Analyzing Cricket: Shot Recognition & Similarity.

Ritik Bompilwar
Khoury College
Northeastern University
Boston, USA
bompilwar.r@northeastern.edu

Pratheesh
Khoury College
Northeastern University
Boston, USA
lnu.prat@northeastern.edu

Easha Meher
Khoury College
Northeastern University
Boston, USA
koppisetty.e@northeastern.edu

***Abstract*—Cricket analytics, an industry valued in the millions, is increasingly enhanced by artificial intelligence technologies. This project employs the backbone of the EfficientNet model, integrated within time-distributed blocks, to classify videos of cricket shots into ten distinct categories. Three variants of this model were trained on the CrickShot10 dataset, and their performance was quantitatively assessed on a test set. Optimization of model parameters was achieved through the implementation of a genetic algorithm, which systematically adjusted hyperparameters to improve accuracy and computational efficiency. The variant based on EfficientNetB0 emerged as the best-performing model, achieving a notable 94% accuracy on the test dataset. This model was subsequently utilized to develop a web application that facilitates real-time video classification and similarity comparisons. The findings of this project underscore the significant potential of machine learning to revolutionize sports analytics by enhancing the precision and utility of player performance evaluations and strategic planning in cricket.**

***Index Terms*—Cricket Shot Classification, Genetic Algorithm, Cricket Shot Similarity, CNN, GRU.**

I. Introduction

Cricket is a sport of substantial global interest that not only garners widespread enthusiasm but also generates significant economic outcomes, with revenues reaching into the billions. The strategic aspects of cricket, particularly shot selection, are crucial for achieving success in matches. As the sport progresses, technological advancements, especially in artificial intelligence (AI), are playing a pivotal role in elevating the analytical aspects of the game. This project explores the application of AI-driven analytics to classify cricket shots from video data into distinct categories, aiming to enhance understanding individual performance.

This project focuses on the development of a systematic approach to shot classification, utilizing AI to categorize video footage of cricket shots. By identifying patterns and similarities across different shots, the project seeks to provide actionable insights that can be used to improve player techniques and coaching strategies. Such classification is instrumental for coaches aiming to refine talent acquisition and development processes, and for players striving to enhance their performance metrics.

Moreover, the strategic application of AI in cricket extends beyond player and coaching benefits to broader economic implications. By optimizing shot selection and overall game strategies through sophisticated data analytics, cricket teams can enhance their competitive edge, potentially leading to improved game outcomes and increased fan engagement. This, in turn, may positively impact the financial dynamics within the cricket industry.

Further advancements would lead to the integration of AI into sports analytics, particularly cricket, offers promising avenues for future research and application. This project not only contributes to the ongoing discourse on sports technology but also sets a precedent for future AI applications in enhancing sporting strategies and engagements across various disciplines. Through continued innovation and research, AI holds the potential to redefine the norms of sports analytics and strategy formulation, ushering in a new era of data-driven sports enhancement.

II. Proposed Methodology

A. Data Acquisition and Preparation

(a) Frame from original video in the dataset.

(b) Frame from cropped and flipped version of video.

Fig. 1: Comparison of frames of videos from original and processed dataset.

The CricShot10 dataset was acquired from the authors of CricShotClassify [1]. The dataset had cricket shot videos of ten different categories namely cover, defense, flick, hook, late cut, lofted, pull, square cut, straight and sweep. The videos in dataset had varied length from 2 seconds to 5 seconds, but all videos were 25 FPS . The dataset was fairly balanced, with good balance of videos for left handed batsmen as well as right handed batsmen across all the formats of cricket.

The videos from all the classes of the dataset were cropped to height of 600 pixels to remove the score cards. Further the videos were converted to mp4 format and horizontal flipping was performed to increase the dataset size and for a better representation of different batting styles. Fig. 1 compares frames of videos from original and processed dataset. The new dataset was then divided into the training, validation,

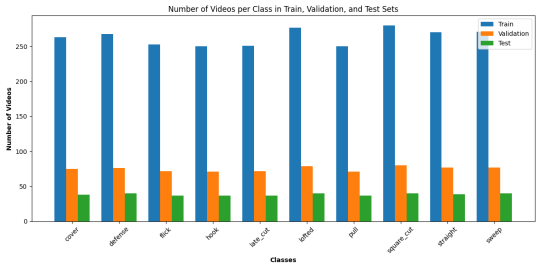


Fig. 2: Class wise distribution of videos in the dataset.

and testing sets with a 70-20-10 split for effective model evaluation. The class-wise distribution of the dataset is as represented in the Fig 2. The dataset is well balanced with sufficient data representation from each classes.

B. Model Architecture

The architecture employs a backbone derived from EfficientNet [2] [3], encased within a time-distributed wrapper. This design enables the model to process input video frames sequentially, maintaining a comprehensive temporal analysis. Subsequently, a time-distributed global average pooling layer condenses the high-dimensional feature data, effectively summarizing important information while reducing computational load.

The temporal features extracted are then fed into Gated Recurrent Unit (GRU) [4] layers, which are instrumental in capturing dynamic temporal relationships within the sequence of frames. This mechanism is pivotal for the model to discern motion patterns and changes over time, thus enhancing its predictive performance for sequential data.

The architecture incorporates dense layers which serve as the classification head, translating the rich, temporal feature set into actionable insights.

For experimental versatility, three distinct feature extractor heads were constructed: EfficientNetB0 [2], EfficientNetV2B0 [3], and EfficientNetB4 [2]. These variants allow for a comparative analysis of performance while keeping the remaining architecture consistent. The summary of the model employing the EfficientNetB0 extractor is presented in the subsequent table.

TABLE I: Summary of the EfficientNetB0 Based Model

Layer (type)	**Output Shape**
TimeDistributed (EfficientNetB0)	(None, None, 7, 7, 1280)
TimeDistributed (Pooling)	(None, None, 1280)
GRU	(None, None, 256)
GRU	(None, 128)
Dense	(None, 1024)
Dropout	(None, 1024)
Dense	(None, 10)

C. Hyperparameter Tuning Using Genetic Algorithm

The optimization of hyperparameters is pivotal for the success of machine learning models, especially those that require deep learning approaches. To efficiently manage this task, genetic algorithm (GA) was implemented using the Distributed Evolutionary Algorithms in Python (DEAP) library [5], renowned for its extensive capabilities in handling evolutionary algorithms.

This genetic algorithm's setup was carefully designed to explore the complex hyperparameter space, leveraging principles from evolutionary computing [6]:

- **Population Size:** A population of ten individuals was maintained, balancing diversity and manageability effectively.
- **Generations:** Planned for up to 50 generations, with an early stopping criterion to halt the process if no improvement was observed in 10 consecutive generations. This was essential to prevent overfitting and reduce computational load.
- **Mutation Rate:** Set at 20%, utilizing Gaussian mutations to introduce variability, crucial for exploring the search space extensively.
- **Crossover Rate:** A two-point crossover strategy facilitated the recombination of genetic material, promoting the inheritance of advantageous traits.

Additionally, **tournament selection** was utilized to choose individuals for the next generation: Selection Method: Tournament selection, a method where a set number of individuals are randomly selected from the population, and the best among them (based on fitness) is chosen. This method was repeated to fill the next generation, ensuring that only the fittest individuals were carried forward, thus maintaining or improving the overall population quality over generations [7].

The GA was tasked with optimizing the learning rate and the number of training epochs:

- **Learning Rate:** Varied between 0.0001 and 0.02, allowing the exploration of both slow and rapid learning dynamics.
- **Epochs:** Ranged from 1 to 20, to accommodate both quick adaptations and more prolonged, gradual learning processes.

Fitness of each individual was assessed based on the model's validation accuracy, ensuring that the selected hyperparameters genuinely improved performance. Throughout the operation of the GA, detailed logs were maintained in pandas DataFrames, recording learning rates, epochs, and fitness outcomes for each individual. This logging provided critical insights into the algorithm's progress, the effectiveness of different hyperparameter combinations, and the evolutionary dynamics within the population.

The use of tournament selection helped to ensure that the genetic algorithm was efficient in propagating only the most promising solutions through successive generations. This

Fig. 3: Schematic of the Sequential Neural Network Architecture

method was instrumental in enhancing the population's overall performance metrics and led to substantial improvements in validation accuracy over baseline configurations.

D. Model Training

The training of the models was executed on an Apple MacBook Pro equipped with the M1 Pro chip, featuring an 8-core CPU, a 14-core GPU, and 16GB of unified memory. The optimization process employed the Adam optimizer with an initial learning rate of 0.001. To enhance convergence and avoid plateauing, a callback function dynamically adjusted the learning rate, reducing it by a factor of 0.2 whenever the validation accuracy did not improve or remained constant for a span of 4 epochs. The loss function chosen for the task was Sparse Categorical Cross-Entropy, suitable for multi-class classification problems where each class is exclusive. All the models were trained for 20 epochs, ensuring adequate exposure to the training data while preventing overfitting.

III. Results and Discussion

A. Evaluation Metrics

In the evaluation of machine learning models for classification tasks, the project utilize a spectrum of metrics to comprehensively assess performance. While accuracy offers a macro-level view of model success, it often falls short of revealing the nuances in multi-class classification problems, such as the imbalance across classes. Therefore, to gain a more granular understanding of model behavior, precision, recall, and the F1-score were employed. Precision, the ratio of true positives to all predicted positives, gauges the model's exactness. Recall, or sensitivity, indicates the model's ability to identify all relevant samples within a class. The F1-score, a harmonic mean of precision and recall, provides a singular metric to evaluate the balance between precision and recall, particularly when the class distribution is skewed.

B. Performance Evaluation

1) Training Dynamics: The progression of the model's accuracy and loss across epochs is graphically represented for both training and validation phases. Figure 4 showcases graphs that depict a consistent rise in training accuracy alongside a steady decline in loss, highlighting the model's ability to learn effectively.

2) Class-specific Analysis: The classification report provides a detailed analysis of performance by class, as shown in Table II, which details the precision, recall, and F1-scores for each shot type. Notably, the 'hook' and 'sweep' shots achieve high F1-scores, demonstrating the model's proficiency in recognizing these actions. Conversely, the 'flick' shot exhibits a lower recall, as indicated in Table II, signaling a need for further refinement of the model or additional data to enhance classification confidence.

Shot Type	Precision	Recall	F1-score
Cover	0.90	0.97	0.94
Defense	0.97	0.97	0.97
Flick	0.97	0.92	0.94
Hook	1.00	0.97	0.99
Late Cut	0.90	0.97	0.94
Lofted	0.93	0.93	0.93
Pull	0.94	0.92	0.93
Square Cut	0.95	0.90	0.92
Straight	0.88	0.90	0.89
Sweep	0.95	0.93	0.94

TABLE II: Classification report showing precision, recall, F1-score, and support for each shot type.

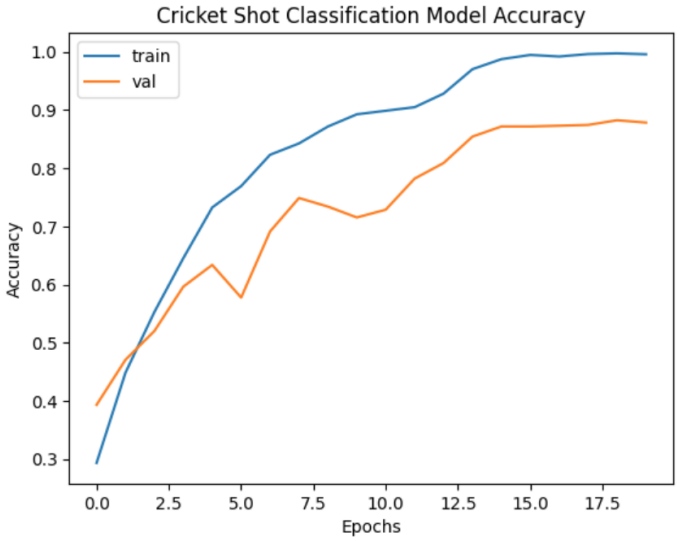
The confusion matrix, shown in Figure 5, elucidates the model's predictive accuracy across classes, with true positives and misclassifications visible at a glance. For instance, discerning between visually similar 'lofted' and 'pull' shots poses a challenge, as reflected by the misclassifications in the figure. This insight is instrumental in driving future enhancements, potentially through re-balancing the dataset or refining the model's feature extraction phase.

3) Overall Model Effectiveness: The culmination of the model's performance is encapsulated by the validation accuracy of 88.24%, achieved with an optimized learning rate of 0.001 over 20 epochs.

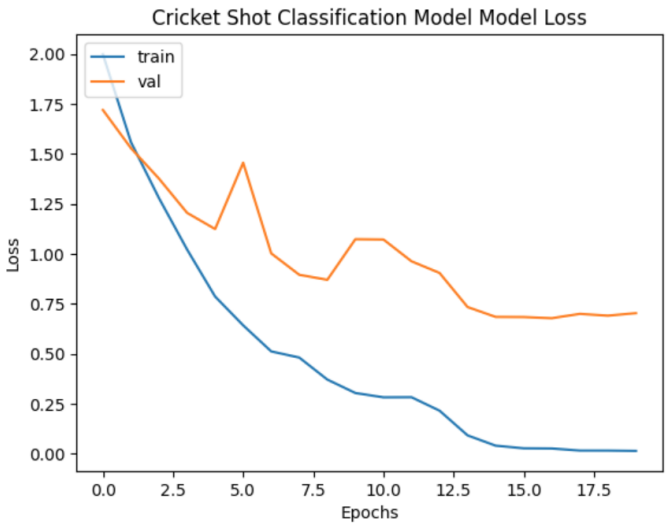
The balance between precision and recall reaffirms the model's potential as a tactical tool for cricket coaches and players.

4) Hyperparameter Optimization Using Genetic Algorithm: During the experiments, genetic algorithm was employed to explore and optimize the hyperparameter space. This algorithm successfully identified a set of hyperparameters that, while not ultimately chosen as the final settings due to the expert adjustments, provided a valuable baseline. The best configuration suggested by the genetic algorithm achieved a learning rate of 0.0017536633005774797 and was trained over 15 epochs, resulting in a validation accuracy of 72.34%. This exploration underscores the efficiency of the genetic algorithm in navigating complex hyperparameter spaces for advanced machine learning tasks.

5) Convergence of the Genetic Algorithm: Figure 6 illustrates the optimization trajectory of the genetic algorithm, displaying the average fitness values of the fittest individuals across generations. The trend in the plot shows an initial increase in fitness scores, reflecting the algorithm's capability to enhance the selection of hyperparameters. As the generations progress, the fitness scores plateau, indicating the convergence



(a) Accuracy of the model across different epochs during training and validation.



(b) Loss of the model across different epochs during training and validation.

Fig. 4: Model performance across different epochs: (a) accuracy and (b) loss during training and validation.

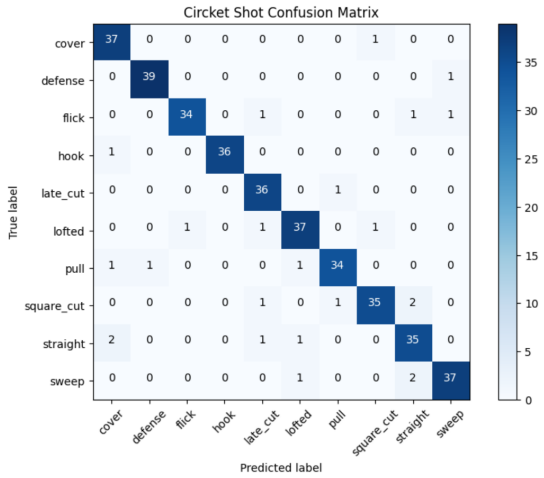


Fig. 5: Confusion matrix visualizing true and false predictions for each cricket shot class.

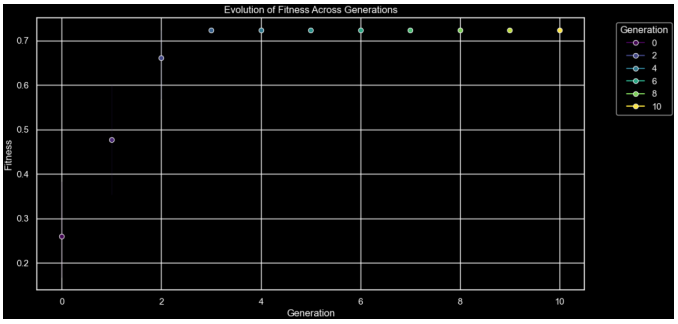


Fig. 6: Average fitness values of selected individuals across generations, demonstrating the genetic algorithm's optimization and convergence.

of the genetic algorithm towards a set of optimal hyperparameters. This stabilization suggests that further iterations may yield marginal improvements, highlighting the efficiency of the algorithm in selecting the best candidates for the model.

Despite the robustness of the genetic algorithm, the project faced constraints with the range of hyperparameters due to limited computational resources, which restricted the ability to explore broader configurations. Future work could extend these limits to discover potentially more optimal settings [8].

C. Model Comparative Evaluation

During the investigation of machine learning architectures for the task of cricket shot classification, the project considered various iterations of the EfficientNet models. These models were evaluated based on a comprehensive set of metrics to determine their efficacy in generalizing from training to unseen test data.

The EfficientNetB0, with its baseline configuration, demonstrated outstanding performance, achieving a testing accuracy, precision, recall, and F1-score all at 94%. This level of performance, detailed in Table III, indicates a model that generalizes well and can accurately classify shots across the test set. In contrast, the EfficientNetV2B0 model showed a lower performance with 81% accuracy and similar precision, recall, and F1-scores, as shown in Table III. This could point to the need for additional tuning or adaptability issues with the dataset. Interestingly, the more complex EfficientNetB4 did not match the performance of its counterparts, managing only a 74% score across these metrics, which challenges the notion that greater model complexity necessarily leads to better performance.

The comparative evaluation of these models reveals a nuanced relationship between model complexity and task-specific performance. The EfficientNetB0's superior results suggest

a better match with the task's requirements and highlight the importance of model architecture suitability over mere complexity, as evidenced by the data in Table III.

Future efforts will be directed towards a deeper analysis of these architectures, fine-tuning to achieve an optimal balance of model complexity, training efficiency, and prediction accuracy. The ultimate aim is to develop a model that not only performs well in theoretical benchmarks but also proves valuable in practical, real-world sports analytics.

Model	Testing Accuracy	Precision	Recall	F1 Score
EfficientNetB0	94%	94%	94%	94%
EfficientNetV2B0	81%	82%	81%	81%
EfficientNetB4	74%	75%	74%	74%

TABLE III: Comparative performance of different EfficientNet model architectures on the test set.

D. GradCam Visualisation

Fig. 7: Grad-CAM Visualisation for four different input videos.

Grad-CAM (Gradient-weighted Class Activation Mapping) [9] is a powerful technique that provides insights into the decision-making process of convolutional neural networks. It works by visualizing the regions of input that are important for predictions from these models, making it an invaluable tool for interpreting the model's behavior.

For video-based inference, Grad-CAM was utilized to highlight the areas within each frame that contributed most significantly to the model's classification decisions. This approach enables an intuitive understanding of which features within the temporal sequence of frames the model perceives as most distinctive. By applying Grad-CAM to specific layers of the model, spatial focus of activations in relation to the frame content was observed.

The Fig. 7 depict the Grad-CAM outputs superimposed on the corresponding frames from the different videos. Each subfigure represents a different moment or aspect of the model's predictions, revealing how the model's attention varies with temporal context.

Model was mostly found to be focusing on batsmen for the set of videos this visualisation was applied to, confirming the robustness of the model prediction.

IV. Similarity Comparison

The fine-tuned EfficientNetB0 backbone based model was used for the feature extraction process to quantitatively assess the similarity between two video sequences by removing the last three inference layers of the network for feature extraction. These layers are adept at capturing generalized, high-level content features without the final classification bias, making them ideal for this purpose.

The resulting model outputs a vector of deep features for each video, representing the abstract semantic content effectively. These feature vectors are then compared using cosine similarity, which measures the cosine of the angle between them. This mathematical approach quantifies the similarity in terms of orientation in the feature space, providing a robust basis for identifying resemblances across videos.

The similarity between the feature vectors from two videos is measured using the cosine similarity, a metric suitable for high-dimensional space comparison. Mathematically, cosine similarity is defined as:

$$\text{similarity} = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} \tag{1}$$

where $\vec{a}$ and $\vec{b}$ are the feature vectors of the first and second video, respectively. The dot product of the vectors $(\vec{a} \cdot \vec{b})$ measures the cosine of the angle between them, providing a measure of orientation similarity, regardless of their magnitude.

This similarity score ranges from -1 to 1, where 1 indicates identical directionality in the multidimensional feature space, 0 indicates orthogonality (no similarity), and -1 indicates diametrically opposed features.

V. Web Application

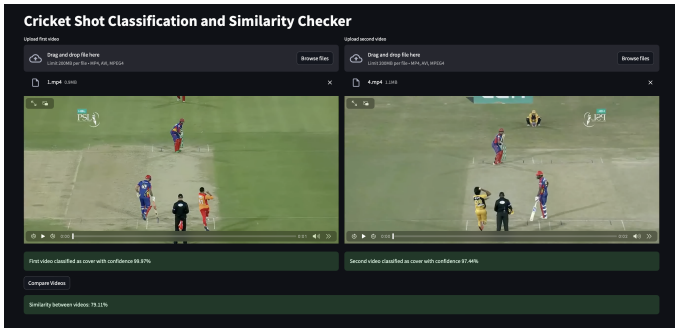


Fig. 8: Web application for cricket shot video classification and similarity comparison.

A streamlit [10] based wep app was created for enhanced user experience. The app interface allow users to upload

videos. Once the video is uploaded it automatically asserts the category of the video as per the model inference. Users can also compare similarity between two videos of same class. If the uploaded videos are of different categories, the app wouldn't compute the similarity as the videos belong to two different classes. Fig 8 represent a screengrab of the web app interface.

VI. Conclusion and Future Scope

This project aimed at advancing cricket analytics by classifying cricket videos into ten different categories of shot and also comparing similarity between two videos. Three different models with EfficientNet based Backbone in timedistributed blocks were fine-tuned on the enhanced CrickShot10 dataset. Genetic Algorithm based hypeparameter tuning was implemented for optimising model performance. Simlarity comparison was carried out using the best performing model's weights. The best performing model was used to create a web app for video classification and similarity computation.

The project's methodology opens several avenues for future enhancement and application. One immediate extension would involve the integration of real-time analysis capabilities, which could provide instant feedback during training sessions and matches. Additionally, expanding the dataset to encompass a broader array of playing styles and scenarios could significantly refine the model's accuracy and robustness, making it more universally applicable across different levels of cricket play.

Further research could also explore the adoption of more sophisticated neural network architectures or alternative similarity metrics to enhance the depth and accuracy of the analysis. The potential for this technology to not only optimize training and performance but also to enrich live broadcasting of sports events presents an exciting frontier for sports analytics. This project has set a foundational framework that promises to redefine the integration of AI in sports, paving the way for a new era of data-driven enhancements in athletic training and audience engagement.

VII. Source Code

For complete source code, dataset access, and interactive application, please visit the following links:

GitHub Repository: GitHub Repository
Original Dataset: Original Dataset
Processed Dataset: Processed Dataset
Web Application: Streamlit Application

VIII. Individual Contributions

Eash Meher worked on data augmentation and dataset segregation. She also trained EfficientNetV2B0 backbone based model.

Pratheesh worked on Genetic Algorthm based Hyperparameter optimisation for model training and also trained EfficientNetB0 backbone based model.

Ritik Bompilwar trained the EfficentNetB4 backbone based model and worked on GradCAM visualisation and similarity comparison of videos. He also designed the streamlit based web-app and deployed it to streamlit cloud.

References

[1] A. Sen, D. Kaushik, P. Dhar, and T. Koshiba, "Cricshotclassify: An approach to classifying batting shots from cricket videos using a convolutional neural network and gated recurrent unit," *Sensors*, vol. 21, p. 2846, 04 2021.

[2] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[3] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," 2021.

[4] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[5] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "Deap: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, 2012.

[6] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer, 2015.

[7] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[8] R. Sarker and C. Newton, *Optimization modeling: A practical approach to solving real-world optimization problems*. CRC Press, 2002.

[9] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, p. 336–359, Oct. 2019.

[10] A. Treuille, A. Kelly, T. Teixeira, *et al.*, "Streamlit," 2020. Software available from https://www.streamlit.io.