Architectural Artificial Intelligence:

Exploring and Developing Strategies, Tools, and Pedagogies Toward the Integration of Deep Learning in the Architectural Profession

Nariddh Khean

A thesis in fulfilment of the requirements for the degree of

Master of Philosophy

Faculty of Built Environment

September 2019

Thesis/Dissertation Sheet

Surname/Family Name	:	**Khean**
Given Name/s	:	**Nariddh**
Abbreviation for degree as give in the University calendar	:	**MPhil**
Faculty	:	**Built Environment**
School	:	
Thesis Title	:	**Architectural artificial intelligence: Exploring and developing strategies, tools, and pedagogies toward the integration of deep learning in the architectural profession**

Abstract 350 words maximum: (PLEASE TYPE)

The growing incessance for data collection is a trend born from the basic promise of data: "save everything you can, and someday you'll be able to figure out some use for it all" (Schneier 2016, p. 40). However, this has manifested as a plague of information overload, where "it would simply be impossible for humans to deal with all of this data" (Davenport 2014, p. 151). Especially within the field of architecture, where designers are tasked with leveraging all available sources of information to compose an informed solution. Too often, "the average designer scans whatever information [they] happen on, […] and introduces this randomly selected information into forms otherwise dreamt up in the artist's studio of mind" (Alexander 1964, p. 4). As data accumulates—less so the "oil", and more the "exhaust of the information age" (Schneier 2016, p. 20)—we are rapidly approaching a point where even the programmers enlisted to automate are inadequate.

Yet, as the size of data warehouses increases, so too does the available computational power and the invention of clever algorithms to negotiate it. Deep learning is an exemplar. A subset of artificial intelligence, deep learning is a collection of algorithms inspired by the brain, capable of automated self-improvement, or "learning", through observations of large quantities of data. In recent years, the rise in computational power and the access to these immense databases have fostered the proliferation of deep learning to almost all fields of endeavour. The application of deep learning in architecture not only has the potential to resolve the issue of rising complexity but introduce a plethora of new tools at the architect's disposal, such as computer vision, natural language processing, and recommendation systems. Already, we are starting to see its impact on the field of architecture. Which raises the following questions: what is the current state of deep learning adoption in architecture, how can one better facilitate its integration, and what are the implications for doing so? This research aims to answer those questions through an exploration of strategies, tools, and pedagogies for the integration of deep learning in the architectural profession.

FOR OFFICE USE ONLY Date of completion of requirements for Award:

ORIGINALITY STATEMENT

‘I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.’

Signed ……

Date 2021.07.20 ……

COPYRIGHT STATEMENT

Signed ……

Date 2021.07.20 ……

AUTHENTICITY STATEMENT

Signed ……

Date 2021.07.20 ……

INCLUSION OF PUBLICATIONS STATEMENT

UNSW is supportive of candidates publishing their research results during their candidature as detailed in the UNSW Thesis Examination Procedure.

Publications can be used in their thesis in lieu of a Chapter if:

- The candidate contributed greater than 50% of the content in the publication and is the "primary author", ie. the candidate was responsible primarily for the planning, execution and preparation of the work for publication
- The candidate has approval to include the publication in their thesis in lieu of a Chapter from their supervisor and Postgraduate Coordinator.
- The publication is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in the thesis

Please indicate whether this thesis contains published material or not:

☐ This thesis contains no publications, either published or submitted for publication
(if this box is checked, you may delete all the material on page 2)

X Some of the work described in this thesis has been published and it has been documented in the relevant Chapters with acknowledgement
(if this box is checked, you may delete all the material on page 2)

☐ This thesis has publications (either published or submitted for publication) incorporated into it in lieu of a chapter and the details are presented below

CANDIDATE'S DECLARATION

I declare that:

- I have complied with the UNSW Thesis Examination Procedure
- where I have used a publication in lieu of a Chapter, the listed publication(s) below meet(s) the requirements to be included in the thesis.

Candidate's Name	**Signature**	**Date (dd/mm/yy)**
Nariddh Khean		**02/08/2021**

Acknowledgements

This research was supported by the Australian Government Research Training Program Scholarship.

I would like to express my utmost gratitude to Alessandra Fabbri and Matthias Hank Haeusler. Even before this degree, you have been so incredibly generous with your time, always willing to go above and beyond with your duties. From praise to criticism, all of the comments you've provided have not only affected the words I would write, but also the person that I am. I would consider these past few years as my most formative, and you two have massively shaped who I am and how I see the world. For that, and everything else, thank you.

And, thank you to my parents, Lychheng and Nounara Khean, for your faith in education.

Table of Contents

List of Figures and Tables

Page

List of Publications, Presentations, and Workshops

2018 March

UNIVERSITY WORKSHOP

"How Machines Learn"

Conducted at the University of New South Wales, Australia.

2018 September

CONFERENCE PAPER AND PRESENTATION

Khean, N, Fabbri, A & Haeusler, MH 2018, 'Learning Machine Learning as an Architect, How to? Presenting and Evaluating a Grasshopper-Based Platform to Teach Architecture Students Machine Learning', *Proceedings of the 36th International eCAADe Conference*, vol. 1, pp. 95-102.

Presented at the Łódź University of Technology, Poland.

2018 April

CONFERENCE WORKSHOP

"Deep Reinforcement Learning in Grasshopper: Using Deep Q-Networks to Train an Intelligent Agent to Act in a Grasshopper Environment"

Conducted at the Victoria University of Wellington, New Zealand.

2018 April

INDUSTRY PRESENTATION

"5x5 Series: Research. Machine Learning in the Built Environment"

Presented at Arup Sydney, Australia.

2019 June

UNIVERSITY WORKSHOP

"Deep Reinforcement Learning in Grasshopper: Using Deep Q-Networks to Train an Intelligent Agent to Act in a Grasshopper Environment"

Conducted at the University of New South Wales, Australia.

2019 June

CONFERENCE PAPER AND PRESENTATION

Khean, N, Fabbri, A, Gerber, D & Haeusler, MH 2019, 'Examining Potential Socio-economic Factors that Affect Machine Learning Research in the AEC Industry', *Proceedings of the 18th International CAAD Futures Conference*, vol. 1, pp. 247-263.

Presented at Korea Advanced Institute of Science and Technology, South Korea.

2019 September

CONFERENCE WORKSHOP

"Deep Reinforcement Learning in Grasshopper: Discovering Emergent Behaviour with Q-Learning"

Conducted at the University of Porto, Portugal.

Abstract

The growing incessance for data collection is a trend born from the basic promise of data: “save everything you can, and someday you’ll be able to figure out some use for it all” (Schneier 2016, p. 40). However, this has manifested as a plague of information overload, where “it would simply be impossible for humans to deal with all of this data” (Davenport 2014, p. 151). Especially within the field of architecture, where designers are tasked with leveraging all available sources of information to compose an informed solution. Too often, “the average designer scans whatever information [they] happen on, […] and introduces this randomly selected information into forms otherwise dreamt up in the artist’s studio of mind” (Alexander 1964, p. 4). As data accumulates—less so the “oil”, and more the “exhaust of the information age” (Schneier 2016, p. 20)—we are rapidly approaching a point where even the programmers enlisted to automate are inadequate.

Yet, as the size of data warehouses increases, so too does the available computational power and the invention of clever algorithms to negotiate it. Deep learning is an exemplar. A subset of artificial intelligence, deep learning is a collection of algorithms inspired by the brain, capable of automated self-improvement, or “learning”, through observations of large quantities of data. In recent years, the rise in computational power and the access to these immense databases have fostered the proliferation of deep learning to almost all fields of endeavour. The application of deep learning in architecture not only has the potential to resolve the issue of rising complexity, but introduce a plethora of new tools at the architect’s disposal, such as computer vision, natural language processing, and recommendation systems. Already, we are starting to see its impact on the field of architecture. Which raises the following questions: what is the current state of deep learning adoption in architecture, how can one better facilitate its integration, and what are the implications for doing so? This research aims to answer those questions through an exploration of strategies, tools, and pedagogies for the integration of deep learning in the architectural profession.

Chapter 1.

Introduction

The dream of artificial intelligence (AI) has been in the minds of technologists, inventors, and authors well before the advent of the computer. One of its earliest known manifestations can be traced back to the 8th century BCE, where Homer depicted Hephaestus' mechanical assistants as having "intellect and [...] skill[s] in subtle crafts" (2009, p. 385). However, this early idea of artificial intelligence came to be better understood as "automata": autonomous, mechanised humanoids. In the 17th century, two philosophers, René Descartes and Thomas Hobbes, continued to regard the human as the archetype for intelligence, and contemplated the general premise behind artificial intelligence. This premise—the assumption that thought can be mechanised and expressed mathematically—lead to the study of computational logic in the 20th century, which substantiated the elusive embodiment of intelligence. The programmable computer further splintered the field, forming two schools of thought that both believed they possessed the algorithms for true artificial intelligence: connectionist and symbolic AI. Ultimately, artificial intelligence is not simply a technology, but an ideal that evolves through time. The modern incarnation of artificial intelligence, deep learning, is a set of algorithms inspired by the brain. Stemming from the notions of connectionist AI, deep learning has come to achieve beyond-human level performance on a plethora of narrow tasks. Most notable is its ability to extract insight from vast quantities of data, a valuable capability in the age of the peta-, exa-, and zettabyte.

Within the field of architecture, there is a demand for architects and designers to leverage all available information to resolve problems of growing complexity. However, in lieu of the ability to glean meaning from vast quantities of data, in tandem with the decreasing effectiveness of the programmer, deep learning is a technology that could prove effective in taming the scourge of

quantity. Thus, the overarching objective of this research is to promote, explore, and develop for a more intelligent field of architecture, through the adoption of deep learning. By doing so, architects and designers, equipped with a proficiency with deep learning, possess a toolset that would yield a more holistic perspective on design problems that are more faceted, intricate, and complex than ever.

The following body of research can be segmented into three sections. The first is a series of studies to capture the current state of deep learning integration within the architecture, engineering, and construction industry. This will explore the extent of investment in deep learning research, the socio-economic barriers to further integration, and the current tools that facilitates its use. The insights gathered will drive the next section, an applied exploration of educational software tools and resource-centred pedagogies. This will include the development of deep learning software tools, as well as how architectural practitioners, educators, and students respond to learning deep learning. Finally, armed with the knowledge and experience of the previous sections, this research will conclude with an inquiry into the implications for the application of deep learning in the field of architecture.

Chapter 2.

Background

As early as the Neolithic Revolution, where previously nomadic tribes congregated to adopt agriculture and geographic permanence, humans have pursued the act of invention. Studying one of the best-preserved Neolithic settlements, the Çatalhöyük archaeological site in Turkey, one of the earliest forms of architectural construction was realised through mud-bricks. Subsequently, the invention of load-bearing arches, Vitruvius' Classical Orders, and computer-aided architectural design (CAAD), have all shaped the evolution of the built environment. Despite the uncertainty behind the technology that will drive the architecture of the future, there is a clarity in the persistence of human invention and its dramatic affect toward the advancement of the field.

A common aphorism states that "necessity is the mother of invention". And for the most part, this is reflected in the development of architecture. Taking the aforementioned examples, the arch was designed to better offset downward forces, Vitruvius' Classical Orders were developed to define column types and entablature designs, and the advent of CAAD sought to replace the tedium of hand-drawing. However, historian and author of *Guns, Germs, and Steel: A Short History of Everybody for the Last 13,000 years*, Jared Diamond, comes to the conclusion that technology "finds most of its uses after it has been invented, rather than being invented to meet a foreseen need" (2017, p. 235). If we set aside the latter half of his assertion, which directly contradicts the original aphorism, one could argue that these two statements are not mutually exclusive, and, in tandem, say something quite profound about the value of an invention. Necessity may act as a catalyst for invention; however, the potentiality of the invention is revealed only after it's ubiquity.

Let's take, for instance, computer-aided architectural design. CAAD was initially intended as the next step in hand-drawing. The earliest CAAD software was ostensibly a glorified pencil and

paper. Yet, parametric design, a modern application of CAAD, has allowed designers to generate forms that go beyond preconceived notions of form from primitives—toward forms that are dictated more by mathematics and logic. Parametric design, along with structural analysis, rendering and visualisation, building information modelling, amongst many more applications, are the result of wide-spread use of an invention that was originally aimed at solving a completely different problem. From this, a trend emerges: the more general an invention, the broader its potential applicability, thus the greater its value.

As a precursor to Erik Brynjolfsson and Andrew McAfee's *The Second Machine Age*, Brynjolfsson presented a talk that demonstrated the importance of general-purpose technology. He places these current and upcoming technologies within the context of a new machine age; one which allocates more value on "knowledge creation than just physical production" (Brynjolfsson 2013). One of the three words used to characterise the new machine age was "combinatorial". Reinforcing the importance of generality, Brynjolfsson explains this characterisation through a comparison between the "stagnationist view", where "ideas get used up like low-hanging fruit", against the reality, where "each innovation creates building blocks for more innovations" (2013). Thus, although necessity may be the mother of invention, the invention with the greatest value are those that have a broad spectrum of applicability.

2.1. The Last Human Invention

Glancing back at the broader history of invention, we can observe an overarching trend. Philosopher and author of *Superintelligence: Paths, Dangers, Strategies*, Nick Bostrom, observes that "history seems to exhibit a sequence of distinct growth modes, each much more rapid than its predecessor" (2014, p. 1). These "modes" are induced by specific technological shifts, caused by an invention that accelerates productivity beyond the invention's initial objective; the general-purpose technology. Steam engines, electricity, and transistors are such exemplars—so much so

that they are attributed as the defining technologies for the first, second, and third industrial revolutions, respectively.

Many believe that we are on the cusp of yet another industrial revolution, this time, propelled by artificial intelligence (AI). In *The Second Machine Age*, Brynjolfsson and McAfee suggests that the next growth mode "will be characterized by countless instances of machine intelligence and billions of interconnected brains" (2014, p. 96), and goes so far as to proclaim that "building intelligent machines [is] perhaps the most important invention in human history (Brynjolfsson, 2013). In conjunction, executive chairman of the World Economic Forum and author of *The Fourth Industrial Revolution*, Klaus Schwab, postulates a revolution that is "characterised by a much more ubiquitous and mobile internet, by smaller and more powerful sensors, and by artificial intelligence" (2017, p. 7).

Modern AI has yet to reach a level of ubiquity beyond the narrow applications it was originally intended to solve, thus, it is difficult to predict its potentiality. But as with the most impactful inventions, the true benefit of AI will be seen after wide-spread use. Some see AI incredibly favourably, suggesting that "the transformations [...] will be profoundly beneficial" (Brynjolfsson & McAfee 2014, p. 9). However, other's beliefs align more with mathematician, I. J. Good's often-quoted prediction about intelligent machines:

> "Let an ultraintelligent machine be defined as a machine that can far surpass all the intellectual activities of any man however clever. Since the design of machines is one of the intellectual activities, an ultraintelligent machine could design even better machines; there would then unquestionably be an "intelligence explosion," and the intelligence of man would be left far behind. Thus, the first ultraintelligent machine is the last invention that man need ever make..."
>
> (Good 1965, p. 33)

However, as it stands, there is a difference between the AI that is currently being used to fuel business decisions and defeat world champions in *Jeopardy!*, compared with that of the superintelligence feared since the dawn of computing. This isn't helped when the very definition of "intelligence" is one that has transformed even before the invention of the computer. As Bostrom cleverly points out, the "advent of such machines was often placed some twenty years into the future. Since then, the expected arrival date has been receding at a rate of one year per year; so that today, futurists [...] often believe that intelligent machines are a couple of decades away" (2014, p. 3-4).

2.2. Reflecting on Intelligence

The field of artificial intelligence has accelerated in recent years, and as a consequence, a rigorous and widely accepted ontology has struggled to keep pace. However, irrespective of the field's momentum, the difficulty of characterising and delineating breakthroughs stems from our poorly assembled and ever-changing ideals of "intelligence". To form an understanding of the current state of AI research, it is beneficial to revisit our earliest efforts in the field: our attempts to invent artificial intelligence.

Founding member of the Association for the Advancement of Artificial Intelligence (AAAI), Bruce G. Buchanan, suggests that "the beginnings of artificial intelligence [can be] traced to philosophy, fictions, and imagination" (Buchanan 2005, p. 53). The prevailing theory places the invention of something comparable to artificial intelligence in Ancient Greece. Homer's 8th century BCE poem, *The Iliad*, depicts Hephaestus, the god of fire, metallurgy, and craftsmen, building golden handmaidens to assist him in his forge.

> "Grasping a thick staff he limped from the forge, supported by servants made of gold, fashioned like living girls, who attended swiftly on their master. As well as the use of their limbs they had intellect, and the immortals gave them skill in subtle crafts."
>
> (Homer 2009, p. 385)

Greek literature is littered with fanciful stories of machines exhibiting intelligence, such as Talos, the colossus tasked with defending Crete, and Daedalus' animated bronze sculptures. Throughout, these stories share a thread of commonality, such that, "humanoid machines were mostly conceived as representing straightforward hope—the ideal servant who always obeys, the perfect soldier who never tires" (Cave & Dihal 2018, p. 474). These ideals painted artificial intelligence as physically embodied, often humanoid, and, despite being imbued with the divine, ultimately mechanical.

For centuries, many would also perceive intelligence as a mechanisable phenomenon. In the 1st century CE, Hero of Alexandria developed a series of "intelligent" inventions, from coin-operated apparatus to an automated puppet theatre. In the 9th century, the Banū Mūsā brothers created a mechanical flute player driven by steam, among a host of other machines showcased in their *Book of Ingenious Devices*. In the 13th century, polymath, Ismail al-Jazari, wrote their *Book of Knowledge of Ingenious Mechanical Devices*, which included designs for a programmable musical quartet and a drink-serving waitress.

But, can we really consider these inventions as "artificial intelligence"?

As time yielded greater comprehension for the nuances of these inventions, many grew uncomfortable calling them "intelligent". Eventually, they adopted Homer's original, and more accurate, classification of "automata": a mechanical device, embodying the intelligence of its human creator, to autonomously act upon the world. And although these inventions may have

seemed intelligent in their respective contexts, a broader understanding has shifted what it means to be intelligent.

It was not until the 17th century that, informed by the views of Aristotle, Euclid, and al-Khwarizmi, philosophers provided a new notion of intelligence. René Descartes, a French philosopher and mathematician, further rejected the notion that the automaton was intelligent. Expounded in Descartes' philosophical treatise published in 1637, *Discourse on the Method of Rightly Conducting one's Reason and of Seeking Truth in the Sciences*, Descartes separates man from machine by highlighting two fallacies of the automata:

1. their inability to "produce different sequences of words so as to give an appropriately meaningful answer to whatever is said in its presence—which is something that the dullest of men can do," and
2. "even though such machines might do some things as well as we do them, or perhaps even better, they would be bound to fail in others; and that would show us that they weren't acting through understanding but only from the disposition of their organs" (Descartes 2007, pp. 22).

In other words, the two failings of automata were their conversational impotence and an ineptitude for generalisation. Extrapolating further, Descartes suggests that "these two factors also tell us how men differ from beasts" (Descartes 2007, pp. 22). In his view, humans were the only beings that possessed the ability to reason. Thus, with humans as the archetype of intelligence, AI was formalised as "a discipline that aims to understand the nature of human intelligence" (Nath 2009, p. 34). Subsequently, two broad philosophies grew from Descartes' supposition, which attempted to rationalise human cognition: the computational and non-computational theory of mind.

A contemporary of Descartes, Thomas Hobbes, is widely considered the earliest proponent for the computational theory of mind. As outlined in his 1655 text, *De Corpore*, Hobbes asserts human reasoning as a mathematical process. "By ratiocination, I mean computation" (Hobbes 1969, p. 3). Hobbes' assertion alludes to the fundamental idea of the computational theory of mind, whereby thought can be distilled to a series of symbolic operations. "To compute, is either to collect the sum of many things that are added together, or to know what remains when one thing is taken out of another... All ratiocination is comprehended in these two operations of the mind, addition and subtraction" (Hobbes 1969, p. 3). Hobbes substantiates his views by illustrating that "of the several conceptions of fours sides, equality of sides, and right angles, is compounded the conception of a square" (Hobbes 1969, p. 4). He further describes propositions and syllogisms, other mental activities, as simple additions.

Many believe Hobbes' *De Corpore* to be a critical moment in AI history, some going so far as to proclaim Hobbes "prophetically launching artificial intelligence" (Haugeland 1985, p. 23). And, as we will discuss later, this methodical, rule-based conception on intelligence was quite pivotal, forming one of the two main paradigms in computational AI. But more about that later.

Returning to Descartes, his famous proposition "cogito, ergo sum" (I think, therefore I am) is suggestive of the non-computational theory of mind. The concept of 'I think', an inherently subjective activity, defies the fabric of computation. Author of *Philosophy of Artificial Intelligence: A Critique of the Mechanistic Theory of Mind*, Rajakishore Nath, makes this argument, suggesting that "the mental processes, for Descartes, are intentional and are free acts of the thinking subject. Hence, they cannot be mapped mechanically in an algorithmic system" (2009, p. 116). Nath continues, arguing that "the human mind is beyond the sphere of computationality, because the human mind has innate ideas, which are embedded as the innate dispositions of the human mind. These ideas a priori in the human mind and are the basic in-born propensities" (Nath 2009, p. 118), a view validated by Descartes, who claimed that

"understanding what is called a thing, or a truth, or a thought, it appears to me that I hold this power from no other source than my own nature" (1911, p. 14). In other words, rational thought is partially formed from an intrinsic intuition that cannot be captured as computation. To Descartes, no matter how "intelligent" a machine may appear to be, the fact that computation does not allow for subjectivity prevents the machine from true intelligence.

Much later, philosopher and author of *Language and Mind*, Noam Chomsky, affirms Descartes' ideas of innate intelligence in the context of linguistics. "As Descartes himself quite correctly observed, language is a species-specific human possession, and even at low levels of intelligence... we find a command of language that is totally unattainable by an ape" (Chomsky 2005, p. 9). Chomsky argues that if there exists a property that persists across every language, the finding would be indicative of a "universal grammar" and suggests that language acquisition is genetically predetermined. Thus, if there is a component to human intelligence that is only attainable through biology, no machine nor computer could ever truly exhibit human-level intelligence. This is the non-computational theory of mind.

From Homer's tales of Hephaestus' automata, where intelligence is *mechanical*, to Hobbes' algebraic operations of thought, where intelligence is *computational*, and even Chomsky's universal grammar, which intelligence is *genetic*, history has shown how the substance of intelligence is one that is ever-changing. The very nature of intelligence is steeped in vacillating philosophical debate. This has led to an aphorism in the modern AI community: "AI is what we don't yet understand." From the perspective of the early 21st century, it may be easy to look back and criticise the naivety. However, definitions of intelligence has been, and will continue to be, snapshots of a greater temporal continuum. Our current perceptions of intelligence may yet be another step in the journey toward what is truly artificial intelligence.

2.3. The Digitised Intelligence

Intelligence has evaded a lasting definition, however, similar to that of Descartes, if one were to equate intelligence to cognitive ability, a consensus emerges. Both philosophers and technologists alike suggests “that human reasoning, learning, and inference comprise one of the most sophisticated thinking systems in existence” (Chen et al. 2016). However, information processing conducted in biological systems are limited in both scalability and bias. If further evolution (natural or otherwise) happened to enlarge the human brain, the axons connecting neurons must extend to compensate, subsequently increasing the distance for the signal to travel. If the speed of the signal remains constant, this increase in distance will extend the time it takes for signals to traverse the brain, thus slowing thought. In fact, almost any change to the brain—increasing the interconnectedness, signalling speed, or neuron count—comes with a detrimental compromise, usually in energy cost or sporadic behaviour (Fox 2011). Thus, biological "intelligence has a threshold" (Gladwell 2011, p. 80). There are, however, other methods toward enhancing our cognitive abilities. Neurological stimulants, eugenics, and genome editing, are "clearly feasible", "however, when compared with possible-breakthroughs in machine intelligence, would be relatively slow and gradual" (Bostrom 2014, p. 50), not to mention the ethical debate that often ensues. Regardless, if an enhanced biological intelligence came to be, would we still call them 'artificial' intelligence? Although stimulants and genome editing is by no means 'natural', their subject is still human. These blurry lines were sharpened between 1936 to 1938, when the advent of the programmable computer gave way to a new type of 'artificial'. Theoretically unrestrained by physical size, energy consumption, or processing power, the computer was immediately identified as the ideal vehicle to realise artificial intelligence.

The programmable computer was first invented with the goal of automating sequences of logical operations. In essence, what we know as a calculator. And although the calculator can operate at a speed and accuracy beyond that of any human, one would not regard it as "intrinsically intelligent" (Fogel 2006, p. 17). Everything a calculator 'knows' is already pre-programmed by

the human. Calculators are only able to solve these problems because a human has already solved it, albeit, perhaps more onerously. However, as delineated in the progression of invention, even though the computer was invented to automate calculation, its potentiality grew with its ubiquity.

Thanks to Descartes, who established the human as the exemplar for intelligence, an early approach to creating machine intelligence was simulating specific domains of human expertise. This initially manifested as two-person games of strategy. Christopher Strachey's 1952 checkers-playing program, and the myriad of chess-playing programs from 1956 and onwards, are examples of what was once considered the forefront of AI. This all culminated in 1997, where IBM's Deep Blue defeated Garry Kasparov, Russian grandmaster at the time, in match play, scoring two wins, one loss, and three draws (Silver 2012). Although the event received widespread media attention and public speculation of machines more intelligent than humans, Deep Blue wasn't an intelligent machine.

Deep Blue was only able to defeat Kasparov with the help of 32 parallel processors and 512 custom application-specific integrated circuits (ASIC), which allowed a search of 200 million chess positions per second. "This level of play requires many millions of times as much computing as a human chess player does" (McCarthy 1997, p. 1518) and takes advantage of its incredible speed to imitate intelligence. Furthermore, Deep Blue did not have the capacity to 'learn'. There were attempts to incorporate "automated tuning", however a member of the Deep Blue team claimed that it was a "clunky process" and they "never found a good way to make [it] work" (Clark 1997, p. 31). Between games in the 1997 match, adjustments were made to Deep Blue based on Kasparov's play, but again, these decisions were derived from the intelligence of the human, not the machine. "Such programs did not embody intelligence and did not contribute to the quest for intelligent machines. A person isn't intelligent because he or she is a chess master; rather, that person is able to master the game of chess because he or she is intelligent" (Schank & Childers 1984, p. 30). David Fogel, author of *Evolutionary Computation: Toward a New*

Philosophy of Machine Intelligence, elegantly places “the dream of the intelligent machine [as] the vision of creating something that does not depend on having people pre-program its problem-solving behaviours” (2006, p. 1). And yet again, the notion of intelligence takes another step in its evolution—this time, possessing the ability to learn.

These aforementioned game-playing systems fall within one of the two paradigms that divided early AI research. The first paradigm, 'symbolic AI', which was later nicknamed by John Haugeland as "Good Old-Fashioned Artificial Intelligence" (1985), denotes a system that leverages explicit and high-level representations of knowledge, in the form of logic and rules. A common example of symbolic AI is the 'expert-system', whereby 'knowledge engineers' painstakingly translate their domain expertise to a symbolic representation understood by the machine. Up until the 1980s, "it seemed like knowledge engineering was about to take over the world, with countries and companies making massive investment in it" (Domingos 2015, p. 35). However, its main flaw was highlighted in, arguably, the longest running project in AI history: the 'Cyc project'.

In 1984, Doug Lenat, a Stanford professor at the time, became frustrated with hand-coding domain-specific knowledge into expert-systems. In response, Lenat started the Cyc project with the objective of creating "a knowledge base spanning all human consensus knowledge" (Lenat et al. 1990, p. 30). In Lenat's documentation of the project, *Cyc: Toward Programs with Common Sense*, published six years after the project's inception, Lenat proudly affirms that "there are currently between one and two million assertions in our knowledge base, many of which are general rules, classifications, constraints, and so on" (Lenat et al. 1990, p. 32). As the project continued, the sheer quantity of assertions that were required continued to delay the initial projected completion date of 10 years. Today, a vocal few consider the Cyc project as a failure, namely Marvin Minsky and Pedro Domingos, the latter indicating in his 2015 book, *The Master Algorithm*, that "thirty years later, Cyc continues to grow without end in sight, and commonsense

reasoning still eludes it" (Domingos 2015, p. 30). Some suggest that perhaps "all [of] human consensus knowledge" was too ambitious of a task. However, even if applied to a more constrained domain, there were still two problems that are inherent to symbolic AI: the stochastic and ever-changing nature of the world, and its inability to revise previously encoded rules. For the most part, symbolic AI are monotonic—the more rules that are added, the more vast its intelligence. However, according to previously defined attributes of intelligence—possessing the ability to learn—unable to change previous assertions, the Cyc project is not intelligent. In fact, all symbolic AI, which requires the human to encode their own knowledge into it, fails to meet this criterion. Despite the later inclusion of learning algorithms, symbolic AI lost traction and gave way to AI's other paradigm, 'connectionism'.

Connectionist AI is characterised by a network of relatively simplistic nodes that perform computations in parallel, activating simultaneously and hierarchically, all contributing to a resultant thought or action. By way of illustration, we can look to the human brain as a representation of connectionist principles. As our retinas absorb light entering the eye, the light is converted to electrochemical impulses, which is then sent to the brain for interpretation. The neurotransmitters, passing the impulses along, can either ‘excite’ or ‘inhibit’ the impulses for subsequent neurotransmitters. This simple operation is one of many that occurs in the network of neurotransmitters, constantly exciting and inhibiting impulses as we continue to perceive. Early work in connectionist AI can be attributed to Warren McCulloch and Walter Pitts. From previous research in theoretical neurophysiology, McCulloch and Pitts adopted the structural unit of the brain and the nature of their excitation to formalise nervous activity as a computational model (McCulloch & Pitts 1943). Later, Frank Rosenblatt invented the 'perceptron', a mathematical model of the neuron (1958). Together, these ideas were the building blocks of the 'artificial neural network', the quintessential algorithm of connectionism. Where symbolic AI "tried to bring about artificial intelligence the way an adult tries to learn a second language;" connectionism "tried to

make it happen in much the same way that children learn their first language" (McAfee & Brynjolfsson 2017, p. 69).

At this stage, connectionists proclaimed that their paradigm was "the obvious path to computers with humanlike intelligence; Turing and others thought it was the only plausible path" (Domingos 2015, p. 35). However, a decade later, a pivotal paper from Martin Minsky and Seymour Papert, *Perceptrons: An Introduction to Computational Geometry*, publicised a glaring limitation of the perceptron model (1969). They mathematically proved that, at the current state of the perceptron, the simple 'exclusive or' (XOR) function—not linearly separable—was something the perceptron could not model (Minsky & Papert 1969). This book alone plunged connectionism into a state of hibernation, where research slowed to an almost-standstill. It was at this point where symbolic AI reached its aforementioned zenith.

The exact year of the most recent paradigm shift—from symbolic back to connectionist—is debated. Though, most ascribes David Rumelhart's 1986 paper, *Learning Representations by Backpropagating Errors*, as the turning point (Rumelhart, Hinton & Williams 1986). Within, they suggest the use of multiple layers of perceptrons to overcome the XOR limitation shown 17 years earlier. This, however, increased the complexity for tweaking the model, which they solved with an algorithm called 'backpropagation'. As ground-breaking as the use of multiple layers would seem, it had already been suggested. In fact, it was suggested in Minsky and Papert's *Perceptrons*, the very same publication that exposed the flaw of the single perceptron model (1969). Furthermore, although Rumelhart's paper may have popularised the backpropagation technique, it also had been previously proposed. In his 1974 PhD thesis, Paul Werbos reversed the automatic differentiation technique and was the first to apply backpropagation to tweak the connections in artificial neural networks. Nonetheless, by 1986, the impression of Rumelhart's paper shifted the AI community back to connectionism.

With multilayer perceptrons overcoming its single perceptron limitation, combined with backpropagation, an effective method to train them, AI research has since been dominated by connectionism. However, by no means does this suggest that connectionism is the true formulation of AI. Far from it. The journey through the history of AI—from automata to the computational theory of mind—is suggestive that our conceptions of intelligence has and will continue to evolve.

2.4. From Connectionism to Deep Learning

As automatic self-improvement, or learning, became an integral part of artificial intelligence, connectionists turned toward statistics. Stemming from the conceptual underpinnings of 18th and 19th century statistics, such as Bayes' Theorem and the method of least squares, a subset of these concepts formed the foundation for the automated self-improvement of connectionist algorithms. This came to be known as machine learning (ML).

In 1951, Marvin Minsky and Dean Edmonds built the *Stochastic Neural Analog Reinforcement Calculator* (SNARC), which is often cited as the first artificial neural network that utilised ML (Crevier 1993, p. 34-35). Due to the subsequent popularity of the multilayer perceptron, combined with the brilliance of the backpropagation learning algorithm, artificial neural networks "have [since] seen a great flourishing" (Brynjolfsson & McAfee 2017, p. 74). In 2006, Geoffrey Hinton published ground-breaking research his seminal paper, *A Fast Algorithm for Deep Belief Nets* (Hinton, Osindero & Teh 2006), which showed how an artificial neural network with several layers could be effectively and automatically trained. And although their work has been superseded by modern breakthroughs, it was the defining moment that proved that multi-layered artificial neural networks had the capacity to produce valuable results. The era of deep learning ensued, and artificial neural networks built with several layers, or 'deep neural networks', are "now

the dominant type of artificial intelligence by far, and they seem likely to stay on top for some time" (Brynjolfsson & McAfee 2017, p. 74).

Deep learning's potential was brought to the public's attention during its involvement with the 'ImageNet' competition. ImageNet is an online database containing millions of labelled images, with over 1,000 images for each of the more than 100,000-word phrase labels. This database is the basis for the annual 'Large Scale Visual Recognition Challenge', a competition that asks participants to train a computer to visually recognise the predominant object within images. The challenge is trivial for the human, with average accuracy scores of 95%, but difficult for machines as their sense of 'sight' is an array of numbered pixel values. A slight change in the image for human vision is a potentially gargantuan change in the array of pixels. In 2010, the year the competition was first issued, the best algorithm achieved an accuracy of 71.8%, which was raised to 74.2% in the following year (Russakovsky et al. 2015). However, in 2012, a team lead by Geoffrey Hinton employed a deep learning approach, and achieved an accuracy of 83.6% (Krizhevsky, Sutskever & Hinton 2012). This staggering increase encouraged the use of more deep learning techniques in latter competitions, eventually enabling the 2015 winner to be the first to surpass human capabilities with 95.06% accuracy (He et al. 2015).

Despite connectionists experimenting with artificial neural networks in the 1950s, and machine learners using backpropagation to train them in the 1970s and 80s, there were two major forces that hindered deep learning's proliferation until after the turn of the century: massively parallel computing enabled by graphics processing units (GPUs) and the accumulation of large, high-detailed, labelled datasets. One of the flaws with deep learning, prior to parallel computing, was the arduousness of training models. As computational speed increases, in conjunction with the expected arrival of ubiquitous quantum and cloud computing, training deep learning models no longer has such a barrier. In addition, deep learning algorithms have been shown to exhibit better results when trained upon larger datasets (that is, up until a certain threshold). With data being

described as "a natural by-product of computing" (Schneier 2016, p. 20), its passive accumulation has inadvertently provided deep learning algorithms with a plethora of training datasets. Deep learning has since gone on to further the fields of speech and audio processing, language modelling and language processing, information retrieval, and beyond.

2.5. A Need for Greater Intelligence in Architecture

Since the inception of computational artificial intelligence, advocates have predicted great transformations that AI will catalyse in their respective fields. Architecture is no exception. The promise of AI's grandeur has sparked speckled interest through the history of architecture, whether that manifests as a tool that optimises certain aspects of the design process, as generative systems that aim to mimic (and replace) the architect, or even as architecture that could embody and exhibit intelligence.

Christopher Alexander, an architect and design-theorist, applied principles of cybernetics and symbolic artificial intelligence in his 1964 dissertation, *Notes on the Synthesis of Form*. Alexander identifies that the increasing complexity of design problems, combined with narrowing specialisation, leads to "widespread, diffuse, and unorganized" information (1964, p. 4). "As a result, although ideally a form should reflect all the known facts relevant to its design, in-fact the average designer scans whatever information [they] happen on, consults a consultant now and then when faced by extra-special difficulties, and introduces this randomly selected information into forms otherwise dreamt up in the artist's studio of [their] mind" (Alexander 1964, p. 4). In an attempt to remedy this, Alexander proposed a novel design process that could isolate design requirements that needed reconsideration, which he called "misfits", through the computational analysis of forms represented as sets of data. In essence, Alexander designed a recommendation engine for the revision of design requirements.

Although Alexander's later work (1968; Alexander, Ishikawa & Silverstein 1977) became a model for system architectures, computer languages, and contemporary interfaces, the ideas put forth in his dissertation stills resonates today. Issues around growing complexity and specialisation have exacerbated, and new factors—such as the ceaseless collection of data and unwieldy magnitudes of unstructured information—has furthered the need for a technological solution. Data has often been referred to as "the new oil", however, more recent thoughts have implied that data has transformed into "the exhaust of the information age" (Schneier 2016, p. 20). Bruce Schneier, author of *Data and Goliath*, identified the modern "promise of big data: save everything you can, and someday you'll be able to figure out some use for it" (2016, p. 40). However, Thomas H. Davenport, author of *Big Data at Work: Dispelling the Myths, Uncovering the Opportunities*, bemoans this mentality behind data collection, and affirms that "the point is not to be dazzled by the volume of data, but rather to analyze it—to convert it into insights, innovations, and business value" (2014, p. 2). Following current trends, it is becoming increasingly difficult for humans to extract meaning from the masses of data.

A solution to a growing complexity was the programmer. However, there is a limit to this solution, made evident when Pedro Domingos identifies the three phases to the growth of a company. The first is where every operation is performed manually, services are personalised, and items are ordered, displayed, and recommended on a case-by-case basis. As the company expands to serve more consumers, there won't be enough workers, thus they attempt to automate aspects through computerisation. "In come the programmers, consultants, and database managers, and millions of lines of code get written to automate all the functions of the company that can be automated" (Domingos 2015, p. 11). And although this may have solved the problem of increasing scale, improving both efficiency and productivity, the quality of their products and services will inadvertently decrease, because "computer programs are too rigid to match humans' infinite versatility" (Domingos, 2015, p. 11). The third and final stage of a company's growth is realised when even the hundreds and even thousands of programmers aren't enough to match demand. At

this stage, Domingos argues that "the company inevitably turns to machine learning" (Domingos 2015, p. 11).

The field of architecture has experienced technological shifts, such as the adoption of building information modelling (BIM) and in-place data visualisation (as you would see with solar analysis and structural analysis), providing the architect with more control over the design process. However, these technologies have also created new methods of data use, generation, and collection. Architecture is undergoing Domingos' second phase of growth, and design methods such as data-driven design and parametricism benefit greatly from more data, fuelling better decision-making in the design process. But as with all industries, architecture will shift to the third phase, "and those organizations that can recognise and react quickly and intelligently have the upper hand" (Davenport 2014, p. 18).

Chapter 3.

Research Objectives

The trajectory of architecture alludes to an impending moment, where programmers—enlisted to automate, analyse, and parametrise—will simply be inadequate to fathom the deluge of data. Design problems are becoming increasingly facetted, unreasonably asking designers to comprehend the vast complexity of issues at grandiose scales, within unprecedented domains, and in real-time. As this trend continues, so too does the need for more intelligent processes to extract meaning from the overabundance of information.

Deep learning has the potential to solve this. Its ability to extract insights from vast quantities of unstructured data, learn intelligent behaviour from simple heuristics, and infer complex trends from past experiences, would be invaluable tools for the designer in the age where data is more 'exhaust' than 'oil'. Its pervasive adoption in almost all fields is an indicator for the value of the technology. And yet, the architecture, engineering, and construction (AEC) field has shown a resistance to its adoption.

The overarching objective of this research is to promote, explore, and develop for a more intelligent field of architecture, through the adoption of deep learning.

Previous approaches that also promote deep learning in the AEC—whereby the desire to apply deep learning precedes the identification of a problem—are necessary steps toward the shared overarching objective; however, becomes an arms race of model accuracy, rather than a lasting fundamental shift in thinking. Moreover, the number of identifiable problems are infinite, further diminishing the impact of the aforementioned approach. In an attempt to counter this myopic

mentality, this research aims to approach the promotion, exploration, and development for deep learning in architecture through:

1. a holistic assessment of the current state of machine learning research in the architecture industry,
2. an exploration of strategies, tools, and pedagogies to facilitate the integration of deep learning, and
3. an exploration of considerations and implications of a more intelligent architecture fuelled by deep learning.

Chapter 4.

Research Questions

Toward the objective of adopting deep learning in the field of architecture, the driving research question is:

What strategies, tools, and pedagogies can be utilised or developed to foster the integration of deep learning in the AEC?

To answer this question, first there is a need to gain a preliminary understanding of the current state of deep learning within the context of the AEC. Understanding the differences of deep learning's uptake—in academia compared to industry, in the different domains of architecture, and in current approaches to facilitate the integration—is imperative. Thus, the sub-question motivating the first of three sections of this research (*chapters 7 and 8*) is:

1. ***To what extent is architectural academia and industry researching, committing, and investing in machine learning?***

Grasping the current position of machine learning in the AEC, this research then focuses on the use and development of strategies, tools, and pedagogies for the adoption of deep learning. The sub-question prompting the second section (*chapter 9*) is:

2. ***What approaches can be utilised or developed to facilitate a greater integration of deep learning in the architectural industry?***

Finally, upon developing and implementing these strategies, tools, and pedagogies, this research intends to reflect upon their effectiveness, consider the impact of deep learning in the AEC, and ask if the adoption of deep learning is fostering a more intelligent field of architecture. The final sub-question (*chapter 10*) is:

3. ***What are the considerations and implications for the adoption of deep learning in the AEC?***

Chapter 5.

Methodology

The three sub-questions delineated in *chapter 4. Research Questions* will be the skeleton for this body of work. Each sub-question will be tackled in their own sections and presented sequentially. This linearity is favoured due to the reliance latter questions' have on the results from the former.

The first sub-question will be answered through an applied, inductive methodology, whereby quantitative and qualitative data will be sourced from the architectural industry and academia. For the primary analysis conducted on the AEC industry, this research will leverage databases cataloguing industry research projects, as well as creating new data from interviews and surveys. Conversely research publications, educational tools and modules, and pedagogies will inform the secondary analysis of academia. Upon inferring conclusions from the cross-sectional data analysis, the research will switch to a deductive methodology to answer the remaining two sub-questions.

The second sub-question can be further divided into two sections. The first, documented in *chapter 9.1. Applying Active Learning Pedagogies for Teaching Backpropagation*, will utilise pre-existing student-centred pedagogies applied to an educational tool developed prior to this research. After using these strategies within the architectural curriculum for participatory primary research, this deductive methodology will leverage data collected by surveys and interviews with the participating students to understand how it was received. The second section of this sub-question, documented in *chapter 9.2. A Tool for Deep Reinforcement Learning in Grasshopper*, aims to further explore approaches for deep learning education. This uses an action research methodology, which can be defined as an iterative cycle of creation, assessment, and improvement. The software package, once developed, will be similarly tested through participatory primary research, where it will be used to teach architects deep reinforcement

learning through workshops at conferences and universities. The qualitative responses will help shape its development.

Finally, the third sub-question will follow a conclusional, deductive research methodology, whereby the results of the former two research activities will inform conclusions about the considerations and implications of deep learning in the architectural profession. The final chapter, *chapter 10.3. Is Deep Learning Really Artificial Intelligence?*, will utilise the knowledge gained from the entirety of this research, to deliberate if deep learning is indeed intelligent, under a definition of intelligence that is unaffected by the wax and wanes of technological progress.

Chapter 6.

Literature Review

6.1. Early Explorations of Intelligence in Architecture

As early as 1969, Nicholas Negroponte, founder of the *MIT Media Lab*, validated the pursuit of architectural artificial intelligence by highlighting the "deficiencies" of the architect. "First, architects cannot handle large scale problems, for they are too complex; second, architects ignore small scale problems, for they are too particular and individual (and, to them, too trivial)" (1969, p. 9). Even still, Negroponte suggests that a machine, designed to follow to rules put in place by the designer, still fails to overcome these problems. Negroponte elaborates through a discussion of ownership and what it means for a machine to be creative. "When a designer supplies a machine with step-by-step instructions for solving a specific problem, the resulting solution is unquestionably attributed to the designer's ingenuity and labors" (Negroponte 1969, p. 9). Thus, if the machine simply embodies the sensibilities of the designer, also embodied are the deficiencies. For the machine to overcome the fallacies of the human architect, Negroponte argues that the machine must also "learn to be adaptable and learn to be relevant" (1969, p. 9).

Inspired by McCulloch and Pitts' 1943 formulation of computational learning, Negroponte proposes a machine "that can intelligently respond to the tiny, individual, constantly changing bits of information that reflect the identity of each urbanite as well as the coherence of the city" (1969, p. 10). Further, Negroponte postulates that, only through a collaboration between the learning machine and the architect, can the aforementioned deficiencies be overcome. “The dialogue would be so intimate—even exclusive—that only mutual persuasion and compromise would bring about ideas, ideas unrealizable by either conversant alone. No doubt, in such a symbiosis it would not be solely the human designer who would decide when the machine is relevant” (Negroponte 1970, p. 11-12).

The architecture machine Negroponte envisions consists of five abstract components: "a heuristic mechanism, a rote apparatus, a conditioning device, a reward selector, and a forgetting convenience" (Negroponte 1969, p. 10). The heuristic mechanism is the decision-maker, designed to apply stored heuristics to minimise potential design solutions. Through repeated use, the rote apparatus updates these heuristics through the association of recurring events and solutions. The collaboration with the architect is enabled through the conditioning device and reward selector, where the architect can influence the learned heuristics through a Skinnerian application of reward and punishment. Finally, Negroponte emphasises that "unlearning is as important as learning" as "information can assume less significance over time and eventually disappear" (1969, p. 11), which is achieved through the forgetting convenience.

To build the architecture machine to a sufficiently complex capacity, however, proved to be quite difficult. In 1967, *URBAN5* was Negroponte's first major attempt to compile his ideas of Skinnerian reward and punishment, human-machine interaction, and artificial intelligence in the field of architecture (built with its predecessor, *URBAN2*, at its core). Three years later, Negroponte reflects with a chapter called *URBANS: A Postmortem*, highlighting the four major shortcomings of *URBAN5*: preinstalled heuristics resisted change, a feigning of generality through a collective of smaller, highly-specific architecture machines, inadequate capturing of slightly-complex design contexts, and a lack of input methods for more expressive interactions (Negroponte 1970, p. 94-96). Where *URBAN5* fell short, Negroponte's 1974 architecture machine sought to fix. However, all future incarnations of the architecture machine failed to achieve ubiquitous adoption and illustrated the difficulty of incorporating artificial intelligence in the design process.

Architect, Cedric Price, conducted similar explorations in the integration of artificial intelligence in architecture. However, rather than its application in the design process, Price focused on its application in the design itself. In tandem with Price's disregard toward a resolved architectural

form, Price preferred design through the lens of information architecture, i.e. "defining a system that delivered information around a building, or employed cybernetic exchanges of information, or supported a new, self-initiated approach to learning" (Steenson 2014, p. 122). Price's unique stance on design has cultivated a collection of influential works, including the *Fun Palace*, *Potteries Thinkbelt*, and the *Inter-Action Centre*, however, none are more prolific than *Generator*: the first intelligent building.

The *Generator* project was a proposal for the *Gilman Corporation* on their White Oak Plantation site in Yulee, Florida. The premise of *Generator*, a retreat centre, sought to facilitate unfamiliar social interactions between their guests, induced by a dynamic, self-organising architecture (*The Generator Project* 2015). The proposal comprised an orthogonal, foundational grid, similar to that of a chessboard. Modular components, such as wall panels, furniture, services, and fittings, would be reorganised by a mobile crane operator, in accordance to one of four programs. Three of these programs would use sensors from the modular components, and the final program would formulate new, unique arrangements, if a change hasn't been made for some time. The fourth program characterised *Generator* as having sense of boredom and creativity, leading to its designation of the first intelligent building. Like many of Price's ideas, *Generator* was never built.

A commonality shared between the works of Negroponte and Price was how their projects were not driven by a technological fetishism. In fact, for Price, technology was something he personally avoided, approaching computational designers, Julia and John Frazer, to develop the four programs for *Generator*. Yet, technologically speaking, *URBAN5* and *Generator* were markedly prescient. They represent "the nexus of architecture and nascent ubiquitous or pervasive computing" (Steenson 2010, p. 15), decades before their explorations are now being seriously pursued. Furthermore, their reliance on artificial intelligence, despite following the simple symbolic paradigm, reinforces the architectural field's need for artificial intelligence.

6.2. Deep Learning Research in Architecture

More recently, increased computational speeds and an excess of available data have brought new, innovative deep learning algorithms to the attention of architectural research. For instance, the convolutional neural network and the recurrent neural network—popular artificial neural network variants designed to comprehend spatial and temporal data respectively—have been shown to radically increase state of the art performance in a plethora of existing problem domains.

To start, the theoretical premise of the convolutional neural network (CNN) is its ability to interpret data of any dimensions. As such, it's most predominant use-case has been its application on image processing tasks, achieving state of the art performance on image classification, object detection, and semantic segmentation. Outlined below are two examples of architectural research that leverage CNNs.

The effort for "community engagement for urban decision-making is often ineffective, uninformed, and only occurs in projects' later stages" (Zhang et al. 2018, pp. 196). In 2018, *CityMatrix*, an interactive urban planning simulator, leveraged CNNs to provide computationally efficient traffic and solar performance predictions through a tactile, non-expert interface (Zhang et al.). The second research project was born of a distaste for the tedium of categorising archival architectural plans and sections. To automate the process, a software package, with a CNN at its core, was developed to take a document of architectural drawings, categorise each page as either plan or section, and returns two documents of either category (Ng et al. 2019). Furthermore, if the CNN showed any uncertainty, that drawing was flagged and presented for human judgement.

On the other hand, recurrent neural networks (RNN) are designed to understand sequences of data by internalising information gained from earlier inputs. Often more difficult to train than CNNs, RNNs have been successfully applied to natural language processing, speech recognition, and

temporal predictions. Outlined below are two applications of RNNs within the field of architecture.

Firstly, commercial agent-based pedestrian simulations unrealistically follow shortest-path algorithms. In an attempt to train an agent to exhibit more realistic and "temporally dynamic behaviour" (Karoji et al. 2019, p. 281), an RNN was employed to include the effects of stopping, congestion, agent visibility, and sightlines with key visual markers (Karoji et al 2019). Secondly, the material performance of viscoelastic materials are difficult to simulate. A step toward simplifying the process has involved the use of RNNs to learn the behaviour of strips of elastomer and synthetic rubber with varying widths (Luo, Wang & Xu 2018). Once trained, the RNN could be used to predict the resultant form from the properties of the strip, as well as deduce the varying widths a strip from a target form.

Within three of the four aforementioned instances of deep learning, conspicuously absent was its use within unique application domains. That is not to say that deep learning has yet to be successfully applied on previously unsolvable problems (as Ng's research does). However, deep learning's use on existing problems implies that a previous technique has been superseded by the performance gains of deep learning. And this alludes to an intriguing question: was it the desire to solve a problem better than existing methods, or the reputed performance of deep learning, that motivated the research? Even further, if motivations aligned more with the latter, would that not paint deep learning as the proverbial hammer in search for a nail?

A 2019 study, "the first attempt to provide a coherent and comprehensive overview of [the] advances in the field of machine learning-aided architectural design" (Papasotiriou, p. 823), provides some insights. The study leveraged term clustering and scientometrics to uncover patterns and insights from over 4000 research papers, published between 1970 and 2019, at the intersection of ML and computer-aided architectural design.

Within this research, a prevailing trend—the accelerating growth of research interest after the turn of the millennium—was not only validated, but quantified. In her concluding remarks, Papasotiriou reflects upon this, suggesting that the rapidity of publications render previous research "obsolete in a timescale of months" (2019, p. 823). This is symptomatic of the expectations of deep learning research in pre-existing problem domains: a model that trumpets an accuracy slightly above the current state of the art. Large jumps of improvements are rarely seen, as the accuracy of these models already encroach 100%. Furthermore, once a trained model overtakes current benchmarks, that in itself is enough reason to publish, and efforts to further model accuracy only delays the publication. This begs the question, what is the value of these incremental improvements? Rather than the application of deep learning in existing problem domains, would it not be more poignant to place research efforts elsewhere—for instance, novel applications of deep learning or the development of education tools and resources to reduce the barrier to entry for architects?

Finally, the study revealed seven main clusters of research interest determined by the prevalence of key terms. These clusters are architectural education, information flow, modelling processes, sustainability, research, urban planning, and feasibility. Interestingly, Papasotiriou identifies that "education appears to attract considerable interest, which underlines the momentum of machine learning, and its future role in architectural design" (2019, p. 822). This could be due simply to the general rising interest in ML or may suggest a shift from the reliance of external ML experts and consultants, to the education of architects for ML's uptake. Regardless, these results suggest that there is movement toward the inclusion of ML within architectural curricula.

The culture of machine learning research fosters a preference for incremental improvements in model accuracy. However, due to the pace of the field, and the minute increases in performance, these improvements are quickly made obsolete. Architectural ML research would yield more

value if applied in novel applications domains. One such domain that has witnessed a large portion of research efforts has been in education.

6.3. A Resistance to Adopting Machine Learning in Industry

In 2017, Klaus Schwab described his assessment of the *Fourth Industrial Revolution* as "rapidly evolving with the rise of artificial intelligence" (Skilton 2017). Almost in parallel, architectural historian, Mario Carpo, observed a shift towards a new design industry (Carpo 2017). Known as the *Second Digital Turn in Architecture*, Carpo describes a world where prediction can be based on sheer information retrieval, and form finding by simulation and optimisation can replace deduction from mathematical formulas. As computation approaches the limit of a "near infinite amount of data recorded, transmitted, and retrieved at almost no cost" (Carpo 2016), computers can exploit incredibly onerous operations, previously deemed too computationally intensive. Operations that can leverage these vast amounts of data to extract patterns and behaviours. Operations capable of 'learning'.

However, these shift changes have taken a considerable amount of time to take effect in the field of architecture, engineering, and construction (AEC). Academia has seen a mere smattering of applications, predominantly adopting deep learning's predilection for computer vision tasks (Fukuda, Kuwamuro & Yabuki 2017; Cao, Fukuda & Yabuki 2019, Ng et al. 2019), and architectural practices have yet to go beyond exploratory research and feasibility prototyping. As revealed in an international study conducted by McKinsey Digital, the AEC industry are consistently among the worst industries for the adoption of new technology, ML included (McKinsey Digital 2015, 2016, 2017a).

This may prove to be a problem, as the diffusion of technological innovation into industry is what "ultimately determines the pace of economic growth and the rate of change of productivity" (Hall

and Khan 2003). This is reflected no better than Hero of Alexandria's *Aeolipile* (*figure 1*). If pursued, invested, and adopted, this early steam engine may have invoked the industrial revolutions sixteen centuries earlier. "Once an inventor has discovered a use for a new technology, the next step is to persuade society to adopt it. Merely having a bigger, faster, more powerful device for doing something is no guarantee for ready acceptance" (Diamond 2017, p. 237).

figure 1: *Hero of Alexandria's Aeolipile.*

And this is where deep learning stands in the AEC. Deep learning has shown greater performance than other statistical techniques, and even other ML methods, as the quantity of data available increases (*figure 2*). However, the AEC maintains a resistance to its adoption. Ruminated below are barriers that affect the uptake of deep learning in other industries, in an attempt to hypothesise the potential causes for the AEC's intransigence towards deep learning.

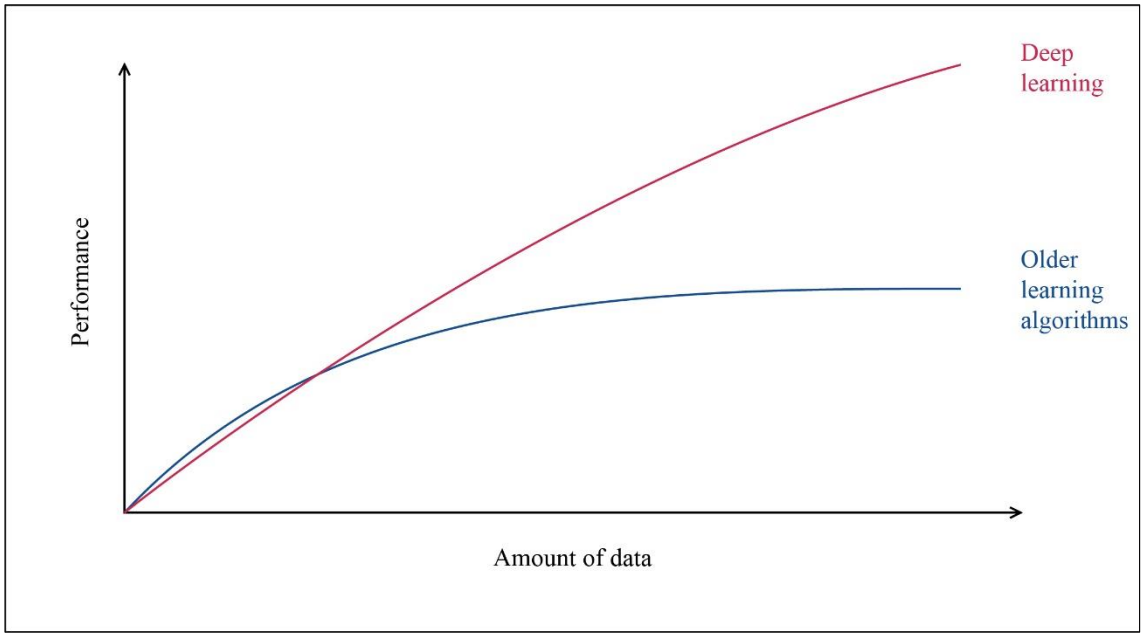

figure 2: *The performance of deep learning compared to older learning algorithms with respect with the amount of training data available.*

Despite the undoubtedly great increases in performance that deep learning offers, all deep learning algorithms are black boxes. A black box is a system that "relate the inputs to the outputs in a mathematically complex, non-transparent, and opaque way" (Baesens, 2014, p. 52). And because of this opacity, there is a hesitancy for its use in situations where a failure has terrible repercussions, as black boxes makes it impossible to understand why. Meredith Broussard, author of *Artificial Unintelligence*, suggests that "we tend to think of data as the immutable truth, but we forget that data and data-collection systems are created by people" (2018, pp. 57). Extrapolating further, these inaccuracies have the potential to be carried over to the deep learning model, who's performance is predicated on the quality of the training data. Combined with the inherent "bias we have for computerized results", "when a computer generates something, we don't question them" (Fry 2018, p. 18), and with deep learning, we *can't* question them. Hannah Fry, author of *Hello World: How to be Human in the Age of the Machine*, identifies that "this tendency of ours to view things in black and white—seeing algorithms as either omnipotent masters or a useless pile of junk—presents quite a problem in our high-tech age" (2018, p. 23). Perhaps the resistance for deep learning's adoption is due to a distrust?

On the other hand, the simple fact that deep learning is a field that is heavily steeped in computer science and mathematics, most likely beyond the comprehension of the average architect, could be the reason for its scarcity. In terms of the mathematics required to understand the nuances of deep learning algorithms, linear algebra, algorithms and complexity, probability theory, and multivariate calculus are the bare minimum. However, there is debate within the ML community as to whether the mathematical minutiae of the algorithm's internal operations are required to develop and train ML models. Some argue that an understanding of the problem domain, and an adequate command over a programming language with ML packages, would suffice. Even so, the latter requirement is becoming less important. The popularity of deep learning has garnered the development of ML platforms and frameworks that remove the need to use any programming language. Platforms such as *Google Cloud AutoML*, *Microsoft Azure ML Studio*, and *IBM Watson*

Studio are a few examples of how ML models can be built with little understanding of either computer science or mathematics.

Key thinkers assert that the *Fourth Industry Revolution* and the *Second Digital Turn in Architecture* are imminent and share the belief that artificial intelligence will be a crucial driver of the change. The rising complexity of design problems in the overflowing pool of data validates this need for deep learning in architecture. And yet, the AEC industry has shown a resistance to its adoption. Looking to other industries, this resistance may be born from a distrust of black-box algorithms, the difficulty of applying deep learning, or something else entirely. Regardless, an investigation into the state of deep learning in the AEC industry, as well as the source of this resistance, is a necessary step to elucidate the barriers for a more intelligent architecture field through deep learning.

6.4. Bridging the Gap

Due to its association with the stigma that surrounds artificial intelligence, until recently, deep learning was regarded by non-experts as an impenetrable enigma. As its popularity grows, in tandem with the number of deep learning success stories, the development of tools and platforms aiming to democratise deep learning, have found its way into all fields of endeavour, architecture included. The rising abundance of easy-to-use deep learning tools and the focus on deep learning education in architectural research is all suggestive of a movement: the combination of the architect and the deep learning engineer.

As Christopher Alexander identified, the narrowing specialisation of expertise, in combination with the sea of unstructured data, obscures the holistic understanding needed to solve complex design problems (1964). Whereas this may pose minimal concern for other industries, permitting the reliance on external consultants, it can pose a challenge in the architectural design process,

where a dynamic process of interaction between the designer and the design is required. Furthermore, an understanding of the domain is imperative for high-accuracy deep learning models. A key step in applied deep learning is the process of feature engineering, whereby knowledge of the domain is leveraged to alter the data to facilitate increased performance. "Some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used" (Domingos 2012, p. 84). Andrew Ng, co-founder of *Google Brain*, goes so far as to proclaim that although "coming up with features is difficult, time-consuming, [and] requires expert knowledge, applied machine learning is basically feature engineering" (Ng 2013, p. 16). Hence, for the AEC industry to adopt deep learning, it is imperative for architects to learn deep learning. "The future belongs to those who understand at a very deep level how to combine their unique expertise with what algorithms do best" (Domingos 2015, p. 45).

Despite the plethora of deep learning tools that reduces the barrier to entry, there remains a need to educate architects in deep learning beyond the use of a tool. Andrej Karpathy, once a research scientist for *OpenAI* and now director of AI at *Tesla*, published the blog post, *A Recipe for Training Neural Networks*, that starts with an outline of two common pitfalls for deep learning development (2019). Firstly, Karpathy asserts that training deep learning models are "a leaky abstraction". The abundance of packages and libraries proclaiming "30-line miracle snippets" gives a false impression for the simplicity of achieving high-accuracy models. "Backprop + SGD does not magically make your network work. Batch norm does not magically make it converge faster. RNNs don't magically let you "plug in" text. And just because you can formulate your problem as RL doesn't mean you should. If you insist on using the technology without understanding how it works you are likely to fail" (Karpathy 2019). To those applying deep learning without the adequate conceptual understanding, the ramifications of his first point worsens with his next: "neural net training fails silently". Within software development, when code is broken or misconfigured, errors are presented to the developer. However, contrasted with

developing for deep learning, everything can be built correctly and training starts without fault, however, some misunderstanding of a deep learning concept, carried over into the code, could prevent the training of the model to reach desired accuracies or even cause the model to learn a completely different task. "Maybe your autoregressive model accidentally takes the thing it's trying to predict as an input due to an off-by-one bug. Or you tried to clip your gradients but instead clipped the loss, causing the outlier examples to be ignored during training. Or you initialized your weights from a pretrained checkpoint but didn't use the original mean" (Karpathy 2019). These aforementioned examples, and all potential misunderstanding or incorrect implementation, will allow the network to commence training, but ultimately fail to learn the intended task. Thus, as beneficial as these tools are, there remains a need for an exploration of pedagogies and educational strategies for teaching architects the conceptual premise and operations of deep learning.

Architectural artificial intelligence started out as methods to solve unique problems. Negroponte and Price used artificial intelligence, not out of a desire to apply the technology, but almost out of need. However, the recent popularity of deep learning has shifted this mentality toward that of technological fetishism, where a thirst for applying artificial intelligence drives research, rather than an appropriate need. This shift has led to a culture of research favouring incremental improvements, resulting in research outcomes that become obsolete in a matter of months. Toward countering this rapid desuetude and the culture of techno-fetishism, the objectives of this research aims to employ strategies, tools, and pedagogies to fuse the role of the architect and deep learning engineer. By doing so, deep learning might be seen more as another tool within the architect's tool belt, to be applied when deemed appropriate.

Chapter 7.

Assessment

Toward the adoption of deep learning in the architectural profession, a preliminary study was conducted to gather an understanding of the current state of machine learning in the architecture, engineering, and construction (AEC) industry. Three avenues of exploration were outlined as preliminary trajectories:

1. assess the current state of research in the AEC industry and to the degree to which machine learning is being invested,
2. diagnose the barriers that affect the adoption of machine learning in the AEC industry beyond that of technical complexity, and
3. compare and contrast existing tools that offer the application of machine learning within architectural software.

Through the aforementioned studies, this research will compile a representation of the AEC industry and their stance on ML, which will direct the exploration and development of strategies, tools and pedagogies in later chapters.

7.1. The State of Machine Learning Research in the Architectural Industry

Introduction

To assess the state of machine learning in the AEC industry, this research partnered with Arup Engineering, a well-established, global architectural engineering firm with over 16,000 employees. Arup is known for their commitment to investing in ideas, which is evident through their *Invest in Arup* initiative: a system where employees can propose research projects and

receive the necessary funding to carry it out. The following report is the culmination of this year-long study, leveraging data from the *Invest in Arup* database, interviews with key proponents for ML at Arup, and a technical skills survey.

(Appendix A. Arup's Machine Learning Assessment was removed in public version)

Conclusion

Rather than an overview of all machine learning projects at Arup, this report investigates how documented ML research projects are perceived, understood, and used. The study uncovers three major findings: the largest concern for those applying ML is data collection and expertise, positive sentiment toward ML research saw an initial rise but has since waned, and standardisation of technologies will foster a more resilient community of ML developers.

7.2. Factors that Affect the Adoption of Machine Learning

Introduction

As part of the partnership with Arup Engineering, this research conducted a series of diagnostic data analyses to identify barriers that have the potential to hinder the adoption of machine learning in the AEC industry. The following conference paper, surmising the analyses, was presented at the 18th international conference on *Computer-Aided Architectural Design Futures* (*CAAD Futures*) held in Daejeon, South Korea. The theme of the conference was "Hello, Culture!", a reference to the iconic "Hello, World" of the first computer program, found in Brian Kernighan and Dennis Ritchie's 1978 book, *The C Programming Language* (p. 5).

(This paper can be found in Appendix B. Factors that Affect Machine Learning)

Conclusion

In accordance to the theme of the *CAAD Futures* conference, this study approached the identification of factors that hinder the adoption of machine learning in the AEC industry through a social, political, economic, and cultural lens. The study revealed that the combination of a hesitancy to apply ML approaches over pre-existing methods, and an underwhelming supportive community, reinforces the importance of digital standards and a more data-centric approach to engineering.

7.3. A Comparison of Machine Learning Tools for Architects

Introduction

Machine learning is a vast field—one that draws from applied mathematics, probability and statistics, data modelling and evaluation, algorithms and complexity, and not to mention programming itself. Thus, as these are not usually found in the architectural curriculum, architects inclined to explore ML are faced with an unusually steep technical barrier to entry. A time-honoured approach to minimise this barrier is through the use of tools. Architectural engineers, and more recently, computational designers, have been able to conduct solar analysis, structural analysis, and fluid simulations, through the use of tools that bridge the gap between mathematical complexity and architecture. As an example, the popular visual scripting plugin, *Grasshopper*, built on top of the 3D modelling software, *Rhino*, has a plethora of community-made tools on a platform called *Food4Rhino*. Respectively, *Ladybug* (Roudsari 2018), *Karamba* (2014), and *RhinoCFD* (2017) are such tools that can be utilised for the aforementioned applications.

Comparison

Author of *Architectural Intelligence: How Designers and Architects Created the Digital Landscapes*, Molly Wright Steenson, observes that "we are in the midst of a new wave of architectural design and architecture pedagogy in which the computer plays an operative role and reshapes how we teach architects and how they conceive their work. In some schools of architecture, instead of drawing, students learn "visualization"; in addition to construction, they adopt approaches to "fabrication"; they capture the information and decision-making around the architectural project in "building information models"" (2017, p. 75). As the necessity of ML tools becomes increasingly apparent, the computational design community have developed a variety of tools that incorporate ML. As of the 19th of July, the time of writing, there are six *Grasshopper* tools that proclaim the inclusion of ML algorithms: *Dodo* first released in 2015 (Greco), *Lunchbox* in 2017 (Proving Ground), *Octopus* in 2018 (Vierlinger), and *Owl* (Zwierzycki 2019a), *Wallacei* (Wallacei), and *Opossum* (Wortmann) in 2019. However, there is an argument to be made for the removal of *Wallacei* and *Opossum* from this list, as they aren't offering the user the ability to use ML. Rather, they offer the use of evolutionary algorithms, *NSGA-2* (Deb 2002) and *RBFOpt* (Costa & Nannicini 2018) respectively, which utilise ML internally. *Wallacei* and *Opossum* does not provide the user with the ability to explore ML directly, thus will be removed from further comparisons. Below is a comprehensive evaluation of the remaining four tools, documenting what ML algorithms they offer (*Table 1*).

Paradigm	Algorithm	Dodo (2015)	Lunchbox (2017)	Octopus (2018)	Owl (2019)
Supervised	Linear regression		x		
	Logistic regression	*	x		*
	Support vector machine			x	
	Nearest neighbour	x		x	
	Random forest	x			
	Naïve Bayes classifier		x		
	Neural network	x	x	x	x
Supervised/ Unsupervised	Restricted Boltzmann		x		
Unsupervised	Self-organising map	x			
	Elastic map	x			
	K-means		x		x
	t-Distributed SNE				x
	Gaussian mixture model		x		
	Hidden Markov model			x	
	Markov chain				^

Table 1: *A comparison of Grasshopper plugins and their machine learning algorithms.*

***** Despite not explicitly included, linear regression and logistic regression can be modelled using a simplified version of neural networks. However, as some of the *Grasshopper* tools does not provide the identity activation function, linear regression isn't possible.

^ Despite the fact that Markov chains are not technically an unsupervised learning algorithm—rather a statistical model—*Owl* categorised the component underneath the 'unsupervised' section.

Immediately, it is glaringly obvious how little overlap there is in terms of included algorithms across the four tools. In fact, only three of the fifteen algorithms are included more than once (four if you include logistic regression). As such, no one tool stands as the most comprehensive; but nor do any shine in a specific ML paradigm. Another observation is the absence of any reinforcement learning, one of the three paradigms. However, it should be noted that on *Owl*'s *GitHub* repository, the author recently created a subdirectory called "QLearning", a type of reinforcement learning algorithm (Zwierzycki 2019b), suggestive that Zwierzycki may include reinforcement learning in *Owl* for future versions. What can be gleaned from this preliminary review is an indication of the state of ML in architecture. Potentially due to the field being in a transitive state, much like the current state of ML use in architecture, these tools are incomplete in their offering and scattered in their domain.

By the very nature of being a tool—a bridge between complexity and the users—the authors must define their own balance between simplicity and control. A tool that is simple to learn becomes too inflexible to be applied to many domains, whereas a tool with too much control becomes convoluted and intimidating to learn. Through a review of the one algorithm that persists across all the four tools, the artificial neural network, and the amount of control each tool offers, the targeted purpose and audience for the tool can be derived (*Table 2*).

Type	Functionality	Dodo (2015)	Lunchbox (2017)	Octopus (2018)	Owl (2019)
Network architecture	Number of hidden layers	x		x	x
	Number of neurons in hidden layers	x	x	x	x
	Varying number of neurons in layers				x
Activation functions	Bipolar sigmoid	x		*	x
	Sigmoid	x	x	*	x
	Sigmoid alpha variable	x	x	*	x
	Softplus	x		*	
	Threshold			*	x
Hyperparameters	Batch size				x
	Stopping at max iterations	x	x	x	x
	Stopping at max compute time			x	
	Stopping at max memory			x	
Regularisation methods	Early stopping by error threshold	x		x	
	Early stopping by divergent steps			x	
Learning algorithms	Resilient backpropagation			x	
	Backpropagation	x	x		x
	Delta rule learning	x			
	Perceptron learning	x			
	Learning rate	x		x	x
	Momentum	x			x
Meta	Initialisation seed		x		x

Table 2: *A comparison of Grasshopper plugins and their functionality with artificial neural networks.*

***** Despite the Encog Engine (the ML engine behind Octopus) offering a wealth of different activation functions (Heaton 2014), Octopus doesn't specific which activation function was used, nor does it give the flexibility to select an activation function.

From the above table, *Lunchbox* is an example of a machine learning tool that abstracts away too much control. It removes the ability to control vital aspects when training neural networks, such as the number of hidden layers, the learning rate, and the ability to allow for mutually exclusive classification with the output activation function. In fact, *Lunchbox* errs so much on the side of simplicity, that artificial neural networks trained using *Lunchbox* are limited in the complexity of problems it can be applied to.

Octopus, on the other hand, provides the greatest interoperability during the training process. Early stopping is a regularisation technique used for controlling the balance between variance and bias. These techniques are used if the intention is to deploy the trained neural network afterward, as opposed to experimentation. That said, even though *Octopus* provides the ability to build deep neural networks, they do not allow for the variation in the number of neurons for each layer. Put another way, if you were using *Octopus* to build a deep neural network of 10 layers (an arbitrarily selected number of layers), each hidden layer is restricted to the same number of neurons—a neural network architecture that is rarely seen in practice.

The only tool that provides control over the number of neurons in each layer—arguably, one of the most important parameters for training neural networks—is *Owl*. Slightly more difficult for the beginner to use, especially with its unfamiliar terminology of 'Tensors' and 'Tensorsets', *Owl* offers almost all the vital functions required to train and use artificial neural networks: batch size, learning rate, and momentum. However, despite its functionality, the tool does not offer any way to observe or stop the training process to prevent overfitting, "a central problem to machine learning" (Domingos, 2015, p. 71).

Conclusion

These tools are spread across the spectrum from simplistic but ineffectual to functional but complex, and carry with them varying pros and cons. At the current state of these tools, the trade-

offs are too great to legitimately use them on problems of any considerable complexity. That is not to say that these tools don't have any merit, as they are great tools to learn ML in a familiar environment.

Chapter 8.

Technical Conceptual Framework

Introduction

The subsequent strategies, tools, and pedagogies discussed in this research relies heavily on a conceptual understanding of the paradigms of machine learning, in tandem with a notional grasp of the two algorithms that all deep learning systems are predicated. There are three paradigms of ML: 'supervised', 'unsupervised', and 'reinforcement' learning. In recent years, a fourth paradigm, 'semi-supervised' learning, has garnered some attention. However, within this conceptual framework, semi-supervised learning will not be discussed, as no semi-supervised learning techniques were used in this research (and an understanding of supervised and unsupervised learning is enough for one to interpolate).

First and foremost, it is imperative to understand that the three paradigms of ML are vastly different and cannot be used interchangeably; they are inherently designed to solve different problems. In the most abstract incarnation: supervised learning is used to teach a system to capture a relationship between example inputs and outputs, so that the system can provide an output prediction when it is posed with a new input; unsupervised learning is used to find an alternative representation of only input data, to uncover hidden patterns, anomalies, or associations; and reinforcement learning is used to train the behaviour of a decision-making agent, to act within an environment according to a schema of rewards and punishments. An effective method to differentiate the three paradigms is to consider what type of data is needed for training. Supervised learning requires both input and output data, unsupervised learning requires only input data, and reinforcement learning doesn't require any data beyond how to reward and punish the agent.

Furthermore, it is pertinent to note that these are the paradigms of machine learning, not deep learning. Deep learning refers to a subset of ML, characterised by a certain type of algorithms.

Whereas ML encompasses both symbolic and connectionist AI algorithms, deep learning includes only the latter. All deep learning algorithms are variations of the multilayered artificial neural network, or the 'deep' neural network, hence the subfield's name. This conceptual framework, and the larger body of research, focuses predominantly on deep learning techniques.

A commonality between deep learning algorithms across the three paradigms of ML is the method by which they learn. Backpropagation, first applied to neural networks by Werbos (1974), makes use of automatic differentiation (Linnainmaa 1970), to adjust the internal state of a neural network. The inner operations of the artificial neural network and the backpropagation algorithm will be explored below; as well as the three paradigms of ML, and their key variations of deep learning algorithms. The purpose of this conceptual framework is to provide an introduction to a basic technical understanding of ML, so that its use in later chapters can be better understood.

Artificial Neural Networks

Rooted in connectionism, artificial neural networks (ANN) were heavily inspired by the operations of the brain. According to the tenets of the *Neuron Doctrine*, the widely accepted scheme by which the brain operates, thought is exhibited when signals are sent through a mass of neurons and synapses. The computational version borrows these ideas of structure, as well as some inspiration about how the neuron fires, to build what we now know as the artificial neural network.

The most basic of artificial neural networks, the architecture considered as the starting point for all of its modern variations, contains a series of differences when compared with biological brain. The most prominent instance is the organisation of neurons into sequential layers. The first layer, known as the 'input layer', conducts no computation, and is merely a placeholder for the input signal. The last layer is the 'output layer', which should take the shape of the desired outputs, and

is where the resultant signal comes out. All the layers in between are known as 'hidden layers', and, with the output layer, is where all the computation occurs.

Each neuron in a layer is connected to every neuron in the layers immediately before and after. Each connection, the computational equivalent of a synapse, takes the form a single number known as the 'weight'. The weights between two layers can be represented as a matrix of floating-point numbers, with dimensions equal to the multiplication of the number of neurons in the connected layers. When an input signal is sent into the input layer, the signal is propagated through the weights that connect the input neurons with the first hidden layer, and the results are held in that layer of neurons. Each neuron in that layer then performs some computation that determines the degree by which the neuron fires (this calculation uses variables called a 'bias' as well as a non-linearity function called an 'activation function'). This altered signal is then sent through another set of weights to the next layer, held in that layer to perform some computation in the neurons, and fires off another set of signals. This process is repeated until that signal comes out the other end through the output layer. What was just described is known as the feedforward pass of an artificial neural network, and is what constitutes as 'thought' in ML.

Backpropagation

Once an artificial neural network model is constructed, immediately conducting an initial feedforward pass would yield an incredibly inaccurate result. In the previous section, two variables were identified that alters the resultant output signal: the weights and biases. When a model is built, these variables are initialised randomly, so it's no wonder why the outputs are unreliable. However, if one had the expected outputs for every given input, they could theoretically tweak the weights and biases until the predicted output looks more like that of the expected. This is the premise of backpropagation.

The question then becomes how to know how much to tweak these variables by. The premise of connectionism makes this a challenge, as all neurons contribute to the output to varying degrees, making it difficult to specify the precise change needed in each weight and bias. Some discount Rumelhart's efforts when attributed with the backpropagation algorithm (1986), as they credit Werbos who first applied it to neural networks years earlier (1974). However, Rumelhart's contribution revealed the greater significance of applying backpropagation to train neural networks and the implications of the efficiency the algorithm allows. Although Rumelhart didn't invent the algorithm, he is the main reason it is still used to train state of the art neural networks today.

Before backpropagation, there are two requirements: target outputs and error functions. The former requirement calls for the possession of the inputs' target outputs, an objective for the predicted outputs. As a side note, it was previously stated that only one ML paradigm, supervised learning, expects both the inputs and outputs in its training dataset. However, that does not necessarily mean that backpropagation is a supervised learning algorithm. In fact, backpropagation can be used across all paradigms of ML. This is possible by finding clever ways to fabricate the target output: such as an unsupervised learning algorithm called the 'autoencoder', which replicates the input data to act as target outputs, and a reinforcement learning algorithm called 'deep q-learning', which creates target outputs by tweaking the predicted output through a system of rewards and punishments. The second requirement of backpropagation is the error function. In this context, 'error', also known as 'cost' or 'loss', can be intuitively described as how far the predicted output is from the target output. The error function is a way to calculate this. With error calculated, backpropagation is tasked with minimising this error value across the entire training dataset.

Using a single training datapoint for illustrative purposes, backpropagation can start after sending that input datapoint through the neural network in a feedforward pass. The neural network returns

a predicted output, which can then be used to calculate the network's error. The goal of backpropagation is to adjust the weights and biases to minimise this error. The algorithm finds the partial derivatives of the network's error with respect to each weight and bias, to inform the extent of these adjustments. In other words, backpropagation determines how much to adjust the weights and biases, by calculating the impact that tweaking a single weight or bias has on the entire network. This calculation cannot happen simultaneously, because the partial derivatives (impact) of the earlier layers relies on the results of the later layers. But through the clever use of a principle in calculus called the 'chain rule', backpropagation calculates the partial derivatives of the layer closest to the output, and iteratively propagates those values back through the network to compute the other layers, hence the name, 'backpropagation'. These partial derivatives are then multiplied by a value known as the 'learning rate', which determines the pace at which the neural network learns, and then carries out these adjustments. Repeated feedforward passes, error calculations, and backpropagation, constitutes the iterative process by which an artificial neural network learns.

It is important to recognise that this process is one of the more basic methods for training artificial neural networks. Since its inception, ML researchers have since developed a plethora of optimisation techniques for improving the performance of training, regularisation techniques to prevent the common pitfalls, and intuitions that help find the best parameters.

Supervised Learning

The objective of supervised learning, one of the paradigms of machine learning, is to capture the relationship between a set of inputs and outputs, so that when posed with a new input, that learnt relationship can be used to predict the output. It's known as 'supervised' because the correct outputs, provided in the training dataset, are given to the algorithm to guide the learning process. This type of learning is similar to that of a child learning the names of shapes, where a mentor

would give the child examples. A parallel can be drawn between supervised learning and observational learning seen in social psychology, which is predicated on imitation.

Algorithms that can be used for supervised learning include regression, decision trees, random forests, k-nearest neighbours, and support vector machines. The algorithms that are explicitly deep learning include the artificial neural network, as well as their structural variants: the convolutional neural network that is designed to capture spatial relationships, and the recurrent neural network, designed to capture temporal relationships.

Supervised learning can be applied to two types of problems: regression, which returns a continuous value; and classification, which returns discrete values. In other words, regression returns a number, whereas classification results in a category.

Unsupervised Learning

Whereas supervised learning shows some semblance to observational learning, unsupervised learning is more analogous to classical conditioning in associative learning. Given a set of input data, what patterns, structures, and/or anomalies can be detected? Revisiting the example of the child learning their shapes, an application of unsupervised learning could be to ask the child to group shapes based on the shape's properties. What the child may come to discover is that some shapes have curved edges, or that others have unequal side lengths, and structure them accordingly. Through this example, the functional differences between the two paradigms become apparent. The supervised learning approach sought to teach a child the properties of shapes, so that they could apply this knowledge to classify new shapes, whereas unsupervised learning used those properties to represent a collection of shapes in a new manner.

Algorithms that can be used for unsupervised learning include k-means clustering, principal component analysis, and t-distributed stochastic neighbour embedding. The algorithms that are

explicitly deep learning include the aforementioned autoencoder, which is often used for data compression or latent space representation, and self-organising maps, a topographic organisation method where spatial distance represents the similarity of input properties.

Unsupervised learning can be applied to a range of different problems—including clustering, dimensionality reduction, and anomaly detection—often where the data is too vast, in quantity or dimension, to be understood upon inspection.

Reinforcement Learning

Reinforcement learning has strong ties to the field behavioural psychology. Often described as a computerised version of operant conditioning, reinforcement learning is a process by which agents are allowed to explore environments and are given rewards or punishments to shape their behaviour. Returning to the example of the child (the 'agent', in this case) learning to identify shapes around them (the 'environment'), the child's behaviour can be moulded with an onlooker rewarding correctly identified shapes and punishing those incorrectly identified (the scheme for rewards and punishments is known as the 'reward function'). Reinforcement learning differs from the other two paradigms largely due to the lack of training data required. Deep learning predicates its superior performance compared to other ML methods as the size of the training dataset increases, however, reinforcement learning creates its own training dataset as the agent explores the environment.

Algorithms that can be used for reinforcement learning include Q-learning, Monte Carlo learning, and temporal difference learning. The algorithms that are explicitly deep learning include the deep Q-network, a variant of the Q-learning method using a deep neural network as the policy (decision maker in the agent), and their variants: double Q-networks and duelling Q-networks.

Reinforcement learning can be applied to any problem where an agent must take actions within an environment, in order to maximise a defined reward function. Reinforcement learning has seen most of its successes in robotics and game-playing scenarios.

Chapter 9.

Strategies, Tools, and Pedagogies

The overarching objective of this research is to explore and develop methods to foster the adoption of deep learning in the AEC field. In *chapter 6.4. Bridging the Gap*, this research identified that the most effectual approach toward this was the fusing of the architect and the ML engineer. Attempting to reduce the inherently steep technical barrier to entry, this chapter presents a collection of strategies, tools, and pedagogies, including:

1. an exploration of active learning pedagogies applied toward the education of computational design students in artificial neural networks and backpropagation, and
2. the development of a software package that facilitates the learning and use of deep reinforcement learning within architectural software.

These applications, the resultant tools, and workshops conducted, are practical instances that reinforce the movement toward furthering the integration of deep learning in architecture.

9.1. Applying Active Learning Pedagogies for Teaching Backpropagation

Introduction

Prior to this body of work, a bachelor-level thesis, in pursuit of similar objectives, ventured into the development of the artificial neural network and backpropagation algorithm in software familiar to architects (Khean et al. 2018). The thesis, entitled *The Introspection of Deep Neural Networks - Towards Illuminating the Black Box*, was published in the proceedings for the 23rd international conference on *Computer-Aided Architectural Design Research in Asia* (*CAADRIA*) held in Beijing, China. The theme of the conference was “Learning, Prototyping and Adapting”,

which asked researchers for “both innovative responses integrating emerging technologies into experimental architectural practice and their critical reflection” (Xu et al. 2018, p. iv).

> "This paper describes the development of a learning tool directed at architects and designers to better understand the inner workings of machine learning. Within the parametric modelling environment of *Grasshopper*, this research develops a framework to express the mathematics and programmatic operations of neural networks in a visual scripting language. This offers a way to segment and parametrise each neural network operation into a basic expression. Unpacking the complexities of machine learning in an intermediary software environment such as *Grasshopper* intends to foster the broader adoption of artificial intelligence in architecture."
>
> (Khean et al. 2018, p. 237)

Comparison

Referring to *chapter 7.3. A Comparison of Machine Learning Tools for Architects*, where four *Grasshopper* plugins that offered the use of ML algorithms were assessed based on their functionality, below is how the developed tool compares (*Table 3*).

Type	Functionality	Dodo (2015)	Lunchbox (2017)	Octopus (2018)	Owl (2019)	Khean (2018)
Network architecture	Number of hidden layers	x		x	x	x
	Number of neurons in hidden layers	x	x	x	x	x
	Varying number of neurons in layers				x	x
Activation functions	Bipolar sigmoid	x			x	x
	Sigmoid	x	x		x	x
	Sigmoid alpha variable	x	x		x	x
	Softplus	x				x
	Softmax					x
	Threshold				x	x
	Rectified linear unit					x
Hyperparameters	Batch size				x	x
	Stopping at max iterations	x	x	x	x	x
	Stopping at max compute time			x		
	Stopping at max memory			x		
Regularisation methods	Early stopping by error threshold	x		x		x
	Early stopping by divergent steps			x		
Learning algorithms	Resilient backpropagation			x		
	Backpropagation	x	x		x	x
	Delta rule learning	x				
	Perceptron learning	x				
	Learning rate	x		x	x	x
	Learning rate decay					x
	Momentum	x			x	x
	Dropout					x
Meta	Initialisation seed		x		x	x

Table 3: *A further comparison of Grasshopper plugins and their functionality with artificial neural networks.*

On face value, the developed tool (Khean et al. 2018) offers more functionality than the four other plugins. However, the tool fails when comparing the number of algorithms provided and the speed of computation. Whereas only one algorithm was included, the artificial neural network (trained with backpropagation), the other four plugins included at least three other algorithms. In doing so, the four plugins are able to offer the user a wider breadth of applications. Furthermore, the developed tool encoded the artificial neural network and backpropagation algorithm using *Grasshopper* components alone. When compared with the other four plugins, which was written in VB.NET or C#.NET, the developed tool is orders of magnitude slower than the plugins. This is due to the computational bottleneck of the *Hoopsnake* component, used to counter *Grasshopper*'s recursive loop avoidance check. As such, although the tools offers greater functionality, using it to train a model would be unnecessarily tedious.

Despite the shortcomings of the developed tool, its intention was never to be used to train neural networks, rather, as a complementary resource for an educational module aimed at teaching architects deep learning. The developed tool, built in a software environment familiar to architects, in a fashion where all operations are transparent and interactable, offers a window into the complexities of deep learning.

At the start of 2018, this educational module took the shape of a four-week series of workshops, which was later taught to third-year Bachelor of Computational Design Students at the University of New South Wales. The development, teaching, and evaluation of the module was documented in the following conference paper, *Learning Machine Learning as an Architect, How to?*, which was presented at the 36th international conference on *Education and Research in Computer-Aided Architectural Design in Europe* (*eCAADe*) held in Łódź, Poland. The theme of the conference was "Computing for a Better Tomorrow", which called for "a revision of methods and tools applied in research, teaching, and practice" (Kępczyńska-Walczak 2018, p. v).

(This paper can be found in Appendix C. Learning Machine Learning)

Conclusion

This study reflects upon the development and execution of an educational module to teach AEC students basic deep learning algorithms in a familiar software environment. The results highlight the importance of deep learning within the architectural curriculum and validates the student-centred approach, namely the resource-based pedagogy, for ML education within the built environment.

9.2. A Tool for Deep Reinforcement Learning in Grasshopper

Introduction

Throughout this body of work, there is a recurring theme with the reinforcement learning paradigm; that is, its scarcity. From a search of the *Cumulative Index about publications in Computer-Aided Architectural Design* database, a database containing over 12,300 publications from six international conferences, only 5 papers used reinforcement learning, of which, only 2 uses deep reinforcement learning (Srinivasan & Malkawi 2005a; 2005b); the analysis of Arup's research database, described in *chapter 7.1. The State of Machine Learning Research in the Architectural Industry*, revealed no instances of reinforcement learning research projects at the time of data extraction; and of the ML plugins examined in *chapter 7.3. A Comparison of Machine Learning Tools for Architects*, none offer the ability to leverage reinforcement learning algorithms. The latter point is a major barrier for the adoption of deep reinforcement learning. Thus, much like the previously mentioned educational *Grasshopper* tool (Khean et al. 2018), a strategy to facilitate the integration of deep learning in the AEC was to develop a reinforcement learning software package aimed at providing architects a method to explore deep reinforcement

learning. The objective is to develop a software package that allows architects to, not only learn, but use deep reinforcement learning algorithms within a familiar software environment.

Development

Reiterating the outline of reinforcement learning in *chapter 8.5*, reinforcement learning is a process to train an agent to take actions within an environment, in order to maximise a defined reward function. Substantial effort in the application of reinforcement learning lies in the creation of an environment wherein the agent can perform their actions. If the objective of using reinforcement learning is to imbue intelligence into a real-world agent, the environment must be set up to be as realistic as possible. Otherwise, if the agent is to remain simulated, this is less important. It is common practice to use game engines or software libraries for the creation and simulation of the agent and environment, such as *Unity* or *OpenAI*'s *Gym*, in tandem with packages that can perform the operations needed to perform reinforcement learning, such as *ML-Agents* and *TRFL*. Thus, in the context of architecture, a combination of *Rhino* and *Grasshopper*, chosen for its architectural familiarity, and *Python*, for its power and simplicity, was identified as a suitable foundation for the software package. Further, offloading the substantial amount of computation required for training ML models to python will provide more efficient training, thus faster training time.

As with each ML paradigm, within reinforcement learning are several algorithms, each with varying strengths and weaknesses. For this software package, the Q-learning algorithm was chosen for two reasons: its ability to use deep learning models as the decision-making policy, and the recent successes of the algorithm's modern variants. Reinforcement learning is predicated on shaping the behaviour of an intelligent decision-making system. This system is called the 'policy'. Q-learning is an algorithm to train this policy, however the policy itself can be another algorithm altogether. As the focus of this research is deep learning, the deep neural network was chosen as the decision-making policy, making this specific type of reinforcement learning algorithm, 'deep

Q-learning'. Furthermore, variations of the Q-learning algorithm, such as duelling Q-learning (Wang et al. 2015) and double Q-learning (Van Hasselt, Guez & Silver 2016) have been used to show beyond human level proficiency on narrow tasks. Q-learning has been successfully applied in game-playing tasks such as ATARI games (Mnih et al. 2013), the board game *Go* (Silver et al. 2017), and even incredibly complex games like *StarCraft II* (Vinyals et al. 2019). Thus, toward the objective of providing a software package that allows architects to learn and apply deep reinforcement learning within *Grasshopper*, the deep Q-learning algorithm was chosen.

For illustrative purposes, a toy example problem—using deep Q-learning to steer a car along a road network—was used to ground the development. For this problem, the car, represented as a rectangle in *Grasshopper*, will act as the agent, interacting with a road network, represented as a planar list of curves. For simplicity, the agent will be continuously propelled forward, rather than giving control of the speed to the algorithm (although the envisioned software package can be extended upon to include this). To avoid collision with the edges of the road, the agent's immediate surroundings are inputted into the deep Q-network, to decide whether to turn left, right, or continue straight.

Having decided that the agent and environment will persist in *Grasshopper*, and the deep Q-learning algorithm will run in *Python*, a method for inter-process communication (IPC) and a data flow diagram (DFD) was required. IPC methods are operating system-specific approaches for the management and transfer of data between processes (an instance of a computing program), which was required for this project to send data between *Grasshopper* and *Python*. After a comparison between a selection of IPCs—sockets, pipes, and shared memory—ultimately, sockets were found to be the most appropriate IPC. Despite being slower than shared memory and pipes, sockets were chosen for its simplicity, control, and inbuilt synchronisation, as well as the potential for extending its capabilities for network sockets with little modification. The DFD to implement deep Q-learning in *Grasshopper* and *Python* through sockets is outlined below (*figure 3*).

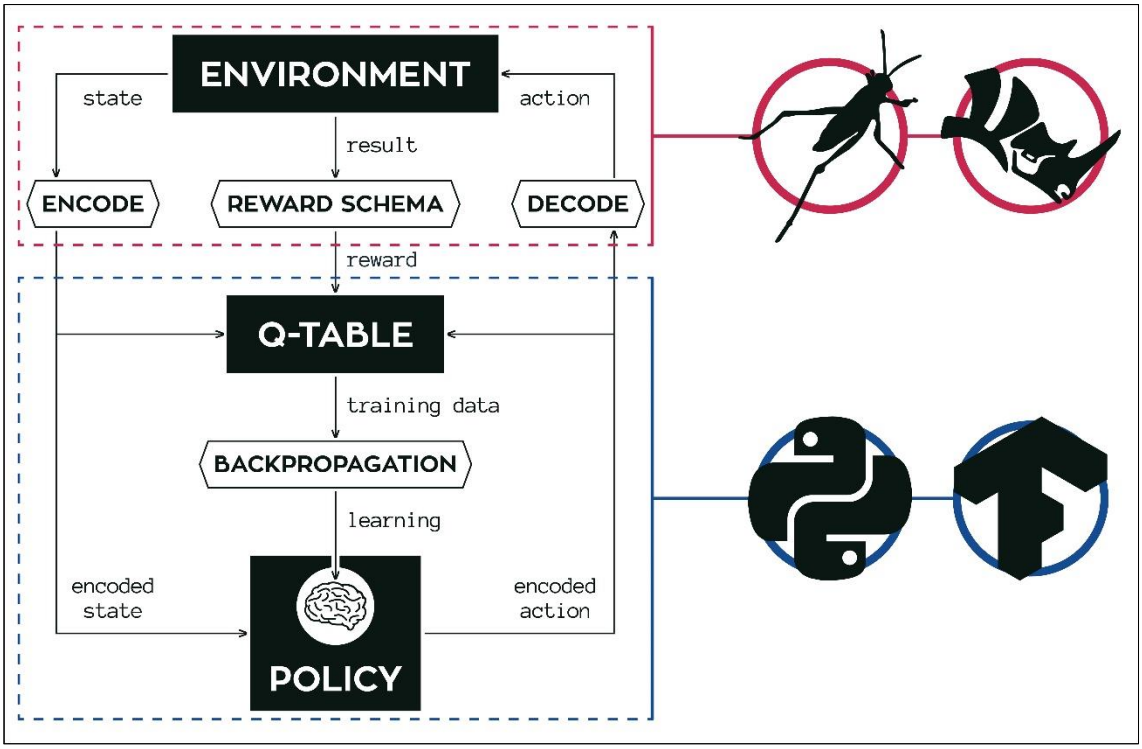

figure 3: *Data flow diagram between Grasshopper and Python for deep Q-learning.*

The versions for *Python* and the important libraries are as follows: *Python* 3.6.8, *tensorflow-gpu* 2.0.0-beta0, and *Numpy* 1.17.0. Below outlines the process by which each component of the software package was created.

1. Initialise Server and Environment

Before the training loop starts, both the local *Python* server and *Grasshopper* environment requires some initialisation steps. In the server, the neural network, acting as the decision-making policy, and the objects used to train it are defined and compiled.

```
import tensorflow as tf

...

class PolicyNetwork(tf.keras.Model):
    def __init__(self, neurons_per_hidden_layer, output_neurons):
        super(PolicyNetwork, self).__init__()
        self.hidden_layers = []
        for i, n in enumerate(neurons_per_hidden_layer):
            self.hidden_layers.append(
                tf.keras.layers.Dense(
```

```
                    units=n,
                    activation='relu',
                    name=f'hidden{i+1}'
                )
            )
        self.output_layer = tf.keras.layers.Dense(
            units=output_neurons,
            activation='sigmoid',
            name='output'
        )
    def call(self, x):
        for layer in self.hidden_layers:
            x = layer(x)
        return self.output_layer(x)

@tf.function
def train_step(inputs, labels):
    with tf.GradientTape() as tape:
        loss = loss_object(labels, model(inputs))
        grad = tape.gradient(loss, model.trainable_varaibles)
        optimiser.apply_gradients(zip(grad, model.trainable_variables))
        mean_loss(loss)

...

if __name__ == '__main__':

    # Build Model
    model = PolicyNetwork(NEURONS_PER_HIDDEN_LAYER, OUTPUT_DIMENSIONS)
    loss_object = tf.keras.losses.MeanSquaredError()
    optimiser   = tf.keras.optimizers.Adam()
    mean_loss   = tf.keras.metrics.Mean()
```

Next, there will be multiple occasions where a socket of differing ports will be bound to. Below are the functions in the server used for sending and receiving data to and from *Grasshopper* through sockets.

```
import socket

...

def recv_from_gh(socket):
    socket.listen()
    conn, _ = socket.accept()
    with conn:
        message_byte = conn.recv(BYTES)
    message_string = message_byte.decode()
    return [float(value) for value in message_string.split()]

def send_to_gh(socket, message):
    socket.listen()
    conn, _ = socket.accept()
    with conn:
        conn.send(str(message).encode())
```

When the server initialises, and the training loop starts, the server binds to the first of two ports, and waits to receive the first inputs from *Grasshopper*.

```
# Training Loop
for _ in range(ITERATIONS):
    with socket.socket(socket.AF_INT, socket.SOCK_STREAM) as s:
        s.bind((HOST, PORT_1))

        ...

        state_input = recv_from_gh(s)
```

In terms of the initialisation required in *Grasshopper*, the environment (the road network) should be drawn and internalised in a *Curves* container, and the initial position of the agent (the rectangular car) should be stored within the *Hoopsnake* component. *Hoopsnake* is a community-made component that subverts *Grasshopper*'s recursive loop avoidance check and is how the training loop will be managed (*figure 4*).

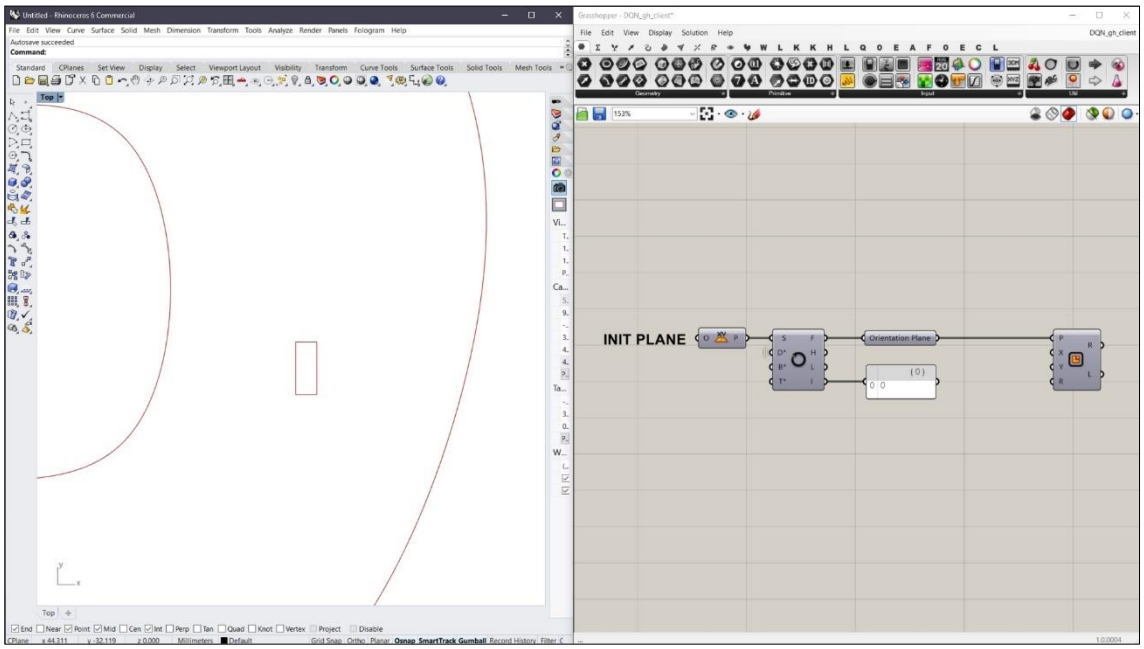

figure 4: *Initialising the Grasshopper environment.*

2. Send State to Server (Grasshopper)

At this stage, the training loop can commence. With the local server running and the initialisation steps completed, right clicking the *Hoopsnake* component and selecting 'Loop' will start training. The server has binded with the first port and is waiting on *Grasshopper* to send the first input state. The input state is how the agent perceives its environment and is the input for the neural network policy to decide upon an appropriate action. Determining what part of the problem can be an input state is a significant detail in framing a problem for deep Q-learning. This will ultimately determine how well the policy can generalise its learnt knowledge to unfamiliar environments. For this toy example, a series of lines that protrude from the front of the rectangle, representing sightlines, and their intersections with the road network will act as the input state to send to the neural network policy as inputs (*figure 5*).

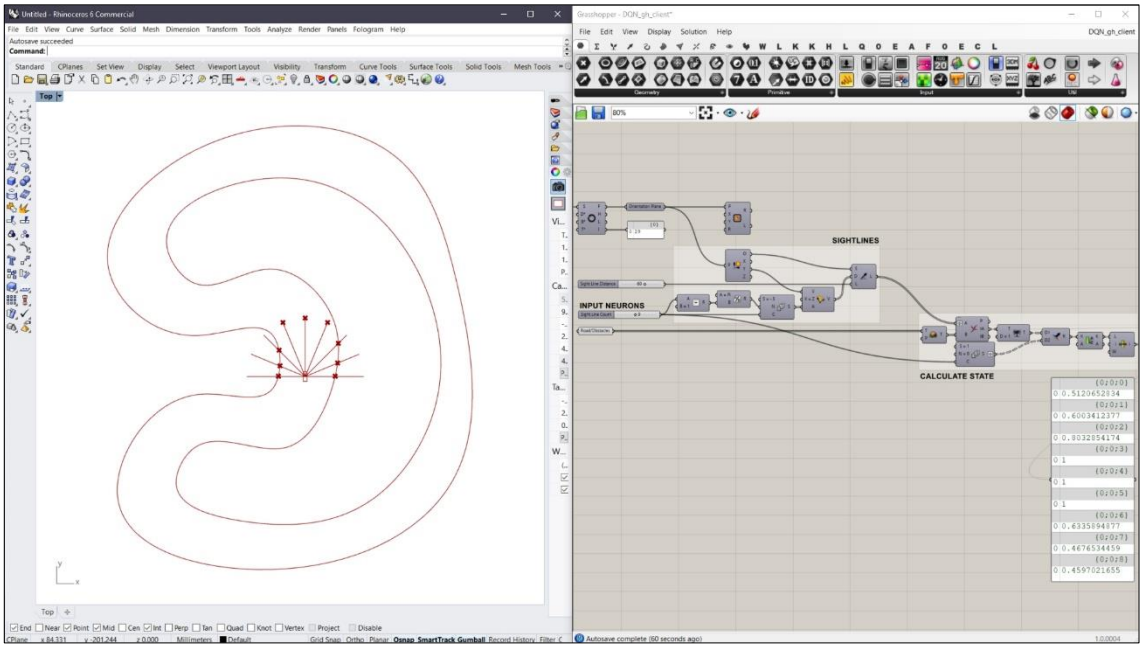

figure 5: *Sightlines from the car intersects with the road network to generate the input states.*

The intersection values (input state) should be sent to the server. For data to be sent through sockets, it first must be serialised into bytes. This is achieved by using *Grasshopper* components to convert the list of floating-point values to a string of separated values, to be encoded in the *GH_CPython* component (*figure 6*).

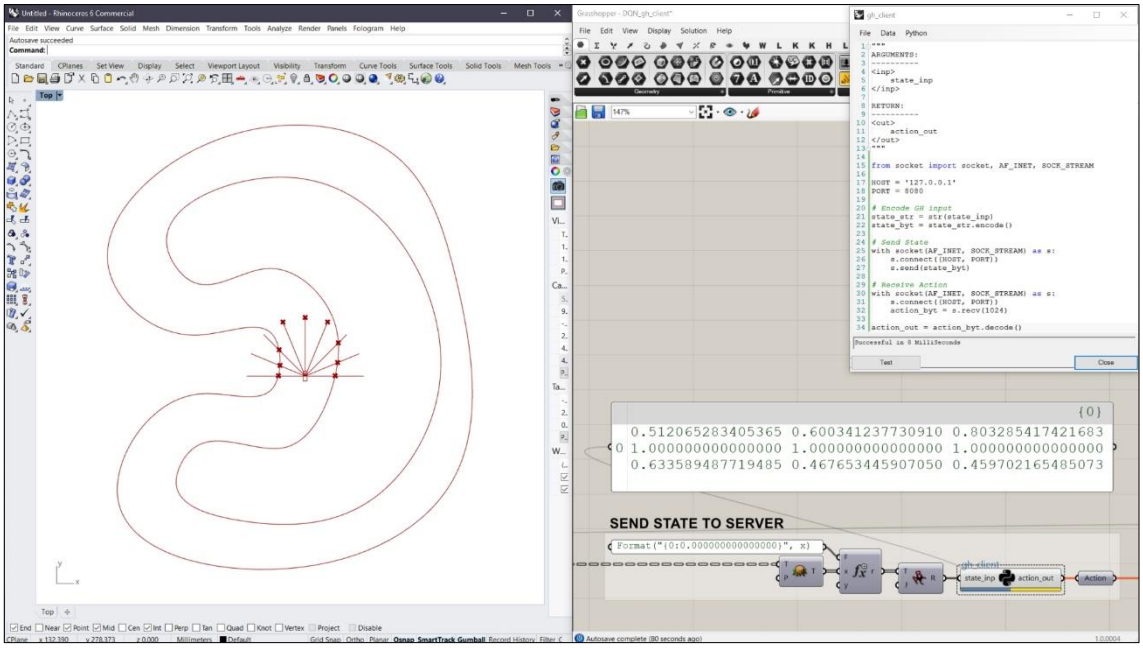

figure 6: Encoding input states and sending to Python server.

After *Grasshopper* has sent the input state to the server, it stays bound to the first port, and waits to receive an action back from the server.

3. Infer Action with Neural Network Policy (Python)

As the server receives the input state from *Grasshopper*, the input state is fed into the neural network. After a feedforward pass, the resultant outputs from the neural network are a list of probability values, corresponding to each action (in the toy example, they are: ‘turn left’, ‘turn right’, or ‘don’t turn’). In theory, the action with the highest value is what the neural network policy decides as the best action to take. This is known as a ‘greedy policy’. As the agent starts to explore the environment, if it quickly finds a behaviour that yields moderate rewards, it would continue to exhibit that behaviour, in favour of a known reward, as opposed to the uncertainty it faces with other options. This can be detrimental if another pattern of behaviour with the potential to yield even greater rewards requires the agent to perform actions that they are uncertain on. In practice, the ‘greedy policy’ leads to suboptimal behaviour. One technique that has shown promising results is the introduction of randomness. The ‘epsilon-greedy policy’ is a controlled

method to force the agent to occasionally perform random actions. Finding the balance between a greedy policy and randomness is known as the ‘exploration vs. exploitation’ problem.

```
import random

import numpy as np

...

def e_greedy_policy(q_estimates):
    if random.random() <= epsilon:
        return int(random.randint(0, OUTPUT_DIMENSIONS - 1))
    else:
        return int(np.argmax(q_estimates))

...

# Training Loop
for _ in range(ITERATIONS):

    ...

    q_estimates = model(np.array([state_input]))
    action = e_greedy_policy(q_estimates)
```

Still bound to the first port, the action is sent back to *Grasshopper* through the same socket. After, using *Python*’s context managers, the server then unbinds with the first port, and binds to the second, awaiting the reward value for the action taken from *Grasshopper*.

```
# Training Loop
for _ in range(ITERATIONS):
    with socket.socket(socket.AF_INT, socket.SOCK_STREAM) as s:

        ...

        send_to_gh(s, action)

    with socket.socket(socket.AF_INT, socket.SOCK_STREAM) as s:
        s.bind((HOST, PORT_2))

        ...

        reward = recv_from_gh(s)[0]
```

4. Perform Action and Calculate Reward (Grasshopper)

Previously, *Grasshopper* was left bound to the first socket, waiting for the action. Upon receiving the action from the server, it unbinds from the first port, and *Grasshopper* updates the position of the agent in accordance to the action (*figure 7*).

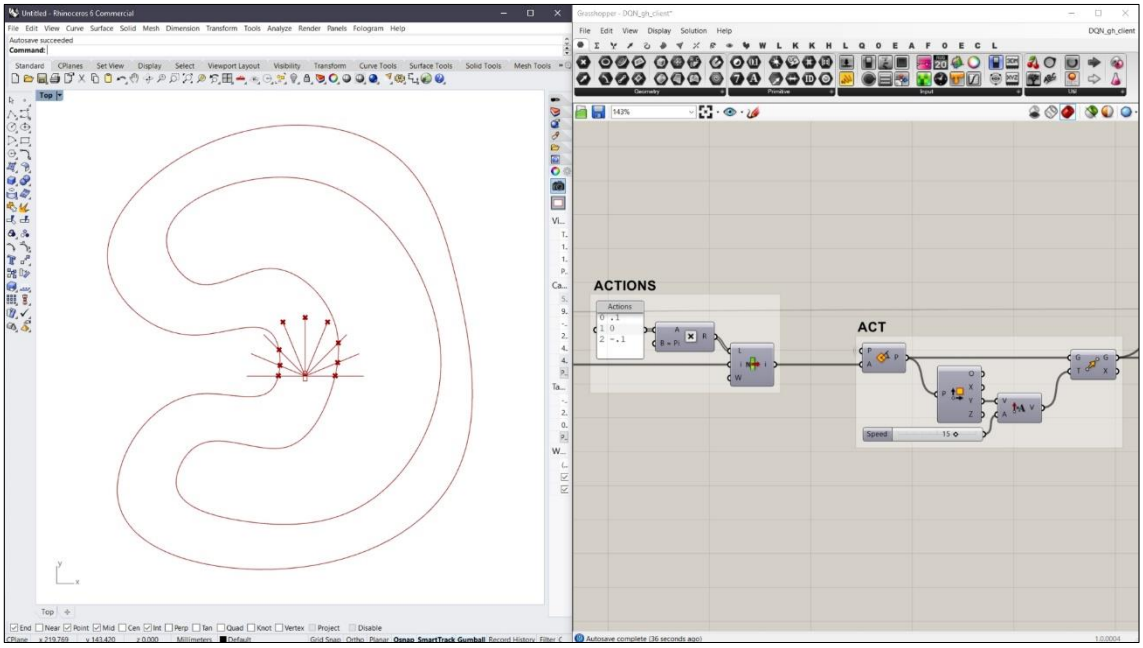

figure 7: *Receiving action from server and performing the associated action.*

In this new position, a reward should be calculated following a 'reward function'. A reward function is an omnipotent, objective measure that quantifies the contemporaneous quality of the agent. Behind deciding on what the problem's input states are, the reward function is another vital aspect of reinforcement learning. It is what ultimately directs the behaviour of the agent. For the toy example, the reward function follows three cumulative rules: add 0.1 for each iteration the car does not collide with the edge of the road, subtract 10 for a collision, and reset to 0 after a collision (*figure 8*).

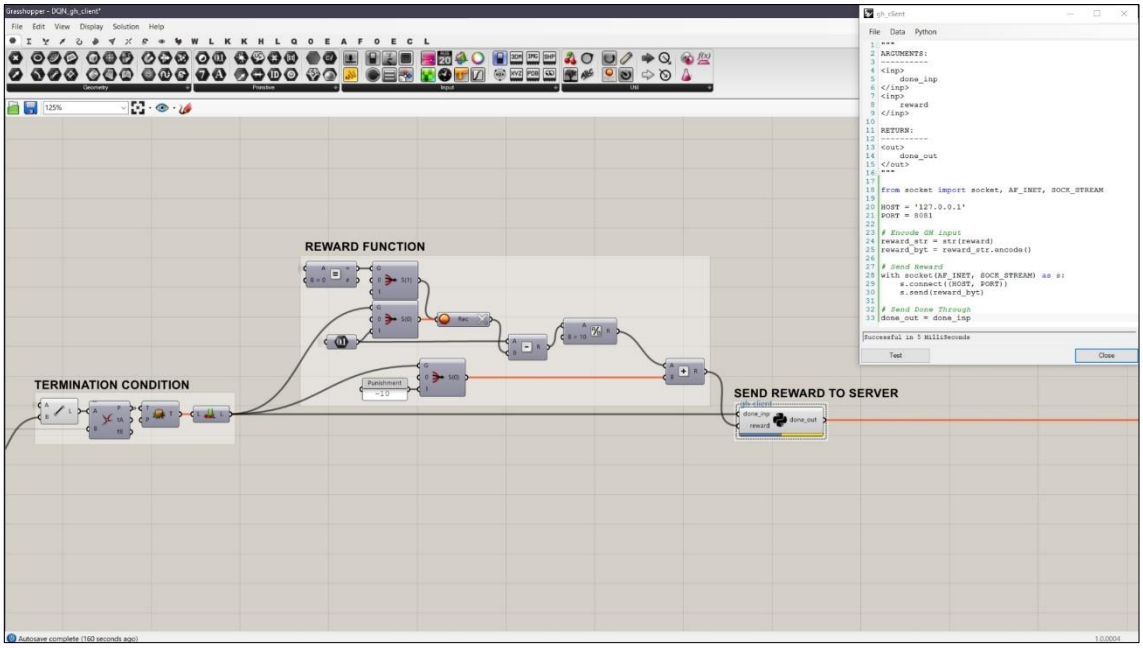

figure 8: *Calculating the reward and sending that to the Python server.*

Having calculated the reward value in accordance to the reward function, the reward is serialised, and then *Grasshopper* binds with the second port, to send the reward to the server.

```
import socket

...

reward_string = str(reward)
reward_byte = reward_string.encode()

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT_2))
    s.send(reward_byte)
```

At this stage, there is nothing left for *Grasshopper* to do besides get ready for the next iteration. To do so, the new agent position is sent back into the *Hoopsnake* component, the input state is recalculated, and binding with first port, ready to send the next input state to the server.

5. Update Policy with Q-Learning (Python)

Bound to the second port, the server should receive the reward value from *Grasshopper*. The way the toy problem is set up, when the agent (car) intersects with the environment (edge of road), the agent will be punished with a reduction of reward, and the position of the agent is reset. Upon every iteration, except those where the agent resets, the server should have four variables: the previous input state, the action decided by the policy for that input state, the reward for performing that action, and a new state. These four variables are stored into the memory of the deep Q-learning algorithm as a memory sample, and acts as the training dataset for the deep Q-network. Below is how the server samples from its accumulating memory, alters the values in accordance to the reward, and trains the deep Q-network.

```
# Training Loop
for _ in range(ITERATIONS):

    ...

    # Store Memory Sample
    memory_sample = [prev_state, action, reward, state_input]
    memory.append(memory_sample)
    if len(memory > MAX_MEMORY:
        memory.pop(0)

    # Sample Batch from Memory
    if BATCH_SIZE > len(memory):
        batch = random.sample(memory, len(memory))
    else:
        batch = random.sample(memory, BATCH_SIZE)

    # Predict Q-Values for Batch
    qsa  = model(np.array([sample[0] for sample in batch]))
    qsad = model(np.array([sample[3] for sample in batch]))

    # Set Up Arrays for Training
    x = np.zeros(shape=(len(batch), INPUT_DIMENSIONS))
    y = np.zeros(shape=(len(batch), OUTPUT_DIMENSIONS))

    for index, sample in enumerate(batch):
        pre_st, actn, rwrd, nxt_st = sample[0], sample[1], sample[2], sample[3]
        current_q = qsa[index]
        current_q[actn] = ALPHA * (rwrd + GAMMA * np.max(qsad[index]))
        x[index] = st_in
        y[index] = current_q

    # Train Model
    train_step(x, y)

    ...
```

After the deep Q-network has been trained for one iteration, the server reaches the end of the loop, and repeats the process by binding with the first port, awaiting *Grasshopper* to send the next state input.

```
# Training Loop

for _ in range(ITERATIONS):
    with socket.socket(socket.AF_INT, socket.SOCK_STREAM) as s:
        s.bind((HOST, PORT_1))

        ...

        state_input = recv_from_gh(s)
```

Application

This software package, facilitating the learning and application of deep Q-learning within *Grasshopper*, acted as the basis for an educational module. The module was formulated as a two-day intensive workshop, aimed at teaching architects with no prerequisite knowledge of machine learning the potential of reinforcement learning within the context of the built environment.

The workshop, titled *Deep Reinforcement Learning in Grasshopper: Using Deep Q-Networks to Train an Intelligent Agent to Act in a Grasshopper Environment*, was first conducted during the 24th international conference on *Computer-Aided Architectural Design Research in Asia* (*CAADRIA*) held in Wellington, New Zealand. The theme of the conference was “Intelligent and Informed”, which was “driven by the intention to take in aspects of machine intelligence, and a wide range of potential research that engages with the intelligent exploitation of computer-mediated techniques in architecture” (Schnabel, Brown & Moleta 2019, p. vi). Sixteen students, researchers, and educators from nine different countries attended the workshop. Later, the workshop was conducted at the University of New South Wales for a selection of five undergraduate computational design students. And finally, the workshop is set to run during the

upcoming 37th international conference on *Education and Research in Computer-Aided Architectural Design in Europe* (*eCAADe*) held in Porto, Portugal. The theme of the conference is "Architecture in the Age of the Fourth Industrial Revolution", provoking an inquiry into "the emerging opportunities and main threats to our discipline caused by the rise of intelligent agents" (*eCAADe 2019*).

Reflection

The described software package includes the fundamental requirements for the training of intelligent agents using deep Q-learning in *Grasshopper*. Contemporary research efforts that offer increased performance and efficiency, such as the previously mentioned Q-learning variants—double Q-learning and dueling Q-learning—can very easily be combined with the package. However, it was decided that the unembellished version would be more amenable as an educational tool.

During the workshops, often questioned was the necessity of communicating with a local *Python* server. Since the *Grasshopper* plugin, *GH_CPython*, was used to import *sockets*, why not also use *GH_CPython* to train the deep Q-network within *Grasshopper*? The answer stems from how *Grasshopper* resolves the flow of data. *GH_CPython* acts similarly to other components, meaning that the operations performed inside the component must completely resolve before any data is outputted. Thus, if a *GH_CPython* component was used to initialise a model and continue to run outside of (and in parallel to) the training loop, since the component is still running (and not resolving), the loop would not even start. If placed within the training loop, not only would it initialise the model at every iteration—effectively undoing the previous efforts of training—it would also import *Tensorflow* at every iteration (a large *Python* library, thus incredibly arduous to compute). As a result, the software package relies on the local *Python* server.

Another characteristic of *Grasshopper*, one that severely restrained the development of the software tool, was its inbuilt recursive loop avoidance check. Simply put, this is *Grasshopper*'s method for preventing a loop in the flow of data. Again, stemming from how the program works, a *Grasshopper* script acts as a blueprint for a series of computations. A loop is unfavourable in *Grasshopper*, as that blueprint would contain an infinitely long series of computations. There are methods for tricking *Grasshopper* into allowing a loop, such as the use of two *Data Dam* components wired up to some *timers*. However, this approach is inefficient, due to the replication of data, and relies on a delay rather than events. A more common work around is through the use of plugins, such as *Hoopsnake*. *Hoopsnake* allows for loops in *Grasshopper* by splitting the infinite series of computations at the point of the *Hoopsnake* component, allowing each loop to conduct the desired operations once, output a result that is fed back into the *Hoopsnake* component, which subsequently triggers the next iteration. As the time of writing, there are two other plugins that offer similar functionality, *Loop* and *Anemone*. Regardless of which method is used to facilitate recursion in *Grasshopper*, the method forces *Grasshopper* to overlook its single-threaded graph logic.

The reliance of a local *Python* server, in tandem with the substantial effort taken to subvert *Grasshopper*'s recursive loop avoidance check, begs the question, to what extent does the endeavour to shoehorn deep learning capabilities in familiar software contribute toward a greater adoption of deep learning in architecture? Do these highly specific and unique workarounds allow architects to learn the algorithm, or does it moreso ask them to learn the intricacies of a singular tool? This will be explored further in the following chapter, *10.1. Should the Architect Learn Machine Learning?*

Conclusion

The goal of the chapter, founded from a scarcity of reinforcement learning in the AEC, was to develop a software package that provided architects the ability to learn about and train deep Q-

networks in a familiar software environment. The resultant software package largely succeeded in that regard. However, its development also brought to light the uncertain value of these endeavours toward the long-term adoption of deep learning in the AEC.

Chapter 10.

Considerations and Implications

To combat the rising complexity of design problems and the prolific quantities of data, the research efforts throughout this body of work have been in the pursuit of a more intelligent field of architecture. With deep learning identified as the technology that could solve these issues, this research explored and developed strategies, tools, and pedagogies, to fuse the role of the architect and deep learning engineer, fostering the adoption of artificial intelligence-driven approaches. However, the specificity of the developed tools combined with the depth of knowledge required to productionise deep learning models brings into question the viability of combining the two roles. Further, simply understanding how to train models does not adequately encompass the exhaustive list of considerations and implications for applying deep learning. These considerations will be explored in this final chapter, which aims to:

1. discuss the necessity for combining the role of the architect and the deep learning engineer,
2. examine the often-overlooked implications for applying deep learning, and
3. ask if deep learning is indeed intelligent.

10.1. Should the Architect Learn Machine Learning

Introduction

The ambition to combine the architect and the deep learning engineer has directed the research objectives toward developing tools and creating modules to teach architects deep learning. However, the tools, developed and existing, that enables architects to train deep learning models in familiar software are often limited in either computational speed, interoperability, or control. These tools are aimed at the exploration of deep learning through exploratory research or

feasibility prototyping, rather than offering the functions needed to realistically train high performing models on complex scenarios. As a result, the educational modules that aim to teach these highly specific and niche tools are forced into teaching skills applicable only to that tool, rather than providing transferable skills to developing deep learning models. This then begs the question: are all these tools, those developed and taught in this body of work as well as the four plugins assessed in *chapter 7.3. A Comparison of Machine Learning Tools for Architects*, truly an effective method toward the objective of a more intelligent field of architecture?

Balancing Simplicity and Control

The tools taught in *chapter 9.1. Applying Active Learning Pedagogies for Teaching Backpropagation* and developed in *chapter 9.2. A Tool for Deep Reinforcement Learning in Grasshopper* have shown to be effective methods for the promotion and exploration of deep learning. They offer a method for AEC practitioners to understand the capabilities of deep learning algorithms, the types of problems they can solve, and the efforts required to train models. However, all tools exist on a spectrum—with abstracted simplicity on one end and complete control on the other—that determines the tools functionality, usability, and flexibility.

Reflecting on the backpropagation algorithm (Khean et at. 2018), although it offers greater functionality than the tools assessed in *chapter 7.3. A Comparison of Machine Learning Tools for Architects*, all operations were created using *Grasshopper* components. The motivation for this was to demystify the complexities of the backpropagation algorithm, by breaking it down to the simplest mathematical and computational operations. Encoding these base operations into *Grasshopper* components allows AEC practitioners familiar with *Grasshopper* to understand what is really happening inside the once-cryptic mass of matrices. Furthermore, *Grasshopper*'s graph logic style of computation provides an interoperability to inspect data at any stage of the process. However, despite the fact that it is fully capable of training deep learning models, the benefits gained from interoperability comes with the detriment of severely decreased efficiency.

Combined with its limited flexibility to control large quantities of data and cumbersome usability, the tool was never intended to legitimately train deep learning models.

In response to the lessons learnt above, the deep Q-learning tool developed in *chapter 9.2.* aimed to facilitate the training of deep learning models, while maintaining interoperability and an introspective quality. As such, a machine learning library, designed for speed and efficiency, was utilised to handle the majority of the deep learning computation. Furthermore, the machine learning library offers the ability to export the model once trained, allowing models to be reused. However, the want for faster computation, forced the use of software that is less familiar to architects, and that limits the potential to inspect inner operations, by hiding complexity behind a veil of "it just works". And despite the flexibility of the tool, where different problem domains and further algorithms can be added, its usability requires the architect to have a considerable grasp of the software, alluding to the learning of a tool rather than the underlying principles of deep reinforcement learning.

Although these tools allow for the exploration of deep learning in the AEC, they are too cumbersome to be used for training high performing deep learning models, as evidenced by the lengths taken in their development. Granted, the original intention of these tools were to be used as educational resources. Rather than its use for productionising models, these tools were designed with explainability, clarity, and simplicity as key pillars.

Algorithms Are Not Enough

As evidenced by *chapter 9.1. Applying Active Learning Pedagogies for Teaching Backpropagation*, a resource-based pedagogy proved an effective method for teaching deep learning within the AEC. However, although these tools, as well as those covered in *chapter 7.3. A Comparison of Machine Learning Tools for Architects*, contribute to the number of ways deep learning can be applied, they all focus on different algorithms. There is so little overlap in the

machine learning algorithms offered by these tools, that a comprehensive understanding of the ML field requires the learner to adjust to the quirks and nuances of each tool.

However, the importance placed on what algorithms are offered reinforces a detrimental "silver bullet" mentality. Too often have deep learning beginners assumed that the use of a "better" algorithm will solve their problems and yield better performing models. This may be true to an extent, however, there are a range of techniques that deep learning practitioners use, that simply aren't provided by these aforementioned tools, thus preventing the architect from training very high performing models on complex problems.

Bart Baesens, author of *Analytics in a Big Data World*, suggests that in building a statistical model "the most time-consuming step is the data selection and preprocessing step, [which] usually takes around 80% of the total efforts" (2014, p. 5). Disregarding the developed deep Q-learning software package, which does not require any training data, all other tools that offer the application of deep learning within *Grasshopper* offers no data processing or feature engineering/selection components. These tools assume that the imputed dataset is ready to be used for training, effectively skipping "80% of the total efforts". This further reinforces the "silver bullet" mentality, as it implies that by plugging data into their components, it will readily return insightful meaning. Even after raw data has been adequately preprocessed for training, the act of tweaking hyperparameters (the variables that determines the architecture of the deep learning model, the nuances of the learning algorithm, and etc.), a vital step, is limited with the simplicity of these tools. There are a plethora of other techniques used to achieve high performing models, such as cross validation, autoML, and ensembles. The shallow emphasis these tools place on algorithms, rather than offering a deep level of control, is reflected in the performance of resulting models in complex scenarios.

By training architects to use tools that abstract the functionality needed to train high performing deep learning models, architects can learn the basics of the algorithms, but would be hard-pressed to productionise any resultant models. There is benefit with learning the fundamentals of machine learning, which can be transferred to more capable machine learning libraries, however, what is the value of teaching architects through the use of these experimental tools for a more intelligent architecture through deep learning?

The Architect and the Engineer

The objective of this chapter was not to bemoan the avenues for deep learning in the AEC, but to provide a realistic perspective for the impact of these tools. A majority of architects wanting to use such tools would be inherently disadvantaged when attempting to train high performance models needed for complex problems, compared to the powerful libraries available to deep learning engineers. The depth of understanding required to deploy production-ready models requires an inordinate amount of knowledge from a broad range of mathematical and computer science topics, usually beyond that of the architectural curriculum. This disparity will inevitably lead to poorer quality models, disenfranchising the AEC upon seeing poor performance. Thus, to more effectively adopt deep learning in the AEC, accurate and reliable models built by deep learning engineers would further the field more than architects experimenting to limited capacities. Then, what is the point of these tools?

Charles Babbage, inventor of computational machines powered by punch cards, was only able to conceive it through a combination of his knowledge and the silk-weaving industry. Similarly, Henry Ford leveraged his knowledge of sewing machines and meat packing plants to invent the car manufacturing assembly line (Teodoridis, Bikard & Vakili 2018). Neither were experts in silk-weaving, sewing machines, or meat packing, but a high-level understanding provided them with the insight of when to apply those techniques. Likewise, architects with a high-level understanding of deep learning algorithms—what types of problems each algorithm was designed

to solve, the training data that is required, and the expected outcomes—allows them to identify when a problem can be solved through deep learning.

Thus, the objective of more intelligent architecture field through the adoption of deep learning, is less so about the fusing of the architect and the ML engineer, and more so an exposure of architects to these tools and algorithms, allowing an understanding of when best to apply it. Identifying when deep learning should be used, and then having the foresight to then approach external experts, is the more impactful approach toward a more intelligent AEC. And, the only way to gain this high-level understanding is by first bridging that gap through the exploration of these tools.

10.2. Considerations for the Application of Deep Learning in Architecture

Introduction

Upon deciding for deep learning to be applied on a problem within the AEC, it is imperative for the designer to thoroughly consider the implications. Almost all deep learning algorithms are treated as 'black box' algorithms, a mathematical model that lacks interpretability and explainability. Thus, when applying neural networks, expect answers without justification. Coming to terms with this fact, to further minimise the potential for negative repercussions in the application of deep learning in the built environment, there are three heuristics that should be adhered to: design for failure; garbage in, garbage out; and intelligence ages.

Failure

Deep learning has a fatal flaw: "they sometimes provide wrong answers that they are confident are right" (Agrawal, Gans & Goldfarb 2018, p. 61). This is known as the false positive and is often incredibly difficult to detect. No matter how elaborate a system of safety nets designed to

catch these failures, ultimately, all models are simplifications, and thus are unable to capture every nuance of our stochastic world. Failure is inevitable.

In conjunction, as identified in *chapter 7.2. Factors that Affect the Adoption of Machine Learning*, ML models heavily favour efficiency over accuracy—a contributing factor to the resistance of engineers. Once trained, although this efficiency has the potential to speed up the time it takes for computationally onerous tasks by orders of magnitude, what the ML model learns will always only ever be an approximation. This might be tolerable in some situations, especially for problems that are itself only approximations (i.e. fluid dynamics simulations). However, if deep learning was applied to problems with small margins for error, at best, the designer would appear foolish, and at worst, quite literally, lives could be at stake.

In lieu of a 100% accurate model, it might be prudent to adopt Domingos' attitude toward statistical models: "all models are wrong, but some are useful" (2015, p. 149). By expecting failure, domain knowledge can be leveraged to understand the tolerance of the problem, and an informed decision can be made as to whether or not deep learning is an apt approach. "When the stakes are high, accuracy is what matters most" (Fry 2018, p. 84).

In the age of deep learning, algorithms are no longer simply right or wrong. In fact, some deep learning algorithms offer no avenue to gauge a measure of accuracy. However, of all the fields of knowledge, the built environment, and more specifically architectural design, is one of the few disciplines that doesn't have a definitive solution to right or wrong. The subjective nature of architecture and design lends itself to the application of these imprecise systems, and to the ability to have different interpretations. Ultimately, deep learning is best applied to processes that are lengthy and have outcomes subject to subjectivity.

Data

One of the two factors that are commonly attributed as the reason for deep learning's success is the increasing number of ways to measure aspects of our world, combined with our incessance to record those measurements. Because of this, a common pitfall for deep learning engineers is an over-emphasis on the quantity of training data. Although prolific quantities of data have propelled deep learning beyond the achievements of other ML approaches, those vast quantities come with a cost. Big data is costly to store, manage, and move, takes longer to train models with, and have a high potential for data pollution. Furthermore, datasets in the thousands, and at times millions, simply don't exist within the targeted problem domain.

Rather than an insistence on quantity, those applying deep learning should offer a similar insistence to data quality. Senior program manager at *Microsoft*, George Krasadakis, suggests that projects that are data-intensive, encompassing a majority of ML projects, "have a single point of failure: data quality" (2017). Particularly within the AEC industry, as identified in *chapter 7.1. The State of Machine Learning Research in the Architectural Industry*, one of the major risks identified by those applying machine learning is data collection. The data collection process is unlikely to ever be perfect, as imprecise measuring implements and poorly worded survey questions plague masses of datasets. However, understanding what determines a quality dataset will aid in its acquisition, thus aid in the training and implementation of deep learning algorithms.

The exact attributes that defines high quality datasets are debated, and different problem domains call for an emphasis of some and a downplaying of others. According to Dan Ortega's 2017 article, *Seven Characteristics that Define Quality Data*, accuracy, validity, reliability, timeliness, completeness, availability, and uniqueness, are what demonstrates a quality dataset.

The call for a focus on quality over quantity requires a mentality shift. Rather than the current attitude of hoarding any and all data in hopes of a prospective usefulness, instead place focus on

identifying reliable sources of data, capturing a profile of the data, and implementing data validation techniques.

Longevity

All data is historical. The act of measuring, storing, and retrieving data immediate dates it. Thus, in the context of statistical models, we ask them to predict *future* outcomes, from a set of *present* input data, based on its understanding of the *past*. Often, we see data as an abstraction of the real world. “Meanwhile, out in the world, these numbers have consequences” (Broussard 2018, p. 114). Exacerbated with the stochastic and ever-changing nature of the world, the longer models are in service, the more inaccurate their understanding of the world becomes. This is known as ‘conceptual drift’.

In a 2018 article, *Lessons Learned Turning Machine Learning Models into Real Products and Services*, David Talby asserts that "models degrade in accuracy *as soon as* they are put in production" (2018). ML is inherently different to other types of software and should be treated as such. The assumption that ML models get better over time is often misconstrued with the improvements gained during the learning process, and not when the model is live (that is, unless there are systems in place to continually collect new data and train the model).

Tably notes the difficulty of maintaining accurate models by stating that, “unlike most things, it’s easier to get started with machine learning than it is to keep going with it… the hardest part of machine learning today is deploying and maintaining accurate models” (2018). However, there are techniques to help deploy, monitor, and uphold the accuracy of ML models: such as periodic evaluation, which requires a feedback mechanism measuring performance and indicating when retraining is needed, or continual learning, where the model is always learning.

Whichever method is employed, the deployment of ML models should be approached differently than that of other software. Upon deciding that deep learning is suitable for the problem domain, consider the longevity of the model, and the systems in place to reduce the effects of conceptual drift.

Conclusion

The proliferation of deep learning has provided built environment academics and practitioners with a new suite of powerful tools. However, the combination of a relatively young field and the rapid speed at which the field is moving, has led to deep learning's application in ill-fitting problem domains. Simply understanding *how* to apply deep learning does not adequately equip ML engineers with the ability to mitigate the host of negative implications. This chapter highlights three considerations for minimising the negative impact of deep learning models in the context of the built environment.

10.3. Is Deep Learning Really Intelligence?

Introduction

Throughout this body of research, the overarching objective was to explore and develop strategies, tools, and pedagogies to facilitate greater intelligence in the architecture industry. And as deep learning was identified as the technology that currently embodies artificial intelligence, research efforts have targeted the barriers, tools, and education for the adoption of deep learning. However, as concluded in *chapter 2.2. Reflecting on Intelligence*, the definition of intelligence is one that changes as we come to understand its previous exemplars. Thus, can we assume that, as we develop new technology that outperforms deep learning, our ideals of intelligence surpasses deep learning's capabilities? In the future, would we still consider deep learning as "artificial intelligence"? This treatise leverages the knowledge gained from two years of working with deep learning—from a mathematical and computational perspective to a more conceptual and

philosophical one—in an attempt to discern if deep learning is indeed intelligent, under a definition of intelligence that is unaffected by the wax and wanes of technological progress.

Definitions of Intelligence

In 2007, Shane Legg and Marcus Hutter collated over 70 definitions of intelligence from artificial intelligence researchers, psychologists, and collective beliefs. At the time, Legg and Hutter claimed that the definitions presented are "the largest and most well referenced collection there is" (2007, p. 1). In their concluding remarks, upon reviewing all definitions and combining the reoccuring attributes, they developed the following definition of intelligence: "intelligence measures an agent's ability to achieve goals in a wide range of environments" (Legg & Hutter 2007, p. 9). The three attributes that formed this definition are explored through the lens of deep learning below.

Is Deep Learning… Contextual?

The first and third of Legg and Hutter's attributes of intelligence suggests that intelligence is a "property that an agent has as it interacts with its environment" and depends on "how the agent is able to adapt to different objectives and environments" (2007, p. 9). This is reinforced by Ben Goertzel's *The Hidden Pattern: A Patternist Philosophy of Mind*, which states that intelligence is "the ability to achieve complex goals in complex environments" (2006, p. 198), and by Pei Wang's *On the Working Definition of Intelligence*, which states that "intelligence is the ability for an information processing system to adapt to its environment with insufficient knowledge and resources" (1995, p. 5).

Immediately, the language used makes it easy to classify one of the three schools of ML, reinforcement learning, as intelligent. Within reinforcement learning, an agent develops behaviours through interactions with its environment. The agent adapts to the environment through the development of behaviours, a process predicated on the maximisation of a reward

function. By these characteristics of intelligence, reinforcement learning agents would be considered intelligent.

Conversely, when one thinks of supervised and unsupervised learning, the environment wherein the agent interacts is often misattributed to the real world. Under this misattribution, the narrow and insular nature of these algorithms, necessitating retraining under any change of context or conceptual drift, would thus exclude these two schools from being considered intelligent. This is a misconception. The environment that these algorithms interact with and adapt to is not the real world, but the data used to train it. Within the training of supervised and unsupervised deep learning algorithms, the interaction with the environment equates to the feedforward pass, and the adaptation of the agent is the learning algorithm, backpropagation. Under this understanding of agent and environment, all deep learning algorithms exhibit the ability to interact and adapt to their respective environments, thus intelligent.

A similarity found in Legg and Hutter's first and third attributes for intelligence is the reliance on an environment. This implies that for an agent to be considered intelligent, it must perform intelligent acts (interactions and adaptation) on or within an environment. Ricardo Gudwin even asserts that "intelligent systems cannot be considered separately from the environment" (2000, p. 2080). Which raises the question: can something be considered intelligent without an environment?

Is Deep Learning… Observed?

The second characteristic that Legg and Hutter attributes to intelligence is an "agent's ability to succeed with respect to some goal or objective" (2007, p. 9). This notion is reinforced by John McCarthy's *What is Artificial Intelligence?*, which suggests that "intelligence is the computational part of the ability to achieve goals in the world" (2007), and by Ray Kurzweil's

The Age of Spiritual Machines, which claims that "intelligence is the ability to use optimally limited resources to achieve goals" (1999, p. 68).

The goal in training deep learning algorithms can be boiled down to how well the model can minimise error. To reiterate, deep learning models learn through a feedforward pass that captures the current state of its understanding, the outputs of which are used to calculate how wrong the model is (the error), which is then used for tweaking the model through backpropagation. The method by which error is calculated, the error function, can be accompanied by a measure of accuracy for supervised classification problems, and validation error, which assesses how well the model is able to generalise. Regardless of what function is used, how successful backpropagation is at reducing the error, or increasing accuracy, is indicative of the intelligence of the model.

Despite how these measures appear to be contained within the intelligent agent (the deep learning model), they are not needed for the agent to exhibit intelligence. For the agent to show intelligence, all that's required is a feedforward pass. Calculations of error and accuracy are tools for the agent to improve and are no longer required when the model is in use. Thus, the method to determine whether the agent achieves its goal, is external to the agent. Intelligence is predicated on the existence of an external observer. Then, without an observer, can an agent be intelligent?

Intelligence is Relative

In accordance with Leg and Hutter's characteristics for intelligence, deep learning is intelligent. Deep learning algorithms can interact with and adapt to an environment and is capable of achieving its goals. Thus, the efforts to adopt deep learning within the AEC industry greatly contributes to a more intelligent field.

But, with that in mind, it is almost arbitrary whether a technology is or is not intelligent, because intelligence is ultimately relative. What we consider as intelligent relies too heavily on context; the environment and the observer. If an agent was placed within an environment that allows it to achieve its goals faster, more accurately, or more efficiently, it would appear more intelligent than an environment that is more hostile. Equally, if the observer understands less of the world, intelligent agents might perform feats perceived as magic, whereas as an observer approaches omniscience, intelligence become benal. This is the reason why the technology we see as intelligent changes through time. We, as the observer, further our understanding, thus, the technology that exhibits intelligence, would seem less so as we come to understand more.

Chapter 11.

Conclusion

The motivation behind this body of research was identified to be the rising complexity of design problems in a world where domain experts and programmers cannot sufficiently develop methods to comprehend the vast quantities of data. As such, the overarching objective was to explore and develop strategies, tools, and pedagogies to promote and explore the integration of deep learning in the architectural profession.

To comprehend the current state of machine learning within the architecture, engineering, and construction field, this research explored how the industry has invested into machine learning-powered exploratory research and feasibility prototyping. Furthermore, this research conducted a study to identify the socio-economic barriers that hinders the adoption of machine learning in the industry and compared a set of tools that aimed to introduce machine learning methods to architectural software. It was found that the comparatively slow adoption of machine learning in the AEC industry is predominantly due to the high-technical barrier to entry; a problem the aforementioned tools aimed to resolve. This portion of the research highlighted the need to combine the roles of the architect and the deep learning engineer, through the use of deep learning tools and pedagogies, within the architectural curriculum.

Toward the goal of fusing the roles of the architect and the deep learning engineer, this research explored the application and development of deep learning tools within architectural software. Leveraging a *Grasshopper* tool developed prior to this research, and through a student-centred and resource-based pedagogy, it was found that the students struggled to find a reason to apply such techniques. Further, a separate software package, designed to teach architects deep reinforcement learning and in what scenarios to apply it, was the central resource used in several workshops. And although both tools and pedagogies were received well overall, it was concluded

that the complexity and depth of knowledge needed to adequately train deep learning models for production will often be beyond the skillset of the architect.

This provoked the final section of this research, which asked whether the aforementioned objective of fusing the architect and the deep learning engineer will truly aid in the integration of deep learning in the AEC. This section concluded that it would be unlikely for the architect to develop deep learning models on par with those created by dedicated deep learning engineers, and that a reliance on external experts would better contribute to the overarching objective by developing highly accurate and reliable models. This does not invalidate the tools and pedagogies explored, as there is still a need for the architect to understand these algorithms at a high level. The explored tools and pedagogies has the potential to teach architects the types of problems deep learning algorithms can solve, the types of data required for training deep learning models, and the expected results—which arms architects with the knowledge of when to apply deep learning. This approach, where architects have the conceptual understanding of knowing when to apply deep learning on a problem, would better facilitate the integration of deep learning in the AEC, galvanising a greater architectural artificial intelligence.

References

Agrawal, A, Gans, J & Goldfarb, A 2018, *Prediction Machines: The Simple Economics of Artificial Intelligence*, Harvard Business Review Press, United States of America.

Alexander, C 1964, *Notes on the Synthesis of Form*, Harvard University Press, United States of America.

Alexander, C 1968, 'Systems Generating Systems', *Architectural Design*, vol. 38, pp. 605-610.

Alexander, C, Ishikawa, S & Silverstein, M 1977, *A Pattern Language: Towns, Buildings, Construction*, Oxford University Press, United States of America.

Basesens, B 2014, *Analytics in a Big Data World: The Essential Guide to Data Science and its Applications*, John Wiley & Sons, United States of America.

Bostrom, N 2014, *Superintelligence: Paths, Dangers, Strategies*, Oxford University Press, United Kingdom.

Broussard, M 2018, *Artificial Unintelligence: How Computers Misunderstand the World*, The MIT Press, United States of America.

Brynjolfsson, E 2013, *The Key to Growth? Race with the Machines*, TED talk, accessed 24 July 2019, <https://www.ted.com/talks/erik_brynjolfsson_the_key_to_growth_race_em_with_em_the_machines>.

Brynjolfsson, E & McAfee, A 2014, *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*, W. W. Norton & Company, United States of America.

Buchanan, BG 2005, 'A (Very) Brief History of Artificial Intelligence', *AI Magazine*, vol. 26, no. 4, pp. 53-60.

Cao, R, Fukuda, T & Yabuki, N 2019, 'Quantifying Visual Environment by Semantic Segmentation Using Deep Learning: A prototype for Sky View Factor', *Proceedings of the 24th CAADRIA Conference*, vol. 2, pp. 623-632.

Carpo, M 2016, *The Second Digital Turn*, keynote lecture, accessed 6 June 2018, <https://taubmancollege.umich.edu/events/2016/10/29/lecture-mario-carpo-acadia-conference-keynote-second-digital-turn>.

Carpo, M 2017, *The Second Digital Turn: Design Beyond Intelligence*, The MIT Press, United States of America.

Cave, S & Dihal, K 2018, 'Ancient Dreams of Intelligent Machines: 3,000 Years of Robots', *Nature*, vol. 559, no. 7715, pp. 473-475, DOI:10.1038/d41586-018-05773-y.

CHAM 2017, *RhinoCFD*, Food4Rhino, accessed 19 July 2019, <https://www.food4rhino.com/app/rhinocfd>.

Chen, Y, Argentinis, JD & Weber, G 2016, 'IBM Watson: How Cognitive Computing Can Be Applied to Big Data Challenges in Life Sciences Research', *Clinical Therapeutics*, vol. 38, no. 3, pp. 688-701.

Chomsky, N 2005, *Language and Mind*, Cambridge University Press, United States of America.

Clark, D 1997, 'Deep Thoughts on Deep Blue', *IEEE Expert: Intelligence Systems and Their Applications*, vol. 12, no. 4, pp. 31.

Costa, A & Nannicini, G 2018, 'RBFOpt: An Open-Source Library for Black-Box Optimization with Costly Function Evaluations', *Mathematical Programming Computation*, vol. 10, no. 4, pp. 597-629.

Crevier, D 1993, *AI: The Tumultuous History of the Search for Artificial Intelligence*, Basic Books, United States of America.

Cross, N 1999, 'Natural Intelligence in Design', *Design Studies*, vol. 20, no. 1, pp. 25-39, DOI: 10.1016/S0142-694X(98)00026-X.

Davenport, TH 2014, *Big Data at Work: Dispelling the Myths, Uncovering the Opportunities*, Harvard Business Review Press, United States of America.

Deb, K, Pratap, A, Agarwal, S & Meyarivan T 2002, 'A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II', *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197.

Descartes, R 1911, *Meditations on First Philosophy*, e-book, translated from French by E S Haldane, accessed 9 July 2019, <http://selfpace.uconn.edu/class/percep/DescartesMeditations.pdf>.

Descartes, R 2007, *Discourse on the Method of Rightly Conducting one's Reason and Seeking Truth in the Sciences*, e-book, translated from French by J Bennett, accessed 8 July 2019, <https://www.earlymoderntexts.com/assets/pdfs/descartes1637.pdf>.

Diamond, J 2017, *Guns, Germs, and Steel: A Short History of Everybody for the Last 13,000 Years*, Penguin Random House, United Kingdom.

Domingos, P 2012, ‘A Few Useful Things to Know About Machine Learning’, *Communications of the ACM*, vol. 55, no. 10, pp. 78-87.

Domingos, P 2015, *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*, Basic Books, United States of America.

eCAADe 2019, *Architecture in the Age of the Fourth Industrial Revolution*, eCAADe, accessed 23 August 2019, <https://ecaadesigradi2019.arq.up.pt/page/theme/>.

Fogel, DB 2006, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, John Wiley & Sons, United States of America.

Fox, D 2011, ‘The Limits of Intelligence’, *Scientific American*, vol. 305, no. 1, pp. 36-43.

Fry, H 2018, *Hello World: How to Be Human in the Age of the Machine*, Doubleday, United Kingdom.

Fukuda, T, Kuwamuro, Y & Yabuli, N 2017, ‘Optical Integrity of Diminished Reality Using Deep Learning’, *Proceedings of the 35th eCAADe Conference*, vol. 1, pp. 241-250.

Gladwell, M 2008, *Outliers: The Story of Success*, Back Bay Books, United States of America.

Goertzel, B 2006, *The Hidden Pattern: A Patternist Philosophy of Mind*, Brown Walker Press, United States of America.

Good, IJ 1965, ‘Speculations Concerning the First Ultraintelligent Machine’, *Advances in Computers*, vol. 6, pp. 31-88.

Greco, L 2015, *Dodo*, Food4Rhino, accessed 19 July 2019, <https://www.food4rhino.com/app/dodo>.

Gudwin, RR 2000, ‘Evaluating Intelligence: A Computational Semiotics Perspective’, *IEEE International Conference on Systems, Man and Cybernetics*, pp. 2080-2085.

Hall, BH & Khan B 2003, ‘Adoption of New Technology’, *New Economy Handbook*, Academic Press, DOI: 10.3386/w9730.

Haugeland, J 1985, *Artificial Intelligence: The Very Idea*, MIT Press, United States of America.

He, K, Zhang, X, Ren, S & Sun, J 2015, ‘Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification’, *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026-1034.

Heaton, J 2014, *org.encog.engine.network.activation*, Encog Core 3.3.0 API, accessed 19 July 2019, <http://heatonresearch-site.s3-website-us-east-1.amazonaws.com/javadoc/encog-3.3/overview-summary.html>.

Hinton, GE, Osindero, S & Teh, YW 2006, 'A Fast Learning Algorithm for Deep Belief Networks', *Neural Computation*, vol. 18, no. 8, pp. 1527-1554.

Hobbes, T 1969, *De Corpore*, e-book, translated from Latin by Sir B W Molesworth, accessed 9 July 2019, <https://www.humanities.mcmaster.ca/~rarthur/phil4A03/thomas_hobbes_the_english_w.pdf>.

Homer 2009, *The Iliad*, translated from Ancient Greek by A S Kline, Poetry in Translation.

Karpathy, A 2019, *A Recipe for Training Neural Networks*, Andrej Karpathy Blog, accessed 21 August 2019, <http://karpathy.github.io/2019/04/25/recipe/>.

Karamba3D 2014, *Karamba3D*, Food4Rhino, accessed 19 July 2019, <https://www.food4rhino.com/app/karamba3d>.

Karoji, G, Hotta, K, Hotta, A & Ikeda, Y 2019, 'Pedestrian Dynamic behaviour Modelling', *Proceedings of the 24th CAADRIA Conference*, vol. 1, pp. 281-290.

Kępczyńska-Walczak, A 2018, 'Theme: Computing for a Better Tomorrow', *Proceedings of the 36th eCAADe Conference*, vol. 1, pp. v-vi.

Khean, N, Kim, L, Martinez, J, Doherty, B, Fabbri, A, Gardner, N & Haeusler, MH 2018, 'The Introspection of Deep Neural Networks: Towards Illuminating the Black Box', *Proceedings of the 23rd CAADRIA Conference*, vol. 2, pp. 237-246.

Krasadakis, G 2017, *Data Quality in the Era of A.I.*, freeCodeCamp, accessed 24 August 2019, <https://www.freecodecamp.org/news/data-quality-in-the-era-of-a-i-d8e398a91bef/>.

Krizhevsky, A, Sutskever, I & Hinton, GE 2012, 'ImageNet Classification with Deep Convolutional Neural Networks', *Advances in Neural Information Processing Systems*, pp. 1097-1105.

Kurzweil, R 1999, *The Age of Spiritual Machines: When Computers Exceed Human Intelligence*, Penguin, United States of America.

Lenat, DB, Guha, R, Pittman, K, Pratt, D & Shepherd, M 1990, 'Cyc: Toward Programs with Common Sense', *Communications of the Association for Computing Machinery*, vol. 33, no. 8, pp. 30-50.

Linnainmaa, S 1970, 'The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors', Master's thesis, University of Helsinki.

Luo, D, Wang, J & Xu, W 2018, 'Applied Automatic Machine Learning Process for Material Computation', *Proceedings of the 36th eCAADe Conference*, vol. 1, pp. 109-118.

McAfee, A & Brynjolfsson, E 2017, *Machine | Platform | Crowd: Harnessing Our Digital Future*, W. W. Norton & Company, United States of America.

McCarthy, J 1997, AI as Sport', *Science*, vol. 276, no. 5318, pp. 1518-1519, DOI:10.1126/science.276.5318.1518.

McCarthy, J 2007, *What is Artificial Intelligence?* accessed 24 August 2019, <http://www-formal.stanford.edu/jmc/whatisai/node1.html>.

McCulloch, WS & Pitts, WH 1943, 'A Logical Calculus of the Ideas Immanent in Nervous Activity', *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133.

Minsky, M & Papert, S 1969, *Perceptrons: An Introduction to Computational Geometry*, The MIT Press, United States of America.

Mnih, V, Kavukcuoglu, K, Silver, D, Graves, A, Antonoglou, I, Wierstra, D & Riedmiller, M 2013, 'Playing ATARI with Deep Reinforcement Learning', arXiv preprint arXiv:1312.5602.

Nath, R 2009, *Philosophy of Artificial Intelligence: A Critique of the Mechanistic Theory of Mind*, Universal Publishers, United States of America.

Negroponte, N 1969, 'Toward a Theory of Architecture Machines', *Journal of Architectural Education*, vol. 23, no. 2, pp. 9-12.

Negroponte, N 1970, *The Architecture Machine*, The MIT Press, United States of America.

Ng, A 2013, *Machine Learning an AI via Brain Simulations*, accessed 21 August 2019, <http://ai.stanford.edu/~ang/slides/DeepLearning-Mar2013.pptx>.

Ng, JMY, Khean, N, Madden, D, Fabbri, A, Gardner, N, Haeulser, MH & Zavoleas, Y 2019, 'Optimising Image Classification: Implementation of Convolutional Neural Network Algorithms to Distinguish Between Plans and Sections within the Architectural, Engineering

and Construction (AEC) Industry', *Proceedings of the 24th CAADRIA Conference*, vol. 2, pp. 795-804.

Ortega, D 2017, *Seven Characteristics that Define Quality Data*, Blazent, accessed 24 August 2019, <https://www.blazent.com/seven-characteristics-define-quality-data/>.

Papasotiriou, T 2019, 'Identifying the Landscape of Machine Learning-Aided Architectural Design: A Term Clustering and Scientometrics Study', *Proceedings of the 24th CAADRIA Conference*, vol. 2, pp. 815-824.

Proving Ground 2017, *Lunchbox*, Food4Rhino, accessed 19 July 2019, <https://www.food4rhino.com/app/lunchbox>.

Rosenblatt, F 1958, 'The Perceptron: A probabilistic Model for Information Storage and Organization in the Brain', *Psychological Review*, vol. 65, no. 6, pp. 386-408.

Roudsari, MS 2018, *Ladybug Tools*, Food4Rhino, accessed 19 July 2019, <https://www.food4rhino.com/app/ladybug-tools>.

Rumelhart, DE, Hinton, GE & Williams, RJ 1986, 'Learning Representations by Back-propagating Errors', *Nature*, vol. 323, no. 6088, pp. 533-536.

Russakovsky, O, Deng, J, Su, H, Krause, J, Satheesh, S, Ma, S, Huang, Z, Karpathy, A, Khosla, A, Bernstein, M, Berg, AC & Fei-Fei, L 2015, 'ImageNet Large Scale Visual Recognition Challenge', *International Journal of Computer Vision*, vol. 155, no. 3, pp. 211-252.

Schank, RC & Childers, PG 1984, *The Cognitive Computer: On Language, Learning, and Artificial Intelligence*, Addison-Wesley, United States of America.

Schnabel, MA, Brown, A & Moleta T 2019, 'Conference Theme: Intelligent & Informed', *Proceedings of the 24th CAADRIA Conference*, vol. 1, pp. vi.

Schneier, B 2016, *Data and Goliath: The Hidden Battles to Collect Your Data and Control the World*, W. W. Norton & Company, United States of America.

Schwab, K 2017, *The Fourth Industrial Revolution*, Currency, United States of America.

Silver, N 2012, *The Signal and the Noise: Why So Many Predictions Fail—but Some Don't*, The Penguin Press, United States of America.

Silver, D, Schrittwieser, J, Simonyan, K, Antonoglou, I, Huang, A, Guez, A, Hubert, T, Baker, L, Lai, M, Bolton, A & Chen, Y 2017, 'Mastering the Game of Go Without Human Knowledge', *Nature*, vol. 550, no. 7676, p. 354-359.

Skilton, M 2017, ‘How do we Prepare for the Artificial Intelligence Society?’, *Huffington Post*, 3 January, accessed 6 June 2018, <https://www.huffingtonpost.com/entry/how-do-we-prepare-for-the-artificial-intelligence-society_us_58b680d1e4b0e5fdf61978b5>.

Srinivasan, RS & Malkawi, AM 2005a, ‘Reinforcement Learning and Real-time Building Thermal Performance Data Visualization’, *Proceedings of the 10th CAADRIA Conference*, vol. 2, pp. 141-148.

Srinivasan, RS & Malkawi, AM 2005b, ‘Real-time Simulations Using Learning Algorithms for Immersive Data Visualization in Buildings’, *International Journal of Architectural Computing*, vol. 3, no. 3., pp. 265-280.

Steenson, MW 2010, ‘Cedric Price's Generator’, *Journal of the American Institute of Architecture Students*, vol. 69, pp. 12-15.

Steenson, MW 2014, ‘Architectures of Information: Christopher Alexander, Cedric Price, and Nicholas Negroponte & MIT's Architecture Machine Group’, PhD thesis, Princeton University.

Steenson, MW 2017, *Architectural Intelligence: How Designers and Architects Created the Digital Landscapes*, The MIT Press, United States of America.

Teodoridis, F, Bikard, M & Vakili, K 2018, ‘When Generalists are Better than Specialists and Vice Versa’, *Harvard Business Review*, accessed 26 August 2019, <https://hbr.org/2018/07/when-generalists-are-better-than-specialists-and-vice-versa>.

The Generator Project 2015, accessed 18 August 2019, <http://www.interactivearchitecture.org/the-generator-project.html>.

Van Hasselt, H, Guez, A & Silver, D 2016, ‘Deep Reinforcement Learning with Double Q-Learning’, *13th AAAI Conference on Artificial Intelligence*.

Vierlinger, R 2018, *Octopus*, Food4Rhino, accessed 19 July 2019, <https://www.food4rhino.com/app/octopus>.

Vinyals, O, Ewalds, T, Bartunov, S, Georgiev, P, Vezhnevets, AS, Yeo, M, Makhzani, A, Küttler, H, Agapiou, J, Schrittwieser, J & Quan, J 2017, ‘Starcraft II: A New Challenge for Reinforcement Learning’, arXiv preprint arXiv:1708.04782.

Wallacei 2019, *Wallacei*, Food4Rhino, accessed 19 July 2019, <https://www.food4rhino.com/app/wallacei-0>.

Wang, P 1995, ‘On the Working Definition of Intelligence’, *Center for Research on Concepts and Cognition CRCC, Indiana University*.

Wang, Z, Schaul, T, Hessel, M, van Hasselt, H, Lanctot, M & de Freitas, N 2015, ‘Dueling Network Architectures for Deep Reinforcement Learning’, arXiv preprint arXiv:1511.06581.

Werbos, P 1974 ‘Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences’, PhD thesis, Harvard University.

Wortmann, T 2019, *Opossum*, Food4Rhino, accessed 19 July 2019, <https://www.food4rhino.com/app/opossum-optimization-solver-surrogate-models>.

Xu, W, Huang, W, Liu, Y, Zhou, Y, Xu, F & Yu, L 2018, ‘Conference Theme: Learning, Prototyping and Adapting’, *Proceedings of the 23rd CAADRIA Conference*, vol. 1, pp. iv-v.

Zhang, Y, Gridnard, A, Aubuchon, A, Lynox, K & Larson, K 2018, ‘Machine Learning for Real-time Urban Metrics and Design Recommendations’, *Proceedings of the 38th ACADIA Conference*, pp. 196-205.

Zwierzycki, M 2019a, *Owl*, Food4Rhino, accessed 19 July 2019, <https://www.food4rhino.com/app/owl>.

Zwierzycki, M 2019b, *Owl/QAgent.vb*, GitHub, accessed 19 July 2019, <https://github.com/mateuszzwierzycki/Owl/blob/master/Owl.Learning/QLearning/QAgent.vb>.

Examining Potential Socio-economic Factors that Affect Machine Learning Research in the AEC Industry

Nariddh Khean[1(✉)], Alessandra Fabbri[1(✉)], David Gerber[2,3(✉)], and Matthias H. Haeusler[1,4(✉)]

[1] Computational Design, University of New South Wales, Sydney, NSW 2052, Australia
{n.khean,a.fabbri,m.haeusler}@unsw.edu.au
[2] Ove Arup and Partners, London, UK
david.gerber@arup.com
[3] Viterbi School of Engineering and School of Architecture, University of Southern California, Los Angeles, CA 90007, USA
dgerber@usc.edu
[4] CAFA Visual Innovation Institute, Beijing, China

Abstract. Machine learning (ML) has increasingly dominated discussions about the shape of mankind's future, permeating almost all facets of our digital, and even physical, world. Yet, contrary to the relentless march of almost all other industries, the architecture, engineering and construction (AEC) industry have lagged behind in the uptake of ML for its own challenges. Through a systematic review of ML projects from a leading global engineering firm, this paper investigates social, political, economic, and cultural (SPEC) factors that have helped or hindered ML's uptake. Further, the paper discusses how ML is perceived at various points in the economic hierarchy, how effective forms of communication is vital in a highly-specialized workforce, and how ML's unexpected effectiveness have forced policy makers to reassess data governance and privacy; all the while considering what this means for the adoption of ML in the AEC industry. This investigation, its methodology, background research, systematic review, and its conclusion are presented.

Keywords: Machine learning · Artificial intelligence · Research and development · Architecture, engineering and construction industry · Social factors · Political factors · Economic factors · Culutral factors

1 Introduction

The field of artificial intelligence (AI) has accelerated over the last decade. According to the *Artificial Intelligence Index*'s 2017 report, the heightened volume of activity – demonstrated by upsurges in published academic papers, course

J.-H. Lee (Ed.): CAAD Futures 2019, CCIS 1028, pp. 247–263, 2019.
https://doi.org/10.1007/978-981-13-8410-3_18

enrolments, job postings, start-ups, venture capitalist funding, and public perception – is indicative of society's willingness to invest in AI [1]. Recent breakthroughs in this field of research have predominantly been fueled by the successes of a subset of AI, known as machine learning [2]. Machine Learning (ML) has been so transformative, that it has come to permeate almost all facets of our digital, and even physical, world.

In 2012, journalist and author Charles Duhigg recounts the events surrounding Andrew Pole, a statistician working for *Target*. Pole was tasked to develop an algorithm to automatically determine *'which of Target's customers were pregnant'* [3]. In response, Pole and his team leveraged Target's data on the buying patterns of their customers. Their solution was so effective that, through targeted advertising, they uncovered a case of teen pregnancy before the teen's family were even aware themselves. As ethically controversial as this example may be, it is illustrative of the value that companies place on data. Fast forward half a decade, sprinkle in a handful of new ML algorithms, and the petabytes of data that has been collected gives rise to powerful new ways through which companies can influence consumers. From determining optimal store layouts and what gets end-of-aisle fame, to clustering collectively purchased items to make consumers spend more. From creating individualized promotions and coupons, to monitoring customer movements and predicting potential theft [4] machine learning is astonishingly pervasive, and it is accelerating.

Companies are undergoing a digital transformation. Our highly-interconnected economy has conditioned consumers to expect an infinite amount of choice, accessible remotely, and purchasable within minutes. How efficiently a company can develop algorithms to sift through the magnitudes of choice and select the few most suited to the individual consumer is one of the modern hallmarks for success. Pedro Domingos posits that *'as companies grow [...] after a point, there just aren't enough programmers and consultants to do all that's needed, and the company inevitably turns to machine learning'* [5]. In conjunction, Thomas H. Davenport argues that in the era of big data, where *'it would simply be impossible for humans to deal with all of this data without an automated process, organizations that can recognize and react quickly and intelligently have the upper hand'* [6]. Companies embrace ML because they are racing to create the best algorithms, where failure to do so means losing that advantage. *'They embrace it because they have no choice'* [5].

The imminent inevitability of ML adoption provides an opportunity to understand how this transition will affect business. As with all transitions, comes transition costs. Variations in the geopolitical landscape, economic environments, and even cultural differences have resounding implications on the rate and success of technological integration. Understanding the factors that influence this critical moment is an opportunity to mitigate barriers and enhance potential. As such, this research examines the social, political, economic, and cultural (SPEC) factors that affect how ML is being adopted by the architecture, engineering, and construction (AEC) industry.

2 Research Questions

In accordance to the purpose of this research – examining SPEC factors that influence the integration of ML in the AEC industry, and therefore understanding the cultural and organizational context of our profession – a series of questions were developed to direct the study and inform the objectives:

- To what extent has the AEC industry invested in ML research and development?
- What insights can be gathered from analyzing previous instances of ML research in the AEC industry?
- What are the potential SPEC factors that have affected the integration of ML in the AEC industry?

3 Objectives

As the AEC industry increasingly invests in ML research and development, social, political, economic, and cultural factors can both facilitate and hinder its success. If identical research projects were conducted in different locations, their outcomes would be heavily influenced by the disposition of their respective SPEC climates. Thus, one can argue for internal (within the organization) and external (the geospatial location of the organization) factors and the overarching objective of this research is to identify and analyze these instances. Consequently, the research has investigated ML research projects within the AEC industry to understand and derive SPEC factors involved.

To do so, this research intends to:

- Co-develop with a global AEC firm to systematically analyze their previous and current investments in ML (internal factors),
- Identify potential SPEC factors that might have affected ML research and development projects (external factors), and
- Examine how the respective research projects may have been influenced by the identified SPEC factors (externals effect on internal factors).

4 Methodology

The present research was carried out in partnership with Arup Engineering due to their global presence and extensive involvement across the AEC industry. Still, the objective of this research is to identify SPEC factors that aren't only specific to Arup Engineering as such but aims to be applicable across the whole industry. Arguing that Arup Engineering, with over 16,000 employees in more than 30 countries, in conjunction with Arup's well-established approach to encouraging and investing in employee ideas, Arup is an apt candidate for this study.

A systematic review – outlined in *Towards Evidence-Based Research* as requiring *'inclusion criteria, search methods, selection procedures, quality assessment, data extraction, and data analysis'* [7] – was conducted to assure valid and

accurate results. Focusing specifically on the *Invest in Arup* (IiA) database, this research started with a quantitative, inductive methodology, seeking to identify patterns and interpolate potential SPEC factors from Arup's research and development in ML. Beyond IiA, there are a number of ML initiatives at Arup that impacts ML culture, with many containing a significant research component. These works are worthwhile to acknowledge, however, as they are not documented within the IiA database, they fall outside the purview of this study.

The IiA database contains over 19,000 investment proposals. A preliminary extraction of those using ML revealed over 170 proposals at varying stages of development. However, further observations showed some to be misleading, incomplete and/or duplicates. Thus, a set of criteria was developed and employed to assure valid inclusion for this study, resulting in 105 proposals. The IiA proposals deemed relevant were spatially and temporally mapped to identify any emerging patterns that can be potentially attributed to SPEC factors.

To examine the validity of these factors, this research isolated a subset of proposals that was investigated further. Shifting into a qualitative, deductive methodology, project managers of the isolated subset were asked to partake in an interview series to discuss how the SPEC disposition of their office, country, and region affected their research. The intention of the interviews was threefold: to validate the factors gathered from the data analytics, to identify other factors undetected from the data, and to examine the impact of these factors to the integration of ML in the AEC industry.

5 Data Analytics

Using a systematic review methodology, the research concentrated its curatorial research process on the intersection of the key contributing fields. A preliminary search through the roughly 8,000 research and development projects in the *Invest in Arup* database revealed over 170 research proposals with an ostensible connection to ML. The connections could have been established as IiA proposals contain information about the project's aims, methods, and intellectual context. Due to the latter, false positives, namely proposals which mentioned ML peripherally, were a common occurrence. Incomplete, dormant, or duplicated proposals also emerged. To narrow the focus of our search and retain the integrity of our project scope, invalid IiA studies, specifically data points which were likely to skew the research analytics, were excluded. For this purpose, a set of criteria was developed to determine the suitability of a proposal. The systematic review revealed that only 105 proposals of the initial 170 directly incorporated ML, thus relevant for the second phase of the research. Two Python libraries (i.e. NumPy and Pandas) were utilized for scientific computing and high-performance data structures respectively. The data underwent a process of mapping, both spatially (i.e. at an office, country, and regional scale) and temporally (i.e. on a yearly basis).

Noteworthy in this context is the correlation between the employee count of an office and the number of IiA proposals originating from that office. Arup's

UKIMEA (UK, India, Middle East and Africa) region, for instance, has roughly three times more employees than other regions, resulting in a proportional increase in IiA proposals. Thus, comparing the number of proposals without taking in consideration other factors (such as an office's or a region's population size) will eventuate in a heavy bias (see Fig. 1).

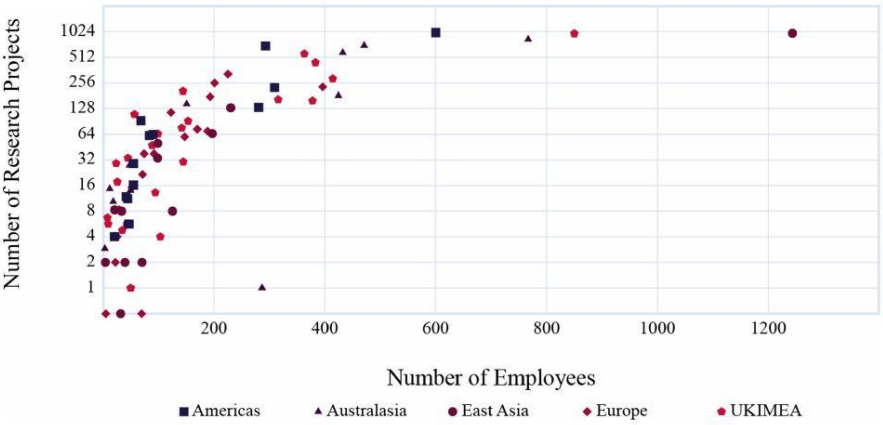

Fig. 1. The number of IiA proposals from Arup offices compared against the office population size and region. (Note: The London office, with an employee count of roughly 3000 and over 5000 IiA proposals, was intentionally omitted for greater clarity on the plot).

In light of this, descriptive and diagnostic data analysis was undertaken with the goal of identifying the potential SPEC factors that could affect ML adoption in the AEC industry. Results from the analyses led to formulating several observations.

First, when the allocated funding for approved ML proposals were averaged across the five regions, it was found that those in the East Asia region were provided with over 50% more funding than other regions (see Fig. 2).

A subsequent question emerged: Was the reason for this disparity an internal anomaly specific to Arup, or is there a larger force at play (a culture of techno-optimism or even government-level financial incentives)?

Second, the approval rate of all IiA proposals appeared to be proportional to those that were R&Ds, while showing a 15% increase when compared to those using ML (see Fig. 3).

In aggregate, ML proposals are more likely to be approved. But it goes further than that. From the analyses of approval rates at a regional level, the largest difference between the R&D proposals and the ML ones could be found in the East Asia and Australasia region. The Australasia region has an approval rate of 100%, which indicates that Australasia has yet to decline any ML proposals. From this second set of observations, a new consideration emerged: Does this

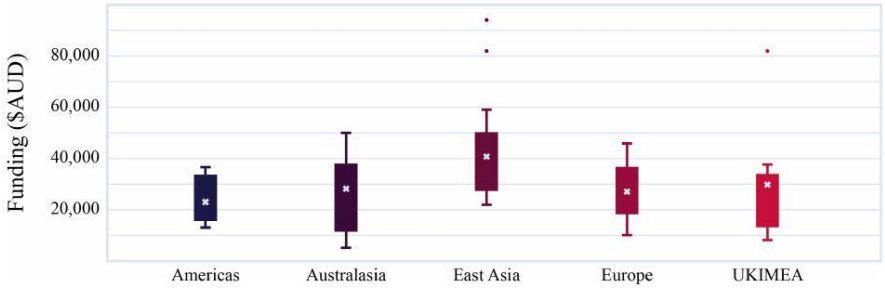

Fig. 2. The funding for machine learning IiA proposals per region.

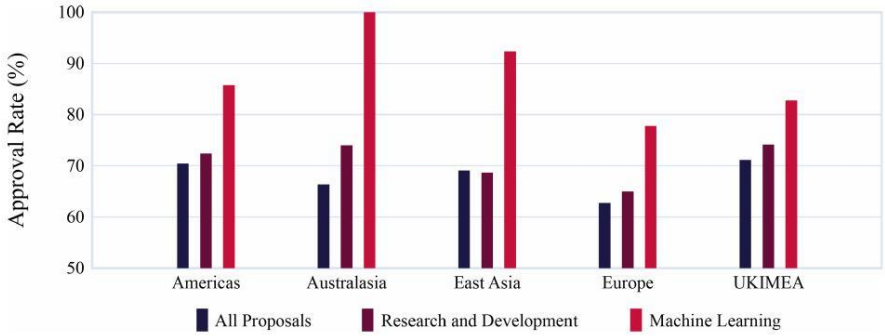

Fig. 3. The average approval rates for all IiA proposals, only R&D proposals, and only ML proposals per region.

imply a forward-thinking investment strategy or a blind pursuit of what author of *Artificial Unintelligence* calls, *'technochauvinism'* [8]? These two hypotheses were the catalyst for the investigations in the following section, where a series of interviews with Arup staff involved in these projects were conducted to identify and examine factors the data could not reveal.

6 SPEC Factors

Thirteen Arup employees, who had been directly involved with the identified ML R&D projects out of the IiA database, were contacted to speak about the insights gathered from the data analytics. The interviewees were based in nine offices across four regions and possessed varying levels of technical ML expertise.

A general structure was developed to direct the interview, while project and context specific questions were also pursued. By way of explanation, if the interviewee was based within the European Union (EU) and their proposal stated concerns about data collection or privacy, they were asked to comment on the implications of the EU General Data Protection Regulation (GDPR) implemented in 2018. Alternatively, interviewees based in Singapore or China were

asked if they experienced any influence from their respective government's AI strategies.

In analyzing and examining the results of the thirteen interviews, eight potential SPEC factors were identified and structured into three overarching categories: perceptions of ML; the communication of knowledge and education; and data governance and privacy. These are listed and outlined in the following sections.

6.1 Perceptions

Doer Resistance. Within the AEC industry, those that implement ML are also those that have shown the most resistance to its adoption. Architects and engineers have been taught how to approach existing problems, emboldening and validating their use of current methods, and in turn, building a resistance to new ones. This leads to the industry's resistance technological adoption, on par with that of the agriculture and transportation industries [9–11]. Due to the considerable distance between the fields of computer science and architectural engineering, and the tendency towards Occam's razor – a philosophical principle stating that the simplest solution is probably better – the doers cause the most immediate form of resistance to ML adoption in the AEC.

An interview with an Arup employee working in digital transformation revealed a substantial willingness at the managerial level to invest in new technologies. The interviewed stated that despite the evident long-term profitability of ML, the doers still show a fundamental resistance to its adoption. Architectural engineers have a toolbelt of programs and software that match up to a list of problems they face in their work. This rigidity facilitates a psychological attachment toward that toolbelt of proven successes. The interviewee further states that the doers more open to ML still embark on their own standards of testing to validate the accuracy of the new ML methods. This is inherently detrimental for ML adoption, as ML processes favor speed and efficiency over accuracy. During an interview with a key proponent for ML processes in the AEC, they stated that a new approach to problems requires *'180-degree thinking'*, and that *'for digital transformation to occur, it takes a certain amount of unlearning before learning'*. This is echoed in a 2017 McKinsey report, that argues the following: to leverage full digital opportunity requires a *'shift [of] cultures and ways of working rather than just systems and tools'* [11].

Further, the field of computer science, wherein ML sits, is of considerable distance from that of architecture. As such, architects are not trained, nor expected, to have the required skills to implement ML. Combined with its steep learning curve, using ML becomes a daunting task. In parallel with Occam's razor, Domingos argues that *'simple theories are preferable because they incur in a lower cognitive cost (for us) and a lower computational cost (for our algorithms)'* [5]. The doers are resistant to ML due to an attachment to current methods and their proven successes, in combination with the sheerness of ML's learning curve. Thus, significant consideration for more effective ML adoption in

the AEC should be placed towards how ML can be integrated rather than what ML can do.

Misconceptions and Funding. Despite management's willingness to invest in ML, how technology is perceived greatly affects its implementation. As with any technology that dominates the public eye, over-hype and misconceptions run rampart. Leadership positions rarely contain *'digital natives'* and misconceptions can lead to an *'uncertain[ty] about what exactly AI can do for them'* [2]. Any resistance to the allocation of funding compounds resistance to the application of ML. Further, even when funding is reduced, expectations remain that lead to inevitably underwhelming results and, as a consequence, a disincentivization of future implementations.

When ML misconceptions underpin business decisions and goals, expectations and results are pushed further apart. A 2017 McKinsey report highlights a lack of *'digital natives'* in leadership positions [11] which is a major contributor to this problem. Any misconceptions at the managerial level exasperates resistances felt by those conducting ML's implementation.

Arup's R&D application process asks applicants to request a specific amount of funding. This could be altered if deemed necessary after receiving Arup's approval. Of the 64 ML-related projects that have been funded, 37 received more or on par with the funding they requested. This leaves 27 projects that have had their funding reduced, with an average reduction of 46% compared against the requested funding (see Fig. 4).

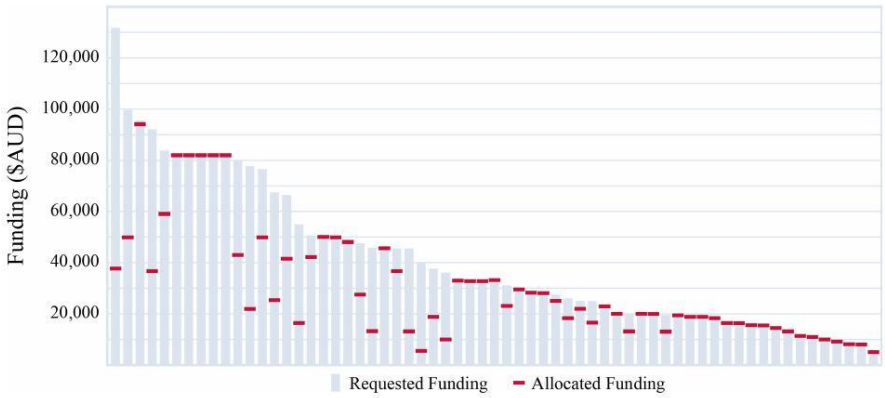

Fig. 4. The difference between requested and allocated funding for the 64 ML projects that received funding.

During the interview series, five interviewees made statements relating to how the scope of their projects were severely limited due to the reduction of funding. One interviewee asserted that the reduction led *'to a simplification of*

the process', while a second one stated that the reduction turned *'an initially global project into a local one'*. Regardless of whether expectations are bred from misconceptions over ambitious scoping, or a lack of research – the disparity between managerial expectations with what the project team can deliver may lead to unmet goals and has the potential to disincentivize future ML research.

Governmental Strategies and Incentives. The impact of ML extends well beyond firms, and as governments catch onto its potential value, several nations have developed and implemented AI/ML strategies to capture this potential. Minute differences in these policies – such as their focus, goals, and the speed at which they were released – may have resounding implications for the implementation of ML in the AEC. Further, as these strategies are developed for a national scale, multinational companies may undergo disproportionate levels of incentives, and ML investment may proceed at different rates. Conversely, these strategies may have little influence on the AEC industry, as they target "digital" firms. However, the cultural significance they instill encourages its investment, as it is demonstrated by the following the data.

The varying perceptions and values governments place on ML is visible through an examination of their AI strategies. With foci ranging from scientific research to ethics and inclusion, certain countries vocalize their ambition to lead the world in AI, while others favor future-proofing their economy. These vast differences have the potential to affect multinational companies, by influencing company goals set by different regions, countries, and offices. Despite all 13 interviewees claiming that there was no direct effect any AI strategy had on their ML-related projects, a correlation between the more aggressive policies with greater funding was evident. China, Singapore, and Japan were part of the earliest countries to have released national AI strategies. The eagerness shown from the East Asia region is similarly echoed as the first two explore ML at Arup, where, in 2012, a team from the Shenzhen office explored how ML can be used to intelligently control the actuation of windows.

Despite how the UKIMEA region has since developed the largest quantity of annual ML IiA proposals – potentially a by-product of the region's comparative population size (see Fig. 5) – the East Asia region commits the largest average financial investment of all the regions by a significant margin (see Fig. 2). Further, comparing the 24 funded ML-related projects in 2017 against the 28 in 2018, the East Asia region was the only region to have an increased average of allocated funding, with an increase of 14.45%. Finally, the approval rate for the East Asia region is second only the Australasian region. It is evident that the East Asia region has a heightened willingness to invest in ML, however, it is ultimately unknowable if the AI strategies directly influenced their successes.

6.2 Communication

Digital-Focused Communities. Firms in the AEC are not traditionally considered "digital" firms, thus digital standards and practices are not yet a priority.

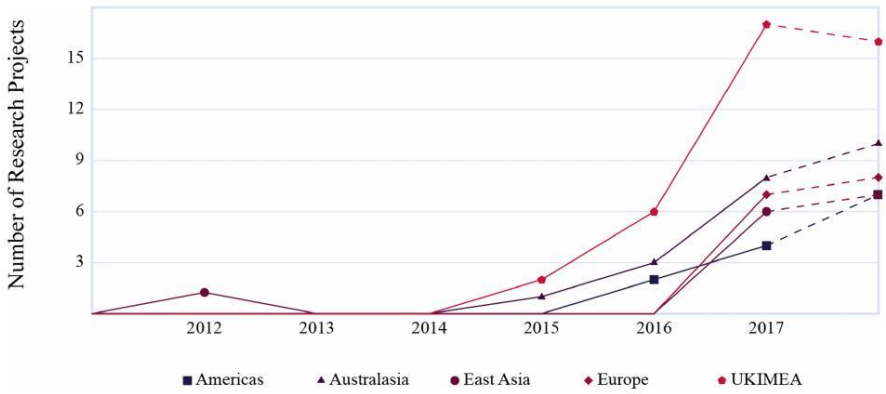

Fig. 5. The number of ML proposals per region per year. (Note: The data for 2018 is only partially complete as the analysis was conducted before the end of 2018).

Key minds in the field of computer science and ML have continuously introduced new approaches in the form of different programming languages, frameworks, and algorithms, each one of with their own nuances and intricacies. As favor and popularity varies over time, learners will be introduced to ML with different preferences and opinions. Furthermore, lacking a central location where an internal community of ML-enthusiasts in AEC firms can communicate and share knowledge and code can lead to overlapping work, incomprehensible code, and incompatible communication.

As ML is still within its infancy in the AEC, Arup has conducted a greater amount of ML research and development compared to incorporating ML in client-facing work. As such, Arup has yet to apply company-wide digital standards and practices (though they are currently in development). One aspect of ML that is vital to standardize is the preference of ML frameworks. TensorFlow, a low-level Python ML framework developed by Google, has garnered the most popularity in the ML community – popularity measured by job posting requirements, usage surveys, GitHub activity, and publication mentions [12]. However, the Arup interviews revealed that very few ML developers were using TensorFlow in favor for less popular frameworks such as CNTK and MXNet. The disparity between Arup's ML developers and the greater ML community has the potential to cause friction in its adoption. The more popular a framework, the more likely it will be maintained and the greater number of resources, both factors that can ease ML adoption. Arguably, the granularity of TensorFlow is not necessary for the applications within the AEC, and that a higher-level framework such as Keras or Caffe would suffice, however Arup developers have yet to reach a consensus. Furthermore, during an interview with an Arup developer, there was mention of a neural network architecture, called "ShuffleNet" [13], being a standard across Arup. However, the other interviewees have not even heard of this architecture. Small differences and inconsistencies, due to a lack of an internal

ML community, is a barrier for effective communication, leading to incompatible code and reduced comprehension.

Dissemination is a big focus for Arup research, however, as stated in an interview, *'just because a model was disseminated, doesn't necessarily mean it's being used'*. At Arup, company-wide dissemination can occur through several channels; namely skills networks. However, Arup lacks a single central repository for trained ML models to be shared and reused. For instance, a ML research project based in the Dublin office used local traffic data to train a ML model for the prediction of traffic congestion after accidents. The model itself would be useless in other locations, however, disseminating the code itself in a central, easily-accessible repository would allow others to modify the training data to train their own local model. What is required is not simply bottom-up innovation, but also a systematized top-down initiative and structure. Without this, what can be expected is, at best, overlapping work and wasted time, and at worst the stagnation of ML integration in the AEC.

Expertise. The AEC is an incredibly multidisciplinary industry, requiring a lot of different domain experts to collaborate on projects that can span several years, and even decades. With expert knowledge becoming more and more specialized, effective communication between fields has become vital. However, due to ML's steep learning curve, and the differences between the architectural engineering and computer science fields, communication when integrating ML in the AEC can become tenuous. Furthermore, with the development of ML-based tools, intended to be used by non-experts, developers not only need to train a robust ML model, but also present it in an easily-accessible, user-friendly interface, for greater ML uptake.

Effective ML models often require problem domain knowledge. More robust ML models can be developed with data validation, data normalization, feature engineering, and edge cases addressed by the domain expert. However, in the age of splintered expertise, rarely is the ML engineer and the domain expert the same person. Thus, effective communication between the parties is key. *'High-proximity, high-bandwidth communication can happen relatively easily in a small start-up firm with only a few employees. It is of course more difficult in a large firm'* [6]. The impact of communication is evident through the examination of two PhD students conducting ML research with Arup. One student, a mathematician, received a comparatively low "objectives/deliverables" rating than that of the other student, a software developer, with their research reported as *'slow progressing'*. When questioned about the cause of the disparity, the candidates' supervisors stated that the field of mathematics is a further step removed from the built environment than that of software development. This added an unforeseen strain to effective communication. Moving forward, *'roles that companies need to fill are the "translators" and data scientists ("quants")'* [2]. *'However, the specific percentage of projects that fail is questionable, there is no doubt that lack of communication in small and big data projects causes big problems'* [6].

The translation of a digital tool or service into one accessible by a non-expert audience is an important process seemingly lacking in the AEC industry. An interviewee recounted a completed research project that yielded a robust and accurate ML model, however, after its dissemination, it was found that the tool itself was not being used. The interviewee attributed this failure to the lack of an interface; they did not make a front end, and so *'it wasn't used in the office'*. On two ends of the spectrum, Arup has the ML experts to train good models, and the engineers who will use the tool, however there is a lack of software developers to bridge the gap between the tool and the users. This ultimately harms the integration of ML in the AEC industry.

Education. ML's climb in popularity has garnered a plethora of formal education, i.e. university courses [1], in addition to informal resources, such as online tutorials and blog posts. Because ML has flourished only recently, a considerable amount of ML developers were not formally taught ML and have gained their ML knowledge from autodidacticism. However, the culture of self-education may lead to incongruencies in ML knowledge as the informal resources of knowledge can be opinionated, outdated, or even invalidated. And with the plethora of resources available, this leads to a fragmentation of knowledge and a difference of opinions with standards and practices.

Some perceive ML as a magic solution, whereby premade algorithms and pretrained models can simply be taken and applied to any problem. With this mentality, understanding the corresponding programming language is all that's required to produce results. However, to effectively train a robust ML model to a high degree of accuracy, a considerable understanding of regularization techniques, hyperparameter tuning, data engineering, and normalization are required. An out-of-the-box solution is inherently limited when shoehorned into more complex problems. An example of this can be seen in an early Arup project exploring ML, where the model achieved a maximum accuracy of 60%. The disseminated research documentation revealed that a standard ML architecture was used without any hyperparameter tuning. This was the main reason for below average results. Inconsistent and incomplete ML education acts as a barrier for ML integration, however, Arup has taken steps to standardize and formalize ML developers' education. On multiple occasions, Arup has partnered with universities to allow Arup employees in ML-focused workshops and courses, in addition to the dissemination of validated educational resources and the development of its own curriculum.

A McKinsey report revealed in an investigation into digital transformation that *'many companies are attempting to make use of the large volume of unstructured data at the disposal, but do not have the data science and data engineering capabilities to generate game-changing insights'* [11]. While Arup is reasonably supported by internal engineers with data science expertise, two interviewees mentioned a lack of ML expertise within specific subdomains of ML, forcing the team to seek experts in other offices.

6.3 Data

Data Governance. The accumulation of data is accelerating. *'Of the 3.5 trillion photos that have been snapped since the first image of a busy Parisian street in 1838, fully 10 percent were taken in the last year'* [14]. With new methods of data analytics and algorithms that can convert vast quantities of data into valuable insights, the governance of data is a question that will become increasingly discussed. Due to the requirement for data in the field of ML, data governance has the potential to greatly determine the shape of ML integration in the AEC.

An often-overlooked step in the process of ML research is the collection of data. And investigation reveals that a lot of time and money is wasted in this process. Following a developed risk classification schema, of Arup's 76 ML projects that had stated potential risks, and second only to the availability of their team, the most prevalent risk identified was the collection of data. When projects were proposed, more teams stated that they were concerned about the collection of data, than about the quality and quantity (see Fig. 6).

Fig. 6. The prevalence of risks indicated by 76 ML proposals in accordance to the developed classification of risks. (Note: 29 proposals indicated no risks).

As the documented risks are completed during the application process, these risks are speculative. Unfortunately, concerns about data collection were founded. When interviewees were asked about the issues of data collection, four voiced concerns about its governance, one stating that *'there is an ongoing issue on who owns the data'*. The sheer quantity of data needed to adequately train a ML model required another team to source their data from multiple third parties, thus requiring *'a lot of time cleaning the data'*. Davenport poignantly states that *'since big data is often external to the organization, governance of it is often a tricky issue. Data ownership is much less clear, and responsibilities*

for ongoing data management haven't been defined either in most cases. We're going to be wrestling with big data governance for quite a while' [6].

Data governance is a web of boundaries that become increasingly difficult to draw as parties undergo constant data revaluation. *'As the economy has changed so, too, must our metrics. More and more what we care about in the second machine age are ideas, not things – mind not matter; bits, not atoms'* [14]. And in a world where data is virtually infinite, and ML algorithms are vying for beyond-human level performance, the question of how data is valued is brought to a head. *'The basic tenet is that the world – and the data that describes it – are in a constant state of change and flux, and those organizations that can recognize and react quickly and intelligently have the upper hand'* [6]. Because of this instability, Arup has faced instances of third parties denying them data. An interviewee described a project focused on geotechnical analysis that received only a fraction of the initial training data from a third party. Another interviewee recounted a project that required a vast amount of tree species data, yet the universities they were working with no longer wanted to share their data. Those in possession of data are starting to reassess what it's worth, which acts as resistance for successful implementation of ML in the AEC.

Data Privacy. The rapid rise of data accumulation, in combination with a plethora of new algorithms, have made way for powerful predictive models. A common statistics aphorism states that *'all models are wrong, but some are useful'*, with some so useful, that their uncanny accuracy has sparked concerns about data and privacy. *'New resources – whether a better spyglass or print or a new method – extend not only what is known but, crucially, the range of what can be known'* [15]. Instances like the case of Andrew Pole and *Target*'s pregnancy predictor, mentioned in the introduction, revealed the benefits of data for companies, but also has amassed a public outcry concerning the power companies wield. Outcries have ignited responses from policy makers to enact restrictions. Any limitation to the accessibility of data is ultimately detrimental for ML, as ML models are only as effective as the data, creating barriers for the use of ML in the AEC.

Despite some governments advocating for more open data, others are limiting and disincentivizing it for fear of data privacy infringement. The governments of Germany, Denmark, India, Japan, and Taiwan have released AI strategy plans that dedicates a portion to *'policies aimed at promoting data access, as well as their circulation and sharing'*, an excerpt taken from France's *For a Meaningful Artificial Intelligence: Towards a French and European Strategy* [16]. However, strict regulations and heavy fines have been enforced on the misuse of personal data, a severe deterrent for ML research. Most notable is the European Union's General Data Protection Regulation (GDPR), with companies set to receive fines of up to four percent of global revenue or twenty-million euros. An interview with an Amsterdam-based employee revealed an ongoing research project that has been stalled due to concerns about the GDPR. The interviewee stated that the external partner for the research project *'is being more difficult,'* as they

'don't want data to leave the Netherlands'. Data's importance for ML combined with new regulations acts as a barrier for the successful implementation of ML.

7 Conclusion

The purpose of this research was to identify and examine potential SPEC factors that influence the adoption of ML in the AEC industry by:

- Co-developing with a global AEC firm (Arup) to systematically analyze their previous and current investments in ML,
- Identifying potential SPEC factors that might have affected ML research and development projects, and
- Examining how the respective research projects may have been influenced by the identified SPEC factors.

To reiterate, the following conclusions have been shaped by conversations with a subset of those from one AEC firm. Further research – analysis of ML innovation outside of formal research databases, conversations with more project managers and technical experts, and investigations in more than one firm – is require to capture a more holistic insight into the state of ML adoption across the entirety of the AEC industry. Upon reflection of the objectives set at the beginning of the paper, one can conclude that out of the eight SPEC factors that were identified and structured into three overarching categories – perceptions of ML, the communication of knowledge and education, and data governance and privacy – there are two SPEC factors that cause the greatest effect on ML adoption, but conversely can be controlled.

As discussed in the subchapter *Doer Resistance*, the hesitancy from those applying ML poses an immediate problem. This is compounded when the desire to validate the accuracy of an inherently inaccurate model yields a misleading underperformance. An interviewee, described as a *'key progenitor and proponent for the shift in ML-based research'*, asserts that the underlying cause can be attributed to a rigidity in the current *'approach'* or *'mindset'*. They call for doers to undergo a *'complete 180-degree turn in thinking'*, and places incredible importance on the process of *'unlearning, before learning'*. Moving toward a greater degree of computer-aided work, cognitive flexibility and agile thinking are skills that are increasingly valued. The more willing those are to adopt machine intelligence into their workflows, the greater their potential output.

From the perspective of AEC firms, the facilitation of ML adoption starts with the development of a supportive digital ecosystem. The subchapter *Digital-Focused Communities*, the second pertinent SPEC factor, emphasizes this. Pragmatic standards and practices, central and accessible repositories, and a common platform for open and informal communication fosters a culture of ML-driven innovation. *'At the speed at which innovation and disruption are taking place, [the] world of data requires new forms of collaboration'* [17].

Returning to Pedro Domingos and his views on the ML-driven future, *'when someone talks about exponential growth, ask yourself: How soon will it turn into*

an "S" curve?' [5]. Many theorists claim that we currently sit at the inflection point of the exponential rise of ML, and that a plateau is on the horizon. During this transition period, it is of vital importance that companies take advantage of the current upwards gradient and mitigate any hindrances (Fig. 7).

'The future belongs to those who understand at a very deep level how to combine their unique expertise with what algorithms do best' [5].

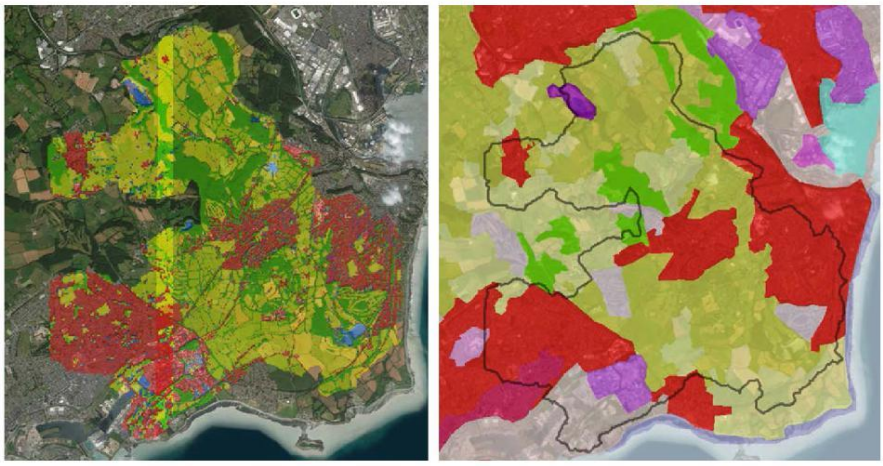

Fig. 7. High-resolution satellite imagery segmentation (left) with the existing equivalent (right). An example of Arup's ML adoption on AEC problems, taken from one of Arup's 2017 Global Research Challenges: *'Unlocking the Potential of Natural Flood Management with Machine Learning'.*

Acknowledgements. A huge thank you to the thirteen Arup interviewees who put aside time to share their knowledge and experience. Despite only a few being directly quoted, each conversation helped shaped the main ideas that drove this research. Thank you to Giulio Antonutto, Steven Downing, Veronika Heidegger, Jorke Odolphi, and Alvise Simondetti for sharing their experience with technological integration at Arup. Further, a big thank you to those in Arup University – Bree Trevena, Alex Sinickas, Esther Wheeler, and Kim Sherwin – without whom this research would not have been possible. Finally, thank you to both Arup Engineering and the University of New South Wales for their continual support.

References

1. Artificial Intelligence Index: 2017 Annual Report. Artificial Intelligence Index. http://cdn.aiindex.org/2017-report.pdf. Accessed 12 Dec 2018
2. Bughin, J., et al.: Artificial Intelligence: The Next Digital Frontier? McKinsey Global Institute, New York City (2017)

3. Duhigg, C.: The Power of Habit: Why We Do What We Do in Life and Business. Random House, Great Britain (2012)
4. De Jesus, A.: Artificial Intelligence in Groceries and Produce. Emerj. https://emerj.com/ai-sector-overviews/artificial-intelligence-in-groceries-and-produce-current-applications/. Accessed 14 Dec 2018
5. Domingos, P.: The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World. Basic Books, New York City (2015)
6. Davenport, T.H.: Big Data at Work: Dispelling the Myths. Uncovering the Opportunities. Harvard Business School Publishing Corporation, Boston (2014)
7. Lund, H., et al.: Towards evidence based research. Br. Med. J. **355**, i5440–i5445 (2016). https://www.bmj.com/content/bmj/355/bmj.i5440.full.pdf
8. Broussard, M.: Artificial Unintelligence: How Computers Misunderstand the World. The MIT Press, Cambridge (2018)
9. Manyika, J., et al.: Digital America: A Tale of the Haves and Have-Mores. McKinsey Global Institute, New York City (2015)
10. Bughin, J., et al.: Digital Europe: Pushing the Frontier. Capturing the Benefits. McKinsey Global Institute, New York City (2016)
11. Blackburn, S., Freeland, M., Grätner, D.: Digital Australia: Seizing the Opportunity from the Fourth Industrial Revolution. McKinsey Global Institute, New York City (2017)
12. Hale, J.: Deep Learning Framework Power Scores 2018. Towards Data Science. https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a. Accessed 14 Dec 2018
13. Zhang, X., Zhou, X., Lin, M., Jian, S.: ShuffleNet: an extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6848–6856 (2018)
14. Brynjolfsson, E., McAfee, A.: The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technology. W. W. Norton & Company Ltd., New York City (2014)
15. Siskin, C.: System: The Shaping of Modern Knowledge. The MIT Press, Cambridge (2016)
16. Villani, C.: For A Meaningful Artificial Intelligence: Towards a French and European Strategy. AI For Humanity. https://www.aiforhumanity.fr/pdfs/MissionVillani_Report_ENG-VF.pdf. Accessed 19 Dec 2018
17. Schwab, K.: The Fourth Industrial Revolution. Crown Publishing Group, New York City (2016)

Learning Machine Learning as an Architect, How to?

Presenting and evaluating a Grasshopper based platform to teach architecture students machine learning

Nariddh Khean[1], Alessandra Fabbri[2], M. Hank Haeusler[3]
[1,2,3]University of New South Wales / Computational Design
[1,2,3]{n.khean|a.fabbri|m.haeusler}@unsw.edu.au

Machine learning algorithms have become widely embedded in many aspects of modern society. They have come to enhance systems, such as individualised marketing, social media services, and search engines. However, contrasting its growing ubiquity, the architectural industry has been comparatively resistant in its adoption; objectively one of the slowest industries to integrate with machine learning. Machine learning expertise can be separate from professionals in other fields; however, this separation can be a major hinderance in architecture, where interaction between the designer and the design facilitates the production of favourable outcomes. To bridge this knowledge gap, this research suggests that the solution lies with architectural education. Through the development of a novel educative framework, the research aims to teach architecture students how to implement machine learning. Exploration of student-centred pedagogical strategies was used to inform the conceptualisation of the educative module, which was subsequently implemented into an undergraduate computational design studio, and finally evaluated on its ability to effectively teach designers machine learning. The developed educative module represents a step towards greater technological adoption in the architecture industry.

Keywords: *Artificial Intelligence, Machine Learning, Neural Networks, Student-Centred Learning, Educative Framework*

INTRODUCTION

Over the last decennial, machine learning (ML) has enhanced key computational processes in almost all economic sectors. Its early adoption offers a powerful drive for innovation and has the potential to augment optimisation, automation, and prediction problems (McKinsey Global Institute 2017). Research suggests that modern ML breakthroughs will be as transformative as electricity a hundred years ago (Brynjolfsson, McAfee 2017). However, despite the advancements and successes of ML across several fields, there is an inherent lack of ML integration within the creative industry. Only sparingly have ML algorithms been employed in the domain of architecture and design. While it is highly promising that ML will play an increasing role in the industry (Kaplan

2016), architecture itself seems to disregard its potential for resolving prediction problems, categorising vast quantities of data, and modelling for optimisation and advanced form finding (Carpo 2017). This research believes that to resolve the disparity between architecture and their adoption of emerging technology, architectural academic programs should incorporate ML into their educational strategies. Accordingly, this research proposes an educative framework for a new generation of designers, architects and urban designers willing to invest in the capacities of ML models for performative design prediction.

RESEARCH AIMS

The argument could be made that architecture has traditionally been a discipline almost totally devoid of the rigorous data analysis. However, data has increasingly become an interactive design parameter. It can be collated from the surrounding, analysed, manipulated and evaluated in the design process, and in some cases, visualised through the final product. In recent years, research efforts have produced a wealth of computational tools for data-driven design useful for a range of applications. However, developing frameworks for data-driven design has continued to depend on a combination of experience, intuition, and manual knowledge building and retrieving. The current research proposes to close this design-optimisation gap by educating the new generation of designers on computational prediction through machine learning.

Our overarching objective is to contribute to the optimisation of design-finding for architecture with respect to functionality, sustainability, and liveability. As such, the initial objective is to educate computational designers, and potentially architects, on the mathematical and performative principles of ML algorithms and to challenge the pre-existing impenetrability of this branch of computing. ML models are generally developed and operated by specialised engineers, at ease with the prerequisite programming knowledge and capable of embedding them in languages such as Python, R, or MatLab - both arguably beyond the usual capacities and understanding of most architects. Whereas this is not a concern in other industries, it can represent a challenge in the architectural industry, where a dynamic process of interaction between the designer and the design ensures the production of bespoken and highly performative projects. One might argue that the design industry has already witnessed the emergence of ML frameworks within Grasshopper (e.g. *Ant*, *Owl*, or *Lunchbox ML*). However, these instances comprise of enclosed components which offer little to no introspective quality. As such, unless the computational designer is also trained in ML, it's incredibly difficult to yield valuable results, let alone anything beyond useless. The research team proposes to remedy this disparity through the development of an educative framework voted to complete transparency.

The research aims are threefold: to conceptualise an educative framework aimed at training designers machine learning; to implement the framework in a live educational environment; and to evaluate the effectiveness of the framework in and beyond the architecture classroom.

RESEARCH QUESTION AND OBJECTIVES

Considering the foregoing arguments, this research aims to investigate and resolve the following key research question:

What strategies could be utilised to effectively teach computational designers about machine learning?

The team intends to resolve the research question by addressing the following research objectives:

1. What implementation and training do ML algorithms require to facilitate the design-finding process?
2. To what extent can the system be used to output a geometrical shape?
3. To what extent can the system be also incorporated into the built shape?
4. At a broader scale, to what extent can ML be employed within the built environment industry?
5. Are there any ideological, theoretical, and practical arguments arising from the applica-

tion of a scientific tool in a creative industry?

METHODOLOGY

An applied research methodology was adopted throughout the investigations, while core concepts within case studies and survey-based results were embraced. The research was divided into three subsequent stages: initial conceptualisation of the educative framework through theoretical and practical research; implementation of that framework in an architectural educational context; and evaluation of its effectiveness as a teaching method.

Firstly, the conceptualisation and development of a preliminary ML educative framework for computational designers and architects was conducted through the foregoing applied research methodology, with the goal of delineating its real-world problem-solving implications.

The framework was then informed by a comprehensive literature review, with the goal of identifying bespoke teaching strategies and pedagogies through a design educational lens. An extensive understanding of learning outcomes rationale and student requirements was gained through theoretical research and practical academic experience. The comprehension of design processes and students' educational levels was fundamental to underpin the framework development. Aligning the teaching, learning, and assessment activities with learning outcomes and graduates' capabilities at program and course levels was a key driver for the initial framing of strategic learning components.

Finally, the implementation of the framework was conducted through a case study methodology. Its execution was predicated on successful teaching methods, found in previously conducted and documented educational case studies. Further, unstructured student interviews were continuously carried out throughout the teaching. These surveys were crucial to distinguish general limitations of the coursework from singular difficulties encountered by students due to their individual academic backgrounds. This facilitated rapid adjustments in the teaching approach, further enhancing the clarity and effectiveness of the framework.

LITERATURE REVIEW

The transition into the new millennium saw a revolution in how academics approach education. A powerful concept emerged, revolutionising "the philosophy of how one teaches, [and] the way in which a classroom is structured" (Norman and Spohrer 1996). 'Student-centred' education, as opposed to prior 'teacher-centred' ideals, embraces the notions of constructivism, collaboration, and technology, to encourage 'active learning' (O'Neill and McMahon 2005). Diverging from the 'teacher-centred' approach, whereby a single purveyor of knowledge lectures to a 'passive' classroom, 'student-centred' education interactively fostered the collective augmentation and dissemination of knowledge. On the spectrum between 'teacher-' and 'student-centred', a plethora of pedagogical approaches exist. For the purpose of this research, inquiry-, project-, and resource-based learning techniques were assessed on their potential benefits for the proposed educative framework.

Inquiry-based learning enables students to experience knowledge creation and can be described as a student-centred, self-directive, and active approach. This learning approach is classified as an 'inductive' method, due to its progression from a set of observations or a complex real-world problem towards the generation of a need for facts, procedures, and guiding principles (Prince and Felder 2006). In the context of computational design education, inquiry-based learning exposes the students to the "unvarnished realities of the highly competitive, and often-contentious, world of design and architecture" (Barron and Darling-Hammond 2008). Furthermore, this method compels the student to independently acquire fundamental skills to meet the multifaceted requirements of a design brief. As such, the research team deems the inquiry-based learning pedagogy a potentially advantageous approach to achieve our research objectives.

Project-based learning is an educational model

that organises learning around an overarching objective, frequently proving to be an incredibly motivational and engaging method of learning (Blumenfeld et al. 1991). A complex set of "tasks, based on challenging questions or real-world problems, that involve students in design, problem-solving, and investigative activities" could represent the 'project'. Students would be motivated to work relatively autonomously and culminate in realistic products or presentations. Design education benefits greatly from project-based learning, as it expects an understanding of how to frame a problem, identify stakeholders and their requirements, and apply theoretical concepts in practice. As with the majority of higher education design curricula, this research aims to similarly employ a project-based learning pedagogy.

Resource-based learning places the interaction between learner and resource as the main structuring device of the learning situation, and foregrounds the usage of technology to reinforce learning (Brown and Smith 2013). The metamorphosis of information systems has shifted educational resources from static texts, to a multimodal collection of "tools for acting on and with, and scaffolds to guide differentiated interpretation" (Hannafin and Hilll 2007). Especially within the field of computational design, where technological processes drive design outcomes, the resources required to learn those processes need to: be flexible, allow multiple literacies, and support different levels of adaptability, proficiency, and intelligence.

The investigation of various educational ML resources culminated in the understanding that they are mainly aimed at an audience proficient with linear algebra, and/or language-based programming. Most methods for ML application unfold through 'text-based' programming languages such as Python and C++. However, the computational design audience, whose computer science curriculum often ceases at visual programming, might find the learning curve for most ML resources too steep. In 2018, an educational ML tool was developed in a 'visual' programming environment (Grasshopper) to ease this learning curve. The tool, generated as part of a short-term research project, reconstructed a neural network algorithm with only basic mathematics and data tree manipulation components (except for a plugin to circumvent Grasshopper's recursive loop avoidance check) and aimed to "unpack the complexities of machine learning in an intermediary software environment" (Khean et al. 2018). However, the scope of their research prevented the team from implementing the tool in a live educational setting. Therefore, we propose to employ it for the proposed educative framework as the centre of a resource-based learning pedagogy.

LEARNING COMPONENTS

With the abovementioned educational tool at the core (Khean et al. 2018), our research proposes an educational framework that encourages active learning through student-centred approaches. The framework utilises key concepts in inquiry-, project-, and resource-based learning pedagogies, to inform a design students' academic curriculum. The ML educational framework constitutes a foundation of solid skills and knowledge for students to build upon and is beneficial to a wide range of design applications. After a brief analysis of existing educational modules focused on computational intelligence, the framework was structured around four major learning components:

Paradigms of Computational Intelligence: Artificial Neural Networks

Introducing the notion of computational intelligence, the first teaching component offers an overview of soft adaptive computing to facilitate intelligent behaviour in complex and evolving environments. Students are introduced to supervised, unsupervised, and reinforcement learning, as well as their fundamental differences, leading to an emphasis in the field of Artificial Neural Networks (ANN). At the successful completion of the first session, students are expected to comprehend and interact with complex mathematical concepts.

Paradigms of Dataset Pre-Processing and Analysis

The importance of collecting a clear, flawless, and extensive data set is introduced and elaborated over the second teaching component, while discussing the nature of a logistic regression model. Focusing on binary classification and dimensional matrix organisation, students are guided in generating, labelling, and implementing a training dataset through the correct computational convention. At the successful completion of the second session, students are expected to demonstrate proficiency in dataset analysis through data pre-processing completion.

Paradigms of Training and Evaluation

As the performance of artificial neural networks improves by increasing the training dataset and the depth of the ANN itself, assessing the model's performance becomes fundamental to improve its accuracy. The third teaching component investigates the nature of feedforward, error calculation, and backpropagation through gradient descent while guiding students to a model's parametric implementation in Grasshopper. At the successful completion of the third session, students are expected to logically and critically evaluate the design through observing neural network error graphs and high-dimensional prediction visualisations.

Paradigms of Spatial Representation

Design-optimisation and computational prediction are consolidated in the fourth and last teaching component, where the students' creativity is assessed in parallel to their critical thinking in employing the neural network model for three-dimensional shapes generation. At the successful completion of the fourth session, students are expected to augment their technical and evaluative skills.

In 2018, the learning components were consolidated into a cohesive four-week curriculum, and implemented in an undergraduate computational design studio at the University of New South Wales in Australia. The graduation year programme provided an apt setting for both the implementation and critical evaluation of the module's effectiveness. While investigating the possibility to design and fabricate a kinetic biomimetic pavilion, adaptive to real-time information collated from the surroundings, students developed their own ML prediction models to analyse the datasets and predict optimised spatial configurations for the pavilion.

INTERVIEW STRUCTURE

After the implementation of the educative curriculum, the research team conducted a series of interviews with the participants involved in the process (students and staff) to extract the collective opinions on various aspects of the curriculum, including the workshops, teaching strategies, and resources. There was a total of 13 students and 2 tutors who partook in the interviews, all with varying interests, experiences, and knowledge with ML in computational design. Table 1 specifies the questions that directed the interviews.

EVALUATION

The anticipated outcomes for partaking in the developed curriculum was an aptitude for training, evaluating, and implementing Khean et al.'s educational resource into the design of a small scale original pavilion. The interview series resulted in a general observation that the educational framework didn't provide the mathematical background required to understand and implement the tool. Specifically, 15% of the students claimed that the framework clearly demonstrated how to apply the tool, but they failed to grasp the purpose and nature of each scripting component. Ostensibly, the purpose of the curriculum, and therefore of the educational framework, was not clarified, thus leading to some misconceptions. The overarching intention was to educate students on how to apply the machine learning algorithm for predicting temporal changes in the design process, not to become an expert in the mathematical principles involved in this operation. In fact, when asked if they had autonomously explored the mathematics behind the ANN script, only 8% of the students answered positively. Given the short-term nature of the curriculum (only four weeks), the faculty might not be able to incorporate additional teaching

Table 1
Post-Educational Framework Interviews

Section	Interviewee	Question
Theoretical Background	All	1. What is/was your area of study? 2. How extensive is your knowledge of algebra and matrix theory? 3. How extensive is your knowledge of multivariate calculus? 4. How you heard of ML prior to the module? 5. How extensive is your expertise in ML?
ML Experience	All	1. Do you have any prior involvement with a project that has used ML? 2. What ML algorithm was used? 3. What programming language was used? 4. What ML library was used? 5. Did you personally develop and implement the ML algorithm? If not, did you have a clear and elaborated understanding of its purpose in the project? 6. Did you, or your team experience any difficulty in working with ML?
Educative Framework	Students only	1. How clear was the Grasshopper ML resource? 2. How clear were the workshops/demonstrations? 3. Was the structure of the educational module appropriate? 4. What were the best things about the module? 5. Did the assignment help facilitate your learning? 6. Was the amount of assignments appropriate? 7. Did the feedback help you learn? 8. Did you feel a part of the collaborative learning community? 9. Did you research additional resources to help you learn? 10. Were you satisfied with the quality of teaching? 11. In your opinion, what could be improved?
Educative Framework	Staff only	1. Were the students engaged in the module? 2. Did the students exhibit accurate understanding of how to apply the required mathematical and computational principles? 3. Were the students able to abstract and apply the mathematical principles in novel situations? 4. Were the students able work independently from the tutors? 5. Did the students independently seek out and locate required information to implement their understanding? 6. Did the students explore beyond the content that was required for the assessments? 7. Did the students exhibit clarity and meticulousness in delivering and communicating their experimentations content? 8. Were the students successful in outputting a geometrical shape? 9. What strategies did you utilise to help teach ML? 10. In your opinion, what could be improved?

content. However, a separate educational framework could be developed with that objective.

A core component of the project-based pedagogy was the biomimetic sourcing of an environmental dataset to train the neural network, used to then predict the pavilion's form. A vocal minority of students questioned the necessity of an ANN algorithm for their specific project, and suggested that it was "unclear exactly the advantage of using the neural network in this instance." In particular, 8% of the students asserted that there was "a bit of a disconnect - what we were doing could just as well have

been achieved using a different method." Clearly, the datasets sourced by the students could have been modelled with conventional methods (such as rule-based models, 'Monte Carlo' methods, and/or time-series modelling). However, the original intention was to have the students exploring more complex, high-dimensional datasets (i.e. several hundred data points in length). A dataset of this complexity would be unfathomable to the human, and difficult for most statistical modelling methods, thus leaving the neural network as an appropriate model. Unfortunately, a combination of the curriculum's time restrictions, and the introductory nature of the course, limited the size and complexity of the dataset to two- or three-dimensions. Even still, questioning the necessity of the neural network was a valid criticism, and perhaps an emphasis on a more complex dataset will be professed in future implementations of this module.

The curriculum was built to encourage collaboration, by grouping the students an evolving, additive configuration. After a preliminary individual assessment, students were organised into six groups (two - three people each) for three weeks. Subsequently, they were recombined into three groups (four - five people each) for six weeks, leading to the final pavilion design. Because the final presentation required students to combine all facets of the design process into a cohesive narrative, 38% of the students noted that their focus shifted from the application of the ANN algorithm to the overall course deliverables. Therefore, due to the very nature of collaborative coursework, each team member specialised in delivering one particular aspect of their final assessment (i.e. dataset evaluation, material explorations, parametric design or physical prototyping). While it may be the case that machine learning education ceased for some, the developers of the curriculum found it pertinent to convey the nature of collaborative working environments. In the design industry, the designer is expected to work alongside an alliance of experts of varying disciplines, each with specialised roles. Thus, it's up to the individual, and by extension, the student, to continuously exchange knowledge and information, allowing for group work to emerge and improve.

The majority of the comments in response to the beginning of the educative curriculum were overwhelmingly positive, stating that it was "clear and well explained," had an "excellent structure," and remained "engaging, especially in a topic we don't really know." However, as the module progressed, the students started to echo a similar thought: "there weren't enough examples." This was premeditated. Throughout the curriculum, we refrained from including instances of past machine learning applications, under the assumption that it would negatively affect the student's creativity, by directing and limiting their perception of the possibilities of neural networks. However, from the interview series, students believe that "being able to see a more concrete example of how these principles are able to output [a form] would be better." 8% of the students claimed that "precedents are one of the biggest things in design." Upon reflection, their point is valid. The essence of the inquiry-based teaching pedagogy rebels against the presentation of established facts. To remedy our assumption, we aim to include various case studies of ML applications within design in future implementation of the educative framework and curriculum.

Retrospectively, evaluating the educational framework reveals its contribution to the design field and margins of improvements. Students can effectively learn the logic behind neural network prediction for design purposes while applying their individual thinking to the final collaborative output. An emphasis on larger, more complex datasets, combined with a showcase of machine learning precedents, will be included into future iterations.

IMPLICATIONS

The development of the educative framework, and its implementation within the design curriculum, is a preliminary step towards a greater educational strategy. Over the coming years, the framework will be iteratively revised and implemented according to the feedback and criticisms provided by students and staff. In conjunction with changes to the compo-

nents, the updated framework will include a wider range of machine learning algorithms, increasing the breadth of potential skills gained.

For the purpose of this research, the framework was integrated into a computational design degree. However, as the educational content subsists within a four-week teaching period, it could effectively be embedded into any other discipline. Industrial designers, landscape architects, and urban planners could integrate the ML learning outcomes within their own curricula, and use the skills gained to inform processes beyond performative design prediction. Moving further than the domain of the Built Environment, the framework could be implemented to serve any other discipline with a predictive design intention (such as Collaborative Robotics, Medicine or Law). However, an intermediate understanding of Grasshopper, the software containing the neural network resource, would be advisable.

Finally, the implementation of the framework need not be contained within a university. Towards the overarching objective of bridging the gap between architecture and machine learning, the framework could be directed at those currently working in the architectural industry. The developed educational framework can be altered and implemented at varying scales, from the classroom to the broader design community.

CONCLUSION

This research project postulates that to resolve the disparity between architecture and their adoption of emerging technology, architectural academic programs should incorporate ML into their educational strategies. Consequently, this research has developed a novel educational framework, implemented in a curriculum module and employed in a computational design studio, exploring the potential for ML-driven design prediction. Subsequent evaluation of the module's implementation revealed that the student-centred approach, utilising inquiry-, project-, and resource-based pedagogies, fosters a collaborative learning environment. Due to its success, the intention is to further refine the framework for its implementation in multidisciplinary design curricula, and even within industry practices.

REFERENCES

Barron, B. and Darling-Hammond, L. 2008, 'Teaching for Meaningful Learning: A Review of Research on Inquiry-Based and Cooperative Learning', in Furger, R. (eds) 2008, *Powerful Learning: What we Know About Teaching for Understanding*, Jossey-Bass, San Francisco

Blumenfeld, P.C., Soloway, E., Marx, R.W., Krajcik, J.S., Guzdial, M. and Palincsar, A. 1991, 'Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning', *Educational Psychologist*, 26(3-4), pp. 369-398

Brown, S. and Smith, B. (eds) 2013, *Resource-Based Learning*, Routledge, London

Brynjolfsson, E. and McAfee, A. 2017, *Machine, Platform, Crowd - Harnessing our Digital Future*, W. W. Norton & Company, New York

Carpo, M. 2017, *The Second Digital Turn - Design Beyond Intelligence*, MIT Press, Cambridge

Hannafin, M.J. and Hill, J. 2007, 'Resource-Based Learning', *Handbook of Research on Educational Communications and Technology*, 3, pp. 525-536

Kaplan, J. 2016, *Artificial Intelligence - What Everyone Needs to Know*, Oxford University Press, New York

Kelly, K. 2016, *The Inevitable - Understanding the 12 Technological Forces that will Shape our Future*, Penguin Books, New York

Khean, N., Kim, L., Martinez, J., Doherty, B., Fabbri, A., Gardner, N. and Haeusler, M.H. 2018 'The Introspection of Deep Neural Networks - Towards Illuminating the Black Box - Training Architects Machine Learning via Grasshopper Definitions', *Proceedings of the 23rd International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2018, Volume 2*, Beijing, pp. 237-246

Norman, D.A. and Spohrer, J.C. 1996, 'Learner-Centered Education', *Communications of the ACM*, 39(4), p. 24.27

O'Neill, G. and McMahon, T. 2005, 'Student-Centred Learning: What Does it Mean for Students and Lecturers', in O'Neill, G., Moore, S. and McMullin, B. (eds) 2005, *Emerging Issues in the Practice of University Learning and Teaching*, AISHE, Dublin, pp. 27-36

Prince, M.J. and Felder, R.M. 2006, 'Inductive Teaching and Learning Methods: Definitions, Comparisons, and Research Bases', *Journal of Engineering Education*, 95(2), pp. 123-138