Rekayasa Hijau: Jurnal Teknologi Ramah Lingkungan Volume 10 | Nomor 1
ISSN [e]: 2579-4264 | DOI: https://doi.org/10.26760/jrh.V10i1.51-66 Maret 2026

Monophonic Guitar Notation Estimation Using YIN Pitch Detection Algorithm and Long Short-Term Memory

Yusup Miftahuddin[1*], Fadhlan Muhammad Azka[1], Youllia Indrawaty N[1]

[1]Institut Teknologi Nasional, Kota Bandung, Indonesia
Email: yusufm@itenas.ac.id, fadhlanazka0@gmail.com, youllia@itenas.ac.id

17 Januari 2026 | Revised 24 Januari 2026 | Accepted 2 Februari 2026

ABSTRAK

Proses transkripsi audio gitar ke dalam bentuk tablature masih menjadi tantangan bagi pemula karena adanya ambiguitas posisi nada pada fretboard*, di mana satu nada yang sama dapat dimainkan pada kombinasi senar dan fret yang berbeda. Penelitian ini bertujuan untuk mengembangkan sistem estimasi tablature gitar monofonik yang efisien menggunakan kombinasi algoritma YIN dan model* Long Short-Term Memory *(LSTM). Algoritma YIN digunakan untuk mengekstraksi frekuensi fundamental (f_0) dari sinyal audio WAV, yang kemudian divalidasi melalui serangkaian tahap* post-processing *untuk memastikan stabilitas nada. Model LSTM kemudian digunakan untuk memetakan urutan nada tersebut ke dalam representasi senar dan fret berdasarkan pola ergonomi jari. Hasil penelitian menunjukkan bahwa model LSTM yang dioptimasi menggunakan Optuna mencapai akurasi tertinggi sebesar 83,22%. Selain itu, penerapan* parameter constraint *pada model terbukti mampu menghasilkan rekomendasi posisi jari yang lebih ergonomis bagi pemain gitar dibandingkan model* baseline*. Sistem ini diimplementasikan dalam aplikasi berbasis web yang memungkinkan transkripsi audio ke tablature secara otomatis dengan beban komputasi yang rendah.*

***Kata kunci**: Tablature Gitar, Monofonik, Algoritma YIN, LSTM, Transkripsi Musik*

ABSTRACT

Transcribing guitar audio into tablature remains a significant challenge for beginners due to pitch ambiguity on the fretboard, where a single note can be played on multiple string and fret combinations. This research aims to develop an efficient monophonic guitar tablature estimation system using a combination of the YIN algorithm and Long Short-Term Memory (LSTM) models. The YIN algorithm is utilized to extract fundamental frequencies (f0) from WAV audio signals, which are further validated through various post-processing stages to ensure pitch stability. Subsequently, an LSTM model maps these pitch sequences into string and fret representations based on fingering patterns. The results indicate that the LSTM model optimized with Optuna achieved the highest accuracy of 83.22%. Furthermore, the implementation of parameter constraints in the model proved effective in generating more ergonomic fingering recommendations compared to the baseline model. This system is implemented as a web-based application, enabling automated audio-to-tablature transcription with low computational overhead.

***Keywords**: Guitar Tablature, Monophonic, YIN Algorithm, LSTM, Music Transcription.*

1. PENDAHULUAN

Musik merupakan salah satu bentuk sinyal audio yang kompleks. Musik sering menjadi objek penelitian pengolahan sinyal digital dan *machine learning*. Dalam kehidupan sehari-hari, musik berperan sebagai media ekspresi dan hiburan [1]. Musik juga dinikmati oleh berbagai kelompok masyarakat pada era modern [2]. Secara bentuk data, musik dapat direpresentasikan sebagai deret frekuensi, amplitudo, dan waktu. Representasi ini dapat dianalisis dan dimanipulasi oleh berbagai algoritma pemrosesan sinyal.
Gitar merupakan instrumen yang banyak digunakan dalam bermain musik. Instrumen ini populer di kalangan musisi profesional maupun sekedar hobi [3]. Gitar umumnya menggunakan setelan standar EADGBE dengan enam senar [4]. Popularitas gitar juga tampak pada penggunaannya di berbagai konteks pertunjukan, baik skala besar maupun informal [5]. Pada gitar, satu nada dapat dimainkan oleh beberapa kombinasi senar dan *fret*.

Dalam praktik pembelajaran gitar, banyak pemula yang memulai secara otodidak. Mereka sering bergantung pada materi dari internet dan tidak selalu memiliki landasan teori musik yang kuat [6]. Sedangkan permainan yang bersifat monofonik menuntut kemampuan untuk menghubungkan nada yang didengar dengan posisi senar dan *fret* [7]. Salah satu media notasi yang sering digunakan gitaris adalah *tablature*. *Tablature* gitar merepresentasikan senar dan *fret* pada leher gitar sebagai bentuk notasi grafis [8]. Notasi ini tidak menyatakan simbol nada, tetapi menyatakan senar dan *fret* yang harus ditekan [9].

Penelitian mengenai kandungan frekuensi harmonik pada akor gitar oleh Khairil Anwar dan Viridi, menunjukkan bahwa sinyal gitar dapat diurai ke dalam komponen harmonik. Ketepatan pemilihan nada dibutuhkan untuk menjamin kualitas musik yang dihasilkan [10]. Kepekaan terhadap suara dan struktur musik berkembang melalui proses pembelajaran yang panjang. Kepekaan terhadap nada dan kunci dasar meningkat seiring bertambahnya pengalaman hingga usia sekolah [11]. Kemampuan membedakan *pitch* juga bervariasi antar individu. Musisi terlatih memiliki kepekaan yang lebih tinggi terhadap perbedaan nada dibandingkan orang yang tidak terbiasa bermain musik [12]. Akibatnya, mengenali nada gitar hanya dari pendengaran merupakan tugas yang sulit bagi pemula, bahkan bagi sebagian musisi berpengalaman.

Dari sisi komputasi, nada musik dapat dikenali melalui pemrosesan sinyal audio. Berbagai algoritma dapat digunakan untuk mengekstraksi frekuensi fundamental dan karakteristik lain dari sinyal [13]. Penggunaan komputer sebagai sistem pengenalan nada gitar memberikan konsistensi pada setiap eksekusi program. Hal ini membantu pemain memperoleh hasil pengenalan nada yang lebih akurat dan stabil [14]. Permasalahan yang muncul adalah adanya ambiguitas posisi nada pada *fretboard*. Solusi dalam mengatasi hal tersebut salah satunya dengan merancang sistem yang mampu mendeteksi *pitch* monofonik dan menyusun urutan *pitch* tersebut ke posisi senar dan *fret* yang realistis dalam bentuk *tablature*.

Berbagai penelitian terdahulu telah mengeksplorasi deteksi nada dan pembangkitan *tablature* berbasis *deep learning*. Salah satunya adalah TabCNN [15], model *convolutional neural network* yang mengintegrasikan estimasi *multipitch* dengan posisi fret gitar melalui pemetaan audio langsung ke *tablature*. Dilatih menggunakan dataset GuitarSet, model ini menunjukkan performa yang lebih unggul dibandingkan metode *multipitch* lainnya. Selain itu, Jadhav dkk. (2022) memanfaatkan CNN dan *transfer learning* dengan mengonversi sinyal audio menjadi spektrogram *Constant-Q Transform* untuk mendeteksi akor solo gitar dengan akurasi 88,7%. Meskipun pendekatan tersebut efektif untuk sinyal polifonik dan representasi dua dimensi, masih sedikit penelitian yang memisahkan tahap deteksi *pitch*

monofonik dari pemetaan urutan *pitch* ke *tablature* menggunakan model sekuensial. Celah ini sangat krusial, terutama untuk skenario sumber audio monofonik yang bervariasi, seperti vokal atau instrumen tunggal.

Dalam konteks deteksi *pitch*, algoritma YIN banyak digunakan karena akurasinya yang tinggi. Algoritma ini menunjukkan tingkat kesalahan yang rendah dalam estimasi frekuensi fundamental pada berbagai kondisi [16]. YIN juga terbukti memberikan estimasi *pitch* yang akurat pada berbagai skenario aplikasi [17]. Pada lingkungan bising, YIN mempertahankan akurasi deteksi *pitch* yang baik. Algoritma ini lebih tangguh terhadap *noise* latar dibandingkan beberapa metode alternatif [18]. Dari sisi efisiensi, YIN cukup ringan untuk skenario pemrosesan waktu nyata. Kompleksitas komputasi YIN lebih rendah dibandingkan metode seperti *Extended Complex Kalman Filter* (ECKF) [19]. YIN juga mampu mengikuti kontur nada yang dinamis. Algoritma ini menunjukkan hasil yang baik dengan variasi *pitch* yang cepat [20]. Karakteristik tersebut menjadikan YIN kandidat yang sesuai untuk ekstraksi frekuensi fundamental dari sinyal audio monofonik.

Setelah *pitch* diekstraksi, selanjutnya adalah menyusun urutan nada ke posisi senar dan *fret* pada gitar. Posisi tersebut perlu konsisten dan ergonomis dalam konteks permainan gitar. *Long Short-Term Memory* (LSTM) merupakan arsitektur jaringan saraf yang relevan untuk penelitian ini. LSTM dirancang untuk menangkap ketergantungan jangka panjang dalam data berurutan, termasuk struktur temporal pada musik [21]. Penelitian lain menunjukkan bahwa LSTM dapat menghasilkan musik yang mengikuti struktur yang ada dan memiliki koherensi musikal [22]. LSTM juga fleksibel untuk dilatih pada dataset musik yang besar dan dapat mempelajari karakteristik gaya tertentu. Dalam konteks *tablature* gitar, LSTM dapat dipandang sebagai pemetaan sekuensial dari deret *pitch* hasil deteksi YIN ke deret token *string* dan *fret*. Konteks nada sebelumnya dapat dimanfaatkan untuk menghasilkan *tablature* lebih realistis.

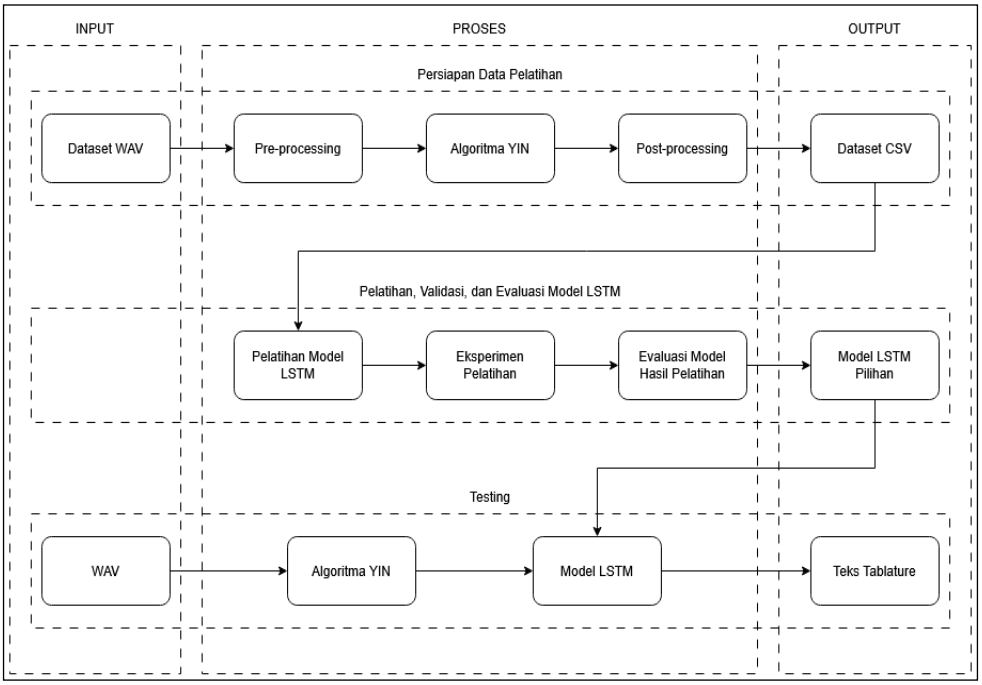
Persoalan utama dalam penelitian ini adalah adanya ambiguitas posisi nada pada *fretboard*. Dalam penelitian ini, perumusan masalahnya adalah bagaimana merancang sistem yang mampu mendeteksi *pitch* monofonik dari *file* audio berformat WAV menggunakan algoritma YIN dan mengonversi urutan *pitch* tersebut menjadi notasi *tablature* gitar. Sistem ini ditujukan untuk membantu pemain gitar, khususnya pemula dan non-musisi, dalam mengatasi keterbatasan kepekaan pendengaran terhadap nada dan kesulitan memetakan nada ke posisi pada *fretboard*.

Tujuan penelitian ini adalah menerapkan algoritma YIN untuk melakukan deteksi *pitch* monofonik dari audio dan menghasilkan representasi nada dalam bentuk frekuensi atau MIDI. menggunakan model LSTM untuk menyusun urutan nada tersebut ke format *tablature* gitar berupa kombinasi senar dan *fret*. Kebaruan penelitian terletak pada integrasi *pipeline* deteksi *pitch* menggunakan YIN dengan pemodelan sekuensial LSTM untuk melakukan estimasi *tablature* monofonik. Penelitian ini berfokus pada pemetaan frekuensi fundamental ke *token* senar dan *fret* pada gitar. Pendekatan ini berbeda dengan penelitian sebelumnya yang banyak bertumpu pada CNN berbasis spektrogram dan sinyal polifonik [3][15]. Penelitian ini mengeksplorasi kombinasi YIN dan LSTM untuk membangun sistem audio ke *tablature* sebagai media bantu pembelajaran bagi musisi pemula dan non-musisi.

2. METODOLOGI

Penelitian ini menggunakan teknik pendekatan eksperimental yang bertujuan untuk menguji dan menganalisis kinerja algoritma deteksi *pitch* fundamental menggunakan YIN *Algorithm* sebagai metode

ekstraksi fitur frekuensi dan *Long Short-Term Memory* (LSTM) sebagai model pemrosesan data sekuensial. Pendekatan eksperimental dipilih karena penelitian ini berfokus pada pengujian model yang dikembangkan untuk mendeteksi nada dari sinyal audio, kemudian melakukan pemetaan menjadi notasi gitar yaitu *tablature* sebagai media yang lebih mudah dibaca oleh pemain gitar.



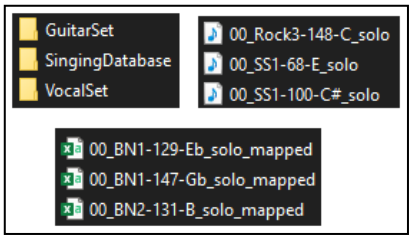
Gambar 1. Diagram Blok Penelitian

Pada fase *training*, frekuensi fundamental yang diperoleh dicocokkan dengan tabel nama nada sebagai label target yang digunakan untuk melatih model LSTM agar mampu mengenali posisi nada pada senar dan *fret*. Fase *validation* dilakukan dengan memproses data validasi serupa seperti pada *training* dan mengevaluasi performa model LSTM berdasarkan hasil pencocokan *pitch* ke tabel nama nada sehingga dapat dipilih model berdasarkan hasil evaluasi pelatihan.

Terakhir pada fase testing, setelah proses *pre-processing* dan deteksi *pitch* selesai serta frekuensi dicocokkan ke tabel nama nada, dilakukan pemetaan notasi gitar berdasarkan *output pitch* sehingga menghasilkan *tablature* gitar sebagai *output* akhir dari sistem. Dengan struktur urutan proses tersebut diharapkan dapat memberikan hasil pemetaan notasi gitar yang sesuai antara nada dengan posisi senar dan *fret*, serta berguna untuk dijadikan aplikasi musik digital maupun pembelajaran instrumen gitar.

2.1 Dataset

Dataset yang digunakan bersumber dari GuitarSet, VocalSet, dan SingingDatabase dengan format audio WAV sejumlah 3.879 *file* WAV yang kemudian dikonversi oleh proses deteksi *pitch* menggunakan YIN menjadi format CSV sejumlah 3.788 *file* CSV. Dataset CSV berisi kolom yang terdiri dari: hz, note, midi, string, fret, token_index.



Gambar 2. Dataset dalam bentuk WAV dan CSV

	A	B	C	D	E	F
1	hz	note	midi	string	fret	token_idx
2	158.2391	D#3	51	5	6	31
3	176.9125	F3	53	5	8	33
4	166.7227	E3	52	5	7	32
5	175.3098	F3	53	5	8	33
6	222.0712	A3	57	4	7	57
7	234.0827	A#3	58	4	8	58
8	221.3494	A3	57	4	7	57
9	295.5128	D4	62	3	7	82

Gambar 3. Isi dataset CSV

a. hz adalah nilai frekuensi fundamental.
b. note adalah nada yang terdeteksi.
c. midi adalah kode midi yang relevan dengan frekuensi atau nada hasil deteksi.
d. string adalah letak senar nada tersebut berada.
e. fret adalah letak fret nada tersebut berada.
f. token_idx adalah kode token dari tokenisasi yang merupakan kombinasi posisi string dan fret.

2.2 *Pre-Processing*

Tahapan *pre-processing* dilakukan untuk membersihkan sinyal audio monofonik dari berbagai *noise* yang dapat mengganggu proses ekstraksi *pitch* menggunakan algoritma YIN. Tahapan ini memastikan bahwa sinyal yang masuk ke tahap analisis *pitch* memiliki kualitas yang stabil, minim gangguan, serta fokus pada komponen nada dominan.

Pada penelitian ini, *pre-processing* terdiri dari beberapa langkah, termasuk *Harmonic–Percussive Source Separation* (HPSS) dan *Bandpass Filter*.

a. *Harmonic–Percussive Source Separation* (HPSS)
   HPSS digunakan untuk memisahkan sinyal audio menjadi komponen *harmonic* dan *percussive*. Proses ini bertujuan untuk mempertahankan bagian sinyal yang mengandung nada stabil, sekaligus menghilangkan suara transien seperti petikan, gesekan jari, serta *noise* yang dapat mengganggu proses deteksi *pitch*. Komponen *harmonic* hasil pemisahan kemudian digunakan sebagai dasar analisis karena memiliki karakteristik yang lebih stabil dan sesuai dengan kebutuhan deteksi *pitch* monofonik.
b. *Bandpass Filter*
   Bandpass Filter digunakan untuk membatasi sinyal audio pada rentang frekuensi yang relevan dengan instrumen gitar monofonik untuk menghilangkan frekuensi rendah yang tidak memiliki informasi *pitch* serta frekuensi tinggi yang cenderung mengandung *noise*. Penerapan *bandpass* filter menghasilkan sinyal yang lebih bersih dan fokus pada frekuensi fundamental.

2.3 Algoritma YIN

Setelah melalui tahap *pre-processing*, sinyal audio kemudian diproses untuk mendeteksi nilai *pitch* menggunakan algoritma YIN. Pada penelitian ini, proses deteksi dilakukan secara *frame-based*, di mana sinyal dibagi ke dalam interval waktu tertentu untuk memperoleh nilai *pitch* pada setiap segmen.

Algoritma YIN digunakan karena mampu memberikan deteksi *pitch* yang stabil pada sinyal monofonik dan memiliki ketahanan terhadap fluktuasi amplitudo. Pada tahap ini, yang diambil adalah nilai *pitch* fundamental dari setiap *frame*, sehingga dapat merepresentasikan perubahan nada sepanjang durasi audio. Hasil deteksi *pitch* kemudian dikonversi menjadi tiga informasi utama, yaitu:

a. Nilai frekuensi (Hz) sebagai keluaran langsung dari algoritma.
b. Nama nada (*note*) berdasarkan konversi frekuensi ke sistem nada musik.
c. Nilai MIDI sebagai representasi numerik dari nada yang terdeteksi.

Ketiga informasi ini selanjutnya digunakan sebagai dasar untuk tahap selanjutnya yaitu menentukan posisi senar dan *fret* dari nada yang terdeteksi.

2.4 *Post-Processing*

Tahap *post-processing* pada penelitian ini dilakukan untuk menstabilkan hasil deteksi *pitch* sehingga data yang digunakan pada proses *mapping* tidak mengandung fluktuasi atau kesalahan minor yang dapat mengganggu akurasi *tablature*. Terdapat tiga langkah utama, yaitu RMS *Noise Filtering*, *Block Sampling*, dan *Cent Stabilizer*.

a. RMS *Noise*
   RMS (*Root Mean Square*) digunakan untuk mengidentifikasi *frame-frame* yang memiliki energi sangat rendah. *Frame* dengan RMS di bawah ambang tertentu dianggap tidak mengandung informasi *pitch* yang valid, karena biasanya berasal dari *noise* latar, bagian hening, atau artefak kecil sebelum atau sesudah sebuah nada dimainkan. *Frame-frame* tersebut dihapus atau ditandai sebagai *silence*, sehingga tidak dihitung sebagai *pitch* dalam proses berikutnya.
b. *Block Sampling*
   Block sampling digunakan untuk merapikan hasil deteksi *pitch* yang terkadang berubah terlalu cepat antar *frame*. Nilai *pitch* dikelompokkan ke dalam blok waktu tertentu, lalu setiap blok diwakili oleh nilai *pitch* yang paling stabil atau dominan.
c. *Cent Stabilizer*
   Cent Stabilizer digunakan untuk menstabilkan perbedaan *pitch* kecil yang masih muncul setelah proses YIN. Fluktuasi antar *frame* yang tidak mencapai satu *semitone* dikoreksi dengan cara menyesuaikan nilai *pitch* ke *cent* terdekat. Langkah ini menghasilkan garis *pitch* yang lebih halus dan mengurangi kesalahan konversi dari frekuensi ke nama nada. *Stabilizer* ini juga membantu menghindari kondisi di mana YIN mendeteksi *pitch* yang sedikit melenceng ke nada lain akibat vibrasi atau *noise* kecil pada sinyal.

2.5 Pelatihan Model LSTM

Pada penelitian ini, model LSTM digunakan untuk memetakan urutan *pitch* hasil deteksi YIN menjadi urutan posisi *tablature* gitar berupa pasangan *string* dan *fret*. Dilakukan 3 eksperimen konfigurasi model dengan nilai *hyperparameter* yang berbeda.

Tabel 1. Daftar Eksperimen

No.	Eksperimen Model	Deskripsi Model
1	LSTM Baseline	Model LSTM dasar tanpa optimasi atau penambahan parameter.
2	LSTM dengan optimasi nilai *hyperparameter* menggunakan Optuna	Optimasi *hyperparameter* LSTM menggunakan pencarian otomatis Optuna.
3	LSTM dengan penambahan parameter constraint	LSTM dengan pembatasan aturan gitar untuk hasil lebih kontekstual mirip manusia.

3. HASIL DAN PEMBAHASAN

3.1 Hasil Pelatihan Model

Proses pelatihan dari ketiga konfigurasi model yang disajikan Tabel 1 secara rinci hasilnya adalah:

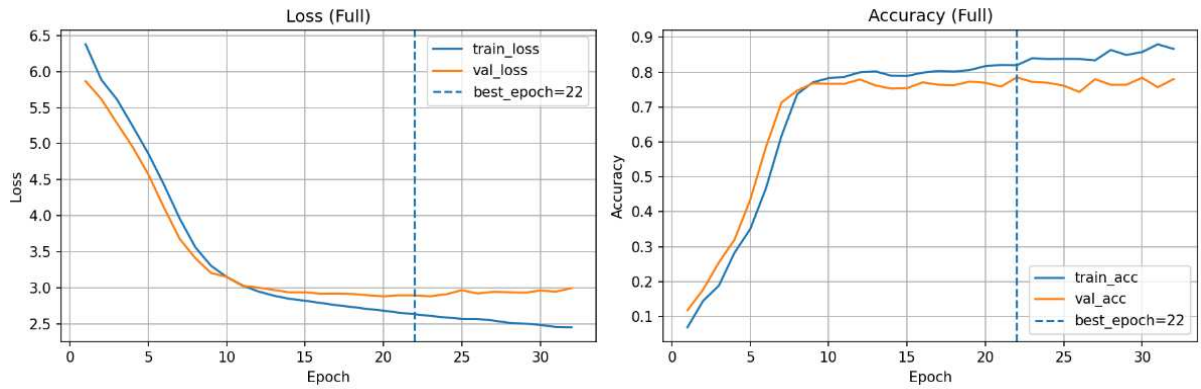
a. Model LSTM Baseline

Dengan konfigurasi dasar, model LSTM Baseline mampu mempelajari pemetaan urutan nada hasil estimasi frekuensi oleh YIN ke token *string* dan *fret* gitar. Model dilatih menggunakan SEQ_LEN = 64 dan SEQ_HOP = 32. Proses pelatihan menggunakan *batch size* sebesar 512, yang memberikan stabilitas gradien dan konvergensi yang relatif halus selama pelatihan.

Arsitektur model terdiri dari *embedding* nada dengan dimensi EMBED_DIM = 72, diikuti oleh dua lapisan LSTM dari NUM_LAYERS = 2 dengan ukuran HIDDEN_SIZE = 512. Konfigurasi ini memberikan kapasitas representasi yang cukup besar untuk menangkap hubungan antara urutan nada dan posisinya pada gitar. Untuk mengurangi risiko *overfitting*, diterapkan *dropout* sebesar 0,25. Pelatihan dirancang berlangsung hingga 300 *epoch*, namun dilengkapi *early stopping* dengan PATIENCE = 10 *epoch*.

Tabel 2. Hyperparameter model LSTM Baseline

Hyperparameter	Nilai
SEQ_LEN	64
SEQ_HOP	32
BATCH_SIZE	512
EMBED_DIM	72
HIDDEN_SIZE	512
NUM_LAYERS	2
DROPOUT	0.25
EPOCHS	300
LEARNING_RATE	0.003
WEIGHT_DECAY	0.0
PATIENCE	10



Gambar 4. Kurva pelatihan model LSTM Baseline

Berdasarkan hasil pelatihan, model mencapai akurasi validasi maksimum sebesar 0,785 pada *epoch* ke-22, sebelum performa validasi cenderung stagnan dan pelatihan dihentikan oleh *early stopping*. Akurasi pelatihan telah mencapai sekitar 0,82, menunjukkan bahwa model mampu mempelajari pola pada data latih, dengan tetap mempertahankan kemampuan generalisasi yang cukup stabil pada data validasi. Nilai loss pelatihan dan validasi juga menunjukkan penurunan pada fase awal hingga menengah pelatihan, yang mengindikasikan proses optimisasi berjalan pada saat pelatihan.

Secara keseluruhan, eksperimen baseline ini digunakan sebagai acuan awal untuk mengevaluasi pengaruh berbagai pengembangan selanjutnya, seperti penyesuaian *hyperparameter* dan penggunaan teknik regularisasi tambahan. Hasil baseline ini memberikan gambaran awal mengenai kemampuan model LSTM dalam memetakan urutan nada ke *tablatur*e gitar, serta menjadi referensi dalam menilai peningkatan performa pada tahap eksperimen berikutnya.

b. Model LSTM dengan optimasi nilai *hyperparameter* menggunakan Optuna

Dengan penerapan optimasi *hyperparameter* menggunakan Optuna, model LSTM menunjukkan peningkatan dalam menyesuaikan konfigurasi arsitektur dan parameter pelatihan terhadap karakteristik data. Proses pencarian dilakukan sebanyak 100 *trial*, di mana Optuna secara otomatis mengeksplorasi kombinasi parameter untuk memaksimalkan akurasi validasi sebagai metrik utama evaluasi. Konfigurasi terbaik yang diperoleh dari Optuna kemudian digunakan sebagai konfigurasi model LSTM.

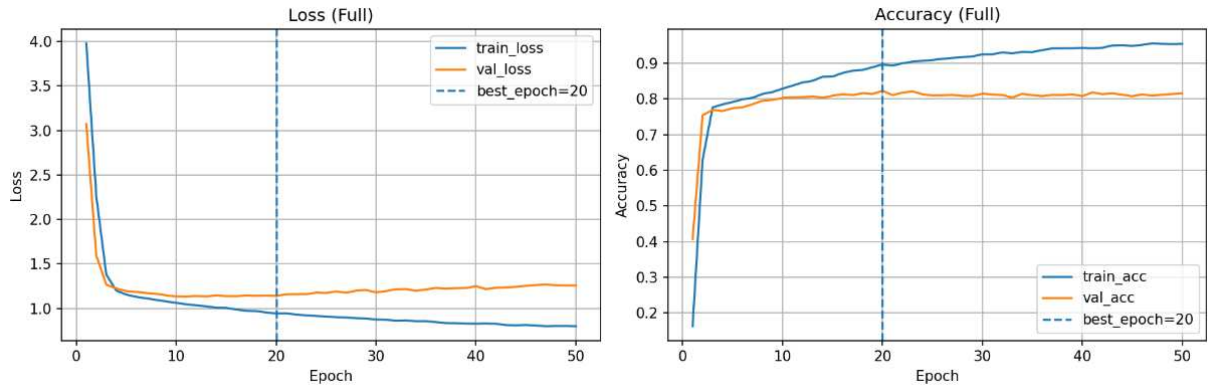
Pada konfigurasi hasil optimasi, panjang urutan *input* tetap dipertahankan pada SEQ_LEN = 64, namun hop diperkecil menjadi SEQ_HOP = 16, sehingga model menerima konteks temporal yang lebih rapat dan detail. Ukuran *batch* berkurang menjadi 32, memungkinkan pembaruan parameter yang lebih sering dan adaptif selama pelatihan. Dimensi embedding nada juga diperkecil menjadi 32, sehingga representasi *input* menjadi lebih ringkas tanpa *menghilangkan* informasi penting.

Tabel 3. Hyperparameter model LSTM dengan optimasi nilai hyperparameter menggunakan Optuna

Hyperparameter	Nilai
SEQ_LEN	64
SEQ_HOP	16
BATCH_SIZE	32
EMBED_DIM	32
HIDDEN_SIZE	64
NUM_LAYERS	2
DROPOUT	0.4526926956977294
EPOCHS	100

Hyperparameter	Nilai
LEARNING_RATE	0.004298322053220097
WEIGHT_DECAY	0.00023906800333279278
PATIENCE	30

Arsitektur LSTM hasil Optuna menggunakan dua lapisan LSTM dengan ukuran *hidden state* 64 unit, yang jauh lebih kecil dibandingkan model baseline. Penurunan kapasitas ini diimbangi dengan penerapan *dropout* yang relatif tinggi, yang berfungsi sebagai regularisasi kuat untuk menekan *overfitting*. Jumlah *epoch* maksimum ditetapkan 100 *epoch*, dengan mekanisme *early stopping* yang lebih toleran melalui *patience* sebesar 30 *epoch*, sehingga model memiliki ruang yang cukup untuk mencapai performa optimal.



Gambar 5. Kurva pelatihan model LSTM dengan optimasi nilai hyperparameter menggunakan Optuna

Hasil pelatihan menunjukkan bahwa model LSTM yang dioptimasi dengan Optuna memiliki performa validasi yang lebih stabil dibandingkan model *baseline*, sebagaimana diilustrasikan pada Gambar 5. Hal ini mengindikasikan bahwa penyesuaian *hyperparameter* mampu menemukan konfigurasi yang lebih seimbang antara kapasitas model dan kemampuan generalisasi. Oleh karena itu, model ini digunakan sebagai perbandingan untuk menilai dampak optimasi *hyperparameter* terhadap kinerja hasil pelatihan.

c. Model LSTM dengan penambahan parameter *constraint*

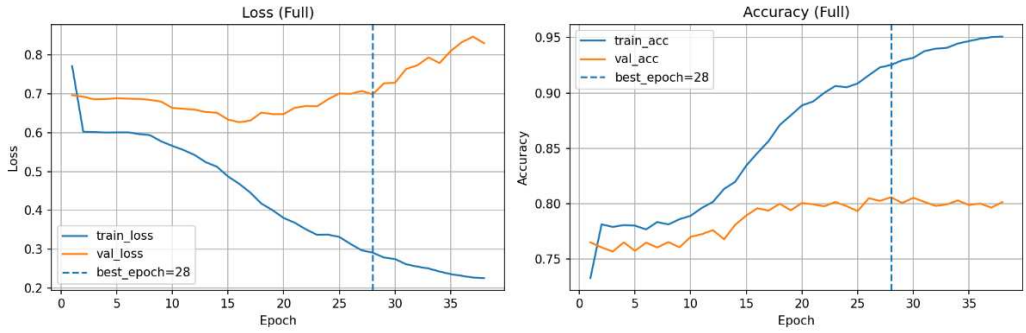
Tabel 4. Hyperparameter Model LSTM dengan penambahan parameter constraint

Hyperparameter	Nilai
SEQ_LEN	64
SEQ_HOP	16
BATCH_SIZE	32
EMBED_DIM	32
HIDDEN_SIZE	64
NUM_LAYERS	2
DROPOUT	0.4526926956977294
EPOCHS	100
LEARNING_RATE	0.004298322053220097
WEIGHT_DECAY	0.00023906800333279278
PATIENCE	10
ERGO_FRET_LIMIT	2
STRING_JUMP_LIMIT	1

Hyperparameter	Nilai
REGION_SIZE	5
REGION_JUMP_LIMIT	1
LAMBDA_FRET_JUMP	1
LAMBDA_STRING_JUMP	0.5
LAMBDA_REGION_SHIFT	0.5

Dengan penambahan *constraint* berbasis aturan *fingering*, model LSTM dengan parameter *constraint* dikembangkan untuk tidak hanya memaksimalkan ketepatan prediksi token, tetapi juga menjaga kewajaran transisi posisi tangan pada gitar. Model dilatih menggunakan panjang urutan SEQ_LEN = 64 dengan pergeseran SEQ_HOP = 16, sehingga perubahan antar nada dapat dimodelkan secara lebih rapat dan berkesinambungan. Proses pelatihan dilakukan dengan *batch size* sebesar 32, yang memungkinkan pembaruan parameter berlangsung lebih adaptif terhadap variasi data. Arsitektur model mempertahankan *embedding* nada berdimensi 32, diikuti oleh dua lapisan LSTM dengan ukuran *hidden state* 64 unit. Untuk mengendalikan kompleksitas model dan mengurangi kecenderungan *overfitting*, diterapkan *dropout* sebesar 0,45. Pelatihan dirancang hingga 100 *epoch*, namun dilengkapi dengan mekanisme *early stopping* dengan *patience* 10 *epoch*, di mana akurasi validasi digunakan sebagai kriteria utama penghentian pelatihan.

Berbeda dengan model sebelumnya, konfigurasi ini menerapkan *constraint* eksplisit pada fungsi *loss*, yang membatasi prediksi model agar tetap sesuai dengan aturan ergonomi *fingering* gitar. *Constraint* tersebut meliputi pembatasan perpindahan *fret* (ERGO_FRET_LIMIT = 2), lompatan antar senar (STRING_JUMP_LIMIT = 1), serta pergeseran wilayah posisi tangan (REGION_JUMP_LIMIT = 1 dengan REGION_SIZE = 5). Setiap pelanggaran terhadap batasan ini dikenai penalti tambahan melalui parameter bobot LAMBDA_FRET_JUMP, LAMBDA_STRING_JUMP, dan LAMBDA_REGION_SHIFT, sehingga model diarahkan untuk memilih prediksi yang lebih konsisten secara fisik meskipun terdapat beberapa alternatif token yang valid secara nada.



Gambar 6. Kurva pelatihan model LSTM dengan penambahan parameter constraint

Berdasarkan kurva pelatihan pada Gambar 6, terlihat bahwa model LSTM dengan *constraint* mengalami pemisahan yang jelas antara performa data latih dan validasi setelah beberapa *epoch* awal. Nilai *training loss* menurun secara konsisten hingga akhir pelatihan, sementara *validation loss* hanya menurun hingga sekitar *epoch* ke-28 sebelum kembali meningkat, yang menandakan mulai terjadinya *overfitting*. Pola yang sama juga terlihat pada kurva akurasi, di mana *training accuracy* terus meningkat hingga mendekati 0,95, sedangkan *validation accuracy* cenderung stagnan di kisaran 0,80. *Epoch* ke-28 ditandai sebagai *epoch* terbaik karena memberikan keseimbangan optimal antara kemampuan belajar model dan generalisasi pada data validasi. Secara keseluruhan, kurva ini menunjukkan bahwa meskipun

constraint membantu menjaga struktur pemetaan, model tetap memiliki kecenderungan *overfitting* jika dilatih terlalu lama, sehingga penggunaan *early stopping* menjadi komponen penting dalam konfigurasi pelatihan.

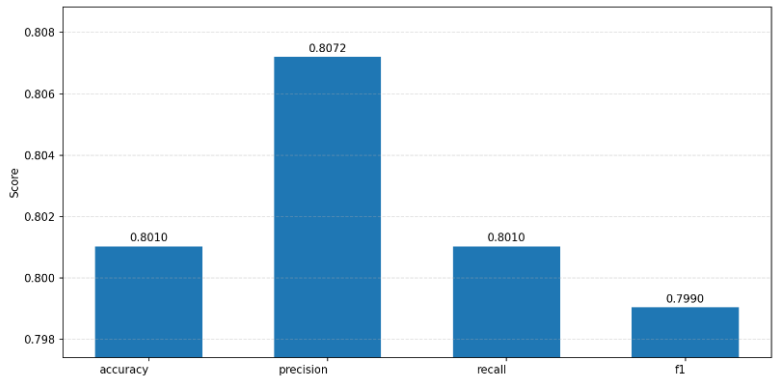
3.2 Evaluasi Kinerja Model

Evaluasi dilakukan untuk menilai kinerja sistem dalam memetakan urutan nada hasil estimasi frekuensi oleh YIN ke representasi token *tablature* gitar. Evaluasi model pada penelitian ini dilakukan menggunakan dataset *test* yang sudah dipisahkan dari semua dataset sebelumnya. Evaluasi difokuskan pada pendekatan *token based*, di mana setiap token senar dan *fret* dibandingkan secara langsung dengan token *ground truth*.

a. Model LSTM Baseline

Gambar 7 menunjukkan nilai *accuracy* sebesar 0,8010 yang artinya bahwa sekitar 80,1% token hasil pilihan model identik dengan token *ground truth* pada evaluasi *token based mapping* dari hasil estimasi YIN ke token *string* dan *fret*. Karena evaluasi dilakukan pada level token, setiap perbedaan token, meskipun masih merepresentasikan nada yang sama dengan posisi berbeda tetap dianggap sebagai kesalahan. *Precision* sebesar 0,8072 mengindikasikan bahwa dari seluruh token yang dipilih model, sekitar 80,7% sesuai dengan token referensi, yang dipengaruhi oleh kecenderungan model memilih representasi *string* dan *fret* tertentu serta adanya variasi token yang secara musikal sama tetapi berbeda indeks. *Recall* sebesar 0,8010, yang bernilai sama dengan akurasi, menunjukkan bahwa proporsi token *ground truth* yang berhasil dipilih secara tepat sebanding dengan akurasi keseluruhan, suatu kondisi yang wajar pada evaluasi multi-kelas berbasis token di mana *true positive* hanya dihitung ketika token benar-benar identik dan konsep *true negative* tidak memiliki makna musikal.

Pada Gambar 7, F1-*score* sebesar 0,7990 mencerminkan keseimbangan antara kecenderungan model dalam menghasilkan token tertentu dan kemampuannya mencakup token *ground truth*, dengan selisih antar metrik yang relatif kecil menandakan distribusi kesalahan yang seimbang tanpa dominasi kesalahan satu arah. Secara keseluruhan, konsistensi nilai metrik di kisaran 0,80 menunjukkan bahwa performa model sangat dipengaruhi oleh definisi benar atau salah pada level token.

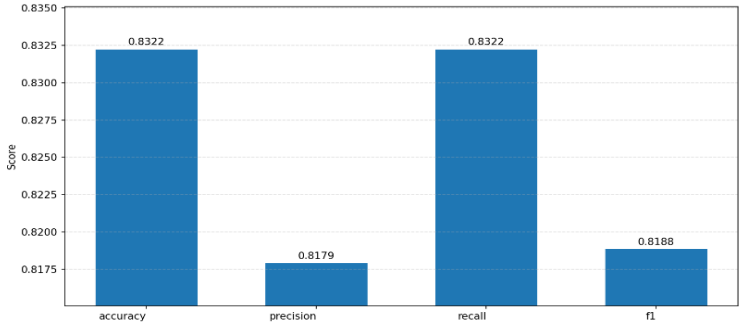


Gambar 7. Hasil evaluasi model LSTM Baseline

b. Model LSTM dengan optimasi nilai *hyperparameter* menggunakan Optuna

Gambar 8 menunjukkan nilai *accuracy* sebesar 0,8322 yakni bahwa sekitar 83,22% token hasil model identik dengan token *ground truth* pada evaluasi token *based string* dan *fret*, yang menandakan adanya perubahan distribusi token dibandingkan model LSTM baseline setelah proses pencarian *hyperparameter* menggunakan Optuna. *Precision* sebesar 0,8179 mengindikasikan bahwa sebagian

token yang dipilih model tidak memiliki kecocokan identik dengan token referensi, kondisi yang dapat muncul ketika model memilih posisi alternatif yang secara musikal merepresentasikan nada yang sama namun tetap dihitung sebagai kesalahan dalam evaluasi. *Recall* yang bernilai sama dengan akurasi, yaitu 0,8322, menunjukkan bahwa proporsi token *ground truth* yang berhasil dipilih secara identik sama dengan rasio benar secara keseluruhan, suatu karakteristik yang konsisten pada skema evaluasi *multi class token based* di mana setiap token hanya dapat menghasilkan satu *true positive* dan kesalahan dianggap sebagai *false negative*.

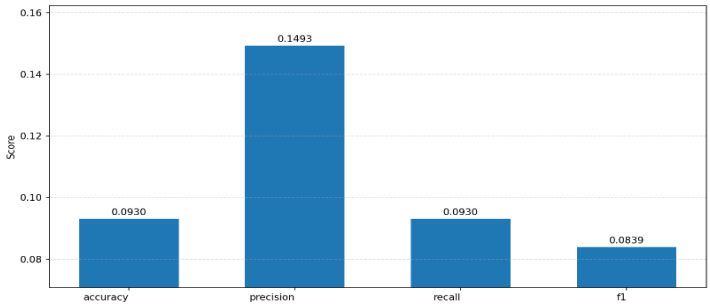


Gambar 8. Hasil evaluasi model LSTM dengan optimasi nilai *hyperparameter* menggunakan Optuna

Pada Gambar 8, F1-*score* sebesar 0,8188 mencerminkan keseimbangan antara ketepatan token yang dipilih dan cakupan token referensi, dengan posisinya yang berada di antara nilai *precision* dan *recall*. Secara keseluruhan, kesamaan nilai *accuracy* dan *recall* serta *precision* yang sedikit lebih rendah menegaskan bahwa seluruh metrik dipengaruhi kesamaan pada level token, bukan pada kebenaran nada atau *pitch* yang direpresentasikan.

c. Model LSTM dengan penambahan parameter *constraint*

Gambar 9 menunjukkan nilai *accuracy* sebesar 0,0930 yakni hanya sekitar 9,3% token hasil dari model yang identik secara indeks dengan token *ground truth*, sehingga mayoritas token yang dipilih model berbeda dari referensi meskipun dapat merepresentasikan hasil musikal yang serupa. *Precision* sebesar 0,1493 mengindikasikan bahwa dari seluruh token yang dipilih model, sekitar 14,9% memiliki kecocokan indeks dengan token *ground truth*, yang berarti ketika model memilih token tertentu sebagian memang sesuai dengan referensi, namun cakupan terhadap token *ground truth* secara keseluruhan masih terbatas. *Recall* yang bernilai sama dengan akurasi, yaitu 0,0930, menegaskan bahwa hanya sebagian kecil token *ground truth* yang berhasil dipilih secara identik, suatu kondisi yang konsisten dengan evaluasi *multi class* berbasis token di mana kecocokan hanya dihitung benar jika indeks token sama persis.

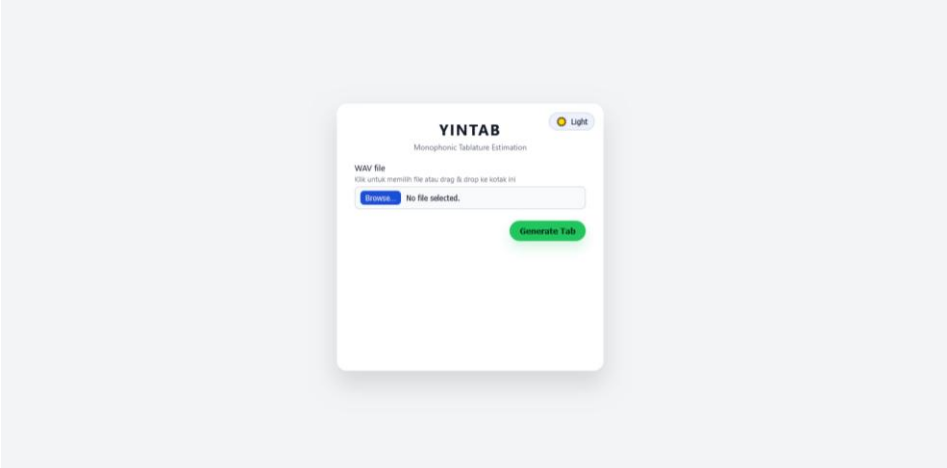


Gambar 9. Hasil evaluasi model LSTM dengan penambahan parameter *constraint*

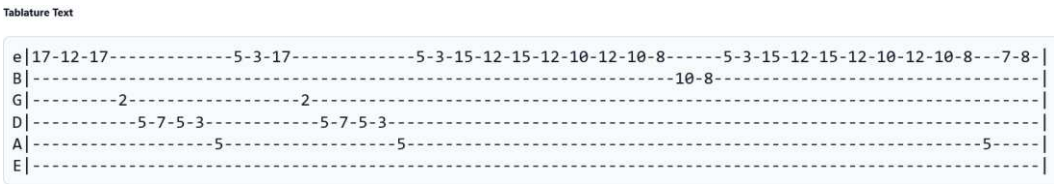
Pada Gambar 9, F1-*score* sebesar 0,0839 mencerminkan keseimbangan *precision* dan *recall* berada pada nilai yang lebih rendah akibat ketimpangan cakupan token *ground truth*. Nilai ini menunjukkan bahwa penerapan *constraint* tidak sepenuhnya meniadakan kemungkinan pemilihan token *ground truth*, tetapi secara signifikan menggeser distribusi token ke token-token alternatif yang berbeda indeks, sehingga hasilnya dapat terdengar musikal serupa namun jarang cocok secara token. Hal ini sejalan dengan karakteristik *constraint* yang membatasi transisi dan mendorong ergonomi tanpa memprioritaskan kesesuaian token *ground truth*, sehingga pada model ini kecocokan indeks hanya terjadi pada sebagian kecil urutan. Oleh karena itu, karena model sebelumnya dioptimalkan untuk mereplikasi distribusi token *ground truth* sementara model ini dioptimalkan untuk mengikuti *constraint* ergonomi, hasil model ini tidak dapat dibandingkan secara langsung dengan sebelumnya.

3.3 Implementasi Aplikasi

Untuk memudahkan penggunaan sistem oleh pengguna, model diimplementasikan dalam bentuk aplikasi berbasis web. Dengan aplikasi ini, pengguna hanya perlu membuka halaman web lokal, *upload file* WAV, dan sistem secara otomatis menjalankan seluruh proses transkripsi audio monofonik menjadi *tablature* tanpa perlu berinteraksi langsung dengan kode program. *Website* ini dikembangkan menggunakan *framework* Python yaitu Flask. Pada penelitian ini *website* dijalankan secara lokal pada URL http://127.0.0.1:5000/. Berikut tampilan halaman pada *website*:



Gambar 10. Halaman input



Gambar 11. Tablature text

Pada Gambar 10 terdapat *button* "Browse..." untuk memilih *file input* berupa audio WAV. Pemilihan *file input* bisa juga dilakukan dengan cara *drag and drop*. Setelah *file input* dipilih bisa dilakukan pemetaan *tablature* oleh model LSTM dengan menekan *button* "Generate Tab". Setelah menekan *button* "Generate Tab", akan menghasilkan *tablature text*. *Tablature text* adalah *tablature* hasil dari pemetaan oleh model LSTM dalam format *text*.

4. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, pelatihan model, serta pengujian yang dilakukan dalam penelitian ini, dapat disimpulkan algoritma YIN berhasil diterapkan untuk mendeteksi *pitch* monofonik dari *file* audio berformat WAV. Dengan bantuan tahapan *pre-processing* HPSS dan *bandpass filter* serta *post-processing*, YIN mampu menghasilkan deret frekuensi nada yang relatif stabil dan representatif terhadap konten nada utama pada sinyal audio. Hal ini membuat hasil YIN layak digunakan sebagai fitur *input* bagi model pemetaan ke *tablature* gitar. Model LSTM berhasil dilatih menggunakan dataset beranotasi yang berisi pasangan urutan nada dan label posisi senar dan *fret* yang direpresentasikan sebagai token indeks. Proses pelatihan memanfaatkan konfigurasi *hyperparameter* sehingga model mampu mempelajari pola hubungan antara urutan nada dan pilihan posisi senar *fret* yang sesuai.

Penelitian ini mengevaluasi tiga varian model LSTM untuk pemetaan urutan nada hasil estimasi *pitch* ke representasi token *string* dan *fret* gitar, yaitu LSTM Baseline, LSTM dengan optimasi *hyperparameter* menggunakan Optuna, dan LSTM dengan penerapan *constraint*, dengan evaluasi dilakukan pada data uji terpisah menggunakan metrik berbasis kecocokan token indeks. Model LSTM Baseline menunjukkan nilai *accuracy*, *precision*, *recall*, dan F1-*score* yang berada pada rentang serupa, mencerminkan bahwa distribusi prediksi token mengikuti distribusi token *ground truth* tanpa pembatasan tambahan. Model LSTM dengan optimasi Optuna memperlihatkan perubahan distribusi prediksi sebagai dampak pencarian *hyperparameter*, dengan peningkatan pada *accuracy* dan *recall* dibandingkan model *baseline*, sementara *precision* dan F1-*score* tetap berada pada kisaran yang sebanding, yang menunjukkan adanya penyesuaian keseimbangan antara cakupan token *ground truth* dan pemilihan token prediksi tanpa mengubah sifat dasar evaluasi *token based*. Sebaliknya, model LSTM dengan *constraint* menghasilkan pola evaluasi yang berbeda secara signifikan, ditandai oleh nilai *accuracy*, *recall*, dan F1-*score* yang jauh lebih rendah, serta *precision* yang relatif lebih tinggi dibandingkan *recall*, yang mengindikasikan bahwa penerapan *constraint* menggeser distribusi prediksi ke token-token alternatif yang memenuhi aturan tertentu sehingga kecocokan indeks dengan *token ground truth* menjadi terbatas, meskipun secara musikal keluaran model dapat tampak serupa, perbedaan representasi token menyebabkan sebagian besar prediksi tidak dihitung sebagai benar dalam evaluasi *token-based* yang ketat.

Secara keseluruhan, hasil ini menunjukkan bahwa masing-masing model mengoptimalkan tujuan yang berbeda, di mana model *baseline* dan optimasi Optuna berfokus pada reproduksi token *ground truth*, sedangkan model dengan *constraint* berfokus pada kepatuhan terhadap aturan eksternal, sehingga perbandingan kinerja antar model menggunakan metrik *token-based* perlu diinterpretasikan secara hati-hati karena metrik tersebut tidak sepenuhnya merepresentasikan kesetaraan musikal dari hasil prediksi, sekaligus menegaskan pentingnya pemilihan metrik evaluasi yang selaras dengan tujuan pelatihan dan representasi *output*. Rangkaian pemrosesan mulai dari deteksi *pitch* dengan YIN hingga pemetaan posisi senar dan *fret* dengan LSTM mampu mengubah audio monofonik menjadi urutan token yang kemudian di-*render* sebagai *tablature* gitar. Hasil *tablature* menunjukkan bahwa sistem dapat merekonstruksi nada dan posisi senar dan *fret*, meskipun masih terdapat perbedaan pada beberapa nada maupun pilihan posisi jari. Sistem ini sudah memenuhi tujuan penelitian memetakan nada monofonik menjadi *tablature* dan dapat dijadikan dasar pengembangan lebih lanjut.

DAFTAR PUSTAKA

[1] M. Z. A. Rachmatullah, E. M. Yuniarno, and M. Attamimi, "Prediksi Akor Musik menggunakan Deep Learning berbasis Notasi Angka," *JURNAL TEKNIK ITS*, vol. Vol. 10, no. No. 1, 2021.

[2] S. Basini, G. N. Pardomuan, and M. S. Marlissa, "Pengenalan Dasar Alat Musik Gitar Untuk Siswa Kelas III Smp Negeri Borme Kabupaten Pegunungan Bintang Provinsi Papua," *Cantata Deo: Jurnal Musik dan Seni*, vol. 1, no. 1, pp. 12–24, Apr. 2023, doi: 10.69748/jmcd.v1i1.7.

[3] Y. Jadhav, A. Patel, R. H. Jhaveri, and R. Raut, "Transfer Learning for Audio Waveform to Guitar Chord Spectrograms Using the Convolution Neural Network," *Mobile Information Systems*, vol. 2022, 2022, doi: 10.1155/2022/8544765.

[4] A. Danika, J. Raharjo, and B. Hidayat, "Deteksi Suara Gitar Dengan Bahan Jenis Senar Berbeda Melalui Ciri Akustik Dengan Mel-Frequency Cepstral Coefficients (MFCC) Dan Support Vector Machine (SVM) Guitar String Detection Through Acoustic Characteristics Using Mel-Frequency Cepstral Coefficients (MFCC) And Support Vector Machine (SVM) Methods," Bandung City, Dec. 2022.

[5] P. Bonten and J. Kepler, "Guitar Tablature Detection using Recurrent Neural Networks," Aug. 2022. [Online]. Available: www.jku.atDVR0093696

[6] Y. Indrawaty, D. Rosmala, and A. M. Ramdhanial, "Aplikasi Pembelajaran Alat Musik Gitar Menggunakan Model Skenario Multimedia Interaktif Timeline Tree," *Jurnal Informatika*, vol. 4, no. 1, 2013.

[7] D. Régnier, N. Martin, and L. Bigo, "Identification of rhythm guitar sections in symbolic tablatures," Lille, Oct. 2021. [Online]. Available: https://hal.science/hal-03335822v1

[8] B. Setiadi and E. B. Setiawan, "Aplikasi Penerjemah Tablatur Gitar Menggunakan Teknologi Augmented Reality Pada Platform Android," *ULTIMA InfoSys*, vol. VII, no. 2, p. 86, 2016.

[9] J. Casco-Rodriguez, "Rock Guitar Tablature Generation via Natural Language Processing," Jan. 2023, [Online]. Available: http://arxiv.org/abs/2301.05295

[10] Y. Afifah Noor, M. Prasetya Aji, and B. Astuti, "Analisis frekuensi Gitar Menggunakan Smartphone. Prosiding Seminar Nasional Pascasarjana UNNES," Kota Semarang, 2020.

[11] R. R. A. Shaleha, "Do Re Mi: Psikologi, Musik, dan Budaya," *Buletin Psikologi*, vol. 27, no. 1, p. 43, Jun. 2019, doi: 10.22146/buletinpsikologi.37152.

[12] K. Sulya, A. Wasika, K. Gede, D. Putra, D. Purnami, and S. Putri, "Klasifikasi Kunci Gitar Menggunakan Spectral Analysis dan K-Nearest Neighbor," Kota Bali, Apr. 2020.

[13] K. Muludi and A. Frank SFB Loupatty, "Chord Identification Using Pitch Class Profile Method with Fast Fourier Transform Feature Extraction," May 2014. [Online]. Available: www.IJCSI.org

[14] R. Solehudin, G. Hermawan, and S. Kom, "Implementasi Metode MFCC (Mel Frequency Cepstral Coeeficient) dan Naive Bayesian untuk Klasifikasi Nada Dasar Gitar," Bandung City, Jan. 2019.

[15] A. Wiggins and Y. Kim, "Guitar Tablature Estimation With A Convolutional Neural Network," Delft, Nov. 2019. [Online]. Available: https://github.com/andywiggins/tab-cnn

[16] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *J Acoust Soc Am*, vol. 111, no. 4, pp. 1917–1930, Apr. 2002, doi: 10.1121/1.1458024.

[17] L. Sukhostat and Y. Imamverdiyev, "A Comparative Analysis of Pitch Detection Methods Under the Influence of Different Noise Conditions," Jul. 01, 2015, *Mosby Inc.* doi: 10.1016/j.jvoice.2014.09.016.

[18] J. Kim, "Automatic Pitch Detection and Shifting of Musical Tones in Real Time," 2014.

[19] D. Jouvet and Y. Laprie, "Performance Analysis of Several Pitch Detection Algorithms on Simulated and Real Noisy Speech Data," Aug. 2017. [Online]. Available: http://www.speech.kth.se/snack/

[20] O. Babacan, T. Drugman, N. d'Alessandro, N. Henrich, and T. Dutoit, "A Comparative Study of Pitch Extraction Algorithms on a Large Variety of Singing Sounds," Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.12609

[21] D. Rahadian Fudholi *et al.*, "The Application of LSTM in the AI-Based Enhancement of Classical Compositions," vol. 7, no. 1, pp. 107–117, 2024, doi: 10.20895/INISTA.V7I1.1628.

[22] N. Vatsya, A. Thipse, P. Dixit, R. Dafe, and K. Shejul, “Melody Generation using Deep Learning: Unleashing the Power of RNN and LSTM,” *International Journal of Innovative Science and Research Technology (IJISRT)*, pp. 1713–1720, May 2024, doi: 10.38124/ijisrt/ijisrt24apr2001.