JEPIN
(Jurnal Edukasi dan Penelitian Informatika)
ISSN(e): 2548-9364 / ISSN(p) : 2460-0741

Vol. 10
No. 1
April 2024

Deteksi *Email Spam* menggunakan Algoritma *Convolutional Neural Network* (CNN)

Chris Moulana Bachri[#1], Wawan Gunawan[#2]

[#]*Teknik Informatika, Fakultas Ilmu Komputer, Universitas Mercu Buana*
Jalan Meruya Selatan No. 1 Jakarta Barat 11650
[1]41517110052@student.mercubuana.ac.id
[2]wawan.gunawan@mercubuana.ac.id

***Abstrak*— Deteksi *email spam* merupakan isu penting dalam keamanan siber di Indonesia, yang menempati posisi delapan teratas di dunia dalam hal pengiriman spam. Untuk mengatasi tantangan ini, penelitian ini memperkenalkan penggunaan algoritma Convolutional Neural Network (CNN). Dengan kemampuan superior dalam mempelajari dan mengenali pola dari dataset besar, CNN menawarkan pendekatan berbasis kecerdasan buatan yang lebih efektif daripada metode tradisional. Penelitian ini mengembangkan model CNN dengan menganalisis teks dari 15.271 email berbahasa Inggris dan Indonesia dengan menggunakan teknik pembersihan teks dan *Tokenization*. Hasilnya menunjukkan keefektivitasan CNN yang signifikan dalam mengklasifikasikan email dengan tingkat akurasi tinggi sebesar 99.67% untuk data uji 20%, 99.64% untuk data uji 30%, dan 99.63% untuk data uji 40%. Berdasarkan hasil pengujian tersebut menunjukkan bahwa algoritma CNN berpotensi kuat dalam meningkatkan keamanan digital.**

***Kata kunci*— CNN, *Email Spam*, Keamanan Siber, Analisis Teks, Teknik Informatika.**

I. PENDAHULUAN

Indonesia saat ini menduduki peringkat ke-8 di dunia dalam hal volume pengiriman spam [1]. Email, sebagai salah satu aplikasi komunikasi yang paling luas digunakan di seluruh dunia, memiliki kelemahan utama yaitu ketidakmampuannya dalam membedakan secara efisien antara isi email yang merupakan spam dan yang bukan spam (*ham*) [2]. Kejahatan siber yang termasuk pengiriman malware melalui email menjadi salah satu tantangan serius dalam ranah ini. Studi terbaru menunjukkan bahwa lebih dari 89% dari aktivitas internet saat ini didominasi oleh spam, dengan teknik baru spammer yang mencakup penggunaan gambar atau link dalam email, sehingga menyulitkan proses deteksi [3].

Penelitian sebelumnya telah mencoba berbagai metode dalam deteksi spam email. Penelitian terdahulu memanfaatkan algoritma *N-Gram* dan *Naive Bayes* melalui REST API dan berhasil mencapai tingkat akurasi deteksi hingga 98% [1]. Penelitian lain yang menggunakan algoritma *Binary Classifier* berhasil mencapai akurasi hingga 99% [2]. Penelitian lain juga mengevaluasi penggunaan algoritma K-NN dan SVM untuk tujuan yang sama [3], [4]. Namun, masih ada kekurangan dalam penelitian mengenai penggunaan algoritma *Convolutional Neural Network* (CNN) untuk identifikasi email spam, terutama dalam konteks keamanan siber.

Penelitian ini ditujukan untuk meningkatkan keamanan digital yang berhubungan dengan *email spam* menggunakan algoritma CNN yang sebelumnya menggunakan metode konvensional [5] dan bergantung pada *signature*, diubah menjadi berbasis AI [6], [7] yang inovatif untuk meningkatkan kemampuan dalam melakukan filter konten spam yang semakin kompleks. Selain itu penelitian ini memiliki relevansi global yang signifikan tidak hanya dalam meningkatkan keamanan siber di Indonesia, akan tetapi memberikan sumbangan penting ke masyarakat internasional. Algoritma CNN yang dikenal dengan kemampuannya yang unggul dalam mengidentifikasi pola dari dataset besar akan berpotensi meningkatkan efisiensi dan efektivitas dalam proses deteksi spam dengan memberikan skor terhadap setiap kata dalam sebuah email guna memberikan solusi yang lebih baik dalam mengatasi spam [8]. Penerapan CNN dalam deteksi *spam* diharapkan dapat memberikan manfaat signifikan bagi pengguna email dan institusi dengan menjamin penggunaan email yang lebih aman dan nyaman serta bebas dari gangguan *spam* [9]. Penelitian ini akan mengembangkan model prediksi yang tidak hanya efisien dalam mendeteksi spam yang ada saat ini, tapi juga fleksibel untuk beradaptasi dengan metode spam yang akan berkembang di masa depan. Langkah ini diharapkan menyediakan perlindungan berkelanjutan terhadap ancaman spam yang terus berubah.

II. LANDASAN TEORI

Keamanan siber sangat terkait dengan fenomena spam email, yang merupakan salah satu bentuk serangan siber paling umum. Spam email sering kali mengandung iklan yang tidak relevan atau link berbahaya yang bisa mengarah ke situs web merugikan atau mengunduh *malware* [1]. Risiko ini termasuk kehilangan data pribadi atau informasi sensitif seperti *password* dan nomor kartu kredit. *Spam* juga bisa mengarahkan pengguna ke situs *web phishing* yang bertujuan mencuri informasi sensitif [10].

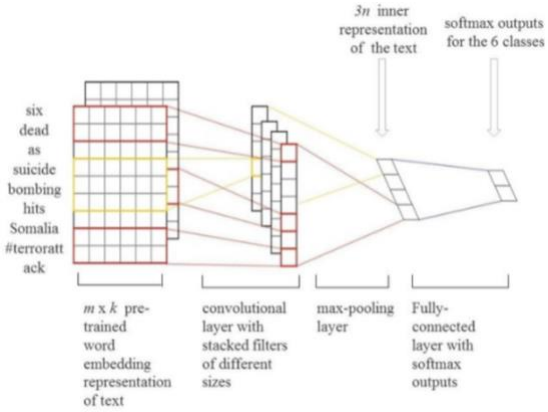
Submitted 04-12-2023; Revised 11-03-2024; Accepted 26-03-2024

Convolutional Neural Network (CNN), awalnya dikembangkan untuk pengenalan gambar [11], kini diaplikasikan pada pemrosesan teks, termasuk deteksi spam email. Dalam konteks ini, teks email direpresentasikan sebagai matriks, dengan baris yang mewakili token, dan vektor yang mewakili kata. *Word embeddings*, seperti *word2vec* digunakan untuk menangkap makna semantik kata-kata dalam ruang vektor. Alternatif lain adalah menggunakan *one-hot vektor*. Dalam CNN, filter konvolusi diaplikasikan pada matriks untuk mengekstraksi fitur dengan langkah konvolusi yang memperhatikan konteks spasial kata-kata. Setelah proses konvolusi, dilakukan *pooling* untuk mengurangi dimensi fitur yang dihasilkan yang kemudian dihubungkan ke lapisan-lapisan berikut dalam jaringan seperti lapisan *fully connected* untuk klasifikasi [12].

Model CNN terbukti efektif dalam deteksi email *spam* dengan akurasi mencapai 98% untuk Bahasa Indonesia dan 99% untuk Bahasa Inggris menunjukkan potensinya dalam deteksi *spam* dalam berbagai bahasa [12]. *Adam optimizer* yang merupakan algoritma optimisasi dalam pelatihan jaringan saraf tiruan termasuk CNN membantu menemukan parameter optimal dengan meminimalkan fungsi kerugian. Adam menggabungkan metode *gradien* turun stokastik dengan momen adaptif memperhitungkan *mean* dan *variance* dari gradien untuk mempercepat pencarian titik minimum global dari fungsi kerugian [13].

Dalam penggunaan API JSON REST yang memfasilitasi komunikasi antar sistem via HTTP dengan format data JSON, REST server menyediakan akses ke sumber data diidentifikasi dengan tautan URI dengan format data yang beragam seperti teks, JSON, atau XML. Metode HTTP yang digunakan antara lain GET untuk membaca, PUT untuk memperbarui, DELETE untuk menghapus, dan POST untuk menciptakan data baru. Dalam konteks deteksi *spam email*, sumber data seperti gmail dihubungkan dengan API JSON untuk visualisasi dalam halaman web [14].

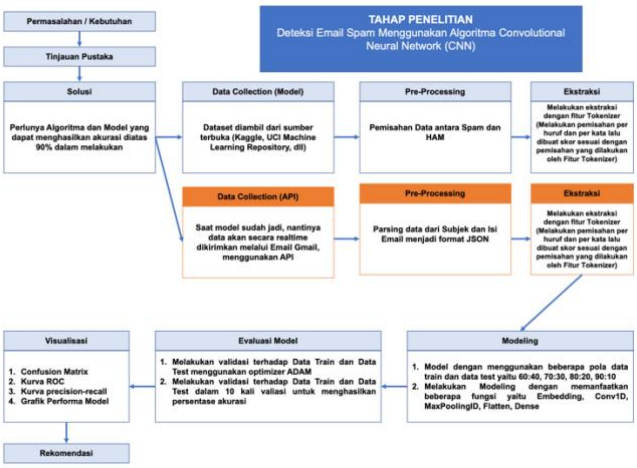
Email *spam* memiliki karakteristik tertentu yang membedakannya dari email *non-spam* termasuk iklan atau promosi yang tidak relevan, *link* mencurigakan, penggunaan kata-kata atau frasa khas *spam*, teks atau gambar menyesatkan, desakan untuk bertindak segera, informasi yang salah atau menyesatkan, lampiran mencurigakan, dan penggunaan bahasa yang tidak pantas [15], [16]. Dengan memanfaatkan arsitektur CNN pada representasi matriks dari teks dapat dimanfaatkan kemampuan CNN dalam mengekstraksi fitur-fitur penting dan mempelajari pola-pola yang berhubungan dengan deteksi *spam email* atau tugas pengolahan teks lainnya [12] seperti yang terlihat pada gambar 1.



Gambar. 1 Cara kerja CNN

III. METODOLOGI PENELITIAN

Penelitian ini mengadopsi pendekatan kuantitatif dengan memfokuskan pada pengembangan dan evaluasi model CNN untuk deteksi spam email, dengan tahapan penelitian seperti terlihat pada gambar 2.



Gambar. 2 Tahap penelitian

A. *Dataset*

Penelitian ini memanfaatkan dataset yang terdiri dari 15.271 email yang diperoleh dari platform Kaggle dan ditambah dengan data yang dikumpulkan langsung dari sumber email dan dilabeli secara manual sebanyak 100 email. Dataset ini dibagi menjadi dua kategori: *spam* dan *ham*. Sebanyak 6.556 email (42,93% dari total) dikategorikan sebagai *spam*, sedangkan 8.715 email (57,7% dari total) dikategorikan sebagai *ham*. Contoh dataset dapat dilihat pada gambar 3.

Keseimbangan dalam distribusi antara *spam* dan *ham* dalam dataset ini adalah aspek kritis yang menunjang keakuratan dan keadilan model pembelajaran mesin. Dengan proporsi yang hampir setengah-setengah antara kedua kategori, model yang akan dikembangkan memiliki peluang yang setara untuk belajar dari ciri-ciri khas masing-masing jenis email mengurangi risiko bias terhadap salah satu kelas.

```
5 baris pertama:
   Unnamed: 0 class                                               text
0         605   ham  Subject: enron methanol ; meter # : 988291\nth...
1        2349   ham  Subject: hpl nom for january 9 , 2001\n( see a...
2        3624   ham  Subject: neon retreat\nho ho ho , we ' re arou...
3        4685  spam  Subject: photoshop , windows , office . cheap ...
4        2030   ham  Subject: re : indian springs\nthis deal is to ...

5 baris terakhir:
       Unnamed: 0 class                                               text
15266          96   ham                                Pembayaran Berhasil
15267          97   ham                                Pembayaran Berhasil
15268          98  spam  Career Paths: SysAdmin, Pentester, SOC Analyst...
15269          99  spam  CND Exam Beta Testing Exercise Dear Member, EC...
15270         100  spam  Weekly Edition of EC-Council's Cyber Brief  ht...
```

Gambar. 3 Contoh *dataset*

B. *Pre-Processing*

1) Pada tahap awal penelitian, data diimpor dari file CSV yang berisi email dalam dua bahasa menggunakan library pandas seperti terlihat pada gambar 4. Selanjutnya dari hasil *import* data dilakukan pembersihan data terhadap data yang tidak lengkap atau rusak dengan menghapus nilai '*NaN'* pada kolom '*class'* yang dapat mengganggu akurasi dan keandalan model seperti ditampilkan pada gambar 5. Tahap pembersihan data ini penting untuk memastikan integritas data dan konsistensi hasil penelitian. Selanjutnya teks dari setiap email dikonversi menjadi string dan nilai pada kolom *'class'* diubah menjadi label biner, nilai 0 merepresentasikan *'ham'* sedangkan nilai 1 merepresentasikan *'spam'*. Konversi ini memudahkan model pembelajaran mesin untuk memproses data dan melakukan klasifikasi dengan lebih efektif.

```
data = pd.read_csv('dataset_spam_english_indo6.csv', encoding='latin-1')
```

Gambar. 4 Import *CSV*

```
data.dropna(subset=['class'], inplace=True)
data.dropna(subset=['label'], inplace=True)
```

Gambar. 5 Cleansing *data*

2) Tahap selanjutnya yaitu pembersihan dan tokenisasi teks, setiap email terlebih dahulu dibersihkan dari tanda baca dan stop words yang digambarkan dalam gambar 6. Langkah ini dilakukan untuk mengurangi *noise* dalam data dan memfokuskan analisis pada kata-kata yang lebih penting dan relevan. Setelah dilakukan proses pembersihan, selanjutnya dilakukan t*okenizer* dari Keras yang digunakan untuk mengubah kata-kata dalam teks menjadi urutan angka. Proses ini memungkinkan representasi teks yang lebih efisien dan konsisten untuk analisis komputasi. Selanjutnya, *padding* diterapkan pada urutan-urutan tersebut untuk memastikan bahwa semuanya memiliki panjang yang sama dan memfasilitasi pemrosesan yang seragam oleh model, seperti terlihat pada gambar 7.

```
X = X.apply(remove_punctuation_and_stopwords)
```

Gambar. 6 Membersihkan dataset menggunakan *punctuation*

```
tokenizer = Tokenizer(num_words=10000)
X = pad_sequences(X, maxlen=150)
```

Gambar. 7 Penggunaan teknik tokenizer

C. *Modelling & Training*

1) Selanjutnya menyiapkan data untuk membangun dan mengevaluasi model deteksi spam, kami menggunakan fungsi train_test_split dari scikit-learn, sebuah pustaka Python yang populer untuk pembelajaran mesin. Fungsi ini mengalokasikan sebagian data sebagai 'set pelatihan' yang digunakan untuk melatih model. Sementara sisanya menjadi 'set pengujian' yang digunakan untuk menguji kinerja model dalam kondisi yang tidak diketahui sebelumnya. Langkah ini untuk memastikan bahwa evaluasi model dapat dilakukan dengan objektif dan memberikan gambaran yang jelas tentang bagaimana model akan berperilaku dengan data baru. Setelah pemisahan data, langkah berikutnya adalah membangun arsitektur model CNN. CNN adalah jenis jaringan saraf tiruan yang sangat efektif dalam mengenali pola dalam data berdimensi tinggi, seperti gambar atau teks. Dalam konteks deteksi spam email, model CNN dirancang dengan lapisan spesifik sebagai berikut:

a. Lapisan *Embedding*: Ini adalah lapisan pertama yang bertindak sebagai peta input untuk mengubah kata-kata dalam teks menjadi vektor numerik. Vektor ini merupakan representasi padat dari kata yang mencakup informasi semantik dan sintaksis. Ini sangat penting karena memungkinkan model untuk memproses teks dalam format yang bisa dipahami oleh algoritma pembelajaran mesin.
b. Lapisan *Konvolusi*: Setelah *embedding*, kami menerapkan lapisan konvolusi yang menggunakan filter untuk memindai vektor dari lapisan sebelumnya. Filter ini secara efektif menangkap pola dan struktur dalam data teks, seperti frasa atau kombinasi kata yang sering muncul dalam konteks spam.
c. Lapisan *Maxpooling*: Berikutnya, lapisan *max pooling* digunakan untuk mereduksi informasi spasial dari output lapisan konvolusi. Ini membantu dalam mengurangi jumlah parameter dan komputasi dalam jaringan, serta membantu dalam membuat model lebih umum dan mengurangi kemungkinan *overfitting*.
d. Lapisan *Fully Connected (Dense Layer):* Fitur yang telah direduksi melalui *max pooling* kemudian disalurkan ke satu atau lebih lapisan padat. Di sini, model mempelajari kombinasi fitur yang kompleks dan melakukan klasifikasi berdasarkan fitur yang telah dipelajari.
e. Lapisan *Output*: Akhirnya, *output* dari lapisan *dense* diarahkan ke lapisan *output*, di mana fungsi aktivasi *sigmoid* digunakan untuk mengklasifikasikan input sebagai *spam* atau *non-spam*. *Sigmoid* menghasilkan probabilitas antara 0 dan 1, yang menunjukkan seberapa besar kemungkinan email tersebut adalah *spam*.

Model yang dibangun kemudian dilatih menggunakan set pelatihan dan validasi untuk menyesuaikan bobot dan bias berdasarkan *feedback* dari fungsi kerugian dengan tujuan meminimalkan kesalahan dalam prediksi, seperti ditampilkan pada gambar 8. Kami menggunakan *Adam optimizer*, sebuah metode yang efisien untuk menyesuaikan

bobot secara adaptif berdasarkan estimasi momen dari gradien. Setelah proses pelatihan, model dievaluasi menggunakan set pengujian untuk mengukur kinerjanya dengan fokus pada metrik seperti akurasi, presisi, recall, dan skor F1. Ini memberikan insight tentang seberapa baik model dapat mengidentifikasi email spam dalam kondisi nyata.

```
model = Sequential()
model.add(Embedding(10000, 100, input_length=X.shape[1]))
model.add(Conv1D(64, 5, activation='relu'))
model.add(MaxPooling1D(pool_size=4))
model.add(Flatten())
model.add(Dense(50, activation='relu', kernel_regularizer=l2(0.001)))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

Gambar. 8 *Modelling*

2) Proses pelatihan model dalam penelitian ini melibatkan dua tahap utama. Pertama, model dikompilasi menggunakan fungsi kerugian *binary_crossentropy* dan *optimizer adam*—keduanya dipilih karena kesesuaiannya dengan tugas klasifikasi biner. Hal ini memastikan bahwa model dapat meminimalkan kesalahan dalam prediksi kategori email sebagai spam atau ham selama fase pembelajaran. Kedua, model tersebut dilatih menggunakan set data pelatihan dan diverifikasi dengan menggunakan set data pengujian untuk memeriksa efektivitasnya. Selama pelatihan, teknik seperti early stopping dan reduce learning rate on plateau diterapkan sebagai callbacks untuk mengoptimalkan pembelajaran dan mencegah *overfitting*. Early stopping memungkinkan model untuk menghentikan pelatihan ketika tidak terjadi peningkatan pada metrik yang diawasi dalam beberapa *epoch*, sementara *reduce learning rate on plateau* secara otomatis menurunkan *learning rate* ketika tidak ada peningkatan pada performa model, membantu dalam mencari solusi yang lebih halus di ruang solusi dan konvergensi yang lebih baik.

D. *Visualisasi Data*

Selama fase pelatihan, kinerja model dipantau dan divisualisasikan menggunakan pustaka matplotlib, yang memungkinkan kita untuk melacak akurasi dan loss seiring waktu. Visualisasi ini sangat penting karena memberikan pemahaman intuitif tentang kemajuan model sepanjang proses pelatihan, seperti ditampilkan dalam gambar 9. Setelah proses pelatihan selesai, model dan *tokenizer* yang digunakan dalam proses tokenisasi teks disimpan ke dalam format file menggunakan pustaka *pickle*. Ini memungkinkan model dan *tokenizer* untuk digunakan kembali nanti, sehingga prediksi dapat dilakukan dengan cepat tanpa kebutuhan pelatihan ulang, yang diilustrasikan dalam gambar 10. Menyimpan model dan tokenizer adalah langkah penting untuk memastikan bahwa kinerja yang telah dicapai selama pelatihan dapat dipertahankan dan digunakan dalam aplikasi praktis di masa mendatang.

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.show()
```

Gambar. 9 Visualisasi hasil pelatihan

```
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle)
```

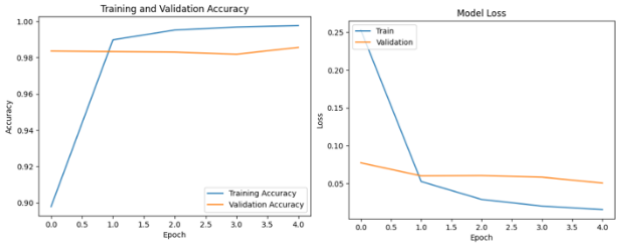
Gambar. 10 Penyimpanan model dan tokenizer

IV. HASIL DAN PEMBAHASAN

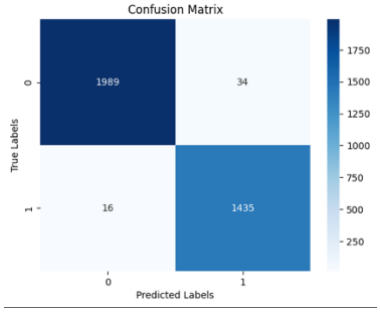
A. *Pengujian*

Model CNN diuji dalam tiga skenario pengujian yang berbeda berdasarkan ukuran dataset pengujian (0.2, 0.3, dan 0.4). Hasilnya sebagai berikut:

1) Pada uji coba dengan test size sebesar 20%, model menunjukkan peningkatan performa yang signifikan selama pelatihan dimulai dari akurasi 87.35% dengan loss sebesar 0.2898, hingga mencapai akurasi 99.67% dan loss sebesar 0.0177 pada epoch terakhir, sementara untuk nilai validasi, tingkat akurasi tertinggi yang dicapai adalah 98.53% dengan loss sebesar 0.0616 seperti ditampilkan pada gambar 11, hal ini dikonfirmasi oleh skor F1 yang sangat tinggi yaitu 0.98 yang menandakan bahwa efektivitas model dalam mempertahankan keseimbangan antara precision dan recall seperti ditampilkan pada gambar 12.

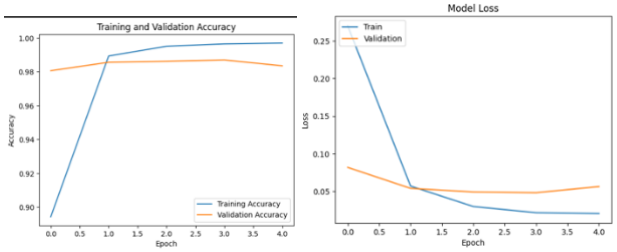


Gambar.11 Grafik accuracy dan *model test size* 0.2

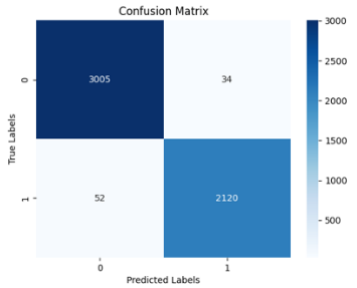


Gambar. 12 Grafik *confusion matrix test size* 0.2

2) Pada pengujian dengan test size 30%, model mengalami peningkatan performa yang konsisten selama fase pelatihan dengan nilai akurasi awal 86.21% dan loss 0.3083, dan akhirnya mencapai akurasi 99.64% dengan loss sebesar 0.0192 seperti terlihat pada gambar 13. Pada fase validasi, model berhasil mencatat akurasi 98.23% dengan loss 0.0674, dan berhasil mempertahankan F1-score yang tinggi yaitu 0.98 seperti terlihat pada gambar 14, yang menunjukkan kemampuan model yang handal dalam mengklasifikasikan email dengan tepat.

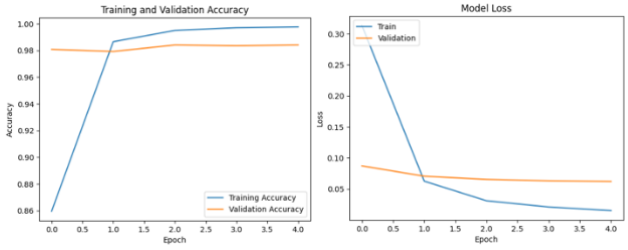


Gambar. 13 Grafik *accuracy* dan *model test size* 0.3

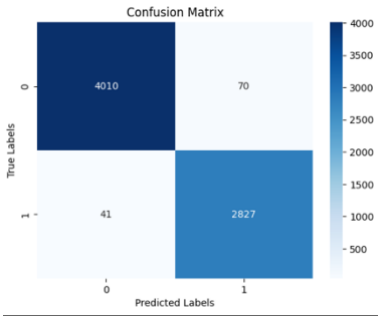


Gambar. 14 Grafik *confusion matrix test size* 0.3

3) Pada skenario pengujian dengan test size 40%, model menunjukkan peningkatan performa yang signifikan selama pelatihan, dimulai dari akurasi awal 82.75% dan meningkat menjadi 99.63% dengan *loss* yang berkurang menjadi 0.0188, sementara akurasi validasi mencapai 98.02% dengan *loss* validasi sebesar 0.0737 seperti terlihat pada gambar 15, dan model berhasil mempertahankan *F1-score* yang tinggi yaitu 0.98 seperti terlihat pada gambar 16, mengindikasikan kemampuan model yang efektif dalam mengklasifikasikan email secara akurat meskipun terjadi peningkatan proporsi data pengujian.



Gambar. 15 Grafik *accuracy dan model test size* 0.4

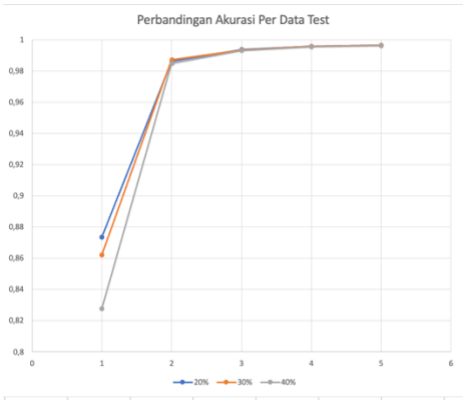


Gambar.16 Grafik *confusion matrix test size* 0.4

4) Secara keseluruhan model menunjukkan kinerja terbaik dengan *test size* 0.2, walau pada *test size* yang lebih besar model tetap mempertahankan *F1-score* yang tinggi meskipun ada penurunan dalam akurasi dan peningkatan *loss*. Ini menunjukkan bahwa model yang digunakan cukup *robust* terhadap perubahan ukuran data tes, seperti yang ditampilkan pada tabel 1 dan pada gambar 17.

TABEL I
RATA-RATA HASIL PENGUJIAN

Test Size	Epoch	Rata-Rata Loss	Rata-Rata Accuracy	Rata-Rata Val_Loss	Rata-Rata Val_Accuracy	F1-score
0.2	1-5	0.0844	0.9693	0.0651	0.9835	0.98
0.3	1-5	0.0896	0.9671	0.0777	0.9794	0.98
0.4	1-5	0.1016	0.9595	0.0808	0.9787	0.98



Gambar.17 Grafik perbandingan akurasi data test 0.2,0.3,0.4

B. *Pengujian Pada Data Real*

Tahap ini melibatkan implementasi sistem pengujian pada data nyata menggunakan script Python untuk menguji efektivitas model klasifikasi spam email yang telah dikembangkan. Langkah-langkah yang dilakukan adalah sebagai berikut:

1) Model klasifikasi spam (my_spam_classifier_model.h5) dan *tokenizer* (tokenizer.pickle) yang telah dilatih sebelumnya dimuat untuk memastikan penggunaan komponen yang sama seperti saat pelatihan model seperti ditampilkan pada gambar 18.

```
model = load_model('my_spam_classifier_model.h5')
```

Gambar.*18 Load* Model

2) Google API digunakan untuk mengakses Gmail memungkinkan pembacaan email secara otomatis dari akun Gmail. Kredensial dan token disiapkan untuk otentikasi seperti yang ditampilkan pada gambar 19.

```python
def get_gmail_service(credentials_file, token_file):
    creds = None
    if os.path.exists(token_file):
        with open(token_file, 'rb') as token:
            creds = pickle.load(token)
    if not creds or not creds.valid:
        if creds and creds.expired and creds.refresh_token:
            creds.refresh(Request())
        else:
            flow = InstalledAppFlow.from_client_secrets_file(credentials_file, SCOPES)
            creds = flow.run_local_server(port=0)
        with open(token_file, 'wb') as token:
            pickle.dump(creds, token)
    service = build('gmail', 'v1', credentials=creds)
    return service
```

Gambar.19 Gmail API

3) Fungsi get_gmail_service dan get_email_snippets digunakan untuk mengambil *snippet* dan *timestamp* dari email terbaru untuk dua akun berbeda dan memastikan pengujian yang lebih luas dapat dilakukan, seperti ditampilkan pada gambar 20.

```python
def get_email_snippets(service, last_email_timestamp=None, max_results=20):
    query = None
    if last_email_timestamp:
        query = f'after:{last_email_timestamp}'
    results = service.users().messages().list(userId='me', labelIds=['INBOX'], maxResults=max_results, q=query).execute()
    messages = results.get('messages', [])
    snippets = []
    timestamps = []
    for message in messages:
        msg_id = message['id']
        msg = service.users().messages().get(userId='me', id=msg_id, format='metadata', metadataHeaders=['Date']).execute()
        snippets.append(msg['snippet'])
        timestamps.append(msg['internalDate'])
    return snippets, timestamps
```

Gambar.20 Penarikan email dari Gmail, memanfaatkan API

4) Email yang diterima dibersihkan dari tanda baca dan *stop words* menggunakan fungsi *clean_email_text* dan memastikan bahwa teks yang diuji hanya berisi konten yang relevan, dengan *script* dapat dilihat pada gambar 21.

```python
def clean_email_text(text):
    translator = str.maketrans('', '', string.punctuation)
    no_punct = text.translate(translator)
    stop_words = set(['the', 'a', 'and', 'is', 'be', 'in'])  # Contoh, sesuaikan dengan stop words yang relevan
    return " ".join([word for word in no_punct.split() if word.lower() not in stop_words])
```

Gambar.21 *Cleansing* hasil penarikan email

5) Model digunakan untuk memprediksi apakah setiap email merupakan *spam* atau *ham* (email biasa) melalui fungsi predict_emails yang ditampilkan pada gambar 22.

```python
def predict_emails(emails, tokenizer, model):
    cleaned_emails = [clean_email_text(email) for email in emails]
    sequences = tokenizer.texts_to_sequences(cleaned_emails)
    data = pad_sequences(sequences, maxlen=150)
    predictions = (model.predict(data) > 0.5).astype(int)
    return ['spam' if prediction else 'ham' for prediction in predictions]
```

Gambar.22 Prediksi email

6) Semua prediksi, *snippet*, dan *timestamp email* dikonsolidasikan ke dalam *DataFrame* pandas sehingga memberikan gambaran yang jelas tentang hasil klasifikasi, seperti script yang ditampilkan pada gambar 23.

```python
prediction_df = pd.DataFrame(prediction_data)
```

Gambar.23 Integrasi email

7) Hasil prediksi disimpan dalam file CSV (email_predictions.csv) yang dapat dilihat pada gambar 24 untuk memudahkan review dan analisis lebih lanjut.

```python
csv_file_path = './spam_email/public/email_predictions.csv'

# Menyimpan DataFrame ke dalam file CSV
if not os.path.isfile(csv_file_path):
    prediction_df.to_csv(csv_file_path, mode='w', index=False)
else:
    prediction_df.to_csv(csv_file_path, mode='a', header=False, index=False)

print("Hasil prediksi telah ditambahkan ke dalam file '/spam_email/public/email_predictions.csv'.")
```

Gambar.24 Penyimpanan hasil CSV

C. Analisis Hasil

Model yang diuji menggunakan data email nyata dari akun Gmail menunjukkan hasil yang menggembirakan dengan tingkat akurasi yang tinggi dalam memprediksi kategori email. Email yang berisi promosi dan iklan telah konsisten diidentifikasi sebagai spam oleh model, sedangkan email yang berisi konten relevan, seperti lowongan pekerjaan, dikenali sebagai non-spam. Hasil ini menunjukkan bahwa model memiliki kemampuan yang baik dalam mengklasifikasikan email berdasarkan isi mereka.

Namun, ditemukan pula beberapa kasus di mana email yang seharusnya masuk kategori *non-spam* tetapi diklasifikasikan sebagai *spam*, dan juga sebaliknya. Ini mengindikasikan bahwa masih terdapat kelemahan dalam model yang berupa *false positives* dan *false negatives*. Salah satu contoh adalah email yang berisi rekomendasi pekerjaan dan informasi terkini seputar keamanan siber yang secara keliru dikategorikan sebagai *spam*. Hal ini menandakan bahwa masih diperlukan penyesuaian pada model untuk meningkatkan kemampuannya dalam membedakan antara *spam* dan konten yang bermanfaat.

Mengatasi masalah *false positives* dan *false negatives* merupakan langkah penting untuk memastikan bahwa model tidak hanya efektif, tetapi juga tidak menghambat alur kerja pengguna dengan mengklasifikasikan email penting secara salah. Hal ini memerlukan analisis lebih lanjut dan penyesuaian pada proses pelatihan model untuk meminimalisir kesalahan klasifikasi dan meningkatkan keakuratan prediksi.

V. KESIMPULAN

Berdasarkan serangkaian eksperimen dan analisis yang dilakukan pada arsitektur model CNN untuk klasifikasi teks email, dapat disimpulkan bahwa model CNN yang dirancang berhasil mempelajari representasi yang bermakna dari teks email dan mengklasifikasikannya ke dalam kategori spam dan ham dengan akurasi yang tinggi, seperti yang dibuktikan oleh nilai *F1-score* sebesar 0.98 pada berbagai skenario pengujian. Keefektivitasan CNN yang signifikan dalam mengklasifikasikan email dengan tingkat akurasi tinggi sebesar 99.67% untuk data uji 20%, 99.64% untuk data uji 30%, dan 99.63% untuk data uji 40%. Peningkatan performa model dari awal hingga akhir pelatihan menunjukkan bahwa proses pembelajaran berlangsung secara efektif dan model mampu menggeneralisasi dari data pelatihan ke data yang tidak dilihat sebelumnya. Pemilihan fitur, seperti penghapusan tanda baca dan *stop words*, serta proses tokenisasi dan *padding*, telah terbukti mempengaruhi dalam meningkatkan kualitas model dalam mengenali pola dari teks email. Terkait dengan penggunaan teknik *early stopping* dan *reduce learning rate on plateau* dalam proses pelatihan membantu mencegah *overfitting* dan memastikan konvergensi model yang lebih stabil.

Korespondensi : Wawan Gunawan

REFERENSI

[1] Y. Vernanda, S. Hansun, and M. B. Kristanda, "Indonesian language email spam detection using n-gram and naïve bayes algorithm," *Bull. Electr. Eng. Informatics*, vol. 9, no. 5, pp. 2012–2019, 2020, doi: 10.11591/eei.v9i5.2444.

[2] M. F. A. Kadir, A. F. A. Abidin, M. A. Mohamed, and N. A. Hamid, "Spam detection by using machine learning based binary classifier," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 26, no. 1, pp. 310–317, 2022, doi: 10.11591/ijeecs.v26.i1.pp310-317.

[3] Y. K. Zamil, S. A. Ali, and M. A. Naser, "Spam image email filtering using K-NN and SVM," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 1, pp. 245–254, 2019, doi: 10.11591/ijece.v9i1.pp245-254.

[4] A. Karim, S. Azam, B. Shanmugam, K. Kannoorpatti, and M. Alazab, "A comprehensive survey for intelligent spam email detection," *IEEE Access*, vol. 7, pp. 168261–168295, 2019, doi: 10.1109/ACCESS.2019.2954791.

[5] S. Larabi-Marie-Sainte, S. Ghouzali, T. Saba, L. Aburahmah, and R. Almohaini, "Improving spam email detection using deep recurrent neural network," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 25, no. 3, pp. 1625–1633, 2022, doi: 10.11591/ijeecs.v25.i3.pp1625-1633.

[6] N. Kumar, S. Sonowal, and Nishant, "Email Spam Detection Using Machine Learning Algorithms," *Proc. 2nd Int. Conf. Inven. Res. Comput. Appl. ICIRCA 2020*, no. September, pp. 108–113, 2020, doi: 10.1109/ICIRCA48905.2020.9183098.

[7] S. Huang *et al.*, "Automatic Modulation Classification Using Compressive Convolutional Neural Network," *IEEE Access*, vol. 7, pp. 79636–79643, 2019, doi: 10.1109/ACCESS.2019.2921988.

[8] S. Kaddoura, O. Alfandi, and N. Dahmani, "A Spam Email Detection Mechanism for English Language Text Emails Using Deep Learning Approach," *Proc. Work. Enabling Technol. Infrastruct. Collab. Enterp. WETICE*, vol. 2020-September, no. April 2021, pp. 193–198, 2020, doi: 10.1109/WETICE49692.2020.00045.

[9] H. Iswanto, E. Seniwati, Y. Astuti, and D. Maulina, "Comparison of Algorithms on Machine Learning For Spam Email Classification," *IJISTECH (International J. Inf. Syst. Technol.*, vol. 5, no. 4, p. 446, 2021, doi: 10.30645/ijistech.v5i4.164.

[10] Y. Fang, C. Zhang, C. Huang, L. Liu, and Y. Yang, "Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism," *IEEE Access*, vol. 7, pp. 56329–56340, 2019, doi: 10.1109/ACCESS.2019.2913705.

[11] K. H. Jin, M. T. McCann, E. Froustey, and M. Unser, "Deep Convolutional Neural Network for Inverse Problems in Imaging," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4509–4522, 2017, doi: 10.1109/TIP.2017.2713099.

[12] B. P. B. I. C. S. Putra, "Deteksi Ujaran Kebencian Dengan Menggunakan Algoritma Convolutional Neural Network Pada Gambar," *e-Proceeding Eng.* , vol. 5, no. 2, pp. 2395–2402, 2018.

[13] I. K. M. Jais, A. R. Ismail, and S. Q. Nisa, "Adam Optimization Algorithm for Wide and Deep Neural Network," *Knowl. Eng. Data Sci.*, vol. 2, no. 1, p. 41, 2019, doi: 10.17977/um018v2i12019p41-46.

[14] M. Haekal and Eliyani, "Token-based authentication using JSON Web Token on SIKASIR RESTful Web Service," *2016 Int. Conf. Informatics Comput. ICIC 2016*, no. Icic, pp. 175–179, 2017, doi: 10.1109/IAC.2016.7905711.

[15] O. Kembuan, G. Caren Rorimpandey, and S. Milian Tompunu Tengker, "Convolutional Neural Network (CNN) for Image Classification of Indonesia Sign Language Using Tensorflow," *2020 2nd Int. Conf. Cybern. Intell. Syst. ICORIS 2020*, no. 26, 2020, doi: 10.1109/ICORIS50180.2020.9320810.

[16] A. Theodorus, T. K. Prasetyo, R. Hartono, and D. Suhartono, "Short Message Service (SMS) Spam Filtering using Machine Learning in Bahasa Indonesia," *3rd 2021 East Indones. Conf. Comput. Inf. Technol. EIConCIT 2021*, pp. 199–202, 2021, doi: 10.1109/EIConCIT50028.2021.9431859.