THE LEARNING FACADE

by

Paul MacKnight

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the dual degree of Master of Science in
Architecture and Information Technology

Charlotte

2018

Approved by:

_____
Dimitris Papanikolaou

_____
Kyoung-Hee Kim

_____
Eric Sauda

ABSTRACT

PAUL MACKNIGHT. The Learning Facade. (Under the direction of DIMITRIS PAPANIKOLAOU)

Despite the rising popularity and utilization of intelligent systems, much of the built environment, especially architecture, remains prescriptive or responsive in nature. Kinetic facades, especially, still rely on the analysis of historic or approximated data to generate a solution through the utilization of a multi-objective optimization (MOO) algorithm. This approach lacks the ability to adapt to the changing forces (i.e. site specific micro-climates or changing occupants) to which buildings are subjected for two reasons: 1) MOO is computationally expensive due to the immense solution space and 2) it can only solve for known objectives. This lack in ability for facades to adapt to changing conditions or be designed using actual site data has been one of the hindrances on the growth of the industry.

Kinetic facades should instead be developed as an integrated portion of an intelligent system that is able to unify environmental data and user input in real-time to create an interior environment that is comfortable, energy efficient, and able to adapt to any future changes. Inputs such as space volume and user preferences, then, must be assumed to be unknown, putting MOO at a disadvantage in intelligent systems. As such, I have developed a control system architecture for an intelligent facade that utilizes a neural network algorithm (a form of machine learning) to address the need of adaptation in kinetic facades. To test my method, I utilized Rhino 5, Grasshopper, and Python with a simulated dataset as a case study.

Contents

List of Figures

Nomenclature

Pre-scripted Facade - Goes through certain motions based on predetermined calculations of environmental data only. Cheapest, but less common than and mostly outdated by the other types of facades with the advancement of the Internet of Things and better sensor technology.

Responsive Facade - Takes input from the environmental data and data from building sensors to determine the best possible positioning for the panels in the system. This only responds to current conditions, or "lives in the moment," and has no memory of past settings. Predominant type of kinetic facade used today.

Adaptive Facade - Similar to responsive, but with an added layer of "learning" over time to require a steadily decreasing amount of computation, as long as it has identified a common pattern between its inputs. This will allow it to compare current sensor readings, environmental data, and possibly even user inputs with past instances and if it finds a close match it can use those settings rather than calculate them from scratch.

Rhino 5 - Officially called Rhinoceros 5, this is a 3D modelling and Computer-Aided Design (CAD) software used across many different disciplines, including architecture and automotive design.

Grasshopper - A parametric modelling software that is a plugin to Rhino 5. It allows for a more intuitive Graphical User Interface (GUI) and better control over script generated geometry in Rhino 5. Grasshopper has many additional plugins - such as the Honeybee, Ladybug, and Diva 4 used in this research - that allow for extended functionalities in creating or analyzing parametric geometry created by Grasshopper.

Introduction

1.1 Intelligent Facades

To understand the current state and future needs of the facade industry, we must first examine three things: what constitutes an intelligent system versus a smart system, what an intelligent facade is, and identify how it is distinguished from standard static and kinetic facades. After a review of literature and noting others in the field noting the same, “intelligent vs. smart” is an issue of ambiguity in the discourse, especially that of architecture. It is an understandable misconception that the two terms are interchangeable, after all the two are very close synonyms, but their definitions take on new meanings when used in reference to devices and other technology. Based on the overall use of the terms in the discourse and for the purposes of this paper, the term “smart” references a quality of interconnectedness, marked by the free ability to communicate with other devices in the system, while the term “intelligent” references the implementation of learning as an extended function of smartness.

In the realm of architecture, use of these two terms has most notably materialized in the form of smart homes. The idea that things should be as convenient for the user as possible has driven the Internet of Things to become partly physical in its implementation, spanning from home assistants such as Google or Alexa to smart locks to smart thermostats. These devices create a sort of web of free use that allows the user to control any part of their house either presently through voice commands or remotely through their smart phones. While there are some aspects of learning, such as Alexa being trained to recognize specific user voices, the vast majority of these devices are

purely responsive in nature, leaving it up to more advanced users or researchers to implement a level of intelligence that allows a predictive nature to arise. Nest, a brand of smart thermostats, is likely the closest commercially available product to an intelligent system we have due to its ability to learn a family's preferences and schedules with minimal input and only basic use of the temperature settings. Another thing to note moving forward is that all of these systems are user centric. That is, they focus on a single or small group of user(s) as their target with all other aspects, such as energy efficient HVAC use for the Nest, taking a back seat.

For smart and intelligent architecture, the stakes are higher. Building performance plays a huge role in the design of the building, while users take a back seat. As such, smart architecture is just as much about the design of the building as it is the clever use of smart systems and sensors throughout. This means both static and kinetic facades can be smart, with their respective design objectives optimizing for a particular facade issue, particularly thermal gain and daylighting. This creates the need for sensors to communicate information back to the building control system to better aid the localized use of lighting fixtures and HVAC. Kinetic facades are the more popular solution for this level of smartness due to their ability to address a wider range of environmental conditions while still achieving the optimal solution. Intelligent facades, then, must address all aspects of the building during its lifetime and have the ability to adapt to any changes in the external and internal environments, which can only mean more complex sensor arrays and control systems.

1.2 Control Systems

Developing the control system for these facade systems is likewise no small feat. Many different methods for controlling a kinetic facade have been attempted, but they can all be categorized into two main categories of systems: centralized and decentralized. These two control systems stem from the ideas of centralized and decentralized computing. Centralized computing consists of a single “master” subsystem, or brain, that contains most or all of the functions for the system tasks and many “slave” subsystems, or subordinates, through which access to the master subsystem is restricted. A common implementation of this is cloud rendering, a type of cloud computing. This structure allows a centralized control system to contain one algorithm to interpret senor data and adjust single or clusters of panels accordingly, as well as integrate with the other building control systems more seamlessly.

Decentralized computing, decentralizes the master subsystem throughout all other subsystems, resulting in a more even spread of software and hardware required for the tasks of the system. The result is an autonomy of subsystems and a redundancy for system tasks. A common implementation is the hardware in the facility for cloud computing, the practice of using remote servers to store, manage, or process large amounts of data. The facility can handle many different models at any given time, requiring the facility to house many equivalent rendering machines to spread out the workload across many machines, which also allows some machines to pick up the slack should another machine fail and need repairs. A decentralized control system would mimic this structure by housing all the computation for the system within each kinetic

panel, effectively localizing panel calculations to its own sensors and the output of its neighbors. This reduced computational load combined with the lack of any central information hub make this system ideal for quick responses, but also prevented any facade with this control system in its ability to become more than purely responsive.

1.3 Problem Statement

What does this mean for designers of kinetic facades? Currently, there is a disconnect between the design and operation of a kinetic facade. The design process utilizes a parametric modelling software, such as Grasshopper, to generate geometry and subsequently simulate the functionality of the facade. The operation is the design and implementation of the control system which mimics the designed functionality. Designers typically lack an understanding of the latter, which can lead to an underutilization of the capabilities of the control system. This in conjunction with the frequent inability to acquire a list of users that will occupy the building cause designers to largely ignore user preferences during the design process in favor of standardized optimizations for light levels, temperature settings, etc. The resulting building leaves users feeling the need to manipulate system settings to match their environment to their preferences, which could undermine or entirely counteract the intent of the kinetic facade. In order to better accommodate user input as well as external and internal changes a building faces throughout its lifetime, two things must occur: an increase in transparency of operation for designers and the control system of the facade gaining the ability to learn patterns between user preferences and environmental conditions.

Background

Finding a balancing point between several environmental aspects is difficult enough due to conflicting inputs, such as daylighting and incident solar radiation, and the adding of user preferences only further muddles the issue. The current approaches for optimizing a kinetic facade for its inputs have many weaknesses that the act of “learning” with an intelligent system could solve. This section will discuss previous solutions to portions of this issue and the specific weaknesses of the current approach.

2.1 Case Studies

The idea of implementing smart and intelligent systems in an architectural setting is by no means recent. For decades, researchers and designers alike have been developing ways with which to create ever increasing levels of optimal energy efficiency or incorporating user preferences into the functions of a building. But many of these fall short in scope for achieving anything other than relative user convenience or minor increases in efficiency that rely on a lack of user adjustments. The following previous works will provide an overview for how this field has evolved and its current state.

2.1.1 Adaptive House Project

In 1998, M. C. Mozer wrote a dissertation titled *The neural network house: an environment that adapts to its inhabitants*, which presented his ideas and experimental house, Adaptive House Project. The goal of the project was to create a house-wide system that could program itself, or learn, based on a variety of occupant activity and input. The actual house, located in Boulder, Colorado, was equipped with more than seventy five sensors that could record anything that either directly affected user comfort

or were evidence of user activity and desired settings, including ambient light, temperature, motion, door and window positions, and weather. The system used this information to control actuators that managed the heating systems, water heating, fans, lighting, speakers, and ventilation.

Mozer implemented two machine learning algorithms to create the intelligent behavior of the house. The first algorithm learned and predicted where the occupants were within the house based on motion sensor readings and a statistical analysis of environmental data for a given period of time. The second algorithm determined which actions needed to occur based on a calculation of discomfort costs and energy costs for a given predicted location of the occupants.

While comprehensive and logically applicable to the issue of balancing energy efficiency with user preferences in office buildings, the very nature of the context (a residential building) is not compatible with an office setting. Furthermore, much of the internal workings of the system were hidden from the user, preventing them from learning about the system and possibly even customizing it. Despite the system allowing for an override of any aspect of the home, the system used standardized values and calculations to determine what would be optimal with little room for more intricate settings provided by the user.

2.1.2 Changing Spaces / House n

Others, such as Stephen Intille and Kent Larson, were more interested in how to use a similar system to inform the user of which changes should occur through unobtrusive notifications. Their project, Changing Spaces / House n, was a full-scale

single-family home with an integrated and ubiquitous sensor architecture, as a model home for the future that would help its occupants learn how to control the environment on their own (Intille 2002). Using similar calculations to Mozer, the computer control systems would read sensor data and calculate which changes, if any, needed to be enacted to minimize discomfort and energy costs, and then alert the occupant of what should be done. The occupant could then decide whether to perform the action, and the system would update with the result to better build a preference model.

This novel approach loses its grounding when moved outside of the residential sector, where users are more preoccupied with and concerned about completing their tasks. places of employment also typically have large spaces with high heating and cooling demands, which take too long to alter for anything other than a predictive, automated system to handle in a similar fashion. The system itself, also, has the flaw of inscrutability. Just as with the Adaptive House Project, the recommended settings were defined by the system designers, and not manipulable by the user.

2.1.3 Media House Project

Further still is the idea that the building itself is something to be manipulated through an automated process while maintaining a level of transparency to the user. Media House Project, developed by directors Vicente Guallart, Enric Ruiz-Geli, Willy Müller, and a team of over 100 collaborators and consultants, envisioned the house as a computer, its structure as an information network. The integrated structure and infrastructure of the house allowed for a more temporal nature while simultaneously

achieving quick and easy access to any system and piece of information from anywhere in the house.

The key to achieving this dynamic nature was a system coined Internet Zero by its creator, Neil Gershenfield. Internet Zero relied on every connected device receiving an embedded computer for control and housing an IP address, allowing even a simple lightbulb to be located in both physical and digital spaces. The operating system of the house, budded DOMOS, effectively decentralized the use and control of the house by "[monitoring and controlling] both the physical elements that the house contains (in terms of comfort, media players, accesses, conditions, etc.) and the information that directly or indirectly affects the house (media, communication, files, Web sites, etc.)" (Guallart, 2004). Several screens throughout the house allowed for the system interface to recognize users and receive their input into the system. The functionality of the project is not easily translated into an office environment, due to the focus of the project being the media, physical and digital, that is consumed by the occupants. Unlike the limitations of previous works, however, the method for interfacing with the system could be altered and developed for a more seamless interaction in an office environment.

2.1.4 Personalized Intelligent Comfort Control

In order to reframe the above work into an office context, Andrew Payne developed a method and control system for sensing environmental conditions, receiving user input, and controlling an integrated network of personal desk devices and building systems. The goal of his dissertation, *Personalized Intelligent Comfort Control for Office Spaces*, was to “develop a novel localized building control strategy which can learn and

adapt over time to improve both user satisfaction and energy efficiency" (Payne, 2014). His work outlines the design of several sensors, personal desk devices, and controllers that are used in conjunction with a neural network to adapt a localized area of an office building to the user. The system was designed to be highly scrutable and manipulable by the user with very little system designer influence over the level of each form of comfort (light, thermal, audible, etc.). To achieve this, Payne developed a dashboard through which a schedule could be set, different device controls could be manipulated, and a survey for preferred relative comfort levels could be completed. These inputs coupled with this control method allowed the system to understand and build an optimal experience for the user without sacrificing energy efficiency.

2.2 Current Approach and its Weaknesses

To date, all research has been focused on a small user group for comfort control, even Payne's research due to localization of comfort zones. However, in order to achieve a more global comfort and environmental optimization for an office building, a more robust, intelligent system must be implemented. Currently, all calculations for facade optimization take place during the design phase in the form of a statistical analysis on weather data, which is in turn used in a multi-objective optimization (MOO) algorithm to determine optimal system actions for a variety of different instances of environmental inputs. This approach inherently limits the ability for the facade to interface with other systems (such as Payne's), prevents any scrutability of the system by the user, and only holds regard for the standardized optimizations for interior environment based on

guidelines such as ASHRAE. The following subsections cover how MOO and weather data are used and the inherent flaws related to those uses.

2.2.1 Multi-Objective Optimization

MOO is a process through which a set of optimal solutions are derived from a collection of input criteria, or objective vectors, such as daylighting and incident solar radiation. These objective vectors are then put through an analysis algorithm that tests many different possible inputs and maps the overall success of the system in relation to each objective vector. The results are then mapped to determine if and where there is a Pareto front. A Pareto front, or more commonly Pareto efficiency, refers to a state of allocation of resources such that it is impossible to reallocate resources to benefit one criterion without negatively affecting at least one other criterion. Pareto fronts can take on various forms depending on the tests or objective vectors, as shown in Figures 1a and 1b. The designer or curator of the MOO then selects from the solutions along the Pareto front the solution(s) for implementation.

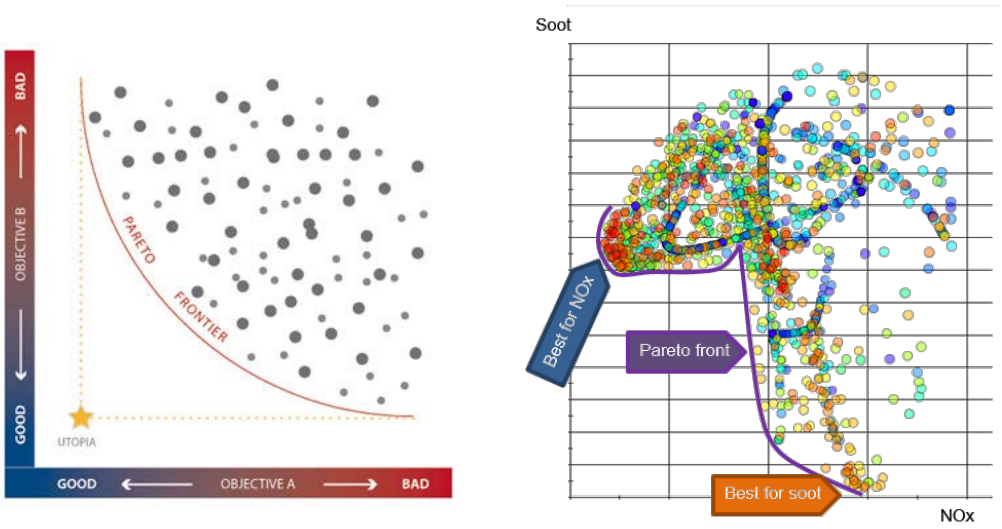

Figure 1a: (Left) Multi-Objective Optimzation.

Figure 1b: (Right) Pareto front for divergent crietria.

Examples of a mapped Multi-Objective Optimization result for two objective vectors.

Despite being a powerful way of optimizing between non-coincident objectives, there are some downsides to using MOO. The first and most prevalent is the time required to complete the analysis process. Due to the high volume of possible combinations of inputs for a facade with all the possible objectives being solved for, the solution space is vast. The other issue, which brings up an earlier point about the deficiency in designing for user preferences, is that MOO can only solve for defined objective vectors. Thus, designers who want to design for occupant preferences currently must rely on standardized values. This is an issue because, as noted from a study published by the Center for the Built Environment (CBE) which surveyed 52,980 occupants in 351 office buildings over the course of ten years, there is no universally acceptable personal comfort set point (Frontczak et al., 2012).

2.2.2 Weather Data

One of the inputs for MOO is weather data in the form of EnergyPlus Weather (EPW) or Typical Meteorological Year (TMY) files; most notably ambient light levels, cloud coverage, and sun position (angle and azimuth). While sun position is well documented for any given coordinate on Earth, all other aspects of weather are variable with imperfect collection methods. Weather stations and other in situ data collection sites are spread across the globe and measure most or all weather data, but the most precise this recording system can be is regional. For example, the state of Oklahoma's Mesonet, an array of environmental monitoring stations, consists of 121 stations, with at least one station in each of its 77 counties (See Figure 2). Across Oklahoma's 69,960 mi² area, that puts an average of ≈578 mi² per station. This density of data collection has difficulty capturing the location of anomalous weather patterns and micro-climates, and certainly fails to capture site specific environmental conditions. As a result, MOO is fed an estimation of what the site's weather should be rather than what the exact conditions are.

Figure 2: Oklahoma Mesonet. Map of in situ sites for the mesoscale network for weather data collection.

Methodology

My contribution is a system control architecture for controlling intelligent kinetic facades and integrating them with a building's HVAC system. The content of this section will outline the logic design I have developed to describe the system (visualized in Figure 3 on the following page). The three driving assumptions for unknown factors for this system are as follows:

1. User preference data (it is unknown during design and may change as occupants move to a different floor or leave the company)
2. Interior space volume (due to possible renovations)
3. Exterior environmental data (due to site specific microclimates)

Without this data, it is impossible to use MOO. Furthermore, allowing the system to learn the relationships between these difference inputs will allow for more accurate responses after a brief teaching period. Additionally, it should reduce analysis time during the design phase and prevent the need to design a brand new control system for each new facade. Here, I explain the details for how the included flowchart (Figure 3) that shows the logic for the system works.

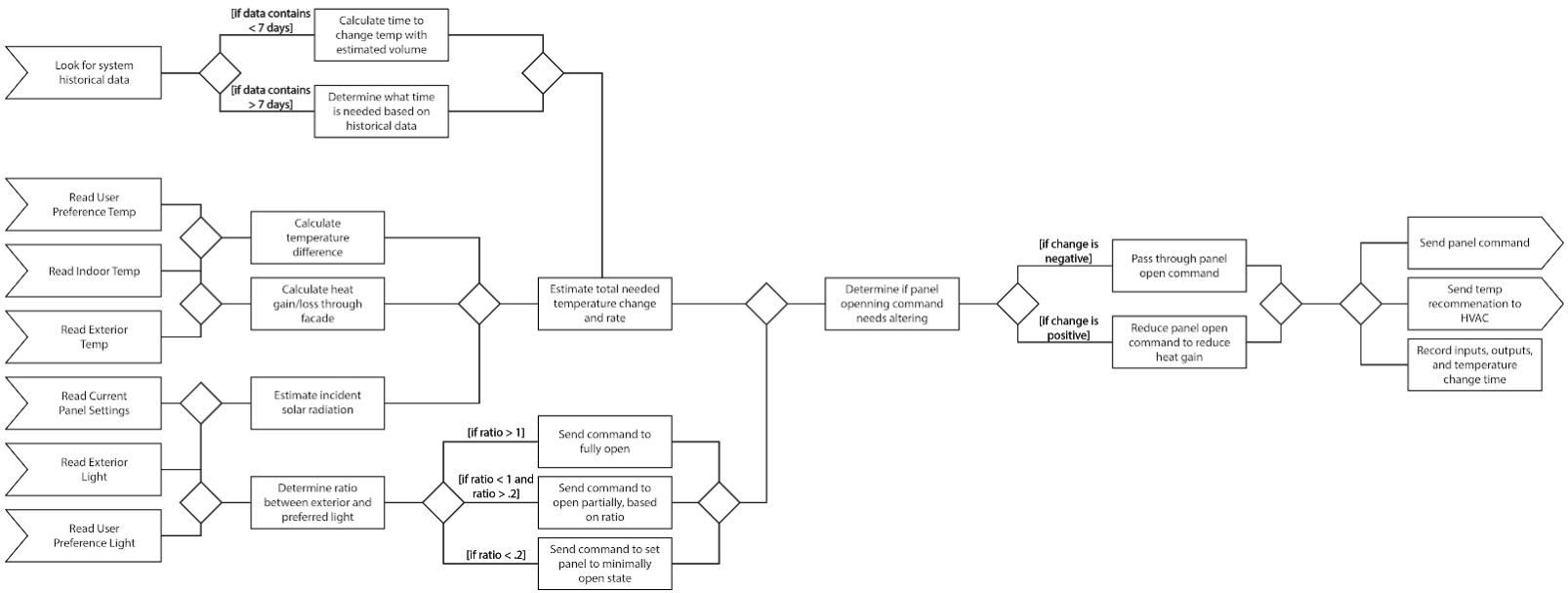

Figure 3: Control System Flow Chart. This flow chart outlines the logic for my control system.

3.1 Inputs

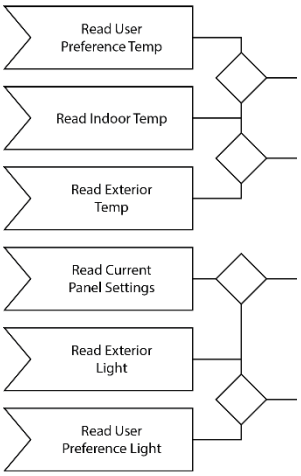

Figure 4: Flowchart Inputs

There are five inputs to the system:

1. oLight - light value (in lux) of outside at each facade
2. pLight - user preferred light value (in lux)
3. oTemp - outside temperature
4. iTemp - inside temperature
5. pTemp - user preferred temperature (weighted average)

These inputs are used two times in my system. The first is in the calculation of several intermediate steps of my control system, as follows:

1. Calculate the temperature change
    a. Rough approximation of the conversion of light energy into a unit of heat
    b. Difference between current and desired temperatures
    c. Approximated thermal gain/loss through convection
2. Determine difference between interior and exterior lux values
3. Determine panel position for lighting needs

4. Alter panel position to increase or decrease incident solar radiation
5. Output recommended temperature value for HVAC system

User preference towards lighting or thermal comfort will not need to be explicitly acquired due to the collection method of inputs, as higher preferred lux values will result in a skew towards lighting comfort and lower preferred lux values will result in a skew towards thermal comfort. For example, if the user preference for light equates to 400 lux and the panels need to be fully opened to allow that light value in, but the temperature gain from the facade dictates a 20% reduction of that openness (process for this explained in detail in section 3.4), then the incoming daylighting would be similarly reduced. If the user wants to create a preference for daylighting in this situation, then their feedback to the system of "prefer brighter" might increase their preference to 500 lux for those same environmental conditions, which would translate into 400 lux for that same reduction.

Regarding the collection of the system inputs, there are two primary methods: sensor arrays and user polls. The sensor arrays will consist of light and temperature sensors on the facade of the building and acquire the interior temperature either from interior sensors or the localized thermostat within the space. User preference data will be collected through a polling system, the design of which is outside the scope of this thesis, but whose output would provide explicit temperature and lux values for user preferences at time intervals throughout the day. Presumably, this polling system will ask the user for their explicit temperature preference and relative lighting preference (brighter/dimmer). The lighting preference collection method then uses interior lighting sensors to determine

a relative increased or decreased value of the space's current lux value. This, along with the explicit temperature value, are passed to the facade control system.

3.2 Neural Network

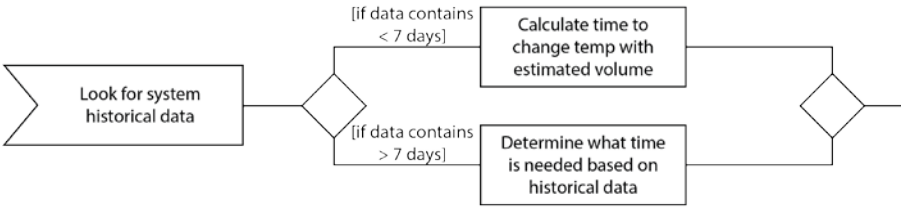

Figure 5: Flowchart Neural Network

The second time these inputs are used is in the refining of the historical data for the neural network I developed for testing. The historical data consists of all inputs paired with the recorded temperature change time, panel position, and recommended temperature to the HVAC system.

Neural networks (NN), or more commonly artificial neural networks, are a form of machine learning that excels in pattern recognition and solving prediction problems where the inputs are too numerous for the programmer or their relationship is not able to be expressed in a linear or structural equation. Due to their pattern recognition capabilities, NNs are frequently in real-time data interpretation, such as image recognition (Figure 6). Collected data and its outputs are then added to the dataset to refine output. This is possible through the use of backpropagation, or the backward propagation of errors, an algorithm for supervised learning using a gradient descent, or an iterative optimization algorithm for finding the minimum of a function. Here, supervised learning refers to the task of teaching the system a function through a series of example input/output pairs, or in my case user and environmental data with resulting outputs.

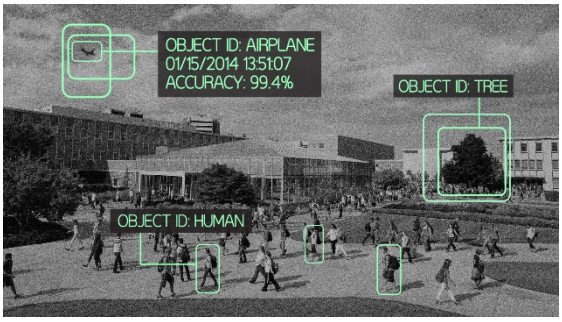

Figure 6: Image Recognition

Thus, backpropagation takes an example dataset and generates weights inside of hidden layers that act as the decision-making neurons of the system. In short, "artificial neural networks provide a computational framework for incorporating individual feedback such that the objectives of the system can change over time – making it ideally suited for the current task" (Payne, 2014).

If a large enough buffer of historical data exists (such as more than 7 days, as show in the flowchart), the NN then uses this collected data paired with the resulting outputs to form the example dataset to create a model of use for the building. Once the NN is trained from these input/output pairs, the system takes inputs from sensors and historical information for user preferences at given time intervals and runs them through the NN to determine what to output (temp. change time, panel position, and recommended temp) and when to start (desired time for result – temp. change time). Determining the amount of time needed for a particular building to have sufficient historical data requires some minor testing, though the range of seven to twenty-one days appears to be a sufficient minimum requirement based on testing a series of different spaces. Of course more example data will only refine the initial function, but anything beyond twenty-one days did not noticeably impact the amount of iterations needed to

train the neural network. Should the historical data not exceed the required amount of time, however, the system goes through the additional calculations mentioned earlier, as outlined in the following sections.

3.3 Temperature Calculation

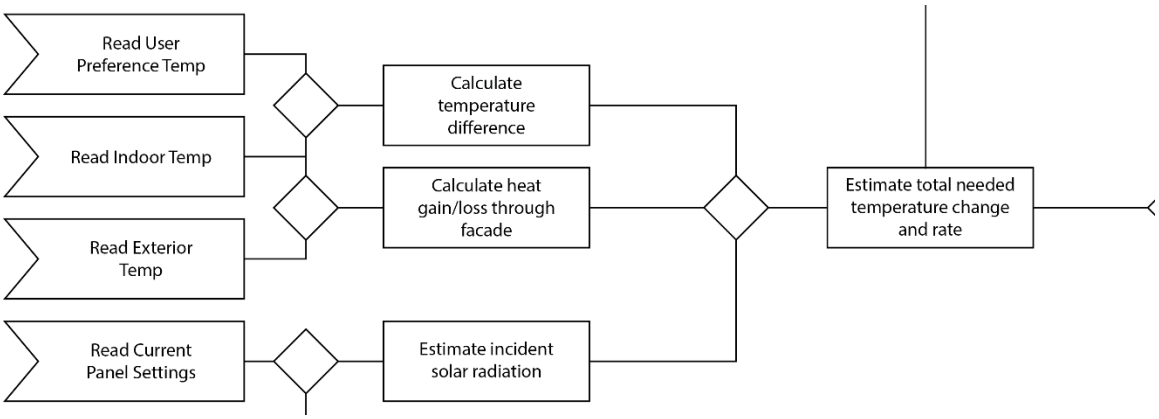

Figure 7: Flowchart Temperature Calculation

First, we will look at how temperature values are utilized. The primary components of temperature for a facade are incident solar radiation (sGain) and thermal conduction (cGain), which are based on two separate calculations. Incident solar radiation refers the heat component of light energy that enters the building, while thermal conduction refers to the property of all materials (in this case, the facade) to transmit heat energy through direct contact. For the purposes of this thesis, I used more general and simplified calculations for these values as a proof of concept. Each of these calculations is heavily reliant on the specifications, primarily the U-value (thermal transmittance property, an inverse of the R-value, or insulation factor of a material) and light transmittance (the ability of the glass to transmit all aspects of light energy into the interior space), of the glass facade being used in the project, and should be refined accordingly upon selection of that component of the project. As it stands, the calculation

for thermal conductivity is as follows, using the U-value of a higher rating curtain wall system:

$$cGain = (iTemp - oTemp) * Uvalue\ factor$$

This provides a rough estimate for the amount of conductive heat gain or loss through the facade. Because this calculation informs the amount needed to heat or cool the space, the positivity of cGain is inverse to what would be expected. That is, a positive cGain value indicates heat loss through the facade, which translates into a heating requirement from the sGain and HVAC, and vice versa for a negative value.

sGain must also be refined upon the selection of the curtain wall system and interior materials that would translate light energy into heat. Currently, sGain is simply a factor of the light energy received by the facade translated into a temperature value. Here, the amount of light is highly variable, as it depends on exterior light values and the openness of the facade panels. oLight, read from the exterior light sensors, is factored by the current panel settings (oPanel), a system variable that is read at this point.

$$sGain = oLight * oPanel * energy\ conversion\ factor$$

The resulting value, a consistently positive value since incident solar radiation always adds heat, is added to cGain to determine the total heating or cooling requirement from external loads (tGain).

$$tGain = cGain + sGain$$

Should tGain be a negative value, indicating a cooling requirement, the panel openness (calculation explained more in section 3.4) may be reduced by up to 50%, depending on the value of tGain.

Finally, to finish understanding the heating/cooling needs of the space, the difference between current interior temperature and the next user preferred temperature (dTemp) is taken, where:

$$dTemp = pTemp - iTemp$$

The output of this calculation closely resembles that of cGain, in regards the positivity of the values. The value of dTemp is then used to determine dTime: the time required to heat the space (based on the calculation for a thermally independent system). This calculation requires the volume of the space. Since the volume is assumed to be unknown and thus not collected after the training period, a simple approximation of the space's initial volume is needed for this calculation.

$$dTime = \frac{volume\ of\ space}{tGain}$$

Once an initial value for dTime is determined, it is then used to determine how much thermal gain/loss is acquired at the rate of tGain during dTime, which will be stored in t'Gain.

$$t'Gain = tGain * dTime$$

This value is then used to in the same calculation used to acquire dTime, with the output of d'Time.

$$d'Time = \frac{volume\ of\ space}{t'Gain}$$

The value of d'Time is added to dTime to get an estimated overall time (ΔTime) needed for temperature to change for a given space behind the facade.

$$\Delta Time = dTime + d'Time$$

This value, ΔTime, is used to calculate when the panels should move and trigger the HVAC system to match the interior temperature with the pTemp for a given hourly interval. During the process of calculating dTime, the total temperature difference needed to be addressed by the HVAC will be summed and passed to that system.

3.4 Lighting Calculation and Panel Adjustment

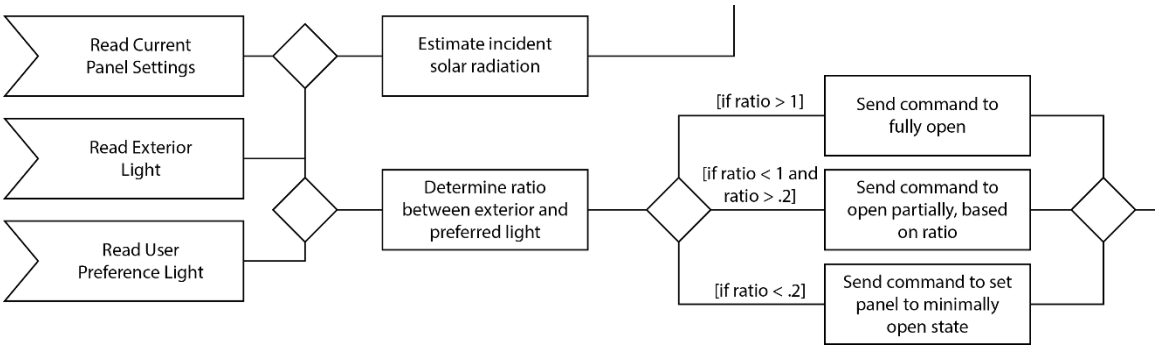

Figure 8: Flowchart Lighting Calculation

To determine the next facade panel position needed for lighting and thermal gain reduction, a simpler process is followed. First, the ratio between exterior and preferred interior light levels must be found. This value, we will call dLight, is the ratio found by dividing pLight (preferred light level) by oLight (outside light level):

$$dLight = \frac{pLight}{oLight}$$

The logic here is that if oLight ≤ pLight, then the maximum amount of natural light must be let into the space through the facade (at a maximum ratio of 1, or full openness), while if oLight > pLight, then the amount of transmitted light must be reduced to match user preferences and attempt to prevent glare. This ratio, dLight, is applied as the value for the panels' percentage of openness. As mentioned previously in regards to cGain, however, it may be necessary to reduce the amount of incoming light in order to minimize heat gain

at the facade. For this, any positive value of cGain that would result in a thermal gain of up to 5°F during a 30 minute time period is remapped to percentage values from 100% - 50% in an inverse relationship, resulting in a factor value we will call cLight. Multiplying dLight by cLight will result in the value d'Light, which will be considered the final percentage of openness for the facade panels.

$$d'Light = dLight * cLight$$

For example, if cGain ≤ 0°F, then cLight would map to a ratio of 1, which would result in no change to the original dLight calculation. However, should cGain = 3°F, cLight would remap to .7, which would reduce the original dLight value by 30%.

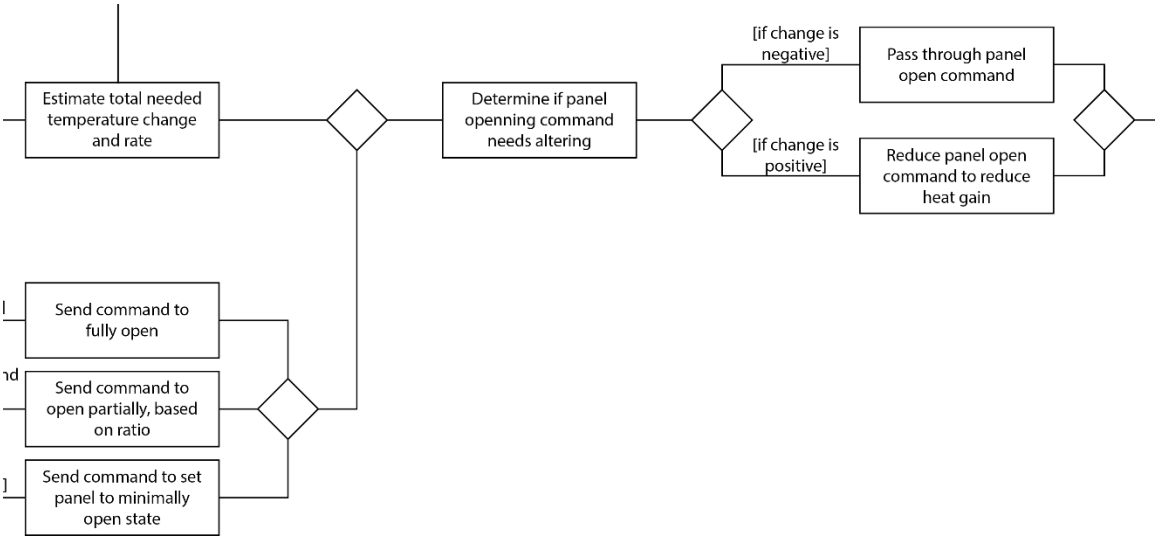

Figure 9: Flowchart Panel Adjustment

This method of calculating panel openness innately gives a preference towards thermal performance. However, due to the method of input of user preferences for light level preferences, users can train the system to provide more open panels by providing the feedback of "brighter" or "more light" to the system. This process will increase the

initial value of dLight for the above calculation, which will result in a higher d'Light value, thus given preference to daylighting over thermal performance.

In summation, these calculations are used to generate the outputs for this system, as follows:

1. Panel open/close command
2. Recommended HVAC temperature setting

In addition to these outputs, the system records all input values, the above outputs, and the actual time taken to adjust temperature. Upon completion of the initial recording phase of the system, where these calculations are completed at regular intervals throughout the day, the system converts to using a neural network to determine the likely user preferred settings for a given hourly interval and how long it will take the system to achieve all aspects of those preferences. Upon completion of that time interval's transition, the system records all previously mentioned inputs, including any user preferences provided during the time interval. After each additional seven days beyond the initial recording phase, the NN is retrained in order to provide more accurate system settings as the following various items change:

1. User groups, indicating some change in the personnel on a given floor
2. Temperature change time, indicating a significant change in interior space, due to renovation, splitting or merging of spaces on a floor, etc.
3. Changes in the exterior environment, indicating a change in season (pertinent for the first full year after system deployment) or urban landscape, such as a new building altering solar patterns for the building

3.5 Testing

To test this system for effectiveness in learning the relationships between the three unknown factors and accurately predicting when changes need to occur, I devised three tests for my script. To aid in these tests and simulate a functioning system, I developed a simple NN, included in Appendix A. This NN takes a historical data set (simulated) and live input data (simulated) to determine the system outputs. Test 1 assumed user input and space remain unchanged to ensure the system could properly record information and predict start time for the adjustment period. Test 2 assumed user input changed while space remained unchanged to simulate a personnel change. This ensured any change in user input would affect system outputs. Test 3 assumed user input remained unchanged while space changed by altering simulated time recordings for temperature change to simulate a remodel/renovation of the space. These tests ensured the system would alter its adjustment start time appropriately within three days. These tests utilized artificial user preference data (See figures 4a and 4b) and historical weather data to simulate live sensor readings. User preference data, regardless of collection method (assumed explicit temperature values and relative light values), must output to my system explicit temperature and light values in Fahrenheit and lux respectively.

Results, Conclusions, and Recommendations

While overall successful, it should be noted that the system, due to lack of a large set of data, can be slightly inaccurate in the beginning, depending greatly on how variant the weather at the site can be. As it stands, a minimum of two days out of each month in a 12 month cycle are required in order to generate an accurate enough preliminary teaching model for the system to understand how it should react to environmental data, with a seven to twenty-one day period of data gathering once deployed.

While not as effective in providing an accurate weather model as the current Multi-Objective Optimization process in the initial stages of the building's life, the continual gathering of input quickly outpaces MOO in generating site-specific readings and outputs. However, as previously noted, this is largely site dependent. There are many cities in which buildings with kinetic facades could be constructed that are standard enough in their climate and weather patterns that existing weather data may be all that is needed to generate an accurate model of a given site. At this point, it comes down to a cost-effectiveness analysis to make a decision between an accurate, stagnant model and an adaptive, learning model for weather analysis. Here, it becomes clear to me that the ability to include user input for the learning model gives a machine learning based system a clear edge, not only in its ability to create a site specific weather model and correlated facade response, but also in its ability to reduce the occupant's need to utilize HVAC and artificial lighting to alter the internal environment of the building. This, then, brings up the question: what do we do with this system once it's operational? For this, I have two recommendations: user satisficing and shareability of data.

For the former of these, a larger study should be conducted with this system to determine how effective user input would be in affecting the user's satisfaction with their environment. Here, there are many different variables to consider. First and foremost are the floor plan, use type of the building, and, even more specifically, the type of business conducted. An easy assumption is that it is a simple office tower, but different business lines may have different preferences for the workspace environment. While computers are almost inescapable in this day and age, the tasks done on them and the amount of time spent at a computer in relation to other tasks could vary wildly. Software developers will spend most of their time at a computer staring at text and will likely prefer a darker background and style on their screen to reduce eye strain. As a result, brighter lights will produce more glare on their screens than, perhaps, an editorial business where most text editing programs primarily have white backgrounds and generate brighter screens. Studying the different types of users, their preferences, and my proposed system's ability to meet and satisfice them is an important step in determining if there are any buildings that should not utilize this system, and to what level my system can satisfice users of buildings that should.

The latter of my recommendations is only possible after my system has been implemented once in a given region. Neighboring buildings could then utilize collected data from the initial building to reduce the teaching period. The designers would determine to what level they want to utilize existing pairing data from surrounding buildings, as some amount of conversion would need to take place to accommodate any difference between the panel designs. This could even be taken a step further by allowing

the two systems to communicate to each other. Take sites A and B, 4 standard city blocks apart. Perhaps site A is neighbored by several other high rises, while site B only has one neighboring mid-rise tower. Readings from sensors in the building on site A could be skewed in some way by the surrounding built environment that affects site B by a lesser amount. While this should be taken into account, by allowing inter-building communication between the two systems, any reading that acts as a statistical outlier can be more easily identified and resolved than if the system were to be isolated.

References

Abdelmohsen, S., & Massoud, P. (2015). Integrating Responsive and Kinetic Systems in the Design Studio: A Pedagogical Framework. *Proceedings of the 33rd eCAADe Conference,2*, 71-80. Retrieved August 26, 2017.

Abdelmohsen, S., Massoud, P., & Elshafei, A. (2016). Using Tensegrity and Folding to Generate Soft Responsive Architectural Skins. *Smart and Responsive Design,1*. Retrieved August 26, 2017.

Askarinejad, A., & Chaaraoui, R. (2015). Spatial Nets: the Computational and Material Study of Reticular Geometries. *ACADIA 2105: Computational Ecologies: Design in the Anthropocene [Proceedings of the 35th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA) ISBN 978-0-692-53726-8]*,123-135. Retrieved August 26, 2017.

Baquero, P., Montas, N., & Giannopoulou, E. (2016). Transformational Intelligent Systems - Parametric Simulation Workshop. *Complexity & Simplicity - Proceedings of the 34th eCAADe Conference,1*, 73-76. Retrieved August 26, 2017.

Beesley, P., Ilgun, Z. A., Kadish, D., Prosser, J., Gorbet, R., Kulić, D., Zwierzycki, M. (2016). Hybrid Sentient Canopy: An implementation and visualization of proprioreceptive curiosity-based machine learning. *ACADIA // 2016: POSTHUMAN FRONTIERS: Data, Designers, and Cognitive Machines [Proceedings of the 36th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA) ISBN 978-0-692-77095-5]*,362-371. Retrieved August 26, 2017.

Biloria, N., & Sumini, V. (2009). Performative Building Skin Systems: A Morphogenomic Approach Towards Developing Real-Time Adaptive Building Skin Systems. *International Journal of Architectural Computing,4*(3), 643-676. Retrieved August 26, 2017.

Borhani, A., & Kalantar, N. (2016). Material Active Geometry - Constituting Programmable Materials for Responsive Building Skins. *Complexity & Simplicity - Proceedings of the 34th eCAADe Conference,1*, 639-648. Retrieved August 26, 2017.

Campbell, C. (2009). Compu-Kinetic Mediapod. *Building a Better Tomorrow [Proceedings of the 29th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA) ISBN 978-0-9842705-0-7]*,281-283. Retrieved August 26, 2017.

Campbell, C. (2012). Shading With Folded Surfaces: Designing With Material, Visual and Digital Considerations.*Digital Physicality - Proceedings of the 30th eCAADe Conference,2*, 613-620. Retrieved August 26, 2017.

Cheverst, K., H. E. Byun, D. Fitton, C. Sas, C. Kray, and N. Villar. 2005. Exploring issues of user model transparency and proactive behaviour in an office environment control system. User modeling and user-adapted interaction. 15 (3): 235-73.

Crawford, S. (2010). A Breathing Building Skin. *ACADIA 10: LIFE in:formation, On Responsive Information and Variations in Architecture [Proceedings of the 30th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA) ISBN 978-1-4507-3471-4]* ,211-217. Retrieved August 26, 2017.

Guallart, V., and Institut dArquitectura Avançada de Catalunya. 2004. Media house project: the house is the computer, the structure is the network. Institut dArquitectura Avançada de Catalunya.

GÜN, O. Y., & GREENBLATT, E. E. (2013). TRAN[S]QUILLITY: THE DYNAMICALLY MEDIATED FAÇADE. *Open Systems: Proceedings of the 18th International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA)*,955-964. Retrieved August 26, 2017.

Intille, S. S. 2002. Designing a home of the future. Pervasive computing, IEEE 1 (2): 76-82.

Klemmt, C., & Sodhi, R. (2017). DOUBLE-CURVED FORM APPROXIMATION WITH IDENTICAL DISCRETE PANEL GEOMETRIES. *Protocols, Flows and Glitches, Proceedings of the 22nd International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA)*,457-467. Retrieved August 26, 2017.

Krietemeyer, E. A., & Dyson, A. H. (2011). Electropolymeric Technology for Dynamic Building Envelopes. *ACADIA Regional 2011: Parametricism*,75-83. Retrieved August 26, 2017.

Mallasi, Z. (2016). INTEGRATING PHYSICAL AND DIGITAL PROTOTYPES USING PARAMETRIC BIM IN THE PURSUIT OF KINETIC FAÇADE. *ASCAAD*. Retrieved August 26, 2017.

McGonagle, John, et al. “Backpropagation.” Brilliant Math & Science Wiki, brilliant.org/wiki/backpropagation/. Accessed April 3, 2018.

Menegotto, J. L. (2012). Fachada Cinética: aplicando aritmética modular para controlar padrões de movimento. *SIGraDi*. Retrieved August 26, 2017.

Moloney, J. (2009). A morphology of pattern for kinetic facades. *CAAD Futures*. Retrieved August 26, 2017.

Mozer, M. C. 1998. The neural network house: an environment that adapts to its inhabitants. Paper presented at Proceedings of AAAI Spring Symposium on Intelligent Environments, Menlo Park, CA 1998, AAAI Press.

S., Ting, Y., & Lu, P. W. (2016). Development of Kinetic Façade Units with Bim-Based Active Control System for the Adaptive Building Energy Performance Service. *CAADRIA*. Retrieved August 26, 2017.

Sharaidin, K., & Samil, F. (2012). Design Consideratiosn for Adopting Kinetic Facades in Building Practice. *eCAADe*. Retrieved August 26, 2017.

Sheikh, M. E., & Gerber, D. J. (n.d.). Building Skin Intelligence: a parametric and algorithmic tool for daylighting performance design integration. *Acadia 2011 proceedings - integration through computation*,170-177. Retrieved August 26, 2017.

Appendix A: Neural Network

```
import numpy as np

X = np.array(([800 , 415, 75, 72.6, 75.0],
              [1000, 430, 76, 71.6, 72.6],
              [1200, 445, 77, 71.1, 71.6],
              [1600, 453, 78, 70.8, 71.1],
              [2000, 458, 79, 70.3, 70.8],
              [2400, 460, 79, 70.1, 70.3],
              [1800, 456, 78, 70.8, 70.1],
              [1500, 447, 76, 70.4, 70.8],
              [1200, 440, 74, 70.9, 70.4],
              [1000, 433, 73, 71.1, 70.9]),
              dtype=float)
y = np.array(([56], [43], [22], [14], [21], [7], [27], [19], [18], [10]), dtype=float)


xPredicted = np.array(([1000, 420, 77, 72, 72.6]), dtype=float)

X = X/np.amax(X, axis=0)
xPredicted = xPredicted/np.amax(xPredicted, axis=0)
y = y/100

class Neural_Network(object):
  def __init__(self):
    self.inputSize = 5
    self.outputSize = 1
    self.hiddenSize = 3

    self.W1 = np.random.randn(self.inputSize, self.hiddenSize)
    self.W2 = np.random.randn(self.hiddenSize, self.outputSize)

  def forward(self, X):
    self.z = np.dot(X, self.W1)
    self.z2 = self.sigmoid(self.z)
    self.z3 = np.dot(self.z2, self.W2)
    o = self.sigmoid(self.z3)
    return o

  def sigmoid(self, s):
    return 1/(1+np.exp(-s))

  def sigmoidPrime(self, s):
```

```
    return s * (1 - s)

  def backward(self, X, y, o):
    self.o_error = y - o
    self.o_delta = self.o_error*self.sigmoidPrime(o)

    self.z2_error = self.o_delta.dot(self.W2.T)
    self.z2_delta = self.z2_error*self.sigmoidPrime(self.z2)

    self.W1 += X.T.dot(self.z2_delta)
    self.W2 += self.z2.T.dot(self.o_delta)

  def train(self, X, y):
    o = self.forward(X)
    self.backward(X, y, o)

  def saveWeights(self):
    np.savetxt("w1.txt", self.W1, fmt="%s")
    np.savetxt("w2.txt", self.W2, fmt="%s")

  def predict(self):
    print "Predicted data based on trained weights: ";
    print "Input (scaled): \n" + str(xPredicted);
    print "Output: \n" + str(self.forward(xPredicted));

NN = Neural_Network()
for i in xrange(5000):
  print " #" + str(i) + "\n"
  print "Input (scaled): \n" + str(X)
  print "Actual Output: \n" + str(y)
  print "Predicted Output: \n" + str(NN.forward(X))
  print "Loss: \n" + str(np.mean(np.square(y - NN.forward(X))))
  print "\n"
  NN.train(X, y)

NN.saveWeights()
NN.predict()
```

Appendix B: Grasshopper Script