National Technical University of Athens

School of Electrical and Computer Engineering

Data Science and Machine Learning

Recognition and localization of audio-visual events with machine learning technique

Postgraduate Diploma Thesis

of

NIKOLAOS D. MAKARIS

Supervisor: Stefanos Kollias
Professor N.T.U.A.

Co-Supervisor: Paraskevi Tzouveli
Labatory teaching staff N.T.U.A.

Artificial Intelligence and Learning Systems (AILS) Laboratory

Athens, March 2023

National Technical University of Athens
School of Electrical and Computer Engineering
Data Science and Machine Learning
Artificial Intelligence and Learning Systems (AILS) Laboratory

Recognition and localization of audio-visual events with machine learning technique

Postgraduate Diploma Thesis

of

NIKOLAOS D. MAKARIS

Supervisor: Stefanos Kollias
Professor N.T.U.A.

Approved by the three-member examination committee on 17th March 2023.

(Sign)	*(Sign)*	*(Sign)*
.....	.....	.....
Stefanos Kollias	Georgios Stamou	Athanasios Voulodimos
Professor N.T.U.A.	Professor N.T.U.A.	Assistant Professor N.T.U.A.

Athens, March 2023

(Sign)

.....

MAKARIS NIKOLAOS
Post graduate of Data Science And Machine Learning N.T.U.A

National Technical University of Athens
School of Electrical and Computer Engineering
Data Science and Machine Learning
Artificial Intelligence and Learning Systems (AILS) Laboratory

Περίληψη

Η αναγνώριση οπτικοακουστικών γεγονότων και η επιχώρια προσαρμογή είναι ένα δύσκολο έργο που περιλαμβάνει την αναγνώριση γεγονότων που είναι τόσο ορατά όσο και ηχητικά σε ένα βίντεο. Σε αυτήν τη μελέτη, προτείνουμε μια καινοτόμο προσέγγιση για την αντιμετώπιση αυτής της πρόκλησης χρησιμοποιώντας έναν οπτικοακουστικό μηχανισμό οπτικής προσοχής για τη διερεύνηση οπτικοακουστικών συσχετίσεων και τη χρήση ενός διπλού πολυτροπικού δικτύου υπολειπόμενων εκπομπών (DMRN) για τη σύντηξη πληροφοριών μεταξύ των δύο τρόπων.

Η μεθοδολογία μας περιλαμβάνει την εξαγωγή χαρακτηριστικών (οπτικοακουστικών ή οπτικών) από διάφορα προ-εκπαιδευμένα μοντέλα, τα οποία έχουν αναπτυχθεί για εργασίες όπως αναγνώριση εικόνας ή αναγνώριση ήχου. Στη συνέχεια ορίζουμε καινοτόμες αρχιτεκτονικές για τα πολυτροπικά δίκτυα, με στόχο τον αποτελεσματικό εντοπισμό των γεγονότων-στόχων στα οπτικοακουστικά δεδομένα.

Για να αξιολογήσουμε την απόδοση της προτεινόμενης προσέγγισής μας, την εφαρμόζουμε στο σύνολο δεδομένων AVE και συγκρίνουμε τα αποτελέσματα με αυτά που αναφέρονται σε άλλες σχετικές μελέτες. Διαπιστώνουμε ότι η προσέγγισή μας επιτυγχάνει καλύτερη ακρίβεια στην αναγνώριση των γεγονότων.

Η μελέτη αυτή συμβάλλει στον τομέα της αναγνώρισης οπτικοακουστικών γεγονότων και της τοπικοποίησης με την εισαγωγή ενός νέου πλαισίου που συγχωνεύει αποτελεσματικά τις οπτικοακουστικές πληροφορίες, οδηγώντας ενδεχομένως σε βελτιωμένη απόδοση και ταχύτερους χρόνους επεξεργασίας σε διάφορες εφαρμογές τοπικοποίησης.

Λέξεις Κλειδιά

Αναγνώριση οπτικοακουστικών συμβάντων, Εντοπισμός συμβάντων, Πολυτροπική σύντηξη, Οπτική προσοχή με γνώμονα τον ήχο, Προεκπαιδευμένα μοντέλα, Εξαγωγή χαρακτηριστικών γνωρισμάτων, Σύνολο δεδομένων AVE

Abstract

Audio-visual event recognition and localization is a challenging task that involves identifying events that are both visible and audible in a video. In this study, we propose a novel approach to address this challenge by employing an audio-guided visual attention mechanism to explore audio-visual correlations and leveraging a dual multimodal residual network (DMRN) to fuse information across the two modalities.

Our methodology includes extracting features (audio or visual) from various pre-trained models, which have been developed for tasks such as image recognition or audio recognition. We then define novel architectures for the multimodal networks, aiming to effectively localize the target events in the audio-visual data.

To evaluate the performance of our proposed approach, we apply it to the AVE dataset and compare the results with those reported in other relevant studies. We find that our approach achieves better accuracy in recognizing the events.

This study contributes to the field of audio-visual event recognition and localization by introducing a novel framework that effectively fuses audio and visual information, potentially leading to improved performance and faster processing times in various localization applications.

Keywords

Audio-visual event recognition, Event localization, Multimodal fusion, Audio-guided visual attention, Pre-trained models, Feature extraction, AVE dataset

Acknowledgements

Ευχαριστώ τον καθηγητή Στέφανο Κόλλια και τα μέλη του εργαστηρίου Συστημάτων Τεχνητής Νοημοσύνης και Μάθησης για την ευκαιρία που μου δόθηκε να εργαστώ στο συγκεκριμένο θέμα της διπλωματικής εργασίας μου.

Ευχαριστώ ιδιαίτερα την ΕΔΙΠ κα Παρασκευή Τζούβελη καθώς και τον διδακτορικό Έντι Δερβάκο, που μου προσέφεραν την κάθε δυνατή βοήθεια και σωστή καθοδήγηση κατά την εκπόνηση της διπλωματικής μου εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω τους καθηγητές Γεώργιο Στάμου και Αθανάσιο Βουλόδημο που συμμετέχουν στην τριμελή επιτροπή.

Τέλος, θέλω να ευχαριστήσω πολύ τους φίλους μου για την υποστήριξη που μου προσέφεραν καθ' όλη τη διάρκεια εκπόνησης της μεταπτυχιακής διπλωματικής μου εργασίας και την οικογένειά μου που είναι πάντα δίπλα μου.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Subject of diploma thesis

In this diploma thesis, the problem of audio-visual event localization in unconstrained videos is examined. An audio-visual event is defined as an event that is both visible and audible in a video segment.

Audio-visual event localization in unconstrained videos is investigated. An audio-visual event is defined as an occurrence that can be both seen and heard within a video segment. The Audio-Visual Event (AVE) dataset is used in order to systematically study supervised audio-visual event localization. An audio-guided visual attention mechanism to explore audio-visual correlations and multiple networks is investigated in order to merge information from the two modalities. The experimental results reveal that jointly modeling auditory and visual modalities surpasses independent modeling.

Furthermore, multiple pre-trained models are used for the feature extraction (Audio and Visual). Some of these models could be trained on a similar task, i.e. image recognition and some other in relevant tasks, i.e. image segmentation for the visual module, or speaker recognition for the audio module.

1.2 Similar work

This diploma thesis is inspired heavily by the "Audio-Visual Event Localization in Unconstrained Videos" paper by Tian et. al [2].

Furthermore, the paper "Dual-modality seq2seq network for audio-visual event localization" by Yan-Bo Lin, Yu-Jhe Li, and Yu-Chiang Frank Wang [1] presents a deep neural network called Audio-Visual sequence-to-sequence dual network (AVSDN) for audio-visual event localization. This task involves identifying events that are both visible and audible in a video, either at the frame or video level.

The proposed AVSDN model jointly takes both audio and visual features at each time segment as inputs and learns global and local event information in a sequence-to-sequence manner. This can be achieved in either fully supervised or weakly supervised settings. The

authors' empirical results confirm that their proposed method performs favorably against recent deep learning approaches in both fully supervised and weakly supervised settings. By considering both audio and visual information, the AVSDN model demonstrates a more effective approach to localizing audio-visual events in videos.

1.3 Document format

In Chapter 2, the concept of AVE Localization and recognition in unconstrained videos is discussed. Chapter 3 delves into the theoretical aspects of machine learning, neural networks, and key concepts necessary for understanding the experiments. A brief description of the audio and visual modules is also provided.

Chapter 4 introduces the datasets utilized in this work for model training and testing, as well as a brief description of the datasets used in the pre-trained models serving as feature extractors.

Chapter 5 details the implementation of the neural networks and models employed in this work. Chapter 6 presents the results of most experiments, and Chapter 7 concludes the discussion with a summary, possible future works, and final thoughts.

Part I

Theory

Chapter 2

Description of the concept

This chapter describes the subject of this diploma thesis about Audio and Visual Event localization.

2.1 Audio and Visual Event localization

In this diploma thesis, the problem of audio-visual event recognition and localization in unconstrained videos is examined, shown in Figure 2.2. An audio-visual event is defined as an event that is both visible and audible in a video segment [2].

A significant amount of visual and auditory signals are present in real-world activities, and numerous studies in neurobiology and human perception indicate that there are considerable perceptual benefits to integrating visual and auditory information. In computational models, this integration is evident in lip reading, where the correlation between speech and lip movements provides a robust cue for language comprehension. Another example is sound synthesis, where physical interactions with various materials produce believable sound patterns. Despite the advancements in these models, they remain limited to certain constrained domains.

As a result, some researchers have started to explore the correspondence between visual scenes and sounds, achieving cross-modality scenarios. However, these cross-modality learning methods often presume that audio and visual content in a video are always present and matched, which may not be practical for analyzing and understanding unconstrained videos in the real world.

Audio-visual event localization aims to address this issue by identifying events of interest that have both visible and audible components in a video, as shown in Figure 2.1. This task helps to better understand and analyze real-world videos by simultaneously considering both the visual and auditory aspects of events.

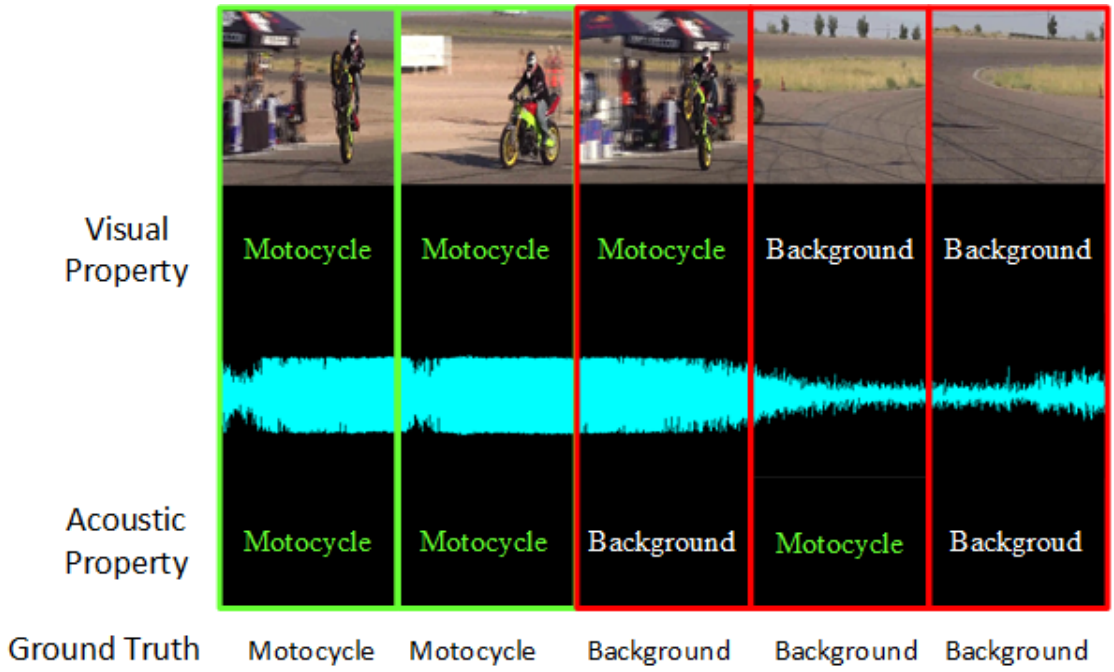


Figure 2.1: Depiction of audio-visual event localization: Observe that the green and red bounding boxes for video or audio data signify the labels observed for single-modality data. Meanwhile, the ground truth labels can be found in the bottom row. It is important to note that non-background event labels are assigned only if cross-modality data displays the same label information.[1]

The Audio-Visual Event (AVE) dataset (described in 4.1.1) is used to systemically investigate the following temporal localization task: supervised audio-visual event localization.

This problem is treated as a sequence labeling problem. The goal of event localization is to predict the event label for each video segment, which contains both audio and visual tracks, for an input video sequence.

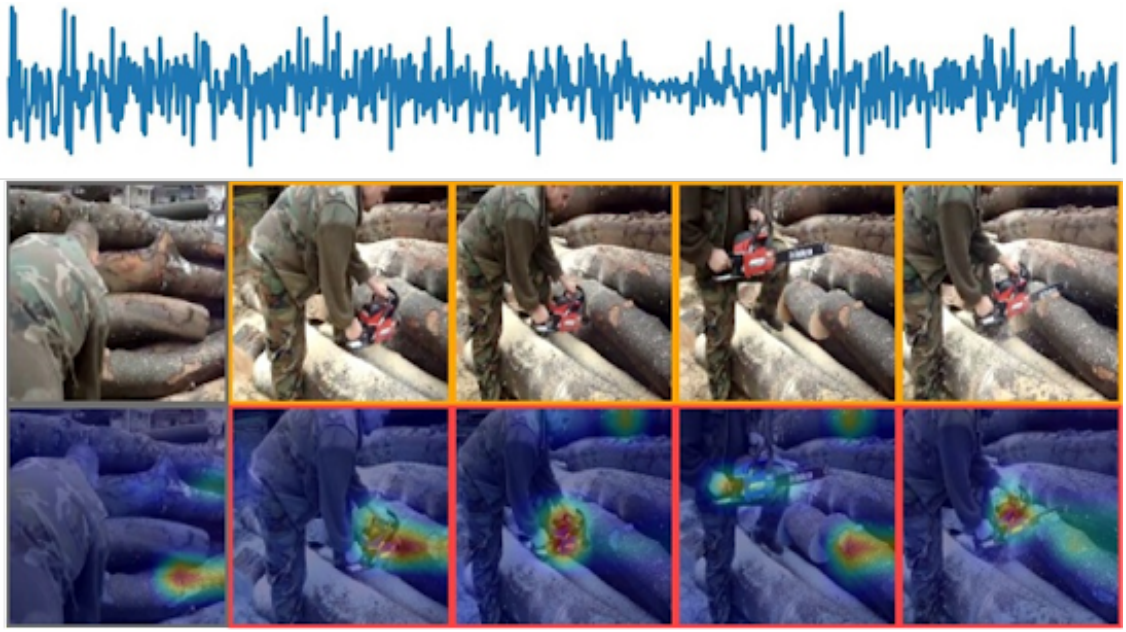


Figure 2.2: Illustration of the audio-visual event localization. [2]

2.2 Audio and Visual module

The audio-visual correlations are explored with several Audio-Visual Event localization networks.

The two deep neural network models that were proposed in [2] for audio-visual event localization, the AV-ATT model and the DMRN model, are used but are modified, so they can be versatile and be used with other feature extraction models.

The AV-ATT model is an attention-based audio-visual fusion model that leverages audio and visual information to localize audio-visual events. The model consists of two branches: an audio branch that processes the audio features and a visual branch that processes the visual features. The outputs of the two branches are then combined using an attention mechanism that takes into account the correlations between the audio and visual information.

The DMRN model is a dual-modality recurrent network that combines audio and visual information in a recurrent manner to localize audio-visual events. The model consists of two parallel recurrent networks: an audio recurrent network and a visual recurrent network. The two networks process the audio and visual features respectively, and the outputs are combined to generate the final prediction.

The network is composed of 5 main modules:

1. feature extraction
2. audio-guided visual attention
3. temporal modeling

4. multimodal fusion

5. temporal labeling

In summary, these five modules work together to perform audio-visual event localization in videos. The feature extraction module extracts the audio and visual features, the audio-guided visual attention module leverages the audio information to guide the visual attention, the temporal modeling module models the temporal dynamics of the features, the multimodal fusion module combines the audio and visual features, and the temporal labeling module predicts the start and end times of the events.

In this diploma thesis, various different models are used for the **feature extraction** module for both the audio and visual part. The specific models used and the way that the feature extraction happens will be described in the following chapters.

Moreover, some different **temporal modeling** techniques will be assessed.

Chapter 3

Theoretical background

3.1 Machine learning and Neural networks

Some core key concepts of neural networks and machine learning are briefly explained.

3.1.1 Neural networks

Neural networks are a type of machine learning model that are designed to recognize patterns in data. They are inspired by the structure and function of the human brain and are capable of learning and generalizing from examples. Neural networks are widely used in a variety of tasks including image and sound recognition. [8]

3.1.2 Transfer Learning

Transfer learning is a commonly used process where a model that has been trained on one problem is utilized for a second, related problem.

In the context of deep learning, transfer learning involves taking a neural network model that has already been trained on a similar problem and using one or more of its layers in a new model that is being trained on a problem of interest.

This technique is often employed in a supervised learning scenario, where the input data remains the same but the target may differ. For instance, the model may be trained on a set of visual categories such as cats and dogs in the first scenario and then trained on a different set of visual categories such as ants and wasps in the second scenario. [8]

3.1.3 Pre-processing

Pre-processing refers to the steps taken to prepare and clean the data before feeding it into a machine learning model. This is an important step in the machine learning pipeline as the quality of the input data has a direct impact on the performance of the model. Pre-processing can involve several tasks such as data cleaning, data normalization, data augmentation, and data transformation.

Data cleaning is the process of removing missing, duplicated, or irrelevant data. This step is necessary to avoid any errors or biases in the data.

Data normalization is the process of scaling the data so that it falls within a specific range. This is important because many machine learning algorithms are sensitive to the scale of the input features and normalizing the data helps to ensure that the features are equally weighted.

Data augmentation is the process of artificially increasing the size of the data set by transforming the existing data in various ways. This can help to improve the performance of the model by providing it with more diverse data.

Data transformation is the process of converting the data into a suitable format for the machine learning model. This can involve feature engineering, dimensionality reduction, and feature scaling.

In summary, pre-processing is an important step in the machine learning pipeline that helps to ensure that the data is clean, normalized, and transformed into a suitable format for the model to effectively learn from it.

3.1.4 Feature extraction

Feature extraction is the process of selecting a set of relevant features from raw data that can be used to train a machine-learning model. In image recognition, features may include the edges, shapes, and textures of objects in the image. In sound recognition, features may include the frequency and time-domain representation of the sound signal.[9]

Furthermore, extracting features can happen by using a pre-trained model. This typically happens by taking the activations of one of the layers of the model, and using these activations as a form of representation for the input data. This is often done when the pre-trained model has already learned useful representations for the input data and these activations can be used as a form of transfer learning.

It is possible to utilize a pre-trained image classification model and extract features from one of its intermediate layers for a different task, such as object detection or segmentation. By utilizing the activations of an intermediate layer, an individual can leverage the knowledge learned by the pre-trained model, avoiding the need for training a large, complex model from scratch.

3.1.5 Embeddings

An embedding is a low-dimensional representation that allows high-dimensional vectors to be transformed into a more manageable form. This is particularly useful in machine learning when working with large input vectors, such as those that represent words in sparse vector form. An effective embedding should preserve the semantic meaning of the input, positioning semantically similar inputs close together in the embedding space. Additionally, embeddings can be learned and utilized across multiple models.

Embeddings can be used as features in machine learning models. An embedding is a learned representation of a high-dimensional input that aims to capture some of its semantic meaning. By transforming the input into a lower-dimensional space, embeddings can be used to extract useful features that can be fed into a machine learning model to make predictions. The embeddings can be learned specifically for the task at hand or pre-trained on a large dataset and then fine-tuned for the specific task.

Audio embeddings and audio features are related concepts, but they serve different purposes and have some key differences.

Audio features are typically hand-crafted or engineered features extracted from audio signals, designed to capture specific characteristics of the signal, such as pitch, tempo, or spectral content. Common audio features include Mel-frequency cepstral coefficients (MFCCs), chroma features, spectral contrast, and zero-crossing rate. These features are often used as inputs to machine learning algorithms for tasks like audio classification, speech recognition, and music analysis.

Audio embeddings are learned representations of audio signals generated by training deep neural networks on large datasets. Audio embeddings are designed to capture high-level and abstract information present in the audio signal, often in a more compact and efficient form. These embeddings can be used as features for various tasks, but they are learned from data rather than being explicitly engineered like traditional audio features.

3.1.6 Dimensionality reduction

Dimensionality reduction is the process of reducing the number of variables or features in a dataset while preserving as much relevant information as possible. It is widely used in data analysis, machine learning, and statistics to improve computational efficiency, reduce noise, and facilitate visualization or interpretation of high-dimensional data. Various methods can be employed for dimensionality reduction, such as calculating the mean of an array, PCA, and other techniques. These methods are based on general principles found in machine learning and statistics literature, such as "Pattern Recognition and Machine Learning" by Christopher M. Bishop [10] and "The Elements of Statistical Learning" by Trevor Hastie, et al [11].

(a) **Calculating the mean**

One simple approach to dimensionality reduction is to calculate the mean of an array, which reduces the dimensionality to a single value. This method can be useful when the primary interest is in understanding the central tendency of the data rather than the detailed variations in the data. However, this approach can result in a significant loss of information, as the mean only captures the data's average value and discards any variations or patterns in the data.

(b) **PCA (Principal Component Analysis)**

PCA is a widely used linear dimensionality reduction technique that projects the original data onto a lower-dimensional space while retaining as much of the data's variance as possible. It involves computing the eigenvectors and eigenvalues of the data's covariance matrix and selecting the principal components corresponding to the highest eigenvalues. PCA can be useful for revealing patterns and structure in high-dimensional data, improving computational efficiency, and reducing noise. It is well-suited for situations where the data's relevant information is contained in the principal components, but it may not perform well when the data has a non-linear structure. Further information about PCA can be found at chapter 3.1.7

(c) **Other methods**

There are several other dimensionality reduction techniques that can be employed depending on the problem's specific characteristics and requirements, such as t-distributed Stochastic Neighbor Embedding (t-SNE), Linear Discriminant Analysis (LDA) and Autoencoders. In this diploma thesis, only the first two methods (a), (b) have been used, so there will be no further analysis of the other methods.

These methods provide different trade-offs in terms of computational complexity, interpretability, and ability to capture the underlying structure of the data. The choice of an appropriate dimensionality reduction technique depends on the specific goals and requirements of the analysis.

3.1.7 Principal Component Analysis (PCA)

Principal Components Analysis (PCA) is a commonly used unsupervised technique for reducing the dimensionality of data. It is a way of constructing relevant features or variables from the original data through linear (linear PCA) or non-linear (kernel PCA) combinations. This article will focus specifically on the popular and widely used linear PCA method.

The goal of PCA is to linearly transform correlated variables into a smaller set of uncorrelated variables. This is accomplished by projecting the original data onto a reduced space using the eigenvectors of the covariance or correlation matrix, also known as the principal components (PCs).

The projected data is a linear combination of the original data and captures most of the variance in the data (Jolliffe [12]). In essence, PCA is an orthogonal transformation of the data into a series of uncorrelated data in the reduced space, where the first component explains the most variance, and each subsequent component explains less.

3.1.8 Principal Component Analysis (PCA) Application

As previously stated in 3.1.7 Principal Components Analysis (PCA) is a commonly used for reducing the dimensionality of data.

Performing PCA (Principal Component Analysis) on audio or visual features can be done in different ways depending on the goal and the structure of the data. Here two scenarios are described.

1. **Performing PCA on each video's audio or visual features separately**

   In this scenario, the analyst applies PCA to the audio (or visual) features of each video individually [10]. The motivation behind this approach is to analyze or visualize the features of each video separately, which might be useful if the audio (or visual) characteristics of the videos are significantly different from each other. By applying PCA separately for each video, the information from different videos is not mixed. However, one potential drawback of this approach is that the principal components for different videos might not be aligned, which can make it challenging to compare the results across videos directly.

2. **Performing PCA on the entire audio or visual features array for all videos**

   In this scenario, the analyst concatenates the features of all videos and applies PCA on the combined dataset. This approach aims to analyze the common patterns or variations across all the videos. By applying PCA on the entire dataset, the principal components will be aligned across videos, which facilitates the comparison of results directly. However, it is crucial to ensure that the data's structure is consistent and that there's no unintended mixing of information from different videos.

3.1.9 Sequence labeling

The sequence labeling problem involves assigning a label to each element in a sequence of inputs. This can be useful for solving a wide range of tasks, such as named entity recognition, part-of-speech tagging, and sentiment analysis.

For example, in the named entity recognition task, the goal is to identify and categorize named entities, such as people, organizations, and locations, in a given sentence or document. A typical sequence labeling approach would involve first extracting features from the input sequence and then using these features to make predictions about the labels of each element in the sequence.

In the paper "Sequence to Sequence Learning with Neural Networks" [13] the authors explore the use of neural networks for solving sequence-to-sequence problems, including sequence labeling. They present an overview of the different types of neural networks that can be used for this task, including recurrent neural networks (RNNs) and long short-term memory (LSTM) networks. They also discuss the challenges of training these networks and present some of the latest techniques for improving their performance. [13]

3.1.10 Recurrent neural networks

Recurrent Neural Networks (RNNs) are popular models used in the field of deep learning for sequential data processing.

RNNs are a type of neural network that are designed to handle sequences of data, such as time series, text, and speech. They have an internal memory that allows them to maintain information about the sequence over time. This allows them to learn patterns and relationships in sequences of data that might not be apparent using traditional feedforward neural networks. They have been widely used in various applications, including speech recognition, machine translation, and sentiment analysis.

3.1.10.1 Long Short-Term Memory Networks (LSTMs)

LSTMs are a type of RNN that are specifically designed to handle long-term dependencies in sequences. They do this by using gates to control the flow of information into and out of the internal memory, allowing them to keep information about the sequence for a longer period of time and avoid the vanishing gradient problem that can occur in traditional RNNs.

Specifically, LSTM, introduced by Hochreiter and Schmidhuber in 1997 [14], is an RNN architecture with a memory cell and three gates (input, forget, and output) that control the flow of information within the network. The memory cell stores the internal state, while the gates regulate the information flow, allowing the network to learn long-range dependencies by selectively remembering and forgetting information.

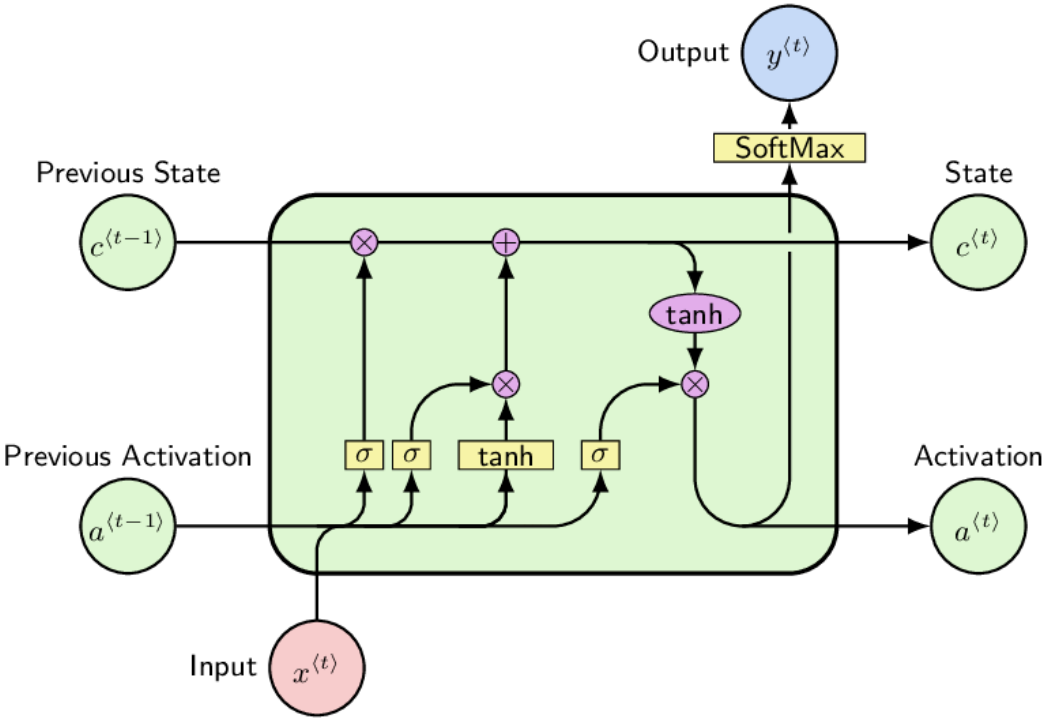


Figure 3.1: The LSTM cell internals. [3]

3.1.10.2 Gated Recurrent Unit (GRU)

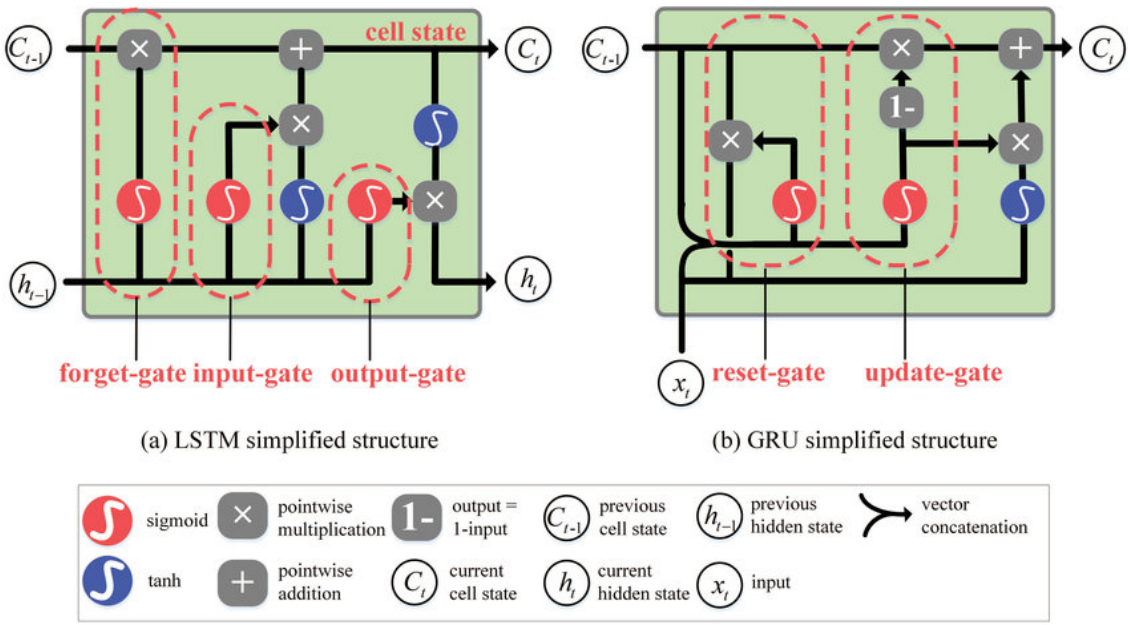
The Gated Recurrent Unit (GRU) is a type of Recurrent Neural Network (RNN) that was introduced in 2014 by Cho et al. at the paper "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches" [15]. The GRU model is designed to handle the vanishing gradient problem that is often encountered when training RNNs.

The GRU model uses gating mechanism to control the flow of information through the network, which allows it to effectively capture long-term dependencies in the input data. The gating mechanism in the GRU model consists of two gates - the reset gate and the update gate. These gates determine the extent to which the hidden state of the network should be updated based on the current input and the previous hidden state.

GRU, is a simplified version of LSTM with fewer parameters. It combines the forget and input gates into a single "update" gate and merges the cell state and hidden state. GRUs have fewer weights and are computationally more efficient than LSTMs, while still effectively capturing long-range dependencies in sequences.

GRUs have been widely used in various sequence modeling tasks, including language modeling, machine translation, and speech recognition. They have been shown to perform well in comparison to traditional RNNs and LSTMs while being computationally more efficient.

More information about the difference between LSTMs and GRUs can be found in [4]



(a) LSTM simplified structure (b) GRU simplified structure

Figure 3.2: LSTM and GRU simplified. [4]

3.1.10.3 Differences and advantages between the two networks

1. **Complexity**

   GRUs have fewer parameters and are computationally more efficient than LSTMs,

making them faster to train and less prone to overfitting on smaller datasets.

2. **Performance**

   LSTMs and GRUs can exhibit similar performance on various tasks, but LSTMs might have a slight edge when capturing very long-range dependencies due to their more complex gating mechanism.

3. **Ease of implementation**

   GRUs are simpler to implement and understand compared to LSTMs, which can be an advantage for researchers and practitioners looking to build custom models or adapt existing ones.

3.1.11 Convolutional Neural Networks (CNN/ConvNet)

Convolutional Neural Networks (CNNs) are a type of deep learning neural network that have been specifically designed for image recognition tasks. They are called ”convolutional” because they use a mathematical operation called convolution to process the input data. The convolution operation involves taking small pieces of the input image, called kernels or filters, and using them to scan the image, producing a new image where each pixel is a function of the values in the original image within the kernel. The result is a filtered version of the original image that can highlight certain features or patterns. [16]

In a CNN, these convolutional layers are stacked on top of each other, each one processing the output of the previous layer. The filtered images produced by the convolutional layers are then processed by additional layers, such as pooling layers and fully connected layers, which help to extract and refine the features of the image.

One of the key strengths of CNNs is their ability to learn hierarchical representations of images. They can learn low-level features, such as edges and textures, and then use these features to learn higher-level features, such as shapes and objects. This hierarchical representation is what makes CNNs so powerful for image recognition tasks.

Overall, CNNs have been widely used in various applications, including image classification, object detection, semantic segmentation, and image generation.

3.2 Multimodal machine learning

The goal of multimodal machine learning is to study and learn the joint representations of multiple input modalities, such as speech and video, image and text. The fusion of features is a critical aspect of multimodal learning, and various fusion models have been proposed, including statistical models, Multiple Kernel Learning, and Graphical models. Despite some research on multimodal deep networks, particularly focusing on joint audio-visual representation learning through Autoencoder or deep Boltzmann machines, the authors of this paper aim to explore the best models to combine audio and visual features for the purpose of localization.

3.3 Audio module

Audio refers to the sound signals that are used to convey information or create music. The term is most often used to describe signals that are recorded or transmitted for the purpose of producing sound through speakers or headphones. Audio signals can be represented in a variety of formats, including analog and digital signals, and can be processed using a wide range of techniques to produce desired effects or enhance the quality of the sound.

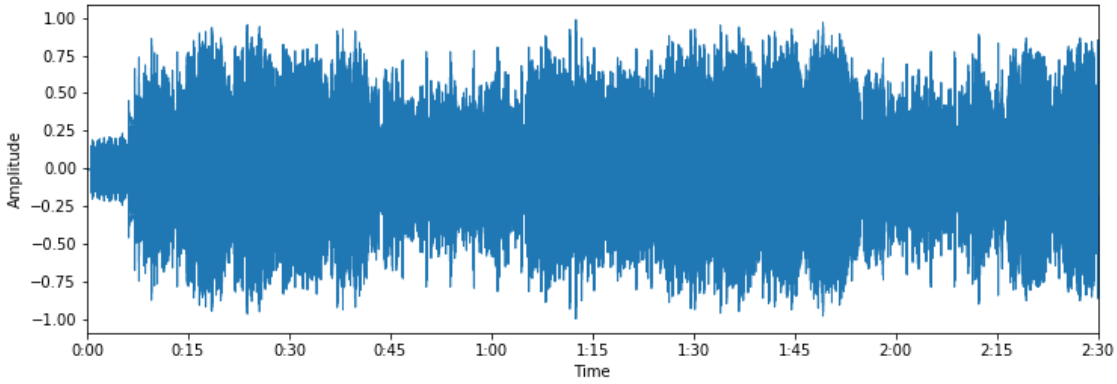


Figure 3.3: Audio signal of a musical track [5]

3.3.1 Representations of audio signals

Spectrograms, Mel spectrograms, Log-Mel spectrograms, and MFCCs are different representations of audio signals that capture different aspects of the spectral and temporal information. They are used as features for various audio processing and machine learning tasks, depending on the specific requirements and constraints of the problem at hand.

1. **Spectrogram**

   A spectrogram is a visual representation of the frequency content of a signal as it varies over time. It is obtained by computing the Short-Time Fourier Transform (STFT) of the input signal. The horizontal axis represents time, the vertical axis represents frequency, and the color intensity at each point corresponds to the magnitude of the signal at that time and frequency. Spectrograms are commonly used in audio processing for tasks such as speech recognition, music analysis, and environmental sound classification.

2. **Mel Spectrogram**

   A Mel spectrogram is a spectrogram where the frequency axis is scaled according to the Mel scale, which approximates the human auditory system's response to different frequencies. The Mel scale is a nonlinear transformation of the frequency axis that emphasizes lower frequencies, which are more relevant for human perception. Mel

spectrograms are used as features for various audio processing tasks, as they provide a more perceptually meaningful representation of the audio signal.

3. **Log-Mel Spectrogram**

   A Log-Mel spectrogram is a Mel spectrogram where the magnitude values are transformed using a logarithmic function. This transformation is applied to mimic the human auditory system's logarithmic perception of loudness. The Log-Mel spectrogram is a common feature representation for speech and audio recognition tasks, as it captures both the spectral and temporal characteristics of the audio signal in a compact and perceptually meaningful way.

4. **Mel-Frequency Cepstral Coefficients (MFCCs)**

   MFCCs are a compact representation of the spectral envelope of an audio signal. They are obtained by computing the Discrete Cosine Transform (DCT) of the log-Mel spectrogram. MFCCs capture the coarse spectral shape of the audio signal while discarding high-frequency details, making them robust to noise and small variations in the signal. MFCCs have been widely used as features for speech and speaker recognition, as well as other audio processing tasks.

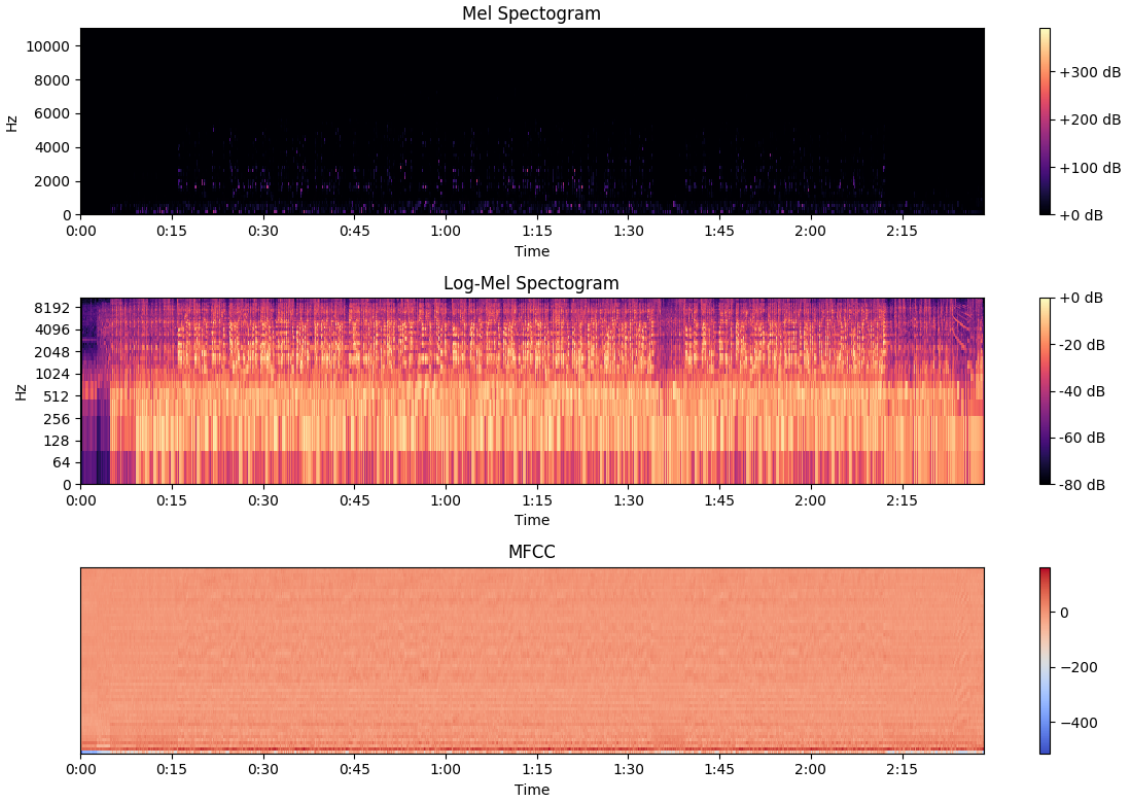


Figure 3.4: Mel Spectrogram, Log-Mel Spectrogram, MFCC [5]

3.3.2 Audio used in machine learning

Audio signals are used in a variety of applications of audio processing using machine learning in the field of music and audio analysis. [17] Some of these include:

- Music Information Retrieval (MIR): This field involves the automatic extraction of information from music and audio signals, such as genre classification, mood classification [18, 5], and content-based retrieval of music.
- Speech Processing: This field deals with the processing of speech signals, such as speech recognition, speaker recognition, and language identification.
- Audio Source Separation: This involves separating audio signals into their constituent sources, such as separating vocals from music in a song, or separating different speakers in a multi-speaker recording.
- Music Generation: This involves generating new music or audio signals using machine learning algorithms, such as generative models or recurrent neural networks.
- Music Transcription: This involves transcribing audio signals into symbolic representations, such as sheet music or MIDI files.

3.3.3 Audio neural networks

Audio Neural Network models are a type of machine learning models that are specifically designed to process audio signals. These models are based on deep learning techniques and are capable of extracting meaningful representations of audio signals in an unsupervised manner. The main goal of these models is to automatically learn high-level representations of audio signals that can be used for various applications such as speech recognition, audio classification, music information retrieval, and sound event detection.

One of the most commonly used audio neural network models is Convolutional Neural Networks (CNNs). CNNs have been used for tasks such as audio classification and sound event detection. They are trained to learn filters that can extract features from raw audio signals and classify them into different categories.

For example, a very popular convolutisonal neural network is the VGG, which stands for ”Very Deep Convolutional Networks for Large-Scale Image Recognition,”. VGG is a deep convolutional neural network architecture that was introduced in the 2014 paper ”Very Deep Convolutional Networks for Large-Scale Image Recognition” by Karen Simonyan and Andrew Zisserman [19]. The architecture consists of multiple convolutional layers, followed by a series of fully connected layers, and is designed for large-scale image recognition. The architecture is named VGG due to its use of multiple convolutional layers and its ability to learn very deep representations of the input image data. The VGGish model is a variant of the VGG architecture specifically designed for audio data. The VGGish model takes log mel spectrogram audio inputs and produces 128-dimensional

embeddings of each audio segment. These embeddings can then be used as input to other models for various audio-related tasks such as event classification or segmentation.

Another type of audio neural network models are Recurrent Neural Networks (RNNs), which have been used for tasks such as speech recognition and audio-to-text conversion. RNNs can handle sequential data, which makes them ideal for processing audio signals. In particular, the Long Short-Term Memory (LSTM) network, a type of RNN, has been used for speech recognition tasks due to its ability to handle long sequences of data and its ability to capture dependencies between audio frames.

Another popular audio neural network model is the Wav2Vec model, which is a self-supervised audio representation learning model. Wav2Vec can be used for various audio-related tasks such as speaker recognition, language identification, and acoustic event detection.

There are various other audio neural network models that have been proposed and used for different applications. Each model has its own strengths and weaknesses and the choice of the model depends on the specific task and the data available for training. [8]

Bellow several models for audio tasks are explained that will be used in this diploma thesis.

3.3.4 VGGish

VGGish is a modification of the VGG model[19], particularly Configuration A with 11 weight layers. The changes made to the model include a change in the input size to 96x64 to accommodate log mel spectrogram audio inputs. The last group of convolutional and max pooling layers have been dropped, reducing the number of groups to four. The final layer, which was a 1000-wide fully connected layer, has been replaced with a 128-wide fully connected layer to act as a compact embedding layer. The model definition provided only includes the layers up to the 128-wide embedding layer, which does not include a final non-linear activation. When using this model for training, it is necessary to add a non-linearity before adding more layers.

The original AudioSet release by Gemmeke et al. in 2017 [20] included 128-dimensional embeddings for each AudioSet segment that were generated from a VGG-style audio classification model. This model was trained on a vast dataset from YouTube, which was later referred to as YouTube-8M).

A TensorFlow definition of this model is provided, which is called VGGish [21], as well as supporting code to extract input features for the model from audio waveforms and to post-process the model embedding output into the same format as the released embedding features.

VGGish can serve a purpose as a feature extractor in audio processing: It transforms audio input into a compact, semantically rich 128-dimensional embedding, which can then be utilized as input to a subsequent classification model. The advantage of using VGGish is that the subsequent model can be relatively shallow, as the VGGish embedding provides

a more condensed semantic representation than raw audio features.

3.3.5 Wav2vec

Wav2vec [22] proposes a new method for unsupervised pre-training of speech recognition models using raw audio data. The method is based on a self-supervised learning approach that learns to predict a set of masked features from the input waveform.

Previous approaches to pre-training speech recognition models have relied on supervised training, where the model is trained on labeled speech data. However, labeled data is often expensive and time-consuming to collect, limiting the scalability of these approaches. In contrast, the Wav2vec method aims to leverage large amounts of unlabeled speech data to improve speech recognition performance.

The core idea behind Wav2vec is to pre-train a model on a large dataset of unlabelled speech data in a self-supervised way. The model learns to predict masked features from the input waveform, where the masked features are a subset of the original features that are randomly masked during training. The model is then fine-tuned on labeled data for the speech recognition task.

The Wav2vec model was trained on the 960 hours of training data from the LibriSpeech dataset [23], described in 4.1.3, which consists of over 1,000 speakers and a wide range of acoustic conditions. The model was trained in a self-supervised way, using a masked prediction task, to learn fixed-length representations of the speech signal that are suitable for downstream speech recognition tasks.

3.3.6 vq-Wav2vec

VQ-Wav2Vec [24] proposes a new self-supervised learning method for learning discrete representations of speech signals. The method is an extension of the original Wav2vec [22] method and uses vector quantization to discretize the continuous speech representations learned by the model.

Previous methods for speech recognition have relied on hand-crafted feature extraction and modeling techniques, which can be time-consuming and difficult to optimize. In contrast, the vq-wav2vec method learns representations directly from the raw audio signal in a self-supervised way.

The core idea behind the vq-wav2vec method is to use a vector quantization algorithm to discretize the continuous speech representations learned by the model into a finite set of discrete codes. These codes can be used as input features for downstream speech recognition models.

The authors of [24] evaluate the vq-wav2vec method on several benchmark datasets and show that it outperforms previous methods for unsupervised speech representation learning. They also demonstrate that the method is robust to variations in audio quality and can be used to learn representations for multilingual speech recognition.

3.3.6.1 VQ-Wav2vec kmeans

VQ-Wav2vec_kmeans [24] builds upon the original VQ-Wav2vec [24] method by incorporating a k-means clustering algorithm to discretize the continuous speech representations learned by the model. This is done in order to further reduce the memory requirements for storing the representations, as well as to make them more interpretable.

3.3.7 Wav2vec 2.0

Wav2Vec 2.0 [25] is a self-supervised learning framework for speech processing developed by Facebook AI Research (FAIR). It is a continuation of the original Wav2Vec framework, which used self-supervised learning to learn speech representations.

The main idea behind Wav2Vec 2.0 is to use a transformer-based model to learn contextualized representations of speech. The model is trained using a self-supervised learning task called contrastive predictive coding (CPC), which involves predicting future audio samples given past samples. This task is designed to encourage the model to learn high-level features of speech, such as phonemes and words, that can be used for downstream tasks.

The Wav2Vec 2.0 framework has achieved state-of-the-art performance on several speech processing tasks, including speech recognition and speaker verification. It has also been used to pretrain speech models for downstream tasks, such as automatic speech recognition and speaker recognition. [25]

3.3.7.1 Wav2vec Vox New

Wav2vec Vox New [25] builds upon the original Wav2vec 2.0 [25] method by incorporating an additional training objective that encourages the model to learn speaker-discriminative representations. The method adds a new contrastive loss term to the original masked prediction loss used in Wav2vec 2.0. This new loss term encourages the model to learn speech representations that are discriminative across different speakers, in addition to being discriminative across different time steps within the same audio clip.

3.3.7.2 Wav2vec and wav2vec 2.0

Wav2vec [22] and wav2vec 2.0 [25] are two different self-supervised learning methods for speech recognition developed by Facebook AI Research. The two methods differ in their approach to feature extraction and modeling.

The main difference between wav2vec and wav2vec 2.0 is the type of neural network used for feature extraction. Wav2vec uses CNNs to learn fixed-length representations, while wav2vec 2.0 uses transformers to learn variable-length representations. The two methods have achieved state-of-the-art results on various speech-related tasks and have contributed to advancing the field of speech recognition and natural language processing.

3.3.8 Musicnn

Musicnn, pronounced as ”musician”, is a collection of pre-trained convolutional neural networks for music audio tagging that are designed with musical considerations. The repository also offers pre-trained vgg-style models. [26]

These pre-trained deep convolutional neural networks for music audio tagging are trained with two different datasets: the MagnaTagATune dataset (the MTT of 19k training songs) and the Million Song Dataset (the MSD of 200k training songs).

Musicnn is a musically motivated convolutional neural network (CNN) trained for music audio tagging. It consists of a CNN front-end that is musically motivated, of a densely connected mid-end, and a temporal-pooling back-end is employed for the output layers.

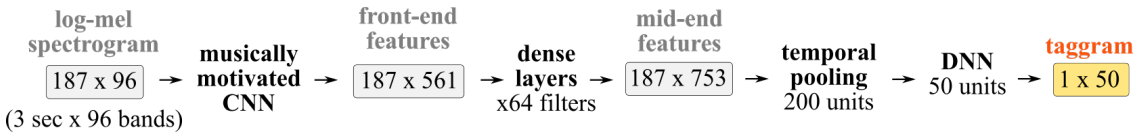


Figure 3.5: Musicnn neural network [6]

Vgg is a computer vision baseline model that we trained for music audio tagging. This naive adaption of a vision CNN for audio-spectrograms stacks several 3x3 CNN layers with max pooling.

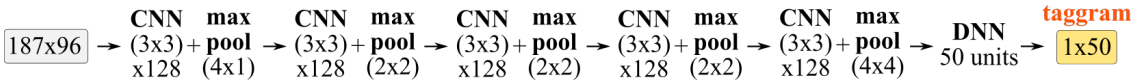


Figure 3.6: Vgg neural network [6]

3.3.9 Yamnet

YAMNet is a deep net that is trained on the AudioSet dataset, which contains millions of 10-second audio clips from YouTube videos. AudioSet is a large-scale dataset of manually annotated audio events covering a wide range of sounds, including human sounds, animal sounds, and environmental sounds. YAMNet is designed to predict audio events from the AudioSet ontology, which consists of a hierarchical structure of 527 audio event classes.

YAMNet is built using the MobileNet_v1 depthwise-separable convolution architecture. The model is designed to be lightweight and efficient, making it suitable for deployment on mobile devices or other resource-constrained environments. YAMNet is capable of providing high-quality audio event classification while maintaining low computational requirements.

YAMNet's audio event classifier is trained using log-mel spectrogram features, which are extracted from the audio clips in the AudioSet dataset. These features are then used

as input to the YAMNet model for audio event prediction. The YAMNet model has been shown to achieve state-of-the-art performance in audio event classification tasks.

3.3.10 Openl3

OpenL3 is an audio (and image) embedding library developed by the research team at OpenAI. It is a deep neural network trained to generate embeddings (a compact, low-dimensional representation) of audio recordings that capture the characteristics of the audio signal. OpenL3 is designed to work with a wide range of audio sources, including music and speech, and can be used in various audio-related applications, such as music classification, content-based retrieval, and recommendation systems.

One of the key features of OpenL3 is its ability to generalize to unseen audio data. This means that it can produce embeddings for new audio recordings that are similar to those it has seen in the training data, even if the new recordings are from different domains or are of different types of audio. This makes OpenL3 well-suited for use in real-world applications where the data is constantly changing.

OpenL3 is also designed to be fast and efficient, making it suitable for use on a wide range of devices, from laptops to mobile devices. It is available as an open-source library, allowing developers to easily integrate it into their audio-related projects.

The audio and image embedding models provided in Openl3 are published as part of the Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings [27] by Cramer et al. and are based on the Look, Listen and Learn approach by Arandjelovic et al. [28]. More details about the embedding models and how they were trained are in Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings [27] by Cramer et al

3.4 Visual module

Visual refers to anything related to the sense of sight or the perception of images. In the context of computer vision and digital processing, visual usually refers to the representation and manipulation of images or videos, which are essentially two-dimensional (2D) or three-dimensional (3D) arrays of pixel values.

In digital representation, images and videos are represented as arrays of discrete pixel values. Each pixel corresponds to a particular point in the image or video frame and has a specific value or set of values that represent its color or intensity.

For grayscale images, each pixel is represented by a single value, usually an integer ranging from 0 to 255, where 0 represents black and 255 represents white. In the case of color images, each pixel is typically represented by a set of three values, corresponding to the red (R), green (G), and blue (B) channels of the image. Each channel has a value range of 0 to 255, where higher values indicate higher intensities of the respective color component. This representation is called the RGB color model.

An example of an image of dimension 10x5 of pixels and their RGB representation can be found in Figure 3.7.

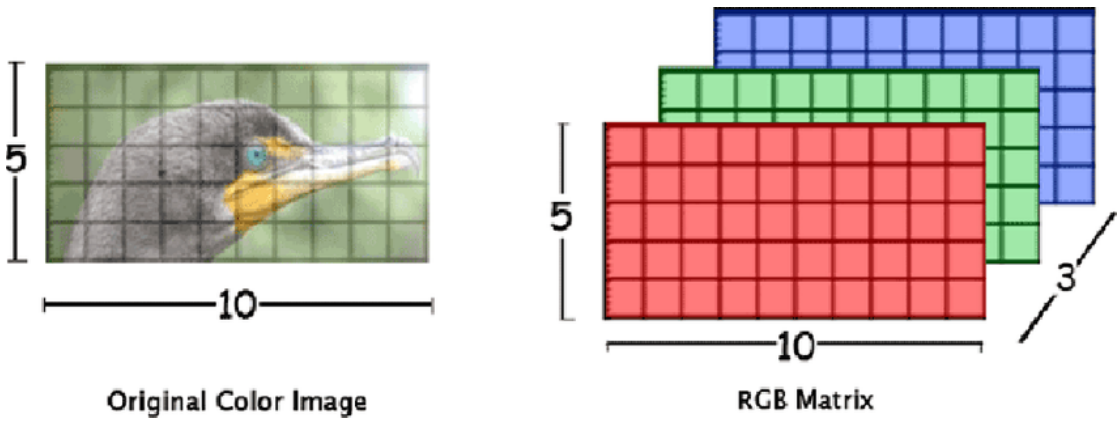


Figure 3.7: Color image representation and RGB matrix [7]

For videos, the visual representation is similar to that of images, but with an additional dimension for time. A video is a sequence of image frames, where each frame is represented as a 2D array of pixel values (grayscale or color). Videos can be represented in various formats and compression schemes to optimize storage and transmission requirements.

In summary, visual representations in digital form involve discretizing the continuous visual world into arrays of pixel values, which can then be processed and analyzed by computer algorithms, including machine learning and neural networks.

More information about the digital representation of images and videos can be found in [29, 30, 31].

3.4.1 Visual tasks

Visual tasks in machine learning and neural networks involve the processing and understanding of visual data, such as images or videos, to perform various tasks. Some common visual tasks include:

Image classification: The goal of image classification is to assign an input image to one of several predefined categories or classes. This is a fundamental task in computer vision and serves as a building block for many other tasks. A popular deep learning model for image classification is the Convolutional Neural Network (CNN) (LeCun et al., 1998) [32].

Object detection: Object detection involves not only classifying objects within an image but also determining their location by providing bounding boxes around each detected object. Models like R-CNN (Girshick et al., 2014) [33], Faster R-CNN (Ren et al., 2015) [34], and YOLO (Redmon et al., 2016) [35] are well-known for object detection.

Semantic segmentation: In semantic segmentation, the objective is to assign a class label to each pixel in the image. This allows for a more detailed understanding of the image content compared to image classification. Models like Fully Convolutional Networks (FCN) (Long et al., 2015) [36] and DeepLab (Chen et al., 2018) [37] are widely used for semantic segmentation.

Instance segmentation: Instance segmentation goes a step further than semantic segmentation by differentiating between individual instances of the same class within an image. Models like Mask R-CNN (He et al., 2017) [38] are popular for instance segmentation tasks.

Image generation: Image generation tasks involve the creation of new, realistic images based on certain conditions, such as generating images conditioned on text descriptions or other images. Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) [39] and Variational Autoencoders (VAEs) (Kingma and Welling, 2013) [40] are widely used for image generation.

These tasks represent just a few examples of the many visual tasks addressed in machine learning and neural networks. The success of deep learning models in tackling these problems has led to significant advancements in computer vision and its applications across various domains.

Some domains are autonomous vehicles, medical imaging (such as ”A Deep Neural Architecture for Harmonizing 3-D Input Data Analysis and Decision Making in Medical Imaging” by Kollias et. al [41] and ”AI-MIA: COVID-19 Detection and Severity Analysis through Medical Imaging” by Kollias et. al [42]), surveillance, facial expressions detection (such as FaceRNET: a Facial Expression Intensity Estimation Network by Kollias et al. [43] and Affect Analysis in-the-wild: Valence-Arousal, Expressions, Action Units and a Unified Framework [44]) and multimedia content analysis.

3.4.2 Segmentation

Segmentation models are a class of deep learning models designed to perform image segmentation tasks. Image segmentation is the process of partitioning an image into multiple segments, where each segment corresponds to an object or a region of interest. The goal is to assign a label to every pixel in the image, such that pixels with the same label belong to the same object or region.

There are two main types of image segmentation:

- **Semantic segmentation**

  In this task, the model assigns a class label to each pixel in the image, grouping all pixels belonging to the same object class. However, it does not differentiate between individual instances of the same class. For example, in an image containing multiple cars, all car pixels would be labeled as "car" without distinguishing between the different cars.

- **Instance segmentation**

  In this task, the model not only assigns a class label to each pixel but also differentiates between individual instances of the same class. For instance, in an image with multiple cars, each car would be assigned a unique label, allowing the model to distinguish between them.

Class query logits and mask query logits are terms that are often used in the context of segmentation models, particularly those based on transformers.

Class query logits:

These refer to the logits generated by the model for predicting the class or category of each object or region in an image. These logits are usually passed through a softmax activation function to generate probabilities for each class, and the class with the highest probability is then assigned to the object or region.

Mask query logits:

These logits are associated with predicting the binary mask for each object or region in an image, indicating which pixels belong to the object and which do not. Mask query logits are usually passed through a sigmoid activation function to generate probabilities for each pixel belonging to the object, and a threshold is applied to convert these probabilities into binary values.

3.5 Visual neural networks

Visual neural networks are a type of deep learning model specifically designed to process and analyze visual data such as images and videos. These networks can learn to recognize patterns and features in visual data by automatically extracting hierarchical features through their layers. Visual neural networks have been highly successful in a

wide range of computer vision tasks, such as image classification, object detection, and semantic segmentation.

3.5.1 VGG-19

VGG-19 is a specific deep convolutional neural network architecture introduced by Simonyan and Zisserman [19].

VGG-19 has 19 layers, including 16 convolutional layers, 3 fully connected layers, and 5 max-pooling layers. VGG-19 is known for its deep architecture and has achieved excellent performance in various computer vision tasks, including the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

One of the key aspects of the VGG-19 architecture is its use of small 3x3 convolutional filters throughout the network, which allows for the efficient capturing of local spatial information in images. Although VGG-19 is highly accurate, it is also computationally intensive due to its depth and large number of parameters. As a result, more recent architectures, such as ResNet and EfficientNet, have been developed to provide similar or better performance with fewer parameters and reduced computational requirements.

3.5.2 Xception

Xception is a deep convolutional neural network architecture that replaces the standard Inception modules with depthwise separable convolutions. It aims to improve the efficiency and performance of the network. More information about Xception can be found in [45]

3.5.3 ResNet50 and ResNet152

ResNet (Residual Network) is a family of deep convolutional neural networks known for their ability to effectively handle very deep architectures by introducing skip connections or residual connections. ResNet50 has 50 layers, while ResNet152 has 152 layers. [46]

3.5.4 Inceptionv3

InceptionV3 is an improved version of the Inception architecture that introduces various optimizations, such as factorized convolutions and label smoothing, to improve the network's performance and efficiency. [47]

3.5.5 MobileNet

MobileNet is a family of efficient convolutional neural networks designed for mobile and embedded vision applications. It uses depthwise separable convolutions to reduce the number of parameters and computational complexity. [48]

3.5.6 Densenet201

DenseNet (Densely Connected Convolutional Network) is a deep convolutional neural network architecture that connects each layer to every other layer in a feed-forward fashion, improving gradient flow and encouraging feature reuse. [49]

3.5.7 Nasnetlarge

NASNet (Neural Architecture Search Network) is a family of convolutional neural networks discovered through a search algorithm that automatically learns to design network architectures. NASNetLarge is a larger variant optimized for better accuracy. [50]

3.5.8 EfficientNetB7

EfficientNet is a family of convolutional neural networks that introduce a systematic approach to scaling up network width, depth, and resolution. EfficientNetB7 is one of the larger variants with better accuracy. [51]

3.5.9 EfficientNetV2l

EfficientNetV2 is an improved version of the EfficientNet family, which introduces various optimizations, such as Fused-MBConv and progressive learning, to achieve better performance and efficiency. [52]

3.5.10 ConvNextXlarge

The dawn of the "Roaring 20s" in visual recognition was marked by the emergence of Vision Transformers (ViTs), which quickly outpaced ConvNets (3.1.11) to become the leading image classification model. However, applying a basic ViT to general computer vision tasks, such as object detection and semantic segmentation, poses certain challenges. It is through hierarchical Transformers, like Swin Transformers, that several ConvNet priors were reintroduced, enabling Transformers to serve as a versatile vision backbone and exhibit impressive performance in a broad range of vision tasks. Despite this, the success of these hybrid approaches is mostly credited to the inherent advantages of Transformers, rather than the innate inductive biases of convolutions. In this research, the authors reevaluate the design spaces and explore the potential of a pure ConvNet. By incrementally updating a standard ResNet towards the design of a vision Transformer, they uncover key components responsible for the performance disparity. This investigation leads to the development of a series of pure ConvNet models, named ConvNeXt. Composed entirely of conventional ConvNet modules, ConvNeXts compete with Transformers regarding accuracy and scalability, achieving an 87.8% ImageNet top-1 accuracy and surpassing Swin Transformers in COCO detection and ADE20K segmentation, all while maintaining the simplicity and efficiency inherent to standard ConvNets. [53]

3.5.11 Maskformer

MaskFormer is a deep-learning model for semantic segmentation tasks. It was introduced in the paper "Per-Pixel Classification is Not All You Need for Semantic Segmentation" by Cheng et al [54]. The model employs a transformer-based architecture to predict per-pixel masks for objects or regions in an image.

In contrast to previous methods, which relied on per-pixel classification, MaskFormer predicts masks for each object or region in a more efficient manner. The model consists of two main components: a class query module and a mask query module. The class query module predicts the class label for each object or region in the image, while the mask query module predicts the corresponding masks.

The architecture uses a single transformer-based backbone, which makes the model more efficient compared to other methods that require separate branches for different tasks.

Part II

Practical part

Chapter 4

Data

This chapter presents the sets of data used in this work, for model training and testing and also describes briefly the datasets that were used on the pre-trained models that were used as feature extractors. The AVE dataset 4.1.1 was used for model training and testing the Audio model, the Visual model, and the Audio-Visual models.

4.1 Datasets

4.1.1 AVE: The Audio-Visual Event Dataset

The Audio-Visual Event (AVE) dataset introduced in the Audio-Visual Event Localization in Unconstrained Videos paper [2] is suitable for our purpose of Audio-Visual event localization.

The AVE dataset is a compilation of 4143 videos selected from AudioSet [20], each showcasing one of 28 different event categories. These videos have been annotated with the temporal boundaries of audio-visual events, with each event lasting a minimum of 2 seconds. The start and the end time of each audio-visual event has been annotated, marking the temporal boundaries with a resolution of 1 second.

bell	man	dog	plane	car	woman	copt.	violin	flute	ukul.	frying	truck	shofar	moto.
188	176	171	184	188	175	178	187	172	163	182	138	102	99
guitar	train	clock	banjo	goat	baby	bus	chain.	cat	horse	toilet	rodent	acco.	mand.
186	176	160	183	100	83	69	160	60	64	180	109	154	156

Table 4.1: *The 28 event categories of the AVE dataset along with their number of videos*

So there is at least a 2-second-long segment, where both the sound source and the sound are clearly visible and audible. AVE encompasses a broad array of audio-visual events, including human speech, animal sounds, musical performances, and vehicle noises, among others, and draws from a diverse range of domains, including human activities, animal activities, music, and vehicles. A visualization of the data and its statistics can be found in Figure 4.1. The number of videos for each event category varies, with the

minimum being 60 and the maximum being 188 (as seen in table 4.1), and over two-thirds (66.4%) of the videos in the AVE include audio-visual events that span the full 10-second duration.

The videos are unconstrained because videos from AudioSet [20] originated from Youtube and have diverse editing artifacts and significant complexity of content.

Videos in AVE dataset are divided into training (3339), validation (402), and testing (402) sets. For supervised audio-visual event localization task, randomly sample videos are selected from each event category to build the train/val/test datasets.

Figure 4.1: The AVE dataset. Some examples in the dataset are shown. The distribution of videos in different categories and the distribution of event lengths are illustrated boundaries. [2]

4.1.2 Audio-Set

AudioSet is a large-scale dataset of audio events, containing more than 2 million 10-second audio clips from YouTube videos. The dataset covers 527 audio event classes organized in a hierarchical ontology. It was created to advance the development of machine learning models for audio event recognition and understanding. AudioSet is often used for tasks such as audio classification, audio event detection, and audio tagging. [20]

4.1.3 LibriSpeech

LibriSpeech [23] is a large-scale corpus of read English speech data created by the Linguistic Data Consortium. The dataset consists of approximately 1,000 hours of speech from public-domain audiobooks (the LibriVox project), read by native English speakers, and is split into several subsets for different purposes, including training, development, and testing. The dataset covers a wide range of topics, speakers, and acoustic conditions, making it suitable for training and evaluating automatic speech recognition (ASR) systems.

The LibriSpeech dataset has become a widely used benchmark for evaluating ASR systems, and has been used to train and evaluate many state-of-the-art ASR models, including the original Wav2vec model [22] (which was described in 3.3.5). The availability of the dataset has enabled the development of more accurate and robust speech recognition systems, with potential applications in a variety of speech-related tasks.

4.1.4 Libri-Light

The LibriLight dataset[55] is a subset of the larger LibriSpeech dataset[23].

Libri-Light is a dataset designed for the development of unsupervised and semi-supervised speech recognition models. It is derived from the LibriSpeech dataset and contains around 60,000 hours of untranscribed audio data.

Libri-Light also includes a smaller, labeled subset that can be used for supervised training and evaluation. The dataset is particularly useful for training automatic speech recognition (ASR) systems and evaluating their performance in low-resource settings.

4.1.5 Imagenet

ImageNet is a large-scale dataset for visual object recognition, consisting of more than 14 million images organized into a hierarchical structure based on the WordNet ontology. The dataset contains images from over 20,000 categories, with each category containing several hundred images. ImageNet has been widely used for tasks such as image classification, object detection, and fine-grained recognition. The annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has spurred the development of many state-of-the-art computer vision models. [56]

4.1.6 MagnaTagATune (MTT) dataset

The MTT dataset considers this 50-tags vocabulary: guitar, classical, slow, techno, strings, drums, electronic, rock, fast, piano, ambient, beat, violin, vocal, synth, female, indian, opera, male, singing, vocals, no vocals, harpsichord, loud, quiet, flute, woman, male vocal, no vocal, pop, soft, sitar, solo, man, classic, choir, voice, new age, dance, male voice, female vocal, beats, harp, cello, no voice, weird, country, metal, female voice, choral.

4.1.7 Million Song Dataset (MSD)

The MSD dataset considers this 50-tags vocabulary: rock, pop, alternative, indie, electronic, female vocalists, dance, 00s, alternative rock, jazz, beautiful, metal, chillout, male vocalists, classic rock, soul, indie rock, Mellow, electronica, 80s, folk, 90s, chill, instrumental, punk, oldies, blues, hard rock, ambient, acoustic, experimental, female vocalist, guitar, Hip-Hop, 70s, party, country, easy listening, sexy, catchy, funk, electro, heavy metal, Progressive rock, 60s, rnb, indie pop, sad, House, happy.

Chapter 5

Implementation

For the implementation, several models were tested. From a simple audio model and a simple visual model to the AV-att and DMRN model, some new models that utilize GRUs were implemented and tested in the AVE dataset.

Various Feature

5.1 Audio-Visual objective

The objective of event localization is to determine the event label for each segment in a video sequence, which encompasses both audio and visual components. The video sequence is divided into T non-overlapping segments, each lasting 1 second, and denoted as V_t, A_t, t=1,..., T where V_t represents the visual content and A_t the corresponding audio. The event label, y_{tk}, for a given segment is represented as $y_{tk}|y_{tk} \in 0, 1, k = 1, ..., C, \sum_{k=1}^{C} y_{tk} =$ 1, where C is the number of AVE events plus one background label.

During training, the event label yt for each visual segment Vt or audio segment At is provided. This task examines whether audio and visual information can improve event localization in audio space, visual space, and the joint audio-visual space.

5.2 Audio-Visual event localization parts

The "Audio-Visual Event Localization Network" is composed of 5 main modules:

(a) feature extraction

(b) audio-guided visual attention

(c) temporal modeling

(d) multimodal fusion

(e) temporal labeling

The network can be seen in Figure 5.1.

For the first module, i.e. the feature extraction module, two branches are used to extract features from the audio and visual modalities of the videos. Several pre-trained models are used to extract the audio and the visual features.

Those features are then used for audio-guided visual attention. The audio features and the output of the audio guided visual attention are then passed to two different recurrent neural networks (RNNs) for temporal modeling. Two types of RNNs where used:

- long short-term memory (LSTM)
- gated recurrent units (GRU)

The LSTMs were used in the paper of [2]

The output of the LSTMs or GRUs is then combined together in the multimodal fusion module.

Two different techniques are used for audiovisual feature fusion. These are:

- a simple concatenation of the audio and visual output
- Dual multimodal residual network (DMRN)

Finally, after the fusion the temporal labeling happens, and the prediction for each video segment is made.

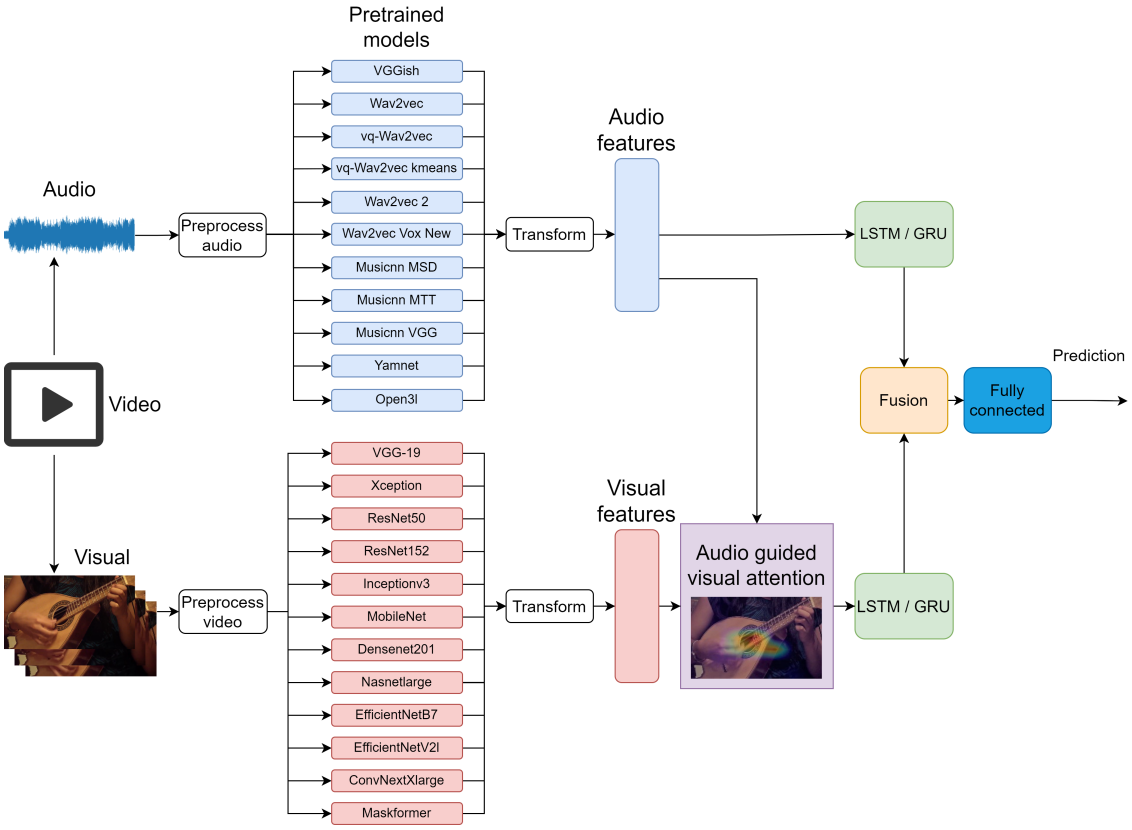


Figure 5.1: Audio-visual event localization network

The network will be divided into two parts.

In the first part, the focus will be on the extraction of features for audio and visual modalities. The process of extracting features involves using a model and taking one of its layers as the output, which can be considered as a representation of the input data.

In the second part, the extracted features will be taken as input. The discussion will encompass the various other modules involved in the audio-visual event localization task, including audio-guided visual attention, temporal modeling, multimodal fusion, temporal labeling, and the neural network models used to implement these. Temporal modeling is an essential aspect of this task, as it helps to capture the temporal dependencies between the audio and visual modalities. The audio-guided visual attention mechanism is used to selectively attend to the most relevant visual information based on the audio input. The multimodal fusion step combines the information from both modalities to produce a final representation. Finally, the temporal labeling step maps the final representation to the desired output, which is the prediction of the event. These parts can be seen in Figure 5.2.

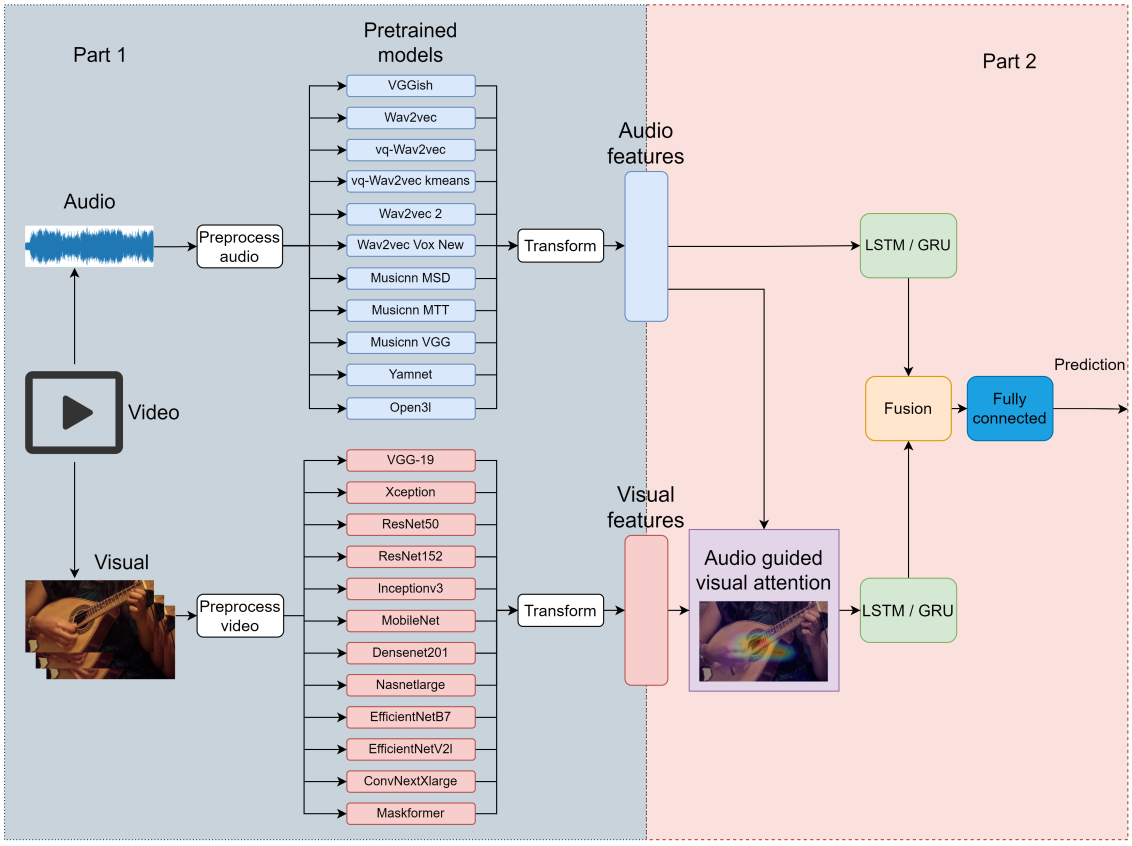


Figure 5.2: Parts for audio-visual event localization network

5.3 Pre-processing, feature extraction and transformation of features

Numerous pre-trained models necessitate that input data, whether audio or visual, adhere to a specific format, a process referred to as preprocessing.

As mentioned earlier, employing a pre-trained model for feature extraction entails utilizing a layer preceding the output layer to obtain representations of the input data.

Subsequently, the data extracted from the layer must be converted into a suitable format for input into classification models, such as AV-att and DMRN.

The conversion involves dimensionality reduction, concatenation, and other methods. The approach varies depending on the pre-trained model and is not restricted to a single method.

5.4 Pretrained models for audio feature extraction

Several pre-trained models were used that transform audio input into a compact, 128-dimensional (or similar dimension) embedding, which can then be utilized as input to a subsequent classification model. These models will be hereby called **feature extraction models**. This dimension is per-second so the output for the 10-second videos will be of dimension 10 x 128 (or in general 10 x d, where d is the dimension).

5.4.1 VGGish

VGGish [57], based on the VGG model [19], is the model that was used in Tian et. al. [2] as the audio feature extraction model.

The length of the audio sequences is set to be 10 seconds and the sample rate to 16kHz.

Then the audio files are loaded in a loop. So, the steps inside the loop are:

- Load an audio file
- Convert audio file into a log mel spectogram.
- The input and output tensors of the VGGish model are then obtained, and the input batch is fed into the model to produce the embedding batch.
- The embedding batch is then stored in the audio features array.

More details on how the VGGish model is trained and used to extract features can be found in Hershey et al. [57] paper and in the repository of the model [58]

5.4.2 Wav2vec, vq-Wav2vec and Wav2vec 2.0

- Wav2vec Large [22] pretrained in the Librispeech dataset [23]
- VQ-Wav2vec [24] pretrained in the Librispeech dataset [23]

- VQ-Wav2vec_kmeans [24] pretrained in the Librispeech dataset [23]
- VQ-Wav2vec Roberta on K-means codes (or beart kmeans)[24] pretrained in the Librispeech dataset [23]
- Wav2Vec 2.0 Base [25] (described in 3.3.7) pretrained in the Librispeech dataset [23] (described in 4.1.3) with no fine-tuning

  The base model pre-trained on 960 hours of Librispeech on 16kHz sampled speech audio
- Wav2vec Vox New [25] (described in 3.3.7.1) pretrained in the Libri-Light dataset (audio data from LibriVox (LV-60k)) (described in 4.1.4)

These pre-trained models were downloaded from Facebook research fairseq repository[59]

The length of the audio sequences is set to be 10 seconds and the sample rate to 16kHz. If the audio is less than 10 seconds then the remaining time is filled with empty value, in order to be exactly 10 seconds.

The wav2vec feature extractor consists of several convolutional layers that transform the input waveform into a higher-level representation. The output shape of z depends on the specific wav2vec model and its architecture, particularly the number and size of the convolutional layers. In general, the output shape of z will be (batch_size, num_features, T), where batch_size is the number of input waveforms (in this case, 1), num_features is the number of feature channels produced by the feature extractor, and T is the number of time steps in the output representation, which is determined by the stride and kernel size of the convolutional layers.

Three approaches were made to extract the features.

1. In the first approach, the audio file was divided into segments of one second each (as an array), and each segment was passed through a pre-trained wav2vec model. The resulting features were then concatenated, and PCA was performed on the entire array of audio features. This method serves as the standard approach.
2. In the second approach, similar to the first, the audio file was divided into segments of one second each (as an array), and each segment was passed through a pre-trained wav2vec model. However, in this case, the dimensions were reduced by performing PCA or calculating the mean for each segment. The reduced feature arrays for each individual second were then concatenated to form the full 10-second audio features. This method is referred to as the ”per second (PS)” approach.
3. In the third approach, the entire audio file (as an array) was passed through a pre-trained wav2vec model. The dimensions were then reduced by calculating the mean. If necessary, zeros were added to the end of the array to create a more rounded shape, making it easier to reduce to the desired format (10 x 128 or 10 x d). This method is referred to as the ”np-mean (np-m)” approach.

From the results of the experiments, (6.1.1 and 6.1.6) it is shown that the standard approach was the best out of the three.

5.4.3 musicnn

Musicnn allows someone to extract features at every layer of the model.

Out of the pretrained extractor, the output of the model is available (the taggram and its associated tags) and all the intermediate representations of it (we refer to those as features). These features are 'timbral', 'temporal', 'cnn1', 'cnn2', 'cnn3', 'mean_pool', 'max_pool', 'penultimate'.

These different key-features correspond to the outputs of the different layers that the musicnn model has. The basic bulding blocks of the model has been shown in Figure 3.5.

The musically motivated CNN front-end is the convolutional layer responsible for processing log-mel spectrogram inputs. A musically motivated CNN is employed, as illustrated in the left figure, comprising a convolutional layer with various filter shapes that have different receptive fields to capture musically relevant contexts, which are depicted in the right figure 5.3:

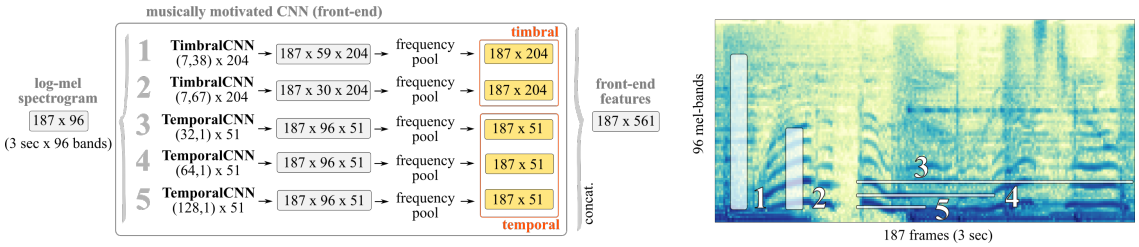


Figure 5.3: Musicnn frontend

TimbralCNNs utilize vertical filters to capture timbral traces. Temporal CNNs use horizontal filters to capture long and short temporal dependencies in spectrograms. Both timbral and temporal features might contain information related to these low-level features.

The dense layers mid-end is tasked with extracting higher-level representations from the low-level features computed by the front-end. It features residual connections that facilitate training and dense connections that allow the back-end to consider information extracted at different hierarchical levels. (figure 5.4)

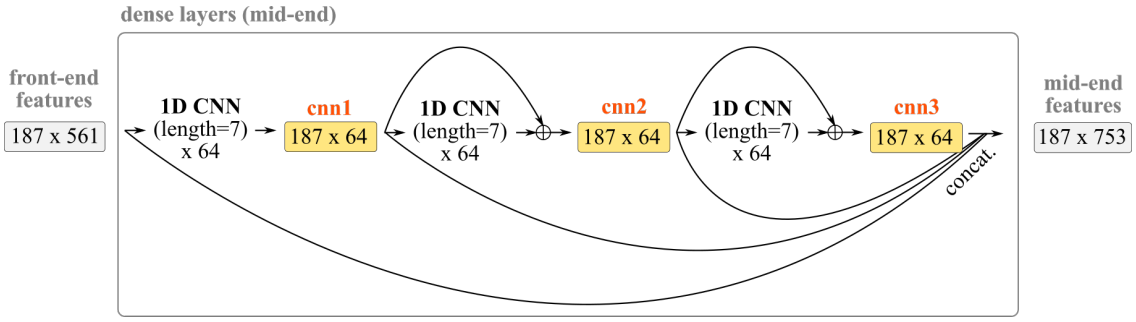


Figure 5.4: Musicnn midend

The dense temporal-pooling back-end is responsible for predicting tags based on the extracted features. Notably, it accommodates variable-length inputs, as the temporal-pooling layers of the back-end can adapt any feature-map length to a fixed feature-map size. (Figure 5.5)

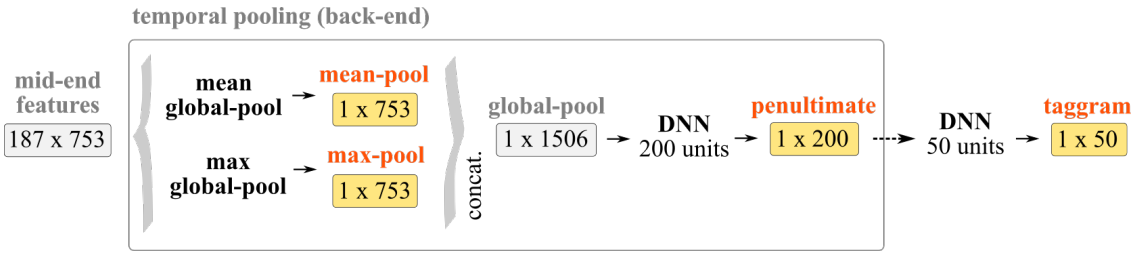


Figure 5.5: Musicnn backend

Features extracted from the musicnn model are then reshaped to a format appropriate for the task. (i.e. 10 x 128 or (10 x d_1))

The full audio of 10 seconds is loaded to the musicnn model.

This happens by calculating the mean and also filling with zeros when appropriate to make the mean to be easier.

Furthermore, features from 'cnn1', 'cnn2', 'cnn3' layers were not only transformed into the appropriate format of 10x64, but also combined together to create a new audio feature table. So, the avg, the sum and the concatenation of them was calculated.

From the results of the experiments, (6.1.1 and 6.1.6) it is shown that the concatenation of these features performed the best.

5.4.3.1 MTT and MSD musicnn

Both MagnaTagATune (MTT) and Million Song Dataset (MSD) musicnn were tried. MTT musicnn seems to perform better.

5.4.3.2 musicnn VGG MTT

Out of the extractor, the output of the model is available(the taggram and its associated tags) and all the intermediate representations of it (referred to as features). The features

are: 'pool1', 'pool2', 'pool3', 'pool4', 'pool5'.

The full audio is loaded to the musicnn VGG model, but it was speedup because the model was trained for 3-second audio. So the audio is speedup a little more than 3x.

These key-features correspond to the outputs of the different layers of the vgg model as shown in figure 3.6.

Features extracted from the musicnn model are then reshaped to a format appropriate for the task. (i.e. 10 x 128 or (10 x d_1))

From the results, (6.1.1 and 6.1.6), pool4 is shown to be the best feature appropriate for the task of this diploma thesis.

5.4.4 Yamnet

YAMNet audio event classifier is used to extract audio embeddings. These audio embeddings then can be used to create the audio features.

The full audio of each song is loaded into the model and the embeddings are exported. Three different approaches were made.

1. In the first approach, dimensionality reduction is made in the embeddings by calculating the mean of some data and reshaping, it multiple times. This results in an array of 10 x 128 for each song.

2. In the second approach, a small dimensionality reduction happens first by calculating the mean. But then the whole audio features array is calculated and PCA is performed in order to get the audio features table, were an audio feature array is of dimension 10x128.

3. In the third approach, a small dimensionality reduction happens first by calculating the mean. Each song retains a high number of features (10 x 1024).

From the results of the experiments, (6.1.1 and 6.1.6) it is shown that PCA is a much better method for retaining information than calculating the mean. Furthermore, it is shown that PCA practically retains high level information about the features, as the accuracy metric of these two features is almost the same.

5.4.5 Openl3

The audio of each song is loaded into the model as clips of 1 sec. The embeddings of every clip are then exported. After this, a dimensionality reduction happens through calculating the mean. Then, all data from the clips are concatenated together and are reduced in the end by a PCA transformation. They are reduced to a dimension of 10x128.

5.5 Pretrained models for visual feature extraction

Several pre-trained models were used that transform visual input into a compact, 7 $\times$ 7 x 512 - D (or similar dimension) feature map embedding, which can then be utilized as

input to a subsequent classification model. This dimension is per-second so the output for the 10-second videos will be of dimension 10 x 7 × 7 x 512 (or in general 10 x d_1 x d_2 x d_3).

In the given process, a loop iterates through each video in the dataset. During each iteration, the video is read and its frames are resized to dimensions of 224 x 224 (or in some cases a different dimension). Sixteen frames are chosen from the video at regular intervals, with one frame per second. A pre-trained model is employed to extract features from each of these selected frames. The extracted features are then averaged across all 16 frames to obtain a single feature vector for each second of the video. Finally, these feature vectors are saved in an array.

5.5.1 VGG-19

The loaded pretrained VGG-19 model generates the output from the block5_pool layer, which consists of the feature maps obtained from the final pooling layer in the VGG19 network. The output is a 7x7x512-D array per second, so a 10x7x7x512-D array per video.

This is the model of Tian et al. [2] that was used as a baseline visual feature extraction model.

5.5.2 Xception

Two approaches were followed in the xception model.

1. The loaded pretrained Xception model generates the output from the avg_pool layer, which consists of the feature maps obtained from the final pooling layer in the Xception network. The avg pool layer has output shape a 1x2048 shape. This was reshaped to a 7x7x42-D array per second (after adding zeros to make it 1x2058), so a 10x7x7x42-D array per video.

2. The loaded pretrained Xception model generates the output from the block14 sepconv2 act layer, which consists of the feature maps obtained from a previous layer in the Xception network. The block14 sepconv2 act layer has an outputof shape 10x10x2048. This was reduced by calculating the mean to a 10x10x512-D array per second, so a 10x10x10x512-D array per video.

5.5.3 ResNet50 and ResNet152

The loaded pretrained ResNet50 and ResNet152 models generates the output from the conv5 block3 out layer, which consists of the feature maps obtained from the last activation layer in the ResNet50 and ResNet152 network. The conv5 block3 out layer has an outputof shape 7x7x2048. This was reduced by calculating the mean to a 7x7x512-D array per second, so a 10x7x7x512-D array per video.

5.5.4 Inceptionv3

The loaded pretrained Inceptionv3 model generates the output from the mixed10 layer, which consists of the feature maps obtained from a previous layer in the Inceptionv3 network. The mixed10 layer has an outputof shape 8x8x2048. This was reduced by calculating the mean to a 8x8x512-D array per second, so a 10x8x8x512-D array per video.

5.5.5 MobileNet

The loaded pretrained MobileNet model generates the output from the conv pw 13 relu layer, which consists of the feature maps obtained from a previous layer in the MobileNet network. The conv pw 13 relu layer has an outputof shape 7x7x1024. This was reduced by calculating the mean to a 7x7x512-D array per second, so a 10x7x7x512-D array per video.

5.5.6 Densenet201

The loaded pretrained Densenet model generates the output from the relu layer, which consists of the feature maps obtained from a previous layer in the Densenet network. The relu layer has an outputof shape 7x7x1920. This was reduced by calculating the mean to a 7x7x512-D array per second, so a 10x7x7x512-D array per video.

5.5.7 Nasnetlarge

In this pretrained model each video is read and its frames are resized to dimensions of 331x331.

The loaded pretrained Nasnetlarge model generates the output from the activation_259 layer, which consists of the feature maps obtained from a previous layer in the Nasnetlarge network. The activation_259 layer has an outputof shape 11x11x4032. This was reduced by calculating the mean to a 11x11x504-D array per second, so a 10x11x11x504-D array per video.

5.5.8 EfficientNetB7

In this pretrained model each video is read and its frames are resized to dimensions of 600x600

The loaded pretrained EfficientNetB7 model generates the output from the top_activation layer, which consists of the feature maps obtained from a previous layer in the EfficientNetB7 network. The top_activation layer has an outputof shape 19x19x2560. Three approaches then were followed.

1. The dimensions were reduced by calculating the mean to a 19x19x256-D so a 10x19x19x256-D array per video. Through a PCA this was transformed into a 10x7x7x256-D

2. The dimensions were reduced by calculating the mean to a and a 19x19x128-D array per second, and 10x19x19x128-D array per video accordingly.

3. The dimensions were reduced by performing PCA when extractinng features, specifically after averaging features for 16 frames in each second, perform a pca from a 10x19x19x2560 to a 10x19x19x128 array.

5.5.9 EfficientNetV2l

In this pretrained model each video is read and its frames are resized to dimensions of 480x480.

The loaded pretrained EfficientNetV2l model generates the output from the top activation layer, which consists of the feature maps obtained from a previous layer in the EfficientNetV2l network. The top activation layer has an output of shape 15x15x1280. This was reduced by performing PCA to each feature to a 15x15x128-D array per second, so a 10x15x15x128-D array per video.

5.5.10 ConvNextXlarge

The loaded pretrained ConvNextXlarge model generates the output from the ”convnext xlarge stage 3 block 2 identity” layer, which consists of the feature maps obtained from a previous layer in the ConvNextXlarge network. The ”convnext xlarge stage 3 block 2 identity” layer has an output of shape 7x7x2048. This was reduced by performing PCA to each feature to a 7x7x204-D array per second, so a 10x7x7x204-D array per video. A second approach was also made, and it was reduced by performing PCA to each feature to a 7x7x490-D array per second, so a 10x7x7x2490-D array per video.

5.5.11 Maskformer

The MaskFormer model, trained on ADE20k semantic segmentation with a large-sized version and a Swin backbone, was initially presented in the paper titled ”Per-Pixel Classification is Not All You Need for Semantic Segmentation” [54].

5.5.11.1 Class query logits and mask query logits as visual features

Class query logits and mask query logits are intermediate representations within a segmentation model, and while they are primarily used for segmentation tasks, they can potentially be used as visual features.

However, there are some caveats to consider:

Representational power: Class query logits and mask query logits may not capture as much information as the embeddings from earlier layers in the model, such as feature maps produced by convolutional layers or transformer blocks. These earlier layers often contain richer and more diverse features, making them more suitable as visual features.

Task specificity: The class query logits and mask query logits are designed specifically for segmentation tasks, meaning that their usefulness as visual features for other tasks (e.g., classification, detection, or retrieval) might be limited. It would be more appropriate to use features extracted from earlier layers in the model or from a model specifically designed for the task of interest.

Dimensionality: Class query logits and mask query logits can be high-dimensional, depending on the number of classes and the size of the masks. This could pose challenges when using them as features, as high-dimensional data often requires more computational resources and can suffer from the curse of dimensionality. In such cases, dimensionality reduction techniques, such as PCA or t-SNE, may be necessary to reduce the dimensionality of the extracted features.

5.5.11.2 Implementation

Class query logits and mask query logits are selected as visual features. Through calculating the mean, and reshaping they are reduced in size.

Class queries logits are reduced to 10x10x151 dimension (per second) and mask queries logits are reduced to 10x16x160(per second).

From the results of the experiments, (6.1.1 and 6.1.7) it is shown that maskformer class logits hold some valuable information about the task of this diploma thesis.

5.6 AV-att model

The authors of [2] propose two deep neural network models for audio-visual event localization: the AV-ATT model and the DMRN model.

The AV-ATT model is an attention-based audio-visual fusion model that leverages audio and visual information to localize audio-visual events. The model consists of two branches: an audio branch that processes the audio features and a visual branch that processes the visual features. The outputs of the two branches are then combined using an attention mechanism that takes into account the correlations between the audio and visual information.

The DMRN model is a dual-modality recurrent network that combines audio and visual information in a recurrent manner to localize audio-visual events. The model consists of two parallel recurrent networks: an audio recurrent network and a visual recurrent network. The two networks process the audio and visual features respectively, and the outputs are combined to generate the final prediction.

Both the AV-ATT and DMRN models are trained end-to-end using a binary cross-entropy loss function that minimizes the difference between the predicted and ground-truth event boundaries. The authors evaluate the two models on a large-scale audio-visual event localization dataset and show that they outperform several state-of-the-art methods for audio-visual event localization.

The audio branch applies a linear transformation to the audio features to project them into a lower-dimensional space, while the visual branch uses a linear transformation to project the visual features into a lower-dimensional space.

The audio-guided visual attention module uses the audio features to guide the attention mechanism to attend to the most relevant parts of the visual features. This is done by computing a set of attention scores, which are used to weight the visual features and generate an attended representation of the visual features. This can be seen in 5.7

For the temporal modeling module, two Bidirectional LSTMs (Bi-LSTMs) are used to capture the temporal dependencies in both the audio and visual features. The audio and visual features are fed into separate Bi-LSTMs, and the outputs from the two Bi-LSTMs are concatenated and passed through two fully connected layers for prediction.

The multimodal fusion module concatenates the outputs from the two Bi-LSTMs, and the resulting representation is passed through two fully connected layers to produce the final prediction.

The final module, temporal labeling, produces a prediction for the audio-visual event category for each time step in the input video. The prediction is made using a softmax activation function.

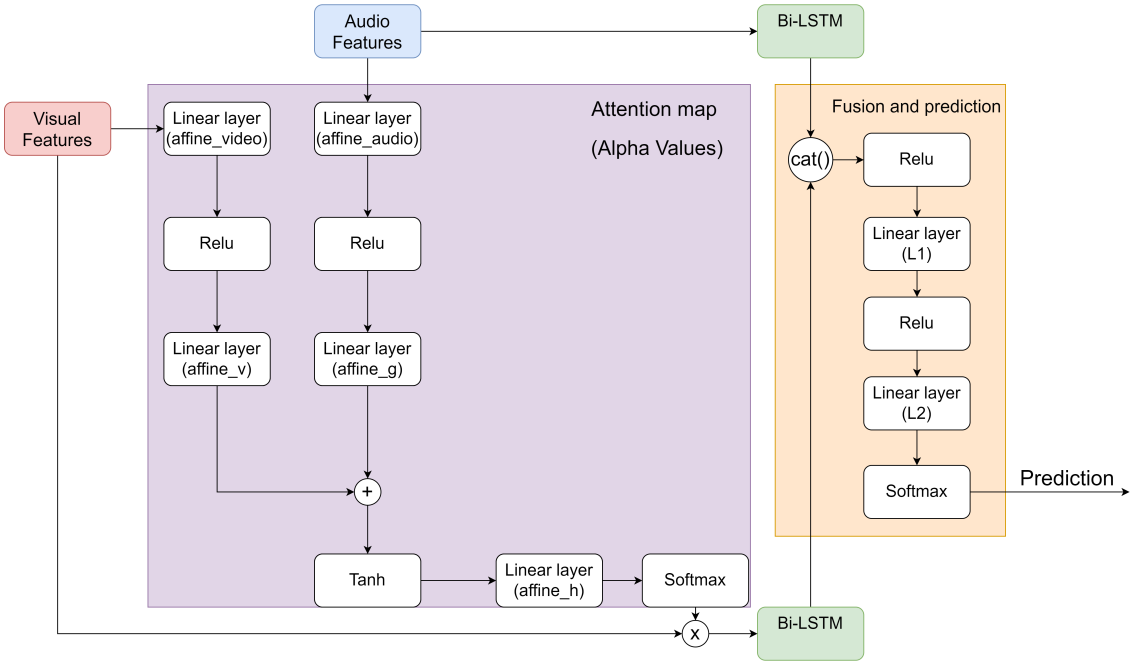


Figure 5.6: AV-att model architecture

5.7 DMRN model

The DMRN model is a deep learning model for audio-visual multi-modal fusion and classification. The model is called TBMRF_Net, which stands for Two-Branch/Dual Multi-Modal Residual Fusion Network.

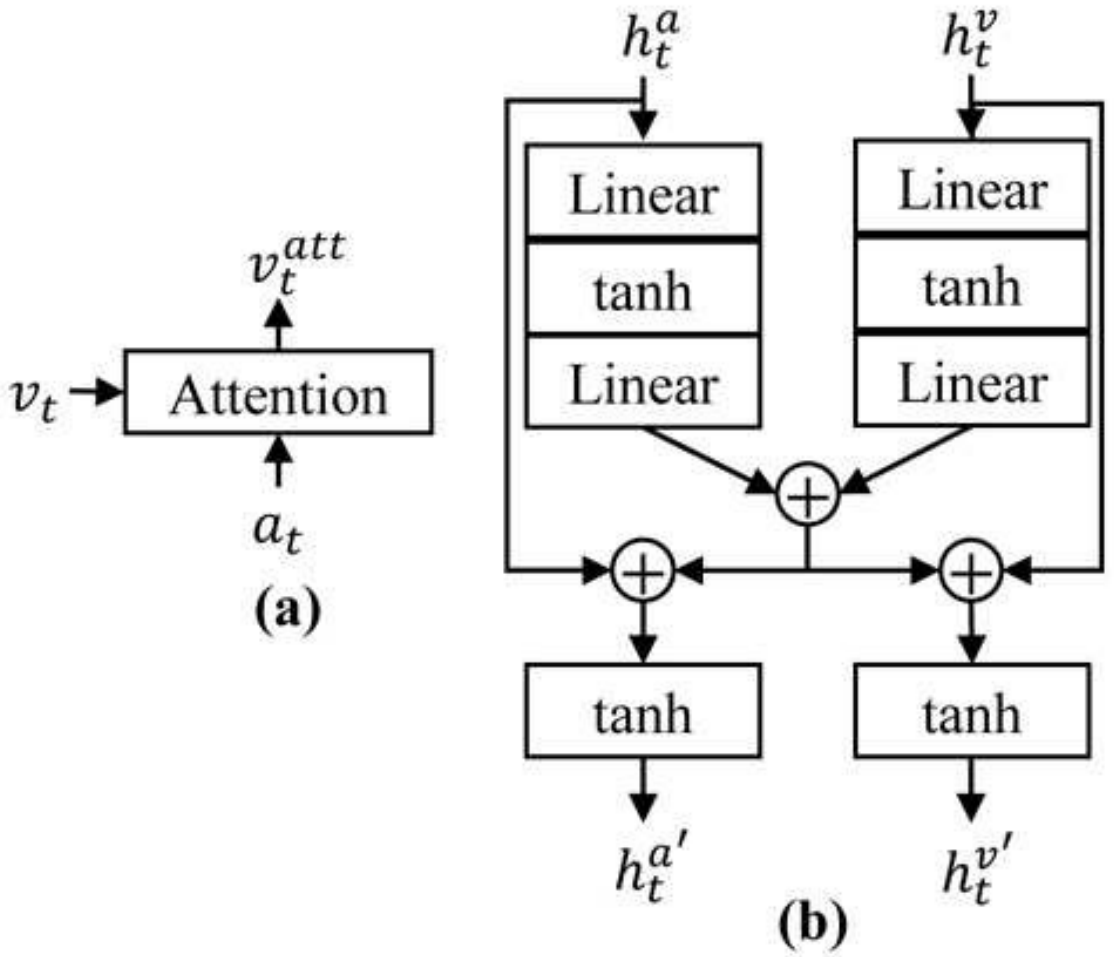


Figure 5.7: (a) Audio-guided visual attention mechanism. (b) Dual multimodal residual network for audiovisual feature fusion [2]

The model takes two inputs, audio and video, and performs multiple operations to generate a prediction. The operations are divided into several blocks, including audio-visual attention, temporal modeling, and feature fusion.

- Audio-Visual Attention The visual input is first passed through a linear layer to produce a dense representation, then the audio input is also passed through a linear layer to produce another dense representation. These two representations are then combined to produce an attention map, which is used to weight the visual input and produce an attended visual representation.
- Temporal Modeling The attended visual representation and audio input are then passed through a bidirectional LSTM layer to capture their temporal information.
- Feature Fusion and prediction The outputs from the bidirectional LSTMs are then passed through multiple TBMRF blocks, which perform multi-modal fusion. This can be seen in Figure 5.7 The final output from the TBMRF blocks is passed through

a linear layer to produce the final prediction.

The model is trained using a softmax cross-entropy loss to classify the input audio-visual data into one of several predefined classes.

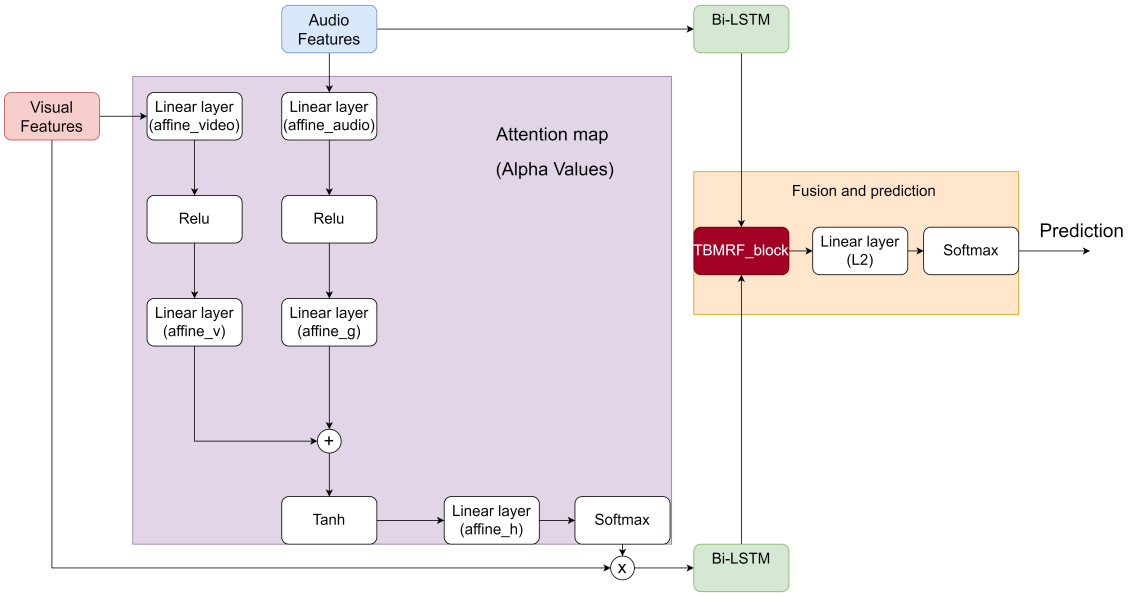


Figure 5.8: DMRN model architecture

5.8 Attention map

The attention network adaptively learns which visual regions in each segment of a video to look for the corresponding sounding object or activity.

5.8.1 Attention map visualization

Visualization takes the affine_h layer as input to visualize the data.

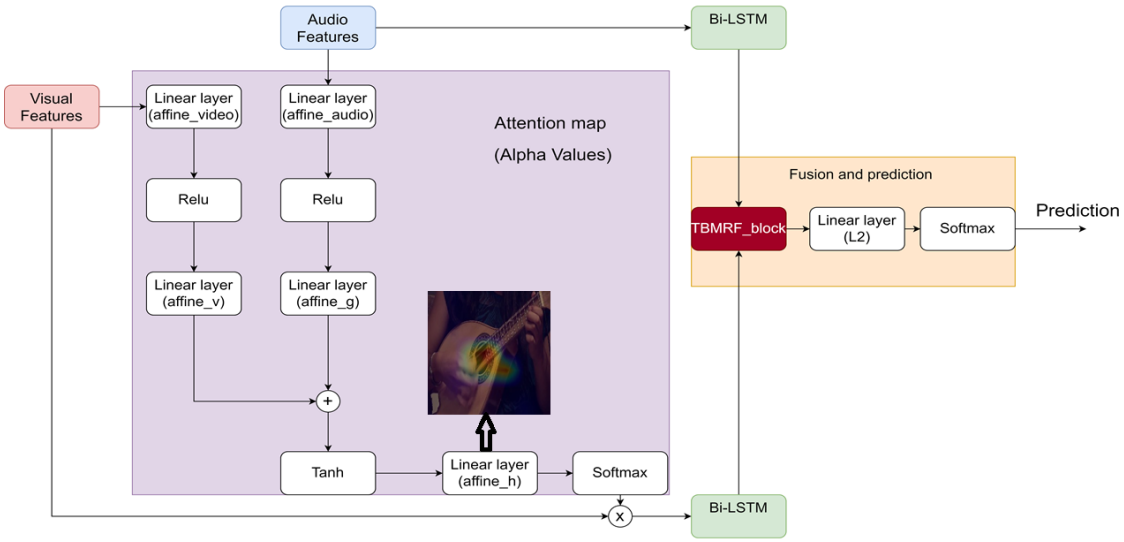


Figure 5.9: DMRN model architecture with attention

5.9 Audio only model

Inspired by the AV_att model 5.6 a simple model for only audio input is computed. This model can predict the labels that we want for a video, by only using the audio input of it.

5.10 Visual only model

Inspired by the AV_att model 5.6 a simple model for only video input is computed. This model can predict the labels that we want for a video, by only using the visual input of it.

5.11 AV att model with GRU

A novel model is proposed that replaces the LSTMs that are used for temporal modeling with GRUs.

5.12 DMRN model with GRU

A novel model is proposed that replaces the LSTMs that are used for temporal modeling with GRUs.

Chapter 6

Experimental results

6.1 Results and comparison between different models

In tables 6.3 and 6.4 there are different results for the various audio and video models used to extract the audio and visual features accordingly. The extracted features were tested with the A+Vatt model and the DMRN model of [2]. The audio features from the paper were extracted using vggish and the visual features from the paper were extracted using VGG-19

Several combinations from audio and visual features have a better accuracy than the inspired paper [2] with the percentage of accuracy at 81%.

Furthermore, the best-performing audio model (with the default visual model VGG-19) is the Openl3 with an accuracy of 0.74, and the best-performing visual model (with the default of VGG-like) is the Convnextxlarge model with an accuracy of 0.8.

6.1.1 Accuracy for best models

Table 6.1: Best models accuracy per class

Models	bell	man	dog	plane	car	woman	copt.	violin	flute	ukul.	frying	truck	shofar	moto.	guitar
Original	0.9	0.72	0.62	0.8	0.72	0.77	0.58	0.71	0.92	0.75	0.87	0.66	0.73	0.8	0.78
Mtt mus - Effb7	0.87	0.78	0.7	0.79	0.83	0.83	0.67	0.91	0.97	0.97	0.86	0.91	0.86	0.9	0.96
openl3-convnxtx	0.89	0.81	0.67	0.89	0.69	0.86	0.79	0.89	0.94	0.93	0.84	0.89	0.92	0.91	0.9
yamnet-effv2l	0.88	0.63	0.61	0.8	0.83	0.65	0.68	0.83	0.94	0.97	0.88	0.94	0.91	0.89	0.94

Models	train	clock	banjo	goat	baby	bus	chain.	cat	horse	toilet	rodent	acco.	mand.	backgr.
Original	0.94	0.9	0.78	0.84	0.73	0.56	0.84	0.3	0.42	0.87	0.6	0.84	0.66	0.53
Mtt mus - Effb7	0.96	0.98	0.89	0.81	0.75	0.8	0.86	0.64	0.78	0.89	0.9	0.85	0.89	0.46
openl3-convnxtx	0.96	0.94	0.9	0.89	0.79	0.77	0.82	0.7	0.52	0.85	0.88	0.82	0.89	0.5
yamnet-effv2l	0.94	1	0.87	0.84	0.53	0.67	0.83	0.49	0.49	0.83	0.88	0.85	0.89	0.43

Table 6.2: Best models accuracy

Models	**accuracy**	**macro avg**	**weighted avg**
Original	**0.73**	0.73	0.73
Mtt mus - Effb7	**0.81**	0.84	0.8
openl3-convnxtx	**0.8**	0.83	0.8
yamnet-effv2l	**0.78**	0.79	0.76

6.1.2 Accuracy for audio models with video vgg-19

Table 6.3: Accuracy of models using audio features extracted by different models and visual features from VGG-19

Model Name	**AV_att**	**DMRN**
Original(vggish)	72.84%	73.08%
Empty Audio Input	53%	57%
wav2vec	63%	59%
wav2vec per second	53%	58%
wav2vec 2 np mean	58%	56%
vq-wav2vec	65%	65%
vq-wav2vec per second	60%	58%
vq-wav2vec-kmeans	63%	63%
vq-wav2vec-kmeans per second	62%	61%
vq-wav2vec-beart-kmeans	63%	63%
wav2vec small	66%	65%
wav2vec vox new	63%	60%
Musicnn mtt timbral 408	57.89%	53.4%
Musicnn mtt temporal 153	55.77%	58.8%
Musicnn mtt cnn1 64	66.37%	64.9%
Musicnn mtt cnn2 64	67.56%	68.08%
Musicnn mtt cnn3 64	66.09%	66%
Musicnn mtt cnn sum 64	68.28%	66.04%
Musicnn mtt cnn avg 64	64.82%	67.06%
Musicnn mtt cnn combined 192	68.43%	63.4%
Musicnn mtt mean_pool 75	59.8%	64.05%
Musicnn mtt max_pool 75	59.65%	60.05%
Musicnn mtt penultimate 20	58.89%	55.79%
Musicnn msd cnn1 64	59.32%	63.13%
Musicnn msd cnn2 64	63.26%	63.63%
Musicnn msd cnn3 64	64.30%	62.76%
Musicnn msd cnn sum 64	64.23%	62.14%
Musicnn msd cnn avg 64	64.03%	62.24%
Musicnn msd cnn combined 192	66.19%	19.5%
Musicnn msd mean_pool 75	59.84%	59.95%
Musicnn msd max_pool 75	59.05%	62.72%
Musicnn msd penultimate 20	56.32%	57.21%
Musicnn vgg pool1 128	56.74%	55.65%
Musicnn vgg pool2 128	62.96%	57.67%
Musicnn vgg pool3 128	65.17%	63.53%
Musicnn vgg pool5 16	54.4%	55.6%
Yamnet	66.27%	67.79%
Yamnet PCA	71.39%	69.4%
Open3l	73.83%	70.85%

6.1.3 Accuracy for video models with audio VGG-like

Table 6.4: Accuracy of models using video features extracted by different models and audio features from VGG-like

Model Name	AV_att	DMRN
Original(VGG-19)	72.84%	73.08%
Empty Visual Input	59.58%	58.41%
Xception 10-10-512	66.12%	67.46%
Xception 7-7-42	66.37%	66.44%
ResNet50	70%	64.55%
ResNet152	70.87%	65.17%
InceptionV3	59.88%	53.78%
MobileNet	68.06%	60.75%
DenseNet201	61.74%	62.43%
NASNetLarge	73.41%	71.52%
EfficientNetB7 7-7-256	63.78%	60.35%
EfficientNetB7 19-19-128	67.56%	59.36%
EfficientNetV2l	77.2%	77.4%
Convnextxlarge	77.9%	78.31%
Convnextxlarge 7-7-490	78.14%	78.22%
MaskFormer masks	60.94%	61.02%
MaskFormer class	62.44%	60.17%

6.1.4 Accuracy for various models

Table 6.5: Accuracy for different models for various pretrained features used (Audio and Visual)

audio / visual	Model	vgg-19	resnet50	resnet152	mobilenet	efficientnetb7	xception	NasNetLarge	convnextxl	convnextxl 490	maskf. mask	maskf. class	efficientnetv2l
VGG-like	AV-att	0.73	0.71	0.7	0.67	0.79	0.64	0.73	0.78	0.79	0.6	0.7	0.77
	AV-att-GRU	0.71	0.68	0.69	0.66	0.79	0.65	0.66	0.78	0.79	0.59	0.68	0.79
	DMRN	0.73	0.68	0.66	0.63	0.78	0.55	0.57	0.77	0.78	0.54	0.67	0.76
	DMRN-GRU	0.69	0.61	0.53	0.58	0.78	0.65	0.69	0.77	0.76	0.56	0.57	0.79
vq-wav2vec	AV-att	0.66	0.58	0.62	0.53	0.7	0.64	0.63	0.75	0.77	0.37	0.59	0.75
	AV-att-GRU	0.64	0.52	0.6	0.55	0.74	0.52	0.61	0.75	0.76	0.32	0.59	0.71
	DMRN	0.62	0.54	0.57	0.5	0.74	0.62	0.6	0.74	0.77	0.38	0.53	0.72
	DMRN-GRU	0.64	0.44	0.46	0.46	0.71	0.52	0.65	0.77	0.76	0.35	0.56	0.74
wav2vec small	AV-att	0.66	0.61	0.64	0.56	0.77	0.59	0.6	0.76	0.78	0.4	0.62	0.7
	AV-att-GRU	0.65	0.6	0.61	0.59	0.75	0.66	0.66	0.75	0.78	0.39	0.62	0.74
	DMRN	0.68	0.57	0.63	0.56	0.74	0.18	0.64	0.75	0.76	0.43	0.59	0.71
	DMRN-GRU	0.67	0.18	0.2	0.2	0.76	0.6	0.62	0.75	0.77	0.43	0.18	0.75
mtt musicnn cnn	AV-att	0.66	0.61	0.62	0.57	0.77	0.64	0.64	0.77	0.78	0.48	0.64	0.77
	AV-att-GRU	0.64	0.58	0.59	0.6	0.81	0.68	0.67	0.78	0.8	0.48	0.63	0.78
	DMRN	0.58	0.04	0.18	0.18	0.67	0.61	0.18	0.71	0.73	0.18	0.21	0.72
	DMRN-GRU	0.55	0.03	0.03	0.03	0.72	0.18	0.57	0.63	0.68	0.34	0.2	0.63
msd musicnn cnn	AV-att	0.63	0.49	0.56	0.58	0.77	0.68	0.68	0.76	0.78	0.46	0.61	0.76
	AV-att-GRU	0.63	0.59	0.54	0.55	0.69	0.69	0.69	0.77	0.77	0.44	0.58	0.77
	DMRN	0.54	0.03	0.03	0.18	0.6	0.18	0.03	0.74	0.72	0.18	0.03	0.36
	DMRN-GRU	0.18	0.03	0.03	0.03	0.6	0.03	0.18	0.18	0.73	0.04	0.03	0.65
musicnn MTT vgg	AV-att	0.65	0.6	0.59	0.58	0.76	0.62	0.65	0.72	0.75	0.42	0.59	0.75
	AV-att-GRU	0.64	0.64	0.6	0.55	0.73	0.66	0.68	0.76	0.76	0.42	0.6	0.75
	DMRN	0.64	0.48	0.57	0.5	0.72	0.62	0.63	0.72	0.74	0.42	0.47	0.7
	DMRN-GRU	0.59	0.18	0.42	0.18	0.69	0.61	0.6	0.73	0.75	0.18	0.18	0.72
yamnet pca	AV-att	0.71	0.65	0.68	0.65	0.74	0.68	0.66	0.77	0.77	0.64	0.68	0.73
	AV-att-GRU	0.7	0.69	0.68	0.67	0.78	0.69	0.74	0.76	0.79	0.63	0.69	0.75
	DMRN	0.69	0.65	0.63	0.66	0.76	0.65	0.67	0.77	0.77	0.65	0.66	0.78
	DMRN-GRU	0.72	0.65	0.65	0.65	0.76	0.65	0.67	0.77	0.77	0.64	0.67	0.76
openl3	AV-att	0.74	0.66	0.67	0.65	0.73	0.65	0.65	0.78	0.79	0.6	0.67	0.77
	AV-att-GRU	0.74	0.68	0.66	0.66	0.79	0.67	0.71	0.8	0.77	0.6	0.72	0.76
	DMRN	0.71	0.63	0.61	0.63	0.74	0.63	0.18	0.78	0.77	0.6	0.63	0.75
	DMRN-GRU	0.72	0.64	0.64	0.61	0.76	0.64	0.63	0.78	0.77	0.6	0.6	0.76

Table 6.6: Training time for different models for various pretrained features used (Audio and Visual)

audio / visual	Model	vgg-19	resnet50	resnet152	mobilenet	efficientnetb7	xception	NasNetLarge	convnextxl	convnextxl 490	maskf. mask	maskf. class	efficientnetv2l
VGG-like	AV-att	1146	1211	1125	1179	8899	2927	4630	774	1429	1073	778	1559
	AV-att-GRU	1138	1182	1115	1072	8802	2353	3167	592	1020	1105	731	1575
	DMRN	1152	1243	1126	1141	8977	2465	4989	906	1151	1174	826	1474
	DMRN-GRU	1321	1119	1151	1167	8907	2589	3225	675	1258	1154	800	1623
vq-wav2vec	AV-att	1065	1171	1270	1261	9022	2553	4689	716	1505	1146	780	1327
	AV-att-GRU	1090	1331	1278	1037	8934	2773	3080	599	1334	1123	775	1192
	DMRN	1178	1133	1239	1229	9015	4117	4412	764	1465	1192	833	1463
	DMRN-GRU	1218	1383	1437	1155	9456	2761	3022	647	1494	1133	792	1482
wav2vec small	AV-att	1147	1146	1778	1879	9600	2595	3066	696	1531	1148	800	1314
	AV-att-GRU	1102	1107	1756	1833	9590	2555	2830	586	1294	1152	778	1204
	DMRN	1181	1314	1851	1983	9618	2582	3111	744	1477	1196	827	1367
	DMRN-GRU	1169	1800	1945	1949	9400	2502	2945	673	1264	1186	838	1340
mtt musicnn cnn	AV-att	1938	1951	1928	1998	8979	2635	3168	718	1558	1116	789	1391
	AV-att-GRU	1946	1963	1929	1951	8901	2604	3224	603	1339	1095	790	1435
	DMRN	1977	2062	2039	1537	2131	2531	3082	782	1512	1161	858	1596
	DMRN-GRU	2010	1979	2045	1310	2111	2413	3096	661	1243	1195	877	1414
msd musicnn cnn	AV-att	1425	1135	1251	1232	2094	2492	3002	710	1332	1185	980	1554
	AV-att-GRU	1356	1070	1130	1218	2575	2382	3041	613	1189	1442	981	1773
	DMRN	1370	1313	1274	1234	2666	2322	3104	773	1424	1665	1062	1822
	DMRN-GRU	1364	1169	1262	1338	2562	2357	2877	694	1317	1749	963	1755
musicnn MTT vgg	AV-att	1112	1463	1345	1109	2136	2775	3159	640	1270	1447	930	1649
	AV-att-GRU	1229	1411	1288	1132	2085	2761	3224	595	1143	1426	880	1587
	DMRN	1148	1486	1382	1147	2125	2595	3083	751	1356	1559	913	1704
	DMRN-GRU	1337	1511	1425	1155	2232	2556	3093	707	1514	1495	1031	1684
yamnet pca	AV-att	1104	1441	1309	1105	2224	2582	3001	687	1345	1535	834	1640
	AV-att-GRU	1460	1373	1154	1063	2068	2503	3016	582	1336	1376	859	1625
	DMRN	1169	1389	1302	1143	2111	2668	3110	741	1498	1523	851	1506
	DMRN-GRU	1695	1565	1275	1111	2222	2669	2871	719	1320	1432	810	1753
openl3	AV-att	1195	1535	1187	1100	2671	2517	5136	654	1268	1382	807	1682
	AV-att-GRU	1528	1380	1044	1069	2646	2412	2959	607	1339	1331	803	1485
	DMRN	1219	1479	1160	1207	2626	2482	5030	742	1362	1375	826	1659
	DMRN-GRU	1515	1447	1157	1125	2790	2379	3021	665	1307	1302	823	1580

Table 6.7: Testing time for different models for various pretrained features used (Audio and Visual)

audio / visual	Model	vgg-19	resnet50	resnet152	mobilenet	efficientnetb7	xception	NasNetLarge	convnextxl	convnextxl 490	maskf. mask	maskf. class	efficientnetv2l
VGG-like	AV-att	12	10	7	8	122	19	36	10	14	15	6	14
	AV-att-GRU	15	9	7	7	120	17	30	7	14	15	10	64
	DMRN	7	28	7	15	122	19	34	8	10	15	10	15
	DMRN-GRU	16	18	7	7	120	19	23	8	15	15	10	16
vq-wav2vec	AV-att	15	7	9	7	123	25	34	7	11	7	10	17
	AV-att-GRU	7	7	10	15	123	29	25	7	13	7	6	16
	DMRN	15	7	9	15	145	27	30	6	15	7	6	10
	DMRN-GRU	7	9	10	7	130	21	23	9	14	7	10	14
wav2vec small	AV-att	15	8	15	17	133	18	77	7	13	7	6	7
	AV-att-GRU	13	15	15	15	159	25	48	9	60	7	10	8
	DMRN	7	16	16	16	127	22	34	7	13	7	6	8
	DMRN-GRU	16	16	16	15	128	20	23	7	10	15	6	13
mtt musicnn cnn	AV-att	20	22	20	19	121	29	30	21	13	27	7	8
	AV-att-GRU	17	18	17	18	116	30	23	7	12	11	10	15
	DMRN	14	24	18	12	25	56	53	7	32	13	8	17
	DMRN-GRU	17	16	19	10	26	28	34	7	12	29	32	13
msd musicnn cnn	AV-att	14	15	7	11	26	59	53	20	12	11	10	28
	AV-att-GRU	15	7	7	7	27	30	35	7	13	11	8	11
	DMRN	11	9	13	10	26	17	31	21	14	14	9	14
	DMRN-GRU	13	39	15	7	27	17	24	8	13	13	9	14
musicnn MTT vgg	AV-att	7	14	13	7	19	30	30	7	12	17	7	13
	AV-att-GRU	15	13	29	15	26	21	24	7	30	13	41	13
	DMRN	7	14	10	15	25	18	52	30	12	17	9	16
	DMRN-GRU	14	13	12	7	14	25	34	7	11	14	7	18
yamnet pca	AV-att	7	11	10	15	28	23	24	7	12	13	9	12
	AV-att-GRU	12	9	26	15	26	30	36	7	7	12	9	15
	DMRN	7	14	10	8	26	20	34	6	9	12	8	13
	DMRN-GRU	20	15	10	15	19	22	47	6	14	13	29	14
openl3	AV-att	8	13	10	15	27	54	36	20	7	13	10	13
	AV-att-GRU	10	14	15	16	28	42	72	7	12	11	10	17
	DMRN	10	11	7	15	20	21	35	27	9	11	7	14
	DMRN-GRU	16	11	7	15	21	28	79	7	12	11	10	12

6.1.5 Wav2vec audio feature extraction models and different methods

Table 6.8: Accuracy for different wav2vec models

Class	VGG-like	wav2vec	wav2vec PS	wav2vec 2 np-mean 128	wav2vec 2 np-mean 99	vq-wav2vec	vq-wav2vec PS	vq-wav2vec kmeans	beart k-means	wav2vec 2 (base)	wav2vec 2 (vox new)
bell	0.9	0.87	0.84	0.79	0.82	0.86	0.85	0.86	0.81	0.87	0.88
man	0.72	0.55	0.52	0.35	0.27	0.56	0.38	0.43	0.42	0.43	0.61
dog	0.62	0.57	0.49	0.31	0	0.52	0.36	0.62	0.57	0.5	0.48
plane	0.8	0.78	0.75	0.79	0.72	0.75	0.72	0.78	0.79	0.84	0.78
car	0.72	0.72	0.6	0.58	0.55	0.67	0.56	0.71	0.73	0.71	0.64
woman	0.77	0.59	0.37	0.68	0.43	0.63	0.47	0.66	0.63	0.67	0.64
copt.	0.58	0.57	0.44	0.53	0.58	0.55	0.48	0.71	0.63	0.55	0.45
violin	0.71	0.66	0.39	0.61	0.57	0.73	0.63	0.83	0.83	0.78	0.65
flute	0.92	0.82	0.44	0.55	0.49	0.65	0.56	0.65	0.69	0.78	0.77
ukul.	0.75	0.82	0.43	0.55	0.19	0.62	0.62	0.61	0.56	0.69	0.72
frying	0.87	0.82	0.8	0.73	0.77	0.82	0.8	0.81	0.78	0.8	0.81
truck	0.66	0.54	0.65	0.53	0.67	0.52	0.54	0.63	0.69	0.62	0.56
shofar	0.73	0.82	0.41	0.58	0.48	0.78	0.58	0.7	0.62	0.75	0.81
moto.	0.8	0.75	0.67	0.77	0.78	0.65	0.84	0.81	0.79	0.78	0.76
guitar	0.78	0.66	0.53	0.57	0.52	0.65	0.62	0.73	0.62	0.67	0.75
train	0.94	0.83	0.81	0.76	0.79	0.76	0.77	0.86	0.83	0.82	0.78
clock	0.9	0.91	0.86	0.86	0.87	0.84	0.85	0.88	0.85	0.88	0.88
banjo	0.78	0.49	0.49	0.71	0.62	0.49	0.56	0.58	0.58	0.7	0.39
goat	0.84	0.46	0.4	0.54	0.56	0.58	0.46	0.62	0.72	0.6	0.59
baby	0.73	0.53	0.36	0.25	0	0.57	0.48	0.56	0.5	0.65	0.54
bus	0.56	0.56	0.72	0.62	0.63	0.48	0.46	0.48	0.64	0.55	0.36
chain.	0.84	0.8	0.7	0.64	0.55	0.78	0.72	0.76	0.71	0.77	0.74
cat	0.3	0.47	0	0.25	0.45	0.21	0.43	0.3	0.48	0	0.32
horse	0.42	0.55	0.55	0.42	0	0.6	0.5	0.39	0.38	0	0.45
toilet	0.87	0.8	0.74	0.67	0.78	0.77	0.74	0.77	0.77	0.72	0.75
rodent	0.6	0.67	0.53	0.62	0.58	0.64	0.58	0.51	0.65	0.62	0.56
acco.	0.84	0.8	0.82	0.87	0.75	0.74	0.82	0.8	0.84	0.83	0.79
mand.	0.66	0.6	0.52	0.41	0.4	0.57	0.39	0.55	0.48	0.41	0.41
backgr.	0.53	0.43	0.43	0.42	0.42	0.42	0.41	0.4	0.46	0.45	0.46
accuracy	0.73	0.66	0.56	0.58	0.55	0.63	0.58	0.65	0.64	0.66	0.63
macro avg	0.73	0.67	0.56	0.58	0.52	0.64	0.59	0.66	0.66	0.64	0.63
weighted avg	0.73	0.65	0.56	0.58	0.53	0.62	0.58	0.64	0.64	0.64	0.63

6.1.6 Comparison between the audio models by using Audio only network

Table 6.9: Model accuracy for Audio-only network part 1

Model	**accuracy**	Model	accuracy
VGG-like	**0.62**	beart k-means	0.31
wav2vec	0.38	wav2vec 2 (base)	0.4
wav2vec PS	0.31	wav2vec 2 (vox new)	0.34
wav2vec 2 np-mean 128	0.18	empty	0.18
wav2vec 2 np-mean 99	0.18	Openl3	**0.59**
vq-wav2vec	0.41	Yamnet	**0.6**
vq-wav2vec PS	0.28	Yamnet pca 128	**0.65**
vq-wav2vec-kmeans	0.31	Yamnet 1024	**0.66**
vq-wav2vec-kmeans PS	0.26		

Table 6.10: Model accuracy for Audio-only network part 2

MTT Musicnn model	Accuracy	MSD Musicnn model	Accuracy	MTT VGG model	Accuracy
cnn1 64	0.39	cnn1 128	0.2	pool1 128	0.32
cnn2 64	0.42	cnn1 64	0.4	pool2 128	0.41
cnn3 64	0.38	cnn2 64	0.44	pool3 128	0.44
cnn avg 64	0.4	cnn3 64	0.41	pool4 128	**0.47**
cnn concat 192	**0.48**	cnn avg 64	0.43	pool5 16	0.28
cnn sum 64	0.39	cnn concat 192	**0.48**		
max pool 75	0.42	cnn sum 64	0.43		
mean pool 75	0.39	max pool 75	0.4		
penultimate 20	0.37	mean pool 75	0.43		
		penultimate 20	0.36		
		temporal 153	0.3		
		timbral 408	0.37		

6.1.7 Comparison between the video models by using Video-only network

Table 6.11: Accuracy for video models by using Video-only network

Video only model	Accuracy
VGG-19	0.56
empty	0.18
xception 7-7-42	0.32
xception 10-10-512	0.52
resnet50	0.56
resnet152	0.54
inceptionv3 8-8-512	0.18
mobilenet	0.48
densenet201 7-7-640	0.22
nasnetlarge 11-11-504	**0.6**
efficientnetb7 7-7-256	0.48
efficientnetb7 19-19-128	0.55
efficientnetb7 19-19-128 pca	**0.67**
efficientnetv2l 15-15-128	**0.71**
convnextxlarge 7-7-204	**0.74**
convnextxlarge 7-7-490	**0.74**
maskformer mask 10-16-160	0.18
maskformer class 10-10-151	0.46

6.1.8 Video model Efficientnetb7 for different transformations and audio features

Table 6.12: Efficientnetb7 model for different transformations

Audio Feature	VGG-like			mtt musicnn
Class / Visual Feature	efficientnetb7 7-7-256	efficientnetb7 19-19-128	efficientnetb7 19-19-128 pca	efficientnetb7 19-19-128 pca
bell	0.89	0.89	0.88	0.87
man	0.52	0.71	0.7	0.78
dog	0.37	0.57	0.65	0.7
plane	0.76	0.77	0.87	0.79
car	0.67	0.65	0.79	0.83
woman	0.64	0.79	0.75	0.83
copt.	0.54	0.45	0.78	0.67
violin	0.76	0.71	0.88	0.91
flute	0.76	0.83	0.93	0.97
ukul.	0.69	0.64	0.91	0.97
frying	0.81	0.88	0.86	0.86
truck	0.7	0.61	0.9	0.91
shofar	0.64	0.61	0.92	0.86
moto.	0.83	0.64	0.86	0.9
guitar	0.76	0.7	0.91	0.96
train	0.91	0.91	0.92	0.96
clock	0.78	0.89	0.87	0.98
banjo	0.68	0.73	0.81	0.89
goat	0.29	0.72	0.67	0.81
baby	0.57	0.62	0.59	0.75
bus	0.72	0.53	0.86	0.8
chain.	0.81	0.89	0.88	0.86
cat	0.3	0.15	0.32	0.64
horse	0.23	0.25	0.23	0.78
toilet	0.76	0.83	0.83	0.89
rodent	0.63	0.63	0.75	0.9
acco.	0.82	0.85	0.88	0.85
mand.	0.46	0.66	0.77	0.89
backgr.	0.39	0.44	0.52	0.46
accuracy	0.64	0.68	0.77	0.81
macro avg	0.65	0.67	0.78	0.84
weighted avg	0.64	0.67	0.77	0.8

6.2 Comparison of models by time and other metrics

Table 6.13: Metrics for Xception for the AV-att model

Metric \ Model	Xception 7-7-42	Xception 10-10-512
accuracy	0.69	0.7
macro avg	0.68	0.67
weighted avg	0.69	0.69
Training time	322	2445
Testing time	3	30

Table 6.14: Metrics for Convnextxlarge for the AV-att model

Metric \ Model	convnextxlarge 7-7-204	convnextxlarge 7-7-408
accuracy	0.78	0.79
macro avg	0.81	0.82
weighted avg	0.78	0.79
Training time	774	1429
Testing time	10	14

Table 6.15: Metrics for openl3-convnextxlarge

Metric \ Model	AV-att	AV-att-GRU	DMRN	DMRN-GRU
Training time	654	607	742	665
Testing time	20	7	27	7
Accuracy	0.78	0.8	0.78	0.78

Table 6.16: Tensorflow keras models and the best accuracy achieved

Model	Size (MB)	Parameters	Depth	Best Accuracy
Xception	88	22.9M	81	0.69
VGG19	549	143.7M	19	0.73
ResNet50	98	25.6M	107	0.71
ResNet152	232	60.4M	311	0.7
InceptionV3	92	23.9M	189	0.61
MobileNet	16	4.3M	55	0.66
DenseNet201	80	20.2M	402	0.62
NASNetLarge	343	88.9M	533	0.73
EfficientNetB7	256	66.7M	438	0.79
EfficientNetV2L	479	119.0M	-	0.79
ConvNeXtXLarge	1310	350.1M	-	0.79

6.3 Comparison between the audio models (Using video VGG-19)

Table 6.17: Classification report for Vgg-original for AV-att and DMRN models

	AV-att			DMRN		
Class	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
bell	0.83	0.99	0.9	0.83	0.95	0.89
man	0.7	0.75	0.72	0.63	0.7	0.66
dog	0.61	0.63	0.62	0.53	0.74	0.62
plane	0.8	0.81	0.8	0.7	0.81	0.75
car	0.71	0.73	0.72	0.73	0.77	0.75
woman	0.73	0.81	0.77	0.81	0.75	0.78
copt.	0.62	0.54	0.58	0.7	0.53	0.61
violin	0.62	0.83	0.71	0.62	0.89	0.73
flute	0.89	0.96	0.92	0.92	0.94	0.93
ukul.	0.75	0.76	0.75	0.69	0.68	0.69
frying	0.82	0.92	0.87	0.83	0.91	0.87
truck	0.71	0.62	0.66	0.89	0.6	0.71
shofar	0.84	0.65	0.73	0.9	0.68	0.77
moto.	0.79	0.81	0.8	0.67	0.92	0.78
guitar	0.9	0.69	0.78	0.86	0.74	0.79
train	0.95	0.92	0.94	0.9	0.91	0.91
clock	0.96	0.85	0.9	0.98	0.8	0.88
banjo	0.76	0.81	0.78	0.76	0.74	0.75
goat	0.88	0.8	0.84	0.65	0.74	0.69
baby	0.89	0.62	0.73	0.84	0.6	0.7
bus	0.42	0.83	0.56	0.43	0.81	0.56
chain.	0.8	0.88	0.84	0.83	0.91	0.87
cat	0.27	0.33	0.3	0.5	0.52	0.51
horse	0.48	0.37	0.42	0.46	0.37	0.41
toilet	0.86	0.87	0.87	0.86	0.96	0.91
rodent	0.67	0.55	0.6	0.84	0.86	0.85
acco.	0.76	0.94	0.84	0.81	0.97	0.88
mand.	0.67	0.65	0.66	0.66	0.53	0.59
backgr.	0.55	0.5	0.53	0.58	0.51	0.54
accuracy			0.73			0.73
macro avg	0.73	0.74	0.73	0.74	0.75	0.74
weighted avg	0.73	0.73	0.73	0.74	0.73	0.73

Table 6.18: Classification report for Wav2vec-small for AV-att and DMRN

	AV-att			**DMRN**		
Class	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
bell	0.78	0.97	0.87	0.8	0.98	0.88
man	0.43	0.43	0.43	0.56	0.46	0.5
dog	0.42	0.6	0.5	0.54	0.51	0.52
plane	0.83	0.86	0.84	0.74	0.85	0.79
car	0.63	0.82	0.71	0.61	0.67	0.64
woman	0.68	0.66	0.67	0.68	0.69	0.69
copt.	0.7	0.46	0.55	0.7	0.47	0.56
violin	0.72	0.87	0.78	0.79	0.87	0.83
flute	0.69	0.9	0.78	0.7	0.86	0.77
ukul.	0.78	0.62	0.69	0.71	0.69	0.7
frying	0.74	0.88	0.8	0.73	0.84	0.78
truck	0.72	0.54	0.62	0.78	0.53	0.63
shofar	0.78	0.73	0.75	0.85	0.73	0.79
moto.	0.7	0.88	0.78	0.77	0.83	0.79
guitar	0.72	0.62	0.67	0.75	0.62	0.68
train	0.85	0.78	0.82	0.79	0.77	0.78
clock	0.96	0.81	0.88	0.95	0.86	0.9
banjo	0.71	0.69	0.7	0.69	0.6	0.64
goat	0.54	0.66	0.6	0.57	0.79	0.66
baby	0.64	0.65	0.65	0.79	0.52	0.63
bus	0.38	0.97	0.55	0.66	0.86	0.75
chain.	0.83	0.72	0.77	0.7	0.83	0.76
cat	0	0	0	0.51	0.55	0.53
horse	0	0	0	0.44	0.37	0.41
toilet	0.65	0.82	0.72	0.8	0.86	0.83
rodent	0.59	0.66	0.62	0.65	0.55	0.59
acco.	0.73	0.96	0.83	0.68	0.95	0.79
mand.	0.45	0.37	0.41	0.59	0.56	0.57
backgr.	0.48	0.42	0.45	0.52	0.51	0.51
accuracy			0.66			0.68
macro avg	0.62	0.67	0.64	0.69	0.7	0.69
weighted avg	0.65	0.66	0.64	0.68	0.68	0.67

Table 6.19: Classification report for Mtt-musicnn-cnn for AV-att and DMRN models

	AV-att			DMRN		
Class	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
bell	0.75	1	0.86	0.78	0.98	0.87
man	0.63	0.59	0.61	0.78	0.4	0.53
dog	0.5	0.53	0.51	0.42	0.43	0.42
plane	0.83	0.74	0.79	0.75	0.75	0.75
car	0.64	0.7	0.67	0.46	0.65	0.54
woman	0.68	0.67	0.67	0.63	0.72	0.67
copt.	0.73	0.52	0.61	0.48	0.55	0.51
violin	0.59	0.72	0.65	0.63	0.74	0.68
flute	0.83	0.86	0.85	0.85	0.88	0.87
ukul.	0.67	0.64	0.66	0.47	0.6	0.53
frying	0.74	0.77	0.76	0.73	0.83	0.78
truck	0.98	0.5	0.66	0.82	0.5	0.62
shofar	0.78	0.83	0.81	0.7	0.83	0.76
moto.	0.68	0.79	0.73	0.78	0.81	0.8
guitar	0.83	0.78	0.8	0.85	0.66	0.74
train	0.82	0.87	0.84	0.86	0.81	0.84
clock	0.88	0.84	0.86	0.91	0.84	0.87
banjo	0.73	0.67	0.7	0.74	0.58	0.65
goat	0.55	0.65	0.59	0.34	0.57	0.43
baby	0.67	0.68	0.68	0	0	0
bus	0.42	0.78	0.54	0.6	0.86	0.7
chain.	0.68	0.87	0.76	0.73	0.83	0.78
cat	0.36	0.24	0.29	0.2	0.27	0.23
horse	0.47	0.7	0.56	0	0	0
toilet	0.88	0.91	0.9	0.82	0.91	0.86
rodent	0.69	0.66	0.68	0.55	0.42	0.48
acco.	0.71	0.94	0.81	0.68	0.88	0.77
mand.	0.55	0.49	0.52	0.57	0.35	0.43
backgr.	0.55	0.48	0.51	0.47	0.48	0.47
accuracy			0.68			0.63
macro avg	0.68	0.7	0.69	0.61	0.62	0.61
weighted avg	0.69	0.68	0.68	0.63	0.63	0.62

Table 6.20: Classification report for Yamnet-pca for AV-att and DMRN-GRU

	AV-att			**DMRN**		
Class	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
bell	0.82	0.96	0.88	0.81	0.96	0.88
man	0.65	0.56	0.6	0.79	0.7	0.74
dog	0.52	0.73	0.6	0.49	0.83	0.62
plane	0.76	0.82	0.79	0.7	0.85	0.77
car	0.7	0.69	0.69	0.65	0.8	0.72
woman	0.6	0.75	0.67	0.7	0.77	0.73
copt.	0.65	0.61	0.63	0.7	0.62	0.66
violin	0.66	0.79	0.72	0.69	0.82	0.75
flute	0.84	0.94	0.89	0.76	0.88	0.81
ukul.	0.76	0.93	0.84	0.72	0.88	0.79
frying	0.76	0.93	0.83	0.79	0.9	0.84
truck	0.74	0.58	0.65	0.79	0.79	0.79
shofar	0.9	0.73	0.81	0.67	0.7	0.69
moto.	0.69	0.91	0.78	0.68	0.81	0.74
guitar	0.83	0.8	0.81	0.86	0.78	0.81
train	0.84	0.93	0.88	0.9	0.92	0.91
clock	0.99	0.94	0.96	0.99	0.97	0.98
banjo	0.86	0.87	0.86	0.8	0.85	0.82
goat	0.61	0.67	0.64	0.61	0.76	0.68
baby	0.53	0.65	0.58	0.71	0.75	0.73
bus	0.51	0.83	0.63	0.43	0.81	0.56
chain.	0.81	0.96	0.88	0.74	0.87	0.8
cat	0.36	0.36	0.36	0	0	0
horse	0	0	0	0	0	0
toilet	0.75	0.82	0.78	0.74	0.93	0.82
rodent	0.62	0.69	0.65	0.66	0.6	0.63
acco.	0.72	0.93	0.81	0.69	0.96	0.8
mand.	0.79	0.72	0.75	0.76	0.73	0.75
backgr.	0.53	0.35	0.42	0.53	0.29	0.38
accuracy			0.71			0.72
macro avg	0.68	0.74	0.7	0.67	0.74	0.7
weighted avg	0.7	0.71	0.7	0.69	0.72	0.69

Table 6.21: Classification report for Openl3 for AV-att and DMRN-GRU models

	AV-att			DMRN-GRU		
Class	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
bell	0.8	0.98	0.88	0.82	0.97	0.89
man	0.72	0.75	0.74	0.6	0.72	0.66
dog	0.59	0.72	0.65	0.58	0.76	0.66
plane	0.84	0.81	0.83	0.73	0.82	0.77
car	0.62	0.71	0.66	0.6	0.58	0.59
woman	0.86	0.83	0.85	0.75	0.83	0.79
copt.	0.78	0.71	0.74	0.78	0.49	0.6
violin	0.73	0.78	0.76	0.79	0.79	0.79
flute	0.81	0.96	0.88	0.9	0.89	0.89
ukul.	0.72	0.89	0.79	0.82	0.75	0.78
frying	0.75	0.88	0.81	0.83	0.93	0.87
truck	0.95	0.69	0.8	0.75	0.77	0.76
shofar	0.9	0.74	0.81	0.86	0.84	0.85
moto.	0.81	0.95	0.87	0.8	0.88	0.84
guitar	0.97	0.77	0.86	0.94	0.89	0.91
train	0.85	0.89	0.87	0.82	0.86	0.84
clock	0.99	0.85	0.92	0.95	0.84	0.89
banjo	0.69	0.7	0.7	0.63	0.82	0.71
goat	0.8	0.55	0.65	0.5	0.57	0.54
baby	0.76	0.94	0.84	1	0.6	0.75
bus	0.52	0.92	0.67	0.43	0.64	0.52
chain.	0.79	0.94	0.86	0.72	0.81	0.76
cat	0	0	0	0.33	0.27	0.3
horse	0.39	0.6	0.48	0	0	0
toilet	0.78	0.85	0.81	0.83	0.78	0.8
rodent	0.67	0.6	0.63	0.79	0.65	0.71
acco.	0.73	0.98	0.84	0.83	0.93	0.88
mand.	0.77	0.76	0.76	0.84	0.66	0.74
backgr.	0.55	0.45	0.5	0.51	0.48	0.49
accuracy			0.74			0.72
macro avg	0.73	0.77	0.74	0.72	0.72	0.71
weighted avg	0.74	0.74	0.73	0.72	0.72	0.71

Table 6.22: Classification report for Openl3

Class	**Precision**	**Recall**	**F1Score**	**Support**
Church bell	0.81	0.96	0.88	140
Male speech, man speaking	0.65	0.64	0.65	126
Bark	0.63	0.72	0.67	89
Fixedwing aircraft, airplane	0.87	0.87	0.87	160
Race car, auto racing	0.70	0.82	0.76	128
Female speech, woman speaking	0.88	0.79	0.84	135
Helicopter	0.83	0.65	0.73	133
Violin, fiddle	0.88	0.74	0.80	151
Flute	0.83	0.92	0.87	161
Ukulele	0.94	0.76	0.84	152
Frying (food)	0.78	0.88	0.83	139
Truck	0.86	0.78	0.82	124
Shofar	0.88	0.80	0.83	88
Motorcycle	0.87	0.71	0.78	75
Acoustic guitar	0.86	0.87	0.86	178
Train horn	0.91	0.88	0.89	158
Clock	0.99	0.89	0.94	160
Banjo	0.69	0.77	0.73	175
Goat	0.75	0.65	0.70	89
Baby cry, infant cry	0.68	0.60	0.64	63
Bus	0.50	0.75	0.60	36
Chainsaw	0.70	0.94	0.80	126
Cat	0.46	0.36	0.41	33
Horse	0.38	0.47	0.42	43
Toilet flush	0.80	0.75	0.78	109
Rodents, rats, mice	0.74	0.74	0.74	80
Accordion	0.82	0.94	0.88	107
Mandolin	0.70	0.76	0.73	147
Background	0.50	0.40	0.45	715
Accordion	0.73	0.93	0.81	107
Mandolin	0.76	0.76	0.76	147
Background	0.50	0.47	0.48	715
Accuracy			**0.74**	**4020**
Macro avg	**0.75**	**0.75**	**0.75**	**4020**
Weighted avg	**0.74**	**0.74**	**0.74**	**4020**

Table 6.23: Classification report for NasNetLarge

Class	Precision	Recall	F1Score	Support
Church bell	0.87	0.99	0.93	140
Male speech, man speaking	0.63	0.74	0.68	126
Bark	0.62	0.60	0.61	89
Fixedwing aircraft, airplane	0.75	0.80	0.77	160
Race car, auto racing	0.73	0.74	0.73	128
Female speech, woman speaking	0.72	0.61	0.66	135
Helicopter	0.63	0.56	0.59	133
Violin, fiddle	0.65	0.85	0.73	151
Flute	0.97	0.94	0.96	161
Ukulele	0.80	0.82	0.81	152
Frying (food)	0.82	0.97	0.89	139
Truck	0.88	0.78	0.83	124
Shofar	0.89	0.66	0.76	88
Motorcycle	0.69	0.79	0.73	75
Acoustic guitar	0.84	0.78	0.81	178
Train horn	0.97	0.99	0.98	158
Clock	0.95	0.97	0.96	160
Banjo	0.84	0.81	0.83	175
Goat	0.67	0.72	0.69	89
Baby cry, infant cry	0.91	0.32	0.47	63
Bus	0.65	0.97	0.78	36
Chainsaw	0.83	0.82	0.82	126
Cat	0.59	0.52	0.55	33
Horse	0.39	0.30	0.34	43
Toilet flush	0.81	0.84	0.83	109
Accordion	0.77	0.93	0.84	107
Mandolin	0.69	0.67	0.68	147
Background	0.49	0.47	0.48	715
Accuracy			**0.73**	**4020**
Macro avg	**0.76**	**0.75**	**0.74**	**4020**
Weighted avg	**0.73**	**0.73**	**0.73**	**4020**

6.4 Comparison between the video models (Using audio VGG-like)

Table 6.24: Classification report for Nasnetlarge for AV-att and DMRN-GRU

	AV-att			DMRN		
Class	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
bell	0.87	0.99	0.93	0.72	0.98	0.83
man	0.63	0.74	0.68	0.6	0.56	0.58
dog	0.62	0.6	0.61	0.53	0.58	0.55
plane	0.75	0.8	0.77	0.72	0.76	0.74
car	0.73	0.74	0.73	0.69	0.59	0.63
woman	0.72	0.61	0.66	0.75	0.61	0.67
copt.	0.63	0.56	0.59	0.59	0.61	0.6
violin	0.65	0.85	0.73	0.57	0.74	0.65
flute	0.97	0.94	0.96	0.92	0.96	0.94
ukul.	0.8	0.82	0.81	0.67	0.83	0.74
frying	0.82	0.97	0.89	0.83	0.93	0.87
truck	0.88	0.78	0.83	0.67	0.68	0.67
shofar	0.89	0.66	0.76	0.9	0.65	0.75
moto.	0.69	0.79	0.73	0.65	0.95	0.77
guitar	0.84	0.78	0.81	0.84	0.69	0.76
train	0.97	0.99	0.98	0.83	0.99	0.9
clock	0.95	0.97	0.96	0.73	0.63	0.68
banjo	0.84	0.81	0.83	0.85	0.83	0.84
goat	0.67	0.72	0.69	0.87	0.78	0.82
baby	0.91	0.32	0.47	0.64	0.51	0.57
bus	0.65	0.97	0.78	0.67	0.86	0.76
chain.	0.83	0.82	0.82	0.84	0.89	0.86
cat	0.59	0.52	0.55	0.22	0.45	0.29
horse	0.39	0.3	0.34	0.41	0.44	0.43
toilet	0.81	0.84	0.83	0.79	0.91	0.84
rodent	0.86	0.78	0.82	0.8	0.64	0.71
acco.	0.77	0.93	0.84	0.67	0.75	0.71
mand.	0.69	0.67	0.68	0.64	0.57	0.6
backgr.	0.49	0.47	0.48	0.54	0.43	0.48
accuracy			0.73			0.69
macro avg	0.76	0.75	0.74	0.69	0.72	0.7
weighted avg	0.73	0.73	0.73	0.69	0.69	0.69

Table 6.25: Classification report for EfficientnetB7 for AV-att and DMRN

	AV-att			**DMRN**		
Class	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
bell	0.83	0.96	0.89	0.82	0.99	0.9
man	0.73	0.74	0.74	0.72	0.81	0.76
dog	0.63	0.67	0.65	0.51	0.69	0.59
plane	0.77	0.82	0.8	0.77	0.85	0.81
car	0.75	0.85	0.8	0.77	0.75	0.76
woman	0.83	0.69	0.75	0.92	0.73	0.81
copt.	0.76	0.65	0.7	0.68	0.68	0.68
violin	0.88	0.92	0.9	0.75	0.93	0.83
flute	0.98	0.99	0.98	0.85	0.94	0.89
ukul.	0.96	0.91	0.94	0.99	0.88	0.93
frying	0.82	0.96	0.88	0.77	0.96	0.86
truck	0.91	0.9	0.9	0.86	0.9	0.88
shofar	0.86	0.91	0.88	0.82	0.99	0.9
moto.	0.89	0.95	0.92	0.9	0.85	0.88
guitar	0.93	0.96	0.95	0.98	0.94	0.96
train	0.93	0.97	0.95	0.92	0.98	0.95
clock	0.98	0.92	0.95	0.96	1	0.98
banjo	0.86	0.79	0.82	0.82	0.86	0.84
goat	0.8	0.85	0.83	0.76	0.85	0.8
baby	0.9	0.6	0.72	0.7	0.63	0.67
bus	0.65	0.97	0.78	0.6	1	0.75
chain.	0.86	0.91	0.88	0.84	0.84	0.84
cat	0.47	0.42	0.44	0.72	0.55	0.62
horse	0.31	0.19	0.23	0.21	0.16	0.18
toilet	0.85	0.91	0.88	0.82	0.9	0.86
rodent	0.87	0.76	0.81	0.86	0.78	0.82
acco.	0.78	0.93	0.85	0.82	0.95	0.88
mand.	0.77	0.88	0.82	0.83	0.86	0.84
backgr.	0.55	0.5	0.52	0.57	0.41	0.48
accuracy			0.79			0.78
macro avg	0.8	0.81	0.8	0.78	0.82	0.79
weighted avg	0.79	0.79	0.79	0.77	0.78	0.77

Table 6.26: Classification report for Efficientnetv2l for AV-att and DMRN-GRU

	AV-att			DMRN		
Class	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
bell	0.83	0.98	0.9	0.82	0.96	0.88
man	0.66	0.71	0.68	0.68	0.72	0.7
dog	0.61	0.61	0.61	0.65	0.74	0.69
plane	0.66	0.89	0.76	0.74	0.91	0.82
car	0.83	0.75	0.79	0.78	0.85	0.82
woman	0.86	0.66	0.75	0.89	0.59	0.71
copt.	0.73	0.62	0.67	0.75	0.69	0.72
violin	0.79	0.79	0.79	0.8	0.9	0.85
flute	0.92	0.97	0.94	0.9	0.96	0.93
ukul.	0.93	0.98	0.96	0.81	0.98	0.89
frying	0.9	0.96	0.93	0.91	0.91	0.91
truck	0.96	0.84	0.9	0.93	0.9	0.91
shofar	0.84	0.81	0.82	0.82	0.84	0.83
moto.	0.89	1	0.94	0.82	0.95	0.88
guitar	0.89	0.87	0.88	0.92	0.79	0.85
train	0.89	1	0.94	0.97	0.96	0.96
clock	0.98	0.97	0.97	0.93	0.99	0.96
banjo	0.77	0.75	0.76	0.84	0.81	0.82
goat	0.92	0.98	0.95	0.91	0.98	0.94
baby	0.87	0.41	0.56	0.65	0.76	0.7
bus	0.76	0.89	0.82	0.66	0.97	0.79
chain.	0.91	0.97	0.94	0.82	0.97	0.89
cat	0.53	0.52	0.52	0.64	0.64	0.64
horse	0.62	0.47	0.53	0	0	0
toilet	0.88	0.91	0.89	0.92	0.94	0.93
rodent	0.89	0.9	0.89	0.9	0.81	0.86
acco.	0.83	0.91	0.87	0.84	0.97	0.9
mand.	0.72	0.78	0.75	0.77	0.86	0.82
backgr.	0.57	0.53	0.55	0.58	0.48	0.53
accuracy			0.79			0.79
macro avg	0.81	0.81	0.8	0.78	0.82	0.8
weighted avg	0.79	0.79	0.78	0.78	0.79	0.78

Table 6.27: Classification report for Convnextxlarge for AV-att-GRU and DMRN-GRU

	AV-att			**DMRN**		
Class	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
bell	0.84	0.92	0.88	0.81	0.97	0.88
man	0.66	0.8	0.73	0.59	0.67	0.63
dog	0.54	0.63	0.58	0.52	0.67	0.59
plane	0.93	0.88	0.9	0.85	0.89	0.87
car	0.75	0.63	0.69	0.79	0.83	0.81
woman	0.75	0.67	0.71	0.66	0.65	0.65
copt.	0.81	0.87	0.84	0.77	0.81	0.79
violin	0.84	0.88	0.86	0.81	0.82	0.81
flute	0.85	0.96	0.9	0.81	0.96	0.88
ukul.	0.96	0.86	0.91	0.92	0.88	0.9
frying	0.83	0.81	0.82	0.81	0.84	0.82
truck	0.88	0.91	0.89	0.84	0.84	0.84
shofar	0.96	0.73	0.83	0.84	0.91	0.87
moto.	0.9	0.88	0.89	0.82	0.87	0.84
guitar	0.86	0.85	0.86	0.89	0.84	0.87
train	0.89	0.99	0.93	0.82	0.99	0.9
clock	0.99	0.91	0.95	0.95	0.9	0.92
banjo	0.91	0.86	0.88	0.82	0.87	0.85
goat	0.82	0.88	0.85	0.91	0.91	0.91
baby	0.97	0.62	0.76	0.72	0.57	0.64
bus	0.65	0.92	0.76	0.75	1	0.86
chain.	0.9	0.82	0.86	0.86	0.9	0.88
cat	0.6	0.55	0.57	0.36	0.48	0.42
horse	0.57	0.67	0.62	0.51	0.42	0.46
toilet	0.85	0.91	0.88	0.9	0.87	0.88
rodent	0.82	0.78	0.79	0.85	0.86	0.86
acco.	0.8	0.95	0.87	0.82	0.95	0.88
mand.	0.82	0.93	0.87	0.81	0.8	0.81
backgr.	0.52	0.49	0.5	0.57	0.42	0.48
accuracy			0.78			0.77
macro avg	0.81	0.81	0.81	0.77	0.81	0.79
weighted avg	0.78	0.78	0.78	0.76	0.77	0.76

Table 6.28: Classification report for Convnextxlarge-7-7-490 for AV-att and DMRN

	AV-att			DMRN		
Class	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
bell	0.86	0.99	0.92	0.85	1	0.92
man	0.64	0.73	0.68	0.71	0.7	0.7
dog	0.67	0.52	0.58	0.58	0.73	0.65
plane	0.91	0.81	0.86	0.91	0.91	0.91
car	0.82	0.73	0.77	0.72	0.75	0.73
woman	0.71	0.76	0.74	0.81	0.73	0.77
copt.	0.77	0.86	0.81	0.76	0.84	0.8
violin	0.83	0.93	0.88	0.81	0.95	0.87
flute	0.91	0.9	0.9	0.86	0.88	0.87
ukul.	0.9	0.93	0.91	0.85	0.93	0.89
frying	0.89	0.86	0.87	0.86	0.91	0.89
truck	0.93	0.78	0.85	0.87	0.83	0.85
shofar	1	0.85	0.92	0.88	0.84	0.86
moto.	0.89	0.99	0.94	0.87	0.88	0.87
guitar	0.87	0.84	0.86	0.86	0.8	0.83
train	0.87	0.92	0.9	0.89	0.89	0.89
clock	0.99	0.88	0.93	0.99	0.88	0.93
banjo	0.91	0.87	0.89	0.85	0.93	0.89
goat	0.88	0.99	0.93	0.8	0.99	0.88
baby	0.82	0.57	0.67	0.8	0.57	0.67
bus	0.71	0.94	0.81	0.66	0.81	0.73
chain.	0.88	0.86	0.87	0.89	0.79	0.84
cat	0.66	0.58	0.61	0.63	0.52	0.57
horse	0.57	0.49	0.53	0.45	0.21	0.29
toilet	0.87	0.89	0.88	0.84	0.87	0.86
rodent	0.88	0.88	0.88	0.84	0.8	0.82
acco.	0.81	0.93	0.87	0.82	0.93	0.87
mand.	0.84	0.93	0.88	0.9	0.84	0.87
backgr.	0.52	0.51	0.52	0.5	0.47	0.48
accuracy			0.79			0.78
macro avg	0.82	0.82	0.82	0.8	0.8	0.79
weighted avg	0.79	0.79	0.79	0.77	0.78	0.77

Part III

Conclusion

Chapter 7

Conclusion and future work

7.1 Conclusions

Numerous models have been discovered that surpass previous state-of-the-art performance in localization tasks.

Wav2vec, despite being designed for speech recognition, can provide valuable audio feature information for event localization.

Similarly, MaskFormer classes, though intended for segmentation, offer valuable visual feature information for event localization.

Using GRU reduces time while maintaining similar accuracy levels.

Feature dimensions are important, but when handled properly, they can be reduced with minimal data loss, resulting in faster models. PCA is a preferable approach compared to mean averaging. Moreover, it is more effective to perform PCA for each audio file, rather than after all features have been extracted.

7.2 Future work

As was shown in musicnn mtt and musicnn msd, combination of multiple layers of the same pre-trained model can lead to better performance of the audio features. This should be further explored. These features could be concatenated or otherwise a PCA could be performed to reduce the dimensions.

Furthermore, there is the possibility that a combination of audio or visual features from different pretrained models could extract a better audio/visual feature. These could happen if these models were trained on different datasets.

There are multiple other pretrained models that could be used for this task.

This task should run for other datasets, such as: VGG sound, VGG-SS (VGG-Sound Source), Kinetics, LLP dataset.

Bibliography

[1] Yan-Bo Lin, Yu-Jhe Li, and Yu-Chiang Frank Wang. Dual-modality seq2seq network for audio-visual event localization, 2019.

[2] Yapeng Tian, Jing Shi, Bochen Li, Zhiyao Duan, and Chenliang Xu. Audio-visual event localization in unconstrained videos, 2018.

[3] Jennifer Joana Gago, Valentina Vasco, Bartek Łukawski, Ugo Pattacini, Vadim Tikhanoff, Juan Victores, and Carlos Balaguer. *Sequence-to-Sequence Natural Language to Humanoid Robot Sign Language*. July 2019.

[4] Junhui Zhao, Yiwen Nie, Shanjin Ni, and Xiaoke Sun. Traffic data imputation and prediction: An efficient realization of deep learning. *IEEE Access*, 8:46713–46722, 2020.

[5] Konstantinos Pyrovolakis, Paraskevi Tzouveli, and Giorgos Stamou. Multi-modal song mood detection with deep learning. *Sensors*, 22(3):1065, jan 2022.

[6] jordipons/musicnn. Musicnn. https://github.com/jordipons/musicnn, 2020.

[7] Bhupendra Pratap Singh. *Imaging Applications of Charge Coupled Devices (CCDs) for Cherenkov Telescope*. May 2015.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[9] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, jan 2015.

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2008.

[11] Jerome Friedman. *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Springer-Verlag New York, 2009.

[12] I. T. Jolliffe. *Principal Component Analysis*. Springer New York, May 2006.

[13] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, nov 1997.

[15] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

[16] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[17] Geoffroy Peeters and Gaël Richard. Deep learning for audio and music. In *Multi-faceted Deep Learning*, pages 231–266. Springer International Publishing, feb 2012.

[18] Konstantinos Pyrovolakis, Paraskevi Tzouveli, and George Stamou. Mood detection analyzing lyrics and audio signal based on deep learning architectures. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, jan 2021.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[20] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, mar 2017.

[21] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. CNN architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, mar 2017.

[22] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. In *Interspeech 2019*. ISCA, sep 2019.

[23] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, apr 2015.

[24] Alexei Baevski, Steffen Schneider, and Michael Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. arXiv, 2019.

[25] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *CoRR*, abs/2006.11477, 2020.

[26] Jordi Pons and Xavier Serra. musicnn: Pre-trained convolutional neural networks for music audio tagging, 2019.

[27] Aurora Linh Cramer, Ho-Hsiang Wu, Justin Salamon, and Juan Pablo Bello. Look, listen, and learn more: Design choices for deep audio embeddings. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, may 2019.

[28] Relja Arandjelović and Andrew Zisserman. Look, listen and learn, 2017.

[29] Rafael C. Gonzalez. *Digital image processing*. Prentice Hall, 2002.

[30] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[31] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, apr 2004.

[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[33] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2013.

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

[35] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016.

[36] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2014.

[37] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2018.

[38] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017.

[39] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[40] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[41] Dimitrios Kollias, Anastasios Arsenos, and Stefanos Kollias. A deep neural architecture for harmonizing 3-d input data analysis and decision making in medical imaging. March 2023.

[42] Dimitrios Kollias, Anastasios Arsenos, and Stefanos Kollias. Ai-mia: Covid-19 detection and severity analysis through medical imaging, 2022.

[43] Dimitrios Kollias, Andreas Psaroudakis, Anastasios Arsenos, and Paraskeui Theofilou. Facernet: a facial expression intensity estimation network, 2023.

[44] Dimitrios Kollias and Stefanos Zafeiriou. Affect analysis in-the-wild: Valence-arousal, expressions, action units and a unified framework, 2021.

[45] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2016.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[47] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[48] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[49] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016.

[50] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2017.

[51] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2019.

[52] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. 2021.

[53] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s, 2022.

[54] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation, 2021.

[55] Jacob Kahn, Morgane Rivière, Weiyi Zheng, Evgeny Kharitonov, Qiantong Xu, Pierre-Emmanuel Mazaré, Julien Karadayi, Vitaliy Liptchinsky, Ronan Collobert, Christian Fuegen, Tatiana Likhomanenko, Gabriel Synnaeve, Armand Joulin, Abdelrahman Mohamed, and Emmanuel Dupoux. Libri-light: A benchmark for asr with limited or no supervision. 2019.

[56] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2009.

[57] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification, 2016.

[58] tensorflow/models. Models for audioset: A large scale dataset of audio events. https://github.com/tensorflow/models/tree/master/research/audioset/vggish, 2021.

[59] Facebook AI Research. Wav2vec example in fairseq. https://github.com/facebookresearch/fairseq/blob/main/examples/wav2vec/README.md, 2021.

[60] Simon Haykin. *Neural Networks and Learning Machines*. Papasotiriou, 3rd edition edition, 2010. Translator: Gkagkatsiou, Eleni.

[61] Liyun Gong, Miao Yu, Vassilis Cutsuridis, Stefanos Kollias, and Simon Pearson. A novel model fusion approach for greenhouse crop yield prediction. *Horticulturae*, 9(1):5, dec 2022.