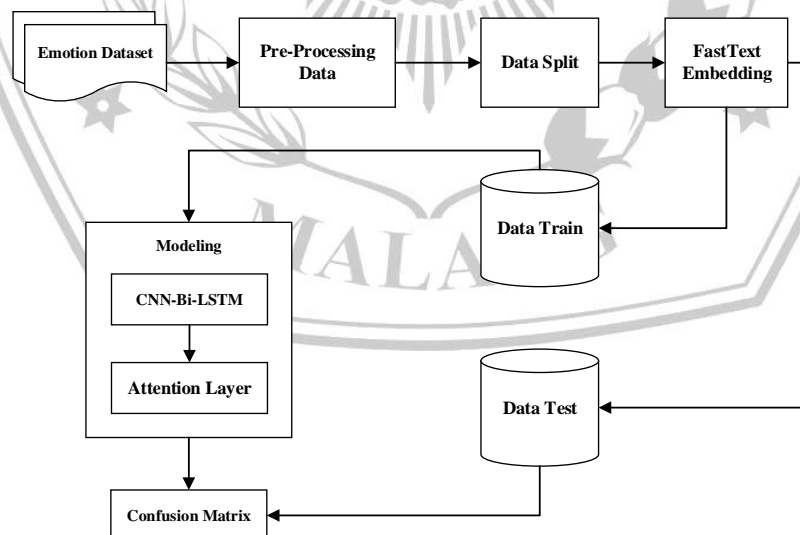


## BAB III

### METODOLOGI

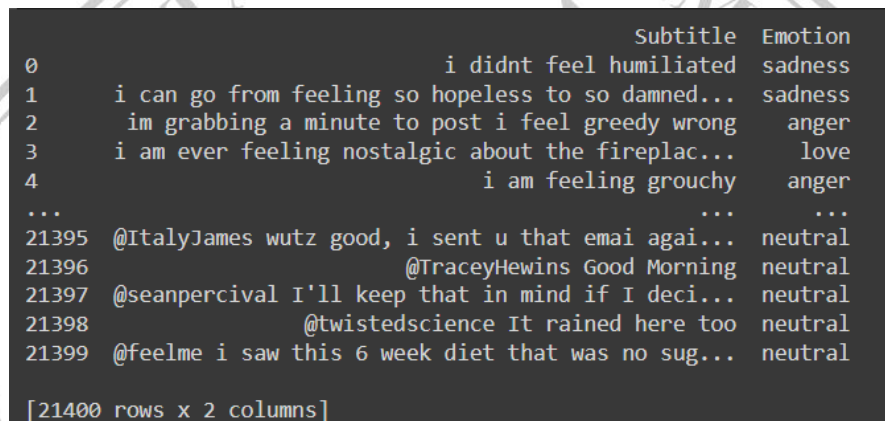
Penelitian ini mengusulkan metodologi yang terstruktur, dimulai dengan tahap pengumpulan dataset yang relevan. Selanjutnya, dataset tersebut menjalani proses pembersihan data untuk menghilangkan *noise* dan memastikan kualitasnya. Kemudian, dilakukan pembobotan kata menggunakan model *fasttext* untuk meningkatkan representasi teks. Selanjutnya, perancangan arsitektur model dilakukan dengan memanfaatkan pendekatan *attention* CNN-BiLSTM, yang menggabungkan keunggulan dari CNN untuk mengekstraksi fitur spasial dan LSTM untuk menangkap dependensi temporal dalam data teks. Selain itu, setelah model dilatih, tahap evaluasi dilakukan untuk mengukur kinerja model. Evaluasi dilakukan menggunakan metrik *precision*, *recall*, dan *F1-score* untuk memperoleh pemahaman yang komprehensif tentang kemampuan model dalam mengklasifikasikan data. Metrik-metrik ini memberikan *insight* tentang seberapa baik model dapat mengidentifikasi kelas-kelas yang relevan dan mengukur *trade-off* antara *presisi* dan *recall*. Dengan demikian, hasil evaluasi ini memberikan pemahaman yang mendalam tentang keefektifan model dalam memecahkan masalah yang diteliti.



Gambar 3.1 Metodologi yang diusulkan.

### 3.1 Dataset Emosi

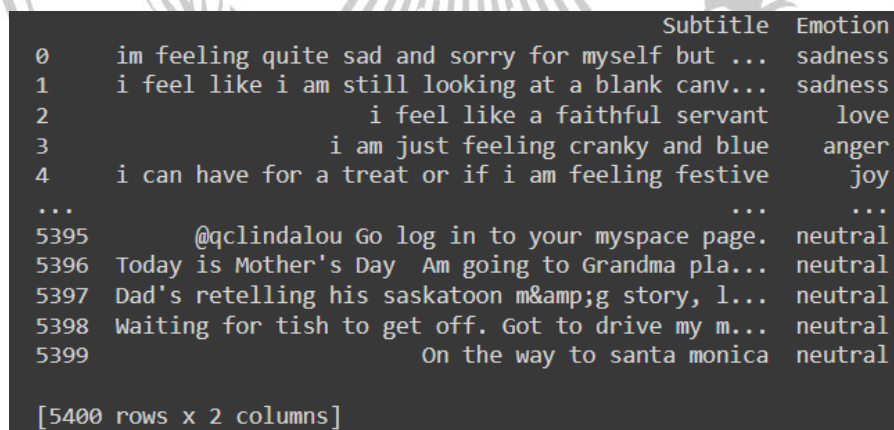
Dataset emosi yang digunakan dalam penelitian ini diambil dari Kaggle dan terdiri dari dua bagian utama: data pelatihan (*train*) gambar 3.1 dan data uji (*test*) gambar 3.2. Dataset ini merupakan gabungan dari dua sumber dataset yang berbeda. Proses awal pada kedua dataset ini melibatkan pemeriksaan untuk mengidentifikasi nilai *null* atau kolom kosong. Data yang mengandung nilai *null* atau kolom kosong dihapus untuk memastikan bahwa setiap entri dalam dataset memiliki teks dan label yang sesuai dalam kolomnya. Jumlah entri dalam dataset pelatihan adalah sebanyak 21.367, sedangkan dalam dataset uji terdapat sebanyak 5.392 entri. Hal ini memungkinkan untuk pengembangan model yang tepat serta evaluasi yang representatif terhadap kinerja model dalam mengklasifikasikan emosi dari teks.



	Subtitle	Emotion
0	i didnt feel humiliated	sadness
1	i can go from feeling so hopeless to so damned...	sadness
2	im grabbing a minute to post i feel greedy wrong	anger
3	i am ever feeling nostalgic about the fireplac...	love
4	i am feeling grouchy	anger
...	...	...
21395	@ItalyJames wutz good, i sent u that emai agai...	neutral
21396	@TraceyHewins Good Morning	neutral
21397	@seanpercival I'll keep that in mind if I deci...	neutral
21398	@twistedscience It rained here too	neutral
21399	@feelme i saw this 6 week diet that was no sug...	neutral

[21400 rows x 2 columns]

Gambar 3.2 *Train* dataset emosi.



	Subtitle	Emotion
0	im feeling quite sad and sorry for myself but ...	sadness
1	i feel like i am still looking at a blank canv...	sadness
2	i feel like a faithful servant	love
3	i am just feeling cranky and blue	anger
4	i can have for a treat or if i am feeling festive	joy
...	...	...
5395	@qclindalou Go log in to your myspace page.	neutral
5396	Today is Mother's Day Am going to Grandma pla...	neutral
5397	Dad's retelling his saskatoon m&g story, l...	neutral
5398	Waiting for tish to get off. Got to drive my m...	neutral
5399	On the way to santa monica	neutral

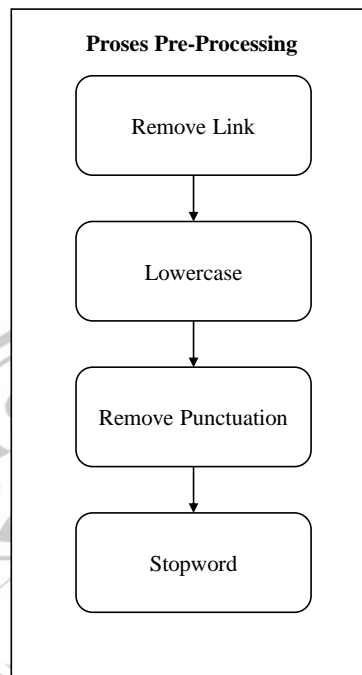
[5400 rows x 2 columns]

Gambar 3.3 *Test* dataset emosi.

### 3.2 Pre-Processing

Teknik yang dilakukan pada tahapan *pre-processing* dengan menggunakan dataset emosi, beberapa tahapan yang dilakukan untuk membersihkan dataset emosi yaitu : *remove link*, *lowercase*, *remove punctuation*, dan *stopword*. Masing-masing

tahapan ini diharapkan dapat membuat dataset yang akan dilakukan proses pemodelan menjadi lebih bersih dari redundansi.



Gambar 3.4 Proses *pre-processing*.

### 3.2.1 *Remove Link*

Menghapus URL teks (*Uniform Resource Locator*) adalah langkah pertama dalam proses pengolahan data. Setiap entri data yang berisi kata "*http://*" dianggap memiliki URL yang tertaut. Tindakan ini diambil untuk memastikan keaslian teks dan mengurangi gangguan yang disebabkan oleh URL yang digunakan dalam analisis. Penghapusan URL ini memungkinkan penekanan yang lebih besar pada teks asli, yang menghasilkan kualitas data yang lebih baik untuk digunakan dalam analisis emosi [37].

```
def remove_links(text):  
    # Menghapus semua URL yang dimulai dengan http atau  
    https  
    text = re.sub(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-  
_@.&+]|!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', '', text)  
    return text
```

### 3.2.2 *Lowercase*

Pada langkah *lowercase folding*, dilakukan penyesuaian seluruh huruf dalam teks menjadi huruf kecil [38] . Tujuannya adalah untuk memastikan

keseragaman dalam penggunaan kata-kata dalam kumpulan data. Dengan menerapkan *lowercase folding*, setiap kata yang sebenarnya memiliki makna yang sama tetapi ditulis dengan huruf besar atau kecil akan dianggap sebagai kata yang sama. Misalnya, perbedaan antara "Data" dan "data" akan dihilangkan. Langkah ini dimaksudkan untuk menyederhanakan proses pemrosesan teks dan menghindari potensi kebingungan dalam analisis. Dengan demikian, data yang telah mengalami proses *lowercase folding* akan memiliki representasi yang seragam dan siap untuk digunakan dalam analisis emosi [39].

```
def lower_text(text):  
    return text.lower()
```

### 3.2.3 Remove Punctuation

Penghapusan tanda baca adalah tahapan yang signifikan dalam rangkaian pra-pemrosesan teks yang ditujukan untuk menghilangkan karakter khusus, termasuk namun tidak terbatas pada tanda seru, tanda tanya, koma, dan sejenisnya. Proses ini penting karena membantu dalam membersihkan teks dari elemen-elemen tambahan yang tidak memberikan kontribusi substansial terhadap pemahaman atau analisis teks. Dengan mengeliminasi tanda baca, teks yang dihasilkan menjadi lebih terfokus pada isi makna kata-kata itu sendiri, yang pada gilirannya memfasilitasi proses analisis lebih lanjut dalam berbagai konteks pemrosesan bahasa alami [40].

```
def remove_punct(text):  
    punctuations = '@#!?+&*[]-%.:/();$=><|}{^' + "'`"  
    for p in punctuations:  
        text = text.replace(p, f' {p} ')  
    return text
```

### 3.2.4 Stopwords

Untuk menjamin akurasi dan efisiensi dalam berbagai tugas NLP seperti pengindeksan, pemodelan topik, klasifikasi teks, dan pengambilan informasi, langkah pra-pemrosesan perlu dilakukan untuk menghilangkan kata-kata yang kurang informatif, yang sering disebut sebagai "*stopwords*". *Stopwords* ini sering kali muncul dalam berbagai dokumen dalam bahasa alami atau bahkan di bagian-bagian tertentu dari teks dalam sebuah dokumen, namun memberikan sedikit informasi tentang konteks teks yang mereka miliki. Dengan menghapus *stopwords* ini, diharapkan dapat meningkatkan rasio sinyal-ke-*noise* dalam teks yang tidak terstruktur, dan dengan demikian meningkatkan signifikansi statistik dari istilah-istilah yang mungkin penting untuk suatu tugas analisis tertentu. Contoh *stopwords* yang umum mencakup kata-kata seperti "*each*", "*about*", "*such*", dan "*the*" [41].

```
import nltk
from nltk.corpus import stopwords
def remove_stopwords(text):
    text = ' '.join([word for word in text.split() if word
not in (stopwords)])
    return text
```

### 3.3 Tokenization

Metode yang digunakan untuk mengonversi kata-kata menjadi token dalam penelitian ini melibatkan penerapan fungsi khusus yang disediakan oleh *Natural Language Toolkit* (NLTK), yang dikenal sebagai *word\_tokenize*. Melalui langkah ini, teks yang kompleks dapat diproses dengan efisien, memungkinkan pemisahan setiap kata menjadi token-token individual. Penggunaan *word\_tokenize* memainkan peran krusial dalam mempersiapkan teks untuk analisis lebih lanjut, karena memungkinkan penanganan yang akurat terhadap struktur dan makna dari teks yang sedang dipelajari. Dengan demikian, penggunaan metode ini memberikan landasan yang kokoh untuk analisis mendalam dalam konteks penelitian mengenai emosi dalam teks [42].

```

from keras.utils import to_categorical
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer()
tokenizer.fit_on_texts(x_train)

X_train = tokenizer.texts_to_sequences(x_train)
X_test = tokenizer.texts_to_sequences(x_test)
TEST = tokenizer.texts_to_sequences(test_df["clean_text"])

vocab_size = len(tokenizer.word_index) + 1

```

### 3.4 Pembobotan *Fasttext*

Pembobotan *fasttext*, yang dikembangkan oleh penelitian facebook, adalah *application programming interface* (API) sumber terbuka yang terkenal karena pembelajarannya yang cepat dan efisien terhadap representasi kata [43]. Model ini menelusuri struktur dalam kata, terutama bermanfaat untuk bahasa yang kaya secara morfologis, dan secara mandiri mempelajari variasi morfologis dari kata-kata dalam data teks [44]. Fitur-fitur dari model *fasttext* sangat bermanfaat untuk teks yang sangat beragam, meningkatkan kemampuan pembelajaran. Mengikuti model *skip-gram*, pembobotan *fasttext* memperluas fungsionalitas model *Word2Vec*. Yang mencolok, model ini dapat menganalisis teks mentah dari pos media sosial. Representasi kata dalam model *fasttext* adalah kombinasi dari *char-n grams*, memanfaatkan informasi urutan kata untuk efisiensi komputasi. Pembelajaran dalam model *fasttext* melibatkan jendela konteks yang luas untuk kata-kata konteks kanan dan kiri, dan sangat baik dalam menangani kata-kata yang hilang, jarang, atau salah eja. Oleh karena itu, pembobotan *fasttext* berfungsi sebagai langkah prapemrosesan penting untuk konten terkait bunuh diri dan depresi sebelum menjalani klasifikasi.

```

def get_embedding_vectors(tokenizer, dim=300):
    embedding_index = {}
    with open('/content/drive/My
Drive/mental/wiki.en.vec','r') as f:
        for line in tqdm.tqdm(f, "Reading FastText"):
            values = line.split()
            word = ''.join(values[:-300])
            vectors = np.asarray(values[-300:],
dtype='float32')
            embedding_index[word] = vectors

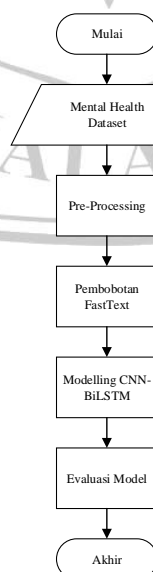
    word_index = tokenizer.word_index
    embedding_matrix = np.zeros((len(word_index)+1, dim))
    for word, i in word_index.items():
        embedding_vector = embedding_index.get(word)
        if embedding_vector is not None:
            # words not found will be 0s
            embedding_matrix[i] = embedding_vector

    return embedding_matrix
embedding_matrix = get_embedding_vectors(tokenizer)

```

### 3.5 Arsitektur Model

Proses pemodelan melibatkan penyesuaian *hyperparameter* model untuk mengoptimalkan kinerja. Setelah itu, model dievaluasi menggunakan metrik yang relevan, seperti akurasi, presisi, *recall*, dan *F1-score*, setelah diuji pada dataset emosi yang telah disiapkan. Alur dari seluruh proses pemodelan dapat dilihat pada *flowchart* yang tersedia pada gambar 3.3.



Gambar 3.5 Arsitektur CNN-BiLSTM-attention.

Penelitian ini menggunakan model *neural network* kompleks yang menggabungkan berbagai jenis lapisan untuk mengolah data sekuensial. Model ini dapat menerima 34 fitur dengan panjang tetap dan menghasilkan output yang memasukkan klasifikasi ke dalam tujuh kelas. Alur model ini dijelaskan secara rinci di sini.

Model dimulai dengan lapisan input data berdimensi (*None*, 34), di mana "*None*" menunjukkan bahwa ukuran *batch* dapat berbeda. Lapisan ini berfungsi sebagai *placeholder* untuk data yang dimasukkan sebelum diproses lebih lanjut. Data kemudian dimasukkan ke dalam lapisan *embedding*, yang mengubah input dimensi (34) menjadi representasi vektor dimensi (34, 300). Lapisan *embedding* sangat penting untuk menampilkan data sekuensial dalam bentuk vektor berdimensi yang lebih kaya informasi.

Selanjutnya, model menggunakan lapisan *dropout* awal dengan tingkat *dropout* sebesar 0,5. Ini menunjukkan bahwa selama pelatihan, setengah dari unit lapisan ini akan dinonaktifkan secara acak. Ini dilakukan untuk menghindari *overfitting*. Dimensi *output* lapisan *dropout* ini tidak berubah (*None*, 34, 300).

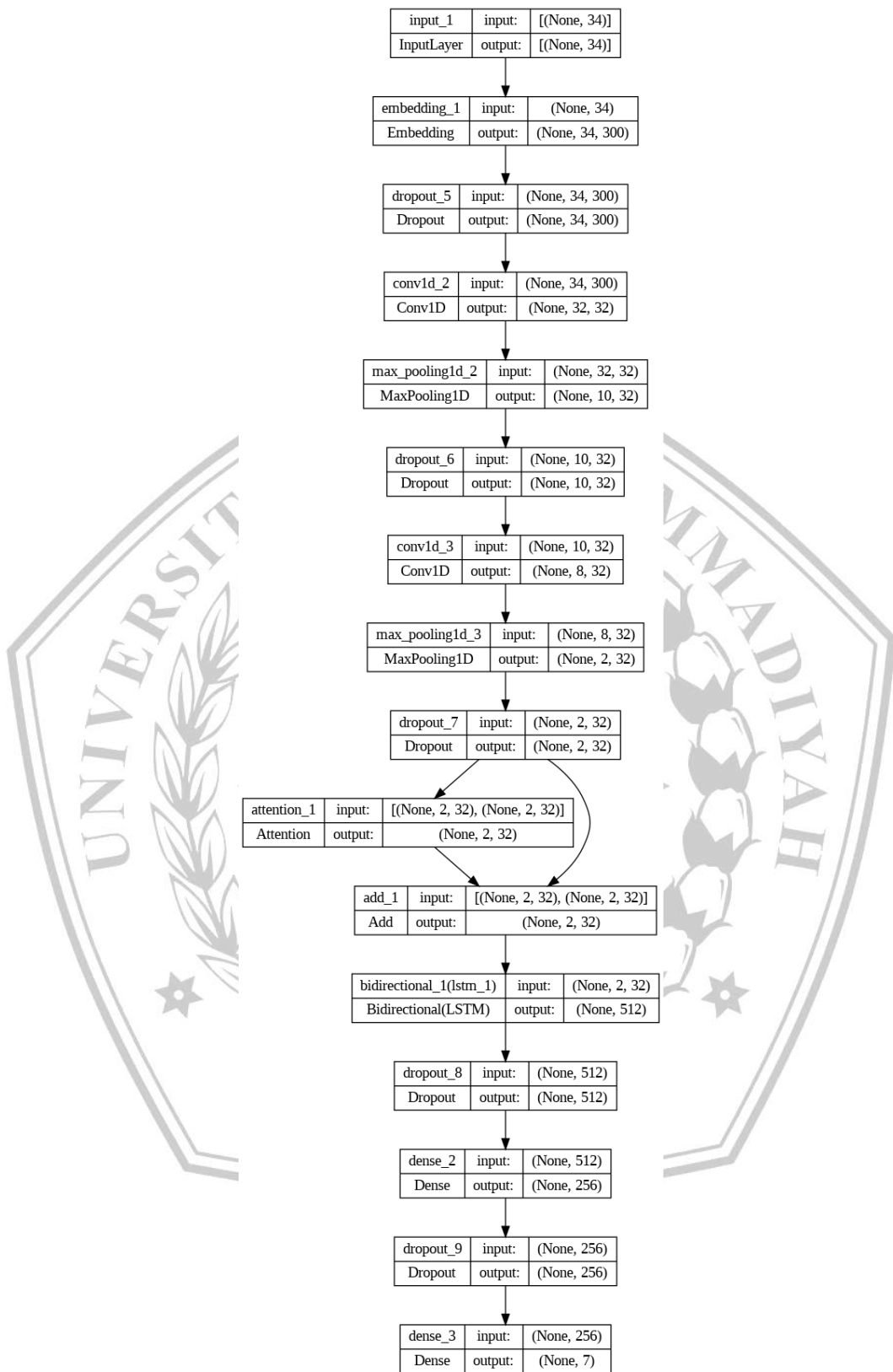
Lapisan konvolusi dan *pooling* adalah tahap berikutnya. Lapisan konvolusi pertama menggunakan 32 filter dengan ukuran kernel 3, yang menghasilkan *output* berdimensi (*None*, 32, 32). Fungsi lapisan konvolusi ini adalah untuk mengekstrak fitur lokal dari data sekuensial. *Output* dari lapisan konvolusi ini kemudian diproses oleh lapisan pemotongan maksimum pertama, yang menggunakan ukuran *pool* 3 untuk mengurangi dimensi temporal data menjadi (*None*, 10, 32). Ini membantu mengurangi kompleksitas model dan mendapatkan fitur yang lebih banyak.

Kemudian, lapisan *dropout* kedua menangani output dari lapisan *max pooling* pertama dengan tingkat *dropout* 0,5, yang menghasilkan *output* dengan dimensi yang sama (*None*, 10, 32). Lapisan konvolusi kedua kemudian digunakan dengan filter 32 dan kernel *size* 3, yang menghasilkan *output* dengan dimensi (*None*, 8, 32). Ini membantu dalam mendalami ekstraksi fitur dari data. Lapisan pemotongan maksimum kedua kemudian menangani *output* ini, yang mengurangi dimensi temporal data menjadi (*None*, 2, 32) dengan ukuran *pool* 3. Selanjutnya, lapisan pemotongan ketiga dengan tingkat pemotongan 0,5 digunakan untuk mencegah penyesuaian yang berlebihan, sehingga dimensi *output* tetap pada (*None*,



2, 32). Kemudian, model memasukkan lapisan perhatian yang menerima input dengan dimensi yang sama (*None, 2, 32*) dan menghasilkan *output* dengan dimensi yang sama. Fungsi lapisan perhatian ini adalah untuk membantu model fokus pada bagian penting dari data sekuensial sehingga dapat menangkap informasi yang lebih relevan. Setelah itu, *output* dari lapisan perhatian digabungkan dengan *output dropout* sebelumnya menggunakan lapisan tambahan, sehingga dimensi *output* tetap sama (*None, 2, 32*). Lapisan LSTM *bidirectional*, yang memiliki 256 unit di setiap arah, kemudian memproses data, menghasilkan *output* dengan dimensi (*None, 512*). Lapisan ini memungkinkan model untuk menangkap informasi dari kedua arah sekuensial (maju dan mundur), yang meningkatkan pemahaman konteks data. Lapisan *dropout* keempat, yang memiliki tingkat *dropout* 0,5, kemudian memproses *output* dari lapisan LSTM *bidirectional*, yang tetap pada *dime*. Untuk menghasilkan probabilitas kelas dengan dimensi *output*, lapisan ini menggunakan fungsi aktivasi *softmax* (*None, 7*).

Arsitektur model ini secara keseluruhan menggabungkan berbagai teknik *deep learning*, termasuk LSTM *bidirectional*, *pooling*, konvolusi, dan perhatian, untuk memaksimalkan kemampuan model dalam menangkap fitur penting dari data sekuensial. Diharapkan model dapat menghasilkan prediksi yang akurat tentang tugas klasifikasi yang diberikan karena penggunaan *dropout* pada beberapa lapisan bertujuan untuk mencegah *overfitting*. Gambar 3.4 adalah arsitektur model yang digunakan pada penelitian yaitu model CNN-BiLSTM *attention*



Gambar 3.6 Arsitektur model CNN-BiLSTM-attention.