Transformer-CNN Automatic Hyperparameter Tuning for Speech Emotion Recognition

Agustinus Bimo Gumelar
Dept. of Electrical Engineering,
The Faculty of Intelligent Electrical and Informatics Technology (F-ELECTICS),
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
bimogumelar@ieee.org

Eko Mulyanto Yuniarno
Dept. of Electrical Engineering,
Dept. of Computer Engineering,
The Faculty of Intelligent Electrical and Informatics Technology (F-ELECTICS),
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
ekomulyanto@ee.its.ac.id

Derry Pramono Adi
Dept. of Electrical Engineering,
The Faculty of Intelligent Electrical and Informatics Technology (F-ELECTICS),
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
derryalbertus@ieee.org

Rudi Setiawan
Dept. of Computer Science,
UIN Surabaya,
Surabaya, Indonesia
rudistwn@gmail.com

Indar Sugiarto
Dept. of Electrical Engineering,
Petra Christian University,
Surabaya, Indonesia
indi@petra.ac.id

Mauridhi Hery Purnomo
Dept. of Electrical Engineering,
Dept. of Computer Engineering,
The Faculty of Intelligent Electrical and Informatics Technology (F-ELECTICS),
Institut Teknologi Sepuluh Nopember and University Center of Excellence on Artificial Intelligence for Healthcare and Society (UCE AIHeS)
Surabaya, Indonesia
hery@ee.its.ac.id

2022 IEEE International Conference on Imaging Systems and Techniques (IST) | 978-1-6654-8102-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IST55454.2022.9827732

***Abstract*—Given the high number of hyperparameters in deep learning models, there is a need to tune automatically deep learning models in specific research cases. Deep learning models require hyperparameters because they substantially influence the model's behavior. As a result, optimizing any given model with a hyperparameter optimization technique will improve model efficiency significantly. This paper discusses the hyperparameter-optimized Speech Emotion Recognition (SER) research case using Transformer-CNN deep learning model. Each speech samples are transformed into spectrogram data using the RAVDESS dataset, which contains 1,536 speech samples (192 samples per eight emotion classes). We use the Gaussian Noise augmentation technique to reduce the overfitting problem in training data. After augmentation, the RAVDESS dataset yields a total of 2,400 emotional speech samples (300 samples per eight emotion classes). For SER model, we combine the Transformer and CNN for temporal and spatial speech feature processing. However, our Transformer-CNN must be thoroughly tested, as different hyperparameter settings result in varying accuracy performance. We experiment with Naïve Bayes to optimize many hyperparameters of Transformer-CNN (it could be categorical or numerical), such as learning rate, dropouts, activation function, weight initialization, epoch, even the best split data scale of training and testing. Consequently, our automatically tuned Transformer-CNN achieves 97.3% of accuracy.**

Keywords—Speech Emotion Recognition, Automatic Hyperparameter Tuning, Transformer-CNN, Naïve Bayes Optimization

I. INTRODUCTION

In SER, the most emotion-related acoustic features are "extracted" from speech data in general. Acoustic features in human speech contain information that characterizes emotion. Due to its vast ability for complex computation, Speech Emotion Recognition (SER) is implemented using varying Deep Learning models.

Moving forward with current improvements, end-to-end SER has gained potential performance with the introduced self-attention network: Transformer, which has been dubbed as a successor for both LSTM and RNN. One outstanding element of the Transformer is that it provides direct access per step via self-attention, which almost eliminates information loss in message transit. Furthermore, future and past sentences can be examined almost concurrently. This behaviour also exists in Bidirectional RNN and Bidirectional LSTM, but both need two times computation [1], whereas Transformer only needs one time computation. Furthermore, Transformer happens to be non-recurrent, which makes training runtime much faster.

Even though the success of Transformer networks in classifying different problems is impressive, network training is strongly dependent on a collection of hyperparameters that control many elements of the specific model. There is no universal hyperparameter configuration for all tasks [2]. The hyperparameters need to be optimized span from the number of hidden neurons, batch size, the constant of dropout and learning rate, even weight-related hyperparameters.

Emotional variability has been demonstrated in some previous studies to significantly affect SER performance [3]. Deep Learning model optimization for detecting the presence of emotion could assist increase the robustness of SER systems. The purpose of this paper is to investigate the impact of hyperparameters in the Transformer network in order to determine which hyperparameter configuration is most suited for SER systems. Furthermore, a total of four encoder and decoder networks are integrated with a CNN in the Transformer network.

This paper is organized as follows: Section 1 discusses the significance of employing hyperparameter optimization for SER by Transformer. Section 2 explains the human emotion concept from speech features. Section 3 extensively explains the need and architecture of Transformer-CNN for this work, as well as the Naïve Bayes Optimization framework for hyperparameter optimization. Results of the Sensitivity, Specificity, Accuracy and Confusion Matrix are presented in Section 4, along with a fine-tuned hyperparameter value and comparisons to previous studies. Section 5 concludes our Transformer-CNN optimization.

II. Emotional Speech

A. Human Emotion Concept

Classifying emotions and defining the boundaries between them, affect, and mood have been the subject of numerous initiatives in an effort to make it easier for computer programs to recognize and evaluate them. Automated emotion recognition and rating complicates the analysis of so many different feelings. When Feidakis et al. classify emotions according to basic models [4], they find that there are 66 different types of feelings: ten basic emotions (anger, anticipation, distrust, fear, happiness, joy, love, sadness, surprise, trust) and 56 secondary emotions, which can be further divided into two categories.

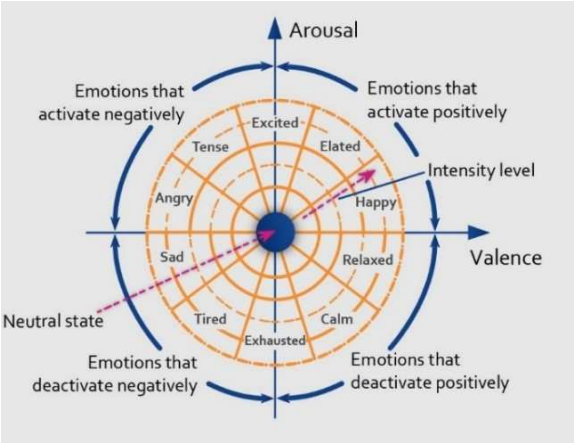

Fig. 1. Emotion Categorization using Arousal-Valence Paradigm

To address this difficulty, most studies on emotion evaluation rely on alternative classifications [5], which contain aspects of emotions and analyze only basic emotions that are more simply described. In most studies, many researchers employ versions of Russel's circumplex model of emotions, as shown in Fig. 1, which provides a two-dimensional representation of basic emotions in terms of valence (activation: negative/positive) and arousal (high or low) [6]. After a network model has been constructed, the next crucial issue is acquiring labelled data for training and testing that conforms to the selected emotion representation model [7]. The relatively high level of subjectivity and uncertainty in the target labelling is, without a doubt, a complex subject to tackle. Luckily, public datasets like RAVDESS labeled its dataset for reproducible research [8].

B. Prosodic Features

In vocal tract-based categories, MFCC is a common cepstral coefficient associated with prosodic features. Formant features become other cepstral coefficients because these properties typically indicate energy distribution in the frequency range of speech [7]. A depiction of the short-term power spectrum, MFCCs is averaged over surrounding frequency bands, and the frequencies are scaled to mimic the psychoacoustic characteristics of the human ear. When it comes to determine pitch distances, the Mel-scale, a scale of pitches deemed by listeners to be equal in distance from one another, might be utilized. In this experiment, we use acoustic features such as the Mel-Frequency Cepstral Coefficient (MFCC), Jitter, Shimmer, and Fundamental Frequency (F0). Furthermore, these features are extracted with Librosa 0.7.1 [9].

The MFCC approaches employs a filter bank approach. A Discrete Cosine Transform is utilized in order to obtain the required quantity of cepstral coefficients once the spectrum of each Hamming-windowed signal has been segmented into Mel-spaced triangular frequency bins. F0, on the other hand, can be detected simply by repeating n samples. Eq. (1) depicts the MFCC calculation formula, where f is the frequency being converted to the Mel scale.

$$Mel(f) = 1125\ln(1 + \frac{f}{700}) \quad (1)$$

$$Jitter = \frac{|T_i - T_{i+1}|}{\frac{1}{N}\sum_{i=1}^{N} T_i} \quad (2)$$

$$Shimmer = \frac{|A_i - A_{i+1}|}{\frac{1}{N}\sum_{i=1}^{N} A_i} \quad (3)$$

We employ the Jitter and Shimmer features to represent voice quality characteristics in an emotional speech. Jitter is a measure of F0 fluctuations from one period to the next. By computing F0 variations from one cycle to the next, Jitter may be measured and characterized in terms of absolute and Relative Average Perturbation (RAP) values [10]. Eq. (2) depicts the Jitter calculation formula, where N is the total number of voiced frames from F0 and T_i is the time instant at the F0 period length.

Shimmer is a calculation of amplitude fluctuations' period-to-period variability [10]. F0 does not provide peak-picking in time-domain scenarios such as speech recognition [11]. Using Shimmer, peak-picking is possible without being restricted by the F0 constraint. Furthermore, Eq. (3) depicts the Shimmer calculation formula, where A_i is the peak-to-peak amplitude at the F0 period. These traits are also utilized in speech and expression recognizers to depict emotions. Jitter and Shimmer are extracted systematically, needing information on how the F0 is achieved [12]. Jitter and Shimmer characteristics, in essence, can capture emotional speech evidence such as jitters, stuttering, and disturbance.

III. Models for Speech Emotion Recognition

A. Transformer Model

There is an attempt to multiply self-attention network (referred in Eq. (4)): the multi-head self-attention. Transformer architecture encodes an input sequence's context vectors as a set of key-value *(K, V)* pairs with the same dimension as the input sequence length. Both the keys (inputs) and values (inputs' hidden states) comprise the encoder's hidden states. The output predicts at the previous

timestep by the decoder is computed into a "query", and the next term in the decoder's output sequence is a mapping from the key-value pairs plus query: *(Q,K,V)*.

$$Att(Q,K,V) = softmax(\frac{Q \times K^T}{\sqrt{n}})V \quad (4)$$

Each output term of the decoder is a weighted sum of all values from the *(K,V)* encoded representation of an input - so, like a regular attention mechanism which decodes a weighted sum of hidden states, but self-attention assigns the (alignment) weights to each value (hidden state) as a sequence-length-scaled dot-product of the query with all the keys. That is, the weighted sum of all the inputs' hidden states is computed $w \times r \times t$ from the previous (last) term in the output sequence and the entire input sequence. This is where the global attention ability of the Transformer originates.

According to Vaswani et al., the scaled dot product self-attention (Q, K, V) is computed over multiple "representation subspaces". Therefore, each query, key and value has its own weight matrix associated [13]. In result, multi-headed (multi-layer) self-attention can compute a term in the output sequence weighted differently according to a region (subspace) of the input sequence.

$$MH(Q,K,V) = [head_1; \cdots; head_h] \times W^O \quad (5)$$

$$head_i = Att(Q \times W_i^Q, K \times W_i^K, V \times W_i^V) \quad (6)$$

Each attention head in multi-headed self-attention still computes a scaled dot product over the entire (K, V) encoded input, just weighted differently $w \times r \times t$ the input values. Complete multi-head self-attention equation is shown in Eq. (5). Eq. (5) needs different representation of attention, which is shown in Eq. (6), with W_i^Q, W_i^K, W_i^V, and W_i^O are parameters of matrices to be learned.

The output of all the attention heads is concatenated and multiplied with a weight matrix which puts the dimension of the encoded state back to that of a single attention head; then a single feedforward layer can operate on the encoded latent space regardless of the number of attention heads, and a softmax prediction is computed from a weighted sum of all layers in the multi-headed attention architecture.

B. CNN Model

In the recent developments for image processing, CNNs with 2D convolutional layers become robust. In this experiment, we need to transform speech data into spectrogram data to get the most out of CNNs capabilities. Input feature maps for two-dimensional spectrograms image convolution layers might be in the (N, C, H, W) or batch size, channel, height, and width formats, respectively. There are 4,320 plots in the form of MFCC plots: 1,440 native and 2,880 Gaussian Noise augmented. Each MFCC plot is 40 x 282 and represents one Mel coefficient pitch range with 282-time steps. Before training, the MFCC input feature tensor will have a shape of (4,320, 1, 40, 282).

In both CNN blocks, we are employing 3×3 kernels in all three layers. One input channel in the first layer creates a 133 filter, with 16 output channels requiring 16 identical 133 filters with nine weights each. There are 16 input channels and 32 output channels for $16 \times 3 \times 3$ filters that each have 144 weights. A total of 22 maxpooled output from the first layer is fed into the second layer, resulting in a $32 \times 35 \times 35$ output feature map after 4×4 stride 4 maxpooling, which applies 32 different weights to the $16 \times 3 \times 3$ filters. 64 distinct filters with 288 weights each make up the final layer's $32 \times 3 \times 3$ filtering structure. After 4×4 stride four maxpooling, the final layer gives an output feature map of 6,418. To achieve the most expressive hierarchical feature representation at the lowest computational cost, we intend to expand filter depth/complexity simultaneously and reduce feature map volume.

C. Naïve Bayes Optimization

One issue with the vanilla model is the substantial number of defining hyperparameters, which can range from a hundred to a thousand. Manually adjusting these hyperparameters is also costly in terms of experiment trials. According to Yang and Nam, there is a 20,480,000,000 hyperparameter-adjusting possibilities for Transformer to achieve maximum accuracy [14]. Many prior studies show that fiddling with the Deep Learning model's hyperparameters is effective [15]. We consider the most basic hyperparameter optimization learner, the well-known Naïve Bayes (hereinafter will be referred as NBO or Naïve Bayes Optimization).

$$k(\theta, \theta'): \Theta^2 \to R \quad (7)$$

$$\mu_t(\theta) = k(\theta)^T [K + \tau^2 I]^{-1} y \quad (8)$$

$$\sigma_t^2(\theta) = k(\theta, \theta) - k(\theta)^T [K + \tau^2 I]^{-1} k(\theta) \quad (9)$$

$$\theta_{t+1} = \arg\max_{\theta \in \Theta} U_t(\theta; S_t) \quad (10)$$

In this study, NBO is generally made as black-box function and to solve global optimization problems. The search space Θ can be a compact subset of R^d, but the NBO can be applied also to search spaces involving categorical or numerical input. The NBO framework consists of two important components: (1) a probabilistic surrogate model consists of a prior distribution that models the unknown objective function, and (2) an optimized acquisition function for determining next sample. The surrogate model generates a posterior probability distribution which describes potential for H(θ) at a candidate configuration θ. The basic idea is every observation of H at a new point θ, NBO updates its posterior distribution. We use the Gaussian Process (GProc) as surrogate model. GProc is completely specified by a mean function μ(θ):Θ → R and a definite positive covariance function, as given in Eq. (7).

NBO begins with an initial set of k hyperparameter configurations $\{\theta_i\}_{i=1}^k$ and their associated function values $\{y_i\}_{i=1}^k$, with $y_i = H(\theta_i)$. At iteration of $t \in \{k+1, \cdots, N\}$, NBO retains past evaluated configurations, and then update the GProc model to obtain posterior distribution on the current training set of $S_t = \{(\theta, y_i)\}_{i=1}^t$. For any configurations $\theta \in \Theta$, the posterior mean $\mu_t(\theta)$ and the posterior variance $\sigma_t^2(\theta)$ of the GProc (conditioned on S_k) is denoted in Eq. (8) and Eq. (9). In Eq. (8) and Eq. (9), K is the $t \times t$ matrix with entries of $K_{i,j} = k(\theta_i, \theta_j), k(\theta)$ is the $t \times 1$ vector of covariance terms between θ and $\{\theta_i\}_{i=1}^t$, y is the $t \times 1$ vector, with i^{th} entry is y_i and τ^2 is the noise variance. By solving an auxiliary

optimization problem in Eq. (10), the next candidate points to assess are chosen. In Eq. (10), U_t is the acquisition function to be maximized.

IV. The Experiment Design

A. Speech Emotion Dataset and Augmentation

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) is a validated public database of American English-spoken emotional speech and song. RAVDESS has eight emotion classes of Surprised, Neutral, Calm, Happy, Sad, Angry, Fearful, and Disgust, with 7.356 files. RAVDESS contains voice and song files in two different audio-only properties, audio-video properties, and one video-only properties. RAVDESS employs 24 professional actors, who has performed two statements, "Kids are talking by the door" and "Dogs are sitting by the door".

The performance of a deep learning model is dependent on a huge dataset. However, we may enhance the model's performance by adding additional data using Gaussian Noise. First, the original data set is perturbed to generate extra datasets, and then the additional datasets are included into a larger dataset as a regularization approach. The built-in data augmentation utilities in deep learning frameworks can be inefficient or lacking in some needed features. While there are several data augmentation strategies for speech classification task available in the literature.

$$x(t) = \sigma\big(n(t)\big) + x(t) \tag{11}$$

In this work, augmentation is done by incorporating Gaussian noise into the speech audio data samples to smooth out the input space. Given that the mean and variance of Gaussian noise $n(t)$ are both zero and one, it is simple to create using a pseudo-random number generator. The amplitude of the noise is a significant aspect in noise. When the noise amplitude (σ) is too tiny, it is difficult to perturb the classifier. When σ value is too big, it is difficult for classifier to learn. We establish a reasonable range for it $x(t)$ is the raw signal, and the range is $[0.001, 0.015]$ and it follows the uniform distribution. The freshly generates sample following the addition of noise can be stated in Eq. (11).

B. Transformer-CNN Architecture

Each three-layer of deep 2D-CNN is extremely similar to the classic LeNet. We use small-stacked filters of three 3×3 kernels on top of each other, the second layer has a 5×5 view of the original input volume, and the third layer a 7×7 view. Non-linearities between each smaller layer, on the other hand, expresses more complicated feature representations. Meanwhile, we try to keep Transformer as vanilla as possible, save that we utilize four encoders rather than the original six encoders. Fig. 2 shown a complete architecture overview of our Transformer-CNN.

NBO's best-fit model will be applied automatically in Transformer-CNN. Furthermore, any result of hyperparameters configuration is limited to 100 iterations (no early stopping) to avoid hardware overburdening. An independent test is performed by optimal Transformer-CNN model after the hyperparameter optimization process by NBO. Sensitivity, Specificity, Accuracy, and the Receiver Operating Characteristic (ROC) curve have all been used to validate the performance of our Transformer-CNN model.

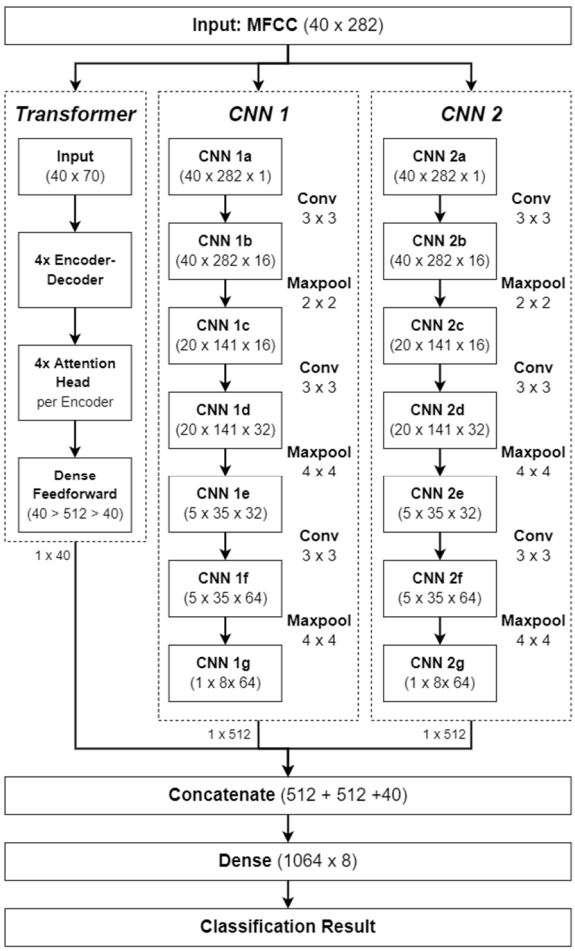

Fig. 2. Architecture Overview of our proposed Transformer-CNN

V. Result and Discussion

Using our tuned Transformer-CNN, we are able to reach 10 hours and 40 minutes of tuning runtime, while learning runtime reached 3 hours, 1 minute, and 26 seconds. Furthermore, Table I shown an optimal configuration of hyperparameter from our tuned Transformer-CNN. According to NBO, 200 epochs is best number for this experiment. Although CNN and Transformer took data directly from 2D spectrogram (40×282), the dropout hyperparameter thought to be best if set respectively.

TABLE I. Optimal Hyperparameters Configuration for Transformer-CNN

Hyperparameters	***Value***
Activation Function	relu
CNN Filter	77
Data Test Size	0.1
Data Train Size	0.9
Dropout (CNN)	0.2159447275666912
Dropout (Transformer)	0.11516595022108196
Epoch	200
Learning Rate	0.09583585446450836
Loss Function	categorical_cross_entropy
Optimizer Function	adadelta
Weight Initialization	xavier_uniform

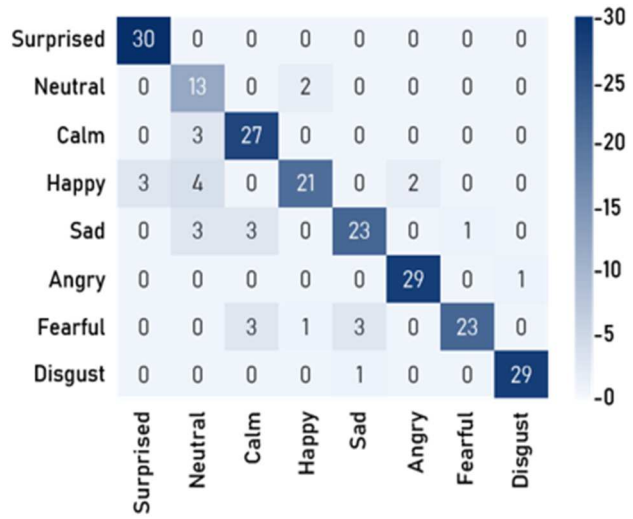

Fig. 3. Validation of Confusion Matrix using our Transformer-CNN

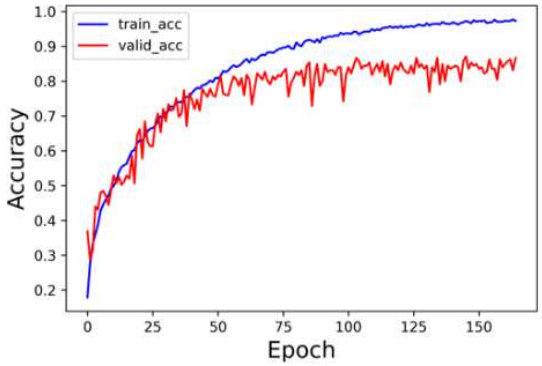

Fig. 4. Accuracy Validation level

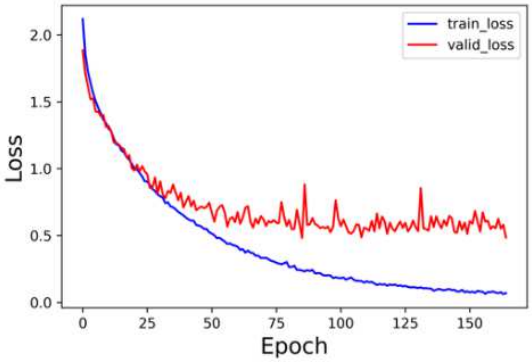

Fig. 5. Loss Validation level

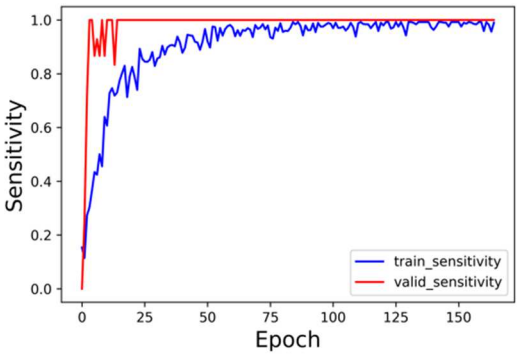

Fig. 6. Sensitivity Validation chart

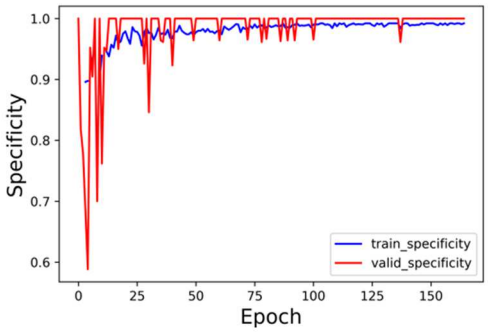

Fig. 7. Specificity Validation chart

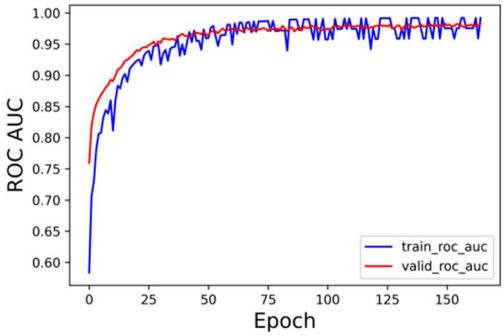

Fig. 8. ROC-AUC Validation chart

Fig. 3 shown a Confusion Matrix of different emotional speech classes from RAVDESS. According to Fig. 3, the "Surprised" class achieves better result by predicting 30 out of 30 test speech samples correctly. Practically, the "Surprised" class should reach accuracy of 100%, but because Accuracy is calculated from whole data and whole classes, our Transformer-CNN reached 97.3%.

Without hyperparameter optimization, our Transformer-CNN is able to achieve only 86% of accuracy. The Accuracy chart is shown in Fig. 4, alongside with other validations of Loss validation in Fig. 5, the Sensitivity validation in Fig 6, the Specificity validation in Fig. 7, and ROC-AUC results of our automatically tuned Transformer-CNN is shown in Fig. 8.

According to Fig. 3, the "Surprised" class is indeed classified easier using our tuned Transformer-CNN. Second best classification is done for "Disgust" emotion with 29 correctly predicted instances out of 30. However, "Neutral" emotion class have worse classification result, with only 13 correctly predicted instances out of 30. This finding may occur because "Neutral" emotion class have subjectively ambiguous emotion class rather than other classes. This result shown RAVDESS really has excellent data (except for "Neutral" class), and our Gaussian Noise performs excellently in adding needed noise. Referring to Fig. 6, best Sensitivity reached 0.99, best Specificity from Fig. 7 reached 0.992, best ROC reached 0.99 as shown in Fig. 8.

The Sensitivity and Specificity results are steady with the Accuracy result of 0.973 (or 97.3%). However, all other classes also performes excellently, as the number of correctly predicted instances in Fig. 3 differs not too much from each emotion classes. It is clear that adding CNN does not give any trouble in learning at all, as long as hyperparameter optimization is employed, whatever model may be used. Furthermore, in Table II, our proposed work of optimized

Transformer-CNN by NBO reaches the best accuracy from previous works.

TABLE II. RESULT COMPARISON WITH PREVIOUS WORKS

Author	*Model*	*Best Accuracy*
[16] Tarantino et al. (2019)	Transformer	64.7%
[17] Wang et al. (2021)	Transformer	76.8%
[18] Han et al. (2021)	Transformer-CNN	80.8%
[19] Zhang et al. (2021)	Transformer	91%
[20] Gong et al. (2021)	Transformer-CNN	97%
Our Proposed Work (2022)	Transformer-CNN	86%
	NBO-optimized Transformer-CNN	97.3%

VI. CONCLUSION

In this study, our tuned Transformer-CNN is able to perform SER task excellently. Accuracy reached 97.3% for eight emotion classes. Sensitivity, Specificity, and ROC reached 99%, 99.2%, and 99%, respectively. Steady result between Sensitivity and Specificity means the result of our Transformer-CNN is valid, and the use of Gaussian Noise to reduce data overfitting has worked. Meanwhile, ROC result above 50% means the model has no trouble in classifying instances between classes. This study has proved that the usage of hyperparameter optimization using NBO for Transformer-CNN in SER task is highly beneficial. After best fit hyperparameter configuration is applied on our Transformer-CNN, our model does not have to struggle in learning, while still retaining quick computation time. The optimization runtime is slower than learning runtime, which is expected, because in tuning process, all probability of best fit hyperparameter configuration is computed thoroughly.

ACKNOWLEDGMENT

We thank the Ministry of Research, Technology, and Higher Education of the Republic of Indonesia for providing research funding through the Doctoral Dissertation Research Grant scheme with contract number 848/PKS/ITS/2021.

REFERENCES

[1] A. B. Gumelar, E. M. Yuniarno, D. P. Adi, A. G. Sooai, I. Sugiarto, and M. H. Purnomo, "BiLSTM-CNN Hyperparameter Optimization for Speech Emotion and Stress Recognition," in *2021 International Electronics Symposium (IES)*, 2021, pp. 156–161.

[2] B. Zhang *et al.*, "On the Importance of Hyperparameter Optimization for Model-based Reinforcement Learning," in *International Conference on Artificial Intelligence and Statistics*, 2021, pp. 4015–4023.

[3] E. Lieskovská, M. Jakubec, R. Jarina, and M. Chmul\'\ik, "A Review on Speech Emotion Recognition using Deep Learning and Attention Mechanism," *Electronics*, vol. 10, no. 10, p. 1163, 2021.

[4] M. Feidakis, T. Daradoumis, and S. Caballé, "Endowing E-Learning Systems with Emotion Awareness," in *2011 Third International Conference on Intelligent Networking and Collaborative Systems*, 2011, pp. 68–75.

[5] M. Feidakis, T. Daradoumis, and S. Caballé, "Emotion Measurement in Intelligent Tutoring Systems: What, When and How to Measure," in *2011 Third International Conference on Intelligent Networking and Collaborative Systems*, 2011, pp. 807–812.

[6] J. A. Russell, "A Circumplex Model of Affect," *J. Pers. Soc. Psychol.*, vol. 39, no. 6, pp. 1161–1178, 1980, doi: 10.1037/h0077714.

[7] N. Colnerič and J. Demšar, "Emotion Recognition on Twitter: Comparative Study and Training a Unison Model," *IEEE Trans. Affect. Comput.*, vol. 11, no. 3, pp. 433–446, 2018.

[8] S. R. Livingstone and F. A. Russo, *The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A Dynamic, Multimodal Set of Facial and Vocal Expressions in North American English*, vol. 13, no. 5. 2018. doi: 10.1371/journal.pone.0196391.

[9] S. Moon, S. Kim, and Y.-H. Choi, "MIST-Tacotron: End-to-End Emotional Speech Synthesis Using Mel-Spectrogram Image Style Transfer," *IEEE Access*, vol. 10, pp. 25455–25463, 2022.

[10] L. Chen, X. Mao, Y. Xue, and L. L. Cheng, "Speech emotion recognition: Features and classification models," *Digit. Signal Process.*, vol. 22, no. 6, pp. 1154–1160, 2012.

[11] C.-N. Anagnostopoulos, T. Iliou, and I. Giannoukos, "Features and classifiers for emotion recognition from speech: a survey from 2000 to 2011," *Artif. Intell. Rev.*, vol. 43, no. 2, pp. 155–177, 2015.

[12] T. Bäckström, "Fundamental Frequency Estimation," *Speech Analysis*, 2019. https://wiki.aalto.fi/display/ITSP/Fundamental+frequency+estimation (accessed Apr. 01, 2022).

[13] A. Vaswani *et al.*, "Attention is All You Need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[14] H. Yang and H. Nam, "Hyperparameter Experiments on End-to-End Automatic Speech Recognition," *Phonetics Speech Sci.*, vol. 13, no. 1, pp. 45–51, Mar. 2021, doi: 10.13064/KSSS.2021.13.1.045.

[15] L. Yang and A. Shami, "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[16] L. Tarantino, P. N. Garner, A. Lazaridis, and others, "Self-Attention for Speech Emotion Recognition," in *Interspeech*, 2019, pp. 2578–2582.

[17] Y. Wang, G. Shen, Y. Xu, J. Li, and Z. Zhao, "Learning Mutual Correlation in Multimodal Transformer for Speech Emotion Recognition," *Proc. Interspeech 2021*, pp. 4518–4522, 2021.

[18] S. Han, F. Leng, and Z. Jin, "Speech Emotion Recognition with a ResNet-CNN-Transformer Parallel Neural Network," in *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, 2021, pp. 803–807.

[19] R. Zhang, H. Wu, W. Li, D. Jiang, W. Zou, and X. Li, "Transformer-based Unsupervised Pre-Training for Acoustic Representation Learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 6933–6937.

[20] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," *arXiv Prepr. arXiv2104.01778*, 2021.