Using eyesight to control a character in a videogame

Timothée Ducros
Dorian Gailhard
Arnaud Minondo

I) Context :

I.1) Motivation :

Last year at Telecom Paris, we had a semester project called PACT for which we made a small videogame where the character could be moved by looking at a direction - i.e. if the player looks up, the character moves upwards, if he looks left the character moves leftwards, etc…

The extraction of the part of the screen that is looked at by the user was done thanks to a pre-trained neural network, but its accuracy was around 60% - which did not satisfy us - and it had too big an impact performance-wise - presumably because it was made to determine *exactly* the part of the screen that is looked at, and not just the approximate direction.

Thus the motivation for our MALIS project was to build ourselves our own pseudo eye-tracking software with attention maintained on its CPU usage acceptable, to better understand the logic behind the collection of data and the training of a model.

I.2) The goal :

The screen is divided into nine parts as in the figure below :

Top-left	Top	Top-right
Left	Center	Right
Bottom-left	Bottom	Bottom-right

The goal is to classify pictures taken thanks to a webcam into these different parts.

II) Collection of data:

II.1) Taking pictures :

As we wanted to completely understand the logic behind the making of a model, we decided to build our own collection of data, thanks to a program of our own making. The idea is simple, the “guinea pig” first sees the webcam feedback and try to fit the eyes into a rectangle by zooming / dezooming and dragging the picture, and then has to stand still - so the eyes remain aligned - while staring at a red dot moving on the screen, like in the two figures below :

JE BIDONNE DES SCREENS PLUS TARD

The program then regularly saves the webcam feedback - in grayscale - and adds its entry into a .csv file storing the location of the red dot.

At first, we only started with 5 persons - our immediate environment - which all had brown eyes and white skin, into three color and lighting set - outdoor under sunlight, indoor under strong artificial yellow light, indoor under light artificial light - totalling 500 pictures for each program use. We then asked 10 additional persons to do the staring and send us their data, totalling 5000 additional pictures.

Below are some examples of the data collected :

PLUS TARD

To further extend the collection of data, we artificially added pictures through small perturbations of the existing ones : translations of a few pixels in a direction, adding a small noise and changing the color balance and contrast. Here are some perturbations applied to the previous photos :

PLUS TARD

As we had only one person with blue eyes amongst the participant, and every person had white skin, we applied image editing filters on our dataset to try to reproduce the look of dark skin and blue eyes, like as follows :

PLUS

II.1) Organizing the dataset :

Then we organized the pictures into nine different folders according to the screen division we introduced earlier, with a .csv file storing the following properties : XDOT, YDOT, IRIS_COLOR, SKIN_COLOR, PERTURBATION.

III) Building the model :

III.1) Choosing the model :

III.2) The libraries :

III.3) Progress :

III.3.1) Training on only one set of data (one person, one ideal lighting setting) :

We first started with a simple feedforward neural network with the following structure : A PRECISIER - the first layer has ? neurons, the second ?,etc … We used A PRECISER learning rate. All these parameters were obtained by trial and errors.

At first we trained the model on only one person having brown eyes, under the same lighting conditions and with no additional perturbation, and had good results. The following is the plot of accuracy after each epoch :

A FAIRE

As can be seen, the results were good, with a training accuracy of A PRECISER and a testing accuracy of A PRECISER. This was expected because the outline of the eyes remains the same and only the position of the iris changes.

III.3.2) Training on a bigger set of data (five persons, ideal lighting setting) :

We then combined the dataset of the five initial persons - each of them had brown eye - with an ideal outdoor sunlight setting, and trained the same neural network on it. The training set was composed of four of the five persons, the last set being the testing data.

The accuracy dropped by A PRECISER %, as seen in the following plot :

A FAIRE

By trial and error, we arrived to the following training and testing accuracy :

A FAIRE

The neural network structure was then : A PRECISER
The learning rate used was the same as before.

To speed up the learning process - it took 10 minutes by epoch and we did 24 - and maintain a small impact on performance, we decided to decrease the pictures' dimensions from A PRECISER to A PRECISER - fewer pixels mean fewer inputs. We expected a slight decrease in accuracy as there would be less information, but as the critical information is the position of the iris and the outline of the eyes, the loss was not so severe.

We obtained the following accuracies :

A FAIRE

This time the neural network had the structure : A FAIRE
The learning rate was : A FAIRE

III.3.3) Improving the performance, first try :

To improve accuracy, we decided to train the model on the outline of the pictures, obtained through edge-detection algorithm - based on the Fourier transform. We applied this algorithm to every picture of the above five persons and came to sample like this :

PLUS TARD

We came to accuracies of ?? for the training set and ?? for the testing set.

III.3.4) Improving the performance, second try :

Then we decided to try Bootstrap Aggregation : we trained ? models, each on a partition of the original dataset, and combined them on a bigger model whose output is the majority vote of the smaller models.

EXPLIQUER FONCTIONNEMENT ? c'est dans le cours déjà donc jsp

PLUS TARD

We came to accuracies of ?? for the training set and ?? for the testing set.

While not ideal, we thought it could be useable in our case as the misclassification appeared mostly for the ambiguous pictures where the eyesight is around the boundaries in the partition of the screen, reminded below :

Top-left	Top	Top-right
Left	Center	Right
Bottom-left	Bottom	Bottom-right

As our goal was to allow players to control a character, such misclassification would not really hamper the game as the player would just have to look a bit more on the zone to make it work. We decided to then test the whole dataset.

III.3.5) On the whole dataset :

The introduction of blue eyes, dark skins and bad lighting conditions made the accuracy drop to A PRECISER %.

PLUS TARD

When isolating the pictures in bad lighting conditions, we arrived to accuracies of : A PRECISER.

PAS DE GRAPHIQUE JE PENSE QUE CE SERA BON

This was expected as the iris is not really seeable in the dark. We decided to put them out of the dataset, and then retrained the model :

PLUS TARD

This time the accuracies were : A PRECISER
This is much better as expected.

IV) Conclusion :

We achieved building a model with accuracies of A PRECISER.

After all that, we could have introduced slanting eyes into the dataset and searched for more eye colors to diversify the samples.

For the model, we could use convolutional neural networks instead of simple feedforward networks.

il faut dire comment on a fait les database et nos tres nombreux probele et reussite en sortant des valeuir d'accuracy d'algorithme aleatoire, style:

- photo de moi et ma famille, accuracy a 70%M mais fonctionne mal sur des amis. (genre chaqu'un de nous a entrainé son code sur un echantillons ses vacances et on calle des graphique de chaqu'un, puis en mélangé)
- difficile de trouver des yeux bleu et/ou vers ce qui reduit l'accuracy de 20 points de ppurcentage en moyenne
- la couleur de peau influ beaucoup, nous n'avions que deuxechantillons dans notre database et le programme devient aléatoire
- malgrès un bon nombre d'yeux plus fermé/d'origine asiatique dans les données, l'accuracy baisse de quelques popints en general
- Des grands yeuxc noirs fonctionne tres tres bien, on approche des 90%
- dans une piece sombre, le code est aleatoire
- (si on a le contraste, on peut en parler)
- e, montant ou reduisant la qualité d'image, on peut avoir des resultats différent (bon graph)
- en fonction du temps d'echantillonage pour le deplacement du personnage (jeu reactif) on a es pourcentage différent