Projects for ETSIST Incoming Students -

FINAL STUDENT REPORT

Student data	
Name: Vandenbroucke Thybris	Enrollment num.:
E-mail: thybris.vandenbroucke@alumnos.upm.es	
Sending institution: Vives Brugge (Belgium)	

Project description	
Project name: Emotion detection for the Blexer middleware	
Academic year: 2023-2024	Semester: ◯ Winter ◯ Spring
Starting date: 01/09/2023	Ending date: 15/12/2023
UPM Centre: CITSEM Campus sur	Hours per week: 21
Total hours: 273	Credits: 9
Supervisor data:	
Name: Martina Eckert	
Department: Audiovisual and Communication Engineering (DIAC)	
E-mail: martina.eckert@upm.es	Phone:

The final report must begin with the project information specified in Annex I. After that, a project report must follow, whose structure should be as specified next. A length of one page per section is recommended, except for section 3, for which recommendation is two pages.

Report Contents

1. INTRODUCTION

The objective of my internship was to develop an application that would enable the prediction of emotions through facial expressions. This application was intended to become a part of a larger project called Blexer (Blender Exergames), which consists in a system that comprises several games designed to assist people in rehabilitation and encourage enjoyable physical activity. The game parameters can be adjusted by the physiotherapist at a distance via a web platform, to achieve personalized playing experiences and improve the efficiency of the rehabilitation exercises. Now, the system will be enhanced with AI algorithms, such that the adjustments could be made automatically. Therefore, different input data are needed, one of those is emotion information.

The application's primary purpose was to assess whether individuals would experience genuine happiness while performing various tasks within the game. To achieve this, it was essential to connect the application with the previous work of my colleagues, which was accomplished through the use of a middleware. This middleware facilitated the connection between human movements captured by a depth image camera (Kinect) and the in-game actions required to control the avatar in the game.

In carrying out this task, I leveraged my prior knowledge of machine learning, which I had acquired over the past two years. I collaborated closely with a doctoral student named Xiya, under the supervision of Professor Martina Eckert, in order to successfully complete this project.

2. CONTEXT

Over the course of the past five months, I had the opportunity to work at Universidad Politécnica de Madrid (UPM). My work was primarily centered at their advanced research center known as CITSEM, which stands for the "Centre for Software and Multimedia Systems Technologies for Sustainability Research."

Within CITSEM, I became a part of research group called GAMMA, which stands for the "Group on Acoustics and MultiMedia Applications."

One of the achievements of the GAMMA research group is the development of a comprehensive platform known as Blexer. This platform was designed with a specific purpose in mind – to assist healthcare professionals in their efforts to monitor and support patients more effectively. It achieves this by incorporating physical assessments into a dynamic and engaging video game environment.

The Blexer project, along with my work, fell under the careful oversight and guidance of Professor Martina Eckert.

3. PROJECT OBJECTIVES AND TASKS

To begin with, I was tasked with exploring all the possibilities of Kinect One and Azure Kinect for emotion detection through facial expressions. Initially, I decided to focus my efforts on the newer hardware, namely Azure Kinect, for the development of this program.

After some time and reaching out through online forums, I received assistance from a Microsoft Azure software expert. However, I was informed that Microsoft had decided to discontinue support for emotion detection. This revelation led me to halt further exploration of the Azure Kinect portal.

This prompted me to seek out a new approach, and I initiated my own project using Python. My first challenge was to find a clear and suitable dataset for my application.

Once I had acquired the dataset, I needed to create a program to transform it into a usable model. The most significant challenge here was determining the type of neural network required and how to set it up.

Furthermore, it wasn't sufficient to just have a neural network; I also wanted to track the degree of improvement or deterioration after each training session.

Once this part was completed, I had to develop an application to utilize the model. For this purpose, I employed a library to access the camera and detect when a face was present on the screen. If a face was detected, the model would intervene and determine the prevailing emotion, providing the result accordingly.

Finally, my goal was to integrate all these components seamlessly with the provided middleware. I opted to run the Python script asynchronously with the middleware to ensure they could both operate simultaneously.

4. TECHNOLOGIES AND EQUIPMENT

The system comprises two key components: The Model Training part implemented in **trainingplus.py** and the Python application using the trained model, which is implemented in **videotester.py.** These components work cohesively in tandem with Middleware adjustments facilitated by MainWindowViewModel.cs and MainView.xaml.cs. Figure 1 shows how they all come together.

In the following, each part will be described in detail:

In the Model Training part (trainingplus.py), a comprehensive approach to training a deep learning model for emotion detection is established. It begins with the importation of essential libraries, including OpenCV for image handling, TensorFlow for machine learning, and Keras for constructing and training the neural network. Notably, it employs Keras-specific functions for building neural network layers, such as layers and models, as well as ImageDataGenerator for augmenting image data and custom callbacks like EarlyStopOnLR. Data preparation involves configuring the dataset path and managing emotion categories, where the focus is on five specific emotions to enhance model performance.

Image preprocessing simplifies the complexity by converting images to grayscale and resizing them to a uniform 72x72 pixel size. This grayscale conversion reduces computational load and standardizes input data. Data normalization scales pixel values to a [0, 1] range for improved gradient management, while one-hot encoding transforms categorical labels into a suitable format for multi-class classification. The dataset is then split into training and testing subsets for unbiased model evaluation. To enhance model robustness and generalization, data augmentation techniques are applied.

Convolutional Neural Network (CNN) architecture is meticulously designed, incorporating convolutional layers for feature extraction, batch normalization for stability, max pooling for dimensionality reduction, dropout for regularization, and dense layers for classification with L2 regularization to prevent overfitting. The model is compiled with the efficient Adam optimizer, categorical cross-entropy loss function, and accuracy as the evaluation metric. Training callbacks, including EarlyStopping, ReduceLROnPlateau, and a custom callback, EarlyStopOnLR, are introduced to control the training process based on specified conditions. During model training, augmented data is utilized, exposing the model to a broader range of data scenarios. The training process is also monitored with validation data to ensure model

generalizability. Following training, model evaluation encompasses prediction, a confusion matrix, and visualizations of accuracy and loss over epochs. The trained model is then saved for future applications, allowing for reusability and further fine-tuning if necessary. A confirmation message signifies the successful completion of training and model saving.

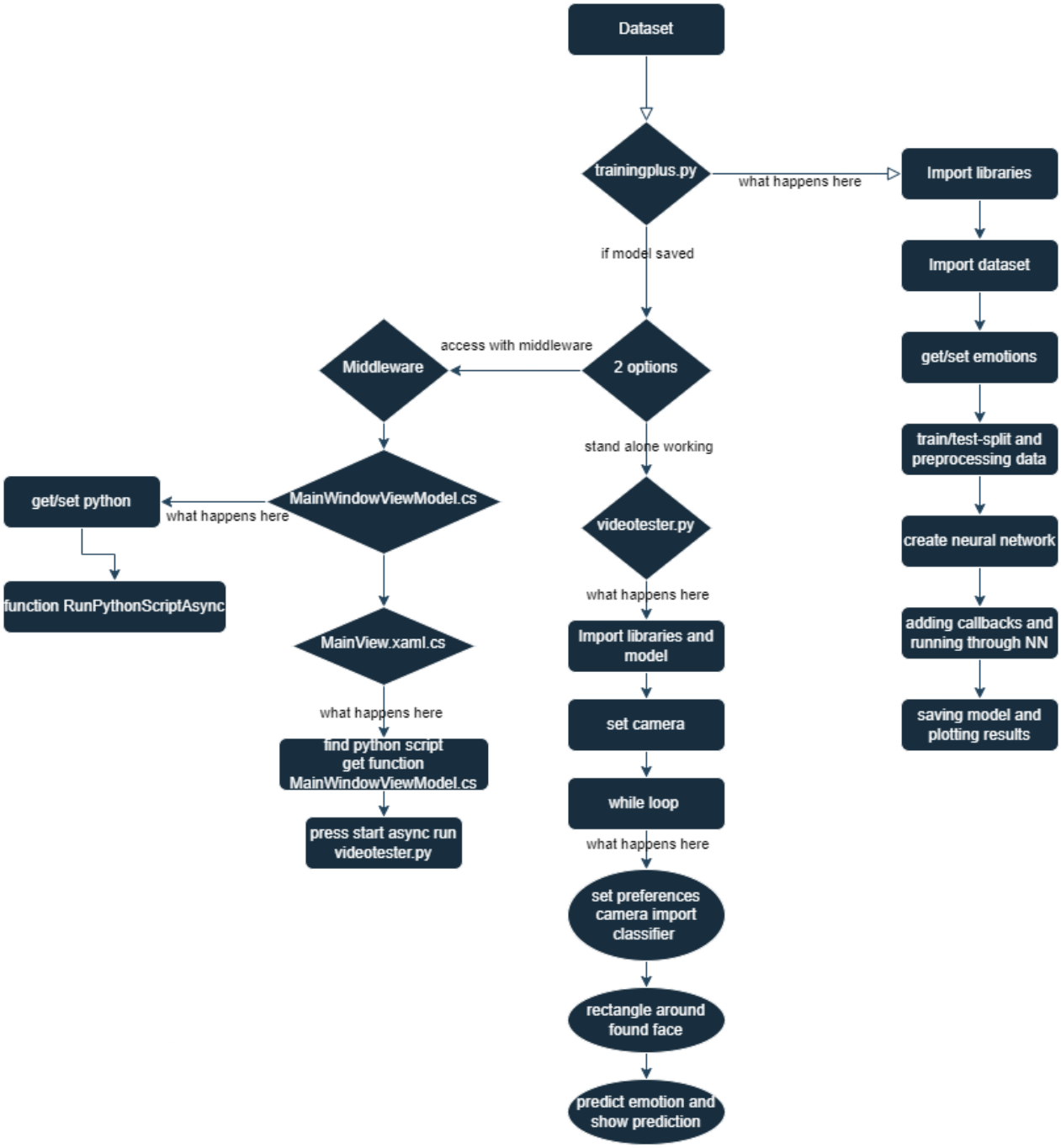

Figure 1: Workflow Diagram

In the Python application using the model (videotester.py), the pre-trained deep learning model is loaded.

OpenCV is employed to capture video from the webcam, process video frames, and display the results in real-time. The script initializes the webcam, checks its readiness, and enters an infinite loop to continuously capture frames. Each frame is converted to grayscale to simplify data processing. Face detection is performed using a Haar cascade classifier, and the first detected face is processed, resized to 72x72 pixels, and pixel values are normalized. Emotion prediction is carried out using the pre-trained model, and the predicted emotion is drawn on the frame. The processed frame with emotion information is displayed continuously until the user presses 'q' to exit. Upon exit, the webcam is released, and OpenCV windows are closed.

The Middleware adjustments (MainWindowViewModel.cs and MainView.xaml.cs) provide the means to run Python scripts asynchronously from the GUI. This includes the management of Python executable and script paths, ensuring error handling and the ability to execute Python scripts without blocking the UI thread.

In summary, this integrated system enables the creation and training of an emotion detection model and its real-time application through a Python script that captures webcam video, processes it using the trained model, and displays the detected emotions. Middleware adjustments facilitate smooth interaction between the GUI and Python script execution, offering a cohesive user experience.

For future work:

- I encountered a challenge in making all file paths variable in the project. A potential solution could be converting the trained model from its current .h5 format to the .onnx format, which might integrate more seamlessly with the C# program.
- Currently, there's an issue where both the middleware and the videotester.py script are attempting to access the camera simultaneously, leading to an error where the Python file cannot access the camera. A possible workaround could be to create two separate display streams: Screen A could display the body model, while Screen B could show the face detection model. This approach would allow the camera to be activated just once but still display two different types of data. Additionally, this strategy could help address the issue of having a static path for the python.exe execution.
- Also maybe review the resizing of the photos. I enlarged them, but I'm not sure if this is an appropriate adjustment to make. The dataset photos were originally 48x48 instead of 72x72. Additionally, a modification could be made in the middleware to allow not only the use of Kinects but also the use of just the webcam.

5. DEVELOPED COMPETENCES AND SKILLS

Engaging in this project was a milestone as a student specializing in artificial intelligence. It represented a contribution to my understanding of machine learning, particularly convolutional neural networks (CNNs), and the practical application of this knowledge. This project served as a comprehensive exercise to test and solidify my fundamental understanding of CNNs.

Through this, I gained insights into the intricacies of building a functional project in Python, encompassing data collection, model training, and deployment. It provided me with a hands-on opportunity to connect various components and modules, integrating datasets, training algorithms, and inference mechanisms.

Overall, this project was instrumental in bridging the gap between theoretical knowledge and practical

implementation. It not only expanded my understanding of machine learning but also equipped me with the skills to tackle real-world challenges in the field of artificial intelligence.

6. CONCLUSIONS

This project has been a profound learning experience that has greatly enriched my knowledge and skills in several key areas:

- Machine Learning Expertise: Through the creation of an emotion detection model, I have deepened my understanding of machine learning techniques, particularly Convolutional Neural Networks (CNNs), and their application in real-world scenarios.
- Data Handling and Preprocessing: Acquiring and integrating datasets, performing data preprocessing, and refining the training data have honed my abilities in data management—a crucial aspect of any machine learning project.
- Python Programming: Extensive work with Python, including scripting, debugging, and library usage, has solidified my proficiency in this versatile programming language.
- Problem-Solving Skills: I have encountered and overcome various challenges during the project, cultivating my problem-solving skills, adaptability, and resilience in the face of obstacles.
- Project Management: Managing a complex project involving research, data acquisition, model development, and integration has improved my project management skills and the ability to meet deadlines.

Supervisor's Influence:

- The role of my supervisor has been instrumental throughout this learning journey:
- Guidance and Mentorship: My supervisor provided invaluable guidance, offering insights and direction when navigating complex technical decisions and project milestones.
- Critical Feedback: Constructive feedback from my supervisor allowed me to refine my approach, make necessary adjustments, and enhance the overall quality of the project.
- Encouragement: My supervisor's encouragement and support were motivating factors, inspiring me to persevere through challenges and strive for excellence.
- Knowledge Transfer: The supervisor's expertise in the field was a valuable resource, enabling me to leverage their knowledge and experience to address project-specific issues.

In conclusion, this project has contributed to my growth as an artificial intelligence student. It has expanded my practical skills, deepened my understanding of machine learning, and fortified my problem-solving abilities. Moreover, the guidance and mentorship of my supervisor have been pivotal in shaping this learning experience.

7. PROJECT LOG

Week 1 and week 2: Extensive research into the capabilities of the Azure Kinect device. Seeking to understand the existing solutions and projects in this domain.

Week 3 and week 4: Developing a program to connect with the Azure network for emotion detection using

the Azure Kinect. A setback occurred when Microsoft discontinued their emotion detection services.

Week 5, 6 and 7: Searching for a suitable dataset and creating a Python script to train on that dataset, including displaying the accuracy. Subsequently, I developed a program for live usage of the dataset within the application I needed.

Week 8: I integrated my developed solution into the existing middleware. This involved ensuring that my model could seamlessly run alongside the middleware, enabling real-time emotion detection during its operation.

Satisfaction Questionnaire

Please, quantitatively evaluate your satisfaction with each of the following project aspects according to the specified scale:

5: excellent, better than expected
4: remarkable, above average
3: adequate, acceptable
2: with some limitations
1: deficient

ASPECTS	**1 – 5**
The time duration of the project was adequate.	4
The project has been adequately planned.	4
The attention and support provided by the supervisor were satisfactory.	5
Social interaction with colleagues and other people involved in the project was satisfactory.	4
Project execution has been as planned.	3
Project activities have been relevant to my training.	5
I expect the project activities carried out to have provided me with useful training for my prospective professional activities.	4
The project has been useful for the lab, department, research group, etc..	4
My training background helped me in developing project activities.	4
The administrative support I have received during the project was adequate.	4
OVERALL SATISFACTION WITH THE PROJECT	41

SUGGESTIONS
Before I started I was still not sure that this was a course or internship, it would be good to highlight in the description for foreign students, that it is practical work or a project.

Date:

Signature: