Jurnal Elektronik Ilmu Komputer Udayana
Volume 10, No 4. May 2022

p-ISSN: 2301-5373
e-ISSN: 2654-5101

Implementasi LSTM pada Analisis Sentimen *Review* Film Menggunakan *Adam* dan *RMSprop Optimizer*

Karlina Surya Witanto[a1], Ngurah Agus Sanjaya ER[a2], AAIN Eka Karyawati[a3], I Gusti Agung Gede Arya Kadyanan[a4], I Ketut Gede Suhartana[a5], Luh Gede Astuti[a6]

[a]Program Studi Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam,
Universitas Udayana
Bali, Indonesia
[1]gabriella.linatan@gmail.com
[2]agus_sanjaya@unud.ac.id
[3]eka.karyawati@unud.ac.id
[4]gungde@unud.ac.id
[5]ikg.suhartana@unud.ac.id
[6]lg.astuti@unud.ac.id

Abstract

Movies are an entertainment that is in great demand by many groups from children, teenagers, adults, and parents. In the current digital era, various films can be watched on television to digital streaming services. Public opinion on the films watched can be in the form of positive opinions or negative opinions. Sentiment analysis is one of the fields of Natural Language Processing (NLP) which is able to build a system to recognize and extract opinions in the form of text, sentiment analysis is usually used to find out people's opinions or assessments of a products, services, politics, or other topics. Through sentiment analysis from the collection of reviews, the public can get various recommendations for films that can be watched. The method implemented to classify review data into positive reviews and negative reviews in this study is LSTM by comparing two different optimizers, namely Adam and RMSprop. This study succeeded in providing sentiment predictions with different optimizers with accuracy values for the LSTM application with Adam Optimizer reaching 77.11% and the LSTM application with RMSprop reaching 80.07%.

Keywords: **Film, Review, Sentiment, NLP, LSTM, Adam, RMSprop**

1. Pendahuluan

Film adalah sebuah hiburan yang banyak diminati oleh banyak golongan dari anak-anak, remaja, dewasa, dan orang tua. Pada era digital saat ini, berbagai film dapat ditonton melalui televisi hingga melalui layanan *digital streaming*. Berbagai layanan *digital streaming* seperti Netflix, Viu, Iflix, We TV, dan layanan *movie streaming* lainnya mampu menarik lebih dari dua ratus juta pelanggan hingga tahun 2020. Pada tahun 2020 jumlah penonton mengalami peningkatan yang sangat mencolok karena adanya keterkaitan dengan pandemi Covid-19 yang membuat minat penonton lebih tinggi untuk menonton berbagai macam film [1]. Opini masyarakat terhadap film yang ditonton dapat berupa opini positif maupun opini negatif. Opini-opini tersebut dapat ditemukan pada sebuah *review* dan dapat dianalisis dengan analisis sentimen. Analisis sentimen atau *opinion mining* merupakan pengolahan bahasa alami untuk melacak sikap, perasaan, atau penilaian masyarakat terhadap suatu topik tertentu, produk, atau jasa. Analisis sentimen merupakan salah satu bidang dari *Natural Language Processing* (NLP) yang mampu membangun suatu sistem untuk mengenali dan melakukan ekstraksi opini dalam bentuk teks, analisis sentimen biasanya digunakan untuk mengetahui opini atau pendapat masyarakat terhadap layanan atau jasa, produk, politik, ataupun topik-topik lain. Melalui analisis sentimen dari kumpulan *review* tersebut, masyarakat bisa mendapatkan berbagai rekomendasi film yang dapat ditonton. *Long Short Term Memory* (LSTM) merupakan pengembangan dari algoritma *Recurrent Neural Network* (RNN) dimana terdapat modifikasi pada RNN dengan menambahkan *memory cell* atau *memory unit* yang dapat menyimpan informasi yang dipelajari LSTM dalam jangka waktu yang panjang [2]. LSTM memberikan solusi untuk mengatasi terjadinya *vanishing gradient* pada RNN saat memproses data *sequential* yang panjang. Permasalahan *vanishing gradient* ini mengakibatkan RNN gagal dalam menangkap *long term dependencies* [3], sehingga mengurangi akurasi dari suatu prediksi pada RNN [4]. Terdapat penelitian dengan mengimplementasikan metode LSTM untuk melakukan analisis sentimen pada situs IMDB dengan menerapkan *word2vec* [5]. Penelitian tersebut memberikan

hasil akurasi yang cukup baik untuk memprediksi sentimen, yaitu dengan nilai akurasi sebesar 80%. Penelitian lain dengan C-LSTM dengan menerapkan *Adam Optimizer* pada Klasifikasi Berita Bahasa Indonesia mampu memberikan nilai akurasi sebesar 93,27% [6]. Pada penelitian ini akan dilakukan analisis sentimen terhadap kumpulan *review* film melalui IMDb Largest Review Dataset yang sudah didapatkan melalui website IEEE Dataport. Metode yang diimplementasikan untuk melakukan klasifikasi terhadap data *review* ke dalam *review* positif dan *review* negatif adalah LSTM. Penelitian ini juga akan membandingkan hasil performa dari perbandingan dua *optimizer* yaitu *Adam Optimizer* dan *RMSprop Optimizer* terhadap penerapan algoritma LSTM, dimana kedua *optimizer* tersebut mampu mengoptimalkan performa dari algoritma LSTM. Sehingga dengan penggunaan *optimizer* yang tepat, diharapkannya *output* yang akan diberikan mampu memberikan rekomendasi film kepada *user* serta memberikan prediksi analisis sentimen *review* yang benar.

2. Metode Penelitian

Penelitian ini meliputi beberapa tahapan yaitu, pengumpulan data teks kumpulan *review* film melalui dataset *IMDB Largest Review* dalam Bahasa Inggris yang diambil melalui web IEEE Dataport [7], *preprocessing*, *word embedding*, pemodelan *Long Short Term Memory* (LSTM) dengan menerapkan *K-Fold Cross Validation* untuk klasifikasi dengan membandingkan dua *optimizer* yang berbeda yaitu *Adam* dan *RMSprop Optimizer*, dan tahap evaluasi.

2.1. Pengumpulan Data

Data yang digunakan pada penelitian ini merupakan data sekunder, dimana data tersebut sudah tersedia sebelum peneliti memulai penelitian, dan data tersebut berhubungan dengan penelitian yang akan dilakukan oleh peneliti [8]. Data yang diambil merupakan dokumen kumpulan *review* film melalui dataset *IMDB Largest Review* yang bersumber dari web IEEE Dataport [7]. Pengumpulan data teks kumpulan *review* film melalui dataset *IMDB Largest Review* dalam Bahasa Inggris yang diambil melalui web IEEE Dataport [7]. Data yang digunakan pada penelitian ini sebanyak 14.000 data *review* film dalam Bahasa Inggris, yang mengandung 7.000 *review* dengan label positif dan 7.000 *review* dengan label negatif dalam format *.*csv*. Data tersebut akan digunakan untuk melakukan proses pelatihan dan pengujian dari model yang akan dibangun menggunakan algoritma LSTM, serta untuk mengembangkan *deep learning model* terhadap *binary classification* berdasarkan *review* film.

2.2. *Preprocessing*

Setelah melakukan pengumpulan data, tahap selanjutnya yaitu *preprocessing.* Langkah ini perlu dilakukan untuk mempersiapkan teks yang menjadi sumber data agar dapat diproses ke tahap selanjutnya [9]. Hal-hal yang dilakukan pada langkah ini, yaitu:

a. *Delete Null Value*
   Proses ini dilakukan dengan menghapus kolom yang memiliki nilai *null* sehingga tidak menyebabkan *error* ketika proses pelatihan dan pengujian model [10].
b. *Tokenizing*
   Kata-kata yang yang terdapat dalam dokumen akan dipecah menjadi bagian yang lebih kecil berupa kata tunggal yang memiliki arti atau sering disebut token, misalnya berupa kata, frasa, atau kalimat [11].
c. *Case Folding*
   Tahapan ini dilakukan untuk mengkonversi keseluruhan teks dalam dokumen menjadi suatu bentuk standar, yaitu huruf kecil atau *lowercase* [12].
d. *Stopword removal*
   Stopword removal merupakan tahap preprocessing dimana dilakukan penghapusan kata-kata yang sering muncul dan memiliki arti sedikit atau tidak penting yang disebut sebagai stop words. Penghapusan *stop words* dilakukan untuk mengurangi kata dalam sistem [13].
e. *Punctuation removal*
   Punctuation removal merupakan proses untuk menghilangkan simbol-simbol yang terdapat pada dokumen teks [14].
f. *Text to Sequence*
   Proses ini merupakan proses merepresentasikan kamus kata (senilai *num_words*) menjadi ke bentuk angka [15].
g. *Split Data*
   Proses pembagian data dengan persentase 90% data pelatihan dan 10% data pengujian dari total jumlah data [16]. Langkah ini merupakan langkah untuk melatih *neural network* agar dapat mengenali pola sehingga model algoritma yang diterapkan bisa mendapatkan akurasi yang baik.

p-ISSN: 2301-5373
e-ISSN: 2654-5101

2.3. *Word Embedding*

Proses *word embedding* dilakukan untuk merubah representasi dari kata-kata yang terdapat dalam *dataset* menjadi sebuah vektor. Proses ini diawali dengan membuat list dari kata-kata yang terdapat dalam teks. Pada penelitian ini peneliti menggunakan *embedding* dari *Keras*. Setelah dilakukan tahapan *preprocessing* dan menghasilkan ulasan dalam bentuk list kata yang sudah ditokenisasi, tahap selanjutnya yaitu memberikan indeks pada setiap kata pada *dataset.* Pada tahap ini diperlukan dua parameter. Parameter yang pertama yaitu parameter *jumlah_vocab* yang berfungsi untuk mengatur ukuran kamus kata yang ingin digunakan. Pada penelitian ini, nilai dari *jumlah_vocab* yang digunakan oleh peneliti sebesar 10.000 *vocab*, dimana 10.000 *vocab* yang disimpan memiliki persentase yang besar terhadap kemunculan suatu kata. Parameter selanjutnya adalah *output_dim* yang berguna untuk mengatur panjang urutan vektor (dimensi vektor).

Pseudocode Word Embedding

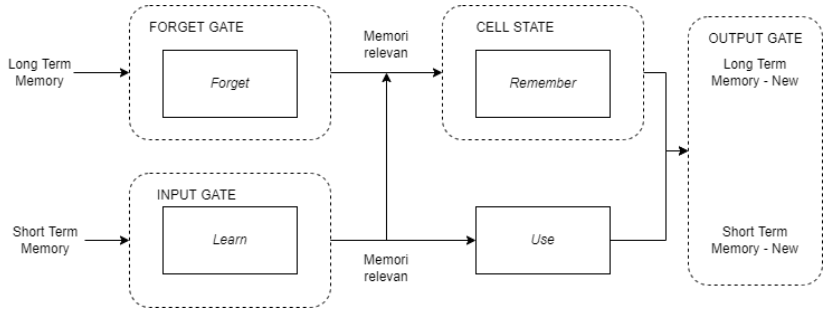
```
algorithm : create embedding
input : d = dataset
output : matrix W(10.000,300) of one-hot vectors for each possible byte value
(0-9999)
let f be a list of tuples(byte_value, frequency)
for i=0 to 9999 do
   freq ← 0

for each item j in d do
   freq ← freq + frequencyOfOccurence(i,j)
end for
   append (i, freq) tuple of f
end for
f ← sort of based on frequencies
W ← embedding(f, 300)
return W
```

Gambar 1. *Pseudocode Word Embedding*

Pada proses *word embedding* seperti pada Gambar 1 terdapat metode yang digunakan yaitu *one hot encoder* atau *text to sequence* yang digunakan untuk mengubah kata menjadi angka atau bentuk numerik yang nantinya angka tersebut akan diproses pada tahap *word embedding* untuk diubah menjadi vektor. *Word embedding* menggunakan kumpulan kosakata dari data teks pelatihan sebagai input yaitu sebanyak 10.000 kata kemudian mempelajari representasi vektor dari kumpulan kata tersebut. *Word embedding* bekerja dengan menangkap informasi pada setiap kata atau *byte* yang memiliki skor kesamaan yang tinggi dan kata-kata yang memiliki tingkat kesamaan yang tinggi akan ditempatkan pada posisi yang berdekatan.

2.4. *Long Short-Term Memory* (LSTM)

LSTM merupakan bagian dari algoritma *Recurrent Neural Network* (RNN). LSTM memiliki koneksi secara berulang dan strukturnya seperti rantai yang dapat mempelajari ketergantungan jangka panjang (*Long-term Dependencies*) dalam kasus prediksi yang sebelumnya menjadi kelemahan algoritma RNN. Selain itu algoritma ini juga untuk menangani permasalahan pada *machine learning*, *speech recognition*, dan lain-lain [17]. LSTM menerapkan memori jangka panjang pada jaringan saraf untuk mengurangi masalah *vanishing gradient.*

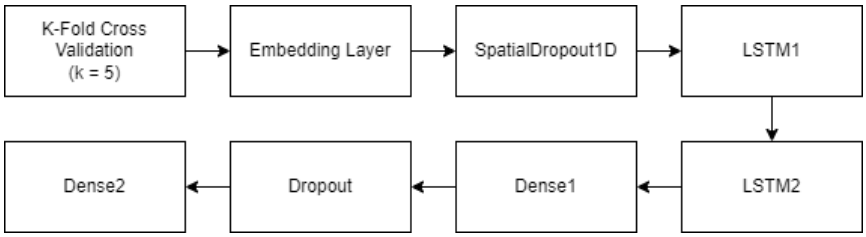


Gambar 2. Arsitektur LSTM secara Umum

Arsitektur LSTM pada Gambar 2 secara umum memiliki empat gerbang yaitu, *forget gate*, *input gate*, *cell state*, dan *output gate. Forget gate*, akan menerima informasi lama (*Long Term Memory*) dan akan dikalikan dengan aktivasi *sigmoid* yang bernilai antara 0 dan 1. Jika nilai yang dihasilkan mendekati 1 maka informasi tersebut dinyatakan relevan dan akan diproses, jika nilainya mendekati nilai 0 maka informasi tersebut akan dilupakan. *Input gate*, akan menerima atau mempelajari informasi baru (*Short*

Term Memory) dan informasi tersebut akan disimpan ke dalam *cell state* dan akan digunakan untuk memproses *output*. *Cell state,* informasi yang tidak dilupakan dan informasi dari *input gate* akan disimpan dalam *cell state*. Setiap informasi yang disimpan dalam setiap *gate* tersebut akan menghasilkan *Long Term Memory New* dan *Short Term Memory New* yang akan dijadikan sebagai *output* dari LSTM yang memberikan prediksi serta memori atau informasi yang paling relevan. Gambaran arsitektur LSTM secara umum tersebut menunjukkan bahwa LSTM mampu menyimpan memori dalam jangka panjang dan dapat mengurangi masalah *vanishing gradient* karena LSTM selalu memperbaharui memori yang dibutuhkan.

Seperti pada Gambar 3, pembuatan model LSTM pada penelitian ini mengimplementasikan metode *K-Fold Cross Validation* serta *sequence model* yang terdiri dari tujuh *layer* yang diantaranya menerapkan *Embedding layer*, *SpatialDropout1D layer*, *LSTM1 layer*, *LSTM2 layer, Dense1 layer, Dropout layer*, serta *Dense2 layer*.



Gambar 3. *Layer* Model LSTM

Pada penelitian ini menerapkan *K-Fold Cross Validation* dengan jumlah k=5 yang digunakan untuk membagi data ke dalam tiap ruang *fold.* Total jumlah data *review* sebanyak 14.000 data, yaitu 7.000 data *review* positif dan 7.000 data *review* negatif. Pada proses *split data*, data yang akan dilatih pada model sebanyak 90% dari total jumlah data. Sehingga 12.600 data akan dilatih dengan model yang akan dibangun. Data tersebut akan dibagi kedalam lima *fold* sama rata dan selanjutnya akan dilatih ke dalam model LSTM.

Embedding layer terdapat parameter ukuran *vocab* dan ukuran vektor *embedding*. Pada penelitian ini, peneliti menerapkan nilai sebesar 10.000 untuk ukuran *vocab,* dimana jumlah tersebut sudah memberikan representasi kata yang paling besar atau yang sering muncul. Lalu untuk ukuran vektor *embedding,* penulis menerapkan nilai sebesar 300. *SpatialDropout1D layer* memiliki fungsi untuk mencegah terjadinya *overfitting* dan *underfitting* pada data tekstual. *SpatialDropout1D* bekerja dengan mempertahankan fitur dalam data yang memiliki korelasi tinggi satu sama lain, sehingga fitur yang memiliki korelasi yang tinggi satu sama lain tidak akan masuk ke dalam proses regulasi dan fitur yang berkorelasi tinggi tersebut dapat digunakan.

LSTM1 layer yaitu LSTM *layer* pertama memiliki nilai *memory unit* yaitu jumlah unit LSTM dan menerapkan atribut *return_sequences=True*. Parameter *memory unit* nilainya harus ditentukan secara eksplisit dengan rentang angka yang sesuai dengan data yang penulis miliki untuk memberikan akurasi yang maksimal dari model tersebut dan dilakukan secara repetitif hingga ditemukan *hyperparameter* terbaik. *LSTM2 layer* yaitu LSTM *layer* kedua memiliki nilai *memory unit* yaitu jumlah unit LSTM, *kernel_regularizers*(L2), dan *recurrent_regularizer*(L2). Parameter *memory unit* nilainya harus ditentukan secara eksplisit dengan rentang angka yang sesuai dengan data yang penulis miliki untuk memberikan akurasi yang maksimal dari model tersebut dan dilakukan secara repetitif hingga ditemukan *hyperparameter* terbaik. *Kernel_regularizer* dan *recurrent_regularizer* merupakan parameter yang digunakan untuk mencegah terjadinya *overfitting* serta mempercepat proses *learning*, nilai parameter acuan dari fungsi regulasi yaitu 0 hingga tak terhingga [18]. Versi L2 pada *regularizer* bekerja dengan memperkirakan rata-rata data untuk menghindari *overfitting.*

Dense1 layer yaitu *dense layer* pertama yang berfungsi untuk menambahkan layer yang *fully connected*, artinya setiap neuron menerima input dari neuron lainnya sehingga saling terhubung. Jumlah unit yang digunakan pada *dense1 layer* sebanyak delapan neuron diikuti dengan fungsi aktivasi *relu*, dimana fungsi aktivasi *relu* berguna untuk mengoptimalkan fungsi aktivasi *sigmoid* yang akan diterapkan pada *Dense2 layer*. Selanjutnya terdapat *dropout layer* yang berfungsi untuk mencegah terjadinya *overfitting* dan mempercepat proses *learning.* Nilai yang digunakan berkisar dari 0 sampai 1, dimana semakin kecil nilai *dropout* maka data *overfit*. Sebaliknya, data akan menjadi *underfit*. *Dense2 layer* yaitu *dense layer* kedua yang berfungsi untuk menambahkan layer yang *fully connected* dan

p-ISSN: 2301-5373
e-ISSN: 2654-5101

disesuaikan dengan jumlah *class* yang ditentukan, artinya setiap neuron menerima input dari semua neuron lainnya sehingga saling terhubung. Jumlah unit yang digunakan pada *dense2 layer* sebanyak dua neuron karena terdapat dua kelas yang telah ditentukan, yaitu kelas positif dan kelas negatif. Penelitian berupa *binary-classification*¸ maka digunakan *loss function = binary_crossentropy*, fungsi optimasi, dan aktivasi *sigmoid*. Fungsi aktivasi *sigmoid* digunakan untuk memilih probabilitas kelas terbesar pada klasifikasi biner.

Pada penelitian ini arsitektur *deep learning* yang sudah dibangun akan dibandingkan dengan beberapa nilai *epoch* dan nilai *batch size*. Nilai *epoch* dapat menentukan berapa kali suatu model bekerja untuk mengolah data latih, satu *epoch* berarti setiap sampel dalam data latih memiliki kesempatan untuk memperbarui parameter model internal. Sedangkan untuk nilai *batch size* merupakan jumlah berapa banyak sampel dalam data latih yang akan digunakan dalam satu iterasi. Penulis akan melakukan penelitian dengan dua fungsi optimasi yang berbeda yaitu *Adam Optimizer* dan *RMSprop Optimizer*. Penulis akan melakukan percobaan secara repetitif hingga mendapatkan hasil yang maksimal dan menghasilkan pembanding diantara kedua fungsi optimasi tersebut.

2.5. *Adam Optimizer*

Adaptive Moment Estimation (Adam) merupakan salah satu algoritma yang dapat menggantikan prosedur *stochastic gradient descent* klasik untuk memperbaharui *weight network* berdasarkan data *training* secara iteratif. Cara kerja *Adam* dapat digambarkan sebagai penggabungan sifat terbaik dari dua ekstensi *stochastic gradient descent* yaitu *adaptive gradient algorithm* dan *root mean square propagation* dengan penggabungan tersebut *Adam* mampu memberikan pengoptimalan suatu algoritma yang mampu menangani *sparse gradients* pada *noisy problem.* [19]. Dengan penggunaan teknik optimasi yang mampu menurunkan gradien, metode ini sangat efisien ketika bekerja pada data yang besar dan parameter yang besar. Sebelum menggunakan fungsi optimasi *Adam*, terdapat beberapa nilai yang harus didefinisikan terlebih dahulu, yaitu:

1. m = 0
2. v = 0
3. ϵ = 10^{-8}
4. t = 0
5. α = 0.001
6. β_1 = 0.9
7. β_2 = 0.999

Tahap-tahap yang dilakukan oleh *Adam Optimizer* yaitu:

1. Tambah t setiap iterasi
$$t = t + 1 \quad (1)$$
2. Menghitung *gradien*
$$gt \leftarrow \nabla\theta ft(\theta t - 1) \quad (2)$$
3. Menghitung bias *first moment*
$$mt \leftarrow \beta 1 mt - 1 + (1 - \beta 1)g \quad (3)$$
4. Menghitung bias *second moment*
$$vt \leftarrow \beta 2 vt - 1 + (1 - \beta 2)g \cdot g \quad (4)$$
5. Memperbaiki bias *first moment*
$$\hat{mt} \leftarrow \frac{m_t}{1-\beta_1^t} \quad (5)$$
6. Memperbaiki bias *second moment*
$$\hat{vt} \leftarrow \frac{v_t}{1-\beta_2^t} \quad (6)$$
7. Memperbaiki parameter
$$\theta t = \theta t - 1 - \alpha \frac{\hat{mt}}{\sqrt{\hat{vt}}+\epsilon} \quad (7)$$

Keterangan:
g = *gradient*
m = *first moment*
v = *second moment*
β_1, β_2 = *Exponential decay rates*
α = *Step size* atau *learning rate*
θ = parameter bobot yang akan diperbaiki

Ketujuh tahapan dilakukan secara berulang sebanyak jumlah *dataset* yang diambil secara acak hingga semua *epoch* selesai. Perbedaan antara *Adam* dengan *RMSProp* terletak pada perubahan *learning*

rate dimana *Adam* melakukan *bias correction* pada perhitungannya seperti pada tujuh langkah diatas [20].

2.6. *RMSprop Optimizer*

RMSprop memiliki kemiripan dengan *Adaprop*, yang merupakan bentuk improvisasi dari *Adagrad*. Gradien fungsi yang sangat kompleks seperti jaringan saraf memiliki kecenderungan untuk menghilang atau terjadinya masalah *vanishing gradient. RMSprop* merupakan fungsi optimasi yang memanfaatkan besarnya gradien terbaru untuk menormalkan gradien, fungsi ini mampu menjaga rata-rata bergerak di atas gradien *root mean square* sehingga disebut RMS. *RMSprop* merupakan salah satu fungsi optimasi yang mempertahankan rata-rata dari kuadrat gradien untuk setiap bobot, yang dapat dilihat seperti pada formula 8.

$$MeanSquare(w,t) = \rho * MeanSquare(w,t-1) + 0.1(\frac{\partial E}{\partial w}(t))^2 \quad (8)$$

Keterangan:
w = bobot
t = *timestep*
ρ = 0.9
$\frac{\partial E}{\partial w}$ = *gradient*

2.7. Evaluasi

Evaluasi merupakan tahapan yang digunakan untuk mengukur kinerja dari suatu model dari metode yang diusulkan. Evaluasi dilakukan dengan menghitung nilai *precision, recall, f-1 score,* dan akurasi.

$$Precision = \frac{TP}{TP + FP} x100 \quad (9)$$

$$Recall = \frac{TP}{TP + FN} x100 \quad (10)$$

$$F1 - Score = \frac{2\ x\ Recall\ x\ Precision}{Recall + Precision} \quad (11)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} x100 \quad (12)$$

Keterangan:
TP = *True Positive* (total prediksi benar dari data positif)
TN = *True Negative* (total prediksi benar dari data negatif)
FP = *False Positive* (total prediksi salah dari data negatif)
FN = *False Negative* (total prediksi salah dari data positif)

3. Hasil dan Pembahasan

Penulis melakukan *hyperparameter tuning* untuk mendapatkan parameter terbaik yang nilainya akan diterapkan pada model *deep learning* yang akan dibangun. Sebelum mengimplementasikan model *Long Short-Term Memory* (LSTM) penulis terlebih dahulu membagi 90% data *training* ke dalam tiap *fold* dengan metode validasi, yaitu *K-Fold Cross Validation* dengan nilai k=5, artinya dari total 90% data *training* akan dibagi menjadi 5. Terdapat dua model *deep learning* yang akan dikembangkan oleh peneliti, yaitu model yang menerapkan fungsi optimasi *Adam* dan model yang menerapkan fungsi optimasi *RMSprop.*

3.1. Analisa Parameter

Pada penelitian ini akan dilakukan pengujian *hyperparameter* yaitu *batch size* dan *epoch* dengan menggunakan metode *K-fold Cross Validation* untuk metode validasi dengan menerapkan nilai *fold* sebanyak lima, tujuannya untuk mendapatkan *hyperparameter* dengan performa terbaik. *Hyperparameter* yang akan diujikan pada proses *training* model untuk klasifikasi *sentiment* dari *review* film dapat dilihat pada Tabel 1. Nilai *epoch* dapat diatur ke nilai integer antara satu dan tak terhingga [21]. Sedangkan nilai *batch size* akan dimulai dari nilai 32, 64, 128, dan 126 [22], dengan memperhatikan pada kasus yang dikerjakan dengan melakukan percobaan.

p-ISSN: 2301-5373
e-ISSN: 2654-5101

Tabel 1. *Hyperparameter* untuk Skenario Pengujian

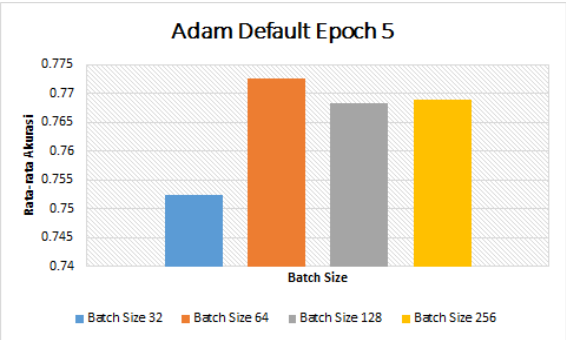
No	Parameter	Ukuran
1	*Batch size*	[32, 64, 128, 256]
3	*Epoch*	[3, 5, 10, 15, 20]

Pada proses ini peneliti menerapkan beberapa parameter yang akan diujikan untuk membandingkan fungsi optimasi *Adam* dan *RMSprop*. *Hyperparameter* yang akan diujikan yaitu nilai *batch size* dan *epoch* yang berbeda-beda. Pengujian yang akan dilakukan yaitu dengan mencari nilai *batch size* yang terbaik terlebih dahulu, nilai *batch size* yang akan diujikan yaitu 32, 64, 128, dan 256. Melalui 5-*fold Cross Validation* penentuan nilai terbaik *batch size* diambil dari rata-rata akurasi dari data validasi. Setelah mendapatkan nilai *batch size* yang terbaik, nilai tersebut akan digunakan sebagai nilai *batch size* pada pengujian untuk mencari nilai *epoch* yang terbaik. Nilai *epoch* yang akan diujikan yaitu, 3, 5, 10, 15, dan 20. Melalui 5-*fold Cross Validation* penentuan nilai terbaik *epoch* diambil dari rata-rata akurasi dari data validasi, lalu selanjutnya model tersebut akan diujikan dengan data *testing* yang sebelumnya belum pernah dikenali oleh sistem. Secara lebih jelas proses tuning *hyperparameter* dipaparkan pada materi di bawah ini.

a. Pengujian pada *Adam Optimizer*
Pengujian yang dilakukan dengan *Adam Optimizer* akan dilakukan dengan parameter-parameter *batch size* dan *epoch*. Hal yang pertama dilakukan adalah mencari nilai *batch size* yang terbaik, untuk mendapatkan nilai tersebut peneliti melakukan pengujian dengan membandingkan nilai *batch size* 32, 64, 128, dan 256 pada *default epoch* sebesar 5 *epoch.* Pengujian dengan *Adam Optimizer* untuk mendapatkan nilai *batch size* yang terbaik dapat dilihat pada Tabel 2 dan Gambar 4.

Tabel 2. Pengujian *Adam Optimizer* Untuk Mencari Nilai *Batch Size* Terbaik

Batch Size	Epoch	Fold					Mean
		1	2	3	4	5	
32	5	0.77381	0.756349	0.72619	0.751587	0.753968	0.752381
64	5	0.786111	0.776587	0.775794	0.763889	0.761111	0.772698
128	5	0.752381	0.775	0.769444	0.761508	0.782937	0.768254
256	5	0.78254	0.767857	0.759127	0.763889	0.771032	0.768889



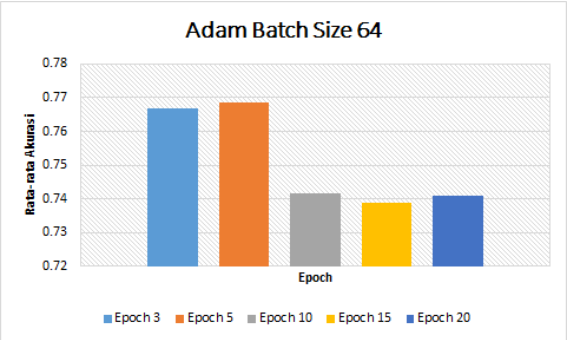
Gambar 4. Grafik Hasil Uji *Adam Optimizer* Dengan *Default Epoch*

Berdasarkan hasil pengujian pada Tabel 2 dan Gambar 4, dengan menerapkan 5-*fold Cross Validation* nilai *batch size* 64 memberikan nilai rata-rata akurasi yang terbaik dibandingkan nilai *batch size* yang lainnya, yaitu sebesar 0.772698.

Nilai *batch size* 64 akan dijadikan nilai *default batch size* untuk diujikan pada nilai *epoch* yang berbeda-beda yaitu 3, 5, 10, 15, 20. Pengujian dengan *Adam Optimizer* terhadap nilai *epoch* yang berbeda-beda dapat dilihat pada Tabel 3 dan Gambar 5.

Tabel 3. Pengujian *Adam Optimizer* Dengan Nilai *Epoch*

Batch Size	Epoch	Fold					Mean
		1	2	3	4	5	
64	3	0.793651	0.753968	0.764683	0.774206	0.756746	0.766667
64	5	0.759524	0.768651	0.768651	0.75873	0.777778	0.768651
64	10	0.722619	0.757143	0.736905	0.743254	0.74881	0.741746
64	15	0.738889	0.740079	0.741667	0.721825	0.751984	0.738889
64	20	0.742063	0.734921	0.736905	0.739286	0.751587	0.740952



Gambar 5. Grafik Hasil Uji *Adam Optimizer* Dengan *Batch Size* Terbaik

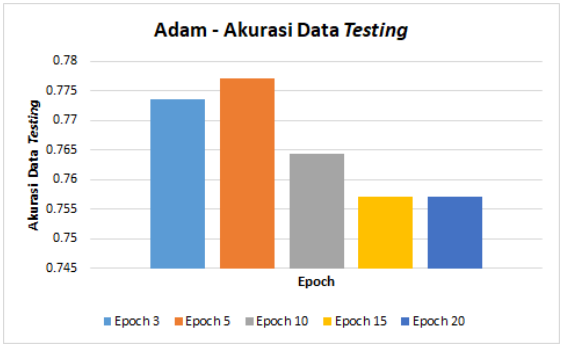
Berdasarkan hasil pengujian pada Tabel 3 dan Gambar 5, dengan menerapkan 5-*fold Cross Validation* nilai *batch size* 64 dan *epoch* 5 memberikan nilai rata-rata akurasi yang terbaik dibandingkan nilai yang lainnya, yaitu sebesar 0.768651.

Berdasarkan hasil pengujian, model dari *Adam Optimizer* yang mampu memberikan nilai rata-rata akurasi yang terbaik akan diujikan pada *data testing* seperti pada Tabel 4 dan Gambar 6.

Tabel 4. Pengujian Parameter Terbaik Pada Data *Testing Adam Optimizer*

Optimizer	Batch Size	Epoch	Acc Test
Adam	64	3	0.7736
	64	5	0.7771
	64	10	0.7643
	64	15	0.7571
	64	20	0.7571

p-ISSN: 2301-5373
e-ISSN: 2654-5101



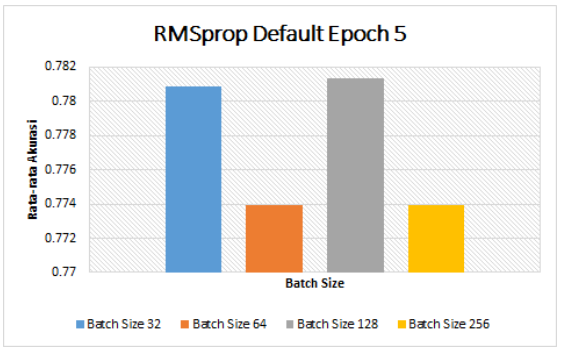
Gambar 6. Grafik Hasil Uji *Adam Optimizer* Dengan Data *Testing*

Berdasarkan hasil pada Tabel 4 dan Gambar 6, *Adam Optimizer* mampu memberikan nilai optimal pada *batch size* 64 dan *epoch* 5 dengan nilai akurasi pada data *testing* sebesar 0.7771.

b. Pengujian pada *RMSprop Optimizer*
Pengujian yang dilakukan dengan *RMSprop Optimizer* akan dilakukan dengan parameter-parameter *batch size* dan *epoch*. Hal yang pertama dilakukan adalah mencari nilai *batch size* yang terbaik, untuk mendapatkan nilai tersebut peneliti melakukan pengujian dengan membandingkan nilai *batch size* 32, 64, 128, dan 256 pada *default epoch* sebesar 5 *epoch.* Pengujian dengan *RMSprop Optimizer* untuk mendapatkan nilai *batch size* yang terbaik dapat dilihat pada Tabel 5 dan Gambar 7.

Tabel 5. Pengujian *RMSprop Optimizer* Untuk Mencari Nilai *Batch Size* Terbaik

Batch Size	Epoch	Fold					Mean
		1	2	3	4	5	
32	5	0.794444	0.780952	0.772222	0.778175	0.778571	0.780873
64	5	0.79127	0.765873	0.756746	0.779365	0.776587	0.773968
128	5	0.803968	0.788889	0.762698	0.773016	0.778175	0.781349
256	5	0.782937	0.772222	0.754762	0.778571	0.781349	0.773968



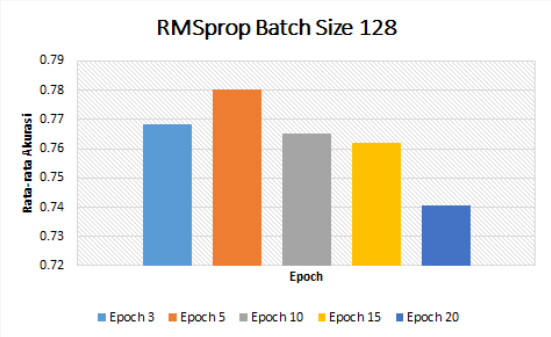
Gambar 7. Grafik Hasil Uji *RMSprop Optimizer* Dengan *Default Epoch*

Berdasarkan hasil pengujian pada Tabel 5 dan Gambar 7, dengan menerapkan 5-*fold Cross Validation* nilai *batch size* 128 memberikan nilai rata-rata akurasi yang terbaik dibandingkan nilai *batch size* yang lainnya, yaitu sebesar 0.781349.

Nilai *batch size* 128 akan dijadikan nilai *default batch size* untuk diujikan pada nilai *epoch* yang berbeda-beda yaitu 3, 5, 10, 15, 20. Pengujian dengan *RMSprop Optimizer* terhadap nilai *epoch* yang berbeda-beda dapat dilihat pada Tabel 6 dan Gambar 8.

Tabel 6. Pengujian *RMSprop Optimizer* Dengan Nilai *Epoch*

Batch Size	Epoch	Fold					Mean
		1	2	3	4	5	
128	3	0.786905	0.786508	0.755159	0.76627	0.747222	0.768413
128	5	0.784921	0.778968	0.780556	0.776984	0.780556	0.780397
128	10	0.778968	0.755159	0.755159	0.757937	0.778968	0.765238
128	15	0.766667	0.771429	0.756746	0.745238	0.770635	0.762143
128	20	0.727381	0.756349	0.738095	0.721429	0.759921	0.740635



Gambar 8. Grafik Hasil Uji *RMSprop Optimizer* Dengan *Batch Size* Terbaik

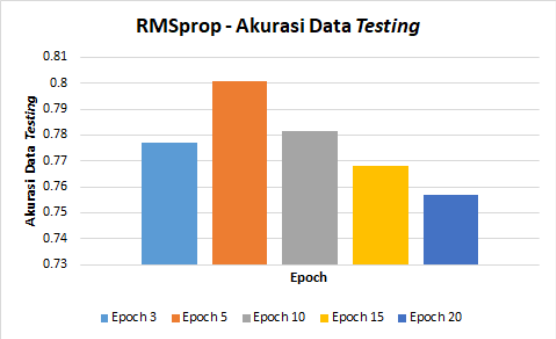
Berdasarkan hasil pengujian Tabel 6 dan Gambar 8, dengan menerapkan 5-*fold Cross Validation* nilai *batch size* 128 dan *epoch* 5 memberikan nilai rata-rata akurasi yang paling terbaik dibandingkan nilai yang lainnya, yaitu sebesar 0.780397.

Berdasarkan hasil pengujian, model dari *RMSprop Optimizer* yang mampu memberikan nilai rata-rata akurasi yang terbaik akan diujikan pada *data testing* seperti pada Tabel 7 dan Gambar 9.

Tabel 7. Pengujian Parameter Terbaik Pada Data *Testing RMSprop Optimizer*

Optimizer	Batch Size	Epoch	Acc Test
RMSprop	128	3	0.7771
	128	5	0.8007
	128	10	0.7814
	128	15	0.7679
	128	20	0.7571

p-ISSN: 2301-5373
e-ISSN: 2654-5101



Gambar 9. Grafik Hasil Uji *RMSprop Optimizer* Dengan *Data Testing*

Berdasarkan hasil pengujian pada Tabel 7 dan Gambar 9, *RMSprop Optimizer* mampu memberikan nilai optimal pada *batch size* 128 dan *epoch* 5 dengan nilai akurasi pada data *testing* sebesar 0.8007.

3.2. Evaluasi

Berikut merupakan hasil evaluasi pengujian dari model yang terbaik pada evaluasi LSTM menggunakan *Adam Optimizer* terhadap data *testing* dengan mendapatkan nilai akurasi, *precision*, *recall*, dan *f-1 score* yang dapat dilihat pada Tabel 8.

Tabel 8. Hasil *Confusion Matrix* Model *Adam Optimizer*

	Precision	Recall	F1-score
Negatif	0.76	0.79	0.78
Positif	0.79	0.76	0.78
Akurasi			0.78

Berikut merupakan hasil evaluasi pengujian dari model yang terbaik pada evaluasi LSTM menggunakan *RMSprop Optimizer* terhadap data *testing* dengan mendapatkan nilai akurasi, *precision*, *recall*, dan *f-1 score* yang dapat dilihat pada Tabel 9.

Tabel 9. Hasil *Confusion Matrix* Model *RMSprop Optimizer*

	Precision	Recall	F1-score
Negatif	0.77	0.85	0.81
Positif	0.84	0.75	0.79
Akurasi			0.80

4. Kesimpulan

Hasil tuning *hyperparameter* pada model LSTM untuk analisis sentimen *review* film menunjukkan bahwa model LSTM dengan *Adam Optimizer* dengan nilai *batch size* 64 dan nilai *epoch* 5 mampu memberikan performa yang terbaik dengan nilai akurasi sebesar 77,11%. Sedangkan model LSTM dengan *RMSprop Optimizer* dengan nilai *batch size* 128 dan nilai *epoch* 5 mampu memberikan performa yang terbaik dengan nilai akurasi sebesar 80,07%. Model LSTM dengan *Adam Optimizer* menunjukkan hasil yang baik yaitu dengan nilai *precision* pada sentimen negatif sebesar 76% dan pada sentimen positif sebesar 79%, nilai *recall* pada sentimen negatif sebesar 79% dan pada sentimen positif

sebesar 76%, nilai *f-1 score* pada sentimen negatif sebesar 78% dan pada sentimen positif sebesar 78%, serta nilai akurasi sebesar 77,11%. Sedangkan model LSTM dengan *RMSprop Optimizer* memberikan hasil yang lebih baik, dengan nilai *precision* pada sentimen negatif sebesar 77% dan pada sentimen positif sebesar 84%, nilai *recall* pada sentimen negatif sebesar 85% dan pada sentimen positif sebesar 75%, nilai *f-1 score* pada sentimen negatif sebesar 81% dan pada sentimen positif sebesar 79%, serta nilai akurasi sebesar 80,07%. Penerapan LSTM dengan membandingkan dua *optimizer* yaitu *Adam* dan *RMSprop* mampu dikembangkan berupa *website* analisis sentimen dimana *user* dapat memilih *optimizer* yang diinginkan serta *user* juga dapat mencari *review* film berdasarkan judulnya yang dimana *review* tersebut sudah diproses nilai prediksi sentimennya dengan model yang terbaik yaitu dengan penerapan *RMSprop Optimizer.*

Referensi

[1] CNN Indonesia, “Pandemi 2020 Buat Netflix Kebanjiran 36,6 Juta Pelanggan Baru,” 2021. https://www.cnnindonesia.com/hiburan/20210120132336-220-596129/pandemi-2020-buat-netflix-kebanjiran-366-juta-pelanggan-baru (accessed Jun. 10, 2021).
[2] N. K. Manaswi, *Deep Learning with Applications Using Python: Chatbots and Face, Object, and Speech Recognition With TensorFlow and Keras.*, 1 penyunt. India: Apress, 2018.
[3] A. Saxena and A. Sukumar, “Predicting bitcoin price using lstm And Compare its predictability with arima model,” *Int. J. Pure Appl. Math.*, 2018.
[4] J. Zhao, Z., Chen, W., Wu, X., Chen, P. C., & Liu, “LSTM network: a deep learning approach for short-term traffic forecast.,” *IET Intell. Transp. Syst.*, pp. 11(2), 68-75., 2017.
[5] A. Hassan and A. Mahmood, “Deep learning for sentence classification,” *2017 IEEE Long Isl. Syst. Appl. Technol. Conf. LISAT 2017*, 2017.
[6] Y. Widhiyasana, T. Semiawan, I. G. A. Mudzakir, and M. R. Noor, “Penerapan Convolutional Long Short-Term Memory untuk Klasifikasi Teks Berita Bahasa Indonesia,” *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 10, pp. 354–361, 2021.
[7] A. Barigidad and A. Mustafi, “IMDB Movie Reviews Dataset,” *IEEE Dataport*, 2020. https://doi.org/10.1109/ICCCS49678.2020.9276893.
[8] A. Anggito and J. Setiawan, *Metodologi Penelitian Kualitatif*. Jawa Barat: CV Jejak, 2018.
[9] K. D. Y. Wijaya and A. E. Karyawati, “The Effects of Different Kernels in SVM Sentiment Analysis on Mass Social Distancing,” *JELIKU*, vol. 9, pp. 161–168, 2020.
[10] A. Ranjan, “Data Cleaning in Natural Language Processing,” 2020. https://medium.com/analytics-vidhya/data-cleaning-in-natural-language-processing-1f77ec1f6406.
[11] I. G. C. P. Yasa, N. A. Sanjaya ER, and L. A. A. R. Putri, “Sentiment Analysis of SnackReview Using the Naïve Bayes Method,” *JELIKU*, vol. 8, pp. 333–338, 2020.
[12] M. Swamynathan, *Mastering Machine Learning with Python in Six Steps*. 2019.
[13] H. Manning, C., Raghavan, P. & Schütze, *An Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2009.
[14] Harshith, “Text Preprocessing in Natural Language Processing,” 2019. https://towardsdatascience.com/text-preprocessing-in-natural-language-processing-using-python-6113ff5decd8.
[15] D. Dahman, “Natural Language Processing,” 2021. https://medium.com/sysinfo/natural-language-processing-nlp-b54d6506efe2.
[16] F. Galbusera, “A Deep Learning Model for the Accurate and Reliable Classification of Disc Degeneration Based on MRI Data,” *Invest. Radiol.*, vol. 56, no. 2, pp. 78–85, 2021, [Online]. Available: https://doi.org/10.1097/RLI.0000000000000709.
[17] J. Brownlee, “A Gentle Introduction to Long Short-Term Memory Networks by the Experts,” 2017.
[18] Keras, “Layer Weight Regularizers,” 2022. https://keras.io/api/layers/regularizers/.
[19] D. P. Kingma and J. L. Ba, *A Method For Stochastic Optimization*. ICLR, 2015.
[20] J. Brownlee, “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning,” 2017. https://machinelearningmastery.com/adam- optimization-algorithm-for-deep-learning/.
[21] J. Brownlee, “Keras: Difference Between a Batch and an Epoch,” 2018. https://lms.onnocenter.or.id/wiki/index.php/Keras:_Difference_Between_a_Batch_and_an_Epoch#:~:text=batch dan epoch.-,Perbedaan Batch dan Epoch%3F,jumlah sampel dalam dataset training.
[22] A. Khumaidi, “Pengujian Algoritma LSTM untuk Prediksi Kualitas Udara dan Suhu Kota Bandung,” *Telematika*, vol. 15, 2021.