software

MDPI

Article

Vision-Autocorrect: A Self-Adapting Approach towards Relieving Eye-Strain Using Facial-Expression Recognition

Leah Mutanu [1,*], Jeet Gohil [1] and Khushi Gupta [2]

[1] Department of Computing, United States International University Africa, Nairobi P.O. Box 14634-0800, Kenya; jgohil@usiu.ac.ke
[2] Department of Computer Science, Sam Houston State University, Huntsville, TX 77341, USA; kxg095@shsu.edu
* Correspondence: lmutanu@usiu.ac.ke

Abstract: Abstract: The last two years have seen a rapid rise in the duration of time that both adults and children spend on screens, driven by the recent COVID-19 health pandemic. A key adverse effect is digital eye strain (DES). Recent trends in human-computer interaction and user experience have proposed voice or gesture-guided designs that present more effective and less intrusive automated solutions. These approaches inspired the design of a solution that uses facial expression recognition (FER) techniques to detect DES and autonomously adapt the application to enhance the user's experience. This study sourced and adapted popular open FER datasets for DES studies, trained convolutional neural network models for DES expression recognition, and designed a self-adaptive solution as a proof of concept. Initial experimental results yielded a model with an accuracy of 77% and resulted in the adaptation of the user application based on the FER classification results. We also provide the developed application, model source code, and adapted dataset used for further improvements in the area. Future work should focus on detecting posture, ergonomics, or distance from the screen.

Keywords: digital eye strain; facial expression recognition; software self-adaptation; image recognition

Citation: Mutanu, L.; Gohil, J.; Gupta, K. Vision-Autocorrect: A Self-Adapting Approach towards Relieving Eye-Strain Using Facial-Expression Recognition. *Software* **2023**, 2, 197–217. https://doi.org/10.3390/software2020009

Academic Editor: Stefan Wagner

Received: 17 January 2023
Revised: 11 March 2023
Accepted: 20 March 2023
Published: 29 March 2023

1. Introduction

Globally, media outlets have highlighted the increased duration of time that adults and children are spending on digital screens. For example, in the United Kingdom, it was reported that people spend up to three and a half hours each day looking at TV screens, four hours staring at laptops, and two hours on mobile phones [1]. This figure almost doubled when stay-at-home measures were enforced due to the COVID-19 health pandemic, with adults spending an average of six and a half hours each day in front of a screen [2,3]. The average American teen spends 7 h and 22 min on a screen outside of their regular schoolwork [4]. Today many learning and work-related activities have gone online, forcing people to spend more and more time on the screen. Staring at your screen for long hours each day can often result in dry eyes or eye strain, gradually contributing to permanent eye problems such as Myopia [3].

The increased internet use for research requires users to navigate through web pages designed using different fonts, font sizes, font colors, and background colors. These websites do not always meet the basic requirements for visual ergonomics. The online eLearning trend requires teachers to post materials on learning management systems (LMS) without paying much attention to the content's appearance. Content developers use their discretion to identify fonts and backgrounds. This freedom often leads to the publishing of content that is difficult to read, even for users with no visual disabilities. As a user navigates through online content, the difference in content presentation introduces temporary visual challenges. Users strain their eyes as they try to adjust to different display settings. Similarly, the user's environment can temporarily influence their ability to read

the information on a screen, for example, where the lighting in a room is poor or in a scenario where the screen is too close or too far.

A popular approach for ensuring that technology addresses user disabilities is assistive technologies, which calls for specialized products that aim at partly compensating for the loss of autonomy experienced by disabled users. Here the user is required to acquire new technology or adapt existing technology using available tools before using the technology. Where user requirements are not known a priori or dynamically change, the approach is ineffective because it forces redeployment or reconstruction of the system [5]. Additionally, the degree of disabilities varies widely in severity, and often mild or undiagnosed disabilities go unsupported. Further, persons with mild disabilities tend to shun assistive technology because it underlines the disability, is associated with dependence, and degrades the user's image [6], thus impairing social acceptance. The net result is that many users have become accustomed to squinting or glaring their eyes to change the focus of items on the screen. Some users will move closer or further from the screen depending on whether they are myopic or hyperopic. In such cases, the burden of adapting to the technology resides with the user's behavior. This approach can present further health challenges to the user, such as damaging their posture.

This research proposes a technique that shifts the burden of adjusting the computer settings from the user to the computer with minimal user involvement. The use of intelligent techniques that can learn and predict the most suitable adaptations autonomously [7] presents potential solutions. Integrating assistive technology into mainstream technology will make it more acceptable, and this calls for new approaches in technology design. The study adopts a self-stimulation approach, where the solution autonomously generates and tests scenarios based on user behavior without affecting the live system [8]. The solution leverages facial expression recognition (FER) techniques to detect eye strain. The main contributions of the research are the development and evaluation of a solution that can autonomously test and detect digital eye strain (DES) at runtime and adapt applications to offer relief for the user. The research objectives, therefore, are (i) designing a seamless assistive technology model that relieves eye strain using FER techniques, (ii) implementing the solution using a self-adaptive approach that supports user-centered validation, and (iii) evaluating the effectiveness of the developed solution through laboratory simulations of facial expressions that mimic digital eye strain features. We organize the rest of the paper as follows: Section 2 discusses the related work in this area, Section 3 presents the solution designed, Section 4 describes the results, and the paper concludes in Section 5.

2. Related Work

Eye strain (medically referred to as asthenopia) refers to an ache felt inside the eye because of the stress of the accommodative and convergence mechanisms of the eye. Externally, a person experiences dryness, irritation of the eye, and compromised vision [9]. This often results in a user's attempt to improve vision by closing the eyes, squinting, or widening the eyelid. Eye strain often occurs while concentrating on a visually intense task such as reading fine print or in a poorly lit environment. Although the condition has no immediate ill health effects, it degrades the eye muscles if it goes on for prolonged periods. The increased use of digital platforms has recently increased eye strain leading to the evolution of new terms such as digital eye strain, computer vision syndrome, or visual fatigue [10,11]. Digital eye strain is defined as eye and vision problems associated with using computers and other electronic displays [12]. This study provides an autonomous solution for relieving digital eye strain, thus addressing a growing concern in the digital society.

2.1. Facial Expression Recognition

Recent advances in image-recognition algorithms have made it possible to detect facial expressions such as happiness, sadness, anger, or fear [13,14], with several reviews also touching on the subject [15,16]. Such initiatives find applications in detecting consumer satisfaction of a product [17–19] or in healthcare to diagnose certain health issues [20,21]

such as autism or stroke. Recently, FER has found applications in detecting fatigue, which can be dangerous, especially for drivers [22–24]. However, our review of the literature revealed limited evidence for using this technology to detect DES. One of the few studies identified used the blink rate and sclera area color to detect DES using a raspberry-pi camera [25]. The research, however, noted poor results with certain skin tones or where limited light-intensity difference between the sclera and skin region existed. Additionally, users with spectacles also generated reflections that interfered with color detection. Therefore, an approach that does not rely on color would address these limitations.

Eye tracking is increasingly becoming one of the most used sensor modalities in affective computing recently for monitoring fatigue. The eye tracker for such experiments also detects additional information, such as blink frequency and pupil diameter changes [26]. A typical eye tracker (such as video-oculography) consists of a video camera that records the movements of the eyes and a computer that saves and analyzes the gaze data [27]. The monitoring of fatigue using this approach differs from the monitoring of basic facial emotions (anger, contempt, disgust, fear, happiness, sadness, surprise) because specific facial points are monitored, such as the percentage eye closure (PERCLOS), head nodding, head orientation, eye blink rate, eye gaze direction, saccadic movement, or eye color. However, fatigue is expressed using a combination of other facial expressions, such as yawning or hands on the face. Therefore, we miss vital fatigue signals by focusing only on the eyes. Our proposed solution uses FER techniques, thus avoiding the light intensity and localization challenges faced when using the eye color or the eyes only to detect fatigue.

2.2. Machine Learning Techniques for FER

Recent studies on facial expression recognition [28] acknowledge that machine learning plays a big role in automated facial expression recognition, with deep learning algorithms achieving state-of-the-art performance for a variety of FER. This section describes the datasets, preprocessing techniques, and algorithms used in machine learning for FER.

2.2.1. FER Datasets

Studies using relatively limited datasets are constrained by poor representation of certain facial expressions, age groups, or ethnic backgrounds. To address this, the authors in [29] recommend using large datasets. In their review of FER studies, they note that the Cohn–Kanade AU-Coded Face Expression Database (Cohn–Kanade) [30] is the most used database for FER. A more recent review [15] introduced newer datasets such as the Extended Cohn–Kanade (CK+) [31] database, which they noted was still the most extensively used laboratory-controlled database for evaluating FER systems. It has 593 images compared to the original version, which only had 210 images. Another notable dataset introduced was the FER2013 [32], a large-scale and unconstrained database collected automatically by the Google image search API. The dataset contains 35,887 images extracted from real-life scenarios. The review [15] noted that data bias and inconsistent annotations are common in different facial expression datasets due to different collecting conditions and the subjectiveness of annotating. Because researchers evaluate algorithms using specific datasets, the same results cannot be replicated with unseen test data. Therefore, using a large dataset on its own is not sufficient. It is helpful to merge data from several datasets to ensure generalizability.

Additionally, when some datasets exhibit class imbalance, the class balance should be addressed during preprocessing by augmenting the data with data from other datasets. These findings motivated our decision to use more than one dataset as well as the use of large datasets. We used images from the CK+ and FER2013 datasets and conducted class balancing during preprocessing. Notably, most FER datasets had images labeled using the seven basic emotions (disgust, fear, joy, surprise, sadness, anger, and neutral). Therefore, preprocessing this study's data called for re-labeling the images to represent digital eye strain expressions such as squint, glare, and fatigue. This exercise called for manually reviewing the images and identifying those that fell in each class. We assigned

the images a new label representing the digital eye strain expressions. For instance,fatigue was labeled 1, whereas glare was labeled 2. To do this, the original images were rebuilt from the FER2013 dataset of pixels to enable the researchers to see the expressions. Once the labeling process was complete, a new dataset of pixels was generated with the new labels. By automatically detecting these facial expressions and autonomously adjusting font sizes or screen contrast, the user does not need to glare or squint to accommodate the screen settings. This also creates awareness for the user when an alert occurs, and they can take a break to address fatigue.

2.2.2. Image Processing and Feature Extraction

Image processing refers to the enhancement of pictures for ease of interpretation [33]. Common image processing activities include adjusting pixel values, image colors, and binarizing [34]. Several image-processing libraries that support these processes exist, such as OpenCV [35] and scikit-image [36]. They easily integrate with popular open-source machine-learning tools such as Python and R. After the image is preprocessed, feature extraction reduces the initial set of raw image data to more manageable sizes for classification purposes. Previous FER reviews [37] describe action unit (AU) and facial points (FP) analysis as two key methods used for feature extraction of classic facial emotion. Action units find applications when analyzing the entire face. This study used OpenCV Libraries in Python for image preprocessing and action units for feature extraction.

2.2.3. FER Algorithms

When our eyes squint, several things occur: the pupils get smaller as they converge, the eyelids pull together, and the edges of the eyelids fold to contract the cornea [9]. Sometimes the eyebrows could bend inwards, and the nose bridge moves upwards to enhance the eyes' focus. FER techniques can detect these expressions and alert the user or adjust text sizes and color contrasts in an application to relieve eye strain. The FER process generally involves the acquisition of a facial image, extracting features useful in detecting the expression, and analyzing the image to recognize the expression [29]. Machine learning algorithms such as deep learning neural network algorithms successfully perform FER. A popular algorithm, according to recent FER reviews [15,16], is the convolutional neural network (CNN), which achieves better accuracy with big data [38]. It has better effects on feature extraction than deep belief networks, especially for expressions of classic emotions such as contempt, fear, and sadness [15]. The results of these studies inspired the choice of CNN as the algorithm for implementing FER in this study [5].

However, it is worth noting that these results depend on the specific dataset used. For instance, the models that yielded the best accuracy in the FER2013 dataset are Ensemble CNNs with an accuracy of 76.82% [32], Local learning Deep+BOW with an accuracy of 75.42% [39], and LHC-Net with an accuracy of 74.42% [40]. The models that yielded the best accuracy in the CK+ dataset include ViT + SE with an accuracy of 99.8% [41], FAN with an accuracy of 99.7% [42], and Nonlinear eval on SL + SSL puzzling with an accuracy of 98.23% [43]. Sequential forward selection yielded the best accuracy on the CK dataset, with 88.7% accuracy [44]. The highest performing models on the AffectNet dataset are EmotionGCN with an accuracy of 66.46% [45], EmoAffectNet with an accuracy of 66.36 [46], and Multi-task EfficientNet-B2 with an accuracy of 66.29% [47]. Although numerous datasets exist for facial expression recognition, this study sought to detect expressions outside of the classic emotions. The absence of labeled datasets in this area called for relabeling of images. The choice of the dataset for relabeling the images was not crucial. Future research should seek to relabel images from larger datasets such as AffectNet.

With CNN, deeper networks with a larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturates [48]. Adding dropout layers increases accuracy by preventing weights from converging at the same position. The key idea is randomly dropping units (along with their connections) from the neural network during training. This prevents units from co-adapting too much [49]. Adding batch

normalization layers increases the test accuracy by normalizing the network input weights between 0 and 1 to address the internal covariate shift that occurs when inputs change during training [50]. Pooling layers included in models decrease each frame's spatial size, reducing the computational cost of deep learning frameworks. The pooling operation usually picks the maximum value for each slice of the image [51]. A summary of this process is depicted in Figure 1.

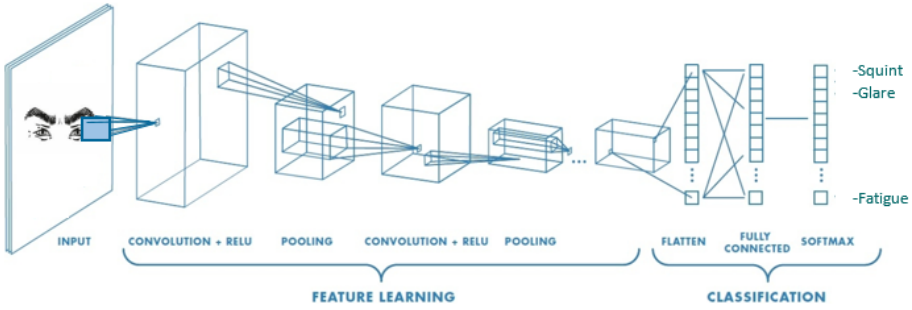


Figure 1. Distribution of facial expressions.

Popular CNN designs are based on the theoretical foundations laid by AlexNet [52], VGG [53], and ResNet [54]. AlexNet uses ReLu (rectified linear unit) given by f(x) = max (0, x) for the non-linear part instead of a tanh or sigmoid function, hence training is faster by resolving the vanishing gradient problem. AlexNet also reduces overfitting by using a dropout layer after every convolutional layer [55]. These present fewer computational requirements than VGGNets. VGGNets advocate for multiple stacked smaller-size kernels rather than one with a larger size because this increases the depth of the network [56], which enables it to learn more complex features. Increasing the depth, however, introduces other challenges, such as the vanishing gradient problem and higher training error values. ResNets address these challenges by introducing a global average pooling layer [57] and residual modules [58]. This reduces the number of parameters and increases the learning of the earlier layers. This study created and tested models based on these architectures to identify a suitable model for the research. Our future work will make use of more accurate approaches such as GoogLeNet.

3. Solution Design

Multimedia collections usually induce multiple emotions in audiences. The data distribution of multiple emotions can be leveraged to facilitate the learning process of emotion tagging, yet few studies have explored this [59]. As proof of concept, we developed applications that use machine learning for real-time facial expression detection to detect the presence or absence of digital eye strain features on the face. A self-adaptation process follows to relieve digital eye strain. We named the application Vision Autocorrect, which symbolizes its ability to edit a user application and automatically correct their temporary computer vision syndrome.

3.1. Developing the Machine Learning Model

A machine learning model is a file that has been trained to recognize patterns in data. It is trained over a set of data using an algorithm that can reason over these data and learn from them. After training the model, it is used to reason over data it has not seen before. This section describes how we built the model through the training process.

3.1.1. Data Labeling

During this process, facial images from FER datasets were manually relabeled with digital eye strain expressions. A total of 36,540 images from FER2013 [31] and CK++48 open

datasets were relabeled from the classic facial expression emotions (happiness, sadness, anger, or fear) to digital eye-strain expressions (squint, glare, and fatigue).

The final dataset (The integrated and relabeled dataset posted on GitHub, https://github.com/Jeetg57/VisionAutocorrectDataset, accessed on 11 March 2023) consists of 6513 images labeled Fatigue, 4188 images labeled Glare, and 9022 images labeled Squint. The rest of the images formed the normal (6546) and negative/none expressions (10,271), as summarized in Figure 2. During the labeling process, images with inward eyebrows and squeezed eyes formed the Squint class, whereas those with arched eyebrows and gorged eyes fell in the "Glare" class, as shown in Figure 3. Images with eyes closed, hands on faces, or yawning were labeled Fatigue, and those without any of these emotions formed the Normal class. This study only focused on expressions of fatigue manifested through facial expressions such as prolonged eye closure and the presence of hands near the face. Facial expressions that did not manifest a squint, glare, or fatigue expression were classified as Normal. For the study, the "none" class comprised expressions where the user was not facing the computer or there were no facial images. We assumed that where the user was facing away from the computer, the facial expressions were not a result of the computer and hence should not be addressed by adapting the user applications.

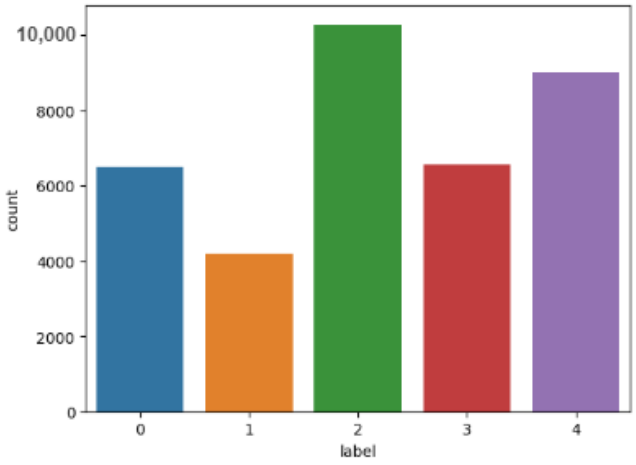


Figure 2. Distribution of facial expressions.

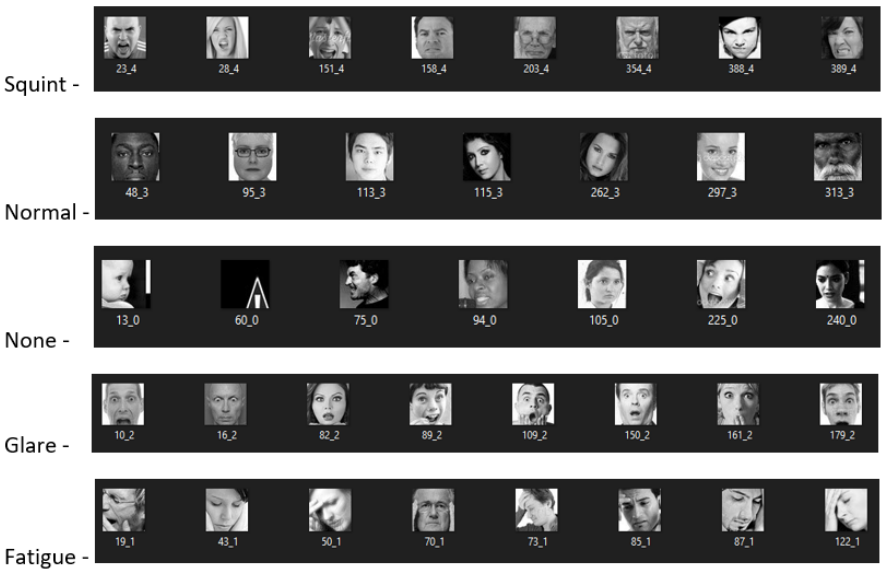


Figure 3. Re-classified facial expressions from [32] and CK++48 [31] datasets.

The extracted pixels of the labeled images formed the training and test data. They were normalized and resized into 48 × 48 grayscale images. The dataset was split into a training set of 29,234 images (80%) and a test set of 3653 (10%) images. The rest of the images (10%) formed private test data used for prediction when evaluating the final model.

3.1.2. Model Development

The process started by training several machine-learning models that use CNN algorithms for image classification. We used Tensorflow and Keras libraries, which are open-source libraries with a Python interface for artificial neural networks. Widely used in computer vision, convolutional neural networks comprise layers of artificial neurons called nodes. The nodes calculate the weighted sum of input pixel values and return a map of important features, such as colors, in the form of an activation function. Three dimensions describing the height, width, and many color channels represent an image mathematically as a tensor. The filter (F—learnable weights), padding (P—elements added on each of the four sides of the image), and stride (S—a step taken in the convolutional product) determine each of these dimensions (d) calculated using Equation (1).

$$Output_d = \left(\frac{Input_d - F_d + 2P}{S_d} \right) \quad (1)$$

We developed and tested several models based on popular CNN architectures to identify a suitable model. The models used the CNN architectures of AlexNets, VGGNets, and ResNets. In this paper, we report six models that provided significant results (Model ID 1, 2, 5, 6, 7, and 8). The choice of the models was based on results obtained from laboratory simulations and user feedback. The evaluation criteria indicated how quickly and accurately a model could detect specific expressions. In Table 1, we summarize the architecture of the selected models and the number of convolutional layers used.

Table 1. Architecture of selected models.

Model ID	Architecture	Number of Convolutional Layers
3	AlexNets [42]	4
7		4
8		8
1	VGGNets [43]	6
6		9
5	ResNets [44]	7, 8 Separable

A virtual machine deployed in Azure was used to train the modules. The virtual machine had six cores with a total of 56 RAM and 380 GB of disk storage. The GPU used was an NVIDIA Tesla K80.

3.2. Digital Eye Strain Expression Recognition

The classification of previously unseen images starts by capturing frames of the user's faces while the user is using an application in real time. This image undergoes several preprocessing steps before classification. The steps, illustrated in Figure 4, include converting the image to grayscale, detecting the face, cropping the face, resizing the image to 48 × 48 pixels, and finally converting the image to a flattened pixels file.

$$h_{j(x)} = \int_{0\ otherwise}^{1\ if\ \ p_j f_{j(x)} < p_j \theta_j)} \quad (2)$$

This study used the Viola–Jones facial detection algorithm, which uses rectangles to identify features in an image [60]. The algorithm trains a classifier using the Adaboost

machine-learning algorithm, an ensemble algorithm that combines the output of several weak classifiers to generate a strong classifier. For each feature (fj) and parity pj, the weak learner (hj(x)) determines the optimal threshold (θj) classification function given in Equation (2), such that the minimum number of examples are misclassified. The output of this classifier detects faces in an image.

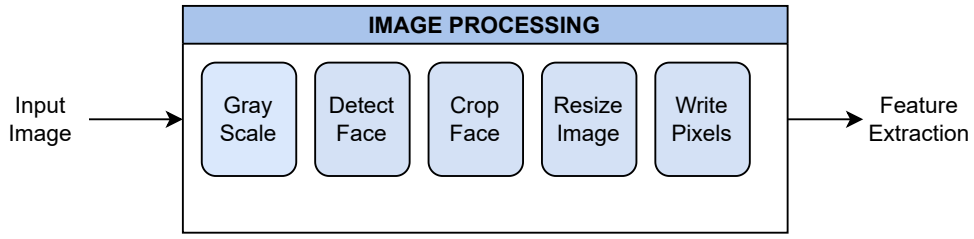


Figure 4. Image preprocessing steps.

This approach allows for the detection of image features in real time. It also works well for frontal faces, which is ideal for detecting features for users working on a computer. Our experiments showed that it worked well for users in rooms with different lighting, users with different skin tones, and those who wore glasses. Our future work will explore opportunities for improving the proposed solution through other methods, such as Kanade–Lucas–Tomasi facial detection algorithms [61]. To improve the accuracy of the proposed model, we employed preprocessing techniques to resize the captured images, convert them to grayscale, and remove noise. Future research will explore how additional preprocessing techniques can improve the accuracy of the proposed solution.

We cropped the detected faces to retain an image with a face only, removing external features that would reduce the prediction accuracy. The flattened pixels of the final image undergo feature extraction. This process used action units, which are numeric codes that describe the activity of different parts of the face, such as cheek raising, brow lowering, and nose wrinkling, or upper lips movements. During classification, comparing these action units with those in the developed model help in making predictions based on the closest match. The predicted class represents the emotion, as illustrated in Figure 5.

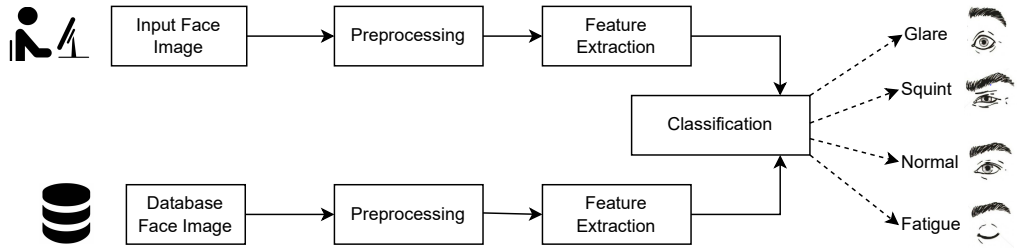


Figure 5. Image preprocessing steps.

3.3. Self-Adaptation System

In this study, the self-adaptation process called for developing a user's application that is reconfigured based on the expression detected and predefined decision rules.

3.3.1. Design and Implementation

The self adaptive system uses a three-tier architecture with presentation, application, and database tiers. The presentation tier monitors the users' on-screen behavior by capturing the facial image. The application layer analyzes the image to detect digital eye strain expressions. This analysis is accomplished through image preprocessing, feature extraction, and classification processes. Following this analysis, predefined decision rules, informed by experts and the user's previous behavior, determine a set of actions to adjust

the client application upon detecting eye strain. The system seeks user consent before making any adjustments, hence the user can accept or reject the settings. The user decision acts as a feedback loop used for training the system in the future. In this way, we design a self-adaptive system that automates the process of relieving digital eye strain, as illustrated in Figure 6.

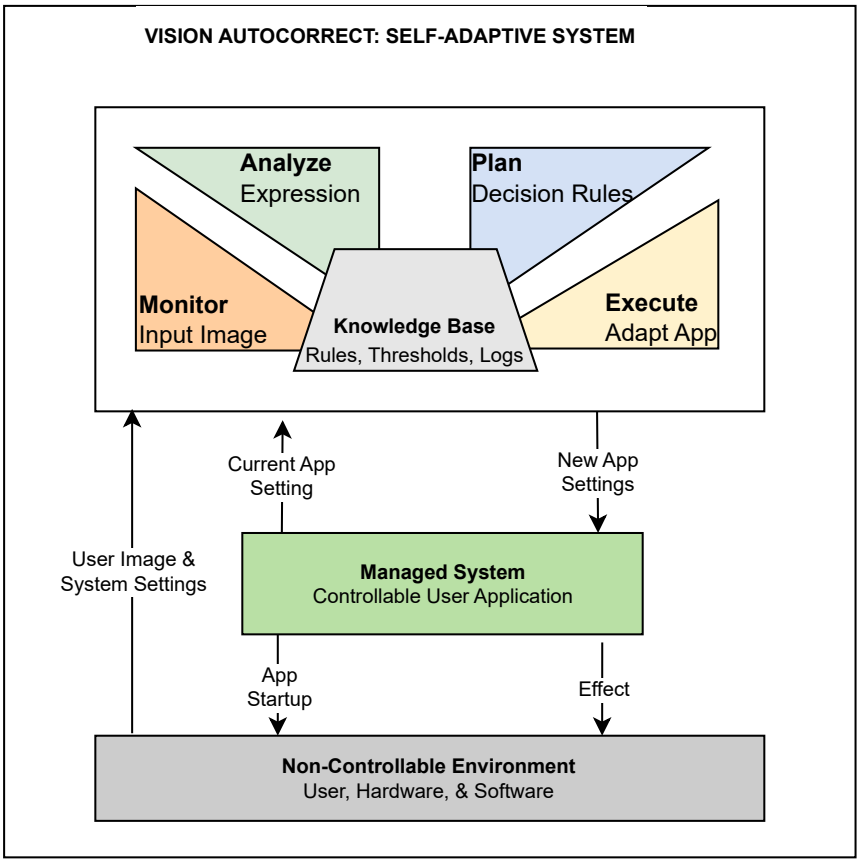


Figure 6. Vision-autocorrect self-adaptive architecture.

As proof of concept, we developed two prototype applications to evaluate the proposed solution based on two use cases. Users typically access software as stand-alone applications installed on their devices or as web-based applications. The proposed solution can work for both scenarios, and we developed prototypes for each use case to evaluate this. The first use case was a web application that represents users reading content on the web. We used JavaScript, HTML, and CSS to publish content for users to read. The web server was designed using the Flask package. Asynchronous HTTP POSTs using jQuery and Ajax methods load captured images from the user to the server. The system monitors the persistence of eye strain expressions within a given period. For example, if more than five consecutive image samples fall under the "Squint" class, then we conclude that the user exhibits a prolonged squint, and an action is required to relieve this. The second use case was a python graphical user interface application that represents users reading content from any software application that is not web-based. The application source code, decision rules, and trained models are available on GitHub (https://github.com/Jeetg57/Vision-Autocorrect-Application, accessed on 11 March 2023). We used a Python GUI toolkit to display the user interface, OpenCV to show a frame from the camera, TensorFlow to predict the class of the image, and XML Element tree library to parse an XML file, which contains decision rules to govern what the application does after classification. Like the web-based application, the open-source machine-learning libraries

OpenCV and TensorFlow implement the Analyze logic using the convolutional neural network trained model.

When the application is in use, pictures of the user taken at an interval of five seconds determine whether the user is squinting, glaring, or fatigued. The application analyzes the picture and logs the expression. Once an expression log exceeds a predefined threshold value, the adaptation process starts by alerting the user. For example, if the user is squinting, the application alerts the user and seeks consent to increase the font size. The developed application periodically deletes the images taken to maintain the user's privacy and save on memory resources.

3.3.2. Decision Rules

We employed decision rules to determine the action required based on the expression detected. A decision rule is a simple IF–THEN statement consisting of a condition (called antecedent) and a prediction (consequence). For example: IF squint (condition), THEN increase the font size (prediction). We used a combination of several rules to make predictions for different facial expressions. Relevant actions were determined by inducing decision rules, creating "if–else–then" type rules when a classified expression is received/detected and generating an output variable representing the alert. The algorithm provided in Figure 7a illustrates the self-adaptation process. The glare and squint expressions called for adjustments of font sizes or colors, whereas the fatigue expression called for a user notification only. Other expressions, such as normal and none, called for no action, as illustrated by the decision rules in Figure 7b. Where an action is required, such as adjusting the font sizes, the application adjusts the font size gradually until the user's facial expression normalizes. The user can also opt to reject any adaptation decision made.

```
1.  DO
2.    image ← postRequest()        -jQuery and Ajax
3.    class ← Classify(image)      - Emotion recognition
4.    IF class = EXPRESSION THEN
5.        Increment Expression_Count
6.    END IF
7.    IF Expressiont_Count > threshold THEN
8.        OUTPUT: Expression alert
9.        INPUT: User Consent
10.       IF User_Consent = true THEN
11.         Rules ← readRules()    –Parse XML file
12.         Execute(Rules)
13.         Reset Expression_Count
14.         Delete all Images
15.       END IF
16.   END IF
17. WHILE (App in Use)
```

(**a**)

```
<rules>
    <rule category="Squint">
        <tolerance>5</tolerance>
        <message>You are squinting, do you want to increase font size?</message>
        <Decision>0</Decision>
    </rule>
    <rule category="Glare">
        <tolerance>5</tolerance>
        <message>You are glaring, do you want to increase font size?</message>
        <Decision>0</Decision>
    </rule>
    <rule category="Fatigue">
        <tolerance>5</tolerance>
        <message>You are fatigued, would you like to take a break?</message>
        <Decision>1</Decision>
    </rule>
        <rule category="None">
        <tolerance>5</tolerance>
        <message></message>
        <Decision>2</Decision>
    </rule>
        <rule category="Normal">
        <tolerance>5</tolerance>
        <message></message>
        <Decision>2</Decision>
    </rule>
</rules>
```

(**b**)

Figure 7. (**a**) Self-adaptation algorithm and (**b**) decision induction rules.

4. Experimental Results and Discussion

We describe the early experimental results obtained from testing the developed model on a sample of users while using digital devices for their daily activities. The results are based on the user's perception of the accuracy of the model and simulated laboratory experiments on the model's ability to detect unseen faces.

4.1. Model Evaluation

To identify a suitable model for the study, we implemented and tested several models. Most of the models stopped training after 30–35 epochs, as seen in Appendix A. The results presented in Table 1 show the training and user evaluation tests. The selected users were

volunteer college students whose ages range from 19 to 24 years. This age group is highly susceptible to digital eye strain given their frequent need to conduct research for their studies using online devices and access social media for entertainment. They were required to use a device with the proposed solution installed and rate how quickly and accurately it detected their expressions.

Although the training results ranked Model IDs 6, 7, and 8 highly as shown in Figure 8, users observed that Models IDs 1 and 6 appeared to perform better as outlined in Table 2. In the table the symbols ○, ◔, ◕, and ● are used to represent the accuracy status ranging from the least to the most models respectively as perceived by the users. The results revealed that Model ID 6 was the most accurate according to both users and the training results obtained. Based on this, Model ID 6 was used to develop the final application, available as a Jupyter notebook for reproducibility on: https://github.com/Jeetg57/VisionAutocorrect/blob/main/Model%206/6R1/Model%206R1.html (accessed on 10 March 2023).

We evaluated the resources used to train our model to establish whether accuracy was obtained at the expense of resources. The results illustrated in Figure 9a showed that Model 8, which was a relatively deeper AlexNet network, took the longest time to complete training. In contrast, the shallower AlexNet networks (Models 7 and 3) took a shorter time. By stacking up the convolutional layers, as in the case of VGGNets, the time taken significantly reduces even for deeper networks such as Model 6. The amount of time taken is proportional to the GPU power usage, as seen in Figure 9b. It is worth noting, however, that Process Memory Available was slightly higher for the VGGNets architectures compared to the AlexNet architectures. The ResNet model (Model 5) consumed relatively fewer resources but did not perform as accurately. Despite the accuracy obtained, the selected model (Model 6) did not use comparatively more resources. This highlights the suitability of the approach for constant model training based on collected user feedback.

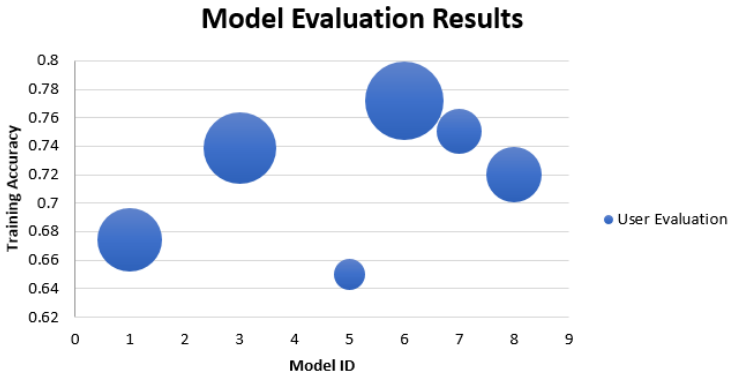
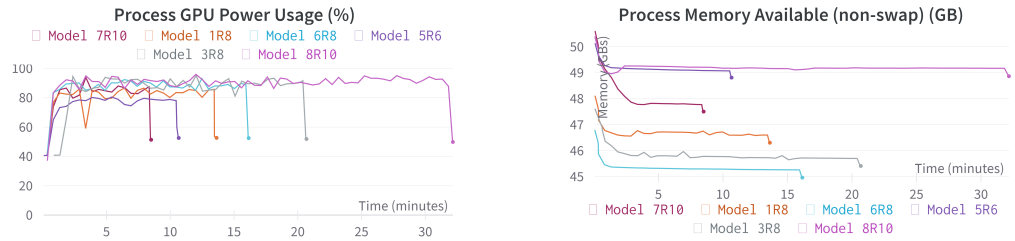


Figure 8. Distribution of facial expressions.



(**a**) (**b**)

Figure 9. Resource utilization metrics for (**a**) GPU power usage (**b**) process memory available.

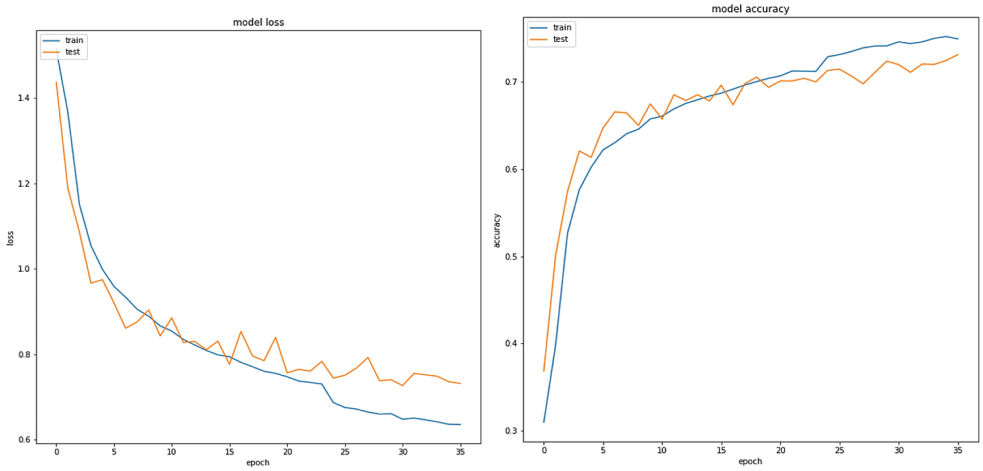
Table 2. Model evaluation results.

	Training Results		User Tests Results				
Model ID	**Attempt**	**Accuracy**	**Squint**	**Glare**	**Fatigue**	**Normal**	**Rank**
Model 1	1	0.634043694	○	●	○	○	5
	2	0.651729226	○	○	○	○	4
	3	0.674306452	○	○	○	○	1
	4	0.660965323	○	○	○	○	2
	5	0.655970991	○	○	○	○	3
Model 3	1	0.692402422	●	○	○	●	4
	2	0.676529944	●	●	●	●	5
	3	0.738822579	○	○	○	○	1
	4	0.705298781	○	○	○	○	2
	5	0.694386482	○	●	○	○	3
Model 5	1	0.648308396	○	○	●	◔	1
	2	0.69226557	●	○	●	◔	2
	3	0.652721226	●	○	●	●	3
	4	0.666575432	●	●	●	●	4
	5	0.647453249	●	●	●	●	5
Model 6	1	0.747887671	○	○	●	○	2
	2	0.771764815	○	○	○	○	1
	3	0.702151656	○	◒	○	○	5
	4	0.708172262	○	○	◒	○	4
	5	0.738788366	○	○	○	○	3
Model 7	1	0.732459903	●	○	○	○	3
	2	0.724934161	●	○	◔	○	4
	3	0.719495118	●	○	◔	●	5
	4	0.750555873	◔	◔	◔	○	1
	5	0.746450901	●	◔	◔	○	2
Model 8	1	0.745014191	●	○	○	○	3
	2	0.655389428	●	○	●	●	5
	3	0.728560209	○	○	●	○	4
	4	0.726165652	○	○	◔	○	2
	5	0.723497391	○	◔	○	○	1

Key: ○Accurate, ◔ Partially Accurate, ◒ Partially Inaccurate, ● Inaccurate

A loss curve during training is one of the most widely used plots to debug a neural network. It provides a snapshot of the training process and the direction in which the network learns. We stopped the training process when the validation and training errors

stopped dropping. The gap between training and validation accuracy is a clear indication of overfitting. The larger the gap, the higher the overfitting. The results presented in Figure 10a show that the validation loss started to increase for the selected model at epoch 30. Hence, the training was stopped early (early stopping) to prevent the model from overfitting. We accounted for this by adding a delay using the patience parameter of 5 before the training was stopped, i.e., after the point where the validation loss starts to increase (indicating model performance has reduced). Similarly, training and validation accuracy also helps assess the model to prevent overfitting. The training stopped when the validation accuracy is equal to or slightly less than the training accuracy, as illustrated in Figure 10b. We used huge datasets to train the model and neural networks with regularization to select relevant features and reduce overfitting. Future studies should employ cross-validation and other techniques to improve accuracy by reducing overfitting.



(**a**) (**b**)

Figure 10. Selected models training and validation (**a**) loss and (**b**) accuracy metrics.

The final model had an accuracy level of 77%. A closer examination of the model's accuracy using the resultant confusion matrix illustrated in Figure 11 revealed that some expressions, such as glare, were more accurately classified than others, such as fatigue. We attributed this to the accuracy of the image labeling process. Because some Fatigue images closely resembled Squint images, there were some overlaps. Cleaning the dataset further would enhance this accuracy.

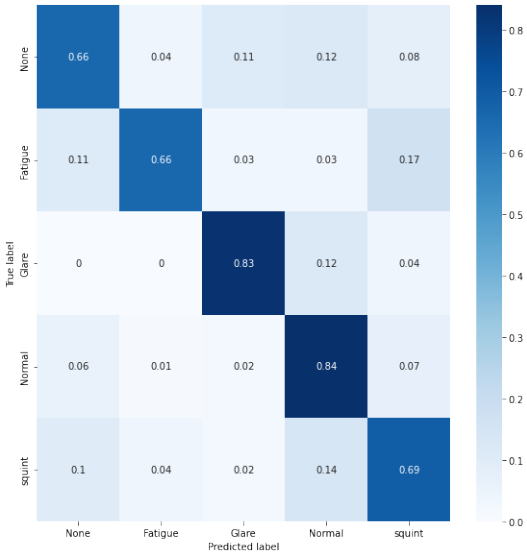


Figure 11. Selected models confusion matrix.

The selected model has nine convolutional layers used to classify facial expressions by identifying features in image patterns. Because a large set of input features may have many small feature patterns, different layers used enhanced different patterns. Three maximum pooling layers are used with a 2×2 pool size applied on fully connected layers to avoid overfitting of the data. The max-pooling layers downsize the features in an image by preserving only the convolved features. Classical neural network layers, called dense layers, assist in learning. We used the ReLU activation function for the neural network learning and the adaptive moment estimation algorithm (ADAM) to optimize the final model, hence improving performance. Figure 12 shows the detailed network architecture.

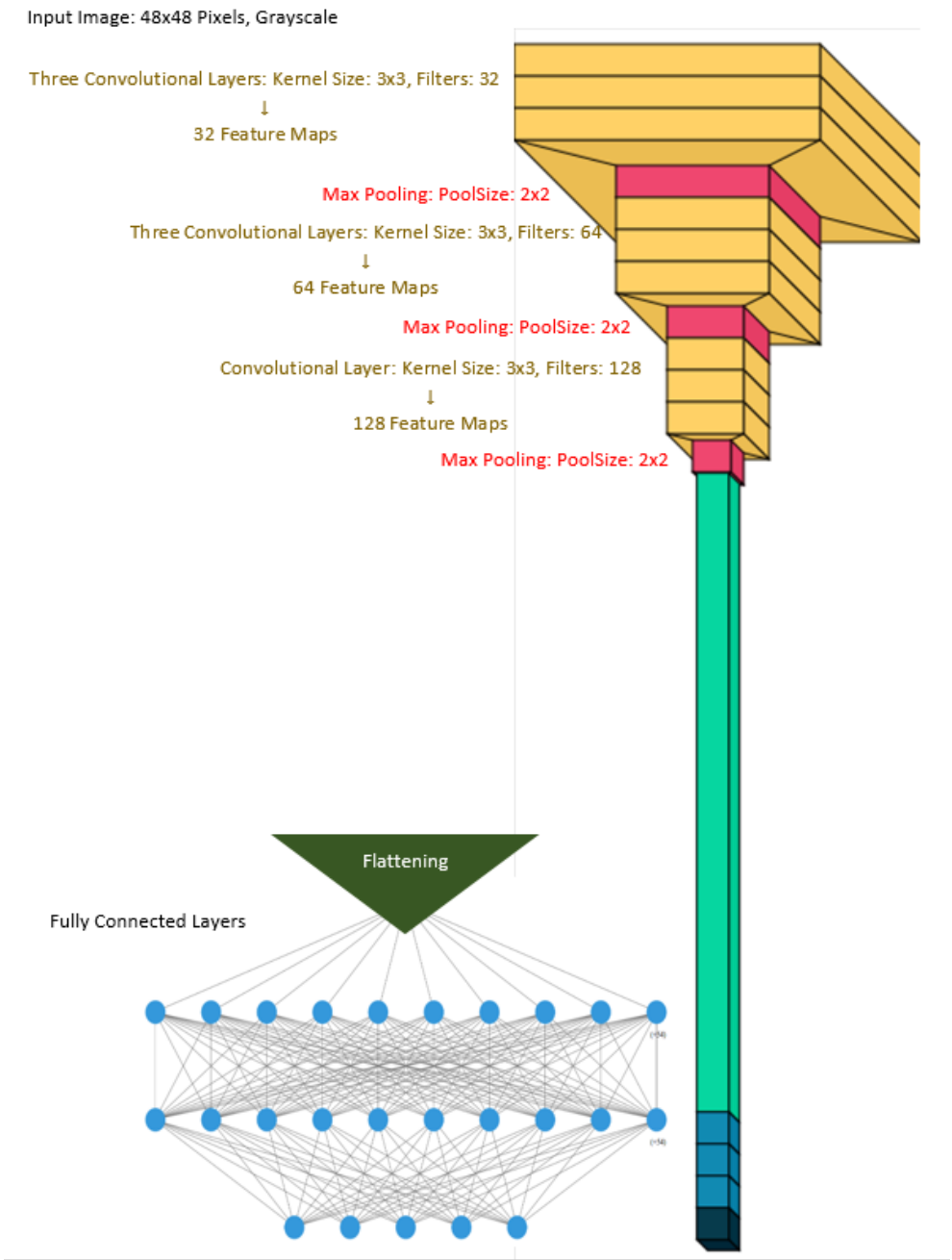


Figure 12. The selected neural network architecture.

4.2. Evaluating the Facial Expression Recognition Process

Image preprocessing occurs in real time while the user is using the application. During this time, a video frame is captured periodically, converted to grayscale, cropped, and

resized, and a pixel file is generated. The classifier predicts the expression on the preprocessed image. The predicted expression displayed on the real-time video from several tests helped evaluate the process. Different users conducted several tests to evaluate the model generalizability using simulated tests. The results presented in Figure 13 show that the model accurately detects the expression for users with spectacles, without spectacles, or with different skin tones. The model also detected different expressions with acceptable accuracy levels where the lighting in the environment differed, such as during the day with natural light or at night with artificial lighting.

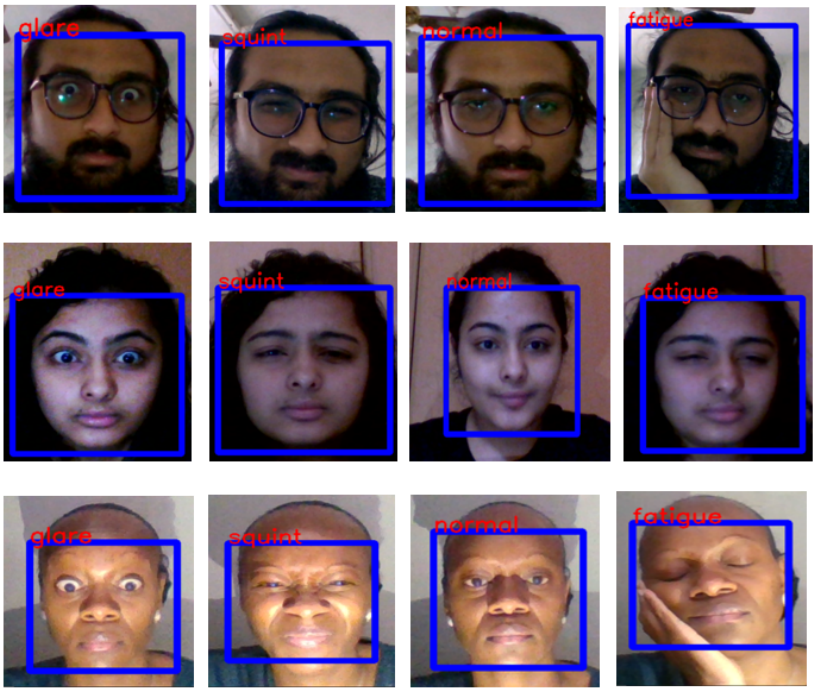


Figure 13. Expression detection results.

4.3. Evaluating the Self-Adaptation Process

To evaluate the self-adaptive application developed, we displayed the output of the classification process alongside the decision taken. The results displayed in Figure 14 show the output of four consecutive emotions logged as fatigue. These images are taken at intervals of 5 s while using the application. This means that the user continuously exhibited signs of fatigue for 20 s. When the fifth reading shows the same expression, the application issues an alert. The threshold period is adjustable to suit user preferences or expert recommendations. In this case, the recommended action was informing the user to take a break. If the user issues consent, the application closes. A different test, Figure 15, shows a user opening a web page that has poorly selected background and text colors. This makes it difficult to read, and the user starts glaring. The results show the app notifying users with a dialog box for their consent. When the user consents, the background and text colors are reverted to the default black and white colors. Both tests discussed here show how the decision rules behave when an adaptation action is required (fatigue or glare).

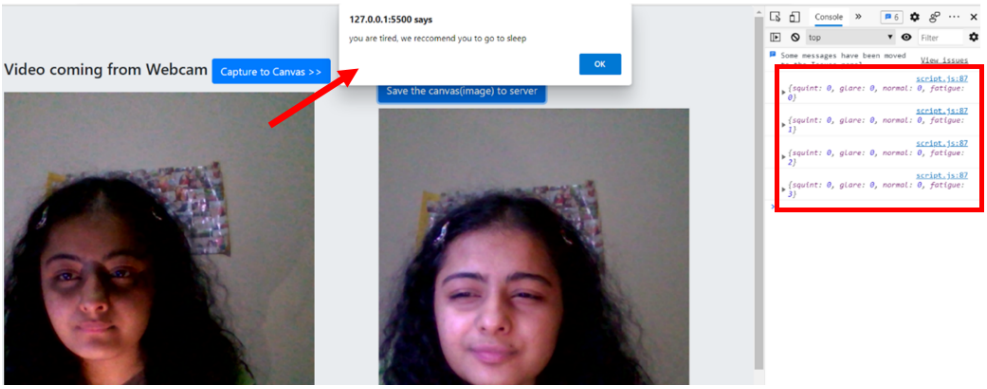


Figure 14. Digital eye strain expression detected, expression logs, and alert issued.

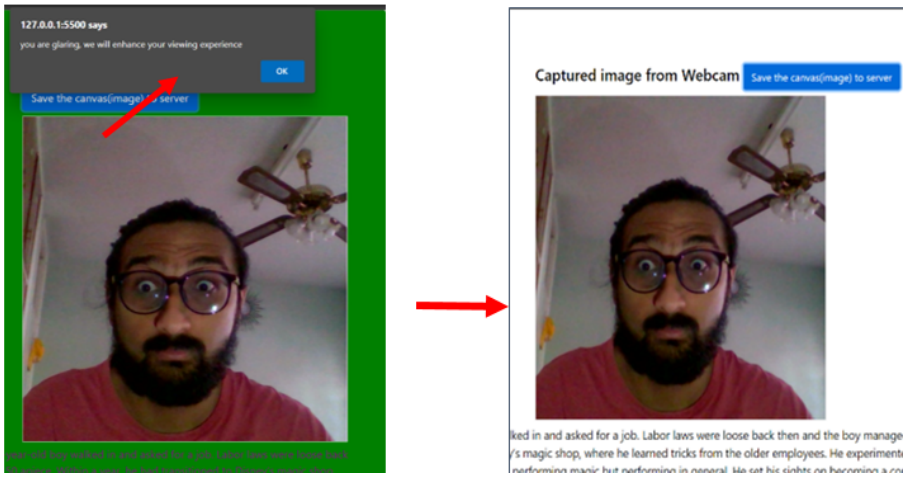


Figure 15. User consent prompt and adaptation.

We displayed the user images and logs for testing purposes, but they run in the background in the live environment. Although the tests only describe simulated lab experiments, they provide insight into the potential for developing self-adaptive applications that relieve digital eye strain. The literature review on digital eye strain highlighted the need for more research in this area to enhance the solution, such as engaging ophthalmology experts. The results of these studies would enrich the solution by providing well-tested threshold values for the self-adaptive application.

5. Conclusions

This research presents the results of a self-adaptive approach to address the challenge of digital eye strain. The study relabeled existing open-set FER datasets and designed an algorithm for digital eye strain expression recognition. We designed a seamless assistive technology model that relieves eye strain using FER techniques and implemented the solution using a self-adaptive approach that supports user-centered validation. Several CNN models were developed, trained, and tested. The study concluded by presenting the evaluation results obtained through laboratory simulations. The results showed that the selected model could classify the expressions with an accuracy of 77%. Although cleaning the dataset to remove confusing images would enhance the results, this study opted to use the entire dataset to mimic real-life expressions that are not always clear-cut. We can enhance the solution by modifying the self-adaptive application with image sampling frequencies and threshold values informed by ophthalmology experts. We envision that future work in this area will scale the solution by detecting and correcting posture, ergonomics, and distance from the screen.

Author Contributions: Conceptualization, L.M.; methodology, K.G.; software, J.G. and L.M.; validation, K.G., and J.G.; formal analysis, J.G. and L.M.; investigation, K.G.; resources, J.G.; data curation, K.G., J.G. and L.M.; writing—original draft preparation, L.M.; writing—review and editing, K.G.; visualization, J.G.; supervision, L.M.; project administration, K.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Institutional Review Board Statement: Ethical review and approval were waived for this study because of the non-intrusive nature of the experiments. Participants were not required to do anything outside their normal activities. However, the researchers were required to seek consent from participants.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Data supporting reported results can be found on the following link: (https://github.com/Jeetg57/VisionAutocorrect, accessed on 10 March 2023) The following publicly archived datasets were analyzed during the study.

1. FER2013 - Goodfellow, I.; Erhan, D.; Carrier, P.; Courville, A.; Mirza, M.; Hamner, B. & Zhou, Y.(2013, November). Challenges in representation learning: A report on three machine learning contests. In Proceedings of the International Conference on Neural Information Processing, pp. 117–124.
2. CK+ - Lucey, P.; Cohn, J.F.; Kanade, T.; Saragih, J.; Ambadar, Z.; Matthews, I. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In Proceedings of the 2010 ieee computer society conference on computer vision and pattern recognition-workshops. IEEE, 2010, pp. 94–101.

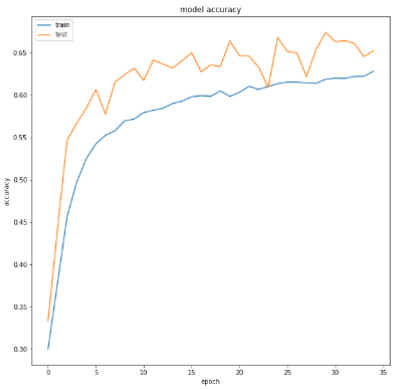
Acknowledgments: The authors acknowledge the United States International University—Africa for the administrative support accorded to the researchers during the research period.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Model details.

	Model Data	Model Accuracy	Confusion Matrix
Model ID	1		
Early Stopping Epochs	35		
Training Accuracy	0.67		
User Evaluation	3		



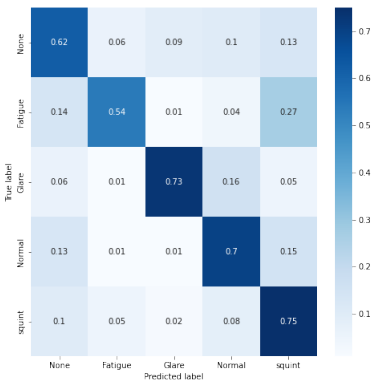


Table A1. *Cont.*

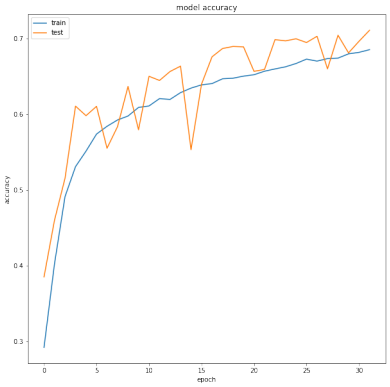
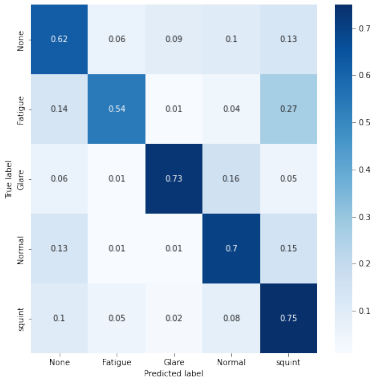
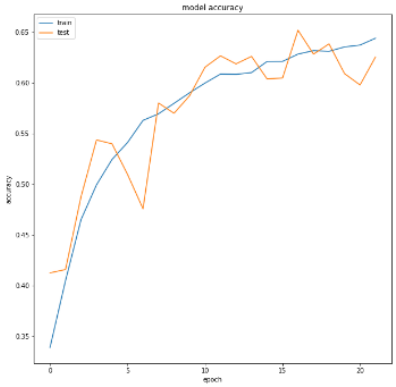
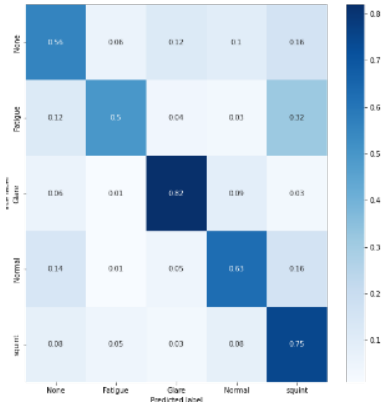
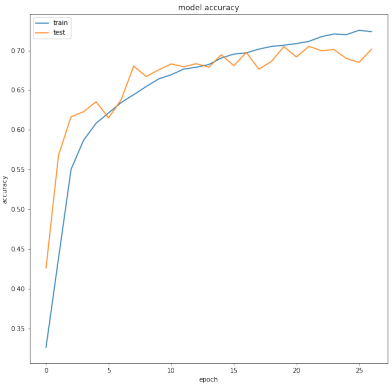
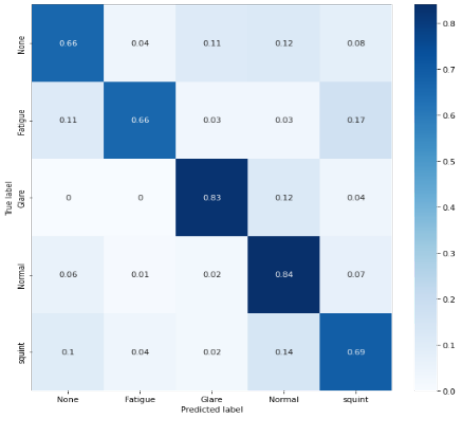
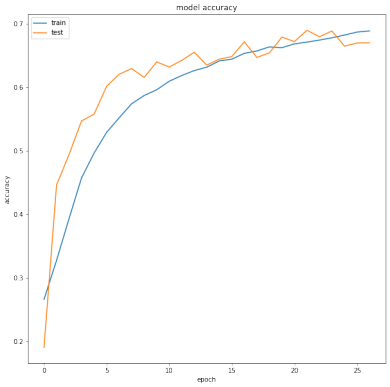
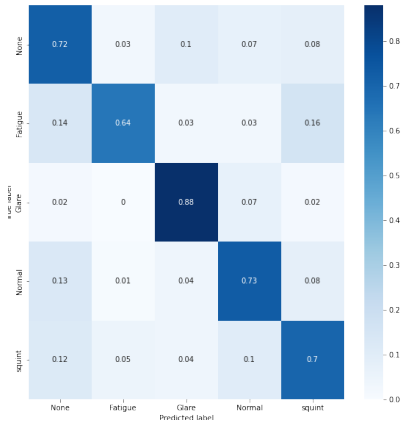
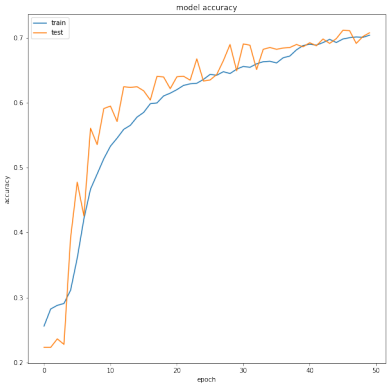
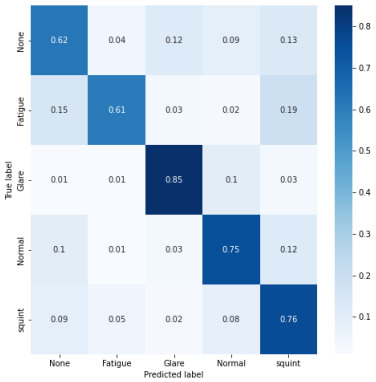
	Model Data	Model Accuracy	Confusion Matrix
Model ID	3		
Early Stopping Epochs	30		
Training Accuracy	0.74		
User Evaluation	2		
Model ID	5		
Early Stopping Epochs	20		
Training Accuracy	0.65		
User Evaluation	6		
Model ID	6		
Early Stopping Epochs	35		
Training Accuracy	0.77		
User Evaluation	1		

Table A1. *Cont.*

	Model Data	Model Accuracy	Confusion Matrix
Model ID	7		
Early Stopping Epochs	20		
Training Accuracy	0.75		
User Evaluation	5		
Model ID	8		
Early Stopping Epochs	35		
Training Accuracy	0.72		
User Evaluation	4		

References

1. Elsworthy, E. Average adult will spend 34 years of their life looking at screens, poll claims. *Independent* **2020**. Available online: https://www.independent.co.uk/life-style/fashion/news/screen-time-average-lifetime-years-phone-laptop-tv-a9508751.html (accessed on 10 March 2023).
2. Nugent, A. UK adults spend 40% of their waking hours in front of a screen. *Independent* **2020**.
3. Bhattacharya, S.; Saleem, S.M.; Singh, A. Digital eye strain in the era of COVID-19 pandemic: An emerging public health threat. *Indian J. Ophthalmol.* **2020**, *68*, 1709. [CrossRef]
4. Siegel, R. Tweens, Teens and Screens: The Average Time Kids Spend Watching Online Videos Has Doubled in 4 Years. *The Washington Post*, 29 October 2019 .
5. Hussain, J.; Ul Hassan, A.; Muhammad Bilal, H.S.; Ali, R.; Afzal, M.; Hussain, S.; Bang, J.; Banos, O.; Lee, S. Model-based adaptive user interface based on context and user experience evaluation. *J. Multimodal User Interfaces* **2018**, *12*, 1–16. [CrossRef]
6. Plos, O.; Buisine, S.; Aoussat, A.; Mantelet, F.; Dumas, C. A Universalist strategy for the design of Assistive Technology. *Int. J. Ind. Ergon.* **2012**, *42*, 533–541. [CrossRef]
7. Firmenich, S.; Garrido, A.; Paternò, F.; Rossi, G. User interface adaptation for accessibility. In *Web Accessibility*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 547–568.
8. Sterritt, R.; Hinchey, M. SPAACE IV: Self-properties for an autonomous & autonomic computing environment—Part IV A Newish Hope. In Proceedings of the 2010 Seventh IEEE International Conference and Workshops on Engineering of Autonomic and Autonomous Systems, Oxford, UK, 22–26 March 2010; pp. 119–125.
9. Sheedy, J.E. The physiology of eyestrain. *J. Mod. Opt.* **2007**, *54*, 1333–1341. [CrossRef]
10. Agarwal, S.; Goel, D.; Sharma, A. Evaluation of the factors which contribute to the ocular complaints in computer users. *J. Clin. Diagn. Res. JCDR* **2013**, *7*, 331. [CrossRef] [PubMed]
11. Sheppard, A.; Wolffsohn, J. Digital eye strain: Prevalence, measurement and amelioration. *BMJ Open Ophthalmol.* **2018**, *3*, e000146. [CrossRef]

12. Rosenfield, M. Computer vision syndrome (aka digital eye strain). *Optom. Pract.* **2016**, *17*, 1–10.
13. Dachapally, P.R. Facial emotion detection using convolutional neural networks and representational autoencoder units. *arXiv* **2017**, arXiv:1706.01509.
14. Joseph, A.; Geetha, P. Facial emotion detection using modified eyemap–mouthmap algorithm on an enhanced image and classification with tensorflow. *Vis. Comput.* **2020**, *36*, 529–539. [CrossRef]
15. Li, K.; Jin, Y.; Akram, M.W.; Han, R.; Chen, J. Facial expression recognition with convolutional neural networks via a new face cropping and rotation strategy. *Vis. Comput.* **2020**, *36*, 391–404. [CrossRef]
16. Huang, Y.; Chen, F.; Lv, S.; Wang, X. Facial expression recognition: A survey. *Symmetry* **2019**, *11*, 1189. [CrossRef]
17. González-Rodríguez, M.R.; Díaz-Fernández, M.C.; Gómez, C.P. Facial-expression recognition: An emergent approach to the measurement of tourist satisfaction through emotions. *Telemat. Inform.* **2020**, *51*, 101404. [CrossRef]
18. Generosi, A.; Ceccacci, S.; Mengoni, M. A deep learning-based system to track and analyze customer behavior in retail store. In Proceedings of the 2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin), Berlin, Germany, 2–5 September 2018; pp. 1–6.
19. Bouzakraoui, M.S.; Sadiq, A.; Enneya, N. Towards a framework for customer emotion detection. In Proceedings of the 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), Agadir, Morocco, 29 November–2 December 2016; pp. 1–6.
20. Baggio, H.C.; Segura, B.; Ibarretxe-Bilbao, N.; Valldeoriola, F.; Marti, M.; Compta, Y.; Tolosa, E.; Junqué, C. Structural correlates of facial emotion recognition deficits in Parkinson's disease patients. *Neuropsychologia* **2012**, *50*, 2121–2128. [CrossRef] [PubMed]
21. Norton, D.; McBain, R.; Holt, D.J.; Ongur, D.; Chen, Y. Association of impaired facial affect recognition with basic facial and visual processing deficits in schizophrenia. *Biol. Psychiatry* **2009**, *65*, 1094–1098. [CrossRef] [PubMed]
22. Khan, S.A.; Hussain, S.; Xiaoming, S.; Yang, S. An effective framework for driver fatigue recognition based on intelligent facial expressions analysis. *IEEE Access* **2018**, *6*, 67459–67468. [CrossRef]
23. Xiao, Z.; Hu, Z.; Geng, L.; Zhang, F.; Wu, J.; Li, Y. Fatigue driving recognition network: Fatigue driving recognition via convolutional neural network and long short-term memory units. *IET Intell. Transp. Syst.* **2019**, *13*, 1410–1416. [CrossRef]
24. Munasinghe, M. Facial expression recognition using facial landmarks and random forest classifier. In Proceedings of the 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), Singapore, 6–8 June 2018; pp. 423–427.
25. Reddy, A.P.C.; Sandilya, B.; Annis Fathima, A. Detection of eye strain through blink rate and sclera area using raspberry-pi. *Imaging Sci. J.* **2019**, *67*, 90–99. [CrossRef]
26. Lim, J.Z.; Mountstephens, J.; Teo, J. Emotion recognition using eye-tracking: Taxonomy, review and current challenges. *Sensors* **2020**, *20*, 2384. [CrossRef]
27. Klaib, A.F.; Alsrehin, N.O.; Melhem, W.Y.; Bashtawi, H.O.; Magableh, A.A. Eye tracking algorithms, techniques, tools, and applications with an emphasis on machine learning and Internet of Things technologies. *Expert Syst. Appl.* **2021**, *166*, 114037. [CrossRef]
28. Li, S.; Deng, W. Deep facial expression recognition: A survey. *IEEE Trans. Affect. Comput.* **2020**, *13*, 1195–1215. [CrossRef]
29. Tian, Y.; Kanade, T.; Cohn, J. *Handbook of Face Recognition*; Li, S.Z., Jain, A.K., Eds.; Springer: London, UK, 2011.
30. Kanade, T.; Cohn, J.F.; Tian, Y. Comprehensive database for facial expression analysis. In Proceedings of the fourth IEEE International Conference on Automatic Face and Gesture Recognition (cat. No. PR00580), Grenoble, France, 28–30 March 2000; pp. 46–53.
31. Lucey, P.; Cohn, J.F.; Kanade, T.; Saragih, J.; Ambadar, Z.; Matthews, I. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops, San Francisco, CA, USA, 13–18 June 2010; pp. 94–101.
32. Goodfellow, I.J.; Erhan, D.; Carrier, P.L.; Courville, A.; Mirza, M.; Hamner, B.; Cukierski, W.; Tang, Y.; Thaler, D.; Lee, D.H.; et al. Challenges in representation learning: A report on three machine learning contests. In Proceedings of the Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Korea, 3–7 November 2013; Proceedings, Part III 20 ; Springer: Berlin/Heidelberg, Germany, 2013; pp. 117–124.
33. Petrou, M.M.; Petrou, C. *Image Processing: The Fundamentals*; John Wiley & Sons: Hoboken, NJ, USA, 2010.
34. Russ, J.C. *The Image Processing Handbook*; CRC Press: Boca Raton, FL, USA, 2006.
35. Joshi, P. *OpenCV with Python by Example*; Packt Publishing Ltd.: Birmingham, UK, 2015.
36. Van der Walt, S.; Schönberger, J.L.; Nunez-Iglesias, J.; Boulogne, F.; Warner, J.D.; Yager, N.; Gouillart, E.; Yu, T. scikit-image: Image processing in Python. *PeerJ* **2014**, *2*, e453. [CrossRef]
37. Liliana, D.Y.; Basaruddin, T. Review of automatic emotion recognition through facial expression analysis. In Proceedings of the 2018 International Conference on Electrical Engineering and Computer Science (ICECOS), Pangkal, Indonesia, 2–4 October 2018; pp. 231–236.
38. Lopes, A.T.; De Aguiar, E.; De Souza, A.F.; Oliveira-Santos, T. Facial expression recognition with convolutional neural networks: coping with few data and the training sample order. *Pattern Recognit.* **2017**, *61*, 610–628. [CrossRef]
39. Georgescu, M.I.; Ionescu, R.T.; Popescu, M. Local learning with deep and handcrafted features for facial expression recognition. *IEEE Access* **2019**, *7*, 64827–64836. [CrossRef]
40. Pecoraro, R.; Basile, V.; Bono, V. Local multi-head channel self-attention for facial expression recognition. *Information* **2022**, *13*, 419. [CrossRef]

41. Aouayeb, M.; Hamidouche, W.; Soladie, C.; Kpalma, K.; Seguier, R. Learning vision transformer with squeeze and excitation for facial expression recognition. *arXiv* **2021**, arXiv:2107.03107.
42. Meng, D.; Peng, X.; Wang, K.; Qiao, Y. Frame attention networks for facial expression recognition in videos. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 3866–3870.
43. Pourmirzaei, M.; Montazer, G.A.; Esmaili, F. Using self-supervised auxiliary tasks to improve fine-grained facial representation. *arXiv* **2021**, arXiv:2105.06421.
44. Gacav, C.; Benligiray, B.; Topal, C. Greedy search for descriptive spatial face features. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 1497–1501.
45. Antoniadis, P.; Filntisis, P.P.; Maragos, P. Exploiting Emotional Dependencies with Graph Convolutional Networks for Facial Expression Recognition. In Proceedings of the 2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021), Jodhpur, India, 15–18 December 2021. Available online: http://xxx.lanl.gov/abs/2106.03487 (accessed on 12 January 2023).
46. Ryumina, E.; Dresvyanskiy, D.; Karpov, A. In search of a robust facial expressions recognition model: A large-scale visual cross-corpus study. *Neurocomputing* **2022**, *514*, 435–450. [CrossRef]
47. Savchenko, A.V.; Savchenko, L.V.; Makarov, I. Classifying emotions and engagement in online learning based on a single facial expression recognition neural network. *IEEE Trans. Affect. Comput.* **2022**, *13*, 2132–2143. [CrossRef]
48. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 15 June 2019 ; pp. 6105–6114.
49. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
50. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 6–11 July 2015; pp. 448–456.
51. Minhas, R.A.; Javed, A.; Irtaza, A.; Mahmood, M.T.; Joo, Y.B. Shot classification of field sports videos using AlexNet Convolutional Neural Network. *Appl. Sci.* **2019**, *9*, 483. [CrossRef]
52. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
53. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
54. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
55. Liu, S.; Deng, W. Very deep convolutional neural network based image classification using small training sample size. In Proceedings of the 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, Malaysia, 3–6 November 2015; pp. 730–734.
56. Chen, X.; Xie, L.; Wu, J.; Tian, Q. Progressive darts: Bridging the optimization gap for nas in the wild. *Int. J. Comput. Vis.* **2021**, *129*, 638–655. [CrossRef]
57. Al-Sabaawi, A.; Ibrahim, H.M.; Arkah, Z.M.; Al-Amidie, M.; Alzubaidi, L. Amended convolutional neural network with global average pooling for image classification. In *Intelligent Systems Design and Applications. ISDA 2020. Advances in Intelligent Systems and Computing*; Abraham, A., Piuri, V., Gandhi, N., Siarry, P., Kaklauskas, A., Madureira, A., Eds.; Springer: Cham, Switzerland, 2020; Volume 1351, pp. 171–180.
58. Liu, D.; Liu, Y.; Dong, L. G-ResNet: Improved ResNet for brain tumor classification. In *Neural Information Processing. ICONIP 2019. Lecture Notes in Computer Science*; Gedeon, T., Wong, K., Lee, M., Eds.; Springer: Cham, Switzerland, 2019; Volume 11953, pp. 535–545.
59. Wang, S.; Peng, G.; Zheng, Z.; Xu, Z. Capturing emotion distribution for multimedia emotion tagging. *IEEE Trans. Affect. Comput.* **2019**, *12*, 821–831. [CrossRef]
60. Viola, P.; Jones, M. 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, Kauai, HI, USA, 8–14 December 2001; Volume 1, p. I.
61. Vijayalakshmi, K.A.A. Comparison of viola-jones and kanade-lucas-tomasi face detection algorithms. *Orient. J. Comput. Sci. Technol.* **2017**, *10*, 151–159 .