Combining Physics and Deep Learning Models to Simulate the Flight of a Golf Ball

William McNally Jacob Lambeth Dustin Brekke
Dunlop Sports Americas
Huntington Beach, CA
willmcnally@clevelandgolf.com

Abstract

We introduce a new golf ball flight model powered by deep learning. Our method combines a physics model with a deep learning model by inserting a neural network directly into the differential equations governing the projectile motion of the golf ball. The role of the neural network is to estimate the aerodynamic coefficients based on the state of the golf ball at each time step. The entire model was made end-to-end differentiable, permitting us to train the neural network using only measured launch conditions and landing positions. However, in experiments we find that using additional loss terms, such as the max height error, improves the accuracy of the predicted landing position. The key to our approach is that we automatically learn the relationship between the aerodynamic coefficients and the state of the golf ball directly from the data as opposed to manually defining a model that imposes a bias. As a result, we are able to reduce the mean landing position error by 28% compared to a published model that learns the coefficients by fitting polynomials to the spin ratio. Our method is also computationally efficient, with a processing time of 35 ms for a single shot using a CPU.

1. Introduction

Physics models and simulations (*e.g.*, computational fluid dynamics [13, 28, 32, 45], finite element analysis [6, 18, 30, 42], contact models [14, 20, 23, 38, 44, 48], biomechanical models [33, 47], *etc.*) play a fundamental role in sports engineering and serve as a vital tool for innovation, especially in regards to the design of sporting equipment. These dynamic models often include physical parameters or variables (*e.g.*, aerodynamic coefficients, material properties, forces, *etc.*) whose values are unknown and need to be identified using empirical data. Identifying these parameters directly often requires rigorous experiments, so a common approach is to perform indirect parameter identification using measurements that are easier to obtain [34, 37]. However, when the parameter in question is a function of the system states, the designer imposes an inductive bias on the model when choosing an appropriate target function [9]. For example, the designer may choose a linear regression model when the true nature of the parameter may be highly complex and non-linear.

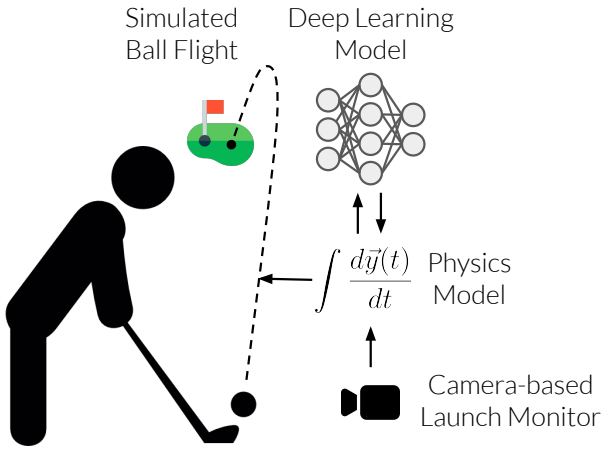

Figure 1. We propose a new ball flight model for golf that integrates deep learning directly into the differential equations governing the motion of a golf ball. After training the neural network using ball flight meta data, realistic golf ball trajectories can be simulated using the initial state of the golf ball as measured by camera-based launch monitors.

Deep learning and artificial neural networks [36] present an opportunity to learn parameters in physics simulations without making assumptions about their true nature. Despite the recent popularity of deep learning and its apparent suitability for parameter identification, its use in physics models and simulations is remarkably scarce. In this paper, we harness the power of deep learning in a physics model that simulates the flight of a golf ball after it is struck. At each time step, the simulation calls upon a neural network to estimate aerodynamic coefficients based on the state of

the golf ball, which are then used to compute the aerodynamic forces and solve the differential equations governing the motion of the golf ball. Moreover, we implement the entire simulation in a manner that is end-to-end differentiable and propose a method for training the neural network using ball flight meta data obtained from radar-based launch monitors. After training, the ball flight model may be used with any launch monitor, including camera-based launch monitors commonly found in golf simulators (see Figure 1).

While existing ball flight models often relate the aerodynamic coefficients to physical properties such as Reynold's number or the spin ratio [11, 27, 50], our approach makes no such assumptions and automatically learns the relationship between the aerodynamic coefficients and the raw golf ball state (*i.e.*, its velocity and spin vectors). As a result, we find that our approach generates more realistic ball flights, as evidenced by a 28% reduction in the mean landing position error compared to a published baseline that estimates the aerodynamic coefficients using the spin ratio as the independent variable in a polynomial regression [27].

1.1. Motivation

Accurately predicting the flight of a golf ball has great utility within the golf industry. Camera-based launch monitors, such as the GCQuad from Foresight Sports, capture the initial conditions of the golf ball, but they can only see a very short distance after impact. A model must be used to predict the remaining flight of the ball. Radar-based launch monitors, such as a TrackMan, capture the full flight of the golf ball. However, a ball flight model must be used when indoors or when the launch monitor cannot capture the full flight of the ball. Additionally, players can make use of a "normalized" output, where the ball flight is adjusted for environmental conditions such as wind, altitude, and temperature.

Ball flight models are also used by ball and club manufacturers to improve the performance of their products. Clubhead properties, such as face curvature, center of gravity, and moments of inertia, can directly affect the launch conditions of the golf ball after impact [22]. An accurate flight model can be used by a club manufacturer to tune these properties to maximize distance or accuracy [46, 55, 64, 65]. A ball manufacturer may use a flight model to more efficiently design a ball's dimple pattern [1, 3, 11]. Ball flight models have also been used to tune biomechanical golfer models [8, 47].

Although a statistical model can also be used to estimate the ball's landing position [16, 27], physics-based flight models provide a variety of benefits. By calculating the full trajectory of the golf ball, visual feedback is provided to the player, which allows them to play and practice virtually. Physics-based flight models can also determine the landing conditions (velocity and spin) of the golf ball more easily. The landing conditions affect how the ball will interact with the turf and is vital to predicting the run of the ball [56]. Although this work focuses on a golf ball flight model, the same approach can be applied to other ball sports. Ball flight models have been developed for many other sports, including baseball [2, 4, 59], tennis [21], soccer [25, 29], table tennis [60], cricket [17], and volleyball [63].

1.2. Related Work

There have been many different models and approaches used to estimate the trajectory and landing position of the golf ball. Cochran and Stobbs [16] showed that driver distance could be estimated as a function of ball speed. Daish [22] developed a numerically integrated model using lift and drag. Erlichson [26] used optimization routines to determine launch conditions to maximize carry. Smits and Smith [62] included spin rate decay and nonlinear drag dependence on Reynold's number. Quintavalla [57] examined the cross-dependance of Reynold's number and spin ratio for estimating drag and lift coefficients. Various ball trajectory models have also been extended to three dimensions to accurately capture the curve of ball flight [49, 50, 50]. Developing a ball flight model generally requires accurate prediction of coefficients of drag and lift. There have been many studies and experiments to measure these coefficients in controlled environments [10, 24, 43, 53, 62, 67]. Others have calculated drag and lift coefficients computationally [7, 12, 19, 61]. With the introduction of radar-based launch monitors, which can measure the launch conditions and ball position over the entire course of its flight, much larger and diverse datasets can be obtained. Sajima *et al.* [58] examined the effect of dimple design on the outdoor flight trajectory of golf balls using a TrackMan. Ferguson *et al.* [27] implemented an artificial neural network to map launch conditions to carry distance, offline distance, and apex height. This model, however, had low accuracy and did not capture the full flight of the golf ball.

Deep learning has led to significant improvements in numerous modeling fields [36] but has often been avoided when modeling physical systems. The traditional application of deep learning would ignore the underlying physics, which could lead to implausible results. A better approach is to insert deep learning models into the differential equations governing a dynamical system (*i.e.*, the equations of motion). The recent work of Lutter *et al.* [39] formalized the integration of feed-forward networks into the Euler-Lagrange equation with so-called Deep Langrangian Networks. Our methodology is similar in that we approximate an inverse model by representing unknown functions in the dynamic equations using feed-forward networks. By contrast, Neural Ordinary Differential Equations (Neural ODEs) [15] instead solve ODEs inside a neural network to model the hidden state as a continuous function of time.

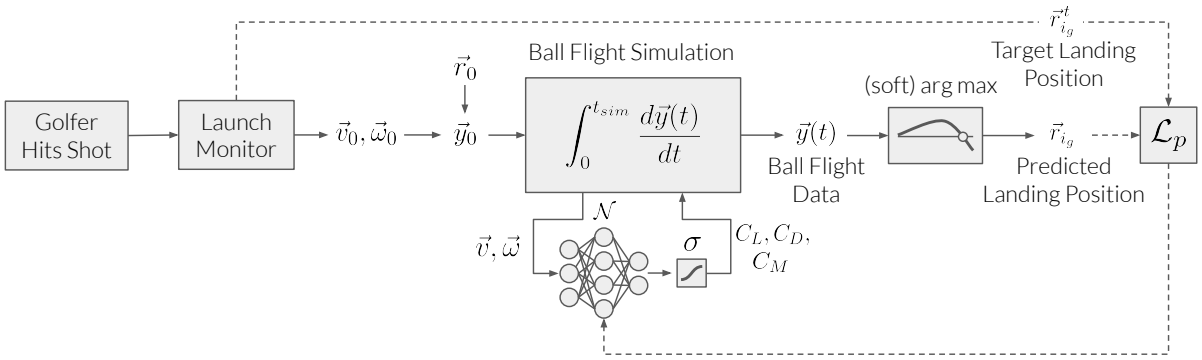

Figure 2. An overview of the proposed method. A launch monitor records a shot hit by a golfer and measures the launch conditions $\vec{v}_0$ and $\vec{\omega}_0$, which, together with the initial golf ball position $\vec{r}_0$, form the initial state of the golf ball $\vec{y}_0$. The differential equations $\frac{d\vec{y}(t)}{dt}$ include a neural network $\mathcal{N}$ that estimates the aerodynamic coefficients C_L, C_D, and C_M. The ball flight is numerically integrated from 0 to t_{sim} seconds. A soft-argmax operation is then used to obtain the predicted landing position $\vec{r}_{i_g}$. During training (dashed arrows), the target landing positions $\vec{r}^{\,t}_{i_g}$ are used to compute the landing position loss $\mathcal{L}_p$ and update the neural network weights.

2. Method

The proposed method integrates a neural network into a physics model of a golf ball in flight. An overview of our approach is given in Figure 2. The solid arrows reflect how the model is used during inference and the dashed arrows reflect the flow of information during training. The following sections describe the physics model and the aerodynamic forces acting on the golf ball (§2.1), the deep learning model used to estimate the aerodynamic coefficients (§2.2), and the integration of physics and deep learning models for simulating multiple golf ball trajectories in parallel and obtaining final landing positions (§2.3).

The reference coordinate system used for the physics model has the X axis pointing away from a right-handed golfer, the Y axis pointing at the target, and the Z axis pointing towards the sky. We briefly introduce the notation used. A three-dimensional vector $\vec{u} \in \mathbb{R}^3$ points in the direction $\hat{u} = \frac{\vec{u}}{||\vec{u}||}$. A continuous time series of $\vec{u}$ vectors is denoted as $\vec{u}(t)$. A discrete time step i of $\vec{u}(t)$ is denoted as $\vec{u}_i$ and has components $u_{i,x}$, $u_{i,y}$, and $u_{i,z}$.

2.1. Physics Model

The physical system is modeled using ordinary differential equations (ODEs), wherein a golf ball with mass m, radius R, position $\vec{r}(t)$, and velocity $\vec{v}(t)$ is treated as particle and obeys Newton's second law of motion:

$$\frac{\mathrm{d}\vec{v}(t)}{\mathrm{d}t} = \frac{\vec{F}(t)}{m} \tag{1}$$

where $\vec{v}(t) = \frac{d\vec{r}(t)}{dt}$. The golf ball also has a spin $\vec{\omega}(t)$ that decays over time due to an aerodynamic spin decay torque $\vec{T}(t)$:

$$\frac{\mathrm{d}\vec{\omega}(t)}{\mathrm{d}t} = \frac{\vec{T}(t)}{I} \tag{2}$$

where $I = \frac{2}{5}mR^2$ is the mass moment of inertia of the golf ball. The aerodynamic forces include the lift force $\vec{F}_L$ and the drag force $\vec{F}_D$. For convenience, we temporarily drop the "(t)" notation and note that all forces and states are time-varying. The drag force acts in the direction opposing the velocity of the golf ball ($-\hat{v}$), and the lift force acts perpendicular to the velocity and spin, in the direction $\frac{\vec{v}\times\vec{\omega}}{||\vec{v}\times\vec{\omega}||}$. Similar to in [11, 50, 57], the magnitudes of the aerodynamic forces and spin decay torque are related to the dynamic air pressure q, the cross-sectional area of the golf ball $A = \pi R^2$, and the dimensionless aerodynamic coefficients C_L, C_D, and C_M. The dynamic air pressure is defined as:

$$q = \frac{1}{2}\,\rho\,||\vec{v}||^2 \tag{3}$$

where ρ is the air density. Finally, the aerodynamic forces are defined using the equations below:

$$\vec{F}_L = C_L\,q\,A\,\frac{\vec{v}\times\vec{\omega}}{||\vec{v}\times\vec{\omega}||} \tag{4}$$

$$\vec{F}_D = -C_D\,q\,A\,\hat{v} \tag{5}$$

$$\vec{T} = -C_M\,q\,d\,A\,\hat{\omega} \tag{6}$$

where d is the golf ball diameter [50]. The net force $\vec{F}$ acting on the golf ball is the sum of the aerodynamic forces and the constant force of gravity $\vec{G} = -mg\,\hat{k}$:

$$\vec{F} = \vec{F}_L + \vec{F}_D + \vec{G}. \tag{7}$$

2.2. Deep Learning Model

At each time step of the ball flight simulation, a fully-connected feed-forward neural network, otherwise known as a multilayer perceptron [51], is used to estimate the aerodynamic coefficients. The inputs to the network are the velocity and spin states $\vec{v}$ and $\vec{\omega}$, which are normalized through

division by 89.4 m/s (200 mph) and 2094 rad/s (20k rpm), respectively. It was found that this normalization scheme stabilized training by ensuring the inputs to the network were bounded between 0 and 1 (typical ball speeds and spin rates in golf lie below these upper bounds – see §3.1 for a description of the dataset used in this study). In the base configuration, which we label `Phys-NN`, the neural network $\mathcal{N}$ has two hidden layers with 256 and 128 units, respectively, and uses the rectified linear unit (ReLU [52]) activation function after each hidden layer. The output layer returns the intermediate features C'_L, C'_D, and C'_M:

$$C'_L, C'_D, C'_M = \mathcal{N}(\vec{v}, \vec{\omega}) \tag{8}$$

The actual aerodynamic coefficients C_L, C_D, and C_M are obtained by passing the intermediate features through a sigmoid function and dividing by 2, 2, and 50, respectively:

$$C_L = \sigma(C'_L)/2 \tag{9}$$
$$C_D = \sigma(C'_D)/2 \tag{10}$$
$$C_M = \sigma(C'_D)/50 \tag{11}$$

This was done to constrain the ranges of the potential values for C_L, C_D, and C_M based on values observed in previous work [11,50,57,62]. C_L and C_D are constrained to [0, 0.5], whereas C_M is constrained to [0, 0.02].

A number of variations to the base neural network configuration are evaluated in §3.3. Specifically, we explore the effects of varying the number of hidden layers and hidden units, among other architectural changes, including predicting a fourth aerodynamic coefficient C_Q that varies the dynamic air pressure (*i.e.*, $q = C_Q\,\rho\,||\vec{v}||^2$). The effects of changing various training hyperparameters, including the batch size and loss function, are also investigated.

2.3. Simulation and Training

This section describes how the physics model and the deep learning model are jointly applied to simulate golf ball flight paths. The entire system was implemented in PyTorch version 1.13 and is differentiable from the end to end (*i.e.*, from the launch conditions to the final landing positions), permitting the neural network, which estimates the aerodynamic coefficients at each time step, to be trained using only measured launch conditions and landing positions.

2.3.1 Initial Conditions

Set up as an initial value problem, the physics simulation begins with the launch conditions of the golf ball, namely the ball speed $||\vec{v}_0||$, launch angle γ, azimuth ϕ, back spin $\omega_{0,x}$, and side spin $\omega_{0,z}$. These launch conditions are typically measured using commercially available launch monitors. The third spin component $\omega_{0,y}$, known as rifle spin, is considered to be less relevant to the flight of the golf ball and is therefore typically not provided by launch monitors. In this study, $\omega_{0,y}$ is assumed to be zero. The components of the initial golf ball velocity $\vec{v}_0$ are computed as:

$$\vec{v}_0 = \begin{bmatrix} v_{0,x} \\ v_{0,y} \\ v_{0,z} \end{bmatrix} = \begin{bmatrix} ||\vec{v}_0||\, cos(\gamma)\, sin(\phi) \\ ||\vec{v}_0||\, cos(\gamma)\, cos(\phi) \\ ||\vec{v}_0||\, sin(\gamma) \end{bmatrix} \tag{12}$$

The initial state of the system $\vec{y}_0$ comprises nine time-dependent variables, including the components of $\vec{v}_0$, $\vec{\omega}_0$, and the initial ball position $\vec{r}_0$, which is set to $[0, 0, 1]^T$ cm to account for the tee height and to facilitate a check for the ball crossing the ground plane. By extension, the state of the system at any time t is $\vec{y}(t) = [\vec{v}(t), \vec{\omega}(t), \vec{r}(t)]$. When the wind velocity $\vec{v}^{\,w}$ is known, the golf ball velocity $\vec{v}(t)$ can be replaced with its velocity relative to the wind: $\vec{v}^{\,r}(t) = \vec{v}(t) - \vec{v}^{\,w}$.

2.3.2 Numerical Integration

To facilitate the simulation of multiple flights in parallel, a vectorized implementation of Euler's method is used to numerically integrate a batch of flights from 0 to t_{sim} seconds using a fixed time step Δt:

$$\vec{y}(t + \Delta t) = \vec{y}(t) + \Delta t\, \frac{d\vec{y}(t)}{dt} \tag{13}$$

where

$$\frac{d\vec{y}(t)}{dt} = \begin{bmatrix} \frac{d\vec{v}(t)}{dt} \\ \frac{d\vec{\omega}(t)}{dt} \\ \frac{d\vec{r}(t)}{dt} \end{bmatrix} = \begin{bmatrix} \frac{\vec{F}(t)}{m} \\ -\frac{\vec{T}(t)}{I} \\ \vec{v}(t) \end{bmatrix} \tag{14}$$

Since $\vec{F}$ and $\vec{T}$ depend on the aerodynamic coefficients, a forward pass through the neural network is performed at each time step. Hence, $\frac{t_{sim}}{\Delta t}$ passes through the neural network are required to the complete the simulation, after which the loss is computed.

In experiments involving a physics-only baseline model (details in §3.2), it was found that setting $\Delta t = 0.1$ s provided adequate convergence of the golf ball landing position. Decreasing Δt further negatively affects the performance of the proposed method by increasing the number of times the neural network needs to be called, which slows down the simulation, increases memory usage, and makes the training process more susceptible to exploding and vanishing gradients [31, 54]. The simulation time t_{sim} was set to 10 s, which is longer than all the flight times in the dataset presented in §3.1. As a result, the neural network is called 100 times per batch of simulated flights.

2.3.3 Loss Computation

We propose a method for training the neural network using the minimum amount of information, namely the launch

conditions and landing positions. Such data is easily obtained using radar-based launch monitors (*e.g.*, a TrackMan or FlightScope). A camera-based launch monitor, such as the GCQuad, could also be used to collect launch conditions together with manually recorded landing positions. Admittedly, a more effective approach to training the network would be to minimize the position error over full flight paths; however, such data can only obtained with special access to the APIs of radar-based launch monitors. We instead choose to take advantage of readily available ball flight meta data that radar-based launch monitors provide by default. In addition to using the landing position error, we explore the use of an additional loss based on the max height of the shot.

To index the time at which the golf ball crosses the ground plane in a differentiable manner, we implement a custom soft-argmax [40, 41, 66] operation and apply it to the height of the golf ball $r_z(t)$. More specifically, the time step index $\mathrm{i_g} \in \mathbb{W}$ at which the golf ball is closest to the ground is computed as:

$$\mathrm{i_g} = \mathrm{i_s} + \arg\max(-|r_{i_s\,:\,\mathrm{end},z}|) \tag{15}$$

where $\mathrm{i_s} \in \mathbb{W}$ is a starting index used to prevent the soft-argmax operation from returning an index near the beginning of the flight when the golf ball is close to the ground. A value of $\lfloor \frac{1}{\Delta t} \rfloor$, corresponding to $t = 1$ s, was arbitrarily chosen for $\mathrm{i_s}$. During validation and inference, $\mathrm{i_g}$ is obtained using a standard $\arg\max$ and the final ball state is refined by interpolating $\vec{y}(t)$ to $r_z = 0$. The interpolation step is omitted during training under the assumption that the mean landing position error resulting from a lack of interpolation is close to zero. It follows that the landing position error $\mathcal{L}_p$ is computed as:

$$\mathcal{L}_p = ||\vec{r}_{i_g} - \vec{r}^{\,t}_{i_g}|| \tag{16}$$

where $\vec{r}^{\,t}_{i_g}$ is the target landing position provided by the launch monitor. We additionally consider the effects of adding a loss for the max height of the golf ball:

$$\mathcal{L}_h = |\max(r_z) - \max(r^t_z)| \tag{17}$$

with equal weighting such that the total loss is:

$$\mathcal{L} = \mathcal{L}_p + \mathcal{L}_h \tag{18}$$

During training, the losses are averaged over the batch of simulated flights.

3. Experiments

In this section we describe the dataset used to evaluate the proposed method (§3.1); re-implement two published baseline models for performance comparison (§3.2); analyze the accuracy obtained using various model configurations and training settings (§3.3); analyze the computational efficiency (§3.4); and finally examine the aerodynamic coefficients (§3.5).

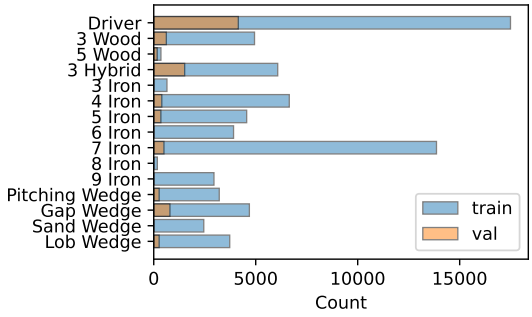

Figure 3. Histogram of the club types used in the dataset.

3.1. Dataset

A ball flight dataset was collected using a TrackMan 3e launch monitor to record shots during outdoor golf robot and player testing. The dataset was prepared by combining TrackMan Performance Studio reports from multiple test sessions. The reports contain measurements (and some estimates) for the club delivery, ball launch, ball flight (max height position only), and ball landing (position and velocity) for each shot. The relevant data for this study included the ball speed $||\vec{v}_0||$, vertical launch angle γ, horizontal launch angle ϕ (azimuth), spin rate $||\vec{\omega}_0||$, spin axis ψ, flat ground landing position $\vec{r}^{\,t}_{i_g}$, max height, and flight time. The back and side components of spin are computed as $\omega_{0,x} = ||\vec{\omega}_0||\,cos(\psi)$ and $\omega_{0,z} = -||\vec{\omega}_0||\,sin(\psi)$, respectively. Wind information was not included in the dataset ($\vec{v}^{\,w} = 0$).

Shots with measurements deemed "invalid" by the TrackMan software were omitted from the dataset. Moreover, shots with a max height of less than 8 yds were omitted to eliminate severe mishits (tops, duffs, *etc.*). The final dataset contained 90,233 shots in total. A validation set was created by combining 10% of the most recent of shots (9,024 shots) from the robot and player tests. The remaining 81,209 shots were used for training. This experimental protocol was more challenging than randomly sampling shots for each subset, which led to nearly identical distributions for the training and validation sets. Figure 4 provides normalized histograms for the relevant measurements in the dataset. The distribution patterns arise from the typical club types used in the tests (*e.g.*, Driver, 7 iron, Lob Wedge, *etc.*). A histogram of the club types used is provided in Figure 3.

The primary error metric considered for accuracy evaluation was the mean landing position error, which is equal to the mean $\mathcal{L}_p$ on the validation set. The mean absolute max height error was considered as a secondary metric, and is equal to the mean $\mathcal{L}_h$.

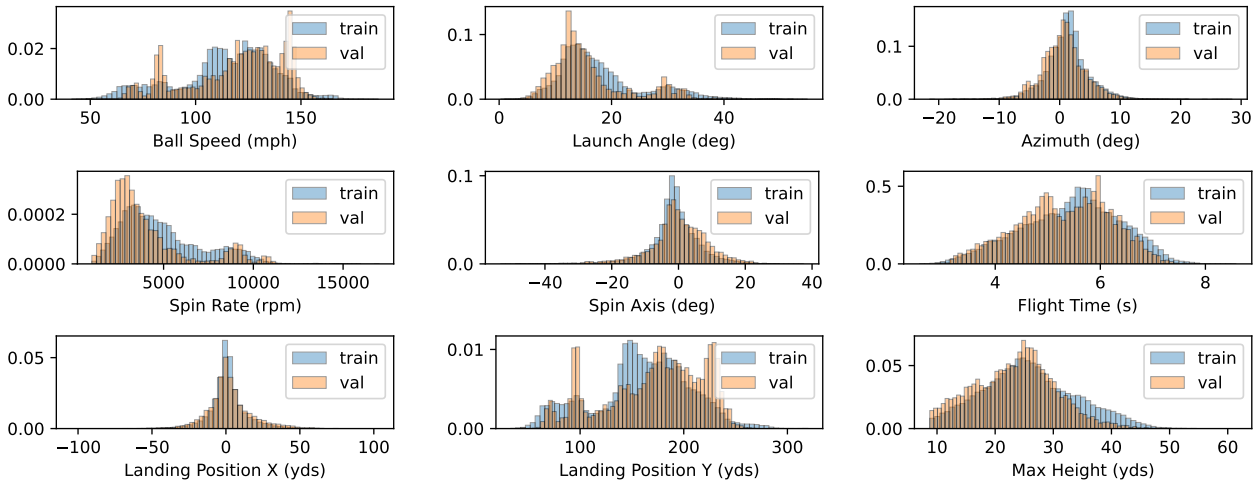

Figure 4. Normalized histograms for the datasets used. The distribution patterns arise from the typical club types used in the golf tests (see Figure 3 for club type histogram).

3.2. Baselines

To evaluate the accuracy of the proposed approach, two published ball flight models from Ferguson *et al.* [27] were re-implemented and used as baselines for comparison. The first was a physics model similar to the proposed method, with the exception that C_L, C_M, and C_D were parameterized using the spin ratio S:

$$S = \frac{R\,||\vec{\omega}||}{||\vec{v}||} \tag{19}$$

$$C_L = p_1 + p_2\,S + p_3\,S^2 \tag{20}$$

$$C_D = p_4 + p_5\,S + p_6\,S^2 \tag{21}$$

$$C_M = p_7\,S \tag{22}$$

This model was labeled `Phys-Q`, for "Physics-Quadratic." The second baseline was a feed-forward neural network that predicted the landing position directly from the launch conditions. This approach was labeled `NN` and was re-implemented by modifying the neural network described in §2.2 to predict the landing position directly:

$$r_{i_g,x}, r_{i_g,y} = \mathcal{N}(\vec{v}_0, \vec{\omega}_0) \tag{23}$$

The parameters of the two baselines (`Phys-Q` and `NN`) and the base configuration of the proposed method (`Phys-NN`) were optimized by minimizing $\mathcal{L}_p$ over 50 epochs using stochastic gradient descent [5]. The parameters p in `Phys-Q` were initialized using the optimal values found in the original work [27]. The default training settings included using the Adam optimizer [35] with a constant learning rate of 0.001 and a batch size of 1024. All training was performed on an NVIDIA RTX A2000 laptop GPU. The training and validation losses (*i.e.*, the mean landing position errors) for each model are shown in Figure 5.

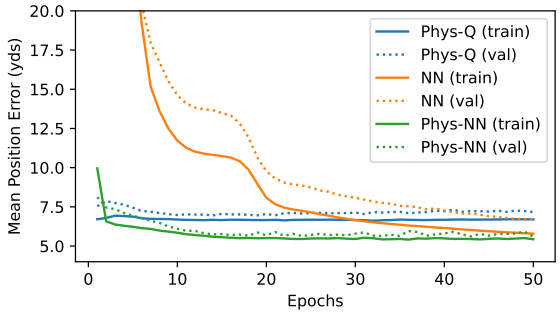

Figure 5. Mean landing position errors during the training of `Phys-Q` and `NN` (baseline models) and the base configuration of the proposed method `Phys-NN`.

The minimum validation errors for the `Phys-Q`, `NN`, and `Phys-NN` models were 6.95 yds, 6.70 yds, and 5.60 yds, respectively. The accuracy of `NN` surpassed that of `Phys-Q`, which was not the case in the original work of Ferguson *et al.* It is possible that the neural network performed poorly in the original work due to the limited amount of training data.

Both neural network models outperformed the `Phys-Q` baseline, and the proposed `Phys-NN` model that combines the physical model with deep learning was the most accurate of the three; however, `NN` took longer to converge and it is likely that it would reach an accuracy similar to `Phys-NN` if trained for more epochs. Nonetheless, we reiterate that the utility of a statistical model that only predicts landing positions is limited.

Model	Description	Aero. Coeff. Equations	Train Epochs	Loss	$\mathcal{L}_p$	$\mathcal{L}_h$
`Phys-NN`	base model	8, 9, 10, 11	20	$\mathcal{L}_p$	5.66	10.7
`Phys-NN-pM`	$C_M = p_M\, S$	9, 10, 24, 25	20	$\mathcal{L}_p$	5.31	5.96
`Phys-NN-pM-cQ`	$q = C_Q\, \rho\, \|\|\vec{v}\|\|^2$	9, 10, 25, 26, 27	20	$\mathcal{L}_p$	**5.24**	3.92
`Phys-NN-pM`	units = {256, 128}	9, 10, 24, 25	20	$\mathcal{L}_p$	5.31	5.96
`Phys-NN-pM-2x`	units = {512, 256}	9, 10, 24, 25	20	$\mathcal{L}_p$	**5.23**	6.69
`Phys-NN-pM-4x`	units = {1024, 512}	9, 10, 24, 25	20	$\mathcal{L}_p$	5.24	6.53
`Phys-NN-pM-3l`	units = {512, 256, 128}	9, 10, 24, 25	20	$\mathcal{L}_p$	5.45	4.83
`Phys-NN-pM-b256`	batch size = 256	9, 10, 24, 25	20	$\mathcal{L}_p$	**5.18**	7.11
`Phys-NN-pM-b512`	batch size = 512	9, 10, 24, 25	20	$\mathcal{L}_p$	5.22	6.71
`Phys-NN-pM`	batch size = 1024	9, 10, 24, 25	20	$\mathcal{L}_p$	5.31	5.96
`Phys-NN-pM-b2048`	batch size = 2048	9, 10, 24, 25	20	$\mathcal{L}_p$	5.49	5.49
`Phys-NN-pM`	loss = $\mathcal{L}_p$	9, 10, 24, 25	20	$\mathcal{L}_p$	5.31	5.96
`Phys-NN-pM`	loss = $\mathcal{L}_p + \mathcal{L}_h$	9, 10, 24, 25	20	$\mathcal{L}_p + \mathcal{L}_h$	5.78	**1.39**
`Phys-Q`	quadratic fit, loss = $\mathcal{L}_p$	20, 21, 22	50	$\mathcal{L}_p$	6.96	3.61
`Phys-Q`	quadratic fit, loss = $\mathcal{L}_p + \mathcal{L}_h$	20, 21, 22	50	$\mathcal{L}_p + \mathcal{L}_h$	6.63	1.64
`NN`	$r_{i_g,x}, r_{i_g,y} = \mathcal{N}(\vec{v}_0, \vec{\omega}_0)$	-	100	$\mathcal{L}_p$	5.24	-
`Phys-NN+`	final model, loss = $\mathcal{L}_p$	9, 10, 25, 26, 27	50	$\mathcal{L}_p$	4.91	3.95
`Phys-NN+`	final model, loss = $\mathcal{L}_p + \mathcal{L}_h$	9, 10, 25, 26, 27	50	$\mathcal{L}_p + \mathcal{L}_h$	**4.75**	**1.31**

Table 1. Results of the ablation and hyperparameter study. $\mathcal{L}_p$ is the minimum mean landing position error on the validation set, in yards. $\mathcal{L}_h$ is mean absolute max height error on the validation set, in yards, for the model checkpoint with the lowest $\mathcal{L}_p$.

3.3. Ablation and Hyperparameter Study

This section explores the effects of modifying the architecture of `Phys-NN` and the training settings. First, knowing that increasing C_L and decreasing C_M both increase the lift force due to the Magnus effect, it was hypothesized that this interaction could impede the convergence of the neural network. A simplified architecture was therefore proposed:

$$C'_L, C'_D = \mathcal{N}(\vec{v}, \vec{\omega}) \tag{24}$$

$$C_M = p_M\, S \tag{25}$$

where the parameter p_M was jointly learned while training the neural network. An additional modification was proposed in which the dynamic pressure was permitted to vary using a new aerodynamic coefficient C_Q:

$$C'_L, C'_D, C'_Q = \mathcal{N}(\vec{v}, \vec{\omega}) \tag{26}$$

$$C_Q = \sigma(C'_Q) \tag{27}$$

$$q = C_Q\, \rho\, ||\vec{v}||^2 \tag{28}$$

In addition to the aforementioned architectural modifications, the batch size was varied, as well as the number of hidden units. Further, we quantify the effects of including the mean absolute max height error $\mathcal{L}_p$ in the loss function. The results of the ablation and hyperparameter study are summarized in Table 1. The best settings (shown in **bold**) were combined to produce the final model, `Phys-NN+`, which was trained for 50 epochs.

The addition of the $\mathcal{L}_h$ loss dramatically improved the mean absolute max height error for both the `Phys-Q` and `Phys-NN+` models. For the models trained using $\mathcal{L}_p$ only, the observed variation in the computed $\mathcal{L}_h$ values suggests that the problem could be underdefined. In other words, there is potential for golf ball trajectories of varying height to reach a similar landing positions. Using the $\mathcal{L}_h$ loss is therefore recommended to constrain the problem and promote realistic simulation results. Furthermore, we find that using $\mathcal{L}_h$ has the added benefit of improving the landing position error $\mathcal{L}_p$. While this was not the original intent, it is possible that $\mathcal{L}_h$ acts as a physical prior that helps steer the neural network optimization towards a global minimum. When using the $\mathcal{L}_h$ loss, the mean landing position error of `Phys-NN+` was 4.75 yds, which is 28% more accurate than `Phys-Q` and 9% more accurate than using a neural network to predict the landing position directly (`NN`). Any remaining error is attributed to random noise resulting from measurement error and environmental factors such as wind, temperature, and humidity.

3.4. Computational Efficiency

Advancements in deep learning have naturally led to the use of larger neural networks that use more memory and

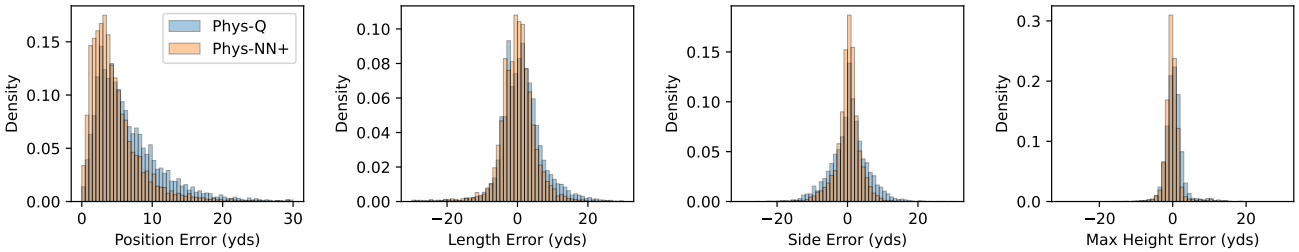

Figure 6. Normalized histograms for the landing position error ($\mathcal{L}_p$), length error ($r_{i_g,y} - r^t_{i_g,y}$), side error ($r_{i_g,x} - r^t_{i_g,x}$), and max height error ($\max(r_z) - \max(r^t_z)$). The difference between the `Phys-Q` and `Phys-NN+` position error histograms corresponds to 28% reduction in the mean landing position error.

Model	1 shot CPU (ms)	1024 shots CPU (μs)	1 shot GPU (ms)	1024 shots GPU (μs)
`Phys-Q`	25.2	97.8	49.8	73.6
`Phys-NN+`	34.9	348	57.9	86.1

Table 2. Latencies of `Phys-Q` and `Phys-NN+` running on a CPU (Intel i7-11850H) and GPU (NVIDIA RTX A2000). Latencies averaged over 256 trials using randomly selected launch conditions from the validation dataset.

require specialized accelerated hardware, such as GPUs, for fast or "real-time" inference. In this application, however, the neural network used is very small in comparison to the ones commonly used in the deep learning literature. In fact, the neural network computation comprises just a few matrix multiplications. As a result, the computational overhead of our approach is not significant in practice.

Table 2 summarizes the latencies (processing time per simulated shot) of `Phys-Q` and `Phys-NN+` running on a CPU (Intel i7-11850H) and GPU (NVIDIA RTX A2000). `Phys-NN+` processes a single shot on a CPU in 35.9 ms, compared to 25.2 ms for `Phys-Q`. Both models process shots much faster than real-time as most golf shots remain in the air for more than 3 seconds (as shown in Figure 4). When processing a batch of shots, `Phys-NN+` suffers more on CPU due to the increased memory usage of the neural network. However, the batched latency is comparable with `Phys-Q` when using the GPU. We therefore recommend using a GPU when simulating many shots in parallel.

3.5. Aerodynamic Coefficients

Finally, we compare the lift and drag coefficients produced by the `Phys-NN+` and `Phys-Q` models trained using the $\mathcal{L}_p + \mathcal{L}_h$ loss. Figure 7 plots the lift and drag coefficients produced by `Phys-NN+` against the spin ratio for all simulation time steps in the validation set (573,308 data points). The polynomial curve fits of `Phys-Q` are overlaid for comparison. Interestingly, the neural network

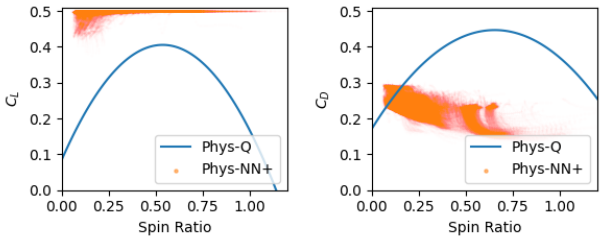

Figure 7. The lift and drag coefficients produced by `Phys-NN+` for all time steps in the simulated validation set shots (573,308 data points). The polynomial curve fits of `Phys-Q` are overlaid for comparison.

in `Phys-NN+` nearly saturated the C_L sigmoid, resulting in many C_L values near 0.5. C_D had significantly more variance, potentially as a result of the saturated C_L. For this reason, it is possible that the coefficients generated by `Phys-NN+` do not reflect the actual physics involved. In future work, we recommend exploring the constraints of the sigmoid function and performing wind tunnel testing to investigate the authenticity of the aerodynamic coefficients produced by the neural network.

4. Conclusion

This paper presents a golf ball flight model that integrates deep learning into the equations of motion to improve the accuracy of simulated golf ball trajectories. We outline a methodology for training a neural network that estimates instantaneous aerodynamic coefficients using measured launch conditions and a landing position loss. Notably, we find that adding a second equally weighted loss for the max height error not only improves the heights of the simulated trajectories, but also reduces the landing position error. Finally, we show that our approach has minimal computational overhead compared to existing models and runs much faster than what is required for practical applications.

References

[1] *Effects of Dimple Size and Depth on Golf Ball Aerodynamic Performance*, volume Volume 2: Symposia, Parts A, B, and C of *Fluids Engineering Division Summer Meeting*, 07 2003. 2

[2] Robert Kemp Adair. *The physics of baseball*. Harper & Row New York, 1990. 2

[3] Firoz Alam, Tom Steiner, Harun Chowdhury, Hazim Moria, Iftekhar Khan, Fayez Aldawi, and Aleksandar Subic. A study of golf ball aerodynamic drag. *Procedia Engineering*, 13:226–231, 2011. 2

[4] LeRoy W Alaways. Comparison between observed and simulated baseball trajectories. *The Engineering of Sport 7*, pages 345–351, 2008. 2

[5] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993. 6

[6] Stephanie Ankrah and NJ Mills. Performance of football shin guards for direct stud impacts. *Sports Engineering*, 6(4):207–219, 2003. 1

[7] Katsumi Aoki, Koji Muto, and Hiroo Okanaga. Aerodynamic characteristics and flow pattern of a golf ball with rotation. *Procedia Engineering*, 2(2):2431–2436, 2010. 2

[8] Daniel Balzerson, Joydeep Banerjee, and John McPhee. A three-dimensional forward dynamic model of the golf swing optimized for ball carry distance. *Sports Engineering*, 19:237–250, 2016. 2

[9] Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000. 1

[10] PW Bearman and JK Harvey. Golf ball aerodynamics. *Aeronautical Quarterly*, 27(2):112–122, 1976. 2

[11] D Beasley and T Camp. Effects of dimple design on the aerodynamic performance of a golf ball. In *Science and Golf IV: Proceedings of the World Scientific Congress of Golf*, pages 328–340, 2002. 2, 3, 4

[12] Nikolaos Beratlis, Kyle Squires, and Elias Balaras. Numerical investigation of magnus effect on dimpled spheres. *Journal of Turbulence*, (13):N15, 2012. 2

[13] Bert Blocken, Thijs van Druenen, Yasin Toparlar, and Thomas Andrianne. CFD analysis of an exceptional cyclist sprint position. *Sports Engineering*, 22:1–11, 2019. 1

[14] Peter Brown and John McPhee. A 3d ellipsoidal volumetric foot–ground contact model for forward dynamics. *Multibody System Dynamics*, 42:447–467, 2018. 1

[15] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *NeurIPS*, 2018. 2

[16] Alastair COCHRAN and John STOBBS. *The Search for the Perfect Swing*. J. B. Lippincott Company, 1968. 2

[17] Peter Coutis. Modelling the projectile motion of a cricket ball. *International Journal of Mathematical Education in Science and Technology*, 29(6):789–798, 1998. 2

[18] Derek Covill, Steven Begg, Eddy Elton, Mark Milne, Richard Morris, and Tim Katz. Parametric finite element analysis of bicycle frame geometries. *Procedia Engineering*, 72:441–446, 2014. 1

[19] Jacob Crabill, Freddie Witherden, and Antony Jameson. High-order computational fluid dynamics simulations of a spinning golf ball. *Sports Engineering*, 22:1–9, 2019. 2

[20] Rod Cross. Grip-slip behavior of a bouncing ball. *American Journal of Physics*, 70(11):1093–1102, 2002. 1

[21] Rod Cross and Crawford Lindsey. Measurements of drag and lift on tennis balls in flight. *Sports Engineering*, 17:89–96, 2014. 2

[22] CB Daish. *The physics of ball games*. English University Press, 1972. 2

[23] Behzad Danaei, William McNally, Erik Henrikson, and John McPhee. Adjusting a momentum-based golf clubhead-ball impact model to improve accuracy. In *Proceedings*, volume 49. MDPI, 2020. 1

[24] John M Davies. The aerodynamics of golf balls. *Journal of Applied Physics*, 20(9):821–828, 1949. 2

[25] Uwe Dick, Daniel Link, and Ulf Brefeld. Who can receive the pass? a computational model for quantifying availability in soccer. *Data Mining and Knowledge Discovery*, 36(3):987–1014, 2022. 2

[26] Herman Erlichson. Maximum projectile range with drag and lift, with particular application to golf. *American Journal of Physics*, 51(4):357–362, 1983. 2

[27] Spencer Ferguson, William McNally, and John McPhee. Predicting the flight of a golf ball: Comparing a physics-based aerodynamic model to a neural network. In *Engineering of Sport 14: Proceedings of the 14th Conference of the International Sports Engineering Association*, 2022. https://doi.org/10.5703/1288284317493. 2, 6

[28] John Eric Goff. A review of recent research into aerodynamics of sport projectiles. *Sports engineering*, 16(3):137–154, 2013. 1

[29] John Eric Goff and Matt J Carré. Trajectory analysis of a soccer ball. *American Journal of Physics*, 77(11):1020–1027, 2009. 2

[30] SR Goodwill, Robert Kirk, and SJ Haake. Experimental and finite element analysis of a tennis ball impact on a rigid surface. *Sports engineering*, 8:145–158, 2005. 1

[31] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *NeurIPS*, 2018. 4

[32] R Keith Hanna. CFD in sport - a retrospective; 1992-2012. *Procedia Engineering*, 34:622–627, 2012. 1

[33] Keaton A Inkol, Colin Brown, William McNally, Conor Jansen, and John McPhee. Muscle torque generators in multibody dynamic simulations of optimal sports performance. *Multibody System Dynamics*, 50:435–452, 2020. 1

[34] Jovana Jovic, Adrien Escande, Ko Ayusawa, Eiichi Yoshida, Abderrahmane Kheddar, and Gentiane Venture. Humanoid and human inertia parameter identification using hierarchical optimization. *IEEE Transactions on Robotics*, 32(3):726–735, 2016. 1

[35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[36] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 1, 2

[37] CSH Lin, CK Chua, and JH Yeo. Analysis and simulation of badminton shuttlecock flight through parameter identification of a slow-speed serve shot. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, 229(4):213–221, 2015. 1

[38] Edward Lozowski, Krzysztof Szilder, and Sean Maw. A model of ice friction for a speed skate blade. *Sports Engineering*, 16:239–253, 2013. 1

[39] Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *ICLR*, 2019. 2

[40] Diogo C Luvizon, David Picard, and Hedi Tabia. 2d/3d pose estimation and action recognition using multitask deep learning. In *CVPR*, 2018. 5

[41] Diogo C Luvizon, Hedi Tabia, and David Picard. Human pose regression by combining indirect part detection and contextual information. *Computers & Graphics*, 85:15–22, 2019. 5

[42] Griselda Lyn and NJ Mills. Design of foam crash mats for head impact protection. *Sports engineering*, 4(3):153–163, 2001. 1

[43] Bin Lyu, Jeff Kensrud, Lloyd Smith, and Taylor Tosaya. Aerodynamics of golf balls in still air. In *Proceedings*, volume 2, page 238. MDPI, 2018. 2

[44] Margarida Machado, Pedro Moreira, Paulo Flores, and Hamid M Lankarani. Compliant contact force models in multibody dynamics: Evolution of the hertz contact theory. *Mechanism and machine theory*, 53:99–121, 2012. 1

[45] Paul Mannion, Yasin Toparlar, Bert Blocken, Magdalena Hajdukiewicz, Thomas Andrianne, and Eoghan Clifford. Improving CFD prediction of drag on paralympic tandem athletes: influence of grid resolution and turbulence model. *Sports Engineering*, 21:123–135, 2018. 1

[46] William McNally, Daniel Balzerson, Daniel Wilson, and John McPhee. Effect of clubhead inertial properties and driver face geometry on golf ball trajectories. *Procedia engineering*, 147:407–412, 2016. 2

[47] William McNally and John McPhee. Dynamic optimization of the golf swing using a six degree-of-freedom biomechanical model. In *Proceedings*, volume 2, page 243. MDPI, 2018. 1, 2

[48] William McNally, John McPhee, and Erik Henrikson. The golf shaft's influence on clubhead-ball impact dynamics. In *Proceedings*, volume 2, page 245. MDPI, 2018. 1

[49] John J McPhee and Gordon C Andrews. Effect of sidespin and wind on projectile trajectory, with particular application to golf. *American Journal of Physics*, 56(10):933–939, 1988. 2

[50] T Mizota, T Naruo, H Shimozono, M Zdravkovich, and F Sato. 3-dimensional trajectory analysis of golf balls. In *Science and Golf IV: Proceedings of the World Scientific Congress of Golf*, pages 349–358, 2002. 2, 3, 4

[51] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6):183–197, 1991. 3

[52] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 4

[53] Takeshi Naruo and Taketo Mizota. The influence of golf ball dimples on aerodynamic characteristics. *Procedia Engineering*, 72:780–785, 2014. 2

[54] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013. 4

[55] A Raymond Penner. The physics of golf. *Reports on progress in physics*, 66(2):131, 2002. 2

[56] A Raymond Penner. The run of a golf ball. *Canadian Journal of Physics*, 80(8):931–940, 2002. 2

[57] SJ Quintavalla. A generally applicable model for the aerodynamic behavior of golf balls. In *Science and Golf IV: Proceedings of the World Scientific Congress of Golf*, pages 341–348, 2002. 2, 3, 4

[58] T Sajima, T Yamaguchi, M Yabu, and M Tsunoda. The aerodynamic influence of dimple design on flying golf ball. In *The Engineering of Sport 6: Volume 1: Developments for Sports*, pages 143–148. Springer, 2006. 2

[59] Gregory S Sawicki, Mont Hubbard, and William J Stronge. How to hit home runs: Optimum baseball bat swing parameters for maximum range trajectories. *American Journal of Physics*, 71(11):1152–1162, 2003. 2

[60] Ralf Schneider, Lars Lewerentz, Stefan Kemnitz, and Christian Schultz. Table tennis and physics. 2022. 2

[61] CE Smith, Nikolaos Beratlis, Elias Balaras, Kyle Squires, and Masaya Tsunoda. Numerical investigation of the flow over a golf ball in the subcritical and supercritical regimes. *International Journal of Heat and Fluid Flow*, 31(3):262–273, 2010. 2

[62] AJ Smits and DR Smith. A new aerodynamic model of a golf ball in flight. In *Science and Golf II: Proceedings of the World Scientific Congress of Golf*, pages 340–347. 1994. 2, 4

[63] Takehiro Tamaru, Shinichiro Ito, and Masaki Hiratsuka. Serve ball trajectory characteristics of different volleyballs and their causes. In *Proceedings*, volume 49. MDPI, 2020. 2

[64] Frank D Werner, Richard C Greig, and Roger P Ganem. *How golf clubs really work and how to optimize their designs*. Origin Incorporated, 2000. 2

[65] DC Winfield and Teong E Tan. Optimization of the clubface shape of a golf driver to minimize dispersion of off-center shots. *Computers & structures*, 58(6):1217–1224, 1996. 2

[66] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *ECCV*, 2016. 5

[67] MV Zagarola, B Lieberman, and AJ Smits. An indoor testing range to measure the aerodynamic performance of golf balls. In *Science and Golf II: Proceedings of the World Scientific Congress of Golf*, pages 443–450. 1994. 2