A DEEP LEARNING FACIAL EXPRESION RECOGNITION BASED SCORING SYSTEM FOR RESTAURANTS

Sahil Kumar Jha
Devansh Rajput

Department of Computer Science & Engineering Galgotias University

Abstract

Automated and unmanned restaurants are becoming more and more common. The lack of personnel prevents a direct assessment of the opinions of the patrons, making it impossible to learn about their experiences with the restaurant idea. This research provides a face expression recognition grading system using convolutional neural network (CNN) models that have already been trained for this purpose. It is made up of an Android mobile application, a web server, and an AI server that has already been taught. Ratings are expected to be given to both the cuisine and the surroundings. The scoring system now offers three expressions: satisfied, neutral, and disappointed. FER applications still have issues despite deep learning's strong feature learning capabilities. To prevent overfitting, deep neural networks first need a lot of training data. Neural network with deep -architecture that showes the most promise in object identification tasks cannot be trained With current facial expression databases, well-known neural networks with deep designs that have done the best job of recognizing objects cannot be learned. Variation in age, sex, race, and other factors may also cause high differences between subjects. expression.

In unconstrained facial expression settings, differences in position, illumination, and occlusions are frequent in addition to subject identification bias. One element of biometric technologies that is quickly growing and finding widespread use is face recognition. It has a wide range of uses, including solutions for industry efficiency and monitoring as well as law enforcement and consumer applications. The recent development of low-cost, strong system and the development enormous

databases have attracted academic attention that is primarily focused on the creation of ever-deeper Neural networks have been developed for every aspect of face recognition problems, from detection and planning to representation and classification in verification and recognition solutions. Real-time, accurate recognition of faces solutions. Real-time, accurate face recognition is still difficult to achieve, despite these advancements, largely because Deep Convolution Neural Networks (DCNN) have a high processing cost and it is necessary to balance accuracy demands with time. Occlusion, lighting, and position invariance are additional key difficulties that affect face recognition and significantly reduce accuracy for both conventional handmade solutions and deep neural networks.

Introduction

Facial expression is a powerful, innate, and ubiquitous means of communicating emotion and intent [1, 2]. Because of its potential utility in social robotics, medical diagnosis, and tiredness monitoring for driversSeveral studies have been done on the topic of automatic facial image analysis. Several methods for facial expression recognition (FER) have been studied to encode expression data from facial expressions in computer vision and machine learning. At the beginning of the 20th century, Ekman and Friesen [3] identified six basic emotions; Their work emerged from cross-cultural research [4] which found that certain basic emotions are experienced in similar ways across cultures.
Facial emotions such as surprise, amusement, disgust, disgust, pleasure, grief, and anger are common. Later, disdain was added to the list of primary feelings [5]. Recent studies in neuroscience and psychology have challenged the idea that everyone experiences the same six fundamental emotions [6]. Facial Action Coding System (FACS) [10] and continuous models with affect parameters [11]. two more emotion description models often believed to capture human emotion.

Goals

After that, a deep FER system's traditional pipeline is explained, along with some background data and ideas for implementations that would be effective at each level. We review the advantages as well as disadvantages For FER, the most cutting-edge deep neural networks and training methods have been designed for both still photos and moving image sequences.. This is the cutting edge of deep FER. Additionally, this section summarizes competitor performance on diversely used references. Then, we broaden our survey to include more pertinent issues and usage examples. The field's remaining challenges, corresponding opportunities, and probable future directions for development of trustworthy deep FER systems are finally covered.

Current system

Numerous thorough studies on automated expression analysis had been published recently [7], [8]. A collection of typical algorithm pipeline of FER has been created using these surveys. Deep learning hasn't gotten much attention, and they emphasise the usage of traditional methods. A concise explanation of FER based on deep learning that omits FER dataset introductions and technical details on deep FER can be found in. So, using both videos and static images In this investigation, we systematically investigate deep learning for FER problems (picture sequences). Our goal is to introduce a newcomer to the methodological underpinnings and skill sets necessary for in-depth FE.

Proposed system

FER applications still have issues despite deep learning's strong feature learning capabilities. To prevent overfitting, deep neural networks first need a lot of training data. Popular deep learning neural networks that have shown the most promise in identification tasks are currently unable to be trained with the current facial expression databases. High inter-subject variances are also a result of individual differences in age, gender, ethnic background, and expressiveness . These variables are nonlinearly correlated with face expressions, which makes deep networks even more necessary to deal with the high variability and to acquire efficient expression-specific representations.

The Project's Scope

The face expression is just one of several angles from which human expressive behaviours in realistic applications must be encoded. Combining many models of a max-level framework, which can include supplementary information, can further boost resilience. Pure expression recognition using only visible facial images can lead to promising outcomes. For instance, participants in the (AVEC) and EmotiW challenges used a range of fusion algorithms of multimodal identification and ranked the audio model as the second-most important component . This is also an intriguing research field because of the great complementarity between facial expression other methods.

Module explanation

Preparing in advance and perfecting was already mentioned, overfitting can happen when deep networks are explicitly trained on a tiny amount of data related to face expressions. To resolve, numerous studies or created their own networks from scratch using extra task-oriented data or enhanced famous, in the opinion of Kahou et al. can help in the
For selecting the (FR) datasets like Useful FER databases include the 2013 and TFD versions, as well as the CASIA WebFace, Wild Celebrity Face, and FaceScrub databases. Like et al., which was trained for FR, outperformed ImageNet, which was designed for object recognition. According to Knyazev et al, pre-training on huge amount FR datasets has been observed to increase the accuracy of emotion recognition, with more FER datasets can improve performance. observation by.

SOFTWARE AND HARDWARE REQUIREMENTS

Soft Ware Requirements

- **Operating System**: Windows
- **Programming language** : python
- **Data Bases** : sql

Hardware Requirements:

- **Processor :** Pentium-III (or) Higher
- **Ram :** 64MB (or) Higher
- **Cache :** 512MB
- **Hard disk :** 10GB

Technology description

Python

Python is an object-oriented, high-level, general-purpose, interactive, and interpreted programming language. Python is an interpreted language with a focus on code readability in its design philosophy. For example, instead of employing curly brackets or keywords, Python employs indented spaces to divide sections of code. Python's syntax helps programmers to express concepts in fewer lines of code than languages such as C++ or Java. It provides building blocks for precise programming on an increasingly large scale. There are Python interpreters available for an array of operating systems. All Python versions, including the standard Python implementation, use the community-based development framework. Python is made possible by the Free Python Software Foundation. Python has its own memory management system as well as its own dynamic type system. It comes with a large and extensive standard library and supports a variety of programming paradigms, like imperative, functional, procedural, and object-oriented.

Django

Django, a high-level Python web framework, encourages rapid iteration and logical, beautiful design. It was designed by experienced programmers and handles most of the pain associated with web development, allowing you to focus on developing your application instead of reinventing the wheel. It is open source and completely free.
Django's main goal is to make it easy to build complex, database-driven websites. Django prioritizes rapid development, the don't repeat yourself principle, and component reusability and "pluggability". Python is used everywhere, including configuration files and data models. Django also provides a customized administrator create, read, update, and delete interface that is created dynamically through introspection.

UML charts

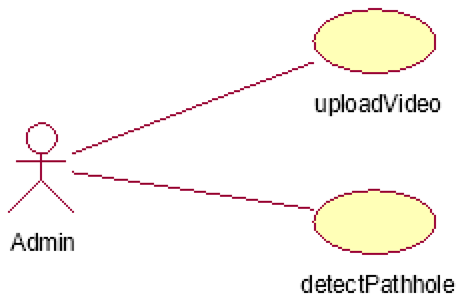
UML stands for Unified Modeling Language. Object-oriented software engineering uses UML, a design language with guidelines that can be used for a wide variety of projects. The standard was created and is monitored by the Object Management Group.

UML aims to become a recognized language for modeling object-oriented computer software. UML now consists of two main parts: a meta-model and syntax. In the future, UML can potentially be combined or enhanced by technology or process. Unified Modeling Language is a standard language for creating, visualizing, and documenting artifacts of software systems as well as business modeling and non-software systems.

UML is a synthesis of great engineering practices that have proven successful in simulating enormous, complex systems.

USE CASE DIAGRAM:

Use cases create and create usage diagrams, a type of behavior diagram in Unified Modeling Language (UML). The goal is to graphically represent the relationship between actors, their goals (as expressed in use cases), and dependencies between use cases in a certain system. The main purpose of the use case diagram is to identify which actors perform which system functions. Each system participant can be identified by the function it plays.



CLASS DIAGRAM:

Class diagrams are a type of static structural diagram used in software engineering's Unified Modelling Language (UML) to depict classes, properties, actions, and connections between the classes that make up a system. It clarifies what kind of data is included.

Sequence Diagram

A system diagram in UML shows the interdependencies between various entities and the conditions under which they operate. Temporal sequence diagrams are also known as "scenario diagrams," "event scenarios," and "scenario charts." Focuses on developing improvements in the software sector. When discussing the architecture of software projects using UML, visual symbols are often used.

ACTIVITY DIAGRAM:

The workflow is represented graphically in a diagram of activities, supporting choices, iterations, and consensus. Activity diagrams can be used to describe operations and workflows of system components in a unified modeling language. A flowchart shows the overall control flow.

State Diagram

A statechart diagram is used to describe the state of many objects in their life cycle. A change in state resulting from some internal or external event is highlighted. This is important for accurate analysis and implementation of the state of the object.

States are best described using state diagrams. A state is defined when an event occurs.

Before creating a Statechart, the following points should be clarified:

• important items to check.

• Count numbers.

• events.

The Statechart image below shows how the state of an Order object is reviewed

The process starts in the first state, which is idle. The following conditions have occurred for actions such as sending a request, confirming a request, and sending an order.

Component diagram :

Component diagrams are used to describe a system's tangible components. There are files, executables, libraries, etc. in this item. This diagram serves a distinct purpose. During the application implementation phase, component diagrams are employed. However, it is planned out in great detail for the actual implementation.

The system is initially created using various UML diagrams, then once the artefacts are complete, component diagrams are utilised to obtain a sense of how the implementation will work.

Without this diagram, the application cannot be implemented effectively, hence it is crucial. Additionally, additional factors like application performance, maintenance, etc. depend on a well created component diagram.

It is important to explicitly identify the following artifacts when designing a component diagram:

• The files the system uses.

• Libraries and other application-related artifacts.

• The connections between the objects.

Following the identification of the artifacts, the following considerations must be made.

• Give the component for which the diagram is to be drawn a meaningful name.

• Before utilising any tools, create a conceptual layout.

• Use notes to elaborate on key issues.

A component schematic for an order management system is shown below. The artifacts in this case are files. The application's files and their relationships are depicted in the diagram. Actually, the component diagram also includes files like directories, libraries, and dlls.

Four files are shown in the diagram that follows, along with the relationships between them. Since component diagrams are created for very distinct purposes, they cannot be directly compared to the other

UML diagrams that have been presented so far.

Deployment diagram :

The deployment view of a system is represented by a deployment diagram. Because the components are deployed using deployment diagrams, it is related to the component diagram. In a deployment diagram, nodes are present. Nodes are merely the actual pieces of hardware that are used to deliver the programme.
System engineers can benefit from deployment diagrams. An effective placement chart is important because it adjusts for these variables:

• Productivity
• Density
• Manageability
• Continuity

Before creating a configuration diagram, it is important to find the following artifacts:

Points and relationships between points

The order management system layout is shown in the layout diagram below. These are the points that show us

Monitors, modems, caches and servers

This application is considered a web-based application hosted in a clustered environment on 1, 2, and 3 servers. Users use the internet to access applications. The integrated environment takes control from the host.

TESTING METHOD

The testing methodologies are as follows:

A unit test.
Testing the integration.
Testing for user acceptance.
Validation Testing.
Output Testing.

Unit Testing

Unit testing focuses testing efforts on modules, the smallest units of software design. For complete coverage and maximum error detection, the testing unit executes individual paths in the module's control hierarchy. These tests focus on each module to make sure they all work together. Unit Testing is the resultant name. Each module is tested separately during this testing, and the module interfaces are checked for conformance with the design specification. All significant processing routes are examined for the anticipated outcomes. Every path for handling errors is tested as well.

Integrity Checks

Concurrent issues of verification and program creation are addressed through integration testing. After integrating the software a series of higher order tests are run. The main goal of this testing process is to take the modules that have passed unit testing and assemble the program with the structure defined by the design.

User Acceptance Testing

User approval is the most crucial factor in any system's success. The system in question is tested for user acceptance during development by continuously talking to potential users of the system and making adjustments as needed. The designed system offers a user interface that is welcoming and simple enough for someone who is unfamiliar with the system to understands

Testing of output

The testing of the output of the proposed system comes after the testing of its validity. It is unable to deliver the necessary output in the format that is required. In order to evaluate the produced or shown by the system under consideration, users are questioned on the format that they like. Because of this, there are two ways to think about the output format: either on the screen or in printed form.

MAINTAINENCE

This includes a wide range of tasks, such as fixing code and design flaws. We have more precisely defined the user needs during the system development process to decrease the long-term demand for maintenance. This system has been created to the greatest extent possible in order to meet the requirements. Future technological advancements may make it possible to add a lot more functions dependent on demand. Maintenance will be simpler because to the straightforward coding and design.

CONCLUSTION

Due to different settings for data collection and the subjective nature of interpretation are more image databases. Researchers often test their algorithms on specific databases and can achieve good results. Preliminary tests of the database showed that there were inconsistencies between the databases due to different compilation environments and build specifications; As a result, algorithms evaluated using database protocols do not generalize to test unseen data and perform worse in database settings. Alternatives to this bias include deep domain adaptation and knowledge distillation. In addition, the performance of FER cannot be improved when increasing the training data by directly concatenating several datasets due to the lack of overlap of expression markers. Another problem in facial expressions is the class inequality that arises from the difficulty of collecting data: while it is simple to create and interpret funny expressions, it can be very difficult to record data. Performance measured by average accuracy, which weighs all classes equally, decreases compared to accuracy measures, as shown in Tables 4 and 7, and this decrease occurs in real world databases (e.g., SFEW 2.0 and AFEW). One of the methods is multiplying and synthesizing data during preprocessing to facilitate class distribution.

REFERENCES

[1] C. Darwin and P. Prodger, The expression of the emotions in man and animals. Oxford University Press, USA, 1998. [2] Y.-I. Tian, T. Kanade, and J. F. Cohn, "Recognizing action units for facial expression analysis," IEEE Transactions on pattern analysis and machine intelligence, vol. 23, no. 2, pp. 97–115, 2001. [3] P. Ekman and W. V. Friesen, "Constants across cultures in the face and emotion." Journal of personality and social psychology, vol. 17, no. 2, pp. 124–129, 1971. [4] P. Ekman, "Strong evidence for universals in facial expressions: a reply to russell's mistaken critique," Psychological bulletin, vol. 115, no. 2, pp. 268–287, 1994. [5] D. Matsumoto, "More evidence for the universality of a contempt expression," Motivation and Emotion, vol. 16, no. 4, pp. 363–368, 1992. [6] R. E. Jack, O. G. Garrod, H. Yu, R. Caldara, and P. G. Schyns, "Facial expressions of emotion are not culturally universal," Proceedings of the National Academy of Sciences, vol. 109, no. 19, pp. 7241–7244, 2012. [7] Z. Zeng, M. Pantic, G. I. Roisman, and T. S. Huang, "A survey of affect recognition methods: Audio, visual, and spontaneous expressions," IEEE transactions on pattern analysis and machine

intelligence, vol. 31, no. 1, pp. 39– 58, 2009. [8] E. Sariyanidi, H. Gunes, and A. Cavallaro, “Automatic analysis of facial affect: A survey of registration, representation, and recognition,” IEEE transactions on pattern analysis and machine intelligence, vol. 37, no. 6, pp. 1113–1133, 2015. [9] B. Martinez and M. F. Valstar, “Advances, challenges, and opportunities in automatic facial expression recognition,” in Advances in Face Detection and Facial Image Analysis. Springer, 2016, pp. 63–100. [10] P. Ekman, “Facial action coding system (facs),” A human face, 2002.