This chapter contains all the experiments on the SEWA and AFEW-VA datasets. Execution of the code was done only on one system, which is powered by an AMD Ryzen 9 5950X with 16 cores. The system has an NVIDIA GeForce RTX 3060 graphics card and 131 GB of memory. Python served as the programming language, and two packages were used during the experiments: PyTorch https://pytorch.org/ for the development of neural networks and scikit-learn https://scikit-learn.org/stable/., which was used for implementing machine learning algorithms such as Support Vector Regression (SVR), Bayesian Ridge (BR), Decision Tree Regression (DTR), as well as regression evaluation metrics.

Baseline:

Before the investigation of fusion methods and applying fusion started, we needed to build a simple model first- a baseline. It served as a starting point from which we could compare how well the complex fusion methods models perform.

In this section, the results of three machine learning algorithms are presented: Support Vector Regression (SVR), Bayesian Ridge (BR), and Decision Tree Regression (DTR). The goal is that the results of the deep learning neural network baseline, which is presented next, are the same or slightly better.

In this section, the results of three machine learning algorithms are presented: Support Vector Regression (SVR), Bayesian Ridge (BR), and Decision Tree Regression (DTR). The goal is for the results of the baseline deep learning neural network, which will be discussed next, to match with or have slightly better results than those of the algorithms.

Let us discuss the results of the AFEW-VA dataset first.

In Bayesian Ridge Regression model, first, datasets were loaded from CSV files and preprocessed by extracting features x and target values y. Since the range of arousal and valence values in AFEW-VA dataset is -10 to 10, additional normalization is needed. It is achieved by simply dividing the values by 10. Separate regression models were trained for arousal and valence using the training data.

Similarly, in Decision Tree Regression, datasets were loaded, features extracted, and target variables normalized by division by ten. It is also important to note that DTRs were initialized with a fixed random state, which controls the randomness involved during the node-splitting process and, therefore, ensures reproducibility. https://towardsdatascience.com/why-do-we-set-a-random-state-in-machine-learning-models-bb2dc68d8431 Afterward, the training on the arousal and valence data took place.

Lastly, in Support Vector Regression, in addition to loading datasets and extracting features, a grid search was conducted to optimize the SVR parameters (including parameter C and kernel type) using a predefined split https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.PredefinedSplit.html that combined training and development data for cross-validation, ensuring model validation before testing. The best models for valence and arousal were then used to predict arousal and valence on development and test data.

The performance of all three algorithms was evaluated using the development and test data using MSE and RMSE metrics.

```
\begin{table}[ht]

\centering

\caption{Results of the used machine learning algorithms on AFEW-VA.}

\label{tab:afew-alg-base}

\begin{tabular}{@{}lcccccc@{}}

\toprule

Algorithm     & MSE (V) & RMSE (V) & MSE (A) & RMSE (A) \\ \midrule

Bayesian Ridge & 0.009   & 0.09     & 0.019   & 0.13     \\

SVR           & 0.009   & 0.09     & 0.017   & 0.13     \\

Decision Tree  & 0.023   & 0.15     & 0.039   & 0.19     \\ \bottomrule

\end{tabular}

\end{table}
```

The table \ref{tab:afew-alg-base} presents the results of the described algorithms in the AFEW-VA dataset. Bayesian Ridge and SVR got approximately the same result. In both, the RMSE for valence equals 0.09, which means that the model's predictions are generally close to the actual data points, with an average error of 0.09 units. RMSE for arousal equals 0.13, meaning that the average magnitude of the errors in the predictions of arousal by the model is 0.13 units. Valence prediction appears to be more accurate than arousal prediction. The reason for this can be the differences in distribution patterns between arousal and valence in the AFEW-VA dataset. When looking back at Figure \ref{fig:afew-histogram} in Chapter 3, it is visible that the valence values are more distributed in the center with a peak around zero. So, extreme positive and negative values are less common.

On the other hand, arousal values have a more even distribution, from negative to positive. Unlike valence, there is no single dominant category. Therefore, it makes it more challenging to predict the arousal values accurately. Also, by looking at Figure \ref{fig:afew-circle}, it can be seen that valence is more concentrated around the center, while arousal is spread along the vertical axis.

This means that the worse arousal results can be due to the increased variability and spread in arousal values, which makes it harder for the model to predict the outcomes accurately. On the contrary, the concentration of valence values around the center is beneficial to the model since there is less extreme variation.

Unlike BR and SVR, the Decision Tree Regression algorithm ended up having worse results. RMSE for arousal is 0.19, and RMSE for valence is 0.15. This can be due to the fact that DTR is prone to overfitting, meaning it predicts well for the training data but can be inaccurate for new data https://medium.com/nerd-for-tech/overfitting-and-pruning-in-decision-trees-improving-models-accuracy-fdbe9ecd1160.

As established earlier in Chapter 2, BRR and SVR are more regularized forms of algorithms compared to DTR. Bayesian Ridge naturally prevents overfitting by including prior distributions on the model parameters and therefore avoids too complex models. SVR can handle non-linear data better by using kernel tricks and C parameter.

Now, let us discuss the SEWA results. The algorithms' implementation stays the same, except for the differences in the datasets.

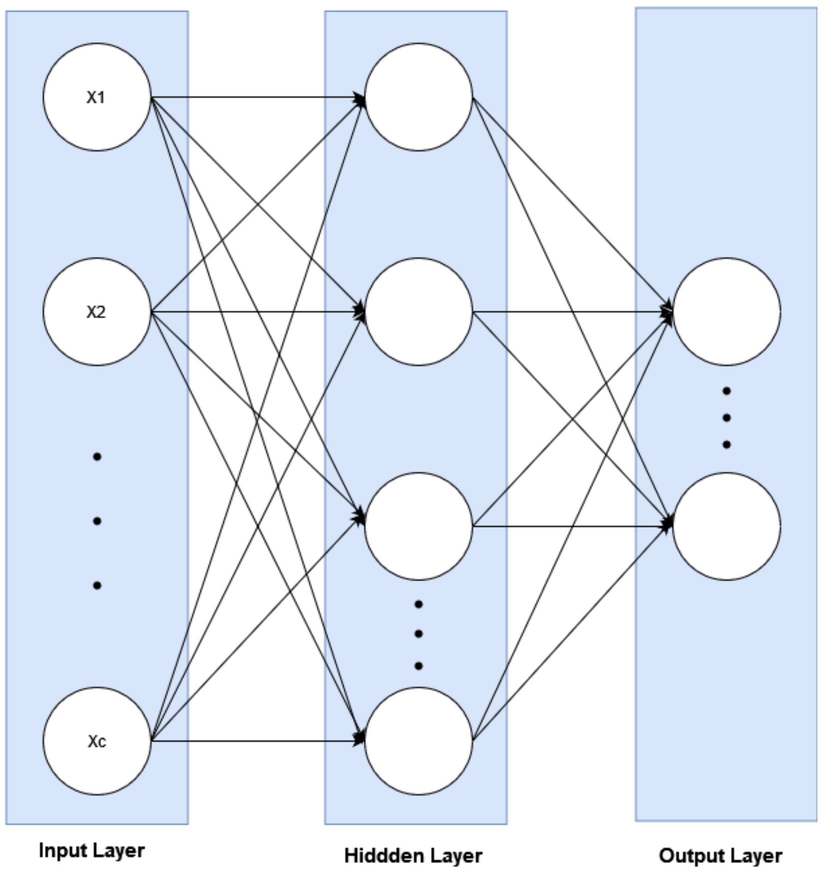
```
\begin{table}[ht]
\centering
\caption{Results of the used machine learning algorithms on SEWA.}
\label{tab:sewa-alg-base}
\begin{tabular}{@{}lcccccc@{}}
\toprule
Algorithm     & MSE (V) & RMSE (V) & MSE (A) & RMSE (A) \\
\midrule
Bayesian Ridge & 0.01    & 0.10     & 0.01    & 0.10     \\
SVR            & 0.01    & 0.11     & 0.01    & 0.10     \\
Decision Tree  & 0.03    & 0.17     & 0.02    & 0.14     \\
\bottomrule
\end{tabular}
\end{table}
```

As seen in Table \ref{tab:sewa-alg-base}, in all three algorithms, the results of valence prediction appear to be slightly worse than arousal. In contrast to the AFEW-VA dataset, in SEWA, the distribution of arousal is more evened out. However, valence seems to be leaning towards positive values with a peak around 0.2 to 0.4, as seen in Figure \ref{fig:sewa-histogram}. Positive valence values are more common in the dataset than neutral or negative values.

The distribution of valence can still be considered normal, except for the concentration of data points around the specific range. So, the slightly worse valence results can be due to the fact that models may struggle to accurately predict less frequent, extreme values. By looking at Figure \ref{fig:sewa-circle}, we can see a concentration around the center but also a broad spread along both axes, meaning a wide range of values in the data.

Here, too, the Decision Tree algorithm produces worse results than the other two algorithms. This suggests that Decision Tree Regression is not the best model choice for SEWA and AFEW-VA datasets in this emotion recognition task.

Neural network:



Now, let us look at the neural network's baseline results. The model's structure looks alike for both AFEW-VA and SEWA datasets.

It consists of:

1. Data loading and data preprocessing are described in Section 4.1.
2. Feed Forward Neural Network with a simple architecture consisting of:
    a. Input Layer
    b. Hidden Layer followed by ReLU activation function
    c. Output Layer followed by Tanh activation function for scaling the output

3. Training setup with Loss Function (RMSE), optimizer, and early stopping.

4. Training loop with epochs, batches, loss calculation, and validation.

5. Evaluation with MAE and RMSE metrics.

```
\begin{enumerate}

    \item Data loading and data preprocessing, which are described in Section 4.1.

    \item Feed Forward Neural Network with a simple architecture consisting of:

    \begin{itemize}

        \item Input Layer

        \item Hidden Layer followed by ReLU activation function

        \item Output Layer followed by Tanh activation for scaling the output

    \end{itemize}

    \item Training setup with Loss Function (RMSE), optimizer and early stopping.

    \item Training loop with epochs, batches, loss calculation and validation.

    \item Evaluation with MAE and RMSE metrics.

\end{enumerate}




\begin{table}[h]

    \centering

    \begin{tabular}{|m{4cm}|c|c|c|c|}

        \hline

        \textbf{Baseline on AFEW-VA} & \textbf{MAE (V)} & \textbf{RMSE (V)} & \textbf{MAE
(A)} & \textbf{RMSE (A)} \\

        \hline

        Feed Forward Neural Network & 0.07 & 0.09 & 0.08 & 0.13 \\

        \hline

    \end{tabular}

    \caption{Results for the baseline Feed Forward Neural Network on AFEW-VA}

     \label{tab:afew-nn-base}
```

```
\end{table}

\begin{table}[h]

   \centering

   \begin{tabular}{|m{4cm}|c|c|c|c|}

      \hline

      \textbf{Baseline with SEWA} & \textbf{MAE (V)} & \textbf{RMSE (V)} & \textbf{MAE (A)}
& \textbf{RMSE (A)} \\

      \hline

      Feed Forward Neural Network & 0.09 & 0.12 & 0.08 & 0.10 \\

      \hline

   \end{tabular}

   \caption{Results for the baseline Feed Forward Neural Network on SEWA}

   \label{tab:sewa-nn-base}

\end{table}
```

When comparing the results of the FFNNs from table \ref{tab:afew-nn-base} and table \ref{tab:sewa-nn-base} with those from the algorithms, we can see that the RMSE results are mostly the same. This means that all the models, except the Decision Tree, predict arousal and valence values well. The MAE results in both datasets suggest that the neural networks perform well. In SEWA, an MAE of 0.09 for valence means that, on average, the model's predictions are off by 0.09 units from the actual values. MAE of 0.10 for arousal shows that the model's predictions are off by 0.10 units on average. Similarly, in the AFEW-VA dataset, the predictions for valence are off by 0.07 units and for arousal by 0.13 units on average. Considering that the range of arousal and valence is frothe m -1 tof o 1, the error rate seems low.

It is crucial that the FFNNs provide good results since the following more complex experiments build upon the baseline.

FOR ME:

Data Loading and Preprocessing

1. **Loading Data:** Data is loaded from CSV files using `pandas`. It appears to be split into training, development, and test datasets.

2. **Feature Extraction:** Features (`X`) and labels (`y`) for valence and arousal are extracted. Features start from the column `emb_0` onward, suggesting some form of embedding or preprocessed features.
3. **Data Conversion:** The features and labels are converted into PyTorch tensors, which are suitable for model training.

DataLoaders

- **TensorDataset and DataLoader:** These are used to wrap the tensors into a dataset and allow for batch processing, shuffling, and efficient data loading during the model training and evaluation phases.

Model Definition

- **Fully Connected Neural Network:** Defined using a simple architecture with one hidden layer.
  - **Input Layer:** Takes inputs of dimension `input_dim`.
  - **Hidden Layer:** Consists of a linear layer followed by a ReLU activation function.
  - **Output Layer:** Another linear layer followed by a Tanh activation to scale the output. The output dimension is 2, corresponding to the two prediction targets: valence and arousal.

Training Setup

- **Loss Function:** Root mean squared error (RMSE) is used separately for arousal and valence predictions. This is customized through a class `RMSELoss` that computes the square root of MSE.
- **Optimizer:** Adam optimizer is used with a learning rate of 0.001.
- **Early Stopping:** Implemented to halt training if no improvement in validation loss is observed over a set number of epochs (`patience`).

Training Loop

- **Epochs and Batching:** The model is trained over a specified number of epochs (`epochs`), and within each epoch, data is processed in batches.
- **Loss Computation:** The model's predictions are split into arousal and valence components, and loss is calculated separately for each before summing them for backpropagation.
- **Validation:** At the end of each epoch, the model is evaluated on the development set, and early stopping criteria are checked.

Functions

- **`predict_on_dev` and `evaluate_on_test`:** These functions handle the evaluation on development and test sets respectively, computing metrics like mean absolute error (MAE) and RMSE for both valence and arousal.

Video model:

Next, we discuss the experiments on the video model. Since the model has to process several frames at a time and make predictions while considering them as one video, the model changes from frame-to-frame to sequence-to-one. Here, the Feed-Forward Neural Network no longer works, so we have to use another type of neural network—a Recurrent Neural Network, specifically GRU. GRU performs well when processing sequence data for predictions, where the context from previous inputs is important in order to understand current inputs.

```
\begin{figure}[h]

        \centering

        \includegraphics[width=0.8\textwidth]{figures/video-model.png}

        \caption{Visual representation of the RNN-based video model used in the
experiments.}

        \label{fig:video-model}

\end{figure}
```

Figure \ref{fig:video-model} shows the architecture of the used video model, which consists of the following:

1. Input Layer
2. GRU Layer with the main component - GRU, which can handle sequences of data
3. Dropout Layer, which randomly deactivates some nodes with their connections to prevent overfitting
4. Pooling: Temporal pooling (1D pooling) is applied to reduce the sequence length and summarize the features across the time dimension. This makes the data simpler by either averaging out (AVG Pool) the features over the sequence or selecting the maximal value from the features over the sequence. Conv1d is used to extract features from the sequence data by applying filters to the sequence to capture patterns over the sequence that may be important for the performance of the following layers.
5. Dense (Fully Connected) Layer, which maps the pooled features to the output size of arousal and valence
6. Tanh activation function, which is used to transform the output to be in the range -1 to 1
7. Output Layer

```
\begin{enumerate}

  \item Input Layer

  \item GRU Layer with the main component - GRU, which can handle sequences of data
```

\item Dropout Layer, which randomly deactivates some nodes with their connections to prevent overfitting

\item Pooling: Temporal pooling (1D pooling) is applied to reduce the sequence length and summarize the features across the time dimension. This makes the data simpler by either averaging out (AVG Pool) the features over the sequence or selecting the maximal value from the features over the sequence. \texttt{Conv1d} is used to extract features from the sequence data by applying filters to the sequence to capture patterns over the sequence that may be important for the performance of the following layers.

\item Dense (Fully Connected) Layer, which maps the pooled features to the output size of arousal and valence

\item Tanh activation function, which is used to transform the output to be in the range -1 to 1

\item Output Layer

\end{enumerate}

Now let us take a look at the results of the video models. Overall, 12 experiments were conducted on each dataset. For each window size (1, 2, 3 or 4 seconds) one of the three pooling types was applied. We will focus on the best results instead of displaying all results.

The best results of the AFEW-VA video model can be seen in Table \ref{tab:afew-best-video}.

\begin{table}[h]

\centering

\begin{tabular}{|m{3.5cm}|m{2cm}|m{2cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|}

\hline

\textbf{Video-based model on AFEW-VA dataset} & \textbf{Seconds, window size} & \textbf{Pooling} & \textbf{MAE (V)} & \textbf{RMSE (V)} & \textbf{MAE (A)} & \textbf{RMSE (A)} \\

\hline

& 2 sec, 10 & MAX Pool & 0.07 & 0.11 & 0.06 & 0.07 \\

\hline

& 3 sec, 15 & MAX Pool & 0.07 & 0.11 & 0.06 & 0.08 \\

```
        \hline
        & 4 sec, 20 & MAX Pool & 0.07 & 0.10 & 0.06 & 0.08 \\
        \hline
    \end{tabular}
    \caption{Best results of the video-based model on the AFEW-VA dataset}
    \label{tab:afew-best-video}
\end{table}
```

Interestingly, the MAX Pool has the best result on the AFEW-VA dataset for every amount of seconds (1 second is not included in the best results, but it still beats the other pooling approaches). In addition, a bigger window size brings the best results. That can be because everything longer than 1 second has a longer context. Compared to the baseline, this video model has the same result in MAE for valence but better in MAE for arousal. Even though the baseline model predicted arousal values worse than valence, in this video model, the prediction performance of arousal improved with more contextual data.

All results of the AFEW-VA video model can be found in Table \ref{tab:afew-video}.

```
\begin{table}[h]
    \centering
    \begin{tabular}{|m{3.5cm}|m{2cm}|m{2cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|}
        \hline
        \textbf{Video-based model on SEWA dataset} & \textbf{Seconds, window size} & \textbf{Pooling} & \textbf{MAE (V)} & \textbf{RMSE (V)} & \textbf{MAE (A)} & \textbf{RMSE (A)} \\
        \hline
        & 2 sec, 10 & AVG Pool & 0.08 & 0.10 & 0.08 & 0.11 \\
        \hline
        & 3 sec, 15 & AVG Pool & 0.07 & 0.10 & 0.08 & 0.11 \\
        \hline
        & 4 sec, 20 & AVG Pool & 0.08 & 0.10 & 0.08 & 0.11 \\
        \hline
    \end{tabular}
    \caption{Best results of the video-based model on the SEWA dataset}
    \label{tab:sewa-best-video}
\end{table}
```

Table \ref{tab:sewa-best-video} shows the best results of the video model on the SEWA dataset. Here, AVG Pool worked out best on all window sizes (again, 1 second is not included in the best results, but it beats the other pooling approaches). The results with 2, 3, and 4 seconds all seem similar, but 3 seconds appear to be the best among them. It has the same MAE for arousal but a slightly better MAE for valence. It appears that 3 seconds is an optimal value since it is not too long, like 4 seconds, but also not too short, like 1 or 2 seconds. Compared to the baseline, MAE for arousal stayed the same, but MAE for valence improved. So, even though the valence prediction performance was worse in the baseline model, in this video model it improved with more contextual data, too, like in AFEW-VA.

All results of the SEWA video model can be found in Table \ref{tab:sewa-video}.

Conv1D approach had bad results compared to AVG Pool and MAX Pool in both datasets. The exact reason behind this is unknown, but it can be because the convolutional filters are not effectively capturing the relevant features, or arousal and valence may have specifics in their temporal dynamics that are better captured by summarizing information over a sequence (with AVG and MAX Pools) instead of trying to detect specific patterns using a convolutional approach. Either way, Conv1D seems to be not the best approach choice in this video model for the AFEW-VA and SEWA datasets in this emotion recognition task.

Video 1dCnn bad results:

Yes, it's entirely possible that the 1D CNN architecture you've implemented may not be as effective for this specific task compared to using pooling layers like Max Pooling or Average Pooling. Each architecture has its strengths and is suited to different types of data and tasks. Here’s why pooling layers might be working better and why the 1D CNN might be struggling:

1. **Feature Sensitivity**:
    - **Pooling Layers**: Max pooling and average pooling are effective at reducing the dimensionality of input data while retaining the most important information (max pooling) or summarizing information (average pooling). This can be particularly useful in tasks where the exact location of features is less

important than their general presence or average characteristics across a window.
    - **1D CNN**: Convolutional layers are designed to detect specific features in the data, depending on the filters learned during training. If the convolutional filters are not effectively capturing the relevant features, or if the kernel size and stride are not appropriately configured, the CNN might not perform well. Specifically, if your kernel size matches the entire window size, it reduces the CNN's role to a transformation that might not extract useful, discriminative temporal features.
2. **Temporal Dynamics**:
    - Emotional states like arousal and valence may have nuances in their temporal dynamics that are better captured by summarizing information over a period rather than trying to detect specific patterns using a convolutional approach. Pooling can abstract these temporal features more effectively in some cases.
3. **Model Complexity and Overfitting**:
    - CNNs add more parameters to the model, which can lead to overfitting especially if the dataset is not large enough or the model is too complex relative to the amount of available training data. Pooling layers, on the other hand, reduce the complexity by summarizing the data, which might help in generalizing better on unseen data.
4. **Data Suitability**:
    - The nature of your data might inherently suit methods that aggregate information (like pooling) over methods that seek to transform and then extract features (like CNNs). This can depend heavily on how emotions are expressed and captured in the dataset.

AUDIO MODEL:

Before discussing the fusion results, we need to examine the audio model's results because fusion is based on audio and video. Four experiments were conducted using the audio model on only the SEWA dataset since AFEW-VA contains no audio.

As stated earlier in Section 3.3, four audio CSV files contain extracted features from 1,2,3 and 4-second temporal windows. The architecture stayed the same as in the baseline SEWA model described earlier, which can also be seen in Figure \ref{fig:FFNN}.

```
\begin{table}[h]

  \centering

  \begin{tabular}{|m{4cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|}

    \hline

    \multirow{5}{*}{Audio-based model on SEWA dataset} & \textbf{Seconds} & \textbf{MAE (V)} & \textbf{RMSE (V)} & \textbf{MAE (A)} & \textbf{RMSE (A)} \\

    \cline{2-6}
```

```
        & 1 sec & 0.13 & 0.16 & 0.13 & 0.17 \\
        \cline{2-6}
        & 2 sec & 0.13 & 0.16 & 0.13 & 0.18 \\
        \cline{2-6}
        & 3 sec & 0.13 & 0.16 & 0.13 & 0.18 \\
        \cline{2-6}
        & 4 sec & 0.13 & 0.16 & 0.13 & 0.18 \\
        \hline
    \end{tabular}
    \caption{Performance of the audio-based model on the SEWA dataset}
    \label{tab:audio}
\end{table}
```

The results of the audio model can be seen in Table \ref{tab:audio}. All of them are the same, so there is no visible observation regarding temporal window length, as observed in the video model. Overall, the audio model performed worse than the video model. It can be because audio often consists of more noise. A person can be, e.g., silent on the video, so the model has nothing to predict. A face is generally more expressive than audio, which gives better results in prediction performance in the SEWA video model.

FUSION:

Lastly, this section presents the results of various fusion methods. Overall, four fusion methods were selected. Each of them was tested on a different window size, just like the video and audio models. Additionally, the last fusion method had three variations. So, in total, 24 experiments were conducted.

Let us go through the methods, starting from the simplest one. The architecture of fusion 1, shown in Figure \ref{fig:Fusion4} consists of the following:

1. Data loading and data preprocessing (synchronization of video and audio)
2. AVG Pool: Video/audio features for each window are pooled across time dimension, reducing them to a single vector
3. Concatenation of pooled video and audio features
4. Dense Layer (Fully Connected Layer with ReLU activation function)
5. Output Layer

\begin{figure}[h]

    \centering

    \includegraphics[width=0.8\textwidth]{figures/Fusion4.png}

    \caption{Visual representation of fusion method 1.}

    \label{fig:Fusion4}

\end{figure}

Starting from Fusion 2, the methods get more complex. They now utilize the modern deep learning architecture - a transformer.

\begin{figure}[h]

    \centering

    \includegraphics[width=0.8\textwidth]{figures/Fusion3.png}

    \caption{Visual representation of fusion method 2.}

    \label{fig:Fusion3}

\end{figure}

Fusion 2, as shown in Figure \ref{fig:Fusion3} consists of:

1. Data loading and data preprocessing (synchronization of video and audio)
2. Linear Layer, where audio features are normalized to match the dimension of the video features
3. Concatenation of normalized audio features and video features
4. 2 Transformer Blocks (self-attention mechanisms) which have:
    a. Multi-Head-Attention
    b. Add and Norm
    c. Position-wise Feed-Forward Network, where a fully connected feed-forward network is applied separately to each position
    d. Positional Encoding, which provides information about each element's position in the sequence. It is important because it helps the model understand the sequence order
5. AVG Pool, which is applied to convert the sequence of output features from the Transformer into a single feature vector
6. Output Layer

Fusion 3 is similar to Fusion 2, but it also incorporates cross-attention. It, as shown in Figure \ref{fig:Fusion2} consists of:

\begin{figure}[h]

\centering

\includegraphics[width=0.8\textwidth]{figures/Fusion2.png}

\caption{Visual representation of fusion method 3.}

\label{fig:Fusion2}

\end{figure}

1. Data loading and data preprocessing (synchronization of video and audio)
2. Linear Layer and Padding, where audio features are normalized to match the dimension of the video features. Afterward, the audio features were padded to match the number of timesteps with those in the video
3. Transformer block (cross attention mechanism), which processes the video and padded audio features together
4. Transformer block (self-attention mechanism), which further processes the output of the cross-attention layer
5. AVG Pool
6. Output Layer

Lastly, fusion 4 has three different variations, but they all incorporate only cross-attention.

\begin{figure}[h]

\centering

\includegraphics[width=0.8\textwidth]{figures/Fusion1v1.png}

\caption{Visual representation of fusion method 4, variation 1.}

\label{fig:Fusion1v1}

\end{figure}

Here, as shown in Figure \ref{fig:Fusion1v1}, is the architecture of the first variation of fusion 4:

1. Data loading and data preprocessing (synchronization of video and audio)
2. Linear Layer, where audio features are normalized to match the dimension of the video features
3. Conv1D Layer, where dimensions of video features are adjusted
4. Transformer block (cross-attention mechanism), which processes the encoded video features as query and the encoded audio features as key and value.
5. Transformer block (self-attention mechanism), which processes the output of the cross-attention layer
6. AVG Pool
7. Linear and Output Layer, where the pooled features are passed through a fully connected layer to predict arousal and valence

The second variation, as shown in Figure \ref{fig:Fusion1v2}, has the following architecture:

```
\begin{figure}[h]

        \centering

        \includegraphics[width=0.8\textwidth]{figures/Fusion1v2.png}

        \caption{Visual representation of fusion method 4, variation 2.}

        \label{fig:Fusion1v2}

\end{figure}
```

1. Data loading and data preprocessing (synchronization of video and audio)
2. Linear Layer, where audio features are normalized to match the dimension of the video features
3. Conv1D Layer, where dimensions of video features are adjusted
4. Transformer block (cross attention mechanism), which processes the encoded video features and the encoded audio features.
5. Transformer block (cross attention mechanism), which reapplies cross-attention to the output of the first cross-attention layer, which is integrated as query and video features are integrated as keys and values
6. AVG Pool
7. Linear and Output Layer, where the pooled features are passed through a fully connected layer to predict arousal and valence

The third variation is almost identical to the second variation, except that in the second transformer block, audio features are integrated as keys and values instead of video. The architecture of the third variation is shown in Figure \ref{fig:Fusion1v3}

```
\begin{figure}[h]

        \centering

        \includegraphics[width=0.8\textwidth]{figures/Fusion1v3.png}

        \caption{Visual representation of fusion method 4, variation 3.}

        \label{fig:Fusion1v3}

\end{figure}



\begin{table}[h]

    \centering

    \begin{tabular}{|m{3cm}|m{2cm}|m{2.5cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|}
```

```
        \hline

        \textbf{Best Fusion Models on test data} & \textbf{Seconds, window size} &
\textbf{Pooling} & \textbf{MAE (V)} & \textbf{RMSE (V)} & \textbf{MAE (A)} & \textbf{RMSE
(A)} \\

        \hline

        Fusion 1 & 2 sec, 10 & AVG Pool 2x & 0.0893 & 0.1145 & 0.0930 & 0.1183 \\

        \hline

        Fusion 2 & 4 sec, 20 & AVG Pool & 0.0902 & 0.1174 & 0.0988 & 0.1284 \\

        \hline

        Fusion 3 & 1 sec, 5 & AVG Pool & 0.0928 & 0.1206 & 0.1025 & 0.1355 \\

        \hline

        Fusion 4 (v2) & 4 sec, 20 & 1D CNN, AVG Pool & 0.0815 & 0.1035 & 0.0956 & 0.1239
\\

        \hline

    \end{tabular}

    \caption{Results of the best fusion method on test data.}

    \label{tab:fusion-best-test}

\end{table}


\begin{table}[h]

    \centering

    \begin{tabular}{|m{3cm}|m{2cm}|m{2.5cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|m{1.5cm}|}

        \hline

        \textbf{Best Fusion Models on dev data} & \textbf{Seconds, window size} &
\textbf{Pooling} & \textbf{MAE (V)} & \textbf{RMSE (V)} & \textbf{MAE (A)} & \textbf{RMSE
(A)} \\

        \hline

        Fusion 1 & 2 sec, 10 & AVG Pool 2x & 0.0606 & 0.0830 & 0.0689 & 0.0972 \\

        \hline
        Fusion 2 & 4 sec, 20 & AVG Pool & 0.0601 & 0.0892 & 0.0639 & 0.0904 \\

        \hline

        Fusion 3 & 1 sec, 5 & AVG Pool & 0.0606 & 0.0822 & 0.0709 & 0.0985 \\

        \hline

        Fusion 4 (v2) & 4 sec, 20 & 1D CNN, AVG Pool & 0.0583 & 0.0809 & 0.0679 & 0.0940
\\

        \hline

    \end{tabular}

    \caption{Results of the best fusion methods on development data.}

    \label{tab:fusion-best-dev}

\end{table}
```

Table \ref{tab:fusion-best-test} demonstrates the best result of each fusion method.

It seems that each fusion type has better results on different window length. In fusion 2 and 4 (variation 2), 4 seconds length has the best results in MAE for valence and MAE for arousal. This means that these types of fusion benefit from longer contextual information and therefore predicts outcomes better. Fusion 1 and 3, on the other hand, work better on shorter window length and have the best performance on 1 and 3 seconds, respectfully.

If we compare arousal and valence prediction performance separately, it can be observed that with longer window length, meaning longer contextual information, valence prediction gets better in fusion 4. In contrary, arousal prediction are worse with longer window length.

One possible reason is that the audio data, after synchronizing it with video, "confuses" the model and makes the predictions worse. The best audio-based model had a window length of 1 second, where arousal predictions were slightly better than valence. So, it makes sense that the fusion model performs arousal predictions better on shorter window lengths.

In addition, it is observable that fusion 4 (variation 2) performed better than fusion 4 (variation 3). The first one had video features integrated into the second cross-attention mechanism and the other audio features. So this also leads to the conclusion that by incorporating additional audio features, they drag the overall performance of the fusion model down, even though the video model alone used to perform better. Similarly, the fusion 4 (v2) model is better at predicting valence than baseline, but is slightly worse in predicting arousal than the baseline.

When we compare the results on test data with the results on demonstration data in Table \ref{tab:fusion-best-dev}, we see that the model learned way better on demonstration data but has worse results on test data. This means that the fusion models adapt worse to new, unseen data.

If we now look back at the video model, it has better results than the fusion methods, which means that the video model's generalization is better than the generalization of the fusion models.

Overall, after carrying out all the fusion experiments, we can say that the audio-visual fusion models outperform the audio model. This is probably due to the fact that the video model alone performs well at prediction arousal and valence values. The video modality alone still has better results than the fusion.

FOR ME:

1. **Cross-Attention Mechanisms**:
    - **First Cross-Attention Layer**: Processes the encoded video features as queries and the encoded audio features as keys and values. This layer allows the model to integrate audio information based on the relevance to the video content.
    - **Second Cross-Attention Blocks**: Applied three times consecutively, these layers further refine the integration between audio and video features by reapplying cross-attention to the output of the first cross-attention layer.
2. **Pooling and Prediction**:
    - **Pooling**: An `nn.AdaptiveAvgPool1d` layer reduces the dimensionality of the final attention output to a single vector per sample, preparing it for the prediction step.
    - **Output Layer**: An `nn.Linear` layer converts the pooled features into final predictions for the emotional states (arousal and valence).

DATA MANIPULATION:

Before starting with the experiments, specific data manipulation is necessary. Firstly, the dataset had to be split into training, development (also known as validation), and test subsets. It is a standard practice in machine learning. The reason behind this is simple: the model needs to have a separate train dataset to learn patterns from the data and relationships between features and outputs. The development dataset is used to tune the model's hyperparameters, allowing developers to adjust the model for better performance. Test data is used to evaluate the model's performance on completely unseen data.

The split ratio used in the experiments is 70% train data, 15% development data, and 15% test data. It is important to note that the split between train/dev/test was the same across all experiments. Since we have separate audio and video datasets when performing fusion, it was necessary to ensure that the models are evaluated under the same conditions.

https://cs230.stanford.edu/blog/split/ https://encord.com/blog/train-val-test-split/

Key Benefits of Dataset Splitting:

- **Prevents Overfitting**: By using separate data for training and validation, you can minimize the risk of overfitting, where a model learns the training data too well, including its noise and errors, and performs poorly on any new data.
- **Enables Hyperparameter Tuning**: The validation set allows for effective tuning of hyperparameters without contaminating the test set, which should remain a truly independent subset for evaluating the model's generalization capability.
- **Improves Model Robustness and Generalization**: By testing on a set that has never been used during training or tuning (the test set), you can measure how well your model is likely to perform under real-world conditions, on data it has never seen before.

Practical Considerations:

- **Dataset Size**: The way you split your dataset can depend on its size. Larger datasets might allow for smaller proportions to be allocated to training, while smaller datasets might require careful techniques like cross-validation to maximize the use of available data.
- **Stratification**: When splitting, it's important to maintain a representative ratio of classes (in classification tasks) across different subsets to avoid biases that could skew the performance measurements.

Overall, proper dataset splitting is crucial for developing robust, effective, and generalizable machine learning models, ensuring that they perform well not just on the data they were trained on, but also on new, unseen data.

DATA PREPROCCESING:

In the experiments, there are three ways how data is preprocessed, depending on the model. They all integrate the embeddings' extraction described in Chapter 3.

For the baseline neural network model, after extracting the embeddings and saving them in a CSV file, each frame that consists of embeddings (features and labels) is transformed into a 1D tensor. Afterward, each frame with embeddings is put into a 2D tensor with the size (m,n), where m are frames and n are features with labels. It is then passed to a DataLoader.

The DataLoader is used in every model in the experiments since it can effectively manage the data during the training process. It groups data into manageable sizes (creates batches), performs shuffling on the data to ensure data randomness, and handles parallel processing to make data loading quicker. https://pytorch.org/tutorials/beginner/basics/data_tutorial.html

So, DataLoader outputs batches of data, which are then fed into the input layer of a neural network model. The visual representation of the data preprocessing for the baseline model can be seen in Figure \ref{fig:preprocess1}.

For the video model, each frame is transformed into a 1D tensor and sequences of frames (windows) are created using a window size that was initialized at the beginning. Then, we need to iterate through each row in the data frame and match the frames with the corresponding embeddings. Then, the embeddings are converted into a PyTorch tensor. Similarly, that window's average arousal and valence labels are also converted into a PyTorch tensor. DataLoader then fetches batches of windowed frame data and corresponding labels during model training, validation, or testing. Figure \ref{fig:preprocess2} presents the data preprocessing process for the video model.

In the fusion model, the preprocessing is more complex because it includes synchronization of video and audio data based on timestamps. First, video IDs from file paths in the video data frame are extracted. This ID is used to group video and audio data by video, ensuring synchronization within each video separately. We filter out the corresponding video and audio data for each video ID. This is done by selecting rows from the video data frame where the path contains the video ID and from the audio data frame where the filename contains the video ID. We experiment with different window lengths in each fusion, so this should be considered while synchronizing video and audio, too. So, we ensure that the timestamps align in video and audio. Afterward, the last video frame's arousal and valence scores are used as labels for the window. Features are also extracted from video and audio data within the window. The synchronized data is then stored in a tuple consisting of the video features, audio features, labels, video name, and timestamp.

Figure \ref{fig:preprocess3} shows the simplified data preprocessing workflow for fusion models.

CONCLUSION:

This work explores various audio-visual fusion methods in the context of emotion recognition. Before proceeding with fusion approaches, video, and audio models were built first. The video model provided good results, which were better than the baseline model results. The audio model’s results were worse than the video, presumably due to the noise in the audio data.

For the fusion approaches, the data was preprocessed first, which involved synchronizing audio and video using a customized PyTorch DataLoader. In the experiments, four different fusion approaches were investigated. One was a simple method incorporating a fully connected layer, and the other three incorporated the modern deep learning architecture - a transformer. The best result of each fusion method was evaluated, and it showed that the best results of all four fusion approaches outperformed the audio model, and two of them had the same result as the video model in valence prediction. The arousal prediction ended up being slightly worse because the audio data seemed to drag the overall performance of the fusion methods down.

In future work, we can experiment with different databases using the same fusion approaches and analyze how well the fusion model performed compared to baseline models. The hyperparameters of the models also could be adjusted, like early stopping or a number of epochs, to see if that provides better results. Another possibility could be to try incorporating different architectures into the fusion models. The work of Huang et al. \cite{huang} shows that combining transformer with LSTM further improves the performance and achieves good results. It can be experimented whether Transformer-LSTM architecture has a better prediction performance.

This thesis shows that audio-visual fusion achieves better results than audio modality alone and has as good results as valence in video modality, which shows the potential for further improving fusion methods to achieve better results.

Abstract:

Recently, there has been a growing interest in multimodal emotion recognition. One modality, such as speech, gestures, or facial features, often does not portray the complete emotional state because an emotion combines several modalities simultaneously.

This work investigates the impact of various fusion methods on emotion recognition performance. The fusion methods incorporated two modalities: video with facial features and audio with audio features. This thesis provides insight into how features can be extracted from a dataset and further preprocessed for models based on individual modalities and an audio-visual fusion model, where data synchronization is crucial.

The performance of video and audio models were evaluated and afterward, the fusion experiments were conducted. Each fusion method had a different architecture, but three of them incorporated a transformer. The experiments prove that even on challenging dataset like SEWA, the proposed fusion methods either outperform singular modality model or partly have the same results as them.

Overall, this work shows the importance of combining different modalities in emotion recognition task and show promising results for future study in that area. Recent state-of-the-art shows that emotion recognition can be done by utilizing multiple approaches that all lead to different results.

Introduction:

Emotion plays a crucial part in everyday human experiences, impacting communication and interaction. The rapid growth of interest in human-computer interaction raises the need for systems that can successfully capture the human's emotional state.

https://www.sciencedirect.com/science/article/pii/S1574013723000126
https://link.springer.com/chapter/10.1007/978-3-540-85099-1_8

https://www.researchgate.net/publication/304124018_Building_Chinese_Affective_Resources_in_Valence-Arousal_Dimensions https://arxiv.org/pdf/2103.15792

Emotion states are usually represented using either a categorical or dimensional approach.

In a categorical approach, Ekman outlines six basic emotions (anger, happiness, fear, sadness, disgust, and surprise) based on a cross-cultural study. This study shows that humans, regardless of cultural background, perceive basic emotions in the same way.

In the dimensional approach, emotions are represented through a continuous numerical value in multiple dimensions, like valence-arousal space. Valence represents the degree of positive and negative feelings, and arousal represents the degree of excitement and calm. Based on this valence-arousal space, every emotional state can be represented as a point in the coordinate plane, as shown in Figure \ref{arousal-valence-space}.

Figure: the valence-arousal space.

Praveen et al. \ref{praveen} note that dimensional modeling of emotions is generally more challenging than categorical one because it is more difficult to get a continuous scale of annotations than an emotion category. The annotations often seem noisy and ambiguous due to the continuous range of emotions. Several datasets were explicitly made for dimensional emotion recognition. Among them is SEWA, which is utilized in this work.

Emotions are usually portrayed nonverbally with facial expressions, gestures, body language, and verbally with speech. https://ieeexplore.ieee.org/document/10252617

In real life, emotions are expressed simultaneously with several nonverbal and verbal modalities. So, to capture more emotional context, a multimodal fusion of modalities can be used. In this work, we investigate the impact of audio-visual fusion methods on emotion recognition performance. First, each individual modality and its performance are examined. Afterward, audio-visual fusion experiments are conducted. The preprocessed audio and video data, which served as input for all models, were also discussed.

Moving forward, this paper will cover everything discussed in the following chapters. Chapter 2 summarizes related work and the background knowledge of machine learning algorithms and deep learning. Chapter 3 covers the used datasets in this work and how audio and video embeddings were extracted from the original data. Additionally, the evaluation metrics used in the experiments are described. Chapter 4 follows the explanation of how experiments were performed. First, the data manipulation process, including data loading and preprocessing for models based on one modality and fusion models, is demonstrated. After setting up the baseline with algorithms and a simple feed-forward neural network, models using only one modality were built, and their results were evaluated in sections 4.3 and 4.4. Lastly, four fusion methods were performed, experimenting with various window lengths and architectures. In the end, Chapter 5 follows a summary and reflection of this work. Also, possible future work was discussed. Appendix A contains tables with the results of experiments on video, audio, and fusion models. Additionally, the visualization of data preprocessing and fusion methods' architectures can be found there.