electronics

Article

Speech Emotion Recognition Using Convolutional Neural Networks with Attention Mechanism

Konstantinos Mountzouris [1], Isidoros Perikos [1,2,*] and Ioannis Hatzilygeroudis [1,*]

[1] Department of Computer Engineering and Informatics, University of Patras, 26504 Patras, Greece; mountzour@ceid.upatras.gr
[2] Computer Technology Institute and Press "Diophantus", 26504 Patras, Greece
* Correspondence: perikos@ceid.upatras.gr (I.P.); ihatz@ceid.upatras.gr (I.H.)

Abstract: Speech emotion recognition (SER) is an interesting and difficult problem to handle. In this paper, we deal with it through the implementation of deep learning networks. We have designed and implemented six different deep learning networks, a deep belief network (DBN), a simple deep neural network (SDNN), an LSTM network (LSTM), an LSTM network with the addition of an attention mechanism (LSTM-ATN), a convolutional neural network (CNN), and a convolutional neural network with the addition of an attention mechanism (CNN-ATN), having in mind, apart from solving the SER problem, to test the impact of the attention mechanism on the results. Dropout and batch normalization techniques are also used to improve the generalization ability (prevention of overfitting) of the models as well as to speed up the training process. The Surrey Audio–Visual Expressed Emotion (SAVEE) database and the Ryerson Audio–Visual Database (RAVDESS) were used for the training and evaluation of our models. The results showed that the networks with the addition of the attention mechanism did better than the others. Furthermore, they showed that the CNN-ATN was the best among the tested networks, achieving an accuracy of 74% for the SAVEE database and 77% for the RAVDESS, and exceeding existing state-of-the-art systems for the same datasets.

Keywords: speech emotion recognition; deep learning; deep belief network; deep neural network; convolutional neural network; LSTM; attention mechanism

Citation: Mountzouris, K.; Perikos, I.; Hatzilygeroudis, I. Speech Emotion Recognition Using Convolutional Neural Networks with Attention Mechanism. *Electronics* **2023**, *12*, 4376. https://doi.org/10.3390/electronics12204376

Academic Editors: Juan M. Corchado, Carlos A. Iglesias, Byung-Gyu Kim, Rashid Mehmood, Fuji Ren and In Lee

Received: 17 September 2023
Revised: 9 October 2023
Accepted: 13 October 2023
Published: 23 October 2023

1. Introduction

Speech is the most natural way of human communication. Affective computing systems based on speech play an important role in promoting human–computer interaction, and emotion recognition is the first step. Due to the lack of a precise definition of emotion and the inclusive and complex influence of emotion generation and expression, accurately recognizing speech emotions is still difficult. Speech emotion recognition (SER) is an important problem that is receiving increasing interest from researchers due to its numerous applications, such as e-learning [1], clinical trials [2], audio monitoring/surveillance, lie detection [3], entertainment, video games [4], and call centers [5]. Machine learning (ML) is a revolutionary method in which we feed a machine an adequate amount of data, and the machine will use the experience gained from the data to improve its own algorithm and process data better in the future [6]. One of the most significant approaches in machine learning is the use of neural networks (NNs). Neural networks are networks of interconnected nodes and are loosely modeled towards the way the human brain processes information. Neural networks store data, learn from it, and improve their abilities to sort new data. For example, a neural network with the task of identifying dogs can be fed a set of characteristic values extracted from various images of dogs tagged with the type of dog. Over time, it will learn what kind of image corresponds to what kind of dog. The machine therefore learns from experience and improves itself. Deep learning (DL) is a predominant machine learning approach, where neural networks are enabled at nodes to remember past values with many layers that are trained using massive amounts of data. In

deep learning, the sprawling artificial neural network is fed representations of raw data (e.g., raw image representations) and not given any other instructions. This means that, in contrast to other machine learning approaches, it determines the important characteristics and purpose of the data itself while storing it as experience. In other words, according to studies, deep neural networks (DNNs) can solve the data representation problem through learning a series of task-specific transformations [7]. The network layers extract abstract representations and filter out the irrelevant information, which leads to a more accurate classification and better generalization. Temporal models were also proposed for modelling sequential data with mid- to long-term dependencies. Deep learning models are currently used to solve problems such as face recognition, voice recognition, image recognition, computational vision, and speech emotion recognition. One of the main advantages of deep learning techniques over other machine learning techniques is the automatic selection of features, which could, for example, be applied to important features inherent in audio files that have a special emotion in the task of recognizing speech emotions.

When it comes to recognizing emotion through speech, deep learning models, such as convolutional neural networks (CNN), deep neural networks (DNNs), deep belief networks (DBNs), etc., approach the detection of high-level features for better accuracy compared to hand-made low-level features. Furthermore, the use of deep neural networks enhances the computational complexity of the entire model. However, according to Mustaqeem and Kwon [8] there are still many challenges in recognizing emotion from speech, such as the fact that the current CNN architectures have not shown significant improvement in speech accuracy and complexity in speech signal processing, or the fact that the use of recurrent neural networks (RNNs) and long short-term memory neurons (LSTMs) is useful for training sequential data, but they are difficult to train effectively and are computationally more complex.

Due to the above issues and challenges, in this paper, we implement and test, apart from single instances, CNN- and LSTM-based architectures integrated with an attention mechanism, aiming to explore the impact of the attention mechanism on their performance. Overall, the CNN architecture, with the addition of an attention mechanism, achieved a better performance. The voice characteristics of the speakers are extracted in the form of Mel Frequency Cepstral Coefficients (MFCCs), with the help of the Librosa library.

The main contributions of our study could be stated as follows. First, we perform a concrete survey of the literature and illustrate the works and approaches used in the field. Second, we design and implement widely utilized deep learning methods, enhance them with attention mechanisms, and perform a concrete comparison of the methods, accessing their performances on widely used datasets. The results show the superiority of the methods that use attention mechanism. The results of the article could be of great interest to the related research community.

The structure of the paper is as follows: Section 2 presents related work. In Section 3, six deep neural network configurations are presented. A detailed presentation of the experiments and a discussion of the proposed method compared to other research methods are provided in Sections 4 and 5 respectively Finally, Section 6 concludes the paper.

2. Related Work

In the literature, there is a huge research interest, and several works attempt to perform emotion detection from speech [9,10]. Various works study the way that emotions can be automatically identified and accurately recognized in speech data [11,12]. In this regard, deep learning techniques have achieved breakthrough performance in recent years, and as a result, have been thoroughly examined by the research community [13,14]. Many existing studies in the literature have focused on improving and extending deep learning techniques [15].

In the work presented in [16], the authors present a new random deep belief network (RDBN) method for speech emotion recognition, which consists of a random subspace, DBN and SVM in the context of ensemble learning. It first extracts the low-level characteristics

of the input speech signal and then applies them to the construction of many random sub-intervals. Second, it creates many different sub-intervals. In addition, the DBN continues to use the stochastic gradient descent method to optimize the parameters. To solve the problem, a random space is applied for the training of the basic classifiers as a whole, where the same classification method is used. The best accuracy achieved is 82.32% on the Emo-DB database, 48.5% on the CASIA database, 48.5% on the FAU database, and 53.60% on the SAVEE database.

In the work presented in [17], the authors introduce a method for identifying speech emotions using a spectrogram and convolutional neural network (CNN). The proposed model consists of three convolution layers and three fully connected layers, which extract distinctive features from spectrograph images and predictions for the seven emotions of the Emo-DB Database. Layer C1 has 120 cores (11 $\times$ 11) applied at a rate of four pixels. The ReLU acts as an activation function instead of the standard sigmoid functions that improve the efficiency of the educational process. Layer C2 has 256 cores of size 5 $\times$ 5 and is applied to the input with one step. Similarly, C3 has 384 cores of size 3 $\times$ 3. Each of these convolution layers are followed by ReLUs. Layer C3 is followed by three FC layers that have 2048, 2048, and 7 nodes, respectively. More than 3000 spectrograms were generated from all the audio files in the dataset. Overall, the proposed method achieved 84.3% accuracy.

In [18], the authors present two convolutional neural networks with a long-short memory network (CNN-LSTM), one one-dimensional (1D) and one two-dimensional (2D), stacking four designed local features learning blocks (LFBL). The 1D CNN-LSTM network is intended to recognize the feeling of speaking from raw audio clips, while the 2D CNN-LSTM network focuses on learning high-level capabilities from log-Mel spectrograms. The experimental study was conducted on the Berlin Emo-DB and IEMOCAP databases. The 1D CNN LSTM network achieved 92.34% and 86.73% recognition accuracy on the speaker-dependent and speaker-independent EmoDB databases, respectively, and also delivered 67.92% and 79.72% recognition accuracy on the IEMOCAP speaker-dependent and speaker-independent databases, respectively. The 2D CNN LSTM network achieved 95.33% and 95.89% recognition accuracy on the speaker-dependent and speaker-independent Emo-DB databases, respectively, and delivered 89.16% and 85.58% recognition accuracy on the IEMOCAP speaker-dependent and speaker-dependent experiment databases, respectively.

In the work presented in [19], the authors proposed a new approach to the multi-modal recognition of emotions from simple speech and text data. The attention network implemented consists of three separate convolutional neural networks (CNNs), two for extracting features from speech spectrograms and word integration sequences and one for the emotion classifier. The CNN outputs from word integration and spectrograms are used to calculate an attention matrix to represent the correlation between word integration and spectrograms in relation to emotion signaling. To evaluate the model, they used audio and text data from the CMU-Multimodal Opinion Sentiment and Emotion Intensity (CMU-MOSEI) dataset. The dataset is organized by video IDs and corresponding segments with six emotion and sentiment labels. The video IDs are then further split into segments. The training set consisted of 3303 video IDs and 23,453 segments, while the validation set consisted of 300 non-overlapping video IDs and 1834 segments. The total accuracy of the proposed method was 83.11%.

In [20], the authors present three methods based on CNNs in combination with extensive features, a CNN + RNN and ResNet, respectively. The authors investigate different types of features as the end-to-end frame input, including primary wave data, the Q-transform constant spectrogram (CQT), and the Fourier transform short-term spectrogram (STFT). In this way, the authors create multiple data samples with slightly modified speed ratios, which helps them achieve significant improvements and handle the overfitting issue in the framework from end to end. For their experiments, they used the EmotAsS dataset. The CNN + RNN model achieved the best performance (45.12%) with data balancing; the

CNN model in combination with features showed a performance of 34.33% with data balancing, while the ResNet model achieved a performance of 37.78%.

In the work presented in [21], the authors propose a new architecture called attention-based 3-dimensional convolutional recurrent neural networks (3-D ACRNN) for recognizing emotion from speech, combining CRNN with an attention mechanism, because they hypothesized that calculating delta and delta–deltas for individual functions not only retains effective emotional information but also reduces the effect of emotionally unrelated factors, leading to a reduction in misclassification. First, the CNN 3-D is applied to the entire logarithmic-Mel spectrogram, which has been compiled into a patch that contains only multiple frames. The attention layer then takes a sequence of high-level attributes as the input to generate expression-level attributes. The authors evaluated the model using the Berlin Emotional Speech Database (Emo-DB) and IEMOCAP database. From the ten speakers, for each evaluation, they selected eight as the training data, one as the validation data, and the rest as the test data. The method achieved an accuracy of 64.74% on IEMOCAP database and 82.82% on Emo-DB.

In the work presented in [22], the authors propose an attention-pooling representation learning method for recognizing emotions from speech (SER). Emotional representation is learned from end to end by applying a deep convolutional neural network (CNN) directly to speech spectrograms extracted from speech. Compared to existing aggregation methods, such as max pooling and average pooling, the proposed attention pooling can effectively integrate bottom–up class-agnostic attention maps and top–down class-specific attention maps. Given an expression, they segment it into 2 s sections for training and use an overlay of 1 s to allow them to receive more training data. Each section corresponds to the same tag with the corresponding expression. They used a 1×1 convolutional layer after Conv5 to create a top–down attention map and used another 1×1 convolutional layer to create bottom–up attention maps. The IEMOCAP improvised dataset was used, and the accuracy achieved by the proposed method was 71.75% for WA and 68.06% for UA.

In [23], the authors explore how to take full advantage of low-level and high-level audio features taken from different aspects and how to take full advantage of DNN's ability to merge multiple information to achieve better classification performance. For this reason, they proposed a hybrid platform consisting of three units, namely, a features extraction unit, a heterogeneous unification unit, and a fusion network unit. Besides low-level acoustic features, such as IS10, MFCCs, and eGemaps, that are extracted, high-level acoustic feature presentations named SoundNet bottleneck feature and VGGish bottleneck feature are considered for speech emotion recognition tasks. The heterogeneous integration unit is a Denoising AutoEncoder (DAE), which is a multilayer feed-forward neural network and is introduced in order to convert the heterogeneous space of various features into a unified representation space by deploying this unsupervised feature learning technique. The fusion network module is utilized to capture the associations between those unified joint features for emotion recognition tasks and is constructed as a four-layer neural network, containing one input layer and three hidden layers. They evaluated the model using the IEMOCAP database, and the proposed method improved the recognition performance, reaching an accuracy of 64%.

In the work presented in [24], the authors propose a platform that, at the training layer, has three main stages, such as verbal/non-verbal audio segmentation, the integration of feature extraction, and the construction of an emotion model. The verbal sections were used to train the CNN-based emotion model to derive emotion features, while the non-verbal sections were used to train the CNN audio model to extract audio features. The CNN's combined features are used as the input to the LSTM-based sequence-to-sequence emotion recognition model. Here, the sequence-to-sequence model based on the LSTM with an attention mechanism was selected for emotion recognition. The LSTM and the attention mechanism for developing a sequence emotion recognition model contained a bidirectional LSTM (Bi-LSTM) as the coder for the attention mechanism and an unidirectional LSTM as the decoder for the emotional sequence output. They evaluated the model using the

NTHU-NTUA Chinese interactive multimodal emotion corpus (NNIME); the proposed method achieved a 52.0% accuracy.

The work presented in [25] introduces a model that includes one-dimensional convolutional layers combined with dropout, batch-normalization, and activation layers. The first layer of their CNN receives 193 × 1 number arrays as the input data. The initial layer is composed of 256 filters with a kernel size of 5 × 5 and a stride of 1. After that, batch normalization is applied, and its output is activated by a rectifier linear units layer (ReLU). The next convolutional layer, consisting of 128 filters with the same kernel size and stride, receives the output of a previous input layer. The final convolutional layer, with the same parameters, is followed by the flattening layer and dropout layer, with a rate of 0.2. Their model was tested in the Berlin (EMO-DB), IEMOCAP, and RAVDESS databases and obtained 71.61% for the RAVDESS with eight classes, 86.1% for EMO-DB with 535 samples in seven classes, 95.71% for EMO-DB with 520 samples in seven classes, and 64.3% for IEMOCAP with four classes on speaker-independent audio classification tasks.

Attention-oriented parallel convolutional neural network encoders that capture the essential features required for emotion classification are introduced in [26]. The authors extracted and encoded features such as paralinguistic information and speech spectrogram data, and distinct CNN architectures were designed for each type of feature, and those encoded features were subsequently passed through attention mechanisms to enhance their representations before undergoing classification. Empirical evaluations were carried out on the EMO-DB and IEMOCAP open datasets, and the proposed model achieved a weighted accuracy (WA) of 71.8% and an unweighted accuracy (UA) of 70.9%. Furthermore, with the IEMOCAP dataset, the model yielded WA and UA recognition rates of 72.4% and 71.1%, respectively.

The authors in [27] present a work on enhancing the overall generalization performance and accuracy of SER with a balanced augmented sampling technique on spectrograms that aims to address the imbalance in sample distribution among emotional categories. A deep neural network is utilized, comprising the combination of a convolutional neural network (CNN) and an attention-based bidirectional long short-term memory network (ABLSTM) for feature extraction. Multitask learning is incorporated to enhance the deep neural network's performance. The methodology is assessed on the IEMOCAP and MSP-IMPROV databases, yielding a weighted average recall and unweighted average recall of 70.27% and 66.27% on the IEMOCAP database, respectively, while on the MSP-IMPROV database, the approach achieves 60.90% and 61.83%, respectively.

In the work presented in [28], the authors introduce a method to enhance SER performance by viewing Mel Frequency Cepstral Coefficients (MFCC), which accelerates the learning process while maintaining a high level of accuracy. The authors employ a supervised learning model, specifically a functional support vector machine (SVM), directly on the MFCCs represented as functional data. This enables the utilization of complete functional information, resulting in more precise emotion recognition. The authors' method demonstrates competitive results in terms of accuracy, underscoring its effectiveness in emotion recognition as well as reducing learning time, making it computationally efficient and practical for real-world applications.

A framework called Convolutional Auto-Encoder and Adversarial Domain Adaptation (CAEADA) for cross-corpus SER is introduced in [29]. The CAEADA framework starts by creating a one-dimensional convolutional auto-encoder (1D-CAE) for feature processing. This 1D-CAE is designed to capture correlations among adjacent one-dimensional statistical features, and the feature representation is enhanced through an encoder–decoder-style architecture. Following this, the adversarial domain adaptation (ADA) module works to reduce the differences in feature distributions between the source and target domains by confusing a domain discriminator. Specifically, it employs the maximum mean discrepancy (MMD) method to achieve effective feature transformation. The evaluation results demonstrate that the method performs quite satisfactorily on SER tasks.

In the work presented in [30], the authors present an attention-based dense long short-term memory (LSTM) approach for speech emotion recognition. The authors integrate LSTM networks, which are well-suited for handling time series data such as speech, with attention-based dense connections. This entails the incorporation of weight coefficients into skip connections for each layer, enabling the differentiation of emotional information across layers and preventing interference from redundant information in the lower layers with valuable information from upper layers. The experiments showcase an improvement in recognition performance by 12% and 7% on the eNTERFACE and IEMOCAP datasets, respectively.

3. Methodology

In this section, we present six deep neural networks for recognizing emotions from speech data. To train and test our models, we use the RAVDESS and the SAVEE database. These databases contain '.wav' audio files, which we process to export Mel Frequencies Cepstral Coefficients (MFCCs) [31]. These factors use the Mel scale, which is based on how humans perceive different signal frequencies (the spectral content of the voice signal, as recognized by human hearing). The basic reason that we chose to use MFCCs is that they can provide rich feature content from the data.

The Librosa library helped us to export the sequences of the first 40 MFCCs, shown in Figures 1 and 2, to a RAVDESS audio file. In Figure 1, the axes are the attitude and the time, while on Figure 2, the axes are the MFCCs and the time.

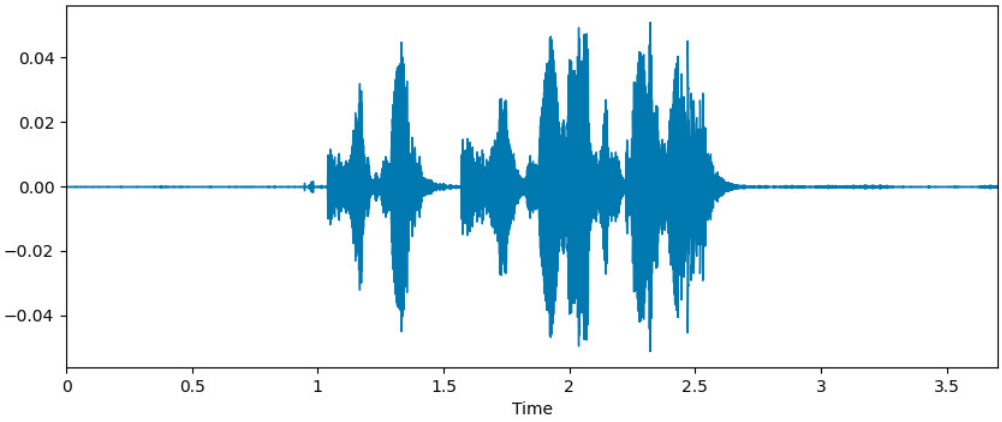

Figure 1. Audio file from the RAVDESS.

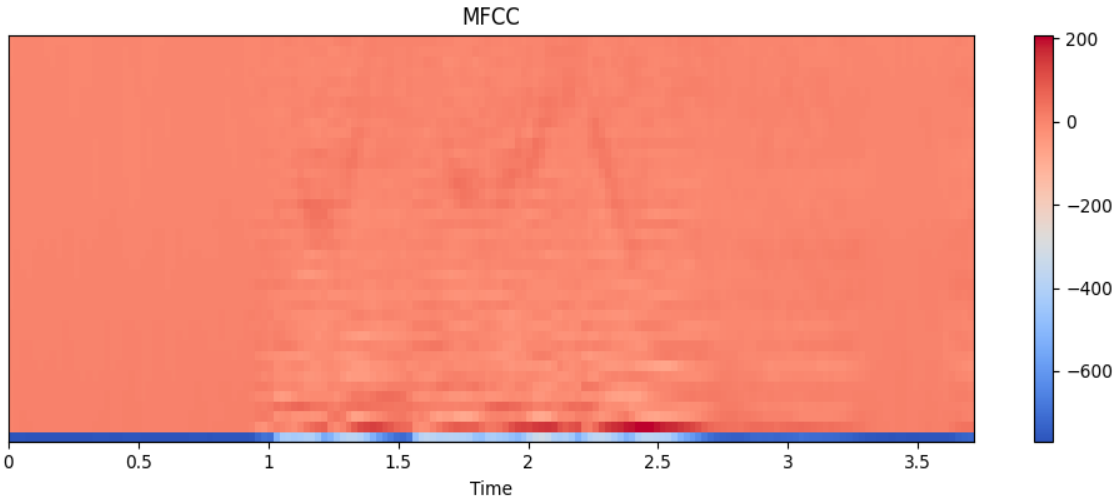

Figure 2. MFCCs of the RAVDESS audio file in Figure 1.

3.1. Deep Belief Network (DBN)

The model of our deep belief network (DBN) is shown in Figure 3. The code of the *deep-belief-network-1.0.3* package from the Github repository (https://github.com/albertbup/deep-belief-network/, accessed on 1 September 2022) was used as the basis for the implementation of this model.

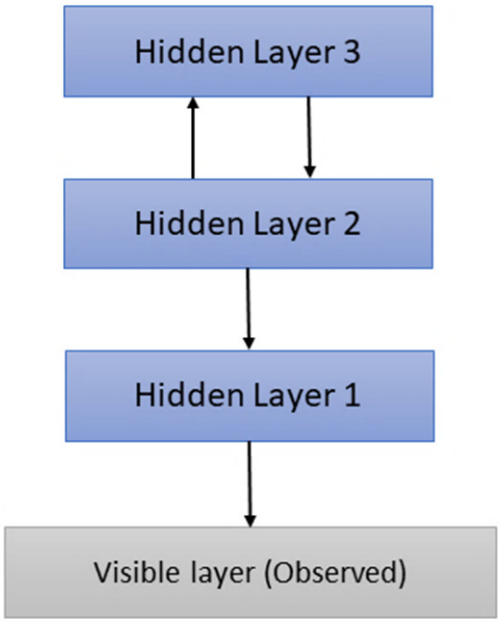

Figure 3. The data flow chart of the deep belief network model.

We created a DBN with three hidden layers (Figure 3) with a size of 128 nodes each, which receives as input the time average of the 40 MFCCs. Each hidden layer can be considered a Restricted Bolzman Machine (RBM), with the hidden layer being the first layer of the next RBM. The values of the hyperparameters of the model we adopted are 0.005 for the learning rate of the RBM hidden layers, 0.05 for the learning rate of the neural network output, and 50 for the number of epochs of the RBM hidden layers. Furthermore, we adopted the value 1000 for the number of iterations for the backpropagation algorithm, used by the neural network's stochastic gradient descent algorithm, and 32 for the batch size. Finally, we used the dropout normalization technique, which is a thoughtful (or random) zeroing of a percentage of connections (which in our case is 20%) between the neurons (i.e., setting the output of the next neuron to 0) of two fully connected layers, as well as the ReLU activation function.

3.2. Simple Deep Neural Network (SDNN)

The model of our simple deep neural network is depicted in Figure 4.

The diagram in Figure 4 shows a deep neural network model that receives the average time of the 40 MFCCs as input. Because we want to implement a DNN model in its simplest form, we use dense layers, for which it is true that each input neuron is connected to the output neuron and that the parameter units simply define the dimension of the output neuron. For the first (hidden) layer of the network, we use a dense layer that receives, as input, a tensor of dimensions (40, 1), due to the 40 MFCCs, and produces an output tensor of dimensions (40, 128), due to the number of 128 parameters we defined. We use the ReLU activation function for this layer. In addition, we employ the dropout normalization technique with a 10% dropout rate. The reason for it is that in each iteration, it trains a modified model that ignores the existence of some of the neurons of the previous or even the next layers, and this results in different groups of network parameters being trained without being affected by other parameters that have been reset each time the code is run. Thus, our network avoids the problem of "overfitting".

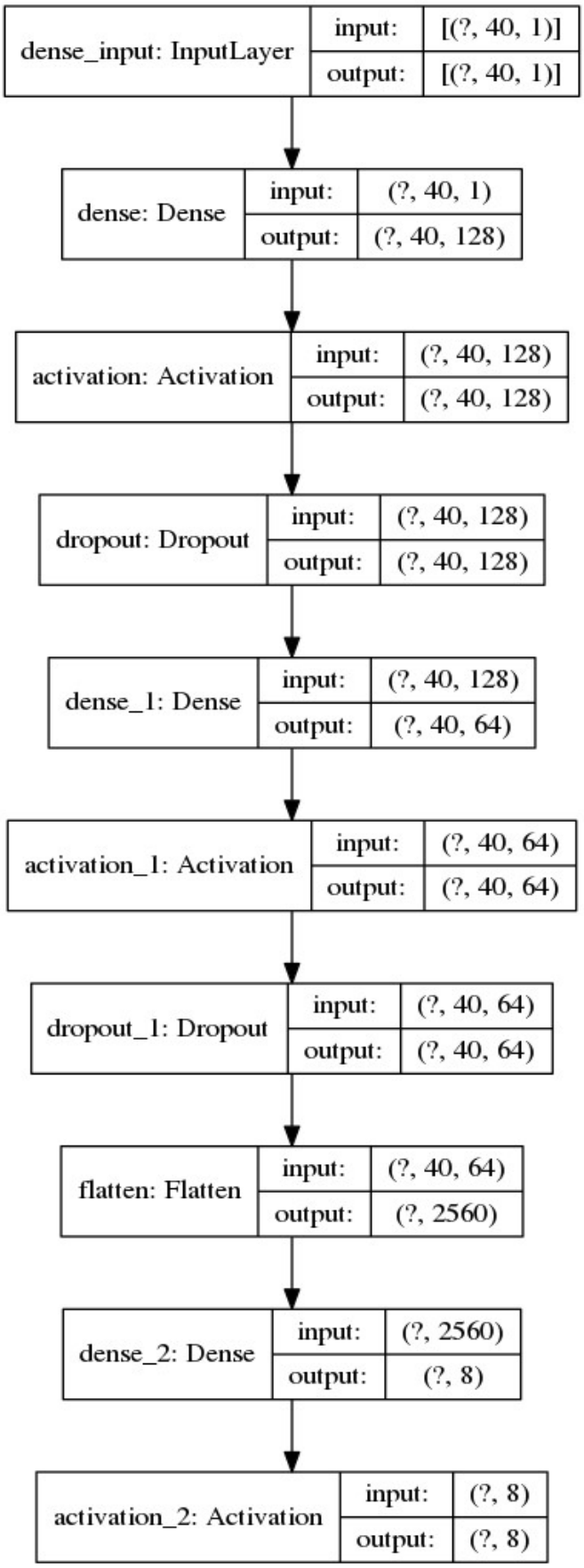

Figure 4. Simple deep neural network model.

For the second (hidden) layer of the network, we use a dense layer that receives, as input, the output of the previous layer, a tensor (40, 128), and an output neuron of dimensions (40, 64), due to the number of 64 parameters we defined for this layer. The ReLU activation function and a dropout of 10% are also used. Successively, we use the flatten function, which reshapes the tensor to have a shape equal to the number of elements contained in the tensor. In our case, the number of elements included in the tensor is 2560.

The third network layer is also a dense layer that receives, as input, the flattened tensor (of 2560 elements) and produces a tensor of size 8 or 7, depending on the database on which we train the model. Essentially, it categorizes the 2560 elements into those 8 or 7 emotion classes. We use the "Softmax" activation function here, which returns a probability distribution over the targeted classes in the multiclass classification problem.

3.3. LSTM-Based Network (LSTM)

As a third model, we configure a DNN that uses LSTM (Long Short-Term Memory) layers, which have been found to achieve very good results in problems with sequential data. The architecture is depicted in Figure 5.

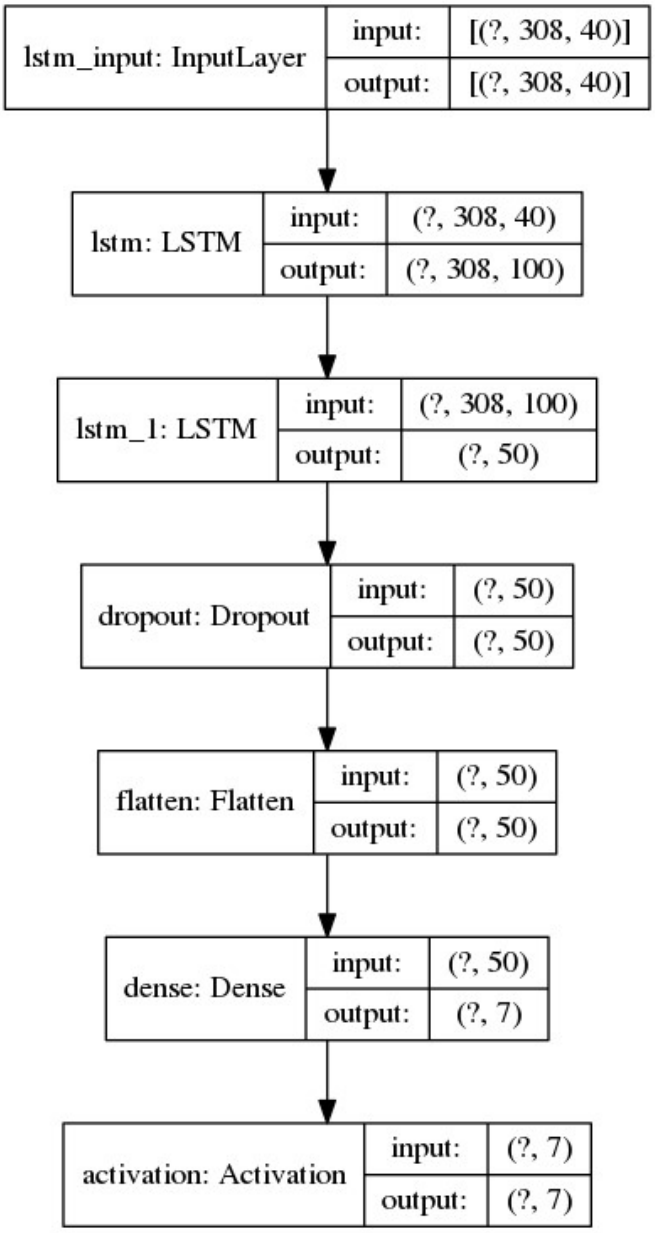

Figure 5. LSTM-based neural network model.

Here, instead of entering the average of the MFCCs over the file in the model, we enter the sequences of the different MFCCs over the length of the file. As the first layer of the network, we use an LSTM layer that receives as input a tensor of dimensions (228, 40) for the RAVDESS and (308, 40) for the SAVEE database, due to the sequence size of the 40 MFCCs (228 and 308, respectively) for each database. It produces output tensors of sizes (228, 100) and (308, 100), respectively, due to the number of parameters (100) that we defined.

As the second layer of the network, we use an LSTM layer that receives as input the output of the previous layer and produces its output tensor of dimension 50, due to the number of parameters, 50 (cells), that we defined. We use the dropout normalization technique here to reset 50% of the connections and then the flatten function, which reshapes the tensor to have a shape equal to the number of elements contained in the tensor (50).

The third network layer is a dense layer that receives as input the flattened tensor of 50 elements and produces a tensor of 8 or 7 components, depending on the database we use to train the model, categorizing the 50 elements into those 8 or 7 classes of emotions. Finally, we use the "Softmax" activation function, which returns a probability distribution over the targeted classes in the multiclass classification problem.

3.4. LSTM-Based Network with Attention Mechanism (LSTM-ATN)

In the previous LSTM model, we add an attention mechanism, and the newly created architecture is presented in Figure 6. As in the previous LSTM-based model, we enter the sequence of the different MFCCs in the file duration. As the first layer of the network, we use the same LSTM layer as the one in the previous model (see Figure 5).

As the second layer of the network, we use the same LSTM layer as in the previous model but without using dropout normalization and the flatten function.

Next, we introduce the attention layer, which applies the dot-product attention mechanism, as it is referred to in Luong [32]. The dot-product attention mechanism calculates scores through the dot product between the current/last target state, $\boldsymbol{h_t}$, and all previous source states, $\widetilde{\boldsymbol{h_s}}$. At each time step, t, in the decoding phase, this approach first takes as input the hidden state, $\boldsymbol{h_t}$, at the top layer of a stacking LSTM. The goal is then to draw a context vector, $\boldsymbol{c_t}$, that retains relevant source information to help predict the current target word, $\boldsymbol{y_t}$. The scoring function is calculated by $score\left(h_t, \widetilde{h}_s\right) = h_t^T \widetilde{h}_s$. In this attention model, an alignment vector of variable length, $\boldsymbol{a_t}$, whose size is equal to the number of time steps in the source side, is obtained by comparing the current hidden target state, $\boldsymbol{h_t}$, with any hidden source state, $\widetilde{\boldsymbol{h_s}}$:

$$a_t(s) = align(h_t, \widetilde{h}_s) = \frac{\exp(score[h_t, \widetilde{h}_s])}{\sum_{s'} \exp(score(h_t, \widetilde{h}_{s'}))}$$

Given the alignment vector as weights, the context vector, $\boldsymbol{c_t}$, is calculated as the dot product of the weighted average over the previous hidden state, $\widetilde{\boldsymbol{h_s}}$. Given the current target state, $\boldsymbol{h_t}$, and the context vector, $\boldsymbol{c_t}$, we use a simple concatenation to combine information from both vectors to create an attentional hidden state as follows: $\widetilde{h}_t = tanh(w_c[c_t; h_t])$. Next, we use the dropout normalization technique to reset the order of 50% of the connections and then the flatten function, which receives the tensor with the attentional vector, $\widetilde{\boldsymbol{h_t}}$, which reshapes it to have a shape equal to the number of elements contained in the tensor. In our case, the number of elements in the tensor is 128.

The third layer of the network is the same as in the previous model (see Figure 5).

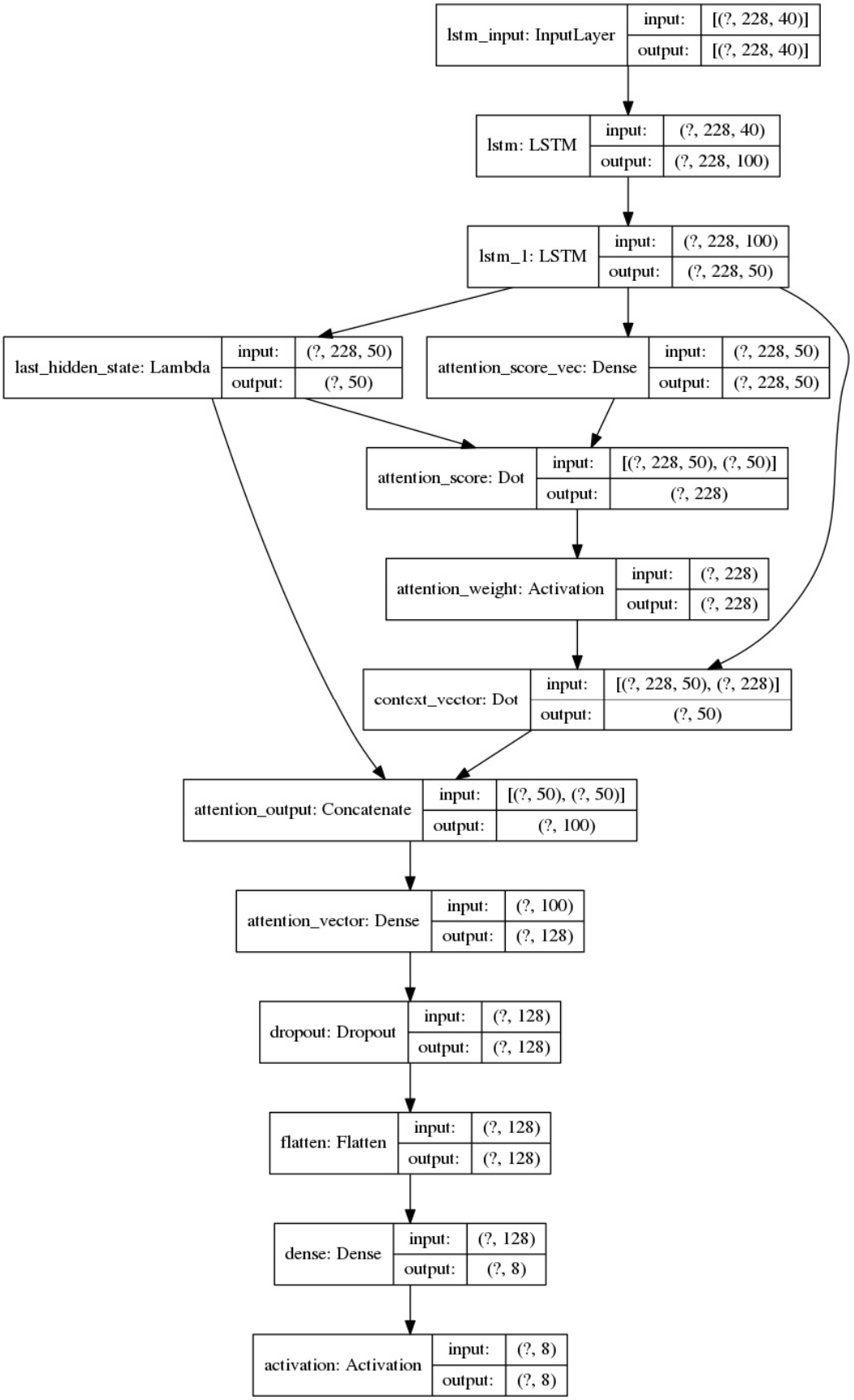

Figure 6. Deep LSTM-based neural network with attention mechanism model.

3.5. Convolutional Neural Network (CNN)

Our next model is a convolutional network that also considers the dimension of time. The architecture of the convolutional network is depicted in Figure 7.

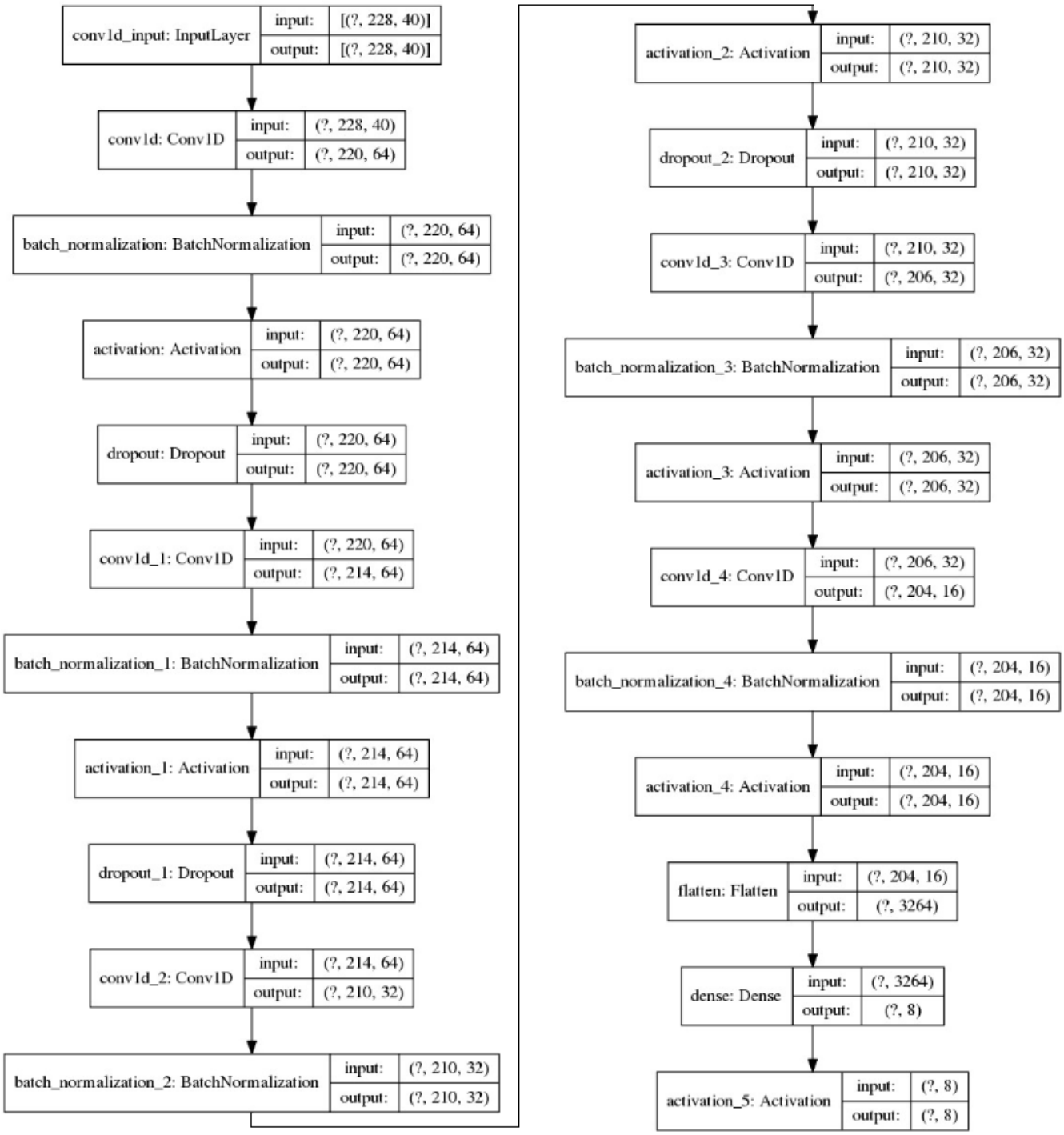

Figure 7. Deep CNN model.

Here, we enter the sequence of the different MFCCs in the file duration too. As the first layer of the network, we use a one-dimension convolutional layer (Conv1D), which receives, as input, a tensor of the dimensions (228, 40) for the RAVDESS and (308, 40) for the SAVEE database, as previously. For the analysis of the model, we use the case of training it with the RAVDESS database, as shown in Figure 7. The first convolutional layer, in this case, will produce an output tensor of dimensions (220, 64), due to the number of

64 filters we have defined. Additionally, we set the kernel size, which refers to the size of the convolution window, to 9, as we are working with a convolutional layer of one dimension. Next, we use the batch-normalization layer, which speeds up training as it allows the use of higher learning rates, given that activations are now less "sensitive" to changes in parameters. Finally, we use the ReLU activation function and the dropout normalization technique at a rate of 50%.

As for the second layer of the network, we use a convolutional layer of one dimension (Conv1D), which receives as input a tensor of dimensions (220, 64) and produces an output tensor of dimensions (214, 64), due to the number of 64 filters that we defined. Additionally, we set the kernel size to 9, as explained above. Next, we use the batch-normalization layer, which speeds up training using the ReLU activation function and a dropout of 50%.

The third layer of the network is implemented as a convolutional layer of one dimension (Conv1D), which receives as input a tensor of dimensions (214, 64) and produces an output tensor of dimensions (210, 32), due to the number of 32 filters that we defined. Additionally, we set the kernel size to 7 and use the batch-normalization layer to speed up training using the ReLU activation function and a dropout normalization rate of 50%.

The fourth layer of the network is the same as the third one.

The fifth layer of the network is implemented as a convolutional layer of one dimension (Conv1D), which receives as input a tensor of dimensions (206, 32) and produces an output tensor of dimensions (204, 16), due to the number of 16 filters that we defined. Additionally, we set the kernel size to 5 and use the batch-normalization layer to speed up training using the ReLU activation function.

In the sequel, we use the flatten function (in this case, the tensor contains 3264 elements). Next, a dense layer that receives as input the flattened tensor (hence it has 3264 elements) produces a tensor of size 8 or 7, depending on the database in which we train the model, categorizing the 3264 elements into their 8 or 7 emotion classes. Finally, we use the "Softmax" activation function.

3.6. Convolutional Neural Network with Attention Mechanism (CNN-ATN)

Our last model is a deep convolution network with the addition of an attention mechanism, as presented in Figure 8.

Here, we again enter the sequence of the different MFCCs in the file duration. The first, the second, and the third layers of the network are the same as those of the plain CNN model.

The fourth layer of the network is implemented as a convolutional layer of one dimension (Conv1D), which receives as input a tensor of dimensions (210, 32) and produces an output tensor of dimensions (206, 32). The first dimension (210) is the result of reducing the input one due to the elimination of zero values, whereas the second dimension is the number of 32 filters we have defined. Additionally, we set the kernel size to 7. Next, we use the batch-normalization layer, which speeds up training using the ReLU activation function.

The fifth layer of the network is the same as the fifth layer of the simple CNN.

After that, we enter the attention layer, which applies the dot-product attention mechanism in the same way as in Section 3.4. An alignment vector of variable length, $\boldsymbol{a_t}$, is calculated by the same formula. Next, we use the flatten function, which receives the tensor with the attentional vector, $\widetilde{\boldsymbol{h}}_{\boldsymbol{t}}$, which reshapes it to have a shape equal to the number of elements contained in the tensor (128 in our case).

Then, a dense layer is used, which receives as input the flattened tensor that has 128 elements and produces an 8- or 7-component tensor, as in previous models, where the "Softmax" activation is used.

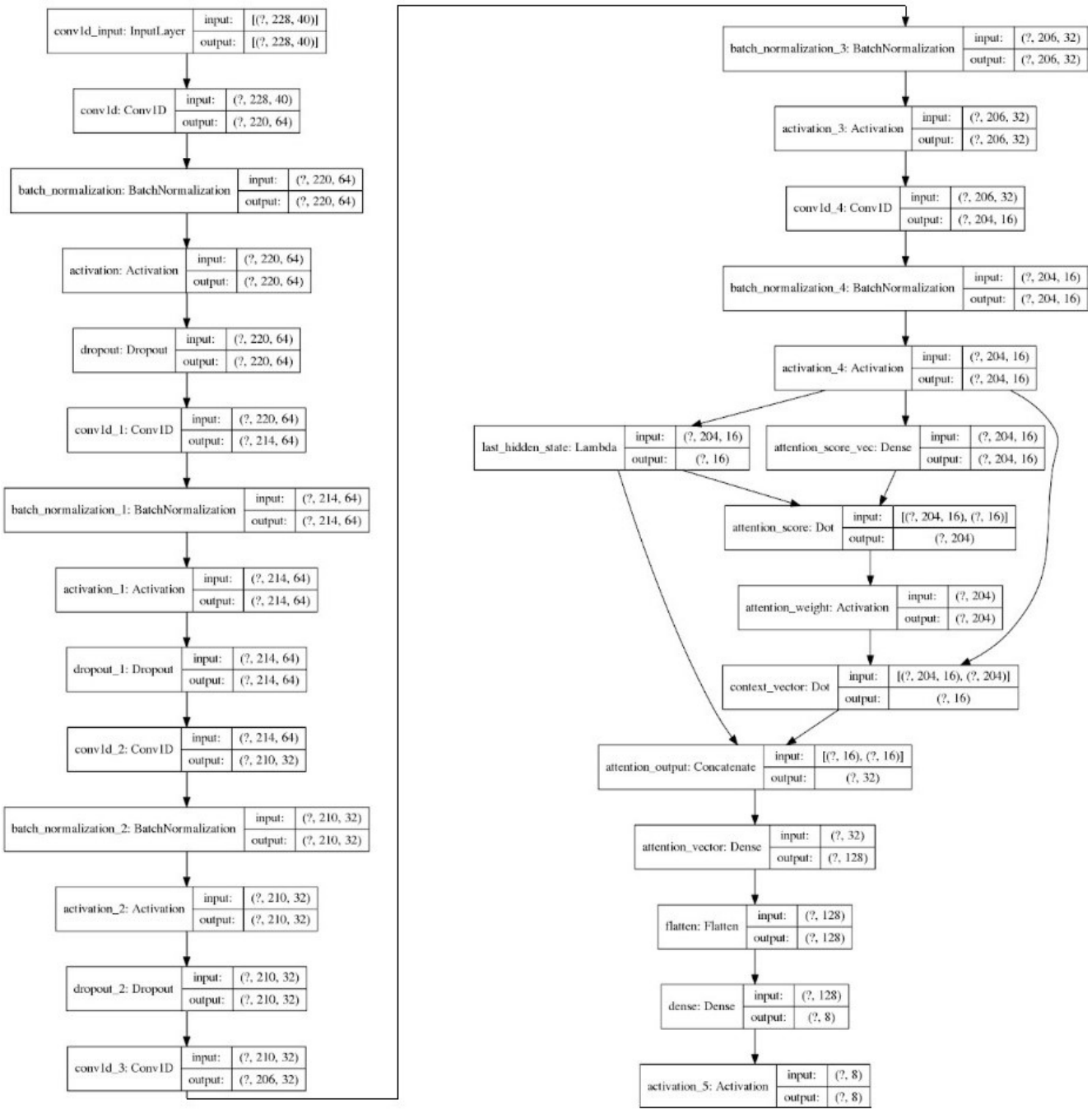

Figure 8. Deep CNN neural network with attention mechanism model.

In the realm of neural networks, hyperparameters serve as pivotal configurations for delineating the learning capacity and intricacy of the model. These encompass critical elements such as the number of neurons, activation function, optimizer choice, learning rate, batch size, and the number of training epochs. Additionally, the selection of an appropriate number of layers represents a crucial hyperparameter with a profound impact on the model's predictive accuracy. To fine-tune these hyperparameters, an initial examination was conducted, drawing insights from established code samples within the domain of speech emotion recognition, accessible from reputable repositories such as https://github.com/xuanjihe/speech-emotion-recognition (accessed on 1 September 2022), among others. Furthermore, the determination of specific hyperparameters, including the activation function, was informed by a comprehensive review of the pertinent literature. In the pursuit

of optimizing the results, a series of tailored adjustments were incorporated, culminating in the development of our final model. For instance, the number of convolutional layers was judiciously determined by considering the interplay between database size and model performance. Notably, an augmented number of layers yielded superior outcomes with larger datasets, while a reduction in layers proved more effective with smaller sample sizes. This deliberation guided our decision to employ two distinct databases, ensuring robust performance across varying data sample magnitudes.

3.7. Implementation Tools

For all the above implementations, except the DBN model, for which we utilized the CPU, we utilized Tensorflow and Keras. *TensorFlow* uses dataflow graphs to represent the calculation, the shared state, and the functions that transform that state. It maps the nodes of a data flow graph to multiple machines in a cluster and within a machine on multiple computing devices, including multiple CPU cores, general-purpose GPUs, and specially designed ASICs (Application-Specific Integrated Circuits), known as Tensor Processing Units (TPUs). *Keras* is an open-source library that provides a Python interface for artificial neural networks. Built on top of TensorFlow, *Keras* is an industry-strength framework that can scale to large clusters of GPUs or an entire TPU pod. As a Python distribution platform, we used *Anaconda*. For training purposes, we used the *Adam* optimizer and "*sparse categorical crossentropy*" as our loss function. The BDN model utilized a learning rate of 0.05, and the simple DNN model utilized a learning rate of 1×10^{-4} (0.0001), where the rest of the models (LSTM with and without attention, CNN with and without attention) utilized a learning rate of 1×10^{-3}, which becomes 1×10^{-4} in 100 epochs. Moreover, the batch size we used for neural network training for all the models, except DBN, is 32. The specifications of our running system are the following: *GPU: NVIDIA GTX 1070*, *CPU:* Intel(R) Core (TM) i7-9750H, *RAM: 16 GB.*

4. Experimental Study and Results

In this section, we first present the datasets that we have used as well as the results of the six implemented models, presented in Section 3, and then compare the best of them (called 'our method') with state-of-the-art models. The models were trained, splitting both databases into a training set and a validation (test) set at a rate of 80–20%, respectively.

4.1. Datasets

For our experiments, we used two datasets, the SAVEE (Surrey Audio-Visual Expressed Emotion) database (http://kahlan.eps.surrey.ac.uk/savee/Download.html, accessed on 1 September 2022) [33], and the RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) database (https://zenodo.org/record/1188976, accessed on 1 September 2022) [34].

SAVEE is an audio–visual dataset that comprises 480 English utterances from four male actors, aged from 27 to 31 years, in seven different emotions, which are anger, disgust, fear, happiness, sadness, surprise, and neutral. The utterances are categorically labeled. The recordings consisted of 15 TIMIT database (https://catalog.ldc.upenn.edu/LDC93s1, accessed on 1 September 2022) sentences per emotion (with an additional 30 sentences for the neutral state). The emotion assessment of the recordings was performed by subjective evaluation under audio, visual, and audio–visual scenarios. The speech data were labeled at the phone level in a semi-automated way. The audio data sampling rate used was 44.1 kHz.

The RAVDESS contains a total of 1440 speech utterances and 1012 song utterances. Each audio file was rated on a scale of 10 on intensity, emotional validity, and genuineness. The emotion in the speech includes surprise, happy, calm, fearful, sad, disgust, and angry. The song section contains happy, calm, sad, angry, and fearful emotions. Every file has two levels of emotional intensity: normal and strong. The RAVDESS is very rich in nature

given that it does not suffer from gender bias and consists of a wide range of emotions at different levels of emotional intensity.

4.2. Experimental Results—Accuracy and Loss Curves

In the following, we present the results in the form of graphs for the accuracy and loss of our six models, through Figures 9–24. Where there is need, we add some comments. It is important to note here that compared to the other models, the deep belief network (DBN) algorithm package provides only loss information, so we have only one graph for each dataset.

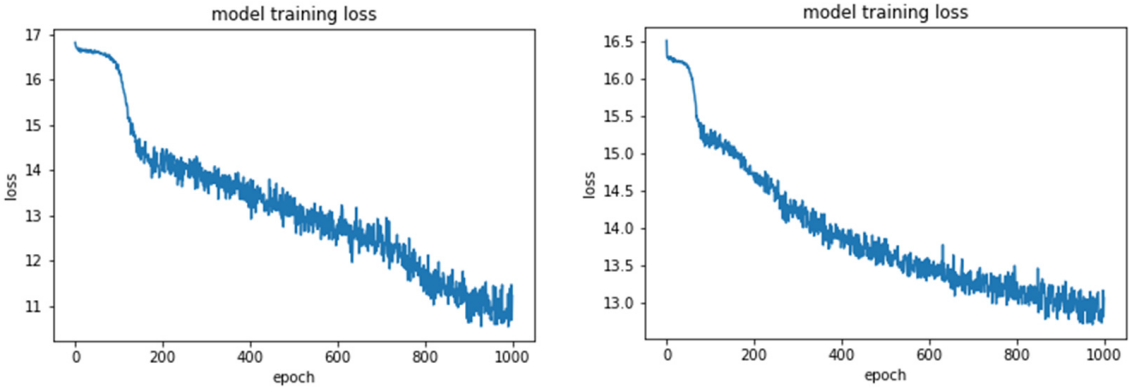

Figure 9. DBN loss training charts for RAVDESS (**left**) and SAVEE (**right**) database.

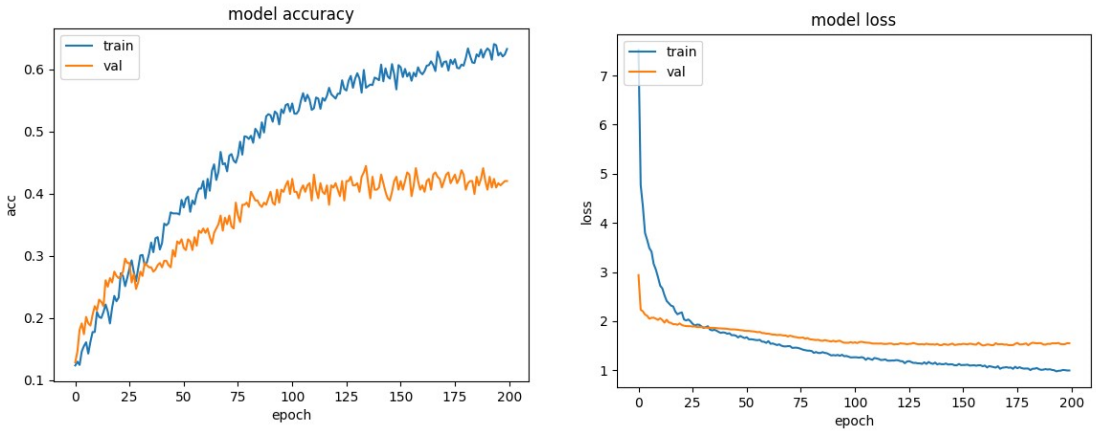

Figure 10. DNN training charts for RAVDESS.

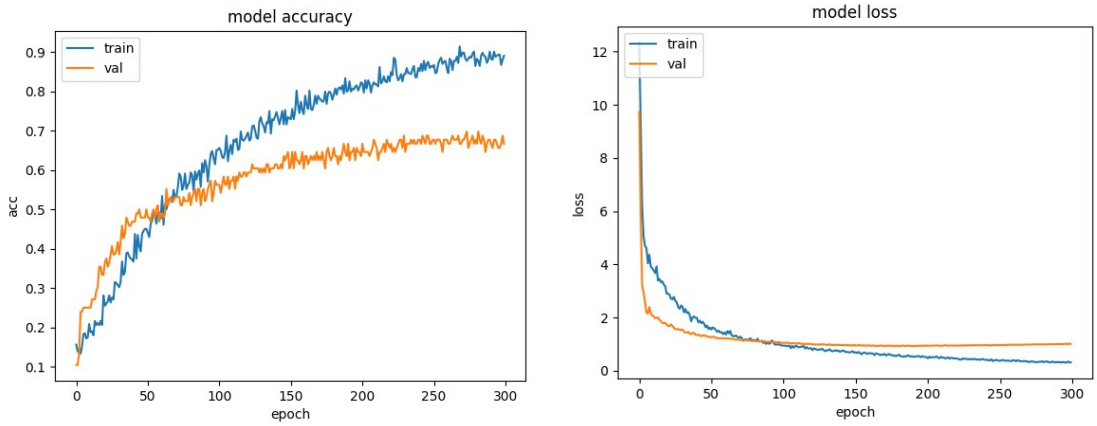

Figure 11. DNN training charts for SAVEE database.

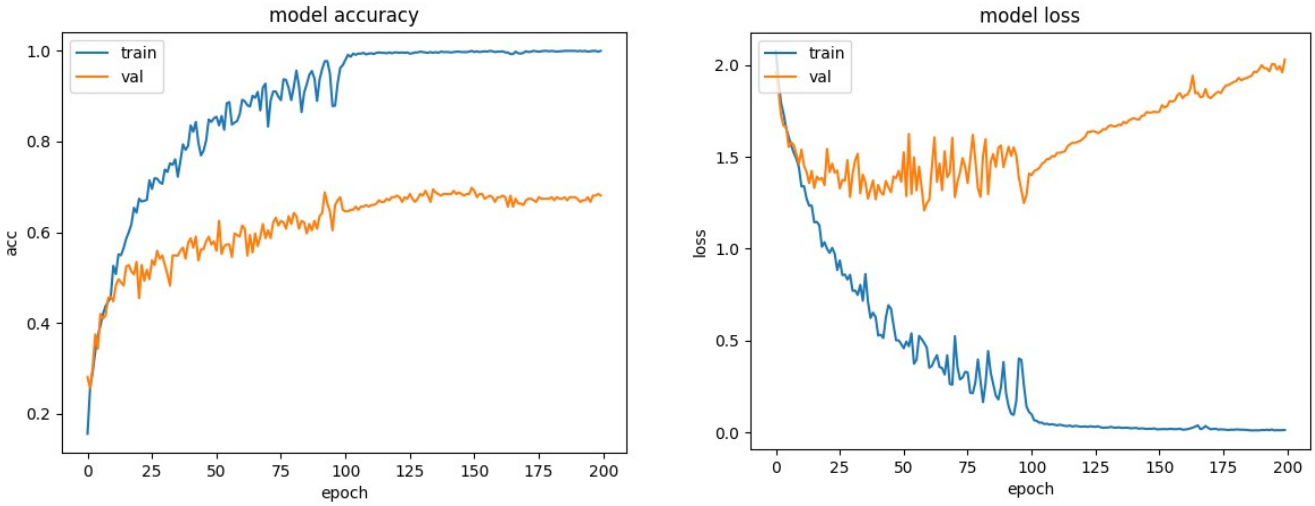

Figure 12. LSTM network training charts for RAVDESS.

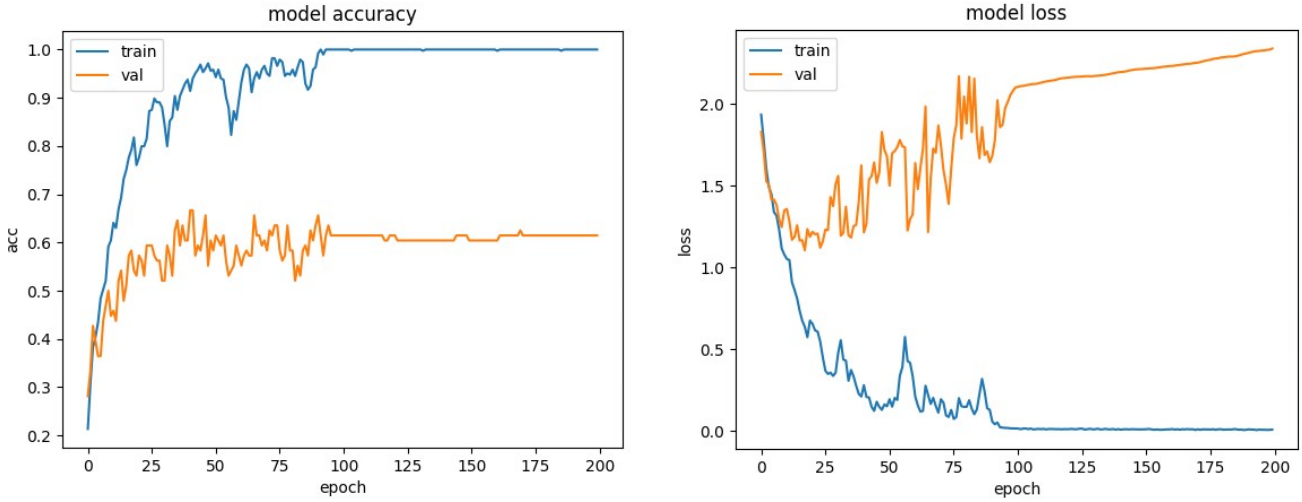

Figure 13. LSTM network training charts for SAVEE database.

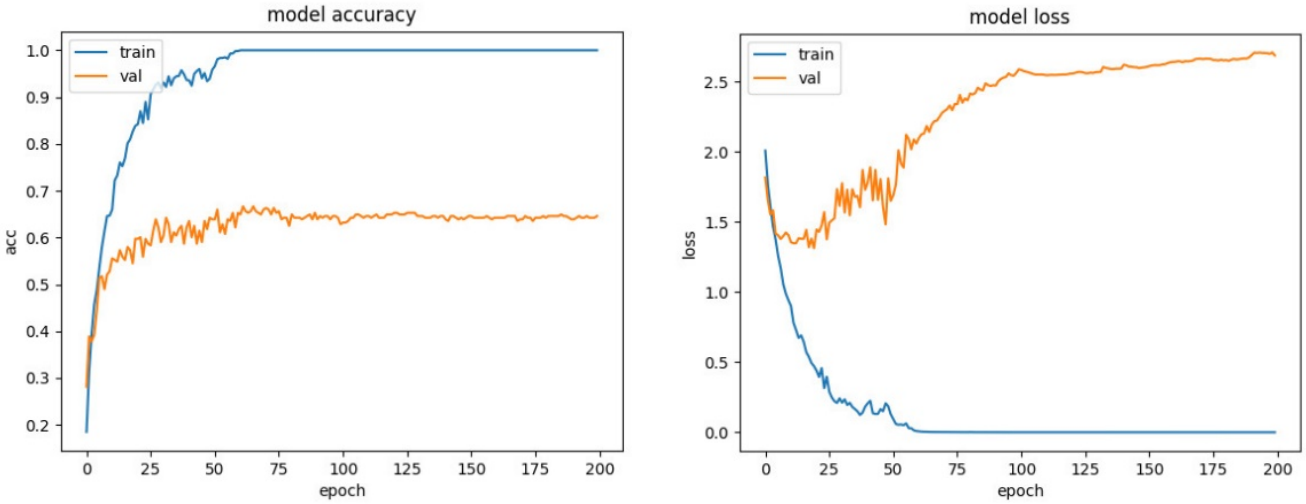

Figure 14. LSTM network with attention mechanism training charts for RAVDESS.

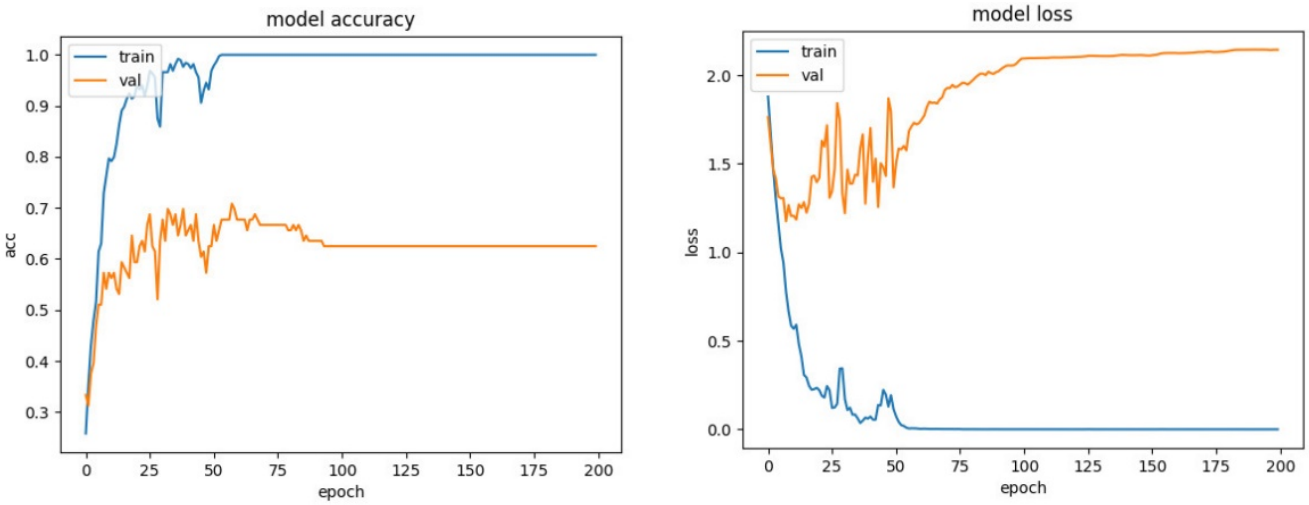

Figure 15. LSTM network with attention mechanism training charts for SAVEE database.

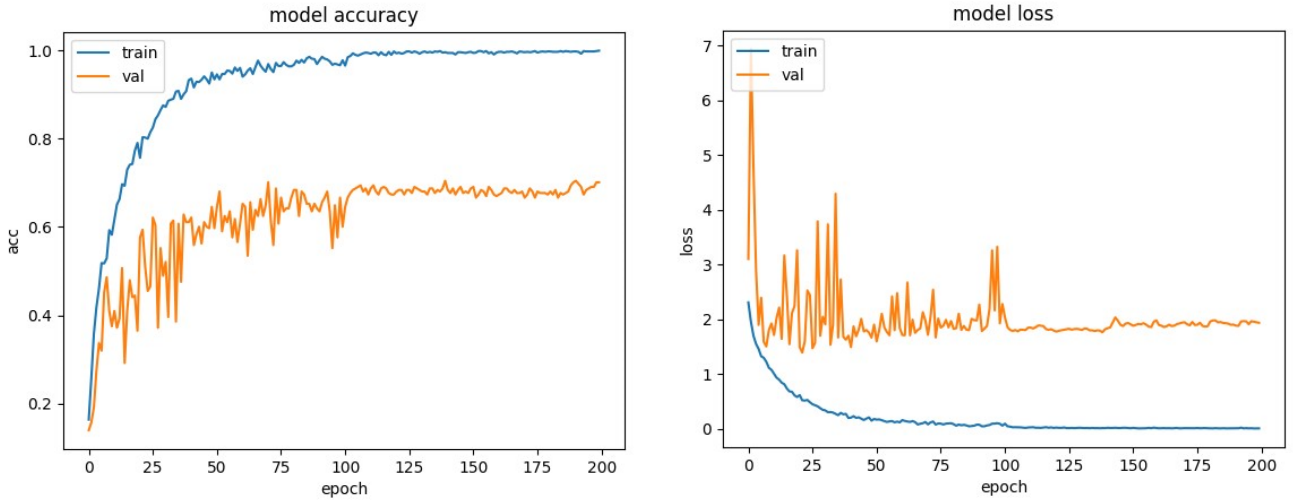

Figure 16. CNN network training charts for RAVDESS.

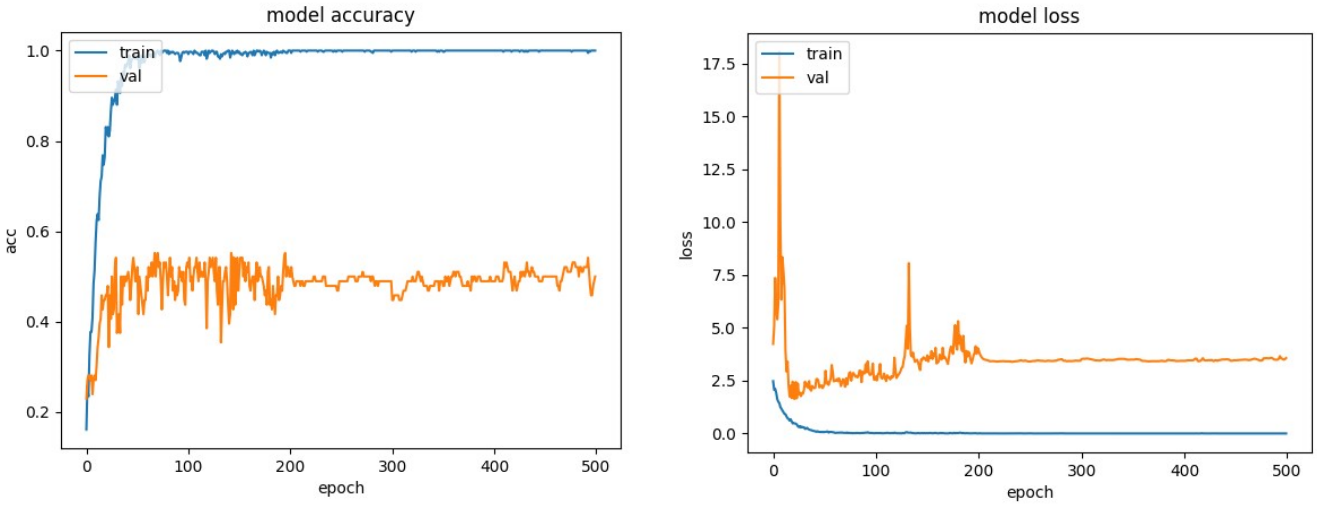

Figure 17. CNN network training charts for SAVEE database.

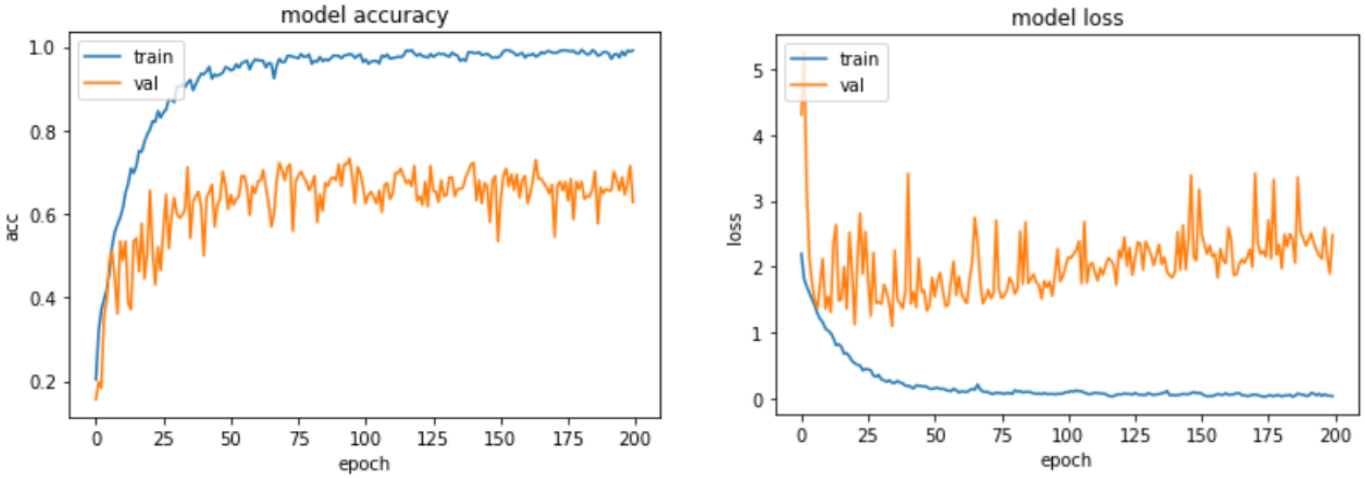

Figure 18. CNN network training charts for RAVDESS with zero padding.

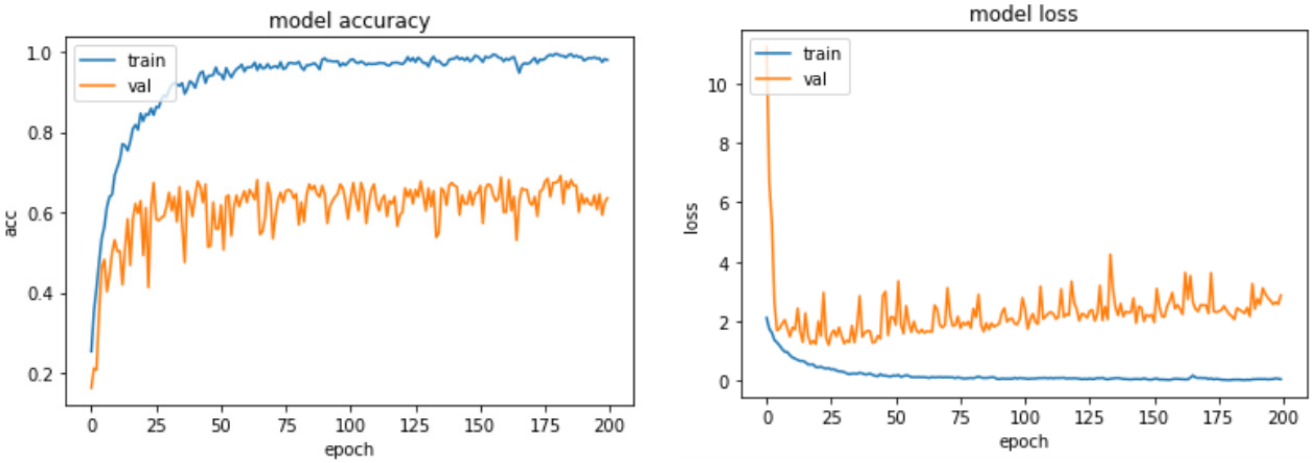

Figure 19. CNN network training charts for RAVDESS with zero padding and LeakyReLU.

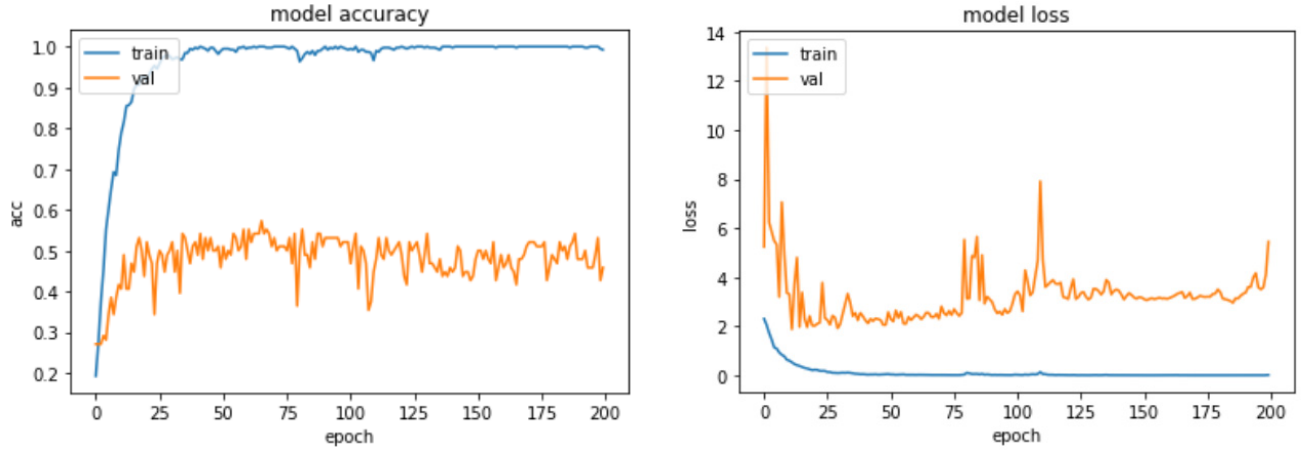

Figure 20. CNN network training charts for SAVEE database with zero padding and LeakyReLU.

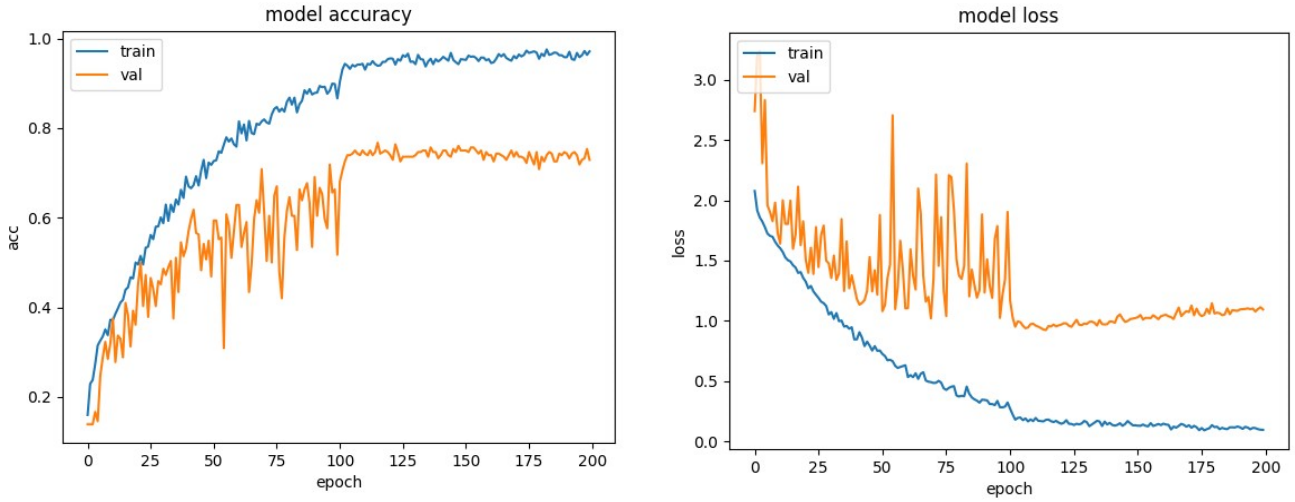

Figure 21. CNN network with attention training charts for RAVDESS.

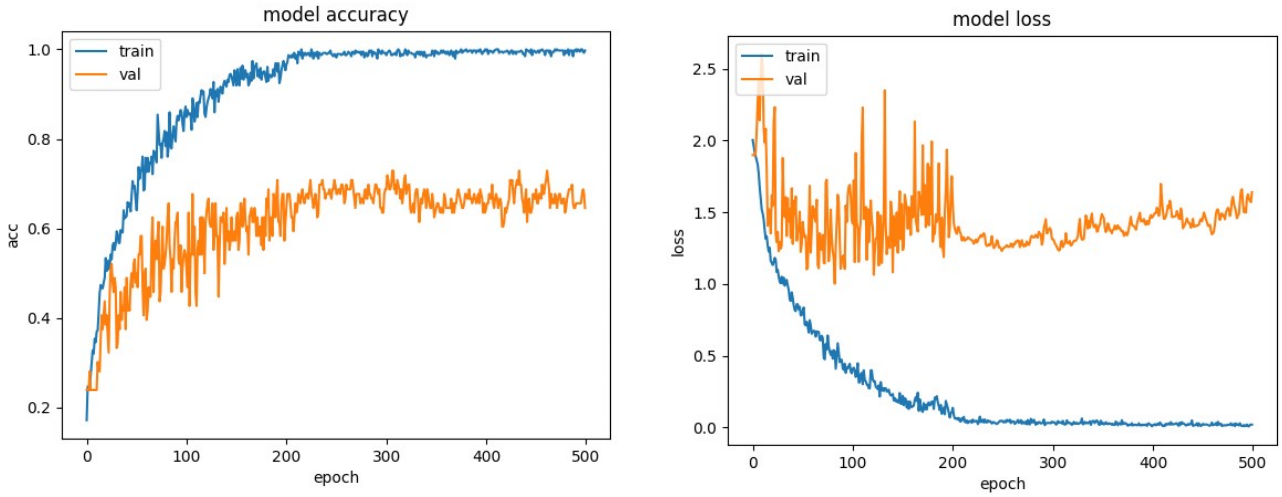

Figure 22. CNN network with attention training charts for SAVEE database.

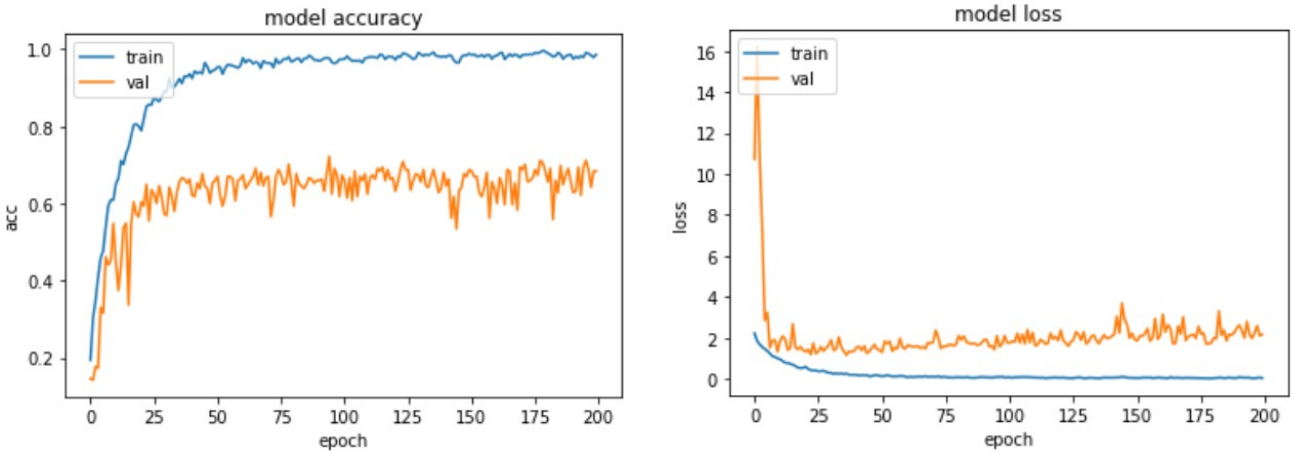

Figure 23. CNN network with attention training charts for RAVDESS without dropout.

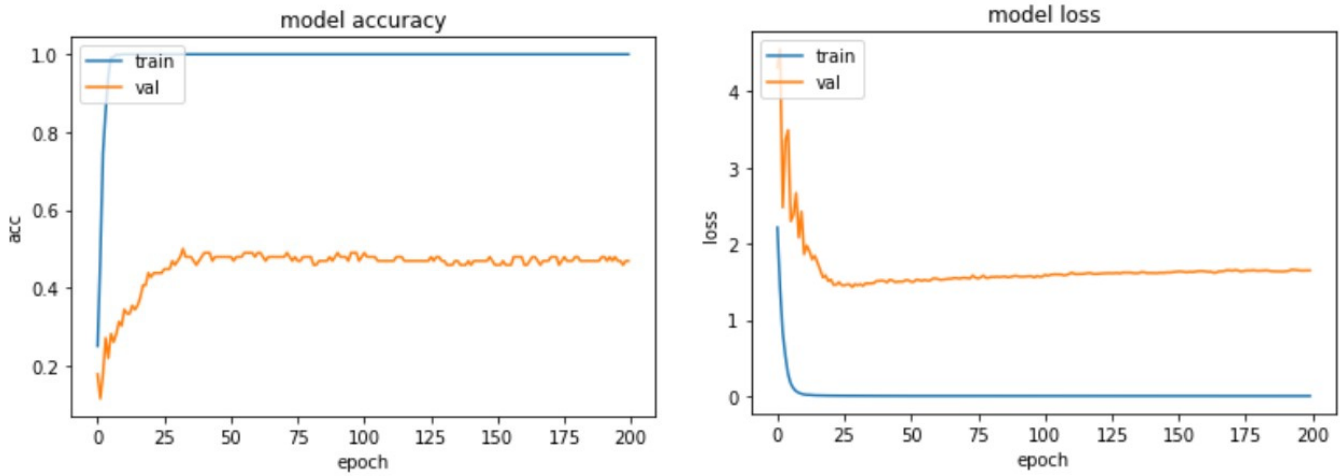

Figure 24. CNN network with attention training charts for SAVEE database without dropout.

The model in Figures 16 and 17 uses batch-normalization and dropout techniques with a 50% reset to prevent a large overfitting of the model, but still, no good generalization is achieved. For this reason, two more variations of our model were tested.

Initially, zero padding was tested on the input vectors at the hidden levels. This was to keep the dimensions of the vector containing the input sequence the same as the model proceeds to the next hidden levels (for RAVDESS 228, and for SAVEE 308). This test was performed on the RAVDESS, and the accuracy and loss curves are presented in Figure 18.

As we can see from the graphics of the test with zero padding compared to our model in Figure 16, it is obvious that the accuracy remains the same (70.5%), as well as the overfit (30%). However, comparing the graph of the loss between the two implementations, we observe that the error from epoch 50 onwards in the zero-padding test gradually increases and does not stabilize after epoch 100 at a value of two, as is the case for the model in Figure 16. On the contrary, it continues to grow and exceeds a value of two.

Then, after the first test for the RAVDESS had not yielded the desired results, a second test was performed using zero padding in combination with the replacement of the ReLU activation function with the LeakyReLU. The ReLU activation function sets all negative values to zero. This makes our network adaptable to ensure that the most important neurons have positive values (>0). However, this can also be a problem, as the gradient of 0 is 0, and therefore, neurons that reach large negative values cannot recover and stick to 0. This causes the neurons to die, and for this reason, this phenomenon is called "The dying ReLU problem".

To avoid this phenomenon, a LeakyReLU was proposed, according to which negative values instead of zero are multiplied by a factor of 0.01:

$$f(x) = \begin{cases} 0.01, \ if \ x < 0 \\ x, \ x \geq 0 \end{cases}$$

The LeakyReLU has a slight negative slope, avoiding the entrapment of neurons with negative values at 0. There have generally been reports of success when adopting this activation function, but the results are not always consistent [35]. The reason we want to apply it to the CNN model is that through its operation, we reduce the phenomenon of overfitting a little more and stabilize more the instabilities of our system, which appear as abrupt changes (spikes) in the graphics.

This test was performed on the RAVDESS and the SAVEE database, and the accuracy and loss curves are depicted in Figures 19 and 20, respectively.

As we can see from the graphs of the test with zero padding and LeakyReLU in the RAVDESS base, compared to our model in Figure 16, it is obvious that the accuracy decreases slightly (67%) and the overfitting increases slightly (33%). Comparing the graphs

of loss between the two implementations, we notice that the instability (spikes) of the system was improved. However, the error from epoch 50 onwards, in the test with zero padding and a LeakyReLU, increases gradually and does not stabilize after epoch 100 at a value of two, as is the case with the model in Figure 16. On the contrary, it continues to increase, approaching values such as three and four.

Comparing the graph of test accuracy with zero padding and the LeakyReLU on the SAVEE database with that of Figure 17, we notice that the accuracy increases dimly (55.7%) compared to the model in Figure 17 (55.2%), so there is a very minimal reduction in overfitting of 0.5%. Then, comparing the loss graphs between the two implementations, we notice that the instability (spikes) of the system was slightly improved. The error from epoch 50 onwards, in the test with zero padding and the LeakyReLU, increases gradually and does not stabilize, approaching 3.5. This is a small improvement of the system, as the loss in Figure 17 approaches 4.5 in epoch 200, but from epoch 200 onwards stabilizes.

The above comparison of the two tests with the implementation of Figure 16 in the RAVDESS concludes that our model, with the use of a ReLU and without zero padding, gives overall better results in terms of accuracy and loss. Regarding the comparison of the test with zero padding and the LeakyReLU with the implementation of Figure 17 for the SAVEE database, we observe that the test with zero padding and the LeakyReLU achieve faintly improved overall results in terms of both accuracy and loss. This may be due to the fact that this database has a smaller number of samples than RAVDESS. Therefore, we conclude that the model implemented with the ReLU and without zero padding performs in general better and that the techniques tested do not give the desired results, i.e., reduce the overfitting effect and the loss to a significant degree.

Because we see that the models show a degree of overfitting that cannot be further reduced, we performed a test where we removed the dropout normalization technique from the model so that only the batch-normalization technique remained to see the effects. The results of this model (without the dropout) for the RAVDESS and SAVEE database are depicted in Figures 23 and 24, respectively.

It is obvious that with the removal of the dropout function, the model shows a significant increase in the overfitting effect, given that for both the RAVDESS and SAVEE dataset, we observe a significant reduction in the accuracy of the model and some increase in loss. More specifically, for the RAVDESS, the model without the dropout function achieves 65% accuracy, in contrast to the model in Figure 21, which achieves 77% accuracy. This means that the model without the dropout technique overfits to a percentage of 45% on this dataset. In addition, the loss of the model without the dropout technique fluctuates (spikes) around 2.5, while the model in Figure 21 fluctuates (spikes) around 1.5 and from epoch 100 onwards, revolves around 1, with a minimal upward trend to 1.2.

For the SAVEE dataset, the model without the dropout technique achieves an accuracy of 47%, in contrast to the model of Figure 22, which achieves an accuracy of 74%. This means that the model without the dropout technique overfits to a percentage of 63% on this dataset. Furthermore, the loss of the model without the dropout technique from epoch 25 onwards has a steady upward trend from 1.3 to 1.6, while the model in Figure 22 fluctuates (spikes) around 1.5. From the above, it is obvious that the proposed CNN-attention model applying the dropout technique against overfitting is the best possible option.

4.3. Experimental Results—Confusion Matrices

For the six implemented models, the confusion matrices for each of the RAVDESS and SAVEE dataset are presented in this subsection via Figures 25–30. The confusion matrices, as it is well known, contain information about the success rate of the models' predictions for each emotion of the selected datasets separately. Table 1 shows the correspondence between the labels in the confusion matrices and the two datasets.

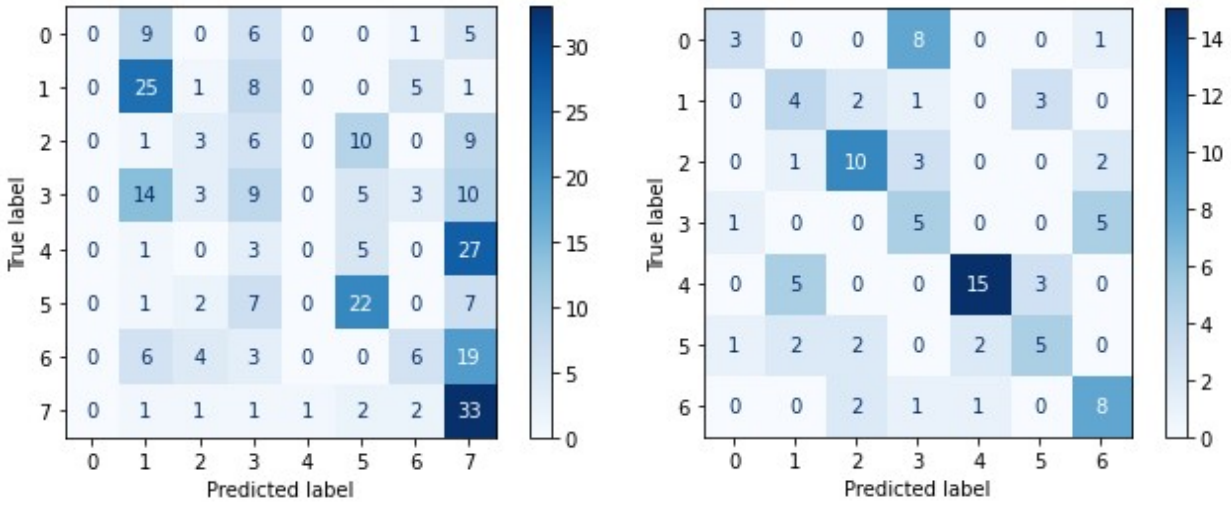

Figure 25. DBN confusion matrices for RAVDESS (**left**) and SAVEE (**right**).

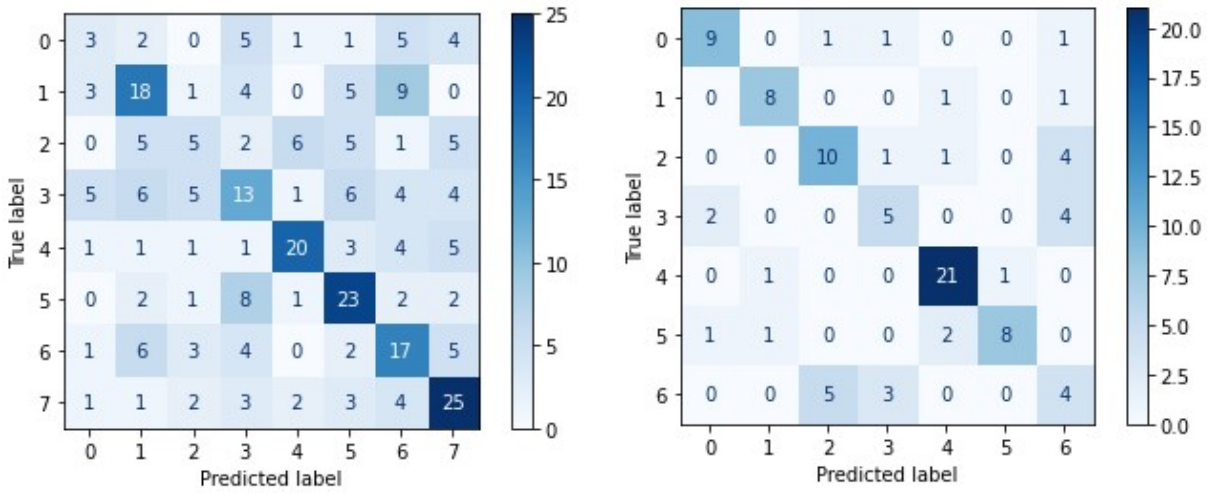

Figure 26. DNN confusion matrices for RAVDESS (**left**) and SAVEE database (**right**).

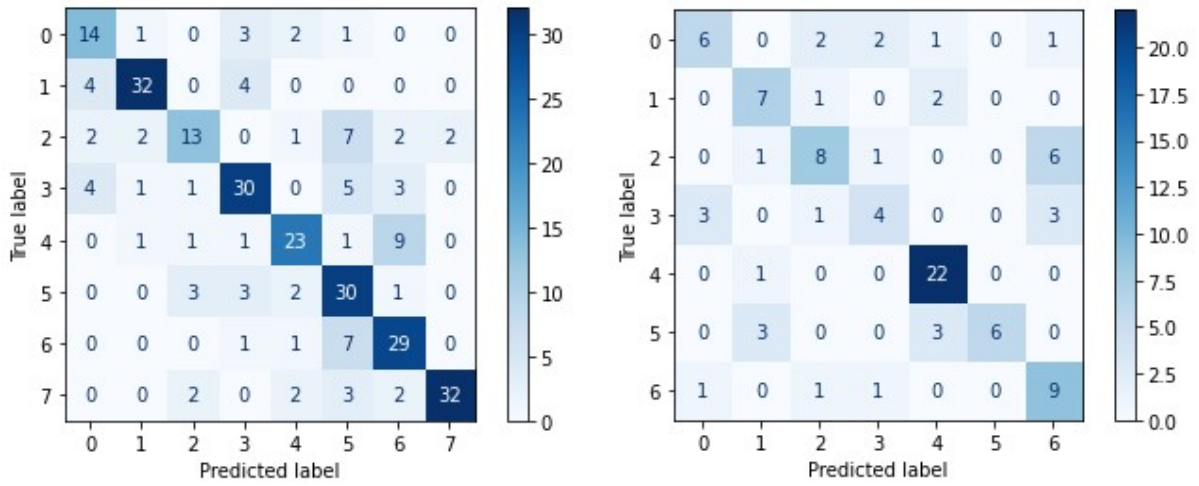

Figure 27. LSTM network confusion matrices for RAVDESS (**left**) and SAVEE database (**right**).

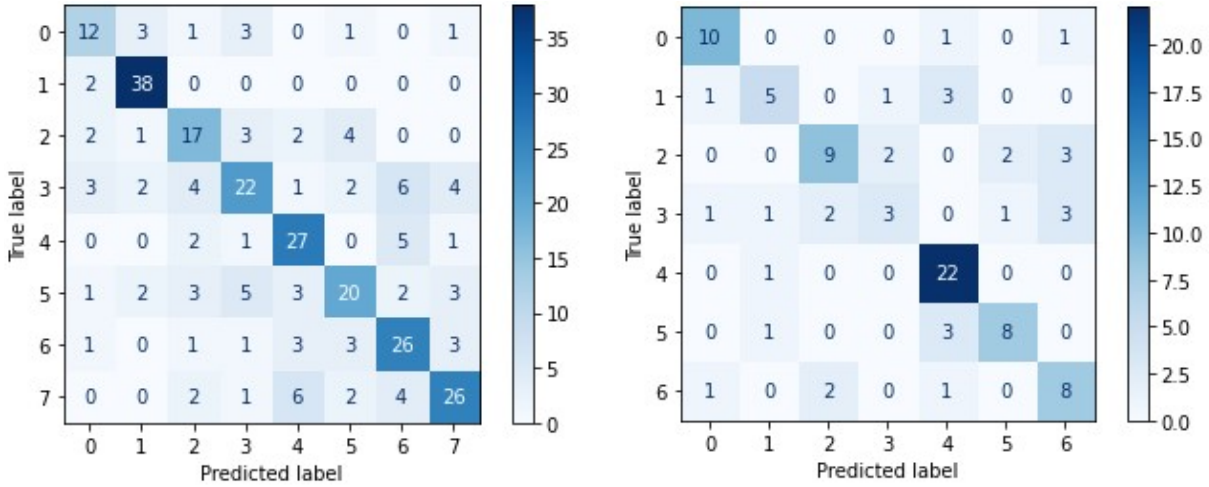

Figure 28. LSTM network with attention mechanism confusion matrices for RAVDESS (**left**) and SAVEE (**right**).

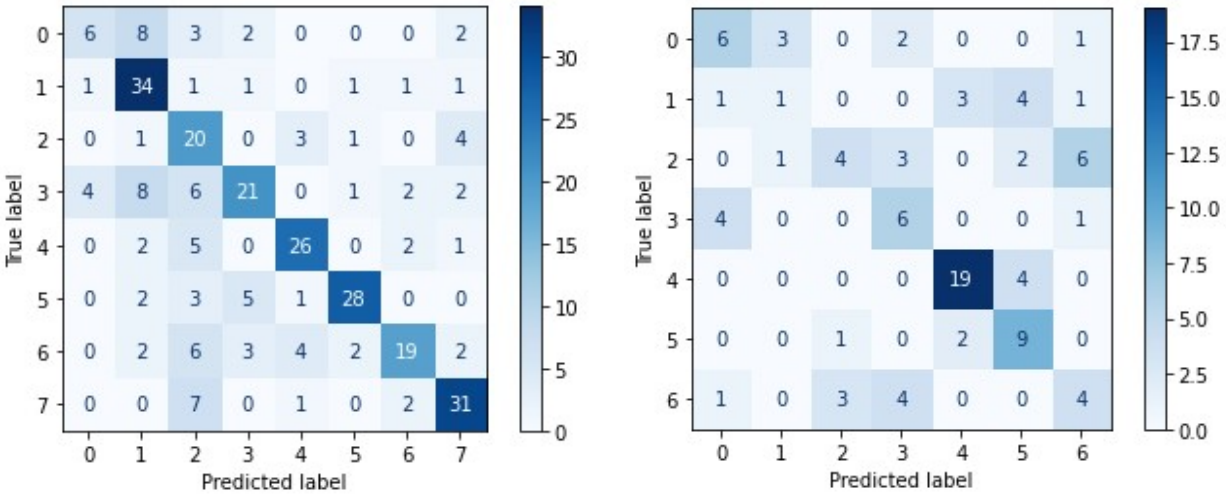

Figure 29. CNN network confusion matrices for RAVDESS (**left**) and SAVEE database (**right**).

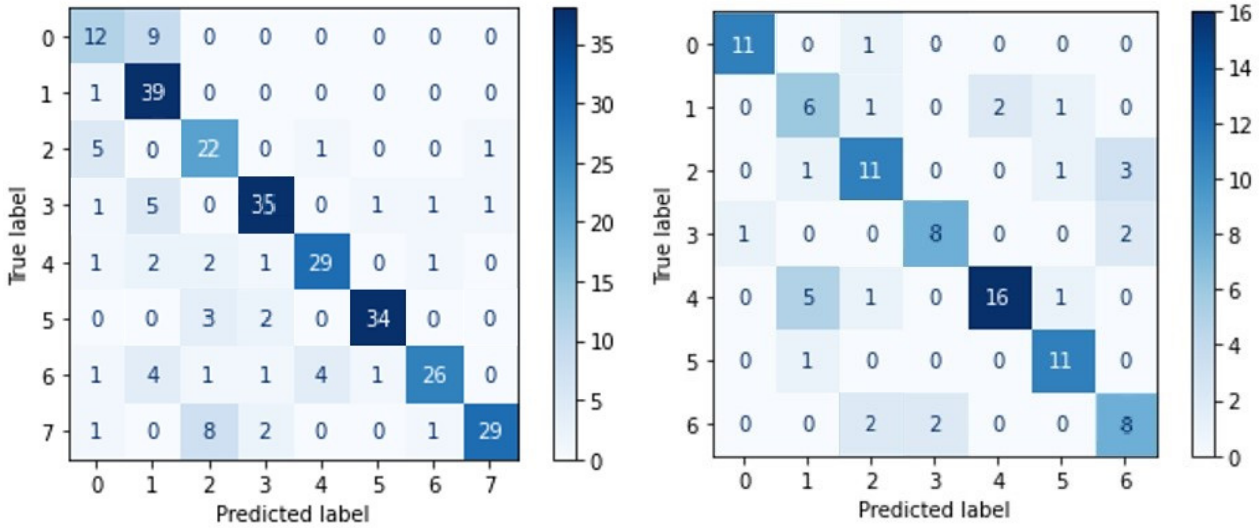

Figure 30. CNN network with attention mechanism confusion matrices for RAVDESS (**left**) and SAVEE database (**right**).

Table 1. Correspondence between the emotion labels and the emotions in the two datasets.

Label of Emotion	Emotion in SAVEE	Emotion in RAVDESS
(0)	anger	neutral
(1)	disgust	calm
(2)	fear	happy
(3)	happiness	sad
(4)	neutral	angry
(5)	sadness	fearful
(6)	surprise	disgust
(7)	-	surprise

From the confusion matrix of the DBN for the RAVDESS (Figure 25), we can conclude that the 'neutral' and 'angry' emotions of the dataset yield the lowest success rate (0%), as the 'neutral' emotion is most often confused with the emotion of 'calm' (in 9 out of 20 samples), and the emotion of 'anger' is most often confused with the emotion of 'surprise' (in 27 out of 36 samples). In contrast, the emotion of 'surprise' receives the best success rate (80.5%) compared to the rest, as it is identified in 33 out of 41 samples.

From the confusion matrix of the DBN for the SAVEE dataset (Figure 25), we can conclude that the 'anger' emotion of the dataset yields the lowest success rate (25%) by identifying only 3 of the 12 samples, as it confuses most times with the emotion of 'sadness' (in 8 out of 12 samples). In contrast, the emotion of 'surprise' receives the best success rate (66.7%) compared to the rest, as it is identified in 8 out of 12 samples.

In the confusion matrix of the DNN for the RAVDESS (Figure 26), we can see that the 'neutral' emotion yields the lowest success rate (14.3%) as it is most often confused with the 'sad' emotion (in 5 out of 21 samples) and the emotion of 'disgust' (in 5 out of 21 samples). In contrast, the emotion of 'surprise' receives the best success rate (61%) compared to the rest as it identifies 25 out of 41 samples.

Similarly, in the confusion matrix of the DNN for the SAVEE dataset (Figure 26), we can see that the emotion of 'surprise' yields the lowest success rate (33.3%), identifies in only 4 out of the 12 samples, as it is confused most often with the feeling of 'fear' (in 5 out of the 12 samples). In contrast, the 'neutral' emotion receives the best success rate (91.3%) compared to the rest as it is identified in 21 out of 23 samples.

From the confusion matrix of LSTM for the RAVDESS (Figure 27), we can conclude that the 'happy' emotion and the 'neutral' emotion of the dataset yield the lowest success rates, achieving 44.8% and 66.7%, respectively. This is because the 'happy' emotion is most often confused with the emotion of 'fearful' (in 7 out of 29 samples), and the 'neutral' emotion is sometimes confused with the 'sad' emotion (in 3 out of 21 samples). In contrast, the emotions of 'calm' and 'surprise' receive the best success rates (80% and 78%), as they are identified in 32 out of 40 and 32 out of 41 samples, respectively.

From the confusion matrix of LSTM for the SAVEE dataset (Figure 27), we can conclude that the emotion of 'happiness' yields the lowest success rate (36.4%), identified in only 4 out of 11 samples, as it is confused several times with both the emotion of 'anger' and the emotion of 'surprise' (in 3 out of 12 samples for each). In contrast, the 'neutral' emotion receives the optimal success rate (95.6%) compared to the rest, as it is identified in 22 of the 23 samples.

From the confusion matrix of the LSTM-ATN for the RAVDESS (Figure 28), we can conclude that the 'sad' emotion achieves the lowest success rate (50%), as it is identified in only 22 out of 44 samples, as it is sometimes confused with the emotion of 'disgust' (in 6 out of 44 samples). In contrast, the 'calm' emotion receives the optimal success rate (95%) compared to the rest, as it is identified in 38 out of 40 samples.

From the confusion matrix of the LSTM-ATN for the SAVEE dataset (Figure 28), we can conclude that the emotion of 'happiness' yields the lowest success rate (27.3%), identified in only 3 of the 11 samples, as it is most often confused with the emotion of 'surprise' (in

3 out of 11 samples). In contrast, the neutral emotion receives the optimal success rate (95.6%) compared to the rest, as it is identified in 22 out of 23 samples.

From the confusion matrix of the CNN for the RAVDESS (Figure 29), we can conclude that the 'neutral' emotion achieves the lowest success rate (28.6%), identified in only 6 of the 21 samples, as it is most often confused with the emotion of 'calm' (in 8 out of 21 samples). In contrast, the emotion of 'calm' receives the optimal success rate (85%) compared to the rest, as it is identified in 34 out of 40 samples.

From the confusion matrix of the CNN for the SAVEE dataset (Figure 29), we can conclude that the emotion of 'disgust' yields the lowest success rate (10%), identified in only 1 out of 10 samples, as it is confused several times with both the emotion of 'sadness' (in 4 out of 10 samples) and the 'neutral' emotion (in 3 out of 10 samples). In contrast, the 'neutral' emotion receives the optimal success rate (82.6%) compared to the rest, as it is identified in 19 out of 23 samples.

From the confusion matrix of the CNN-ATN for the RAVDESS (Figure 30), we can conclude that 'neutral' emotion yields the lowest success rate (57%), as it is often confused with the emotion of 'calm' (in 9 out of 21 samples). In contrast, the 'calm' emotion receives the optimal success rate (97%) compared to the rest, as it is identified in 39 out of 40 samples.

From the confusion matrix of the CNN-ATN for the SAVEE dataset (Figure 30), we can conclude that the emotion of 'disgust' yields the lowest success rate (60%), being identified in 6 out of 10 samples, as it is confused a few times with the 'neutral' emotion (in 2 out of 10 samples). In contrast, both the emotion of 'anger' and the emotion of 'sadness' receive the best success rate (91.6%) compared to the rest, as they are identified in 11 out of 12 samples.

In conclusion, from all the above, we observe that for the RAVDESS, our models can, on average, perceive more successfully the emotions of 'calm' and 'surprise', while they find it difficult to determine the 'neutral' emotion. This may be due to the fact that in this database, there are two basic emotions that approach the 'neutral' emotion, one from a negative point of view and the other from a positive point of view, making it difficult to determine the right emotion. Regarding the SAVEE database, we observe that our models can perceive the 'neutral' emotion more successfully, as in this database, it is unique, while they find it difficult to identify the emotions of 'happiness' and 'disgust'.

5. Discussion

In Table 2, we present the results of the accuracy of the prediction of the models we implemented for the RAVDESS and SAVEE database. Deep neural networks (DNNs) are feed-forward neural networks, and in these networks, information moves in only one direction: forward from the input nodes, through the hidden nodes, to the output nodes. Deep belief networks (DBNs), however, have non-directed connections between some layers, making the topologies of these two models different by definition. Comparing DBNs with DNNs, we notice that a DBN achieves significantly lower accuracy results for both databases. Concerning DNNs, comparing an SDNN with the two LSTM-based networks, we observe that the specialization of LSTM in the classification, processing, and making predictions on time series data resulted in a significant improvement of accuracy in the large database (RAVDESS), in contrast to the result in the small database (SAVEE), where there was no improvement (deterioration in the case of simple LSTM). Then, comparing the LSTM networks with the convolutional neural network (CNN), we notice that the CNN achieved better results in the (large) RAVDESS, while in the (small) SAVEE database, the results were significantly reduced. This is because the CNN model is designed with a large number of hidden layers (five), which burdens performance if the amount of data is small, while it improves performance for large amounts of data. The addition of the attention mechanism to the convolutional neural network (as to the LSTM) contributes significantly to the improvement of results for both databases, as the addition of this mechanism equips the neural network with the ability to focus on a subset of inputs/attributes using weights on previous hidden states, $\tilde{h_s}$, thus retaining relevant information from the input side to

assist in prediction. So, by combining the information from the input side with those in the current target state, h_t, as explained in Sections 3.4 and 3.6, better prediction rates are achieved (identified by bold numbers in Table 2).

Table 2. Prediction results of the models in the RAVDESS and SAVEE database (weighted accuracy).

Models	Input Features	SAVEE	RAVDESS
DBN	Average of 40 MFCCs	47.92	32.64
DNN	Average of 40 MFCCs	69.79	45.18
LSTM	40 MFCCs	67	70
LSTM-ATN	40 MFCCs	69.79	66
CNN	40 MFCCs	55.2	70.5
CNN-ATN	40 MFCCs	**74**	**77**

In the following, we present the comparative study of our proposed method, which is the CNN-ATN, with existing models in the literature that provide state-of-the-art results. The model we propose consists of five convolution layers, each using the batch normalization technique and the ReLU activation function. The first, second, and third convolution layers also use the dropout normalization technique to reset 50% of the connections. The result of the last convolutional layer is fed to the attention mechanism, which emphasizes the use of weights in a subset of inputs holding more input information to optimize the performance of the model prediction.

Table 3 presents the performance of the proposed method in comparison with other state-of-the-art proposals for the SAVEE database. Sivanagaraja et al. [36] propose a multiscale convolution network (MCNN) for SER using rawWav to train a DNN, which consists of three stages: (i) the signal transformation stage, (ii) the local convolution stage, and (iii) the global convolution stage. Latif et al. [37] introduce a deep belief Network (DBN) with three RBM layers using the eGeMAPS features set. Fayek et al. [38] developed a deep neural network (DNN) that recognizes emotions from a one-second frame of raw speech spectrograms. Chenchah and Lachiri et al. [39] utilize the Hidden Markov Model (HMM), categorizing the characteristics exported using Linear Frequency Cepstral Coefficients (LFCCs) and Mel-Frequency Cepstral Coefficients (MFCCs). It is obvious that our method significantly outperforms the other methods.

Table 3. Comparison with the previous works using SAVEE database.

Models	Input Features	SAVEE Test Accuracy (%)
MCNN Sivanagaraja [36]	rawWav	50.28
DBN Latif [37]	eGeMAPS	56.76
DNN Fayek, Lech and Cavedon [38]	Spectrogram	59.7
HMM Chenchah and Lachiri [39]	LFCCs/MFCCs	45/61.25
Proposed method	MFCCs	**74**

Table 4 presents the performance of the proposed method in comparison with other state-of-the-art proposals for the RAVDESS database. Rajak and Mall [40] present a convolutional neural network (CNN) model, which consists of four one-dimensional (1D) local convolutional filters and receives, as input, 12 MFCCs, which are extracted from the audio files with a 10 ms step of a 50 ms window. In addition, after the second hidden layer, a max-pooling layer is applied. Jalal et al. [7] developed a robust emotion classification method using a bidirectional long short-term memory network (BLSTM), a CNN, and capsule networks using a feature vector, which consists of the fundamental frequency (F0), 23-dimensional MFCC, and log-energy augmented by delta and delta–delta. The overall network is composed of two BLSTM layers, a 1D conv-capsule layer consisting of capsules, and a capsule-routing layer. The input layer consists of 70 nodes (the length of the feature vector), and each BLSTM hidden layer contains 256 units. The BLSTM deals with the tempo-

ral dynamics of the speech signal by effectively representing forward/backward contextual information, while the CNN, along with the dynamic routing of the capsule net, learn temporal clusters. Venkataramanan and Rajamohan et al. [41] present a 2D CNN with global average pooling, which consists of four layers of convolution with a filter size of 12 × 12 for the first layer, 7 × 7 for the second, and 3 × 3 for the other two layers and receives as input Log-Mel Spectrograms. Mohanty et al. [42] developed a convolutional neural network (CNN), which consists of 18 convolution layers and receives, as input, MFCCs. Huang and Bao et al. [43] utilizes a 2D CNN, which consists of four convolution layers, and after the second and fourth hidden layer, a max-pooling layer is applied. This model also uses 13 MFCCs as the input. Mustaqeem and Soonil Kwon et al. [8] described a deep stride CNN (DSCNN) for SER using raw spectrograms for training, which consists of seven layers of convolution, using the 2 × 2 stride to sub-sample the size of feature maps instead of using pooling layers. Issa, Demirci, and Yazici [19], in their work, present a model that includes six one-dimensional convolutional layers combined with dropout, batch normalization, and activation layers. The first layer of their CNN receives 193 × 1 number arrays as the input data. The initial layer is composed of 256 filters with a kernel size of 5 × 5 and a stride of one, followed by a batch normalization and ReLU layer. The next convolutional layer, consisting of 128 filters with the same kernel size and stride, is also using a ReLU and dropout with a rate of 0.1. Next, batch normalization is implemented, feeding its output to the max-pooling layer with a window size of eight. Next, three convolution layers with 128 filters of size 5 × 5 are located, two of which are followed by ReLU activation layers and finally by batch normalization, ReLU activation, and dropout layers with a rate of 0.2. The final convolutional layer with the same parameters is followed by the flattening layer and dropout with the rate of 0.2. The output of the flattening layer is received by a fully connected layer with eight, seven, or two units, depending on the number of predicted classes. After that, batch normalization and Softmax activation are applied.

Table 4. Comparison with the previous works using the RAVDESS.

Models	Input Features	RAVDESS-Test Accuracy (%)
CNN Rajak and Mall [40]	MFCCs	47.92
BLSTM-CNN-Capsule Network Jalal [7]	F0, MFCCs, Log spectrogram augmented by delta and delta–delta	56.2
2D CNN with Global Avg Pool Venkataramanan and Rajamohan [41]	Log-Mel spectrogram	69.79
CNN Mohanty [42]	MFCCs	67
CNN Huang and Bao [43]	MFCCs	69.79
DSCNN Mustaqeem and Soonil Kwon [8]	Raw spectrograms/clean spectrograms	68/80
1D CNN Issa, Demicri and Yazici [25]	MFCCs, chromagram and Mel spectrogram	71.61
Proposed method	MFCCs	**77**

6. Conclusions

The broader literature on speech emotion recognition (SER) systems addresses many challenges in improving emotion recognition accuracy but also seeks to reduce the computational complexity of models. In recent years, the attention mechanism has begun to be used to improve neural machine translation (NMT) by selectively focusing on parts of the source sentence during translation. However, not many studies have been presented that implement this technique. For this reason, in our study, we propose a model of a convolutional neural network with the addition of an attention mechanism, which significantly outperforms other attempts of ours and the state-of-the-art methods, as shown in Tables 3 and 4, giving the best prediction results: 74% for the SAVEE database, and 77% for

the RAVDESS. This is due to the following reasons. First, for our model, we decided to export Mel Frequency Cepstral Coefficients (MFCCs) from the audio data we used for input, as they provide rich feature content from the data in a consistent manner. Another reason is that these coefficients use the Mel scale, which is based on how humans perceive different signal frequencies. Finally, the addition of the attention mechanism to the simple CNN enabled our model to focus on the parts of the audio input that contain more emotional information than parts where the speaker does not say something or words that are not emotionally intense, helping to further improve the ability to predict emotion.

There are many directions that future work will focus on. First, a main direction of our future work concerns testing our model on other databases and trying different modern deep learning architectures, such as Transformers [44]. Furthermore, additional and more complex attention mechanisms will also be examined, and their impact will also be assessed. Finally, another direction for future work concerns the design of bigger-scale experimental evaluations on additional databases too.

Author Contributions: Conceptualization, I.P. and K.M.; methodology, I.P.; software, K.M. and I.P.; validation, K.M.; investigation, I.P.; data curation, K.M.; writing—original draft preparation, I.P. and K.M.; writing—review and editing, I.H. supervision, I.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data available in publicly accessible repositories (mentioned in the paper).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, X.; Zhang, Y.; Yu, S.; Liu, X.; Yuan, Y.; Wang, F. E-learning recommendation framework based on deep learning. In Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; pp. 455–460. [CrossRef]
2. Gligorijevic, J.; Gligorijevic, D.; Pavlovski, M.; Milkovits, E.; Glass, L.; Grier, K.; Vankireddy, P.; Obradovic, Z. Optimizing clinical trials recruitment via deep learning. *J. Am. Med. Inform. Assoc.* **2019**, *26*, 1195–1202. [CrossRef]
3. Davatzikos, C.; Ruparel, K.; Fan, Y.; Shen, D.; Acharyya, M.; Loughead, J.; Gur, R.; Langleben, D.D. Classifying spatial patterns of brain activity with machine learning methods: Application to lie detection. *NeuroImage* **2005**, *28*, 663–668. [CrossRef]
4. Justesen, N.; Bontrager, P.; Togelius, J.; Risi, S. Deep Learning for Video Game Playing. *IEEE Trans. Games* **2020**, *12*, 1–20. [CrossRef]
5. Lavrentyeva, G.; Novoselov, S.; Malykh, E.; Kozlov, A.; Kudashev, O.; Shchemelinin, V. Audio Replay Attack Detection with Deep Learning Frameworks. In Proceedings of the Interspeech 2017, Stockholm, Sweden, 20–24 August 2017; pp. 82–86. [CrossRef]
6. Ajuzieogu, U. *The Role of AI in Modern Computing and Education*; Lulu Publisher: Morrisville, NC, USA, 2019; ISBN 978-0-359-72121-4.
7. Jalal, M.A.; Loweimi, E.; Moore, R.K.; Hain, T. Learning Temporal Clusters Using Capsule Routing for Speech Emo-tion Recognition. In Proceedings of the Interspeech 2019, Graz, Austria, 15–19 September 2019; pp. 1701–1705. [CrossRef]
8. Mustaqeem; Kwon, S. A CNN-Assisted Enhanced Audio Signal Processing for Speech Emotion Recognition. *Sensors* **2020**, *20*, 183. [CrossRef] [PubMed]
9. Singh, Y.B.; Goel, S. A systematic literature review of speech emotion recognition approaches. *Neurocomputing* **2022**, *492*, 245–263. [CrossRef]
10. Wani, T.M.; Gunawan, T.S.; Qadri, S.A.A.; Kartiwi, M.; Ambikairajah, E. A Comprehensive Review of Speech Emotion Recognition Systems. *IEEE Access* **2021**, *9*, 47795–47814. [CrossRef]
11. Yadav, S.P.; Zaidi, S.; Mishra, A.; Yadav, V. Survey on Machine Learning in Speech Emotion Recognition and Vision Systems Using a Recurrent Neural Network (RNN). *Arch. Comput. Methods Eng.* **2022**, *29*, 1753–1770. [CrossRef]
12. Lieskovská, E.; Jakubec, M.; Jarina, R.; Chmulík, M. A review on speech emotion recognition using deep learning and attention mechanism. *Electronics* **2021**, *10*, 1163. [CrossRef]
13. Khalil, R.A.; Jones, E.; Babar, M.I.; Jan, T.; Zafar, M.H.; Alhussain, T. Speech Emotion Recognition Using Deep Learning Techniques: A Review. *IEEE Access* **2019**, *7*, 117327–117345. [CrossRef]
14. Abbaschian, B.J.; Sierra-Sosa, D.; Elmaghraby, A. Deep Learning Techniques for Speech Emotion Recognition, from Databases to Models. *Sensors* **2021**, *21*, 1249. [CrossRef]
15. de Lope, J.; and Graña, M. An ongoing review of speech emotion recognition. *Neurocomputing* **2023**, *528*, 1–11. [CrossRef]
16. Wen, G.; Li, H.; Huang, J.; Li, D.; Xun, E. Random Deep Belief Networks for Recognizing Emotions from Speech Signals. *Comput. Intell. Neurosci.* **2017**, *2017*, 1945630. [CrossRef] [PubMed]

17. Badshah, A.M.; Ahmad, J.; Rahim, N.; Baik, S.W. Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network. In Proceedings of the 2017 International Conference on Platform Technology and Service (PlatCon), Busan, Republic of Korea, 13–15 February 2017; pp. 1–5. [CrossRef]
18. Zhao, J.; Mao, X.; Chen, L. Speech emotion recognition using deep 1D & 2D CNN LSTM networks. *Biomed. Signal Process. Control* **2019**, *47*, 312–323. [CrossRef]
19. Lee, C.; Song, K.Y.; Jeong, J.; Choi, W.Y. Convolutional Attention Networks for Multimodal Emotion Recognition from Speech and Text Data. *arXiv* **2019**, arXiv:1805.06606.
20. Tang, D.; Zeng, J.; Li, M. An End-to-End Deep Learning Framework for Speech Emotion Recognition of Atypical Individuals. In Proceedings of the Interspeech 2018, Hyderabad, India, 2–6 September 2018; pp. 162–166. [CrossRef]
21. Chen, M.; He, X.; Yang, J.; Zhang, H. 3-D Convolutional Recurrent Neural Networks With Attention Model for Speech Emotion Recognition. *IEEE Signal Process. Lett.* **2018**, *25*, 1440–1444. [CrossRef]
22. Li, P.; Song, Y.; Mcloughlin, I.; Guo, W.; Dai, L. An Attention Pooling Based Representation Learning Method for Speech Emotion Recognition. In Proceedings of the INTERSPEECH 2018, Hyderabad, India, 2–6 September 2018.
23. Jiang, W.; Wang, Z.; Jin, J.S.; Han, X.; Li, C. Speech Emotion Recognition with Heterogeneous Feature Unification of Deep Neural Network. *Sensors* **2019**, *19*, 2730. [CrossRef] [PubMed]
24. Huang, K.; Wu, C.; Hong, Q.; Su, M.; Chen, Y. Speech Emotion Recognition Using Deep Neural Network Considering Verbal and Nonverbal Speech Sounds. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 5866–5870. [CrossRef]
25. Issa, D.; Demirci, M.; Yazici, A. Speech emotion recognition with deep convolutional neural networks. *Biomed. Signal Process. Control* **2020**, *59*, 101894. [CrossRef]
26. Makhmudov, F.; Kutlimuratov, A.; Akhmedov, F.; Abdallah, M.S.; Cho, Y.-I. Modeling Speech Emotion Recognition via Attention-Oriented Parallel CNN Encoders. *Electronics* **2022**, *11*, 4047. [CrossRef]
27. Liu, Z.-T.; Han, M.-T.; Wu, B.-H.; Rehman, A. Speech emotion recognition based on convolutional neural network with attention-based bidirectional long short-term memory network and multi-task learning. *Appl. Acoust.* **2023**, *202*, 109178. [CrossRef]
28. Saumard, M. Enhancing Speech Emotions Recognition Using Multivariate Functional Data Analysis. *Big Data Cogn. Comput.* **2023**, *7*, 146. [CrossRef]
29. Wang, Y.; Fu, H.; Tao, H.; Yang, J.; Ge, H.; Xie, Y. Convolutional Auto-Encoder and Adversarial Domain Adaptation for Cross-Corpus Speech Emotion Recognition. *IEICE Trans. Inf. Syst.* **2022**, *105*, 1803–1806. [CrossRef]
30. Xie, Y.; Liang, R.; Liang, Z.; Zhao, L. Attention-Based Dense LSTM for Speech Emotion Recognition. *IEICE Trans. Inf. Syst.* **2019**, *102*, 1426–1429. [CrossRef]
31. Abdul, Z.K.; Al-Talabani, A.K. Mel Frequency Cepstral Coefficient and its Applications: A Review. *IEEE Access* **2022**, *10*, 122136–122158. [CrossRef]
32. Luong, M.T.; Pham, H.; Manning, C.D. Effective Approaches to Attention-based Neural Machine Translation. *arXiv* **2015**, arXiv:1508.04025.
33. Jackson, P.; Haq, S. Surrey Audio-Visual Expressed Emotion (SAVEE) Database. 2011. Available online: http://kahlan.eps.surrey.ac.uk/savee/Database.html (accessed on 1 September 2022).
34. Livingstone, S.R.; Russo, F.A. The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PLoS ONE* **2018**, *13*, e0196391. [CrossRef]
35. Dubey, A.K.; Jain, V. Comparative study of convolution neural network's relu and leaky-relu activation functions. In *Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC 2018*; Springer: Singapore, 2019; pp. 873–880.
36. Sivanagaraja, T.; Ho, M.K.; Khong, A.W.H.; Wang, Y. End-to-end speech emotion recognition using multi-scale convolution networks. In Proceedings of the 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Kuala Lumpur, Malaysia, 12–15 December 2017; pp. 189–192. [CrossRef]
37. Latif, S.; Rana, R.; Younis, S.; Qadir, J.; Epps, J. Transfer Learning for Improving Speech Emotion Classification Accuracy. *arXiv* **2018**, arXiv:1801.06353.
38. Fayek, H.M.; Lech, M.; Cavedon, L. Towards real-time Speech Emotion Recognition using deep neural networks. In Proceedings of the 2015 9th International Conference on Signal Processing and Communication Systems (ICSPCS), Cairns, QLD, Australia, 14–16 December 2015; pp. 1–5. [CrossRef]
39. Chenchah, F.; Lachiri, Z. Acoustic Emotion Recognition Using Linear and Nonlinear Cepstral Coefficients. *Int. J. Adv. Comput. Sci. Appl. (IJACSA)* **2015**, *6*, 135–138. [CrossRef]
40. Rajak, R.; Mall, R. Emotion recognition from audio, dimensional and discrete categorization using CNNs. In Proceedings of the TENCON 2019—2019 IEEE Region 10 Conference (TENCON), Kochi, India, 17–20 October 2019; pp. 301–305. [CrossRef]
41. Venkataramanan, K.; Rajamohan, H.R. Emotion Recognition from Speech. *arXiv* **2019**, arXiv:1912.10458.
42. Mohanty, H.; Budhvant, S.; Gawde, P.; Shelke, M. Implementation of Mood Detection through Voice Analysis using Librosa and CNN. *Int. Res. J. Eng. Technol. (IRJET)* **2020**, *7*, 5876–5879.

43. Huang, A.; Bao, P. Human Vocal Sentiment Analysis. *arXiv* **2019**, arXiv:1905.08632.
44. Wagner, J.; Triantafyllopoulos, A.; Wierstorf, H.; Schmitt, M.; Burkhardt, F.; Eyben, F.; Schuller, B.W. Dawn of the Transformer Era in Speech Emotion Recognition: Closing the Valence Gap. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 10745–10759. [CrossRef] [PubMed]