Terbit *online* pada laman : http://teknosi.fti.unand.ac.id/

Jurnal Nasional Teknologi dan Sistem Informasi

| ISSN (Print) 2460-3465 | ISSN (Online) 2476-8812 |

Artikel Penelitian

Deteksi Stres Teks Percakapan Digital Menggunakan Model LSTM

Agni Musadad [1,*], *Heni Sulastri* [2]

[1,2] *Sistem Informasi, Fakultas Teknik, Universitas Siliwangi, Indonesia*

INFORMASI ARTIKEL

Sejarah Artikel:
Diterima Redaksi: 13 November 2025
Revisi Akhir: 08 Mei 2026
Diterbitkan *Online*: 12 Mei 2026

KATA KUNCI

Deteksi Stres,
Long Short-Term Memory (LSTM),
Pemrosesan Bahasa Alami,
Kesehatan Mental,
Analisis Hyperparameter

KORESPONDENSI

E-mail: 227007090@student.unsil.ac.id

A B S T R A C T

Deteksi stres dari teks percakapan digital menjadi krusial untuk kesehatan mental, namun penelitian pada Bahasa Indonesia masih terbatas. Penelitian ini merancang dan mengevaluasi model deep learning berbasis *Long Short-Term Memory* (LSTM) untuk mengklasifikasikan teks berbahasa Indonesia sebagai stres atau non-stres. Model dilatih dan diuji menggunakan dataset 11.000 sampel teks berlabel. Metodologi mencakup pra-pemrosesan teks, pelatihan model, dan analisis sensitivitas *hyperparameter* (*learning rate*, *batch size*, unit LSTM) untuk menemukan konfigurasi optimal. Model yang diusulkan mencapai performa kuat dengan akurasi 86,48% serta F1-Score yang seimbang (0,87 untuk non-stres dan 0,86 untuk stres), mengungguli beberapa *baseline* penelitian sebelumnya. Analisis kurva pelatihan mengidentifikasi adanya *overfitting* yang jelas. Selain itu, analisis sensitivitas *hyperparameter* mengungkap temuan penting: konfigurasi model utama (64 unit LSTM) bersifat sub-optimal, dan performa dapat ditingkatkan lebih lanjut menggunakan 128 unit LSTM atau *batch size* 128. Penelitian ini mengonfirmasi efektivitas LSTM untuk teks stres berbahasa Indonesia, sekaligus menunjukkan adanya ruang signifikan untuk optimasi *hyperparameter* dan perlunya teknik penanganan *overfitting* yang lebih kuat.

1. PENDAHULUAN

Seiring meningkatnya intensitas komunikasi digital melalui aplikasi pesan singkat dan media sosial, data tekstual menjadi sumber informasi yang kaya untuk menganalisis berbagai aspek kehidupan manusia, termasuk kondisi psikologis. Perkembangan pesat dalam bidang kecerdasan buatan (*Artificial Intelligence*/AI), khususnya pemrosesan bahasa alami (*Natural Language Processing*/NLP), menawarkan metode inovatif untuk mendeteksi kondisi kesehatan mental [1], seperti tingkat stres, secara non-intrusif dari interaksi digital. Namun, sebagian besar penelitian masih berfokus pada data berbahasa Inggris, sementara penerapan pada teks berbahasa Indonesia masih terbatas. Deteksi dini tingkat stres menjadi krusial untuk mencegah dampak negatif yang lebih serius terhadap kesehatan fisik dan mental individu.

Beberapa penelitian dalam lima tahun terakhir telah menunjukkan potensi besar dari penerapan *machine learning* dan *deep learning* dalam analisis kesehatan mental. Zhang et al. [1] dalam tinjauan naratif mereka mengonfirmasi bahwa teknik NLP semakin banyak diterapkan untuk deteksi berbagai penyakit mental melalui analisis teks. Algoritma seperti *Long Short-Term Memory* (LSTM) terbukti efektif karena kemampuannya dalam menangani data sekuensial dan mengenali pola jangka panjang dalam teks [2], yang sangat penting dalam analisis sentimen dan emosi [3] [4] [5]. Penelitian oleh Lisanthoni et al. [6] juga berhasil mengimplementasikan LSTM untuk analisis sentimen pada platform digital di Indonesia dengan akurasi mencapai 92%, menunjukkan relevansi dan potensi metode ini dalam pengolahan data berbahasa Indonesia.

Penelitian terkait deteksi stres dari teks media sosial telah menjadi fokus yang berkembang pesat. Berbagai penelitian telah mengeksplorasi penggunaan dataset publik, terutama Dreaddit, yang merupakan korpus dari platform Reddit yang dirancang khusus untuk analisis stres [7]. Pendekatan untuk menganalisis

https://doi.org/10.25077/TEKNOSI.v12i1.2026.152-159

dataset ini pun beragam, dengan beberapa penelitian berfokus pada model *machine learning* klasik.

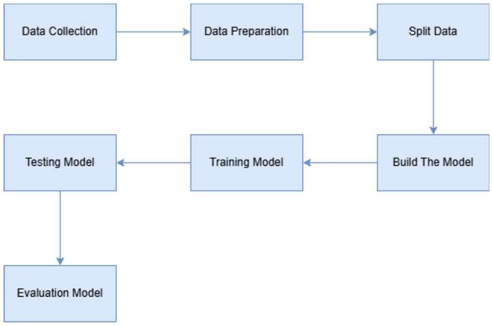
Beberapa penelitian berfokus pada model *machine learning* klasik. Sebagai contoh, Inamdar et al. [8] membandingkan berbagai algoritma seperti *Logistic Regression* (LR), SVM, dan XGBoost, dan menemukan bahwa LR yang dikombinasikan dengan *embedding* ELMo memberikan F1-score tertinggi di angka 0,76. Di sisi lain, penelitian yang lebih modern mengeksplorasi arsitektur *deep learning* yang lebih kompleks. Wan & Tian [9] mengusulkan model *Bidirectional Long Short-Term Memory* (Bi-LSTM) yang dilengkapi dengan *attention mechanism*, dan berhasil mencapai F1-score sebesar 81,21%.

Selain itu, model berbasis Transformer seperti BERT dan variannya (misal, MentalBERT) juga telah dieksplorasi secara ekstensif, menunjukkan potensi besar dalam menangkap nuansa bahasa yang terkait dengan kondisi kesehatan mental [7]. Nijhawan et al. [10] juga menggunakan BERT untuk klasifikasi emosi sebagai dasar deteksi stres dari interaksi sosial. Penelitian-penelitian ini memberikan landasan yang kuat dan tolok ukur untuk mengevaluasi efektivitas model LSTM yang diusulkan dalam konteks dataset berbahasa Indonesia.

Meskipun banyak penelitian telah dilakukan tantangan tetap ada, terutama dalam menangani dataset yang relatif kecil yang rentan terhadap *overfitting* serta dalam memastikan interpretabilitas model untuk aplikasi klinis. Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk merancang dan mengimplementasikan model LSTM untuk mengklasifikasikan tingkat stres dari teks percakapan berbahasa Indonesia. Kontribusi utama dari penelitian ini adalah penerapan sistematis dari tahap pengumpulan data hingga evaluasi model, disertai analisis mendalam terhadap tantangan yang muncul, seperti risiko *overfitting* pada dataset berbahasa Indonesia yang terbatas.

2. METODE

Metodologi penelitian ini dirancang secara sistematis melalui beberapa tahapan utama yang digambarkan pada Gambar 1. Alur kerja ini mencakup pengumpulan data, pra-pemrosesan, pembagian data, perancangan arsitektur model, pelatihan, hingga evaluasi performa model.



Gambar 1. Diagram Alur Proses *Machine learning.*

2.1. Pengumpulan dan Pembagian Data

Penelitian ini menggunakan dataset sekunder yang dikumpulkan dari repositori publik seperti Kaggle dan GitHub. Dataset ini berisi sekitar 11.000 sampel teks percakapan anonim dalam Bahasa Indonesia. Setiap sampel teks telah diberi label "Positive" (tidak stres) dan "Negative" (stres) seperti yang tertuang dalam tabel berikut ini.

Tabel 1. Contoh data teks berlabel Positif

Text	Label
Barang sudah diterima ini kak, makasih yaa	Positive
Gampang dibawa-bawa, terlalu imut ukurannya	Positive
LANGGANAN ??????????	Positive
bagus, pengiriman cepet banget bakal jadi lang...	Positive
Kartu bekerja dengan baik begitupun sellernya ...	Positive

Tabel 2. Contoh data teks berlabel Negatif

Text	Label
Kecemasan saya memberitahu saya untuk tidak ju...	Negative
Khawatir saya menderita kanker ovarium. Semakin...	Negative
Untuk Penderita HA ang memiliki kecemasan yan...	Negative
Eye floaters karena stres? Halo, Saya baru-bar...	Negative
Ada rasa cemas berlebihan saat ini, .Dan, masi...	Negative

Dataset kemudian dibagi (split) menjadi 80% data latih (*training data*) dan 20% data uji (*testing data*). Alokasi 80% untuk data latih dipilih untuk memaksimalkan jumlah sampel bagi model *deep learning* (LSTM) agar dapat mempelajari pola dan konteks secara efektif. Sisa 20% data uji digunakan sebagai data independen yang tidak pernah dilihat model selama pelatihan, sehingga dapat memberikan evaluasi yang objektif terhadap kemampuan generalisasi model . Selama proses pelatihan, 20% dari data latih (atau 16% dari total dataset) dialokasikan sebagai data validasi (*validation data*).

2.2. Pra-pemrosesan Data

Data teks mentah yang telah dikumpulkan kemudian melalui tahapan pra-pemrosesan data (*data preparation*). Tahapan ini krusial untuk membersihkan dan mentransformasi data menjadi format yang siap diolah oleh model. Proses yang dilakukan konsisten dengan praktik standar dalam NLP untuk teks Bahasa Indonesia, yang meliputi beberapa langkah kunci sebagai berikut [8].

2.2.1. *Case Folding*

Tahap *case folding* adalah proses awal untuk menyeragamkan teks dengan mengubah seluruh huruf menjadi huruf kecil (*lowercase*) [8]. Langkah ini penting untuk memastikan bahwa model tidak memperlakukan kata yang sama secara berbeda hanya karena perbedaan kapitalisasi (misalnya, memperlakukan "stres" dan "STRES" sebagai token yang identik).

2.2.2. *Punctuation & Stopword Removal*

Selanjutnya, dilakukan pembersihan *noise* dari data teks. Tanda baca (seperti titik, koma, tanda tanya) dihapus karena umumnya tidak membawa makna semantik yang signifikan untuk tugas klasifikasi ini. Selain itu, *stopwords* kata-kata umum yang sering muncul namun memiliki sedikit nilai informasional (seperti "yang", "di", "dan", "dengan") juga disaring dan dihilangkan [8]. Proses ini membantu model untuk lebih fokus pada kata-kata yang kaya akan konten dan signifikan dalam menentukan sentimen stres.

2.2.3. *Tokenization*

Setelah teks dibersihkan, proses *tokenization* diterapkan. *Tokenization* adalah proses memecah kalimat atau teks yang utuh menjadi unit-unit individual yang lebih kecil, yang disebut sebagai 'token' (biasanya per kata) [8]. Ini adalah langkah fundamental untuk mengubah data teks non-terstruktur menjadi format terstruktur yang dapat diolah oleh model sekuensial.

2.2.4. *Sequencing dan Padding*

Model *deep learning* tidak dapat memproses kata-kata secara langsung; model tersebut memproses angka. Oleh karena itu, setiap token unik dalam korpus data dipetakan ke sebuah nilai integer (angka) unik, proses ini disebut *sequencing*. Karena model LSTM memerlukan *input* dengan panjang yang seragam, setiap sekuens angka kemudian diseragamkan panjangnya melalui proses *padding*, di mana sekuens yang lebih pendek akan ditambahkan nilai "0" hingga mencapai panjang yang ditentukan, sementara sekuens yang lebih panjang akan dipotong [8].

2.3. *Arsitektur Model*

Model klasifikasi dalam penelitian ini dibangun menggunakan arsitektur *Long Short-Term Memory* (LSTM). LSTM dipilih karena merupakan pengembangan dari *Recurrent Neural Network* (RNN) yang dirancang khusus untuk mengatasi masalah *vanishing gradient* dan mampu mempelajari dependensi jangka panjang dalam data sekuensial [11]. Kemampuan ini sangat cocok untuk memproses data teks percakapan, di mana konteks dan urutan makna antar kata sangat penting.

Arsitektur model yang digunakan (diringkas pada Tabel 3) dibangun menggunakan *library* TensorFlow/Keras dan terdiri dari empat lapisan utama yang dijelaskan di bawah ini.

Tabel 3. Ringkasan Arsitektur Model LSTM

Model: "sequential"		
Layer (type)	Output Shape	Param #
Embedding	(None, None, 128)	640000
LSTM	(None, 64)	49408
Dropout	(None, 64)	0
Dense	(None, 1)	65
Total params: 689,473		
Trainable params: 689,473		
Non-trainable params: 0		

2.3.1. *Embedding Layer*

Lapisan ini merupakan lapisan pertama dalam model. Fungsinya adalah untuk mentransformasikan sekuens integer (yang mewakili kata) dari data *input* menjadi representasi vektor yang padat (*dense vector*). Dalam penelitian ini, digunakan dimensi *embedding* sebesar 128. Lapisan ini memungkinkan model untuk mempelajari representasi semantik kata, di mana kata-kata dengan makna serupa akan memiliki vektor yang mirip [12].

2.3.2. *LSTM Layer*

Lapisan ini adalah inti dari arsitektur model. Lapisan LSTM (dengan 64 unit) memproses sekuens vektor yang dihasilkan oleh *Embedding Layer*. Tidak seperti RNN standar yang rentan terhadap *vanishing gradient*, LSTM dirancang khusus untuk mengatasi masalah ini dengan memperkenalkan "memori sel" (*cell state*) yang diatur oleh tiga gerbang (*gates*) utama [2].

1. *Forget Gate*: Gerbang pertama ini memutuskan informasi apa yang akan dibuang dari *cell state* sebelumnya. Ini penting karena memungkinkan model untuk "melupakan" konteks yang tidak lagi relevan (misalnya, topik percakapan yang sudah berganti) dan hanya menyimpan informasi yang penting.
2. *Input Gate*: Gerbang ini memutuskan informasi baru apa dari *input* saat ini yang akan disimpan ke dalam *cell state*. Ini memungkinkan model untuk secara selektif memperbarui memorinya hanya dengan informasi yang dianggapnya signifikan dari kata-kata baru yang masuk.
3. *Output Gate*: Gerbang terakhir ini menentukan *output* (atau *hidden state*) dari sel LSTM berdasarkan *cell state* yang telah diperbarui.

Kombinasi gerbang-gerbang ini memungkinkan LSTM untuk menyimpan, memodifikasi, dan mengakses informasi dalam jangka waktu yang panjang. Kemampuan ini sangat krusial untuk menangkap konteks dan dependensi antar kata yang mungkin terpisah jauh dalam sebuah percakapan [12].

2.3.3. *Dropout Layer*

Sebuah lapisan *dropout* ditambahkan setelah lapisan LSTM. *Dropout* adalah teknik regularisasi yang fundamental untuk mencegah *overfitting* pada model *deep learning*. Selama pelatihan, lapisan ini secara acak menonaktifkan sebagian unit *neuron* (misalnya 20% atau 50%). Hal ini memaksa jaringan untuk belajar dengan cara yang lebih terdistribusi dan tidak terlalu bergantung pada neuron tertentu, sehingga meningkatkan kemampuan generalisasi model terhadap data baru [12].

https://doi.org/10.25077/TEKNOSI.v12i1.2026.152-159

2.3.4. *Dense Output Layer*

Lapisan terakhir adalah lapisan *Dense* (atau *fully connected*) yang berfungsi sebagai pengklasifikasi. Lapisan ini hanya memiliki satu unit *neuron* dan menggunakan fungsi aktivasi *sigmoid*. Fungsi *sigmoid* akan menghasilkan nilai keluaran berupa probabilitas antara 0 dan 1. Nilai ini merepresentasikan prediksi model, di mana nilai yang mendekati 1 mengindikasikan kelas 'Stres' (*Negative*) dan nilai yang mendekati 0 mengindikasikan kelas 'Non-Stres' (*Positive*) [12].

2.4. Metrik Evaluasi

Untuk mengevaluasi performa dan keandalan model yang telah dilatih pada data uji (tahap *Evaluation Model* pada Gambar 1), penelitian ini menggunakan metrik standar klasifikasi yang umum digunakan.

2.4.1. *Akurasi (Accuracy)*

Akurasi adalah metrik yang paling umum, mengukur rasio total prediksi yang benar (baik *True Positive* maupun *True Negative*) terhadap keseluruhan jumlah data uji. Meskipun mudah dipahami, akurasi dapat memberikan gambaran yang kurang tepat jika dataset yang digunakan tidak seimbang (*imbalanced*).

2.4.2. *Presisi (Precision)*

Presisi (Precision) Presisi mengukur proporsi prediksi positif ('Stres') yang diidentifikasi oleh model yang terbukti benar-benar positif. Metrik ini menjawab pertanyaan: "Dari semua teks yang diprediksi model sebagai 'Stres', berapa persen yang sebenarnya 'Stres'?". Presisi yang tinggi penting untuk meminimalkan *false positives*.

2.4.3. *Recall (Sensitivitas)*

Recall (atau sensitivitas) mengukur proporsi kasus positif ('Stres') aktual dalam dataset yang berhasil diidentifikasi dengan benar oleh model. Metrik ini menjawab pertanyaan: "Dari semua teks yang sebenarnya 'Stres', berapa persen yang berhasil ditemukan oleh model?". Recall yang tinggi sangat krusial dalam konteks kesehatan mental untuk meminimalkan *false negatives* (kasus stres yang terlewat).

2.4.4. *F1-Score*

F1-Score dihitung sebagai rata-rata harmonik (*harmonic mean*) dari Presisi dan Recall. Metrik ini memberikan satu angka tunggal yang menyeimbangkan kedua metrik tersebut. F1-Score sangat berguna untuk mengevaluasi kinerja model secara keseluruhan, terutama ketika terdapat pertukaran (*trade-off*) antara Presisi dan Recall atau ketika distribusi kelas dalam dataset tidak seimbang [8].

2.5. Eksperimen Hyperparameter

Selain perancangan model utama, penelitian ini juga melakukan analisis sensitivitas *hyperparameter*. Tujuan dari eksperimen ini adalah untuk memberikan justifikasi empiris terhadap pilihan arsitektur dan untuk memahami bagaimana parameter pelatihan kunci memengaruhi performa model. Pendekatan ini sejalan dengan praktik standar dalam penelitian *deep learning* untuk menemukan konfigurasi yang optimal dan memahami dampaknya terhadap performa.

Tiga *hyperparameter* kunci dievaluasi secara independen. Saat satu parameter diuji dalam rentang tertentu, dua parameter lainnya dijaga konstan pada nilai *baseline* model (*Learning rate*: 0.001, *Batch size*: 64, Unit LSTM: 64). Performa dari setiap konfigurasi model dievaluasi menggunakan metrik F1-Score (*Macro*) pada test set. Rincian parameter yang diuji adalah sebagai berikut.

2.5.1. *Learning rate*

Learning rate (tingkat pembelajaran) dapat dianggap sebagai *hyperparameter* paling kritis dalam melatih model *deep learning*, karena ia mengontrol seberapa besar bobot model diperbarui sebagai respons terhadap *error* yang dihitung. *Learning rate* yang terlalu tinggi dapat menyebabkan model "melompat-lompat" di sekitar titik optimal dan gagal konvergen. Sebaliknya, *learning rate* yang terlalu rendah dapat membuat proses pelatihan sangat lambat dan berisiko terjebak dalam *local minima*. Rentang pengujian ditetapkan pada [0.01, 0.005, 0.001, 0.0005] untuk mengidentifikasi "*sweet spot*" ini, mencakup nilai yang agresif, nilai *default* Adam, dan nilai yang lebih konservatif.

2.5.2. *Batch size*

Batch size menentukan jumlah sampel data yang diproses oleh model sebelum bobotnya diperbarui. Parameter ini memiliki dampak langsung pada stabilitas gradien dan generalisasi model. Nilai yang diuji adalah [32, 64, 128]. *Batch size* yang kecil (32) seringkali memberikan generalisasi yang lebih baik karena adanya "*noise*" dalam estimasi gradien yang membantu model keluar dari *local minima*. Namun, *batch size* yang lebih besar (128) menawarkan konvergensi yang lebih cepat dan estimasi gradien yang lebih stabil. Nilai 64 dipilih sebagai *baseline* yang umum digunakan.

2.5.3. *Jumlah Unit LSTM*

Jumlah unit dalam lapisan LSTM (seperti 32, 64, atau 128) secara langsung menentukan kapasitas atau kompleksitas model dalam mempelajari pola. Model dengan unit yang terlalu sedikit (misal, 32) mungkin mengalami *underfitting*, di mana ia gagal menangkap pola yang rumit dalam data teks. Sebaliknya, model dengan unit yang terlalu banyak (misal, 128) memiliki kapasitas tinggi namun sangat rentan terhadap *overfitting* (seperti yang sudah teridentifikasi pada Gambar 3), di mana ia "menghafal" data latih. Eksperimen ini bertujuan untuk menemukan keseimbangan optimal antara kapasitas model dan kemampuan generalisasinya.

3. HASIL

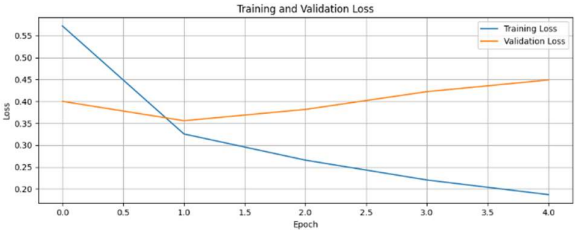
Bagian ini menyajikan hasil dari implementasi model yang telah dirancang. Penyajian hasil dijabarkan secara sistematis, sejalan dengan tahapan akhir pada alur metodologi penelitian (Gambar 1), yaitu mencakup hasil dari proses pelatihan (*Training Model*) serta hasil pengujian dan evaluasi akhir (*Testing Model* dan *Evaluation Model*).

3.1. *Hasil Pelatihan Model (Training Model)*

Proses pelatihan model (*Training Model*) dilakukan menggunakan data latih (64% dari total data) dan data validasi (16% dari total data). Pelatihan dijalankan menggunakan *optimizer* Adam dan dilengkapi dengan mekanisme *Early Stopping*. Mekanisme ini dirancang untuk menghentikan proses pelatihan secara otomatis jika metrik *validation loss* tidak menunjukkan perbaikan (penurunan) selama 3 *epoch* berturut-turut, sekaligus menyimpan bobot model terbaik dari *epoch* tersebut [5].

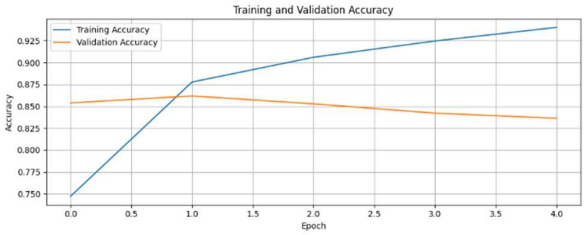
Selama proses pelatihan, metrik loss dan accuracy pada data latih dan data validasi dipantau secara ketat pada setiap *epoch* untuk mengevaluasi performa dan mendeteksi potensi *overfitting*. Visualisasi dari hasil pemantauan ini disajikan pada Gambar 2 dan Gambar 3.

Pada Gambar 2, disajikan kurva *Training and Validation Loss*. Terlihat jelas bahwa nilai *training loss* (garis biru) terus menurun secara konsisten dari *epoch* 0 hingga *epoch* 4, yang mengindikasikan bahwa model berhasil mempelajari pola dari data latih. Namun, pada saat yang sama, validation loss (garis oranye) mencapai titik terendahnya pada *epoch* 1 (sekitar 0.36) sebelum kemudian mulai meningkat pada *epoch-epoch* berikutnya. Fenomena ini memberikan indikasi kuat bahwa model mulai mengalami *overfitting*.



Gambar 2. Grafik Training dan Validation Loss

Indikasi *overfitting* tersebut diperkuat oleh kurva accuracy yang disajikan pada Gambar 3. *Training accuracy* (garis biru) terlihat terus meningkat secara signifikan, dari sekitar 75% hingga mencapai lebih dari 94%. Sebaliknya, *validation accuracy* (garis oranye) mencapai titik puncaknya pada *epoch* 1 (sekitar 86%), kemudian cenderung stagnan dan perlahan menurun pada *epoch* berikutnya . Karena mekanisme *Early Stopping* mengembalikan bobot terbaik, model yang digunakan untuk pengujian adalah model dari *epoch* 1, di mana *validation loss* berada di titik terendah.



Gambar 3. Grafik Training dan Validation Accuracy

3.2. *Hasil Pengujian dan Evaluasi (Testing & Evaluation Model)*

Setelah proses pelatihan selesai dan model terbaik (dari *epoch* 1) diperoleh, model tersebut diuji menggunakan 20% data uji (*test set*) yang telah disisihkan sejak awal (tahap *Testing Model*). Data ini sama sekali belum pernah dilihat oleh model selama proses pelatihan maupun validasi, sehingga dapat memberikan evaluasi yang objektif terhadap kemampuan generalisasi model.

Pada tahap pengujian akhir ini, model berhasil memperoleh akurasi sebesar 86,48%. Untuk analisis yang lebih mendalam (tahap *Evaluation Model*), performa model dianalisis menggunakan metrik Presisi, Recall, dan F1-Score untuk kedua kelas (0 untuk 'Non-Stres' dan 1 untuk 'Stres'). Hasil evaluasi rinci dari data uji disajikan pada Tabel 4.

Tabel 4. Hasil Evaluasi Performa Model

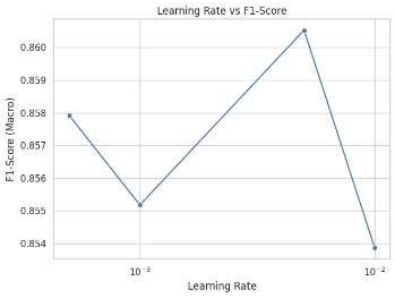
Label	Precision	Recall	F1-Score	Support
0 (Non-Stres)	0.84	0,90	0,87	1181
1 (Stres)	0,89	0,83	0,86	1171
Accuracy			0,86	2352
Macro Avg	0,87	0,86	0,86	2352
Weighted Avg	0,87	0,86	0,86	2352

Hasil pada Tabel 4 menunjukkan bahwa model memiliki performa yang sangat seimbang. F1-Score untuk kelas 'Non-Stres' (Label 0) mencapai 0,87, sementara untuk kelas 'Stres' (Label 1) mencapai 0,86. Keseimbangan ini—didukung oleh nilai Presisi (0,89 untuk Stres) dan Recall (0,90 untuk Non-Stres) yang kuat—mengonfirmasi bahwa model tidak bias terhadap salah satu kelas dan dapat diandalkan untuk mengidentifikasi kedua kondisi secara efektif.

3.3. *Analisis Sensitivitas Hyperparameter*

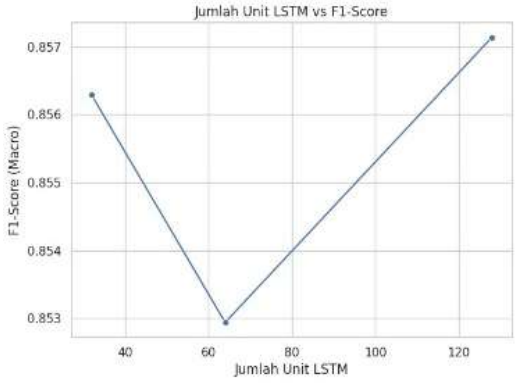
Hasil dari eksperimen analisis sensitivitas *hyperparameter* disajikan secara visual pada Gambar 4. Analisis ini memberikan wawasan penting tentang bagaimana setiap parameter kunci memengaruhi performa akhir model secara independen.

https://doi.org/10.25077/TEKNOSI.v12i1.2026.152-159



Gambar 4a. Hasil Analisis Sensitivitas *Hyperparameter* Learning Rate vs F1-Score

Gambar 4b. Hasil Analisis Sensitivitas *Hyperparameter* Batch Size vs F1-Score



Gambar 4c. Hasil Analisis Sensitivitas *Hyperparameter* Jumlah Unit LSTM vs F1-Score

3.3.1. Analisis Learning rate

Grafik pada Gambar 4a menunjukkan bahwa *learning rate* adalah parameter yang sangat sensitif. Performa model mencapai puncaknya (F1-Score ~0,861) pada *learning rate* 0.001 (10^{-3}). Menggunakan *learning rate* yang lebih tinggi (0.005 atau 0.01) atau lebih rendah (0.0005) secara signifikan menurunkan performa. Temuan ini memvalidasi penggunaan optimizer 'adam' dengan *learning rate* 0.001 sebagai pilihan yang optimal untuk model ini.

3.3.2. Analisis Batch size

Grafik pada Gambar 4b menunjukkan tren linear yang jelas dan signifikan: performa F1-Score meningkat secara konsisten seiring dengan meningkatnya *batch size*. Performa terendah (F1 ~0.854) diperoleh pada *batch size* 32, dan performa tertinggi (F1 > 0,860) dicapai pada *batch size* 128. Ini mengindikasikan bahwa *batch size* 64 yang digunakan pada model utama (F1-Score ~0.857) adalah performa yang baik, namun sub-optimal. Peningkatan performa dengan *batch size* yang lebih besar menunjukkan bahwa gradien yang lebih stabil dari batch yang lebih besar lebih bermanfaat bagi model ini daripada "noise" yang dihasilkan oleh batch yang lebih kecil.

3.3.3. Analisis Jumlah Unit LSTM

Grafik pada Gambar 4c menunjukkan temuan yang paling menarik dan non-linear. Berbeda dengan ekspektasi, performa terendah (F1 ~0.853) secara tak terduga terjadi saat menggunakan 64 unit (konfigurasi model utama). Sebaliknya, model dengan arsitektur yang lebih sederhana (32 unit, F1 ~0.856) dan arsitektur yang lebih kompleks (128 unit, F1 > 0.857) keduanya menghasilkan performa yang sedikit lebih baik. Temuan ini mengindikasikan bahwa 64 unit mungkin bukan merupakan konfigurasi optimal, dan berpotensi terjebak dalam *local minima* selama pelatihan. Hasil ini juga menunjukkan bahwa arsitektur yang lebih sederhana (32 unit) mungkin sudah cukup efisien, sementara arsitektur yang lebih kompleks (128 unit) mampu menangkap pola data dengan lebih baik dan masih dapat digeneralisasi.

4. PEMBAHASAN

Hasil akurasi pengujian sebesar 86,48% menunjukkan bahwa model LSTM yang dikembangkan memiliki kemampuan yang kuat dalam melakukan generalisasi dan mengklasifikasikan teks percakapan yang belum pernah dilihat sebelumnya. Pencapaian ini sejalan dengan penelitian-penelitian sebelumnya yang juga menunjukkan efektivitas LSTM untuk tugas analisis sentimen dan emosi dalam teks, baik secara global maupun dalam konteks Bahasa Indonesia [3] [13] [4].

Pencapaian ini menjadi lebih signifikan jika dibandingkan secara langsung dengan penelitian lain yang menggunakan dataset Dreaddit (Reddit) untuk deteksi stres. Sebagai contoh, F1-score 0,86 yang diperoleh model ini (Tabel 2) secara substansial melampaui F1-score 81.21% yang dilaporkan oleh Wan & Tian [9], yang menggunakan arsitektur Bi-LSTM dengan *Attention mechanism*. Kinerja ini juga jauh mengungguli model *machine learning* klasik, seperti yang ditunjukkan oleh Inamdar et al. [8] di mana F1-score tertinggi yang dicapai hanya 0.76. Perbandingan ini mengindikasikan bahwa arsitektur LSTM yang diimplementasikan dalam penelitian ini merupakan pilihan yang sangat solid dan efektif untuk konteks data berbahasa Indonesia.

Meskipun akurasi keseluruhan tergolong tinggi, analisis mendalam terhadap model mengungkap dua tantangan utama. Tantangan pertama adalah indikasi *overfitting* yang jelas [14], dimana performa pada data latih terus meningkat sementara performa pada data validasi menurun setelah *epoch* pertama (Gambar 2 dan 3).adalah temuan yang signifikan. Fenomena ini sejalan dengan tantangan umum dalam *deep learning*, dimana Ilias et al. [7] juga menyoroti pentingnya kalibrasi model (model *calibration*) untuk menghindari kepercayaan berlebih (*overconfidence*) Hal ini umum terjadi pada model *deep learning* yang kompleks dan menunjukkan bahwa model mulai "menghafal" data latih alih-alih mempelajari pola yang dapat digeneralisasi.

Tantangan kedua, yang terungkap melalui analisis sensitivitas *hyperparameter* (Gambar 4), adalah bahwa konfigurasi model utama (64 unit LSTM, *batch size* 64) ternyata bersifat sub-optimal. Temuan pada Gambar 4 (kanan) secara tak terduga menunjukkan bahwa F1-Score untuk 64 unit (~0.853) lebih rendah daripada arsitektur yang lebih sederhana (32 unit, ~0.856) dan arsitektur yang lebih kompleks (128 unit, ~0.857). Tren serupa terlihat pada Gambar 4 (tengah), di mana *batch size* 128 (F1 > 0,860) menghasilkan performa lebih baik daripada *batch size* 64 (F1 ~0.857). Ini mengindikasikan bahwa meskipun model utama berkinerja baik, terdapat ruang signifikan untuk optimasi lebih lanjut.

Di sisi lain, terlepas dari tantangan *overfitting* dan sub-optimalitas tersebut, hasil pada Tabel 2 menunjukkan kinerja model yang sangat baik dan seimbang dalam mengklasifikasikan kedua kelas. Model mampu mengidentifikasi kelas "Non-Stres" (Label 0) dengan recall 0,90 dan F1-score 0,87, serta kelas "Stres" (Label 1) dengan recall 0,83 dan F1-score 0,86.

Nilai precision yang tinggi untuk kelas "Stres" (0,89) patut diperhatikan; ini menunjukkan bahwa ketika model memprediksi seseorang mengalami stres, prediksi tersebut memiliki tingkat kebenaran yang tinggi. Keseimbangan F1-score antara kedua kelas (0,87 dan 0,86) mengonfirmasi bahwa model tidak lagi bias terhadap satu kelas tertentu dan dapat mengidentifikasi kedua kondisi secara efektif. Temuan ini menggarisbawahi bahwa akurasi saja tidak cukup, dan analisis metrik seperti precision dan recall (seperti pada Tabel 2) sangat penting untuk memvalidasi kemampuan model dalam kasus penggunaan di dunia nyata [15].

4.1. Keterbatasan Penelitian dan Analisis Kelemahan

Meskipun model yang diusulkan mencapai akurasi yang kuat, penelitian ini memiliki beberapa keterbatasan yang perlu dibahas untuk memberikan konteks pada hasil.

Pertama, terkait kompleksitas bahasa percakapan digital. Model LSTM, meskipun baik dalam menangani sekuens, mungkin masih kesulitan dengan aspek linguistik yang lebih rumit seperti sarkasme atau makna ganda (ambiguitas). Sebuah teks bisa saja mengandung kata-kata negatif ("marah", "benci") namun digunakan dalam konteks bercanda (non-stres), yang dapat menyebabkan *False Positive*. Wimbassa et al. [3] juga menyoroti masalah ini, di mana "teks yang sama dapat mengandung banyak emosi yang berbeda serta kata-kata yang digunakan dapat memiliki makna ganda".

Kedua, tantangan dalam mendeteksi stres yang implisit. *False Negatives* (kasus stres yang gagal dideteksi) kemungkinan besar terjadi ketika pengguna mengekspresikan stres secara halus tanpa menggunakan kata-kata pemicu yang jelas seperti "cemas" atau "sakit". Frasa seperti "sudah tidak tahu harus bagaimana lagi" atau "lelah dengan semuanya" menunjukkan tekanan psikologis secara implisit, yang mungkin sulit ditangkap oleh model tanpa pemahaman kontekstual yang lebih dalam.

Ketiga, keterbatasan dalam *input features* model. Penelitian ini menggunakan *embedding* yang dilatih dari awal. Pendekatan lain, seperti yang dibahas oleh Liang & Niu [4], adalah dengan meningkatkan *input* LSTM menggunakan *sliding window* untuk menangkap n-gram atau menggabungkan *embedding* statis (seperti TF-IDF) dengan *embedding* dinamis (word2vec). Peningkatan pada level fitur *input* ini tidak dieksplorasi dalam penelitian ini dan dapat menjadi arahan untuk perbaikan di masa depan.

5. KESIMPULAN

Penelitian ini berhasil mengimplementasikan model *deep learning* dengan arsitektur LSTM untuk klasifikasi tingkat stres dari teks percakapan digital, dengan hasil akurasi pengujian mencapai 86,48%. Hasil ini mengonfirmasi potensi LSTM sebagai metode yang efektif untuk analisis teks terkait kesehatan mental. Model menunjukkan kinerja yang seimbang dan andal dalam mengidentifikasi kelas "Stres" (F1-score 0,86) dan "Non-Stres" (F1-score 0,87), seperti yang telah ditunjukkan secara rinci pada Tabel 2.

Namun, penelitian ini juga secara transparan mengidentifikasi dua keterbatasan utama. Pertama, adanya indikasi *overfitting* yang terdeteksi dari kurva pelatihan (Gambar 2 dan 3). Fenomena ini tidak hanya berdampak pada generalisasi, tetapi juga pada keandalan prediksi, sejalan dengan temuan Ilias et al. [7] yang menyoroti pentingnya kalibrasi model (model calibration). Kedua, analisis *hyperparameter* (Gambar 4) menunjukkan bahwa konfigurasi model utama (64 unit) bersifat sub-optimal dan masih dapat ditingkatkan.

Berdasarkan temuan ini, penelitian di masa depan disarankan untuk fokus pada beberapa area. Pertama, untuk mengatasi *overfitting*, selain dapat diterapkan teknik regularisasi yang lebih

https://doi.org/10.25077/TEKNOSI.v12i1.2026.152-159

kuat (seperti menambah lapisan *dropout* dengan *rate* lebih tinggi) atau menggunakan augmentasi data, penelitian selanjutnya dapat mengeksplorasi teknik kalibrasi model. Sebagai contoh, Ilias et al. [7] menyarankan penggunaan label *smoothing* yang terbukti efektif untuk mengurangi *overconfidence* model.

Kedua, berdasarkan temuan langsung dari analisis sensitivitas *hyperparameter* (Gambar 4), penelitian selanjutnya sangat disarankan untuk mengeksplorasi optimasi arsitektur LSTM ini secara spesifik. Implementasi model dengan 128 unit LSTM dan *batch size* 128 terbukti menunjukkan F1-Score yang lebih unggul daripada konfigurasi *baseline* yang digunakan dalam penelitian ini.

Setelah optimasi tersebut dilakukan, barulah arsitektur yang lebih canggih seperti Bidirectional LSTM (Bi-LSTM) yang dilengkapi dengan *Attention mechanism* [9], atau model berbasis Transformer (misal, IndoBERT [16] atau MentalBERT [7]) dapat dieksplorasi untuk perbandingan lebih lanjut.

DAFTAR PUSTAKA

[1] T. Zhang, A. M. Schoene, S. Ji, and S. Ananiadou, 'Natural language processing applied to mental illness detection: a narrative review', Dec. 01, 2022, *Nature Research*. doi: 10.1038/s41746-022-00589-7.

[2] A. Sherstinsky, 'Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network', Jul. 2023, doi: 10.1016/j.physd.2019.132306.

[3] M. Dwirizqy Wimbassa, T. Marsyah Noor, S. Yasara, and T. Muhammad Arsyah, 'Emotional Text Detection dengan Long Short Term Memory (LSTM) 1', *Jurnal Format*, vol. 12, 2023.

[4] M. Liang and T. Niu, 'Research on Text Classification Techniques Based on Improved TF-IDF Algorithm and LSTM Inputs', in *Procedia Computer Science*, Elsevier B.V., 2022, pp. 460–470. doi: 10.1016/j.procs.2022.10.064.

[5] K. S. Witanto, N. A. Sanjaya ER, A. E. Karyawati, I. G. A. G. A. Kadyanan, I. K. G. Suhartana, and L. G. Astuti, 'Implementasi LSTM Pada Analisis Sentimen Review Film Menggunakan Adam Dan RMSprop Optimizer', *JELIKU (Jurnal Elektronik Ilmu Komputer Udayana)*, vol. 10, no. 4, p. 351, Jun. 2022, doi: 10.24843/jlk.2022.v10.i04.p05.

[6] A. Lisanthoni *et al.*, 'Penerapan LSTM dalam Analisis Sentimen Berbasis Lexicon untuk Meningkatkan Sistem Pemantauan Citra PLN di Platform Digital', *Seminar Nasional Sains Data*, vol. 2024.

[7] L. Ilias, S. Mouzakitis, and D. Askounis, 'Calibration of Transformer-based Models for Identifying Stress and Depression in Social Media', Jul. 2023, doi: 10.1109/TCSS.2023.3283009.

[8] S. Inamdar, R. Chapekar, S. Gite, and B. Pradhan, 'Machine Learning Driven Mental Stress Detection on Reddit Posts Using Natural Language Processing', *Human-Centric Intelligent Systems*, vol. 3, no. 2, pp. 80–91, Mar. 2023, doi: 10.1007/s44230-023-00020-8.

[9] X. Wan and L. Tian, 'User Stress Detection Using Social Media Text: A Novel Machine Learning Approach', *International Journal of Computers, Communications and Control*, vol. 19, no. 5, pp. 1–15, 2024, doi: 10.15837/ijccc.2024.5.6772.

[10] T. Nijhawan, G. Attigeri, and T. Ananthakrishna, 'Stress detection using natural language processing and machine learning over social interactions', *J Big Data*, vol. 9, no. 1, Dec. 2022, doi: 10.1186/s40537-022-00575-6.

[11] M. Krichen and A. Mihoub, 'Long Short-Term Memory Networks: A Comprehensive Survey', Sep. 01, 2025, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/ai6090215.

[12] K. Taha, P. D. Yoo, C. Yeun, D. Homouz, and A. Taha, 'A comprehensive survey of text classification techniques and their research applications: Observational and experimental insights', Nov. 01, 2024, *Elsevier Ireland Ltd*. doi: 10.1016/j.cosrev.2024.100664.

[13] F. Zakariya, J. Zeniarja, and S. Winarno, 'Pengembangan Chatbot Kesehatan Mental Menggunakan Algoritma Long Short-Term Memory', *Jurnal Media Informatika Budidarma*, vol. 8, no. 1, p. 251, Jan. 2024, doi: 10.30865/mib.v8i1.7177.

[14] R. Ilyas, F. Kasyidi, and M. N. Eriyadi, 'Penanganan *Overfitting* pada Klasifikasi Berita Hoax berbasis Neural Networks dengan *Dropout* dan Regularization', vol. 10, no. 2, pp. 156–162, 2024, doi: 10.31294/jtk.v4i2.

[15] N. Nordin, Z. Zainol, M. H. Mohd Noor, and L. F. Chan, 'An explainable predictive model for suicide attempt risk using an ensemble learning and Shapley Additive Explanations (SHAP) approach', *Asian J Psychiatr*, vol. 79, p. 103316, 2023, doi: 10.1016/j.ajp.2022.103316.

[16] J. Cahyani, S. Mujahidin, and T. P. Fiqar, 'Implementasi Metode Long Short Term Memory (LSTM) untuk Memprediksi Harga Bahan Pokok Nasional', *Jurnal Sistem dan Teknologi Informasi (JustIN)*, vol. 11, no. 2, p. 346, Jul. 2023, doi: 10.26418/justin.v11i2.57395.

BIODATA PENULIS

Agni Musadad

Penulis adalah mahasiswa yang memiliki ketertarikan dalam bidang teknologi informasi, khususnya kecerdasan buatan, analisis data, dan pemrosesan bahasa alami. Penulis aktif mengembangkan wawasan akademik serta keterampilan praktis melalui penelitian dan kerja sama tim lintas bidang. Minat utama penulis meliputi pengembangan model pembelajaran mesin, analisis teks, serta penerapan teknologi digital untuk mendukung inovasi di berbagai sektor.

Heni Sulastri

Penulis adalah salah satu dosen program studi Sistem Informasi yang memiliki ketertarikan dalam bidang data mining, software testing, social software engineering, sistem informasi, decission support system dan artificial intelligence. Selain mengajar aktif juga sebagai fasilitator Koding dan Kecerdasan Artifisial (KKA) Kemendikdasmen dibawah naungan Yayasan Sakata Inovation Center.