Eye-Rubbing Detection Tool Using Artificial Intelligence on a Smartwatch in the Management of Keratoconus

Ines Drira, Ayoub Louja, Layth Sliman, Vincent Soler, Maha Noor, Abdellah Jamali, Pierre Fournie

tvst

Cornea & External Disease

Eye-Rubbing Detection Tool Using Artificial Intelligence on a Smartwatch in the Management of Keratoconus

Ines Drira[1,2], Ayoub Louja[3], Layth Sliman[4], Vincent Soler[1], Maha Noor[5], Abdellah Jamali[3], and Pierre Fournie[1]

[1] Department of Ophthalmology, Centre Hospitalier Universitaire de Toulouse, Toulouse, France
[2] Department of Ophthalmology, Centre Hospitalier Intercommunal de Créteil, Créteil, France
[3] Hassan First University, Faculty of Science and Technology, IR2M Laboratory, Settat, Morocco
[4] EFREI Paris, Paris, France
[5] Manchester Royal Eye Hospital, Manchester, UK

Correspondence: Ines Drira, Pierre-Paul Riquet Hospital, Avenue du Professeur Jean Dausset, Toulouse 31300, France. e-mail: ines.drira@chicreteil.fr

Received: May 16, 2023
Accepted: October 27, 2024
Published: December 12, 2024

Keywords: keratoconus; ectasia; eye rubbing; artificial intelligence; deep learning

Citation: Drira I, Louja A, Sliman L, Soler V, Noor M, Jamali A, Fournie P. Eye-rubbing detection tool using artificial intelligence on a smartwatch in the management of keratoconus. Transl Vis Sci Technol. 2024;13(12):16, https://doi.org/10.1167/tvst.13.12.16

Purpose: Eye rubbing is considered to play a significant role in the progression of keratoconus and of corneal ectasia following refractive surgery. To our knowledge, no tool performs an objective quantitative evaluation of eye rubbing using a device that is familiar to typical patients. We introduce here an innovative solution for objectively quantifying and preventing eye rubbing. It consists of an application that uses a deep-learning artificial intelligence (AI) algorithm deployed on a smartwatch.

Methods: A Samsung Galaxy Watch 4 smartwatch collected motion data from eye rubbing and everyday activities, including readings from the gyroscope, accelerometer, and linear acceleration sensors. The training of the model was carried out using two deep-learning algorithms, long short-term memory (LSTM) and gated recurrent unit (GRU), as well as four machine learning algorithms: random forest, K-nearest neighbors (KNN), support vector machine (SVM), and XGBoost.

Results: The model achieved an accuracy of 94%. The developed application could recognize, count, and display the number of eye rubbings carried out. The GRU model and XGBoost algorithm also showed promising performance.

Conclusions: Automated detection of eye rubbing by deep-learning AI has been proven to be feasible. This approach could radically improve the management of patients with keratoconus and those undergoing refractive surgery. It could detect and quantify eye rubbing and help to reduce it by sending alerts directly to the patient.

Translational Relevance: This proof of concept could confirm one of the most prominent paradigms in keratoconus management, the role of abnormal eye rubbing, while providing the means to challenge or even negate it by offering the first automated and objective tool for detecting eye rubbing.

Introduction

Keratoconus is a progressive degenerative eye disease affecting the cornea, with an estimated prevalence of 1.38/1000.[1] At the anatomical level, it results in progressive thinning and deformation of the cornea. The primary clinical manifestation is an irreversible deterioration in visual acuity that does not improve with optical systems or corrective measures. The exact pathophysiology of the disease remains poorly understood but probably involves genetic predisposition,[2,3] potentially combined with environmental factors such as chronic corneal mechanical trauma (e.g., eye rubbing, eye compression at night).[4,5]

Initially described by Ridley[6] in 1961, the role of eye rubbing in the development and progression of keratoconus has since been established by numerous studies.[7–9] Some authors have described eye rubbing in keratoconus patients as abnormal,[10] not only because of a higher frequency but also because of its intensity, duration, and repetitiveness. In addition to the progression of keratoconus, this abnormal rubbing could also lead to corneal ectasia following refractive surgery,

which millions of patients undergo. This complication can occur even with no preoperative risk factors.[11] The pathophysiological mechanism would be the same as in keratoconus: repeated rubbing distorting a cornea thinned and weakened by surgery.

Cessation of eye rubbing[4] requires, first, the treatment of any underlying predisposing circumstance (e.g., allergic disease, dry eye syndrome) followed by therapeutic education.[12] The need for objective quantification of eye rubbing makes it difficult to evaluate from a methodological point of view. If the disease worsens despite eye rubbing management, corneal cross-linking[13] will be the only treatment that slows down the change.

Eye rubbing is currently subjectively quantified by questioning or using standardized questionnaires. Given the above, we wanted to explore a new approach to the care of these patients that would lead to more efficient management of these diseases, as well as an ability to quantify the impact of eye rubbing on the development of these pathologies.

Several studies in the literature have employed artificial intelligence (AI) techniques to improve the diagnosis of keratoconus,[14] but they mainly focus on classification algorithms using various types of precollected eye data. The most relevant study on eye rubbing was presented by Nokas et al.[15] However, that study employed a specially designed, wired Raspberry Pi device and collected data only from the gyroscope and accelerometer sensors. This approach requires a unique setting and cannot easily be used by patients. To our knowledge, no published work exists on using motion recognition on a smartwatch for eye-rubbing detection. In addition to the objective detection of eye rubbing using different movements, our method provides the convenience of allowing typical patients to use a familiar device which does not impair their mobility.

We developed a software application and employed a deep learning (DL) AI algorithm deployed on a smartwatch to detect, quantify, and display the frequency of eye rubbing. We aimed to demonstrate a novel solution for quantifying and preventing eye rubbing in patients with keratoconus or with corneal ectasia following refractive surgery.

Materials and Methods

Study Design

We conducted a proof-of-concept study of a new software method to establish the feasibility and rationale for using a DL AI deployed on a smartwatch to detect and quantify eye rubbing. One investigator (AL) and one other participant were included as study participants. They consented to participate and provided the data for the neural network training. The study was performed in accordance with the tenets of the Declaration of Helsinki.

General Concepts

Smartwatches have three sensors that are of interest for rubbing detection: a gyroscope, an accelerometer, and a linear acceleration sensor. The application we developed is for watches with the Android operating system Wear OS. We relied on the app development kit provided by Google according to best practice recommendations. A Galaxy Watch4 (Samsung, Suwon, South Korea) was used to test our application. The development of user interface mockups was carried out in Android Studio.

Data Collection

An initial data collection application has been developed. The data of interest within the scope of our objective include the date, axes of the accelerometer (x, y, z), gravity (x, y, z), linear acceleration data *Ma* (Mag accelero) and *Mg* (Mag gyro), and the gyroscope axes *Gx*, *Gy*, and *Gz*. The accelerometer measures changes in the speed and position of the watch along three axes and, therefore, tracks translational movements. The gyroscope detects rotational movements such as pitching and rolling. The continuous data recorded were segmented into 1-minute periods at a frequency of 20 to 30 samples of information per second and exported to a file in standardized comma-separated values (.CSV) format.

For this proof of concept, we identified and processed three categories of rubbing: first, with the palms or heels of the hands, which we refer to as type A; second, with the pulp of the index fingers, which we refer to as type B; and, third, with the phalanges of the index fingers, which we refer to as type C (Fig. 1). Data were collected by mimicking these eye rubs during various activities and positions: standing, sitting, walking, and climbing or descending stairs. Those movements were made in front of but without any contact with the eye so as to prevent adverse effects on the cornea. A fourth category included all other movements, except eye rubbing, so as to take into account the movements of everyday life. The data distribution is shown in Figure 2.

To access the motion data generated by the sensors, the framework set up in Android uses a number of different categories: Sensor (used to record monitor-

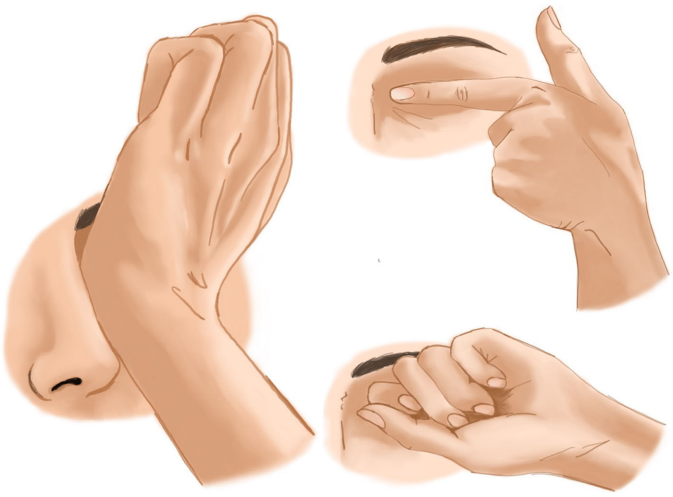

Figure 1. Illustrated examples of the abnormal eye rubbing used to train the algorithm: with the palms or heels of the hands (type A), with the pulp of the index fingers (type B), and with the phalanges of the index fingers (type C).

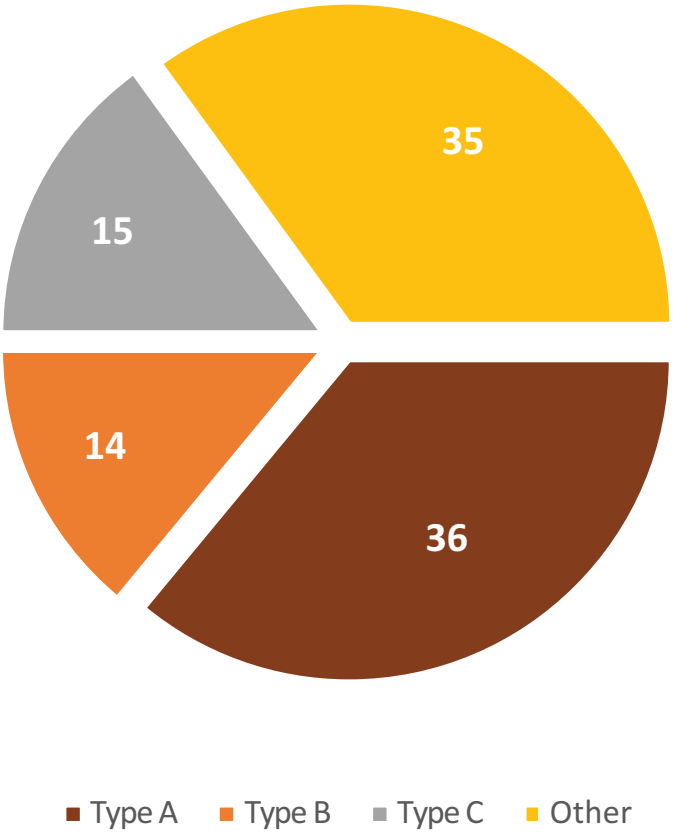

Figure 2. Distribution of the different types of movements (in percent): with the palms or heels of the hands (type A), with the pulp of the index fingers (type B), and with the phalanges of the index fingers (type C).

ing of the sensors), SensorManager (used to instantiate the Sensor class), and SensorEvent (containing the new values available from the sensors). All collected motion-recognition data were used to create a Sensor Motion Recognition Dataset (Figs. 3, 4), which was then used to train a neural network that we developed for this purpose. The collected data showed a sufficient repetition of the types of movements to be classified, despite the variations introduced by the diversity and variety of recorded activities. This indicated that the amount of raw data was large enough to capture the necessary patterns for model training.

Data Cleaning and Preprocessing

To ensure the integrity and reliability of our data, we implemented a comprehensive series of data-cleaning techniques to address duplicate entries, signal variation normalization, noise, missing values, imputation, exclusion, and feature extraction and selection . To mitigate these issues, we applied Kalman filtering, a widely used data filtering method known for its effectiveness in removing noise. Duplicate entries were removed to maintain the uniqueness of each record, using the drop_duplicates() function in Pandas to identify and eliminate exact duplicates across all columns. Missing data were addressed by employing imputation methods, where missing values were filled using the mean of the corresponding columns. For columns with a significant number of missing entries, such as accelerometer (x, y, z), interpolation techniques were used. Records with more than 20% missing data were excluded to avoid compromising the analysis. Relevant features were then extracted by calculating various statistical measures, providing deeper insights into the data.

We collected data covering the different types of eye rubbing (types A, B, and C) and various daily life movements to ensure a comprehensive representation of the target behaviors. Before cleaning, the collected data showed a sufficient repetition of the types of movements to be classified, despite the variations introduced by the diversity and variety of recorded activities. This indicated that the amount of raw data was large enough to capture the necessary patterns for model training. The cleaned dataset represented a total of 12 MB, amounting to about 70 minutes of recording.

Exploratory Data Analysis

Data collected on smartwatches produce a large, complex set of information. To help visualize this infor-

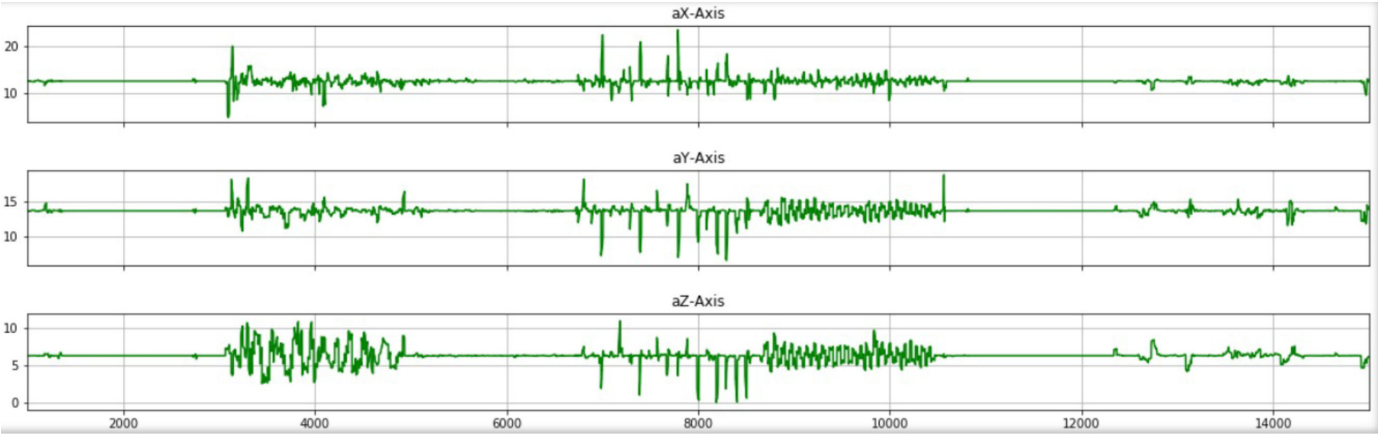

Figure 3. Graphic representation of some data obtained by simulating eye rubbing with the palm or heel of the hand.

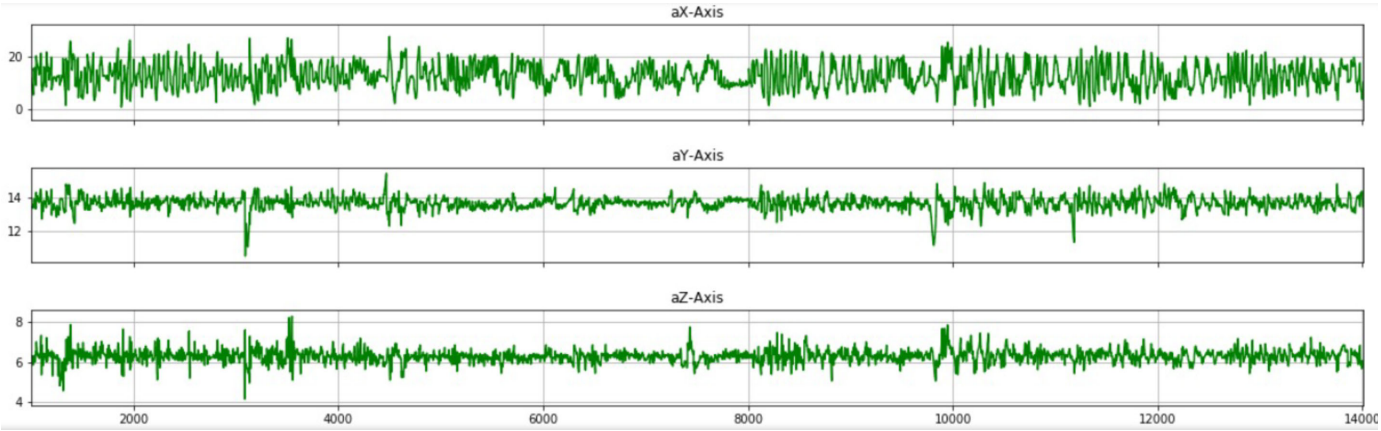

Figure 4. Graphic representation of some data obtained for non–eye-rubbing movements.

mation, we used a *t*-distributed stochastic neighbor embedding (t-SNE) algorithm (Fig. 5). t-SNE allows dimensional reduction of the data while preserving the relationships between samples. Changing the value for the perplexity parameter allows for representing data by focusing more on the details or on the overall structure.

Data Concatenation

We used Python OS and Panda libraries to merge the .CSV data record files before transferring them to a DataFrame data type (two-dimensional storage format) suitable for our neural network inputs.

Neural Network

A five-layer, sequential, long short-term memory (LSTM) neural network was developed (Fig. 6). The LSTM network is part of a recurrent neural network that analyzes data sequences of different lengths. This type of network has the advantage of having a high capacity to receive, as input, a set of continuous values in the form of a two-dimensional matrix. In contrast, most other existing neural networks require discontinuous or discrete values. Because our eye-rubbing data consisted of temporal sequences, LSTM appeared to be the most relevant.

The model was implemented in Python using the Keras and TensorFlow libraries. The different layers are parameterized to avoid overfitting by batch normalization and dropout techniques. Overfitting is a perfect but biased adaptation of the results of the model to the training and validation dataset, not transferable to another dataset. Batch normalization[16] consists of formatting the input data of each neural network layer to obtain a 0 mean and a standard deviation of 1, making it possible to obtain more stable data belonging to the same scale. Dropout[17] works by randomly deacti-

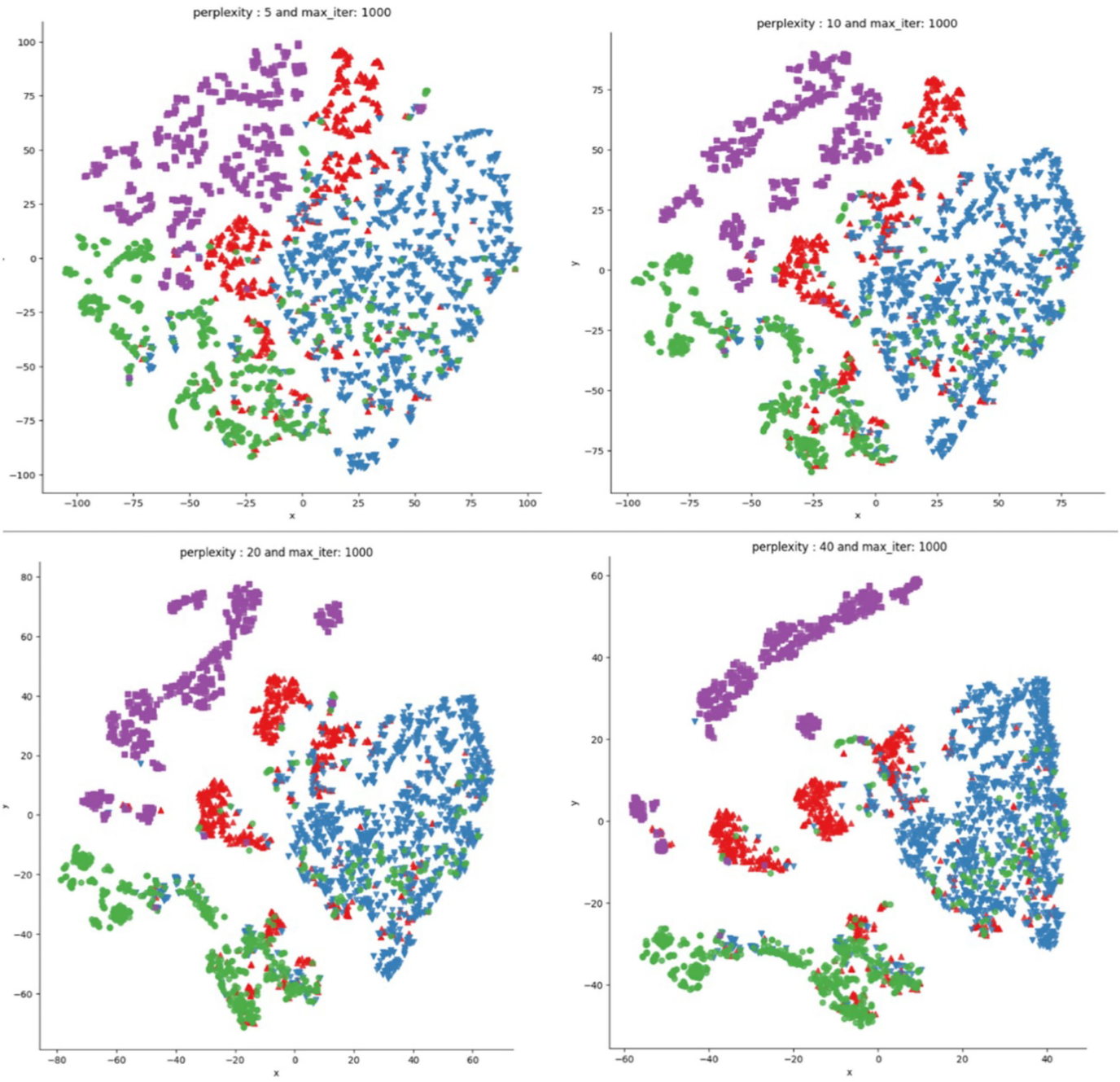

Figure 5. t-SNE representations with different perplexity values (type A, *purple squares*; type B, *red triangle*; type C, *green circle*; other, *blue reversed triangle*).

vating neurons during network training to exploit each neuron individually.

For each type of sensor (e.g., gyroscope, accelerometer), the model receives as input 112 types of data: accelerometer (x, y, z), gyroscope (x, y, z), gravity (x, y, z), and linear acceleration. These data types have undergone preprocessing to extract meaningful features by calculating statistical measures such as mean, standard deviation, median absolute deviation, maximum, minimum, energy, entropy, interquartile range, and correlation for each feature. This process resulted in a total of 112 features.

The dataset was then divided into 80% for the training dataset and 20% for the testing dataset and then divided for training into a series of 120 records using the Python code class generate_sequence. The interval and number of different characteristics or features

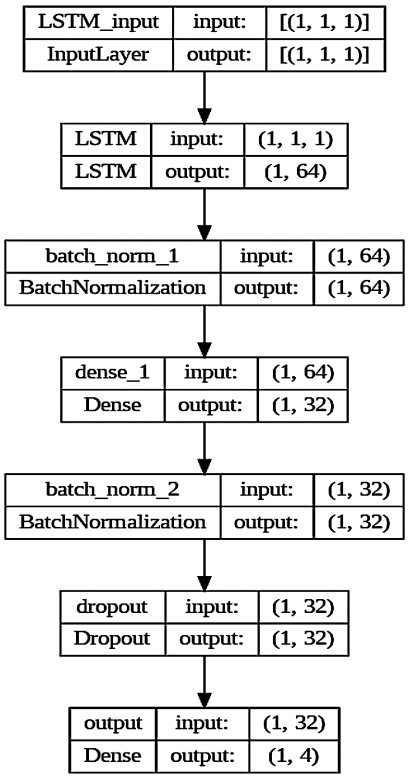

Figure 6. Structure of the LSTM neural network.

(columns) were defined, passed to the class, and stored in a variable. This provided an output table of four values corresponding to the percentage prediction for each class.

Training

The training was conducted over 20 minutes on a GeForce GTX 1080 Ti graphics card (NVIDIA, Santa Clara, CA). It was configured to run on 300 epochs (i.e., iterations, in machine learning language) with an early stopping method and a batch size of 16. We chose 300 epochs as a high value to overcome and detect late convergence cases, if any. We decided on a batch size of 32 for memory reasons. Additionally, a small batch size allows better stability during training with faster convergence and better generalization. The adaptive momentum estimation (ADAM) optimizer[18] was used, with a decreasing learning rate. The ADAM optimizer uses the principles of momentum, a gradient descent algorithm in which a learning step depends on the derivative of the current step and the steps that immediately preceded it to prevent the learning from getting stuck at a local minimum. Thus, as long as the descent gradient is in the same direction as the previous ones, the gradient descent velocity will be accelerated.

The early stopping method was used to avoid overfitting. In our case, we showed convergence of the results between 100 and 120 epochs. The stochastic gradient descent allowed us to find the minimum of a convex function by gradually converging toward it. Gradient descent was used to minimize the cost function, which is a convex function. The cost function, or least squares of errors, method was used to measure the performance of the model by measuring the error between the model and the dataset. We also used a

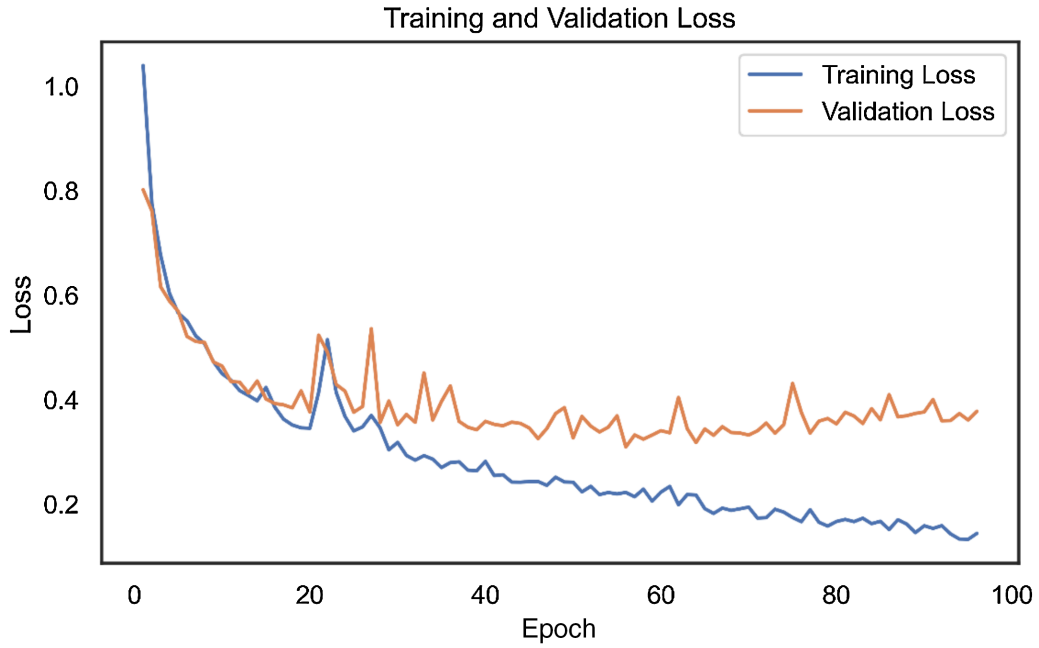

Figure 7. Loss function of the LSTM neural network.

Table 1. Comparison of Fine-Tuned and Best Parameters for the Different Models

Model	Fine-Tuned Parameters	Best Parameters
SVM	C: [0.125, 0.5, 1, 2, 8, 16]	C = 2
	Degree: [1, 2, 3, 4, 5]	Degree = 3
KNN	N neighbors: [5, 7, 9, 11, 13, 17]	N neighbors = 11
	Weights: ['uniform', 'distance']	Weights = 'distance'
	Algorithm: ['kd tree', 'brute', 'ball tree']	Algorithm = 'kd tree'
XGBoost	N estimators: [100, 150, 200]	N estimators = 150
	Max depth: [3, 5, 7, 9]	Max depth = 5
	Learning rate: [0.1, 0.2, 0.3]	Learning rate = 0.1
Random forest	N estimators: [100, 150, 200]	N estimators = 200
	Max depth: [none, 5, 10]	Max depth = 10
	Min samples split: [2, 5, 10]	Min samples split = 2

decreasing learning rate (hyperparameter affecting the speed of gradient descent) to refine the training as it progressed, to save training time, and to avoid undesirable divergent behaviors of the loss function. Finally, the loss function evaluated the extent to which a specific algorithm modeled the data (Fig. 7).

Comparison With Other Machine Learning Approaches

To compare our LSTM model with another neural network, we developed and trained a gated recurrent unit (GRU) neural network. To compare our results with the existing literature,[15] we also developed support vector machine (SVM), K-nearest neighbor (KNN), random forest classifier, and XGBoost algorithms using the same dataset.

Hyperparameter Tuning

For each of the machine learning models (random forest, XGBoost, KNN, and SVM), we conducted a thorough hyperparameter tuning process using a fivefold cross-validation technique. Specifically, we utilized GridSearchCV to systematically explore a predefined range of hyperparameters for each model, identifying the combination that yielded the highest performance (Table 1). For the DL algorithms, including the LSTM and GRU models, we adopted a robust approach that involved fine-tuning key hyperparameters, applying dropout, and utilizing early stopping. The combination of these strategies ensured that the model could learn complex temporal patterns in the data while maintaining the ability to generalize well to unseen data. To ensure that the LSTM model, along with other DL algorithms, converged effectively during the training phase, we chose a sufficiently high number of epochs to allow the model architecture ample opportunity to learn.

Results

Model Accuracy and Other Evaluation Metrics

We used k-fold cross-validation to evaluate the machine learning models (SVM, KNN, XGBoost, random forest) where we created $k = 5$ folds for each training set. To ensure a comprehensive assessment of the ability of the model to generalize, the training and test sets included recordings from distinct subjects. The datasets were partitioned 274 times into an 80% train and 20% test split, with each partition initialized through a distinct random shuffle. In our classification task, we chose categorical cross-entropy loss function and evaluated the performance of the model using the categorical accuracy metric, which gives a clear view of performance and is highly appropriate for multiclass classification. Beyond categorical accuracy, we used a range of metrics designed for multiclass classification to gain a more comprehensive insight into the performance of our model, including precision, recall, and F1 score.

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

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 Score} = \frac{2TP}{2TP + FP + FN}$$

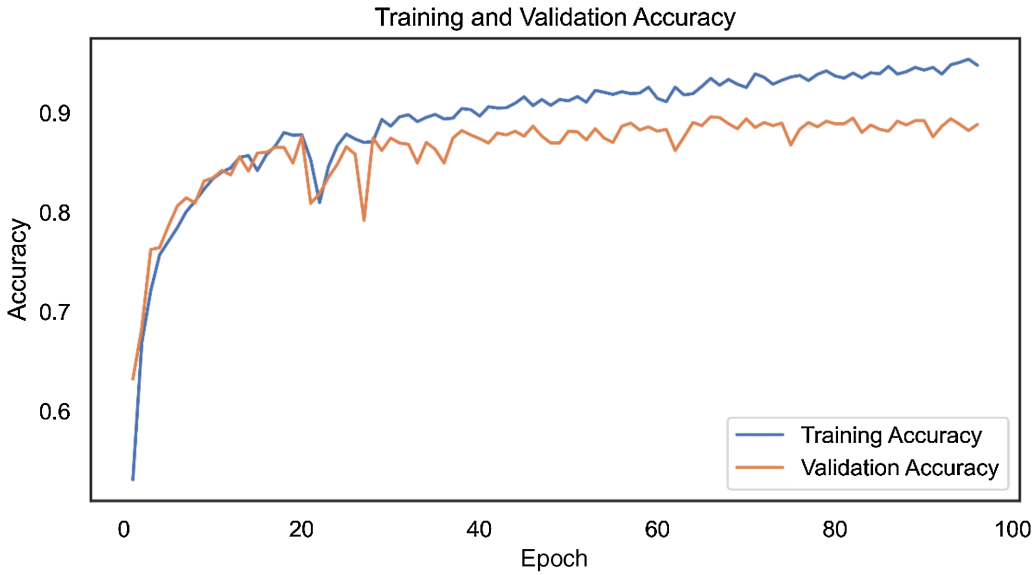

Figure 8. Model accuracy of the LSTM neural network.

Table 2. Performance Metrics of Neural Networks and Machine Learning Algorithms

			Confidence Interval (95%)	
Algorithm	Accuracy	Average Precision	Lower Bound	Upper Bound
LSTM	0.943	0.892	0.880	0.905
GRU	0.921	0.871	0.860	0.882
SVM	0.888	0.882	0.868	0.895
KNN	0.903	0.900	0.888	0.913
XGBoost	0.931	0.931	0.925	0.937
Random forest	0.933	0.928	0.917	0.938

where *TP* corresponds to true positives, *TN* to true negatives, *FP* to false positives, and *FN* to false negatives.

The accuracy obtained for the LSTM was 0.94 (Fig. 8), with an average precision of 0.89. We obtained an accuracy of 0.92 for the GRU with an average precision of 0.87. For the machine learning algorithms, we obtained an accuracy of 0.89 for the SVM, 0.90 for KNN, 0.93 for XGBoost, and 0.93 for random forest. Confidence intervals are shown in Table 2 and Figures 9 and 10. As each machine learning algorithm was trained using fivefold cross-validation, involving multiple rounds of training and testing, we present the best classification precision, recall, and F1 scores for each class in our results in Table 3.

Positive and Negative Predictive Values

The positive predictive values (PPVs) and negative predictive values (NPVs) were calculated for the different categories. The PPVs for types A, B, C, and other were, respectively, 0.989, 0.904, 0.855, and 0.876. The NPVs were, respectively, 0.992, 0.965, 0.959, and 0.939.

Smartwatch Application

The application developed as part of this study makes it possible to recognize, count, and display the number of eye rubbings carried out. The resulting algorithm is compatible with the Android platform by saving the model graph in the chkp format and then converting it to a file in pb format using the Python function Freeze_graph. The access codes for permission to use the necessary sensors (accelerometer and gyroscope) were obtained through the AndroidManifest file.xml. The TensorFlow framework was used to develop the application with the help of the Gradle build automation tool (build.gradle). The TensorFlow_inference library is required to load the

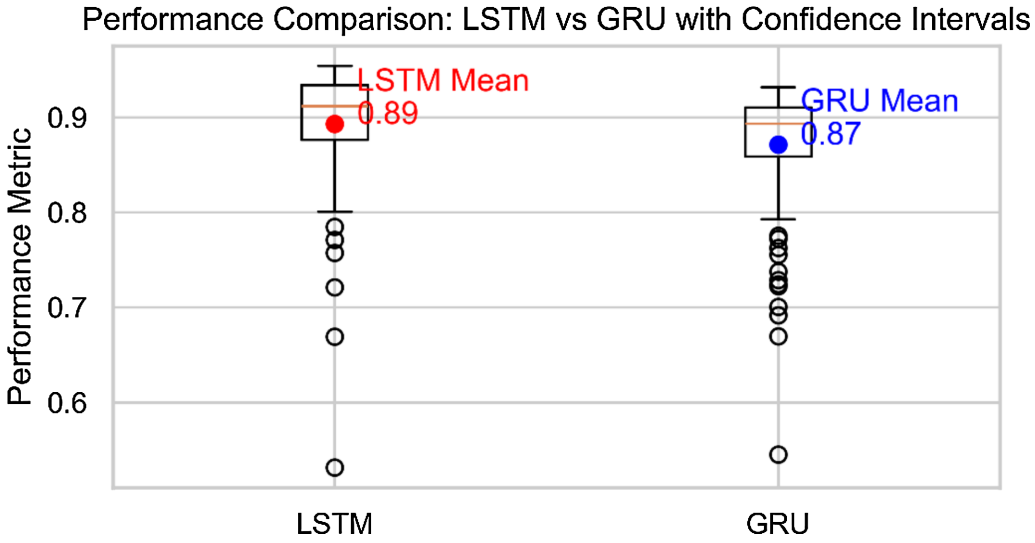

Figure 9. Precision comparison for neural networks.

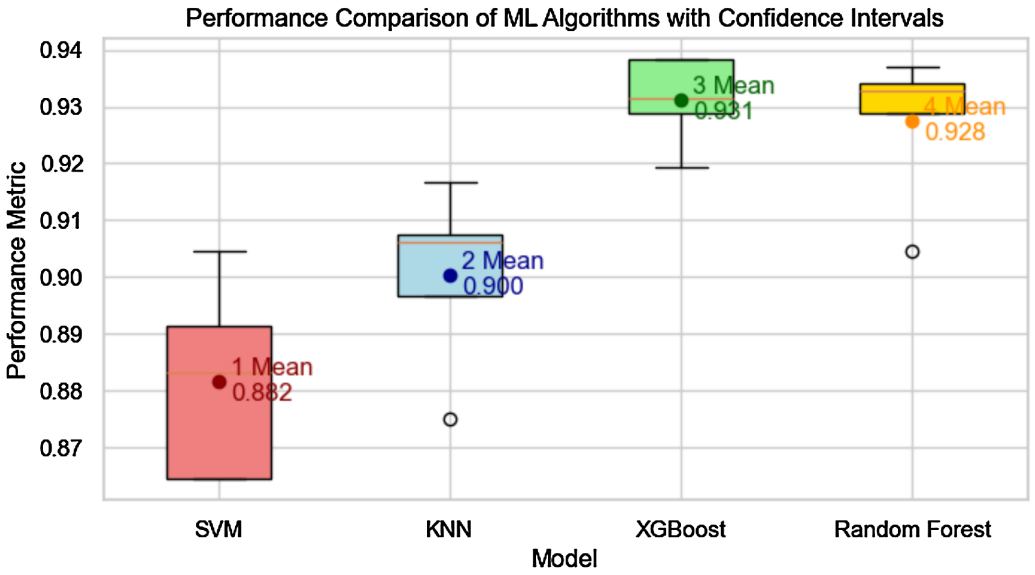

Figure 10. Precision comparison for machine learning algorithms.

model so as to make model predictions in the Android app.

Discussion

The results seem promising and could significantly change the management of patients with keratoconus. To our knowledge, this is the first publication of an automated eye-rubbing detection solution using a smartwatch. We obtained good results with an accuracy of nearly 94%. The confusion matrix (Fig. 11) shows that most confusion in classifying movements is between different types of eye rubbing, not with non–eye-rubbing movements. This is undoubtedly due to the similarities between those eye-rubbing movements. This tool will allow ophthalmologists to evaluate eye rubbing objectively and thus consider this information in therapeutic decision making. It could also be used as an educational tool for patients, giving them insight into the reality and frequency of their eye-rubbing behaviors.

As described in the introduction, eye rubbing is thought to be the leading cause of keratoconus development. However, this supposition is based on obser-

Table 3. Detailed Performance Metrics of Neural Networks and Machine Learning Algorithms for Each Category of Movements

Model	Category	Precision	Recall	F1 Score
LSTM	Other	0.89	0.90	0.89
	Type A	0.99	0.98	0.98
	Type B	0.87	0.87	0.87
	Type C	0.85	0.84	0.84
GRU	Other	0.83	0.90	0.86
	Type A	0.98	0.97	0.97
	Type B	0.86	0.80	0.83
	Type C	0.80	0.73	0.77
SVM	Other	0.87	0.91	0.89
	Type A	0.99	0.99	0.99
	Type B	0.93	0.79	0.86
	Type C	0.79	0.84	0.81
KNN	Other	0.88	0.94	0.91
	Type A	1.00	0.99	1.00
	Type B	0.90	0.85	0.87
	Type C	0.85	0.79	0.82
XGBoost	Other	0.92	0.95	0.94
	Type A	1.00	0.97	0.99
	Type B	0.93	0.91	0.92
	Type C	0.89	0.86	0.87
Random forest	Other	0.91	0.95	0.93
	Type A	1.00	0.98	0.99
	Type B	0.94	0.91	0.92
	Type C	0.90	0.85	0.88

vations of patients' behavior without sustainable proof. This detection tool will provide objective data to confirm or refute this significant hypothesis. This will impact the quality of care given to patients with keratoconus and patients undergoing or considering refractive surgery.[24] Additionally, it may be used in patients under care for allergic conjunctivitis, so as to evaluate therapeutic efficacy.

Our work is a proof of concept, to be continued. We need to develop a more proven and acceptable application from a medical point of view. The main limitation is the small number of participants used to create the dataset, which limits the external validity of the algorithm. The next step is to obtain a real-life dataset from a more important number of keratoconus patients and to validate the algorithm on an external dataset.

Other potentially relevant neural networks are transformers that can be combined with time-series models. They can model long-distance dependencies and capture complex structures in sequential data. The plan is to employ these models in a further study with more participants and data, although those models have been designed for language analysis. Other objectives will be, first, to adapt the algorithm to each patient (detection of the type of eye rubbing performed); second, to be able to send alerts to the patient to stop the rubbing; and, finally, to be able to communicate the results to the medical software used by the ophthalmologist in charge of the patient.

Regarding the adaptation of the algorithm to the patient, the ideal would be to produce an application to be configured at the beginning of use. The first option would be to ask the patient to rub their eyes and record movement, limiting the amount of data shared between patients for training, which could be a barrier to acceptability and deployment from an ethical point of view. The second option would be to request confirmation during an initial period for each rubbing detected, so as to improve the performance of the model by reinforcement learning (another type of real-time machine learning model) or by using a transformer architecture.[19]

One limitation of this tool is the choice of watch lateralization: On which wrist does the patient wear their smartwatch? We have considered various ways of dealing with this problem. For patients with unilateral or asymmetric keratoconus, one can equip the same side as the eye mainly affected. For patients with symmetrical keratoconus, either we could equip one wrist (the one on which the patient would usually choose to wear their watch) or we could add a smart bracelet on the other wrist, or even on both wrists, to make it less bulky. But, that means buying a device just for this use. Using the watch on the side of the most severely affected eye may be sufficient.

In addition to detecting and counting those movements, this tool could help stop eye rubbing. Alerts should be sent to the patient promptly upon proven detection of repeated rubbing; they could be sent as a vibration or as a sound. We could also consider a system of notifications specifying the number of eye rubs each day, with a "serious game"[20] system for younger patients, to encourage a reduction in eye rubbing. Previous studies have assessed the acceptability of such monitoring devices and reported good acceptance from patients and caregivers[21,22] and good compliance,[23] but they will have to be assessed specifically for the keratoconus population, which is younger than those usually targeted by this kind of device.

The communication of results must be done securely. Direct integration of data into business software is possible due to the standardization of data formats. Relevant data could thus be integrated into

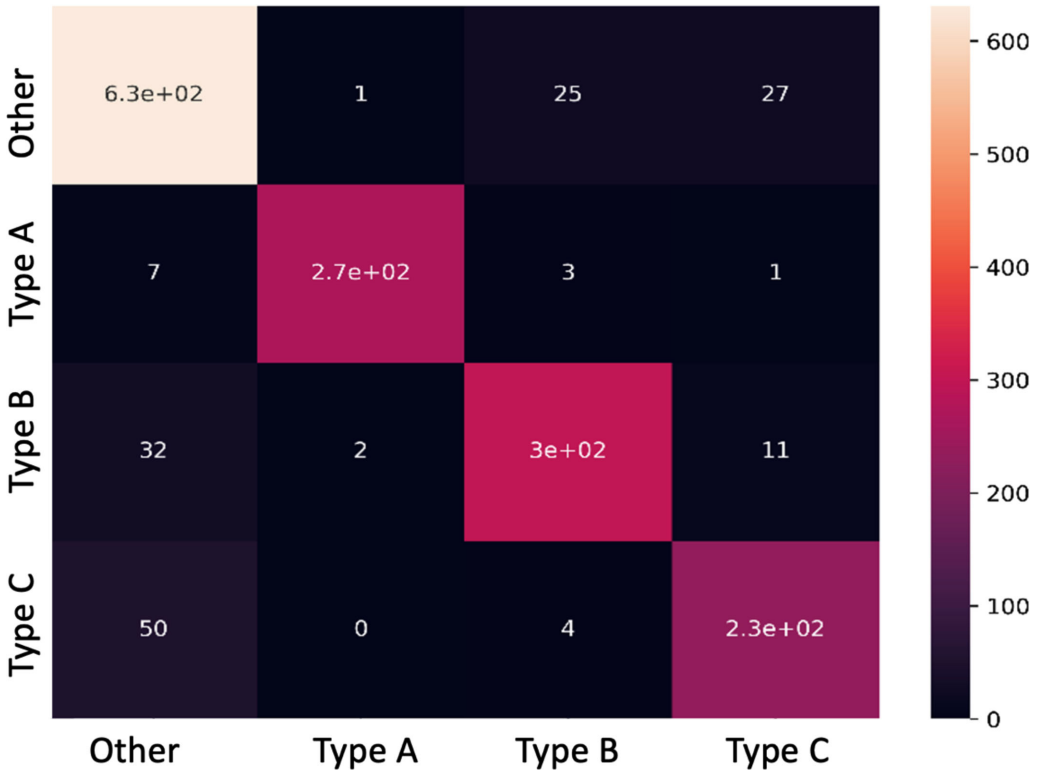

Figure 11. Confusion matrix for the LSTM neural network: with the palms or heels of the hands (type A), with the pulp of the index fingers (type B), and with the phalanges of the index fingers (type C).

the electronic patient record data of interest, such as the number of eye rubs during the last period. Several correlations could be made, such as with periods of the week and day, the effectiveness of warning signals in reducing the duration of eye rubbing, and so forth. Finally, it will also be necessary to develop an application compatible with iOS for users of the Apple Watch, thus covering most of the smartwatch technologies.

From a technical point of view, the challenge would be to develop an application connected to a Django or Flask (Python frameworks) platform on a server accessible via the internet so as to obtain a continuously updated model. However, this would require that the smartwatch be constantly connected to the internet or delayed transmission if the watch memory allows storage over a period of time.

Conclusions

Eye rubbing is described by many authors as the most important modifiable behavior in the management of keratoconus, in addition to night-time compression. Until now, eye rubbing has been evaluated by questioning or filling in questionnaires, but automated detection of eye rubbing by machine and deep learning is technically feasible. Millions of patients could benefit from this novel solution for detecting eye rubbing and reducing harmful movements. However, to be usable in practice, further projects should be undertaken using more extensive datasets to provide greater adaptability to the patient.

Acknowledgments

Presented at the Congress of the French Association of Implants and Refractive Surgery (SAFIR) 2023, Paris, France.

Disclosure: **I. Drira**, None; **A. Louja**, None; **L. Sliman**, None; **V. Soler**, None; **M. Noor**, None; **A. Jamali**, None; **P. Fournie**, None

References

1. Hashemi H, Heydarian S, Hooshmand E, et al. The prevalence and risk factors for keratoconus: a systematic review and meta-analysis. *Cornea*. 2020;39(2):263–270.

2. Edwards M, McGhee CN, Dean S. The genetics of keratoconus. *Clin Exp Ophthalmol*. 2001;29(6):345–351.
3. Bykhovskaya Y, Rabinowitz YS. Update on the genetics of keratoconus. *Exp Eye Res*. 2021;202:108398.
4. Najmi H, Mobarki Y, Mania K, et al. The correlation between keratoconus and eye rubbing: a review. *Int J Ophthalmol*. 2019;12(11):1775–1781.
5. Rabinowitz YS, Galvis V, Tello A, Rueda D, García JD. Genetics vs chronic corneal mechanical trauma in the etiology of keratoconus. *Exp Eye Res*. 2021;202:108328.
6. Ridley F. Eye-rubbing and contact lenses. *Br J Ophthalmol*. 1961;45(9):631.
7. Karseras AG, Ruben M. Aetiology of keratoconus. *Br J Ophthalmol*. 1976;60(7):522–525.
8. Lindsay RG, Bruce AS, Gutteridge IF. Keratoconus associated with continual eye rubbing due to punctal agenesis. *Cornea*. 2000;19(4):567–569.
9. Krachmer JH. Eye rubbing can cause keratoconus. *Cornea*. 2004;23(6):539–540.
10. McMonnies CW. Management of chronic habits of abnormal eye rubbing. *Cont Lens Anterior Eye*. 2008;31(2):95–102.
11. Bohac M, Koncarevic M, Pasalic A, et al. Incidence and clinical characteristics of post LASIK ectasia: a review of over 30,000 LASIK cases. *Semin Ophthalmol*. 2018;33(7–8):869–877.
12. McMonnies CW. Behaviour modification in the management of chronic habits of abnormal eye rubbing. *Cont Lens Anterior Eye*. 2009;32(2):55–63.
13. McGhee CNJ, Kim BZ, Wilson PJ. Contemporary treatment paradigms in keratoconus. *Cornea*. 2015;34(suppl 10):S16–S23.
14. de Almeida Gusmão Lyra JM, Leão EV, Machado AP. Artificial intelligence in keratoconus diagnosis. In: Almodin E, Nassaralla BA, Sandes J, eds. *Keratoconus: A Comprehensive Guide to Diagnosis and Treatment*. Cham: Springer International Publishing; 2022:215–228.
15. Nokas G, Kotsilieris T. Preventing keratoconus through eye rubbing activity detection: a machine learning approach. *Electronics*. 2023;12(4):1028.
16. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv. 2015, https://doi.org/10.48550/arXiv.1502.03167.
17. 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(56):1929−1958.
18. Kingma DP, Ba J. Adam: a method for stochastic optimization. arXiv. 2014, https://doi.org/10.48550/arXiv.1412.6980.
19. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. arXiv. 2017, https://doi.org/10.48550/arXiv.1706.03762.
20. Tori AA, Tori R, de Lourdes dos Santos Nunes F. Serious game design in health education: a systematic review. *IEEE Trans Learning Technol*. 2022;15(6):827–846.
21. Auepanwiriyakul C, Waibel S, Songa J, Bentley P, Faisal AA. Accuracy and acceptability of wearable motion tracking for inpatient monitoring using smartwatches. *Sensors (Basel)*. 2020;20(24):7313.
22. Stefana E, Marciano F, Rossi D, Cocca P, Tomasoni G. Wearable devices for ergonomics: a systematic literature review. *Sensors (Basel)*. 2021;21(3):777.
23. Rouzaud Laborde C, Cenko E, Mardini MT, et al. Satisfaction, usability, and compliance with the use of smartwatches for ecological momentary assessment of knee osteoarthritis symptoms in older adults: usability study. *JMIR Aging*. 2021;4(3):e24553.
24. Pradhan B, Bhattacharyya S, Pal K. Iot-based applications in healthcare devices. *J Healthc Eng*. 2021;2021:6632599.