2018 17th IEEE International Conference on Machine Learning and Applications

Classification of Eye Tracking Data using a Convolutional Neural Network

Yuehan Yin, Chunghao Juan, Joyram Chakraborty, Michael P. McGuire
Department of Computer and Information Sciences
Towson University
Towson, USA
{ yyin1, cjuan1}@students.towson.edu, { jchakraborty, mmcguire }@towson.edu

Abstract—**Historically, eye tracking analysis has been a useful approach to identify areas of interest (AOIs) where users have specific regions of the user interface (UI) in which they are interested. Many algorithms have been proposed to analyze eye tracking data in order to make user interfaces more effective. The objective of this study is to use convolutional neural networks (CNNs) to classify eye tracking data. First, a CNN was used to classify two different web interfaces for browsing news data. Then in a second experiment, a CNN was used to classify the nationalities of users. In addition, techniques of data-preprocessing and feature-engineering were applied. The algorithm used in this research is convolutional neural network (CNN), which is famous in deep learning field. Keras framework running on top of TensorFlow was used to define and train our CNN model. The purpose of this research is to explore how feature-engineering can affect evaluation metrics about our model. The results of the study show a number of interesting patterns and generally that deep learning shows promise in the analysis of eye tracking data.**

Keywords—eye tracking; CNN; deep learning; feature-engineering; gaze point; Keras; TensorFlow

I. Introduction and Related work

Eye tracking is a methodology that helps researchers understand visual attention by collecting eye measurements of where users look at a point in time, how long they look at something and the path their eyes follow [1]. The position of the eye can be reordered or captured by an eye-tracking device. Most modern eye trackers rely on a method called corneal reflection to detect and track the location of the eye as it moves [1]. Eye tracking has been applied in several fields, such as cognitive psychology, marketing, and human-computer interaction.

Machine learning is an emerging field in the analysis of eye tracking data where algorithms mimic human behavior and thinking by feeding large amounts of data with a known output (supervised learning) or an unknown output (unsupervised learning) [2]. However, earlier machine learning approaches, such as standard artificial neural networks, require strong domain expertise to extract important features as neurons to activate the learning process which limits their ability to be used effectively [3]. As an alternative machine learning method, deep learning can strengthen standard machine learning techniques in that it can simulate the human learning process by providing several sub-processes in handling data. When people start to learn information, they begin with looking at many specific features and form a basic summary. Then, this process is continued to form mid-level or to high-level summaries and finally extract the most important features to help judge if a decision is correct. If not, the process is reorganized and the human "learns" from the mistake and makes an adjustment. The combination of several processes forms a deep learning model. There are several deep learning models such as the Convolutional Neural Network (CNN) known to be successful in handling large scale real world problems [4]. A number of CNN models have been developed over the years. The LeNet-5 model was built to recognize hand-written digits with an accuracy of more than 99% [5]. The AlexNet CNN was the first to demonstrate the processing power of GPUs [6]. GoogLeNet went much deeper than above by using the inception module which operates as a subnetwork consisting of convolution, pooling, and a depth concatenation layers [7]. Resnet CNN speeds up train process with the concept of a residual unit with a skip connection. The skip connection outputs a copy of its inputs if the resulting weights are close to zero thus allowing inputs to traverse the network quicker [8].

Machine learning and deep learning are being applied to the analysis of eye tracking. For example, machine learning techniques have been shown to lead to superior detection of eye tracking events [9]. Another study developed the iTracker, which is a deep convolutional neural network and runs on mobile devices such as cellphones and tablets without connecting to additional equipment. In the study, the CNN was trained on the GazeCapture dataset, which was the first large-scale dataset for eye tracking, and could predict gaze robustly, and reach an error between 1.04cm and 1.69cm on cellphones and tablets respectively [10].

In this research, a modified LeNet-5 CNN model was applied to an eye-tracking study to classify two specific web user interfaces which users browsed in the experiment and two nationalities of users who browsed web user interfaces during the experiment. One of the challenges in using CNNs for eye tracking is the creation of meaningful features. Feature engineering is the process of using your own knowledge about the data and about the machine-learning algorithm at hand (in this case, a neural network) to make the algorithm work better by applying hardcoded (nonlearned) transformations to the data before it goes into the model [11]. A number of feature engineering techniques were applied to our dataset that was used to train and test our model. In following sections, we will introduce and describe models and our dataset in section II, and then depict data preprocessing and feature engineering which we applied in section III, and also discuss interesting results we found, and elaborate results of our research, and future work in section IV and V.

II. Models and Dataset

A. CNN Model

In this experiment, the LeNet-5 model [5] was adjusted and used as a benchmark. The model was then modified in an effort to increase performance. Our model uses 4 convolutional layers instead of 3, and keeps the same number of fully connected layers. The kernel size for each convolutional layer was 9-by-9, and the number of strides was 1. Same padding was used to pad the input in order to make the values of the dimensions of the output shape have the same numbers as the original input. The ReLU activation was used for convolutional layers and max pooling was used instead of average pooling. Each max pooling layer had a kernel size of 3-by-3 and a stride of 3 to make sure pooled areas do not overlap. Dropout layers were inserted after the second, third and fourth max pooling layers and the first fully connected layer to avoid the overfitting problem. The dropout rate for the dropout layers after max pooling layers was 0.2. The dropout rate for the dropout layer after first fully connected layer was 0.5. The dropout rate is the fraction of the features that are zeroed out; it's usually set between 0.2 and 0.5 [11]. The loss function used in the model was categorical cross-entropy. Cross-entropy is a metric from the field of Information Theory that measures the distance between probability distributions or, in this case, between the ground-truth distribution and the predictions [11]. The loss function expects the class labels encoded by using the one-hot encoding. The activation function for the last fully connected layer was softmax. In the case of this model, the softmax function generates an array of 2 probability scores summing to 1. Each score represents the probability that the current prediction is classified as one of two categories in our classification tasks. In addition, we used the Adam optimizer, which is an extension to stochastic gradient descent. For this implementation, a learning rate 0.0001 was used. The overall model architecture is shown in Figure 1. The figure also indicates the number of filters we used, the number of neurons in fully connected layers and shapes of inputs and outputs for each layer.

B. Dataset Summary

The dataset used in this paper was collected in an eye-tracking experiment including 20 participants (10 American and 10 Saudi Arabian) with designated tasks to find certain information from two websites: Google News and News Map as shown in Figure 2 respectively. Users were divided into 2 groups where first group uses Google News first and the second uses News Map first. The data was collected and exported through the Tobii X60 eye tacking device. A comprehensive collecting data fields are not provided, but a few key attributes used in the data-preprocessing step are listed here. These attributes are Recording Date, Participant Name, and Local Timestamp, GazePointX (ADCSpx) and GazePointY (ADCSpx).

III. Data Preprocessing And Feature Engineering

In this section, we will elaborate the data-preprocessing step, corresponding database setup and feature-engineering step. The overall work flow path with related technology and libraries used is shown as in Figure 3.

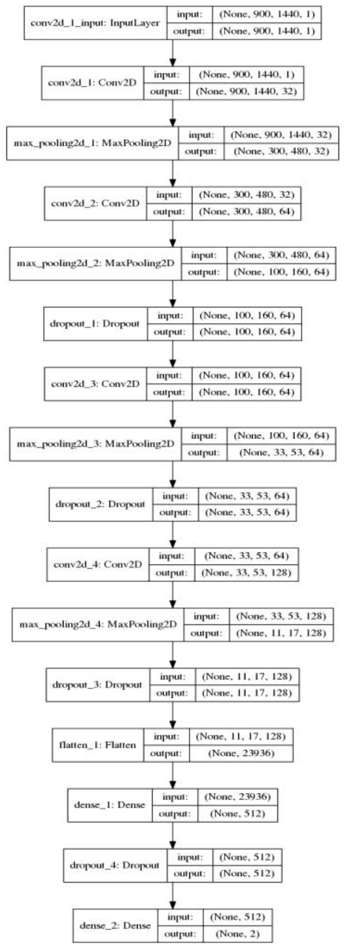
conv2d_1_input: InputLayer | input: (None, 900, 1440, 1) | output: (None, 900, 1440, 1)
conv2d_1: Conv2D | input: (None, 900, 1440, 1) | output: (None, 900, 1440, 32)
max_pooling2d_1: MaxPooling2D | input: (None, 900, 1440, 32) | output: (None, 300, 480, 32)
conv2d_2: Conv2D | input: (None, 300, 480, 32) | output: (None, 300, 480, 64)
max_pooling2d_2: MaxPooling2D | input: (None, 300, 480, 64) | output: (None, 100, 160, 64)
dropout_1: Dropout | input: (None, 100, 160, 64) | output: (None, 100, 160, 64)
conv2d_3: Conv2D | input: (None, 100, 160, 64) | output: (None, 100, 160, 64)
max_pooling2d_3: MaxPooling2D | input: (None, 100, 160, 64) | output: (None, 33, 53, 64)
dropout_2: Dropout | input: (None, 33, 53, 64) | output: (None, 33, 53, 64)
conv2d_4: Conv2D | input: (None, 33, 53, 64) | output: (None, 33, 53, 128)
max_pooling2d_4: MaxPooling2D | input: (None, 33, 53, 128) | output: (None, 11, 17, 128)
dropout_3: Dropout | input: (None, 11, 17, 128) | output: (None, 11, 17, 128)
flatten_1: Flatten | input: (None, 11, 17, 128) | output: (None, 23936)
dense_1: Dense | input: (None, 23936) | output: (None, 512)
dropout_4: Dropout | input: (None, 512) | output: (None, 512)
dense_2: Dense | input: (None, 512) | output: (None, 2)

Figure 1. Convolutional Neural Network Architecture.

Japanese prime minister stands by apologies for Japan's WWII abuses
Tension between gay, religious rights plays out at top US court

Figure 2. Left: Google News UI. Right: News Map UI.

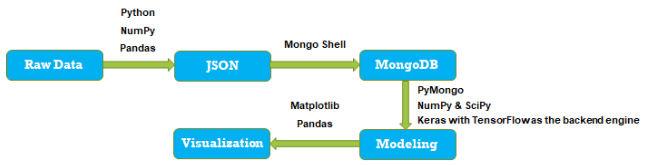

Figure 3. Overall work flow path.

A. Data Preprocessing

The raw data file contains about 650,953 records. For our classification tasks, we chose to observe patterns of users' eye movements during short time intervals. We can tally all the records in a time interval, and then convert them into one 2 dimensional (2D) array whose size is 900 x 1440 which is based on the screen resolution. A gaze point location in a 2D array is determined by its X and Y coordinates. Since the size of a 2D array is the same as the size of the screen resolution, it is very easy to locate gaze points in a 2D array according to its coordinates. When a gaze point is initially recorded in the 2D array, the value of the location is set to 1. If another gaze point occupies the same location, the value in that location will be increased by 1. If a location does not have any gaze points, the value of the location is 0. The short time interval for each array was 10 seconds meaning that each array represents 10 seconds of eye tracking data.

The JavaScript Object Notation (JSON) format was chosen to store the eye tracking data. The JSON specification states that data can be structured in either of the two following compositions, which are a collection of name/value pairs and an ordered list of values [13]. Another reason that JSON was chosen was that MongoDB was used to store the transformed data. The JSON representation of the transformed data consists of e eight fields with different data types. The data_id field is used to identify each JSON representation in our data set for our CNN model. The gp_image field is a 2D array containing gaze points for each 10 seconds. The gp_sequence field has a custom structured data type that includes a timestamp sequence for each 10 seconds and corresponding existed pairs of gaze point coordinates (GazePointX and GazePointY). The interface field is a class label to show the user interface for the particular set of gaze points. This can be Google News or News map. The nationality field is a class label to indicate a user's nationality, which is America or Saudi Arabia. The participant_name field is used to identify an individual participant by using an English letter concatenated with a number, such as A1 and S1. The timestamp_begin field and the timestamp_end field regard the start and the end for each 10 seconds interval.

B. Database Setup

One single JSON file containing a certain number of JSON documents was generated using the Python programming language. As was mentioned before, MongoDB was used to store our dataset. MongoDB is an open source and popular NoSQL database management system. The data model and persistence strategies are built for high read-and-write throughput and the ability to scale easily with automatic failover [14]. Information stored in MongoDB is in documents instead of rows. MongoDB can handle both schema-less and with-schema data and store them with BSON format for optimization. The single JSON file was inserted into MongoDB by using commands in the Mongo Shell. Also, each JSON document can be identified by the data_id field, and user-defined ascending indexes on data_id field of each document were created. This reduces overhead in retrieving data from the database and speed up our training time a little bit.

C. Feature Engineering

The purpose of feature engineering is to make a machine learning model to use data preprocessing steps to enhance features in the data, thus making the recognition of features easier for the model. As we mentioned before, there was a field called gp_image, which is a 2D array including gaze points for each 10 seconds. Figure 4 depicts six graphs which are plotted by using two 2D arrays from our data set and have titles to indicate corresponding web user interfaces, which are Google_News and News_Map. The plotted gaze points are not easy to see by just using raw data. When the graph becomes much larger, the positions of those points can be seen but the locations are not obvious. The positions of gaze points are much easier to see after maximum filters, which are nonlinear filters [15], are applied with different sizes to the 2D arrays. We use two sizes of maximum filters, which have filter sizes of 5-by-5 and 15-by-15. The maximum filter enhances the bright points in an image. In addition, before maximum filters were applied to 2D arrays from gp_image fields or the raw 2D arrays were fed into our model, we performed min-max scaling on them first in order to benefit learning of our model.

In addition to maximum filter, we performed another way of feature engineering. The gp_sequence field in one JSON document consists of a timestamp sequence for each 10 seconds and corresponding pairs of gaze point coordinates. In this apporach, the objective is to connect the gaze points together according to the timestamp sequence to mimic eye movements to generate the corresponding scan paths. A scan path is still represented in a 2D array which is associated with each 10 seconds. In Figure 5 shown below, six graphs were produced from our program based on a time sequence within 10 seconds and pairs of gaze point coordinates. The path without applying a maximum filter is very thin, and not obvious because of the displayed graph size. Also, maximum filters were applied so as to make the scan path bolder. The bigger the maximum filter size is, the bolder the scan path will become. We assumed it could make our model have good accuracy by applying feature engineering rather than just by using raw data with min-max scaling.

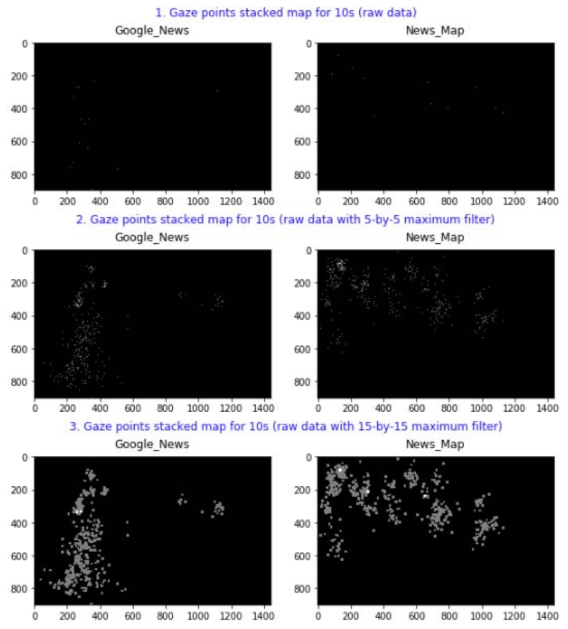

Figure 4. Gaze points stacked map for 10s (raw data).

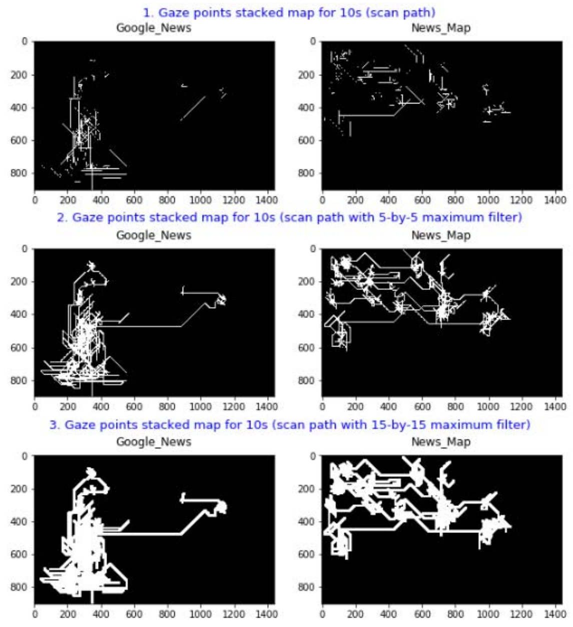

Figure 5. Gaze points stacked map for 10s (scan path).

IV. Results and Discussion

This section introduces the deep-learning framework used in this research, describes the data partitioning method, and evaluation metrics. The research focuses on two binary classification tasks including classifying web user interfaces and classifying nationalities of users. In addition, we will compare the experiment results using visualization related to our two classification tasks.

A. Deep-learning Framework

We chose Keras [12], which is a deep-learning framework for Python that provides an interface to the TensorFlow deep learning library developed by Google [16]. Keras was initially developed for researchers, with the aim of enabling fast experimentation. In addition, it is a model-level library, providing high-level building blocks for developing deep-learning models and provides a convenient way to define and train almost any kind of deep-learning model [11]. Therefore, we used Keras to build and train our model on CPUs.

B. Data Partitioning Method

We used the 70/30 holdout method to split the dataset into training sets and testing sets for our two classification tasks. In TABLE I below, it displays information in detail about our data partition. During the splitting process, our program also randomly shuffled the JSON documents to make them unordered. Training sets were generally kept evenly distributed where each class had an even number of class values for each class. However, classes in testing sets could not be evenly distributed since classes were unbalanced in original dataset. We used one-hot encoding for class labels. There are 33 JSON documents having unknown values in nationality fields. Hence, the total number of JSON documents for the two classification problems was different. In addition, two csv files were generated to include data index lists of JSON documents for training sets and testing sets for our two classification tasks. The order that data was fed into the model was determined by the order of the csv files. The last preprocessing step was to reshape the shapes of inputs before they were fed into the model.

C. Evaluation Metrics

A number of evaluation metrics were used to assess how well our classifier performs its job to make predictions. This includes accuracy (or recognition rate), recall (or sensitivity), specificity, precision and negative predictive value. Each metric was calculated using a confusion matrix that included true positives, true negatives, false positives and false negatives. The confusion matrix is a useful tool for analyzing how well your classifier can recognize tuples of different classes [17]. The accuracy or recognition rate is the percentage of correct prediction performed by the classifier on a given test set. Recall (or sensitivity) can show how sensitive the model is at catching true positives. Specificity can measure the performance of the model at detecting true negatives. Precision can measure how often the prediction result is true positive when a positive value is predicted. Negative predictive value can measure how often the prediction result is true negative when a negative value is predicted. In our test sets for two classification tasks, classes are unbalanced, but both classes are important.

D. Web User Interfaces Classification

TABLE II and III show the evaluation results for six experiments where a number of feature engineering techniques were tested. They include raw data (gaze points), scan path, raw data with maximum filter whose size is 5, raw data with maximum filter whose size is 15, scan path with maximum filter whose size is 5, and scan path with maximum filter whose size is 15, respectively. Our first task is to use eye tracking data to classify whether the user was using Google News or News Map. The model was trained using 20 epochs. The training accuracy was around 99% or close to 100% with training loss continually decreasing for all 6 experiments. We then tested our model for each set of features. The Google News class was encoded to [1, 0] and the News Map class was encoded to [0, 1]. Hence, index 0 indicates Google News, and index 1 implies News Map. The null accuracy is the percentage of correct prediction accomplished by always predicting the most frequent class. The null accuracy of our test set is 57.72%.

Figure 6 shows visualizations of confusion matrices of our 6 experiments. In our test set for this task, the two classes are unbalanced. Hence, just using accuracy or recognition rate to evaluate the performance of our model will not be a good option. We computed the evaluation metrics to see how the model performed after 20 epochs. The results of the evaluation measures are shown in TABLE II below. The recall and precision are related to News Map web UI class, because it indicates News Map when the model predicts an index value equal to 1. The specificity and negative predictive value relate to the Google News UI class, since it indicates Google News when the model predicts an index value equal to 0. By using raw data only to train and test our model, it has the highest values which are 81.28% and 76.03% for recall, which can show the model can detect a true News Map web user interface class with the probability which is 81.28%, and for negative predictive value, which can tell the model has the probability about 76.03% to have a true prediction result when it predicts a class which is Google News web user interface. By using raw data with maximum filter with a filter size of 5, it has the highest values for specificity and precision. The raw data with a maximum filter size of 5 produced the best results in predicting Google News correctly 89.78% of the time and News Map correctly 90.85% of the time. Comparing the results of by using raw data with maximum filter whose size is 5 and by using raw data with maximum filter whose size is 15, we found that specificity and precision would decrease, and recall and negative predictive value would increase when the size of maximum filter increased. Comparing the results of by using scan path with maximum filter whose size is 5 and by using scan path with maximum filter whose size is 15, specificity and precision would increase, and recall and negative predictive value would decrease when the size of maximum filter increased.

E. Nationalities of Users Classification

The second task is to classify nationalities of users. We also used the same model with same hyper parameters setting. The inputs were same, but the class labels were the nationalities of users instead of web user interfaces. The shape of inputs was the same as classifying web user interfaces. The model was trained 20 epochs on the training set and an accuracy of around 99% or close to 100% was achieved with continually decreasing training loss for all 6 experiments. We then tested our model for the 6 experiments. The American class was encoded to [1, 0] and the Saudi Arabian class was encoded to [0, 1]. Therefore, index 0 represents American users, and index 1 means Saudi Arabian users. The null accuracy of the test set is 58.28%.

Figure 7 shows visualizations of confusion matrices of the 6 experiments for this task. The results of the evaluation metrics are displayed in TABLE III below. The recall and precision are related to the Saudi Arabian class, because it represents Saudi Arabian when the model predicts index 1. The specificity and negative predictive value are related to American class, because it represents American when the model predicts index 0. Comparing the results of raw data and scan path with no filter, the approaches are different where using the scan path data is a little bit better than raw data except for specificity. It has the highest values regarding accuracy, specificity, precision and negative predictive value. It can catch true American class with 74.05% and have the percentage which is 80.00% to correctly predict a Saudi Arabian class when a prediction made by the model is Saudi Arabian class. Using raw data with a filter size of 5 and scan path data with a filter size of 15 have the highest values for recall which can show the model can detect a true Saudi Arabian class with the probability which is 74.86%. Overall, the performance of classifying nationalities of users is worse than the performance of classifying web user interfaces. One reason for this might be that the text reading habits are different between Americans and Saudi Arabians. Americans read from left to right, and Saudi Arabians read from right to left. In our data transformation step, we did not encode timestamps with the scan path. Therefore, it cannot infer reading habits between these two classes. If the timestamps can be encoded in feature engineering process, the performance of our model will be boosted.

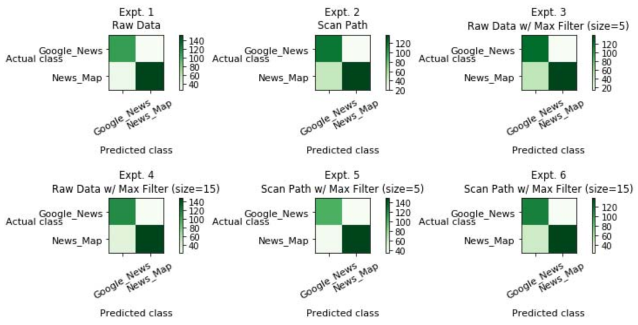

Figure 6. Confusion matrices for classifying web UIs.

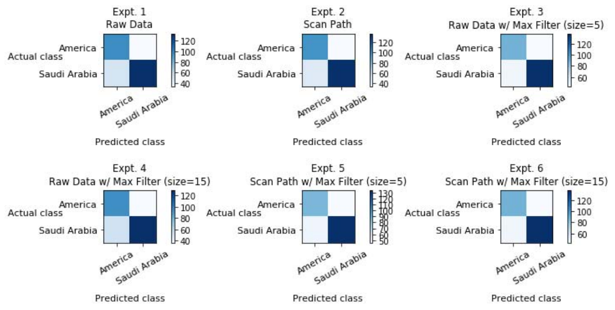

Figure 7. Confusion matrices for classifying nationalities of users.

TABLE I. DATA PARTITION

Web User Interfaces Classification			Nationalities of Users Classification		
	Training Set	***Test Set***		***Training Set***	***Test Set***
Google News	379	137	***America***	367	131
News Map	378	187	***Saudi Arabia***	367	183
Total Number for each set	757	324	***Total Number for each set***	734	314
Total Number	1081		***Total Number***	1048	

TABLE II. RESULTS FOR CLASSIFYING WEB UIS

	Accuracy	**Recall**	**Specificity**	**Precision**	**Neg. Pred. Val.**
Raw Gaze Points Max Filter = none	81.17%	81.28%	81.02%	85.39%	76.03%
Scan Path Max Filter = none	78.40%	73.26%	85.40%	87.26%	70.06%
Raw Gaze Points Max Filter = 5	80.86%	74.33%	89.78%	90.85%	71.93%
Raw Gaze Points Max Filter = 15	81.48%	79.14%	84.67%	87.57%	74.84%
Scan Path Max Filter = 5	77.78%	79.68%	75.18%	81.42%	73.05%
Scan Path Max Filter = 15	77.78%	73.80%	83.21%	85.71%	69.94%

TABLE III. RESULTS FOR CLASSIFYING NATIONALITIES OF USERS

	Accuracy	**Recall**	**Specificity**	**Precision**	**Neg. Pred. Val.**
Raw Gaze Points Max Filter = none	72.93%	72.13%	74.05%	79.52%	65.54%
Scan Path Max Filter = none	74.20%	74.32%	74.05%	80.00%	67.36%
Raw Gaze Points Max Filter = 5	71.66%	74.86%	67.18%	76.11%	65.67%
Raw Gaze Points Max Filter = 15	71.34%	70.49%	72.52%	78.18%	63.76%
Scan Path Max Filter = 5	70.06%	73.22%	65.65%	74.86%	63.70%
Scan Path Max Filter = 15	71.66%	74.86%	67.18%	76.11%	65.67%

V. CONCLUSION AND FUTURE WORK

In this research, a CNN model was trained and tested for each experiment for two classification tasks. Confusion matrices were visualized for each experiment and a number of evaluation metrics were computed to assess the performance of the model in the two classification tasks. The CNN model performed better in classifying specific web user interfaces than the recognition of the nationalities of users.

In the future, we will need to develop a different data structure to represent our data with encoding timestamps. Also, a new feature-engineering approach will be developed for boosting the performance of our model. We will apply a method to find out suitable hyper parameters to make our model perform better. More data will be collected and put into training and testing. The data can be further used to develop a user-auto-adapted system to improve user experience based on his/her browsing pattern.

REFERENCES

[1] J. R. Bergstrom and A. J. Schall, Eye Tracking in User Experience Design. Waltham, MA: Elsevier Inc., 2014.

[2] J. Schmidhuber, "Deep learning in neural networks: An Overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[3] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[4] Y. LeCun, B. E. Boser, , J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*, pp. 396-404, 1990.

[5] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. *86*(11), pp. 2278-2324, 1998.

[6] Krizhevsky, A., Sutskever, I., and Hinton G. E, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, page 4, 2012

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778, 2016.

[9] R. Zemblys, D. C. Niehorster, O. Komogortsev and K. Holmqvist, "Using machine learning to detect events in eye-tracking data", *Behavior Research Methods*, vol. 50, no. 1, pp. 160-181, 2018.

[10] K. Krafka*, A. Khosla*, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik and A. Torralba, "Eye Tracking for Everyone", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] F. Chollet, Deep Learning with Python, Shelter Island, NY: Manning Publications Co., 2018.

[12] Keras. *Keras*. [online] Available at: https://keras.io/ [Accessed 1 March. 2018]

[13] B. Smith, Beginning JSON. New York City, NY: Apress, 2015

[14] K. Banker, P. Bakkum, S. Verch, D. Garrett and T. Hawkins, MongoDB in Action, 2nd ed. Shelter Island, NY: Manning Publications Co., 2016.

[15] R. Chityala and S. Pudipeddi, Image Processing and Acquisition using Python. Boca Raton, FL: CRC Press, Taylor & Francis Group, LLC, 2014.

[16] TensorFlow. *TensorFlow*. [online] Available at: https://www.tensorflow.org/ [Accessed 1 March. 2018].

[17] J. Han, M. Kamber and J. Pei, Data Mining Concepts and Techniques, 3rd ed. Waltham, MA: Morgan Kaufmann Publishers, Elsevier Inc., 2012.