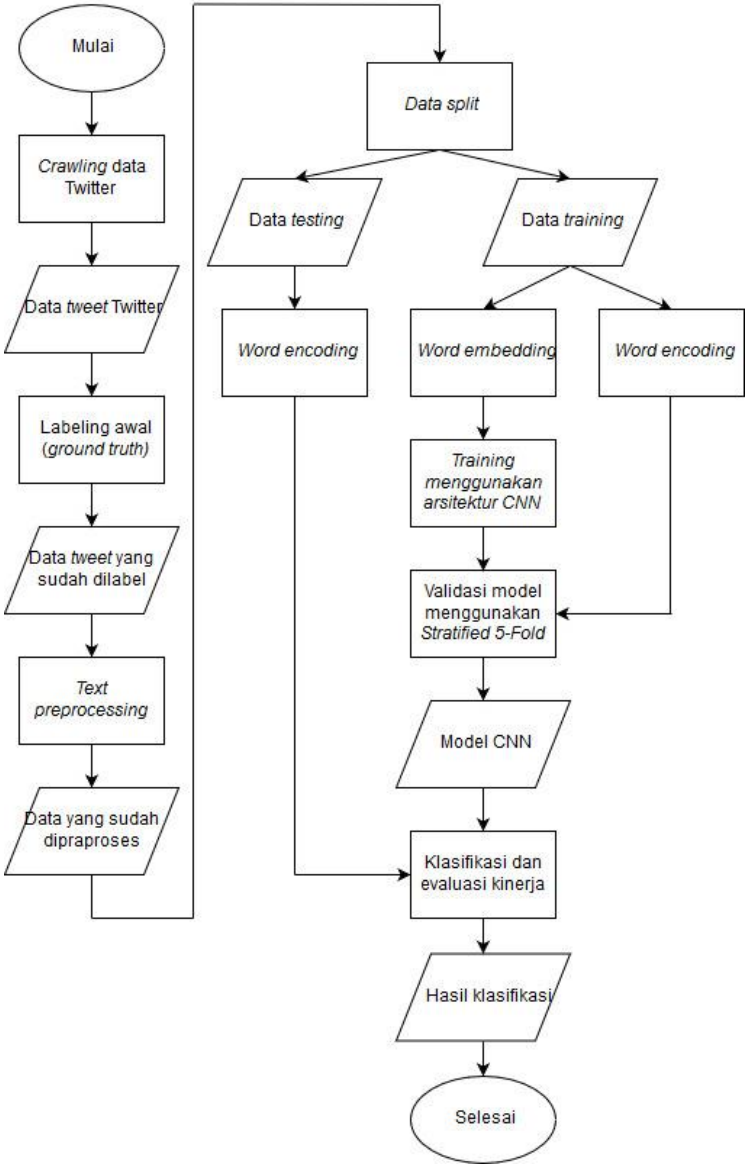
Setelah data teks dari Twitter telah didapatkan, penulis melakukan pelabelan secara manual dengan membaca dan memahami maksud dari *tweet* tersebut.

3.1.2. Data Keluaran

Data masukan diproses dan menghasilkan data keluaran berupa prediksi hasil klasifikasi depresi atau bukan depresi (normal) serta nilai akurasi dari proses klasifikasi yang telah dilakukan. Program yang dibuat juga menghasilkan *file* model *Convolutional Neural Network* dari proses *training* yang telah dilakukan. Model tersebut menyimpan nilai *weight* dari proses klasifikasi sebelumnya sehingga dapat digunakan untuk melakukan klasifikasi langsung tanpa melakukan training terlebih dahulu.

3.2. Desain Umum Sistem

Pada subbab ini akan dibahas mengenai desain perancangan sistem yang dilakukan untuk menjelaskan secara rinci setiap alur implementasi pada proses klasifikasi depresi pada *tweet* Twitter. Gambar 3.1 menunjukkan diagram alir Tugas Akhir ini secara umum.



Gambar 3.1 Diagram alir program

3.2.1. *Crawling* Data *Tweet* Twitter

Crawling atau proses pengambilan data *tweet* pada Twitter dilakukan dengan metode *webcrawling*. Metode *webcrawling* merupakan cara untuk mengambil informasi pada sebuah situs web melalui struktur HTML halaman situs tersebut. Program *webcrawler* diambil dari program berbahasa Python yang sebelumnya sudah dirancang dan dipublikasikan oleh Jefferson Henrique pada halaman GitHub miliknya [13].

Parameter yang digunakan pada proses *crawling* ini yaitu *tweet* yang dikirim berlokasi di Indonesia sebelum tanggal 25 November 2018 serta mengandung kata ataupun frasa yang telah ditentukan penulis dari hasil diskusi oleh psikolog dan mahasiswa jurusan psikologi. Kata dan frasa kunci yang digunakan adalah:

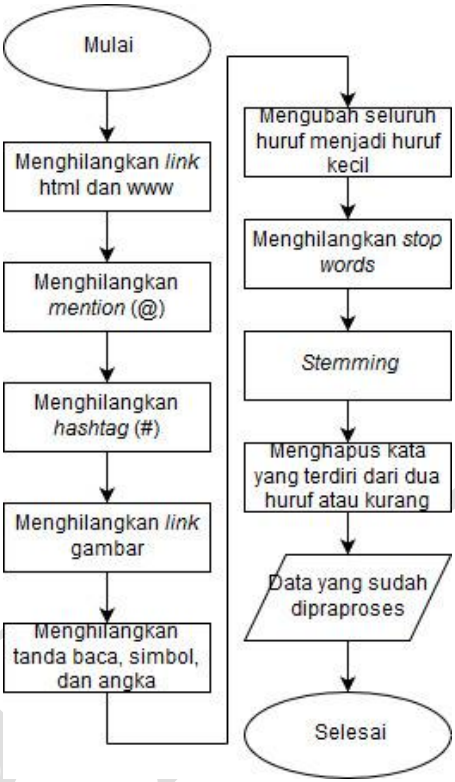
1. Bosan hidup
2. Hidup tidak berarti
3. Ingin mati
4. Kesepian
5. Lelah hidup
6. Mengakhiri hidup
7. Patah semangat
8. Putus asa
9. Saya depresi
10. Selamat tinggal dunia

3.2.2. Labeling Awal

Data *tweet* Twitter yang sudah didapatkan kemudian dilakukan pelabelan secara manual dengan cara membaca dan memahami maksud dari *tweet* tersebut apakah benar memiliki indikasi depresi atau tidak. Contoh data yang sudah dilabeli dapat dilihat pada Tabel 3.1.

Tabel 3.1 Contoh data terlabeli

No	Data *tweet*	Kelas
1	berkutik dengan depresi akhir akhir ini sampai benar benar ingin mati. ah aku butuh teman.	Depresi
2	Ku ingin mati saja tidur selamanya daripada ku tak dapat bercerita kepada siapapun, ternyata ini menyiksa...	Depresi
3	Saat ini lah titik terendah semasa hidup saya, jiwa dan otak saya tiap hari mendidih..mempertanyakan sebenernya siapa saya? Apa yg bisa saya lakukan? Saya depresi karena diri sendiri... pekerjaan saya amburadul, bahkan saya tidak bisa berpikir atau melakukan hal kecil...	Depresi
4	Jika kau merasa bosan dengan hidup. Mungkin kau sedang melalui titik di mana kau tidak mengharapkan apapun dalam hidupmu.	Normal
5	Hidup tidak berarti apa-apa tanpa persahabatan" - Quintus Ennius"	Normal
6	Semua orang pasti pernah merasakan jatuh dan ingin mati. Astagfirullah.	Normal



Gambar 3.2 Diagram alir *text preprocessing*

3.2.3. *Text Preprocessing*

Setelah data mendapatkan label, data teks dari isi *tweet* akan dilakukan *preprocessing* dengan tahapan *cleansing* untuk membersihkan teks dari kata atau karakter yang tidak dibutuhkan untuk tahap selanjutnya. Kemudian akan dilakukan *stop word removal*, *stemming*, dan *tokenization*. Gambar 3.2 menunjukkan diagram alir *text preprocessing* yang dilakukan pada tugas akhir ini.

Proses *cleansing* yang dilakukan dalam pengerjaan ini terdiri dari beberapa tahap yaitu:

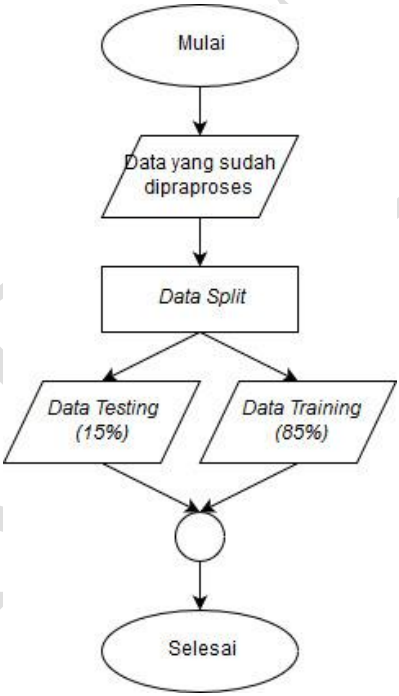
1. Menghilangkan *link* html dan www.
2. Menghilangkan *mention* (@).
3. Menghilangkan *hashtag* (#).
4. Menghilangkan gambar yang memiliki format *link* pic.twitter.com/…
5. Menghilangkan tanda baca, simbol, dan angka.
6. Mengubah seluruh huruf menjadi huruf kecil.
7. Menghilangkan *stop words.*
8. Mengubah seluruh kata menjadi kata dasar dari kata tersebut atau *stemming*.
9. Menghapus kata yang terdiri dari dua huruf atau kurang.

Pada tahap penghapusan *stop word*, penulis menggunakan *library* dari sastrawi yang dapat dilakukan dengan Bahasa Indonesia. Terdapat beberapa perubahan yang dilakukan untuk menyesuaikan dengan kasus yang diangkat. Perubahan tersebut adalah mengurangi daftar *stop words* yang akan dihapus. *Stop words* yang tidak akan dihapus adalah ‘dia’ dan ‘saya’ dengan pertimbangan kata ini dapat menentukan siapa yang mengalami depresi, ‘belum’ dengan pertimbangan kata ini adalah negasi dari konteks, serta ‘ingin’ karena kata tersebut terdapat pada frasa kunci yang digunakan sebagai pertimbangan indikasi depresi yaitu ‘ingin mati’.

3.2.4. Data *Split*

Data yang sudah dipraproses kemudian dibagi menjadi data *training* dan data *testing* dengan rasio pembagian 85% untuk data *training* dan 15% untuk data *testing*. Tahap data *split* ini menghasilkan dua data yang terpisah dengan jumlah 850

tweet untuk data *training* dan 150 *tweet* untuk data *testing*. Pada tahapan selanjutnya data *training* akan dilakukan proses *training word embedding* dan model CNN sedangkan pada data *testing* akan dilakukan evaluasi dari model yang dihasilkan pada proses *training* dengan dicari akurasinya. Gambar 3.3 menunjukkan tahapan data *split* yang dilakukan pada tugas akhir ini.

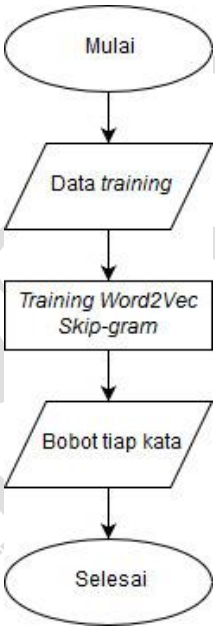


Gambar 3.3 Diagram alir data *split*

3.2.5. *Word Embedding*

Pada tugas akhir ini penulis menggunakan model untuk merepresentasikan kata menjadi vektor berdimensi $Xi \times W$ di

mana *Xi* adalah elemen dari teks dan *W* adalah matriks bobot *word embedding* yang akan dijadikan *input* pada *embedding layer* pada model CNN. Metode yang digunakan adalah metode *Word2Vec* yang dibangun oleh Tomas Mikolov pada tahun 2013 [9]. Model yang digunakan adalah *Skip-gram*. Gambar 3.4 menunjukkan diagram alir proses *word embedding*.



Gambar 3.4 Diagram alir *word embedding*

3.2.6. *Word Encoding*

Proses ini memiliki tujuan untuk mengubah data teks menjadi data yang dapat diproses oleh model CNN. Proses ini terdiri dari dua tahapan yaitu *sequencing* dan *padding*. Diagram

alir *word encoding* yang dilakukan pada tugas akhir ini dapat dilihat pada Gambar 3.5.

Sequencing bertujuan untuk merepresentasikan data teks menjadi angka yang menjadi indeks tiap kata pada *vocabulary*. Selanjutnya adalah proses *padding*. Proses ini akan merubah kalimat menjadi kumpulan array yang terdiri dari kumpulan kata-kata. Dalam proses ini jumlah dimensi array harus ditentukan terlebih dahulu supaya seluruh data teks yang diproses memiliki ukuran dimensi yang sama. Tahap ini penting dilakukan karena CNN membutuhkan data dengan dimensi yang tetap untuk diproses.

Gambar 3.5 Diagram alir *word encoding*

3.2.7. Proses Training Data Menggunakan CNN

Pengerjaan tugas akhir ini menggunakan enam arsitektur model CNN yaitu model AlexNet, ZFNet, VGG16, VGG19, LeNet-5, dan model yang dibangun oleh penulis. Pada tugas akhir ini penulis menambahkan *embedding layer* pada awal setiap arsitektur untuk menggunakan bobot kata dari hasil proses *word embedding* serta mengubah fungsi aktivasi pada *layer* terakhir menjadi *sigmoid* karena paling efektif dengan data yang memiliki dua kelas.

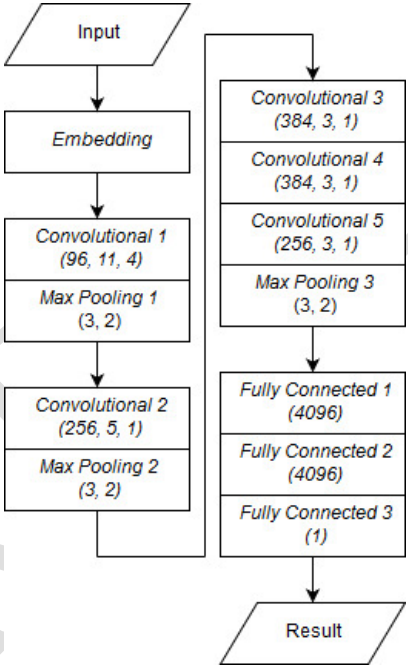
Pada data *training* kemudian dilakukan *word encoding* supaya data tersebut dapat diproses algoritma CNN pada proses validasi. Pada proses *validasi* ini digunakan *stratified K-fold cross validation*. Metode ini membagi data *training* menjadi K bagian dengan penyebaran data yang acak, dalam Tugas Akhir ini K sudah ditentukan oleh penulis yaitu sebesar 5. Dalam pembagian data tersebut, *stratification* bertujuan untuk memastikan bahwa tiap bagian atau *fold* memiliki representasi yang tepat terhadap data secara keseluruhan. Kemudian *training* hanya akan dilakukan pada empat *fold* dan dilakukan validasi terhadap satu *fold* sisanya. *Training* dilakukan sebanyak lima kali hingga masing-masing *fold* sudah dijadikan data validasi.

Berikut adalah penjelasan dari arsitektur-arsitektur model CNN yang digunakan:

a. AlexNet

   AlexNet dibangun oleh Alex Krizhevsky, Ilya Sutskever, serta Geoffrey E. Hinton untuk mengklasifikasi 1,2 juta gambar pada kontes ImageNet Large Scale Visual Recognition Challenge (ILSVRC) pada tahun 2010. Kemudian memenangkan kontes tersebut pada tahun 2012

dengan *error rate* sebesar 15,3%. Pada model ini terdapat 5 *convolutional layer* dan 3 *fully connected layer* dengan *ReLU* sebagai fungsi aktivasi pada empat *layer* pertama serta *softmax* pada *layer* terakhir [14]. Gambar 3.6 menunjukkan arsitektur model Alexnet.

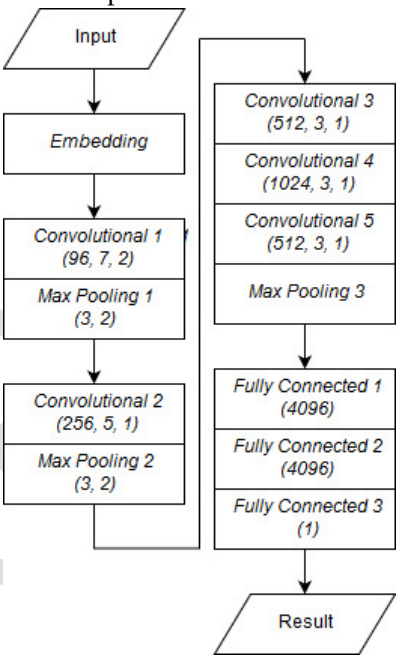


Gambar 3.6 Arsitektur model AlexNet

b. ZFNet

ZFNet dibangun oleh Matthew D. Zeiler dan Rob Fergus pada ILSVRC tahun 2013. Walaupun tidak menjadi pemenang kontes tersebut, model ini memiliki *error rate* yang lebih baik dari AlexNet

yaitu sebesar 14,7%. Model ini memiliki arsitektur yang mirip dengan AlexNet, perbedaannya yaitu ukuran *kernel* pada *convolutional layer* pertama diubah menjadi 7x7 dengan *stride* 2x2 serta pada *convolutional layer* 3, 4, dan 5 jumlah *filter* menjadi 512, 1024, dan 512 [15]. Arsitektur model ZFnet dapat dilihat pada Gambar 3.7.

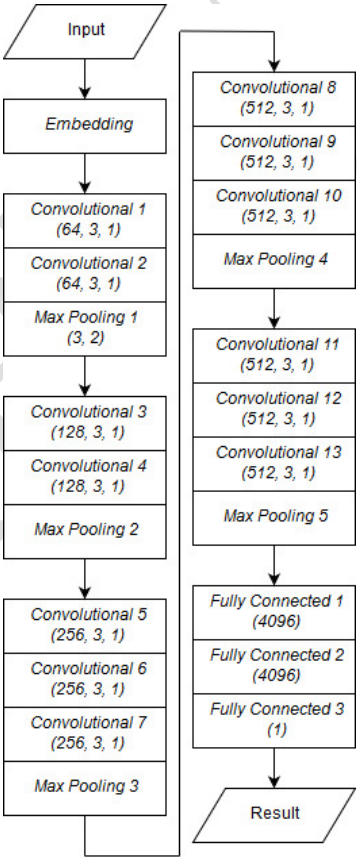


Gambar 3.7 Arsitektur model ZFNet

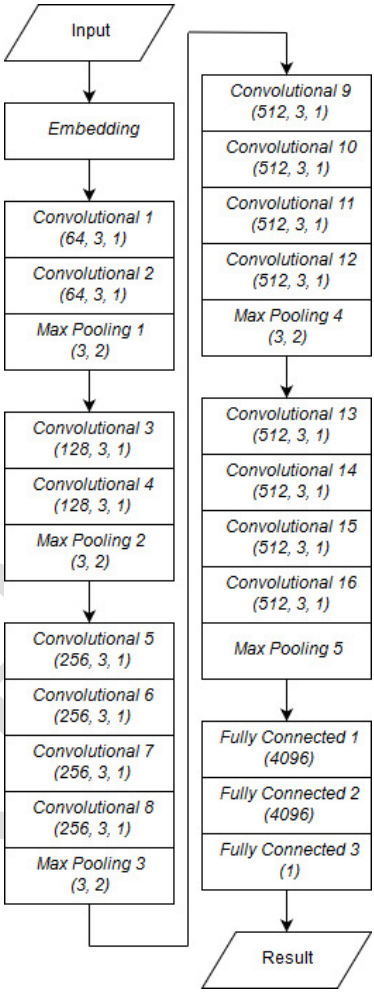
c. VGG16

VGG16 dibangun oleh Visual Geometry Group (VGG) yang terdiri dari Karen Simonyan dan Andrew Zisserman. Model ini menjadi juara kedua pada ILSVRC tahun 2014 dengan *error rate* yang jauh lebih baik daripada AlexNet yaitu sebesar

7,2%. VGG menggunakan *filter* yang lebih sedikit namun memiliki banyak *convolutional layer*, pengurangan *filter* ini berpengaruh pada jumlah parameter yang lebih sedikit sehingga mengurangi masalah *overfitting*. Model ini memiliki total 16 *layer* yang terdiri dari 13 *convolutional layer* dan 3 *fully connected layer* [16]. Gambar 3.8 menunjukkan arsitektur model VGG16.



Gambar 3.8 Arsitektur model VGG16



Gambar 3.9 Arsitektur model VGG19

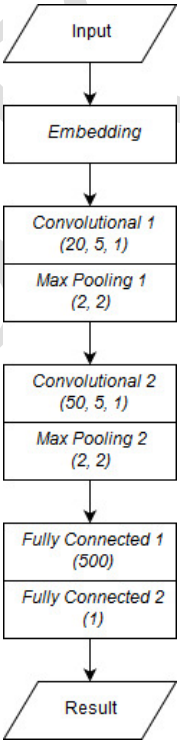
d. VGG19

VGG19 adalah salah satu variasi model yang dibangun oleh VGG. Model ini mendapatkan

error rate sebesar 7,1%. VGG19 memiliki total 19 *layer* dengan 16 *convolutional layer* dan 3 *fully connected layer* [16]. Arsitektur model VGG19 dapat dilihat pada Gambar 3.9.

e. LeNet-5

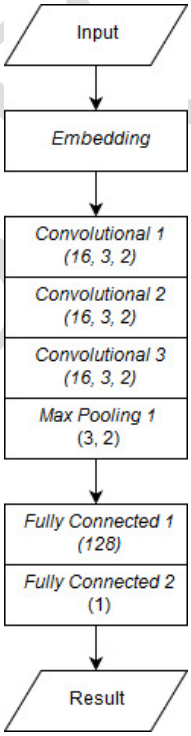
LeNet-5 diperkenalkan oleh Yann LeCun pada tahun 1998. Model ini menggunakan 2 *convolutional layer* dan 2 *fully connected layer* [17]. Gambar 3.10 menunjukkan arsitektur model LeNet-5.



Gambar 3.10 Arsitektur model LeNet-5

f. 3 *layers*

Arsitektur dari model ini dirancang oleh penulis dengan percobaan yang dilakukan selama masa pengerjaan tugas akhir ini. Model ini terdiri dari 3 *convolutional layer* dan 2 *fully connected layer*. Arsitektur model ini dapat dilihat pada Gambar 3.11.



Gambar 3.11 Arsitektur model 3 *layers*

3.2.8. Proses Klasifikasi dan Evaluasi

Model CNN yang telah didapat dari proses *training* kemudian digunakan untuk klasifikasi data *testing*. Kemudian evaluasi dari model yang digunakan untuk klasifikasi ini akan ditampilkan juga akurasi dan *loss* yang menggunakan metode *binary crossentropy*. Akurasi dari masing-masing model dapat dilihat pada bagian skenario uji coba sedangkan *loss* dari masing-masing model dapat dilihat pada lampiran a.

BAB IV
IMPLEMENTASI

Pada bab ini diuraikan mengenai implementasi perangkat lunak dari rancangan metode yang telah dibahas pada Bab III meliputi kode program dalam perangkat lunak. Selain itu, implementasi dari tiap proses, parameter masukan, keluaran, dan beberapa keterangan yang berhubungan dengan program juga dijelaskan.

4.1. Lingkungan Implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan tugas akhir memiliki spesifikasi perangkat keras dan perangkat lunak seperti ditampilkan pada Tabel 4.1.

Tabel 4.1 Spesifikasi lingkungan implementasi

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz 2.5 GHz
	Memori	4 GB
	GPU	NVIDIA GeForce GTX 1050
Perangkat Lunak	Sistem Operasi	Windows 10 64-bit
	Bahasa Pemrograman	Python 3.6
	Perangkat Pengembang	Anaconda3 5.3.1

4.2. Implementasi Proses

Pada subbab ini akan dijelaskan secara detail tentang implementasi proses yang dilakukan berdasarkan perancangan proses yang sudah dijelaskan pada bab perancangan sistem. Program dibangun dengan menggunakan bahasa pemrograman Python 3.6 dan *framework* Keras dengan *backend* Tensorflow sebagai pustaka utama.

Proses *training* dilakukan dengan metode *Convolutional Neural Network* (CNN). Terdapat enam arsitektur model CNN yang dibuat pada implementasi tugas akhir ini, yaitu model AlexNet, model ZFNet, model VGG16, model VGG19, model LeNet-5 dan model yang dirancang oleh penulis.

4.2.1. Implementasi *Training* Data Teks Menggunakan CNN

Sebelum memasuki tahapan proses *training*, dilakukan tahap *text preprocessing* dan *word embedding*. Berikut penjelasan dan implementasi dari masing-masing tahapan:

4.2.1.1. Implementasi Preprocessing Data Teks

Tahap pertama yang dilakukan adalah data yang telah terkumpul dimuat ke *dataframe* dan membuang data yang tidak dibutuhkan untuk proses *training* serta menambahkan kolom label pada data. Fungsi yang digunakan pada implementasi tahap ini adalah fungsi-fungsi yang disediakan oleh *library* pandas, yaitu read_csv untuk memuat *file* dengan tipe csv ke dalam *dataframe*, kemudian *drop* dan *assign* berfungsi untuk mengurangi dan menambah kolom pada *dataframe* serta *concat* untuk menggabung dua *dataframe* yang berbeda dan to_csv untuk menyimpan *dataframe* dalam bentuk *file* dengan tipe csv.

Kode Sumber 4.1 menunjukkan implementasi untuk memuat data.

```
import pandas as pd

file_0 = 'data/0.csv'
file_1 = 'data/1.csv'

def load_and_label(file_):
    df = pd.read_csv(file_, header=0, index_col=None,
encoding='latin1',sep=';')
    df.drop(["username","date","retweets","favorites","geo","mentions","h
ashtags","id","permalink"],axis=1,inplace=True)
    if (file_ is file_0):
        df = df.assign(depresi = 0)
    elif (file_ is file_1):
        df = df.assign(depresi = 1)
    return df

df_0 = load_and_label(file_0)
df_1 = load_and_label(file_1)
df = [df_0, df_1]
df_join = pd.concat(df, axis=0, ignore_index=True)
df_join.to_csv('data/all.csv',encoding='utf-8')
```

Kode Sumber 4.1 Memuat data

Setelah data berhasil dimuat ke dalam *dataframe*, tahap selanjutnya adalah *data cleansing*. Tahap ini akan menghilangkan *link* html dan www, *mention* (@), *hashtag* (#), *link* gambar, tanda baca, simbol, angka, *stop words*, dan mengubah seluruh kata menjadi kata dasar atau *stemming*, serta menghilangkan kata yang memiliki huruf kurang dari dua.

Fungsi-fungsi yang digunakan dalam implementasi tahap ini adalah StopWordRemoverFactory dan StemmerFactory yang disediakan oleh *library* sastrawi pada python yang berfungsi untuk membuang *stop word* serta melakukan *stemming* dalam bahasa Indonesia.

```
from bs4 import BeautifulSoup
from nltk.tokenize import WordPunctTokenizer
from Sastrawi.StopWordRemover.StopWordRemoverFactory import Sto
pWordRemoverFactory
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import re

swfactory = StopWordRemoverFactory()
stopword = swfactory.create_stop_word_remover()
stfactory = StemmerFactory()
stemmer = stfactory.create_stemmer()
tok = WordPunctTokenizer()

removehtml = r'http\S+'
removemention = r'@\S+'
removewww = r'www.\S+'
removepic = r'pic\S+'

def tweet_cleaner(text):
    soup = BeautifulSoup(text, 'lxml')
    souped = soup.get_text()
    soup1 = BeautifulSoup(souped, 'html.parser')
    souped1 = soup1.get_text()
    stripped = re.sub(removehtml,'',souped1)
    stripped = re.sub(removemention,'',stripped)
    stripped = re.sub(removewww, '', stripped)
    stripped = re.sub(removepic, '', stripped)
    letters_only = re.sub('[^a-zA-Z]',' ',stripped)
    stopw = stopword.remove(letters_only)
    stemmed = stemmer.stem(stopw)
    words = [x for x in tok.tokenize(stemmed) if len(x) > 2]
    return (" ".join(words)).strip()

print("Cleaning and parsing the tweets...\n")
clean_tweet_texts = []
for i in range(0,1000):
    if((i+1)%100 == 0):
        print("Tweets %d of %d has been processed" % (i+1, 1000))
    clean_tweet_texts.append(tweet_cleaner(df['text'][i]))

clean_df = pd.DataFrame(clean_tweet_texts, columns=['text'])
clean_df['target'] = df.depresi
clean_df.to_csv('data/clean_all.csv',encoding='utf-8')
```

Kode Sumber 4.2 Implementasi *text preprocessing*

Kemudian digunakan juga fungsi *sub* dari *library regular expression* (re) yang berfungsi untuk mengganti suatu karakter dalam suatu string dengan karakter lainnya yang sudah ditentukan. Contoh penggunaan fungsi pada tahap ini adalah mengganti seluruh karakter pada kata yang diawali dengan "http" menjadi tidak ada karakter. Implementasi *text preprocessing* dapat dilihat pada Kode Sumber 4.2.

4.2.1.2. *Split* Data Menjadi *Data Training* dan *Data Testing*

Tahapan selanjutnya adalah membagi data menjadi data *training* sebesar 85% dan data *testing* sebesar 15%. Data *training* akan digunakan untuk proses *training* dengan model CNN yang telah dibangun. Kemudian akan dilakukan evaluasi model dengan melakukan klasifikasi terhadap data *testing*.

Fungsi yang digunakan pada implementasi tahap ini adalah train_test_split dari *library* Scikit-learn yang memiliki fungsi untuk membagi data menjadi data *training* dan data *testing*. Kode Sumber 4.3 menunjukkan implementasi data *split*.

```
import pandas as pd
from sklearn.model_selection import train_test_split

csv = "data/clean_all.csv"
df = pd.read_csv(csv, header=0, index_col=None)
df['text'] = df.text.astype(str)

x = df.text
y = df.target
SEED = 42

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.15,
random_state=SEED)

df_train = pd.DataFrame({'text':x_train, 'target':y_train})
df_train.to_csv('data/ready/data_train.csv',encoding='utf-8')
```

```
df_test = pd.DataFrame({'text':x_test, 'target':y_test})
df_test.to_csv('data/ready/data_test.csv',encoding='utf-8')
```

Kode Sumber 4.3 Implementasi data *split*

4.2.1.3. *Training* Model *Word Embedding*

Pada tahap ini dilakukan *training* model *word embedding* pada data *training*. Model yang digunakan adalah *Word2Vec Skip-gram*.

```
import multiprocessing
import pandas as pd
from gensim.models.word2vec import Word2Vec
from gensim.models.doc2vec import TaggedDocument
from gensim.models import KeyedVectors

csv = "data/ready/data_train.csv"
df = pd.read_csv(csv, header=0, index_col=None)
df['text'] = df.text.astype(str)

tweets = df.text

def tagging_tweets(tweets,tag):
    result = []
    prefix = tag
    for i, t in zip(tweets.index, tweets):
        result.append(TaggedDocument(t.split(), [prefix + '_%s' % i]))
    return result

all_x_w2v = tagging_tweets(tweets, 'tweet')

cores = multiprocessing.cpu_count()
model_sg = Word2Vec(sg=1, size=100, workers=cores, min_count=3)
model_sg.build_vocab([x.words for x in all_x_w2v])
model_sg.train([x.words for x in all_x_w2v],
total_examples=len(all_x_w2v), epochs=20)

model_sg.save('model/w2v/model_sg.w2v')
```

Kode Sumber 4.4 Implementasi *word embedding*

Pada implementasi tahap ini digunakan fungsi Word2Vec yang disediakan oleh *library* gensim. Fungsi ini memiliki beberapa parameter yang akan menentukan bagaimana fungsi ini akan melakukan *training* terhadap data yang diberikan. Penulis menentukan parameter "sg" bernilai 1 yang artinya Word2Vec akan menggunakan model *Skip-gram*, "size" bernilai 100 yang menjadi jumlah vektor bobot tiap kata, dan "min_count" bernilai 3 yang artinya kata dengan frekuensi kemunculan lebih kecil dari 3 tidak akan dihitung. Implementasi *word embedding* dapat dilihat pada Kode Sumber 4.4.

4.2.1.4. Implementasi *Word Encoding*

Proses ini bertujuan untuk merubah data yang berupa teks menjadi data yang dapat diproses oleh model CNN. Proses ini terdiri dari *sequencing* dan *padding*.

Fungsi texts_to_sequences yang digunakan bertujuan untuk mengubah data teks menjadi angka yang merepresentasikan indeks kata tersebut dari *vocabulary*. Fungsi pad_sequences digunakan untuk mengubah dimensi tiap data menjadi seragam yaitu berjumlah maxlen yang sudah ditentukan sebelumnya yaitu 45. Variabel maxlen ditentukan lebih besar dari jumlah kata terbanyak dari setiap data *tweet* yaitu 41. Implementasi dapat dilihat pada Kode Sumber 4.5.

```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(num_words=vocab_size)
tokenizer.fit_on_texts(x)
sequences_train = tokenizer.texts_to_sequences(x)
x_seq = pad_sequences(sequences_train, maxlen=maxlen)
```

Kode Sumber 4.5 Implementasi *word encoding*

4.2.1.5. Perancangan Arsitektur Model CNN

Tahap selanjutnya adalah perancangan arsitektur model CNN yang akan digunakan untuk *training*. Model yang dibangun yaitu AlexNet, ZFNet, VGG16, VGG19, LeNet-5, dan model yang dibangun oleh penulis. Berikut adalah implementasi arsitektur model yang dibangun:

a. AlexNet

Model ini terdiri dari 5 *convolutional layer* dan 3 *fully connected layer* dengan fungsi aktivasi *relu* pada empat *layer* pertama dan *sigmoid* pada *layer* terakhir.

Parameter pada *convolutional layer* pertama yaitu *filter* berjumlah 96, *kernel* dengan ukuran 11, dan *strides* 4 kemudian diikuti dengan *layer MaxPooling* dengan ukuran 3 dan *strides* 4.

Kemudian *convolutional layer* kedua dengan *filter* berjumlah 256, ukuran *kernel* 5, dan *strides* 2 serta diikuti dengan *layer MaxPooling* yang memiliki parameter sama dengan *MaxPooling* sebelumnya.

Kemudian ketiga *convolution layer* selanjutnya memiliki jumlah *filter* 384, 383, dan 256 dengan ukuran *kernel* dan *strides* sama yaitu 3 dan 1 serta diikuti dengan *layer MaxPooling* yang memiliki parameter sama dengan *MaxPooling* sebelumnya.

Pada *fully connected layer* parameternya adalah jumlah *neuron* 4096 pada kedua *layer* pertama dan 1 pada *layer* terakhir untuk klasifikasi. Implementasi dapat dilihat pada Kode Sumber 4.6.

```
def get_model_alexnet():
    model = Sequential()
    em = Embedding(input_dim=embedding_matrix.shape[0],
output_dim=embedding_matrix.shape[1], input_length=maxlen,
weights=[embedding_matrix], trainable=False)
    model.add(em)

    model.add(Conv1D(96, kernel_size=11, strides=4))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=3, strides=2))
    model.add(BatchNormalization())

    model.add(Conv1D(256, kernel_size=5, strides=1, padding='same'))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))
    model.add(BatchNormalization())

    model.add(Conv1D(384, kernel_size=3, strides=1, padding='same'))
    model.add(Activation('relu'))
    model.add(Conv1D(384, kernel_size=3, strides=1, padding='same'))
    model.add(Activation('relu'))
    model.add(Conv1D(256, kernel_size=3, strides=1, padding='same'))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))

    model.add(Flatten())
    model.add(Dense(4096))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    model.add(Dense(4096))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    model.summary()

    return model
```

Kode Sumber 4.6 Implementasi arsitektur AlexNet

b. ZFNet

Model ini hampir sama dengan AlexNet, perbedaannya terdapat pada *layer* pertama dengan ukuran *kernel* dan *strides* lebih kecil yaitu 7 dan 4, serta pada *layer* ketiga, keempat dan kelima dengan jumlah *filter* 512, 1024, dan 512. Implementasi model ZFNet dapat dilihat pada Kode Sumber 4.7.

```
def get_model_zfnet():
    model = Sequential()
    em = Embedding(input_dim=embedding_matrix.shape[0],
output_dim=embedding_matrix.shape[1], input_length=maxlen,
weights=[embedding_matrix], trainable=False)
    model.add(em)

    model.add(Conv1D(96, kernel_size=7, strides=2))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=3, strides=2))
    model.add(BatchNormalization())

    model.add(Conv1D(256, kernel_size=5, strides=1, padding='same'))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))
    model.add(BatchNormalization())

    model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
    model.add(Activation('relu'))
    model.add(Conv1D(1024, kernel_size=3, strides=1, padding='same'))
    model.add(Activation('relu'))
    model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))

    model.add(Flatten())
    model.add(Dense(4096))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
```

```
    model.add(Dense(4096))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    model.summary()

    return model
```

Kode Sumber 4.7 Implementasi arsitektur ZFNet

c. VGG16

Model ini memiliki total 16 *layer* yang terdiri dari 13 *convolutional layer* dan 3 *fully connected layer*. Dengan ukuran *filter* 64 pada dua *convolutional layer* pertama yang diikuti *MaxPooling layer*, ukuran *filter* 128 pada *convolutional layer* ketiga dan keempat yang diikuti *MaxPooling layer*, ukuran *filter* 256 pada *convolutional layer* kelima, keenam, dan ketujuh yang diikuti *MaxPooling layer*, serta ukuran *filter* 512 pada *convolutional layer* sisanya yang diikuti *MaxPooling layer*. Implementasi dari model VGG16 dapat dilihat pada

```
model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    model.summary()

    return model
```

Kode Sumber 4.8.

```
def get_model_vgg16():
    model = Sequential()
    em = Embedding(input_dim=embedding_matrix.shape[0],
output_dim=embedding_matrix.shape[1], input_length=maxlen,
weights=[embedding_matrix], trainable=False)
    model.add(em)
```

```
model.add(Conv1D(64, kernel_size=3, strides=1))
model.add(Activation('relu'))
model.add(Conv1D(64, kernel_size=3, strides=1))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=3, strides=2))

model.add(Conv1D(128, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))

model.add(Conv1D(256, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(256, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(256, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))

model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))

model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))

model.add(Flatten())
model.add(Dense(4096))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(4096))
model.add(Activation('relu'))
model.add(Dropout(0.5))
```

```
model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    model.summary()

    return model
```

Kode Sumber 4.8 Implementasi arsitektur VGG16

d. VGG19

Model ini hampir sama dengan VGG19, namun memiliki total 19 *layer* dengan 16 *convolutional layer* dan 3 *fully connected layer*. Perbedaannya terdapat tambahan tiga *convolutional layer*, pada *convolutional layer* kedelapan dengan ukuran *filter* 256, serta 512 pada *convolutional layer* kedua belas dan keenam belas. Implementasi dari model VGG19 dapat dilihat pada Kode Sumber 4.9.

```
def get_model_vgg19():
    model = Sequential()
    em = Embedding(input_dim=embedding_matrix.shape[0],
output_dim=embedding_matrix.shape[1], input_length=maxlen,
weights=[embedding_matrix], trainable=False)
    model.add(em)

    model.add(Conv1D(64, kernel_size=3, strides=1))
    model.add(Activation('relu'))
    model.add(Conv1D(64, kernel_size=3, strides=1))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=3, strides=2))
```

```
model.add(Conv1D(128, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))

model.add(Conv1D(256, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(256, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(256, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(256, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))

model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))

model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(512, kernel_size=3, strides=1, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))

model.add(Flatten())
model.add(Dense(4096))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(4096))
model.add(Activation('relu'))
```

```
    model.add(Dropout(0.5))

    model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    model.summary()

    return model
```

Kode Sumber 4.9 Implementasi arsitektur VGG19

e. LeNet-5

Model ini terdiri dari 2 *convolutional layer* dan 2 *fully connected layer*. Parameter pada *convolutional layer* pertama adalah ukuran *filter* 20, ukuran *kernel* 5, dan *strides* 1 kemudian diikuti oleh *MaxPooling layer* dengan ukuran *pool* 2 dan *strides* 2. Sedangkan parameter pada *convolutional layer* kedua adalah ukuran *filter* 50, ukuran *kernel* 5, dan *strides* 1 kemudian diikuti oleh *MaxPooling layer* dengan ukuran *pool* 2 dan *strides* 2. Kemudian pada *fully connected layer* pertama *neuron* berjumlah 500 dan 1 pada *fully connected layer* kedua. Implementasi model ini dapat dilihat pada Kode Sumber 4.10.

```
def get_model_lenet():
    model = Sequential()
    em = Embedding(input_dim=embedding_matrix.shape[0],
output_dim=embedding_matrix.shape[1], input_length=maxlen,
weights=[embedding_matrix], trainable=False)
    model.add(em)
```

```
    model.add(Conv1D(20, kernel_size=5, strides=1))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=2, strides=2))

    model.add(Conv1D(50, kernel_size=5, strides=1, padding='same'))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))

    model.add(Flatten())
    model.add(Dense(500))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    model.summary()

    return model
```

Kode Sumber 4.10 Implementasi arsitektur LeNet-5

f. 3 *layers*

Model ini terdiri dari 3 *convolutional layer* dan 2 *fully connected layer*. Parameter yang digunakan adalah ukuran *filter* 16, ukuran *kernel* 3, dan *strides* 1 pada semua *convolutional layer* kemudian diikuti *MaxPooling layer* dengan ukuran *pool* 3 dan *strides* 1. Sedangkan parameter pada *fully connected layer* pertama yaitu *neuron* 128 dan *neuron* 1 pada *fully connected layer* kedua. Model ini melakukan *training* dengan optimal ketika menggunakan *optimizer* Nadam dengan *learning rate* 0,0001 yang akan dijelaskan pada Bab V tentang uji coba dan evaluasi. Implementasi model yang

dirancang penulis ini dapat dilihat pada Kode Sumber 4.11.

```
def get_model_3layer():
    model = Sequential()
    em = Embedding(input_dim=embedding_matrix.shape[0],
output_dim=embedding_matrix.shape[1], input_length=maxlen,
weights=[embedding_matrix], trainable=False)
    model.add(em)

    model.add(Conv1D(16, kernel_size=3, strides=2))
    model.add(Activation('relu'))
    model.add(Conv1D(16, kernel_size=3, strides=2))
    model.add(Activation('relu'))
    model.add(Conv1D(16, kernel_size=3, strides=2))
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=3, strides=2, padding='same'))

    model.add(Flatten())
    model.add(Dense(128))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
    model.summary()

    return model
```

Kode Sumber 4.11 Implementasi arsitektur 3 *layers*

Setelah model arsitektur dibangun, dilakukan proses *training* dengan *cross validation stratified 5-fold* sebagai metode validasi. Kemudian program menyimpan model dari *training* pada *fold* terakhir. Implementasi dari proses *training* dapat dilihat pada Kode Sumber 4.12.

```
model = get_model_alexnet()

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=SEED)
i=0
for train_index, val_index in skf.split(x_seq, y):
    x_train, x_val = x_seq[train_index], x_seq[val_index]
    y_train, y_val = y[train_index], y[val_index]

    print("\nFold ", i+1)
    model.fit(x_train, y_train, batch_size=68, validation_data=(x_val,
y_val), epochs=10)

    i += 1

model.save('model/cnn/alexnet.h5')
```

Kode Sumber 4.12 Implementasi proses *training*

4.2.2. Implementasi Klasifikasi Data *Test* dan Evaluasi

Setelah proses *training* telah dilakukan tahapan selanjutnya adalah tahapan uji coba terhadap *data testing*. Kemudian hasil prediksi akan disimpan dalam bentuk file. Implementasi dapat dilihat pada Kode Sumber 4.13.

```
cnn = "model/cnn/alexnet.h5"
model = load_model(cnn)

print("\nScore on Test data")
score = model.evaluate(x_test_seq, y_test)
accuracy = str(score[1]*100)
loss = str(score[0])
print("Accuracy: ", accuracy)
print("Total loss: ", loss)

preds = model.predict_classes(x_test_seq)
df = df.assign(prediction=preds)
df.to_csv("data/result/alexnet_%sacc_%sloss.csv"%(accuracy, loss),
sep=';')
```

Kode Sumber 4.13 Implementasi klasifikasi dan evaluasi

BAB V
UJI COBA DAN EVALUASI

Pada Bab ini akan dijelaskan mengenai skenario uji coba pada perangkat lunak yang telah dibangun. Selanjutnya, hasil uji coba akan dianalisa kinerjanya sehingga dapat diketahui kriteria, kelebihan dan kekurangan dari klasifikasi data *tweet* Twitter menggunakan CNN. Secara garis besar, bab ini berisikan pembahasan mengenai lingkungan pengujian, data pengujian, dan uji kinerja.

5.1. Lingkungan Pengujian

Lingkungan pengujian tugas akhir ini menggunakan spesifikasi perangkat keras dan perangkat lunak seperti yang ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi lingkungan uji coba

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz 2.5 GHz
	Memori	4 GB
	GPU	NVIDIA GeForce GTX 1050
Perangkat Lunak	Sistem Operasi	Windows 10 64-bit
	Bahasa Pemrograman	Python 3.6
	Perangkat Pengembang	Anaconda3 5.3.1

5.2. Data Pengujian

Data yang digunakan sebagai masukan uji coba adalah 1000 data *tweet* Twitter. Data terbagi menjadi dua kelas yaitu normal dan depresi dengan jumlah 500 data untuk masing-masing kelas. Kemudian data dibagi menjadi data *training* sebanyak 850 *tweet* dan data *testing* sebanyak 150 *tweet*. Seluruh data yang digunakan dalam proses *testing* berbeda dengan data yang digunakan saat proses *training*.

5.3. Hasil Uji Fungsionalitas

Pengujian fungsionalitas dilakukan sesuai dengan tahapan pada desain umum sistem yang sudah dijelaskan pada Bab II dengan implementasi yang sudah dijelaskan. Hasil pengujian dapat dijabarkan pada subbab berikut ini.

5.3.1. Hasil *Text Preprocessing*

Tahap ini dilakukan setelah data mendapatkan label dengan cara manual. Data yang sudah dilabeli akan melalui tahap *cleansing*, *stopwords removal*, *stemming*, dan *tokenizing*. Contoh hasil *text preprocessing* dapat dilihat pada Tabel 5.2.

Tabel 5.2 Contoh hasil *text preprocessing*

No	Data *tweet*	Hasil *Text Preprocessing*
1	berkutik dengan depresi akhir akhir ini sampai benar benar ingin mati. ah aku butuh teman.	kutik depresi akhir akhir sampai benar benar ingin mati aku butuh teman
2	Ku ingin mati saja tidur selamanya daripada ku tak dapat bercerita kepada	ingin mati tidur lama tak cerita siapa nyata siksa

	siapapun, ternyata ini menyiksa...	
3	Saat ini lah titik terendah semasa hidup saya, jiwa dan otak saya tiap hari mendidih..mempertanyakan sebenernya siapa saya? Apa yg bisa saya lakukan? Saya depresi karena diri sendiri... pekerjaan saya amburadul, bahkan saya tidak bisa berpikir atau melakukan hal kecil...	saat lah titik rendah masa hidup saya jiwa otak saya tiap hari didih tanya sebenernya siapa saya apa saya laku saya depresi diri sendiri kerja saya amburadul bahkan saya bisa pikir laku kecil

5.3.2. Hasil *Word Embedding*

Metode yang digunakan pada tahap ini adalah *skip-gram* dari *Word2Vec*. Contoh hasil *word embedding* dari hasil *text preprocessing* data *tweet* nomor 1 pada Tabel 5.2 dapat dilihat pada Tabel 5.3.

Tabel 5.3 Contoh hasil *word embedding*

Kata	Hasil *word embedding*
Kutik	Kata ini tidak terdapat di *vocabulary*
Depresi	[0.0740053 -0.18648677 -0.10990714 -0.15998779 0.12550208 …]
Akhir	[0.10621303 -0.0560851 -0.03344292 -0.10922116 0.05492952 …]
Sampai	[0.0921046 -0.02839691 -0.0015291 -0.08244325 0.09586354 …]

Benar	[0.27671915 -0.24620165 -0.07012851 -0.22119413 0.30746502 …]
Ingin	[0.10060851 -0.01581069 0.00640759 -0.16726646 0.04159175 …]
Mati	[1.09981574e-01 -6.78237602e-02 1.51207363e-02 -1.37527779e-01 …]
Aku	[0.13000272 -0.01528139 0.03113079 -0.11873585 0.06205969 …]
Butuh	[0.1121779 -0.06411174 -0.00399083 -0.14889325 0.12200766 …]
Teman	[0.12330348 -0.10066358 -0.01837594 -0.14403366 0.0880699 …]

Dapat dilihat pada Tabel 5.3 yang menampilkan hasil *word embedding* bahwa kata "kutik" tidak terdapat pada *vocabulary*. Hal ini disebabkan oleh nilai pada parameter min_count dari fungsi Word2Vec telah ditentukan penulis yaitu sebesar 3. Artinya, sistem hanya akan memasukkan kata-kata dengan frekuensi kemunculan berjumlah 3 atau lebih besar ke dalam *vocabulary*. Dengan kata lain, kata "kutik" berarti memiliki frekuensi kemunculan lebih rendah dari 3, sehingga tidak dimasukkan ke dalam *vocabulary*.

5.3.3. Hasil *Word Encoding*

Proses ini terdiri dari dua tahapan yaitu *sequencing* dan *padding*. *Sequencing* bertujuan untuk merepresentasikan data

teks menjadi angka yang menjadi indeks tiap kata pada *vocabulary*.

Selanjutnya adalah proses *padding*. Proses ini akan merubah kalimat menjadi kumpulan array yang terdiri dari kumpulan kata-kata. Dalam proses ini jumlah dimensi array harus ditentukan terlebih dahulu supaya seluruh data teks yang diproses memiliki ukuran yang sama. Contoh hasil *word encoding* dapat dilihat pada Tabel 5.4.

Tabel 5.4 Contoh hasil *word encoding*

<table>
<tr><th>No</th><th colspan="2">Teks</th></tr>
<tr><td rowspan="3">1</td><td colspan="2">kutik depresi akhir akhir sampai benar benar ingin mati aku butuh teman</td></tr>
<tr><td>Sequence</td><td>Padding</td></tr>
<tr><td>[5, 10, 10, 177, 88, 88, 4, 8, 3, 132, 63]</td><td>[0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 5 10
10 177 88 88 4 8 3 132
63]</td></tr>
<tr><td rowspan="3">2</td><td colspan="2">ingin mati tidur lama tak cerita siapa nyata siksa</td></tr>
<tr><td>Sequence</td><td>Padding</td></tr>
<tr><td>[4, 8, 117, 50, 26, 230, 95, 80, 508]</td><td>[0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
4 8 117 50 26 230 95 80
508]</td></tr>
<tr><td rowspan="3">3</td><td colspan="2">saat lah titik rendah masa hidup saya jiwa otak saya tiap hari didih tanya sebenernya siapa saya apa saya laku saya depresi diri sendiri kerja saya amburadul bahkan saya bisa pikir laku kecil</td></tr>
<tr><td>Sequence</td><td>Padding</td></tr>
<tr><td>[73, 87, 220, 575, 120, 1, 2, 118, 272, 2, 92, 49, 397,</td><td>[0 0 0 0 0 0 0 0 0 0
0 0 0 0 73 87 220 575</td></tr>
</table>

	576, 95, 2, 22, 2, 72, 2, 5, 24, 23, 82, 2, 90, 2, 85, 38, 72, 249]	120 1 2 118 272 2 92 49 397 576 95 2 22 2 72 2 5 24 23 82 2 90 2 85 38 72 249]

5.4. Skenario Uji Coba

Pada subbab ini akan dijelaskan mengenai skenario uji coba yang telah dilakukan. Terdapat beberapa skenario uji coba yang telah dilakukan, diantaranya yaitu:

1. Skenario pengujian 1: Penghitungan nilai akurasi pada model CNN berdasarkan jumlah *layer*.
2. Skenario pengujian 2: Penghitungan nilai akurasi pada model CNN berdasarkan jumlah *filter*.
3. Skenario pengujian 3: Penghitungan nilai akurasi pada model CNN berdasarkan jumlah *filter* pada masing-masing *layer*.
4. Skenario pengujian 4: Penghitungan nilai akurasi pada model CNN berdasarkan *learning rate optimizer* Adam.
5. Skenario pengujian 5: Penghitungan nilai akurasi pada model CNN berdasarkan jenis *optimizer*.
6. Skenario pengujian 6: Perbandingan nilai akurasi pada model AlexNet, ZFNet, VGG16, VGG19, LeNet-5, serta model yang dibangun penulis.

Proses *training* pada setiap model skenario uji coba memiliki beberapa parameter-parameter tetap seperti yang ditunjukkan pada Tabel 5.5.

Tabel 5.5 Parameter tetap skenario uji coba

Parameter	Nilai
Epoch	10
Kernel size	3
Pool size	3
Strides	2
Dropout rate	0,5
Embedding dimention	100
Sequence length	45

Pada setiap model skenario uji coba digunakan *stratified 5-fold* kemudian diambil model hasil dari *training* pada *fold* terakhir. Percobaan *training* dilakukan sebanyak lima kali, kemudian dilakukan prediksi terhadap data *test* dengan model yang didapat dari *training* dan hasil akurasi pada data *test* akan dirata-rata.

Setiap data akan dilakukan proses prediksi kelas pada data *test* dengan menggunakan model CNN yang sudah dibangun. Masing-masing model akan memprediksi kelas sebuah data dengan memberikan nilai 0 atau 1. Jika nilai kelas adalah 0 artinya data tersebut termasuk kelas normal dan berlaku sebaliknya untuk kelas *depresi*.

5.4.1. Skenario Uji Coba Penghitungan Nilai Akurasi Berdasarkan Jumlah *Layer*

Skenario ini bertujuan untuk menentukan jumlah *layer* yang paling optimal untuk *training*. Percobaan dilakukan dari 1 *layer* hingga 4 *layers* dengan jumlah *filter* yang sama pada tiap *layer* yaitu 16 serta *optimizer* adam dengan *learning rate* tetap yaitu sebesar 0,001. Hasil uji coba dapat dilihat pada Tabel 5.6.

Tabel 5.6 Hasil uji coba berdasarkan jumlah *layer*

Akurasi (%)	1 *layer*	2 *layers*	3 *layers*	4 *layers*
Percobaan 1	65,33	70	74,66	63,33
Percobaan 2	67,33	70	71,99	71,99
Percobaan 3	64,66	65,33	72,66	72
Percobaan 4	62,66	66,66	71,99	70,66
Percobaan 5	67,33	66,66	71,33	74
Rata-rata	65,46	67,73	**72,53**	70,40

Dari hasil uji coba pada Tabel 5.6 dapat disimpulkan bahwa model dengan 3 *layers* memiliki rata-rata nilai akurasi tertinggi yaitu 72,53%. Maka uji coba selanjutnya jumlah *layer* yang akan digunakan pada tiap model adalah 3.

5.4.2. Skenario Uji Coba Penghitungan Nilai Akurasi Berdasarkan Jumlah *Filter*

Skenario ini bertujuan untuk menentukan jumlah *filter* yang paling tepat pada model dengan hasil optimal yang didapatkan dari skenario sebelumnya. Percobaan ini dilakukan dengan *optimizer* adam dan *learning rate* tetap yaitu 0,001. Hasil uji coba dapat dilihat pada Tabel 5.7.

Tabel 5.7 Hasil uji coba berdasarkan jumlah *filter*

Akurasi (%)	*Filter* 8	*Filter* 16	*Filter* 32	*Filter* 64	*Filter* 128
Percobaan 1	71,33	74,66	70,67	68	70
Percobaan 2	71,33	71,99	72,66	67,99	65,99
Percobaan 3	72,66	72,66	70,66	69,33	66,66

Percobaan 4	72	71,99	65,99	65,99	64,66
Percobaan 5	71,99	71,33	72,66	69,33	67,33
Rata-rata	71,86	**72,53**	70,53	68,13	66,93

Dari hasil uji coba pada Tabel 5.7 dapat disimpulkan bahwa model dengan jumlah *filter* sebesar 16 pada tiap *layer* memiliki rata-rata nilai akurasi tertinggi yaitu 72,53%. Maka uji coba selanjutnya menggunakan jumlah *filter default* sebesar 16.

5.4.3. Skenario Uji Coba Penghitungan Nilai Akurasi Berdasarkan Jumlah *Filter* pada Masing-masing *Layer*

Pada skenario uji coba ini dilakukan penghitungan nilai akurasi dengan kombinasi jumlah *filter* dan jumlah *layer* optimal yang didapat dari skenario sebelumnya. Percobaan ini menggunakan *optimizer* adam dan *learning rate* tetap yaitu 0,001. Hasil uji coba berdasarkan jumlah *filter* pada *layer* pertama dapat dilihat pada Tabel 5.8.

Tabel 5.8 Hasil uji coba berdasarkan jumlah *filter layer* pertama

Akurasi (%)	*Filter* 8	*Filter* 16	*Filter* 32	*Filter* 64	*Filter* 128
Percobaan 1	71,33	74,66	70,67	74	67,8
Percobaan 2	73,99	71,99	70	67,33	66
Percobaan 3	72,66	72,66	69,99	69,99	67,33
Percobaan 4	72,66	71,99	71,33	71,33	66,66

Percobaan 5	71,33	71,33	76	69,99	67,33
Rata-rata	72,39	**72,53**	71,6	70,53	67,04

Dari hasil uji coba pada Tabel 5.8 dapat disimpulkan bahwa model 2 yaitu model dengan jumlah *filter* pada *layer* pertama sebesar 16 memiliki rata-rata nilai akurasi tertinggi yaitu 72,53%. Maka uji coba selanjutnya *filter* pada *layer* pertama akan menggunakan nilai 16 pada tiap model dan akan dilakukan perubahan jumlah *filter* pada *layer* kedua. Hasil uji coba berdasarkan jumlah *filter* pada *layer* kedua dapat dilihat pada Tabel 5.9.

Tabel 5.9 Hasil uji coba berdasarkan jumlah *filter layer* kedua

Akurasi (%)	*Filter* 8	*Filter* 16	*Filter* 32	*Filter* 64	*Filter* 128
Percobaan 1	71,33	74,66	73,99	71,33	69,99
Percobaan 2	68,66	71,99	69,99	68,66	67,99
Percobaan 3	73,99	72,66	70,66	71,99	69,33
Percobaan 4	70,66	71,99	72	71,99	63,99
Percobaan 5	71,99	71,33	71,33	72,66	68,66
Rata-rata	71,33	**72,53**	71,59	71,33	67,99

Dari hasil uji coba pada Tabel 5.9 dapat disimpulkan bahwa model 2 yaitu model dengan jumlah *filter* pada *layer* kedua sebesar 16 memiliki rata-rata nilai akurasi tertinggi yaitu 72,53%. Maka uji coba selanjutnya *filter* pada *layer* pertama dan kedua akan menggunakan nilai 16 pada tiap model dan akan dilakukan perubahan jumlah *filter* pada *layer* ketiga. Hasil uji

coba berdasarkan perubahan jumlah *filter* pada *layer* ketiga dapat dilihat pada Tabel 5.10.

Tabel 5.10 Hasil uji coba berdasarkan jumlah *filter layer* ketiga

Akurasi (%)	*Filter* 8	*Filter* 16	*Filter* 32	*Filter* 64	*Filter* 128
Percobaan 1	71,33	74,66	72,66	68,66	70,66
Percobaan 2	69,99	71,99	66	70	65,99
Percobaan 3	69,33	72,66	67,99	69,33	65,99
Percobaan 4	69,99	71,99	67,33	68	68,66
Percobaan 5	73,33	71,33	71,33	69,33	67,33
Rata-rata	70,79	**72,53**	69,06	69,06	67,73

Dari hasil uji coba pada Tabel 5.10 dapat disimpulkan bahwa model 2 yaitu model dengan jumlah *filter* pada *layer* ketiga sebesar 16 memiliki rata-rata nilai akurasi tertinggi yaitu 72,53%. Maka uji coba selanjutnya *filter* pada *layer* pertama, kedua, dan ketiga akan menggunakan nilai 16 pada tiap model.

5.4.4. Skenario Uji Coba Penghitungan Nilai Akurasi Berdasarkan *Learning Rate Optimizer* Adam

Skenario ini bertujuan untuk mencari nilai *learning rate* pada *optimizer* Adam yang paling sesuai dengan model yang telah didapatkan dari skenario sebelumnya. Hasil uji coba dapat dilihat pada Tabel 5.11.

Tabel 5.11 Hasil uji coba berdasarkan *learning rate optimizer* (η) Adam

Akurasi (%)	η 0,1	η 0,01	η 0,001	η 0,0001	η 0,00001
Percobaan 1	53,33	72	74,66	75,99	70,66
Percobaan 2	46,66	71,33	71,99	74,66	55,33
Percobaan 3	53,33	71,99	72,66	74,66	46,66
Percobaan 4	53,33	67,99	71,99	74,66	46,66
Percobaan 5	46,66	71,33	71,33	74,66	73,33
Rata-rata	50,66	70,93	72,53	**74,93**	58,53

Dari hasil uji coba yang didapatkan dengan *optimizer* Adam, nilai rata-rata akurasi tertinggi adalah model dengan *learning rate* sebesar 0,0001 yaitu sebesar 74,93%.

5.4.5. Skenario Uji Coba Penghitungan Nilai Akurasi Berdasarkan Jenis *Optimizer*

Skenario ini bertujuan untuk mencari jenis *optimizer* yang paling sesuai dengan model yang sudah didapat dari skenario-skenario sebelumnya. Pada skenario ini akan digunakan lima jenis *optimizer* yang tersedia pada *library* keras yaitu Adam, SGD, Adagrad, Nadam, dan RMSprop dengan *learning rate* tetap yang sudah didapat dari skenario uji coba sebelumnya yaitu sebesar 0,0001. Hasil uji coba dapat dilihat pada Tabel 5.12.

Tabel 5.12 Hasil uji coba berdasarkan jenis *optimizer*

Akurasi (%)	Adam	SGD	Adagrad	Nadam	RMSprop
Percobaan 1	75,99	53,33	64	76,66	75,99
Percobaan 2	74,66	53,33	73,33	75,33	74,66
Percobaan 3	74,66	46,66	72,66	75,33	74,66
Percobaan 4	74,66	46,66	46	74,66	75,33
Percobaan 5	74,66	56,66	46	74,66	75,33
Rata-rata	74,93	51,33	60,4	**75,33**	75,19

Dari Tabel 5.12 dapat disimpulkan bahwa model dengan *optimizer* Nadam memiliki rata-rata akurasi tertinggi yaitu 75,33%.

5.4.6. Skenario Uji Coba Perbandingan Nilai Akurasi Berdasarkan Arsitektur CNN

Skenario ini bertujuan untuk membandingkan nilai akurasi yang dihasilkan dari setiap model. Pada arsitektur 3 *layers* akan diambil akurasi tertinggi dari skenario sebelumnya. Hasil uji coba dapat dilihat pada Tabel 5.13.

Tabel 5.13 Hasil uji coba berdasarkan arsitektur CNN

Akurasi (%)	Alex Net	ZF Net	VGG 16	VGG 19	LeNet-5	3 *Layers*
Percobaan 1	59,99	59,33	61,33	60,66	70	76,66
Percobaan 2	60	59,33	61,33	65,33	69,33	75,33

Percobaan 3	57,33	63,33	63,99	64,66	70,66	75,33
Percobaan 4	60	61,99	62	62	70,66	74,66
Percobaan 5	61,33	61,33	62	60	70	74,66
Rata-rata	59,73	61,06	62,13	62,53	70,13	**75,33**

5.5. Analisis Uji Coba

Pada skenario uji coba pertama dilakukan percobaan untuk mencari jumlah *layer* yang optimal untuk *training* data dengan hasil tertinggi pada uji coba skenario pertama yaitu model 3 *layers* dengan rata-rata akurasi 72,53%.

Skenario uji coba kedua menggunakan model 3 *layers* yang sebelumnya diuji coba untuk dilakukan perubahan jumlah *filter*. Hasil percobaan skenario kedua menunjukkan bahwa rata-rata akurasi tertinggi didapat dari model dengan jumlah *filter* 16 pada tiap *layer* dengan nilai 72,53%.

Skenario ketiga dilakukan perubahan jumlah *filter* pada tiap *layer*. Rata-rata akurasi tertinggi pada skenario ketiga didapat dari model dengan jumlah *filter* 16 pada tiap *layer* yaitu sebesar 72,53%.

Kemudian pada skenario keempat model yang sudah didapat dari skenario sebelumnya diuji menggunakan *optimizer* Adam dengan *learning rate* berbeda untuk mencari *learning rate* yang menghasilkan rata-rata akurasi terbaik. Hasil uji coba pada skenario keempat menunjukkan bahwa model dengan *learning rate* 0,0001 mendapatkan rata-rata akurasi sebesar 74,93%.

Pada skenario uji coba kelima dilakukan percobaan menggunakan semua optimizer dengan *learning rate* yang mendapatkan akurasi terbaik skenario sebelumnya. Rata-rata akurasi tertinggi pada skenario kelima didapat dari model

dengan *optimizer* Nadam yang menggunakan *learning rate* sebesar 0,0001, yaitu sebesar 75,33%.

Skenario keenam dilakukan percobaan pada enam arsitektur CNN yaitu AlexNet, ZFNet, VGG16, VGG19, LeNet-5 dan 3 *layers* yang menggunakan *optimizer* Nadam dengan *learning rate* 0,0001. Hasil uji coba pada skenario keenam menunjukkan bahwa arsitektur dengan rata-rata akurasi tertinggi adalah 3 *layers* dengan nilai 75,33%.

Hal ini menunjukkan bahwa arsitektur yang kompleks atau memiliki banyak *layer* dan *filter* seperti AlexNet, ZFNet, VGG16, dan VGG19 cenderung *overfit* sehingga menghasilkan nilai rata-rata akurasi validasi tinggi tetapi rata-rata akurasi *testing* yang rendah. Tabel yang menampilkan *loss* dari hasil semua skenario uji coba dapat dilihat pada lampiran a.

Kemudian dilakukan juga analisis data yang memiliki kecenderungan mengalami kesalahan prediksi. Contoh data yang terdapat kesalahan klasifikasi dapat dilihat pada Tabel 5.14.

Tabel 5.14 Contoh data yang terdapat kesalahan klasifikasi

No	*Tweet*	Aktual	Prediksi
1	Baru lepas reinstall netflix. Selamat tinggal dunia luar.	normal	depresi
2	/tlw/ boleh minta semangat nya, lagi2 saya merasa saya bosan hidup dan saya merasa sendiri .	depresi	normal
3	Jangan mengakhiri ,hidup itu bukan penyesalan dikemudian hari.	normal	depresi

4	Bosan Hidup, Ilmuwan Berusia 104 tahun Ini Ingin Disuntik Mati http://dlvr.it/QpkkjS	normal	depresi
5	Aku dah putus asa nak pertahankan benda yang belum tentu pun sebenarnya ikhlas dengan kau atau tak sebenarnya. Aku cuma ikut ke mana takdir bawak aku	depresi	normal

Dari data nomor 1, 3 dan 4 yang seharusnya masuk kategori normal, diprediksi depresi. Kemungkinan hal ini karena data mengandung kata yang memiliki konteks yang lebih condong ke arah kelas depresi seperti ‘ingin mati’, ‘selamat tinggal dunia’ dan ‘mengakhiri hidup’. Sedangkan data nomor 2 dan 5 mengalami kesalahan prediksi yang seharusnya depresi, hasil prediksi menjadi normal. Hal ini kemungkinan dikarenakan mengandung kata yang memiliki konteks yang lebih condong ke arah kelas normal seperti ‘semangat’ dan ‘ikhlas’. Kesalahan prediksi ini juga dapat diakibatkan oleh tahap *text preprocessing* yang belum sempurna. Semua data yang terdapat kesalahan klasifikasi dapat dilihat pada lampiran b.

BAB VI
KESIMPULAN DAN SARAN

Bab VI ini membahas tentang kesimpulan yang didasari oleh hasil uji coba pada bab sebelumnya. Kesimpulan tersebut nantinya menjawab rumusan masalah yang telah ada pada pendahuluan. Selain itu, juga terdapat saran sebagai acuan untuk mengembangkan topik Tugas Akhir ini lebih lanjut di masa depan.

6.1. Kesimpulan

Dari hasil pengujian pada skenario-skenario yang telah dilakukan, dapat diambil kesimpulan bahwa:

1. Arsitektur dengan *layer* yang berjumlah tiga adalah arsitektur paling optimal dengan rata-rata akurasi 72,53%.
2. Jumlah *filter* paling optimal pada arsitektur dengan *layer* berjumlah tiga adalah *filter* berjumlah 16 dengan rata-rata akurasi 72,53%.
3. *Optimizer* yang paling tepat digunakan pada arsitektur ini adalah Nadam dengan *learning rate* sebesar 0,0001 dengan rata-rata akurasi 75,33%.
4. Arsitektur yang dalam (terdiri dari banyak *layer*) dan menggunakan parameter yang banyak akan menghasilkan rata-rata akurasi yang kecil.

6.2. Saran

Berikut adalah beberapa saran untuk pengembangan penelitian dengan kasus sama di masa yang akan datang:

1. Memperbanyak jumlah dan variasi data untuk menghindari *overfit* dan mendapatkan hasil akurasi yang diharapkan dari model CNN yang dibuat.
2. Metode *preprocessing* dapat diganti dengan cara lain seperti tidak melakukan *stop word rermoval* dan *stemming* karena metode tersebut dapat mengubah arti atau konteks data teks.
3. Dibutuhkan ketelitian yang lebih saat melakukan pelabelan data *tweet* karena tidak mudah untuk memahami maksud dari *tweet* tersebut.
4. Dapat dilakukan juga pengambilan model dengan cara berbeda dari Tugas Akhir ini, yang menggunakan model hasil *training* pada *fold* terakhir, yaitu mengambil model dengan akurasi tertinggi pada salah satu *fold*.

Daftar Pustaka

[1] "Southeast Asia Digital, Social, and Mobile 2018," Asean Up, 18 April 2018. [Online]. Available: https://aseanup.com/southeast-asia-digital-social-mobile/. [Accessed 4 June 2018].

[2] "Suicide Rate by Country 2018," Wolrd Population Review, [Online]. Available: http://www.worldpopulationreview.com/countries/suicide-rate-by-country/. [Accessed 15 January 2019].

[3] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, "Handwritten Digit Recognition with a Back-Propagation Network," AT&T Laboratories, Holmdel, 1990.

[4] A. H. Orabi, P. Buddhitha, M. H. Orabi and D. Inkpen, "Deep Learning for Depression Detection of Twitter Users," *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pp. 88-97, 2018.

[5] D. Singh and A. Wang, "Detecting Depression Through Tweets," Stanford University, Stanford, 2017.

[6] "Twitter," Wikipedia, [Online]. Available: http://www.id.wikipedia.org/wiki/Twitter. [Accessed 4 June 2018].

[7] "Number of monthly active international Twitter users from 1st quarter 2010 to 1st 2018 (in millions)," Statista, 2018. [Online]. Available: http://www.statista.com/statistics/274565/monthly-active-international-twitter-users/. [Accessed 4 June 2018].

[8] "What is Depression?," American Psychiatric Association, January 2017. [Online]. Available: https://www.psychiatry.org/patients-families/depression/what-is-depression/. [Accessed 4 June 2018].

[9] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Mountain View, 2013.

[10] Y. Le Cun, K. Kavukcuoglu and F. Clement, "Convolutional Networks and Application in Vision," International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems, Paris, 2010.

[11] Y. Zhang and B. C. Wallace, "A Sensitivity Analysis of (and Practitioner's Guide to) Convolutional Neural Networks for Sentence Classification," 2015.

[12] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms," 2017.

[13] J. Henrique, "Get Old Tweets Programmatically," 2015. [Online]. Available: https://github.com/Jefferson-Henrique/GetOldTweets-python. [Accessed 25 November 2018].

[14] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Network," 2012.

[15] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," New York, 2013.

[16] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2015.

[17] Y. Le Cun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," IEEE, 1998.

[Halaman ini sengaja dikosongkan]

LAMPIRAN A

A1. Hasil uji coba berdasarkan jumlah *layer*

Loss	1 *layer*	2 *layers*	3 *layers*	4 *layers*
Percobaan 1	0,68	0,64	0,61	0,72
Percobaan 2	0,69	0,64	0,6	0,59
Percobaan 3	0,69	0,67	0,55	0,62
Percobaan 4	0,68	0,58	0,59	0,65
Percobaan 5	0,65	0,58	0,65	0,62
Rata-rata	0,68	0,63	0,6	0,64

A2. Hasil uji coba berdasarkan jumlah *filter*

Loss	*Filter* 8	*Filter* 16	*Filter* 32	*Filter* 64	*Filter* 128
Percobaan 1	0,61	0,61	0,73	1,1	1,1
Percobaan 2	0,56	0,6	0,72	1,19	1,3
Percobaan 3	0,59	0,55	0,72	0,94	1,2
Percobaan 4	0,65	0,59	0,89	1,1	1,19
Percobaan 5	0,58	0,65	0,73	1,2	1
Rata-rata	0,59	0,6	0,76	1,11	1,59

A3. Hasil uji coba berdasarkan jumlah *filter layer* pertama

Loss	*Filter* 8	*Filter* 16	*Filter* 32	*Filter* 64	*Filter* 128
Percobaan 1	0,6	0,61	0,62	0,67	0,79

Percobaan 2	0,59	0,6	0,72	0,79	0,85
Percobaan 3	0,59	0,55	0,67	0,69	0,85
Percobaan 4	0,67	0,59	0,64	0,75	0,76
Percobaan 5	0,67	0,65	0,69	0,72	0,82
Rata-rata	0,62	0,6	0,67	0,72	0,81

A4. Hasil uji coba berdasarkan jumlah *filter layer* kedua

Loss	*Filter* 8	*Filter* 16	*Filter* 32	*Filter* 64	*Filter* 128
Percobaan 1	0,59	0,61	0,65	0,76	0,69
Percobaan 2	0,7	0,6	0,66	0,81	0,84
Percobaan 3	0,8	0,55	0,64	0,72	0,7
Percobaan 4	0,61	0,59	0,63	0,66	0,7
Percobaan 5	0,63	0,65	0,68	0,7	0,69
Rata-rata	0,63	0,6	0,65	0,73	0,72

A5. Hasil uji coba berdasarkan jumlah *filter layer* ketiga

Loss	*Filter* 8	*Filter* 16	*Filter* 32	*Filter* 64	*Filter* 128
Percobaan 1	0,61	0,61	0,68	0,68	0,68
Percobaan 2	0,61	0,6	0,65	0,75	0,82
Percobaan 3	0,66	0,55	0,73	0,68	0,71

Percobaan 4	0,61	0,59	0,71	0,76	0,74
Percobaan 5	0,61	0,65	0,7	0,75	0,81
Rata-rata	0,62	0,6	0,69	0,72	0,75

A6. Hasil uji coba berdasarkan *learning rate* (η) *optimizer* Adam

Loss	η 0,1	η 0,01	η 0,001	η 0,0001	η 0,00001
Percobaan 1	0,69	0,69	0,61	0,62	0,69
Percobaan 2	0,69	0,6	0,6	0,6	0,69
Percobaan 3	0,69	0,61	0,55	0,61	0,69
Percobaan 4	0,69	0,68	0,59	0,6	0,69
Percobaan 5	0,69	0,64	0,65	0,61	0,68
Rata-rata	0,69	0,64	0,6	0,61	0,69

A7. Hasil uji coba berdasarkan jenis *optimizer*

Loss	Adam	SGD	Adagrad	Nadam	RMSprop
Percobaan 1	0,62	0,69	0,69	0,69	0,68
Percobaan 2	0,6	0,69	0,69	0,69	0,68
Percobaan 3	0,61	0,69	0,68	0,69	0,68
Percobaan 4	0,6	0,69	0,69	0,69	0,68
Percobaan 5	0,61	0,68	0,69	0,69	0,67

Rata-rata	0,61	0,69	0,69	0,69	0,68

A8. Hasil uji coba berdasarkan arsitektur CNN

Loss	Alex Net	ZF Net	VGG 16	VGG 19	LeNet-5	3 *Layers*
Percobaan 1	2,5	2,29	2,24	1,85	0,58	0,63
Percobaan 2	1,97	2,45	2,13	1,55	0,58	0,59
Percobaan 3	2,33	2,67	1,81	1,78	0,58	0,63
Percobaan 4	2,39	2,51	1,76	2,19	0,58	0,6
Percobaan 5	2,79	2,25	1,88	1,74	0,58	0,64
Rata-rata	2,4	2,43	1,96	1,82	0,58	0,62

LAMPIRAN B

B1. Hasil uji coba yang salah prediksi

No	*Tweet*	Aktual	Prediksi
1	[Idm] bolehkah aku numpang curhat, lelah banget, capek sama hidup sendiri @ BTS_twt pic.twitter.com/BgpJosHLLz	Depresi	Normal
2	Lelah sama hidup ini, ingin istirahat yg Tak terganggu oleh siapapun.	Depresi	Normal
3	Kadang kala aku jealous aku menyerah aku putus asa aku tak keruan Tapi aku kena tahu inilah kehidupan Ada atas ada bawah Setiap orang lain ceritanya Mcm mana nk improving myself ni? Aku betul2 down la... aduhaiiii Tak leh gak nk sentiasa positive	Depresi	Normal
4	Selamat tinggal Diah,selamat tinggal dunia,selamat tinggal semuanya ,aku harus pamit mau pulang,sampai ketemu di akhirat	Depresi	Normal
5	Dokter, sampai kapan pengobatan depresi saya berakhir? Saya capek Lama mba..	Depresi	Normal
6	Nah penyakitnya dateng lagi, putus asa, merasa sendiri, padahal tahu Allah bersamaku	Depresi	Normal

7	Evey single day of my life serasa ingin mengakhiri hidup. Overthinking makes everything even worse.	Depresi	Normal
8	Teman adalah orang yang menyelamatkanmu dari neraka yang bernama kesepian	Normal	Depresi
9	Dan sang adiksi ini sengaja saya munculkan dalam benak sebagai distraksi atas rasa depresi yang saya alami belakangan ini. Kasarnya, sang adiksi adalah obat penenang.	Depresi	Normal
10	Baru lepas reinstall netflix. Selamat tinggal dunia luar.	Normal	Depresi
11	jadi aku di sekolah itu jujur aja kesepian.. punya temen paling satu dua orang, jujur aja bosen ya.. aku ngeliat temen2 aku pada bikin projek semua, ya mau buka booth bazaar bareng lah, ada yang bikin clothing line bareng, ada yang bikin projek rescue anjing bareng, etc.	Depresi	Normal
12	Apa lebih baik saya mati? Kerja ndak dapat-dapat, stress berat, siklus haid berantakan sejak lulus kuliah. AKU JUGA BISA DEPRESI	Depresi	Normal
13	Jangan Pernah Patah Semangat Ya Sayang Della_JKT48	Normal	Depresi
14	Apa lebih baik saya mati? Kerja ndak dapat-dapat, stress berat, siklus haid	Depresi	Normal

	berantakan sejak lulus kuliah. AKU JUGA BISA DEPRESI		
15	Terima kasih semangatnya. Sangat membantu dan menenangkan saya yg sering depresi. Thanks so much @ BTS_AHC_IDN @ BTS_twt	Depresi	Normal
16	Kalian tau, gua ini gak suka banget sama org yang bilang begitu bercanda. Kalo gtu rasanya ingin beneran mati aja karena ya lu percuma gak di hargai kan di hidup ini ? :)	Depresi	Normal
17	Hm adanya pas TA di kampus. Disaat dosbing materi gua ngejatohin, bikin patah semangat, gak pede buat sidang dan optimis gak lulus, dosen teknis gua menyemangati habis-habisan!	Depresi	Normal
18	/tlw/ boleh minta semangat nya, lagi2 saya merasa saya bosan hidup dan saya merasa sendiri .	Depresi	Normal
19	Allah, aku benar-benar lelah untuk hidup. aku benar-benar payah, bukan? hem :'(	Depresi	Normal
20	Semangat terus ip! Ayok jan patah semangat yeh. GBU	Normal	Depresi
21	Dalam segala hal, berharap lebih baik daripada PUTUS ASA ..	Normal	Depresi
22	Ingin mengakhiri hidup tapi inget orang tua udah kerja keras	Depresi	Normal

23	Ini yg saya rasain dok, saya ingin mengakhiri hidup saya sebab saya hidup pun tidak ada gunanya untuk siapapun. Saya stress dok	Depresi	Normal
24	Ya allah..kasih petunjukmu biar aku bisa untuk nglupain dia untuk selamanya. Hilangkan dia dari pikiranku, hilangkan dia dari patah semangat hidupku ini. Jika aku untuk memilih..aku bersumpah aku ingin pulang secepatnya untuk di duniaini.	Depresi	Normal
25	Jangan mengakhiri ,hidup itu bukan penyesalan dikemudian hari.	Normal	Depresi
26	Cari duit cara nya jangan gitu2 amat.... Emang takut miskin...kalo tetap idealis...gak juga. Jangan jual harga diri . Hidup tidak berarti lagi.	Depresi	Normal
27	So on, kebablasan mikir kejauhan di masa depan. Jadinya panik sendiri. Jadinya jatuh sendiri perlahan terhisap dalam lumpur yang ga saya sadari. Alhasil depresi lagi karena mikir kejauhan. Frustasi. Ga bisa gerak. Aware tapi denial juga.	Depresi	Normal
28	Hidup harus punya tujuan, kalau tidak punya tujuan berarti tidak Hidup.	Normal	Depresi
29	The last. Saya tidak malu pernah mengalami depresi, pernah cutting, dan pernah	Depresi	Normal

	terbesit ingin bunuh diri. Saya malah bersyukur, karena ketika saya tersadar, saya jadi bisa lebih mensyukuri hidup. # HariPencegahanBunuhDiri		
30	Semua tempat riset menolak aku dan tim karena alasan surat kami blm disposisi. Udah putus asa :(. Tapi tau tau aja di X (salah satu kontraktor yang aku risetin) aku nekat aja langsung ke divisi yang perlu aku risetin dan cobain aja dulu	Depresi	Normal
31	Menyerah dan ingin mati berarti hidupmu sudah busuk.!! #bot	Normal	Depresi
32	Gue lelah deh rey di permainkan hidup ini	Normal	Depresi
33	Bosan Hidup, Ilmuwan Berusia 104 tahun Ini Ingin Disuntik Mati http://dlvr.it/QpkkjS	Normal	Depresi
34	Tanpa politik bosan juga hidup ye dok.	Normal	Depresi
35	Aku dah putus asa nak pertahankan benda yang belum tentu pun sebenarnya ikhlas dengan kau atau tak sebenarnya. Aku cuma ikut ke mana takdir bawak aku	Depresi	Normal
36	Ini yang bisa membuat gila.. Membuat tak memiliki semangat dalam berkarya.. Patah segala yang ada..	Depresi	Normal
37	Jangan patah semangat deh # BersamaPilih01	Depresi	Normal

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Faris Salbari, lahir di Jakarta pada 17 Februari 1996. Penulis menempuh Pendidikan mulai dari TK Aisyiyah 54 Rawamangun (1999-2001), SDN 05 Pagi Rawamangun (2001-2002), SD Islam Al-Azhar 13 Rawamangun (2002-2008), SMP Islam Al-Azhar 12 Rawamangun (2008-2011), SMAN 12 Jakarta (2011-2012), SMAN 53 Jakarta (2012-2014), dan sekarang sedang menjalani Pendidikan S1 Informatika ITS. Penulis pernah aktif dalam organisasi Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS, dan panitia Schematics 2015 dan 2016. Di antaranya adalah menjadi staff Departemen Dalam Negeri HMTC ITS (2014-2015), staff ahli Departemen Dalam Negeri HMTC ITS (2015-2016), dan staff REEVA Sechematics (2015-2016). Penulis mengambil bidang minat Komputasi Cerdas dan Visi (KCV). Untuk menghubungi penulis dapat melalui email farissalbari@gmail.com.

[Halaman ini sengaja dikosongkan]