ML-BEHAVIOR CLASS PARTITIONING: AN EQUIVALENCE CLASS PARTITIONING-INSPIRED APPROACH TO ML TESTING

Timothy Elvira , Jaxon Selzer , Juan Ortiz Couder and Omar Ochoa

Department of Electrical Engineering and Computer Science, Embry-Riddle Aeronautical University, Florida, United States

ABSTRACT

Currently, the rise of software engineering, specifically requirements specifications, are being implemented in the design and workflow of ML applications, namely MLOps. Software Engineering has many untapped techniques, in both validation and verification, which could carry over into the ML context, providing a well-engineered approach to eveloping ML models. One such technique is Equivalence Class Partitioning (ECP) which is a method of testing wherein testing a single test is valid for an ECP class, typically derived from a functional requirement. This paper explores a framework for deriving requirements, equivalence classes, and testing to develop a new form of equivalence class portioning, named ML-Behavior class partitioning to identify behavior of an ML model. This paper outlines the necessary changes of altering the traditional-SE ECP to accommodate the non-determinism and stochasticity of ML. To test the ML-Behavioral Class Partitioning testing (ML-BPC), an off-the-shelf YOLOv8 model is tested to examine behavior. Requirements are derived for the model using five image transforms: gaussian blurring, elastic distortion, translation, rotation, and brightness. The object is to identify the ML model's behavior when incrementally increasing these image transforms, using ML-Behavioral Class Partitioning to identify the limits of the object detector by testing and fulfilling corresponding requirements.

KEYWORDS

Software Engineering, ML Engineering, Verification, Machine Learning, Object Detection

1. INTRODUCTION

Requirements Engineering for Machine Learning is a growing field of research, with non-functional requirements growing in interest and functional requirements remaining relatively unexplored [1, 2]. There are many facets to Requirements Engineering, and more broadly, Software Engineering, including validation, verification, and elicitation. In Machine Learning (ML), testing plays a critical role in determining whether a model has acquired the desired learned behavior and assessing its ability to generalize. This is typically evaluated using a validation set during the training and testing phases. While Software Engineering offers numerous verification techniques, many remain underutilized in the context of ML.

This paper explores the implementation of one such verification technique, Equivalence Class Partitioning (ECP). To conduct this investigation, an off-the-shelf YOLOv8 model, a popular and stable object detection model, is used to test and analyze ML behavior. Notably, the YOLOv8 model has not been explicitly trained on the specific image transformations applied in this implementation. By examining how the model responds to these transformations, the modified

 DOI:10.5121/ijci.2026.150301

ECP testing strategy aims to evaluate its generalization capabilities. The image transformations used in this study include Gaussian blurring, elastic distortion, translation, rotation, and brightness adjustments. Through testing the model on these controlled transformations, the effectiveness of ECP as a verification technique for ML models is further assessed by evaluating the number of requirements that the model successfully fulfills.

The motivation for this paper is to explore a traditional SE-testing technique and to investigate the modifications required to implement the traditional testing strategy into the context of ML. Isbell et al. outline the necessity of exploring software techniques in the ML space to attain the benefits of software engineering techniques while developing ML applications [3]. Investigating these techniques and the tradeoffs of implementing them in the ML space provides a better understanding of how these techniques change to accommodate ML's stochasticity and non-deterministic nature. While the domain of SE-based ML techniques and concepts is growing in diverse directions, investigating these concepts can help refine and advance these emerging techniques. Furthermore, continued academic interest can further develop these techniques by expanding upon or challenging the strategies implemented in this paper, ultimately contributing to the continued refinement of the field.

The paper contains the following sections. The background contains information on verification, specifically ECP and its logic. The background also contains information on object detection, the model under evaluation using the modified ECP methods. Approach outlines the framework and the experiment with the YOLOv8 model as well as example requirements from the YOLOv8 model. The results section shows examples of the ML-Behavior class partitioning testing (ML-BCP) and example visualizations. The related work section reviews prior research relevant to this study and highlights this paper's contributions to the field. The future work outlines improvements and further experiments. Finally, the authors give their final remarks in the conclusion.

2. BACKGROUND

This background section covers verification, software testing, equivalence class partitioning, and some object detection, all required to understand the contents of this work.

2.1. Software Verification & Testing

Software verification falls under software quality assurance, which focuses on ensuring that software meets customer requirements, satisfies relevant standards, and is engineered reliably. Software quality encompasses many parts of software engineering such as Software Testing, Software Requirements, Software Design and Software Maintenance, which is encompassed in the Software Engineering Body of Knowledge (SWEBOK) [4]. One aspect of Software Quality is Verification and Validation (V&V), which are two distinct aspects of Software Quality. Verification is ensuring that the product matches the intended design while validation is ensuring that the product meets the customers' goals [5, 6].

- Verification: answers the questions "Are we building the product right?"
- Validation: answers the question "Are we building the right product?"

V&V activities are used jointly during the software development life cycle to ensure a high-quality software product is not only built correctly but also meets customer expectations. In the case of this paper, verification is the focus. Some examples of verification activities are reviews and walkthroughs, which are means of stepping through the code and analyzing its functionality

against derived requirement specifications [7]. Furthermore, another activity of verification is testing techniques and strategies, such as ECP.

Software Testing is any activity which is aimed at evaluating the capability of a program to determine if it meets required standards. Typically, these standards are derived from requirements specifications. Ideally, the goal of testing is to identify defects, which improves the quality of the system [8]. Within Software Testing, there are two types of testing opacity: black box and white box. Black box, also known as functional testing, is when the code is not revealed to the tester with white box having access to the code that is under evaluation [9]. Black box testing is strictly based on the behavior of the code with the choice of test cases being strictly based on requirements [10]. Some types of black box testing are boundary value analysis and ECP [11].

2.2. Equivalence Class Partitioning

The As mentioned, ECP is a form of black box partitioning. ECP aims to divide input data of software into equivalence classes, which are derived from requirement specifications [12]. ECP is most effective when applied to scenarios with a range of input fields. Once these input fields are divided into ECP classes, only a single test per class is required. This is based on the theory that all inputs within an ECP class will exhibit the same behavior, making additional tests within the same class redundant [13]. ECP is a type of test case selector which is useful in identifying test cases but also reducing the number of test cases, such as not implementing ECP would lead to needing to test every value within the input field.

- ECP: The theory of ECP states that if you test any input within a class and it is valid, then all inputs in that class are also valid. Conversely, if one input within a class is invalid, then all inputs in the class are considered invalid, indicating a fault in the code.

An example of ECP is as follows: Suppose a system which takes in a user provided input which the value is expected to fall within the valid range of 30 to 60. If the input is outside this range (invalid), the user is directed to an error message stating “Invalid range.” If the input is within the valid range, the user is taken to a link which announces them as a winner. The table below details the ECP rules.

Table 1. ECP rules for example problem.

Equivalence Partition Classes		
Invalid	Valid	Invalid
<30	30-60	>60

When the user inputs 45, they are directed to the winner's announcement, indicating that the system works as intended, and the class is valid. If the user inputs 15 or 95, the system correctly redirects them to the "Invalid range" page, as those inputs fall outside the valid range. Any unintended behavior, such as entering 45 and triggering an “Invalid range” message, would indicate the presence of a bug. Ultimately, only three tests are required to properly implement ECP: one for each of the three ECP classes—below 30, between 30 and 60, and above 60. These tests are sufficient to validate the behavior of the system across all possible input ranges.
ECP can have visualizations of the implemented testing strategy [13]. A single variable ECP would resemble a line plot with a two variable ECP resembling a typical scatter plot. An example of each is shown below in figure 1 & figure 2.

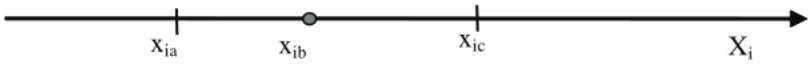


Figure 1. The generic set up of an equivalence class partition test. The point Xib is the test point which is randomly selected.

Figure 1 shows that the Xib point is a test for that equivalence class. Figure 2 shows the two-dimensional visualization, which has two variables, X1 and X2.

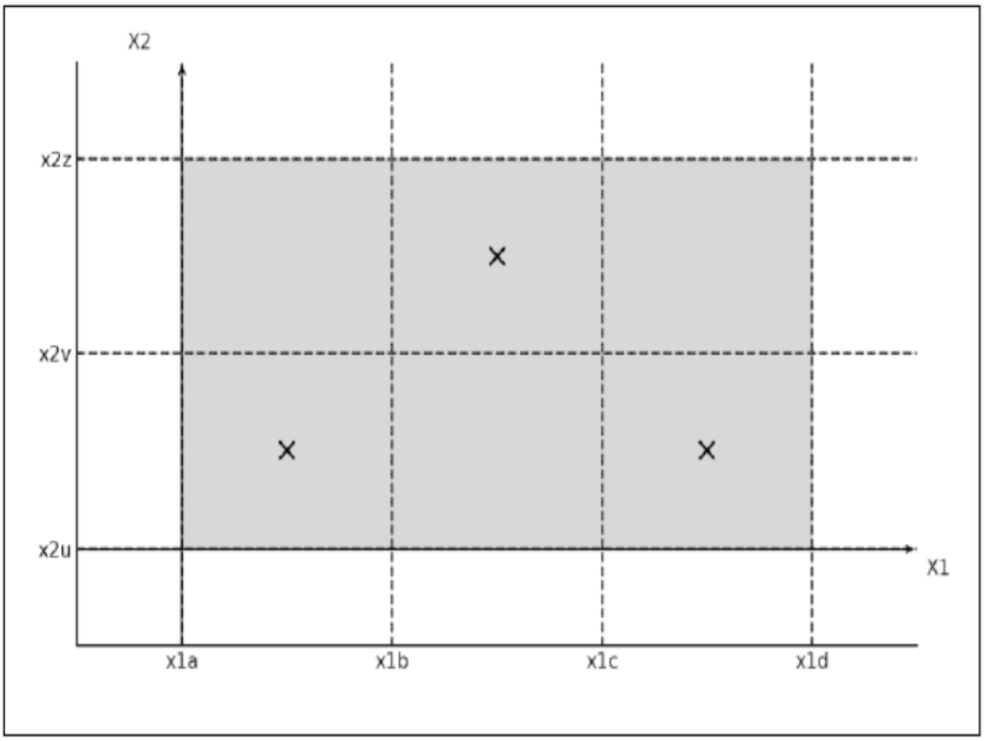


Figure 2. The two-dimensional ECP visualization with two variables.

2.3. Machine Learning, Object Detection, & YOLOv8

As stated previously, ML is a subset of Artificial Intelligence which is the field that aims to enable computers to mimic and exhibit human-like intelligence [14]. The field of AI began with Alan Turing who posed the question: “Can Machines Think?” and surmised the Turing Test to assess a machine’s likeness to a human [15]. Over the years since Turing proposed the Turing test, AI has evolved through extensive research in mathematical, computing, and algorithmic to create a specialized branch of AI called ML. ML is the study of algorithms and statistical models which computers use to perform specific tasks. Unlike traditional programming, where humans do define the rules and logic, ML models generate their own logic through learning from data and experiences thus are not explicitly programmed by a programmer [16].

ML models can be categorized by their learning paradigms or tasks. ML is often organized based on the paradigm of learning: supervised learning, unsupervised learning, and reinforcement learning. Alternatively, ML can also be categorized by learning tasks such as classification, regression, clustering, and dimensionality reduction [17]. Learning tasks refer to specific learning

tasks or objectives that an ML algorithm is designed to accomplish. The learning paradigms define how algorithms learn from data while the learning tasks refer to the specific problems the algorithms attempt to solve [18]. Understanding these groupings is crucial for accurately modeling your data and selecting an appropriate ML algorithm.

One such learning task is object detection, which focuses on generating bounding boxes around desired object classes. Object detectors typically use specialized architecture designed to detect and learn object features, generating inferences that include bounding boxes and the detector's confidence in the object's presence. These can be generated from Convolutional Networks (CNNs), however, recently Fast Regional CNNs (R-CNN) have been better at object detection and tracking [19, 20, 21]. R-CNNs operate using a two-step process: first, they identify regions of interest and apply bounding boxes, and then they classify the detected objects. This process, however, is relatively slow and computationally intensive. In contrast, the "You Only Look Once" (YOLO) model streamlines this approach by combining region identification and classification into a single, faster process [22].

Because YOLO streamlines the process, it is considered a one-stage detector. YOLOv8, which came out in 2023, is an advanced real-time object detector capable of generating bounding boxes and inferences within 60 frames per second [23]. YOLOv8 is still a favorable model for its fine-tuning capabilities, ease of use with the Ultralytics code base, and its superb performance in object detection [24]. Figure 3 depicts the YOLO model using both bounding boxes and confidence while simultaneously calculating a class probability map to generate bounding boxes for an image.

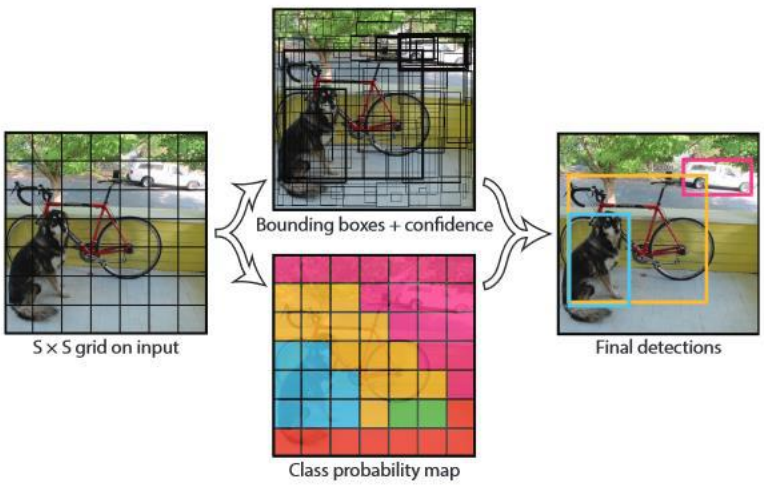


Figure 3. A simplified view of the original YOLO's one-stage object detection [33].

3. PROPOSED FRAMEWORK & IMPLEMENTATION

This section outlines the proposed framework which outlines the process of implementing the ML-BCP testing. Furthermore, this section outlines the approach to implementing the ML-BCP with an off-the-shelf YOLOv8 model to detect behaviors with requirement specifications.

3.1. ML-BCP Testing Framework

The To implement the traditional ECP-inspired ML testing framework, there are nine steps to generate the ML-Behavioral Class visualizations, which look identical to the visualization in Figure

2. These nine steps are as follows and shown in Figure 4.

- Identify Scenarios: These are scenarios to identify the variables in which a model might experience, which affect the model's ability to generate successful inferences.
- Identify Raw Requirements: These represent the model's expected performance in various scenarios, typically defined within an abstract range, such as Gaussian blurring from 0 to 20.
- Separate Raw Requirements into Behavioral Classes: These are behavioral classes where behavior can be considered the "same." Ideally, these classes correspond to smaller ranges that fit within the larger range defined in the previous step, such as Gaussian blurring ranges of 0–2, 2–4, 4–6, and so on up to 20.
- Generate Refined Behavioral Requirements: These are specifications which are directly connected to test cases, used to "fulfill" and identify model behavior.
- Generate Test Cases: These test cases are values derived from the requirements and are used to generate the visualizations in the final step.
- Generate Test Cases Dataset for Each Behavioral Class: This is where ML-BCP diverges from ECP, to generate a small test dataset, instead of a single random point. This approach accommodates ML by avoiding reliance on a single point, which could be an outlier, to verify behavior and fulfill requirements. The ML-BCP value for a class is the averaged performance for that class, using a metric defined in the generate behavioral requirements stage.
- Test ML Model: This process generates inferences for the ML model and calculates metrics such as mean Average Precision (mAP), accuracy, or even confidence. Any valid metric can be used, provided it is relevant to the requirement specification.
- Record Test Case Results: This is to generate the average performance which is to be visualized.
- Generate ML-Behavioral Class Visualization: Finally, generate a visualization to identify the successful or failed behavior of the ML-Model.

3.2. Implementation

The As mentioned, this framework is tested through an implementation using an off-the-shelf YOLOv8 model which is trained on the COCO dataset. The model is only tested with a single object

class under evaluation, the "Person" object class. The five transforms are again as follows: gaussian blurring, rotation, brightness, translation, and elastic distortion. Gaussian blurring is meant to implement a blurred effect on the image. This blurred effect is caused by a Gaussian Kernel. This image transformation is meant to mimic focus levels or even atmospheric effects of fog. Rotation is to rotate the image around a certain point. This affects objects within images by presenting them in different angles to which the object detection model might not have encountered during the training set. Brightness adjusts pixel values by a certain factor to either increase or decrease the image's brightness. Translation is just the slight movement, in terms of pixels, of an image offset from the corners of the image pane. Finally, elastic distortion is a subtle transformation which is used to implement variations in a camera lens that is bent or flexible, distorting the subject. Elastic distortion involves applying random, non-linear deformations to an image, mimicking the way elastic materials might be stretched or squeezed [25].

Table 2. The Range for each transformation, step size, and number of tests given the step size and range.

Transformation	Range	Step Size	Num tests
Gaussian Blur (σ)	(0,20)	2 σ	10
Brightness Intensity	(-30,30)	6.0	10
Rotation	(-50,+50)	10 degrees	10
Translation	(-50,+50)	10 pixels	10
Elastic Distortion(α)	(0,50)	5 α	10

Following the framework, behavioral requirement specifications are needed to generate the test cases and test datasets. Table 2 below shows the ranges for each image transformation as well as the increments for each ML-BCP class. Furthermore, each transformation is treated as a single variable and then paired with another, giving two variables. Overall, since there are ten increments for each transformation, there are a total of 100 classes for the two variable ML-BCP visualization.

The equation below shows the combination of the two variable visualizations, showing ten unique combinations. This is relevant to identify the combinations, plus the total number of two-variable requirement specifications.

$$C(5,2) = \frac{5!}{2!\,(5-2)!} = \frac{5!}{2!3!} = \frac{5x4}{2x1} = 10$$

With each image transformation containing ten tests. There is a total of 100 tests for each two-transformation visualization. One visualization encapsulates 100 requirement specifications. The requirement specifications are for both the single variable and double variable. An example of them is as follows. The full set of requirement specifications, consisting of 50 single-variable requirements (5 transforms with 10 tests each) and 1,000 two-variable requirements (10 combinations with 100 tests each), can be found in the provided GitHub repository. Single and double transform requirements are made for each of the image transformations for the “Person” object class and the ranges from Table 2.

Single Transform Requirement: The YOLOv8 shall detect a "Person" class under a gaussian blur with a standard deviation between 0.0 to 2.0 with a mean confidence of at least 0.6.

Double Transform Requirement: The YOLOv8 shall detect a "Person" class under a gaussian blur with a standard deviation between 0.0 to 2.0 and a rotation of -50.0 to -40.0 degrees with a mean confidence of at least 0.6.

To generate these images for model testing, transformations are applied using the OpenCV Python library. A seed dataset of n=235 is used, which is taken from the COCO validation dataset. This seed dataset is used for every ML-BCP class with each transform or transform pair (such as rotation and translation) applied for every range. The images are input to the YOLOv8 model where the confidence is obtained. Finally, the performance of the dataset is averaged and applied

to the test case. It is important to note that the YOLOv8 model is not explicitly trained on these transformations. Therefore, this implementation aims to examine the model's behavior in terms of its ability to generalize. The complete code is available in the provided GitHub link. Overall, a single object class can correspond to 1,050 behavioral requirements for five image transformations.

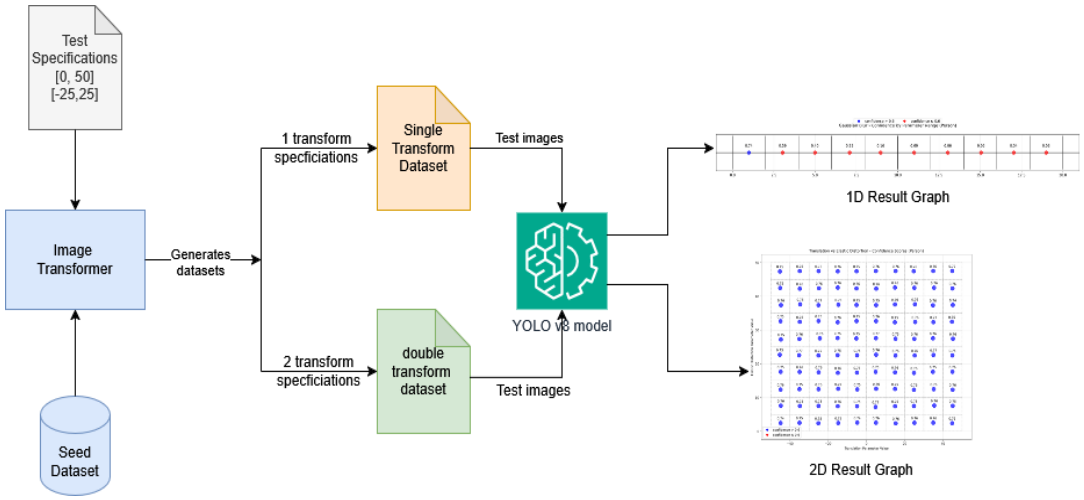


Figure 5. The YOLOv8's entire pipeline from data generation to test. In the case of the double transform, the image transformer would simply apply another transformation with specifications.

4. RESULTS

This section outlines the results. Due to the number of visualizations, 5 single variable visualizations and 10 two-variable visualizations, only a few examples are discussed. The results section also discusses the number of requirements verified for the "Person" class, as that is the evaluation metric for the technique. The entire list of requirements and visualizations for the person class are on the GitHub repository. For reference, the YOLOv8's performance on the seed dataset, without any transformations, is 0.862027 average confidence. Confidence is selected instead of mAP due to the lack of adjusted bounding boxes for transformations that may alter the position of the object class, such as rotation or translation. A future experiment is to include mAP. It is worth noting that these specifications would be customer-defined and could encompass any range of performance metrics. The same applies to the 0.6 threshold, which is arbitrary and used here solely to illustrate the concept. In practice, the threshold would be customer-derived and would directly influence the acceptance criteria for model behavior.

4.1. Single-Variable ML-BCP Visualization

The single-variable ML-BCP involves applying only one image transformation. Each point on the visualization corresponds to the average performance of a seed dataset with values within that BCP range. For example, when applying random brightness values within a specified range to the entire seed dataset, the transformed images are input into the YOLOv8 model. The model's performance is then evaluated and averaged across the dataset based on the chosen metric, which is derived from the requirements.

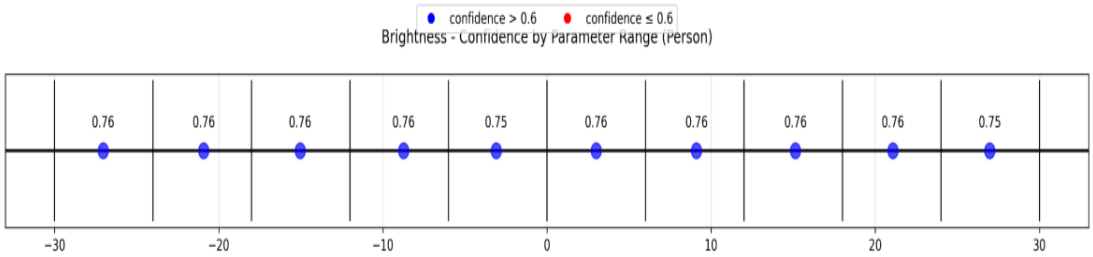


Figure 6. The YOLOv8's performance on the Brightness Transformation, with around 0.76 to 0.76 average confidence for each ML-BCP Class.

Shown in Figure 6, the model shows that brightness is not a transformation which negatively affects model performance to the point where requirements are unfulfilled. Furthermore, increasing the brightness only diminishes the performance by 0.01. Additionally, transformations that exhibited similar behavior across their complete range of requirements, such as translation and elastic distortion, demonstrate consistent performance patterns. Rotation and gaussian blurring exhibit weaker behavior from the YOLOv8 model.

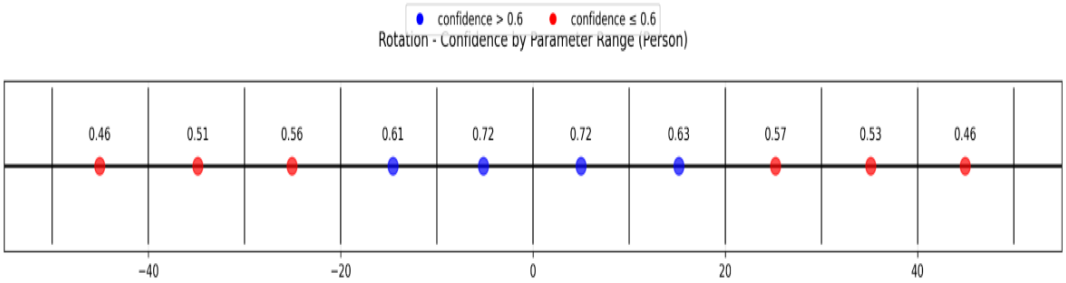


Figure 7. The YOLOv8's performance on the Rotation Transformation, with the successful tests in the ±20-degree rotation range.

Figure 7 illustrates the performance of the rotation transform, where failed requirements begin to appear beyond the ±20-degree range. Beyond this threshold, the model shows reduced performance at higher degrees of rotation. Based on successful tests and fulfilled requirements, it is reasonable to conclude that the model can perform effectively within a ±20-degree rotation range. With the requirement threshold being 0.6 average confidence for the classes' seed dataset, the model begins to fail at higher or lower rotation than ±20 degrees. The performance degradation is not linear and degrades at a larger rate than 20 to 30 degrees versus 40 to 50 degrees.

4.2. Double Variable ML-BCP Visualization

Two-variable ML-BCP visualizations have 10 total visualizations; however, only 4 are shown to show a set of behaviors that can be gleaned from these visualizations. The first visualization is of two model-resilient transformations, discovered to be resilient in the single variable ML-BCP testing: brightness and elastic distortion. Figure 8 shows that even if two transformations increase in intensity, the model remains resilient to both image transformations applied.

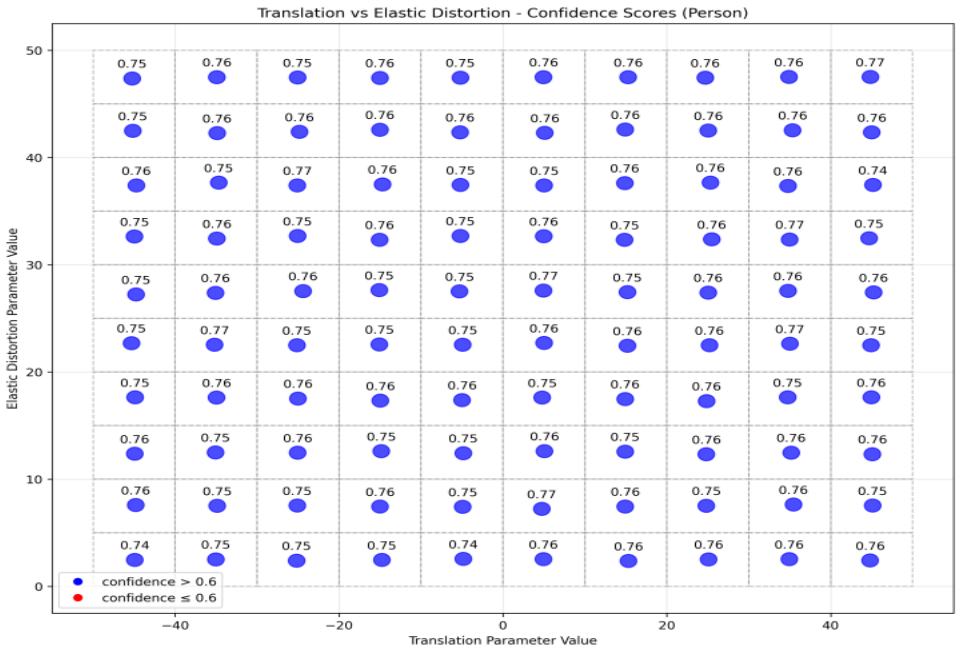


Figure 8. The YOLOv8's performance on the Brightness vs. Elastic Distortion Transformation, showing no degradation when experiencing two image transformations.

Figure 8 is when a model-resilient transformation, translation, is combined with a model-weak transformation, or rotation. This demonstrates interesting behavior where the model is on the verge of exceeding the previously identified ±20-degree rotation threshold, with certain areas extending into the ±25-degree range under specific translation values. However, the model's performance degrades to its weakest level, with an average confidence of approximately 0.46, in the most extreme ranges of the transform.

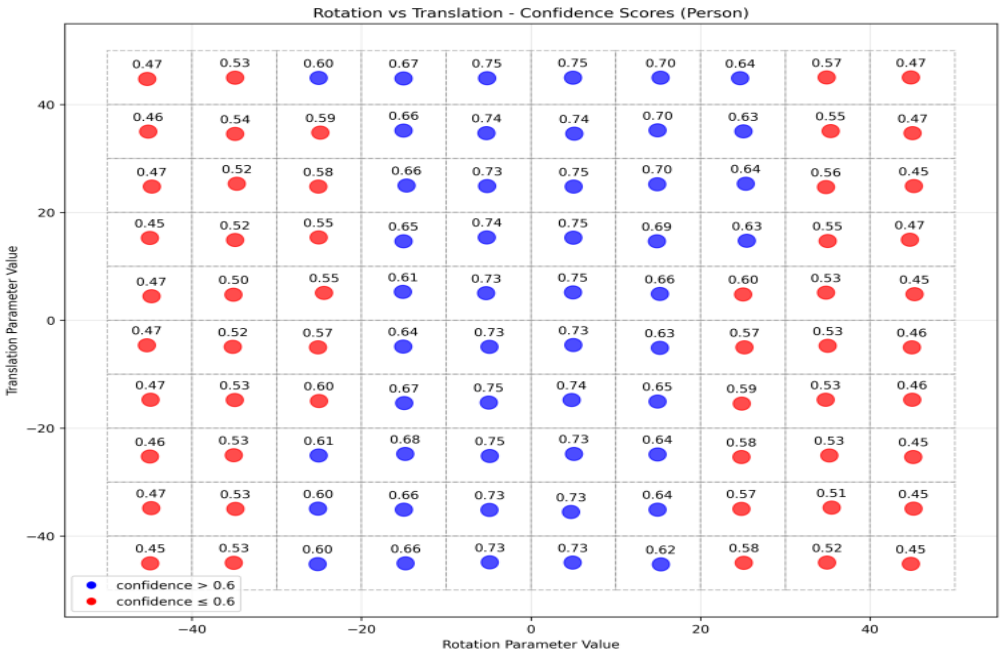


Figure 9. The YOLOv8's performance on the Rotation vs. Translation Transformation, showing similar degradation of behavior as rotation's single variable counterpart

Now a model-resilient transformation, brightness, is paired with the model's weakest and most vulnerable transformation scenario, Gaussian blurring. In Gaussian Blurring's single-variable ML-BCP visualization, the YOLOv8's behavior begins to fail outside of acceptable requirement range around 0-2 σ. In Figure 10, similar behavior is observed as in Figure 9, where the model occasionally meets requirements in ranges previously considered areas of weaker performance. This suggests the presence of an amorphous threshold where the model can still produce somewhat acceptable inferences. It's important to note that in some areas, a confidence score of 0.6 is considered acceptable, while in others, it is not. This discrepancy arises from rounding, where a value slightly below 0.6 is rounded up to 0.6, while successful tests may be rounded down to 0.6. Despite this, the behavior remains valid, as each test is manually classified as passed or failed before rounding occurs for graphing purposes.

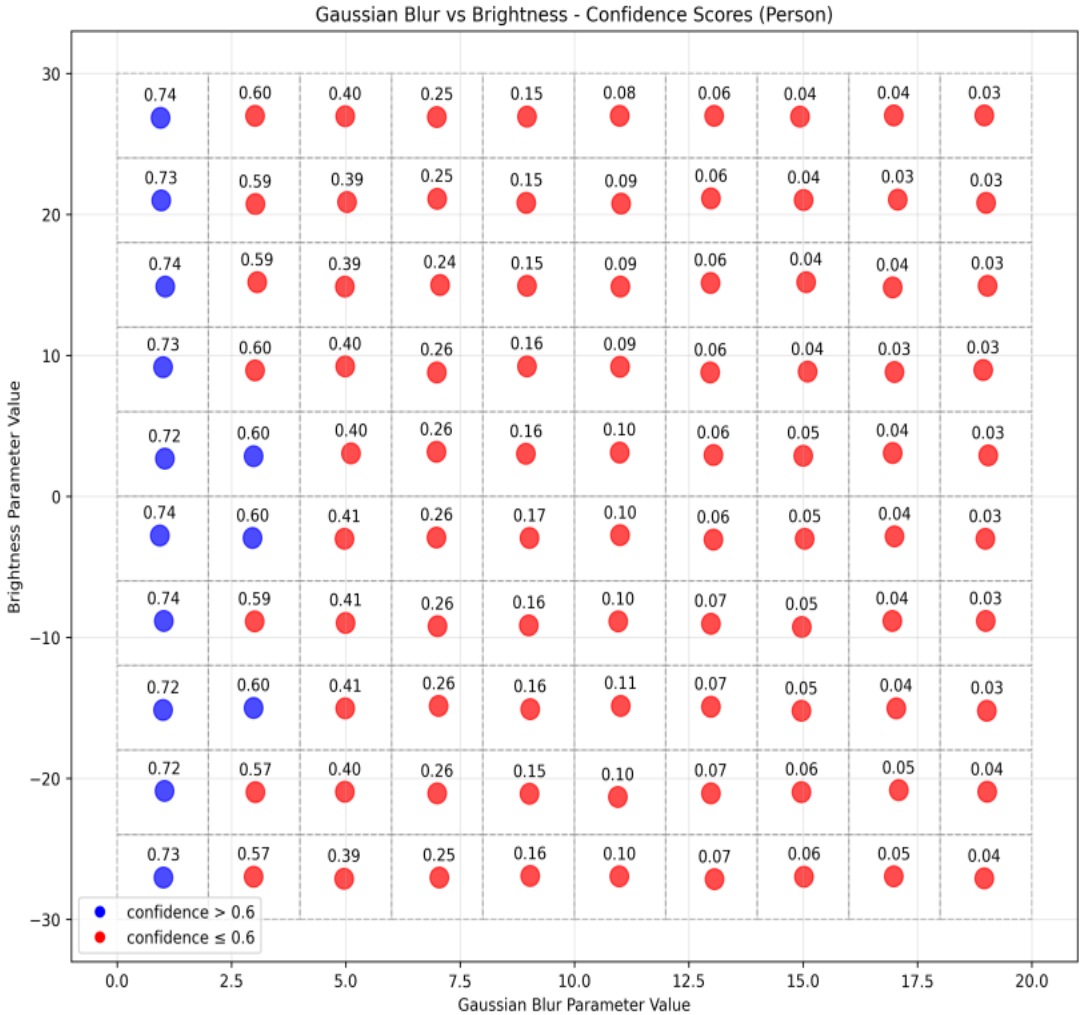


Figure 10. The YOLOv8's performance on the Gaussian Blurring vs. Brightness, showing the model can extend successful inferencing into the range thought to be failing, 2-4 σ

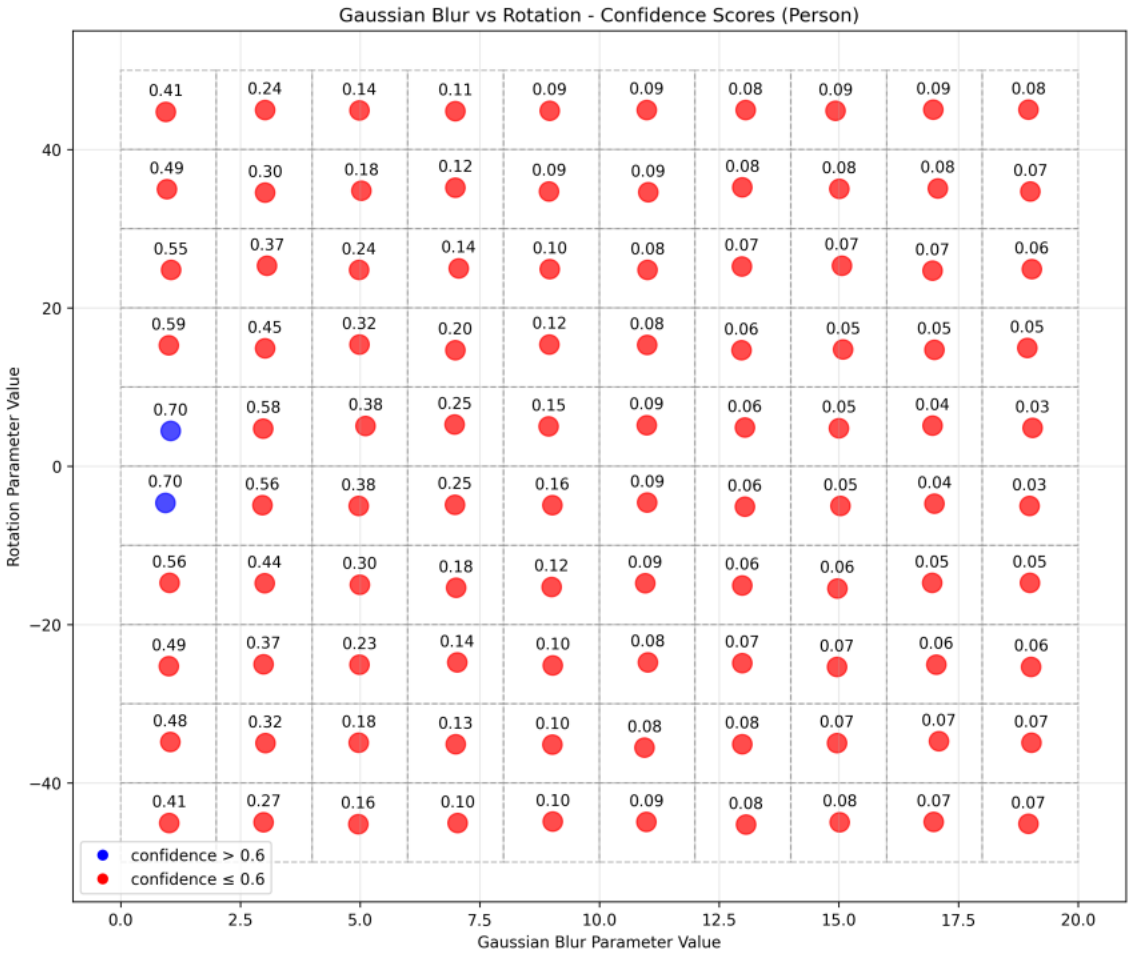


Figure 11. The YOLOv8's performance on the Gaussian Blurring vs. Rotation, showing the model is vulnerable to experiencing rotation and Gaussian Blurring

Figure 11 is the final type of behavior gleaned from the ML-BCP visualizations. This figure is the combination of rotation, a model-weak transformation, and then gaussian blurring, a model-vulnerable transformation. This visualization depicts the model's worst performance, with only two passing requirements, both involving low levels of Gaussian blurring and rotation.

Overall, this testing method is linked to requirement specifications, as it is in traditional software engineering. Ideally, testing would fulfill successful requirements with unfulfilled requirement specifications driving the creation for the dataset to improve behavior. Furthermore, if retraining is undesirable, the requirements reflect the scenarios in which a model can produce successful inferences according to requirement specifications.

4.3. Evaluation

Section To evaluate this method, the number of requirements fulfilled across the five single-variable and ten double-variable ML-BCPs are aggregated. For the "Person" class, a total of 500 out of 1,050 requirements are fulfilled, indicating that the off-the-shelf model successfully meets 47% of the total requirements. As traditional ECP is a form of test case selector and generator, the number of test cases, or individual tests generated per ML-BCP class is recorded. For ECP single transform, each class was the size of the seed dataset or n=235. This results in 2350 images

for a single individual ML-BCP class. Since there are 100 classes in the two-variable visualization, this results in 23,500 per visualization. In total, there are 11,750 images generated for the "Person" class, covering all the five image transformation. In terms of the two variable visualizations, the total number of images augmented are 235,000 images, totaling 246,750 for this work. The complete evaluation table for the "Person" object class is shown in Table 3.

Table 3. Complete Evaluation Metrics of the "Person" Class for YOLOv8, trained on COCO, using ML-BCP.

	Evaluation Statistics for "Person" Class for ML-BCP				
	Total Requirement	Requiremnts tested	Requirements Fulfilled	Test Cases Generated	Visualizations Generated
ECP – 1 transform	50	50	35/60	11,750(2350 x 5)	5 (10 Reqseach)
ECP – transforms	1000	1000	465/1000	235,000 (23500 x10)	10(100 Reqseach)

Overall, the method presented in this paper can fulfill requirement specifications; however, it may require refinement in both the ML-BCP method and the elicitation of the requirement specifications themselves. Ultimately, this work remains an open challenge, but the author hopes that it will encourage further investigation and implementation of other software engineering verification methods, with the goal of reinvigorating ML verification through inspiration from traditional software engineering.

5. RELATED WORK

In terms of implementing software practices within the ML context, there has been limited progress in adapting and augmenting testing methods. However, there is considerable work in the implementation of requirements, particularly non-functional requirements (NFRs), with more modest efforts focused on functional requirements (FRs). This paper emphasizes the importance of requirement specifications, as ML-BCP, much like its traditional software engineering counterpart, relies heavily on well-defined requirements.

In terms of NFRs, Horkoff et al. established the beginning workings of an NFR quality model, identifying non-functional requirements with varying changes to their definitions [26]. Horkoff et al.'s work was expanded by validating the new list of requirements with industry developers who are involved in applications with ML, identifying and verifying their derived list of non-functional requirements [2]. Additionally, another work improved on Horkoff et al.'s work by clustering the NFR's into 6 separate clusters, grouping them based on their definitions and relevance to each other [1]. In a similar manner to Horkoff et al. Vogelsang & Borg performed a systematic literature review for Requirements Engineering for AI (RE4AI) from top RE conferences and journals, identifying 2,048 papers but filtering them down to 26 relevant works [27]. The authors conclude that ML systems lack requirement specifications and suggest building reference quality models to align novels, emerging NFRs. Furthermore, Vogelsang & Borg emphasize the understanding of performance measures to implement adequate functional requirements. With that in mind, the functional requirements found in this work utilize confidence, albeit not the best metric for an object detector; however, a metric that is still usable for a functional requirement.

With respect to FRs and testing, there is modest work; however, no work aims at leveraging software engineering testing methods from the FRs. A paper by Farell et al. examines possible functional requirement patterns for ML software from a NASA project [28]. An example of these patterns is shown in Figure 12. It's important to note that these are just derived patterns and have not been used in an experimental setting in the paper. Farrel et al. aggregate other requirements by Gauerhof et al. and Hawkins et al. who produce safety requirements for self-driving vehicles [29, 30]. It's important to note that the functional requirements from Hawkins et al. do link to test cases; however, not derived from traditional SE strategies as this paper does.

Overall, the field of incorporating functional requirements into ML is expanding. This work builds on the use of requirement specifications by applying a traditional software engineering verification technique, ECP, and adapting it to function within the context of an ML model, specifically an object detection model. The primary contribution of this paper is the investigation of traditional SE methods, and the necessary modifications required to accommodate the non-determinism and stochasticity inherent in ML. However, this is only the starting point for integrating SE verification techniques into ML, and further experimentation and case studies in real-world scenarios are needed before establishing this as a reliable verification method.

6. FUTURE WORK

There are several areas where this method can be improved. One is generating bounding boxes to utilize mAP, which would provide a more comprehensive metric for evaluating object detection. Additionally, this could serve as an interesting case study to observe the degradation of confidence compared to mAP, a metric that incorporates both confidence and ground truth bounding boxes. Another area for future improvement is incorporating non-functional requirements. Ongoing efforts to define a quality model for NFRs align with this work, which is primarily performance-based—a recognized NFR. Expanding this approach could address additional NFRs, such as robustness, performance, and model trust, offering a more comprehensive evaluation of ML models. It's important to note that this work does not expand on the validity of the ML requirement specifications, verifying the requirements used in this work with an ML Engineer would be valuable to ensure they are appropriately elicited and testable using this method. Additionally, validating the ML-BCP method with the same ML Engineer would provide further assurance of its effectiveness in verifying model performance across various scenarios.

For further experimentation, incorporating this testing method into an industry setting could introduce an AGILE development approach, potentially refining these methods to become more practical and effective in meeting high-demand customer requirements. Another area to explore is the scenarios themselves, as they are not limited to image transformations, which were used merely as proof of concept. Semantic transformations, such as modifying the count of object classes by increasing the number of objects in an image, represent one possible direction for further investigation, among many other potential avenues.

One rapidly growing domain is Large Language Models (LLMs) which are models aimed at understanding and generating natural language [31]. An experiment could involve identifying an evaluation benchmark and introducing perturbed or convoluted language to assess a model's ability to understand and generate content in such scenarios. One example would be implementing a benchmark for multilingual language processing by gradually increasing the percentage of input text in one language over another [32]. While this method is currently applicable to object detection models, further investigation is needed to determine the adaptations that ML-BCP would require to accommodate large models with billions of parameters—if such adaptations are even feasible.

CONCLUSIONS

In conclusion, this work explored and investigated the implementation of ECP testing in the context of ML. This new form of ML testing is called ML Behavior Class Partitioning Testing. This method aims to identify requirements which correspond to BCP classes, test them, and then identify successful behavior. A framework is developed in this work to outline the process of implementing the ML-BCP strategy. This framework and strategy are tested using a COCO-trained off-the-shelf YOLOv8 model. A total of 1,050 requirement specifications were generated for this model, focusing on its most represented object class, “Person.” Using ML-BCP, 500 of the 1050 requirements specifications are tested and found to be successful. The model from Ultralytics is found to be capable of producing successful inferences in brightness, translation, and elastic distortion; however, are weak to rotation and exceptionally weak to gaussian blurring. Ultimately, this is just an example implementation and the single and two variable ML-BCP can be implemented in a range of other scenarios or ML models. Future work includes implementing ML-BCP with more semantic based scenarios, such as multiple of the same objects and identify if performance degrades with the inclusion of multiple of an observed object class. Requirement specifications, and the testing which uses them, can be a powerful tool in identifying desired ML behavior.

Dance-based therapy is relevant here because it combines rhythm, cueing, coordination, balance, and enjoyment in one activity. Reviews of Parkinson's dance interventions have reported benefits in gait, balance, and motor symptom severity, suggesting that therapeutic dance can be both clinically meaningful and motivating when compared with doing nothing or with some conventional exercise programs [7][8][9]. The long-run problem, then, is not only symptom management. It is how to make rehabilitation accessible, repeatable, and appealing enough that people will continue doing it.

Dance-based rehabilitation aims to improve balance, gait, and motor performance through rhythmic, expressive movement. Its major strength is engagement, but it often depends on class access, instructor availability, and transportation. MirrorMove PD tries to preserve dance's motivational quality while making delivery more repeatable at home [7][8][9].

Home-based telerehabilitation focuses on access and continuity. It reduces travel demands and helps patients maintain exercise routines outside the clinic, but some programs are generic, minimally engaging, or highly dependent on supervision. MirrorMove PD improves this by making the home program more guided, dance-centered, and visually structured [10][11][13].

Exergaming and cueing systems attempt to improve movement by adding interactive feedback, rhythm, or immersive environments. These systems can be highly motivating, but the evidence remains mixed and some require specialized hardware. MirrorMove PD lowers the hardware barrier, though it still needs stronger scoring validation and broader clinical testing [14][15][16][17].

This project proposes a therapeutic dance rehabilitation prototype called MirrorMove PD that combines a Flutter mobile application, a Python desktop companion, and a local-network control bridge to deliver guided dance sessions and track user progress.

The solution addresses the problem by reducing friction between rehabilitation planning and rehabilitation execution. On the mobile side, the system handles sign-in, progress visualization, daily goal tracking, song selection, and session launching. On the desktop side, the system displays reference choreography beside the user's live camera feed and exposes local control

endpoints that the phone can access. Together, the two interfaces create a lightweight home-use workflow: the user opens the app, selects a session, connects to a nearby computer, and then follows a guided dance routine that is easier to repeat than a paper exercise sheet or a one-time clinic demonstration. The repository also contains pose-analysis research scripts that use MediaPipe landmarks to compare reference motion against observed motion, which gives the project a clear path toward more structured feedback in future versions.

This approach is a plausible improvement over more fragmented rehabilitation methods for three reasons. First, dance itself already has evidence supporting its usefulness in Parkinson's rehabilitation, especially for balance and motor symptoms [7][8]. Second, home-based and telerehabilitation approaches have shown that remote or at-home exercise can maintain meaningful benefits when in-person care is limited [10][11][13]. Third, the prototype joins motivation and measurement: it does not merely show videos, but also stores goals, progress snapshots, and session history in a way that can support adherence over time. The project is still a prototype rather than a validated clinical product, but as a design direction it is coherent, accessible, and better aligned with sustained daily use than isolated exercise instructions alone.

The experiments focused on two practical questions: whether users would trust the scoring system, and whether repeated use could improve short-term confidence in movement. The first approximate experiment compared self-rated performance with system scores and then asked participants to judge fairness. The most important finding was that ratings were generally positive when the score gap was small, but trust dropped sharply when the score differed too much from user expectation. The second approximate experiment tracked self-reported movement confidence before and after a brief repeated-use period. Here, confidence improved for every participant, and the largest improvement appeared in the participant with the highest session count. Together, the experiments suggest that user experience is shaped by both perceived accuracy and adherence. In other words, the system must feel fair enough to trust and enjoyable enough to repeat. Those two factors likely influence long-term rehabilitation usefulness more than novelty alone.

1. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

1.1. Improving Robustness in Pose Analysis and Move Validation

One major component of the program is the pose analysis and move validation system. A central challenge is that human motion captured by an ordinary webcam is noisy, incomplete, and sensitive to lighting, camera angle, occlusion, and body size. To handle this, the system could use landmark-based pose estimation and normalize comparisons so that the user's body is not unfairly penalized for standing closer to or farther from the camera. It could also compare poses over short time windows rather than on a single frame. If landmark confidence becomes too low, the application should reduce scoring weight or temporarily withhold judgment rather than provide misleading feedback.

1.2. Ensuring Reliable Mobile – Desktop Communication

Another major component is mobile-desktop communication. The program depends on a phone and a nearby computer exchanging state over a local network, which creates practical risks such as Wi-Fi mismatch, incorrect IP entry, dropped requests, or commands arriving in the wrong order. A robust solution could expose a small set of predictable HTTP endpoints for connection,

status, song transfer, and session commands, then pair this with periodic status polling so the phone always reflects the desktop's current state. The bridge should also send explicit song metadata and connection feedback so that session playback, song selection, and user expectations remain synchronized.

2.3. Balancing Workout Difficulty and Progress Tracking

The third major challenge is level design and workout metrics. Therapeutic sessions must be difficult enough to be useful but not so difficult that they discourage or fatigue users with Parkinson's disease. This means the project needs a structured way to represent duration, difficulty, and daily progress without pretending that one score fully captures rehabilitation quality. A practical solution could store per-user goal minutes, changes to those goals over time, current-day progress, and session history, then use those fields to build weekly summaries. It should also separate content difficulty from medical outcome claims, so the interface remains motivating without overstating what the numbers mean.

3. SOLUTION

The program is organized around three main components: a mobile app, a desktop app, and a local connection bridge. The mobile app is written in Flutter and initialized with Firebase services. It handles authentication, user profile creation, daily progress display, weekly workout history, song retrieval from Firestore, and the session flow that asks the user to connect to a nearby desktop device. The desktop side is written in Python and uses CustomTkinter for the local window, OpenCV for video handling, and Flask for a lightweight control server. It is responsible for showing the guided dance reference media, exposing the local connection code, receiving remote commands, and keeping track of which song is active. The bridge between them is a small HTTP-based protocol over the local network that supports `/connect`, `/disconnect`, `/status`, `/song_data`, and `/command`.

From start to finish, the flow is straightforward. A user signs in on the phone and lands on the home dashboard. The app loads songs and progress information from Firestore and lets the user open a session. During the startup session, the user enters a connection code or host for the desktop companion. The phone sends a connection request, then transmits song metadata and later playback commands such as previous, next, play, pause, difficulty, and volume. The desktop app responds with current state, including whether it is connected, which song is active, and whether playback is running. This allows the phone to act like a remote rehabilitation controller while the larger desktop display handles the guided visual experience. Supporting scripts in `analysis/` also use MediaPipe and landmark extraction to explore motion comparison and future feedback features [18][19].

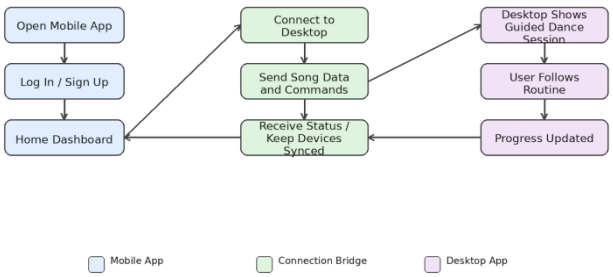
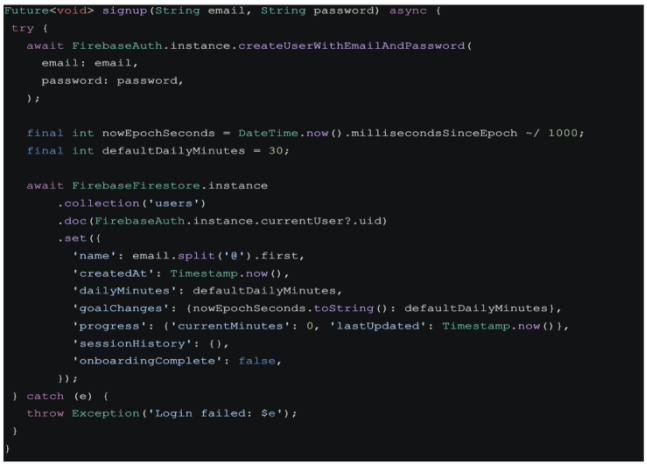


Figure 1. Overview of the solution

The mobile app is a user-facing rehabilitation hub. It uses Flutter for interface delivery, Firebase Authentication for account management, and Cloud Firestore for persistent user and song data. Its main purpose is to lower entry friction: sign in, select a routine, connect to the desktop, and review progress without technical overhead.

Figure 2. Mobile application architecture and user workflow overview



```
Future<void> signup(String email, String password) async {
 try {
   await FirebaseAuth.instance.createUserWithEmailAndPassword(
     email: email,
     password: password,
   );

   final int nowEpochSeconds = DateTime.now().millisecondsSinceEpoch ~/ 1000;
   final int defaultDailyMinutes = 30;

   await FirebaseFirestore.instance
       .collection('users')
       .doc(FirebaseAuth.instance.currentUser?.uid)
       .set({
         'name': email.split('@').first,
         'createdAt': Timestamp.now(),
         'dailyMinutes': defaultDailyMinutes,
         'goalChanges': {nowEpochSeconds.toString(): defaultDailyMinutes},
         'progress': {'currentMinutes': 0, 'lastUpdated': Timestamp.now()},
         'sessionHistory': {},
         'onboardingComplete': false,
       });
 } catch (e) {
   throw Exception('Login failed: $e');
 }
}
```

Figure 3. Firebase user initialization and Firestore schema setup

This method runs when a new user signs up through the Flutter application. The first step is standard Firebase authentication: the program calls `createUserWithEmailAndPassword`, which creates the account and makes the user available as the current authenticated identity. After that, the code seeds a Firestore document in the `users` collection. This is important because the rest of the application depends on rehabilitation-specific fields that do not exist in default Firebase auth records.

The variables `nowEpochSeconds` and `defaultDailyMinutes` establish the starting rehabilitation profile. `goalChanges` records the timestamped history of the user's daily goal, `progress` stores the current minute count and the last update time, and `sessionHistory` creates a container for workout records later. `onboardingComplete` is also initialized for future flow control. In practice, this snippet turns a generic account into an app-specific rehabilitation profile. Once this document exists, the home screen and progress components can derive today's minutes, weekly history, and daily goal summaries from a consistent schema.

The desktop app is the program's guided-session engine. Its role is to provide a large-screen rehabilitation experience that a phone alone would not communicate as well: reference choreography, camera-facing participation, and local playback control. It uses Python, CustomTkinter, OpenCV, and Flask. Conceptually, it acts as both a user interface and a lightweight local server.

```
@flask_app.route('/status', methods=['GET'])
def status():
   return jsonify({
       "connected": is_connected,
       "ip": get_lan_ip(),
       "port": SERVER_PORT,
       "currentSong": dict(current_song_state),
   }), 200

@flask_app.route('/song_data', methods=['POST'])
def song_data():
   data = request.json
   _register_song_metadata(data)
   video_filenames = os.listdir('videos')
   target_name = data.get("title", "")

   for filename in video_filenames:
       name_without_ext = os.path.splitext(filename)[0]
       if name_without_ext.lower() == target_name.lower():
           _set_current_song(name_without_ext, playing=False)
           command_queue.put(f"start_new_session:{filename[:-4]}")

   return jsonify({"status": "received"}), 200
```

Figure 4. Desktop Flask server routes for session control and state management

This code runs while the desktop companion is active. The `status` route reports whether a phone is connected, which IP and port the user should target, and which song is currently selected. The `song_data` route receives metadata from the phone, registers the song, finds the matching local video file, and places a command into a queue so the interface thread can start the correct session. This is a clean separation of concerns: Flask handles network traffic, while the UI thread remains responsible for the actual media state and display logic.

That separation matters in a rehabilitation context because playback controls, connection state, and song selection have to remain predictable. If networking directly blocked rendering, the user experience would feel unstable, and the guided session would be harder to trust or repeat.

The connection bridge is the glue between the phone and the desktop. Its purpose is not to store user data or render video, but to synchronize state. This component matters because the project would feel unreliable if the mobile interface said one thing while the desktop window did another. It therefore relies on repeated HTTP status checks, command posts, and song metadata transfer over the local network.

```
Future<void> _syncDesktopStatus() async {
if (_isFetchingStatus || dataService.desktopAppUrl.isEmpty) {
  if (dataService.desktopAppUrl.isEmpty) {
    _setWavePlaying(false);
  }
  return;
}

_isFetchingStatus = true;
final url = Uri.parse('${dataService.desktopAppUrl}/status');
try {
  final response = await http.get(url);
  if (response.statusCode != 200) {
    setState(() {
      isConnected = false;
      _isDesktopPlaying = false;
    });
    _setWavePlaying(false);
    return;
  }

  final payload = Map<String, dynamic>.from(
    jsonDecode(response.body) as Map,
  );
  final connected = payload['connected'] == true;
  final currentSong = payload['currentSong'] is Map
      ? Map<String, dynamic>.from(payload['currentSong'] as Map)
      : <String, dynamic>{};
  final isPlaying = connected && currentSong['playing'] == true;
```

Figure 5. Mobile–desktop synchronization loop via HTTP status polling

This code runs repeatedly during a session. It guards against overlapping requests, builds a `/status` URL from the saved desktop host, and polls the Flask server for current state. If the server fails, the app clears the connection indicator and stops the playback animation. If it succeeds, it decodes JSON, extracts connection and song state, and updates the screen. In other words, this is the synchronization loop that keeps the mobile controller aligned with the desktop rehabilitation display.

The bridge is therefore not just a convenience feature. It is the mechanism that turns two separate applications into one coordinated rehabilitation workflow. Without it, song control, playback awareness, and session feedback would all become fragmented.

4. EXPERIMENT

4.1. Experiment 1

A critical blind spot is user perception of score fairness. If users believe the motion score is arbitrary or inaccurate, they are less likely to trust the system or continue using it.

To test perceived fairness, a small pilot-style procedure can be used after a guided dance session. Each participant completes one session, receives a system score, compares it to a self-rated performance estimate, and then rates how fair the system score feels on a five-point Likert scale. This setup is appropriate because the prototype's immediate risk is not only algorithmic error, but also user confidence in the feedback. The control comparison is the participant's own judgment of performance, which is imperfect but useful for studying trust. The responses are collected through the following survey: https://forms.gle/ArJSVibjwZLLLxBH9.

Participant	Self-Rated Performance (0-100)	System Score (0-100)	Absolute Gap	Fairness Rating (1-5)
P1	82	79	3	4
P2	76	72	4	4
P3	88	90	2	5
P4	69	60	9	3
P5	91	86	5	4
P6	74	75	1	5
P7	83	70	13	2
P8	78	80	2	4

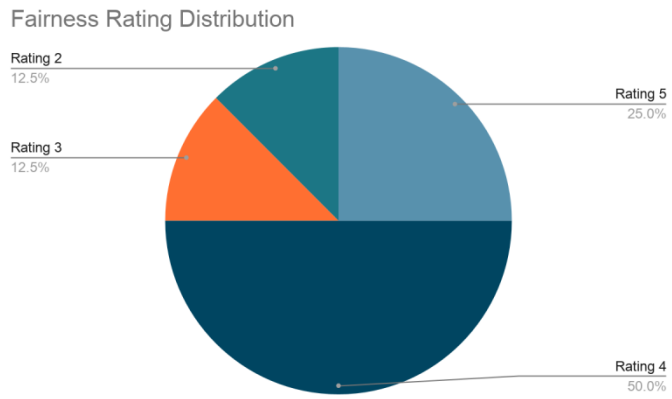


Figure 6. Comparison of self-rated performance and system-generated scores across participants”

The approximate results suggest moderately positive but not uniformly strong trust in the scoring output. The mean fairness rating is 3.88 out of 5, while the median is 4. The lowest value is 2 and the highest is 5. The average absolute gap between self-rated performance and system score is 4.88 points, with a median gap of 3.5, a minimum of 1, and a maximum of 13. The pattern is intuitive: higher fairness ratings cluster around smaller score gaps, while the one rating of 2 appears in the largest mismatch case. What is most noteworthy is that participants still rated several sessions as fair even when the score was not identical to their self-assessment. This implies users may tolerate some imperfection if the feedback is directionally reasonable. The biggest factor affecting perceived fairness appears to be large disagreement between user

expectation and machine output, especially when the system undershoots a participant's self-rated effort.

4.2. Experiment 2

Another blind spot is whether repeated use improves short-term confidence and willingness to move. If the system is engaging but does not encourage continued participation, its rehabilitation value remains limited.

The second approximate experiment measures self-reported mobility confidence before and after a short period of repeated home use. Participants are to complete between three and six sessions in one week. Before the first session and after the final session, they rate their confidence performing guided therapeutic movement on a ten-point scale. Session count is also logged to show whether better outcomes align with stronger adherence. This design fits the project because MirrorMove PD is meant to encourage repeatable home exercise, not only one-time novelty. The responses are collected through the following survey: https://forms.gle/ArJSVibjwZLLLxBH9. The data below is an early prototype-style pilot approximation, suitable for shaping future study design rather than proving medical efficacy.

Participant	Sessions Completed	Pre-use Confidence (1-10)	Post-use Confidence (1-10)	Improvement
P1	4	5	7	2
P2	5	4	6	2
P3	3	6	7	1
P4	6	3	6	3
P5	4	5	7	2
P6	3	4	5	1

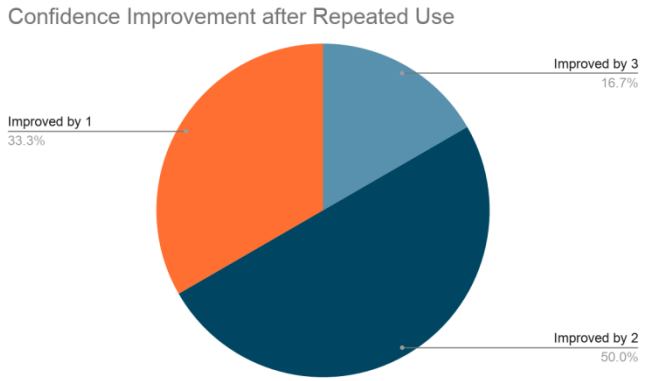


Figure 7. Pre- and post-intervention self-reported confidence scores across participants

The approximate pilot data show a positive short-term trend. Average confidence increases from 4.50 before use to 6.33 after the short trial period. The mean improvement is 1.83 points, the median improvement is 2, the minimum improvement is 1, and the maximum improvement is 3. Average session completion is 4.17 sessions, with a median of 4, a minimum of 3, and a maximum of 6. The participant with the largest gain also completed the most sessions, which suggests that repetition may matter more than a single exposure. The results are encouraging, but they should be interpreted carefully. Because the measure is self-report, confidence may rise partly because the interface becomes familiar, not only because motor function has objectively

improved. Even so, that finding is still useful. Rehabilitation systems must sustain participation, and a tool that makes users feel more confident and more willing to continue may support better long-term adherence than a technically advanced but disengaging system.

5. RELATED WORK

One major methodology for the same problem is dance-based intervention delivered through structured classes or supervised programs. Reviews by Sharp and Hewitt and by Carapellotti and colleagues show that dance can improve balance, gait, and motor symptom severity in people with Parkinson's disease, while more recent work suggests that even short dance exposure can have measurable benefits [7][8][9]. This approach is effective because rhythm, coordination, cueing, and enjoyment are naturally integrated. Its limitations are access, scheduling, transportation, and uneven availability of trained instructors. MirrorMove PD improves on this by packaging dance-based guidance into a repeatable home workflow, though it remains less clinically validated than supervised programs.

Another methodology is home-based telerehabilitation or home exercise prescription. Reviews of home-based exercise and telerehabilitation suggest that remote or self-managed programs can improve motor symptoms, balance, walking speed, and functional mobility, especially when exercise dose is sufficient and the format supports adherence [10][11][12][13]. This literature is strong because it targets the same access problem that Parkinson's patients often face. However, many of these systems focus on general exercise rather than a motivating arts-based format, and some depend on therapist supervision or narrow exercise modules. MirrorMove PD attempts to improve this by pairing home delivery with dance content, song selection, and lightweight remote control between devices.

The third methodology is exergaming, rhythmic cueing, or immersive virtual rehabilitation. Reviews in this area show promise for gait, balance, and quality of life, especially when cueing or interactive feedback helps structure movement [14][15][16][17]. These methods are appealing because they can make repetitive therapy more engaging and may produce measurable improvements in gait-related outcomes. Their limitations are inconsistency across studies, uncertainty about superiority over standard therapy, and in some cases dependence on specialized hardware. MirrorMove PD improves on this space by using ordinary consumer devices and dance-oriented content, but it still lacks the stronger scoring validation and hardware integration seen in more mature exergaming systems.

6. CONCLUSIONS

This project has several important limitations. First, it needs more songs and more varied therapeutic routines. A rehabilitation platform becomes more useful when it offers a broader range of tempos, styles, and difficulty levels, especially for users whose symptoms and stamina vary day to day. Second, the scoring system still needs refinement. The repository contains pose-analysis experiments and reference landmark generation, but the current prototype does not yet present a clinically validated movement score with strong reliability testing. Third, the project could be expanded into a broader platform that includes more professional therapeutic dance instructors, physical therapists, or rehabilitation specialists who can contribute curated content. If more development time were available, the next steps should include integrating pose-analysis more cleanly into the live session loop, improving network robustness, validating metrics against therapist judgment, adding more accessible interface options, and building a content pipeline that allows guided routines to grow beyond a small proof-of-concept library.

MirrorMove PD is a credible rehabilitation prototype because it connects evidence-informed therapeutic dance with a practical home-use delivery model. Its current value lies in accessibility, structure, and motivation. Its future value will depend on better scoring validation, richer content, and stronger collaboration with rehabilitation professionals and end users.

REFERENCES

[1] Balestrino, Roberta, and Anthnoy HV Schapira. "Parkinson disease." European journal of neurology 27.1 (2020): 27-42.

[2] Brown, Emery N., and Robert E. Kass. "What isstatistics?." The American Statistician 63.2 (2009): 105-110.

[3] Corti, Olga, Suzanne Lesage, and Alexis Brice. "What genetics tells us about the causes and mechanisms of Parkinson's disease." Physiological reviews (2011).

[4] Marras, Connie, et al. "Prevalence of Parkinson’s disease across North America." NPJ Parkinson's disease 4.1 (2018): 21.

[5] Willis, A. W., et al. "Incidence of parkinson disease in North America." npj Parkinson's Disease 8.1 (2022): 170.

[6] Yang, Wenya, et al. "Current and projected future economic burden of Parkinson’s disease in the US." npj Parkinson's Disease 6.1 (2020): 15.

[7] Sharp, Kathryn, and Jonathan Hewitt. "Dance as an intervention for people with Parkinson's disease: a systematic review and meta-analysis." Neuroscience & Biobehavioral Reviews 47 (2014): 445-456.

[8] Carapellotti, Anna M., Rebecca Stevenson, and Michail Doumas. "The efficacy of dance for improving motor impairments, non-motor symptoms, and quality of life in Parkinson’s disease: A systematic review and meta-analysis." PloS one 15.8 (2020): e0236820.

[9] Carapellotti, Anna M., Matthew Rodger, and Michail Doumas. "Evaluating the short-term effects of dance on motor and non-motor outcomes in people living with Parkinson’s: A crossover study." Plos one 20.7 (2025): e0328293.

[10] Hare, Marianne, James Hill, and Andrew Clegg. "Home-Based Exercise And People With Parkinson's Disease: A Systematic Review." British journal of neuroscience nursing 16.5 (2020): 230-232.

[11] Yang, Yong, et al. "The effect of home-based exercise on motor symptoms, quality of life and functional performance in Parkinson's disease: a systematic review and meta-analysis." BMC geriatrics 23.1 (2023): 873.

[12] Gao, Xianqi, et al. "The effect of home‐based exercise on motor and non‐motor symptoms with Parkinson's disease patients: A systematic review and network meta‐analysis." Journal of clinical nursing 33.7 (2024): 2755-2774.

[13] D’Souza, Arnold Fredrick, et al. "Effect of Home-based Telerehabilitation on Balance, Functional Mobility, and Quality of Life in Persons with Parkinson’s Disease: A Systematic Review and Meta-Analysis." International Journal of Telerehabilitation 17.2 (2025): 6725.

[14] Rocha, Poliany Silva, et al. "Exergaming in the treatment of gait, balance, and quality of life in Parkinson's disease: overview of systematic reviews." Physiotherapy Research International 28.3 (2023): e2002.

[15] Ye, Xiaofan, et al. "Rhythmic auditory stimulation promotes gait recovery in Parkinson's patients: a systematic review and meta-analysis." Frontiers in neurology 13 (2022): 940419.

[16] Burrai, Francesco, Luigi Apuzzo, and Renzo Zanotti. "Effectiveness of rhythmic auditory stimulation on gait in Parkinson disease: a systematic review and meta-analysis." Holistic Nursing Practice 38.2 (2024): 109-119.

[17] Ishaq, Shahid, et al. "Effectiveness of head-mounted virtual reality rehabilitation in individuals with Parkinson’s disease: A systematic review and meta-analysis." Assistive Technology 38.2 (2026): 187-197.

[18] Lugaresi, Camillo, et al. "Mediapipe: A framework for building perception pipelines." arXiv preprint arXiv:1906.08172 (2019).

[19] Grishchenko, Ivan, et al. "Blazeposeghum holistic: Real-time 3d human landmarks and pose estimation." arXiv preprint arXiv:2206.11678 (2022).