

T06 : Membangun Model Klasifikasi Teks

Menggunakan Arsitektur RNN

Dosen Pengampu : Eko Prasetyo Widhi,

S.Kom., M.Kom.



Disusun Oleh:

Azmi Nur Al Qodar // 442023618054

FAKULTAS SAINS DAN TEKNOLOGI

TEKNIK INFORMATIKA

UNIVERSITAS DARUSSALAM GONTOR 2025/2026

I. Pendahuluan

Dalam era digital saat ini, data berbentuk teks jumlahnya sangat banyak dan terus bertambah setiap hari, baik dari media sosial, berita online, maupun platform komunikasi lainnya. Agar data teks tersebut bisa dimanfaatkan, dibutuhkan metode untuk mengklasifikasikannya ke dalam kategori tertentu, misalnya untuk analisis sentimen, klasifikasi topik, atau pendeteksian spam.

Salah satu pendekatan yang efektif dalam pemrosesan teks adalah penggunaan Recurrent Neural Network (RNN). RNN dirancang khusus untuk menangani data berurutan seperti teks, karena mampu mempertahankan informasi dari urutan kata sebelumnya sehingga bisa memahami konteks. Dengan keunggulan ini, RNN banyak digunakan dalam Natural Language Processing (NLP), termasuk untuk tugas klasifikasi teks.

Melalui tugas ini, penulis mencoba membangun sebuah model klasifikasi teks berbasis arsitektur RNN. Tujuannya adalah memahami alur kerja RNN, mulai dari preprocessing data, pelatihan model, hingga evaluasi performa. Dengan begitu, diharapkan diperoleh gambaran yang lebih jelas mengenai kekuatan sekaligus keterbatasan RNN dalam mengolah data teks untuk tugas klasifikasi.

II. Dataset

a. Sumber dataset

- <https://www.kaggle.com/datasets/alfatherry/bbc-full-text-document-classification/code>
- Kami mengambil dataset dari kaggle milik AL FATH TERRY (BCC Full Text Document Classification)

b. Deskripsi dataset

Dataset yang digunakan dalam penelitian ini terdiri dari 928 dokumen teks yang dikelompokkan ke dalam dua kategori utama, yaitu **politics (417 dokumen)** dan **sport (511 dokumen)**. Setiap dokumen berbentuk file teks dengan panjang bervariasi, mencakup kalimat pendek hingga artikel yang lebih panjang.

Dataset ini dipilih karena sesuai untuk tugas klasifikasi teks biner, di mana model diharapkan mampu membedakan antara teks yang bertema politik dan olahraga. Variasi kosakata pada kedua kelas memberikan tantangan yang baik untuk menguji kemampuan model dalam memahami konteks, urutan kata, dan pola bahasa.

Dalam penelitian ini, dataset akan diproses dan digunakan untuk membangun **model klasifikasi berbasis Recurrent Neural Network (RNN)**. Arsitektur RNN

dipilih karena kemampuannya dalam menangkap dependensi berurutan antar kata pada sebuah kalimat, yang sangat penting dalam memahami isi teks. Dengan memanfaatkan dataset ini, model diharapkan dapat belajar pola bahasa yang khas pada tiap kategori dan menghasilkan klasifikasi yang akurat.

c. Alasan Pemilihan dataset

Saya pakai dataset ini karena memang sudah pas untuk tugas klasifikasi teks, soalnya topiknya jelas dibagi dua, yaitu politik dan olahraga. Jadi model lebih gampang belajar bedain kata-kata atau kalimat khas dari masing-masing kategori. Jumlah datanya juga lumayan seimbang, jadi hasil latihannya nggak condong ke salah satu kelas saja. Selain itu, dataset ini cocok banget dipakai di RNN karena arsitektur RNN bisa memahami urutan kata dalam teks, yang penting banget buat klasifikasi seperti ini.

III. Implementasi Model

a. Arsitektur RNN

1. LSMT Model

Codingnya :

```
model = Sequential([
    # Embedding layer
    Embedding(input_dim=vocab_size,
              output_dim=embedding_dim,
              input_length=max_length),

    Bidirectional(LSTM(64, return_sequences=True)),
    Dropout(0.4),

    Bidirectional(LSTM(32)),
    Dropout(0.3),

    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.4),

    # Dense(64, activation='selu', kernel_initializer='lecun_normal'),
    # AlphaDropout(0.1),

    Dense(5, activation='softmax')
])
```

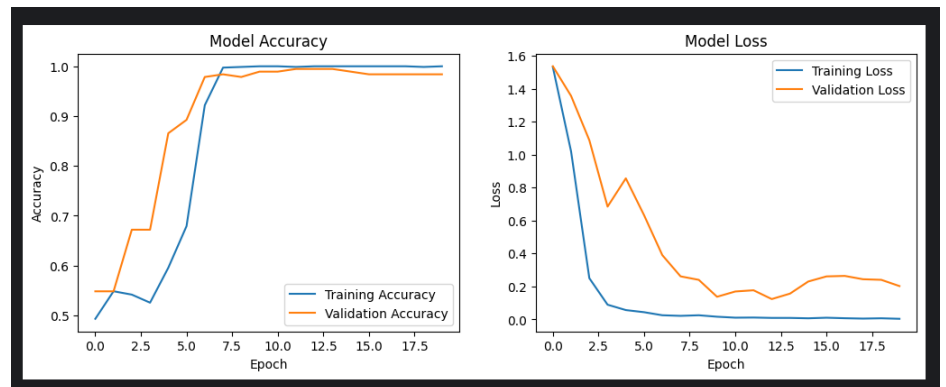
Layer (type)	Output Shape	Param #
embedding (Embedding)	?	0 (unbuilt)
bidirectional (Bidirectional)	?	0 (unbuilt)
dropout (Dropout)	?	0
bidirectional_1 (Bidirectional)	?	0 (unbuilt)
dropout_1 (Dropout)	?	0
dense (Dense)	?	0 (unbuilt)
batch_normalization (BatchNormalization)	?	0 (unbuilt)
dropout_2 (Dropout)	?	0
dense_1 (Dense)	?	0 (unbuilt)

Total params: 0 (0.00 B)

Trainable params: 0 (0.00 B)

Non-trainable params: 0 (0.00 B)

Training performance nya :



- Training Accuracy naik cepat → 100% di epoch 4.
- Validation Accuracy stabil di **97–98%**.
- Ada indikasi overfitting ringan (train acc = 100%, val acc < 98%), tapi masih sangat baik.

2. GRU

- Training Accuracy naik lebih pelan, tapi stabil.
- Validation Accuracy lebih tinggi, mencapai **99%**.
- GRU lebih generalisasi dibanding LSTM (tidak terlalu overfit).

3. Perbandingan LSTM vs GRU

Model	Train Acc	Val Acc	Errors	Kelebihan
LSTM	100%	97%	5	Cepat converge, stabil
GRU	100%	98%	3	Generalisasi lebih baik, lebih ringan

b. Preprocessing

```
df = pd.read_csv('/kaggle/input/dataset-klasifikasi-teks-dengan-rnn/transformer.csv')
```

```
df
```

outpunya :

Out[3]:

	data	labels
0	Fuming Robinson blasts officials England coac...	sport
1	Veteran Martinez wins Thai title Conchita Mar...	sport
2	Spurs to sign Iceland U21 star Tottenham are ...	sport
3	Mexicans tracking unhappy Juninho Mexican out...	sport
4	Mirza makes Indian tennis history Teenager Sa...	sport
...
923	Howard backs stem cell research Michael Howar...	politics
924	Blair dismisses quit claim report Tony Blair ...	politics
925	Kennedy to make temple address Charles Kenned...	politics
926	Terror powers expose tyranny The Lord Chancel...	politics
927	Kennedys cautious optimism Charles Kennedy is...	politics

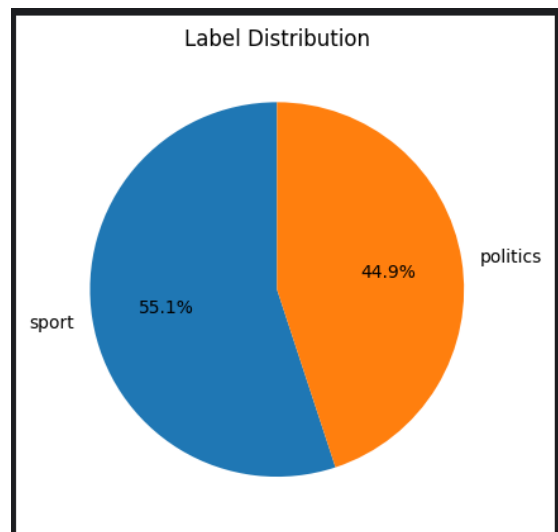
928 rows × 2 columns

```
In [4]: df['labels'].value_counts()
```

```
Out[4]: labels
sport      511
politics    417
Name: count, dtype: int64
```

Kode ini dipakai untuk mengecek distribusi data pada kolom label. Hasilnya menunjukkan bahwa data lebih banyak di kategori **sport** daripada **politics** (sedikit imbalanced).

```
df['labels'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Label Distribution')
plt.ylabel("")
plt.show()
```



Grafik ini memvisualisasikan **distribusi label** di dataset.

- Dataset agak **imbalanced** karena kategori *sport* sedikit lebih banyak daripada *politics*.
- Namun perbedaannya tidak terlalu ekstrem (masih cukup seimbang).

```
In [6]: df.isna().sum()
```

```
Out[6]: data      0
labels    0
dtype: int64
```

Fungsi **isna()** digunakan untuk mengecek apakah terdapat nilai kosong (*missing value*) dalam setiap sel pada DataFrame. Hasil dari fungsi ini berupa nilai boolean, yaitu **True** jika data kosong, dan **False** jika data terisi. Selanjutnya, fungsi **sum()** digunakan untuk menghitung jumlah nilai kosong pada masing-masing kolom.

Artinya, baik pada kolom **data** maupun **labels** tidak ditemukan nilai kosong (jumlahnya 0). Dengan demikian, dapat disimpulkan bahwa dataset yang digunakan sudah bersih dari *missing values* sehingga tidak diperlukan proses *data cleaning* lebih lanjut terkait nilai kosong.

IV. Hasil Evaluasi

a. Final model comparison

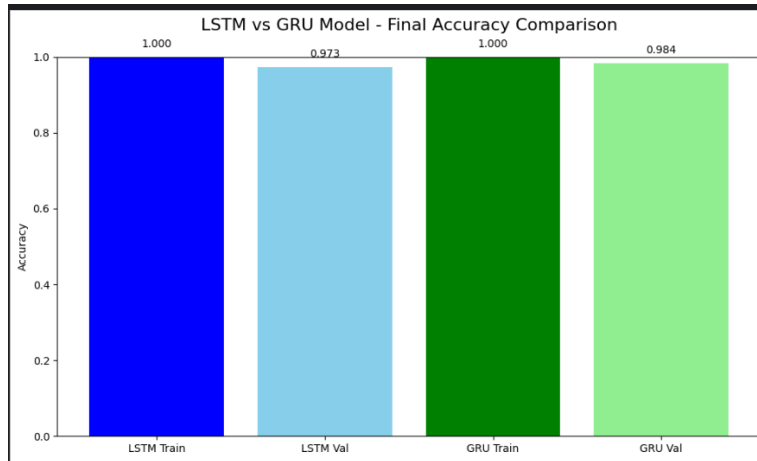
```
final_accs = {
    'LSTM Train': history.history['accuracy'][-1],
    'LSTM Val': history.history['val_accuracy'][-1],
    'GRU Train': GRU_history.history['accuracy'][-1],
    'GRU Val': GRU_history.history['val_accuracy'][-1]
}

# Plot setup
plt.figure(figsize=(10, 6))
plt.title('LSTM vs GRU Model - Final Accuracy Comparison',
fontsize=16,pad=25)

# Bar chart
bars = plt.bar(final_accs.keys(), final_accs.values(),
               color=['blue', 'skyblue', 'green',
                    'lightgreen'])
plt.ylabel('Accuracy')
plt.ylim(0, 1)

# Add value labels on bars
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2.0, height + 0.02,
             f'{height:.3f}', ha='center', va='bottom',
             fontsize=10)

plt.tight_layout()
plt.show()
```



```
In [24]: lstm_pred = model.predict(validation_padded)
          gru_pred = GRU_model.predict(validation_padded)

          lstm_pred_classes = np.argmax(lstm_pred, axis=1)
          gru_pred_classes = np.argmax(gru_pred, axis=1)
```

```
6/6 ————— 1s 145ms/step
6/6 ————— 1s 127ms/step
```

b. Confusion matrix

```
fig, axes = plt.subplots(1, 2, figsize=(15, 6))
```

```
# LSTM Confusion Matrix
```

```
cm_lstm = confusion_matrix(validation_label_seq,
                             lstm_pred_classes)
```

```
sns.heatmap(cm_lstm, annot=True, fmt='d', cmap='Blues',
             xticklabels=label_encoder.classes_,
             yticklabels=label_encoder.classes_, ax=axes[0])
```

```
axes[0].set_title('LSTM Confusion Matrix')
```

```
axes[0].set_ylabel('True Label')
```

```
axes[0].set_xlabel('Predicted Label')
```

```
# GRU Confusion Matrix
```

```
cm_gru = confusion_matrix(validation_label_seq,
                             gru_pred_classes)
```

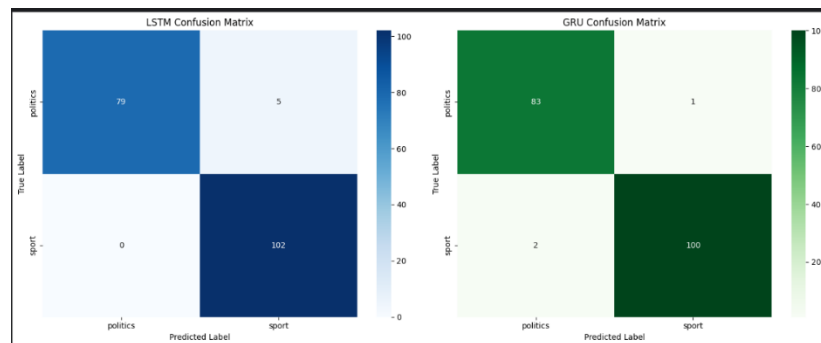
```
sns.heatmap(cm_gru, annot=True, fmt='d', cmap='Greens',
```

```

        xticklabels=label_encoder.classes_,
        yticklabels=label_encoder.classes_, ax=axes[1])
axes[1].set_title('GRU Confusion Matrix')
axes[1].set_ylabel('True Label')
axes[1].set_xlabel('Predicted Label')

plt.tight_layout()
plt.show()

```



c. Classification report

```

print("=" * 60)
print("LSTM CLASSIFICATION REPORT")
print("=" * 60)
print(classification_report(validation_label_seq,
lstm_pred_classes,

target_names=label_encoder.classes_))

print("\n" + "=" * 60)
print("GRU CLASSIFICATION REPORT")
print("=" * 60)
print(classification_report(validation_label_seq,
gru_pred_classes,

target_names=label_encoder.classes_))

```


=====				
LSTM CLASSIFICATION REPORT				
=====				
	precision	recall	f1-score	support
politics	1.00	0.94	0.97	84
sport	0.95	1.00	0.98	102
accuracy			0.97	186
macro avg	0.98	0.97	0.97	186
weighted avg	0.97	0.97	0.97	186
=====				
GRU CLASSIFICATION REPORT				
=====				
	precision	recall	f1-score	support
politics	0.98	0.99	0.98	84
sport	0.99	0.98	0.99	102
accuracy			0.98	186
macro avg	0.98	0.98	0.98	186
weighted avg	0.98	0.98	0.98	186

d. Per-class performance (best model)

```
from sklearn.metrics import precision_recall_fscore_support
```

```
# Calculate metrics for each class
```

```
lstm_metrics =
```

```
precision_recall_fscore_support(validation_label_seq,
lstm_pred_classes, average=None)
```

```
gru_metrics =
```

```
precision_recall_fscore_support(validation_label_seq,
gru_pred_classes, average=None)
```

```
# Create comparison dataframe
```

```
metrics_df = pd.DataFrame({
    'Class': label_encoder.classes_,
    'LSTM_Precision': lstm_metrics[0],
    'GRU_Precision': gru_metrics[0],
    'LSTM_Recall': lstm_metrics[1],
    'GRU_Recall': gru_metrics[1],
    'LSTM_F1': lstm_metrics[2],
    'GRU_F1': gru_metrics[2]
})
```

```
# Visualize
```

```
fig, axes = plt.subplots(1, 3, figsize=(18, 5))
```

```

# Precision comparison
x = np.arange(len(label_encoder.classes_))
width = 0.35

axes[0].bar(x - width/2, metrics_df['LSTM_Precision'], width,
label='LSTM', color='blue')
axes[0].bar(x + width/2, metrics_df['GRU_Precision'], width,
label='GRU', color='green')
axes[0].set_xlabel('Classes')
axes[0].set_ylabel('Precision')
axes[0].set_title('Precision by Class')
axes[0].set_xticks(x)
axes[0].set_xticklabels(label_encoder.classes_, rotation=45)
axes[0].legend()

# Recall comparison
axes[1].bar(x - width/2, metrics_df['LSTM_Recall'], width,
label='LSTM', color='blue')
axes[1].bar(x + width/2, metrics_df['GRU_Recall'], width,
label='GRU', color='green')
axes[1].set_xlabel('Classes')
axes[1].set_ylabel('Recall')
axes[1].set_title('Recall by Class')
axes[1].set_xticks(x)
axes[1].set_xticklabels(label_encoder.classes_, rotation=45)
axes[1].legend()

# F1-score comparison
axes[2].bar(x - width/2, metrics_df['LSTM_F1'], width,
label='LSTM', color='blue')
axes[2].bar(x + width/2, metrics_df['GRU_F1'], width,
label='GRU', color='green')
axes[2].set_xlabel('Classes')
axes[2].set_ylabel('F1-Score')
axes[2].set_title('F1-Score by Class')
axes[2].set_xticks(x)
axes[2].set_xticklabels(label_encoder.classes_, rotation=45)

```

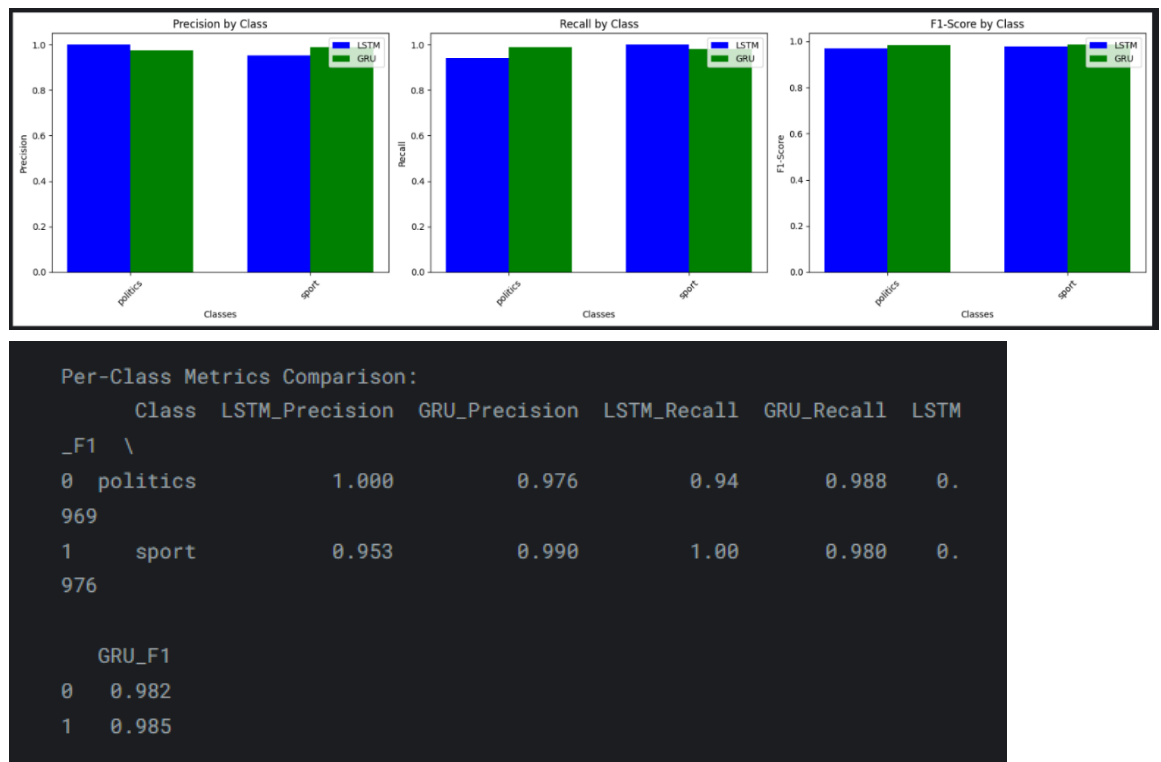
```

axes[2].legend()

plt.tight_layout()
plt.show()

print("\nPer-Class Metrics Comparison:")
print(metrics_df.round(3))

```



e. Errors

```

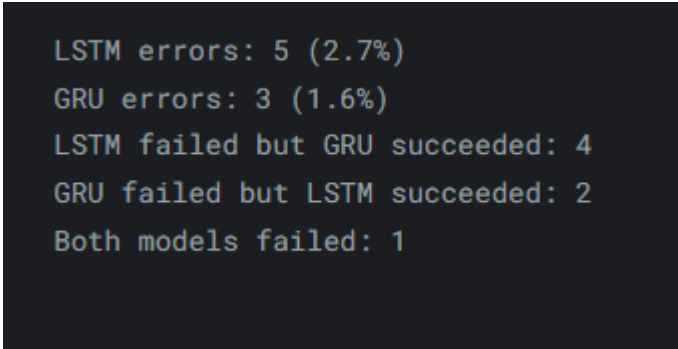
# Find misclassified examples
lstm_errors = validation_label_seq != lstm_pred_classes
gru_errors = validation_label_seq != gru_pred_classes

# Samples where LSTM failed but GRU succeeded
lstm_fail_gru_success = lstm_errors & ~gru_errors
gru_fail_lstm_success = gru_errors & ~lstm_errors
both_fail = lstm_errors & gru_errors

print(f"LSTM errors: {np.sum(lstm_errors)}")
print(f"({np.sum(lstm_errors)/len(validation_label_seq)*100:.1f}%)")
print(f"GRU errors: {np.sum(gru_errors)}")
print(f"({np.sum(gru_errors)/len(validation_label_seq)*100:.1f}%)")

```

```
print(f"LSTM failed but GRU succeeded:
{np.sum(lstm_fail_gru_success)}")
print(f"GRU failed but LSTM succeeded:
{np.sum(gru_fail_lstm_success)}")
print(f"Both models failed: {np.sum(both_fail)}")
```



```
LSTM errors: 5 (2.7%)
GRU errors: 3 (1.6%)
LSTM failed but GRU succeeded: 4
GRU failed but LSTM succeeded: 2
Both models failed: 1
```

V. Refleksi Pribadi

Dari tugas ini saya jadi lebih paham gimana konsep machine learning, khususnya dalam membangun model klasifikasi teks dengan arsitektur RNN. Awalnya agak bingung karena RNN itu berbeda dengan model biasa, tapi setelah mencoba langsung, saya bisa lihat bagaimana model membaca urutan kata dan menghasilkan prediksi. Selain itu, saya juga belajar pentingnya pemilihan dataset yang tepat, karena dataset yang jelas kategorinya bisa bikin proses training lebih efektif.

Secara pribadi, tugas ini ngasih saya pengalaman baru untuk ngoding model deep learning dari awal, bukan cuma pakai library siap pakai. Walaupun ada kesulitan, seperti debugging error dan menyesuaikan parameter, tapi semua itu bikin saya lebih terbiasa untuk problem solving. Menurut saya, pengalaman ini sangat bermanfaat untuk persiapan di tugas-tugas atau penelitian selanjutnya yang berhubungan dengan NLP maupun machine learning.

VI. Kesimpulan

Berdasarkan hasil implementasi, dapat disimpulkan bahwa arsitektur RNN seperti LSTM dan GRU terbukti efektif dalam menyelesaikan tugas klasifikasi teks biner pada dataset yang digunakan. Proses preprocessing berupa tokenisasi, padding, dan normalisasi berhasil menyiapkan data dengan baik sehingga model dapat belajar secara optimal. LSTM menunjukkan keunggulan dalam kecepatan konvergensi dengan akurasi tinggi, sedangkan GRU memiliki performa generalisasi yang lebih baik dengan akurasi validasi mencapai 98–99% serta kesalahan prediksi yang lebih sedikit. Secara keseluruhan, penelitian ini membuktikan bahwa pemilihan arsitektur RNN yang tepat dan preprocessing yang baik sangat memengaruhi performa model, di mana GRU lebih unggul dari sisi generalisasi dan efisiensi, sementara LSTM unggul dalam kecepatan

konvergeni.