AN ASSESSMENT OF TOOL INTEROPERABILITY AND ITS EFFECT ON TECHNOLOGICAL UPTAKE FOR URBAN MICROCLIMATE PREDICTION WITH DEEP LEARNING MODELS

NARIDDH KHEAN[1], SERJOSCHA DÜRING[2], ANGELOS CHRONIS[3], REINHARD KÖNIG[4] and MATTHIAS HANK HAEUSLER[5]
[1,2,3,4]*City Intelligence Lab, Austrian Institute of Technology, Austria.*
[1,2,4]*Bauhaus-Universität Weimar, Germany.*
[5]*Computational Design, UNSW, Australia.*
[1]*nariddh.khean@ait.ac.at, 0000-0003-0549-7784*
[2]*serjoscha.duering@ait.ac.at, 0000-0002-1003-1236*
[3]*angelos.chronis@ait.ac.at, 0000-0002-6961-2975*
[4]*reinhard.koenig@uni-weimar.de, 0000-0002-3579-8855*
[5]*m.haeusler@unsw.edu.au, 0000-0002-8405-0819*

Abstract. The benefits of deep learning (DL) models often overshadow the high costs associated with training them. Especially when the intention of the resultant model is a more climate resilient built environment, overlooking these costs are borderline hypocritical. However, the DL models that model natural phenomena—conventionally simulated through predictable mathematical modelling—don't succumb to the costly pitfalls of retraining when a model's predictions diverge from reality over time. Thus, the focus of this research will be on the application of DL models in urban microclimate simulations based on computational fluid dynamics. When applied, predicting wind factors through DL, rather than arduously simulating, can offer orders of magnitude of improved computational speed and costs. However, despite the plethora of research conducted on the training of such models, there is comparatively little work done on deploying them. This research posits: to truly use DL for climate resilience, it is not enough to simply train models, but also to deploy them in an environment conducive of rapid uptake with minimal barrier to entry. Thus, this research develops a Grasshopper plugin that offers planners and architects the benefits gained from DL. The outcomes of this research will be a tangible tool that practitioners can immediately use, toward making effectual change.

Keywords. Deep Learning; Technological Adoption; Fluid Dynamics; Urban Microclimate Simulation; Grasshopper; SDG 11.

1. Introduction

Deep learning (DL)—a sub-field of machine learning predicated on a family of algorithms known as neural networks—has been toyed with for decades. However, DL

POST-CARBON, Proceedings of the 27th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2022, Volume 1, 273-282.

has only gained its recently burst in popularity in 2012. Professor and computer scientist, Olga Russakowsky, notes '2012 was really the year there was a massive breakthrough in accuracy, but it was also a proof of concept for deep learning models, which had been around for decades' (Agrawal, 2018, p. 28-29). Most attribute this revitalisation to a met threshold in 'huge, flexible stores of data' combined with increased 'access to [more] powerful computers' (Browne, 2019, p. 64). Since then, where DL took a landmark leap in accuracy, it has been applied in both academia and industry, in almost all fields ranging from medical diagnosis (Hafiz and Bhat, 2020) to agriculture (Kamilaris and Prenafeta-Boldú, 2018). Even in the urban planning industry, DL has made its mark. However, despite its wide breath of application, DL is not a 'silver bullet'. While 2012 was the year that we saw deep learning's rise, 2018 was the year it became cool to criticise deep learning (Broussard).

The efficacy of deep learning algorithms is fickle. 'Neural networks can model very complex patterns [...] and, as such, are very powerful. In fact, they are so powerful that they can even model the noise in the training data' (Baesens, 2014, p. 51), an insidious phenomenon known as 'overfitting'. Furthermore, DL algorithms are 'black boxes', non-transparent and opaque modelling methods which leaves any attempt to understand why the model predicted what it did a fruitless endeavour. Finally, DL models are only as good as the data used to train it. 'We tend to think of data as the immutable truth, but we forget that data and data-collection systems are create by people' (Broussard, 2018, p. 57), thus is susceptible to error. These factors, and other mathematical and computational restrictions (Baesens, 2014, p. 58), make DL suitable for only certain types of problems, and when misapplied, can lead to horrific ramifications (Angwin et al., 2016).

This research will focus on the application of DL within urban microclimate analysis prediction for climate resilience. (Its suitability will be discussed in the literature review). These models, when applied to a computationally intensive simulation such as wind factor analysis, provides orders of magnitude improvement through prediction rather than simulation. However, discounting the costs associated with training are incredibly myopic and selective. Training DL models—a costly process involving data acquisition and engineering, consuming computation time and resources, as well as the need for continual model assessment and retraining upon conceptual drift—all contribute to a horrific carbon footprint. A study from the University of Massachusetts, Amherst, concluded that the process of training large AI models emits roughly five times the lifetime emissions of the average American car (Strubell et al., 2019). Despite the righteous proclamation for sustainability and environmental resilience through deep learning (which could arguably be seen as an oxymoron in itself), DL models need to go beyond a costly and environmentally damaging research toy and move toward a commercially viable product to be adopted by those in industry who would use it to make tangible positive environmental impact.

2. Literature Review

'All models are wrong, but some are useful' (Domingos, 2015, p. 149). Each modelling method has a variety of strengths and weakness. And deep learning is no exception. An understanding and careful consideration of these attributes will be the difference between success and gross misapplication. Thus, what problems within urban planning

can be most suitably addressed with deep learning?

One such set of problems focus on predicting the outcomes of a computationally intensive simulation process based on natural phenomena. Issues of data quantity and quality—'multiple data sources, subjective judgement, limited computing facilities, and size of data' (Baesens, 2014, p. 152)—are ever-present when it comes to training deep learning models. However, by creating a training dataset through simulation over a variety of real-world and generated urban forms, the engineer has complete control and can drastically minimise these issues. Furthermore, because the DL model is trained upon data generated in simulation, there is minimal concern to do with human- and equipment-error. And the fact that the data is generated means that the size of the dataset is never an issue, because if one needs a larger dataset, they could simply run more simulations; time is the only limiting factor. 'Data makes prediction better' (Agrawal, 2018, p. 174). Through these sets of problems, the issues that surround both data quality and quantity are drastically minimised. Conversely, the benefits gained from using DL methods over typical simulation approaches come down to the staggering gains in computational speed and efficiency. Because, despite the large amounts of time and computational power needed to train a DL model, once trained, the resources needed to make a prediction is orders of magnitude less than that of running a simulation. Once trained, the simulation (now prediction) process is made much more efficient.

This research focuses on the application of deep learning on urban microclimate analyses, which in itself includes a breath of studies, such as urban heat island, rainfall, and pedestrian comfort simulations. Typically, these studies are conducted through arduous simulation approaches. The promise of DL for microclimate prediction is to model the patterns between urban form and the input parameters with the simulation results; to predict, rather than simulate. A 2019 meta-analysis concludes that the then current majority of DL application in urban planning focuses on 'urban growth or urban land use analysis while there [was] less interest in environmental studies' (Grekousis, 2019, p. 253). For research contribution toward that of lesser studied, this research will further narrow down its scope by assessing the application of deep learning on wind factor analysis based on computational fluid dynamics (CFD).

There has been a plethora of research done toward the development of deep learning models for CFD prediction (Zhu et al., 2018; Kulkarni et al., 2019; Erdemir et al., 2020), however, there is a lesser subset dedicated for their application in urban planning (Ding and Lam, 2019; Kaseb et al., 2020; Mokhtar et al., 2021; Sun et al., 2021). A 2021 review provides a comprehensive overview of the published research on the application of DL in CFD analysis, for both the indoor and outdoor environment. Within their review, they identify that the 'vast majority of applications in the built environment are limited to substituting [the conventional approach] [...] to achieve faster predictions' and that their main objectives 'are to reduce computational cost while keeping the same order of accuracy' (Calzolari and Liu, 2021). However, the study also concedes that 'most often, fast predictions come at a cost of degraded accuracy'.

While the 2021 review compares research outcomes on algorithm choice and model speed and accuracy, what is not taken into account is their accessibility. 'Technology is a tool. That is true whether it's a hammer or a deep neural network' (McAfee, 2017, p. 330). Stretching that further, unless that tool is designed and offered

in a manner that is highly accessibly and available with minimal barriers to entry, such a tool may lay unused, affecting no one. Yes, although it is true that conventional 'CFD simulation requires engineering knowledge and expensive computational resources, [creating] a barrier for people to use [it]' (Ding and Lam, 2019), DL models also require a highly specific set of engineering skills and knowledge to use; that of MLOps. MLOps ('machine learning operations', borrowing from the word, 'DevOps') is the task of deploying or embedding a machine learning model into a program, application, or API that allows access to said model. Without sufficient MLOps, the complexity barrier still exists, but has just shifted. Of the aforementioned research published on DL CFD prediction, none make any mention of its deployment or use (albeit such focus is arguably outside the scope of the reviewed publications).

3. Research Objective

There has been recent innovation toward improving or replacing conventional computational fluid dynamics simulations with deep learning models for the sheer efficiency gains offered by such methods. The common objective of these efforts is to increase the speed and decrease the computational costs of analyses, while minimising accuracy loss. However, there is significantly less research emphasis placed on their usage and access, namely through MLOps. Despite the evidence suggesting that the models improve upon the resource costs of conventional methods by literal orders of magnitude, this research suggests that to make a tangible and measurable impact with these models, they need to be positioned with a low barrier to entry so that they can be used by other researchers and, more importantly, those in industry.

The objective of this research is to contribute to the broader uptake of DL models for rapid CFD studies, through the development and implementation of an API wrapper within a visual programming environment more familiar to urban planners (Grasshopper).

4. Methodology

This research builds upon the work and research of the City Intelligence Lab, namely, upon the 'Intelligent Framework for Resilient Design' (InFraReD) (Chronis et al., 2020). InFraReD is a web-based platform for intelligent and resilient urban design that leverages deep learning to accelerate a variety of urban microclimate analyses. As of the time of writing, these models are accessible via two methods: the InFraReD web application as well as through a GraphQL application programming interface (API). It is the latter method by which this research leverages to create a Grasshopper plugin that takes a step toward lowering the barrier of usage.

The Grasshopper plugin developed during this research was done so following an applied, action research methodology. Widely attributed to social psychologist, Kurt Lewin, action research started as a 'methodology for intervening in and researching the major social problems of the day' (Hopkins, 2008, p. 48). According to Lewin, action research 'consisted in analysis, fact-finding, conceptualisation, planning execution, more fact-finding or evaluation; and then a repetition of this whole circle of activities' (Kemmis, 1988, p. 13). This research largely follows the original incarnation of the action research methodology for the case study.

The software development for this research was conducted within the span of a week and was done so with the intention to use the outcomes for the 2021 Digital Futures workshop.

5. Case Study

Contributing toward the broader uptake of DL models for urban microclimate studies, this research intends to develop a wrapper around an existing platform of DL models in a software environment more familiar to urban planners: the 3D modelling software, Rhino 6, and its parametric scripting counterpart, Grasshopper. The existing platform that provides access to DL microclimate models is the Intelligent Framework for Resilient Design (InFraReD), a web application and API made available by the City Intelligence Lab (Chronis et al., 2020). However, these two methods of use still prove some difficulty, and thus, resistance. The web app asks the user to shift their software ecosystem and the data import/export function is not yet available (forcing the user to recreate all their geometry in the app), and conversely the API still requires some understand of programming which has proven to be too high a barrier of entry. Consequently, a third, more accessible method, is the projected research outcome.

At the time of writing, InFraReD has four microclimate studies available to users: solar radiation, sunlight hours, wind speed, and wind comfort. To perform a prediction using these models, there are two required inputs: urban geometry, and the input parameters that effect the post-processing for each model (i.e., wind speed, wind direction, etc.). However, before even making a prediction, one needs to overcome the challenges of formulating an authenticated request, of synchronising data between Grasshopper and InFraReD's servers, and then visualising the resultant predictions. By far, the greatest challenge was the shift of mental model between that of a web application and a Grasshopper plugin, not helped with the time restraints of the task. Each of these challenges and their designed solutions are explained below.

There are several programming languages by which a Grasshopper plugin can be written, however the choice of Python was one made considering the expertise of the research group and the time allotted for the plugin's development.

5.1. AUTHENTICATION

To access InFraReD's services, one first needs to authenticate. As the DL models are resource demanding for the hardware it is currently deployed on, we are monitoring and limiting its use by keeping track of who has access.

Authentication is handled quite simply through the use of cookies. Using either the web application or the API, all a user would need to do is send a POST request to InFraReD's authentication route with their log in credentials. Upon valid credentials, a response storing their authentication cookie would be returned. Within the cookie, is a JSON web token (JWT), 'a compact, URL-safe means of representing claims to be transferred between two parties' (Jones et al., 2015). This claim can be used to statelessly verify a user on InFraReD's backend, and any subsequent request they may send, so long as the request contains that cookie. Thus, the task of leveraging this authentication technique within a Grasshopper plugin is an exercise of extracting this cookie and injecting it in all subsequent requests from the other plugin components.

First, an authentication component was developed that required the username and password of InFraReD credentials. The component would send a request to InFraReD, and upon successful authentication, the component would receive a response with the cookie containing the JWT. To preserve this cookie and reuse it in the other component's requests, the cookie was extracted and stored in a 'sticky' variable, made available in the 'scriptcontext' module, making the variable accessible in other components in the Grasshopper script.

```
...
class Authenticate(component):
    ...
    def RunScript(self, U, P):
        ...
        # Extract cookies
        for cookie in response.Cookies.GetEnumerator():
            if cookie.Name == 'InFraReD':
                auth_token = cookie.Value
        ...
        # Store stickies
        sc.sticky['InFraReDAuthToken'] = auth_token
        ...
```

Each InFraReD Grasshopper component checks for this sticky variable for the authentication cookie before it computes anything further.

```
...
    def RunScript(self, ...):
        ...
        # Validate authentication
        auth_token = sc.sticky.get('InFraReDAuthToken')
        if not auth_token:
            component.AddRuntimeMessage(
                self,
                Grasshopper.Kernel.GH_RuntimeMessageLevel.Error,
                "[Message]"
            )
        ...
```

5.2. DATA SYNCHONISATION

InFraReD is a 'stateful' application. As opposed to its counterpart known as stateless applications, stateful applications store data and events that may have occurred with a user's interactions, which then allows future experiences to be adjusted. Common examples of stateful web applications include online banking and email. For the case of InFraReD, the stored data includes project metadata, 'snapshots' (which is InFraReD's term for design variants), urban geometries, and a variety of analysis configuration variables (including which analyses are set to run project-wide, what input parameters for each analysis, etc.). The fact that InFraReD's database stores urban geometry, brings with it a host of benefits, significantly, removing the need to always send geometry data within a request each time a prediction needs to be run. Conversely, if an external application, such as Rhino/Grasshopper, were to create geometry outside of InFraReD, this geometry would need to be mirrored, and the issue of data synchronisation becomes a potential challenge (addressed in the Evaluation section).

The components designed to synchronise Rhino/Grasshopper geometry with the InFraReD server must parse that data into a format that InFraReD can comprehend. InFraReD represents building geometry as a building footprint and a height. And so, any geometry type that may be considered a building (i.e., boundary representations, meshes, etc.) need to be simplified and reduced to a curve and floating-point number.

```
import rhinoscriptsyntax as rs
import ghpythonlib.components as ghcomp
...
def extract_footprint_and_height(building):
    mesh = rs.coercemesh(building)

    # Footprint
    crv = ghcomp.MeshShadow(
        mesh,
        rs.CreateVector([0, 0, 1]),
        rs.CreatePlane([0, 0, 0])
    )
    footprint = ghcomp.SimplifyCurve(crv).curve

    # Height
    z = [v.Z for v in ghcomp.DeconstructMesh(mesh).vertices]
    height = max(z) - min(z)
    ...
```

Note, there are some complexities removed in favour for brevity and concision (such as a recursive function that called 'ghcomp.SmoothMesh()' if the 'ghcomp.MeshShadow()' method failed to return a valid curve).

Having extracted the footprint and height, to formulate a request to the InFraReD server, the building data is then converted to a GeoJSON string a8nd converted to a GraphQL mutation request.

5.3. PREDICTION VISUALISATION

The InFraReD web application was designed to be asynchronous. Despite being orders of magnitude faster than a CFD simulation, making a prediction could take up to 3 seconds, not to mention the pre- and post-processing involved (depending on hardware). At the time of writing, InFraReD is deployed on a single node (however the research team has been working on using Kubernetes to scale to a multi-node cluster). Thus, if there were to be several prediction requests sent to InFraReD in a very short span of time, these jobs would be added to a queue (handled by Apache Kafka) and be resolved sequentially.

When InFraReD makes a prediction, the resultant inference is postprocessed and stored on the server as a normalised matrix. This was an intentional design decision, as, although this offsets the responsibility of visualising the results up to the technology providing access to InFraReD (i.e. Unity in the web browser, or whichever technology one chooses if they access InFraReD's API), this maintains a high level of interoperability, and assures that no technology cannot use and manipulate the visualisation in their own fashion. Thus, the developed Grasshopper plugin must also include a method to visualise the matrix of results. Due to the time restrictions of the development period, this process was achieved with Aviary.

5.4. OTHER CHALLENGES

This research paper could not hope to include all the challenges that were encountered and resolved (and not resolved) during the development of the Grasshopper plugin. The three that were discussed here were intentionally selected as they posed the most development difficultly, in conjunction with providing the most fuel for discussion in the evaluation section. For the documentation and source code, please visit [redacted].

6. Evaluation

Due to the brevity of development time, there are some software design decisions made to favour the rapid completion of the project over a robust solution. As economist Thomas Sowell so famously positions, 'there are no solutions, only trade-offs' (Sowell, 2007). However, if the objective of this research is for the general uptake and use of deep learning models, this work needs some further action research iterations to reach 'production-ready'. Below are what this research has identified as the outcome's major pain-points, which coincide with the team's next steps.

The first major inconvenience is the compute graph. Despite early efforts in the project, the developed Grasshopper plugin was almost a one-to-one conversion of the GraphQL API into a series of Python requests wrapped in Grasshopper components. There is a general failure to consider the difference in the mental model between that of a web application API and a Grasshopper script. As such, using the Grasshopper plugin, there is a rigorously strict method by which the plugin should be used, so not as to disrupt the compute graph. If so much as one connection triggers a component that should not run, an entire InFraReD project could be ruined (this would not affect the geometry and such on the Grasshopper side, only on InFraReD's servers). Coupled with the fact that InFraReD is stateful, and how data on Grasshopper must be mirrored with that on InFraReD's servers, if one ill-timed API request was triggered, the local and server-side projects could lose synchronisation and the disparities would continue to diverge on subsequent requests. The way to resolve this is to rethink how the deep learning models would best be used from the perspective of a Grasshopper user, rather than that of a web application. Then, collating these chunks of behaviour, recreating the plugins to include the necessary GraphQL API requests to perform such actions.

Despite the computational speed and efficiency gained over conventional simulation approaches, deep learning predictions still take some time to compute, not to mention the overhead of sending that data over the web. A single prediction made with the Grasshopper plugin takes roughly 3 seconds. However, because geometry is an input for the 'predict' component, every geometric manipulation would add at least those 3 seconds for computation. This is unnecessary for each incremental adjustment. One simple method to prevent such issues, offered in the workshop and in the developed tutorial video, is to have every InFraReD Grasshopper component disabled unless intending to send a request. If a user wants to send a request, they would re-enable the required components, and then re-disable them once complete. This is arguably a very 'un-Grasshopper-like' solution. Thus, simple buttons on the components, used to trigger the request to the API, could be the better solution.

Finally, Grasshopper and InFraReD represents building geometry differently, and in fact, in Grasshopper, this is entirely up to the user. Because of the 'one-size-fits-all'

solution of converting geometry to meshes, there is potential for loss of detail. However, this problem is drastically exacerbated when considering how InFraReD represents building geometry: footprint and height. Thus, if a user has geometry more complex than a vertical extrusion, the building would be simplified. The reason for this is primarily due to how the deep learning models were trained. There is no solution that could have been implemented within the development of the plugin, and requires the recreation of a training dataset, retraining the DL models, and rebuilding the data input pipelines on InFraReD's servers, well beyond the scope of this research. This is a vital stipulation that all users of InFraReD should be aware of.

Some smaller issues include: the method of credential input is not secure and can easily be stolen if a user saves the Grasshopper script and disseminates it (a pop-up box would be the solution here); the process of result visualisation would be more reliable if included in the plugin rather than relying on an external plugin (which would allow the implementation of domain-specific colours gradients for the different analyses), amongst others. The outcomes of this research are far from ready for wide-spread adoption—the feedback from the participants of the 2021 Digital Futures workshop indicated as much. However, this research succeeds in illuminating a side to microclimate analysis through deep learning that has not received enough attention and provides a method by which one can tangibly contribute. This research hopes to act as a steppingstone for further developments in this field.

7. Conclusion

While there has been a general trend toward embedding deep learning processes for improving or replacing conventional approaches in almost all areas of research since 2012, there has been substantially less emphasis placed on the use of such technology. This imbalance is especially problematic, and even hypocritical, as one considers how incredibly resource-intensive it is to train a DL model, upon the application of deep learning for climate resilience. The outcomes of this research draw attention to the necessity of developing methods to access DL-based microclimate predictions, and further develops one such method. Leveraging the existing InFraReD API, which offers rapid DL-based prediction on solar radiation, sunlight hours, and wind factor analysis, this research implemented a wrapper around the API within a software environment more familiar to urban planners. This developed Grasshopper plugin was then used in the 2021 Digital Futures conference workshops.

References

Agrawal, A. (2018). *Prediction Machines: The Simple Economics of Artificial Intelligence*. Boston, Massachusetts: Harvard Business Review Press. ISBN: 978-1-6336-9567-2.

Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016). *Machine Bias*. Propublica. Retrieved November 27, 2021, from https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

Baesens, B. (2014). *Analytics in a Big Data World: The Essential Guide to Data Science and its Applications*. Hoboken, New Jersey: Wiley. ISBN: 978-1-1188-9270-1.

Broussard, M. (2019). *Artificial Unintelligence: How Computers Misunderstand the World*. Cambridge, Massachusetts: The MIT Press. ISBN: 978-0-2620-3800-3.

Browne, J. (2019). *Think, Make, Imagine: A Brief History of the Future*. London: Bloomsbury Publishing. ISBN: 978-1-5266-0572-6.

Calzolari, G., & Lui, W. (2021). Deep learning to replace, improve, or aid CFD analysis in build environment applications: A review. *Building and Environment,* 206, 108315. https://doi.org/10.1016/j.buildenv.2021.108315.

Chronis, A., Aichinger, A., Duering, S., Galanos, T., Fink, T., Vesely, O., & Koenig, R. (2020). InFraReD: An Intelligence Framework for Resilient Design. In *25th International Conference on Computer-Aided Architectural Design Research in Asia*. TheAssociation for Computer-Aided Architectural Design Research in Asia (CAADRIA).

Ding, C. & Lam, K. P. (2019). Data-driven model for cross ventilation potential in high-density cities based on coupled CFD simulation and machine learning. *Building and Environment,* 165, 106394. https://doi.org/10.1016/j.buildenv.2019.106394.

Domingos, P. (2015). *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. New York: Basic Books. ISBN: 978-0-4650-6570-7.

Erdemir, G., Zengin, A. T., & Akinci, T. C. (2020). Short-term wind speed forecasting system using deep learning for wind turbine applications. *International Journal of Electrical and Computer Engineering,* 10(6). https://doi.org/10.11591/ijece.v10i6.pp5779-5784.

Grekousis, G. (2019). Artificial neural networks and deep learning in urban geography: A systematic review and meta-analysis. *Computers, Environment and Urban Systems,* 74, 244-256. https://doi.org/10.1016/j.compenvurbsys.2018.10.008.

Hafiz, A. M., & Bhat, G. M. (2020). A survey of deep learning techniques for medical diagnosis. *Information and communication technology for sustainable development,* 161-170. https://doi.org/10.1007/978-981-13-7166-0_16.

Hopkins, D. (2008). *A teacher's guide to classroom research*. Maidenhead: Open University Press. ISBN: 978-0-3352-2175-2.

Jones, M., Bradley, J., & Sakimura, N. (2015, May). *JSON Web Token (JWT).* RFC 7519. Retrieved November 27, 2021, from http://www.rfc-editor.org/rfc/rfc7519.txt.

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture. *Computers and Electronics in Agriculture,* 147, 70-90. https://doi.org/10.1016/j.compag.2018.02.016.

Kaseb, Z., Hafezi, M., Tahbaz, M. & Defani, S. (2020). A framework for pedestrian-level wind conditions improvement in urban areas: CFD simulation and optimization. *Building and Environment,* 184, 107191. https://doi.org/10.1016/j.buildenv.2020.107191.

Kemmis, S. (1988). *The Action research planner.* Waurn Ponds, Victoria: Deakin University Press. ISBN: 978-0-7300-0521-6.

Kulkarni, P. A., Dhoble, A. S., & Padole, P. M. (2019). Deep neural network-based wind speed forecasting and fatigue analysis of a large composite wind turbine blade. *Journal of Mechanical Engineering Science,* 233(8), 2794-2812. https://doi.org/10.1177/0954406218797972.

McAfee, A. (2017). *Machine, Platform, Crowd: Harnessing our Digital Future*. New York: W.W. Norton & Company. ISBN: 978-0-3932-5429-7.

Mokhtar, S., Beveridge, M., Cao, Y., & Drori, I. (2021). Pedestrian Wind Factor Estimation in Complex Urban Environments. In *13th Asian Conference on Machine Learning*.

Sowell, T. (2007). *A conflict of visions: ideological origins of political struggles*. New York: Basic Books. ISBN: 978-0-4650-0205-4.

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. In *57th Annual Meeting of the Association for Computational Linguistics, ACL 2019,* The Association for Computational Linguistics (ACL).

Sun, C., Zhang, F., Zhao, P., Zhao, X., Huang, Y., & Lu, X. (2021). Automated Simulation Framework for Urban Wind Environments Based on Aerial Point Clouds and Deep Learning. *Applications of AI and Remote Sensing in Urban Systems,* 13(12), 2383. https://doi.org/10.3390/rs13122383.