Subject: Deep Learning (DPL302m)

PROJECT

SPEECH EMOTION RECOGNITION MODEL REPORT

FPT UNIVERSITY

TOPIC:

Development and Evaluation of a Speech Recognition Model

LECTURER

Ho Le Minh Toan

MEMBERS

Tran Hoang Tuan Hung - SE193182

Phan Quoc Anh - SE193726

Huynh Han Dong - SE193065

Tran Quoc Huan - SE193642

Nguyen Van Anh Duy - SE181823

Nguyen Truong Phuc Thinh - SE190168

Nguyen Huu Gia Bao - SE193507

CLASS

AI1914

FPT UNIVERSITY

TABLE OF CONTENT

Abstract

This paper presents the design and whole structure of the *Speech Emotion Recognition Model,* which was trained on Multiple Datasets of Voice and Speech. Also, it is one of the most important projects for us to prove our knowledge and the lessons we have learned in class *DPL302m*.

So, Why Speech Emotion Recognition? Moreover, what use is it in the real world? To answer that, we first need to identify and understand the problem. If you notice, especially in these close years (2024, 2025), you must have heard of AI or more specifically… ChatGPT proves to be quite helpful and great for improving your productivity, like almost anything that can be applied to various tasks. However, there is one problem: some of our life branches are pretty difficult for them to reach, such as Customer Service, Mental Health Monitoring, E-Learning, Social Robotics, and Therapy. All those fields require emotions and understanding to be done, but for a machine, that is an almost impossible task to do because AI, or "machine", does not feel, or to be a bit harsher, they do not have emotion or soul. They just understand the word, code, number, and "order" that we give out for them. So this model we have been working on can make significant changes because we filter the voice that has much parasitic noise into a clean one. Based on that, we transfer all of it to a log-mel spectrogram.

The model structure we are using is a 3D CNN (Convolutional Neural Network), and the main goal we aim for is accuracy in the range of 80-90+ percent.

---

1. Introduction

1.1 Why does Speech Emotion Recognition matter?

Automatic emotion detection empowers AI systems to become more empathetic and human-aware. In a world where digital interactions are becoming the norm, the ability for machines to understand how something is said opens the door to more natural, emotionally intelligent communication. Imagine a virtual assistant that senses frustration and responds with patience, or a mental health application that detects emotional distress through subtle changes in tone. These scenarios are no longer science fiction.

Speech Emotion Recognition (SER) serves as the bridge between functional AI and emotionally aware AI. By enabling machines to perceive and respond to human emotions, we can expand the role of AI into traditionally human-centric fields such as therapy, education, entertainment, and even conflict resolution. This creates a deeper, more meaningful synergy between humans and machines.

1.2 Motivation

Recognizing emotions in speech is a challenging task due to the complexity and variability of human expression. Different people express emotions differently; the same sentence can convey joy, sarcasm, or sadness, depending on factors such as tone, pitch, tempo, and energy. This subtlety makes speech emotion recognition a fascinating challenge.

Our motivation lies in this challenge—the opportunity to teach machines how to decode the emotional layers within speech. Successfully achieving this would bring us closer to building AI that not only responds but also understands. As the saying goes, "Nothing in the world is impossible. It only becomes impossible when you believe it is." That is the spirit driving this project forward.

1.3 Inspiration

Our inspiration stems from pioneering neural network architectures, such as LeNet-5 and AlexNet. These foundational models have paved the way for numerous innovations in deep learning, providing us with the structural blueprints and architectural insights necessary to design our own models. Their proven success gave us a solid starting point, reducing the trial-and-error phase and allowing us to focus more on experimenting with datasets and fine-tuning our system.

Beyond their technical influence, these models also served as a source of confidence and motivation. They reminded us that innovation often begins by standing on the shoulders of giants. Their legacy not only equipped us with practical tools but also inspired belief in our ability to contribute meaningfully to the future of Artificial Intelligence. We see this project as a stepping stone toward a broader vision—one where AI is more human-aware, emotionally intelligent, and capable of transforming how we live and interact.

---

2. Methodology

2.1 Datasets

To build a comprehensive speech emotion recognition model, we combined four widely used emotional speech datasets: RAVDESS, TESS, SAVEE, and EmoDB. There are 5147 audio samples in WAV format, encompassing seven emotion categories: neutral, happy, sad, angry, fearful, surprised, and disgusted.

2.1.1 RAVDESS (Livingstone & Russo, 2018)

The portion of the RAVDESS contains 1440 files: 60 trials per actor x 24 actors = 1440. The RAVDESS contains 24 professional actors (12 female, 12 male), vocalizing two lexically matched statements in a neutral North American accent. Speech emotions include calm, happy, sad, angry, fearful, surprised, and disgusted expressions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.

2.1.2 TESS (Sahar & Dupuis, 2010)

TESS includes 2800 audio files featuring two actresses (aged 26 and 64) speaking 200 target words in the carrier phrase "Say the word _". The dataset is organised such that each of the two female actors and their emotions are contained within their own folder. Moreover, within that, all 200 target words' audio files can be found.

2.1.3 SAVEE (Haq & Jackson, 2009)

The SAVEE database was recorded from four native English male speakers, postgraduate students and researchers at the University of Surrey, aged from 27 to 31 years. Each speaker recorded 15 phonetically-balanced TIMIT sentences per emotion: 3 common, two emotion-specific, and 10 generic sentences that were different for each emotion and phonetically-balanced. The three common and $2 \times 6 = 12$ emotion-specific sentences were recorded as neutral to give 30 neutral sentences. There are a total of 480 audio files.

2.1.4 EmoDB (Burkhardt et al., 2005)

The EmoDB database is a German database of emotions. Ten professional speakers (five males and five females) participated in data recording. The database contains a total of 535 utterances. This dataset comprises seven emotions: anger, boredom, anxiety, happiness, sadness, disgust, and neutral. The *boredom* class was excluded to maintain consistency with the other datasets' emotion categories. So, the total number of samples we used is 454 (after removing the *boredom* category).

2.2 Audio processing and Feature extraction

The foundation of any speech emotion recognition model lies in the quality and representation of its audio input. Our preprocessing pipeline begins with standardization and normalization of the raw audio waveforms to ensure uniform amplitude and sampling rates across different recordings. This step mitigates the impact of variations in recording devices and ensures consistency for downstream processing.

2.2.1 Feature Extraction: Mel-Spectrogram

To convert Raw Audio Signals into a format suitable for deep learning models, we transformed each waveform into a Mel-Spectrogram using the Librosa library. This transformation captures both frequency and temporal characteristics of speech, which are critical for emotion recognition. Specifically, we extracted 3D features in the form of a fixed-sized array with dimensions:

- *Timesteps*: the number of chunks split from an audio
- *Number of n_mels*: the number of Mel-frequency bins (typically 128).
- *Channel*: number of channels.
- *Frame_per_step:* the number of frames in one specific chunk

The Frame_per_step is resized by padding and truncating to fix the size (60 frames per chunk).

The power spectrogram is converted to decibels (dB) scale, which is logarithmic, better to reflect human auditory perception of loudness and pitch.

Lastly, we normalize decibel values between 0 and 1 by the Min-Max scaling technique. This step helps to ensure all the input values stay in the same scale, which encourages the model to learn more stably and faster.

2.2.2 Label Mapping

For supervised learning, each extracted mel-spectrogram was labeled based on the emotion class derived from the audio file's metadata. Classes included emotions such as neutral, happy, sad, angry, fearful, disgusted, and surprised. These labels were encoded numerically for model input, forming the feature-label pairs used during training.

2.2.3 Load and preprocess data

We load and preprocess files at the same time, and the label is encoded by using one-hot encoding

Next, we apply tf.data.Dataset from TensorFlow to split the batch on the dataset.

2.2.4 Data Augmentation Techniques

To improve model generalization and simulate real-world audio variability, we augmented the dataset using the audiomentations library. Key techniques included:

- *AddGaussianNoise:* Injected random noise to simulate low-quality or background-dense recordings.
- *PitchShift:* Modified pitch slightly to account for natural variations in speaker tone.
- *TimeStretch:* Stretched or compressed speech tempo without changing pitch, mimicking fast or slow speaking styles.

- *Shift:* Introduced random temporal shifts to simulate audio misalignment or timing offsets.

These transformations were applied probabilistically to ensure diversity in the augmented training data. This strategy effectively prevented overfitting and made the model more robust to different recording conditions and speaker variations.

2.2.4 Create Datasets

After combining all four datasets, we realized that we needed to convert the emotion labels into indices to ensure smooth training. Furthermore, we load spectrograms into a built-in function (py_function in Tensor-Flow), with which we can ensure the main feature does not break during the transformation. Then we set it into a shape as a sequence of numbers, such as (depth, height, width, channel). For that part, we one-hot encoded the label to match its data type and inserted it using the pipeline technique.

2.3 Model Training

The model training process is a critical phase in building robust Speech Emotion Recognition models. Our model architecture is based on a Convolutional Neural Network (CNN), which has proven effective in extracting spatial and temporal features from mel-spectrogram representations of audio.

2.3.1 Model Architecture

The architecture was designed to capture hierarchical patterns in spectrogram data. It consists of:

- *Input Layer*: accepts a fixed-size 3D mel-spectrogram array (e.g., shape: 3, 128, 60, 1).
- *Convolutional Blocks*: a sequence of Conv3D layers with increasing filter sizes (e.g., 64, 128, 216, 512), each followed by:
  - ❖ Batch Normalization to stabilize training.
  - ❖ ReLU Activation to introduce non-linearity.
  - ❖ Pooling to reduce dimensionality.
- *Global Average Pooling Layer:* fuses all local features and prepares input for global learning at the dense layers.
- *Fully Connected (Dense) Layers:* learn high-level emotional features.
- *Dropout Layers:* randomly deactivate neurons to prevent overfitting.
- *Output Layer:* a softmax classifier that predicts the probability distribution over emotion classes.

This structure enables the model to effectively learn local frequency-time patterns as well as global emotion characteristics in the input signal.

2.3.2 Training Strategy

The dataset was divided into training, validation, and testing subsets using a 70:15:15 split to ensure reliable evaluation. The model was trained using the Focal Loss, suitable for multi-class classification tasks. The Adam optimizer was employed for its adaptive learning rate capabilities and fast convergence. To further improve training stability and convergence, we integrated several *Keras callbacks*:

- *EarlyStopping:* monitors validation loss and stops training when no improvement is observed for a predefined number of epochs (patience), preventing unnecessary overfitting.
- *ModelCheckpoint:* saves the best-performing model based on validation accuracy.
- *ReduceLROnPlateau:* automatically reduces the learning rate when the validation loss plateaus, allowing finer tuning as the model approaches convergence.

2.3.3 Training with Augmented Data

In addition to the training set being used as the augmented data from the original training set, the training set was enriched with audio data augmented through techniques like Gaussian noise addition, pitch shifting, and time stretching. These samples were processed through the same mel-spectrogram pipeline, thereby increasing the model's robustness to real-world audio variations such as background noise or speaker diversity.

Ensuring the augmented data on the training data avoids getting all of it augmented before the split into train/val/test sets will help prevent the model from leaking data that leads to a false outcome.

The training was conducted over multiple epochs (typically 50–150), and batch sizes were tuned (e.g., 8–32) based on available GPU, CPU memory, and convergence behavior. After each epoch, validation metrics were monitored to track the model's ability to generalize.

---

3. Hyperparameter Tuning

Hyperparameter Tuning plays a pivotal role in improving the performance and generalization of deep learning models. In this project, we systematically explored and optimized several critical hyperparameters that influence model training dynamics and final accuracy.

3.1 Architectural Tuning

During our research, the diversity of Neural Network architecture was complicated, so we came up with various approaches to balance between the model complexity and the ability to give precise predictions. In order to achieve the accuracy that we wish for, we have considered our willingness to choose in the architectural design.

It also includes an acceptable and reasonable training time to support the iterative experimentation, maintaining the memory space to accommodate hardware limitations, to regularizing strategies to avoid the risk of early overfitting, especially in scenarios with a small amount of datasets. Furthermore, we focus all our priorities on model design, which allows us to have a stable convergence and generalizable learning, pushing up the overall effectiveness of training in the model. This includes:

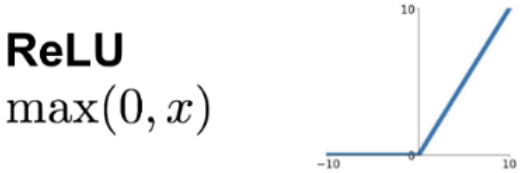
- *Number of Convolutional Layers*: tested models with 2 to 4 convolutional blocks.
- *Filter Sizes*: explored small (3×3) and larger (5×5) kernel sizes to capture different levels of frequency-temporal patterns, (1×1) increasing the depth of the feature map.
- *Number of Filters per Layer*: tuned the filter count (e.g., 32, 64, 128, 256, 512) to control the model's capacity to learn features at different depths.
- *Testing some hybrid architectures*: 3D-CNN combined with Transformer encoders and 3D-CNN combined with LSTM, aiming to leverage both local spatial-temporal features and global contextual dependencies for improved representation learning.

These experiments were guided by validation accuracy, training loss curves, and overfitting indicators.

3.2 Activation Functions

The non-linear transformation applied after each convolutional layer significantly impacts the model's learning ability. We conducted a comparison of these transformations:

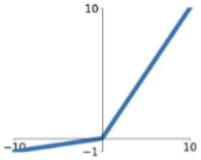
- *ReLU (Rectified Linear Unit)*: fast convergence and widely used in CNNs.



- *LeakyReLU*: addresses the dying ReLU problem by allowing a small gradient when the unit is inactive.

Leaky ReLU

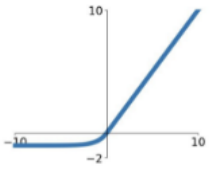
$$\max(0.1x, x)$$



- *ELU (Exponential Linear Unit)*: explored for its smooth gradient properties.

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Using each of the aforementioned activation functions, we discovered that ReLU might work well for our model.

3.3 Optimization Algorithms

The optimization algorithm plays an important role in training deep learning models. It updates new weights and biases based on computing the gradient descent of the loss function, which is known as backpropagation.

We are also using some techniques that optimize algorithms, including:

- *Adam*: an adaptive learning rate optimizer, demonstrated strong convergence behavior and robustness to noisy gradients.
- *RMSProp*: while generally suitable for models with recurrent structures, it showed comparable or slightly lower performance than Adam in our experiments.
- *SGD with Momentum*: slower but tested for comparative baselines.

Based on overall training stability, speed of convergence, and performance on augmented data, Adam was ultimately selected for its effectiveness in handling sparse updates and noisy gradients, making it well-suited for our training regime.

3.4 Learning Rate Schedules

Learning rate is one of the most sensitive hyperparameters in deep learning. It has a lot in deep learning optimization and manages how quickly the models update their weights and bias in response to the estimated error, and even the smallest changes can make the difference between fast convergence and divergence. After our experiments, we explored multiple ways to manipulate the learning rate, improving stability. Here are these strategies we got :

- *Fixed Learning Rates*: tried values from 1e-3 to 1e-5 for coarse tuning.
- *ReduceLROnPlateau*: automatically decreases the learning rate when validation loss stagnates, enabling finer updates as training progresses.
- *Warm Restarts and Step Decay (experimented but not adopted)*: provided limited gains in this application.

The dynamic scheduler helped prevent overshooting and promoted smoother convergence.

3.5 Regularization Techniques

To combat overfitting, especially with a limited dataset, the model should be reliant on these specific datasets instead of learning more generalizable, emotional features. So these are the solutions we came up with to prevent that from happening:

- *Dropout Layers*: applied dropout rates of 0.3–0.5 between dense layers to randomly deactivate neurons during training.
- *Batch Normalization*: positioned after convolutional layers to normalize intermediate outputs and accelerate training.

$$\mu = \frac{1}{m}\sum_i z^{(i)} \qquad z^{(i)}_{\text{norm}} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2 \qquad \tilde{z}^{(i)} = \gamma z^{(i)}_{\text{norm}} + \beta$$

Formula of Batch Normalizations

- *EarlyStopping*: monitored validation loss and terminated training once improvements plateaued.

These techniques significantly improved model stability and reduced variance across training runs. In total, it could be a major point or even a slight pattern for us to evolve it and turn it into an excellent, smooth model.

3.6 Loss Function:

- *Cross Entropy Loss*: At first, we applied Cross Entropy Loss and observed that certain emotions—such as *happy*, *sad*, *disgust*, and *fear*—were consistently harder to classify than others, resulting in uneven performance across classes.
- *Focal Loss*: To address this, we implemented a customized Focal Loss function. By fine-tuning class-specific weights, we adapted the loss to reflect the difficulty of each emotion better. This allowed the model to focus more on hard-to-classify examples and helped reduce the overall loss.

Focal Loss formula:

$$\mathrm{FL}(p_\mathrm{t}) = -\alpha_\mathrm{t}(1-p_\mathrm{t})^\gamma \log(p_\mathrm{t}).$$

Class weights were computed using scikit-learn's compute_class_weight function to reflect the distribution of emotion labels in the training set. These weights were normalized to form the alpha parameter, ensuring proportional emphasis on underrepresented or challenging classes. Our implementation incorporates both the focusing parameter γ (gamma) and the class-specific alpha, allowing the model to dynamically reduce the contribution of well-classified examples while giving greater weight to difficult ones.

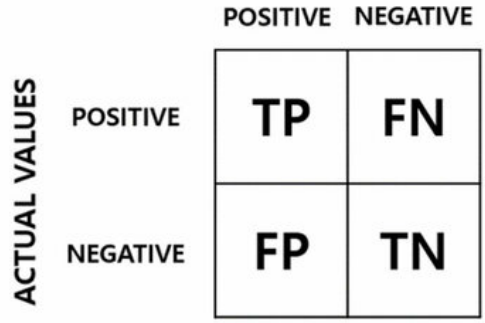
---

4. Model Evaluation

Model Evaluation is essential to assess the effectiveness, robustness, and generalization ability of the trained Speech Emotion Recognition Model. Our evaluation framework employed both quantitative metrics and qualitative analysis to understand the model's performance across different emotion classes.

4.1 Evaluation Metrics

We used a variety of standard classification metrics to measure performance:



$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- *Accuracy:* the proportion of correctly predicted samples out of the total samples. While useful for overall trends, it can be misleading in imbalanced datasets.
- *Precision:* the ratio of true positives to the sum of true and false positives. Precision helps evaluate how reliably the model is when predicting a specific emotion class.
- *Recall (Sensitivity):* the ratio of true positives to the total actual positives. High recall indicates that the model successfully detects most samples of a particular emotion.
- *F1-Score:* the harmonic mean of precision and recall, providing a balanced measure, especially in the presence of class imbalance.

These metrics were computed using scikit-learn's classification_report and provided per-class and average results (macro and weighted).

4.2 Confusion Matrix Analysis

To evaluate the classification performance beyond overall accuracy, a confusion matrix was generated to visualize the distribution of correct and incorrect predictions across emotion classes. This matrix provides a detailed breakdown of how the model performs across each emotion category by comparing labels (rows) with the model-predicted labels (columns). It serves as a powerful diagnostic tool to reveal class-specific strengths and weaknesses. Each row of the matrix represents the actual class, while each column shows the predicted class. This analysis revealed patterns such as:

- High confusion between acoustically similar emotions like fearful and angry.
- Good discrimination of distinct classes, like neutral and happy.

Insights from the confusion matrix were used to refine the preprocessing pipeline and guide future augmentation or class-weighting strategies.

4.3 Training Curves Visualization

Throughout the training process, we track both training and validation accuracy and loss curves to diagnose learning behavior. Initially, validation accuracy plateaued while training accuracy continued to climb, which is a classic sign of reaching overfitting. In response, we adjusted dropout rates and reduced the learning rate, helping it to get smoother and more convergent curves. Below are several techniques that we see that can potentially affect the model:

- *A steady convergence of both curves indicated effective learning.*
- *Divergence between training and validation metrics flagged potential overfitting.*
- *Sharp fluctuations in validation metrics suggested sensitivity to learning rate or batch size.*

These plots were crucial for identifying optimal stopping points and evaluating regularization effectiveness.

4.4 Error Analysis

We conducted manual inspection of misclassified samples, especially those consistently mispredicted across different model versions.

Most errors occurred in cases with:

- Overlapping prosodic features (e.g., sad, happy, and disgust).
- Low signal-to-noise ratio.
- Very short audio segments.

By analyzing these edge cases, we identified potential directions for improvement, such as:

- One additional layer (Block 4), which makes the network bigger and more complex to learn harder parameters and features such as: sad, happy, and disgust.
- Collected more labeled data (EmoDB), which is an increasing dataset for underrepresented emotions.
- Including temporal attention mechanisms.
- Incorporating speech rate or prosodic rhythm as additional features.

5. Model

After many approaches back in the past and several failures of our seniors, we're choosing this model and this design because of its performance and ease of handling. It's robust enough for complex audio patterns and simple enough to train without needing massive computational power. This is the full structure of the 3D CNN model that we have been building and working on:

First, we standardize and label all audio files by emotion, then balance the dataset using oversampling, so the model does not bias toward other emotions. Each emotion turns into a 3D tensor and a melspectrogram, and each clip is sliced into chunks of nearly 2 seconds. For every frame, we extract a melspectrogram, which is a matrix of rows and columns. Each of these spectrograms becomes a 2D image. Then we stack these spectrogram images into a 3D volume (ex, [3, 128, 60, 1] as the {time steps, mel bins, frames per step, channel}). This is the final input shape that we have in the input layer for our processing part.

- **Block 1:** "Low-level Feature Extraction": This block consists of 4 layers in total. The first layer is the Conv3D (64, 1×5×5): slides a 1×5×5 filter over each frame (frequency × time), operating independently on each temporal slice. This preserves the temporal structure across frames while learning low-level spectro-temporal patterns within each frame. The second layer is the Batch Normalization, which normalizes activations to speed up and stabilize training. Third is the MaxPooling 3D (1×2×2), which halves the frequency and time dimensions - removes small shifts and reduces data size. The final layer is the Dropout (20%) to prevent overfitting early in the network.

- **Block 2:** "Mid-level pattern learning": There are also four layers contained in this block, the first layer is the Conv3D (128, 1×3×3). Now with the 128 filters, it can learn more complex mid-level features, for example, like rising pitch contours, formant transitions. Slide a 1×3×3 over each frame, shifting it down so that it operates much more smoothly. The second layer is Batch Normalization, whose task is the same as the first block. It also applies to the third and fourth layers, which are MaxPooling3D (1×2×2) and the Dropout (30%).

- **Block 3:** "High-Level Abstraction": also a 4-layer block, the first layer is Conv3D (256, 1×3×3): With 256 filters, it can now capture sophisticated spatiotemporal features, such as timbral shifts, rhythm patterns. The second layer is the same old Batch Normalization. The third layer is the MaxPooling3D (1×2×2) and the fourth layer is the Dropout (30%) maybe it can be a repeating function but this technique

help out a lot with our model performance and along side with that we can show you how good or how our model performance working on the environment that not to technical or advance.

- **Block 4:** "Advanced Pattern Learning": Contains only four layers, and the most important layer in this block is the Conv3D (512, 1×3×3): Projects each voxel's channel vector into a higher 512-dimensional space. Deepens the abstraction, learning high-level emotion-related features. The second layer is the Batch Normalization, which speeds up learning, reduces the risks of vanishing gradient descent, and provides regularization. Next is the MaxPooling3D (1×2×2) layer, which keeps the time axis intact while shrinking the spatial footprint, allowing the network to keep track of temporal context more accurately. The last layer is the Dropout (30%). This block increases the depth (number of channels) dramatically to give the model more "feature channels" to work with.

- **Block 5:** "Bottleneck–Feature Expansion": Designated as the Bottleneck Block, serves to significantly expand the representational capacity of the model while preserving spatial and temporal dimensions. It contains only two layers, the first layer is the Conv3D (512, 1×1×1): This 1×1×1 convolution operates independently at each spatial-temporal voxel, projecting the existing channel vector into a higher-dimensional 512-space. This operation increases the feature depth dramatically without altering the shape of the input tensor, effectively acting as a dimensionality expansion layer. The second layer is the BatchNorm, which is immediately after the convolution to stabilize activations, improve convergence, and prevent internal covariate shift. The special part about this model is that it does not include the Dropout Layer, because it focuses more on the function and elevation. The feature space before the global aggregation ensures that the subsequent pooling layer has access to a highly expressive representation.

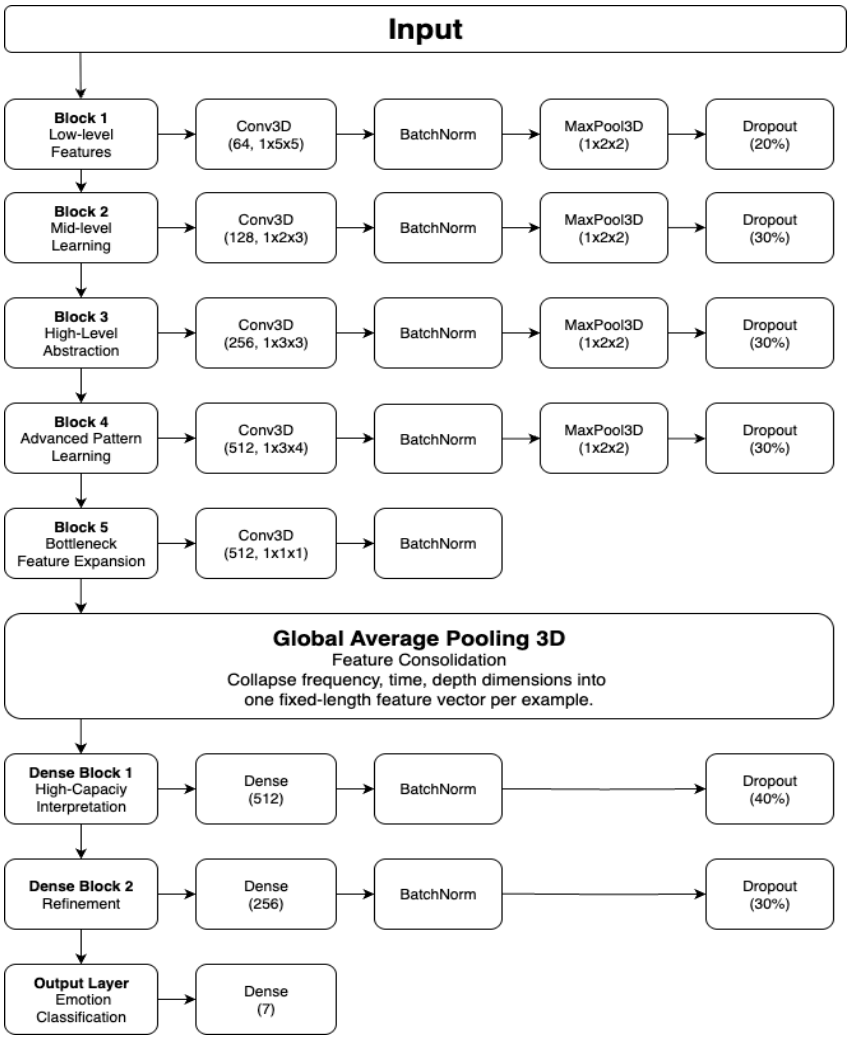
- **Global Average Pooling:** "Feature Consolidation": Its task is to collapse all remaining frequency, time, and depth dimensions into one fixed-length feature vector per example. To solve the difficult tasks we have been on, which involve dimensionality (losing semantics). Because it turns a large 3D volume into a manageable embedding that feeds into dense classifiers.

- **Dense Block 1:** "High-Capacity Interpretation": it contains a total of 3 layers, and the first main layer is the Dense (512), which is used to learn the complex combinations of the pool features. The second and third layers are the Batch Normalization and the Dropout (40%). Reduces the risk of overfitting by randomly dropping (40%) of the connections. This block provides a powerful, fully connected layer to interpret the global embedding and start the classification decision.

- **Dense Block 2:** “Refinement”: also a 3-layer block, with the first one being Dense (256). It further distills information to a more compact decision space. The second and third layers are the Batch Normalization and the Dropout (30%), providing additional regularization before the output.

- **Output:** “Emotion Classification”: which has only two layers. The first layer is Dense (7), with one node per emotion category. The second layer is the Softmax: Converts raw scores into probabilities that sum to 1. This block's ultimate goal is to produce the final probability distribution over the seven emotions (Neutral, Happy, Sad, Angry, Fearful, Disgust, Surprised).

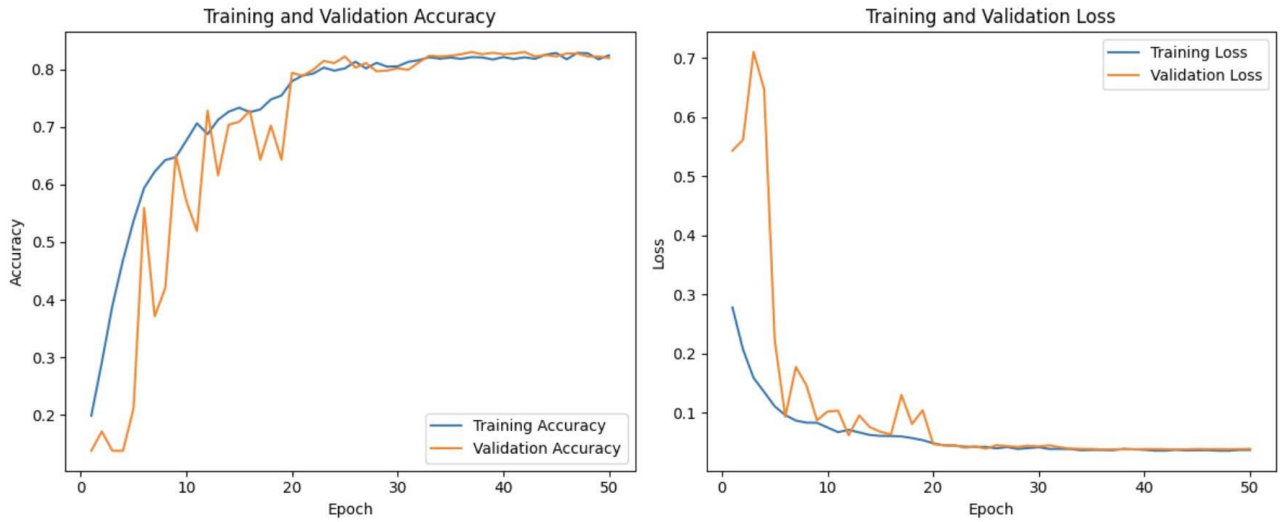
Now, this is the diagram of the layer flow of our model:



6. Result

6.1 Training History

The training history of the model over 50 epochs is illustrated in the figure below, showing both accuracy and loss trends for the training and validation datasets.



In terms of training and validation accuracy, the model learned well overall, with high final accuracy. Both training and validation accuracy increase steadily in the early epochs. Training accuracy improves smoothly, while validation accuracy shows more fluctuations, especially between epochs 5–20. The fluctuations in validation accuracy suggest some variability or noise in the validation data or sensitivity to class imbalance. From epoch 25 onward, both curves stabilize around 80–85%, indicating convergence. There are no signs of severe overfitting, as training and validation accuracy remain close near the end.

In respect of training and validation loss, both drop sharply in the first 10 epochs. The rapid loss reduction suggests effective learning during early training. After that, the losses remain consistently low, with a minimal gap between the two curves. Minimal difference between training and validation loss indicates good generalization.

Overall, the model shows strong learning behavior with good generalization and stable convergence. There is no clear overfitting, and training has successfully plateaued with high performance. Minor validation instability early on suggests room for further improvement through data balancing or augmentation.

6.2 Model Performance Evaluation

The table below summarizes the model's classification performance across seven emotion categories. The metrics include *precision, recall, F1-score*, and *support* for each class.

Classification Report:

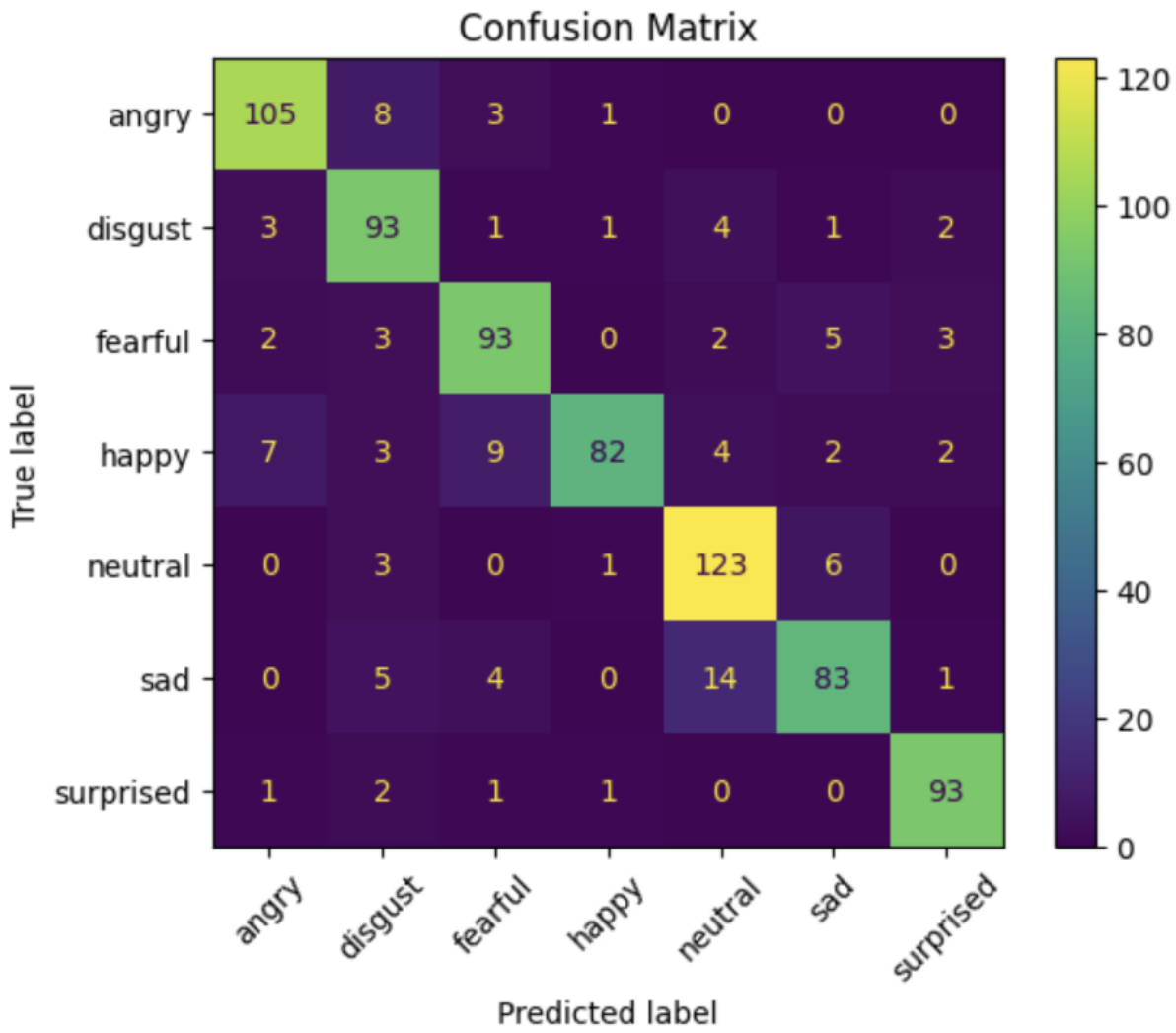
	precision	recall	f1-score	support
angry	0.89	0.90	0.89	117
disgust	0.79	0.89	0.84	105
fearful	0.84	0.86	0.85	108
happy	0.95	0.75	0.84	109
neutral	0.84	0.92	0.88	133
sad	0.86	0.78	0.81	107
surprised	0.92	0.95	0.93	98
accuracy			0.86	777
macro avg	0.87	0.86	0.86	777
weighted avg	0.87	0.86	0.86	777

Overall accuracy is 86% (667 out of 777 samples correctly classified). The model performs consistently well across all emotion categories, with precision and recall values generally above 0.84. "Surprised" achieved the highest F1-score (0.93), indicating both strong precision and recall in detecting this emotion. "Happy" shows a high precision (0.95) but lower recall (0.75), suggesting the model is conservative in predicting "happy"—it's accurate when it does, but misses some true positives. The "Disgust" class has the lowest precision (0.79), but high recall (0.89), which may indicate the model tends to over-predict "disgust".

The classification report indicates that the model is well-balanced across all classes and capable of generalizing to a multi-class emotion recognition task. While overall accuracy is high at 86%, further improvements in class-specific performance—especially in recall for "happy" and precision for "disgust"—could be achieved with additional data balancing or targeted fine-tuning.

6.3 Confusion Matrix

The confusion matrix visualizes the model's classification performance across seven emotion categories. Each row represents the true label, and each column represents the predicted label. Diagonal values indicate correct predictions, while off-diagonal values represent misclassifications.



The model shows strong performance across all classes, with the highest values concentrated along the diagonal. Correct classifications (diagonal cells):

- *Angry*: 105 out of 117
- *Disgust*: 93 out of 105
- *Fearful*: 93 out of 108
- *Happy*: 82 out of 109
- *Neutral*: 123 out of 133
- *Sad*: 83 out of 107
- *Surprised*: 93 out of 98

The model demonstrates high precision and recall, consistent with the earlier classification report. Misclassifications tend to occur between emotionally similar classes, such as:

- Happy ↔ Fearful

- Sad ↔ Neutral

This indicates that while the model is robust, finer emotional boundaries (e.g., between positive and negative valence emotions) still pose a challenge.

---

7. Conclusion

The model is a TensorFlow/Keras-based neural network utilizing a custom Focal Loss function, which addresses class imbalance (where the neutral class has a higher sample count). Spectrograms are used as input features, generated using Librosa and augmented with techniques such as Gaussian noise, pitch shift, and time stretch via audiomentations.

Training performance is visualized through accuracy and loss plots for both training and validation, indicating that the model was evaluated for convergence and generalization. The model is tested on a subset of the EmoDB dataset (5 audio files), predicting emotions like happy, neutral, and angry with confidence scores. Predictions include the top-3 emotions with probabilities, demonstrating the model's ability to rank likely emotions (with 90% confidence for "angry" in one case). The use of spectrograms and a custom prediction function indicates a focus on feature extraction and classification.

The system successfully processes an audio dataset, trains a deep learning model, and predicts emotions with reasonable confidence on an EmoDB test file. The model's performance (based on plots and predictions) suggests it generalizes well, though the slight accuracy drop for some predictions (36% for "happy") indicates potential for further tuning or data balancing. The use of data augmentation and focal loss shows attention to robustness and handling imbalanced classes.

---

8. Reference

1. **RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song):**
   Livingstone, S. R., & Russo, F. A. (2018). 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, 13(5), e0196391.
   Dataset available at https://doi.org/10.1371/journal.pone.0196391

2. **TESS (Toronto Emotional Speech Set):**
   Sahar, M., & Dupuis, K. (2010). Toronto Emotional Speech Set (TESS). Department of Psychology, Toronto Metropolitan University.
   Dataset available at https://tspace.library.utoronto.ca/handle/1807/24487

3. **SAVEE (Surrey Audio-Visual Expressed Emotion):**
   Haq, S., & Jackson, P. J. B. (2009). Speaker-dependent audio-visual emotion recognition. In *Proceedings of the International Conference on Auditory-Visual Speech Processing (AVSP)*.
   Dataset available at http://kahlan.eps.surrey.ac.uk/savee/Download.html

4. **EmoDB (Berlin Database of Emotional Speech):**
   Burkhardt, F., Paeschke, A., Rolfes, M., Sendlmeier, W. F., & Weiss, B. (2005). A database of German emotional speech. In Interspeech 2005 (pp. 1517–1520).
   Dataset available at https://doi.org/10.21437/Interspeech.2005-482