DAFTAR PUSTAKA

1. Deng, L., Yu, D., Deng, L., & Yu, D. (2013). the essence of knowledge Deep Learning Methods and Applications Foundations and Trends ® in Signal Processing Deep Learning Methods and Applications Deep Learning: Methods and Applications. *Foundations and Trends R in Signal Processing*.
2. Maruf, M. R., Faruque, M. O., Mahmood, S., Nelima, N. N., Muhtasim, M. G., & Pervez, M. J. A. (2020). Effects of Noise on RASTA-PLP and MFCC based Bangla ASR Using CNN. *2020 IEEE Region 10 Symposium, TENSYMP 2020*. https://doi.org/10.1109/TENSYMP50017.2020.9231034
3. Automatic Speaker Recognition using MFCC and Artificial Neural Network. (2019). *International Journal of Innovative Technology and Exploring Engineering*. https://doi.org/10.35940/ijitee.a1010.1191s19
4. Kurzekar, P. K., Deshmukh, R. R., Waghmare, V. B., & Shrishrimal, P. P. (2014). A Comparative Study of Feature Extraction Techniques for Speech Recognition System. *International Journal of Innovative Research in Science, Engineering and Technology*. https://doi.org/10.15680/ijirset.2014.0312034
5. Anusuya, M. A., & Katti, S. K. (2011). Front end analysis of speech recognition: A review. In *International Journal of Speech Technology*. https://doi.org/10.1007/s10772-010-9088-7
6. Hermansky, H., & Morgan, N. (1994). RASTA Processing of Speech. *IEEE Transactions on Speech and Audio Processing*. https://doi.org/10.1109/89.326616
7. Labied, M., & Belangour, A. (2021). Automatic Speech Recognition Features Extraction Techniques: A Multi-criteria Comparison. *International Journal of Advanced Computer Science and Applications*. https://doi.org/10.14569/IJACSA.2021.0120821
8. Shao, M., Kuang, J., Wang, C., Zuo, W., & Wang, G. (2024). Multi-Dimensional Dynamic Pruning: Exploring Spatial and Channel Fuzzy Sparsity. *IEEE Transactions on Fuzzy Systems*. https://doi.org/10.1109/TFUZZ.2024.3363220
9. Juang, B. H., & Rabiner, L. R. (1991). Hidden markov models for speech recognition. *Technometrics*. https://doi.org/10.1080/00401706.1991.10484833
10. Sun, D. X., & Jelinek, F. (1999). Statistical Methods for Speech Recognition. *Journal of the American Statistical Association*. https://doi.org/10.2307/2670189
11. Leini, Z., & Xiaolei, S. (2021). Study on Speech Recognition Method of Artificial Intelligence Deep Learning. *Journal of Physics: Conference Series*. https://doi.org/10.1088/1742-6596/1754/1/012183

12. Özkural, E. (2018). The foundations of deep learning with a path towards general intelligence. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-319-97676-1_16
13. Lei, N., An, D., Guo, Y., Su, K., Liu, S., Luo, Z., Yau, S. T., & Gu, X. (2020). A Geometric Understanding of Deep Learning. *Engineering*. https://doi.org/10.1016/j.eng.2019.09.010
14. Dewi, I. N., Firdausillah, F., & Supriyanto, C. (2013). Sphinx-4 Indonesian isolated digit speech recognition. *Journal of Theoretical and Applied Information Technology*.
15. Broad, D. J. (1972). Basic directions in automatic speech recognition. *International Journal of Man-Machine Studies*. https://doi.org/10.1016/S0020-7373(72)80026-9
16. Sagheer, A., & Kotb, M. (2019). Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2018.09.082
17. Besacier, L., Barnard, E., Karpov, A., & Schultz, T. (2014). Automatic speech recognition for under-resourced languages: A survey. *Speech Communication*. https://doi.org/10.1016/j.specom.2013.07.008
18. Morris, A. C., Maier, V., & Green, P. (2004). From WER and RIL to MER and WIL: Improved evaluation measures for connected speech recognition. *8th International Conference on Spoken Language Processing, ICSLP 2004*. https://doi.org/10.21437/interspeech.2004-668
19. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.
20. Mastromichalakis, S. (2021). SigmoReLU: An improvement activation function by combining Sigmoid and ReLU. *Preprints*.
21. Tazi, E. B., & El Makhfi, N. (2017). An hybrid front-end for robust speaker identification under noisy conditions. *2017 Intelligent Systems Conference, IntelliSys 2017*. https://doi.org/10.1109/IntelliSys.2017.8324215
22. Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *Advances in Neural Information Processing Systems*.
23. Gupta, M., Camci, E., Keneta, V. R., Vaidyanathan, A., Kanodia, R., Foo, C.-S., Wu, M., & Lin, J. (2022, September 22). *Is complexity required for neural network pruning? A case study on global magnitude pruning*. 10.48550/arXiv.2209.14624

24. Anusuya, M. A., & Katti, S. K. (2011). Comparison of Different Speech Feature Extraction Techniques with and without Wavelet Transform to Kannada Speech Recognition. *International Journal of Computer Applications*. https://doi.org/10.5120/3092-4242
25. Ekštein, K., & Mouček, R. (2003). Time-domain structural analysis of speech. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/3-540-36456-0_53
26. Kinoshita, K., Ochiai, T., Delcroix, M., & Nakatani, T. (2020). Improving Noise Robust Automatic Speech Recognition with Single-Channel Time-Domain Enhancement Network. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. https://doi.org/10.1109/ICASSP40776.2020.9053266
27. Nian, Z., Du, J., Yeung, Y. T., & Wang, R. (2022). A TIME DOMAIN PROGRESSIVE LEARNING APPROACH WITH SNR CONSTRICTION FOR SINGLE-CHANNEL SPEECH ENHANCEMENT AND RECOGNITION. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. https://doi.org/10.1109/ICASSP43922.2022.9746609
28. Frankel, J., & King S. (2020). Speech Recognition In The Articulatory Domain: Investigating And Alternative To Acoustic HMMs. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. https://doi.org/10.1109/ICASSP40776.2020.9053266
29. D. S. Simangunsong, A. A. Zahra, and A. Hidayatno, "ESTIMASI PERBAIKAN NILAI SNR (SIGNAL TO NOISE RATIO) PADA PROSES DENOISING MENGGUNAKAN METODE WAVELET TERHADAP SUATU SINYAL BERDERAU," *Transient: Jurnal Ilmiah Teknik Elektro*, vol. 2, no. 4, pp. 895-899, Dec. 2013. https://doi.org/10.14710/transient.v2i4.895-899

Lampiran

Lampiran 1. Hasil Cek Turnitin

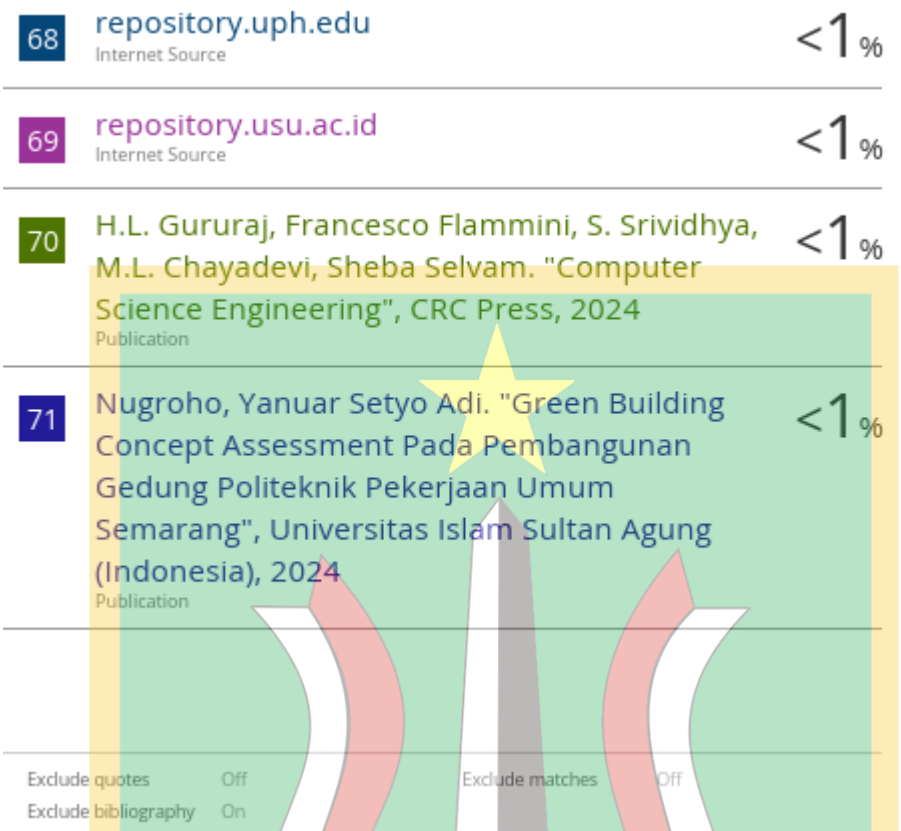
ORIGINALITY REPORT

10%	9%	6%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Southville International School and Colleges Student Paper	2%
2	Submitted to Universitas Indonesia Student Paper	1%
3	"Effects of Noise on RASTA-PLP and MFCC based Bangla ASR Using CNN", 2020 IEEE Region 10 Symposium (TENSYMP), 2020 Publication	<1%
4	www.semanticscholar.org Internet Source	<1%
5	repository.nurulfikri.ac.id Internet Source	<1%
6	www.ijitee.org Internet Source	<1%
7	123dok.com Internet Source	<1%
8	lib.ui.ac.id Internet Source	<1%
9	Submitted to Universitas Islam Indonesia Student Paper	<1%
10	repositori.uma.ac.id Internet Source	<1%

11	Alvinus Cardova, Arief Hermawan. "Implementasi Metode LSTM Untuk Mengklasifikasi Berita Palsu Pada PolitiFact", JURNAL FASILKOM, 2023 Publication	<1%
12	etd.umy.ac.id Internet Source	<1%
13	etheses.uin-malang.ac.id Internet Source	<1%
14	Rina Aprilianti, Ika Purnamasari, Surya Prangga. "Peramalan Jumlah Wisatawan Mancanegara di Provinsi Kalimantan Timur Menggunakan Fuzzy Backpropagation Neural Network", Jurnal Statistika dan Komputasi, 2023 Publication	<1%
15	www.scribd.com Internet Source	<1%
16	Submitted to University of North Carolina, Greensboro Student Paper	<1%
17	repository.usbypkp.ac.id Internet Source	<1%
18	repository.unsri.ac.id Internet Source	<1%
19	Submitted to Sriwijaya University Student Paper	<1%
20	ejournal-pasca.undiksha.ac.id Internet Source	<1%
21	eprints.kwikkiangie.ac.id Internet Source	<1%



Untuk melihat keluruhan hasil turnitin dapat dilihat melalui klik link ini.

Lampiran 2. Coding Project

```
# Configuration
class ModelConfigs(BaseModelConfigs):
    def __init__(self):
        super().__init__()
        self.model_path = os.path.join("/kaggle/working/", datetime.strftime(datetime.now(), "%Y%m%d%H%M"))
        os.makedirs(self.model_path, exist_ok=True)
        self.frame_length = 384
        self.frame_step = 512
        self.fft_length = 384

        self.vocab = "abcdefghijklmnopqrstuvwxyz'?!,. "
        self.input_shape = None
        self.max_text_length = None
        self.max_spectrogram_length = None

        self.batch_size = 32
        self.learning_rate = tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate=0.0005,
                                                                            decay_steps=10000,
                                                                            decay_rate=0.9)
        self.train_epochs = 100
        self.train_workers = 4
```

```python
# Hybrid features Extraction
class HybridFeatureExtractor:
    def __init__(self, frame_length, frame_step, fft_length, n_mfcc=13, sample_rate=22000):
        self.frame_length = frame_length
        self.frame_step = frame_step
        self.fft_length = fft_length
        self.n_mfcc = n_mfcc
        self.sample_rate = sample_rate

    def __call__(self, file_path, annotation):
        mfcc = extract_mfcc(file_path, self.frame_length, self.frame_step, self.fft_length, self.n_mfcc)
        rasta_plp = extract_rasta_plp(file_path, self.frame_length, self.frame_step, self.fft_length)

        # Sesuaikan jumlah frame
        mfcc_features, rasta_plp_features = align_features(mfcc, rasta_plp)
        # Gabungkan fitur MFCC dan RASTA-PLP
        hybrid_features = np.concatenate((mfcc_features, rasta_plp_features), axis=1)
        hybrid_features = (hybrid_features - np.mean(hybrid_features, axis=0)) / (np.std(hybrid_features, axis=0) + 1e-8)

        return hybrid_features, annotation
```

```python
# Function to apply padding safely
def safe_pad_spectrogram(spectrogram, max_length, padding_value=0):
    if spectrogram.shape[0] >= max_length:
        return spectrogram  # No need to pad if it's already larger or equal
    # Calculate padding value
    padding_shape = ((0, max_length - spectrogram.shape[0]), (0, 0))
    return np.pad(spectrogram, padding_shape, mode='constant', constant_values=padding_value)

# Modifikasi di tempat spectrogram diproses
class SpectrogramPadding:
    def __init__(self, max_spectrogram_length, padding_value=0):
        self.max_spectrogram_length = max_spectrogram_length
        self.padding_value = padding_value

    def __call__(self, spectrogram, annotation):
        # Gunakan safe_pad_spectrogram untuk mencegah error nilai negatif
        padded_spectrogram = safe_pad_spectrogram(spectrogram, self.max_spectrogram_length, self.padding_value)
        return padded_spectrogram, annotation
```

```python
# Function to save audio file correctly
def save_audio_file(audio_array, sample_rate, file_path):
    try:
        sf.write(file_path, audio_array, sample_rate)
    except Exception as e:
        print(f"Error saving audio file {file_path}: {e}")

# Function for normalize audio
def normalize_audio(audio_array):
    audio_array = audio_array / np.max(np.abs(audio_array))
    return audio_array
```

```python
# Modify the extract_mfcc function to include visualization option
def extract_mfcc(file_path, frame_length, frame_step, fft_length, n_mfcc=13, sr=22000, visualize=False):
    # Load and normalize audio
    audio, sr = librosa.load(file_path, sr=sr)
    audio = audio / (np.max(np.abs(audio)) + 1e-8)

    # Pre-emphasis filter
    pre_emphasis = 0.97
    audio = np.append(audio[0], audio[1:] - pre_emphasis * audio[:-1])

    # Extract MFCCs
    mfcc = librosa.feature.mfcc(
        y=audio,
        sr=sr,
        n_mfcc=n_mfcc,
        n_fft=fft_length,
        hop_length=frame_step,
        window='hamming',
        center=True,
        power=2.0
    )

    # Normalize MFCC features
    mfccs = (mfcc - np.mean(mfcc, axis=1, keepdims=True)) / (np.std(mfcc, axis=1, keepdims=True) + 1e-8)

    return mfccs.T
```

```python
# Function to adjust number of frame
def align_features(mfcc, rasta_plp):
    min_frames = min(mfcc.shape[0], rasta_plp.shape[0])
    mfcc_aligned = mfcc[:min_frames, :]
    rasta_plp_aligned = rasta_plp[:min_frames, :]
    return mfcc_aligned, rasta_plp_aligned
```

```python
# Function for RASTA filter
def rasta_filter(signal):
    # RASTA filter coefficient
    pole = 0.94
    zero = np.array([2.0, -2.0]) / 3.0
    pole_z = np.array([1.0, -0.94])

    # Apply filter
    filtered = scipy.signal.lfilter(zero, pole_z, signal)
    return filtered
```

```python
def extract_rasta_plp(file_path, frame_length, frame_step, fft_length, n_mels=26, n_plp=13, sr=22000):
    # 1. Load and normalize audio
    signal, sr = librosa.load(file_path, sr=sr)
    signal = signal / (np.max(np.abs(signal)) + 1e-8)

    # Ensure minimum signal length
    if len(signal) < frame_length:
        signal = np.pad(signal, (0, frame_length - len(signal)), mode='reflect')

    # 2. Pre-emphasis
    pre_emphasis = librosa.effects.preemphasis(signal, coef=0.97)

    # 3. Short-time Fourier transform
    windows = scipy.signal.windows.hann(frame_length)
    frames = librosa.util.frame(pre_emphasis, frame_length=frame_length, hop_length=frame_step)

    # 4. Power spectrum
    power_spec = np.abs(np.fft.rfft(frames.T * windows, n=fft_length, axis=1)) ** 2 + 1e-10

    # 5. Mel filterbank energies
    mel_basis = librosa.filters.mel(sr=sr, n_fft=fft_length, n_mels=n_mels)
    mel_spec = np.dot(power_spec, mel_basis.T)

    # 6. Log compression
    log_mel_spec = np.log(mel_spec + 1e-6)
```

```python
# 7. Apply RASTA filtering to each frequency band
rasta_filtered = np.array([rasta_filter(band) for band in log_mel_spec.T]).T

# Alternative normalization techniques
rasta_plp_features = scipy.fftpack.dct(np.exp(rasta_filtered), axis=1, norm='ortho')[:, :n_plp]

# Three alternative normalization methods
normalized_features = [
    # Method 1: Standard scaling
    (rasta_plp_features - np.mean(rasta_plp_features, axis=0)) / (np.std(rasta_plp_features, axis=0) + 1e-8),

    # Method 2: Min-Max scaling
    (rasta_plp_features - np.min(rasta_plp_features, axis=0)) /
    (np.max(rasta_plp_features, axis=0) - np.min(rasta_plp_features, axis=0) + 1e-8),

    # Method 3: Robust scaling (using median and IQR)
    (rasta_plp_feat (variable) rasta_plp_features: Any axis=0)) /
    (np.percentile(rasta_plp_features, 75, axis=0) - np.percentile(rasta_plp_features, 25, axis=0) + 1e-8)
]

return normalized_features[0]
```

Untuk codingan project lengkap dapat diakses melalui klik link ini.