Predicting Classroom Attention from Images

Bachelor Project Fall 2018

Marc Bickel

A Bachelor Project for the CHILI Lab and the CVLAb at EPFL

Computer Science
EPFL
Switzerland
11.01.19

Abstract

This project uses videos labelled by hand as data for a Deep Neural Network constructed using PyTorch. The goal is to be able to predict student attention levels in a classroom, on the only basis of their position. These positions are extracted using the OpenPose library, normalised and precomputed, and fed to fully connected Network. The scale used to represent the attention levels is a linear scale from 1 to 10, and the Network achieves a mean Error to the ground truth of 0.706 and an Accuracy of 57.6%. The final output is a visual representation of the student attention levels.

CONTENTS

I. Introduction

THE level of attention students display in class can vary from one minute to another. It is visually easy for us humans to be able to judge if a person is attentive, but is it the same for a computer ? This project is a bridge between a world-class computer vision task, position recognition, and psychology theory that links position to attention levels using Deep Learning. It aims to answer the question asked beforehand and to provide professors, teachers and online gurus to get feedback on the quality of their teaching. Having a feedback on the engagement of the audience allows for growth and improvement over time. A secondary goal is to be able to provide data for Psychology studies related to classroom attention, by removing the need to label hours of video by hand.

II. Exploratory Data Analysis

A. Starting Data

The data used for this project is divided into different categories. First and foremost, the videos make up the most important part. They have been provided by the CHILI Lab at EPFL. These are recordings of Prof. Dillembourg during one of his lectures. They are standard videos, each 45 minutes long, as this is the length of a single lecture at EPFL. There are 6 cameras recording from different angles. The first four record from the front, while the other two record either from the back or are filming the professor. The lastest two are not used in this project. One single camera angle does not capture the entirety of the room, which means that, to get data about every student, every front-located camera has to be used.

Next, coming from the Psychology Department of the University of Zurich, there are the attention levels for 2 recorded lectures. These have been encoded manually by two psychologists. They have done this by manually pausing the videos previously mentioned every 10 seconds, and writing down the attention level of each visible student, according to section II-B.

The frames from which the attention levels have been fixed are extracted using the media player VLC. To recap, the extraction happens once every 300 frames, or 10 seconds. Fig. 1 shows an example of such a frame.

Fig. 1. Example of frame extracted from video

Alongside the videos, there was also a file linking the students to their IDs. The connection between this file and the videos is described in section IV-A.

Fig. 2. Example of annotations linking frames and student IDs

Further videos than the one shown in Fig. 1 were provided, but they were useless in the scope of this project, as they were not labelled.

B. Underlying Psychology Theory

As explained in Hommel (2012) [5], and Ehrhardt, Findeisen, Marinello, Reinartz-Wenzel (1981) [6], the attention level theory on which this project is based is dual-faced. First, it is possible to quantify the attention level in a numerical manner, which sparks the 1-to-10 attention scale used in this project, and second, 4 grand categories of attention exist.

TABLE I
Attention Categories

	Attention	
Passivity	Attentive Active	Distracted Active
	Attentive Passive	Distracted Passive

The Attention refers to whether the student is following along the lesson or not, and the Passivity measures the commitment while doing so. To give some examples, Attentive Active would correspond to a student listening to the professor and taking notes, while Attentive Passive would be listening in a laid back position. Distracted Passive would include looking at one's phone or outside the window, and Distracted Active would refer to talking with one's neighbor.

III. Tools of the Trade

This project has been developed using the Python programming language, the library OpenPose, and the PyTorch framework. The Python language and wrappers allow to fuse every aspect of the project seamlessly in Jupyter Notebooks. OpenPose is used to extract the student position in the pictures, and PyTorch allows to easily create a Neural Network used for the prediction based on the data from OpenPose.

A. Python

The Python language is used, as it is widely regarded as the best language for Machine Learning applications. The version used is 3.6, and the standard Numpy and Scipy libraries are included. These are respectively in versions 1.14.3 and 1.1.0. The coding environment used was a Jupyter Notebook, using the IPython Kernel. A few utilities-oriented libraries that also saw some use are: MatPlotlib, PIL, tqdm, Pandas.

B. OpenPose

OpenPose is an open source coordinate detection library developed by the Carnegie Mellon University[1] . It uses deep learning and caffe models to perform one of the state-of-the-art computer vision tasks: position recognition.

Fortunately, it has a python wrapper, but the installation is not very polished. Integrating it into the Jupyter Notebooks is not a native feature of the python wrapper, and required some addionnal work. The IPython kernel is at fault, and a few modifications to global system variables and other little hacks were necessary. It would be too long to simply list them here, but please do contact marc.bickel@epfl.ch should reproducibility be an issue.

The way in which OpenPose has been used is the following: given an input image (itself extracted from the videos), it computes the coordinates of 25 keypoints (shown in Fig. 3) of every person detected in the picture, as shown in Fig. 4. It outputs either the keypoints coordinates in the referential of the input picture, with the $(0, 0)$ origin point in the upper left corner, or a picture composed of the original input image plus the keypoints directly displayed on top of it. For debugging purposes, the output image has seldom been used, but the usage has otherwise been focused on the numerical coordinates of the keypoints.

What OpenPose allows to do is to focus on the learning and prediction aspects of the project. No complicated Computer Vision algorithm has been needed, because everything was already taken care of by OpenPose. A second benefit is to avoid using a Convolutional Neural Network. As explained in section III-D, a fully connected deep Neural Network has been used instead.

This approach was preferred over feeding the images to the Neural Network directly, because it is independent from the environment. A student in another classroom will still be recognized by OpenPose, and its keypoints detected. At the contrary, a new classroom/background can throw a Convolutional Neural Network way off.

C. PyTorch

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs[2]. It allows for an easy way to construct and train deep Neural Networks. It has been used in its 1.0 version.

D. Deep Neural Network

A lot of iteration and experimentation has taken place throughout the entire project in order to find the best Neural Network for the task at hand. As mentioned above, since OpenPose takes care of keypoints extraction from the image, the Network is made of fully connected layers. No convolutions are needed, since there is no image as input to the proper Network. There is a singular output node, thus the Network is not a classifier. This was chosen to take advantage of the fact that an attention level of 5 is closer to 6 than it is to 2. With categories, this information would have been lost. Moreover, the continuous output allows to decide on a parameter for the metric of the performances of the Network, as discussed in section V-B.

Fig. 5 gives a summary of the architecture of the Network used to achieve the results presented in the section V. Linear layers are the PyTorch version of fully connected layers. Batch Normalization layers are used to further normalize the data, after the manual one performed in section IV-C, enhancing the performance slightly. The depth of the Network and number of neurons for each layer have been determined by trial and error.

[1]https://github.com/CMU-Perceptual-Computing-Lab/openpose
[2]https://pytorch.org/

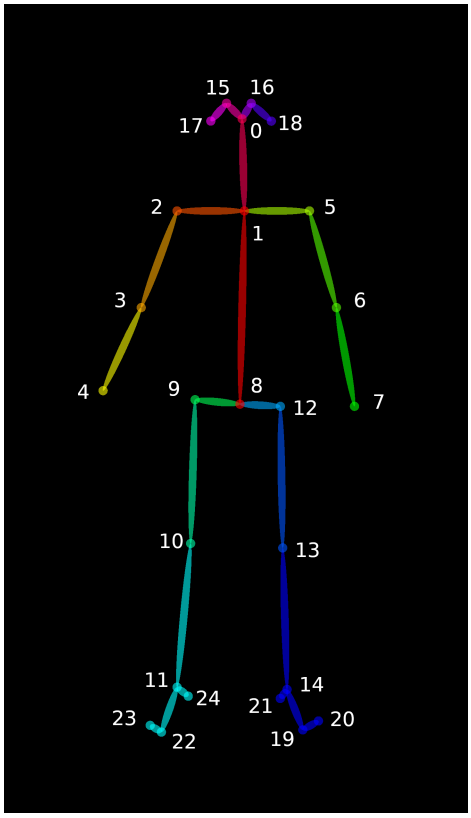

Fig. 3. OpenPose keypoints and their ordering

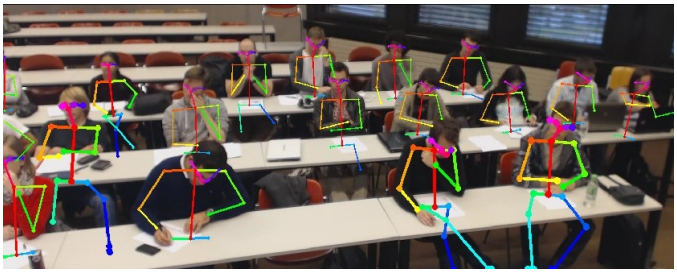

Fig. 4. Example of frame processed by OpenPose

IV. Development and encountered difficulties

A. Data articulation

Because the data was separated, as explained in section II, a way to link every piece together had to be found. The

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1500)	76500
batch_normalization_1 (Batch	(None, 1500)	6000
dense_2 (Dense)	(None, 1500)	2251500
re_lu_1 (ReLU)	(None, 1500)	0
dense_3 (Dense)	(None, 1500)	2251500
re_lu_2 (ReLU)	(None, 1500)	0
dense_4 (Dense)	(None, 1500)	2251500
re_lu_3 (ReLU)	(None, 1500)	0
dense_5 (Dense)	(None, 1500)	2251500
re_lu_4 (ReLU)	(None, 1500)	0
dense_6 (Dense)	(None, 1500)	2251500
re_lu_5 (ReLU)	(None, 1500)	0
dense_7 (Dense)	(None, 1500)	2251500
re_lu_6 (ReLU)	(None, 1500)	0
dense_8 (Dense)	(None, 1500)	2251500
re_lu_7 (ReLU)	(None, 1500)	0
dense_9 (Dense)	(None, 1500)	2251500
re_lu_8 (ReLU)	(None, 1500)	0
dense_10 (Dense)	(None, 1500)	2251500
re_lu_9 (ReLU)	(None, 1500)	0
dense_11 (Dense)	(None, 1500)	2251500

Total params: 22,597,500
Trainable params: 22,594,500
Non-trainable params: 3,000

Fig. 5. Summary of the PyTorch Deep Neural Network

solution implemented lies in creating by hand a mapping of coordinates to IDs, and running a variation of the closest neighbour algorithm for each set of keypoints in the input image. It makes sense to use the closest neighbour algorithm, because the setting of the project is a classroom. The students are supposed to be seated at their place, and not move around too much. The most movement that can be expected is to lean, but even then, it should not last more that one or two frames. Because of this, the difference between the location of different students is orders of magnitude greater than the difference between different positions of the same students, making the algorithm a perfect suit for the problem, even if the posture is not exactly the same between consecutive frames.

The map is a Python dictionary having as keys the ID of a student taking part in the experience, and as values the coordinates for the keypoint 1 (center of torso) for that student on one of the first extracted frames. That particular point was chosen because it can be seen on the body of every participant. It is created by running OpenPose on that selected (judging if people are sat down at their place as early as possible in the video) frame and mapping manually the returned coordinates to the ID from the students provided in the student-to-ID document (Fig. 2). The algorithm that uses the map is the following: for every set of keypoints (or equivalently, every student) in a frame, it isolates the coordinates of the center of the torso, and compares them with every torso coordinates stored in the map. It then chooses the closest pair (using Euclidean L_2 distance) and assigns the key associated to that pair to the whole set of keypoints.

This allows to bridge the distance from the file containing the attention levels to the images containing the positions. Once every student is linked to an ID for every frame, the next step is to find the corresponding attention level directly in the excel file. This is easy to do because exactly one attention level is encoded by person by frame, and each frame yields one set of keypoints for every student, making it a one-to-one relationship.

B. Multiples camera link

As seen in Fig. 2, for a given camera, not every visible student level of attention is encoded. This happens because every student is only examined once every ten seconds, but may be visible in more than one camera. This is both positive and negative. It provides us with more data, since one student with one level of attention has more than one set of keypoints, This is the beneficial part, but the flip side is that it requires more work to organize the data to be able to profit from this. The map mentioned in section IV-A is, in the most simple case, used only for a given camera. The trick used here is to create a list of maps, corresponding to each camera (in order), and record the camera number in addition to the keypoints set given back by OpenPose. For every set of keypoints, it is straightforward to find the corresponding map thanks to this additional recorder number.

C. Data Normalization

An interrogation that surfaced very early in the project is the fact that the seating (the place he/she occupies within the classroom space) of the student matters to the Neural Network, as it works with coordinates. This could create problems for the predictions if a person sits at a place nobody has ever sat before. The same observation holds true for the distance from the camera, ie a student further away from the camera appears smaller, and its attention level may prove more difficult to predict if fewer data from this use case was provided. Last but not least, OpenPose does not detect the keypoints from all students equally. For certain people and certain frames, it is able do see or guess the position of the legs. While this could be seen as useful information, it was judged not important enough to keep over the fact that it would confuse the Neural Network if only part of the samples had this additional data.

Thus, data normalization is needed. It is composed of three distinct phases. The first step is to remove every keypoint related to the lower body. As shown in Fig. 4, OpenPose is not able to pinpoint the location of the legs keypoints, as they are often hidden behind tables and such. We remove them as in Fig. 6.

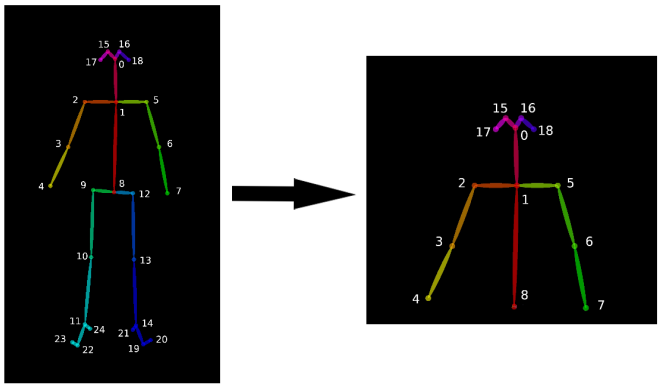

Fig. 6. Keypoints kept after first step of Normalization

Next, the set of keypoints is re-sized. This is achieved by isolating a set of keypoints representing a student, and computing the minimal and maximum values for each dimension, resulting in $x_{min}, x_{max}, y_{min}, y_{max}$. The difference between the minimum and maximum on each dimension is computed, and the biggest is stored as the *ratio*.

$$ratio := \max((x_{max} - x_{min}), (y_{max} - y_{min})) \quad (1)$$

Every coordinate in that keypoints set is then divided by $ratio$, which reduces the total size to the one of a unit rectangle. The length of the rectangle has size 1, and the other side is smaller than 1, since *ratio* is the biggest difference. This re-sized rectangle is contained in a unit square, which is a standard component to convert to in Computer Graphics.

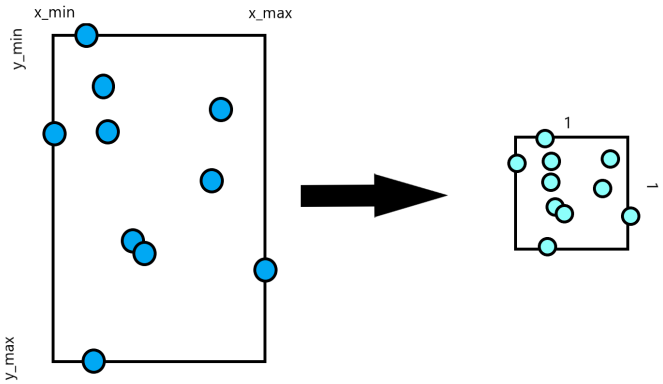

Fig. 7. Re-sizing the cloud of keypoints composing the body

Once the points have been re-sized, the next step is to center them around the $(0, 0)$ origin. This removes the location variable mentioned earlier.

The whole process allows to remove some components that do not influence attention levels, like student location in the classroom, from being processed by the Neural Network. Thus, the predictions on new images do not depend on these variables.

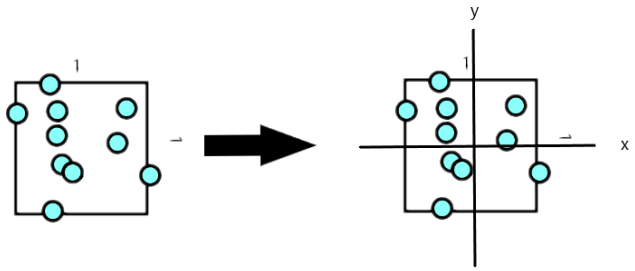

Fig. 8. Centering the cloud of points aroung the origin

D. Linear interpolation and Data Augmentation

The starting dataset size for this project was not big at first. However, an observation can be made: attention levels are recorded from the videos only once every 10 seconds, effectively not using all the other frames and information contained in them. Taking the frames immediately before and after the one where the attention level was recorded could lead to overfitting, because the keypoint coordinates would not change very much. Instead, let's think about how attention is represented in this project. It is a number on a linear scale from 1 to 10, sampled every 10 seconds. It would not be far fetched to think that attention is continuous, ie that to go from 3 to 7 on the scale, you need to go through to 5. Focusing is something gradual, there are very few instances where you can get razor sharp focus in an instantaneous manner.

Now that this is established, an **hypothesis** is made: attention is linear. To clarify, if a student is at attention level 3 at frame t, and at attention level 5 at frame $t + 1$ (10 seconds later), then he is at 4 in the middle, exactly 5 seconds away from frame t and from frame $t + 1$.

$$y^{t+0.5}_{interpolated} = \frac{y^t + y^{t+1}}{2} \quad (2)$$

This allows to basically double the size of our dataset. The videos are sampled twice as often, resulting in frames that are 5 seconds apart in the videos, and the attention levels go through the following process: for every attention level y, compute the mean between y and the next attention level for the same student (except for the last, because it does not have a next y). This creates a linear interpolation, and computes the corresponding attention level (based on the hypothesis) for each frame newly extracted.

While the hypothesis may be true, it is best not to try and abuse it. It could be possible not to limit the algorithm to a frame and an attention level every 5 second, but run it in iterations to create data for every second of video. While this would be beneficial for the size of the dataset, it would abuse the hypothesis. The way it is used still offers room for the attention not to be completely linear, and only creates an expected value in the middle of already existing values. To fix attention level values every second would introduce error. This negative aspect outweights the benefits of enlarging the dataset, hence this interpolation has been kept to only one iteration.

V. Results

A. Training Set and Validation Set

A Validation Set is used in Machine learning to simulate the prediction capabilities of the algorithm when faced with new inputs, different from the ones it has been trained on.

The easiest Validation Set that can be created in this project is to simply take 10% images, and store them separately from the other 90%. The percentages can of course be modified to suit different needs. The 90% is the Training Set, and the remaining is the Validation Set. However, this is not a good approximation of a brand new input, as the Network has already seen the people in the Validation frames, only in slightly different positions.

A much more representative approach is to separate Training Set and Validation Set by participant ID. A certain number of people are taken apart from the others, like they were missing in the pictures, and stored in the Validation Set. This way, the students that are used as Validation have never been seen before by the Network, making the Loss and Accuracy (defined in section V-B) closer to a real-world situation. To make this as random as possible, the IDs that will compose the Validation Set are taken randomly from the pool of IDs that took part in the experiment.

B. Loss and Accuracy Definition

To be able to measure performance of the Neural Network, two metrics have been defined. The **Loss** will refer to the distance between the attention level ground truth and the predicted value for the same attention level. This distance, or Error, is computed using the Mean Absolute Error. Essentially, it takes the difference between the ground truth and the predicted and computes its absolute value. Of course, the Absolute Error for only one value has very little significance, the mean over the whole Validation Set makes more sense. The second metric is one named **Accuracy**. It represents, as a percentage, the number of correct attention levels the Neural Network was able to predict. A prediction is judged correct if

$$y_{true} - 0.5 < y_{pred} < y_{true} + 0.5 \tag{3}$$

and false otherwise. Computed over the entirety of the Validation Set, it gives us an estimate of how many predictions were correct. The interval value 0.5 is a modifiable parameter. Its default value is, however, 0.5, which makes sense because it maps the continuous value that is the output of the Neural Network into one of the 10 possible attention levels used as labels. For example, a value x such that $5.5 < x < 6.5$ will be mapped to 6, which is simply a rounding function. The bigger the interval is made, the more error room it allows for the predictions, by being less strict on the truth value interval.

C. Results quantified

The best results currently attainable with the current state of the project are represented in Fig. 9 and Fig. 10.

The best Accuracy obtained is 57.6% and the minimal Error is 0.706.

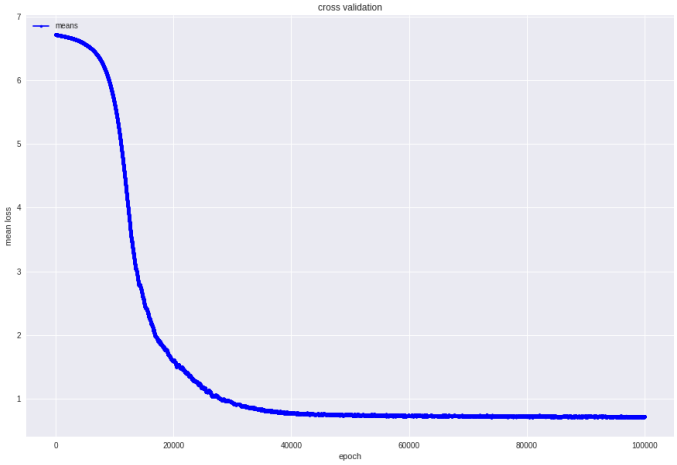

Fig. 9. Validation Loss over epochs

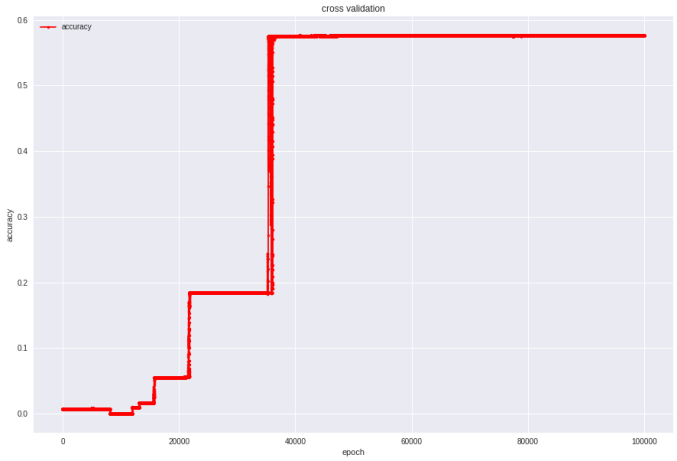

Fig. 10. Accuracy over epochs

D. Hyper-parameters values

To obtain the results just discussed, the following hyper-parameters have been used:

TABLE II
HYPER-PARAMETERS

Name	Value
Learning Rate λ	$1e^{-7}$
Number of epochs trained	10^5
Batch size	512
Optimizer	Adam
Loss Function	Mean Square Error

E. Comparison to other studies

The following comparisons have to be made with a thought in mind. The scale used to quantify the student's attention level and judge the accuracy of the result is not the same as in the other studies. The dataset is also different. This means that the results are not strictly comparable, a mere indication is made instead.

In the thesis of Mirko Raca [1], the attention is classified into one of three categories: *low*, *medium* and *high*. A score

of 60% accuracy is achieved. It also uses the gaze direction and head travel of the students. Janez Zaletelj and Andrej Koir [2] use both facial and body properties of a student, including gaze point and body posture. The attention is again divided into three categories, and the final accuracy is 75%. Kinect Face Tracker [4], including heart rate, achieves an accuracy of 74% on a binary (engaged vs. not engaged) attention scale.

As already said, the results are not directly comparable, but because the scale and the metrics used for this project are much less permissive than the ones used by the cited authors, it would be reasonable to believe that the performances are comparable. If the room for error (the 0.5 value presented in section V-B) is allowed to be bigger, the accuracy of the results would be significantly higher. An advantage of this project is however that the dataset is relatively smaller when compared to other studies.

F. Final output

The study resulted in the constitution of a more graphical and representative output, to be more understandable for the maybe less-technical psychologists that provided the data. It is simply representing the predicted attention level for a given student on the top of its head, along with a red rectangle around its head.

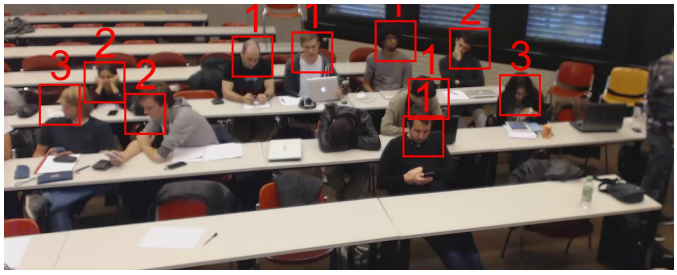

Fig. 11. Final Visual output

It allows more immediate and visual interpretation of the results for a single image.

VI. What's next

This project can be taken further. The easiest way would be to get more data, and thus enhance the size of the dataset, surely making the Neural Network more robust. This would require more data to be manually labelled, and would be costly. However, it would most certainly improve the Network performance.

Another approach to the whole project would be to use the images given back by OpenPose with the keypoints directly in them. This would require a Convolutional Neural Network, thus needing more hardware resources. This approach could yield better results.

A less costly alternative development could be to combine this project with the Thesis from Mirko Raca [1], and use both the keypoints from OpenPose and the gaze direction used in that study. This combines the approach from this project, focused on the posture and position from the students, and their gaze, since it is a non-trivial data when dealing with classroom attention. Both datasets compliment each other, and would certainly benefit from being processed together accordingly.

VII. Conclusion

The results achieved show that there lies an answer to the question asked in the introduction. Computers can detect attention levels using machine learning. The final Error of 0.706 and the Accuracy of 57.6% are interesting and suggest that there is something to develop. While I have not travelled all the way down the trail, I believe I have at least shown that it is not a dead end. As mentioned in section VI, the next couple of steps are already laid out, and should show significant improvement over the current performances. While it may not be able to produce significant results on the 1-to-10 scale given by the dataset, if the output categories are reduced in number, it can already yield interesting results. This would allow for very time-consuming tasks, like labelling the videos for further research and studies, to be automated, and gives a nice and fulfilling purpose to this project. The application domain is very diverse: a MOOC can get direct feedback about knowing if a video is boring/uninteresting/too difficult to understand and try to correct it accordingly, a psychology study can get enormous amount of data to try and test an hypothesis, a beginner teacher can have his/her classes recorded, and monitor the mean level attention of the class, to know which moment of his/her class were good, which deserve to be reworked, etc. A few of these possibilities are already doable with the current performance.

This project was my first time working on a research-oriented subject. I really enjoyed the process of solving difficulties as they arose. Is is so different than any other class I had until now, and it inspires me to maybe become a PhD one day. I learned a lot about PyTorch, one of the most popular Deep Learning Frameworks nowadays, and simply about Deep Learning in general. Having a hands-on project like this really helped me understand core Machine Learning concepts, while developing my own coding habits for dealing with Deep Learning problems. Because of these various reasons, this has been one of my favorite projects in my whole EPFL student career.

References

[1] Mirko Raca (2015) *Camera-based estimation of student's attention in class*

[2] Janez Zaletelj, Andrej Koir (2017) *Predicting students attention in the classroom from Kinect facial and body features*

[3] Jacob Whitehill, Zewelanji Serpell, Yi-Ching Lin, Aysha Foster, Javier R. Movellan (2012) *The Faces of Engagement: Automatic Recognition of Student Engagement from Facial Expressions*

[4] Hamed Monkaresi, Nigel Bosch, Rafael A. Calvo, Sidney K. D'Mello (2008) *Automated Detection of Engagement Using Video-Based Estimation of Facial Expressions and Heart Rate*

[5] Mandy Hommel (2012) *Kodierhandbuch des Beobachtungsinventars zur systematischen und videobasierten Erfassung der Aufmerksamkeit von Lernenden*

[6] Klaus Jrgen Ehrhardt, Peter Findeisen, Gloria Marinello, Hiltrud Reinartz-Wenzel (1981) *Systematische Verhaltensbeobachtung von Aufmerksamkeit bei Grundschlern wrhend des Unterrichts*