Machine Learning for Inverse Design

Evan Thomas
Department of Physics, University of Ottawa

Thesis submitted to the University of Ottawa in partial fulfillment of the requirements for the degree of Master of Science in Physics

Declaration

I hereby declare that this thesis is my original work. All content has been written by me in its entirety unless explicitly stated otherwise. I have duly acknowledged all the sources of information that have been used in this thesis. This thesis has not been submitted to any other university or institution for any degree requirements or for any other purposes.

Evan Thomas
February 7, 2023

Acknowledgements

I would like to thank my thesis advisor Dr. Isaac Tamblyn for the opportunity to work at the forefront of an exclusive and exciting field. Dr. Tamblyn, your honest advice has been invaluable for me, and has left a lasting impression on my character. It has been a privilege to work with you and all of the other talented scientists at the National Research Council of Canada. A special thanks to Dr. Frederic Tessier at the NRC, whose guidance and assistance regarding EGSnrc enabled our work. Thank you also to all the members of Dr. Tamblyn's research group, past and present. Thank you to Kevin Ryczko for his advice and research on inverse design that inspired the surrogate model project, To Kyle Mills for his technical support and his work that preceded the Ising project. Thank you to Chris Beeler for his discretion and advice, and to Christoff Reimer for his camaraderie and encouragement. Lastly, thank you to my family and to Natasha.

Statement of Contributions

An original simulation setup was authored by Dr. Frederic Tessier at the National Research Council of Canada. As part of this work, it is distributed alongside the original EGSnrc source code under its license, the GNU Affero General Public License version 3 (GNU AGPL v3). As part of the "PickAI" project, the code for a Hamiltonian was provided by Kyle Mills. This is used, modified and distributed with his permission.

Figures 2, 6, 11, and 17 are either from, are derived from images from the public domain or from the creative commons. These figures are appropriately cited and used under their respective licenses. The validity of these images is confirmed by the author, since the credibility of these images cannot be derived from their sources. Furthermore, the caption of each of these figures contains a reputable reference that either describes topic of the image or shares a similar image.

This work leverages open-source and permissively licensed machine learning libraries. The random forest regressor was implemented by the Scikit-Learn library. In compliance with the third clause of its software license, the copyright holders of this library, "The scikit-learn developers", in no way endorse this work. Furthermore, Figure 15 is from the scikit-learn documentation, its use is permitted by the same license.

Automation, optimization, and other development is written in Python and Dash. These are used freely under license along with other free and open-source software. Use of these software and libraries does not imply endorsement of this work by any holders of copyrights. The code for the surrogate model simulated annealing is found at *https://www.gitlab.com/ethom/EGSnrc* or *https://github.com/CLEANit/EGSnrc*. The code for "PickAI" design-space sampler is found at *https://www.gitlab.com/ethom/PickAI* or *https://github.com/CLEANit/PickAI*.

Abstract

"Inverse design" formulates the design process as an inverse problem; optimal values of a parameterized design space are sought so to best reproduce quantitative outcomes from the forwards dynamics of the design's intended environment. Arguably, two subtasks are necessary to iteratively solve such a design problem; the generation and evaluation of designs. This thesis work documents two experiments leveraging machine learning (ML) to facilitate each subtask. Included first is a review of relevant physics and machine learning theory. Then, analysis on the theoretical foundations of ensemble methods realizes a novel equation describing the effect of Bagging and Random Forests on the expected mean squared error of a base model.

Complex models of design evaluation may capture environmental dynamics beyond those that are useful for a design optimization. These constitute unnecessary time and computational costs. The first experiment attempts to avoid these by replacing EGSnrc, a Monte Carlo simulation of coupled electron-photon transport, with an efficient ML "surrogate model". To investigate the benefits of surrogate models, a simulated annealing design optimization is twice conducted to reproduce an arbitrary target design, once using EGSnrc and once using a random forest regressor as a surrogate model. It is found that using the surrogate model produced approximately an $100\times$ speed-up, and converged upon an effective design in fewer iterations. In conclusion, using a surrogate model is faster and (in this case) also more effective per-iteration.

The second experiment of this thesis work leveraged machine learning for design generation. As a proof-of-concept design objective, the work seeks to efficiently sample 2D Ising spin model configurations from an optimized design space with a uniform distribution of internal energies. Randomly sampling configurations yields a narrow Gaussian distribution of internal energies. Convolutional neural networks (CNN) trained with NeuroEvolution, a mutation-only genetic algorithm, were used to statistically shape the design space. Networks contribute to sampling by processing random inputs, their outputs are then regularized into acceptable configurations. Samples produced with CNNs had more uniform distribution of internal energies, and ranged across the entire space of possible values. In combination with conventional sampling methods, these CNNs can facilitate the sampling of configurations with uniformly distributed internal energies.

Terminology

Design	A representation of a real world thing that captures everything of consequence relative to its intended behaviour and limitations.
(Design) Environment	All elements required to achieve an objective that are external to the design. Characterized by needing the design to maintain or achieve a desired state.
(Design) Behaviour	The effect of the design when deployed into, and acting upon its environment.
(Design) Performance	The alignment between the design's behaviour and the intended effect. Measured relative to all design objectives.
Performance metrics	Quantitative measurements of design performance.
Design space	The set of all possible combinations of design parameters.
Design optimization	A process that selects an optimal or merely sufficient design from the design space.
Objective function	A function that calculates and aggregates performance metrics into a score. This can either be positive (e.g. candidate rank) or negative (e.g. loss, cost or risk).
Training data	Data separated from the data source and used to train a model. Typically used in the context of machine learning models.
Test data	Data separated from the data source and used to test a model. Typically used in the context of machine learning models.
Goodness-of-fit (test)	A measure or test that determines the degree to which the results of a given model or hypothesis "agree" with observed data from the "modelled" phenomenon.
Bootstrapping	A method of estimating a probability distribution by repeatedly calculating the value of interest relative to a subsample of available data. Samples are taken repeatedly with replacement, and are called "bootstraps".
Expected value	The value of a random variable that is "expected". Typically refers to the weighted average all possible values, weighted according to their probability density. This value is called the "mean". Occasionally may refer to the middle or "median" value, or the value which has the single highest probability density, or "mode".
Coefficient of determination	The percentage of variance in a set of data, accounted for by a model. Typically denoted r^2.

Contents

1 The Present Work

This work investigates the suitability of machine learning methods for the purposes of "inverse design". The major research project of this work uses a surrogate model for candidate design evaluation in an inverse design optimization. The design problem considered is that of designing radiation masks, shielding, and lenses. To do so, the surrogate models developed here are used to predict the behaviour of gamma radiation incident upon a designed "scattering object". The source of our data is EGSnrc, which is discussed in depth in Chapter 3: "Overview of EGSnrc". This includes high level descriptions of each simulated phenomena, and an in-depth explanation of the means by which all of these phenomena are included in a Monte Carlo multi-physics simulation. Subsequently, in Chapter 4: "Machine Learning Methods", the fundamentals of machine learning are laid out. The machine model of decision trees are introduced and methods of implementation and training are explained. Ensemble methods, which combine the estimations of many machine learning models are introduced. The method of Bagging, introduced by Breiman [1] is evaluated in depth, leveraging a diverse background of machine learning theory and statistical analysis. Other ensemble methods such as the Random Subspace Methods (RSM) [2] are also discussed. The ensemble method of Random Forests [3] is explained as the combination of Bagging and a modified Random Subspace method. An original derivation for the impact of this combination is presented. In Chapter 5: "Results", a side by side comparison on the efficiency of using an EGSnrc [4] simulation to evaluate designs, relative to generating and using a machine learning surrogate model to evaluate designs. The design optimization used is simulated annealing. It is found that despite the additional overhead costs of generating training data and training a surrogate model it is significantly faster to use a surrogate model in the design optimization. Furthermore, the quality of the design generated using the surrogate model is equivalent or superior to that generated using the simulation directly. Moreover, when using the surrogate model for design evaluation, the design optimization yielded a useful design in fewer iterations than when the simulation itself was used. A second section in Chapter 5 documents a proof of concept that demonstrates the design-generating ability of Convolutional Neural Networks (CNNs). This project, nicknamed "PickAI", was motivated by the need for effective uniform sampling from a design space with non-uniform characteristics. Configurations for a two-dimensional Ising spin model are taken to be individual "designs". CNNs transform Gaussian random noise into outputs that when binned are

acceptable designs. This regularization includes non-continuous transformations, so the CNNs are trained using NeuroEvolution. Each CNN ranges over a design latent-space, each with its own distribution of characteristics. Finally, this work is concluded by a summary, suggestions for future works, and an outlook on the future of machine learning's application in inverse design.

2 Introduction to Inverse Design

The topics of inverse design, design automation and computational or numerical design assistance are of particular interest to both academic [5][6][7] and industrial [8][9][10] researchers. The implementation of machine-assisted design methods vary in involvement from minor optimization and simulation to full design conception and construction planning.

2.1 Conceptual Motivation

Imagine a spectrum of design methods. At one far end, one finds design methods of human intuition, where a design is evaluated by "gut feeling" and designs are generated from pure inspiration. Much of the history of design has dealt nearly exclusively at this far end. Opposite to human intuition, at the other far end of the spectrum of design methods, is absolute random search. There, all possible aspects or properties of a generated design are sampled uniformly without discretion. An analogous random-search design method of evaluation is trial by error, wherein the entirety of a design's performance is evaluated simultaneously and in-situ. Effective design at either pole requires either profound understanding and creativity, or substantial resources for computation and creation of design candidates. This design spectrum mirrors a range of complexity trade-offs. On the side of intuition, effective design requires a complex understanding of the circumstance. Therefore, the design space is simpler. On the other side, that of random-search, the design space is massively complex. There, a minimum of understanding of the design environment is necessary for a simple "success" or "failure" measure of feedback. Therefore, one is motivated to find a middle way. Ideally, one could leverage some of the understanding necessary for human intuition, and combine that with the computational power associated with random search and trial-and-error. In Inverse design in nanophotonics Moesky, et al. describe the components or motivations of inverse design [11]

> "There are at least two central thrusts: first, to determine the extent that the characteristics of a solution, either actual or desired, determine the system from which they are derived; and second, to find effective algorithms for working from solutions characteristics to physical systems"

A successful formalism of such a middle-way design method would be effective, predictable, flexible and robust. Commonly, in the research community, design of this sort is referred to as "inverse design." This term derives from the so-called "inverse-problem", a problem that has a desired outcome, but lacks inputs or initial conditions to reach the desired outcome. The term is often used casually to describe any means of algorithmically accelerated, automated, artificially intelligent, computer-aided, evolutionary, or novel means of design. One can imagine inverse design as an umbrella term that covers the middle of a spectrum of design methods. This thesis is concerned with finding and demonstrating models, methods and algorithms to perform or facilitate the "middle-way" of inverse design.

2.2 Design Terminology

To describe a design process, let us leverage some design terminology. A *design* is an object, process, method, or other noun that solves a *problem*. Such a problem may be absolute (e.g. damn a river) or partial (e.g. limit flooding as much as possible). An individual who seeks designs is a *designer*. The circumstance in which the design is intended to operate is the *design environment*. The set of all possible designs constitutes a *design space*, which one can imagine is dimensionalized by parameters, properties or characteristics of a design. The effect of an implemented design in the design environment is described as the design's *behaviour*. Judgment of a design can leverage derived quantities. These quantities are *performance metrics* and quantify design *performance* (i.e. how well a design solves its problem). Design behaviour and performance, similar to the design space, can be multi-dimensional. Designs are realized over the course of a *design process*. This may include a stage in which candidate designs are sequentially modified to improve the design's performance. Such a stage is here described as a *design optimization* or *design optimization process*.

2.3 What is Inverse Design

> "Form follows Function."
> - Louis Sullivan [12]

The term "inverse design" is derived from a related term in mathematics, "inverse problem" [13]. The "inverse problem" is the inverse of the "forward problem" [14], defined

by

$$d = g(m) \tag{1}$$

Where d is the data observed, resulting from a model of an environment, g, operating upon some input to the model, m. To solve an inverse problem, one seeks to recreate the initial conditions or inputs to a system that, through the forwards dynamics of the system, produces effects that are observed as a finite set of data. The process of design, as previously defined, seeks to solve for an input (i.e. design) to a system that given the forwards dynamics of that system (i.e. environment) yield a desired outcome or effect (i.e. objective). In there terminology previously introduced, d is the resulting performance metrics, g is the design environment including the quantifying of performance, and m is the parameterization of a design. So, as defined, inverse design is design formulated as an inverse problem. It is worth noting that the measurement and calculation of performance metrics, in inverse design, are a part of the environment. This is because the measurement and calculation of performance constitute additional mappings between design and observation. Therefore, the measurement and calculation of performance metrics are a part of the design environment. Whereas, the resulting performance metrics themselves are observables.

It is worthwhile to consider how others have defined inverse design. Loonen, de Vries and Goia, in Figure 15.1 [13], differentiate "inverse design" from "direct design" by a matter of direction. "Direct design", according to them, begins with a candidate design, which is evaluated by a "physical-mathematical model" yielding the design's performance. Conversely, "inverse design" begins with some desired performance, which is evaluated by some "physical-mathematical model" yielding an optimal design. Arguably, in the context of an iterative design optimization, the distinction between inverse design and "direct design" is equivalent to that between gradient-based and gradient-free optimization methods. However, Loonen, de Vries and Goia immediately abandon this distinction, promptly suggesting a systematic parameter sweep as a part of an inverse design process. This is probably for the best since inverse problems can be solved using gradient-free optimization methods. Many papers that invoke the term "inverse design" combine gradient-free optimization (e.g. genetic algorithm) with some method to either evaluate [15] or generate [5] candidate designs. Lastly, this is not how the term is typically used. Colloquially, both in industry and academia, "inverse design" refers to any algorithmically assisted design process.

2.3.1 Inverse Design Framework

The present work attempts to realize a complete framework for inverse design. This framework has three parts.

1. Generate designs
2. Evaluate designs
3. (Plan) the construction of designs

The first two of these, to generate and evaluate designs are investigated in this work.

2.4 Inverse Design Generation

Return to our imagined spectrum of design methods. This spectrum can be interpreted to represent either methods of design generation or methods of design evaluation. For design generation, on one end of the spectrum is pure inspiration. Many important designs have been realized through spontaneous inspiration, but inspiration is not reliably available. Therefore, it is desirable to have methods to generate or 'imagine' designs that are more reliable and predictable. At the other end of the spectrum of design methods is random search. One can imagine a limiting search in design space with a maximum of design parameters (up to the individual atomic composition of a design) are sampled uniformly. Though reliably produced, designs generated in this manner may have poor performance, unrealistic characteristics, be unviable, or all of the above. Therefore, it is desirable to use methods of design generation that are more targeted.

A predictable, flexible and robust inverse design formalism for the generation of designs has many advantages. It is possible (and possibly likely) that optimal designs occupy an unintuitive region of their design space. Work by Hornby and Globus leveraging inverse design to produce an "evolved antenna" yielded irregular shapes [16]. Without leveraging computational inverse design, they describe feasibility of realizing such a design would be "severely limited" due the massive demands of "time and labour" from those rare individuals who posses "a significant amount of domain knowledge". Hornby and Globus do not reject the possibility that such a design could be realized by human design alone. However, they do demonstrate a design project that approaches a reasonable limit of what human-only design could accomplish. With only an incremental assumption, a conclusion can be drawn that there must be design projects that are unrealizable without

leveraging inverse design. The unfeasibility of a design project can arise from complexity of the design environment or complexity of design space. Optimal design of either case would respectively be unintuitive or 'remote'. Both challenges can be mitigated, since inverse design methods can operate in unintuitive and complex design environments, as demonstrated by Hornby and Globus, and can rapidly generate (and evaluate) a large multiplicity of designs, facilitated by random sampling.

Lastly, though not unique to an inverse design, a design formalism should provide means to measure design trade-offs. By leveraging mathematical design generation and parameterization, these trade-offs become measurable.

2.4.1 Methods of Design Optimization

Design optimizations are processes that pair (or combine) design generation and design evaluation to produce an optimal or at least optimized design. There is a huge diversity of design optimization methods [17]. Some methods neatly pair design generation and evaluation (such as simulated annealing or genetic algorithms), while others blur our proposed dichotomy (such as neural-style transfer). In this section, a selection of methods of design optimization are explained. Original sample code is reference and included in various appendices of this work.

For the sake of clarification, much of the work on inverse design concerns itself with entire methods of design optimization. In such works, the elements of design generation, evaluation, and additional aspects of the optimization are all considered as an aggregate. Consider simulated annealing and mutation-only genetic algorithm [18][17]. In both cases, the means of design generation is random mutation on one or more existing candidate design. Moreover, the means of design evaluation is not integrated into the optimization. Any means of design evaluation can be "plugged into" these optimizations. Therefore, these two distinct methods of design optimization have equivalent methods of design generation and evaluation. Where the two algorithms differ is in additional aspects of optimization. One might consider the differences to constitute the essential elements of the optimization. However, this is not typically the case; often descriptions of the design optimization bleed into describing the means for design generation.

2.4.1.1 Simulated Annealing Simulated annealing [18] is a gradient-free optimization method, very similar to other Metropolis-Hastings optimizations (a class of Monte Carlo algorithms). Inputs are presented to an objective function, which calculates an aggregates performance metrics producing a "score" for the utility of the inputs. As with other Monte Carlo optimizations the inputs are iteratively modified/mutated. If the mutated input performs favourably with respect to the objective function, the mutation is "accepted". Any mutation that improves the score is kept. Now, as a Metropolis-Hasting variant, if a mutation decreases the performance there then it is only probably reverted. A probability is calculated depending on the original score and the new score. An "unfavourable" mutation may still be accepted. The intention of doing so facilitates the optimization to "climb out" of local optima. Simulated annealing draws inspiration from thermodynamics by calculating the probability of acceptance according to a Boltzman distribution, proportional to the ratio between the difference in energy (performance metric) and a "temperature" hyperparameter. At the beginning of the optimization the temperature is high, so the probability of accepting a worse performing solution is also high. Over the course of the optimization the "temperature" is decreased, according to some predetermined cooling regiment, which in-turn decreases the probability of accepting a worse performing solution. Simulated annealing differs from a standard Metropolis-Hastings optimization since the probability of a 'misstep' decreases over-time. The reasoning for this is to begin the optimization in a regime that promotes exploration of the design space, and to end the optimization with reliable improvements narrowing in on a (ideally) global optima. This optimization process mimics that of annealing metal.

One may further augment simulated annealing by having an adaptive cooling regime. Rather than have the temperature decrease in a predetermined manner relative to the number of iterations, one can lower the temperature once a degree of equilibrium is reached. If the expected performance of sequential solutions vary around a mean value, rather than exhibiting a favourable trend, then the temperature is likely too "hot" for the optimization to progress. Similarly, the step-down in temperature can be predetermined or proportionate to the variation around a mean performance. An adaptive cooling regime is particularly useful if the performance of a solution is randomly distributed. In such a case, fluctuations in performance are to be expected. It is necessary to either sample the performance distribution of each proposed solution, or to monitor for a favourable trend in sampled performance. The former is more computationally

expensive, but is more appropriate later in the simulation. In Chapter 5: "Results", it will be demonstrated that variance introduced a stochastic simulation can interfere with simulated annealing. Original code for a simulated annealing process with an adaptive cooling regime is included in "Appendix: Design Optimization".

There are other modifications to simulated annealing that have been introduced. These include quantum annealing [19], which analogously adds "memory" to the optimization. The algorithm "remembers" more favourable previous solutions, and allows for spontaneous returns them occurring with a probability proportional to both the degree of similarity of solutions (or "distance" in the design space) and the difference in performance. This is analogous to quantum-tunnelling, a physical phenomenon in which particles spontaneously tunnel to more energetically favourable states. The probability of quantum-tunnelling also depends on the "favourability" difference (energy or performance) and the distance or dissimilarity between states.

2.4.1.2 Genetic Algorithms Genetic algorithms are a class of optimization that are inspired by Darwinian evolution. First, a "population" of candidate designs (or solutions) are generated randomly. Each design is evaluated according to its behaviour in the design environment (or its ability to solve the design problem). This evaluation yields a "fitness score" for each solution. Solutions that perform better than others, or perform better than some threshold, are allowed to be candidates for reproduction. The reproduction of well performing design candidates result in "descendant" designs, which are modifications of the original. Solutions that perform worse are "culled", and are replaced by "descendants" of better performing solutions. This process constitutes a "generation", a single iteration of the optimization. Over the course of the optimization process, features that contribute to favourable performance (fitness) are more likely to remain.

There are many particulars for the matter of solution reproduction. For example, whether a well performing solution is guaranteed to have a particular number of descendants, or if it merely is given a probability of reproduction proportional to its relative performance. Furthermore, the specific modifications of a descendant design can either be produced by random mutation or *feature cross-over* [20]. Random mutation modifies the values of each parameter or feature of the "ancestor" design by a small random amount. Feature cross-over select parameters or features from well performing designs and combines them with

features of other designs. This again can be done deterministically or stochastically and per feature. These two methods of modification of course mirror the biological analogue of sexual and asexual reproduction respectively. There are constraints on the utility of cross-over [21]. Cross-over is (more) appropriate when modifying 'macroscopic' features of a design; incorporating parts of a feature may result in non-functional features. This distinction is referred to by Fogel and Stayton as 'phenotypic' versus 'genotypic'. This is better explained with an example. Consider again the "evolved antenna" of Hornby and Globus [16]. One could formulate the cross-over phenotypically so to exchange macroscopic features, such as initial angle of the antenna. One could also formulate cross-over so to exchange exact locations of metal points, which would be more genotypic. Fogel and Stayton describe the benefits of cross-over deriving from a "large initial step size." The combination of a large step size and miniature design elements (exact locations of small metal pieces) would yield an inefficient design optimization. A different example that highlights the same constraint on the utility of cross over is *NeuroEvolution*. This is an optimization method where one trains an artificial neural network using evolutionary based methods [5]. If one included cross-over in training a convolutional neural network, it would be appropriate to exchange the (phenotypic) convolutional kernels between well performing networks. Convolutional kernels include multiple weights. It would be inappropriate to exchange only the (genotypic) weights from one well-performing network's kernels to those of another. This is because the weights in a kernel function together to emphasize particular features. Exchanging individual weights might still eventually produce an effective network, but this is only because cross-over is acting an ineffective form of mutation.

There are also means of augmenting genetic algorithms. For example, One can imagine how hyperparameters similar to that of "temperature" in simulated annealing might be used here. For instance, at the beginning of the optimization the reproduction of mediocre or poor performing design candidates might be allowed with some probability (according to some "temperature"). Over the course of the simulation this probability could decrease, thereby limiting the minimum allowed relative performance.

2.4.2 Selecting an Optimization Method

> "If I could remember the names of all these particles, I'd be a botanist" - Enrico Fermi

It seems there is a never ending stream of optimization methods, each with a novel name [10]. The implications of a chosen design optimization depend on the design problem, environment and sort of solution required. It is prudent to be mindful of which one chooses. For example, consider a stochastic design environments that yield a variable measure of design behaviour or performance. Such an environment would prove troublesome for methods of design optimization such as quantum annealing; if a design performs unlikely (but stochastically possible) well, quantum annealing optimization may repeatedly return to this actually mediocre design. What is therefore recommended is a careful and mindful process of selection. Knowledge of any theoretical groundings for design optimization methods can facilitate this. Moreover, it may be worth empirically comparing the efficacy of different methods. Methods that are problematic for particular design problems may be modified. For the example, consider again using simulated quantum annealing in a stochastic design environment. One method of mediating the unintended optimization behaviour would be to update the remembered performance of the design following a "tunnelling" event. The proposed mediation is merely an appropriate and mindful reaction, necessary when selecting methods of design optimization.

2.4.2.1 Classification of Methods of Design Generation

In a review of inverse design methods of generation used in nanophotonics, Molesky *et al.* [11] described two early works as "stand[ing] as archetypes for classification" of a dichotomy of methods of design generation, "genetic (evolutionary) or gradient-based approaches." This might be more generally stated as gradient-free and gradient-based approaches. For the sake of extending the analogy, this dichotomy splits the centre of the spectrum of design methods (the centre where inverse design is situated). To the 'intuition side' of centre are gradient based methods, since a 'differential understanding' of the design performance is required. Opposite it, on the random search side of centre are evolutionary methods of design generation, since evolutionary methods randomly mutate a design.

2.4.2.2 Comparison of Gradient-Based and Gradient Free Methods

There are costs and benefits associated with any particular method of design. Methods categorized as either gradient-based or gradient-free share many of these costs and benefits with other methods of the category. Sigmund, in On the usefulness of non-gradient approaches in topology optimization [17] argues against any benefit of gradient-free methods. He considers the commonly believed benefits of gradient-free methods to be,

1. Wider and more exploratory search.

2. Ability to handle discrete designs or inputs.
3. Avoid difficult to implement calculations of gradients.
4. Efficient and easy to parallelize

Sigmund then attempts to refute each of these points. In summary, the first and fourth points can be basically matched by a well implemented gradient-based optimization. The third and second points are precise issues that boil down to the problem that gradients are not always easy to come-by. However, as Sigmund argues,

> "[E]ven if gradients were not available, as could be the case when using an off-the-shelf analysis software, a finite difference-based gradient method would solve the problem more efficiently[.]"

For gradient-based methods Sigmund demonstrates that there is a much smaller number of function-calls (design evaluations) required to converged upon an optimal design. Furthermore, Sigmund argues, the suitability of gradient-free methods fails as the scope of design candidates increases; with a linear increase optimization parameters, there is an exponential increase in the number of possible design candidates (each a unique combination of the design parameters).

There are also arguments against gradient-based methods of optimization. A 2010 paper by Wu and Tseng [20], which Sigmund disparages in depth, claims that gradient-based methods produce different optimal solutions depending on what the initial candidate design is. They state that this is because gradient based methods are not "a global optimization method". This implies that a unique and truly optimal is not being reliably found by gradient-based methods. Shim and Manoochehri state that gradient-based methods fail to find global optima, getting stuck in local optima [22]. Furthermore, Gradient-free methods allow for more flexibility in optimization. For example, genetic algorithms can include diverse scores of fitness directly in the optimization process. The evaluation of a given generation can be given by one of many distinct objective functions. For example, absolute pass/fail tests can impose hard performance requirements on a design population. The objective functions need not be continuous or differentiable. This work does not argue that one category of optimization is better than the other. For most problems there will be more than one effective method to converge upon an optimal solution; it is more interesting (and important) to facilitate design optimizations for diverse design problems. What is relevant to the present work is the application of

surrogate models to various design optimizations. According to Sigmund gradient-based optimizations require fewer 'function evaluations' than gradient-free optimizations. It follows that gradient-free optimization methods would benefit more from the training and inclusion of surrogate models.

2.5 Inverse Design Evaluation

The compliment to design generation is design evaluation. Consider again the imagined spectrum of inverse design methods. If a designer were to rely entirely on human intuition, then designs would be evaluated on "gut feeling". Such a method of evaluation is unrobust. Such expertise requires a substantial amount of understanding and forethought. Few (if any) individuals can have regular success evaluating a design by intuition or qualitative judgment. On the other side of the spectrum of design methods, is the pole of random search. Stretching the analogy slightly, one can imagine that random evaluation of designs would require a full in-situ test of aggregated performance for each design candidate. Doing so would be expensive. Therefore, it is desirable to approximate or abstract measures of design behaviour or performance. Doing so requires understanding of the dynamics of the design environment, and so moves one towards inverse design methods of evaluation. Tests may be done so to limit the cost or scope of evaluation, i.e. targeted experiment. Beyond this one may use a simulation, either deterministic or stochastic. To abstract beyond experiment or simulation requires a *surrogate* method of evaluation. This work considers models that substitute actual or simulated evaluation of design candidates. Such models are called *surrogate models*.

2.5.1 Surrogate Models

George E. P. Box popularized the sentiment "all models are wrong, but some are useful" [23]. Though apocryphal, this quote nicely applies to surrogate models. The motivation for inverse design methods of design evaluation is to replace slow "traditional" models with likely less accurate but more useful models. A surrogate model attempts to capture the behaviour of a candidate design in the design environment It does so by predicting the outcome of a simulation or experiment in a simpler (or merely faster) calculation. Therefore, a surrogate model can provide an alternative means for determining (estimating) the behaviour of a design candidate in the design environment.

The primary benefit of using a surrogate model during a design optimization is speed

and evaluation cost. Surrogate models are typically a simpler or more available computation. Once a surrogate model is developed, per design evaluation involved in a design optimization will be faster. This should be considered more carefully. First, briefly, there is a basic dichotomy of costs in accounting, variable and fixed costs. Fixed costs are those that enable one to repeatedly perform a task or produce a good. Fixed costs are sometimes called "overhead" costs. Variable costs are incurred per task performed or unit produced [24]. Since the quantity of concern here is time, consider the following example. From the perspective of an employer, a factory work's lunch break is a fixed cost; the lunch break is essential for the worker to perform any work what-so-ever. Say that factory worker has to traverse the length of the factory to complete two sequential tasks using two machines at either end of the factory. The time required to walk from one machine to another is incurred *per* unit produced and is a variable cost.

There are additional time costs to developing or training a surrogate model. Namely, there are overhead costs required to produce a sufficient quantity of training data. Another overhead cost of developing a surrogate model is the computational time of training the model (between seconds and hours). One might determine it to be "worth" incurring these overhead time costs if it will be made up for by a sufficient decrease in the variable time costs per optimization iteration. The speed-up per optimization iteration, over the course of the design optimization process, must be greater than the time required to produce the surrogate model. Figure 1 illustrates a simple trade-off between the costs of using the source of training data (simulation or experiment) and developing and using a surrogate model. If the time-cost of optimization is small (or the number of iterations is small), then there is no benefit of time in using a surrogate model. However, beyond a minimum number of design optimization iterations, there is an overall benefit to the time required for the design optimization.

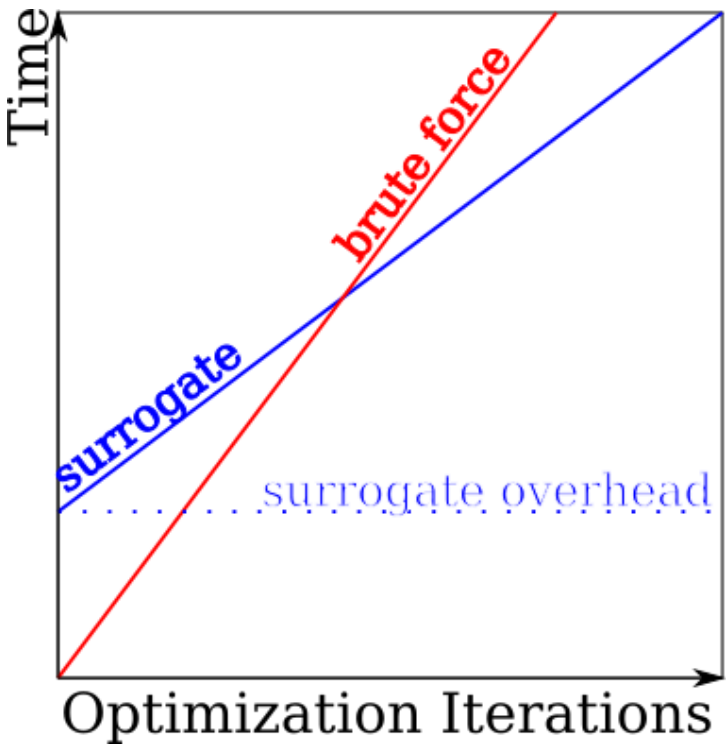


Figure 1: Illustration of the trade-offs in time costs. The per iteration costs of using a standard simulation must be greater than that of a surrogate model. Using a "brute force" approach might be faster for design optimizations with few iterations.

There are a number of ways to reduce the overhead costs relative to the net time cost of a design optimization. First, the production of training data can typically be done in parallel. This is in contrast to the many design optimization methods. Even when supplemented using parallelization, these have an essential and unavoidable sequential aspect (e.g. genetic algorithms, adversarial generation, other gradient-based methods). Furthermore, using a surrogate model, one can directly predict a performance metric that would otherwise be derived from the simulation or experimental results. This skips the per iteration time costs of interpretation or processing of design behaviour.

Lastly, another overhead costs of developing a surrogate should be considered as well. It is the designer's time required to plan, implement and test a workflow that converts output from the data source to a usable format, pre-processes the data, then generates and saves the model. Since this matter is the topic of a Master's thesis, the amount of developers time spent dwarfs the vast majority of time-intensive computational design optimizations. This does not preclude the time benefits of using a surrogate model.

There are some design optimizations that would be otherwise untenable without variable cost speed-ups. Furthermore, once a workflow is developed for producing one kind of surrogate model, it may be re-used or modified for another. Lastly, there is research in automated design that suggests the possibility of automating the production of surrogate models [25]

2.5.2 Other Benefits of Surrogate Models

Not withstanding any improvements in overall time costs, there are a limited number of other benefits of using a surrogate model. First, surrogate models can be more easily integrated with design optimization processes. As will be discussed in Chapter 5: "Results", the data source demonstrated in this work is written in C++, Fortran and a dialect called "Mortran" [4]. Without knowing these languages, a large amount of associated programs were developed to run the simulation and process the results. This contrasts to the surrogate model that is (the scikit-learn implementation is) written in python [26]. Extending this idea, imagine that a design optimization iterates towards a region of the design space, which is computationally expensive or intractable (e.g. a region where calculations are difficult to converge upon.) Some simulations may either fail, or enter a never-ending attempt to converge upon a design that may be unviable. Whereas a simple surrogate model could either return an inadmissible result or run an error. Either of which can be simply handled as exceptions in the design optimization.

Furthermore, using a surrogate model may allow for the use of design optimization methods that are otherwise unavailable. One potentially powerful application of this is using differentiable surrogate models to replace non-differentiable models of evaluation. Doing so allows for the use of gradient-based optimization methods. For example, one could use a differentiable neural network for a surrogate model, to predict performance of candidate designs over the course of a non-differentiable stochastic simulation. In this case the gradient of performance relative to the parameters of the design can be calculated using the surrogate model. Therefore gradient descent, a gradient-based optimization method, could be used to modify the candidate design in the "direction" of optimal performance.

2.5.3 Multi-Objective Optimization

Design projects often have many goals, i.e. design objectives. Difficulty can arise if the pursuit of one goal interferes with one or more of the other goals. A classic example

is designing structural components to be as strong as possible while using the least possible material. An additional difficulty of balancing competing goals is that they can be difficult to compare. For example, a designer of an airplane wing might have to decide how much additional strength is worth having the wing be 10% heavier overall.

2.5.4 Utility and Accuracy

The accuracy and goodness-of-fit of a model impacts its usefulness for design optimizations. For example, model accuracy may vary among regions of the design space. If so, a model may guide the optimization unfairly away from optimal subspaces. In another case, the accuracy of a model can be limited in precision. Logically, the useful course of the optimization process would end at the precision limit of the model. The accuracy of a (surrogate) model, used for design, limits the useful extent of optimization.

2.5.4.1 Utility of Models with Rejectable P-Values As described in Chapter 4: "Machine Learning Methods", in the "Probabilistic Loss" section, one can measure goodness-of-fit using likelihood. By bootstrapping a histogram of results for a single input, the likelihood for a given value at each point is calculated. Using the likelihood statistic and histogram, a P-value is estimated equal to the probability that a less characteristic sample be created by simulation. P-Values are often used to reject hypotheses. Here one can use P-values to reject the predictions of a surrogate model as uncharacteristic. Merely that the predictions of a model can be distinguished from results of a simulation, does not mean that the model is not useful for design. Many design schemes (and ensemble algorithms) can make use of inaccurate or weak models. What might be more important is that the *expected value* of predicted and actual design performance are equivalent.

Variance in predicted or actual performance may interfere with the design optimization. The potential consequence of this ranges from an inconvenient extension of the optimization to making the design optimization entirely unviable. Results of this sort are included in Chapter 5: "Results", in the "Brute Force Optimization" section. In that case the random forest regressor used for a surrogate model appears to have trained towards (or nearly) a mean value. Therefore, the predicted performance has much less variance between pixels (with similar surroundings) than that variance one would expect from the EGSnrc simulation. However, since the surrogate model predicted a mean value

(and so has much less variance than the simulation data source), the design optimization using that surrogate model converged upon a viable design. Whereas using EGSnrc, the simulation data source had a higher variance in the design behaviour, which interfered with the design optimization. The design optimization using the simulation data source eventually converged upon a viable solution. Using the source simulation required more time per design optimization iteration than the surrogate model, as is expected. It also required a larger total number of iterations required to produce a viable design. One can imagine a circumstance in which the efficiency advantages per iteration and in total number of iterations could mean the difference between a tenable and untenable design optimization.

2.5.4.2 Accounting for Variance in the Design Environment In the case previously discussed, using a surrogate model in a design optimization improved the overall efficiency. However, the benefits of using a surrogate model "trained to the mean" can have negative consequences as well. Imagine designing the shape of a building to resist wind. The wind where the building is to be constructed comes, at varying times, from each direction with equal probability of direction and intensity. Therefore, the *expectation* would nominally be that there is no wind whatsoever (it all "cancels out"). This is of course a terrible design approach. It illustrates how, in some cases, surrogate models that regress to a mean may incorrectly influence a design process. A more reasonable surrogate model would predict a distribution of design performance or behaviour, reflecting an estimated distribution of design environment conditions. Such a surrogate model would again produce variance that could interfere with a design optimization. However, depending on the design goals, it may be a necessary correction or complimentary tool to a "regress-to-the-mean" type surrogate model.

Engineered designs often require *factors of safety*, a standard of performance multiple times that of what is expected within the design environment or over the course of the design's lifetime. For example, if a building is expected to be subjected to winds of 50 km/h then it may be chosen to design that building to be capable of withstanding 150 km/h wind intensity. In a design optimization, an appropriate surrogate model may even need to predict a distribution of design environment circumstance that exceeds that of what is expected from the data source. This is a form of extrapolation, particularly tricky if the training data is provided from real-world sources. Factors of safety may exceed what can be readily observed through experiment (or even simulation!) Such

circumstance are beyond the scope of this work. It is an interesting matter of research that machine-learning models can usefully extrapolate emergent phenomena beyond that observed in its training data [27].

2.6 General Considerations in Inverse Design

2.6.1 Biases in Design

Since models will perform better on training data, an analogy can be drawn (in a casual way) to biases of human intuition; designs previously seen will be more accurately considered than entirely novel ones, either in a positive or negative sense. Consider an example case of designing an airplane wing. If using a surrogate model, one may choose to limit the design environment to wing-shaped objects moving through air within a range of pressures. Doing so would simplify the phenomena predicted, and likely improve *observed* generalization of the surrogate model. Similarly, the observed performance of a design generation algorithm would be improved since one expects a higher average performance per design. However, selecting the training data in this way could reduce the prediction of the surrogate model on uncharacteristic designs. Furthermore, the likelihood of a generated design deviating substantially from the training set may be limited. This is not withstanding surrogate model generalization or design space exploration by the generator algorithm. Rather, this example intends to highlight a trade-off in design evaluation and generation; selective searches may be more reliable, but may not find or "appreciate" novel improvements.

2.6.2 Rigorous Design Evaluation

A final design produced from an optimization with an (in)accurate surrogate model may not be representative of a true optimal solution. In a numerical methods course for engineering students, nicknamed by the professor "How not to get sued", the professor advised the author that if he were ever to design something, to leverage every possible tool of evaluation at his disposal (up to and including the kitchen sink). Since the accuracy and generalization of machine learning methods can be unpredictable, there exist genuine risk in trusting the predictions of such a model. Any design produced under the judgment of a surrogate model should be validated by the original model. A careful designer would include validation from other unrelated methods as well; it is possible for a surrogate model to mimic both accurate and inaccurate simulation results.

However, design results using an uncharacteristic surrogate model can be used as a starting place for a subsequent design optimization using the simulation itself. Doing so could reduce the overall time required for optimization.

3 Overview of EGSnrc

> "The correct simulation of an electromagnetic cascade shower can be decomposed into a simulation of the transport and interactions of a single particle, along with some necessary bookkeeping."
> - EGS4 Manual p.13 [28]

EGSnrc [4] is a Monte Carlo simulation software package of coupled electron and photon transport. Each of the phenomena simulated must be described as a probabilistic model. EGSnrc simulates a diversity of distinct phenomena. It has been the task of the developers of EGSnrc, Tessier *et al.*, and the computational radiation physics community to reduce the complexity of these physics into a self-consistent probabilistic model. This may be viewed, broadly, as a systematic process. First, a physical description of a phenomenon of sufficient sophistication for its purpose is selected. Second, the description is simplified such that under reasonable assumptions it is summarized by a single or set of algebraic equations. Based on the distribution of possible inputs (or initial conditions) a distribution of possible outcomes is generated. This generally involves the derivation of a differential cross-section with respect to all possible output variables. If successful, the resulting model will capture many reduced phenomena under self-consistent assumptions into a probabilistic multi-physics model suited for a Monte Carlo simulation.

3.1 Monte Carlo simulation

The EGSnrc manual provides a concise account of one simulation iteration,

> "particles are "born" according to distributions describing the source, they travel certain distances, determined by a probability distribution depending on the total interaction cross-section, to the site of a collision and scatter into another energy and/or direction according to the corresponding differential cross-section, possibly producing new particles that have to be transported as well. This procedure is continued until all particles are absorbed or leave the geometry under consideration."

For a single particle travelling through a medium, the mean free path, λ, is described by [28],

$$\lambda = \frac{M}{N_a \rho} \frac{1}{\sigma_t} \tag{2}$$

Where the first fraction accounts for the spatial frequency of sites of interaction (atoms). The first fraction is equal to the molecular weight, M, divided by the product of Avogadro's number, N_a, and the material density ρ. The second fraction, σ_t, here represents the atomic total interaction cross-section. A "cross-section" [29] is an effective cross-sectional area, weighted by the probability of interaction. The cross-section accounts for the probability of interaction as an equivalent fractional area (if the interactions were guaranteed but dependent on position). It was originally named this since it similarly describes the ratio within some incident area, that is occupied by a target.

Within a differential of distance travelled, dx, the probability of interaction is

$$P = \frac{dx}{\lambda} \tag{3}$$

Therefore, between some initial position x_0 and the location of the next interaction, x, the number of mean free paths travelled is included below. From this equation, the distance to the next interaction may be sampled using the transformation method.

$$N_\lambda = \int_{x_0}^{x} \frac{1}{\lambda(x)} \cdot dx \tag{4}$$

Solving the above integral, according to the EGSnrc manual is "numerically intensive" [4]. Therefore, the "fictitious cross-section" is employed. This method modifies the total macroscopic interaction cross-section, Σ_{tot}.

$$\Sigma_{tot} = \frac{1}{\lambda} \tag{5}$$

A fictitious interaction macroscopic cross-section is added to the total macroscopic interaction cross-section.

$$\Sigma' = \Sigma_{tot}(x) + \Sigma_{fict}(x) \tag{6}$$

This is done so that the sum always equals some constant,

$$\Sigma_{fict}(x) \equiv \Sigma' - \Sigma_{tot}(x) \tag{7}$$

The augmented cross-section is used as a proportionality constant relating the logarithm of the value sampled on the uniform range, and the path-length.

$$s\Sigma' = -\ln \zeta \tag{8}$$

This will produce more frequent interactions. To recover the correct rate of real interactions, interactions are rejected with the probability

$$P(\text{reject fictitious}) = 1 - \frac{\Sigma_{tot}}{\Sigma'} \tag{9}$$

This means the sampling efficiency goes down (in terms of values rejected). The "fictitious cross-section method" is identical to the "acceptance-rejection method" described by Cowan in Statistical Data Analysis [30]. In this method numbers are sampled in the following way. First, a number is uniformly sample from across the range of possible values. Next, a second number is uniformly sampled between 0 and the maximum probability density, f_{max}. If the second number is greater than the probability density at/of the selected number (the first number, sampled on the range of possible values) then it is rejected. Therefore, the rate of accepting a particular possible value is proportional to the ratio between its probability density and the maximum probability density.

For curves with a tall and narrow peak, there is a large discrepancy between a uniform probability density and that of the desired pdf. So, it requires many 'rejects' for a single successful sample according to the desired distribution. Cowan summarizes a method for improving the efficiency this method [30].

> "For a highly peaked density function the efficiency may be quite low, and [...] one can improve the efficiency by enclosing the p.d.f. $f(x)$ in any other curve $g(x)$ for which numbers can be generated according to $g(x)/\int g(x')dx'$ "

To improve the sampling efficiency, one can instead generate numbers across the range of possible values (the first number sampled) according to a different probability distribution $g(x)$. It is required that this other curve, unnormalized, envelops the desired sampling distribution, ideally with a narrow margin. Furthermore, when normalized into a pdf, the second curve $g(x)$ should be able to be sampled by a more efficient method like the transformation method. Therefore, fewer 'rejects' will be necessary. EGSnrc uses this technique for simulating Compton scattering, as will be discussed in section 3.3.2.6.

After transporting the particle this random distance, an interaction must be selected. This is done by segmenting a uniform range by relative cross-section for each possible interaction. By randomly sampling on the uniform range, one may select an interaction by

$$F(i-1) < \zeta < F(i) \tag{10}$$

Where ζ is the randomly sample value and $F(i)$ represents the leading boundary of the uniform range segment representing the interaction i. Based on the type of interaction, the number and type of outgoing particles is pre-determined. What must be determined is

the state of each of these particles, which is accomplished by sampling with the differential cross-section.

$$f(\vec{\mu}) = \frac{d^n\sigma}{d\vec{\mu}} \tag{11}$$

According to the EGS4 manual, the differential cross-section when normalized "has the properties of a joint distribution function." Therefore, the joint distribution may be factored into conditional probabilities,

$$f(\mu_1, \mu_2, ..., \mu_n) = h_1(\mu_1) \times h_2(\mu_2|\mu_1)... \times h_n(\mu_n|\mu_1, \mu_2, ..., \mu_{n-1}) \tag{12}$$

Where the conditional probabilities are calculated,

$$h_m(\mu_m|\{\mu\}_{m-1}) = \frac{g_m(\{\mu\}_m)}{g_{m-1}(\{\mu\}_{m-1})} \tag{13}$$

The above equation includes some simplified notation,

$$g(\mu_1, \mu_2, ..., \mu_m) \equiv g(\{\mu\}_m) \tag{14}$$

Furthermore, g_m is the marginal probability distributions, successively calculated by integrating according to each dimension.

$$g_m(\{\mu\}_m) = \int g(\{\mu\}_{m+1}) \cdot d\mu_{m+1} \tag{15}$$

Therefore, having calculated conditional probabilities, one may proceed with sampling methods appropriate for single dimensions. The method used by EGS4, the predecessor of EGSnrc, is a "combination of the "composition" and "rejection" techniques" presented in the EGS4 manual.

3.2 Included Physical Phenomena

The following sections detail the physics included in EGSnrc and demonstrates the reduction of those physics into a probabilistic form. Many of the improvements to the probabilistic models in EGSnrc are described here. However, the reader is encouraged to read the documentation of EGSnrc, which documents many impressive additions to the field that are eloquently simulated in their software.

The name of EGSnrc and that of its predecessor EGS4 are variants on the acronym, **E**lectron **G**amma **S**hower [4]. Another term for this phenomenon is *electromagnetic cascade shower* [28]. Initiated by incident radiation, the "cascade shower" is a chain reaction

whereby interactions of incident radiation with the target material produce secondary particles at sufficient energies that they may also "reproduce". For example, an incident electron may, and is typically, slowed through *Bremsstrahlung* scattering, changing the direction of the electron and releasing a gamma ray. This secondary gamma-ray, depending on its energy may, according the EGS4 manual, "return energy to the system in the form of electrons" [28] either by *electron-positron pair production* or *Compton scattering*. The result of either process increases the number of free particles by one; an additional electron in the case of Compton scattering, and an additional positron-electron pair less a gamma ray in the case of pair production. The initial incident radiant energy is spread through the target material among an increasing number of free particles. Eventually, additional particles yielded from high-energy interactions are at too low an energy to produce additional particles. Low energy particles are more likely to lose energy through inelastic collisions and non-radiative losses following photo-absorption. In the EGSnrc simulation, particles eventually cease to be simulated once they fall below a minimum threshold energy.

3.3 Scattering of Light

A relevant distinction may be made among light-scattering phenomena; scattering that changes the wavelength of the radiation and that which does not. The former is referred to as "inelastic scattering" and the latter "elastic scattering" [29]. The following two sections will investigate these two categories of scattering.

Furthermore, light-scattering phenomena may be ordered according to the energy range over which is has the largest impact. This is to say that the scattering of photons depends on the energy of the radiation, as well as the scattering medium. There are discrete changes in the light-scattering phenomena with continuous changes of energy.

3.3.1 Elastic Scattering

Elastic scattering is scattering that does not change the kinetic energy of radiation. For light, three scattering phenomena are considered distinct: Rayleigh scattering, Mie scattering and geometric scattering [31]. These three phenomena dominate the inelastic scattering regime when the radiation wavelength is smaller, similar or larger than the scattering particle, respectively. Only Rayleigh scattering is implemented in EGSnrc.

3.3.1.1 Dipole Approximation of Rayleigh Scattering

Example 11.1 from Griffith's *Introduction to Electrodynamics* (p.471) describes how these three phenomena may be described similarly, though with certain approximations applying at difference wavelengths [32]. This example is here mathematically demonstrated, for Rayleigh scattering. Beginning with the definition of the Poynting vector,

$$S = \frac{1}{\mu_0}(\mathbf{E} \times \mathbf{B}) \tag{16}$$

Where the two fields have the associated potentials

$$\mathbf{E} = -\nabla V - \frac{\partial \mathbf{A}}{\partial t} \tag{17}$$

$$\mathbf{B} = \nabla \times \mathbf{A} \tag{18}$$

As described by Griffith (p.358), S is the "The energy per unit time, per unit area, transported by the [transposed electric and magnetic] fields" [32]. He also concisely refers to S as the "energy flux density". From a source, the power passing through a sphere of radius r is

$$P(r,t) = \oint S \cdot d\mathbf{a} = \frac{1}{\mu_0} \oint (\mathbf{E} \times \mathbf{B}) \cdot d\mathbf{a} \tag{19}$$

For a perfect dipole source, the following assumptions are made:

$$d \ll \frac{c}{\omega} \ll r \tag{20}$$

That the distance from the observer, r, to the centre of the dipole is much greater than the observed wave length, $\frac{c}{\omega}$ and that the observed wavelength is greater than the physical size of the dipole, d. The latter assumption is what distinguishes Mie and Rayleigh scattering, since Mie scattering occurs in the is in the presence of large particles (relative to the scattered wavelength). In these circumstances, the potential is found to be,

$$V(r,\theta,t) = \frac{p_0 \omega}{4\pi\epsilon_0 c}\left(\frac{\cos\theta}{r}\right)\sin\left(\omega\left(t - r/c\right)\right) \tag{21}$$

and the vector potential is,

$$\mathbf{A}(r,\theta,t) = -\frac{\mu_0 p_0 \omega}{4\pi r}\sin\left(\omega\left(t - r/c\right)\right)\hat{\mathbf{z}} \tag{22}$$

Averaged over one period, the intensity is

$$\langle S \rangle = \frac{\mu_0 {p_0}^2 \omega^4}{32\pi^2 c}\frac{\sin^2\theta}{r}\hat{\mathbf{r}} \tag{23}$$

Therefore the total power radiated to a point at radius r is,

$$\langle P \rangle = \frac{\mu_0 {p_0}^2 \omega^4}{12\pi c} \tag{24}$$

This shows the characteristic result that Rayleigh scattering depends highly on the frequency of light scattered.

3.3.1.2 Thompson Scattering: Classical Theory

From the total power emitted from the atom, the Thompson scattering cross-section can be derived [33].

$$\sigma_T = \frac{\text{total energy scattered per second}}{\text{energy incident per square meter per second}} \tag{25}$$

The differential Thompson scattering cross-section per electron is [34],

$$\frac{d\sigma_T}{d\Omega} = r_e^2 \frac{(1+\cos^2\theta)}{2} \tag{26}$$

3.3.1.3 Corrected Cross-sections for Coherent Scattering

Thompson scattering was originally derived for Thompson's model of the atom. As more modern models were accepted, the scattering theory was updated with corrections. The resulting approximation depends on "form-factors" and "scattering-functions" [34]. The model that EGSnrc "inherited" defines the cross-section for Rayleigh scattering as

$$\sigma_{coh} = \frac{3}{8}\sigma_T \int_{-1}^{1} \left(1+\cos^2\theta\right) \left[F(x,Z)\right]^2 d(\cos\theta) \tag{27}$$

And had a differential cross-section of [4],

$$\frac{d\sigma_R}{d\Omega} = r_0^2 \frac{(1+\cos^2\theta)}{2} [F_T(\mathbf{q})]^2 \tag{28}$$

Where $F_T(\mathbf{q})$ is the form factor, and q is the momentum transfer vector equal to,

$$q = k\frac{1-\cos\theta}{2} \tag{29}$$

There, k is the ratio between photon energy and electron rest mass. The form factor is defined as the Fourier transform of the total electron density,

$$F(\mathbf{q}, Z) = \int \rho(\mathbf{r}) e^{i\mathbf{q}\cdot\mathbf{r}} d\mathbf{r} \tag{30}$$

but can also be expressed as [34] [35]

$$F(\mathbf{q}, Z) = \sum_{n=1}^{Z} \langle \Psi_0 | e^{i\mathbf{q}\cdot r_n} | \Psi_0 \rangle \tag{31}$$

Where Ψ_0 is the ground state of the electron wave function,

3.3.1.4 Sampling Outcomes of Rayleigh Scattering

When sampling the phenomena, the maximum possible momentum transfer is taken to be $q_{max} = k$ (i.e. $\theta = 0$). Furthermore, the form factor is normalized and used for a probability density function in the following manner [4].

$$f(\mathbf{q}) = \frac{[F_T(\mathbf{q})]^2}{\int_0^{\mathbf{q_{max}}^2} d\mathbf{q}'^2 [F_T(\mathbf{q}')]^2} \tag{32}$$

$(1 + \cos^2\theta)/2$ is used for a rejection function. This can be interpreted physically. The probability of a collision between the two particles depends on the ratio between the "cross-section" of the scattering target and the solid angle projected to a distance r. At each possible value of momentum transfer, the effective size of scattering target varies according to the form factor. Therefore, the sampled value of momentum transfer is distributed according to the form factor.

3.3.2 Inelastic Scattering: Compton Scattering

Inelastic scattering is any process by which diverts radiation from its incident direction and changes its kinetic energy. Common types of inelastic scattering for photon radiation are: Brillouin, Raman, and Compton scattering. Only Compton scattering is simulated in EGSnrc. Compton scattering is a scattering interaction between an incident photon and a bound electron in a target material [4]. The photon is scattered off of the electron, lowering its energy, and transfers sufficient energy to expel the electron from its shell [36]. By a classical definition of *elastic and inelastic* collisions, Compton scattering is classically considered an *inelastic* collision, since the photon loses energy. In contrast, in a relativistic frame, if the total kinetic energy is conserved during a process, it is therefore an *elastic* process [32][29]. However, in general, Compton scattering occurs for bound electrons. To release the bound electron some kinetic energy is consumed. Therefore, Compton scattering off of a bound electron is *inelastic* [37].

3.3.2.1 Compton's Derivation

Compton's original paper explains the effect by applying conservation of energy and momentum in a relativistic frame. By demonstrating that, Compton demonstrated that light must carry momentum. Momentum may only be carried by localized quanta. Therefore, Compton demonstrated that light is insufficiently described as a wave. Griffiths in example 12.9 of Introduction to Electrodynamics [32], introduces the topic in a relativistic electrodynamic frame, for the *elastic* scattering of

a photon off of a free and stationary electron. This provides a similar description to Compton's original.

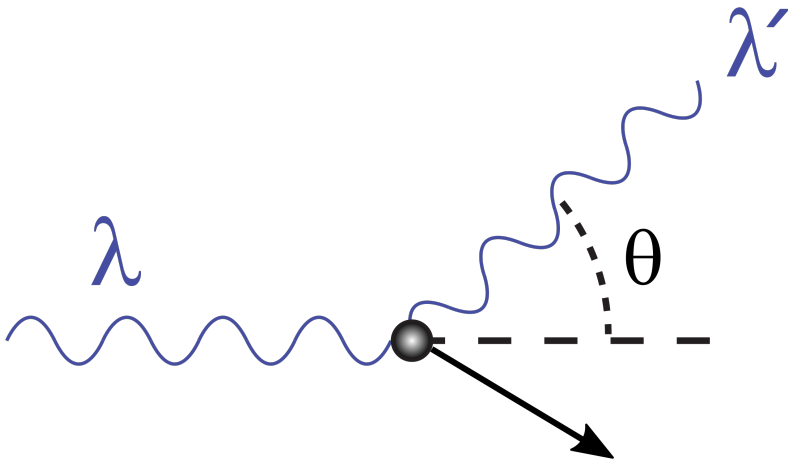


Figure 2: Diagram of Compton scattering [38]: A photon is scattered of off an electron. Energy is transferred to the electron, lower the frequency of the photon [4]

Above [38], illustrates the scattering of the incident photon off of a stationary electron. For momentum to be conserved, the system's total momentum perpendicular to the incident direction must be zero.

$$p_e \sin\phi = \frac{E'}{c}\sin\theta \tag{33}$$

Similarly, the total momentum of the system in the incident direction must be equal to the original momentum of the incident photon.

$$\frac{E}{c} = p_e \cos\phi + \frac{E'}{c}\cos\theta \tag{34}$$

The momentum of the ejected photon is of course,

$${p_e}^2 = {p_e}^2 \cos^2\phi + {p_e}^2 \sin^2\phi \tag{35}$$

Where the second term has been shown to be equivalent to the perpendicular momentum of the scattered photon. So $\cos\phi$ is found to be

$$\cos\phi = \sqrt{1 - \left(\frac{E'}{p_e c}\sin\theta\right)^2} \tag{36}$$

So,

$$E_0 = \sqrt{p_e^2 c^2 - (E'\sin\theta)^2} + E'\cos\theta \tag{37}$$

Following Griffith's derivation, simplified form of conservation of total momentum is

$$p_e^2c^2 = E_0^2 - 2E_0E'\cos\theta + E'^2 \tag{38}$$

Following the interaction, there is the kinetic energy and mass energy of the electron and the lowered energy of the scattered photon. The electron's kinetic energy is found in the simplified equation for conservation for momentum.

$$E_0 + m_ec^2 = E' + \sqrt{m^2c^4 + E_0^2 - 2E_0E'\cos\theta + E'^2} \tag{39}$$

When expanded and simplified, the final energy of the photon is found,

$$E = \frac{1}{\frac{1}{E_0} + \frac{1-\cos\theta}{mc}} \tag{40}$$

Griffith finishes the derivation in terms of photon energy,

$$\lambda = \lambda + \frac{h}{mc}(1 - \cos\theta)$$

and ends his example on p.541 [32] "The quantity (h/mc) is called the **Compton wavelength** of the electron."

3.3.2.2 Klein-Nishina Formula The Klein-Nishina formula describes the differential cross-section of Compton scattering. To derive their equation, Klein and Nishina began with the Dirac equation, which extends the Schrödinger equation to a relativistic frame [39].

$$\left(\frac{i\hbar}{c}\gamma^\mu\partial_\mu - m\right)\psi = 0 \tag{41}$$

The form that Klein and Nishina derive for the differential scattering cross-section is

$$\frac{d\sigma_{KN}}{dcos\theta} = \pi r_0^2 Z\left(\frac{k_c}{k}\right)^2\left[\frac{k_c}{k} + \frac{k}{k_c} - \sin^2\theta\right] \tag{42}$$

Where k_c is the energy of the scattered photon, which travels at an angle θ relative to the incident direction after the event. k is the initial energy of the photon. For their derivation, Klein and Nishina assumed that the electron is initially at rest and that the binding energy to the atom is negligible.

3.3.2.3 Interpretation using the Impulse Approximation Similar to the form factor correction of Rayleigh scattering to Thompson scattering, the differential cross-section for incoherent scattering can be expressed by the product of a correcting function and the Klein-Nishina cross-section.

$$\frac{d\sigma}{d\Omega} = \frac{d\sigma_{KN}}{d\Omega'}S(\omega, \theta, Z) \tag{43}$$

Where $S(\omega, \theta, Z)$ is the **incoherent-scattering function**. Ribberfors and Berggren describe this having a magnitude that is "a measure of electron-binding." [40]. Their work aims to correct the free-electron derivation of Klein-Nishina, by reformulating the common experimentally measured Compton profiles, $J(p_z)$, using a relativistic impulse approximation. Ribberfors, in his 1975 paper, derives the differential cross-section from the total relativistic cross-section by applying the *Impulse approximation*. Ribberfors describes the Impulse approximation [37],

> "it is assumed that the energy transfer is so large that binding effects for the electrons may be neglected and that the final state of the excited electron may be approximated by a plane-wave state. This means that the scattering process can be viewed as a photon being scattered inelastically against a stationary wave packet of superimposed plane-waves states."

Their work aimed to improve established derivations for differential cross-section under this non-relativistic Impulse approximation,

$$\left[\frac{d^2\sigma}{d\omega' d\Omega'}\right]_{IA} = \frac{{r_0}^2}{2}\frac{\omega'}{\omega}\frac{m}{q}\left(1+\cos^2\theta\right) J(p_z) \tag{44}$$

Where $J(p_z)$, the Compton profile, was calculated

$$J(p_z) = 2\pi \int_{|p_z|}^{\infty} dp\ p \cdot \rho(p) \tag{45}$$

and $\rho(p)$ is the distribution of momentum prior to the collision.

By creating a relativistic impulse approximation, Ribberfors preserves the Compton profile,

$$\frac{d^2\sigma}{d\omega' d\Omega'} \approx \frac{{r_0}^2}{2}\frac{\omega'}{\omega}\frac{m^2}{\left|\vec{k}-\vec{k'}\right|}\frac{\bar{X}(R,R')}{(m^2+{p_z}^2)^{1/2}} \cdot J(p_z) \tag{46}$$

Ribberfors and Berggren describe the last two factors [40] as,

> "$J(p_z)$ gives rise to a broadening of the Compton line because of the motion of the electron before collision and $\bar{X}/(m^2+{p_z}^2)^{1/2}$ reflects relativistic effects at the scattering process. "

This approximation is made because $\bar{X}$ is a "slowly varying function", defined by

$$\bar{X}(R,R') = \frac{R}{R'} + \frac{R'}{R} + 2m^2\left(\frac{1}{R}-\frac{1}{R'}\right) + m^4\left(\frac{1}{R}-\frac{1}{R'}\right)^2 \tag{47}$$

The variables R and R' are defined as,

$$R = \omega \left[\left(m^2 + {p_z}^2 \right)^{1/2} + \frac{(\omega - \omega' \cos\theta)\, p_z}{\left| \vec{k'} - \vec{k} \right|} \right] \tag{48}$$

$$R' = R - \omega\omega' (1 - \cos\theta) \tag{49}$$

Where p_z is "the projection of the momentum of the initial electron on the scattering vector $\vec{q} = \vec{k'} - \vec{k}$, i.e. $p_z = \vec{p} \cdot \vec{q}/q$." This projection can be expressed as

$$p_z = \frac{\omega\omega' (1 - \cos\theta) - m(\omega - \omega')}{\left| \vec{k} - \vec{k'} \right|} \tag{50}$$

and,

$$\left| \vec{k} - \vec{k'} \right| = \left[\omega^2 + (\omega')^2 - 2\omega\omega' \cos\theta \right]^{1/2} \tag{51}$$

Once the double differential cross-section is derived using the relativistic Impulse approximation, Ribberfors and Berggren then make simplifications where "relativistic effects are less important". By setting the projection of initial electron momentum onto the scattering vector, p_z to zero, the double differential cross-section simplifies to a "Klein-Nishina type of expression",

$$\left(\frac{d^2\sigma}{d\omega' d\Omega'} \right)_{KN} = \frac{r_0^2}{2} \frac{\omega'}{\omega} \frac{m}{\left| \vec{k'} - \vec{k} \right|} \bar{X}_{KN} J(p_z) \tag{52}$$

This can be more easily integrated to be consistent with the previously stated equation for Klein-Nishina differential cross-section. Next, the Klein-Nishina differential cross-section is factored out of the original double differential cross-section from the relativistic impulse approximation. This has a simplified form,

$$\frac{d^2\sigma}{d\omega' d\Omega'} = \left(\frac{d\sigma}{d\Omega'} \right)_{KN} F(\omega, \theta, p_z) J(p_z) \tag{53}$$

Where $F(\omega, \theta, p_z)$ is a catch-all for all terms and factors required or produced from the factoring. Integrating with respect to ω'

$$\frac{d\sigma}{d\Omega'} = \left(\frac{d\sigma}{d\Omega'} \right)_{KN} \int F(\omega, \theta, p_z) J(p_z) \cdot d\omega' \tag{54}$$

The EGSnrc manual, as well as Bursa *et al.* [41] modify this so to leave $F(\omega, \theta, p_z)$ in terms of p_z. To do so, a change of variables is performed in the integral. The additional differential factor, dp_z/dw' can be incorporated into $F(\omega, \theta, p_z)$ Therefore, by equation 43,

$$S(\omega, \theta, Z) = \int F(\omega, \theta, p_z) J(p_z) \cdot dp_z \tag{55}$$

3.3.2.4 Improvements to Simulating Compton Scattering in EGSnrc It is to the incoherent scattering function, $S(\omega, \theta, p_z)$ that Tessier *et al.* modify the approximation of Ribberfors and Berggren. The EGSnrc manual claims [4] that the resulting approximation improves:

> "In EGSnrc we have included binding effects and Doppler broadening according to the impulse approximation (IA). The IA assumes that the potential in which the target electrons move is constant so that their states can be represented by plane waves."

First these two phenomena will be defined. Then their implementations, and how these are consistent with the impulse approximation will be explained.

3.3.2.4.1 Doppler Broadening and the Doppler Effect The Doppler effect describes the *relative* difference between the observed wavelength of incident light between different frames. A frame in which a light source is approaching will observe a decreased wavelength, relative to the frame of the source. Similarly, in a frame in which the distance between observer and source is increasing, an observer will measure a longer wavelength relative to the frame of the source. This effect is described by classical mechanics, but is correctly extended by special relativity. The equation for the ratio between the wavelength in observer frame and source frame is here derived.

Consider a light source and receiver moving away from one another at a speed $\vec{v}$. Initially, when a wavefront has reached the detector, the following wavefront is λ_s away. However, since in the source frame the detector is moving way, by the time the next wavefront hits the detector it will have travelled $\lambda_s + \vec{v}t_s$.

$$\lambda_s + vt_{r,s} = ct_{r,s} \tag{56}$$

Where t_s is the time for one oscillation of the wave. This is stated,

$$t_{r,s} = \frac{\lambda_s}{c - v} \tag{57}$$

Therefore,

$$\lambda_r = \lambda_s \frac{c}{1 - \frac{v}{c}} \tag{58}$$

This is further described by considering time dilation in the receiver's frame.

$$t_r = \frac{t_s}{\gamma} = t_s\sqrt{1 - \beta^2} \tag{59}$$

It is found that the relative Doppler shift is [31],

$$\frac{\lambda'}{\lambda} = \sqrt{\frac{1-\frac{v}{c}}{1+\frac{v}{c}}} \tag{60}$$

And where $v \ll c$,

$$\frac{\lambda'}{\lambda} = 1 - \frac{v}{c} \tag{61}$$

The Doppler effect influences the availability of electrons that may absorb an incident photon. This is because of the quantization of possible states in which an electron may be. Having discrete states, with finite energy and momentum differences, enumerates the photons that an electron may absorb. However, due to the Doppler effect, the relative energy and momentum of an incident photon varies according to the energy and momentum distributions of the electron. Therefore, around each distinct accepted state for the electron, there is a distribution of possible acceptable photon energies (or momenta). Below is included an illustration of how the Doppler effect changes the states available to an electron.

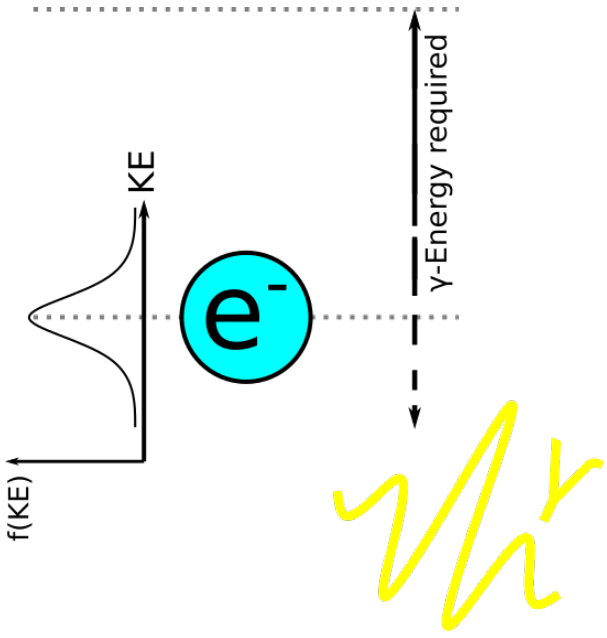


Figure 3: Depending on the distribution of electron energy, the total energy required to excite an electron to a higher energy state varies relative to the difference in frames.

Doppler broadening contributes to the differential cross-section by means of the Compton profile. To consider the Doppler broadening on each quantized state, the contribution for each shell are considered separately. This is described by Bursa *et al.*

as the "atomic Compton profile" [41].

$$J(p_z) = \sum_i Z_i J_i(p_z) \tag{62}$$

Where Z_i is the number of electrons in the i^{th} shell, and each term J_i

$$J_i = 2\pi \int_{|p_{min}|}^{\infty} p\rho_i(p_z) \cdot dp \tag{63}$$

Where p_{min} is the lowest momentum transfer that can occur during scattering. This is equal to [37]

$$p_{min} = \frac{|\omega(\omega - \omega') - \omega\omega'(1 - \cos\theta)|}{\left|\vec{k} - \vec{k'}\right|} \tag{64}$$

This is approximated by,

$$p_{min} \approx p_z \left(1 \pm \frac{(\omega - \omega')\, p_z}{2m \left|\vec{k} - \vec{k'}\right|}\right) \tag{65}$$

By the plus-minus term, the breadth of the energy spectrum is accounted for.

3.3.2.4.2 Binding Effects Bursa *et al.* describe the double differential cross-section derived by Ribberfors using a relativistic Impulse approximation [37], as "incorperat[ing] binding effects and Doppler broadening in a consistent way." However, Bursa *et al.* continue and incorporate an additional consideration for binding effects. The Compton profile is redefined as,

$$J(p_z) = \sum_i Z_i J_i(p_z) \Theta(\omega - \omega' - U_i) \tag{66}$$

Where Θ is a Heaviside step function, shifted so that the difference in electron energy, $E - E'$ must exceed the ionization energy U_i, for the factor to be non-zero. This effectively limits the allowed scattering events to those where the energy imparted is sufficient to excite the target electron to a free state. Therefore, the incoherent scattering function is defined,

$$S(\omega, \theta, Z) = \sum_i Z_i \Theta(\omega - \omega' - U_i) \int_{-\infty}^{p_{i,max}} J_i(p_z), F(\omega, \theta, p_z) \cdot dp_z \tag{67}$$

Note that the incoherent scattering function is integrated up to a maximum momentum. According to Ribberfors and Berggren, this "is highest p_z value for which an electron in orbital i can be excited". This value is defined as

$$p_{i,max}(\omega, \theta) = \frac{\omega(\omega - U_i)(1 - \cos\theta) - mc^2 U_i}{2\sqrt{2\omega(\omega - U_i)(1 - \cos\theta) + {U_i}^2}} \tag{68}$$

3.3.2.5 Further Approximations for Compton Scattering The EGSnrc manual approximates the Compton profile as using the "one-electron profiles" of Bursa *et al.* [41]

$$J_i(p_z) = J_{i,0}(1 + 2J_{i,0}|p_z|) \cdot exp\left[\frac{1}{2} - \frac{1}{2}(1 + 2J_{i,0}|p_z|)^2\right] \tag{69}$$

Where $J_{i,0} \equiv J_i(0)$, "obtained from the Hartree-Fock orbital." Unlike Ribberfors and Berggren, and Bursa *et al.*, EGSnrc approximates the catch-all function F as,

$$F(\omega, \theta, p_z) = \begin{cases} 1 - \alpha p, & p_z \leq -p \\ 1 - \alpha p_z, & |p_z| < -p \\ 1 + \alpha p, & p_z \geq p \end{cases} \tag{70}$$

Where,

$$\alpha = \frac{q_c}{\omega}\left(1 + \frac{\omega_c(\omega_c - \omega\cos\theta)}{q_c^2}\right) \tag{71}$$

and

$$q_c = \sqrt{\omega^2\omega'^2 - 2\omega\omega'\cos\theta} \tag{72}$$

ω_c defines the *Compton line*, where $p_z = 0$,

$$\omega_c = \frac{m\omega}{[\omega(1 - \cos\theta) + m]} \tag{73}$$

3.3.2.6 Simulating Compton Scattering To simulate stochastic Compton scattering events, EGSnrc uses a "fictions cross-section method." [4] This simplifies the required sampling. EGSnrc leverages the KN sampling implemented in EGS4, using Σ_{KN} as an enveloping curve that is easily sampled from. This was discussed in Section 3.1. Therefore, events can be tracked using the total Klein-Nishina cross-section. Compton interactions are therefore rejected with the probability $1 - \sigma_{Comp}^{(tot)}/\sigma_{KN}^{(tot)}$. The total incoherent scattering differential cross-section, relative to the total Klein-Nishina cross-section is,

$$\frac{d^2\sigma_{comp}}{\sigma_{KN}^{(tot)}} = \sum \frac{Z_i}{Z}\Theta(\omega - U_i)\sigma_i d\Omega dp_z \tag{74}$$

Where,

$$\sigma_i d\Omega dp_z = S_i \frac{d\phi}{2\pi} \frac{X_{KN}(\cos\theta)d\cos\theta}{\int_{-1}^{1} X_{KN}(\cos\theta)d\cos\theta} \frac{J_i(p_z)F(\omega, \cos\theta, p_z)\Theta(p_i - p_z)dp_z}{\int_{-\infty}^{p_i} J_i(p_z)F(\omega, \cos\theta, p_z)dp_z} \tag{75}$$

The remainder of the extensive sampling methodology that EGSnrc implements is beyond the scope of this review.

3.4 Other Phenomena Simulated in EGSnrc

3.4.1 Pair and Triplet production

Pair production is a spontaneous generation of an electron-positron pair from the energy of a photon incident upon a neighbourhood of an atom or particle [4][28].

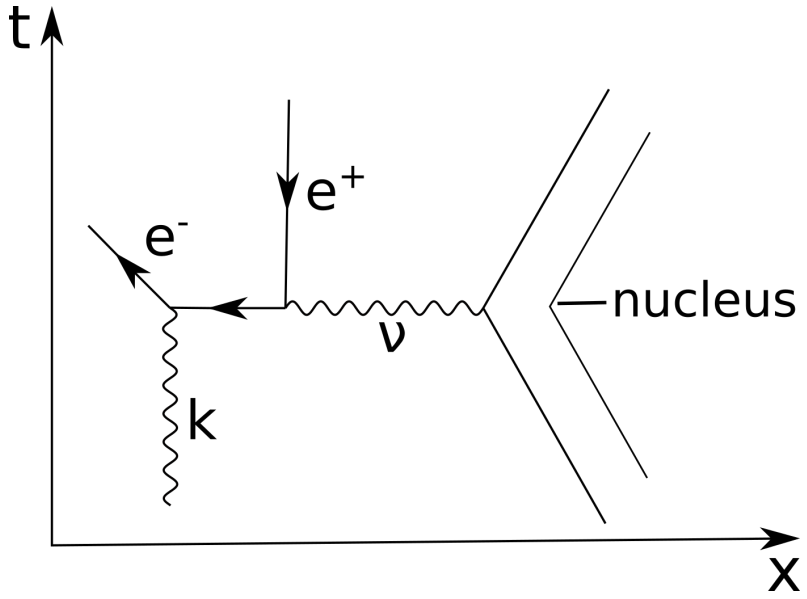


Figure 4: Diagram of pair production: An incident photon exchanges a virtual photon with a nucleus, and spontaneously spawns an electron positron pair. Equivalently, with the positron as an electron moving backwards in time, it is scattered (in time) by the nucleus, exiting the interaction as an electron moving forward in time [28].

The proximity to an atom or particle is required to conserve both energy and momentum. Any residual momentum or momentum deficit can be sunk into (or supplied by) the neighbouring atom or particle. A further phenomenon may follow the pair-production. In cases where the energy of the incident photon exceeds required for pair production, it is possible for a secondary photon to eject a bound electron from a neighbouring atom. The EGSnrc manual describes this [4].

> "the materialization into an e^+e^- pair takes place with the participation of an atomic electron which, after receiving sufficient energy, is set free."

This additional phenomenon is referred to as "triplet production". Triplet production is simulated indirectly. EGSnrc uses the "extreme relativistic first Born approximation" for

the differential cross-section of pair production. The differential cross-section is [4],

$$\begin{aligned}\frac{d\sigma_{pair}(Z,k,E_+)}{dE_+} =& \frac{A'_p(Z,k)r_0^2\alpha Z(Z+\xi(Z))}{k}\times \\ & \left[\left({E_+}^2+{E_-}^2\right)\left(\phi_1(\delta)-\frac{4}{3}\ln Z-4\bar{f}_c(k,Z)\right)\right. \\ & \left.+\frac{2}{3}E_+E_-\left(\phi_2(\delta)-\frac{4}{3}\ln Z-4\bar{f}_c(k,Z)\right)\right]\end{aligned} \tag{76}$$

Where,

$$\delta = 136Z^{-1/3}\cdot 2\frac{km}{2E_+E_-} \tag{77}$$

and $\bar{f}_c(Z)$ provides a Coulomb correction above 50 MeV,

$$\bar{f}_c(Z) = \begin{cases} f_c(Z), & k \geq 50\,\text{MeV} \\ 0, & \text{else} \end{cases} \tag{78}$$

$A'_p(k,Z)$ is an empirical correction factor, taken as unity above 50 MeV. Furthermore, in the derivation, the substitution of $Z^2 \approx Z(Z+\xi(Z))$ accounts for triplet production,

$$\xi(Z) = \frac{L'_{rad}(Z)}{L_{rad}(Z)-f_c(Z)} \tag{79}$$

Where f_c is the same as previously defined. L' and L are *radiation logarithms*. Lastly, $\phi_1(\delta)$ and $\phi_2(\delta)$ account for screening effects.

3.4.2 Photo-electric Absorption

Photo-electric absorption occurs with the absorption of a photon by a bound electron.

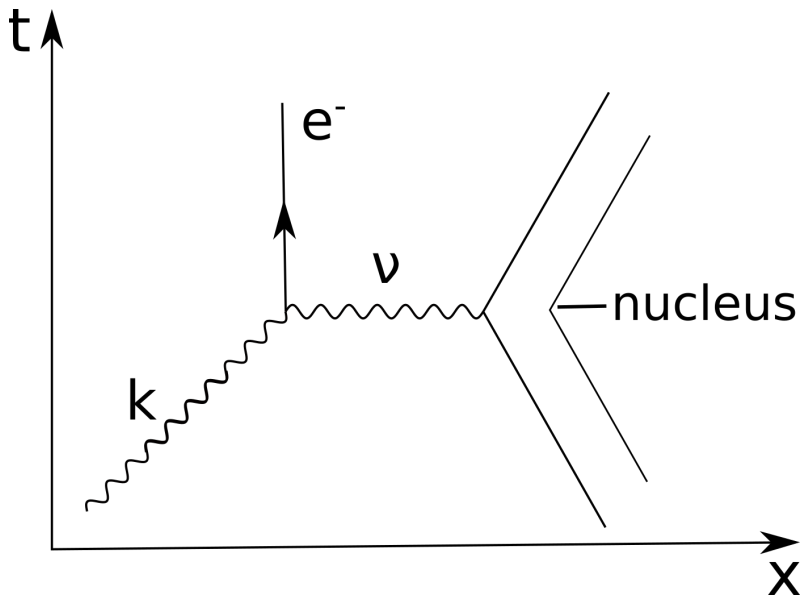


Figure 5: Diagram of photoelectric absorption: Here illustrated, the electron absorbs sufficient energy to be freed from its bound state.

Above the atomic binding energy, enough energy is transferred to the electron so as to eject it from its shell. This differs from the Compton effect where scattering, rather than absorption, causes the emission of an electron. Moreover, the incident photon disappears during the Photoelectric effect. Photo-electric absorption is more likely to occur for lower-energy photons, whereas the Compton scattering is more common at higher-energies [29]. As with Compton electrons, once ejected, photo-electrons leave a vacant state in its electron shell. If the vacancy is in a lower shell, it can be filled by an electron from a higher shell. Below the atomic binding energy, the electron that absorbs the photon is left bound and in an excited state. Atomic relaxation can occur emitting either a fluorescent photon or an Auger electron. This is discussed further in section 3.4.3.

3.4.3 Atomic Relaxations

Atomic relaxations are a set of outcomes resulting from the absorption of a photon by an electron (Photo-electric absorption). Fluorescence, Phosphorescence, and the Auger effect are example outcomes, though only fluorescence and the Auger effect are simulated in EGSnrc. **Fluorescence** [42] is an extended series of processes by which an excited electron returns to its ground state. This is illustrated in Figure 6 [43].

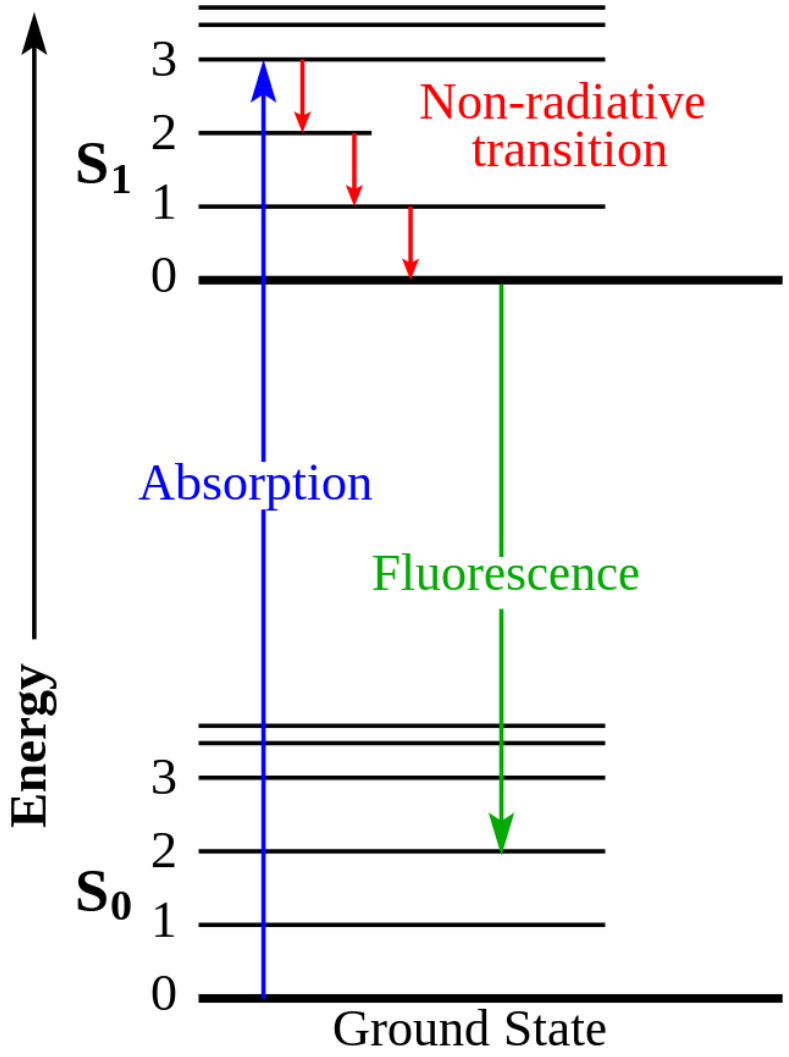


Figure 6: "Jablonski diagram of absorbance, non-radiative decay, and fluorescence." [43]: An electron absorbs a photon and is left in an excited state. Through radiative and non-radiative processes the electron releases energy [42].

From its excited state, the electron undergoes a series of non-radiative transitions: vibrational relaxation and internal conversion. Through these, some energy is lost to heat. The electron settles in a semi-stable excited state. After some time, probabilistically described by exponential decay, the electron returns to its ground state by emitting a photon equal to the difference between the ground state and the semi-stable state. Therefore, by the process of fluorescence a photon is down-scattered and delayed in its motion. Phosphorescence [42] is similar to Fluorescence, but includes an additional non-radiative process called "intersystem-crossing". This is a *spin-forbidden transition* that changes the spin of the electron through spin-orbit coupling. This additional pro-

cess further delays the re-emission of a photon and causes further losses of energy to heat.

Lastly, **the Auger effect** [4][29] can occur as a consequence of photo-electric absorption. However, the Auger effect results in the emission of an electron rather than a photon.

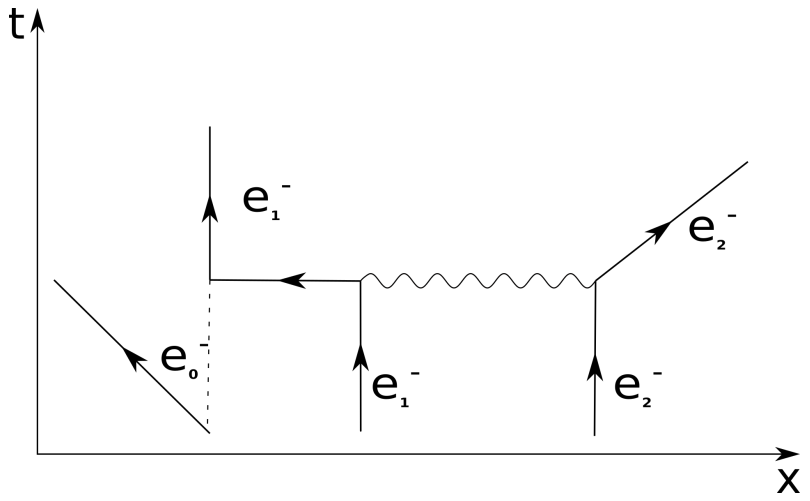


Figure 7: Diagram of the Auger effect: To fill a vacancy an inner electron transfers energy to an outer electron. The energy transferred is sufficient to free the outer electron from its bound state.

In the case where an inner-shell electron is excited, the remaining vacancy can is likely to be filled by an electron from a higher shell. For the "downwards" transition to occur, the electron from the higher shell emits a photon. This photon may either escape or cause the emission of an electron in further higher shell. The new vacancy, caused by the "downwards" transition of the higher-shell electron is also susceptible to this process.

3.5 Electron Transport

The movement of electrons through a medium can be described by the transport equation,

$$\frac{d\Phi(\vec{x},\vec{\Omega},E,t)}{dt} = S(\vec{x},\vec{\Omega},E,t) + vI[\Phi] \tag{80}$$

Above, the change of electron fluence in time is equated to the sum of two terms. A source term, S, dependent on position $\vec{x}$, velocity, $\vec{v} = (v,\vec{\Omega})$, and energy, E, at a given

time t. The second term, $vI[\Phi]$, is a collision term. Where,

$$\begin{aligned} I[\Phi] = &- n(\vec{x})\Phi(\vec{x},\vec{\Omega},E,t)\int_0^E\int_{4\pi}\sigma(E,E',\Omega,\vec{x})dE'd\Omega' \\ &+ n(\vec{x})\int_E^\infty\int_{4\pi}\Phi(\vec{x},\vec{\Omega},E',t)\sigma(E',E-E,\vec{\Omega'}\cdot\vec{\Omega},\vec{x})dE'd\Omega' \end{aligned} \tag{81}$$

Which depends on the cross-section σ. The solution to the transport equation, coupled with other photon interactions previously described is, according to the EGS4 manual, "prohibitively difficult to solve except under severe approximation." Therefore, the transport equation, and other coupled phenomena is computationally handled by using a Monte Carlo simulation.

However, even using a Monte Carlo simulation to determine the motion of electrons becomes computationally difficult. This is because of the exceedingly large number of minor interactions that occur of an average electron's lifetime. The EGSnrc manual describes this challenge [4],

> "In the process of slowing down, a typical fast electron and the secondary particles creates undergo hundreds of thousands of interactions with surrounding matter. Because of this large number of collisions, an event-by-event simulation of electron transport is not possible due to limitations in computing power. "

The "Condensed History" technique is employed to mitigate the computational challenges of simulating the full transport of individual electrons.

3.5.1 Condensed History Technique

The "Condensed History" (CH) technique groups many electrons and samples the outcome of their combined interactions. This is done in individual "steps".

> "The cumulative effect of the individual interactions is taken into account by sampling the charge of the particle's energy, direction of motion, and position, at the end of the step from appropriate multiple scattering distributions. "

For a single particle, the fluence can be written as

$$\Phi(\vec{x},\vec{\Omega},E,t) = \int_0^t\int_E^\infty\int\int_{4\pi}S(\vec{x_0},\vec{\Omega_0},E_0,t_0)\Phi_0(\vec{x},\vec{\Omega},E,t)\cdot dt_0dE_0dx_0d\Omega_0 \tag{82}$$

This equation is repeatedly solved in the Monte Carlo simulation.

The total microscopic cross-section for electrons is,

$$\sigma(E, E', \Omega, \vec{x}) = \sigma_{brem}(E, E', \Omega, \vec{x}) + \sigma_{inel}(E, E', \Omega, \vec{x}) + \sigma_{el}(E, \Omega, \vec{x})\delta(0) \qquad (83)$$

Positrons have an additional term for annihilation. The collision integral is split into two, according to energy loss, where one integral corresponds to inelastic collisions and Bremsstrahlung radiation, and the other to elastic collisions. The former, are referred to as "catastrophic" collisions and are simulated directly. The latter, elastic collisions, radiative events and sub-threshold inelastic events are indirectly simulated by grouping. Effectively, this requires deriving the appropriate spread of particles between "catastrophic" events. Then, prior to simulating the catastrophic event, the position and direction of all grouped particles are resampled within that spread. This resampling is done several times prior to reaching the "catastrophic" event.

3.5.2 Bremsstrahlung Radiation

Bremsstrahlung radiation occurs when an electron (or other charged particle), is deflected by an atomic core (or other charged particle). This changes the direction of the incident electron, accelerating the charge away, and therefore releasing a photon.

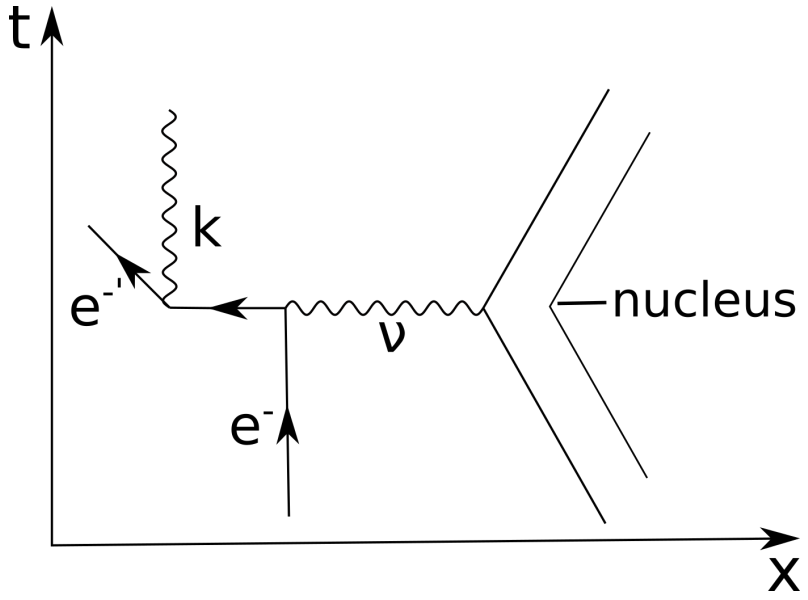


Figure 8: Diagram of Bremsstrahlung radiation: Scattering off of an atom accelerates the charged electron, emitting a photon.

There is a "cross-symmetry" [28] between Bremsstrahlung radiation and pair production, and the implementation of their differential cross-sections is very similar. The only difference is a factor for the positron's energy that is factored away; the remaining terms are replaced with those of initial electron energy [4].

3.5.3 Other Electron Interactions

Other discrete interaction involving transported electrons are Møller scattering, Bhabha scattering and electron-positron annihilation. **Møller scattering** is the inelastic scattering of one electron off of another electron.

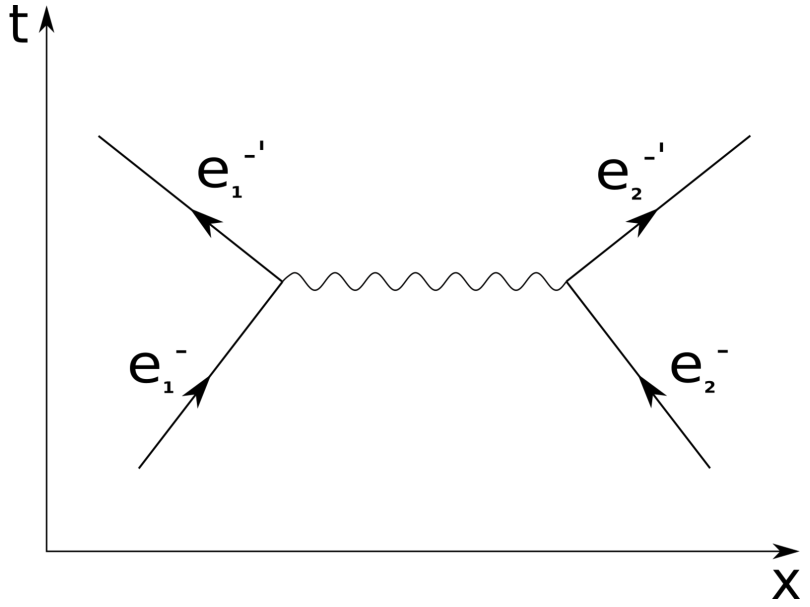


Figure 9: Diagram of Møller scattering: Two electrons are elastically scattered off of each other, exchanging a photon in the process.

The differential cross-section with respect to scattered kinetic energy, T', is

$$\frac{d\sigma^{-}_{inel}}{T'} = \frac{2\pi {r_0}^2 m}{\beta^2}\frac{1}{T'^2}\left[1 + \frac{T'^2}{(T-T')^2} + \frac{\tau^2}{(\tau+1)^2}\left(\frac{T'}{T}\right)^2 - \frac{2\tau+1}{(\tau+1)^2}\frac{T'}{T-T'}\right] \tag{84}$$

Where, β is the ratio between incident electron velocity and the speed of light, $T|T'$ are the kinetic energies of the electron before and after scattering, and $\tau = T/m$. This cross-section applies to both electrons.

Bhabha scattering is similar to Møller scattering, though with one of the electrons being a positron.

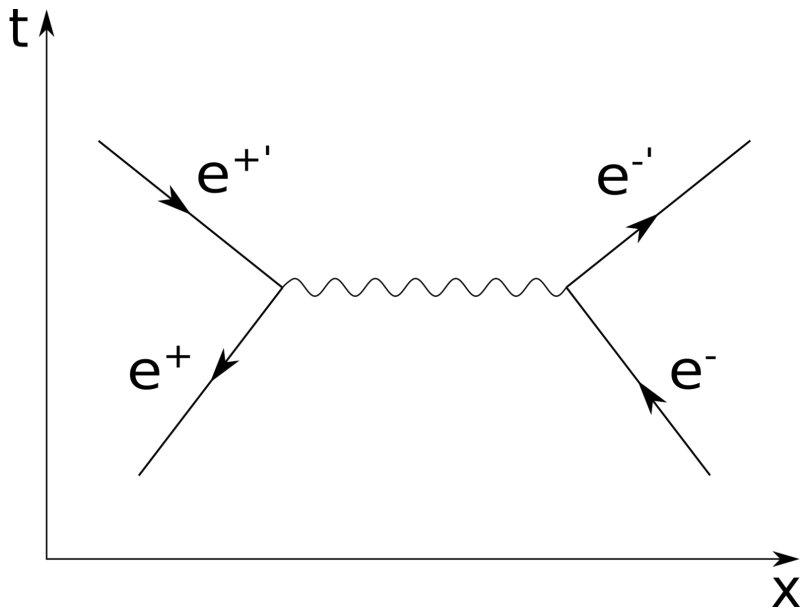


Figure 10: Diagram of Bhabha scattering: An electron and positron are elastically scattered off of each other, exchanging a photon in the process.

The differential cross-section relative to scattered kinetic energy, T', is

$$\frac{d\sigma^{+}_{inel}}{dT'} = \frac{2\pi r_0^2 m}{T^2}\left[\frac{T}{T'}\left(\frac{T}{T'\beta^2} - 2 + y^2\right) + (1-2y)(3+y^2) + \frac{T'}{T}\left(\frac{T'}{T}(1-2y)^3 - (1-2y)^3 - (1-2y)^2\right)\right] \tag{85}$$

Where $\beta = \frac{\tau(\tau+1)}{(\tau+1)^2}$, $y = 1/(\tau+2)$ and again $\tau = T/m$. The sampling implementation for Bhabha scattering is similar to that of Møller scattering.

Lastly, **annihilation** occurs where an electron and positron interact and both the electron and positron are consumed during the interaction. One, two or more photons are emitted. The most likely outcome is the release of two photons. Less likely outcomes are ignored in EGSnrc. Annihilation is a "catastrophic" event and so is simulated directly. The differential cross-section relative to the energy of one of the annihilation photons is,

$$\frac{d\sigma_{annih}}{dk} = \frac{\pi {r_0}^2}{\tau(\tau+2)}\left[S_1(\kappa) + S_1(\tau+2-\kappa)\right] \tag{86}$$

Where τ is positron kinetic energy and κ is the photon energy, both in units of m. The term S_1 is a simplification,

$$S_1(x) = \frac{1}{x}\left(\tau + 2 + 2\frac{\tau+1}{\tau+2} - \frac{1}{x}\right) - 1 \tag{87}$$

4 Machine Learning Methods

4.1 Machine Learning

The term "Machine Learning" (ML) is the field of study of mathematical models that can be optimized to predict, calculate, categorize or otherwise manipulate data. Here the term "model" may refer to anything that can substitute or simulate human intelligence. The degree of success or failure of an optimization is measured by *error*, calculated by an *error function*. The term "error" is often substituted for terms such as *loss* or *risk*. The method of optimization may either be deterministic or stochastic. For example, when doing linear regression, the parameters are calculated by equations that solve a loss functional. For problems where analytical solutions for loss minimization are untenable or intractable, numerical and sometimes iterative algorithms are employed to approximate the optimal solution.

4.2 Optimization

Methods of machine learning optimization are often categorized as either supervised, or unsupervised [44]. ML models may either be optimized (or trained) by *supervised learning* methods or *unsupervised learning* methods.

4.2.1 Supervised Learning

Supervised learning requires *training data*, which are example mappings of input data to output data (known as *labels*). For some given input data, the model will produce an estimate of the corresponding label(s). Using some method of optimization, the relationship between input training data and labels will be estimated or generalized. This optimization (known as *training*) seeks to minimize a *loss function*, which calculates some measure of discrepancy or difference between the model's prediction and the correct value of the corresponding label. Common examples of loss functions include *Mean Squared Error* for regression and *Cross-Entropy* for categorization. Once the model is trained it can be tested for efficacy. To do so the loss is calculated on a set of *test data*. For more information on test data, training data, and their separation refer to 4.3.2.1.

4.2.2 Unsupervised Learning

Unsupervised learning, similar to supervised learning, optimizes a model by minimizing a loss function. However, in the case of unsupervised learning, the loss function describes some property of the transformed data. The optimized model transforms input data to exaggerate, separate or minimize particular aspects of the data.

Unsupervised learning may be used on data where supervised learning may be untenable. For example, if input data is of too high a dimensionality, complexity, or quantity to be meaningfully labelled. This major advantage is offset by the subversive consequence that the results of model trained by unsupervised learning cannot be simply interpreted. Hastie, Tibshirani, and Friedman in The Elements of Statistical Learning end the introduction to their chapter on Unsupervised learning with the following description of this [44].

> "In the context of unsupervised learning, there is no such direct measure of success. It is difficult to ascertain the validity of inferences drawn from the output of most unsupervised learning algorithms. One must resort to heuristic arguments ... This uncomfortable situation has led to heavy proliferation of proposed methods, since effectiveness is a matter of opinion and cannot be verified directly"

Furthermore, there is not the same distinction between train and test data in unsupervised learning. Test data may be used to verify the stability of the algorithm (i.e. that the algorithm gives similar results on different sets of training data).

4.2.3 Comparing Unsupervised and Supervised Learning

Though these domains of supervised and unsupervised learning appear very different, they in fact have much in common. Since they both rely ultimately on a loss function they may both be use for training many of the same types of models. Similarly, they may also be employed for solving the same problem. Take for example the task of sorting a shuffled deck of pictures, where each picture illustrates either a cat or a dog. Let the chosen model be a simply connected neural network that outputs a 2-vector. Ideally this 2-vector would provide the probability of an input image being either a cat or a dog.

One could use supervised learning to train the model with a set of labelled training pictures. It is also possible to use an unsupervised learning optimization, where the loss is proportional to lower of the two probabilities (thereby promoting a decisive prediction). Both approaches have their pros and cons. Supervised learning requires one to label a substantial (perhaps even a majority) of the data prior to the training. However, provided an appropriate set of training data, supervised learning more reliably provides learning of the difference between cats and dogs. Unsupervised learning on the other hand does not require one to label data, this is done at the cost of no assurance of what the model will optimize for. For example, it is possible that an unsupervised training regime will categorize images according to the colour of the animal, the brightness of the image, or some other characteristic of the images that may not even be perceptible to a human eye.

4.2.4 Probabilistic Loss

To train models to imitate probabilistic targets, one must modify or use alternative training regimes. If a model is scored against a probabilistic label, it may appear to perform poorly due to a label that deviates from expected behaviour. Within some limited amount of variance the effect of a probabilistic label may only reduce the apparent generalization of the model. It is easy to imagine a decision tree regressor that would simply regress to the mean of the different labels in a leaf. This behaviour is normal and intended for a decision tree regressor. However, if the variation of a label has a similar scale to the variation between labels, then the probabilistic label could interfere with training.

Furthermore, when using a model as a surrogate during optimization, as was described in Chapter 2: "Introduction to Inverse Design", it may be favourable to include a degree of variance in the model's prediction. As a simple example, imagine that one is trying to design a building to resist high winds. To do so a surrogate model is trained to simulate the stress applied to a building shape. Since wind can blow from any different direction, a model that simulates the most likely stress would only simulate wind coming from a single (most likely) direction.

4.2.4.1 Goodness-of-fit Tests To estimate the "goodness-of-fit", a χ^2 test may be employed for dimensional predictive models. This requires that: [30] each point be in-

dependent and normally distributed. Furthermore, the model must be linear in its parameters and the functional form of the model must be "correct". The later two of these requirements may be relaxed (within reason). By relaxing these, the goodness-of-test fit precisely answers the question "What is the goodness-of-fit for a model that is both linear in its parameters and has a correct functional form, that *also performs identically* to the model in question?" So, to apply the goodness-of-fit test using a single test case, a distribution of each prediction (point) is produced by successive simulation with a varying random seed. The mean value of each distribution is distributed normally, according to central-limit theorem. The χ^2 statistic, will be distributed according to a χ^2 distribution with degrees of freedom equal to the number of points predicted divided by the number of dimensions used as input to for a single prediction.

$$\chi^2 = \sum_{i=1}^{N} \frac{(y_i - \lambda(x_i))^2}{\sigma_i^2} \tag{88}$$

Where y_i is the label value, $\lambda(x_i)$ is the prediction of the label based off of input data x_i, and ${\sigma_i}^2$ is the variance of the label. This is summed over N data points. Using the χ^2 distribution, f_{χ^2}, with n_d degrees of freedom, a P-value can be calculated.

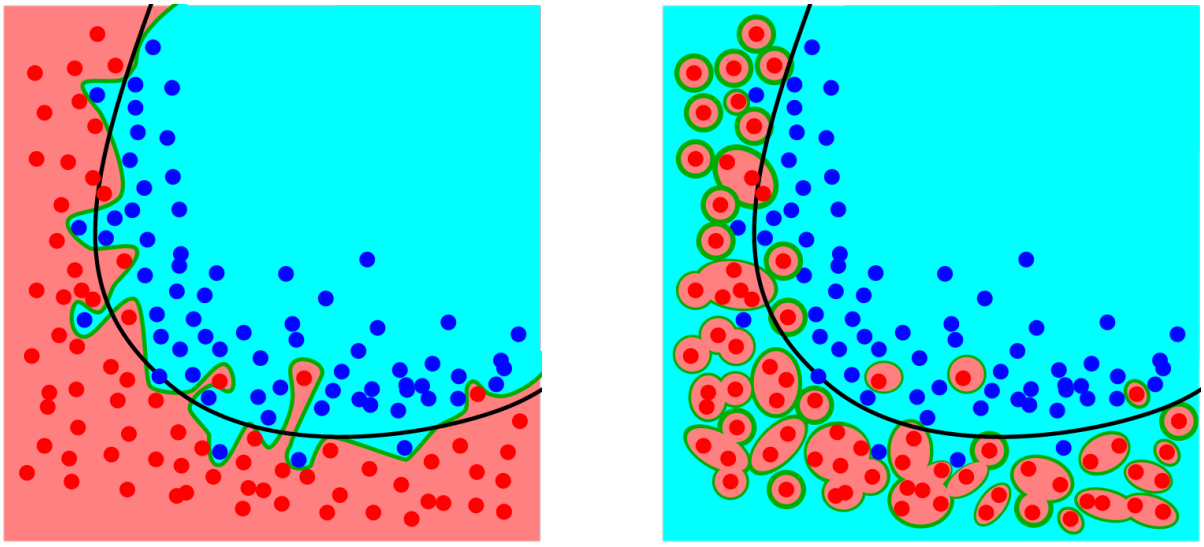
$$P = \int_{\chi_0^2}^{\infty} f_{\chi^2}(\hat{y}; n_d) d\hat{y} \tag{89}$$

This value, P, represents the probability that a simulation would yield a result less *characteristic* than the prediction. With this one can calculate the probability that a prediction from the model would be found from a simulation. P-values are commonly used to reject hypotheses, and here can be used to judge whether a prediction from a model is an appropriate estimation of the simulation. The threshold for a P-value worth rejecting depends on the application.

4.2.5 Overfitting

For every task of classification or regression, there is some minimum level of model complexity required for "meaningful" generalization. For now, this can remain loosely defined. For models that meet or exceed this minimum level of complexity, there additional predictive power possesses a risk for *overfitting* [44]. In such a case, if a model is trained beyond a "meaningful" generalization, its predictions will increasingly represent the particularities of the training set. This tends to decrease overall generalization. "Overfitting" describes a spectrum of inaccuracies of this sort. On the near end, an overfit

model will accurately predict invalid or outlier points (points that do not characterize the distribution). At the far end of the overfitting spectrum, are models that have effectively "memorized" the training set.



a) b)

Figure 11: Illustrations of overfitting, adapted from [45]. A target distinction delineates an optimal division, in black, between two categories of data. The predictions for these two categories are marked in blue and pink. A green line delineates a "decision boundary", which shows the model's predicted division between the two categories [44]. In subfigure a) the decision boundary of an overfit model is illustrated. Some points beyond the optimal division are misclassified Despite this, the model's decision boundary is continuous, and it demonstrates some generalization. Subfigure b) illustrates a highly overfit model, which has "memorized" the pink points. This model has very poor generalization.

Overfitting is illustrated in Figure 11. On the left, Figure 11 a) illustrates some minor overfitting. There, the model's green classification boundary has deviated from a black target boundary. Noise from the training set is now reflected in the model's prediction. On the right, in Figure 11 b), is the logical conclusion of overfitting- memorization. The model's green classification boundary is entirely disjoint, and does not resemble the target boundary at all. The degree to which a model is overfit can be measured, using the "*relative overfitting rate*" [44],

$$\hat{R} = \frac{\hat{Err}^{(1)} - \hat{err}}{\hat{\gamma} - \hat{err}} \tag{90}$$

Where $\hat{err}$ is the training error rate, $\hat{Err}^{(1)}$ is the "leave-one-out bootstrap estimate of prediction error", and γ is the no-information error rate. The no-information circumstance provides interesting point for comparison. It is the circumstance were inputs and associated outputs are independent. Training in the case of no-information yields a model that still maps from the input space to the output space, but in an incoherent way.

In the limiting case, an overfit model will provide have "memorized" the mapping from input data to labels. In the "no-information" scenario, an overfit model will yield a random label y_i for some input x_j. Therefore, the no-information-loss-rate estimation becomes

$$\hat{\gamma} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} L(y_i, y_j) \tag{91}$$

Which is twice the variance. This provides an interesting baseline for the worst that one can expect of a model's performance, while still mapping between input and outputs spaces.

4.2.5.1 Mitigating Overfitting Methods for mitigating overfitting can either be general or model-specific. First, and foremost, the best way to reduce inaccuracy due to overfitting is to limit the complexity of one's model. Doing so will limit the extent the model is even capable of overfitting. For tree-based models, limiting model complexity and stopping training early are equivalent. For other model types, such as neural networks, one may stop training "early". To determine when this is, one can use a validation set or do k-fold cross-validation. A final means to limit overfitting is to have training data that is characteristic of the target distribution. In theory, if a model is overfit to such data would not suffer inaccuracies. This is not withstanding both sufficient availability of data and the possibility of training methods interfering.

4.3 Error

As has been mentioned, supervised learning requires distinct and non-overlapping sets of train and test data. Test data must have no items in common with the training data, but must be sampled from the same distribution as the training data. The loss calculated from the test data is called the *empirical error*. It estimates the expected value of loss, the *generalized error*, based on the average loss of the test set. The performance of a model on an independent set of test data is referred to as the *generalization*.

4.3.1 Generalization Error

Generalized error has three fundamental components, irreducible error, bias and variance. As will be seen, there have been many attempts to formulate definitions and an associated general decomposition that is applicable for all loss functions. The first component, irreducible error, can be more consistently described. Given the input data X from the space $\mathcal{X}$ and labels Y from the label space $\mathcal{Y}$, let

$$Y = f(X) + \epsilon \tag{92}$$

Here ϵ, is the component of Y that cannot be described by a function of X. This typically is noise with the properties that $E[\epsilon] = 0$ and $E[\epsilon^2] = \sigma^2$.

The remaining two components bias and variance constitute, casually speaking, a nature vs. nurture dichotomy. The Bousquet and Elisseeff (2002) paper Stability and Generalization broadly defines variance and bias [46].

> "In broad terms, the bias is the best error that can be achieved and the variance is the difference between the typical error and the best error."

The bias in the sense of Bousquet and Elisseeff is a lower-limit on generalization determined by the type of model, and the characteristics and quantity of data. This is analogous to the "nature" of the problem. Similarly, the variance is determined by the complexity of the model, the characteristics of the data, and the training regime. It is a consequence of how one "nurtures" or deals with the problem.

When selecting a model type, there are often trade-offs between these two components of error. As a rule of thumb, more complex models have lower bias but greater variance. This is illustrated in Figure 12, which is adapted from [44].

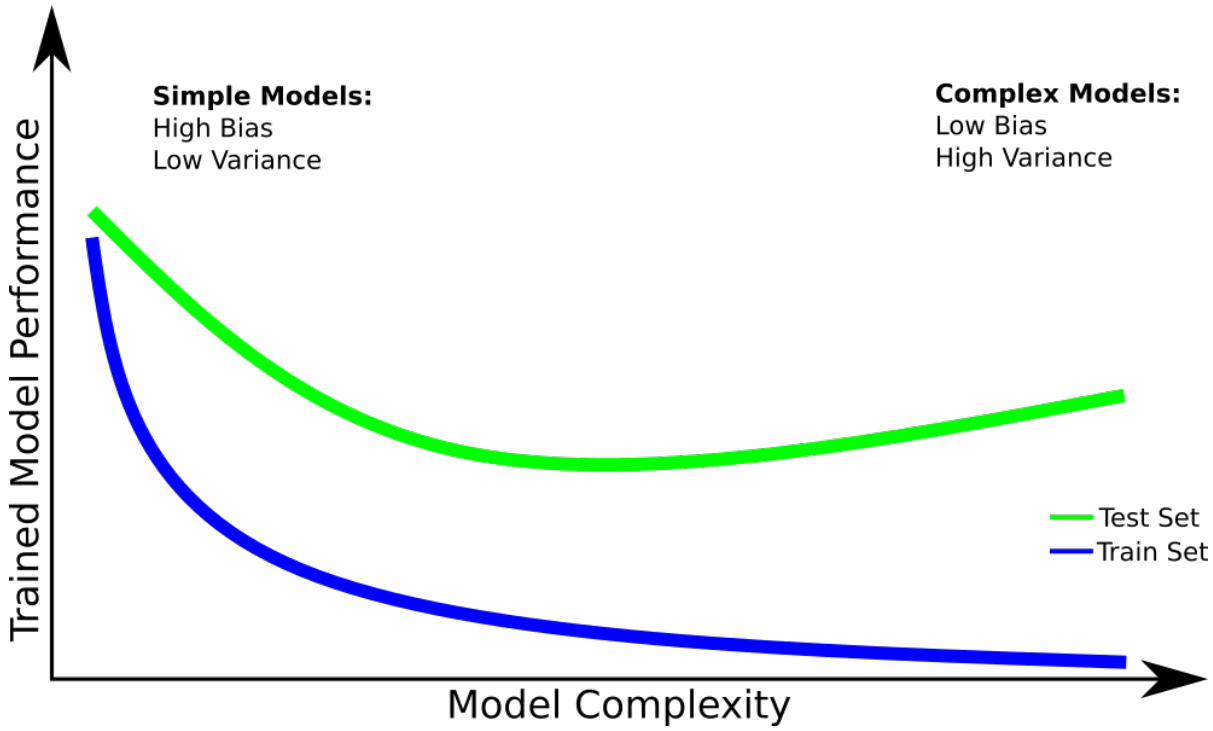


Figure 12: Illustration of generalization of trained machine learning models as a function of model complexity. Simple models, occupying the left of the figure, tend to have low variance and high bias. This means, respectively, that low complexity models are reliably optimized, but limited in their performance. Overly complex models can also have low performance, though often for the opposite reasons. Complex models tend to be sufficiently complex to minimize bias, but are difficult to reliably train.

4.3.1.1 Use a bigger hammer With the advent of "big data", our most complex methods have been empowered by massive training sets. With a training set more characteristic of an underlying distribution of error, optimization of complex models will reduce the expected variance.

4.3.2 Approximating Generalization Error

4.3.2.1 Cross Validation Though generalized error can sometimes be estimated without the test/train split, as is the case with the "out-of-bag error" of random forests [47], it is a widely accepted practice to have mutually exclusive data sets for training/fitting and for testing [44][48]. When building models, if the resulting training time is not prohibitive, it is useful to experiment with different hyperparameters, training, and other aspects of the model or training regime. Through this process, one hopes to narrow in

on the most effect means of generalization given some data. However, one risks optimizing the training process for a specific testing-subset. Therefore, when tweaking models, it is appropriate to have a third mutually exclusive set of data called *validation*. The validation set is useful for tweaking the model, as it provides a measure of generalization independent of the training set. This also leaves the "true" testing dataset to be a data and training regime independent measure of generalization.

Best practices of generalization estimation extend this practice to *K-fold cross-validation*. This involves breaking the total data up into K partitions, each of which will be used as a test set for an independently trained model on the other $K-1$ pieces. Figure 13 illustrates a 4-fold cross validation.

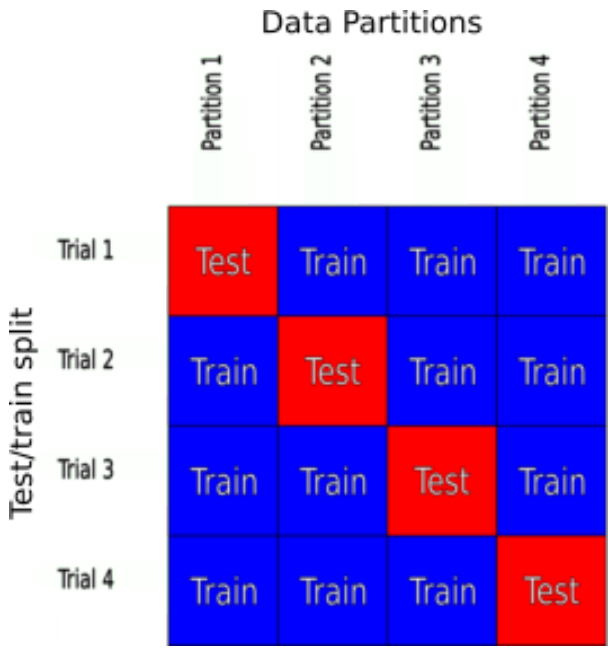


Figure 13: 4-fold cross validation regime. In each trial a model is trained on a different three-quarters of the available data. The error of the model is measured against remaining quarter. The average of multiple "folds" provide a more reliable estimate of generalization.

For the i^{th} trial, a fresh model is trained on all but the i^{th} partition of data. The i^{th} partition is reserved as the test split. K-fold cross-validation estimates a distribution of generalization for a particular training regime, a given model type, optimization method, and data. This is particularly important when evaluating the suitability or generalizability of a highly stochastic regimes, or if the available data is limited. It also allows one to avoid having to allocate data for a verification set.

Given a finite amount of data available for cross-validation, there is a bias introduced when approximating generalization error by using mutually exclusive training and test sets. This is merely because truly independent and identically distributed samples would allow for the same datum to appear in both sets. It is to the degree that the test set is characteristic of the distribution that its corresponding error approximates generalization error. Therefore, the introduced bias is (very) small with sufficient data. Furthermore, there are expressions derived using test error to minimize any effects of this bias. Several of such are discussed in the coming subsection.

4.3.2.2 Bootstrapping Cross Validation is a powerful tool for estimating generalization error, optimizing hyperparameters, and designing data (pre-)processes. In its purest form however, cross validation introduces a misrepresentation of the distribution of data. Since the data must be partitioned in to distinct segments, it requires sampling without replacement. The mode of the distribution, regardless of how likely, is only ever present in a single partition. To relax the requirement on sampling with replacement, is to use *bootstrapping*. This term refers to many methods and processes. The application of bootstrapping to ensembles, or combination of models, is discussed at length in 4.5.1. Neither bootstrapping nor cross-validation are "better" than the other. Rather, there are appropriate uses for both. When estimating the generalization error, bootstrapping provides a more accurate approximation. However, cleanly partitioned cross validation allows one to more methodically investigate generalization. For example, one could search through cross-validations to find the worst possible performance for a model trained on some fraction of the data. Doing so could provide important bounds on performance.

A common use of bootstrapping is to approximate generalization error [44].

$$\hat{Err}_{boot} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{n=1}^{N} L(y_i, \hat{f}^{*b}(x_i)) \tag{93}$$

Where B is the number of bootstraps, N is the amount of data, L is the loss function, y_i is the ith label, $\hat{f}^{*b}(x_i)$ is the prediction of the label, by the bootstrapped model $\hat{f}^{*b}$ based on the ith input data x_i. Estimating generalization without a test-train split, risks underestimating the error due to common elements between test and train dataset. As is discussed in 4.2.5, this underestimating is due to *overfitting*. Applying a test-train split

avoids this underestimation,

$$\hat{Err}^{(1)} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{|C|}\sum_{b\in C} L(y_i, \hat{f}^{*b}(x_i)) \tag{94}$$

Where C is the number of elements excluded from a bootstrap. This returns the error-estimation to cross-validation, since the test-error must be a compliment to the characteristic bootstrap. Therefore, this can overestimate generalization error.

Analytic expressions have been derived for estimators that mitigate the overestimation of cross-validation error. For example ".632" estimator is a weighted sum of the cross-validation error and the training rate [44].

$$\hat{Err}^{(.632)} = 0.368 \cdot \bar{err} + 0.632 \cdot \hat{Err}^{(1)} \tag{95}$$

Where $\bar{err}$ is the training error. Training error is in a sense the limiting case subject to the problems found with a non-segregated bootstrap sample. All items in the training set a prone to overfitting. Error calculated on a sample without a test-train split is inaccurate due to overfitting, rather than bias in the sample. In a circumstance where no overfitting occurs, the error calculated on a non-segregated sample would provide a better estimation than that of a segregated sample. Leveraging a measure of overfitting, can improve the ".632" estimator. By adjusting the weighting, according to the *relative overfitting rate*,

$$\hat{R} = \frac{\hat{Err}^{(1)} - \hat{err}}{\hat{\gamma} - \hat{err}} \tag{96}$$

Where $\hat{err}$ is the training error rate, $\hat{Err}^{(1)}$ is the "leave-one-out bootstrap estimate of prediction error", and γ is the "*no-information-error-rate*",

$$\hat{\gamma} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} L(y_i, \hat{f}(x_j)) \tag{97}$$

The improvements upon the ".632" estimator, the ".632+" estimator is,

$$\hat{Err}^{(.632+)} = (1 - \hat{w}) \cdot \bar{err} + \hat{w} \cdot \hat{Err}^{(1)} \tag{98}$$

Where the weighting, $\hat{w}$, depends on the relative overfitting rate,

$$\hat{w} = \frac{.632}{1 - .368\hat{R}} \tag{99}$$

4.3.3 Error Decomposition

4.3.3.1 Bias-Variance Decomposition The advantage of a general definition for bias and variance is the ability to then derive equations that imply or demonstrate the ideal characteristics for model selection and training regime. In Figure 12, there is some optimal "complexity" of a model for any given task. To locate this minimum, the overall contributions of bias and variance relative to model complexity, and for a given loss function, must be determined. There are also major implications (and useful insights) when leveraging ensemble methods like bagging and boosting, that will be covered later in section 4.5.1. Domingos in A Unified Bias-Variance Decomposition states plainly [49],

> "the bias-variance decomposition of error has become a cornerstone of our understanding of inductive learning"

Domingos, using his "unified bias-variance decomposition",

$$E_{\mathcal{D},t}[L(t,y)] = c_1 E_t[L(t,y_*)] + L(y_*,y_m) + c_2 E_{\mathcal{D}}[L(y_m,y)] \tag{100}$$

$$= c_1 N(x) + B(x) + c_2 V(x) \tag{101}$$

labels the three resulting terms "noise" (equivalent to irreducible error), "bias" and "variance". In Domingos' BVD, t represents the true output value, y_* represents the ideal prediction that is possible by any model, y_m is the "central tendency" of a particular method trained on the possible space of training data, and y is the prediction of a single given model. It is then demonstrated that, as a special case of this, is a widely used decomposition from Geman *et al.* [50] for a mean squared error loss function.

The unified bias-variance decomposition is applied to a zero-one loss function for categorization, wherein the loss is zero if the estimation is correct and one if not. The generalization error equals,

$$E_t[L(t,y)] = L(y_*,y) \pm E_t[L(t,y*)] \tag{102}$$

Where the second term (variance) is positive for non-biased scenarios (I.e. $y_m = y_*$) and negative otherwise. Using the BVD, Domingos proves that

> "The fact that variance is additive in unbiased examples but subtractive in biased ones has significant consequences. If a learner is biased on an example, increasing variance decreases loss. This is markedly different from that of squared loss."

It is also proved that this applies in a decreasing way with increasing loss function asymmetry or with increasing dimensionality.

4.4 Decision Trees

A decision tree is a supervised learning method comprised of a set of if-then-else statements, which categorize or map some input data onto one of many final outcomes.

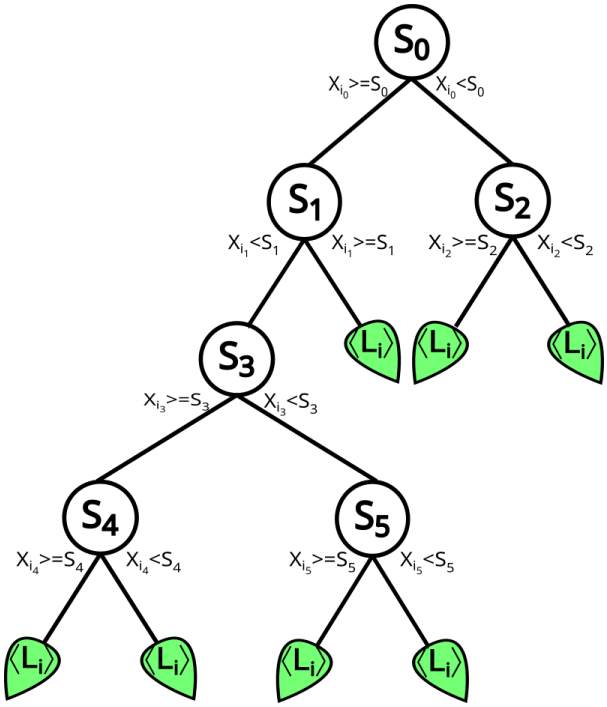


Figure 14: Illustration of the logical structure of a decision tree. Label data is iteratively split into increasing pure subsets. Each circle S_i denotes a split. The two possible outcomes are labelled on either side and extend to either further splits or a terminal node of sufficient purity (a "leaf").

Decision trees are a highly flexible type of model. According to Hastie, Tibshirani and Friedman [44], decision and regression trees have many advantages, including: handling of mixed data types, adaptive to missing values, robust to outliers, "[i]nsensitive to monotone transformations of inputs", computational scalability, are able "to deal with irrelevant inputs", and are reasonably interpretable. However, they also state that trees have poor predictive power and have difficulty extracting linear combinations of features. Decision trees are also well known to be unstable [44][49][51]. The decision tree structure does allow for the handling of non-regular and mixed data, i.e. mixed-data of

different types (categorical, floats, strings) etc. However, in practice, the incorporation of diverse data types may not so neatly be included into the learning algorithms (such as CART [26]). Furthermore, Decision trees are "whitebox models", and so plainly outline the criteria by which a particular decision was made. However, these criteria represent correlations and should not be interpreted as a causal or determining factors. Moreover, criteria may be degenerate, if the same criterion has discriminatory power among different pre-split subsets.

4.4.1 Guiding Principle

Decision trees categorize input data by successively splitting a data-set into increasingly pure sub-sets. This is introduced by Breiman *et al.* in Classification and Regression Trees [52],

> "Tree structured classifiers, or, more correctly, binary tree structured classifiers, are constructed by repeated splits of sub-sets of [the dataset] X into two descendant subsets, beginning with X itself."

Each split is a conditional statement regarding one variable of each input data, the "splitting variable", which is here denoted by j.

$$R_1(j) \equiv \{X|S_j\} \tag{103}$$

$$R_2(j) \equiv \{X|\neg S_j\} \tag{104}$$

Where S_j is the split condition on j. There are therefore two sub-sets or regions, R_1 and R_2, where the condition is true or false respectively. The condition S_j and the splitting variable j are selected to minimize the sum of loss in both resulting regions. The minimization takes the form,

$$\min_{j,S_j} \left[L(y_i|x_i \in R_1(j,S_j)) + L(y_i|x_i \in R_2(j,S_j))\right] \tag{105}$$

The above expression is slightly misleading. It seems to imply the splitting variable is selected on criteria other than S_j. The reverse is true. Of $\{S\}$, the totality of conditions on *any* single variable. It is the conditional S_j that is selected, the variable j just tags along.

After two regions are defined, The process is repeated recursively on each sub-sample until a stop-criterion is reached.

4.4.2 CART

CART, a contraction of "Classification and Regression Trees", is a computer program and algorithm developed by Breiman [52] to "grow" decision and regression trees. Though not the only method for growing trees, it is widely implemented [26]. In CART, and otherwise typically, data is successively partitioned according to successive greater/less-than statements, relative to some "splitting value" or cut-off s_j in the variable j. Input data with values of the splitting variable greater than the splitting value are partitioned into one sub-set, and inputs that have a value lower are sectioned into another. This leaves two regions

$$R_1(j,s) \equiv \{X|X_j \leq s\} \tag{106}$$

$$R_2(j,s) \equiv \{X|X_j > s\} \tag{107}$$

Splits are chosen by finding the point for each variable that minimizes a loss function. The variable, j, which most minimizes the impurity of sub-samples is chosen for the split. This is repeated recursively on each sub-sample until a stop-criterion is reached. The minimization takes the form [44]

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)} L(y_i,c_1) + \min_{c_2}\sum_{x_i\in R_2(j,s)} L(y_i,c_2)\right] \tag{108}$$

Where c_1, c_2 are calculated from their respective regions, are the final predictions of the tree, and minimize the preceding expression given a particular loss function.

For categorization, the loss function is an "impurity" function. Impurity is a metric is calculated by any function with a maximum that is an equal mix of all categories, and minima of zero for samples that only have one type of data (pure). One such impurity function for classification is a zero-one error loss function. In this case, the minimizing values of c_1 and c_2 are just the most common value in their respective regions.

$$\min_{j,s}\left[\min_{c_k}\sum_{k}\sum_{x_i\in R_k(j,s)} L(y_i,c_k)\right] \tag{109}$$

4.4.3 Adaptation for Regression

Decision trees (or classification trees) can be modified for regression by using a different loss function. This also will change the method by which $\hat{c}_k$ is evaluated. A common

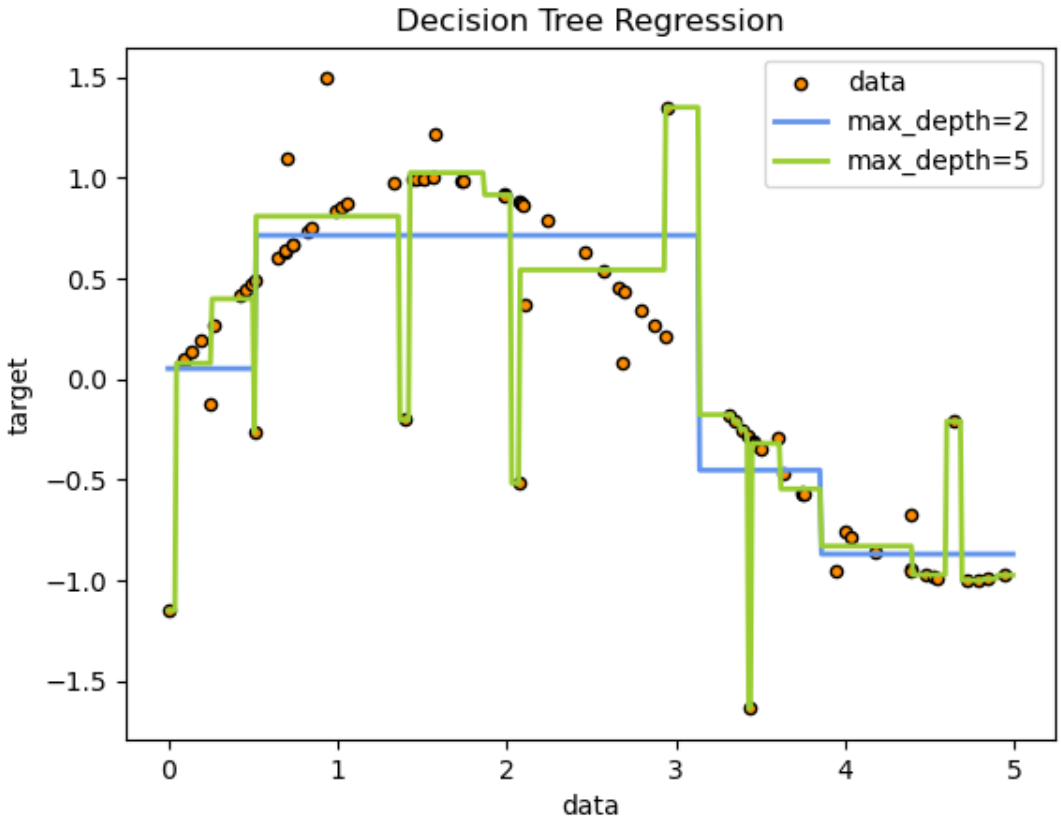


Figure 15: Image from the scikit-learn documentation [26]. Two regression trees are fitted to a noisy sinusoid. The simpler tree, with a maximum depth of two, performs better than a more complex model with maximum depth of five.

implementation of regression trees uses mean-squared error as a loss function. This is minimized by $\hat{c}_k$ equal to the mean value of all items in region k. What results from such a regression tree is a piece-wise function, where in each region the value remains constant at the mean of the training sub-sample. In Figure 15 is an illuminating plot of the ability of a simple decision tree to predict the value of the function $\sin x$. This is sourced from the scikit-learn documentation [26]

4.4.4 Stopping Criteria

Determining when to stop "growing" a tree, by making additional splits, is a non-trivial matter. Since estimation of generalization error is limited to a training set, attempting to minimize empirical error would result in an overfit tree. Wherein, all terminal nodes of a classification tree would contain only a single category (or value) of label, and in the more

extreme case of regression trees, each point would have its own terminal node. Trees that are overfit do not generalize well. In other machine learning models, e.g. neural-networks, one may choose a particular architecture and avoid over-fitting by either stopping training early, or by using other training practices such as "dropout". However, since Decision Trees are "grown" rather than "trained", the degree of "training" is synonymous with tree structure. Therefore, an adjustment to our optimization objective must be made. In The Elements of Statistical Learning Hastie, Tibshirani, and Friedman state [44],

> "One approach would be to split tree nodes only if the decrease in sum-of-squares due to the split exceeds some threshold. This strategy is too short-sighted, however, since a seemingly worthless split might lead to a very good split below it. The preferred strategy is to grow a large tree T_0, stopping the splitting process when some minimum node size (say 5) is reached. Then this large tree is pruned using *cost-complexity-pruning*, "

Cost-complexity pruning modifies the loss function to penalize complex models (at risk for over-fitting). This is justified by the effect of complexity on bias-variance trade-off. As was described in the section 4.3, increased model complexity increases the expected variance and therefore reduces generalization. In section 4.5.1.2, the actual costs associated with complexity were investigated. Since analytical relationships between model complexity and variance are rare, the cost of complexity is estimated by a linear no-threshold model.

$$C = \sum_{m=1}^{|T|} \alpha |T| + \sum_{k} \sum_{x_i \in R_k(j,s)} L(y_i, \hat{c}_k) \tag{110}$$

Where α is a "tuning parameter" depending on the modelling circumstance. $|T|$ is the number of terminal nodes, which is equal to one greater than the number of splits in the tree. This functions as a simple measure of complexity. It is therefore first assumed, reasonably, that the (decreasing) effect on average error per leaf, Q, on the training set is equivalent the generalized effect. The second term contains two more assumptions. It is assumed that the (increasing) effect upon generalized error due to complexity (nominally, variance) increases linearly and without a threshold. Implicit in this assumption is that the variance of model will increase linearly and without threshold according to T.

Though the above metric C is described as a "cost complexity criterion" [44], it is more precisely described as an estimation of the generalization error adjusted to account for

variance resulting from model complexity. Though this estimation is convenient, work by Domingos [49] shows that this assumption is not valid in the case of biased training sets. Furthermore, Breiman derives heuristic evidence that the increase in variance due to additional splits is, at least for classification, bounded by [52],

$$R_2^*(L) \leq \sqrt{\frac{L}{N}} \tag{111}$$

Where R_2^* is a variance-bounding "slow growth factor" increasing slowly with the number of terminal nodes, L, and N is the amount of data. This may provide a more accurate estimation for the cost of complexity. However, even Breiman's CART algorithm uses a linear no-threshold model. As such, as is so common, it is recommended to approximate the parameter α using cross-validation.

4.5 Ensemble Methods

Ensemble methods are algorithms that attempt to leverage and combine the results of learning algorithms to improve overall predictive power. An ensemble model may combine multiple models of the same type, or models of different types. The models that are combined by the ensemble method are called *base models* or alternatively *component models* [53]. Occasionally in this work these models are referred to as *constituent models*. Any advantages gained from ensemble models require a diversity of predictions or outputs from the component models. Therefore, there should be variability in one or more of the fundamental aspects of the component models. Component models may vary on: training set, data type, data dimensionality, output type, output dimensionality, model type, training regime, hyperparameters, model complexity, or any other element necessary for a machine learning model.

Once an assembly of component models has been trained, it must be decided how to leverage any or all of them. Methods to combine the outputs of various models include: a diversity of voting schemes, averaging, stacking, and mixture of experts [53]. The domain of ensemble methods exponentially increases the complexity of machine learning and its possible implementations. In Figure 16 the effect of different ensemble methods on a classification task are compared alongside the performance of the base model.

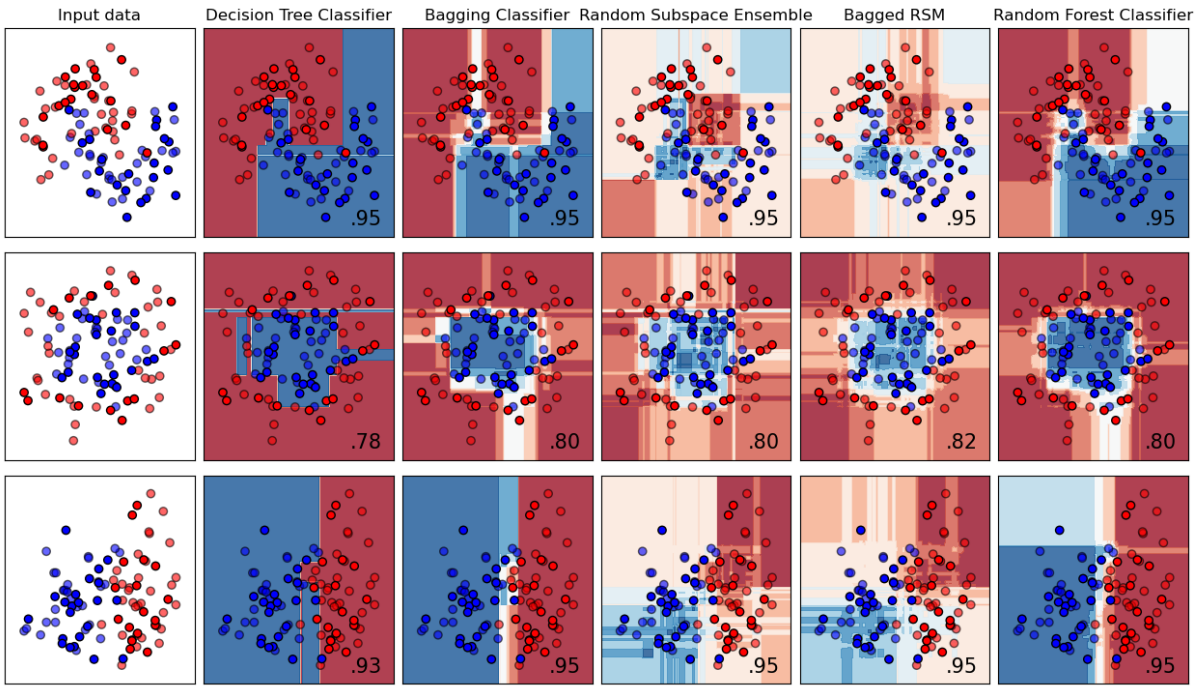


Figure 16: Comparison of ensemble methods. In the far left column, three geometric classifications problems are illustrated. Next, moving right to the second column from the left, is the performance of a single base model, a decision tree. The third and fourth columns from the right illustrate the performances of two different ensemble methods, bagging and the random subspace method (RSM). These methods are combined coarsely in the fifth column from the right; both dimension and training data are bootstrapped. In the column on the far right the performance of a random forest, which modifies the combination of bagging and RSM, is illustrated.

4.5.1 Bootstrap Aggregating

Bootstrap Aggregating or "Bagging" is an ensemble method that improves upon single "base learners" (models) by bootstrapping (sampling with replacement) the training data, then training individual base learner on each sample. Once the models are trained, they are used together as an "ensemble model" when evaluating new or testing data. Bagging is illustrated in Figure 17. Different methods that combine the output of the many base models can be used, such as stacking [54], voting, and averaging [1], depending on the desired model. Breiman describes that the mean can be used for regression models, and voting can be used for classification; though other aggregation methods exist.

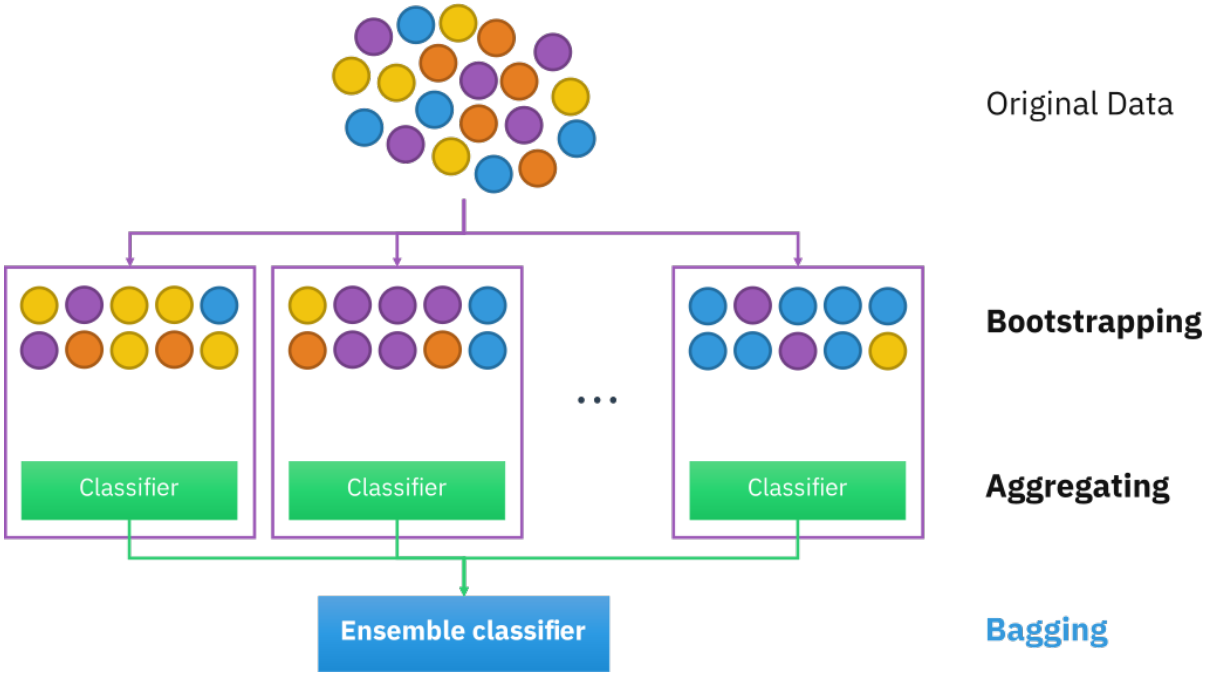


Figure 17: Bootstrap Aggregating Ensemble [55]. Training data is sampled with replacement (bootstrapped) so to train many distinct base models. The predictions of these base models are aggregated to produce an ensemble prediction [44].

4.5.1.1 Accuracy Improvements in Regression with Bagging

Breiman's justification for bagging depends on a standard bias-variance decomposition [1][54][46]. Following Breiman's paper Bagging Predictors [1], given some input $\vec{x}$, the expected value of a bootstrap aggregated predictor $\phi_A(\vec{x})$ is defined as,

$$\phi_A(\vec{x}) \equiv E\left[\phi(\vec{x}, \Gamma)\right] \tag{112}$$

is the average of decision trees (or other sub models) trained on random samples of the dataset, in the sample space Γ. The generalization error, for a single model $\phi(\vec{x}, \Gamma)$ with a mean-squared loss function, is

$$E[y - \phi(\vec{x}, \Gamma)]^2 \tag{113}$$

Which is expanded, and the definition from $\phi_A(\vec{x})$ is substituted in.

$$y^2 - 2E[y \cdot \phi(\vec{x}, \Gamma)] + E[\phi(\vec{x}, \Gamma)]^2 \tag{114}$$

The last term of this can be used in the inequality,

$$E[\phi(\vec{x},\Gamma)^2] - E[\phi(\vec{x},\Gamma)]^2 = V \tag{115}$$

$$E[\phi(\vec{x},\Gamma)^2] \geq E[\phi(\vec{x},\Gamma)]^2 \tag{116}$$

So, returning to the expansion,

$$y^2 - 2E[y \cdot \phi(\vec{x},\Gamma)] + E[\phi(\vec{x},\Gamma)^2] = y^2 - 2E[y \cdot \phi(\vec{x},\Gamma)] + E[\phi(\vec{x},\Gamma)]^2 + V[\phi] \tag{117}$$

$$\geq y^2 - 2E[y \cdot \phi(\vec{x},\Gamma)] + E[\phi(\vec{x},\Gamma)]^2 \tag{118}$$

By omitting the variance of the prediction, the inequality can be written. This is mathematically sound, though what Breiman is actually doing is positing is a definition for the square error of a bagged ensemble model. The most contentious part of the statement is that the omission of the variance of a model's prediction claims that variance of a (constituent) does not affect the ensemble's square error. To reflect the expected error of the aggregated model, Breiman assumes that the prediction of the model and the target value are independent.

$$\geq y^2 - 2y[\phi(\vec{x},\Gamma)] + E[\phi(\vec{x},\Gamma)]^2 \tag{119}$$

$$\geq y^2 - 2y\phi_A(\vec{x}) + \phi_A(\vec{x})^2 \tag{120}$$

Furthermore it is assumed that the variance of the target y is zero, so that

$$E[y^2] = E[y]^2 = y^2 \tag{121}$$

The right-hand side of the inequality, representing the would-be squared error of the aggregated model's prediction, can be factored. The left-hand side is un-expanded to yield Breiman's final statement demonstrating the effect of bagging.

$$E[(y - \phi(\vec{x},\Gamma))^2] \geq (y - \phi_A(\vec{x}))^2 \tag{122}$$

This straight forwards statement neatly describes how an aggregate model will have a mean squared error less than or equal to that expected for a single model. The benefit of the aggregate model depends on the variance in the model's predictions due to the sample Γ, rather than the bias that is implicit in the base model. As a result of bagging, characteristics that appear due to uncharacteristic samples will be mediated by the bootstrapping. Breiman concludes the abstract to Classification and Regression Trees [52] with:

> "If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy."

4.5.1.2 Bagging's Effect on Decomposed Error

Breiman's demonstration appears substantially stronger using the classic bias-variance decomposition of Geman *et al.* [50].

$$E[y^2] - 2E[y\phi(\vec{x},\Gamma)] + E[\phi(\vec{x},\Gamma)^2] \geq E[y^2] - 2E[y\phi(\vec{x},\Gamma)] + E[\phi(\vec{x},\Gamma)]^2 \tag{123}$$

If one assumes that $E[y]$ and $E[\phi_A(\vec{x})]$ are independent, as Breiman does, then

$$-2E[y\phi(\vec{x},\Gamma)] = -2E[y]E[\phi(\vec{x},\Gamma)] \tag{124}$$

$$= (E[y] - E[\phi(\vec{x},\Gamma)])^2 - E[y]^2 - E[\phi(\vec{x},\Gamma)]^2 \tag{125}$$

This provides the elements for the bias-variance decomposition. Returning to the expanded inequality,

$$N + B^2 + (E[\phi(\vec{x},\Gamma)^2] - E[\phi(\vec{x},\Gamma)]^2) \geq N + B^2 + (E[\phi(\vec{x},\Gamma)]^2 - E[\phi(\vec{x},\Gamma)]^2) \tag{126}$$

$$N + B^2 + V \geq N + B^2 \tag{127}$$

Where N, B^2 and V are the noise, squared bias and variance terms of bias-variance decomposition shown in the section Bias-Variance Decomposition. Given the assumptions of a given target y, that y and $\phi(\vec{x})$ are independent, and that bagging does approximate a central tendency of a model distribution (implied since bootstrapping is assumed to be a valid method of sampling), the variance component of generalization error vanishes! Each of these assumptions are not always true, and according to some, are never entirely correct [54][56].

Wolpert's work, On Bias Plus Variance, thoroughly evaluates the circumstance in which a simple BVD is acceptable by using the *Extended Bayesian Formalism*. This is described [56],

> "It is only for a rather specialized kind of analysis, where both q [a central expectation of the test set] and f[the target function], are fixed, that one can ignore the covariance term. "

The identified covariance term is unnecessary, as Wolpert describes, when estimating $E(C|f,m,q)$ i.e. the expected loss given a fixed target, f, an amount of data m, and a test set point q. Given a mean-squared error loss function, the BVD produced conforms to that of Geman [50], Domingos [49], and James [57].

4.5.1.3 Assumptions of Breiman Re-Evaluated

When those requirements are broken, alternative bias-variance-covariance decompositions are derived, along with new assumptions/circumstance. Consider that point in Breiman's justification for Bagging (equation 124), where he assumes that the target and the model are independent. At that point one may instead allow for potential dependence. Then, a covariance term is included, consistent with the derivation of Wolpert,

$$E[C|m,q] = \sigma^2_{m,q} + (bias_{m,q})^2 + variance_{m,q} - 2 \cdot cov_{m,q} \tag{128}$$

Where,

$$\sigma_{m,q} \equiv E[Y_F^2|q] - (E[Y_F|q])^2 \tag{129}$$

$$bias_{m,q} \equiv E[Y_F|q] - E[Y_H|m,q] \tag{130}$$

$$variance_{m,q} \equiv E[Y_H^2|m,q] - (E[Y_H|m,q])^2 \tag{131}$$

$$cov_{m,q} \equiv \sum_{y_F, y_H} P(y_H, y_F|m,q) \times (y_H - E[Y_H|m,q] \times (y_F - E[Y_F|q]) \tag{132}$$

Wolpert intuits the covariance term as a "tracking" of the model to the target over the space of training sets. The sign of the covariance term means that it can contribute meaningfully, even with low noise, bias and variance, in the case of poor tracking between the model and the target.

In Combining Stacking With Bagging to Improve A Learning Algorithm, [54] Wolpert uses this bias-variance-covariance decomposition to describe this circumstance. An improvement of generalization for the bagged version of an algorithm depends on a favourable inner product with the target. This term is contained in the covariance correction. The covariance correction term at least opens the door for scenarios where bagging decreases generalization. In the typical case where the learning algorithm has a positive covariance with the target, the intuition of a favourable inner-product is that the bagged-version will "track" the target more closely than the base-learner. Applying this to Breiman's "cross-over point between instability and stability" defines in greater detail those cases where bagging decreases generalization.

4.5.2 When Bagging does not Improve Accuracy

Imagine a learning algorithm that has great stability in its training but has a high and reliable sensitivity to the amount of training data. In this case the bootstrapped models

would consistently have looser covariance with the target. This bagged ensemble would have a greater expected error than a non-bagged or single base learner.

One such scenario is where the variation within each sample is much smaller than the variation across the whole sample space.

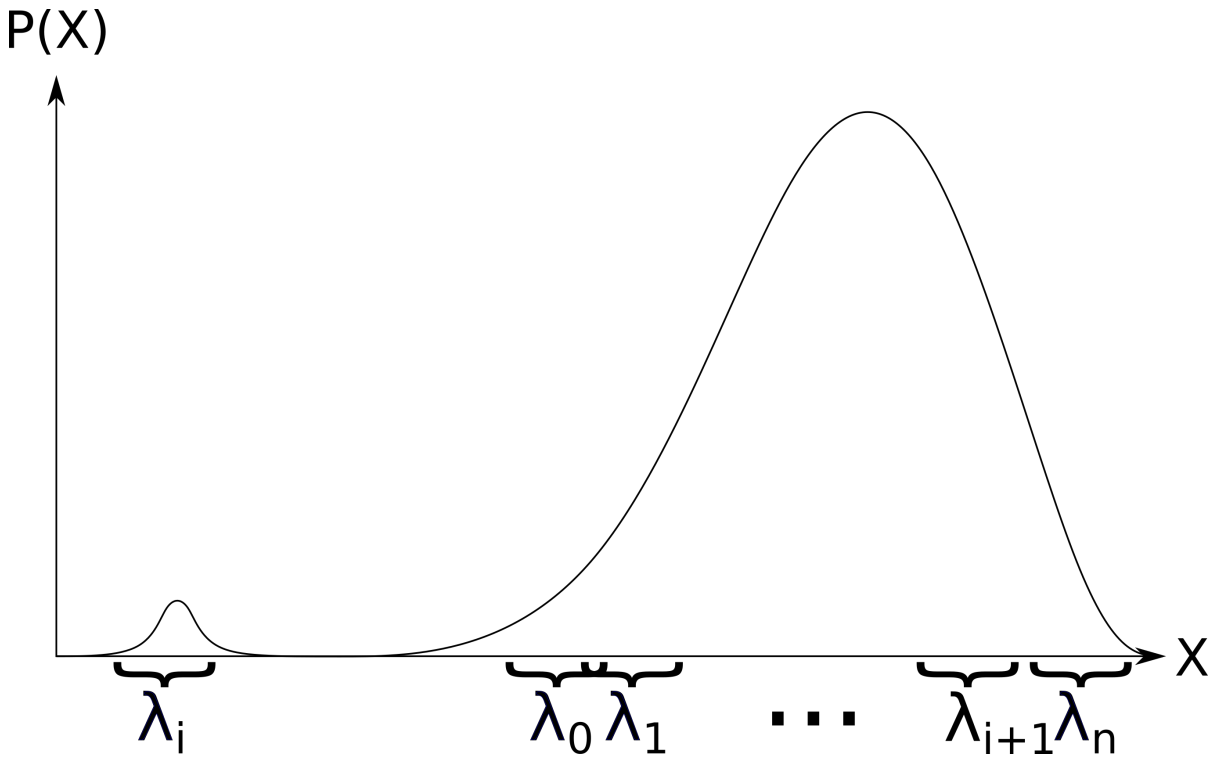


Figure 18: In some circumstance, Bagging may increase error relative to that of a single base model trained on all of the training data. One likely scenario is in the case of asymmetric probabilities among classes or characteristics of data. Data that is more likely to be sampled will dominate the majority of constituent models. Even if some λ_i model does capture the behaviour of the unlikely data, coarse methods of aggregation will overwhelm its contribution.

In Figure 18 is an asymmetric bi-modal distribution will result in a majority (or entirety) of models being having samples from the far right peak. These models would provide inaccurate predictions for parts of the distribution outside of their sample. Even if one or more models were trained using samples included unlikely points, the simple aggregation proposed by Breiman would "swamp" the predictions made by this model.

4.5.3 Stability

At the end of his original bagging derivation for regression, Breiman concludes[1]:

> "a cross-over point between instability and stability at which [the bagged ensemble model] ϕ_B stops improving and on $\phi(\vec{x}, \Gamma)$ and does worse."

The terms "instability" and "stability" here remain qualitative. Moreover, no guidelines are suggested for when such a "cross-over point" is.

In future works, metrics for "stability" are quantitatively articulated, and related to error bounds and probability of performance [58][59]. A framework for stability developed by Bousquet and Elisseeff [46] three different variables, Hypothesis Stability, Pointwise Hypothesis Stability, and Uniform Stability. A connection between stability and expected lower bounds of performance of a trained estimator has also been investigated.

4.5.3.1 Notation Let the input data X and the labels Y constitute the data-space $Z = X \times Y$. A sample D from Z consists of m points. A learning algorithm $\mathcal{A}$, trained on D will produce the mapping f_D that maps $x_i \in X$ to $\hat{y}_i \in Y$, an estimation of $y_i \in Y$. For each estimation, the loss of the estimation is calculated by the loss function l.

4.5.3.2 Generalization, Empirical and Leave-one-out Error Now there is sufficient notation to define the first type of error for the estimator f_D. *Generalization error*, $R_{gen}[f_D]$, is the expected loss for a learned estimator, f_D, on the point $z = (x, y) \in Z$.

$$R_{gen}[f_D] = \mathbf{E}_z[l(f_D, z)] \tag{133}$$

The second type of error for trained estimators in the literature about stability, *Empirical Error* is an estimator of generalization error.

$$R_{emp}[f_D] = \frac{1}{m}\sum_{i=1}^{m} l(f_D, z) \tag{134}$$

The last type of error describes an estimator trained on a slightly modified sample. The dataset D, with a single point z_i removed is denoted $D^{\setminus i}$. *Leave-one-out Error* is the error for a similarly trained model after a marginal change training set.

$$R_{emp}[f_{D^{\setminus i}}] = \frac{1}{m}\sum_{i=1}^{m} l(f_{D^{\setminus i}}, z) \tag{135}$$

4.5.3.3 Metrics for Stability

Elisseeff, Evgeniou and Pontil write [59],

> An algorithm $\mathcal{A}$ has pointwise hypothesis stability β_m w.r.t the loss function l if the following holds:

$$\forall i \in \{1, \ldots, m\}, \quad \mathbf{E}_{D,z}\left[|l(f_D, z) - l(f_{D^{\setminus i},z})|\right] \leq \beta_m \tag{136}$$

Here, it is worth pausing to consider how the stability metric "Hypothesis Stability", β_m describes common-language notions of so-called stability. Algorithms that have a high value of β_m may produce large variations in performance for a marginal change in training set. Conversely, algorithms that provide consistent performance despite changes to training set will have a low value of β_m. The definition of β_m seems somewhat upside-down, though in a useful way. For a perfectly "stable" algorithm, i.e. producing uniform performance regardless of training data, β_m will equal zero.

4.5.3.4 Relationship between Stability and Variance

Following the broad definition of Bousquet and Elisseeff for bias and variance, the Generalized error can be decomposed into these two terms,

$$R_{gen} = E_z[l(f, z)] = B + E[V_z] \tag{137}$$

So, Hypothesis stability can be expressed as,

$$E[|(B + V_{\mathcal{D},z}) - (B - V_{D^{\setminus i},z})|] \leq \beta_m E[|V_{\mathcal{D},z}) - V_{D^{\setminus i},z}|] \tag{138}$$

$$\leq \beta_m \tag{139}$$

Therefore, hypothesis stability is understood to be **the upper limit on expected marginal difference in variance**. Elisseeff, Evgeniou and Pontil continue, describing a relationship between Leave-one-out error and generalized error.

> "[W]hen an algorithm has hypothesis stability β_m, and for *all* training sets D we have, for every $z_i \in Z$, that $0 \leq l(f_D, z) \leq M$, M being a positive constant, then the following relation between the leave-one-out and the expected error holds"

$$R_{gen}[f_D] \leq R_{loo}[f_D] + \sqrt{\delta^{-1}\frac{M^2 + 12Mm\beta_m}{2m}} \tag{140}$$

Where $1 - \delta$ probability of drawing the dataset D from Z.

If bagging increases stability (decreasing β_m) then the above equation vindicates Breiman's conjecture that bagging models from an unstable algorithm will produce an ensemble model with lower expected loss. What remains to be demonstrated is Breiman claim of a "cross-over point" in stability that results in decreased performance of a bagged model.

4.5.3.5 Bagging as a Randomized Learning Algorithm In their extensive paper Stability of Randomized Learning Algorithms [59], Elisseeff, Evgeniou, and Pontil extend the previous framework to non-deterministic training algorithms. Creatively, they propose that bagging of deterministic (or non-deterministic models) can be understood as a non-deterministic model.

With the assumption of bootstrapping (sub-sampling with replacement), and that this can be done while satisfying the requirements for Leave-One-Out Error, the authors prove that the following holds with probability equal to that of training set.

$$R_{gen}(f_{D,\mathbf{r}}) \leq R_{emp}(f_{D,\mathbf{r}}) + 2\beta_m + \left(\frac{M + 4m\beta_m}{\sqrt{2m}} + \sqrt{2T}\rho\right)\left(\frac{\sqrt{\log 2}}{\delta}\right) \tag{141}$$

Where T is the number of models included in the bagging aggregate, and ρ is the supremum of marginal difference in loss relative to a single model $\mathbf{r}_i$. With the additional assumption that the loss function is "B-lipschitzian" with respect to f_D, Elisseeff *et al.* also prove that the stability of a bagged regression model is bounded by

$$\beta_m \leq B\sum_{k=1}^{m}\frac{k\gamma_m}{m}\mathbb{P}_{\mathbf{r}}\left[d(\mathbf{r}) = k\right] \tag{142}$$

Where each possibly quantity of distinct points in a bootstrap is represented by $d(\mathbf{r})$, and γ_k is the stability of a constituent model trained on k distinct points.

For a very strict scenario, Elisseeff *et al.* do demonstrate the stability improvements of bagging relative to a single model. Where the number of distinct points is "sharply concentrated", and $B = 1$, and that γ_m "scales appropriately with m"; then bagging predictors will have a lower value of β_m , and therefore a lower ceiling on expected loss, than a single model.

4.5.3.6 Conclusions on Bagging reducing accuracy Though the theory developed by Elisseeff *et al.* does yield some interesting properties regarding stability, they admit:

> "Although we can derive bounds for bagging using our theory [...] our results for bagging do not show that bagging actually improves performance."

This point was even more strongly stated by Yves Grandvalet [60], when he presented a case study of bagging increasing variance of a decision tree model,

> "We hope that the present counter-example [...] will contribute to amend the folk belief that bagging simply reduces variance in its averaging process [sic]"

He further suggested that,

> "[T]he stability of bagging global predictors such as decision trees seems not to be amenable to a theoretical analysis"

Despite a lack of definitive guidelines of when bagging and how to use bagging in order to improve model performance, there is a well established "belief" that bagging does *often* improve performance. This is bolstered by the great multitude of studies where bagging is empirical shown to improve performance.

4.5.4 Random Subspace Method

The Random Subspace Method (RSM) and its variants create an ensemble of methods where for each at some point during training, the prediction (or predictive subset) is limited to a sample of the possible input variables [2][3].

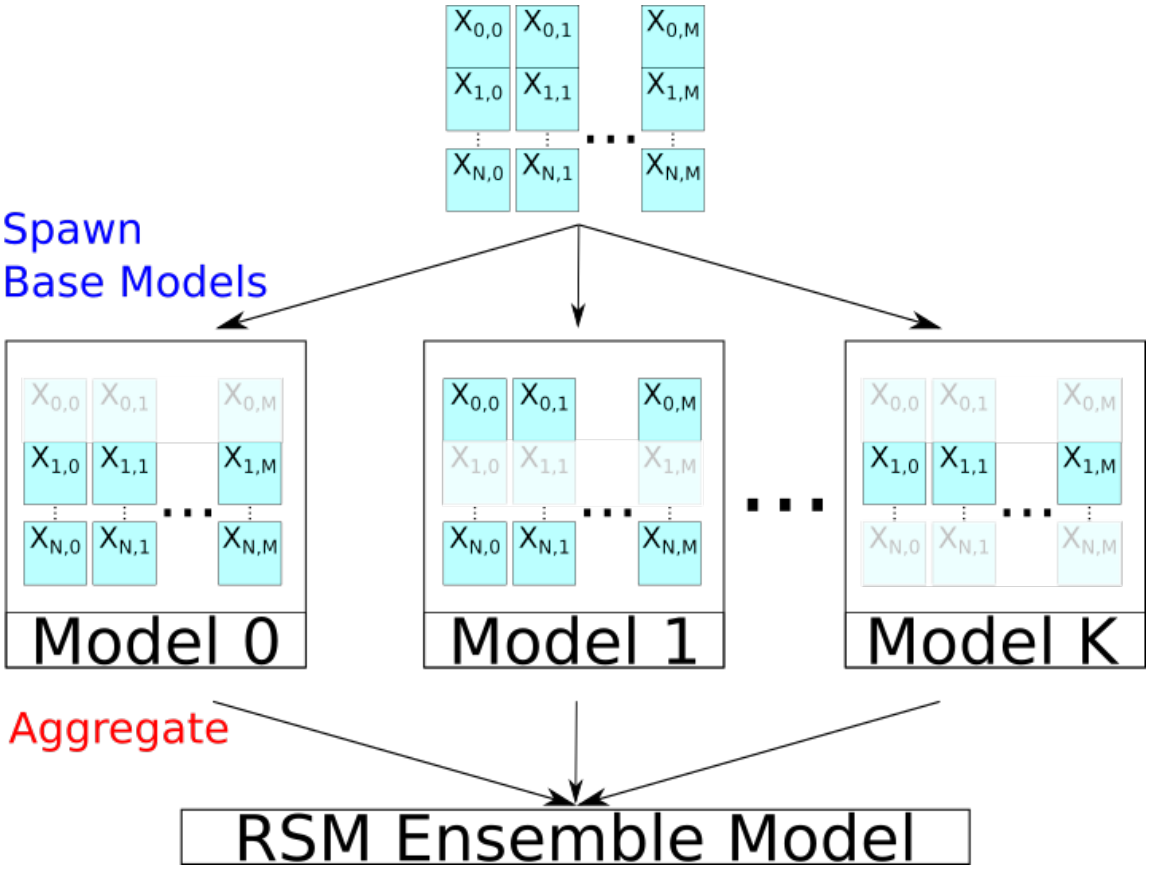


Figure 19: Illustration of the random subspace methods (RSM). The dimensions of input data are sampled with replacement (bootstrapped) so to train constituent models.

By doing so, each constituent model thus specializes in the training data's structure in each subspace. Ho describes one straight forwards benefit of this,

> "For each [model], the classification is invariant for points that are different from the training points only in the unselected dimensions"

It seems evident to the author that this may also be a drawback; omitting discriminating variables can similarly make points invariant from dissimilar training points. Ho claims that this avoids the "curse of dimensionality"; the incorporation of additional dimensions does not add create additional training requirements for each constituent model. Therefore, each subspace may be limited to a more manageable size for the base model type.

4.5.4.1 Effect of Random Subspace Method on model performance

In the author's review of the literature, no applied mathematical description of the benefits of RSM have been found. Ho does direct his reader to consider *Stochastic Discrimination* as the foundation for the apparent improvement. All subsequent analysis and comparison between RSM and either single models or ensemble model methods, have been empirical.

From empirical studies Ho suggests that RSM improves classification performance on compact datasets with smooth boundaries. This paper contrasted RSM with Bagging, which Ho suggests improves sparse datasets [61]. In a further empirical investigation, Skurichina and Duin provide evidence for conditional improvements on performance [62]. In their "User Guide", based on their experiments, they suggest RSM with Fisher Linear Discriminators for training data amounts less than or similar to the dimensionality of the data (but not greater), for Pseudo Fisher Linear Discriminators with an amount of training data similar to the dimensionality of the data (but not less or more than), and for Nearest Mean Classifiers with less than or similar amounts of training data relative to the dimensionality of the data.

These two empirical investigations into the effect of RSM do not provide consistent descriptions of the benefit of RSM. However, they do demonstrate the important point that any benefits of RSM are conditional on the type of constituent model, the amount of training data, the data dimensionality, and likely other factors. Therefore, for RSM to be used, it is reasonable to empirically verify that the particulars of one's application do in fact benefit from RSM.

4.6 Random Forests

Breiman, in his 2001 paper Random Forests [3], combines and extends the two ensemble methods of bagging and the RSM. He then applies this to decision trees. As with bagging, each constituent model is trained on a bootstrapped sample. Breiman then extends RSM by randomizing the selection of features for each "split" of the decision tree. This extends RSM, since in RSM the selection of features was only randomized once per constituent model.

4.6.1 Combining Bagging and RSM

The arguments made Breiman assumes the expected value of a base learner, h, trained on any randomly selected vector Θ, evaluated at some point X, is equal to the expected value of the target, Y. It is then provable that the generalization error for a random forest is bounded by a scaled expectation error for a single base learner. The scaling factor $\bar{\rho}$ is the weighted correlation of residuals between two base learners in the ensemble. This theorem is expressed,

$$PE^*(forest) = \bar{\rho}(E_\Theta[\sigma(\Theta)])^2 \leq \bar{\rho}PE^*(tree) \tag{143}$$

Where $\bar{\rho}$ is the "weighted correlation between the residuals $Y - h(\mathbf{X}, \Theta)$ and $Y - h(\mathbf{X}, \Theta')$ where Θ, Θ' are independent". Then he states,

> "[This theorem] pinpoints the requirements for accurate regression forests - low correlation between residuals and low error trees. The random forests decreases the average error of the trees employed by a factor of $\bar{\rho}$. The randomization employed needs to aim at low correlation"

The former of these two requirements is more simply understood as that among the ensemble there should be few collective weak points, when all the constituent models have high error. The latter requirement of "low error trees" seems trivial; low error is always nice. Similarly, Hastie [63] includes as an exercise to derive the variance of a bagged ensemble's prediction. This is accomplished by beginning with an equation for the variance of a sample mean [30],

$$V[\bar{x}] = E[\bar{x}^2] - (E[\bar{x}])^2 \tag{144}$$

Expanding, with the expected variance and mean of different constituent models to be identical (though not necessarily independent)

$$V[\bar{x}] = \frac{1}{n^2}\sum_{i,j=1}^{n} E[x_i \cdot x_j] - \mu^2 \tag{145}$$

$$= \frac{1}{n^2}\left(\sum_{i\neq j}^{n} E[x_i \cdot x_j] + \sum_{i=1}^{n} E[x_i^2]\right) - \mu^2 \tag{146}$$

$$= \frac{1}{n^2}\left(\sum_{i\neq j}^{n} (cov[x_i, x_j] + \mu^2) + \sum_{i=1}^{n} E[x_i]^2 + V[x_i]\right) - \mu^2 \tag{147}$$

$$= \frac{1}{n^2}\left((n^2 - n)(\rho V[x] + \mu^2) + n(\mu^2 + V[x])\right) - \mu^2 \tag{148}$$

$$= \rho V[x] + (1 - \rho)\frac{V[x]}{n} \tag{149}$$

Where ρ is the pairwise correlation among constituent models, σ^2 is variance of each constituent model and n is the total number of constituent trees. Clearly, as the correlation among the prediction of different constituent models approaches 1, the variance of the prediction approaches that of a single constituent model. Whereas, with increasing correlation (greater than zero) the variance approaches the expectation of a sum of independent and identically distributed variables. This is why the variation of the RSM is included; doing so decreases the correlation among constituent models, and therefore decreases the expected variance of the ensemble.

4.6.2 Derivation of Difference in Error

What is not demonstrated by Hastie or Breiman is that decreasing the variance of the aggregate by decreasing correlation between constituent models, does in fact decreases the overall expected error. Moreover, to prove that his bagging method works, Breiman explicitly left out the variance of the prediction to create an inequality (see equation 117). This was an implicit definition of the square error of a bagged ensemble model. That definition asserted that the variance of a constituent model is not included in the square error of the ensemble model.

Here, instead, a relation between an estimate for ensemble variance and variance of a constituent model is derived. Beginning with the expected square error of the ensemble

model,

$$E\left[\left(y-\bar{\phi}\right)^2\right] = E\left[y^2\right] - 2E\left[y\cdot\bar{\phi}\right] + E\left[\bar{\phi}^2\right] \tag{150}$$

$$= E\left[y^2\right] - 2E\left[y\cdot\bar{\phi}\right] + E\left[\bar{\phi}\right]^2 + V_{\bar{\phi}} \tag{151}$$

Since the $\bar{\phi}$ is the mean of a set of identically distributed predictions, ϕ, therefore they have the same expectation value. Assuming, as Breiman does, that the prediction ϕ and target y are independent, one can replace the second and third terms with those of the expanded expected square error of the single base model.

$$= E\left[y^2\right] - 2E\left[y\right]E\left[\phi\right] + E\left[\phi\right]^2 + V_{\bar{\phi}} \tag{152}$$

Replacing the squared expected prediction using the identity for variance allows for the expected square error of a single base model to be found,

$$E\left[\left(y-\bar{\phi}\right)^2\right] = E\left[y^2\right] - 2E\left[y\right]E\left[\phi\right] + E\left[\phi^2\right] - V_{\phi} + V_{\bar{\phi}} \tag{153}$$

$$= E\left[\left(y-\phi\right)^2\right] - V_{\phi} + V_{\bar{\phi}} \tag{154}$$

Substituting in the estimator for ensemble variance,

$$E\left[\left(y-\bar{\phi}\right)^2\right] = E\left[\left(y-\phi\right)^2\right] - V_{\phi} + \rho V_{\phi} + (1-\rho)\frac{V_{\phi}}{N} \tag{155}$$

Rearranging finally yields a statement regarding the effect on expected error by using bagging.

$$E\left[\left(y-\bar{\phi}\right)^2\right] = E\left[\left(y-\phi\right)^2\right] - \frac{N-1}{N}(1-\rho)V_{\phi} \tag{156}$$

This new derivation plainly shows the impact of bagging on the expected error of a base model. Included are factors describing the effect of increasing the number of constituent models, the impact of the correlation among those models, and the (expected) variance of each model. The latter two affirms Breiman's assertion that less stable models benefit more from bagging. Decreased correlation among constituent models increases performance of a bagged ensemble relative to the base model. Furthermore, base models with high expected variance, V_{ϕ}, will have a greater improvement when bagged.

This analysis has so far ignored the potential impact of covariance between model and target. There still exists potential circumstance where bagging could reduce generalization. Using the Bias-Variance-Covariance decomposition by Wolpert [56] one can imagine

that by decreasing the correlation between constituent models, one could also decrease the covariance between a model and its target. The same uncertainty that was described in the improvements by bagging are present for random forests.

$$E\left[\left(y-\bar{\phi}\right)^2\right] = E\left[y^2\right] - 2E\left[y\bar{\phi}\right] + E\left[\bar{\phi}\right]^2 + V_{\bar{\phi}} \tag{157}$$

$$= E\left[y^2\right] - 2\left(E[y]E\left[\bar{\phi}\right] + cov\left[y,\bar{\phi}\right]\right) + E\left[\bar{\phi}\right]^2 + V_{\bar{\phi}} \tag{158}$$

$$= E\left[y^2\right] - 2(E\left[y\phi\right] - cov[y,\phi]) + E\left[\phi^2\right] - V_{\phi} + V_{\bar{\phi}} - 2cov\left[y,\bar{\phi}\right] \tag{159}$$

$$= E[(y-\phi)^2] - \frac{N-1}{N}(1-\rho)V_{\phi} - 2(cov\left[y,\bar{\phi}\right] - cov\left[y,\phi\right]) \tag{160}$$

By including the covariance similar to Wolpert, another intuitive description is found. Using an intuition akin to Wolpert's, one finds that a further difference between the error of the bagged ensemble to a single base model. To the degree that the bagged ensemble "tracks" the true better (over the space of training sets) than a single base model, there will be an additional decrease in expected squared error.

5 Results

This results chapter documents two projects demonstrating how machine learning can be applied to inverse design. The first project constitutes the majority of the research completed for this work. It investigates the suitability of surrogate machine learning models for substitution into a design process. In this first design project we acquire data from the Monte Carlo simulation software EGSnrc, then train a random forest regressor to act as a surrogate model. The surrogate model is used in a simulated annealing design optimization demonstration to predict the behaviour of previously unsimulated designs. The results and costs of a design optimization using the surrogate model are contrast with those using the EGSnrc simulation directly. It is found that the random forest regressor produces a simple estimation sufficient for effective design. A fast surrogate greatly decreases the overall time required for the design optimization, without any depreciation in resulting design relative to that of the EGSnrc-based optimization.

The second project, nicknamed "PickAI", is a proof of concept for a design-space spanning neural network. It demonstrates how neural networks can act as a transform from an arbitrary random data-space, and have a surjective range within a design-space. Configurations of a two-dimensional Ising model are taken to be 'designs'. These are produced as the output of a convolutional neural network. Different configurations are produced by varying the random input to the network. Optimization on the convolutional neural network is achieved by NeuroEvolution. This entails many candidate networks being evaluated for their 'fitness', and the most successful network (along with new variations) proceeding. The 'fitness' of the candidate networks is determined by the properties of the distribution of characteristics. In the proof of concept, candidate networks that produce a set of configurations with more uniformly distributed internal energies are measured as more fit. It is found that this method produces a more uniform distribution of configurations than an untrained network.

5.1 EGSnrc Simulation

The EGSnrc simulation setup used here is a beam of γ rays incident upon a 'scattering object'. This scattering object could serve as a mask or shielding. The γ-ray beam has a Gaussian spectrum, with a standard deviation varying from 0.01 MeV to 1 MeV, depending on the trial. The simulation setup is shown isometrically in Figure 20.

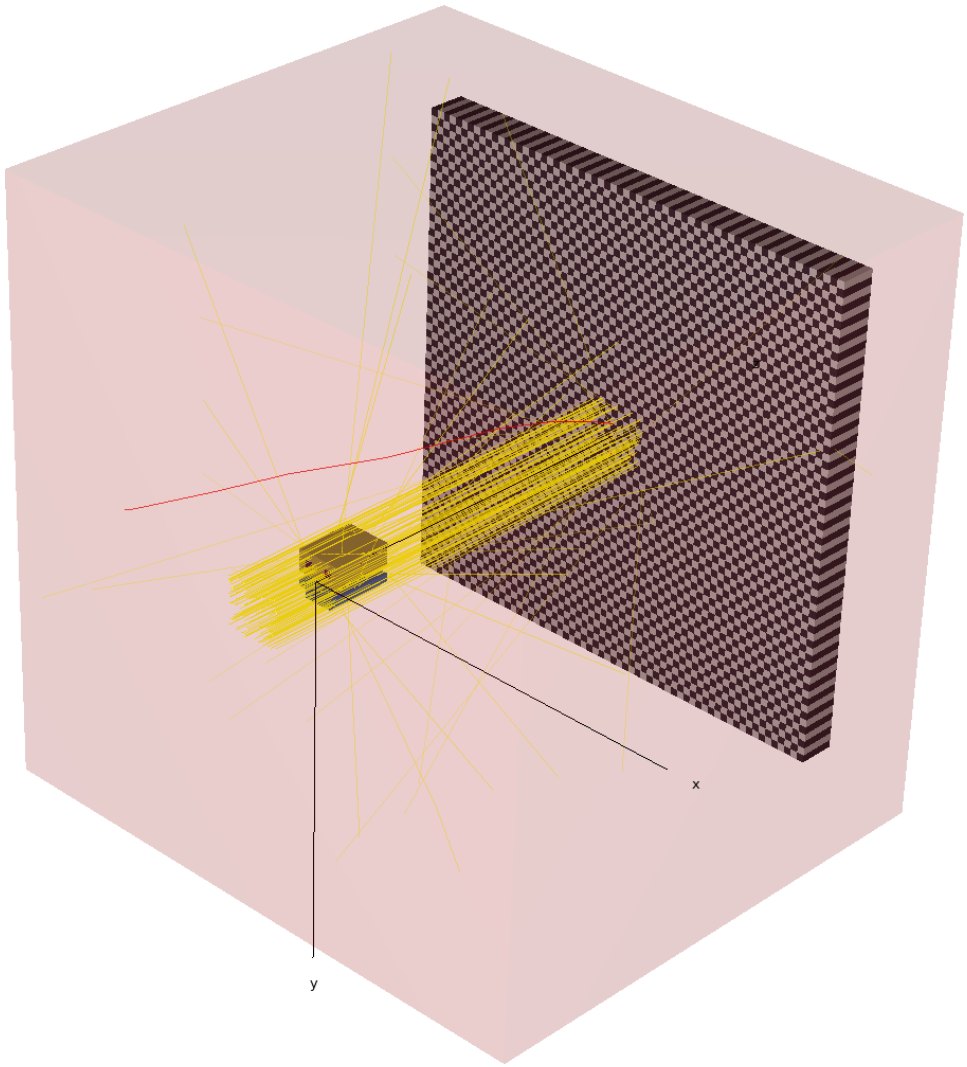


Figure 20: Isometric view of EGSnrc simulation setup. γ-rays, marked by yellow lines are incident upon a scattering target. The γ-rays are scattered in different directions and some produce secondary electrons, marked by red lines. Radiation is collected behind the scattering target by a detector wall.

The γ-ray beam is incident along the z-axis. The γ-rays are depicted as yellow trajectories. Interaction with the scattering object is observed to produce electron radiation. Any present electron trajectories are shown in red. The materials involved in this particular simulation are better observed in the orthographic diagram in Figure 21

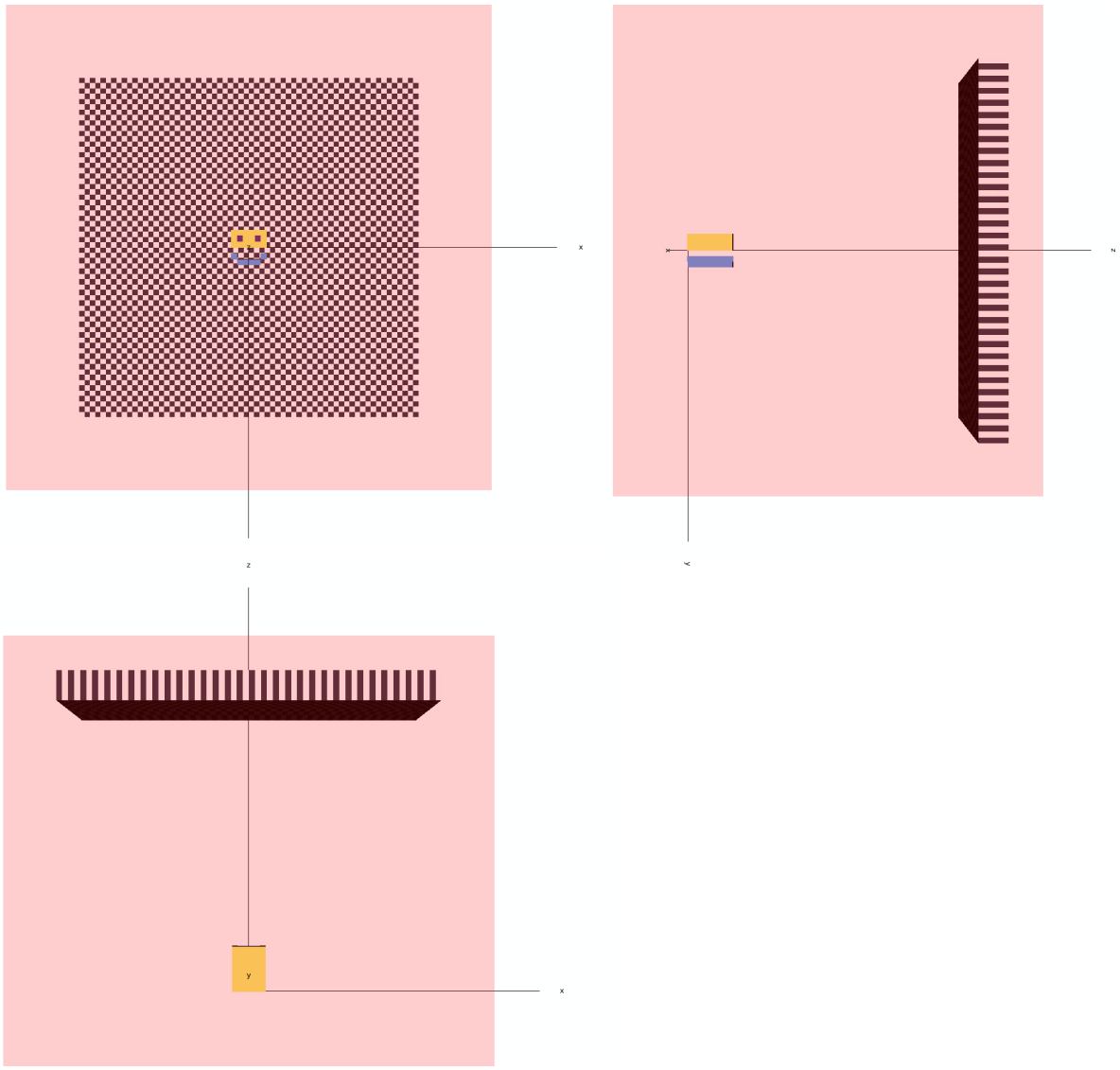


Figure 21: Orthographic view of simulation setup

The scattering object is made of voxels of different materials. The faint red cube is air (near sea-level). The detector wall, behind the scattering object, is composed of germanium. The checkerboard pattern only contributes to the ease of visualization. The scattering object is composed of graphite, aluminum, lead and air (i.e. points can also be left empty, and so filled with the atmosphere material air). In this representation graphite is beige, aluminum is light blue-grey, and lead is purple-grey. These materials were chosen as common examples of materials with varying radiation stopping powers.

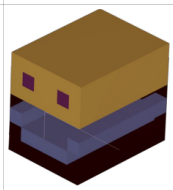
EGSnrc simulations break down the energy absorbed by each region of the simulation. The absorbed dose values for the germanium detector are collected as one label (or source

of scoped labels) for the surrogate model. Representations of the simulation results are also illustrated in Figure 23.

5.2 Surrogate Model

5.2.1 Generation of Training Data

In order to conduct an unbiased search of the 'design-space', scattering objects are randomly sampled as follows. To clearly demonstrate the predictive ability of a surrogate model, we limit the design space to designs invariant along the axis of incident radiation (z-axis). Therefore, for an $8 \times 8 \times 8$ scattering object there are $8 \times 8 = 64$ 'indices' for which a material must be chosen. Let there be N materials be available for scattering object designs. Each possible design can be represented by a 64-digit number in base N. Therein, each digit represents the material chosen for a particular position in the design. In the table below, the conversion between random numbers and a scattering object is illustrated. A single digit corresponds to a single voxel. Voxels with the same digit value are composed of the same material. A 64 digit number (repeated 8 times) corresponds to a $8 \times 8 \times 8$ scattering object.

Random Number	Random number in base N (e.g. 4)	Object
1	1	
1365	111111	
43259488272928240024240031531008O	1111110013113100111111000000000200002000222 20000000000 (repeating 8x)	

Once the random number from 1 to N^{64} is sampled, a simulation produces data according to the following procedure. The random number is converted into base N. The material configuration file is automatically generated. Copies of other configuration files are made with a file name that contains the random number. Placeholder simulation parameters are stream edited to have a specific or default value. Some examples of these simulation parameters are, the number of particles simulated, the standard deviation of

particle energy, and the simulation's random seed. Then the input file is passed into the simulation software. Since the file name is tagged with the random number, the simulation reads from the recently generated configuration copies. The output of the simulation, a human-readable text document, is then parsed into a `.csv` file counting the absorbed dose of each voxel of the detector. The material configuration file is parsed into a `.csv` file. This provides an input to the surrogate model.

To speed up the data generation process, GNU parallel [64] is used to run many of the above described 'single-run' at once. The POSIX shell script to parallelize the data generation is included in Annex 1 of the supplemental material to this work. A posix shell script that handles the generation of input files and the parsing of output files for a single EGSnrc simulation is included in Annex 2 of the supplemental material to this work. For the generation of data from many simulations, a directory is created to store all of the outputs. Similarly, all of the produced `.csv` are loaded into a `.hdf5` data file. Here, materials still follow a simple numeric encoding.

5.2.2 Surrogate Model Development

The python code that processes the testing and training data and trains the surrogate model is included in Annex 3 of the supplemental material to this work.

5.2.2.1 Scoping Scattering Objects For a given detector voxel, at some distance from the scattering object and source, there is a field of view that captures all consequential parts of the scattering object. Moreover, the relative effect the voxels of the scattering object have upon a voxel of the detector typically decreases proportional to the tangential element of the radial displacement between voxels (tangential relative to the direction of the incident irradiation). This tangential component comes from scattering off of the target scattering object.

The consequence for training a surrogate model is that a majority of the relevant information for predicting absorbed dose on a detector voxel is limited to a narrow field of view (from the perspective of the detector voxel). This provides an opportunity to more effectively leverage the training data. By 'scoping' fields of view for each detector voxel, 64 different data points are created from a single simulation. Instead of using the entire scattering object as an input datum and using the entire detector results as the label,

one can make these 64 predictions voxel-wise using a narrow field of view as the input datum for each voxel label. This is illustrated in Figure 22.

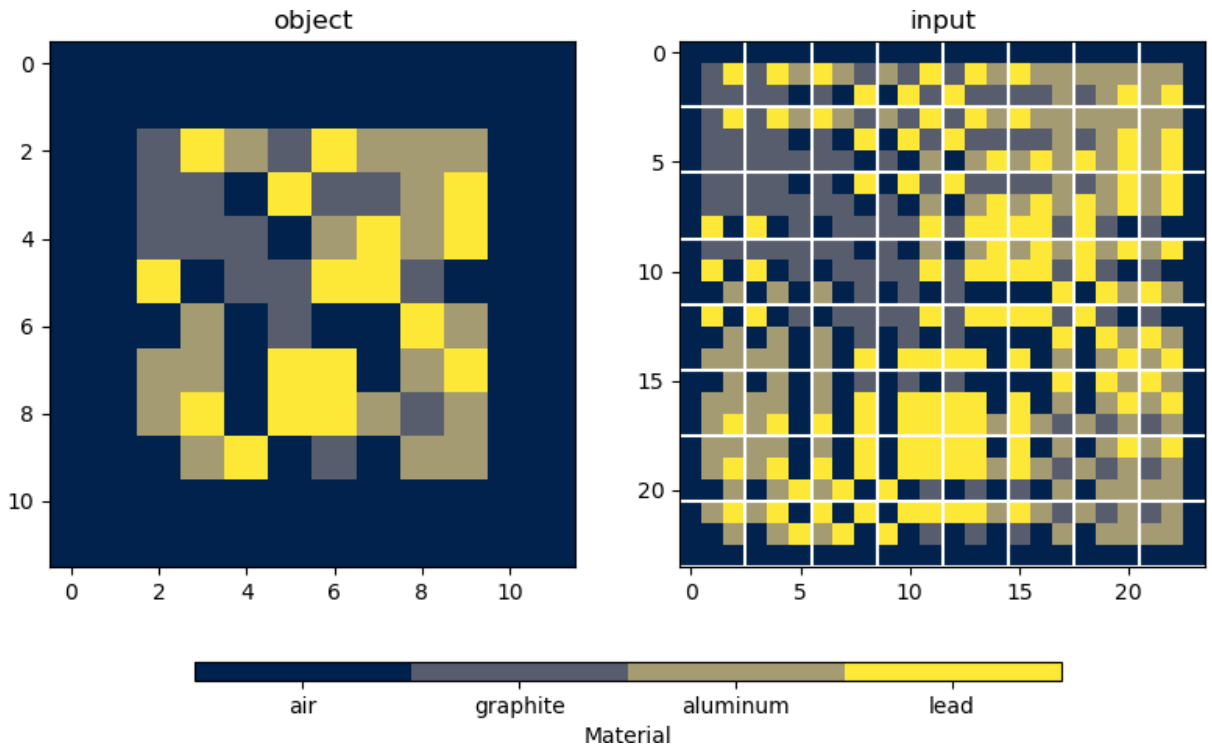


Figure 22: For each voxel of the scattering object, a datum including its field of view is produced. This is used as the input for a single prediction from the surrogate model.

On the left a full view of the scattering object from the perspective of the beam. For each voxel of the detector, directly in front of the scattering object, a 3×3 field of view is selected. This field of view is the input data for the predicted absorbed dose on that detector voxel.

5.2.2.2 Implications of Scoping Simulation Data Making 64 field-of-view / voxel-wise data points for each simulation has similar implications as those proposed by Ho regarding the "Random Subspace Method" (RSM) [2]. As was quoted in section 7 of Results, by omitting partial input information there is convergence between distinct items in training and test sets. Segments of a single design or configuration may be common between distinct items in the test and train sets. Ho regarded this as an advantage of the RSM method. Furthermore, Ryczko *et al.* show that test/train overlap due to voxelization does not impede effective generalization [65]. It may seem incorrect to interfere with the test-train split. However, as with any other decision made during

the preparation of a machine learning model, this should ultimately be considered in context. What is desired is a model capable of accurately predicting the absorbed dose of novel scattering objects. Scoping the simulation results into voxels does not interfere with this goal. However if one were probing the predictive power of the model, rather than the utility of such a model for design, then a more strict test-train split would be necessary. Such a split would ensure no common fields of view existed in both the test and train sets.

5.2.3 Surrogate Model Performance

The Scikit-Learn implementation of a random forest regressor is selected for this work [26]. Random Forests, as discussed in depth in the Methods chapter, are a reliable and flexible machine learning method [44]. We found the performance of random forests regressors to be reasonably good. As a consequence of the mechanics of decision trees, random forest regressors train to an average value of 'grouped' or similarly identified values.

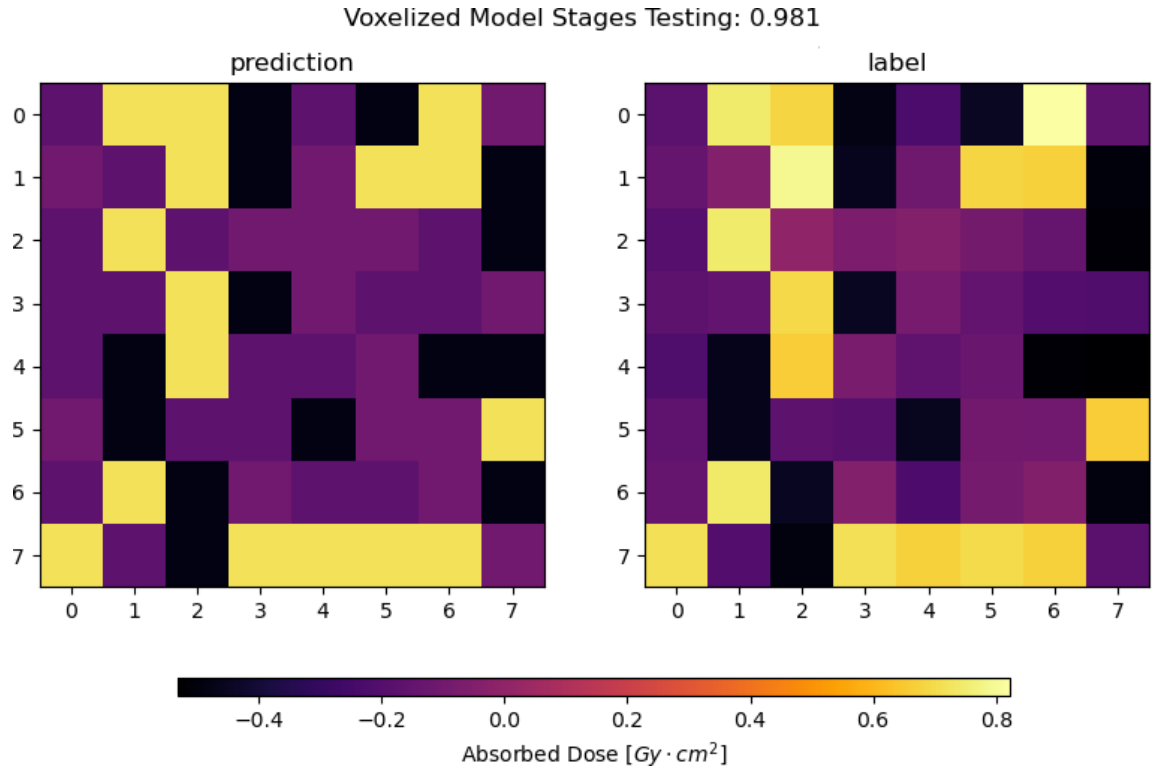


Figure 23: The output of the surrogate model (left) provides an accurate prediction of the EGSnrc simulation output (right). The coefficient of determination for the entire test set is 0.981

Moreover, the performance of the model is positively correlated to the number of particles simulated and amount of data produced. Model performance is negatively correlated with variance of the source energy. In Figure 24, three contour plots are included. They illustrate the r^2 value for random forest regressors for a range of simulated particles, on each x-axis, a range of beam energy standard deviations, on each y-axis. Both x and y axes are plotted as log of their original values. Each plot illustrates the performance of models, trained on the range of simulation parameters, but each having been trained on a different quantity of data. From left to right, the amount of data produced for each parameter sweep is 10 and 100.

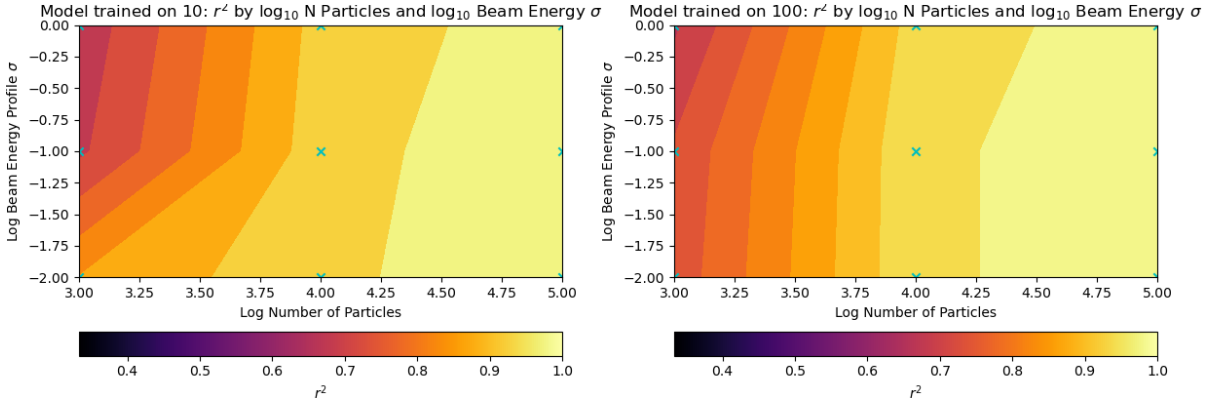


Figure 24: Sensitivity analysis of the coefficient of determination for random forest regressor models produced with different amounts of training data. Both plots are log-log heat maps that illustrate the effect of the variance in beam energy and the number of particles simulated. The models trained for each plot were tested and trained on the results of 10 and 100 simulations, for left and right respectively.

In Figure 25 the variation in model performance is illustrated for a model trained and tested on 1000 simulations.

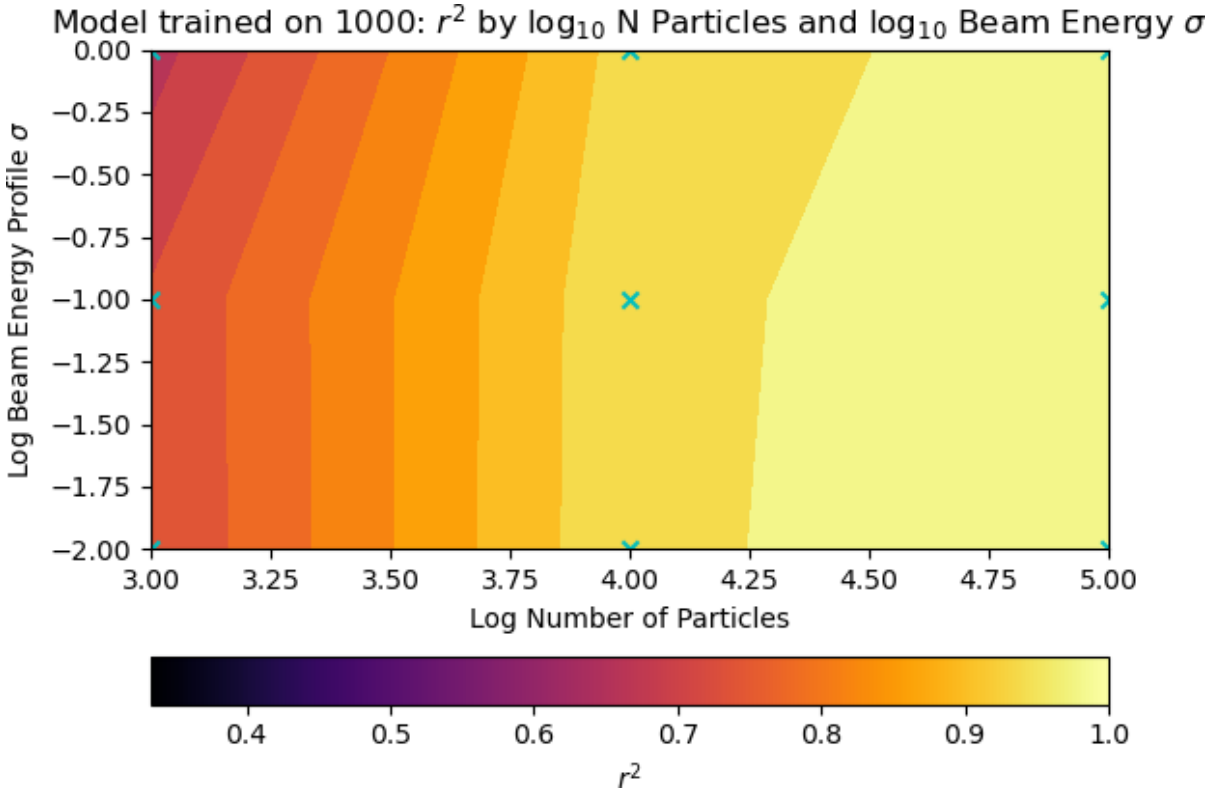


Figure 25: Sensitivity analysis for the coefficient of determination for random forest regressor produced and tested using 1000 training data points.

The points marked in cyan crosses are the r^2 values for the models trained in the parameter sweep.

Random forests do not require as much data for reasonable performance as one might expect for neural networks [44]. Above some sufficient amount of data, (on the order of one hundred data points), the marginal improvement of model performance from additional training data decreases. The number of particles simulated has a large effect on the model's explicit performance. This is because the majority of variance in absorbed dose, in this particular set up, is due to spatial variance of particles incident. In Figure 26, it is observed that the spread of observed values for a given prediction shrinks as the number of particles simulated increases.

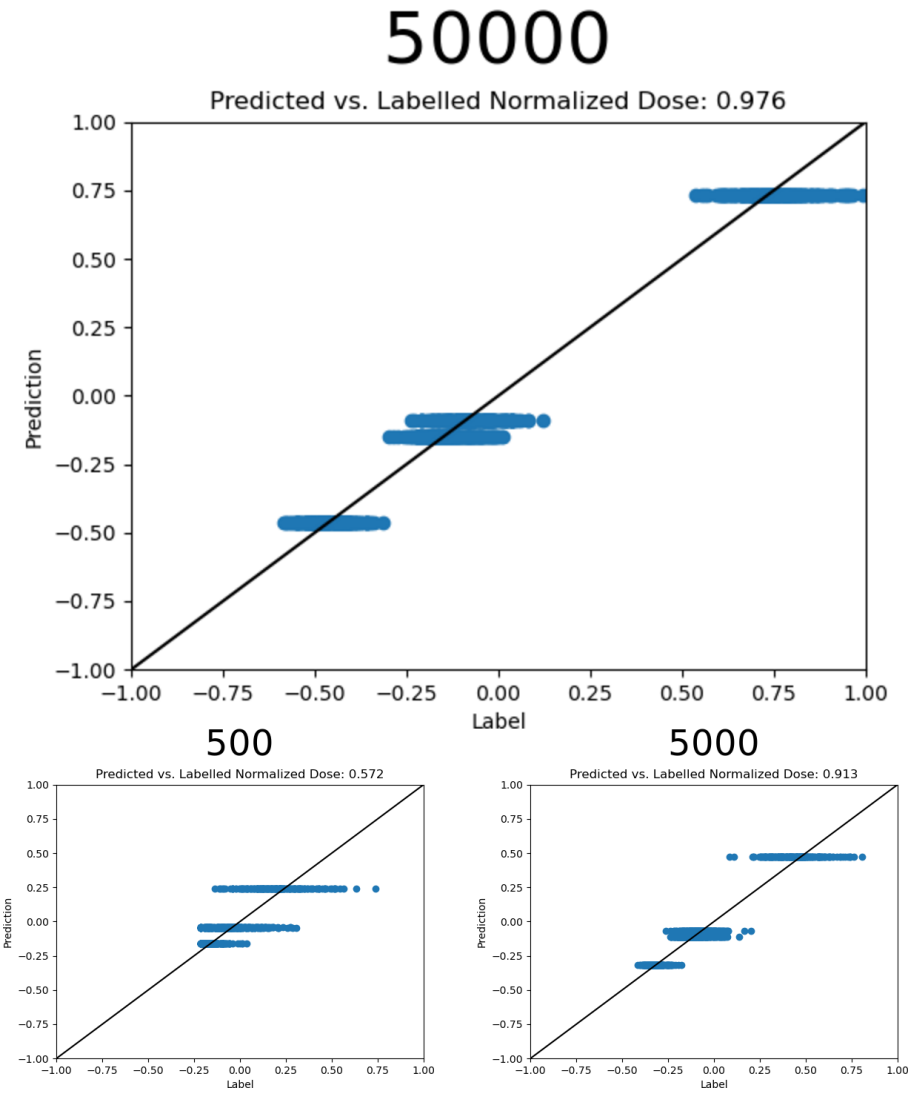


Figure 26: Predicted vs Expected values of normalized absorbed dose for ML models produced with different amounts of training data.

The r^2 value calculated, from the test-set, for the plots in Figure 26, 0.572, 0.913 and 0.976 for 500, 5000 and 50000 particles simulated respectively. It is further observed that there are four distinct predictions that the surrogate model makes. This aligns with the number of materials included in the scattering object. Below in Figure 27 the spread of normalized absorbed dose is illustrated per material, along with the surrogate model's prediction (marked by the red in the box).

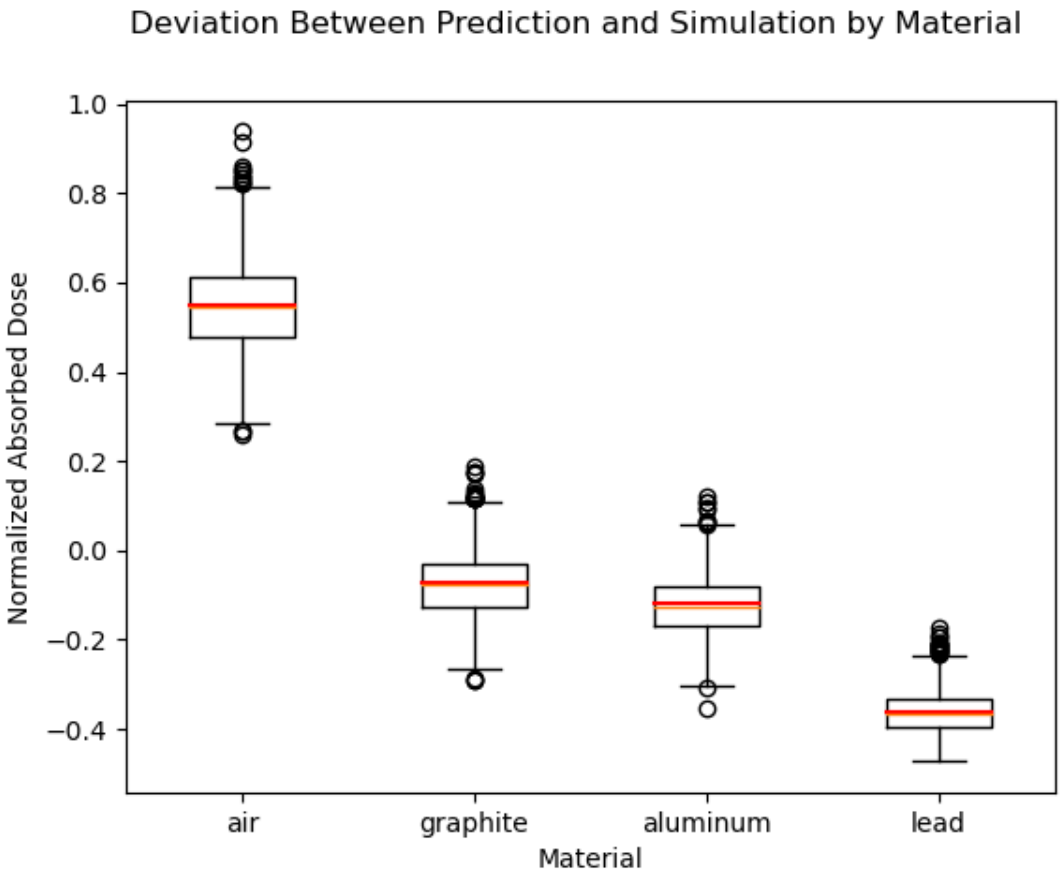


Figure 27: Distribution of normalized absorbed dose. Results are separated by the material type. The surrogate model's predictions (marked by a red line in each box) are consistent for all materials. Simulation parameters were 10000 particles simulated with a 0.1σ spread on a 1 MeV Gaussian beam and measured on $124 \times 64 = 7936$ test voxels.

As is discussed in Chapter 2: "Introduction to Inverse Design", this has benefits for design optimizations in highly stochastic design environments. Though such a prediction minimizes the mean squared error [44], it does mean that the predictions of the present model may be uncharacteristically regular. This is considered in the next section. Since the simulation that is to be replaced is stochastic, it is appropriate to measure performance according to goodness of fit. This is described in the section "Utility and Accuracy" in Chapter 2.

5.2.4 Goodness of Fit Measurements

For stochastic simulations measuring the generalization using the coefficient of determination, r^2, or a loss function, e.g.

mean squared error, can obscure the performance of the trained model. A goodness of fit metric, such as likelihood, provides a different perspective on a model's performance.

Here it is advisable to avoid assumptions about how the absorbed dose at each voxel is distributed. In each case, a single test-case is repeatedly simulated to produce a histogram of normalized absorbed dose for each detector voxel in the region of interest. The likelihood of the prediction provides a measure of goodness-of-fit. Unlike using a χ^2 test, how the likelihood of a particular fit is distributed is unknown apriori [30]. Therefore, as a test-case, the results from a single scattering object are repeatedly simulated. Then a distribution of likelihood is estimated by bootstrapping. Each iteration of bootstrapping proceeds as follows. A 64-dimensional (voxel-wise) histogram is produced from a bootstrap of the repeated simulation results. Then the likelihood of a single simulation result is estimated relative to the histogram. This is repeated many times to produce many combinations of single simulation results and histograms. This provides an estimate of how the likelihood of a single simulation result is characterized relative to the simulation generally.

For this, a scattering object was used as the input to a simulation 100 times. The bootstrap for each histogram comprised two thirds of the full sample, or the results of 66 simulations. Ten thousand bootstraps were used to construct the log-likelihood histogram. In Figure 28, the histogram for log-likelihood is illustrated.

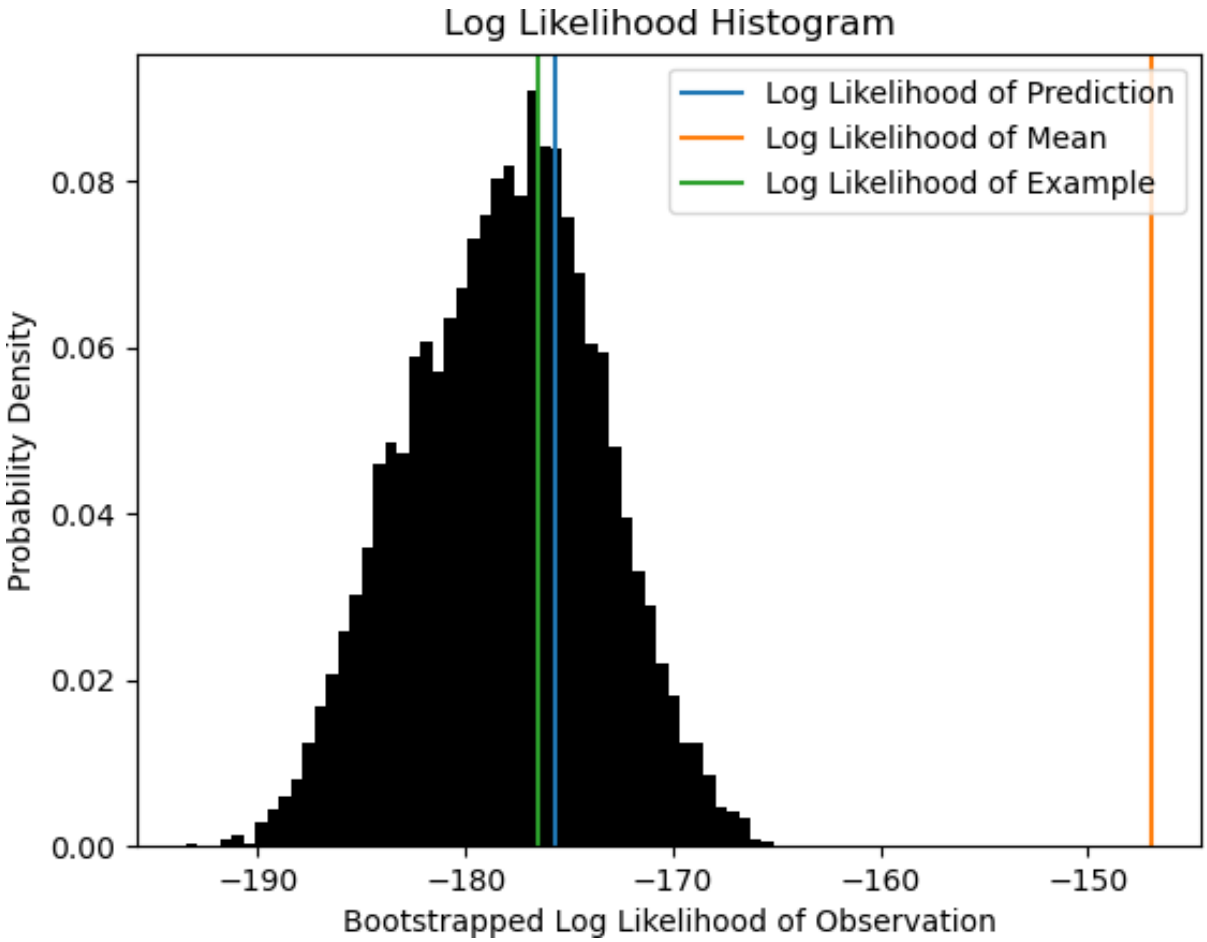


Figure 28: The distribution of log-likelihood for single simulation results relative to bootstrapped histogram. Though the mean of repeated simulation is more likely, it is uncharacteristically so. In contrast, the predictions from the surrogate model have a likelihood consistent with that of individual simulation results.

With this characterization of the simulation's likelihood, one can the compare the likelihood of the surrogate model's prediction to that of the bootstrapped distribution of likelihood. Moreover, the likelihood of the surrogate model's prediction can be used as a test statistic. The likelihood of the surrogate model's prediction is calculated relative to all of the repeated simulation results.

The mean of the log-likelihoods is -178.05. The log-likelihood of the surrogate model's prediction is -175.65. 32% of the observed log-likelihoods were greater than that of the surrogate model's prediction. The probability of the surrogate model's bin is 4.3%. The likelihood of the surrogate model's prediction is consistent with that of independent simulation results. This is contrasted with the (voxel-wise) mean of the simulation re-

sults. The mean of the simulation results has a likelihood, calculated against all of the repeated simulation results, of -146.90. All observed value of log-likelihood are less than this. The best estimate for probability of this log-likelihood is 0.01%. This is likely an over estimation of the probability.

It is interesting to compare these likelihood results to the coefficient of determination results for these cases. The mean coefficient of determination, relative to each repeated simulation result, is calculated for both the voxel-wise mean and the surrogate model's prediction. Similar to likelihood, the coefficient of determination of the voxel-wise mean is greater than that of the surrogate model's prediction (0.966 and 0.951 respectively). Furthermore, the mean coefficient of determination for each repeated simulation result, relative to every other repeated simulation result, was calculated. Taking the mean of means, the value is found to be 0.931.

In summary, though the (voxel-wise) mean of simulation results had a higher likelihood and coefficient of determination than the prediction of the surrogate model, it is uncharacteristic of the distribution of simulation results. Whereas, the prediction of the surrogate model was consistent with the distribution of repeated simulation results. Furthermore, the surrogate model's prediction had a higher likelihood and coefficient of determination that the average repeated simulation.

5.2.5 Feature Importance of Trained Random Forest Regressors

Feature importance is the relative predictive power of a given variable among decision trees in a random forest. All feature importance is found to concentrate upon the central voxel, for an amount of training data greater than 10. This indicates that variance among detector voxels beneath scattering object voxels of the same materials comes from variance in the radiation source.

5.2.5.1 Average Constituent Tree Depth Similar to the feature importance, for a given amount of training data the average depth of constituent trees of the random forest regressors converges to 3. It seems likely that there is a causal relationship between the average depth and one less the number of materials. One explanatory relationship would be that the average depth of constituent tree is the number of binary splits necessary to determine a material with a simple numeric encoding from N materials. Since feature

importance converges to a single scattering object voxel, it would be logical that average depth of decision tree shrank to span only the space of that voxel. In Figure 29, the convergence of average constituent tree is illustrated for $10,000$ data points. The points marked by red crosses (around the perimeter and in the centre) are those points included in the parameter sweep.

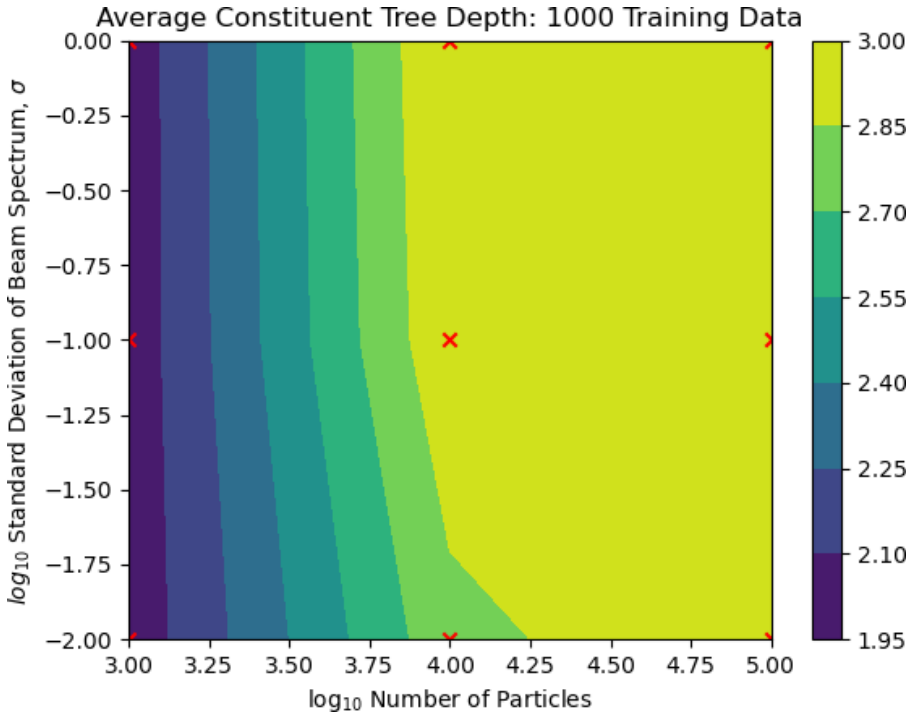


Figure 29: Sensitivity analysis of average depth of consistent regression trees. As the amount of data increases, the average depth of each constituent tree converges to 3.

5.2.5.2 How Much is There Left to Learn? The fact that the average depth of tree in the random forest converges to a depth of one less than the number of materials, (see Figure 29) suggests that the only predictive learning from training depends entirely on the central voxel. This is further demonstrated by the singular feature importance, and it also readily observed in Figure 26. This suggests the question, "is there any dependency on the surrounding voxels?". Having a machine learning model ignore the surrounding voxels could suggest against an existing dependency. However, this also may be caused by some failure of the model or training regime.

This can be further considered by comparing the distribution of repeated simulation results from a single voxel of a single "design". The results from a single voxel of a re-

peated simulation should be produced from a distribution that is distinct from that of all voxels of a common material. By applying a likelihood test, one could determine the probability that the two histograms are produced by distinct distributions. If they are, then the results illustrated in Figure 26 are insufficient to capture all of the impact that a design has upon the simulation results. In Figure 30, the results of two voxels are compared to the distribution of results for their corresponding material type.

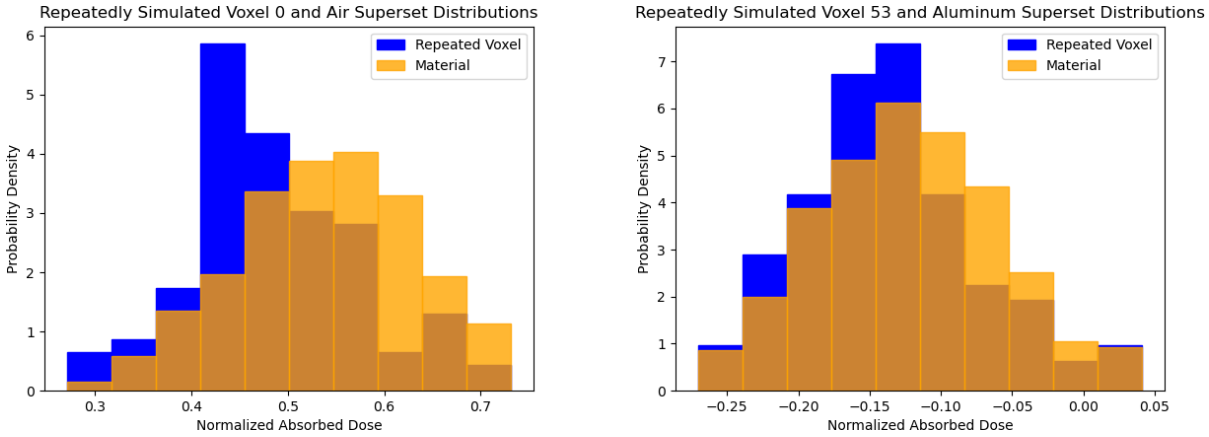


Figure 30: Comparison of repeated results for single voxels (within a single design) and those of the corresponding material type. The left figure illustrates the distribution for a voxel of air on the outer corner of the scattering object (in blue). This is superimposed upon the histogram of all inputs in the training set that have a scope-central voxel of air. The right figure illustrates that of a voxel of aluminum near the middle of the scattering object (in blue).

5.3 Design Optimization Comparison

To test the utility of using a surrogate model in a design optimization, a toy design was chosen. This ideal design is passed through an EGSnrc simulation so as to produce an 'ideal behaviour'. The ideal behaviour is the effect that the ideal design produced in the design environment. This is used to calculate a loss, measuring a candidate design's performance. Candidate designs are evaluated according to the effect they have on the design environment. The optimization seeks a design that produces the same effect on the design environment as the ideal design. A simulated annealing design optimization was written so to enable 'plugging in' either a surrogate model or the EGSnrc simulation to predict or simulate the effect of a candidate design on the design environment. The script for the simulated annealing design optimization are included in Annex 4 of the supplemental material to this work. The code for the optimizer is included in Annex 5 of the supplemental material to this work.

The design environment is an EGSnrc simulation of a square γ-ray beam with a spectral mean of 1 MeV and standard deviation of 0.1 MeV. 10000 particles are simulated for each call of the EGSnrc simulation. 100 data points are generated to train and test the surrogate model. The simulated annealing design optimization mutates the design at each iteration. Then the design is evaluated. For the 'brute-force' optimization using the EGSnrc simulation, an evaluation of the design calls a new EGSnrc simulation and processes the results. These results are compared to the behaviour of the ideal design. A loss function calculates the mean squared error, summing over all simulated voxels, to be the loss. When using a surrogate model to predict a candidate design's behaviour the loss is calculated as the mean squared error between predicted voxels (from the surrogate model) and ideal behaviour. Again this is summed over all voxels.

For a fair comparison the time required to generate training data and train the surrogate model were included in the comparison. The time required to produce data and train the surrogate model was two minutes and twenty-two seconds. For an optimization of 2000 iterations, the simulated annealing using the Random Forest Regressor surrogate model yielded a reasonable design in about 48 seconds. The results of the design optimization using the surrogate model are illustrated in Figure 31.

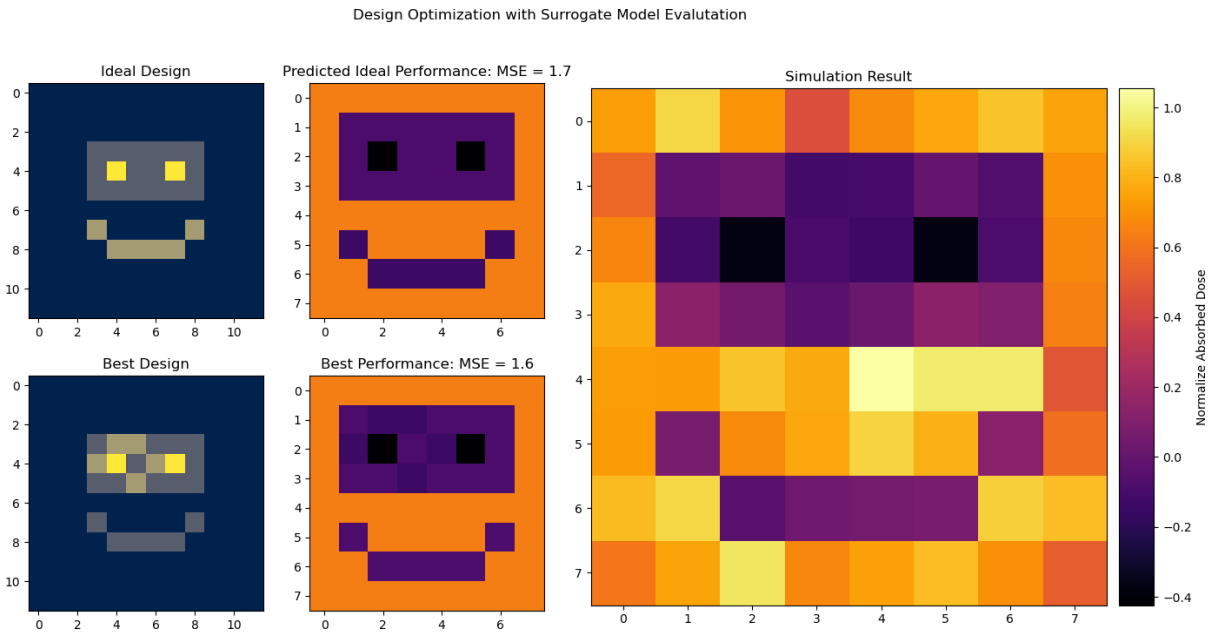


Figure 31: Design optimization results after 2000 iterations, using a surrogate model to evaluate candidate designs. The deviation in the design reflects variation in the design target itself.

Figure 31 is laid out as follows. In the far left column, there are two designs illustrated. The top sub-figure (top-left corner) illustrates the original toy design objective. Below, in the bottom-left corner, is the best performing design found during to course of the design optimization. In the central column are the predicted performance of the two previously illustrated designs. The top-central sub-figure is the surrogate model's prediction of how the ideal design would perform. The bottom-central sub-figure is the predicted performance of the optimal design (illustrated bottom-left). Last, occupying the right half of the Figure, is the optimization target. This illustrates the normalized absorbed dose of the ideal design calculated from an initial EGSnrc simulation. The central sub-figures and the right-most sub-figure both have their values quantified in the far right colour bar with an 'inferno' colour map [66].

Interestingly, the optimal design is predicted by the surrogate model to have a lower loss than the ideal design. This reflects the variance of the simulation results. The small variations in absorbed dose on detector voxels were included as a part of the optimization objective. Therefore, the optimal design had some scattering object voxels replaced with different material. This produces an expected performance more similar to the goal behaviour (the output of the EGSnrc simulation). A further limitation of the design

is illustrated in Figure 31. It is observed that the maximum values of normalized absorbed dose in both the predicted ideal performance and the predicted best performance are less than that of the simulation results. This is a consequence of how predictions are produced from the surrogate model. A decision tree categorizes the inputs into a reasonably homogeneous group (called a 'leaf'). Among all the constituent models of the random forest regressor, the predicted value is the mean of labels found in the leaf. Therefore, what is likely predicted by the random forest regressor is the mean value for detector voxels found below scattering object voxels of a particular material. The maximum observed dose will deviate from the mean values due to fluctuations in the source or randomly simulated scattering events. Therefore, the maximum predicted value is less than the maximum observed value.

To complete the comparison the same simulated annealing design optimization was executed with the design evaluation completed by simulating each candidate design in the same manner as the toy objective. Using the same computational platform as the surrogate model design optimization, each simulation and processing of EGSnrc results takes a few seconds. The full duration of the 'brute-force' design optimization was one hour and seventeen minutes. This is nearly 100$\times$ longer than the full time required to produce training data, train a surrogate model, and complete the design optimization using that surrogate model. This demonstrates the utility of leveraging machine learning surrogate models for inverse design. The results after 2000 iterations, the same number as the previous design optimization are illustrated in Figure 32.

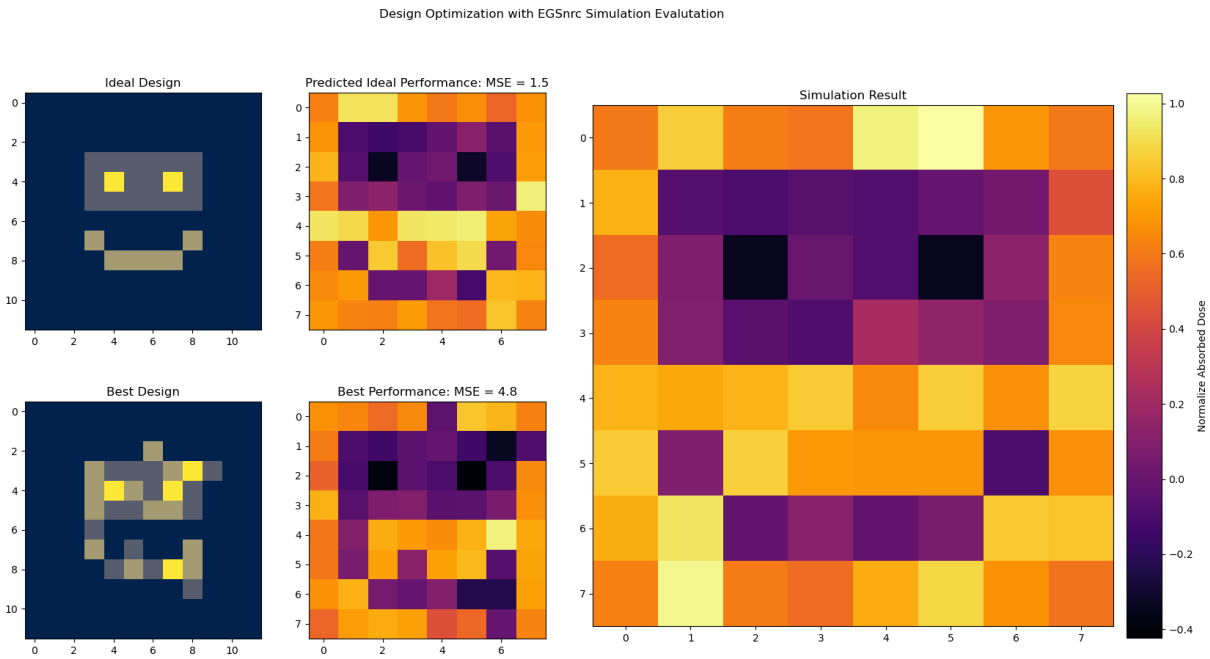


Figure 32: Design optimization results after 2000 iterations, using EGSnrc simulations to evaluate candidate designs. The variance in the simulation output has interfered with the design optimization.

It is observed that the same design optimization using the EGSnrc simulation for candidate design evaluation failed to converge upon an effective design. This is because the variance in the design environment produces a variable candidate design performance. This interferes with the design optimization process. It is still the case that the expectation performance for an optimal candidate design is greater than the expected performance of a non-optimal candidate design. Therefore, it is expected that this design optimization will eventually converge upon a reasonable candidate design. Indeed, in other design optimization experiments the 'brute-force' optimization did yield a reasonably effective design. Up to 5000 optimization iterations were required for a successful optimization.

5.4 PickAI

The second research project of this work is nicknamed "PickAI" (pronounced Pick-y). This is a NeuroEvolution algorithm that trains convolutional neural networks capable of generating distributions of designs. Convolutional neural networks (CNN) are evolved to produce samples of designs with a desirable distribution of properties or performance. The trained CNN take in random noise as an input, of some size, with the first dimension of input being equal to the desired sample size of output designs. One can imagine the evolved CNN mapping between different random distributions, but with more flexibility and complexity than a transformation or accept/reject method. The evolved CNN maps from an arbitrary random space into a targeted design space. The *distribution* of design properties or performance of the output sample is used to score the CNN that generated them. This score is called the network fitness score. The network fitness score is calculated based on the desired *distribution properties* of that latent space (e.g. distribution central moments or cumulants). The target of the optimization is not the design properties, but *how* those design properties are distributed over a generated sample.

5.4.1 Background: Ising Model

The Ising model is a simple many-particle system to illustrate the energy states associated with ferromagnetism. In a two-dimensional Ising system each index in an $N \times N$ grid are filled with either $+1$ or -1. These two states represent two possible states for a particle's spin. The internal energy of the system depends on the configuration of the spins. Neighbouring particles with aligned spins [67][68]. In this case we are using a Hamiltonian that applies the following sum over all nearest-neighbours in order to calculate the internal energy of the configuration. Periodic boundary conditions are applied.

$$\hat{H} = -J \sum_{i,j} \sigma_i \sigma_j \tag{161}$$

Where $\hat{H}$ is the Hamiltonian operator, σ_i and σ_j are the spins of nearest neighbours and J is the interaction strength.

5.4.2 Motivation

PickAI was created to facilitate the uniform data set sampling from non-uniform distributions. It is typically desirable for data sets to have a uniform distribution of properties [69]. There are some distributions from which using a transformation or acceptance-

rejection method are prohibitive. One such case, which we were trying to solve, was the distribution of energies for a two-dimensional Ising model. The highest energy configuration for a 6-by-6 Ising configuration has all constituent particles in their spin-up orientation. There is a single configuration of this sort. There are 2^{6^2} possible configurations in a 6-by-6 frame. The probability of finding a maximum energy configuration is 0.0000000015%. Uniformly sampling the configuration space to find the maximum energy would be prohibitively expensive. Therefore, an algorithm to produce artificially intelligent sampler, PickAI, was designed.

5.4.3 Implementation

5.4.3.1 Network Topology A convolutional neural network takes as an input one or more random configurations. Each input configuration has the desired shape of the output configuration. Input configurations have values on the range $(0, 1)$. A first layer of convolutions, with kernel size 3 and stride 1, expand the space up to 16 channels. The next layer further expands the number of channels to 32. Then following two layers symmetrically shrinking the number of layers. The output of the network is a single channel with size n by n, where n is the desired two-dimensional shape of an Ising configuration. Between each hidden layer there is hyperbolic Tan activation. At each layer of convolutions padding of zeros are also applied to the incoming channels. This is to keep the size of the input constant. The output of the network is passed into a function that ensures that the values are appropriate for an Ising configuration. That function is similar to a step function though with a minimum of -1, a maximum of 1 and the value at $x = 0$ being (arbitrarily) attributed to $f(x) = 1$. The network topology code is included in the Annex 6 of the supplemental material to this work.

5.4.3.2 NeuroEvolution Training A population of CNNs, each with the above described topology, are created. A batch of N_{in} random input configurations an input. From each network N_{in} Ising configurations are produced. The energy of each configuration is calculated by a Hamiltonian. Now, each network in the population has produced a sample of N_{in} energies. The first four sample central moments of each sample are calculated. These sample central moments are compared to analytically calculated population central moments of a discrete uniform distribution. The loss function is calculated as the sum of squared error of each estimated central moment. A discussion of using central moments as design objectives is included in the following subsection.

The loss is then simply used as "negative" fitness score. Networks are judged by amount that the sample central moments of their produced target distributions deviate from the analytically calculated values. The population of networks is ordered according to 'fitness'. The ten best performing networks are kept in the population and allowed to reproduce. The rest of the population is overwritten with mutated versions of the ten best. The implement reproduction scheme is as follows. 33% of the subsequent generation are mutations upon the best performing network. Another third of the subsequent generation are mutations upon the second, third and fourth best performing networks (i.e. the descendants of each account for a ninth of the subsequent population). The last third is split among the descendants of the fifth to tenth best performing networks (roughly one eighteenth each). Any residual space in the subsequent generation is iteratively given out, looping around the ten best, starting with the best performing network.

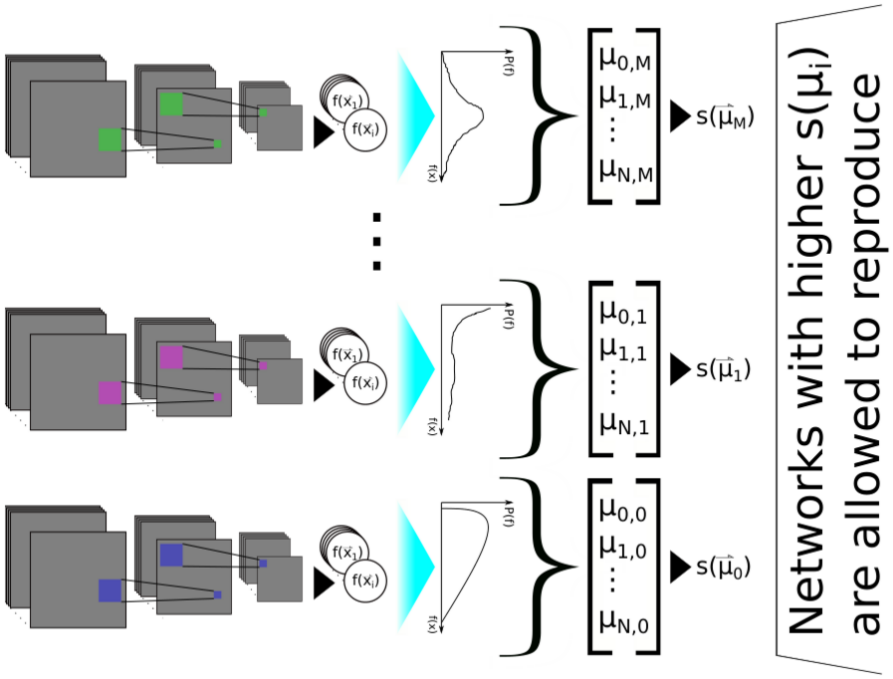


Figure 33: Visualization of PickAI sampler training by genetic algorithm. Distribution properties of derived quantities, calculated from the output of candidate neural networks are the target of optimization.

To create a 'descendant' of a successful network, a random number is added to each parameter of that network. The size of the random number shrinks over the course of the optimization. Conversely, the size of sample produced from each network increases

over the course of the optimization. This was inspired by simulated annealing. Initially, the optimization process allows for 'exploration'; evaluation is less stringent, and more networks are evaluated. Over course of the optimization, the evaluation of each network becomes more computationally expensive. Also, 'descendants' are created with smaller mutations. This is to 'narrow-in' on an optimal design. Code for the NeuroEvolution optimization is included in Annex 7 of the supplemental material to this work.

5.4.3.3 Objective Properties of Produced Samples

Since the goal of this optimization is to train artificially-intelligent 'samplers', therefore is must be properties of a distribution that are used for training. In this case, the first four central moments were selected. Central moments are useful and sufficient for characterizing a distribution [70][71]. Any sample that has sufficiently similar moments to a desired distribution will behave sufficiently similar for the purposes of design. Another means of scoring generated samples is using a measure of goodness-of-fit like likelihood. One could use the probability of a generated-sample being from the target distribution.

The analytical calculation of population central moments for a discrete uniform distribution require knowing the range of the distribution. For the sake of a general algorithm, the true range of the energies distribution is withheld during the optimization. The range of energies is initially guessed from the observed energy values from a random sample of the configuration space. From this observed range initial guesses for the analytical population central moments are calculated. As the optimization progresses, other values are observed. When an observed value (calculated from the network-produced configurations) exceeds the believed range of the energies-space, the range and theoretical population central moments are updated. This is something akin to 'accidental exploration' of the target space. It is likely however that the analytically calculated population central moments are (at least initially) incorrect. Ideally they reach the correct values at some point during the optimization. Despite this, there is a risk that the full range of the target space is not found. This is a necessary evil for the algorithm to begin with no prior knowledge of the target space.

5.4.4 Sampling of Ising Potentials

Using the above described topology and training scheme, an intelligent sampler was trained with the additional hyperparameters. The initial size of the generation was 500,

this linearly decreased over 1024 epochs to a minimum size of 25. Initially, each candidate produced and was evaluated on the minimum number of 100 qualifying configurations. Over the course of the training, the number of qualifying configurations increased by nearest powers of 2 to a maximum of 1024. The training was completed within six hours. In Figure 34, the histogram of Ising energies for a random sampler is superimposed onto that of the intelligent sampler.

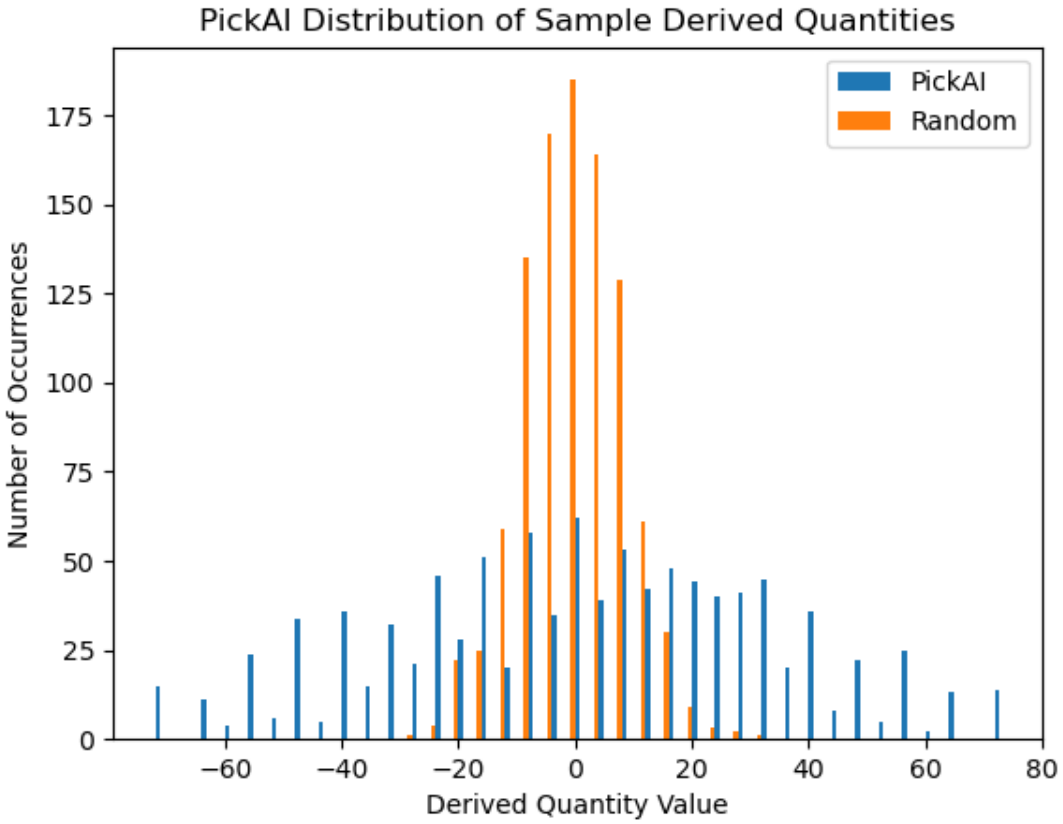


Figure 34: Comparison of the distributions of Ising energies, sampled either randomly, or from the outputs of the intelligent sampler. Though not uniform, the histogram produced by the intelligent sampler does yield results for the majority of possible values.

6 Conclusion

6.1 Summary

This work sought to investigate the benefits of using machine learning for the purposes of inverse design. Inverse design is a broad field of study comprising analytical, computational and heuristic means of producing effective designs. When introducing the topic, in Chapter 2: "Introduction to Inverse Design" this work outlined a framework for inverse design with three components. First the generation of designs, second the evaluation of designs, and lastly the planning of design construction. The first two of these components were considered in this work. Means for design generation can be either be gradient-based or gradient-free. A gradient-based method incrementally changes a design according to a some gradient in a differentiable objective function. Gradient-free methods randomly generate or modify designs, and propagate designs or design elements that provide net improvements according to an arbitrary objective function. Though gradient-based methods are considered to be more computationally effective, gradient-free methods allow for designs to optimize for non-differentiable objectives.

Designs generated by any method must be evaluated according to one or more performance metrics. Typically, performance metrics are calculated or combined by an objective function. To calculate a performance metric, one must observe, simulate or estimate how a candidate design will behave in a design environment. Many design optimizations evaluate candidate designs using simulation. This is a means to accelerate design processes by avoiding physically constructing a prototype. One might further accelerate the evaluation of designs by predicting the behaviour or performance of a candidate design using a *surrogate model*. The major research of this work demonstrated the benefits of using a machine learning surrogate model to substitute simulation of candidate designs. Chapter 2: "Introduction to Inverse Design" discusses the advantage and possible costs of using a machine learning surrogate model in inverse design, as well as model type selection, the potential of diminishing returns on accuracy, the role of environmental and design variance, and general considerations for inverse design. Chapter 3: "Overview of EGSnrc" provides a background of physics theory involved in the simulation, EGSnrc, which is used to produce a surrogate model. EGSnrc is a Monte Carlo simulation of radiation transport, capable of reproducing high fidelity Electron-γ showers. It seeks to solve the transport equation by tracking individual radiation particles as they traverse defin-

able geometry. To review the physics simulated in EGSnrc, sections in Chapter 3 roughly follow the following outline. First a physical phenomenon is introduced at a high level, using classical or simplified physics to impress the 'mechanics' of the phenomenon on the reader. From this intuitive description modern physical descriptions are introduced, providing as much context and theoretical continuity as possible. Lastly, the derivations of differential cross-sections for each physical phenomenon are included. These provide a common-interface to stochastic simulation. Additional improvements in EGSnrc are discussed and justified to a relevant extent. Curious or questioning readers are encouraged to review the extensive EGSnrc documentation and user manual, where there is a full description of the included physical phenomena and their implementations [4]. With the physical phenomena targeted for surrogate simulation described, Chapter 4: "Machine Learning Methods" introduces the fundamentals of machine learning theory. Machine learning concepts and applications are introduced, building a theoretical foundation for the analysis of random forests regressors. Random forest regressors are an ensemble method, that combine the outputs of many regression trees. Each constituent regression tree is trained on bootstrapped samples of training data. Furthermore, the constituent regression tress repeatedly sample from the dimensions of input data at each "branch" of the tree. These two attributes are known as "bagging" and a variation on the random subspace method. Once introduced, an analysis on the quantitative impact of bagging (and random forests) is conducted and a novel equation for expected improvements on a mean squared loss function is derived.

Having reviewed both the relevant physics and machine learning theory, Chapter 5: "Results" documents an experiment regarding machine learning surrogate models. Using EGSnrc to simulate radiation transport, our optimization sought to design a radiation mask. With radiation incident upon it, the mask would produce a desired radiation pattern on a detector behind it. It was found that a random forest regressor is capable of reproducing likely simulation results given a previously unseen radiation mask, and is a reasonable choice for a surrogate model. When substituting the surrogate model into a simulated annealing design optimization, it was found that the design optimization still converged upon effective designs. Using the surrogate model also decreased the overall time required for the design optimization to converge upon an effective design. There is even a time savings when one includes the time required to generate training data and train the surrogate model. This work then compared the design behaviour predicted

by the surrogate model to the output of the EGSnrc simulation. Unlike the simulation, the surrogate model produced a deterministic estimation. This is admittedly a deficit of the surrogate model. It may be rectified by using a different type of surrogate model, which includes some variance in its estimations. An unexpected benefit of the surrogate model's deterministic predictions was that the simulated annealing design optimization yielded more effective designs in fewer iterations. This suggests that of variance in the output of the EGSnrc simulation interfereswith the design optimization. Other potential impacts of using a surrogate model, both in method and consequence were discussed.

Lastly, the results from another experiment using machine learning for inverse design are included. A two-dimensional Ising spin model calculates the internal energy for $N \times N$ matrix configurations of either (-1, 1) values. Randomly sampling configurations yields a distribution of internal energies that is narrowly Gaussian. Maximum or minimum values of internal energy are very unlikely. This work finds that convolutional neural networks to capable of mapping between an input of random noise to a latent-space, which when binned, consists of two-dimensional Ising configurations. By training the neural network with NeuroEvolution, the distribution of internal energies of that latent space are shaped. The model fails to map to a latent space with a perfectly uniform distribution of internal energies. However, the distribution of internal energies of the latent space is significantly more uniform, and effectively facilitates uniform sampling using conventional sampling methods. This second experiment is an example of how machine learning methods may serve to generate designs, and design-spaces.

6.2 Future Work

6.2.1 Analysis of Bagging Ensemble Methods

The original expression here derived for the effect of bagging is novel to the extent discovered by the author. However, given that one of the key components in the derivation is included as an exercise in a textbook, it is suspected that similar analysis either exists or has been refuted. Additional research and review was beyond the scope of this work, and certainly beyond the scope of a chapter of review. Furthermore, a notion of stability, as was briefly seen when reviewing the works of Elisseeff, Evgeniou and Pontil [59], is missing from the present derivation. Now, "stability" might manifest in difference in model-data-covariance between ensemble and constituent models. If so this should be

comparable to other theoretical metrics of stability. Lastly, the derivation relied on a Mean-Squared Error loss function. It would be interesting to investigate if unified bias-variance decompositions such as those of Domingos [49] or James [57] might yield results for a generic loss function.

6.2.2 Surrogate Model Research

Designing "radiation masks" is a marginal application of this research. Though the surrogate model is capable of producing accurate and likely results, it is not an impressive feat of prediction. Works done by Casert, Tamblyn and Whitelam that demonstrate how "Transformer" neural networks are capable of reproducing emergent phenomena, are much more impressive. This is an important capability for surrogate models for inverse design, since the surrogate model would be able to extend the surrogate evaluation beyond a limited training environment or dataset. Such a capability would further improve the time-saving capabilities of surrogate models.

The author is also interested in further analysis, discussion and research regarding the design implications of using a deterministic surrogate model to replace a stochastic simulation. It is certainly possible, though not conducted here to produce a surrogate model that provides stochastic results. This could be accomplished, for example, by a stacking ensemble. Instead of taking the mean of the values (either in a leaf of a regression tree or of all the predictions of tress in a random forest regressor), a distribution would be fit to the values and sampled. It is expected that the time-savings of such a surrogate model would be comparable to those reported here. However, there is still the possibility of the variance among results to interfere with a design optimization.

For differentiable surrogate models, a neural network for example, it is possible by "freezing" the model to calculate a gradient on the input, relative to a desired output. Excitingly, optimization by this method blurs the lines of the inverse design framework presented in Chapter 2. This would allow a designer to optimize for any performance metric that can be predicted by such a surrogate model. Though as will next be discussed the constraints of some design spaces make gradient-based surrogates difficult to train.

6.2.3 PickAI

The results of PickAI, although useful when combined with conventional sampling methods, were not nearly uniform. Three possible solutions are here considered. First, it is possible that modifications to the distribution of input noise might produce more reliable results. Likely, by having an input of uniform noise the output would be more uniform. Second, there are many "features" that can be included in convolutional neural networks that were not included in the present network topology. These can serve to either increase or decrease the complexity or uniformity of the weight-space. For example, "drop-out" is a method to avoid overfitting during the training of neural networks. This entails leaving out randomly selected weights or kernels, batch-by-batch, during training. As a consequence, one can produce a sample of outputs from a single input. Therefore, more variation can be leveraged from a single model. Lastly, the objective function used during NeuroEvolution could be modified. There is a convenient statistical justification for using the moments of the distribution of properties, and this method is appropriate for any distribution with derived moments equations. However, there are other metrics that could be used or included in the objective function. Relevant to the two-dimensional Ising spin model, one could maximize the likelihood of the distribution of internal energy. More arbitrary objectives might also prove useful. For example maximizing how many distinct configurations produced by the candidate networks would encourage a diversity of outcomes. "PickAI" presently has no guarantee from sampling from a few configurations that happen to balance the distribution favourably.

Another potential improvement to "PickAI" would be to train the convolutional neural network by a gradient-based optimization method. This was attempted but proved difficult since both the Hamiltonian and the configuration constraints had discontinuities in their derivatives. There are methods such as "Soft-Max" activation functions that might prove helpful in this regard.

6.3 Outlook

Inverse design is a methodical and reliable approach to solving design problems. Design has long been an expensive, unreliable, and sometimes deadly process. Since the industrial revolution, the application of science and engineering to our constructed environment has produce increasingly effective machines and processes. The advent of digital computing and the internet is sometimes called the "information revolution". Un-

like previous "revolutions", ours is not limited by what can be imagined by contemporary designers. Recent advances in generative models can convert textual prompts into images, often with creative interpretations and artistic outputs. This demonstrates that modern artificial intelligence can range over spaces that are otherwise numerically inaccessible. By merely combining machine learning with conventional optimization, one can reliably reproduce inspiration concerning the most esoteric or technical matters. Skilled creativity is no longer preciously rare.

References

[1] Breiman, L. *Bagging Predictors*. Machine learning, **24**(2):123– (1996). ISSN 0885-6125. (Cited on pages 1, 64, 65, and 70.)

[2] Ho, T.K. *Random decision forests*. In *Proceedings of 3rd International Conference on Document Analysis and Reco gnition*, volume 1, pages 278–282 vol.1. IEEE (1995). ISBN 0818671289. (Cited on pages 1, 73, and 85.)

[3] Breiman, L. *Random Forests*. Machine learning, **45**(1):5–32 (2001). ISSN 0885-6125. (Cited on pages 1, 73, and 75.)

[4] Kawrakow, I. *et al. The EGSnrc Code System: Monte Carlo Simulation of Electron and Photon T ransport*. NRCC Report, **PIRS-701** (2020). (Cited on pages 1, 16, 21, 22, 24, 27, 28, 29, 33, 36, 37, 38, 41, 42, 44, and 106.)

[5] Ryczko, K. *et al. Inverse Design of a Graphene-Based Quantum Transducer via Neuroevolution*. The Journal of Physical Chemistry C, **124**(48):26117–26123 (2020). http://dx.doi.org/10.1021/acs.jpcc.0c06903. (Cited on pages 3, 5, and 10.)

[6] Victor Fung, J.Z. and Hu, G. *Inverse design of two-dimensional materials with invertible neural networks*. npj Computational Materials, **7**(200) (2021). (Cited on page 3.)

[7] *Center for Inverse Design Home Page*. [Online]. https://www.centerforinversedesign.org/. (Cited on page 3.)

[8] BMW. *Artificial intelligence gets artistic: the creative potential of AI | BMW.com*. [Online]. (Cited on page 3.)

[9] General Electric. *GE Integrating AI to Enable Performance-Informed Gas Turbine Inverse Design*. (Cited on page 3.)

[10] NVIDIA. *ArchiGAN: a Generative Stack for Apartment Building Design*. [Online] (2019). https://developer.nvidia.com/blog/archigan-generative-stack-apartment-building-design/. (Cited on pages 3 and 11.)

[11] Molesky, S. *et al. Inverse design in nanophotonics*. Nature photonics, **12**(11):659–670 (2018). ISSN 1749-4885. (Cited on pages 3 and 11.)

[12] Sullivan, L.H. *The Tall Office Building Artistically Considered*. Lippincott's Magazine, Philadelphia (1896). (Cited on page 4.)

[13] Loonen, R.C. *et al. 15 - Inverse design for advanced building envelope materials, systems and operation*. In E. Gasparri, A. Brambilla, G. Lobaccaro, F. Goia, A. Andaloro and A. Sangiorgio, editors, *Rethinking Building Skins*, Woodhead Publishing Series in Civil and Structural Engineering, pages 377–402. Woodhead Publishing (2022). ISBN 978-0-12-822477-9. http://dx.doi.org/https://doi.org/10.1016/B978-0-12-822477-9.00022-X. (Cited on pages 4 and 5.)

[14] Tarantola, A. *Inverse problem theory and methods for model parameter estimation*. Society for Industrial and Applied Mathematics SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104, Philadelphia, Pa (2005). ISBN 9780898717921. (Cited on page 4.)

[15] Wang, Q. and Zhang, L. *Inverse design of glass structure with deep graph neural networks*. Nature communications, **12**(1):5359–5359 (2021). ISSN 2041-1723. (Cited on page 5.)

[16] Hornby, G. *et al. Automated Antenna Design with Evolutionary Algorithms*. Space (2006). http://dx.doi.org/10.2514/6.2006-7242. (Cited on pages 6 and 10.)

[17] Sigmund, O. and Maute, K. *Topology optimization approaches: A comparative review*. Structural and multidisciplinary optimization, **48**(6):1031–1055 (2013). ISSN 1615-147X. (Cited on pages 7 and 11.)

[18] Ramunno, L. *Lecture 18: Stochastic Optimization*. University of Ottawa (2021). (Cited on pages 7 and 8.)

[19] Albeverio, S. *et al. A numerical implementation of quantum annealing*. In *Proceedings of the Ascona-Locarno Conference*, Stochastic Processes, Physics and Geometry. http://dx.doi.org/10.1142/9789814541107. (Cited on page 9.)

[20] Wu, C.Y. and Tseng, K.Y. *Topology optimization of structures using modified binary differential evolution*. Structural and multidisciplinary optimization, **42**(6):939–953 (2010). ISSN 1615-147X. (Cited on pages 9 and 12.)

[21] Fogel, D.B. and Stayton, L.C. *On the effectiveness of crossover in simulated evolutionary optimization*. BioSystems, **32**(3):171–182 (1994). ISSN 0303-2647. (Cited on page 10.)

[22] Shim, P.Y. and Manoochehri, S. *Generating OptimaL Configurations In Structural Design Using Simulated Annealing.* International journal for numerical methods in engineering, **40**(6):1053–1069 (1997). ISSN 0029-5981. (Cited on page 12.)

[23] Box, G.E.P. *Science and Statistics.* Journal of the American Statistical Association, **71**(356):791–799 (1976). ISSN 0162-1459. (Cited on page 13.)

[24] N.Gregory Mankiw et al. *Principles of Microeconomics.* Nelson Education, Boca Raton, FL, fourth canadian edition. edition (2008). (Cited on page 14.)

[25] Aldeghi, M. *et al. Golem: an algorithm for robust experiment and process optimization.* Chem. Sci., **12**:14792–14807 (2021). http://dx.doi.org/10.1039/D1SC01545A. (Cited on page 16.)

[26] Pedregosa, F. *et al. Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, **12**:2825–2830 (2011). (Cited on pages 16, 59, 60, 61, and 86.)

[27] Casert, C. *et al. Learning stochastic dynamics and predicting emergent behavior using transformers.* arXiv e-prints, arXiv:2202.08708 (2022). http://dx.doi.org/10.48550/ARXIV.2202.08708. (Cited on page 19.)

[28] Nelson, W. *et al. The EGS4 Code System.* Stanford Linear Accelerator Center, **SLAC-265** (1985). (Cited on pages 21, 24, 25, 37, and 44.)

[29] Lamarsh, J. and Baratta, A. *Introduction to Nuclear Engineering: Pearson New International Edition.* Pearson, 11 Community Centre, Panchsheel Park, New Delhi 110 017, India, 3 edition (2013). ISBN 978-93-325-3670-8. (Cited on pages 22, 25, 28, 39, and 41.)

[30] Cowan, G. *Statistical Data Analysis.* Oxford University Press, New York (2002). (Cited on pages 23, 48, 76, and 91.)

[31] Pedrotti, F.L. *Introduction to optics.* Pearson Prentice Hall, Upper Saddle River, NJ, 3rd ed. edition (2007). ISBN 0131499335. (Cited on pages 25 and 34.)

[32] Griffiths, D. *Introduction to Electrodynamics.* Pearson, Glenview, Illinois, fourth edition edition (2013). ISBN 9780321856562. (Cited on pages 26, 28, and 30.)

[33] Feynman *et al. The Feynman Lectures on Physics*, volume I: Mainly Mechanics, Radiation, and Heat, chapter 32: Radiation Damping, Light Scattering. Basic Books,

1290 Avenue of the Americas, New York, NY 10104 (1963). ISBN 9780465040858. https://www.feynmanlectures.caltech.edu/I_32.html. (Cited on page 27.)

[34] Hubbell, J.H. *et al. Atomic form factors, incoherent scattering functions, and photon scattering c ross sections.* Journal of physical and chemical reference data, **4**(3):471–538 (1975). ISSN 0047-2689. (Cited on page 27.)

[35] Nelms, A.T. and Oppenheim, I. *Data on the Atomic Form Factor: Computation and Survey.* Research of the National Bureau of Standards, **55**(1):53–62 (1955). (Cited on page 27.)

[36] Storm, L. and Israel, H.I. *Photon cross sections from 1 keV to 100 MeV for elements Z=1 to Z=100.* Atomic data and nuclear data tables, **7**(6):565–681 (1970). ISSN 0092-640X. (Cited on page 28.)

[37] Ribberfors, R. *Relationship of the relativistic Compton cross section to the momentum distribution of bound electron states.* Physical Review B, **12**(6):2067–2074 (1975). (Cited on pages 28, 31, and 35.)

[38] JabberWok. *Compton-scattering.svg.* [Creative Commons] (2006). https://commons.wikimedia.org/wiki/File:Compton-scattering.svg. This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/3.0/legalcode. (Cited on page 29.)

[39] Yazaki, Y. *How the Klein-Nishina formula was derived: Based on the Sangokan Nishina Source Materials.* Proceedings of the Japan Academy. Series B. Physical and biological sciences, **93**(6):399–421 (2017). ISSN 0386-2208. (Cited on page 30.)

[40] Ribberfors, R. and K.-F., B. *Incoherent-x-ray-scattering-functions and cross sections* $(d\sigma/d\Omega')_{incoh}$ *by means of a pocket calculator.* Physical Review A, **26**(6):3325–3333 (1982). (Cited on page 31.)

[41] Brusa, D. *et al. Fast sampling algorithm for the simulation of photon Compton scattering.* Nuclear instruments & methods in physics research. Section A, Accelerators, spectrometers, detectors and associated equipment, **379**(1):167–175 (1996). ISSN 0168-9002. (Cited on pages 32, 35, and 36.)

[42] Prasad, P. *Introduction to Biophotonics.* Wiley, Hoboken, New Jersey (2003). ISBN 9788126560981. (Cited on pages 39 and 40.)

[43] Д.Ильин. *File:Jablonski Diagram of Fluorescence Only-en.svg - Wikipedia*. [Public Domain]. https://commons.wikimedia.org/wiki/File:Jablonski_Diagram_of_Fluorescence_Only-en.svg. This work is licensed under the Creative Commons CC0 1.0 Universal Public Domain Declaration. To view a copy of this license, visit https://creativecommons.org/publicdomain/zero/1.0/legalcode. (Cited on pages 39 and 40.)

[44] Hastie, Trevor., T.R.F.J. *The elements of statistical learning : data mining, inference, and prediction*. Springer series in statistics. Springer, New York (2001). ISBN 9780387216065. (Cited on pages 46, 47, 49, 50, 52, 53, 55, 56, 58, 60, 62, 65, 86, 88, and 90.)

[45] Chabacano. *diagram showing overfitting of a classifier*. [Creative Commons] (2008). https://commons.wikimedia.org/wiki/File:Overfitting.svg. This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/3.0/legalcode. (Cited on page 50.)

[46] Bousquet, O. and Elisseeff, A. *Stability and Generalization*. J. Mach. Learn. Res., **2**:499–526 (2002). http://jmlr.org/papers/v2/bousquet02a.html. (Cited on pages 52, 65, and 70.)

[47] Breiman, L., C.A. *Random Forests*. [Online]. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. (Cited on page 53.)

[48] *PyTorch 1.12 documentation*. [Online]. https://pytorch.org/docs/stable/index.html. (Cited on page 53.)

[49] Domingos, P. *A unified bias-variance decomposition*. In *Proceedings of 17th international conference on machine learning*, pages 231–238. Morgan Kaufmann Stanford (2000). (Cited on pages 57, 58, 63, 67, and 108.)

[50] Geman, S. *et al. Neural Networks and the Bias/Variance Dilemma*. Neural Computation, **4**(1):1–58 (1992). ISSN 0899-7667. http://dx.doi.org/10.1162/neco.1992.4.1.1. (Cited on pages 57 and 67.)

[51] Gey, S. and Poggi, J.M. *Boosting and instability for regression trees*. Computational statistics & data analysis, **50**(2):533–550 (2006). ISSN 0167-9473. (Cited on page 58.)

[52] Breiman, L. *Classification and Regression Trees*. CRC Press, Boca Raton, FL, first edition. edition (2017). ISBN 9781315139470. (Cited on pages 59, 60, 63, and 66.)

[53] Zhou, Z.H. *Ensemble methods : foundations and algorithms*. Chapman & Hall/CRC machine learning & pattern recognition series. CRC Press, Boca Raton, Fla (2012). ISBN 9780429151095. (Cited on page 63.)

[54] Wolpert, D. and Macready, W. *Combining Stacking With Bagging To Improve A Learning Algorithm* (1996). (Cited on pages 64, 65, 67, and 68.)

[55] Sirakorn. *English: Illustration of a bootstrap aggregating (bagging) method for ensemble learning.* [Creative Commons] (2020). https://commons.wikimedia.org/wiki/File:Ensemble_Bagging.svg. This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/4.0/legalcode. (Cited on page 65.)

[56] Wolpert, D.H. *On Bias Plus Variance*. Neural computation, **9**(6):1211–1243 (1997). ISSN 0899-7667. (Cited on pages 67 and 78.)

[57] James, G.M. *Variance and Bias for General Loss Functions*. Machine learning, **51**(2):115–135 (2003). ISSN 0885-6125. (Cited on pages 67 and 108.)

[58] Ho, T.K. *The Random Subspace Method for Constructing Decision Forests*. IEEE Trans. Pattern Anal. Mach. Intell., **20**(8):832–844 (1998). http://dx.doi.org/10.1109/34.709601. (Cited on page 70.)

[59] Elisseeff, A. *et al. Stability of Randomized Learning Algorithms*. J. Mach. Learn. Res., **6**:55–79 (2005). (Cited on pages 70, 71, 72, and 107.)

[60] Grandvalet, Y. *Stability of Bagged Decision Trees*. In *Proceedings of the XLIII Scientific Meeting of the Italian Statistical S ociety*, pages 221–230. CLEUP (2006). (Cited on page 73.)

[61] Ho, T.K. *A Data Complexity Analysis of Comparative Advantages of Decision Forest Constructors*. Pattern analysis and applications : PAA, **5**(2):102–112 (2002). ISSN 1433-7541. (Cited on page 75.)

[62] Skurichina, M. and Duin, R.P.W. *Bagging, Boosting and the Random Subspace Method for Linear Classifiers*. Pattern analysis and applications : PAA, **5**(2):121–135 (2002). ISSN 1433-7541. (Cited on page 75.)

[63] Hastie, T. *The Elements of Statistical Learning Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer New York, New York, NY, 2nd ed. 2009. edition (2009). ISBN 9780387848587. (Cited on page 76.)

[64] Tange, O. *GNU Parallel 20210422 ('Ever Given')* (2021). http://dx.doi.org/10.5281/zenodo.4710607. GNU Parallel is a general parallelizer to run multiple serial command line programs in parallel without changing them. (Cited on page 84.)

[65] Ryczko, K. *et al.* *Machine Learning Diffusion Monte Carlo Energies*. arXiv e-prints, arXiv:2205.04547 (2022). http://dx.doi.org/10.48550/ARXIV.2205.04547. (Cited on page 85.)

[66] Hunter, J.D. *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, **9**(3):90–95 (2007). http://dx.doi.org/10.1109/MCSE.2007.55. (Cited on page 97.)

[67] Ramunno, L. *Lecture 16: Monte Carlo simulations in thermostatistics*, chapter Computational Physics II. University of Ottawa (2021). (Cited on page 100.)

[68] Mills, K. and Tamblyn, I. *Deep neural networks for direct, featureless learning through observation: The case of two-dimensional spin models*. Phys. Rev. E, **97**:032119 (2018). http://dx.doi.org/10.1103/PhysRevE.97.032119. (Cited on page 100.)

[69] Sharma Sharma, Colin Bellinger, B.O. and Nathalie Japkowicz. *Synthetic Oversampling with the Majority Class: A New Perspective on Handling Extreme Imbalance*. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 447–456. IEEE (2018). ISBN 1538691590. ISSN 2374-8486. (Cited on page 100.)

[70] Johnson, N.L. and Kotz, S. *Use of Moments in Deriving Distributions and some Characterizations*. Mathematical Scientist, **15**:42–52 (1990). (Cited on page 103.)

[71] Johnson, N.L. and Kotz, S. *Randomly weighted averages: Some aspects and extensions*. The American Statistician, **44**(3):245–249. (Cited on page 103.)