Klasifikasi *Multi-label* pada Hadis Sahih Bukhari Terjemahan Bahasa Indonesia Menggunakan *Convolutional Neural Networks*

Muzakki Ahmad Al Farisi[1], Widi Astuti[2], Adiwijaya[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]muzakkialfarisi@student.telkomuniversity.ac.id, [2]widiwdu@telkomuniversity.ac.id,
[3]adiwijaya@telkomuniversity.ac.id

Abstrak

Hadis merupakan pedoman hidup umat muslim, sumber hukum Islam kedua setelah Al- Quran sehingga menjadikannya penting dipelajari. Hadis baik dipelajari jika hadis itu sahih, salah satu hadis sahih yaitu yang diriwayatkan oleh al-Bukhari. Namun terdapat kesulitan dalam mempelajarinya yaitu untuk menentukan hadis mana saja yang tergolong ke dalam topik anjuran, larangan, dan informasi. Oleh karena itu perlu dilakukan klasifikasi *multi- label* pada hadis untuk menggolongkannya ke dalam salah satu atau lebih topik. Algoritma klasifikasi yang digunakan adalah CNN, karena algoritma tersebut tergolong ke dalam *deep learning* dan proses perhitungan CNN dilakukan secara paralel. Pada penelitian ini menghasilkan performansi CNN tanpa *padding* dan *strides* 1 menggunakan skenario *preprocessing* tanpa *stemming* dengan nilai *hamming loss* 0,0693 dan waktu eksekusi pemodelan 67,7613 detik dibandingkan dengan LSTM yang memiliki nilai *hamming loss* 0,1128 dan waktu eksekusi pemodelan 1006,6985 detik serta RNN memiliki nilai *hamming loss* 0,1145 dan waktu eksekusi pemodelan 262,8086 detik.

Kata Kunci: CNN, *preprocessing, stemming, hamming loss*

Abstract

Hadith is the life guidance of the Muslims, second only to the authority of the Qur'an, which makes it necessary to learn. It is good to learn hadith when it is sahih (authentic), one of which is narrated by al-Bukhari. However, there are difficulties found while learning as determining which one is included in the topic of advice, prohibition, and information. Therefore, it is necessary to carry out the multi-label classification to categorize the hadith into topics. Here, researchers used the Convolutional Neural Network (CNN) algorithm. CNN was chosen because it is included in deep learning and its calculation was done in parallel. The result showed that CNN performance without padding and 1 strides using the preprocessing scenario without stemming, obtained the hamming loss value of 0,0693 with model execution time of 67,7613 seconds. It is better than LSTM and RNN which obtained the hamming loss value of 0,1128 and 0,1145 with model execution time of 1006,6985 and 262,8086 seconds.

Keywords: CNN, preprocessing, stemming, hamming loss

1. Pendahuluan

Latar Belakang

Al-qur'an memerintahkan untuk taat kepada Allah dan Rasulullah Muhammad SAW [1]. Mentaati Allah diinterpretasikan mematuhi perintah dan larangan-Nya, sedangkan taat kepada Rasulullah Muhammad SAW yaitu mematuhi sunnah dan hadisnya [2]. Hadis adalah suatu perkataan, perbuatan, dan persetujuan yang datang dari Muhammad SAW [3]. Hal tersebutlah yang menjadikan hadis sebagai pedoman hidup atau sumber hukum kedua umat Islam setelah al-qur'an. Terdapat kumpulan-kumpulan hadis yang dianggap sahih untuk dijadikan salah satu sumber hukum Islam oleh umat muslim, salah satunya adalah hadis Rasulullah SAW yang periwayatnya bernama Imam Bukhari [4].

Klasifikasi *multi-label* pada hadis merupakan pengkategorian hadis ke dalam beberapa kelas, dimana tiap hadis dapat dikategorikan ke dalam dua kelas sekaligus atau lebih, sesuai dengan kata yang terkandung dalam hadis tersebut. Hal ini dapat mempermudah dan mempersingkat waktu manusia untuk mengetahui dan mempelajari bahwa hadis tersebut tergolong dalam suatu kelompok atau lebih. Terdapat beberapa penelitian terdahulu mengenai klasifikasi *multi-label* dengan performansi yang baik. Penelitian [5] melakukan klasifikasi menggunakan *Recurrent Neural Network* menghasilkan *hamming loss* sebesar 0.0597. Sedangkan penelitian [6] melakukan klasifikasi menggunakan *Convolutional Neural Networks* (CNN) menghasilkan *hamming loss* sebesar 0.0013.

Pada penelitian [5] dengan dataset hadis namun tidak menggunakan algoritma CNN, sedangkan pada penelitian [6] menggunakan algoritma CNN namun dataset yang digunakan adalah al-qur'an. Algoritma CNN merupakan salah satu metode yang selalu dipilih untuk menjawab masalah-masalah dalam *computer vision* [7].

Selain itu, algoritma CNN juga diterapkan pada klasifikasi teks dapat menghasilkan performansi yang lebih baik [6]. Pada penelitian ini penulis menggunakan algoritma CNN untuk mengatasi masalah dalam klasifikasi teks dengan dataset yang digunakan yaitu hadis shahih bukhari Bahasa Indonesia dan memodifikasi tahap *preprocessing* terutama pada *stemming* serta penggunaan *padding* dan *strides* pada *convolutional layer.* Diharapkan pada penelitian kali ini dapat menghasilkan performansi yang lebih baik.

Topik dan Batasannya

Ada beberapa batasan yang ditentukan diantaranya adalah penggunaan dataset hadis terjemahan Bahasa Indonesia dengan data *multi-label* yang memiliki tiga jenis label yaitu anjuran, larangan, dan informasi yang diperoleh dari penelitian Mediamer G., Adiwijaya and S. Al Faraby [11]. Pengukuran performansi dari sistem klasifikasi yang dibangun dengan kasus *multi-label* menggunakan matriks evaluasi *hamming loss*.

Tujuan

Tujuan dari penelitian ini yaitu untuk mengetahui perbedaan nilai *hamming loss* dan waktu eksekusi terhadap performansi dari model CNN dengan model lain yaitu LSTM dan RNN dalam mengklasifikasikan hadis *multi-label* terjemahan Bahasa Indonesia. Selain itu juga akan dilakukan analisis terhadap hasil performansi yang diperoleh dari model yang telah dibangun.

Organisasi Tulisan

Pada bagian ini menjelaskan mengenai organisasi pada penelitian yaitu sebagai berikut. Poin pendahuluan menjelaskan latar belakang, batasan, tujuan dan organisasi dari jurnal ini. Kemudian poin studi literatur terkait dengan penelitian yang sudah dilakukan sebelumnya dan beberapa tinjauan pustaka yang terkait dengan penelitian. Kemudian poin selanjutnya yaitu sistem yang akan dibangun menjelaskan klasifikasi pada dataset dengan menggunakan metode CNN. Kemudian poin evaluasi, menjelaskan analisis dari hasil pengujian yang telah dilakukan. Terakhir poin kesimpulan berisi kesimpulan dan saran dari penelitian yang telah dilakukan.

2. Studi Terkait

Penelitian mengenai klasifikasi *multi-label* sudah dilakukan dalam berbagai macam penelitian dengan topik dan model yang beragam, seperti pada topik jenis penyakit, ayat al-qur'an, *review film*, produk kecantikan dan yang lainnya. Pada penelitian dengan topik hadis juga sudah pernah dilakukan sebelumnya, seperti pada penelitian [5] oleh Raden Rizky Falih Pridyandhika menggunakan *Recurrent Neural Networks* dengan *Sort Long-Term Memory* dengan memodifikasi tahap *preprocessing* nya yaitu melewati tahap *case folding* dan *remove punctuation* menghasilkan nilai *hamming loss* terbaik 0,0597. Penelitian kedua [8] oleh Hendro Prasetyo menggunakan *Mutual Information and Backpropagation Neural Networks* dengan *preprocessing* tanpa *stemming* dan tanpa menerapkan *mutual information* serta menerapkan *learning rate* sebesar 0,02 menghasilkan nilai *hamming loss* terbaik 0,0892. Pada penelitian ketiga [9] oleh Adhitia Wiraguna menggunakan *Random Forest* menghasilkan nilai *hamming loss* terbaik 0,0663. Pada penelitian keempat [10] oleh Amran menggunakan *Multinomial Naïve Bayes* menghasilkan nilai *hamming loss* terbaik 0,1222. Pada penelitian kelima [11] oleh Gugun Mediamer Menggunakan *Support Vector Machine* menghasilkan nilai *hamming loss* terbaik 0,0623.

Kemudian ada juga penelitian [6] oleh Sarah Fauziah Lestari terkait klasifikasi *multi-label* namun pada teks terjemah al-qur'an bahasa inggris menggunakan *Convolutional Neural Networks* (CNN) yang termasuk ke dalam *deep learning algorithm* dengan menggunakan *pretrained word embedding* GloVe, *dropout layer*, dan 512 filter menghasilkan *hamming loss* sebesar 0,0963. Dapat disimpulkan dari penelitian sebelumnya yang telah disebutkan bahwa performansi yang didapat *neural networks* terbukti baik digunakan dalam mengklasifikasikan teks dengan mengexplore eksperimen untuk meningkatkan performansi dari model yang dibuat.

2.1. *Preprocessing*

Preprocessing adalah bagian penting dari pemrosesan bahasa. Tujuan dari Pre-processing adalah untuk menghilangkan *noise* yang dimiliki oleh dataset, karena jenis kata atau karakter yang diidentifikasi pada tahap ini akan berpengaruh untuk proses selanjutnya. Dalam kasus klasifikasi teks, pada umumnya ada berbagai macam proses yang dilakukan untuk *preprocessin*g diantaranya *remove punctuation, case folding*, *stemming, tokenization* dan sebagainya [12]. Proses *remove punctuation* dan *case folding* perlu dilakukan pada *preprocessing* dataset karena pada penelitian [5] menghasilkan performansi yang baik dari pada tidak menggunakannya. Pada penelitian ini, *preprocessing* dilakukan dalam kombinasi beberapa tahap yang akan dijelaskan pada bab tiga.

2.2. *Library Python* Sastrawi

Library Python Sastrawi umum dipakai untuk melakukan proses *stemming* Bahasa Indonesia. Berawal dari *Library* PHP Sastrawi kemudian dikembangkan menjadi *Library* Python Sastrawi, *library* sederhana berbahasa PHP yang dibuat agar berkualitas tinggi dan mudah didokumentasikan. *Library Python*

Sastrawi pada dasarnya menerapkan algoritma Nazief Adriani, setelah itu dikembangkan oleh Algoritma *Confix Stripping*, lalu dikembangkan lagi oleh algoritma ECS (*Enhanced Confix Stripping*), lalu dikembangkan lagi oleh *Modified* ECS [13]. *Library* Python Sastrawi sudah mengalami perbaikan dan merupakan *stemming* Bahasa Indonesia terbaru saat ini. Oleh karena itu, library ini layak digunakan untuk tahap *preprocessing* khususnya *stemming*.

2.3. *Convolutional Neural Network*s

Convolutional Neural Network (CNN) termasuk jenis *neural network* yang pada umumnya digunakan pada data berbasis *image*, namun secara garis besar dalam prosesnya tidak jauh berbeda dengan *neural network* lainya. Pada saat ini penelitian mengenai CNN diterapkan juga pada *Natural Language Processing* (NLP), kata atau kalimat yang berbentuk matriks dijadikan sebagai masukan untuk proses didalam pengolahannya [14]. Pada umumnya CNN terdapat dua layer utama yaitu *Feature Extraction Layer* berguna menghubungkan dan mengubah suatu masukan menjadi *features* berdasarkan ciri dari masukan, *classifier layer* berfungsi mengelompokan tiap neuron yang telah diekstraksi fitur pada sebelumnya. Kedua layer tersebut terdiri dari beberapa *layer* [6] yaitu:

1. *Convolution Layer*
   Dalam kasus NLP *Convolution Layer* bertujuan untuk mempelajari representasi fitur dari *layer* sebelumnya atau *embedding layer*. *Convolution layer* memiliki beberapa peta fitur dimana tiap peta fitur yang sama digunakan untuk mengekstraksi karakteristik lokal dari posisi yang berbeda di layer sebelumnya [15]. Fungsi aktivasi diperlukan pada layer ini untuk mendapatkan fitur baru, salah satunya adalah *Rectified Linear Unit* (ReLU). Adapun persamaan rumus ReLU [14] yaitu:
   $$\boldsymbol{f(x) = max(x, 0)} \quad \textbf{(1)}$$

2. *Pooling Layer*
   Pooling layer adalah lapisan yang berfungsi untuk mempercepat komputasi. Prosesnya yaitu mengurangi dimensi dari *feature map* atau mengurangi parameter yang harus di-*update* dan biasa disebut proses *downsampling*. Salah satu *pooling* yang dapat digunakan adalah *globalmaxpooling* dimana tiap ada pergeseran *filter* dapat menentukan nilai maksimumnya.

3. *Dropout Layer*
   Dropout layer berfungsi untuk mencegah *overfitting* dengan cara menge-*set* masukan menjadi 0 secara acak pada tiap pembaruan selama waktu *training* [14].

4. *Fully Connected Layer*
   Sesuai dengan namanya *fully connected layer* berfungsi untuk menghubungkan tiap neuron ke semua fitur. Hasil dari *layer* sebelumnya yang berupa fitur hasil ekstraksi data digunakan pada *layer* ini untuk memindahkan data masukan ke dalam kelas-kelas yang ada pada dataset [16]. *Fully connected layer* memerlukan fungsi aktivasi untuk proses klasifikasi dengan menggunakan regresi, aktivasi tersebut adalah *sigmoid* yang memiliki nilai probabilitas dengan skala 0 sampai dengan 1. Adapun persamaan rumus *sigmoid* yaitu:
   $$\boldsymbol{sigmoid(x) = \frac{1}{1 + e^{-x}}} \quad \textbf{(2)}$$

2.4. *Hamming L*oss

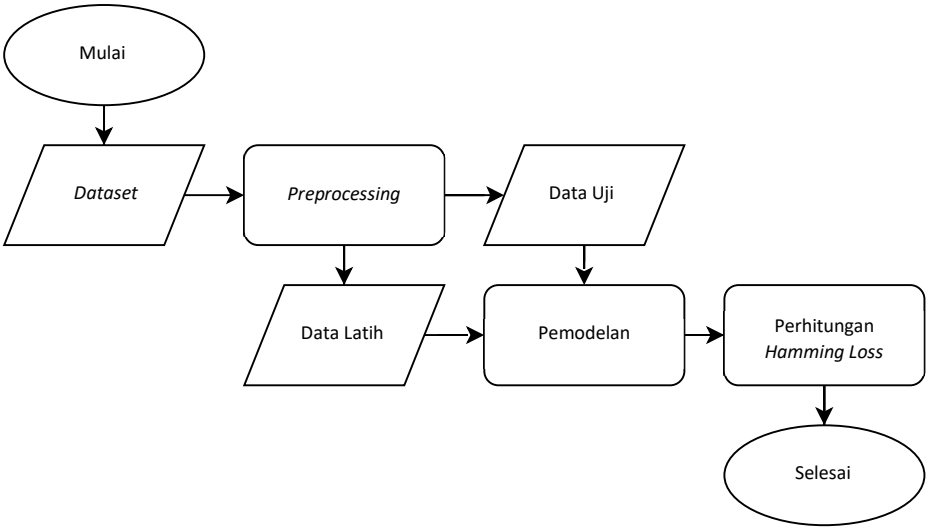
Hamming loss merupakan bagian label yang salah diprediksi [17]. Secara umum *hamming los*s digunakan untuk kasus *multi-class* dimana tugasnya yaitu mengkalkulasi seberapa besar *misclassification* yang ada, berbeda dengan *precision, recall,* dan *f1-measure* yang dirancang untuk kelas biner. Idealnya nilai *hamming loss* pada sistem adalah nol, yang berarti semakin kecil atau mendekati nol nilai maka semakin bagus performansi dari model yang dibangun. Adapun persamaan rumus *hamming loss* [18] yaitu:

$$\boldsymbol{Hamming\ Loss = \frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L} \mathbf{I}\left[\hat{y}_j^{(i)} \neq y_j^{(i)}\right]} \quad \textbf{(3)}$$

N adalah jumlah data, *L* adalah panjang data keluaran *multi-label*, $\hat{y}_j^{(i)}$ target multi-label, dan $y_j^{(i)}$ prediksi *multi-label* hasil dari klasifikasi.

3. Sistem yang Dibangun

Sistem yang dibangun pada penelitian ini bertujuan untuk mengklasifikasi *multi-label* terhadap dataset hadis sahih bukhari Bahasa Indonesia menggunakan *Convolutional Neural Networks* (CNN). Adapun tahapan yang dilakukan untuk membangun sistem yang meliputi *preprocessing* dilanjutkan proses klasifikasi menggunakan algoritma CNN, dan perhitungan *hamming loss*. Berikut Gambar 1 merupakan gambaran umum sistem yang dibangun.



Gambar 1. Diagram Alur Sistem.

3.1. Dataset

Penggunaan dataset pada penelitian ini adalah data hadis shahih bukhari Bahasa Indonesia yang didapat dari penelitian [11]. Data tersebut sudah memiliki label yang bersifat *multi-label*, dimana terdiri dari tiga kelas yaitu anjuran, larangan dan informasi. Panjang kata tiap hadis-pun bervariasi mulai dari 4 kata sampai dengan 732 kata. Hadis dengan kata terpanjang akan menjadi nilai *max sequence length* untuk parameter *embedding layer*. Representasi data akan ditampilkan seperti pada Tabel 1.

Tabel 1. Representasi Data Multi-label.

Data	Anjuran	Larangan	Informasi
Kamu memberi makan, mengucapkan salam kepada orang yang kamu kenal dan yang tidak kamu kenal.	1	0	1
Iman memiliki lebih dari enam puluh cabang, dan malu adalah bagian dari iman.	0	0	1
Janganlah kamu pergi hingga engkau mendengar suara atau mencium bau.	0	1	1

3.2. *Preprocessing*

Setelah dataset tersedia, tahap selanjutnya adalah normalisasi data dengan membersihkan data dari noise yaitu melakukan *preprocessing*. Berikut adalah macam *preprocessing* yang digunakan pada penelitian ini:

1. *Remove Punctuation*

   Remove punctuation berfungsi untuk menghilangkan simbol-simbol dan tanda baca yang tidak diperlukan pada proses klasifikasi. Tabel 2 menunjukan contoh perubahan data hasil *remove punctuation*.

Tabel 2. *Remove Punctuation.*

Data masukan	Data keluaran *Remove Punctuation*
Tanda iman adalah mencintai (kaum) Anshar dan tanda nifaq adalah membenci (kaum) Anshar.	Tanda iman adalah mencintai kaum Anshar dan tanda nifaq adalah membenci kaum Anshar

2. *Case Folding*

Case Folding berfungsi untuk merubah setiap huruf yang terdapat pada dataset menjadi huruf kecil dengan menggunakan fungsi *lowercase*. Tabel 3 menunjukan contoh perubahan data hasil *case folding*.

Tabel 3. *Case Folding.*

Data masukan	Data keluaran *Case Folding*
Tanda iman adalah mencintai kaum Anshar dan tanda nifaq adalah membenci kaum Anshar	tanda iman adalah mencintai kaum anshar dan tanda nifaq adalah membenci kaum anshar

3. *Stemming*

Stemming merupakan proses mengubah kata pada dataset ke bentuk dasar atau dengan kata lain menghapus kata-kata imbuhan. Tabel 4 menunjukan contoh perubahan data hasil *stemming*.

Tabel 4. *Stemming.*

Data masukan	Data keluaran *Stemming*
tanda iman adalah mencintai kaum anshar dan tanda nifaq adalah membenci kaum anshar	tanda iman adalah cinta kaum anshar dan tanda nifaq adalah benci kaum anshar

4. *Tokenization*

Tokenization berdasarkan dokumentasi keras pada proses *tokenization* memungkinkan pembuatan vektor *corpus* teks, dengan mengubah teks menjadi urutan kata-kata yang dipisahkan oleh ruang. Urutan tersebut kemudian dibagi menjadi daftar token, yang selanjutnya akan di indeks atau diubah menjadi vektor.

Pada tahap preprocessing ini akan dilakukan beberapa skenario pengujian untuk mengetahui seberapa besar pengaruh *preprocessing* khususnya pada *stemming* dalam klasifikasi hadis *multi-label* menggunakan algoritma CNN.

3.3. *Convolutional Neural Networks*

Arsitektur sederhana diterapkan dalam pembangunan model *convolutional neural networks*, terdiri dari 5 *layers* yang berbeda. Arsitektur dari CNN yang dibangun dapat dilihat pada Gambar 2.

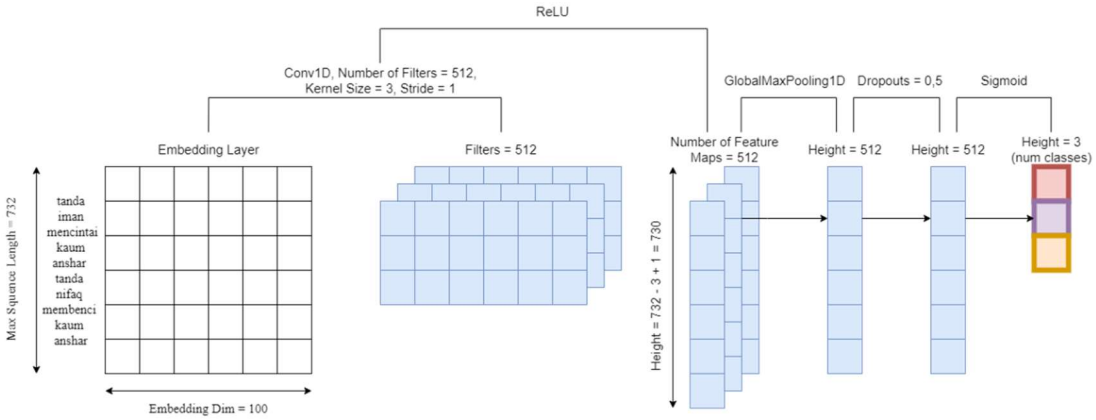
Gambar 2. Arsitektur CNN

Dari kelima layer tersebut, berikut penjelasan parameter setiap layer yang digunakan:

1. *Embedding layer*
   Embedding layer adalah inisialisasi *layer* pada pemodelan CNN dan merupakan input untuk *convolution layer* yang berisi vektor-vektor kata hasil dari tahap *preprocessing* hadis shahih bukhari bahasa Indonesia khususnya tahap tokenisasi. Inisialisasi *input length* yaitu 732 yang merupakan nilai terpanjang dari jumlah kata pada hadis dalam dataset. Tujuan dari pemilihan nilai tersebut adalah agar semua kata dalam setiap hadis terinput tanpa ada hadis yang terpotong.
2. *Conv1D*
   Layer konvolusi dengan parameter jumlah *filter* 512, *padding* sama dengan *valid*, *strides* atau panjang langkah konvolusi adalah 1, dan *kernel size* dengan ukuran sama dengan 3, kemudian menggunakan fungsi aktivasi ReLU pada persamaan (1).
3. *GlobalMaxPooling1D*
   Pada *pooling layer* menggunakan *GlobalMaxPooling1D* untuk mengurangi dimensi yang ada pada *feature map* agar menghasilkan ukuran output yang sama dengan jumlah *feature map* yaitu 512.
4. *Dropout*
   Setelah *pooling layer* pada Gambar 2 adalah *dropout layer*, dengan nilai 0.5 atau sebanyak 50% data.

5. *Fully Connected Layer*
   Pada *layer* ini bertujuan untuk mengurangi vektor 512 menjadi vektor 3 yaitu ukuran yang sama dengan jumlah kelas dari dataset. Fungsi aktivasi menggunakan *sigmoid* pada persamaan (2) untuk proses klasifikasi, memberikan input pada tiap kelas dengan probabilitas yang independen.

Untuk lebih jelasnya berdasarkan penelitian [6] oleh Sarah Fauziah Lestari berikut Gambar 3 menunjukan model klasifikasi dengan menggunakan Convolutional Neural Networks yang telah dimodifikasi sesuai model yang dibangun.



Gambar 3. Model CNN.

3.4. *Hamming Loss*

Setelah model terbangun, maka tahap terakhir yaitu perhitungan *hamming loss* dengan menggunakan persamaan (3) untuk mengukur seberapa besar *misclassification* atau kesalahan kinerja model yang dibangun.

4. Evaluasi

4.1. Skenario Pengujian

Untuk melihat hasil evaluasi dan mengetahui performansi sistem yang dibangun perlu dilakukannya proses skenario pengujian. Berikut ini adalah macam skenario yang digunakan pada penelitian ini:

1. Pengujian pengaruh *preprocessing* berdasarkan *stemming* terhadap nilai *hamming loss* dan waktu eksekusi pada CNN
2. Pengujian pengaruh *padding* pada *convolutional layer* terhadap nilai *hamming loss* dan waktu eksekusi pada CNN
3. Pengujian pengaruh *strides* pada *convolutional layer* terhadap nilai *hamming loss* dan waktu eksekusi pada CNN
4. Membandingkan antara model CNN dengan model yang menggunakan algoritma dengan input yang sama berbentuk *sequence* yaitu LSTM dan RNN berdasarkan nilai *hamming loss* dan waktu eksekusi.

Preprocessing pada model lain yaitu LSTM dan RNN yang akan digunakan pada skenario ke-4 merupakan skenario *preprocessing* berdasarkan *stemming* yang menghasilkan *hamming loss* terbaik. Selain itu skenario dengan nilai *hamming loss* terbaik dari semua skenario yang melibatkan CNN dijadikan pembanding untuk model lain.

Konfigurasi untuk model LSTM dan RNN sama dengan CNN. *Embedding* dengan inisialisasi input hasil tokenisasi dan *input_length* 732, *layer* LSTM dan RNN dengan parameter 512 unit, GlobalMaxPool1D, *Dropout* 0,5, *dense* dengan aktivasi *sigmoid*, konfigurasi *loss* menggunakan *binary_crossentropy*, *optimizer* adam dengan *learning_rate* 0,01, dan pembagian dataset menjadi 80% data untuk *training* serta 20% data untuk *testing*.

4.2. Hasil Pengujian

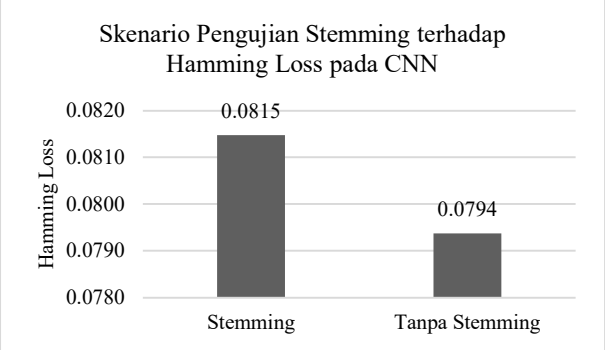
Nilai hamming loss dari semua skenario pengujian yang melibatkan model CNN menghasilkan nilai yang berbeda. Berikut adalah Tabel 5 menunjukan nilai *hamming loss* pada tiap pemodelan yang melibatkan CNN. Deskripsi kolom *stemming* huruf Y berarti menggunakan *stemming* dan huruf N tanpa *stemming*, kolom *padding* huruf Y berarti menggunakan *padding* dan huruf N tanpa *padding*, pada kolom *strides integer* 1 berarti 1 langkah konvolusi dan *integer* 3 berarti 3 langkah konvolusi.

Tabel 5. Skenario CNN berdasarkan nilai hamming loss dan waktu eksekusi

Stemming	*Padding*	*Strides*	Waktu Eksekusi	*Hamming Loss*
Y	Y	1	68.5111	0.0794
Y	Y	3	35.878	0.0886
Y	N	1	67.6085	0.0702
Y	N	3	36.1323	0.0877
N	Y	1	68.392	0.0794
N	Y	3	35.9005	0.0844
N	N	1	67.7613	**0.0693**
N	N	3	36.2349	0.0844

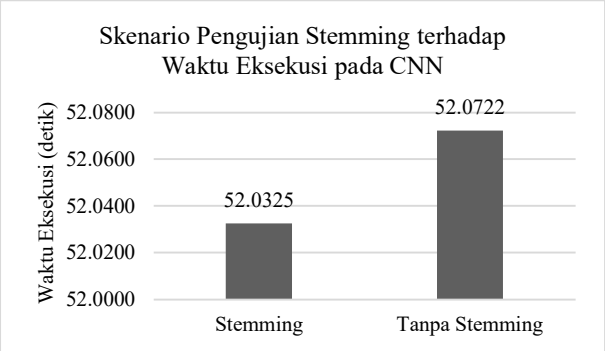
1. Skenario *preprocessing* berdasarkan *stemming* terhadap nilai *hamming loss* dan waktu eksekusi pada CNN

   Nilai *hamming loss* pemodelan CNN pada skenario *preprocessing* berdasarkan *stemming* memiliki nilai yang berbeda. Berikut berdasarkan Tabel 5, Gambar 4 menunjukan rata-rata *hamming loss* dan Gambar 5 menunjukan rata-rata waktu eksekusi.



Gambar 4. Nilai *hamming loss* CNN berdasarkan *stemming*.

Gambar 4 menunjukan skenario pengujian tanpa menggunakan *stemming* terhadap hamming loss pada CNN menghasilkan nilai terbaik yaitu 0,0794.

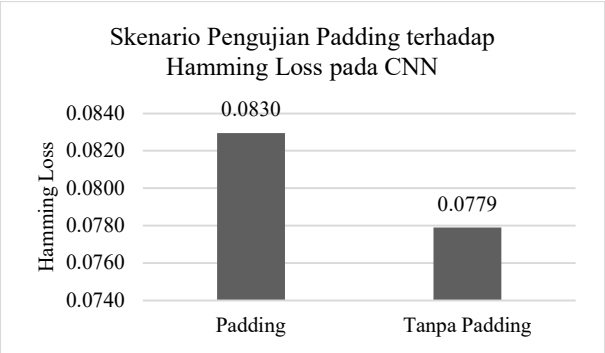


Gambar 5. Nilai waktu eksekusi CNN berdasarkan *stemming*.

Gambar 5 menunjukan skenario pengujian dengan menggunakan *stemming* terhadap waktu eksekusi pada CNN menghasilkan nilai terbaik yaitu 52,0325 detik.

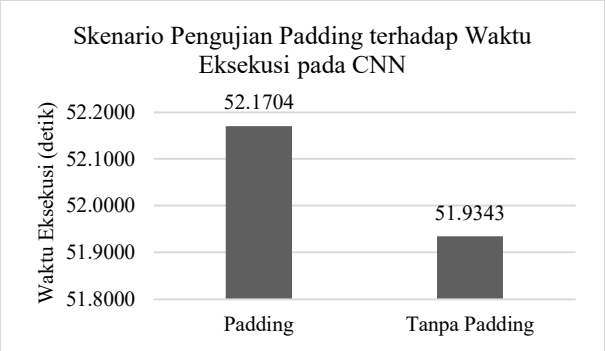
2. Skenario penggunaan *padding* terhadap nilai *hamming loss* dan waktu eksekusi pada CNN

   Nilai *hamming loss* pemodelan CNN dengan skenario penggunaan *padding* pada *convolutional layer* memiliki nilai yang berbeda. Berikut berdasarkan Tabel 5 adalah Gambar 6 menunjukan rata-rata nilai *hamming loss* dan Gambar 7 menunjukan rata-rata waktu eksekusi.



Gambar 6. Nilai *hamming loss* CNN berdasarkan *padding*.

Gambar 6 menunjukan skenario pengujian tanpa menggunakan *padding* terhadap hamming loss pada CNN menghasilkan nilai terbaik yaitu 0,0779.

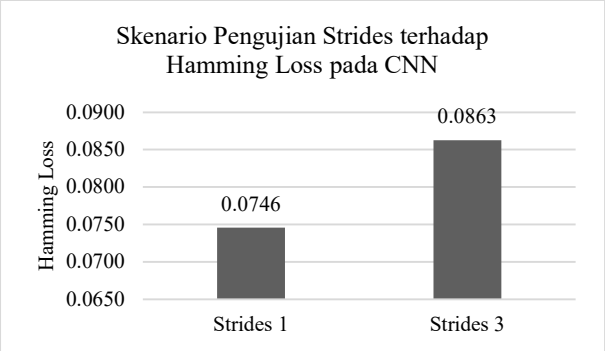


Gambar 7. Nilai waktu eksekusi CNN berdasarkan *padding*.

Gambar 7 menunjukan skenario pengujian dengan tanpa *padding* terhadap waktu eksekusi pada CNN menghasilkan nilai terbaik yaitu 51,9343 detik.

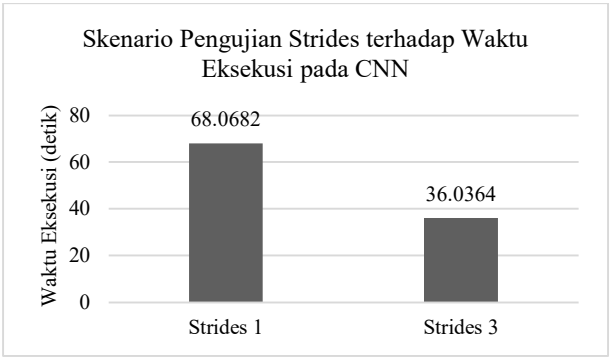
3. Skenario penggunaan *strides* terhadap nilai *hamming loss* dan waktu eksekusi pada CNN

   Nilai *hamming loss* pemodelan CNN dengan skenario penggunaan *strides* pada *convolutional layer* memiliki nilai yang berbeda. Berikut berdasarkan Tabel 5 adalah Gambar 8 menunjukan rata-rata nilai *hamming loss* dan Gambar 9 menunjukan rata-rata waktu eksekusi.



Gambar 8. Nilai *hamming loss* CNN berdasarkan *strides*.

Gambar 8 menunjukan skenario pengujian dengan menggunakan 1 *strides* terhadap hamming loss pada CNN menghasilkan nilai terbaik yaitu 0,0746.



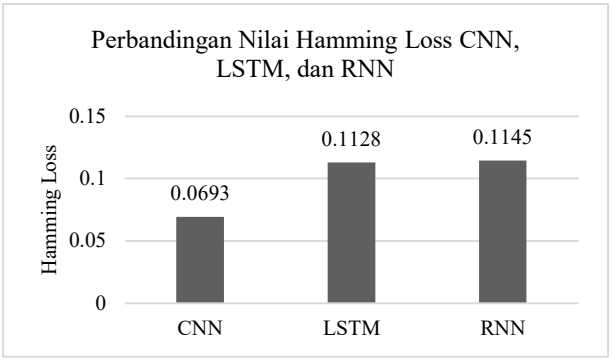
Gambar 9. Nilai waktu eksekusi CNN berdasarkan *strides*.

Gambar 9 menunjukan skenario pengujian dengan menggunakan 3 *strides* terhadap waktu eksekusi pada CNN menghasilkan nilai terbaik yaitu 36,0364 detik.

4. Skenario pengujian perbandingan CNN, LSTM, dan RNN berdasarkan nilai *hamming loss* dan waktu eksekusi

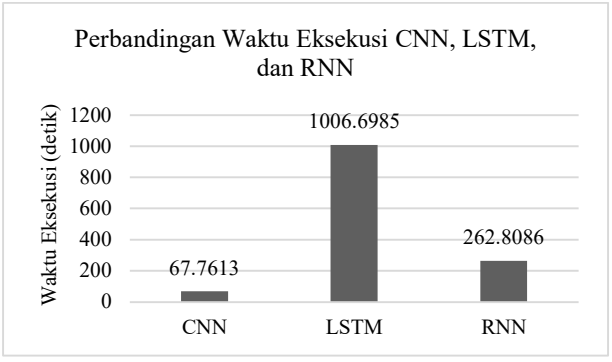
   Gambar 4 dan Gambar 5 pada hasil skenario ke-1 menunjukan *preprocessing* tanpa *stemming* pada CNN menghasilkan nilai *hamming loss* terbaik yaitu sebesar 0,0794, maka skenario *preprocessing* tersebut yang digunakan pada pengujian pemodelan LSTM dan RNN. Kemudian pada Tabel 5 untuk semua skenario yang melibatkan CNN menghasilkan nilai *hamming loss* terbaik 0,0693 dengan waktu eksekusi 67,7613 detik, maka nilai CNN tersebut yang digunakan untuk membandingkan performansi dengan model lain.

   . Nilai *hamming loss* dan waktu eksekusi pemodelan pada tiap model memiliki nilai yang berbeda. Berikut adalah Gambar 10 menunjukan nilai *hamming loss* dan Gambar 11 menunjukan waktu eksekusi pada masing-masing model yang telah dibangun.



Gambar 10. Nilai *hamming loss* CNN, LSTM, dan RNN.

Gambar 10 menunjukan pemodelan dengan menggunakan CNN menghasilkan *hamming loss* lebih baik dari LSTM dan RNN dengan nilai sebesar 0,0693.



Gambar 11. Nilai waktu eksekusi CNN, LSTM dan RNN.

Gambar 11 menunjukan pemodelan dengan menggunakan CNN menghasilkan waktu eksekusi tercepat dari LSTM dan RNN dengan nilai sebesar 67,7613 detik.

4.3. Analisis Hasil Pengujian

Berdasarkan beberapa hasil pengujian yang telah dilakukan, dari skenario pengujian *stemming* terhadap CNN menunjukan skenario *preprocessing* tanpa *stemming* mendapatkan nilai *hamming loss* terbaik namun waktu eksekusinya menjadi sedikit lebih lama. Terkait hal ini, ditinjau berdasarkan nilai *hamming loss* tanpa melibatkan proses *stemming* pada dataset berfungsi mengubah kata menjadi kata dasar yang mengakibatkan kalimat *multi-label* pada dataset menjadi tidak kompleks. Contoh pada kata "shalatlah" seharusnya bermakna anjuran, namun setelah proses *stemming* menjadi kata "shalat", yang mana kata tersebut jika berada pada kalimat lain belum tentu bermakna anjuran, namun bermakna larangan seperti pada kalimat "jika datang haidmu maka tinggalkanlah shalat". Hal tersebut yang menjadikan nilai *hamming loss preprocessing* tanpa *stemming* CNN menjadi lebih baik yaitu sebesar 0,0794. Ditinjau berdasarkan waktu eksekusi CNN, skenario *preprocessing* dengan melibatkan *stemming* menghasilkan waktu yang tercepat yaitu sebesar 52,0325 detik dibandingkan tanpa menggunakan *stemming*. Hal ini terjadi karena adanya perubahan bentuk kata pada proses *stemming* menjadi kata dasar yang berarti huruf dalam suatu kata akan menjadi lebih sedikit yang dapat mempersingkat waktu proses *training*.

CNN tanpa *padding* mendapatkan nilai *hamming loss* terbaik yaitu 0,0779 dibandingkan dengan menggunakan *padding* yaitu sebesar 0,0830. Selain itu waktu eksekusi pemodelan juga lebih cepat CNN tanpa *padding* yaitu 51,9343 detik dibandingkan dengan menggunakan *padding* yaitu sebesar 52,1704 detik. Hal ini terjadi karena dengan menggunakan *padding* maka ada penambahan data fitur di setiap sisi dimensi pada proses pemodelan yang dapat menambah waktu saat konvolusi dan dapat menyebabkan posisi fitur tambahan tersebut bisa jadi tidak sesuai dengan label yang telah ditentukan.

CNN dengan menggunakan *strides* 1 mendapatkan nilai *hamming loss* terbaik yaitu sebesar 0,0746 dibandingkan dengan *strides* 3 yaitu sebesar 0,0863. Namun waktu eksekusi pemodelan dengan *strides* 3 lebih cepat yaitu sebesar 36,0364 detik dibandingkan dengan *strides* 1 yaitu sebesar 68,0682 detik. Hal ini terjadi karena saat menggunakan *strides* 3, langkah saat konvolusi atau pengambilan peta fitur ke n maka pengambilan peta fitur ke n+1 dan seterusnya tidak tepat disisinya yang berarti ada fitur yang terlewati dan menyebabkan proses saat learning ada beberapa kata tidak sesuai dengan label yang telah ditentukan, namun dengan strides 3 karena ada beberapa fitur yang tidak diambil atau dilewati maka dapat mempercepat proses konvolusi dan proses pembelajarannya.

Skenario CNN dengan nilai *hamming loss* terbaik untuk perbandingan dengan model lain yaitu meliputi skenario *preprocessing* tanpa *stemming*, layer konvolusi tanpa *padding*, dan *strides* 1 mendapatkan nilai *hamming loss* sebesar 0,0693. Karena tanpa adanya *stemming* dapat menjaga keaslian data pada hadis, tanpa *padding* tidak akan ada fitur atau kata tambahan pada tiap sisi dimensi, dengan *strides* 1 maka langkah pengambilan fitur saat konvolusi akan secara merata dan menyeluruh. Hal ini berdampak pada saat proses klasifikasi dapat meningkatkan kinerja sistem untuk dikelompokan sesuai kelas dan labelnya.

Perbandingan CNN dengan model lain yaitu LSTM dan RNN memiliki nilai *hamming loss* dan waktu eksekusi yang berbeda. *Hamming loss* terbaik pada CNN memiliki nilai yang lebih baik yaitu 0,0693 dibandingkan dengan LSTM yang memiliki nilai *hamming loss* 0,1128 dan RNN 0,1145. Waktu eksekusi pemodelan CNN juga memiliki waktu tercepat sebesar 67,7613 detik dibandingkan dengan LSTM yang memiliki waktu eksekusi pemodelan 1006,6985 detik dan RNN 262,8086 detik. Hal ini terjadi karena CNN memiliki *layer* konvolusi yang memanfaatkan *filter* dengan *kernel* untuk mengekstrak fitur yang relevan dan mempelajarinya secara otomatis tanpa mengetahui secara eksplisit, selain itu proses perhitungan CNN dilakukan secara paralel.

5. Kesimpulan

Berdasarkan hasil pengujian dan analisis yang telah dilakukan, dengan dataset *multi-label* hadis shahih bukhari terjemah Bahasa Indonesia dan penggunaan data *testing* sebesar 20%. Hasil pengujian *preprocessing* tanpa *stemming* menghasilkan *hamming loss* terbaik pada pemodelan CNN dibandingkan menggunakan *stemming* namun penggunaan *stemming* dapat sedikit mempercepat waktu eksekusinya. Penggunaan *padding* pada CNN perlu dihindari, karena tanpa menggunakan *padding* CNN mendapatkan performansi terbaik. *Strides* 1 saat konvolusi mendapatkan nilai *hamming loss* terbaik namun waktu eksekusi menjadi lebih lama dibandingkan dengan *strides* 3. Selain itu performansi CNN juga lebih baik dibandingkan dengan LSTM dan RNN yaitu dengan nilai *hamming loss* 0,0693 dan waktu eksekusi pemodelan 67,7613 detik. Saran untuk penelitian selanjutnya perlu dilakukan menambahkan dataset hadis karena CNN tergolong dalam *deep learning* untuk proses *training* yang lebih baik.

Referensi

[1] Al-Qur'an,3:32.
[2] Aw, L. C. (2011). Memahami Makna Hadis Secara Tekstual Dan Kontekstual. Ulumuna, 15(2), 391-414.
[3] Khon, A. M. (2012). Ulumul hadis. Amzah.
[4] Albani, M. N. A. D. (2003). Ringkasan Shahih Bukhari 1. Gema Insani.
[5] Pridyandhika, R. R. F., Adiwijaya, A., & Astuti, W. (2019). Klasifikasi Teks Multi-Label pada Hadis menggunakan Recurrent Neural Network.
[6] Lestari S Adiwijaya, A., & Al Faraby, S. (2019). Klasifikasi Multi-label pada Terjemahan Al-Quran Berbahasa Inggris Menggunakan Convolutional Neural Networks.
[7] Harjoseputro, Y. (2018). Convolutional Neural Network (CNN) Untuk Pengklasifikasian Aksara Jawa.
[8] Prasetyo, H., Adiwijaya, A., & Astuti, W. (2019). Klasifikasi Multi-label Pada Hadis Bukhari Dalam Terjemahan Bahasa Indonesia Menggunakan Mutual Information Dan Backpropagation Neural Network. eProceedings of Engineering, 6(2).
[9] Wiraguna, A., Al Faraby, S., & Adiwijaya, A. (2019). Klasifikasi Topik Multi Label pada Hadis Bukhari dalam Terjemahan Bahasa Indonesia Menggunakan Random Forest. eProceedings of Engineering, 6(1).
[10] Amran., Adiwijaya, A., & Astuti, W. (2019). Penerapan Multinomial Naive Bayes Dalam Klasifikasi Multilabel Pada Hadis.
[11] Mediamer, G., & Adiwijaya, F. SA (2019).“Development of Rule-Based Feature Extraction in Multi-label Text Classification”. International Journal on Advanced Science, Engineering and Information Technology, 9(4), 1460-1465.
[12] Wahyuni, R. T., Prastiyanto, D., & Supraptono, E. (2017). Penerapan algoritma cosine similarity dan pembobotan tf-idf pada sistem klasifikasi dokumen skripsi. Jurnal Teknik Elektro, 9(1), 18-23.
[13] Fauziah, I. A. N., Sibaroni, Y., & Lhaksmana, K. M. (2020) Pengaruh Stemming Bahasa Indonesia Terhadap Analisis Sentimen pada Twitter (Menggunakan Dataset: Gojek)
[14] P. Li, F. Zhao , Y. Li and Z. Zhu, "Law Text Classification Using Semi-supervised Convolutional Neural Networks," in The 30th Chinese Control and Decision Conference (2018 CCDC), 2018.
[15] Keras, "Keras Documentation," [Online]. Available: https://keras.io/layers/core/.
[16] Guo, T., Dong, J., Li, H., & Gao, Y. (2017, March). Simple convolutional neural network on image classification. In 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)((pp. 721-724). IEEE.
[17] J. Rathod, V. Waghmode, A. Sodha and D. Bhavathankar, "Diagnosis of skin diseases using Convolutional Neural Networks," in Proceedings of the 2nd International conference on Electronics, Communication and Aerospace Technology (ICECA 2018), 2018.
[18] Venkatesan, R., Er, M. J., Wu, S., & Pratama, M. (2016, July). A novel online real-time classifier for multi-label data streams. In 2016 International Joint Conference on Neural Networks (IJCNN) (pp. 1833-1840). IEEE.
[19] J. Read, "Multi-label Classification," 5 September 2015. [Online]. Available: https://users.ics.aalto.fi/jesse/talks/Tutorial-MLC-Porto.pdf.