UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

MOVIE SHOT TYPE RECOGNITION USING DEEP LEARNING

Diploma Thesis

Antonia Petrogianni

Supervisor: Panagiota Tsompanopoulou

February 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

MOVIE SHOT TYPE RECOGNITION USING DEEP LEARNING

Diploma Thesis

Antonia Petrogianni

Supervisor: Panagiota Tsompanopoulou

February 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΝΑΓΝΩΡΙΣΗ ΤΥΠΟΥ ΚΙΝΗΜΑΤΟΓΡΑΦΙΚΩΝ ΣΚΗΝΩΝ ΜΕ ΧΡΗΣΗ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ

Διπλωματική Εργασία

Αντωνία Πετρόγιαννη

Επιβλέπουσα: Παναγιώτα Τσομπανοπούλου

Φεβρουάριος 2022

Approved by the Examination Committee:

Supervisor **Panagiota Tsompanopoulou**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Theodoros Giannakopoulos**
Researcher (B), National Center for Scientific Research "Demokritos"

Member **Athanasios Fevgas**
Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

This diploma thesis was completed in the Computational Intelligence Laboratory (CIL) of the Institute of Informatics and Telecommunications of the National Centre for Scientific Research “Demokritos”.

Throughout the writing of this diploma thesis I received a lot of help and encouragement. First and foremost I want to express my gratitude and appreciation to Mr. Theodoros Giannakopoulos, Researcher (B) of NCSR Demokritos, whose insight and knowledge into the subject steered me through this research. I would also like to thank Mr. Panagiotis Koromilas, Research Assistant at the NCSR Demokritos, for his invaluable advise and continuous support. Without their guidance, this task would have been much more difficult.

Of course I would like to thank Mr. Elias Houstis, Professor Emeritus at University of Thessaly, who introduced me to the field of Data Science. I also appreciate all the support I received from my thesis supervisor, associate professor Dr. Panagiota Tsompanopoulou, as well as all of the help I received over these five years at the University of Thessaly. I also thank Mr. Athanasios Fevgas for the support.

But, most importantly, I want to express my heartfelt gratitude to my family, who has always supported and helped me to accomplish my goals and dreams!

The best is yet to come...

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Antonia Petrogianni

Diploma Thesis

MOVIE SHOT TYPE RECOGNITION USING DEEP LEARNING

Antonia Petrogianni

Abstract

Visual information contains the most important characteristics of a movie regarding the related content and filming techniques. Especially the way the camera moves to capture the scene is vital to define the director's aesthetics. However, most of the machine learning tasks existing in the literature treat the movie as shallow content, rather than as an artistic work, and therefore focus on detecting objects and faces, recognizing activities and extracting plot-related topics. On the other hand, cinematography is closely connected to the choice of different ways to handle the camera, and thus camera movements include information that is useful in order to analyse the artistic style of a movie. In this work we present an original, publicly available* dataset for film shot type classification that is associated with the distinction across 10 types of camera movements that cover the vast majority of types of shots in real movies. In addition, two different methods are evaluated on the new dataset, one static that is based on feature statistics across frames, and one sequential that tries to predict the target class based on the input frame sequence using LSTMs. According to the evaluation process it is inferred that the sequential method is more suited for modeling the camera movements.

Keywords:

shot classification, camera movement classification, movie analysis, machine learning, deep learning, LSTM

*https://github.com/magcil/movie_shot_classification_dataset

Διπλωματική Εργασία

ΑΝΑΓΝΩΡΙΣΗ ΤΥΠΟΥ ΚΙΝΗΜΑΤΟΓΡΑΦΙΚΩΝ ΣΚΗΝΩΝ ΜΕ ΧΡΗΣΗ ΒΑΘΙΑΣ ΜΑΘΗΣΗΣ

Αντωνία Πετρόγιαννη

Περίληψη

Οι οπτικές πληροφορίες περιέχουν τα πιο σημαντικά χαρακτηριστικά μιας ταινίας όσον αφορά το περιεχόμενο και τις τεχνικές κινηματογράφησης. Ειδικά ο τρόπος που κινείται η κάμερα για να απαθανατίσει τη σκηνή είναι καθοριστικός για τον καθορισμό της αισθητικής του σκηνοθέτη. Ωστόσο, οι περισσότερες από τις έρευνες που υπάρχουν στη βιβλιογραφία με χρήση μηχανικής μάθησης αντιμετωπίζουν την ταινία ως ρηχό περιεχόμενο, παρά ως καλλιτεχνικό έργο, και ως εκ τούτου επικεντρώνονται στην ανίχνευση αντικειμένων και προσώπων, στην αναγνώριση δραστηριοτήτων και στην εξαγωγή συμπερασμάτων που σχετίζονται με την πλοκή. Από την άλλη πλευρά, η κινηματογραφική τέχνη συνδέεται στενά με την επιλογή διαφορετικών τρόπων χειρισμού της κάμερας. Έτσι οι κινήσεις της κάμερας περιλαμβάνουν πληροφορίες που είναι χρήσιμες για την ανάλυση του καλλιτεχνικού στυλ μιας ταινίας. Σε αυτή τη διπλωματική εργασία παρουσιάζουμε ένα πρωτότυπο, δημοσίως διαθέσιμο σύνολο δεδομένων[†] για την ταξινόμηση του τύπου πλάνων ταινιών. Συγκεκριμένα στοχεύουμε στη διάκριση μεταξύ 10 τύπων κινήσεων της κάμερας που καλύπτουν τη συντριπτική πλειοψηφία των τύπων λήψεων σε πραγματικές ταινίες. Επιπλέον, δύο διαφορετικές μέθοδοι αξιολογούνται στο νέο σύνολο δεδομένων, μία στατική που βασίζεται σε στατιστικά χαρακτηριστικά σε όλα τα καρέ ενός βίντεο και μία διαδοχική που προσπαθεί να προβλέψει την κλάση με βάση την ακολουθία καρέ ενός βίντεο, με τη χρήση LSTMs. Σύμφωνα με τη διαδικασία αξιολόγησης των μεθόδων συμπεραίνουμε ότι η διαδοχική μέθοδος είναι πιο κατάλληλη για τη μοντελοποίηση των κινήσεων της κάμερας.

Λέξεις-κλειδιά:

ταξινόμηση πλάνων, ταξινόμηση κινήσεων κάμερας, ανάλυση ταινιών, μηχανική μάθηση, βαθιά μάθηση, LSTM

[†]`https://github.com/magcil/movie_shot_classification_dataset`

Table of contents

List of figures

List of tables

Abbreviations

ie.	id est
e.g.	exempli gratia
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
LSTM	Long Short Term Memory
ML	Machine Learning
NLP	Natural Language Processing
RGB	Red Green Blue
RNN	Recurrent Neural Network

Chapter 1

Introduction

A movie shot is a *"series of frames that runs for an uninterrupted period of time"* [18]; a single take that can last from a few seconds to several minutes. Movie producers use a variety of video shot types to draw the audience's attention and enhance their viewing experience. As a result, video shot classification is critical for comprehending and accessing the movie's content [19]. Since there are so many different movie genres, shooting techniques, and shot types, movie shot classification is a necessary but difficult task.

Both Machine and Deep learning methodologies have been greatly applied in order to conduct movie analysis. Extensive research has been published on different kinds of problems such as movie reccomentation [20] [21] [22] [23], movie summarization [24] [25] [26], movie genre classification [27] [28] [29] or movie scene detection [30] [31] [32]. These problems have been faced using different types of approaches *(e.g., traditional machine learning or deep learning architectures)* or different types of representations *(e.g., multimodal recommendation [33] or summarization [34])*.

Most of the existing tasks in computer vision primarily focus on content rather than style understanding. That is the analysis of movies, which are a work of art, cannot be properly conducted when excluding cinematographic style. Shot type classification could enrich movie analysis techniques and result in a more inclusive automated movie understanding.

The existing datasets presented in the works described above, are either private or do not focus on camera movements. However, camera movement is important for expressing mood and style in a movie, and as such, it is crucial to be considered as an attribute in any movie analysis system: movie indexing and search, movie visualization and, of course, movie recommendation systems. To put it simple, the way the directors decide to move their cameras

makes us, the viewers, like or not a movie, among other characteristics.

In this thesis, we propose a publicly available dataset that contains shots coming strictly from movies. The proposed collection is concerned about camera movements and proposes the categorization of shots in the following extensive 10 classes: *static, panoramic, zoom-in, travelling-out, vertical movement, aerial, travelling-in, tilt, handheld, panoramic lateral.* We report classification metrics on our dataset based on *(i)* a static method which is based on aggregated statistics on the feature sequence, and *(ii)* a sequential method where an LSTM is applied directly on the sequential features, which is not the usual case in the literature since most of the existing works depend on feature aggregations [35].

The rest of this thesis is organized as follows. In Chapter 2 a detailed theoretical presentation is provided in order to comprehend the fields of machine and deep learning, which serve as the cognitive background for our model's implementation. In Chapter 3 several related works are discussed in the first place. The dataset compilation along with the annotation procedure is discussed. The feature extraction process, as well as the proposed methodologies are also explained in this chapter. In Chapter 4 the experimental results are provided, where Chapter 5 concludes our work and proposes some future work directions.

Chapter 2

Background

2.1 Artificial Intelligence

Artificial intelligence (AI) is an application in computer science and information technology related to the development of intelligent machines to accomplish activities that would require human intellect. That is, systems that mimic human actions [36]; Systems that not only act like human beings, but systems that also think rationally. They are machines that can do a variety of human-like tasks in terms of achieving a specific goal. Besides, the word itself says so. With the word *'intelligence'* we refer to the computational component of one's capacity to perform tasks in the real world.

AI possesses tremendous advantages and is constantly evolving in a way that enhances our quality of life. It affects our everyday lives, since it's used in a variety of applications; including recommender systems, self-driving cars, image classification, human speech recognition, mathematics, computer vision, and more [37]. *Machine Learning* and *Deep Learning*, both subcategories of AI (Figure 2.1), were employed in this thesis. The subchapters that follow examine and explain these subjects.

2.2 Introduction to Machine Learning

With such an exponential growth in AI, Machine learning (ML) is becoming one of the most popular fields of the century. It is one of AI's branches and it investigates computer systems that learn and evolve based on data over time. Observations, data analysis, and past experience, all contribute to the learning process and to the predictions made. When we talk

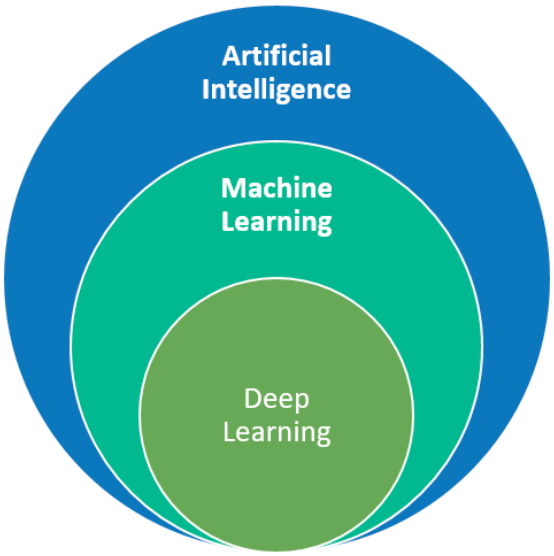


Figure 2.1: Machine and Deep Learning as sub-fields of AI[1]

about *'past experience'*, we refer to previously accessible knowledge, or data that has already been highlighted and labeled. The accuracy of the predictions, is dependent on the quality and quantity of data. Essentially, computers seek patterns in data in order to make better decisions in the future. The ultimate goal is for them to learn and adapt their behavior to the circumstances, without the need for human involvement. Machine Learning algorithms have had a lot of success in a variety of applications, especially in cases where manually built techniques to address issues is challenging; such as *Computer Vision* and *Natural Language Processing*.

Computer vision is a field, in which machines learn to study and analyze digital multimedia. Its goal is to understand and automate operations that a human is capable of doing. This involves strategies of acquiring, processing and analyzing, as well as techniques for extraction of data to produce valuable insight [38].

Natural language processing (NLP) constitutes a subject of computer science. Here, the computer systems use software to automate the manipulation of natural language. It is built on the capability to interpret both written and spoken words in the same manner a human-being does [39] [40]. It is a subset of linguistics.

We will not deal with text or words in this research; instead, we will focus on extracting information from videos.

2.2.1 Types of Machine Learning

An interesting topic in ML is the categorization into three subfields: Supervised, Unsupervised, Reinforcement Learning. We will discuss on when to use the algorithms of each category along with their applications (Figure 2.3).

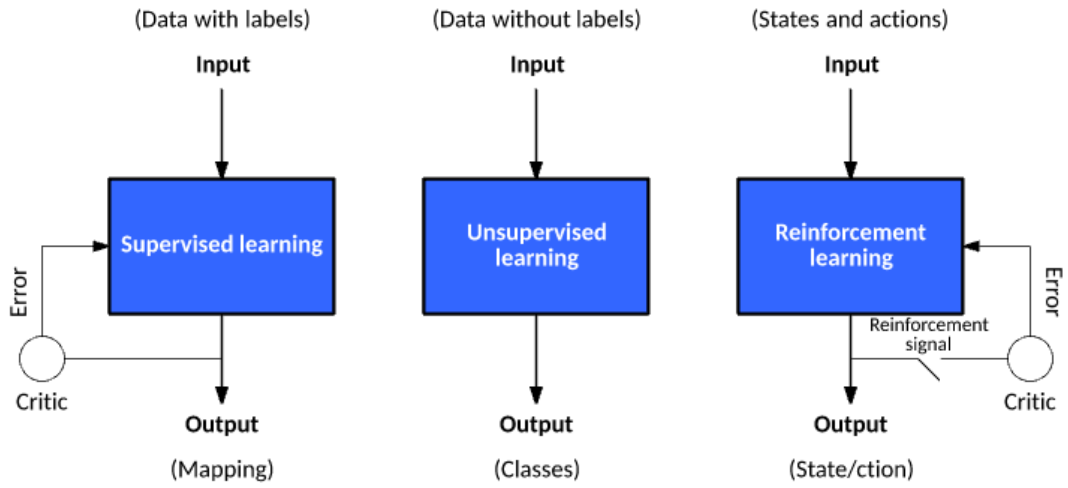


Figure 2.2: Distinction between types of machine learning [2]

Supervised Learning

One of the most fundamental forms of machine learning is supervised learning. This method needs labeled data -which means that the output is known- to train models. The algorithm will then forecast the label for each case and provide feedback on whether it anticipated the right answer or not. By this process and after identifying new patterns and correlations, the model will improve. In the end, the algorithm will be able to predict the output of new data that has never seen before [41].

Supervised algorithms are split into two main categories: *Regression* and *Classification*. In the former case, datasets contain continues values and the algorithms work by estimating how one variable influences the other. In the latter case, the idea is to forecast discrete values. There are two significant categories of classification problems to examine: *binary classification*, where the instances of a task are classified into two categories (e.g., if something is true or not true), and *multi-class classification*, where the instances of a task are classified to more than two different categories [42]. In this thesis, we are going to provide results on experiments for both binary and multi-label classification.

Unsupervised Learning

The definition of the term *'Unsupervised'* is to act without being overseen or without somebody's guidance. This word here refers to the absence of labeled data. In this type of learning, unlabeled data are getting analyzed and clustered into groups either because they have similarities or due to their differences. The machines must discover features and patterns in order to forecast the output [43].

Reinforcement Learning

A number of software and computers employ this type of learning to find the best possible route to take in a specific scenario. The algorithms learn to attain a goal by either receiving a reward for a successful move or receiving a punishment for acting incorrectly; e.g., an algorithm can learn, by playing a game repeatedly, until the maximum points of the game are gained [44]. This domain of learning permits agents to fulfill a task while increasing the value of a numerical reward signal, until it gets the maximum value.

Figure 2.2 depicts in which type of learning labeled and unlabeled data are being used, and when agents need to take actions in order to boost the performance of the models. The separation of the most popular applications between the different kinds of Machine Learning is illustrated in Figure 2.3.

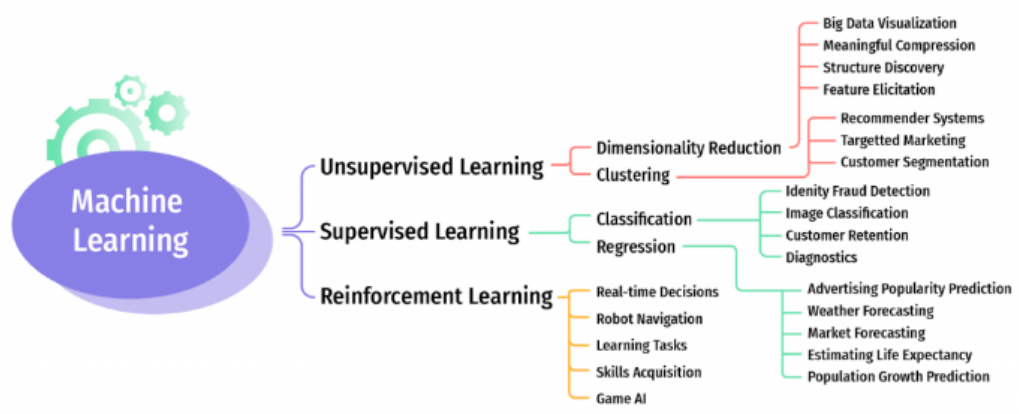


Figure 2.3: Types of Machine Learning[3]

2.2.2 Supervised Model performance assessment

Any project should include an evaluation procedure. The data should be divided into three unique splits for this purpose. The basic idea is to split them into *training*, *validation* and *test* sets.

First of all, in the *training* process, the algorithm is fed input data which correlate to specific outputs. Then, the model examines the data regularly and tries to find hidden features and patterns in order to get a better understanding of the data's behavior. It adjusts to achieve the model's aim. The training set should always comprise a broad range of inputs, so that the model may be trained in all possible scenarios and forecast any unseen future data samples.

The *validation* set is a small subset of the original dataset, which is independent from the training set described above. This set serves as an initial test against unknown data and provides information that we could take advantage of and use in optimizing the model's hyperparameters and settings. The major goal we want to achieve using this set is to avoid overfitting, which occurs when a model becomes extremely effective at categorizing samples in the training procedure, but is unable to generalize and make accurate predictions on unseen data from the test set later on.

The *test* set is a distinct collection of data that is used to put the model through the process of making predictions, once it has been trained and validated [45][4]. The illustration in Figure 2.4 helps us understand better how these sets work.

The test set provides an answer to our queries; if the model performs good enough, if we can verify that the algorithm was properly trained and if it could predict new unseen data. But how do we understand if the model performs well or not? The *evaluation metrics* will provide us with such information.

Confusion Matrix

In an either binary or multi-label classification task, the *confusion matrix* is a graphic depiction of actual versus predicted values; it is a performance metric that gives a quick overview of the forecast outcomes [7]. It is also called *'error matrix'*. It represents the four different combinations of actual and predicted values, as shown in Figure 2.5.

Let's examine what the components of the confusion matrix are:

- **TP (True Positive):** The positive values were correctly forecasted as positive ones

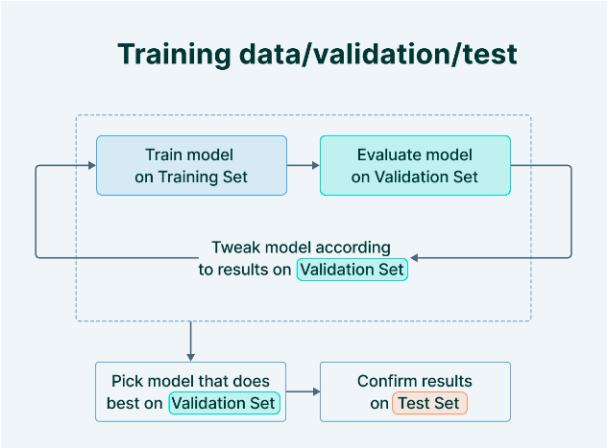


Figure 2.4: Training, validation and test procedure [4]

- **FN (False Negative):** The positive values were mistakenly forecasted as negative ones
- **FP (False Positive):** The negative values were mistakenly forecasted as positive ones
- **TN (True Negative):** The negative values were correctly forecasted as negative ones

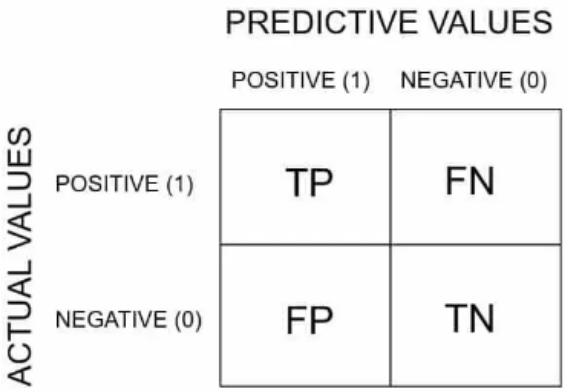


Figure 2.5: Confusion Matrix for the binary task [5]

Whether the problem is a complex one or if the classes are imbalanced, we cannot always rely on the confusion matrix; there are multiple metrics we may use to evaluate the performance of our model. We are going to refer to some of them below.

Accuracy

The equation below (Figure 2.6) shows how many of the total instances were correctly forecasted. Although the level of accuracy should be high, using this metric as the primary indicator to assess our model's performance will fail in the scenario of imbalanced data.

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

Figure 2.6: Accuracy [6]

Precision

The equation in Figure 2.7 shows how many of the total instances that were forecasted as positive were truly positive ie. how many times the model is precise when predicting a value as a positive one? The level of precision should be as high as feasible. Let's say that we have to identify if a person has cancer or not. If our model's level of precision is unsatisfactory, then many people will be mistakenly informed that they have cancer [46]].

$$\textit{Precision} = \frac{TP}{TP + FP} \text{ or } \frac{\text{True Positive}}{\text{Predictive Results}}$$

Figure 2.7: Precision [5]

Recall or Sensitivity

The equation below (Figure 2.8) shows how many instances were correctly forecasted as positive out of a total number of positive outcomes; ie. out of all the positive values, how valid are the outcomes? The level of recall should be high as well. For the example presented above, if our model's level of recall is poor, then many people will be mistakenly informed that they do not have cancer, when they do have.

$$\textit{Recall} = \frac{TP}{TP + FN} \text{ or } \frac{\text{True Positive}}{\text{Actual Results}}$$

Figure 2.8: Recall [5]

F-measure

The F-score (Figure 2.9) is a tool for assessing both recall and precision at the same time and that's why it is also characterized as the harmonic mean of those values. A strong F1 score indicates that the FP and FN rate are both low. As a result, we come to the conclusion that the F1-score should be as high as possible. When applying the macro-averaged F1-score

as our criterion for assessing the effectiveness of our model, each label will be given the same priority. A high macro-F1 score means that our model works well not just for the neutral classes, but also for the non-neutral ones. This would be useful when dealing with an imbalanced dataset [6] [47].

$$F\text{-}score = \frac{2 * Recall * Precision}{Recall + Precision}$$

Figure 2.9: F1-score [7]

2.2.3 Fundamental Machine Learning Algorithms

Linear Regression

A collection of input variables *(x)* that is utilized to calculate an output variable *(y)*. These variables have a connection and the purpose is to quantify this connection. By fitting the optimal line, we may establish a link between the independent *(input)* and dependent *(output)* variables. The aim is to find one that fits the majority of the points the best [8]. It is represented by the linear equation $Y = a * X + b$ (Figure 2.10). The purpose of linear regression is to determine the coefficients a *(intercept)* and b *(slope)*.

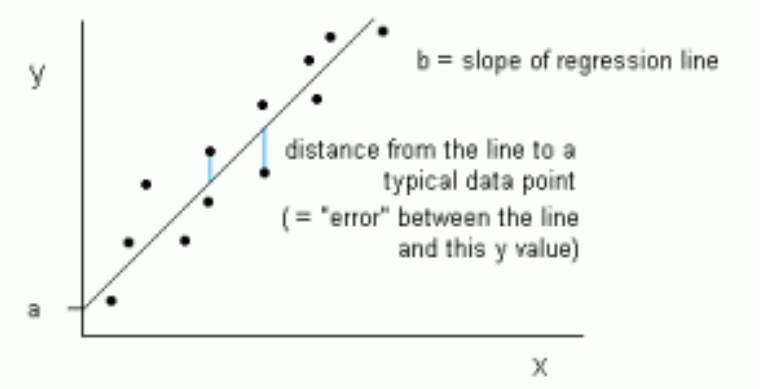


Figure 2.10: Linear Regression represented as a line [8]

There are two forms of linear regression: *simple linear regression* and *multiple linear regression*. One independent variable characterizes the former, while multiple independent variables characterize the latter.

Decision Trees

A supervised learning method that's extensively used to solve both regression and classification tasks. An example is illustrated in Figure 2.11. It has the structure of a tree, as it's name implies, where *internal nodes* contain the dataset attributes (ie. the condition), *branches* represent decision rules, and their end, which cannot be split anymore represents a *leaf* node [48]. These nodes are the result of those decisions and do not include any branches [8].

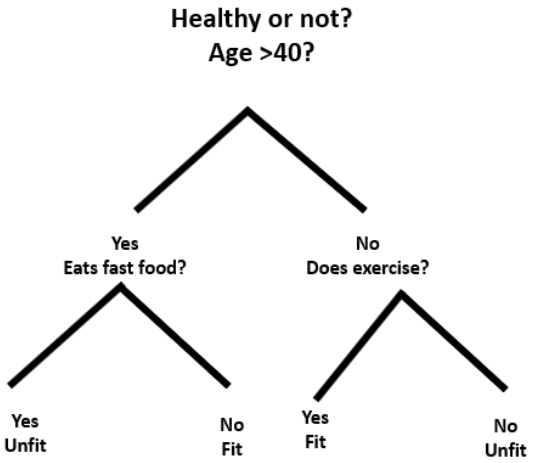


Figure 2.11: A Decision tree example [9]

Choosing which characteristics to employ and which conditions to use for splitting, as well as understanding when to stop, are all parts of the process of growing a tree. They are generally easy to understand, but also complicated, since they have several layers.

Support Vector Machine

A method of classification. Support Vector Machine (SVM) is an algorithm responsible for determining the decision boundary that will divide the various groups and maximize the margin. A margin shows how far the data, as shown in Figure 2.12, are from the line. SVM must determine the ideal line to separate the data of each group. This will be the line along which the distances between the two groups' nearest points are the greatest [49][50]. This is how the optimal hyperplane is created.

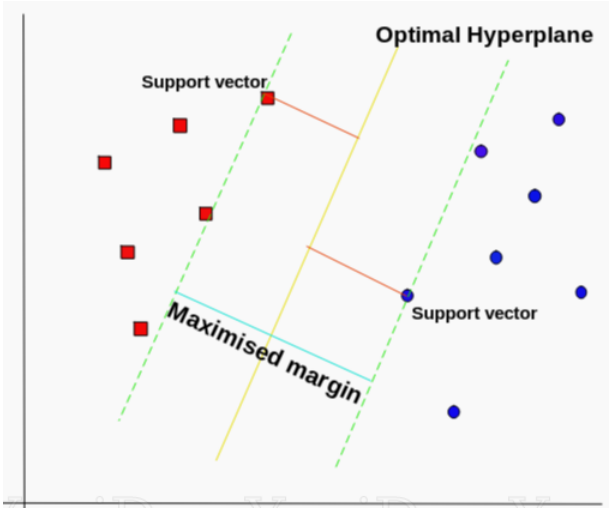


Figure 2.12: Support Vector Machine - Optimal Hyperplane [10]

2.3 Deep Learning

The line between *Deep Learning* and Machine Learning is, in some ways, hazy. Deep learning is a machine learning approach that allows computers to learn by acting in the same way that people do; they imitate how people acquire knowledge. It deals with *artificial neural networks*, which are algorithms inspired by the structure and function of the brain. A computer model in this type of learning, executes categorization tasks directly from pictures, texts, audios or videos. Many AI applications and services rely on deep learning to enhance automation by performing analytical and physical activities without the need for human intervention. These techniques have vastly improved the state-of-the-art in voice recognition, visual object identification, and a variety of other domains. Models are trained by utilizing massive quantities of labeled data [51][52]. But how all these things work?

2.3.1 Artificial Neural Networks

Let us imagine we need to create a model that can tell if a vehicle is a car or a bus. In typical machine learning we instruct the computer what sorts of things it should search for, to determine if a picture contains either a car or a bus. Deep learning offers the benefit of creating feature vectors automatically without the need for human intervention.

Initially, the program will be given training data, which will consist of a set of photographs classified by a person, indicating whether it is a car or a bus. Then feature vectors for the vehicles are created and a prediction model using the information is built. In this scenario,

the computer's first model may suggest that everything in the picture that consists of wheels and windows should be identified as a car. Naturally, the software is not aware of the terms 'wheels' and 'windows'. It will just search the digital data for pixel patterns. After each iteration, the prediction model becomes more and more accurate [51]. This is the general idea, but let's explore how exactly this is done.

2.3.2 Architecture of Artificial Neural Networks

The term *"neural networks"* along with their structure are derived from the human brain, and they resemble the way real *(human)* neurons communicate. Neural networks are multi-layer networks of neurons which attempt to mimic the human brain and discover correlations between the data they have to process [53].

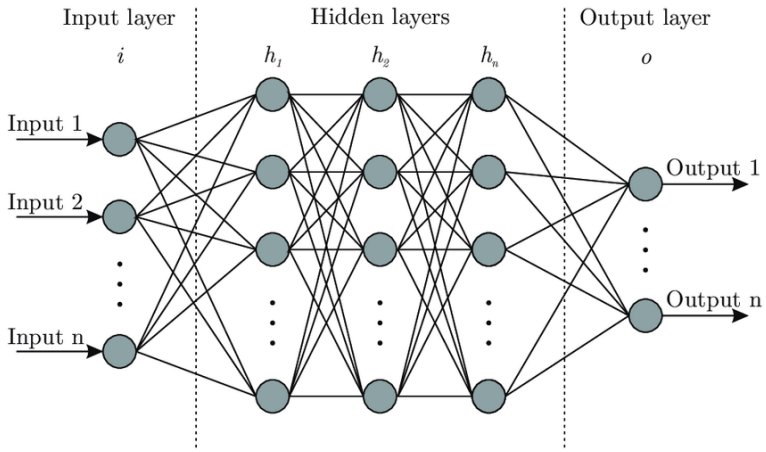


Figure 2.13: Artificial Neural Network Architecture [11]

Artificial neural networks (ANN) consist of an input and an output layer and, in between, one or more hidden layers, as illustrated in Figure 2.13. These networks are made up of layers of neurons. The artificial neurons (or nodes) are the core processing units of the network. The input layer, which connects with hidden layers, feeds data into a neural network. A node, which is a small unit, is connected to the others and has a weight and threshold attached to it. The hidden layer's nodes receive input signals, that are multiplied by the *'weights'* of the connections they pass through [54]. They perform some processing and calculations with the use of *"activation functions"*, which specify how far a signal must travel through the network to effect the final output. If the node's output reaches a specific threshold, the node gets 'activated', and sends the data to the next unit connected to it. Finally, the output layer is where the outputs of the last hidden layer are collected [55].

Weights are the links between units and they may be positive or negative; they reflect how much of an impact a previous unit's input has on the output of the next unit. Altering a model's weights is the primary method that networks are taught. Throughout testing, they are fixed, but during training, we will modify these quantities in order to 'tune' our network. When compared to features with larger weights, features with weights close to zero are deemed to be less important in the prediction process. A neural network learns when given feedback about whether it took the right action or not. The network will make modifications based on the feedback to address the problem [54] [56].

Activation Functions

They are an important aspect of a neural network's architecture. Transfer Functions is another name for them. They determine if a neuron should be activated. This means that they will use mathematical operations to identify whether the neuron's input to the network is essential. The type of predictions the model may produce is determined by the activation function used in the output layer [57]. The principal aim is to convert the node's summed weighted input into an output value that may be passed to the next hidden layer or used as output. The Activation Functions may be classified into two categories: *Linear Activation Function* and *Non-linear Activation Functions* [12].

- **Linear or Identity Activation Function**

  It is also known as a straight-line function, where the activation is proportional to the input, which is the weighted sum of neurons. It does not affect the input's weighted sum in any way and simply returns the value; it does not capture complex patterns [58] [59]. It has a straightforward function based on the equation: $f(x) = ax + c$ (Figure 2.14).

- **Non-linear activation functions**

  - **Sigmoid**

    The sigmoid function's curve is S-shaped, as its name implies. It transforms data in the range of 0 to 1 (Figure 2.15). It is extensively used when our model must predict probabilities. It is continuously differentiable and is also non-symmetric near zero. As a result, all of the neurons' output will be of the same sign. From its

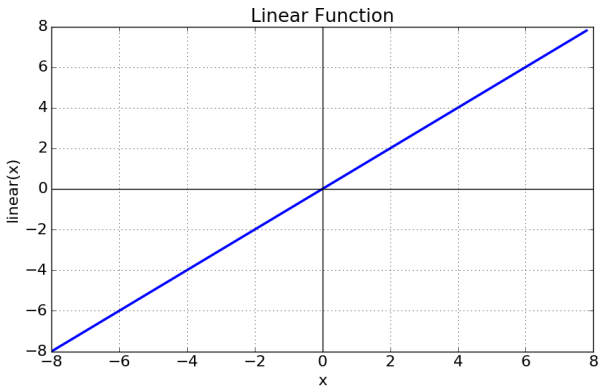


Figure 2.14: Linear Activation Function [12]

derivative we observe that for ranges -3 and 3, the gradient values are considerable, but the graph becomes significantly flatter. This means that gradients will be very modest for values more than 3 or less than -3. The network is not truly learning when the gradient value approaches zero [58].

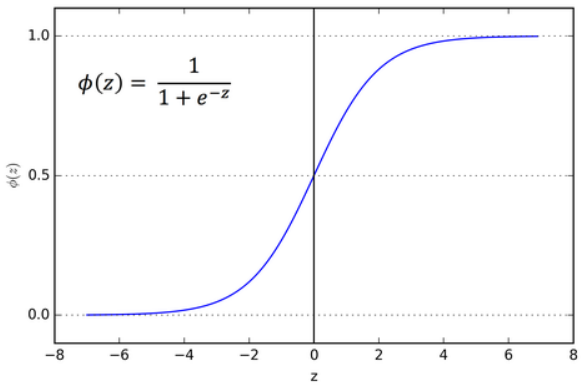


Figure 2.15: Sigmoid Function [12]

– **Tanh or hyperbolic tangent**

The tanh function and the sigmoid function are quite similar; tanh also has an S-shaped structure. It is symmetric around the origin, which is where these functions differ. The output is in the range of -1 to 1. It has a non-negative derivative at each point and is defined for all real input data. The diagram below (Figure 2.16) shows how it is plotted.

– **ReLU**

The benefit of employing the ReLU function is that it does not simultaneously stimulate all of the neurons. This means they'll only be silenced if the outcome of

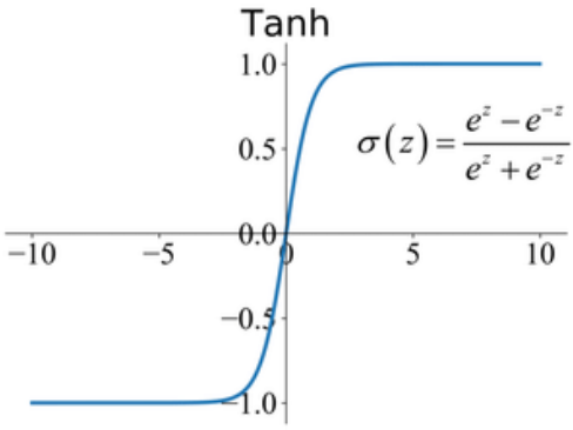


Figure 2.16: Tanh or hyperbolic tangent function [13]

the linear transformation is less than zero. As we can see from the diagram below (Figure 2.17), if the input value is negative, then it is returned as zero; otherwise, the same value is returned. When compared to the sigmoid and tanh functions, it is considerably faster since only a small number of neurons remain active [60].

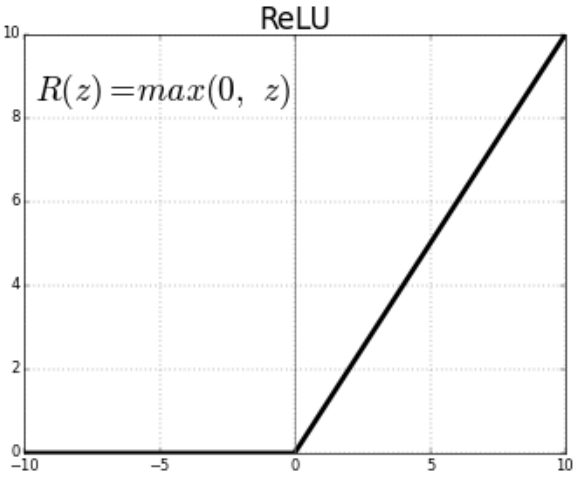


Figure 2.17: ReLU Function [12]

– **LeakyReLU**

This function is a better variant of the ReLU function. The gradient for the ReLU function is 0 for negative values, as we observed, deactivating the neurons in that area. Instead of the value being zero, a tiny slope is now added in the negative range. As a result, there would be no more dead neurons in that area. It is depicted in Figure 2.18. The mathematical representation is: $f(x) = max(0.05x, x)$ [61].

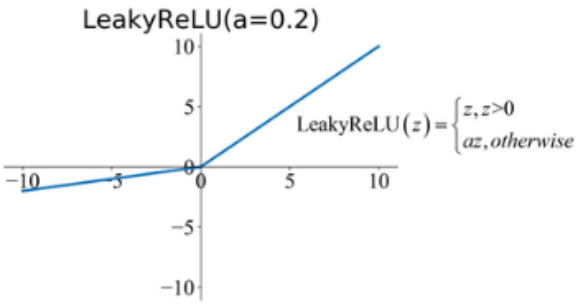


Figure 2.18: LeakyReLU Function [13]

Loss Function

One of the most significant components of Neural Networks is the *Loss Function*. It is the method for calculating the prediction error. It demonstrates how much the predicted from the real values differ. *Optimizers* are also used to reduce the loss and improve predictions' accuracy. The gradients, which are used to update neural network's weights, are calculated using the loss function [62]. It is how the Neural Network is getting trained and we can obtain information on how well or poorly the model is doing.

- **Categorical Cross Entropy Loss**

  In multi-label classification tasks, cross-entropy-based loss functions are often utilized. The difference between two probability distributions is measured by cross entropy [62]. It operates in such a way that when the predicted probability approaches the ground truth, the loss diminishes. The last layer's output should go through a softmax activation, giving each node a probability value between [0-1]. The loss is calculated according to the formula (2.1).

$$L_{CE} = -\sum_{i=1}^{n} y_i \log(p_i), \tag{2.1}$$

  where y_i is the actual value (0 or 1) and p_i is the corresponding forecasted value for the i^{th} class.

- **Binary Cross Entropy Loss**

  This is used in binary classification tasks. The binary loss is calculated according to the formula (2.2), which results from the above equation (2.1) for N=2 [63]. The output value should be processed via a sigmoid activation function, with an output range between [0-1].

$$BCE = -(y \log(p) + (1 - y) \log(1 - p)), \tag{2.2}$$

where y is the actual value (0 or 1) and p is the corresponding forecasted value.

- **Mean Squared Error Loss**

  For regression tasks, MSE() loss is employed. This is estimated by calculating the mean of squared discrepancies between actual and forecasted values (2.3). Because MSE is sensitive to outliers, it would be preferable that the actual values are normally distributed around a mean value [63].

$$L_{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - p_i)^2, \tag{2.3}$$

  where y_i is the actual value (0 or 1) and p_i is the corresponding forecasted value for the i^{th} class.

Optimization

As it was mentioned previously, a small loss indicates that our model will perform better. *Optimization* is the process of reducing the loss. Optimizers are techniques or approaches that adjust some of the characteristics of a neural network, such as *weights* and *learning rate*, to decrease the loss. Optimization algorithms are in charge of minimizing losses and delivering the most accurate outcomes [64].

There are many different types of optimizers: **Gradient Descent**, is widely used in linear regression and classification techniques. It indicates which way the weights should be modified to bring the function closer to a minimum. The gradient descent technique is also used in backpropagation in neural networks. It may be easy to compute and to implement, but it is possible that it gets 'trapped' at a local minimum. Also, only after computing the gradient on the entire dataset, the weights get updated [65]. This means that if the dataset is too vast, the process may take even years to be completed. **Stochastic Gradient Descent**, tries to update the parameters of the model more often in comparison to the former. After each training example's loss has been computed, the model parameters are modified. Model parameters here are updated often, resulting in faster convergence. **Adam**, retains a learning rate for each network parameter and modifies them individually during training. It adjusts the learning rate for each weight based on estimates of the first and second moments of gradient [66].

Backpropagation

The core of neural net training is backpropagation. The chain rule is used by the backpropagation method in ANNs to determine the gradient of the loss function for a single weight. What exactly happens is after the feed-forward process discussed above, when reaching to the output layer of the network, the loss is fed backwards to the hidden layers. This will result to modified weights in such a way so the error decreases [67].

Dropout

Dropout refers to the practice of disregarding neurons during the training phase of a randomly selected group. By "disregarding", we mean that forces a neural network to learn more robust characteristics that can be applied to a number of different random subsets of other neurons [68] . These units are not taken into account during a forward or backward pass. It is a strategy for decreasing overfitting. Large networks are time-consuming, making it difficult to avoid overfitting by merging predictions from several neural networks at test time. Dropout is dealing with this issue. This method has been discovered to increase the performance of neural nets in a range of applications as stated in [69]. A neural network before applying Dropout and after applying it is shown in Figure 2.19

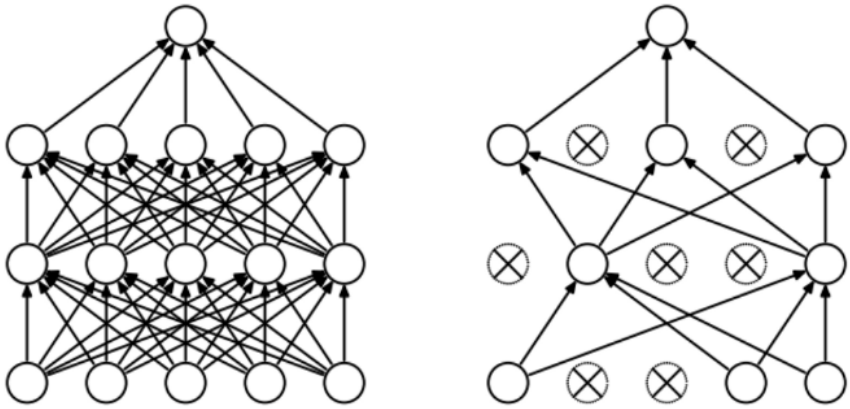


Figure 2.19: Dropout Neural Network Model [14]

Normalization

Deep neural networks with many layers are difficult to train because they are sensitive to the learning algorithm's initial random weights and setup. *Normalization* is a data pre-

processing technique for converting numerical data to a common scale without changing the form of the data, to guarantee that our model can generalize correctly [70]. Regularization techniques aid in the improvement of a model and allow it to converge more quickly and prevents the model from over-fitting.

Batch-Normalization, entails the use of the current batch's mean (μ) and variance (σ^2) to normalize activation vectors from hidden layers.It speeds up the training process by normalizing the hidden layer activation [71]. Batch normalization smooths the loss function and makes the model more consistent, while eliminating the requirement for *Dropout* [72].

2.3.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a type of neural network that specializes in processing data with a grid-like architecture, such as images and videos. A binary representation of visual data is a digital picture. It consists of a grid-like arrangement of pixels with pixel values.

A convolutional layer, a pooling layer, and a fully connected layer are the three layers that compose a CNN, as illustrated in the Figure 2.20.

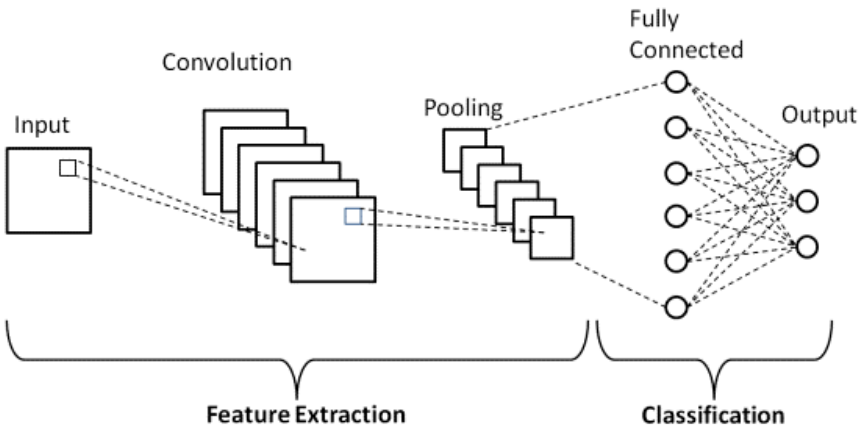


Figure 2.20: Convolutional neural Network [15]

In convolutional neural networks, the fundamental building elements are *convolutional layers*. This is the initial layer that extracts the different characteristics from the input photos. The convolution mathematical operation is done between the input picture and a filter *or 'kernel'* of a certain size $M * M$ in this layer. The dot product between the filter and the input picture is calculated and a feature map is created displaying the positions and intensity of a recognized feature in an input. This is then supplied to the next layers [73] [74].

The *Pooling layer* aims to decrease the size of the map and it is accomplished by reducing the connections between layers. There are numerous sorts of Pooling procedures: such as Max Pooling, Average Pooling, Sum Pooling. How they are chosen depends on the mechanism utilized. The preceding layers' input images are flattened and supplied to the *Fully Connected Layer* layer. The classification procedure follows [75].

2.3.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are the state-of-the-art method for time-series or sequential data due to their internal memory. Recurrent neural networks (RNNs) are named like this, because they complete the same function for each element of a sequence, with the result being dependent on the prior computations [76]. They include loops in them that allow data to endure.

All of the inputs are interconnected [77], as we can see from the Figure 2.21.

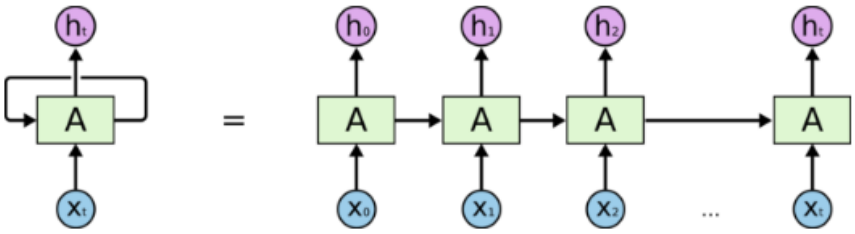


Figure 2.21: An unrolled RNN [16]

One of the allures of RNNs is the possibility of connecting earlier data to the current task, such as using previous video frames to inform comprehension of the current frame. But are they capable to do so? They work just fine when the problem consists of short-term dependencies. They do not have the ability to keep valuable insight in memory for a long period of time though. The solution to this problem was the improved RNNs, called Long Short-Term Memomry (**LSTM**) networks, that make it simpler to recall past information in memory [16]. They consist of 3 gates: *Forget Gate*, *Input gate* and *Output Gate* as illustrated in Figure 2.22.

The Input gate is responsible for how the memory will be modified. The *Sigmoid* function will decide which data will pass to next layers and the *tanh* function will assign a weight to the data, based on how significant they are to the network.

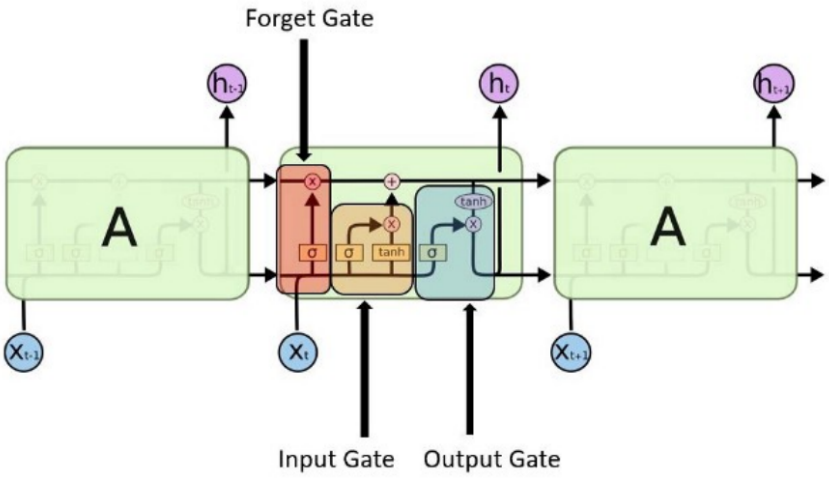


Figure 2.22: LSTM gates [17]

The Forget gate figures out which data should not be kept. The Sigmoid function is responsible for this process, looking both at the previous state and the input, it produces a number in the range [0, 1]. It omits the number if it's zero, but maintains the number if it's one.

The Output gate will be determined by the block's input and the memory. The Sigmoid function is responsible of which values are allowed through, while tanh function assigns weight to values that pass through [77].

In this project we have implemented an LSTM network, using sequences of visual features, in order to classify movie shots.

2.4 Hyperparameter Tuning

Hyperparameter tuning will be mentioned many times in the following chapters, since it is an important procedure. A hyperparameter is a parameter whose value is determined prior to the start of the learning process. The main difference between the model parameters and the hyper-parameters is that the former are acquired directly from the data, while the latter are the parameters that specify the model architecture. The height of the tree in a decision tree algorithm, the parameter 'C' in an SVM algorithm, the learning rate, the batch size in a neural network; these are all hyperparameters. Default hyperparameter settings cannot ensure the optimal performance of the algorithms. Therefore, the process while these are getting fine-

tuned in order to find the best architecture is called *hyperparameter tuning* [78]. Because this process can be time-consuming, techniques such as grid search and random search are employed [79].

2.5 Cross-Validation process

To construct a machine learning model, we frequently divide the dataset into train, validation and test sets of data as mentioned previously. But this is often done randomly. Because the model's accuracy fluctuates when the random state of the split changes, we are unable to obtain a fixed accuracy for the model. *Cross-validation* is a technique for analyzing and testing the performance of a machine learning model. Cross-validating a model may be accomplished using a variety of methods. Nonetheless, they all use the same algorithm [80].

k-Fold Cross-Validation, for example, includes the k parameter, which specifies the number of groups into which a given data sample should be divided. The basic idea is that the dataset is split into $k-1$ groups with approximately equal data distribution, and for each unique fold the $k-1$ groups form a training set. The data of the group that was not used in the training set form the test set. This process is repeated k times.

This methods helps us use our data properly, and gives us valuable insight about our algorithm's performance.

Chapter 3

Methodology

3.1 Related Work

The literature on shot classification is mostly centered around machine learning methods in sports events (e.g., [81]) that classify the corresponding shots into one of the following basic categories: *long*, *medium*, *close* and *out-field* respectively. Most of the movie shot analysis methods adopt this type of categorization in order to form the shot scale classification problem, where a shot is classified based on the apparent distance of the camera lens from the main subject of a scene. Most of the works (e.g., [82], [83]) examine the three-class problem of long, medium and close-up shots. Recent methods have employed Deep Learning architectures [84] and others enrich their overall architecture with the use of semantic segmentation [85]. However, in these methods, the shot analysis is on frame level rather than on video level, and thus the Deep Learning architecture does not include the temporal dimension.

One of the first studies to introduce a cinematographic shot taxonomy based on camera movements is [86], which ends up with the following classes: stationary, contextual-tracking, focus-tracking, focus-in, focus-out, intermittent/planing establishment and chaotic. This taxonomy is followed by [87] where a new dataset of 5226 shots is created. A method for videography-based video analysis is introduced in [88], where camera operation classification is used as a module on the proposed architecture in order to classify the shot among four categories, namely static, pan, tilt and zoom.

The so far mentioned works on camera movement classification and their predecessors (e.g., [89]) evaluate their respective approaches on their own private collections, which are not made available. The first publicly available dataset is introduced in [90] with shots cate-

gorized as aerial, bird eye, crane, dolly, establishing, pan, tilt, and zoom. However this collection is rather small, consisting of just 263 shots, where most of them are not from movies. The most relevant work is that of [35] MovieShot, a publicly available dataset that contains classes for both scale and camera movement. Still, the proposed camera movement types are rather generic including only static, motion, push shot (zoom in) and pull shot (zoom out).

Most of these studies laid the groundwork for the detection of objects or classification of actions in films, but they overlooked one of the most important procedures, the camera movement. Also, some of them are not movie-oriented since they include other types of videos. The camera motion is critical, as it has a significant impact on whether he likes the movie or not, since it helps arouse his emotions. This is the purpose of the work we present.

3.2 Dataset compilation

A data collection procedure, along with a labeling and an agreement process, were followed for the training and evaluation of the classifiers. Because there are no publicly available labeled movie datasets on the internet, this data collection method was performed in the context of the research. The sections that follow outline all of the steps that were performed to produce a fully annotated movie shot dataset.

3.2.1 Video shot generation

Video *shots* are basic structural elements in film-making and video production that contain a series of frames and run for an uninterrupted period of time [91]. Types of shots can be characterized (among others) with regards to the respective camera movement. Camera movement in film-making is a technique that causes a change in frame or perspective through the movement of the camera. The types of camera movement are crucial in the direction film process, since through these, directors can cause separate feelings to the audience: for example, fast camera movements can cause the viewer anxiety or irritation.

In this thesis, our goal is to classify the types of shots based on the cinematic aesthetics of camera movements. Towards this end, we have created a manually annotated dataset. To our best knowledge, there is no corresponding dataset that represents a wide range of camera movement styles that is able to cover the vast majority of shot types in any movie. The dataset is created using video shots from 48 films in which a basic shot detection algorithm

was applied to generate successive shots. The generated detected video shots were randomly sampled and then led to a multi-annotator pipeline, leaning to the final annotated dataset.

In order to detect shots from movies, we adopted a very basic and fast shot-detection algorithm that is based on three thresholding criteria. In particular, as a first step, the following features are computed on a frame basis:

- Average value of magnitudes of optical flow vectors (mag_mu). Optical flow vectors are modeling local movements of video blocks in successive frames [92]. In other words, a flow vector indicates how a particular block from a frame will 'move' into the next frame. The rationale behind using this feature is that, if the value of magnitudes of flow vectors changed abruptly from the current frame to the next, then it would probably occur a shot change.

- Proportion of pixels with high absolute differences between two successive frames ($gray\_diff$): in particular, we count pixels whose differences is over 50 between two successive frames.

- Average absolute diffs in the histograms of the gray values between two successive video frames (f_diff).

In order to select the aforementioned parameters, we created a small dataset of manually annotated shots (in terms of segment endpoints) and selected the parameters values $mag\_mu = 0.08$, $gray\_diff = 0.65$ and $f\_diff = 0.02$ which resulted to an almost 80 macro F1 performance in the binary task of endpoint shot detection, with a tolerance of 1 second (i.e., the allowed time distance from the ground truth shot endpoints). The aforementioned method has been implemented in the GitHub repository that also implements the basic feature extraction adopted in this thesis, called *multimodal_movie_analysis* [1].

As soon as this parameter setting of the shot detection algorithm was completed, we executed the algorithm on 48 movies from various genres and eras. This process resulted in around 80,000 detected shots. We then discarded shots shorter than 2 seconds and randomly selected 4000 shots as the final "annotation pool". In the next Sections we describe the annotation process of these 4000 shots and the way we handled the inter-annotator agreement to form the final ground-truth.

[1] `https://github.com/tyiannak/multimodal_movie_analysis`

3.2.2 Video shot taxonomy

In order to define the set of classes in which the shots have been labelled, we collaborated with a professional director and at the same time we focused in having a minimum set of classes that covers as many shots as possible from all movies. Our basic criterion for defining the classes was the *type of camera movement*. Based on this, the following classes have been defined:

1. **Static**, The camera is locked on a tripod or pedestal and remains still. Among other types of scenes, it is commonly used in dialogues. A static camera does not necessarily indicate a static scene. Actors and even the background can move while the camera remains still. (Link-for-static-video)

2. **Vertical movement**, of the camera lens while the camera remains locked on a tripod. It is equivalent to someone tilting his head up/down. (Link-for-vertical-movement-video)

3. **Tilt**, Moving the entire camera up or down without moving its lens. Tilting up is like one is moving up his entire body from a sitting position. (Link-for-Tilt-video)

4. **Panoramic**, Lateral movement of the camera lens while the camera remains locked down on its tripod or pedestal. It is like someone is moving his head from one side to another. (Link-for-Panoramic-video)

5. **Panoramic Lateral**, The camera follows the action moving parallel to characters. Specifically, the camera captures the lateral movement of the subject, for example the camera moves parallel to a person walking down the street to keep them in the frame. (Link-for-Panoramic-lateral-video)

6. **Travelling in**, in this type of shot, the camera moves forward, pushes in a character or follows a character. (Link-for-Travelling-in-video)

7. **Travelling out**, in this type of shot, the camera pulls out, moving away from the subject and revealing the surroundings. (Link-for-Travelling-out-video)

8. **Zoom in**, In this type of shot, the camera lens are adjusted so that the image gradually appears larger and closer. (Link-for-Zoom-in-video)

9. **Aerial**, the camera is flown above the action, using a helicopter, drone or a plane. (Link-for-Aerial-video)

10. **Handheld**, the camera is moving throughout the filming set, while the camera operator is physically holding it. These camera shots are shaky and create a hectic feeling. (Link-for-Handheld-video)

Initially we used five more classes that have been found in the literature, but these have been rarely found in the annotation process (in particular they gathered less than 30 annotations after the aggregation procedure described in the next Section). These classes are: (1) **Car Front Windshield**, the camera is mounted on the front windshield, (2) **Car Side Mirror**, the camera is mounted on the side mirror and the viewer can see the driver (and co-driver) from the side (3) **Zoom out**, the entire image appears much smaller and further away (4) **Vertigo**, a combination of travelling and zoom and (5) **Panoramic 360**, a semicircular movement of the camera. These types of shots appeared in less than 30 annotations (i.e., in less than 0.75% of the total data), and therefore their recognition would be of low significance in a real-world scenario of movie analysis. Furthermore, any supervised procedure would fail to model these classes with such few data points.

3.2.3 Annotation process and agreement

As soon as the 4000 shots had been selected from the 48 films, we began the annotation procedure. Towards this end, we used a simple Python-based video annotation web tool [2]. Figure 3.1 presents a screenshot of the tool: the user can simply view the video and assign it a video shot label. Note that the users had also the ability to annotate a video as 'N/A': this label was used for cases that corresponded to either corrupted videos, or to videos with a non clear and fuzzy type of camera movement, or even to videos with more than one types of camera movements. Video shots that have been annotated as 'N/A' were discarded from the final dataset.

This process has been carried out by 17 human annotators that annotated an arbitrary number of randomly selected shots. In total, 7500 individual annotations have been made. Each sample must have a single label at the end. For this to happen, we proceeded to applying a simple aggregation rule: *for each video shot, two minimum annotations (by two different*

[2] `https://github.com/theopsall/video_annotator`

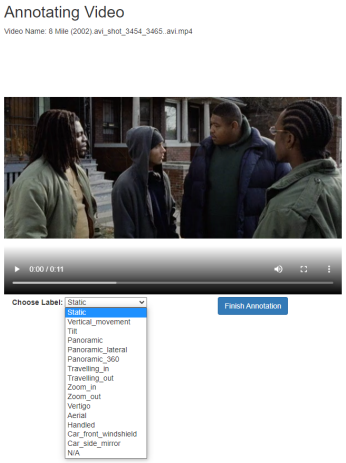


Figure 3.1: Annotation tool

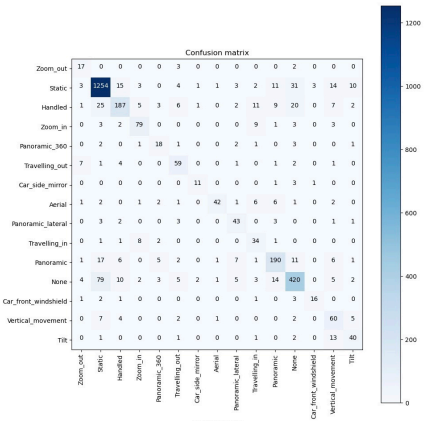


Figure 3.2: Inter-annotator agreement

Figure 3.3: Web tool and annotation agreement

annotators) were required, along with a minimum annotator agreement of 60%. This means that if a shot was annotated by two persons, this shot will be used in the final ground truth only if both annotators agreed on the label. Similarly, if three persons annotated a video shot, this would be used in the final ground truth if at least two of the annotators agreed on the label (higher than 60% agreement). Figure 3.2 visualizes the agreement between the individual annotators for all adopted classes. The overall inter-annotator agreement was above 80%.

After this process, 1877 videos from all 15 classes "survived" with a confident aggregated label. However, for 5 of these classes, the number of samples was below 30 and therefore have not been used in the final dataset. Given that, the final size of the dataset was 1803. Table 3.1 shows the final aggregated counts per final class (for the 10 adopted classes).

3.3 Experimental set-up

3.3.1 Feature extraction

Feature extraction is extensively used in pattern recognition, machine learning and image processing. This is a method of a low-dimensional representation, where limited resources are used to describe a large dataset. That is, the main goal is to extract the most significant insight from the original dataset and present it in a low-dimensional space [93]. Image classification methods are usually based on features extracted from well-established Deep Learning computer vision architectures, such as the VGG [94] and ResNet [95] net-

Table 3.1: Number of samples per class of the final dataset

Class	Samples
Static	985
Handheld	273
Panoramic	207
Travelling in	55
Vertical movement	52
Aerial	51
Zoom in	51
Travelling out	46
Panoramic lateral	46
Tilt	37

works. However, in the task of Video Shot Classification we are more interested in capturing the image flow rather than the actual visual information.

That is the reason why the hand-crafted video features presented in [96], which meaningfully describe the flow of the visual information, where chosen. Particularly, every 0.2 sec, these 88 features of visual information are being extracted from the corresponding frame:

- *Color-related* features (45 features):
  - 8-bin histogram of the red values
  - 8-bin histogram of the green values
  - 8-bin histogram of the blue values
  - 8-bin histogram of the grayscale values
  - 5-bin histogram of the max-by-mean-ratio for each RGB triplet
  - 8-bin histogram of the saturation values
- *Average absolute difference* between two successive frames in grey scale (1 feature)
- *Facial features* (2 features): The Viola-Jones [97] OpenCV implementation is used to detect frontal faces and the following features are extracted per frame:
  - number of faces detected

  - average ratio of the faces' bounding boxes areas divided by the total area of the frame

- *Optical-flow* related features (3 features): The optical flow is estimated using the Lucas-Kanade method [92] and the following 3 features are extracted:
  - average magnitude of the flow vectors
  - standard deviation of the angles of the flow vectors
  - a hand-crafted feature that measures the possibility that there is a camera tilt movement – this is achieved by measuring a ratio of the magnitude of the flow vectors by the deviation of the angles of the flow vectors.

- *Current shot duration* (1 feature): a basic shot detection method is implemented in this library. The length of the shot (in seconds) in which each frame belongs to, is used as a feature.

- *Object-related* features (36 features): We use the Single Shot Multibox Detector [98] method for detecting 12 categories of objects. For each frame, as soon as the object(s) of each category are detected, three statistics are extracted: number of objects detected, average detection confidence and average ratio of the objects' area to the area of the frame. So in total, 3x12=36 object-related features are extracted. The 12 object categories we detect are the following: *person, vehicle, outdoor, animal, accessory, sports, kitchen, food, furniture, electronic, appliance and indoor* [96].

For the feature extraction procedure described above, the *multimodal_movie_analysis* library[3] was used. Therefore, features representing visual characteristics are obtained.

The aforementioned features allow for a variety of representation levels, including low *(simple color aggregates)*, mid *(optical flows)* and high *(presence of objects and faces)* ones. Our goal is to describe a broad range of flow information that may be associated with camera movements, which is why we chose such a diverse set of attributes.

3.3.2 Feature selection for static and sequential methods

In order to approach the problem from two different perspectives, which will be discussed in Section 3.3.3, we create two different types of features:

[3] `https://github.com/tyiannak/multimodal_movie_analysis`

- **Static features**

  The following video-level statistics are calculated for each frame sequence:

  - six (6) video-level statistics of the 52 non-object features. In particular, mean, standard deviation (std), median by std ratio, top-10 percentile, mean of the delta features and std of the delta features.
  - for the object detection, the frame-level predictions are post processed under local time windows with two different ways: (i) the object frame-level confidences are smoothed across time windows in terms of increasing the accuracy of the predictions and (ii) every object that is not present to at least a minimum number *(threshold)* of subsequent frames, is excluded from the final feature vector. However, this smoothing procedure is the only post-processing performed on the object-related features: no other statistics are extracted for the whole video, other than the object features' simple averages.

  This process therefore results to $(52 * 6) + 36 = 348$ feature statistics that describe the whole video in a static feature vector.

- **Sequential features**

  A post processing procedure is applied on the feature sequence in order to smooth the representation across successive frames. Median-filtering with kernel size of 4 (i.e., 0.8 seconds) was chosen. The processed sequence can be used in a temporal modeling.

3.3.3 Baseline classification

In this section we propose two distinct classification methods that can give an intuition about the separability of the introduced classes. These methods will serve in order to report baseline classification metrics and understand the latent correlations across classes.

Static method

The first method is based on the static video representation described in Section 3.3.1. It adopts an SVM algorithm with the appropriate data normalization and parameter tuning. This approach aims on finding significant correlations between statistical frame representations

and the target shot styles. Such methods can learn to separate continuous camera movements from static shots, but may under-perform in cases of abrupt shot style changes.

Sequential method

This method is aimed to predict the shot style from the temporal feature sequence. In such a case, it would be easier to model different kinds of camera movements since the approach is expected to capture the temporal frame dependencies.

An LSTM architecture was chosen, since it is a well established Recurrent Neural Network that can capture long-term temporal information [99]. Batch-normalization [71] and linear layers where used along with ReLU activation functions in order to perform the classification task. Temporal standard scaling was applied to the input sequence, that is each instance of a feature along the sequence were standarized using the same statistical values *(mean value and standard deviation)*.

Chapter 4

Experiments

In this section, experimental results are presented for both static and sequential methods.

4.1 Classification tasks

By combining different shot categories, four classification tasks, one binary and three multi-label, are defined.

4.1.1 Binary classification task

The binary task includes the *static* and *non-static* classes. The former consists of shots that have been annotated as **static**, while the latter contains all the classes from the original dataset that are associated with any type of camera movement. That is the corresponding sub-classes are **Panoramic Lateral, Vertical Movement, Handheld, Zoom-in, Travelling-in, Panoramic, Aerial, Travelling-out, Tilt**. The *static* class, along with the classes that represent the *non-static* class, are those that have been presented in Section 3.2.2.

4.1.2 Multi-Label classification tasks

As it will be described in the results section below, predicting the type of camera movement is not a trivial task. At the same time, as explained in the introduction (*in Chapter 1*), it is a task that can be mostly used as an automated attribute movie extractor, that is used in the context of a more complex system *(such as movie recommendation)*. Therefore, it is not a task that can form a standalone application or product. Given that, it makes sense to analyze

the movie in various levels of detail with regards to its camera movement styles. In this thesis, apart from the initial 10-class classification task, we have defined two more classification tasks, by merging classes of similar camera movement *(e.g., all vertical movements)*.

So in total, we define three multi-label classification tasks in which the static class is the original annotated class, while the others are merges from other class combinations.

- **3-class** The corresponding classes are *Zoom*, *Static* and *Vertical & Horizontal Movements*. The **Zoom** class consists of the *Zoom-in, Travelling-in and Travelling-out* sub-classes, which all contain shots in which the perimeter image changes at very fast intervals, while the centre image remains static or changes at a slower rate. The **Vertical & Horizontal Movements** class consists of the *Vertical Movement, Tilt, Panoramic and Panoramic Lateral* sub-classes from the original dataset, where the position of the camera is moving either in a vertical or in a horizontal way.

- **4-class** In this task, the **Static** and **Zoom** classes of the 3-class problem were kept, while the *Vertical & Horizontal Movements* class was separated into 2 sub-classes; **Tilt**, which includes all vertical movements and consists of the *Vertical Movement* and *Tilt* original classes, and **Panoramic** that contains shots with lateral movements and consists of the *Panoramic* and *Panoramic Lateral* original classes.

- **10-class** This task includes all provided class from the original dataset, namely: **Static**, **Panoramic**, **Zoom-in**, **Travelling-out**, **Vertical Movement**, **Aerial**, **Travelling-in**, **Tilt**, **Handheld** and **Panoramic Lateral**.

4.2 Experimental procedure

With regard to the *static* method mentioned in section 3.3.3, an SVM classifier with *rbf* kernel was initialized. After the data has been scaled using a zero-mean normalization, parameter tuning was performed over the *'C'* parameter using *GridSearchCV()*.

Concerning the *sequential* method mentioned in section 3.3.3, we initially performed a sequential zero-mean normalization. That is, by treating each frame as a different instance, the mean and standard deviation of all features across all frames of the training set of the dataset was calculated and then used to normalize each frame of the dataset, resulting in normalized

feature sequences. The most crucial part of the task was choosing the hyperparameters of the model, so that the best macro-averaged $f1\_score$ is achieved. The *'LSTM'* architecture was tuned for each of the four tasks in order to achieve the optimal dropout, weight decay, learning rate and LSTM's hidden dimension. For the binary task, the binary cross entropy loss was used, while for the rest, cross entropy was adopted. Adam was chosen to be the optimizer with the integration of a reduce-on-plateau learning rate scheduler, while the best model was chosen using early stopping.

In order to evaluate the classifiers a cross validation procedure of totally 10 iterations was performed. For each iteration, the shots of the dataset were randomly split in a stratified manner *(i.e., keep the initial class distribution)* into 20% test set for testing, 9.6% validation set for parameter tuning and 70.4% train set for training.

4.3 Results

The use of the macro-averaged F1-score *(or macro-F1 score)* to assess classification performance in both binary and multi-label categorization is common. Each class will be given the same weight when using the $f1_{macro}$, since all classes contribute equally, regardless of how frequently they appear in the dataset. In this thesis, we to obtain valuable insight about the performance of our models using $f1_{macro}$. The outcomes of the experiments will be examined, both per classification task and at an overall level, so they can be discussed as binary and multi-label classification tasks separately as well.

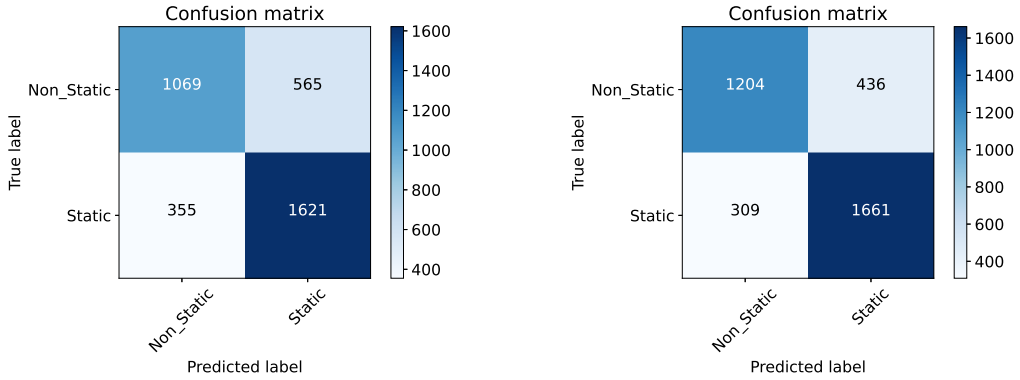
4.3.1 Binary task

The resulted evaluation metrics for the binary task are listed in Table 4.1, while the corresponding, aggregated from all iterations, confusion matrices are presented in Figure 4.1.

Classification Metrics	Static	Sequential
$f1_{macro}$	73.9%	**79%**
accuracy	74.5%	79.4%

Table 4.1: Classification metrics for the binary task for both Static and Sequential methods

As can be clearly seen from the results above, the sequential method completely outperforms the static one in terms of the binary task, by scoring a **7%** relatively higher score. That is an expected result since the basic distinction of the static and non-static classes is the camera movement, which is easier to capture by a sequential method.



(a) Static Method (binary confusion matrix)

(b) Sequential Method (binary confusion matrix)

Figure 4.1: Confusion matrices based on aggregated predictions for the binary task

4.3.2 3-class task

With regard to the 3-class classification task, the resulted evaluation metrics are listed in Table 4.2, while the confusion matrices for this task are presented in Figure 4.2.

Classification Metrics	Static	Sequential
$f1_{macro}$	50.6%	**53%**
accuracy	72.6%	66.6%

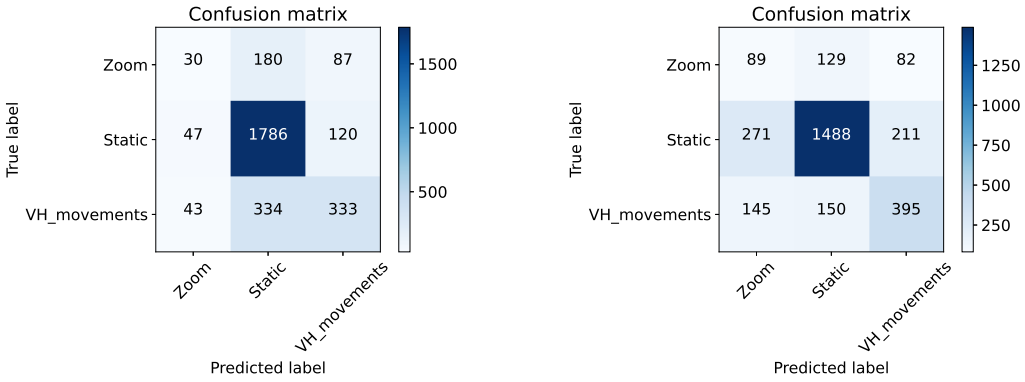
Table 4.2: Classification metrics for the 3-class classification task for both Static and Sequential methods

As we can see from the metrics above, the sequential method achieves a higher by **4.7%** performance compared to the static method *(based on the macro-F1 score)*. The accuracy of the static method seems to be greater than the accuracy of the sequential method. However, as it is clearly depicted in the Figure 4.2 below, the Sequential method provides a reliable

Classification Metrics	Class	Static	Sequential
Precision	Zoom	25%	18%
	Static	78%	84%
	Vertical and Horizontal Movements	62%	57%
Recall	Zoom	10%	30%
	Static	91%	76%
	Vertical and Horizontal Movements	53%	57%

Table 4.3: Precision and Recall per class for both Static and Sequential methods for the 3-class task

prediction of the non-neutral classes *(Zoom, VH_movements*[1]*)* in comparison with the static method. Macro-averaged f1-score of the non-neutral classes (ie. excluding static class), which is calculated on the aggregated predictions across all 10 cross-validated iterations, is 35.7% for the Static and 39.8% for the Sequential method. Precision and Recall per class are presented in Table 4.3.



(a) Static Method (3-class confusion matrix)

(b) Sequential Method (3-class confusion matrix)

Figure 4.2: Confusion matrices based on aggregated predictions for the 3-class classification task

[1]VH_movements class represents “Vertical And Horizontal Movements” class

4.3.3 4-class task

In terms of the 4-class classification task, the evaluation metrics obtained are shown in Table 4.4, while the confusion matrices for this task are presented in Figure 4.3.

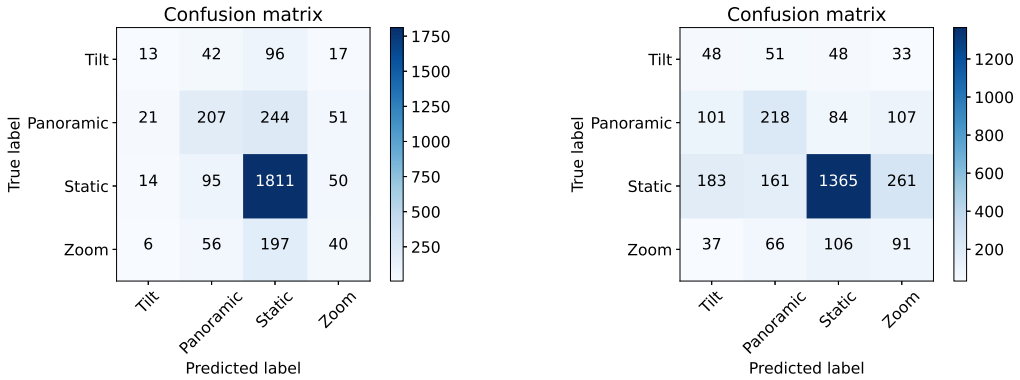
Classification Metrics	Static	Sequential
$f1_{macro}$	39.5%	**40%**
accuracy	69.9%	58.2%

Table 4.4: Classification metrics for the 4-class classification task for both Static and Sequential methods

Classification Metrics	Class	Static	Sequential
Precision	Tilt	24%	13%
	Panoramic	52%	44%
	Static	77%	85%
	Zoom	25%	18%
Recall	Tilt	8%	27%
	Panoramic	40%	43%
	Static	92%	69%
	Zoom	13%	30%

Table 4.5: Precision and Recall per class for both Static and Sequential methods for the 4-class task

From the Table above, we can see that the sequential method performs slightly better by **1.3%** than the static one *(based on the macro-F1 score)*. The accuracy of the static method is again greater than the accuracy of the sequential method. Nevertheless, as illustrated in the confusion matrix of Figure 4.3, we observe that the Sequential method predicts the non-neutral classes *(Tilt, Panoramic Zoom)* significantly better than the static method. Macro-averaged f1-score of the non-neutral classes, for all 10 cross-validated iterations, is 24.8% for the Static and 27.8% for the Sequential method. Precision and Recall per class are presented in Table 4.5.



(a) Static Method (4-class confusion matrix)

(b) Sequential Method (4-class confusion matrix)

Figure 4.3: Confusion matrices based on aggregated predictions for the 4-class classification task

4.3.4 10-class task

The evaluation metrics for the 10-class classification task are reported in Table 4.6, while the confusion matrices for this task are illustrated in Figure 4.4.

Classification Metrics	Static	Sequential
$f1_{macro}$	**15.7%**	15.4%
accuracy	58.6%	32.7%

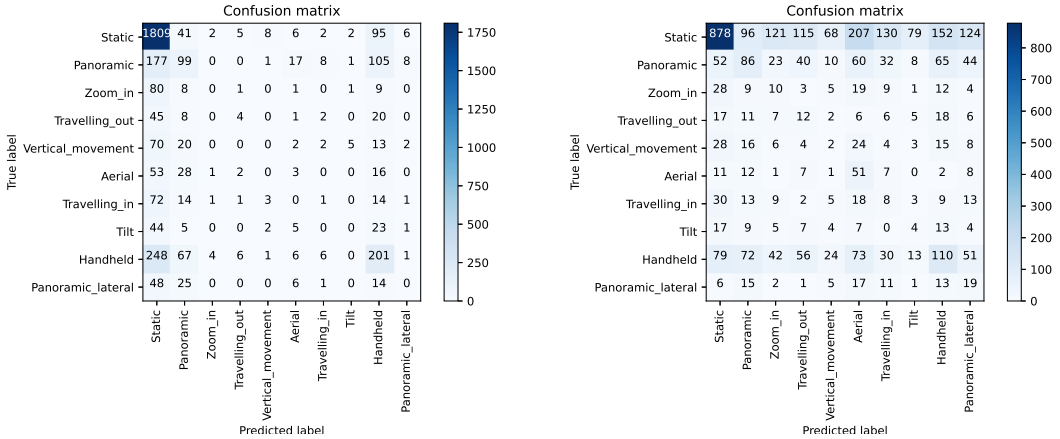
Table 4.6: Classification metrics for the 10-class classification task for both Static and Sequential methods

For this 10-class task, the static method performs **1.9%** better than the sequential method, as demonstrated in the table above. Despite this, comparing the confusion matrices in Figure 4.4, it is apparent that while the Sequential method does not predict as many instances for the dominant class as the Static method does, it does identify instances for all the non-neutral classes, where the static method fails to predict even one instance correctly. Precision and

Classification Metrics	Class	Static	Sequential
Precision	Static	68%	77%
	Panoramic	31%	25%
	Zoom in	0%	4%
	Travelling out	21%	5%
	Vertical movements	0%	2%
	Aerial	6%	11%
	Travelling in	5%	3%
	Tilt	0%	3%
	Handheld	39%	27%
	Panoramic lateral	0%	7%
Recall	Static	92%	45%
	Panoramic	24%	20%
	Zoom in	0%	10%
	Travelling out	5%	13%
	Vertical movements	0%	2%
	Aerial	3%	51%
	Travelling in	1%	7%
	Tilt	0%	6%
	Handheld	37%	20%
	Panoramic lateral	0%	21%

Table 4.7: Precision and Recall per class for both Static and Sequential methods for the 10-class task

Recall per class are presented in Table 4.7.



(a) Static Method (10-class confusion matrix)

(b) Sequential Method (10-class confusion matrix)

Figure 4.4: Confusion matrices based on aggregated predictions for the 10-class classification task

4.4 Commentary

In terms of the binary task, the sequential method completely beats the static method, as mentioned above. This is to be expected, given that the primary difference between the static and non-static classes is camera movement of the lens, which is easier to capture using a sequential method.

It is observed, though, that the more the classes are in the tasks, the less the difference between the two methods is. That is mostly due to the fact that some of the 10 classes are underrepresented and thus a deep learning architecture is harder to learn from that amount of data compared to a traditional machine learning algorithm. As far as it concerns the multi-label task, where the neutral class *(static class)* is dominant in terms of number of instances, the macro-averaged f1 metric cannot properly represent the performance of the methods. By examining the confusion matrices, it can be clearly seen that the sequential method achieves to adequately predict the non-neutral classes. More specifically, as illustrated in Table 4.8, which includes the macro-averaged f1 metrics based only on the non-neutral classes *(ie. excluding the static class)*, the sequential method is 11.5% and 12.1% relatively better for the 3 and

Classification task	Static	Sequential
3-class	35.7%	**39.8%**
4-class	24.8%	**27.8%**

Table 4.8: Macro-averaged f1 of the non-neutral classes (ie. excluding static class) calculated on the aggregated predictions across all 10 cross-validated iterations

4-class problem respectively. Thus it can be inferred that including temporal information in the classifier leads to better distinction across camera movements.

The SVC() algorithm outperformed other classic Machine Learning classifiers in the static approach, which is why we maintained its results, to compare them to the LSTM for the sequential technique.

It needs to be noted that the selection of the features used in our method was not random. More specifically, our pipeline was evaluated by training and testing the LSTM on feature sequences extracted from the first linear layer of the pre-trained VGG16 network. The results showed that when the LSTM is trained on our features (produced by the *multimodal_movie_analysis* library), it performs 21.5%, 23.6%, 24% and 7.7% better for the 2, 3, 4 and 10-class tasks respectively, compared to the model trained on the $VGG16$ features. This outcome is reasonable since while the VGG features contain better image-related information, our features achieve to better model the camera movement, an attribute that is crucial for shot type classification.

Chapter 5

Conclusion & Future Challenges

In this Thesis we explore a very interesting task, that of movie analysis, based on the visual part of a movie shot. As a result of completing the literature review, for this purpose, a new publicly available [1] dataset for film shot classification with annotations in the camera movement styles was presented. This data collection fills the gap in the literature of cinematographic style analysis, since the existing datasets are either private or are not concerned about camera movements in an extensive manner. Additionally, we experimented with two types of model architectures, where two different methods were used as baseline evaluation; one static and a sequential one, showing that camera movement classification is better performed when modeling sequential information. More particular, we observed that the LSTM algorithm outperforms the SVM algorithm in the binary task, whereas in multi-label tasks, the difference between the two techniques appears to be less significant in terms of metrics. Yet, the LSTM succeeds in recognizing non-common classes to a greater extent. This arises because some of the classes are unbalanced, making it more difficult for a deep learning architecture to be trained with such a small volume of data, compared to a typical machine learning method.

It would be interesting to train the suggested approach on a robust training dataset featuring a large selection of films from many genres, annotated by a larger number of individuals. Models trained on this dataset could be used in a future work in order to create artistic profiles of either directors or movies that can describe in an informative way the corresponding movie. Moving a step forward, this research can provide a new way of representing the movies in recommender systems, so that the extracted recommendations are content-driven

[1] https://github.com/magcil/movie_shot_classification_dataset

and, in particular, driven by the actual underlying visual aesthetics of the movies. One of the most difficult aspects of multimodal data is combining information from various modalities in such a way that complementary information is combined.

Several of the methodologies followed in the studies [100] [33] could be exploited to expand the scope of the project, for multimodal representation. We could also study [101] which presents a model trained from raw pixels of videos and [28] for automatic genre classification, where visual features are extracted from trailers.

Bibliography

[1] Artificial intelligence, machine learning, and deep learning: Same context, different concepts. https://master-iesc-angers.com/artificial-intelligence-machine-learning-and-deep-learning-same-context-different-concepts/. Last accessed 2 Feb 2022.

[2] Supervised, unsupervised, & reinforcement learning. https://docs.paperspace.com/machine-learning/wiki/supervised-unsupervised-and-reinforcement-learning. Last accessed 28 Jan 2022.

[3] Types of machine learning out there. https://idapgroup.com/blog/types-of-machine-learning-out-there/. Last accessed 2 Feb 2022.

[4] The train, validation, and test sets: How to split your machine learning data. https://www.v7labs.com/blog/train-validation-test-set. Last accessed 30 Jan 2022.

[5] Decoding the confusion matrix. https://keytodatascience.com/confusion-matrix/. Last accessed 31 Jan 2022.

[6] Confusion matrix : Let's clear this confusion. https://medium.com/@aatish_kayyath/confusion-matrix-lets-clear-this-confusion-4b0bc5a5983c. Last accessed 31 Jan 2022.

[7] Understanding confusion matrix. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62. Last accessed 31 Jan 2022.

[8] The 10 best machine learning algorithms. https://www.dataquest.io/blog/top-10-machine-learning-algorithms-for-beginners/. Last accessed 10 Jan 2022.

[9] Decision tree in machine learning. https://www.educba.com/decision-tree-in-machine-learning/. Last accessed 10 Jan 2022.

[10] Support vector machines(svm) — an overview. https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989. Last accessed 10 Jan 2022.

[11] Facundo Bre, Juan Gimenez, and Víctor Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158, 11 2017.

[12] Activation functions in neural networks. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. Last accessed 15 Jan 2022.

[13] Junxi Feng, Xiaohai He, Qizhi Teng, Chao Ren, Honggang Chen, and Yang Li. Reconstruction of porous media from extremely limited information using conditional generative adversarial networks. *Physical Review E*, 100, 09 2019.

[14] Implementing drop out regularization in neural networks. https://www.tech-quantum.com/implementing-drop-out-regularization-in-neural-networks/. Last accessed 10 Jan 2022.

[15] Implementation of convolutional neural network. https://capablemachine.com/2020/05/07/convolutional-neural-network/. Last accessed 15 Jan 2022.

[16] Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Last accessed 18 Jan 2022.

[17] Long short-term memory (lstm) rnn in tensorflow. https://www.javatpoint.com/long-short-term-memory-rnn-in-tensorflow. Last accessed 18 Jan 2022.

[18] Wikipedia contributors. Shot (filmmaking) — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Shot_(filmmaking)&oldid=1043535530`, 2021. [Online; accessed 12-February-2022].

[19] Min Xu, Jinqiao Wang, Muhammad A Hasan, Xiangjian He, Changsheng Xu, Hanqing Lu, and Jesse S Jin. Using context saliency for movie shot classification. In *2011 18th IEEE International Conference on Image Processing*, pages 3653–3656. IEEE, 2011.

[20] George Lekakos and Petros Caravelas. A hybrid approach for movie recommendation. *Multimedia tools and applications*, 36(1):55–70, 2008.

[21] Sang-Min Choi, Sang-Ki Ko, and Yo-Sub Han. A movie recommendation algorithm based on genre correlations. *Expert Systems with Applications*, 39(9):8079–8085, 2012.

[22] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 193–202, 2014.

[23] V Subramaniyaswamy, R Logesh, M Chandrashekhar, Anirudh Challa, and Varadarajan Vijayakumar. A personalised movie recommendation system based on collaborative filtering. *International Journal of High Performance Computing and Networking*, 10(1-2):54–63, 2017.

[24] Jitao Sang and Changsheng Xu. Character-based movie summarization. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 855–858, 2010.

[25] Chia-Ming Tsai, Li-Wei Kang, Chia-Wen Lin, and Weisi Lin. Scene-based movie summarization via role-community networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(11):1927–1940, 2013.

[26] Ijaz Ul Haq, Amin Ullah, Khan Muhammad, Mi Young Lee, and Sung Wook Baik. Personalized movie summarization using deep cnn-assisted facial expression recognition. *Complexity*, 2019, 2019.

[27] Howard Zhou, Tucker Hermans, Asmita V Karandikar, and James M Rehg. Movie genre classification via scene categorization. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 747–750, 2010.

[28] Gabriel S Simões, Jônatas Wehrmann, Rodrigo C Barros, and Duncan D Ruiz. Movie genre classification with convolutional neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 259–266. IEEE, 2016.

[29] Ali Mert Ertugrul and Pinar Karagoz. Movie genre classification from plot summaries using bidirectional lstm. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 248–251. IEEE, 2018.

[30] Ijaz Ul Haq, Khan Muhammad, Tanveer Hussain, Soonil Kwon, Maleerat Sodanil, Sung Wook Baik, and Mi Young Lee. Movie scene segmentation using object detection and set theory. *International Journal of Distributed Sensor Networks*, 15(6):1550147719845277, 2019.

[31] Zeeshan Rasheed and Mubarak Shah. Scene detection in hollywood movies and tv shows. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–343. IEEE, 2003.

[32] Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. Shot and scene detection via hierarchical clustering for re-using broadcast video. In *International Conference on Computer Analysis of Images and Patterns*, pages 801–811. Springer, 2015.

[33] Konstantinos Bougiatiotis and Theodoros Giannakopoulos. Enhanced movie content similarity based on textual, auditory and visual information. *Expert Systems with Applications*, 96:86–102, 2018.

[34] Theodoros Psallidas, Panagiotis Koromilas, Theodoros Giannakopoulos, and Evaggelos Spyrou. Multimodal summarization of user-generated videos. *Applied Sciences*, 11(11), 2021.

[35] Anyi Rao, Jiaze Wang, Linning Xu, Xuekun Jiang, Qingqiu Huang, Bolei Zhou, and Dahua Lin. A unified framework for shot type classification based on subject centric lens. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 17–34. Springer, 2020.

[36] FATHIMA ANJILA PK. What is artificial intelligence? *"Success is no accident. It is hard work, perseverance, learning, studying, sacrifice and most of all, love of what you are doing or learning to do".*, page 65, 1984. Last accessed 9 Dec 2021.

[37] Shuai Zhao, Frede Blaabjerg, and Huai Wang. An overview of artificial intelligence applications for power electronics. *IEEE Transactions on Power Electronics*, 36(4):4633–4658, 2020. Last accessed 12 Jan 2022.

[38] Asharul Islam Khan and Salim Al-Habsi. Machine learning in computer vision. *Procedia Computer Science*, 167:1444–1451, 2020. Last accessed 2 Feb 2022.

[39] IBM. Natural language processing (nlp). `https://www.ibm.com/cloud/learn/natural-language-processing`. Last accessed 2 Feb 2022.

[40] Elizabeth D Liddy. Natural language processing. 2001. Last accessed 2 Feb 2022.

[41] What are the types of machine learning? `https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f`. Last accessed 2 Feb 2022.

[42] Meng Joo Er, Rajasekar Venkatesan, and Ning Wang. An online universal classifier for binary, multi-class and multi-label classification. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 003701–003706, 2016. Last accessed 3 Feb 2022.

[43] Supervised vs. unsupervised learning: What's the difference? `https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning`. Last accessed 3 Feb 2022.

[44] Pierre Yves Glorennec. Reinforcement learning: An overview. In *Proceedings European Symposium on Intelligent Techniques (ESIT-00), Aachen, Germany*, pages 14–15. Citeseer, 2000.

[45] Training data vs. validation data vs. test data for ml algorithms. `https://www.applause.com/blog/training-data-validation-data-vs-test-data`. Last accessed 30 Jan 2022.

[46] Evaluation metrics for machine learning - accuracy, precision, recall, and f1 defined. https://wiki.pathmind.com/accuracy-precision-recall-f1. Last accessed 31 Jan 2022.

[47] Macro f1-score. https://peltarion.com/knowledge-center/documentation/evaluation-view/classification-loss-metrics/macro-f1-score. Last accessed 31 Jan 2022.

[48] Decision tree classification algorithm. https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm. Last accessed 10 Jan 2022.

[49] Commonly used machine learning algorithms. https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/. Last accessed 10 Jan 2022.

[50] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. *New advances in machine learning*, 3:19–48, 2010.

[51] What is deep learning? https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network. Last accessed 20 Jan 2022.

[52] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[53] AD Dongare, RR Kharde, Amit D Kachare, et al. Introduction to artificial neural network. *International Journal of Engineering and Innovative Technology (IJEIT)*, 2(1):189–194, 2012. Last accessed 15 Jan 2022.

[54] Deep learning neural networks explained in plain english. https://www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english/. Last accessed 16 Jan 2022.

[55] Neural networks: What they are & why they matter. https://www.sas.com/el_gr/insights/analytics/neural-networks.html#technical. Last accessed 16 Jan 2022.

[56] Neural networks. https://www.ibm.com/cloud/learn/neural-networks. Last accessed 15 Jan 2022.

[57] Bin Ding, Huimin Qian, and Jun Zhou. Activation functions and their characteristics in deep neural networks. In *2018 Chinese control and decision conference (CCDC)*, pages 1836–1841. IEEE, 2018.

[58] 12 types of neural network activation functions: How to choose? https://www.v7labs.com/blog/neural-networks-activation-functions#activation-function. Last accessed 15 Jan 2022.

[59] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

[60] Fundamentals of deep learning – activation functions and when to use them? https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/. Last accessed 15 Jan 2022.

[61] Understanding activation functions in neural networks. https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0. Last accessed 15 Jan 2022.

[62] Understanding different loss functions for neural networks. https://shivaverma.medium.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718. Last accessed 16 Jan 2022.

[63] An introduction to neural network loss functions. https://programmathically.com/an-introduction-to-neural-network-loss-functions/. Last accessed 16 Jan 2022.

[64] Optimization algorithms in neural networks. https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html. Last accessed 12 Jan 2022.

[65] Various optimization algorithms for training neural network. https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6. Last accessed 12 Jan 2022.

[66] Neural network optimization. https://towardsdatascience.com/neural-network-optimization-7ca72d4db3e0. Last accessed 12 Jan 2022.

[67] Raul Rojas. The backpropagation algorithm. In *Neural networks*, pages 149–182. Springer, 1996.

[68] Dropout in (deep) machine learning. https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5. Last accessed 12 Jan 2022.

[69] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[70] Introduction to batch normalization. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/. Last accessed 17 Jan 2022.

[71] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. Last accessed 4 Sep 2021.

[72] Batch normalization in 3 levels of understanding. https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338. Last accessed 17 Jan 2022.

[73] A basic introduction to convolutional neural network. https://medium.com/@himadrisankarchatterjee/a-basic-introduction-to-convolutional-neural-network-8e39019b27c4. Last accessed 18 Jan 2022.

[74] Convolutional neural networks, explained. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939. Last accessed 18 Jan 2022.

[75] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[76] A guide to rnn: Understanding recurrent neural networks and lstm networks. https://builtin.com/data-science/recurrent-neural-networks-and-lstm. Last accessed 18 Jan 2022.

[77] Understanding rnn and lstm. https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e. Last accessed 18 Jan 2022.

[78] Patrick Schratz, Jannes Muenchow, Eugenia Iturritxa, Jakob Richter, and Alexander Brenning. Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data. *Ecological Modelling*, 406:109–120, 2019.

[79] A practical introduction to grid search, random search, and bayes search. https://towardsdatascience.com/a-practical-introduction-to-grid-search-random-search-and-bayes-search-d5580b1d941d. Last accessed 31 Jan 2022.

[80] Understanding 8 types of cross-validation. https://towardsdatascience.com/understanding-8-types-of-cross-validation-80c935a4976d. Last accessed 19 Jan 2022.

[81] Rabia A Minhas, Ali Javed, Aun Irtaza, Muhammad Tariq Mahmood, and Young Bok Joo. Shot classification of field sports videos using alexnet convolutional neural network. *Applied Sciences*, 9(3):483, 2019.

[82] Luca Canini, Sergio Benini, and Riccardo Leonardi. Classifying cinematographic shot types. *Multimedia tools and applications*, 62(1):51–73, 2013.

[83] Sergio Benini, Michele Svanera, Nicola Adami, Riccardo Leonardi, and András Bálint Kovács. Shot scale distribution in art films. *Multimedia Tools and Applications*, 75(23):16499–16527, 2016.

[84] Mattia Savardi, Alberto Signoroni, Pierangelo Migliorati, and Sergio Benini. Shot scale analysis in movies by convolutional neural networks. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2620–2624. IEEE, 2018.

[85] Hui-Yong Bak and Seung-Bo Park. Comparative study of movie shot classification based on semantic segmentation. *Applied Sciences*, 10(10), 2020.

[86] Hee Lin Wang and Loong-Fah Cheong. Taxonomy of directing semantics for film shot classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(10):1529–1542, 2009.

[87] Muhammad Abul Hasan, Min Xu, Xiangjian He, and Changsheng Xu. Camhid: Camera motion histogram descriptor and its application to cinematographic shot classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(10):1682–1695, 2014.

[88] Kang Li, Sheng Li, Sangmin Oh, and Yun Fu. Videography-based unconstrained video analysis. *IEEE Transactions on Image Processing*, 26(5):2261–2273, 2017.

[89] Sang-Cheol Park, Hyoung-Suk Lee, and Seong-Whan Lee. Qualitative estimation of camera motion parameters from the linear composition of optical flow. *Pattern Recognition*, 37(4):767–779, 2004.

[90] Subhabrata Bhattacharya, Ramin Mehran, Rahul Sukthankar, and Mubarak Shah. Classification of cinematographic shots using lie algebra and its application to complex event recognition. *IEEE Transactions on Multimedia*, 16(3):686–696, 2014.

[91] Leo Braudy. Film: An international history of the medium. *Film Quarterly (ARCHIVE)*, 48(3):59, 1995. Last accessed 12 Nov 2021.

[92] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. Vancouver, British Columbia, 1981. Last accessed 20 Dec 2021.

[93] Gaurav Kumar and Pradeep Kumar Bhatia. A detailed review of feature extraction in image processing systems. In *2014 Fourth international conference on advanced computing & communication technologies*, pages 5–12. IEEE, 2014. Last accessed 3 Nov 2021.

[94] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Last accessed 2 Oct 2021.

[95] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. Last accessed 20 Jan 2022.

[96] Theodoros Psallidas, Panagiotis Koromilas, Theodoros Giannakopoulos, and Evaggelos Spyrou. Multimodal summarization of user-generated videos. *Applied Sciences*, 11(11):5260, 2021. Last accessed 20 Jan 2022.

[97] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001. Last accessed 10 Jan 2022.

[98] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. Last accessed 18 Jan 2022.

[99] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016. Last accessed 4 Sep 2021.

[100] Panagiotis Koromilas and Theodoros Giannakopoulos. Deep multimodal emotion recognition on human speech: A review. *Applied Sciences*, 11(17):7962, 2021.

[101] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2012.

Appendix

Code

The whole code for this study can be found in the github repository `https://github.com/tyiannak/multimodal_movie_analysis`, specifically in the 'main' branch and in the branches: *pipeline, LSTM_binary, multi-class, VGG16_LSTM*. Indicatively, the basic LSTM architecture for the binary task is implemented as follows:

```
class LSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers,
                 output_size, dropout_prob):
        super(LSTMModel, self).__init__()

        self.hidden_size = hidden_size
        self.num_layers = num_layers

        self.lstm = nn.LSTM(input_size, hidden_size, num_layers,
                            batch_first=True)

        self.drop = nn.Dropout(p=dropout_prob)
        self.fc = nn.Linear(hidden_size, output_size)

        self.fnn = nn.Sequential(OrderedDict([
            ('relu1', nn.ReLU()),
            ('bn1', nn.BatchNorm1d(self.hidden_size)),
            ('fc1', nn.Linear(self.hidden_size, output_size))]))
```

```
def forward(self, X, lengths):
    packed_output, _ = self.lstm(X)
    output, _ = unpack(packed_output, batch_first=True)
    last_states = self.last_by_index(output, lengths)

    last_states = self.drop(last_states)
    output = self.fnn(last_states)

    return output

@staticmethod
def last_by_index(outputs, lengths):
    # Index of the last output for each sequence.
    idx = (lengths - 1).view(-1, 1).expand(outputs.size(0),
    outputs.size(2)).unsqueeze(1)

    return outputs.gather(1, idx.type(torch.int64)).squeeze()
```